Package 'asteRisk'

July 5, 2021

Type Package

Title Computation of Satellite Position

Version 1.1.0

Description Provides basic functionalities to calculate the position of satellites given a known state vector. The package includes implementations of the SGP4 and SDP4 simplified perturbation models to propagate orbital state vectors, as well as utilities to read TLE files and convert coordinates between different frames of reference. Several of the functionalities of the package (including the high-precision numerical orbit propagator) require the coefficients and data included in the 'asteRiskData' package, available in a 'drat' repository. To install this data package, run 'install.packages(``asteRiskData'', repos=``https://rafael-ayala.github.io/drat/'')'.
Felix R. Hoots, Ronald L. Roehrich and T.S. Kelso (1988) <https://celestrak.com/NORAD/documentation/spacetrk.pdf>.
David Vallado, Paul Crawford, Richard Hujsak and T.S. Kelso (2012) <doi:10.2514/6.2006-6753>.
Felix R. Hoots, Paul W. Schumacher Jr. and Robert A. Glover (2014) <doi:10.2514/1.9161>.

Acknowledgements The development of this software is supported by the Spanish Ministry of Science and Innovation (grant code PID2019-105471RB-I00) and the Regional Government of Andalusia (grant code P18-RT-1060).

License GPL-3

Imports deSolve, stats

Suggests asteRiskData, knitr, formatR, webshot, BiocStyle, RUnit, BiocGenerics, plotly, lazyeval, dplyr, ggmap, rmarkdown, markdown

Additional_repositories https://rafael-ayala.github.io/drat/

VignetteBuilder knitr

Encoding UTF-8

NeedsCompilation no

Author Rafael Ayala [aut, cre] (<https://orcid.org/0000-0002-9332-4623>),
Daniel Ayala [aut] (<https://orcid.org/0000-0003-2095-1009>),

David Ruiz [aut] (<https://orcid.org/0000-0003-4460-5493>),
Lara Selles Vidal [aut] (<https://orcid.org/0000-0003-2537-6824>)

Maintainer Rafael Ayala <rafael.ayala@oist.jp>

Repository CRAN

Date/Publication 2021-07-05 14:50:02 UTC

R topics documented:

ECEFtoGCRF *Convert coordinates from ECEF to GCRF*

Description

The ECEF (Earth Centered, Earth Fixed) is a non-inertial frame of reference where the origin is placed at the center of mass of Earth, and the frame rotates with respect to the stars to remain fixed with respect to the Earth surface as it rotates.

The GCRF (Geocentric Celestial Reference Frame) frame of reference is an Earth-centered inertial coordinate frame, where the origin is also placed at the center of mass of Earth and the coordinate frame is fixed with respect to the stars (and therefore not fixed with respect to the Earth surface in its rotation). The X-axis is aligned with the mean equinox of Earth at 12:00 Terrestrial Time on the 1st of January, 2000, and the Z-axis is aligned with the Earth´s rotation axis.

This function requires the `asteRiskData` package, which can be installed by running `install.packages('asteRiskData',`

Usage

```
ECEFtoGCRF(position_ECEF, velocity_ECEF, dateTime)
```

Arguments

position_ECEF	Vector with the X, Y and Z components of the position of an object in ECEF frame, in m.
velocity_ECEF	Vector with the X, Y and Z components of the velocity of an object in ECEF frame, in m/s.
dateTime	Date-time string with the date and time in UTC corresponding to the provided position and velocity vectors.

Value

A list with two elements representing the position and velocity of the satellite in the GCRF (Earth-centered non-intertial) frame of reference. Position values are in m, and velocity values are in m/s. Each of the two elements contains three values, corresponding to the X, Y and Z components of position and velocity in this order.

References

https://celestrak.com/columns/v02n01/

Examples

```
if(requireNamespace("asteRiskData", quietly = TRUE)) {
# The following were the position and velocity of satellite MOLNIYA 1-83
# the 25th of June, 2006 at 00:33:43 UTC in the ECEF frame (in m and m/s).

position_ECEF <- c(1.734019e+06, -1.510972e+07, 39.08228)
velocity_ECEF <- c(1468.832, -3962.464, 4007.039)

# Lets convert them into the GCRF frame

coordinates_GCRF <- ECEFtoGCRF(position_ECEF, velocity_ECEF, "2006-06-25 00:33:43")
}
```

ECEFtoLATLON *Convert coordinates from ECEF to geodetic latitude, longitude and altitude*

Description

The ECEF (Earth Centered, Earth Fixed) frame of reference is a non-inertial coordinate frame, where the origin is placed at the center of mass of the Earth and the frame rotates with respect to the stars to remain fixed with respect to the Earth surface as it rotates. This function converts Cartesian coordinates in the ECEF frame to geodetic latitude, longitude and altitude.

Usage

```
ECEFtoLATLON(position_ECEF, degreesOutput=TRUE)
```

Arguments

position_ECEF	Vector with the X, Y and Z components of the position of an object in ECEF frame, in m.
degreesOutput	Logical indicating if the output should be in sexagesimal degrees. If degreesOutput=FALSE, the output will be in radians.

Value

A vector with three elements, corresponding to the latitude and longitude in degrees and the altitude in m.

References

https://arc.aiaa.org/doi/10.2514/6.2006-6753

Examples

```
# The following orbital parameters correspond to an object with NORAD catalogue
# number 24208 (Italsat 2) the 26th of June, 2006 at 00:58:29.34 UTC.

n0 <- 1.007781*((2*pi)/(1440))  # Multiplication by 2pi/1440 to convert to radians/min
e0 <- 0.002664 # mean eccentricity at epoch
i0 <- 3.8536*pi/180 # mean inclination at epoch in radians
M0 <- 48.3*pi/180 # mean anomaly at epoch in radians
omega0 <- 311.0977*pi/180 # mean argument of perigee at epoch in radians
OMEGA0 <- 80.0121*pi/180 # mean longitude of ascending node at epoch in radians
Bstar <- 1e-04 # drag coefficient
epochDateTime <- "2006-06-26 00:58:29.34"

# Lets calculate the position and velocity of the satellite 1 day later

state_1day_TEME <- sgdp4(n0=n0, e0=e0, i0=i0, M0=M0, omega0=omega0, OMEGA0=OMEGA0,
                         Bstar=Bstar, initialDateTime=epochDateTime, targetTime=1440)

# We can now convert the results in TEME frame to ECEF frame, previously
# multiplying by 1000 to convert the km output of sgdp4 to m

state_1day_ECEF <- TEMEtoECEF(state_1day_TEME$position, state_1day_TEME$velocity,
                              "2006-06-27 00:58:29.34")

# Finally, we can convert the ECEF coordinates to geodetic latitude, longitude
# and altitude

state_1day_geodetic <- ECEFtoLATLON(state_1day_ECEF$position)
```

ECItoKOE *Calculate ECI coordinates from Keplerian orbital elements*

Description

Keplerian orbital elements are a set of six parameters used to described the orbits of celestial objects, including satellites. While satellites do not follow a perfectly Keplerian orbit, their state at any point can be defined by the orbital parameters that they would have if they were located at the same position with the same velocity following a perfectly Keplerian orbit (i.e., if perturbations were absent). These are called osculating orbital elements.

Keplerian orbital elements can be unequivocally determined from a satellite if its position and velocity are known. This function calculates orbital elements from the position and velocity of a satellite in an ECI (Earth-centered inertial) frame of reference. The elements (such as the equatorial plane) with respect to which the resulting orbital elements will be defined are the same as those used for the ECI frame of reference. The function calculates the six standard orbital elements, plus some alternative elements useful for the characterization of special orbits, such as circular ones or orbits with no inclination.

Usage

```
ECItoKOE(position_ECI, velocity_ECI)
```

Arguments

`position_ECI` Vector with the X, Y and Z components of the position of an object in an ECI frame, in m.

`velocity_ECI` Vector with the X, Y and Z components of the velocity of an object in an ECI frame, in m/s.

Value

A list with the following standard and alternative orbital elements:

`semiMajorAxis` Semi-major axis of orbital ellipse in meters.

`eccentricity` Numerical eccentricity of the orbit. Eccentricity measures how much the orbit deviates from being circular.

`inclination` Inclination of the orbital plane in radians. Inclination is the angle between the orbital plane and the equator.

`meanAnomaly` Mean anomaly of the orbit in radians. Mean anomaly indicates where the satellite is along its orbital path, and is defined as the angle between the direction of the perigee and the hypothetical point where the object would be if it was moving in a circular orbit with the same period as its true orbit after the same amount of time since it last crossed the perigee had ellapsed.

`argumentPerigee` Argument of perigee in radians. This is the angle between the direction of the ascending node and the direction of the perigee (the point of the orbit at which the object is closest to the Earth).

`longitudeAscendingNode`
: Longitude of the ascending node (also called right ascension of the ascending node) in radians. This is the angle between the direction of the ascending node (the point where thesatellite crosses the equatorial plane moving north) and the direction of the First Point of Aries (which indicates the location of the vernal equinox).

`trueAnomaly`
: True anomaly of the orbit in radians. Unlike mean anomaly, true anomaly is the angle between the direction of the perigee and the actual position of the satellite.

`argumentLatitude`
: Argument of latitude of the orbit in radians. Defined as the angle between the equator and the position of the satellite. It is useful to define the position of satellites in circular orbits, where the argument of perigee and true anomaly are not well defined.

`longitudePerigee`
: Longitude of perigee of the orbit in radians. Defined as the angle between the vernal equinox and the perigee. It is useful for cases of orbits with 0 inclination, where the longitude of the ascending node and the argument of perigee are not well defined.

`trueLongitude`
: Longitude of perigee of the orbit in radians. Defined as the angle between the vernal equinox and the position of the satellite. It is useful for cases of circular orbits with 0 inclination, where the longitude of the ascending node, the argument of perigee and true anomaly are not well defined.

References

https://www.gsc-europa.eu/system-service-status/orbital-and-technical-parameters https://celestrak.com/columns/v02n01/
https://www.faa.gov/about/office_org/headquarters_offices/avs/offices/aam/cami/library/online_libraries/aerospace_medicin

Examples

```
# The following were the position and velocity of satellite MOLNIYA 1-83
# the 25th of June, 2006 at 00:33:43 UTC in the GCRF frame (in m and m/s).

position_GCRF <- c(-14471729.582, -4677558.558, 9369.461)
velocity_GCRF <- c(-3251.691, -3276.008, 4009.228)

# Let's calculate the orbital elements of the satellite at that time

orbital_elements <- ECItoKOE(position_GCRF, velocity_GCRF)
```

`GCRFtoECEF` *Convert coordinates from GCRF to ECEF*

Description

The GCRF (Geocentric Celestial Reference Frame) frame of reference is an Earth-centered inertial coordinate frame, where the origin is placed at the center of mass of Earth and the coordinate frame is fixed with respect to the stars (and therefore not fixed with respect to the Earth surface in its rotation). The X-axis is aligned with the mean equinox of Earth at 12:00 Terrestrial Time on the 1st of January, 2000, and the Z-axis is aligned with the Earth´s rotation axis.

It is equivalent to the J2000 frame of reference (also called EME2000), and in some contexts it is also referred to as ICRF frame (although in its strict definition, the origin of coordinates is placed at the barycenter of the Solar System).

In the ECEF frame, the origin is also placed at the center of mass of Earth, but the frame rotates with respect to the stars to remain fixed with respect to the Earth surface as it rotates.

The coordinates and velocities input and calculated with the high-precision orbital propagator (hpop) are in the GCRF frame of reference.

This function requires the `asteRiskData` package, which can be installed by running `install.packages('asteRiskData',`

Usage

```
GCRFtoECEF(position_GCRF, velocity_GCRF, dateTime)
```

Arguments

`position_GCRF`	Vector with the X, Y and Z components of the position of an object in GCRF frame, in m.
`velocity_GCRF`	Vector with the X, Y and Z components of the velocity of an object in GCRF frame, in m/s.
`dateTime`	Date-time string with the date and time in UTC corresponding to the provided position and velocity vectors.

Value

A list with two elements representing the position and velocity of the satellite in the ECEF (Earth Centered, Earth Fixed) frame of reference. Position values are in m, and velocity values are in m/s. Each of the two elements contains three values, corresponding to the X, Y and Z components of position and velocity in this order.

References

https://celestrak.com/columns/v02n01/

Examples

```
if(requireNamespace("asteRiskData", quietly = TRUE)) {
# The following were the position and velocity of satellite MOLNIYA 1-83
# the 25th of June, 2006 at 00:33:43 UTC in the GCRF frame (in m and m/s).

position_GCRF <- c(-14471729.582, -4677558.558, 9369.461)
velocity_GCRF <- c(-3251.691, -3276.008, 4009.228)
```

```
# Lets convert them into the ECEF frame

coordinates_ECEF <- GCRFtoECEF(position_GCRF, velocity_GCRF, "2006-06-27 00:58:29.34")
}
```

`GCRFtoLATLON` *Convert coordinates from GCRF to geodetic latitude, longitude and altitude*

Description

The GCRF (Geocentric Celestial Reference Frame) frame of reference is an Earth-centered inertial coordinate frame, where the origin is placed at the center of mass of Earth and the coordinate frame is fixed with respect to the stars (and therefore not fixed with respect to the Earth surface in its rotation). The X-axis is aligned with the mean equinox of Earth at 12:00 Terrestrial Time on the 1st of January, 2000, and the Z-axis is aligned with the Earth´s rotation axis. This function converts position in GCRF to geodetic latitude, longitude and altitude, which can be considered to be a non-inertial, Earth-centered frame of reference.

This function requires the `asteRiskData` package, which can be installed by running `install.packages('asteRiskData',`

Usage

```
GCRFtoLATLON(position_GCRF, dateTime, degreesOutput=TRUE)
```

Arguments

`position_GCRF`	Vector with the X, Y and Z components of the position of an object in TEME frame, in m.
`dateTime`	Date-time string with the date and time in UTC corresponding to the provided position vector.
`degreesOutput`	Logical indicating if the output should be in sexagesimal degrees. If `degreesOutput=FALSE`, the output will be in radians.

Value

A vector with three elements, corresponding to the latitude and longitude in degrees and the altitude in m.

References

https://arc.aiaa.org/doi/10.2514/6.2006-6753

Examples

```
if(requireNamespace("asteRiskData", quietly = TRUE)) {
# The following orbital parameters correspond to an object with NORAD catalogue
# number 24208 (Italsat 2) the 26th of June, 2006 at 00:58:29.34 UTC.

n0 <- 1.007781*((2*pi)/(1440))  # Multiplication by 2pi/1440 to convert to radians/min
e0 <- 0.002664 # mean eccentricity at epoch
i0 <- 3.8536*pi/180 # mean inclination at epoch in radians
M0 <- 48.3*pi/180 # mean anomaly at epoch in radians
omega0 <- 311.0977*pi/180 # mean argument of perigee at epoch in radians
OMEGA0 <- 80.0121*pi/180 # mean longitude of ascending node at epoch in radians
Bstar <- 1e-04 # drag coefficient
epochDateTime <- "2006-06-26 00:58:29.34"

# Lets calculate the position and velocity of the satellite 1 day later

state_1day_TEME <- sgdp4(n0=n0, e0=e0, i0=i0, M0=M0, omega0=omega0, OMEGA0=OMEGA0,
                         Bstar=Bstar, initialDateTime=epochDateTime, targetTime=1440)

# We can now convert the results in TEME frame to GCRF frame, previously
# multiplying by 1000 to convert the km output of sgdp4 to m

state_1day_GCRF <- TEMEtoGCRF(state_1day_TEME$position*1000,
                              state_1day_TEME$velocity*1000,
                              "2006-06-27 00:58:29.34")

# Finally, we convert the results in GCRF frame to geodetic latitude, longitude
# and altitude

state_1day_geodetic <- GCRFtoLATLON(state_1day_GCRF$position, "2006-06-27 00:58:29.34")
}
```

---

getLatestSpaceData *Retrieves the latest space data*

Description

The asteRiskData package provides the data and coefficients required for calculation of forces for hpop and other functions such certain conversions between reference frames. Some of the data tables included in the package are updated periodically with new data. These include Earth orientation parameters, space weather data and solar and geomagnetic storms. In order to perform the calculations dependent on such data for the most recent times, the latest available data must be retrieved.

This function automatically updates the data tables, enabling such calculations for the most recent times.

Usage

```
getLatestSpaceData()
```

Value

This function is invoked for its side effect, which is updating the data tables used internally for calculations requiring `asteRiskData` package, such as those performed by hpop.

References

http://www.celestrak.com/SpaceData/EOP-All.txt https://celestrak.com/SpaceData/SW-All.txt https://sol.spacenvironment.n

Examples

```
if(interactive()) {
if(requireNamespace("asteRiskData", quietly = TRUE)) {
# The table of Earth orientation parameters distributed with asteRiskData
# comprises data up to the 21st of March, 2021

asteRiskData::earthPositions[nrow(asteRiskData::earthPositions),]

# The table can be easily updated to include the most recent available data

getLatestSpaceData()
asteRiskData::earthPositions[nrow(asteRiskData::earthPositions),]
}
}
```

---

hpop — *High-precision numerical orbital propagator*

Description

Given the position and velocity of a satellite at a given time (in the GCRF system of coordinates), propagates its position by calculating its acceleration (based on a force model) and solving the resulting second-order ODE through numerical integration. This allows propagation of orbits with considerably higher accuracy than other propagators such as SGP4 and SDP4, but at the expense of a much higher computational cost. The forces and effects currently considered are gravitational attraction by the Earth (using a geopotential model based on spherical harmonics); effects of Earth ocean and solid tides; gravitational attraction by the Moon, Sun and planets (considered as point masses); solar radiation pressure; atmospheric drag, and relativistic effects. The force field is based on the forces described in Satellite Orbits: Models, Methods and Applications (Oliver Montenbruck and Eberhard Gill) and Fundamentals of Astrodynamics and Applications (David Vallado). Parts of this implementation are based on a previous MATLAB implementation by Meysam Mahooti. The NRLMSISE-00 model is used to calculate atmospheric density for the calculation of atmospheric drag. The high-precision numerical orbital propagator requires the `asteRiskData` package, which provides the data and coefficients required for calculation of the modeled forces. `asteRiskData` can be installed by running `install.packages('asteRiskData',repos='https://rafael-ayala.github.io/drat/')`

Usage

```
hpop(position, velocity, dateTime, times, satelliteMass, dragArea,
     radiationArea, dragCoefficient, radiationCoefficient, ...)
```

Arguments

- `position` — Initial position of the satellite in the GCRF system of coordinates. Should be provided as a numeric vector with 3 components that indicate the X, Y and Z components of the position in meters.
- `velocity` — Initial velocity of the satellite in the GCRF system of coordinates. Should be provided as a numeric vector with 3 components that indicate the X, Y and Z components of the position in meters/second.
- `dateTime` — Date time string in the YYYY-MM-DD HH:MM:SS format indicating the time corresponding to the initial position and velocity, in UTC time.
- `times` — Vector with the times at which the position and velocity of the satellite should be calculated, in seconds since the initial time.
- `satelliteMass` — Mass of the satellite in kilograms.
- `dragArea` — Effective area of the satellite for atmospheric drag in squared meters.
- `radiationArea` — Effective area of the satellite subject to the effect of radiation pressure in squared meters.
- `dragCoefficient` — Drag coefficient (Cd) used for the calculation of atmospheric drag. For low Earth-orbiting satellites, a value of 2.2 is frequently employed if a better approximation is not available.
- `radiationCoefficient` — Coefficient for the force resulting from radiation pressure. This parameter is usually referred to as reflectivity coefficient (Cr) and the value varies for different satellites and orbits. If unknown, a value of 1.2 is usually a decent approximation.
- `...` — Additional parameters to be passed to ode to control how numerical integration is performed. By default, the RADAU5 solver is used.

Value

A matrix with the results of the numerical integration at the requested times. Each row contains the results for one of the requested times. The matrix contains seven columns: time (indicating the time for the corresponding row in seconds since the initial time), X, Y, Z (indicating the X, Y and Z components of the position for that time in meters), dX, dY and dZ (indicating the X, Y and Z components of the velocity for that time in meters/second). Positions and velocities are returned in the GCRF frame of reference.

References

Satellite Orbits: Models, Methods and Applications. Oliver Montenbruck and Eberhard Gill. Fundamentals of Astrodynamics and Applications. David Vallado. https://www.mathworks.com/matlabcentral/fileexchange/551
high-precision-orbit-propagator https://ccmc.gsfc.nasa.gov/modelweb/models/nrlmsise00.php

Examples

```
if(requireNamespace("asteRiskData", quietly = TRUE)) {
# The following are the position and velocity in the GCRF frame of satellite
```

```
# MOLNIYA 1-83 the 25th of June, 2006 at 00:33:43 UTC.

initialPosition <-c(-14568679.5026116, -4366250.78287623, 9417.9289105405)
initialVelocity <- c(-3321.17428902497, -3205.49400830455, 4009.26862308806)
initialTime <- "2006-06-25 00:33:43"

# Molniya satellites have a mass of approximately 1600 kg and a cross-section of
# 15 m2. Additionally, lets use 2.2 and 1.2 as approximately values of the
# drag and reflectivity coefficients, respectively.

molniyaMass <- 1600
molniyaCrossSection <- 15
molniyaCr <- 1.2
molniyaCd <- 2.2

# Lets calculate the position and velocity of the satellite for each minute of
# the following 10 minutes.

targetTimes <- seq(0, 600, by=60)
hpop_results <- hpop(initialPosition, initialVelocity, initialTime, targetTimes,
                     molniyaMass, molniyaCrossSection, molniyaCrossSection,
                     molniyaCr, molniyaCd)
}
```

KOEtoECI *Calculate ECI coordinates from Keplerian orbital elements*

Description

Keplerian orbital elements are a set of six parameters used to described the orbits of celestial objects, including satellites. While satellites do not follow a perfectly Keplerian orbit, their state at any point can be defined by the orbital parameters that they would have if they were located at the same position with the same velocity following a perfectly Keplerian orbit (i.e., if perturbations were absent). These are called osculating orbital elements.

A complete set of six Keplerian elements defines unequivocally the position and velocity of the satellite in a given frame of reference, and therefore can be used to calculate its cartesian coordinates. This function calculates the coordinates of a satellite in an ECI (Earth-centered inertial) frame of reference from a set of Keplerian orbital elements. The exact ECI frame of the resulting coordinates is the same used to define the supplied orbital elements.

Usage

```
KOEtoECI(a, e, i, M, omega, OMEGA, keplerAccuracy=10e-8, maxKeplerIterations=100)
```

Arguments

a	Semi-major axis of orbital ellipse in meters.
e	Numerical eccentricity of the orbit. Eccentricity measures how much the orbit deviates from being circular.

`i` Inclination of the orbital plane in radians. Inclination is the angle between the orbital plane and the equator.

`M` Mean anomaly of the orbit in radians. Mean anomaly indicates where the satellite is along its orbital path, and is defined as the angle between the direction of the perigee and the hypothetical point where the object would be if it was moving in a circular orbit with the same period as its true orbit after the same amount of time since it last crossed the perigee had ellapsed.

`omega` Argument of perigee in radians. This is the angle between the direction of the ascending node and the direction of the perigee (the point of the orbit at which the object is closest to the Earth).

`OMEGA` Right ascension of the ascending node in radians. This is the angle between the direction of the ascending node (the point where the satellite crosses the equatorial plane moving north) and the direction of the First Point of Aries (which indicates the location of the vernal equinox).

`keplerAccuracy` Accuracy to consider Kepler's equation solved when calculating eccentric anomaly from mean anomaly. If two consecutive solutions differ by a value lower than this accuracy, integration is considered to have converged.

`maxKeplerIterations` Maximum number of iterations after which fixed-point integration of Kepler's equation will stop, even if convergence according to the accuracy criterion has not been reached.

Value

A list with two elements representing the position and velocity of the satellite in the same ECI (Earth Centered, Earth Fixed) frame of reference into which the provided orbital elements were defined. Position values are in m, and velocity values are in m/s. Each of the two elements contains three values, corresponding to the X, Y and Z components of position and velocity in this order.

References

https://www.gsc-europa.eu/system-service-status/orbital-and-technical-parameters https://celestrak.com/columns/v02n01/
https://downloads.rene-schwarz.com/download/M001-Keplerian_Orbit_Elements_to_Cartesian_State_Vectors.pdf

Examples

```
# Let's calculate the ECI coordinates from the orbital elements provided by a
# TLE. It should be noted that this is often not recommended, since the orbital
# elements supplied in a TLE are not osculating orbital elements, but instead
# mean orbital elements set to fit a range of actual observations. The
# recommended procedures are to use TLE only in conjunction with the SGP4/SDP4
# models, and viceversa.
# The following orbital parameters correspond to an object with NORAD catalogue
# number 24208 (Italsat 2) the 26th of June, 2006 at 00:58:29.34 UTC.

n0 <- 1.007781*((2*pi)/(86400))  # Multiplication by 2pi/86400 to convert to radians/s
e0 <- 0.002664 # mean eccentricity at epoch
i0 <- 3.8536*pi/180 # mean inclination at epoch in radians
M0 <- 48.3*pi/180 # mean anomaly at epoch in radians
```

```
omega0 <- 311.0977*pi/180 # mean argument of perigee at epoch in radians
OMEGA0 <- 80.0121*pi/180 # mean longitude of ascending node at epoch in radians

# The semi-major axis can be calculated from the mean motion in radians/s
# as follows: (mu is the standard gravitational parameter of Earth)

mu <- 3.986004418e14 # in units of m3 s-2
a0 <- (mu^(1/3))/(n0^(2/3))

# The ECI coordinates can then be calculated. In this case, they will be in TEME
# frame, since the original orbital elements are derived from a TLE
coordinates_ECI <- KOEtoECI(a0, e0, i0, M0, omega0, OMEGA0)
```

`parseTLElines` *Parse the lines of a TLE*

Description

TLE (Two-/Three- Line Element) is the standard format for representing orbital state vectors. This function parses a character vector where each element represents a line of the TLE. The supplied character vector can have either 2 (for Two Line Elements) or 3 (for Three Line Elements) elements. The two lines of a Two Line Element contain all the information. The additional line in a Three Line Element is optional, and contains just the satellite name. For a detailed description of the TLE format, see https://celestrak.com/columns/v04n03/#FAQ01.

Usage

```
parseTLElines(lines)
```

Arguments

`lines` Character vector where each element is a string corresponding to a line of the TLE. The character vector must have either 2 or 3 elements.

Value

A list with the following elements that define the orbital state vector of the satellite:

`NORADcatalogNumber`
: NORAD Catalog Number, also known as Satellite Catalog Number, assigned by United States Space Command to each artificial object orbiting Earth

`classificationLevel`
: Classification level of the information for the orbiting object. Can be unclassified, classified, secret or unknown

`internationalDesignator`
: International Designator, also known as COSPAR ID, of the object. It consists of the launch year, separated by a hyphen from a three-digit number indicating the launch number for that year and a set of one to three letters indicating the piece for a launch with multiple pieces.

- `launchYear` The launch year of the object
- `launchNumber` The launch number of the object during its launch year
- `launchPiece` The piece for the launch of the object, if it was a launch with multiple pieces
- `dateTime` Date time string to which the orbital state vector corresponds
- `elementNumber` Element number for the object. In principle, every time a new TLE is generated for an object, the element number is incremented, and therefore element numbers could be used to assess if all the TLEs for a certain object are available. However, in practice it is observed that this is not always the case, with some numbers skipped and some numbers repeated.
- `inclination` Mean orbital inclination of the satellite in degrees. This is the angle between the orbital plane of the satellite and the equatorial plane
- `ascension` Mean longitude of the ascending node of the satellite at epoch, also known as right ascension of the ascending node, in degrees. This is the angle between the direction of the ascending node (the point where the satellite crosses the equatorial plane moving north) and the direction of the First Point of Aries (which indicates the location of the vernal equinox)
- `eccentricity` Mean eccentricity of the orbit of the object. Eccentricity is a measurement of how much the orbit deviates from a circular shape, with 0 indicating a perfectly circular orbit and 1 indicating an extreme case of parabolic trajectory
- `perigeeArgument` Mean argument of the perigee of the object in degrees. This is the angle between the direction of the ascending node and the direction of the perigee (the point of the orbit at which the object is closest to the Earth)
- `meanAnomaly` Mean anomaly of the orbit of the object in degrees. This indicates where the satellite is along its orbital path. It is provided as the angle between the direction of the perigee and the hypothetical point where the object would be if it was moving in a circular orbit with the same period as its true orbit after the same amount of time since it last crossed the perigee had ellapsed. Therefore, 0 denotes that the object is at the perigee
- `meanMotion` Mean motion of the satellite at epoch in revolutions/day
- `meanMotionDerivative` First time derivative of the mean motion of the satellite in revolutions/day^2^
- `meanMotionSecondDerivative` Second time derivative of the mean motion of the satellite in revolutions/day^3^.
- `Bstar` Drag coefficient of the satellite in units of (earth radii)^-1^. Bstar is an adjusted value of the ballistic coefficient of the satellite, and it indicates how susceptible it is to atmospheric drag.
- `ephemerisType` Source for the ephemeris (orbital state vector). Most commonly, it is distributed data obtained by combaining multiple observations with the SGP4/SDP4 models
- `epochRevolutionNumber` Number of full orbital revolutions completed by the object
- `objectName` Name of the object, retrieved from the first line of the TLE if a Three Line Element was provided

References

https://celestrak.com/columns/v04n03/#FAQ01

Examples

```
# The following lines correspond to a TLE for Italsat 2 the 26th of June, 2006
# at 00:58:29.34 UTC.

italsat2_lines <- c("ITALSAT 2",
"1 24208U 96044A   06177.04061740 -.00000094  00000-0  10000-3 0  1600",
"2 24208   3.8536  80.0121 0026640 311.0977  48.3000  1.00778054 36119")

italsat2_TLE <- parseTLElines(italsat2_lines)
italsat2_TLE
```

---

readGLONASSNavigationRINEX

Read a RINEX navigation file for GLONASS satellites

Description

RINEX (Receiver Independent Exchange Format) is one of the most widely used formats for providing data of satellite navigation systems. The RINEX standard defines several file types, among which navigation files are used to distribute positional information of the satellites. The exact information provided in a RINEX navigation file varies for each satellite navigation system. This function reads RINEX navigation files for satellites of the GLONASS constellation, operated by Russia.

Usage

```
readGLONASSNavigationRINEX(filename)
```

Arguments

`filename`	Path to the GLONASS RINEX navigation file.

Value

A list with two elements. The first element, named `header`, is a list with the following elements:

`rinexVersion`	Version of the RINEX format used in the file
`rinexFileType`	Type of RINEX file
`generatorProgram`	Program used to generate the RINEX file
`generatorEntity`	Individual or organization that generated the file

`fileCreationDateString`
: Date-time string indicating when the file was created

`refYear`
: Reference year for system time correction

`refMonth`
: Reference month for system time correction

`refDay`
: Reference day for system time correction

`sysTimeCorrection`
: Correction to system time scale to fine-tune GLONASS time to UTC in seconds. Since GLONASS time is linked to UTC, it should be a very small amount

`leapSeconds`
: Leap seconds introduced since 1980. Useful to convert to GPS time

`comments`
: Miscellaneous comments found in the header of the RINEX file

The second element is named `messages`, and it contains one element for each navigation message found in the RINEX file. Each of these elements is a list with the following elements that provide information about the position of the GLONASS satellite:

`satelliteNumber`
: Slot number of the satellite within the GLONASS constellation. It can be converted to a PRN code by adding 37 to it

`epochYearShort`
: Epoch year in 2-digit format. If lower than 80, the meaning should be taken as 20XX, while if larger than 80, it refers to 19XX.

`epochMonth`
: Epoch month

`epochDay`
: Epoch day

`epochHour`
: Epoch hour

`epochMinute`
: Epoch minute

`epochSecond`
: Epoch second

`UTCepochDateTime`
: Date-time string indicating the time corresponding to the reported position in the present message. The time corresponds to that of the satellite system, which is in GLONASS time and therefore can be considered as equivalent to UTC time for most purposes. For an even more accurate conversion to actual UTC time, the clock bias and clock drift (described in the next two elements), and possibly the system time correction provided in the header.

`clockBias`
: Clock bias (i.e., constant offset) that should be applied to the satellite time in order to obtain an even more accurate UTC time. In seconds

`relativeFreqBias`
: Clock drift of the satellite clock that should be applied in combination with the time difference to the reference time in order to obtain an even more accurate UTC time. In seconds per second

`messageFrameTime`
: Second of the UTC day when the message was transmitted

`positionX`
: X coordinate of the position of the satellite in km, in the GCRF system of coordinates

`positionY`
: Y coordinate of the position of the satellite in km, in the GCRF system of coordinates

`positionZ`	Z coordinate of the position of the satellite in km, in the GCRF system of coordinates
`velocityX`	X component of the velocity of the satellite in km/s, in the GCRF system of coordinates
`velocityY`	Y component of the velocity of the satellite in km/s, in the GCRF system of coordinates
`velocityZ`	Z component of the velocity of the satellite in km/s, in the GCRF system of coordinates
`accelX`	X component of the accel of the satellite in km/s, in the GCRF system of coordinates
`accelY`	Y component of the accel of the satellite in km/s, in the GCRF system of coordinates
`accelZ`	Z component of the accel of the satellite in km/s, in the GCRF system of coordinates
`satelliteHealthCode`	Code indicating the health of the satellite. 0 if healthy
`freqNumber`	Frequency number (k) of the GLONASS satellite. The two frequencies in MHz, f1 and f2, used by the satellite to transmit data can be calculated as follows: f1 = 1602 + k*9/16 and f2 = 1246 + k*7/16
`informationAge`	Age in days of the observation data used to generate the provided ephemeris

References

https://gage.upc.edu/gFD/ https://www.navcen.uscg.gov/pubs/gps/rinex/rinex.txt ftp://www.ngs.noaa.gov/cors/RINEX211.tx
http://acc.igs.org/misc/rinex304.pdf

Examples

```
# The file testGLONASSRINEX.txt provided with the package includes 5 navigation
# messages from 4 GLONASS satellites

testGLONASSnav <- readGLONASSNavigationRINEX(paste0(path.package("asteRisk"),
"/testGLONASSRINEX.txt"))
testGLONASSnav$header
testGLONASSnav$messages
```

`readGPSNavigationRINEX` *Read a RINEX navigation file for GPS satellites*

Description

RINEX (Receiver Independent Exchange Format) is one of the most widely used formats for providing data of satellite navigation systems. The RINEX standard defines several file types, among which navigation files are used to distribute positional information of the satellites. The exact information provided in a RINEX navigation file varies for each satellite navigation system. This function reads RINEX navigation files for satellites of the GPS constellation, operated by the USA.

Usage

```
readGPSNavigationRINEX(filename)
```

Arguments

`filename` Path to the GPS RINEX navigation file.

Value

A list with two elements. The first element, named `header`, is a list with the following elements:

- `rinexVersion` Version of the RINEX format used in the file
- `rinexFileType` Type of RINEX file
- `generatorProgram` Program used to generate the RINEX file
- `generatorEntity` Individual or organization that generated the file
- `fileCreationDateString` Date-time string indicating when the file was created
- `ionAlphaA0` Coefficient for ionospheric correction A0
- `ionAlphaA1` Coefficient for ionospheric correction A1
- `ionAlphaA2` Coefficient for ionospheric correction A2
- `ionAlphaA3` Coefficient for ionospheric correction A3
- `ionBetaB0` Coefficient for ionospheric correction B0
- `ionBetaB1` Coefficient for ionospheric correction B1
- `ionBetaB2` Coefficient for ionospheric correction B2
- `ionBetaB3` Coefficient for ionospheric correction B3
- `deltaUTCA0` A0 parameter to compute accurate time in UTC
- `deltaUTCA1` A1 parameter to compute accurate time in UTC
- `referenceTimeUTC` Time in seconds of current UTC week for reference time
- `referenceWeekUTC` UTC reference week number
- `leapSeconds` Leap seconds introduced since 1980. Useful to convert to UTC time (UTC time = GPS time - leap seconds)
- `comments` Miscellaneous comments found in the header of the RINEX file

The second element is named `messages`, and it contains one element for each navigation message found in the RINEX file. Each of these elements is a list with the following elements that provide information about the position of the GPS satellite:

- `satellitePRNCode` PRN code of the satellite. Unique PRN codes are assigned to all satellites in global navigation satellite systems, and therefore provide an identifier for each of them

`epochYearShort`
: Epoch year in 2-digit format. If lower than 80, the meaning should be taken as 20XX, while if larger than 80, it refers to 19XX.

`epochMonth`
: Epoch month

`epochDay`
: Epoch day

`epochHour`
: Epoch hour

`epochMinute`
: Epoch minute

`epochSecond`
: Epoch second

`UTCepochDateTime`
: Date-time string indicating the time corresponding to the reported position in the present message. The time is in UTC, obtained by subtracting the leap seconds (if available in the header) from the time of the satellite system (which is in GPS time). If leap seconds are not provided in the header, the time will be in GPS. For an even more accurate conversion to actual UTC time, the clock bias, clock drift and possibly even clock drift rate (described in the next three elements) must be considered

`clockBias`
: Clock bias (i.e., constant offset) that should be applied to the satellite time in order to obtain an even more accurate UTC time. In seconds

`clockDrift`
: Clock drift of the satellite clock that should be applied in combination with the time difference to the reference time in order to obtain an even more accurate UTC time. In seconds per second

`clockDriftRate`
: Rate of change for the clock drift of the satellite clock. It is frequently 0, but if not, it should be applied in combination with clock bias and clock drift in order to obtain an even more accurate UTC time. In seconds per square second.

`IODE`
: Issue-of-data ephemeris. It acts as a time-stamp or unique identifier for the provided navigation data. In particular, the IODE of a given navigation message should never be the same as the IODE for any other navigation message broadcasted by the same satellite in the past 6 days, although violations of this rule have been observed. Most frequently, IODE are not reused in a period of 7 days, so that they match exactly the IODC.

`radiusCorrectionSine`
: Amplitude of the sine harmonic component for the correction of orbital radius. In meters

`deltaN`
: Mean motion difference from computed value. In radians per second In order to obtain the real (perturbed) mean motion, first the Keplerian mean motion should be calculated from the semi-major axis. Then, deltaN should be added to it.

`meanAnomaly`
: Mean anomaly of the satellite at epoch. In radians. This indicates where the satellite is along its orbital path. It is provided as the angle between the direction of the perigee and the hypothetical point where the object would be if it was moving in a circular orbit with the same period as its true orbit after the same amount of time since it last crossed the perigee had ellapsed. Therefore, 0 denotes that the object is at the perigee. This is a Keplerian orbital element.

`latitudeCorrectionCosine`
: Amplitude of the cosine harmonic component for the correction of latitude argument. In radians

`eccentricity` Eccentricity of the orbit of the satellite at epoch. This is a Keplerian orbital element.

`latitudeCorrectionSine`
Amplitude of the sine harmonic component for the correction of latitude argument. In radians

`semiMajorAxis` Semi-major axis of the orbit of the satellite at epoch. In meters. This is a Keplerian orbital element

`timeOfEphemeris`
Time of the GPS week (in seconds) for the ephemeris

`inclinationCorrectionCosine`
Amplitude of the cosine harmonic component for the correction of inclination. In radians

`ascension` Longitude of the ascending node of the satellite at epoch, also known as right ascension of the ascending node, in radians. This is the angle between the direction of the ascending node (the point where the satellite crosses the equatorial plane moving north) and the direction of the First Point of Aries (which indicates the location of the vernal equinox). This is a Keplerian orbital element.

`inclinationCorrectionSine`
Amplitude of the sine harmonic component for the correction of inclination. In radians

`inclination` Mean orbital inclination of the satellite in radians. This is the angle between the orbital plane of the satellite and the equatorial plane. This is a Keplerian orbital element.

`radiusCorrectionCosine`
Amplitude of the cosine harmonic component for the correction of orbital radius. In meters.

`perigeeArgument`
Mean argument of the perigee of the object in radians. This is the angle between the direction of the ascending node and the direction of the perigee (the point of the orbit at which the object is closest to the Earth). This is a Keplerian orbital element.

`OMEGADot` Angular velocity of the satellite with respect to the vernal equinox. In radians/second.

`codesL2Channel`
Flag indicating if coarse/acquisition (C/A) code is being transmitted on the L2 channel (value of 1) or not (value of 0)

`GPSWeek` GPS week number at epoch

`dataFlagL2P` Flag indicating if precise (P) code is being transmitted on the L2 channel (value of 1) or not (value of 0)

`satelliteAccuracy`
Accuracy of the position of the satellite, in meters.

`satelliteHealthCode`
Code indicating the health of the satellite. 0 if healthy.

`totalGroupDelay`
Bias difference between codes broadcasted on L1 and the ionospheric-free combination of the codes broadcasted at L1 and L2, in seconds. This parameter, also

known as timing group delay (TGD), should be considered when calculating satellite clock error.

IODC
: Issue-of-data clock. It acts as a time-stamp or unique identifier for the provided navigation data. In particular, the IODC of a given navigation message should never be the same as the IODC for any other navigation message broadcasted by the same satellite in the past 7 days, although violations of this rule have been observed. Most frequently, IODE are not reused in a period of 7 days instead of the mandatory 6 days, so that they match exactly the IODC.

transmissionTime
: Transmission time for the navigation message, in seconds of GPS week.

fitInterval
: Flag indicating for how long the broadcasted ephemeris are valid since the last time the data was updated. It should be noted that in order to obtain positional values/orbital elements at times other than epoch, the corrections for perturbed orbital elements should be applied and propagated. If 0, the ephemeris data are valid for up to 4 hours. If 1, they are valid for more than 4 hours.

References

https://gage.upc.edu/gFD/ https://www.navcen.uscg.gov/pubs/gps/rinex/rinex.txt ftp://www.ngs.noaa.gov/cors/RINEX211.tx
http://acc.igs.org/misc/rinex304.pdf https://www.icao.int/Meetings/AMC/MA/2004/GNSS/icd.pdf

Examples

```
# The file testGPSRINEX.txt provided with the package includes 3 navigation
# messages from 3 GPS satellites

testGPSnav <- readGPSNavigationRINEX(paste0(path.package("asteRisk"),
"/testGPSRINEX.txt"))
testGPSnav$header
testGPSnav$messages
```

---

readTLE *Read a TLE file*

Description

TLE (Two-/Three- Line Element) is a standard format for representing orbital state vectors. This function reads a TLE file containing one or more TLEs. The TLE file can contain either Two Line Elements or Three Line Elements, but all the TLE in a single file must be of the same type. The two lines of a Two Line Element contain all the ephemeris information. The additional line in a Three Line Element is optional, and contains just the satellite name. For a detailed description of the TLE format, see https://celestrak.com/columns/v04n03/#FAQ01.

Usage

```
readTLE(filename)
```

Arguments

`filename` Path to the TLE file.

Value

A list with the following elements that define the orbital state vector of the satellite (or, if the file contained multiple TLE, a nested list, where each element of the top level list represents an orbital state vector, and comprises the following elements):

`NORADcatalogNumber`
: NORAD Catalog Number, also known as Satellite Catalog Number, assigned by United States Space Command to each artificial object orbiting Earth

`classificationLevel`
: Classification level of the information for the orbiting object. Can be unclassified, classified, secret or unknown

`internationalDesignator`
: International Designator, also known as COSPAR ID, of the object. It consists of the launch year, separated by a hyphen from a three-digit number indicating the launch number for that year and a set of one to three letters indicating the piece for a launch with multiple pieces.

`launchYear` The launch year of the object

`launchNumber` The launch number of the object during its launch year

`launchPiece` The piece for the launch of the object, if it was a launch with multiple pieces

`dateTime` Date time string to which the orbital state vector corresponds

`elementNumber` Element number for the object. In principle, every time a new TLE is generated for an object, the element number is incremented, and therefore element numbers could be used to assess if all the TLEs for a certain object are available. However, in practice it is observed that this is not always the case, with some numbers skipped and some numbers repeated.

`inclination` Mean orbital inclination of the satellite in degrees. This is the angle between the orbital plane of the satellite and the equatorial plane

`ascension` Mean longitude of the ascending node of the satellite at epoch, also known as right ascension of the ascending node, in degrees. This is the angle between the direction of the ascending node (the point where the satellite crosses the equatorial plane moving north) and the direction of the First Point of Aries (which indicates the location of the vernal equinox)

`eccentricity` Mean eccentricity of the orbit of the object. Eccentricity is a measurement of how much the orbit deviates from a circular shape, with 0 indicating a perfectly circular orbit and 1 indicating an extreme case of parabolic trajectory

`perigeeArgument`
: Mean argument of the perigee of the object in degrees. This is the angle between the direction of the ascending node and the direction of the perigee (the point of the orbit at which the object is closest to the Earth)

`meanAnomaly` Mean anomaly of the orbit of the object in degrees. This indicates where the satellite is along its orbital path. It is provided as the angle between the direction of the perigee and the hypothetical point where the object would be if it was moving in a circular orbit with the same period as its true orbit after the same amount of time since it last crossed the perigee had ellapsed. Therefore, 0 denotes that the object is at the perigee

`meanMotion` Mean motion of the satellite at epoch in revolutions/day

`meanMotionDerivative`
First time derivative of the mean motion of the satellite in revolutions/day^2^

`meanMotionSecondDerivative`
Second time derivative of the mean motion of the satellite in revolutions/day^3^.

`Bstar` Drag coefficient of the satellite in units of (earth radii)^-1^. Bstar is an adjusted value of the ballistic coefficient of the satellite, and it indicates how susceptible it is to atmospheric drag.

`ephemerisType` Source for the ephemeris (orbital state vector). Most commonly, it is distributed data obtained by combaining multiple observations with the SGP4/SDP4 models

`epochRevolutionNumber`
Number of full orbital revolutions completed by the object

`objectName` Name of the object, retrieved from the first line of the TLE if a Three Line Element was provided

References

https://celestrak.com/columns/v04n03/#FAQ01 http://www.celestrak.com/publications/aiaa/2006-6753/AIAA-2006-6753.pdf

Examples

```
# The file testTLE.txt provided with the package includes 29 TLE covering a
# variety of satellites, extracted from Revisiting Space Track Report #3

test_TLEs <- readTLE(paste0(path.package("asteRisk"), "/testTLE.txt"))
test_TLEs
```

`sdp4` *Propagate an orbital state vector with the SDP4 model*

Description

Given an orbital state vector of a satellite, applies the SDP4 model to propagate its orbit to the desired time point. This allows the calculation of the position and velocity of the satellite at different times, both before and after the time corresponding to the known state vector (referred to as "epoch"). Kepler's equation is solved through fixed-point integration. The SDP4 model is a modified version of the SGP4 model that takes into account the secular and periodic perturbations of the Moon and the Sun on the orbit of the satellite. It also considers the Earth resonance effects on 24-hour geostationary and 12-hour Molniya orbits. Thanks to this, the SDP4 model can correctly propagate the orbit of objects in deep space (with orbital periods larger than 225 minutes, corresponding to altitudes higher than 5877.5 km). However, it should be noted that SDP4 employs only simplified drag equations, and lacks corrections for low-perigee orbits. Therefore, it is recommended to apply the standard SGP4 model (available through the sgp4 function) for satellites that are not in deep space. This implementation is based on a previous SDP4 implementation in Julia (SatelliteToolbox).

Usage

```
sdp4(n0, e0, i0, M0, omega0, OMEGA0, Bstar, initialDateTime, targetTime,
     keplerAccuracy=10e-8, maxKeplerIterations=100)
```

Arguments

`n0`	Mean motion of the satellite at epoch in radians/min.
`e0`	Mean eccentricity of the orbit of the satellite at epoch. Eccentricity ranges from 0 (perfectly circular orbit) to 1 (parabolic trajectory).
`i0`	Mean orbital inclination of the satellite at epoch in radians.
`M0`	Mean anomaly of the satellite at epoch.
`omega0`	Mean argument of perigee of the satellite at epoch.
`OMEGA0`	Mean longitude of the ascending node of the satellite at epoch. Also known as right ascension of the ascending node.
`Bstar`	Drag coefficient of the satellite in units of (earth radii)^-1^. Bstar is an adjusted value of the ballistic coefficient of the satellite, and it indicates how susceptible it is to atmospheric drag.
`initialDateTime`	Date-time string in UTC indicating the time corresponding to the known state vector of the satellite. Unlike for SGP4, it must be provided in all cases since it is required to calculate Moon and Sun perturbations.
`targetTime`	Time at which the position and velocity of the satellite should be calculated. It can be provided in two different ways: either as a number corresponding to the time in minutes counting from epoch at which the orbit should be propagated, or as a date-time string in UTC.
`keplerAccuracy`	Accuracy to consider Kepler´s equation solved. If two consecutive solutions differ by a value lower than this accuracy, integration is considered to have converged.
`maxKeplerIterations`	Maximum number of iterations after which fixed-point integration of Kepler's equation will stop, even if convergence according to the accuracy criterion has not been reached.

Value

A list with three elements. The first two elements represent the position and velocity of the satellite at the target time, in the TEME (True Equator, Mean Equinox) frame of reference. Position values are in km, and velocity values are in km/s. Each of these two elements contains three values, corresponding to the X, Y and Z components of position and velocity in this order. The third element indicates the algorithm used to propagate the orbit (sdp4).

References

https://juliapackages.com/p/satellitetoolbox https://celestrak.com/NORAD/documentation/spacetrk.pdf http://www.celestrak.com/publications/aiaa/2006-6753/AIAA-2006-6753.pdf

Examples

```
# The following orbital parameters correspond to an object with NORAD catalogue
# number 24208 (Italsat 2) the 26th of June, 2006 at 00:58:29.34 UTC.

n0 <- 1.007781*((2*pi)/(1440))  # Multiplication by 2pi/1440 to convert to radians/min
e0 <- 0.002664 # mean eccentricity at epoch
i0 <- 3.8536*pi/180 # mean inclination at epoch in radians
M0 <- 48.3*pi/180 # mean anomaly at epoch in radians
omega0 <- 311.0977*pi/180 # mean argument of perigee at epoch in radians
OMEGA0 <- 80.0121*pi/180 # mean longitude of ascending node at epoch in radians
Bstar <- 1e-04 # drag coefficient
epochDateTime <- "2006-06-26 00:58:29.34"

# Calculation of the orbital period

2*pi/n0

# The period is higher than 225 min, and therefore the SDP4 model should be
# applied. Furthermore, from the mean motion in revolutions/day, it can be
# seen that it is a geostarionary satellite with a 24-hour period. Lets
# calculate and compare the position and velocity of the satellite at epoch
# and 1 day later.

state_0 <- sdp4(n0=n0, e0=e0, i0=i0, M0=M0, omega0=omega0, OMEGA0=OMEGA0,
                Bstar=Bstar, initialDateTime=epochDateTime, targetTime=0)
state_1day <- sdp4(n0=n0, e0=e0, i0=i0, M0=M0, omega0=omega0, OMEGA0=OMEGA0,
                   Bstar=Bstar, initialDateTime=epochDateTime, targetTime=1440)
state_0
state_1day

# The position and velocity are very similar after a full day, in accordance
# with the geostationary orbit
```

---

sgdp4 *Propagate an orbital state vector with the SGP4/SDP4 model*

Description

Given an orbital state vector of a satellite, applies the SGP4 or SDP4 model to propagate its orbit to the desired time point, as appropriate depending on the orbital period. The model will be automatically chosen depending on the orbital period. For objects in deep space (with orbital periods larger than 225 minutes, equivalent to altitudes higher than 5877.5 km) the SDP4 model will be applied. For objects near Earth (orbital periods shorter than 225 minutes, or altitudes lower than 5877.5 km) the SGP4 model will be used. It is not recommended to apply SGP4 to objects in deep space or SDP4 to objects near Earth, but this can be forced by calling directly the sgp4 and sdp4 functions.

Usage

```
sgdp4(n0, e0, i0, M0, omega0, OMEGA0, Bstar, initialDateTime=NULL, targetTime,
      keplerAccuracy=10e-8, maxKeplerIterations=100)
```

Arguments

`n0` Mean motion of the satellite at epoch in radians/min.

`e0` Mean eccentricity of the orbit of the satellite at epoch. Eccentricity ranges from 0 (perfectly circular orbit) to 1 (parabolic trajectory).

`i0` Mean orbital inclination of the satellite at epoch in radians.

`M0` Mean anomaly of the satellite at epoch.

`omega0` Mean argument of perigee of the satellite at epoch.

`OMEGA0` Mean longitude of the ascending node of the satellite at epoch. Also known as right ascension of the ascending node.

`Bstar` Drag coefficient of the satellite in units of (earth radii)^-1^. Bstar is an adjusted value of the ballistic coefficient of the satellite, and it indicates how susceptible it is to atmospheric drag.

`initialDateTime` Date-time string in UTC indicating the time corresponding to the known state vector of the satellite. It must be provided for objects in deep space, and also for objects near Earth if `targetTime` is provided as a date-time string.

`targetTime` Time at which the position and velocity of the satellite should be calculated. It can be provided in two different ways: either as a number corresponding to the time in minutes counting from epoch at which the orbit should be propagated, or as a date-time string in UTC, in which case the date-time string for epoch must be provided through `initialDateTime`.

`keplerAccuracy` Accuracy to consider Kepler´s equation solved. If two consecutive solutions differ by a value lower than this accuracy, integration is considered to have converged.

`maxKeplerIterations` Maximum number of iterations after which fixed-point integration of Kepler's equation will stop, even if convergence according to the accuracy criterion has not been reached.

Value

A list with three elements. The first two elements represent the position and velocity of the satellite at the target time, in the TEME (True Equator, Mean Equinox) frame of reference. Position values are in km, and velocity values are in km/s. Each of these two elements contains three values, corresponding to the X, Y and Z components of position and velocity in this order. The third element indicates the algorithm used to propagate the orbit (sgp4 or sdp4).

References

https://celestrak.com/NORAD/documentation/spacetrk.pdf http://www.celestrak.com/publications/aiaa/2006-6753/AIAA-2006-6753.pdf

Examples

```
# The following orbital parameters correspond to an object with NORAD catalogue
# number 24208 (Italsat 2) the 26th of June, 2006 at 00:58:29.34 UTC.

n0 <- 1.007781*((2*pi)/(1440))  # Multiplication by 2pi/1440 to convert to radians/min
e0 <- 0.002664 # mean eccentricity at epoch
i0 <- 3.8536*pi/180 # mean inclination at epoch in radians
M0 <- 48.3*pi/180 # mean anomaly at epoch in radians
omega0 <- 311.0977*pi/180 # mean argument of perigee at epoch in radians
OMEGA0 <- 80.0121*pi/180 # mean longitude of ascending node at epoch in radians
Bstar <- 1e-04 # drag coefficient
epochDateTime <- "2006-06-26 00:58:29.34"

# Calculation of the orbital period

2*pi/n0

# The period is higher than 225 min, and therefore the SDP4 model should be
# applied. Lets calculatethe position and velocity of the satellite 12 hours
# after epoch.

italsat_12h <- sgdp4(n0=n0, e0=e0, i0=i0, M0=M0, omega0=omega0, OMEGA0=OMEGA0,
                     Bstar=Bstar, initialDateTime=epochDateTime, targetTime=0)
italsat_12h$algorithm

# The SDP4 model was correctly chosen.
```

---

sgp4 *Propagate an orbital state vector with the SGP4 model*

Description

Given an orbital state vector of a satellite, applies the SGP4 model to propagate its orbit to the desired time point. This allows the calculation of the position and velocity of the satellite at different times, both before and after the time corresponding to the known state vector (referred to as "epoch"). Kepler's equation is solved through fixed-point integration. The SGP4 model can only accurately propagate the orbit of objects near Earth (with an orbital period shorter than 225 minutes, corresponding approximately to an altitude lower than 5877.5 km). For propagation of objects in deep space, the SDP4 model should be used, available through the sdp4 function. This implementation is based on the theory and implementation described in Space Track Report #3, and includes the corrections summarized in Revisiting Space Track Report #3.

Usage

```
sgp4(n0, e0, i0, M0, omega0, OMEGA0, Bstar, initialDateTime=NULL, targetTime,
     keplerAccuracy=10e-8, maxKeplerIterations=100)
```

Arguments

`n0`	Mean motion of the satellite at epoch in radians/min.
`e0`	Mean eccentricity of the orbit of the satellite at epoch. Eccentricity ranges from 0 (perfectly circular orbit) to 1 (parabolic trajectory).
`i0`	Mean orbital inclination of the satellite at epoch in radians.
`M0`	Mean anomaly of the satellite at epoch.
`omega0`	Mean argument of perigee of the satellite at epoch.
`OMEGA0`	Mean longitude of the ascending node of the satellite at epoch. Also known as right ascension of the ascending node.
`Bstar`	Drag coefficient of the satellite in units of (earth radii)^-1^. Bstar is an adjusted value of the ballistic coefficient of the satellite, and it indicates how susceptible it is to atmospheric drag.
`initialDateTime`	Optional date-time string in UTC indicating the time corresponding to the known state vector of the satellite. It must be provided if `targetTime` is provided as a date-time string.
`targetTime`	Time at which the position and velocity of the satellite should be calculated. It can be provided in two different ways: either as a number corresponding to the time in minutes counting from epoch at which the orbit should be propagated, or as a date-time string in UTC, in which case the date-time string for epoch must be provided through `initialDateTime`.
`keplerAccuracy`	Accuracy to consider Kepler´s equation solved. If two consecutive solutions differ by a value lower than this accuracy, integration is considered to have converged.
`maxKeplerIterations`	Maximum number of iterations after which fixed-point integration of Kepler's equation will stop, even if convergence according to the accuracy criterion has not been reached.

Value

A list with three elements. The first two elements represent the position and velocity of the satellite at the target time, in the TEME (True Equator, Mean Equinox) frame of reference. Position values are in km, and velocity values are in km/s. Each of these two elements contains three values, corresponding to the X, Y and Z components of position and velocity in this order. The third element indicates the algorithm used to propagate the orbit (sgp4).

References

https://celestrak.com/NORAD/documentation/spacetrk.pdf http://www.celestrak.com/publications/aiaa/2006-6753/AIAA-2006-6753.pdf

Examples

```
# The following orbital parameters correspond to an object with NORAD catalogue
# number 88888 the 1st of October, 1980 at 23:41:24 UTC.
```

```
n0 <- 16.05824518*((2*pi)/(1440))  # Multiplication by 2pi/1440 to convert to radians/min
e0 <- 0.0086731 # mean eccentricity at epoch
i0 <- 72.8435*pi/180 # mean inclination at epoch in radians
M0 <- 110.5714*pi/180 # mean anomaly at epoch in radians
omega0 <- 52.6988*pi/180 # mean argument of perigee at epoch in radians
OMEGA0 <- 115.9689*pi/180 # mean longitude of ascending node at epoch in radians
Bstar <- 0.66816e-4 # drag coefficient

# Calculation of the orbital period

2*pi/n0

# The period is lower than 225 min, and therefore the SGP4 model is valid.
# Lets calculate the position and velocity of the satellite 40 minutes after
# epoch

new_state <- sgp4(n0=n0, e0=e0, i0=i0, M0=M0, omega0=omega0, OMEGA0=OMEGA0,
                  Bstar=Bstar,targetTime = 40)
new_state
```

`TEMEtoECEF` *Convert coordinates from TEME to ECEF*

Description

The TEME (True Equator, Mean Equinox) frame of reference is an Earth-centered inertial coordinate frame, where the origin is placed at the center of mass of Earth and the coordinate frame is fixed with respect to the stars (and therefore not fixed with respect to the Earth surface in its rotation). The coordinates and velocities calculated with the SGP4 and SDP4 models are in the TEME frame of reference. This function converts positions and velocities in TEME to the ECEF (Earth Centered, Earth Fixed) frame of reference. In the ECEF frame, the origin is also placed at the center of mass of Earth, but the frame rotates with respect to the stars to remain fixed with respect to the Earth surface as it rotates.

Usage

```
TEMEtoECEF(position_TEME, velocity_TEME, dateTime)
```

Arguments

`position_TEME`	Vector with the X, Y and Z components of the position of an object in TEME frame, in m.
`velocity_TEME`	Vector with the X, Y and Z components of the velocity of an object in TEME frame, in m/s.
`dateTime`	Date-time string with the date and time in UTC corresponding to the provided position and velocity vectors.

Value

A list with two elements representing the position and velocity of the satellite in the ECEF (Earth Centered, Earth Fixed) frame of reference. Position values are in m, and velocity values are in m/s. Each of the two elements contains three values, corresponding to the X, Y and Z components of position and velocity in this order.

References

https://celestrak.com/columns/v04n03/#FAQ01

Examples

```
# The following orbital parameters correspond to an object with NORAD catalogue
# number 24208 (Italsat 2) the 26th of June, 2006 at 00:58:29.34 UTC.

n0 <- 1.007781*((2*pi)/(1440))  # Multiplication by 2pi/1440 to convert to radians/min
e0 <- 0.002664 # mean eccentricity at epoch
i0 <- 3.8536*pi/180 # mean inclination at epoch in radians
M0 <- 48.3*pi/180 # mean anomaly at epoch in radians
omega0 <- 311.0977*pi/180 # mean argument of perigee at epoch in radians
OMEGA0 <- 80.0121*pi/180 # mean longitude of ascending node at epoch in radians
Bstar <- 1e-04 # drag coefficient
epochDateTime <- "2006-06-26 00:58:29.34"

# Lets calculate the position and velocity of the satellite 1 day later

state_1day_TEME <- sgdp4(n0=n0, e0=e0, i0=i0, M0=M0, omega0=omega0, OMEGA0=OMEGA0,
                         Bstar=Bstar, initialDateTime=epochDateTime, targetTime=1440)

# We can now convert the results in TEME frame to ECEF frame, previously
# multiplying by 1000 to convert the km output of sgdp4 to m

state_1day_ECEF <- TEMEtoECEF(state_1day_TEME$position*1000,
                              state_1day_TEME$velocity*1000,
                              "2006-06-27 00:58:29.34")
```

TEMEtoGCRF — *Convert coordinates from TEME to GCRF*

Description

The TEME (True Equator, Mean Equinox) and GCRF (Geocentric Celestial Reference Frame) are both ECI frames of reference, i.e., Earth-centered inertial coordinate frames, where the origin is placed at the center of mass of Earth and the coordinate frame is fixed with respect to the stars (and therefore not fixed with respect to the Earth surface in its rotation).

The difference between the two resides in the fact that in the GCRF frame, the X-axis and Z-axis are aligned respectively with the mean equinox and rotation axis of Earth at 12:00 Terrestrial Time on the 1st of January, 2000, while in the TEME frame they are aligned with the mean equinox and rotation axis at the time of the corresponding TLE. Due to the change of the direction of the vernal equinox and the rotation axis over time, coordinates in the two frames differ slightly.

This function requires the `asteRiskData` package, which can be installed by running `install.packages('asteRiskData',`

Usage

```
TEMEtoGCRF(position_TEME, velocity_TEME, dateTime)
```

Arguments

`position_TEME`	Vector with the X, Y and Z components of the position of an object in TEME frame, in m.
`velocity_TEME`	Vector with the X, Y and Z components of the velocity of an object in TEME frame, in m/s.
`dateTime`	Date-time string with the date and time in UTC corresponding to the provided position and velocity vectors.

Value

A list with two elements representing the position and velocity of the satellite in the ECEF (Earth Centered, Earth Fixed) frame of reference. Position values are in m, and velocity values are in m/s. Each of the two elements contains three values, corresponding to the X, Y and Z components of position and velocity in this order.

References

https://celestrak.com/columns/v04n03/#FAQ01

Examples

```
if(requireNamespace("asteRiskData", quietly = TRUE)) {
# The following orbital parameters correspond to an object with NORAD catalogue
# number 24208 (Italsat 2) the 26th of June, 2006 at 00:58:29.34 UTC.

n0 <- 1.007781*((2*pi)/(1440))  # Multiplication by 2pi/1440 to convert to radians/min
e0 <- 0.002664 # mean eccentricity at epoch
i0 <- 3.8536*pi/180 # mean inclination at epoch in radians
M0 <- 48.3*pi/180 # mean anomaly at epoch in radians
omega0 <- 311.0977*pi/180 # mean argument of perigee at epoch in radians
OMEGA0 <- 80.0121*pi/180 # mean longitude of ascending node at epoch in radians
Bstar <- 1e-04 # drag coefficient
epochDateTime <- "2006-06-26 00:58:29.34"

# Lets calculate the position and velocity of the satellite 1 day later

state_1day_TEME <- sgdp4(n0=n0, e0=e0, i0=i0, M0=M0, omega0=omega0, OMEGA0=OMEGA0,
                         Bstar=Bstar, initialDateTime=epochDateTime, targetTime=1440)

# We can now convert the results in TEME frame to GCRF frame, previously
# multiplying by 1000 to convert the km output of sgdp4 to m
```

```
state_1day_GCRF <- TEMEtoGCRF(state_1day_TEME$position*1000,
                              state_1day_TEME$velocity*1000,
                              "2006-06-27 00:58:29.34")
}
```

TEMEtoLATLON *Convert coordinates from TEME to geodetic latitude, longitude and altitude*

Description

The TEME (True Equator, Mean Equinox) frame of reference is an Earth-centered inertial coordinate frame, where the origin is placed at the center of mass of Earth and the coordinate frame is fixed with respect to the stars (and therefore not fixed with respect to the Earth surface in its rotation). The coordinates and velocities calculated with the SGP4 and SDP4 models are in the TEME frame of reference. This function converts position in TEME to geodetic latitude, longitude and altitude, which can be considered to be a non-inertial, Earth-centered frame of reference.

Usage

```
TEMEtoLATLON(position_TEME, dateTime, degreesOutput=TRUE)
```

Arguments

`position_TEME`	Vector with the X, Y and Z components of the position of an object in TEME frame, in m.
`dateTime`	Date-time string with the date and time in UTC corresponding to the provided position vector.
`degreesOutput`	Logical indicating if the output should be in sexagesimal degrees. If `degreesOutput=FALSE`, the output will be in radians.

Value

A vector with three elements, corresponding to the latitude and longitude in degrees and the altitude in m.

References

https://arc.aiaa.org/doi/10.2514/6.2006-6753

Examples

```
# The following orbital parameters correspond to an object with NORAD catalogue
# number 24208 (Italsat 2) the 26th of June, 2006 at 00:58:29.34 UTC.

n0 <- 1.007781*((2*pi)/(1440))  # Multiplication by 2pi/1440 to convert to radians/min
e0 <- 0.002664 # mean eccentricity at epoch
```

```
i0 <- 3.8536*pi/180 # mean inclination at epoch in radians
M0 <- 48.3*pi/180 # mean anomaly at epoch in radians
omega0 <- 311.0977*pi/180 # mean argument of perigee at epoch in radians
OMEGA0 <- 80.0121*pi/180 # mean longitude of ascending node at epoch in radians
Bstar <- 1e-04 # drag coefficient
epochDateTime <- "2006-06-26 00:58:29.34"

# Lets calculate the position and velocity of the satellite 1 day later

state_1day_TEME <- sgdp4(n0=n0, e0=e0, i0=i0, M0=M0, omega0=omega0, OMEGA0=OMEGA0,
                         Bstar=Bstar, initialDateTime=epochDateTime, targetTime=1440)

# We can now convert the results in TEME frame to geodetic latitude, longitude
# and altitude, previously multiplying by 1000 to convert the km output of
# sgdp4 to m

state_1day_geodetic <- TEMEtoLATLON(state_1day_TEME$position*1000,
                                    "2006-06-27 00:58:29.34")
```

Index