

## NOVAS Routines

This document describes MATLAB versions of several of the subroutines contained in the NOVAS (Naval Observatory Vector Astrometry Subroutines) software package, version F2.0 (1 Nov 98). These functions were created by porting the Fortran versions of the NOVAS software suite to MATLAB.

Additional information about NOVAS can be found at the Naval Observatory web site located at

[http://aa.usno.navy.mil/AA/software/novas/novas\\_info.html](http://aa.usno.navy.mil/AA/software/novas/novas_info.html)

An excellent discussion about the algorithms used in the NOVAS software can be found in “Mean and Apparent Place Computations in the New IAU System. III. Apparent, Topocentric, and Astrometric Places of Planets and Stars”, G. H. Kaplan, J. A. Hughes, P. K. Seidelmann, C. A. Smith and B. D. Yallop, *The Astronomical Journal*, Vol. 97, No. 4, pages 1197-1210, 1989.

### **aberrat.m – correction for the aberration of light**

```
function pos2 = aberrat (pos1, ve, tlight)

% this function corrects position vector for aberration of light.
% algorithm includes relativistic terms. see murray (1981)
% mon. notices royal ast. society 195, 639-648.

% input

% pos1 = position vector, referred to origin at center of
% mass of the earth, components in au
% ve = velocity vector of center of mass of the earth,
% referred to origin at solar system barycenter,
% components in au/day
% tlight = light time from body to earth in days
% if tlight = 0, this function will compute tlight

% output

% pos2 = position vector, referred to origin at center of
% mass of the earth, corrected for aberration,
% components in au
```

### **etilt1.m – orientation of the Earth’s rotation axis**

```
function [oblm, oblt, eqeq, dpsi, deps] = etilt1 (tjd)

% this function computes quantities related to the orientation
% of the earth's rotation axis at julian date tjd

% nutation parameters from a jpl binary ephemeris

% input

% tjd = tdb julian date for orientation parameters
```

## Orbital Mechanics with MATLAB

```
% output

% oblm = mean obliquity of the ecliptic at date tjd (degrees)
% oblt = true obliquity of the ecliptic at date tjd (degrees)
% eqeq = equation of the equinoxes at date tjd (arc seconds)
% dpsi = nutation in longitude at date tjd (arc seconds)
% deps = nutation in obliquity at date tjd (arc seconds)
```

### etilt2.m – orientation of the Earth's rotation axis

```
function [oblm, oblt, eqeq, dpsi, deps] = etilt2 (tjd)

% this function computes quantities related to the orientation
% of the earth's rotation axis at julian date tjd

% nutation parameters obtained from function nod

% input

% tjd = tdb julian date for orientation parameters

% output

% oblm = mean obliquity of the ecliptic at date tjd (degrees)
% oblt = true obliquity of the ecliptic at date tjd (degrees)
% eqeq = equation of the equinoxes at date tjd (arc seconds)
% dpsi = nutation in longitude at date tjd (arc seconds)
% deps = nutation in obliquity at date tjd (arc seconds)
```

### funarg.m – fundamental arguments

```
function [el, elprim, f, d, omega] = funarg (t)

% this function computes fundamental arguments (mean elements)
% of the sun and moon. see seidelmann (1982) celestial
% mechanics 27, 79-106 (1980 iau theory of nutation).

% t = tdb time in julian centuries since j2000.0 (in)
% el = mean anomaly of the moon in radians
% at date tjd (out)
% elprim = mean anomaly of the sun in radians
% at date tjd (out)
% f = mean longitude of the moon minus mean longitude
% of the moon's ascending node in radians
% at date tjd (out)
% d = mean elongation of the moon from the sun in
% radians at date tjd (out)
% omega = mean longitude of the moon's ascending node
% in radians at date tjd (out)
```

## Orbital Mechanics with MATLAB

### geocen.m – move coordinates origin to the Earth center-of-mass

```
function [pos2, tlight] = geocen (pos1, pe)

% this function moves the origin of coordinates from the
% barycenter of the solar system to the center of mass of the
% earth, i.e., this function corrects for parallax.

% input

% pos1 = position vector, referred to origin at solar system
%       barycenter (au)
% pe   = position vector of center of mass of the earth,
%       referred to origin at solar system barycenter (au)

% output

% pos2 = position vector, referred to origin at center of
%       mass of the earth (au)
% tlight = light time from body to earth in days
```

### nod.m – nutations in longitude and obliquity

```
function [dpsi, deps] = nod(jdate)

% this function evaluates the nutation series and returns the
% values for nutation in longitude and nutation in obliquity.
% wahr nutation series for axis b for gilbert & dziewonski earth
% model 1066a. see seidelmann (1982) celestial mechanics 27,
% 79-106. 1980 iau theory of nutation.

% jdate = tdb julian date (in)
% dpsi  = nutation in longitude in arcseconds (out)
% deps  = nutation in obliquity in arcseconds (out)

% NOTE: requires first time initialization via inutate flag
```

### nutatel.m – nutation transformation

```
function pos2 = nutatel (tjd, pos1)

% this function nutates equatorial rectangular coordinates from
% mean equator and equinox of epoch to true equator and equinox of
% epoch. see pages 41-45 of the explanatory supplement to the ae.

% jpl binary ephemeris and etilt1 function

% input

% tjd = tdb julian date of epoch
% pos1 = position vector, geocentric equatorial rectangular
%       coordinates, referred to mean equator and equinox
%       of epoch
```

## *Orbital Mechanics with MATLAB*

```
% output

%   pos2 = position vector, geocentric equatorial rectangular
%           coordinates, referred to true equator and equinox
%           of epoch (out)

% note:
% if tjd is negative, inverse nutation (true to mean) is applied
```

### **nutate2.m – nutation transformation**

```
function pos2 = nutate2 (tjd, pos1)

% this function nutates equatorial rectangular coordinates from
% mean equator and equinox of epoch to true equator and equinox of
% epoch. see pages 41-45 of the explanatory supplement to the ae.

% uses nod function via etilt2

% input

%   tjd = tdb julian date of epoch
%   pos1 = position vector, geocentric equatorial rectangular
%           coordinates, referred to mean equator and equinox
%           of epoch

% output

%   pos2 = position vector, geocentric equatorial rectangular
%           coordinates, referred to true equator and equinox
%           of epoch (out)

% note:
% if tjd is negative, inverse nutation (true to mean) is applied
```

### **pns.w.m - Earth-fixed to space-fixed transformation**

```
function vecs = pns.w (tjd, gast, x, y, vece)

% transforms a vector from earth-fixed system to space-fixed system
% by applying rotations for wobble, spin, nutation, and precession.
% (combined rotation is symbolized p n s w.) specifically,
% it transforms a vector from earth-fixed geographic system to
% space-fixed system based on mean equator and equinox of j2000.0.

% input

%   tjd = tdt julian date
%   gast = greenwich apparent sidereal time (hours)
%   x     = conventionally-defined x coordinate of rotational
%           pole with respect to cio (arc seconds)
%   y     = conventionally-defined y coordinate of rotational
%           pole with respect to cio (arc seconds)
%   vece = vector in geocentric rectangular
%           earth-fixed system, referred to geographic
```

## *Orbital Mechanics with MATLAB*

```
% equator and greenwich meridian

% output

% vecs = vector in geocentric rectangular space-fixed system,
% referred to mean equator and equinox of j2000.0

% note
% tjd = 0 means no precession/nutation transformation
% gast = 0 means no spin transformation
% x = y = 0 means no wobble transformation
```

### **propmo.m – apply proper motion correction**

```
function pos2 = propmo (tjd1, pos1, vell, tjd2)

% this function applies proper motion, including
% foreshortening effects, to a star's position.

% input

% tjd1 = tdb julian date of first epoch
% pos1 = position vector at first epoch
% vell = velocity vector at first epoch
% tjd2 = tdb julian date of second epoch

% output

% pos2 = position vector at second epoch
```

### **solsys.m – interface to JPL ephemerides**

```
function [pos, vel, ierr] = solsys (tjd, body, origin)

% purpose

% this is solsys version '2-da'. it is intended to provide
% an interface between the jpl direct-access solar system
% ephemerides and the 'novas' astrometric function library

% references

% standish, e.m. and newhall, x x (1988). "the jpl
% export planetary ephemeris"; jpl document dated 17 june 1988.

% kaplan, g.h. (1988). "novas: naval observatory vector astrometry
% subroutines"; usno document dated 20 october 1988

% input

% tjd = julian date of the desired time, on the tdb time scale
% body = body identification number for the
% solar system object of interest;
% mercury = 1,...,pluto = 9, sun = 10, moon = 11
% origin = origin code; solar system barycenter = 0,
```

## Orbital Mechanics with MATLAB

```
%           center of mass of the sun = 1

% output

% pos  = position vector of 'body' at tjd;
%       equatorial rectangular coordinates in
%       au referred to the mean equator and
%       equinox of j2000.0
% vel  = velocity vector of 'body' at tjd;
%       equatorial rectangular system referred
%       to the mean equator and equinox of
%       j2000.0, in au/day
% ierr = 0 ... everything ok .
%       = 1 ... invalid value of 'body'
%       = 2 ... invalid value of 'origin'
%       = 3 ... error detected by jpl software;
%       tjd out of range
```

### **spin.m - rotating to non-rotating transformation**

```
function pos2 = spin (st, pos1)

% this subroutine transforms geocentric rectangular coordinates
% from rotating system based on rotational equator and orthogonal
% reference meridian to non-rotating system based on true equator
% and equinox of date.

% input

% st   = local apparent sidereal time at reference meridian (hours)
% pos1 = vector in geocentric rectangular rotating system, referred
%         to rotational equator and orthogonal reference meridian

% output

% pos2 = vector in geocentric rectangular non-rotating system,
%         referred to true equator and equinox of date
```

### **sunfld.m – correct for the deflection of light**

```
function pos2 = sunfld (pos1, pe)

% this function corrects position vector for the deflection
% of light in the gravitational field of the sun. see murray (1981)
% mon. notices royal ast. society 195, 639-648. this function valid
% for bodies within the solar system as well as for stars.

% input

% pos1 = position vector, referred to origin at center of mass
%         of the earth, components in au
% pe   = position vector of center of mass of the earth,
%         referred to origin at center of mass of the sun,
%         components in au
```

## Orbital Mechanics with MATLAB

```
% output

% pos2 = position vector, referred to origin at center of mass
%       of the earth, corrected for gravitational deflection,
%       components in au
```

### **tdtimes.m – compute TDT julian date**

```
function [tdtjd, secdif] = tdtimes (tdbjd)

% this function computes the terrestrial dynamical
% time (tdt) julian date corresponding to a barycentric
% dynamical time (tdb) julian date. expressions used in
% this version are approximations resulting in accuracies
% of about 20 microseconds.

% input

% tdbjd = tdb julian date

% output

% tdtjd = tdt julian date

% secdif = difference tdbjd-tdtjd, in seconds (out)
```

### **terra.m – position and velocity vectors of a terrestrial observer**

```
function [pos, vel] = terra (glon, glat, ht, st)

% this functions computes the position and velocity vectors of
% a terrestrial observer with respect to the center of the earth.

% input

% glon = longitude of observer with respect to reference
%       meridian (east +) in degrees
% glat = geodetic latitude (north +) of observer in degrees
% ht   = height of observer in meters
% st   = local apparent sidereal time at reference meridian
%       in hours
% output

% pos = position vector of observer with respect to center
%       of earth, equatorial rectangular coordinates,
%       referred to true equator and equinox of date,
%       components in au
% vel = velocity vector of observer with respect to center
%       of earth, equatorial rectangular coordinates,
%       referred to true equator and equinox of date,
%       components in au/day

% note: if reference meridian is greenwich and st=0, pos
%       is effectively referred to equator and greenwich
```

**vectrs.m – convert angular quantities to vectors**

```
function [pos, vel] = vectrs (ra, dec, pmra, pmdec, parllx, rv)

% this function converts angular quantities to vectors

% input

% ra      = right ascension (hours)
% dec     = declination (degrees)
% pmra    = proper motion in ra (seconds of time per julian century)
% pmdec   = proper motion in dec per julian century (seconds of arc)
% parllx  = parallax (seconds of arc)
% rv      = radial velocity (kilometers per second)

% output

% pos = position vector, equatorial rectangular coordinates (au)
% vel = velocity vector, equatorial rectangular coordinates (au/day)
```

**wobble.m - polar motion correction**

```
function pos2 = wobble (x, y, pos1)

% this subroutine corrects earth-fixed geocentric rectangular
% coordinates for polar motion. it transforms a vector from
% earth-fixed geographic system to rotating system based on
% rotational equator and orthogonal greenwich meridian through
% axis of rotation.

% input

% x      = conventionally-defined x coordinate of rotational
%         pole with respect to cio (arc seconds)
% y      = conventionally-defined y coordinate of rotational
%         pole with respect to cio (arc seconds)
% pos1   = vector in geocentric rectangular earth-fixed system,
%         referred to geographic equator and greenwich meridian

% output

% pos2 = vector in geocentric rectangular rotating system,
%         referred to rotational equator and orthogonal greenwich meridian
```