Ephemeris Functions

This document describes a series of MATLAB scripts and functions that compute ephemerides of the moon, sun, planets and stars. Routines are provided with different levels of fidelity along with demonstration scripts that illustrate the proper interaction with many of these functions. MATLAB scripts are also provided that calculate both apparent geocentric and topocentric coordinates of these objects using MATLAB versions of Fortran subroutines from the Naval Observatory Vector Astrometry Subroutines (NOVAS) software suite.

The next ten MATLAB functions are based on the algorithms described in *Low-Precision Formulae for Planetary Positions*, T. C. Van Flandern and K. F. Pulkkinen, *The Astrophysical Journal Supplement Series*, **41**:391-411, November 1979. To the precision of this algorithm (one arc minute) these coordinates can be considered to be true-of-date.

Each algorithm uses time arguments given by

$$t = JD - 2451545$$

$$T = t / 36525 + 1$$

where *JD* is the Julian date.

Each function uses fundamental trigonometric arguments (in revolutions) of the following form:

$$G_s = 0.993126 + 0.00273777850t$$

$$G_2 = 0.140023 + 0.00445036173t$$

$$G_4 = 0.053856 + 0.00145561327t$$

$$G_5 = 0.056531 + 0.00023080893t$$

$$F_4 = 0.849694 + 0.00145569465t$$

$$L_4 = 0.987353 + 0.00145575328t$$

The heliocentric, ecliptic longitude λ, latitude β and distance *r* are computed from series involving these arguments. These series are of the form

$$\lambda = L_4 + 38451\sin G_4 + 2238\sin\left(2G_4\right) + 181\sin\left(3G_4\right) + \ldots$$

$$\beta = 6603\sin F_4 + 622\sin\left(G_4 - F_4\right) + 615\sin\left(G_4 + F_4\right) + \ldots$$

$$r = 1.53031 - 0.1417\cos G_4 - 0.0066\cos\left(2G_4\right) + \ldots$$

where the unit of *r* is Astronomical Units (AU). In these functions, the value of AU is equal to `149597870.691` kilometers.

The heliocentric, ecliptic position vector of the planet is determined from

$$\mathbf{r} = r\begin{Bmatrix} \cos\beta\cos\lambda \\ \cos\beta\sin\lambda \\ \sin\beta \end{Bmatrix}$$

sun1.m – solar ephemeris

This function calculates the true-of-date geocentric right ascension, declination and position vector of the Sun.

The syntax of this MATLAB function is

```
function [rasc, decl, rsun] = sun1 (jdate)

% solar ephemeris

% input

%  jdate = Julian date

% output

%  rasc = right ascension of the sun (radians)
%         (0 <= rasc <= 2 pi)
%  decl = declination of the sun (radians)
%         (-pi/2 <= decl <= pi/2)
%  rsun = eci position vector of the sun (km)
```

moon.m – lunar ephemeris

This function calculates the true-of-date geocentric right ascension, declination and position vector of the Moon.

The syntax of this MATLAB function is

```
function [rasc, decl, rmoon] = moon (jdate)

% lunar ephemeris

% input

%  jdate = julian date

% output

%  rasc  = right ascension of the moon (radians)
%          (0 <= rasc <= 2 pi)
%  decl  = declination of the moon (radians)
%          (-pi/2 <= decl <= pi/2)
%  rmoon = eci position vector of the moon (km)
```

mercury.m – Mercury ephemeris

This function calculates the true-of-date heliocentric ecliptic position vector of Mercury.

The syntax of this MATLAB function is

```
function rmercury = mercury (jdate)

% true-of-date heliocentric, ecliptic
% position vector of Mercury

% input

%  jdate = julian date

% output

%  rmercury = position vector of Mercury (km)
```

venus.m – Venus ephemeris

This function calculates the true-of-date heliocentric ecliptic position vector of Venus.

The syntax of this MATLAB function is

```
function rvenus = venus (jdate)

% true-of-date heliocentric, ecliptic
% position vector of venus

% input

%  jdate = Julian date

% output

%  rvenus = position vector of Venus (km)
```

earth.m – Earth ephemeris

This function calculates the true-of-date heliocentric ecliptic position vector of the Earth.

The syntax of this MATLAB function is

```
function rearth = earth (jdate)

% true-of-date heliocentric, ecliptic
% position vector of the Earth

% input

%  jdate = Julian date
% output

%  rearth = position vector of the Earth (km)
```

mars.m – Mars ephemeris

This function calculates the true-of-date heliocentric ecliptic position vector of Mars.

The syntax of this MATLAB function is

```
function rmars = mars (jdate)

% true-of-date heliocentric, ecliptic
% position vector of Mars

% input

%  jdate = Julian date

% output

%  rmars = position vector of Mars (km)
```

jupiter.m – Jupiter ephemeris

This function calculates the true-of-date heliocentric ecliptic position vector of Jupiter.

The syntax of this MATLAB function is

```
function rjupiter = jupiter (jdate)

% true-of-date heliocentric, ecliptic
% position vector of Jupiter

% input

%  jdate = Julian date

% output

%  rjupiter = position vector of Jupiter (km)
```

saturn.m – Saturn ephemeris

This function calculates the true-of-date heliocentric ecliptic position vector of Saturn.

The syntax of this MATLAB function is

```
function rsaturn = saturn (jdate)

% true-of-date heliocentric, ecliptic
% position vector of Saturn

% input

%  jdate = Julian date

% output

%  rsaturn = position vector of Saturn (km)
```

uranus.m – Uranus ephemeris

This function calculates the true-of-date heliocentric ecliptic position vector of Uranus.

The syntax of this MATLAB function is

```
function ruranus = uranus (jdate)

% true-of-date heliocentric, ecliptic
% position vector of Uranus

% input

%  jdate = Julian date

% output

%  ruranus = position vector of Uranus (km)
```

neptune.m – Neptune ephemeris

This function calculates the true-of-date heliocentric ecliptic position vector of Neptune.

The syntax of this MATLAB function is

```
function rneptune = neptune (jdate)

% true-of-date heliocentric, ecliptic
% position vector of Neptune

% input

%  jdate = Julian date

% output

%  rneptune = position vector of Neptune (km)
```

pluto.m – Pluto ephemeris

This function calculates the heliocentric position vector of Pluto relative to the ecliptic and equinox of J2000. This algorithm is based on the method described in Chapter 36 of *Astronomical Algorithms* by Jean Meeus.

The fundamental time argument for this method is a function of the Julian Ephemeris Date *JED* as follows:

$$T = \frac{JED - 2451545}{36525}$$

The heliocentric ecliptic coordinates of Pluto are computed from series of the form

$$\lambda = \lambda^m + \sum_{i=1}^{43} A \sin\alpha + B\cos\alpha$$

where

$$\begin{aligned} \lambda^m &= \text{ coordinate mean value} \\ \alpha &= iJ + jS + kP \\ J, S, P &= \text{ mean longitudes of Jupiter, Saturn and Pluto} \\ i, j, k &= \text{ integer constants} \\ A, B &= \text{ coefficients of periodic term} \end{aligned}$$

The syntax of this MATLAB function is

```
function rpluto = pluto(jdate)

% heliocentric coordinates of pluto

% heliocentric position vector of Pluto
% ecliptic and equinox of J2000

% input

%  jdate = Julian date

% output

%  rpluto = position vector of Pluto (km)
```

sun2.m – precision Sun ephemeris

This MATLAB function computes a true-of-date geocentric ephemeris of the Sun based on the data and numerical methods described in the book, *Planetary Programs and Tables* by Pierre Bretagnon and Jean-Louis Simon.

The fundamental time argument of this method is the number of days relative to the Julian epoch January 1, 2000 normalized with respect to 3652500 Julian days. This value can be calculated for any Julian Ephemeris Date *JED* with the following expression

$$U = \frac{JED - 2451545}{3652500}$$

The geocentric, ecliptic *mean* longitude of the Sun is calculated with a trigonometric series of the form

$$\lambda_s^m = \lambda_0 + \lambda_1 U + \sum_{i=1}^{50} l_i \sin\left(\alpha_i + \nu_i U\right)$$

The geocentric distance of the Sun is calculated with another series of the form

$$r_s = r_0 + r_1 U + \sum_{i=1}^{50} r_i \cos\left(\alpha_i + \nu_i U\right)$$

The longitude of the Sun is corrected for the effect of *aberration* (in radians) with the following equation:

$$\Delta\lambda_s^a = 10^{-7}\left\{-993 + 17\cos\left(3.10 + 62830.14U\right)\right\}$$

The nutation in longitude (in radians) is calculated from

$$\Delta\psi = 10^{-7}\left(-834\sin A_1 - 64\sin A_2\right)$$

where

$$A_1 = 2.18 - 3375.70U + 0.36U^2$$
$$A_2 = 3.51 + 125666.39U + 0.10U^2$$

The apparent, geocentric ecliptic longitude of the Sun is determined as the combination of these three components with the next equation

$$\lambda_s = \lambda_s^m + \Delta\lambda_s^a + \Delta\psi$$

The three components of the geocentric, ecliptic position vector of the Sun are given by

$$x_s = r_s \cos\lambda_s$$
$$y_s = r_s \sin\lambda_s$$
$$z_s = 0$$

The apparent geocentric, equatorial right ascension α_s and declination δ_s of the Sun can be found from

$$\alpha_s = \tan^{-1}\left(\cos\lambda_s, \sin\varepsilon\sin\lambda_s\right)$$
$$\delta_s = \sin^{-1}\left(\sin\varepsilon\sin\lambda_s\right)$$

where ε is the true obliquity of the ecliptic. This number is calculated from the mean obliquity of the ecliptic ε_m and the nutation in obliquity $\Delta\varepsilon$ in this function with the following expressions:

$$\varepsilon = \varepsilon_m + \Delta\varepsilon$$
$$\varepsilon_m = 10^{-7}\left(4090928 + 446\cos A_1 + 28\cos A_2\right)$$
$$\Delta\varepsilon = 10^{-7}U\left(-226938 + U\left(-75 + U\left(96926 + U\left(-2491 - 12104U\right)\right)\right)\right)$$

Finally, we can compute the three components of the *apparent*, geocentric equatorial position vector of the Sun with the following three expressions:

$$r_x = r\cos\alpha_s\cos\delta_s$$
$$r_y = r\sin\alpha_s\cos\delta_s$$
$$r_z = r\sin\delta_s$$

where *r* is the geocentric distance of the Sun.

This function requires initialization the first time it is called. The following statement in the main MATLAB script will accomplish this:

```
suncoef = 1;
```

This variable should also be placed in a `global` statement at the beginning of the main script that calls this function.

The syntax of this MATLAB function is

```
function [rasc, decl, rsun] = sun2 (jdate)

% precision ephemeris of the Sun

% input

%  jdate = julian ephemeris date

% output

%  rasc = right ascension of the Sun (radians)
%         (0 <= rasc <= 2 pi)
%  decl = declination of the Sun (radians)
%         (-pi/2 <= decl <= pi/2)
%  rsun = eci position vector of the Sun (km)
```

elp2000.m – osculating orbital elements of the moon

This function calculates the osculating classical orbital elements of the Moon in the mean ecliptic and mean equinox of date coordinate system. It is based on the book *Lunar Tables and Programs From 4000 B.C. TO A.D. 8000* by Michelle Chapront-Touze and Jean Chapront. This book and its optional companion software are available from Willmann-Bell (www.willbell.com).

The fundamental time argument of this algorithm is

$$t = \frac{JED - 2451545}{36525}$$

where *JED* is the Julian Ephemeris Date.

The osculating orbital elements are calculated from series of the form

$$a = 383397.6 + S_a + tS_a''$$

where

$$S_a = \sum_{n=1}^{30} a_n \cos\left(\xi_n^{(0)} + \xi_n^{(1)} t + \xi_n^{(2)} 10^{-4} t^2 + \xi_n^{(3)} 10^{-6} t^3 + \xi_n^{(4)} 10^{-8} t^4\right)$$

$$S_a'' = \sum_{n=1}^{3} a_n'' \cos\left(\xi_n''^{(0)} + \xi_n''^{(1)} t\right)$$

The first few multipliers and trigonometric arguments for this orbital element are

n	a_n	$\xi_n^{(0)}$	$\xi_n^{(1)}$	$\xi_n^{(2)}$	$\xi_n^{(3)}$	$\xi_n^{(4)}$
1	3400.4	235.7004	890534.2230	-32.601	3.664	-1.769
2	-635.6	100.7370	413335.3554	-122.571	-10.684	5.028

n	a_n''	$\xi_n''^{(0)}$	$\xi_n''^{(1)}$
1	-0.55	238.2	854535.2
2	0.10	103.2	377336.3

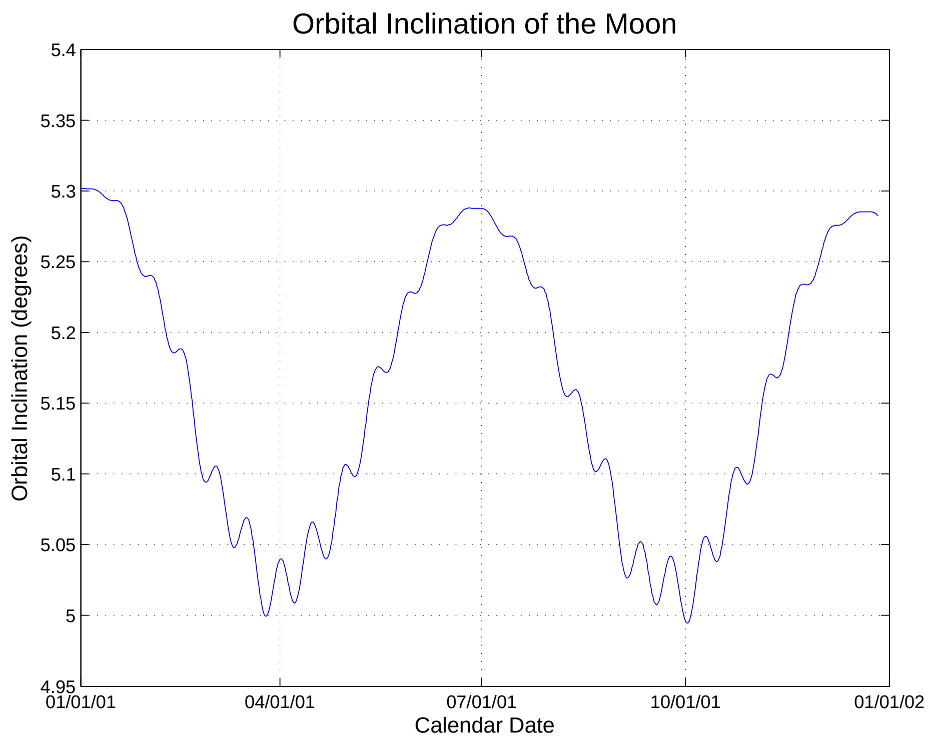
The syntax for this MATLAB function is

```
function oev = elp2000(tjd)

% osculating orbital elements of the moon

% mean ecliptic and mean equinox of date

% input
%  tjd = tdt Julian date

% output

%  oev(1)  = semimajor axis (kilometers)
%  oev(2)  = orbital eccentricity (non-dimensional)
%            (0 <= eccentricity < 1)
%  oev(3)  = orbital inclination (radians)
%            (0 <= inclination <= pi)
%  oev(4)  = argument of perigee (radians)
%            (0 <= argument of perigee <= 2 pi)
%  oev(5)  = right ascension of ascending node (radians)
%            (0 <= raan <= 2 pi)
%  oev(6)  = true anomaly (radians)
%            (0 <= true anomaly <= 2 pi)
```

This software suite includes a script called `demo_elp.m` that demonstrates how to interact with this function. The following is a typical user interaction with this script.

```
         program demo_elp

< orbital elements of the moon >


please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,2001


please input the simulation period (days)
? 360


please input the graphics step size (days)
? .25

please select the orbital element to plot

  <1> semimajor axis
  <2> eccentricity
  <3> orbital inclination
  <4> argument of perigee
  <5> right ascension of the ascending node
  <6> true anomaly
  <7> apogee radius
  <8> perigee radius
  <9> geocentric distance

? 3
```

The following is the graphics display created for this example.



planet.m – mean planetary ephemerides

This MATLAB function calculates the position and velocity vectors of the planets with respect to the mean ecliptic and equinox of date. These calculations are based on the algorithm described in Chapter 30 of *Astronomical Algorithms* by Jean Meeus. Each orbital element is represented by a cubic polynomial of the form

$$a_0 + a_1 T + a_2 T^2 + a_3 T^3$$

where the fundamental time argument T is given by

$$T = \frac{JED - 2451545}{36525}$$

In this expression JED is the Julian ephemeris date.

The syntax of this MATLAB function is

```
function [r, v] = planet(ip, jdate)

% planetary ephemeris

% input

%  ip    = planet index (1 = mercury, 2 = venus, etc.)
%  jdate = Julian date

% output

%  r = heliocentric position vector (kilometers)
%  v = heliocentric velocity vector (kilometers/second)
```

p2000.m – J2000 planetary ephemerides

This MATLAB function calculates the position and velocity vectors of the planets with respect to the mean ecliptic and equinox of J2000. These calculations are based on the algorithm described in Chapter 30 of *Astronomical Algorithms* by Jean Meeus. Each orbital element is represented by a cubic polynomial of the form

$$a_0 + a_1 T + a_2 T^2 + a_3 T^3$$

where the fundamental time argument *T* is given by

$$T = \frac{JED - 2451545}{36525}$$

In this expression *JED* is the Julian ephemeris date.

The syntax of this MATLAB function is

```
function [r, v] = p2000(ip, jdate)

% planetary ephemeris

% input

%  ip    = planet index (1 = mercury, 2 = venus, etc.)
%  jdate = julian date

% output

%  r = heliocentric position vector (kilometers)
%  v = heliocentric velocity vector (kilometers/second)

% Note: coordinates are with respect to the
%       ecliptic and equinox of J2000.
```

pecl2000.m – J2000 ecliptic planetary ephemerides – JPL binary source

This MATLAB function calculates the position and velocity vectors of the planets with respect to the Earth mean *ecliptic* and equinox of J2000. The source ephemeris is a JPL binary file which provides planetary coordinates in the Earth mean equator and equinox of J2000 system.

These state vectors are transformed to the ecliptic frame using the following transformation matrix:

$$\begin{bmatrix} 1 & -0.000000479966 & 0 \\ 0.000000440360 & 0.917482137087 & 0.397776982902 \\ -0.000000190919 & -0.397776982902 & 0.917482137087 \end{bmatrix}$$

The syntax of this MATLAB function is

```
function [r, v] = pecl2000(ntarg, jdate)

% heliocentric planetary state vector

% earth mean ecliptic and equinox of j2000
% coordinate system (jpl ephemeris)

% input

%  jdate = TDB julian date
%  ntarg = "target" body

% output

%  r = position vector (kilometers)
%  v = velocity vector (kilometers/second)

% NOTE: requires equatorial to ecliptic
%       transformation matrix eq2000 via global

% eq2000 = [+1.000000000000d0 -0.000000479966d0  0.000000000000d0]
%          [+0.000000440360d0 +0.917482137087d0 +0.397776982902d0]
%          [-0.000000190919d0 -0.397776982902d0 +0.917482137087d0]
```

Please note that the `eq2000` transformation matrix must be provided to this function using a `global eq2000` statement in this function and the main script. The main script provides this matrix using the following MATLAB source code.

```
% equatorial-to-ecliptic transformation matrix

eq2000 = [[ +1.000000000000d0 -0.000000479966d0  0.000000000000d0]; ...
          [ +0.000000440360d0 +0.917482137087d0 +0.397776982902d0]; ...
          [ -0.000000190919d0 -0.397776982902d0 +0.917482137087d0]];
```

slp96.m and demo_slp.m – lunar and planetary ephemeris - Bureau of Longitudes/IMCCE

This MATLAB function and script demonstrate how to read SLP96 binary ephemeris files created using the Fortran programs and ASCII data files provided by the Institut de Mecanique Celeste et de Calcul des Ephemerides (IMCCE), and calculate the position and velocity vectors of a planet or the moon in either the J2000 FK5 or dynamical coordinate systems. The `slp96.m` function was ported to MATLAB using the Fortran source code subroutine provided by the IMCCE. These ASCII files and source code can be found on the Internet at ftp://ftp.imcce.fr/pub/ephem/sun/slp96/.

The `slp96` function requires initialization the first time it is called. The following statement in the main MATLAB script will accomplish this:

```
islp96 = 1;
```

This variable should also be placed in a `global` statement at the beginning of the main script which calls this function.

`demo_slp` is a simple MATLAB script that demonstrates how to call the `slp96` routine. A sample binary file (`slp96.bin`) for IBM-PC compatible computers can be downloaded from the Celestial and Orbital Mechanics web site.

The following is a summary of the inputs and outputs for this MATLAB function:

```
function [result, ierr] = slp96 (tjd, ibody, icent, ipv, iframe, icoord)

% solar, lunar and planetary coordinates j2000

% input

%     tjd       julian date tdb

%     ibody     body index

%               ibody=01 : mercury       ibody=07 : uranus
%               ibody=02 : venus         ibody=08 : neptune
%               ibody=03 : e-m baryc.    ibody=09 : void
%               ibody=04 : mars          ibody=10 : moon
%               ibody=05 : jupiter       ibody=11 : sun
%               ibody=06 : saturn        ibody=12 : earth

%     icent     frame center index

%               icent=00 : barycenter of solar system
%               icent=01 : mercury       icent=07 : uranus
%               icent=02 : venus         icent=08 : neptune
%               icent=03 : e-m baryc.    icent=09 : void
%               icent=04 : mars          icent=10 : moon
%               icent=05 : jupiter       icent=11 : sun
%               icent=06 : saturn        icent=12 : earth

%     ipv       position-velocity index

%               ipv=1 : position.
%               ipv=2 : position and velocity.

%     iframe    frame index

%               iframe=1 : equinox and equator  j2000 (fk5)
%               iframe=2 : equinox and ecliptic j2000 (dynamical)

%     icoord    coordinates index
%               icoord=1 : rectangular coordinates
%               icoord=2 : spherical coordinates

%     fname     name of the slp96 data file
%               note: passed via global

% output

%     result    results table

%               rectangular coordinates (icoord = 1)

%               position

%               result(1) : equatorial or ecliptic component x (au)
%               result(2) : equatorial or ecliptic component y (au)
%               result(3) : equatorial or ecliptic component z (au)

%               velocity

%               result(4) : equatorial or ecliptic component x (au/day)
%               result(5) : equatorial or ecliptic component y (au/day)
%               result(6) : equatorial or ecliptic component z (au/day)

%               spherical coordinates (icoord = 2)

%               position

%               result(1) : right ascension or longitude (radians)
%               result(2) : declination or latitude (radians)
%               result(3) : geometric distance (au)

%               velocity

%               result(4) : right ascension or longitude (rad/day)
%               result(5) : declination or latitude (rad/day)
%               result(6) : geometric distance (au/day)

%     ierr      error index

%               ierr=0  : no error
%               ierr=10 : file error
%               ierr=11 : date error (tjd)
%               ierr=12 : body error (ibody)
%               ierr=13 : frame center error (icent)
%               ierr=14 : position-velocity error (ipv)
%               ierr=15 : frame error (iframe)
%               ierr=16 : coordinates error (icoord)
```

The processing steps used in this script are as follows:

- input a calendar date on the UTC time scale
- compute the UTC Julian date
- compute the TDB Julian date
- select the target body

- select the central body
- define type of coordinate system
- define type of coordinate calculations
- specify the name of the SLP96 binary data file
- evaluate the ephemeris

The following is a typical user interaction with this script.

```
demo_slp - demonstrates how to use the slp96.m function


please input a UTC calendar date between 12/14/1949 and 1/2/2050

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 10,21,1998

please input a value for TAI-UTC in seconds
? 31

    target body menu

  <1> Mercury
  <2> Venus
  <3> Earth-Moon barycenter
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> void
  <10> Moon
  <11> Sun
  <12> Earth

please select the target body
? 10

    central body menu

  <0> solar system barycenter
  <1> Mercury
  <2> Venus
  <3> Earth-Moon barycenter
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> void
  <10> Moon
  <11> Sun
  <12> Earth

please select the central body
```

```
? 12


    coordinate system menu

  <1> equinox and equator j2000 (fk5)

  <2> equinox and ecliptic j2000 (dynamical)

please select the coordinate system type
? 1


    coordinate type menu

  <1> rectangular coordinates

  <2> spherical coordinates

please select the coordinate type
? 1
```

The following is the program output created for this example.

```
program demo_slp

equinox and equator j2000 (fk5)

target body        'Moon'

central body       'Earth'


UTC calendar date       21-Oct-1998

UTC Julian date          2451107.500000

TDB Julian date          2451107.500731


state vector

       rx (km)                 ry (km)                 rz (km)                rmag (km)
 -3.37366628520218e+005  -2.19024631174909e+005  -5.98176564646602e+004  +4.06652410696565e+005

       vx (kps)                vy (kps)                vz (kps)               vmag (kps)
 +5.39083860486982e-001  -7.55355619416334e-001  -2.86179804099060e-001  +9.71119148557765e-001
```

jplephem.m and demo_jpl.m – JPL lunar and planetary ephemeris

This MATLAB function and main script demonstrate how to read binary ephemeris files created using the Fortran programs and ASCII data files provided by the Jet Propulsion Laboratory, and calculate the position and velocity vectors of a planet, the sun or the Moon. The `jplephem.m` function was ported to MATLAB using the Fortran source code subroutine provided by JPL. These ASCII files can be found on the Internet at ftp://ssd.jpl.nasa.gov/pub/eph/planets/ascii/. A

CD ROM containing these ASCII data files and the companion Fortran source code is also available from Willmann-Bell, www.willbell.com.

The `jplephem` function requires initialization the first time it is called. The following statement in the main MATLAB script will accomplish this:

```
iephem = 1;
```

This variable should also be placed in a `global` statement at the beginning of the main script which calls this function. The value of the Astronomical Unit, in kilometers, used in a particular JPL ephemeris is available as the constant `au` which is placed in global by the `jplephem` function described below.

`demo_jpl` is a simple MATLAB script that demonstrates how to call the `jplephem` routine. Binary ephemeris files for IBM-PC compatible computers can be downloaded from the Celestial and Orbital Mechanics web site. The coordinates calculated by this function are with respect to the International Celestial Reference System (ICRS) which is described in "The International Celestial Reference Frame as Realized by Very Long Baseline Interferometry", C. Ma, E. Arias, T. Eubanks, A. Fey, A. Gontier, C. Jacobs, O. Sovers, B. Archinal and P. Charlot, *The Astronomical Journal*, 116:516-546, 1998, July.

The following is the syntax for this MATLAB function:

```
function rrd = jplephem (et, ntarg, ncent)

% reads the jpl planetary ephemeris and gives
% the position and velocity of the point 'ntarg'
% with respect to point 'ncent'

% input

%   et    = julian ephemeris date at which interpolation is wanted

%   ntarg = integer number of 'target' point

%   ncent = integer number of center point

%   the numbering convention for 'ntarg' and 'ncent' is:

%        1 = mercury            8 = neptune
%        2 = venus              9 = pluto
%        3 = earth             10 = moon
%        4 = mars              11 = sun
%        5 = jupiter           12 = solar-system barycenter
%        6 = saturn            13 = earth-moon barycenter
%        7 = uranus            14 = nutations (longitude and obliq)
%                              15 = librations, if on ephemeris file

%        if nutations are wanted, set ntarg = 14.
%        for librations, set ntarg = 15.  set ncent = 0.

% output
%   rrd = output 6-word array containing position and velocity
%         of point 'ntarg' relative to 'ncent'.  the units are au and
%         au/day.  for librations the units are radians and radians
%         per day.  in the case of nutations the first four words of
%         rrd will be set to nutations and rates, having units of
%         radians and radians/day.

%         the option is available to have the units in km and km/sec.
%         for this, set km = 1 via global in the calling program.
```

The following is a typical user interaction with this script using the DE421 ephemeris file.

```
demo_jpl - demonstrates how to use the jplephem.m function

please input a UTC calendar date

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 10,21,1998

please input a value for TAI-UTC in seconds
? 31

    target body menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto
  <10> Moon
  <11> Sun

please select the target body
? 10

    central body menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto
  <10> Moon
  <11> Sun
  <12> solar-system barycenter
  <13> Earth-Moon barycenter
please select the central body
? 3
```

The following is the script output for this example.

```
 program demo_jpl


 ephemeris file     de421.bin

 target body        'Moon'

 central body       'Earth'


 UTC calendar date       21-Oct-1998

 UTC Julian date          2451107.500000

 TDB Julian date          2451107.500731


 state vector

        rx (km)                 ry (km)                 rz (km)                 rmag (km)
 -3.37366639992093e+005  -2.19024588322606e+005  -5.98176034774959e+004  +4.06652389339142e+005

        vx (kps)                vy (kps)                vz (kps)                vmag (kps)
 +5.39083723067669e-001  -7.55355631361622e-001  -2.86179850104219e-001  +9.71119095122677e-001
```

Please note the following extracted from JPL IOM 343R-08-003, dated 31-March 2008.

In a resolution adopted by the International Astronomical Union in 2006 (GA26.3), the time scale TDB (Temps Dynamique Barycentrique, Barycentric Dynamical Time) was defined to be consistent with the JPL ephemeris time.

The axes of the ephemeris are oriented with respect to the International Celestial Reference Frame (ICRF).

Furthermore, the relationship between the ICRF and EME 2000 frames is described in the `moon_080317.tf` text file which is located on the Internet at ftp://naif.jpl.nasa.gov/pub/naif/generic_kernels/fk/satellites/,

The IERS Celestial Reference Frame (ICRF) is offset from the J2000 reference frame (equivalent to EME 2000) by a small rotation; the J2000 pole offset magnitude is about 18 milliarcseconds (mas) and the equinox offset magnitude is approximately 78 milliarcseconds.

A MATLAB function (`j2000_icrs.m`) that transforms vectors between these two systems follows later in this section.

jplephem_inpop.m and demo_inpop.m – INPOP06C binary ephemeris

This MATLAB function and script demonstrate how to read the INPOP06C binary ephemeris file and calculate the position and velocity vectors of a planet, the sun or the Moon. The `jplephem_inpop.m` function is identical to the `jplephem.m` described above with a minor update for the format of the INPOP06C binary file. This binary file for several computer platforms can be found at http://www.imcce.fr/fr/presentation/equipes/ASD/inpop/.

The `jplephem_inpop` function requires initialization the first time it is called. The following statement in the main MATLAB script will accomplish this:

```
      iephem = 1;
```

This variable should also be placed in a `global` statement at the beginning of the main script which calls this function.

`demo_inpop` is a simple MATLAB script that demonstrates how to interact with the `jplephem_inpop` routine.

The following is a typical user interaction with this script.

```
demo_inpop - demonstrates how to use the jplephem_inpop.m function

please input a UTC calendar date

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 10,21,1998

please input a value for TAI-UTC in seconds
? 31

    target body menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto
  <10> Moon
  <11> Sun

please select the target body
? 10

    central body menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto
  <10> Moon
  <11> Sun
  <12> solar-system barycenter
  <13> Earth-Moon barycenter

please select the central body
? 3
```

The following is the script output for this example.

```
  program demo_inpop


  ephemeris file     inpop06c.bin

  target body        'Moon'

  central body       'Earth'


  UTC calendar date       21-Oct-1998

  UTC Julian date           2451107.500000

  TDB Julian date           2451107.500731


  state vector

         rx (km)                  ry (km)                  rz (km)                rmag (km)
  -3.37366643255620e+005  -2.19024582193350e+005  -5.98176002387135e+004  +4.06652388268967e+005

         vx (kps)                 vy (kps)                 vz (kps)               vmag (kps)
  +5.39083706462739e-001  -7.55355639411607e-001  -2.86179853720789e-001  +9.71119093232223e-001
```

hcb.m – ephemeris of a heliocentric celestial body

This MATLAB function determines the position and velocity vectors of a heliocentric celestial body such as an asteroid or comet.

The following is the syntax for this MATLAB function:

```
   function [r, v] = hcb(jdate, oev)

   % ephemeris of a heliocentric celestial body

   % input
```

```
%  jdate  = julian date
%  oev(1) = radius at perihelion passage (au)
%  oev(2) = orbital eccentricity (non-dimensional)
%  oev(3) = orbital inclination (radians)
%  oev(4) = argument of perihelion (radians)
%  oev(5) = longitude of ascending node (radians)
%  oev(6) = julian date at perihelion passage

% output

%  r = heliocentric position vector (kilometers)
%  v = heliocentric velocity vector (kilometers/second)

% Note: state vector is output in the same coordinate
%       system as the input orbital elements
```

The state vector is output in the same coordinate system as the input orbital elements.

The orbital elements of an asteroid or comet relative to the ecliptic and equinox of J2000 coordinate system can be obtained from the JPL Small-Body Database Browser (http://ssd.jpl.nasa.gov/sbdb.cgi) or the MPC database at Harvard (http://cfa-www.harvard.edu).

These orbital elements consist of the following items:

- calendar date of perihelion passage
- perihelion distance (AU)
- orbital eccentricity (non-dimensional)
- orbital inclination (degrees)
- argument of perihelion (degrees)
- longitude of ascending node (degrees)

The software determines the mean anomaly of the asteroid or comet at any simulation time *JD* using the following equation:

$$M = \sqrt{\frac{\mu_s}{a^3}}\, t_{pp} = \sqrt{\frac{\mu_s}{a^3}}\left(JD - JD_{pp}\right)$$

where μ_s is the gravitational constant of the sun, *a* is the semimajor axis of the celestial body, and t_{pp} is the time since perihelion passage.

The semimajor axis is determined from the perihelion distance r_p and orbital eccentricity *e* according to

$$a = \frac{r_p}{(1-e)}$$

This MATLAB function uses `kepler1.m` to calculate the true anomaly.

j2000_icrs.m – J2000-to-ICRS transformation matrix

This MATLAB function returns the transformation matrix from the mean dynamical equator and equinox at J2000 to the International Celestial Reference System (ICRS). There are two options for this matrix based on different data types.

The following is the syntax for this MATLAB function:

```
function tmatrix = j2000_icrs(itype)

% transformation matrix from the mean dynamical
% equator and equinox at j2000 to the
% International Celestial Reference System (ICRS)

% input

%  itype = type of transformation
%      1 = LLR + VLBI
%      2 = Chapront et al. LLR

% output

%  tmatrix = transformation matrix

% reference: Rotation Matrix from the Mean Dynamical
% Equator and Equinox at J2000.0 to the ICRS, Astronomy
% and Astrophysics, October 1, 2003.
```

To transform coordinates from the ICRS system to the J2000 system, the programmer should use the transpose of the matrix returned by this function.

applan1.m – apparent coordinates of a planet or the Moon - JPL ephemeris

This MATLAB function determines the apparent geocentric and topocentric coordinates of a planet or the Moon. The source ephemeris for this routine is a JPL binary file. The algorithms used in this function are based on the Fortran versions found in `NOVAS`.

The following is the syntax for this function.

```
function [ra, dec, dis] = applan1 (tjd, ujd, l, n, topo, glon, glat, ht)

% this function computes the apparent geocentric or topocentric place
% of a planet or other solar system body.  rectangular coordinates of
% solar system bodies are obtained from function solsys.

% input

%  tjd  = tdt julian date for apparent geocentric place
%  ujd  = ut1 julian date for apparent topocentric place
%  l    = body identification number for desired planet
%  n    = body identification number for the earth
%  topo = type of apparent place calculation
%       = 0 ==> geocentric
%       = 1 ==> topocentric
%  glon = geodetic longitude of observer (east +, degrees)
%  glat = geodetic latitude of observer (north +, degrees)
%  ht   = height of observer (meters)

% output

%  ra  = apparent geocentric or topocentric right ascension,
%        referred to true equator and equinox of date (hours)
%  dec = apparent geocentric or topocentric declination,
%        referred to true equator and equinox of date (degrees)
%  dis = true distance from earth to planet (astronomical units)
```

This software suite includes a script called `demo_aplanet1` that demonstrates how to use this function. The following is a typical user interaction with this demonstration script.

```
   demo_aplanet1 - apparent coordinates


please input a UTC calendar date between 12/18/1997 and 1/20/2010

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 10,20,30

    target body menu

  <1> Mercury
  <2> Venus
  <3> Earth
  <4> Mars
  <5> Jupiter
  <6> Saturn
  <7> Uranus
  <8> Neptune
  <9> Pluto
  <10> Sun
  <11> Moon

please select the target body
? 11

coordinate type menu

 <1> geocentric

 <2> topocentric
```

```
please select coordinate type
? 2

please input the geographic latitude of the ground site
(-90 <= degrees <= +90, 0 <= minutes <= 60, 0 <= seconds <= 60)
(north latitude is positive, south latitude is negative)
? 40,0,0

please input the geographic longitude of the ground site
(0 <= degrees <= 360, 0 <= minutes <= 60, 0 <= seconds <= 60)
(east longitude is positive, west longitude is negative)
? -105,0,0

please input the altitude of the ground site (meters)
(positive above sea level, negative below sea level)
? 1000
```

The following is the program output created for this example.

```
apparent topocentric coordinates

'Moon'


UTC calendar date       01-Jan-1998

universal time          10:20:30

UTC julian date          2450814.9309

observer latitude        -105d 00m 00.00s

observer east longitude  +40d 00m 00.00s

observer altitude           1000.0000 meters

right ascension          +21h 12m 2.9587s

declination              -14d 17m 47.57s

topocentric distance      376284.4187 kilometers
```

applan2.m – apparent coordinates of a planet or the Moon - SLP96 ephemeris

This MATLAB function determines the apparent geocentric and topocentric coordinates of a planet or the Moon. The source ephemeris for this routine is the SLP96 binary file `slp96.bin`. The algorithms used in this function are based on the Fortran versions found in `NOVAS`.

The following is the syntax for this function.

```
function [ra, dec, dis] = applan2(tjd, ujd, l, n, topo, glon, glat, ht)

% this function computes the apparent geocentric or topocentric place
% of a planet or other solar system body.  rectangular coordinates of
% solar system bodies are obtained from function slp96.

% input

%  tjd  = tdt julian date for apparent geocentric place
%  ujd  = ut1 julian date for apparent topocentric place
%  l    = body identification number for desired planet
%  n    = body identification number for the earth
%  topo = type of apparent place calculation
%       = 0 ==> geocentric
%       = 1 ==> topocentric
%  glon = geodetic longitude of observer (east +, degrees)
%  glat = geodetic latitude of observer (north +, degrees)
%  ht   = height of observer (meters)

% output

%  ra  = apparent geocentric or topocentric right ascension,
%        referred to true equator and equinox of date (hours)
%  dec = apparent geocentric or topocentric declination,
%        referred to true equator and equinox of date (degrees)
%  dis = true distance from earth to planet (astronomical units)
```

This software collection includes a script called `demo_aplanet2` which demonstrates how to use this function. The user interaction with this script is identical to the interaction with the `demo_aplanet1` script described above.

apstar1.m – apparent coordinates of a star - JPL ephemeris

This MATLAB function determines the apparent geocentric and topocentric coordinates of a star. The source ephemeris for this routine is a JPL binary file.

This software suite includes scripts that demonstrate how to calculate the apparent topocentric and geocentric coordinates of a star. The first script is called `demo_astar1` and it uses a JPL binary ephemeris file as its source ephemeris. The second script is called `demo_astar2` and it uses SLP96 as its source ephemeris. These applications use several functions ported to MATLAB from the Fortran versions of the NOVAS (Naval Observatory Vector Astrometry Subroutines) source code that was developed at the United States Naval Observatory.

This MATLAB script includes the following types of coordinate corrections:

(1) precession

(2) nutation

(3) aberration

(4) gravitational deflection of light

(5) proper motion

(6) parallax

(7) radial velocity

Please note that this program does not include a correction for atmospheric refraction. For extragalatic objects the proper motions, parallax and radial velocity should all be set to 0.

An excellent discussion about these corrections 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.

The processing steps used in this program are as follows:

(1) input a calendar date on the UT1 time scale

(2) compute the UT1 Julian date

(3) compute the TDT Julian date

(4) compute the TDB Julian date

(5) read the star's coordinates, proper motions, parallax and radial velocity

(6) define type of coordinates

(7) calculate apparent coordinates

The syntax and arguments for the MATLAB function that actually performs most of the calculations are as follows:

```
function [ra, dec] = apstar1 (tjd, ujd, n, topo, glon, glat, ht, ...
                             ram, decm, pmra, pmdec, parlax, radvel)

% this subroutine computes the geocentric or topocentric apparent place
% of a star, given its mean place, proper motion, parallax, and radial
% velocity for j2000.0.

% input

%  tjd    = tdt julian date for apparent place
%  ujd    = ut1 julian date for apparent topocentric place
%  n      = body identification number for the earth
%  topo   = type of apparent place calculation
%         = 0 ==> geocentric
%         = 1 ==> topocentric
%  ram    = mean right ascension j2000.0 (hours)
%  decm   = mean declination j2000.0 (degrees)
%  pmra   = proper motion in ra (seconds of time/julian century)
%  pmdec  = proper motion in dec (seconds of arc/julian century)
%  parlax = parallax (seconds of arc)
%  radvel = radial velocity (kilometers per second)

% output
%  ra  = apparent geocentric or topocentric right ascension,
%        referred to true equator and equinox of date (hours)
%  dec = apparent geocentric or topocentric declination,
%        referred to true equator and equinox of date (degrees)
```

The `demo_astar` software will prompt you for the calendar date, universal time, observer coordinates, the name of the star data file and the type of ephemeris (topocentric or geocentric).

The following is a typical star data file named `altair.dat`. Do not delete any lines of text or data as the software expects to find exactly 20 lines of information.

```
star name
ALTAIR

J2000 right ascension (hours)
19.8463894440

J2000 declination (degrees)
8.8683416670

J2000 proper motion in right ascension (seconds/Julian century)
3.6290

J2000 proper motion in declination (arcseconds/Julian century)
38.6300

parallax (arcseconds)
0.1981

radial velocity (kilometers/second)
-26.30
```

The following is the syntax of the MATLAB function that reads this data file.

```
function [fid, sname, ram, decm, pmra, pmdec, parlax, radvel] =
                                            readstar(filename)

% read star data file

% required by star*.m

% input

%  filename = name of star data file

% output

%  sname  = star name
%  ram    = J2000 right ascension (hours)
%  decm   = J2000 declination (degrees)
%  pmra   = J2000 proper motion in right ascension
%           (seconds/Julian century)
%  pmdec  = J2000 proper motion in declination
%           (arcseconds/Julian century)
%  parlax = parallax (arcseconds)
%  radvel = radial velocity (kilometers/second)
%  fid    = file id
```

The following is a typical user interaction with the `demo_astar1` (and `demo_astar2`) scripts. It illustrates the topocentric coordinates of the star Altair relative to a sea level observer at 40° north latitude and 105° west longitude, at 0 hours UT.

```
demo_astar1 - apparent coordinates - jplephem ephemeris

please input a UTC calendar date

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1998

please input the universal time
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 0,0,0


please input the star data file name
(be sure to include the file name extension)
? altair.dat

coordinate type menu

 <1> geocentric

 <2> topocentric

please select coordinate type
? 2

please input the geographic latitude of the ground site
(-90 <= degrees <= +90, 0 <= minutes <= 60, 0 <= seconds <= 60)
(north latitude is positive, south latitude is negative)
? 40,0,0

please input the geographic longitude of the ground site
(0 <= degrees <= 360, 0 <= minutes <= 60, 0 <= seconds <= 60)
(east longitude is positive, west longitude is negative)
? -105,0,0

please input the altitude of the ground site (meters)
(positive above sea level, negative below sea level)
? 0
```

Here's the program output computed by this MATLAB script.

```
 apparent topocentric coordinates

ALTAIR

UTC calendar date        01-Jan-1998
```

```
universal time           00:00:00

UTC julian date          2450814.5000

observer latitude        -105d 00m 00.00s

observer east longitude  +40d 00m 00.00s

observer altitude              0.0000 meters

right ascension          +19h 50m 39.1674s

declination              +08d 51m 46.09s
```

apstar2.m – apparent coordinates of a star - SLP96 ephemeris

This MATLAB function determines the apparent geocentric and topocentric coordinates of a star. The source ephemeris for this routine is the SLP96 binary file `slp96.bin`.

The syntax for this MATLAB function is

```
function [ra, dec] = apstar2 (tjd, ujd, n, topo, glon, glat, ht, ...
                              ram, decm, pmra, pmdec, parlax, radvel)

% this subroutine computes the geocentric or topocentric apparent place
% of a star, given its mean place, proper motion, parallax, and radial
% velocity for j2000.0.  the coordinates of the earth are determined
% using slp96.

% input

%  tjd    = tdt julian date for apparent place
%  ujd    = ut1 julian date for apparent topocentric place
%  n      = body identification number for the earth
%  topo   = type of apparent place calculation
%         = 0 ==> geocentric
%         = 1 ==> topocentric
%  ram    = mean right ascension j2000.0 (hours)
%  decm   = mean declination j2000.0 (degrees)
%  pmra   = proper motion in ra (seconds of time/julian century)
%  pmdec  = proper motion in dec (seconds of arc/julian century)
%  parlax = parallax (seconds of arc)
%  radvel = radial velocity (kilometers per second)

% output

%  ra  = apparent geocentric or topocentric right ascension,
%        referred to true equator and equinox of date (hours)
%  dec = apparent geocentric or topocentric declination,
%        referred to true equator and equinox of date (degrees)
```

The prior discussion about the `demo_apstar1` script also applies to the demonstration `demo_apstar2` script included with this software suite.

BINARY EPHEMERIS FILES

This section summarizes information about the binary ephemeris files used by many of these MATLAB functions and scripts. This summary includes the name of the disk file and the approximate size of the disk files for each ephemeris along with the valid date range.

JPL DE405 ephemeris

file name – de405_1959_2019.bin
file size – 5.5 MB
valid dates – 1959-2019

file name – de405_1999_2060.bin
file size – 11 MB
valid dates – 1999-2060

file name – de405_2000_2030.bin
file size – 2 MB
valid dates – 2000 – 2020

JPL DE421 ephemeris

file name – de421.bin
file size – 14 MB
valid dates – 1/1/1900 – 12/31/2049

IMMCE SLP96 ephemeris

file name – slp96.bin
file size – 13 MB
valid dates – 12/24/1999 – 1/2/2050

IMMCE INPOP06C ephemeris

file name – inpop06c.bin
file size – 13 MB
valid dates – J2000 +/– 100 years

These files can be downloaded from the Celestial and Orbital Mechanics website.