Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики
УЧЕБНЫЙ ЦЕНТР ОБЩЕЙ ФИЗИКИ ФТФ

УНИВЕРСИТЕТ ИТМО

Группа М3113 К работе допущен

Студент Ананьин Н. Н. Работа выполнена

Преподаватель Зинчик А.А. Отчет принят

Рабочий протокол и отчет по моделированию №1

Моделирование лунного модуля.

1. Цель работы.

Написать программу, предназначенную для численного моделирования манёвров космических аппаратов в непосредственной близости безатмосферных небесных тел сферической формы (Луна).

2. Задачи, решаемые при выполнении работы.

1) Расчёт координат ОС.
2) Расчёт угла наклона корабля.
3) Расчёт расхода массы топлива корабля.
4) Расчёт скорости корабля в каждый момент времени(Δt) по уравнению Мещерского.
5) Расчёт координат корабля.
6) Получение координат ОС и корабля, при которых был передан груз.

2. Объект исследования.

Космические аппараты, в непосредственной близости безатмосферных небесных тел сферической формы (Луна).

3. Метод исследования.

Теоретическое исследование(построение формульной модели и рисунка).

Написание программы, вычисляющей параметры, необходимые для решения задачи.

4. Рабочие формулы и исходные данные

$X[0]_{ОС}$ - вводится самим человеком
$Y[0]_{ОС}$ - вводится самим человеком

R(L) = 1738000 м

G(L) = $6{,}67\times 10^{-11}$ н*м^2/кг^2

M(L) = 7.35×10^{22} кг

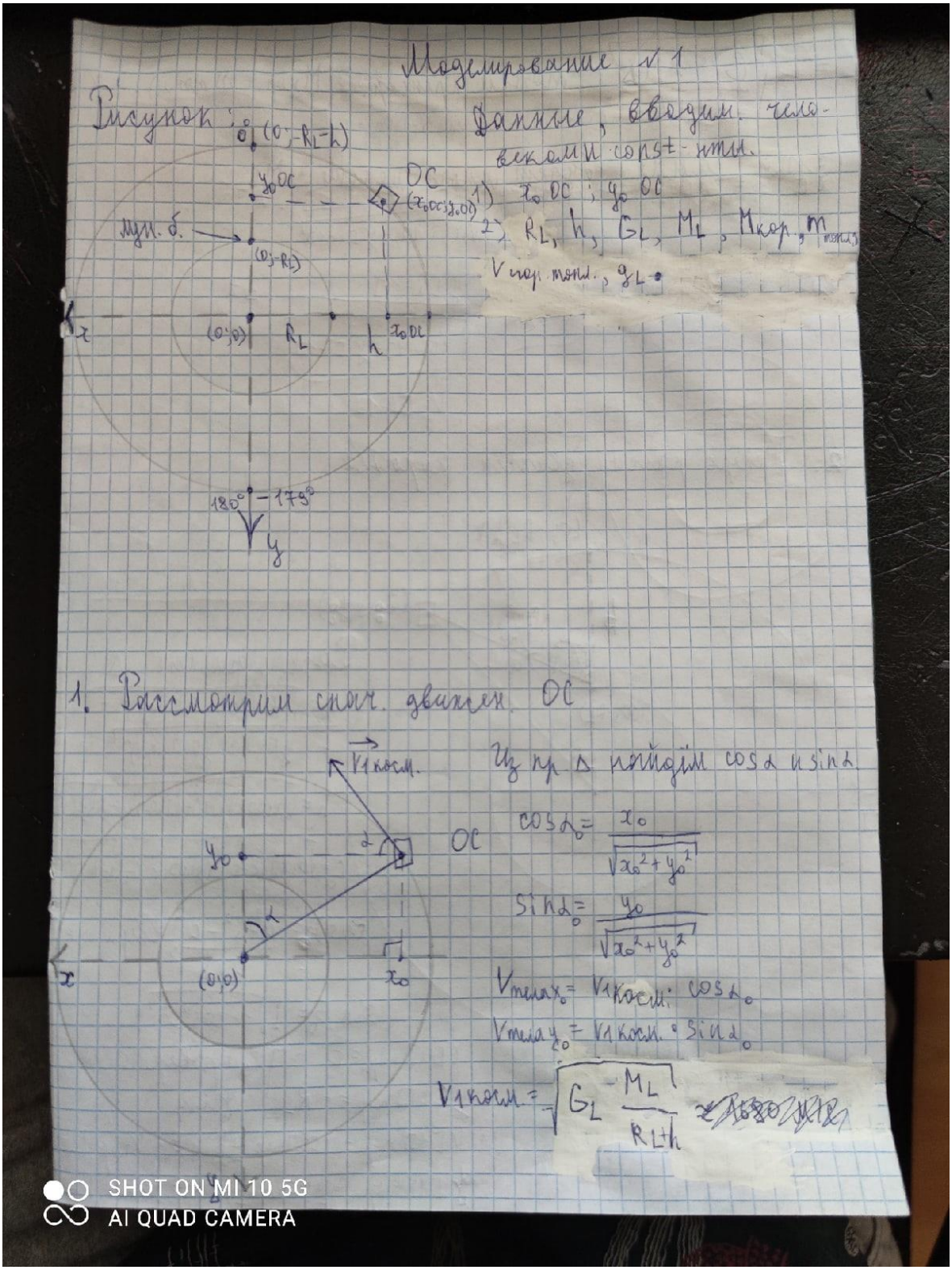
$M_{кор}$ = 2000 кг

$m_{топлива}$ = 1000 кг

g(L) = 1.62 м/с^2

$V_{сгорания\ топлива}$ = 3660 м/с

Решения в теоретическом виде:



$x_{i\,ос} = x_{предыд.(i-1)ос} + V_{тек.(i-1)ox} \cdot \Delta t$

$y_{i\,ос} = y_{предыд.(i-1)ос} + V_{тек.(i-1)oy} \cdot \Delta t$

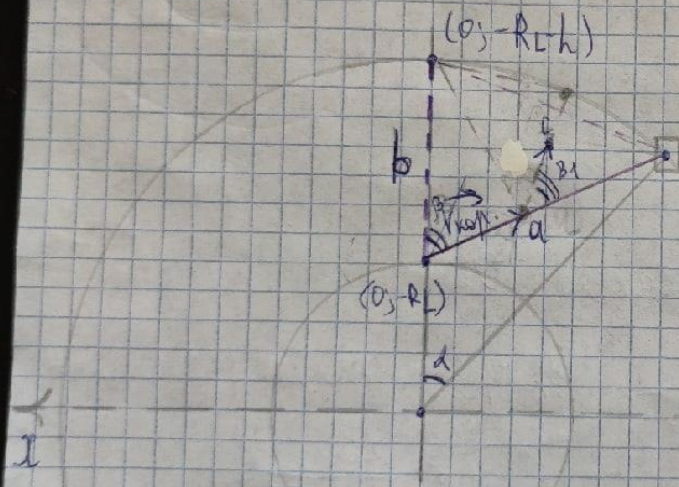
$V_{x[i-1]}$

$x[i] = x[i-1] + V_{1\,косм.} \cdot \frac{x[i-1]}{\sqrt{x^2[i-1] + y^2[i-1]}} \cdot \Delta t$

$y[i] = y[i-1] + V_{1\,косм.} \cdot \frac{y[i-1]}{\sqrt{x^2[i-1] + y^2[i-1]}} \cdot \Delta t$

$V_{y[i-1]}$

2. Теперь рассм. движение корабля



Рассм. Δ:

$a_c = \sqrt{(x[0]_{ос} - x[0]_{кор.})^2 + (y[0]_{ос} - y[0]_{кор.})^2}$

$b_0 = \sqrt{(x[0]_{кор.} - 0)^2 + (y[0]_{кор.} - (-R_L - h))^2}$

$c_0 = \sqrt{(x[0]_{ос} - 0)^2 + (y[0]_{ос} - (-R_L - h))^2}$

$\cos\beta_0 = \frac{a^2 + b^2 - c^2}{2ab}$, выраж. $\angle\beta_0$

$\sin\beta_0 = |\sqrt{1 - \cos^2\beta_0}|$

$x[0]_{кор.} = 0$
$y[0]_{кор.} = -R_L$ → проверка на угол

$x[1]_{кор.} = x[0]_{кор.} \pm \sin\beta_0 \cdot V_{кор.} \cdot \Delta t$

$y[1]_{кор.} = y[0]_{кор.} \pm \cos\beta_0 \cdot V_{кор.} \cdot \Delta t$

$$a[i] = \sqrt{(x[i-1]_{ос} - x[i]_{кор.})^2 + (y[i-1]_{ос} - y[i]_{кор.})^2}$$

$$b[i] = \sqrt{(x[i]_{ос} - x[i]_{кор.})^2 + (y[i]_{ос} - y[i]_{кор.})^2}$$

$$c[i] = \sqrt{(x[i]_{ос} - x[i-1]_{ос})^2 + (y[i]_{ос} - y[i-1]_{ос})^2}$$

$$\cos\beta[i] = \frac{a[i]^2 + b[i]^2 - c[i]^2}{2\,a[i]\,b[i]}$$

Находим β_1

$\gamma_1 = \beta_1 + \beta_0$

$$x[i]_{кор.} = x[i-1]_{кор.} \pm |\sin\gamma[i-1] \cdot v_{кор.}[i-1]| \cdot \Delta t$$

$$y[i]_{кор.} = y[i-1]_{кор.} \pm |\cos\gamma[i-1] \cdot v_{кор.}[i-1]| \cdot \Delta t$$

Проверка встречи кор. и ОС - $b[i] \le 50$ м

$v_{кор.}$ - ур. Мещерского, общ. масса тела (М).

$$M[0] = M_{кор.} + m_{топл.}[0]$$

Δm - скор. изм. м.

$$m[i] = m[i-1] - \Delta m \cdot \Delta t$$

$$M[i] = M_{кор.} + m[i] = M_{кор.} + m[i-1] + \Delta m\,\Delta t$$

Ур-е Мещерского:

$$M_{тела}\frac{\vec{V}}{\Delta t} = \vec{F}_{ст.} + \left(-\vec{V}_{кор.т.}\frac{\Delta m}{\Delta t}\right)$$

$\le F_{реакт.}$

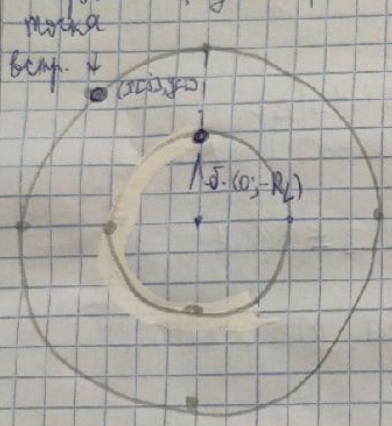
$F_{ст.} = M_{тела} \cdot g_L$, берем $F_т$, т.к. $M_{тела} < M_L$

$$V[0] = \left|\frac{M[0] \cdot g_L - V_{кор.т.}\frac{\Delta m}{\Delta t} \cdot \Delta t}{M[0]}\right| = \left|\frac{M[0]\,g_L \cdot \Delta t - V_{кор.}\,\Delta m}{M[0]}\right|$$

$$V_{кор.}[i] = \left| \frac{M[i] g_L \Delta t - V_{отор.т.} \Delta m}{M[i]} \right|$$

Тела пересек., запоминаем $x_{кор}[j]$, $y_{кор.}[j]$, $M[j]$ - остаток. топливо.

Если $M[i]$ - досл., тогда просто даём ему вращ. по орбите, до пересеч. с Кормильцем, проведен. из лунной базы.



Т.к. нам надо Косад. тело с точностью до 5 м, а почти все расстояния и перемещ. измер. в км => если сделать тело не вечн., то будет огромная погреш. => ~~ошибка~~

$$\Rightarrow V_{верт.посадки} = \left| \frac{M[j] g_L \Delta t - V_{отор.т.} \Delta m}{M[j]} \right| \frac{\text{км}}{\text{с}}$$

5. Код программы, проводящий вычисления.

```
[59]: import math

R_MOON = 1737100
M_MOON = 7.36 * (10**22)
G_MOON = 6.67 * (10**-11)
H_ORBIT = 53000
M_BODY = 2000
M_FUEL = 1000
U_FUEL_BURN = 3660
g_MOON = 1.62
DELTA_M = int(input())
```

```
5
```

```
def kost_alpha(x, y):
    return math.sqrt(x**2 + y**2)

class Station:

    def __init__(self, first_coord_x, first_coord_y):
        self.first_coord_x = first_coord_x
        self.first_coord_y = first_coord_y
        self.u_space_first  = math.sqrt(G_MOON * (M_MOON
                                / (kost_alpha(self.first_coord_x, self.first_coord_y))))
        self.coords_x = [first_coord_x]
        self.coords_y = [first_coord_y]
        self.speed_x = []
        self.speed_y = []
        self.times = []
        for i in range(0, 10001, 2):
            self.times.append(i)

    def calculate_information(self):
        for i in range(4999):

            self.speed_x.append(self.u_space_first *
                                (self.coords_y[i] / (kost_alpha(self.coords_x[i], self.coords_y[i]))))
            self.speed_y.append(self.u_space_first *
                                 (self.coords_x[i] / (kost_alpha(self.coords_x[i], self.coords_y[i]))))

            self.coords_x.append(self.coords_x[i] + self.speed_x[i] * 2)
            self.coords_y.append(self.coords_y[i] + self.speed_y[i] * 2)

    @staticmethod
    def get_coords(self, index):
        return (self.coords_x[index - 1], self.coords_y[index - 1])
```

```
class Body(Station):

    def __init__(self, first_body_x = 0, first_body_y = -R_MOON,
                 first_station_x = 0, first_station_y = 0):

        self.coords_x = [0] * 5000
        self.coords_y = [0] * 5000
        self.coords_x[0] = first_body_x
        self.coords_y[0] = first_body_y
        self.first_station_x = first_station_x
        self.first_station_y = first_station_y
        self.a_vector = [0] * 5000
        self.b_vector = [0] * 5000
        self.c_vector = [0] * 5000
        self.speed = [0] * 5000
        self.times = []
        self.mesh = [0] * 5000
        self.mesh[0] = 3000

        for i in range(2, 10001, 2):
            self.times.append(i)

        for i in range(1, 5000):
            self.mesh.append(self.mesh[i - 1] - DELTA_M * 2)
```

```
def calculate_information(self):
    station = Station(self.first_station_x, self.first_station_y)
    station.calculate_information()
    for i in range(4999):

        x_st, y_st = station.get_coords(station, i)
        x_st_past, y_st_past = station.get_coords(station, i - 1)

        if self.b_vector[i] <= 50:
            print(self.b_vector[i])
            print("x_oc, y_oc:", x_st, y_st)
            print("x_body, y_body:", x_st - math.sqrt(124.1339130130), y_st - math.sqrt(124.130310301931))
            print("FUEL:", self.mesh[i] - 2645)
            break

        if i != 0:
            gamma = self.gamma_by_index(self, i)
```

```
         if abs(gamma)<=0.5:
             self.speed[i] = abs((self.mesh[i] * g_MOON * 2  - U_FUEL_BURN * DELTA_M) / self.mesh[0])
             #добавил умножение на 2, тк не увидел в формуле учета временного промежутка
             self.coords_x[i] = self.coords_x[i - 1] +  self.speed[i] * abs(math.cos(self.gamma_by_index(self, i - 1)))

         else:
             self.coords_x[i] = self.coords_x[i - 1] -  self.speed[i] * abs(math.cos(self.gamma_by_index(self, i - 1)))
             self.speed[i] = abs((self.mesh[i] * g_MOON * 2  + U_FUEL_BURN * DELTA_M) / self.mesh[0])

         if gamma<=0:
             self.coords_y[i] = self.coords_y[i - 1] - self.speed[i] * abs(math.sin(self.gamma_by_index(self, i - 1)))
         else:
             self.coords_y[i] = self.coords_y[i - 1] + self.speed[i] * abs(math.sin(self.gamma_by_index(self, i - 1)))



         self.a_vector[i] = (math.sqrt((x_st - self.coords_x[i - 1])**2
                                       + (y_st - self.coords_y[i - 1])**2))

         self.b_vector[i] = (math.sqrt((x_st - self.coords_x[i])**2
                                       + (self.coords_x[i] - self.coords_y[i]))**2)

         self.c_vector[i] = (math.sqrt((x_st - self.coords_x[i - 1])**2
                                       + (y_st - self.coords_y[i - 1])**2))
```

```
        elif i == 0:
            print(type(self.coords_x[0]))
            self.a_vector[i] = (math.sqrt((x_st - self.coords_x[0])**2
                                          + (y_st - self.coords_y[0])**2))
            self.b_vector[i] = (math.sqrt((self.coords_x[0] - 0)**2
                                          + (self.coords_y[0] + R_MOON + H_ORBIT)**2))
            self.c_vector[i] = (math.sqrt((x_st - 0)**2
                                          + (y_st + R_MOON + H_ORBIT)**2))
            cos = self.cos_by_index(self, i)
            if abs(math.acos(cos)) <= 0.5:
                self.speed[0] = abs((self.mesh[0] * g_MOON * 2 - U_FUEL_BURN * DELTA_M) / self.mesh[0])
                self.coords_x[1] = self.coords_x[0]  \
                                   - abs(self.speed[0] * math.sin(math.acos(cos)) * 2)
                self.coords_y[1] = self.coords_y[0]  \
                                   - abs(self.speed[0] * cos * 2)
            else:
                self.speed[0] = abs((self.mesh[0] * g_MOON * 2 + U_FUEL_BURN * DELTA_M) / self.mesh[0])
                self.coords_x[1] = self.coords_x[0] + abs(self.speed[0] * math.sin(math.acos(cos)) * 2)
                self.coords_y[1] = self.coords_y[0] + abs(self.speed[0] * cos * 2)


    @staticmethod
    def cos_by_index(self, index):
        return (self.a_vector[index]**2 + self.b_vector[index]**2 - self.c_vector[index]**2) \
               / (2 * self.a_vector[index] * self.b_vector[index])

    @staticmethod
    def gamma_by_index(self, index):
        return  math.acos(self.cos_by_index(self, index - 1)) - math.acos(self.cos_by_index(self, index))
```

```
: elem = Body(0, -R_MOON, -1762100, -1762100).calculate_information()

  x_oc, y_oc: -11684738.83596857 -11684738.83596857
  x_body, y_body: -11684749.977508545 -11684749.977346865
  FUEL: 355
```

6. Окончательные результаты.

Числа, выводимые программой.

7. Выводы и анализ результатов работы.

По завершению работы программы мы получаем координаты корабля и массу оставшегося топлива, так как нам нужно посадить корабль с очень большой точностью(попасть в окружность радиуса 5 м), а большинство перемещений и расстояний в задаче выражены в км, то получаем очень большую погрешность и сложность попадания в эту окружность, во всех вариантах кроме одного – рассмотрим этот вариант.

Чтобы точно посадить тело – будет сажать его строго по вертикали(сразу же выполнится условие про горизонтальную скорость), соответственно : 1)проведём вертикаль из лунной базы, задав её уравнением 2) пустим корабль под неким углом, до пересечения с этой вертикалью 3) посадим корабль.