

常见数值微分，数值积分算法简介与应用

摘要

本文基于 Burden^[1]中的内容简要介绍对比了常见的几种传统数值微分，数值积分算法. 进而基于李东风^[2]介绍了基于大数定律的 Monte Carlo 积分方法及其应用，并通过算例比较了其传统的数值积分算法的优缺点。最后给出了数值积分算法在波浪能最大化利用问题中的应用。

关键词：数值微分；数值积分；Monte Carlo 积分；大数定律；波浪能

目录

一 传统数值微积分方法	2
1.1 数值微分方法	2
1.1.1 几种算法的关联	3
1.1.2 几种算法的比较	4
1.1.3 算法的不足与改进	4
1.2 数值积分方法	5
1.2.1 几种算法的关联	5
1.2.2 几种算法的比较	6
1.2.3 算法的不足与改进	6
二 Monte Carlo 积分方法	6
2.1 Monte Carlo 积分方法简介	6
2.1.1 问题化简	7
2.1.2 算法介绍	7
2.2 算法误差分析	9
2.2.1 算法一误差分析	9
2.2.2 算法二误差分析	9
2.2.3 算法一，二的误差对比	9
2.3 与传统数值分析算法的对比	10
2.4 数值实验	10
2.4.1 面积的计算	10
2.4.2 区间上积分的计算	11
三 数值积分方法应用——波浪能装置优化	11
3.1 问题描述	11
3.2 问题的求解	12
3.2.1 运动模型的建立	12

3.2.2	最优化模型的建立	13
3.2.3	模型求解	14
A	代码	16
1.1	数值微分	16
1.2	数值积分	18
1.3	Monte Carlo 积分计算单位圆面积	21
1.4	三种积分误差对比	22
1.5	波浪能装置优化	24
1.5.1	情况 1	25
1.5.2	情况 2	27
B	实验数据	31
2.1	数值微分	31
2.2	数值积分	32
2.3	装置数据	33
C	符号说明	34

一 传统数值微积分方法

1.1 数值微分方法

数值微分方法是用来计算无法求出表达式的函数等不易用传统方法求出导数值的函数的导数的方法。主要包括向前或向后差分的两点法，以及三点法，五点法所对应的端点公式与中点公式。记 f 为区间 $[a, b]$ 上的连续可微函数，我们将其公式陈列于此：

两点向前差分公式 $f'(x_0) \approx \frac{f(x_0+h)-f(x_0)}{h}$

两点向后差分公式 $f'(x_0) \approx \frac{f(x_0)-f(x_0-h)}{h}$

三点端点公式 $f'(x_0) \approx \frac{-3f(x_0)+4f(x_0+h)-f(x_0+2h)}{2h}$

三点中点公式 $f'(x_0) \approx \frac{f(x_0+h)-f(x_0-h)}{2h}$

五点中点公式 $f'(x_0) \approx \frac{f(x_0-2h)-8f(x_0-h)+8f(x_0+h)-f(x_0+2h)}{12h}$

五点端点公式 $f'(x_0) \approx \frac{-25f(x_0)+48f(x_0+h)-36f(x_0+2h)+16f(x_0+3h)-3f(x_0+4h)}{12h}$

1.1.1 几种算法的关联

上面这些公式实际上都来自于给定 $n+1$ 个点 $\{x_0, x_1, \dots, x_n\}$ 的 Lagrange 插值公式

$$f(x) = \sum_{k=0}^n f(x_k) L_k(x) + \frac{(x-x_0) \cdots (x-x_n)}{(n+1)!} f^{(n+1)}(\xi(x))$$

由此有

$$\begin{aligned} f'(x) &= \sum_{k=0}^n f(x_k) L'_k(x) + D_x \left[\frac{(x-x_0) \cdots (x-x_n)}{(n+1)!} \right] f^{(n+1)}(\xi(x)) \\ &\quad + \frac{(x-x_0) \cdots (x-x_n)}{(n+1)!} D_x [f^{(n+1)}(\xi(x))] \end{aligned}$$

取 x 为 $\{x_0, x_1, \dots, x_n\}$ 中一点得到 $(n+1)$ 点公式

$$f(x) = \sum_{k=0}^n f(x_k) L'_k(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!} \prod_{k=0, k \neq j}^n (x_j - x_k)$$

令 n 取 1, 2, 4, 并将被求值点取为不同序号的节点, 即得到了上述的 6 个公式。

理论上, 在上面的 $(n+1)$ 点公式中, 若令 n 增大, 即取更多的节点, 则公式的精确度会严格的增加。但要注意, 随着节点的增加, 公式中函数的求值次数也会增加, 这导致了舍入误差的积累。虽然如此, 随着 n 的上升, 公式精度仍会上升, 但最常用的两种公式仍是三点公式与五点公式。

事实上, 由三点公式出发, 经过 Richardson 外推方法, 可以得到五点公式, 下面举一个例子介绍: 设 f 具有下面所用到的所有阶导数, 且导数连续, 于是有

$$f'(x_0) = \frac{f(x_0+h) - f(x_0-h)}{2h} - \frac{h^2}{6} f'''(x_0) - \frac{h^4}{120} f^{(5)}(x_0) - \dots$$

若记

$$N_1(h) = \frac{f(x_0+h) - f(x_0-h)}{2h}$$

上面的公式变为

$$f'(x_0) = N_1(h) - \frac{h^2}{6} f'''(x_0) - \frac{h^4}{120} f^{(5)}(x_0) - \dots$$

在上式中用 $\frac{h}{2}$ 代替 h 得到

$$f'(x_0) = N_1\left(\frac{h}{2}\right) - \frac{h^2}{24} f'''(x_0) - \frac{h^4}{1920} f^{(5)}(x_0) - \dots$$

利用上面两式消去 h^2 项, 记

$$N_2(h) = \frac{4N_1\left(\frac{h}{2}\right) - N_1(h)}{4-1}$$

则

$$f'(x_0) = N_2(h) + \frac{h^4}{480} f^{(5)}(x_0) - \dots$$

注意到 $N_2(2h)$ 即为五点中点公式。由此也可以知道五点公式是比三点公式更精确的公式。

不仅如此，重复上面的方法，即令

$$N_j(h) = \frac{4^{j-1}N_{j-1}(\frac{h}{2}) - N_{j-1}(h)}{4^{j-1} - 1}, j = 2, 3, \dots$$

可以得到一系列估计 $N_2(h), N_3(h), N_4(h), \dots$ 。由此，重复上述过程，也得到了一个基于 Richardson 外推方法的新的数值微分算法。

1.1.2 几种算法的比较

比较

两点，三点，五点公式均为基于多项式插值法得到的数值微分近似公式。根据上一节中 $(n+1)$ 点公式的推导过程可知，使用的插值节点越多，得到的计算结果越精确，但使用更多的插值节点也会造成计算速度的下降。

数值实验

下面通过使用不同的公式计算一些函数的数值微分与真实导数值的误差，来验证以上的论断。代码见1.1，结果见附录中表2.4，表中每个单元格内数据对应的步长由上至下依次为 $h = 0.01, 0.1, 0.02, 0.2$ 。

由实验结果知，上述论断是正确的。

1.1.3 算法的不足与改进

上文所提到的两点，三点，五点公式在计算时均需要在算式中除以较小的数，这会导致被除数的误差被极大的放大。因此，上文提到的几个公式最终的误差均无法被很好的控制，为不稳定算法。表1.1.3为三点中点公式计算 $f(x) = \sin(x)$ 在 $x = 0.900$ 处的数值微分以及误差所得的数据^[1]，用以说明算法误差并不随差分步长减小而下降

步长	数值微分	误差
0.010	0.62500	0.00339
0.002	0.62250	0.00089
0.005	0.62200	0.00039
0.010	0.62150	-0.00011
0.020	0.62150	-0.00011
0.050	0.62140	-0.00021
0.100	0.62055	-0.00106

以上算法还有一个不足：在实际生产过程中，获取的数据通常带有一定的误差，而数值微分在运算过程中需要除以较小的数，于是数据误差就会被极大的放大，大大影响运算结果。这与上一段中所说的误差产生原因几乎相同。

为解决这一问题，研究者进行了许多探索。其中一个路径为将求数值微分的问题转化为积分方程，进而用泛函分析的手法给出稳定的数值微分算法，参见 ANH^[3]与 ZHAO^[4]。具体数值实验与算法原理在以上两文中均得到了详细的描述，故在此不再赘述。

1.2 数值积分方法

数值积分是用来计算无法求得原函数的函数，无法求得表达式的函数等不便于使用 Newton-Leibniz 公式直接计算的积分表达式的一种方法。常用的数值积分方法有：复化的梯形法，Simpson 方法，Gauss-Legendre 方法以及基于 Richardson 外推的 Romberg 方法。这些算法的具体公式与流程可见 Burden^[1]，为节省篇幅，在此不做赘述。

1.2.1 几种算法的关联

以下记 f 为有充分高的可微性的被积函数， $[a, b]$ 为积分区间。计算

$$\int_a^b f(x)dx$$

梯形法与 Simpson 分别使用两个，三个等间距分布的节点进行 Lagrange 插值，使用得到的多项式的积分来近似被积函数的积分。这两种方法归结于 Newton-Cotes 公式，即使用 n 个等间距分布的节点进行插值，再使用得到的多项式的积分来近似被积函数的积分。而由 Epperson^[5]中的例子，可以得知 Newton-Cotes 公式在一些情况下会因 Runge 现象导致计算误差不随节点数量 n 的上升而减小。因此，为了提高精度的同时回避对高次多项式的使用，可以对梯形法与 Simpson 方法进行复化，得到复化的梯形法与 Simpson 方法，可以证明，这两种方法都是数值稳定且收敛的。

Newton-Cotes 方法基于等间距的节点的差值多项式来计算积分，一个自然的想法是，能否通过选择更好的插值节点来提高公式的精度，答案是肯定的。Gauss-Legendre 公式体现了这一想法：通过一系列的数学计算，选择了 Legendre 多项式的零点在积分区间上的对应点作为插值节点¹。与 Newton-Cotes 公式不同，可以证明，在 Gauss-Legendre 方法中，随着插值节点个数的上升，所得的积分的近似值会收敛到所求的积分，并且这个过程是数值稳定的。

虽然 Gauss-Legendre 方法本身是数值收敛的，但它涉及到 Legendre 多项式的零点的使用，以及高次多项式的计算，这都导致其运算所需时间的延长。于是可以对 Gauss-Legendre 方法进行复化处理，得到复化的 Gauss-Legendre 方法。

将上一节中提到的 Richardson 外推方法运用于复化梯形方法，通过与上一节中几乎相同的推导，即可以得到 Romberg 算法。

¹同样体现这一想法的一类求积公式统称为 Gauss 型求积公式，Gauss-Legendre 求积公式是其中一个例子。其中的求积公式与 Gauss-Legendre 公式具有一定的相似性，在此不作详述。

1.2.2 几种算法的比较

比较

在代数精度 (degree of accuracy) 方面, 复化梯形法的插值节点数为 2, 复化 Simpson 法的插值节点数为 3, 故其代数精度分别为 1, 3。对于 $n+1$ 个节点的复化 Gauss-Legendre 公式, 代数精度为 $2n+1$ 。

计算精度方面, 由于 Romberg 算法经过了多次外推, 其计算结果比其他几个算法都更加精确。并且。复化的 Simpson 方法精度要高于复化梯形方法; 在 n 较小时, 随着 n 的上升, n 节点的复化 Gauss-Legendre 方法的精度也应该上升。

数值实验

下面通过不同使用不同步长的几种算法计算积分 $I = \int_1^{100} xe^{-x^3} dx$ 来对上上述论断进行验证。代码见附录1.2, 结果如附录表2.5, 2.6, 2.7, 2.8。

由实验结果知, 上述论断是正确的。

1.2.3 算法的不足与改进

在以上几节中, 本文已经列举了数种数值积分方法, 他们都是收敛且数值稳定的。但他们都只适用于具有一定可微性或连续性的函数。然而在数学中, 常常需要计算如 $1_{\mathbb{Q}}$ 这样没有连续性的函数的积分²。另一方面, 在计算高维空间上的积分时, 积分区域常常是不规则的。在区域比较规则的情况下, 尚可以设法将区域上的积分转化为重积分再利用以上方法进行数值计算, 但操作也较为复杂; 当积分区域变得极为不规则时, 这样的方法就变得无力。

在第二节中, 将介绍基于概率论中大数定律的 Monte Carlo 积分法, 这一方法可以化解上述问题。

二 Monte Carlo 积分方法

2.1 Monte Carlo 积分方法简介

Monte Carlo 积分方法是随机模拟方法在数值积分问题中的一个应用, 是一种利用计算机大量生成的随机数来求解数值积分的方法。下面先给出要解决的问题:

问题 1. D 为 \mathbb{R}^n 中的一个可测子集, f 为 D 上的 Lebesgue 可积函数, 求数值积分

$$I = \int_D f(x) dx$$

$$^2 1_{\mathbb{Q}} \text{ 定义为 } 1_{\mathbb{Q}}(x) = \begin{cases} 1, & x \in \mathbb{Q} \\ 0, & x \notin \mathbb{Q} \end{cases}.$$

2.1.1 问题化简

当 D 为无界区域时, 通常可以将积分区域进行拆分, 再通过一定合适的几何变换 $\phi: \tilde{D} \rightarrow D$ 将 D 上的积分

$$I = \int_D f(x) dx$$

变为有界集 \tilde{D} 上的积分

$$I = \int_{\tilde{D}} f(\phi(\tilde{x})) \det(J(\phi(\tilde{x}))) d\tilde{x}$$

其中 $J(\phi(\tilde{x}))$ 为 Jacobi 矩阵。于是只要考虑 D 为有界集的情况。

进一步, 设 $D \subseteq [a_1, b_1] \times \cdots \times [a_n, b_n] = D'$, 则

$$I = \int_D f(x) dx = \int_{D'} f(x) 1_D(x) dx$$

于是可设 D 具有 $[a_1, b_1] \times \cdots \times [a_n, b_n]$ 的形式。

2.1.2 算法介绍

下面给出两种算法:

算法一: 记 f 为 $D = [a_1, b_1] \times \cdots \times [a_n, b_n]$ 上的非负有界可积函数, M 为它的一个上界, 再记

$$C = \{(x_1, x_2, \cdots, x_n, y) : (x_1, x_2, \dots, x_n) \in D, 0 \leq y \leq f(x_1, x_2, \dots, x_n)\}$$

$$F = \{(x_1, x_2, \cdots, x_n, y) : (x_1, x_2, \dots, x_n) \in D, 0 \leq y \leq M\}$$

若 U_1, U_2, \cdots 为一列服从 F 上的均匀分布的 i.i.d., 令 $\{\xi_i\}$ 为与 $\{U_i\}$ 定义域相同的随机变量, 且满足

$$\xi_i = \begin{cases} 1, & U_i \in C \\ 0, & U_i \notin C \end{cases}$$

则 $\{\xi_i\}$ 为一列服从 $b(1, p)$ 的 i.i.d., 其中

$$p = \frac{I}{V(F)} = \frac{I}{M \prod_{j=1}^n (b_j - a_j)}$$

则由强大数定律

$$p_N = \frac{\sum_{i=1}^N \xi_i}{N} \rightarrow p, a.s. (N \rightarrow \infty)$$

于是

$$I_N = p_N M \prod_{j=1}^n (b_j - a_j) \rightarrow I, a.s. (N \rightarrow \infty)$$

为 I 的一个估计值。

由此得到以下算法

输入: \mathbb{R}^n 中的有界可测集 $D = [a_1, b_1] \times \cdots \times [a_n, b_n]$; D 上的非负可测函数 f ; 生成随机数个数 N

输出: 积分值 $I = \int_D f(x) dx$

Step 1 生成 N 个服从 D 上均匀分布的 i.i.d. 随机向量 U_1, U_2, \dots, U_N

Step 2 对于 $i = 1, 2, \dots, N$, 令

$$\xi_i = \begin{cases} 1, & U_i \in C \\ 0, & U_i \notin F \setminus C \end{cases}$$

Step 3 令

$$I = \frac{\sum_{i=1}^N \xi_i}{N} M \prod_{j=1}^n (b_j - a_j)$$

Step 4 输出 I

算法二: 记 f 为 $D = [a_1, b_1] \times \dots \times [a_n, b_n]$ 上的可积函数, 令 $\{U_i\}$ 为一列服从 D 上均匀分布的 i.i.d., 再令

$$\xi_i = f(U_i)$$

则 ξ_i 为一列可积的 i.i.d., 其期望为

$$\begin{aligned} \mu &= \mathbb{E}(\xi_i) \\ &= \mathbb{E}(f(U_i)) \\ &= \int_D f(x) \frac{1}{\prod_{j=1}^n (b_j - a_j)} dx \\ &= \frac{I}{\prod_{j=1}^n (b_j - a_j)} \end{aligned}$$

由强大数定律

$$\frac{\sum_{i=1}^N \xi_i}{N} \rightarrow \mu, a.s. (N \rightarrow \infty)$$

于是

$$I_N = \frac{\sum_{i=1}^N \xi_i}{N} \prod_{j=1}^n (b_j - a_j)$$

为 I 的一个估计值。

由此得到以下算法

输入: \mathbb{R}^n 中的有界可测集 $D = [a_1, b_1] \times \dots \times [a_n, b_n]$; D 上的可积函数 f ; 生成随机数个数 N

输出: 积分值 $I = \int_D f(x) dx$

Step 1 生成 N 个服从 D 上均匀分布的 i.i.d. 随机向量 U_1, U_2, \dots, U_N

Step 2 对于 $i = 1, 2, \dots, N$, 令

$$\xi_i = f(U_i)$$

Step 3 令

$$I = \frac{\sum_{i=1}^N \xi_i}{N} \prod_{j=1}^n (b_j - a_j)$$

Step 4 输出 I

2.2 算法误差分析

2.2.1 算法一误差分析

沿用上一节算法一中的记号，由中心极限定理，有

$$\frac{(p_N - p)/N}{\sqrt{p(1-p)/N}} \xrightarrow{d} N(0, 1)$$

于是

$$\sqrt{N}(I_N - I) \xrightarrow{d} N(0, (M \prod_{j=1}^n (b_j - a_j))^2 p(1-p))$$

故 I_N 的渐近方差为

$$\frac{(M \prod_{j=1}^n (b_j - a_j))^2 p(1-p)}{N}$$

因此， I_N 的误差以大约 95.45% 的概率不超过

$$\frac{M \prod_{j=1}^n (b_j - a_j) \sqrt{p(1-p)}}{\sqrt{N}}$$

即其随机误差为 $O_p(\frac{1}{\sqrt{N}})$ 。

2.2.2 算法二误差分析

沿用上文算法二中的记号，由 $I_N \rightarrow I, a.s.$ ，知

$$\text{Var}(I_N) = \frac{\text{Var}(f(U_1))}{N} (\prod_{j=1}^n (b_j - a_j))^2$$

同上知算法二的随机误差也为 $O_p(\frac{1}{\sqrt{N}})$

2.2.3 算法一，二的误差对比

假设 f 为 $D = [a_1, b_1] \times \cdots \times [a_n, b_n]$ 上的有界非负可测函数，下面证明算法二的随机误差不超过算法一的随机误差。记 N 个随机数下，算法一的估计为 $I_N^{(1)}$ ，算法二的估计为 $I_N^{(2)}$ ，即证

$$\text{Var}(I_N^{(2)}) \leq \text{Var}(I_N^{(1)})$$

由齐次性，不妨设 $0 \leq f \leq 1$ 。令 $\{X_i\}$ 为一列服从 D 上的均匀分布的 i.i.d.， $\{Y_i\}$ 为一列服从 $[0, 1]$ 上一致分布的 i.i.d. 且 $\{X_i\}, \{Y_i\}$ 独立。再令

$$\eta_i = f(X_i), i = 1, 2, \dots$$

$$\xi_i = \begin{cases} 1, & Y_i \leq f(X_i) \\ 0, & Y_i > f(x_i) \end{cases}, i = 1, 2, \dots$$

则

$$I_N^{(1)} = \prod_{j=1}^n (b_j - a_j) \frac{\sum_{i=1}^N \xi_i}{N}$$

$$I_N^{(2)} = \prod_{j=1}^n (b_j - a_j) \frac{\sum_{i=1}^N \eta_i}{N}$$

于是

$$\text{Var}(I_N^{(1)}) - \text{Var}(I_N^{(2)}) = \frac{\prod_{j=1}^n (b_j - a_j)}{N} \int_D f(x) - f^2(x) dx \geq 0$$

得证。

2.3 与传统数值分析算法的对比

由上一节的计算可知，对于 \mathbb{R} 上的具有较强可微性的被积函数，Monte Carlo 积分法不仅的收敛速度远远慢于传统的数值积分算法，而且运算量也远远大于传统方法。但 Monte Carlo 积分方法的优点在于，当函数的可微性与连续性或积分区域的形状变的很差，传统数值积分方法就无法求出积分的值，而 Monte Carlo 算法仍然适用，并保持 $O_p(\frac{1}{\sqrt{N}})$ 的误差量级。另一方面，当函数定义域的维数上升时，传统数值积分方法的运算量大大提升，导致所需时间指数级的增加，而 Monte Carlo 方法所需的计算时间对随维数升高而上升的速度仅仅为线性级，这也让它的使用范围变得更广泛。

Monte Carlo 积分法可以有效的探寻一些来自物理学或工程学中性质完全未知的函数的积分性质，因此虽然收敛速度不快，它仍然是一个十分重要的数值积分方法。

2.4 数值实验

2.4.1 面积的计算

作为 Monte Carlo 方法的实践，本节使用 Monte Carlo 积分法来计算单位圆盘的面积³，记 \mathbb{D} 为 \mathbb{R}^2 中的单位圆盘，问题即计算积分

$$I = \int_{\mathbb{D}} 1 dx$$

本问题中，上述两种 Monte Carlo 积分法是相同的，使用 Matlab 编写计算程序，代码见附录1.3。计算结果见表2.1，可知 $\lg(\text{随机点个数})=n$ 时，计算误差约为 0.45841 的 $\frac{1}{\sqrt{10^{n-1}}}$ 倍，与预期相符。

³本问题中使用的代码经过简单的修改即可计算两圆相交而得的区域的面积，圆与其它图形（如正方形，三角形）相交而得的区域面积，也可以计算高维空间中几何体的体积。本节选用圆盘为例是因为单位圆盘的面积 π 的精确值较容易获取。

表 2.1: 计算结果及误差

数据类型 \ lg(随机点个数)	1	2	3	4	5	6	7	8
计算结果	3.6	3.16	3.18	3.1272	3.1362	3.14164	3.14119	3.14145
计算误差	0.45841	0.01841	0.03841	-0.01439	-0.00539	0.00005	-0.00041	-0.00014

2.4.2 区间上积分的计算

本节使用两种 Monte Carlo 积分法与复化 Simpson 积分法计算积分

$$I = \int_0^{\pi} \sin(x^2) dx$$

进而对比计算效率。代码见附录1.4，计算误差的结果见表2.2,2.3。

由表知，结果完全符合上一节中的计算，即 Monte Carlo 算法精度明显小于传统数值积分算法，并且 Monte Carlo 算法中的算法二的计算误差要小于算法一的计算误差。

表 2.2: Monte Carlo 算法误差

算法类型 \ lg(数据点个数)	1	2	3	4	5	6	7	8
算法一	0.089732	-0.00027	-0.02427	-0.00037	0.003642	-0.00032	2.97E-05	6.59E-05
算法二	-0.00307	0.006256	0.004205	0.001101	-3.4E-05	-0.00015	2.4E-06	-4.8E-06

表 2.3: 复化 Simpson 算法误差

算法类型 \ lg(积分区间数)	1	2	3	4	5
复化 Simpson	-8.1E-06	-8E-10	-8E-14	-2.8E-16	6.66E-16

三 数值积分方法应用——波浪能装置优化

本问题来自 2022 年全国大学生数学建模竞赛 A 题，其中使用复化梯形法数值积分对波浪能发电装置的平均功率进行计算。

3.1 问题描述

随着经济和社会的发展，人类面临能源需求和环境污染的双重挑战，发展可再生能源产业已成为世界各国的共识。波浪能作为一种重要的海洋可再生能源，分布广泛，储

量丰富，具有可观的应用前景。波浪能装置的能量转换效率是波浪能规模化利用的关键问题之一。

图3.1为一种波浪能装置示意图，由浮子、振子、中轴以及能量输出系统（PTO，包括弹簧和阻尼器）构成，其中振子、中轴及 PTO 被密封在浮子内部；浮子由质量均匀分布的圆柱壳体和圆锥壳体组成；两壳体连接部分有一个隔层，作为安装中轴的支撑面；振子是穿在中轴上的圆柱体，通过 PTO 系统与中轴底座连接。在波浪的作用下，浮子运动并带动振子运动，通过两者的相对运动驱动阻尼器做功，并将所做的功作为能量输出。考虑海水是无粘及无旋的，浮子在线性周期微幅波作用下会受到波浪激励力（矩）、附加惯性力（矩）、兴波阻尼力（矩）和静水恢复力（矩）。在分析下面问题时，忽略中轴、底座、隔层及 PTO 的质量和各种摩擦。

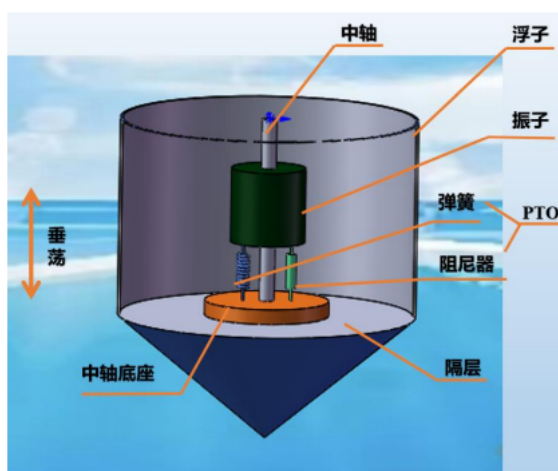


图 3.1: 波浪能装置示意图

考虑浮子在波浪中只做垂荡运动，分别对以下两种情况建立确定直线阻尼器的最优阻尼系数的数学模型，使得 PTO 系统的平均输出功率最大：

情况 1 阻尼系数为常量，阻尼系数在区间 $[0, 100000]$ 内取值；

情况 2 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。

利用附录表3.11和附录表2.10提供的参数值（波浪频率取 2.2143s^{-1} ）分别计算两种情况的最大输出功率及相应的最优阻尼系数。

3.2 问题的求解

本节中使用的符号意义见附录C。

3.2.1 运动模型的建立

以海平面与中轴的交点为坐标原点，竖直向上为正方向建立一维坐标系。

由于波浪能装置中的各种摩擦可以忽略不计，故振子的受力情况较为简单，只受重力、弹簧弹力和直线阻尼器的阻尼力作用。其中弹簧弹力与位移成正比，阻尼力与浮子和振子相对速度成正比。于是由牛顿第二定律可以写出振子的运动方程：

$$\lambda_1(v_2 - v_1) - m_1g + k(l_0 - (x_1 - x_2 - h_2)) = m_1a_1 \quad (3.1)$$

再对浮子在竖直方向上进行受力分析，浮子在竖直方向上受重力、浮力（重力和浮力的综合影响合称为静水恢复力）、波浪激励力、兴波阻尼力、附加惯性力、弹簧弹力和直线阻尼器的阻尼力作用。其中由阿基米德浮力定律知 $F_{\text{浮}} = \rho g V_{\text{排}}$ ，波浪激励力大小为 $f \cos \omega t$ ，兴波阻尼力与浮体速度成正比，附加惯性力为附加质量所受惯性力。于是可以同样地由牛顿第二定律写出浮子的运动方程：

$$-\lambda_1(v_2 - v_1) - m_2g - k(l_0 - (x_1 - x_2 - h_2)) + f \cos \omega t + \rho g V(x_2) - v_2\lambda_2 = (m_0 + m_2)a_2 \quad (3.2)$$

其中 V 为波浪能装置排开水的体积，是一个关于 x_2 的分段函数：

$$V(x) = \begin{cases} 0 & x \geq 0 \\ \frac{1}{3}\pi R^2 h_2 (-x)^3 & -h_2 \leq x < 0 \\ \frac{1}{3}\pi R^2 h_2 + \pi R^2 (-h_2 - x) & -(h_1 + h_2) \leq x < -h_2 \\ \frac{1}{3}\pi R^2 h_2 + \pi R^2 h_1 & x < -(h_1 + h_2) \end{cases} \quad (3.3)$$

结合初始时浮子和振子在静水中的平衡状态，经计算知波浪能装置的锥体部分完全浸没在海水中，由浮力定律和弹簧弹性定律给出 x_1 与 x_2 的初值 $x_1(0), x_2(0)$ 如下：

$$\begin{cases} x_1(0) = -(\frac{(m_1+m_2)g}{\rho g} - \frac{1}{3}\pi R^2 h_2)/\pi R^2 + l_0 - \frac{m_1g}{k} \\ x_2(0) = -(\frac{(m_1+m_2)g}{\rho g} - \frac{1}{3}\pi R^2 h_2)/\pi R^2 - h_2 \end{cases} \quad (3.4)$$

综合方程 (3.1,3.2,3.3,3.4) 建立关于浮子与振子运动的微分方程模型：

$$\begin{cases} \lambda_1(\dot{x}_2 - \dot{x}_1) - m_1g + k(l_0 - (x_1 - x_2 - h_2)) = m_1\ddot{x}_1 \\ -\lambda_1(\dot{x}_2 - \dot{x}_1) - m_2g - k(l_0 - (x_1 - x_2 - h_2)) + f \cos \omega t + \rho g V(x_2) - \dot{x}_2\lambda_2 = (m_0 + m_2)\ddot{x}_2 \\ x_1(0) = -(\frac{(m_1+m_2)g}{\rho g} - \frac{1}{3}\pi R^2 h_2)/\pi R^2 + l_0 - \frac{m_1g}{k} \\ x_2(0) = -(\frac{(m_1+m_2)g}{\rho g} - \frac{1}{3}\pi R^2 h_2)/\pi R^2 - h_2 \\ \dot{x}_1(0) = \dot{x}_2(0) = 0 \end{cases} \quad (3.5)$$

其中 V 关于 x_2 的函数由 (3.3) 给出。在情况 1 中， $\lambda_1 = 10000Ns/m$ ；在情况 2 中， $\lambda_1 = 10000 * |v_2 - v_1|^{0.5}Ns/m$ 。

3.2.2 最优化模型的建立

由于阻尼系数并非已知量，要求在问题给定的直线阻尼器的阻尼系数变化范围的限制下，求解 PTO 系统的最大平均输出功率，并给出相应最优的阻尼系数 λ_1 ，因此需求解一个最优化的问题。

由于浮子与振子的相对运动驱动阻尼器做功，PTO 系统将所做的功转化为能量输出，因此可以根据方程 $P = Fv$ 解出浮子和振子进行垂荡运动时相对阻尼器做功的功率，即 PTO 系统的输出功率。其中，由于阻尼器的质量忽略不计，因此浮子和振子对阻尼器施加的力与直线阻尼器的阻尼力大小相等，与浮子和振子的相对速度成正比，比例系数为 λ_1 ， v 为浮子与振子的相对速度。代入方程并对时间积分得到 PTO 系统的平均输出功率的函数表达式：

$$P = \frac{1}{b-a} \int_a^b \lambda_1 (v_1 - v_2)^2 dt \quad (3.6)$$

其中 $[b,a]$ 为计算平均功率的时间取值区间，当 a 与 b 取值较大时，平均功率趋于稳定，为了确保平均输出功率的稳定性，本文的积分区间定为 $[T,2T]$ ，其中 T 为 40 个波浪周期的时间。

由此，可建立如下单目标优化模型：

决策变量：在情况 1 中为阻尼系数 λ_1 ；在情况 2 中由于阻尼系数与浮子和振子的相对速度的绝对值的幂成正比，故决策变量为比例系数 β 和幂指数 α

目标函数：目标为 PTO 系统的平均输出功率最大，即

$$\max P(\lambda_1) = \frac{1}{T} \int_T^{2T} \lambda_1 (v_1 - v_2)^2 dt \quad (3.7)$$

约束条件：情况 1 中要求阻尼系数在区间 $[0,100000]$ 内，故约束条件为：

$$0 \leq \lambda_1 \leq 100000$$

情况 2 要求比例系数在区间 $[0,100000]$ 内，幂指数在 $[0,1]$ 内，故约束条件为：

$$\begin{cases} 0 \leq \beta \leq 100000 \\ 0 \leq \alpha \leq 1 \end{cases}$$

3.2.3 模型求解

本节代码见附录1.5。

利用四阶五级 Runge-Kutta 方法求解上述运动模型，得 v_1, v_2 随时间变化的函数；**利用复化梯形法，与方程3.6计算装置的平均功率**；最后使用 Matlab 的遗传算法工具求解最优化问题得：

情况 1 直线阻尼器的阻尼系数为 36614.141Ns/m 时，平均输出功率取到最大值 239.0895W。

情况 2 比例系数和幂指数取 $\beta = 92557.039, \alpha = 0.382$ 时，平均输出功率取到最大值 239.2856W，对应的最大阻尼系数为 $92557.039|v_2 - v_1|^{0.382}$ 。

可以作出情况 1 中平均输出功率相对阻尼系数的函数图像如图3.2；情况 2 中平均输出功率相对比例系数 β 和幂指数 α 的函数图像如图3.3：

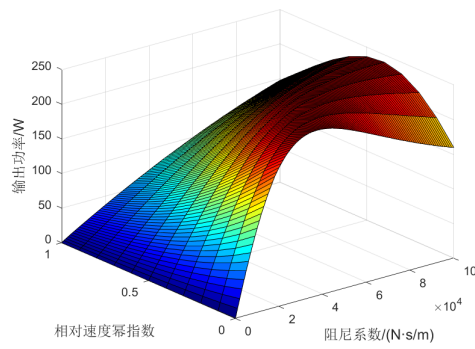
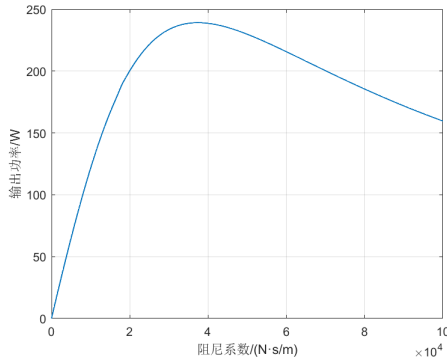


图 3.2: 平均输出功率随阻尼系数变化 图 3.3: 平均输出功率随比例系数与幂指数变化图

参考文献

- [1] BURDEN A, BURDEN R, FAIRES J. Numerical Analysis, 10th ed.[M]. 2016. DOI: [10.13140/2.1.4830.2406](https://doi.org/10.13140/2.1.4830.2406).
- [2] 李东风. 统计计算, 随机模拟积分[Z]. https://www.math.pku.edu.cn/teachers/lidf/docs/statcomp/html/_statcompbook/sim-ranquad.html, 最后编辑于 2022-09-19. 2022.
- [3] AHN S, CHOI U J, RAMM A G. A scheme for stable numerical differentiation [J/OL]. Journal of Computational and Applied Mathematics, 2006, 186(2): 325-334. <https://www.sciencedirect.com/science/article/pii/S0377042705000610>. DOI: <https://doi.org/10.1016/j.cam.2005.02.002>.
- [4] ZHAO Z, MENG Z, HE G. A new approach to numerical differentiation[J/OL]. Journal of Computational and Applied Mathematics, 2009, 232(2): 227-239. <https://www.sciencedirect.com/science/article/pii/S037704270900346X>. DOI: <https://doi.org/10.1016/j.cam.2009.06.001>.
- [5] EPPERSON J F. On the Runge Example[J/OL]. The American Mathematical Monthly, 1987, 94(4): 329-341. eprint: <https://doi.org/10.1080/00029890.1987.12000642>. <https://doi.org/10.1080/00029890.1987.12000642>. DOI: [10.1080/00029890.1987.12000642](https://doi.org/10.1080/00029890.1987.12000642).

附录

A 代码

1.1 数值微分

数值微分函数程序

```
1 function ndif=ndif(f,x,h,n,m)
2 switch n
3     case 1
4         switch m
5             case 1
6                 ndif=(f(x+h)-f(x))./h;%two points forward
7             case 2
8                 ndif=(f(x)-f(x-h))./h;%two points backward
9         end
10    case 2
11        switch m
12            case 1
13                ndif=(-3.*f(x)+4.*f(x+h)-f(x+2.*h))./(2.*h);
14            %three points endpoint
15            case 2
16                ndif=(f(x+h)-f(x-h))./(2.*h);
17            %three points midpoint
18        end
19    case 3
20        switch m
21            case 1
22                ndif=(-25.*f(x)+48.*f(x+h)-36.*f(x+2.*h)...
23                +16.*f(x+3.*h)-3.*f(x+4.*h))./(12.*h);%five point endpoint
24            case 2
25                ndif=(f(x-2.*h)-8.*f(x-h)+8.*f(x+h)...
26                -f(x+2.*h))./(12.*h);%five points midpoint
27        end
28 end
29 end
```


数值微分及误差的计算

```
1 clear
2 clc
3 h=[0.01,0.1,0.02,0.2];%
4 x=1;
5 a=zeros(8,25);
6 e=zeros(8,24);
7 for n=1:8
8     a(n,25)=df(x,n);%exact value
9     for k=1:3
10         for m=1:2
11             for t=1:4
12                 a(n,4*(2*(k-1)+m-1)+t)=...
13 ndif(@(x)f(x,n),x,h(t),k,m);%numerical value
14                 e(n,4*(2*(k-1)+m-1)+t)=...
15 a(n,4*(2*(k-1)+m-1)+t)-a(n,25);%error
16             end
17         end
18     end
19 end
```

被求值函数及其导函数

```
1 function f=f(x,n)
2 switch n
3     case 1
4         f=1./(1+x.^2);
5     case 2
6         f=sin(x);
7     case 3
8         f=tan(x);
9     case 4
10        f=x.*log(1+x);
11    case 5
12        f=x.*exp(x);
13    case 6
14        f=exp(x).*sin(x);
```

```

15     case 7
16         f=(x.^2+1).^0.5;
17     case 8
18         f=sinh(x);
19 end
20 end

```

```

1 function df=df(x,n)
2 switch n
3     case 1
4         df=-2.*x./(1+x.^2).^2;
5     case 2
6         df=cos(x);
7     case 3
8         df=1./(cos(x)).^2;
9     case 4
10        df=log(1+x)+x./(1+x);
11    case 5
12        df=(x+1).*exp(x);
13    case 6
14        df=exp(x).*(sin(x)+cos(x));
15    case 7
16        df=x.*(x.^2+1).^(-0.5);
17    case 8
18        df=cosh(x);
19 end
20 end

```

1.2 数值积分

被积函数

```

1 function f1=f1(x)
2 f1=x.*exp(-x.^3);
3 end

```

复化 Simpson 方法

```

1  function CS=CS(f,a,b,n)%composite Simpson's method
2  %f is the function & a,b is the interval
3  %(b-a)/n is the step
4  h=(b-a)/n;
5  S0=f(a)+f(b);
6  S1=0;
7  S2=0;
8  for i=1:n-1
9      X=a+i.*h;
10     if mod(i,2)==0
11         S2=S2+f(X);
12     else
13         S1=S1+f(X);
14     end
15 end
16 CS=h.*(S0+2.*S2+4.*S1)/3;
17 end

```

Romberg 方法

```

1  function Rbg=Rbg(f,a,b,n)%Romberg's method
2  %f is the function & a,b is the interval
3  %(b-a)/n is the step
4  h=b-a;
5  R(1,1)=h./2.*(f(a)+f(b));
6  for i=2:n
7      R(2,1)=1/2.*(R(1,1)+h.*sum(f(a+(1:2^(i-2))-0.5).*h)));
8      for j=2:i
9          R(2,j)=(4^(j-1).*R(2,j-1)-R(1,j-1))./(4^(j-1)-1);
10     end
11     h=h/2;
12     for j=1:i
13         R(1,j)=R(2,j);
14     end
15 end
16 Rbg=R(1,n);
17 end

```

复化 Gauss-Legendre 方法

```
1 function CGL=CGL(f,a,b,n,m)%composite G-L method
2 %f is the function & a,b is the interval
3 %(b-a)/n is the step & m is number of node
4 switch m
5     case 2%2-node
6         h=(b-a)./n;
7         CGL=h./2.*sum(f(a+((1:n)-1/2).*h-h./(2*sqrt(3)))...
8 +f(a+((1:n)-1/2).*h+h./(2*sqrt(3))));
9     case 3%3-node
10        h=(b-a)./(2*n);
11        CGL=h./9.*sum(5.*f(a+(2.*(1:n)-1).*h...
12 -h.*sqrt(3/5))+5.*f(a+(2.*(1:n)-1).*h...
13 +h.*sqrt(3/5))+8.*f(a+(2.*(1:n)-1).*h));
14 end
15 end
```

数值积分及误差的计算

```
1 clear
2 clc
3 ev=0.1014764825947195668201483098935834921538325145;
4 %exact value
5
6 %calculate the quadrature using these methods
7 %with different step
8 cs(1,1)=CS(@f1,1,100,10);
9 cs(1,2)=CS(@f1,1,100,20);
10 cs(1,3)=CS(@f1,1,100,100);
11 cs(1,4)=CS(@f1,1,100,1000);
12 cs(1,5)=CS(@f1,1,100,10000);
13
14 rbg(1,1)=Rbg(@f1,1,100,5);
15 rbg(1,2)=Rbg(@f1,1,100,10);
16 rbg(1,3)=Rbg(@f1,1,100,15);
17 rbg(1,4)=Rbg(@f1,1,100,20);
18
```

```

19  cgl2(1,1)=CGL(@f1,1,100,10,2);
20  cgl2(1,2)=CGL(@f1,1,100,20,2);
21  cgl2(1,3)=CGL(@f1,1,100,100,2);
22  cgl2(1,4)=CGL(@f1,1,100,1000,2);
23  cgl2(1,5)=CGL(@f1,1,100,10000,2);
24
25  cgl3(1,1)=CGL(@f1,1,100,10,3);
26  cgl3(1,2)=CGL(@f1,1,100,20,3);
27  cgl3(1,3)=CGL(@f1,1,100,100,3);
28  cgl3(1,4)=CGL(@f1,1,100,1000,3);
29  cgl3(1,5)=CGL(@f1,1,100,10000,3);
30
31  eocs=ev-cs;
32  eorbg=ev-rbg;
33  eocgl2=ev-cgl2;
34  eocgl3=ev-cgl3;
35
36  %get the results
37  xlswrite("C:\Users\rixinner\Desktop\数值分析大作业\数值分析 ...
38  Ch_4\Numerical integration\cs.xls",[cs;eocs],'A1:E2')
39  xlswrite("C:\Users\rixinner\Desktop\数值分析大作业\数值分析 ...
40  Ch_4\Numerical integration\rbg.xls",[rbg;eorbg],'A1:D2')
41  xlswrite("C:\Users\rixinner\Desktop\数值分析大作业\数值分析 ...
42  Ch_4\Numerical integration\cgl2.xls",[cgl2;eocgl2],'A1:E2')
43  xlswrite("C:\Users\rixinner\Desktop\数值分析大作业\数值分析 ...
44  Ch_4\Numerical integration\cgl3.xls",[cgl3;eocgl3],'A1:E2')

```

1.3 Monte Carlo 积分计算单位圆面积

```

1  clear
2  clc
3  t=8;
4  area=zeros(t,1);
5  for k=1:t
6      area(k)=area1(10^k);
7  end
8  error=area-pi;

```

```

9  function area=area1(n)
10 % 定义圆心坐标为 (0,0), 半径为 1 的圆
11 x_center = 0;
12 y_center = 0;
13 radius = 1;
14
15 % 设定点的数量
16 num_points = n;
17
18 % 生成随机点
19 x = rand(1, num_points);
20 y = rand(1, num_points);
21
22 % 计算随机点到圆心的距离
23 distance = sqrt((x - x_center).^2 + (y - y_center).^2);
24
25 % 统计在圆内的点的数量
26 num_points_in_circle = sum(distance < radius);
27
28 % 计算圆的面积
29 area = (num_points_in_circle / num_points) * 4;
30 end

```

1.4 三种积分误差对比

算法一计算积分

```

1  function integral = MC1(n)
2  %本函数使用算法一计算  $\sin(x^2)$  在 0 到 1 上的积分
3
4  % 设定 x 的范围和点的数量
5  xmin = 0;
6  xmax = 1;
7  num_points = n;
8
9  % 设定 y 的范围
10 ymin = 0;
11 ymax = 1;

```

```

12
13 % 生成随机点
14 x = rand(1, num_points) * (xmax - xmin) + xmin;
15 y = rand(1, num_points) * (ymax - ymin) + ymin;
16
17 % 计算随机点对应的函数值
18 z = sin(x.^2);
19
20 % 统计图像下方点的个数
21 num_points_under_curve = sum(y < z);
22
23 % 计算积分值
24 integral = (num_points_under_curve / num_points) ...
25     * (xmax - xmin) * (ymax - ymin);

```

算法二计算积分

```

1 function integral=MC2(n)
2 %本函数使用算法二计算 sin(x^2)在0到1上的积分
3
4 % 设定 x 的范围和点的数量
5 xmin = 0;
6 xmax = 1;
7 num_points = n;
8
9 % 生成随机点
10 x = rand(1, num_points) * (xmax - xmin) + xmin;
11
12 % 计算随机点对应的函数值
13 y = sin(x.^2);
14
15 % 计算积分值
16 integral = (xmax - xmin) * (sum(y) / num_points);
17 end

```

计算各方法的误差

```

1 clear
2 clc

```

```

3  t1=8;
4  t2=5;
5  ev=0.3102683017233811018081524231653965075745093888324467;
6  int_mc1=zeros(1,t1);
7  int_mc2=zeros(1,t1);
8  int_cs=zeros(1,t2);
9  for k=1:t1
10     int_mc1(k)=MC1(10^k);
11     int_mc2(k)=MC2(10^k);
12 end
13 for k=1:t2
14     int_cs(k)=CS(@(x) sin(x^2),0,1,10^k);
15 end
16 error_mc1=int_mc1-ev;
17 error_mc2=int_mc2-ev;
18 error_cs=int_cs-ev;

```

1.5 波浪能装置优化

排开水体积的计算

```

1  %本函数对于给定的浮子下端高度坐标，返回浮子排开水的体积
2  function V_a = Va(x)
3
4  %定义本问中用到的常量
5  lmd1=10^4;%阻尼系数
6  lmd2=656.3616;%垂荡兴波阻尼系数
7  m0=1335.535;%垂荡附加质量
8  m1=2433;%振子质量
9  k=80000;%弹簧刚度
10 m2=4866;%浮子质量
11 omg=1.4005;%波浪频率
12 l0=0.5;%弹簧原长
13 rou=1025;%海水密度
14 R=1;%浮子底半径
15 h1=3;%浮子圆柱部分高度
16 h2=0.8;%浮子圆锥部分高度
17 g=9.8;%重力加速度

```



```

18 f=4890;%垂荡激励力振幅
19 x1_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)+l0-m1*g/k;%初值
20 x2_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)-h2;%初值
21
22 if x>=0%装置整个离开水面时
23     V_a=0;
24 else if x>=-h2 && x<0
25     %装置有且仅有圆锥部分与水面相接触
26     V_a=h2*pi*R^2/3*(-x)^3;
27     else if x<-h2 && x>=-(h1+h2)
28     %装置有且仅有圆柱部分与水面相接触
29     V_a=pi*R^2*h2/3+pi*R^2*(-h2-x);
30     else%装置完全沉入水中
31     V_a=pi*R^2*h2/3+pi*R^2*h1;
32     end
33 end
34 end
35 end

```

1.5.1 情况 1

运动方程

```

1 function dy1234 = odefun3(t,y,lmd)
2
3 %定义本问中用到的常量
4 lmd2=167.8395;%垂荡兴波阻尼系数
5 m0=1165.992;%垂荡附加质量
6 m1=2433;%振子质量
7 k=80000;%弹簧刚度
8 m2=4866;%浮子质量
9 omg=2.2143;%波浪频率
10 l0=0.5;%弹簧原长
11 rou=1025;%海水密度
12 R=1;%浮子底半径
13 h1=3;%浮子圆柱部分高度
14 h2=0.8;%浮子圆锥部分高度
15 g=9.8;%重力加速度

```

```

16 f=4890;%垂荡激励力振幅
17 x1_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)+l0-m1*g/k;%初值
18 x2_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)-h2;%初值
19 %lmd为阻尼系数
20
21 %用于使用ode45函数解得x1,x2
22 %其中y(1)=x1,y(2)=x2,y(3)=x1',y(4)=x2'
23 %以海平面为0高度面,x1为振子下端高度
24 %x2为浮子底端高度,竖直向上为正方向
25 dy1234=[y(3);y(4);(lmd*(y(4)-y(3))-g*m1...
26 +k*(l0-(y(1)-y(2)-h2)))/m1;(-lmd*(y(4)-y(3))...
27 -g*m2-k*(l0-(y(1)-y(2)-h2))+f*cos(omg*t)...
28 +rou*g*Va(y(2))-lmd2*y(4))/(m0+m2)];
29 end

```

平均功率的计算

```

1 %本函数对于一个给定的阻尼系数给出对应装置的平均功率的相反数
2 function P = Pa1(lmd)
3 %定义本问中用到的常量
4 lmd2=167.8395;%垂荡兴波阻尼系数
5 m0=1165.992;%垂荡附加质量
6 m1=2433;%振子质量
7 k=80000;%弹簧刚度
8 m2=4866;%浮子质量
9 omg=2.2143;%波浪频率
10 l0=0.5;%弹簧原长
11 rou=1025;%海水密度
12 R=1;%浮子底半径
13 h1=3;%浮子圆柱部分高度
14 h2=0.8;%浮子圆锥部分高度
15 g=9.8;%重力加速度
16 f=4890;%垂荡激励力振幅
17 x1_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)+l0-m1*g/k;%初值
18 x2_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)-h2;%初值
19
20 T=200;%给定一段时长
21 a=[x1_0,x2_0,0,0];%给定初值

```

```

22 %调用ode45函数解方程
23 [t,y]=ode45(@(t,y)odefun3(t,y,lmd),[T:0.01:2*T],a);
24 %由于方程的解在一段时间后稳定下来
25 %可以用[T,2T]时间内的平均功率代表装置的平均功率
26 %为使用MATLAB中工具箱求解最大值
27 %这里将函数定义为目标函数的相反数
28 P=-lmd*trapz(t,abs(y(:,3)-y(:,4)).^2)/T;
29 end

```

主程序

```

1 %对@Pa1调用优化工具箱中的遗传算法
2 %左下角的 Final Point 即为取到最大平均功率的阻尼系数
3 %左侧显示的 Objective function value 的相反数即为所求最大功率

```

作图

```

1 clear%更新变量内容
2
3 t=[0:100:100000];
4 for n=1:1001
5     m(n)=-Pa1(t(n));%接收平均输出功率
6 end
7
8 plot(t,m);%绘制功率分布图
9 xlabel('阻尼系数/(N·s/m)')
10 ylabel('输出功率/W')
11 hline = findobj(gca,'Type','line');
12 set(hline,'LineWidth',0.9);
13 grid on

```

1.5.2 情况 2

```

1 function dy1234 = odefun4(t,y,lmd,alpha)
2
3 %定义本问中用到的常量
4 lmd2=167.8395;%垂荡兴波阻尼系数
5 m0=1165.992;%垂荡附加质量
6 m1=2433;%振子质量

```

```

7 k=80000;%弹簧刚度
8 m2=4866;%浮子质量
9 omg=2.2143;%波浪频率
10 l0=0.5;%弹簧原长
11 rou=1025;%海水密度
12 R=1;%浮子底半径
13 h1=3;%浮子圆柱部分高度
14 h2=0.8;%浮子圆锥部分高度
15 g=9.8;%重力加速度
16 f=4890;%垂荡激励力振幅
17
18 x1_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)+l0-m1*g/k;%初值
19 x2_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)-h2;%初值
20
21 %lmd, alpha 分别为阻尼系数与幂指数
22 %用于使用 ode45 函数解得 x1, x2
23 %其中 y(1)=x1, y(2)=x2, y(3)=x1', y(4)=x2'
24 %以海平面为 0 高度面, x1 为振子下端高度
25 %x2 为浮子底端高度, 竖直向上为正方向
26 dy1234=[y(3);y(4);(lmd*(y(4)-y(3))*(abs(y(4)-y(3)))^alpha...
27 -g*m1+k*(l0-(y(1)-y(2)-h2)))/m1;...
28 (-lmd*(y(4)-y(3))*(abs(y(4)-y(3)))^alpha-g*m2...
29 -k*(l0-(y(1)-y(2)-h2))+f*cos(omg*t)+rou*g*Va(y(2))...
30 -lmd2*y(4))/(m0+m2)];
31 end

```

平均功率的计算

```

1 %本函数对于给定的阻尼系数与幂指数
2 %给出对应装置的平均功率的相反数
3 function P = Pa2(x)
4 lmd=x(1);
5 alpha=x(2);
6
7 %定义本问中用到的常量
8 lmd2=167.8395;%垂荡兴波阻尼系数
9 m0=1165.992;%垂荡附加质量
10 m1=2433;%振子质量

```

```

11 k=80000;%弹簧刚度
12 m2=4866;%浮子质量
13 omg=2.2143;%波浪频率
14 l0=0.5;%弹簧原长
15 rou=1025;%海水密度
16 R=1;%浮子底半径
17 h1=3;%浮子圆柱部分高度
18 h2=0.8;%浮子圆锥部分高度
19 g=9.8;%重力加速度
20 f=4890;%垂荡激励力振幅
21 x1_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)+l0-m1*g/k;%初值
22 x2_0=-((m1+m2)/rou-pi*R^2*h2/3)/(pi*R^2)-h2;%初值
23
24 %以下同Pa1.m中的注释
25 T=200;
26 a=[x1_0,x2_0,0,0];
27 [t,y]=ode45(@(t,y)odefun4(t,y,lmd,alpha),[T:0.01:2*T],a);
28 P=-lmd.*trapz(t,abs(y(:,3)-y(:,4)).^(alpha+2))/T;
29 end

```

主程序

```

1 %对@Pa2调用优化工具箱中的遗传算法
2 %左下角的 Final Point 即为取到最大平均功率的阻尼系数与幂指数
3 %左侧显示的 Objective function value 的相反数即为所求最大功率
4 %具体截面与参数设置见论文

```

作图

```

1 clear%清除以防止变量重叠
2
3 p = 0:1000:100000;
4 q = 0:0.1:1;
5
6 %获取功率分布和取到值点
7 for n=1:101
8     for k=1:11
9         m(n,k)=-Pa2([p(n),q(k)]);
10    end

```

```

11 end
12
13 %阻尼系数、振子浮子相对速度幂指数与输出功率关系图形化
14
15 surface(p,q,m');
16 hline = findobj(gca,'Type','line');
17 set(hline,'LineWidth',0.9)
18 colormap('jet')
19 xlabel('阻尼系数/(N·s/m)', 'FontSize',12)
20 ylabel('相对速度幂指数', 'FontSize',12)
21 zlabel('输出功率/W', 'FontSize',12)
22 view(3)
23 grid on

```

B 实验数据

2.1 数值微分

表 2.4: Error of Methods

value $f(x)$ method	$\frac{1}{1+x^2}$	$\sin(x)$	$\tan(x)$	$x \log(x)$	xe^x	$e^x \sin(x)$	$\sqrt{x^2+1}$	$\sinh(x)$
two-point forward	0.002499876	-0.004216325	0.054311136	0.003741693	0.040956014	0.014659268	0.001758961	0.005901773
	0.024886878	-0.042938553	0.648000505	0.036691806	0.426444322	0.143746615	0.016826342	0.06138213
	0.00499902	-0.008450449	0.110615557	0.007466874	0.082277882	0.029261654	0.003500443	0.011855278
	0.049180328	-0.0874618	1.648200667	0.071861079	0.892728737	0.279568047	0.032075083	0.128220174
two-point backward	-0.002499874	0.004198315	-0.052420457	-0.003758359	-0.040593574	-0.014713848	-0.001776639	-0.005850337
	-0.024861878	0.041138446	-0.45302375	-0.038360351	-0.390173373	-0.149229395	-0.018595205	-0.056235955
	-0.00499898	0.00837841	-0.103045892	-0.007533543	-0.080828089	-0.029480004	-0.003571155	-0.01164953
	-0.048780488	0.080272164	-0.786672983	-0.078557937	-0.747318229	-0.301799494	-0.039163207	-0.107604578
three-point endpoint	7.32687E-07	1.77991E-05	-0.001993286	1.65117E-05	-0.000365855	5.68824E-05	1.74796E-05	-5.17316E-05
	0.000593428	0.001584693	-0.352199656	0.001522533	-0.039840093	0.007925183	0.001577602	-0.005455914
	5.726E-06	7.03474E-05	-0.008422255	6.54373E-05	-0.00147724	0.000236891	6.91324E-05	-0.000208123
	0.003766061	0.005431843	-3.879269823	0.005596709	-0.175474181	0.043084258	0.005628219	-0.023229787
three-point midpoint	1.24999E-09	-9.00499E-06	0.000945339	-8.33343E-06	0.00018122	-2.729E-05	-8.83889E-06	2.57181E-05
	1.24997E-05	-0.000900054	0.097488378	-0.000834272	0.018135475	-0.00274139	-0.000884431	0.002573087
	0.00000002	-3.60194E-05	0.003784833	-3.33348E-05	0.000724897	-0.000109175	-3.53562E-05	0.000102874
	0.00019992	-0.003594818	0.430763842	-0.003348429	0.072705254	-0.011115724	-0.003544062	0.010307798
five-point endpoint	-2.85025E-08	-1.05242E-09	-8.3088E-06	2.16477E-09	-3.32607E-08	3.04377E-08	1.41797E-09	-3.12573E-09
	-0.000166225	-7.852E-06	-0.912352107	1.56362E-05	-0.000396893	0.00033679	1.75922E-05	-3.52829E-05
	-4.32305E-07	-1.63816E-08	-0.00016082	3.33421E-08	-5.42644E-07	4.93168E-07	2.39402E-08	-5.0663E-08
	-0.001331734	-7.48131E-05	-278.0327971	0.000181245	-0.007753463	0.005815231	0.000258821	-0.000653321
five-point midpoint	-0.000000005	-1.80097E-10	-1.15851E-06	3.75025E-10	-5.43667E-09	5.008E-09	2.20886E-10	-5.14369E-10
	-4.99738E-05	-1.79886E-06	-0.013603444	3.77999E-06	-5.4452E-05	5.00545E-05	2.11256E-06	-5.14973E-06
	-0.00000008	-2.88147E-09	-1.86218E-05	6.00191E-09	-8.69906E-08	8.01274E-08	3.52935E-09	-8.23016E-09
	-0.000793323	-2.86792E-05	-0.414536482	6.1965E-05	-0.000875389	0.000799592	2.90634E-05	-8.26904E-05

2.2 数值积分

表 2.5: Composite Simpson's Method

Value or error \ step	$\frac{99}{10}$	$\frac{99}{20}$	$\frac{99}{100}$	$\frac{99}{1000}$	$\frac{99}{10000}$
value	1.214002156	0.607001078	0.122393163	0.101470417	0.101476482
error	-1.112525673	-0.505524595	-0.020916681	6.06587E-06	5.89142E-10

表 2.6: Romberg's Method

Value or error \ step	$\frac{99}{5}$	$\frac{99}{10}$	$\frac{99}{15}$	$\frac{99}{20}$
value	0.694194116	0.101606423	0.101476483	0.101476483
error	-0.592717633	-0.000129941	-1.38778E-17	2.498E-16

表 2.7: Composite Gauss-Legendre Method with 2 Nodes

Value or error \ step	$\frac{99}{10}$	$\frac{99}{20}$	$\frac{99}{100}$	$\frac{99}{1000}$	$\frac{99}{10000}$
value	2.21439E-12	0.000965015	0.105283673	0.10147673	0.101476483
error	0.101476483	0.100511468	-0.003807191	-2.47303E-07	-2.45424E-11

表 2.8: Composite Gauss-Legendre Method with 3 Nodes

Value or error \ step	$\frac{99}{10}$	$\frac{99}{20}$	$\frac{99}{100}$	$\frac{99}{1000}$	$\frac{99}{10000}$
value	0.000448374	0.048843459	0.10142398	0.101476482	0.101476483
error	0.101028109	0.052633024	5.2503E-05	1.28445E-10	1.52656E-16

2.3 装置数据

表 2.9: 装置数据 1

入射波浪频率 (s^{-1})	垂荡附加质量 (kg)	纵摇附加转动惯量 ($\text{kg} \cdot \text{m}^2$)	垂荡兴波阻尼系数 ($\text{N} \cdot \text{s}/\text{m}$)
2.2143	1165.992	7131.29	167.8395
纵摇兴波阻尼系数 ($\text{N} \cdot \text{m} \cdot \text{s}$)	垂荡激励力振幅 (N)	纵摇激励力矩振幅 ($\text{N} \cdot \text{m}$)	
2992.724	4890	2560	

表 2.10: 装置数据 2

参数	取值
浮子质量 (kg)	4866
浮子底半径 (m)	1
浮子圆柱部分高度 (m)	3
浮子圆锥部分高度 (m)	0.8
振子质量 (kg)	2433
振子半径 (m)	0.5
振子高度 (m)	0.5
海水的密度 (kg/m^3)	1025
重力加速度 (m/s^2)	9.8
弹簧刚度 (N/m)	80000
弹簧原长 (m)	0.5
扭转弹簧刚度 ($\text{N} \cdot \text{m}$)	250000
静水恢复力矩系数 ($\text{N} \cdot \text{m}$)	8890.7

C 符号说明

表 3.11: 符号说明

符号	说明	单位
ρ	海水密度	kg/m^3
R	浮子底半径	m
r	振子底半径	m
l	振子高度	m
d	浮子重心到底板的距离	m
dp	初始时刻浮子底端坐标	m
h_1	浮子圆柱部分高度	m
h_2	浮子圆锥部分高度	m
g	重力加速度	m/s^2
x_1	振子底端与中轴相交位置的坐标	m
v_1 或 \dot{x}_1	振子的速度	m/s
x_2	浮子底部尖端的坐标	m
v_2 或 \dot{x}_2	浮子的速度	m/s
a_1 或 \ddot{x}_1	振子的加速度	m/s^2
a_2 或 \ddot{x}_2	浮子的加速度	m/s^2
m_1	振子质量	kg
m_2	浮子质量	kg
m_0	垂荡附加质量	kg
l_0	弹簧原长	m
k	弹簧刚度	N/m
λ_1	直线阻尼器的阻尼系数	Ns/m
λ_2	垂荡兴波阻尼系数	Ns/m
f	波浪激励力振幅	N
ω	波浪频率	rad/s
μ_1	静水恢复力矩系数	Nm