astro Documentation

Release 0.1.10

Shankar Kulumani

Dec 10, 2018

Contents:

CHAPTER 1

About Me

CHAPTER 2

Astro Package

2.1 constants module

Astrodynamic constants

This module will load a wide variety of WGS84 derived constants as module attributes. These constants may be referenced in subsequent scripts and functions. These variables are case specific and must be noted.

Notes

To use these constants in your scripts/functions, simply import this module

```
from astro import constants
```

Then use any of the attributes as:

```
deg2rad = constants.deg2rad
```

Constants are defined for each planetary body in the solar system, as well as many moons. Each body is defined as a `namedtuple` with all of the parameters as member attributes.

References

class astro.constants.**Body**(*rot_per*, *radius*, *mu*, *mass*, *orbit_sma*, *orbit_per*, *orbit_ecc*, *orbit_inc*, *p*)

Bases: `tuple`

mass
Alias for field number 3

mu
Alias for field number 2

orbit_ecc
Alias for field number 6

orbit_inc
Alias for field number 7

orbit_per
Alias for field number 5

orbit_sma
Alias for field number 4

p
Alias for field number 8

radius
Alias for field number 1

rot_per
Alias for field number 0

class astro.constants.**Earth**(*rot_per*, *radius*, *mu*, *mass*, *orbit_sma*, *orbit_per*, *orbit_ecc*, *orbit_inc*, *p*, *omega*, *sidepersol*, *radperday*, *flat*, *eesqrd*, *ee*, *J2*, *J3*, *J4*)

Bases: tuple

J2
Alias for field number 15

J3
Alias for field number 16

J4
Alias for field number 17

ee
Alias for field number 14

eesqrd
Alias for field number 13

flat
Alias for field number 12

mass
Alias for field number 3

mu
Alias for field number 2

omega
Alias for field number 9

orbit_ecc
Alias for field number 6

orbit_inc
Alias for field number 7

orbit_per
Alias for field number 5

orbit_sma
Alias for field number 4

p
Alias for field number 8

radius
Alias for field number 1

radperday
Alias for field number 11

rot_per
Alias for field number 0

sidepersol
Alias for field number 10

astro.constants.**halfpi = 1.5707963267948966**
float – pi/2 constant

astro.constants.**twopi = 6.283185307179586**
float – 2 times pi

2.2 geodetic module

2.3 kepler module

Keplerian method to do astrodynamics

astro.kepler.**a2n**(*a*, *mu*)
Convert semi major axis to mean motion

astro.kepler.**anom2nu**(*E*, *ecc*)
Calculate true anomaly given eccentric anomaly for all orbits

(nu) = anom2nu(E,ecc)

Inputs:

- ecc - eccentricity of orbit 0 < ecc < inf
- **E - (elliptical/parabolic/hyperbolic) eccentric anomaly in rad** 0 < E < 2*pi

Outputs:

- nu - true anomaly in rad -2*pi < nu < 2*pi

Dependencies:

- numpy - we are lost without numpy
- kinematics.attitude.normalize - normalize angles

Notes

This function is valid for all orbit types.

Author:

- **Shankar Kulumani 23 Nov 2017**
  - Add function in python

References

- AAE532 notes
- Vallado 3rd Ed

astro.kepler.**coe2rv**(*p_in*, *ecc_in*, *inc_in*, *raan_in*, *arg_p_in*, *nu_in*, *mu*)

Purpose:

- **Convert the classical orbital elements (COEs) to position (R)** and velocity (V) vectors.

[R_ijk,V_ijk,R_pqw,V_pqw] = coe2rv(p,ecc,inc,raan,arg_p,nu,mu,arglat,truelon,lonper)

Inputs:

- p - semi-major axis (km)
- ecc - eccentricity
- raan - right acsension of the ascending node (rad) 0 < raan <2*pi
- inc - inclination (rad) 0 < inc < pi
- arg_p - argument of periapsis (rad) 0 < arg_p < 2*pi
- nu - true anomaly (rad) 0 < nu < 2*pi
- mu - gravitational parameter of central body (km^3/sec^2).
- arglat - argument of latitude(CI) rad 0 <arglat < 2*pi
- truelon - true longitude (CE) rad 0 < truelon < 2*pi
- lonper - longitude of periapsis rad 0 < lonper < 2*pi

Outpus:

- R_ijk - position vector in inertial frame (km)
- V_ijk - velocity vector in inertial frame (km/sec)
- R_pqw - position vecotr in perifocal frame (km)
- V_pqw - velocity vector in perifocal frame (km/sec)

Dependencies:

- ROT1 - elementary rotation about first axis
- ROT2 - elementary rotation about second axis
- ROT3 - elementary rotation about third axis

Author:

- **Shankar Kulumani 18 Aug 2012**
  - revised from code written at USAFA Fall 2007
- **Shankar Kulumani 18 Sept 2012**
  - modified rotation matrix from PQW to ECI
- **Shankar Kulumani 29 Sept 2012**
  - modified input to take semi-latus rectum to allow calculation

  for all orbit types
- **Shankar Kulumani 7 Dec 2014**

- loop for vector inputs

- **Shankar Kulumani 7 Dec 2016**
  - Convert to python and remove vectorized inputs

astro.kepler.**conic_orbit**(*p*, *ecc*, *inc*, *raan*, *arg_p*, *nu_i*, *nu_f*, *mu=398600.5*)

Plot conic orbit

Purpose:

- Uses the polar conic equation to plot a conic orbit

state_eci, state_pqw, state_lvlh, state_sat_eci, state_sat_pqw, state_sat_lvlh = conic_orbit(p,ecc,inc,raan,arg_p,nu_i,nu_f)

Inputs:

- p - semi-major axis (km)
- ecc - eccentricity
- raan - right acsension of the ascending node (rad) 0 < raan <

2*pi - inc - inclination (rad) 0 < inc < pi - arg_p - argument of periapsis (rad) 0 < arg_p < 2*pi - nu_i - initial true anomaly (rad) 0 < nu < 2*pi - nu_f - final true anomaly (rad) 0 < nu < 2*pi - mu - gravitational paramter of central body (km^3/sec^2)

Outputs: state_eci : (1000, 6) numpy array of satellite orbit in inertial frame state_pqw : (1000, 6) numpy array of satellite orbi in perifocal frame state_lvlh : (1000, 6) numpy array of satellite orbit in local veritical local horizontal frame state_sat_eci : (6,) numpy array of satellite state in inertial frame state_sat_pqw : (6,) numpy array of satellite state in perifocal frame state_sat_lvlh : (6,) numpy array of satellite state in local veritical and local horizontal fram

the state is defined as 6 elements state[0:3] - position in reference frame in kilometer state[3:6] - velocity in reference frame in kilometer/second

Dependencies:

- ROT1,ROT2,ROT3 - principle axis rotation matrices

Author:

- **Shankar Kulumani 1 Dec 2012**
  - list revisions
- **Shankar Kulumani 5 Dec 2014**
  - added outputs for orbit gui functions
- **Shankar Kulumani 5 Oct 2017**
  - modify to include velocity and all three reference frames

References

- AAE532
- MAE3145

astro.kepler.**elp_orbit_el**(*p*, *ecc*, *inc*, *raan*, *arg_p*, *nu*, *mu*)

Elliptical Orbit Characteristics/Elements

Purpose:

- Calculates elliptical orbital parameters using conic equations

Inputs:

- p - semi-major axis (km)
- ecc - eccentricity
- raan - right acsension of the ascending node (rad) 0 < raan <

2*pi - inc - inclination (rad) 0 < inc < pi - arg_p - argument of periapsis (rad) 0 < arg_p < 2*pi - nu - true anomaly (rad) 0 < nu < 2*pi - mu - gravitational parameter of central body (km^3/sec^2). - arglat - argument of latitude(CI) rad 0 <arglat < 2*pi - truelon - true longitude (CE) rad 0 < truelon < 2*pi - lonper - longitude of periapsis rad 0 < lonper < 2*pi

Outputs:

- a - semi-major axis in km
- h - magnitude of angular momentum vector in km^2/sec
- period - period of orbit in seconds
- sme - specific mechanical energy of orbit in km^2/sec^2
- r_per - radius of periapsis in km
- r_apo - radius of apoapsis in km
- r - current radius in km
- v - current velocity in km/sec
- v_circ - equivalent circular orbit velocity at current radius in

km/sec - v_esc - escape speed at current radius km/sec - fpa - flight path angle in rad 0 < fpa < pi/2 - E - eccentric anomaly in rad 0 < E < 2*pi - M - mean anomaly in rad 0 < M < 2*pi - n - mean motion in 1/sec

Dependencies:

- fpa_solve - find flight path angle
- ecc_anomaly - find eccentric and mean anomaly given true

anomaly - aae532_constants - AAE532 class constants - ROT3 - simple rotation about third axis - use orbit_el as driver function to print results to screen

Author:

- **Shankar Kulumani 16 Sept 2012**
  - used code from AAE532 PS4
- **Shankar Kulumani 19 Sept 2012**
  - added escape speed
  - added print capability
  - added constants
  - added mean motion
  - added lvlh and pqw vectors
- **Shankar Kulumani 28 June 2017**
  - Convert to Python for MAE3145 class

References

- AAE532

- Any astrodynamics book/internet

astro.kepler.**fg_propogate**(*r_old*, *v_old*, *nu_old*, *nu_new*, *p*, *ecc*, *mu*)

F and G relationship propogator

Purpose:

- Find new position and velocity vectors in inertial frame using f

and g relationships

[r_new v_new f g f_dot g_dot delta_nu] = fandg_nu(r_old,v_old,nu_old,nu_new,p,ecc,mu)

Inputs:

- r_old - position vector in inertial frame in km
- v_old - velocity vecotr in inertial frame in km/sec
- nu_old - true anomaly at initial condition in rad
- nu_new - true anomaly at final position in rad
- p - semi-latus rectum of orbit in km
- ecc - eccentricity of orbit
- mu - gravitational parameter of central body in km^3/sec^2

Outputs:

- r_new - new position vector (3,)
- v_new - new velocity vector (3,)
- f - f function value
- g - g function value
- f_dot - fdot value
- g_dot - gdot value
- delta_nu - change in true anomaly

Dependencies:

- None

Author:

- **Shankar Kulumani 13 Oct 2012**
  - list revisions
- **Shankar Kulumani 13 December 2017**
  - moving to python

References

- AAE532 LSN 14 notes
- AAE532_PS7.pdf

astro.kepler.**fg_velocity**(*r1*, *r2*, *delta_nu*, *p*, *mu*)

F and G function using delta true anomaly

Purpose:

- Solves for new velocity vectors using f and g functions adn a

change in true anomaly

[v1 v2 f g f_dot g_dot] = fg_nu(r1,r2,dnu,p,mu)

Inputs:

- r1 - initial position vector (1x3 or 3x1) in km
- r2 - final position vector (same size as r1) in km
- dnu - delta true anomaly between r1 and r2
- p - semi-parameter of orbit in km
- mu - gravitational parameter in km^3/sec^2

Outputs:

- v1 - initial velocity vector in km/sec
- v2 - final velocity vector in km/sec
- f - f function
- g - g function
- f_dot - f dot function
- g_dot - g dot function

Dependencies:

- none

Author:

- **Shankar Kulumani 5 Nov 2012**
  - list revisions

References

- AAE532 LSN 18

astro.kepler.**fpa_solve**(*nu*, *ecc*)

Calculate flight path angle

Inputs:

- nu - true anomaly of orbit -2*pi < nu < 2*pi
- ecc - eccentricity of orbit 0 < ecc < inf

Outputs:

- fpa - flight path angle in rad -pi < fpa < pi

Dependencies:

- none

Author:

- **Shankar Kulumani 15 Sept 2012**
  - created for AAE532 PS4
- **Shankar Kulumani 29 Sept 2012**
  - modified for diferennt orbit types

References

- AAE532 Notes
- Vallado 3rd Ed pg 113

astro.kepler.**hne_vec**(*r*, *v*, *mu*)

Compute fundamental vectors associated with orbit

This will compute the angular momentum, h, nodal, n, and eccentricity vector, e, for a Keplerian two-body orbit.

Parameters **r** (*array_like and type*) – Description of the variable

Returns **describe** – Explanation of return value named describe

Return type type

Other Parameters **only_seldom_used_keywords** (*type*) – Explanation of this parameter

Raises BadException – Because you shouldn't have done that.

See also:

other_func() Other function that this one might call

Notes

You may include some math:

$$X(e^{j\omega}) = x(n)e^{-j\omega n}$$

Shankar Kulumani GWU skulumani@gwu.edu

References

Cite the relevant literature, e.g. [1]_. You may also cite these references in the notes section above.

Bell Labs Technical Journal 28.4 (1949): 656-715

Examples

An example of how to use the function

```
>>> a = [1, 2, 3]
>>> print [x + 3 for x in a]
[4, 5, 6]
>>> print "a\n\nb"
a
b
```

astro.kepler.**hyp_orbit_el**(*p*, *ecc*, *inc*, *raan*, *arg_p*, *nu*, *mu*)

Hyperbolic Orbit Characteristics/Elements

Purpose:

- calculates orbital parameters for a hyperbolic orbit

[a, v_inf, b, sme, flyby, nu_inf, h, fpa, r_per, r_ijk, v_ijk, r_pqw, v_pqw, r_lvlh, v_lvlh, r, v, v_circ, v_esc, H, M_H, n] = hyp_orbit_el(p, ecc, inc, raan, arg_p, nu, mu)

Inputs:

-

Outputs:

- List/describe outputs of function

Dependencies:

- coe2rv.m - convert COE to position and velocity vector

Author:

- **Shankar Kulumani 29 Sept 2012**
  - modified outputs
- **Shankar Kulumani 5 September 2017**
  - convert to Python for MAE3145

References

- AAE532 and any astrodynamics book

`astro.kepler.`**`hyp_per2sma`**(*rp*, *ecc*)

Convert periapsis to semimajor axis for hyperbolic orbits

Determine semi-major axis and semi-latus rectum for hyperbolic orbits

Parameters

- **rp** (*float*) – Periapsis distance in kilometers or other distance unit
- **ecc** (*float*) – Eccentricty of orbit - should be greater than 1

Returns

- **a** (*float*) – Semimajor axis in kilometers
- **p** (*float*) – Semilatus rectum in same units as input distance
- *Author*
- ——
- *Shankar Kulumani GWU skulumani@gwu.edu*

References

Cite the relevant literature, e.g. [1]_. You may also cite these references in the notes section above.

Bell Labs Technical Journal 28.4 (1949): 656-715

`astro.kepler.`**`kepler_eq_E`**(*M_in*, *ecc_in*)

Solve Kepler's Equation for all orbit types

(E, nu, count) = kepler_eq_E(M, ecc)

Purpose:

- This function solves Kepler's equation for eccentric anomaly

given a mean anomaly using a newton-rapson method.

- Will work for elliptical/parabolic/hyperbolic orbits

Inputs:

- M - mean anomaly in rad -2*pi < M < 2*pi
- ecc - eccentricity 0 < ecc < inf

Outputs:

- E - eccentric anomaly in rad 0 < E < 2*pi
- nu - true anomaly in rad 0 < nu < 2*pi
- count - number of iterations to converge

Dependencies:

- numpy - everything needs numpy
- kinematics.attitude.normalize - normalize an angle

Author:

- **Shankar Kulumani 15 Sept 2012**
  - rewritten from code from USAFA
  - solve for elliptical orbits add others later
- **Shankar Kulumani 29 Sept 2012**
  - added parabolic/hyperbolic functionality
- **Shankar Kulumani 7 Dec 2014**
  - added loop for vector inputs
- **Shankar Kulumani 2 Dec 2016**
  - converted to python and removed the vector inputs

References

- USAFA Astro 321 LSN 24-25
- Vallado 3rd Ed pg 72

`astro.kepler.`**`n2a`**(*n*, *mu*)

Convert mean motion to semi major axis

`astro.kepler.`**`nu2anom`**(*nu*, *ecc*)

Calculates the eccentric and mean anomaly given eccentricity and true anomaly

(E, M) = ecc_anomaly(nu,ecc)

Inputs:

- nu - true anomaly in rad -2*pi < nu < 2*pi
- ecc - eccentricity of orbit 0 < ecc < inf

Outputs:

- **E - (elliptical/parabolic/hyperbolic) eccentric anomaly in rad** 0 < E < 2*pi
- M - mean anomaly in rad 0 < M < 2*pi

Dependencies:

- numpy - we are lost without numpy
- kinematics.attitude.normalize - normalize angles

Notes

This function is valid for all orbit types.

Author:

- **Shankar Kulumani 20 Nov 2017**
  - only now realized I already implemented other orbit types
- **Shankar Kulumani 5 Dec 2016**
  - Convert to python
- **Shankar Kulumani 15 Sept 2012**
  - modified from USAFA code and notes from AAE532
  - only elliptical case will add other later
- **Shankar Kulumani 17 Sept 2012**
  - added rev check to reduce angle btwn 0 and 2*pi

References

- AAE532 notes
- Vallado 3rd Ed

`astro.kepler.`**`nu_solve`**(*p*, *e*, *r*)

Solve conic equation for true anomaly

nu, nu_neg = nu_solv(p, ecc, r)

Parameters

- **`p`** (*`float`*) – Semi parameter
- **`e`** (*`float`*) – eccentricity
- **`r`** (*`float`*) – radius of orbit

Returns

- **nu** (*float*) – True anomaly in radians
- **nu_neg** (*float*) – Negative true anomaly in radians
- *Author*
- ——
- *Shankar Kulumani GWU skulumani@gwu.edu*

`astro.kepler.`**`orbit_el`**(*p*, *ecc*, *inc*, *raan*, *arg_p*, *nu*, *mu*, *print_flag=False*)

Orbit Characteristics/Elements

Purpose:

- Calculates orbital parameters using conic equations

orbit_el(p,ecc,inc,raan,arg_p,nu,mu,print_flag)

Inputs:

- a - semi-major axis in km
- ecc - eccentricity of orbit

- mu - gravitational parameter in km^3/sec^2
- nu - true anomaly in rad 0 < nu < 2*pi
- print_flag - 'true' or 'false' to print outputs to screen

Outputs:

- none - prints data to screen

Dependencies:

- elp_orbit_el.m - elliptical orbit elements
- hyp_orbit_el.m - hypberbolic orbit elements
- par_orbit_el.m - parabolic orbit elements

Author:

- **Shankar Kulumani 16 Sept 2012**
  - used code from AAE532 PS4
- **Shankar Kulumani 19 Sept 2012**
  - added escape speed
  - added print capability
  - added constants
  - added mean motion
  - added lvlh and pqw vectors
- **Shankar Kulumani 27 Sept 2012**
  - modifying to add hyperbolic orbit capability
  - moved elliptical stuff to elp_orbit_el.m
  - removed outputs
- **Shankar Kulumani 29 Sept 2012**
  - modified fprintf commands to make it more like STK
- **Shankar Kulumani 5 September 2017**
  - modify for Python for MAE3145

References

- AAE532 Notes

astro.kepler.**par_orbit_el**(*p*, *ecc*, *inc*, *raan*, *arg_p*, *nu*, *mu*)

astro.kepler.**perapo2aecc**(*r_per*, *r_apo*)

Apoapsis/Periapsis to Semi-major axis and Eccentricity

a, p, ecc = perapo2aecc(r_per,r_apo)

Inputs: - r_per - periapsis distance in km - r_apo - apoapsis distance in km

Outputs: - a - semi-major axis in km - ecc - eccentricity

Dependencies: - none

Author: - Shankar Kulumani 19 Sept 2012 - list revisions - Shankar Kulumani 31 Oct 2012 - added semi-latus rectum - Shankar Kulumani 2 Oct 2017

- convert to Python

References - AAE532 Notes/PS5

astro.kepler.**period2sma**(*period*, *mu*)
Convert period to semi major axis

astro.kepler.**rv2coe**(*r*, *v*, *mu*)
Position and Velocity vectors to classical orbital elements

[p,a,ecc,inc,raan,arg_p,nu,m,arglat,truelon,lonper] = rv2coe(r,v, mu)

Inputs:

- r - position vector in inertial frame (km)
- v - velocity vector in inertial frame (km/sec)
- mu - gravitational parameter of central body (km^3/sec^2)

Outputs:

- p - semi-major axis (km)
- ecc - eccentricity
- raan - right acsension of the ascending node (rad) 0 < raan <

2*pi - inc - inclination (rad) 0 < inc < pi - arg_p - argument of periapsis (rad) 0 < arg_p < 2*pi - nu - true anomaly (rad) 0 < nu < 2*pi - m - mean anomaly in rad - arglat - argument of latitude(CI) rad 0 <arglat < 2*pi - truelon - true longitude (CE) rad 0 < truelon < 2*pi - lonper - longitude of periapsis rad 0 < lonper < 2*pi

Dependencies:

- numpy - everything is dependent on numpy
- nu2anom - convert true anomaly to eccentric and mean anomaly

Author:

- **Shankar Kulumani 30 Sept 2012**
  - used old USAFA code and AAE532
- **Shankar Kulumani 5 September 2017**
  - convert to Python for use in MAE3145

References

- AAE532 Notes
- Vallado 3rd Edition

astro.kepler.**semilatus_rectum**(*a*, *ecc*)
Compute the semilatus rectum

Given the semimajor axis and eccentricty, this will compute the semilatus rectum.

Parameters

- **a** (*float*) – Semimajor axis (distance unit)
- **ecc** (*float*) – Eccentricity of orbit (unitless)

Returns **p** – Semilatus rectum in the same units as a

Return type float

Notes

$$p = a(1 - e^2)$$

Shankar Kulumani GWU skulumani@gwu.edu

References

astro.kepler.**tof_delta_t**(*p*, *ecc*, *mu*, *nu_0*, *delta_t*)

Use time of fligt to compute future true anomaly for eccentric orbits

Calculate change in orbital position (true anomaly) given a delta_t change in time

Inputs:

- p - semi-latus rectum in km
- ecc - eccentricity of orbit 0 < ecc < 1
- mu - gravitational parameter
- nu_0 - initial true anomaly in rad 0 < nu_0 < 2*pi
- delta_t - change in time in seconds

Outputs:

- nu_f - true anomaly after delta_t in rad 0 < nu_f < 2*pi

Dependencies:

- ecc_anomaly.m - calculates eccentric and mean anomaly from true

anomaly - kepler_eq_E.m - solves for eccentric anomaly and true anomaly given a mean anomaly

Author:

- **Shankar Kulumani 15 Sept 2012**
  - written using USAFA code and AAE532 PS4 homework
- **Shankar Kulumani 19 Sept 2012**
  - added E and M outputs
- **Shankar Kulumani 1 Oct 2012**
  - added semi-latus rectum input
- **Shankar Kulumani 9 Oct 2017**
  - now in Python

References

- Vallado 3rd Edition
- AAE532 Notes

astro.kepler.**tof_nu**(*p*, *ecc*, *nu_1*, *nu_2*, *mu=398600.5*)

Calculate TOF between two known true anomaly positions

Inputs:

- p - semi-major axis (or semi-parameter for parabolic) in km
- ecc - eccentricity 0 < ecc < 1
- mu - gravitational parameter of central body in km^3/sec^2
- nu_0 - initial true anomaly in rad 0 < nu_0 < 2*pi
- nu_f - final true anomaly in rad 0 < nu_f < 2*pi

Outputs:

- tof - time of flight in seconds

Dependencies:

- ecc_anomaly - calculates eccentric anomaly from true anomlay

Author:

- **Shankar Kulumani 16 Sept 2012**
  - uses code from AAE532 PS4 in function form
- **Shankar Kulumani 29 Sept 2012**
  - added eccentricity logic for different orbit types
- **Shankar Kulumani 1 Oct 2012**
  - added semi-latus rectum instead of semi-major axis
- **Shankar Kulumani 11 Oct 2017**
  - convert to Python

Note: Currently only supports elliptical orbits. Need to modify

---

References

- AAE532 Notes
- Vallado 3rd Edition
- Bate,Mueller,White

2.4 astro

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

S

T