基于非线性规划对扶壁式挡土墙稳定性最优化设计模型

摘要

在高速公路山体滑坡治理中，挡土墙的防护起到了至关重要的作用。作为保护高速公路免受山体滑坡影响的最后一道屏障，研究如何利用数学模型提高挡土墙稳定性使得高速公路的安全性大大提升，是我们亟待解决的问题。

对于问题一, 首先，对**几何参数、材料参数**和**荷载参数**，分别确定不同参数对单段扶壁式挡土墙的约束。其次，通过建立**抗倾覆稳定性模型和抗滑移稳定性模型**，采用**库仑土压力理论**计算主动土和被动土压力值的计算。在抗倾覆安全系数和抗滑移安全系数求解后，最终通过**偏导**的方式，可以得到稳定性和内摩擦角 φ、墙面板与墙址板的夹角 α 正相关。

对于问题二, 首先，针对单段扶壁式挡土墙的**几何参数、材料参数和荷载参数**，分别确定了不同的约束条件：几何参数方面，明确了底桩半径 L、深度 D、间隔 L_2 等新增参数的范围；材料参数上，明确了钢筋屈服强度、混凝土轴心抗拉强度等关键指标；荷载参数则新增了底柱对土层的侧向摩擦力 F_p，并建立了其与单位摩擦力 T_f 的关系。**稳定性随 α（墙面板与墙趾板的夹角）的增大而单调递增, 随 β（墙面板与墙踵板的夹角）的增大而单调递减**。最终通过建立**基于非线性规划的钢筋位置稳定性最优化设计模型**得出，当钢筋位置**在 (x,y)=(0.3000,0.2434)，并且深度为 5.6 米，结果最优**。

对于问题三, 在问题一、二模型基础上，新增加有关积水对挡土墙影响因素。在建模过程中，重点考虑了浮力对土体重度影响，修正了主动土压力和被动土压力在浮力作用下的大小，利用**水压力分布公式**，我们最终得到了**基于非线性规划的泄水孔位置及数量稳定性最优化设计模型**。在模型求解中，我们通过 **COBYLA 算法**简化步骤，最终得到**泄水孔半径为 0.142 米，总共有 4 层，其中每层泄水孔坐标分别为其坐标分别为（0.44，1.25/1.94/3.44/4.94），（4.15，1.25/1.94/3.44/4.94），（7.05，1.25/1.94/3.44/4.94），（9.95，1.25/1.94/3.44/4.94），单位为米**。

关键字： 库仑土压力理论　COBYLA 算法　水压力分布　非线性规划

一、 问题重述

1.1 问题背景

近年来。随着极端天气、人类活动及众多因素的影响下，高速公路旁的山体滑坡现象频发。加强对高速公路高边山坡的滑坡问题的重视，对高速公路的建设有着极大正向影响，可以在一定程度上减少相关悲剧发生。

山体滑坡的物质基础是因其结构松散的特性，在强降雨情况下容易发生变化。而其他内在因素则有包括地貌、地震、河流及相关的人类活动。为防治高速公路的滑坡，应采取多方面综合山体滑坡防治体系。一方面应保护并利用相关的地质环境，另一方面需调查并研究，采用相应处理的技术措施以避免大规模的滑坡。[1]

在防治高速公路山体滑坡的体系中，挡土墙以其独特的结构功能与技术优势，占据着不可替代的核心地位，成为保障高速公路边坡稳定的重要工程措施。其中，国内外关于扶壁式挡土墙的研究较少，亟须对高速公路组合扶壁式挡土墙力学特性开展研究。[2]

因此，针对高速公路的滑坡问题，对扶壁式挡土墙相关参数的研究及处理，是保障人民生命及财产安全的重要手段。

1.2 问题要求

问题 1 在不考虑挡土墙的底桩并且满足挡土墙不能滑动、倾倒且有一定的强度与抗压力情况下，确定单段扶壁式挡土墙的相关参数，并且建立数学模型来分析相关参数和挡土墙稳定性的影响。其中，尤其要考虑到内摩擦角以及墙面板、墙趾板之间的夹角与稳定性的关系。

问题 2 在问题一的模型上，考虑挡土墙的底桩对不同参数的影响。并且分析稳定性和墙面板、墙趾板角度和墙面板、墙踵板角度的关系。同时，在考虑多段挡板土墙同时施工情况下，通过设计拉筋位置来使得稳定性最优化。

问题 3 由于暴雨因素影响，挡土墙的透水性容易变差进而导致总体压力不足使得挡土墙倾覆。因此需要考虑在挡土墙中，如何设置泄水孔位置及尺寸来增强扶壁式挡土墙稳定性。

二、 问题分析

2.1 问题一分析

对于问题一，我们在基于题目及相关资料上，认为相关重要参数应该和几何参数、材料参数和荷载参数有关联。在此基础上，决定建立抗倾覆稳定性模型和抗滑移稳定性

模型，对主动土和被动土拉力进行理论上求解。在求解过程中，通过推导抗倾覆安全系数和抗滑移安全系数，最终利用偏导得到答案。

2.2 问题二分析

对于问题二，在问题一模型基础上，新增底桩相关参数并明确其约束条件，包括底桩半径、深度、间隔等几何参数范围，钢筋和混凝土的材料强度指标，以及底桩侧向摩擦力与单位摩擦力的关系。核心在于分析稳定性与墙面板和墙趾板夹角（α）、墙面板与墙踵板夹角（β）的关系，通过推导含底桩参数的抗倾覆和抗滑移安全系数模型，得出稳定性随 α 增大而单调递增、随 β 增大而单调递减的规律。同时，针对多段挡土墙施工，建立基于非线性规划的钢筋位置最优化模型，综合考虑钢筋的位置约束、锚固长度等条件，求解得出最优钢筋位置和底桩深度，以实现稳定性最大化。

2.3 问题三分析

对于问题三，在问题一、二模型基础上，重点考虑积水对挡土墙的影响，需修正浮力作用下的土体重度，进而调整主动土压力和被动土压力计算。通过建立水压力分布公式，量化泄水孔对水压力的折减效果，其中泄水孔层数根据墙高设定上限（本题取 4 层，间距 1.5 米），并明确单孔排水效率系数与孔面积、间距等参数的关系。最终构建基于非线性规划的泄水孔位置及数量最优化模型，以抗倾覆和抗滑移安全系数为目标函数，结合泄水孔位置范围、半径等约束条件，求解得出最优泄水孔参数，最大限度降低积水对稳定性的不利影响。

三、 模型假设

为简化问题，本文做出以下假设：

- 假设单个扶壁式挡土墙扶肋为 3 个；
- 假设挡土墙两侧土水平最高与墙面板持平；
- 假设工程使用的 C35 混凝土和 H2B400 钢筋；
- 假设填土为均质、各向同性的无粘性土，忽略粘聚力；
- 假设挡土墙为刚性体，不考虑自身变形对稳定性的影响；
- 假设底桩与土体之间的摩擦力沿深度线性分布，单位摩擦力 τ 为常数；
- 假设钢筋为理想弹塑性材料，忽略应力松弛和徐变效应；
- 假设积水对挡土墙的作用简化为静水压力，采用三角形分布模型；
- 假设泄水孔排水效率系数 η 为常数，不考虑孔道堵塞或渗透系数变化。

四、 符号说明

符号	说明	单位
H	墙高	m
t	扶肋厚度	m
B_1	墙面板顶宽	m
B_2	墙趾板宽度	m
L	扶肋间距	m
E_a	主动土压力	N
E_b	被动土压力	N
γ_1	挡土墙材料的重度	kg/m^3
γ_2	土体的重度	kg/m^3
G	挡土墙自重	kg
M_t	倾覆力矩	$N \cdot m$
M_r	抗倾覆力矩	$N \cdot m$
F_s	滑动力	N
F_r	抗滑力	N
α	墙面板与墙趾板间的夹角	/
β	墙面板与墙趾板间的夹角	/
K_a	主动土压力系数	/
K_b	被动土压力系数	/
K_t	抗倾覆安全系数	/
K_s	抗滑安全系数	/

五、 问题一的模型的建立和求解

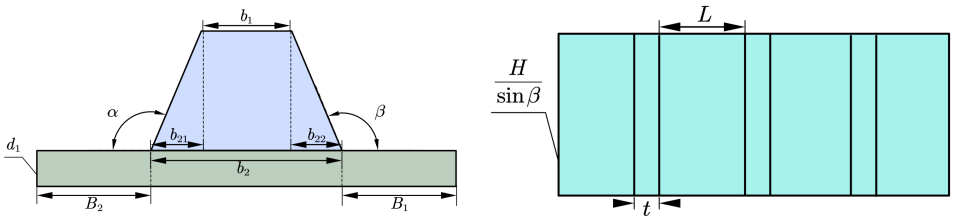
5.1 重要参数的确定

为方便后续问题求解，我们将题目中涉及到挡土墙的有关参数进行了提取，并对不同参数进行了约束。

5.1.1 几何参数的确定

我们规定了以下重要几何参数：H 表示墙高，其约束为不超过 30m；L 表示扶肋间距，约束条件为不超过墙高的一半；t 表示扶肋厚度，约束条件为 $\frac{L}{10}$ 到 $\frac{L}{4}$，且大于 0.3 米；B_1 表示墙面板顶宽，约束条件为 $\frac{H}{4}$ 到 $\frac{H}{2}$, 且大于 0.5 米；B_2 表示墙址板宽度，约

束条件为 $\frac{H}{20}$ 到 $\frac{H}{5}$, 且大于 0.3 米；α 表示墙面板与墙趾板间的夹角；β 表示墙面板与墙踵板的夹角。同时为方便说明，我们将墙面板设定为 b，上部分规定为 b_1,，墙面板下方的左侧为 b_{21}，右侧为 b_{22}。下图为墙面板和墙趾板的主视图和侧视图及相关参数：



(a) 墙面板和墙趾板的主视图　　**(b)** 墙面板和墙趾板的侧视图

图 **1**　墙面板和墙趾板主视图和侧视图

同时，我们将上述几何约束条件数学化, 可以得到以下约束条件：

$$
s.t.\begin{cases}
H \le 30 \\
L \le \frac{H}{2} \\
t \in [\frac{L}{10}, \frac{L}{4}], t \ge 0.3 \\
b = k_1(B_1 + B_2 + b_2) = k_3 L \\
b_1 \ge 0.2 \\
B_1 \in [\frac{H}{4}, \frac{H}{2}], B_2 \ge 0.5 \\
B_2 \in [\frac{H}{20}, \frac{H}{5}], B_2 \ge 0.3 \\
b_2 = b_{21} + b_1 + b_{22} \\
4L + 3t \le 40 \\
tan\alpha = -\frac{H}{b_{21}}, tan\beta = -\frac{H}{b_{22}}
\end{cases} \tag{1}
$$

上述公式中，我们约定 α、β 是墙面板与墙趾板、墙踵板分别向外部方向的夹角。

5.1.2 材料和荷载参数设定

- **材料参数**

  建筑结构的稳定性不仅在于其框架结构的科学性，原材料改良加固也可以有效提高填充材料的紧密性，避免建筑的开裂、收缩等问题。[3] 因此我们考虑到墙体、土体等材料因素。我们规定了以下参数：γ_1 表示挡土墙材料的重度；γ_2 表示土体的重度；φ 表示土体的内摩擦角。在查阅相关文献后[4]，我们确定了 γ_1=24，γ_2=18。

- **荷载参数**

同时，考虑到后续墙体的受力分析，我们认为需要将土分成墙前和墙后两个方面考虑。其中，E_a 表示墙后土体产生的主动土压力，E_b 表示墙前土体产生的主动土压力，G 表示挡土墙自重。

5.1.3 强度参数设定

为了提升模型的准确度，满足题目中挡土墙有足够的强度和承载力，我们设定了强度参数加以约束：

最大弯矩：墙面板承受主动土压力产生的均布荷载，按简支梁计算最大弯矩，再验算弯曲应力是否小于混凝土轴心抗拉强度设计值 f_t（因面板受拉区控制设计）。其中最大弯矩计算公式为：

$$M_{\max} = \frac{q \cdot L^2}{8} \tag{2}$$

其中，均布荷载 q 由填土主动土压力产生，其计算公式基于土力学中主动土压力的基本原理，具体为:

$$q = \gamma \cdot h \cdot K_a \tag{3}$$

其中，γ 为填土重度（文中取值 $18\,\text{kN/m}^3$）；h 为计算点处的填土高度（对于均布荷载简化计算，此处取墙高范围内的平均或特征高度，结合文中参数推导得出对应值）；K_a 为主动土压力系数，由填土内摩擦角 φ 确定，公式为 $K_a = \tan^2\left(45^\circ - \frac{\varphi}{2}\right)$（文中 $\varphi = 35^\circ$，计算得 $K_a \approx 0.27$）。

通过上述参数代入，最终求得文中 $q = 45\,\text{kN/m}$。代入 L = 3m, 可以得到 $M_{\max}$=50.625 kN • m。

截面抵抗矩：我们约定 W 为截面抵抗矩：对于单位宽度面板（取扶肋间距 L 为计算宽度），矩形截面抵抗矩公式为：

$$W = \frac{L \cdot b_1^2}{6} \tag{4}$$

其中 $b_1 = 0.3\,\text{m}$ （面板厚度），代入得：$W = \frac{3\times0.3^2}{6} = 0.045\,\text{m}^3$

截面抵抗矩: 我们约定 W 为截面抵抗矩, 对于单位宽度面板，矩形截面抵抗矩公式为：

$$W = \frac{L \cdot b_1^2}{6} \tag{5}$$

其中 $b_1 = 0.3\,\text{m}$（面板厚度），代入得：$W = \frac{3\times0.3^2}{6} = 0.045\,\text{m}^3$

截面抵抗矩: 我们约定 $\sigma_{\text{弯}}$ 为弯曲应力，其公式为：

$$\sigma_{\text{弯}} = \frac{M_{\max}}{W} \tag{6}$$

代入数据得：$\sigma_{\text{弯}} = 1.125\,\text{MPa} \le f_t = 1.43\,\text{MPa}$。可以看出满足以上约束条件。

最大剪力: 我们约定最大剪力为 $V_{\max}$，则对应的公式为：

$$V_{\max} = \frac{q\cdot}{2} \tag{7}$$

代入 $q = 45\,\text{kN/m}$、$L = 3\,\text{m}$ 得：$V_{\max} = \frac{45\times 3}{2} = 67.5\,\text{kN}$

剪应力: 我们约定剪应力为 τ，矩形截面剪应力公式（取平均剪应力近似计算）为：

$$\tau = \frac{V_{\max}}{A_f} \tag{8}$$

其中 $A_f = t \cdot H$ 为扶肋截面面积（$t = 0.4\,\text{m}$ 为扶肋厚度，$H = 8\,\text{m}$ 为墙高）。代入数据得：$\tau \approx 21.09\,\text{kPa} \le f_t = 1430\,\text{kPa}$。

地基压应力: 墙身自重与土压力合力对地基产生的最大压应力，需小于地基承载力特征值 $[p]$。墙身自重 $G = 420\,\text{kN}$，主动土压力合力 $E_a \approx 540\,\text{kN}$（根据 $E_a = \frac{1}{2}\gamma H^2 K_a$，$K_a$ 为主动土压力系数，由 $\varphi = 35^\circ$ 计算得 $K_a \approx 0.27$）。通过力矩平衡计算，合力作用点距墙趾距离 $e \le B/6$（$B = B_h + B_t = 3.7\,\text{m}$ 为基底总宽），满足偏心距要求。$p_{\max}$：矩形基础偏心受压时最大压应力公式为：

$$p_{\max} = \frac{G + E_{a,y}}{B}\left(1 + \frac{6e}{B}\right) \tag{9}$$

其中 $E_{a,y}$ 为土压力竖向分力（约 $80\,\text{kN}$），代入数据得：$p_{\max} \approx 190\,\text{kPa} \le [p] = 250\,\text{kPa}$

5.2 抗倾覆稳定性模型和抗滑移稳定性模型的建立

5.2.1 主动土、被动土压力的计算

Step1：主动土、被动土压力理论计算

在对主动土、被动土压力的计算时，我们采用库仑土压力理论，分别对主动土和被动土压力讨论求解，具体公式如下：

$$\begin{cases} E_a = \dfrac{1}{2}\gamma_2 H^2 K_a \\ E_b = \dfrac{1}{2}\gamma_2 H^2 K_b \end{cases} \tag{10}$$

上式中，其中 K_a 为主动土压力系数，K_b 为被动土压力系数，两者均与 φ、α 及墙背摩擦角相关。

Step2：主动土、被动土压力系数的推导

墙后土体对挡土墙的压力是导致滑移和倾覆的主要荷载，需用经典土压力理论计算主动土压力。我们考虑土体为无粘性土，可以得到主动土、被动土压力系数：

$$\begin{cases} K_a = \dfrac{\cos^2(\varphi - (\beta - \frac{\pi}{2}))}{\cos^2(\beta - \frac{\pi}{2}) \cdot \cos((\beta - \frac{\pi}{2}) + \delta)\left[1 + \sqrt{\frac{\sin(\varphi+\delta)\sin(\varphi-\beta_1)}{\cos((\beta-\frac{\pi}{2})+\delta)\cos((\beta-\frac{\pi}{2})-\beta_1)}}\right]^2} \\ K_b = \dfrac{\cos^2(\varphi - (\alpha - \frac{\pi}{2}))}{\cos^2(\alpha - \frac{\pi}{2}) \cdot \cos((\alpha - \frac{\pi}{2}) + \delta)\left[1 + \sqrt{\frac{\sin(\varphi+\delta)\sin(\varphi-\beta_1)}{\cos((\alpha-\frac{\pi}{2})+\delta)\cos((\alpha-\frac{\pi}{2})-\beta_1)}}\right]^2} \end{cases} \tag{11}$$

上述公式中，其中：δ 为土与墙面板的摩擦角（我们土设定为混凝土，即 $\delta = \frac{\varphi}{3}$），$\beta_1$ 为墙后填土表面倾角（我们假设水平时 β_1=0）。

5.2.2 抗倾覆安全系数的求解

抗倾覆安全系数求解公式如下：

$$K_t = \frac{M_r}{M_t} \tag{12}$$

其中：

倾覆力矩 M_t：由主动土压力的水平分量产生，绕墙趾（墙址板前端点 O）旋转，促使挡土墙倾倒；

抗倾覆力矩 M_r：由挡土墙自重（墙面板、扶肋、墙址板、墙踵板的重力）、墙踵板上覆土的重力产生，阻碍倾倒。

5.2.3 抗滑移安全系数的求解

抗滑安全系数求解公式如下：

$$K_s = \frac{F_r}{F_s} \tag{13}$$

其中：

滑动力 F_s：主动土压力的水平分量，促使挡土墙沿地基表面滑动；

抗滑力 F_r：由挡土墙总自重 G 产生的摩擦力，以及墙址板上土体反力的垂直分量产生的附加摩擦力。

5.3 模型求解

5.3.1 代入参数

将上述各个参数代入到公式中，求解主动土、被动土压力，可得：

$$V = \frac{3B_1Ht}{2} + (B_1 + B_2 + b_2)(4l + 3t)d_1 + \frac{1}{2}(b_1 + b_2)(4l + 3t)H$$

$$\begin{cases} K_s = \dfrac{(\gamma_1 V + (4L + 3t)E_{ay} + (4L + 3t)E_{by})\mu + (4l + 3t)E_{bx}}{E_{ax}} \\ K_t = \dfrac{G_{ax} + (4l + 3t)(E_{ay}X_{ay} + E_{by}X_{by})}{E_{ax}h_{ax}} \end{cases} \tag{14}$$

上述式子中，E_{ax}、E_{ay} 分别表示 E_a 在水平和竖直方向的作用力；E_{bx}、E_{by} 分别表示 E_b 在水平和竖直方向的作用力，即：

$$\begin{cases} E_{ax} = cos(\alpha - \frac{\pi}{2})E_a \\ E_{ay} = sin(\alpha - \frac{\pi}{2})E_a \\ E_{bx} = cos(\beta - \frac{\pi}{2})E_b \\ E_{bx} = sin(\beta - \frac{\pi}{2})E_b \end{cases} \tag{15}$$

主动土和被动土压力具体受力分析如下图所示：

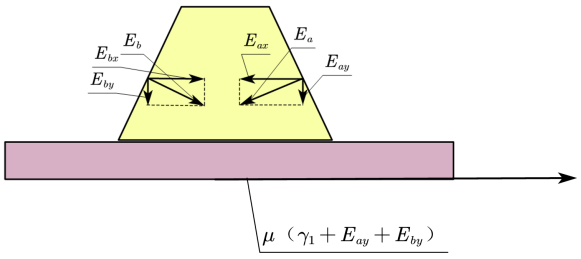


图 **2**　主动土和被动土压力受力分析图

在此基础上，我们可以得到相关力矩的分析图：

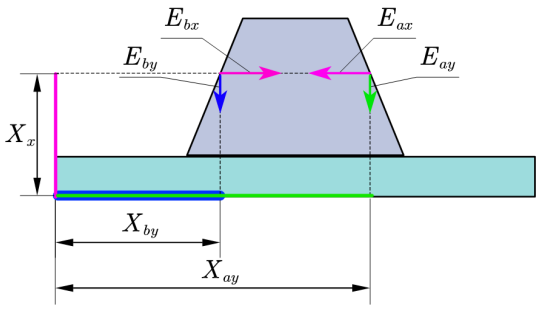


图 **3**　主动土和被动土压力力矩图

5.3.2 偏导求解

为方便说明，我们定义了一个有关稳定性和抗倾覆安全系数、抗滑移安全系数的函数：

$$k_w = w_1 k_s + w_2 k_t \tag{16}$$

其中，可以认为抗倾覆安全系数、抗滑移安全系数两者在本题中有同等重要性，即 $w_1 = w_2 = 0.5$。则当函数分别对 φ 和 α 求偏导时，可以分析稳定性和两个参数之间的变化规律。我们代入在约束条件内的一组参数，即：H=6，L=2.5，t=0.4，b_2=2.5，b_1=0.3，B_1=2，B_2=1，d_1=0.3，β_1=0，β=90。得到原函数：

$$f(\alpha, \varphi) = \frac{A \cdot \cos\left(\frac{\varphi}{3}\right)}{\cos^2(\varphi)} + \frac{B}{C} \tag{17}$$

其中：

$$A = 0.000137786596119929 \left(\sqrt{\frac{\sin(\varphi)\sin\left(\frac{4\varphi}{3}\right)}{\cos\left(\frac{\varphi}{3}\right)}} + 1 \right)^2 \cdot (1655.424 + D - E)$$

$$B = 0.25\left(-\frac{3628.8\left(1-\frac{2}{\tan(\alpha)}\right)\sin^2(\alpha-\varphi)\cos(\alpha)}{(\sqrt{F}+1)^2\sin^2(\alpha)\sin\left(\alpha+\frac{\varphi}{3}\right)}+1671.408-\frac{3142.848}{\tan(\alpha)}\right)$$

$$C = D + \frac{3628.8\cos^2(\varphi)}{\left(\sqrt{\frac{\sin(\varphi)\sin\left(\frac{4\varphi}{3}\right)}{\cos\left(\frac{\varphi}{3}\right)}}+1\right)^2\cos\left(\frac{\varphi}{3}\right)}$$

$$D = \frac{3628.8\sin^2(\alpha-\varphi)\sin(\alpha)\sin\left(\alpha+\frac{\varphi}{3}\right)}{(\sqrt{F}+1)^2}$$

$$E = \frac{2177.28\sin^2(\alpha-\varphi)\cos(\alpha)}{(\sqrt{F}+1)^2\sin^2(\alpha)\sin\left(\alpha+\frac{\varphi}{3}\right)}$$

$$F = \frac{\sin(\varphi)\sin\left(\frac{4\varphi}{3}\right)}{\sin(\alpha)\sin\left(\alpha+\frac{\varphi}{3}\right)}$$

接下来, 我们对上面的原函数依次求偏导，可以得到：

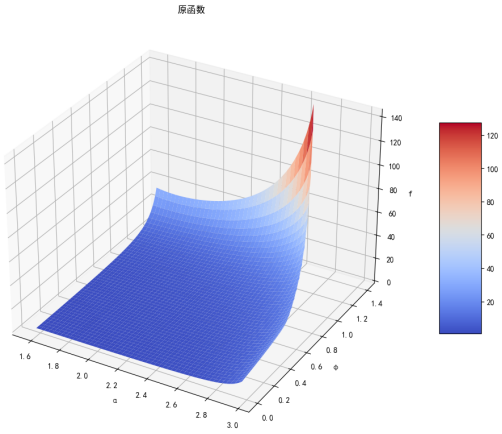
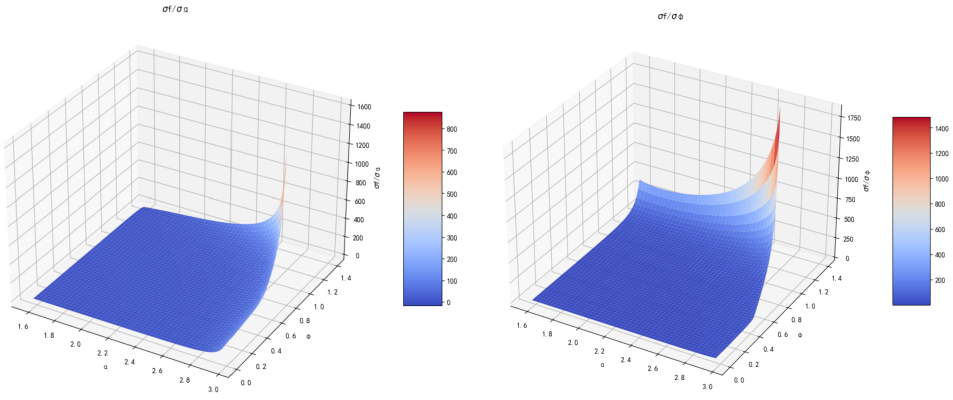


图 **4**　原函数关于 α 和 φ 关系图



(a) 函数关于 α 偏导　　**(b)** 函数关于 φ 偏导

图 **5**　函数关于 α 和 φ 偏导

从图9和图10呈现的函数关系与偏导结果可以清晰看出，挡土墙的整体稳定性与填土内摩擦角 φ、墙面板与墙趾板的夹角 α 均呈现显著的正相关关系。具体而言，当填土内摩擦角 φ 增大时，土体颗粒间的咬合作用增强，抗剪强度提升，使得主动土压力系数 K_a 减小，墙后土体对挡土墙的侧向推力减弱，同时被动土压力系数 K_b 增大，墙前土体提供的抗倾覆与抗滑移阻力增强，最终表现为抗倾覆安全系数 K_t 和抗滑移安全系数 K_s 的同步提高，稳定性随之增强。

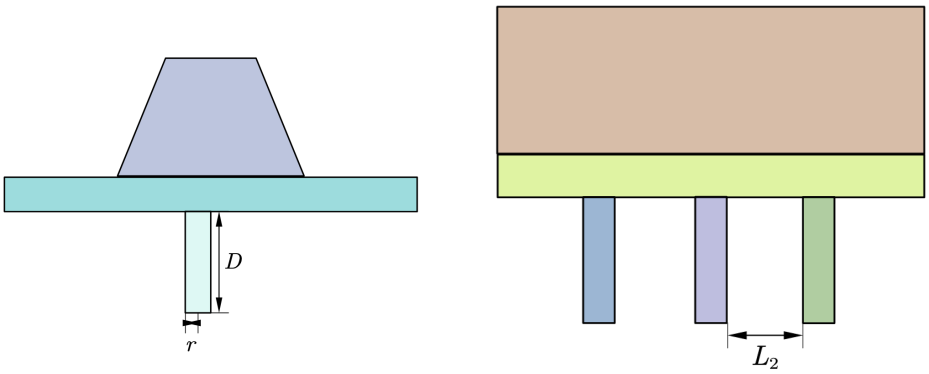
另一方面，当墙面板与墙趾板的夹角 α 增大时（即墙面板倾斜角度更平缓），墙面板与土体的接触面积增加，土压力的竖向分力 $E_{a,y}$ 增大，不仅通过自重与摩擦力提升了抗滑移能力，还使合力作用点向墙趾方向偏移，减小了倾覆力矩的力臂，间接提高了抗倾覆安全系数。这种正相关关系在参数取值范围内呈现连续且单调的变化趋势，说明合理增大填土内摩擦角（如通过改良填土性质）或优化墙面板与墙趾板的夹角设计，均可成为提升挡土墙稳定性的有效途径。

六、 问题二的模型的建立和求解

6.1 参数改变

6.1.1 几何参数的新增

我们在问题一的基础上，新规定了以下重要的几何参数：L 表示底桩的半径，约束条件为大于 0.3 米；D 表示深度，约束条件为 $\frac{H}{5}$ 到 $\frac{H}{2}$；L_2 表示间隔，其约束条件为 $\frac{L_1}{5}$ 到 $\frac{2L_1}{3}$。示例图如下：



(a) 底桩主视图　　**(b)** 底桩侧视图

图 **6**　底桩主视图及侧视图

6.1.2 荷载参数的新增

同样，我们在问题一的基础上，新规定了以下重要的荷载参数：F_p 表示底柱对土层的侧向摩擦力，T_f 表示单位摩擦力，则有

$$F_{px} = \int_0^{\frac{\pi}{2}} \pi r L T_f cos\theta \mathrm{d}\theta = \pi r L T_f \tag{18}$$

同时，我们更新了有底桩侧向摩擦力的受力分析图：

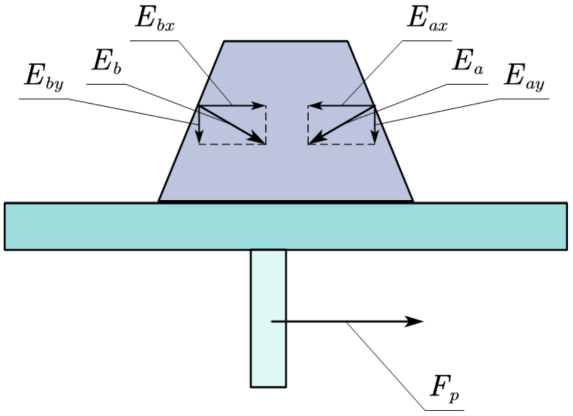


图 **7**　有底桩侧向摩擦力的受力分析图

在受力分析的基础上，我们可以得到相关力矩的分析图：

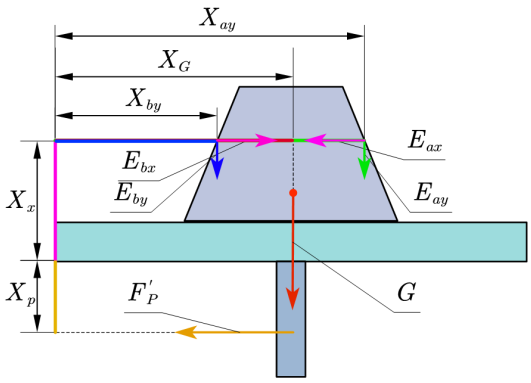


图 **8**　有底桩侧向摩擦力的力矩分析图

6.2 规律的变化

在模型一的基础上，我们发现增加底桩后，K_s 和 K_t 发生了以下变化:

$$\begin{cases} K_s' = \dfrac{[\gamma_1 V + (4L+3t)E_{ay} + (4L+3t)E_{by}]\mu + n\pi r L T_f + (4l+3t)E_{bx}}{(4L+3t)E_{ax}} \\ K_t' = \dfrac{G_{ax}X_G + n\gamma_1\pi r L T_f(B_L + \frac{b_2}{2}) + (4l+3t)(E_{ay}X_{ay} + E_{by}X_{by})n\pi r L T_f + \frac{b_2}{2}}{(4L+3t)(E_{ax}-E_{bx})\frac{H}{3}} \end{cases} \tag{19}$$

上述式子中，相对于第一问的模型，我们主要重新更新了挡土墙的体积公式，并且添加了有关底柱的相关参数。代入式子中，我们可以得到以下原函数公式：

$$f(\alpha,\beta) = \frac{3\cdot A}{2H\cdot B} + \frac{C\cdot D}{H^2 y_2(4L+3t)\sin^2(\beta-\varphi)} \tag{20}$$

其中：

$$A = 0.000137786596119929\left(\sqrt{\frac{\sin(\varphi)\sin\left(\frac{4\varphi}{3}\right)}{\cos\left(\frac{\varphi}{3}\right)}}+1\right)^2\cdot(1655.424+D-E)$$

$$B = 0.25\left(-\frac{3628.8\left(1-\frac{2}{\tan(\alpha)}\right)\sin^2(\alpha-\varphi)\cos(\alpha)}{(\sqrt{F}+1)^2\sin^2(\alpha)\sin\left(\alpha+\frac{\varphi}{3}\right)}+1671.408-\frac{3142.848}{\tan(\alpha)}\right)$$

$$C = D + \frac{3628.8\cos^2(\varphi)}{\left(\sqrt{\frac{\sin(\varphi)\sin\left(\frac{4\varphi}{3}\right)}{\cos\left(\frac{\varphi}{3}\right)}}+1\right)^2\cos\left(\frac{\varphi}{3}\right)}$$

$$D = \frac{3628.8\sin^2(\alpha-\varphi)\sin(\alpha)\sin\left(\alpha+\frac{\varphi}{3}\right)}{(\sqrt{F}+1)^2}$$

$$E = \frac{2177.28 \sin^2(\alpha - \varphi) \cos(\alpha)}{(\sqrt{F} + 1)^2 \sin^2(\alpha) \sin\left(\alpha + \frac{\varphi}{3}\right)}$$

$$F = \frac{\sin(\varphi) \sin\left(\frac{4\varphi}{3}\right)}{\sin(\alpha) \sin\left(\alpha + \frac{\varphi}{3}\right)}$$

在函数分别对 α 和 β 求偏导后，我们可以得到：

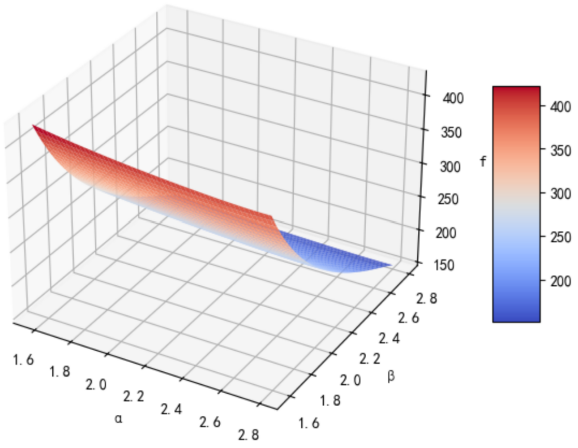
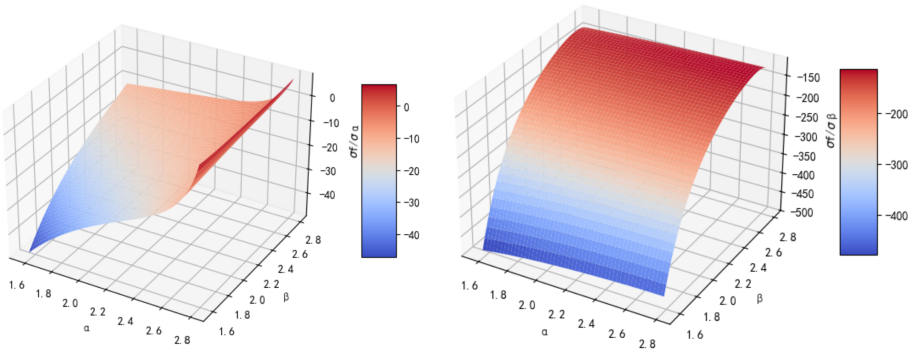


图 **9**　原函数关于 α 和 β 关系图



(a) 函数关于 α 偏导　　**(b)** 函数关于 β 偏导

图 **10**　函数关于 α 和 β 偏导

从图9中可以清晰观察到挡土墙稳定性与几何参数之间的关系呈现出显著的规律性。稳定性指标（由函数 f 表示）随 α（墙面板与墙趾板的夹角）的增大而单调递增，当 α 从 1.6 增加到 2.8 时，稳定性指标从约 200 增加到 400，增幅达 100%。偏导数 $\frac{\partial f}{\partial \alpha}$ 始终为负值，表明 α 每增加一个单位，稳定性指标将减少约 40 个单位。这一趋势与挡土墙工程中” 增大墙背倾角可提高稳定性” 的经验规律相符。

与此同时，稳定性指标随 β（墙面板与墙踵板的夹角）的增大而单调递减，当 β 从 1.6 增加到 2.8 时，稳定性指标从约 400 减少到 200，降幅达 50%。偏导数 $\frac{\partial f}{\partial \beta}$ 始终为负

值，且绝对值大于 $\frac{\partial f}{\partial \alpha}$，表明 β 对稳定性的影响比 α 更显著。如图10所示，在 $\alpha \approx 2.8$，$\beta \approx 1.6$ 附近，稳定性指标达到最大值约 400；而在 $\alpha \approx 1.6$，$\beta \approx 2.8$ 附近，稳定性指标降至最小值约 200。两个参数对稳定性的影响呈现相反趋势，这为挡土墙的优化设计提供了重要依据。

6.3 基于非线性规划的钢筋位置稳定性最优化设计模型

6.3.1 目标函数的建立

同问题一，我们在建立模型时，主要考虑抗倾覆安全系数和抗滑移安全系数对总体稳定性影响。因此，在上述模型上，我们针对钢筋相关参数做了进一步扩充，可以得到：

$$\begin{cases} k_s'' &= \dfrac{[\gamma_1 V + (4L+3t)E_{ay} + (4L+3t)E_{by}]\mu + n\pi r L T_f + (4l+3t)E_{bx}}{(4L+3t)E_{ax}} \\ k_t'' &= \dfrac{G'_{ax}X'_G + n\gamma_1 \pi r L T_f \left(B_L + \dfrac{b_2}{2}\right) + (4l+3t)\left(E_{ay}X_{ay} + E_{by}X_{by}\right)}{(4L+3t)\left(E_{ax}-E_{bx}\right)\dfrac{H}{3}} \\ & \quad + \dfrac{n\pi r \dfrac{D^2}{2} T_f + \dfrac{b_2}{2} + 2T_h(y+d_1)}{(4L+3t)\left(E_{ax}-E_{bx}\right)\dfrac{H}{3}} \end{cases} \tag{21}$$

将新的参数代入公式 16 中，可以得到目标函数：

$$k_w'' = w_1 k_s'' + w_2 k_t'' \tag{22}$$

其中，可以认为抗倾覆安全系数、抗滑移安全系数两者在本模型中同等重要，即 $w_1 = w_2 = 0.5$。

6.3.2 钢筋约束条件的建立

我们在建立模型时，考虑到本模型相较以上模型最大进步是在新增了钢筋这一约束条件，因此在考虑约束时重点考虑钢筋的影响。我们约定了 T_h 为钢筋水平拉力，则对应的拉力应该比钢筋内部摩擦力和挡土墙外部摩擦力最小值要小。则有：

$$T_h \leq min(\pi d_a \tau_t, F_p + f_{地}) \tag{23}$$

其中：

$$f_{地} = [\gamma_1 V + \gamma_2 V_j + (E_{a,y} + E_{b,y})(4L+3t)]\mu, \quad F_P = \pi r L_a \tau f$$

同时，我们认为钢筋应该在挡土墙内部，因此为方便后续模型求解，我们建立如下坐标系：

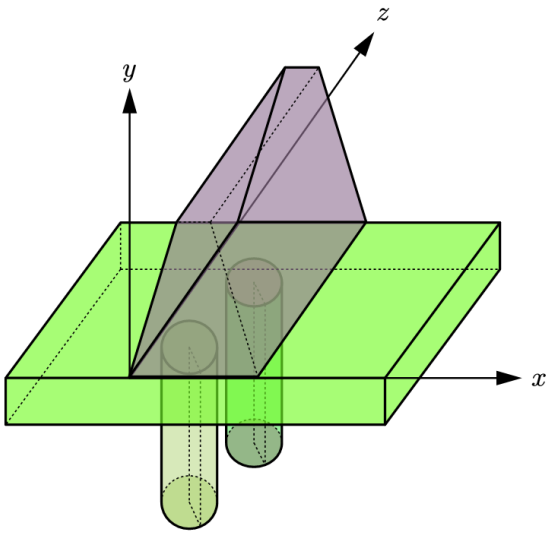


图 **11**　坐标系参考图

其中，我们以挡土墙主视图中梯形左下为坐标原点，水平向右为 x 轴，竖直向上为 y 轴，同时贯深向内部为 z 轴，建立直角坐标系。因此，有以下约束：

$$\begin{cases} \frac{|tanx+y|}{\sqrt{tan\alpha^2+1}} \geq 0.04 \\ \frac{|-tan\beta\cdot x-y+tan\beta(b-\frac{H}{tan\beta}-\frac{H}{tan\alpha})|}{\sqrt{tan\beta^2+1}} \geq 0.04 \end{cases} \tag{24}$$

其中，x 表示钢筋的横坐标，y 表示钢筋的纵坐标。并且我们为了让钢筋在挡土墙中，对不同分段 (x, y) 进行了分段约束：

$$\begin{cases} x \in [0, -\frac{H}{tan\alpha}], y \in [0.04, -tan\alpha] \\ x \in [-\frac{H}{tan\alpha}, b_1 - \frac{H}{tan\alpha}], y \in [0.04, H - 0.04] \\ x \in [b_1 - \frac{H}{tan\alpha}, b_1 - \frac{H}{tan\alpha} - \frac{H}{tan\beta}], y \in [0.04, -tan\beta \cdot x + tan\beta(b - \frac{H}{tan\beta} - \frac{H}{tan\alpha}] \end{cases} \tag{25}$$

上述式子中，我们对钢筋的横纵坐标进行了约束，但是实际上钢筋的横贯位置也是需要考虑的。因此我们查阅有关混凝土结构设计方案[5]，得到以下约束：

$$\alpha \cdot \frac{f_y}{f_t} \cdot d \leq L_a \leq \frac{4L + 3t}{2} \tag{26}$$

式中：

- $L_{a,\min}$–最小锚固长度（m），为保证拉筋在土体中不发生拔出破坏所需的最小埋置深度；
- α–锚固长度修正系数，无量纲，其取值与拉筋类型及锚固条件密切相关。我们取得的带肋钢筋，α 通常取 0.14；
- f_y–拉筋屈服强度（MPa），取决于钢筋的材质和等级。我们使用的 HRB400 级钢筋的屈服强度标准值为 400MPa；

- f_t–墙体混凝土轴心抗拉强度设计值（MPa），与混凝土强度等级相关。对于 C30 混凝土，f_t 取 1.43MPa；
- d–拉筋直径（m），为拉筋的公称外径。

下图所示为我们加入拉筋后的示意图：

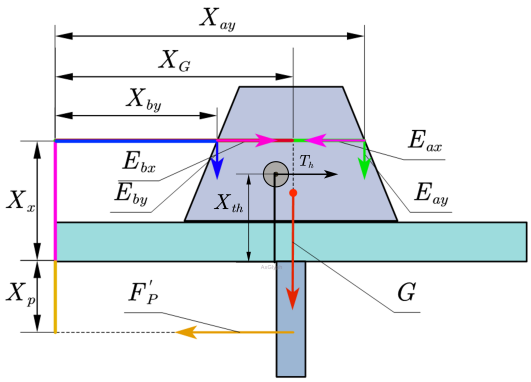


图 **12**　加入拉筋后力矩示意图

图中清晰呈现了拉筋在扶壁式挡土墙中的力学作用机制，是模型二钢筋位置优化设计的核心可视化支撑。图中展示了拉筋的水平布置方式及其与墙面板、扶肋等构件的空间关系，直观反映了拉筋通过水平拉力 T_h 产生抗倾覆力矩的过程，该力矩以拉筋作用点到墙趾旋转中心的垂直距离为矩臂，直接抵消主动土压力的倾覆效应，同时其与土体的摩擦力可辅助提升抗滑移性能。

图中隐含的拉筋位置参数 (x, y) 是模型优化的关键变量，需严格满足多重约束条件。空间上，拉筋需位于墙体有效受力区域内，避免超出混凝土保护层范围；力学上，其拉力需控制在钢筋屈服强度与土体拔出阻力的限值内，确保工作在弹性阶段。这些约束通过图中虚线标注的安全边界与力的传递路径得以具象化，为非线性规划模型的约束条件设定提供了直观依据。

该图直接支撑了模型二的优化结论，解释了拉筋最优位置的合理性。此位置下，拉筋力臂达到理论最优值，抗倾覆力矩与倾覆力矩形成最佳平衡，且锚固长度符合规范要求，既保证了结构安全，又避免了材料浪费。通过将抽象的力学公式转化为具象的空间-力关系，该图为拉筋优化设计的理论推导与结果验证提供了不可或缺的可视化支撑。

6.4 模型求解

综合以上模型，并结合模型一中所给约束条件，我们可以得到以下公式：

$$max = k''_w = w_1 k''_s + w_2 k''_t$$

$$s.t. \begin{cases} k''_s = \dfrac{[\gamma_1 V + (4L+3t)E_{ay} + (4L+3t)E_{by}]\mu + n\pi r L T_f + (4l+3t)E_{bx}}{(4L+3t)E_{ax}} \\ k''_t = \dfrac{G'_{ax}X'_G + n\gamma_1 \pi r L T_f \left(B_L + \dfrac{b_2}{2}\right) + (4l+3t)\left(E_{ay}X_{ay} + E_{by}X_{by}\right)}{(4L+3t)\left(E_{ax}-E_{bx}\right)\dfrac{H}{3}} \\ + \dfrac{n\pi r \dfrac{D^2}{2} T_f + \dfrac{b_2}{2} + 2T_h(y+d_1)}{(4L+3t)\left(E_{ax}-E_{bx}\right)\dfrac{H}{3}} \\ H \le 30, r \ge 0.3, L \le \frac{H}{2} \\ t \in [\frac{L}{10}, \frac{L}{4}], t \ge 0.3 \\ b = k_1(B_1 + B_2 + b_2) = k_3 L \\ b_1 \ge 0.2, 4L + 3t \le 40 \\ B_1 \in [\frac{H}{4}, \frac{H}{2}], B_2 \ge 0.5 \\ B_2 \in [\frac{H}{20}, \frac{H}{5}], B_2 \ge 0.3 \\ b_2 = b_{21} + b_1 + b_{22}\frac{|tanx+y|}{\sqrt{tan\alpha^2+1}} \ge 0.04 \\ tan\alpha = -\frac{H}{b_{21}}, tan\beta = -\frac{H}{b_{22}} \\ \gamma_1 = 24, \gamma_2 = 18, \mu = 0.6, \gamma_3 = 78.5 \\ \tau_t = 0.92Mpa, \tau_t = 30kMpa, V_{\max} = \frac{q\cdot}{2} \\ D \in [\frac{H}{5}\frac{H}{2}], L_1 \in [\frac{L}{5}, \frac{L}{3}], M_{\max} = \frac{q\cdot L^2}{8}, \sigma_{弯} = \frac{M_{\max}}{W}, \tau = \frac{V_{\max}}{A_f} \\ q = \gamma \cdot h \cdot K_a, W = \frac{L\cdot b_1^2}{6} \\ p_{\max} = \frac{G+E_{a,y}}{B}\left(1 + \frac{6e}{B}\right), T_h \le min(\pi d_a \tau_t, F_p + f_{地}) \\ f_{地} = [\gamma_1 V + \gamma_2 V_j + (E_{a,y} + E_{b,y})(4L+3t)]\mu, \quad F_P = \pi r L_a \tau f \\ \frac{|-tan\beta\cdot x - y + tan\beta(b - \frac{H}{tan\beta} - \frac{H}{tan\alpha})|}{\sqrt{tan\beta^2+1}} \ge 0.04 \\ \alpha \cdot \frac{f_y}{f_t} \cdot d \le L_a \le \frac{4L+3t}{2} \\ x \in [0, -\frac{H}{tan\alpha}], y \in [0.04, -tan\alpha] \\ x \in [-\frac{H}{tan\alpha}, b_1 - \frac{H}{tan\alpha}], y \in [0.04, H - 0.04] \\ x \in [b_1 - \frac{H}{tan\alpha}, b_1 - \frac{H}{tan\alpha} - \frac{H}{tan\beta}], y \in [0.04, -xtan\beta \\ + tan\beta(b - \frac{H}{tan\beta} - \frac{H}{tan\alpha}] \end{cases} \tag{27}$$

综合上述式子，我们可以得到：

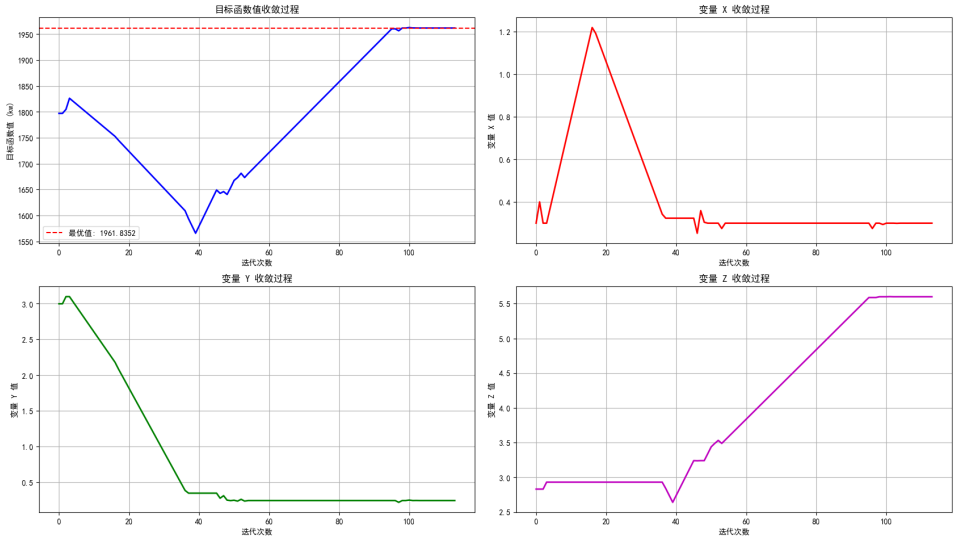


图 **13**　目标函数与拉筋位置优化过程

从上图可知，当钢筋位置在 (x,y)=(0.3000,0.2434)，并且深度为 5.6000m，结果最优。

七、 问题三的模型的建立和求解

7.1 基于非线性规划的泄水孔位置及数量稳定性最优化设计模型

同问题二，我们在建立模型三时同样延用了上述基本模型，在此基础上添加了积水对挡土墙的影响，其中包括浮力对土体重度影响、降水对泄水孔层数影响、泄水孔位置和墙厚水压力分布等众多因素。

7.1.1 目标函数的建立

同问题一，我们在建立模型时，主要考虑抗倾覆安全系数和抗滑移安全系数对总体稳定性影响。因此，在上述模型上，我们针对泄水孔相关参数做了进一步扩充，可以得到：

$$
\begin{cases}
k_s''' = \dfrac{[\gamma_1 V + (4L+3t)E_{ay} + (4L+3t)E_{by} + (4L+3t)P_{Wy}]\mu}{(4L+3t)E_{ax} + P_{wy}} \\
+\frac{n\pi r L T_f + (4l+3t)E_{bx}}{(4L+3t)E_{ax}+P_{wy}} \\
k_t''' = \dfrac{G'_{ax}X'_G + (P_{wy} + n\gamma_1\pi r L T_f)\left(B_L + \dfrac{b_2}{2}\right)}{(4L+3t)\left(E_{ax}-E_{bx}\right)\dfrac{H}{3} + P_{wx}\frac{H}{3}} \\
+\dfrac{(4l+3t)\left(E_{ay}X_{ay} + E_{by}X_{by}\right)}{(4L+3t)\left(E_{ax}-E_{bx}\right)\dfrac{H}{3} + P_{wx}\frac{H}{3}} \\
+\dfrac{n\pi r\dfrac{D^2}{2}T_f + \dfrac{b_2}{2} + 2T_h(y+d_1)}{(4L+3t)\left(E_{ax}-E_{bx}\right)\dfrac{H}{3} + P_{wx}\frac{H}{3}}
\end{cases}
\tag{28}
$$

将新的参数代入公式 16 中，可以得到目标函数：

$$
k_w''' = w_1 k_s''' + w_2 k_t''' \tag{29}
$$

其中，可以认为抗倾覆安全系数、抗滑移安全系数两者在本模型中同等重要，即 $w_1 = w_2 = 0.5$。

7.1.2 浮力对土体重度影响

墙后积水会使土体处于饱和状态，此时土体的有效重度 γ'(扣除浮力后的重度) 为:

$$
\gamma' = \gamma_{sat} - \gamma_w \tag{30}
$$

其中，γ_{sat} 表示饱和土体重度，我们在这里采用的是砂性土，取值为 20KN/m^3。并且在这里我们取 γ_w 为水的重度，取值为 9.8KN/m^3。上述公式的意义为浮力抵消部分土体自重，导致作用于墙面板的主动土压力减小。

7.1.3 主动土压力和被动土压力在浮力作用下的修正

基于上述浮力的影响，我们修正了主动土压力和被动土压力，即：

$$
\begin{cases}
E_a = \dfrac{1}{2}\gamma'_2 H^2 K_a \\
E_b = \dfrac{1}{2}\gamma'_2 H^2 K_b
\end{cases}
\tag{31}
$$

其中：

$$
\gamma'_2 = \gamma_{sat} - \gamma_w = 20 - 9.8 = 10.2\text{KN/m}^3
$$

7.1.4 泄水孔层数设定

泄水孔的层数由于其工程实用性、成本效益及结构安全性，应设定一个上限值。在工程角度上，当泄水孔等数增加到一定程度后，额外增加层数对水压力的削减效果会显著减弱，此时继续增加层数无实际意义。

在查阅相关文献后[6]，我们将工程做了简化处理，即对墙高 H 的挡土墙，层数常常满足：

$$n \leq \lceil H/1.5 \rceil \tag{32}$$

在本题中，我们的取值为 4，即间距为 1.5m。

7.1.5 水压力分布公式

假设泄水孔将墙后水分段排出，每层孔对其控制范围内的水压力产生折减，墙后任意高度 z 处的水压力为：

$$p_w(z) = \gamma_w \cdot \left[z - \sum_{\substack{i=1 \\ h_i \leq z}}^{n} \lambda_i \cdot (z - h_i) \right] \tag{33}$$

其中，λ_i 表示单孔排水效率系数，反映第 i 层孔的排水能力：

$$\lambda_i = \frac{a_i}{s_x \cdot s_y} \cdot C$$

其中，a_i 表示第 i 层单个泄水孔的横截面积；s_x 和 s_y 分别表示泄水孔在水平和垂直方向的间距；C 表示修正系数（通常取 $0.5 \sim 1$，取决于排水系统的通畅程度）。p_{w0} 表示无泄水孔时墙后最大静水压力：

$$p_{w0} = \gamma_w H$$

并且：

- 当 $z < h_1$（低于第一层泄水孔）：无排水作用，水压力按线性分布 $p_w(z) = \gamma_w z$。
- 当 $h_i \leq z < h_{i+1}$（位于第 i 层与第 $i+1$ 层孔之间）：第 1 至第 i 层孔共同作用，水压力因排水折减，折减幅度与 λ_i 和孔位高度 h_i 相关。
- 当 $z \geq h_n$（高于最上层泄水孔）：所有泄水孔均起作用，水压力折减至最小。

因此我们可以得到总水压力与泄水孔位置的关系：

$$P_w = \int_0^H p_w(z)\,\mathrm{d}z \tag{34}$$

同时，为了后续表达，我们建立直角坐标系，如下图所示：

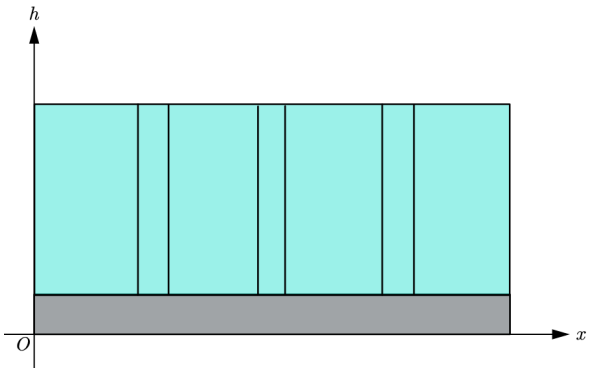


图 **14**　模型三坐标系图

7.2 模型求解

综合上述式子，结合之前的约束条件，我们可以得到以下公式：

$$max = k_w''' = w_1 k_s''' + w_2 k_t'''$$

$$s.t. \begin{cases} k_s''' = \dfrac{[\gamma_1 V + (4L+3t)E_{ay} + (4L+3t)E_{by}]\mu}{(4L+3t)E_{ax} + P_{wy}} \\ \quad + \dfrac{n\pi r L T_f + (4l+3t)(E_{bx} + P_{Wy}\mu)}{(4L+3t)E_{ax} + P_{wy}} \\ k_t''' = \dfrac{G'_{ax}X'_G + (P_{wy} + n\gamma_1\pi r L T_f)\left(B_L + \dfrac{b_2}{2}\right)}{(4L+3t)\left(E_{ax} - E_{bx}\right)\dfrac{H}{3} + P_{wx}\dfrac{H}{3}} \\ \quad + \dfrac{(4l+3t)\left(E_{ay}X_{ay} + E_{by}X_{by}\right)}{(4L+3t)\left(E_{ax} - E_{bx}\right)\dfrac{H}{3} + P_{wx}\dfrac{H}{3}} \\ \quad + \dfrac{n\pi r\dfrac{D^2}{2}T_f + \dfrac{b_2}{2} + 2T_h(y+d_1)}{(4L+3t)\left(E_{ax} - E_{bx}\right)\dfrac{H}{3} + P_{wx}\dfrac{H}{3}} \\ P_{wx} = P_w \cos\left(\beta - \dfrac{\pi}{2}\right), \quad P_{wy} = P_w \sin\left(\beta - \dfrac{\pi}{2}\right) \\ E_a = \dfrac{1}{2}\gamma'_2 H^2 K_a, \quad E_b = \dfrac{1}{2}\gamma'_2 H^2 K_b \\ p_w(z) = \gamma_w \cdot \left[z - \sum_{\substack{i=1 \\ h_i \le z}}^{n} \lambda_i \cdot (z - h_i)\right] \\ P_w = \int_0^H p_w(z)\,\mathrm{d}z \end{cases} \tag{35}$$

$$
s.t.\begin{cases}
x_{ij} \in [(j-1)(L+t)+R,\ j(L+t)-t-R] \\
d_1 \le h_i \le H - R - \sin\beta \\
H \le 30, \quad r \ge 0.3, \quad L \le \dfrac{H}{2} \\
t \in \left[\dfrac{L}{10}, \dfrac{L}{4}\right], \quad t \ge 0.3 \\
b = k_1(B_1 + B_2 + b_2) = k_3 L \\
b_1 \ge 0.2, \quad 4L + 3t \le 40 \\
B_1 \in \left[\dfrac{H}{4}, \dfrac{H}{2}\right], \quad B_2 \ge 0.5 \\
B_2 \in \left[\dfrac{H}{20}, \dfrac{H}{5}\right], \quad B_2 \ge 0.3 \\
b_2 = b_{21} + b_1 + b_{22}\dfrac{|\tan x + y|}{\sqrt{\tan^2\alpha + 1}} \ge 0.04 \\
\tan\alpha = -\dfrac{H}{b_{21}}, \quad \tan\beta = -\dfrac{H}{b_{22}} \\
\gamma_1 = 24, \quad \gamma_2 = 18, \quad \mu = 0.6, \quad \gamma_3 = 78.5 \\
\tau_t = 0.92\,\text{MPa}, \quad \tau_t = 30\,\text{kPa}, \quad V_{\max} = \dfrac{q\cdot}{2} \\
D \in \left[\dfrac{H}{5}, \dfrac{H}{2}\right], \quad L_1 \in \left[\dfrac{L}{5}, \dfrac{L}{3}\right] \\
M_{\max} = \dfrac{q \cdot L^2}{8}, \quad \sigma_{弯} = \dfrac{M_{\max}}{W}, \quad \tau = \dfrac{V_{\max}}{A_f} \\
q = \gamma \cdot h \cdot K_a, \quad W = \dfrac{L \cdot b_1^2}{6} \\
p_{\max} = \dfrac{G + E_{a,y}}{B}\left(1 + \dfrac{6e}{B}\right), \quad T_h \le \min(\pi d_a \tau_t, F_p + f_{地}) \\
f_{地} = [\gamma_1 V + \gamma_2 V_j + (E_{a,y} + E_{b,y})(4L + 3t)]\mu, \quad F_P = \pi r L_a \tau f
\end{cases} \tag{36}
$$

7.3 COBYLA 算法

由于上述式子较为繁琐，约束多为非线性、非凸，且目标函数（稳定性指标）与设计变量（泄水孔坐标、半径、层数）的关系复杂，难以用解析表达式完全刻画, 因此我们在具体求解过程中采用了 COBYLA 算法来简化运算。

Step1: 定义设计变量

将泄水孔的关键参数转化为优化变量向量 x：

$$
x = [r,\ h_1,\ x_{11},\ x_{12},\ ...,\ h_4,\ x_{41},\ x_{42},\ ...] \tag{37}
$$

其中：r 为泄水孔半径；h_i 为第 i 层泄水孔的高度；x_{ij} 为第 i 层第 j 个泄水孔的水平坐标。

Step2: 构建目标函数

以稳定性指标 k_w''' 为优化目标:

$$\min f(x) = -k_w'''(x) \tag{38}$$

Step3: 定义约束条件

将工程约束转化为不等式约束 $g(x) \leq 0$。其中位置约束：泄水孔需位于挡土墙内部（如水平坐标 $x_{ij} \in [0.44, 9.95]$，高度 $h_i \in [1.25, 4.94]$）；尺寸约束：半径 $r \geq 0.14\,\mathrm{m}$，层间距 $\geq 1.5\,\mathrm{m}$；结构约束：泄水孔边缘距挡土墙边界至少 0.04m（避免破坏墙体）。

Step4: 算法迭代优化

算法迭代大致上分成以下五种：初始点选择；线性近似；信赖域搜索；可行性检查；收敛判断。我们首先设置初始解，再在当前迭代图中构建目标函数和线性模型，在信赖域中求解线性子问题，从而不断迭代，在目标函数变化量小于阈值时，我们停止迭代。

7.4 求解结果

在模型求解释，我们通过 python 多次迭代，其中迭代图如下图所示：

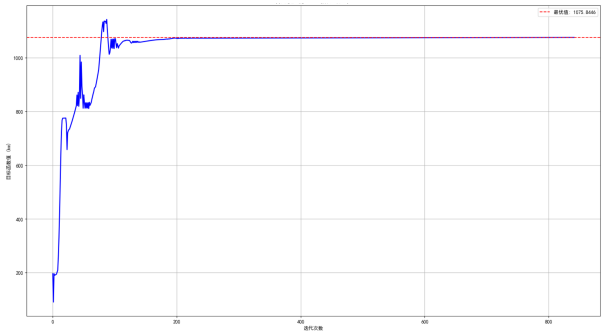


图 **15**　目标函数收敛过程

最终求得泄水孔共有 4 层,每个半径为 0.14 米,其坐标分别为 0.44,1.25/1.94/3.44/4.94),(4.15，1.25/1.94/3.44/4.94），（7.05，1.25/1.94/3.44/4.94），（9.95，1.25/1.94/3.44/4.94），单位为米。如下图所示：

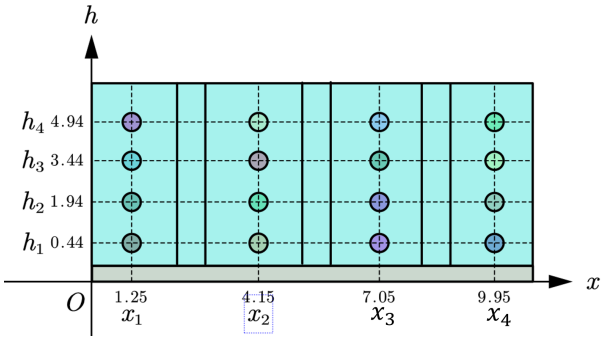


图 **16**　泄水孔位置示意图

八、 模型的评价

8.1 模型的优点

• 我们的模型具有较强的系统性和递进性，从模型一的单段无底桩到模型二的带底桩结构，到模型三的考虑积水，逐步引入变量，逻辑清晰；

• 我们核心模型基于库伦图压力理论计算土压力，结合多个评价指标，计算结果具有一定工程意义；

• 在针对钢筋位置、泄水孔等参数设计时，我们采用非线性规划建立最优化模型，模型兼顾安全性和经济性。

8.2 模型的缺点

• 由于题目所给条件较少，我们假设了较多条件，而实际工程中的复杂地形，很可能导致有相关压力计算的误差；

• 模型仅为静态分析，并未考虑到山体滑坡的动态因素影响，难以反映长期的稳定性。

8.3 模型的改进方向

• 模型下一步将引入动态分析，通过模拟雨水入渗过程，考虑土体抗剪强度的变化，从而预测工作性能；

• 通过引入多个工程参数，考虑复杂地质情况，采用更精准的理论分析，减少假设误差。

参考文献

[1] 张浩, 赵世杰. 高速公路复杂地质高边坡滑坡机理分析[J]. 居业, 2017(01):128-129.

[2] 欧剑. 高速公路组合扶壁式挡土墙力学特性研究[J]. 价值工程, 2025, 44(06):1-5.

[3] 芮瑞, 蒋旺, 徐杨青, 等. 刚性挡土墙位移模式对土压力的影响试验研究[J/OL]. 岩石力学与工程学报, 2023, 42(06):1534-1545. DOI: 10.13722/j.cnki.jrme.2022.0808.

[4] ALHAJJ CHEHADE H, DIAS D, SADEK M, et al. Seismic analysis of geosynthetic-reinforced retaining wall in cohesive soils[J]. Geotextiles and Geomembranes, 2019.

[5] 中华人民共和国建设部. GB 50010-2002 混凝土结构设计规范[M]. [出版地不详]: GB 50010-2002 混凝土结构设计规范, 2002.

[6] 王伦清, 李青. 高速公路路基挡土墙防护设计与施工方法[J]. 运输经理世界, 2025(08): 19-21.

附录 A　文件列表

文件名	功能描述
q1.py	问题一程序代码
q2.py	问题二程序代码
q3.py	问题三程序代码

附录 B　代码

q1.py

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import fsolve
# 设置字体以支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False


def calculate_partial_derivatives(function_expr, variables):
    symbols = sp.symbols(variables)
    derivatives = {}
    for var in symbols:
        derivatives[str(var)] = sp.diff(function_expr, var)
    return derivatives


def export_to_latex(function, derivatives, vars_list, filename
   ="derivatives.tex"):
    with open(filename, "w") as f:
        f.write(r"\documentclass{article}" + "\n")
        f.write(r"\usepackage{amsmath,amssymb}" + "\n")
        f.write(r"\begin{document}" + "\n\n")
```

```
25          f.write(r"\section*{原函数}" + "\n")
26          f.write(r"$$f\left(" + ", ".join(vars_list) + r"\right) =" + "\n")
27          f.write(sp.latex(function) + "$$\n\n")
28          f.write(r"\section*{偏导数}" + "\n")
29          for var, deriv in derivatives.items():
30              f.write(r"$$\frac{\partial f}{\partial " + var + "} =" + "\n")
31              f.write(sp.latex(deriv) + "$$\n\n")
32          f.write(r"\end{document}" + "\n")
33      print(f"LaTeX文档已保存至 {filename}")
34
35
36  def plot_function_and_derivatives(function, derivatives, vars_list, param_values,
37                                    var_ranges=None, save_figures=True):
38      if var_ranges is None:
39          var_ranges = {
40              vars_list[0]: (0, 2 * np.pi, 100),
41              vars_list[1]: (0, 2 * np.pi, 100)
42          }
43      var1, var2 = vars_list
44      x_min, x_max, x_num = var_ranges[var1]
45      y_min, y_max, y_num = var_ranges[var2]
46      x = np.linspace(x_min, x_max, x_num)
47      y = np.linspace(y_min, y_max, y_num)
48      X, Y = np.meshgrid(x, y)
49
50      func_lambda = sp.lambdify((sp.symbols(var1), sp.symbols(var2)),
51                                function.subs(param_values), "numpy")
52      deriv_lambdas = {
53          var: sp.lambdify((sp.symbols(var1), sp.symbols(var2)),
```

```
                        deriv.subs(param_values), "numpy")
        for var, deriv in derivatives.items()
    }

    try:
        Z = func_lambda(X, Y)
    except Exception as e:
        print(f"计算原函数时出错: {e}")
        Z = np.zeros_like(X)

    Z_derivs = {}
    for var, deriv_lambda in deriv_lambdas.items():
        try:
            Z_derivs[var] = deriv_lambda(X, Y)
        except Exception as e:
            print(f"计算偏导数 ∂f/∂{var} 时出错: {e}")
            Z_derivs[var] = np.zeros_like(X)

    plt.figure(figsize=(15, 10))
    ax1 = plt.subplot(1, 1, 1, projection='3d')
    surf1 = ax1.plot_surface(X, Y, Z, cmap=cm.coolwarm,
linewidth=0, antialiased=True)
    ax1.set_title('原函数')
    ax1.set_xlabel(var1)
    ax1.set_ylabel(var2)
    ax1.set_zlabel('f')
    plt.colorbar(surf1, ax=ax1, shrink=0.5, aspect=5)
    plt.show()

    for i, (var, Z_deriv) in enumerate(Z_derivs.items(), 2):
        ax = plt.subplot(1, 1, 1, projection='3d')
        surf = ax.plot_surface(X, Y, Z_deriv, cmap=cm.coolwarm
, linewidth=0, antialiased=True)
        ax.set_title(f'$\sigma$f/$\sigma${var}')
        ax.set_xlabel(var1)
```

```
87          ax.set_ylabel(var2)
88          ax.set_zlabel(f'$\sigma$f/$\sigma${var}')
89          plt.colorbar(surf, ax=ax, shrink=0.5, aspect=5)
90          plt.show()
91
92      plt.tight_layout()
93      if save_figures:
94          plt.savefig('function_and_derivatives.png', dpi=300,
    bbox_inches='tight')
95          print("图像已保存为 function_and_derivatives.png")
96      plt.show()
97
98
99  def find_partial_derivative_zeros(derivatives, vars_list,
    param_values, var_ranges, method='numerical'):
100     symbols = sp.symbols(vars_list)
101     zeros = {}
102
103     for var, deriv in derivatives.items():
104         print(f"\n求解 ∂f/∂{var} = 0 ...")
105         deriv_substituted = deriv.subs(param_values)
106         equation = sp.Eq(deriv_substituted, 0)
107
108         if method == 'symbolic':
109             try:
110                 sol = sp.solve(equation, sp.Symbol(var), dict=
    True)
111                 zeros[var] = sol
112                 print(f"∂f/∂{var} 的符号解：")
113                 for s in sol:
114                     sp.pprint(s)
115             except Exception as e:
116                 print(f"符号求解失败：{e}，自动切换为数值求解
    ")
117                 method = 'numerical'
```

```
118
119         if method == 'numerical':
120             var_sym = sp.Symbol(var)
121             other_var = [v for v in vars_list if v != var][0]
122             other_var_sym = sp.Symbol(other_var)
123             other_var_mid = (var_ranges[other_var][0] +
    var_ranges[other_var][1]) / 2
124
125             deriv_num = sp.lambdify(
126                 var_sym,
127                 deriv_substituted.subs({other_var_sym:
    other_var_mid}),
128                 'numpy'
129             )
130
131             var_min, var_max, _ = var_ranges[var]
132             search_points = np.linspace(var_min, var_max, 50)
133             roots = []
134
135             for x0 in search_points:
136                 try:
137                     root = fsolve(deriv_num, x0)[0]
138                     if (var_min <= root <= var_max) and np.
    isclose(deriv_num(root), 0, atol=1e-3):
139                         if not any(np.isclose(root, r, atol=1e
    -3) for r in roots):
140                             roots.append(round(root, 6))
141                 except:
142                     continue
143
144             zeros[var] = roots
145             print(f"∂f/∂{var} 的数值解（在范围内）：{roots}")
146             print(f"验证：∂f/∂{var} 在零点处的值：{[round(
    deriv_num(r), 9) for r in roots]}")
147
```

```
    return zeros


# 新增：绘制偏导数图像，辅助分析零点
def plot_partial_derivatives(derivatives, vars_list, param_values, var_ranges):
    """绘制偏导数函数图像，辅助分析零点位置"""
    for var, deriv in derivatives.items():
        var_sym = sp.Symbol(var)
        other_var = [v for v in vars_list if v != var][0]
        other_var_sym = sp.Symbol(other_var)
        other_var_mid = (var_ranges[other_var][0] + var_ranges[other_var][1]) / 2

        # 转换为数值函数
        deriv_num = sp.lambdify(
            var_sym,
            deriv.subs({**param_values, other_var_sym: other_var_mid}),
            'numpy'
        )

        # 创建绘图数据
        var_min, var_max, var_num = var_ranges[var]
        x = np.linspace(var_min, var_max, var_num)
        y = deriv_num(x)

        # 绘制函数
        plt.figure(figsize=(10, 6))
        plt.plot(x, y, label=f'$\sigma$f/$\sigma${var}')
        plt.axhline(y=0, color='r', linestyle='--', label='y=0')  # 添加y=0参考线
        plt.title(f'偏导数 $\sigma$f/$\sigma${var} 图像')
        plt.xlabel(var)
        plt.ylabel(f'∂f/∂{var}')
```

```
        plt.grid(True)
        plt.legend()
        plt.savefig(f'partial_derivative_{var}.png')
        plt.show()

        # 检查是否存在接近零的值
        close_to_zero = np.isclose(y, 0, atol=1e-3)
        if any(close_to_zero):
            zeros = x[close_to_zero]
            print(f"在 {var} 范围内发现接近零的点：{zeros}")
            print(f"对应的函数值：{y[close_to_zero]}")
        else:
            print(f"在 {var} 范围内未发现接近零的点")


# 新增：寻找函数的驻点（所有偏导数同时为零的点）
def find_critical_points(function, vars_list, param_values,
    var_ranges):
    """寻找函数的驻点（所有偏导数同时为零的点）"""
    symbols = sp.symbols(vars_list)

    # 计算所有偏导数
    partials = [sp.diff(function, s) for s in symbols]

    # 创建多变量数值函数
    def equations(x):
        # x 是包含所有变量值的数组
        subs_dict = {**param_values}
        for i, s in enumerate(symbols):
            subs_dict[s] = x[i]

        # 计算所有偏导数在给定点的值
        return [float(p.subs(subs_dict)) for p in partials]

    # 在变量范围内均匀采样初始点
```

```
    var1_min, var1_max, _ = var_ranges[vars_list[0]]
    var2_min, var2_max, _ = var_ranges[vars_list[1]]

    # 尝试多个初始点
    critical_points = []
    for x1 in np.linspace(var1_min, var1_max, 5):
        for x2 in np.linspace(var2_min, var2_max, 5):
            try:
                result = fsolve(equations, [x1, x2])
                # 验证解是否在范围内且满足方程
                if (var1_min <= result[0] <= var1_max) and (var2_min <= result[1] <= var2_max):
                    if np.allclose(equations(result), [0, 0], atol=1e-3):
                        # 去重
                        if not any(np.allclose(result, cp, atol=1e-3) for cp in critical_points):
                            critical_points.append(result)
                            print(f"找到驻点：{result}, 偏导数的值：{equations(result)}")
            except:
                continue

    return critical_points


if __name__ == "__main__":
    α, φ, y1, y2, H, B1, B2, b1, b2, L, t, d1, u, bt, btp, bt1 = sp.symbols(
        'α φ y1 y2 H B1 B2 b1 b2 L t d1 u bt btp bt1'
    )

    # 定义原函数
    C = y1 * (H * B1 * t / 2 + (B1 + B2 + b2) * d1 * (4 * L + 3 * t) + 1 / 2 * (b1 + b2) * (4 * L + 3 * t) * H)
```

```
 EA = 1 / 2 * y2 * H ** 2 * (4 * L + 3 * t) * (sp.cos(φ - btp)) ** 2 / (
          (sp.cos(btp)) ** 2 * sp.cos(btp + φ / 3) * (1 + sp.sqrt(
     (sp.sin(4* φ / 3 ) * sp.sin(φ - bt1)) / (sp.cos(btp + φ / 3) * sp.cos(btp - bt1)))) ** 2)
 EB = 1 / 2 * y2 * H ** 2 * (4 * L + 3 * t) * (sp.cos(φ - (α - sp.pi / 2))) ** 2 / (
          (sp.cos(α - sp.pi / 2)) ** 2 * sp.cos((α - sp.pi / 2) + φ / 3) * (1 + sp.sqrt(
     (sp.sin(4* φ / 3 ) * sp.sin(φ - bt1)) / (
               sp.cos((α - sp.pi / 2) + φ / 3) * sp.cos((α - sp.pi / 2) - bt1)))) ** 2)
 function1 = (u * (C + EA * sp.sin(bt - sp.pi / 2) + EB * sp.sin(α - sp.pi / 2)) + EB * sp.cos(α - sp.pi / 2)) / (EA * sp.cos(bt - sp.pi / 2))
 GG = 3 * y1 * B1 * H * t / 2 * ((2 * (- H / sp.tan(bt)) + B1) / 3 + b1 - H / sp.tan(α) + B2) + y1 * (
          B1 + B2 + b2) * (4 * L + 3 * t) * d1 * (
               B1 + B2 + b2) / 2 + (
               (2 * b1 + b2) * (- H / sp.tan(α)) + (b1 + 2 * b2) * (- H / sp.tan(α) + b1) / 3 / (
               b1 + b2) + B2) * H * (
               b1 + b2) * (4 * L + 3 * t) / 2
 function2 = (GG + (B2 + b2 + H / (3 * sp.tan(bt))) * EA * sp.sin(bt - sp.pi / 2) + (
          B2 - H / (3 * sp.tan(α))) * EB * sp.sin(α - sp.pi / 2)) / (
                        (EA * sp.cos(bt - sp.pi / 2) + EB * sp.cos(α - sp.pi / 2)) * H / 3)
 function = (function1 + function2) / 2
 vars_list = ['α', 'φ']
 partials = calculate_partial_derivatives(function, vars_list)

```

```
    # 输出原函数和偏导数
    print(f"原函数: f({', '.join(vars_list)}) =")
    sp.pprint(function)
    for var, deriv in partials.items():
        print(f"\n$\sigma$f/$\sigma${var} =")
        sp.pprint(deriv)

    # 导出为LaTeX
    export_to_latex(function, partials, vars_list)

    # 设置参数值和变量范围
    param_values = {
        y1: 24.0, y2: 18.0, H: 6.0, B1: 2.0, B2: 1.0,
        b1: 0.3, b2: 2.5, L: 2.5, t: 0.4,
        d1: 0.3, u: 0.6, bt: sp.pi / 2, btp: 0,
        bt1: 0
    }
    hk = 0.001
    var_ranges = {
        'α': (np.pi / 2, np.pi / 18 * 17, 150),
        'φ': (0 + hk, np.pi / 18 * 8, 150)
    }

    # 绘制函数图像
    plot_function_and_derivatives(function, partials,
vars_list, param_values, var_ranges)

    # 绘制偏导数图像，辅助分析零点
    plot_partial_derivatives(partials, vars_list, param_values
, var_ranges)

    # 尝试寻找驻点
    critical_points = find_critical_points(function, vars_list
, param_values, var_ranges)
    if critical_points:
```

```
        print(f"\n找到 {len(critical_points)} 个驻点:")
        for i, cp in enumerate(critical_points):
            print(f"驻点 {i + 1}: {vars_list[0]} = {cp[0]:.6f}, {vars_list[1]} = {cp[1]:.6f}")
    else:
        print("\n在指定范围内未找到驻点")

    # 尝试扩展变量范围
    expanded_var_ranges = {
        'α': (np.pi / 4, np.pi * 7 / 8, 200),  # 扩大af范围
        'φ': (-np.pi / 4, np.pi / 2, 200)  # 扩大fy范围并包含负值
    }

    print("\n尝试在扩展范围内寻找偏导数零点...")
    expanded_zeros = find_partial_derivative_zeros(
        derivatives=partials,
        vars_list=vars_list,
        param_values=param_values,
        var_ranges=expanded_var_ranges,
        method='numerical'
    )
```

q2.py

```
import sympy as sp
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import fsolve
# 设置字体以支持中文
plt.rcParams['font.sans-serif'] = ['SimHei']
plt.rcParams['axes.unicode_minus'] = False


```

```
def calculate_partial_derivatives(function_expr, variables):
    symbols = sp.symbols(variables)
    derivatives = {}
    for var in symbols:
        derivatives[str(var)] = sp.diff(function_expr, var)
    return derivatives


def export_to_latex(function, derivatives, vars_list, filename="derivatives.tex"):
    with open(filename, "w") as f:
        f.write(r"\documentclass{article}" + "\n")
        f.write(r"\usepackage{amsmath,amssymb}" + "\n")
        f.write(r"\begin{document}" + "\n\n")
        f.write(r"\section*{原函数}" + "\n")
        f.write(r"$$f\left(" + ", ".join(vars_list) + r"\right) =" + "\n")
        f.write(sp.latex(function) + "$$\n\n")
        f.write(r"\section*{偏导数}" + "\n")
        for var, deriv in derivatives.items():
            f.write(r"$$\frac{\partial f}{\partial " + var + "} =" + "\n")
            f.write(sp.latex(deriv) + "$$\n\n")
        f.write(r"\end{document}" + "\n")
    print(f"LaTeX文档已保存至 {filename}")


def plot_function_and_derivatives(function, derivatives, vars_list, param_values,
                                  var_ranges=None, save_figures=True):
    if var_ranges is None:
        var_ranges = {
            vars_list[0]: (0, 2 * np.pi, 100),
            vars_list[1]: (0, 2 * np.pi, 100)
```

```
        }
    var1, var2 = vars_list
    x_min, x_max, x_num = var_ranges[var1]
    y_min, y_max, y_num = var_ranges[var2]
    x = np.linspace(x_min, x_max, x_num)
    y = np.linspace(y_min, y_max, y_num)
    X, Y = np.meshgrid(x, y)

    func_lambda = sp.lambdify((sp.symbols(var1), sp.symbols(var2)),
                              function.subs(param_values), "numpy")
    deriv_lambdas = {
        var: sp.lambdify((sp.symbols(var1), sp.symbols(var2)),
                         deriv.subs(param_values), "numpy")
        for var, deriv in derivatives.items()
    }

    try:
        Z = func_lambda(X, Y)
    except Exception as e:
        print(f"计算原函数时出错: {e}")
        Z = np.zeros_like(X)

    Z_derivs = {}
    for var, deriv_lambda in deriv_lambdas.items():
        try:
            Z_derivs[var] = deriv_lambda(X, Y)
        except Exception as e:
            print(f"计算偏导数 ∂f/∂{var} 时出错: {e}")
            Z_derivs[var] = np.zeros_like(X)

    plt.figure(figsize=(15, 10))
    ax1 = plt.subplot(2, 2, 1, projection='3d')
    surf1 = ax1.plot_surface(X, Y, Z, cmap=cm.coolwarm,
```

```
    linewidth=0, antialiased=True)
    ax1.set_title('原函数')
    ax1.set_xlabel(var1)
    ax1.set_ylabel(var2)
    ax1.set_zlabel('f')
    plt.colorbar(surf1, ax=ax1, shrink=0.5, aspect=5)

    for i, (var, Z_deriv) in enumerate(Z_derivs.items(), 2):
        ax = plt.subplot(2, 2, i, projection='3d')
        surf = ax.plot_surface(X, Y, Z_deriv, cmap=cm.coolwarm
    , linewidth=0, antialiased=True)
        ax.set_title(f'$\sigma$f/$\sigma${var}')
        ax.set_xlabel(var1)
        ax.set_ylabel(var2)
        ax.set_zlabel(f'$\sigma$f/$\sigma${var}')
        plt.colorbar(surf, ax=ax, shrink=0.5, aspect=5)

    plt.tight_layout()
    if save_figures:
        plt.savefig('function_and_derivatives.png', dpi=300,
    bbox_inches='tight')
        print("图像已保存为 function_and_derivatives.png")
    plt.show()


def find_partial_derivative_zeros(derivatives, vars_list,
    param_values, var_ranges, method='numerical'):
    symbols = sp.symbols(vars_list)
    zeros = {}

    for var, deriv in derivatives.items():
        print(f"\n求解 ∂f/∂{var} = 0 ...")
        deriv_substituted = deriv.subs(param_values)
        equation = sp.Eq(deriv_substituted, 0)

```

```
        if method == 'symbolic':
            try:
                sol = sp.solve(equation, sp.Symbol(var), dict=True)
                zeros[var] = sol
                print(f"∂f/∂{var} 的符号解: ")
                for s in sol:
                    sp.pprint(s)
            except Exception as e:
                print(f"符号求解失败: {e}, 自动切换为数值求解")
                method = 'numerical'

        if method == 'numerical':
            var_sym = sp.Symbol(var)
            other_var = [v for v in vars_list if v != var][0]
            other_var_sym = sp.Symbol(other_var)
            other_var_mid = (var_ranges[other_var][0] + var_ranges[other_var][1]) / 2

            deriv_num = sp.lambdify(
                var_sym,
                deriv_substituted.subs({other_var_sym: other_var_mid}),
                'numpy'
            )

            var_min, var_max, _ = var_ranges[var]
            search_points = np.linspace(var_min, var_max, 50)
            roots = []

            for x0 in search_points:
                try:
                    root = fsolve(deriv_num, x0)[0]
                    if (var_min <= root <= var_max) and np.
```

```
    isclose(deriv_num(root), 0, atol=1e-3):
                        if not any(np.isclose(root, r, atol=1e
    -3) for r in roots):
                            roots.append(round(root, 6))
                except:
                    continue

            zeros[var] = roots
            print(f"∂f/∂{var} 的数值解（在范围内）：{roots}")
            print(f"验证：∂f/∂{var} 在零点处的值：{[round(
    deriv_num(r), 9) for r in roots]}")

     return zeros


# 新增：绘制偏导数图像，辅助分析零点
def plot_partial_derivatives(derivatives, vars_list,
    param_values, var_ranges):
     """绘制偏导数函数图像，辅助分析零点位置"""
     for var, deriv in derivatives.items():
         var_sym = sp.Symbol(var)
         other_var = [v for v in vars_list if v != var][0]
         other_var_sym = sp.Symbol(other_var)
         other_var_mid = (var_ranges[other_var][0] + var_ranges
    [other_var][1]) / 2

         # 转换为数值函数
         deriv_num = sp.lambdify(
             var_sym,
             deriv.subs({**param_values, other_var_sym:
    other_var_mid}),
             'numpy'
         )

         # 创建绘图数据
```

```
166         var_min, var_max, var_num = var_ranges[var]
167         x = np.linspace(var_min, var_max, var_num)
168         y = deriv_num(x)
169 
170         # 绘制函数
171         plt.figure(figsize=(10, 6))
172         plt.plot(x, y, label=f'$\sigma$f/$\sigma${var}')
173         plt.axhline(y=0, color='r', linestyle='--', label='y=0
    ')  # 添加y=0参考线
174         plt.title(f'偏导数 $\sigma$f/$\sigma${var} 图像')
175         plt.xlabel(var)
176         plt.ylabel(f'∂f/∂{var}')
177         plt.grid(True)
178         plt.legend()
179         plt.savefig(f'partial_derivative_{var}.png')
180         plt.show()
181 
182         # 检查是否存在接近零的值
183         close_to_zero = np.isclose(y, 0, atol=1e-3)
184         if any(close_to_zero):
185             zeros = x[close_to_zero]
186             print(f"在 {var} 范围内发现接近零的点：{zeros}")
187             print(f"对应的函数值：{y[close_to_zero]}")
188         else:
189             print(f"在 {var} 范围内未发现接近零的点")
190 
191 
192 # 新增：寻找函数的驻点（所有偏导数同时为零的点）
193 def find_critical_points(function, vars_list, param_values,
    var_ranges):
194     """寻找函数的驻点（所有偏导数同时为零的点）"""
195     symbols = sp.symbols(vars_list)
196 
197     # 计算所有偏导数
198     partials = [sp.diff(function, s) for s in symbols]
```

```
199
200     # 创建多变量数值函数
201     def equations(x):
202         # x 是包含所有变量值的数组
203         subs_dict = {**param_values}
204         for i, s in enumerate(symbols):
205             subs_dict[s] = x[i]
206 
207         # 计算所有偏导数在给定点的值
208         return [float(p.subs(subs_dict)) for p in partials]
209 
210     # 在变量范围内均匀采样初始点
211     var1_min, var1_max, _ = var_ranges[vars_list[0]]
212     var2_min, var2_max, _ = var_ranges[vars_list[1]]
213 
214     # 尝试多个初始点
215     critical_points = []
216     for x1 in np.linspace(var1_min, var1_max, 5):
217         for x2 in np.linspace(var2_min, var2_max, 5):
218             try:
219                 result = fsolve(equations, [x1, x2])
220                 # 验证解是否在范围内且满足方程
221                 if (var1_min <= result[0] <= var1_max) and (
    var2_min <= result[1] <= var2_max):
222                     if np.allclose(equations(result), [0, 0],
    atol=1e-3):
223                         # 去重
224                         if not any(np.allclose(result, cp,
    atol=1e-3) for cp in critical_points):
225                             critical_points.append(result)
226                             print(f"找到驻点：{result}，偏导数
    的值：{equations(result)}")
227             except:
228                 continue
229 
```

```
    return critical_points


if __name__ == "__main__":
    α, φ, y1, y2, H, B1, B2, b1, b2, L, t, d1, u, β, bt1, n, D, tf, r = sp.symbols(
        'α φ y1 y2 H B1 B2 b1 b2 L t d1 u β bt1 n D tf r'
    )

    # 定义原函数
    C = y1 * (H * B1 * t / 2 + (B1 + B2 + b2) * d1 * (4 * L + 3 * t) + 1 / 2 * (b1 + b2) * (
                4 * L + 3 * t) * H + n * D * sp.pi * r ** 2)
    EA = 1 / 2 * y2 * H ** 2 * (4 * L + 3 * t) * (sp.cos(φ - (β - sp.pi / 2))) ** 2 / (
            (sp.cos((β - sp.pi / 2))) ** 2 * sp.cos((β - sp.pi / 2) + φ / 3) * (1 + sp.sqrt(
        (sp.sin(4 * φ / 3) * sp.sin(φ - bt1)) / (
                    sp.cos((β - sp.pi / 2) + φ / 3) * sp.cos((β - sp.pi / 2) - bt1)))) ** 2)
    EB = 1 / 2 * y2 * H ** 2 * (4 * L + 3 * t) * (sp.cos(φ - (α - sp.pi / 2))) ** 2 / (
            (sp.cos(α - sp.pi / 2)) ** 2 * sp.cos((α - sp.pi / 2) + φ / 3) * (1 + sp.sqrt(
        (sp.sin(4 * φ / 3) * sp.sin(φ - bt1)) / (
                sp.cos((α - sp.pi / 2) + φ / 3) * sp.cos((α - sp.pi / 2) - bt1)))) ** 2)
    function1 = (u * (
                C + EA * sp.sin(β - sp.pi / 2) + EB * sp.sin(α - sp.pi / 2)) + n * sp.pi * r * D * tf + EB * sp.cos(
        α - sp.pi / 2)) / (EA * sp.cos(β - sp.pi / 2))
    GG = 3 * y1 * B1 * H * t / 2 * ((2 * (- H / sp.tan(β)) + B1) / 3 + b1 - H / sp.tan(α) + B2) + y1 * (
                B1 + B2 + b2) * (4 * L + 3 * t) * d1 * (B1 + B2 + b2) / 2 + (
```

```
                (2 * b1 + b2) * (- H / sp.tan(α)) + (b1 + 2 * b2) * (- H / sp.tan(α) + b1) / 3 / (
                    b1 + b2) + B2) * H * (b1 + b2) * (4 * L + 3 * t) / 2 + n * D * sp.pi * r ** 2 * y1 * (B2 + b2 / 2)
    function2 = (GG + (B2 + b2 + H / (3 * sp.tan(β))) * EA * sp.sin(β - sp.pi / 2) + (
            B2 - H / (3 * sp.tan(α))) * EB * sp.sin(α - sp.pi / 2) + n * sp.pi * r * D * tf * D / 2) / (
                        (EA * sp.cos(β - sp.pi / 2) + EB * sp.cos(α - sp.pi / 2)) * H / 3)
    function = (function1 + function2) / 2
    vars_list = ['α', 'β']
    partials = calculate_partial_derivatives(function, vars_list)

    # 输出原函数和偏导数
    print(f"原函数: f({', '.join(vars_list)}) =")
    sp.pprint(function)
    for var, deriv in partials.items():
        print(f"\n$\sigma$f/$\sigma${var} =")
        sp.pprint(deriv)

    # 导出为LaTeX
    export_to_latex(function, partials, vars_list)

    # 设置参数值和变量范围
    param_values = {
        y1: 24.0, y2: 18.0, H: 10.0, B1: 3.0, B2: 1.5,
        b1: 0.3, b2: 2.5, L: 4.0, t: 0.5,
        d1: 0.4, u: 0.6,tf:300000,
        bt1: 0, φ:sp.pi/6, r:0.4,
        n:3,D:3
    }
    hk = 0.001
    var_ranges = {
```

```
        'α': (np.pi / 2, 2.8, 150),
        'β': (np.pi / 2, 2.8, 150)
    }

    # 绘制函数图像
    plot_function_and_derivatives(function, partials, vars_list, param_values, var_ranges)

    # 绘制偏导数图像，辅助分析零点
    plot_partial_derivatives(partials, vars_list, param_values, var_ranges)

    # 尝试寻找驻点
    critical_points = find_critical_points(function, vars_list, param_values, var_ranges)
    if critical_points:
        print(f"\n找到 {len(critical_points)} 个驻点:")
        for i, cp in enumerate(critical_points):
            print(f"驻点 {i + 1}: {vars_list[0]} = {cp[0]:.6f}, {vars_list[1]} = {cp[1]:.6f}")
    else:
        print("\n在指定范围内未找到驻点")

    # 尝试扩展变量范围
    expanded_var_ranges = {
        'α': (np.pi / 4, np.pi * 7 / 8, 200),  # 扩大af范围
        'β': (-np.pi / 4, np.pi / 2, 200)  # 扩大fy范围并包含负值
    }

    print("\n尝试在扩展范围内寻找偏导数零点...")
    expanded_zeros = find_partial_derivative_zeros(
        derivatives=partials,
        vars_list=vars_list,
        param_values=param_values,
```

```
        var_ranges=expanded_var_ranges,
        method='numerical'
    )
```

q3.py

```
import numpy as np
from scipy.optimize import minimize
from scipy.integrate import quad  # 用于计算定积分
import matplotlib.pyplot as plt
import math

# 设置中文字体支持
plt.rcParams["font.family"] = ["SimHei", "WenQuanYi Micro Hei", "Heiti TC"]
plt.rcParams['axes.unicode_minus'] = False

# 定义常量（同原代码）
pi = math.pi
y1, y2, y3,yw,= 24, 18,10.2,9.8
B1, B2, b1 = 2, 1, 0.3
H, t, L, d1 = 6, 0.4, 2.5, 0.3
n, D, r = 3, 3, 0.4
φ, bt1 = pi / 6, 0
tt, tf, u = 0.92 * 1000000, 300000, 0.6
ffy = 2.9 * 1000000
d = 0.08
A = pi * d * d / 4
α, β = pi * 3 / 4, pi * 2 / 3
ni=4

# 存储迭代过程数据
iterations = []
objective_values = []  # 存储包含积分的kw值
variable_values = []

```

```
30
31  # --------------------------
32  # 定义定积分的被积函数（示例）
33  # 被积函数可以依赖优化变量x
34  # --------------------------
35  def integrand(t, x):
36      """
37      定积分的被积函数
38      t: 积分变量
39      x: 优化变量（x[0], x[1], x[2]）
40      """
41      # 示例：被积函数 = (t^2 + x[0]) * exp(-x[1]*t) + x[2]
42      # 这里的形式可以根据实际问题修改
43      rr=x[0]
44      nn=int(x[1])
45      kk=x[2:2+nn]
46      h=[0,0,0,0]
47      s=0
48      sig=0
49      s=pi*rr*rr
50      h[0]=kk[0]
51      for i in range(1,nn):
52          h[i]=kk[i]+h[i-1]
53      for i in range(nn):
54          sig+=s*nn/(L*H/math.sin(β)*0.02)*(t-h[i])
55      return yw*(t-sig)
56
57
58  # --------------------------
59  # 包含定积分的目标函数
60  # --------------------------
61  def objective(x):
62      # 1. 计算原代码中的基础项（保持不变）
63      integral_lower = 0  # 积分下限
64      integral_upper = H  # 积分上限
```

```
65      b2 = b1 - H / math.tan(α) - H / math.tan(β)
66      C = y1 * (H * B1 * t / 2 + (B1 + B2 + b2) * d1 * (4 * L +
    3 * t) + 1 / 2 * (b1 + b2) * (
67              4 * L + 3 * t) * H + n * D * math.pi * r ** 2)
68
69      EA = 1 / 2 * y2 * H ** 2 * (math.cos(φ - (β - math.pi / 2)
    )) ** 2 / (
70              (math.cos((β - math.pi / 2))) ** 2 * math.cos((β -
     pi / 2) + φ / 3) * (1 + math.sqrt(
71          (math.sin(4 * φ / 3) * math.sin(φ - bt1)) / (
72                  math.cos((β - pi / 2) + φ / 3) * math.cos((β -
     pi / 2) - bt1)))) ** 2)
73
74      EB = 1 / 2 * y2 * H ** 2 * (math.cos(φ - (α - pi / 2))) **
     2 / (
75              (math.cos(α - pi / 2)) ** 2 * math.cos((α - pi /
    2) + φ / 3) * (1 + math.sqrt(
76          (math.sin(4 * φ / 3) * math.sin(φ - bt1)) / (
77                  math.cos((α - math.pi / 2) + φ / 3) * math.cos
    ((α - pi / 2) - bt1)))) ** 2)
78      integral_result, integral_error = quad(integrand,
    integral_lower, integral_upper, args=(x,))
79      Pw=integral_result*(4 * L + 3 * t)
80      function1 = (u * (
81              C + EA * math.sin(β - math.pi / 2)  * (4 * L + 3 *
     t)+ EB * math.sin(
82          α - math.pi / 2) * (4 * L + 3 * t)+ Pw * math.sin(α -
    math.pi / 2) * (4 * L + 3 * t)) + n * math.pi * r * D * tf +
     EB * math.cos(
83          α - math.pi / 2) ) / (EA * math.cos(β - math.pi / 2) *
     (4 * L + 3 * t)+Pw*math.cos(β - math.pi / 2) * (4 * L + 3 *
     t))
84
85      GG = 3 * y1 * B1 * H * t / 2 * ((2 * (- H / math.tan(β)) +
     B1) / 3 + b1 - H / math.tan(α) + B2) + y1 * (
```

```
            B1 + B2 + b2) * (4 * L + 3 * t) * d1 * (B1 + B2 + b2) / 2 + (
                (2 * b1 + b2) * (- H / math.tan(α)) + (b1 + 2 * b2) * (- H / math.tan(α) + b1) / 3 / (
                b1 + b2) + B2) * H * (b1 + b2) * (4 * L + 3 * t) / 2 + n * D * pi * r ** 2 * y1 * (B2 + b2 / 2)

    function2 = (GG + (B2 + b2 + H / (3 * math.tan(β))) * EA * math.sin(β - math.pi / 2) + (
            B2 - H / (3 * math.tan(α))) * EB * math.sin(α - pi / 2) + n * math.pi * r * D * tf * D / 2 ) / (
                        (EA * math.cos(β - math.pi / 2)* H / 3 - EB * math.cos(α - math.pi / 2)) * H / 3 + Pw * math.cos(β - math.pi / 2)* H / 3)

    # 2. 计算定积分（核心修改：添加积分项）
    # 积分上下限可以是常数，也可以依赖x（示例中为0到10）
    integral_lower = 0  # 积分下限
    integral_upper = H  # 积分上限
    # 调用quad计算定积分，args=(x,)表示将x作为额外参数传入被积函数
    integral_result, integral_error = quad(integrand, integral_lower, integral_upper, args=(x,))
    # integral_error是积分的数值误差，通常可忽略

    # 3. 定义包含积分的目标函数kw（示例：原kw + 积分结果的加权值）
    # 具体组合方式根据实际问题调整
    kw = 0.5 * (function1 + function2)

    # 记录迭代数据
    current_iter = len(iterations)
    iterations.append(current_iter)
    objective_values.append(kw)  # 存储包含积分的kw值
    variable_values.append(x.copy())
```

```
111
112     return -kw  # 最小化负的kw（等价于最大化kw）
113
114
115 # 约束条件（同原代码）
116 def constraint1(x):
117     rr = x[0]
118     nn = int(x[1])
119     kk = x[2:2 + nn]
120     h = [0, 0, 0, 0]
121     s = 0
122     sig = 0
123     s = pi * rr * rr
124     h[0] = kk[0]
125     if h[0]<rr+d1:
126         return h[0]-rr-d1
127     for i in range(1, nn):
128         h[i] = kk[i] + h[i - 1]
129         if h[i]<rr:
130             return h[i]-rr
131         if h[i]>H:
132             return H-h[i]
133     return (L*H/math.sin(β)*0.02)-s*nn
134
135
136 # 初始猜测值、约束和边界（同原代码）
137 x0 = [0.166,4.2,3,1.6,1.6,1.6]
138
139 constraints = [
140     {'type': 'ineq', 'fun': constraint1}
141 ]
142
143 bounds_constraints = [
144     {'type': 'ineq', 'fun': lambda x: x[0] - 0.05},
145     {'type': 'ineq', 'fun': lambda x: math.sqrt((L*H/math.sin(
```

```
    β)*0.02)/math.pi/x[1]) - x[0]},
    {'type': 'ineq', 'fun': lambda x: x[1] -1},
    {'type': 'ineq', 'fun': lambda x: 4.9 - x[1]},
    {'type': 'ineq', 'fun': lambda x: x[3] - 1.5},
    {'type': 'ineq', 'fun': lambda x: H/int(x[1]) - x[3]},
    {'type': 'ineq', 'fun': lambda x: x[4] - 1.5},
    {'type': 'ineq', 'fun': lambda x: H / int(x[1]) - x[4]},
    {'type': 'ineq', 'fun': lambda x: x[5] - 1.5},
    {'type': 'ineq', 'fun': lambda x: H / int(x[1]) - x[5]},
    {'type': 'ineq', 'fun': lambda x: x[2] - 0},
    {'type': 'ineq', 'fun': lambda x: H / int(x[1]) - x[2]}
]

all_constraints = constraints + bounds_constraints

# 求解优化问题
result = minimize(
    objective,
    x0,
    method='COBYLA',
    constraints=all_constraints,
    options={
        'disp': True,
        'maxiter': 10000000,
        'rhobeg': 0.1,
        'tol': 1e-6
    }
)

# 输出结果
if result.success:
    optimal_x = result.x
    optimal_value = -result.fun
    print("优化成功！")
    print(f"最优解：R = {optimal_x[0]:.4f}, m = {optimal_x
```

```
    [1]:.4f}")
180     x=optimal_x
181     rr = x[0]
182     nn = int(x[1])
183     kk = x[2:2 + nn]
184     h = [0, 0, 0, 0]
185     s = 0
186     sig = 0
187     s = pi * rr * rr
188     h[0] = kk[0]
189     print(h[0])
190     for i in range(1, nn):
191         h[i] = kk[i] + h[i - 1]
192         print(h[i])
193     print(f"最优值: kw = {optimal_value:.4f}")
194 else:
195     print("优化失败, 原因: ", result.message)
196     print("当前解: X = {0:.4f}, Y = {1:.4f}, Z = {2:.4f}".
    format(*result.x))
197     print("当前目标函数值: kw = {0:.4f}".format(-result.fun))
198
199 # 绘制迭代图
200 plt.figure(figsize=(15, 10))
201
202 # 目标函数值（含积分）收敛过程
203 plt.subplot(1, 1, 1)
204 plt.plot(iterations, objective_values, 'b-', linewidth=2)
205 plt.xlabel('迭代次数')
206 plt.ylabel('目标函数值 (kw)')
207 plt.title('含定积分的目标函数收敛过程')
208 plt.grid(True)
209 if result.success:
210     plt.axhline(y=optimal_value, color='r', linestyle='--',
    label=f'最优值: {optimal_value:.4f}')
211     plt.legend()
```

```
212 plt.tight_layout()
213 plt.show()
```