EPM UCC

Versión 12/05/2020

javi martinez

13 de mayo de 2020

Índice general

1

Contents:

1.1 Getting Started

El codigo esta optimizado para que funcione en linux(50 veces mas rapido que en windows), para ello el metodo mas sencillo es instalar WSL en windows y utilizarlo con Visual Studio Code

-https://pbpython.com/wsl-python.html

Hay va un link que os explicara como hacerlo todo

1.1.1 Instalacion

He incluido dos ficheros:

- requirements.txt
- environment.yml

ambos sirven para crear un environment con conda o con pip con los archivos necesarios para que funcione (y alguno mas, no hice limpieza sorry)

Si quieres utilizar el primero .. code-block:: python

```
pip install -r requirements.txt
```

ó

para el environment.yml .. code-block:: python

```
conda env create -f environment.yml
```

1.1.2 Uso

Para utilizar el modelo es bastante facil solo es necesario el uso de SatelliteV4.py todos lo necesario esta explicado en el archivo tambien esta inlcuido el visual.ypnb que permite ver la imagen del satelite CAD con los ejes.

1.2 src

1.2.1 src package

Subpackages

src.data package

Submodules

src.data.OrbitasV4 module

Orbitas Basicamente un monton de utilidades creadas para el programa principal

returns [description]

rtype [type]

src.data.OrbitasV4.**EclipseLocator**(Orbit)
Declara si el satelite esta en eclipse o no, con la posicion y el tiempo

Parámetros **Orbit** (poliastro.Orbit) – La orbita del satelite

Devuelve True si esta en eclipse, False sino lo esta

Tipo del valor devuelto Bool

src.data.OrbitasV4.**SolarAngle_a_VNB**(Orbit)

SolarRay_to_VNB Cambio del vector Sol en GCRSS a ejes satelite VNB en la epoch de la posicion del satelite https://ai-solutions.com/_freeflyeruniversityguide/attitude_reference_frames.htm https://help.agi.com/stk/11.0.1/Content/gator/eq-coordsys.htm

Parámetros **Orbit** (poliastro Orbit) – La orbita del satelite definida con la clase poliastro.orbit

Devuelve Vector sol en GCRSS

Tipo del valor devuelto array(3,1)

src.data.OrbitasV4.**SolarRay_to_LVLH**(Orbit)

Cambio del vector Sol en GCRSS a ejes satelite LVLH en la epoch de la posicion del satelite https://ai-solutions.com/_freeflyeruniversityguide/attitude_reference_frames.htm

Parámetros **Orbit** (poliastro Orbit) – La orbita del satelite definida con la clase poliastro.orbit

Devuelve Vector sol en GCRSS

Tipo del valor devuelto np.array(3,1)

src.data.OrbitasV4.**propagate_fast_one_orbit**(a, ecc, inc, raan, nu, time_ini, argp_ini, EPM, iteraciones_orbita=100, num_orbitas=1)

propaga la orbita a traves de saltos en la posicion del satelite debido a ser mas rapido decla
de poliastro. se puede modificar para incluir las pertubaciones

Parámetros

- **a** (astropy.units.km) – Semieje mayor de la orbita
- **ecc** (astropy.units.deg) – excentricidad de la orbita
- **inc** (astropy.units.deg) – inclinacion de la orbita
- **raan** (astropy.units.deg) – Raan de la orbita
- **nu** (astropy.units.deg) – Anomalia verdadera de la orbita
- **time_ini** (Time) – Tiempo inicial de la orbita
- **argp_ini** (astropy.units.km) – argumento inicial de la orbita
- **EPM** (SolarPowerSystem) – Sistema de potencia
- **iteraciones_orbita** (int, optional) – Numero de iteraciones por orbita. Defaults to 100.
- **num_orbitas** (int, optional) – Numero de orbitas a propagar. Defaults to 1.

Devuelve ,iteraciones_orbita)) : Valores de potencia de cada cara Area_potencia (array(:,iteraciones_orbita)) : Area que produce potencia pro cada cara Ang (array(:,iteraciones_orbita)) : Angulo de incidencia del vector sol en la cara Angulo_giro (array(:,iteraciones_orbita)): Angulo de giro de la orbita

Tipo del valor devuelto w (array(

src.data.OrbitasV4.**propagate_orbit**(a, ecc, inc, raan, nu, time_ini, argp_ini, EPM, iteraciones_orbita=100, num_orbitas=1)

propaga numericamente la orbita a traves de poliastro. se puede modificar para incluir las pertubaciones

Parámetros

- **a** (astropy.units.km) – Semieje mayor de la orbita
- **ecc** (astropy.units.deg) – excentricidad de la orbita
- **inc** (astropy.units.deg) – inclinacion de la orbita
- **raan** (astropy.units.deg) – Raan de la orbita
- **nu** (astropy.units.deg) – Anomalia verdadera de la orbita
- **time_ini** (Time) – Tiempo inicial de la orbita
- **argp_ini** (astropy.units.km) – argumento inicial de la orbita
- **EPM** (SolarPowerSystem) – Sistema de potencia

- **iteraciones_orbita** (int, optional) – Numero de iteraciones por orbita. Defaults to 100.
- **num_orbitas** (int, optional) – Numero de orbitas a propagar. Defaults to 1.

Devuelve ,iteraciones_orbita)) : Valores de potencia de cada cara Area_potencia (array(:,iteraciones_orbita)) : Area que produce potencia pro cada cara Ang (array(:,iteraciones_orbita)) : Angulo de incidencia del vector sol en la cara Angulo_giro (array(:,iteraciones_orbita)): Angulo de giro de la orbita

Tipo del valor devuelto w (array(

src.data.OrbitasV4.**propagate_orbit2**(a, ecc, inc, raan, nu, time_ini, argp_ini, EPM, iteraciones_orbita=100, num_orbitas=1)

Intento de pasar de realizar los calculos en ejes satelites a hacerlos en GCRSS :param a: Semieje mayor de la orbita :type a: astropy.units.km :param ecc: excentricidad de la orbita :type ecc: astropy.units.deg :param inc: inclinacion de la orbita :type inc: astropy.units.deg :param raan: Raan de la orbita :type raan: astropy.units.deg :param nu: Anomalia verdadera de la orbita :type nu: astropy.units.deg :param time_ini: Tiempo inicial de la orbita :type time_ini: Time :param argp_ini: argumento inicial de la orbita :type argp_ini: astropy.units.km :param EPM: Sistema de potencia :type EPM: SolarPowerSystem :param iteraciones_orbita: Numero de iteraciones por orbita. Defaults to 100. :type iteraciones_orbita: int, optional :param num_orbitas: Numero de orbitas a propagar. Defaults to 1. :type num_orbitas: int, optional

Devuelve ,iteraciones_orbita)) : Valores de potencia de cada cara Area_potencia (array(:,iteraciones_orbita)) : Area que produce potencia pro cada cara Ang (array(:,iteraciones_orbita)) : Angulo de incidencia del vector sol en la cara Angulo_giro (array(:,iteraciones_orbita)): Angulo de giro de la orbita

Tipo del valor devuelto w (array(

src.data.OrbitasV4.**random_generator**(size=6, chars='ABCDEFGHIJKLMNOPQRSTUVWXYZ012345678

random_generator Generador de codigos aleatorios

Parámetros

- **size** (int, optional) – Numero de caracteres del codigo. Defaults to 6.
- **chars** (str, optional) – Tipo de sistema de simbolos. Defaults to string.ascii_uppercase+string.digits.

Devuelve Codigo random

Tipo del valor devuelto str

src.data.OrbitasV4.**to_LVLH**(Orbit)

Matriz de cambio de ejes de GCRSS a LVLH en la epoch de la posicion del satelite
https://ai-solutions.com/_freeflyeruniversityguide/attitude_reference_frames.htm

Parámetros **Orbit** (poliastro Orbit) – La orbita del satelite definida con la clase poliastro.orbit

Devuelve Matriz de cambio de base en una matriz 3x3

Tipo del valor devuelto np.array(3,3)

src.data.OrbitasV4.**to_VNB**(Orbit)

Matriz de cambio de ejes de GCRSS a VNB en la epoch de la posicion del satelite
https://ai-solutions.com/_freeflyeruniversityguide/attitude_reference_frames.htm

Parámetros **Orbit** (poliastro Orbit) – La orbita del satelite definida con la clase poliastro.orbit

Devuelve [Matriz de cambio de base en una matriz 3x3]

Tipo del valor devuelto np.array(3,3)

src.data.SatelitteActitud module

SatelitteActitud Sistema de actitud

class src.data.SatelitteActitud.**SatelitteActitud**(eje_de_spin, control)
Bases: object

Sistema de actitud

Parámetros **object** ([type]) – [description]

Apuntar_Sol(Sun_vector, Cara_principal)

src.data.SatelitteSolarPowerSystemV4 module

class src.data.SatelitteSolarPowerSystemV4.**SatelitteSolarPowerSystem**(direccion, SatelitteActitud, panel_despegable_dual=
Despegables_orientables

Bases: object

Add_prop_Panel(e)
Añade propiedades al panel

Parámetros **e** (Panel_Solar) – Panel Solar

Calculo_potencia(Sun_vector, WSun=1310)
Funcion general para llamar a las distintas funciones para calcular la potencia

Parámetros

- **Sun_vector** ([type]) – [description]
- **WSun** (int, optional) – [description]. Defaults to 1310.

Devuelve Potencia generada area_potencia (array(,n)) : Areas que generan potencia ang (array(,n)) : Angulo de incidencia del vector sol con las caras angulo_giro (array(,n)) : Angulo de giro del satelite

Tipo del valor devuelto W (array(,n))

n : numero de caras

apply_transform(matrix)
creada para hacer coincidir correctamente las caras aplica una transformacion al satelite y reinicia los nombres

Parámetros **matrix** (array(4,4)) – matriz de transformacion

caras_despegables()
Localiza los paneles despegables, es un metodo bastante dificil

Devuelve es el numero de las caras

Tipo del valor devuelto caras_despeables

cargar_modelo(direccion)

Parámetros **direccion** – string con la direcion del y con el tipo del archivo Ex. .STL, .OBJ, .PLY

Devuelve trimesh.mesh

celdas_activas(sun_vector)

Localiza las celdas activas de un mallado al buscarse los puntos donde golpearia un rayo en la malla des los puntos_sol

Parámetros **sun_vector** (array(,3)) – Vector sol

Devuelve El numero de triangulo que esta activo al ser golpeado por el sol

Tipo del valor devuelto index_tri [array(n)]

nombrar_caras()

Nombra las caras del modelo para poder utilizarlas se realiza al principio porque si se gira cambiara Simplemente nombra las caras con X, Y, Z

Devuelve devuelve el nombre de las caras de manera X, Y, Z

Tipo del valor devuelto name

posible_sombra()

buscar la cara mas cercana a los paneles que puede dar sombra :returns: numero de las caras que pueden tener sombra :rtype: sombra

power_panel_con_actitud(Sun_vector, WSun)

Obtiene la potencia producida por el satelite con actitud apuntando al sol

Parámetros

- **Sun_vector** (array(,3)) – Vector sol en LVLH
- **WSun** (float) – Potencia irradiada por el sol

Devuelve Potencia generada area_potencia (array(,n)) : Areas que generan potencia ang (array(,n)) : Angulo de incidencia del vector sol con las caras angulo_giro (array(,n)) : Angulo de giro del satelite

Tipo del valor devuelto W (array(,n))

n : numero de caras

power_panel_solar(index_tri, Sun_vector, WSun)

Obtiene la potencia producida por el satelite con actitud fija

Parámetros

- **(array(,** (index_tri) –)): Celdas activas por el rayo
- **Sun_vector** (array(,3)) – Vector sol en LVLH
- **WSun** (float) – Potencia irradiada por el sol

Devuelve Potencia generada area_potencia (array(,n)) : Areas que generan potencia ang (array(,n)) : Angulo de incidencia del vector sol con las caras

Tipo del valor devuelto W (array(,n))

n : numero de caras

puntos_sol()
Crea un conjunto de puntos de aquellos que darian sombra con los centros de los paneles

Devuelve Plano de puntos

Tipo del valor devuelto trimesh.mesh

separar_satelite()
Separa el satelite en mallas

Devuelve [description]

Tipo del valor devuelto [type]

visual()
Crea una imagen visual del satelite con unos ejes funciona muy bien en notebook y en linux tambien deberia de poder funcionar

Devuelve retoma una escena con los ejes

Tipo del valor devuelto (scene)

src.data.Satellite_Bateria module

Created on Mon Mar 2 18:30:07 2020

@author: ignacio.garciasanche

class src.data.Satellite_Bateria.**Bateria**
Bases: object

ATRIBUTOS

bateria()

bateria_carga(t, t0)
Cantidad de capacidad cargada

bateria_descarga(t, t0)

bateria_estado()

src.data.Satellite_Panel_Solar module

Paneles Solares : definimos tanto los atributos (tanto parametros inmutables como variables) y los estados que poseen los paneles solares.

Funcion Estado " psolar() " [input = none] output = descripcion de los paneles solares usados (tipo,masa,tension,intensidad)

Funcion Estado " psolar_estado() " [input = none] output = estado del panel, es decir, si esta generando, o no, potencia

class src.data.Satellite_Panel_Solar.**Panel_Solar**(name)
Bases: object

ATRIBUTOS

psolar()

psolar_estado()

Module contents

src.visualization package

Submodules

src.visualization.ConsumoEscenarios module

src.visualization.visualize module

Module contents

Module contents

2

Indices and tables

- genindex
- modindex
- search

Índice de Módulos Python

S