src.core

core.py

Реализация математического аппарата программы моделирования

- View Source

EARTH_FM = 398600441800000.0

Значение гравитационного параметра для Земли

```
def kepler2eci(
 a: float,
 e: float,
 i: float,
 long: float,
 periapsis: float,
 m0: float
) -> numpy.ndarray:
```

- View Source

Рассчитывает радиус-вектор и вектор скорости материальной точки на эллиптической орбите в проекциях на ИГЦСК по заданным кеплеровым элементам орбиты

Parameters

- **a**: Большая полуось в метрах
- **e**: Эксцентриситет (0 < `e` < 1)
- **i**: Наклонение в радианах
- **long**: Долгота восходящего узла в радианах
- **periapsis**: Аргумент перицентра в радианах
- **m0**: Средняя аномалия в радианах

Returns

Матрица 2x3, в первой строке проекции радиус-вектора в метрах, во второй – вектора скорости материальной точки в м/сек.

Ссылки на литературу

```
M.Eng. René Schwarz, "Keplerian Orbit Elements → Cartesian State Vectors (Memorandum № 1)",
https://downloads.rene-schwarz.com/download/M001-Keplerian_Orbit_Elements_to_Cartesian_State_Vectors.pdf
```

```
def get_state_relative(
 src_state: numpy.ndarray,
 distance: float,
 yaw: float,
 pitch: float
) -> tuple:
```

- View Source

Рассчитывает радиус-вектор точки, относительно заданной по известному расстоянию между ними и ориентации искомой точки

Искомый радиус-вектор получается путем сложения заданного радиус-вектора и вектора-слагаемого. Длина вектора-слагаемого равна заданному расстоянию `distance`. Ориентация вектора-слагаемого задается двумя последовательными поворотами вектора скорости исходной точки. Первый поворот на угол `yaw` осуществляется в плоскости, образованной радиус-вектором и вектором скорости исходной точки. Для второго поворота рассчитывается вектор-нормаль, как векторное произведение вектора скорости на радиус-вектор. Второй поворот на угол `pitch` осуществляется в плоскости

повернутого на угол `yaw` вектора скорости исходной точки и рассчитанного вектора-нормали.

Parameters

- **src_state**: Матрица 2x3, в первой строке проекции радиус-вектора в метрах, во второй – вектора скорости исходной точки в м/сек.
- **distance**: Расстояние между точками в метрах
- **yaw**: Перый угол поворота вектора скорости исходной точки в радианах
- **pitch**: Второй угол поворота вектора скорости исходной точки в радианах

Returns

Кортеж, первый элемент которого единичный вектор-слагаемое, а второй – матрица 2x3, в первой строке проекции радиус-вектора в метрах, во второй – вектора скорости искомой точки в м/сек.

def event_decoupling(t: float, y_vecs: numpy.ndarray, *constants) -> float: • View Source

Функция обработки события "расстыковки" между РН и КА при численном решении системы ДУ динамики по известной конечной длине пружины

Parameters

- **t**: Не используется
- **y_vecs**: Решение системы ДУ в момент времени `t`
- **constants**: Постоянные величины, необходимые для решения ДУ (используется только конечная длина пружины)

Returns

Результат сравнения расстояния между РН и КА и конечной длиной пружины

def motion_equation_rhs(t: float, y_vecs: numpy.ndarray, *constants) -> numpy.ndarray: • View Source

Функция расчета производной правой части системы ОДУ динамики поступательного движения РН и КА

Parameters

- **t**: Момент времени решения в секундах (не используется)
- **y_vecs**: Решение системы ОДУ в момент времени `t`
- **constants: Постоянные величины, необходимые для решения ДУ**: параметры пружины (жесткость в Н/м, конечная длина, длина в недеформированном состоянии в метрах), масса РН и КА в кг, единичный вектор `e_1v2sc`, задающий направление действия силы упругости пружинного толкателя

Returns

Значение производной правой части системы ОДУ