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

ОТЧЕТ

По	Лабораторной работе №2 по Механике Космического Полета. Вариант 4.

Направление:	01.03.02 Прикладная математика и информатика
	(код направления / название направления)
Профиль:	Математические методы механики полета раке-носителей и космических аппаратов
	(название профиля)

Тема:	Определение возмущающего ускорения, обусловленного сопротивлением атмосферы Земли.
	(название лабораторной / курсовой)

Выполнено студентом:	Критским Матвеем Димитриевичем		
	(ФИО)		
Группа:			ИПМбд-02-22
№ студенческого:			1132226149

Москва, 2023

Цель работы:
Исследовать возмущения, вызываемые сопротивлением атмосферы Земли, подсчитать возмущенное ускорение и его состовляющие. Увидеть зависимость возмущенного ускорения и его состовляющих от уровня солнечной активности на зафиксированной высоте.

Исходные данные:

- Информация об орбите:

  Высота апоцентра(h_a) - 850 км
  Высота перицентра(h_p) - 350 км
  Наклонение(i) - 45 град. / $\frac{\pi}{4}$ рад. / ~ 0.785398 рад.
  Долгота восходящего узла(**Ω**) - 20 град. / $\frac{\pi}{9}$ рад. / ~ 0.349066 рад.
  Аргумент перицентра(ϖ) - 0 град. / 0 рад.
  Средняя аномалия(M) - 15 град. / $\frac{\pi}{12}$ рад. / ~ 0.261799 рад.

  Радиус апоцентра(r_a) - $R_з + h_a$ = 7228.1 км
  Радиус перицентра(r_p) - $R_з + h_a$ = 6728.1 км

  Большая полуось(a) - $\frac{r_a + r_p}{2}$ = 6978.1 км
  Эксцентриситет(e) - $\frac{r_a - r_p}{r_a + r_p}$ = ~ 0.0358264
  Фокальный параметр(p) - $a \cdot (1 - e^2)$ = 6969.14 км

- Информация о планете Земля:

  Радиус($R_з$) - 6378.1 км
  Масса($M_з$) - $5.9726 \cdot 10^{24}$ кг
  Гравитационный параметр(μ) = 398600.4415
  Гравитационная постоянная(G) - $6.674 \cdot 10^{-11} \frac{(\text{Н} \cdot \text{м}^2)}{\text{кг}^2}$
  Угловая скорость($ϖ_з$) - $7.2921158553 \cdot 10^{-5} \frac{\text{рад}}{\text{с}}$
  Плотность ночной атмосферы на высоте 120 км($ρ_0$) - $1.58868 \cdot 10^{-8} \frac{\text{кг}}{\text{м}^3}$
  Эксцентриситет Земли($e_з^2$) - 0.0067385254
  Большая полуось ОЗЭ($a_з$) - 6378136 м

- Информация о КЛА:
  Масса КЛА(m) - 1500 кг

Коэф. Силы лобового сопротивления(C_{xa}) - 2
Площадь КЛА(S_a) - 12 м^2
Баллистический коэф. КЛА(σ_x) - $\frac{C_{xa} \cdot S_a}{2 \cdot m}$ = 0.008

Ход работы:

1. С помощью исходных параметров орбиты найти координаты заданной точки в АГЭСК (x_a, y_a, z_a).
2. Найти скорость и ее составляющие (трансверсальная скорость, радиальная скорость).
3. На базе значений координат АГЭСК найти положение этой точки в ГСК и рассчитать соответствующие им геодезические координаты L, B и H.
4. Рассчитать плотность атмосферного давления $\rho_{атм}(H)$.
5. Найти составляющие возмущающего ускорения, обусловленного влиянием атмосферы.
6. Подвести вывод.

1) С помощью исходных параметров орбиты найти координаты заданной точки в АГЭСК (x_a, y_a, z_a).

Для расчета координат необходимо найти истинную аномалию(θ), эксцентрическую аномалию(E), аргумент широты орбиты(u) и модуль радиус-вектора КА в АГЭСК(r_A).

Эксцентрическую аномалию можно найти методом приближений из уравнения Кеплера:

1) Задаем начальное значение $E_0 = M$.
2) Рассчитываем новое значение E по формуле: $E_{i+1} = M + e \cdot \sin(E_i)$, где i - номер итерации
3) Если $|E_{i+1} - E_i| \leq \varepsilon$, где ε - заранее заданное малое число (в нашем случае ε = 0.001 град. / $\frac{\pi}{180000}$рад. / ~ 0.000017453 рад.), то $E = E_{i+1}$, иначе $E_i = E_{i+1}$. Алгоритм повторяется с шага 2)

   В итоге E = 0.271403.

С помощью найденной эксцентрической аномалии мы можем найти истинную аномалию и модуль радиус-вектора КА в АГЭСК по след. формулам:

$\theta = 2 \cdot \arctan\left(\sqrt{\frac{1+e}{1-e}} \cdot \tan\left(\frac{E}{2}\right)\right) = 0.28118$

$r_A = a \cdot (1 - e \cdot \cos(E)) = 6737.25$ км (или можно выразить радиус-вектор через истинную аномалию: $\frac{a \cdot (1-e^2)}{1 + e \cdot \cos(\theta)}$)

С помощью истинной аномалии вычислим аргумент широты орбиты: $u = \theta + \varpi$ (т.к. $\varpi = 0$ согласно начальным данным, то $u = \theta = 0.28118$)

Теперь можно вычислить координаты АГЭСК:

$x_a = r_A \cdot (\cos(u) \cdot \cos(\Omega) - \sin(u) \cdot \sin(\Omega) \cdot \cos(i))$
$y_a = r_A \cdot (\cos(u) \cdot \sin(\Omega) + \sin(u) \cdot \cos(\Omega) \cdot \cos(i))$
$z_a = r_A \cdot \sin(u) \cdot \sin(i)$
Конечные координаты КЛА в АГЭСК: [5630.19, 3456.01, 1321.95].

2) Найти скорость и ее составляющие (трансверсальная скорость, радиальная скорость).

Для упрощения выражений будем считать, что атмосфера Земли неподвижна ($V_{пер} = 0$). Тогда:

$V = V_{пер} + V_{отн} = V_{отн}$

В проекциях на S, T, W имеет вид $V = [-V_r, -V_\tau, 0]$.

V_r и V_τ можно найти по след. формулам:

$$V_r = \sqrt{\frac{\mu}{p}} \cdot e \cdot \sin(\theta) = \sim 0.0751845 \frac{км}{с}$$

$$V_\tau = \sqrt{\frac{\mu}{p}} \cdot (1 + e \cdot \cos(\theta)) = \sim 7.82305 \frac{км}{с}$$

Зная радиальную и трансверсальную скорости, можно посчитать результирующую скорость V:

$V = \sqrt{V_r^2 + V_\tau^2} = \sim 7.82341 \frac{км}{с}$ (или ее можно посчитать, не вычисляя ее составляющих по след. формуле: $\sqrt{\frac{\mu}{p}} \cdot \sqrt{1 + 2 \cdot e \cdot \cos(\theta) + e^2}$)

3) На базе значений координат АГЭСК найти положение этой точки в ГСК и рассчитать соответствующие им геодезические координаты L, B и H.

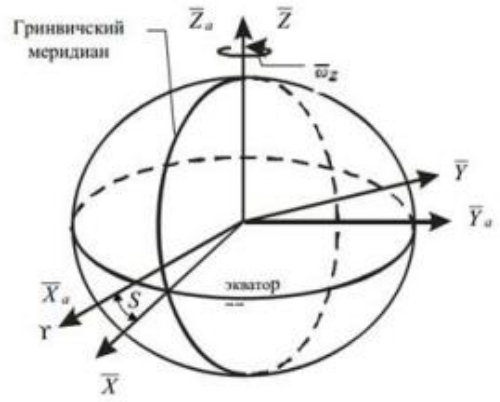

Рис. 1: Абсолютная и относительная экваториальные СК

Взаимное положение систем координат АГЭСК и ГСК определяется углом поворота Земли вокруг оси вращения S(t), которое отвечает звездному моменту времени на момент t времени UTC. Для того, чтобы получить ГСК из АГЭСК, надо умножить координаты АГЭСК на матрицу перехода
$S(t) = \varpi_з \cdot t$

$$\Psi = \begin{bmatrix} \cos(S(t)) & \sin(S(t)) & 0 \\ -\sin(S(t)) & \cos(S(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Тогда:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(S(t)) & \sin(S(t)) & 0 \\ -\sin(S(t)) & \cos(S(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix}$$

Или:

$$x = \cos(S(t)) \cdot x_a + \sin(S(t)) \cdot y_a$$
$$y = -\sin(S(t)) \cdot x_a + \cos(S(t)) \cdot y_a$$
$$z = z_a$$

Получим координаты в ГСК, изменяющиеся в пределе t = [0; T]:

x ∈ [5630.19; 6552.65] (чем больше t, тем больше x)
y ∈ [840.117; 3456.01] (чем больше t, тем меньше y)
z = 1321.95 (не зависит от t)

Связь координат (x, y, z) в ГСК с геодезическими (L, B, H) задаются формулами

$x = (N + H) \cdot \cos(B) \cdot \cos(L)$
$y = (N + H) \cdot \cos(B) \cdot \sin(L)$
$z = \left((1 - e_з^2) \cdot N + H\right) \cdot \sin(B)$

, где

$N = \frac{a_з}{\sqrt{1 - e_з^2 \cdot \sin^2(B)}}$

Так как при обратном преобразовании по вышеупомянутым формулам в определенных случаях имеется деление на ноль, то необходимо использовать специальный алгоритм перехода от ГСК в геодезическую СК:

Вычисляем величину D по формуле: $D = \sqrt{x^2 + y^2}$

1. Если D = 0, то $B = \frac{\pi}{2} \cdot \frac{z}{|z|}$, L = 0, $H = z \cdot \sin(B) - a_з \cdot \sqrt{1 - e_з^2 \cdot \sin^2(B)}$. Конец алгоритма.

D может быть равно одному из трех значений:
{6606.2856436078000115, 6606.285643607800921, 6606.2856436078018305}

2. Если D > 0, то $L_a = \arcsin\left(\frac{y}{a_з}\right)$

1) Если y < 0 и x > 0, то $L = 2 \cdot \pi - L_a$;
2) Если y < 0 и x < 0, то $L = \pi + L_a$;
3) Если y > 0 и x < 0, то $L = \pi - L_a$;
4) Если y > 0 и x > 0, то $L = L_a$;

3. Анализируем значение z:
   a. Если z = 0, то B = 0 и $H = D - a_з$. Конец алгоритма.
   b. Если z != 0, то необходимо найти доп. величины

$R = \sqrt{x^2 + y^2 + z^2}$ =~ (аналогично с D и если $D = D_1$, то $R = R_1$)
{6737.2511006441927748; 6737.2511006441936843; 6737.2511006441945938}

$C = \arcsin\left(\frac{z}{r}\right)$ =~ (аналогично с D и R)
{0.1974958863116911545; 0.1974958863116911267; 0.1974958863116910990}

$P = \frac{e_з^2 \cdot a_з}{2 \cdot r}$ =~ (аналогично с D, R и с C)
{0.0031896711877448707; 0.0031896711877448703; 0.0031896711877448698}

и реализовать итерационный процесс:

$S_1 = 0$; $b = C + S_1$; $S_2 = \arcsin\left(\frac{P \cdot \sin(2 \cdot b)}{\sqrt{1 - e_з^2 \cdot \sin^2(b)}}\right)$

Если $|S_2 - S_1| < \varepsilon$, где ε = 0.0001’’(угловая секунда)/$\frac{\pi}{648000000}$ рад - заведомо малая положительная величина, то B = b, а

$$H = D \cdot \cos(B) + z \cdot \sin(B) - a_э \cdot \sqrt{1 - e_э^2 \cdot \sin^2(B)}$$

, иначе $S_2 = S_1$, и снова идет пересчет b.

Решение найдено, выход из алгоритма.

С помощью данного алгоритма получим координаты в геодезической СК:

L ∈ [0.127514624313105; 0.550530265079059] (чем больше t, тем меньше L]

B =~ 3456.0071147812259369 (не изменяется)

H = (аналогично с доп. параметрами D, R, C и P)
{359.94761356643084582; 359.94761356643175532; 359.94761356643266481}
В дальнейшем будем считать, что H не изменяется и равен 359.94761.

Погрешность нахождения H по данному алгоритму не превышает 0.003 м. Положения объектов на геодезических картах представляются в геодезических координатах, кроме того, геодезические координаты применяются в теории полета космических аппаратов, в расчетах, связанных с возмущением движения КА. Использование геодезических координат и этот алгоритм позволяет решать задачи, в которых используется оценка взаимного расположения объектов на поверхности Земли и КА дистанциооного зондирования Земли.

4) Рассчитать плотность атмосферного давления $\rho_{атм}(H)$.

Плотность атмосферы ρ, $\frac{кг}{м^3}$, вычисляют по формуле:

$\rho = \rho_н \cdot K_0 \cdot (1 + K_1 + K_2 + K_3 + K_4)$

, где

$\rho_н = \rho_0 \cdot e^{(a_0 + a_1 \cdot H + a_2 \cdot H^2 + a_3 \cdot H^3 + a_4 \cdot H^4 + a_5 \cdot H^5 + a_6 \cdot H^6)}$(e - экспанента, а не эксцентриситет!)

$\rho_н$ - плотность ночной атмосферы, $\frac{кг}{м^3}$

a_i - коэф. модели, используемые для расчета плотности атмосферы при различных значениях фиксированного уровня солнечной активности F_0;
K_i - нормирующие коэф., учитывающие суточные, полугодовые отклонения плотности атмосферы, геомагнитную активность Солнца. Будем полагать, что $K_0 = 1$, а остальные $K_i = 0$. При таких K мы получаем, что

$\rho = \rho_н$

Значения a_i берутся из таблицы №1 или таблицы №2.

Таблица №1. Коэф. Модели плотности атмосферы для первого высотного диапазона (120-500 км)

Коэффициент		Значение при фиксированном уровне солнечной активности F_0, 10^{-22} Вт/(м^2 · Гц)						
Обозначение	Размерность	75	100	125	150	175	200	250
a_h	км	120	120	120	120	120	120	120
a_0	-	26,8629	27,4598	28,6395	29,6418	30,1671	29,7578	30,7854
a_1	км$^{-1}$	-0,451674	-0,463668	-0,490987	-0,514957	-0,527837	-0,517915	-0,545695
a_2	км$^{-2}$	0,00290397	0,002974	0,00320649	0,00341926	0,00353211	0,00342699	0,00370328
a_3	км$^{-3}$	-1,06953e-5	-1,0753e-5	-1,1681e-5	-1,25785e-5	-1,30227e-5	-1,24137e-5	-1,37072e-5
a_4	км$^{-4}$	2,21598e-8	2,17059e-8	2,36847e-8	2,5727e-8	2,66455e-8	2,48209e-8	2,80614e-8
a_5	км$^{-5}$	-2,42941e-11	-2,30249e-11	-2,51809e-11	-2,75874e-11	-2,85432e-11	-2,58413e-11	-3,00184e-11
a_6	км$^{-6}$	1,09926e-14	1,00123e-14	1,09536e-14	1,21091e-14	1,25009e-14	1,09383e-14	1,31142e-14

Таблица №2. Коэф. Модели плотности атмосферы для второго высотного диапазона (500-1500 км)

Коэффициент		Значение при фиксированном уровне солнечной активности F_0, 10^{-22} Вт/(м^2 · Гц)						
Обозначение	Размерность	75	100	125	150	175	200	250
a_h	км	500	500	500	500	500	500	500
a_0	-	17,8781	-2,54909	-13,9599	-23,3079	-14,7264	-4,912	-5,40952
a_1	км$^{-1}$	-0,132025	0,0140064	0,0844951	0,135141	0,0713256	0,0108326	0,00550749
a_2	км$^{-2}$	0,000227717	-0,00016946	-0,000328875	-0,000420802	-0,000228015	-8,10546e-5	-3,78851e-5
a_3	км$^{-3}$	-2,2543e-7	3,27196e-7	5,05918e-7	5,73717e-7	2,8487e-7	1,15712e-7	2,4808e-8
a_4	км$^{-4}$	1,33574e-10	-2,8763e-10	-3,92299e-10	-4,03238e-10	-1,74383e-10	-8,13296e-11	4,92183e-12
a_5	км$^{-5}$	-4,50458e-14	1,22625e-13	1,52279e-13	1,42846e-13	5,08071e-14	3,04913e-14	-8,65011e-15
a_6	км$^{-6}$	6,72086e-18	-2,05736e-17	-2,35576e-17	-2,01726e-17	-5,34955e-18	-4,94989e-18	1,9849e-18

Учитывая, что H попадает в первый высотный диапазон, используем данные из таблицы №1. Таким образом, мы можем расчитать $\rho_н$ при

различных уровнях солнечной активности. Вывод данных будет в порядке возрастания солнечной активности, как показано в таблице №1 и таблице №2.

$\rho = \{1.68257 \cdot 10^{-12}; 3.00353 \cdot 10^{-12}; 4.60543 \cdot 10^{-12}; 6.47784 \cdot 10^{-12}; 8.65708 \cdot 10^{-12}; 1.10293 \cdot 10^{-11}; 1.63142 \cdot 10^{-11}\}$

Так как H практически одинаков в любой точке, плотность не изменяется.

5) Найти составляющие возмущающего ускорения, обусловленного влиянием атмосферы.

Запишем модули ускорений S, T, W:

S = $-\sigma_x \cdot \rho \cdot V \cdot V_r$

T = $-\sigma_x \cdot \rho \cdot V \cdot V_\tau$

W = 0

Результирующее ускорение считается по формуле:

a = $\sqrt{S^2 + T^2 + W^2}$ (с этого момента “a” - это ускорение, а не большая полуось орбиты)

Ускорение силы притяжения определяется по следующему соотношению:

g = $G \cdot \frac{M_з}{(R_з + H)^2}$

Подставив H, получим:

g_1 = ~ 8.78013 $\frac{м}{с^2}$

Подставив значения, получим 7 ускорений при 7-и разных уровнях солнечной активности:

Для первого высотного диапазона:

S = {-7.91748· 10^{-12}; -1.41334· 10^{-11}; -2.16713· 10^{-11}; -3.04821· 10^{-11}; -4.07367· 10^{-11}; -5.18993· 10^{-11}; -767678· 10^{-11}}

T = {-8.23824· 10^{-10}; -1.4706· 10^{-9}; -2.25493· 10^{-9}; -3.1717· 10^{-9}; -4.23871· 10^{-9}; -5.4002· 10^{-9}; -8.98779· 10^{-9}}

W = 0

a = {8.23862· 10^{-10}; 1.47066· 10^{-9}; 2.25503· 10^{-9}; 3.17185· 10^{-9}; 4.2389· 10^{-9}; 5.40044· 10^{-9}; 7.98816· 10^{-9}}

Графики зависимости возмущающего ускорения и его составляющих от уровня солнечной активности:

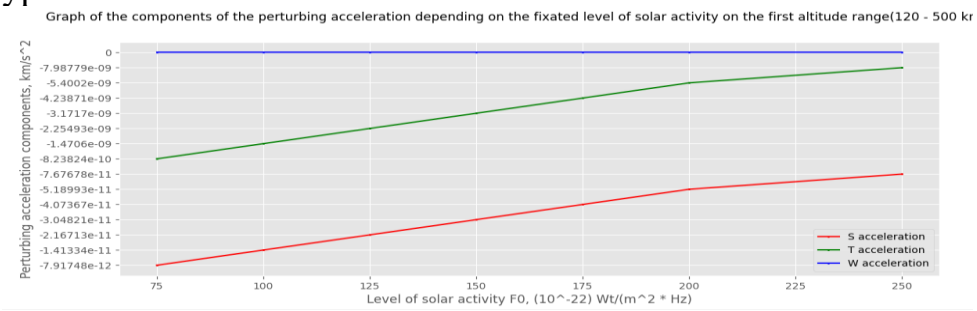

Рис. 2 График зависимости составляющих возмущающего ускорения от уровня солнечной активности. (первый высотный диапазон)

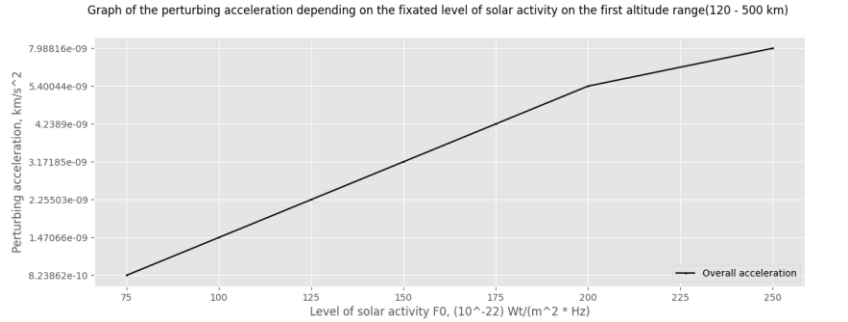

Рис. 3 График зависимости возмущающего ускорения от уровня солнечной активности. (первый высотный диапазон)

6)Вывод.

Из пункта №3 можно заметить, что во всех точках координаты ГСК x, y всегда положительны, а координата z не равна 0, т.е. мы всегда попадаем в пункт 3 в алгоритме расчета L, B, H, где L всегда равен L_a. Также, не смотря на изменения координат x, y в ГСК, дополнительные параметры R, C и P во всех точках приблизительно равны - числа начинают различаться на ~11 знаке после запятой. Отсюда следует, что B и H во всех точках приблизительно одинаковы. Поэтому из-за округления при расчетах в коде значения плотности, возмущающего ускорения и его составляющих равны. Также с увеличением уровня солнечной активности плотность, а в последствии и возмущающее ускорение с его составляющими увеличиваются.

Модуль возмущающего ускорения S принимает значения [-7.91748$\cdot 10^{-12}$; -7.67678$\cdot 10^{-11}$]$\frac{м}{с^2}$ на высоте 359.94761 км.

Модуль возмущающего ускорения T принимает значения [-8.23824$\cdot 10^{-10}$; -7.98779$\cdot 10^{-9}$]$\frac{м}{с^2}$на высоте 359.94761 км

Модуль возмущающего ускорения W равен 0 и не изменяется.

Возмущающее ускорение принимает значения [8.23862$\cdot 10^{-10}$; 7.98816$\cdot 10^{-9}$] на высоте 359.94761 км

Из этих показателей видно, что чем выше уровень солнечной активности, тем больше значения S, T и a.

При сравнении ускорения силы притяжения g и возмущающего ускорения(a), видим, что g сильно больше a.

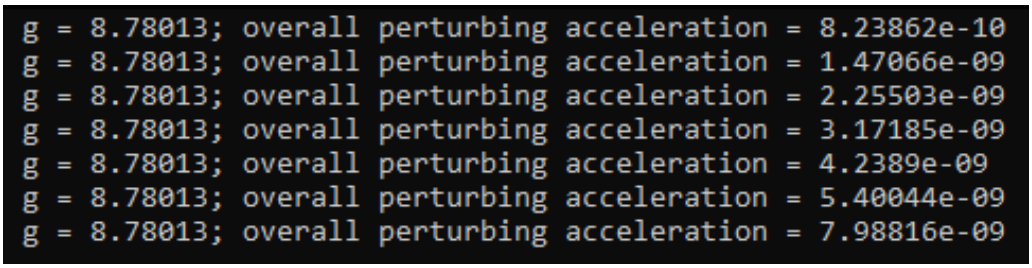

Рис. 4 Вывод консоли со сравнением g и a на разных высотах.

7) Приложение.

Ссылка на GitHub: https://github.com/rudnmk/MSF/tree/main/2/code/lab%202

Код программы:

Calculation.h :

```
#pragma once
#include <iostream>
#include <math.h>
#include <vector>
#include <array>
#include <fstream>
#include <iomanip>


#define PI 3.1415926

//Earth parameters
#define apogee_radius 7228.1 //apogee height(850 km) + Earth radius(6378.1 km)
#define perigee_radius 6728.1 //perigee height(350 km) + Earth radius
#define grav_param 398600.4415 //Gravitational parameter of the Earth
#define angular_vel 7.2921158553 * pow(10.0, -5.0) //Angular velocity of the
Earth

#define i 45 * PI / 180.0 //inclanation
#define Omega 20 * PI / 180.0 // ascending node longitude
#define omega 0 // pericenter argument
#define M 15 * PI / 180.0 // average anomaly
#define accuracy 0.001 * PI / 180.0

#define SMA ((apogee_radius + perigee_radius) / 2) // semi-major axis
```

```
#define ECC ((apogee_radius - perigee_radius) / (apogee_radius + perigee_radius)) //eccentricity
#define P (SMA * (1 - pow(ECC, 2.0))) // focal parameter


#define night_density 1.58868 * pow(10.0, -8.0) //night density of the atmosphere on the height of 120 km
#define T 2 * PI * sqrt(pow(SMA, 3.0) / grav_param)

#define SIGMA (12.0 * 2.0) / (2.0 * 1500.0)



std::pair<double, double> calculate_anomalies();
std::pair<double, double> calculate_velocities(double THETA);

std::vector<double> calculate_AGECS(double E, double THETA);
std::vector<double> calculate_GCS(std::vector<double> AGECS_coords, double time);
std::vector<double> calculate_geodetic_coords(std::vector<double> GCS_coords);

void calculate_density_and_acceleration(double H, double radial_vel, double transversal_vel, double vel);

void calculation(double time);
```

Calculation.cpp

```
#include "Calculation.h"


void calculation(double time) {
double E_anomaly;
double THETA_anomaly;
double radial_velocity;
double transversal_velocity;
double velocity;
std::vector<double> AGECS_coords = { 0.0, 0.0, 0.0 };
std::vector<double> GCS_coords = { 0.0, 0.0, 0.0 };
std::vector<double> geodetic_coords = { 0.0, 0.0, 0.0 };
```

```
std::ofstream data_input;
data_input.open("C:/Users/mk170/MSF/2/code/lab 2/lab 2 graph/DATA.txt",
std::ofstream::app);

//anomaly calculations
std::pair <double, double> anomaly_calculation_result = calculate_anomalies();
E_anomaly = anomaly_calculation_result.first;
THETA_anomaly = anomaly_calculation_result.second;

//velocities calculation
std::pair <double, double> velocity_calculation_result =
calculate_velocities(THETA_anomaly);
radial_velocity = velocity_calculation_result.first;
transversal_velocity = velocity_calculation_result.second;
velocity = sqrt(pow(radial_velocity, 2.0) + pow(transversal_velocity, 2.0));

//calculating spacecraft's position in different coordinate systems
AGECS_coords = calculate_AGECS(E_anomaly, THETA_anomaly);
//--------------from here on the calculation is time-based-------------------//
GCS_coords = calculate_GCS(AGECS_coords, time);
geodetic_coords = calculate_geodetic_coords(GCS_coords);

//calculating density and acceleration
calculate_density_and_acceleration(geodetic_coords[2], radial_velocity,
transversal_velocity, velocity);

//OUTPUT
data_input << "----------------------TIME: " << time << " ----------------------" <<
std::endl;
data_input << "Semi - major axis(SMA): " << SMA << "; Eccentricity(ECC) : " <<
ECC << "; Focal param: " << P << "; E: " << E_anomaly << "; THETA: " <<
THETA_anomaly << std::endl;
data_input << "Radial velocity: " << radial_velocity << "; Transversal velocity: "
<< transversal_velocity << "; Velocity: " << velocity << std::endl;
data_input << "AGECS coordinates: [" << AGECS_coords[0] << ", " <<
AGECS_coords[1] << ", " << AGECS_coords[2] << "]" << std::endl;
data_input << "GCS coordinates: [" << GCS_coords[0] << ", " << GCS_coords[1]
<< ", " << GCS_coords[2] << "]" << std::endl;
data_input << std::setprecision(20) <<"Geodetic coordinates: [" <<
geodetic_coords[0] << ", " << AGECS_coords[1] << ", " << geodetic_coords[2]
<< "]" << std::endl;
```

```
data_input.close();
}



std::pair<double, double> calculate_anomalies() {
double E_anomaly_past = M;
double E_anomaly = M + ECC * sin(E_anomaly_past);


while (fabs(E_anomaly - E_anomaly_past) > accuracy) {
E_anomaly_past = E_anomaly;
E_anomaly = M + ECC * sin(E_anomaly_past);
}

double THETA_anomaly = 2 * atan(sqrt((1.0 + ECC) / (1.0 - ECC)) *
tan(E_anomaly / 2.0));

return std::make_pair(E_anomaly, THETA_anomaly);
}

std::pair <double, double> calculate_velocities(double THETA) {
double radial_velocity = sqrt(grav_param / P) * ECC * sin(THETA);
double transversal_velocity = sqrt(grav_param / P) * (1 + ECC * cos(THETA));
return std::make_pair(radial_velocity, transversal_velocity);
}

std::vector<double> calculate_AGECS(double E, double THETA) {
std::vector<double> AGECS_coords = { 0.0, 0.0, 0.0 };
double AGECS_radius_vector = SMA * (1 - ECC * cos(E));
double u = THETA + omega;

AGECS_coords[0] = AGECS_radius_vector * (cos(u) * cos(Omega) - sin(u) *
sin(Omega) * cos(i));
AGECS_coords[1] = AGECS_radius_vector * (cos(u) * sin(Omega) + sin(u) *
cos(Omega) * cos(i));
AGECS_coords[2] = AGECS_radius_vector * sin(u) * sin(i);

return AGECS_coords;
}
```

```
std::vector<double> calculate_GCS(std::vector<double> AGECS_coords, double
time) {
double ang_rotation = angular_vel * time;
std::vector<double> GCS_coords = { 0.0, 0.0, 0.0 };

GCS_coords[0] = cos(ang_rotation) * AGECS_coords[0] + sin(ang_rotation) *
AGECS_coords[1];
GCS_coords[1] = cos(ang_rotation) * AGECS_coords[1] - sin(ang_rotation) *
AGECS_coords[0];
GCS_coords[2] = AGECS_coords[2];

return GCS_coords;
}

std::vector<double> calculate_geodetic_coords(std::vector<double> GCS_coords)
{
std::vector<double> geodetic_coords = { 0.0, 0.0, 0.0 };

double a = 6378.136;
double e = 0.0067385254;
double x = GCS_coords[0];
double y = GCS_coords[1];
double z = GCS_coords[2];
double D = sqrt(pow(x, 2.0) + pow(y, 2.0));
if (D == 0) {
geodetic_coords[0] = 0;
geodetic_coords[1] = (PI / 2.0) * (z / fabs(z));
geodetic_coords[2] = z * sin(geodetic_coords[1]) - a * sqrt(1 - e *
pow(sin(geodetic_coords[1]), 2.0));
return geodetic_coords;
}

else {
double L = asin(y / D);
if (y < 0 && x > 0) {
geodetic_coords[0] = 2 * PI - L;
}
else if (y < 0 && x < 0) {
geodetic_coords[0] = PI + L;
}
```

```
else if (y > 0 && x < 0) {
geodetic_coords[0] = PI - L;
}
else {
geodetic_coords[0] = L;
}

if (z == 0) {
geodetic_coords[1] = 0;
geodetic_coords[2] = D - a;
return geodetic_coords;
}
else {
double r = sqrt(pow(x, 2.0) + pow(y, 2.0) + pow(z, 2.0));
double c = asin(z / r);
double p = (e * a) / (2 * r);
double s1 = 0;
double b = c + s1;
double s2 = asin((p * sin(2 * b)) / (sqrt(1 - e * pow(sin(b), 2.0))));
while (fabs(s2 - s1) >= ((0.0001 / 3600.0) * PI / 180.0)){
s1 = s2;
b = c + s1;
s2 = asin((p * sin(2 * b)) / (sqrt(1 - e * pow(sin(b), 2.0))));
}
geodetic_coords[1] = b;
geodetic_coords[2] = D * cos(b) + z * sin(b) - a * sqrt(1 - e * pow(sin(b), 2.0));
return geodetic_coords;
}
}

}


void calculate_density_and_acceleration(double H, double radial_vel, double
transversal_vel, double vel) {
std::ofstream data_input;
double density;
double S_acc;
double T_acc;
double W_acc = 0;
```

```
double acc;
double g = (6.67430 * 5.9726 * pow(10.0, 7.0)) / pow(6378.1 + H, 2.0);
bool lower_than_500 = 0;

double a_param_0_120[7] = { 26.8629, 27.4598, 28.6395, 29.6418, 30.1671,
29.7578, 30.7854 };
double a_param_1_120[7] = { -0.451674, -0.463668, -0.490987, -0.514957, -
0.527837, -0.517915, -0.545695 };
double a_param_2_120[7] = { 0.00290397, 0.002974, 0.00320649, 0.00341926,
0.00353211, 0.00342699, 0.00370328 };
double a_param_3_120[7] = { -1.06953 * pow(10.0, -5.0), -1.0753 * pow(10.0, -
5.0), -1.1681 * pow(10.0, -5.0), -1.25785 * pow(10.0, -5.0), -1.30227 * pow(10.0, -
5.0), -1.24137 * pow(10.0, -5.0), -1.37072 * pow(10.0, -5.0) };
double a_param_4_120[7] = { 2.21598 * pow(10.0, -8.0), 2.17059 * pow(10.0, -
8.0), 2.36847 * pow(10.0, -8.0), 2.5727 * pow(10.0, -8.0), 2.66455 * pow(10.0, -
8.0), 2.48209 * pow(10.0, -8.0), 2.80614 * pow(10.0, -8.0) };
double a_param_5_120[7] = { -2.42941 * pow(10.0, -11.0), -2.30249 * pow(10.0, -
11.0), -2.51809 * pow(10.0, -11.0), -2.75874 * pow(10.0, -11.0), -2.85432 *
pow(10.0, -11.0), -2.58413 * pow(10.0, -11.0), -3.00184 * pow(10.0, -11.0) };
double a_param_6_120[7] = { 1.09926 * pow(10.0, -14.0), 1.00123 * pow(10.0, -
14.0), 1.09536 * pow(10.0, -14.0), 1.21091 * pow(10.0, -14.0), 1.25009 *
pow(10.0, -14.0), 1.09383 * pow(10.0, -14.0), 1.31142 * pow(10.0, -14.0) };

double a_param_0_500[7] = { 17.8481, -2.54909, -13.9599, -23.3079, -14.7264, -
4.912, -5.40952 };
double a_param_1_500[7] = { -0.132025, 0.0140064, 0.0844951, 0.135141,
0.0713256, 0.0108326, 0.00550749 };
double a_param_2_500[7] = { 0.000227717, -0.00016946, -0.000328875, -
0.000420802, -0.000228015, -8.10546 * pow(10.0, -5.0), -3.78851 * pow(10.0, -
5.0) };
double a_param_3_500[7] = { -2.2543 * pow(10.0, -7.0), 3.27196 * pow(10.0, -
7.0), 5.05918 * pow(10.0, -7.0), 5.73717 * pow(10.0, -7.0), 2.8487 * pow(10.0, -
7.0), 1.15712 * pow(10.0, -7.0), 2.4808 * pow(10.0, -8.0) };
double a_param_4_500[7] = { 1.33574 * pow(10.0, -10.0), -2.8763 * pow(10.0, -
10.0), -3.92299 * pow(10.0, -10.0), -4.03238 * pow(10.0, -10.0), -1.74383 *
pow(10.0, -10.0), -8.13296 * pow(10.0, -11.0), 4.92183 * pow(10.0, -12.0) };
double a_param_5_500[7] = { -4.50458 * pow(10.0, -14.0), 1.22625 * pow(10.0, -
13.0), 1.52279 * pow(10.0, -13.0), 1.42846 * pow(10.0, -13.0), 5.08071 *
pow(10.0, -14.0), 3.04913 * pow(10.0, -14.0), -8.65011 * pow(10.0, -15.0) };
```

```
double a_param_6_500[7] = { 6.72086 * pow(10.0, -18.0), -2.05736 * pow(10.0, -17.0), -2.35576 * pow(10.0, -17.0), -2.01726 * pow(10.0, -17.0), -5.34955 * pow(10.0, -18.0), -4.94989 * pow(10.0, -18.0), 1.9849 * pow(10.0, -18.0) };


if (H < 500) {
lower_than_500 = 1;
}

data_input.open("C:/Users/mk170/MSF/2/code/lab 2/lab 2 graph/acceleration_data.txt");
for (int step = 0; step < 7; step++) {
if (lower_than_500) {
density = night_density * exp(a_param_0_120[step] + a_param_1_120[step] * H + a_param_2_120[step] * pow(H, 2.0) + a_param_3_120[step] * pow(H, 3.0) + a_param_4_120[step] * pow(H, 4.0) + a_param_5_120[step] * pow(H, 5.0) + a_param_6_120[step] * pow(H, 6.0));
}
else {
density = night_density * exp(a_param_0_500[step] + a_param_1_500[step] * H + a_param_2_500[step] * pow(H, 2.0) + a_param_3_500[step] * pow(H, 3.0) + a_param_4_500[step] * pow(H, 4.0) + a_param_5_500[step] * pow(H, 5.0) + a_param_6_500[step] * pow(H, 6.0));
}
S_acc = (-1.0) * SIGMA * density * vel * radial_vel * 1000.0;
T_acc = (-1.0) * SIGMA * density * vel * transversal_vel * 1000.0;
acc = sqrt(pow(S_acc, 2.0) + pow(T_acc, 2.0));
data_input << S_acc << " " << T_acc << " " << W_acc << " " << acc << std::endl;
std::cout << "  g = " << g << "; overall perturbing acceleration = " << acc << std::endl;
}
data_input << lower_than_500 << std::endl;
data_input.close();
}
```

lab.cpp

```
#include "Calculation.h"
```

```
int main() {
//можно менять, что именно подается в функцию, убирая "//" перед вызовом функции или меняя значения.
double time;
double perigee_time = 0;
double apogee_time = T / 2.0;
//calculation(perigee_time);
calculation(apogee_time);
//for (time = 0.0; time < T; time++) {
//        calculation(time);
//}
return 0;
}
```