**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**"Уфимский университет науки и технологий"**

**Кафедра** Высокопроизводительных вычислительных технологий и систем

**Дисциплина:** Математическое моделирование

**Отчет по лабораторной работе № 1**

**Тема:** «Компьютерное моделирование движения космических тел»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Группа ПМ-453 | Фамилия И.О. | Подпись | Дата | Оценка |
| Студент | Садыков Р.А. |  |  |  |
| Принял | Лукащук С.Ю. |  |  |  |

**Уфа 2024**

**Цель работы:** получить навык численного расчета траекторий движения космических тел под действием гравитационных сил.

**Задание на лабораторную работу**

**Задача I.** Рассматривается динамика трех разновеликих небесных тел: звезды, планеты и ее спутника. В качестве примера рассматривается Солнечная система. Масса Солнца . Параметры двух других тел выбираются в соответствии с индивидуальным номером варианта из таблицы.

1) Составить уравнения движения второго и третьего тела в системе отсчета, связанной с первым (самым массивным) телом. Предполагается, что движение всех тел происходит в одной плоскости.

2) Написать программу численного интегрирования составленных уравнений движения и построить траектории движения тел. В качестве начальных условий принять следующие: все тела находятся на одной прямой, вектора скоростей движения второго и третьего тела сонаправлены. Расстояния между первым и вторым, а также вторым и третьим телами приведены в таблице. Там же указаны значения начальных скоростей второго и третьего тела. Исследовать отклонение орбиты планеты и спутника от круговой с течением времени, а также характер изменения их модулей скорости

**Задача II.** На круговой орбите высотой Н второго тела находится космический корабль. В некоторый момент времени его двигатели включаются и работают в течение времени Т выводя корабль на новую орбиту, пересекающую орбиту третьего тела. Вектор тяги двигателя в любой момент времени направлен по касательной к траектории движения. Определить местоположение космического корабля на первоначальной орбите в момент включения двигателя из условия минимума массы топлива, необходимой для доставки на поверхность третьего тела полезного груза массой М0. Местоположение определяется относительно прямой, соединяющей центры второго и третьего тел. Масса корабля складывается из массы топлива, полностью выгорающего за время Т, массы конструкции (0.025 стартовой массы) и массы полезной нагрузки Ма. В конце активного участка траектории (через время 7) происходит отделение полезного груза, который движется далее только под действием гравитационных сил. Скорость полезного груза при достижении поверхности третьего тела не ограничивается.

**Практическая часть**

**Задача I.**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| № вар | Параметры второго тела | | | | Параметры третьего тела | | | |
| , км | , км | , млн. км | , км/с | , кг | , км | , тыс. км | , км/с |
| 9 |  | 71500 | 780 | 13 |  | 2634 | 1070 | 10.9 |

В системе действует сила притяжения космических тел, которая подчиняется закону всемирного тяготения:

– гравитационная постоянная, – массы космических тел.

Уравнение движения:

Для системы планета-спутник уравнения движения, относительно Солнца будут иметь вид:

Далее составим систему ОДУ:

Решив данную задачу Коши с помощью Python-библиотеки SciPy, получаем следующие траектории небесных тел:

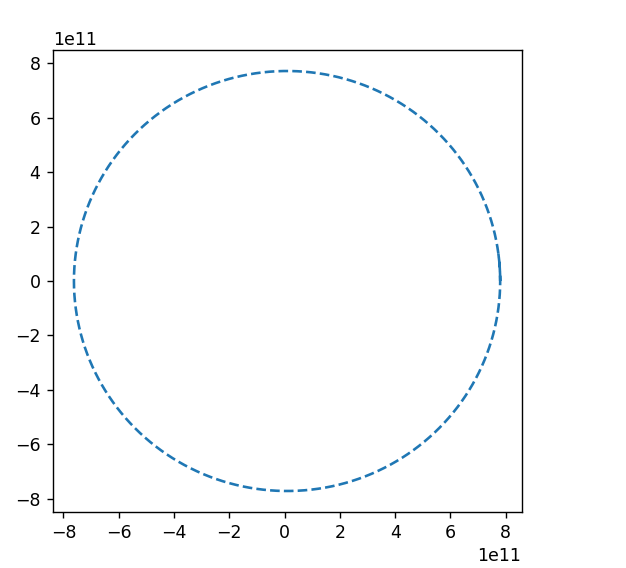


Рисунок 1 – Траектория движения планеты вокруг Солнца

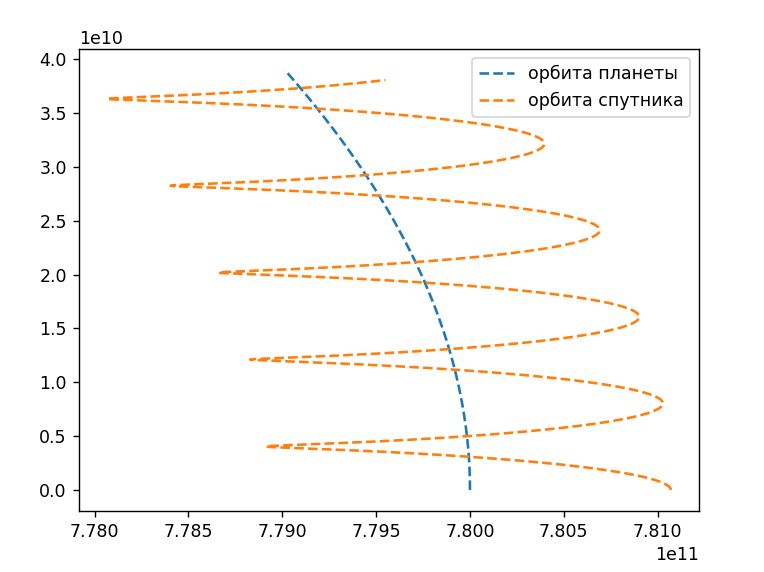


Рисунок 2 – Траектория движения планеты и спутника вокруг Солнца

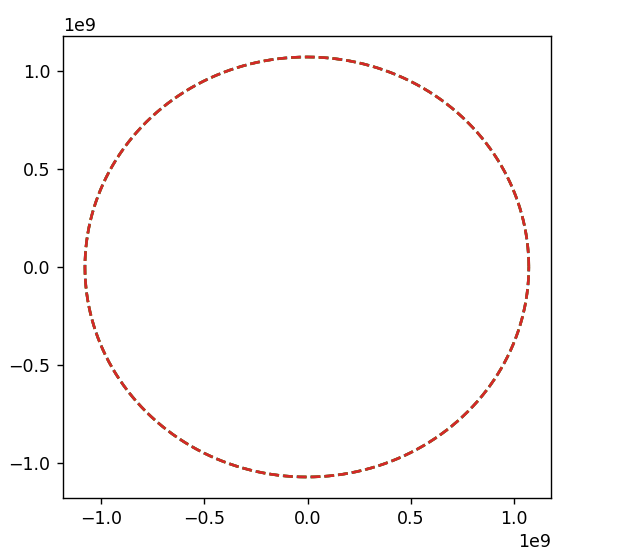


Рисунок 3 Траектория движения спутника вокруг планеты

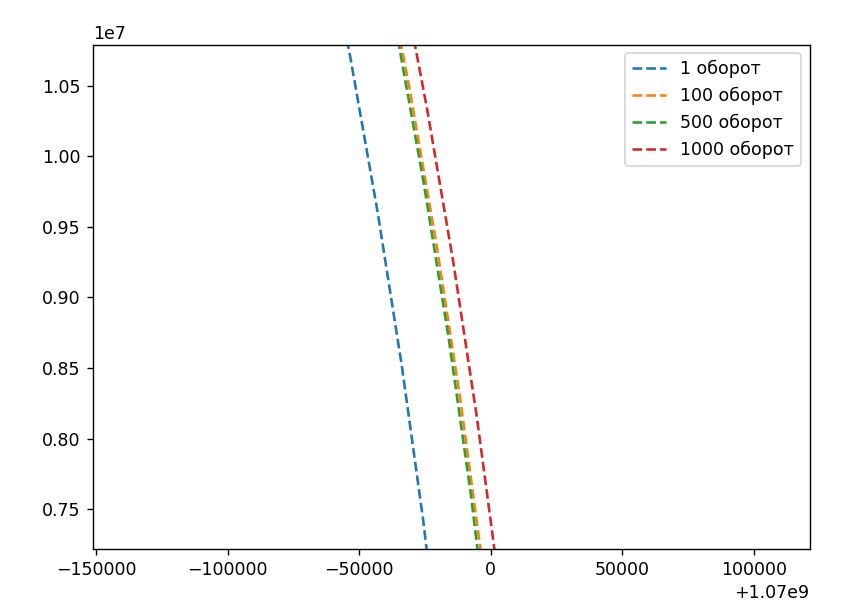


Рисунок 4 Орбита спутника после 1, 100, 500 и 1000 оборотов

Максимальная скорость планеты 13,31 км/с, минимальная – 13 км/с

**Задача II.**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| № варианта | H, км |  |  | Характеристики топлива | | |
| Горючее | Окислитель | Скорость истечения, м/с |
| 6 | 900 | 1600 | 75 | Аммиак(жидкий) | Кислород (жидкий) | 3170 |

Для описания движения точки переменной массы воспользуемся уравнением Мещерского:

где – скорость ракеты, – скорость истечения.

– общая масса ракеты, – масса конструкции ракеты, – масса топлива, – масса полезной нагрузки.

Масса ракеты вычисляется по формуле:

Таким образом, в систему ОДУ добавляются уравнения:

где – координаты ракеты.

Начальное положение и скорость ракеты на орбите заданной высоты:

Для решения задачи была использована Python-библиотека SciPy, в которой реализована схема Рунге-Кутты 4 порядка. Так как интервал времени достаточно мал, планета считалась фиксированной точкой. В качестве начального времени берется момент t0, когда спутник сделал 1000 оборотов вокруг планеты, а первое, второе и третье тело находятся на одной прямой. Для поиска оптимального угла использовался метод Нелдера-Мида. Наименьшее количество топлива для доставки полезного груза на спутник составило 12067 кг при угле 353,8. Время полета составляет 370588 секунд.

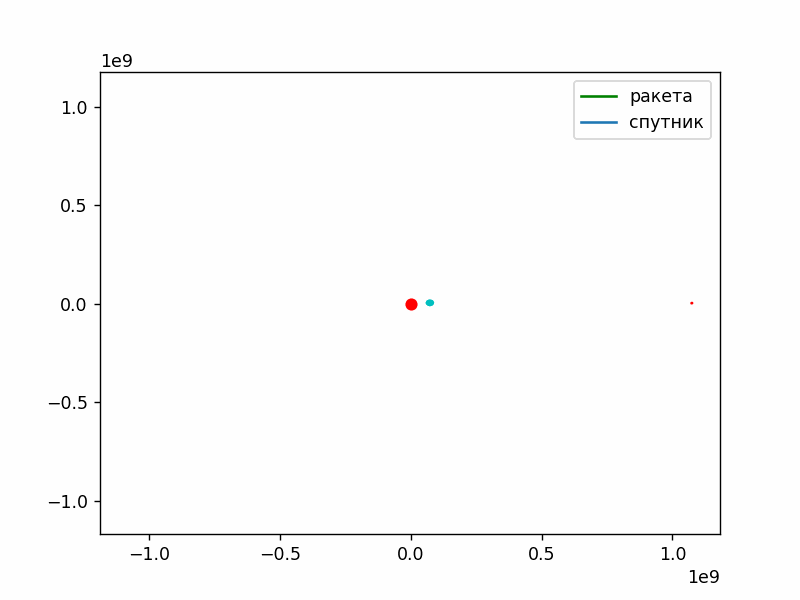


Рисунок 5 – Траектория движения спутника и ракеты до момента попадания полезного груза на спутник

**Вывод**

В ходе данной лабораторной работы были получены навыки численного расчета траекторий движения небесных тел под действием гравитационных сил. Построены траектории движения планеты, спутника и ракеты, которые получены в результате решения системы дифференциальных уравнений.

Также для построения траектории движения ракеты были произведены расчеты необходимого количества топлива для доставки груза на спутник. В результате минимальное необходимое количество топлива – 12067 кг при угле 353,8. Время полета составило 370588 секунд.

**Приложение**

from scipy import integrate  
from scipy.integrate import solve\_ivp  
import matplotlib.pyplot as plt  
import numpy as np  
from matplotlib import animation  
import math  
import random  
from z2 import \*  
ox1,oy1,ox100,oy100,ox500,oy500,ox1000,oy1000=[],[],[],[],[],[],[],[]  
orbitsx=[[] for i in range(1100)]  
orbitsy=[[] for i in range(1100)]  
def event(x,y):  
 if not hasattr(event, "counter"):  
 event.counter = 0  
 if not hasattr(event, "lastvalue"):  
 event.lastvalue = y[3]-y[1]  
 if not hasattr(event, "xstop"):  
 event.xstop = -1  
 orbitsx[int(event.counter/2)].append(y[2]-y[0])  
 orbitsy[int(event.counter / 2)].append(y[3] - y[1])  
 if (y[3]-y[1])\*event.lastvalue<0:  
 event.counter += 1  
 if y[3] - y[1]!=0:  
 event.lastvalue = y[3] - y[1]  
 if ((event.counter>=2000 and (abs(y[0]/math.sqrt(y[0]\*\*2+y[1]\*\*2)-(y[2]-y[0])/math.sqrt((y[2]-y[0])\*\*2+(y[3]-y[1])\*\*2))<0.001) and  
 (abs(y[1]/math.sqrt(y[0]\*\*2+y[1]\*\*2)-(y[3]-y[1])/math.sqrt((y[2]-y[0])\*\*2+(y[3]-y[1])\*\*2))<0.001)) and  
 (math.sqrt(y[0]\*\*2+y[1]\*\*2) < math.sqrt(y[2]\*\*2+y[3]\*\*2))):  
 event.xstop=x  
 return x-event.xstop  
  
event.terminal = True  
m1,m2,m3=2\*(10\*\*30),1.9\*(10\*\*27),1.5\*(10\*\*23)  
G=6.67\*(10\*\*-11)  
def F(t,y):  
 r12x,r12y,r13x,r13y,v2x,v2y,v3x,v3y=y  
 r2=math.sqrt(r12x\*\*2+r12y\*\*2)  
 r3 = math.sqrt(r13x \*\* 2 + r13y \*\* 2)  
 r23x=r13x-r12x  
 r23y=r13y-r12y  
 r=math.sqrt(r23x\*\*2+r23y\*\*2)  
 return [v2x,v2y,v3x,v3y,  
 G\*(-(m1/(r2\*\*3)\*r12x)+(m3/(r\*\*3)\*r23x)),  
 G\*(-(m1/(r2\*\*3)\*r12y)+(m3/(r\*\*3)\*r23y)),  
 G\*(-(m1/(r3\*\*3)\*r13x)-(m2/(r\*\*3)\*r23x)),  
 G\*(-(m1/(r3\*\*3)\*r13y)-(m2/(r\*\*3)\*r23y))]  
  
t\_span=(0,1\*3000000000)  
tau=2000  
y0=[780\*(10\*\*9),0,780\*(10\*\*9)+1070\*(10\*\*6),0,0,13000,0,23900]  
solution=solve\_ivp(F,t\_span,y0, max\_step=tau, atol=1, rtol=1,method="RK45",events=event)  
  
  
plt.plot(orbitsx[0], orbitsy[0],linestyle='--',label='1 оборот')  
plt.plot(orbitsx[99], orbitsy[99],linestyle='--',label='100 оборот')  
plt.plot(orbitsx[499], orbitsy[499],linestyle='--',label='500 оборот')  
plt.plot(orbitsx[999], orbitsy[999],linestyle='--',label='1000 оборот')  
  
plt.legend()  
plt.show()  
print('z2')  
  
if False:  
 z2(solution.y[2][-1],solution.y[3][-1],  
 solution.y[6][-1]-solution.y[4][-1],solution.y[7][-1]-solution.y[5][-1],  
 10,solution.y[0][-1],solution.y[1][-1],3.4,5300,1)

from scipy import integrate  
from scipy.integrate import solve\_ivp  
import matplotlib.pyplot as plt  
from matplotlib import animation  
import random  
from matplotlib.animation import FuncAnimation  
import math  
import numpy as np  
  
m1,m2,m3=2\*(10\*\*30),1.9\*(10\*\*27),1.5\*(10\*\*23)  
G=6.67\*(10\*\*-11)  
m0=75  
mt=4950  
T=1600  
koef=0.001  
u=3170  
Rpl = 71500 \* (10 \*\* 3)  
H = 900 \* (10 \*\* 3)  
Rsat=2634\*(10\*\*3)  
def m(t):  
 if t>=T:  
 return m0  
 else:  
 return (m0+mt)/(1-koef)-mt\*t/T  
def dm(t):  
 if t>=T:  
 return 0  
 else:  
 return -mt / T  
  
r12x=0  
r12y=0  
  
def event2(t,y):  
 if not hasattr(event2, "counter"):  
 event2.stop = -1  
 if event2.stop == -1 and (math.sqrt((y[2] - y[0]) \*\* 2 + (y[3] - y[1]) \*\* 2) - Rsat) < 0:  
 event2.stop=t  
 return t-event2.stop  
event2.terminal = True  
def F2(t,y):  
 rx,ry,r13x,r13y,vx,vy,v3x,v3y=y  
 r=math.sqrt(rx\*\*2+ry\*\*2)  
 v = math.sqrt(vx \*\* 2 + vy \*\* 2)  
 r2=math.sqrt((rx-r12x)\*\*2+(ry-r12y)\*\*2)  
 r3 = math.sqrt((rx - r13x) \*\* 2 + (ry - r13y) \*\* 2)  
 r13=math.sqrt(r13x\*\*2+r13y\*\*2)  
 r23x=(r13x - r12x)  
 r23y=(r13y - r12y)  
 r23 = math.sqrt((r13x - r12x) \*\* 2 + (r13y - r12y) \*\* 2)  
  
 return [vx,vy,v3x,v3y,  
 (-u \*dm(t)/m(t) \* vx / v+G\*(-m1\*rx/(r\*\*3)-m2\*(rx-r12x)/(r2\*\*3)-m3\*(rx-r13x)/(r3\*\*3))),  
 (-u \* dm(t)/m(t) \* vy / v+ G \* (-m1 \* ry / (r \*\* 3)- m2 \* (ry - r12y) / (r2 \*\* 3) - m3 \* (ry - r13y) / (r3 \*\* 3))),  
 G \* (-(m1 / (r13 \*\* 3) \* r13x) - (m2 / (r23 \*\* 3) \* r23x)),  
 G \* (-(m1 / (r13 \*\* 3) \* r13y) - (m2 / (r23 \*\* 3) \* r23y))  
 ]  
def z2(r13x0,r13y0,v3x0,v3y0,tau,r12x0,r12y0,fi,m):  
 global r12x,r12y  
 global mt  
 print(m)  
 mt=m  
 r12x=r12x0  
 r12y=r12y0  
 r3x=r13x0-r12x  
 r3y=r13y0-r12y  
 r3=math.sqrt(r3x\*\*2+r3y\*\*2)  
 v0=1.0\*math.sqrt(G\*m2/(Rpl+H))  
 rx0=(Rpl+H)\*(r3x\*math.cos(fi)-r3y\*math.sin(fi))/r3  
 ry0=(Rpl+H)\*(r3x\*math.sin(fi)+r3y\*math.cos(fi))/r3  
 r0=math.sqrt(rx0\*\*2+ry0\*\*2)  
 vx0=-v0\*ry0/r0  
 vy0=v0\*rx0/r0  
 rx0=r12x+rx0  
 ry0=r12y+ry0  
 s0=[rx0,ry0,r13x0,r13y0,vx0,vy0,v3x0,v3y0]  
 t\_span=(0,800000)  
 satel=solve\_ivp(F2,t\_span,s0, max\_step=tau, atol=1, rtol=1,method="RK45",events=event2)  
 return satel  
def optimization(solution,m):  
 R = [math.sqrt((solution.y[2][i] - solution.y[0][i]) \*\* 2 + (solution.y[3][i] - solution.y[1][i]) \*\* 2) - Rsat for i in  
 range(len(solution.y[0]))]  
 Res = math.sqrt((solution.y[2][-1] - solution.y[0][-1]) \*\* 2 + (solution.y[3][-1] - solution.y[1][-1]) \*\* 2) - Rsat  
 if Res < 0:  
 Res = 0  
 Res += m  
 return max(min(R),0)+m  
def show\_animation(satel,r13x0,r13y0,tau):  
 r13=math.sqrt(r13x0\*\*2+r13y0\*\*2)  
 alpha=math.acos(r13x0/r13)  
 s = int(300 / tau)  
 kr = 10000000  
 sx = [(satel.y[0][s \* i] - r12x)\*math.cos(alpha)-(satel.y[1][s \* i] - r12y)\*math.sin(alpha) for i in range(int(len(satel.y[0]) / s))]  
 sy = [(satel.y[0][s \* i] - r12x)\*math.sin(alpha)+(satel.y[1][s \* i] - r12y)\*math.cos(alpha) for i in range(int(len(satel.y[0]) / s))]  
 lx = [(satel.y[2][s \* i] - r12x)\*math.cos(alpha)-(satel.y[3][s \* i] - r12y)\*math.sin(alpha) for i in range(int(len(satel.y[0]) / s))]  
 ly = [(satel.y[2][s \* i] - r12x)\*math.sin(alpha)+(satel.y[3][s \* i] - r12y)\*math.cos(alpha) for i in range(int(len(satel.y[0]) / s))]  
 fig, ax = plt.subplots()  
 line, = ax.plot(sx, sy, color='g',label='ракета')  
 line2, = ax.plot(lx, ly,label='спутник')  
 plt.plot(0, 0, 'ro')  
 plt.legend()  
 moon = plt.Circle((r13x0 - r12x, r13y0 - r12y), Rsat, color='r')  
 rocket = plt.Circle((r13x0 - r12x, r13y0 - r12y), 5\*Rsat, color='c')  
 p = ax.add\_patch(moon)  
 q = ax.add\_patch(rocket)  
 def animate(i):  
 line.set\_xdata(sx[0:i])  
 line.set\_ydata(sy[0:i]) # update the data  
 line2.set\_xdata(lx[0:i]) # update the data  
 line2.set\_ydata(ly[0:i]) # update the data  
 moon.set\_center((lx[i], ly[i]))  
 rocket.set\_center((sx[i], sy[i]))  
 return moon, rocket, line, line2,  
 ani = animation.FuncAnimation(fig, animate, np.arange(1, int(len(satel.y[0])/s)), interval=1)  
 plt.show()  
 ani.save('myAnimation.gif', writer='pillow', fps=30)  
 print('gif saved')  
  
def NelderMid():  
 a1=random.random()\*math.pi\*2  
 mt1=random.randint(1,20000)  
 tau=0.1  
 a2 = random.random() \* math.pi \* 2  
 mt2 = random.randint(1, 20000)  
 a3 = random.random() \* math.pi \* 2  
 mt3 = random.randint(1, 20000)  
 res1 = optimization(z2(-166864782814.52066, -751687780697.0953, 10608.237381615552, -2321.8734688953678, tau,  
 -166632056766.0584, -750639289651.4109, a1, mt1),mt1)  
 res2 = optimization(z2(-166864782814.52066, -751687780697.0953, 10608.237381615552, -2321.8734688953678, tau,  
 -166632056766.0584, -750639289651.4109, a2, mt2),mt2)  
 res3 = optimization(z2(-166864782814.52066, -751687780697.0953, 10608.237381615552, -2321.8734688953678, tau,  
 -166632056766.0584, -750639289651.4109, a3, mt3),mt3)  
  
 for i in range(15):  
 if res2==max([res1,res2,res3]):  
 res2,res1=res1,res2  
 a2,a1=a1,a2  
 mt2,mt1=mt1,mt2  
 if res2<res3:  
 res2, res3 = res3, res2  
 a2, a3 = a3, a2  
 mt2, mt3 = mt3, mt2  
 if res3==max([res1,res2,res3]):  
 res3,res1=res1,res3  
 a3,a1=a1,a3  
 mt3,mt1=mt1,mt3  
 if res2<res3:  
 res2, res3 = res3, res2  
 a2, a3 = a3, a2  
 mt2, mt3 = mt3, mt2  
 a0 = (a2 + a3)/2  
 mt0 = (mt2 + mt3)/2  
 an=(a2+a3)-a1  
 mtn=(mt2+mt3)-mt1  
 an = max(0, an)  
 mtn = max(0, mtn)  
 res=optimization(z2(-166864782814.52066, -751687780697.0953, 10608.237381615552, -2321.8734688953678, tau,  
 -166632056766.0584, -750639289651.4109, an, mtn),mtn)  
 if res<max(res2,res3):  
 an2 = a0+2\*(an-a0)  
 mtn2 = mt0+2\*(mtn - mt0)  
 an2,mtn2 = max(0, an),max(0, mtn)  
 res4 = optimization(z2(-166864782814.52066, -751687780697.0953, 10608.237381615552, -2321.8734688953678, tau,  
 -166632056766.0584, -750639289651.4109, an2, mtn2),mtn2)  
 if res4<min(res2,res3):  
 an,mtn,res=an2,mtn2,res4  
 if res>max(res3,res2):  
 an = a0 + 0.5 \* (a1 - a0)  
 mtn = mt0 + 0.5 \* (mt1 - mt0)  
 an, mtn= max(0, an),max(0, mtn)  
 res = z2(-166864782814.52066, -751687780697.0953, 10608.237381615552, -2321.8734688953678, tau,  
 -166632056766.0584, -750639289651.4109, an, mtn)  
 if res<max(res3,res2):  
 a1,mt1,res1 = an,mtn,res  
 else:  
 a1 = a3+0.5\*(a1-a3)  
 mt1 = mt3+0.5\*(mt1-mt3)  
 a2 = a3 + 0.5 \* (a2 - a3)  
 mt2= mt3 + 0.5 \* (mt2 - mt3)  
 a1 = max(0, a1)  
 mt1 = max(0, mt1)  
 a2 = max(0, a2)  
 mt2 = max(0, mt2)  
 res1 = z2(-166864782814.52066, -751687780697.0953, 10608.237381615552, -2321.8734688953678, tau,  
 -166632056766.0584, -750639289651.4109, a1, mt1)  
 res2 = z2(-166864782814.52066, -751687780697.0953, 10608.237381615552, -2321.8734688953678, tau,  
 -166632056766.0584, -750639289651.4109, a2, mt2)