



中国研究生创新实践系列大赛
“华为杯”第十七届中国研究生数学建模竞赛

学　　校		北京理工大学
参赛队号		**20100070005**
队员姓名	**1.**	高冠强
	2.	鲁　赛
	3.	郭　苗

中国研究生创新实践系列大赛
“华为杯”第十七届中国研究生
数学建模竞赛

题 目 飞行器质心平衡供油策略优化

摘 要：

在任务执行过程中，飞行器的质心变化对飞行器的控制有着重要的影响，各个油箱内油量的分布和供油策略将导致飞行器质心的变化，进而影响到对飞行器的控制。因此，制定各油箱的供油策略是这类飞行器控制的一项重要任务。本文针对题目中飞行器质心平衡供油策略优化问题建立了数学模型，并提出了求解方法。通过对题目中给定的数据集进行测试，验证了本文所提出的求解方法的有效性，可以高效地制定出优质的供油策略。

对于问题一，建立了飞行器（载油）质心与飞行器俯仰角、供油策略、初始油量以及油箱尺寸等因素的数学关系模型，并根据给定的测试数据求解出了飞行器质心在飞行器坐标系下的位置，给出了此次任务执行过程中的质心变化曲线，飞行器质心距飞行器坐标系原点的最大欧式距离为 1.109137m。

对于问题二，在满足约束的前提下，要求最小化飞行器每一时刻的质心位置与理想质心的欧氏距离的最大值。提出了**基于供油模式的混合超启发式(hybrid hyper-heuristic)方法**，求解出了高质量的供油策略。首先，本文根据问题特征归纳出了 35 种供油模式作为基础，每种模式包括采用的供油箱及其相应的供油速度。其次，本文将总供油周期分解为 120 个等于 60s 最小供油时间窗口。在最小供油时间窗口内，采用的供油模式不发生变化。由此，该问题从一个超高纬度的连续变量优化问题，转换为一个 120 维的离散变量优化问题。当给定供油模式的选择规则后，可以以最小供油时间窗口为周期，通过滚动时域的方法产生唯一的供油模式序列及每秒供油方案。然而，供油模式的选择规则因需要考虑当前油量、未来供油需求、采用供油模式后会产生的最大瞬时偏差等多种因素，所以人为手动设计选择规则往往不能得到很好的效果。本文采用了基于**遗传编程(genetic programming)的超启发式方法优化供油模式的选择规则**，在供油模式生成的启发式解空间上优化供油方案来降低与理想质心的最大瞬时偏差。该超启发式方法可以更进一步地降低问题的维度和约束处理难度。最后，在遗传编程超启发式方法生成供油方案的基础上，本文又采用了**差分进化方法（Differential Evolution）来优化每一个时刻的供油速度**。在附件 3 所给数据下，飞行器瞬时质心与理想质心距离的最大值为 **0.10233m**，4 个主油箱的总供油量为 **6483.8242kg**。同时，本文对自动设计的超启发式规则函数进行了分析与论证。

对于问题三，在问题一的基础上假定初始油量未定，要求最小化飞行器每一时刻的质心位置与理想质心的欧氏距离的最大值，并且要求任务结束时 6 个油箱剩余燃油总量至少 $1m^3$。本文设计了**基于差分进化的多阶段优化方法**，制定出了较优的油箱供油方案以及初始载油量。在第一阶段中，采用差分进化方法来优化初始载油量和供油方案。在第一阶段的差分进化方法中，种群中的个体被表示为 6 维向量，每一个维度上的数值用来表示油箱初始油量。差分进化中的个体解码方法以问题二中的遗传编程获取的供油模式的选择规则为基础，以个体所表达的初始载油量为起始条件，以最小供油时间窗口为周期滚动生成供

油方案并计算该供油方案下瞬时最大偏差作为个体的评价指标。由此，建立了关于种群中的个体与初始油箱容量、供油方案和目标函数之间的对应关系。接着，第一阶段差分进化算法进行差分、变异、选择等操作直至满足终止条件后输出最佳个体所表达的初始载油量和供油方案。在第二阶段，在第一阶段获取的供油方案的基础上，本文又采用了差分进化方法来优化每一个时刻的供油速度。在附件 4 所给数据下，通过差分进化算法优化得到飞行器各油箱初始载油量分别为 **0.23665 m^3、1.91664 m^3、2.09303 m^3、2.54540 m^3、2.68764 m^3、0.79375 m^3**，总初始载油量为 10.27311m^3，飞行器瞬时质心与理想质心距离的最大值为 **0.038803m**，4 个主油箱的总供油量为 **6851.9747kg**。

对于问题四，在问题一的基础上考虑飞行器俯仰角随时间变化的情况，要求最小化飞行器每一时刻的质心位置与飞行器（不载油）质心的欧氏距离的最大值。与问题二类似，本文以在启发式解空间内搜索为出发点，提出了**基于供油模式的混合超启发式方法**，求解出了高质量的供油方案。不同于问题二，该问题中供油模式的选择规则不仅仅需要考虑当前油量、未来供油需求等因素，还需要额外考虑当前俯仰角度和未来俯仰角度等因素。因此，本文在考虑俯仰角度的基础上重新设计了遗传编程的特征集合。在新的特征集合上，基于遗传编程的超启发式方法优化生成了高效的供油模式选择规则，有效地降低了最大瞬时偏差。最后，在遗传编程超启发方法生成供油方案的基础上，本文又采用了差分进化方法来优化每一个时刻的供油速度，制定出了较优的油箱供油策略。在附件 5 所给数据下，飞行器瞬时质心与飞行器（不载油）质心距离的最大值为 **0.13798m**，4 个主油箱的总供油量为 **7089.8452kg**。同时，本文对自动设计的超启发式规则函数进行了分析与论证。

关键词：质心计算、超启发式方法、遗传编程、差分进化、多阶段优化

目录

一、问题重述

1.1 问题背景

随着科学技术的不断进步，飞行器的种类用途也越来越丰富。机载燃油系统在现代飞行器各个系统中的地位越来越重要，其基本功能是在整个飞行包线范围内与正常工作过程中，以及系统故障状态下，用合适的压力和温度为动力系统（发动机）提供连续的燃油资源[1]。下图为机载燃油管理系统。

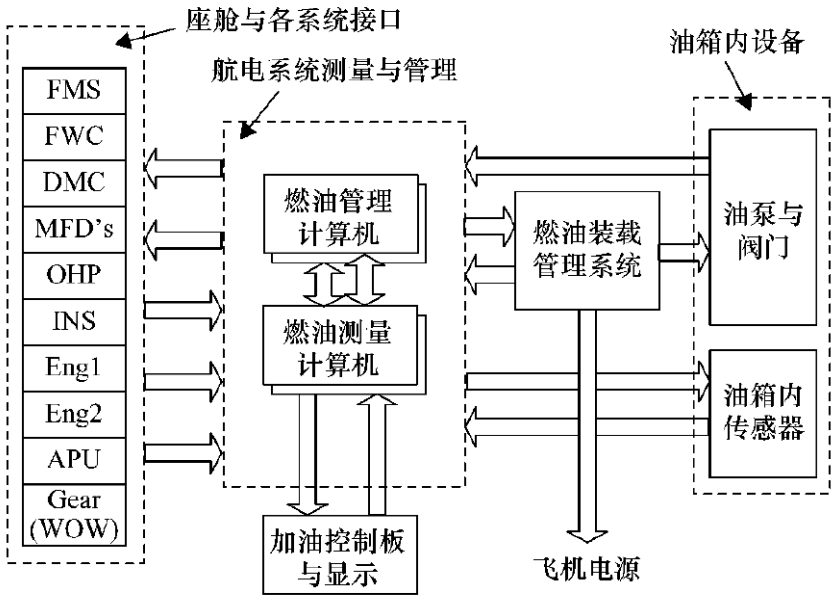


图 1-1 燃油管理系统

某一类飞行器携带有多个油箱，在飞行过程中，通过若干个油箱的联合供油以满足飞行任务要求和发动机工作需求。在任务执行过程中，飞行器的质心变化对飞行器的控制有着重要的影响，各个油箱内油量的分布和供油策略将导致飞行器质心的变化，进而影响到对飞行器的控制。因此，制定各油箱的供油策略是这类飞行器控制的一项重要任务。

1.2 待解决的问题

基于附件 1 提供的数据，包括飞行器基本参数如油箱中心位置、飞行器（不载油）质量等，以及题目给定的供油约束等，建立飞行器质心平衡供油策略优化问题的数学模型，并在给定的数据集下，求解相应问题。

问题一：针对附件 2 的数据，包括飞行器 6 个油箱供油速度以及飞行器飞行过程中的俯仰角变化数据，求解出此次执行任务过程中飞行器在飞行器坐标系下的质心位置。

问题二：针对附件 3 的数据，包括飞行器计划耗油速度数据，与飞行器在飞行器坐标系下的理想质心位置数据，制定出满足约束条件时飞行器油箱的供油策略，且使得飞行器每一时刻的质心位置与理想质心位置的欧氏距离的最大值达到最小，并讨论算法的有效性和复杂度。

问题三：针对附件 4 的数据，包括飞行器计划耗油速度数据，与飞行器在飞行器坐标系下的理想质心位置数据，制定出此次任务满足约束条件的 6 个油箱初始油量及供油策略，使得本次任务结束时 6 个油箱剩余燃油总量至少 $1m^3$，并且飞行器每一时刻的质心位置与

理想质心位置的欧氏距离的最大值达到最小，并讨论算法的有效性和复杂度。

问题四：针对附件 5 的数据，包括飞行器俯仰角的变化数据和耗油速度数据，制定出此次任务的油箱供油策略，使得飞行器瞬时质心与飞行器(不载油)质心的最大距离达到最小，并讨论算法的有效性和复杂度。

二、模型假设及符号说明

2.1 模型假设

（1）假设一：油箱均为长方体且固定在飞行器内部，飞行器的长、宽、高的三个方向与飞行器坐标系的 x,y,z 轴三个方向平行。

（2）假设二：第 i 个油箱的供油速度上限为，U_i ($U_i>0$)，i=1,2,…,6。每个油箱一次供油的持续时间不少于 60 秒。

（3）假设三：主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别为油箱 2 和油箱 5 供油，不能直接向发动机供油。

（4）假设四：由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油。

（5）假设五：飞行器在执行任务过程中，各油箱联合供油的总量应至少满足发动机的对耗油量的需要（若某时刻供油量大于计划耗油量，多余的燃油可通过其它装置排出飞行器），发动机在每个时刻的耗油速度可用一条耗油速度曲线表示。

（6）假设六：飞行器在飞行过程中可能发生姿态改变，主要是飞行航向上的上下俯仰或左右偏转。为简化问题，假设本题目中飞行器姿态的改变仅考虑平直飞与俯仰情况。飞行器的俯仰将导致各油箱相对地面的姿态发生倾斜，在重力作用下，油箱的燃油分布也随之发生变化，从而使得飞行器质心发生偏移。

2.2 符号说明

本文中所用符号的说明如下表所示。

表 2-1 符号说明

符号定义	符号
第i个油箱内部长、宽、高(m)	a_i,b_i,c_i
飞行器（不载油）质心	$\vec{c}_0$（0，0，0）
每一时刻 t 飞行器（不载油）质心	$\vec{c}_1(t)$
每一时刻 t 飞行器（不载油）理想质心	$\vec{c}_2(t)$
飞行器（不载油）总质量(kg)	M
第 i 个油箱初始燃油体积(m^3)	O_i

燃料的密度(kg/m^3)	ρ
飞行器 t 时刻的俯仰角(度)	$\theta(t)$
第 i 个油箱 t 时刻的供油速度(kg/s)	$v_i(t)$
每时一刻 t 飞行器（载油）总质量(kg)	$Q(t)$
第 i 个油箱的中心位置(m)	x_i, y_i, z_i
每一时刻 t 第 i 个油箱燃油的质量(kg)	$q_i(t)$
每一时刻 t 第 i 个油箱燃油的质心位置(m)	$p_i(t)$
第 i 个油箱的供油速度上限(kg/s)	U_i
6 个油箱剩余燃油总量(m^3)	L
每一时刻 t 发动机耗油速度(kg/s)	R(t)
第 i 油箱在飞行过程中每次供油持续时间的最小值	$t_{oil}(i)$
飞行器飞行总时长(s)	T

三、问题一的模型建立与求解

3.1 问题分析

问题一需要给出该飞行器在此次任务执行过程中的质心变化曲线，即计算出其质心在飞行器坐标系下的位置数据。本问题核心和难点主要集中在建立油箱质心随油箱油量及飞行器俯仰角变化的模型，飞行器倾角的不同和油量剩余的多少都会影响到燃油在油箱几何尺寸约束下的质心位置。此题目求解质心的实质，是给出质心位置与飞行器各参素的关系模型。质心是质点系质量分布的平均位置，飞行器的系统可看作由 6 个油箱的燃油和飞行器自身组成,每一部分的质量为 m_i,每一部分相对于飞行器坐标系的矢径为 r_i,则飞行器系统的质心位置 r 为：

$$r = \frac{\sum_i^n m_i r_i}{\sum_i^n m_i}, n = 7 \qquad (3\text{-}1)$$

因此，质心位置与飞行器各参素的关系模型中，考虑的变量有：

（1） 每一时刻飞行器各油箱内燃油的质量；

（2） 每一时刻飞行器各油箱内燃油的质心位置；

（3） 飞行器（不载油）的质量；

（4） 飞行器（不载油）的质心位置。

以上变量中（1）可通过计算初始油量与消耗油量的差值而得到，需要用到的参数有：

每一时刻飞行器各油箱的供油速度、燃油的密度、飞行器各油箱的初始油量；以上变量中（2）可通过分情况计算不同俯仰角及剩余油量情况下的燃油质心，需要的参数有：每一时刻飞行器俯仰角、每一时刻飞行器各油箱的供油速度、燃油的密度、飞行器各油箱的初始油量、每个油箱的中心位置、每个油箱内部的长、宽、高；以上变量中（3）（4）已知，飞行器（不载油）的质心位置在飞行器坐标系下即为（0，0，0）。因此，本题求解飞行器质心的关键在于计算出每个油箱内燃油的质心，即变量（2）。

3.2 质心计算模型

在飞行器坐标系下，飞行器的质心为：

$$\vec{c}_1(t)=\sum_{i=1}^{6}\frac{\vec{p}_i(t)q_i(t)+\vec{c}_0M}{Q(t)} \tag{3-2}$$

其中，飞行器的总质量等于飞行器（不载油）总质量加上各油箱内燃油的总质量：

$$Q(t)=M+\sum_{i=1}^{6}q_i(t) \tag{3-3}$$

$$\begin{aligned}
&q_i(t)=O_i\rho-\int_0^t v_i(t)dt+,i=1,3,4,6\\
&q_i(t)=O_i\rho-\int_0^t v_i(t)dt+\int_0^t v_1(t)dt,i=2\\
&q_i(t)=O_i\rho-\int_0^t v_i(t)dt+\int_0^t v_6(t)dt,i=5
\end{aligned} \tag{3-4}$$

由于飞行器只有俯仰角变化，油箱内燃油的质心位置的 y 轴坐标位置与油箱的中心位置相同。以飞行器坐标系的 $x0z$ 平面（飞行器的纵剖面）为切平面绘制油箱剖面图，并分情况讨论不同情形下的油箱燃油质心。

当第 i 个油箱的燃油为 q_i kg 时，显然可得燃油在油箱的纵剖面的截面积（平方米）为：

$$S=\frac{q_i}{\rho b_i} \tag{3-5}$$

3.2.1 飞行器俯仰角为 0 的质心计算模型

当飞行器俯仰角为 $\theta=0$ 时，油箱内燃油的质心位置的 x 轴坐标位置与油箱的中心位置相同。燃油的高度为：

$$h=\frac{S}{a_i} \tag{3-6}$$

则，燃油质心相对于油箱中心位置的差值为：

$$h'=\frac{S}{2a_i}-\frac{c_i}{2} \tag{3-7}$$

那么燃油的质心绝对坐标为

$$\vec{p}_i=(x_i,y_i,\frac{S}{2a_i}-\frac{c_i}{2}+z_i) \tag{3-8}$$

3.2.2 飞行器俯仰角大于 0 的质心计算模型

当飞行器俯仰角为$\theta>0$时，可分四种情况：

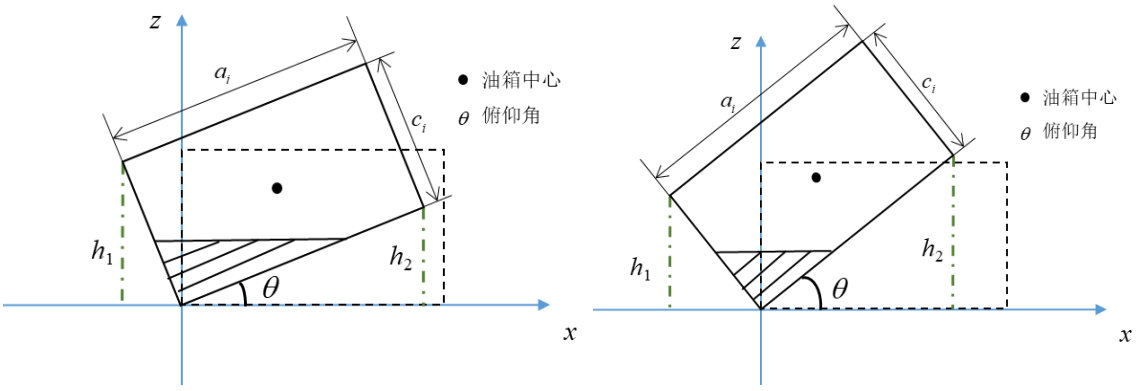
情况一：

设飞行器俯仰角为θ，当燃油的截面积呈三角形，即符合情况：

$$S \le \min\{\frac{a_i^2 \tan\theta}{2}, \frac{c_i^2}{2\tan\theta}\} \tag{3-9}$$

此情况依据可通过以下过程分析：

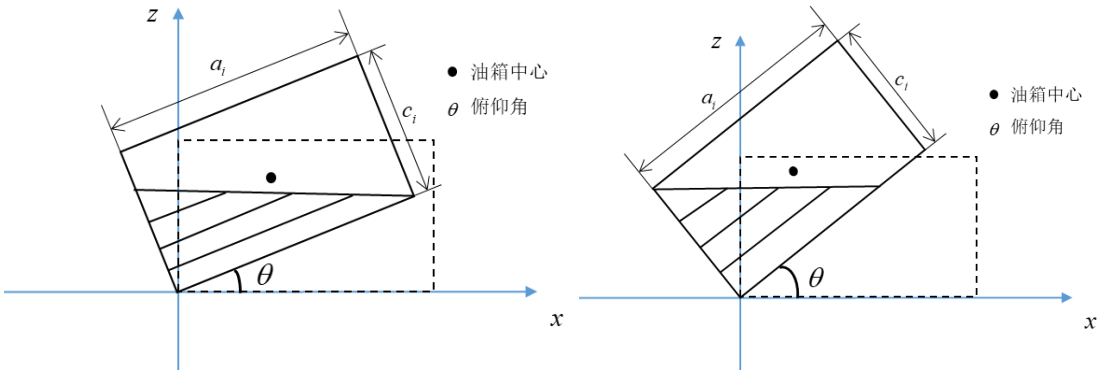
如下图（a）（b）所示，存在两种符合当$h_1>h_2$时，俯仰角满足$\tan\theta<c_i/a_i$；反之如上图（b），俯仰角满足$\tan\theta \ge c_i/a_i$。



(a) (b)

图 3-1 燃油截面积为三角形（情况一）

当满足$h_1>h_2$和$h_1 \le h_2$时，横截面最大时分别为以下两种情况：



(a) (b)

图 3-2 燃油截面积为三角形时的横截面积最大值（情况一）

这两种情况对应的面积大小为：

$$S_{11}=\frac{a_i^2\tan\theta}{2} \quad 和 \quad S_{12}=\frac{c_i^2}{2\tan\theta} \tag{3-10}$$

因此，燃油的截面为三角形，即符合情况：

$$S \le \min\{\frac{a_i^2 \tan\theta}{2}, \frac{c_i^2}{2\tan\theta}\} \tag{3-11}$$

这种分类情况下的燃油质心求解方法如下：

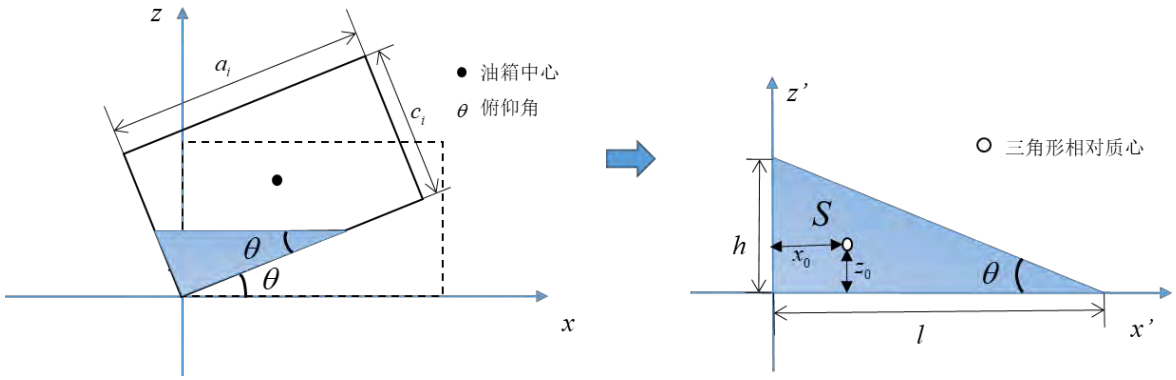


图 3-3 燃油（三角形部分）相对质心位置（相对于油箱的左下顶点）

首先，求解三角形自身相对于油箱的相对质心位置，以油箱左下顶点为原点，长边为 x 轴方向，高边为 z 轴方向，如上图所示。联立以下方程求解：

$$S = \frac{hl}{2} \tag{3-12}$$

$$h = l\tan\theta$$

可得油箱中燃油对应的三角形截面积的长和高如下：

$$l = \sqrt{\frac{2S}{\tan\theta}}, h = \sqrt{2S\tan\theta} \tag{3-13}$$

进一步地，油箱中燃油对于油箱的相对质心位置为：

$$x_o = \frac{1}{3}\sqrt{\frac{2S}{\tan\theta}}, z_0 = \frac{1}{3}\sqrt{2S\tan\theta} \tag{3-14}$$

又因为油箱是长方体结构，油箱中燃油对于油箱的相对质心位置对于油箱中心位置的位置差为

$$\Delta x = x_0 - \frac{a_i}{2}, \Delta z = z_0 - \frac{c_i}{2} \tag{3-15}$$

$$\Delta x = \frac{1}{3}\sqrt{\frac{2S}{\tan\theta}} - \frac{a_i}{2}, \Delta z = \frac{1}{3}\sqrt{2S\tan\theta} - \frac{c_i}{2} \tag{3-16}$$

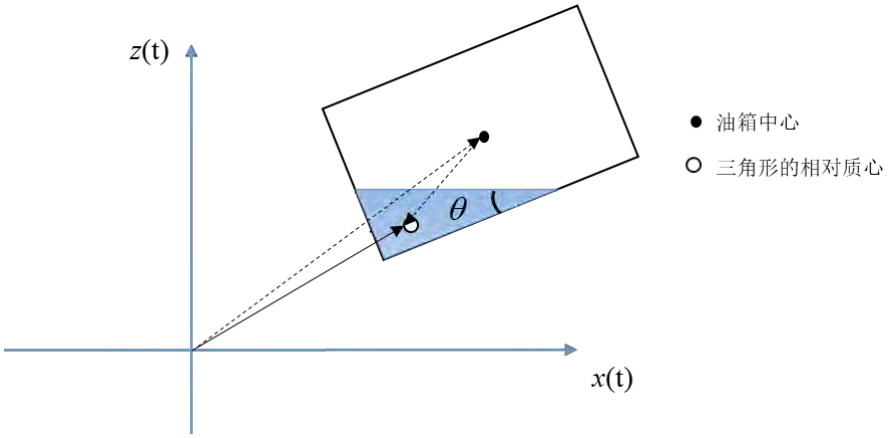


图 3-4 燃油（三角形部分）绝对坐标位置（对于飞行器坐标系）

如上图，中心位置的绝对坐标（对于飞行器坐标系）为（x_i, y_i, z_i），又因为前述油箱内燃油的质心位置的 y 轴坐标位置与油箱的中心位置相同，则燃油质心的绝对坐标为

$$\vec{p}_i = (\Delta x + x_i, y_i, \Delta z + z_i) \tag{3-17}$$

$$\vec{p}_i = (\frac{1}{3}\sqrt{\frac{2S}{\tan\theta}} - \frac{a_i}{2} + x_i, y_i, \frac{1}{3}\sqrt{2S\tan\theta} - \frac{c_i}{2} + z_i) \tag{3-18}$$

情况二：

设飞行器俯仰角为 θ，如下图所示，当燃油未填满油箱的部分的截面为三角形时，类似于情况一的推导，即符合情况：

$$S \geq a_i c_i - \min\{\frac{a_i^2 \tan\theta}{2}, \frac{c_i^2}{2\tan\theta}\} \tag{3-19}$$

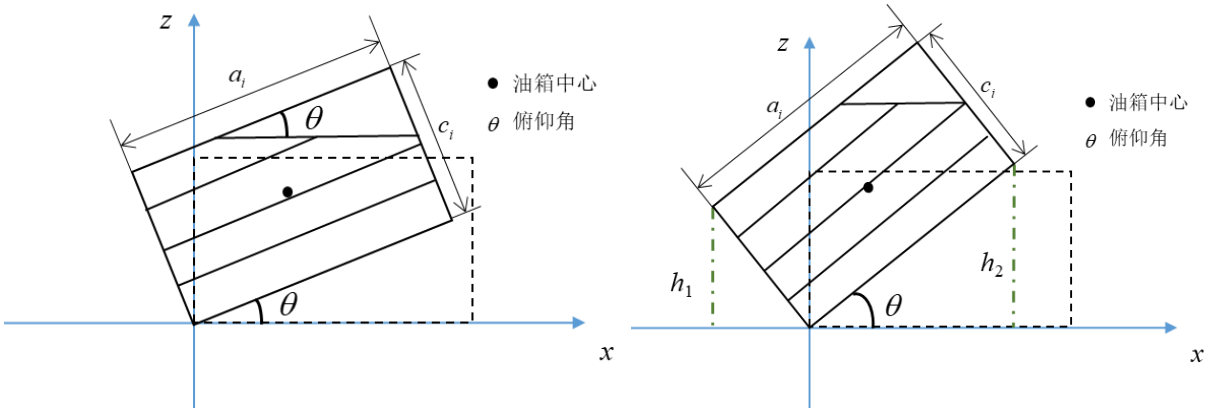


图 3-5 燃油未填满油箱部分的燃油量截面为三角形（情况二）

如下图所示，由情况一可知油箱中未填满燃油的部分对于油箱右上顶点的相对位置为：

$$x_o = \frac{1}{3}\sqrt{\frac{2(a_i c_i - S)}{\tan\theta}}, z_0 = \frac{1}{3}\sqrt{2(a_i c_i - S)\tan\theta} \tag{3-20}$$

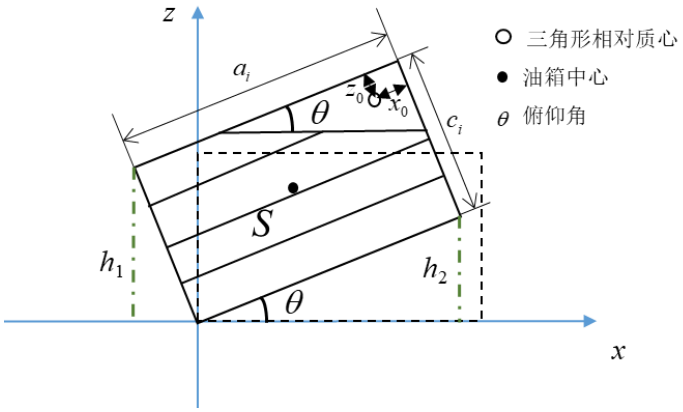


图 3-6 未填满燃油部分相对位置

首先，进行坐标系变换，求解燃油自身（阴影部分）相对于油箱的相对质心位置，以油箱左下顶点为原点，长边为 *x* 轴方向，高边为 *z* 轴方向，如下图所示。

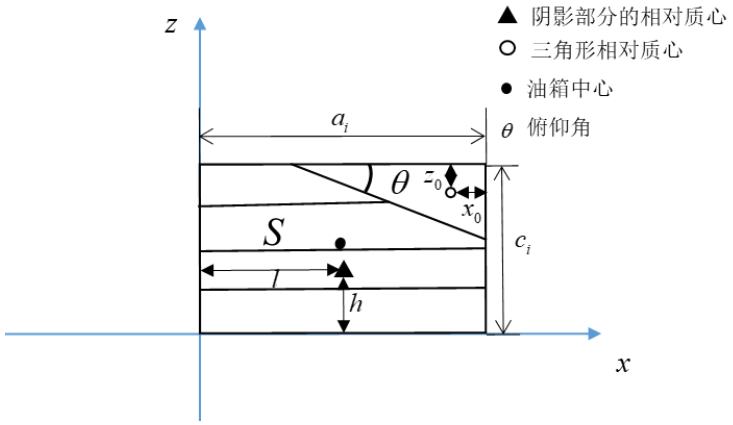


图 3-7 燃油相对质心位置（相对于油箱的坐下顶点）

根据力矩平衡原理可知，整个油箱填满燃油的力矩（整个长方形）等于当前已填满燃油的力矩（阴影部分）加上未填满部分填上燃油的力矩（三角形），因此可列出如下方程：

$$\begin{aligned} Sl+(a_ic_i-S)(a_i-x_0)&=a_ic_i\times\frac{a_i}{2}\\ Sh+(a_ic_i-S)(c_i-z_0)&=a_ic_i\times\frac{a_i}{2}\end{aligned}\tag{3-21}$$

求解可得，

$$\begin{aligned} l&=-\frac{a_i^2c_i}{2S}+(\frac{a_ic_i}{3S}-\frac{1}{3})\sqrt{\frac{2(a_ic_i-S)}{\tan\theta}}+a_i\\ h&=-\frac{a_ic_i^2}{2S}+(\frac{a_ic_i}{3S}-\frac{1}{3})\sqrt{2(a_ic_i-S)\tan\theta}+c_i\end{aligned}\tag{3-22}$$

又因为油箱是长方体结构，油箱中燃油对于油箱的相对质心位置对于油箱中心位置的位置差为

$$\Delta x = l - \frac{a_i}{2}, \Delta z = h - \frac{c_i}{2} \tag{3-23}$$

$$\begin{aligned} \Delta x &= -\frac{a_i^2 c_i}{2S} + (\frac{a_i c_i}{3S} - \frac{1}{3})\sqrt{\frac{2(a_i c_i - S)}{\tan\theta}} + \frac{a_i}{2} \\ \Delta z &= -\frac{a_i c_i^2}{2S} + (\frac{a_i c_i}{3S} - \frac{1}{3})\sqrt{2(a_i c_i - S)\tan\theta} + \frac{c_i}{2} \end{aligned} \tag{3-24}$$

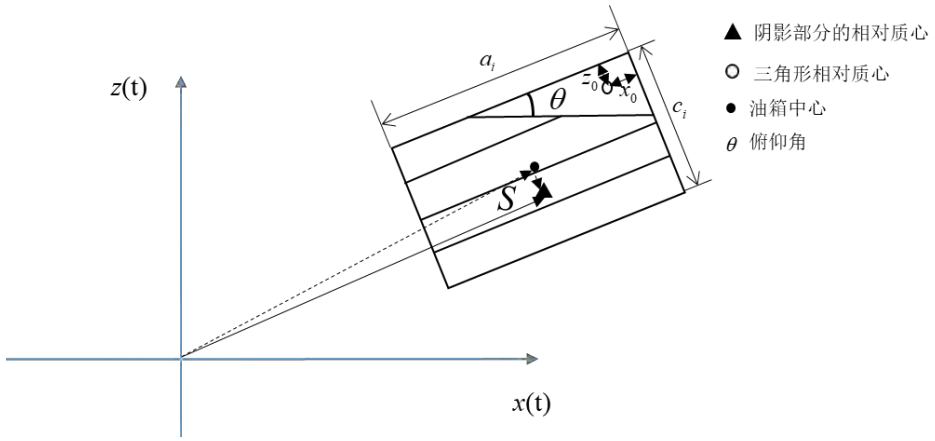


图 3-8 燃油绝对坐标位置（对于飞行器坐标系）

如上图所示，与情况一同理可得，则燃油质心的绝对坐标系为

$$\vec{p}_i = (l - \frac{a_i}{2} + x_i, y_i, h - \frac{c_i}{2} + z_i) \tag{3-25}$$

$$\vec{p}_i = (-\frac{a_i^2 c_i}{2S} + (\frac{a_i c_i}{3S} - \frac{1}{3})\sqrt{\frac{2(a_i c_i - S)}{\tan\theta}} + \frac{a_i}{2} + x_i, y_i, -\frac{a_i c_i^2}{2S} + (\frac{a_i c_i}{3S} - \frac{1}{3})\sqrt{2(a_i c_i - S)\tan\theta} + \frac{c_i}{2} + z_i) \tag{3-26}$$

情况三：

设飞行器俯仰角为θ，当燃油的截面如下图所示时，即符合情况：

$$\frac{c_i^2}{2\tan\theta} < S < a_i c_i - \frac{c_i^2}{2\tan\theta}, \frac{c_i^2}{2\tan\theta} \le \frac{a_i^2 \tan\theta}{2} \tag{3-27}$$

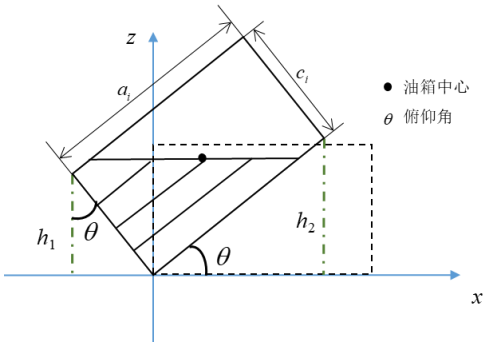


图 3-9 燃油截面为梯形时，且$h_1 \le h_2$（情况三）

首先，进行坐标系变换，求解燃油自身（阴影部分）相对于油箱的相对质心位置，以

油箱左下顶点为原点，长边为 x 轴方向，高边为 z 轴方向，如下图所示。

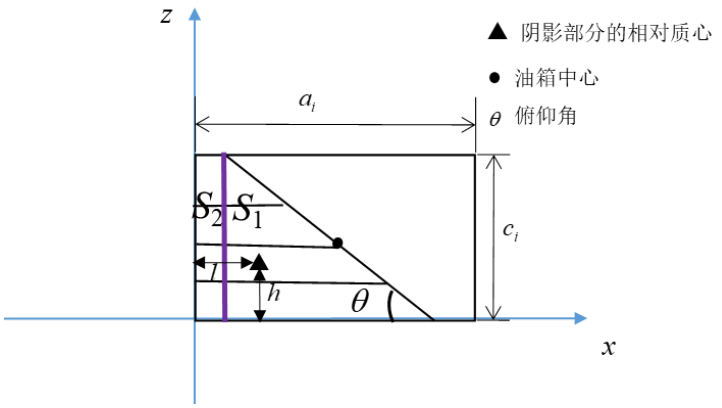


图 3-10 燃油相对质心位置（相对于油箱的坐下顶点）

根据力矩平衡原理可知，全部燃油的力矩（阴影部分梯形，面积 S）等于矩形的力矩（面积 S_2）加上三角形部分的力矩（面积 S_1），因此可列出如下方程：

$$\begin{aligned} S &= S_1 + S_2 \\ S_1 &= \frac{c_i^2}{2\tan\theta} \\ Sl &= S_2\frac{S_2}{2c_i} + S_1(\frac{S_2}{c_i} + x_0) \\ Sh &= S_2\frac{c_i}{2} + S_1 z_0 \\ x_o &= \frac{1}{3}\sqrt{\frac{2S_1}{\tan\theta}}, z_0 = \frac{1}{3}\sqrt{2S_1\tan\theta} \end{aligned} \tag{3-28}$$

可解得，

$$\begin{aligned} l &= \frac{S}{2c_i} + \frac{c_i^3}{24S\tan^2\theta} \\ h &= \frac{c_i}{2} - \frac{c_i^3}{12S\tan\theta} \end{aligned} \tag{3-29}$$

又因为油箱是长方体结构，油箱中燃油对于油箱的相对质心位置对于油箱中心位置的位置差为

$$\Delta x = l - \frac{a_i}{2}, \Delta z = h - \frac{c_i}{2} \tag{3-30}$$

$$\Delta x = \frac{S}{2c_i} + \frac{c_i^3}{24S\tan^2\theta} - \frac{a_i}{2}, \Delta z = -\frac{c_i^3}{12S\tan\theta} \tag{3-31}$$

同理可得，则燃油质心的绝对坐标系为

$$\vec{p}_i = (l - \frac{a_i}{2} + x_i, y_i, h - \frac{c_i}{2} + z_i) \tag{3-32}$$

$$\vec{p}_i = (\frac{S}{2c_i} + \frac{c_i^3}{24S\tan^2\theta} - \frac{a_i}{2} + x_i, y_i, -\frac{c_i^3}{12S\tan\theta} + z_i) \tag{3-33}$$

情况四：

设飞行器俯仰角为θ，当燃油的截面如下图所示时，即符合情况：

$$\frac{a_i^2\tan\theta}{2} < S < a_i c_i - \frac{a_i^2\tan\theta}{2}, \frac{c_i^2}{2\tan\theta} > \frac{a_i^2\tan\theta}{2} \tag{3-34}$$

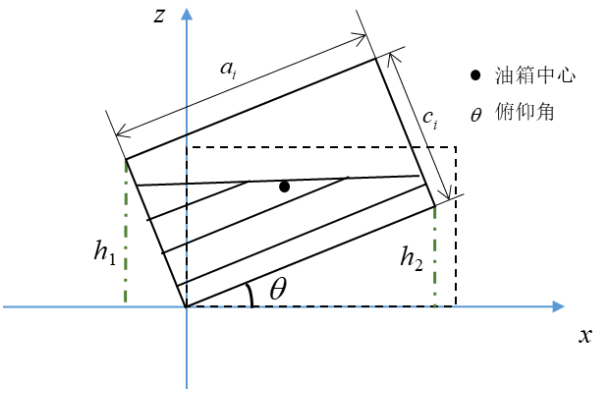


图 3-11 燃油截面为梯形时，且$h_1 > h_2$（情况四）

首先，进行坐标系变换，求解燃油自身（阴影部分）相对于油箱的相对质心位置，以油箱左下顶点为原点，长边为 x 轴方向，高边为 z 轴方向，如下图所示。

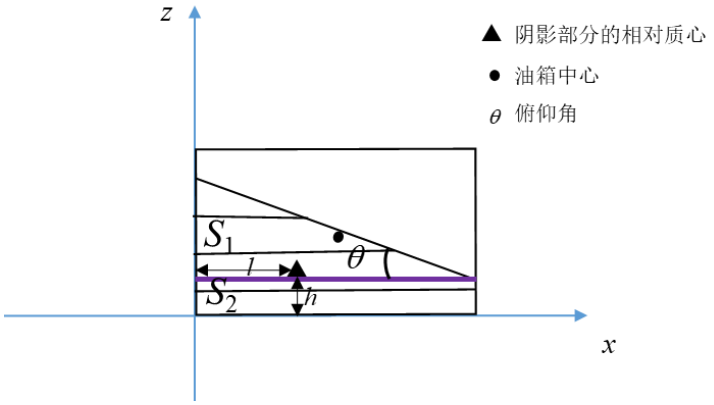


图 3-12 燃油相对质心位置（相对于油箱的坐下顶点）

根据力矩平衡原理可知，全部燃油的力矩（阴影部分梯形，面积 S）等于矩形的力矩（面积 S_2）加上三角形部分的力矩（面积 S_1），因此可列出如下方程：

$$
\begin{aligned}
&S = S_1 + S_2 \\
&S_1 = \frac{a_i^2 \tan\theta}{2} \\
&Sl = S_2 \frac{a_i}{2} + S_1 x_0 \\
&Sh = S_2 \frac{S_2}{2a_i} + S_1(\frac{S_2}{a_i} + z_0) \\
&x_o = \frac{1}{3}\sqrt{\frac{2S_1}{\tan\theta}}, z_0 = \frac{1}{3}\sqrt{2S_1 \tan\theta}
\end{aligned}
\tag{3-35}
$$

可解得，

$$
\begin{aligned}
&l = \frac{a_i}{2} - \frac{a_i^3 \tan\theta}{12S} \\
&h = \frac{S}{2a_i} - \frac{a_i^3 \tan^2\theta}{8S} + \frac{a_i^3}{6S}\tan^2\theta
\end{aligned}
\tag{3-36}
$$

又因为油箱是长方体结构，油箱中燃油对于油箱的相对质心位置对于油箱中心位置的位置差为

$$
\Delta x = l - \frac{a_i}{2}, \Delta z = h - \frac{c_i}{2} \tag{3-37}
$$

故可得，则燃油质心的绝对坐标系为

$$
\vec{p}_i = (l - \frac{a_i}{2} + x_i, y_i, h - \frac{c_i}{2} + z_i) \tag{3-38}
$$

$$
\vec{p}_i = (-\frac{a_i^3 \tan\theta}{12S} + x_i, y_i, \frac{S}{2a_i} - \frac{a_i \tan\theta}{4} + \frac{a_i^3}{6S}\tan^2\theta - \frac{c_i}{2} + z_i) \tag{3-39}
$$

3.2.3 飞行器俯仰角小于 0 的质心计算模型

飞行器俯仰角小于 0 时，与飞行器俯仰角大于 0 的情况类似，不同之处在于油箱中燃油对于油箱的相对质心位置与油箱中心位置的位置差为在 z 轴上是一样的，x 轴是相反的，故

$$
\Delta x' = -\Delta x, \Delta z' = \Delta z \tag{3-40}
$$

故可得，则燃油质心的绝对坐标系为

$$
\vec{p}_i = (-\Delta x + x_i, y_i, \Delta z + z_i) \tag{3-41}
$$

其中，$\Delta x, \Delta z$ 取值为将实际中的俯仰角取绝对值，再带入至俯仰角大于 0 的质心计算过程中，即上一小节中的计算方法（飞行器俯仰角大于 0 的质心计算）。

3.3 求解结果及分析

针对附件 2 中的数据求解此次任务执行过程中的质心位置，得到质心变化曲线如下：

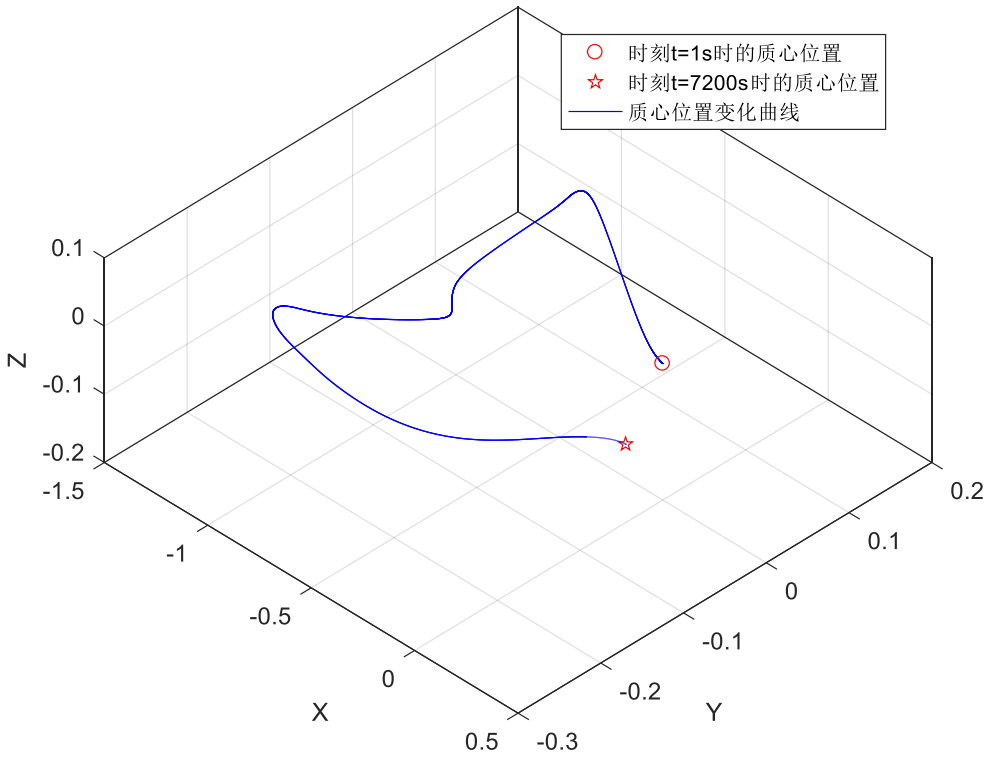


图 3-13 飞行器在此次任务执行过程中的质心变化曲线

飞行器在此次任务执行过程中的质心位置 *x*,*y*,*z* 坐标随时间的变化过程如下：

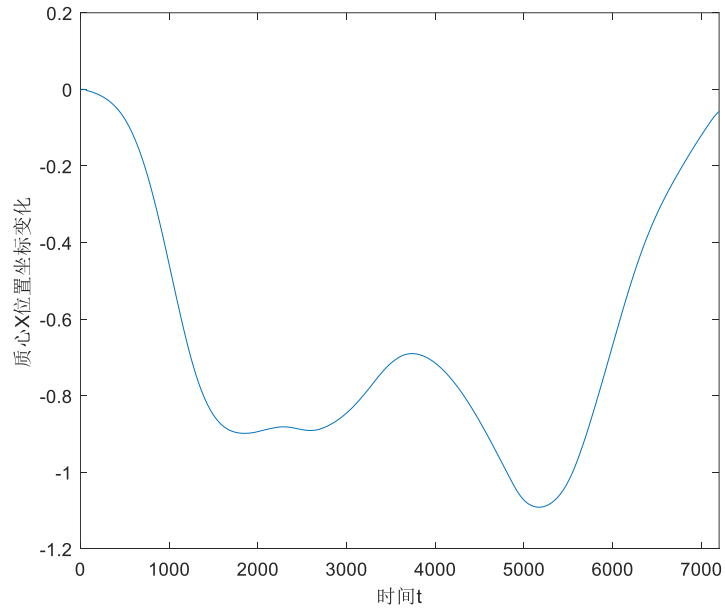


图 3-14 飞行器在此次任务执行过程中的质心位置 *x* 坐标变化曲线

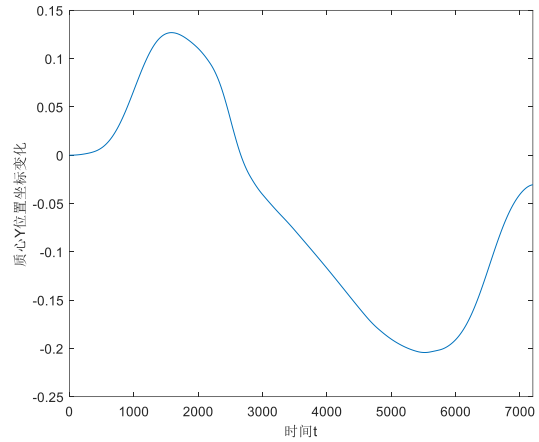


图 3-14 飞行器在此次任务执行过程中的质心位置 *y* 坐标变化曲线

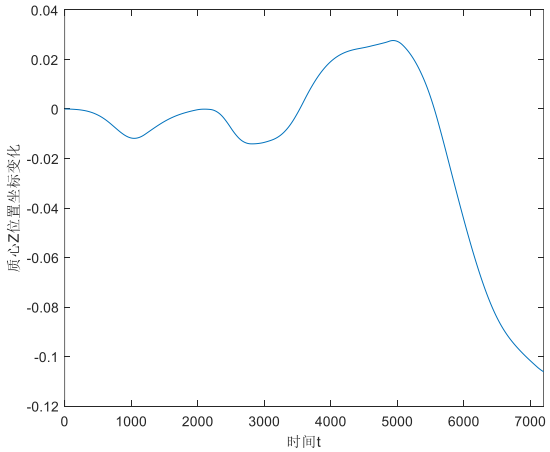


图 3-14 飞行器在此次任务执行过程中的质心位置 *z* 坐标变化曲线

飞行器在此次任务执行过程中的质心位置 *x*,*y*,*z* 坐标随时间的变化过程如下，其中离原点最优距离为 1.109137m：

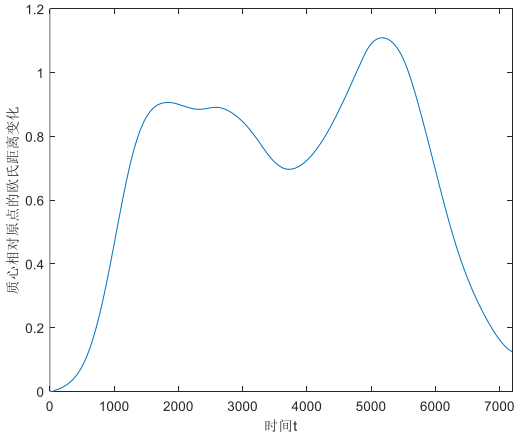


图 3-15 飞行器在此次任务执行过程中的质心相对原点的欧氏距离变化

四、问题二的模型建立与求解

4.1 问题分析及数学模型的建立

4.1.1 问题二的分析

问题二需要制定出飞行器在满足约束条件下的油箱供油策略，使得飞行器每一时刻的质心位置与理想质心位置的欧氏距离的最大值达到最小。此题目中的飞行情形为飞行器始终保持平飞，即俯仰角为 0，质心位置可通过 3.2.1 节中的方法计算。此外，制定的油箱供油策略还需要满足以下约束：

约束 1：第 i 个油箱的供油速度上限为，U_i (U_i >0)，i=1,2,⋯,6，且供油速度为非负值；

约束 2：每个油箱一次供油的持续时间不少于 60 秒；

约束 3：由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油；

约束 4：各油箱联合供油的总量应至少满足发动机的对耗燃油的需要；

约束 5：主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别为油箱 2 和油箱 5 供油，不能直接向发动机供油；各油箱中油量不能超过油箱内部容积，且不能为负值。

4.1.2 数学模型的构建

问题中的决策变量为每一时刻各油箱的供油策略，即供油速度，定义为 $v_i(t)$，表示油箱 i 在 t 时刻的供油速度。

（1）目标函数：最小化飞行器每一时刻的质心位置与理想质心位置的欧氏距离的最大值：

$$\min \max_t \left\| \vec{c}_1(t) - \vec{c}_2(t) \right\|_2 \tag{4-1}$$

（3）约束条件：

约束 1：第 i 个油箱的供油速度上限为，U_i (U_i >0)，i=1,2,⋯,6，且供油速度为非负值；

$$\forall t \in [0,T],\ 0 \le v_i(t) \le U_i \tag{4-2}$$

约束 2：每个油箱一次供油的持续时间不少于 60 秒；

$$t_{oil}(i) \ge 60 \tag{4-3}$$

约束 3：由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油；

$$\forall t \in [0,T],\ \sum_{i=2}^{5} (v_i(t) == 0) \ge 2 \tag{4-4}$$

$$\forall t \in [0,T],\ \sum_{i=1}^{6} (v_i(t) == 0) \ge 3 \tag{4-5}$$

约束 4：各油箱联合供油的总量应至少满足发动机的对耗燃油的需要；

$$\forall t \in [0,T],\ \sum_{i=2}^{5} v_i(t) \ge R(t) \tag{4-6}$$

约束 5：主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别为油箱 2 和油箱 5 供油，不能直接向发动机供油；各油箱中油量不能超过油箱内部容积，且不能为负值。

$$\forall t \in [0,T],\ 0 \le q_i(t) \le a_i b_i c_i \rho \tag{4-7}$$

其中，

$$\begin{aligned} q_i(t) &= O_i\rho - \int_0^t v_i(t)dt +, i = 1,3,4,6 \\ q_i(t) &= O_i\rho - \int_0^t v_i(t)dt + \int_0^t v_1(t)dt, i = 2 \\ q_i(t) &= O_i\rho - \int_0^t v_i(t)dt + \int_0^t v_6(t)dt, i = 5 \end{aligned} \tag{4-8}$$

（3）综上，可建立如下的数学模型：

$$\min \max_t \left\| \vec{c}_1(t) - \vec{c}_2(t) \right\|_2$$

$$s.t. \begin{cases} \forall t \in [0,T],\ 0 \le v_i(t) \le U_i \\ t_{oil}(i) \ge 60 \\ \forall t \in [0,T],\ \sum_{i=2}^{5} (v_i(t) == 0) \ge 2 \\ \forall t \in [0,T],\ \sum_{i=1}^{6} (v_i(t) == 0) \ge 3 \\ \forall t \in [0,T],\ \sum_{i=2}^{5} v_i(t) \ge R(t) \\ \forall t \in [0,T],\ 0 \le q_i(t) \le a_i b_i c_i \rho \end{cases} \tag{4-9}$$

根据数学模型分析可知，问题二是一个十分具有挑战性的问题。其一，因为其时间上的动态特性（后续油箱的状态会被前期油箱输油量所影响），传统的优化方法（数学规划，遗传算法）因高昂的计算代价将导致传统的优化方法不能处理该问题。其二，油箱的最大输油量、最大供油箱数量、油箱最短供油时间、发动机的燃油等因素将会导致不同油箱之间一个复杂且强耦合的关系。

4.2 算法描述

为了减少因为问题维度和动态特性所带来的巨大计算代价，本文采用了一种基于供油模式的多阶段优化方法框架。在该方法框架下，本文首先设计了一种遗传编程的超启发式方法，来获取高性能表现的切换供油模式的选择规则。并且根据生成的超启发式选择规则生成一个高质量的初始供油方案。其次，为了优化每时刻的供油速度来降低最大瞬时偏差，本文设计了一种差分进化算法。其算法设计细节如下文所示。

（1）阶段一：遗传编程的超启发方法获取一个高质量初始供油方案

本小节所研究问题的维度为 7200*6，传统方法因其巨大的计算代价无法处理如此高维度的问题。遗传编程的超启发式（genetic programing hyper-heuristic）方法以被广泛的应用在动态车间生产调度问题和不确定的车辆路径问题等问题。遗传编程的超启发式方法在启发式解空间搜索，而不是直接在问题的解空间中搜索，因此其有效地降低了问题的维度。遗传编程的超启发式方法因其灵活的表达性和强收敛性，获得了相比人为设计的启发式贪心方法更好的表现性能[2][3][4][6]。综上所述，遗传编程的超启发式方法十分适合本小节所研究问题的求解。

为了快速获取一个高质量的初始供油方案，同时为了设计启发式解空间供遗传编程来搜索[5]，本文设计了 35 种供油模式，如表 4-1 所示。在设计的 35 种供油模式中，若供油油箱数量为 1 时，其每秒供油量等于当前时刻发动机要求的最小供油量；若供油油箱的数量为 2 时，且供油油箱种无后备油箱（油箱 1、6），每个供油油箱的速度等于当前时刻发

动机要求的最小供油量除以 2； 若供油油箱数量为 2 时，且其中有一后备油箱（油箱 1 或 6），直接供油油箱（油箱 2、3、4 和 5）的每秒供油量等于当前时刻发动机要求的最小供油量，后备油箱按照其约束下的最大供油速度来供油；若供油油箱数量为 3 时，且其中之一为后备油箱，每个直接供油油箱（油箱 2、3、4 和 5）的速度等于当前时刻发动机要求的最小供油量除以 2，后备油箱按照约束下的最大供油速度来供油；若供油油箱数量为 3 时，且其中之二为后备油箱，直接供油油箱（油箱 2、3、4 和 5）的速度等于当前时刻发动机要求的最小供油量，后备油箱（油箱 1、6）按照约束下的最大供油速度来供油；若供油箱数量为 0 时，显然供油速度也为 0。

表 4-1 供油模式下的供油油箱和供油速度。

供油模式编号	供油油箱	供油速度
1	[2, 3]	$[R(t)/2,\ R(t)/2]$
2	[2, 4]	$[R(t)/2,\ R(t)/2]$
3	[2, 5]	$[R(t)/2,\ R(t)/2]$
4	[3, 4]	$[R(t)/2,\ R(t)/2]$
5	[3, 5]	$[R(t)/2,\ R(t)/2]$
6	[4, 5]	$[R(t)/2,\ R(t)/2]$
7	[1, 2, 3]	$[U_1,\ R(t)/2,\ R(t)/2]$
8	[1, 2, 4]	$[U_1,\ R(t)/2,\ R(t)/2]$
9	[1, 2, 5]	$[U_1,\ R(t)/2,\ R(t)/2]$
10	[1, 3, 4]	$[U_1,\ R(t)/2,\ R(t)/2]$
11	[1, 3, 5]	$[U_1,\ R(t)/2,\ R(t)/2]$
12	[1, 4, 5]	$[U_1,\ R(t)/2,\ R(t)/2]$
13	[2, 3, 6]	$[R(t)/2,\ R(t)/2,\ U_6]$
14	[2, 4, 6]	$[R(t)/2,\ R(t)/2,\ U_6]$
15	[2, 5, 6]	$[R(t)/2,\ R(t)/2,\ U_6]$
16	[3, 4, 6]	$[R(t)/2,\ R(t)/2,\ U_6]$
17	[3, 5, 6]	$[R(t)/2,\ R(t)/2,\ U_6]$
18	[4, 5, 6]	$[R(t)/2,\ R(t)/2,\ U_6]$
19	[2]	$[R(t)]$
20	[3]	$[R(t)]$
21	[4]	$[R(t)]$
22	[5]	$[R(t)]$
23	[1, 2]	$[U_1,\ R(t)]$
24	[1, 3]	$[U_1,\ R(t)]$
25	[1, 4]	$[U_1,\ R(t)]$

26	[1, 5]	$[U_1, R(t)]$
27	[2, 6]	$[R(t), U_6]$
28	[3, 6]	$[R(t), U_6]$
29	[4, 6]	$[R(t), U_6]$
30	[5, 6]	$[R(t), U_6]$
31	[1, 2, 6]	$[U_1, R(t), U_6]$
32	[1, 3, 6]	$[U_1, R(t), U_6]$
33	[1, 4, 6]	$[U_1, R(t), U_6]$
34	[1, 5, 6]	$[U_1, R(t), U_6]$
35	空	空

如约束（4-3）所示，每个油箱一次供油的持续时间不少于 60 秒。本文以该约束为基础，将总供油周期分解为由 120 个等于 60s 最小供油时间窗口组成。在最小供油时间窗口内，采用的供油模式不发生变化。如图 4-1 所示， 选择的 120 个最小供油时间窗口的供油模式与 7200*6 的每个油箱的供油速度存在严格的映射关系。由此，该问题从一个 7200*6 的超高纬度的连续优化问题，转换为一个 120 维的离散变量优化问题。

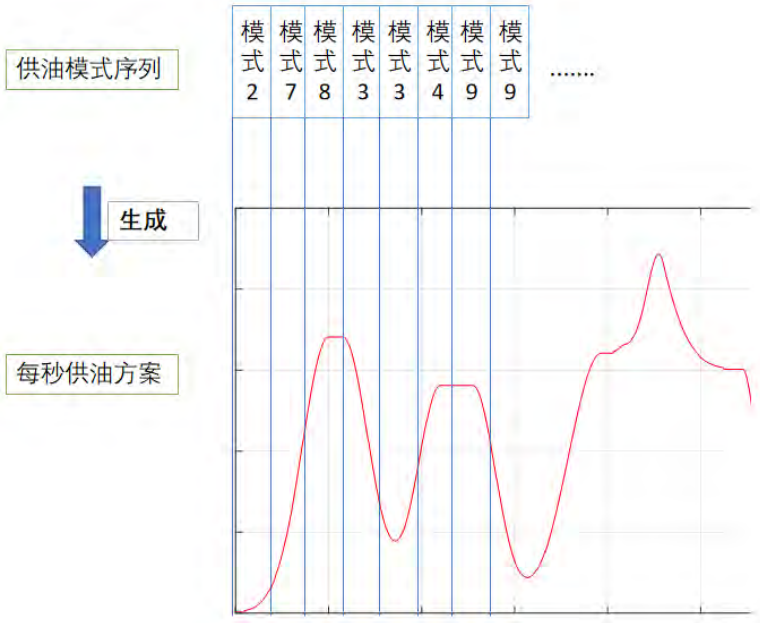


图 4-1 供油模式序列与每秒供油方案对应关系示意图

为了得到一个高质量的供油模式序列，本文采用了遗传编程的超启发式方法来自动设计高质量供油模式序列的选择规则。自动设计的供油模式选择规则相比较人为设计的贪心启发式规则可以规避很多不足。例如，只考虑如何减少最大瞬时偏差的启发式贪心算法会导致在供油后期无油可用的情况。自动设计的选择规则可以有效的分析综合当前油箱的状态和后续供油量的需求。遗传编程的超启发流程如图所示，首先，随机生成一种群超启发式选择规则个体。之后，交叉、变异、重构生产新的子代种群。 同时对每一个超启发选择规则个体进行评价，再通过锦标赛规则筛选子代种群和父代获取新的种群。判断是否满

足终止条件，若不满足重复上述步骤，若满足，遗传编程的超启发方法输出最佳个体所代表的最佳选择规则。最后通过优化得到的选择规则个体选择供油模式进一步生成供油方案。

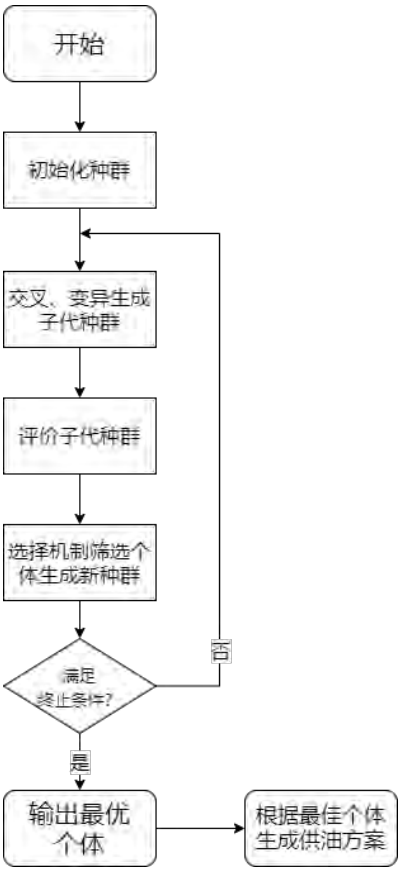


图 4-2 遗传编程流程图

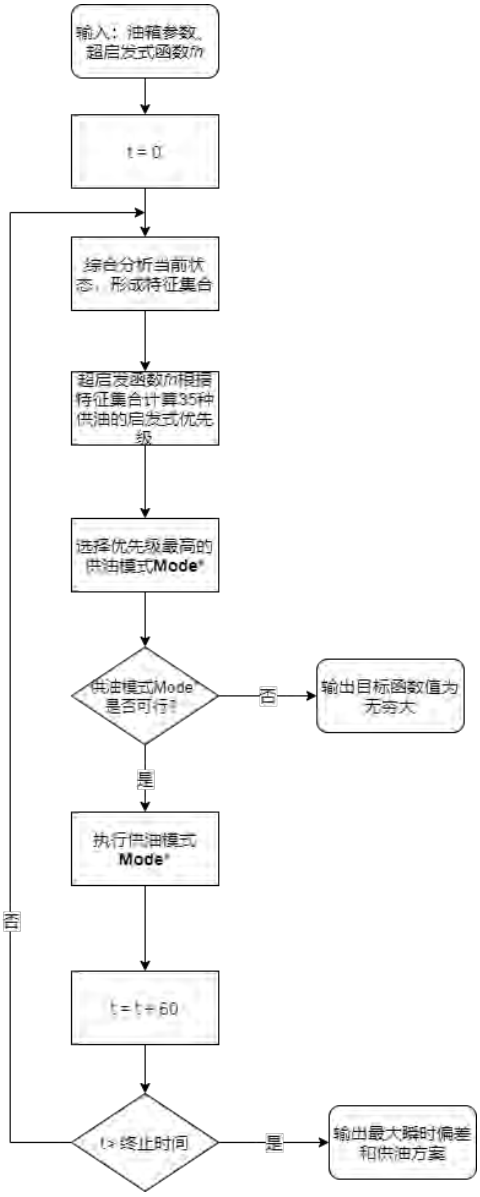


图 4-3 供油仿真解码流程图

为了完成遗传编程的超启发算法中的对每个个体评价环节，需在遗传编程个体与供油方案之间建立唯一映射。本文设计了一种供油仿真解码方法来建立遗传编程个体 fh-供油模式序列-每秒供油方案之间的映射。其分为三个过程，分别为：初始化过程、决策过程、执行过程，如图 所示。 在初始化环节中：油箱的各个参数和超启发式的输油模式的选择规则函数 *fh* 输入到解码方法中，同时供油仿真时间设置为 0. 在决策环节中，如图所示，超启发式 *fh* 根据当前和预测的状态信息（当前时刻、当前时刻在整体时间轴占比、当前时刻下油箱中剩余的油量、执行该供油模式后会产生的最大瞬时偏差、当前时刻下质心的位置、未来的最小总耗油量等）计算 35 种供油模式下的启发式优先级。选择启发式优先级最高的供油模式 mode*作为下一个时间窗口所采用的供油模式。在下一个时间窗口执行供油模式 mode*后，判断是否完成到达任务终止时间。若是，则输出在整个飞行任务阶段时间内

供油方案和最大瞬时偏差，若为否，更新当前所有的状态信息重复上述的决策和执行过程。值得注意的是，本过程十分类似于贪心算法的解构造过程。但是贪心算法所构造过程中供油模式的选择规则函数往往是人为设定的（例如根据选择会产生最小偏差的供油模式），而本文中供油模式的选择规则是被遗传编程所优化得到的。

图 4-4 超启发式函数选择供油模式的示意图

（2）阶段二：差分进化方法对获取的高质量解进行进一步优化获取供油方案

遗传编程的优化方式最大的意义在于能够得到可行的油箱选择和简易的油箱流速规划方案，但该方法对问题的深度探索能力稍显乏力。为此本文为了进一步对解进行改进，协调燃料流速和质心偏差的多目标差分进化算法[7][8][9]。

本多目标差分进化算法的主要思路是在原有油箱使用方案的基础上，调节各油箱每一秒的燃油流速，对质心偏移量和燃油使用量进行局部改进。

基于上面思路，本优化问题的解可以表示如下：

$$V_k'(t)=\{v_{k,1}'(t),v_{k,2}'(t),v_{k,3}'(t),v_{k,4}'(t),v_{k,5}'(t),v_{k,6}'(t)\} \tag{4-10}$$

其中，v_i' 为油箱 i 的燃油流速。为了保证原有每一时刻的油箱选择方案不会产生不可行解，并对原有供油方案进行改进，本算法的中各油箱的燃油流速被约束到了原供油流速以下，即约束（6）：

$$v_i'(t)\le v_i(t)\quad ,\ i\in\{1,2,...,5,6\} \tag{4-11}$$

此外，本文 4.1.2 中的 5 项模型约束同样适用于本问题。

本算法的目的为在不增加当前燃油使用量的基础上改进质心偏移量，因此，本问题可归纳为多目标优化问题，即最小化燃油使用量及质心偏移量。为了便于求解，通过设定权重值，将多目标优化问题转化为单目标优化问题进行优化。故优化问题目标函数可以表示如下：

$$\min\ \alpha\cdot(\|\vec{c}_1(t)-\vec{c}_2(t)\|_2)+(1-\alpha)\cdot\sum_{i=1}^{6}v_i(t) \tag{4-11}$$

其中，α 为两个子目标函数的相对权重值。

约束汇总如下：

$$\min\ \alpha\cdot(\left\|\vec{c}_1(t)-\vec{c}_2(t)\right\|_2)+(1-\alpha)\cdot\sum_{i=1}^{6}v_i(t)$$

$$s.t.\begin{cases}\forall t\in[0,T],\ 0\le v_i(t)\le U_i\\ t_{oil}(i)\ge 60\\ \forall t\in[0,T],\ \sum_{i=2}^{5}(v_i(t)==0)\ge 2\\ \forall t\in[0,T],\ \sum_{i=1}^{6}(v_i(t)==0)\ge 3\\ \forall t\in[0,T],\ \sum_{i=2}^{5}v_i(t)\ge R(t)\\ \forall t\in[0,T],\ 0\le q_i(t)\le a_ib_ic_i\rho\\ v_i'(t)\le v_i(t)\end{cases}\qquad (4\text{-}12)$$

上述问题为连续变量优化问题，考虑使用差分进化算法进行优化，步骤如下：

步骤一：初始化算法及种群参数，主要包括迭代次数、变异概率、交叉概率、种群规模等参数；

步骤二：通过随机采样的方式生成初代种群，具体对解中每一分量的采样方式表示如下；

$$v_i'(t)=\text{rand}()\cdot v_i(t)\qquad (4\text{-}13)$$

步骤三：根据问题目标函数对种群中各个体进行适应度值评价；

步骤四（选择）：根据个体适应度值，采用 25%截断选择的方式选择优势个体用于产生下一代新个体；

步骤五（变异）：通过下面公式完成对新个体每一分量的变异操作：

$$v_{k,i}'(t)=\begin{cases}v_{q,i}'(t)+Fc\cdot(v_{l,i}'(t)-v_{m,i}'(t)), & \text{rand}()<Pm\\ v_{k,i}'(t), & \text{else.}\end{cases}\qquad (4\text{-}14)$$

其中，$v_{q,j}'(t),v_{l,j}'(t),v_{m,j}'(t)$ 为从优势个体中随机选择的个体 $V_q'(t),V_l'(t),V_m'(t)$ 对应位置上 j 的分量，Pm,Fc 分别为变异概率和缩放因子。

步骤六（交叉）：通过下面公式完成对变异步骤中产生的新个体的每一分量的交叉操作：

$$v_{k,i}'(t)=\begin{cases}v_{k,i}(t), & \text{rand}()<Pc\\ v_{k,i}'(t), & \text{else.}\end{cases}\qquad (4\text{-}14)$$

步骤七：检查算法迭代终止判据，若满足，输出当前找到最优解；否则，返回步骤三。

4.3 求解结果及分析

4.3.1 算法的有效性分析

上述算法中的参数设置为：遗传编程的种群规模为 20，进化代数为 50，交叉概率为 0.5，变异概率为 0.2，选择策略为锦标赛选择策略和精英选择策略；差分进化算法的种群规模为 100，迭代次数为 20，交叉概率为 0.5，变异概率为 0.9。通过基于遗传编程的超启发式和差分进化方法共同对数据集进行求解，可以制定出较优的供油策略，所得供油策略中飞行器瞬时质心与理想质心距离的最大值为 0.10233m，4 个主油箱的总供油量为 6483.8242kg。

飞行器飞行过程中 6 个油箱各自的供油速度曲线如下，可以看出在油箱 1 和 6 作为副油箱在供油时大都处于最高供油速度，而油箱 2~5 给发动机供油时的供油峰值大都未达到最大供油速度。

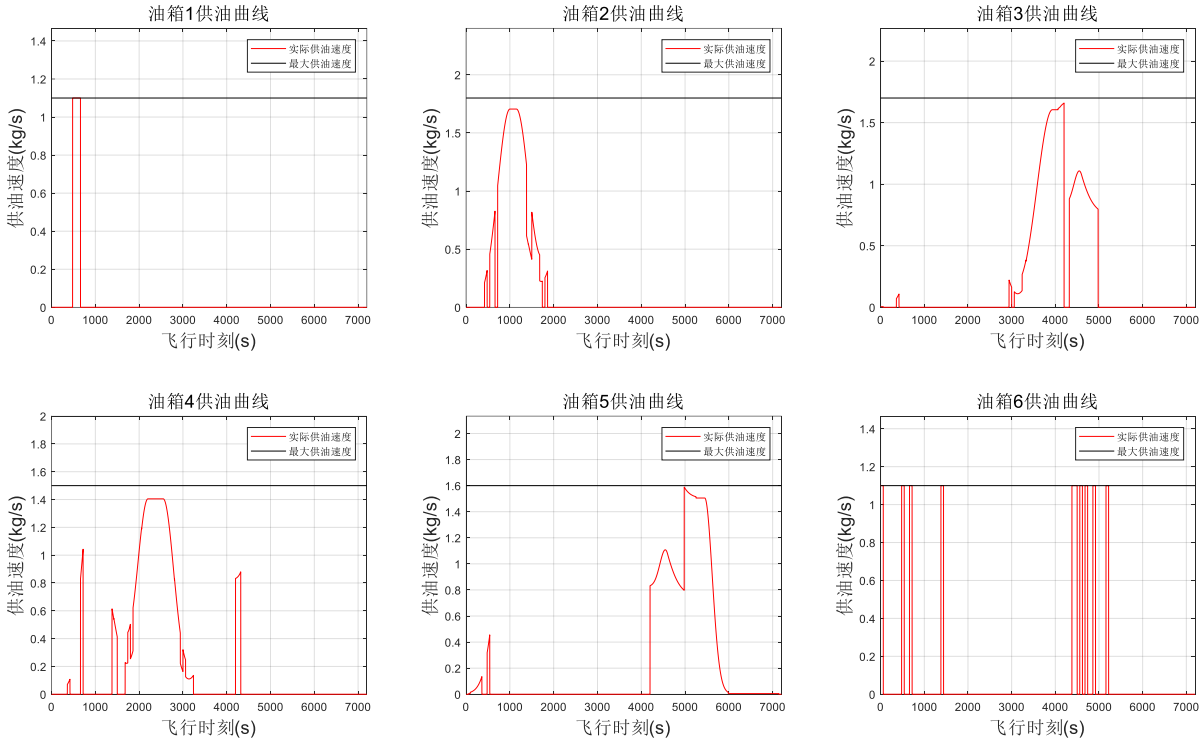


图 4-4 6 个油箱的供油速度曲线

飞行器飞行过程中 4 个主油箱的总供油速度曲线(时间间隔为 1s)如下：

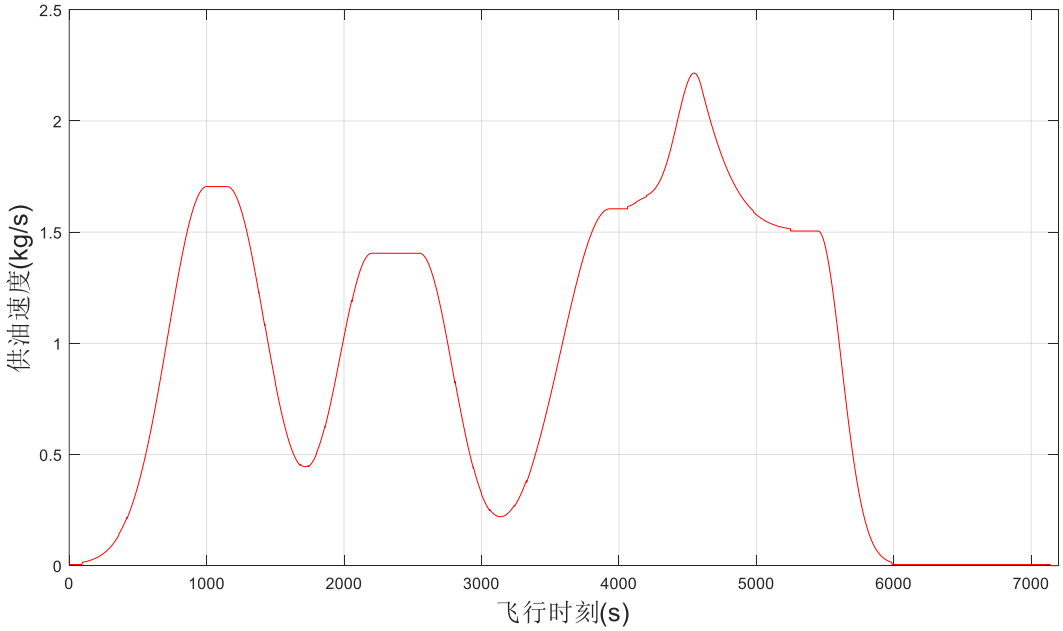


图 4-5 4 个主油箱的总供油速度曲线

飞行器飞行过程中飞行器瞬时质心与理想质心的距离曲线如下，飞行器的瞬时质心与理想质心的距离在 4000 秒之前一直处于 0.04 米以下，说明算法优化的供油方案可以较好地使飞行器质心跟随理想质心位置，而在 4000~5000 秒时，产生突变，偏差增大，根据燃油剩余量推断原因应该是由于飞行过程后期油箱油量并非十分充足，但是最大偏差仍不到 0.12 米，可见本文所提出算法产生的供油策略的有效性。

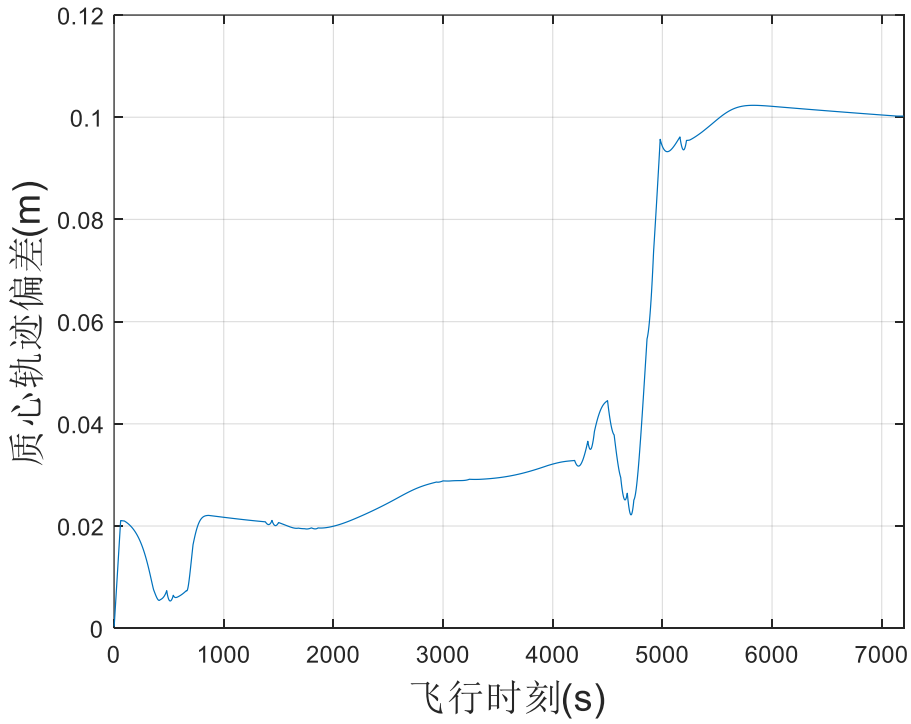


图 4-6 飞行器瞬时质心与理想质心的距离曲线

飞行器飞行过程中各个油箱的使用情况甘特图如下所示，可以从图中看出来，每个油箱都使用了一段时间，在前半段时间内飞行器的飞行主要靠油箱 2 和油箱 4 为发动机供油，后半段主要由油箱 3 和油箱 5 供油；在 6 个油箱中，油箱 1 使用的次数和时间最少，油箱 5 使用的时间最长，尤其，在后半段时间内连续使用了两千多秒，因此也可以看出油箱 6 在间断给油箱 5 供油，保证油箱 5 可以给发动机正常供油。

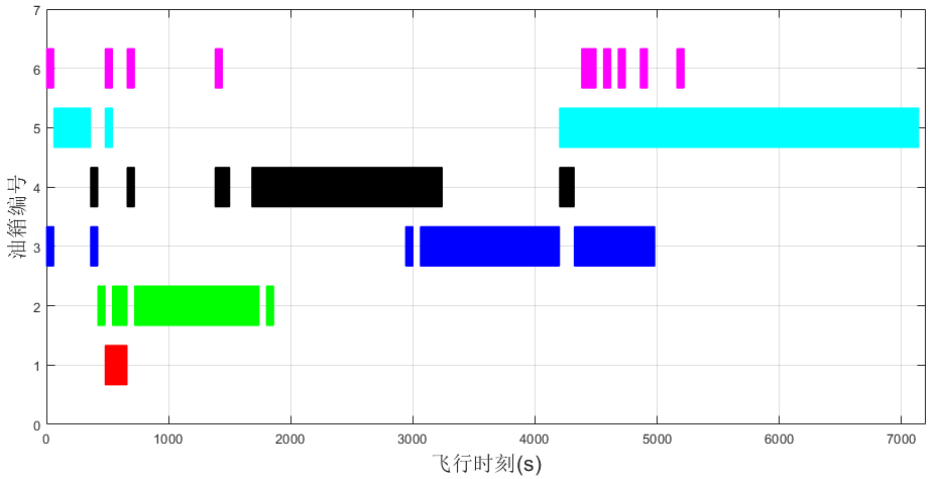


图 4-7 飞行器油箱使用情况甘特图

飞行器质心实际轨迹与理想轨迹图如下图所示。

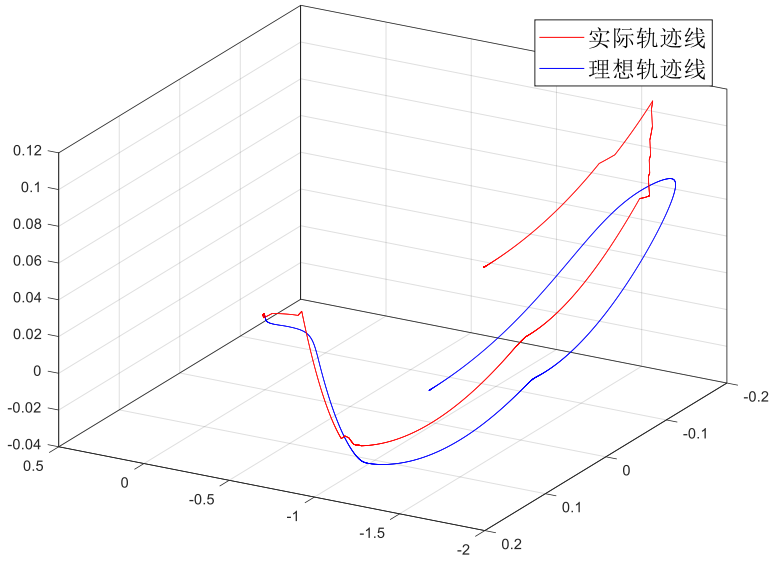


图 4-5 飞行器实际轨迹与理想轨迹曲线

4.3.2 算法的分析

本文设计的基于遗传编程的超启发式算法中，其采用的终端集合（考虑的当前信息及预测的状态）和函数集合如下表所示：

表 4-2 基于遗传编程的超启发式算法终端集合和函数集合

表示	描述
pdev	采用该供油模式会在下一个时间窗口产生的最大瞬时偏差
Tank2Oil	采用该供油模式，在下一个时间窗口油箱 2 在当前时刻的剩余油量（m^3）
Tank5Oil	采用该供油模式，在下一个时间窗口油箱 5 在当前时刻的剩余油量（m^3）
Tank6Oil	采用该供油模式，在下一个时间窗口油箱 6 在当前时刻的剩余油量（m^3）
pTank2Oil	采用该供油模式，在下一个时间窗口油箱 2 在当前时刻的剩余油量与初始油量的比值
pTank5Oil	采用该供油模式，在下一个时间窗口油箱 5 在当前时刻的剩余油量与初始油量的比值
pTank6Oil	采用该供油模式，在下一个时间窗口油箱 6 在当前时刻的剩余油量与初始油量的比值
time	当前时刻
函数集合	add, sub, negative, max, mul, div

其中一次进化得到的遗传编程树所表示的供油模式选择规则，如下图 4-6 所示。从遗传编程自动优化得到的选择规则可知，当候选的供油模式产生的最大瞬时偏差越大时，其被选择的优先级越低；当候选供油模式对油箱 2 和 5 中的油量消耗越小时，其被选择的优先级越高。尽管所设计的特征参量没有全部用在自动设计的超启发式规则中，但该超启发式进化算法框架相对于人为设计的贪心算法更能适应给定数据的变化。**该多阶段的混合超启发式方法具有良好的适用性和扩展性。**

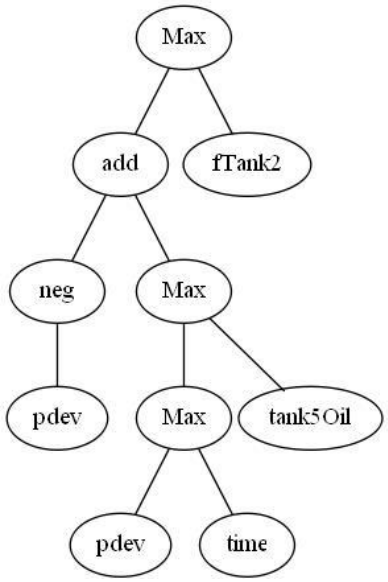


图 4-6 供油模式选择规则的超启发式函数树状表示图

算法复杂度：

遗传编程算法复杂度：假设油箱的数量为 m，优选供油模式的数量为 w，迭代次数为 gen，计算优先级的函数为 $O(1)$，供油仿真过程时间复杂度为 $O(n\times w\times gen)$。

差分进化算法复杂度：假设油箱的数量为 m，迭代次数为 gen，供油仿真过程时间复杂度为 $O(m\times gen)$。

五、问题三的模型建立与求解

5.1 问题分析及数学模型的建立

5.1.1 问题三的分析

问题三需要制定出飞行器在满足约束条件下的 6 个油箱的初始载油量和油箱供油策略，使得每一时刻飞行器瞬时质心位置与飞行器理想质心的欧氏距离的最大值达到最小，并且任务结束时 6 个油箱剩余燃油总量至少 $1m^3$。此题目中的飞行情形为飞行器始终保持平飞，即俯仰角为 0，质心位置可通过 3.2.1 节中的方法计算。此外，制定的油箱供油策略还需要满足以下约束：

约束 1：第 i 个油箱的供油速度上限为，U_i (U_i >0)，i=1,2,…,6，且供油速度为非负值；

约束 2：每个油箱一次供油的持续时间不少于 60 秒；

约束 3：由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油；

约束 4：各油箱联合供油的总量应至少满足发动机的对耗燃油的需要；

约束 5：主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别

为油箱 2 和油箱 5 供油，不能直接向发动机供油；各油箱中油量不能超过油箱内部容积，且不能为负值；

约束 6：任务结束时 6 个油箱剩余燃油总量至少 $1m^3$。

5.1.2 数学模型的构建

问题中的决策变量为每一时刻各油箱的供油策略，即供油速度，定义为 $v_i(t)$，表示油箱 i 在 t 时刻的供油速度。

（1）目标函数：最小化飞行器每一时刻的质心位置与飞行器（不载油）质心位置的最大欧氏距离偏差：

$$\min \max_t \left\| \vec{c}_1(t) - \vec{c}_2(t) \right\|_2 \tag{5-1}$$

（3）约束条件：

约束 1：第 i 个油箱的供油速度上限为，U_i (U_i >0)，i=1,2,⋯,6，且供油速度为非负值：

$$\forall t \in [0,T],\ 0 \le v_i(t) \le U_i \tag{5-2}$$

约束 2：每个油箱一次供油的持续时间不少于 60 秒：

$$t_{oil}(i) \ge 60 \tag{5-3}$$

约束 3：由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油：

$$\forall t \in [0,T],\ \sum_{i=2}^{5} (v_i(t) == 0) \ge 2 \tag{5-4}$$

$$\forall t \in [0,T],\ \sum_{i=1}^{6} (v_i(t) == 0) \ge 3 \tag{5-5}$$

约束 4：各油箱联合供油的总量应至少满足发动机的对耗燃油的需要：

$$\forall t \in [0,T],\ \sum_{i=2}^{5} v_i(t) \ge R(t) \tag{5-6}$$

约束 5：主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别为油箱 2 和油箱 5 供油，不能直接向发动机供油；各油箱中油量不能超过油箱内部容积，且不能为负值：

$$\forall t \in [0,T],\ 0 \le q_i(t) \le a_i b_i c_i \rho \tag{5-7}$$

其中，

$$\begin{aligned} q_i(t) &= O_i\rho - \int_0^t v_i(t)dt +, i = 1,3,4,6 \\ q_i(t) &= O_i\rho - \int_0^t v_i(t)dt + \int_0^t v_1(t)dt, i = 2 \\ q_i(t) &= O_i\rho - \int_0^t v_i(t)dt + \int_0^t v_6(t)dt, i = 5 \end{aligned} \tag{5-8}$$

约束 6：任务结束时 6 个油箱剩余燃油总量至少 $1m^3$：

$$\sum_{i=1}^{6} O_i - \frac{1}{\rho} \sum_{t=1}^{T} q_i(t) \ge 1 \tag{5-9}$$

（3）综上，可建立如下的数学模型：

$$\min \max_{t} \left\| \vec{c}_1(t) - \vec{c}_2(t) \right\|_2$$

$$s.t. \begin{cases} \forall t \in [0,T],\ 0 \le v_i(t) \le U_i \\ t_{oil}(i) \ge 60 \\ \forall t \in [0,T],\ \sum_{i=2}^{5} (v_i(t) == 0) \ge 2 \\ \forall t \in [0,T],\ \sum_{i=1}^{6} (v_i(t) == 0) \ge 3 \\ \forall t \in [0,T],\ \sum_{i=2}^{5} v_i(t) \ge R(t) \\ \forall t \in [0,T],\ 0 \le q_i(t) \le a_i b_i c_i \rho \\ \sum_{i=1}^{6} O_i - \frac{1}{\rho} \sum_{t=1}^{T} q_i(t) \ge 1 \end{cases} \tag{5-10}$$

5.2 算法描述

针对上述问题与模型，本文制定了基于差分进化的多阶段优化方法。在第一阶段，设计了差分进化方法[10][11]在供油模式的基础上优化初始载油量和供油方案。在阶段二，差分进化方法对阶段一获取的高质量解进行进一步优化获取最终供油方案和初始载油量。由于阶段二采用的差分进化方法与问题二所描述的差分进化方法一致，本小节的算法描述略过阶段二的算法描述。

第一阶段的差分进化的编解码方法下图 5-1 所示：

种群中的个体被表示为 6 维的向量 $X = [x_1, x_2, x_3, x_4, x_5, x_6]$，每一个维度 x_i 上取值区间为[0,1]。每一个维度上的数值用来表示第 i 个油箱初始载油量 O_i 。表示关系为 $O_i = x_i a_i b_i c_i$，其实际意义为初始载油量为满载载油量的 x_i 倍。为了将个体表示的初始载油量，进一步的映射成每时刻各个油箱的供油方案。由此采用问题二中设计的**供油仿真过程（见上文图 4-3），同时采用遗传编程训练得到的供油模式的选择规则（见上文图 4-7）**为供油仿真过程中的超启发式函数，以每一个个体所表达的初始载油量为初始状态，以最小供油时间窗口为周期滚动生成供油方案并计算该供油方案下瞬时最大偏差。该最大瞬时偏差的倒数作为个体的适应度值。由此，建立了关于种群中个体与初始载油量、供油方案和最大瞬时偏差之间的编解码关系。

由此，关于油箱初始载油量的优化问题可以建模如下：

$$\min F(X) = \frac{1}{f(X)} + \beta_{punish}$$

$$s.t. \begin{cases} \forall t \in [0,T],\ 0 \le v_i(t) \le U_i \\ t_{oil}(i) \ge 60 \\ \forall t \in [0,T],\ \sum_{i=2}^{5} (v_i(t) == 0) \ge 2 \\ \forall t \in [0,T],\ \sum_{i=1}^{6} (v_i(t) == 0) \ge 3 \\ \forall t \in [0,T],\ \sum_{i=2}^{5} v_i(t) \ge R(t) \\ \forall t \in [0,T],\ 0 \le q_i(t) \le a_i b_i c_i \rho \\ \sum_{i=1}^{6} O_i - \frac{1}{\rho} \sum_{t=1}^{T} q_i(t) \ge 1 \\ 0 \le x_i \le 1 \end{cases} \tag{5-11}$$

其中，$f(X)$为在该初始载油量下，以最小供油时间窗口为周期滚动生成供油方案并计算得到的该供油方案下瞬时最大偏差；由于在解空间中存在大量的不可行解，为了避免浪费大量的不可行解的信息，在目标函数中添加了惩罚参数β_{punish}实现对所有解的评价。

$$\beta_{punish}=\begin{cases}0 & ,X\text{为可行解}\\ 1000 & ,X\text{为不可行解}\end{cases} \tag{5-12}$$

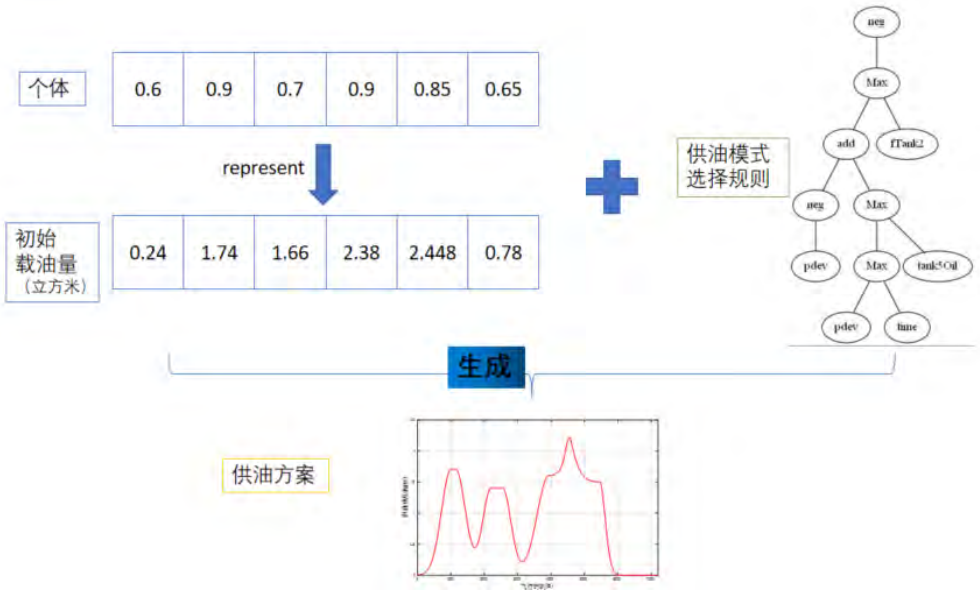


图 5-1 进化算法中的个体编解码方法示意图

进化过程如下：

第一阶段差分进化算法进行交叉、变异、选择等操作直至满足终止条件后输出最佳个体所表达的初始载油量和供油方案。DE/rand1/bin 筛选个体的选择机制为锦标赛选择和精英选择策略。差分进化算法搜索过程如下所示：

步骤一：初始化算法及种群参数，主要包括迭代次数、变异概率、交叉概率、种群规模等参数；

步骤二：通过随机采样的方式生成初代种群，具体对解中每一分量的采样方式表示如下；

$$x_i=\text{rand}() \tag{5-13}$$

步骤三：根据问题目标函数对种群中各个体进行适应度值评价；

步骤四（变异）：通过下面公式完成对新个体每一分量的变异操作：

$$x_i'=\begin{cases}x_j+Fc\cdot(x_k-x_l), & \text{rand}()<Pm\\ x & ,\quad \text{else.}\end{cases} \tag{5-14}$$

其中，x_j,x_k,x_l为从优势个体中随机选择的三个个体中对应位置上的分量，Pm,Fc分别为变异概率和缩放因子。

步骤六（交叉）：通过下面公式完成对变异步骤中产生的新个体的每一分量的交叉操作：

$$x_i'=\begin{cases} x_i & ,\ \text{rand()}<Pc \\ x_i' & ,\ \text{else.} \end{cases} \tag{5-14}$$

步骤六（选择）：根据个体适应度值，采用锦标赛选择和精英选择策略选择优势个体用于产生下一代新种群；

步骤七：检查算法迭代终止判据，若满足，输出当前找到最优解；否则，返回步骤三。

5.3 求解结果及分析

5.3.1 算法的有效性分析

通过上述算法，求解出 6 个油箱的初始载油量如下表所示，飞行器质心与理想质心距离的最大值为 0.038803m，4 个主油箱的总供油量为 6851.9747 kg。

表 5-1 油箱初始载油量

油箱编号	1	2	3	4	5	6
初始载油量(m^3)	0.23665	1.91664	2.09303	2.54540	2.68764	0.79375

差分进行算法的收敛曲线如下图所示，可以从图中看出该算法在较小的迭代次数下，最终优化所得解比初始解的性能提高了至少 60%，因此该算法的优化过程是有效的。

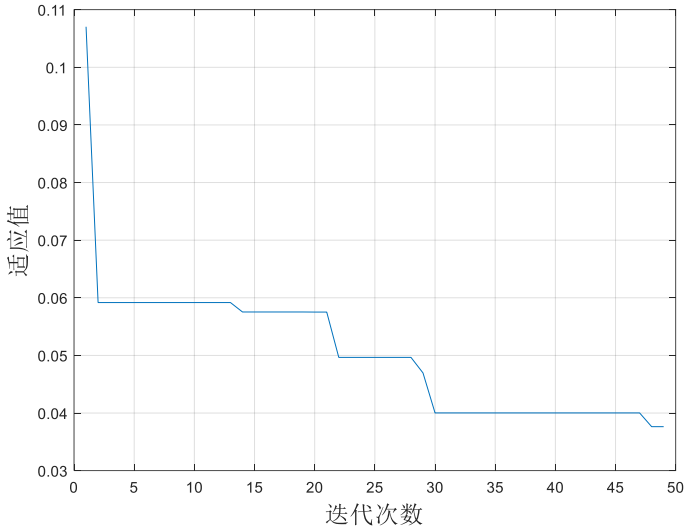


图 5-1 算法收敛曲线

飞行器飞行过程中 6 个油箱各自的供油速度曲线如下，油箱 1 和油箱 6 的供油时间分别与油箱 2 和油箱 5 重合，可见油箱 2 和油箱 5 给发动机供油造成自身油量不足，通过副油箱的供油可以补充自身油箱油量，也有利于后期的质心调节。

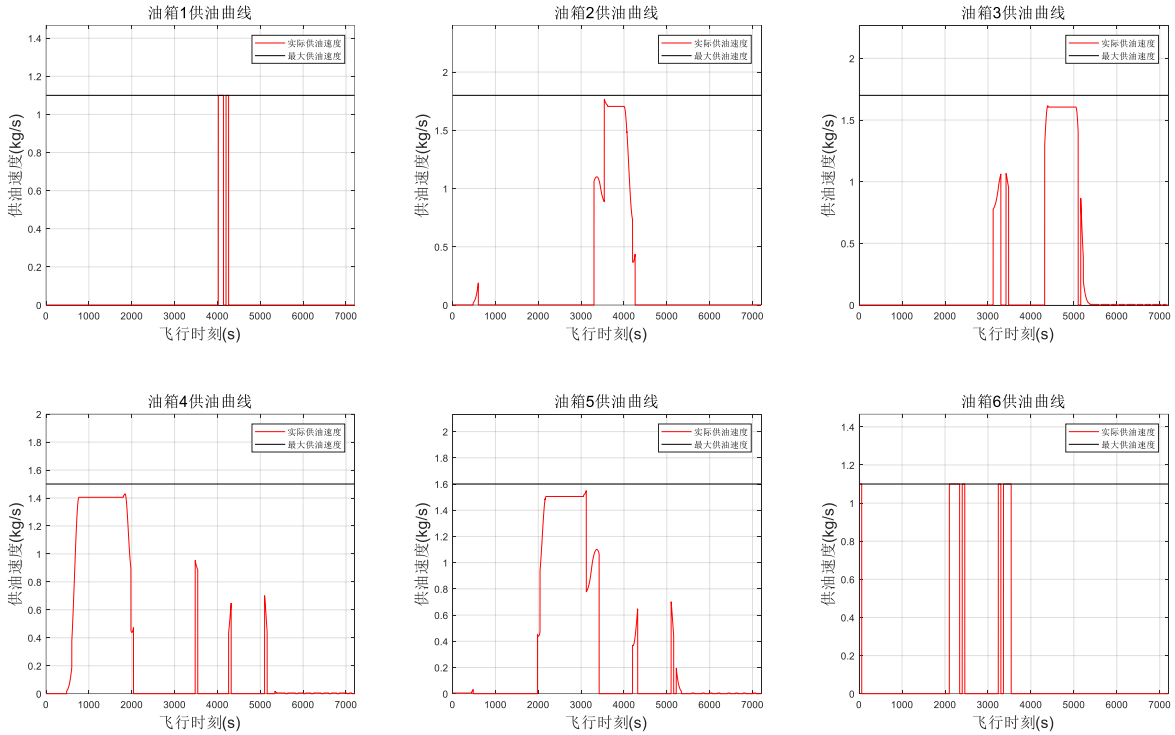


图 5-2 6 个油箱的供油速度曲线

飞行器飞行过程中 4 个主油箱的总供油速度曲线(时间间隔为 1s)如下：

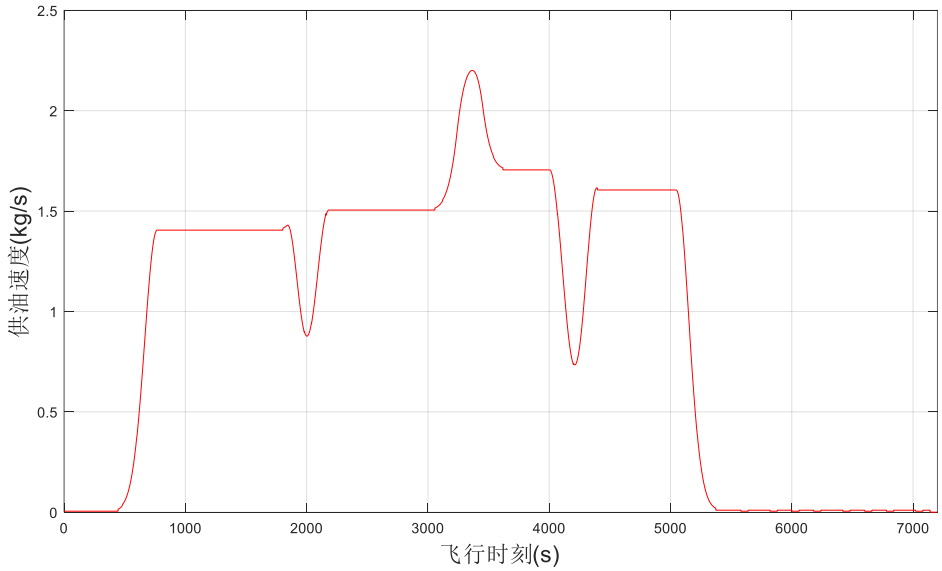


图 5-3 4 个主油箱的总供油速度曲线

飞行器飞行过程中飞行器瞬时质心与理想质心的距离曲线如下，飞行器在飞行中间出现了质心偏离抖动，但是后续质心又保持了平衡，因此，证明了算法在一定程度上可以调节飞行器质心的平衡。

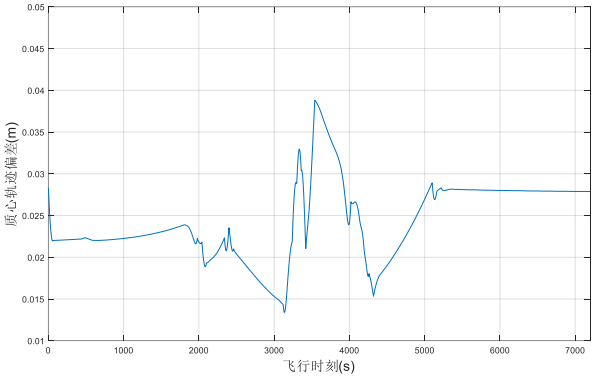


图 5-4 飞行器瞬时质心与理想质心的距离曲线

飞行器飞行过程中各个油箱的使用情况甘特图如下所示，油箱 3、4、5 在飞行后期交替组合供油，调节了飞行器与理想质心间的距离。

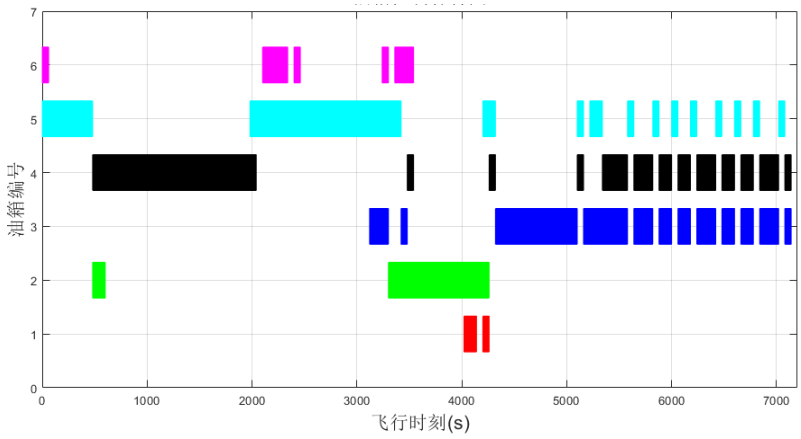


图 5-5 飞行器油箱使用情况甘特图

飞行器质心实际轨迹与理想轨迹图如下所示，能够看到飞行器基本能够沿着理想质心轨迹完成飞行，虽然实际质心与理想质心未完全重合，但并不会出现较大的偏差。

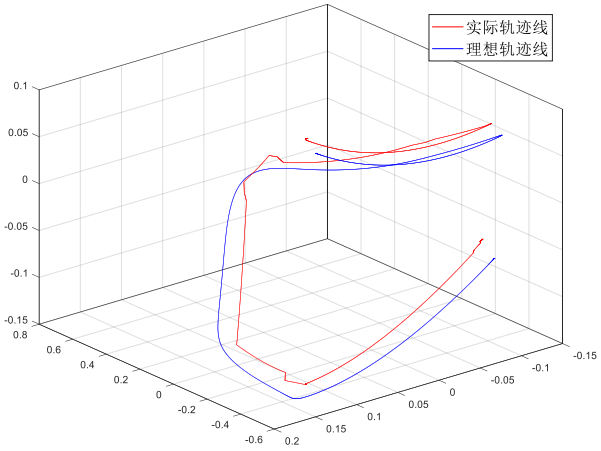


图 5-6 飞行器实际轨迹与理想轨迹曲线

飞行器飞行过程中油量消耗曲线如下图所示，飞行器的从所有油箱初始总共油量为 10.27311m^3，随着飞行过程不断燃油，最终结束时刻油量剩余大于 2 m^3，能够满足题中给

出的剩余油量不少于 1 m^3 的要求。

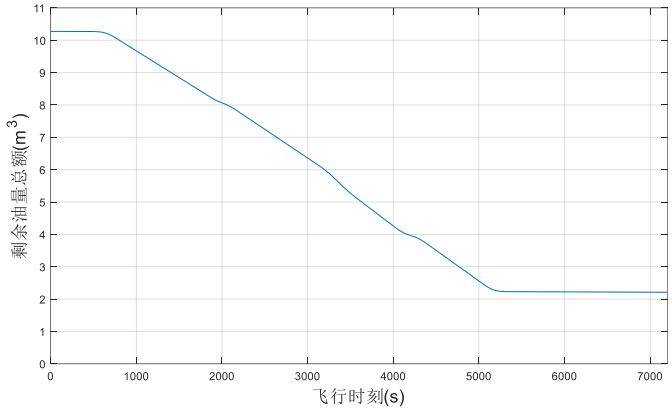


图 5-7 总油量剩余情况

5.3.2 算法的复杂度分析

阶段一算法复杂度：假设油箱的数量为 m，优选供油模式的数量为 w， 迭代次数为 gen，计算优先级的超启发式函数为 $O(1)$ ，供油仿真过程时间复杂度为 $O(m\times w\times gen)$ 。

阶段二算法复杂度：假设油箱的数量为 m，迭代次数为 gen ，供油仿真过程时间复杂度为 $O(m\times gen)$ 。

六、问题四的模型建立与求解

6.1 问题分析及数学模型的建立

6.1.1 问题四的分析

问题四需要制定出当飞行器俯仰角随时间变化时，飞行器在满足约束条件下的油箱供油策略，使得飞行器瞬时质心位置与飞行器（不载油）质心的欧氏距离的最大值达到最小。此题目中的飞行情形为飞行器俯仰角随时间变化，质心位置可通过 3.2.2 和 3.3.3 节中的方法计算。此外，制定的油箱供油策略还需要满足以下约束：

约束 1：第 i 个油箱的供油速度上限为，U_i (U_i >0)，i=1,2,…,6，且供油速度为非负值；

约束 2：每个油箱一次供油的持续时间不少于 60 秒；

约束 3：由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油；

约束 4：各油箱联合供油的总量应至少满足发动机的对耗燃油的需要；

约束 5：主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别为油箱 2 和油箱 5 供油，不能直接向发动机供油；各油箱中油量不能超过油箱内部容积，且不能为负值。

6.1.2 数学模型的构建

问题中的决策变量为每一时刻各油箱的供油策略，即供油速度，定义为 $v_i(t)$，表示油箱 i 在 t 时刻的供油速度。

（1）目标函数：最小化飞行器每一时刻的质心位置与飞行器（不载油）质心位置的最大欧氏距离偏差：

$$\min \max_t \left\| \vec{c}_1(t) - \vec{c}_0(t) \right\|_2 \qquad (6\text{-}1)$$

（3）约束条件：

约束 1：第 i 个油箱的供油速度上限为，U_i (U_i >0)，i=1,2,⋯,6，且供油速度为非负值；

$$\forall t \in [0,T],\ 0 \le v_i(t) \le U_i \tag{6-2}$$

约束 2：每个油箱一次供油的持续时间不少于 60 秒；

$$t_{oil}(i) \ge 60 \tag{6-3}$$

约束 3：由于受到飞行器结构的限制，至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油；

$$\forall t \in [0,T],\ \sum_{i=2}^{5}(v_i(t) == 0) \ge 2 \tag{6-4}$$

$$\forall t \in [0,T],\ \sum_{i=1}^{6}(v_i(t) == 0) \ge 3 \tag{6-5}$$

约束 4：各油箱联合供油的总量应至少满足发动机的对耗燃油的需要；

$$\forall t \in [0,T],\ \sum_{i=2}^{5} v_i(t) \ge R(t) \tag{6-6}$$

约束 5：主油箱 2、3、4、5 可直接向发动机供油，油箱 1 和油箱 6 作为备份油箱分别为油箱 2 和油箱 5 供油，不能直接向发动机供油；各油箱中油量不能超过油箱内部容积，且不能为负值。

$$\forall t \in [0,T],\ 0 \le q_i(t) \le a_i b_i c_i \rho \tag{6-7}$$

其中，

$$\begin{aligned} q_i(t) &= O_i\rho - \int_0^t v_i(t)dt +, i = 1,3,4,6 \\ q_i(t) &= O_i\rho - \int_0^t v_i(t)dt + \int_0^t v_1(t)dt, i = 2 \\ q_i(t) &= O_i\rho - \int_0^t v_i(t)dt + \int_0^t v_6(t)dt, i = 5 \end{aligned} \tag{6-8}$$

（3）综上，可建立如下的数学模型：

$$\min \max_t \left\| \vec{c}_1(t) - \vec{c}_0(t) \right\|_2$$

$$s.t. \begin{cases} \forall t \in [0,T],\ 0 \le v_i(t) \le U_i \\ t_{oil}(i) \ge 60 \\ \forall t \in [0,T],\ \sum_{i=2}^{5}(v_i(t) == 0) \ge 2 \\ \forall t \in [0,T],\ \sum_{i=1}^{6}(v_i(t) == 0) \ge 3 \\ \forall t \in [0,T],\ \sum_{i=2}^{5} v_i(t) \ge R(t) \\ \forall t \in [0,T],\ 0 \le q_i(t) \le a_i b_i c_i \rho \end{cases} \tag{6-9}$$

6.2 算法描述

根据上述分析建模可知，该问题与问题二相类似，因此本文仍采用基于多阶段的混合超启发式方法。本问题中与问题二的区别在于飞行器俯仰角随时间变化，质心计算模型不同，需要考虑俯仰角对质心的影响，即算法中目标函数的不同，需采用带有俯仰角的质心

计算模型作为目标函数。该方法包含两个阶段：阶段一，遗传编程的超启发方法获取一个高质量初始供油方案；阶段二，采用差分进化方法对获取的高质量解进行进一步优化获取最终方案。由于阶段二采用的差分进化方法与问题二所描述的差分进化方法一致，本小节的算法描述略过阶段二的算法描述。

阶段一：遗传编程的超启发式方法获取一个高质量初始供油方案

遗传编程的超启发式流程：首先，种群内的个体被随机初始化。接着，新的子代种群在交叉、变异、重构算子下生成。同时对每一个超启发选择规则个体进行评价，再通过锦标赛规则筛选子代种群和父代获取新的种群。判断是否满足终止条件，若不满足重复上述步骤，若满足，遗传编程的超启发方法输出最佳个体所代表的最佳选择规则。最后通过优化得到的选择规则个体生成供油方案。

建立的供油仿真解码方法分为三个环节，分别为：初始化过程、决策过程、执行过程。在初始化环节中：油箱的各个参数和超启发式的输油模式的选择规则函数 *fh* 输入到解码方法中，同时供油仿真时间设置为 0. 在决策环节中，超启发式 *fh* 根据当前的状态信息（当前时刻、当前飞行器的俯仰角、当前时刻在整体时间轴占比、当前时刻下油箱中剩余的油量、执行该供油模式后产生的偏差、当前时刻下质心的位置、未来的最小总耗油量等）计算候选供油模式下的启发式优先级。选择启发式优先级最高的供油模式 mode*作为下一个时间窗口所采用的供油模式。在下一个时间窗口执行供油模式 mode*后，判断是否完成到达任务终止时间。若是，则输出在整个飞行任务阶段时间内供油方案和最大瞬时偏差，若为否，更新当前所有的状态信息重复上述过程。值得注意的是在该问题中，超启发式 *fh* 根据当前状态信息包括当前飞行器的俯仰角信息，这是在问题二中不做分析综合考虑的特征参量。同时，在问题二中质心计算的方法是该问题下的一种特例，该问题采用的详细质心计算方法见问题一。

6.3 求解结果及分析

6.3.1 算法的有效性分析

通过上述算法，求解出飞行器俯仰角随时间变化时，飞行器瞬时质心与飞行器（不载油）质心的最大距离偏差为：0.13798m，4 个主油箱的总供油量为 7089.8452kg。

飞行器飞行过程中 6 个油箱各自的供油速度曲线如下：

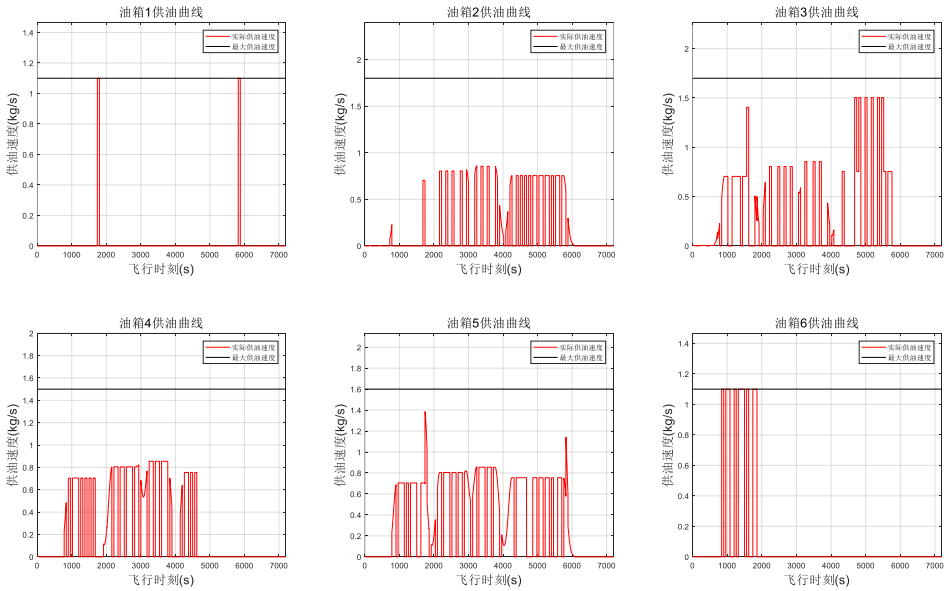


图 6-1 6 个油箱的供油速度曲线

飞行器飞行过程中 4 个主油箱的总供油速度曲线(时间间隔为 1s)与计划耗油速度曲线如下图所示。从图中能够发现，本文指定的供油策略能够满足计划耗油速度，且不会造成很大的燃料浪费。

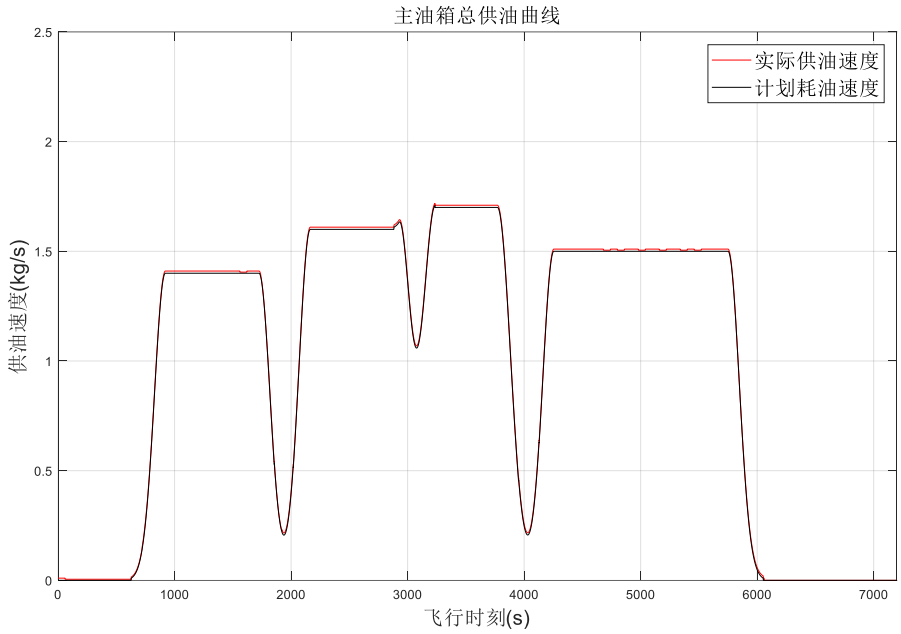


图 6-2 4 个主油箱的总供油速度曲线与计划耗油速度

飞行器飞行过程中飞行器瞬时质心与理想质心的距离曲线如下图所示。从图中能够发现飞行器质心与理想质心的距离偏差基本被缩小到了 0.15 米以内。

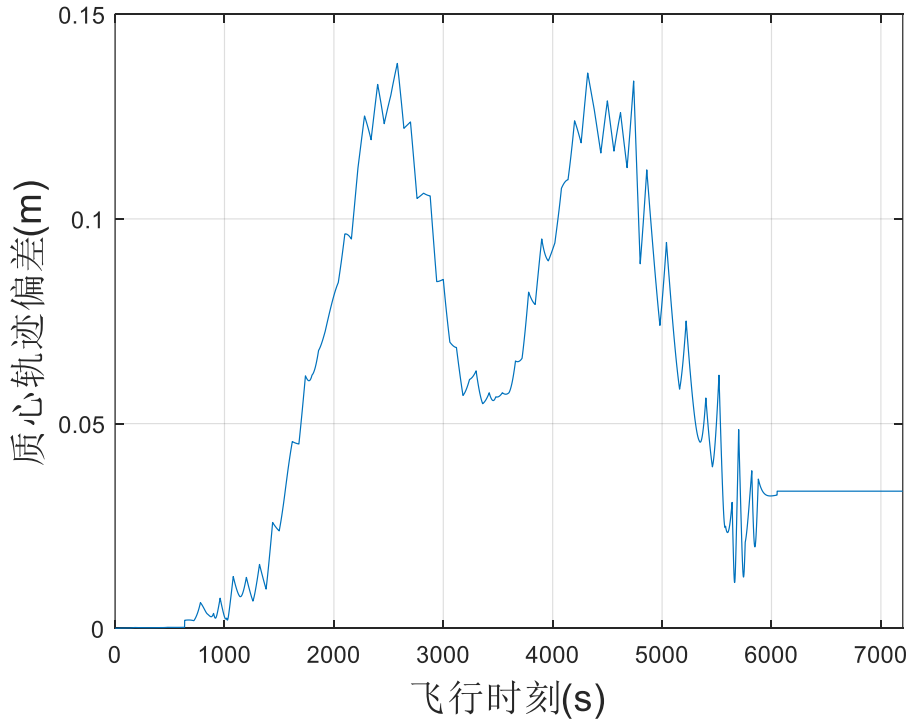


图 6-3 飞行器瞬时质心与理想质心的距离曲线

飞行器飞行过程中各个油箱的使用情况甘特图如下图所示。位于飞行器不同位置的油箱在飞行器飞行过程中进行着密集的交替供油动作，尽可能保证实际质心不偏离飞行器质心位置。

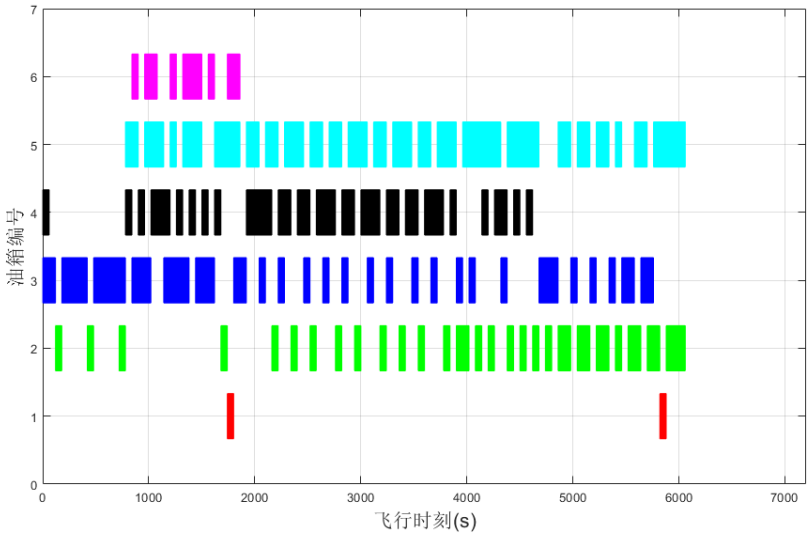


图 6-4 飞行器油箱使用情况甘特图

6.3.2 算法的复杂度分析

本文设计的基于遗传编程的超启发式算法中，其采用的终端集合（考虑的当前及预测的状态信息）和函数集合如下表所示：：

表 6-1 基于遗传编程的超启发式算法终端集合和函数集合

表示	描述
pdev	采用该供油模式会在下一个时间窗口产生的最大瞬时偏差
tank2Oil	采用该供油模式，在下一个时间窗口油箱 2 在当前时刻的剩余油量（m^3）
tank5Oil	采用该供油模式，在下一个时间窗口油箱 5 在当前时刻的剩余油量（m^3）
tank6Oil	采用该供油模式，在下一个时间窗口油箱 6 在当前时刻的剩余油量（m^3）
pTank2Oil	采用该供油模式，在下一个时间窗口油箱 2 在当前时刻的剩余油量与初始油量的比值
pTank5Oil	采用该供油模式，在下一个时间窗口油箱 5 在当前时刻的剩余油量与初始油量的比值
pTank6Oil	采用该供油模式，在下一个时间窗口油箱 6 在当前时刻的剩余油量与初始油量的比值
angle	当前时刻的飞行器俯仰角度
sumOil	采用该供油模式，在下一个时间窗口全部油箱内油量总和
time	当前时刻

函数集合	add, sub, negative, max, mul, div

其中一次进化得到的遗传编程树所表示的供油模式选择规则，如下图 6-5 所示。从遗传编程优化得到的选择规则可知，当候选的供油模式产生的最大瞬时偏差越大时，其被选择的优先级越低；当候选的供油模式产生的最大瞬时偏差越大时，其被选择的优先级越低；当候选供油模式对油箱中的油量消耗越小时，其被选择的优先级越高。尽管所设计的特征参量没有全部用在最佳的超启发式规则中，但该超启发式进化算法框架相对于人为设计的贪心算法更能适应给定数据的变化。**该多阶段的混合超启发式方法具有良好的适用性和扩展性。**

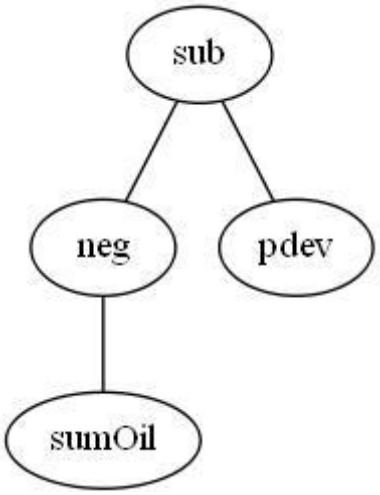


图 6-5 供油模式选择规则的超启发式函数树状表示图

算法复杂度：

遗传编程算法复杂度：假设油箱的数量为m，优选供油模式的数量为w，迭代次数为gen，计算优先级的函数为$O(1)$，供油仿真过程时间复杂度为$O(n\times w\times gen)$。

差分进化算法复杂度：假设油箱的数量为m，迭代次数为gen，供油仿真过程时间复杂度为$O(m\times gen)$。

七、总结

本文首先分析了飞行器系统中所涉及的要素，从问题的决策变量、约束和优化目标出发，建立了各问题的详细数学模型，并基于飞行器俯仰角、油量等参数，提出了质心计算方法。本文中的飞行器质心平衡供油策略优化问题具有供油油箱数量上限等约束，且供油策略规划时刻数量多等因素，造成问题约束强、复杂度高、规模大，采用常规方法进行搜索容易产生大量不可行解，从而带来大量的计算代价。因此，本文针对各问题提出了问题知识启发的优化方法。具体地，针对问题二，提出了基于供油模式的混合超启发式(hybrid hyper-heuristic)方法，该方法以每 60 秒为一个周期，每个周期内存在一种供油模式，根据问题特征一共设计了 35 种供油模式，每种模式包括采用的供油箱及其供油速度，遗传编程方法在基于供油模式的启发式搜索空间优化选择供油方案，并利用差分进化方法优化每一个时刻的供油速度。针对问题三，飞行器各油箱初始油量未知，提出了基于差分进化的多阶段优化方法，差分进化方法来优化油箱初始载油量和供油方案，其中供油方案的优化方法采用了第二问的方法。对于问题四，采用了与问题二类似的混合超启发式方法，但因

需考虑俯仰角因素，在目标函数值计算部分采用了俯仰角不为 0 的质心计算模型。

问题一到四的结论如下：

问题一：在附件 2 的测试数据下，飞行器质心距飞行器坐标系原点的最大欧式距离为 1.109137m，并绘制了飞行器质心随时间变化的曲线。

问题二：在附件 3 的数据下测试方法，飞行器瞬时质心与理想质心距离的最大值为 0.10233m，4 个主油箱的总供油量为 6483.8242kg。

问题三：在附件 4 的数据下测试方法，飞行器各油箱初始载油量分别为 0.23665 m^3、1.91664 m^3、2.09303 m^3、2.54540 m^3、2.68764 m^3、0.79375 m^3，总初始载油量为 10.27311m^3，飞行器瞬时质心与理想质心距离的最大值为 0.038803m，4 个主油箱的总供油量为 6851.9747kg。

问题四：在附件 5 的数据下测试方法，飞行器瞬时质心与飞行器（不载油）质心的最大距离偏差为 0.13798 米，4 个主油箱的总供油量为 7089.8452kg。

本文中所有算法的代码与结果表均作为附件上传。

关于飞行器质心平衡供油策略优化的指导性意见

已知油箱初始燃油量的飞行器质心平衡供油策略优化问题从整体上看是一个带时序约束的复杂混合整数优化问题，我们曾尝试双层优化思路优化每一时刻点的各油箱供油策略，即包括：在上层考虑油箱位置、剩余油量、最大供油速度、供油时间及质心变化方向等问题知识和约束，通过启发式构造规则选择该时刻点的供油油箱，保证使用选择的油箱能够产生可行供油策略；在下层使用差分进化算法迭代调整各油箱的供油策略，最小化飞行器的质心偏差量。

然而，在设计算法优化过程中，由于问题的复杂性及油量约束较为严苛，虽然算法在飞行器飞行前半段（0~4000s）能够通过构造出较好的前期供油方案，但由于耗油量较大，导致在飞行器飞行后半段剩余油量不足，无法产生供油方案。虽然在改进的启发式规则中同时考虑了燃油消耗和质心偏差，但依然会无法构造出可行解。因此，团队对在上层用启发式规则确定油箱选择方案的方法产生了质疑；最终，决定采用**基于供油模式的混合超启发式方法**，对上层的供油方案进行搜索优化。最后，一系列的高质量的可行解的产生证实了我们最终方法的可行性。

基于上述的算法设计经验，给出关于飞行器质心平衡供油策略优化的意见如下：

1、飞行器供油油箱的选择是同时影响发动机供油总量和飞行器质心的因素。合适的油箱选择可以扩大后半段的飞行器质心调整范围，也会减小主油箱向发动机供油总量；

2、通过总结目前已知可优解的规律，发现飞行器油箱选择在飞行器飞行的前半段（即油量充足时）倾向于多个油箱分时调整；

3、本问题可以尝试预先对轨迹特点进行分析，然后合适的油箱并为以后可能出现的轨迹保留适当油量。

八、参考文献

[1] 钱向农, 孙康, 禹继晖, 等人. 现代飞机燃油管理系统分析研究[C]. 航空智能装备与试验测试技术年会, pp.1-5, 2017.

[2] 李钧, 王忠群, 刘涛. 基于遗传编程的网格资源调度算法[J]. 计算机技术与发展, Vol. 18, No. 2, pp. 129-132, 2008.

[3] 王彩虹, 韩国震, 吕海利, 王恺, 陈建华. 基于改进遗传编程的装配作业车间调度研究, Vol. 16, pp. 76-77, 2017.

[4] 王忠群, 李钧, 刘涛, 王勇. 基于遗传编程和效用最优的网格资源调度及仿真[J]. 计算机技术与应用进展, pp. 1502-1507, 2007.
[5] D Jakobovic, K Marasovic. Evolving priority scheduling heuristics with genetic programming[J]. Applied Soft Computing, Vol. 12, No. 9, pp. 2781-2789.
[6] S Nguyen, Y Mei, M Zhang. Genetic programming for production scheduling: a survey with a unified framework[J]. Complex & Intelligent Systems, Vol. 3, No. 1, pp. 41-66, 2017.
[7] 董明刚. 基于差分进化算法的优化算法及应用研究[D]. 浙江: 浙江大学, 2012.
[8] 王万良, 范丽霞, 徐新黎，等. 多目标差分进化算法求解柔性作业车间批量调度问题[J]. 计算机集成制造系统, Vol. 19, No. 10, pp. 2481-2492, 2013.
[9] 翁志远, 方杰, 孔敏, 程颖. 改进差分进化算法的作业车间调度优化策略[J]. 控制工程, Vol. 24, No. 6, pp.1283-1285, 2017.
[10] N Damak, B Jarboui, P Siarry 等. Differential evolution for solving multi-mode resource-constrained project scheduling problems[J]. Complex & Intelligent Systems, Vol. 36, No. 9, pp. 2653-2659, 2009.
[11] W Warisa, K Voratas. Differential Evolution Algorithm for Job Shop Scheduling Problem[J]. Industrial Engineering & Management Systems An International Journal, Vol.10, No. 3, pp. 203-208, 2011.

九、附录

完整代码较多，处于篇幅考虑，附录仅包含各问题中关键函数的完整代码及其他函数的函数说明，其他代码见附件。

9.1 问题一代码

主函数：功能计算质心

```
load F_data.mat;
used_oil=Data_P1(:,2:7)/Density_oil;
theta_t=Data_P1(:,8)*pi/180;
%% 开始算
sum_oil=zeros(1,6);
remain_oil=zeros(1,6);
det_cent=zeros(length(used_oil(:,1)),3);
for t=1:length(used_oil(:,1))
    for i=1:6
        sum_oil(i)=sum(used_oil(1:t,i));
        remain_oil(i)=Oilbox_initial(i)-sum_oil(i);
    end
    remain_oil(2)=sum_oil(1)+remain_oil(2);
    remain_oil(5)=sum_oil(6)+remain_oil(5);
    det_cent(t,:)=cal_cent(Oilbox_geo,remain_oil,Oilbox_xyz,Mass_air,Density_oil,theta_t(t));
end
function det_cent=cal_cent(Oilbox_geo,Oilbox_v,Oilbox_xyz,Mass_air,Density_oil,theta)
%% 本函数主要为了计算存在倾斜角之后，质心的变化情况。
det_cent=zeros(1,3);
%% 计算旋转后各个油箱
```

```
Oilbox_xyz_new=zeros(size(Oilbox_xyz));
Oilbox_xyz_new(:,2)=Oilbox_xyz(:,2);
for i=1:6
    det_xz=det_single(theta,Oilbox_geo(i,:),Oilbox_v(i));
    Oilbox_xyz_new(i,1)=Oilbox_xyz(i,1)+det_xz(1);
    Oilbox_xyz_new(i,3)=Oilbox_xyz(i,3)+det_xz(2);
end
Mass_sum=sum(Oilbox_v)*Density_oil+Mass_air;
for i=1:6
    det_cent(1)=Oilbox_v(i)*Density_oil*Oilbox_xyz_new(i,1)+det_cent(1);
    det_cent(2)=Oilbox_v(i)*Density_oil*Oilbox_xyz_new(i,2)+det_cent(2);
    det_cent(3)=Oilbox_v(i)*Density_oil*Oilbox_xyz_new(i,3)+det_cent(3);
end
det_cent=det_cent/Mass_sum;
function det_xz=det_single(theta,geo,v)
% 本函数主要为了计算在已知剩余油量，油箱相对于大地坐标系倾角的条件下，
% 计算油箱内剩余油箱质心，相对于原油箱质心在大地坐标系下的偏移量 det_xz，即油箱质心（大地坐标系）=det_xz+旋转矩阵*原油箱质心坐标
% 由于只考虑 xz 方向的倾斜角，所以略去 y 方向的计算。
%% 参数说明
theta:旋转角度
geo:该油箱的几何参数
v:油箱的油量
density:油的密度
det_xyz:在某一角度下的油箱质心的相对偏移量（相对于空油箱质心）
%}
l=geo(1);%箱体长度
w=geo(2);%箱体宽度
h=geo(3);%箱体高度
S=h*l;%整个箱体在 xz 方向的投影面积
S1=v/w;%箱体油量在 xz 方向的投影面积
theta1=abs(theta);
h2t=h^2/2/tan(theta1);
l2t=l^2*tan(theta1)/2;
minhl=min(h2t,l2t);
rot_mat=[cos(0),-sin(0);sin(0),cos(0)];
%% 开始分类讨论
if(theta==0)
    det_x0=l/2-l/2;
    det_z0=S1/l/2-h/2;
    det_xz=rot_mat*[det_x0;det_z0];
end
%% 计算旋转后的油箱质心相对于大地坐标系的偏移向量
if(theta~=0)
```

```
    if(S1<minhl)%投影面积很小的时候
        x_f=sqrt(2*S1/tan(theta1))/3;
        y_f=sqrt(2*S1*tan(theta1))/3;
        det_x0=x_f-l/2;
        det_z0=y_f-h/2;
        det_xz=rot_mat*[det_x0;det_z0];
    elseif(minhl>(S-S1))%投影面积很大的时候
        S2=S-S1;
        x_f=sqrt(2*S2/tan(theta1))/3;
        y_f=sqrt(2*S2*tan(theta1))/3;
        det_x0=(l*l*h/2-S2*(l-x_f))/S1-l/2;
        det_z0=(h*h*l/2-S2*(h-y_f))/S1-h/2;
        det_xz=rot_mat*[det_x0;det_z0];
    else
        if(h2t<l2t)
            S11=h2t;
            S12=S1-S11;
            x_f=sqrt(2*S11/tan(theta1))/3;
            y_f=sqrt(2*S11*tan(theta1))/3;
            det_x0=(S11*(S12/h+x_f)+S12*S12/2/h)/S1-l/2;
            det_z0=(S11*y_f+S12*h/2)/S1-h/2;
            det_xz=rot_mat*[det_x0;det_z0];
        else
            S11=l2t;
            S12=S1-S11;
            x_f=sqrt(2*S11/tan(theta1))/3;
            y_f=sqrt(2*S11*tan(theta1))/3;
            det_x0=(S11*(x_f)+S12*l/2)/S1-l/2;
            det_z0=(S11*(S12/l+y_f)+S12*S12/2/l)/S1-h/2;
            det_xz=rot_mat*[det_x0;det_z0];
        end
    end
end
if(theta<0)
        det_x0=-det_x0;
        det_xz=rot_mat*[det_x0;det_z0];
end
end
```

9.2 问题二代码

```
# from simulator2 import Point3D, AirPlaneSimulator
from simulator2Pro import Point3D, AirPlaneSimulator
import xlrd
worksheet = xlrd.open_workbook(u'附件 3-问题二数据.xlsx')
sheet_names = worksheet.sheet_names()
```

```
# print(sheet_names)
sheet = worksheet.sheet_by_name('发动机耗油速度')
oilCostLst = sheet.col_values(1)
oilCostLst.remove('耗油速度(kg/s)')
# print(oilCostLst)
# exit()
all_pointLst = []
sheet = worksheet.sheet_by_name('飞行器理想质心数据')
for i in range(3):
    unitLst = sheet.col_values(i + 1)
    if 'X 坐标（米）' in unitLst:
        unitLst.remove('X 坐标（米）')
    if 'y 坐标（米）' in unitLst:
        unitLst.remove('y 坐标（米）')
    if 'z 坐标（米）' in unitLst:
        unitLst.remove('z 坐标（米）')
    all_pointLst.append(unitLst)
idealCentroidLst = []
for i in range(len(all_pointLst[0])):
    unitPoint = Point3D(all_pointLst[0][i],all_pointLst[1][i],all_pointLst[2][i])
    idealCentroidLst.append(unitPoint)
# print(idealCentroidLst)
all_oilSupplyLst = []

g_air_plane_simulator = AirPlaneSimulator(idealCentroidLst,oilCostLst, all_oilSupplyLst)
import operator
import math
import random
import numpy
from deap import algorithms
from deap import base
from deap import creator
from deap import tools
from deap import gp

# Define new functions
def protectedDiv(left, right):
    try:
        return left / right
    except ZeroDivisionError:
        return 1
def max(a, b):
    if a>=b:
        return a
```

```
        else:
            return b
def min(a, b):
        if a<=b:
            return a
        else:
            return b
pset = gp.PrimitiveSet("MAIN", 12)
pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.sub, 2)
pset.addPrimitive(operator.mul, 2)
pset.addPrimitive(protectedDiv, 2)
pset.addPrimitive(operator.neg, 1)
# pset.addPrimitive(math.cos, 1)
# pset.addPrimitive(math.sin, 1)
pset.addPrimitive(min, 2,name="Min")
pset.addPrimitive(max, 2,name="Max")
# pset.renameArguments(ARG0='x')
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", gp.PrimitiveTree, fitness=creator.FitnessMin)
toolbox = base.Toolbox()
toolbox.register("expr", gp.genHalfAndHalf, pset=pset, min_=1, max_=3)
toolbox.register("individual", tools.initIterate, creator.Individual, toolbox.expr)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("compile", gp.compile, pset=pset)
def evalSymbReg(individual):
        # Transform the tree expression in a callable function
        func = toolbox.compile(expr=individual)
        fitness = g_air_plane_simulator.run(func)
        return fitness,
        # Evaluate the mean squared error between the expression
        # and the real function : x**4 + x**3 + x**2 + x
        # sqerrors = ((func(x) - x ** 4 - x ** 3 - x ** 2 - x) ** 2 for x in points)
        # return math.fsum(sqerrors) / len(points),

toolbox.register("evaluate", evalSymbReg)
toolbox.register("select", tools.selTournament, tournsize=7)
toolbox.register("mate", gp.cxOnePoint)
toolbox.register("expr_mut", gp.genFull, min_=0, max_=2)
toolbox.register("mutate", gp.mutUniform, expr=toolbox.expr_mut, pset=pset)
toolbox.decorate("mate", gp.staticLimit(key=operator.attrgetter("height"), max_value=5))
toolbox.decorate("mutate", gp.staticLimit(key=operator.attrgetter("height"), max_value=5))
import numpy as np
def main():
```

```
randomSeed = 3
random.seed(randomSeed)
np.random.seed(randomSeed)
ind = toolbox.individual()
pop = toolbox.population(n = 10)
hof = tools.HallOfFame(1)
stats = tools.Statistics(lambda ind: ind.fitness.values)
stats.register("avg", numpy.mean)
stats.register("std", numpy.std)
stats.register("min", numpy.min)
stats.register("max", numpy.max)
logbook = tools.Logbook()
logbook.header = "gen", "evals", "std", "min", "avg", "max"
f_con = open('rgp_'+ str(randomSeed)+'.txt','w')
CXPB, MUTPB, NGEN = 0.5, 0.2, 40
cxpb = 0.8
mutpb = 0.15
# Evaluate the entire population
for ind in pop:
    ind.fitness.values = toolbox.evaluate(ind)
hof.update(pop)
record = stats.compile(pop)
logbook.record(gen=0, evals=len(pop), **record)
print(logbook.stream)
for g in range(1, NGEN):
    # Select the offspring
    offspring = toolbox.select(pop, len(pop))
    # Clone the offspring
    offspring = [toolbox.clone(ind) for ind in offspring]

    # Apply crossover and mutation
    for i in range(1, len(offspring), 2):
        if random.random() < cxpb:
            offspring[i - 1], offspring[i] = toolbox.mate(offspring[i - 1], offspring[i])
            del offspring[i - 1].fitness.values, offspring[i].fitness.values
    for i in range(len(offspring)):
        if random.random() < mutpb:
            offspring[i], = toolbox.mutate(offspring[i])
            del offspring[i].fitness.values
    # Evaluate the individuals with an invalid fitness
    invalids = [ind for ind in offspring if not ind.fitness.valid]
    for ind in invalids:
        ind.fitness.values = toolbox.evaluate(ind)
    # Replacement of the population by the offspring
```

```
            pop = offspring
            hof.update(pop)
            record = stats.compile(pop)
            logbook.record(gen=g, evals=len(invalids), **record)
            print(logbook.stream)
            f_con.write('gen '+ str(g) + ' ')
            f_con.write(' '+ str(record['min']) + '\n')
            f_con.write(str(hof[0]) + '\n')
            f_con.write('min    ' + str(hof[0].fitness.values[0]) + '\n')
            f_con.flush()
            if g%7 == 1:
                func = toolbox.compile(expr=hof[0])
                g_air_plane_simulator.saveDataBool = True
                fitness = g_air_plane_simulator.run(func)
                g_air_plane_simulator.saveDataBool = False
    print('Best individual : ', hof[0][0], hof[0].fitness)
    f_con.write(str(hof[0]) + '\n')
    f_con.write('min    ' + str(hof[0].fitness.values[0]) + '\n')
    return pop, stats, hof
    # print log
    return pop, log, hof
if __name__ == "__main__":
    main()
```

9.3 问题三代码

```
import random
import array
import numpy
from itertools import chain
from deap import base
from deap import benchmarks
from deap import creator
from deap import tools
# Problem dimension
NDIM = 6
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", array.array, typecode='d', fitness=creator.FitnessMin)
def mutDE(y, a, b, c, f):
    size = len(y)
    for i in range(len(y)):
        y[i] = a[i] + f * (b[i] - c[i])
    return y
def cxBinomial(x, y, cr):
    size = len(x)
```

```
        index = random.randrange(size)
        for i in range(size):
            if i == index or random.random() < cr:
                x[i] = y[i]
        return x
def cxExponential(x, y, cr):
        size = len(x)
        index = random.randrange(size)
        # Loop on the indices index -> end, then on 0 -> index
        for i in chain(range(index, size), range(0, index)):
            x[i] = y[i]
            if random.random() < cr:
                break
        return x
import simulator3Pro
from simulator3Pro import Point3D,Size3D
import xlrd
worksheet = xlrd.open_workbook(u'附件 4-问题三数据.xlsx')
sheet_names = worksheet.sheet_names()
# print(sheet_names)
sheet = worksheet.sheet_by_name('发动机耗油数据')
oilCostLst = sheet.col_values(1)
oilCostLst.remove('耗油速度(kg/s)')
# print(oilCostLst)
# exit()
all_pointLst = []
sheet = worksheet.sheet_by_name('飞行器理想质心')
for i in range(3):
        unitLst = sheet.col_values(i + 1)
        if 'X 坐标（米）' in unitLst:
            unitLst.remove('X 坐标（米）')
        if 'y 坐标（米）' in unitLst:
            unitLst.remove('y 坐标（米）')
        if 'z 坐标（米）' in unitLst:
            unitLst.remove('z 坐标（米）')
        all_pointLst.append(unitLst)
idealCentroidLst = []
for i in range(len(all_pointLst[0])):
        unitPoint = Point3D(all_pointLst[0][i],all_pointLst[1][i],all_pointLst[2][i])
        idealCentroidLst.append(unitPoint)
# print(idealCentroidLst)
import xlrd
worksheet = xlrd.open_workbook(u'附件 2-问题一数据.xlsx')
sheet_names= worksheet.sheet_names()
```

```
print(sheet_names)
sheet = worksheet.sheet_by_name('油箱供油曲线')
timeLst = sheet.col_values(0)
timeLst.remove('时间(s)')
# print(timeLst)
all_oilSupplyLst = []
for i in range(6):
    unitLst = sheet.col_values(i + 1)
    unitLst.remove(str(i + 1 ) +'号油箱(kg/s)')
    all_oilSupplyLst.append(unitLst)
print('sum oil cost = ', numpy.sum(oilCostLst)/850)
g_simulator = simulator3Pro.AirPlaneSimulator(idealCentroidLst,oilCostLst,[])
tankSizeLst = [Size3D(1.5, 0.9, 0.3), Size3D(2.2, 0.8, 1.1), Size3D(2.4, 1.1, 0.9),
                        Size3D(1.7, 1.3, 1.2), Size3D(2.4, 1.2, 1), Size3D(2.4, 1, 0.5)]
tankMaxLst = [_.x *_.y *_.z for _ in tankSizeLst]
print(tankMaxLst)
print(numpy.sum(tankMaxLst))
weightFuncLst = tankMaxLst
# weightFuncLst = []*
def griewank(individual):
    sum = 0
    for i in range(6):
        sum = sum + weightFuncLst[i] * individual[i]
    # print(sum)
    if sum < 9.4:
        return 999999999,
    inputIndividul = []
    for i in range(6):
        if individual[i] > 0.99:
            inputIndividul.append(0.99)
        elif individual[i] < 0.01:
            inputIndividul.append(0.01)
        else:
            inputIndividul.append(individual[i])
    tankInitOilLst = []
    for i in range(6):
        tankInitOilLst.append(weightFuncLst[i] * inputIndividul[i])
    fitnessValue = g_simulator.run(tankInitOilLst)
    return fitnessValue,
toolbox = base.Toolbox()
toolbox.register("attr_float", random.uniform, 0, 1)
toolbox.register("individual", tools.initRepeat, creator.Individual, toolbox.attr_float, NDIM)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("mutate", mutDE, f=0.8)
```

```
toolbox.register("mate", cxExponential, cr=0.8)
toolbox.register("select", tools.selTournament, k=3, tournsize    = 10)
toolbox.register("evaluate", griewank)
def main():
    # Differential evolution parameters
    MU = NDIM * 10
    NGEN = 50
    randomSeed = 2
    numpy.random.seed(randomSeed)
    random.seed(randomSeed)
    pop = toolbox.population(n=MU);
    hof = tools.HallOfFame(1)
    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("avg", numpy.mean)
    stats.register("std", numpy.std)
    stats.register("min", numpy.min)
    stats.register("max", numpy.max)
    logbook = tools.Logbook()
    logbook.header = "gen", "evals", "std", "min", "avg", "max"
    # Evaluate the individuals
    fitnesses = toolbox.map(toolbox.evaluate, pop)
    for ind, fit in zip(pop, fitnesses):
        ind.fitness.values = fit
    f_con = open('r_'+ str(randomSeed)+'.txt','w')
    record = stats.compile(pop)
    logbook.record(gen=0, evals=len(pop), **record)
    print(logbook.stream)
    f_con.write(str(logbook.stream) + '\n')
    for g in range(1, NGEN):
        children = []
        for agent in pop:
            # We must clone everything to ensure independance
            a, b, c = [toolbox.clone(ind) for ind in toolbox.select(pop)]
            x = toolbox.clone(agent)
            y = toolbox.clone(agent)
            y = toolbox.mutate(y, a, b, c)
            z = toolbox.mate(x, y)
            del z.fitness.values
            children.append(z)
        fitnesses = toolbox.map(toolbox.evaluate, children)
        for (i, ind), fit in zip(enumerate(children), fitnesses):
            ind.fitness.values = fit
            if ind.fitness > pop[i].fitness:
                pop[i] = ind
```

```
            hof.update(pop)
            # f_con.write(str(hof[0]) + '\n')
            # f_con.write('min    ' + str(hof[0].fitness.values[0]) + '\n')
            record = stats.compile(pop)
            logbook.record(gen=g, evals=len(pop), **record)
            print(logbook.stream)
            # f_con.write(str(logbook.stream) + '\n')
            f_con.write('gen '+ str(g) + ' ')
            f_con.write(' '+ str(record['min']) + '\n')
            f_con.flush()
        print("Best individual is ", hof[0])
        print("with fitness", hof[0].fitness.values[0])
        f_con.write(str(hof[0]) + '\n')
        f_con.write('min    ' + str(hof[0].fitness.values[0]) + '\n')
        return logbook
if __name__ == "__main__":
        main()
        if True:
            g_simulator.saveDataBool = True
            individual = [0.6, 0.9, 0.7, 0.9, 0.85, 0.65]
            inputIndividul = []
            for i in range(6):
                if individual[i] > 0.99:
                    inputIndividul.append(0.99)
                elif individual[i] < 0.01:
                    inputIndividul.append(0.01)
                else:
                    inputIndividul.append(individual[i])
            tankInitOilLst = []
            for i in range(6):
                tankInitOilLst.append(weightFuncLst[i] * inputIndividul[i])
            print('初始油量为 = ',tankInitOilLst)
            g_simulator.run(tankInitOilLst)
```

9.4 问题四代码

```
from simulator4Pro import Point3D, AirPlaneSimulator
import xlrd
worksheet = xlrd.open_workbook(u'附件 5-问题四数据.xlsx')
sheet_names = worksheet.sheet_names()
# print(sheet_names)
sheet = worksheet.sheet_by_name('发动机耗油数据')
oilCostLst = sheet.col_values(1)
oilCostLst.remove('耗油速度(kg/s)')
import math
```

```
sheet = worksheet.sheet_by_name('飞行器俯仰角')
angleLst = sheet.col_values(1)
angleLst.remove('俯仰角(度)')
angleLst = [(_/360)*math.pi *2 for _ in angleLst ]
print(angleLst)
idealCentroidLst = []
for i in range(7200):
	unitPoint = Point3D(0,0,0)
	idealCentroidLst.append(unitPoint)
# print(idealCentroidLst)
# exit()
g_air_plane_simulator = AirPlaneSimulator(angleLst, idealCentroidLst, oilCostLst, [])
import operator
import math
import random
import numpy
from deap import algorithms
from deap import base
from deap import creator
from deap import tools
from deap import gp
# Define new functions
def protectedDiv(left, right):
	try:
		return left / right
	except ZeroDivisionError:
		return 1
def max(a, b):
	if a>=b:
		return a
	else:
		return b
def min(a, b):
	if a<=b:
		return a
	else:
		return b
pset = gp.PrimitiveSet("MAIN", 13)
pset.addPrimitive(operator.add, 2)
pset.addPrimitive(operator.sub, 2)
pset.addPrimitive(operator.mul, 2)
pset.addPrimitive(protectedDiv, 2)
pset.addPrimitive(operator.neg, 1)
# pset.addPrimitive(math.cos, 1)
```

```
# pset.addPrimitive(math.sin, 1)
pset.addPrimitive(min, 2,name="Min")
pset.addPrimitive(max, 2,name="Max")
# pset.renameArguments(ARG0='x')
creator.create("FitnessMin", base.Fitness, weights=(-1.0,))
creator.create("Individual", gp.PrimitiveTree, fitness=creator.FitnessMin)
toolbox = base.Toolbox()
toolbox.register("expr", gp.genHalfAndHalf, pset=pset, min_=1, max_=3)
toolbox.register("individual", tools.initIterate, creator.Individual, toolbox.expr)
toolbox.register("population", tools.initRepeat, list, toolbox.individual)
toolbox.register("compile", gp.compile, pset=pset)
def evalSymbReg(individual):
    # Transform the tree expression in a callable function
    func = toolbox.compile(expr=individual)
    fitness = g_air_plane_simulator.run(func)
    return fitness,
toolbox.register("evaluate", evalSymbReg)
toolbox.register("select", tools.selTournament, tournsize=7)
toolbox.register("mate", gp.cxOnePoint)
toolbox.register("expr_mut", gp.genFull, min_=0, max_=2)
toolbox.register("mutate", gp.mutUniform, expr=toolbox.expr_mut, pset=pset)
toolbox.decorate("mate", gp.staticLimit(key=operator.attrgetter("height"), max_value=5))
toolbox.decorate("mutate", gp.staticLimit(key=operator.attrgetter("height"), max_value=5))
import numpy as np
def main():
    randomSeed = 1
    random.seed(randomSeed)
    np.random.seed(randomSeed)
    ind = toolbox.individual()

    pop = toolbox.population(n = 10)
    hof = tools.HallOfFame(1)
    stats = tools.Statistics(lambda ind: ind.fitness.values)
    stats.register("avg", numpy.mean)
    stats.register("std", numpy.std)
    stats.register("min", numpy.min)
    stats.register("max", numpy.max)
    logbook = tools.Logbook()
    logbook.header = "gen", "evals", "std", "min", "avg", "max"
    f_con = open('pro4rgp_'+ str(randomSeed)+'.txt','w')
    CXPB, MUTPB, NGEN = 0.5, 0.2, 40
    cxpb = 0.8
    mutpb = 0.15
    # Evaluate the entire population
```

```
for ind in pop:
        ind.fitness.values = toolbox.evaluate(ind)
hof.update(pop)
record = stats.compile(pop)
logbook.record(gen=0, evals=len(pop), **record)
print(logbook.stream)
for g in range(1, NGEN):
        # Select the offspring
        offspring = toolbox.select(pop, len(pop))
        # Clone the offspring
        offspring = [toolbox.clone(ind) for ind in offspring]
        # Apply crossover and mutation
        for i in range(1, len(offspring), 2):
                if random.random() < cxpb:
                        offspring[i - 1], offspring[i] = toolbox.mate(offspring[i - 1], offspring[i])
                        del offspring[i - 1].fitness.values, offspring[i].fitness.values
        for i in range(len(offspring)):
                if random.random() < mutpb:
                        offspring[i], = toolbox.mutate(offspring[i])
                        del offspring[i].fitness.values
        # Evaluate the individuals with an invalid fitness
        invalids = [ind for ind in offspring if not ind.fitness.valid]
        for ind in invalids:
                ind.fitness.values = toolbox.evaluate(ind)
        # Replacement of the population by the offspring
        pop = offspring
        hof.update(pop)
        record = stats.compile(pop)
        logbook.record(gen=g, evals=len(invalids), **record)
        print(logbook.stream)
        f_con.write('gen '+ str(g) + ' ')
        f_con.write(' '+ str(record['min']) + '\n')
        f_con.write(str(hof[0]) + '\n')
        f_con.write('min    ' + str(hof[0].fitness.values[0]) + '\n')
        f_con.flush()
        if g%7 == 1:
                func = toolbox.compile(expr=hof[0])
                g_air_plane_simulator.saveDataBool = True
                fitness = g_air_plane_simulator.run(func)
                g_air_plane_simulator.saveDataBool = False
print('Best individual : ', hof[0][0], hof[0].fitness)
f_con.write(str(hof[0]) + '\n')
f_con.write('min    ' + str(hof[0].fitness.values[0]) + '\n')
return pop, stats, hof
```

```
        # print log
        return pop, log, hof
if __name__ == "__main__":
        main()
```