Федеральное государственное автономное образовательное учреждение высшего образования
«Российский университет дружбы народов имени Патриса Лумумбы»
Инженерная академия
Кафедра механики и процессов управления

ОТЧЕТ

По **Курсовой работе**

Направление: **01.03.02 Прикладная математика и информатика**
(код направления / название направления)

Профиль: **Математические методы механики космического полёта и анализа геоинформационных данных**
(название профиля)

Тема: **Подспутниковая трасса космического аппарата**
(название лабораторной / курсовой)

Выполнено студентом: **Рябинин Владислав Денисович**
(ФИО)

Группа: **ИПМбд-01-23**

№ студенческого: 1132233497

Москва, 2024

Оглавление

Актуальность

Актуальность построения подспутниковой трассы обусловлена постоянно растущей потребностью в точной информации о положении спутников и их траекториях для решения широкого круга задач в различных областях. Точное определение местоположения космического аппарата важно, например, для систем наблюдения Земли, включая мониторинг военных объектов и предотвращение чрезвычайных ситуаций

Цель

Построение подспутниковой трассы космического аппарата Молния 3-50 на сферической Земле

Задачи

1) Изучить понятие «подспутниковая трасса»

2) Составить план программы, которая на основе ранее полученных данных о спутнике будет проектировать его положение на поверхность сферы

3) Согласно плану разработать программу на языке Python

4) Проанализировать полученную подспутниковую трассу

Теоретическая часть

Подспутниковая трасса – это воображаемая линия на поверхности Земли, которая отображает путь, по которому движется проекция космического аппарата на Землю. Для её получения строится вертикальная линия, между центром масс Земли и центром масс спутника; подспутниковая трасса соединяет последовательные точки пересечения этой линии с поверхностью в каждый момент времени.

Построение подспутниковой трассы требует знания нескольких параметров:

- **Элементы орбиты спутника:** это набор параметров, определяющих форму и ориентацию орбиты спутника в пространстве (*большая полуось* **(a)**, *эксцентриситет* **(e)**, *наклонение* **(i)**, *долгота восходящего узла* (**Ω**), *аргумент перигея* (**ω**));
- **Временные метки:** для каждого момента времени требуется знать точные координаты спутника в пространстве;
- **Модель Земли:** для точной проекции на поверхность Земли используется математическая модель Земли (сфера, радиусом 6371 километр), учитывающая ее форму и размеры.

На основе этих данных *рассчитываются координаты (широта и долгота)* точки на поверхности Земли, непосредственно под спутником, для каждого момента времени. Соединение этих точек образует *подспутниковая трассу*.

Факторы, влияющие на подспутниковую трассу:

- **Форма Земли:** Поскольку Земля не является идеальной сферой, а имеет форму геоида (неправильный эллипсоид), подспутниковая трасса будет отклоняться от идеальной геометрической фигуры;
- **Возмущения орбиты:** Гравитационные поля Луны, Солнца и других небесных тел, а также атмосферное сопротивление вызывают отклонения орбиты от идеальной, влияющие на подспутниковую трассу;
- **Точность данных:** Точность подспутниковой трассы зависит от точности определения элементов орбиты, временных меток и модели Земли.

Практическая часть

После рассмотрения теории о спутниковой трассе, можно перейти к реализации программы на языке Python, которая бы строила её. Но прежде нужно составить план того, как именно она будет это делать:

Пункты, которые должна выполнять программа:

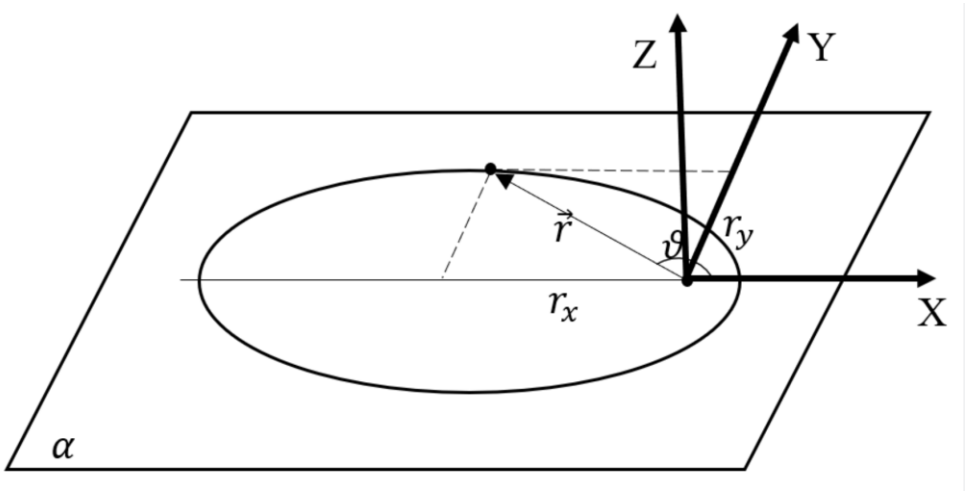
1. Считывание истинной аномалии из файла (File_Nu.txt), полученного в лабораторной работе 1;
2. Вычисление радиус-вектора спутника в каждой точке по формуле (2.1) и получение полярных координат космического аппарата в плоскости орбиты;

$$r(t) = \frac{p}{1+e\cos(\nu)} \quad (2.1)$$

3. Вычисление координат спутника в орбитальной системе координат, связанной с орбитой спутника;

   Введем систему координат (см. рисунок ниже), связанную с орбитой спутника, пусть

   - Ось X направлена вдоль большой полуоси орбиты (*в точку перицентра*);
   - ОсьY перпендикулярна оси X в плоскости орбиты (*направлена в точку, где эксцентрическая аномалия* ***(E)*** *= 90°*);
   - Ось Z перпендикулярна плоскости орбиты.



Из рисунка видно, что декартовы координаты в этой системе связаны с полярными через соотношения:

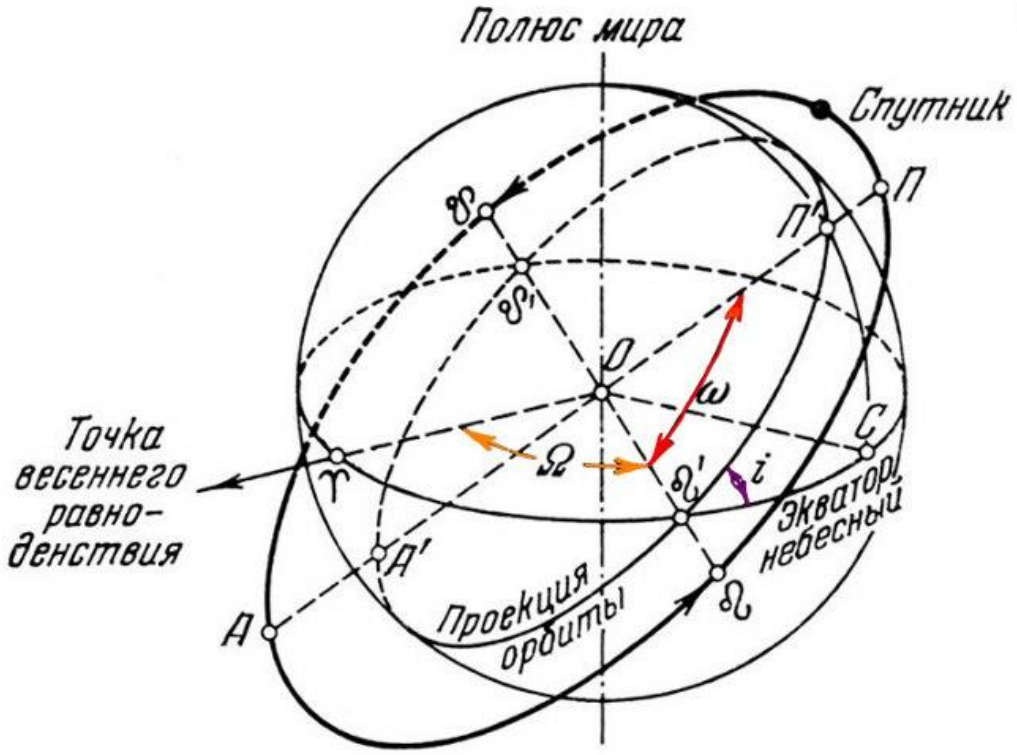
$x = r \cdot cos(v)$ $y = r \cdot sin(v)$ $z = 0$

v — истинная аномалия космического аппарата в данный момент t

r — радиус-вектор центра масс спутника

Координата z в любой точке траектории равна нулю.

4. Преобразование координат из орбитальной системы в геоцентрическую (см. рисунок ниже) с помощью матрицы поворота



Для перехода в геоцентрическую систему координат (АГЭСК) *потребуется выполнить три последовательных вращения орбитальной системы координат XYZ*:

Из рисунка видно, что сначала нужно выполнить поворот системы координат относительно оси Oz по часовой стрелке на угол (ω), затем относительно оси Ox по часовой стрелке сделать поворот на угол (i), и, наконец, снова выполнить поворот по часовой стрелке относительно оси Oz, но теперь на угол (Ω).

- Вращение вокруг оси Z на угол **ω** (аргумент перигея): переводит ось X с линии большой полуоси орбиты на линию узлов;
- Вращение вокруг оси X на угол **i** (наклонение): совмещает плоскость Oxy с плоскостью экватора;
- Вращение вокруг оси Z на угол **Ω** (долгота восходящего узла): совершает поворот в плоскости экватора на угол Ω, совмещая тем самым оси двух систем;

Для выполнения одного такого вращения в математике используется *матрица поворота (вращения)*.

Результирующая матрица преобразования будет произведением трех матриц вращения: $R_z(\Omega)$ задаёт *поворот оси Z* на угол **Ω**, $R_x(i)$ задаёт *поворот оси X* на угол **i**, $R_z(\omega)$ задаёт *поворот оси Z* на угол **ω**

$R = R_z(\Omega) * R_x(i) * R_z(\omega)$,

где матрицы $R_z(x)$ = ([cos(x), -sin(x), 0]
[sin(x), cos(x), 0]
[0, 0, 1])

$R_x(x)$ = ([1, 0, 0]
[0, cos(x), -sin(x)]
[0, sin(x), cos(x)])

После умножения, получим, что элементы матрицы R:

a_{11} = cos(Ω) * cos(ω) - sin(O) * sin(ω) * cos(i),

a_{12} = -cos(Ω) * sin(ω) - sin(Ω) * cos(ω) * cos(i),

a_{13} = sin(Ω) * sin(i),

a_{21} = sin(Ω) * cos(ω) + cos(Ω) * sin(ω) * cos(i),

a_{22} = -sin(Ω) * sin(ω) + cos(Ω) * cos(ω) * cos(i),

a_{23} = -cos(Ω) * sin(i),

a_{31} = sin(ω) * sin(i),

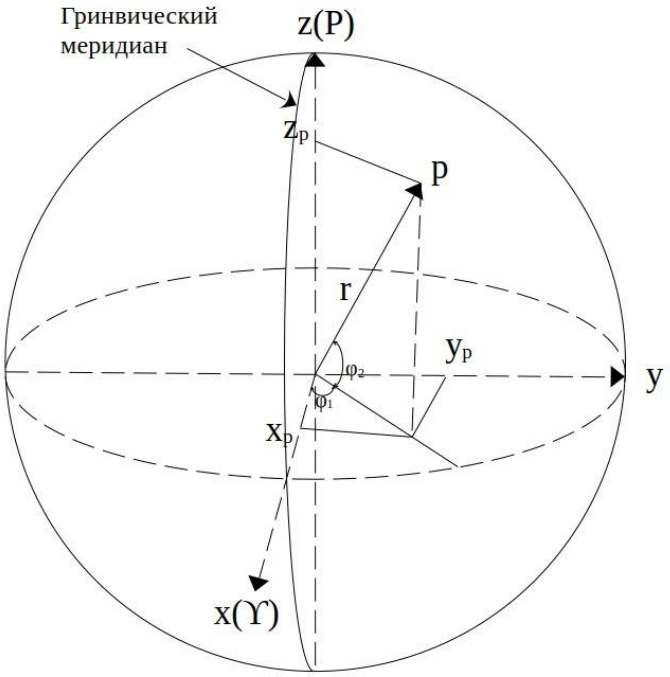
a_{32} = cos(ω) * sin(i),

a_{33} = cos(i)

Таким образом, чтобы перейти от орбитальной системы координат к геоцентрической (АГЭСК), нужно воспользоваться формулой:

$$[X, Y, Z]^T = R * [x, y, z]^T$$

5. Преобразование геоцентрических координат в географические (широту и долготу)



Для этого перехода нужно понимать, что пока спутник движется по своей орбите, Земля продолжает вращаться. Это накладывает некоторые сложности на вычисление долготы и широты. Поскольку у обеих систем отсчёта (АГЭСК и системы географических координат) одна точка отсчёта (центра масс Земли), то угол долготы определяется как (*для упрощения вычислений* будем считать, что в момент времени t = 0 (в момент когда начинаем вычислять положение спутника) *истинное звездное время на меридиане Гринвича = 0*, т. е. меридиан Гринвича находится в этот момент в плоскости OXZ):

$\varphi_1 = \arctan(Y / X)$

X, Y — координаты спутника в АГЭСК

Но с учётом того, что Земля продолжает вращаться, нужно также отнимать угол, на который она повернулась за время полета спутника. Таким образом мы получим долготу, которая определена относительно невращающейся системы. Итого:

$\varphi_1 = \arctan(Y / X) — w * t$

w — угловая скорость вращения Земли ($7.29 * 10^{-5}$ радиан в секунду)

t — время, которое прошло с момента начала отсчёта

На вторую координату вращение Земли влияния не оказывает, поэтому формула, выражающая зависимость не содержит время:

$\varphi_2 = \arctan(Z / \sqrt{(X^2 + Y^2)})$

Таким образом, используя эти формулы, можно получить углы φ_1 и φ_2, которые выражают географическую широту и долготу соответственно через координаты АГЭСК.

6. Преобразование широты и долготы точки в координаты на сфере

Последним шагом, который нужно сделать, чтобы получить проекцию спутника на Землю, это с использованием сферических координат задать точку на поверхности Земли. Зная широту **(φ_1)** и долготу **(φ_2)** точки, а также радиус Земли **(R)**, формулы, выражающие координаты точки на сфере, (как легко видеть из последнего рисунка), принимают вид:

$x = \cos(\varphi_2) * \cos(\varphi_1) * R$

$y = \cos(\varphi_2) * \sin(\varphi_1) * R$

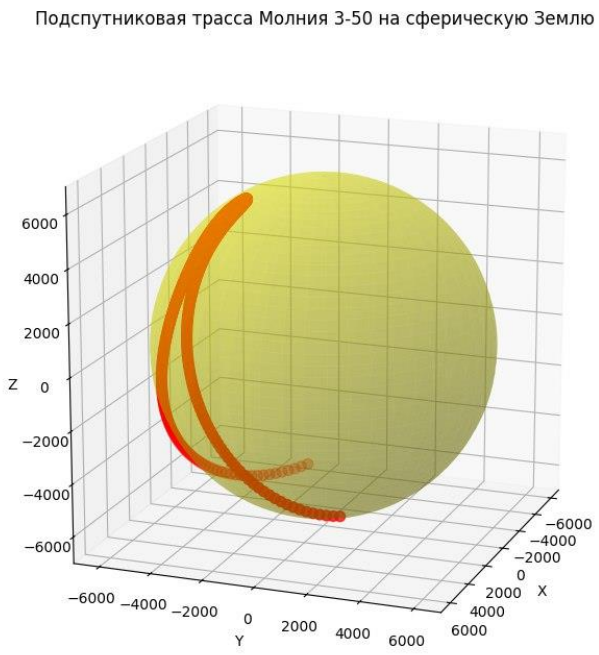
$z = \sin(\varphi_1) * R$

После подробного описания плана, в котором указаны все необходимые преобразования систем координат и формулы, была разработана программа на языке Python (BuildingTrack.py), реализующая каждый из пунктов. Подробное описание каждой строки программы приведено в комментариях к коду (а также в *Приложении 1*). Основная структура программы:

- Сначала подключаются необходимые библиотеки (numpy *для работы с матрицами,* matplotlib *для построения графиков,* math *для основных математических функций*);
- Далее идёт объявление в программе необходимых констант (μ и R для Земли) и кеплеровских элементов для спутника Молния 3-50;
- При запуске программы открывается файл File_Nu.txt, который содержит в двух столбцах значения времени и истинной аномалии спутника в этот момент времени. Эти значения считываются в программу;
- Для полученных значений вычисляются сначала орбитальные координаты спутника, далее они передаются в функцию *orbit_to_geocentric*, где вычисляются геоцентрические координаты спутника в АГЭСК
- В следующих строках вычисляются географические координаты точки на спутнике, путем передачи геоцентрических координат в функцию *geocentric_to_geografh*
- Значения этих координат заносятся в массив. Цикл повторяется до тех пор, пока не будет достигнут конец файла
- После этого массив получившихся координат передаётся в функцию *plot_spherical_points*, в которой происходит построение сферы и точек подспутниковой трассы на ней

Описание результата

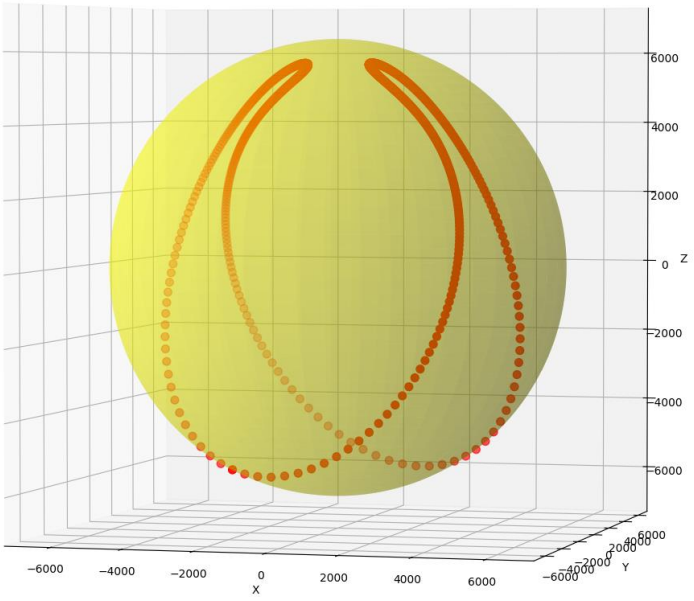
После запуска программы получился следующий график:



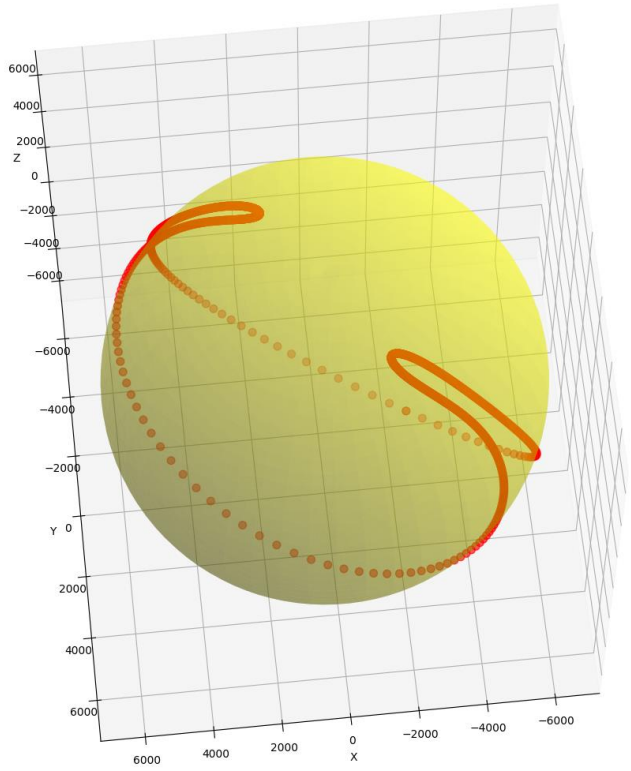
После некоторых размышлений, было сделано предположение о том, что построенный график отражает не всю подспутниковую трассу. Видно, что кривая не замкнута, что представляется практически невозможным в условиях упрощенной модели. Для построения этого графика было взято 1001 деление на отрезке [0; T], где T — период обращения спутника.

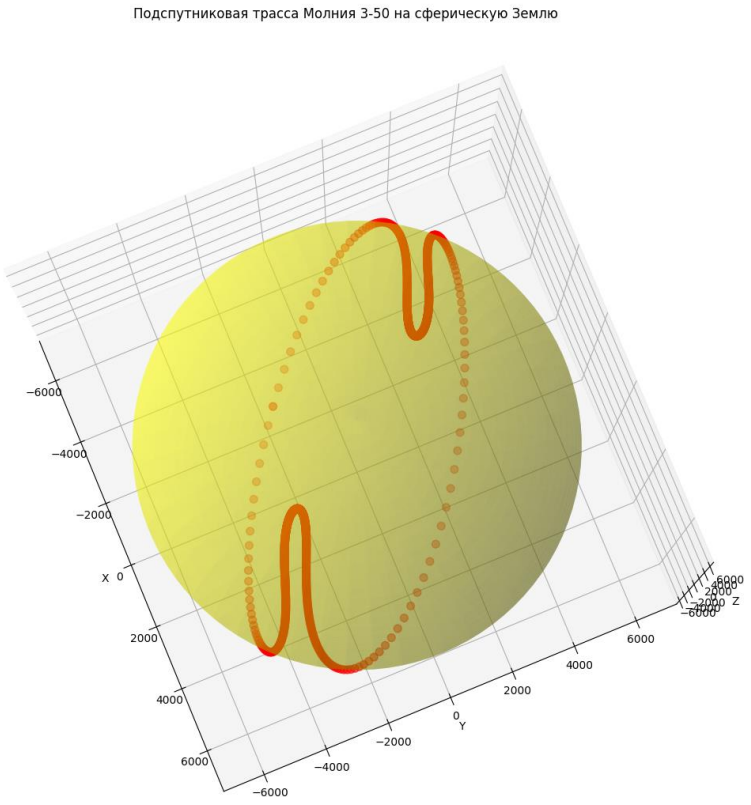
Следующим очевидным шагом стало увеличение отрезка времени, на котором строится трасса. Теперь график строился на отрезке от [0; 2 * T], результат стал выглядеть так:

Подспутниковая трасса Молния 3-50 на сферическую Землю



Подспутниковая трасса Молния 3-50 на сферическую Землю





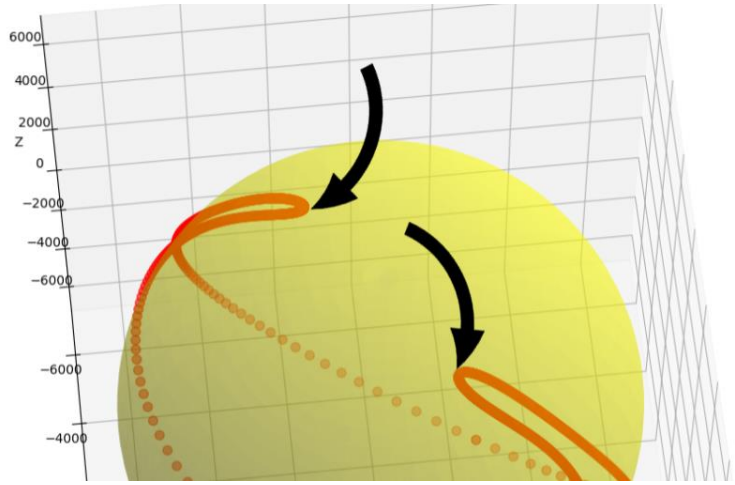
То есть после увеличения отрезка времени в два раза, трасса замкнулась.

Проанализируем полученную трассу:

- Спутник проходит по всей подспутниковой трассе (то есть приходит в точку, откуда начал движение) за 2 периода обращения;
- Из графика видно, что не вся трасса непрерывна, существуют её области, где она выражена точками. В эти моменты космический аппарат, очевидно, находится близко к Земле (перицентру), и его скорость имеет большое значение. Поскольку точки трассы вычисляются за равные промежутки времени, то, очевидно, количества этих точек недостаточно, чтобы построить на этих участках непрерывную кривую;
- Поскольку спутник Молния 3-50 расположен на эллиптической орбите с большим эксцентриситетом, то большую часть времени он находится в точках, для которых истинная аномалия (ν) принадлежит от 90° до 270°. В

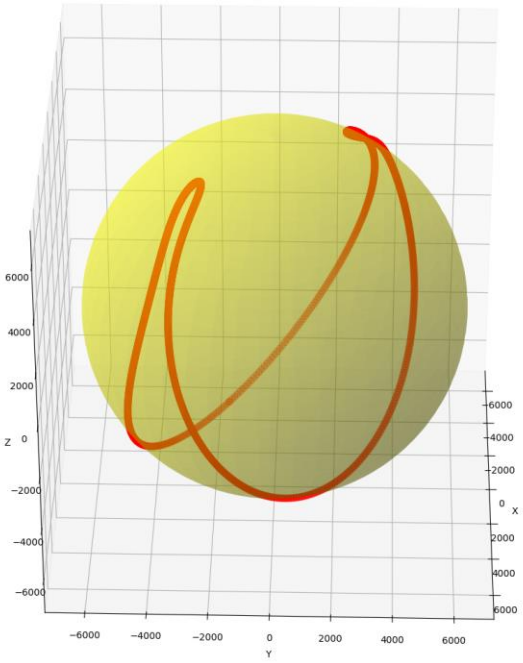
эти моменты космический аппарат движется медленно, поэтому трасса выражена линией, а не точками. То есть становится ясно, что на тех участках трассы, где линия непрерывна, спутник удален от Земли и находится близко к апоцентру

- На подспутниковой трассе выделяются две точки, которые находятся на непрерывном участке (см. фото ниже). Поскольку в этих точках спутник начинает двигаться в обратном направлении, то нетрудно понять, что это проекция точки апоцентра

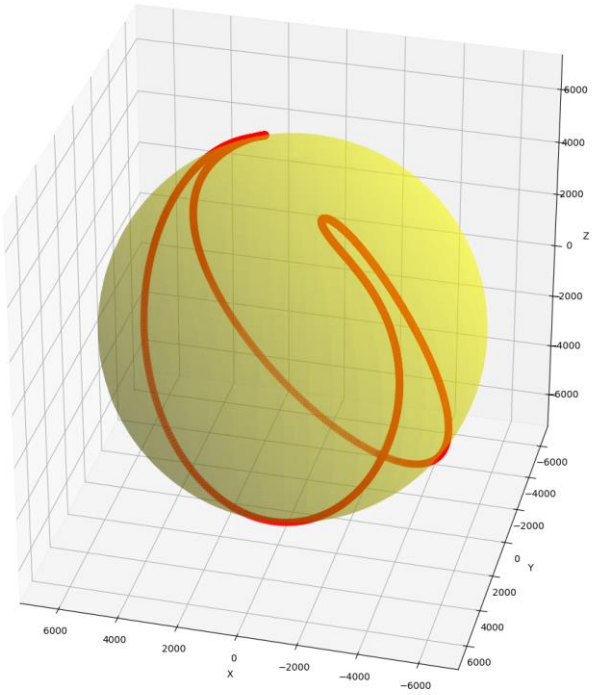


Также, для получения непрерывной линии на всей подспутниковой трассе было увеличено количество точек (с 1000 до 4000) на отрезке [0; 2 * T]. Ожидаемо получаем непрерывный график:

Подспутниковая трасса Молния 3-50 на сферическую Землю



Подспутниковая трасса Молния 3-50 на сферическую Землю



Заключение

В данной курсовой работе было рассмотрено построение подспутниковой трассы космического аппарата «Молния 3-50». В ходе работы были изучены основные характеристики орбиты спутника типа «Молния». Была разработана программа на языке Python, которая строит необходимую трассу, проектируя на сферическую Землю положение спутника.

В дальнейшем исследования можно расширить за счет более детального учета возмущающих факторов, использования более точных моделей гравитационного поля Земли и атмосферы. Полученные результаты могут быть использованы для планирования спутниковых наблюдений, расчета зоны видимости и оптимизации работы систем связи, использующих спутники типа «Молния-3». Разработанный алгоритм может быть адаптирован для построения подспутниковых трасс других типов космических аппаратов с высокоэллиптическими орбитами.

Все поставленные задачи были успешно выполнены, возникшие проблемы решены.

Приложение 1

```
import numpy as np

import matplotlib.pyplot as plt

from math import sqrt, sin, cos, pi, atan2, degrees, atan


# Константы

R = 6371

w_earth = 7.29 * (10 ** (-5))


# Зададим орбиту с помощью кеплеровских элементов

a = 26557.559030

e = 0.6910996

i = (63.5089 / 180) * pi

raan = (213.8149 / 180) * pi

w = (281.3930 / 180) * pi



def plot_spherical_points(fi_1, fi_2, radius, point_color='red'):

    fi_1_rad = np.radians(fi_1)  # radians для каждого элемента массива (долгота)

    fi_2_rad = np.radians(fi_2)  # пересчитывает градусы в радианы      (широта)
```

считаем декартовые координаты точек (x, y, z), которые отображают проекцию спутника в данный момент на сферу

x = radius * np.cos(fi_2_rad) * np.cos(fi_1_rad)

y = radius * np.cos(fi_2_rad) * np.sin(fi_1_rad)

z = radius * np.sin(fi_2_rad)

plt.figure - функция, которая создаёт новый объект фигуры, который содержит все элементы графика: оси, точки, линии, объекты

fig = plt.figure(figsize=(20, 16)) # параметр "figsize" задает размер фигуры в дюймах

функциця fig.add_subplot добавляет к созданной фигуре подграфик и настраивает его как трехмерный, создается объект Axes3D

ax = fig.add_subplot(111, projection='3d')

u = np.linspace(0, 2 * np.pi, 100) # создаём массив для первой сферической координаты на отрезке [0; 2*pi]

v = np.linspace(0, np.pi, 100) # создаем массив для второй сферической координаты на отрезке [0; pi]

вычисляется для каждой заданной в предыдущих массивах точки координата x (используется аппарат внешнего произведения, т. е. создается массив 100 на 100)

sphere_x = radius * np.outer(np.cos(u), np.sin(v))

аналогично вычисляется y (в этом массиве a[i][j] = u[i] * v[j])

sphere_y = radius * np.outer(np.sin(u), np.sin(v))

для координаты z искуственно задаётся массив из 100 единиц np.ones размером с массив u, дальше как и раньше

sphere_z = radius * np.outer(np.ones(np.size(u)), np.cos(v))

ax.plot_surface(sphere_x, sphere_y, sphere_z, color='yellow', alpha=0.35, edgecolor='none')

метод plot_surface объекта Axes3D строит трехмерную поверхность: первые три аргумента это ранее заданные координаты точек сферы

alpha - параметр прозрачности (35%)

используя метод scatter объекта Axes3D строим точки, которые посчитаны решением уравнения Эйлера; параметр s=50 задаёт размер точек

ax.scatter(x, y, z, color=point_color, s=50)

ax.set_xlabel('X') # название осей координат: X, Y и Z соответственно

ax.set_ylabel('Y')

ax.set_zlabel('Z')

ax.set_title('Подспутниковая трасса Молния 3-50 на сферическую Землю') # а также название самого рисунка

```
    ax.grid(True)  # отражаем сетку

    ax.set_box_aspect([1, 1, 1])  # для равных масштабов по всем осям устанавливаем соотношение

    plt.show()  # выводим результат на экран


def orbit_to_geocentric(orbit_coords, i, raan, w):

    # найдем нужные компоненты для матрицы поворота

    cos_O = cos(-raan)
    sin_O = sin(-raan)
    cos_i = cos(-i)
    sin_i = sin(-i)
    cos_w = cos(-w)
    sin_w = sin(-w)

    # запишем формулу матрицы поворота вокруг трёх осей X, Y, Z
```

```
    rotation_matrix = np.array([
        [cos_O * cos_w - sin_O * sin_w * cos_i, -cos_O * sin_w - sin_O * cos_w * cos_i, sin_O * sin_i],
        [sin_O * cos_w + cos_O * sin_w * cos_i, -sin_O * sin_w + cos_O * cos_w * cos_i, -cos_O * sin_i],
        [sin_w * sin_i, cos_w * sin_i, cos_i]
    ])

    geocen_coords = rotation_matrix @ orbit_coords

    return geocen_coords


# Функция для вычисления географических координат из геоцентрических
def geocentric_to_geografh(r_geo, t):
    x, y, z = r_geo
    # функция degrees переводит значения углов из радиан в градусы; делением на 360 приводит градусы к (-360; 360)
    fi_1 = (degrees(atan2(y, x)) - degrees(w_earth * t)) % 360
    fi_2 = degrees(atan(z / sqrt(x**2 + y**2)))
    return fi_2, fi_1
```

```
if __name__ == "__main__":


    fi_2 = []  # широта

    fi_1 = []  # долгота


    with open("File_Nu.txt", "r") as file:

        str = file.readline()

        while str != '':

            index = str.find(' ')

            t, true_anomaly = float(str[:index]), float(str[index + 1:-1])

            r = a * (1 - e ** 2) / (1 + e * cos(true_anomaly))


            # координаты в орбитальной системе системе

            orbit_coords = np.array([r * cos(true_anomaly), r * sin(true_anomaly), 0])


            # координаты в геоцентрической системе координат

            geocen_coords = orbit_to_geocentric(orbit_coords, i, raan, w)


            # в конце переведём геоцентрические координаты в географические

            curr_fi_2, curr_fi_1 = geocentric_to_geografh(geocen_coords, t)

            fi_2.append(curr_fi_2)
```

```
        fi_1.append(curr_fi_1)


        str = file.readline()


plot_spherical_points(fi_1, fi_2, radius=R, point_color='red')
```