基于微分方程迭代求解方法的波浪能装置动力学建模和评估

波浪能是一种清洁的海洋可再生能源，具有绿色环保，储量丰富的特点，近期频繁受到科研工作者的广泛关注。因此，如何利用波浪能实现能量的最大转换效率，从而实现波浪能规模化利用，成为社会关注的重点。本文的目标是通过物理学原理，建立在波浪周期性辐射波作用下波浪能俘获装置的动力学模型，同时通过优化算法，确定使输出功率最大化的相关参数。

对于问题一，根据牛顿定律所反映的受力平衡与加速度的关系式，采用整体法和隔离法相结合，我们分别建立了对于浮子和振子运动状态下的非线性运动学微分方程。同时根据受力平衡原理对模型初始状态求解后，我们采用**三四阶龙格库塔法**，对微分关系式进行泰勒展开并进行迭代差分，以求得最终的微分方程数值解，解出指定时间内的组合体运动状态和平均输出功率。结果可以发现，弹簧-质量系统在经过初期的胁振后，在约 30 秒后便能达到相对稳定的状态，稳定且有周期性地输出功率。在题目给的第二种情况中,时间 10s 时,浮子位移、浮子速度、振子位移、振子速度分别为:-0.20647,-0.65284,-0.93525,-0.69996.(其余结果见论文,第三问同上)

我们还对提出的迭代算法进行了误差累积传播稳定度分析以证明算法的稳定有效性，以及对波浪频率的灵敏度分析。**结果表明频率提升 1%,总平均输出功率便会提高约 7%**。

针对问题二，我们考虑在 20 秒内的输出功率。使用（1）中所得方程，对于第一小问我们采用了**左右逼近变步长搜索算法**，得到在阻尼系数取 40355.97 时，平均输出功率达到约 163.8 瓦的最大值;对于第二位的双变量优化问题，我们采用了**自适应变步长网格搜索算法**搜索最优解，得到以(40355.97,0)为解组合时，系统平均输出功率最大，仍为 163.8 瓦。

针对问题三，在问题一的基础上，考虑组合体的纵摇角度，我们引入浮子，振子偏移角作为新的变量并对（1）中所得**加速度-受力模型**进行修正。同时，我们对组合体所受力矩以及内外力产生的力矩以及产生的科氏力进行分析，建立了**角加速度-力矩模型**，以与上式联立求解。为了简化问题，我们采用了力分解的方法，分别建立浮子中心轴随动坐标系和世界坐标系。我们在第一问三四阶龙格库塔法的基础上，设计了一种**基于简单变换的交替迭代法**，对两个坐标系下所解得动力学相关矢量进行交替迭代，交替更新，在保证精确度的同时简便快速对非线性方程进行求解。对上述结果分析可以发现，由于旋转弹簧-阻尼系统的存在，振子的绝对角位移与浮子高度耦合，且可以在短时间内结束胁振并稳定输出功率。

问题四和问题二原理类似，同为双变量优化问题。使用问题二模型求解，得到以解组合（39837，100000）为参数时，功率取得最大值 56.896W。

关键词：动力学建模，受力分析，左右逼近变步长搜索算法，自适应变步长网格搜索算法，交替迭代法

一、问题重述

1.1 问题背景

随着经济不断发展，人们对能源的需求愈来愈大，传统的不可再生能源已不能满足人们日常的需求。为了扭转这一局面，世界各地的研究者纷纷把目光投向新能源，其中最具有潜力的便是海洋新能源的波浪能，由于分布广泛、能量密度大，可开发利用时间长等特点，成为广大研究者的研究重点。

目前，由于海波复杂多变，开发利用波浪能仍存在许多问题，对于波浪能的开采的研究仍处在初级阶段。因此，如何提高波浪能装置效率是波浪能开采研究的重点。

在题目中研究的波浪能装置类比振荡浮子式波能发电装置，通过浮子在波浪中的垂荡、纵摇等运动将波浪的机械能转化为发电装置的动能，再通过 PTO（能量输出系统）传向发电机，进而转化成电能。

1.2 问题提出

题目中提到了一种波浪能装置，由浮子、振子、中轴以及能量传输系统（PTO，由弹簧和阻尼器构成）在波浪的作用下，浮子运动并带动振子运动，通过二者的相对运动来驱动阻尼器做功。要求建立数学模型解决以下问题：

- 考虑浮子在波浪中只做垂荡运动，建立浮子与振子的运动模型，利用附件提供的参数值，分别对两种情况计算浮子和振子前 40 个波浪周期内时间间隔为 0.2s 的垂荡位移的速度。
- 考虑浮子在波浪中只做垂荡运动，分别对两种情况建立阻尼器的最优阻尼系数的数学模型，让 PTO 系统的平均输出功率最大。
- 考虑浮子在波浪中做垂荡和纵摇运动，除了直线阻尼器，在转轴上还安装了旋转阻尼器和扭转弹簧，建立浮子与振子的运动模型，计算浮子与振子前 40 个波浪周期内时间间隔为 0.2s 的垂荡位移与速度和纵摇角位移与角速度
- 考虑浮子只做垂荡和纵摇运动，确定两个阻尼器的最优阻尼系数的数学模型，计算最大输出功率。

二、问题分析

2.1 问题一的分析

问题一要求对波浪能装置在波浪辐射波影响下的运动进行动力学建模，即根据牛顿定律列出加速度-受力微分方程式并求解即可。由于组合体运动时间区间较长，各力耦合性较高，综合考虑算法复杂度和求解精确度，我们选择龙格库塔法使用泰勒展开式对离散化后的微分方程组用迭代算法迭代求解。同时由于上述原因，为了确定算法的有效性，我们还需通过添加随机误差的方式对算法误差传递累积的稳定性进行分析。

2.2 问题二的分析

问题二含有一个单变量优化和一个双变量优化问题。由于目标变量值域较小，解空间不大，且（1）中微分方程求解算法复杂度较低，我们选择采用网格搜索

法等遍历算法进行求解。在下文中我们将使用自适应步长等机制来构建算法，以在保证解精确度的前提下提高搜索速度。

2.3 问题三的分析

问题三相对问题二多考虑了力矩的作用力，其机制与（1）类似。我们注意到角速度矢量在不同的转轴上难以叠加，故为简化求解假设组合体旋转轴位于扭转弹簧附近。对力矩的分析，我们除分析其所受各个方向外力以外，还应考虑科氏力所提供的力矩。除此之外，注意到对于此两自由度运动模型计算各力矢量叠加较为繁琐，故我们采用了建立随动坐标系的方式，对于垂荡仅分析各力作用在其组合体中心轴上的分量，同时在每一次迭代后通过简单变换重建世界坐标系下的各运动学相关矢量，从而化双自由度为单自由度，体现了降维的思想，简化求解过程。

2.4 问题四的分析

问题四和问题二问题类似，故直接套用第二问双变量优化算法求解即可。

三、模型假设

(1) 忽略中轴、底座、隔层及 PTO 的质量和各种摩擦。
(2) 考虑海水是无粘及无旋的。
(3) 问题一和问题二只考虑浮子在波浪中只做垂荡运动，问题三和问题四只考虑浮子在波浪重做垂荡和纵摇运动。
(4) 假设初始时刻浮子和振子是平衡于静水中
(5) 忽略浮子壳体厚度，以及旋转弹簧铰接点与圆柱圆锥交界面的距离

四、符号说明

符号	说明
F_f	浮力
ρ	海水密度
g	重力加速度
F_L	波浪激励力
F_{CZ}	波浪兴波阻尼力
f	入射波浪频率
M	浮子质量
m	振子质量
F_t	弹簧弹力
F_Z	直线阻尼力
K_t	弹簧刚度
l_y	弹簧原长
K	直线阻尼系数
V_x	浮子与振子相对速度
M_e	垂荡附加质量
h_{co}	浮子圆锥部分高度
$\boldsymbol{\theta}$	浮子角位移
$\boldsymbol{\alpha}$	振子相对角位移
M_z	纵摇激励力矩
M_H	静水恢复力矩
M_{ZZ}	纵摇阻尼力矩

其他符号将在下文中说明。

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 受力分析

① 对浮子和振子整体分析

由于初始时刻浮子和振子平衡于静水中，根据受力关系有：

$$\boldsymbol{F_{f=}\rho g V=(m+M)g} \tag{1}$$

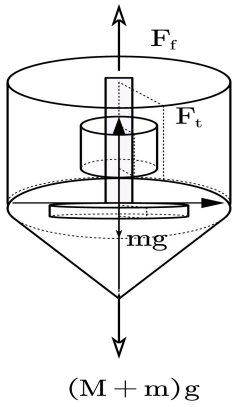


图 1 组合体受力分析图初始状态

根据公式(1)，我们可以求最初组合体(浮子+振子)排开液体的体积：

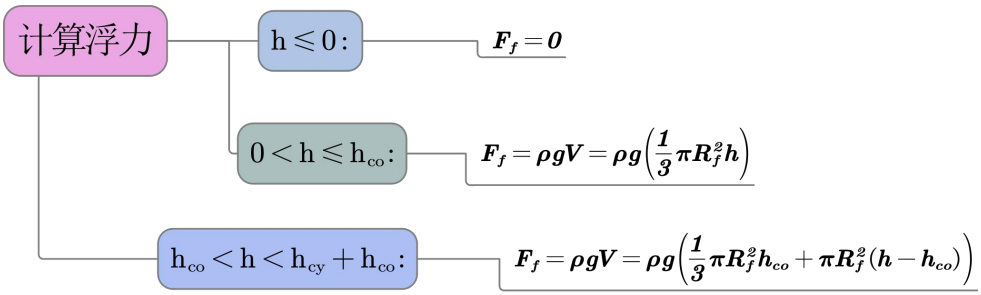
$$\boldsymbol{V=\frac{(m+M)}{\rho}} \tag{2}$$

从而求出最初物体沉浸在水中的高度：

$$\boldsymbol{H_0=\frac{\left(V-\frac{1}{3}\pi h_{co}\right)}{\pi}+h_{co}} \tag{3}$$

因为浮子在波浪中做垂荡运动，所以在计算浮力的时候其沉浸在水中的高度 h 要减去垂荡位移s_C，结合浮子是质量均匀分布的圆柱和圆锥壳体的特点，我们进行分类讨论得对于不同浸没高度对浮力得计算公式：

(R_f为浮子底半径)



其中，$\boldsymbol{h=H_0-S_c}$

根据题意，还知组合体受波浪激励力F_L和波浪兴波阻尼力F_Z ：

$$\boldsymbol{F_L=fcoswt} \tag{4}$$

$$\boldsymbol{F_{CZ}=K_cV_z} \tag{5}$$

其中，F_{CZ}的方向与组合体的速度方向相反
整体受力分析如图所示：

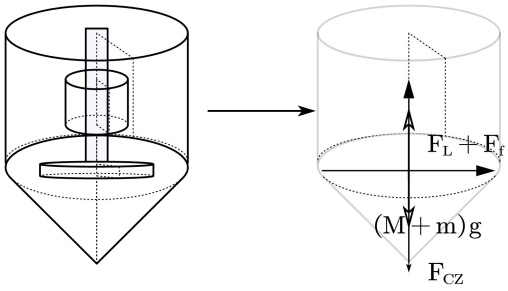


图 2 整体受力分析图

②对振子单独分析

由题意可知，振子是穿在中轴上的圆柱体，通过 PTO 与中轴底座连接。所以我们对振子单独进行受力分析，可以得出，振子除自身重力外，还受弹簧的弹力和直线阻尼的阻力:

$$\boldsymbol{F_t=K_t(l_t-l_y)} \tag{7}$$

$$\mathbf{F}_Z=\mathbf{kv}_x \tag{8}$$

其中l_t为弹簧此时的长度
由题目中,该直线阻尼器的系数分为两种情况，将会分别分析。第一种情况:直线阻尼器的阻尼系数为 10000 N·s/m；第二种情况: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。

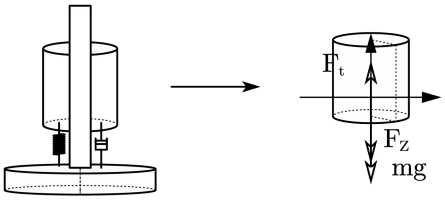


图 3 振子受力分析图

5.1.2 垂荡动力学方程的建立

在本节中，我们使用牛顿定律，建立波浪能装置垂荡非线性动力方程。本节中我们定竖直向上为正方向。

设$\boldsymbol{a_1},,\boldsymbol{a_2}$分别为浮子，振子的加速度，同时设$\boldsymbol{v_1},\boldsymbol{v_2}$为其瞬时速度。由牛顿定律可知,一组合体内各个物体加速度和质量乘积矢量和等于组合体所受外力矢量和。故在波浪能装置运动的每个时刻均有：

$$(\boldsymbol{M+m_e})\boldsymbol{a_1}+\boldsymbol{ma_2}=-(\boldsymbol{M+m})\boldsymbol{g}+\boldsymbol{F_f}+\boldsymbol{F_L}+\boldsymbol{sign(v1)F_{CZ}}$$

其中兴波阻力F_{CZ}的方向永远与浮子的运动方向相反。

同时，对于振子，其加速度-受力关系式为：

$$\boldsymbol{ma_2}=-\boldsymbol{mg}+\boldsymbol{F_t}+\boldsymbol{sign}(v_2-\boldsymbol{v1})\boldsymbol{F_Z}$$

直线阻尼的阻尼力F_Z方向始终与振子相对浮子的运动方向相反。

以上即为该弹簧-质量组合体的加速度-受力方程。分别设浮子，振子在某一时刻 t 的 d 对于初始状态的位移为函数$\boldsymbol{S_1(t)},\boldsymbol{S_2(t)}$，则有：

$$\begin{cases}(\boldsymbol{M+m_e})\ddot{\boldsymbol{S_1}}+\boldsymbol{m}\ddot{\boldsymbol{S_2}}=-(\boldsymbol{M+m})\boldsymbol{g}+\boldsymbol{F_f}+\boldsymbol{F_L}+\boldsymbol{sign}(\dot{\boldsymbol{S_1}})\boldsymbol{F_{CZ}}\\ \boldsymbol{m}\ddot{\boldsymbol{S_2}}=-\boldsymbol{mg}+\boldsymbol{F_t}+\boldsymbol{sign}(\dot{\boldsymbol{S_2}}-\dot{\boldsymbol{S_1}})\boldsymbol{F_Z}\end{cases}$$

设弹簧原长为$\boldsymbol{L}$，长度与时间的函数为$\boldsymbol{L(t)}$ ，以上式中各力与$\boldsymbol{S_1(t)},\boldsymbol{S_2(t)}$的关系式为:

$$\begin{cases}\boldsymbol{F_t}=\boldsymbol{k_t}(\boldsymbol{L}-\boldsymbol{L(t)})\\ \boldsymbol{L(t)}=\boldsymbol{L(0)}+\boldsymbol{S_2(t)}-\boldsymbol{S_1(t)}\\ \boldsymbol{F_f}=\boldsymbol{\rho gV}\\ \boldsymbol{F_L}=\boldsymbol{f cos(\omega t)}\\ \boldsymbol{F_Z}=\boldsymbol{k}(\dot{\boldsymbol{S_2}}-\dot{\boldsymbol{S_1}})\\ \boldsymbol{F_{CZ}}=\boldsymbol{k_C}\dot{\boldsymbol{S_1}}\end{cases}$$

其中$\boldsymbol{\rho}$为海水密度，$\boldsymbol{V}$为组合体水下部分的体积。$\boldsymbol{k_t}$为弹簧刚性系数，$\boldsymbol{k}$为直线阻尼器的阻尼系数

对于浮力的计算，需根据不同的浸水深度 h，计算不同的水下部分体积 V

联立以上方程，则得到了垂荡条件下组合体的非线性动力学二阶微分方程

$$\begin{cases}(\boldsymbol{M+m_e})\ddot{\boldsymbol{S_1}}+\boldsymbol{m}\ddot{\boldsymbol{S_2}}=-(\boldsymbol{M+m})\boldsymbol{g}+\boldsymbol{F_f}+\boldsymbol{F_L}+\boldsymbol{sign}(\dot{\boldsymbol{S_1}})\boldsymbol{F_{CZ}}\\ \boldsymbol{m}\ddot{\boldsymbol{S_2}}=-\boldsymbol{mg}+\boldsymbol{F_t}+\boldsymbol{sign}(\dot{\boldsymbol{S_2}}-\dot{\boldsymbol{S_1}})\boldsymbol{F_Z}\\ \boldsymbol{F_t}=\boldsymbol{k_t}(\dot{\boldsymbol{S_2}}-\dot{\boldsymbol{S_1}})\\ \boldsymbol{F_f}=\boldsymbol{\rho gV}(\boldsymbol{h(t)})\\ \boldsymbol{F_L}=\boldsymbol{f cos(\omega t)}\\ \boldsymbol{F_Z}=\boldsymbol{k}(\dot{\boldsymbol{S_2}}-\dot{\boldsymbol{S_1}})\\ \boldsymbol{h(t)}=\boldsymbol{h(0)}+\boldsymbol{S_2}\\ \boldsymbol{F_{CZ}}=\boldsymbol{k_{CZ}}\dot{\boldsymbol{S_1}}\end{cases}$$

对于此比较常规的二阶微分方程组，由于难以解出解析解，故选择使用**龙格库塔算法的基本原理**来求解常微分方程组

5.1.3 使用龙格库塔法对微分方程组的求解

龙格库塔算法的主要原理为，在 t 点的附近选取一些特定的点，后将其的函数值进行线性组合，使用组合值代替泰勒展开中 t 点的导数值

根据微分中值定理，存在点$\delta\in[x_{n,}x_{n+1}]$,使得：

$$\frac{y(x_{n+1})-y(x_{n,})}{h}=y(\dot{\delta})$$

从而利用$\dot{y}=f$得：

$$y(x_{n+1}) = y(x_{n,}) + hf(\delta, y(\delta))$$

根据龙格库塔算法，可以改写成：

$$\begin{cases} y_{n+1} = y_n + \dfrac{h}{2}(K_1 + K_2) \\ K_1 = f(x_n, y_n) \\ K_2 = f(x_{n+1}, y_n + hK_1) \end{cases}$$

故在下文求解中，我们将采用三四阶龙格库塔法对微分方程组进行求解。

5.1.3.1 方程初始条件的确立

由以上受力分析可知，在初始静水条件中组合体处于受力平衡状态。其力学方程组为

$$\begin{cases} \boldsymbol{mg = k_t|L - L(0)|} \\ \boldsymbol{(m+M)g = V(h(0)} \\ \boldsymbol{L(0) \le L} \end{cases}$$

联立解得：

$$\begin{cases} \boldsymbol{L(0) = 0.201} \\ \boldsymbol{h(0) = 2.8} \end{cases}$$

故该微分方程组起始条件可表示为:

$$\begin{cases} \boldsymbol{L(0) = 0.201} \\ \boldsymbol{h(0) = 2.8} \\ \boldsymbol{\ddot{S_1}(0) = 0} \\ \boldsymbol{\ddot{S_2}(0) = 0} \\ \boldsymbol{\dot{S_1}(0) = 0} \\ \boldsymbol{\dot{S_2}(0) = 0} \\ \boldsymbol{S_1(0) = 0} \\ \boldsymbol{S_2(0) = 0} \end{cases}$$

5.1.3.2 离散化求解

设最小时间步长 $\boldsymbol{\Delta t}$。由上文原理，将该方程离散化为：

$$\begin{cases} \boldsymbol{S_1(t+1) = S_1(t) + \dot{S_1}(t)\Delta t + \frac{1}{2}\ddot{S_1}(t)\Delta t^2 + \frac{1}{6}\frac{\ddot{S_1}(t) - \ddot{S_1}(t-1)}{\Delta t}\Delta t^3} \\ \boldsymbol{S_2(t+1) = S_2(t) + \dot{S_1}(t)\Delta t + \frac{1}{2}\ddot{S_1}(t)\Delta t^2 + \frac{1}{6}\frac{\ddot{S_1}(t) - \ddot{S_1}(t-1)}{\Delta t}\Delta t^3} \\ \boldsymbol{\dot{S_1}(t+1) = \dot{S_1}(t) + \ddot{S_1}(t)\Delta t + \frac{\ddot{S_1}(t) - \ddot{S_1}(t-1)}{2\Delta t}\Delta t^2} \\ \boldsymbol{\dot{S_2}(t+1) = \dot{S_2}(t) + \ddot{S_2}(t)\Delta t + \frac{\ddot{S_2}(t) - \ddot{S_2}(t-1)}{2\Delta t}\Delta t^2} \\ \boldsymbol{(M+m_e)\ddot{S_1}(t) + m\ddot{S_2}(t) = -(M+m)g + F_f(t) + F_L(t) + sign\left(\dot{S_1}(t)\right)F_{CZ}} \\ \boldsymbol{m\ddot{S_2}(t) = -mg + F_t + sign(\dot{S_2}(t) - \dot{S_1}(t))F_Z} \\ \boldsymbol{F_t(t) = k_t\left(\dot{S_2}(t) - \dot{S_1}(t)\right)} \\ \boldsymbol{F_f(t) = \rho g V\left(h(t)\right)} \\ \boldsymbol{F_{L(t)} = f\cos(\omega t)} \\ \boldsymbol{F_Z(t) = k\left(\dot{S_2}(t) - \dot{S_1}(t)\right)} \\ \boldsymbol{h(t) = h(0) + S_2(t)} \\ \boldsymbol{F_{CZ}(t) = k_{CZ}\dot{S_1}(t)} \end{cases}$$

我们使用迭代算法来求解以上离散化的方程组。流程如下所示：

STEP1:设置总时长 T，时间步长 Δt。建立多维数组 P
STEP2:设置 t=0,i=0。将微分方程初始条件输入程序，记为 P[0]
STEP3:根据上述离散化微分方程,通过 P[i]所储存数据计算一个时间步长后各参数的值，记为 P[i+1]
STEP4:t=t+Δt。i=i+1 若 t>T,则结束程序，否则返回 STEP3。

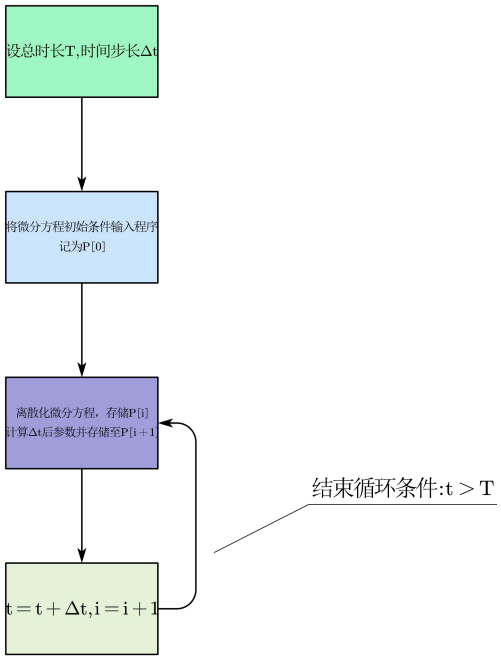


图 4 迭代算法图 1

5.1.4 模型求解结果

本问中两种情况所得结果分别如下图所示。总时长 $T = 40\frac{2\pi}{\omega}$的相关数据已写入相关材料。注意**以下提及物理量均为在世界坐标系下的绝对物理量，取竖直向上为正方向。**

① 情况一结果

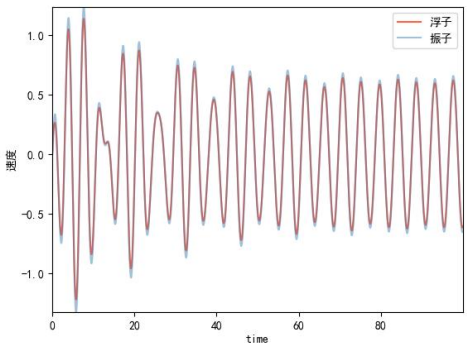


图 5 振子/浮子 速度-时间关系图

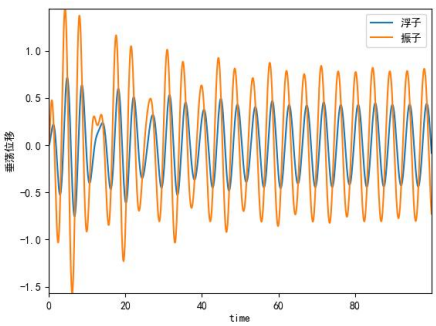


图 6 振子/浮子 垂荡位移-时间关系图

时间t	浮子位移	浮子速度	振子位移	振子速度
10s	-0.19129	-0.64101	-0.90634	-0.69398
20s	-0.59136	-0.24089	-0.90783	-0.27274
40s	0.28488	0.31276	0.62863	0.33265
60s	-0.31490	-0.47953	-0.84774	-0.51583
100s	-0.08378	-0.60427	-0.72734	-0.64308

表 1 浮子振子的运动状态表

② 情况二结果

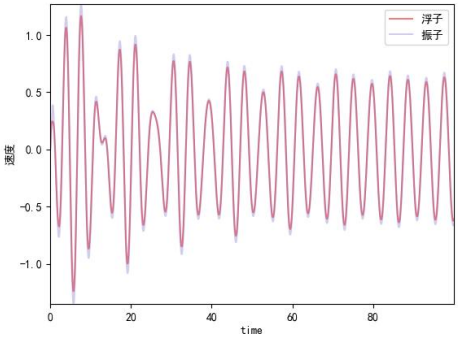


图 7 振子/浮子 速度-时间关系图

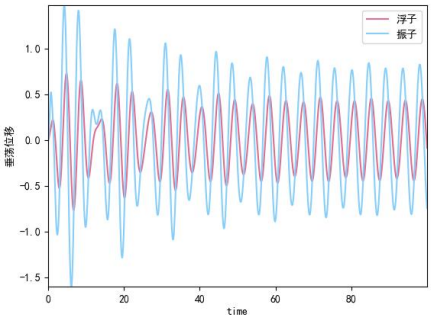


图 8 振子/浮子 垂荡位移-时间关系图

时间t	浮子位移	浮子速度	振子位移	振子速度
10s	-0.20647	-0.65284	-0.93525	-0.69996
20s	-0.61185	-0.25478	-0.939	-0.27702
40s	0.26815	0.29499	0.59164	0.31216
60s	-0.32769	-0.49172	-0.87604	-0.52581
100s	-0.08864	-0.60998	-0.74402	-0.65025

表 2 浮子振子的运动状态表

根据对上述所得结果进行分析可以发现，弹簧-质量系统在经过初期的胁振后，在约 30 秒后便能达到相对稳定的状态，稳定且有周期性地输出功率。

5.1.5 算法稳定度分析

微分方程组是一个对初始参数条件和参数误差高度敏感的非线性系统。使用连续差分法求解微分方程组由于其每一步迭代实际都在以各种方式逼近目标数值解，故可能会产生误差累积传递，最终导致最后的求解结果偏离实际解。在本节中，我们对上述求解方法的稳定性进行分析，以确定上述算法的误差累积传递程度，从而确定算法的有效性。

对于目标物理量$S_1, \dot{S}_1, \ddot{S}_1, S_2, \dot{S}_2, \ddot{S}_2$，我们对其初始状态值施加一个大小为 0.1 的随机偏移后，运行算法将所得结果与正常结果对比，计算其误差累积量。计算结果如下表所示：

物理量	期望结果	实际结果	累计误差	误差传播率
S_1	0.14	0.12	0.02	20%
$\dot{S}_1$	-0.43	-0.44	0.01	10%
$\ddot{S}_1$	-0.43	-0.43	<0.01	<10%
S_2	-0.28	-0.18	0.1	100%
$\dot{S}_2$	-0.44	-0.45	0.01	10%
$\ddot{S}_2$	-0.48	-0.48	<0.01	<10%

表 3 误差对比图

经分析知，对S_2施加的随机误差应发了灵敏的弹簧-阻尼系统的非正常形变，从而导致了较大的误差传播率。**基于此原因，我们认为S_2相关误差数据可以忽略。**其他误差数据表明上述算法稳定性较好，误差传播率低。所以，该算法对于微分方程组求解是有效的。

5.1.6 模型灵敏度分析

为了研究波浪激励波频率对装置平均输出功率的影响，在本节中我们对波浪激励波频率做模型的灵敏度分析。对于两个变量 x，y，x 对 y 的灵敏度为$S(x,y) = \frac{dy}{dx}\frac{x}{y}$。

以第一问中波浪频率$\omega = 1.4005$ 为起始数据，以 0.0001 为步长，以 20 秒为总时间，计算 10000 次，所得频率-平均功率关系图如下图所示。

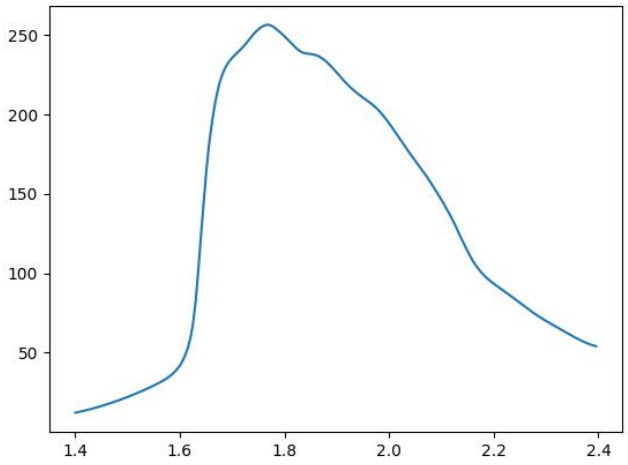


图 9 频率-平均功率关系图

通过分析得出，波浪能俘获装置的平均输出功率对于波浪频率极为敏感。波浪周期过小，振子位置难以回正，影响做功；波浪周期过大，相同时间内振子所受力过小，同样影响平均输出功率。观察可得对于第一问条件下，最优波浪频率在 1.8/s 左右。

计算得模型在$\omega = 1.4005$ 处关于其的灵敏度 $S(\omega, P) = 7.38$。即波浪激励力频率提高 1%，平均输出功率可以提高约 7.38%。

5.2 问题二模型的建立与分析

5.2.1 阻尼器输出功率

设直线阻尼器瞬时输出功率为 P,则有:

$$\boldsymbol{P = \frac{1}{2} K V^2}$$

故在 T 时间内总平均功率为：

$$\boldsymbol{\overline{P} = \frac{\int_0^T \frac{1}{2} K V^2 dt}{T}}$$

同样，我们使用龙格库塔法差分迭代计算其数值积分。得到：

$$\begin{cases} \boldsymbol{\overline{P} = \frac{\sum P(t)}{T}} \\ \boldsymbol{P(t+1) = \frac{1}{2} \Delta t K (\dot{S}(t) + \ddot{S}(t) \Delta t + \frac{\ddot{S}(t) - \ddot{S}(t-1)}{2\Delta t} \Delta t^2)^2} \end{cases}$$

经观察发现，平均输出功率在约 20s 后收敛至某一确定值。**下文计算所提平均功率，均指 20s 内得平均输出功率。**

5.2.2 情况一:常量阻尼系数

决策变量：直线阻尼系数 K

目标函数：PTO 平均输出功率

首先，我们利用问题一建立的模型以及动力学方程，结合附件中给的参数，我们在[0,100000]之间均匀选取 100 个点绘制出了直线阻尼系数 K 与 PTO 平均输出功率$\overline{P}$的关系图：

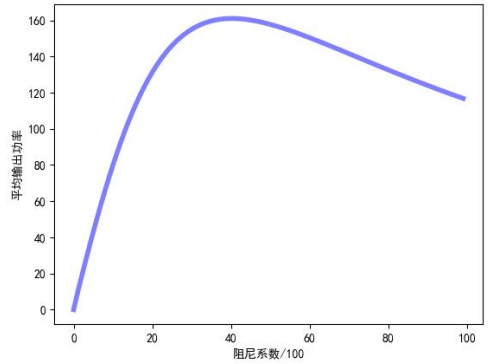


图 10

随着 K 的增大，$\overline{P}$呈先增大后减小（类斜抛物线）的变化趋势。由于决策变量唯一且迭代算法运算时间较短，故确定**左右逼近变步长搜索**的方法找到最优的阻尼系数，该算法的特性是：当目标函数值点远离最大功率点时，加大步长以获得更快的收敛速度；**当目标函数值点接近最大功率点时，减小步长以获得较小的稳态误差。**优化过程如下：

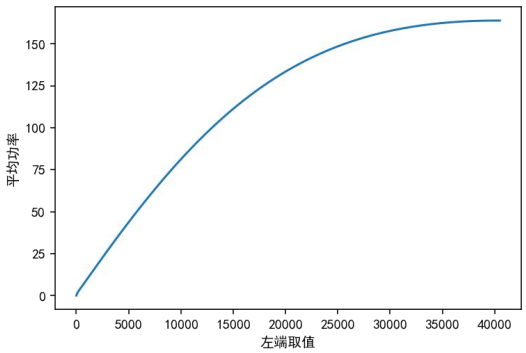


图 11

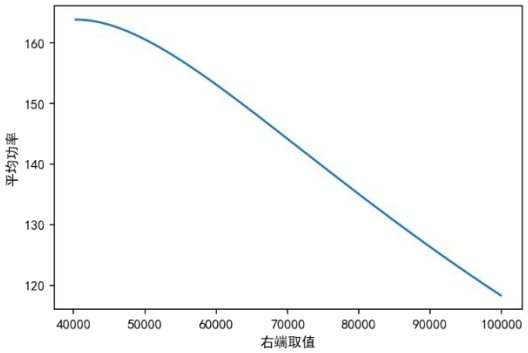


图 12

具体算法步骤如下：

STEP1:始化左端初值 left 和右端初值 right，设置精度 ThreshHold 以及初始步长 step，存储选取的变量值

STEP2:#*开始迭代左端逼近*

当精度大于所需的阈值时：

左端取值 left 自增步长 step

计算 left 对应的功率

分别存储取值以及对应的功率

若此时的取值小于上一个 left 的取值：

left 回溯至上一个 left 取值

根据相邻的 left 差值缩短步长 step

重置精度差值 delta

#右端逼近

与左端类似，从右端开始减少 step

STEP3：*#检查二者的取值，输出结果*

检查左右两者的取值，在二者之间重复上述步骤，直至精度满足条件，输出结果

最后得出当阻尼系数取值$K = 40355.97$时，在时长为 20s 内取得最大平均功率$P = 163.83$W。

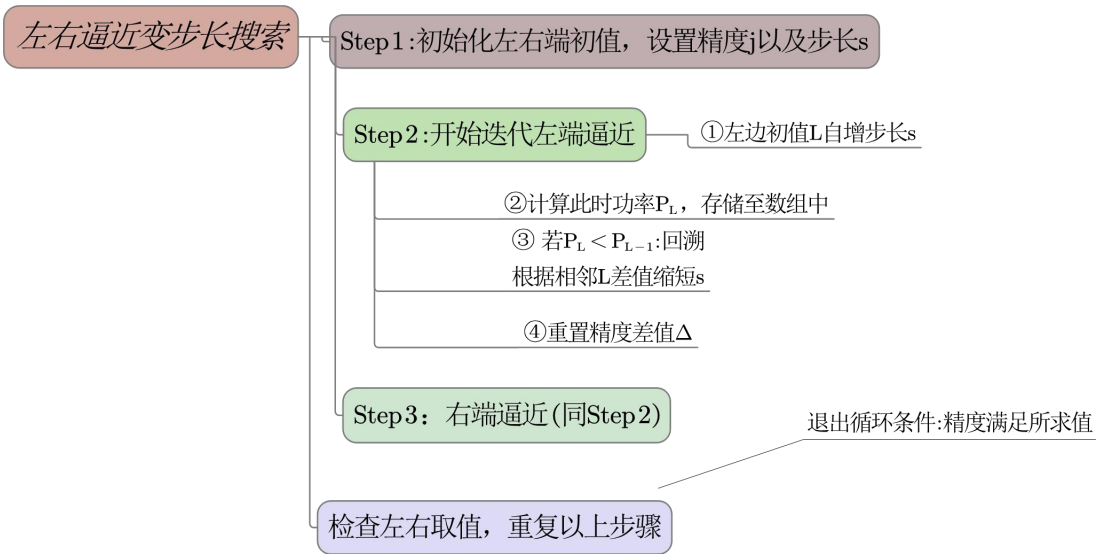


图 13 左右逼近变步长搜索算法框图

5.2.3 情况二：双变量优化

按题目要求，该情况阻尼系数与浮子和振子的相对速度的绝对值的幂成正比，比例系数在区间 [0,100000] 内取值，幂指数在区间 [0,1] 内取值。

建立以下模型：

决策变量：比例系数 K，幂指数 P

目标函数：PTO 平均输出功率

$$\max \mathrm{P}(\boldsymbol{K}, \boldsymbol{P})$$

$$\boldsymbol{s.t.}\ \boldsymbol{K} \in [\boldsymbol{0}, \boldsymbol{100000}]$$

$$\boldsymbol{P} \in [\boldsymbol{0}, \boldsymbol{1}]$$

我们首先在区间 K ∈ [0,100000]以及 P ∈ [0,1]均匀选取 30*30 个点，绘制出如下三维图，发现该图像类似与马鞍面，我们绘制出了其热力图

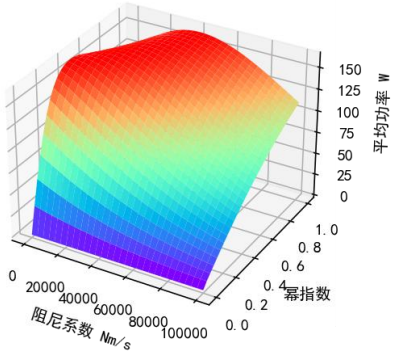


图 14

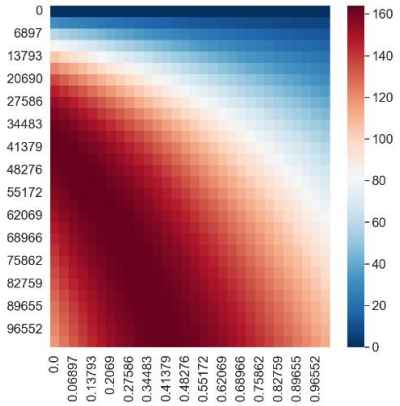


图 15

针对这两个决策变量，我们采用**自适应变步长网格搜索法来**寻找最优参数组合
具体算法步骤如下：

STEP1： *#设置初值*

初始化差值：delta;

定义阈值：ThreshHold;

设置搜索密度：Den=10

设置参数：pl2=[];l1=0

创建空列表用于存储数组

STEP2:*#开始搜索*

当精度大于所需阈值时：

计算对应的平均功率

存储计算的功率

存储功率对应的序号

计算局部最大值

计算局部最大值的序号

if delta<0:

变化搜索密度

重置精度

重新计算搜索网格范围

更新搜索范围

STEP3：

输出搜索结果

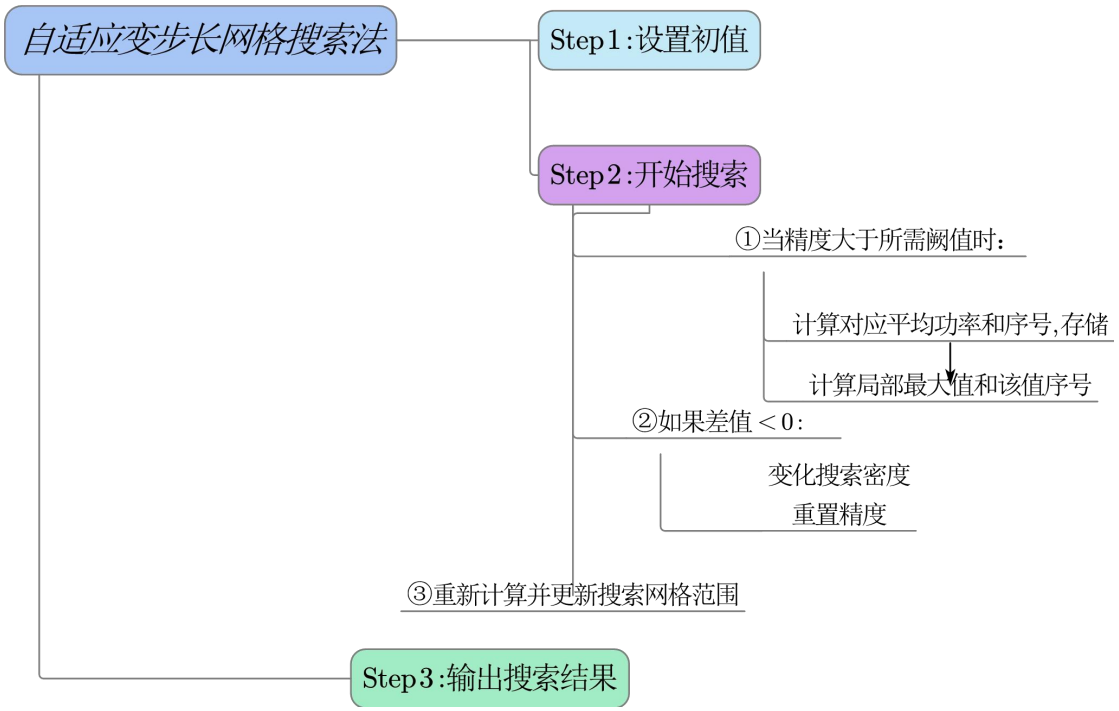


图 16 自适应变步长搜索算法框图

得出当阻尼系数取值$K = 40355.97$,幂指数取值$P = 0$时，在时长为 20s 内取得最大平均功率$P = 163.83W$。

5.3 问题三模型的建立与分析

5.3.1 问题分析

本问为一个两自由度的水动力学浮体方程，除了竖直方向的垂荡，还需考虑水平方向的纵摇。本节中，我们采用了一种基于简单线性变换的交替迭代法来求解。对于本问中较为复杂的运动状况，我们建立了二维随动坐标系，将多维问题降维，并进行问题的求解。

为了简化问题，我们建立了两个坐标系：**以水平面和重力方向为轴的二维世界坐标系，和始终以世界截面内浮子圆锥部分和圆柱部分交接线以及组合体中轴线为轴的二维随动坐标系**。在每一次迭代中，对于系统外力，我们仅考虑其在随动坐标系下沿 y 轴方向的分量，并仅计算在该方向上的物理量的值。在进行下一次迭代前，我们将随动坐标系下的各物理分量还原为世界坐标系下的数值并对世界坐标系下的数据进行更新。依次重复这个过程，我们便可以以较为简便的方法来进行问题的求解。

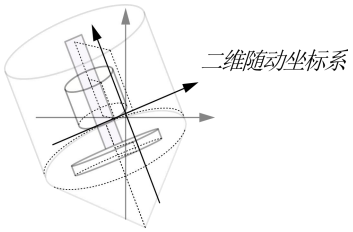


图 17

质心计算公式为：

$$M_{总体} \cdot z_{总体} = \Sigma_{i=1}^{n} m_i \cdot z_i$$

由于浮子质量均匀，我们假设浮子密度为ρ_f,以底部圆锥尖端为原点，竖直向上为z轴建立坐标系，将浮子拆分为上部圆盘，中部圆筒，下部圆锥壳分别计算质心再计算浮子等效质心。假设圆盘半径为R,圆筒与圆锥高度分别为H, h,圆锥母线长为l

圆盘：

$$\begin{cases} m_1 = \rho_f \pi R^2 \\ z_1 = H + h \end{cases}$$

圆筒：

$$\begin{cases} m_2 = 2\pi RH\rho \\ z_2 = h + \dfrac{H}{2} \end{cases}$$

圆锥：

$$\begin{cases} m_3 = \rho \pi R l \\ z_3 = \dfrac{\int_0^R 2\,\pi r^2 \rho \cdot \dfrac{hl}{R^2}}{m_3} \end{cases}$$

解得圆锥质心为：

$$z_3 = \frac{2}{3}h$$

联立解得:

$$Oz_{浮} = m_1 z_1 + m_2 z_2 + m_3 z_3$$

解得质心相对于底部高度约为:$O = 2.2$

下面计算浮子各部分相对质心的转动惯量,其中$d_{圆锥}, d_{圆柱}, d_{圆盘}$分别是各部分质心相对于质心的距离，我们利用平行轴定理求得：

圆锥：

$$I_{圆锥} = \iiint_V (\rho r^2 dV) + m_{圆锥} d_{圆锥}^2$$

圆柱：

$$I_{圆柱}=\frac{m_{圆柱}(6R^2+H^2)}{12}+m_{圆柱}d^2_{圆柱}$$

圆盘：

$$I_{圆盘}=\frac{mR^2}{4}+m_{圆盘}d圆盘^2$$

故浮子相对于质心的转动惯量为：

$$I=I_{圆锥}+I_{圆柱}+I_{圆盘}$$

计算得浮子的转动惯量约为 $\mathrm{I}=8421\ kg\cdot m^2$

5.3.2 考虑旋转角的随动坐标系组合体受力-加速度方程的建立

在本节中，我们考虑浮子，振子的偏移角度，建立包含偏移角的受力-加速度方程。**我们设$\boldsymbol{\theta}$为浮子在世界坐标系下偏移 y 轴的角度，$\boldsymbol{\alpha}$为振子在随动坐标系下偏移 y 轴的角度，即振子对浮子的相对角位移。同时，为简便计算，本节中建立的方程组的正方向为随动坐标系下 y 轴正方向。**

设$\boldsymbol{S_1(t),S_2(t)}$为在随动坐标系下浮子，振子的位移。**此处的两个位移-时间函数仅为一个存储历史迭代结果的虚拟物理量，其作用为作为计算世界坐标系下垂荡方向位移的过渡，而并无实际物理意义。**每一次迭代开始时，$\boldsymbol{S_1(t),S_2(t)}$均会被在当前世界坐标系下随动坐标系 y 轴方向的精确位移更新。具体流程将在下文给出，本节仅考虑运动方程的建立。

在上文中，我们有：

$$\begin{cases}(\boldsymbol{M+m_e})\ddot{S_1}+\boldsymbol{m}\ddot{S_2}=-(\boldsymbol{M+m})g+\boldsymbol{F_f}+\boldsymbol{F_L}-\boldsymbol{sign}(\dot{S_1})\boldsymbol{F_{CZ}}\\ \boldsymbol{m}\ddot{S_2}=-\boldsymbol{mg}+\boldsymbol{F_t}-\boldsymbol{sign}(\dot{S_2}-\dot{S_1})\boldsymbol{F_Z}\end{cases}$$

考虑到第三问情景下，弹簧-阻尼-振子套杆可以自由旋转，偏离组合体中轴线。故此问需要使用**隔离法**分别对浮子振子进行分析。

在振子相对中轴线偏离$\boldsymbol{\alpha}$时，直线弹簧和直线阻尼对振子的合力为：

$$\boldsymbol{F_t-sign(\dot{S_2}-\dot{S_1})F_Z}$$

其在中轴线方向的分量为：

$$\boldsymbol{(F_t-sign(\dot{S_2}-\dot{S_1})F_Z)cos\alpha}$$

其他浮子受到的竖直方向的力在该方向的分力为：

$$\boldsymbol{(-Mg+F_f+F_L-sign(\dot{S_1})F_{CZ})sin\,(\theta)}$$

故对浮子，我们有：

$$\boldsymbol{(M+m_e)\ddot{S_1}=\left(-Mg+F_f+F_L-sign(\dot{S_1})F_{CZ}\right)sin\,(\theta)+(F_t-sign(\dot{S_2}-\dot{S_1})F_Z)cos\alpha}$$

同理，对振子，我们有：

$$\boldsymbol{m}\ddot{S_2}=-\boldsymbol{mgcos(\alpha+\theta)}+\boldsymbol{F_t}-\boldsymbol{sign}(\dot{S_2}-\frac{\dot{S_1}}{\boldsymbol{cos\alpha}})\boldsymbol{F_Z}$$

其中$\frac{\dot{S_1}}{\boldsymbol{cos\alpha}}$为$\boldsymbol{\dot{S_1}}$在弹簧-振子线上的分量。

故在随动坐标系下 y 轴方向的动力学方程可表示为：

$$\begin{cases}\boldsymbol{(M+m_e)\ddot{S_1}=\left(-Mg+F_f+F_L+sign(\dot{S_1})F_{CZ}\right)sin(\theta)+(F_t-sign(\dot{S_2}-\dot{S_1})F_Z)cos\alpha}\\ \boldsymbol{m}\ddot{S_2}=-\boldsymbol{mgcos(\alpha+\theta)}+\boldsymbol{F_t}-\boldsymbol{sign}(\dot{S_2}-\frac{\dot{S_1}}{\boldsymbol{cos\alpha}})\boldsymbol{F_Z}\end{cases}$$

5.3.3 力矩-角加速度方程的建立

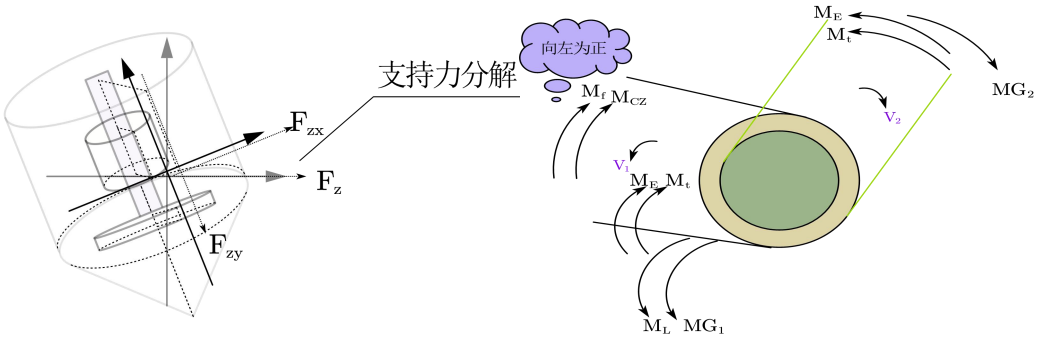


图 18　　**图 19**

本节中，我们将建立描述组合体纵摇方向的动力学方程。由于在不同转轴下角速度，角加速度的矢量叠加难以计算，我们选择旋转弹簧于振子-弹簧杆的铰接点作为浮子的旋转点，同时忽略其到圆锥-圆柱交接平面的距离。因此，我们忽略由直线弹簧和阻尼对浮子产生的反作用力平行于这个平面的分量产生的力矩。同时，尽管在纵摇运动中已有外来力矩对组合体作用，我们依然会考虑由（1）问中所提及的产生垂荡作用的力所产生的力矩。

分别设$\omega_1, \omega_2, \beta_1, \beta_2, r_{1,}, r_2$为浮子，振子的角速度,角加速度和旋转中心。**令使组合体逆时针旋转的方向为正方向**

首先对振子分析，我们有：

$$\boldsymbol{i\beta_2 = mgsin(\alpha + \theta)r_2 - 2sign(\omega_2 - \omega_1)k_{MZ}(\omega_2 - \omega_1) - 2k_t\alpha}$$

k_{MZ} 为旋转阻尼系数，k_t为旋转弹簧系数，$\boldsymbol{i}$ 为振子的转动惯量。振子可以看作一个实心圆柱，其旋转点在扭转弹簧的铰接处，且弹簧长度处于变化中。当旋转轴为振子对称轴时，其转动惯量可表示为：

$$\boldsymbol{i_0 = \frac{1}{12}m(3r^2{}_z + h^2{}_z) = 202.75}$$

故由平行轴定理可得：

$$\boldsymbol{i = 202.75 + mr_2{}^2}$$

旋转阻尼的阻尼力矩始终处于阻碍振子运动的方向。同时，旋转弹簧和旋转阻尼会反作用于浮子产生反力矩。这个力矩的有效分量为：

$$\boldsymbol{-(2sign(\omega_2 - \omega_1)k_{MZ}(\omega_2 - \omega_1) - 2k_t\alpha)sin\,(\alpha)}$$

浮子重力所产生的力矩为：

$$\boldsymbol{Mgsin(\theta)(O - h_{co})}$$

$\boldsymbol{O}$ 为浮子重心到组合体最下端的垂直距离。对于其他水动力相关的受力，我们设其作用于浮心。为简便起见，我们设浮心到组合体最下端垂直距离为组合体浸没高度的一半，即$\frac{\boldsymbol{h}}{\boldsymbol{2}}$。

故对于上文所提及的波浪相关力，其对浮子所产生的力矩为：

$$\boldsymbol{\left(F_f + F_L - sign(\dot{S}_1)F_{CZ}\right)sin\,(\theta)\left|\frac{h}{2} - h_{co}\right|}$$

上式中各力的求得须使用世界坐标系下浮子的垂直速度。同时依题意可得，浮子还受纵摇激励力矩，静水恢复力矩，纵摇阻尼力矩等。联立上式，浮子的力矩-角加速度方程可表示为 ：

$$(I+I_e)\beta_1 = -(2sign(\omega_2-\omega_1)k_{MZ}(\omega_2-\omega_1)-2k_t\alpha)sin(\alpha)+Mgsin(\theta)(O-h_{co})$$
$$+\left(F_f+F_L+sign(\dot{S_1})F_{CZ}\right)sin(\theta)\left|\frac{h}{2}-h_{co}\right|+M_Z+M_H+M_{ZZ}$$

其中 I 为浮子转动惯量，I_e为附加转动惯量

由相对运动性质，我们得到：

$$\begin{cases}\omega_2-\omega_1=\dot{\alpha}\\ \beta_2=\ddot{\alpha}+\ddot{\theta}\end{cases}$$

联立上文各式，我们得到了考虑纵摇情况下的组合体动力学方程：

$$\begin{cases}(M+m_e)\ddot{S_1}=\left(-Mg+F_f+F_L+sign(\dot{S_1})F_{CZ}\right)sin(\theta)+(F_t-sign(\dot{S_2}-\dot{S_1})F_Z)cos\alpha\\ m\ddot{S_2}=-mgcos(\ +\theta)+F_t+sign(\dot{S_2}-\frac{\dot{S_1}}{cos\alpha})F_Z\\ i(\ddot{\alpha}+\ddot{\theta})=mgsin(\alpha+\theta)r_2+2sign(\dot{\alpha})k_{MZ}(\dot{\alpha})-2k_t\alpha\\ (I+I_e)\ddot{\theta}=-(2sign(\dot{\alpha})k_{MZ}(\dot{\alpha})-2k_t\alpha)sin(\alpha)+Mgsin(\theta)(O-h_{co})+\\ \left(F_f+F_L+sign(\dot{S_1})F_{CZ}\right)sin(\theta)\left|\frac{h}{2}-h_{co}\right|+M_Z+M_H+M_{ZZ}\\ r_2=0.25+L_t\end{cases}$$

L_t为 t 时刻直线弹簧的长度。

各波浪力和弹力的计算方法上文已有详细说明，故这里不再赘述。

5.3.4 求解微分方程

同样地，我们使用龙格库塔算法的思想来求解常微分方程组

组合体受力-加速度方程初始条件的确立

由以上受力分析可知，在初始静水条件中组合体处于受力平衡状态。其力学方程组为

$$\begin{cases}mg=k_t|L-L(0)|\\ (m+M)g=V(h(0)\\ L(0)\le L\end{cases}$$

联立解得：

$$\begin{cases}L(0)=0.201\\ h(0)=2.8\end{cases}$$

故该微分方程组起始条件可表示为:

$$\begin{cases}L(0)=0.201\\ h(0)=2.8\\ \ddot{S_1}(0)=0\\ \ddot{S_2}(0)=0\\ \dot{S_1}(0)=0\\ \dot{S_2}(0)=0\\ S_1(0)=0\\ S_2(0)=0\end{cases}$$

同样的，力矩-角加速度初始条件为：

$$\begin{cases}\ddot{\boldsymbol{\theta}}(\mathbf{0})=\mathbf{0}\\ \ddot{\boldsymbol{\alpha}}(\mathbf{0})=\mathbf{0}\\ \dot{\boldsymbol{\theta}}(\mathbf{0})=\mathbf{0}\\ \dot{\boldsymbol{\alpha}}(\mathbf{0})=\mathbf{0}\\ \boldsymbol{\theta}(\mathbf{0})=\mathbf{0}\\ \boldsymbol{\alpha}(\mathbf{0})=\mathbf{0}\end{cases}$$

5.3.4.2 离散化求解

上文已对离散化方法有具体说明，这里不再赘述。结合力矩-角速度方程，我们可以得到考虑纵摇情况下的组合体动力学方程，将其离散化得到：

$$\begin{cases}\boldsymbol{S_1(t+1)=S_1(t)+\dot{S_1}(t)\Delta t+\frac{1}{2}\ddot{S_1}(t)\Delta t^2+\frac{1}{6}\frac{\dddot{S_1}(t)-\ddot{S_1}(t-1)}{\Delta t}\Delta t^3}\\ \boldsymbol{S_2(t+1)=S_2(t)+\dot{S_1}(t)\Delta t+\frac{1}{2}\ddot{S_1}(t)\Delta t^2+\frac{1}{6}\frac{\dddot{S_1}(t)-\ddot{S_1}(t-1)}{\Delta t}\Delta t^3}\\ \boldsymbol{\dot{S_1}(t+1)=\dot{S_1}(t)+\ddot{S_1}(t)\Delta t+\frac{\ddot{S_1}(t)-\ddot{S_1}(t-1)}{2\Delta t}\Delta t^2}\\ \boldsymbol{\dot{S_2}(t+1)=\dot{S_2}(t)+\ddot{S_2}(t)\Delta t+\frac{\ddot{S_2}(t)-\ddot{S_2}(t-1)}{2\Delta t}\Delta t^2}\\ \boldsymbol{\alpha(t+1)=\alpha(t)+\dot{\alpha}(t)\Delta t+\frac{1}{2}\ddot{\alpha}(t)\Delta t^2+\frac{1}{6}\frac{\ddot{\alpha}(t)-\ddot{\alpha}(t-1)}{\Delta t}\Delta t^3}\\ \boldsymbol{\theta(t+1)=\theta(t)+\dot{\theta}(t)\Delta t+\frac{1}{2}\ddot{\theta}(t)\Delta t^2+\frac{1}{6}\frac{\ddot{\theta}(t)-\ddot{\theta}(t-1)}{\Delta t}\Delta t^3}\\ \boldsymbol{(M+m_e)\ddot{S_1}=\left(-Mg+F_f+F_L+sign(\dot{S_1})F_{CZ}\right)sin(\theta)+(F_t-sign(\dot{S_2}-\dot{S_1})F_Z)cos\alpha}\\ \boldsymbol{m\ddot{S_2}=-mgcos(\alpha+\theta)+F_t-sign(\dot{S_2}-\frac{\dot{S_1}}{cos\alpha})F_Z}\\ \boldsymbol{i(\ddot{\alpha}+\ddot{\theta})=mgsin(\alpha+\theta)r_2+2sign(\dot{\alpha})k_{MZ}(\dot{\alpha})-2k_t\alpha}\\ \boldsymbol{(I+I_e)\ddot{\theta}=-(2sign(\dot{\alpha})k_{MZ}(\dot{\alpha})-2k_t\alpha)sin(\alpha)+Mgsin(\theta)(O-h_{co})+}\\ \boldsymbol{\left(F_f+F_L+sign(\dot{S_1})F_{CZ}\right)sin(\theta)\left|\frac{h}{2}-0.8\right|+M_Z+M_H+M_{ZZ}}\\ \boldsymbol{F_t(t)=k_t\left(\dot{S_2}(t)-\dot{S_1}(t)\right)}\\ \boldsymbol{F_f(t)=\rho gV(h(t))}\\ \boldsymbol{F_{L(t)}=fcos(\omega t)}\\ \boldsymbol{F_Z(t)=k\left(\dot{S_2}(t)-\dot{S_1}(t)\right)}\\ \boldsymbol{h(t)=h(0)+S_2(t)}\\ \boldsymbol{F_{CZ}(t)=k_{CZ}\dot{S_1}(t)}\\ \boldsymbol{r_2=0.25+L_t}\end{cases}$$

5.3.4.3 迭代求解

本节中，我们设计了一种基于简单变换的交替迭代法以高效地求解上式。算法流程如下所示：

STEP1：算法初始化。建立瞬时状态储存多维数组 P。设置时间步长 Δt，总时长 T。t=0。记虚拟物理量 t 时刻的 S(t)值为$P[S]_t$，分别记 t 时刻世界坐标系竖直方向位移，速度为 $P[\overline{S}]_t, P[\overline{v}]_t$。其他物理量存储方法如上，并以微分方程组初始

状态赋值

STEP2：更新$P[S]_t = P[\bar{S}]_t cos(\mathrm{P}[\theta]_{t-1}), P[v]_t = P[\overline{v}]_t cos(\mathrm{P}[\theta]_{t-1})$

STEP3：以上述微分方程差分形式，根据$\mathbf{P}_t$和$\mathbf{P}_{t-1}$的数据对各物理量依次差分，得到$\mathbf{P}_{t+1}$。

STEP4：更新$P[\bar{S}]_t = \frac{P[S]_t}{cos(P[\theta]_t)}, P[\overline{v}]_t = \frac{P[v]_t}{cos(P[\theta]_t)}, t = t + \Delta t$

STEP5： 若 t<T，则返回 STEP2。反之结束算法，输出 P。

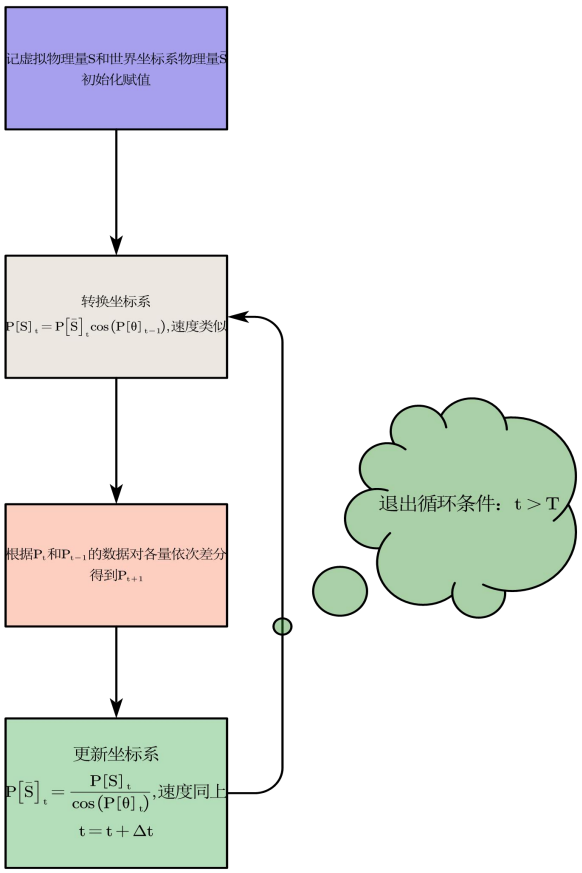


图 20 交替迭代法算法框图

5.3.4.4 问题三求解结果

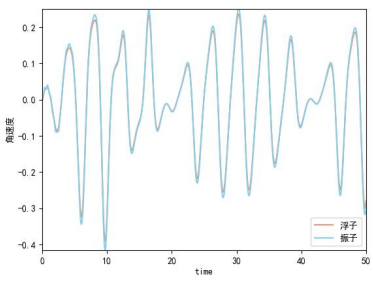


图 21 浮子/振子 角速度-时间关系图

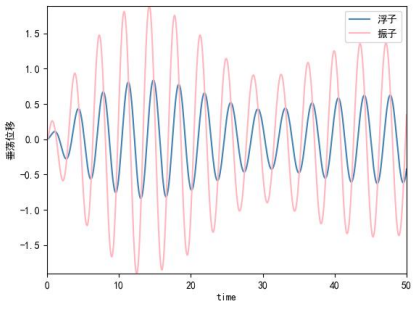


图 22 浮子/振子 垂荡位移-时间关系图

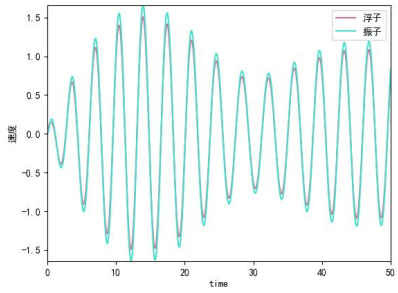


图 23 浮子/振子 速度-时间关系图

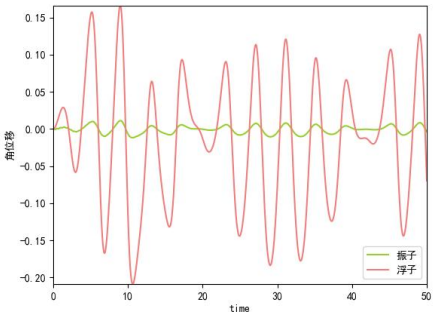


图 24 浮子/振子 角位移-时间关系图

指定时间下各物理量如下表所示：

时间 (s)	浮子				振子			
	垂荡位移 (m)	垂荡速度 (m/s)	纵摇角位移	纵摇角速度 (s^{-1})	垂荡位移 (m)	垂荡速度 (m/s)	纵摇角位移	纵摇角速度 (s^{-1})
10	-0.52214	0.99918	-0.11612	-0.31312	0.48847	1.07378	-0.12282	-0.33066
20	-0.70996	-0.23228	-0.01307	-0.03132	-1.06377	-0.27781	-0.01351	-0.03302
40	0.37348	0.76969	0.02422	-0.06902	1.2448	0.85764	0.0251	-0.07341
60	-0.32014	-0.70032	-0.06902	0.12811	-1.13141	-0.77667	-0.07303	0.13635
100	-0.05407	-0.95301	-0.12075	0.16326	-1.11374	-1.042	-0.12803	0.17308

表 4 浮子/振子运动状态表

垂荡速度和位移均取竖直向上为正方向，角速度角位移均取逆时针方向（向左）为正方向。

5.3.5 算法评价：

优点：简便高效，以龙格库塔算法的基本思想用差分求解的方式在保证精确度的前提下大大提高求解速度。

缺点：求解结果对Δt的选取敏感性较高，对于较大的Δt容易出现误差累积从而导致数值爆炸现象的出现。

5.4 问题四模型的建立与求解

基于问题二建立的双变量优化模型，根据题意有：

决策变量:直线阻尼器和旋转阻尼器最优阻尼系数

目标函数:PTO 平均输出功率

$$\max \mathrm{P}(K, K_C)$$

$$s.t.\ K \in [0,100000]$$

$$\mathrm{K_C} \in [0,100000]$$

5.2.1 阻尼器输出功率

力矩 M 的瞬时功率为：

$$\boldsymbol{P = M\omega}$$

故对题给旋转阻尼，其瞬时输出功率为：

$$P=\frac{1}{2}k\omega^2$$

故在时间 T 内，平均功率可表示为：

$$\overline{P}=\frac{\int_0^T\frac{1}{2}k\omega^2dt}{T}$$

和上文类似，对$\omega(t)$函数进行泰勒展开逼近，并在每一次迭代内对其进行差分计算，平均功率的数值解可表示为：

$$\begin{cases}\overline{P}=\dfrac{\sum P_t\Delta t}{T}\\ P_{t+1}=M(\omega_t+\dfrac{1}{2}\dot{\omega}_t\Delta t^2+\dfrac{1}{6}\dfrac{\omega_t-\omega_{t-1}}{\Delta t}\Delta t^3)\end{cases}$$

联合上文提及式子，第四问平均功率数值积分计算公式为：

$$\begin{cases}\overline{P}=\dfrac{\sum P_t\Delta t}{T}\\ P(t+1)=\dfrac{1}{2}\Delta tK(\dot{S}(t)+\ddot{S}(t)\Delta t+\dfrac{\ddot{S}(t)-\ddot{S}(t-1)}{2\Delta t}\Delta t^2)^2+2M(\omega_t+\dfrac{1}{2}\dot{\omega}_t\Delta t^2+\dfrac{1}{6}\dfrac{\omega_t-\omega_{t-1}}{\Delta t}\Delta t^3)\end{cases}$$

其中ω_t为浮子振子在 t 时刻的相对角速度的绝对值，即$|\dot{\alpha}(t)|$。

因本问求解模型与第二问类似，故可直接套用第二问提出的**自适应变步长网格搜索算法**求解。**解得直线阻尼系数 39837，旋转阻尼系数 100000 为参数时，功率取得最大值 56.896W。**具体变化情况和热力图如下所示：

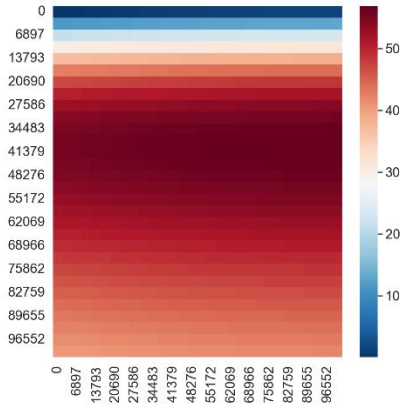


图 24

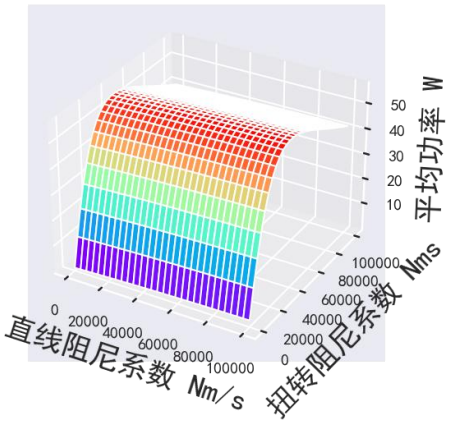


图 25

六、模型评价与推广

模型优点：简便易用，求解迅速，误差传播率较好，可以较为精确地对题给情景进行建模拟真。

模型缺点：时间步长取过大时，因阻尼力的特殊作用方向影响，对高度耦合的物理系统求解所得结果可能出现数值震荡现象，使精确度降低。

考虑到真实的海洋环境波浪条件复杂,且题中理想化，高度周期化的波浪条件在现实中并无存在。真实的波浪环境一般是多个方向不同频率，强度的波浪的叠加。同时，由于海上的气压环境与陆地上气压环境大不相同，导致海上空气流

动往往相比陆地强度更大，故认为真实环境中风力对装置的作用力实质上是不可忽略的。故在本节模型推广部分，我们设计了一种改进型的波浪能利用装置，以充分利用真实海洋环境的波浪能和风能，从而提高输出功率。设计简图如下图所示。

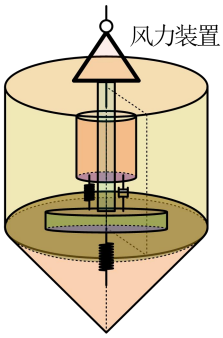


图 26

此装置相比题目所给装置，除上方添加小型风电装置利用海风和部分波浪势能外，还在旋转弹簧装置下方加入一个新的扭转弹簧-旋转阻尼组合装置，以影响上方主装置在横向方向发生的角偏移。在面对不同方向的波浪时，该新扭转弹簧可以使上方主要功率输出转置发生横向偏移。根据上述推导式，振子重力所提供的有效力矩为：

$$\boldsymbol{mgsin(\alpha+\theta)r_2}$$

新增装置可以提高$\boldsymbol{\alpha}$值，从而增大振子对浮子的相对角加速度，最终达到增大纵摇能量利用率的目的。但同时，若新增转置对主装置产生扭转力过小，可能造成主装置难以回正，影响垂荡输出功率，进而影响总输出功率。设振子最小期望偏移角度为γ_1，最大期望偏移角为γ_2，最大期望角速度为ω_{max}故为了保证主输出装置可以回正，该弹簧-阻尼装置所产生的扭矩应至少满足：

$$\boldsymbol{k_t\gamma_2+mgsin(\gamma_2+\theta)L\geq M_{new}\geq mgsin(\gamma_1+\theta)L+k_t\gamma_1+k_{MZ}\omega_{max}}$$

同时除开横摇方向的运动，现实场景的组合体六自由度运动还可以使主装置发生横向偏移，从而实现对各个方向产生的波浪能的利用。

七、参考文献

[1] 郑永红,状态空间模型在海洋波能转换系统中的应用[A],2003

[2] 黄宇明,振荡浮子式波能装置水动力性能及优化控制,2019

[3] 李荣华,刘播,《微分方程数值解法》

[4] 黄舒予,自适应变步长 MPPT 算法，2011

八、相关代码

```
#Problem 1

import math

import matplotlib.pyplot as plt

w=1.4005
```

```
extram=1335.535
VerticalResistance=656.3616
f=6250
mfu=4866
fr=1
fcylinderh=3
fconeh=0.8
mzhen=2433
ro=1025
g=9.8
springk=80000
springl=0.5

deltatime=0.0001

def CalculateInitialHeight():
    V=(mfu+mzhen)/(ro)
    InitialHeight=(V-(1/3)*math.pi*fconeh)/(math.pi)+0.8
    return  InitialHeight

InitialHeight=CalculateInitialHeight()
def CalculateFlowPower(height):
    if height<=0:
        f=0
    elif height<=fconeh:
        f=ro*g*(1/3)*math.pi*math.pow(fconeh/height,2)*height
    elif height<fcylinderh+fconeh:
        f=ro*g*((1/3)*math.pi*fconeh+math.pi*(height-fconeh))
    else:
        f=ro*g*((1/3)*math.pi*fconeh+math.pi*fcylinderh)
    return f



def GetWaveForce(time):
    return f*math.cos(w*time)

def GetReDire(v):
    if(v>=0):
        return -1
    if(v<0):
        return 1
def GetZunik(C,v,p):
    return C*math.pow(math.fabs(v),p)
```

```
def main():
    Zunik=10000
    time=0
    FullTime=10
    Power=0
    p=[]
    x=[]
    LwholeS=[0,0]
    LZhengS=[0,0]
    LSpringS=[springl-mzhen*g/springk]
    LZhengv=[0,0]
    LWholev=[0,0]
    LWholeA=[0,0]
    LZhengA=[0,0]
    Ltime=[0]
    while time<=FullTime:
        wholev = LWholev[-1]
        Zhengv=LZhengv[-1]
        wholeA=LWholeA[-1]
        ZhengA=LZhengA[-1]

        wholeS=LwholeS[-1]
        ZhengS=LZhengS[-1]
        SpringS=LSpringS[-1]
        x.append(wholev-Zhengv)
        Ltime.append(time)
        Zunik=GetZunik(10000,Zhengv-wholev,0.5)
        #整体
        LWholeA.append((-mfu*g+CalculateFlowPower(InitialHeight-wholeS)+GetWaveForce(time)+VerticalResistance*GetReDire(wholev)*math.fabs(wholev)-(springk*(-SpringS+springl)+GetReDire(Zhengv-wholev) * Zunik * math.fabs(Zhengv - wholev)))/(mfu+extram))
        LWholev.append(wholev+wholeA*deltatime+deltatime*(LWholeA[-2]-LWholeA[-3])/2)
        LwholeS.append(wholeS+wholev * deltatime+(1/2)*wholeA*deltatime*deltatime+(1/6)*(LWholeA[-2]-LWholeA[-3])*deltatime*deltatime)
        #振子
        LZhengA.append((springk*(-SpringS+springl)-mzhen*g+GetReDire(Zhengv-wholev) * Zunik * math.fabs(Zhengv - wholev))/(mzhen))
        LZhengv.append(Zhengv+ZhengA*deltatime+deltatime*(LZhengA[-2]-LZhengA[-3])/2)
        LZhengS.append(ZhengS+Zhengv*deltatime+ZhengA*deltatime+(1/2)*ZhengA*deltatime*deltatime)
        #弹簧
        LSpringS.append(SpringS+(Zhengv-wholev)*deltatime+(1/2)*(ZhengA-wholeA)*deltatime*deltatime+(1/6)*(LZhengA[-2]-LZhengA[-3]-LWholeA[-2]+LWholeA[-3])*deltatime*deltatime)
        #功
```

```
        Power+=Zunik * math.fabs(Zhengv - wholev)*math.fabs(Zhengv - wholev)*deltatime/2
        if(time!=0):
          p.append(Power/time)
        time+=deltatime
    plt.plot(LZhengS)
    #print(Power)
    #plt.plot(LSpringS)
    #plt.plot(fu)
    #plt.plot(s)
    #plt.show()
    #plt.plot(p)
    plt.show()
    return Power/time

print(main())

# Problem2

l=[]
for i in range(0,10):
    print(i)
    for j in range(0,10):
        l.append(main(i*10000,j*0.1))
print(l.index(max(l)))
# %%
from winsound import SND_PURGE
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from tqdm import tqdm
import numpy as np

#定义坐标轴
x=np.linspace(0,100000,30)
y = np.linspace(0,1,30)
xx, yy = np.meshgrid(x, y)
z=[]
with tqdm(total=900) as pbar:
    for xxx in x:
        for yyy in y:
            z_tmp=(main(xxx,yyy))
            z.append(z_tmp)
            pbar.update(1)
# %%
delta=100;
ThreshHold=1e-5
```

```
pl2=[]
l1=0
l2=0
r1=100000
r2=1

LocalMaxPList=[0]
power_ezlist=[]
pl3=[]

SRList=[]
Den=10
SRange=math.floor(Den/2)
ZunikList=np.linspace(l1,r1,Den)
exList=np.linspace(l2,r2,Den)
while delta>ThreshHold:
    pl2=[]
    power_ezlist=[]
    with tqdm(total=Den*Den) as pbar:
        z_index=0
        e_index=0
        for Zuniki in ZunikList:
            for ex in exList:
                currentp=(main(Zuniki,ex))
                pl2.append(currentp)
                pl3.append(currentp)
                power_ezlist.append([z_index,e_index])
                pbar.update(1)
                e_index+=1
                e_index=e_index % Den
            z_index+=1
            z_index=z_index % Den

    LocalMaxP=max(pl2)
    LocalMaxPList.append(LocalMaxP)
    delta=LocalMaxPList[-1]-LocalMaxPList[-2]
    LocalMaxP_index=pl2.index(max(pl2))

    def Scale(n,Den):
        if n>=Den-1:
            return Den-1
        elif n<0:
            return 0
        else:
```

```
            return n

    if delta>=0:

        l1=ZunikList[Scale(power_ezlist[LocalMaxP_index][0]-SRange,Den)]
        r1=ZunikList[Scale(power_ezlist[LocalMaxP_index][0]+SRange,Den)]
        l2=exList[Scale(power_ezlist[LocalMaxP_index][1]-SRange,Den)]
        r2=exList[Scale(power_ezlist[LocalMaxP_index][1]+SRange,Den)]
        SRList.append([[l1,r1],[l2,r2]])
    elif delta<0:
        Den*=2
        delta=abs(delta)
        [[l1,r1],[l2,r2]]=SRList[-1]
        SRList.append([[l1,r1],[l2,r2]])
    else:
        break
    maxp_Zunik=ZunikList[power_ezlist[LocalMaxP_index][0]]
    maxp_ez=exList[power_ezlist[LocalMaxP_index][1]]

    SRange=math.floor(Den/4)
    ZunikList=np.linspace(l1,r1,Den)
    exList=np.linspace(l2,r2,Den)
    print(f'current power:{LocalMaxP},ex:{maxp_ez},Zunik={maxp_Zunik}\ncurrent range:Zunik:{l1}~{r1},
ex:{l2}~{r2}')


# %%
temp_list=np.linspace(40355.960750705744,40355.980800790865,1000)
find_max_p=[]
for temp_each in temp_list:
    find_max_p.append(main(temp_each,0))

plt.plot(find_max_p)

# %%
print(temp_list[500],temp_list[800])


# %%
main(40355.970785783386,0)

# %%

#Problem 3
import math
```

```
import matplotlib.pyplot as plt
import csv
w=1.7152
extram=1028.876
Iextra=7001.914
VerticalResistance=683.4558
f=3640
L=1690
mfu=4866
fr=1
fcylinderh=3
fconeh=0.8
mzhen=2433
ro=1025
g=9.8
springk=80000
springl=0.5

deltatime=0.001

def CalculateInitialHeight():
    V=(mfu+mzhen)/(ro)
    InitialHeight=(V-(1/3)*math.pi*fconeh)/(math.pi)+0.8
    return  InitialHeight

InitialHeight=CalculateInitialHeight()
def CalculateFlowPower(height):
    if height<=0:
        f=0
    elif height<=fconeh:
        f=ro*g*(1/3)*math.pi*math.pow(height/fconeh,2)*height
    elif height<fcylinderh+fconeh:
        f=ro*g*((1/3)*math.pi*fconeh+math.pi*(height-fconeh))
    else:
        f=ro*g*((1/3)*math.pi*fconeh+math.pi*fconeh)
    return f


def GetWaveForce(time):
    return f*math.cos(w*time)

def GetReDire(v):
    if(v>=0):
        return -1
```

```
    if(v<0):
        return 1
def GetZunik(C,v,p):
    return C*math.pow(math.fabs(v),p)

def GetIZhen(l):
   return (1/12)*mzhen*(3*math.pow(0.5,2)+math.pow(0.5,2))+mzhen*math.pow(0.25+l,2)

def main():
    tt=0
    Zunik=10000
    Torquek=250000
    torque_zuni=1000
    time=0
    FullTime=40*2*math.pi/w
    Power=0
    p=[]
    LwholeW=[0]
    LzhenW=[0]
    Lalpha=[0]
    Ltheta=[0]
    LwholeS=[0]
    LZhengS=[0]
    LSpringS=[springl-mzhen*g/springk,springl-mzhen*g/springk]
    LZhengv=[0]
    LWholev=[0]
    LWholeA=[0]
    LZhengA=[0]
    fu=[]
    Ltime=[0]
    Power=0
    Lchuiv=[0]
    Lchuis=[0]
    fulla = []
    WB = [0,0]
    ZB = [0,0]
    absoluteAlpha=[0]
    res=[]
    step=200
    while time<=FullTime:
        Zhengv=LZhengv[-1]
        wholeA=LWholeA[-1]
        ZhengA=LZhengA[-1]
```

```
        wholeS=Lchuis[-1]*math.cos(Ltheta[-1])
        ZhengS=LZhengS[-1]
        SpringS=LSpringS[-1]
        wholeW=LwholeW[-1]
        zhenW=LzhenW[-1]
        alpha=Lalpha[-1]
        theta=Ltheta[-1]
        wholev=LWholev[-1]

        wholeBeta=WB[-1]
        zhenBeta=ZB[-1]



        Ltime.append(time)

        chuiv=Lchuiv[-1]
        chuis=Lchuis[-1]

        #整体
        LWholeA.append(((-mfu*g+CalculateFlowPower(InitialHeight-chuis)+GetWaveForce(time)+VerticalRe
sistance*GetReDire(wholev)*math.fabs(wholev))*math.cos(theta)-(springk*(-SpringS+springl)+GetReDire
(Zhengv-wholev) * Zunik * math.fabs(Zhengv - wholev))*math.cos(alpha))/(mfu+extram))
        LWholev.append(wholev+wholeA*deltatime+deltatime*(LWholeA[-1]-LWholeA[-2])/2)
        Lchuiv.append(LWholev[-1]/math.cos(theta))
        LwholeS.append(wholeS+wholev * deltatime+(1/2)*wholeA*deltatime*deltatime+(1/6)*(LWholeA[-1]-
LWholeA[-2])*deltatime*deltatime)
        Lchuis.append(chuis+chuiv*deltatime)

        LZhengA.append((springk*(-SpringS+springl)-mzhen*g*math.cos(alpha+theta)+GetReDire(Zhengv-who
lev/math.cos(alpha)) * Zunik * math.fabs(Zhengv - wholev/math.cos(alpha)))/(mzhen))
        LZhengv.append(Zhengv+ZhengA*deltatime+deltatime*(LZhengA[-1]-LZhengA[-2])/2)
        LZhengS.append(ZhengS+Zhengv*deltatime+ZhengA*deltatime+(1/2)*ZhengA*deltatime*deltatime)

        # 弹簧
        LSpringS.append(SpringS + (Zhengv - wholev) * deltatime + (1 / 2) * (ZhengA - wholeA) * deltat
ime * deltatime + (1 / 6) * (LZhengA[-1] - LZhengA[-2] - LWholeA[-1] + LWholeA[-2]) * deltatime * delt
atime)

        #扭矩
        #whole
        wholeJu=mfu*g*math.sin(theta)*1.1\
        -(VerticalResistance*GetReDire(chuiv)*math.fabs(wholev)+CalculateFlowPower(InitialHeight-chui
s)+GetWaveForce(time))*math.sin(theta)*(math.fabs(InitialHeight-chuis)/2)\
        +L*math.cos(w*time)\
```

```
        -(-Torquek*alpha+GetReDire(zhenW-wholeW)*math.fabs(zhenW-wholeW)*torque_zuni)*math.sin(alpha)

        -654.3383*wholeW\
        -theta*8890.7\

        WB.append(wholeJu/(Iextra+14948))
        LwholeW.append(wholeW+wholeBeta*deltatime+(1/2)*(WB[-2]-WB[-3])*deltatime)
        Ltheta.append(theta+wholeW*deltatime+(1/2)*WB[-2]*math.pow(deltatime,2)+(1/6)*(WB[-2]-WB[-3])
*math.pow(deltatime,2))
        #zhen

        zhenJu=(-Torquek*alpha+GetReDire(zhenW-wholeW)*math.fabs(zhenW-wholeW)*torque_zuni)+mzhen*g*m
ath.sin(theta+alpha)*math.fabs(LSpringS[-1]+0.25)+2*mzhen*alpha*zhenW*(0.25+LSpringS[-1])
        ZB.append(zhenJu/(GetIZhen(math.fabs(LSpringS[-2]+0.25))))
        LzhenW.append(zhenW+zhenBeta*deltatime+(1/2)*(ZB[-2]-ZB[-3])*deltatime)
        Lalpha.append(alpha+(zhenW-wholeW)*deltatime+(1/2)*(WB[-2]-ZB[-2])*math.pow(deltatime,2)+(1/6)
*((ZB[-2]-WB[-2]-(ZB[-3]-WB[-3]))*math.pow(deltatime,2)))

        Power+=torque_zuni*(math.pow(math.fabs(wholeW-zhenW),2))*deltatime/2
        #Power+=Zunik * math.fabs(Zhengv - wholev)*math.fabs(Zhengv - wholev)*deltatime/2
        p.append(1000*(math.fabs(wholeW-zhenW)))
        fulla.append(alpha+theta)
        absoluteAlpha.append(Lalpha[-1]+Ltheta[-1])
        if(step==200):
            step=0
            res.append([tt,chuis,chuiv,theta,wholeW,ZhengS,Zhengv,alpha+theta,zhenW])
            tt += 0.2
        time+= deltatime
        step+=1

    return res

main()

#Problem 4
from winsound import SND_PURGE
from matplotlib import pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from tqdm import tqdm
import numpy as np

#定义坐标轴


x=np.linspace(1,100000,20)
```

```
y = np.linspace(1,100000,20)
xx, yy = np.meshgrid(x, y)
z=[]
with tqdm(total=400) as pbar:
    for xxx in x:
        for yyy in y:
            z_tmp=(main(xxx,yyy))
            z.append(z_tmp)
            pbar.update(1)



# %%
plt.rcParams['font.sans-serif']=['SimHei'] #用来正常显示中文标签
plt.rcParams['axes.unicode_minus']=False #用来正常显示负号
%config InlineBackend.figure_format = 'retina'
%matplotlib inline
fig = plt.figure()
ax = plt.axes(projection='3d')
zz=np.reshape(z,(20,20))
ax.plot_surface(xx, yy, zz, rstride=1, cstride=1, cmap='rainbow')
ax.set_xlabel('直线阻尼系数 Nm/s')
ax.set_ylabel('扭转阻尼系数')
ax.set_zlabel(' 平均功率 W')
plt.show()

# %%
import pandas as pd
data=pd.DataFrame(zz)
zcol=np.linspace(0,100000,20)
zrow=np.linspace(0,100000,20)
zrow=[round(each) for each in zrow ]
zcol=[round(each) for each in zcol ]
data.columns=zcol
data.index=zrow
data

# %%
import seaborn as sns
%matplotlib inline
sns.set(font_scale=1.5)
sns.set_context({"figure.figsize":(8,8)})

sns.heatmap(data=data,cmap="RdBu_r",center=20)
```

```
# %%
delta=100;
ThreshHold=1e-5
pl2=[]
l1=20000
l2=0
r1=50000
r2=100000

LocalMaxPList=[0]
power_ezlist=[]
pl3=[]

SRList=[]
Den=10
SRange=math.floor(Den/4)
ZunikList=np.linspace(l1,r1,Den)
exList=np.linspace(l2,r2,Den)
while delta>ThreshHold:
    pl2=[]
    power_ezlist=[]
    with tqdm(total=Den*Den) as pbar:
        z_index=0
        e_index=0
        for Zuniki in ZunikList:
            for ex in exList:
                currentp=(main(Zuniki,ex))
                pl2.append(currentp)
                pl3.append(currentp)
                power_ezlist.append([z_index,e_index])
                pbar.update(1)
                e_index+=1
                e_index=e_index % Den
            z_index+=1
            z_index=z_index % Den

    LocalMaxP=max(pl2)
    LocalMaxPList.append(LocalMaxP)
    delta=LocalMaxPList[-1]-LocalMaxPList[-2]
    LocalMaxP_index=pl2.index(max(pl2))

    def Scale(n,Den):
```

```
        if n>Den-1:
            return Den-1
        elif n<0:
            return 0
        else:
            return n

    if delta>=0:
        l1=ZunikList[Scale(power_ezlist[LocalMaxP_index][0]-SRange,Den)]
        r1=ZunikList[Scale(power_ezlist[LocalMaxP_index][0]+SRange,Den)]
        l2=exList[Scale(power_ezlist[LocalMaxP_index][1]-SRange,int(Den/4))]
        r2=exList[Scale(power_ezlist[LocalMaxP_index][1]+SRange,int(Den/4))]
        SRList.append([[l1,r1],[l2,r2]])
    elif delta<0:
        Den*=2
        delta=abs(delta)
        [[l1,r1],[l2,r2]]=SRList[-1]
        SRList.append([[l1,r1],[l2,r2]])
    else:
        break
    maxp_Zunik=ZunikList[power_ezlist[LocalMaxP_index][0]]
    maxp_ez=exList[power_ezlist[LocalMaxP_index][1]]

    SRange=math.floor(Den/4)
    ZunikList=np.linspace(l1,r1,Den)
    exList=np.linspace(l2,r2,Den)
    print(f'current power:{LocalMaxP},niu_k:{maxp_ez},Zunik={maxp_Zunik}\ncurrent range:Zunik:{l1}~{r
1},niu_k:{l2}~{r2}')

# %%
plt.plot(pl3)

# %%
temp_list=np.linspace(39837,39838,10)
find_max_p=[]
with tqdm(total=10) as pbar:
    for temp_each in temp_list:
        find_max_p.append(main(temp_each,100000))
        pbar.update(1)

plt.plot(find_max_p)

# %%
print(temp_list[2],temp_list[6])
```

```
# %%
max(find_max_p)


# %%
```