A MATLAB Script for Time and Coordinate Calculations

This document is the user's manual for an interactive MATLAB script called `csystems.m` which can be used to perform a variety of time and coordinate calculations. This script performs the following types of time calculations;

- convert UTC calendar date and time to UTC Julian date
- convert UTC Julian date to UTC calendar date and time
- Greenwich apparent sidereal time
- convert Universal Coordinated Time (UTC) to Terrestrial Time (TT)
- convert Universal Coordinated Time (UTC) to Barycentric Dynamical Time (TDB)
- convert Barycentric Dynamical Time (TDB) to Universal Coordinated Time (UTC)

The `csystems` script can also be used to perform the following types of coordinate calculations;

- convert geodetic coordinates to eci position vector
- convert eci state vector to ecf state vector
- convert eci state vector to classical orbital elements
- convert classical orbital elements to eci state vector
- convert flight path coordinates to eci state vector
- convert eci state vector to relative flight path coordinates
- convert classical orbital elements to modified equinoctial elements
- convert modified equinoctial elements to classical orbital elements
- convert osculating orbital elements to mean elements
- convert eci state vector to Two Line Elements (TLE)

Interacting with the script

The `csystems` script begins by displaying the following "type of calculation" menu. This menu is displayed prior to the first calculation and after each subsequent calculation is completed.

```
TIME AND COORDINATE CALCULATIONS
--------------------------------


TYPE OF CALCULATION MENU

 <1> time

 <2> coordinates

 <3> quit

selection?
```

If the user selects the "time" type of calculation, the script will display the following menu.

```
TIME CALCULATION MENU

 <1> convert UTC calendar date and time to UTC Julian date

 <2> convert UTC Julian date to UTC calendar date and time

 <3> Greenwich apparent sidereal time

 <4> convert Universal Coordinated Time (UTC) to Terrestrial Time (TT)

 <5> convert Universal Coordinated Time (UTC) to Barycentric Dynamical Time (TDB)

 <6> convert Barycentric Dynamical Time (TDB) to Universal Coordinated Time (UTC)

selection?
```

For time calculations, the script will also display the following "data source" menu. This menu allows the user to either interactively input the data required for a time calculation or use data previously calculated (internal) by the script.

```
DATA SOURCE MENU

 <1> user input

 <2> internal data

selection?
```

When using the "internal data" menu option, the user must ensure that the information needed for a particular calculation in available within the script. For example, to compute Greenwich apparent sidereal time using the "internal data" option, the user must first compute a UTC Julian date.

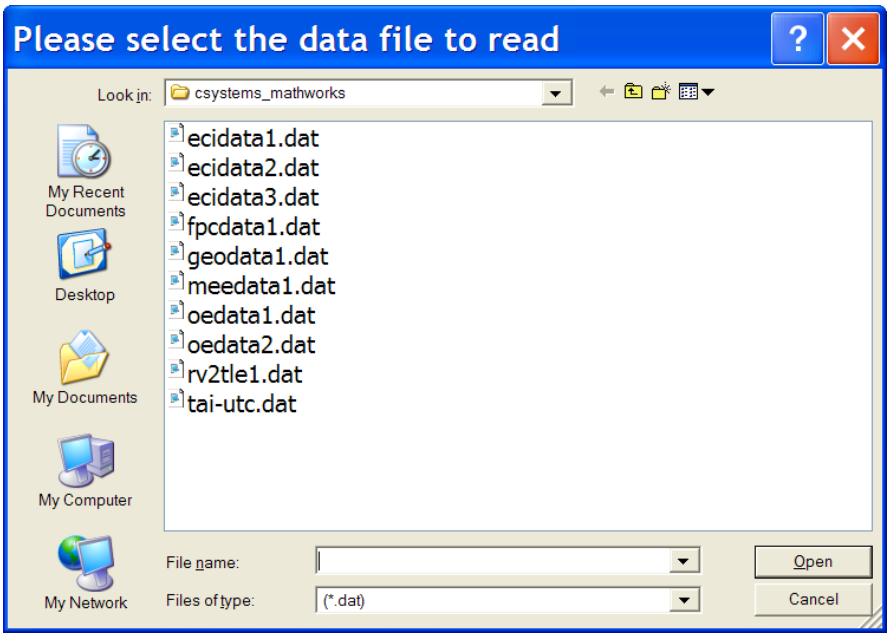
If the user selects the "coordinate" type of calculation, the script will display the following menu.

```
COORDINATE CALCULATION MENU

 <1>  convert geodetic coordinates to eci position vector

 <2>  convert eci state vector to ecf state vector

 <3>  convert eci state vector to classical orbital elements

 <4>  convert classical orbital elements to eci state vector

 <5>  convert flight path coordinates to eci state vector

 <6>  convert eci state vector to relative flight path coordinates

 <7>  convert classical orbital elements to modified equinoctial elements

 <8>  convert modified equinoctial elements to classical orbital elements

 <9>  convert osculating orbital elements to mean elements

 <10> convert eci state vector to Two Line Elements (TLE)

selection?
```

For coordinate calculations, the script will also display the following "data source" menu. This menu allows the user to either interactively input the data required for a time calculation, read the required data from a simple text file, or use data previously calculated (internal) by the script.

```
DATA SOURCE MENU

  <1> user input

  <2> data file

  <3> internal data

selection?
```

When the user selects the "data file" input option for coordinate calculations, the software will display the following interactive window requesting the name of the data file to use.



The file type defaults to names with a `*.dat` filename extension. However, you can select any `csystems` compatible ASCII data file by selecting the Files of type: field or by typing the name of the file directly in the File name: field.

Typical data files

The `csystems` MATLAB script can be "data-driven" by a simple text file created by the user. Please see Appendix C for several examples of compatible data files. This script option only applies to coordinate calculations.

The software also evaluates a simple MATLAB function named `om_constants.m`. This function allows the user to define fundamental utility and astrodynamic data values for the simulation. The following illustrates the contents of a typical function. The data items are self-explanatory. However, please note the proper units for each data value.

```
function om_constants

% astrodynamic and utility constants

% Orbital Mechanics with MATLAB

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

global dtr rtd mu mmu smu omega req flat j2 aunit

dtr = pi / 180.0;

rtd = 180.0 / pi;

% earth gravitational constant (km**3/sec**2)

mu = 398600.436233;

% moon gravitational constant (km**3/sec**2)

mmu = 4902.800076;

% sun gravitational constant (km**3/sec**2)

smu = 132712440040.944;

% earth inertial rotation rate (radians/second)

omega = 7.292115486e-5;

% earth equatorial radius (kilometers)

req = 6378.1363;

% earth flattening factor (non-dimensional)

flat = 1.0 / 298.257;

% earth oblateness gravity coefficient (non-dimensional)

j2 = 0.00108263;

% astronomical unit (kilometers)

aunit = 149597870.691;
```

Script examples

This section provided the `csystems` script output for four typical calculations. The first example computes the Greenwich apparent sidereal time using information provided by the user. The second example converts classical orbital elements, also provided by the user, to the corresponding ECI state vector (position and velocity vectors). The third example converts an ECI state vector to classical orbital elements for an elliptical orbit. The final example converts an ECI state vector to classical orbital elements for a hyperbolic orbit. The user inputs for each example are in bold courier font.

Greenwich apparent sidereal time

```
TIME AND COORDINATE CALCULATIONS
--------------------------------


TYPE OF CALCULATION MENU

 <1> time

 <2> coordinates

 <3> quit

selection? 1


TIME CALCULATION MENU

 <1> convert UTC calendar date and time to UTC Julian date

 <2> convert UTC Julian date to UTC calendar date and time

 <3> Greenwich apparent sidereal time

 <4> convert Universal Coordinated Time (UTC) to Terrestrial Time (TT)

 <5> convert Universal Coordinated Time (UTC) to Barycentric Dynamical Time (TDB)

 <6> convert Barycentric Dynamical Time (TDB) to Universal Coordinated Time (UTC)

selection? 3


DATA SOURCE MENU

 <1> user input

 <2> internal data


 selection? 1


Greenwich apparent sidereal time from UTC calendar date and time
----------------------------------------------------------------

please input the calendar date
(1 <= month <= 12, 1 <= day <= 31, year = all digits!)
? 5,7,2013

please input the Universal Coordinated Time (UTC)
(0 <= hours <= 24, 0 <= minutes <= 60, 0 <= seconds <= 60)
? 9,0,0

UTC calendar date         07-May-2013

UTC time                  09:00:00.000

UTC Julian date           2456419.87500000

Greenwich apparent sidereal time    +00h 01m 29.0571s

Greenwich apparent sidereal time      0.37107120 degrees
```

Greenwich apparent sidereal time is the true-of-date right ascension of the Greenwich meridian relative to the Vernal Equinox.

The script requires all four digits of the calendar year and time is input in 24 hour format.

Convert classical orbital elements to ECI state vector

```
 TIME AND COORDINATE CALCULATIONS
 --------------------------------

 TYPE OF CALCULATION MENU

  <1> time

  <2> coordinates

  <3> quit

 selection? 2


 COORDINATE CALCULATION MENU

  <1>  convert geodetic coordinates to eci position vector

  <2>  convert eci state vector to ecf state vector

  <3>  convert eci state vector to classical orbital elements

  <4>  convert classical orbital elements to eci state vector

  <5>  convert flight path coordinates to eci state vector

  <6>  convert eci state vector to relative flight path coordinates

  <7>  convert classical orbital elements to modified equinoctial elements

  <8>  convert modified equinoctial elements to classical orbital elements

  <9>  convert osculating orbital elements to mean elements

  <10> convert eci state vector to Two Line Elements (TLE)

 selection? 4


 DATA SOURCE MENU

  <1> user input

  <2> data file

  <3> internal data

selection? 1


convert classical orbital elements to eci state vector
------------------------------------------------------

please input the semimajor axis (kilometers)
(semimajor axis > 0)
? 8000

please input the orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
? .015

please input the orbital inclination (degrees)
(0 <= inclination <= 180)
? 28.5

please input the argument of perigee (degrees)
(0 <= argument of perigee <= 360)
? 100
```

```
please input the right ascension of the ascending node (degrees)
(0 <= raan <= 360)
? 240

please input the true anomaly (degrees)
(0 <= true anomaly <= 360)
? 45

eci state vector

        rx (km)                 ry (km)                 rz (km)                rmag (km)
 +6.69635065361484e+003  +3.61976006550616e+003  +2.16602932328762e+003  +7.91425663201181e+003

        vx (kps)                vy (kps)                vz (kps)               vmag (kps)
 -2.33842950782795e+000  +6.14606682466057e+000  -2.76808196379309e+000  +7.13475071285258e+000


classical orbital elements

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00000000000000e+003  +1.50000000000000e-002  +2.85000000000000e+001  +1.00000000000000e+002

      raan (deg)          true anomaly (deg)         arglat (deg)            period (hrs)
 +2.40000000000000e+002  +4.50000000000000e+001  +1.45000000000000e+002  +1.97807822980718e+000
```

The following is a brief description of the data provided by this script option.

sma (km) = semimajor axis in kilometers

eccentricity = orbital eccentricity (non-dimensional)

inclination (deg) = orbital inclination in degrees

argper (deg) = argument of periapsis in degrees

raan (deg) = right ascension of the ascending node in degrees

true anomaly (deg) = true anomaly in degrees

arglat (deg) = argument of latitude in degrees. The argument of latitude is the sum of true anomaly and argument of perigee.

period (hrs) = orbital period in hours

rx (km) = x-component of the spacecraft's position vector in kilometers

ry (km) = y-component of the spacecraft's position vector in kilometers

rz (km) = z-component of the spacecraft's position vector in kilometers

rmag (km) = scalar magnitude of the spacecraft's position vector in kilometers

vx (kps) = x-component of the spacecraft's velocity vector in kilometers per second

vy (kps) = y-component of the spacecraft's velocity vector in kilometers per second

vz (ksp) = z-component of the spacecraft's velocity vector in kilometers per second

vmag (kps) = scalar magnitude of the spacecraft's velocity vector in kilometers per second

Convert ECI state vector to classical orbital elements – elliptical orbit (`ecidata1.dat` *data file)*

```
TIME AND COORDINATE CALCULATIONS
 --------------------------------
```

```
 TYPE OF CALCULATION MENU

  <1> time

  <2> coordinates

  <3> quit

 selection? 2


 COORDINATE CALCULATION MENU

  <1>  convert geodetic coordinates to eci position vector

  <2>  convert eci state vector to ecf state vector

  <3>  convert eci state vector to classical orbital elements

  <4>  convert classical orbital elements to eci state vector

  <5>  convert flight path coordinates to eci state vector

  <6>  convert eci state vector to relative flight path coordinates

  <7>  convert classical orbital elements to modified equinoctial elements

  <8>  convert modified equinoctial elements to classical orbital elements

  <9>  convert osculating orbital elements to mean elements

  <10> convert eci state vector to Two Line Elements (TLE)

 selection? 3


 DATA SOURCE MENU

  <1> user input

  <2> data file

  <3> internal data


 selection? 2


convert eci state vector to classical orbital elements
------------------------------------------------------

UTC calendar date         25-Apr-2013

UTC time                  10:20:30.000

Greenwich apparent sidereal time    8.72340317 degrees


eci state vector

        rx (km)                 ry (km)                 rz (km)                 rmag (km)
 +3.87156734351188e+003  +6.36521709672617e+003  -2.67028756008413e+003  +7.91425663201180e+003

        vx (kps)                vy (kps)                vz (kps)                vmag (kps)
 -5.20544460471574e+000  +4.25847877360187e+000  +2.38188428278297e+000  +7.13475071285258e+000


classical orbital elements

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 +8.00000000000001e+003  +1.50000000000014e-002  +2.85000000000000e+001  +2.70000000000002e+002

       raan (deg)          true anomaly (deg)         arglat (deg)            period (hrs)
 +1.00000000000000e+002  +4.49999999999981e+001  +3.15000000000000e+002  +1.97807822980718e+000
```

```
       lan (deg)          energy (km^2/sec^2)          fpa (deg)             rasc (deg)
+9.12765968255329e+001  -2.49125272645625e+001  +6.01313310679043e-001  +5.86904408962078e+001

   declination (deg)        latitude (deg)        east longitude (deg)       altitude (km)
-1.97186805653134e+001  -1.98173080798631e+001  +4.99670377217407e+001  +1.53856281310617e+003

     r-perigee (km)          r-apogee (km)          v-perigee (kps)         v-apogee (kps)
+7.88000000000000e+003  +8.12000000000002e+003  +7.16537290587628e+000  +6.95358848501293e+000

     h-perigee (km)          h-apogee (km)          lat-perigee (deg)       lat-apogee (deg)
+1.50674668436851e+003  +1.74674608113135e+003  -2.86306704370729e+001  +2.86267971998004e+001
```

The following is a brief description of the additional data provided by this script option.

lan (deg) = east longitude of the ascending node in degrees

energy (km^2/sec^2) = specific (per unit mass) orbital energy in km^2/sec^2

fpa (deg) = inertial flight path angle in degrees

rasc (deg) = right ascension of the spacecraft in degrees

declination (deg) = geocentric declination of the subpoint in degrees

latitude (deg) = geodetic latitude of the subpoint in degrees

east longitude (deg) = east longitude of the subpoint in degrees

altitude (km) = geodetic altitude in kilometers

r-perigee (km) = radius of perigee in kilometers

r-apogee (km) = radius of apogee in kilometers

v-perigee (kps) = perigee velocity in kilometers/second

v-apogee (kps) = apogee velocity in kilometers/second

h-perigee (km) = geodetic altitude of perigee in kilometers

h-apogee (km) = geodetic altitude of apogee in kilometers

lat-perigee (deg) = geodetic latitude of perigee in degrees

lat-apogee (deg) = geodetic latitude of apogee in degrees

Convert ECI state vector to classical orbital elements – hyperbolic orbit (`ecidata3.dat` *data file)*

```
TIME AND COORDINATE CALCULATIONS
 --------------------------------


 TYPE OF CALCULATION MENU

  <1> time

  <2> coordinates

  <3> quit

 selection? 2


 COORDINATE CALCULATION MENU

  <1>  convert geodetic coordinates to eci position vector

  <2>  convert eci state vector to ecf state vector
```

```
  <3>  convert eci state vector to classical orbital elements

  <4>  convert classical orbital elements to eci state vector

  <5>  convert flight path coordinates to eci state vector

  <6>  convert eci state vector to relative flight path coordinates

  <7>  convert classical orbital elements to modified equinoctial elements

  <8>  convert modified equinoctial elements to classical orbital elements

  <9>  convert osculating orbital elements to mean elements

  <10> convert eci state vector to Two Line Elements (TLE)

 selection? 3


 DATA SOURCE MENU

  <1> user input

  <2> data file

  <3> internal data


 selection? 2


convert eci state vector to classical orbital elements
------------------------------------------------------

UTC calendar date         01-May-2013

UTC time                  10:25:45.000

Greenwich apparent sidereal time   15.95338081 degrees


eci state vector

        rx (km)                 ry (km)                 rz (km)                rmag (km)
 -4.78649275664045e+003  -3.85286985447614e+003  -2.30735522767897e+003  +6.56346000000000e+003

        vx (kps)                vy (kps)                vz (kps)               vmag (kps)
 +2.99280761783281e+000  -9.54679394718429e+000  +5.53554782769536e+000  +1.14341795446834e+001


classical orbital elements

        sma (km)              eccentricity          inclination (deg)         argper (deg)
 -4.29526257862447e+004  +1.15042827656182e+000  +3.97389482427168e+001  +3.12784284311210e+002

       raan (deg)          true anomaly (deg)         arglat (deg)
 +2.45683986407545e+002  +1.38556045691745e+001  +3.26639888880384e+002

        lan (deg)          energy (km^2/sec^2)           fpa (deg)              rasc (deg)
 +2.29730605593558e+002  +4.64000080247305e+000  +7.41478466450011e+000  +2.18832199551264e+002

    declination (deg)        latitude (deg)        east longitude (deg)       altitude (km)
 -2.05818777000989e+001  -2.07051993025675e+001  +2.02878818737277e+002  +1.87977722860927e+002

      r-perigee (km)         v-perigee (kps)          h-perigee (km)        lat-perigee (deg)
 +6.46128947082941e+003  +1.15178592414681e+001  +8.78788448716468e+001  -2.81390339322261e+001

     C3 (km^2/sec^2)         rasc-asy (deg)           decl-asy (deg)
 +9.28000160494607e+000  +3.52589997116184e+002  +3.85000004270181e+001

    ta-infinity (deg)       v-infinity (kps)
 +1.50370616120653e+002  +3.04630950577023e+000
```

The following is a brief description of the additional data provided by this script option. This data is unique to hyperbolic orbits. The orbital period and apogee conditions do not exist for hyperbolic orbits and are not displayed.

C3 (km^2/sec^2) = twice the specific (per unit mass) orbital energy in km^2/sec^2

rasc-asy (deg) = right ascension of the asymptote in degrees

decl-asy (deg) = declination of the asymptote in degrees

ta-infinity (deg) = true anomaly at infinity in degrees

v-infinity (kps) = velocity at infinity in kilometers/second

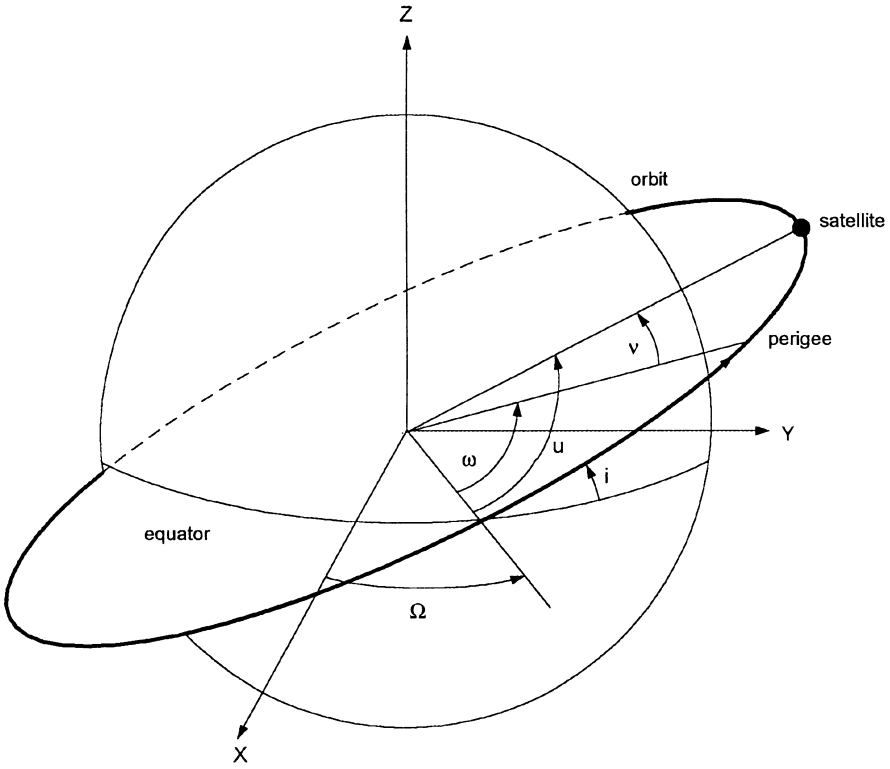
Appendix A

Algorithms for Computing Aerospace Coordinates

This appendix summarizes the defining equations for several coordinate calculation algorithms implemented in the `csystems` MATLAB script.

CLASSICAL ORBITAL ELEMENTS

This section describes MATLAB functions that can be used to convert between inertial state vectors and classical orbital elements. The following diagram illustrates the geometry of these orbital elements.



where

a = semimajor axis
e = orbital eccentricity
i = orbital inclination
ω = argument of perigee
Ω = right ascension of the ascending node
θ = true anomaly

The semimajor axis defines the size of the orbit and the orbital eccentricity defines the shape of the orbit. The angular orbital elements are defined with respect to a fundamental x-axis, the vernal equinox, and a fundamental plane, the equator. The z-axis of this system is collinear with the spin axis of the Earth, and the y-axis completes a right-handed coordinate system.

The orbital inclination is the angle between the equatorial plane and the orbit plane. Satellite orbits with inclinations between 0 and 90 degrees are called *direct* orbits and satellites with inclinations greater than 90 and less than 180 degrees are called *retrograde* orbits. The right ascension of the ascending node (RAAN) is the angle measured from the x-axis (vernal equinox) eastward along the equator to the ascending node. The argument is the angle from the ascending node, measured along the orbit plane in the direction of increasing true anomaly, to the argument of perigee. The true anomaly is the angle from the argument of perigee, measured along the orbit plane in the direction of motion, to the satellite's location. Also shown is the argument of latitude, *u*, which is the angle from the ascending node, measured in the orbit plane, to the satellite's location in the orbit. It is equal to $u = \nu + \omega$.

The orbital eccentricity is an indication of the type of orbit. For values of $0 \le e < 1$, the orbit is circular or elliptic. The orbit is parabolic when $e = 1$ and the orbit is hyperbolic if $e > 1$

The semimajor axis is calculated using the following expression:

$$a = \frac{1}{\frac{2}{r} - \frac{v^2}{\mu}}$$

where $r = |\mathbf{r}| = \sqrt{r_x^2 + r_y^2 + r_z^2}$ and $v = |\mathbf{v}| = \sqrt{v_x^2 + v_y^2 + v_z^2}$. The angular orbital elements are calculated from modified equinoctial orbital elements which are calculated from the ECI position and velocity vectors.

The scalar orbital eccentricity is determined from *h* and *k* as follows:

$$e = \sqrt{h^2 + k^2}$$

The orbital inclination is determined from *p* and *q* using the following expression

$$i = 2\tan^{-1}\left(\sqrt{p^2 + q^2}\right)$$

For values of inclination greater than a small value ε, the right ascension of the ascending node (RAAN) is given by

$$\Omega = \tan^{-1}(p, q)$$

Otherwise, the orbit is equatorial and there is no RAAN. If the value of orbital eccentricity is greater than ε, the argument of perigee is determined from

$$\omega = \tan^{-1}(h, k) - \Omega$$

Otherwise, the orbit is circular and there is no argument of perigee. In the MATLAB code for these calculations, $\varepsilon = 10^{-8}$.

Finally, the true anomaly is found from the expression $\theta = \lambda - \Omega - \omega$.

In these equations, all two argument inverse tangent calculations use the four quadrant MATLAB function `atan3.m` included with this software suite.

The right ascension of a satellite is determined from the x and y components of the ECI *unit* position vector as follows:

$$\alpha = \tan^{-1}\left(r_{y_{eci}}, r_{x_{eci}}\right)$$

The subpoint of a satellite is the point on the Earth's surface directly below the satellite. The locus of subpoints describes the ground track of a satellite. The east longitude of the subpoint of a satellite can be determined from the x and y components of the ECF *unit* position vector as follows:

$$\lambda = \tan^{-1}\left(r_{y_{ecf}}, r_{x_{ecf}}\right)$$

These two position components can be determined from the x and y components of the ECI position vector and the Greenwich sidereal time θ_g as follows:

$$r_{x_{ecf}} = r_{x_{eci}} \cos\theta_g + r_{y_{eci}} \sin\theta_g$$
$$r_{y_{ecf}} = r_{y_{eci}} \cos\theta_g - r_{x_{eci}} \sin\theta_g$$

The geocentric declination of the satellite subpoint is

$$\delta = \sin^{-1}\left(\frac{r_z}{r}\right)$$

The geocentric declination of perigee and apogee are given by the next two equations:

$$\delta_p = \sin^{-1}\left(\sin i \sin\omega\right)$$
$$\delta_a = \sin^{-1}\left[\sin i \sin\left(180 + \omega\right)\right]$$

If the orbit is elliptic $\left(a > 0\right)$, the geocentric radius of perigee and apogee are given by the next two equations:

$$r_p = a\left(1 - e\right)$$
$$r_a = a\left(1 + e\right)$$

The geodetic latitude and altitude of perigee and apogee can be determined from these geocentric coordinates using the `geodet1.m` conversion function.

The perigee and apogee speed can be determined from

$$V_p = \sqrt{2\mu \frac{r_a}{r_p\left(r_p + r_a\right)}} \qquad V_a = \sqrt{2\mu \frac{r_p}{r_a\left(r_p + r_a\right)}}$$

If the orbit is hyperbolic $(a < 0)$, there is no apogee and the perigee radius is equal to

$$r_p = a(1+e)$$

and the perigee speed is equal to

$$V_p = \sqrt{\frac{2\mu}{r_p} + \frac{\mu}{a}}$$

The flight path angle is the angle between the velocity vector and the local horizontal or tangent plane at the satellite's location. It can be calculated with the following expression:

$$\gamma = \sin^{-1}\left(\frac{\mathbf{r} \bullet \mathbf{v}}{|\mathbf{r} \bullet \mathbf{v}|}\right)$$

The specific orbital energy, or the energy per unit mass is determined from this next expression:

$$E = \frac{v^2}{2} - \frac{\mu}{r} = -\frac{\mu}{2a}$$

The orbital energy is the sum of a satellite's kinetic and potential energy. Notice that it depends only on the satellite's semimajor axis and the gravitational constant of the central body.

Hyperbolic orbit

The asymptote unit vector of a hyperbolic orbit is given by

$$\hat{\mathbf{s}} = \begin{Bmatrix} \cos\delta_\infty \cos\alpha_\infty \\ \cos\delta_\infty \sin\alpha_\infty \\ \sin\delta_\infty \end{Bmatrix}$$

In this expression, α_∞ is the right ascension of the asymptote (RLA), and δ_∞ is the declination of the asymptote (DLA).

The asymptote unit vector at any trajectory time can be computed from

$$\hat{\mathbf{s}} = \frac{1}{1 + C_3 \dfrac{h^2}{\mu^2}} \left\{ \left(\frac{\sqrt{C_3}}{\mu} \right) \mathbf{h} \times \mathbf{e} - \mathbf{e} \right\} = \frac{1}{1 + C_3 \dfrac{p}{\mu}} \left\{ \left(\frac{\sqrt{C_3}}{\mu} \right) \mathbf{h} \times \mathbf{e} - \mathbf{e} \right\}$$

where **h** and **e** are the angular momentum and orbital eccentricity vectors, respectively. In the second expression, *p* is the semiparameter of the hyperbolic orbit which can be computed from

$$p = a\left(1 - e^2\right)$$

The angular momentum and eccentricity vectors are computed using the following equations;

$$\mathbf{h} = \mathbf{r} \times \mathbf{v}$$

$$\mathbf{e} = \frac{\mathbf{v} \times \mathbf{h}}{\mu} - \frac{\mathbf{r}}{r} = \frac{1}{\mu}\left[\left(v^2 - \frac{\mu}{r}\right)\mathbf{r} - \left(\mathbf{r} \bullet \mathbf{v}\right)\mathbf{v}\right]$$

C3 is the "twice specific" orbital energy which is determined from the position **r** and velocity **v** vectors according to

$$C_3 = |\mathbf{v}|^2 - \frac{2\mu}{|\mathbf{r}|}$$

The right ascension and declination of the asymptote can be computed from components of the unit asymptote vector according to

$$\alpha_\infty = \tan^{-1}\left(s_x, s_y\right)$$

$$\delta_\infty = \sin^{-1}\left(s_z\right)$$

The right ascension and declination will be with respect to the coordinate system of the input state vector (EME2000 for example).

ECI STATE VECTOR

The rectangular components of the inertial position vector are determined from

$$r_x = r\left[\cos\Omega\cos\left(\omega + \theta\right) - \sin\Omega\cos i\sin\left(\omega + \theta\right)\right]$$

$$r_y = r\left[\sin\Omega\cos\left(\omega + \theta\right) + \cos\Omega\cos i\sin\left(\omega + \theta\right)\right]$$

$$r_z = r\sin i\sin\left(\omega + \theta\right)$$

where $r = \sqrt{r_x^2 + r_y^2 + r_z^2}$.

The components of the inertial velocity vector are computed using the following equations:

$$v_x = -\sqrt{\frac{\mu}{p}}\left[\cos\Omega\left\{\sin\left(\omega + \theta\right) + e\sin\omega\right\} + \sin\Omega\cos i\left\{\cos\left(\omega + \theta\right) + e\cos\omega\right\}\right]$$

$$v_y = -\sqrt{\frac{\mu}{p}}\left[\sin\Omega\left\{\sin\left(\omega + \theta\right) + e\sin\omega\right\} - \cos\Omega\cos i\left\{\cos\left(\omega + \theta\right) + e\cos\omega\right\}\right]$$

$$v_z = \sqrt{\frac{\mu}{p}}\left[\sin i\left\{\cos\left(\omega+\theta\right)+e\cos\omega\right\}\right]$$

where

a = semimajor axis
e = orbital eccentricity
i = orbital inclination
ω = argument of perigee
Ω = right ascension of the ascending node
θ = true anomaly
$p = a\left(1-e^2\right)$

MODIFIED EQUINOCTIAL ORBITAL ELEMENTS

The relationship between modified equinoctial and classical orbital elements is given by

$$p = a\left(1-e^2\right)$$

$$f = e\cos\left(\omega+\Omega\right)$$

$$g = e\sin\left(\omega+\Omega\right)$$

$$h = \tan\left(i/2\right)\cos\Omega$$

$$k = \tan\left(i/2\right)\sin\Omega$$

$$L = \Omega+\omega+\theta$$

where

p = semiparameter
a = semimajor axis
e = orbital eccentricity
i = orbital inclination
ω = argument of perigee
Ω = right ascension of the ascending node
θ = true anomaly
L = true longitude

The relationship between classical and modified equinoctial orbital elements is as follows:

semimajor axis

$$a = \frac{p}{1-f^2-g^2}$$

orbital eccentricity

$$e = \sqrt{f^2 + g^2}$$

orbital inclination

$$i = 2\tan^{-1}\left(\sqrt{h^2 + k^2}\right)$$

argument of periapsis

$$\omega = \tan^{-1}(g/f) - \tan^{-1}(k/h)$$

right ascension of the ascending node

$$\Omega = \tan^{-1}(k/h)$$

true anomaly

$$\theta = L - (\Omega + \omega) = L - \tan^{-1}(g/f)$$

The relationship between ECI position and velocity vectors and modified equinoctial elements is defined by the following series of equations:

position vector

$$\mathbf{r} = \begin{bmatrix} \frac{r}{s^2}\left(\cos L + \alpha^2 \cos L + 2hk \sin L\right) \\ \frac{r}{s^2}\left(\sin L - \alpha^2 \sin L + 2hk \cos L\right) \\ \frac{r}{s^2}\left(h \sin L - k \cos L\right) \end{bmatrix}$$

velocity vector

$$\mathbf{v} = \begin{bmatrix} -\frac{1}{s^2}\sqrt{\frac{\mu}{p}}\left(\sin L + \alpha^2 \sin L - 2hk \cos L + g - 2fhk + \alpha^2 g\right) \\ -\frac{1}{s^2}\sqrt{\frac{\mu}{p}}\left(-\cos L + \alpha^2 \cos L + 2hk \sin L + g - f + 2ghk + \alpha^2 f\right) \\ \frac{2}{s^2}\sqrt{\frac{\mu}{p}}\left(h \cos L + k \sin L + fh + gk\right) \end{bmatrix}$$

where

$$\alpha^2 = h^2 - k^2 \qquad s^2 = 1 + h^2 + k^2$$

$$r = \frac{p}{w} \qquad w = 1 + f\cos L + g \sin L$$

The conversion of ECI position and velocity vectors **r** and **v** to modified equinoctial elements is defined by the following series of equations:

$$p = h^2/\mu$$

$$k = \hat{\mathbf{h}}_x/1+\hat{\mathbf{h}}_z$$

$$h = -\hat{\mathbf{h}}_y/1+\hat{\mathbf{h}}_z$$

where $\hat{\mathbf{h}}_x$ $\hat{\mathbf{h}}_y$, and $\hat{\mathbf{h}}_z$ are the x, y, and z components of the unit angular momentum vector.

$$f = \mathbf{e} \bullet \tilde{\mathbf{f}}$$

$$g = \mathbf{e} \bullet \tilde{\mathbf{g}}$$

The eccentricity vector $\mathbf{e}$ points in the direction of perigee and is computed from

$$\mathbf{e} = \frac{\mathbf{v} \times \mathbf{h}}{\mu} - \frac{\mathbf{r}}{|\mathbf{r}|}$$

$$\tilde{\mathbf{f}} = \hat{\mathbf{f}}/1+k^2+h^2 \qquad \tilde{\mathbf{g}} = \hat{\mathbf{g}}/1+k^2+h^2$$

$$\hat{\mathbf{f}} = \begin{Bmatrix} 1-k^2+h^2 \\ 2kh \\ 2k \end{Bmatrix}$$

$$\hat{\mathbf{g}} = \begin{Bmatrix} 2kh \\ 1+k^2-h^2 \\ 2h \end{Bmatrix}$$

The angular momentum vector is computed from

$$\mathbf{h} = \mathbf{r} \times \mathbf{v}$$

where **r** and **v** are the inertial position and velocity vectors, respectively.

Finally, the true longitude is determined from

$$L = \tan^{-1}\left(\hat{\mathbf{r}}_x + \hat{\mathbf{v}}_y, \hat{\mathbf{r}}_y - \hat{\mathbf{v}}_x\right)$$

where $\hat{\mathbf{r}}_x$ $\hat{\mathbf{r}}_y$, and $\hat{\mathbf{v}}_x$ $\hat{\mathbf{v}}_y$ are the x and y components of the unit position and velocity vectors.

FLIGHT PATH COORDINATES

One form of the relative flight path coordinates can be defined as follows;

$r =$ geocentric radius
$V =$ speed
$\gamma =$ flight path angle
$\delta =$ geocentric declination
$\lambda =$ geographic longitude $(+ \text{ east})$
$\psi =$ flight azimuth $(+ \text{ clockwise from north})$

Please note the sign and direction convention.

The following are several useful equations that summarize the relationships between inertial and relative flight path coordinates.

$$v_r \sin\gamma_r = v_i \sin\gamma_i$$

$$v_r \cos\gamma_r \cos\psi_r = v_i \cos\gamma_i \cos\psi_i$$

$$v_r \cos\gamma_r \sin\psi_r + \omega_e r \cos\delta = v_i \cos\gamma_i \sin\psi_i$$

where the *r* subscript denotes relative coordinates and the *i* subscript inertial coordinates.

The inertial speed can also be computed from the following expression

$$v_i = \sqrt{v^2 + 2v\,r\omega\cos\gamma\sin\psi\cos\delta + r^2\omega^2\cos^2\delta}$$

The inertial flight path angle can be computed from

$$\cos\gamma_i = \sqrt{\frac{v^2\cos^2\gamma + 2vr\omega\cos\gamma\cos\psi\cos\delta + r^2\omega^2\cos^2\delta}{v^2 + 2vr\omega\cos\gamma\cos\psi\cos\delta + r^2\omega^2\cos^2\delta}}$$

The inertial azimuth can be computed from

$$\cos\psi_i = \frac{v\cos\gamma\cos\psi + r\omega\cos\delta}{\sqrt{v^2\cos^2\gamma + 2vr\omega\cos\gamma\cos\psi\cos\delta + r^2\omega^2\cos^2\delta}}$$

where all coordinates on the right-hand-side of these equations are relative to a rotating Earth.

The transformation of an Earth-centered inertial (ECI) position vector $\mathbf{r}_{ECI}$ to an Earth-centered fixed (ECF) position vector $\mathbf{r}_{ECF}$ is given by the following vector-matrix operation

$$\mathbf{r}_{ECF} = [\mathbf{T}]\mathbf{r}_{ECI}$$

where the elements of the transformation matrix $[\mathbf{T}]$ are given by

$$[\mathbf{T}] = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and θ is the Greenwich apparent sidereal time at the moment of interest. Greenwich sidereal time is given by the following expression:

$$\theta = \theta_{g0} + \omega_e t$$

where θ_{g0} is the Greenwich apparent sidereal time at 0 hours UTC, ω_e is the inertial rotation rate of the Earth, and *t* is the elapsed time since 0 hours UTC.

Finally, the flight path coordinates are determined from the following set of equations

$$r = \sqrt{r_{ECF}^2 + r_{ECF_y}^2 + r_{ECF}^2}$$

$$v = \sqrt{v_{ECF}^2 + v_{ECF_y}^2 + v_{ECF_z}^2}$$

$$\lambda = \tan^{-1}\left(r_{ECF_y}, r_{ECF_x}\right)$$

$$\delta = \sin^{-1}\left(\frac{r_{ECF_z}}{|\mathbf{r}_{ECF}|}\right)$$

$$\gamma = \sin^{-1}\left(-\frac{v_{R_z}}{|\mathbf{v}_R|}\right)$$

$$\psi = \tan^{-1}\left[v_{R_y}, v_{R_x}\right]$$

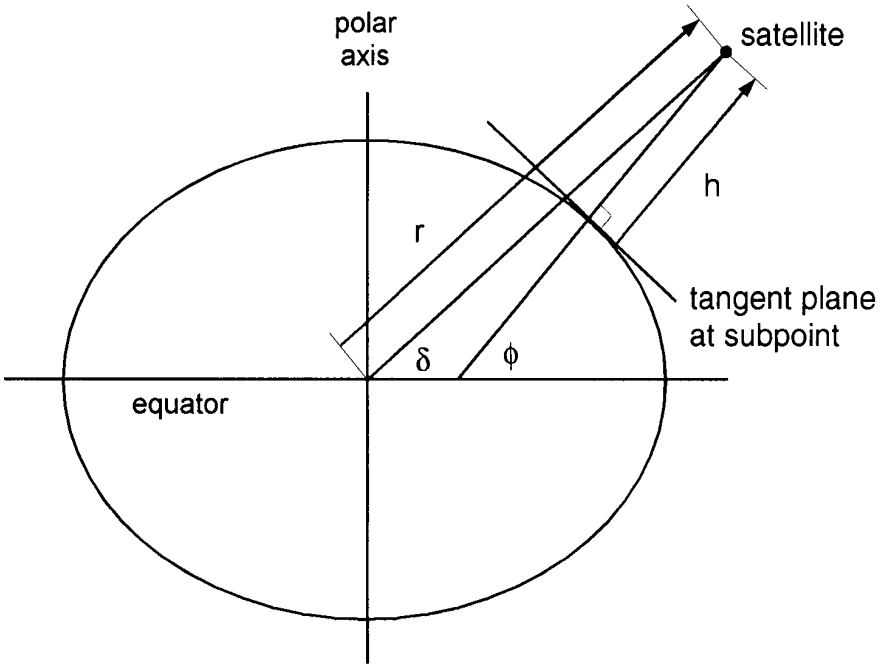
where

$$\mathbf{v}_R = \begin{bmatrix} -\sin\delta\cos\lambda & -\sin\delta\sin\lambda & \cos\delta \\ -\sin\lambda & \cos\lambda & 0 \\ -\cos\delta\cos\lambda & -\cos\delta\sin\lambda & -\sin\delta \end{bmatrix} \mathbf{v}_{ECF}$$

Please note that the two argument inverse tangent calculation is a four quadrant operation.

GEODETIC COORDINATES

The following diagram illustrates the geometric relationship between geocentric and geodetic coordinates.



In this diagram, δ is the geocentric declination, ϕ is the geodetic latitude, *r* is the geocentric distance, and *h* is the geodetic altitude. The exact mathematical relationship between geocentric and geodetic coordinates is given by the following system of two nonlinear equations

$$(c+h)\cos\phi - r\cos\delta = 0$$

$$(s+h)\sin\phi - r\sin\delta = 0$$

where the geodetic constants *c* and *s* are given by

$$c = \frac{r_{eq}}{\sqrt{1-(2f-f^2)\sin^2\phi}} \qquad s = c(1-f)^2$$

and r_{eq} is the Earth equatorial radius and f is the flattening factor for the Earth.

In this MATLAB script, the geodetic latitude is determined from the geocentric declination using the following expression

$$\phi = \delta + \left(\frac{\sin 2\delta}{\rho}\right) f + \left[\left(\frac{1}{\rho^2} - \frac{1}{4\rho}\right)\sin 4\delta\right] f^2$$

which is a series expansion in flattening factor (NASA TN D-7522).

The geodetic altitude is calculated from

$$\hat{h} = (\hat{r} - 1) + \left\{ \left(\frac{1 - \cos 2\delta}{2} \right) f + \left[\left(\frac{1}{4\rho} - \frac{1}{16} \right) (1 - \cos 4\delta) \right] f^2 \right\}$$

In these equations, ρ is the geocentric distance of the satellite, $\hat{h} = h / r_{eq}$ and $\hat{r} = \rho / r_{eq}$.

Appendix B

Time Scales

This appendix is a brief explanation of the time scales used in this MATLAB script.

Coordinated Universal Time, UTC

Coordinated Universal Time (UTC) is the time scale available from broadcast time signals. It is a compromise between the highly stable atomic time and the irregular earth rotation. UTC is the international basis of civil and scientific time.

Terrestrial Time, TT

Terrestrial Time is the time scale that would be kept by an ideal clock on the geoid - approximately, sea level on the surface of the Earth. Since its unit of time is the SI (atomic) second, TT is independent of the variable rotation of the Earth. TT is meant to be a smooth and continuous "coordinate" time scale independent of Earth rotation. In practice TT is derived from International Atomic Time (TAI), a time scale kept by real clocks on the Earth's surface, by the relation **TT = TAI + 32^s.184.** It is the time scale now used for the precise calculation of future astronomical events observable from Earth.

TT = TAI + 32.184 seconds

TT = UTC + (number of leap seconds) + 32.184 seconds

Barycentric Dynamical Time, TDB

Barycentric Dynamical Time is the time scale that would be kept by an ideal clock, free of gravitational fields, co-moving with the solar system barycenter. It is always within 2 milliseconds of TT, the difference caused by relativistic effects. TDB is the time scale now used for investigations of the dynamics of solar system bodies.

TDB = TT + periodic corrections

where typical periodic corrections (USNO Circular 179) are

$$\begin{aligned} TDB = TT &+ 0.001657\sin\left(628.3076T + 6.2401\right) \\ &+ 0.000022\sin\left(575.3385T + 4.2970\right) \\ &+ 0.000014\sin\left(1256.6152T + 6.1969\right) \\ &+ 0.000005\sin\left(606.9777T + 4.0212\right) \\ &+ 0.000005\sin\left(52.9691T + 0.4444\right) \\ &+ 0.000002\sin\left(21.3299T + 5.5431\right) \\ &+ 0.000010T\sin\left(628.3076T + 4.2490\right) + \cdots \end{aligned}$$

In this equation, the coefficients are in seconds, the angular arguments are in radians, and T is the number of Julian centuries of TT from J2000; T = (Julian Date(TT) – 2451545.0) / 36525.

The following is the MATLAB source code for the routine ported from the NOVAS Fortran subroutine. Notice that the NOVAS name was simply `times` and the ported version is named `novas_times` to avoid confusion with the built-in MATLAB function.

```
function [ttjd, secdif] = novas_times (tdbjd)

% this function computes the terrestrial time (tt) julian date
% corresponding to a barycentric dynamical time (tdb) julian date.
% the expression used in this version is a truncated form of a
% longer and more precise series given by fairhead & bretagnon
% (1990) a&a 229, 240. the result is good to about 10 microseconds.

% input

%  tdbjd = tdb julian date

% output

%  ttjd = tt julian date

%  secdif = difference tdbjd - ttjd, in seconds

% ported from NOVAS 3.0

%%%%%%%%%%%%%%%%%%%%%%%

% t0 = tdb julian date of epoch j2000.0 (tt)

t0 = 2451545.0d0;

t = (tdbjd - t0) / 36525.0d0;

% expression given in usno circular 179, eq. 2.6

secdif = 0.001657d0 * sin(628.3076d0 * t + 6.2401d0) ...
    + 0.000022d0 * sin(575.3385d0 * t + 4.2970d0) ...
    + 0.000014d0 * sin(1256.6152d0 * t + 6.1969d0) ...
    + 0.000005d0 * sin(606.9777d0 * t + 4.0212d0) ...
    + 0.000005d0 * sin(52.9691d0 * t + 0.4444d0) ...
    + 0.000002d0 * sin(21.3299d0 * t + 5.5431d0) ...
    + 0.000010d0 * t * sin(628.3076d0 * t + 4.2490d0);

ttjd = tdbjd - secdif / 86400.0d0;
```

The csystems script includes the option to convert a Barycentric Dynamical Time (TDB) to the corresponding Universal Coordinated Time (UTC). The following is the MATLAB source code for a function which iteratively performs this calculation using Brent's root-finding method.

```
function jdutc = tdb2utc (jdtdb)

% convert TDB julian date to UTC julian date

% input

%  jdtdb = TDB julian date

% output
```

```
%  jdutc = UTC julian date
% Orbital Mechanics with MATLAB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global jdsaved
jdsaved = jdtdb;
% convergence tolerance
rtol = 1.0d-8;
% set lower and upper bounds
x1 = jdsaved - 0.1;
x2 = jdsaved + 0.1;
% solve for UTC julian date using Brent's method
[xroot, froot] = brent ('jdfunc', x1, x2, rtol);

jdutc = xroot;
end
```

This function calls the following MATLAB objective function during the calculations.

```
function fx = jdfunc (jdin)
% objective function for tdb2utc
% input
%  jdin = current value for UTC julian date
% output
%  fx = delta julian date
% Orbital Mechanics with MATLAB
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global jdsaved
tai_utc = findleap(jdin);
fx = utc2tdb (jdin) - jdsaved;
end
```

Notice that this function requires the `findleap` function which calculates the number of leap seconds for the current UTC Julian date value. The `jdfunc` function is computing the difference between the

TDB Julian date input by the user and the value computed by the `utc2tdb` MATLAB function. The algorithm has converged whenever this value is less than or equal to the user-defined tolerance `rtol`.

Leap seconds calculation

The difference between International Atomic Time (TAI) and Universal Coordinated Time (UTC) is the number of current leap seconds. International Atomic Time (TAI**,** Temps Atomique International) is a physical time scale with the unit of the SI (System International) second and derived from a statistical timescale based on a large number of atomic clocks. Coordinated Universal Time (UTC) is the time scale available from broadcast time signals. It is a compromise between the highly stable atomic time and the irregular earth rotation. UTC is the international basis of civil and scientific time.

The calculation of leap seconds in this MATLAB script is performed by a function that reads a simple ASCII data file and evaluates the current value of leap seconds. The leap second function must be initialized by including the following statements in the main script.

```
% read leap seconds data file

readleap;
```

The `readleap` MATLAB function reads the contents of the following simple comma-separated-variable (csv) two column data file. The name of this file is `tai-utc.dat`.

```
 2441317.5,  10.0
 2441499.5,  11.0
 2441683.5,  12.0
 2442048.5,  13.0
 2442413.5,  14.0
 2442778.5,  15.0
 2443144.5,  16.0
 2443509.5,  17.0
 2443874.5,  18.0
 2444239.5,  19.0
 2444786.5,  20.0
 2445151.5,  21.0
 2445516.5,  22.0
 2446247.5,  23.0
 2447161.5,  24.0
 2447892.5,  25.0
 2448257.5,  26.0
 2448804.5,  27.0
 2449169.5,  28.0
 2449534.5,  29.0
 2450083.5,  30.0
 2450630.5,  31.0
 2451179.5,  32.0
 2453736.5,  33.0
 2454832.5,  34.0
 2456109.5,  35.0
```

The first column of this data file is the Julian date, on the UTC time scale, at which the leap second became valid. The second column is the leap second value, in seconds.

Note that this data is passed between the leap second MATLAB functions by way of a global statement.

```
global jdateleap leapsec
```

The MATLAB function that actually reads and evaluates the current value of leap seconds has the following syntax and single argument.

```
function leapsecond = findleap(jdate)

% find number of leap seconds for utc julian date

% input

%  jdate = utc julian date

% input via global

%  jdateleap = array of utc julian dates
%  leapsec   = array of leap seconds

% output

%  leapsecond = number of leap seconds
```

Please note that this function does not extrapolate outside the range of dates contained in the data file.

The leap seconds data file should be updated whenever the International Earth Rotation and Reference Systems Service (IERS) announces a new leap second.

Appendix C

Typical Data Files

This appendix describes several typical input data files for the `csystems` MATLAB script. Each data item within an input file is preceded by one or more lines of *annotation* text. Do not delete any of these annotation lines or increase or decrease the number of lines reserved for each comment. However, you may change them to reflect your own explanation. The annotation line also includes the correct metric units and when appropriate, the valid range of the input. In the following summary, the user-defined data items in each file are indicated in bold courier font.

Please note that each data file begins with three and only three lines of information. These three lines can be modified by the user but should not be deleted.

State vector data file

```
*********************************************************************
* state vector (position and velocity) data file - hyperbolic orbit
*********************************************************************

UTC calendar date (month, day, year)
5, 1, 2013

UTC time (hours, minutes, seconds)
10, 25, 45

eci position vector (kilometers)

-4786.49275664045
-3852.86985447614
-2307.35522767897

eci velocity vector (kilometers/second)

+2.99280761783281
-9.54679394718429
+5.53554782769536
```

Classical orbital elements data file

```
*****************************
* orbital elements data file
*****************************

semimajor axis (kilometers)
(semimajor axis > 0)
8000

orbital eccentricity (non-dimensional)
(0 <= eccentricity < 1)
0.015

orbital inclination (degrees)
(0 <= inclination <= 180)
28.5
```

```
argument of perigee (degrees)
(0 <= argument of perigee <= 360)
270.0

right ascension of the ascending node (degrees)
(0 <= RAAN <= 360)
100.0

true anomaly (degrees)
(0 <= true anomaly <= 360)
45.0
```

Flight path coordinates data file

```
**********************************
* flight path coordinates data file
**********************************

UTC calendar date (month, day, year)
4, 25, 2013

UTC time (hours, minutes, seconds)
10, 20, 30

east longitude (degrees)
(0 <= east longitude <= 360)
188.17547382

geocentric declination (degrees)
(-90 <= declination <= +90)
-19.38148629

flight path angle (degrees)
(-90 <= flight path angle <= +90)
-6.2

relative azimuth (degrees)
(0 <= azimuth <= 360)
38.98298719

position magnitude (kilometers)
6497.6909512

velocity magnitude (kilometers/second)
10.71074956
```

Modified equinoctial orbital elements data file

```
************************************************
* modified equinoctial orbital elements data file
************************************************

semiparameter (kilometers)
(|semiparameter| > 0)
7998.2

f equinoctial element (non-dimensional)
0.01477212
```

```
g equinoctial element (non-dimensional)
0.00260472

h equinoctial element (non-dimensional)
-0.04410102

k equinoctial element (non-dimensional)
0.25010931

true longitude (degrees)
(0 <= true longitude <= 360)
55.0
```

Geodetic coordinates data file

```
*******************************
* geodetic coordinates data file
*******************************

UTC calendar date (month, day, year)
5, 1, 2013

UTC time (hours, minutes, seconds)
10, 25, 45

geodetic latitude (degrees)
(-90 <= latitude <= +90)
30.0

east longitude (degrees)
(0 <= longitude <= 360
120.0

geodetic altitude (kilometers)
(altitude > 0)
300.0
```