计算物理-第七次作业

吴远清 2018300001031

第一题

数据设置

本题目中使用的各星体质量为:

星体	质量
木星	$1.90 \times 10^{27}\ \mathrm{kg}$
太阳	$1.9891 \times 10^{30}\ \mathrm{kg}$
地球	$5.95 \times 10^{24}\ \mathrm{kg}$

各行星到太阳的距离:

星体	距离
木星	$7.7833 \times 10^{11}\ \mathrm{m}$
地球	$1.496 \times 10^{11}\ \mathrm{m}$

各行星的速度:

星体	速度
木星	$13.07\ \mathrm{km \cdot s^{-1}}$
地球	$30\ \mathrm{km \cdot s^{-1}}$
太阳	TBD

环境与常数

引入相关包并设置常数

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm


G = 6.67259*10**-11
Ms = 1.9891*10**30
Mj = 1.90*10**27
Me = 5.95*10**24
Lsj = 7.7833*10**11
Lse = 1.496*10**11
```

其中,计算均由`numpy`实现,并引入了`tqdm`库来实现进度显示

各常数均设置为SI单位制.

计算类的实现

```
class multiBodiesSolver():
    def __init__(self, dt, mass, coords, velocs, name):
        self.__n = len(mass)
        self.__mass = np.array(mass)
        self.__x0 = []
        self.__y0 = []
        self.__idx = np.array(range(self.__n))
        for i in coords:
            self.__x0.append(i[0])
            self.__y0.append(i[1])
        self.__vx0 = []
        self.__vy0 = []
        for i in velocs:
            self.__vx0.append(i[0])
            self.__vy0.append(i[1])
        self.__dt = dt
        self.__name = name
    def setTotalPToZero(self, idx):
        totPx = np.sum(self.__mass * self.__vx0)
        totPy = np.sum(self.__mass * self.__vy0)
        self.__vx0[idx] -= totPx / self.__mass[idx]
        self.__vy0[idx] -= totPy / self.__mass[idx]
    def __Solver(self, idx):
        x,y = self.x[idx - 1], self.y[idx - 1]
        for i in range(self.__n):
            oIdx = (self.__idx != i)
            r3 = np.power(np.sqrt(np.power(x[oIdx] - x[i],2) + np.power(y[oIdx]
- y[i],2)),3)
            self.vx[idx][i] = self.vx[idx - 1][i] + np.sum(G * self.__dt *
self.__mass[oIdx]*(x[oIdx] - x[i])/r3)
            self.vy[idx][i] = self.vy[idx - 1][i] + np.sum(G * self.__dt *
self.__mass[oIdx]*(y[oIdx] - y[i])/r3)
        self.x[idx] = self.x[idx - 1] + self.vx[idx] * self.__dt
        self.y[idx] = self.y[idx - 1] + self.vy[idx] * self.__dt
        self.t.append(self.t[-1] + self.__dt)
    def showSystemInfo(self, idx):
        x,y = self.x[idx], self.y[idx]
        totV = 0
        totK = np.sum(0.5*self.__mass * np.power(self.__vx0, 2))
        totPx = np.sum(self.__mass * self.__vx0)
        totPy = np.sum(self.__mass * self.__vy0)
        for i in range(self.__n):
            oIdx = (self.__idx != i)
            r = np.sqrt(np.power(x[oIdx] - x[i],2) + np.power(y[oIdx] - y[i],2))
            totV += -0.5 * G * np.sum(self.__mass[i] * self.__mass[oIdx] / r)
        print("总动能:%e\t总势能%e\t总能量:%e\tx方向动量:%e\ty方向动量:%e\t总动量:%e"%
(totK, totV, totV + totK, totPx, totPy, np.sqrt(totPx**2 + totPy**2)))
    def calculate(self, t0 = 50000):
        n = int(t0/self.__dt) + 1
        self.t = [0]
        self.x = np.zeros((n,self.__n))
```

```
        self.y = np.zeros((n,self.__n))
        self.vx = np.zeros((n,self.__n))
        self.vy = np.zeros((n,self.__n))
        self.x[0], self.y[0], self.vx[0], self.vy[0] = self.__x0, self.__y0,
self.__vx0, self.__vy0
        for i in tqdm(range(1, n)):
            self.__Solver(i)
        self.showSystemInfo(0)
        self.showSystemInfo(n - 1)
    def showResult(self, typec = 0):
        totM = np.sum(self.__mass)
        cx = np.mean(self.__mass*self.x, axis = 1)/totM
        cy = np.mean(self.__mass*self.y, axis = 1)/totM
        plt.scatter(cx,cy,label = 'Mass Center', s = 1)
        for i in range(self.__n):
            plt.scatter(self.x[:,i], self.y[:,i], label = self.__name[i], s = 1)
        plt.legend()
        plt.show()
```

- 在`__Solver`中,我实现了对任意多体问题的通用求解器.

  由于求解物体数量不定,我使用`numpy`数组可通过`Bool`数组访问的特性,来实现对物体受力的计算.

  并且,由于没有使用循环,而是使用`numpy`内置的对数组的运算,从而大大提高了运算的速度(约6000代/s).

- 在`showSystemInfo`中,我们实现了计算任意次迭代后的系统状态,包括动能,势能,总能量,动量的分量以及总动量

- 在`calculate`中,我实现了计算的主体,其中,我们没有选择创建空数组并使用`append`方法来增加数据, 而是事先通过`t0/dt`计算出需要计算的代数,预先声明数组大小,这是因为, `numpy`数组的`append`方法时间效率很低. 因此,我们每次求解时均向`Solver`传递当前迭代的代数,由`Solver`将计算结果直接存入数组对应位置.并同时使用`tqdm`实现进度条功能.

- 在`setTotalPToZero`中,我计算了系统的总动量,并通过改变指定星体的速度,来将总动量设为0,以保证质心不动.

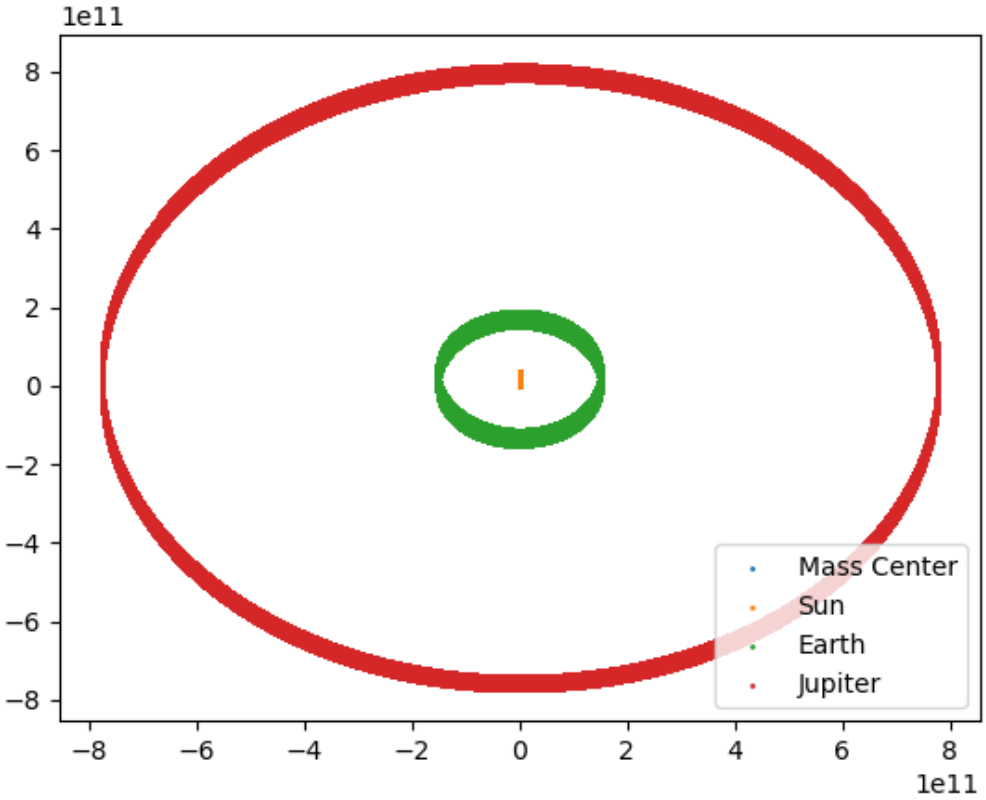
- 在`showResult`中,我实现了对计算结果的展示,在`__init__`中,我们传入了`name`参数,由此我们可以实现对任意多体的轨迹的标识.

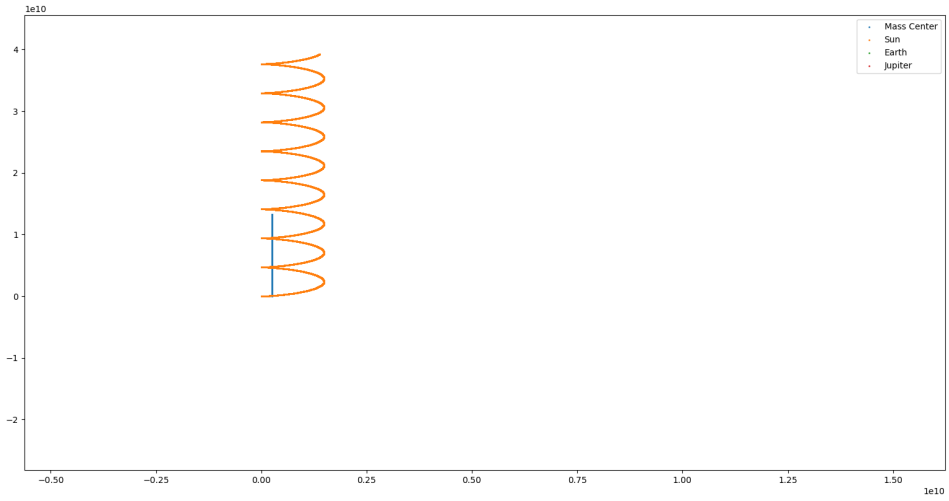
计算结果与讨论

首先我们计算太阳初始速度为0时的情况.

```
if __name__ == '__main__':
    s = multiBodiesSolver(6000,[Ms,Me,Mj],[(0,0),(Lse,0),(Lsj,0)],[(0,0),
(0,30000),(0,13070)],['Sun','Earth','Jupiter'])
    s.setTotalPToZero(0)
    s.calculate(100*365*24*3600)
```

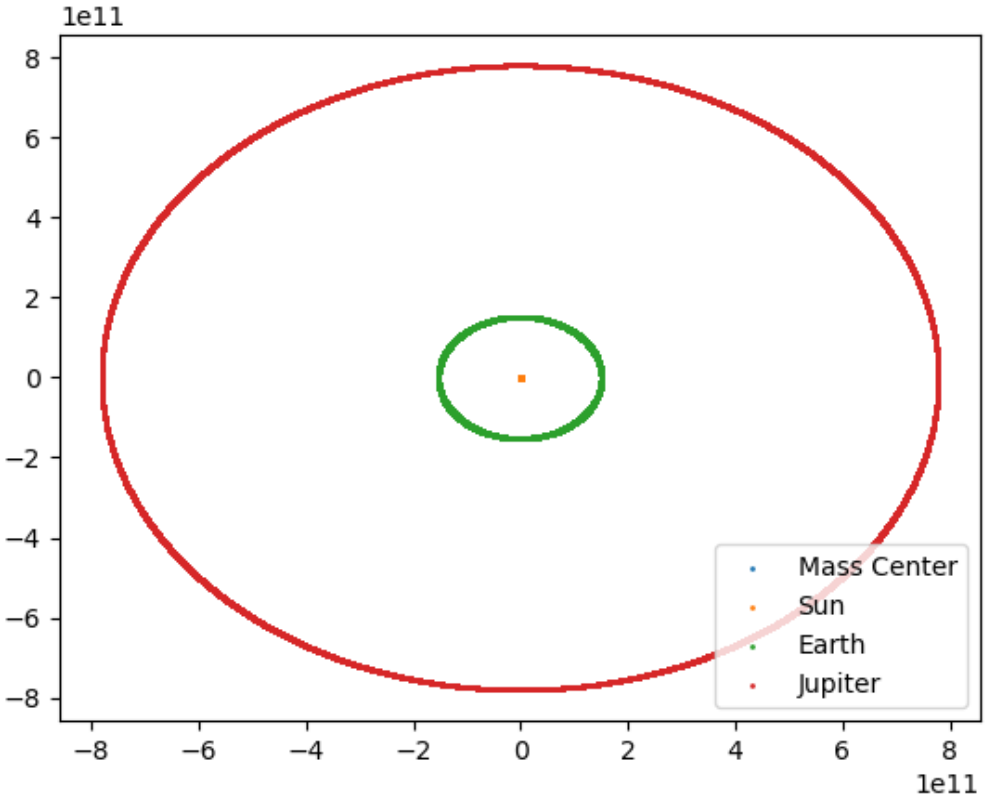
我们通过改变是否调用`setTotalPToZero`方法来比较是否将总动量设为0的差别.

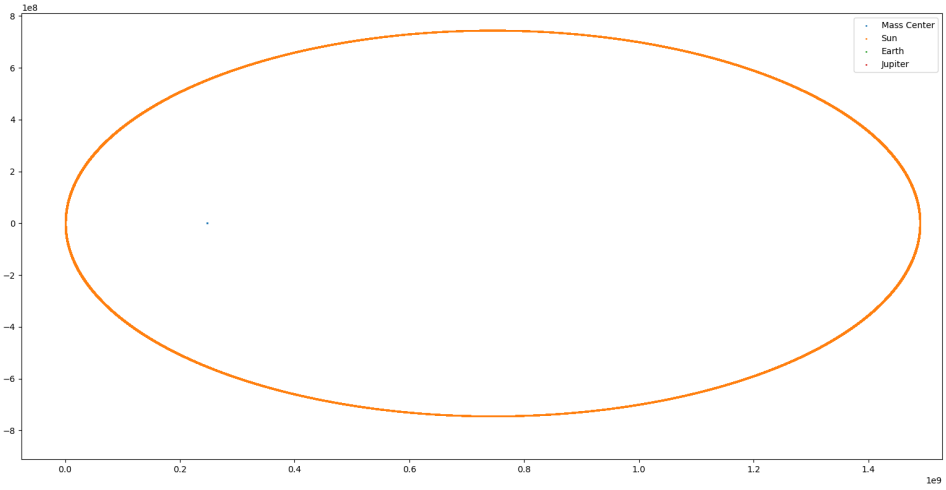




上两图分别是不调整系统总动量为0时不同尺度下的轨迹图,可以发现,当总动量不为0时,系统质心发生较大程度位移.并使得木星和地球轨道表现出较大的进动现象,太阳的运动轨迹也表现出奇异的特征.

而当我们将系统总动量调整为0时,结果如下:





可以发现,此时,质心在长达100年的计算时间内,仍然表现为一个点(尺度也缩小为之前的100倍).同时,轨道的进动现象明显缩小.

```
100%|
总动能:1.649612e+35    总势能-3.292769e+35    总能量:-1.643158e+35    x方向动量:0.000000e+00  y方向动量:2.501150e+31  总动量:2.501150e+31
总动能:1.649612e+35    总势能-3.287439e+35    总能量:-1.637828e+35    x方向动量:0.000000e+00  y方向动量:2.501150e+31  总动量:2.501150e+31
```

同时可以发现,对于此情况下,即使取$\Delta t = 6000$ s,计算结果也比较精确, 对于能量守恒的误差不超过1%

第二题

温度模型

首先考虑到,对于气体的内能,我们有:

$$E = \frac{3}{2}kT \propto T$$

对于辐照强度,我们忽略星际尘埃等对太阳辐照强度的吸收,明显的,辐照强度应与r^2成反比(能量均分在球面上):

$$J(r) = \frac{SJ_0}{r^2}$$

即:

$$\frac{\mathrm{d}\,T(t)}{\mathrm{d}\,t} = \frac{2SJ_0}{3k}\frac{1}{r^2}$$

并由牛顿冷却定律:

$$\frac{\mathrm{d}\,T(t)}{\mathrm{d}\,t} = -\alpha(T(t) - H)$$

我们假设,无论地球与木星和太阳之间的距离如何变化,α和H均保持不变.

所以我们可以推定:

$$\frac{\mathrm{d}\,T(t)}{\mathrm{d}\,t} = -\alpha(T(t) - H) + \frac{2SJ_0}{3k}\frac{1}{r^2}$$

假设温度已达到稳定,则由:

$$\alpha(T - H) = \frac{2SJ_0}{3k}\frac{1}{r^2}$$

即:

$$T = \frac{2SJ_0}{3k\alpha}\frac{1}{r^2} + H$$

至此,我们建立了一个温度模型.

数值设置

对于我们模型中的常数,H为"真空"的温度(此处的真空指宇宙空间, 此时我们不假设宇宙空间是完全的真空了),我们假设$H = 2.7\mathrm{K}$,即为背景辐射的温度.

对于$\frac{2SJ_0}{3k\alpha}\frac{1}{r^2}$项,我么不妨将$\frac{2SJ_0}{3k\alpha}$看作一个常数,假设地球上的平均气温为$298.15\ \mathrm{K}$,则结合日地平均距离,有:

$$\frac{2SJ_0}{3k\alpha} = 6.612 \times 10^{24}$$

计算类的实现

我们重新建立一个继承自第一题的`multiBodiesSolver`的子类`multiBodiesSolverWithHeat`,来计算指定星体受到的辐照强度并由此计算温度.

```
from Q1 import multiBodiesSolver
import numpy as np
from matplotlib import pyplot as plt
```

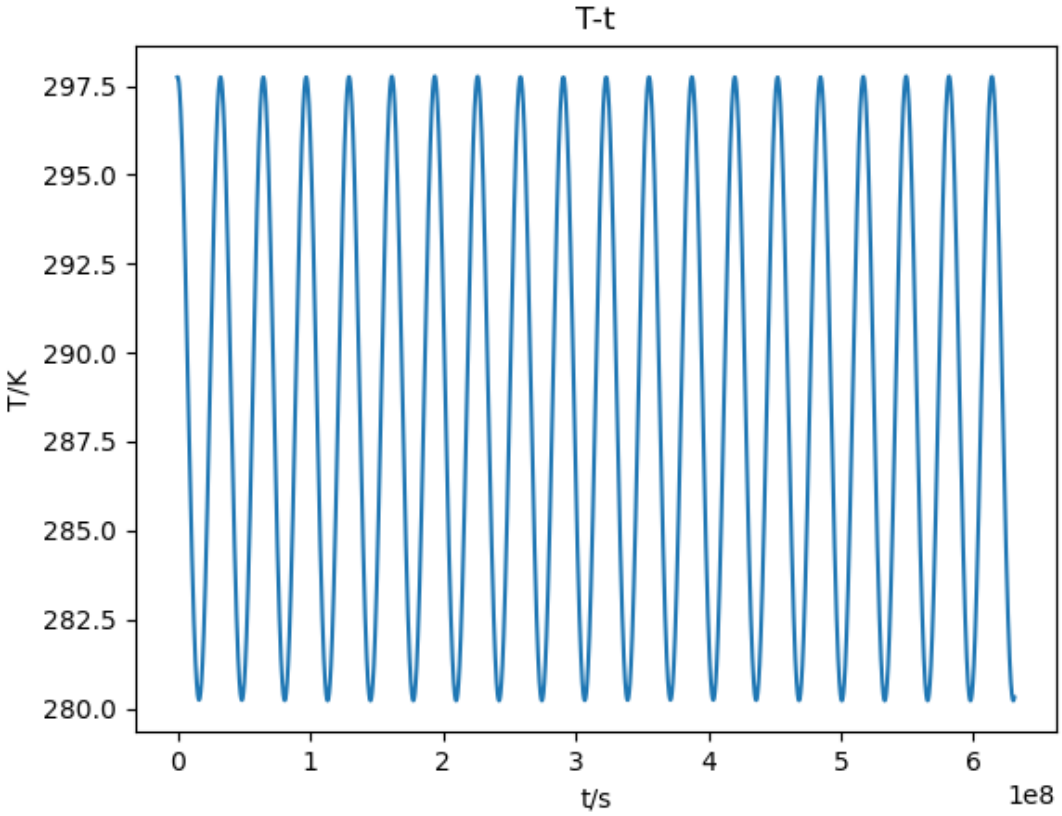
```
Ms = 1.9891*10**30
Mj = 1.90*10**27
Me = 5.95*10**24
Lsj = 7.7833*10**11
Lse = 1.496*10**11

c1 = 6.612*10**24
h = 2.3

class multiBodiesSolverWithHeat(multiBodiesSolver):
    def calcTemp(self,sunIdx, earthIdx):
        T = h
        for i in sunIdx:
            r2 = np.power(self.x[:,i] - self.x[:,earthIdx], 2) +
np.power(self.y[:,i] - self.y[:,earthIdx], 2)
            T += c1/r2
        plt.plot(self.t,T,label = 'Temperature')
        plt.xlabel('t/s')
        plt.ylabel('T/K')
        plt.title('10x$M_J$T-t')
        plt.show()
```

由此,我可以计算各情况下地球温度的变化情况:

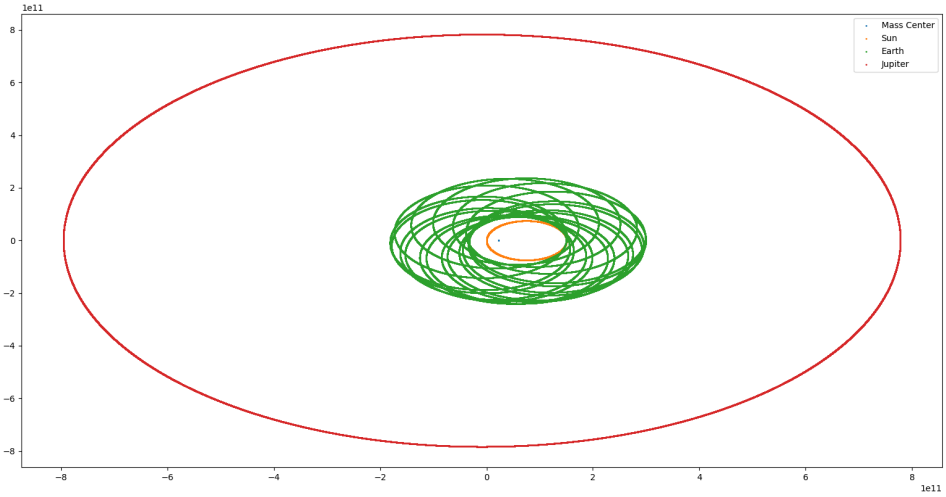
```
s = multiBodiesSolverWithHeat(6000,[Ms,Me,Mj],[(0,0),(Lse,0),(Lsj,0)],[(0,0),
(0,30000),(0,13070)],['Sun','Earth','Jupiter'])
s.setTotalPToZero(0)
s.calculate(100*365*24*3600)
s.calcTemp(0,1)
```

计算结果与讨论

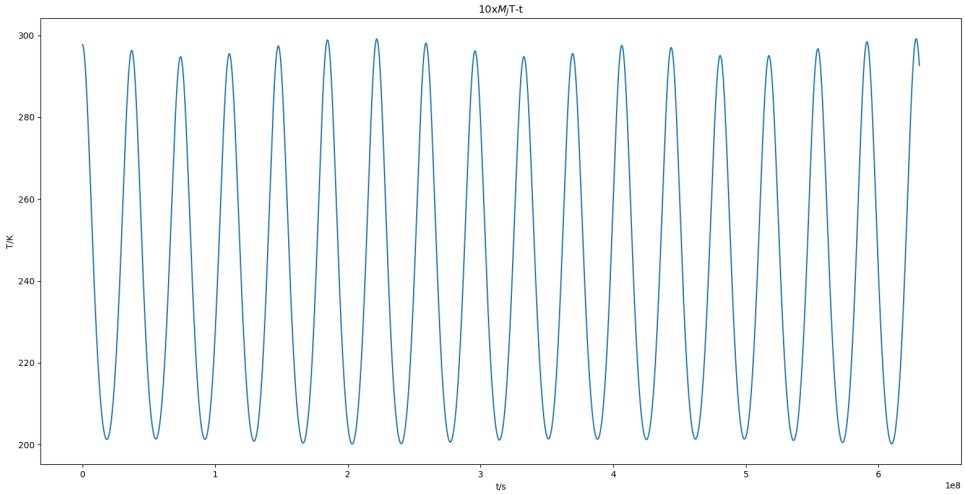
对于普通情况,结果如图,符合生活中的实际情况,周期性变化也符合(1年一个周期),温差不超过20度



当我们将木星的质量增大到100倍时,此时木星的质量接近太阳的质量,观察轨道可以发现,此时地球轨道较为紊乱,太阳轨道也被摄动:



而温度分布仍表现出周期性分布,但是温差已经达到100度左右.



如果将木星质量调至500倍,此时太阳轨道被严重摄动,温差进一步上升到250度左右,并且周期性进一步消失