基于受力分析和多目标优化的系泊模型设计

摘要

系泊系统中各个部件的参数对于其抗风浪性和稳定性尤其重要。题目要求分析系泊系统中部件的参数如倾斜角度等，并且根据不同的风速和海水流速设计出合理的系泊模型。本文建立了基于集中质量的多边形近似的受力分析模型和多目标优化模型，采用遗传算法，极限分析，基于pareto粒子群算法为系泊模型的建立提供了解决方案。

第一问要求在海水静止的条件下求出钢桶和各节钢圈的倾斜角度，锚链形状，浮标的吃水深度和游动区域。通过对系统整体和各个部件的进行集中质量的受力分析，可以得到锚链末端与锚的链接处的切线方向与海床的夹角和浮标吃水深度的非解析关系式。将夹角和锚链的拖地数作为2个输入变量，运用遗传算法通过不断地迭代算出最优解。结果显示系泊系统在风速为$12m/s$和风速为$24m/s$的条件下都出现了锚链拖地的现象，说明系泊系统非常稳定，风速在系泊系统的承受范围之内。

第二问首先要求在风速为$36m/s$的条件下系泊系统的参数。运用遗传算法求解，结果显示系统无法维持平衡，会出现锚被拖走的情况。为了使系统恢复静止状态，我们考虑增加重物球的质量。考虑到限制条件为两个夹角的范围，我们采用极限分析的思想，求解整个模型在浮标不完全沉没的条件下恰好保持平衡时的重物球质量，设计算法通过迭代求出重物球质量的上限和下限。

第三问要求建立在风力，海水流力和水深的影响下的合适的系泊模型，要求吃水深度和游动区域及钢桶的倾斜角度尽可能的小。在受力平衡的条件下，我们建立了在静力学条件下的受力分析和不躺底的悬链线模型。系泊系统的设计是一个典型的多目标优化问题，其目标相互冲突和影响，因此我们运用基于pareto熵的粒子群算法，成功的求出了全局最优解，构造了合理的系泊模型。通过对极端条件的计算，进一步说明了模型的抗干扰性强，可行性高。

关键字：集中质量　遗传算法　极限分析　多目标优化
基于pareto的粒子群算法

一. 问题重述

近浅海观测网的传输节点由浮标系统、系泊系统和水声通讯系统组成（如图 1 所示）。某型传输节点的浮标系统可简化为底面直径 2m、高 2m 的圆柱体，浮标的质量为 1000kg。系泊系统由钢管、钢桶、重物球、电焊锚链和特制的抗拖移锚组成。锚的质量为 600kg，锚链选用无档普通链环，近浅海观测网的常用型号及其参数在附表中列出。钢管共 4 节，每节长度 1m，直径为 50mm，每节钢管的质量为 10kg。要求锚链末端与锚的链接处的切线方向与海床的夹角不超过 16 度，否则锚会被拖行，致使节点移位丢失。水声通讯系统安装在一个长 1m、外径 30cm 的密封圆柱形钢桶内，设备和钢桶总质量为 100kg。钢桶上接第 4 节钢管，下接电焊锚链。钢桶竖直时，水声通讯设备的工作效果最佳。若钢桶倾斜，则影响设备的工作效果。钢桶的倾斜角度（钢桶与竖直线的夹角）超过 5 度时，设备的工作效果较差。为了控制钢桶的倾斜角度，钢桶与电焊锚链链接处可悬挂重物球。

问题 1 某型传输节点选用 II 型电焊锚链 22.05m，选用的重物球的质量为 1200kg。现将该型传输节点布放在水深 18m、海床平坦、海水密度为 $1.025\times10^3 kg/m^3$ 的海域。若海水静止，分别计算海面风速为 12m/s 和 24m/s 时钢桶和各节钢管的倾斜角度、锚链形状、浮标的吃水深度和游动区域。

问题 2 在问题 1 的假设下，计算海面风速为 36m/s 时钢桶和各节钢管的倾斜角度、锚链形状和浮标的游动区域。请调节重物球的质量，使得钢桶的倾斜角度不超过 5 度，锚链在锚点与海床的夹角不超过 16 度。

问题 3 由于潮汐等因素的影响，布放海域的实测水深介于 16m~20m 之间。布放点的海水速度最大可达到 1.5m/s、风速最大可达到 36m/s。请

给出考虑风力、水流力和水深情况下的系泊系统设计，分析不同情况下钢桶、钢管的倾斜角度、锚链形状、浮标的吃水深度和游动区域

二．模型的假设

1.风力水平方向，浮标竖直向上。

2.近海处海水流向方向水平。

3.锚链和重力球的体积可以忽略不计。

4.锚链选用的无档普通链环之间不会产生拉伸和摩擦。

5.系泊系统各个部件链接处未焊死，可以转动。

三．符号说明

h	浮标的吃水深度
f	海床对锚的摩擦力
G	系泊系统的总体重量
V	系统的排水体积
P	每个单元的锚链的重量
Q	每个单元的钢管的重量减浮力的差
O	钢桶和重物球的重量减浮力的差
g	重物球的重量
U_w	近海风荷载

F_D	近海水流力
B	支持力

四．问题一模型的建立与求解

4.1 问题一的分析

题目要求出系泊系统中浮标，钢桶和钢管的一系列参数。为此，我们建立了基于集中质量的多边形近似法的受力分析模型，运用遗传算法成功得解决了问题。首先采用集中质量法处理系泊系统，将每段仪器部件单独作为一个节点进行集中处理，原则上保证主要仪器部件所受的张力不失真[1]。锚链在分段处理时，每段锚链的质量等参数信息集中到其几何中心。在上一步的基础上采用多边形近似法，首先计算前一个节点的重力和拉力，合成拉力矢量，顺着拉力矢量的反方向做折线至下一个节点，即近似地求出整个系泊系统的状态[2]。

由于整个系统中部件众多，每个部分单独的受力分析非常复杂。为了避免计算各个部件之间复杂的内力，并且考虑到整个系统在水平方向上只受到两个方向相等且相反的力，故先进行整体的受力分析，得到了浮标的吃水深度h和锚链末端与锚链接处的切线方向与海床的夹角α的第一个关系式。之后对锚链进行分析，由于假设海水是静止的，所以本问中不考虑水运动对锚链的震荡作用，阻力等动力学问题。为了直观并准确的说明锚链的受力状态，本文运用静力学分析，对每个锚链单元进行受力分析，通过每个锚链单元与第一个锚链单元的关系式，并得到第二个$\alpha-h$的关系式。接着对圆管与圆筒进行受力分析。由于圆管的形状，体积与重心等均与锚链类似，所以我们将其近似看成是锚链，用类似的方法得到了圆管的表达式。至此，所有的部件均已分析完毕。

在以上的步骤中我们主要得到了$\alpha-h$的关系，通过对锚链，圆桶,圆管和浮标吃水深度在竖直方向上投影的相加，我们得到了H'。由于变量过多且复杂，无法求出解析解，我们采用数值解法，应用遗传算法迭代寻找最优解，得到的最优解即为本题的答案。

4.2 问题一模型的建立

4.2.1 受力分析

1. 对系统整体进行受力分析

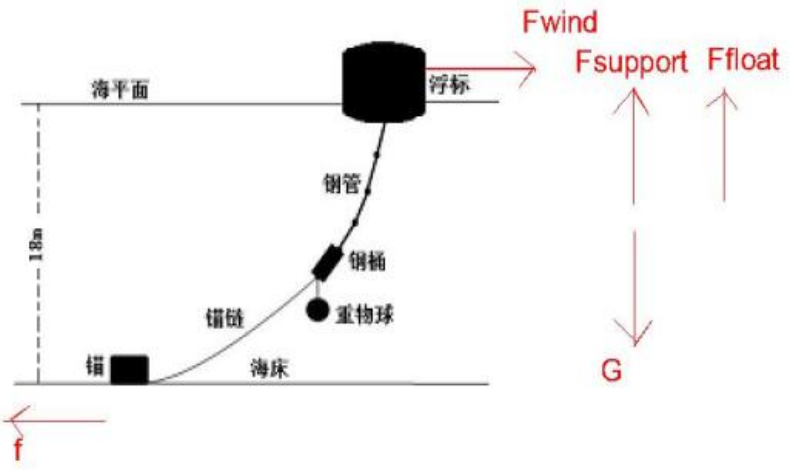

图 1：对系统进行整体的受力分析

将浮标、系泊、水声通讯系统看成一个整体进行受力分析。在水平平面上，系统仅受到对浮标的风力和海底对锚的摩擦力作用。由于整个系统是静止的，摩擦力必然等于风力。即：

$$\begin{cases} f = F_{wind} \\ \mathrm{F_{support}}+\mathrm{F_{float}}=\mathrm{G} \\ F_{wind} = 0.625\bullet\left(2\text{-}h\right)\bullet v^2 \\ F_{float} = \rho_{seawater} g V \end{cases} \qquad (1)$$

由于系统受力平衡处于静止状态，所以最左端锚链切线与海床的夹角固定，从而锚链形状，钢桶与各节钢管的倾斜角度固定。因此浮标的游动区域便可确定：以锚链与锚连接处到浮标的距离为半径的圆。

2. 对锚进行受力分析

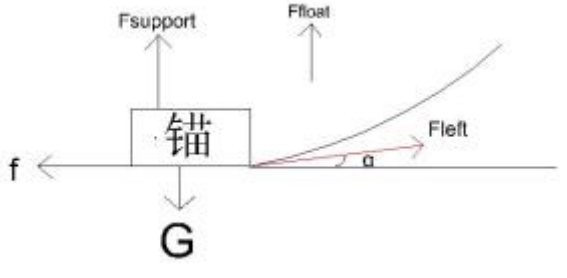

图 2：对锚进行受力分析

如图所示：锚受到五个力，分别是海底对系统的摩擦力，系统整体

的重力，海底对整个的支持力，整体所受到的浮力和锚链左端（对应锚的右端）对其的拉力。列出以下方程式：

$$\begin{cases} f = F_{left} \bullet \cos\alpha = F_{wind} \\ F_{sup\ port} + F_{left} \bullet \sin\alpha = G_{anchor} \end{cases} \tag{2}$$

其中α是锚链末端与锚的链接处的切线方向与海床的夹角。取值范围是$\left[0, 16^{\circ}\right]$，当达到$16^{\circ}$时达到极限状态，系统处于平衡边缘，即摩擦力达到最大。

根据(1),(2)式，可以得到以下关系式，即：

$$\begin{cases} F_{sup\ port} = G - \rho_{seawater} g V \\ F_{left} \bullet \sin\alpha = G_{anchor} - F_{sup\ port} \\ F_{left} \bullet \cos\alpha = F_{wind} \end{cases} \tag{3}$$

由此，得到第一个α与h的关系式，即：

$$\tan\alpha = \frac{32185h - 24138.875}{\left(2.5 - 1.25h\right)v^2} \tag{1-1}$$

3. 对锚链进行受力分析

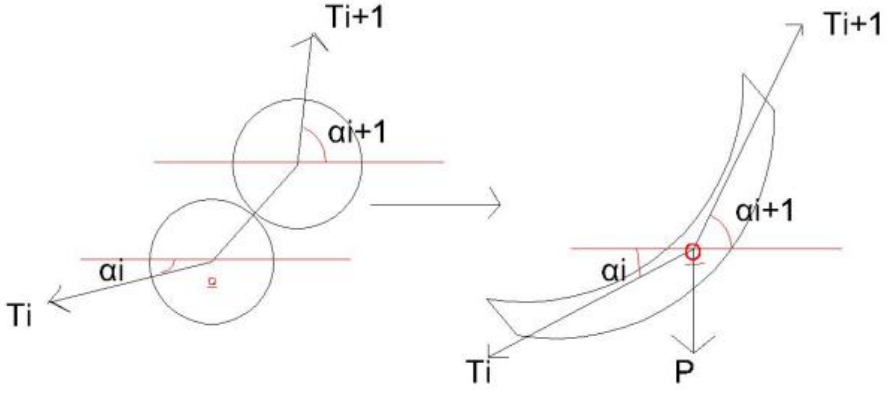

图 3：对锚链的受力分析

根据题目信息，22.05 米的锚链是由 210 条Ⅱ型电焊锚链单元组成的。由于假定海水静止且系统处于平衡状态，所以本问暂不考虑海水对锚链

的法向阻力和切向阻力。设锚链i左右节点的端点分别为i和i+1。作用于锚链上的外力有锚链的重力和锚链两端受到的张力。如上图所示，为了便于分析，我们将第i个锚链的右端和第$i+1$个锚链的左端作为一个单元，同理第$i+1$个锚链的右端和第$i+2$个锚链的左端作为一个单元，依次列推（即图中圆圈部分）。该单元的体积与质量均与一个锚链相同。计算时将单元重量及外荷载均集中在单元的中心上，即图中的O点。对单元i进行受力分析，可得到有以下差分方程，即：

$$\begin{cases} T_i \cos\alpha_i = T_{i+1}\cos\alpha_{i+1} \\ T_i \sin\alpha_i + P = T_{i+1}\sin\alpha_{i+1} \end{cases} \qquad (i = 1,2,\cdots 210) \tag{4}$$

因此得到第i锚链单元和与锚链接的第一单元锚链的关系式，即：

$$\tan\alpha_i = \tan\alpha_1 + \frac{(i-1)\bullet P}{T_1 \bullet \cos\alpha_1} \tag{5}$$

其初始条件为：$T_1 = F_{left}, \alpha_1 = \alpha$

4. 对钢桶和钢管进行受力分析

a. 对钢管进行受力分析

由于钢管的不可形变性，所以可近似将其看为有特殊形状与质量的锚链。其受力分析也与锚链类似。

$$\begin{cases} T_j \cos\beta_j = T_{j+1}\cos\beta_{j+1} \\ T_j \sin\beta_j + Q = T_{j+1}\sin\beta_{j+1} \end{cases} \qquad (j = 1,2,\cdots 4) \tag{6}$$

其中β_j是第j个钢管与水平面的夹角。

b. 对钢桶进行受力分析

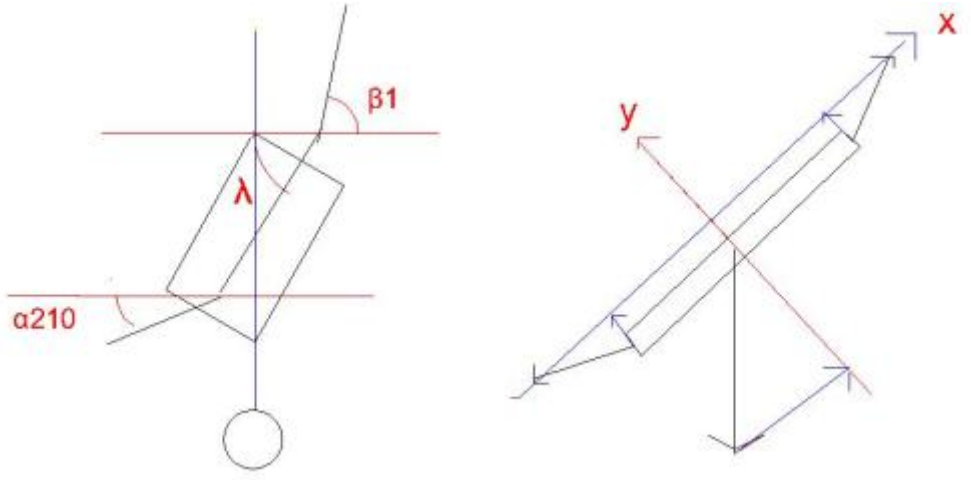

图 4：对钢桶的受力分析

钢桶左边与第 210 根锚链相连，右边与第 1 根钢管相连。其中涉及到三个角度的求解：α，β，λ。由几何知识可知，由于钢桶和重物球都是质量均匀且形状规则的物体，故重心必然在竖直方向。为了避免求解重心纵坐标的复杂过程，我们将钢桶和重心球分开讨论：将钢桶看成一个轻杆，其重心在轻杆的几何中心。而吊着的重物球则可以看成是作用在重心上的外力。

首先无论 λ 为何值，钢桶两边都一定会受到锚链和钢圈的拉力，根据上文分析，下列方程一定会成立：

$$\begin{cases} \overset{i}{T}_{211}\bullet\sin\alpha_{210}+O=\overset{j}{T}_{1}\bullet\sin\beta_{1} \\ \overset{i}{T}_{211}\bullet\cos\alpha_{210}=\overset{j}{T}_{1}\bullet\cos\beta_{1} \end{cases} \tag{7}$$

接着建立平行于钢桶的高的 x 轴和过钢桶的几何中心并垂直于钢桶的高的 y 轴的坐标轴进行受力分解：

$$\begin{cases} \overset{i}{T}_{211}\bullet\sin\left(\lambda-\alpha_{210}\right)=\overset{j}{T}_{1}\bullet\sin\left(\beta_{1}-\lambda\right) \\ \overset{i}{T}_{211}\bullet\cos\left(\dfrac{\pi}{2}-\lambda+\alpha_{210}\right)+\overset{j}{T}_{1}\bullet\cos\left(\dfrac{\pi}{2}-\beta_{1}+\lambda\right)=O\bullet\cos\lambda \end{cases} \tag{8}$$

将 β_1 带回到钢管的受力分析式中进行计算，即可得到钢管的表达式。

5. 由此我们可以得到水深的表达式和浮标的游动区域，即：

$$H' = \sum_{i=1}^{210} S_{anchorarm} \sin\alpha_i + S_{barrel} \sin\lambda + \sum_{j=1}^{4} S_{pipe} \sin\beta_j \tag{9}$$

$$R = \sum_{i=1}^{210} S_{anchorarm} \cos\alpha_i + S_{barrel} \cos\lambda + \sum_{j=1}^{4} S_{pipe} \cos\beta_j \tag{10}$$

$$Area: \ x^2 + y^2 = \left(\sum_{i=1}^{210} S_{anchorarm} \sin\alpha_i + S_{barrel} \sin\lambda + \sum_{j=1}^{4} S_{pipe} \sin\beta_j \right)^2 \tag{11}$$

4.2.2 遗传算法求解最优解

1.遗传算法的执行过程

遗传算法是由 John Holland 教授于 1962 年提出,以达尔文的生物进化论和孟德尔的遗传理论为基础，模拟自然界中生物遗传机制的仿生优化算法[3]。该算法具有与问题域无关的全局搜索能力，且不易陷入局部最优。它的基本步骤如下[4]：

（1）初始化：确定种群规模为 $C=100$，交叉概率 P_c，变异概率 P_m 和终止进化准则——迭代次数 1000 次；设置变量个数：2 个变量，浮标的吃水深度 h 和锚链末端与锚的链接处的切线方向与海床的夹角 α；并确定好变量上下界：$0<h<2, 0\le\alpha\le16^\circ$。随机生成 40 个个体作为初始种群 $X_{(0)}$；置进化代数计数器 $t\to0$。

（2）个体评价：计算 $X_{(t)}$ 中各个个体的适应度函数，即：

$$\mathrm{Fit}(f_{(x)}) = -\mathrm{abs}\left(H' - 18\right)$$

（3）按照个体适应值的大小，从种群中选出适应值较大的一些个体构成交配池；

（4）依次进行选择操作、交叉操作和变异操作；

4.3 问题一模型的求解与分析（取迭代的最后 5 次进行分析）

（一）风速为$12m/s$时系泊系统的各个参数和状态

通过多次迭代后，我们得到了锚链形态的稳定。当风速是$12\mathrm{m}/s$时，如下图所示：与锚的相连处共有 69 节锚链拖在海床上，即此时的α为 0。将拖地的锚链等效成锚的一部分，因此我们改变α的含义，令α为第一个离开海床的锚链与其前一个锚链的链接处切线与海床的夹角，此处即为α_{70}。

给出锚链形态的示意图和其 3D 仿真图。

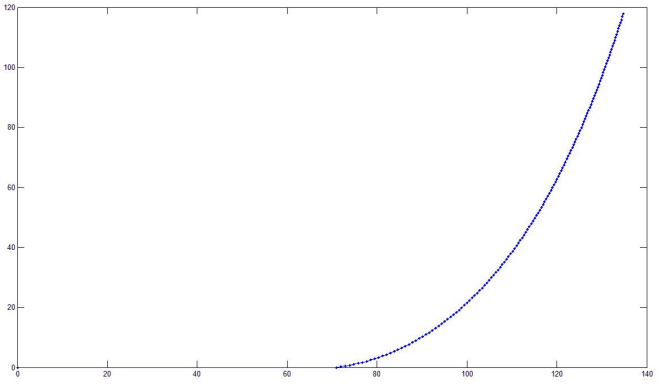

图 5：风速 v=12m/s 时锚链形态的示意图

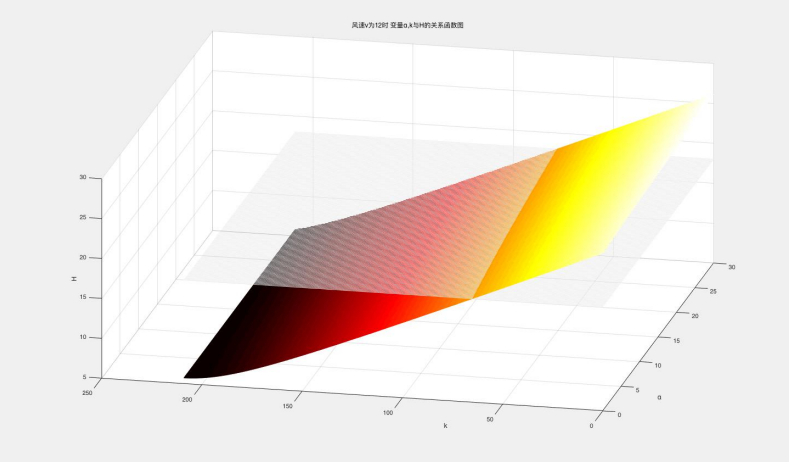

图 6：风速为$v=12m/s$时锚链形状的 3D 图

其中 x 轴是 α ， y 轴是锚链的拖地数量 k ， z 轴是 H ，

下面给出 $v=12m/s$ 时系泊系统的各个参数。

表 1： $v=12m/s$ 时系泊系统的各个参数分析

α	λ	θ_1	θ_2	θ_3	θ_4	h	R
7.8	1.54	1.5538	1.5539	1.554	1.5541	0.7352	14.2560
7.8	1.54	1.5538	1.5539	1.554	1.5541	0.7352	14.2564
7.8	1.54	1.5538	1.5539	1.554	1.5541	0.7352	14.2559
7.8	1.54	1.5538	1.5539	1.554	1.5541	0.7352	14.2789
7.8	1.54	1.5538	1.5539	1.554	1.5541	0.7352	14.2560

根据以上分析：我们可以得到以下结果：在风速为 12m/s 的状态下，与锚相连的前 69 个锚链处于拖地状态，而第 70 个锚链与海床的夹角为 7.8° 。钢桶的倾斜角度为 1.54° ，4 个钢管的倾斜角度分别为 1.5399° ， 1.5538° ， 1.5539° 和 1.5540° 。浮标的吃水深度是 0.7352m，游动区域为 $x^2+y^2=203.2$ 。

（二）风速为 $24m/s$ 是系泊系统的各个参数和状态

当风速是 $24m/s$ 时，如下图所示：与锚的相连处共有 69 节锚链拖在海床上，即此时的 α 为 0。同样将 α 作为第一个离开海床的锚链与其前一个锚链的链接处切线与海床的夹角，此处即为 α_{15} 。

下面给出锚链形态的示意图和其 3D 仿真图。

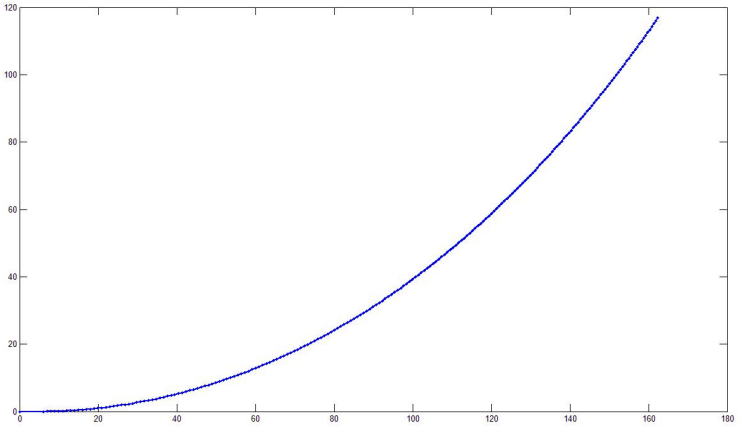

图 7：风速 $24m/s$ 时锚链形态的示意图

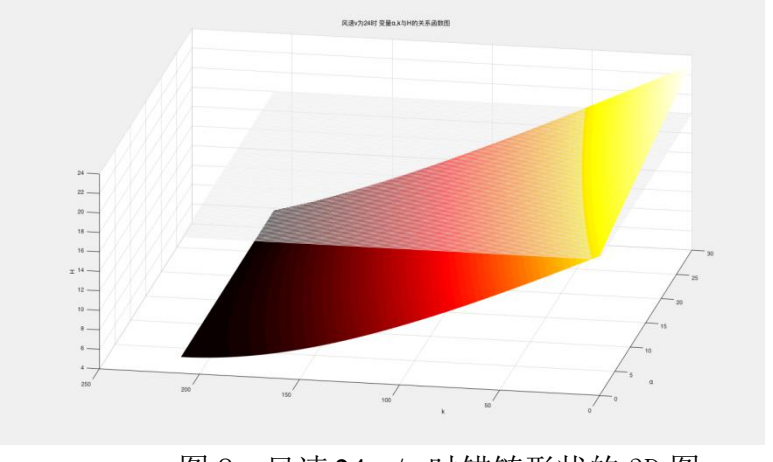

图 8：风速 $24m/s$ 时锚链形状的 3D 图

下面给出 $24m/s$ 时系泊系统的各个参数。

表 2： $24m/s$ 时系泊系统的各个参数分析

α	λ	θ_1	θ_2	θ_3	θ_4	h	R
5.0	1.5	1.506	1.5064	1.5067	1.5071	0.7492	17.3959
4.9	1.5	1.506	1.5063	1.5067	1.5071	0.7492	17.3996
5.0	1.5	1.506	1.5063	1.5067	1.5071	0.7492	17.3995
5.0	1.5	1.506	1.5063	1.5067	1.5071	0.7492	17.3995
4.8	1.5	1.506	1.5063	1.5067	1.5071	0.7491	17.4188

根据以上分析：我们可以得到以下结果：在风速为 24m/s 的状态下，与锚相连的前 14 个锚链处于拖地状态，而第 15 个锚链与海床的夹角为 4.9°。钢桶的倾斜角度为1.5°，4 个钢管的倾斜角度分别为1.5060°，1.5063°，1.5067°和1.5071°。浮标的吃水深度是 0.7492m，游动区域为 $x^2+y^2=302.7$。

五．问题二模型的建立与求解

5.1 问题二的分析

问题二首先要求我们分析$v=36m/s$时系泊系统的各个参数，应用上文的受力分析模型和遗传算法，我们可以得到系泊系统此时的状态，其中α的大小超过了上限16°，说明系统在不改变重物球质量的情况下，已经无法维持平衡。为了维持系统的平衡，需要增加重物球的质量。

根据第一问的分析，重物球的重量应该满足以下条件：①确保系统受力平衡②浮标的吃水深度小于 2 米，即浮标必须有一部分浮在海面之上③$\alpha\le16^\circ$④$\theta\le5^\circ$。因此本文可转换为在这4个约束条下的目标规划问题。因此本问设计算法通过多次的迭代寻找出极限条件的解（即系统刚好处于平衡状态时的极值）。

5.2 问题二模型的建立

5.2.1 风速$v=36m/s$时的系统状态

当风速是36m/s时，α=24.37°明显超过了上限，此时系统将出现不平衡运动。由于上文的算法建立在受力平衡并系统静止的条件下，所以我们现在仍假设下此时系统静止。得出结果：与锚的相连处共有 19 节锚链拖在海床上。钢桶的倾斜角度为1.3434°，4 个钢管的倾斜角度分别为1.4359°，1.4367°，1.4374°和1.4381°。浮标的吃水深度是 0.7733m，游动区域为$x^2+y^2=340.122$。并给出锚链形态 3D 仿真图。

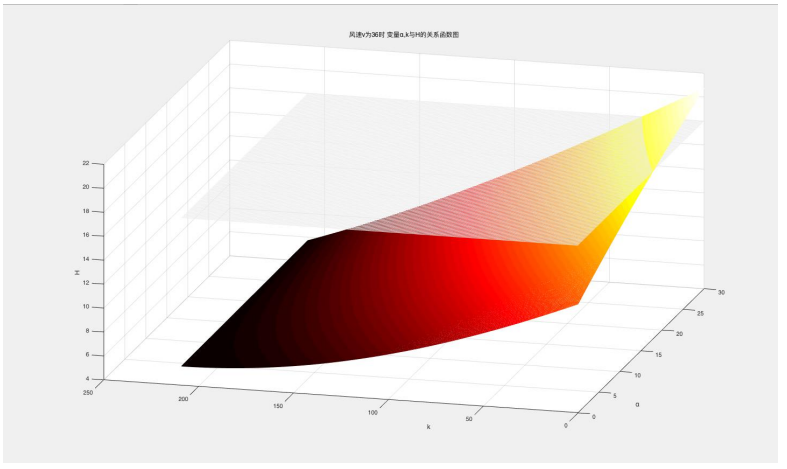

图 9： $v=36m/s$ 时锚链的 3D 仿真图

从以上的数据我们明显可以看出矛盾：当α=24.37°时，系统必定有运动趋势，此时锚链会拉直，也就是不存在拖地的现象。而且根据经验:风速越大，锚链拖地的个数越少。但是此时有 19 节锚链拖地，比 $v=24m/s$ 时的锚链拖地数更多。因此：我们需要增加重物球的重量使系统重新达到平衡。

5.2.2 重物球质量范围的计算

将重物球质量的计算分成 2 步，分别计算其质量的上限和下限。

Step1：上限的计算：根据生活经验可知：重物球的质量远远大于海水对其的浮力，因此连续的增加重物球的质量会使得浮标的吃水深度逐渐增加。当质量增加到某一个值时， h 刚好达到 2m，浮标完全沉没在海水中，该值便是质量的上限。

根据第一问的受力分析：系统整体有： $f=F_{wind}$ 。而此时的风力为 0，所以系统整体的摩擦力为 0，即在水平方向不受力。根据对锚的分析：$f=F_{left}\bullet\cos\alpha$ ，所以整个锚链都不存在拉力，也就是所有不拖地的锚链的与水平面的夹角 α_i 都一样。同理可分析钢桶和钢圈的受力。因此锚链，钢桶，钢管可以看成是一条连接锚和浮标的直线。而质量的上限就是恰好可以使这种极端平衡状态成立的质量。

Step2：下限的计算：题目给出了两个限制条件： $0\le\alpha\le16^\circ$ 和 $85^\circ\le\beta\le90^\circ$ 。根据重物球的悬挂位置，可以初步判断 β 的限制条件为强

条件，因此本问我们选择在α的极限情况下对重物球的质量进行迭代，使得β满足限制条件。根据公式（1-1），我们得到在风速，重物球质量不变的条件下：α和h成反向关系。因此我们将极限条件选为α=16°，即浮标吃水深度最小的情况。

因此质量下限的计算即可转换成在：

$$\begin{cases}\alpha=16^{\circ}\\ 85^{\circ}\le\beta\le 90^{\circ}\\ H^{'}=18\\ \sum\overrightarrow{F}=0\end{cases} \tag{12}$$

条件下对重物球质量的目标规划。由于涉及到过多变量，无法用lingo求出最优解，所以采用数值计算法迭代出重物球的下限值。

给出目标规划求解的算法流程图如下：

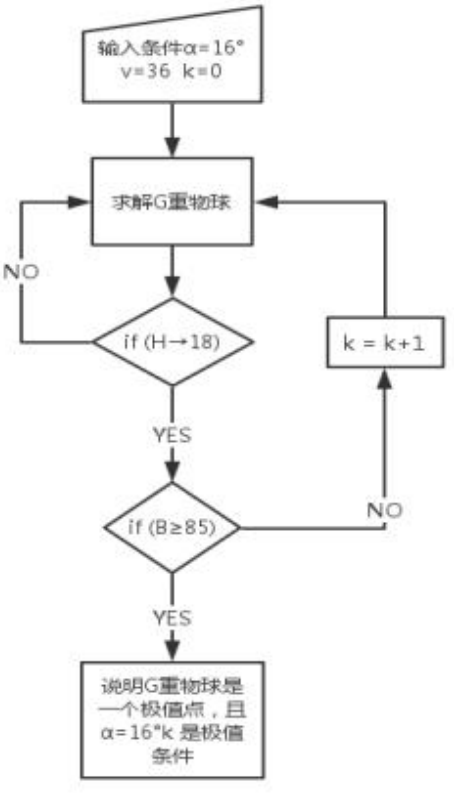

图 10：重物球质量下限的目标规划算法流程图

5.3 问题二的求解与分析

通过计算，得到了重物球质量的上下限分别为 2860kg 和 4626.4kg。因此在问题一的假设下，风速$v=36m/s$时，重物球的质量须介于 2860kg 和 4626.4kg 之间系统才有可能保持平衡。

六．问题三的分析与求解

6.1 问题三的分析

题目要求在考虑风力，水流力和水深情况下的系泊系统的设计,并且要求浮标的吃水深度和游动区域以及钢桶的倾斜角度尽可能的小。根据上文分析，在给定水深，风速，海水流速的情况下，已知重物球质量和锚链的长度就可以构造出以下的函数：

$$\begin{cases} \min h = f_1(g,L) \\ \max \lambda = f_2(g,L) \\ \min R = f_3(g,L) \end{cases} \tag{13}$$

这是一个典型的多目标优化问题,然而吃水深度尽可能小和倾斜角度尽可能小是一对矛盾的目标，运用一般的规划模型很难得到最优解，因此应用粒子群算法。粒子群优化算法因形式简洁，收敛快速和参数调节机制灵活等优点，同时一次运行可得到多个解，因而本认为是求解多目标优化问题最具潜力的方法之一[5]。考虑到粒子算法有容易陷入到局部最优解的缺点，引入 pareto 熵，构造基于 Pareto 熵的多目标粒子群优化算法。

6.2 问题三模型的建立

6.2.1 对模型进行静力分析

根据上文分析，风速和海水速度越大，系统越容易进入扰动状态。而水深越大对锚链的压力越大，锚链趋向拉伸，使得α有减少的趋势。因此,我们便得到了几组极限条件：$V_{water}=1.5m/s$，$V_{wind}=36m/s$，$H=20m$。考虑到系统整体的平衡并保证设备的工作效果，我们仍然沿用第二问的条件：$0\le\alpha\le16^\circ$，$85^\circ\le\beta\le90^\circ$。

首先对浮标进行受力分析，如图所示：浮标此时静止，受到浮力，重力，钢管的拉力，风力和水动力。

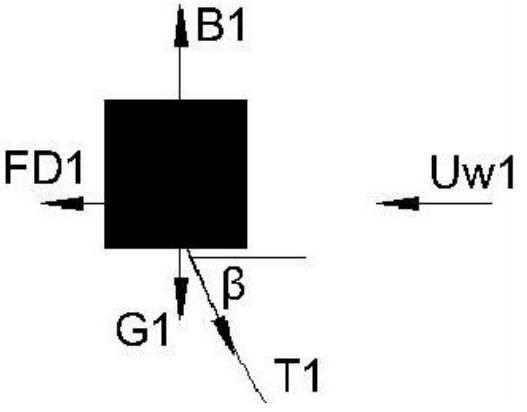

图 11：浮标的受力分析

$$\begin{cases} F_{D1} + U_{W1} = T_1 \bullet \cos\theta_1 \\ B_1 = G_1 + T\sin\theta_1 \end{cases} \tag{14}$$

然后对钢管进行受力分析：

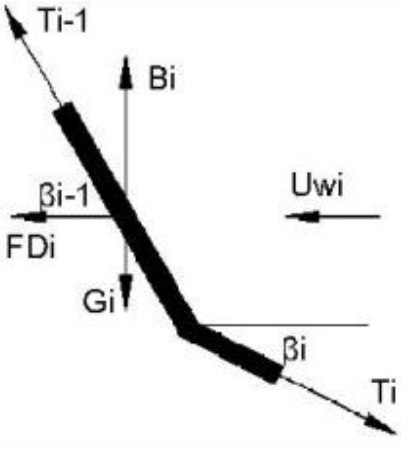

图 12：钢管的受力分析

$$\begin{cases} B_i - G_i + T_{i-1} \bullet \sin\beta_{i-1} = T_i \bullet \sin\beta_i \\ F_{Di} + T_{i-1} \bullet \cos\beta_{i-1} = T_i \bullet \cos\beta_i \end{cases} \tag{15}$$

$$\begin{cases} T_i = \sqrt{\left(B_i - G_i + T_{i-1} \bullet \sin\beta_{i-1}\right)^2 + \left(F_{Di} + T_{i-1} \bullet \cos\beta_{i-1}\right)^2} \\ \tan\beta_i = \dfrac{F_i - G_i + T_{i-1} \bullet \sin\beta_{i-1}}{F_{Di} + T_{i-1} \bullet \cos\beta_{i-1}} \end{cases} \tag{16}$$

对锚链进行受力分析，由于锚链不可弹性形变，当索缆较重，重力远远大于流阻力和附加流体惯性力时，可悬链线模型进行求解[6]。同时根据受力分析可知，一旦有锚链拖地则系统必然处于平衡的状态中，因此我们不考虑有锚链躺底的情况，仅考虑锚链不躺底的情况。

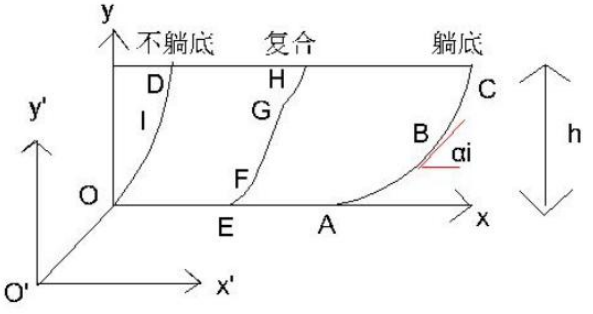

图 13：锚链形状图示

其中 x 和 y 分别为锚泊线水平和垂直跨距，张力的水平和垂向分量分别记为T_h和T_v。根据经典悬链线方程，下端点与海底相切的锚链满足[7]

$$y = a\left(\cosh\frac{x}{a} - 1\right) \tag{17}$$

$$S = a \bullet \sinh\frac{x}{a} = \sqrt{y\left(y + 2a\right)} \tag{18}$$

其中$a = T_h/w$，w为锚链在水中的线密度。

由此：得出锚链不躺底的受力分析式：

$$y_{OD} = \frac{T_h}{w}\left[\cosh\left(\frac{wx_{O'D}}{T_h}\right) - \cosh\left(\frac{wx_{O'O}}{T_h}\right)\right] \tag{19}$$

$$y_{OD} = \frac{T_h}{w}\left[\sinh\left(\frac{wx_{O'D}}{T_h}\right) - \sinh\left(\frac{wx_{O'O}}{T_h}\right)\right] \tag{20}$$

定义：$\alpha=\frac{wx_{O'D}}{T_h}$，$\beta=\frac{wx_{O'O}}{T_h}$，$\lambda=\frac{wx_{OD}}{2T_h}$，则式(16)和(17)可写为：

$$\frac{y_{OD}}{x_{OD}}=\frac{\cosh\bullet\alpha-\cosh\bullet\beta}{2\lambda} \tag{21}$$

$$\frac{S_{OD}}{x_{OD}}=\frac{\sinh\bullet\alpha-\sinh\bullet\beta}{2\lambda} \tag{22}$$

由此可得式：

$$\frac{\sinh\bullet\lambda}{\lambda}=\frac{\sqrt{S_{OD}{}^2-y_{OD}{}^2}}{x_{OD}} \tag{23}$$

对于 OD 之间任意一点 I 有：

$$\alpha_O=\arctan\frac{T_{VO}}{T_h} \tag{24}$$

$$T_{VI}=T_{VO}+wS_{OI} \tag{25}$$

$$\alpha_I=\arctan\frac{T_{VI}}{T_h} \tag{26}$$

$$x_I=\frac{T_h}{w}\left[\ln\left(\frac{1+\sin\alpha_I}{\cos\alpha_I}\right)-\ln\left(\frac{1+\sin\alpha_O}{\cos\alpha_O}\right)\right] \tag{27}$$

$$y_I=\frac{T_h}{w}\left[\frac{1}{\cos\alpha_I}-\frac{1}{\cos\alpha_O}\right] \tag{28}$$

6.2.2 用基于 perato 熵的粒子群算法求解多目标优化问题

Step1:初始化种群：在搜索空间中按照均匀分布随机生成具有初始位置和零速度的 N 个粒子，并根据 MOP 对 N 个粒子分别计算 M 个目标函数值。为每个粒子初始化个体外部档案。

Step2：更新迭代计数器。

Step3:评估进化环境：计算外部档案的 perato 熵和差熵，评估种群进化状态。计算每个最优解个体密度和个体占优强度。

Step4：计算粒子运动参数。

Step5：更新种群：选择粒子i的全局最优解和个体最优解，更新速度和位置，更新个体外部档案和全局外部档案。

Step6：检验算法终止条件。

6.3 问题三的结果与分析

随意取水深，风速和海水流速进行计算即可得到对应的系统参数。为了证明模型的科学性和普适性，我们将极限条件带入模型，计算在$H=20m, v_{wind}=36m/s, v_{seawater}=1.5m/s$的条件下系泊系统的各个参数。

首先给出锚链形状，此时锚链的长度为$S=21.6m$，

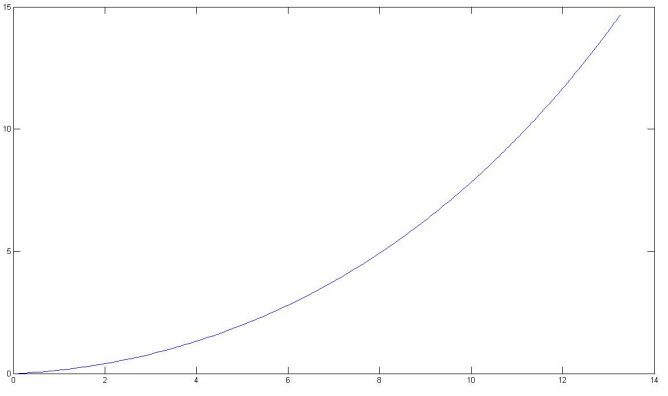

图 15：锚链形状示意图

表 4：$H=20m, v_{wind}=36m/s, v_{seawater}=1.5m/s$时系泊系统的参数

α	λ	θ_1	θ_2	θ_3	θ_4	h	R
5.01	1.3284	1.3085	1.3213	1.3237	1.3260	0.70	14.5

此时满足$0\le\alpha\le16^\circ, 0\le\beta\le5^\circ$的约束条件，即使在极限条件下仍然保持了系统的稳定性和受力平衡，说明模型的抗干扰能力非常强。

七．模型的优缺评价，改进和推广

7.1 模型的优点

1.对锚链受力分析时，考虑到浅水中水流运动比较平稳，将锚链看成集中质量的悬链线，科学的分析了浅海中锚链的受力情况。
2.在无法得到解析解的情况下，成功的运用遗传算法得到了得到了数值解。
3.将求解极限的思想成功的运用到了第二问的分析中，利用受力分析中刚好达到平衡状态的极限成功解出了重物球的质量范围。
4.粒子群算法成功的求解了多目标的优化问题，并设计出了合适的系泊模型。

7.2 模型的缺点

1.模型仅使用了静力学分析，而忽视了水质点对受力物体产生的惯性力等动力学问题，使得模型可能存在误差。
2.没有考虑锚链的横截面，忽视了水质点对锚链的流动阻力。
3.忽视了相邻锚链单元之间的摩擦，受力分析可能不准确。

7.3 模型的推广与应用

1.对处于平衡状态的系统进行静力学分析的方法可用于其余浅海中的系泊分析模型。
2.遗传算法和粒子群算法时效极高，很适合分析最优解，得到合理的数值解。
3.基于集中质量的多边形近似的受力分析模型有效的简化了受力分析的困难度，对于深海系泊系统，如弹簧性锚链也可以用这种方法进行分析。

八．参考文献

[1],[2]王磊，单点系泊系统的动力学研究，2012
[3]马斌，Matlab 语言及实践教程，北京:清华大学出版社，2013:183
[4]韩祯祥，文福拴，模拟进化优化方法及其应用——遗传算法 [J] ，计算机科学，22(2)： 47 -56，1995
[5]胡旺，Gary G.YEN,张鑫，基于 pareto 熵的多目标粒子群优化算法，软件学报，
25（5）：1025-1050，2014
[6]郑艳娜，波浪与浮式结构物相互作用的研究，2006
[7]袁梦，范菊，朱仁传，缪国峰，基于悬链线理论的系泊系统势能，上海交通大学学报，Vol.45 No .4，2011

九. 附录

9.1 遗传算法

best.m

```
function [best_individual, best_fit]=best(pop,fit_value)
% 计算种群中的最适应个体
% param: pop 种群基因序列
% param: fit_value 整个种群的每个个体的适应度
% return:
%           best_individual 最适应个体的基因
%           best_fit 最适应个体的适应度
%=========================================================================
====

[px,py] = size(pop);
best_individual = pop(1,:);
best_fit = fit_value(1);
for i=2:px
    if fit_value(i)>best_fit
        best_individual = pop(i,:);
        best_fit = fit_value(i);
    end
end
```

biary2demical.m

```
function [pop_10_alpha, pop_10_k] = binary2decimal(pop, spl)
% 将种群基因转化为十进制的α值和 k 值
% param: pop 种群基因序列
% param: spl  α基因与 k 基因的划分位置
% return:
%       pop_10_alpha 十进制α值
%       pop_10_k 十进制 k 值
%=========================================================================
====

temp = char(pop+'0');  % 将数组转化为字符串
py = size(pop, 2);
```

```
pop_10_alpha = bin2dec(temp(:, 1:spl));  % 将α基因序列转化为10进制
pop_10_alpha = pop_10_alpha * 16 / (2^spl-1);  % 控制在0~16的范围内
pop_10_k = bin2dec(temp(:, spl+1: py));  % 将k基因序列转化为10进制

end
```

cal_fit_value.m

```
function fit_value=cal_fit_value(pop, spl, v)
% 计算适应度值，所求得的H值越接近18则适应度越高
% param: pop 种群基因序列
% param: spl  α与k的基因分割点
% param: v 风速 get_H()函数所需的参数
% return: fit_value 整个种群的每个个体的适应度
%=========================================================================
====

[x, y] = binary2decimal(pop, spl);
pop_num = size(x);
fit_value = zeros(pop_num);
for i = 1:pop_num
    fit_value(i) = -abs(get_H(v, x(i), y(i))-18);  % v,alpha,k
end
```

crossover.m

```
function fit_value=cal_fit_value(pop, spl, v)
% 计算适应度值，所求得的H值越接近18则适应度越高
% param: pop 种群基因序列
% param: spl  α与k的基因分割点
% param: v 风速 get_H()函数所需的参数
% return: fit_value 整个种群的每个个体的适应度
%=========================================================================
====

[x, y] = binary2decimal(pop, spl);
pop_num = size(x);
fit_value = zeros(pop_num);
for i = 1:pop_num
```

```
    fit_value(i) = -abs(get_H(v, x(i), y(i))-18);  % v,alpha,k
End

initpop.m
function pop = initpop(pop_size,chromlength)
% 随机生成一个种群基因序列
% param: pop_size 种群大小
% param: chromlength 基因长度
% return: pop 种群基因序列
%=========================================================================
====

pop = round(rand(pop_size,chromlength));
End


mutation.m
function [new_pop] = mutation(pop, pm)
% 原来的种群基因序列有一定的机率发生变异
% param: pop 种群的基因序列
% param: pm 基因发生变异的机率
% return: new_pop 变异后的新的基因序列
%=========================================================================
====
[px,py] = size(pop);
new_pop = ones(size(pop));
for i = 1:px
    if(rand<pm)
        mpoint = round(rand*py);
        if mpoint<=0
            mpoint = 1;
        end
        new_pop(i,:) = pop(i,:);
        if new_pop(i, mpoint)==0
            new_pop(i, mpoint) = 1;
        else new_pop(i, mpoint) = 0;
        end
```

```
    else
        new_pop(i, :)=pop(i, :);
    end
end
End

selection.m
function new_pop = selection(pop,fit_value)
% 按照适应度的大小，随机选取个体存活。适应度越大的个体，存活的机率越高。
% param: pop 种群基因序列
% param: fit_value 整个种群的每个个体的适应度
% return:
%           new_pop 新的种群的基因序列
%=========================================================================
====

% 构造轮盘
[px, py]=size(pop);
fit_value = mapminmax(fit_value', 0, 1);  % 由于适应度都为负值，需进行归
一化
total_fit = sum(fit_value);
p_fitvalue = fit_value / total_fit;
p_fitvalue = cumsum(p_fitvalue);  % 概率求和排序

ms = sort(rand(px,1));  % 从小到大排列
fitin = 1;
newin = 1;
while newin <= px
     if (ms(newin)) < p_fitvalue(fitin)
        new_pop(newin, :) = pop(fitin, :);
         newin = newin + 1;
     else fitin = fitin +1 ;
     end

 End

alpha_h.m
```

```
function [h, T1]=alpha_h(alpha_1, v, k)
% param: alpha_1 第一段锚链与水平面之间的角度
% param: v 风速
% return: h 浸水深度
%         T1 锚在水中受到的拉力
%=========================================================================
====
%40738.875
h          =          (2.5*tan(alpha_1)*v^2+24138.875-7.35*k)          /
(32185+1.25*tan(alpha_1)*v^2);
T1 = ((2.5-1.25*h)*v^2) / cos(alpha_1);
end

Alpha_next.m
function [alpha_2,T2] = alpha_next( alpha_1, p, T1 )
% param: alpha_1 前段与水平面之间的角度
% param: p 每一段所受的重力与浮力之差
% param: T1 前段受到的拉力
% return:
%   alpha_2 后段与水平面之间的角度
%   T2 后段受到的拉力
%=========================================================================
====
alpha_2 = atan(tan(alpha_1) + p / (T1 * cos(alpha_1)));
T2 = T1 * cos(alpha_1) / cos(alpha_2);
end

beta_forward.m
function [beta_1, T1] = beta_forward(beta_2, p, T2)
% param: beta_2 后段与水平面之间的角度
% param: p 每一段所受的重力与浮力之差
% param: T2 后段受到的拉力
% return:
%   beta_1 前段与水平面之间的角度
%   T1 前段受到的拉力
%=========================================================================
====
```

```
beta_1 = atan((tan(beta_2) - p / (T2 * cos(beta_2))));
T1 = T2 * cos(beta_2) / cos(beta_1);
end

get_H.m
function [H, L] = get_H(v,alpha_1, k, is_print)
%param: alpha_1 第一段钢管与海床之间的角度
%param: k 躺下的钢条的个数
%param: v 风速
%=========================================================================
====
if nargin < 4
    is_print = 0;  % 默认不 print 出相关数据
end
H = 0;
L = 0;
alpha = zeros(1,211);
alpha(1) = alpha_1 / 180 * pi;
T = zeros(1,211);
beta = zeros(1,4);
F = zeros(1,4);
[h, T(1)] = alpha_h(alpha(1), v, k);
for i= 1:210-k
    [alpha(i+1),T(i+1)]=alpha_next(alpha(i),7.35,T(i));
    H = H + 0.105 * sin(alpha(i));
    L = L + 0.105 * cos(alpha(i));
end
    beta(4) = atan((32185*h-10000)/((2.5 - 1.25*h) * v^2));
    F(4)= (2.5 - 1.25*h) * v^2/ cos(beta(4));

for i=4:-1:1
    if i > 1
    [beta(i-1),F(i-1)] = beta_forward(beta(i),79.88,F(i));
    end
     H = H + sin(beta(i));
     L = L + cos(beta(i));
end
```

```
 H = H + h;
 yita                                                                    =
atan((T(210-k)*sin(alpha(210-k))+F(4)*sin(beta(4)))/(T(210-k)*cos(alpha
(210-k))+F(4)*cos(beta(4))));
 H = H + sin(yita);
 L = L + cos(yita)+k*0.105;

% print 出相关数据
if is_print==1
    fprintf(' β _1 = %f    β _2 = %f    β _3 = %f    β _4
= %f\n',beta(1),beta(2),beta(3),beta(4));
    fprintf(' λ = %f\n', yita);
    fprintf('h = %f\n', h);
    fprintf('L = %f\n', L);
    fprintf('H = %f\n', double(H));
end
end
```

9.2 迭代求解重物球重量的算法

alpha_a.m

```
function [h, T1]=alpha_h(alpha_1, v, k, g)
% param: alpha_1 第一段锚链与水平面之间的角度
% param: v 风速
% param: g 重物球的重量
% return: h 浸水深度
%         T1 锚在水中受到的拉力
%=========================================================================
====
syms h;
h = solve(tan(alpha_1) == ((32185*h - 24138.875+7.35*k-g+12000) / ((2.5 -
1.25*h) * v^2)), h);
h = vpa(h);
syms T1;
T1 = solve(T1 * cos(alpha_1) == (2.5-1.25*h) * v^2, T1);
T1 = vpa(T1);
end
```

alpha_next.m

```
function [alpha_2,T2] = alpha_next( alpha_1 , p, T1 )
% param: alpha_1 前段与水平面之间的角度
% param: p 每一段所受的重力与浮力之差
% param: T1 前段受到的拉力
% return: alpha_2 后段与水平面之间的角度
%         T2 后段受到的拉力
%=========================================================================
====
alpha_2 = atan(tan(alpha_1) +   p / (T1 * cos(alpha_1)));
T2= T1*cos(alpha_1)/cos(alpha_2);
end
```

best_forward.m

```
function [beta_1,T1] = beta_forward(beta_2 ,p , T2)
% param: T2 后段受到的拉力
% param: beta_2 后段与水平面之间的角度
% param: p 每一段所受的重力与浮力之差
% retun: T1 前段受到的拉力
%        beta_1 前段与水平面之间的角度
%=========================================================================
====
beta_1 = atan((tan(beta_2) -   p / (T2 * cos(beta_2))));
T1= T2*cos(beta_2)/cos(beta_1);
end
```

get_H.m

```
function [H, L, yitas] = get_H(v,alpha_1, k, g)
%param: alpha_1 第一段钢管与海床之间的角度
%param: k 躺下的钢条的个数
%param: v 风速
%param: g 重物球的重量
%=========================================================================
====
H = 0;
L = 0;
alpha = zeros(1,211);
```

```
alpha(1) = alpha_1 / 180 * pi;
T = zeros(1,211);
beta = zeros(1,4);
F = zeros(1,4);
[h, T(1)] = alpha_h(alpha(1), v, k, g);
for i= 1:210-k
    [alpha(i+1),T(i+1)]=alpha_next(alpha(i),7.35,T(i));
    H = H + 0.105 * sin(alpha(i));
    L = L + 0.105 * cos(alpha(i));
end
    beta(4) = atan((32185*h-10000)/((2.5 - 1.25*h) * v^2));
    F(4)= (2.5 - 1.25*h) * v^2/ cos(beta(4));

for i=4:-1:1
    if i > 1
    [beta(i-1),F(i-1)]=beta_forward(beta(i),79.88,F(i));
    end
     H = H + sin(beta(i));
     L = L + cos(beta(i));
end
 H = H + h;
 yita                                                                    =
atan((T(210-k)*sin(alpha(210-k))+F(4)*sin(beta(4)))/(T(210-k)*cos(alpha
(210-k))+F(4)*cos(beta(4))));
 yitas = yita / pi * 180;
 H = H + sin(yita);
 L = L + cos(yita);
end
```

main.m

```
clc;
clear;

minD = 1000;
ming = 0;
He=zeros(1,4163);
Ke=zeros(1,4163);
```

```
Ge=zeros(1,4163);
 for k=0:1:22
     for g=10600:100:28600
       [H, L, yitas] = get_H(36,16,k,g);
       He((g-10600)/100+1+k*181)=H;
        Ke((g-10600)/100+1+k*181)=k;
        Ge((g-10600)/100+1+k*181)=g;
       if ((18-H)^2 < minD)
          minD = (18-H)^2;
          ming = g;
          HI = H;
          LI = L;
          Yita = yitas;
       end
     end
 end
plot3(Ke,Ge,He);
disp(HI)
% minDistance
% minAlpha
```

9.3 基于 pareto 的粒子群算法

main.m

```
clc;
clear;

%% 初始参数
dim = 3;  % 粒子维数
x_size = 50;  % 种群个数
max_lteration = 200;  % 迭代次数
wmax = 1.2;  % 惯性因子
wmin = 0.1;  % 惯性因子
c1 = 0.8;  % 算法参数
c2 = 0.8;  % 算法参数

% 粒子初始化
g = randsrc(x_size, 1, 8000:20000);
```

```
h = rand(x_size, 1);
n = randsrc(x_size, 1, 120:300);
x = [g, h, n];

v = zeros(x_size,dim);  % 速度初始化

x_best = x;  % 个体最佳值
g_best = x(1,:);  % 粒子群最佳位置

% 粒子适应度值
yita_f = zeros(1, x_size);  % yita(1)约束  85 < yita(1) < 90
alpha_f = zeros(1, x_size);  % alpha 约束  0 < alpha < 16
T0_f = zeros(1, x_size);  % T0 约束 T0 > 0
Hu_f = zeros(1, x_size);  % 粒子 Hu 目标值 Hu->20
C_f = zeros(1, x_size);  % 粒子 C 目标值 C->0 且 C>0

% 上一次的值
yita_fPrior = zeros(1, x_size);
alpha_fPrior = zeros(1, x_size);
T0_fPrior = zeros(1, x_size);
Hu_fPrior = zeros(1, x_size);
C_fPrior = zeros(1, x_size);

% 计算初始目标向量
for i=1:x_size
    [yita, alpha, T0, Hu, C] = Gain_H(x(i,1), x(i,2), x(i,3));
    yita_f(i) = yita;
    alpha_f(i) = alpha;
    T0_f(i) = T0;
    Hu_f(i) = Hu;
    C_f(i) = C;
end

% 粒子最优适应度
yita_fBest = yita_f;
alpha_fBest = alpha_f;
T0_fBest = T0_f;
```

```
Hu_fBest = Hu_f;
C_fBest = C_f;



%% 初始筛选非劣解
flj = [];  % 非劣解
fljx = [];  % 非劣解的位置
%两个实数相等精度
tol = 1e-7;
for i = 1:x_size
    flag=0;  % 支配标志
    for j=1:x_size
        if j~=i
            if ( (yita_f(j)>90)||(yita_f(j)<85)...
                    ||(C_f(j)<0)...
                    ||(alpha_f(j)<0)||(alpha_f(j)>16)...
                    ||(T0_f(j)<0)...

||((abs(Hu_f(i)-20)>abs(Hu_f(j)-20))&&(abs(C_f(i)-0)>abs(C_f(j)-0))) )
                flag = 1;
                break;
            end
        end
    end

    %判断有无被支配
    flj_num = 0;
    if flag==0
        flj_num = flj_num+1;
        % 记录非劣解
        flj(flj_num, 1) = yita_f(i);
        flj(flj_num, 2) = alpha_f(i);
        flj(flj_num, 3) = T0_f(i);
        flj(flj_num, 4) = Hu_f(i);
        flj(flj_num, 5) = C_f(i);
        % 非劣解位置
```

```
        fljx(flj_num, :) = x(i, :);
    end
end

%% 循环迭代
for iter=1:max_lteration
    % 权值更新
    w = wmax - (wmax-wmin) * iter / max_lteration;  % 惯性因子随着迭代越来越小

    % 从非劣解集中随机选择粒子作为全局最优解
    s = size(fljx,1);
    index = randi(s,1,1);
    g_best = fljx(index,:);

    %% 群体更新
    for i=1:x_size
        % 速度更新

v(i,:)=w*v(i,:)+c1*rand(1,1)*(x_best(i,:)-x(i,:))+c2*rand(1,1)*(g_best-x(i,:));

        % 位置更新
        x(i,:)=x(i,:)+v(i,:);
        % 处理 g h n 越界问题
        if x(i,1) < 8000  % 8000<g<20000
            x(i,1) = x(i,1) + 12000;
        end
        if x(i,1) > 20000
            x(i,1) = 8000 + mod(x(i,1), 20000);
        end
        if x(i,2)<0 || x(i,2)>1  %  0<h<1
            x(i,2) = rand(1,1);
        end
        if x(i,3) < 120  % 120<n<300
            x(i,3) = x(i,3) + 180;
        end
```

```
        if x(i,3) > 300
            x(i,3) = 120 + mod(x(i,3), 300);
        end
    end

    %% 计算个体适应度
    yita_fPrior(:) = 0;
    alpha_fPrior(:) = 0;
    T0_fPrior(:) = 0;
    Hu_fPrior(:) = 0;
    C_fPrior(:) = 0;
    for i=1:x_size
        [yita, alpha, T0, Hu, C] = Gain_H(x(i,1), x(i,2), x(i,3));
        yita_fPrior(i) = yita;
        alpha_fPrior(i) = alpha;
        T0_fPrior(i) = T0;
        Hu_fPrior(i) = Hu;
        C_fPrior(i) = C;
    end

    %% 更新粒子历史最佳
    for i=1:x_size
        %现在的支配原有的，替代原有的
        if( (yita_f(i)>90)||(yita_f(i)<85)...
                    ||(C_f(i)<0)...
                    ||(alpha_f(i)<0)||(alpha_f(i)>16)...
                    ||(T0_f(i)<0)...

||((abs(Hu_f(i)-20)>abs(Hu_fPrior(i)-20))&&(abs(C_f(i)-0)>abs(C_fPrior(
i)-0))) )
                x_best(i,:)=x(i,:);
        end
        %彼此不受支配，随机决定
        if ~( (yita_f(i)>90)||(yita_f(i)<85)...
                    ||(C_f(i)<0)...
                    ||(alpha_f(i)<0)||(alpha_f(i)>16)...
                    ||(T0_f(i)<0)...
```

```
||((abs(Hu_f(i)-20)>abs(Hu_fPrior(i)-20))&&(abs(C_f(i)-0)>abs(C_fPrior(
i)-0))) )...
                &&~( (yita_fPrior(i)>90)||(yita_fPrior(i)<85)...
                    ||(C_fPrior(i)<0)...
                    ||(alpha_fPrior(i)<0)||(alpha_fPrior(i)>16)...
                    ||(T0_fPrior(i)<0)...

||((abs(Hu_f(i)-20)<abs(Hu_fPrior(j)-20))&&(abs(C_f(i)-0)<abs(C_fPrior(
j)-0))) )
            if rand(1,1)<0.5
                x_best(i,:)=x(i,:);
                % 更新目标值
                yita_fBest(i)=yita_fPrior(i);
                alpha_fBest(i)=alpha_fPrior(i);
                T0_fBest(i)=T0_fPrior(i);
                Hu_fBest(i)=Hu_fPrior(i);
                C_fBest(i)=C_fPrior(i);
            end
        end
    end

    %% 更新非劣解集合
    yita_f = yita_fBest;
    alpha_f = alpha_fBest;
    T0_f = T0_fBest;
    Hu_f = Hu_fBest;
    C_f = C_fBest;

    % 更新升级非劣解集合
    s=size(flj,1);  % 目前非劣解集合中元素个数

    %先将非劣解集合和 xbest 合并
    yita_f_b = zeros(1,s+x_size);
    alpha_f_b = zeros(1,s+x_size);
    T0_f_b = zeros(1,s+x_size);
    Hu_f_b = zeros(1,s+x_size);
```

```
    C_f_b = zeros(1,s+x_size);
    yita_f_b(1:x_size) = yita_fBest; yita_f_b(x_size+1:end)=flj(:, 1)';
    alpha_f_b(1:x_size) = yita_fBest; alpha_f_b(x_size+1:end)=flj(:, 2)';
    T0_f_b(1:x_size) = yita_fBest; T0_f_b(x_size+1:end)=flj(:, 3)';
    Hu_f_b(1:x_size) = yita_fBest; Hu_f_b(x_size+1:end)=flj(:, 4)';
    C_f_b(1:x_size) = yita_fBest; C_f_b(x_size+1:end)=flj(:, 5)';
    xxbest=zeros(s+x_size,dim);
    xxbest(1:x_size,:)=x_best;
    xxbest(x_size+1:end,:)=fljx;

    %筛选非劣解
    flj=[];
    fljx=[];
    k = 0;
    tol=1e-7;
    for i=1:x_size+s
        flag = 0;  % 没有被支配
        % 判断该点是否非劣
        for j=1:x_size+s
            if j~=i
                if( (yita_f_b(j)>90)||(yita_f_b(j)<85)...
                    ||(C_f_b(j)<0)...
                    ||(alpha_f_b(j)<0)||(alpha_f_b(j)>16)...
                    ||(T0_f_b(j)<0)...

||((abs(Hu_f_b(i)-20)>abs(Hu_f_b(j)-20))&&(abs(C_f_b(i)-0)>abs(C_f_b(j)
-0))) )
                    flag=1;
                    break;
                end
            end
        end

        %判断有无被支配
        if flag==0
            k = k + 1;
            flj(k, 1) = yita_f_b(i);
```

```
            flj(k, 2) = alpha_f_b(i);
            flj(k, 3) = T0_f_b(i);
            flj(k, 4) = Hu_f_b(i);
            flj(k, 5) = C_f_b(i);
            fljx(k,:) = xxbest(i,:);%非劣解位置
        end
    end

    %去掉重复粒子
    repflag = 0;  % 重复标志
    k = 1;  % 不同非劣解粒子数
    flj_new = [];  % 存储不同非劣解
    fljx_new = [];  % 存储不同非劣解粒子位置
    flj_new(k,:) = flj(1,:);
    fljx_new(k,:) = fljx(1,:);
    for j=2:size(flj,1)
        repflag = 0;  % 重复标志
        for i=1:size(flj_new, 1)
            result=(fljx(j,:)==fljx_new(i,:));
            if length(find(result==1))==dim
                repflag=1;  % 有重复
            end
        end
        %粒子不同，存储
        if repflag==0
            k=k+1;
            flj_new(k,:)=flj(j,:);
            fljx_new(k,:)=fljx(j,:);
        end
    end
    %非劣解更新
    flj=flj_new;
    fljx=fljx_new;
end

len = size(flj, 1)
for i = 1:len
```

```
    i
    flj(i,:)
    fjlx(i,:)
    fprintf('==================');

end
```

betas_foeward.m

```
function [beta_1,T1] = betas_forward(beta_2 ,T2,p,S,v0)
B = 374*S*sin(beta_2)*v0^2;
beta_1 = atan((tan(beta_2) -   p / (T2 * cos(beta_2)+B)));
T1= T2*(cos(beta_2)+B)/cos(beta_1);
end
```

Gain_H.m

```
function [zetas,alphas,T0,Hu,C] = Gain_H(g,h,n)

w = 125;
p=15;
v = 36;
v0 = 1.5;
Th=10250*(h*pi+0.15^2*pi+0.025^2*4*pi);
zeta = zeros(1,5);
yita = zeros(1,5);
T = zeros(1,5);
Fd = zeros(1,5);
F = [724.163 ,20.116 , 20.116, 20.116 ,20.116];
G = [1000,100,100,100,100];
S = [0.3,0.05,0.05,0.05,0.05];

Fd6 = 748*h*v0^2;
zeta(5) = atan((Th-10000)/((2.5-1.25*h)*v^2+Fd6));
T(5) = (Th-10000)/sin(zeta(5));
Hu = h;
Fds = Fd6;
C = 0;
```

```
for i = 5:-1:1
  if i > 1
    [yita(i),zeta(i-1),T(i-1)] = GetNext(zeta(i),T(i), G(i), S(i),0, v0);
  else
     [yita(i),zeta0,T0] = GetNext(zeta(i),T(i), G(i), S(i), g, v0);
  end
    Hu = Hu + sin(yita(i));
    C = C + cos(yita(i));
end
alpha = zeros(1,n+1);
Tb = zeros(1,n+1);

alpha(n+1)=zeta0;
Tb(n+1) = T0;
for i= n:-1:1
    [alpha(i),Tb(i)]=betas_forward(alpha(i+1),Tb(i+1),p,0,0);
   Hu = Hu + 0.12 * sin(alpha(i));
    C = C + 0.12 * cos(alpha(i));
end

   C=C                                                                    -
T0*cos(zeta0)/w*(log((1+sin(zeta0)/cos(zeta0)))-log((1+sin(alpha0)/cos(
alpha0))));
End

GetNext.m
Nextfunction [yita, alpha, T0] = GetNext(beta,T, P, S, g, v0);
t4 = 374*S*v0^2;
e = - P + 2*T*sin(beta) - t4*2;
a = 4* T*cos(beta)/e;
b = 2 * t4 /e;
d = P / e;
% yita = 2 * atan(Fours(a,b,a,d ));
 syms yita;
 yita     =     solve(P+t4-t4*cos(yita)     ==     2*T*(sin(beta)     -
cos(beta)*tan(yita)),yita);
```

```
 yita = real(vpa(yita));
t1 = T*sin(beta-yita) + g*cos(beta);
t2 = t1 / (T*sin(beta)-P+g);
t3 = (t2 + cos(yita))/sin(yita);
alpha = atan(1/t3);
T0 = (T*sin(beta) - P - g) / sin(alpha);
end
```