15. Other Lunar, Solar and Planetary Ephemeris Algorithms

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 on the UT1 time scale.

Each MATLAB 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.

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

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

sun.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] = sun (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

$\lambda^m =$ coordinate mean value

$\alpha = iJ + jS + kP$

$J, S, P =$ mean longitudes of Jupiter, Saturn and Pluto

$i, j, k =$ integer constants

$A, B =$ coefficients of periodic term

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)
```

sun1.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

$$\alpha_s = \arctan\left(\cos\lambda_s, \sin\varepsilon\sin\lambda_s\right)$$
$$\delta_s = \arcsin\left(\sin\varepsilon\sin\lambda_s\right)$$

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] = sun1 (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 following is the syntax for this MATLAB function.

```
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)
```

A MATLAB script called `demoelp.m` which demonstrates how to interact with this ephemeris function is included with this software. The following is a typical user interaction with this script.

```
                demoelp

   < orbital elements of the moon >


please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 1,1,1999


please input the simulation period (days)
? 360
```

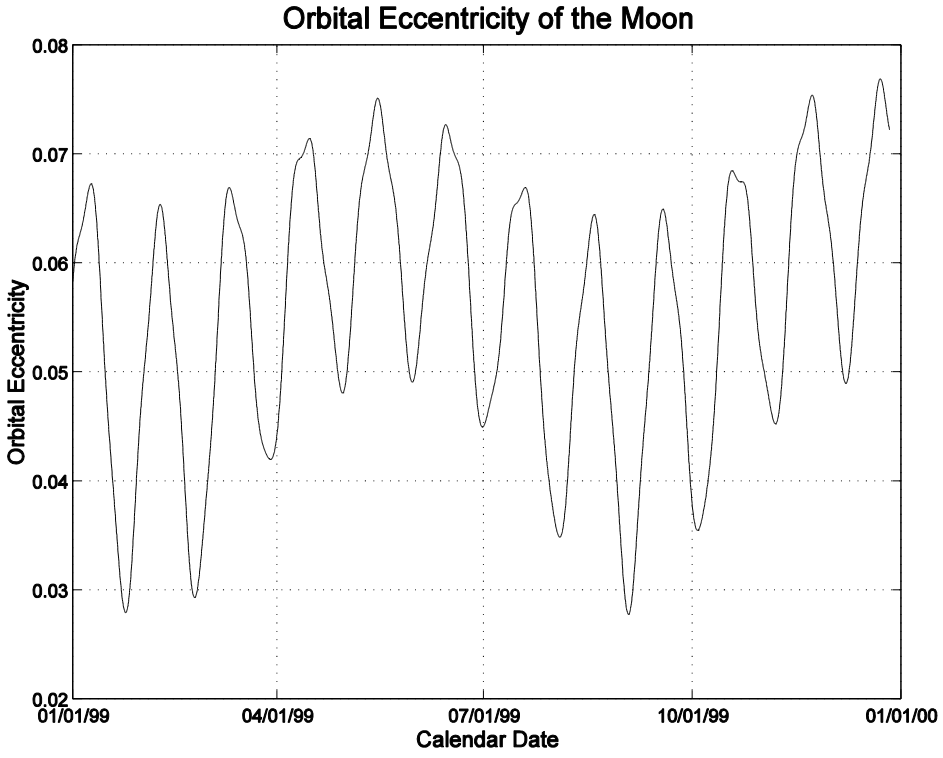
```
please input the graphics step size (days)
? .1

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

? 2
```

The following is the graphics display for this example.



planet1.m – mean ecliptic and equinox of data planetary ephemeredes

This MATLAB 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] = planet1(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)
```

planet2.m – mean ecliptic and equinox of J2000 planetary ephemerides

This MATLAB 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] = planet2(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)
```

seleno.m – selenographic coordinate transformation

This MATLB function computes a transformation matrix that can be used to convert vectors in the ECI mean of date coordinates to selenographic mean of date or true of date coordinate system. The transformation is based on the algorithms given in Section 4.8 of the classic text *Methods of Orbit Determination*.

The syntax of this MATLAB function is

```
function a = seleno(jd, it)

% transformation matrix from eci mean of date to
% selenographic mean of date or true of date

% input

%  jd = julian date
%  it = transformation flag
%     = 0 transform to mean selenographic
%    <> 0 transform to true selenographic
%         (include physical librations)

% output

%  a(3, 3) = transformation matrix
```

A MATLAB script that demonstrates how to use this function is included in the *Celestial Computing with MATLAB* software suite. The script is called `demosen.m` and it graphically displays the topocentric coordinates of the Sun relative to a selenographic location on the Moon.