波浪能最大输出功率设计

摘要

波浪能的转化具有极为广阔的应用前景，而提高波浪能装置的能量转换效率对波浪能的大规模使用有着重要的意义。本文针对波浪能转换装置的运动状态和参数设置，通过建立微分方程模型，实现了对于其运动状态的拟合以及功率最优化的求解。

针对问题一，考虑浮子在波浪中只做垂荡运动，对装置整体和振子分别进行竖直方向的动力学分析，结合牛顿运动定律，构建了浮子与振子的**二阶常微分方程模型**。基于阻力系数为常数与变量的情况，分别使用**四阶龙格-库塔法**进行求解，得到浮子和振子的垂荡位移与速度的曲线图，并将所求结果存放在 result1-1.xlsx 和 result1-2.xlsx 中。

针对问题二，在运动达到稳定状态后，选取一定时长，构建平均输出功率的表达式，并以其为目标函数，分别建立单参数和双参数的**单目标优化模型**。对于以阻尼系数为单参数的优化模型，使用**变步长策略**的搜索遍历算法进行求解。对于以比例系数与幂参数为双参数的优化模型，采用变步长搜索算法得到初始解，然后采用模拟退火算法进行解的优化，以提高求解的速度和精度。最终得到最优阻尼系数为 **37409** $\boldsymbol{kg/s}$，且相应的最大输出功率为 **228.799** $\boldsymbol{w}$，第二种情况的最优阻尼系数为 **100000** $\boldsymbol{kg/m}$，**0.413** 相应的最大输出功率为 **229.523** $\boldsymbol{w}$。

针对问题三，基于垂荡运动与纵横运动，对浮子的受力和力矩进行分析，获得了基于位移以及纵摇角的**二阶微分方程模型**。使用**四阶龙格-库塔法**进行求解，分别得到浮子与振子的垂荡位移与速度曲线图、纵摇角位移与角速度的曲线图，并将所求结果存放在 result3.xlsx 中。

针对问题四，同时考虑直线阻尼系数和旋转阻尼系数对输出功率的影响，建立双参数单目标优化模型，以最大平均输出功率为目标函数，将问题三中建立的微分方程组模型作为约束条件。沿用问题二变步长的搜索遍历算法进行初值求解，之后使用模拟退火算法进行求解，得到最优直线阻尼系数与旋转阻尼器系数分别为 **59048** $\boldsymbol{kg/s}$、**100000** $\boldsymbol{kg\cdot m^2/s}$，相应的最大输出功率为 **322.347** $\boldsymbol{w}$。

最后，我们进行了**灵敏度分析**，发现各参数变化对模型的结果影响较小，说明模型的稳定性较高。此外，本文对于解空间较复杂的优化模型，采用了模拟退火算法，有效降低了其时间复杂程度。

关键词：微分方程模型；龙格-库塔法；单目标优化；变步长搜索；灵敏性分析

一、问题重述

1.1 问题背景

在经济和社会的急速发展下，能源的需求和环境污染是人类面临的两大挑战，因此，可再生能源的发展成为了全世界的共同目标。其中，波浪能应用前景十分可观，其为一种的十分重要的海洋可再生能源，分布广泛，储量丰富。为使波浪能够得到规模化利用，波浪能转化效率问题亟待解决。

目前来讲，虽然波浪能的利用装置在研发技术和开发成本方面都不够成熟，难以与传统能源相竞争，但是在能源相对较为短缺的海岛和离岸较远的海洋平台等的能源供给上，波浪能具有其特有的优势与价值。而且现如今能源短缺问题是我国许多海岛开发和国防建设的关键问题,因此,波浪能转换装置的应用对我国某些地区和岛屿的经济开发与发展有着十分重要的意义。

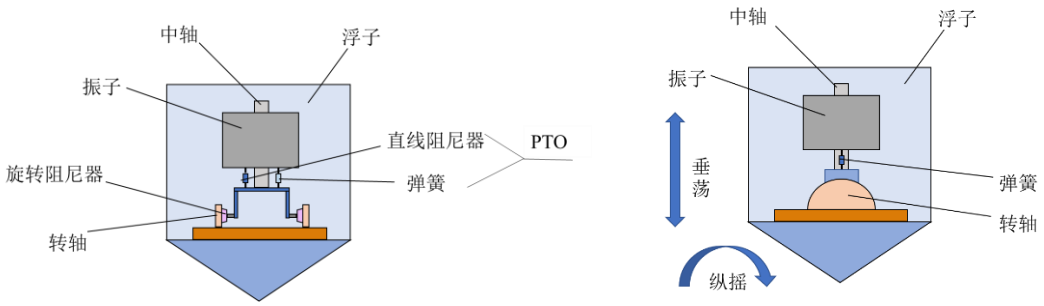

图 1： 模型正视图和侧视图展示

1.2 要解决的问题

问题 1：已知直线阻尼器的阻尼力以及浮子和振子的相对速度之间呈正比，且比例系数恰好为直线阻尼器的阻尼系数。根据浮子与振子竖直方向的运动关系建立数学模型。已知初始时刻的浮子和振子平衡于静水中，根据已有参数值求出题中直线阻尼器的阻尼系数固定，直线阻尼器的阻尼系数与振子的相对速度绝对成正比的两种情况下，在波浪激励力作用下浮子和振子在规定时间规定间隔下的垂荡位移和速度，并将指定结果写入论文中。

问题 2：只考虑浮子在波浪中竖直方向的运动，为使能量输出系统的平均输出功率最大，根据题中两种情况建立数学模型，确定直线阻尼器的最佳阻尼系数，并分别计算最大输出功率以及最佳阻尼系数。

问题 3：在问题 1 的基础上，加入了旋转阻尼器和扭转弹簧，使得浮子会在波浪的作用下进行摇荡运动，即根据浮子与振子在竖直方向和水平方向的运动关系建立数学模型。已知初始时刻的浮子和振子平衡于静水中，根据给定直线阻尼器和旋转阻尼器的阻尼系数，求出在波浪激励力和波浪激励力矩作用下浮子和振子在规定时间规定间隔下的垂荡位移和速度以及纵摇角位移和角速度，并将指定

结果写入论文中。

问题 4：考虑浮子在波浪中水平方向和竖直方向的运动，在阻尼系数均为常量的情况下，建立数学模型确定给定区间内的最佳阻尼系数，根据已有参数值求解最大输出功率以及最佳阻尼系数。

二、问题分析

2.1 问题一的分析：

问题一需要建立浮子与振子的运动模型，对前 40 个波浪周期内浮子与振子的垂荡位移和速度进行求解。由题意，振子会在中轴方向上做往复运动，因此只需考虑竖直方向上的垂荡作用力。考虑以整体和振子为研究对象，建立牛顿第二定理的微分方程组，进行求解。方程需考虑浮子在周期性波浪作用下收到的多种力来建立，且初始值以及关键参数需要根据静态平衡方程来确立。本文中的第一种情况的阻尼系数带入固定值即可，第二种情况需要与振子的相对速度建立关系，带入求解。在对微分方程求解时，考虑使用精度较高的龙格库塔法进行求解数值解，最后等间隔取出所目标时间的垂荡位移和速度，并将求解结果储存到表格内。

2.2 问题二的分析：

问题二对于上述的两种情况进行实际分析，确定直线阻尼器的最优阻尼系数。同样考虑浮子只做垂荡运动，本文以 PTO 系统的平均输出功率最大为目标函数，通过对问题一中的受力分析模型进行改进，作为约束条件，建立单目标优化模型。而阻尼系数的区间在一定范围内取值，考虑使用变步长策略的遍历搜索算法进行最优解的搜索，最终得到最优的阻尼系数，使得系统的平均输出功率值最大，并将求解结果储存到表格内。

2.3 问题三的分析：

本文需要在第一问的基础上添加浮子横向的纵摇运动，即在波浪的作用下，浮子进行摇荡运动。因此需要对问题一的模型进行进一步的修正与改进。考虑浮子 只做垂荡和纵摇运动，自由度为 2，因此建立直线坐标系对其进行受力分析，而纵向垂荡与横向纵摇会相互影响，因此需要将两个方向的运动方程进行耦合，建立完微分受力模型后，继续使用龙格库塔法进行求解，从而较高精度的求解数值解，最后获得两种情况下浮子与振子的垂荡位移与速度和纵摇角位移与角速度，并将求解结果储存到表格内。

2.3 问题四的分析：

问题四是对问题三模型的进一步分析，需要考虑直线阻尼器和旋转阻尼器两者的最优阻尼系数。首先考虑两个阻尼器的阻尼系数均为常量的情况，确定直线阻尼器和旋转阻尼器最优阻尼系数的数学模型。继续沿用第二问思路，建立本文以 PTO 系统的平均输出功率最大为目标函数，通过对问题三中的受力分析模型进行改进，作为约束条件，建立单目标优化模型。而阻尼系数的区间在一定范围内取值，考虑使用变步长策略的遍历搜索算法进行最优解的搜索，最终得到最优的阻尼系数，使得系统的平均输出功率值最大，并将求解结果储存到表格内。

三、模型假设

1. 为了构建更为精确的数学模型，本文根据实际情况做出以下合理的假设或条件约束：
2. 假设海水为理想流体，即海水为不可压缩且无粘、无旋，且海水的各部分都自由地流动。
3. 在进行受力分析时，忽略中轴、底座、隔层以及弹簧和阻尼器的质量和各种摩擦。
4. 假设浮子的圆柱和圆锥壳体以及振子的质量为均匀分布
5. 波浪均为常规波浪，认为其做规律的周期波动，不考虑狂风暴雨等特殊天气的影响。

四、符号说明

符号	说明
m_1	浮子质量
m_2	振子质量
m_3	附加质量
H_1	圆锥部分高度
H_2	圆柱部分高度
H_3	振子高度
R	浮子半径
r	振子半径

注：未申明的变量以其在符号出现处的具体说明为准。

五、模型的建立与求解

5.1 问题一模型的建立与求解

5.1.1 模型的准备

（1）直线坐标系的建立

根据题目要求，考虑浮子在波浪中只做垂荡运动，对其进行受力分析，建立浮子与振子的运动方程模型。

首先建立坐标系，以便于受力讨论。以振子中轴的最下端为坐标原点，振子中轴朝上为 x 轴正方向，建立坐标轴。

（2）对浮子和振子整体初始位置分析

首先，我们对浮子和振子初始平衡于静水中的状态进行分析，以便于确定其初始位置，求解关键参数。

由题目可知，初始时刻浮子和振子平衡于静水中，对于浮子和振子整体受力分析，有

$$F_{浮} = G \tag{1}$$

式中，$F_{浮}$为装置所受浮力，G为装置所受重力，重力又可以表示为：

$$G = (m_1 + m_2)g \tag{2}$$

式中，m_1，m_2分别为浮子和振子的质量，g为重力加速度。

浮力又可以表示为：

$$F_{浮} = \rho g V \tag{3}$$

式中，ρ为水的密度，V为浮子在水下的体积。

联立（2）、（3）式，有

$$(m_1 + m_2)g = \rho g V \tag{4}$$

由题目已知可得

$$\begin{cases} m1 = 4866kg \\ m2 = 2433kg \\ \rho = 1025kg/m^3 \\ g = 9.8m/s^2 \end{cases}$$

求解得

$$V = 7.12m^3$$

浮子圆柱部分体积为：

$$V_1 = \pi * R^2 * H_1 = 9.42m^3$$

式中，R为浮子的半径，H_1为浮子圆柱部分高度。

浮子圆锥部分体积为：

$$V_2 = \frac{1}{3} * \Pi * R^2 * H_2 = 0.83m^3$$

式中，H_2为浮子圆锥部分高度。

由于

$$V < V_1 + V_2$$

故浮子圆柱部分会露出水面，露出水面高度 h 为：

$$h_1 = \frac{V_1 + V_2 - V}{V_1} * H_1 = 1m$$

浮子圆柱部分在水下的高度为

$$h_2 = H_1 - h_1 = 2m$$

（3）对振子初始位置分析

由题目可知，初始时刻振子平衡于静水中，由于振子受力平衡，有

$$G_2 = F_4 \quad (5)$$

其中，G_2为振子的重力，F_4为振子所受弹簧弹力，两者分别为：

重力大小为：

$$G_2 = m_2 g \quad (6)$$

式中，m_2为振子质量，g为重力加速度。

弹簧弹力大小为

$$F_4 = k_1 x_0 \quad (7)$$

式中，k_1为弹簧弹性系数，x_0为弹簧初始时刻压缩量。

联立（6）、（7）式可解得

$$x_0 = 0.298m$$

5.1.2 模型的建立

（1）浮子与振子受力分析

对波浪能转换器整体和振子在线性周期微幅波作用下的受力情况进行分析，根据题意，其会受到以下自身重力以及 4 种。

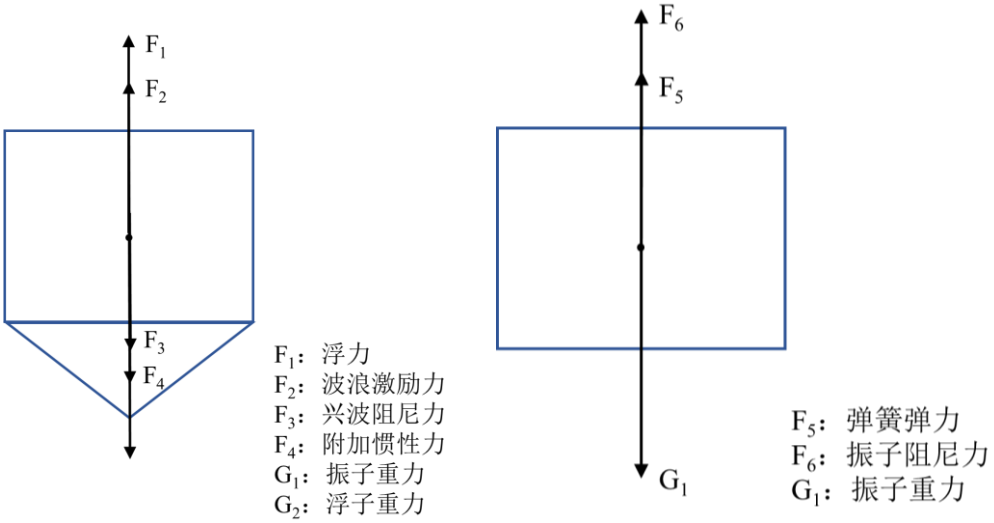

图 2： 对浮子、振子进行受力分析

- **浮子静水恢复力**

而题目中所出现的静水恢复力，根据定义，其可以表示为刚体受到的浮力减去重力，假设其为F_7，即：

$$F_7 = F_1 - (G_1 + G_2) \tag{8}$$

从题中可和参考文献[1]可知，垂荡浮子的净水恢复力 F 可以表示为：

$$F_7 = V_{(x_1)}g \tag{9}$$

式中，$V_{(x_1)}$可以表示为

$$V_{(x_1)} = \begin{cases} \pi R x_1 & x_1 \le 2 \\ 2\pi + (2.8 - x_1)^3 * \pi * 0.5208 & 2 \le x_1 < 2.8 \end{cases}$$

- **浮子兴波阻尼力**

浮体在海水中做垂荡运动时，产生的波浪会对浮体摇荡运动造成阻力，称为兴波阻尼力。兴波阻尼力与浮体垂荡的速度成正比，方向相反，兴波阻尼系数已知，记为q，则有

$$F_3 = q \cdot v_2 \tag{10}$$

式中，q为兴波阻尼系数，v_2为浮子的运动速度。

- **浮子波浪激励力**

浮子和振子在波浪激励力$fcos\omega t$的作用下进行垂荡运动，其中，垂荡激励力振幅f和入射波浪频率ω均已给出，波浪激励力大小和方向只和时间t有关，方向始终为竖直方向。

$$F_2 = fcos\omega t \tag{11}$$

式中，f为垂荡激励力振幅，ω为入射波浪频率。

- **附加惯性力**

推动浮体做摇荡运动的力不仅要推动浮体运动，还要推动浮体周围的流体运动。因此，如果要使浮体在波浪中获得加速度，则需要施加额外的力，即为附加惯性力。

附加惯性力会对应产生一个虚拟质量的，即为附加质量，记为m_3，则附加惯性力为：

$$F_4 = m_3 \frac{d^2 x_1}{dt^2} \tag{12}$$

式中，m_3为附加质量，x_1为浮子的位移。

- **振子弹簧弹力**

振子沿中轴做往复运动，一端始终连接着弹簧，在垂荡运动中，始终与弹簧有作用力，力的大小与浮子和振子的位移差成正比，则振子所受弹力大小为：

$$F_5 = k_1(x_1 - x_2 + x_0) \tag{13}$$

式中，k_1为弹簧弹性系数，x_1为浮子的位移，x_2为振子的位移，x_0为初始时刻弹簧的压缩量。

- **振子阻尼力**

当振子和浮子相对运动时，直线阻尼器会对振子和浮子产生阻尼力[2]，阻尼力与相对速度成正比，记直线阻尼器的阻尼系数为k_2，则有

$$F_6 = k_2 \cdot (v_1 - v_2) \tag{14}$$

式中，v_1为浮子的运动速度，v_2为振子的运动速度。

（2）动力学方程组建立

联立上述公式将各力的具体表达式带入受力分析方程组中，得到浮子与振子的运动模型如下：

- 以浮子和振子整体为对象建立动力学方程：

  浮子和振子受到的合外力包括波浪激励力和兴波阻尼力，以及静水恢复力，对浮子和振子列出牛顿第二定律方程：

$$(m_1 + m_3)\frac{d^2x_1}{dt^2} + m_2\frac{d^2x_2}{dt^2} = f\cos\ \omega t - c\frac{dx_1}{dt} - V_{(x_1)}\rho g \tag{15}$$

- 以振子为对象建立动力学方程：

  对于振子而言，只收到重力，弹簧弹力、阻尼力，其中，重力与弹簧初始时的弹力相抵消，故可以简化情景为：弹簧初始时刻为原长，且振子不受重力，对振子有牛顿第二定律方程：

$$m_2\frac{d^2x_2}{dt^2} = \left[-k_1(x_2 - x_1) - k_2\left(\frac{dx_2}{dt} - \frac{dx_1}{dt}\right)\right] \tag{16}$$

- 初始状态坐标的选取

  浮子和振子沿竖直方向运动，记浮子和振子初始位置位于坐标原点，有

$$\begin{cases} x_1(0) = 0 \\ x_2(0) = 0 \end{cases}$$

  同时，由题目可知，浮子和振子初始时静止于水面，初速度为0，由

$$\begin{cases} \dfrac{dx_1(0)}{dt} = 0 \\ \dfrac{dx_2(0)}{dt} = 0 \end{cases}$$

- 建立装置总的运动方程组如下：

$$\begin{cases}(m_1+m_3)\dfrac{d^2x_1}{dt^2}+m_2\dfrac{d^2x_2}{dt^2}=f\cos\ \omega t-c\dfrac{dx_1}{dt}-V_{(x_1)}\rho g\\ m_2\dfrac{d^2x_2}{dt^2}=\left[-k_1(x_2-x_1)-k_2\left(\dfrac{dx_2}{dt}-\dfrac{dx_1}{dt}\right)\right]\\ V_{(x_1)}=\begin{cases}\pi R x_1 & x_1\le 2\\ 2\pi+(2.8-x_1)^3*\pi*0.5208 & 2\le x_1<2.8\end{cases}\\ x_1(0)=0\\ x_2(0)=0\\ \dfrac{dx_1(0)}{dt}=0\\ \dfrac{dx_2(0)}{dt}=0\end{cases} \tag{17}$$

5.1.3 四阶龙格-库塔方法求解微分方程

上述建立的模型为二元二阶微分方程组，可以使用四阶龙格-库塔方法进行求解。龙格-库塔法是用于求解非线性常微分方程的一类隐式或显式迭代法，以泰勒公式和使用斜率近似代替微分，具有较高的求解精度。

本文使用经典的四阶龙格-库塔法，利用软件编程进行仿真运用，省去了求解微分方程的复杂过程。

由于古典的龙格-库塔法常用来解决一阶常微分方程，本文引入新的变量将二阶微分方程组转化为一阶微分方程组来求解，引入新的变量u_1,u_2,w_1,w_2，并令：

$$\begin{cases}u_1=x_1\\ u_2=\dfrac{dx_1}{dt}\\ w_1=x_2\\ w_2=\dfrac{dx_2}{dt}\end{cases} \tag{18}$$

因此可以得到降阶后的微分方程组：

$$\begin{cases}f_1(t)=u_2=\dfrac{dx_1}{dt}\\ f_2(t)=\dfrac{dx_2}{dt}=\dfrac{d^2x_2}{dt^2}=\left[-k_1(x_2-x_1)-k_2\left(\dfrac{dx_2}{dt}-\dfrac{dx_1}{dt}\right)\right]\dfrac{1}{m_2}\\ f_3(t)=w_2=\dfrac{dx_2}{dt}\\ f_4(t)=\dfrac{dw_1}{dt}=\dfrac{d^2x_1}{dt^2}=\left[f cos\omega t-c\dfrac{dx_1}{dt}-S_{(x_1}\rho x_1 g+k_1(x_2-x_1)+k_2\left(\dfrac{dx_2}{dt}-\dfrac{dx_2)}{dt}\right)\right]\left(\dfrac{1}{m_1+m_3}\right)\end{cases} \tag{19}$$

将附件 3，附件 4 中各参数带入，对问题一的两种情况分别进行求解。

（1）直线阻尼器的阻尼系数为 10000

此时直接将其系数作为已知参数，带入微分方程组中求解即可：

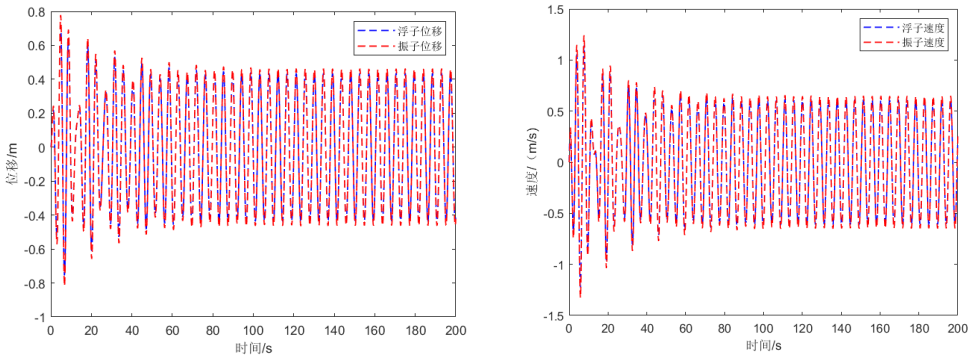

图 3：第一种情况浮子与振子的位移与速度变化曲线

选取浮子和振子关键点的位移与速度，填入下表中:

表 1：浮子与振子某些时间的位移与速度

时间（s）	位移（m）		速度（m/s）	
	浮子	振子	浮子	振子
10	-0.191	-0.212	-0.641	-0.694
20	-0.591	-0.634	-0.241	-0.273
40	0.285	0.296	0.313	0.333
60	-0.315	-0.331	-0.479	-0.516
100	-0.084	-0.084	-0.604	-0.643

其余详细结果见附件 result1-1.xlsx 。

（2）直线阻尼器的阻尼系数与浮子和振子的相对速度的绝对值的幂成正比，其中比例系数取 10000，幂指数取 0.5

此时需要建立阻尼系数与相对速度的关系式：即

$$k_2 \propto \left|\frac{dx_2}{dt} - \frac{dx_1}{dt}\right|^{0.5} \tag{20}$$

将其带入求解可得:

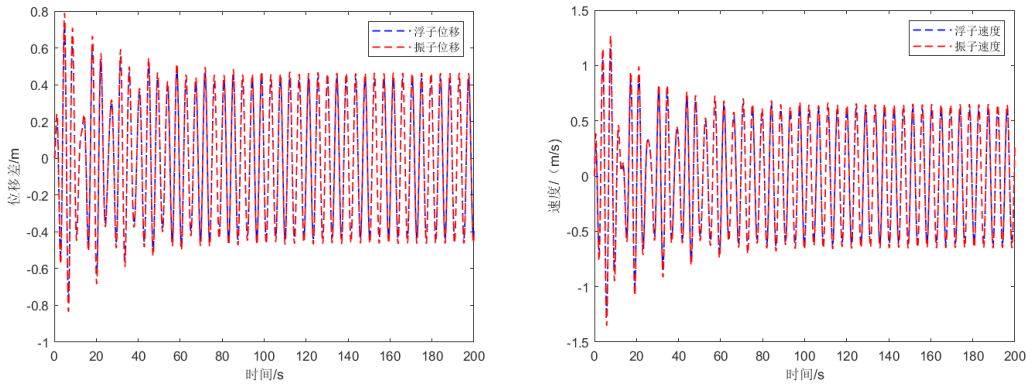

图 4：第二种情况浮子与振子的位移与速度变化曲线

选取浮子和振子关键点的位移与速度，填入下表中:

表 2：浮子与振子某些时间的位移与速度

时间（s）	位移（m）		速度（m/s）	
	浮子	振子	浮子	振子
10	-0.206	-0.235	-0.653	-0.700
20	-0.611	-0.661	-0.255	-0.277
40	0.269	0.280	0.295	0.313
60	-0.327	-0.350	-0.492	-0.526
100	-0.088	-0.093	-0.610	-0.650

5.1.4 结果分析

对浮子与振子的位移进行求差操作

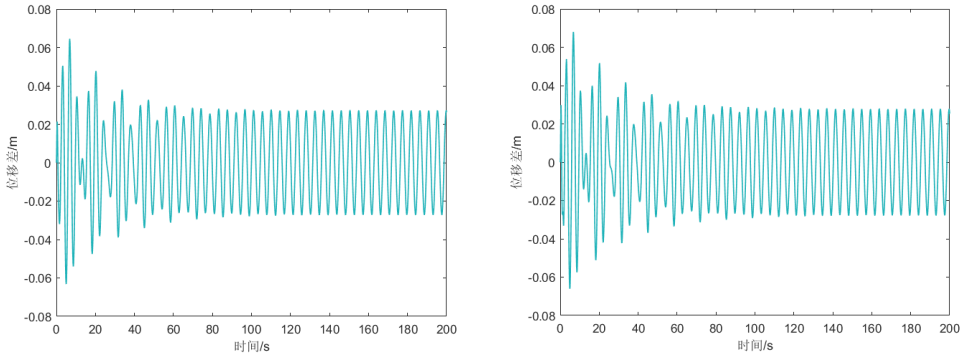

图 5：两情况浮子与振子位移差

经过画图分析可得，两种情况下，浮子与振子的最大位移差都在 0.1m 以内，大于弹簧的最大压缩量，结果合理可信。

5.2 问题二模型的建立与求解

5.2.1 单目标规划模型的建立

题目中要求在一定区间内改变直线阻尼器的阻尼系数，从而使得振子与浮子即 PTO 系统的平均输出功率最大，直接确定一定的步长，对区间进行遍历操作会造成一定的偏差，影响求解的精确性。因此，建立如下基于变步长搜索算法的单目标优化模型。

1.单参数单目标优化模型

- **目标函数：**

根据题目要求，需要使得 PTO 系统的平均输出功率最大，即max（P），其中 P 为系统垂荡稳定后的平均功率，记瞬时平均功率为P_1

$$P_1(t) = k_2 * (\frac{dx_2}{dt} - \frac{dx_1}{dt})^2 \quad (21)$$

由第一问结果分析可知，装置波动 200s 后已基本达到垂荡稳定状态，故可以认为400s~500s这段时间的平均输出功率可以作为垂荡稳定后的平均输出功率。

$$P = \frac{\int_{t1}^{t2} P_1(t)dt}{t2 - t1} \quad (22)$$

式中，取时间上限为t_2为 500s，时间下限为t_1为 400s。

- **约束条件：**

a）弹簧的最大压缩率不能大于其原长，考虑到坐标原点为振子下端，而弹簧在初始状态时已经处于压缩状态，且$x_0 = 0.298m$,而弹簧原长为 0.5m，即

$$x_1 - x_2 < 0.202m \quad (23)$$

b)由题目可知，阻尼系数只能在[0,100000]内取值，即

$$0 \le k_2 \le 100000\ N \cdot s/m \quad (24)$$

c)包含问题一中动力学分析方程

- **模型建立：**

综上所述，建立单目标优化模型

$$max(P)$$

$$\begin{cases} (m_1 + m_3)\frac{d^2x_1}{dt^2} + m_2\frac{d^2x_2}{dt^2} = f\cos\ \omega t - c\frac{dx_1}{dt} - V_{(x_1)}\rho g \\ m_2\frac{d^2x_2}{dt^2} = \left[-k_1(x_2 - x_1) - k_2\left(\frac{dx_2}{dt} - \frac{dx_1}{dt}\right)\right] \\ V_{(x_1)} = \begin{cases} \pi R x_1 & x_1 \le 2 \\ 2\pi + (2.8 - x_1)^3 * \pi * 0.5208 & 2 \le x_1 < 2.8 \end{cases} \\ P = \frac{\int_{t1}^{t2} P_1(t)dt}{t2 - t1} \\ P_1(t) = k_2 * (\frac{dx_2}{dt} - \frac{dx_1}{dt})^2 \\ x_1(0) = 0 \\ x_2(0) = 0 \\ \frac{dx_1(0)}{dt} = 0 \\ \frac{dx_2(0)}{dt} = 0 \\ x_1 - x_2 < 0.202m \\ 0 \le k_2 \le 100000 \end{cases}$$

2 双参数单目标优化模型

- **目标函数**

根据题目要求，同样需要使得平均功率最大，但是需要对平均功率模型进行

修正，修正模型如下：

$$P_1(t) = k_2 * |\frac{dx_2}{dt} - \frac{dx_1}{dt}|^{0.5} * \left(\frac{dx_2}{dt} - \frac{dx_1}{dt}\right) \quad (25)$$

由上个优化模型可知，装置波动 200s 后已基本达到垂荡稳定状态，同样的

$$P = \frac{\int_{t1}^{t2} P_1(t)dt}{t2 - t1} \quad (26)$$

式中，取时间上限为t_2为 500s，时间下限为t_1为 400s。

- **约束条件：**

由题目可知，本文后半部分为双参数优化模型，

其中比例系数在区间 [0,100000] 内取值，幂指数在区间 [0,1] 内取值。

则参数取值范围为：

$$\begin{cases} 0 \le k_2' \le 100000 \\ 0 \le k_3 \le 1 \end{cases} \quad (27)$$

- **模型建立：**

综上所述，建立单目标优化模型

$$max(P)$$

$$\begin{cases} (m_1 + m_3)\frac{d^2x_1}{dt^2} + m_2\frac{d^2x_2}{dt^2} = f\cos\ \omega t - c\frac{dx_1}{dt} - V_{(x_1)}\rho g \\ m_2\frac{d^2x_2}{dt^2} = \left[-k_1(x_2 - x_1) - k_2\left(\frac{dx_2}{dt} - \frac{dx_1}{dt}\right)\right] \\ V_{(x_1)} = \begin{cases} \pi R x_1 & x_1 \le 2 \\ 2\pi + (2.8 - x_1)^3 * \pi * 0.5208 & 2 \le x_1 < 2.8 \end{cases} \\ P = \frac{\int_{t1}^{t2} P_1(t)dt}{t2 - t1} \\ P_1(t) = k_2 * |x_2 - x_1|^{k3} * (x_2 - x_1)^2 \\ x_1(0) = 0 \\ x_2(0) = 0 \\ \frac{dx_1(0)}{dt} = 0 \\ \frac{dx_2(0)}{dt} = 0 \\ x_1 - x_2 < 0.202m \\ 0 \le k_2 \le 100000 \end{cases}$$

xZ

5.2.2 模型的求解

1）对单参数优化方程求解—变步长搜索算法

根据题目，阻尼系数在所给定区间中取值，理论上只需要对其继续遍历，计算所得到的功率 P 即可比较得出最优的平均输出功率，但步长的确定会对结果的求解产生很大的影响，如果步长确定过大，可能会造成解的局部性，或者降低求解的精确度；如果步长过小，会极大的增加求解的运算时间，造成求解效率的减小，因此我们考虑采用变步长遍历算法进行求解。

变步长搜索算法根据求解的区间不同而进行步长的变化与迭代，在较宽的区间上设置较大的迭代步长，当最优解对于的自变量区间逐渐缩小时，也逐渐缩小步长，实现精度的提高。

其算法基本流程步骤如下：

Step1：编写微分方程组，带入附件 3、附件 4 的初始数值数据做初始化处理；

Step2：将直线阻尼器阻尼系数的初始值设置为 0，终止值设置为为 100000，步长设置为 100，对其进行遍历寻优迭代。

Step3：对每一步遍历所得到的阻尼系数进行约束条件的带入，如果不满足约束条件，则舍弃该值，如果满足，则记录该值与其对应的功率值

Step4：将符合条件的阻尼系数所对应的功率值进行比较，记录一个较小的最优值对应区间。

Step5：调节步长为 1，在上述过程所得到的最优区间中继续进行遍历操作，得到更为精确的阻尼系数取值，获得更优的结果。

Step6：输出最优的功率值与其对应的阻尼系数。

经过编程计算，阻尼系数与平均输出功率的对应关系如下图所示

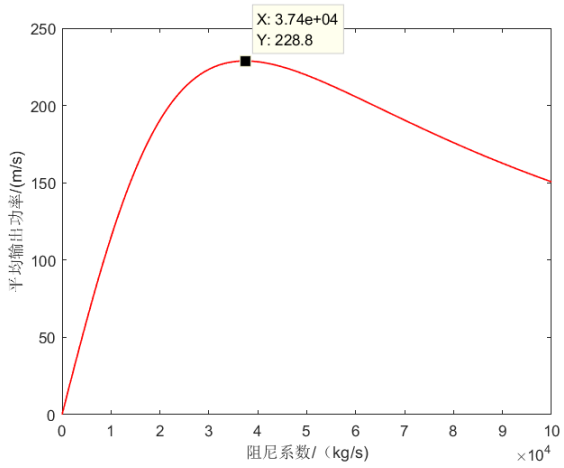

图 6：第一种情况阻尼系数与平均输出功率关系图

通过编程可得，当阻尼系数为 37409 时，PTO 系统的平均输出功率最大，达到 228.799948。

2）对双参数优化方程求解—变步长搜索算法与模拟退火算法

- **变步长搜索算法**

当阻尼系数与浮子和振子的相对速度的绝对值的幂成正比时，我们继续考虑使用变步长搜索算法。此时 Step2 略有变化，需要进行双重循环遍历，即将比例系数的取值变化量设置为 0：100：100000，幂指数在取值变化量设置为 0：0.01：1，对其进行遍历寻优。

求解结果如图所示：

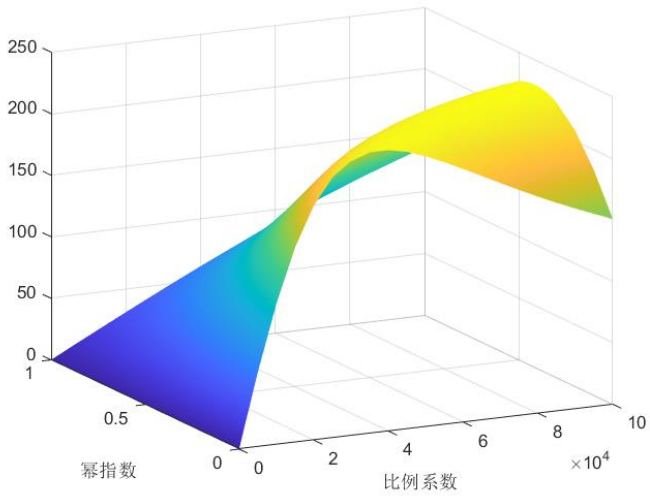

图 7：阻尼参数与平均功率关系图

发现当阻尼系数为 100000 时，幂指数为 0.413 时，功率达到最大值 229.5232。对其结果进行分析，发现当阻尼系数在 95000 到 10000 的区间内，幂指数在 0.4 到 0.5 的区间内变化时，其最优功率基本不变，因此猜想其附件的解平面较为平稳，使用原变步长搜索算法难以精确求解，且求解的效率较低。因此考虑使用现代智能优化算法进行求解，在这里使用模拟退火算法。

- **模拟退火算法**

模拟退火算法是基于迭代求解策略的一种随机寻优算法，其出发点是基于物理中固体物质的退火过程与一般组合优化问题之间的相似性。该算法的优点为求解速度快，可达到全局最优解。

与爬山算法不同的是为了能使算法求解结果尽量靠近全局最优解，模拟退火算法使用蒙特卡洛判断准则来判断接收新解的可能，在本题中，E_t代表当前解为P_t时对应的系统能量。

$$P=\begin{cases}1, & E_{t+1}<E_t\\ e^{\frac{-(E_{t+1}-E_t)}{kT}}, & E_{t+1}\geq E_t\end{cases} \tag{28}$$

当前迭代产生的解比目前最优解更优时，其中T为当前温度，ΔT为当前解所对应的目标函数值与最优解所对应的目标函数值的差值。可以看出，当初始温度足够高，降温足够慢时，每一温度下抽样足够长、最终温度趋近于 0 时，最终降低概率 1 收敛到全局最优解。

其算法的步骤具体为：

Step1：由定步长搜索算法产生初始解。

Step2：对初始解对应的阻尼参数进行随机扰动，计算其周围阻尼参数的对应平均功率，获得新解的产生。

Step3：假设新解与旧解的代价差函数为$\Delta P = P_1 - P_2$，如果$\Delta T < 0$，则接受新的分组；否则，以概率$exp(-\Delta T/T)$接受新的分组，即用计算机产生一个[0,1]区间上均匀分布的随机数 rand，若rand <= $exp(-\Delta T/T)$则接受。

Step4：利用选定的降温系数 α 进行降温，取新的温度 T 为 AT(这里 T 为上一步迭代的温度)，这里选 α=0.999。

Step5：结束条件。用选定的终止温度$e = 10^{-30}$，判断退火过程是否结束。若 T<e，则算发结束，输出当前状态。

算法流程图如下：

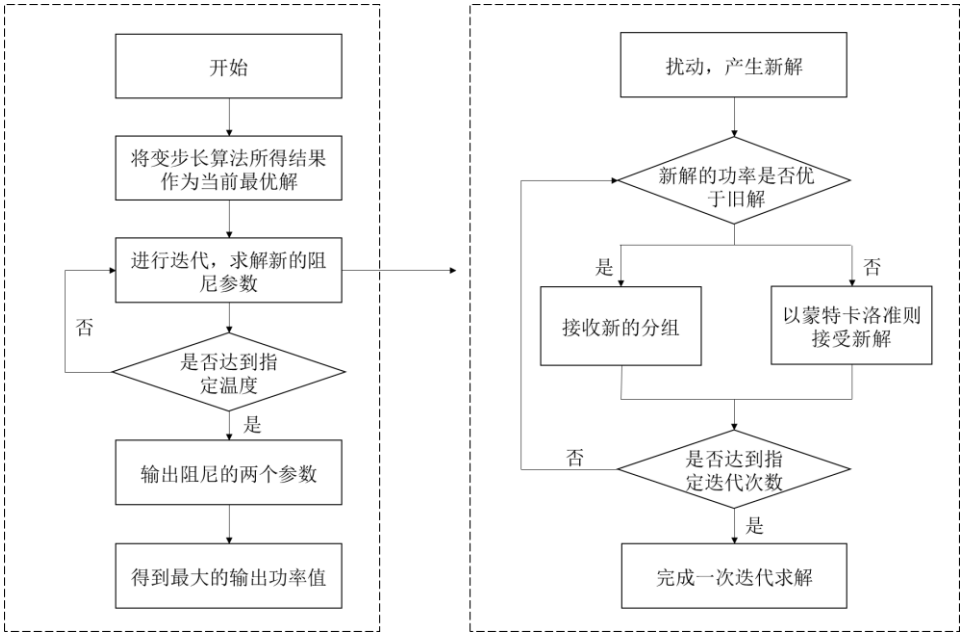

图 8：模拟退火算法流程图

经过编程运行，求解得到最大输出功率为 229.523254w，对应的最优阻尼比例系数为 100000，幂指数为 0.41295。

将其结果与变步长搜索算法进行比较，发现模拟退火算法具有更高的精度。

5.3 问题三模型的建立与求解

5.3.1 模型的准备

此问中，需要考虑浮子的纵摇运动，即在海浪的作用下，其不仅会随着海浪上下起伏，还会通过对浮子的中轴施加作用力矩，使其发生左右摇摆运动，使其运动状况变得更为复杂。一般来说，纵摇和浮子的沉浮相关，因此需要对浮子的质心进行确定，以便于分析其运动状态。

（1）浮子转动惯量的计算

由于浮子是旋转体，故其质心在其轴线上，可以分开求得圆锥部分和圆柱部分的质心位置已经质量，再进行质心公式计算出浮子的质心位置。

记浮子质量为m_1，圆锥部分质量m_4，圆柱部分质量m_5，由于浮子质量分布均匀。故质量所占比重与其表面积大小成正比，记S_{11}为圆锥部分表面积，S_{22}为圆柱部分表面积，其值均可由已知条件求出。

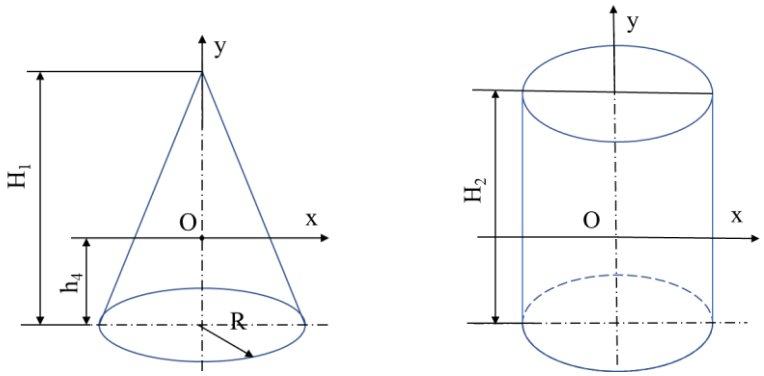

图 9：圆锥和圆柱部分示意图

记浮子质心在圆柱圆锥交界面上方距离为h，根据质心公式有

$$h = h_4 \cdot m_4 - h_5 \cdot m_5 \tag{29}$$

根据平衡轴定理可得，浮子的转动惯量为：

$$J_4 = J_1 + J_2 + m_4 * h_4^2 + m_5 * h_5^2 \tag{30}$$

其中J_1为圆锥部分的转动惯量，J_2为圆柱部分的转动惯量。

（2）振子转动惯量计算

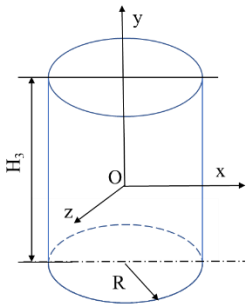

图 3：**振子部分示意图**

当振子以过其质心的直线为回转轴时，其转动惯量类似于圆柱侧面部分转动，为

$$J_5 = \frac{m_2}{2}\left(r^2 + \frac{1}{6}H_3^2\right) \tag{31}$$

式中，m_2为振子质量，r 为振子半径，H_3为振子高度。

实际上，振子绕着底部转轴作纵摇运动，由于忽略了忽略中轴底座和转轴的高度，所以其轴位于隔层平面内，平行于转轴架。振子质心到隔层平面圆心距离为

$$l = (x_2 - x_1 + x_{00}) + \frac{H_3}{2} \tag{32}$$

式中，l为振子质心到隔层平面圆心的距离，x_2为振子垂荡位移，x_1为浮子垂荡

位移，x_{00}为弹簧初始时刻长度，根据平行轴定理，可得振子的转动惯量为

$$J_6 = J_5 + m_2 l \tag{33}$$

式中，J_6为振子实际运动时的转动惯量。

5.3.1 模型的建立

从题目可知，浮子和振子不仅会在力的作用下垂摇，也会在力矩的作用下做纵摇运动，纵摇示意图如下：

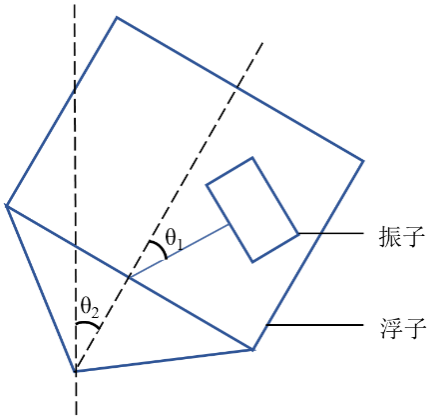

图 12：角度参数分析示意图

图中，将振子中轴线与浮子中轴线的夹角记为θ_1，将铅垂线与浮子的中垂线夹角记为θ_2。

（1）对不同的力矩进行具体分析：

- **浮子静水恢复力矩**

垂荡浮子的净水恢复力 F 可以表示为：

$$M_1 = c_2 \cdot \theta_1 \tag{34}$$

式中，θ_1为浮子的位移，c_2为浮子的静水垂荡恢复力矩系数，其值题目中已给出。

- **浮子波浪激励力矩**

浮子和振子在波浪激励力$Lcos\omega t$的作用下进行纵摇运动，其中，纵摇激励力矩振幅f和入射波浪频率ω题目均已给出，波浪激励力矩大小和方向只和时间t有关，方向始终为水平方向。

$$M_2 = Lcos\omega t \tag{35}$$

式中，L为纵摇激励力矩振幅振幅，ω为入射波浪频率。

- **浮子兴波阻尼力矩**

浮体在海水中做摇荡运动时，产生的波浪会对浮体摇荡运动造成阻力，称为兴波阻尼力矩。兴波阻尼力锯与浮体摇荡的角速度成正比，方向相反，纵摇兴波阻尼系数，记为q_2，则有

$$M_3 = q_2 \cdot w_1 \tag{36}$$

式中，q_2纵摇兴波阻尼系数，w_1为浮子摇荡的角速度。

➢ **扭转弹簧的扭矩**

振子在摇荡运动中，振子连接着转轴架，其一端始终连接着扭转弹簧，在摇荡运动中，转轴架始终与弹簧有力的作用，与浮子和振子的相对角位移成正比，则振子所受扭转弹簧的扭矩为[4]：

$$M_4 = k_3(\theta_1 - \theta_2) \tag{37}$$

式中，k_3为扭转弹簧的刚度，θ_1为浮子的角位移，θ_1为振子的角位移。

➢ **旋转阻尼器的扭矩**

当振子和浮子相对运动时，直线阻尼器会对振子和浮子产生阻尼力，旋转阻尼器的扭矩与浮子和振子的相对角速度成正比，则旋转阻尼器的扭矩为：

$$M_5 = k_4 \cdot (w_1 - w_2) \tag{38}$$

式中，k_4旋转阻尼器的阻尼系数，w_1为浮子的运动速度，w_2为振子的运动速度。

➢ **弹簧弹力**

当浮体发生纵摇运动时，弹簧会发生倾斜，因此重物向弹簧施加的压力会变小，等价于有一个倾斜向上的力在向上拉弹簧，对这个力进行正交分解后，将竖直方向上的力耦合到垂荡运动的受力方程中进行分析。

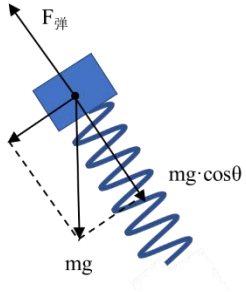

图 13：重力分解示意图

$$F_5{}' = mg - mgcos(\theta_1 + \theta_2) \tag{39}$$

如图，力的方向与竖直方向夹角为$\theta_1 + \theta_2$，方向向左。

➢ **振子阻尼力**

当振子和浮子发生纵摇运动时，阻尼力会倾斜，不再是竖直方向，新的振子阻尼力为：

$$F_6' = -F_6 \tag{35}$$

（2）动力学方程组建立

联立上述公式将各力的具体表达式带入受力分析方程组中，得到浮子与振子的运动模型如下：

➢ 以浮子和振子整体为对象建立动力学方程：

浮子和振子受到的合外力包括波浪激励力和兴波阻尼力，以及静水恢复

力，对浮子和振子列出牛顿第二定律方程：

$$(m_1+m_3)\frac{d^2x_1}{dt^2}+m_2\frac{d^2x_2}{dt^2}=f\cos\omega t-c\frac{dx_1}{dt}-V_{(x_1)}\rho g \tag{36}$$

- 以振子为对象建立动力学方程

  对于振子而言，只收到重力，弹簧弹力、阻尼力，其中，重力与弹簧初始时的弹力相抵消，

$$m_2\frac{d^2x_2}{dt^2}=\mathrm{m_2g}[1-\cos(\theta_1-\theta_2)]-k_1(x_2-x_1)-k_2\left(\frac{dx_2}{dt}-\frac{dx_1}{dt}\right) \tag{37}$$

- 以浮子和振子整体为对象建立纵摇方程：

$$(J_1+J_3)\frac{d^2\theta_1}{dt^2}=Lcos(\omega t)-c\frac{dx_1}{dt}-S_{(x_1}\rho x_1 g+k_1(x_2-x_1)+k_2\left(\frac{dx_2}{dt}-\frac{dx_2}{dt}\right)-h7m_1gsin\theta_1 \tag{38}$$

  式中，h_7为浮子质心距离隔板的距离。

- 以振子为对象建立纵摇方程

$$J_2(x_2)\frac{d^2\theta_2}{dt^2}=-k_3(\theta_2-\theta_1)-k_4\left(\frac{d\theta_2}{dt}-\frac{d\theta_1}{dt}\right)+(\Delta L_1+x_2-x_1+\Delta L_2)\mathrm{sin}(\theta_1+\theta_2) \tag{39}$$

- 初始时刻状态和问题一相同，初始时浮子和质子初始时速度为零，位移为 0
- 建立装置垂摇和纵摇耦合方程组如下：

$$\begin{cases}(m_1+m_3)\dfrac{d^2x_1}{dt^2}+m_2\dfrac{d^2x_2}{dt^2}=f\cos\omega t-c\dfrac{dx_1}{dt}-V_{(x_1)}\rho g\\ m_2\dfrac{d^2x_2}{dt^2}=\mathrm{m_2g}[1-\cos(\theta_1-\theta_2)]-k_1(x_2-x_1)-k_2\left(\dfrac{dx_2}{dt}-\dfrac{dx_1}{dt}\right)\\ (J_1+J_3)\dfrac{d^2\theta_1}{dt^2}=Lcos(\omega t)-c\dfrac{dx_1}{dt}-S_{(x_1}\rho x_1 g+k_1(x_2-x_1)+k_2\left(\dfrac{dx_2}{dt}-\dfrac{dx_2}{dt}\right)-h7m_1gsin\theta_1\\ J_2(x_2)\dfrac{d^2\theta_2}{dt^2}=-k_3(\theta_2-\theta_1)-k_4\left(\dfrac{d\theta_2}{dt}-\dfrac{d\theta_1}{dt}\right)+(\Delta L_1+x_2-x_1+\Delta L_2)\mathrm{sin}(\theta_1+\theta_2)\\ x_1(0)=0\\ x_2(0)=0\\ \dfrac{dx_1(0)}{dt}=0\\ \dfrac{dx_2(0)}{dt}=0\\ \theta_1(0)=0\\ \theta_2(0)=0\\ \dfrac{d\theta_1(0)}{dt}=0\\ \dfrac{d\theta_2(0)}{dt}=0\end{cases}$$

5.3.2 模型的求解

与问题 1 相同，继续采用龙格库塔法进行求解，其为四元二阶微分方程组，需要将其进行变量替换，得到一阶的微分方程组，进行求解。

引入新的变量u_1, u_2, w_1, w_2，并令：

$$\begin{cases} u_1 = x_1，\ u_2 = \dfrac{dx_1}{dt} \\ w_1 = x_2，\ w_2 = \dfrac{dx_2}{dt} \\ p_1 = \theta_1，\ p_2 = \dfrac{d\theta_1}{dt} \\ q_1 = \theta_2，\ q_2 = \dfrac{d\theta_2}{dt} \end{cases} \tag{40}$$

因此可以得到降阶后的微分方程组，

$$\begin{cases} (m_1 + m_3)\dfrac{d^2x_1}{dt^2} = -m_2\dfrac{d^2x_2}{dt^2} = f\cos\omega t - c\dfrac{dx_1}{dt} - V_{(x_1)}\rho g \\ \dfrac{d^2x_2}{dt^2} = \mathrm{g}[1 - \cos(\theta_1 - \theta_2)] - \dfrac{k_1(x_2 - x_1)}{m_2} - \dfrac{k_2}{m_2}\left(\dfrac{dx_2}{dt} - \dfrac{dx_1}{dt}\right) \\ \dfrac{d^2\theta_1}{dt^2} = \dfrac{\left[Lcos(\omega t) - c\dfrac{dx_1}{dt} - S_{(x_1)}\rho x_1 g + k_1(x_2 - x_1) + k_2\left(\dfrac{dx_2}{dt} - \dfrac{dx_2}{dt}\right) - h7m_1 g sin\theta_1\right]}{\left(J_1 + J_3\right)} \\ J_2(x_2)\dfrac{d^2\theta_2}{dt^2} = -k_3(\theta_2 - \theta_1) - k_4\left(\dfrac{d\theta_2}{dt} - \dfrac{d\theta_1}{dt}\right) + (\Delta L_1 + x_2 - x_1 + \Delta L_2)\mathrm{si\,n}(\theta_1 + \theta_2) \end{cases}$$

使用四阶龙格库塔法，设置步长为 0.001，进行求解拟合得到下图:

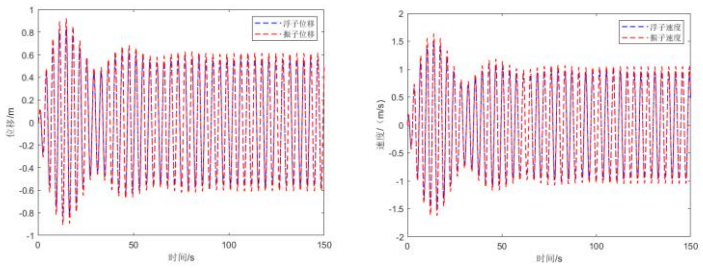

图 14：浮子与振子位移与速度的变化曲线

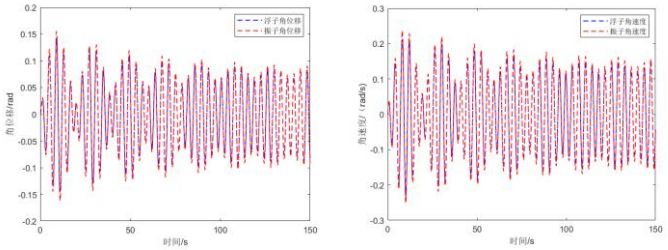

图 15：浮子与振子角速度变化曲线

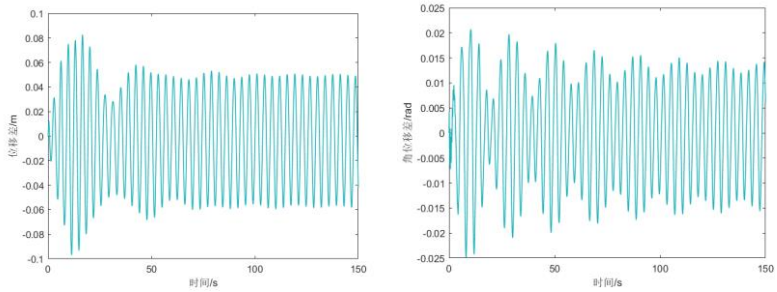

图 16：浮子与振子位移差和角度差变化曲线

将浮子与振子关键时间的位移与速度、角位移、角速度整理如下表：

表 3：浮子与振子某些时间的位移与速度

时间（s）	位移（m）		速度（m/s）	
	浮子	振子	浮子	振子
10	-0.525	-0.595	0.975	1.031
20	-0.706	-0.773	-0.272	-0.321
40	0.369	0.393	0.757	0.846
60	-0.321	-0.341	-0.724	-0.798
100	-0.050	-0.042	-0.948	-1.037

表 4：浮子与振子某些时间的角位移与角速度

时间（s）	角位移（rad）		角速度（rad/s）	
	浮子	振子	浮子	振子
10	0.008	0.008	-0.232	-0.251
20	0.021	0.022	0.004	0.004
40	-0.037	-0.041	-0.024	-0.028
60	0.040	0.043	0.054	0.059
100	0.024	0.027	0.103	0.113

5.3.3 结果分析

浮子和振子的运动仍符合开始波动较大，后期逐渐稳定，做周期运动，与问题一垂摇运动结果相似，非常合理。

由图像可知，浮子和振子的角度差始终在 0.1 弧度内，没有发生较大的角度偏移。

5.4 问题四模型的建立与求解

5.4.1 模型的建立

根据题意，其于第二问相类似，建立单目标优化模型，对平均功率的取值进行优化。

目标函数：根据题目要求，需要使得 PTO 系统的平均输出功率最大，即$max（P）$，其中 P 为直线阻尼和扭转阻尼做功组合而成的，其表达式为：

$$P_1(t) = k_2 * (\frac{dx_2}{dt} - \frac{dx_1}{dt})^2 + k_4\left(\frac{d\theta_2}{dt} - \frac{d\theta_1}{dt}\right)^2 \tag{35}$$

综上，建立双参数单目标优化模型如下：

$$max（P）$$

$$\begin{cases} (m_1 + m_3)\frac{d^2x_1}{dt^2} + m_2\frac{d^2x_2}{dt^2} = f\cos\omega t - c\frac{dx_1}{dt} - V_{(x_1)}\rho g \\ m_2\frac{d^2x_2}{dt^2} = \mathrm{m_2g}[1-\cos(\theta_1-\theta_2)] - k_1(x_2-x_1) - k_2\left(\frac{dx_2}{dt} - \frac{dx_1}{dt}\right) \\ \left(J_1 + J_3\right)\frac{d^2x_1}{dt^2} = Lcos(\omega t) - c\frac{dx_1}{dt} - S_{(x_1}\rho x_1 g + k_1(x_2-x_1) + k_2\left(\frac{dx_2}{dt} - \frac{dx_2}{dt}\right) - h7m_1 g sin\theta \\ J_2(x_2)\frac{d^2\theta_2}{dt^2} = -k_3(\theta_2-\theta_1) - k_4\left(\frac{d\theta_2}{dt} - \frac{d\theta_1}{dt}\right) + (\Delta L_1 + x_2 - x_1 + \Delta L_2)\mathrm{sin}(\theta_1+\theta_2) \\ P_1(t) = k_2 * (\frac{dx_2}{dt} - \frac{dx_1}{dt})^2 + k_4\left(\frac{d\theta_2}{dt} - \frac{d\theta_1}{dt}\right)^2 \\ P = \frac{\int_{t1}^{t2} P_1(t)dt}{t2-t1} \\ x_1(0) = 0 \\ x_2(0) = 0 \\ \frac{dx_1(0)}{dt} = 0 \\ \frac{dx_2(0)}{dt} = 0 \\ 0 < k_2 < 100000 \\ 0 < k_2 < 100000 \end{cases}$$

5.4.2 模型的求解—变步长搜索算法与模拟退火算法

- 变步长搜索算法

本问与问题二相类似，均为求解最大输出功率以及对于的最优阻尼系数。由于直线阻尼器和旋转阻尼器的阻尼系数均在区间 [0,100000] 内取值，如果使用变步长搜索算法，其初始步长的确定十分困难，若步长过短，模型的求解时间过长，若步长过长，则会造成解空间的遍历跨度极大，易造成最优解的丢失。因此在本问中考虑

使用现代智能优化算法进行求解，从而加速收敛求得最优解。

使用变步长搜索算法进行平均输出功率最大值的初始化求解，求得直线阻尼器和旋转阻尼器的阻尼系数分别为 59047 与 100000 时，获得最大的平均功率为 322.337 w。

- 模拟退火算法求解

其算法的步骤具体为：

Step1：由定步长搜索算法产生初始解。

Step2：对初始解对应的阻尼参数进行随机扰动，计算其周围阻尼参数的对应平均功率，获得新解的产生。

Step3：假设新解与旧解的代价差函数为$\Delta P = P_1 - P_2$，如果$\Delta T < 0$，则接受新的分组；否则，以概率$exp(-\Delta T/T)$接受新的分组，即用计算机产生一个[0,1]区间上均匀分布的随机数 rand，若rand <= $exp(-\Delta T/T)$则接受。

Step4：利用选定的降温系数 α 进行降温，取新的温度 T 为 AT(这里 T 为上一步迭代的温度)，这里选 α=0.999。

Step5：结束条件。用选定的终止温度$e = 10^{-30}$，判断退火过程是否结束。若 T<e，则算发结束，输出当前状态。

将其作为模拟退火的初始值，按照问题二模拟退火的算法思路进行编程计算，得到以下结果：

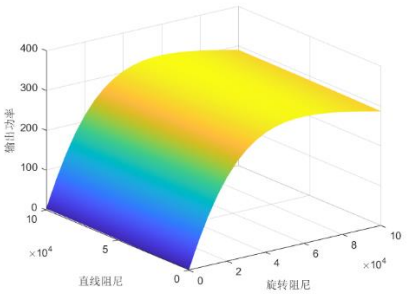

图 17：直线和旋转阻尼器的阻尼系数与功率关系图

得到直线阻尼器和旋转阻尼器的阻尼系数分别为 59048 与 100000 时，获得最大的平均功率为 322.347w。其速度和精度优于变步长搜索算法。

六、灵敏度分析

根据对运动微分方程的分析可知，振子质量、阻尼系数、弹簧刚度、兴波阻尼系数等自变量的变化对因变量直接的影响，因此我们通过对修改这些参数从而对进行灵敏度分析，使其值经过上下 5%的数据波动，利用软件编程 绘制这 4 个参数分别与浮子垂摇稳定的 500s 时的位移与速度关系图，模型灵敏度分析图如下

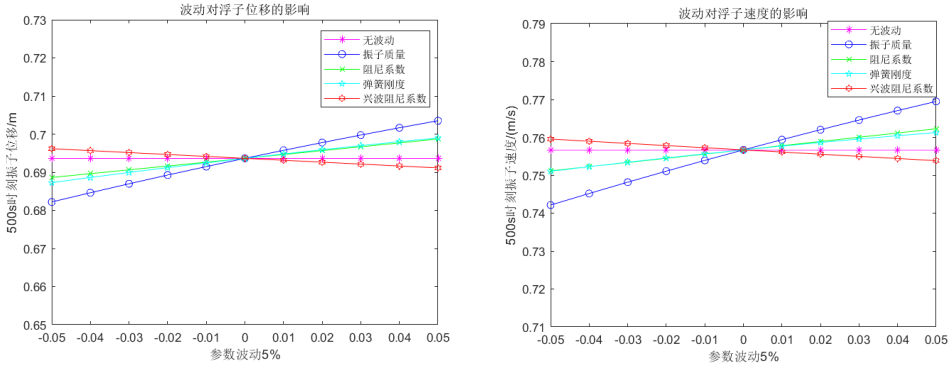

图 4： 灵敏度分析图

如图所示，兴波阻尼系数对稳定后振子的位移和速度影响都较小， 而振子质量、阻尼系数、弹簧刚度都会对稳定后振子的位移和速度影响都非常小。因此，可以推断得出，一般情况下，该模型较为合适，这些参数的变化，不会导致浮子实际运动情况造成较大影响，说明**该模型的稳定性较好**。

七、模型的评价

7.1 模型优点

1. 在进行二阶微分微分方程组的求解时，选用龙格-库塔法，保证了求解精度值。
2. 对于阻尼系数的单参数确定问题，选用了变步长搜索算法进行遍历，使得在保证求解的精度的同时，较快的提高的求解的运算速度。
3. 对于问题二的第二部分、问题四，采用了模拟退火算法，启发式算法的运用使得解空间较为复杂的优化模型更快的收敛，同时也保证了解的较高精确度。

7.2 模型缺点

1. 在建模过程中，对运动方程完全耦合的研究还不够深入，未能分析其对模型逻辑和结果优化的影响。
2. 对振荡的稳定周期没有给出明确的证明过程，存在改进空间。

八、参考文献

[1] 周丙浩. 纵摇浮子式波浪能转换装置研究[D]. 哈尔滨工程大学.
[2] 黄晶华. 振荡浮子液压式波浪能利用装置的研究[D]. 华北电力大学, 2012.
[3] 蔡蓉, 陈佳, 李力锋,等. 控制波浪能发电装置中浮筒对海平面进行跟踪所用的装置:, CN206816434U[P]. 2017.
[4] 周丙浩. 纵摇浮子式波浪能转换装置研究[D]. 哈尔滨工程大学.

附录

问题一的第一问求解

```
#用于第一题第一问求解
import math
import matplotlib.pyplot as plt
import csv

#步长，初始值，及终止值
step = 0.001
start = 0
end = 200

#排开水体积计算
def V(u1):
    if u1 <= 2:
        return math.pi * u1
    else:
        return 2 * math.pi +(1-math.pow((2.8-u1)/0.8,3)) * (0.8 * math.pi/3)

def f1(x,u1,u2,w1,w2):
    return u2

def f2(x,u1,u2,w1,w2):
    return (6250 * math.cos(1.4005 * x) - 656.3616 * u2 -V(u1) * 1025 * 9.8 + 80000 * (w1 - u1) + 10000 * (w2 - u2))*(1/(4866+1335.535))

def f3(x,u1,u2,w1,w2):
    return w2

def f4(x,u1,u2,w1,w2):
    return (1/2433) * (-1*80000 * (w1 - u1) - 10000 * (w2 - u2))

#龙格库塔
def RK4(u1,u2,w1,w2,x):
    for i in range(len(x) - 1):
        k11 = f1(x[i], u1[i], u2[i], w1[i], w2[i])
        k21 = f2(x[i], u1[i], u2[i], w1[i], w2[i])
        L11 = f3(x[i], u1[i], u2[i], w1[i], w2[i])
        L21 = f4(x[i], u1[i], u2[i], w1[i], w2[i])

```

```
        k12 = f1(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
        k22 = f2(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
        L12 = f3(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
        L22 = f4(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)

        k13 = f1(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
        k23 = f2(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
        L13 = f3(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
        L23 = f4(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)

        k14 = f1(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
        k24 = f2(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
        L14 = f3(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
        L24 = f4(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)

        u1[i + 1] = u1[i] + step / 6 * (k11 + 2 * k12 + 2 * k13 + k14)
        u2[i + 1] = u2[i] + step / 6 * (k21 + 2 * k22 + 2 * k23 + k24)
        w1[i + 1] = w1[i] + step / 6 * (L11 + 2 * L12 + 2 * L13 + L14)
        w2[i + 1] = w2[i] + step / 6 * (L21 + 2 * L22 + 2 * L23 + L24)
    return [u1,u2,w1,w2]

#主函数
if __name__ == "__main__":

    x = []
    temp = start
    while temp <= end:
        x.append(temp)
```

```
        temp += step

    u1 = [0 for i in range(len(x))]
    u2 = [0 for i in range(len(x))]
    w1 = [0 for i in range(len(x))]
    w2 = [0 for i in range(len(x))]

    ans = RK4(u1,u2,w1,w2,x)


    with open("D:\Desktop\\T1.1.csv", "w", newline="") as csvfile:
        writer = csv.writer(csvfile)
        writer.writerows(list(map(list, zip(*ans))))

    #作图
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.plot(x, u2, color="red" ,linewidth=1.0, linestyle="-")  #将散点连在一起
    plt.plot(x, w2, color="blue", linewidth=1.0, linestyle="-")
    plt.xlabel('时间/s')
    plt.ylabel('位移/m')
    plt.show()

    #作图
    cha = [(w1[i] - u1[i]) for i in range(len(w1))]
    plt.plot(x, cha, color="red", linewidth=1.0, linestyle="-")  #将散点连在一起
    plt.xlabel('时间/s')
    plt.ylabel('位移/m')
    plt.show()
```

问题一的第二问求解

```
##用于第一题第二问求解
import math
import matplotlib.pyplot as plt
import csv

#步长，初始值，及终止值
step = 0.001
start = 0
end = 200

#排开水体积计算
```

```
12.def V(u1):
13.    if u1 <= 2:
14.        return math.pi * u1
15.    else:
16.        return 2 * math.pi +(1-math.pow((2.8-u1)/0.8,3)) * (0.8 * m
   ath.pi/3)
17.
18.#比例系数
19.def f1(x,u1,u2,w1,w2):
20.    return u2
21.
22.def f2(x,u1,u2,w1,w2):
23.    return (6250 * math.cos(1.4005 * x) - 656.3616 * u2 -V(u1) * 10
   25 * 9.8 + 80000 * (w1 - u1) + 10000 * math.pow(abs(w2 - u2) , 0.5)
    * (w2 - u2))*(1/(4866+1335.535))
24.
25.def f3(x,u1,u2,w1,w2):
26.    return w2
27.
28.def f4(x,u1,u2,w1,w2):
29.    return (1/2433) * (-1*80000 * (w1 - u1) - 10000 * math.pow(abs(
   w2 - u2) , 0.5) * (w2 - u2))
30.
31.#龙格库塔
32.def RK4(u1,u2,w1,w2,x):
33.    for i in range(len(x) - 1):
34.        #第一个点
35.        k11 = f1(x[i], u1[i], u2[i], w1[i], w2[i])
36.        k21 = f2(x[i], u1[i], u2[i], w1[i], w2[i])
37.        L11 = f3(x[i], u1[i], u2[i], w1[i], w2[i])
38.        L21 = f4(x[i], u1[i], u2[i], w1[i], w2[i])
39.
40.        #第二个点
41.        k12 = f1(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
   tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
42.        k22 = f2(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
   tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
43.        L12 = f3(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
   tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
44.        L22 = f4(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
   tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
45.
46.        #第三个点
```

```
47.         k13 = f1(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
    tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
48.         k23 = f2(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
    tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
49.         L13 = f3(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
    tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
50.         L23 = f4(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
    tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
51.
52.         #第三个点
53.         k14 = f1(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23)
54.         k24 = f2(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23)
55.         L14 = f3(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23)
56.         L24 = f4(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23)
57.
58.         #第四个点
59.         u1[i + 1] = u1[i] + step / 6 * (k11 + 2 * k12 + 2 * k13 + k
    14)
60.         u2[i + 1] = u2[i] + step / 6 * (k21 + 2 * k22 + 2 * k23 + k
    24)
61.         w1[i + 1] = w1[i] + step / 6 * (L11 + 2 * L12 + 2 * L13 + L
    14)
62.         w2[i + 1] = w2[i] + step / 6 * (L21 + 2 * L22 + 2 * L23 + L
    24)
63.     return [u1,u2,w1,w2]
64.
65.
66. if __name__ == "__main__":
67.
68.
69.     x = []
70.     temp = start
71.     while temp <= end:
72.         x.append(temp)
73.         temp += step
74.
75.     #初始化矩阵
76.     u1 = [0 for i in range(len(x))]
77.     u2 = [0 for i in range(len(x))]
78.     w1 = [0 for i in range(len(x))]
```

```
    w2 = [0 for i in range(len(x))]

    ans = RK4(u1,u2,w1,w2,x)

    #将结果输出
    with open("D:\Desktop\\T1.2.csv", "w", newline="") as csvfile:
        writer = csv.writer(csvfile)
        writer.writerows(list(map(list, zip(*ans))))

    #作图
    plt.rcParams['font.sans-serif'] = ['SimHei']
    plt.rcParams['axes.unicode_minus'] = False
    plt.plot(x, u1, color="red" ,linewidth=1.0, linestyle="-")  #将散点连在一起
    plt.plot(x, w1, color="blue", linewidth=1.0, linestyle="-")
    plt.xlabel('时间/s')
    plt.ylabel('位移/m')
    plt.show()
```

变步长搜索法求解最佳平均功率

```
#第二题第一问，变步长搜索法求解
import math
import matplotlib.pyplot as plt
import csv
import copy

#步长，初始值，及终止值
step = 0.01
start = 0
end = 450

c1 = 0

#排开水体积计算
def V(tt):
    if tt <= 2:
        return math.pi * tt
    else:
        return 2 * math.pi +(1-math.pow((2.8-tt)/0.8,3)) * (0.8 * math.pi/3)

def f1(x,u1,u2,w1,w2):
    return u2

```

```
def f2(x,u1,u2,w1,w2):
    return (4890 * math.cos(2.2143 * x) - 167.8395 * u2 -V(u1) * 1025 * 9.8 + 80000 * (w1 - u1) + c1 * (w2 - u2))*(1/(4866+1165.992))

def f3(x,u1,u2,w1,w2):
    return w2

def f4(x,u1,u2,w1,w2):
    return (1/2433) * (-1*80000 * (w1 - u1) - c1 * (w2 - u2))

#龙格库塔
def RK4(u1,u2,w1,w2,x):
    for i in range(len(x) - 1):
        k11 = f1(x[i], u1[i], u2[i], w1[i], w2[i])
        k21 = f2(x[i], u1[i], u2[i], w1[i], w2[i])
        L11 = f3(x[i], u1[i], u2[i], w1[i], w2[i])
        L21 = f4(x[i], u1[i], u2[i], w1[i], w2[i])

        k12 = f1(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
        k22 = f2(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
        L12 = f3(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
        L22 = f4(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)

        k13 = f1(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
        k23 = f2(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
        L13 = f3(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
        L23 = f4(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)

        k14 = f1(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
        k24 = f2(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
        L14 = f3(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
        L24 = f4(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
```

```
55.
56.         u1[i + 1] = u1[i] + step / 6 * (k11 + 2 * k12 + 2 * k13 + k
    14)
57.         u2[i + 1] = u2[i] + step / 6 * (k21 + 2 * k22 + 2 * k23 + k
    24)
58.         w1[i + 1] = w1[i] + step / 6 * (L11 + 2 * L12 + 2 * L13 + L
    14)
59.         w2[i + 1] = w2[i] + step / 6 * (L21 + 2 * L22 + 2 * L23 + L
    24)
60.         # print(u1[i])
61.
62.#主函数
63.if __name__ == "__main__":
64.     list1 = []
65.     temp = start
66.     while temp <= end:
67.         list1.append(temp)
68.         temp += step
69.     list2 = [0 for i in range(len(list1))]
70.
71.     x = copy.deepcopy(list1)
72.     aver = []
73.     list = range(37300,37501,1)
74.
75.     for c in list:
76.         u1 = copy.deepcopy(list2)
77.         u2 = copy.deepcopy(list2)
78.         w1 = copy.deepcopy(list2)
79.         w2 = copy.deepcopy(list2)
80.         print(c)
81.         c1 = c
82.         RK4(u1,u2,w1,w2,x)
83.         p = [c * (u2[i] - w2[i]) * (u2[i] - w2[i]) for i in range(4
    0000,45000)]
84.         aver.append([sum(p)/len(p)])
85.
86.     #将结果打印
87.     with open("D:\Desktop\\T2.1.csv", "w", newline="") as csvfile:
88.         writer = csv.writer(csvfile)
89.         writer.writerows(aver)
90.
91.     #作图
92.     plt.rcParams['font.sans-serif'] = ['SimHei']
93.     plt.rcParams['axes.unicode_minus'] = False
```

```
94.     plt.plot(list, aver, color="red" ,linewidth=1.0, linestyle="-")
    # 将散点连在一起
95.     plt.xlabel('时间/s')
96.     plt.ylabel('位移/m')
97.     plt.show()
```

第二题第二问使用模拟退火算法求解

```
1. #第二题第二问，退火算法求解
2. import math
3. import matplotlib.pyplot as plt
4. import csv
5. import copy
6. import random
7.
8. step = 0.01
9. start = 0
10.end = 450
11.
12.c1 = 0
13.d1 = 0
14.
15.T = 1
16.
17.def V(tt):
18.     if tt <= 2:
19.         return math.pi * tt
20.     else:
21.         return 2 * math.pi +(1-math.pow((2.8-tt)/0.8,3)) * (0.8 * m
   ath.pi/3)
22.
23.def f1(x,u1,u2,w1,w2):
24.     return u2
25.
26.def f2(x,u1,u2,w1,w2):
27.     return (4890 * math.cos(2.2143 * x) - 167.8395 * u2 -V(u1) * 10
   25 * 9.8 + 80000 * (w1 - u1) + c1 * math.pow(abs(w2 - u2) , d1) * (
   w2 - u2))*(1/(4866+1165.992))
28.
29.def f3(x,u1,u2,w1,w2):
30.     return w2
31.
32.def f4(x,u1,u2,w1,w2):
```

```
    return (1/2433) * (-1*80000 * (w1 - u1) - c1 * math.pow(abs(w2 - u2) , d1) * (w2 - u2))

def RK4(u1,u2,w1,w2,x):
    for i in range(len(x) - 1):
        k11 = f1(x[i], u1[i], u2[i], w1[i], w2[i])
        k21 = f2(x[i], u1[i], u2[i], w1[i], w2[i])
        L11 = f3(x[i], u1[i], u2[i], w1[i], w2[i])
        L21 = f4(x[i], u1[i], u2[i], w1[i], w2[i])

        k12 = f1(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
        k22 = f2(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
        L12 = f3(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)
        L22 = f4(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + step * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2)

        k13 = f1(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
        k23 = f2(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
        L13 = f3(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)
        L23 = f4(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + step * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2)

        k14 = f1(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
        k24 = f2(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
        L14 = f3(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)
        L24 = f4(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23)

        u1[i + 1] = u1[i] + step / 6 * (k11 + 2 * k12 + 2 * k13 + k14)
        u2[i + 1] = u2[i] + step / 6 * (k21 + 2 * k22 + 2 * k23 + k24)
        w1[i + 1] = w1[i] + step / 6 * (L11 + 2 * L12 + 2 * L13 + L14)
```

```
60.         w2[i + 1] = w2[i] + step / 6 * (L21 + 2 * L22 + 2 * L23 + L
    24)
61.         # print(u1[i])
62.
63. def newans(one,two,st1,st2):
64.     rd = random.randint(1, 4)
65.     one1 = one
66.     two1 = two
67.
68.     if rd == 1:
69.         one1 += st1
70.     if rd == 2:
71.         one1 -= st1
72.     if rd == 3:
73.         two1 += st2
74.     if rd == 4:
75.         two1 -= st2
76.     if one1 <= 100000 and two1 <= 100000 and one1 >= 0 and two1 >=
    0:
77.         return [one1,two1]
78.     else:
79.         return [one,two]
80.
81.
82. if __name__ == "__main__":
83.     list1 = []
84.     temp = start
85.     while temp <= end + 0.00001:
86.         list1.append(temp)
87.         temp += step
88.
89.     u1 = [0 for i in range(len(list1))]
90.     u2 = [0 for i in range(len(list1))]
91.     w1 = [0 for i in range(len(list1))]
92.     w2 = [0 for i in range(len(list1))]
93.
94.     x = copy.deepcopy(list1)
95.     aver = []
96.     list = range(0,100001,5000)
97.
98.     li = []
99.     step2 = 0.1
100.     start2 = 0
101.     end2 = 1
```

```
    temp = start2
    while temp <= end2 +0.0001:
        li.append(temp)
        temp += step2
    c1 = 100000
    d1 = 0.4129

    ii = 0
    for yy in range(10000):

        u1[0] = 0
        u2[0] = 0
        w1[0] = 0
        w2[0] = 0
        [tempc,tempd] = newans(c1,d1,0.5,0.00001)
        tc = c1
        td = d1

        c1 = tempc
        d1 = tempd

        RK4(u1, u2, w1, w2, x)
        p = [c1 * math.pow(abs(w2[i] - u2[i]), d1) * (u2[i] - w2[
   i]) * (u2[i] - w2[i]) for i in range(40000, 45000)]
        tempii = sum(p) / len(p)
        if tempii > ii:
            print(c1,d1,tempii)
            ii = tempii
        else:
            rr = random.randint(0, 100000000)/100000000
            if rr < math.exp(-1 * (ii - tempii) / T):
                print(c1, d1, tempii)
                ii = tempii
            else:
                c1 = tc
                d1 = td

        T = T * 0.5

    # ans = []
    # for c in list:
    #     aa = []
    #     for d in li:
    #         u1[0] = 0
```

```
#         u2[0] = 0
#         w1[0] = 0
#         w2[0] = 0
#         print(c,d)
#         c1 = c
#         d1 = d
#         RK4(u1, u2, w1, w2, x)
#         p = [c1 * math.pow(abs(w2[i] - u2[i]) , d1) * (u2[i] - w2[i]) * (u2[i] - w2[i]) for i in range(40000, 45000)]
#         ii = sum(p) / len(p)
#         aver.append([c,d,ii])
#         aa.append(ii)
#     ans.append(aa)
#
#
# with open("D:\Desktop\\T2.2.csv", "w", newline="") as csvfile:
#     writer = csv.writer(csvfile)
#     writer.writerows(aver)
#
# with open("D:\Desktop\\T2.4.csv", "w", newline="") as csvfile:
#     writer = csv.writer(csvfile)
#     writer.writerows(ans)
```

第三问求解代码

```
#第三题求解
import math
import matplotlib.pyplot as plt
import csv

#步长，初始值，及终止值
step = 0.01
start = 0
end = 500

J1 = 14137.79

#体积
def V(u1):
    if u1 <= 2:
        return math.pi * u1
```

```
    else:
        return 2 * math.pi +(1-math.pow((2.8-u1)/0.8,3)) * (0.8 * math.pi/3)

def J2(x2,x1):
    return 2433 * (0.25 + 0.2019575 + x2 - x1) * (0.25 + 0.2019575 + x2 - x1) + 354.8125

def f1(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return u2

def f2(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return (3640 * math.cos(1.7152 * x) - 683.4558 * u2 -V(u1) * 1025 * 9.8 + 80000 * (w1 - u1) + 10000 * (w2 - u2)-(1-math.cos(p1+q1)) * 9.8 * 2433)*(1/(4866+1028.876))

def f3(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return w2

def f4(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return  (1-math.cos(p1+q1)) * 9.8 - 1 * (1/2433) * (80000 * (w1 - u1) + 10000 * (w2 - u2))

def f5(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return p2

def f6(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return (1690 * math.cos(1.7152 * x) - 654.3383 * p2 - 8890.7 * p1 + 250000 * (q1 - p1) + 1000 * (q2 - p2) - 1.2 * 4866 * 9.8 * math.sin(p1)) / (J1 + 7001.914)

def f7(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return q2

def f8(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return  ((0.25 + 0.2019575 + w1 - u1) * math.sin(p1 + q1) * 9.8 * 2433- 250000 * (q1 - p1) - 1000 * (q2 - p2))/J2(w1,u1)

def RK4(u1,u2,w1,w2,p1,p2,q1,q2,x):
    for i in range(len(x) - 1):
        k11 = f1(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i],q2[i])
        k21 = f2(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i],q2[i])
```

```
        L11 = f3(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i]
    ,q2[i])
        L21 = f4(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i]
    ,q2[i])
        r11 = f5(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i]
    ,q2[i])
        r21 = f6(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i]
    ,q2[i])
        t11 = f7(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i]
    ,q2[i])
        t21 = f8(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i]
    ,q2[i])

        k12 = f1(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
        k22 = f2(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
        L12 = f3(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
        L22 = f4(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
        r12 = f5(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
        r22 = f6(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
        t12 = f7(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
        t22 = f8(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
```

```
    + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
   tep * t21/2)
66.
67.
68.         k13 = f1(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
69.         k23 = f2(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
70.         L13 = f3(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
71.         L23 = f4(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
72.         r13 = f5(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
73.         r23 = f6(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
74.         t13 = f7(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
75.         t23 = f8(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
76.
77.         k14 = f1(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
   3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
    + step * r23,q1[i] + step * t13,q2[i] + step * t23)
78.         k24 = f2(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
   3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
    + step * r23,q1[i] + step * t13,q2[i] + step * t23)
```

```
79.         L14 = f3(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
     + step * r23,q1[i] + step * t13,q2[i] + step * t23)
80.         L24 = f4(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
     + step * r23,q1[i] + step * t13,q2[i] + step * t23)
81.         r14 = f5(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
     + step * r23,q1[i] + step * t13,q2[i] + step * t23)
82.         r24 = f6(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
     + step * r23,q1[i] + step * t13,q2[i] + step * t23)
83.         t14 = f7(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
     + step * r23,q1[i] + step * t13,q2[i] + step * t23)
84.         t24 = f8(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
    3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
     + step * r23,q1[i] + step * t13,q2[i] + step * t23)
85.
86.         u1[i + 1] = u1[i] + step / 6 * (k11 + 2 * k12 + 2 * k13 + k
    14)
87.         u2[i + 1] = u2[i] + step / 6 * (k21 + 2 * k22 + 2 * k23 + k
    24)
88.         w1[i + 1] = w1[i] + step / 6 * (L11 + 2 * L12 + 2 * L13 + L
    14)
89.         w2[i + 1] = w2[i] + step / 6 * (L21 + 2 * L22 + 2 * L23 + L
    24)
90.         p1[i + 1] = p1[i] + step / 6 * (r11 + 2 * r12 + 2 * r13 + r
    14)
91.         p2[i + 1] = p2[i] + step / 6 * (r21 + 2 * r22 + 2 * r23 + r
    24)
92.         q1[i + 1] = q1[i] + step / 6 * (t11 + 2 * t12 + 2 * t13 + t
    14)
93.         q2[i + 1] = q2[i] + step / 6 * (t21 + 2 * t22 + 2 * t23 + t
    24)
94.
95.
96.     return [u1,u2,w1,w2,p1,p2,q1,q2]
97.
98.
99. if __name__ == "__main__":
100.
101.
102.       x = []
```

```
103.        temp = start
104.        while temp <= end:
105.            x.append(temp)
106.            temp += step
107.        #初始化
108.        u1 = [0 for i in range(len(x))]
109.        u2 = [0 for i in range(len(x))]
110.        w1 = [0 for i in range(len(x))]
111.        w2 = [0 for i in range(len(x))]
112.        p1 = [0 for i in range(len(x))]
113.        p2 = [0 for i in range(len(x))]
114.        q1 = [0 for i in range(len(x))]
115.        q2 = [0 for i in range(len(x))]
116.
117.        a = RK4(u1,u2,w1,w2,p1,p2,q1,q2,x)
118.
119.
120.        # with open("D:\Desktop\\T3.1.csv", "w", newline="") as csv
    file:
121.        #     writer = csv.writer(csvfile)
122.        #     writer.writerows(list(map(list, zip(*a))))
123.
124.        #画图
125.        plt.rcParams['font.sans-serif'] = ['SimHei']
126.        plt.rcParams['axes.unicode_minus'] = False
127.        plt.plot(x, u1, color="red" ,linewidth=1.0, linestyle="-")
     # 将散点连在一起
128.        plt.plot(x, w1, color="blue", linewidth=1.0, linestyle="-")
129.        plt.xlabel('时间/s')
130.        plt.ylabel('位移/m')
131.        plt.show()
132.
133.        plt.rcParams['font.sans-serif'] = ['SimHei']
134.        plt.rcParams['axes.unicode_minus'] = False
135.        plt.plot(x, p1, color="red", linewidth=1.0, linestyle="-")
     # 将散点连在一起
136.        plt.plot(x, q1, color="blue", linewidth=1.0, linestyle="-")
137.        plt.xlabel('时间/s')
138.        plt.ylabel('位移/m')
139.        plt.show()
140.
141.        cha = [(w1[i] - u1[i]) for i in range(len(w1))]
142.        plt.plot(x, cha, color="red", linewidth=1.0, linestyle="-")
      # 将散点连在一起
```

```
143.        plt.xlabel('时间/s')
144.        plt.ylabel('cha/m')
145.        plt.show()
146.
147.        cha = [(p1[i] - q1[i]) for i in range(len(w1))]
148.        plt.plot(x, cha, color="red", linewidth=1.0, linestyle="-")
      # 将散点连在一起
149.        plt.xlabel('时间/s')
150.        plt.ylabel('cha/m')
151.        plt.show()
152.
```

问题四模拟退火算法求解

```
1. import math
2. import matplotlib.pyplot as plt
3. import csv
4. import random
5.
6. #步长
7. step = 0.01
8. start = 0
9. end = 250
10.
11.J1 = 14137.79
12.
13.#退火初始温度
14.T = 0.5
15.
16.#体积
17.def V(u1):
18.    if u1 <= 2:
19.        return math.pi * u1
20.    else:
21.        return 2 * math.pi +(1-math.pow((2.8-u1)/0.8,3)) * (0.8 * m
   ath.pi/3)
22.
23.def J2(x2,x1):
24.    return 2433 * (0.25 + 0.2019575 + x2 - x1) * (0.25 + 0.2019575
   + x2 - x1) + 354.8125
25.
26.def f1(x,u1,u2,w1,w2,p1,p2,q1,q2):
27.    return u2
28.
```

```
def f2(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return (1760 * math.cos(1.9806 * x) - 528.5018 * u2 -V(u1) * 1025 * 9.8 + 80000 * (w1 - u1) + c1 * (w2 - u2)-(1-math.cos(p1+q1)) * 9.8 * 2433)*(1/(4866+1091.099))

def f3(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return w2

def f4(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return  (1-math.cos(p1+q1)) * 9.8 - 1 * (1/2433) * (80000 * (w1 - u1) + c1 * (w2 - u2))

def f5(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return p2

def f6(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return (2140 * math.cos(1.9806 * x) - 1655.909 * p2 - 8890.7 * p1 + 250000 * (q1 - p1) + d1 * (q2 - p2) - 1.2 * 4866 * 9.8 * math.sin(p1)) / (J1 + 7142.493)

def f7(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return q2

def f8(x,u1,u2,w1,w2,p1,p2,q1,q2):
    return  ((0.25 + 0.2019575 + w1 - u1) * math.sin(p1 + q1) * 9.8 * 2433- 250000 * (q1 - p1) - d1 * (q2 - p2))/J2(w1,u1)

def RK4(u1,u2,w1,w2,p1,p2,q1,q2,x):
    for i in range(len(x) - 1):
        k11 = f1(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i],q2[i])
        k21 = f2(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i],q2[i])
        L11 = f3(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i],q2[i])
        L21 = f4(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i],q2[i])
        r11 = f5(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i],q2[i])
        r21 = f6(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i],q2[i])
        t11 = f7(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i],q2[i])
```

```
59.         t21 = f8(x[i], u1[i], u2[i], w1[i], w2[i],p1[i],p2[i],q1[i]
    ,q2[i])
60.
61.         k12 = f1(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
62.         k22 = f2(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
63.         L12 = f3(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
64.         L22 = f4(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
65.         r12 = f5(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
66.         r22 = f6(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
67.         t12 = f7(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
68.         t22 = f8(x[i] + step / 2, u1[i] + step * k11 / 2, u2[i] + s
    tep * k21 / 2, w1[i] + step * L11 / 2, w2[i] + step * L21 / 2,p1[i]
     + step * r11/2,p2[i] + step * r21/2,q1[i] + step * t11/2,q2[i] + s
    tep * t21/2)
69.
70.
71.         k13 = f1(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
    tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
     + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
    tep * t22/2)
72.         k23 = f2(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
    tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
```

```
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
73.         L13 = f3(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
74.         L23 = f4(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
75.         r13 = f5(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
76.         r23 = f6(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
77.         t13 = f7(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
78.         t23 = f8(x[i] + step / 2, u1[i] + step * k12 / 2, u2[i] + s
   tep * k22 / 2, w1[i] + step * L12 / 2, w2[i] + step * L22 / 2,p1[i]
    + step * r12/2,p2[i] + step * r22/2,q1[i] + step * t12/2,q2[i] + s
   tep * t22/2)
79.
80.         k14 = f1(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
   3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
    + step * r23,q1[i] + step * t13,q2[i] + step * t23)
81.         k24 = f2(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
   3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
    + step * r23,q1[i] + step * t13,q2[i] + step * t23)
82.         L14 = f3(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
   3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
    + step * r23,q1[i] + step * t13,q2[i] + step * t23)
83.         L24 = f4(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
   3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
    + step * r23,q1[i] + step * t13,q2[i] + step * t23)
84.         r14 = f5(x[i] + step, u1[i] + step * k13, u2[i] + step * k2
   3, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i]
    + step * r23,q1[i] + step * t13,q2[i] + step * t23)
```

```
        r24 = f6(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i] + step * r23,q1[i] + step * t13,q2[i] + step * t23)
        t14 = f7(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i] + step * r23,q1[i] + step * t13,q2[i] + step * t23)
        t24 = f8(x[i] + step, u1[i] + step * k13, u2[i] + step * k23, w1[i] + step * L13, w2[i] + step * L23,p1[i] + step * r13, p2[i] + step * r23,q1[i] + step * t13,q2[i] + step * t23)

        u1[i + 1] = u1[i] + step / 6 * (k11 + 2 * k12 + 2 * k13 + k14)
        u2[i + 1] = u2[i] + step / 6 * (k21 + 2 * k22 + 2 * k23 + k24)
        w1[i + 1] = w1[i] + step / 6 * (L11 + 2 * L12 + 2 * L13 + L14)
        w2[i + 1] = w2[i] + step / 6 * (L21 + 2 * L22 + 2 * L23 + L24)
        p1[i + 1] = p1[i] + step / 6 * (r11 + 2 * r12 + 2 * r13 + r14)
        p2[i + 1] = p2[i] + step / 6 * (r21 + 2 * r22 + 2 * r23 + r24)
        q1[i + 1] = q1[i] + step / 6 * (t11 + 2 * t12 + 2 * t13 + t14)
        q2[i + 1] = q2[i] + step / 6 * (t21 + 2 * t22 + 2 * t23 + t24)
        # print(p2[i])

def newans(one,two,st1,st2):
    rd = random.randint(1, 4)
    one1 = one
    two1 = two

    if rd == 1:
        one1 += st1
    if rd == 2:
        one1 -= st1
    if rd == 3:
        two1 += st2
    if rd == 4:
        two1 -= st2
    if one1 <= 100000 and two1 <= 100000 and one1 >= 0 and two1 >= 0:
        return [one1,two1]
```

```
114.     else:
115.         return [one,two]
116. 
117. #主函数
118. if __name__ == "__main__":
119. 
120.     x = []
121.     temp = start
122.     while temp <= end:
123.         x.append(temp)
124.         temp += step
125. 
126.     list1 = range(0,100001,5000)
127.     list2 = range(0,100001,5000)
128. 
129.     aver = []
130.     u1 = [0 for i in range(len(x))]
131.     u2 = [0 for i in range(len(x))]
132.     w1 = [0 for i in range(len(x))]
133.     w2 = [0 for i in range(len(x))]
134.     p1 = [0 for i in range(len(x))]
135.     p2 = [0 for i in range(len(x))]
136.     q1 = [0 for i in range(len(x))]
137.     q2 = [0 for i in range(len(x))]
138. 
139.     c1 = 59738
140.     d1 = 100000
141. 
142.     ii = 0
143. 
144.     for i in range(10000):
145.         u1[0] = 0
146.         u2[0] = 0
147.         w1[0] = 0
148.         w2[0] = 0
149.         p1[0] = 0
150.         p2[0] = 0
151.         q1[0] = 0
152.         q2[0] = 0
153.         [tempc, tempd] = newans(c1, d1, 1, 1)
154. 
155.         tc = c1
156.         td = d1
157. 
```

```
158.         c1 = tempc
159.         d1 = tempd
160.
161.         RK4(u1, u2, w1, w2, p1, p2, q1, q2, x)
162.         p = [c1 * (u2[i] - w2[i]) * (u2[i] - w2[i]) + d1 * (p2[i]
    - q2[i]) * (p2[i] - q2[i]) for i in range(15000, 25000)]
163.         tempii = sum(p) / len(p)
164.
165.         if tempii > ii:
166.             print(c1,d1,tempii)
167.             ii = tempii
168.         else:
169.             rr = random.randint(0, 100000000)/100000000
170.             if rr < math.exp(-1 * (ii - tempii) / T):
171.                 print(c1, d1, tempii)
172.                 ii = tempii
173.             else:
174.                 c1 = tc
175.                 d1 = td
176.
177.         T = T * 0.5
178.
179.     for c in list1:
180.         for d in list2:
181.             print(c,d)
182.             c1 = c
183.             d1 = d
184.
185.             u1 = [0 for i in range(len(x))]
186.             u2 = [0 for i in range(len(x))]
187.             w1 = [0 for i in range(len(x))]
188.             w2 = [0 for i in range(len(x))]
189.             p1 = [0 for i in range(len(x))]
190.             p2 = [0 for i in range(len(x))]
191.             q1 = [0 for i in range(len(x))]
192.             q2 = [0 for i in range(len(x))]
193.
194.             RK4(u1,u2,w1,w2,p1,p2,q1,q2,x)
195.             p = [c * (u2[i] - w2[i]) * (u2[i] - w2[i]) + d * (p2[
   i] - q2[i]) * (p2[i] - q2[i])for i in range(15000,25000)]
196.             aver.append([c,d,sum(p) / len(p)])
197.
198.     with open("D:\Desktop\\T4.csv", "w", newline="") as csvfile:
199.         writer = csv.writer(csvfile)
```

```
200.         writer.writerows(list(map(list, zip(*aver))))
201.
202.     # plt.rcParams['font.sans-serif'] = ['SimHei']
203.     # plt.rcParams['axes.unicode_minus'] = False
204.     # plt.plot(x, u1, color="red" ,linewidth=1.0, linestyle="-")
    # 将散点连在一起
205.     # plt.plot(x, w1, color="blue", linewidth=1.0, linestyle="-")
206.     # plt.xlabel('时间/s')
207.     # plt.ylabel('位移/m')
208.     # plt.show()
209.     #
210.     # plt.rcParams['font.sans-serif'] = ['SimHei']
211.     # plt.rcParams['axes.unicode_minus'] = False
212.     # plt.plot(x, p1, color="red", linewidth=1.0, linestyle="-")
    # 将散点连在一起
213.     # plt.plot(x, q1, color="blue", linewidth=1.0, linestyle="-")
214.     # plt.xlabel('时间/s')
215.     # plt.ylabel('位移/m')
216.     # plt.show()
217.     #
218.     # cha = [(w1[i] - u1[i]) for i in range(len(w1))]
219.     # plt.plot(x, cha, color="red", linewidth=1.0, linestyle="-")
     # 将散点连在一起
220.     # plt.xlabel('时间/s')
221.     # plt.ylabel('cha/m')
222.     # plt.show()
223.     #
224.     # cha = [(p1[i] - q1[i]) for i in range(len(w1))]
225.     # plt.plot(x, cha, color="red", linewidth=1.0, linestyle="-")
     # 将散点连在一起
226.     # plt.xlabel('时间/s')
227.     # plt.ylabel('cha/m')
228.     # plt.show()
229.
```