UNITED STATES NAVAL OBSERVATORY

CIRCULAR NO. 180

U.S. Naval Observatory, Washington, D. C. 20392 December 31, 2009

User's Guide to NOVAS 3.0

by

George H. Kaplan

John A. Bangert

Jennifer L. Bartlett

Wendy K. Puatua

Alice K. B. Monet

Table of Contents

Preface

The **N**aval **O**bservatory **V**ector **A**strometry **S**oftware (NOVAS) is an integrated package of subroutines and functions for computing various commonly needed quantities in positional astronomy. The package can supply, in one or two subroutine or function calls, the instantaneous coordinates of any star or planet in a variety of coordinate systems. At a lower level, NOVAS also provides astrometric utility transformations, such as those for precession, nutation, aberration, parallax, and gravitational deflection of light. The computations are accurate to better than one milliarcsecond. The NOVAS library is an easy-to-use facility that can be incorporated into data reduction programs, telescope control systems, and simulations. The U.S. parts of *The Astronomical Almanac* are prepared using NOVAS.

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 10 megaparsecs (2.06×10^{12} AU).

Compared with NOVAS 2.0 of 1998, this version improves the accuracy of star and planet position calculations (apparent places) by including several small effects not previously implemented in the code. New convenience subroutines and functions have also been added. NOVAS now fully implements recent International Astronomical Union (IAU) resolutions in positional astronomy, including new reference system definitions and updated models for precession and nutation. Kaplan et al. (1989, Astron. J., 97, 1197) describe the overall computational strategy used by NOVAS, although many of the individual algorithms described have been improved. *The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models* (Kaplan 2005; USNO Circular 179[1]) describes the IAU recommendations that underpin much of NOVAS 3.0, and is the basic reference for NOVAS algorithms relating to time, Earth orientation, and the transformations between various astronomical reference systems.

NOVAS 3.0 is available in two editions: Fortran and C. In identical full-accuracy calculations using the same data sources, differences in the results from the two editions should not exceed 6×10^{-8} arcseconds (3×10^{-13} radians) for solar system bodies and 7×10^{-10} arcseconds (3×10^{-15} radians) for stars. Both libraries may be downloaded from the U.S. Naval Observatory (USNO) website.[2]

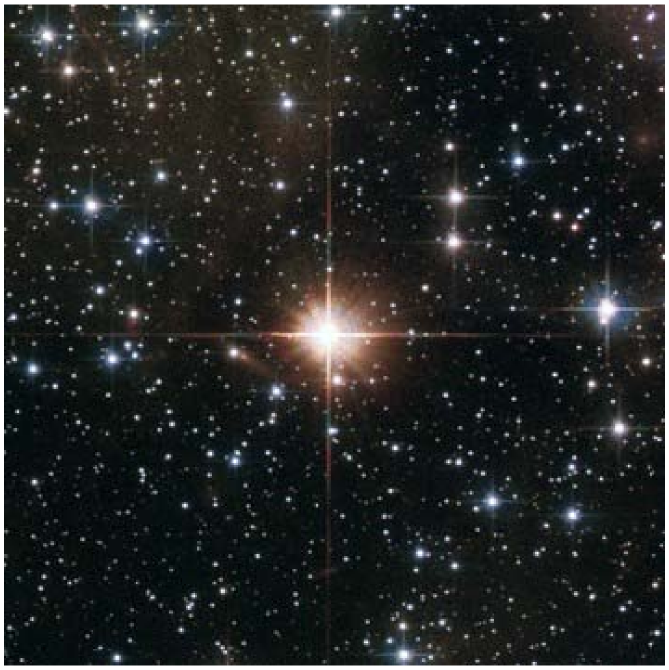
This circular consists of the user's guides to both NOVAS editions. An electronic copy of this circular and each individual part is available through the USNO website.[3] Errata and updates will be provided there as necessary. Users may also wish to consult *Nutation Series Evaluation in NOVAS 3.0* (Kaplan 2009; USNO Circular 181[4]).

[1] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179
[2] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/novas
[3] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-180
[4] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-181

User's Guide
to NOVAS Version F3.0

Naval Observatory Vector Astrometry Software

Fortran Edition



George Kaplan
Jennifer Bartlett
Alice Monet
John Bangert
Wendy Puatua

Part I of USNO Circular 180

U.S. Naval Observatory
December 2009

In this document, hyperlinks are in blue.

Table of Contents

Introduction

The Naval Observatory Vector Astrometry Software (NOVAS) is a source-code library for computing various commonly needed quantities in positional astronomy. The code exists in two languages, Fortran and C. The Fortran version goes back to the late 1970s, but has been updated periodically to use new, more accurate models that represent the evolving standards of the international astronomical and geodetic communities. The C version was first introduced in 1996 and provides the same functionality and accuracy. In full-accuracy calculations using the same data sources, differences in the results from Fortran and C editions of NOVAS, for the same calculation, should not exceed 6×10^{-8} arcseconds (3×10^{-13} radians) for solar system bodies and 7×10^{-10} arcseconds (3×10^{-15} radians) for stars.

NOVAS can provide, in one or two subroutine or function calls, the instantaneous celestial position of any star or planet in a variety of coordinate systems. NOVAS also provides access to all of the "building blocks" that go into such computations—single-purpose subroutines for common astrometric algorithms, such as those for precession, nutation, aberration, parallax, etc. NOVAS calculations are accurate at the sub-milliarcsecond level. The NOVAS package is an easy-to-use facility that can be incorporated into data reduction programs, telescope control systems, and simulations. NOVAS is used in the production of the U.S. parts of *The Astronomical Almanac*.

The algorithms used by the NOVAS routines are based on a vector and matrix formulation that is rigorous and consistent with recent recommendations by the International Astronomical Union (IAU). Objects within and outside the solar system are treated similarly and the position vectors formed and operated on by the NOVAS routines are defined within either the Barycentric Celestial Reference System (BCRS) or the Geocentric Celestial Reference System (GCRS), as appropriate. Both of these systems are described in IAU resolutions passed in 2000. GCRS quantities are converted to more familiar coordinate systems, such as the equator and equinox of date, by applying standard rotations.

Three levels of subroutines or functions are involved: basic, utility, and supervisory. Basic-level subroutines supply the values of fundamental variables, such as the nutation angles and the heliocentric positions of solar system bodies, for specific epochs. Utility-level subroutines perform computations corresponding to individual physical effects or transformations (aberration, light-bending, precession, polar motion, etc.). Supervisory-level subroutines call the basic and utility subroutines in the proper order to compute the apparent coordinates of stars or solar system bodies for specific dates and times. If desired, users can interact exclusively with the supervisory-level routines and not become concerned with the details of the geometry or physical models involved in the computation.

Previous versions of NOVAS were based on the algorithms described in Kaplan, et al. (1989) Astron. J., 97, 1197.[1] Although the phenomena that are considered and the overall sequence of calculations remains much the same, many of the models have been improved substantially over the last two decades in response to increased accuracy in observing techniques. Specifically, version 3.0 of NOVAS, released in 2009 and described in this document, implements the resolutions on astronomical reference systems and Earth rotation models passed at the IAU General Assemblies in 1997, 2000, and 2006. An explanation of the recent IAU resolutions can be found in Kaplan (2005), USNO Circular 179,[2] *The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models: Explanation and Implementation*, which contains much more information on topics only briefly touched on in this document. This version of NOVAS also improves the accuracy of its star and planet position calculations by including several small effects not previously implemented in the code; see, for example, Klioner (2003) Astron. J., 125, 1580. Several new convenience functions have also been added.

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.

[1] Parts were reprinted in Chapter 3 of Seidelmann (1992), *Explanatory Supplement to the Astronomical Almanac* (University Science Books)
[2] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179
[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

Abbreviations and Symbols Frequently Used

ΔAT	TAI – UTC (an integral number of seconds)
ΔT	TT – UT1
θ	Earth Rotation Angle (also ERA)
A&A	*Astronomy and Astrophysics* (journal)
AA	Astronomical Applications Department of USNO
Astron. J.	*The Astronomical Journal*
AU	astronomical unit
BCRS	Barycentric Celestial Reference System
Bull. AAS	*Bulletin* of the American Astronomical Society
CIO	Celestial Intermediate Origin
CIP	Celestial Intermediate Pole
dec	declination
EO	Earth Orientation Department of USNO
ERA	Earth Rotation Angle (also θ)
GCRS	Geocentric Celestial Reference System
IAU	International Astronomical Union
ICRS	International Celestial Reference System
IERS	International Earth Rotation and Reference Systems Service
ITRS	International Terrestrial Reference System
J2000.0	The epoch 2000 January 1, 12h TT (JD 2451545.0 TT) at the geocenter ("J2000.0 system" is shorthand for the celestial reference system defined by the mean equator and equinox of J2000.0)
JD	Julian date
JPL	Jet Propulsion Laboratory
NOVAS	Naval Observatory Vector Astrometry Software
RA	right ascension
SOFA	Standards of Fundamental Astronomy (software)
TAI	International Atomic Time (TAI = UTC + ΔAT)
TDB	Barycentric Dynamical Time
TIO	Terrestrial Intermediate Origin
TT	Terrestrial Time (TT = TAI + 32.184 s)
USNO	U.S. Naval Observatory
UT1	Universal time that is a measure of the Earth's rotational angle (UT1 = TT – ΔT)
UTC	Coordinated Universal Time, an atomic time scale that is the basis for worldwide civil time (replaces Greenwich Mean Time)
WGS-84	World Geodetic System 1984
x_p, y_p	Polar motion components; coordinates of the CIP with respect to the ITRS

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. (1979) 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]

A very cursory overview of some of the most important aspects of the astronomical calculations performed by NOVAS follows. In this chapter, names of subroutines relevant to the subject being discussed are shown in [BRACKETS]. Special terms referred to in IAU resolutions are printed in **bold** when first mentioned; other relevant terms with widely accepted meanings in astronomy are initially printed in *italics*.

1.1 Astronomical Coordinate Systems

Astronomical coordinate systems have traditionally been based on the extension of the Earth's equatorial plane to infinity, along with a fiducial direction in that plane, the *equinox*. The direction of the equinox is along the line of nodes where the equatorial and ecliptic planes meet. Because the Earth's equator and ecliptic are both in motion (the equator's motion is described by precession and nutation; the ecliptic's motion is due to the Earth's orbital variations), an infinite number of such coordinate systems exist, each one corresponding to the orientation of the two planes at a specific date and time. The situation is complicated by the fact that for some purposes in the past it was convenient to consider only precession—and to neglect nutation—in defining celestial coordinate systems, so that we can have, for any given date and time, both a *mean* system (in which only precession is considered) and a *true* system (in which both precession and nutation are taken into account). Thus we have the mean system of [some] date, the true system of date, the mean system of 2000.0, etc. [PRECES, NUTATE, GCRSEQ]

[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

Sidereal time is closely tied to the equatorial coordinate systems: one day of sidereal time is marked by successive transits of the equinox across a specific geographic meridian, and local sidereal time is just the apparent right ascension of stars transiting the observer's meridian. Like the equatorial coordinate systems, sidereal time comes in two flavors, *mean* and *apparent*, based on, respectively, the mean and true equinox and the system of right ascension that each defines. [SIDTIM]

In the last few decades of the 20th century, several IAU working groups led a general re-examination of astronomical coordinate systems at a very basic level. Part of the motivation was quite practical: most large ground-based telescopes were no longer built on equatorial mounts, and the equatorial coordinate systems were irrelevant to increasingly important space observations. Furthermore, the most precise fundamental measurements—those that by their nature are most closely related to the equatorial systems—are now obtained from Very Long Baseline Interferometry (VLBI), a radio technique that has no direct sensitivity to the equinox. Another important consideration was the need for astronomical coordinate systems that were part of a general relativistic framework. All of these factors were folded into some very important resolutions passed by the IAU in 1997 and 2000, which form the basis for the coordinate systems used in the current version of NOVAS.

These resolutions have introduced concepts and terminology that may not be familiar to astronomers used to the traditional systems and names, which were used in the previous versions of NOVAS. The new systems are outlined in the following paragraphs, but see USNO Circular 179[9] for a much more complete description.

One of the resolutions passed by the IAU in 2000 defined two systems of space-time coordinates, one for the solar system and the other for the near-Earth environment, within the framework of General Relativity, by specifying the form of the metric tensors for each and the 4-dimensional space-time transformation between them. The former is called the **Barycentric Celestial Reference System (BCRS)** and the latter is called the **Geocentric Celestial Reference System (GCRS)**. The BCRS is the system appropriate for the basic ephemerides of solar system objects and astrometric reference data on galactic and extragalactic objects, i.e., the data in astrometric star catalogs. The GCRS is the system appropriate for describing the rotation of the Earth, the orbits of Earth satellites, and geodetic quantities such as instrument locations and baselines. The directions of astronomical objects as seen from the geocenter can also be expressed in the GCRS. The analysis of precise observations inevitably involves quantities expressed in both systems and the transformations between them. Subroutines in NOVAS may work with BCRS vectors, GCRS vectors, or both with appropriate conversions.

If the orientation of the BCRS axes in space is specified, the orientation of the GCRS axes then follows from the relativistic transformation between the two systems. The orientation of the BCRS is given by what is called the **International Celestial Reference System (ICRS)**. The ICRS is a triad of coordinate axes with their origin at the solar system barycenter and

[9] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

with axis directions effectively defined by the adopted coordinates of about 200 extragalactic radio sources observed by VLBI (listed in Section H of *The Astronomical Almanac*). The abbreviations BCRS and ICRS are often used interchangeably, because the two concepts are so closely related: the ICRS is the orientation of the BCRS; the BCRS is the metric for the ICRS.

The extragalactic radio sources that define the ICRS orientation are assumed to have no observable intrinsic angular motions. Thus, the ICRS is a "space-fixed" system (more precisely, a kinematically non-rotating system) and as such it has no associated epoch—its axes always point in the same directions with respect to distant galaxies. However, the ICRS was set up to approximate the conventional system defined by the Earth's mean equator and equinox of epoch J2000.0; the alignment difference is at the 0.02 arcsecond level, which is negligible for many applications. [FRAME]

Reference data for positional astronomy, such as the data in astrometric star catalogs (e.g., Hipparcos, UCAC, or 2MASS) or barycentric planetary ephemerides (e.g., JPL's DE405) are now specified within the ICRS; more precisely stated, they are specified within the BCRS, with respect to the ICRS axes.

In the near-Earth environment, celestial coordinates and related quantities are expressed with respect to the GCRS or reference systems that are derived from it. Because the orientation of the GCRS is derived from that of the BCRS, the GCRS can be thought of as the "geocentric ICRS."

Besides the GCRS itself, the two reference systems most commonly used for expressing the apparent directions of astronomical objects as seen from the Earth are the true equator and equinox of date and the **Celestial Intermediate Reference System**. Both are obtained from simple rotations of the GCRS, and in both cases, the fundamental plane is the true equator of date. [GCRSEQ] In the new terminology, the true or instantaneous equator is the plane orthogonal to the **Celestial Intermediate Pole (CIP)**, which is the celestial pole defined by the adopted precession and nutation algorithms (see section 1.4). The only difference between these two systems is the origin of right ascension: the points on the equator where RA = 0 are, respectively, the true equinox and the **Celestial Intermediate Origin (CIO).** The CIO is a recently introduced fiducial point on the equator that rigorously defines one rotation of the Earth (see section 1.5). [CIOLOC]

The GCRS and the two equatorial systems obtained from it have their origin at the geocenter. For topocentric coordinates or vectors, referred to an observer at a specific location on or near the surface of the Earth, there are analogous coordinate systems, although no semantic distinction is usually made between them and their geocentric equivalents. In NOVAS, the topocentric equivalent of the GCRS is referred to as the "local GCRS", and its spatial axes are assumed to be obtained from the GCRS by a Galilean transformation (simple shift of origin without a change in orientation). The two topocentric equator-of-date systems are obtained similarly.

Reference System for NOVAS Input Data: NOVAS now assumes that input reference data, such as catalog star positions and proper motions, and the basic solar system ephemerides, are provided in the ICRS (that is, within the BCRS as aligned to the ICRS axes), *or at least are consistent with it to within the data's inherent accuracy*. [PLACE, SOLSYS] The latter case will probably apply to most FK5-compatible data specified with respect to the mean equator and equinox of J2000.0 (the "J2000.0 system"). The distinction between the ICRS and the J2000.0 system becomes important only when an accuracy of 0.02 arcsecond or better is important. [FRAME] Nevertheless, because NOVAS is designed for the highest accuracy applications, the ICRS is mentioned as the reference system of choice for many input arguments to NOVAS subroutines.

Reference Systems for NOVAS Output Data: For output coordinates (e.g., the position of Mars on a certain date), there are three options for the coordinate system. (The coordinates themselves can be right ascension and declination or the components of a unit vector.) You can request coordinates expressed in the GCRS, the equator and equinox of date, or the Celestial Intermediate Reference System (equator and CIO of date). These coordinate systems can be requested for either geocentric or topocentric output. [PLACE, GCRSEQ]

NOVAS can also convert topocentric right ascension and declination, with respect to the equator and equinox of date, to local horizon coordinates, zenith distance and azimuth. [ZDAZ] The angular shift due to atmospheric refraction can be included as an option. Another routine is available to transform right ascension and declination to ecliptic longitude and latitude. Still another transforms right ascension and declination to galactic longitude and latitude. [EQECL, EQGAL]

Reference System for the Location of the Observer: The location of an Earth-based observer is specified in NOVAS by longitude, latitude, and height with respect to the World Geodetic System 1984 (WGS-84) reference ellipsoid. Coordinates provided by GPS (if uncorrected for the local datum) are referred to WGS-84, which is also sometimes called the Earth-centered Earth-fixed (ECEF) system. The **International Terrestrial Reference System (ITRS)** is a geocentric rectangular coordinate system used for high-precision work. WGS-84 coordinates are functionally equivalent (within a few centimeters) to ITRS coordinates. Thus, the geodetic positions used by NOVAS are consistent with the ITRS. [GEOPOS]

1.2 Computing Observable Quantities

NOVAS is mostly used for computing, for a selected object, the instantaneous angular coordinates (or the equivalent unit vector components) at which it might be observed, within one of several user-selected coordinate systems. Obviously the values of the angular coordinates computed by NOVAS depend on the coordinate system requested, but several phenomena can affect the observed position of a star or planet and are independent of the coordinate system. For stars, these effects are proper motion, parallax, gravitational light bending, aberration, and atmospheric refraction.

- Proper motion (generalized): the 3-D space motion of the star, relative to that of the solar system barycenter, between the catalog epoch and the date of interest. Assumed linear and computed from the catalog proper motion components, radial velocity, and parallax. Projected onto the sky, the motion amounts to less than 1 arcsecond per year (usually much less) except for a few nearby stars. [VECTRS, PROPMO]

- Parallax: the change in our perspective on stars in the solar neighborhood due to the position of the Earth in its orbit. Its magnitude is (distance in parsecs)$^{-1}$ and hence is always less than 1 arcsecond. [GEOCEN]

- Gravitational light bending: the apparent deflection of the light path in the gravitational field of the Sun and (to a much lesser extent) the other planets. Although it reaches 1.8 arcsecond at the limb of the Sun, it falls to 0.05 arcsecond 10º from the Sun and amounts to no more than a few milliarcseconds over the hemisphere of the sky opposite the Sun. [GRVDEF, GRVD]

- Aberration: the change in the apparent direction of light caused by the observer's velocity with respect to the solar system barycenter. Independent of distance, it is equal approximately to v/c, expressed as an angle. Therefore, it can reach 21 arcseconds for observers on the surface of the Earth and somewhat more for instruments in orbit. [ABERAT]

- Atmospheric refraction: the total angular change in the direction of the light path through the Earth's atmosphere; applies only to an observer on or near the surface of the Earth. The direction of refraction is always assumed to be parallel to the local vertical and a function only of zenith distance (although these assumptions may not be true in all cases). At optical wavelengths, its magnitude is zero at the zenith, about 1 arcminute at a zenith distance of 45°, and 0.5° at the horizon. Refraction is roughly proportional to the atmospheric pressure at the observer, but it also depends on other atmospheric parameters and the observational wavelength. [ZDAZ, REFRAC]

The same effects are relevant to objects in the solar system, except that the proper motion calculation is replaced by a function that retrieves the object's barycentric position from its ephemeris, as part of an iterative light-time calculation. [LITTIM, SOLSYS] Extragalactic objects can be considered to be stars with zero parallax and proper motion. The star or planet positions computed by considering all these effects obviously depend on the location of the observer; so that an observer on the surface of the Earth will see a slightly different position than one at the geocenter, the differences being greater for solar system objects, especially nearby ones (reaching about 1º for the Moon).

In computing these effects, the same subroutines are used for stars and planets, because NOVAS uses position vectors rather than directions; that is, internally it places all objects at their computed distance from the solar system barycenter. (Objects of unknown distance are placed on the "NOVAS celestial sphere" at a radius of 1 Gpc = 2×10^{14} AU.) [VECTRS] These vectors are all defined within the BCRS until the relativistic aberration calculation is applied, which effectively takes an input vector in the BCRS and produces an output vector in the GCRS.

Nomenclature: When all these effects are accounted for, we obtain star or planet coordinates in the GCRS that reflect where the star or planet actually appears in the sky. The coordinates can then be transformed to other reference systems if desired. We will call the results of this process, generically, the "apparent position" or "observed position" of the object.

However, some caution with the semantics is in order, because the term *apparent place* has traditionally been reserved specifically for the star or planet position we obtain by applying all these effects (except refraction) for a geocentric observer, with the coordinates expressed with respect to the true equator and equinox of date. For an observer on the surface of the Earth, the corresponding term is *topocentric place*. If the apparent star or planet position is instead expressed with respect to the mean equator and equinox of J2000.0, the terms used in NOVAS have been *virtual place* and *local place*, respectively. (Although these last two terms were suggested in the Kaplan et al. (1989) *Astronomical Journal* paper previously cited, there is no evidence that they have been widely used outside of the context of NOVAS.)

The above terminology is reflected in the names of the high-level subroutines that perform the computations, where "AP" stands for "apparent place", "TP" stands for "topocentric place", etc. [APSTAR, APPLAN, TPSTAR, TPPLAN, VPSTAR, VPPLAN, LPSTAR, LPPLAN] In addition, there is an *astrometric place* calculation used for some differential measurements; it is the same as virtual place except that light bending and aberration (and refraction) are *not* computed, under the assumption that these effects are the same for all objects within a small field of view. [ASSTAR, ASPLAN]

In response to the introduction of the new IAU-recommended coordinate systems, we must make some adjustments and additions to the nomenclature. The mean equator and equinox of J2000.0, considered as a geocentric system, has been replaced by the GCRS. The IAU Working Group on Nomenclature (2003–2006) recommended that the term *proper place* be used for what is called virtual place in NOVAS. With the introduction of the Celestial Intermediate Reference System, with its right ascension origin at the CIO, we now have two more possibilities for apparent positions, one geocentric and one topocentric. The geocentric coordinates are called the object's *intermediate place*, and right ascension measured with respect to the CIO is called *intermediate right ascension*.

The complete table of nomenclature for apparent positions of various types, updated for NOVAS 3.0, is given below.

Final Coordinate System	*Observer at Geocenter*	*Observer near Surface of Earth*
True equator and equinox of date	apparent place	Topocentric place
Celestial Intermediate Reference System	intermediate place	[no name]
GCRS	proper place or virtual place	local place
GCRS	astrometric place*	[no name]

*a variant of proper or virtual place, in which some calculations are omitted

The only difference between a position expressed in the Celestial Intermediate Reference System and the same position expressed with respect to the true equator and equinox of date is an offset in right ascension, the *equation of the origins*. The equation of the origins is the angle between the equinox and the CIO, both of which lie on the instantaneous equator. [EQXRA]

NOVAS subroutine PLACE can be used to compute all of these types of positions; the input arguments allow you to select both the location of the observer and the coordinate system in which the computed position is to be expressed. These selections make the nomenclature superfluous. PLACE does not apply atmospheric refraction, but that can be added by a subsequent call to ZDAZ. [PLACE, ZDAZ]

1.3 Time Scales for Astronomy

As explained in USNO Circular 179,[10] there are basically two kinds of time scales used in astronomy, those that are based on the Système International (SI) second ("atomic time") and those that are tied to the irregular rotation of the Earth; in essence, "lab" time and "astronomical" time, respectively. There are also theoretical time scales, not kept by any real clocks, that are the time basis for—that is, the 4th dimension of—the BCRS and GCRS.

We almost always start with **Coordinated Universal Time (UTC)**, which is the worldwide basis for civil time, distributed by GPS, WWV, cell phones, TV, etc. UTC is based on SI seconds at sea level on the rotating Earth. From UTC, one adds an integral number of (SI) seconds to obtain **International Atomic Time (TAI)**: TAI = UTC + ΔAT, where ΔAT is a total count of leap seconds in UTC. (For example, for 2009, ΔAT = 34s.) For a complete table of ΔAT values, see the EO web site.[11] The addition of a leap second to UTC, which increases ΔAT by 1s, is usually done, when needed, at 23:59:59 UTC on December 31 and is announced about six months in advance.

NOVAS Time Arguments: Typically, the first input argument to most NOVAS subroutines is the time of interest (for example, the time of an observation), expressed as a Julian date. Julian dates are simply a convenient format for representing a date and time in *any* time

[10] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179
[11] http://maia.usno.navy.mil/ser7/tai-utc.dat

scale, and are discussed below. There are two time scales that are used as the basis for most of the Julian dates that are input arguments to the higher level NOVAS routines.

The first is **Terrestrial Time (TT)**, which is effectively just a constant offset from TAI: $\mathrm{TT} = \mathrm{TAI} + 32.184\,\mathrm{s}$. Therefore, $\mathrm{TT} = \mathrm{UTC} + \Delta\mathrm{AT} + 32.184\,\mathrm{s}$. Historically, TT is considered continuous with the obsolete time scales Ephemeris Time (ET) and Terrestrial Dynamical Time (TDT). It is meant to be a smooth and continuous "coordinate" time scale independent of the rotation of the Earth.

The second time scale is **Universal Time (UT1)**, which is based on the rotation of the Earth. It is needed for computing sidereal time or the Earth Rotation Angle (ERA), which in turn allows one to compute hour angles, altitude and azimuth, or other topocentric quantities. UT1 is also obtained from UTC: UT1 = UTC + (UT1−UTC). The value of UT1−UTC is available in a daily-interval tabulation on the IERS web site[12] (data marked "P" are predictions); IERS publishes historical values is Bulletin B.[13] Note that the values of UT1−UTC often change at the millisecond level over one day. In computing the topocentric direction of a celestial object with respect to Earth-fixed axes (e.g., altitude and azimuth), 1 arcsecond accuracy in the final angles requires to 67 ms accuracy in UT1. Since UT1−UTC can have an absolute value up to 900 ms, it is an important correction for all but the crudest applications; that is, in most cases it is *not* acceptable to approximate UT1 as being equal to UTC.

A few of the lower level NOVAS subroutines use time arguments based on **Barycentric Dynamical Time (TDB)**. TDB differs from TT only by periodic variations (due mainly to the Earth's elliptical orbit and described by General Relativity), the largest of which has an amplitude of 1.6 ms and a period of one year. [TIMES] The difference between the two time scales can often be neglected in practice and this is noted in the subroutine preambles where appropriate. TDB is equivalent to $T_{\mathrm{eph},}$ the barycentric coordinate time argument of the Jet Propulsion Laboratory planetary and lunar ephemerides.

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

[12] http://maia.usno.navy.mil/ser7/mark3.out
[13] http://www.iers.org/MainDisp.csl?pid=36-9
[14] http://www.usno.navy.mil/USNO/astronomical-applications/data-services/jul-date

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. [JULDAT, CALDAT]

The epoch **J2000.0** is considered to be an event at the geocenter at Julian date 2451545.0 TT, which is 2000 January 1, 12h TT.

The difference ΔT = TT – UT1, expressed in seconds, should be pre-specified to NOVAS for use in subsequent calculations for a specific date. [SETDT] This is important for certain NOVAS subroutines that need to use both TT and UT1 internally but that require only one type of input time argument. A table of historical values of ΔT is on pages K9–K10 of *The Astronomical Almanac* and more recent values and predictions can be found online at EO web site.[15] Values of ΔT can also be computed from ΔT = 32.184s + ΔAT – (UT1–UTC). For example, on 2009 January 1, ΔT = 65.7768 s, which is based on a ΔAT value of 34s and a UT1–UTC value of +0.4071612s. More information on ΔT is given in section 1.6 below.

1.4 The Adopted Models for Precession and Nutation

Astronomers realized over a decade ago that the old standard models for the precession and nutation were in need of revision. The value of the angular rate of precession in longitude adopted by the IAU in 1976—and incorporated into the widely used precession formulation by Lieske and collaborators—is too large by about 0.3 arcsecond per century (3 milliarcseconds per year). The amplitudes of a number of the largest nutation components specified in the 1980 IAU Theory of Nutation are also known to be in error by several milliarcseconds. Both the precession and nutation errors are significant relative to current observational capabilities.

Thus, the resolutions passed by the IAU in 2000 mandated an improvement to the precession and nutation formulations. NOVAS 3.0 incorporates the models adopted in response to these resolutions. [PRECES, NUTATE] The precession model is the P03 solution of Capitaine, et al. (2003) A&A, 412, 567, as recommended by the IAU Working Group on Precession and the Ecliptic. The final report of the working group is Hilton et al. (2006) Celest. Mech. Dyn. Astr., 94, 351. The P03 precession model was formally adopted by the IAU in 2006. The nutation model is taken from Mathews et al. (2002) J. Geophys. Res., 107, B4, ETG 3. This model, referred to as the IAU 2000A nutation, consists of 1365 trigonometric terms, more than ten times the number in the previous model. [NOD, NU2000A] Because evaluation of nutation has always been the most computationally intensive task in NOVAS, you may notice an increase in execution time for some NOVAS applications.

To reduce execution time, NOVAS 3.0 provides an optional low-accuracy mode in which a truncated nutation series is used. This nutation series is specific to NOVAS and is referred to as 2000K. [LOACC, NU2000K] It consists of the largest 488 terms in the IAU 2000A

[15] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/long-term

series and provides an accuracy of about 0.1 milliarcsecond (specifically, 0.1 milliarcsecond for $\Delta\psi$ and about 0.04 milliarcsecond for $\Delta\varepsilon$ and $\Delta\psi \sin \varepsilon$). There is an even shorter IAU-approved nutation series, IAU 2000B, accurate to about 1 milliarcsecond, but this series is currently available only in the C version of NOVAS. More information on the implementation of nutation in NOVAS can be found section 2.6 and in Kaplan (2009), USNO Circular 181,[16] *Nutation Series Evaluation in NOVAS 3.0.*

As mentioned in section 1.1, the celestial pole described by the new precession and nutation models (with very small observational corrections) is called the **Celestial Intermediate Pole (CIP)**. The true equator of date, also called the instantaneous equator or the intermediate equator, is the plane orthogonal to the direction of the CIP. The CIP is also the rotational pole on the surface of the Earth (see section 1.6).

1.5 A New Model for the Rotation of the Earth about its Axis

IAU resolutions passed in 2000 also dealt in a very fundamental way with how the Earth's spin *around* its axis is described. The conventional treatment is based on the equinox and sidereal time; Greenwich (or local) sidereal time is just the Greenwich (or local) hour angle of the equinox of date. However, the equinox is constantly moving due to precession, so that sidereal time combines two angular motions, the Earth's rotation and the precession of its axis. (In the case of apparent sidereal time, nutation is also mixed in.) One rotation of the Earth is about 8 ms second longer than one mean sidereal day.

For about two decades, people who routinely deal with the most precise measurements of the Earth's rotation have been advocating for a change in the way it is described, and their ideas were introduced in resolutions passed by the IAU in 2000. In this new paradigm, the reference point on the moving celestial equator for the description of Earth rotation is called the **Celestial Intermediate Origin (CIO)**. Unlike the equinox, this point has no motion along the equator at all; as the orientation of the equator changes in space due to precession and nutation, the CIO remains on the equator but its instantaneous motion is always at right angles to it. [CIOLOC, CIORA] Thus, loosely speaking, two transits of the CIO across a terrestrial meridian define one rotation of the Earth. The CIO is a point on the celestial equator near RA = 0 (in the Celestial Intermediate Reference System it *defines* RA = 0), and there is a corresponding point on the terrestrial equator near longitude = 0 called the **Terrestrial Intermediate Origin (TIO).** For all astronomical purposes, the TIO can be considered a point fixed at geodetic longitude zero on the Earth's rotational equator.[17] In the new paradigm, the rotation of the Earth is specified by the angle (in the instantaneous equatorial plane) between the TIO and the CIO, which is a linear function of universal time (UT1). This angle is called the **Earth Rotation Angle (ERA)** and is designated by θ. [EROT]

[16] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-181

[17] The CIO and TIO are technically examples of *non-rotating origins*, although neither is fixed within its respective coordinate system. However, the slow drift of the TIO, due to polar motion, with respect to standard geodetic coordinates (the International Terrestrial Reference System, or, effectively, WGS-84) amounts to only 1.5 millimeters per century and is completely negligible for astronomical purposes.

Some internal calculations in NOVAS can be performed using either the equinox or the CIO as the fundamental fiducial point on the moving astronomical equator. The user can select the basis for these calculations; the default is to use the equinox. The results are identical to within a microarcsecond around the current time and the computational burden is about the same. [EQINOX, CIOTIO]

1.6 Terrestrial-Celestial Relationships

NOVAS uses the International Terrestrial Reference System (ITRS) for specifying locations and directions on or near the surface of the Earth. As mentioned at the end of section 1.1, the ITRS is consistent, to within a few centimeters, with WGS-84 coordinates provided by GPS, and it is sometimes referred to as the Earth-centered Earth-fixed system (ECEF). The ITRS is a geocentric system with the directions of its axes defined by the coordinates of a large number of observing stations, in a way completely analogous to the definition of the ICRS by the coordinates of extragalactic radio sources. The ITRS z-axis is toward the north geodetic pole and its x-axis is toward a point at longitude and latitude zero; the y-axis forms a right-handed system with the other two axes.

Practical applications of astronomical data often require relating terrestrial coordinates to celestial coordinates and vice versa. For example, we may want the position of a celestial object expressed with respect to the local horizon system. [ZDAZ] Or, we may have a vector, expressed in an Earth-fixed system, that represents some instrumental axis, and we would like to know where that vector is pointed on the celestial sphere. NOVAS can perform the terrestrial-to-celestial transformation or its inverse; specifically, the transformation from the ITRS to the GCRS, or the GCRS to the ITRS. [TERCEL] These transformations are a series of rotations that, taken together, represent the instantaneous orientation of the Earth in space. [PRECES, NUTATE, CIOBAS, SIDTIM, EROT, WOBBLE]

Not all aspects of the Earth's orientation are predictable. Polar motion represents the small shift of the geodetic north pole (the ITRS z-axis) with respect to the rotational axis (the CIP), the largest part of which must be determined from observations. Typically, the total shift amounts to a few tenths of an arcsecond (1-2 μrad, 10 meters) and is specified by the parameters x_p and y_p. The observational determinations are designated simply as x and y, and current values are available from IERS Bulletin A.[18] Past values can be obtained at the EO web site.[19] For most purposes we can set x_p=x and y_p=y (see USNO Circular 179[20] section 6.5.2 or the IERS Conventions 2009 section 5.5.1). Several NOVAS subroutines require as input the x_p and y_p values for the date of interest, although, if the final accuracy requirements are no better than 1 arcsecond, these values can be set to zero.

[18] http://maia.usno.navy.mil/ser7/ser7.dat
[19] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/daily
[20] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

Section 1.3 above recommends that the value of ΔT be specified to NOVAS before calculations are performed for a given range of dates. [SETDT] Because this value is used in only a few internal computations where the conversion from one time scale to another is not critical, the value of ΔT needs to be accurate to only about one second; therefore, one value will typically apply to all dates for a given year.

Finally, the new IAU precession and nutation models are neither perfect nor complete, and for very high accuracy applications, observational corrections are sometimes needed. These corrections now amount to less than 1 milliarcsecond (5 nrad). These corrections are available from the same sources as the polar motion determinations, and are designated as dX and dY (note that the units are *milli*arcseconds). In the rare cases where they are needed, they are pre-specified to NOVAS for use in subsequent calculations for a specific date. [CELPOL]

(Return to Table of Contents)

Chapter 2 Installing NOVAS

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 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
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)

2.2 Basic Installation Procedure

The main distribution file, NOVAS_F3.0.f, is the "standard" set of NOVAS F3.0 subroutines—except for subroutine SOLSYS, which is a low-level subroutine that supplies information on the position and velocity of the Earth and other solar system bodies. Three versions of SOLSYS are supplied as described below. (Each of the SOLSYS files is actually a set of several interrelated subroutines.) A functional set of NOVAS subroutines includes both the standard set of subroutines and one version of SOLSYS. We suggest you start with SOLSYS version 3, because it is self-contained and does not require access to any data files.

The code is mostly written to the Fortran 77 standard but there are several subroutines that have names longer than 6 characters and there are also several common constructs (now universally supported) from later versions of Fortran. The following is written for a Unix command-line environment, but the corresponding operations in other environments should be obvious.

[21] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/novas/novas-fortran

The source code can be compiled and object files produced by typing

```
      gfortran -c NOVAS_F3.0.f   NOVAS_F3.0_solsys1.f
                  NOVAS_F3.0_solsys2.f NOVAS_F3.0_solsys3.f
```

where you can substitute for “gfortran” the name of the particular Fortran compiler you use. The four object files that are created will have a “.o” extension rather than “.f”, but the main part of the name will be the same as for the corresponding source code file.

NOVAS is simply a group of subroutines, so there is no main (calling) program; you must supply that. You may wish initially to use the validation program checkout.f:

```
      gfortran    checkout.f   NOVAS_F3.0.o   NOVAS_F3.0_solsys3.o
```

If successful, this will produce a working executable, a.out.

For now, make sure that the file CIO_RA.TXT is *not* in the working directory. If you type a.out on the command line, the executable will run and produce an output file called checkout.out. (On some systems, you will need to specify the path for a.out in order to execute the program.) At the beginning of execution, you will receive two warning messages on your display about Jupiter and Saturn not being available; these messages simply alert you to the fact that the gravitational deflection of light subroutine cannot use Jupiter or Saturn as deflecting bodies when SOLSYS version 3 is in use (only the Sun’s deflection is evaluated). You should compare the results in the file checkout.out to the contents of file checkout.out.3 supplied in the distribution. The results should be identical.

You should also re-run a.out with the file CIO_RA.TXT in the working directory. When present in the working directory, NOVAS will use this file for certain computations relating to the Celestial Intermediate Origin; otherwise (as in the first run) NOVAS will use an internal computational procedure for those computations. The results should be identical at the microarcsecond level and should therefore produce no numerical differences in the output file, although the label on the sidereal time results that reads “using internal CIO” should change to “using external CIO”. If computational speed is a factor for your application, you may wish to test which method is faster on your system (but read section 2.4 below before doing so).

This setup of NOVAS—using SOLSYS version 3—will provide the positions of stars or extragalactic objects with errors not exceeding 0.3 milliarcsecond (if your input catalog data is that good) or the Sun with an error not exceeding 1 arcsecond. It will not provide the positions of solar system objects other than the Sun. For more general or demanding applications, NOVAS must use another version of SOLSYS.

2.3 Using External Solar System Ephemeris Files

If you need to compute star or radio source positions to better than 0.3 milliarcsecond, or the positions of the Sun to better than 1 arcsecond, or the positions of solar system bodies other than the Sun, you will have to use subroutine SOLSYS version 1 or 2, which use external ephemeris files.

SOLSYS version 2 calls the JPL software that accesses one of the JPL "development ephemerides" (DEnnn). If, for example, you have already installed the JPL software and file for ephemeris DE405, the command for linking NOVAS to DE405 might be something like

```
gfortran    myprog.f   NOVAS_F3.0.o  NOVAS_F3.0_solsys2.o
            ~/mylibs/jpl/jplsubs.o
```

where ~/mylibs/jpl is the directory in which you have stored the compiled JPL software, here called jplsubs.o. The ephemeris file itself might be stored elsewhere, in a directory path known to jplsubs; NOVAS does not access the JPL ephemeris file directly. In most installations, the JPL software reads the file on Fortran logical unit 12. If successful, the above command will produce a working executable, a.out.

Establishing a working copy of the JPL software and the DE405 binary files on your system is not, unfortunately, a trivial process. The files for doing that can be obtained directly from JPL[22] as discussed in Appendix C.

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.f
```

The file SS_EPHEM.TXT (or its alias) must be in the working directory at execution time. The file is read on Fortran logical unit 20. You can direct NOVAS to use a different file

[22] http://ssd.jpl.nasa.gov/?planet_eph_export

name or logical unit number at execution time by calling subroutine FILDEF at the start of your program, before any other NOVAS calls.

You can re-run the test program, checkout.f, with either SOLSYS version 1 or SOLSYS version 2 linked in. For SOLSYS version 1, or for SOLSYS version 2 when the JPL DE405 ephemeris is used, the output should be identical to the contents of checkout.out.1.

2.4 Making the NOVAS External Data Files More Efficient

As described above, NOVAS can be run without any reference to external data files by using SOLSYS version 3 and making sure that CIO_RA.TXT is *not* in the working directory. However, if either of the files SS_EPHEM.TXT (read by SOLSYS version 1) or CIO_RA.TXT (read by CIOLOC) are used, you can eliminate some unproductive search time by simply truncating them to include only the range of dates that are relevant for your particular applications.

Both files are plain-text (ASCII) files that can be modified by any text editor. The first record in each file is a header, which should remain as such (although its contents are not used by NOVAS), but all the other records contain a Julian date and information for that date. The records are in ascending date order. Simply trim from the beginning and ending of those files any records for dates that you are *sure* you will never need (be sure to allow at least ten extra dates on each end to allow the internal interpolation scheme to work properly). For example, SS_EPHEM.TXT contains data for years 2000 through 2020 but you may need only the range from 2010 to 2015. It is more important to trim off the "before" dates than it is the "after" dates. Leave the data within your overall anticipated date range as is—that is, do not attempt to make several groups of dates within the same file. The file that you create must end up (after the header record) as a fixed-interval file running from the first date to the last date with no gaps.

The file CIO_RA.TXT, which contains the right ascension of the CIO in the GCRS, contains six centuries of data, most of which is seldom used, so trimming it down is a good idea. In addition, you can make access to the data in CIO_RA.TXT even more efficient by converting it to binary direct-access format. Use program CIO_file.f to do that. This program is simple and self-contained—it does not need a link to NOVAS or anything else—and it requires no input from you. Simply make sure that CIO_RA.TXT (as you have edited it) is in the working directory and type

```
gfortran CIO_file.f
a.out
```

The program will create a file called CIO_RA.DA. To get NOVAS to use it instead of CIO_RA.TXT, use the following subroutine call, before any other NOVAS call:

```
CALL CIOFIL ( lun, 'CIO_RA.DA', 2 )
```

where *lun* is the Fortran logical unit number of your choice that NOVAS will use for reading the file. You can replace the file name CIO_RA.DA with whatever name you choose for the file; it can be a complete path (for example, /users/myaccount/novas/cio.bin) up to 40 characters long.

2.5 Do I have to Use a CIO File?

If a CIO file (either CIO_RA.TXT or CIO_RA.DA or whatever you choose to call them) is not present when NOVAS needs to determine the location of the CIO, NOVAS will simply revert to using an internal computation for this information. The results differ by at most only a few microarcseconds, and those differences are reached only for dates before 1900 or after 2200. The two approaches represent two independent algorithms for determining the location of the CIO on the equator, and the tiny differences for dates that are several centuries from now are of no practical consequence (see Chapter 5). There may be small differences in execution time for the two approaches, but those timing differences are likely to vary with the specific application—depending, for example, on the order and spread of the dates that are processed by NOVAS. There may also be applications for which an external file is not desirable. Because in many simple NOVAS applications, the location of the CIO is never needed, the best scheme is probably to start *without* using a CIO file. If your application's execution time is critical, you might want to experiment to see whether using one of the CIO files affects its performance one way or the other.

2.6 The NOVAS Low-Accuracy Mode and the Alternative Subroutines

NOVAS has a "low-accuracy" mode that can be invoked at any time by calling subroutine LOACC (it has no arguments). In this mode, a shortened nutation series is used; evaluating the series for nutation is usually the main computational burden in NOVAS, so using low accuracy mode improves execution time, often noticeably. This mode can be used when the accuracy requirements are not better than 0.1 milliarcsecond. The resulting time for Earth-rotation computations may be reduced by about two-thirds. You can revert to high-accuracy mode at any time by calling HIACC; high-accuracy mode is the default so it is not necessary to call HIACC initially. In high-accuracy mode, the IAU 2000A nutation series (1,365 terms) is used; in low-accuracy mode, the NOVAS 2000K nutation series (488 terms) is used. See section 1.4.

If your accuracy requirements are no better than about 0.05 arcsecond, you may wish to consider using alternative versions of two NOVAS subroutines. Swapping these subroutines can reduce the computational load for most applications *considerably*. The file NOVAS_F3.0_alt.f contains alternative versions of GRVDEF and NOD. Subroutine GRVDEF computes the gravitational deflection of light while NOD supervises the nutation series computation. The alternative version of GRVDEF is a dummy, which does not make any change to the input direction vector. The alternative version of NOD contains a very short nutation series (13 terms) that is used in NOVAS low-accuracy mode. Even after switching to the alternative version of NOD, calling LOACC is required in order to use this 13-term nutation series.

The standard version of GRVDEF should always be used if you have to compute the positions of objects that may pass close to the Sun. The gravitational deflection can exceed 0.05 arcsecond within 10º of the Sun.

(Return to Table of Contents)

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, e.g.

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

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.*

3.1 Initialization

CALL LOACC or CALL HIACC: Determines accuracy mode, which affects only certain computations (most importantly, nutation) relating to the Earth's orientation in space. The default mode is high accuracy. Call LOACC for calculations that need not be more accurate than 0.1 milliarcsecond, which will be the case for many practical applications. (See section 2.6 on installing alternative subroutines for even lower accuracy calculations.) The accuracy mode stays in effect until another call to LOACC or HIACC.

CALL EQINOX or CALL CIOTIO: Determines internal computation scheme—whether the equinox or the CIO will be used as the origin of right ascension on the moving equator within certain NOVAS subroutines that can work either way. This call does not affect any output options; NOVAS produces the same data using either scheme. However, it is best to call CIOTIO if you are specifically requesting data with respect to the CIO. Equinox mode is the default.

IEARTH, a variable, appears as an input argument in several subroutine calls shown below. In previous versions of NOVAS, IEARTH was the value of the body identification number for the Earth. In NOVAS 3.0, the value of this argument is not used; the argument has been kept as a placeholder in the subroutines that previously required it (for backwards compatibility), but it can have any integer value.

```
      IEARTH = 3
```

is a logical choice.

3.2 Setting Time Arguments

CALL SETDT (DELTAT): Sets the value of ΔT in seconds (see sections 1.3 and 1.6) for a group of dates to be processed, usually about a year's worth. The value of DELTAT should be good to the nearest second or better for the range of dates to be covered. Generally, that will require a call to SETDT for each year of dates to be processed, with the value of DELTAT appropriately set.

You will also have to consider how you will handle dates and times. As described in section 1.3, NOVAS uses either of two time scales, TT or UT1, as the basis for the time input argument to its higher-level subroutines. However, you may be working with UTC, Coordinated Universal Time. UTC is the basis for civil time systems worldwide and, because it is distributed quite accurately by GPS, is often used as the time-tag for observations. The key relationship is

$$\mathrm{TT} = \mathrm{UTC} + \Delta\mathrm{AT} + 32.184\mathrm{s}$$

where ΔAT is an integer representing the total count of leap seconds in UTC (for example, for 2009, ΔAT = 34s). Equivalently,

$$\mathrm{TT} = \mathrm{TAI} + 32.184\mathrm{s}$$

where TAI is International Atomic Time. If you will only be dealing with the geocentric celestial coordinates of objects, TT is all you will need.

If you will also be computing topocentric coordinates (for a specific location on the surface of the Earth), you will also need to obtain UT1. The key relationship is

$$\mathrm{UT1} = \mathrm{UTC} + (\mathrm{UT1}-\mathrm{UTC})$$

where UT1–UTC is interpolated from the daily values of this quantity published by the IERS.[23] Alternatively,

$$\mathrm{UT1} = \mathrm{TT} - \Delta\mathrm{T}$$

although for this purpose, a more accurate value of ΔT (and one that is changed more often) will be needed than is generally used for the argument of SETDT. See section 1.3 for more information on time scales, including sources of data that can be used for the values of ΔAT, UT1–UTC, and ΔT.

You can convert a date and time in the common format (year/month/day/hour/minute/second) to a Julian date, which is used by NOVAS for time arguments, by calling JULDAT. Dates and times based on UTC, TT, or UT1 (or any other time scale) can be converted using JULDAT; the output Julian date simply has the same basis as the input date and time. In the examples below, we will be using 2008 April 24, 10:36:18.0 UTC as the time of interest; this corresponds to a Julian date of 2454580.9441875 UTC.[24] We will also use a ΔAT value of 33 s and a UT1–UTC value of −0.387845 s, which are appropriate for this date.

[23] http://maia.usno.navy.mil/ser7/mark3.out

```
      DATA IYEAR, MONTH, IDAY, HOUR, LEAPS, UT1UTC, XP, YP /
     .     2008, 4, 24, 10.605D0, 33, -0.387845D0, -0.002D0, +0.529D0 /
```

So if we use

```
      CALL JULDAT  ( 2008, 4, 24, 10.605D0, UTCJD )
```

the output argument, UTCJD, will have a value of 2454580.9441875. The next few lines of code should be something like

```
      TTJD = UTCJD + ( LEAPS + 32.184D0 ) / 86400.D0
      UT1JD = UTCJD + UT1UTC / 86400.D0
      DELTAT = 32.184D0 + LEAPS - UT1UTC
      CALL SETDT ( DELTAT )
```

where 86400.D0 is the number of seconds in a day, and the value of DELTAT (ΔT) would be computed to be 65.571845 (seconds). If we had known the value of DELTAT (ΔT) to start with, rather than the value of UT1UTC (UT1–UTC), then the lines immediately above could have been replaced by

```
      TTJD = UTCJD + ( LEAPS + 32.184D0 ) / 86400.D0
      UT1JD = TTJD - DELTAT / 86400.D0
      CALL SETDT ( DELTAT )
```

3.3 Example 1—Position of a Star

Suppose we have the catalog data from star FK6 1307 (Groombridge 1830) for epoch J2000.0, expressed in the ICRS:

ICRS right ascension at J2000.0 (hours):	`RA2000 = 11.88299133D0`
ICRS declination at J2000.0 (degrees):	`DC2000 = 37.71867646D0`
Proper motion in RA (milliarcsceonds per year):	`PMRA = 4003.27D0`
Proper motion in dec (milliarcseconds per year):	`PMDEC = -5815.07D0`
Parallax (milliarcseconds):	`PARX = 109.21D0`
Radial velocity (kilometers per second):	`RV = -98.8D0`

To obtain the apparent geocentric place of the star on our date of interest, with respect to the equator and equinox of that date, simply call APSTAR, supplying it with the catalog quantities:

```
      CALL APSTAR ( TTJD, IEARTH, RA2000, DC2000, PMRA, PMDEC, PARLX, RV,
     .             RA, DEC )
```

[24] A “UTC Julian date” is something of a non sequitur, because UTC is not continuous (because of leap seconds). Here we are simply using JULDAT with UTC input to obtain a value that allows us to compute Julian dates in more well-behaved time scales.

The output coordinates, RA and DEC (hours and degrees, respectively), represent the apparent geocentric coordinates of the star, with respect to the true equator and equinox of date. The computation takes into account all time-dependent effects that shift the star's position from its catalog coordinates (except atmospheric refraction, which is location- and weather-dependent): the star's space motion to the date of interest, parallax due to the Earth's position in its orbit, gravitational light-bending in the solar system, aberration due to the Earth's orbital velocity, and the precession and nutation of the Earth's axis.

Important note: Hipparcos catalog data, although expressed with respect to the ICRS, refer to epoch 1991.25 and must be converted to epoch J2000.0 before being used as input arguments to any NOVAS subroutine. Use subroutine GETHIP to do this. (For example, see the code in checkout.f.) Most other modern catalogs, including the FK6 (used above), Tycho-2, UCAC, etc., provide data for epoch J2000.0 that need no conversion.

If we want the apparent topocentric place of the star, that is, the star's position as it would be seen (except for refraction) from a particular location on Earth, such as off the Atlantic Coast near Truro, Massachusetts

```
      DATA GLON, GLAT, HT / -70.D0, 42.D0, 0.D0 /
```

where GLON and GLAT are the location's geodetic longitude and latitude (degrees, with east longitude and north latitude positive) and HT is the height of the location above sea level (meters).

Then, the call to APSTAR should be followed by one to TPSTAR:

```
      CALL TPSTAR ( UT1JD, GLON, GLAT, HT, RAT, DECT )
```

where RAT and DECT reflect the position of the star as it would be seen from that particular location—the small differences from the geocentric coordinates RA and DEC arise mainly from diurnal aberration. TPSTAR uses catalog data on the star from the previous call to APSTAR.

3.4 Example 2—Position of the Moon

Obtaining the coordinates of solar system objects other than the Sun requires that SOLSYS version 1 or 2 must be in use. Use subroutine APPLAN:

```
      MOON = IDSS ('MOON')
      CALL APPLAN ( TTJD, MOON, IEARTH, RA, DEC, DIS )
```

where, again, RA and DEC are the apparent geocentric coordinates of the Moon (hours and degrees, respectively), with respect to the true equator and equinox of date, and DIS is the true (Euclidean) geocentric distance (AU) at time TTJD. IDSS returns the body identification number for a solar system object based on the version of SOLSYS in use. The computation of the angular coordinates of solar system objects takes light-time into account, along with the other effects (light bending, aberration, precession, and nutation) that also apply to stars.

To get the topocentric celestial coordinates of the Moon, call TPPLAN immediately after APPLAN:

```
      CALL TPPLAN ( UT1JD, GLON, GLAT, HT,   RAT, DECT, DIST )
```

However, there is a single subroutine, PLACE, that can be used for all types of positions of both stars and solar system objects. In fact, APSTAR, TPSTAR, APPLAN, and TPPLAN (and several other similar subroutines) are actually just special-purpose front-ends to PLACE. PLACE uses three array arguments, STAR and OBSERV for input and SKYPOS for output, dimensioned as follows:

```
      DIMENSION STAR(6), OBSERV(6), SKYPOS(7)
```

as well as several scalar arguments. For the coordinates of solar system bodies, array STAR does not need to be loaded (it is used only for stars). For observers on or near the surface of the Earth, OBSERV should be loaded with the observer's geodetic coordinates:

```
      OBSERV(1) = GLON
      OBSERV(2) = GLAT
      OBSERV(3) = HT
      OBSERV(4) = 0.D0
      OBSERV(5) = 0.D0
      OBSERV(6) = 0.D0
```

(If only geocentric coordinates are of interest, this array need not be loaded.) When using PLACE to compute the topocentric positions of nearby solar system objects, it is advisable, for high-accuracy work, to load OBSERV(4) with the value of ΔT (in seconds) *for the date requested*.   Otherwise, PLACE uses the value of ΔT from the last call to SETDT, which may not be accurate for the date and time of interest. In this case, it does not matter, because we are dealing with only a single date and a single value of ΔT, which we have provided to SETDT.

The call to PLACE to obtain topocentric coordinates of the Moon, with respect to the true equator and equinox of date then is

```
      CALL PLACE ( TTJD, 'MOON', 1, 1, STAR, OBSERV, SKYPOS )
```

after which the output array will contain:

SKYPOS(1):	Unit vector x component
SKYPOS(2):	Unit vector y component
SKYPOS(3):	Unit vector z component
SKYPOS(4):	Right Ascension (hours)
SKYPOS(5):	Declination (degrees)
SKYPOS(6):	True distance (AU)

SKYPOS(7): Radial velocity (kilometer per second)

Here, (x,y,z) is a unit vector in the apparent direction of the Moon, in the same coordinate system as the right ascension and declination (i.e., it is exactly equivalent to the spherical coordinates). PLACE has many options for both input and output; refer to its description in Chapter 4 or look at the beginning of the PLACE code.

Once you have the topocentric celestial coordinates of an object, these can be transformed into local altitude and azimuth by a call to ZDAZ. If we have used PLACE to obtain the topocentric coordinates, we set

```
      RAT = SKYPOS(4)
      DECT = SKYPOS(5)
```

then

```
      CALL ZDAZ ( UT1JD, XP, YP, GLON, GLAT, HT, RAT, DECT, IREFR,
     .                  ZD, AZ, RAR, DECR )
```

where IREFR is a refraction option (set IREFR = 0 for no refraction, or IREFR = 1 for standard atmospheric refraction), and XP and YP are the IERS pole coordinates x_p and y_p for the date of interest. (You can set XP = YP = 0.D0 if the required accuracy is not better than 1 arcsecond.) The output coordinates, ZD, AZ, RAR, and DECR, are, respectively, the zenith distance (degrees), azimuth (degrees), right ascension (hours), and declination (degrees). The output values of ZD, RAR, and DECR are affected by atmospheric refraction if IREF = 1. If IREF = 0, the output right ascension and declination values are the same as the input values. ZD and AZ are referred to the horizon system that is tangent to the Earth's reference ellipsoid at the observer's location; that is, the deflection of the vertical (the local undulation of the geoid) is not taken into account.

3.5 Example 3—Greenwich Sidereal Time

To obtain Greenwich sidereal time, call SIDTIM:

```
      CALL SIDTIM ( UT1JD, 0.D0, K, GST )
```

where GST is the output sidereal time (hours): Greenwich mean sidereal time if K = 0, Greenwich apparent sidereal time if K = 1. This subroutine and several others allow for a "split" input UT1 Julian date—high- and low-order parts in the first two arguments—for increased precision. Generally, the split would be at the Julian date's decimal point, with the day count in the first argument and the fraction of a day in the second. However, using two arguments for the Julian date provides more precision *only if the fractional part of the Julian date has been handled separately all along*. We have not done that, so here, the entire Julian date is just placed in the first argument and the second argument is set to 0.D0.

To compute local sidereal time (either mean or apparent), add the longitude (east positive) expressed in hours:

```
      ST = GST + GLON / 15.D0
```

The result may have to be reduced to the range 0 to 24 hours by statements similar to the following:

```
      IF ( ST .GE. 24.D0 ) ST = ST - 24.D0
      IF ( ST .LT.  0.D0 ) ST = ST + 24.D0
```

The quantity that is analogous to Greenwich apparent sidereal time in the CIO-based paradigm is θ, the ERA. It can be computed from:

```
      CALL EROT ( UT1JD, 0.D0, ERA )
```

where ERA is the Earth Rotation Angle (degrees). EROT, like SIDTIM, allows for a split input UT1 Julian date.

See Chapter 5 for information on the difference between Greenwich apparent sidereal time and the ERA, and how hour angles are computed in the two paradigms.

3.6 Example 4—Other Frequently Requested Quantities

In the following subroutine calls, vectors are used. Vectors are simply double-precision arrays with a dimension of 3. Most NOVAS internal calculations are performed with vectors and matrices. The following vectors are referred to below:

```
      DIMENSION POS(3), VEL(3), POSE(3), VTER(3), VCEL(3)
```

To obtain the barycentric or heliocentric coordinates (BCRS vectors) of a solar system body, for example, Mars, call SOLSYS:

```
      CALL SOLSYS ( TDBJD, IDSS('MARS'), K, POS, VEL, IERR )
```

where we can often approximate TDBJD = TTJD[25], and the output position and velocity vectors are POS and VEL (components in AU and AU/day, respectively). For barycentric positions, set K = 0; for heliocentric positions, set K = 1. IERR is an error flag returned, which normally has a value of 0 if everything is OK. Each version of SOLSYS has a corresponding version of IDSS, which returns the required body identification number.

If POS is a heliocentric vector, it can be transformed to the ecliptic system (fixed ecliptic of J2000.0) by calling EQEC:

```
      CALL EQEC ( 0.D0, 0, POS, POSE )
```

[25] Strictly, SOLSYS requires a TDB-based Julian date as input. TDB differs from TT by at most 1.7 ms; see section 1.3. If this time difference is important, use subroutine TIMES to determine the difference between the two time scales and adjust TDBJD accordingly. For the example here, neglecting the difference can lead to an error in the position of Mars of about 50 meters.

where POSE is the output vector in the ecliptic system (same units as POS). POSE could then easily be converted to heliocentric spherical coordinates: ecliptic longitude, ecliptic latitude, and radius vector.

```
      CALL ANGLES ( POSE, ELON, ELAT )
      ELON = ELON * 15.D0
      R = DSQRT (POSE(1)**2 + POSE(2)**2 + POSE(3)**2)
```

Finally, it is sometimes useful to transform a vector from the terrestrial reference system to the celestial reference system. The vector might represent a geographic position, a geodetic reference line or direction, or an instrumental axis. For this transformation, the vector starts out as an Earth-fixed vector expressed with respect to the ITRS axes. For example, the vector toward the local vertical (orthogonal to the ellipsoid at the place of interest) is simply ($\cos\phi\cos\lambda$, $\cos\phi\sin\lambda$, $\sin\phi$), where ϕ is the geodetic latitude and λ is the longitude. A vector along a telescope's polar axis would nominally point toward (0,0,1) in this system. Any such ITRS vector can be transformed into the equivalent GCRS vector with a single call to TERCEL:

```
      CALL TERCEL ( UT1JD, 0.D0, XP, YP, VTER, VCEL )
```

where VTER is the input vector (terrestrial, ITRS) and VCEL is the equivalent output vector (celestial, GCRS). The components of VTER can be in any units; VCEL will be in the same units. VCEL will sweep around the celestial sphere as the Earth rotates, i.e., as UT1JD advances. (TERCEL, like SIDTIM, allows for a split input UT1 Julian date.) Use ANGLES to obtain VCEL's instantaneous spherical coordinates:

```
      CALL ANGLES ( VCEL, RA, DEC )
```

where RA and DEC are the GCRS right ascension and declination (hours and degrees, respectively) of the point on the celestial sphere toward which VCEL points. At any UT1JD, VCEL can be compared to the directions of stars computed by PLACE for the equivalent TTJD, with LOCATN = 1 and ICOORD = 0.

(Return to Table of Contents)

Chapter 4 Descriptions of Individual Subroutines

NOVAS FORTRAN SUBROUTINE, FUNCTION, AND ENTRY NAMES

Entry Name	Level	Purpose
PLACE *	super	Computes the apparent direction of a star or solar system body, given the time and observer's location. Direction is expressed in one of several selectable coordinate systems.
SIDTIM	super	Computes Greenwich sidereal time, either mean or apparent.
TERCEL **	super	Transforms arbitrary vector in rotating Earth-fixed (ITRS) system to space-fixed (ICRS) system. Terrestrial to celestial transformation.
ZDAZ	super	Transforms topocentric right ascension and declination to zenith distance and azimuth. Optionally accounts for atmospheric refraction.
CATRAN	super	Transforms a star's catalog quantities for a change of epoch and/or equator and equinox.
GETHIP	super	Converts Hipparcos catalog data at epoch J1991.25 to epoch J2000.0.
APSTAR	super	Computes the apparent place of a star, given its catalog data.
TPSTAR	super	If called after APSTAR, returns topocentric place of same star, given geodetic location of observer.
VPSTAR	super	Computes the virtual place of a star, given its catalog data.
LPSTAR	super	If called after VPSTAR, returns local place of same star, given geodetic location of observer.
ASSTAR	super	Computes the astrometric place of a star, given its catalog data.
MPSTAR	super	Computes the ICRS/J2000.0 place of a star, given its apparent place.
APPLAN	super	Computes the apparent place of a planet or other solar system body.
TPPLAN	super	If called after APPLAN, returns topocentric place of same planet, given geodetic location of observer.
VPPLAN	super	Computes the virtual place of a planet or other solar system body.
LPPLAN	super	If called after VPPLAN, returns local place of same planet, given geodetic location of observer.
ASPLAN	super	Computes the astrometric place of a planet or other solar system body.
ABERAT	util	Adjusts position vector for aberration of light due to motion of Earth.
ANGLES	util	Converts position vector to RA and dec.
CIOBAS *	util	Returns orthonormal basis vectors, with respect to GCRS, of right-handed system defined by CIP (z-direction) and CIO (x-direction).
DLIGHT *	util	For a star, returns difference in light-time between solar system barycenter and observer. Or, returns the light-time from observer to point on light ray closest to a given solar system body.
ECEQ *	util	Converts ecliptic position vector to equatorial position vector.
EQEC *	util	Converts equatorial position vector to ecliptic position vector.
EQECL *	util	Converts RA and dec to ecliptic longitude and latitude.
EQGAL *	util	Converts ICRS RA and dec to galactic longitude and latitude.
EROT *	util	Returns ERA (θ) for given UT1 Julian date.
FRAME *	util	Transforms vector between dynamical reference system (mean equator and equinox of J2000.0) and ICRS.
GCRSEQ *	util	Converts GCRS RA and dec to coordinates with respect to the equator of date (mean or true).
GEOCEN	util	Changes origin of coordinates from barycenter of solar system to center of mass of Earth.

Entry Name	Level	Purpose
GEOPOS *	util	Computes geocentric position and velocity, in GCRS, of observer on or near the surface of Earth.
GETVEC *	util	Retrieves last computed position on the sky as a unit vector.
GRVD *	util	Corrects position vector for deflection of light in gravitational field of given body.
GRVDEF **	util	Computes total gravitational deflection of light for an object due to major solar system bodies.
LIMANG *	util	Determines angle of object above or below earth's limb (horizon).
LITTIM *	util	Computes the position of a solar system body antedated for light-time.
NUTATE	util	Applies nutation to position vector.
PRECES	util	Applies precession to position vector.
PROPMO	util	Updates the position vector of a star to allow for its space motion.
RADVL *	util	Predicts radial velocity of observed object as would be measured by spectroscopy.
SETDT *	util	Allows specification of ΔT value for calculation of sidereal time, etc.
SPIN	util	Rotates vector by specified angle about the z-axis.
TERRA	util	Converts geodetic coordinates to geocentric position vector.
VECTRS	util	Converts a star's RA, dec, proper motion, etc., to position and velocity vectors.
WOBBLE	util	Adjusts Earth-fixed vector for polar motion.
ASTCON *	basic	Supplies values of astronomical constants.
CALDAT	basic	Computes calendar date and time, given Julian date.
CELPOL	basic	Allows for the specification of celestial pole offsets for high-precision applications.
CIOFIL *	basic	Specifies logical unit number and file identifier of input file containing series of CIO RA values.
CIOLOC *	basic	Returns location of CIO as RA with respect to either GCRS origin or true equinox of date, given TDB Julian date.
CIORA *	basic	Computes true right ascension of the CIO, given TT Julian date.
CIORD *	basic	Returns array of Julian dates and corresponding values of RA of CIO (in GCRS), given TDB Julian date and number of dates desired.
CIOTIO *	basic	Switches internal computation scheme to use CIO as origin of RA for certain routines; best choice when requesting data with respect to CIO.
EECT2000 *	basic	Returns complementary terms for equation of the equinoxes.
EQINOX *	basic	Switches internal computation scheme to use equinox as origin of RA for certain routines; default mode.
EQXRA *	basic	Computes intermediate RA of the equinox, given TDB Julian date.
ETILT	basic	Provides information on orientation of Earth's axis: obliquity, nutation parameters, etc.
FUNARG	basic	Computes fundamental arguments (mean elements) of the Sun and Moon.
GETDT *	basic	Returns current value of ΔT.
HIACC *	basic	Switches to high-accuracy mode for certain calculations relating to Earth orientation; default mode.
IDSS *	basic	Returns body identification number, given name of solar system body, for version of SOLSYS in use.
JULDAT	basic	Computes Julian date, given calendar date and time.
LOACC *	basic	Switches to low-accuracy mode for certain calculations relating to Earth orientation; acceptable for applications that do not require accuracy better than 0.1 milliarcsecond.
NOD	basic	Supervises the calculation of nutation parameters and provides nutation in longitude and obliquity.

Entry Name	Level	Purpose
NU2000A *	basic	Evaluates high-accuracy nutation series (IAU 2000A model).
NU2000K *	basic	Evaluates low-accuracy nutation series (truncated version of IAU 2000A).
REFDAT	basic	Allows for the specification of weather observations, etc., for refraction calculation.
REFRAC	basic	Computes atmospheric refraction in zenith distance.
SOLSYS	basic	Provides position and velocity vectors of a solar system body.
TIMES	basic	Converts Barycentric Dynamical Time (TDB) to Terrestrial Time (TT).

*   New subroutine, function, or entry in NOVAS F3.0.
** Name change from NOVAS F2.0: TERCEL was PNSW; GRVDEF was SUNFLD.

Descriptions of some of the most frequently used NOVAS subroutines follow.   The comments in the source code at the beginning of each NOVAS subroutine are intended to provide enough information for correct usage.

In the documentation that follows, arguments marked **in** are input arguments to the subroutine: values for these must be supplied by the calling program. Arguments marked **out** are output arguments from the subroutine: the subroutine computes and returns values for these.   Note that all floating point arguments are DOUBLE PRECISION.

PLACE

Entry:	CALL PLACE (TJD, OBJECT, LOCATN, ICOORD, STAR, OBSERV, SKYPOS)
Purpose:	Computes the apparent direction of a star or solar system body at a specified time and in a specified coordinate system.
Arguments:	INTEGER LOCATN, ICOORD DOUBLE PRECISION TJD, STAR(6), OBSERV(6), SKYPOS(7) CHARACTER OBJECT

TJD:	TT Julian date for place (in).
OBJECT:	Character string identifying object of interest (in): For a solar system body, specify the name using all upper-case letters (‘SUN’, ‘MOON’, ‘JUPITER’, etc.), or specify the body identification number in a 4-character string of the form ‘=nnn’, where nnn is the body identification number. For a star, provide a blank string, the word ‘STAR’, or any string beginning with ‘*’.
LOCATN:	Code specifying location of observer (in): LOCATN = 0 for observer at geocenter LOCATN = 1 for observer on or near surface of the Earth LOCATN = 2 for observer on near-Earth spacecraft
ICOORD:	Code specifying coordinate system of output position (in): ICOORD = 0 for GCRS (or “local GCRS”) ICOORD = 1 for true equator and equinox of date ICOORD = 2 for true equator and CIO of date (Celestial Intermediate Reference System) ICOORD = 3 for ICRS astrometric coordinates, i.e., without light deflection or aberration
STAR:	Array of catalog data for star (not referred to if solar system body requested)(in): STAR(1) = ICRS right ascension at J2000.0 in hours STAR(2) = ICRS declination at J2000.0 in degrees STAR(3) = ICRS proper motion in RA in milliarcseconds/year STAR(4) = ICRS proper motion in dec in milliarcseconds/ year STAR(5) = parallax in milliarcseconds STAR(6) = radial velocity in kilometers/second (Further star array elements are reserved for future use.)
OBSERV:	Array of data specifying location of observer (in): For LOCATN = 0, not used

For LOCATN = 1,
OBSERV(1) = observer's geodetic (WGS-84) longitude (east positive) in degrees
OBSERV(2) = observer's geodetic (WGS-84) latitude (north positive) in degrees
OBSERV(3) = observer's height above Earth's reference ellipsoid (approximately, height above mean sea level) in meters
OBSERV(4) = value of ΔT in seconds (ΔT = TT – UT1)
OBSERV(5) = (not used, reserved for future use)
OBSERV(6) = (not used, reserved for future use)
For LOCATN = 2,

OBSERV(1) = geocentric X in km OBSERV(2) = geocentric Y in km OBSERV(3) = geocentric Z in km OBSERV(4) = geocentric X-dot in km/s OBSERV(5) = geocentric Y-dot in km/s OBSERV(6) = geocentric Z-dot in km/s	Z is toward CIO; X is toward true equinox

SKYPOS: Array of output data specifying object's place on the sky at time TJD, with respect to the specified output coordinate system (out):

SKYPOS(1) = X, dimensionless SKYPOS(2) = Y, dimensionless SKYPOS(3) = Z, dimensionless	unit vector toward object

SKYPOS(4) = apparent, topocentric, or astrometric right ascension in hours
SKYPOS(5) = apparent, topocentric, or astrometric declination in degrees
SKYPOS(6) = true (geometric, euclidian) distance to solar system body in AU at time TJD, or 0.D0 for star
SKYPOS(7) = radial velocity in kilometers per second
(Further SKYPOS array elements are reserved for future use.)

Discussion:

This subroutine computes the apparent direction of a star or solar system body at a specified time and in a specified coordinate system. The algorithms used are based on Kaplan, et al. (1989), Astron. J., 97, 1197, with some enhancements from Klioner (2003), Astron. J., 125, 1580.

The apparent direction of a star computed by this subroutine takes into account the star's proper motion (linear 3-D space motion) from the catalog epoch to the date requested, parallax, the gravitational deflection of light by solar system bodies (mostly the Sun), and aberration. The same effects are computed for solar system bodies, except that the proper motion calculation is replaced by an algorithm that retrieves the object's barycentric position from its ephemeris, as part of an iterative light-time calculation. Extragalactic objects are treated as stars with zero proper motion and parallax. The result in all cases is an apparent direction expressed in the GCRS, which is optionally transformed into either of two other output coordinate systems, as specified by argument ICOORD.

"Astrometric place" is a variant of the above calculation that is appropriate for some types of differential measurements. Light bending and aberration are ignored under the assumption that they are the same for all objects within a small area of the sky. Astrometric places are expressed in the ICRS.

The observer's location may be at the geocenter, on or near the surface of the Earth, or in orbit around the Earth, as specified by the arguments LOCATN and OBSERV.

PLACE does not take into account atmospheric refraction (which would be appropriate only for observers on or near the surface of the Earth), but its effect can be added by a subsequent call to subroutine ZDAZ.

For stars, the required input data, stored in array STAR, are the standard five astrometric quantities from a catalog, together with radial velocity if known. Any parameter should be set to 0.D0 if its value is unknown. All catalog data used as input to this subroutine must apply to epoch J2000.0 and be expressed with respect to the ICRS. (For Hipparcos catalog data, see subroutine GETHIP.) Extragalactic objects should be treated as stars, but with all input parameters set to 0.D0 except for the catalog right ascension and declination. For solar system bodies, array STAR is not referred to; instead, the argument OBJECT must contain the name of the body (or its identification number), from the list of objects supported by the version of subroutine SOLSYS in use.

The values of LOCATN and ICOORD for various kinds of place are:

LOCATN = 0, ICOORD = 0	⇒ virtual place (= proper place)
LOCATN = 1, ICOORD = 0	⇒ local place
LOCATN = 0, ICOORD = 1	⇒ apparent place
LOCATN = 1, ICOORD = 1	⇒ topocentric place
LOCATN = 0, ICOORD = 2	⇒ intermediate place
LOCATN = 1, ICOORD = 2	⇒ topocentric intermediate place
LOCATN = 0, ICOORD = 3	⇒ astrometric place
LOCATN = 1, ICOORD = 3	⇒ topocentric astrometric place

(Only the place names underlined above are widely recognized outside of the context of NOVAS.)

NOVAS subroutines APSTAR, TPSTAR, APPLAN, TPPLAN, etc., are now just special-purpose front-ends to PLACE.

Important: If LOCATN = 1 (observer on or near surface of Earth), a value of ΔT in seconds is required. The value will be taken from OBSERV(4), unless that element is 0.D0, in which case the last value of ΔT defined by the user in a call to SETDT will be used. Whether taken from OBSERV(4) or SETDT, the value of ΔT will be used internally for determining the ERA (θ), needed for computing topocentric places. An error in ΔT of 1s can result a topocentric place error of up to 0.3 arcsecond for the Moon, but proportionally less for more distant bodies (e.g., 3 milliarcseconds for Venus at its closest). Distance errors of up to 500 m (3×10^{-9} AU) can also result, independent of distance. If errors of this magnitude are important, care needs to be taken in specifying a more accurate value of ΔT. An error in ΔT of 1s will not result in a significant error in the topocentric places of stars.

SKYPOS(7), the radial velocity, is the predicted radial velocity measure (z) times the speed of light, an inherently spectroscopic quantity. For a star, it includes all effects, such as gravitational red shift, contained in the catalog barycentric radial velocity measure, which is assumed given in STAR(6). For a solar system body, it applies to a fictitious emitter at the center of the observed object, assumed to be massless (no gravitational red shift), and does not in general apply to reflected light.

Arrays STAR and SKYPOS may be expanded in the future. Such expansion could be provided for in the calling code by dimensioning these arrays with 20 and 10 elements, respectively, even though elements beyond STAR(6) and SKYPOS(7) are not now referred to in this subroutine.

(Return to subroutine entry list)

SIDTIM

Entry:	CALL SIDTIM (TJDH, TJDL, K, GST)
Purpose:	Computes the Greenwich sidereal time, either mean or apparent, at a specific UT1 time.
Arguments:	INTEGER K DOUBLE PRECISION TJDH, TJDL, GST

TJDH:	UT1 Julian date for sidereal time, high-order part (in).
TJDL:	UT1 Julian date for sidereal time, low-order part (in).
K:	Sidereal time selection code (in): K = 0 produces Greenwich mean sidereal time K = 1 produces Greenwich apparent sidereal
GST:	Greenwich (mean or apparent) sidereal time in hours (out).

Discussion:

This subroutine computes Greenwich sidereal time, either mean (if K = 0) or apparent (if K = 1).

The input Julian date, which must be in the UT1 time scale, may be split into two parts to ensure the highest precision in the computation. For example, set TJDH equal to the integral part of the Julian date and set TJDL equal to the fractional part. Generally, this will be advantageous only if the low-order part has been treated separately within the calling program; for example, if the time of day has been stored in its own variable(s), from which TJDL is constructed.

For many applications the position of the split is not critical as long as the sum TJDH + TJDL is correct: in particular, when used with computers providing 16 decimal digits of precision in DOUBLE PRECISION variables, this subroutine will yield values of GST precise to about 0.1 millisecond even if TJDH contains the entire Julian date and TJDL = 0.D0.

If K = 1 (for apparent sidereal time), the output value of GST will correctly reflect the celestial pole offset in longitude if routine CELPOL has previously been called.

(Return to subroutine entry list)

TERCEL

Entry:	CALL TERCEL (TJDH, TJDL, XPOLE, YPOLE, VEC1, VEC2)
Purpose:	Transforms a vector from the terrestrial system (ITRS) to the celestial system (GCRS).
Arguments:	DOUBLE PRECISION TJDH, TJDL, XPOLE, YPOLE, VEC1(3), VEC2(3)

TJDH:	UT1 Julian date for transformation, high-order part (in).
TJDL:	UT1 Julian date for transformation, low-order part (in).
XP:	Conventionally defined x_p coordinate of Celestial Intermediate Pole with respect to ITRS pole, in arcseconds (in).
YP:	Conventionally defined y_p coordinate of Celestial Intermediate Pole with respect to ITRS pole, in arcseconds (in).
VEC1:	Vector referred to ITRS axes (terrestrial system) (in).
VEC2:	Vector referred to GCRS axes (celestial system) (out).

Discussion:

This subroutine rotates a vector from the terrestrial to the celestial system. Specifically, it transforms a vector in the ITRS (a rotating Earth-fixed system) to the GCRS (a local space-fixed system) by applying rotations for polar motion, Earth rotation, nutation, precession, and the dynamical-to-GCRS frame tie. The input vector might represent a cardinal direction at the observer's position, a geodetic baseline, or some instrumental axis. The units for the vector components are arbitrary and the output vector will have the same units as the input vector. Geodetic coordinates in the WGS-84 system, also sometimes called the Earth-centered Earth-fixed (ECEF) system, can be considered to be compatible with the ITRS.

This subroutine allows for the input UT1 time to be represented as a split Julian date. See the discussion in the description of subroutine SIDTIM.

Set XP = YP = 0.D0 to omit the polar motion rotation.

See also subroutine SETDT to set the value of ΔT (ΔT = TT – UT1) to be used by this subroutine and others.

Both TJDH and TJDL should be non-negative for normal use (TJDL = 0.D0 is OK). A negative value of TJDH is used to invoke a special option where the output vector is produced with respect to the equator and equinox of date, and the date for which the transformation applies is taken from TJDL only. This special option works only in equinox mode.

(Return to subroutine entry list)

ZDAZ

Entry: CALL ZDAZ (UJD, XP, YP, GLON, GLAT, HT, RA, DEC, IREFR, ZD, AZ, RAR, DECR)

Purpose: Transforms topocentric right ascension and declination to zenith distance and azimuth; can also adjust coordinates for atmospheric refraction.

Arguments: INTEGER IREFR
DOUBLE PRECISION UJD, XP, YP, GLON, GLAT, HT, RA, DEC, ZD, AZ, RAR, DECR

UJD:	UT1 Julian date, or equivalent Greenwich apparent sidereal time in hours, for coordinate calculation (in).
XP:	Conventionally defined x_p coordinate of Celestial Intermediate Pole with respect to ITRS pole, in arcseconds (in).
YP:	Conventionally defined y_p coordinate of Celestial Intermediate Pole with respect to ITRS pole, in arcseconds (in).
GLON:	Observer’s geodetic (WGS-84) longitude (east positive) in degrees (in).
GLAT:	Observer’s geodetic (WGS-84) latitude (north positive) in degrees (in).
HT:	Height of observer above the Earth’s reference ellipsoid (approximately, height above mean sea level) in meters (in).
RA:	Topocentric right ascension of object of interest, in hours, referred to true equator and equinox of date (in).
DEC:	Topocentric declination of object of interest, in degrees, referred to true equator and equinox of date (in).

IREFR:	Atmospheric refraction option (in):
	IREFR = 0 omits refraction
	IREFR = 1 applies refraction

ZD:	Topocentric zenith distance in degrees, affected by refraction if IREFR = 1 (out).
AZ:	Topocentric azimuth (measured east from north) in degrees (out).
RAR:	Topocentric right ascension of object of interest, in hours, referred to true equator and equinox of date, affected by refraction if IREFR = 1 (out).
DECR:	Topocentric declination of object of interest, in degrees, referred to true equator and equinox of date, affected by refraction if IREFR = 1 (out).

Discussion:

This subroutine takes the topocentric celestial coordinates of an object and computes the equivalent local horizon coordinates. The subroutine uses a method that properly accounts for polar motion, which is significant at the sub-arcsecond level. Atmospheric refraction can be included in the transformation, and if so, refraction is applied to both kinds of coordinates (this can be useful for telescope pointing). Refraction, when requested, is computed by subroutine REFRAC.

RA and DEC, the input topocentric right ascension and declination, can be obtained from PLACE (with LOCATION = 1 and ICOORD = 1), or TPSTAR or TPPLAN. UJD, the input time argument, may be specified either as a UT1 Julian date (e.g., 2451251.299358D0) or an hour and fraction of Greenwich apparent sidereal time (e.g., 19.184592D0). Values for XP and YP, the coordinates of the pole, can be obtained from IERS[26] Bulletins A and B, although XP and YP can be set to 0.D0 if sub-arcsecond accuracy is not needed. (If refraction is requested, sub-arcsecond accuracy is unlikely.) The observer's height, HT, is used only for refraction, that is, if IREFR = 1.

In this subroutine, the directions ZD = 0.D0 (the zenith) and AZ = 0.D0 (north) are considered fixed in the terrestrial frame. Specifically, the zenith is along the geodetic normal (orthogonal to the ellipsoid at the observer's position), and north is toward the ITRS pole. Geophysical deflection of the vertical is not considered.

If IREFR = 0 (no refraction), then RAR = RA and DECR = DEC.

(Return to subroutine entry list)

[26]http://hpiers.obspm.fr

CATRAN

Entry: CALL CATRAN (IT, DATE1, RA1, DEC1, PMRA1, PMDEC1, PARX1, RV1, DATE2, RA2, DEC2, PMRA2, PMDEC2, PARX2, RV2)

Purpose: Transforms a star's catalog quantities for a change of epoch and/or equator and equinox. It can also be used to transform catalog quantities on the dynamical equator and equinox of J2000.0 to the ICRS, or vice versa.

Arguments: INTEGER IT
DOUBLE PRECISION DATE1, RA1, DEC1, PMRA1, PMDEC1, PARX1, RV1, DATE2,RA2,DEC2,PMRA2,PMDEC2,PARX2,RV2

IT:	Transformation option (in): IT = 1 changes epoch (same mean equator and equinox) IT = 2 changes mean equator and equinox (same epoch) IT = 3 changes mean equator and equinox and epoch IT = 4 changes mean equator and equinox of J2000.0 to ICRS IT = 5 changes ICRS to mean equator and equinox of J2000.0
DATE1:	TT Julian date, or year, of original catalog data (the following six arguments) (in).
RA1:	Original mean right ascension in hours (in).
DEC1:	Original mean declination in degrees (in).
PMRA1:	Original proper motion in RA in milliarcseconds/year (in).
PMDEC1:	Original proper motion in dec in milliarcseconds/year (in).
PARX1:	Original parallax in milliarcseconds (in).
RV1:	Original radial velocity in kilometers per second (out).
DATE2:	TT Julian date, or year, for transformed output data (the following six arguments) (in).
RA2:	Transformed mean right ascension in hours (out).
DEC2:	Transformed mean declination in degrees (out).
PMRA2:	Transformed proper motion in RA in milliarcseconds/year (out).
PMDEC2:	Transformed proper motion in dec in milliarcseconds/year (out).
PARX2:	Transformed parallax in milliarcseconds (out).
RV2:	Transformed radial velocity in kilometers/second (out).

Discussion:

Subroutine CATRAN performs various transformations on star catalog data. Only catalog reference data, not observed quantities, should be processed by this subroutine.

For IT = 1, 2, or 3, two dates, DATE1 and DATE2, must be specified: the input data is associated with the first date, and the output data is associated with the second date. Two transformations are available:

IT = 1: The star's data is updated to account for the star's space motion between the first and second dates, within a fixed reference system. That is, the *epoch* of the data is changed, but not the equator and equinox (or other system).

IT = 2: The reference frame within which the star's coordinates and proper motion are expressed is rotated corresponding to precession between the first and second dates. The star's position in space is not changed. That is, the *equator and equinox* of the data are changed, but not the epoch.

Setting IT = 3 requests both transformations, and is the most common case.

The two date arguments, DATE1 and DATE2, may be specified either as a Julian date (e.g., 2433282.5D0) or a Julian year and fraction (e.g., 1950.0D0). (Values less than 10000.D0 are assumed to represent years.)

The IT = 1 and IT = 3 transformations are appropriate only for objects with linear (or no) space motion; do not use them for components of binary stars. Also, this subroutine cannot be properly used to bring data from old star catalogs into the modern system, because old catalogs were compiled using a set of constants (in particular, the rate of precession) that are incompatible with modern values.

The IT = 2 and IT = 3 transformations involve the dynamical system, that is, the moving mean equator and equinox. The mean equator and equinox of J2000.0 was the most common reference system for modern astrometric catalog data before the ICRS was introduced in 1998. Now, catalog data is usually referred to the ICRS, which is a reference system fixed with respect to distant extragalactic objects, not defined by any Earth motions and with no associated date. The IT = 4 transformation is used to convert catalog quantities from the mean equator and equinox of J2000.0 (the "J2000.0 system") to the ICRS. The IT = 5 transformation is the opposite. The arguments DATE1 and DATE2 are ignored for these transformations.

Subroutine FRAME can be used to transform vectors from the J2000.0 system to the ICRS or vice versa.

See subroutine GETHIP for transforming Hipparcos catalog data to epoch J2000.0.

(Return to subroutine entry list)

GETHIP

Entry: CALL GETHIP (RAH, DECH, PMRAH, PMDECH, PARXH, RVH, RA2, DEC2, PMRA2, PMDEC2, PARX2, RV2)

Purpose: Converts Hipparcos catalog data at epoch J1991.25 to epoch J2000.0.

Arguments: DOUBLE PRECISION RAH, DECH, PMRAH, PMDECH, PARXH, RVH, RA2, DEC2, PMRA2, PMDEC2, PARX2, RV2

RAH:	Hipparcos right ascension in degrees (in).
DECH:	Hipparcos declination in degrees (in).
PMRAH:	Hipparcos proper motion in milliarcseconds/year (in).
PMDECH:	Hipparcos proper motion in declination in milliarcseconds/year (in).
PARXH:	Hipparcos parallax in milliarcseconds (in).
RVH:	Radial velocity at Hipparcos epoch in kilometers/second (in).
RA2:	ICRS right ascension at J2000.0 in hours (out).
DEC2:	ICRS declination at J2000.0 in degrees (out).
PMRA2:	ICRS proper motion in RA at J2000.0 in milliarcseconds/year (out).
PMDEC2:	ICRS proper motion in dec at J2000.0 in millarcseconds/year (out).
PARX2:	Parallax at J2000.0 in milliarcseconds (out).
RV2:	Radial velocity at J2000.0 in kilometers per second (out).

Discussion:

This subroutine takes Hipparcos catalog data, which is published for epoch J1991.25, and transforms it to epoch J2000.0 for use with NOVAS subroutines such as PLACE, APSTAR, TPSTAR, VPSTAR, etc. Note that the Hipparcos (input) right ascension is expressed in degrees, as in the catalog, while the J2000.0 (output) right ascension is given in hours, compatible with other NOVAS subroutines. Subroutine CATRAN (with IT = 1) is called internally to perform the epoch transformation. The reference frame for both input and output is the ICRS.

This subroutine should be used only for Hipparcos stars with linear space motion.

Radial velocity (RVH) is not given in the Hipparcos catalog and must be obtained from another source. If a value is not known, set RVH = 0.D0. The radial velocity is important for only a small number of nearby, high-proper-motion stars.

(Return to subroutine entry list)

APSTAR

Entry:	CALL APSTAR (TJD, N, RAI, DECI, PMRA, PMDEC, PARLAX, RADVEL, RA, DEC)
Purpose:	Computes the apparent place of a star at a specific time, given its position, proper motion, and other catalog data (at epoch J2000.0, with respect to ICRS).
Arguments:	INTEGER N DOUBLE PRECISION TJD, RAI, DECI, PMRA, PMDEC, PARLAX, RADVEL, RA, DEC

TJD:	TT Julian date for apparent place (in).
N:	Any integer. Formerly, the body identification number for the Earth — now just a place holder, no longer used (in).
RAI:	ICRS right ascension of star at J2000.0 in hours (in).
DECI:	ICRS declination of star at J2000.0 in degrees (in).
PMRA:	ICRS proper motion in RA at J2000.0 in milliarcseconds/year (in).
PMDEC:	ICRS proper motion in dec at J2000.0 in milliarcseconds/year (in).
PARLAX:	Parallax in milliarcseconds (in).
RADVEL:	Radial velocity in kilometers per second (in).
RA:	Apparent right ascension of star at time TJD, in hours, referred to the true equator and equinox of date (out).
DEC:	Apparent declination of star at time TJD, in degrees, referred to the true equator and equinox of date (out).

Discussion:

This subroutine computes the apparent place of a star for time TJD. The word “star” as used here refers to any object outside the solar system. For extragalactic objects, the input arguments PMRA, PMDEC, PARLAX, and RADVEL should be set to 0.D0. These input arguments should also be set to 0.D0 if the corresponding data are unknown or zero within the errors of measurement.

Efficiency is maximized when successive calls to APSTAR have the same value for TJD, since many quantities which are functions only of time are thereby saved and reused.

APSTAR works by calling PLACE with LOCATN = 0 and ICOORD = 1.

(Return to subroutine entry list)

TPSTAR

Entry:	CALL TPSTAR (UJD, GLON, GLAT, HT, RA, DEC)
Purpose:	Computes the topocentric place of a star at a specific time and observer location, using data from a previous call to subroutine APSTAR.
Arguments:	DOUBLE PRECISION UJD, GLON, GLAT, HT, RA, DEC

UJD:	UT1 Julian date, or the equivalent Greenwich apparent sidereal time in hours, for topocentric place (in).
GLON:	Observer's geodetic (WGS-84) longitude (east positive) in degrees (in).
GLAT:	Observer's geodetic (WGS-84) latitude (north positive) in degrees (in).
HT:	Observer's height above the Earth's reference ellipsoid (approximately, height above mean sea level) in meters.
RA:	Topocentric right ascension of star at time UJD, in hours, referred to the true equator and equinox of date (out).
DEC:	Topocentric declination of star at time UJD, in degrees, referred to the true equator and equinox of date (out).

Discussion:

This routine computes the topocentric place of a star (neglecting atmospheric refraction) for the location specified by the arguments GLON, GLAT, and HT, for time UJD. TPSTAR assumes that APSTAR has been previously called; it uses data from the previous APSTAR call in its computation. In particular, all information on the star is obtained from the prior APSTAR call.

The UT1 epoch that is specified here in argument UJD must correspond to the TT epoch previously supplied to APSTAR in its argument TJD. That is, the same instant must be specified for APSTAR in the TT time scale and for TPSTAR in the UT1 time scale. (The difference TT – UT1 is currently of order one minute.) Note that TPSTAR allows the value of the Greenwich apparent sidereal time, in hours, equivalent to the UT1 epoch, to be specified in the argument UJD instead of the UT1 Julian date.

Atmospheric refraction can be subsequently applied to RA and DEC by subroutine ZDAZ.

TPSTAR works by calling PLACE with LOCATN = 1 and ICOORD = 1.

(Return to subroutine entry list)

VPSTAR

Entry:	CALL VPSTAR (TJD, N, RAI, DECI, PMRA, PMDEC, PARLAX, RADVEL, RA, DEC)
Purpose:	Computes the virtual place (proper place) of a star at a specific time, given its position, proper motion, and other catalog data (at epoch J2000.0, with respect to ICRS).
Arguments:	INTEGER N DOUBLE PRECISION TJD, RAI, DECI, PMRA, PMDEC, PARLAX, RADVEL, RA, DEC

TJD:	TT Julian date for virtual place (in).
N:	Any integer. Formerly, the body identification number for the Earth — now just a place holder, no longer used (in).
RAI:	ICRS right ascension of star at J2000.0 in hours (in).
DECI:	ICRS declination of star at J2000.0 in degrees (in).
PMRA:	ICRS proper motion in RA at J2000.0 in milliarcseconds/year (in).
PMDEC:	ICRS proper motion in dec at J2000.0 in milliarcseconds/year (in).
PARLAX:	Parallax in milliarcseconds (in).
RADVEL:	Radial velocity in kilometers/second (in).
RA:	Virtual (or proper) right ascension of star at time TJD, in hours, referred to the GCRS (out).
DEC:	Virtual (or proper) declination of star at time TJD, in degrees, referred to the GCRS (out).

Discussion:

See the discussion for subroutine APSTAR. Subroutine VPSTAR is identical to APSTAR in input arguments and use. Here, however, the output arguments provide the virtual place (also called the proper place) of the star. The virtual place (proper place) is essentially the apparent place expressed in the GCRS.

VPSTAR works by calling PLACE with LOCATN = 0 and ICOORD = 0.

(Return to subroutine entry list)

LPSTAR

Entry:	CALL LPSTAR (UJD, GLON, GLAT, HT, RA, DEC)
Purpose:	Computes the local place of a star at a specific time and observer location, using data from a previous call to subroutine VPSTAR.
Arguments:	DOUBLE PRECISION UJD, GLON, GLAT, HT, RA, DEC

UJD:	UT1 Julian date, or the equivalent Greenwich apparent sidereal time in hours, for local place (in).
GLON:	Observer's geodetic (WGS-84) longitude (east positive) in degrees (in).
GLAT:	Observer's geodetic (WGS-84) latitude (north positive) in degrees (in).
HT:	Observer's height above the Earth's reference ellipsoid (approximately, height above mean sea level) in meters.
RA:	Local right ascension of star at time UJD, in hours, referred to the mean equator and equinox of J2000.0 (out).
DEC:	Local declination of star at time UJD, in degrees, referred to the mean equator and equinox of J2000.0 (out).

Discussion:

See the discussion for TPSTAR. LPSTAR is identical to TPSTAR in input arguments and use. Here, however, it is subroutine VPSTAR that must have been previously called and here the output arguments provide the local place of the star. The local place is essentially the topocentric place expressed in the "local GCRS".

LPSTAR works by calling PLACE with LOCATN = 1 and ICOORD = 0.

(Return to subroutine entry list)

ASSTAR

Entry: CALL ASSTAR (TJD, N, RAI, DECI, PMRA, PMDEC, PARLAX, RADVEL, RA, DEC)

Purpose: Computes the astrometric place of a star at a specific time, given its position, proper motion, and other catalog data (at epoch J2000.0, with respect to ICRS).

Arguments: INTEGER N
DOUBLE PRECISION TJD, RAI, DECI, PMRA, PMDEC, PARLAX, RADVEL, RA, DEC

TJD:	TT Julian date for astrometric place (in).
N:	Any integer. Formerly, the body identification number for the Earth — now just a place holder, no longer used (in).
RAI:	ICRS right ascension of star at J2000.0 in hours (in).
DECI:	ICRS declination of star at J2000.0 in degrees (in).
PMRA:	ICRS proper motion in RA at J2000.0 in milliarcseconds/year (in).
PMDEC:	ICRS proper motion in dec at J2000.0 in milliarcseconds/year (in).
PARLAX:	Parallax in milliarcseconds (in).
RADVEL:	Radial velocity in kilometers per second (in).
RA:	Astrometric right ascension of star at time TJD, in hours, referred to the ICRS (out).
DEC:	Astrometric declination of star at time TJD, in degrees, referred to the ICRS (out).

Discussion:

See the discussion for subroutine APSTAR. Subroutine ASSTAR is identical to APSTAR in input arguments and use. Here, however, the output arguments provide the astrometric place of the star in the ICRS.

ASSTAR works by calling PLACE with LOCATN = 0 and ICOORD = 3.

(Return to subroutine entry list)

APPLAN

Entry:	CALL APPLAN (TJD, L, N, RA, DEC, DIS)
Purpose:	Computes the apparent place of a planet or other solar system body at a specific time, using the barycentric ephemeris provided by subroutine SOLSYS.
Arguments:	INTEGER L,N DOUBLE PRECISION TJD, RA, DEC, DIS

TJD:	TT Julian date for apparent place (in).
L:	Body identification number for the planet or other body whose apparent place is desired; see argument M of subroutine SOLSYS (in).
N:	Any integer. Formerly, the body identification number for the Earth — now just a place holder, no longer used (in).
RA:	Apparent right ascension of body L at time TJD, in hours, referred to the true equator and equinox of date (out).
DEC:	Apparent declination of body L at time TJD, in degrees, referred to the true equator and equinox of date (out).
DIS:	True geometric distance between body L and the Earth at time TJD, in AU (out).

Discussion:

This subroutine computes the apparent place of a planet or other solar system body. Your choice of the version of subroutine SOLSYS in use determines the source of the body's barycentric rectangular coordinates used in the calculation.   The value of input argument L, which identifies the planet or other solar system body of interest, must correspond to argument M of SOLSYS.   The value of L can be obtained from a call to IDSS, e.g., L = IDSS ('SUN').

Efficiency is maximized when successive calls to APPLAN have the same value for TJD, because many quantities that are functions only of time are thereby saved and reused.

APPLAN works by calling PLACE with LOCATN = 0 and ICOORD = 1.

(Return to subroutine entry list)

TPPLAN

Entry:	CALL TPPLAN (UJD, GLON, GLAT, HT, RA, DEC, DIS)
Purpose:	Computes the topocentric place of a planet or other solar system body at a specific time and observer location, using data from a previous call to subroutine APPLAN.
Arguments:	DOUBLE PRECISION UJD, GLON, GLAT, HT, RA, DEC, DIS

UJD:	UT1 Julian date, or the equivalent Greenwich apparent sidereal time in hours, for topocentric place (in).
GLON:	Observer's geodetic (WGS-84) longitude (east positive) in degrees (in).
GLAT:	Observer's geodetic (WGS-84) latitude (north positive) in degrees (in).
HT:	Observer's height above the Earth's reference ellipsoid (approximately, height above mean sea level) in meters.
RA:	Topocentric right ascension of body at time UJD, in hours, referred to the true equator and equinox of date (out).
DEC:	Topocentric declination of body at time UJD, in degrees, referred to the true equator and equinox of date (out).
DIS:	True geometric distance between body and observer at time UJD, in AU (out).

Discussion:

This routine computes the topocentric place of a planet or other solar system body (neglecting atmospheric refraction) for the location specified by the arguments GLON, GLAT, and HT, for the time specified by the argument UJD. TPPLAN assumes that APPLAN has been previously called; it uses data from the previous APPLAN call in its computation. In particular, the solar system body is identified and its ephemeris is obtained in the prior APPLAN call.

The UT1 epoch that is specified here in argument UJD must correspond to the TT epoch previously supplied to APPLAN in its argument TJD. That is, the same instant must be specified for APPLAN in the TT time scale and for TPPLAN in the UT1 time scale. (The difference TT – UT1 is currently of order one minute.)   Note that TPPLAN allows the value of the Greenwich apparent sidereal time, in hours, equivalent to the UT1 epoch, to be specified in the argument UJD instead of the UT1 Julian date.

Atmospheric refraction can be applied to RA and DEC by subroutine ZDAZ.

TPPLAN works by calling PLACE with LOCATN = 1 and ICOORD = 1.

(Return to subroutine entry list)

VPPLAN

Entry: CALL VPPLAN (TJD, L, N, RA, DEC, DIS)

Purpose: Computes the virtual place (proper place) of a planet or other solar system body at a specific time, using the barycentric ephemeris provided by subroutine SOLSYS.

Arguments: INTEGER L, N
DOUBLE PRECISION TJD, RA, DEC, DIS

TJD:	TT Julian date for virtual place (in).
L:	Body identification number for the planet or other body whose virtual place is desired; see argument M of subroutine SOLSYS (in).
N:	Any integer. Formerly, the body identification number for the Earth — now just a place holder, no longer used (in).
RA:	Virtual (proper) right ascension of body L at time TJD, in hours, referred to the GCRS (out).
DEC:	Virtual (proper) declination of body L at time TJD, in degrees, referred to the GCRS (out).
DIS:	True geometric distance between body L and the Earth at time TJD, in AU (out).

Discussion:

See the discussion for subroutine APPLAN. Subroutine VPPLAN is identical to APPLAN in input arguments and use. Here, however, the output arguments provide the virtual place (also called the proper place) of the planet. The virtual place (proper place) is essentially the apparent place expressed in the GCRS.

VPPLAN works by calling PLACE with LOCATN = 0 and ICOORD = 0.

(Return to subroutine entry list)

LPPLAN

Entry: CALL LPPLAN (UJD, GLON, GLAT, HT,   RA, DEC, DIS)

Purpose: Computes the local place of a planet or other solar system body at a specific time and observer location, using data from a previous call to subroutine VPPLAN.

Arguments: DOUBLE PRECISION UJD, GLON, GLAT, HT, RA, DEC, DIS

UJD:	UT1 Julian date, or the equivalent Greenwich apparent sidereal time in hours, for local place (in).
GLON:	Observer's geodetic (WGS-84) longitude (east positive) in degrees (in).
GLAT:	Observer's geodetic (WGS-84) latitude (north positive) in degrees (in).
HT:	Observer's height above the Earth's reference ellipsoid (approximately, height above mean sea level) in meters.
RA:	Local right ascension of body at time UJD, in hours, referred to the local GCRS (out).
DEC:	Local declination of body at time UJD, in degrees, referred to the local GCRS (out).
DIS:	True geometric distance between body and observer at time UJD, in AU (out).

Discussion:

See the discussion for TPPLAN. LPPLAN is identical to TPPLAN in input arguments and use. Here, however, it is subroutine VPPLAN that must have been previously called and here the output arguments provide the local place of the planet. The local place is essentially the topocentric place expressed in the "local GCRS".

LPPLAN works by calling PLACE with LOCATN = 1 and ICOORD = 1.

(Return to subroutine entry list)

ASPLAN

Entry:	CALL ASPLAN (TJD, L, N, RA, DEC, DIS)
Purpose:	Computes the astrometric place of a planet or other solar system body at a specific time, using the barycentric ephemeris provided by subroutine SOLSYS.
Arguments:	INTEGER L, N DOUBLE PRECISION TJD, RA, DEC, DIS

TJD:	TT Julian date for astrometric place (in).
L:	Body identification number for the planet or other body whose virtual place is desired; see argument M of subroutine SOLSYS (in).
N:	Any integer. Formerly, the body identification number for the Earth — now just a place holder, no longer used (in).
RA:	Astrometric right ascension of body L at time TJD, in hours, referred to the ICRS (out).
DEC:	Astrometric declination of body L at time TJD, in degrees, referred to the ICRS (out).
DIS:	True geometric distance between body L and the Earth at time TJD, in AU (out).

Discussion:

See the discussion for subroutine APPLAN. Subroutine ASPLAN is identical to APPLAN in input arguments and use. Here, however, the output arguments provide the astrometric place of the planet in the ICRS.

ASPLAN works by calling PLACE with LOCATN = 0 and ICOORD = 3.

(Return to subroutine entry list)

PRECES

Entry: CALL PRECES (TJD1, POS1, TJD2, POS2)

Purpose: Precesses a position vector from the mean equator and equinox of an arbitrary date to the mean equator and equinox of J2000.0, or vice versa.

Arguments: DOUBLE PRECISION TJD1, POS1(3), TJD2, POS2(3)

TJD1:	TDB (or TT) Julian date of the first equatorial system (in).
POS1:	Position vector in the first equatorial system (mean equator and equinox of TJD1); units and origin arbitrary (in).
TJD2:	TDB (or TT) Julian date of the second equatorial system (in).
POS2:	Position vector in the second equatorial system (mean equator and equinox of TJD2); units and origin same as POS1 (out).

Discussion:

This subroutine precesses the input position vector, POS1, from the equator and equinox of TJD1 to the equator and equinox of TJD2; the resulting vector is POS2.

One of the two input Julian dates *mus*t be standard epoch J2000.0—either TJD1 or TJD2 must be 2451545.0D0 exactly. To precess a vector from one arbitrary date to another, call PRECES twice, using J2000.0 as the "middle" date. That is, in the first call, TJD1 = first Julian date, and TJD2 = 2451545.0D0; in the second call, TJD1 = 2451545.0D0, and TJD2 = second Julian date.

Formally, the current precession algorithm is a function of Barycentric Dynamical Time (TDB), but using TT as the basis for the input Julian dates results in a maximum error of only about 3×10^{-9} arcseconds, which is totally negligible. Standard epoch J2000.0, although formally defined in the TT time scale, is the same in the TT and TDB time scales to the precision given by double-precision Julian dates: at J2000.0, TT – TDB $\approx 10^{-4}$ second $\approx 10^{-9}$ day.

(Return to subroutine entry list)

EQECL

Entry: CALL EQECL (TJD, ICOORD, RA, DEC, ELON, ELAT)

Purpose: Converts right ascension and declination to ecliptic longitude and latitude.

Arguments: INTEGER ICOORD
DOUBLE PRECISION TJD, RA, DEC, ELON, ELAT

TJD:	TDB (or TT) Julian date of equator, equinox, and ecliptic used for coordinates (in).
ICOORD:	Coordinate system selection (in): ICOORD = 0: Mean equator and equinox of date ICOORD = 1: True equator and equinox of date
RA:	Right ascension in hours, referred to selected equator and equinox of date (in).
DEC:	Declination in degrees, referred to selected equator and equinox of date (in).
ELON:	Ecliptic longitude in degrees, referred to the ecliptic and selected equinox of date (out).
ELAT:	Ecliptic latitude in degrees, referred to the ecliptic of date (out).

Discussion:

This subroutine converts the equatorial position of an object into the equivalent ecliptic position: equatorial coordinates, RA and DEC, are converted to ecliptic coordinates, ELON and ELAT. This subroutine can be used for any kind of barycentric or geocentric coordinates—the conversion involves a simple rotation and should be regarded as just formalism. As in subroutine PRECES, the input Julian date can be based on either the TDB or TT time scales, with negligible resulting error.

RA and DEC can be expressed with respect to either the mean equator and equinox of date TJD (if ICOORD = 0) or the true equator and equinox of date TJD (if ICOORD = 1).

The representation of the ecliptic used for celestial coordinates is a smoothly moving mean plane described as part of the precession development. However, the mean and true equators intersect this ecliptic at different points. Therefore, the equinox, which serves as the origin of ecliptic longitude as well as the origin of right ascension, is different in the two cases. ELON will be expressed with respect to the same equinox as RA.

If TJD = 0.D0 and ICOORD = 0, the subroutine assumes RA and DEC are expressed with respect to the ICRS and provides ELON and ELAT with respect to the ecliptic and mean equinox of J2000.0.

See subroutines EQEC and ECEQ for the conversion of vectors between equatorial and ecliptic systems.

(Return to subroutine entry list)

CIORA

Entry: CALL CIORA (TJD, RACIO)

Purpose: Provides the right ascension of the Celestial Intermediate Origin (CIO).

Argument: DOUBLE PRECISION TJD, RACIO

TJD:	TT Julian date for CIO right ascension (in).
RACIO:	Right ascension of the CIO, in hours (+ or –), with respect to the true equinox of date (out).

Discussion:

This call supplies the true right ascension of the Celestial Intermediate Origin (CIO).

RACIO = – (equation of the origins) =

RACIO =Greenwich apparent sidereal time – ERA

where all quantities are expressed in hours.

EROT

Entry: CALL EROT (TJDH, TJDL, THETA)

Purpose: Provides the Earth Rotation Angle (ERA), θ.

Argument: DOUBLE PRECISION TJDH, TJDL, THETA

TJDH:	UT1 Julian date for ERA, high-order part (in).
TJDL:	UT1 Julian date for ERA, low-order part (in).
THETA:	ERA in degrees (out).

Discussion:

This call supplies the ERA, θ, which is the geocentric angle, in the instantaneous equatorial plane (true equator), between the directions toward the Terrestrial Intermediate Origin (TIO) and the Celestial Intermediate Origin (CIO).

This subroutine allows for the input UT1 time to be represented as a split Julian date. See the discussion in the description of subroutine SIDTIM.

(Return to subroutine entry list)

HIACC

Entry: CALL HIACC

Purpose: Specifies that Earth orientation calculations are to be performed at full (IAU standard) accuracy.

Arguments: (None)

Discussion:

This routine instructs NOVAS to perform Earth orientation calculations, in particular, the evaluation of the nutation series and the equation of the equinoxes, at full accuracy according to recommended IAU models. This is the default; normally it would only be necessary to call HIACC to resume full accuracy calculations after a call to LOACC.

LOACC

Entry: CALL LOACC

Purpose: Specifies that Earth orientation calculations are to be performed at reduced accuracy.

Arguments: (None)

Discussion:

This routine instructs NOVAS to perform Earth orientation calculations, in particular the evaluation of the nutation series and the equation of the equinoxes, at reduced accuracy. Even at reduced accuracy, the calculations are good to 0.1 milliarcsecond or better if the standard set of NOVAS subroutines is used. The reduced accuracy mode will provide noticeably faster performance for many types of calculations while still providing overall angular accuracy that is adequate for most applications. To resume full accuracy calculations, call HIACC.

If the alternative version of subroutine NOD is in use (see installation instructions in section 2.6), a call to LOACC will result in a significant improvement in performance (up to a factor of 100 for some calculations), although angular errors of up to 0.05 arcsecond may arise.

(Return to subroutine entry list)

GETVEC

Entry: CALL GETVEC (UVEC)

Purpose: Provides the unit vector corresponding to the last-computed object coordinates.

Argument: DOUBLE PRECISION UVEC(3)

UVEC:	Unit vector corresponding to last-computed object position, expressed in same reference system as the scalar coordinates (out).

Discussion:

This call supplies the unit vector that corresponds to the last object position computed by any NOVAS subroutine. It provides a simple way to obtain the vector that is equivalent to a pair of spherical coordinates, for example, right ascension and declination. It would normally be used following a call to one of the NOVAS subroutines that return only scalar results (APSTAR, APPLAN, EQECL, etc.). The reference system in which the vector is expressed is the same as for the previously calculated spherical coordinates.

(Return to subroutine entry list)

SETDT

Entry:	CALL SETDT (DELTAT)
Purpose:	Specifies the value of ΔT = TT – UT1 for subsequent internal use by NOVAS subroutines.
Argument:	DOUBLE PRECISION DELTAT
	DELTAT: Value of ΔT in seconds (in).

Discussion:

This routine allows for the specification of ΔT, the difference in seconds between the TT and UT1 time scales, to be used by NOVAS in subsequent calculations. The value of ΔT that you specify by a call to SETDT will be used by various NOVAS subroutines until you explicitly change it by another call.

Generally, the value of DELTAT needs to be accurate to only about one second, which means that a single call can be used for about a year's span of dates. The only exception is for the computation of accurate topocentric positions of bodies in the inner solar system (especially the Moon); see the note on this in the description of subroutine PLACE.

(Return to subroutine entry list)

CELPOL

Entry:	CALL CELPOL (TJD, ITYPE, DPOLE1, DPOLE2)
Purpose:	Allows for the specification of celestial pole offsets for high-precision applications.
Arguments:	INTEGER ITYPE DOUBLE PRECISION TJD, DPOLE1, DPOLE2

TJD:	TDB or TT Julian date for celestial pole offset values (in).

ITYPE:	Type of pole offsets (in):
	ITYPE = 1: dψ and dε (ecliptic angles)
	ITYPE = 2: dX and dY (GCRS angles)

DPOLE1:	Pole offset 1 (dψ or dX) in milliarcseconds (in).
DPOLE2:	Pole offset 2 (dε or dY) in milliarcseconds (in).

Discussion:

This routine is actually an entry point into subroutine ETILT that allows for the specification of celestial pole offsets for very high precision applications. The offsets describe the observed position of the Celestial Intermediate Pole (CIP) with respect to the position computed from the standard precession and nutation models. The offsets are subsequently applied as corrections to the nutation in longitude and nutation in obliquity within ETILT. Thus, ETILT's output arguments OBLT, EQEQ, DPSI, and DEPS will be affected. Because other NOVAS subroutines, such as SIDTIM, call ETILT to obtain data related to the Earth's orientation in space, the celestial pole offsets specified here are propagated through the data that the various NOVAS subroutines provide.

Daily values of the celestial pole offsets are published, for example, in IERS[27] Bulletins A and B. The celestial pole offsets effectively correct for errors or incompleteness in the standard precession or nutation models. If you use CELPOL, make sure it is called before any other routines for a given date. Values of the pole offsets that you specify by a call to CELPOL will be used by ETILT until you explicitly change them.

Important: For compatibility with the NOVAS version 3.0 precession and nutation models, specify ITYPE = 2 and use *only* IERS dX and dY values with respect to "IAU 2000A" (sometimes labeled "IAU 2000"). These pole offset values will generally not exceed 0.5 milliarcsecond and therefore CELPOL would need to be called only when very high accuracy is required.

(Return to subroutine entry list)

[27] http://www.iers.org/MainDisp.csl?pid=36-9

ETILT

Entry: CALL ETILT (TJD,   OBLM, OBLT, EQEQ, DPSI, DEPS)

Purpose: Computes quantities related to the orientation of the Earth's rotation axis at a given time.

Arguments: DOUBLE PRECISION TJD, OBLM, OBLT, EQEQ, DPSI, DEPS

TJD:	TDB (or TT) Julian date for requested data (in).
OBLM:	Mean obliquity of the ecliptic at time TJD, in degrees (out).
OBLT:	True obliquity of the ecliptic at time TJD, in degrees (out).
EQEQ:	Equation of the equinoxes at time TJD, in time seconds (out).
DPSI:	Nutation in longitude at time TJD, in arcseconds (out).
DEPS:	Nutation in obliquity at time TJD, in arcseconds (out).

Discussion:

This subroutine computes various quantities related to the orientation of the Earth's rotation axis (vector toward Celestial Intermediate Pole) with respect to the ecliptic plane at a specific time. The computation involves a call to subroutine NOD to evaluate the nutation series.

The output values of the last four arguments will correctly reflect the celestial pole offsets if routine CELPOL has previously been called.

(Return to subroutine entry list)

SOLSYS

Entry:	CALL SOLSYS (TJD,M,K, POS,VEL,IERR)	
Purpose:	Provides the position and velocity vectors of a planet or other solar system body at a specific time. The origin of coordinates may be either the barycenter of the solar system or the center of mass of the Sun.	
Arguments:	INTEGER M, K, IERR DOUBLE PRECISION TJD, POS(3), VEL(3)	
	TJD:	TDB Julian date for planet position and velocity (in).
	M:	Body identification number for the planet or other solar system body whose position and velocity are desired (in).
	K:	Origin selection code (in): K = 0 for origin at solar system barycenter K = 1 for origin at center of mass of Sun
	POS:	ICRS position vector of body M at time TJD, in AU (out).
	VEL:	ICRS velocity vector of body M at time TJD, in AU/day (out).
	IERR:	Return code (out) IERR = 0 is normal exit, everything OK IERR = 1 means TJD before first possible date IERR = 2 means TJD after last possible date IERR = 3 means some other problem, e.g., invalid value for M, etc.

Discussion:

This subroutine supplies values for the components of the position vector POS and velocity vector VEL for body M at time TJD. The vectors computed by SOLSYS are expressed with respect to ICRS axes, in the BCRS metric. The vectors are barycentric if K = 0 and heliocentric if K = 1.

There are several versions of SOLSYS, each with its own internal logic. Some use internally stored data or series expansions; others refer to external data files. Additional documentation (see below) is usually required for the proper use of each version. You are, of course, free to supply your own version(s), providing that the arguments conform to the above specifications.

The values of the body identification number, M, will in general differ from one SOLSYS version to another; consult the documentation for the specific version in use. Usually, M = 1 refers to Mercury, M = 2 refers to Venus, M = 3 refers to the Earth, etc., but the identification numbers for bodies such as the Sun or Moon differ across implementations. Furthermore, some versions of SOLSYS support only a subset of the major solar system bodies. The minimum requirement is support for the Sun and Earth. Here, "Earth" refers to the geocenter and not the Earth/Moon barycenter.

Each version of SOLSYS is packaged together with a function called IDSS, which provides the body identification number, given a solar system body's name; for example, M = IDSS ('MARS'). See the description of IDSS immediately following.

If you supply your own version of SOLSYS, you must also supply a corresponding version of IDSS.

Some versions of SOLSYS support "split" Julian dates across successive calls. That is, they recognize $-1.D0 \le \text{TJD} \le 1.D0$ as representing a fraction of a day that is to be paired with the integral part of the Julian date from a preceding call. This allows the fraction of the day to be specified to much greater precision than is possible in a single call. For example, suppose you call SOLSYS with TJD equal to d_1+f_1, an ordinary Julian date—d_1 is the Julian day count and f_1 is the fraction of a day. If the time of day needs to be specified more precisely (equivalent to better than 0.1 millisecond), you can simply make a second call to SOLSYS with TJD equal to the fraction of a day f_2; without the day count included in TJD, f_2 can be given to much more precision than f_1. In the second call, SOLSYS would interpret the Julian date that you want as d_1+f_2, with the extra precision in f_2 preserved. (It does not matter whether the fractions f_1 and f_2 are similar.) If you continue to call SOLSYS with fractional TJD values (f_3, f_4, …), it will interpret them in the same way (paired with the same integer day number d_1) until you again set TJD to be a full Julian date. You can test whether the version of SOLSYS in use supports this option: if IDSS ('JD') equals 2, the split Julian date option is supported, otherwise not.

Example of split Julian dates: Suppose IDSS('JD') = 2, indicating that the version of SOLSYS in use supports split Julian dates in successive calls. Then, in the following sequence of calls, the requested Julian date is interpreted as shown:

	Effective Julian date of data returned:
CALL SOLSYS (2462387.1234567D0, …)	2462387.1234567
CALL SOLSYS (0.98765432109876543D0, …)	2462387.98765432109876543
CALL SOLSYS (0.6624965422153D0, …)	2462387.6624965422153
CALL SOLSYS (2471932.7892D0, …)	2471932.7892
CALL SOLSYS (–0.1D0, …)	2471931.9

However, if IDSS('JD') = 1, the split Julian date option is not supported. In that case, in the second, third, and fifth calls above, the TJD values would be interpreted as complete Julian dates, which would correspond to calendar dates in early 4713 B.C. This would in most cases lead to an IERR = 1 return.

Documentation follows for the following versions of SOLSYS:

Version 1	reads 1-day planetary coordinate file
Version 2	provides interface to JPL ephemeris software
Version 3	evaluates self-contained Earth and Sun

(Return to subroutine entry list)

Integer Function
IDSS

Entry: IDSS (NAME)

Purpose: Returns value of the body identification number for the named solar system body.

Argument: CHARACTER*(*) NAME

NAME:	Name of solar system body, in all capital letters (in).

Function Value Out: INTEGER IDSS

Value of the body identification number for solar system body named, or -9999 if the body is not supported or the name cannot be recognized.

Discussion:

This function provides the body identification number for the named body, for use as an argument to NOVAS subroutines such as SOLSYS, APPLAN, etc. The body name should be all caps and can be any length; most versions of this subroutine use only the first three or four letters.

A version of this function is supplied with each version of subroutine SOLSYS. SOLSYS defines the body identification numbers to be used.

If NAME is 'JD', the returned function value indicates whether the version of SOLSYS in use supports split Julian dates in successive calls. In this case, a value of 1 indicates that split Julian dates are not supported, and a value of 2 indicates that they are supported.

(Return to subroutine entry list)

SOLSYS Version 1

This version of SOLSYS reads and interpolates a formatted file of planetary coordinates at one-day intervals. This ephemeris file must have the following characteristics:

The first record in the file is a header record of no specific format. The header is followed by data records. Each data record must be formatted (plain-text ASCII) and must contain the barycentric (BCRS) rectangular coordinates of the Sun and planets for a specific date. The coordinates are given in AU, with respect to the ICRS axes. The data records must be at 1-day intervals of TDB, with dates increasing. The format for each data record is (F10.2, *n*(3F16.12)), where *n* is the number of bodies with coordinates in the file (data for at least the Sun and Earth must be given, so *n* must be at least 2). The order of quantities in the data records is:

(1) TDB Julian date
(2) Sun X
(3) Sun Y
(4) Sun Z
(5) First planet X
(6) First planet Y
(7) First planet Z
(8) Second planet X
(9) Second planet Y
:
etc.

The Sun's coordinates must always be given first in the record. The body identification numbers will be M = 0 for the Sun, M = 1 for the first planet, M = 2 for the second planet, etc. The Earth (geocenter, not the Earth-Moon barycenter) must be one of the planets.

This version of SOLSYS reads the ephemeris file, converts the barycentric coordinates to heliocentric coordinates if required, computes velocities, and performs the necessary interpolation (7-point Lagrangian interpolation) to supply the correct position and velocity vector components for any arbitrary value of the time argument TJD. TJD values must, of course, be confined to within the span of time represented by the coordinate file. The file-positioning logic in this version is such that values of TJD may, in successive calls to this subroutine, be in any order; but efficiency is maximized if the TJD values are in ascending order in successive calls.

A one-day ephemeris file named SS_EPHEM.TXT, conforming to these specifications, is provided as part of the NOVAS Fortran distribution. The file contains the coordinates of 11 bodies (Sun, eight planets, Pluto, and the Moon), taken from the JPL DE405 ephemeris, and covers years 2000 to 2020, inclusive (see sections 2.3 and 2.4). The file is read on Fortran logical unit 20. The body identification numbers to be used with it are: Sun, M=0; Mercury, M=1; Venus, M=2; Earth, M=3; Mars, M=4; Jupiter, M=5; Saturn, M=6; Uranus, M=7; Neptune, M=8; Pluto, M=9; Moon, M=10. (Use function IDSS to obtain these numbers.)

The interpolation errors for most bodies are less than 1 meter in position and 1 millimeter per second in velocity. The maximum interpolation error of the Earth's position can reach 6 meters. These errors are all negligible. For Mercury, however, the interpolation errors in position and velocity can reach 280 m and 10 mm/s, respectively; and for the Moon, the errors can reach 480 m and 18 mm/s. The position errors for Mercury and the Moon can affect angular coordinates computed by NOVAS by as much as 0.0008 arcsecond for Mercury and 0.3 arcsecond for the Moon, although typical errors will be much less (the average error is about 1/6 the maximum error). Since these errors are due to interpolation, they will be similar even if another ephemeris file is used.

You are free to supply your own ephemeris file as long as it conforms to the above specifications. You can change the file name, unit number on which it is read, number of bodies in each record, and the format used to read each record by calling subroutine FILDEF, which is packaged with SOLSYS version 1 (see code). If you change the ephemeris file, you may also have to modify function IDSS.

SOLSYS version 1 does not support split Julian dates in successive calls.

SOLSYS version 1 has an additional error code: IERR = 4 means that there was a problem opening the ephemeris file.

(Return to subroutine entry list)

SOLSYS Version 2

This version of SOLSYS reads a planetary/lunar ephemeris file from the Jet Propulsion Laboratory and supplies barycentric or heliocentric positions and velocities. This version calls JPL subroutines CONST and DPLEPH, which in turn call several other JPL routines. It is the JPL subroutines that read, decode, and evaluate the ephemeris data; SOLSYS version 2 simply provides a "front end" for the JPL ephemerides. You are responsible for obtaining both the JPL ephemeris file and the corresponding JPL subroutines (see Appendix C). This version requires the JPL package of 1997, which supports the JPL planetary/lunar ephemerides[28] DE200, DE405, and DE406. DE405 and DE406 provide ICRS compatibility.

The body identification numbers to be used with this version are: Sun, M=10; Mercury, M=1; Venus, M=2; Earth, M=3; Mars, M=4; Jupiter, M=5; Saturn, M=6; Uranus, M=7; Neptune, M=8; Pluto, M=9; Moon, M=11. (Use function IDSS to obtain these numbers.)

Body identification numbers outside the range 1–11 will result in a call from SOLSYS to subroutine AUXPOS. AUXPOS is meant to supply positions of "auxiliary" solar system bodies, such as asteroids or comets. A dummy version of AUXPOS is provided, but you can substitute a working version of your own. This would allow you to obtain, for example, apparent places of these other bodies simply by using a body identification number outside the range 1–11 when calling subroutines such as PLACE or APPLAN. The specifications for AUXPOS are given below. If this is of no interest, just leave the dummy version of AUXPOS in place.

SOLSYS version 2 supports split Julian dates in successive calls. If you supply a working version of AUXPOS, it does not need to support split Julian dates.

(Return to subroutine entry list)

[28] The current version of JPL planetary/lunar ephemerides and accompanying software is available at http://ssd.jpl.nasa.gov/?planet_eph_export

Specifications for Subroutine
AUXPOS

User-supplied subroutine called by SOLSYS version 2 (dummy version is provided)

Entry: CALL AUXPOS (TJD, M, K,   POS, VEL, JERR)

Purpose: Provides the position and velocity vectors of a planet, asteroid, comet, or other solar system body at a specific time. The origin of coordinates may be either the barycenter of the solar system or the center of mass of the Sun. Called from SOLSYS version 2 when the body identification number passed to SOLSYS is outside the range 1–11.

Arguments: INTEGER M, K, JERR
DOUBLE PRECISION TJD, POS(3), VEL(3)

TJD:	TDB Julian date for auxiliary body position and velocity (in).
M:	Body identification number for the auxiliary body whose position and velocity are required (in).
K:	Origin selection code (in): K = 0 sets origin at solar system barycenter K = 1 sets origin at center of mass of Sun
POS:	ICRS position vector of body M at time TJD, in AU (out).
VEL:	ICRS velocity vector of body M at time TJD, in AU/day (out).
JERR:	Return code (out): IERR = 0 is normal exit, everything OK IERR = 1 means TJD before first possible date IERR = 2 means TJD after last possible date IERR = 3 means some other problem, e.g., invalid value for M, etc.

Discussion:

This subroutine can be used to provide the positions of asteroids, comets, trans-neptunian objects, or even spacecraft. It can also be used to provide an alternative to the JPL ephemeris for any of the major planets. Because AUXPOS is called only from SOLSYS version 2 (when necessary), use of AUXPOS is transparent to supervisory-level NOVAS subroutines.

The values of the arguments TJD, M, and K passed to this routine are identical to those passed to SOLSYS. However, this routine will be called only when M < 1 or M > 11. Subroutine AUXPOS should be coded in such a way that it can efficiently supply the vectors POS and VEL for random values of TJD, since the values of TJD in successive calls to AUXPOS will not necessarily be in ascending order. AUXPOS does *not* need to support split Julian dates in successive calls.

The supervisory-level NOVAS subroutines that call SOLSYS, such as PLACE or APPLAN, require velocity information only for the Earth. Positions of solar system bodies required by NOVAS subroutines are all barycentric. Therefore, although the argument list for AUXPOS is given above in its most general form, for calls originating from within NOVAS (via SOLSYS), K = 0 and POS is the only vector that needs to be returned.

The internal logic of this routine is entirely your choice. The components of the vectors POS and VEL may be read in from an external file, calculated internally from orbital elements, computed from the evaluation of Fourier or Chebyshev series, interpolated from an internally stored array, or obtained in any other way. You can also assign values of M (outside the range 1–11) to specific objects as you like.

If you supply you own version of AUXPOS, you might consider also replacing the version of IDSS that is packaged with SOLSYS version 2 so that IDSS could supply the body identification numbers for all the bodies that are supported by SOLSYS and AUXPOS.

(Return to subroutine entry list)

SOLSYS Version 3

This version of SOLSYS provides the position and velocity of the Earth or Sun without reference to any external data file. The heliocentric position and velocity of the Earth are computed by evaluating a trigonometric series that is a truncated and modified version of Newcomb's Theory of the Sun. When barycentric positions and velocities are required, a number of additional approximations are involved; therefore, barycentric positions and velocities computed by this version of SOLSYS are less accurate than heliocentric positions and velocities. This version of SOLSYS produces data within the following error limits (compared to the JPL DE405 ephemeris) for dates within two centuries of J2000.0:

Earth heliocentric positions:	650 km
Earth heliocentric velocities:	0.2 m/s
Earth barycentric positions:	5000 km
Earth barycentric velocities:	0.4 m/s

The maximum errors in the Sun's barycentric position and velocity are similar to those of the Earth, although of course the Sun's heliocentric position and velocity are identically zero.

When this version of SOLSYS is used in the computation of the apparent place of the Sun, it will contribute not more than 1 arcsecond error (usually less than 0.5 arcsecond error). When this version of SOLSYS is used in the computation of apparent places of stars, it will contribute not more than 0.3-milliarcsecond error.

The errors in positions and velocities from this version of SOLSYS will increase gradually outside of the four-century span, centered on J2000.0, for which it was optimized. This version of SOLSYS will return IERR = 1 for Julian dates prior to 2340000.5 (August 1694) and IERR =2 for Julian dates after 2560000.5 (December 2296).

This version of SOLSYS does not support split Julian dates in successive calls.

This version of SOLSYS calls subroutine PRECES, and certain expressions in the SOLSYS algorithm have been adjusted to conform to the IAU 2006 precession.

The body identification numbers to be used with this version are: Sun, M=0 or M=1; and Earth, M=2 or M=3.

(Return to subroutine entry list)

(Return to Table of Contents)

Chapter 5 Equinox- and CIO-Based Paradigms Compared

5.1 Computing Hour Angles

The equinox- and CIO-based celestial reference systems are part of two computational schemes for accounting for the Earth's instantaneous orientation with respect to the stars. These two methods represent the same phenomena (as they obviously must) but in slightly different order. The overall matrix that embodies, for a given instant, the terrestrial-to-celestial (ITRS-ICRS) transformation is the same for both schemes. Therefore, the value of observable quantities will not be affected by the choice of which paradigm is used for the computations.

Therefore, in NOVAS, quantities such as declination and hour angle, which are in principle measurable angles, should have the same values regardless of the way in which they are computed. Since both the equinox-based and CIO-based paradigms are based on the instantaneous (true) equator of date—the plane orthogonal to the Celestial Intermediate Pole (CIP)—declinations are, in fact, completely unaffected.

How are hour angles of celestial objects computed in the old and new paradigms? Assume that we are considering Greenwich hour angles, that is, hour angles measured from the meridian of geodetic longitude zero (the X-Z plane of the ITRS), without polar motion. In the equinox-based scheme, we compute the topocentric place of the object of interest with respect to the true equator and equinox of date. Then we compute Greenwich apparent sidereal time and subtract the object's apparent right ascension to form the hour angle. In the CIO-based scheme, we compute the object's topocentric place with respect to the true equator and CIO of date. To form the hour angle, we compute the ERA and subtract the CIO-based right ascension (also called the intermediate right ascension). The two results should be identical.

The following table summarizes the two equivalent procedures for hour angle and the NOVAS subroutines that would be used for each, assuming that polar motion is neglected. The procedures outlined here provide the Greenwich hour angle (GHA) of a star; only the first step would be different for a solar system body.

	Equinox-Based Method	CIO-Based Method
Use subroutine	APSTAR followed by TPSTAR — or — PLACE with OBJECT = ′STAR′, LOCATN = 1, and ICOORD = 1	PLACE with OBJECT = ′STAR′, LOCATN = 1, and ICOORD = 2
… to obtain	RA and DEC topocentric right ascension and declination of the star with respect to the true equator and equinox of date (in hours and degrees, respectively)	RA and DEC topocentric right ascension and declination of the star with respect to the true equator and CIO of date (in hours and degrees, respectively)
Then use subroutine	SIDTIM with K = 1	EROT
… to obtain	GST Greenwich apparent sidereal time (in hours)	THETA Earth Rotation Angle, θ (in degrees)
Compute Greenwich hour angle	GHA = GST – RA (in hours)	GHA = THETA / 15.D0 – RA (in hours)

The computed GHA may have to be reduced to the range -12^h to $+12^h$. Subroutines APSTAR and PLACE require time arguments in the TT time scale, while TPSTAR, SIDTIM, and EROT require time arguments in the UT1 time scale. The two procedures should yield the same value of GHA to within a microarcsecond around the present time and identical values for DEC. To obtain the local hour angle in either method, simply add to the GHA the observer's longitude (east positive) in appropriate units.

The common notion of hour angle becomes somewhat problematic when polar motion is taken into account, because what we usually regard as the Greenwich (or observer's) meridian—a plane of constant geodetic longitude—is not, in general, parallel to an hour circle on the celestial sphere when the geodetic pole and the CIP are not coincident. See the discussion in section 6.5.4 of USNO Circular 179.[29]

5.2 Other Computational Considerations

Two high-level NOVAS subroutines that involve Earth rotation, SIDTIM and TERCEL, can perform their internal calculations using either the equinox-based paradigm or the CIO-based paradigm. As previously mentioned, SIDTIM computes sidereal time. TERCEL performs the terrestrial-to-celestial transformation. (Note: ZDAZ is indirectly involved because it calls TERCEL.) The method used within either routine is selected by a prior call to either EQINOX or CIOTIO (without any arguments), which remains in effect until changed by the

[29] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

opposite call. Because there is no external difference in how SIDTIM or TERCEL are used, and the two computational paradigms yield answers that are consistent within a few micro-arcseconds over many centuries, there is seldom a practical basis for a choice. However, the equinox method is much more efficient if mean sidereal time is to be computed. The equinox-based paradigm is the default, that is, it is used unless CIOTIO has been called. That will, of course, be the case for any existing programs that are not updated to make this choice explicit.

Finally, another of the new Earth-rotation-related subroutines is worth mentioning. For a given TT Julian date, CIORA provides the right ascension of the CIO with respect to the true equinox of date. With a sign reversal, this quantity is the equation of the origins, the direction of the true equinox measured in the equator eastward (+) from the CIO. Because the equinox and CIO are simply different right ascension origins on the instantaneous equator, CIORA provides the angular difference between the origins of these two systems. The equation of the origins is also the difference, expressed as an angle, between the ERA and Greenwich apparent sidereal time.

5.3 How NOVAS Implements the CIO-Based Paradigm

The equinox-based paradigm is, of course, the historical basis for NOVAS. One of its key pieces is the precession algorithm (implemented by PRECES), which uses the equinox as its azimuthal coordinate; that is, it transforms celestial coordinates from the mean equator and equinox of one date to the mean equator and equinox of another date. Even though the recommended precession formulation has been replaced twice over the last half-century, this aspect of it has remained unchanged. Another key piece is the algorithm for sidereal time (implemented by SIDTIM), which is based on a sidereal day that is defined by successive transits of the equinox. The sidereal time formula must always be matched to the precession algorithm, since mean sidereal time must account for the precession of the equinox in right ascension; this has been consistently done in NOVAS.

To use the CIO-based paradigm, we must know where the CIO is in some well-defined coordinate system. Unlike the equinox, the CIO is not defined by static geometry but by its motion, so its position at any time is given by the result of an integral that has been evaluated either analytically or numerically. In NOVAS, both results are available: the position of the CIO can be taken from an external file that is the output of a numerical integration, or it can be obtained from an analytical expression for the equation of the origins (see sections 6.5.1.1 and 6.5.1.2 of USNO Circular 179[30]).

The NOVAS implementation of the CIO-based Earth rotation paradigm for a given date is based on the construction of the Celestial Intermediate Reference System for that date, using vectors toward the Celestial Intermediate Pole (CIP) and the Celestial Intermediate Origin (CIO). These two directions define, respectively, the z-axis and x-axis of the celestial intermediate system. The direction toward the CIP in the GCRS can be computed by passing

[30] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

the vector (0,0,1) through subroutines NUTATE, PRECES, and FRAME in succession. Given the direction of the CIP, the other piece of required information is the location of the CIO for the same date, which is provided by CIOLOC, described below. The basis vectors of the intermediate system, with respect to the GCRS, are computed by CIOBAS (see section 6.5.1 of USNO Circular 179 for the algorithms). Having these basis vectors available allows NOVAS to easily transform any vector in the GCRS to the intermediate system. The only other quantity used in the CIO-based paradigm is the ERA, which is trivial to compute and provided by EROT.

Subroutine CIOLOC obtains the location of the CIO for a given date in one of two ways, and an output argument, K, indicates which way was used. If an external file of CIO right ascension values is available (nominally called CIO_RA.TXT and located in the current directory) then CIOLOC will provide the GCRS right ascension of the CIO, and will set K to 1. If this file is not available, then CIOLOC will provide the true right ascension of the CIO (the arc on the instantaneous equator from the equinox to the CIO), obtained from a series expansion, and will set K to 2. CIOBAS can work with either coordinate of the CIO. The two methods are equivalent within several microarcseconds over six centuries centered on the year 2000 (it is not clear which is more correct).

To do the hard work, CIOLOC calls either CIORD (for K = 1) or EQXRA (for K = 2). CIOLOC always initially calls CIORD to see if the external file of CIO right ascensions is present. If it is, CIORD reads and interpolates the file, which is the output from a numerical integration covering years 1700 to 2300 and directly provides the right ascension of the CIO in the GCRS. You can specify the path/name of this file, its type (sequential or direct-access), and the logical unit number on which it is to be read, by using a call to CIOFIL, which must precede any CIO-based computation. If you don't call CIOFIL, CIORD will look for a formatted sequential file named CIO_RA.TXT in the current directory (folder) and, if present, will read it on logical unit 24. A copy of CIO_RA.TXT (7.5 Mbytes) is provided as part of the NOVAS Fortran distribution, along with a utility program called CIO_file.f to convert it to a binary direct-access file (2.9 Mbytes) if desired (see sections 2.2, 2.4, and 2.5).

If the file is not present, then CIOLOC calls EQXRA to evaluate the equation of the origins from a closed-form expression that includes the evaluation of nutation in longitude, a lengthy series of trigonometric terms. The result locates the CIO with respect to the equinox on the instantaneous equator.

Note that at no point does NOVAS use the CIO locator, *s*, which is described in IERS documents and *The Astronomical Almanac*.

(Return to Table of Contents)

Appendix A Overview of How NOVAS Has Changed

A detailed list of the changes in the NOVAS Fortran code from the previous version (F2.0 of 1998) is given in Appendix B. The following paragraphs are meant to provide some perspective on these changes for people who are already familiar with NOVAS. Most of the modifications have been made in response to resolutions passed by the IAU in 2000 and 2006 that recommended new models for fundamental astronomy, within a new conceptual framework. To the greatest extent possible, the calling sequences for the highest-level (and most used) functions from the previous versions of NOVAS have been preserved—but there are a few important exceptions. There are many new calls.

A.1 Important Changes in Calls

Probably the most important change to existing NOVAS calls is the change of proper motion and parallax units in the calls to APSTAR, VPSTAR, and ASSTAR, CATRAN, and GETHIP, all of which deal with star positions. The units have been changed as follows:

proper motion in RA:	from seconds of RA/century to milliarcseconds/ year
proper motion in dec:	from arcseconds per century to milliarcseconds/year
parallax:	from arcseconds to milliarcseconds

These changes have been made to conform to the units used in most modern star catalogs (e.g., Hipparcos, Tycho-2, or the FK6), which in turn follow from the observational techniques now used in the construction of such catalogs. Obviously, star data previously used with NOVAS must either be replaced or transformed. The transformation equations from “old” to “new” units are as follows:

```
PMRNEW = PMROLD * 150.D0 * DCOS (DEC0 * DEGRAD) ; proper motion, RA
PMDNEW = PMDOLD * 10.D0                         ; proper motion, dec
PAXNEW = PAXOLD * 1000.D0                       ; parallax
```

where DEC0 is the catalog declination (J2000.0 or ICRS) of the star in degrees and DEGRAD is the degrees-to-radians conversion factor (0.01745329…).

The other major change to a high-level subroutine is that PNSW has been renamed to TERCEL (it carries out the terrestrial-to-celestial transformation), with a change to the time argument. All other changes to existing NOVAS calls involve lower-level routines not frequently invoked by most users; these are detailed in Appendix B.

A.2 PLACE: A New General-Purpose “Place” Subroutine

All computational code to compute apparent, topocentric, virtual, astrometric, etc., places of stars or planets has now been consolidated into a single new subroutine call PLACE. The familiar calls to APSTAR, APPLAN, TPSTAR, etc., still work as before but are now just "front-ends" to PLACE. This change eliminated much duplicate code and also provides more flexibility and possible future additions (such as binary star orbits or nonlinear terms in proper motion). PLACE can also provide star or planet positions within the Celestial Intermediate Reference System that is part of the new paradigm for Earth rotation calculations (see below). PLACE provides its output position both in spherical coordinates (right ascension, declination, and, for solar system bodies, geometric distance) and as a unit vector. PLACE also provides radial velocity. PLACE accepts the specification of solar system bodies by name, e.g., 'MARS', 'SATURN', or 'SUN', thus increasing the readability of code. You may want to consider changing your calls to APSTAR, APPLAN, etc., to the equivalent calls to PLACE.

A.3 New Reference Systems

The IAU resolutions of 2000 defined several new reference systems for fundamental astronomy. These are described in section 1.1 of this document and in Chapters 1, 3, and 6 of USNO Circular 179.[31] Briefly, they are:

> Barycentric Celestial Reference System (BCRS) – Replaces the barycentric system based on the mean equator and equinox of J2000.0. Used for data tabulated in astrometric catalogs and fundamental solar system ephemerides.
>
> Geocentric Celestial Reference System (GCRS) – Replaces the geocentric system based on the mean equator and equinox of J2000.0. Used for geocentric apparent positions of celestial objects, measurements and coordinates in the near-Earth environment, and artificial Earth satellite ephemerides.
>
> The BCRS and GCRS are nearly parallel systems, related by a relativistic transformation.
>
> International Celestial Reference System (ICRS) – The name applied to the orientation of the axes of the BCRS, based on the adopted coordinates of several hundred extragalactic radio sources that are assumed to have no net systematic motion. The resulting orientation is close to, but not exactly aligned with the mean equator and equinox of J2000.0. The ICRS is a "space-fixed" (kinematically non-rotating) system.
>
> Because of their close relationship, the abbreviations "BCRS" and "ICRS" are often used interchangeably.
>
> Celestial Intermediate Reference System – A system for geocentric apparent positions of stars and planets based on the true (instantaneous) equator of date and a zero point of right ascension at the Celestial Intermediate Origin (CIO) (see section A.5).

[31] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

The geocentric system based on the true equator and equinox of date is also still used for geocentric apparent positions of stars and planets.

These reference systems are now used for the input and output arguments to NOVAS subroutines.

For example, NOVAS now assumes that input reference data, such as catalog star positions and proper motions, and the basic solar system ephemerides, are provided in the ICRS (that is, within the BCRS as aligned to the ICRS axes), or at least are consistent with it to within the data's inherent accuracy. The distinction between the ICRS and the system defined by the mean equator and equinox of J2000.0 (the "J2000.0 system") becomes important only when an accuracy of 0.02 arcsecond or better is needed. Nevertheless, because NOVAS is designed for the highest accuracy applications, you will now see the ICRS mentioned as the reference system of choice for many input arguments to NOVAS subroutines.

Because the ICRS axes are not precisely aligned to those of the J2000.0 system, there is a new subroutine called FRAME to transform vectors between the two systems. This transformation is a very small fixed rotation. FRAME is used for both barycentric vectors (BCRS to or from the barycentric J2000.0 system) and geocentric vectors (GCRS to or from the geocentric J2000.0 system). FRAME is called many times, in both directions, within the NOVAS code. It is needed because precession (and nutation) can properly be applied only to vectors in a real equatorial system; vectors in the GCRS (geocentric ICRS) must be transformed, via FRAME, to the J2000.0 system before PRECES is used. If your code only interacts with the highest-level NOVAS subroutines, all this is transparent to you. However, if you use PRECES within your own code, you should precede it by a call to FRAME (with the middle argument $K > 0$) if your input vector is expressed in the GCRS, that is, if it is derived from an input source based on the ICRS.

Output data from many of the supervisory-level NOVAS subroutines can be expressed in the GCRS or either of two equator-of-date systems, the true equator and equinox of date, or the Celestial Intermediate Reference System. The latter two systems differ only in their right ascension origins, and in the new paradigm they are understood to be derived from the GCRS by applying a few rotations.

A.4 New Models for Precession and Nutation

As described in section 1.4, new models for both precession and nutation have been adopted by the IAU and have been incorporated into NOVAS. Although the underlying developments for these effects are different than in NOVAS 2.0, from a programming point of view, little has changed. The subroutines that directly involve precession and nutation — PRECES, NUTATE, ETILT, NOD, and SIDTIM—work the same as before, but with slightly different results. It should be noted that the new nutation model has more than ten times the number of trigonometric terms than the previous model. Since evaluation of nutation has always been the most computationally intensive task in NOVAS, you may notice an increase in execution time for some NOVAS applications.

However, that extra computation time can be reduced: Earth rotation calculations can be performed in either high- or low-accuracy mode. A call to either HIACC or LOACC (without arguments) sets the accuracy, which remains in effect until changed. High-accuracy mode is the default, with the various models evaluated at the few-microarcsecond level. For nutation, for example, this means that a 1365-term trigonometric series is evaluated for each unique date. Neither the models nor current observations are accurate at this level, however; so much of the increased computational burden is unproductive. A call to LOACC sets the Earth rotation computations (and *only* those computations) in NOVAS to an accuracy of 0.1 milliarcsecond. The computation time for these calculations is thereby reduced by about two-thirds.

A.5 New Model for the Rotation of the Earth about its Axis

IAU resolutions passed in 2000 established a new geometric paradigm for how we describe the Earth's spin *around* its axis. Both the old and new paradigms are based on the instantaneous (true) equator of date, but they use different fiducial points on the equator as the origin of right ascension, and different time-like quantities (actually, time-dependent angles) to describe the rotation of the Earth. As described in section 1.4 and Chapter 5, the conventional scheme is based on the equinox and sidereal time; in the new paradigm, the reference point is called the Celestial Intermediate Origin (CIO) and the time-like quantity is called the ERA. A more complete explanation of the new concepts, along with the algorithms used with them, can be found in Chapter 6 of USNO Circular 179.[32]

NOVAS 3.0 implements both the equinox- and CIO-based computational schemes. Implementing the CIO-based paradigm has required the addition of many new subroutines, along with new code added to existing subroutines. First, subroutine PLACE is coded to provide output right ascensions with respect to either the equinox or the CIO. Subroutine GCRSEQ can similarly convert GCRS right ascensions and declinations to their equatorial equivalents (for a given date), with output right ascensions measured with respect to either zero point. CIORA provides the angle between the two zero points, that is, the difference between the two right ascension systems. Subroutines CIOLOC, CIOBAS, and CIORD provide lower-level support to these computations. Subroutine EROT computes the ERA for any instant.

Two high-level NOVAS subroutines that involve Earth rotation, SIDTIM and TERCEL (the latter replaces the old PNSW) have been re-coded to perform their internal calculations using either the equinox-based or CIO-based paradigm. The method used is selected by a prior call to either EQINOX or CIOTIO (without arguments), which remains in effect until changed. The equinox-based paradigm is the default, that is, it is used unless CIOTIO has been called.

[32] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

A.6 New Features

Subroutines have been added to NOVAS that provide new functionality and convenience:

PLACE: A new general-purpose apparent place subroutine (see section A.2).

IDSS: An integer function that returns a solar system body's identification number (which is used in various NOVAS subroutine calls), given the body's name as a character string. For example, `IDSS('MARS')` usually equals 4. Because IDSS is a function, it can be referred to within calls to other NOVAS subroutines, e.g.,

```
     CALL APPLAN (TTJD,IDSS('JUPITER'),IDSS('EARTH'),RAJUP,DECJUP,DISJUP)
```

(If you supply your own version of subroutine SOLSYS, you must also now supply a corresponding version of IDSS.)

GETVEC: A subroutine that returns the last NOVAS-computed celestial position (apparent or astrometric place, etc.) as a unit vector. The vector is expressed in the same reference system as the previously supplied spherical coordinates.

EQECL: Converts right ascension and declination to ecliptic longitude and latitude. Also, EQEC and ECEQ convert vectors from an equatorial to an ecliptic basis and vice versa, respectively.

EQGAL: Converts ICRS right ascension and declination to galactic longitude and latitude.

GCRSEQ: Converts GCRS (geocentric ICRS) right ascension and declination to one of the equatorial systems of date.

ASTCON: Supplies the value of an astronomical constant, given its name as a character string. The values of all fundamental astronomical constants used by NOVAS are stored within this subroutine and nowhere else. The names of the constants available and the units used for each are listed in the subroutine's preamble. For example,
`CALL ASTCON ('ERAD',1.D0,RADIUS)` returns, in argument `RADIUS`, the value of the equatorial radius of the Earth in meters.

A.7 New Terminology

Not surprisingly, the IAU resolutions on reference systems and Earth rotation have required some new terminology, and an IAU Working Group on Nomenclature for Fundamental Astronomy was established for the 2003–2006 triennium to sort it all out. New terms and abbreviations now appear in comment statements in many of the NOVAS subroutines, including the preambles where the input and output arguments are described. The most important terms are described in Chapter 1 of this document and further information can be found in USNO Circular 179.[33]

(Return to Table of Contents)

[33] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

Appendix B List of Changes to NOVAS Subroutines from Version F2.0 to Version F3.0

B.1 New Subroutines

NU2000A – from IERS (Wallace), evaluates IAU 2000A nutation series (nutation only).

NU2000K – modification of NU2000A, evaluates truncated version of full IAU 2000 A. More accurate than IAU 2000 B series. Also uses a consistent set of expressions for the fundamental arguments, those of Simon et al. (1994). Accuracy: about 0.1 milliarcsecond for $\Delta\psi$, about 0.04 milliarcsecond for $\Delta\varepsilon$ and $\Delta\psi \sin \varepsilon$.

EECT2000 – from IERS (Wallace), evaluates 34-term series for "complementary terms" in equation of the equinoxes. Uses new function ANMP from same source to normalize angles in the range of $-\pi \leq \text{angle} \leq +\pi$.

EROT – evaluates the ERA, θ.

FRAME – sets up frame tie matrix and transforms vector from dynamical mean J2000.0 system to ICRS, or vice versa. FRAME implements a first-order matrix with second-order corrections to the diagonal elements, patterned after what is given in the Hilton and Hohenkerk (2004) A&A paper. Given the smallness of the angles involved and their uncertainties, this is quite adequate.

PLACE – New, general-purpose subroutine for computing apparent, topocentric, virtual, astrometric, etc., places of stars and planets. All substantive code for performing these calculations has been moved from APSTAR, TPSTAR, APPLAN, etc., into PLACE. In the call to PLACE, the object requested is specified by name, using a character argument, e.g., 'SUN', 'MOON', 'JUPITER', 'STAR', etc. The type of place requested is specified by two input codes, one indicating the location of the observer and the other indicating the coordinate system of the output positions. APSTAR, TPSTAR, APPLAN, etc., now are just "front-ends" to PLACE.

SETVEC – stores the last-computed celestial position vector.

GETVEC – allows the user to retrieve the last-computed celestial position as a unit vector.

IDSS – returns the body identification number of a specified solar system body, to be used in calls to SOLSYS, APSTAR, APPLAN, etc. Actually a FUNCTION. The solar system body is specified by its name (all upper case letters) in the character variable that is the single input argument. For example, IDSS('EARTH') = 3 (usually). A version of IDSS must now be packaged with each version of SOLSYS.

ASTCON – provides values of astronomical constants.

SETDT – allows user specification of ΔT (= TT – UT1) value in seconds. The ΔT value set here is used both in SIDTIM and TERCEL and, in certain circumstances, in PLACE.

GETDT – retrieves ΔT value (in days) previously specified via SETDT (in seconds).

GCRSEQ – transforms GCRS RA & dec to RA & dec on mean or true equator of date. For true equator of date, either the true equinox or the CIO can be specified as the origin of right ascension.

EQECL – converts equatorial RA & dec to ecliptic longitude and latitude.

EQEC – converts an equatorial position vector to an ecliptic position vector.

ECEQ – converts an ecliptic position vector to an equatorial position vector.

EQGAL – converts ICRS RA & dec to galactic longitude and latitude.

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

GRVDEF – replacement for SUNFLD that supervises the evaluation of gravitational deflection of light due to the Sun, Jupiter, and other solar system bodies. Calls new subroutine GRVD to do the deflection calculation for each body.

GEOPOS – called from PLACE to compute the geocentric position and velocity vectors of an observer on or above the surface of the Earth.

LITTIM – called from PLACE to antedate the position of a solar system body for light-time.

LIMANG – evaluates where an observed object is with respect to the Earth's limb (horizon), given the geocentric position vectors of the observer and the object. PLACE calls LIMANG for the topocentric cases in deciding whether to include the gravitational deflection of light due to the Earth itself.

CIORA – returns the value of the true right ascension of the CIO for a given TDB Julian date.

CIOLOC – returns the right ascension of the CIO at a given TDB Julian date, either with respect to the GCRS or the true equator and equinox of date.

CIORD – called from CIOLOC, reads and returns a set of values of the GCRS right ascension of the CIO, near a given TDB Julian date, from an external file (either formatted sequential or binary direct-access).

CIOFIL – allows the specification of the external file of CIO right ascensions that CIORD reads.

CIOBAS – returns orthonormal basis vectors for Celestial Intermediate Reference System with respect to the GCRS. Requires previous call to CIOLOC.

EQXRA – 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.

RADVL – called from PLACE to compute the radial velocity of observed object with respect to the observer.

SETMOD – sets method/accuracy mode for Earth rotation calculations.

GETMOD – retrieves method/accuracy mode for Earth rotation calculations.

EQINOX – specifies that equinox-based method is to be used for Earth rotation calculations.

CIOTIO – specifies that the CIO-based method is to be used for Earth rotation calculations.

HIACC – specifies that high-accuracy (~1 μas) algorithms are to be used for Earth rotation calculations.

LOACC – specifies that low-accuracy (~0.1 milliarcsecond) algorithms are to be used for Earth rotation calculations.

RESUME – reverts to method/accuracy mode used prior to latest change (by one of the above subroutines).

B.2 Changes to Calling Sequences

APSTAR, VPSTAR, ASSTAR, CATRAN, GETHIP, VECTRS – proper motion units (in both RA and dec) changed to milliarcseconds per year (proper motion in RA includes cos δ factor), parallax units changed to milliarcseconds.

TPSTAR, TPPLAN, LPSTAR, LPPLAN – the user's option to specify the input time argument as apparent sidereal time in hours is now discouraged; specifying the corresponding UT1 Julian date is now recommended. Sidereal time input is still supported but might not be in future NOVAS releases.

PRECES, CATRAN – one of the input epochs must now be 2451545.0 (J2000.0). Can no longer do two arbitrary epochs (the new precession expressions are not as flexible as Newcomb's or Lieske's).

CATRAN – has two new transformation options: IT = 4 rotates data from the mean equator and equinox of J2000.0 to the ICRS and IT = 5 does the opposite rotation.

WOBBLE – Julian date argument added.

PNSW – name changed to TERCEL (TERrestrial-to-CELestial transformation). Input argument changed to UT1 Julian date in a pair of double-precision words.

CELPOL – input corrections to pole position can now be either (dX, dY) or (dψ, dε), the choice specified by a new input argument. Units must now be in milliarcseconds. A Julian date is also now required input.

SPIN – no longer specifically associated with sidereal time. Now applies a rotation about the current z-axis, with angle expressed in degrees.

SUNFLD – replaced by GRVDEF, a more general subroutine that evaluates the gravitational deflection of light due to several solar system bodies.

SOLSYS version 2 – can accept a split Julian date if provided in two successive calls; see description in section B.3.

SOLSYS version 3 – M=10 now provides coordinates of the Earth, not the Sun; other values of M unaffected.

All of the high-level subroutines (PLACE, APSTAR, APPLAN, etc.) now assume that they are working with ICRS data; this goes for the input RA, dec, and proper motion components for the star routines, and the position and velocity vectors obtained from SOLSYS (e.g., from DE405) in both the star and planet routines. VPSTAR, LPSTAR, VPPLAN, LPPLAN, ASSTAR, ASPLAN, and MPSTAR produce output positions in the GCRS or ICRS.

B.3 Significant Internal Changes to Code

Common error conditions will now generate error messages sent to unit=* (standard output, usually the terminal screen). Each error message always begins with the name of the subroutine that produced it and is a plain-English description of the problem.

All subroutines that need astronomical constants now call ASTCON to obtain the values they need on their first call. Those values are SAVEd for use on subsequent calls. Those values are:

SPEED OF LIGHT IN METERS/SECOND — A DEFINING PHYSICAL CONSTANT:

$c = 299{,}792{,}458$

LIGHT-TIME FOR ONE ASTRONOMICAL UNIT IN TDB SECONDS, FROM DE405:

$a(\text{sec}) = 499.0047838061$

SPEED OF LIGHT IN AU/DAY:

$c(\text{AU/day}) = 86400 / a(\text{sec})$

LENGTH OF ASTRONOMICAL UNIT IN METERS:

$a = a(\text{sec}) \times c$

HELIOCENTRIC GRAVITATIONAL CONSTANT IN METERS^3/SECOND^2, FROM DE405:

$GS = 1.32712440017987 \times 10^{20}$

GEOCENTRIC GRAVITATIONAL CONSTANT IN METERS^3/SECOND^2, FROM DE405:

$GM = 3.98600433 \times 10^{14}$

EQUATORIAL RADIUS OF EARTH IN METERS, FROM IERS CONVENTIONS (2003):

$r_\oplus = 6{,}378{,}136.6$

FLATTENING FACTOR OF EARTH, FROM IERS CONVENTIONS (2003):

$f = 1 / 298.25642$

NOMINAL MEAN ROTATIONAL ANGULAR VELOCITY OF EARTH, IN RADIANS/SECOND, FROM IERS CONVENTIONS (2003):

$\omega = 7.2921150 \times 10^{-5}$

RECIPROCAL MASSES (SUN MASS/BODY MASS) FOR SOLAR SYSTEM BODIES

SUN = 1

MOON = 27,068,700.387534

MERCURY = 6,023,600

VENUS = 408,523.71

EARTH = 332,946.050895

MARS = 3,098,708

JUPITER = 1,047.3486

SATURN = 3,497.898

URANUS = 22,902.98

NEPTUNE = 19,412.24

PLUTO = 135,200,000

EARTH-MOON BARYCENTER = 328,900.561400

DE405 values are used for many of these, which are in TDB (T_{eph}) units. NOVAS output is practically insensitive to changes in low-order digits of the above constants; they are mostly used for relatively small corrections, such as the gravitational deflection of light. Probably the light-time for 1 AU is the most important, because it is used for the light-time correction. The constants that really matter in NOVAS are the coefficients to the series expansions in the individual subroutines, i.e., the constants that are embedded in the models for precession, nutation, etc.

APSTAR, TPSTAR, APPLAN, TPPLAN, VPSTAR, LPSTAR, VPPLAN, LPPLAN, ASSTAR, ASPLAN – now are simply "front-ends" to specific calls to PLACE. All substantive apparent place calculations of various kinds are now done only in PLACE. The following changes in the basic algorithms were made:

(1) Calls to FRAME were added in appropriate places to transform between the ICRS and the dynamical system.

(2) In updating a star's position for proper motion, the epoch of interest is now corrected for the difference in light-time between the solar system barycenter (the reference point for the input catalog data) and the Earth itself. (This affects only stars with the greatest proper motions, and then only at the 0.1 milliarcsecond level). Uses the new subroutine DLIGHT to compute the epoch offset.

(3) The "Doppler factor", k, is included in the computation of stars' space motion vectors (see note on VECTRS).

(4) Modifications were made related to the change in gravitational deflection algorithms from SUNFLD to the more-general GRVDEF (see note on GRVDEF).

(5) Code has been introduced that allows a place to be expressed in the Celestial Intermediate Reference System (equator of date with CIO as right ascension origin).

(6) Code has been added that allows the input of an observer's instantaneous geocentric position and velocity vectors (with respect to the true equator and equinox of date) for a topocentric place calculation; this is included to support satellite observations.

TERCEL, SOLSYS – calls to FRAME added at appropriate places. (In SOLSYS, the call to FRAME is commented out for DE405 and later JPL ephemerides, since DE405 is in ICRS.)

CATRAN, GETHIP, VECTRS – code adjusted for new proper motion and parallax units.

CATRAN – code added to call FRAME for new IT = 4 and IT = 5 options that rotate data between dynamical J2000.0 system and ICRS.

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.

SIDTIM – returns value of sidereal time, either mean or apparent. Internally can work by either of two methods, set by previous call to SETMOD, EQINOX, or CIOTIO:

Equinox-based method: Evaluates expression for mean sidereal time given in Capitaine et al. (2003), A&A, 412, 567, eq. (42). The ERA, θ, is obtained from EROT. For apparent sidereal time, the equation of the equinoxes, including the "complementary terms", is obtained from ETILT.

CIO-based method: Obtains sidereal time from eq. (6) given in Kaplan (2003) in *IAU XXV, Joint Discussion 16: The International Celestial Reference System, Maintenance and Future Realizations* (USNO), p. 196. That equation is based on the position of the true equinox of date in the Celestial Intermediate Reference System, the basis of which is obtained from CIOBAS. The ERA, θ, is obtained from EROT. Mean sidereal time, when requested, is obtained by *subtracting* the equation of the equinoxes, obtained from ETILT.

In either method, SIDTIM/EROT evaluates θ using the input UT1 epoch, but other components of sidereal time are evaluated using TDB (set equal to TT), with TT = UT1 + ΔT. Default value is ΔT = 64 sec, applicable at or near 2000; for highest precision applications, ΔT value can be set via prior call to SETDT.

TERCEL – performs the terrestrial-to-celestial transformation on a given vector, i.e., the total rotation from the ITRS to the ICRS. Internally can work by either of two methods, set by previous call to SETMOD, EQINOX, or CIOTIO:

Equinox-based method: Evaluates the old-style transformation as per previous subroutine PNSW, but with a call to FRAME added at the end to put final vector in ICRS. Uses apparent sidereal time, obtained from SIDTIM.

CIO-based method: Performs the transformation of eq. (4) given in Kaplan (2003) (reference above), based on the Celestial Intermediate Reference System. The orthonormal basis of the system is obtained from CIOBAS and the ERA, θ, is obtained from EROT.

In either method, the "fast angle" (rotation about z axis) is evaluated using the input UT1 epoch, but other components of the transformation are evaluated using TDB (set equal to TT), with TT=UT1+ΔT. Default value is ΔT=64 sec, applicable at or near 2000; for highest precision applications, ΔT value can be set via prior call to SETDT.

ETILT – now evaluates a more complete series for the complementary terms in the equation of the equinoxes (formerly just the two largest terms). Internally works in either high- or low-accuracy mode, set by previous call to SETMOD, HIACC, or LOACC:

High-accuracy mode: Obtains the sum of the terms from IERS function EECT2000.

Low-accuracy mode: Obtains the sum of the terms from a 9-term internal series.

ETILT uses the expression for the mean obliquity from the P03 precession formulation.

PRECES – now evaluates precession-angle polynomials for the IAU 2006 precession model (also called P03). Some code changes made to ensure reversibility of transformation (to/from J2000.0).

NOD – now just calls either of the nutation subroutines, NU2000A (from the IERS) or NU2000K (a reduced-accuracy version of NU2000A), to do the hard work; does not contain nutation series itself. Which nutation subroutine is called depends on whether high-accuracy or low-accuracy mode has been chosen for Earth rotation calculations (see new subroutines SETMOD, LOACC, HIACC).

FUNARG – now evaluates expressions for the fundamental solar and lunar arguments from Simon et al. (1994), A&A, 282, 663. However, IERS subroutine NU2000A, that evaluates the full nutation series, develops its fundamental arguments internally (a mixed bag of expressions).

WOBBLE – very tiny (inconsequential for most applications) rotation about z axis added to matrix to correct ITRS longitude origin to TIO, using recently published approximation to TIO longitude as a function of time (which required the new time argument added to this subroutine). Essentially, this changes W rotation to W'. Also changed matrix element expressions from first-order approximations to exact expressions for increased precision.

ZDAZ – improved algorithm for refraction geometry implemented.

SOLSYS version 1 – ephemeris file that is read has been changed to formatted (ASCII); the coordinates in the file must now be barycentric, with respect to the ICRS axes; and the interpolation is now 7-point Lagrangian. The code has also been considerably cleaned up.

SOLSYS version 2 – call to the JPL ephemeris-access routine has been changed from the single-argument JD call to the double-argument JD call; that is, from PLEPH to DPLEPH. SOLSYS now splits the integral part of the input Julian date from the fractional part and sends these to DPLEPH separately. The result is the same as a call to PLEPH with the JD in one piece. But there is now code such that if, in a subsequent call to SOLSYS, the JD is between −1 and +1, SOLSYS will interpret it as a *fraction of a day* to be used with the *integral part of the JD from the previous call*. This scheme ensures that there is no effect on existing applications that directly call SOLSYS, but it provides a higher-precision option if two calls in succession are used in this way. (The way to test whether any version of SOLSYS can use split JDs is to check whether the value of IDSS(‘JD’) is 1 or 2, with 2 meaning that Julian dates can be split between successive calls.)

SOLSYS version 3 – adjustments for IAU 2006 precession; more sophisticated (and accurate) conversion from heliocentric to barycentric coordinates.

B.4 Other Internal Code Changes

Many minor changes have been made in the code. Obviously many of the comment statements had to be revised, and others added, too numerous to try to list. Some of the code is now more Fortran-77-like and less Fortran-66-like, especially in the subroutines in which other changes had to be made; but a uniform scrub was not done. NOVAS still has plenty of ancient-looking code, it's still all-caps, and there are still some GO TOs. On the other hand, since NOVAS is mostly computational, flowing top-to-bottom within each subroutine, without any complicated logic, it hardly matters.

Some variable names were changed. For example, the variable PI in some subroutines was used for the parallax and not the mathematical constant π=3.14159…, which could be confusing. In these cases, the parallax variable name is now PX. The input (catalog) RA and dec for many subroutines had been named RAM and DECM, the M indicating "mean;" these are now RAI and DECI, the I indicating "ICRS." Many similar trivial changes have been made.

(Return to Table of Contents)

Appendix C How to Set Up the JPL Ephemerides

C.1 Overview

As described elsewhere in this document, NOVAS requires access to a high-accuracy solar system ephemeris in order to compute places of solar-system bodies and the highest-accuracy star places. High-accuracy solar system ephemerides are now being produced by groups in the U. S., France, and Russia. NOVAS directly supports only the "development ephemerides," typically designated as "DEnnn," produced by the JPL in the U.S. In NOVAS, SOLSYS version 2 provides access to these ephemerides. This version of SOLSYS is simply a front-end to JPL-provided software, which reads and interpolates a binary, direct-access ephemeris file. The binary ephemeris file is created from ASCII data files and software supplied by JPL. The JPL software must be tailored for your specific computer architecture.

This appendix describes how to set up the binary ephemeris file and the JPL software that reads it on your system. After this has been successfully done, the JPL software must be linked into any NOVAS application that uses SOLSYS version 2. The procedures outlined below worked on an Intel-based Mac OS X system using the open-source gfortran compiler and JPL software available in February 2009. The resulting binary, direct-access ephemeris file was successfully transferred to and used on other Intel-based computers running Microsoft Windows XP and Linux. Thus, our procedures are tailored for computers containing Intel processors; this includes many systems running Microsoft Windows, Mac OS X, and Linux. Providing specific procedures for all combinations of computer processors, operating systems, and compilers is beyond the scope of this User's Guide. Furthermore, the procedures in the appendix are intended simply as a guide; USNO cannot provide technical support regarding the JPL software.

C.2 Step-by-Step Guide

Step 1: Connect to the JPL ftp site. All the files needed to install the JPL ephemerides are available via anonymous ftp from ssd.jpl.nasa.gov. This can be accomplished through most modern Web browsers by typing:
`ftp://ssd.jpl.nasa.gov`
in the address (URL) field. When connected, go to the `pub/eph/planets/` directory.

Step 2: Download the JPL software, ASCII ephemeris files, and corresponding test data file. Follow the "CONTENTS TO BE RETRIEVED BY THE USER" instructions in the file README.txt, which is also available as a webpage.[34] *We* recommend *following the instructions for "non-UNIX users" regardless of your computer's operating system*. Our experience at USNO has been that the installation process is more a function of your computer's processor (hardware) than its operating system. The test data file, testpo.xxx, may be found in both the same directory as the ASCII ephemeris files and in the separate test data directory.

[34] http://ssd.jpl.nasa.gov/?planet_eph_export

The characters used to terminate lines, the end-of-line codes, vary among operating systems. Some systems use carriage returns (CR), some use line feeds (LF), and some use both. The end-of-line codes may not be properly translated between operating systems when files are copied from one system to another, depending on how the files are transferred and on what options are invoked. If you have trouble reading or using any of the files downloaded from JPL, check the encoding to ensure it is appropriate to your system.

Step 3: Configure the asc2eph.f file provided by JPL. For computer systems with Intel processors, select (uncomment) the following statement at the beginning of the program:

```
PARAMETER ( NRECL = 4 )
```

Step 4: Split the testeph.f file provided by JPL into two files: jplsubs.f and testeph.f. The first new file, jplsubs.f, should contain JPL subroutines FSIZER3, PLEPH (including ENTRY DPLEPH), INTERP, SPLIT, STATE, and CONST extracted from the original JPL test program. The second revised file, testeph.f, should contain only the main program and no subroutines. JPL subroutines FSIZER1 and FSIZER2 will not be needed and are not retained in either file.

The creation of the new file, jplsubs.f, is advantageous for the long-term use of the JPL subroutines and binary ephemeris files with NOVAS. The new file contains no extraneous material and is named descriptively.

Step 5: Configure the new jplsubs.f file.

a) For computer systems with Intel processors, in SUBROUTINE FSIZER3 of jplsubs.f, uncomment and set
```
NRECL=4
```
b) In SUBROUTINE FSIZER3 of **jplsubs.f**, identify the binary ephemeris file to be used by setting
```
NAMFIL= 'JPLEPH'
```
You may wish to specify a more complete file name including the appropriate directories, such as
```
NAMFIL= '/users/mystuff/ephem/JPLEPH'
```
to avoid having to keep an alias to this binary file in all your working directories.

c) In SUBROUTINE FSIZER3 of jplsubs.f, set KSIZE to the correct value for the specific ephemeris being used (see the comments in the code); e.g., set
```
KSIZE = 2036
```
for DE405.

d) In SUBROUTINE STATE of jplsubs.f, select (uncomment)
```
CALL FSIZER3(NRECL,KSIZE,NRFILE,NAMFIL)
```
leaving the lines containing `CALL FSIZER1` and `CALL FSIZER2` as comments.

Step 6: Compile and create executables.

a) Create executable application asc2eph by compiling and linking file asc2eph.f.
b) Create executable application testeph by compiling and linking files testeph.f and jplsubs.f.

NOTE for Windows systems: Some Fortran compilers may produce files with an ".exe" extension, i.e., asc2eph.exe and testeph.exe. If so, ignore the ".exe" extension when using those applications in the following steps.

Step 7: Concatenate the ASCII data files and convert the concatenated file to binary form. Follow the instructions in the "ASCII to BINARY (for non-UNIX users)" section of the usrguide file, which is located in the `pub/eph/planets` directory. The resulting binary ephemeris file will be named JPLEPH, or whatever name you specified in Step 5b.

Step 8: Test the binary ephemeris file. Follow the instructions in the "TESTING THE BINARY FILE" section of the usrguide file, which is located in the `pub/eph/planets`. You should have already downloaded the appropriate test data file in Step 2.

Step 9: Use the binary ephemeris file with NOVAS. After successfully completing the test, compile and link NOVAS subroutine SOLSYS version 2 (NOVAS_F3.0_solsys2.f) and the JPL subroutines contained in jplsubs.f with the remaining relevant parts of NOVAS and your application code. Unless you specified otherwise in Step 5b, the binary JPL ephemeris file, or an alias to it, should reside in the same directory as your executable application.

(Return to Table of Contents)

Appendix D A Comparison of SOFA and NOVAS

The Standards of Fundamental Astronomy (SOFA)[35] library is the official collection of approved software for positional astronomy, operating under the auspices of IAU Division 1 (Fundamental Astronomy). There are both Fortran and C versions. An international SOFA Reviewing Board manages the collection.

Generally, NOVAS is independent of SOFA, although there are two IERS routines in NOVAS that have code identical to their SOFA counterparts: NU2000A, which evaluates the full 1365-term IAU 2000A nutation series, is the same as iau_NUT00A in SOFA; and function EECT2000, which evaluates the "complementary terms" in the equation of the equinoxes, is the same as iau_EECT00 in SOFA.

In the document *SOFA Tools for Earth Attitude*, also known as the "SOFA Cookbook", there are several examples of the transformation between terrestrial and celestial coordinate systems. Here we look at how one of those examples plays out in both SOFA and NOVAS.

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 methods at full accuracy. With SOFA, we used the "IAU 2006/2000A, CIO-based" method. 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.

D.2 Procedure

The test programs for NOVAS and SOFA follow as Addendums 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 NUT06A includes small corrections to the nutation series from the P03 precession that are not used in the NOVAS calculations. The corrections amount to only a few microarcseconds (μas) for current dates.

[35] http://www.iausofa.org/index.html

A series of tests were done, with and without corrections for polar motion, precession and nutation, and the P03 correction in SOFA, and the resulting rotation matrices were compared.

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 subroutine TERCEL.

D.3 Results

Initially, the rotation matrices differed by about 3 μas, which was almost entirely a rotation about the z-axis and was larger than expected. This discrepancy was traced to the way the UT1 Julian date was divided into two parts, which is necessitated because of the high precision that was needed. The best agreement with the SOFA results is obtained when the UT1 Julian date in the NOVAS program is expressed as

```
2400000.5 + 54195.4999991658…
```

That is, it was necessary to split the UT1 Julian date for NOVAS in exactly the same place as in the SOFA example.

Using this representation of UT1, the following results were obtained:

Polar Motion	CIP Offsets	P03 Corrections	CIO_RA file / EQINOX mode	Difference (μas)
No	No	No	No	0.25814
No	No	Yes	No	1.6752
No	Yes	Yes	No	1.6728
Yes	Yes	Yes	No	1.6735
Yes	Yes	No	No	0.28679
Yes	Yes	No	CIO_RA file	0.34369
Yes	Yes	No	EQINOX mode	0.28644

The table shows that the latest Fortran 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 equinox mode for the NOVAS computations does not have a significant effect on the results.

Alice Monet
George Kaplan
27 Feb 2009

D.4 Addendum I: NOVAS Code

```
PROGRAM terceltest

C-----Transform vectors from ITRS to GCRS

      DOUBLE PRECISION tjdh, tjdl, xp, yp, delt, vec1(3),
     . vec2(3), tjd, dx, dy
      INTEGER num, mode

      DATA delt /65.25607389d0/, num / 0 /
      CALL setdt (delt)

      dx = +0.1750d0
      dy = -0.2259d0
C     dy = 0d0
C     dx = 0d0

C-----Open the input file of Julian dates,CIO coords,ITRS vector

      OPEN (UNIT = 15, FILE = 'tercel-test-input.dat',
     . STATUS = 'OLD', ACCESS = 'SEQUENTIAL')

   10 READ (15,*, END = 20) num, tjdh, tjdl, xp, yp, vec1

C----- Set transformation method, accuracy level, and UT1-UTC.

      tjd = tjdh + tjdl

        CALL celpol (tjd,2,dx,dy)
C       CALL ciotio
        CALL eqinox
        CALL hiacc

C----- Rotate vec1 from ITRS to GCRS = vec2

        CALL TERCEL(tjdh,tjdl,xp,yp,vec1,vec2)
        WRITE (*,101) num, vec2
         GO TO 10
   20 CONTINUE

  101 FORMAT (i1,3(4x,f20.17))

      STOP
      END
```

D.5 Addendum II: SOFA Code

```
*  SOFA examples

      IMPLICIT NONE

*  Arcseconds to radians
      DOUBLE PRECISION AS2R
      PARAMETER ( AS2R = 4.848136811095359935899141D-6 )

*  2Pi
      DOUBLE PRECISION D2PI
      PARAMETER ( D2PI = 6.283185307179586476925287D0 )

      CHARACTER PM
      INTEGER IY, IM, ID, IH, MIN, J
      INTEGER IHMSF(4)
      DOUBLE PRECISION SEC, XP, YP, DUT1,
     :                 DDP80, DDE80, DX00, DY00, DX06, DY06,
     :                 DJMJD0, DATE, TIME, UTC, DAT,
     :                 TAI, TT, TUT, UT1, RP(3,3), DP80, DE80,
     :                 DPSI, DEPS, EPSA, RN(3,3), RNPB(3,3),
     :                 EE, GST, RC2TI(3,3), RPOM(3,3),
     :                 RC2IT(3,3), X, Y, S,
     :                 RC2I(3,3), ERA, DP00, DE00, RB(3,3),
     :                 RPB(3,3), V1(3), V2(3), DDP00, DDE00
      DOUBLE PRECISION R1(3,3), R2(3,3), R3(3,3), R4(3,3), R5(3,3)
      DOUBLE PRECISION iau_OBL80, iau_EQEQ94, iau_ANP, iau_GMST82,
     :                 iau_ERA00, iau_SP00, iau_EE00, iau_GMST00,
     :                 iau_S06,TERMAT(3,3), TERTRANS(3,3),
     :                 SOFMAT(3,3), SOFTRANS(3,3)
      DOUBLE PRECISION DROT, DJMJDATE, UTHI, UTLO
C.....
      OPEN (UNIT = 16, FILE = 'tercel_matrix', STATUS = 'OLD',
     . ACCESS = 'SEQUENTIAL')
      READ (16,90) TERMAT
   90 FORMAT ( 1x, 3e24.12 )
      WRITE (*,'( 3(f20.15) )') TERMAT
C
      OPEN (UNIT = 17, FILE = 'sofa_matrix', STATUS = 'OLD',
     . ACCESS = 'SEQUENTIAL')
      READ (17,95) SOFMAT
   95 FORMAT ( 1x, 3f24.17 )
C     WRITE (*,'( 3(f20.15) )') SOFMAT
C.....
*  UTC.
      IY = 2007
      IM = 4
      ID = 5
      IH = 12
      MIN = 0
      SEC = 0D0
      WRITE ( *, '(1X,''date'',I6.4,2(''/'',I2.2))' ) IY, IM, ID
      WRITE ( *, '(1X,''time'',I4,I3,F5.1,''  UTC'')' ) IH, MIN, SEC

*  Polar motion (arcsec->radians).
      XP = 0.0349282D0 * AS2R
      YP = 0.4833163D0 * AS2R
      WRITE ( *, '(/1X,''X_p ='',SP,F13.9,'' arcsec'')' ) XP/AS2R
      WRITE ( *, '( 1X,''Y_p ='',SP,F13.9,'' arcsec'')' ) YP/AS2R
```

```
*  UT1-UTC (s).
      DUT1 = -0.072073685D0
      WRITE ( *, '(/1X,''UT1-UTC ='',SP,F16.12,'' s'')' ) DUT1

*  Nutation corrections wrt IAU 1976/1980 (mas->radians).
      DDP80 = -55.0655D0 * AS2R/1000D0
      DDE80 =  -6.3580D0 * AS2R/1000D0
      WRITE ( *, '(/1X,''dDpsi (1980) ='',SP,F13.9,'' arcsec'')' )
     :                                                  DDP80 / AS2R
      WRITE ( *, '( 1X,''dDeps (1980) ='',SP,F13.9,'' arcsec'')' )
     :                                                  DDE80 / AS2R

*  CIP offsets wrt IAU 2000A (mas->radians).
      DX00 =  0.1725D0 * AS2R/1000D0
      DY00 = -0.2650D0 * AS2R/1000D0
      WRITE ( *, '(/1X,''dX (2000) ='',SP,F13.9,'' arcsec'')' )
     :                                                   DX00 / AS2R
      WRITE ( *, '( 1X,''dY (2000) ='',SP,F13.9,'' arcsec'')' )
     :                                                   DY00 / AS2R

*  CIP offsets wrt IAU 2006/2000A (mas->radians).
      DX06 = 0.1750D0 * AS2R/1000D0
      DY06 = -0.2259D0 * AS2R/1000D0
C     DX06 = 0d0
C       DY06 = 0d0  * AS2R/1000D0
      WRITE ( *, '(/1X,''dX (2006) ='',SP,F13.9,'' arcsec'')' )
     :                                                   DX06 / AS2R
      WRITE ( *, '( 1X,''dY (2006) ='',SP,F13.9,'' arcsec'')' )
     :                                                   DY06 / AS2R

*  TT (MJD).
      CALL iau_CAL2JD ( IY, IM, ID, DJMJD0, DATE, J )
      TIME = ( 60D0*(60D0*DBLE(IH) + DBLE(MIN)) + SEC ) / 86400D0
      UTC = DATE + TIME
      CALL iau_DAT ( IY, IM, ID, TIME, DAT, J )
      TAI = UTC + DAT/86400D0
      TT = TAI + 32.184D0/86400D0
      WRITE ( *, '(/1X,''TT  = 2400000.5 +'',F22.15 )' ) TT

*  UT1.
      TUT = TIME + DUT1/86400D0
      UT1 = DATE + TUT
      WRITE ( *, '(/1X,''UT1 = 2400000.5 +'',F22.15 )' ) UT1
*  =========================
*  IAU 2006/2000A, CIO-based
*  =========================

      WRITE ( *, '(/1X,''   ========================='' //
     :             '/1X,''4) IAU 2006/2000A, CIO-based'' //
     :             '/1X,''   ========================='')' )

*  CIP and CIO, IAU 2006/2000A.

C-----Report inputs to CIP calculation


      CALL iau_XYS06A ( DJMJD0, TT, X, Y, S )

*  Add CIP corrections.
      X = X + DX06
      Y = Y + DY06

*  Report CIP and s.
```

```
      WRITE ( *, '(/1X,''X ='',SP,F22.15)' ) X/AS2R
      WRITE ( *, '( 1X,''Y ='',SP,F22.15)' ) Y/AS2R
      WRITE ( *, '( 1X,''s ='',SP,F13.9,'' arcsec'')' ) S/AS2R

*  GCRS to CIRS matrix.
      CALL iau_C2IXYS ( X, Y, S, RC2I )

C     WRITE (*,*) 'X,Y,S/AS2R: ',X, Y, S/AS2R

*  Report.
      CALL REPMAT ( 'NPB matrix, CIO-based (RC2I)', RC2I )

*  Earth rotation angle.
C
C----- Set TUT to the value used in Tercel

      WRITE (*,*) 'DJMJD0, DATE, TUT: ', DJMJD0, DATE, TUT

C       DJMJDATE = 2400000.5d0
C       TUT = 54195.4999991658d0
C     UTHI = 2454195.0d0
C       UTLO = 0.99999916581146d0
C
      ERA = iau_ERA00 ( DJMJD0, DATE+TUT )

C     ERA = iau_ERA00 ( UTHI, UTLO )
C     ERA = iau_ERA00 ( DJMJD0+DATE, TUT )

      WRITE ( *, '(1X)' )
      WRITE ( *, '(1X,''ERA ='',F19.16,'' rad'' )' ) ERA
      WRITE ( *, '(1X,''    ='',F16.12,'' deg'' )' ) ERA*360D0/D2PI
      CALL iau_D2TF ( 9, ERA/D2PI, PM, IHMSF )
      WRITE ( *, '(1X,''    ='',3I3.2,''.'',I9.9 )' ) IHMSF

*  Form celestial-terrestrial matrix (no polar motion yet).
      CALL iau_CR ( RC2I, RC2TI )
      CALL iau_RZ ( ERA, RC2TI )

*  Report.
      CALL REPMAT ( 'celestial to terrestrial matrix (no polar motion)',
     :                                                           RC2TI )
      CALL iau_CR (RC2TI,R3)

*  Polar motion matrix (TIRS->ITRS, IERS 2003).
      CALL iau_POM00 ( XP, YP, iau_SP00(DJMJD0,TT), RPOM )

*  Form celestial-terrestrial matrix (including polar motion).
      CALL iau_RXR ( RPOM, RC2TI, RC2IT )

C Transpose TERMAT
      CALL TRANSPOSE ( TERMAT, TERTRANS )
        CALL TRANSPOSE ( SOFMAT, SOFTRANS )

*  Report.
      CALL REPMAT ( 'celestial to terrestrial matrix', RC2IT )
        CALL REPMAT ( 'tercel_transposed matrix', TERTRANS )

C       WRITE ( *, '(1X)' )
C     WRITE (*,'( 3(f20.15) )') TERMAT

*  Copy for later comparison.
      CALL iau_CR ( RC2IT, R4 )
```

```
*  Compare result to TERCEL_Transposed matrix
      WRITE ( *, '(1X)' )
      WRITE (*,'( 1x,''w/ pm result vs tercel ='',e20.10,'' uas'')')
     . DROT ( R4, TERTRANS ) *1D6 / AS2R

* Compare SOFA w/P03 and SOFA w/o P03
      WRITE ( *, '(1X)' )
        WRITE (*,'( 1x,''sofa w/p03 vs w/o p03 ='',e20.10,'' uas'')')
     . DROT ( R4, SOFTRANS ) *1D6 / AS2R

      END

*----------------------------------------------------------------------*

      SUBROUTINE REPMAT ( S, R )
      IMPLICIT NONE
      CHARACTER*(*) S
      DOUBLE PRECISION R(3,3)
      WRITE (*,'(/1X,A/SP,2(3F22.17/),3F22.17)') S,R(1,1),R(1,2),R(1,3),
     :                                            R(2,1),R(2,2),R(2,3),
     :                                            R(3,1),R(3,2),R(3,3)
C     WRITE ( *, '(1X)' )
C     WRITE (*,*) S
C       WRITE (*,'(3f22.17)') R
      END

*----------------------------------------------------------------------*

      DOUBLE PRECISION FUNCTION DROT ( RMA, RMB )
*
*  Express the difference between two rotation matrices RMA,RMB as an
*  amount of rotation R about some arbitrary axis.
*
      IMPLICIT NONE
      DOUBLE PRECISION RMA(3,3), RMB(3,3)

      DOUBLE PRECISION RMBT(3,3), RM(3,3), RV(3), R

*  Multiply the first matrix by the inverse of the second.
      CALL iau_TR ( RMB, RMBT )
      CALL iau_RXR ( RMBT, RMA, RM )

*  Express the result as an r-vector.
      CALL iau_RM2V ( RM, RV )

*  Return the magnitude (the amount of rotation in radians).
      CALL iau_PM ( RV, R )
      DROT = R

      END

*----------------------------------------------------------------------*
      SUBROUTINE TRANSPOSE (R, RT)
*
*  Transpose 3x3 matrix R ---> RT
*
      IMPLICIT NONE
        DOUBLE PRECISION  R(3,3), RT(3,3)
        INTEGER i, j
*
      DO i = 1, 3
           DO j = 1, 3
             RT(i,j) = R(j,i)
```

```
               ENDDO
          ENDDO
*
        END

*________________________________________________________________________________*
```

(Return to Table of Contents)

Appendix E List of Internal Calls

ABERAT called by:

PLACE

ANGLES called by:

GCRSEQ MPSTAR PLACE

ANMP called by:

EECT2000

PLACES entry APPLAN not called

PLACES entry APSTAR not called

PLACES entry ASPLAN not called

PLACES entry ASSTAR not called

ASTCON called by:

ABERAT CATRAN DLIGHT GEOCEN GEOPOS GRVD

GRVDEF LIMANG PLACE RADVL TERRA VECTRS

AUXPOS called by:

SOLSYS

CALDAT not called

CATRAN called by:

GETHIP

ETILT entry CELPOL not called

CIOBAS called by:

CIORA GCRSEQ PLACE SIDTIM TERCEL

CIORD entry CIOFIL not called

CIOLOC called by:

CIORA GCRSEQ PLACE SIDTIM TERCEL

CIORA not called

CIORD called by:

CIOLOC

SETMOD entry CIOTIO not called

CONST called by:

SOLSYS

DLIGHT called by:

GRVDEF PLACE

DPLEPH called by:

SOLSYS

ECEQ not called

EECT2000 called by:

ETILT

EQEC called by:

EQECL

EQECL not called

EQGAL not called

SETMOD entry EQINOX called by:

GEOPOS ZDAZ

EQXRA called by:

CIOLOC SIDTIM

EROT called by:

SIDTIM TERCEL

ETILT called by:

ECEQ EQEC EQXRA GEOPOS NUTATE SIDTIM

FRAME called by:

CATRAN CIOBAS CIORA ECEQ EQEC ETILT

GCRSEQ GEOPOS PLACE SIDTIM TERCEL

FUNARG called by:

ETILT NU2000K

GCRSEQ not called

GEOCEN called by:

GRVDEF LITTIM PLACE

GEOPOS called by:

PLACE

SETDT entry GETDT called by:

GEOPOS SIDTIM TERCEL

GETHIP not called

SETMOD entry GETMOD called by:

ETILT NOD SIDTIM TERCEL

GETVEC not called

GRVD called by:

GRVDEF

GRVDEF called by:

PLACE

SETMOD entry HIACC not called

IDSS called by:

GRVDEF LITTIM PLACE

JULDAT not called

LIMANG called by:

PLACE

LITTIM called by:

PLACE

SETMOD entry LOACC not called

PLACES entry LPPLAN not called

PLACES entry LPSTAR not called

MPSTAR not called

NOD called by:

ETILT

NU2000A called by:

NOD

NU2000K called by:

NOD

NUTATE called by:

CIOBAS CIORA GCRSEQ GEOPOS PLACE SIDTIM

TERCEL

PLACE called by:

MPSTAR PLACES

PLACES not called

GEOPOS entry PLACST called by:

PLACES

PRECES called by:

CATRAN CIOBAS CIORA ETILT GCRSEQ GEOPOS

MPSTAR PLACE SIDTIM TERCEL

PROPMO called by:

PLACE

RADVL called by:

PLACE

REFRAC entry REFDAT not called

REFRAC called by:

ZDAZ

SETMOD entry RESUME called by:

GEOPOS ZDAZ

SETDT not called

SETMOD not called

GETVEC entry SETVEC called by:

ANGLES CATRAN ECEQ EQEC EQGAL MPSTAR

PLACE TERCEL

SIDTIM called by:

GEOPOS TERCEL

SOLSYS called by:

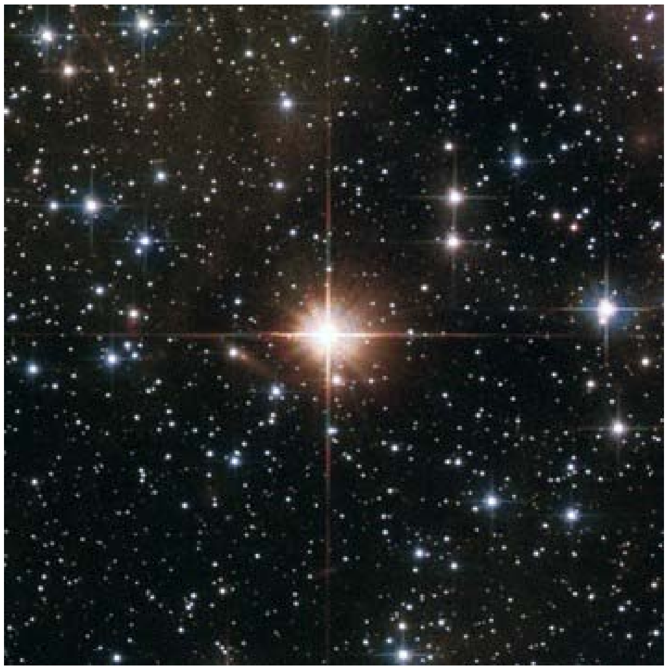
```
      GRVDEF      LITTIM      PLACE

SPIN called by:

      TERCEL

TERCEL called by:

        ZDAZ

TERRA called by:

      GEOPOS

TIMES called by:

       CIORA       ECEQ       EQEC      GCRSEQ      PLACE      SIDTIM

      TERCEL

PLACES entry TPPLAN not called

PLACES entry TPSTAR not called

VECTRS called by:

      MPSTAR      PLACE

PLACES entry VPPLAN not called

PLACES entry VPSTAR not called

WOBBLE called by:

      TERCEL

ZDAZ not called
```

(Return to Table of Contents)

User’s Guide
to NOVAS Version C3.0

Naval Observatory Vector Astrometry Software

C Edition



John Bangert
Wendy Puatua
George Kaplan
Jennifer Bartlett
Alice Monet

Part II of USNO Circular 180

U.S. Naval Observatory
December 2009

In this document, hyperlinks are in blue.

Table of Contents

Introduction

The **N**aval **O**bservatory **V**ector **A**strometry **S**oftware (NOVAS) is a source-code library in Fortran and C that provides common astrometric quantities and transformations. It can supply, in one or two function calls, the instantaneous celestial position of any star or planet in a variety of coordinate systems. The library also provides access to all of the “building blocks” that go into such computations—single-purpose functions for common astrometric algorithms, such as those for precession, nutation, aberration, parallax, etc. NOVAS calculations are accurate at the sub-milliarcsecond level. The package is an easy-to-use facility that can be incorporated into data reduction programs, telescope control systems, and simulations. The United States (U.S.) Nautical Almanac Office uses NOVAS in the production of its sections of *The Astronomical Almanac*.

The NOVAS algorithms are based on a vector and matrix formulation that is rigorous and consistent with recent recommendations of the International Astronomical Union (IAU). Objects inside and outside the solar system are treated similarly. The position vectors formed and operated on by the NOVAS functions are defined within either the Barycentric Celestial Reference System (BCRS) or the Geocentric Celestial Reference System (GCRS), as appropriate. Both of these systems are described in IAU resolutions passed in 2000. GCRS quantities are converted to more familiar coordinate systems, such as the equator and equinox of date, by applying standard rotations.

Three levels of functions are involved: basic, utility, and supervisory. Basic-level functions supply the values of fundamental variables, such as the nutation angles and the heliocentric positions of solar system bodies, for specific epochs. Utility-level functions perform computations corresponding to individual physical effects or transformations (aberration, light-bending, precession, polar motion, etc.). Supervisory-level functions call the basic and utility subroutines in the proper order to compute the apparent coordinates of stars or solar system bodies for specific dates and times. If desired, users can interact exclusively with the supervisory-level functions and not become concerned with the details of the geometry or physical models involved in the computation.

The Fortran version of NOVAS goes back to the late 1970s, but has been updated periodically to use new, more accurate models that represent the evolving standards of the international astronomical and geodetic communities. The success of the initial Fortran package led to requests for a C-language version. In the early 1990s, the U.S. Naval Observatory (USNO)/Naval Research Laboratory (NRL) optical interferometer group converted parts of NOVAS to C for use in their project. That work formed the basis for the first complete edition of NOVAS in C (designated NOVAS Version C1.0), which the USNO Astronomical Applications (AA) Department completed and released in 1996. A major revision of the NOVAS Fortran code took place in 1998, with the primary goal of supporting data conforming to the International Celestial Reference System (ICRS). Shortly thereafter, the C version of NOVAS was updated to version C2.0 to reflect the changes in the Fortran code and to extend its capabilities. In full-accuracy calculations using the same data sources, differences in the results from Fortran and C editions of NOVAS, for the same calculation, should not exceed 6×10^{-8} arcseconds (3×10^{-13} radians) for solar system bodies and 7×10^{-10} arcseconds (3×10^{-15} radians) for stars.

Previous versions of NOVAS were based on the algorithms described in Kaplan et al. (1989).[1] Although the phenomena that are considered and the overall sequence of calculations remains much the same in the current release, many of the models have been improved substantially over the last two decades in response to increased accuracy in observing techniques. Specifically, version 3.0 of NOVAS, released in 2009 and described in this document, implements the resolutions on astronomical reference systems and Earth rotation models passed at the IAU General Assemblies in 1997, 2000, and 2006. An explanation of the recent IAU resolutions can be found in *The IAU Resolutions on Astronomical Reference Systems, Time Scales, and Earth Rotation Models: Explanation and Implementation* (Kaplan 2005; hereafter USNO Circular 179[2]), which contains much more information on topics only briefly touched on in this document. This version of NOVAS also improves the accuracy of its star and planet position calculations by including several small effects not previously implemented in the code; see, for example, Klioner (2003). In addition, some new convenience functions have been added.

Although Chapter 1 provides some background material, the intent of this user's guide is not to explain positional astronomy. Many books have been written on this subject. NOVAS users who desire additional information on the terminology and concepts used in the software should consult *The Explanatory Supplement to the Astronomical Almanac* (Seidelmann 1992; hereafter *The Explanatory Supplement*) and USNO Circular 179. Until the revision of the former publication is completed, the latter serves as an important update providing an explanation of the recent IAU resolutions.

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 II of USNO Circular 180 (the Fortran guide is Part I), 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]

[1] Hohenkerk et al. (1992) reprinted parts of this article in Chapter 3 of *The Explanatory Supplement to the Astronomical Almanac.*
[2] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179
[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

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

Acknowledgements

Current version: William T. Harris of the USNO AA Department wrote most of the C code that accesses the JPL binary solar-system ephemerides (files **eph_manager.c** and **solsys1.c**).

Previous versions: Thomas K. Buchanan, working as part of the USNO/NRL Optical Interferometer team, did the initial conversion of many of the NOVAS Fortran subroutines to C. William T. Harris was largely responsible for completing the conversion, and for NOVAS Version C1.0. David Buscher, James Hilton, Christian Hummel, and Sandra Martinka, users of early C versions of NOVAS, provided valuable comments and suggestions.

References

Hohenkerk, C. Y., Yallop, B. D., Smith, C. A., & Sinclair, A. T. 1992, in *The Explanatory Supplement to the Astronomical Almanac*, ed. P. K. Seidelmann (Mill Valley, CA: Univ. Sci. Books) 95

Kaplan, G. 1990, *Bull. AAS, 22*, 930

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)

Kaplan, G., Bangert, J., Bartlett, J., Puatua, W., & Monet, A. 2009, *User's Guide to NOVAS 3.0*, USNO Circular 180. (Washington, DC: USNO) http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-180

Kaplan, G. H., Hughes, J. A., Seidelmann, P. K., Smith, C. A., & Yallop, B. D. 1989, *AJ*, 97, 1197

Klioner, S. A. 2003, *AJ*, 125, 1580

Seidelmann, K., ed. 1992, *The Explanatory Supplement to the Astronomical Almanac* (Mill Valley, CA: Univ. Sci. Books) (*The Explanatory Supplement*)

Abbreviations and Symbols Frequently Used

ΔAT	TAI – UTC (an integral number of seconds)
ΔT	TT – UT1
θ	Earth Rotation Angle (also ERA)
A&A	*Astronomy and Astrophysics* (journal)
AA	Astronomical Applications, a USNO department
AJ	*The Astronomical Journal* (journal)
ASCII	American Standard Code for Information Interchange
AU	astronomical unit
BCRS	Barycentric Celestial Reference System
Bull. AAS	*Bulletin of the American Astronomical Society* (journal)
c	speed of light
CIO	Celestial Intermediate Origin
CIP	Celestial Intermediate Pole
dec	declination
ECEF	Earth-centered Earth-fixed
EO	Earth Orientation, a USNO department
ERA	Earth Rotation Angle (also θ)
GCRS	Geocentric Celestial Reference System
IAU	International Astronomical Union
ICRS	International Celestial Reference System
IERS	International Earth Rotation and Reference Systems Service
ITRS	International Terrestrial Reference System
J2000.0	epoch 2000 January 1, 12h TT (JD 2451545.0 TT) at the geocenter ("J2000.0 system" is shorthand for the celestial reference system defined by the mean equator and equinox of J2000.0)
JD	Julian date
JPL	Jet Propulsion Laboratory
NOVAS	Naval Observatory Vector Astrometry Software
NRL	Naval Research Laboratory
RA	right ascension
SI	Système International
SOFA	Standards of Fundamental Astronomy (software)
TAI	International Atomic Time (TAI = UTC + ΔAT)
TDB	Barycentric Dynamical Time
TIO	Terrestrial Intermediate Origin
TT	Terrestrial Time (TT = TAI + 32.184 s = UT1 + ΔT)
U.S.	United States
USNO	U.S. Naval Observatory
UT1	"Universal Time 1"; universal time that is a measure of the Earth's rotational angle and subject to irregularities in the Earth's rotation (UT1 = TT – ΔT)
UTC	Coordinated Universal Time, an atomic time scale that is the basis for worldwide civil time (replaces Greenwich Mean Time); currently kept within 0.9 s of UT1 through the addition or subtraction of leap seconds

VLBI	Very Long Baseline Interferometry
WGS-84	World Geodetic System 1984
x_p, y_p	Polar motion components; coordinates of the CIP with respect to the ITRS

(Return to Table of Contents)

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. Such positions can then be expressed in a number of coordinate systems. 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 (RA), declination (dec), hour angle, ecliptic, equinox, precession, and sidereal time should be familiar. Any number of texts on fundamental astronomy—for example, *Spherical Astronomy* (Green 1985)—can 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. USNO Circular 179 documents the algorithms for many important calculations in NOVAS. Others are described in the Kaplan et al. (1989) paper mentioned in the Introduction or in the *Explanatory Supplement*. 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]

A very cursory overview of some of the most important aspects of the astronomical calculations performed by NOVAS follows. In this chapter, names of functions relevant to the subject being discussed are shown in [*brackets*]. In the background material, special terms referred to in IAU resolutions are printed in **bold** when first mentioned; other relevant terms with widely accepted meanings in astronomy are initially printed in *italics*. In descriptions of specific C code, **bold text** will be used to refer to file names and *italic text* will be used to refer to function names. Variable names or code snippets will be presented in a `typewriter-like font`.

1.1 Astronomical Coordinate Systems

Astronomical coordinate systems have traditionally been based on the extension of the Earth's equatorial plane to infinity, along with a fiducial direction in that plane, the *equinox*. The direction of the equinox is along the line of nodes where the equatorial and ecliptic planes meet. Because the Earth's equator and ecliptic are both in motion (the equator's motion is described by precession and nutation; the ecliptic's motion is due to the Earth's orbital variations), an infinite number of such coordinate systems exist, each one corresponding to the orientation of the two planes at a specific date and time. The situation is complicated by the fact that for some purposes in the past it was convenient to consider only precession—and to neglect nutation—in defining celestial coordinate systems, so that we can have, for any given date and time, both a *mean* system (in which only precession is considered) and a *true* system (in which both precession and nutation are taken into account).

[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

Thus we have the mean system of some date, the true system of some date, the mean system of 2000.0, etc. [*precession*, *nutation*, *gcrs2equ*]

Sidereal time is closely tied to the equatorial coordinate systems: one day of sidereal time is marked by successive transits of the equinox across a specific geographic meridian, and local sidereal time is just the apparent right ascension of stars transiting the observer's meridian. Like the equatorial coordinate systems, sidereal time comes in two flavors, *mean* and *apparent*, based on, respectively, the mean and true equinox and the system of right ascension that each defines. [*sidereal_time*]

In the last few decades of the 20th century, several IAU working groups led a general re-examination of astronomical coordinate systems at a very basic level. Part of the motivation was quite practical: most large ground-based telescopes were no longer built on equatorial mounts, and the equatorial coordinate systems were irrelevant to increasingly important space observations. Furthermore, the most precise fundamental measurement—those that by their nature are most closely related to the equatorial systems—are now obtained from Very Long Baseline Interferometry (VLBI), a radio technique that has no direct sensitivity to the equinox. Another important consideration was the need for astronomical coordinate systems that were part of a general relativistic framework. All of these factors were folded into some very important resolutions passed by the IAU in 1997 and 2000, which form the basis for the coordinate systems used in the current version of NOVAS.

These resolutions have introduced concepts and terminology that may not be familiar to astronomers accustomed to the traditional systems and names, which were used in the previous versions of NOVAS. The new systems are outlined in the following paragraphs, but see USNO Circular 179[9] for a much more complete description.

One of the resolutions passed by the IAU in 2000 defined two systems of space-time coordinates, one for the solar system and the other for the near-Earth environment, within the framework of General Relativity, by specifying the form of the metric tensors for each and the 4-dimensional space-time transformation between them. The former is called the **Barycentric Celestial Reference System (BCRS)** and the latter is called the **Geocentric Celestial Reference System (GCRS)**. The BCRS is the system appropriate for the basic ephemerides of solar system objects and astrometric reference data on galactic and extragalactic objects, i.e., the data in astrometric star catalogs. The GCRS is the system appropriate for describing the rotation of the Earth, the orbits of Earth satellites, and geodetic quantities such as instrument locations and baselines. The directions of astronomical objects as seen from the geocenter can also be expressed in the GCRS. The analysis of precise observations inevitably involves quantities expressed in both systems and the transformations between them. Functions in NOVAS may work with BCRS vectors, GCRS vectors, or both with appropriate conversions.

If the orientation of the BCRS axes in space is specified, the orientation of the GCRS axes then follows from the relativistic transformation between the two systems. The orientation of the BCRS is given by what is called the **International Celestial Reference System (ICRS)**.

[9] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

The ICRS is a triad of coordinate axes with their origin at the solar system barycenter and with axis directions effectively defined by the adopted coordinates of about 200 extragalactic radio sources observed by VLBI (listed in Section H of *The Astronomical Almanac*). The abbreviations BCRS and ICRS are often used interchangeably, because the two concepts are so closely related: the ICRS is the orientation of the BCRS; the BCRS is the metric for the ICRS.

The extragalactic radio sources that define the ICRS orientation are assumed to have no observable intrinsic angular motions. Thus, the ICRS is a "space-fixed" system (more precisely, a kinematically non-rotating system) and as such it has no associated epoch—its axes always point in the same directions with respect to distant galaxies. However, the ICRS was set up to approximate the conventional system defined by the Earth's mean equator and equinox of epoch J2000.0; the alignment difference is at the 0.02 arcsecond level, which is negligible for many applications. [*frame_tie*]

Reference data for positional astronomy, such as the data in astrometric star catalogs (e.g., Hipparcos, UCAC, or 2MASS) or barycentric planetary ephemerides (e.g., JPL's DE405) are now specified within the ICRS; more precisely stated, they are specified within the BCRS, with respect to the ICRS axes.

In the near-Earth environment, celestial coordinates and related quantities are expressed with respect to the GCRS or reference systems that are derived from it. Because the orientation of the GCRS is derived from that of the BCRS, the GCRS can be thought of as the "geocentric ICRS."

Besides the GCRS itself, the two reference systems most commonly used for expressing the apparent directions of astronomical objects as seen from the Earth are the true equator and equinox of date and the **Celestial Intermediate Reference System**. Both are obtained from simple rotations of the GCRS, and in both cases, the fundamental plane is the true equator of date. [*gcrs2equ*] In the new terminology, the true or instantaneous equator is the plane orthogonal to the **Celestial Intermediate Pole (CIP)**, which is the celestial pole defined by the adopted precession and nutation algorithms (see section 1.4). The only difference between these two systems is the origin of right ascension: the points on the equator where RA = 0 are, respectively, the true equinox and the **Celestial Intermediate Origin (CIO).** The CIO is a recently introduced fiducial point on the equator that rigorously defines one rotation of the Earth (see section 1.5). [*cio_location*]

The GCRS and the two equatorial systems obtained from it have their origin at the geocenter. For topocentric coordinates or vectors, referred to an observer at a specific location on or near the surface of the Earth, there are analogous coordinate systems, although no semantic distinction is usually made between them and their geocentric equivalents. In NOVAS, the topocentric equivalent of the GCRS is referred to as the "local GCRS," and its spatial axes are assumed to be obtained from the GCRS by a Galilean transformation (simple shift of origin without a change in orientation). The two topocentric equator-of-date systems are obtained similarly.

<u>Reference System for NOVAS Input Data</u>: NOVAS now assumes that input reference data, such as catalog star positions and proper motions, and the basic solar system ephemerides, are provided in the ICRS (that is, within the BCRS as aligned to the ICRS axes), *or at least*

are consistent with it to within the data's inherent accuracy. [*place*, *ephemeris*] The latter case will probably apply to most FK5-compatible data specified with respect to the mean equator and equinox of J2000.0 (the "J2000.0 system"). The distinction between the ICRS and the J2000.0 system becomes important only when an accuracy of 0.02 arcsecond or better is important. [*frame_tie*] Nevertheless, because NOVAS is designed for the highest-accuracy applications, the ICRS is mentioned as the reference system of choice for many input arguments to NOVAS functions.

Reference Systems for NOVAS Output Data: For output coordinates (e.g., the position of Mars on a certain date), three options for the coordinate system are available. (The coordinates themselves can be right ascension and declination or the components of a unit vector.) You can request coordinates expressed in the GCRS, the equator and equinox of date, or the Celestial Intermediate Reference System (equator and CIO of date). These coordinate systems can be requested for either geocentric or topocentric output. [*place*, *gcrs2equ*]

NOVAS can also convert topocentric right ascension and declination, with respect to the equator and equinox of date, to local horizon coordinates, zenith distance and azimuth. The angular shift due to atmospheric refraction can be included as an option. [*equ2hor*] Another function is available to transform right ascension and declination to ecliptic longitude and latitude. [*equ2ecl*] Still another transforms right ascension and declination to galactic longitude and latitude. [*equ2gal*]

Reference System for the Location of the Observer: The location of an Earth-based observer is specified in NOVAS by longitude, latitude, and height with respect to the World Geodetic System 1984 (WGS-84) reference ellipsoid. Coordinates provided by GPS (if uncorrected for the local datum) are referred to WGS-84, which is also sometimes called the Earth-centered Earth-fixed (ECEF) system. The **International Terrestrial Reference System (ITRS)** is a geocentric rectangular coordinate system used for high-precision work. WGS-84 coordinates are functionally equivalent (within a few centimeters) to ITRS coordinates. Thus, the geodetic positions used by NOVAS are consistent with the ITRS. [*geo_posvel*]

1.2 Computing Observable Quantities

NOVAS is mostly used for computing, for a selected object, the instantaneous angular coordinates (or the equivalent unit vector components) at which it might be observed, within one of several user-selected coordinate systems. Obviously the values of the angular coordinates computed by NOVAS depend on the coordinate system requested, but there are several phenomena that affect the observed position of a star or planet that are independent of the coordinate system. For stars, the effects are

- Proper motion (generalized): the three-dimensional space motion of the star, relative to that of the solar system barycenter, between the catalog epoch and the date of interest. Assumed linear and computed from the catalog proper motion components, radial velocity, and parallax. Projected onto the sky, the motion amounts to less than 1 arcsecond per year (usually much less) except for a few nearby stars. [*starvectors*, *proper_motion*]

- Parallax: the change in our perspective on stars in the solar neighborhood due to the position of the Earth in its orbit. Its magnitude is (distance in parsecs)$^{-1}$ and hence is always less than 1 arcsecond. [*bary2obs*]
- Gravitational light bending: the apparent deflection of the light path in the gravitational field of the Sun and (to a much lesser extent) the other planets. Although it reaches 1.8 arcsecond at the limb of the Sun, it falls to 0.05 arcsecond 10º from the Sun and amounts to no more than a few milliarcseconds over the hemisphere of the sky opposite the Sun. [*grav_def*, *grav_vec*]
- Aberration: the change in the apparent direction of light caused by the observer's velocity (v) with respect to the solar system barycenter. Independent of distance, it is approximately equal to v/c, expressed as an angle. Therefore, it can reach 21 arcseconds for observers on the surface of the Earth and somewhat more for instruments in orbit. [*aberration*]
- Atmospheric refraction: the total angular change in the direction of the light path through the Earth's atmosphere; applies only to an observer on or near the surface of the Earth. The direction of refraction is always assumed to be parallel to the local vertical and a function only of zenith distance (although these assumptions may not be true in all cases). At optical wavelengths, its magnitude is zero at the zenith, about 1 arcminute at a zenith distance of 45°, and 0.5° at the horizon. Refraction is roughly proportional to the atmospheric pressure at the observer, but it also depends on other atmospheric parameters and the observational wavelength. [*equ2hor*, *refract*]

The same effects are relevant to objects in the solar system, except that the proper motion calculation is replaced by a function that retrieves the object's barycentric position from its ephemeris, as part of an iterative light-time calculation. [*light_time*, *ephemeris*] Extragalactic objects can be considered to be stars with zero parallax and proper motion. The star or planet positions computed by considering all these effects obviously depend on the location of the observer; so that an observer on the surface of the Earth will see a slightly different position than one at the geocenter, the differences being greater for solar system objects, especially nearby ones (reaching about 1º for the Moon).

In computing these effects, the same functions are used for stars and planets, because NOVAS uses position vectors rather than directions; that is, internally it places all objects at their computed distance from the solar system barycenter. (Objects of unknown distance are placed on the "NOVAS celestial sphere" at a radius of 1 Gigaparsec = 2×10^{14} astronomical units). [*starvectors*] These vectors are all defined within the BCRS until the relativistic aberration calculation is applied, which effectively takes an input vector in the BCRS and produces an output vector in the GCRS.

Nomenclature: When all these effects are included, we obtain star or planet coordinates in the GCRS that reflect where the star or planet actually appears in the sky. The coordinates can then be transformed to other reference systems, if desired. We will call the results of this process, generically, the "apparent position" or "observed position" of the object.

However, some caution with the semantics is in order, because the term *apparent place* has traditionally been reserved specifically for the star or planet position we obtain by applying all these effects (except refraction) for a geocentric observer, with the coordinates expressed with respect to the true equator and equinox of date. For an observer on the surface of the Earth, the corresponding term is *topocentric place*. If the apparent star or planet position is instead expressed with respect to the mean equator and equinox of J2000.0, the terms used in NOVAS have been *virtual place* and *local place*, respectively. (Although these last two terms were suggested in the Kaplan et al. (1989) paper previously cited, there is no evidence that they have been widely used outside of the context of NOVAS.)

The above terminology is reflected in the names of the high-level functions that perform the computations, where "app" stands for "apparent place", "topo" stands for "topocentric place", etc. [*app_star*, *app_planet*, *topo_star*, *topo_planet*, *virtual_star*, *virtual_planet*, *local_star*, *local_planet*] In addition, an *astrometric place* calculation can be used for some differential measurements; it is the same as virtual place except that light bending and aberration (and refraction) are *not* computed, under the assumption that these effects are the same for all objects within a small field of view. [*astro_star*, *astro_planet*]

In response to the introduction of the new IAU-recommended coordinate systems, we must make some adjustments and additions to the nomenclature. The mean equator and equinox of J2000.0, considered as a geocentric system, has been replaced by the GCRS. The IAU Working Group on Nomenclature (2003–2006) recommended that the term *proper place* be used for what is called virtual place in NOVAS. With the introduction of the Celestial Intermediate Reference System, with its right ascension origin at the CIO, we now have two more possibilities for apparent positions, one geocentric and one topocentric. The geocentric coordinates are called the object's *intermediate place*, and right ascension measured with respect to the CIO is called *intermediate right ascension*.

The complete table of nomenclature for apparent positions of various types, updated for NOVAS 3.0, is then

Final Coordinate System	**Observer at Geocenter**	**Observer near Surface of Earth**
True equator and equinox of date	apparent place	topocentric place
Celestial Intermediate Reference System	intermediate place	[no name]
GCRS	proper or virtual place	local place
GCRS	astrometric place*	[no name]

*variant of proper or virtual place, in which some calculations are omitted

The only difference between a position expressed in the Celestial Intermediate Reference System and the same position expressed with respect to the true equator and equinox of date is an offset in right ascension, the *equation of the origins*. The equation of the origins is the angle between the equinox and the CIO, both of which lie on the instantaneous equator. [*ira_equinox*]

NOVAS function *place* can be used to compute all of these types of positions; the input arguments allow you to select both the location of the observer and the coordinate system in which the computed position is to be expressed. These selections make the nomenclature superfluous. *place* does not apply atmospheric refraction, but that can be added by a subsequent call to *equ2hor*.

1.3 Time Scales for Astronomy

As explained in USNO Circular 179,[10] basically two kinds of time scales are used in astronomy, those that are based on the Système International (SI) second ("atomic time") and those that are tied to the irregular rotation of the Earth; in essence, "lab" time and "astronomical" time, respectively. Theoretical time scales, not kept by any real clocks, are also used that are the time basis for—that is, the 4th dimension of—the BCRS and GCRS.

We almost always start with **Coordinated Universal Time (UTC)**, which is the worldwide basis for civil time. UTC (USNO) is obtained from the Global Positioning System (GPS). In addition, UTC can also be obtained with varying accuracy from Network Time Protocol (NTP) services on the Internet and from radio time broadcasts (e.g., WWV, WWV, WWVB, and CHU), cell phones, TV, etc. UTC is based on SI seconds at sea level on the rotating Earth. From UTC, one adds an integral number of (SI) seconds to obtain **International Atomic Time (TAI)**: TAI = UTC + ΔAT, where ΔAT is a total count of leap seconds in UTC (for example, ΔAT = 34 s for 2009). The USNO Earth Orientation (EO) Department[11] provides a complete table of ΔAT values on-line. The current value may also be found in at the beginning of Bulletin A[12] published by the International Earth Rotation and Reference System Service (IERS). The addition of a leap second to UTC, which increases ΔAT by 1 second, is usually done, when needed, at 23:59:59 UTC on December 31 and is announced about six months in advance.

NOVAS Time Arguments: Typically, the first input argument to most NOVAS functions is the time of interest (for example, the time of an observation), expressed as a Julian date (JD). Julian dates are simply a convenient format for representing a date and time in *any* time scale, and are discussed below. Two time scales are used as the basis for most of the Julian dates that are input arguments to the higher-level NOVAS functions.

The first is **Terrestrial Time (TT)**, which is effectively just a constant offset from TAI: $\text{TT} = \text{TAI} + 32.184\ \text{s}$. Therefore, $\text{TT} = \text{UTC} + \Delta\text{AT} + 32.184\ \text{s}$. Historically, TT is considered continuous with the obsolete time scales Ephemeris Time (ET) and Terrestrial Dynamical Time (TDT). It is meant to be a smooth and continuous "coordinate" time scale independent of the rotation of the Earth. The high-level NOVAS functions that compute the apparent direction of an object at a specified time use Julian dates based on TT. [*place*, *app_star*, *topo_star*, *app_planet*, *transform_cat*]

[10] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179
[11] http://maia.usno.navy.mil/ser7/tai-utc.dat
[12] http://www.iers.org/MainDisp.csl?pid=36-9

The second time scale is **Universal Time (UT1)**, which is based on the rotation of the Earth. It is needed for computing sidereal time [*sidereal_time*] or the Earth Rotation Angle (ERA or θ) [*era*], which in turn allows one to compute hour angles, altitude and azimuth, or other topocentric quantities. UT1 is also obtained from UTC: UT1 = UTC + (UT1−UTC). The value of UT1−UTC is available in a daily-interval tabulation on the IERS web site[13] (data marked "P" are predictions); IERS publishes historical values in Bulletin B.[14] The values of UT1−UTC often change at the millisecond level over one day. In computing the topocentric direction of a celestial object with respect to Earth-fixed axes (e.g., altitude and azimuth), 1-arcsecond accuracy in the final angles requires 67-ms accuracy in UT1. Because UT1−UTC can have an absolute value up to 900 ms, it is an important correction for all but the crudest applications; that is, in most cases, it is *not* acceptable to approximate UT1 as being equal to UTC.

A few of the lower-level NOVAS functions use time arguments based on **Barycentric Dynamical Time (TDB)**. [*e_tilt*, *precession*, *ephemeris*, *solarsystem*] TDB differs from TT only by periodic variations (due mainly to the Earth's elliptical orbit and described by General Relativity), the largest of which has an amplitude of 1.6 ms and a period of one year. [*tdb2tt*] The difference between the two time scales can often be neglected in practice and this is noted in the function preambles where appropriate. TDB is equivalent to T_{eph}, the barycentric coordinate time argument of the Jet Propulsion Laboratory (JPL) planetary and lunar ephemerides.

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*]

The epoch **J2000.0** is considered to be an event at the geocenter at Julian date 2451545.0 TT, which is 2000 January 1, 12h TT.

The difference ΔT = TT – UT1, expressed in seconds, is required for certain NOVAS functions that use both TT and UT1 internally but require only one type of input time

[13] http://maia.usno.navy.mil/ser7/mark3.out
[14] http://www.iers.org/MainDisp.csl?pid=36-9
[15] http://www.usno.navy.mil/USNO/astronomical-applications/data-services/jul-date

argument. A table of historical values of ΔT is on pages K9–K10 of *The Astronomical Almanac* and more recent values and predictions can be found online at EO web site.[16] Values of ΔT can also be computed from

$$\Delta T = 32.184\,\mathrm{s} + \Delta AT - (UT1{-}UTC)$$

For example, on 2009 January 1, $\Delta T = 65.7768$ s, which is based on a ΔAT value of 34 s and a UT1-UTC value of +0.407161 s. More information on ΔT is given in section 1.6 below.

1.4 Adopted Models for Precession and Nutation

Astronomers realized over a decade ago that the old standard models for the precession and nutation were in need of revision. The value of the angular rate of precession in longitude adopted by the IAU in 1976—and incorporated into the widely used precession formulation by Lieske and collaborators (1977)—is too large by about 0.3 arcsecond per century (3 milli-arcseconds per year). The amplitudes of a number of the largest nutation components specified in the 1980 IAU Theory of Nutation are also known to be in error by several milliarcseconds. Both the precession and nutation errors are significant relative to current observational capabilities.

Thus, the resolutions passed by the IAU in 2000 mandated an improvement to the precession and nutation formulations. NOVAS 3.0 incorporates the models adopted in response to these resolutions. [*precession*, *nutation*] The precession model is the P03 solution of Capitaine, Wallace, and Chapront (2003), as recommended by the IAU Working Group on Precession and the Ecliptic (Hilton et al. 2006). The P03 precession model was formally adopted by the IAU in 2006. The nutation model is taken from Mathews, Herring, and Buffett (2002). This model, referred to as the IAU 2000A nutation, consists of 1,365 trigonometric terms, more than ten times the number in the previous model. [*nutation_angles*, *iau2000a*] Because evaluation of nutation has always been the most computationally intensive task in NOVAS, you may notice an increase in execution time for some applications.

To reduce execution time, NOVAS 3.0 provides an optional reduced-accuracy mode in which a truncated nutation series is used. This nutation series is specific to NOVAS and is referred to as 2000K. [*nu2000k*] It consists of the largest 488 terms in the IAU 2000A series and provides an accuracy of about 0.1 milliarcsecond (specifically, 0.1 milliarcsecond for $\Delta\psi$ and about 0.04 milliarcsecond for $\Delta\varepsilon$ and $\Delta\psi \sin \varepsilon$). 2000K is the default reduced-accuracy nutation series in both Fortran and C versions of NOVAS, but the C version also includes the 77-term, IAU-approved truncated nutation series, IAU 2000B [*iau2000b*], which is accurate to about 1 milliarcsecond in the interval 1995–2050 (McCarthy & Luzum 2003). More information on the implementation of nutation in NOVAS can be found section 2.6, section 4.3, and in USNO Circular 181,[17] *Nutation Series Evaluation in NOVAS 3.0* (Kaplan 2009).

[16] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/long-term
[17] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-181

As mentioned in section 1.1, the celestial pole described by the new precession and nutation models (with very small observational corrections) is called the **Celestial Intermediate Pole (CIP)**. The true equator of date, also called the instantaneous equator or the intermediate equator, is the plane orthogonal to the direction of the CIP. The CIP is also the rotational pole on the surface of the Earth (see section 1.6).

1.5 New Model for the Rotation of the Earth about its Axis

IAU resolutions passed in 2000 also dealt in a very fundamental way with how the Earth's spin *around* its axis is described. The conventional treatment is based on the equinox and sidereal time; Greenwich (or local) sidereal time is just the Greenwich (or local) hour angle of the equinox of date. However, the equinox is constantly moving due to precession, so that sidereal time combines two angular motions, the Earth's rotation and the precession of its axis. (In the case of apparent sidereal time, nutation is also mixed in.) One rotation of the Earth is about 8 ms longer than one mean sidereal day.

For about two decades, people who routinely deal with the most precise measurements of the Earth's rotation have been advocating for a change in the way it is described. Their ideas were introduced in resolutions passed by the IAU in 2000. In this new paradigm, the reference point on the moving celestial equator for the description of Earth rotation is called the **Celestial Intermediate Origin (CIO)**. Unlike the equinox, this point has no motion along the equator at all; as the orientation of the equator changes in space due to precession and nutation, the CIO remains on the equator but its instantaneous motion is always at right angles to it. [*cio_location*, *cio_ra*] Thus, loosely speaking, two transits of the CIO across a terrestrial meridian define one rotation of the Earth. The CIO is a point on the celestial equator near RA = 0 (in the Celestial Intermediate Reference System, it *defines* RA = 0), and there is a corresponding point on the terrestrial equator near longitude = 0 called the **Terrestrial Intermediate Origin (TIO)**. For all astronomical purposes, the TIO can be considered a point fixed at geodetic longitude zero on the Earth's rotational equator.[18] In the new paradigm, the rotation of the Earth is specified by the angle (in the instantaneous equatorial plane) between the TIO and the CIO, which is a linear function of universal time (UT1). This angle is called the **Earth Rotation Angle (ERA)** and is designated by θ. [*era*]

Some internal calculations in NOVAS can be performed using either the equinox or the CIO as the fundamental fiducial point on the moving astronomical equator. The user can select the basis for these calculations via an input parameter `method`. The results are identical to within a microarcsecond around the current time and the computational burden is about the same.

1.6 Terrestrial-Celestial Relationships

NOVAS uses the ITRS for specifying locations and directions on or near the surface of the Earth. As mentioned at the end of section 1.1, the ITRS is consistent, to within a few

[18] The CIO and TIO are technically examples of *non-rotating origins*, although neither is fixed within its respective coordinate system. However, the slow drift of the TIO, due to polar motion, with respect to standard geodetic coordinates (the ITRS, or, effectively, WGS-84) amounts to only 1.5 millimeters per century and is completely negligible for astronomical purposes.

centimeters, with WGS-84 coordinates provided by GPS, and it is sometimes referred to as the Earth-centered Earth-fixed system (ECEF). The ITRS is a geocentric system with the directions of its axes defined by the coordinates of a large number of observing stations, in a way completely analogous to the definition of the ICRS by the coordinates of extragalactic radio sources. The ITRS z-axis is toward the north geodetic pole and its x-axis is toward a point at longitude and latitude zero; the y-axis forms a right-handed system with the other two axes.

Practical applications of astronomical data often require relating terrestrial coordinates to celestial coordinates and vice versa. For example, we may want the position of a celestial object expressed with respect to the local horizon system. [*equ2hor*] Or, we may have a vector, expressed in an Earth-fixed system, that represents some instrumental axis, and we would like to know where that vector is pointed on the celestial sphere. NOVAS can perform the terrestrial-to-celestial transformation or its inverse; specifically, the transformation from the ITRS to the GCRS, or the GCRS to the ITRS. [*ter2cel*] These transformations are a series of rotations that, taken together, represent the instantaneous orientation of the Earth in space. [*precession*, *nutation*, *cio_basis*, *sidereal_time*, *era*, *wobble*]

Not all aspects of the Earth's orientation are predictable. Polar motion represents the small shift of the geodetic north pole (the ITRS z-axis) with respect to the rotational axis (the CIP), the largest part of which must be determined from observations. Typically, the total shift amounts to a few tenths of an arcsecond (1-2 μrad, 10 meters) and is specified by the parameters x_p and y_p. The observational determinations are designated simply as x and y, and current values are available from IERS Bulletin A.[19] Past values can be obtained at the EO web site.[20] For most purposes, we can set x_p=x and y_p=y (see USNO Circular 179[21] section 6.5.2). Several NOVAS functions require as input the x_p and y_p values for the date of interest, although, if the final accuracy requirements are no better than one arcsecond, these values can be set to zero.

Because the difference $\Delta T = TT - UT1$ is used in only a few internal computations where the conversion from one time scale to another is not critical, the value of ΔT needs to be accurate to only about one second. Therefore, one ΔT value will typically apply to all dates for a given year.

Finally, the new IAU precession and nutation models are neither perfect nor complete, and for very high-accuracy applications, observational corrections are sometimes needed. These corrections now amount to less than one milliarcsecond (5 nrad). These corrections are available from the same sources as the polar motion determinations, and are designated as dX and dY (note that the units are *milli*arcseconds). In the rare cases where they are needed, they are pre-specified to NOVAS for use in subsequent calculations for a specific date. [*cel_pole*]

[19] http://maia.usno.navy.mil/ser7/ser7.dat
[20] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/daily
[21] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

1.7 References

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

Green, R. 1985, *Spherical Astronomy* (New York: Cambridge Univ. Press)

Hilton, J. L. et al. 2006, *Celest. Mech. & Dynamical Astron.*, 94, 351

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

Kaplan, G. H. 2009, *Nutation Series Evaluation in NOVAS 3.0*, USNO Circular 181 (Washington, DC: USNO)
http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-181

Kaplan, G. H., Hughes, J. A., Seidelmann, P. K., Smith, C. A., & Yallop, B. D. 1989, *AJ*, 97, 1197

Lieske, J. H., Lederle, T., Fricke, W., & Morando, B. 1977, *A&A*, 58, 1

Mathews, P. M., Herring, T. A., & Buffett, B. A. 2003, *J. of Geophys. Res.*, 107, ETG 3-1

McCarthy, D. D. & Luzum, B. J. 2003, *Celest. Mech. & Dynamical Astron.*, 85, 37

Seidelmann, K., ed. 1992, *The Explanatory Supplement to the Astronomical Almanac* (Mill Valley, CA: Univ. Science Books) (*The Explanatory Supplement*)

USNO & HMNAO. 2008, *The Astronomical Almanac for the Year 2010*, (Washington, DC: GPO) and subsequent editions

(Return to Table of Contents)

Chapter 2 Installing NOVAS

2.1 List of Distribution Files

The table below lists the fifteen files that comprise the main distribution of NOVAS C3.0.[22] Except for the user's guide, which is in Portable Document Format (PDF), the files are all plain ASCII text. Files with a **.c** extension are C source code; their associated header files are designated by a **.h** extension. The file with a **.f** extension contains Fortran source code.

File name	Description
novas.c	contains all supervisory and utility functions and most basic functions (see Section 4.2 for a full listing)
novas.h	header file for **novas.c** (includes structure definitions and function prototypes)
novascon.c	contains most mathematical and physical constants used by NOVAS
novascon.h	header file for **novascon.c**
nutation.c	contains functions implementing the IAU 2000A, IAU 2000B, and 2000K nutation models
nutation.h	header file for **nutation.c**
solsys1.c	versions of functions *solarsystem* and *solarsystem_hp* that, when used with **eph_manager.c**, provide a complete "all C" interface between NOVAS and JPL's lunar and planetary ephemerides (see detailed discussion in Chapter 4)
solsys2.c	versions of functions *solarsystem* and *solarsystem_hp* that, when used with **jplint.f**, serve as an interface between NOVAS and JPL's Fortran solar system ephemeris-access code (see detailed discussion in Chapter 4)
solsys3.c	versions of functions *solarsystem* and *solarsystem_hp* that provide the position and velocity of the Earth and Sun without reference to an external data file (see detailed discussion in Chapter 4)
solarsystem.h	header file for the **solsys1.c**, **solsys2.c**, and **solsys3.c** files
eph_manager.c	C implementation of JPL's solar system ephemeris-access code for use with **solsys1.c**.

[22] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/novas/novas-c

File name	Description
eph_manager.h	Header file for **eph_manager.c**
readeph0.c	dummy version of function *readeph*, the highest-level call to the USNO/AE98[23] minor planet ephemerides software. When positions of selected minor planets are desired, replace this file with **readeph.c** from the USNO/AE98 package.
jplint.f	Fortran subroutines that serve as the interface between NOVAS and JPL's Fortran ephemeris-access code for use with **solsys2.c**.
NOVAS_C3.0_Guide.pdf	*User's Guide to NOVAS C3.0*, this manual

In addition, the following six files are provided to assist you in validating the installation of NOVAS on your local system. Files with a **.c** extension are the C source code that created the associated ASCII text files ending with **.txt** extensions. Compare the USNO-generated output files provided with the results produced by your local installation as described in Section 2.2 (basic installation), Section 2.3 (including a JPL solar system ephemeris), and Section 2.4 (including a USNO/AE98[24] minor planet ephemeris).

File name	Description
checkout-stars.c	main function that calls functions in **novas.c** and **solsys3.c** to validate a basic local installation
checkout-stars-usno.txt	output from **checkout-stars** application computed at USNO
checkout-stars-full.c	main function that calls functions in **novas.c** and **solsys1.c** with full accuracy to validate a local installation for use with solar system ephemerides; requires binary JPL ephemeris file
checkout-stars-full-usno.txt	output from **checkout-stars-full** application computed at USNO
checkout-mp.c	main function that calls functions in **novas.c**, **solsys1.c**, and USNO/AE98 minor planet software to validate a local installation for use with minor planet ephemerides; requires minor planet ephemeris file and binary JPL ephemeris files
checkout-mp-usno.txt	output from **checkout-mp** application computed at USNO

[23] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/usnoae98
[24] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/usnoae98

The following two files contain sample code to get you started developing your own applications using NOVAS. Chapter 3 discusses each of the examples included and provides instructions for using the sample code. The file with a **.c** extension is the C source code that produced the associated ASCII text file (**.txt)**.

File name	Description
sample.c	main function that calls selected functions in **novas.c** and **solsys1.c**; example code discussed in Chapter 3; requires a binary JPL ephemeris
sample-usno.txt	output from **sample** application computed at USNO

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_ra.c	main function that converts **CIO_RA.TXT** (ASCII) to **cio_ra.bin** (binary)

2.2 Installation and Basic Validation

To install NOVAS and perform a basic validation of the code on your local system, follow the instructions given below. These instructions assume that you know how to compile and link C source code on your computer system. Details of the process are dependent on your particular computer system. In particular, some C compilers require an explicit link to the math library, which is typically accomplished using the `-lm` option in the command line; see your compiler's documentation for details. NOVAS has been successfully implemented on Microsoft Windows systems, Apple Macintosh systems, and Red Hat Enterprise Linux systems.

The most basic validation of NOVAS requires no external ephemeris files.

a. Copy all NOVAS files to a directory on your local system.
b. Compile and link files **checkout-stars.c**, **novas.c**, **novascon.c**, **nutation.c, solsys3.c**, and **readeph0.c**. Name the resulting application **checkout-stars**.
c. Run the **checkout-stars** application.
d. Compare the results that you get from **checkout-stars** with the data in file **checkout-stars-usno.txt**. If the results agree, the installation was probably successful, but see the important note below.

Important Note

The **checkout-stars** application exercises one supervisory function and most, but not all, of the low-level functions in **novas.c**. The use of the **checkout-stars** application is not a complete test of NOVAS. Comparing the results from the NOVAS C supervisory functions with results from the analogous NOVAS Fortran supervisory functions will constitute a more complete check of your NOVAS implementation.

This setup of NOVAS, using *solarsystem* version 3, will provide the positions of stars or extragalactic objects with errors not exceeding 1.5 milliarcseconds (if your input catalog data is that good) or the Sun with an error not exceeding 2 arcseconds. However, it will not provide the positions of solar system objects other than the Sun. For more general or demanding applications, NOVAS must use another version of *solarsystem*.

2.3 Using External Solar System Ephemeris Files

NOVAS must have access to a solar system ephemeris, which provides NOVAS with the heliocentric and barycentric positions and velocities of desired solar system objects referred to the ICRS. A solar system ephemeris is required even when only precise star positions are needed—in that case, the "desired solar system objects" are the Earth and Sun. Thus, an ephemeris of the barycentric Earth and the barycentric Sun is the minimum requirement.

NOVAS C3.0 accesses such data for solar system objects through function *ephemeris*. As provided, *ephemeris* supports access to a JPL ephemeris of the major solar system bodies (here defined as Sun, Moon, eight planets, and Pluto) and provides direct support for access to the USNO Ephemerides of the Largest Asteroids (Hilton 1999; hereafter USNO/AE98[25]). Function *ephemeris* accesses the JPL ephemerides by calling the appropriate version of *solarsystem* or *solarsystem_hp*.

If you need to compute star or radio-source positions to better than 1.5 milliarcseconds, positions of the Sun to better than 2 arcseconds, or positions of solar system bodies other than the Sun, you will have to use function *solarsystem* version 1 or version 2, which require external ephemerides, instead of *solarsystem* version 3, which is self-contained.

solarsystem version 1, along with functions in **eph_manager.c**, provides a complete C language interface between NOVAS and one of the JPL "development ephemerides" (DEnnn), such as DE405. The *Planet_Ephemeris* function is the C version of the Fortran subroutine *PLEPH*, and it, in turn, calls other functions in the JPL ephemeris software package. Alternatively, *solarsystem* version 2, along with subroutines in **jplint.f**, provides an interface to the JPL ephemeris access code, which is available in Fortran. Subroutine *jplint* contains a single call to JPL's Fortran subroutine *PLEPH*, which in turn calls other Fortran subroutines in the JPL ephemeris software package.

Establishing a working copy of the JPL software and the DEnnn binary files on your system is not, unfortunately, a trivial process. The files for doing that can be obtained directly from JPL[26] as discussed in Appendix C.

Once you have generated a DEnnn binary file, you can test the combined set-up with *solarsystem* version 1 on your system as follows:

a. Compile and link files **checkout-stars-full.c**, **novas.c**, **novascon.c**, **nutation.c,** **solsys1.c**, **eph_manager.c**, and **readeph0.c**.
b. Name the resulting application **checkout-stars-full**.

[25] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/usnoae98
[26] http://ssd.jpl.nasa.gov/?planet_eph_export

c. Run the **checkout-stars-full** application.
d. Compare the results that you get from **checkout-stars-full** with the data in file **checkout-stars-full-usno.txt**. If the results agree, the installation was probably successful, but see the important note above.

The USNO output file, **checkout-stars-full-usno.txt**, was generated using DE405. The application, **check-out-stars-full**, uses the full-accuracy mode of NOVAS, which includes the IAU 2000A nutation model, a three-body gravitational deflection model, two-part Julian dates in calls to function *ephemeris*, and the full series when computing the "complementary terms" in the equation of the equinoxes [*iau2000a, nutation_angles, grav_def, ephemeris, ee_ct*]. If your accuracy requirements are no better than about 1 milliarcsecond, you may wish to consider "reduced-accuracy" mode, which is described in Section 2.6, for your own applications.

2.4 Using External Minor Planet Ephemeris Files

The USNO/AE98[27] minor planet ephemerides contain positions for fifteen of the largest asteroids and are used in the production of *The Astronomical Almanac*. These ephemerides along with software to compact, read, and interpolate them are available on CD-ROM from the USNO. The software is written in C and designed to be used with NOVAS. To include these ephemeris files in your NOVAS programs, you will need to install a working copy of the USNO/AE98 software and convert the relevant ASCII ephemerides to binary Chebyshev polynomial ephemerides. You will still need one version of *solarsystem*; *solarsystem* version 1 with access to the JPL lunar and planetary ephemeris is recommended. Instructions[28] for installing and testing the USNO/AE98 software and for converting ephemerides are provided with the software. Section 2.3 and Appendix C discuss the use and installation of a JPL lunar and planetary ephemeris.

In order to access the USNO/AE98 minor planet ephemerides, function *ephemeris* calls function *readeph*, which is part of the USNO/AE98 package and is not part of, or supplied with, NOVAS. A dummy version of *readeph* is provided in file **readeph0.c**. The dummy function enables NOVAS to be used without the minor planet package (i.e., for computing positions of major solar system bodies and "stars" only). To use USNO/AE98 with NOVAS, replace file **readeph0.c** provided with NOVAS, with **readeph.c**, **allocate.c**, and **chby.c** from the USNO minor planet ephemerides software when compiling and linking.

Once you have a DEnnn binary solar system ephemeris file, the USNO/AE98 software, and a binary Chebyshev polynomial ephemeris of a minor planet, you can test the combined set-up with *solarsystem* version 1 on your system as follows:

a. Compile and link files **checkout-mp.c**, **novas.c**, **novascon.c**, **nutation.c**, **solsys1.c**, **eph_manager.c**, **readeph.c**, **allocate.c**, and **chby.c**. The last three code files are from the USNO/AE98 software.
b. Name the resulting application **checkout-mp**.
c. Run the **checkout-mp** application.

[27] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/usnoae98
[28] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/usnoae98/ae98-rm

d. Compare the results that you get from **checkout-mp** with the data in file **checkout-mp-usno.txt**. If the results agree, the installation was probably successful, but see the important note above.

The USNO output file, **checkout-mp-usno.txt**, was generated using DE405 from JPL and **pallas.chby** from the minor planet ephemerides. Like the **check-out-stars-full** application, **checkout-mp** uses the full-accuracy mode of NOVAS. If you do not require accuracy better than about 1 milliarcsecond, you may wish to consider the "reduced-accuracy" mode discussed in Section 2.6.

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_ra.c** to a directory on your local system.
b. Compile **cio_ra.c.**
c. Name the resulting application **cio_ra**.
d. Run the **cio_ra** 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.

The file **CIO_RA.TXT**, as supplied, contains six centuries of data, most of which is seldom used, so trimming it down can reduce the file size. It is plain text (ASCII) that can be modified by any text editor. The first record is a header, which should remain as such (although its contents are not used by NOVAS), but all the other records contain a Julian date and information for that date in ascending date order. Simply trim from the beginning and ending of the file any records for dates that you are *sure* you will never need (be sure to leave at least ten extra dates on each end to allow the internal interpolation scheme to work properly). For example, **CIO_RA.TXT** contains data for years 1700 through 2300 but you may need only the range from 2010 to 2015. Leave the data within your overall anticipated date range as is—that is, do not attempt to make several groups of dates within the same file. The final file that you create must be a fixed-interval list running from the first date to the last date with no gaps after the header record. When you have modified **CIO_RA.TXT** appropriately, follow the steps above to convert your custom version to the binary direct-access format useable by NOVAS. The output message and file will differ from that above based on the changes you made.

If a CIO file (**cio_ra.bin**) is not present when NOVAS needs to determine the location of the CIO, NOVAS will simply revert to using an internal computation for this information. The results differ by a few microarcseconds at most, and those differences are reached only for dates before 1900 or after 2200. The two approaches represent independent algorithms for determining the location of the CIO on the equator, and the tiny differences for dates that are several centuries from now are of no practical consequence. Small differences in execution time may occur for the two approaches, but those timing differences are likely to vary with the specific application—depending, for example, on the order and spread of the dates that are processed by NOVAS. For some applications using such an external file may be undesirable. Because, in many simple NOVAS applications, the location of the CIO is never needed, the best scheme is probably to start *without* using the CIO file. If your application's execution time is critical, you might want to experiment to see whether using a CIO file affects its performance one way or the other.

2.6 Reduced-accuracy Mode

NOVAS has "full-accuracy" and "reduced-accuracy" modes that must be selected when calling about half of the functions in **novas.c**; see the prolog to each function and the corresponding description in Chapter 4 for specific functions. Each of those functions has a short integer input parameter, `accuracy`. If `accuracy` is set to zero (0), the function and any lower-level functions that it calls will proceed with full accuracy. Alternatively, if `accuracy` is set to one (1), the function and any lower-level functions that it calls will proceed with reduced accuracy.

In full-accuracy mode,

- nutation calculations use the IAU 2000A model [*iau2000a, nutation_angles*];
- gravitational deflection is calculated using three bodies: Sun, Jupiter, and Saturn [*grav_def*];
- the equation of the equinoxes includes the entire series when computing the "complementary terms" [*ee_ct*];
- geocentric positions of solar system bodies are adjusted for light travel time using split, or two-part, Julian dates in calls to *ephemeris* and iterate with a convergence tolerance of 10^{-12} days [*light_time, ephemeris*];
- *ephemeris* calls the appropriate solar system ephemeris using split, or two-part, Julian dates primarily to support light-time calculations [*ephemeris, solarsystem_hp, light_time*].

In reduced-accuracy mode,

- nutation calculations use the 2000K model, which is the default for this mode, or the IAU 2000B model, which may be set manually by user [*nu2000k, iau2000b, nutation_angles*];
- gravitational deflection is calculated using only one body, the Sun [*grav_def*];

- the equation of the equinoxes excludes terms smaller than 2 microarcseconds when computing the "complementary terms" [*ee_ct*];
- geocentric positions of solar system bodies are adjusted for light travel time using single-value Julian dates in calls to *ephemeris* and iterate with a convergence tolerance of 10^{-9} days [*light-time, ephemeris, solarsystem*];
- *ephemeris* calls the appropriate solar system ephemeris using single-value Julian dates [*ephemeris, solarsystem*].

In full-accuracy mode, the IAU 2000A nutation series (1,365 terms) is used [*iau2000a*]. Evaluating the series for nutation is usually the main computational burden in NOVAS, so using reduced-accuracy mode improves execution time, often noticeably. In reduced-accuracy mode, the NOVAS 2000K nutation series (488 terms) is used by default [*nu2000k*]. This mode can be used when the accuracy requirements are not better than 0.1 milliarcsecond for stars or 3.5 milliarcseconds for solar system bodies. Selecting this approach can reduce the time required for Earth-rotation computations by about two-thirds. However, if your reduced-accuracy application requires the IAU-approved truncated nutation series, IAU 2000B (77 terms), you can edit the function *nutation_angles* slightly so that it calls *iau2000b* in reduced-accuracy mode instead of *nu2000k*; the necessary changes are described in the prolog to that function. Section 1.4 and USNO Circular 181[29] (Kaplan 2009) provide some additional information about nutation series.

2.7 References

Hilton, J. L. 1999, *AJ*, 117, 1077 http://www.usno.navy.mil/USNO/astronomical-applications/software-products/usnoae98 (USNO/AE98)

Kaplan, G. H. 2009, *Nutation Series Evaluation in NOVAS 3.0*, USNO Circular 181 (Washington, DC: USNO)
http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-181

USNO & HMNAO. 1998, *The Astronomical Almanac for the Year 2000*, (Washington, DC: GPO) and subsequent editions

(Return to Table of Contents)

[29] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-181

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.

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).

3.1 Initialization

You may choose to call many NOVAS functions in either full- or reduced-accuracy modes. Usually all functions in a single application will have the same accuracy requirements. Therefore, you may wish to define a short integer constant for this purpose at the beginning of your main function. **sample** operates in full-accuracy mode, so `accuracy` is set to zero

```
const short int accuracy = 0;
```

A value of one (1) would have set reduced accuracy. For more information on the two modes, see section 2.6.

Two high-level NOVAS functions *sidereal_time* and *ter2cel* can perform their internal calculations using either the CIO-based method or equinox-based method. If you will be calling these functions multiple times, you may also wish to define a short integer constant for this purpose, with a value of zero (0) for the CIO-based method or one (1) for the equinox-based method. In **sample**, the appropriate method is chosen when these functions are called. Because the equinox-based method is more efficient for computing sidereal time, **sample** chooses this mode. For more information about these two methods, see section 5.2.

3.2 Setting Time Arguments

You should also consider how you will handle dates and times. As described in section 1.3, NOVAS uses either of two time scales, TT or UT1, as the basis for the time input argument to its higher-level functions. However, you may be working with UTC, Coordinated Universal Time. UTC is the basis for civil time systems worldwide and, because it is distributed quite accurately by GPS, is often used as the time-tag for observations. The key relationship is

$$\mathrm{TT} = \mathrm{UTC} + \Delta\mathrm{AT} + 32.184\mathrm{s}$$

where ΔAT is an integer representing the total count of leap seconds in UTC (for example, ΔAT = 34 s for 2009). Equivalently,

$$\mathrm{TT} = \mathrm{TAI} + 32.184\mathrm{s}$$

where TAI is International Atomic Time. If you will only be dealing with the geocentric celestial coordinates of objects, TT is all you will need.

If you will also be computing topocentric coordinates (for a specific location on the surface of the Earth), you will also need to obtain UT1. The key relationship is

$$\mathrm{UT1} = \mathrm{UTC} + (\mathrm{UT1}–\mathrm{UTC})$$

where UT1–UTC is interpolated from the daily values of this quantity published by the IERS.[30] Alternatively,

$$\mathrm{UT1} = \mathrm{TT} – \Delta\mathrm{T}$$

See section 1.3 for more information on time scales, including sources of data that can be used for the values of ΔAT, UT1–UTC, and ΔT.

To convert a date and time in the common format (year/month/day/hour/minute/second) to a Julian date, which is used by NOVAS for time arguments, call *julian_date*. Dates and times based on UTC, TT, or UT1 (or any other time scale) can be converted using *julian_date*; the output Julian date simply has the same basis as the input date and time. In the examples below, we will be using 2008 April 24, 10:36:18.0 UTC as the time of interest; this corresponds to a Julian date of 2454580.9441875 UTC.[31] Consequently, we will use a ΔAT value of 33 s and a UT1–UTC value of −0.387845 s, which are appropriate for this date.

```
const short int year = 2008;
const short int month = 4;
const short int day = 24;
const short int leap_secs = 33;

const double hour = 10.605;
const double ut1_utc = -0.387845;

const double x_pole = -0.002;
const double y_pole = +0.529;
```

[30] http://maia.usno.navy.mil/ser7/mark3.out

[31] A "UTC Julian date" is something of a non sequitur, because UTC is not continuous due to leap seconds. Here we are simply using *julian_date* with UTC input to obtain a value that allows us to compute Julian dates in better-behaved time scales.

where `x_pole` and `y_pole` are the coordinates of the CIO with respect to the ITRS pole for 2008 April 24. So, if we use

```
jd_utc = julian_date (year,month,day,hour);
```

the output argument, `jd_utc`, will have a value of `2454580.9441875`. The next few lines of code should be something like

```
jd_tt = jd_utc + ((double) leap_secs + 32.184) / 86400.0;
jd_ut1 = jd_utc + ut1_utc / 86400.0;
delta_t = 32.184 + leap_secs - ut1_utc;
```

where `86400.0` is the number of seconds in a day, and the value of `delta_t` would be computed to be `65.571845` (seconds). If we had known the value of ΔT (`delta_t`) at the start, rather than the value of `ut1_utc` (UT1–UTC), then the lines immediately above could have been replaced by

```
jd_tt = jd_utc + ((double) leap_secs + 32.184) / 86400.0;
jd_ut1 = jd_tt - delta_t / 86400.0;
```

3.3 Example 1—Position of a Star

Suppose we have the catalog data from star FK6 1307 (Groombridge 1830) for epoch J2000.0, expressed in the ICRS:

ICRS right ascension at J2000.0 (hours):	`11.88299133`
ICRS declination at J2000.0 (degrees):	`37.71867646`
Proper motion in RA (milliarcsceonds per year):	`4003.27`
Proper motion in dec (milliarcseconds per year):	`-5815.07`
Parallax (milliarcseconds):	`109.21`
Radial velocity (kilometers per second):	`-98.8`

To obtain the apparent geocentric place of the star on our date of interest, with respect to the equator and equinox of that date, first make a structure of type `cat_entry`,

```
make_cat_entry ("GMB 1830","FK6",1307,11.88299133,37.71867646,4003.27,-
5815.07,
   109.21,-98.8, &star);
```

Then, simply call *app_star* supplying it with the catalog quantities:

```
error = app_star (jd_tt,&star,accuracy,&ra,&dec)
```

In this example and other examples in this chapter, the returned value from the function—`error`—is an error indicator. Non-zero values of `error` indicate an error condition inside the function; see the function prolog for details. The calling function should take appropriate action in such cases. The output coordinates, `ra` and `dec` (hours and degrees, respectively), represent the apparent geocentric coordinates of the star, with respect to the true equator and equinox of date. The computation takes into account all time-dependent effects that shift the star's position from its catalog coordinates (except atmospheric refraction, which is location- and weather-dependent): the star's space motion to the date of interest, parallax due to the Earth's position in its orbit, gravitational light-bending in the solar system, aberration due to the Earth's orbital velocity, and the precession and nutation of the Earth's axis.

Important note: Hipparcos catalog data, although expressed with respect to the ICRS, refer to epoch 1991.25 and must be converted to epoch J2000.0 before being used as input arguments to any NOVAS function. Use function *transform_hip* to do this. Most other modern catalogs, including the FK6 (used above), Tycho-2, UCAC, etc., provide data for epoch J2000.0 that need no conversion.

If we want the apparent topocentric place of the star, that is, the star's position as it would be seen (except for refraction) from a particular location on Earth, such as off the Atlantic coast near Truro, Massachusetts

```
const double latitude = 42.0;
const double longitude = -70;
const double height = 0.0;
const double temperature = 10.0;
const double pressure = 1010.0;
```

where `longitude` and `latitude` are the location's geodetic longitude and latitude (degrees, with east longitude and north latitude positive) and `height` is the height of the location above sea level (meters). The `temperature` and `pressure` (in degrees Celsius and millibars, respectively) are only used in calculations involving atmospheric refraction; here, the values are simply placeholders. You should create a structure of type `on_surface` for that location

```
make_on_surface (latitude,longitude,height,temperature,pressure,&geo_loc);
```

Now, call *topo_star*

```
error = topo_star (jd_tt,delta_t,&star,&geo_loc,accuracy,&rat,&dect)
```

where `rat` and `dect` reflect the position of the star as it would be seen from that particular location—the small differences from the geocentric coordinates `ra` and `dec` arise mainly from diurnal aberration. *topo_star* uses the catalog data on the star created earlier.

3.4 Example 2—Position of the Moon

Obtaining the coordinates of solar system objects other than the Sun requires that either *solarsystem* version 1 or version 2 be used. First, you will need to create a structure of type `cat_entry` for a "dummy" star, which is, then, used as a placeholder in a structure of type `object` for the Moon.

```
make_cat_entry ("DUMMY","xxx",0,0.0,0.0,0.0,0.0,0.0,0.0,&dummy_star);

error = make_object (0,11,"Moon",&dummy_star,&moon)
```

Then, call function *app_planet*

```
error = app_planet (jd_tt,&moon,accuracy,&ra,&dec,&dis)
```

where, again, `ra` and `dec` are the apparent geocentric coordinates of the Moon (hours and degrees, respectively), with respect to the true equator and equinox of date, and `dis` is the true (Euclidean) geocentric distance (astronomical units or AU) at time `jd_tt`. The computation of the angular coordinates of solar system objects takes light-time into account, along with the other effects (light bending, aberration, precession, and nutation) that also apply to stars.

To get the topocentric celestial coordinates of the Moon, call *topo_planet*

```
error = topo_planet(jd_tt,&moon,delta_t,&geo_loc,accuracy,&rat,&dect,&dist)
```

However, a single function, *place*, can be used for all types of positions of both stars and solar system objects. In fact, *app_star*, *topo_star*, *app_planet*, and *topo_planet*, along with several other similar functions, are actually just special-purpose front-ends to *place*. *place* uses three structure arguments (`cel_object` of type `object` and `location` of type `observer` for input and `output` of type `sky_pos` for output) as well as several scalar arguments. The `cel_object` of type `object` is same as was created earlier for use by *app_planet* and *topo_planet* in calculating the position of the Moon. Function *make_observer_on_surface* creates a structure of type `observer` for an observer on or near the surface of the Earth:

```
make_observer_on_surface (latitude,longitude,height,temperature,pressure,
   &obs_loc);
```

The call to *place* to obtain topocentric coordinates of the Moon, with respect to the true equator and equinox of date, then is

```
error = place (jd_tt,&moon,&obs_loc,delta_t,1,accuracy,&t_place)
```

where short integer `coord_sys` has been set to one (1) so that the output coordinates in `t_place`, a structure of type `sky_pos`, will be based on the true equator and equinox of date:

```
typedef struct
{
      double r_hat[3];
      double ra;
      double dec;
      double dis;
      double rv;
} sky_pos;
```

Here, `double r_hat[3]` is a dimensionless unit vector in the apparent direction of the Moon, in the same coordinate system as the right ascension and declination (i.e., it is exactly equivalent to the spherical coordinates). *place* has many options for both input and output; refer to its description in Chapter 4 or look at its prolog.

Once you have the topocentric celestial coordinates of an object, these can be transformed into local altitude and azimuth by a call to *equ2hor*. If we have used *place* to obtain the topocentric coordinates, then

```
equ2hor (jd_ut1,delta_t,accuracy,0.0,0.0,&geo_loc,rat,dect,1,&zd,&az,&rar,
   &decr);
```

where the refraction option, short integer `ref_option`, is set here to one (1) and the x- and y-coordinates of the CIO with respect to the ITRS pole, double `x` and double `y`, have been set to zero (0.0). The refraction option selected here is for “standard atmospheric” conditions. The other options for refraction are zero (0) for no refraction or two (2) for refraction based on the atmospheric conditions indicated by `location`, a structure of type `on_surface`. Setting the CIO coordinates to zero is appropriate when sub-arcsecond accuracy is unnecessary; otherwise these arguments should contain the appropriate pole coordinates for the date of interest. The output coordinates, `zd`, `az`, `rar`, and `decr`, are, respectively, the zenith distance (degrees), azimuth (degrees), right ascension (hours), and declination (degrees). The output values of `zd`, `rar`, and `decr` are affected by atmospheric refraction for refraction options 1 and 2. If `ref_option` equals 0, the output right ascension and declination values are the same as the input values. `zd` and `az` are referred to the horizon system that is tangent to the Earth's reference ellipsoid at the observer's location; that is, the deflection of the vertical (the local undulation of the geoid) is not taken into account.

3.5 Example 3—Greenwich Sidereal Time

To obtain Greenwich sidereal time, call *sidereal_time*:

```
error = sidereal_time (jd_ut1,0.0,delta_t,1,1,accuracy,&gast)
```

where short integer `gst_type` and short integer `method` have each been set to one (1) to compute Greenwich apparent sidereal time using the equinox-based method. In this example, the output sidereal time `gast` is Greenwich apparent sidereal time in hours. If `gst_type` had been set to zero (0), the output sidereal time would have been Greenwich mean sidereal time in hours. Choosing between equinox-based and CIO-based methods is discussed in section 3.1 and section 5.2.

sidereal_time and several other functions allow for a "split" input UT1 Julian date—high- and low-order parts in the first two arguments—for increased precision. Generally, the split would be at the Julian date's decimal point, with the day count in the first argument and the fraction of a day in the second. However, using two arguments for the Julian date provides more precision *only if the fractional part of the Julian date has been handled separately all along*. We have not done that, so here, the entire Julian date is just placed in the first argument and the second argument is set to zero (0.0).

To compute local sidereal time (either mean or apparent), add the longitude (east positive) expressed in hours:

```
last = gast + geo_loc.longitude / 15.0;
```

The result may have to be reduced to the range 0 to 24 hours by statements similar to the following:

```
if (last >= 24.0)
   last -= 24.0;
if (last < 0.0)
   last += 24.0;
```

The quantity that is analogous to Greenwich apparent sidereal time in the CIO-based paradigm is θ, the ERA. It can be computed

```
theta = era (jd_ut1,0.0);
```

where `theta` is the ERA (degrees). *era*, like *sidereal_time*, allows for a split input UT1 Julian date.

See section 5.1 for information on the difference between Greenwich apparent sidereal time and the ERA, and how hour angles are computed in the two paradigms.

3.6 Example 4—Other Frequently Requested Quantities

In the following function calls, vectors are used. Vectors are simply double-precision arrays with a dimension of 3. Most NOVAS internal calculations are performed with vectors and matrices. The following vectors are referred to below:

```
double pos[3], vel[3], pose[3], vter[3], vcel[3];
```

To obtain the barycentric or heliocentric coordinates (BCRS vectors) of a solar system body, for example, Mars, first create an appropriate structure of type `object`

```
error = make_object (0,4,"Mars",&dummy_star,&mars)) != 0
```

which uses the `dummy_star` structure created in Example 2. Then, call *ephemeris*:

```
jd_tdb = jd_tt;     /* Approximation good to 0.0017 seconds. */
jd[0] = jd_tdb;
jd[1] = 0.0;
error = ephemeris (jd,&mars,1,accuracy,pos,vel)
```

where short integer `origin` has been set to one (1) to obtain a heliocentric position, we have approximated `jd_tdb = jd_tt`,[32] and the output position and velocity vectors are `pos` and `vel` (components in AU and AU/day, respectively). For barycentric positions, set `origin` to zero (0).

If `pos` is a heliocentric vector, it can be transformed to the ecliptic system (fixed ecliptic of J2000.0) by calling *equ2ecl_vec*:

```
error = equ2ecl_vec (T0,2,accuracy,pos,pose)
```

where `pose` is the output vector in the ecliptic system (same units as `pos`). `pose` could then easily be converted to heliocentric spherical coordinates: ecliptic longitude, ecliptic latitude, and radius vector.

```
error = vector2radec (pose,&elon,&elat)
elon *= 15.0;
r = sqrt (pose[0] * pose[0] + pose[1] * pose[1] + pose[2] * pose[2]);
```

Finally, transforming a vector from the terrestrial reference system to the celestial reference system can be useful. The vector might represent a geographic position, a geodetic reference line or direction, or an instrumental axis. For this transformation, the vector starts out as an Earth-fixed vector expressed with respect to the ITRS axes. For example, the vector toward the local vertical (orthogonal to the ellipsoid at the place of interest) is simply ($\cos\phi \cos\lambda$, $\cos\phi \sin\lambda$, $\sin\phi$), where ϕ is the geodetic latitude and λ is the longitude. A vector along a telescope's polar axis would nominally point toward (0,0,1) in this system.

```
lon_rad = geo_loc.longitude * DEG2RAD;
lat_rad = geo_loc.latitude * DEG2RAD;
sin_lon = sin (lon_rad);
cos_lon = cos (lon_rad);
sin_lat = sin (lat_rad);
```

[32] Strictly, *ephemeris* requires a TDB-based Julian date as input. TDB differs from TT by at most 1.7 ms; see section 1.3. If this time difference is important, use function *tbd2tt* to determine the difference between the two time scales and adjust `jd_tdb` accordingly. For the example here, neglecting the difference can lead to an error in the position of Mars of about 50 meters.

```
cos_lat = cos (lat_rad);
vter[0] = cos_lat * cos_lon;
vter[1] = cos_lat * sin_lon;
vter[2] = sin_lat;
```

Any such ITRS vector can be transformed into the equivalent GCRS vector with a single call to *ter2cel*:

```
error = ter2cel (jd_ut1,0.0,delta_t,1,accuracy,0,x_pole,y_pole,vter,vcel)
```

where `vter` is the input vector (terrestrial, ITRS) and `vcel` is the equivalent output vector (celestial, GCRS). The components of `vter` can be in any units; `vcel` will be in the same units. `vcel` will sweep around the celestial sphere as the Earth rotates, i.e., as `jd_ut1` advances. (*ter2cel*, like *sidereal_time*, allows for a split input UT1 Julian date.) Use *vector2radec* to obtain `vcel`'s instantaneous spherical coordinates:

```
error = vector2radec (vcel,&ra,&dec)
```

where `ra` and `dec` are the GCRS right ascension and declination (hours and degrees, respectively) of the point on the celestial sphere toward which `vcel` points. At any `jd_ut1`, `vcel` can be compared to the directions of stars computed by *place* for the equivalent `jd_tt`, with GCRS output coordinates selected.

(Return to Table of Contents)

Chapter 4 Data Structures and Functions

NOVAS can be used several different ways. Some users will simply want to adopt a subset of the basic or utility functions for use in their own code systems. For these users, the function prologs should be sufficient for providing the information needed to use the functions. Other users will want to implement the supervisory functions to compute, for example, apparent places of stars or topocentric places of solar system bodies. This section of the user's guide is intended for these users.

4.1 Important Data Structures

Seven important data structures are used throughout NOVAS. They are formally declared in file **novas.h**.

Structure `cat_entry`

Structure `cat_entry` contains the astrometric catalog data for a celestial object; equator and equinox and units will depend on the catalog. While this structure can be used as a generic container for catalog data, all high-level NOVAS functions require ICRS catalog data with the appropriate units, which are shown in parentheses below.

```
typedef struct
{
   char starname[51];
   char catalog[4];
   long int starnumber;
   double ra;
   double dec;
   double promora;
   double promodec;
   double parallax;
   double radialvelocity;
} cat_entry;
```

where:

```
   starname[51]       = name of celestial object
   catalog[4]         = 3-character catalog designator
   starnumber         = integer identifier assigned to object
   ra                 = ICRS right ascension (hours)
   dec                = ICRS declination (degrees)
   promora            = ICRS proper motion in right ascension
                        (milliarcseconds/year)
   promodec           = ICRS proper motion in declination
                        (milliarcseconds/year)
   parallax           = parallax (milliarcseconds)
   radialvelocity     = radial velocity (km/s)
```

Structure `object`

Structure `object` designates a celestial object.

```
typedef struct
{
   short int type;
   short int number;
   char name[51];
   cat_entry star;
} object;
```

where:

```
   type                 = type of object
                        = 0 ... major planet, Sun, or Moon
                        = 1 ... minor planet
                        = 2 ... object located outside the solar system
                                (star, nebula, galaxy, etc.)
   number               = object number
                          For 'type' = 0: Mercury = 1, ..., Pluto = 9,
                                          Sun = 10, Moon = 11
                          For 'type' = 1: minor planet number
                          For 'type' = 2: set to 0 (object is
                          fully specified in 'struct cat_entry')
   name                 = name of the object (limited to 50 characters)
   star                 = basic astrometric data for any celestial object
                          located outside the solar system; the catalog
                          data for a star
```

Structure `on_surface`

Structure `on_surface` contains data for the observer's location. The weather parameters (temperature and pressure) are used only by the refraction function (*refract*) called from function *equ2hor* when `ref_option = 2`; dummy values can be used otherwise. Parameters can be added to this structure, if a more sophisticated refraction model is substituted.

```
typedef struct
{
   double latitude;
   double longitude;
   double height;
   double temperature;
   double pressure;
} on_surface;
```

where:

```
   latitude              = geodetic (ITRS) latitude; north positive (degrees)
   longitude             = geodetic (ITRS) longitude; east positive (degrees)
   height                = height of the observer (meters)
   temperature           = temperature (degrees Celsius)
   pressure              = atmospheric pressure (millibars)
```

Structure `in_space`

Structure `in_space` contains data for an observer's position and velocity in a near-Earth spacecraft.

```
typedef struct
{
   double sc_pos[3];
   double sc_vel[3];
} in_space;
```

where:

```
   sc_pos[3]          = geocentric position vector (x, y, z), components
                        in km
   sc_vel[3]          = geocentric velocity vector (x_dot, y_dot,
                        z_dot), components in km/s

                        Both vectors with respect to true equator and
                        equinox of date
```

Structure `observer`

Structure `observer` is a general container for information specifying the location of the observer.

```
typedef struct
{
   short int where;
   on_surface on_surf;
   in_space near_earth;
} observer;
```

where:

```
   where              = integer code specifying location of observer
                        = 0: observer at geocenter
                        = 1: observer on surface of earth
                        = 2: observer on near-earth spacecraft
   on_surface         = structure containing data for an observer's
                        location on the surface of the Earth (where = 1)
   near_earth         = data for an observer's location on a near-Earth
                        spacecraft (where = 2)
```

Structure `sky_pos`

Structure `sky_pos` contains data specifying a celestial object's place on the sky; contains the output from function *place*.

```
typedef struct
{
   double r_hat[3];
   double ra;
   double dec;
   double dis;
   double rv;
} sky_pos;
```

where:

```
   r_hat[3]            = unit vector toward object (dimensionless)
   ra                  = apparent, topocentric, or astrometric
                         right ascension (hours)
   dec                 = apparent, topocentric, or astrometric
                         declination (degrees)
   dis                 = true (geometric, Euclidian) distance to solar
                         system body or 0.0 for star (AU)
   rv                  = radial velocity (km/s)
```

Structure `ra_of_cio`

Structure `ra_of_cio` contains the right ascension of the Celestial Intermediate Origin (CIO) with respect to the GCRS.

```
typedef struct
{
   double jd_tdb;
   double ra_cio;
} ra_of_cio;
```

where:

```
   jd_tdb              = TDB Julian date
   ra_cio              = right ascension of the CIO with respect
                         to the GCRS (arcseconds)
```

4.2 Function List

The following functions are contained in file **novas.c**:

Function name	Level	Purpose
place	super	Computes apparent direction of a star or solar system body, given the time and observer's location. Direction is expressed in one of several selectable coordinate systems.
sidereal_time	super	Computes Greenwich sidereal time, either mean or apparent
ter2cel	super	Transforms arbitrary vector in rotating Earth-fixed (ITRS) system to space-fixed (ICRS) system, terrestrial to celestial transformation
equ2hor	super	Transforms topocentric RA and dec to zenith distance and azimuth, optionally accounts for atmospheric refraction.
transform_cat	super	Transforms star's catalog quantities for a change of epoch and/or equator and equinox
transform_hip	super	Converts Hipparcos catalog data at epoch J1991.25 to epoch J2000.0
app_star	super	Computes apparent place of a star, given its catalog data
topo_star	super	Computes topocentric place of a star, given geodetic location of observer
virtual_star	super	Computes virtual place of a star, given its catalog data
local_star	super	Computes local place of a star, given geodetic location of observer
astro_star	super	Computes astrometric place of a star, given its catalog data
mean_star	super	Computes ICRS/J2000.0 place of a star, given its apparent place
app_planet	super	Computes apparent place of a planet or other solar system body
topo_planet	super	Computes topocentric place of a planet or other solar system body, given geodetic location of observer
virtual_planet	super	Computes virtual place of a planet or other solar system body

Function name	Level	Purpose
local_planet	super	Computes local place of a planet or other solar system body, given geodetic location of observer
astro_planet	super	Computes astrometric place of a planet or other solar system body
aberration	util	Adjusts position vector for aberration of light due to motion of Earth
bary2obs	util	Changes origin of coordinates from barycenter of solar system to center of mass of Earth
cio_basis	util	Computes orthonormal basis vectors, with respect to GCRS, of right-handed system defined by CIP (z-direction) and CIO (x-direction)
d_light	util	For a star, returns difference in light-time between solar system barycenter and observer. Or, returns the light-time from observer to point on light ray closest to a given solar system body.
ecl2equ_vec	util	Converts ecliptic position vector to an equatorial position vector
equ2ecl	util	Converts RA and dec to ecliptic longitude and latitude
equ2ecl_vec	util	Converts equatorial position vector to an ecliptic position vector
equ2gal	util	Converts ICRS RA and dec to galactic longitude and latitude
era	util	Returns ERA (θ) for given UT1 Julian date
frame_tie	util	Transforms vector between dynamical reference system (mean equator and equinox of J2000.0) and ICRS
gcrs2equ	util	Converts GCRS RA and dec to coordinates with respect to the equator of date (mean or true)
geo_posvel	util	Computes geocentric position and velocity, in GCRS, of observer on or near the surface of Earth
grav_def	util	Computes total gravitational deflection of light for an object due to major solar system bodies
grav_vec	util	Corrects position vector for deflection of light in gravitational field of given body
light_time	util	Computes position of a solar system body antedated for light-time
limb_angle	util	Determines angle of object above or below Earth's limb (horizon)
make_cat_entry	util	Creates a structure of type `cat_entry` containing catalog data for a star or "star-like" object
make_object	util	Creates structure of type `object`—specifying a celestial object - based on the input parameters
make_observer	util	Creates structure of type `observer`—specifying the location of the observer
make_observer_at_geocenter	util	Creates structure of type `observer`—specifying that the "observer" is at the geocenter
make_observer_in_space	util	Creates structure of type `observer`—specifying the position and velocity of an observer situated on a near-Earth spacecraft
make_observer_on_surface	util	Creates structure of type `observer`—specifying the location of and weather for an observer on the surface of the Earth
make_in_space	util	Creates structure of type `in_space`—specifying the position and velocity of an observer situated on a near-Earth spacecraft
make_on_surface	util	Creates structure of type `on_surface`—specifying the location of and weather for an observer on the surface of the Earth
nutation	util	Applies nutation to position vector
precession	util	Applies precession to position vector
proper_motion	util	Updates position vector of a star to allow for its space motion
rad_vel	util	Predicts radial velocity of observed object as would be measured by spectroscopy
radec2vector	util	Converts RA, dec, and distance to a position vector
spin	util	Rotates vector by specified angle about the z-axis

Function name	**Level**	**Purpose**
starvectors	util	Converts a star's RA, dec, proper motion, etc., to position and velocity vectors
terra	util	Converts geodetic coordinates to geocentric position vector
vector2radec	util	Converts position vector to RA and dec
wobble	util	Adjusts Earth-fixed vector for polar motion
cal_date	basic	Computes calendar date and time, given Julian date
cel_pole	basic	Allows for the specification of celestial pole offsets for high-precision applications
cio_array	basic	Returns array of Julian dates and corresponding values of RA of CIO (in GCRS), given TDB Julian date and number of dates desired
cio_location	basic	Returns location of CIO as RA with respect to either GCRS origin or true equinox of date, given TDB Julian date
cio_ra	basic	Computes true right ascension of the CIO, given TT Julian date
ee_ct	basic	Returns complementary terms for equation of the equinoxes
ephemeris	basic	Retrieves position and velocity of a solar system body from a fundamental ephemeris
e_tilt	basic	Provides information on orientation of Earth's axis: obliquity, nutation parameters, etc.
fund_args	basic	Computes fundamental arguments (mean elements) of the Sun and Moon
ira_equinox	basic	Computes intermediate RA of the equinox, given TDB Julian date
julian_date	basic	Computes Julian date, given calendar date and time
mean_obliq	basic	Computes the mean obliquity of the ecliptic
norm_ang	basic	Normalize angle into the range 0 <= angle < (2 * pi)
nutation_angles	basic	Supervises calculation of nutation parameters and provides nutation in longitude and obliquity
refract	basic	Computes atmospheric refraction in zenith distance
tdb2tt	basic	Converts Barycentric Dynamical Time (TDB) to Terrestrial Time (TT).

The following functions are contained in file **nutation.c**:

Function name	**Level**	**Purpose**
iau2000a	basic	Evaluates high-accuracy nutation series (IAU 2000A model)
iau2000b	basic	Evaluates low-accuracy nutation series (IAU 2000B model)
nu2000k	basic	Evaluates low-accuracy nutation series (2000K, truncated version of IAU 2000A model)

4.3 Important Functions

Descriptions of some of the most frequently used NOVAS functions follow. The prologs in the source code at the beginning of each NOVAS function are intended to provide enough information for correct usage. They are reproduced here followed by additional discussion and recommendations.

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. (1997), 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).
```

```
RETURNED
VALUE:
   = 0         ... No problems.
   = 1         ... invalid value of 'coord_sys'
   = 2         ... invalid value of 'accuracy'
   = 3         ... Earth is the observed object, and the observer is
                   either at the geocenter or on the Earth's surface
                   (not permitted)
   > 10, < 40  ... 10 + error from function 'ephemeris'
   > 40, < 50  ... 40 + error from function 'geo_posvel'
   > 50, < 70  ... 50 + error from function 'light_time'
   > 70, < 80  ... 70 + error from function 'grav_def'
   > 80, < 90  ... 80 + error from function 'cio_location'
   > 90, < 100 ... 90 + error from function 'cio_basis'
```

Discussion:

This function computes the apparent direction of a star or solar system body at a specified time and in a specified coordinate system. The word "star" as used here refers to any object outside the solar system.

The apparent direction of a star computed by this function takes into account the star's proper motion (linear three-dimensional space motion) from the catalog epoch to the date requested, parallax, gravitational deflection of light by solar system bodies (mostly the Sun), and aberration. The same effects are computed for solar system bodies, except that the proper motion calculation is replaced by an algorithm that retrieves the object's barycentric position from its ephemeris, as part of an iterative light-time calculation. Extragalactic objects are treated as stars with zero proper motion and parallax. The result in all cases is an apparent direction expressed in the GCRS, which is optionally transformed into either of two other output coordinate systems, as specified by argument `coord_sys`.

"Astrometric place" is a variant of the above calculation that is appropriate for some types of differential measurements. Light bending and aberration are ignored under the assumption that they are the same for all objects within a small area of the sky. Astrometric places are expressed in the ICRS.

The observer's location may be at the geocenter, on or near the surface of the Earth, or in orbit around the Earth, as specified by the `where` member of the argument `location`.

place does not take into account atmospheric refraction (which would be appropriate only for observers on or near the surface of the Earth), but its effect can be added by a subsequent call to function *equ2hor*.

For stars, the required input data, stored in the `cel_object->star` argument, are the standard five astrometric quantities from a catalog, together with radial velocity if known. Any parameter should be set to `0.0` if its value is unknown. All catalog data used as input to this function must apply to epoch J2000.0 and be expressed with respect to the ICRS. (For Hipparcos catalog data, see function *transform_hip*.) Extragalactic objects should be treated as stars, but with all input parameters set to `0.0` except for the catalog right ascension and declination. For solar system bodies, the argument `cel_object->number` must contain the identification number from the list of objects supported by the ephemeris in use.

The values of location->where and coord_sys for various kinds of place are listed below.

location->where	**coord_sys**	**Type of Place**
0	0	virtual place* = proper place
1	0	local place*
0	1	apparent place
1	1	topocentric place
0	2	intermediate place
1	2	topocentric intermediate place*
0	3	astrometric place
1	3	topocentric astrometric place*

*Place name not widely recognized outside of NOVAS.

NOVAS functions *app_star*, *topo_star*, *app_planet*, *topo_planet*, etc., are now just special-purpose front-ends to *place*.

Important: The input value of ΔT (delta_t) is used only when location->where = 1 or 2 (observer is on surface of Earth or in a near-Earth satellite). An error in ΔT of 1 s can result in a topocentric place error of up to 0.3 arcsecond for the Moon, but proportionally less for more distant bodies (e.g., 3 milliarcseconds for Venus at its closest). Distance errors of up to 500 m (3×10^{-9} AU) can also result, independent of distance. If errors of this magnitude are important, care needs to be taken in specifying a more accurate value of ΔT. An error in ΔT of 1 s will not result in a significant error in the topocentric places of stars. Values of ΔT are published annually in *The Astronomical Almanac* or can be obtained from the EO web site.[33]

output->rv, the radial velocity, is the predicted radial velocity measure (z) times the speed of light, an inherently spectroscopic quantity. For a star, it includes all effects, such as gravitational red shift, contained in the catalog barycentric radial velocity measure, which is assumed given in cel_object->star.radialvelocity. For a solar system body, it applies to a fictitious emitter at the center of the observed object, assumed to be massless (no gravitational red shift), and does not in general apply to reflected light.

“Loose” catalog data can be assembled into a structure of type cat_entry by using function *make_cat_entry*. Similarly, we recommend using function *make_object* to create the input structure cel_object.

(Return to Function List)

[33] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/long-term

sidereal_time

```
short int sidereal_time (double jd_high, double jd_low,
                         double delta_t,short int gst_type,
                         short int method, short int accuracy,

                         double *gst)

   PURPOSE:
      Computes the Greenwich apparent sidereal time, at Julian date
      'jd_high' + 'jd_low'.

   REFERENCES:
      Kaplan, G. (2005), US Naval Observatory Circular 179.

   INPUT
   ARGUMENTS:
      jd_high (double)
         High-order part of UT1 Julian date.
      jd_low (double)
         Low-order part of UT1 Julian date.
      delta_t (double)
         Difference TT-UT1 at 'jd_high'+'jd_low', in seconds
         of time.
      gst_type (short int)
         = 0 ... compute Greenwich mean sidereal time
         = 1 ... compute Greenwich apparent sidereal time
      method (short int)
         Selection for method
            = 0 ... CIO-based method
            = 1 ... equinox-based method
      accuracy (short int)
         Selection for accuracy
            = 0 ... full accuracy
            = 1 ... reduced accuracy

   OUTPUT
   ARGUMENTS:
      *gst (double)
         Greenwich apparent sidereal time, in hours.

   RETURNED
   VALUE:
      (short int)
         = 0         ... everything OK
         = 1         ... invalid value of 'accuracy'
         = 2         ... invalid value of 'method'
         > 10, < 30  ... 10 + error from function 'cio_rai'
```

Discussion:
This function computes Greenwich sidereal time, either mean, if `gst_type = 0`, or apparent, if `gst_type = 1`.

The input Julian date, which must be in the UT1 time scale, may be split into two parts to ensure the highest precision in the computation. For example, set `jd_high` equal to the integral part of the Julian date and set `jd_low` equal to the fractional part. Generally, this split will be advantageous only if the low-order part has been treated separately within the calling program; for example, if the time of day has been stored in its own variable(s), from which `jd_low` is constructed.

For many applications, the position of the split is not critical as long as the sum `jd_high + jd_low` is correct: in particular, when used with computers providing 16 decimal digits of precision in double variables, this function will yield values of `gst` precise to about 0.1 millisecond even if `jd_high` contains the entire Julian date and `jd_low = 0.0`.

Values of ΔT (`delta_t`) are published annually in *The Astronomical Almanac* or can be obtained from the EO web site.[34]

If `gst_type = 1` for apparent sidereal time, the output value of `gst` will correctly reflect the celestial pole offset in longitude if function *cel_pole* has previously been called.

(Return to Function List)

[34] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/long-term

ter2cel

```
short int ter2cel (double jd_ut_high, double jd_ut_low, double delta_t,
                   short int method, short int accuracy, short int option,
                   double x, double y, double *vect,

                   double *vecc)

   PURPOSE:
      This function rotates a vector from the terrestrial to the
      celestial system.  Specifically, it transforms a vector in the
      ITRF (rotating earth-fixed system) to the GCRS (a local space-
      fixed system) by applying rotations for polar motion, Earth
      rotation, nutation, precession, and the dynamical-to-GCRS
      frame tie.

   REFERENCES:
      Kaplan, G. H. et. al. (1989). Astron. Journ. 97, 1197-1210.
      Kaplan, G. H. (2003), 'Another Look at Non-Rotating Origins',
         Proceedings of IAU XXV Joint Discussion 16.

   INPUT
   ARGUMENTS:
      jd_ut_high (double)
         High-order part of UT1 Julian date.
      jd_ut_low (double)
         Low-order part of UT1 Julian date.
      delta_t (double)
         Value of Delta T (= TT - UT1) at the input UT1 Julian date.
      method (short int)
         Selection for method
            = 0 ... CIO-based method
            = 1 ... equinox-based method
      accuracy (short int)
         Selection for accuracy
            = 0 ... full accuracy
            = 1 ... reduced accuracy
      option (short int)
            = 0 ... The output vector is referred to GCRS axes.
            = 1 ... The output vector is produced with respect to the
                    equator and equinox of date.
      x (double)
         Conventionally-defined X coordinate of celestial intermediate
         pole with respect to ITRF pole, in arcseconds.
      y (double)
         Conventionally-defined Y coordinate of celestial intermediate
         pole with respect to ITRF pole, in arcseconds.
      vect[3] (double)
         Position vector, geocentric equatorial rectangular coordinates,
         referred to ITRF axes (terrestrial system) in the normal case
         where 'option' = 0.
```

```
OUTPUT
ARGUMENTS:
   vecc[3] (double)
      Position vector, geocentric equatorial rectangular coordinates,
      referred to GCRS axes (celestial system) or with respect to
      the equator and equinox of date, depending on 'option'.

RETURNED
VALUE:
   =  0  ... everything is ok.
   =  1  ... invalid value of 'accuracy'
   =  2  ... invalid value of 'method'
   > 10 ... 10 + error from function 'cio_location'
   > 20 ... 20 + error from function 'cio_basis'
```

Discussion:
This function rotates a vector from the terrestrial to the celestial system. Specifically, it transforms a vector in the ITRS (a rotating Earth-fixed system) to the GCRS (a local space-fixed system) by applying rotations for polar motion, Earth rotation, nutation, precession, and the dynamical-to-GCRS frame tie. The input vector might represent a cardinal direction at the observer's position, a geodetic baseline, or some instrumental axis. The units for the vector components are arbitrary and the output vector will have the same units as the input vector. Geodetic coordinates in the WGS-84 system, also sometimes called the Earth-centered Earth-fixed (ECEF) system, can be considered to be compatible with the ITRS.

This function allows for the input UT1 time to be represented as a split Julian date. See the discussion in the description of function *sidereal_time*. Both `jd_ut_high` and `jd_ut_low` should be non-negative for normal use; `jd_ut_low = 0.0` is acceptable.

Values of ΔT (`delta_t`) are published annually in *The Astronomical Almanac* or can be obtained from the EO web site.[35]

The `option` flag only works for the equinox-based method.

Set `x=y=0` to omit the polar motion rotation.

(Return to Function List)

[35] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/long-term

equ2hor

```
void equ2hor (double jd_ut1, double delta_t, short int accuracy,
              double x, double y, on_surface *location, double ra,
              double dec, short int ref_option,

              double *zd, double *az, double *rar, double *decr)

   PURPOSE:
      This function transforms topocentric right ascension and
      declination to zenith distance and azimuth.  It uses a method
      that properly accounts for polar motion, which is significant at
      the sub-arcsecond level.  This function can also adjust
      coordinates for atmospheric refraction.

   REFERENCES:
      Kaplan, G. (2008). USNO/AA Technical Note of 28 Apr 2008,
         "Refraction as a Vector."

   INPUT
   ARGUMENTS:
      jd_ut1 (double)
         UT1 Julian date.
      delta_t (double)
         Difference TT-UT1 at 'jd_ut1', in seconds.
      accuracy (short int)
         Selection for method and accuracy
            = 0 ... full accuracy
            = 1 ... reduced accuracy
      x (double)
         Conventionally-defined x coordinate of celestial intermediate
         pole with respect to ITRS reference pole, in arcseconds.
      y (double)
         Conventionally-defined y coordinate of celestial intermediate
         pole with respect to ITRS reference pole, in arcseconds.
      *location (struct on_surface)
         Pointer to structure containing observer's location (defined
         in novas.h).
      ra (double)
         Topocentric right ascension of object of interest, in hours,
         referred to true equator and equinox of date.
      dec (double)
         Topocentric declination of object of interest, in degrees,
         referred to true equator and equinox of date.
      ref_option (short int)
         = 0 ... no refraction
         = 1 ... include refraction, using 'standard' atmospheric
                 conditions.
         = 2 ... include refraction, using atmospheric parameters
                 input in the 'location' structure.
```

```
OUTPUT
ARGUMENTS:
   *zd (double)
      Topocentric zenith distance in degrees, affected by
      refraction if 'ref_option' is non-zero.
   *az (double)
      Topocentric azimuth (measured east from north) in degrees.
   *rar (double)
      Topocentric right ascension of object of interest, in hours,
      referred to true equator and equinox of date, affected by
      refraction if 'ref_option' is non-zero.
   *decr (double)
      Topocentric declination of object of interest, in degrees,
      referred to true equator and equinox of date, affected by
      refraction if 'ref_option' is non-zero.

RETURNED
VALUE:
   None.
```

Discussion:
This function takes the topocentric celestial coordinates of an object and computes the equivalent local horizon coordinates. The function uses a method that properly accounts for polar motion, which is significant at the sub-arcsecond level. Atmospheric refraction can be included in the transformation, and if so, refraction is applied to both sets of coordinates (this can be useful for telescope pointing). Refraction, when requested, is computed by function *refract*.

`ra` and `dec`, the input topocentric right ascension and declination, can be obtained from *place* (with `location->where = 1` and `coord_sys = 1`), or *topo_star* or *topo_planet*. `jd_ut1`, is the UT1 time at which the topocentric place was computed. The difference TT–UT1 (often called ΔT) is passed to the function via argument `delta_t`. Values of ΔT are published annually in *The Astronomical Almanac* or can be obtained from the EO web site.[36]

The coordinates of the pole, `x` and `y`, can be obtained from IERS Bulletins A and B,[37] although `x` and `y` can be set to zero (0.0) if sub-arcsecond accuracy is not needed. (If refraction is requested, sub-arcsecond accuracy is unlikely.)

The height of the observer and meteorological conditions at the observer, contained in structure `location`, are used only for refraction, i.e., if `ref_option` is not equal to zero. In this function, the directions `zd = 0.0` (the zenith) and `az = 0.0` (north) are considered fixed in the terrestrial frame. Specifically, the zenith is along the geodetic normal, and north is toward the ITRS reference pole.

If `ref_option = 0` for no refraction, then `rar = ra` and `decr = dec`.

(Return to Function List)

[36] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/long-term
[37] http://www.iers.org/MainDisp.csl?pid=36-9

transform_cat

```
short int transform_cat (short int option, double date_incat,
                         cat_entry *incat, double date_newcat,
                         char newcat_id[4],

                         cat_entry *newcat)

   PURPOSE:
      To transform a star's catalog quantities for a change of epoch
      and/or equator and equinox.  Also used to rotate catalog
      quantities on the dynamical equator and equinox of J2000.0 to the
      ICRS or vice versa.

   REFERENCES:
      None.

   INPUT
   ARGUMENTS:
      option (short int)
         Transformation option
            = 1 ... change epoch; same equator and equinox
            = 2 ... change equator and equinox; same epoch
            = 3 ... change equator and equinox and epoch
            = 4 ... change equator and equinox J2000.0 to ICRS
            = 5 ... change ICRS to equator and equinox of J2000.0
      date_incat (double)
         TT Julian date, or year, of input catalog data.
      *incat (struct cat_entry)
         An entry from the input catalog, with units as given in
         the struct definition (struct defined in novas.h).
      date_newcat (double)
         TT Julian date, or year, of transformed catalog data.
      newcat_id[4] (char)
         Three-character abbreviated name of the transformed catalog.

   OUTPUT
   ARGUMENTS:
      *newcat (struct cat_entry)
         The transformed catalog entry, with units as given in
         the struct definition (struct defined in novas.h).

   RETURNED
   VALUE:
      = 0 ... Everything OK.
      = 1 ... Invalid value of an input date for option 2 or 3.
```

Discussion:
Function *transform_cat* performs mean place to mean place transformations on star catalog data. Only catalog reference data, not observed quantities, should be processed by this function.

For `option = 1, 2`, or `3`, two dates, `date_incat` and `date_newcat`, must be specified: the input data is associated with the first date, and the output data is associated with the second date. Two transformations are available:

`option = 1`: The star's data is updated to account for the star's space motion between the first and second dates, within a fixed reference system. That is, the *epoch* of the data is changed, but not the equator and equinox (or other system).

`option = 2`: The reference frame within which the star's coordinates and proper motion are expressed is rotated corresponding to precession between the first and second dates. The star's position in space is not changed. That is, the *equator and equinox* of the data are changed, but not the epoch.

Setting `option = 3` requests both transformations and is the most common case.

The two date arguments, `date_incat` and `date_newcat`, may be specified either as a Julian date (e.g., `2433282.5`) or a Julian year and fraction (e.g., `1950.0`). (Values less than `10000.0` are assumed to represent years.)

The `option = 1` and `option = 3` transformations are appropriate only for objects with linear (or no) space motion; do not use them for components of binary stars. Also, this function cannot be properly used to bring data from old star catalogs into the modern system, because old catalogs were compiled using a set of constants (in particular, the rate of precession) that are incompatible with modern values.

The `option = 2` and `option = 3` transformations involve the dynamical system, that is, the moving mean equator and equinox. The mean equator and equinox of J2000.0 was the most common reference system for modern astrometric catalog data before the ICRS was introduced in 1998. Now, catalog data is usually referred to the ICRS, which is a reference system fixed with respect to distant extragalactic objects, not defined by any Earth motions and with no associated date. The `option = 4` transformation is used to convert catalog quantities from the mean equator and equinox of J2000.0 (the "J2000.0 system") to the ICRS. The `option = 5` transformation is the opposite. The arguments `date_incat` and `date_newcat` are ignored for these transformations.

Function *frame_tie* can be used to transform vectors from the J2000.0 system to the ICRS or vice versa.

See function *transform_hip* for transforming Hipparcos catalog data to epoch J2000.0.

(Return to Function List)

transform_hip

```
void transform_hip (cat_entry *hipparcos,

                    cat_entry *hip_2000)

   PURPOSE:
      To convert Hipparcos catalog data at epoch J1991.25 to epoch
      J2000.0, for use within NOVAS.  To be used only for Hipparcos or
      Tycho stars with linear space motion.  Both input and output data
      is in the ICRS.

   REFERENCES:
      None.

   INPUT
   ARGUMENTS:
      *hipparcos (struct cat_entry)
         An entry from the Hipparcos catalog, at epoch J1991.25, with
         all members having Hipparcos catalog units.  See Note 1
         below (struct defined in novas.h).

   OUTPUT
   ARGUMENTS:
      *hip_2000 (struct cat_entry)
         The transformed input entry, at epoch J2000.0.  See Note 2
         below (struct defined in novas.h).

   RETURNED
   VALUE:
      None.

   NOTES:
      1. Input (Hipparcos catalog) epoch and units:
         Epoch: J1991.25
         Right ascension (RA): degrees
         Declination (Dec): degrees
         Proper motion in RA: milliarcseconds per year
         Proper motion in Dec: milliarcseconds per year
         Parallax: milliarcseconds
         Radial velocity: kilometers per second (not in catalog)
      2. Output (modified Hipparcos) epoch and units:
         Epoch: J2000.0
         Right ascension: hours
         Declination: degrees
         Proper motion in RA: milliarcseconds per year
         Proper motion in Dec: milliarcseconds per year
         Parallax: milliarcseconds
         Radial velocity: kilometers per second
```

Discussion:
This function takes Hipparcos catalog data, which is published for epoch J1991.25, and transforms it to epoch J2000.0 for use in NOVAS functions such as *place, app_star, topo_star, virtual_star,* etc. Note that the Hipparcos (input) right ascension is expressed in degrees, as in the catalog, while the J2000.0 (output) right ascension is given in hours, compatible with other NOVAS functions. Function *transform_cat* (with `option = 1`) is called to perform the epoch transformation. The reference frame for both input and output is the ICRS.

This function should be used only for Hipparcos stars with linear space motion.

Radial velocity (`hipparcos->radialvelocity`) is not given in the Hipparcos catalog. If a value is not known, set `hipparcos->radialvelocity = 0.0`. The radial velocity is important for only a small number of nearby, high-proper-motion stars.

(Return to Function List)

app_star

```
short int app_star (double jd_tt, cat_entry *star, short int accuracy,

                    double *ra, double *dec)

   PURPOSE:
      Computes the apparent place of a star at date 'jd_tt', given its
      catalog mean place, proper motion, parallax, and radial velocity.

   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 apparent place.
      *star (struct cat_entry)
         Pointer to catalog entry structure containing catalog data for
         the object in the ICRS (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)
         Apparent right ascension in hours, referred to true equator
         and equinox of date 'jd_tt'.
      *dec (double)
         Apparent declination in degrees, referred to true equator
         and equinox of date 'jd_tt'.

   RETURNED
   VALUE:
      (short int)
          =  0 ... Everything OK.
          > 10 ... Error code from function 'make_object'.
          > 20 ... Error code from function 'place'.
```

Discussion:
This function computes the apparent place of a star for time `jd_tt`. The word “star” as used here refers to any object outside the solar system. If the values of `promora`, `promodec`, `parallax`, or `radialvelocity` within structure `star` are unknown (or zero within the errors of measurement), the calling program should set them to zero. For extragalactic objects, these input values should also be set to zero.

app_star works by calling *place* with `location->where = 0` and `coord_sys = 1`.

"Loose" catalog data can be assembled into a structure of type `cat_entry` by using function *make_cat_entry*.

(Return to Function List)

topo_star

```
short int topo_star (double jd_tt, double delta_t, cat_entry *star,
                     on_surface *position, short int accuracy,

                     double *ra, double *dec)

   PURPOSE:
      Computes the topocentric place of a star at date 'jd_tt', given its
      catalog mean place, proper motion, parallax, and radial velocity.

   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.
      delta_t (double)
         Difference TT-UT1 at 'jd_tt', in seconds of time.
      *star (struct cat_entry)
         Pointer to catalog entry structure containing catalog data for
         the object in the ICRS (defined in novas.h).
      *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 'jd_tt'.
      *dec (double)
         Topocentric declination in degrees, referred to true equator
         and equinox of date 'jd_tt'.

   RETURNED
   VALUE:
      (short int)
           =  0 ... Everything OK.
           =  1 ... Invalid value of 'where' in structure 'location'.
           > 10 ... Error code from function 'make_object'.
           > 20 ... Error code from function 'place'.
```

Discussion:
This function computes the topocentric place of a star (neglecting atmospheric refraction) for the location specified by the argument `location`, for time `jd_tt`. Note that `jd_tt` is the TT time at which the topocentric place is to be computed. The word "star" as used here refers to any object outside the solar system. If the values of `promora`, `promodec`, `parallax`, or `radialvelocity` within structure `star` are unknown (or zero within the errors of measurement), the calling program should set them to zero. For extragalactic objects, these input values should also be set to zero. The difference TT–UT1 (often called ΔT) is passed to the function via argument `delta_t`. Values of ΔT are published annually in *The Astronomical Almanac* or can be obtained from the EO web site.[38]

Atmospheric refraction can be subsequently applied to `ra` and `dec` by function *equ2hor*.

topo_star works by calling *place* with `location->where = 1` and `coord_sys = 1`.

"Loose" catalog data can be assembled into a structure of type `cat_entry` by using function *make_cat_entry*.

(Return to Function List)

[38] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/long-term

virtual_star

```
short int virtual_star (double jd_tt, cat_entry *star,
                        short int accuracy,

                        double *ra, double *dec)
/*
------------------------------------------------------------------------

   PURPOSE:
      Computes the virtual place of a star at date 'jd_tt', given its
      catalog mean place, proper motion, parallax, and radial velocity.

   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 virtual place.
      *star (struct cat_entry)
         Pointer to catalog entry structure containing catalog data for
         the object in the ICRS (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)
         Virtual right ascension in hours, referred to the GCRS.
      *dec (double)
         Virtual declination in degrees, referred to the GCRS.

   RETURNED
   VALUE:
      (short int)
          =  0 ... Everything OK.
          > 10 ... Error code from function 'make_object'.
          > 20 ... Error code from function 'place'.
```

Discussion:
See the discussion for function *app_star*. Function *virtual_star* is identical to *app_star* in input arguments and use. Here, however, the output arguments provide the virtual place (also called the proper place) of the star. The virtual place (proper place) is essentially the apparent place expressed in the GCRS.

virtual_star works by calling *place* with `location->where = 0` and `coord_sys = 0`.

(Return to Function List)

local_star

```
short int local_star (double jd_tt, double delta_t, cat_entry *star,
                      on_surface *position, short int accuracy,

                      double *ra, double *dec)

   PURPOSE:
      Computes the local place of a star at date 'jd_tt', given its
      catalog mean place, proper motion, parallax, and radial velocity.

   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 local place.
      delta_t (double)
         Difference TT-UT1 at 'jd_tt', in seconds of time.
      *star (struct cat_entry)
         Pointer to catalog entry structure containing catalog data for
         the object in the ICRS (defined in novas.h).
      *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)
         Local right ascension in hours, referred to the 'local GCRS'.
      *dec (double)
         Local declination in degrees, referred to the 'local GCRS'.

   RETURNED
   VALUE:
      (short int)
           =  0 ... Everything OK.
           =  1 ... Invalid value of 'where' in structure 'location'.
           > 10 ... Error code from function 'make_object'.
           > 20 ... Error code from function 'place'.
```

Discussion:
See the discussion for function *topo_star*. Function *local_star* is identical to *topo_star* in input arguments and use. Here, however, the output arguments provide the local place of the star. The local place is essentially the topocentric place expressed in the “local GCRS”.

local_star works by calling *place* with `location->where = 1` and `coord_sys = 0`.

(Return to Function List)

astro_star

```
short int astro_star (double jd_tt, cat_entry *star, short int accuracy,

                      double *ra, double *dec)

   PURPOSE:
      Computes the astrometric place of a star at date 'jd_tt', given
      its catalog mean place, proper motion, parallax, and radial
      velocity.

   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 astrometric place.
      *star (struct cat_entry)
         Pointer to catalog entry structure containing catalog data for
         the object in the ICRS (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)
         Astrometric right ascension in hours (referred to the ICRS,
         without light deflection or aberration).
      *dec (double)
         Astrometric declination in degrees (referred to the ICRS,
         without light deflection or aberration).

   RETURNED
   VALUE:
      (short int)
          =  0 ... Everything OK.
          > 10 ... Error code from function 'make_object'.
          > 20 ... Error code from function 'place'.
```

Discussion:
See the discussion for function *app_star*. Function *astro_star* is identical to *app_star* in input arguments and use. Here, however, the output arguments provide the astrometric place of the star in the ICRS.

astro_star works by calling *place* with `location->where = 0` and `coord_sys = 3`.

(Return to Function List)

app_planet

```
short int app_planet (double jd_tt, object *ss_body, short int accuracy,

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

   PURPOSE:
      Compute the apparent place of a planet or other 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 apparent place.
      *ss_body (struct object)
         Pointer to structure containing the body designation for the
         solar system body (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)
         Apparent right ascension in hours, referred to true equator
         and equinox of date.
      *dec (double)
         Apparent 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 'type' in structure 'ss_body'.
          > 10 ... Error code from function 'place'.
```

Discussion:
This function computes the apparent place of a planet or other solar system body. Your chosen version of function *solarsystem*, accessed from function *ephemeris*, determines the source of the body's barycentric rectangular coordinates used in the calculation.

app_planet works by calling *place* with `location->where = 0` and `coord_sys = 1`.

(Return to Function List)

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)
         Apparent right ascension in hours, referred to true equator
         and equinox of date.
      *dec (double)
         Apparent 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'.
```

Discussion:
This function computes the topocentric place of a planet or other solar system body (neglecting atmospheric refraction) for the location specified by the argument `location` at the time specified by the argument `jd_tt`. Note that `jd_tt` is the TT time at which the topocentric place is to be computed. The difference TT–UT1 (often called ΔT) is passed to the function via argument `delta_t`. Values of ΔT are published annually in *The Astronomical Almanac* or can be obtained from the EO web site.[39] The user's choice of ephemerides determines the values to be used in structure `ss_body`, which identifies the solar system object.

Atmospheric refraction can be subsequently applied to `ra` and `dec` by function *equ2hor*.

topo_planet works by calling *place* with `location->where = 1` and `coord_sys = 1`.

(Return to Function List)

[39] http://www.usno.navy.mil/USNO/earth-orientation/eo-products/long-term

virtual_planet

```
short int virtual_planet (double jd_tt, object *ss_body,
                          short int accuracy,

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

   PURPOSE:
      Compute the virtual place of a planet or other 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 virtual place.
      *ss_body (struct object)
         Pointer to structure containing the body designation for the
         solar system body (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)
         Virtual right ascension in hours, referred to the GCRS.
      *dec (double)
         Virtual declination in degrees, referred to the GCRS.
      *dis (double)
         True distance from Earth to planet in AU.

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

Discussion:
See the discussion for function *app_planet*. Function *virtual_planet* is identical to *app_planet* in input arguments and use. Here, however, the output arguments provide the virtual place (also called the proper place) of the planet. The virtual place (proper place) is essentially the apparent place expressed in the GCRS.

virtual_planet works by calling *place* with `location->where = 0` and `coord_sys = 0`.

(Return to Function List)

local_planet

```
short int local_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 local 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), Ch. 3.

   INPUT
   ARGUMENTS:
      jd_tt (double)
         TT Julian date for local 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)
         Local right ascension in hours, referred to the 'local GCRS'.
      *dec (double)
         Local declination in degrees, referred to the 'local GCRS'.
      *dis (double)
         True distance from Earth to planet in AU.

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

Discussion:
See the discussion for function *topo_planet*. Function *local_planet* is identical to *topo_planet* in input arguments and use. Here, however, the output arguments provide the

local place of the star. The local place is essentially the topocentric place expressed in the "local GCRS."

local_planet works by calling *place* with `location->where = 1` and `coord_sys = 0`.

(Return to Function List)

astro_planet

```
short int astro_planet (double jd_tt, object *ss_body,
                        short int accuracy,

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

   PURPOSE:
      Compute the astrometric place of a planet or other solar system
      body.

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

   INPUT
   ARGUMENTS:
      jd_tt (double)
         TT Julian date for astrometric place.
      *ss_body (struct object)
         Pointer to structure containing the body designation for the
         solar system body (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)
         Astrometric right ascension in hours (referred to the ICRS,
         without light deflection or aberration).
      *dec (double)
         Astrometric declination in degrees (referred to the ICRS,
         without light deflection or aberration).
      *dis (double)
         True distance from Earth to planet in AU.

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

Discussion:
See the discussion for function *app_planet*. Function *astro_planet* is identical to *app_planet* in input arguments and use. Here, however, the output arguments provide the astrometric place of the planet in the ICRS.

astro_planet works by calling place with `location->where = 0` and `coord_sys = 3`.

(Return to Function List)

precession

```
short int precession (double jd_tdb1, double *pos1, double jd_tdb2,

                      double *pos2)

   PURPOSE:
      Precesses equatorial rectangular coordinates from one epoch to
      another.  One of the two epochs must be J2000.0.  The coordinates
      are referred to the mean dynamical equator and equinox of the two
      respective epochs.

   REFERENCES:
      Explanatory Supplement To The Astronomical Almanac, pp. 103-104.
      Capitaine, N. et al. (2003), Astronomy And Astrophysics 412,
         pp. 567-586.
      Hilton, J. L. et al. (2006), IAU WG report, Celest. Mech., 94,
         pp. 351-367.

   INPUT
   ARGUMENTS:
      jd_tdb1 (double)
         TDB Julian date of first epoch.
      pos1[3] (double)
         Position vector, geocentric equatorial rectangular coordinates,
         referred to mean dynamical equator and equinox of first epoch.
      jd_tdb2 (double)
         TDB Julian date of second epoch.

   OUTPUT
   ARGUMENTS:
      pos2[3] (double)
         Position vector, geocentric equatorial rectangular coordinates,
         referred to mean dynamical equator and equinox of second epoch.

   RETURNED
   VALUE:
      (short int)
         = 0 ... everything OK.
         = 1 ... Precession not to or from J2000.0; 'jd_tdb1' or 'jd_tdb2'
                 not 2451545.0.
```

Discussion:
This function precesses the input position vector, `pos1`, from the equator and equinox of `jd_tdb1` to the equator and equinox of `jd_tdb2`; the resulting vector is `pos2`.

One of the two input Julian dates *must* be standard epoch J2000.0—either `jd_tdb1` or `jd_tdb2` must be `2451545.0` exactly. To precess a vector from one arbitrary date to another, call *precession* twice, using J2000.0 as the “middle” date. That is, in the first call, `jd_tdb1 =` first Julian date, and `jd_tdb2 = 2451545.0`; in the second call, `jd_tdb1 = 2451545.0`, and `jd_tdb2 =` second Julian date.

Formally, the current precession algorithm is a function of Barycentric Dynamical Time (TDB), but using TT as the basis for the input Julian dates results in a maximum error of only about 3×10^{-9} arcseconds, which is totally negligible. Standard epoch J2000.0, although formally defined in the TT time scale, is the same in the TT and TDB time scales to the precision given by double-precision Julian dates: at J2000.0,
TT − TDB ≈ 10^{-4} second ≈ 10^{-9} day.

(Return to Function List)

equ2ecl

```
short int equ2ecl (double jd_tt, short int coord_sys,
                   short int accuracy, double ra, double dec,

                   double *elon, double *elat)

   PURPOSE:
      To convert right ascension and declination to ecliptic longitude
      and latitude.

   REFERENCES:
      None.

   INPUT
   ARGUMENTS:
      jd_tt (double)
         TT Julian date of equator, equinox, and ecliptic used for
         coordinates.
      coord_sys (short int)
         Coordinate system selection.
            = 0 ... mean equator and equinox of date 'jd_tt'
            = 1 ... true equator and equinox of date 'jd_tt'
            = 2 ... ICRS
            (ecliptic is always the mean plane)
      accuracy (short int)
         Selection for accuracy
            = 0 ... full accuracy
            = 1 ... reduced accuracy
      ra (double)
         Right ascension in hours, referred to specified equator and
         equinox of date.
      dec (double)
         Declination in degrees, referred to specified equator and
         equinox of date.

   OUTPUT
   ARGUMENTS:
      *elon (double)
         Ecliptic longitude in degrees, referred to specified ecliptic
         and equinox of date.
      *elat (double)
         Ecliptic latitude in degrees, referred to specified ecliptic
         and equinox of date.

   RETURNED
   VALUE:
      (short int)
         = 0 ... everything OK
         = 1 ... invalid value of 'coord_sys'
```

Discussion:
This function converts the equatorial position of an object into the equivalent ecliptic position: equatorial coordinates `ra` and `dec` are converted to ecliptic coordinates `elon` and `elat`. This function can be used for any kind of barycentric or geocentric coordinates—the conversion involves a simple rotation and should be regarded as just a formalism. As in function *precession*, the input Julian date can be based on either the TDB or TT time scales, with negligible resulting error.

`ra` and `dec` can be expressed with respect to either the mean equator and equinox of date `jd_tt` (if `coord_sys = 0`) or the true equator and equinox of date `jd_tt` (if `coord_sys = 1`).

The representation of the ecliptic used for celestial coordinates is a smoothly moving mean plane described as part of the precession development. However, the mean and true equators intersect this ecliptic at different points. Therefore, the equinox, which serves as the origin of ecliptic longitude as well as the origin of right ascension, is different in the two cases. `elon` will be expressed with respect to the same equinox as `ra`.

If `jd_tt = 0.0` and `coord_sys = 0`, the function assumes `ra` and `dec` are expressed with respect to the ICRS and provides `elon` and `elat` with respect to the ecliptic and mean equinox of J2000.0.

See functions *equ2ecl_vec* and *ecl2equ_vec* for the conversion of vectors between equatorial and ecliptic systems.

(Return to Function List)

cio_ra

```
short int cio_ra (double jd_tt, short int accuracy,

                  double *ra_cio)

   PURPOSE:
      This function computes the true right ascension of the celestial
      intermediate origin (CIO) at a given TT Julian date.  This is
      -(equation of the origins).

   REFERENCES:
      Kaplan, G. (2005), US Naval Observatory Circular 179.

   INPUT
   ARGUMENTS:
      jd_tt (double)
         TT Julian date.
      accuracy (short int)
         Selection for accuracy
            = 0 ... full accuracy
            = 1 ... reduced accuracy

   OUTPUT
   ARGUMENTS:
      *ra_cio (double)
         Right ascension of the CIO, with respect to the true equinox
         of date, in hours (+ or -).

   RETURNED
   VALUE:
      (short int)
         = 0  ... everything OK.
         = 1  ... invalid value of 'accuracy'.
         > 10 ... 10 + the error code from function 'cio_location'.
         > 20 ... 20 + the error code from function 'cio_basis'.
```

Discussion:
This function supplies the true right ascension of the Celestial Intermediate Origin (CIO).

`ra_cio` = – (equation of the origins)

`ra_cio` = Greenwich apparent sidereal time – Earth Rotation Angle

where all quantities are expressed in hours.

(Return to Function List)

era

```
double era (double jd_high, double jd_low)

   PURPOSE:
      This function returns the value of the Earth Rotation Angle
      (theta) for a given UT1 Julian date.  The expression used is
      taken from the note to IAU Resolution B1.8 of 2000.

   REFERENCES:
      IAU Resolution B1.8, adopted at the 2000 IAU General Assembly,
         Manchester, UK.
      Kaplan, G. (2005), US Naval Observatory Circular 179.

   INPUT
   ARGUMENTS:
      jd_high (double)
         High-order part of UT1 Julian date.
      jd_low (double)
         Low-order part of UT1 Julian date.

   OUTPUT
   ARGUMENTS:
      None.

   RETURNED
   VALUE:
      (double)
         The Earth Rotation Angle (theta) in degrees.
```

Discussion:
This function supplies the Earth Rotation Angle, θ, which is the geocentric angle, in the instantaneous equatorial plane (true equator), between the directions toward the Terrestrial Intermediate Origin (TIO) and the Celestial Intermediate Origin (CIO).

This function allows for the input UT1 time to be represented as a split Julian date. See the discussion in the description of function *sidereal_time*.

(Return to Function List)

cel_pole

```
short int cel_pole (double tjd, short int type, double dpole1,
                    double dpole2)

   PURPOSE:
      This function allows for the specification of celestial pole
      offsets for high-precision applications.  Each set of offsets is
      a correction to the modeled position of the pole for a specific
      date, derived from observations and published by the IERS.

   REFERENCES:
      Kaplan, G. (2005), US Naval Observatory Circular 179.
      Kaplan, G. (2003), USNO/AA Technical Note 2003-03.

   INPUT
   ARGUMENTS:
      tjd (double)
         TDB or TT Julian date for pole offsets.
      type (short int)
         Type of pole offset
            = 1 for corrections to angular coordinates of modeled pole
                referred to mean ecliptic of date, that is,
                delta-delta-psi and delta-delta-epsilon.
            = 2 for corrections to components of modeled pole unit
                vector referred to GCRS axes, that is, dx and dy.
      dpole1 (double)
         Value of celestial pole offset in first coordinate,
         (delta-delta-psi or dx) in milliarcseconds.
      dpole2 (double)
         Value of celestial pole offset in second coordinate,
         (delta-delta-epsilon or dy) in milliarcseconds.

   OUTPUT
   ARGUMENTS:
      None.

   RETURNED
   VALUE:
      (short int)
         = 0 ... Everything OK.
         = 1 ... Invalid value of 'type'.
```

Discussion:
This function allows for the specification of celestial pole offsets for high-precision applications. The offsets describe the observed position of the Celestial Intermediate Pole (CIP) with respect to the position computed from the standard precession and nutation models. The offsets are subsequently applied as corrections to the nutation in longitude and nutation in obliquity within *e_tilt*. Thus, *e_tilt* output arguments `tobl`, `ee`, `dpsi`, and `deps` will be affected. Because other NOVAS functions, such as *sidereal_time*, call *e_tilt* to obtain

data related to the Earth's orientation in space, the celestial pole offsets specified here are propagated through the data that the various NOVAS functions provide.

Daily values of the celestial pole offsets are published, for example, in IERS Bulletins A and B.[40] The celestial pole offsets effectively correct for errors or incompleteness in the standard precession or nutation models. If you use *cel_pole*, make sure it is called before any other functions for a given date. Values of the pole offsets that you specify by a call to *cel_pole* will be used by *e_tilt* until you explicitly change them.

Important: For compatibility with the NOVAS version 3.0 precession and nutation models, specify `type = 2` and use *only* IERS dX and dY values with respect to "IAU 2000A" (sometimes labeled "IAU 2000"). These pole offset values will generally not exceed 0.5 milliarcsecond and therefore *cel_pole* would need to be called only when very high accuracy is required.

(Return to Function List)

[40] http://www.iers.org/MainDisp.csl?pid=36-9

e_tilt

```
void e_tilt (double jd_tdb, short int accuracy,

             double *mobl, double *tobl, double *ee, double *dpsi,
             double *deps)

   PURPOSE:
      Computes quantities related to the orientation of the Earth's
      rotation axis at Julian date 'jd_tdb'.

   REFERENCES:
      None.

   INPUT
   ARGUMENTS:
      jd_tdb (double)
         TDB Julian Date.
      accuracy (short int)
         Selection for accuracy
            = 0 ... full accuracy
            = 1 ... reduced accuracy

   OUTPUT
   ARGUMENTS:
      *mobl (double)
         Mean obliquity of the ecliptic in degrees at 'jd_tdb'.
      *tobl (double)
         True obliquity of the ecliptic in degrees at 'jd_tdb'.
      *ee (double)
         Equation of the equinoxes in seconds of time at 'jd_tdb'.
      *dpsi (double)
         Nutation in longitude in arcseconds at 'jd_tdb'.
      *deps (double)
         Nutation in obliquity in arcseconds at 'jd_tdb'.

   RETURNED
   VALUE:
      None.
```

Discussion:
This function computes various quantities related to the orientation of the Earth's rotation axis (vector toward Celestial Intermediate Pole) with respect to the ecliptic plane at a specific time. The computation involves a call to function *nutation_angles* to evaluate the nutation series.

The output values of the last four arguments will correctly reflect the celestial pole offsets if function *cel_pole* has previously been called.

(Return to Function List)

ephemeris

```
short int ephemeris (double jd[2], object *cel_obj, short int origin,
                     short int accuracy,

                     double *pos, double *vel)

   PURPOSE:
      Retrieves the position and velocity of a solar system body from
      a fundamental ephemeris.

   REFERENCES:
      None.

   INPUT
   ARGUMENTS:
      jd[2] (double)
         TDB Julian date split into two parts, where the sum
         jd[0] + jd[1] is the TDB Julian date.
      *cel_obj (struct object)
         Pointer to structure containing the designation of the body
         of interest (defined in novas.h).
      origin (int)
         Origin code; solar system barycenter   = 0,
                      center of mass of the Sun = 1.
      accuracy (short int)
         Selection for accuracy
            = 0 ... full accuracy
            = 1 ... reduced accuracy

   OUTPUT
   ARGUMENTS:
      pos[3] (double)
         Position vector of the body at 'jd_tdb'; equatorial rectangular
         coordinates in AU referred to the ICRS.
      vel[3] (double)
         Velocity vector of the body at 'jd_tdb'; equatorial rectangular
         system referred to the mean equator and equinox of the ICRS,
         in AU/Day.

   RETURNED
   VALUE:
      (short int)
         0    ... Everything OK.
         1    ... Invalid value of 'origin'.
         2    ... Invalid value of 'type' in 'cel_obj'.
         3    ... Unable to allocate memory.
         10+n ... where n is the error code from 'solarsystem'.
         20+n ... where n is the error code from 'readeph'.
```

Discussion:
This function serves as the single interface between NOVAS and ephemerides of solar system bodies. The version of *ephemeris* distributed with NOVAS provides direct support for the JPL ephemerides of major solar system bodies (such as JPL's DE405 and DE406), and the USNO minor planet ephemerides (USNO/AE98[41]). The function *ephemeris* calls an appropriate version of function *solarsystem* (or *solarsystem_hp*) in order to access an ephemeris of the "major" bodies (in this context Sun, Moon, and Mercury through Pluto). It accesses the minor planet ephemerides by calling *readeph*. Neither the USNO minor planet ephemerides nor the JPL ephemerides are part of NOVAS and must be obtained elsewhere as discussed in sections 2.4 and 2.3, respectively.

Modifying function *ephemeris* to support ephemerides other than the two mentioned above is relatively easy. In function *ephemeris*, a `switch` structure is controlled by the value of `type` in a data structure of type `object`. Currently, two `case`s within the `switch` are defined: `type = 0` for major bodies to be handled via a call to *solarsystem* and `type = 1` for minor planets to be handled via a call to *readeph*. To support ephemerides of other bodies, simply define a new value of `type` and add another `case` block containing code that accesses the new ephemeris. Alternate versions of *solarsystem* can be written to support alternate ephemerides, i.e., non-JPL, of major solar system bodies.

Additional information concerning function *solarsystem* is provided in the following sections.

(Return to Function List)

[41] http://www.usno.navy.mil/USNO/astronomical-applications/software-products/usnoae98

solarsystem

```
short int solarsystem (double tjd, short int body, short int origin,

                        double *position, double *velocity)


   PURPOSE:
      Provides the position and velocity vectors of a planet or other
      solar system body at a specific time. The origin of coordinates
      may be either the barycenter of the solar system or the center
      of mass of the Sun.

   REFERENCES:
      JPL. 2007, "JPL Planetary and Lunar Ephemerides: Export Information,"
        (Pasadena, CA: JPL) http://ssd.jpl.nasa.gov/?planet_eph_export.
      Kaplan, G. H. "NOVAS: Naval Observatory Vector Astrometry
         Subroutines"; USNO internal document dated 20 Oct 1988;
         revised 15 Mar 1990.

   INPUT
   ARGUMENTS:
      tjd (double)
         TDB Julian date.
      body (short int)
         Body identification number for the solar system object of
         interest;  Mercury = 1,...,Pluto = 9, Sun = 10, Moon = 11.
      origin (short int)
         Origin code; solar system barycenter   = 0,
                      center of mass of the Sun = 1.

   OUTPUT
   ARGUMENTS:
      position[3] (double)
         Position vector of 'body' at tjd; equatorial rectangular
         coordinates in AU referred to the mean equator and equinox
         of J2000.0.
      velocity[3] (double)
         Velocity vector of 'body' at tjd; equatorial rectangular
         system referred to the mean equator and equinox of J2000.0,
         in AU/Day.

   RETURNED
   VALUE:
      (short int)
         0...Everything OK.
         Other values depend upon version in use.
```

Discussion:
Another NOVAS function, *ephemeris,* calls this function, *solarsystem* to provide the position vector `pos` and velocity vector `vel` for `body` at time `tjd`. The vectors computed by

solarsystem are expressed with respect to ICRS axes, in the BCRS metric. The vectors are barycentric if `origin = 0` and heliocentric if `origin = 1`.

Three different versions of *solarsystem* are supplied in NOVAS, each with its own internal logic. One uses internally-stored data or series expansions while the other two use the JPL ephemerides, which exist as external data files. Additional documentation (see below) is usually required for the proper use of each version. You are, of course, free to supply your own version(s) of *solarsystem*, providing that the arguments conform to the above specifications.

The values of the body identification number, `body`, will in general differ from one *solarsystem* version to another; consult the documentation for the specific version in use. Usually, `body = 1` refers to Mercury, `body = 2` refers to Venus, `body = 3` refers to the Earth, etc., but the identification numbers for bodies such as the Sun or Moon differ across implementations. [*Note*: The data structure of type `object`, input to function *ephemeris*, specifies the way that NOVAS identifies solar-system bodies. Code within *ephemeris* does the "translation" between the `body` numbers required by *ephemeris* and the `body` numbers required by *solarsystem*.] Furthermore, some versions of *solarsystem* support only a subset of the major solar system bodies. The minimum requirement is support for the Sun and Earth. Here, "Earth" refers to the geocenter and not the Earth/Moon barycenter.

For highest-precision applications using "split" Julian dates, call *solarsystem_hp*.

(Return to Function List)

solarsystem, version 1
(File solsys1.c)

```
RETURNED
VALUE:
   None.
```

Discussion:
This version of *solarsystem* provides an "all C" interface between NOVAS and the JPL lunar and planetary ephemerides. Specifically, it serves as the interface between USNO's C version of the JPL ephemeris software (contained in file **eph_manager.c**) and the main set fof NOVAS functions. This version of *solarsystem* calls *Planet_Ephemeris* (the C version of JPL's Fortran subroutine *PLEPH*), which in turn calls other functions in the ephemeris software package. The user must set up the binary, random-access ephemeris file (see Appendix C).

Important: When using this version of *solarsystem*, the user's program that calls the NOVAS functions must make a call to function *Ephem_Open* prior to calling the NOVAS functions. *Ephem_Open* opens the binary JPL ephemeris file. Similarly, the user's program should call *Ephem_Close* to close the binary ephemeris file once ephemeris access is no longer required. *Ephem_Open* and *Ephem_Close* are located in **eph_manager.c**.

The body identification numbers to be used with this version are listed below.

Name	**body**
Sun	10
Mercury	1
Venus	2
Earth	3
Mars	4
Jupiter	5
Saturn	6
Uranus	7
Neptune	8
Pluto	9
Moon	11

(Return to Function List)

solarsystem, version 2
(File solsys2.c)

```
RETURNED
VALUE:
   (short int)
      0...Everything OK.
      1...Invalid value of body or origin.
      2...Error detected by JPL software.
```

Discussion:
This version of *solarsystem* serves as the interface between the JPL's own Fortran-based lunar and planetary ephemeris software and NOVAS. The function contains a single call to Fortran subroutine *jplint_*, which in turn calls *PLEPH* and other Fortran subroutines in the JPL ephemeris software package. The user is responsible for obtaining the Fortran ephemeris code and data, setting up the binary, random-access ephemeris file, and linking the JPL Fortran code with NOVAS. See the implementation notes below.

The body identification numbers to be used with this version are listed below.

Name	body
Sun	10
Mercury	1
Venus	2
Earth	3
Mars	4
Jupiter	5
Saturn	6
Uranus	7
Neptune	8
Pluto	9
Moon	11

Implementation Notes:
In order to use NOVAS with *solarsystem* version 2, you must first obtain the planetary ephemeris export package from JPL as discussed in Appendix C. If the verification process is successful, the ephemeris file is ready to use. The ephemeris data is obtained from the binary file by calling the access subroutines provided in the JPL export package.

Version 2 of *solarsystem* obtains ephemeris data from the binary file by calling Fortran subroutine *jplint_*, which is contained in NOVAS file **jplint.f**. Subroutine *jplint_*, in turn, calls JPL Fortran subroutine *PLEPH* and all other supporting Fortran subroutines. The C function *solarsystem* has a few features that make it possible for it to exchange data with the Fortran subroutine *jplint_*. First, all of the C arguments of the call to *jplint_* are addresses, because Fortran uses call by address instead of call by value for arguments of subroutines.

Second, all of the integer arguments in the call are designated as type `long int` in the C function to match the Fortran `INTEGER` default. The `DOUBLE PRECISION` arguments in the subroutine are designated as type `double` in the C function.

Probably the biggest hurdle in implementing version 2 of *solarsystem* involves the proper compiling and linking of files containing different languages. The procedures will be specific to your computing platform; therefore, you will have to consult your compiler manual for detailed instructions. The following instructions are offered only as a guideline; they provide a specific example of how such files were successfully handled on a PC running Red Hat Enterprise Linux ES release 4 (Nahant Update 8).

1. Create a single file named **jplsubs.f** with all of the JPL Fortran ephemeris access subroutines as discussed in Appendix C.

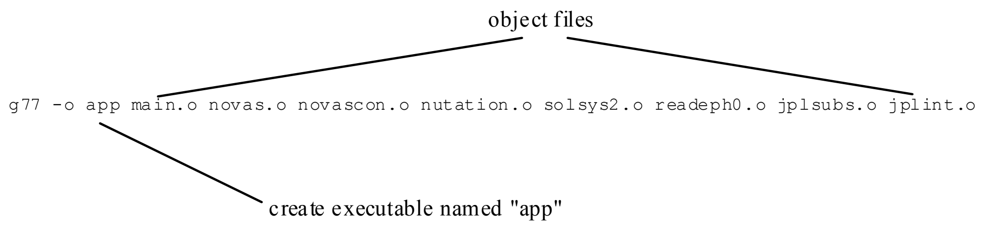
2. Compile the Fortran files, **jplsubs.f** and **jplint.f**, without invoking the linkage editor to create the object file **jplsubs.o** and **jplint.o**. The Fortran compiler/linker is `g77`.

```
g77 -c jplsubs.f jplint.f
```

3. Compile, again without invoking the linkage editor, the C files **main.c**, **novas.c**, **novascon.c**, **nutation.c**, **solsys2.c**, and **readeph0.c**, which creates the object files **main.o**, **novas.o**, **novascon.o**, **nutation.o**, **solsys2.o**, and **readeph0.o**. In this example, **main.c** is the name of the file containing the user's program. The C compiler/linker is `gcc`.

```
gcc -c main.c novas.c novascon.c nutation.c solsys2.c readeph0.c
```

4. Finally, link the object files using the Fortran linker:



```
g77 -o app main.o novas.o novascon.o nutation.o solsys2.o readeph0.o jplsubs.o jplint.o
```

In this example, the resulting executable file is named **app**.

(Return to Function List)

solarsystem, 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'.
```

Discussion:
This version of *solarsystem* provides the position and velocity of the Earth or Sun *only* without reference to any external data file. The heliocentric position and velocity of the Earth are computed by evaluating trigonometric series. When barycentric positions and velocities are required, a number of additional approximations are involved; therefore, barycentric positions and velocities computed by this version of *solarsystem* are less accurate than heliocentric positions and velocities. This version of *solarsystem* produces data within the following error limits (compared to the JPL DE405 ephemeris) for dates within the interval 1800–2050:

Earth heliocentric positions:	840	km
Earth heliocentric velocities:	1.4	m/s
Earth barycentric positions:	2500	km
Earth barycentric velocities:	1.4	m/s

When this version of *solarsystem* is used in the computation of the apparent place of the Sun, it should contribute less than 2 arcseconds error. When this version of *solarsystem* is used in the computation of apparent places of stars, it should contribute less than 1.5 milliarcseconds error. This error assessment applies to the interval 1800–2050.

This version of *solarsystem* will return `ierr = 1` for Julian dates prior to 2340000.5 (August 1694) or Julian dates after 2560000.5 (December 2296).

This version of *solarsystem* calls NOVAS function *precession*, and certain expressions in the *solarsystem* algorithm have been adjusted to conform to the IAU 2006 precession.

The body identification numbers to be used with this version are listed below.

Name	**body**
Sun	0
Sun	1
Sun	10
Earth	2
Earth	3

(Return to Function List)

solarsystem_hp

```
short int solarsystem_hp (double tjd[2], short int body,
                          short int origin,

                          double *position, double *velocity)

   PURPOSE:
      Provides an interface between the JPL direct-access solar system
      ephemerides and NOVAS-C for highest precision applications.

   REFERENCES:
      JPL. 2007, "JPL Planetary and Lunar Ephemerides: Export
         Information," (Pasadena, CA: JPL)
         http://ssd.jpl.nasa.gov/?planet_eph_export.
      Kaplan, G. H. "NOVAS: Naval Observatory Vector Astrometry
         Subroutines"; USNO internal document dated 20 Oct 1988;
         revised 15 Mar 1990.

   INPUT
   ARGUMENTS:
      tjd[2] (double)
         Two-element array containing the Julian date, which may be
         split any way (although the first element is usually the
         "integer" part, and the second element is the "fractional"
         part). Julian date is on the TDB or "T_eph" time scale.
      body (short)
         Body identification number for the solar system object of
         interest; Mercury = 1, ..., Pluto= 9, Sun= 10, Moon = 11.
      origin (short)
         Origin code; solar system barycenter = 0,
                      center of mass of the Sun = 1,
                      center of Earth = 2.

   OUTPUT
   ARGUMENTS:
      position (vectors)
         Position vector of 'body' at tjd; equatorial rectangular
         coordinates in AU referred to the ICRS.
      velocity (vectors)
         Velocity vector of 'body' at tjd; equatorial rectangular
         system referred to the ICRS.

   RETURNED
   VALUE:
      None.
```

Discussion:
Function *solarsystem_hp* provides positions and velocities for the major bodies of the solar system when the highest precision is required. This function supports the "split" Julian date feature for highest precision. The two parts of the input Julian date are stored in the two-element array, `tjd`.

(Return to Function List)

solarsystem_hp, version 1
(File solsys1.c)

```
RETURNED
VALUE:
   None.
```

Discussion:
This version of *solarsystem_hp* provides an "all C" interface between NOVAS and the JPL lunar and planetary ephemerides when the highest precision is required. Specifically, it serves as the interface between USNO's C version of the JPL ephemeris software (contained in file **eph_manager.c**) and the main set of NOVAS functions. This version of *solarsystem_hp* calls *Planet_Ephemeris* (the C version of JPL's Fortran subroutine *PLEPH*), which in turn calls other functions in the ephemeris software package. The user must set up the binary, random-access ephemeris file (see Appendix C).

Important: When using this version of *solarsystem_hp*, the user's program that calls the NOVAS functions must make a call to function *Ephem_Open* prior to calling the NOVAS functions. *Ephem_Open* opens the binary JPL ephemeris file. Similarly, the user's program should call *Ephem_Close* to close the binary ephemeris file once ephemeris access is no longer required. *Ephem_Open* and *Ephem_Close* are located in **eph_manager.c**.

The body identification numbers to be used with this version are listed below.

Name	**Body**
Sun	10
Mercury	1
Venus	2
Earth	3
Mars	4
Jupiter	5
Saturn	6
Uranus	7
Neptune	8
Pluto	9
Moon	11

(Return to Function List)

solarsystem_hp, version 2
(File solsys2.c)

```
RETURNED
VALUE:
   (short int)
      0...Everything OK.
      1...Invalid value of body or origin.
      2...Error detected by JPL software.
```

Discussion:
This version of *solarsystem_hp* serves as the interface between the JPL's own Fortran-based lunar and planetary ephemeris software and NOVAS when the highest precision is required. The function contains a single call to Fortran subroutine *jplihp_*, which in turn calls *DPLEPH* and other Fortran subroutines in the JPL ephemeris software package. The user is responsible for obtaining the Fortran ephemeris code and data, setting up the binary, random-access ephemeris file, and linking the JPL Fortran code with NOVAS. See the implementation notes below.

The body identification numbers to be used with this version are listed below.

Name	**Body**
Sun	10
Mercury	1
Venus	2
Earth	3
Mars	4
Jupiter	5
Saturn	6
Uranus	7
Neptune	8
Pluto	9
Moon	11

Implementation Notes:
In order to use NOVAS with *solarsystem_hp* version 2, you must first obtain the planetary ephemeris export package from JPL (see Appendix C for details). If the verification process is successful, the ephemeris file is ready to use. The ephemeris data is obtained from the binary file by calling the access subroutines provided in the JPL export package.

Version 2 of *solarsystem_hp* obtains ephemeris data from the binary file by calling Fortran subroutine *jplihp_*, which is in the NOVAS **jplint.f** file. Subroutine *jplihp_*, in turn, calls JPL subroutine *DPLEPH* (Fortran code) and all other supporting Fortran subroutines. The C function *solarsystem_hp* has a few features that make it possible for it to exchange data with the Fortran subroutine *jplihp_*. First, all of the C arguments of the call to *jplihp_* are addresses, because Fortran uses call by address instead of call by value for arguments of subroutines. Second, all of the integer arguments in the call are designated as type `long int` in the C function to match the Fortran `INTEGER` default. The `DOUBLE PRECISION` arguments in the subroutine are designated as type `double` in the C function.

Probably the biggest hurdle in implementing version 2 of *solarsystem_hp* involves the proper compiling and linking of files containing different languages. The procedures will be specific to your computing platform; therefore, you will have to consult your compiler manual for detailed instructions. The instructions given for version 2 of *solarsystem* provide an example of how such files could be handled.

(Return to Function List)

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`, 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.

(Return to Function List)

Nutation Models (File nutation.c)

The C version of NOVAS provides three implementations of the nutation model, all of which are found in file **nutation.c**. Each model computes the nutation angles, $\Delta\psi$ and $\Delta\varepsilon$, which are the nutation in longitude and obliquity, respectively. These functions have the following input and output:

```
INPUT
   ARGUMENTS:
      jd_high (double)
         High-order part of TT Julian date.
      jd_low (double)
         Low-order part of TT Julian date.

   OUTPUT
   ARGUMENTS:
      *dpsi (double)
         Nutation (luni-solar + planetary) in longitude, in radians.
      *deps (double)
         Nutation (luni-solar + planetary) in obliquity, in radians.

   RETURNED
   VALUE:
      None.
```

Discussion:
The nutation model is invoked via function *nutation_angles* in file **novas.c**. As previously mentioned, the nutation model can be the most computationally intensive part of NOVAS. It makes no sense to evaluate the full IAU 2000A nutation model (1,365 terms) [*iau2000a*] unless the highest level of accuracy is needed. Thus, in addition to the full IAU 2000A model, two reduced-accuracy models—truncated versions of IAU 2000A—are also provided: IAU 2000B [*iau2000b*] and 2000K [*nu2000k*]. Section 1.4 discussed these models in greater detail. Several higher-level NOVAS functions have an input argument, `accuracy`, which specifies whether a full- or reduced-accuracy calculation is desired. Whenever `accuracy` is set to 0 (full accuracy), IAU 2000A is used. By default, whenever `accuracy` is set to 1 (reduced accuracy), 2000K is used. However, a small coding change can be made to *nutation_angles* to replace 2000K with IAU 2000B as the reduced-accuracy model. The prolog and comments in *nutation_angles* explain how to make this change. A summary of the models follows.

Nutation Model	**Function Name**	**Terms**	**Accuracy Compared to IAU 2000A (mas)**	**Time Span**
IAU 2000A	*iau2000a*	1,365	…	…
2000K	*nu2000k*	488	0.1	1700–2300
IAU 2000B	*iau2000b*	77	1	1995–2050

(Return to Function List)

Chapter 5 Equinox- and CIO-Based Paradigms Compared

5.1 Computing Hour Angles

The equinox- and CIO-based celestial reference systems are part of two computational schemes for accounting for the Earth's instantaneous orientation with respect to the stars. These two methods represent the same phenomena (as they obviously must) but in slightly different order. The overall matrix that embodies, for a given instant, the terrestrial-to-celestial (ITRS-ICRS) transformation is the same for both schemes. Therefore, the value of observable quantities will not be affected by the choice of which paradigm is used for the computations.

In NOVAS, quantities such as declination and hour angle, which are in principle measurable angles, should have the same values regardless of the way in which they are computed. Because both the equinox-based and CIO-based paradigms are constructed on the instantaneous (true) equator of date—the plane orthogonal to the CIP—declinations are, in fact, completely unaffected.

How are hour angles of celestial objects computed in the old and new paradigms? Assume that we are considering Greenwich hour angles, that is, hour angles measured from the meridian of geodetic longitude zero (the X-Z plane of the ITRS), without polar motion. In the equinox-based scheme, we compute the topocentric place of the object of interest with respect to the true equator and equinox of date. Then, we compute Greenwich apparent sidereal time and subtract the object's apparent right ascension to form the hour angle. In the CIO-based scheme, we compute the object's topocentric place with respect to the true equator and CIO of date. To form the hour angle, we compute the ERA and subtract the CIO-based right ascension (also called the intermediate right ascension). The two results should be identical. The computed `GHA` may have to be reduced to the range -12^h to $+12^h$. The table below summarizes the two equivalent procedures for hour angle and the NOVAS functions that would be used for each, assuming that polar motion is neglected. The procedures outlined here provide the Greenwich hour angle (`GHA`) of a star; only the first step would be different for a solar system body.

Functions *app_star*, *topo_star*, and *place* require time arguments in the TT time scale (*topo_star* also requires ΔT), while *sidereal_time* and *era* require time arguments in the UT1 time scale. The two procedures should yield the same value of `GHA` to within a micro-arcsecond around the present time and identical values for `DEC`. To obtain the local hour angle in either method, simply add to the `GHA` the observer's longitude (east positive) in appropriate units.

	Equinox-Based Method	**CIO-Based Method**
Use function	*topo_star* — or — *place* with `cel_object (type) = '2'`, `location = 1`, and `coord_sys = 1`	*place* with `cel_object (type) = '2',`, `location = 1`, and `coord_sys = 2`
…to obtain	`RA` and `DEC` topocentric RA and dec of the star with respect to the true equator and equinox of date (in hours and degrees, respectively)	`RA` and `DEC` topocentric RA and dec of the star with respect to the true equator and CIO of date (in hours and degrees, respectively)
Then, use function	*sidereal_time* with `gst_type = 1`	*era*
…to obtain	`GST` Greenwich apparent sidereal time (in hours)	`THETA` Earth Rotation Angle, θ (in degrees)
Compute Greenwich hour angle	`GHA = GST - RA` (in hours)	`GHA = THETA / 15.0 - RA` (in hours)

The common notion of hour angle becomes somewhat problematic when polar motion is taken into account, because what we usually regard as the Greenwich (or observer's) meridian—a plane of constant geodetic longitude—is not, in general, parallel to an hour circle on the celestial sphere when the geodetic pole and the CIP are not coincident. See the discussion in section 6.5.4 of USNO Circular 179.[42]

5.2 Other Computational Considerations

Two high-level NOVAS functions that involve Earth rotation, *sidereal_time* and *ter2cel*, can perform their internal calculations using either the equinox-based paradigm or the CIO-based paradigm. As previously mentioned, *sidereal_time* computes sidereal time. *ter2cel* performs the terrestrial-to-celestial transformation. *equ2hor* is indirectly involved because it calls *ter2cel*. The method used within either function is selected when that function is called. The integer input argument, `method`, must be set to either zero (0) to use the CIO-based method or to one (1) to use the equinox-based method. Because there is no external difference in how *sidereal_time* or *ter2cel* are used, and the two computational paradigms yield answers that are consistent within a few microarcseconds over many centuries, there is seldom a practical basis for a choice. However, the equinox method is much more efficient if mean sidereal time is to be computed.

[42] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

Finally, another of the new Earth-rotation-related functions is worth mentioning. For a given TT Julian date, *cio_ra* provides the right ascension of the CIO with respect to the true equinox of date. With a sign reversal, this quantity is the equation of the origins, the direction of the true equinox measured in the equator eastward (+) from the CIO. Because the equinox and CIO are simply different right ascension origins on the instantaneous equator, *cio_ra* provides the angular difference between the origins of these two systems. The equation of the origins is also the difference, expressed as an angle, between the ERA and Greenwich apparent sidereal time.

5.3 How NOVAS Implements the CIO-Based Paradigm

The equinox-based paradigm is, of course, the historical basis for NOVAS. One of its key pieces is the precession algorithm [*precession*], which uses the equinox as its azimuthal coordinate; that is, it transforms celestial coordinates from the mean equator and equinox of one date to the mean equator and equinox of another date. Even though the recommended precession formulation has been replaced twice over the last half-century, this aspect of it has remained unchanged. Another key piece is the algorithm for sidereal time [*sidereal_time*], which is based on a sidereal day that is defined by successive transits of the equinox. The sidereal time formula must always be matched to the precession algorithm, because mean sidereal time must account for the precession of the equinox in right ascension; this has been consistently done in NOVAS.

To use the CIO-based paradigm, we must know where the CIO is in some well-defined coordinate system. Unlike the equinox, the CIO is not defined by static geometry but by its motion, so its position at any time is given by the result of an integral that has been evaluated either analytically or numerically. In NOVAS, both results are available: the position of the CIO can be taken from an external file that is the output of a numerical integration, or it can be obtained from an analytical expression for the equation of the origins [*cio_location*]. See sections 6.5.1.1 and 6.5.1.2 of USNO Circular 179.[43]

The NOVAS implementation of the CIO-based Earth rotation paradigm for a given date is based on the construction of the Celestial Intermediate Reference System for that date, using vectors toward the CIP and the CIO. These two directions define, respectively, the z-axis and x-axis of the celestial intermediate system. The direction toward the CIP in the GCRS can be computed by passing the vector (0,0,1) through functions *nutation*, *precession*, and *frame_tie* in succession. Given the direction of the CIP, the other piece of required information is the location of the CIO for the same date, which is provided by *cio_location*, described below. The basis vectors of the intermediate system, with respect to the GCRS, are computed by *cio_basis* (see section 6.5.1 of USNO Circular 179 for the algorithms). Having these basis vectors available allows NOVAS to easily transform any vector in the GCRS to the intermediate system. The only other quantity used in the CIO-based paradigm is the ERA, which is trivial to compute and provided by *era*.

Function *cio_location* obtains the location of the CIO for a given date in one of two ways, and sets an output argument, `ref_sys`, that indicates which way was used. If an external

[43] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

file of CIO right ascension values is available (nominally called **cio_ra.bin** and located in the current directory) then *cio_location* will provide the GCRS right ascension of the CIO, and will set `ref_sys` to one (1). If this file is not available, then *cio_location* will provide the true right ascension of the CIO (the arc on the instantaneous equator from the equinox to the CIO), obtained from a series expansion, and will set `ref_sys` to two (2). *cio_basis* can work with either coordinate of the CIO. The two methods are equivalent within several microarcseconds over six centuries centered on the year 2000; it is not clear which method is more correct.

To do the hard work, *cio_location* calls either *cio_array* (for `ref_sys = 1`) or *ira_equinox* (for `ref_sys = 2`). *cio_location* always initially checks to see if the external file of CIO right ascensions is present. If it is, *cio_array* reads the file, which is the output from a numerical integration covering years 1700 to 2300, and returns an array to be interpolated; the interpolation directly provides the right ascension of the CIO in the GCRS. A copy of **CIO_RA.TXT** is provided as part of the NOVAS distribution, along with a utility program called **cio_file.c** to convert the text file to a binary direct-access file; section 2.5 describes creating **cio_ra.bin** (2.9 Mbytes) from **CIO_RA.TXT** (7.5 Mbytes).

If **cio_ra.bin** is not present, then *cio_location* calls *ira_equinox* to evaluate the equation of the origins from a closed-form expression that includes the evaluation of nutation in longitude, a lengthy series of trigonometric terms. The result locates the CIO with respect to the equinox on the instantaneous equator.

At no point does NOVAS use the CIO locator, *s*, which is described in IERS documents and *The Astronomical Almanac*.

5.4 References

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)

USNO & HMNAO. 2004, *The Astronomical Almanac for the Year 2006*, (Washington, DC: GPO) and subsequent editions

(Return to Table of Contents)

Appendix A Overview of How NOVAS Has Changed

A detailed list of the changes in the NOVAS C code from the previous version (C2.0.1 of 2000) is given in Appendix B. This appendix provides some perspective on these changes for people who are already familiar with NOVAS. Most of the modifications have been made in response to resolutions passed by the IAU in 2000 and 2006 that recommended new models for fundamental astronomy within a new conceptual framework. To the greatest extent possible, the calling sequences for the highest-level (and most used) functions from the previous versions of NOVAS have been preserved, but there are a few important exceptions. Many new calls are now available as well.

A.1 Important Changes in Calls

Probably the most important change to existing NOVAS calls is the change of proper motion and parallax units in the calls to *app_star*, *virtual_star*, *astro_star*, *transform_cat*, and *transform_hip*, all of which deal with star positions. The units have been changed as follows:

- proper motion in right ascension: from seconds per century to milliarcseconds per year
- proper motion in declination: from arcseconds per century to milliarcseconds per year
- parallax: from arcseconds to milliarcseconds

These changes have been made to conform to the units used in most modern star catalogs (e.g., Hipparcos, Tycho-2, or the FK6), which in turn follow from the observational techniques now used in the construction of such catalogs. Obviously, star data previously used with NOVAS must either be replaced or transformed. The transformation equations from “old” to “new” units are as follows:

```
pmrnew = pmrold * 150.0 * cos ( dec0 * DEG2RAD ); /* proper motion in RA */
pmdnew = pmdold * 10.0;    /* proper motion in declination */
paxnew = paxold * 1000.0;    /* parallax */
```

where `dec0` is the catalog declination (J2000.0 or ICRS) of the star in degrees and `DEG2RAD` is the degrees-to-radians conversion factor (`0.01745329...`).

Another important change to the high-level functions is that the argument lists of the traditional “place” functions (*app_star, topo_star, app_planet, topo_planet. virtual_star,* etc.) have changed. The `earth` data structure (type `body`) has been removed. This structure is now created within the new *place* function described in the following section. A new input parameter, `accuracy`, has been added to select either full-accuracy or reduced-accuracy calculations.

Finally, function *pnsw* has been renamed to *ter2cel* with a change to the time argument; this function carries out the terrestrial-to-celestial transformation.

All other changes to existing NOVAS calls involve lower-level functions not frequently invoked by most users; these are detailed in Appendix B.

A.2 *place*: A New General-Purpose “Place” Function

All computational code to compute apparent, topocentric, virtual, astrometric, etc., places of stars or planets has now been consolidated into a single new function called *place*. The familiar calls to the traditional place functions from earlier versions of NOVAS still work essentially as before—except for the important changes described in A.1 above—but are now just “front-ends” to *place* [*app_star, topo_star, app_planet, topo_planet. virtual_star, local_star, virtual_planet, local_planet, astro_star, astro_planet*]. This change eliminated much duplicate code, provides more flexibility, and allows for possible future additions (such as binary star orbits or nonlinear terms in proper motion). *place* can produce star or planet positions within the Celestial Intermediate Reference System that is part of the new paradigm for Earth rotation calculations (see below). *place* provides its output position both in spherical coordinates (right ascension, declination, and, for solar system bodies, geometric distance) and as a unit vector. In addition, *place* furnishes radial velocity. It also allows the observer to be located at the geocenter, on or near the Earth’s surface, or in a near-Earth spacecraft. You may want to consider changing your calls to *app_star*, *app_planet*, etc., to the equivalent calls to *place*; the code for the new versions of the traditional place functions specifies the appropriate input parameters.

A.3 New Reference Systems

The IAU resolutions of 2000 defined several new reference systems for fundamental astronomy. The Barycentric Celestial Reference System (BRCS), Geocentric Celestial Reference System (GCRS), International Celestial Reference System (ICRS), and Celestial Intermediate Reference System are defined briefly below. More detailed descriptions of these systems are in section 1.1 of this document and in Chapters 1, 3, and 6 of USNO Circular 179.[44]

Barycentric Celestial Reference System (BCRS) replaces the barycentric system based on the mean equator and equinox of J2000.0. It is used for data tabulated in astrometric catalogs and fundamental solar system ephemerides.

Geocentric Celestial Reference System (GCRS) replaces the geocentric system based on the mean equator and equinox of J2000.0. It is used for geocentric apparent positions of celestial objects, measurements and coordinates in the near-Earth environment, and artificial Earth satellite ephemerides.

The BCRS and GCRS are nearly parallel systems, related by a relativistic transformation.

International Celestial Reference System (ICRS) is the name applied to the orientation of the axes of the BCRS, based on the adopted coordinates of several hundred extragalactic radio sources that are assumed to have no net systematic motion. The resulting orientation is close to, but not exactly aligned with, the mean equator and equinox of J2000.0. The ICRS is a “space-fixed” (kinematically non-rotating) system.

[44] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

Because of their close relationship, the abbreviations "BCRS" and "ICRS" are often used interchangeably.

Celestial Intermediate Reference System is a system for geocentric apparent positions of stars and planets based on the true (instantaneous) equator of date and a zero point of right ascension at the CIO (see section A.5).

The geocentric system based on the true equator and equinox of date is also still used for geocentric apparent positions of stars and planets.

These reference systems are now used for the input and output arguments to NOVAS subroutines. For example, NOVAS C3.0 assumes that input reference data, such as catalog star positions and proper motions, and the basic solar system ephemerides, are provided in the ICRS (that is, within the BCRS as aligned to the ICRS axes), or at least are consistent with it to within the data's inherent accuracy. The distinction between the ICRS and the system defined by the mean equator and equinox of J2000.0 (the "J2000.0 system") becomes important only when an accuracy of 0.02 arcseconds or better is needed. Nevertheless, because NOVAS is designed for the highest accuracy applications, the ICRS is given as the reference system of choice for many input arguments to NOVAS functions.

Because the ICRS axes are not precisely aligned to those of the J2000.0 system, a new function called *frame_tie* is available to transform vectors between the two systems. This transformation is a very small fixed rotation. *frame_tie* is used for both barycentric vectors (BCRS to or from the barycentric J2000.0 system) and geocentric vectors (GCRS to or from the geocentric J2000.0 system). *frame_tie* is called many times, in both directions, within the NOVAS code. It is needed because precession (and nutation) can properly be applied only to vectors in a real equatorial system; vectors in the GCRS (geocentric ICRS) must be transformed, via *frame_tie*, to the J2000.0 system before *precession* is used. If your code only interacts with the highest-level NOVAS functions, all this is transparent to you. However, if you use *precession* within your own code, you should precede it by a call to *frame_tie* (with the middle argument direction > 0) if your input vector is expressed in the GCRS, that is, if it is derived from an input source based on the ICRS.

Output data from many of the supervisory-level NOVAS functions can be expressed in the GCRS or either of two equator-of-date systems: the true equator and equinox of date or the Celestial Intermediate Reference System. The latter two systems differ only in their right ascension origins, and, in the new paradigm, they are understood to be derived from the GCRS by applying a few rotations.

A.4 New Models for Precession and Nutation

As described in section 1.4, new models for both precession and nutation have been adopted by the IAU and have been incorporated into NOVAS. Although the underlying developments for these effects are different than in NOVAS 2.0, from a programming point of view, little has changed. The functions that directly involve precession and nutation [*precession*, *nutation*, *nutation_angles*, *sidereal_time*] essentially work the same as before, but with slightly different results. The new nutation model has more than ten times the number of trigonometric terms than the previous model. Because evaluation of nutation has always been the most computationally intensive task in NOVAS, you may notice an increase in execution time for some NOVAS applications. However, that extra computation time can be reduced.

Earth rotation calculations can be performed in either full- or reduced-accuracy mode as described in section 2.6. Many functions include an input argument, `accuracy,` that determines which mode will be used by that function and by any function it calls. In full-accuracy mode, the various models are evaluated at the few-microarcsecond level. For nutation, using this mode means that a 1,365-term trigonometric series is evaluated for each unique date. However, neither the models nor current observations are accurate at this level; so much of the increased computational burden is unproductive. Reduced accuracy mode can be used when the accuracy requirements are not better than 0.1 milliarcsecond for stars or 3.5 milliarcseconds for solar system bodies. The computation time for these calculations is thereby reduced by about two-thirds.

A.5 New Model for the Rotation of the Earth about its Axis

IAU resolutions passed in 2000 established a new geometric paradigm for how we describe the Earth's spin *around* its axis. Both the old and new paradigms are based on the instantaneous (true) equator of date, but they use different fiducial points on the equator as the origin of right ascension and different time-like quantities (actually, time-dependent angles) to describe the rotation of the Earth. As described in section 1.4 and Chapter 5, the conventional scheme is based on the equinox and sidereal time; in the new paradigm, the reference point is called the CIO and the time-like quantity is called the ERA. A more complete explanation of the new concepts, along with the algorithms used with them, can be found in Chapter 6 of USNO Circular 179.[45]

NOVAS 3.0 implements both the equinox- and CIO-based computational schemes. Implementing the CIO-based paradigm has required the addition of many new functions, along with new code added to existing functions. First, the function *place* is coded to provide output right ascensions with respect to either the equinox or the CIO; the choice is made through input argument `coord_sys`. Function *gcrs2equ* can similarly convert GCRS right ascensions and declinations to their equatorial equivalents (for a given date), with output right ascensions measured with respect to either zero point. *cio_ra* provides the angle between the two zero points, that is, the difference between the two right ascension systems. Functions *cio_location*, *cio_basis*, *cio_array, ira_equinox* provide lower-level support to these computations. Function *era* computes the ERA for any instant.

Two high-level NOVAS functions that involve Earth rotation, *sidereal_time* and *ter2cel* (the latter replaces the old *pnsw*) have been re-coded to perform their internal calculations using either the equinox-based or CIO-based paradigm. When each function is called, an input argument, `method`, is set to either zero (0) for the CIO-based approach or one (1) for the equinox-based approach.

[45] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

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 *equ2ecl_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.

A.7 New Terminology

Not surprisingly, the IAU resolutions on reference systems and Earth rotation have required some new terminology, and an IAU Working Group on Nomenclature for Fundamental Astronomy was established for the 2003–2006 triennium to systematize the usage. New terms and abbreviations now appear in the comment statements of many NOVAS functions, including the prologs where the input and output arguments are described. The most important terms are described in Chapter 1 of this document and further information can be found in USNO Circular 179.

A.8 References

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)

Appendix B List of Changes to Functions Between C2.0.1 and C3.0

This appendix lists in detail the changes between the current C3.0 version of NOVAS and the previous C2.0.1 version of 2000. Appendix A provides some perspective on these changes for people who are already familiar with NOVAS. Most of the modifications are in response to resolutions passed by the IAU in 2000 and 2006 that recommended new models for fundamental astronomy within a new conceptual framework. To the greatest extent possible, the calling sequences for the highest-level (and most used) functions from the previous versions of NOVAS have been preserved, but there are a few important exceptions. Many new calls are now available as well.

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 IERS code[46] (Wallace 2003a).

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.

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

gcrs2equ transforms GCRS RA and dec to RA and dec on mean or true equator of date. For true equator of date, either the true equinox or the CIO can be specified as the origin of right ascension.

geo_posvel is called from *place* to compute the geocentric position and velocity vectors of an observer on, or above, the surface of the Earth.

grav_def replaces *sun_field;* it supervises the evaluation of gravitational deflection of light due to the Sun, Jupiter, and other solar system bodies. It calls a new function *grav_vec* to do the deflection calculation for each body.

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[47] (Wallace 2003b).

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

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.

light_time is called from *place* to antedate the position of a solar system body for light-time.

limb_angle evaluates where an observed object is with respect to the Earth's limb (horizon), given the geocentric position vectors of the observer and the object. *place* calls *limb_angle* for the topocentric cases in order to decide whether to include the gravitational deflection of light due to the Earth itself.

make_observer creates a structure of type `observer`, specifying the location of the observer.

make_observer_at_geocenter creates a structure of type `observer`, specifying an observer at the geocenter.

make_observer_in_space creates a structure of type `observer`, specifying the position and velocity of an observer situated on a near-Earth spacecraft.

make_observer_on_surface creates a structure of type `observer`, specifying the location of and weather for an observer on the surface of the Earth.

make_in_space creates a structure of type `in_space`, specifying the position and velocity of an observer situated on a near-Earth spacecraft.

make_on_surface creates a structure of type `on_surface`, specifying the location of and weather for an observer on the surface of the Earth.

mean_obliq is called from *e_tilt* and *cel_pole* to compute the mean obliquity of the ecliptic.

norm_ang is called by *ee_ct* to normalizes angles into the range 0 <= angle < (2 π).

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

nu2000k is a modification of *iau2000a*. It evaluates a truncated version (2000K) of the full IAU 2000A nutation series and uses a consistent set of expressions for the fundamental arguments, those of Simon et al. (1994). It is more accurate than the IAU 2000B series: about 0.1 milliarcseconds for Δψ and about 0.04 milliarcseconds for Δε and Δψ sin ε over a longer interval of time than IAU 2000B.

place is a new, general-purpose function for computing apparent, topocentric, virtual, astrometric, etc., places of stars and planets. All substantive code for performing these calculations has been moved from *app_star*, *topo_star*, *app_planet*, etc., into *place*. In the call to *place*, the object requested is specified by the input parameter `cel_object`. The type of place requested is specified by two input parameters, one indicating the `location` of the observer and the other indicating the coordinate system of the output positions. *app_star*, *topo_star*, *app_planet*, etc., now are just "front-ends" to *place*.

rad_vel is called from *place* to compute the radial velocity of observed object with respect to the observer.

B.2 Changes to NOVAS C2.0.1 Structures and Calling Sequences

All of the high-level functions (*place*, *app_star*, *app_planet*, etc.) now assume that they are working with ICRS data, including input RA, dec, and proper motion components for the stellar functions and position and velocity vectors obtained from *solarsystem* for both stellar and planetary functions. *virtual_star, local_star, virtual_planet, local_planet*, *astro_star*, *astro_planet,* and *mean_star* produce output positions in the ICRS.

app_star, virtual_star, astro_star, app_planet, virtual_planet, astro_planet, mean_star, topo_star, topo_planet, local_star, and *local_planet* have a new `accuracy` input parameter. The former `earth` input parameter was removed from each of these functions.

aberration no longer returns an error code.

bary_to_geo was renamed *bary2obs*.

cel_pole can now accept input corrections to the pole positions as either (dψ, dε) or (dX, dY); the choice of correction is specified by a new input parameter `type`. In either case, the units of the correction must be in milliarcseconds. In addition, this function now returns an error code.

earthtilt was renamed *e_tilt* with a new `accuracy` input parameter.

ephemeris now returns an error code.

get_earth was eliminated and its functionality moved to *place*.

make_cat_entry changed its proper motion units (in both RA and dec) to milliarcseconds/year and changed its parallax units to milliarcseconds. Proper motion in RA includes a cos dec factor.

nutate was renamed *nutation*. It has a new `accuracy` input parameter, and it no longer returns an error code.

nutation_angles has a new `accuracy` input parameter, and it no longer returns an error code.

pnsw was renamed *ter2cel* (terrestrial-to-celestial transformation) with the former input argument `tjd` changed to a UT1 Julian date split into a pair of double-precision words, `jd_ut_high` and `jd_ut_low`, and with a new `accuracy` input parameter added.

precession can no longer process two arbitrary epochs. Therefore, one of the input epochs must be `2451545.0` (J2000.0), because the new precession expressions are not as flexible as those of Lieske et al. (1977, 1979). In addition, this function now returns an error code.

set_body was renamed *make_object*.

sidereal_time has new `delta_t`, `gst_type`, `method`, and `accuracy` input parameters. The former `ee` input parameter was removed. In addition, this function now returns an error code.

spin is no longer specifically associated with sidereal time. It now applies a rotation about the current z-axis with the angle expressed in degrees.

`struct body` was renamed `object` with the length of `name` member shortened to 50 characters and structure `cat_entry` member added.

`struct site_info` was renamed `on_surface`.

sun_field was replaced by *grav_def*, a more general function that evaluates the gravitational deflection of light due to several solar system bodies.

tdb2tdt was renamed *tdb2tt*.

transform_cat has two modified and two new transformation options. Existing `option=2` and existing `option=3` can no longer process two arbitrary epochs. Therefore, one of the input epochs for these options must be `2451545.0` (J2000.0), because the new precession expressions are not as flexible as those of Lieske et al. (1977, 1979). New `option=4` rotates data from the dynamical equator and equinox of J2000.0 to the ICRS while new `option=5` does the opposite rotation.

wobble has a new Julian date argument.

B.3 Significant Internal Changes to Code

app_star, *topo_star*, *app_planet*, *topo_planet*, *virtual_star*, *local_star*, *virtual_planet*, *local_planet*, *astro_star*, and *astro_planet* are now simply "front-ends" to specific calls to *place*. All substantive apparent place calculations of the various kinds are now done only in *place*. The following changes in the basic algorithms were made.

1. Calls to *frame_tie* were added in appropriate places to transform between the ICRS and the dynamical system.
2. In updating a star's position for proper motion, a correction to the epoch of interest for the difference in light-time between the solar system barycenter (the reference point for the input catalog data) and the Earth itself is now included. This change affects only stars with the greatest proper motions, and, then, only at the 0.1-milliarcsecond level. The new function *d_light* is used to compute the epoch offset.
3. The "Doppler factor," k, is included in the computation of stellar space motion vectors as discussed in the note on *starvectors* below.

4. Modifications were made related to the change in gravitational deflection algorithms from *sun_field* to the more general function *grav_def* as discussed in the note above.
5. Code has been introduced that allows a place to be expressed in the Celestial Intermediate Reference System (equator of date with CIO as the right ascension origin).
6. Code has been added that allows the input of an observer's instantaneous geocentric position and velocity vectors (with respect to the true equator and equinox of date) for a topocentric place calculation; this change supports satellite observations.

e_tilt now evaluates a more complete series for the complementary terms in the equation of the equinoxes; formerly, it just evaluated the two largest terms. It uses the expression for the mean obliquity from the P03 precession formulation. Internally, the function works in either full- or reduced-accuracy mode as set by the `accuracy` input parameter.

Full-accuracy mode obtains the sum of the terms from IERS function *ee_ct*.
Reduced-accuracy obtains the sum of the terms from a nine-term internal series.

equ2hor implements an improved algorithm for refraction geometry.

fund_args now evaluates expressions for the fundamental solar and lunar arguments from Simon et al. (1994).

nutation_angles now just calls either of the nutation functions, *iau2000a* (from the IERS) or *nu2000k* (a reduced-accuracy version of *iau2000a*), to do the hard work; it does not contain a nutation series itself. The nutation function called depends on the value of the `accuracy` input parameter. In reduced-accuracy mode, *iau2000b* may be called instead of *nu2000k* by making a small code modification as explained in the prolog to the *nutation_angles* function.

precession now evaluates the precession-angle polynomials for the Capitaine, Wallace, and Chapront (2003) P03 model, which was recommended by an IAU resolution in 2006. Some code changes were made to ensure reversibility of transformation (to/from J2000.0).

sidereal_time returns the value of sidereal time, either mean or apparent. Internally, this function can work by either of two methods as set by the `method` input parameter.

Equinox-based method evaluates the expression for mean sidereal time following the discussion in section 2.6.2 of USNO Circular 179.[48] The ERA, θ, is obtained from *era*. For apparent sidereal time, the equation of the equinoxes, including the "complementary terms," is obtained from *e_tilt*.
CIO-based method obtains the sidereal time using the method of section 6.5.4 of USNO Circular 179. That equation is based on the position of the true equinox of date in the Celestial Intermediate Reference System, the basis of which is obtained from *cio_basis*. The ERA, θ, is obtained from *era*. Mean sidereal time, when requested, is obtained by *subtracting* the equation of the equinoxes, obtained from *e_tilt*.
In either method, *sidereal_time*/*era* evaluates θ using the input UT1 epoch, but other components of sidereal time are evaluated using TDB (set equal to TT), with TT=UT1+ΔT where the `delta_t` input parameter sets the ΔT value.

[48] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-179

ter2cel performs the terrestrial-to-celestial transformation on a given vector, i.e., the total rotation from the ITRS to the ICRS; it includes calls to *frame_tie* at appropriate places. Internally, this function can work by either of two methods as set by `method` input parameter.

> Equinox-based method evaluates the old-style transformation as per the previous function *pnsw*, but with a call to *frame_tie* added at the end to put final vector in ICRS. This method uses apparent sidereal time, obtained from *sidereal_time*.
> CIO-based method performs the transformation of equations 3 and 4 given by Kaplan (2003), based on the Celestial Intermediate Reference System. The orthonormal basis of the system is obtained from *cio_basis* and the ERA, θ, is obtained from *era*.
> In either method, the "fast angle" (rotation about z-axis) is evaluated using the input UT1 epoch, but other components of sidereal time are evaluated using TDB (set equal to TT), with TT=UT1+ΔT where the `delta_t` input parameter sets the ΔT value.

transform_cat, *transform_hip*, and *starvectors* were modified to accommodate the new proper motion and parallax units.

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.

transform_cat calls *frame_tie* for new transformation `option=4` and for the new transformation `option=5`, which rotate data between dynamical J2000.0 system and ICRS.

wobble now includes a very tiny (inconsequential for most applications) rotation about the z-axis in the matrix to correct the ITRS longitude origin to TIO. This rotation uses the recently published approximation to TIO longitude as a function of time, which required that the new time argument also be added to this function. Essentially, this modification changes a W rotation to a W′ rotation. Also, the matrix-element expressions changed from first-order approximations to exact expressions for increased precision.

The following changes were made to constants:

- renamed `MAU` to `AU` and updated value
- renamed `KMAU` to `AU_KM` and updated value
- added `AU_SEC` from DE405
- added `C_AUDAY`
- changed units of `C` from AU/day to meters/second
- updated value of `GS` from DE405
- added `GE` from DE405
- renamed `EARTHRAD` to `ERAD`, changed units from kilometers to meters, and updated value
- updated value of `F`
- renamed `OMEGA` to `ANGVEL`
- added `RMASS[12]`
- updated value of `TWOPI`

- added `ASEC360`
- removed `RAD2SEC`
- added `ASEC2RAD`

B.4 Other Internal Code Changes

Many minor changes have been made in the code. Obviously, many of the comment statements had to be revised, and others had to be added; such changes are too numerous to try to list. Some variable names were changed. For example, the output RA and dec for the *mean_star* function was named `mra` and `mdec`, the `m` indicating “mean”; these are now `ira` and `idec`, the `i` indicating “ICRS.” Many similar trivial changes have been made.

B.5 References

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

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

Wallace, P. 2003a, EECT2000 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, 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. 2003c, 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

(Return to Table of Contents)

Appendix C How to Set Up the JPL Ephemerides

C.1 Overview

As described elsewhere in this document, NOVAS requires access to a high-accuracy solar system ephemeris in order to compute places of solar system bodies and the highest-accuracy star places. Groups in the U.S., France, and Russia now construct high-accuracy solar system ephemerides. In NOVAS C, *solarsystem version 1* and *solarsystem version 2* provide direct access to the "developmental ephemerides," designated as "DEnnn," which are produced by JPL in the U.S. The former is a front-end to a USNO-supplied C implementation of JPL's ephemeris-access software, while the latter is a front-end to the JPL-provided Fortran software; both versions enable reading and interpolating a binary, direct-access ephemeris file. The binary ephemeris file is created from ASCII data files and software supplied by JPL. The JPL software must be tailored for your specific computer architecture.

This appendix describes how to set up the binary ephemeris file and the JPL software that reads it on your system. After these tasks have been done successfully, the JPL software must be linked into any NOVAS application that uses *solarsystem version 2*. Alternatively, the binary ephemeris file can be used directly with *solarsystem version 1* and the software in file **eph_manager.c**. The procedures outlined below worked on an Intel-based Mac OS X system using the open-source gfortran compiler and JPL software available in February 2009. The resulting binary, direct-access ephemeris file was successfully transferred to and used on other Intel-based computers running Microsoft Windows XP and Linux. Thus, our procedures are tailored for computers containing Intel processors, including many systems running Microsoft Windows, Mac OS X, and Linux. Providing specific procedures for all combinations of computer processors, operating systems, and compilers is beyond the scope of this user's guide. Furthermore, the procedures in the appendix are intended simply as a guide; the USNO cannot provide technical support regarding JPL software.

C.2 Step-by-Step Guide

Step 1: Connect to the JPL ftp site. All the files needed to install the JPL ephemerides are available via anonymous ftp from `ssd.jpl.nasa.gov`. This can be accomplished through most modern Web browsers by typing
`ftp://ssd.jpl.nasa.gov`
in the address (URL) field. When connected, go to the `pub/eph/planets/` directory.

Step 2: Download the JPL software, ASCII ephemeris files, and corresponding test data file. Follow the "CONTENTS TO BE RETRIEVED BY THE USER" instructions in the file **README.txt**, which is also available as a webpage.[49] *We* recommend *following the instructions for "non-UNIX users" regardless of your computer's operating system*. Our experience at USNO has been that the installation process is more a function of your computer's processor (hardware) than its operating system. The test data file, testpo.xxx, may

[49] http://ssd.jpl.nasa.gov/?planet_eph_export

be found in both the same directory as the ASCII ephemeris files and in the separate test data directory.

The characters used to terminate lines, the end-of-line codes, vary among operating systems. Some systems use carriage returns (CR), some use line feeds (LF), and some use both. The end-of-line codes may not be properly translated between operating systems when files are copied from one system to another, depending on how the files are transferred and on what options are invoked. If you have trouble reading or using any of the files downloaded from JPL, check the encoding to ensure it is appropriate to your system.

Step 3: Configure the **asc2eph.f** file provided by JPL. For computer systems with Intel processors, select (uncomment) the following statement at the beginning of the program:
`PARAMETER ( NRECL = 4 )`

Step 4: Split the **testeph.f** file provided by JPL into two files: **jplsubs.f** and **testeph.f**. The first new file, **jplsubs.f**, should contain JPL subroutines *FSIZER3*, *PLEPH* (including `ENTRY DPLEPH`), *INTERP*, *SPLIT*, *STATE*, and *CONST* extracted from the original JPL test program. The second revised file, **testeph.f**, should contain only the main program and no subroutines. JPL subroutines *FSIZER1* and *FSIZER2* will not be needed and are not retained in either file.

The creation of the new file, **jplsubs.f**, is advantageous for the long-term use of the JPL subroutines and the binary ephemeris files with NOVAS. The new file contains no extraneous material and is named descriptively.

Step 5: Configure the new **jplsubs.f** file.

a. For computer systems with Intel processors, in subroutine *FSIZER3* of **jplsubs.f**, uncomment and set
`NRECL=4`
b. In subroutine *FSIZER3* of **jplsubs.f**, identify the binary ephemeris file to be used by setting
`NAMFIL= 'JPLEPH'`
You may wish to specify a more complete file name including the appropriate directories, such as
`NAMFIL= '/users/mystuff/ephem/JPLEPH'`
to avoid having to keep an alias to this binary file in all your working directories.
c. In subroutine *FSIZER3* of **jplsubs.f**, set `KSIZE` to the correct value for the specific ephemeris being used (see the comments in the code); e.g., set
`KSIZE = 2036`
for DE405.
d. In subroutine *STATE* of **jplsubs.f**, select (uncomment)
`CALL FSIZER3(NRECL,KSIZE,NRFILE,NAMFIL)`
leaving the lines containing `CALL FSIZER1` and `CALL FSIZER2` as comments.

Step 6: Compile and create executables.

a. Create executable application **asc2eph** by compiling and linking file **asc2eph.f**.
b. Create executable application **testeph** by compiling and linking files **testeph.f** and **jplsubs.f**.

NOTE for Windows systems: Some Fortran compilers may produce files with an **.exe** extension, i.e., **asc2eph.exe** and **testeph.exe**. If so, ignore the **.exe** extension when using those applications in the following steps.

Step 7: Concatenate the ASCII data files and convert the concatenated file to binary form. Follow the instructions in the "ASCII to BINARY (for non-UNIX users)" section of the **usrguide** file, which is located in the `pub/eph/planets` directory. The resulting binary ephemeris file will be named **JPLEPH**, or whatever name you specified in Step 5b.

Step 8: Test the binary ephemeris file. Follow the instructions in the "TESTING THE BINARY FILE" section of the **usrguide** file, which is located in `pub/eph/planets`. You should have already downloaded the appropriate test data file in Step 2.

Step 9: Use the binary ephemeris file with NOVAS. After successfully completing the test, compile and link NOVAS function *solarsystem version 2* (**solsys2.c**) and the JPL subroutines contained in **jplsubs.f** with the remaining relevant parts of NOVAS and your application code. Alternatively, the binary ephemeris file can be used directly with *solarsystem version 1* and the software in file **eph_manager.c**. Unless you specified otherwise in Step 5b, the binary JPL ephemeris file, or an alias to it, should reside in the same directory as your executable application.

(Return to Table of Contents)

Appendix D A Comparison of SOFA and NOVAS

The Standards of Fundamental Astronomy (SOFA)[50] library is the official collection of approved software for positional astronomy, operating under the auspices of IAU Division 1 (Fundamental Astronomy). Both Fortran and C editions are available. An international SOFA Reviewing Board manages the collection.

Generally, the C version of NOVAS is independent of the C version of SOFA, although three IERS routines in NOVAS were derived from their SOFA Fortran counterparts. Specifically, NOVAS C3.0 functions *iau2000a* and *iau2000b* were derived from *iau_NUT00A* and *iau_NUT00B*, respectively, in the SOFA Fortran collection. NOVAS C function *ee_ct,* which evaluates the "complementary terms" in the equation of the equinoxes, was derived from *iau_EECT00* in SOFA Fortran.

SOFA Tools for Earth Attitude, also known as the "SOFA Cookbook," contains a detailed example of the transformation between the celestial and terrestrial reference systems. Appendix D of the *User's Guide to NOVAS Version F3.0*[51] (Fortran section of this circular) describes how this example was reproduced using NOVAS. The same input time and Earth orientation parameters were used. Tests that exercised various NOVAS options showed that NOVAS F3.0 duplicated the SOFA results at the sub-microarcsecond level. Because the C and Fortran versions of NOVAS produce nearly identical results, NOVAS C3.0 should have a similar level of agreement with SOFA.

(Return to Table of Contents)

[50] http://www.iausofa.org/index.html
[51] http://www.usno.navy.mil/USNO/astronomical-applications/publications/circ-180