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	杭州电子科技大学
参赛队号	**20103360037**
队员姓名	**1.**邓超禹
	2.周红
	3.李勇杰

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：

本文针对飞行器质心平衡供油策略优化问题，建立了三种飞行器动态供油质心模型，以飞行器质心偏移最小为优化目标，得到了在不同条件下飞行器油箱的供油策略。

针对问题一，本文针对飞行器质心求解问题，建立了**飞行器-燃油组合体质心模型**求解飞行器质心。不同时刻俯仰角不同会导致飞行器中油箱内燃油的形状出现五种情况，依据飞行器俯仰角大于 0 和小于 0 的对称性和五种情况的**几何特征**计算燃油的质心，通过**多质点质心公式**得到飞行器的质点坐标。对于问题一提供的数据，通过飞行器-燃油组合体质心模型算法得到了 0-7200s 内**平滑**的飞行器质心变化曲线，该质心变化曲线在 XOY 平面上的**投影为“心”形。算法的实际运行时间为 3.89s。**

针对问题二，基于飞行器在飞行过程中油箱供油速度的约束，总供油量的约束，油箱选择的约束，油箱供油时长的约束，以及结合问题一中飞行器中行器-燃油组合体质心模型建立了**飞行器水平飞行-质心模型**。使用**理想质心偏移补偿法**，此方法首先确定后一时刻与当前时刻**理想质心的偏移向量**，然后采用**“01”编码**确定油箱供油的组合方式，遍历符合约束条件的供油方案，用**供油方案产生的质心偏移向量对理想质心的偏移向量进行补偿，**求解使得两个偏移向量的差值达到最小时对应的供油方案。对于问题二提供的数据，通过理想质心偏移补偿法得到 0-7200s 内的瞬时质心坐标与理想质心坐标的最大距离为**0.068697722m，**瞬时质心坐标轨迹与理想质心坐标轨迹基本**吻合**，四个主油箱的总供油量为 **6441.524212kg，主**油箱 2、3、4、5 的供油速度曲线比较**连续平滑**。**算法的实际运行时间为 464.3s。**

针对问题三，相比问题二未给出油箱中燃油的初始体积，并且增加了在飞行结束时刻 6 个油箱剩余的总油量大于 $1m^3$ 的约束条件，结合问题二建立了**飞行器水平飞行-质心模型**。使用**粒子群算法**，以油箱中燃油的初始体积为粒子坐标，飞行器在飞行时飞行器瞬时质心与理想质心距离的最大值为适应值。其中适应值使用**反向理想质心偏移补偿法**进行求解，此方法根据油箱中剩余燃油体积，从最后一个时刻往前确定每一时刻的供油方案，并得到飞行器瞬时质心与理想质心距离的最大值。对于问题三提供的数据，通过**粒子群算法**与**反向理想质心偏移补偿法**得到 6 个油箱的初始油量**分别为：268.5kg、1025.49kg、1315.8kg、2247.43269kg、1778.151979kg、1019.8kg，**得到 0-7200s 内的瞬时质心坐标与理想质心坐

标的最大距离为 **0.120308m**，四个主油箱的总供油量为 **6805.174669kg**，**主**油箱 2、3、4、5 的供油速度曲线比较**连续平滑**。**算法的实际运行时间为** 758.6s**。**

针对问题四，相较于问题二的水平飞行，飞行器可以前后俯仰，建立了**飞行器俯仰飞行-质心模型。**使用**理想质心偏移补偿法**，此方法首先确定后一时刻与当前时刻**理想质心的偏移向量**，然后采用**“01”编码**确定油箱供油的组合方式，基于当前时刻**俯仰角**遍历符合约束条件的供油方案，用**供油方案产生的质心偏移向量对理想质心的偏移向量进行补偿，**求解使得两个偏移向量的差值达到最小时对应的供油方案。对于问题二提供的数据，通过理想质心偏移补偿法得到 0-7200s 内的瞬时质心坐标与坐标原点的最大距离为 **0.14345552m**，瞬时质心坐标轨迹**在原点附近小幅度波动**，四个主油箱的总供油量为 **7035.545kg**，主油箱 2、3、4、5 的供油速度曲线比较**连续平滑**。**算法的实际运行时间为 390.2s。**

关键词：多质点质心公式；理想质心偏移补偿法；粒子群算法；反向理想质心偏移补偿法

目录

一、 问题重述

1.1 问题背景

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，这里，油箱的供油策略可用其向发动机或其它油箱供油的速度曲线来描述。

1.2 问题提出

附件 1 给出了飞行器的相关参数，附件 2-附件 5 给出了该类飞行器在执行某任务过程中飞行和控制的相关数据，请你们团队根据任务要求，建立数学模型，设计算法，并分析算法的有效性和复杂度，完成以下问题：

问题一. 附件 2 给出了某次任务中飞行器的 6 个油箱的供油速度及飞行器在飞行过程中的俯仰角变化数据，每秒记录一组数据（下同）。请给出该飞行器在此次任务执行过程中的质心变化曲线，并将其质心在飞行器坐标系下的位置数据按时间（每秒一组）先后顺序存入附件 6 结果表“第一问结果”中。

问题二. 附件 3 给出了某次任务的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。根据任务需求，在飞行器始终保持平飞(俯仰角为 0)的任务规划过程中，请为飞行器制定该次任务满足条件(1)～(6)的 6 个油箱供油策略，使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小：

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

请给出飞行器飞行过程中 6 个油箱各自的供油速度曲线和 4 个主油箱的总供油速度曲线(时间间隔为 1s)、以及飞行器瞬时质心与理想质心距离的最大值和 4 个主油箱的总供油量，并将 6 个油箱的供油速度数据按时间（每秒一组）先后顺序存入附件 6 结果表“第二问结果”中。

问题三. 假定初始油量未定，飞行器其他相关参数如附件 1 所示，附件 4 给出了某次任务的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。在飞行器始终保持平飞(俯仰角为 0)的任务规划过程中，请为飞行器制定该次任务满足条件(1)～(6)的 6 个油箱初始载油量及供油策略，使得本次任务结束时 6 个油箱剩余燃油总量至少 1m3，并且飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小：

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

请给出 6 个油箱的初始载油量、飞行器飞行过程中 6 个油箱的供油速度曲线和 4 个主油箱的总供油速度曲线(时间间隔为 1s)、以及飞行器质心与理想质心距离的最大值和 4 个主油箱的总供油量。请将 6 个油箱的初始油量存入附件 6 结果表“第三问结果”

中的提示位置，并将 6 个油箱的供油速度数据按时间（每秒一组）先后顺序存入附件 6 结果表“第三问结果”中。

问题四. 在实际任务规划过程中，飞行器俯仰角随时间变化。附件 5 给出了飞行器俯仰角的变化数据和耗油速度数据。请为本次任务制定油箱供油策略，使得飞行器瞬时质心$\vec{c}_1(t)$与飞行器(不载油)质心$\vec{c}_0$的最大距离达到最小，即：

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

请绘出飞行器飞行过程中 6 个油箱各自的供油速度曲线，再将 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绘于一个图中，给出飞行器瞬时质心与飞行器(不载油)质心$\vec{c}_0$的最大距离偏差以及 4 个主油箱的总供油量，并将 6 个油箱的供油速度数据按时间（每秒一组）先后顺序存入附件 6 结果表“第四问结果”中。

二、 问题分析

2.1 问题一

问题一是要建立模型计算飞行器在每一时刻的质心坐标。

由于飞行器在俯仰飞行过程中俯仰角度会发生改变，油箱内燃油液体的形状会出现不同的情况，然后讨论油箱内液体在不同情况下的质心求解，最后分别得到每个油箱的质心坐标。

最后将飞行器分成 7 个部分，不装油的飞行器和 6 个油箱中的燃油，由于飞行器在飞行过程中形状不发生改变，所以在求飞行器质心时只用考虑 6 个油箱中的燃油对质心的影响即可。把 6 个油箱中的燃油看作质点，根据前面得到的每个油箱的质心坐标，最后通过空间中多质点的质心求解公式得到飞行器的质点坐标。

2.2 问题二

问题二需要在约束条件下建立模型求解，制定策略确定的每个飞行器飞行过程中每一时刻 6 个油箱各自的供油速度以及 4 个主油箱的总供油速度，使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小。

可以根据两个时刻之间理想质心位置的偏移向量，调整燃油的供给，使得燃油体积变化引起的质心位置偏移向量与想质心位置的偏移向量的差值达到最小。

通过问题一的方法求解两个时刻的飞行器的质点坐标，将两个坐标相减就可以得到燃油体积变化引起的质心位置偏移向量。

这样在每一时刻得到的供油方案都能使飞行器质心位置与理想质心位置的欧氏距离达到最小。就能使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小。

2.3 问题三

问题三需要在约束条件下建立模型求解，制定策略确定的每个飞行器飞行过程中每一时刻 6 个油箱各自的供油速度以及 4 个主油箱的总供油速度，使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小。相比问题二，在问题三中没有给出飞行器油箱中的初始油量，以及新增了在飞行结束时飞行器中的总油量大于 1 的约束条件。

对于问题三，由于不知道初始油量，但是知道最后一时刻每个油箱中的燃油体积。可以从最后一时刻往前确定供油策略。供油策略的求法同问题二根据两个时刻之间理想质心位置的偏移向量，调整燃油的供给使得燃油体积变化引起的质心位置偏移向量与想质心位置的偏移向量的差值达到最小。就可以得到飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小时的值。

这样在最后一时刻每个油箱中的燃油体积确定以后，就可以得到飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小时的值。

于是问题转化为求解最后一时刻每个油箱中的燃油体积分配方案使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小。

2.3 问题四

问题二需要在约束条件下建立模型求解，制定策略确定的每个飞行器飞行过程中每一时刻 6 个油箱各自的供油速度以及 4 个主油箱的总供油速度，使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离的最大值达到最小。相比问题二，问题四中飞行器不再只是水平飞行，还有俯仰飞行。

由问题一分析可知，飞行器在俯仰飞行时油箱中燃油形状会出现不同情况，所以需要在不同时刻根据不同的情况计算飞行器的质心坐标。求解过程同问题二一样，先计算在每个时刻各个油箱的质心坐标，再通过计算再每一时刻飞行器的质心坐标。最后通过空间中多质点的质心求解公式得到飞行器的质点坐标

三、 模型假设

1、假设飞行器油箱燃油密度均匀。

2、假设燃油在飞行过程中不会晃动。

3、假设飞行器在飞行过程中只考虑上下俯仰，不考率左右偏转。

4、假设油箱向发动机供油的过程中，供油和耗油相等，不产生多余的燃油。

5、假设油箱在供油速度为 0 时，可以添加一个极小的供油量，保持油箱持续供油。

四、 符号说明

变量名	符号说明	单位
α	飞行器俯仰角度	°
L	油箱在X方向的长度	m
H	油箱在Z方向的高度	m
W	油箱在Y方向的高度	m
V_m	油箱的体积	m^3
V	油箱中燃油的体积	m^3
x_c, y_c, z_c	油箱中燃油的质心坐标	
c_{1x}, c_{1y}, c_{1z}	飞行器质心坐标	
$O_i(t)$	油箱i在t时刻的供油速度	m^3
$v_i(t)$	油箱i在t时刻的燃油体积	m^3
$m_i(t)$	油箱i在t时刻的燃油质量	kg
ρ	燃油的密度	kg/m^3
$d_{\min}$	实际质心与理想质心欧式距离最大值	m
α	飞行器俯仰角度	°
L	油箱在X方向的长度	m
H	油箱在Z方向的高度	m
W	油箱在Y方向的高度	m
V_m	油箱的体积	m^3

五、 问题一模型建立和求解

5.1 飞行器-燃油组合体质心模型建立

5.1.1 燃油形状的分类和判断

如图 1 所示，建立油箱坐标系 O(t)-X(t)Y(t)Z(t)：以油箱右下角顶点为原点 O(t)，飞行器纵向中心轴为 X(t)轴，以飞行器前方为正向，Y(t)轴垂直于 X(t)轴所在的飞行器纵剖面，且 O(t)- X(t)Y(t)组成右手坐标系，通过右手法则确定 Z(t)轴。

飞机俯仰飞行时（$\alpha \neq 0$）：

油箱中燃油的形状取决于飞行器的俯仰角a，液体体积V，以及油箱的尺寸。

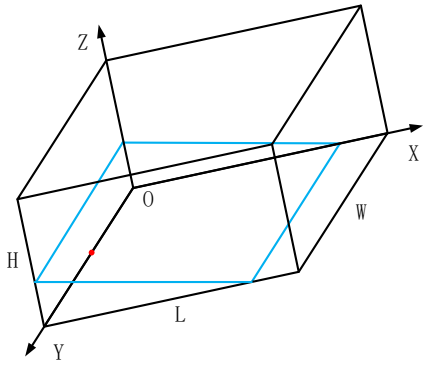


图 1　飞行器俯仰时油箱燃油形状示意图

与油箱尺寸相关的角度α_0由下式确定：

$$\alpha_0=\text{arc}tg(\text{H}/\text{L}) \tag{5-1}$$

式中α_0为与油箱尺寸有关的角度，L为油箱在X方向的长度，H为油箱在Z方向的高度，如图 2 所示。

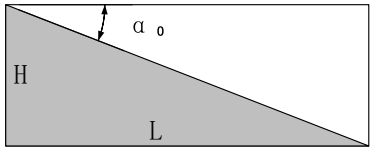


图 2 油箱尺寸相关的角度α_0

当飞机俯仰的时候（$\alpha \neq 0$）时油箱中液体的形状取决于飞行器俯仰角α，燃油体积V以及油箱的尺寸，但最多只能出现如图 3、图 4、图 5 和图 6 所示的四种形状，因为飞行器姿态的改变仅考俯仰情况，不考虑左右偏转，所以图 3、图 4、图 5 和图 6 是沿油箱横向对称面的剖面示意图。

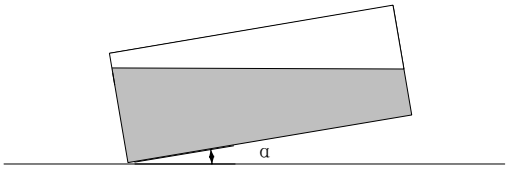


图 3 第一种形状

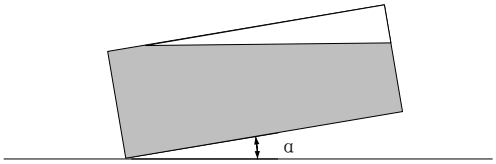


图 4 第二种形状

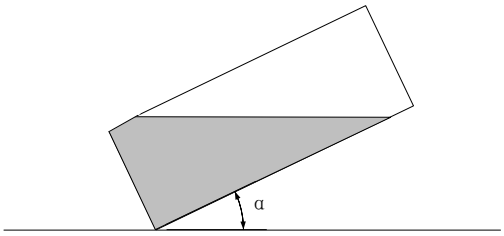


图 5 第三种形状

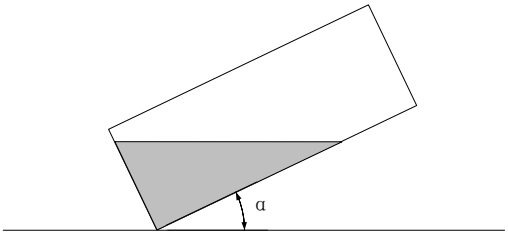


图 6 第四种形状

油箱中燃油的形状不同，其质心位置的计算公式也不相同，因此有必要先对液体形状的分类及其判别方法作如下讨论。

图 7 是当$\alpha < \alpha_0$时，油箱中燃油形状的分类界限示意图。设V_m是油箱的总体积，V为实际燃油体积，V_1是图 7a 中燃油的体积，此时油面刚好通过 B 点。V_2是图 7b 中燃油的体积，此时油面刚好通过 A 点。

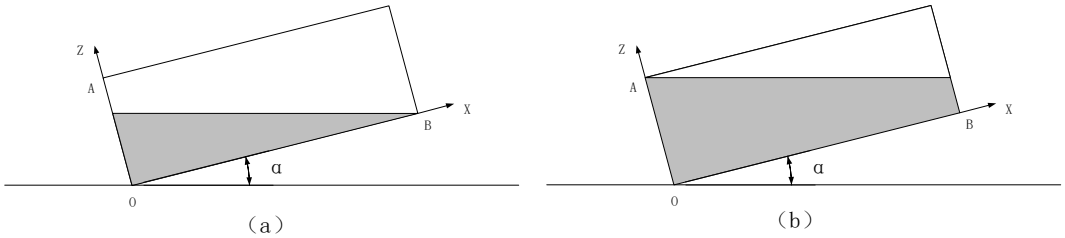


图 7 $\alpha < \alpha_0$时液面形状分类界限示意图

油箱总体积V_m：

$$V_m = LWH \tag{5-2}$$

其中H为油箱在Y轴方向的高。

在图 7a 中，根据几何图形特征计算燃油体积V_1：

$$V_1 = \frac{1}{2}\tan(\alpha)WL^2 \tag{5-3}$$

在图 7b 中，由对称性可计算燃油体积V_2：

$$V_2 = V_m - V_1 \tag{5-4}$$

由图 7 可得如下结论：

a) 当$\alpha < \alpha_0$，$0 \le V < V_1$时，油箱中燃油形状一定是第四种情况；

b) 当$\alpha<\alpha_0$，$V_1\le V<V_2$时，油箱中燃油形状一定是第一种情况；

c) 当$\alpha<\alpha_0$，$V_2\le V<V_{\mathrm{m}}$时，油箱中燃油形状一定是第二种情况。

图 8 是当$\alpha\ge\alpha_0$时，油箱中燃油形状的分类界限示意图。设V_m是油箱的总体积，V为实际燃油体积，V_3是图 8a 中燃油的体积，此时油面刚好通过 A 点。V_4是图 8b 中燃油的体积，此时油面刚好通过 B 点。

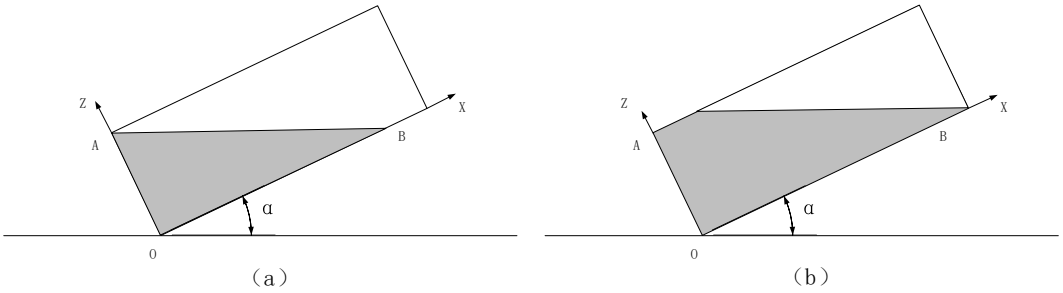


图 8 $\alpha\ge\alpha_0$时液面形状分类界限示意图

在图 8a 中，根据几何图形计算燃油体积V_3：

$$\mathrm{V_3}=\frac{1}{2\tan(\alpha)}W\mathrm{H}^2 \tag{5-5}$$

在图 8b 中，由对称性可计算燃油体积V_4：

$$\mathrm{V_4}=\mathrm{V}_m-\mathrm{V_3} \tag{5-6}$$

由图 8 可得如下结论：

a) 当$\alpha\ge\alpha_0$，$0\le V<V_3$时，油箱中燃油形状一定是第四种情况；

b) 当$\alpha\ge\alpha_0$，$V_3\le V<V_4$时，油箱中燃油形状一定是第三种情况；

c) 当$\alpha\ge\alpha_0$，$V_4\le V<V_{\mathrm{m}}$时，油箱中燃油形状一定是第二种情况。

飞机水平飞行时（$\alpha=0$）：

当飞机水平飞行时，油箱中液体的形状由图 9 所示，此时油箱中的液体的形状只取决于燃油体积V以及油箱的尺寸。

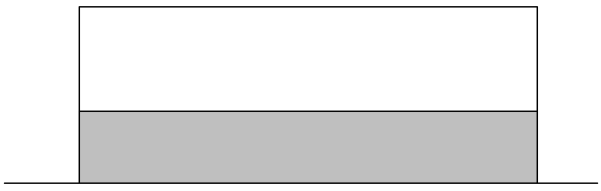


图 9 第五种形状

根据以上讨论可知，已知油箱（飞行器）的俯仰角α，燃油的体积V以及油箱的尺

寸可以判断出燃油在油箱中的形状属于五种情况中的哪一种。

5.1.2 燃油质心位置坐标计算

在计算燃油质心位置坐标时没有特殊说明时，坐标均是在油箱坐标系下得到。

当已经判断出了油箱中液体的形状后，就可以根据不同的形状，采用不同的策略来计算燃油的质心位置坐标。假设油箱中装的燃油是一种匀质液体，装燃油的油箱为长方体，燃油的形状可以看作是宽度 Y 方向的棱柱，五种燃油形状图 3、图 4、图 5、图 6 中分别为四棱柱、五棱柱、四棱柱、三棱柱，图 9 为长方体。对应底面图形分别为四边形、五边形、四边形、三角形，长方形。已知燃油体积V和油箱宽度W，则油箱燃油剖面多边形的面积A为：

$$A = \frac{V}{W} \tag{5-7}$$

在下 XOZ 二维坐标平面中，多边形的形心坐标（$\mathrm{x_c}, \mathrm{z_c}$）即为燃油质心的油箱坐标系对应$x$坐标和$z$坐标。

三角形的质心坐标求解：三角形的质心同时也是三角形的形心，若三角形在 XOY 坐标系下顶点按逆时针方向排列的坐标分别为$(x_1, z_1), (x_2, z_2), (x_3, z_3)$，则三角形质心的坐标$(x_c, z_c)$为：

$$\begin{cases} x_c = \dfrac{x_1 + x_2 + x_3}{3} \\ z_c = \dfrac{z_1 + z_2 + z_3}{3} \end{cases} \tag{5-8}$$

同时可以得到三角形的面积为A：

$$A = \frac{x_1 z_2 + x_2 z_3 + x_3 z_1 - y_1 z_2 - y_2 z_3}{2} \tag{5-9}$$

对于 n 边形，可以拆分为 n-2 个三角形，然后利用组合图形的质心坐标公式求多边形的质心(x_c, z_c)：

$$\begin{cases} x_c = \dfrac{\sum_{i=1}^{n-2} A_i x_{ci}}{A} \\ z_c = \dfrac{\sum_{i=1}^{n-2} A_i z_{ci}}{A} \end{cases} \tag{5-10}$$

其中A_i为多边形拆分第i个三角形的面积，x_{ci}和z_{ci}为多边形拆分的第i个三角形的质心坐标， A 为多边形的面积：

$$A=\sum_{i=1}^{n-2}A_i \tag{5-11}$$

设 n 边形逆时针方向的坐标为 $(x_1,z_1),(x_2,z_2),(x_3,z_3),\cdots,(x_n,z_n)$。则第 i 个三角形的面积 A_i 为：

$$A_i=\frac{x_1y_{i+1}+x_{i+1}y_{i+2}+x_{i+2}y_1-y_1x_{i+1}-y_{i+1}x_{i+2}}{2} \tag{5-12}$$

第 i 个三角形的质心 x_{ci}，z_{ci} 为：

$$\begin{cases} x_{ci}=\dfrac{x_1+x_{i+1}+x_{i+2}}{3} \\ z_{ci}=\dfrac{z_1+z_{i+1}+z_{i+2}}{3} \end{cases} \tag{5-13}$$

代入得 n 边形的形心（x_c,z_c）为：

$$\begin{cases} x_c=\dfrac{\sum_{i=1}^{n-2}\frac{(x_1z_{i+1}+x_{i+1}z_{i+2}+x_{i+2}z_1-y_1z_{i+1}-y_{i+1}z_{i+2})\cdot(x_1+x_{i+1}+x_{i+2})}{6}}{\sum_{i=1}^{n-2}A_i} \\ z_c=\dfrac{\sum_{i=1}^{n-2}\frac{(x_1z_{i-1}+x_{i-1}z_i+x_iz_1-z_1x_{i-1}-z_{i-1}x_i)\cdot(z_1+z_{i+1}+z_{i+2})}{6}}{\sum_{i=1}^{n-2}A_i} \end{cases} \tag{5-14}$$

由棱柱的对称性质可以得到：

$$y_c=\frac{W}{2} \tag{5-15}$$

至此在得到多边形的逆时针排序的坐标时就可以得到多边形的质心坐标。于是在得到油箱内的燃油液面与油箱相交点的坐标以后就可以求出油箱内燃油的质心位置。

下面分类讨论在不同邮箱内燃油形状下，油箱内燃油质心的求解。

第一种形状下质心坐标求解

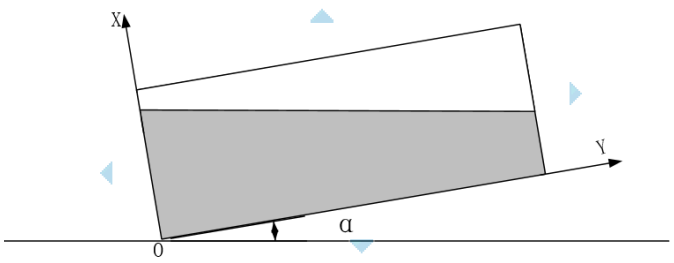


图 10 第一种形状下油箱内的燃油液面与油箱交点坐标示意图

从图可知在第一种形状下油箱内的燃油液面与油箱相交点有四个。以原点为起始点逆时针方向依次计算交点坐标。

容易得到前两个点的坐标为：$(x_1,z_1)=(0,0)$，$(x_2,z_2)=(L,0)$。

由几何关系可以得：

$$\begin{aligned} z_3 &= \frac{\left(A-\frac{1}{2}\cdot L\right)}{L} \\ z_4 &= L\cdot\tan\alpha+z_3=L\cdot\tan\alpha+\frac{\left(A-\frac{1}{2}\cdot L\right)}{L} \end{aligned} \tag{5-16}$$

则可以得到后面两个点的坐标为：$(x_3,z_3)=\left(L,\dfrac{\left(A-\frac{1}{2}\cdot L\right)}{L}\right)$，

$(x_4,z_4)=\left(0,L\cdot\tan\alpha+\dfrac{\left(A-\frac{1}{2}\cdot L\right)}{L}\right)$。将坐标带入式 5-14 以及 5-15 就可以得到油箱内燃油为第一种形状时的质心(x_c,y_c,z_c)：

$$\begin{cases} x_c=\dfrac{\sum_{i=1}^{2}\frac{\left(x_1z_{i+1}+x_{i+1}z_{i+2}+x_{i+2}z_1-y_1z_{i+1}-y_{i+1}z_{i+2}\right)\cdot\left(x_1+x_{i+1}+x_{i+2}\right)}{6}}{\sum_{i=1}^{2}A_i} \\ y_c=\dfrac{W}{2} \\ z_c=\dfrac{\sum_{i=1}^{2}\frac{\left(x_1z_{i-1}+x_{i-1}z_i+x_iz_1-z_1x_{i-1}-z_{i-1}x_i\right)\cdot\left(z_1+z_{i+1}+z_{i+2}\right)}{6}}{\sum_{i=1}^{2}A_i} \end{cases} \tag{5-17}$$

第二种形状下质心坐标求解

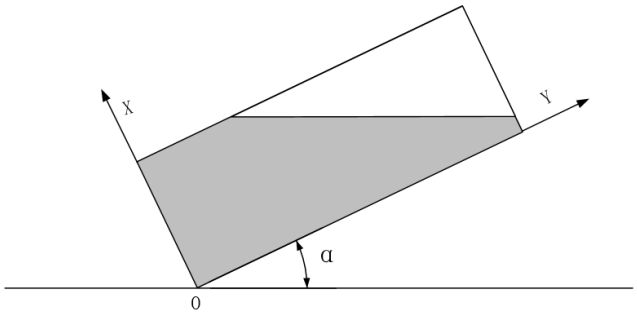


图 11 第二种形状下油箱内的燃油液面与油箱交点坐标示意图

从图可知在第二种形状下油箱内的燃油液面与油箱相交点有五个。以原点为起始点逆时针方向依次计算交点坐标。

容易得到前两个点的坐标为：$(x_1,z_1)=(0,0)$，$(x_2,z_2)=(L,0)$。

由几何关系可以得：

$$\begin{aligned} z_3 &= W-\sqrt{2\cdot(L\times H-A)\cdot\tan\alpha} \\ x_4 &= L-\sqrt{\frac{2\cdot(L\times H-A)}{\tan\alpha}} \end{aligned} \tag{5-18}$$

则可以得到接下来两个点的坐标为：$(x_3,z_3)=\left(L,W-\sqrt{2\cdot(L\times H-A)\cdot\tan\alpha}\right)$，$(x_4,z_4)=\left(L-\sqrt{\frac{2\cdot(L\times H-A)}{\tan\alpha}},H\right)$。

容易得到最后一个点的坐标为：$(x_5,z_5)=(0,H)$。

将坐标带入式 5-14 以及 5-15 就可以得到油箱内燃油为第二种形状时的质心(x_c,y_c,z_c)：

$$\begin{cases} x_c=\dfrac{\sum_{i=1}^{3}\frac{(x_1z_{i+1}+x_{i+1}z_{i+2}+x_{i+2}z_1-y_1z_{i+1}-y_{i+1}z_{i+2})\cdot(x_1+x_{i+1}+x_{i+2})}{6}}{\sum_{i=1}^{3}A_i} \\ y_c=\dfrac{W}{2} \\ z_c=\dfrac{\sum_{i=1}^{3}\frac{(x_1z_{i-1}+x_{i-1}z_i+x_iz_1-z_1x_{i-1}-z_{i-1}x_i)\cdot(z_1+z_{i+1}+z_{i+2})}{6}}{\sum_{i=1}^{3}A_i} \end{cases} \tag{5-19}$$

第三种形状下质心坐标求解

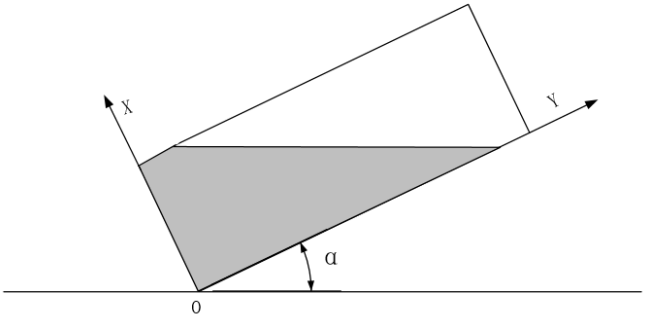


图 12 第三种形状下油箱内的燃油液面与油箱交点坐标示意图

从图可知在第三种形状下油箱内的燃油液面与油箱相交点有四个。以原点为起始点逆时针方向依次计算交点坐标。

容易得到第一个点的和最后一个点的坐标为：$(x_1,z_1)=(0,0)$，$(x_4,z_4)=(0,H)$。

由几何关系可以得：

$$\begin{aligned} x_2 &= \frac{A\cdot\tan\alpha}{H}-\frac{H}{2}+\frac{H}{\tan\alpha} \\ x_3 &= \frac{A\cdot\tan\alpha}{H}-\frac{H}{2} \end{aligned} \tag{5-20}$$

则可以得到第二、三个点的坐标为：$(x_2,z_2)=\left(\frac{A\cdot\tan\alpha}{H}-\frac{H}{2}+\frac{H}{\tan\alpha},0\right)$，$(x_3,z_3)=\left(\frac{A\cdot\tan\alpha}{H}-\frac{H}{2},H\right)$。将坐标带入式 5-14 以及 5-15 就可以得到油箱内燃油为第三种形状时的质心(x_c,y_c,z_c)：

$$\begin{cases} x_c = \dfrac{\sum_{i=1}^{2}\dfrac{(x_1z_{i+1}+x_{i+1}z_{i+2}+x_{i+2}z_1-y_1z_{i+1}-y_{i+1}z_{i+2})\cdot(x_1+x_{i+1}+x_{i+2})}{6}}{\sum_{i=1}^{2}A_i} \\ y_c = \dfrac{W}{2} \\ z_c = \dfrac{\sum_{i=1}^{2}\dfrac{(x_1z_{i-1}+x_{i-1}z_i+x_iz_1-z_1x_{i-1}-z_{i-1}x_i)\cdot(z_1+z_{i+1}+z_{i+2})}{6}}{\sum_{i=1}^{2}A_i} \end{cases} \tag{5-21}$$

第四种形状下质心坐标求解

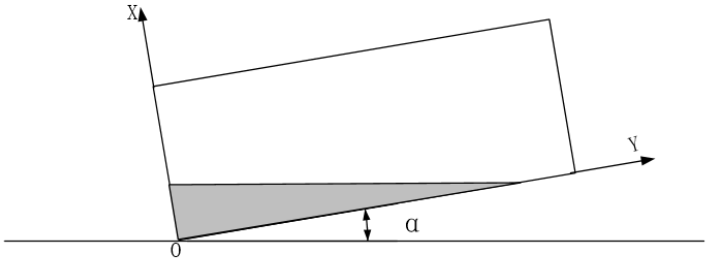


图 13 第四种形状下油箱内的燃油液面与油箱交点坐标示意图

从图可知在第三种形状下油箱内的燃油液面与油箱相交点有三个。以原点为起始点逆时针方向依次计算交点坐标。

容易得到第一个点的和最后一个点的坐标为：$(x_1,z_1)=(0,0)$，$(x_4,z_4)=(0,H)$。

由几何关系可以得：

$$\begin{aligned} x_2 &= \sqrt{2A\tan\alpha} \\ z_3 &= \sqrt{2A^2\tan\alpha}\cdot\tan\alpha \end{aligned} \tag{5-22}$$

则可以得到第二、三个点的坐标为：$(x_2,z_2)=\left(\sqrt{2A\tan\alpha},0\right)$，$(x_3,z_3)=\left(0,\sqrt{2A\tan\alpha}\cdot\tan\alpha\right)$。由三角形的质心公式 5-8，以及式 5-16 可以得到油箱内燃油为第四种形状时的质心(x_c,y_c,z_c)：

$$\begin{cases} x_c=\dfrac{x_1+x_2+x_3}{3} \\ y_c=\dfrac{W}{2} \\ z_c=\dfrac{z_1+z_2+z_3}{3} \end{cases} \tag{5-23}$$

第五种形状下质心坐标求解

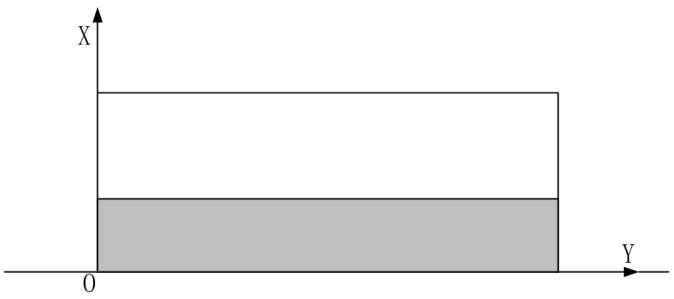


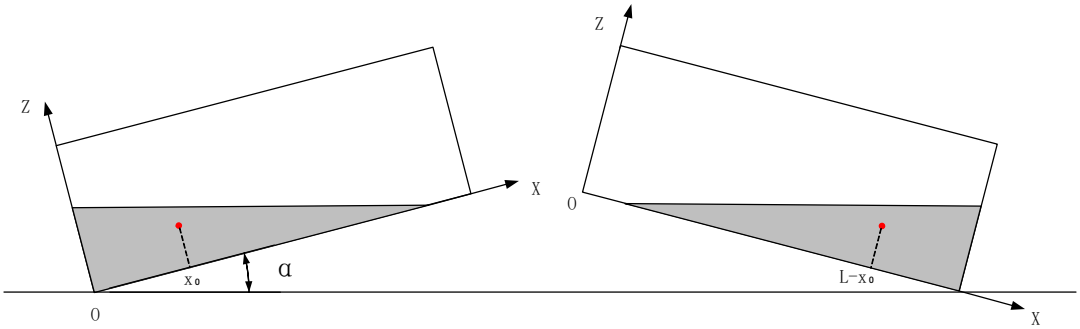
图 14 第五种形状下油箱内的燃油液面与油箱交点坐标示意图

当油箱内燃油为第五种形状时，由几何关系可以得到此时的质心(x_c, y_c, z_c)为：

$$\begin{cases} x_c = \dfrac{A}{2L} \\ y_c = \dfrac{W}{2} \\ z_c = \dfrac{L}{2} \end{cases} \tag{5-24}$$

当俯仰角小于 0 时（$\alpha < 0$）质心坐标求解

以上讨论均为飞行器俯仰角大于等于 0（飞行器头部朝上）的情况，对于俯仰角小于 0（飞行器头部朝下）的情况需要另作讨论，如图 15 所示：



（俯仰角大于0）　　（俯仰角小于0）

图 15 俯仰角情况

由图 15，可以将俯仰角小于 0 的情况先转换为俯仰角大于 0 的情况，令：

$$\alpha = -\alpha, \alpha < 0 \tag{5-25}$$

再按照仰角大于 0 的情况进行处理。得到对应油箱内燃油的质心(x_c, y_c, z_c)。

由对称性可以得到俯仰角小于 0 时油箱燃油质心位置坐标(x_c^-, y_c^-, z_c^-)：

$$\begin{cases} x_c^- = \mathrm{L} - x_c \\ y_c^- = y_c \\ z_c^- = z_c \end{cases}$$

5.1.3 飞行器质心位置坐标的计算

无特殊说明飞行器质心位置坐标均是在飞行器坐标下表示。

对于带有 6 个油箱的飞行器，其每个油箱中的燃油都可以通过上述燃油质心位置坐标的计算方法计算各自油箱坐标系内的质心位置坐标$(\mathrm{x}_{cj}, \mathrm{y}_{cj}, \mathrm{z}_{cj})$，其中$j = 1,2,\ldots,6$。每个油箱的燃油体积$v_j$已知，可以计算对应的燃油质量为：

$$m_j = v_j \cdot \rho \tag{5-26}$$

将 6 个油箱中燃油质心在各自油箱坐标系下的位置坐标转换到飞行器坐标系下，第 j 个空油箱在飞行器坐标系下中心位置为 $\vec{p}_j = (x_{0j}, y_{0j}, z_{0j})$，得到油箱 j 中的燃油质心在飞行器坐标系下的坐标 $(\mathrm{x}_j, \mathrm{y}_j, \mathrm{z}_j)$：

$$\begin{cases} x_j = x_{0j} - \dfrac{\mathrm{L}_j}{2} + x_{cj} \\ y_j = \mathrm{y}_{0j} \\ z_j = z_{0j} - \dfrac{H_j}{2} + z_{cj} \end{cases}, j = 1,2,3,\cdots,6 \tag{5-27}$$

其中 L_j， H_j， W_j 分别为油箱 j 的长宽高。

飞行器（不载油）时质心为(0,0,0)，总重量为 M，由于飞行器飞行时结构不会发生改变，所以在计算飞行器质心时不用考虑飞行器自身。结合前面 6 个油箱燃油的质心坐标和质量，共得到 6 个油箱内燃油的质心坐标 $(\mathrm{x}_j, \mathrm{y}_j, \mathrm{z}_j)$ 和质量 m_j，其中 $k = 1,2,\ldots,7$。利用组合体的质心公式计算飞行器（装油）时的质心在飞行器坐标系下的坐标 $c_1 = \left[c_{1x}, c_{1y}, c_{1z}\right]$ 为：

$$\begin{cases} c_{1x} = \dfrac{\sum_{j=1}^{6} m_j x_j}{\sum_{k=1}^{6} m_j} \\ c_{1y} = \dfrac{\sum_{j=1}^{6} m_j y_j}{\sum_{k=1}^{6} m_j} \\ c_{1z} = \dfrac{\sum_{j=1}^{6} m_j z_j}{\sum_{k=1}^{6} m_j} \end{cases}$$

5.2 模型求解

基于上述飞行器-燃油组合体质心模型求解算法流程如图 16 所示：

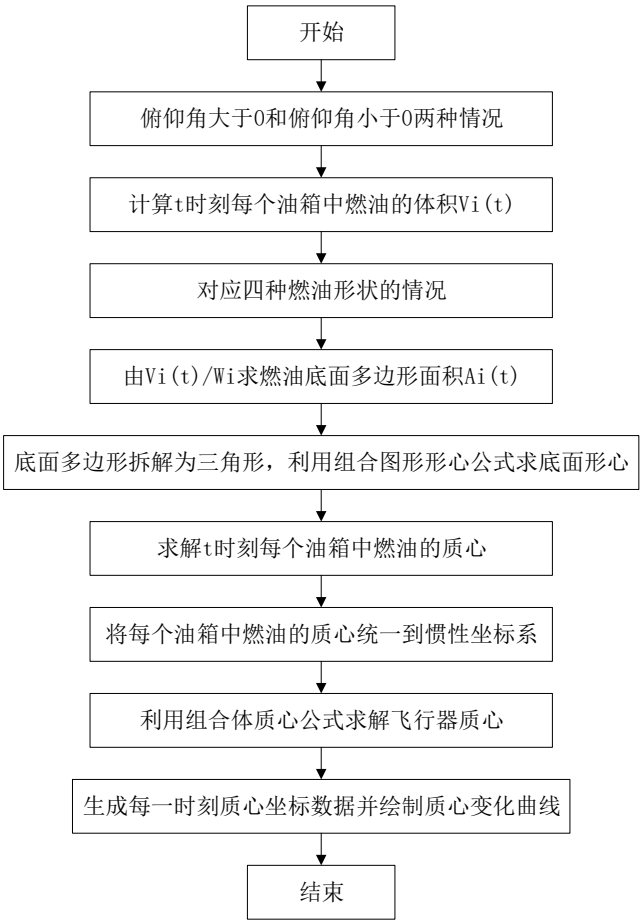


图 16 问题一算法流程图

5.3 求解结果

通过编写 python 和 matlab 程序得到飞行器每一时刻质心数据，结果保存在“附件 6-结果表.xlsx”中，飞行器质心变化曲线如下图所示：

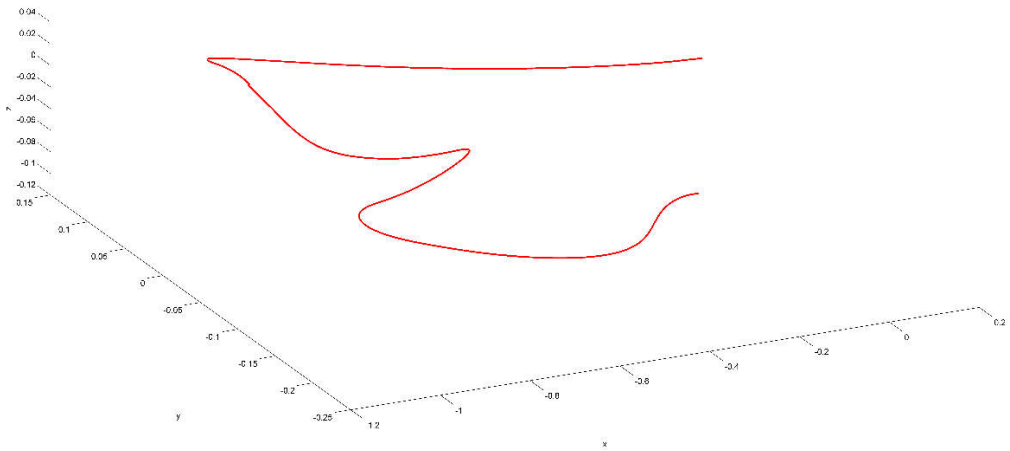


图 17 质心变化曲线。

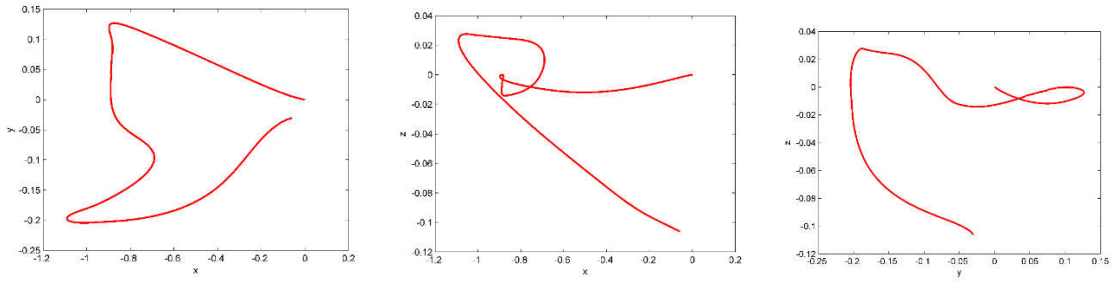


图 18 质心在不同平面投影曲线

5.4 结果分析

对飞行器 6 个油箱中燃油形状在不同时间进行分析，得到如下结果：

表 1 各时间段燃油形状分析

	燃油形状 1	燃油形状 2	燃油形状 3	燃油形状 4	燃油形状 5
油箱 1	66～620s 5324～5356s 6616～7199s	621～2330s	2331～2477s	2478～5323s 5357～6615s	0～65s 7200s
油箱 2	66～1209s 2710～3769s 4414～7199s	/	/	1210～2709s 3770～4413s	0～65s 7200s
油箱 3	66～605s 2073～3610s 4885～7199s	606～2072s	/	3611～4885s	0～65s 7200s
油箱 4	66～7199s	/	/	/	0～65s 7200s
油箱 5	66～526s 3115～3458s 4856～7199s	527～3114s 3459～4850s	/	/	0～65s 7200s
油箱 6	66～754s 2956～3575s 4664～7199s	755～2955s 3576～4663s	/	/	0～65s 7200s

质心变化曲线平滑，满足连续时间内质心的变化。

有 N 个飞行器运行质心，需要建立 N*3 的矩阵来存储，有 N*6 个油桶的数据，需要建立 N*6 的矩阵来存储，空间复杂度为 O(9*N)，首先计算每个时刻的供油后的体积，时间复杂度为 O(6*N)，接下来采用通过面积计算质心算法，每个油桶通过油量计算三角形或者四边形的质心，然后将面积质心平移到油桶的 Y 轴中心。每次求解面积需要 M 次简单四则运算，时间复杂度为 O(6*N*M)，空间复杂度为 O(9*N)。程序总运行时间（读 IO，算法处理，写 IO）3.89s。问题一得到的质心空间曲线的光顺度近似，问题 2 给出的数据的质心空间曲线的光顺度，表明算法的有效性。

六、 问题二模型建立和求解

6.1 飞行器水平飞行-质心模型建立

6.1.1 约束条件

油箱供油速度约束：

$$O_i(t) \le U_i \tag{6-1}$$

其中$O_i(t)$表示表示t时刻油箱i的供油速度，U_i表示油箱i的最大供油速度。

总供油量的约束：每个油箱的供油量加在一起大于飞机总共需要的供油量。

$$\sum_{i=2}^{5} O_i(t) \ge O(t) \tag{6-2}$$

其中$O(t)$表示t时刻战斗机总共需要的供油量。

油箱供油时间约束：

每个油箱一次供油的持续时间不少于 60 秒。

$$O_i(t) \sim O_i(t+59) > 0, O_i(t) > 0 \tag{6-3}$$

油箱选择约束：

$D_i(t)$表示t时刻是否使用油箱i进行供油，表达如下：

$$D_i(t) = \begin{cases} 1, O_i(t) > 0 \\ 0, O_i(t) = 0 \end{cases} \tag{6-4}$$

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。

$$\begin{cases} 1 \le \sum_{i=2}^{5} D_i(t) \le 2 \\ 1 \le \sum_{i=1}^{6} D_i(t) \le 3 \end{cases} \tag{6-5}$$

6.1.2 飞行器质心求解

求解t时刻飞行器质心$c_1(t)$前先要求解飞行器质心需要先求解飞行器中每一个油箱中燃油的质心，方法同问题一中求解第五种形状下i个油箱中燃油的质心坐标$(x_{ci}(t), y_{ci}(t), z_{ci}(t))$：

$$\begin{cases} x_{ci}(t)=\dfrac{H_i}{2}=\dfrac{A_i(t)}{2L_i} \\ y_{ci}(t)=\dfrac{W_i}{2} \\ z_{ci}(t)=\dfrac{L_i}{2} \end{cases} ,i=1,2,3,\cdots,6 \tag{6-6}$$

其中 $A_i=\dfrac{v_i(t)}{W_i}$，其中 L_j，H_j，W_j 分别为油箱 j 的长宽高。

同问题一解法一样求得 t 时刻油箱 i 中的燃油质心在飞行器坐标系下的坐标 $\left(\mathrm{x}_i(t),\mathrm{y}_i(t),\mathrm{z}_i(t)\right)$ 为：

$$\begin{cases} x_i(t)=x_{0i}(t)-\dfrac{\mathrm{L}_i}{2}+x_{ci}(t) \\ y_i(t)=\mathrm{y}_{0i}(t) \\ z_i(t)=z_{0i}(t)-\dfrac{H_j}{2}+z_{ci}(t) \end{cases} ，i=1,2,3,\cdots,6 \tag{6-7}$$

同问题一解法一样求得 t 时刻飞行器在飞行器坐标系下的质点坐标 $\left(c_{1x}(t),c_{1y}(t),c_{1z}(t)\right)$ 为：

$$\begin{cases} c_{1x}(t)=\dfrac{\sum_{i=1}^{N}m_i(t)x_c(t)}{\sum_{i=1}^{N}m_i(t)} \\ c_{1y}(t)=\dfrac{\sum_{i=1}^{N}m_i(t)y_c(t)}{\sum_{i=1}^{N}m_i(t)} \\ c_{1z}(t)=\dfrac{\sum_{i=1}^{N}m_i(t)z_c(t)}{\sum_{i=1}^{N}m_i(t)} \end{cases} \tag{6-8}$$

其中 $v_i(t)$ 为 t 时刻 i 号油箱内剩余的油体积，定义为：

$$\begin{cases} v_i(t)=v_i(t-1)-O_i(t),i\neq 2,5 \\ v_2(t)=v_2(t-1)-O_2(t)+O_1(t) \\ v_5(t)=v_5(t-1)-O_5(t)+O_6(t) \end{cases} \tag{6-9}$$

其中与副油箱连接的 2，5 号油箱内燃油体积不能超过其油箱的体积：

$$v_i(t) \le v_i, i = 2,5 \tag{6-10}$$

其中 v_i 代表油箱 i 的体积。

把每个油箱看作质点，可以求解飞行器的质点 $c_1(t) = \left[c_{1x}(t), c_{1y}(t), c_{1z}(t)\right]$ 为：

其中 $m_i(t)$ 为 t 时刻 i 号油箱油的重量。

$$m_i(t) = v_i(t) \cdot \rho \tag{6-11}$$

其中 ρ 为油的密度。

6.1.3 目标函数

为本次任务制定油箱供油策略，使得飞行器瞬时质心$\vec{c}_1(t)$与飞行器（不载油）质心$\vec{c}_0$的最大距离达到最小，即：

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

7.1.4 模型展示

根据以上变量、目标及约束条件的分析，建立飞机供油的动态模型：

其中 $\vec{c}_i(t) = \left[c_i(1), c_i(2), ..., c_i(t_{end})\right], i = 1,2$ 。

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

$$
s.t.\begin{cases}
O_i(t) \le U_i \\
\sum_{i=2}^{5} O_i(t) \ge O(t) \\
O_i(t) \sim O_i(t+59) > 0, O_i(t) > 0 \\
D_i(t) = \begin{cases} 1, O_i(t) > 0 \\ 0, O_i(t) = 0 \end{cases} \\
\begin{cases} 1 \le \sum_{i=2}^{5} D_i(t) \le 2 \\ 1 \le \sum_{i=1}^{6} D_i(t) \le 3 \end{cases} \\
v_i(t) \le v_i, i = 2,5 \\
v_i(t) = v_i(t-1) - O_i(t) \\
\begin{cases} x_{ci}(t) = \dfrac{H_i}{2} = \dfrac{A_i(t)}{2L_i} \\ y_{ci}(t) = \dfrac{W_i}{2} \qquad\qquad , i = 1,2,3,\cdots,6 \\ z_{ci}(t) = \dfrac{L_i}{2} \end{cases} \\
A_i = \dfrac{v_i(t)}{W_i(t)} \\
\begin{cases} x_i(t) = x_{0i}(t) - \dfrac{\mathrm{L}_i(t)}{2} + x_{ci}(t) \\ y_i(t) = \mathrm{y}_{0i}(t) \qquad\qquad , \ i = 1,2,3,\cdots,6 \\ z_i(t) = z_{0i}(t) - \dfrac{H_j(t)}{2} + z_{ci}(t) \end{cases} \\
m_i(t) = v_i(t) \cdot \rho \\
\begin{cases} c_{1x}(t) = \dfrac{\sum_{j=1}^{6} m_j(t) x_j(t)}{\sum_{k=1}^{6} m_j(t)} \\ c_{1y}(t) = \dfrac{\sum_{j=1}^{6} m_j(t) y_j(t)}{\sum_{k=1}^{6} m_j(t)} \\ c_{1z}(t) = \dfrac{\sum_{j=1}^{6} m_j(t) z_j(t)}{\sum_{k=1}^{6} m_j(t)} \end{cases}
\end{cases} \tag{6-12}
$$

6.2 基于理想质心偏移补偿法的模型求解

6.2.1 求解分析

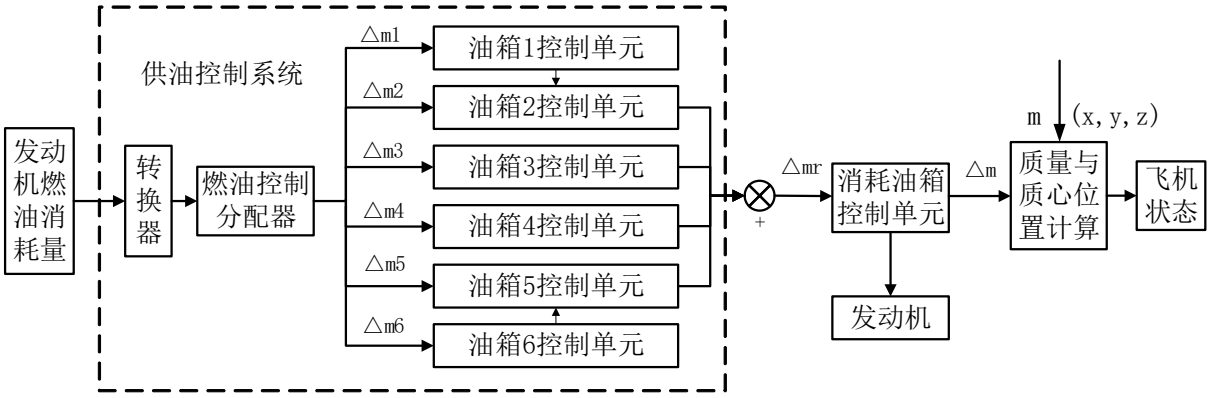


图 19 供油控制系统图

因为飞行器执行任务时需要满足瞬时质心接近理想质心和油耗满足耗油速度这两个条件，我们首先基于相邻时间前后两个质心位置得到理想质心偏移向量，对理想质心偏移向量进行向量合成，然后遍历油箱供油的组合方式，最后在约束条件下通过油箱供油的组合方式遍历得到每个油箱的供油量和瞬时质心坐标。

理想质心偏移向量合成补偿

首先计算t到$t+1$时刻相邻两个理想质心坐标的偏移向量 $\vec{P}(t)$：

$$\vec{P}(t)=(x(t+1)-x(t),y(t+1)-y(t),z(t+1)-z(t)) \tag{6-13}$$

其中$(x(t),y(t),z(t))$为t时刻的理想质心坐标。

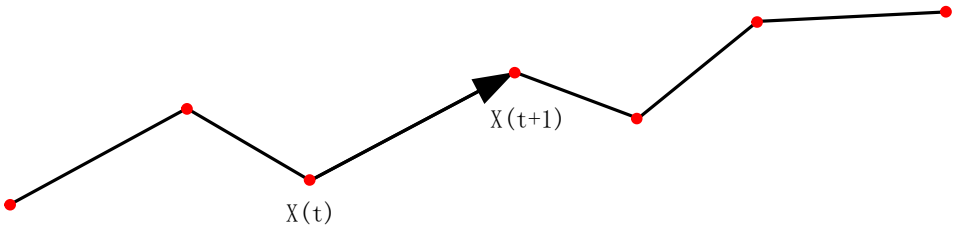


图 20 相邻理想质心偏移向量

t时刻，第i个油箱向发动机输送燃油，单位（1kg）油量减少对整个飞行器质心会产生偏移向量$\vec{r}_i(t)$：

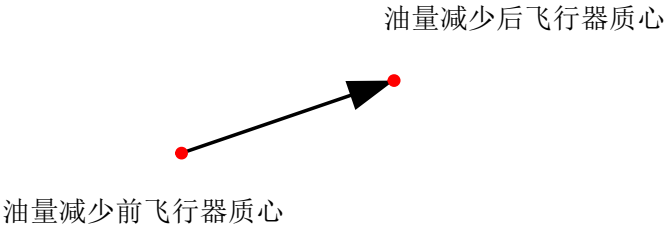


图 21 第i个油箱减少单位油量飞行器的质心偏移向量

存在油箱向油箱供油的情况，即 1 号油箱向 2 号油箱供油或者 6 号油箱向 5 号油箱供油产生如下的质心偏移情况：

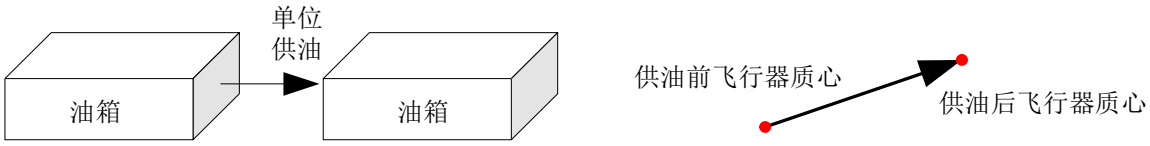


图 22 油箱向油箱供油对飞行器的质心偏移向量

偏移向量可以通过组合体质心公式$\vec{r}_i(\mathrm{t})=\dfrac{\sum m_k(\mathrm{t})\vec{q}_k(\mathrm{t})}{\sum m_k(\mathrm{t})},k=1,2,...,7$求得，$m_k(\mathrm{t})$为飞行器七个组成部分 t 时刻的质量，$\vec{q}_k(\mathrm{t})$为七个部分各自的质心。

一个油箱油量减少对飞行器质心产生的偏移影响，可能不满足飞行器从一个理想质心移动到下一个理想质心，所以需要考率多个油箱油量变化时对飞行器质心的影响,通过向量合成的方法使飞行器质心移到理想位置，如图 23 所示：

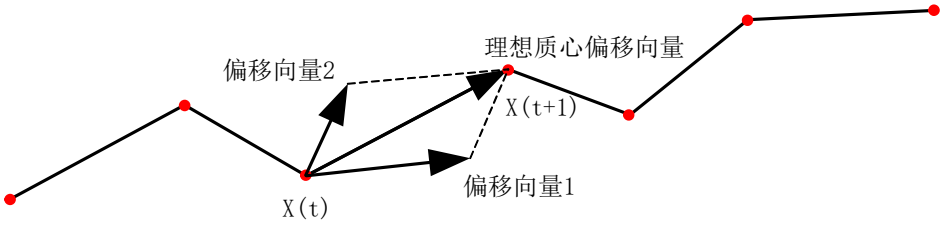


图 23 理想质心偏移向量合成方法

供油油箱选择

对每个油箱进行 01 编码，如图 24 所示,0 表示油箱不供油，1 表示油箱供油，即图中表示第 3 号、5 号和 6 号油箱向发动机或者其他油箱供油。

0	1	0	0	1	1
油箱1	油箱2	油箱3	油箱4	油箱5	油箱6

图 24 油箱供油编码

在约束条件下确定油箱供油的组合方式，即至多 2 个油箱可同时向发动机供油，至多 3 个油箱可同时供油，得到如下油箱供油位码表，其中红色部分表示油箱向油箱供油的情况。

表 2 油箱供油位码表

一个油箱供油	两个油箱供油		三个油箱供油	
010000	110000	001100	111000	011100

001000	000011	001010	110100	011010
000100	011000	000110	110010	010110
000010	010100		000111	001110
	010010		001011	

即存在单油箱，双油箱和三油箱对飞机理想质心偏移产生影响，我们在每一个时间 t 对表 2 中供油方案逐一进行遍历，在约束条件内确定理想质心偏移向量合成方案，使得飞行器每一时刻的质心位置$\vec{c}_1(t)$与理想质心位置$\vec{c}_2(t)$的欧氏距离达到最小：

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

多油箱供油速度的分配方案

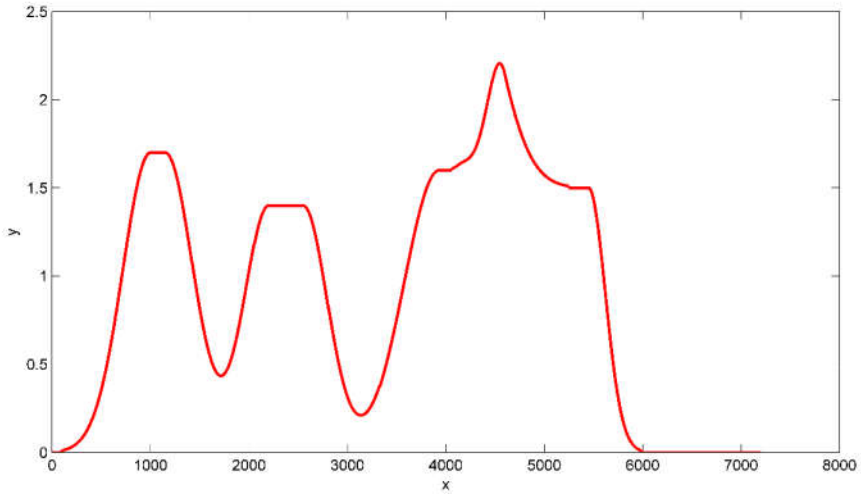


图 25 飞行器发动机耗油速度曲线

飞行器发动机耗油速度曲线如上图所示，四个主油箱向发动机供油速度需要大于发动机耗油速度，为了方便求解，我们将四个主油箱的供油速度和发动机耗油速度相等，供油速度曲线即为耗油速度曲线。

以 0.01kg 为精度对发动机在每一时刻的瞬时油耗进行划分，分配给不同的油箱进行供油，如发动机瞬时油耗为 2.2kg，可划分为 1.2kg 和 1.0kg，也可划分为 1.4kg 和 0.8kg，油箱供油存在多种组合方式。

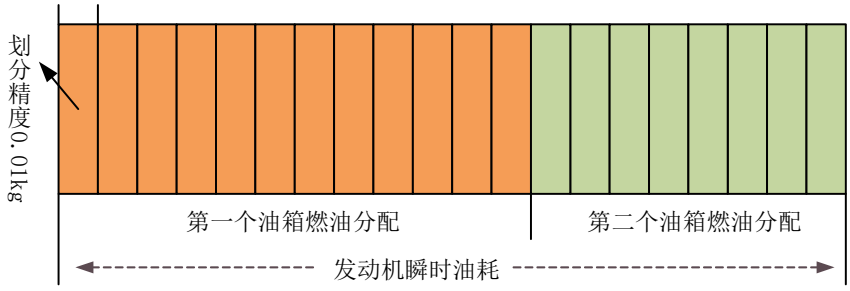


图 26 不同的油箱供油分配示意图

这样确定供油的方案，用$t\sim t+1$时刻油箱供油产生的质心偏移对$t\sim t+1$到理想质心的偏移进行补偿，可以使得补偿以后得到的距离达到最小。

6.2.2 算法流程

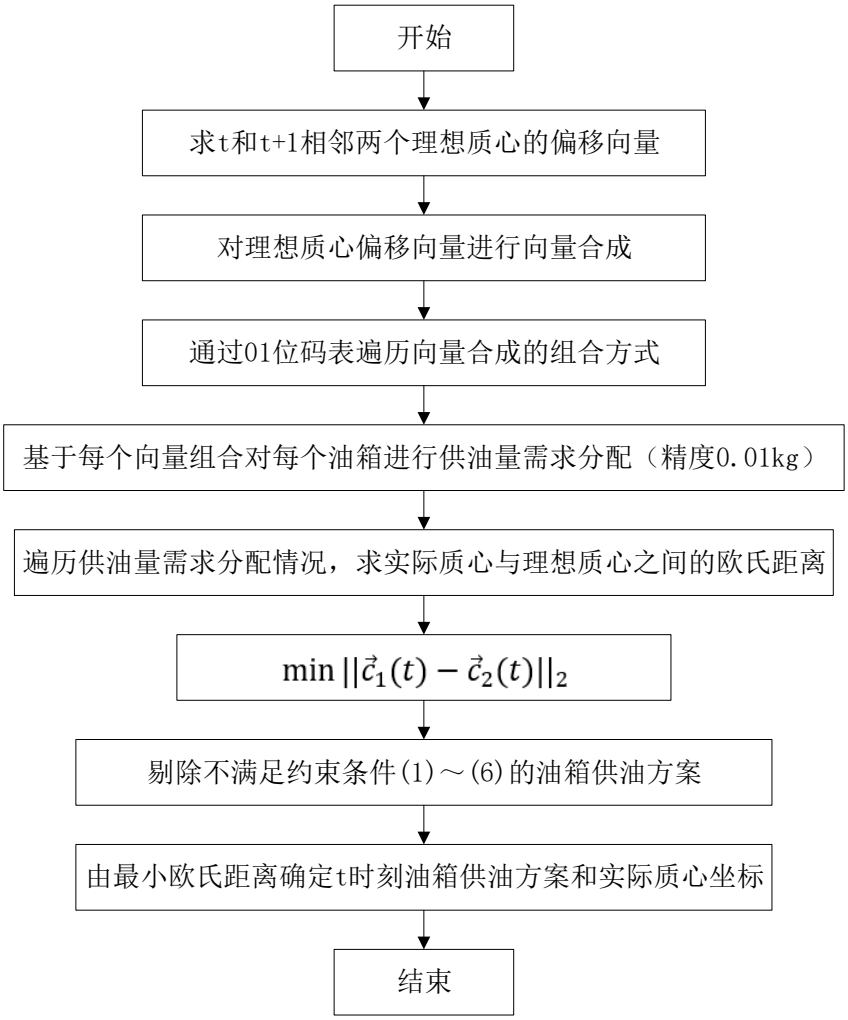


图 27 问题二算法流程图

6.3 求解结果

通过编写 python 和 matlab 程序在约束条件下得到飞行器飞行过程中 6 个油箱各自的供油速度曲线和 4 个主油箱的总供油速度曲线(时间间隔为 1s)如下：

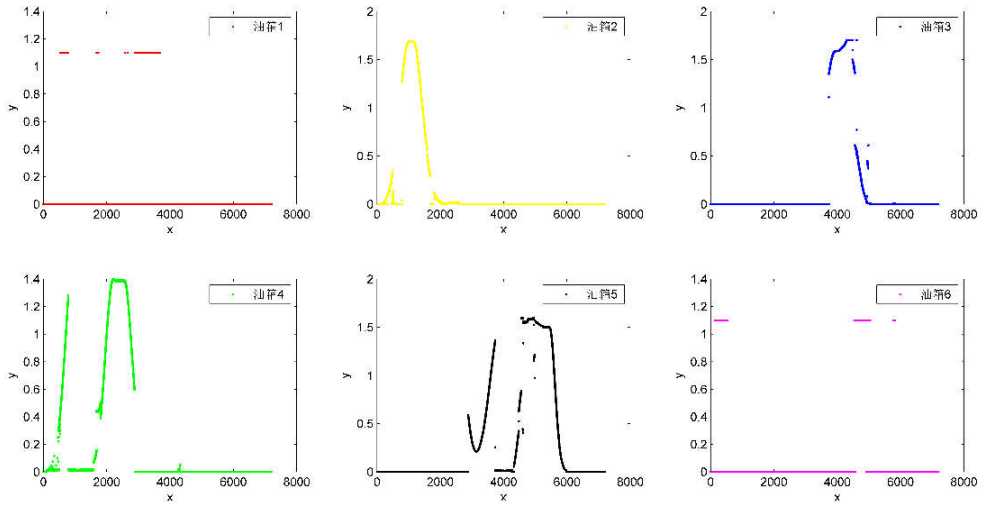


图 28 6 个油箱各自的供油速度曲线

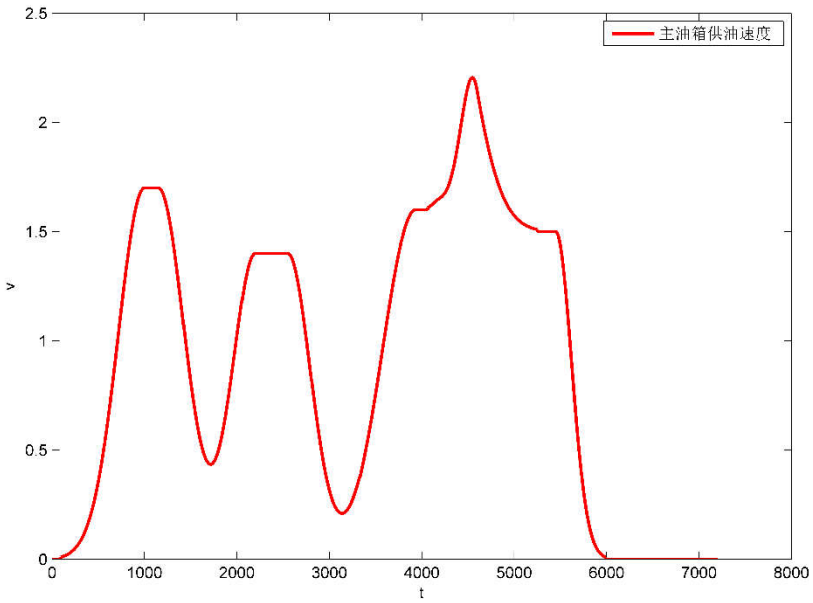


图 29 4 个主油总的箱供油速度曲线

得到飞行器瞬时质心与理想质心的距离如图 29，**最大距离最小值为：0.068697722m，4 个主油箱的总供油量为 6441.524212kg。**

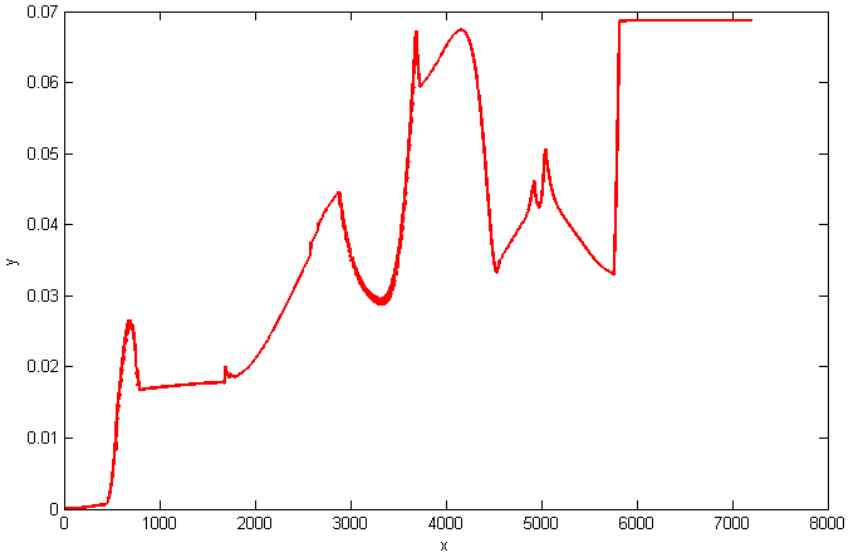


图 30 飞行器瞬时质心与理想质心的距离

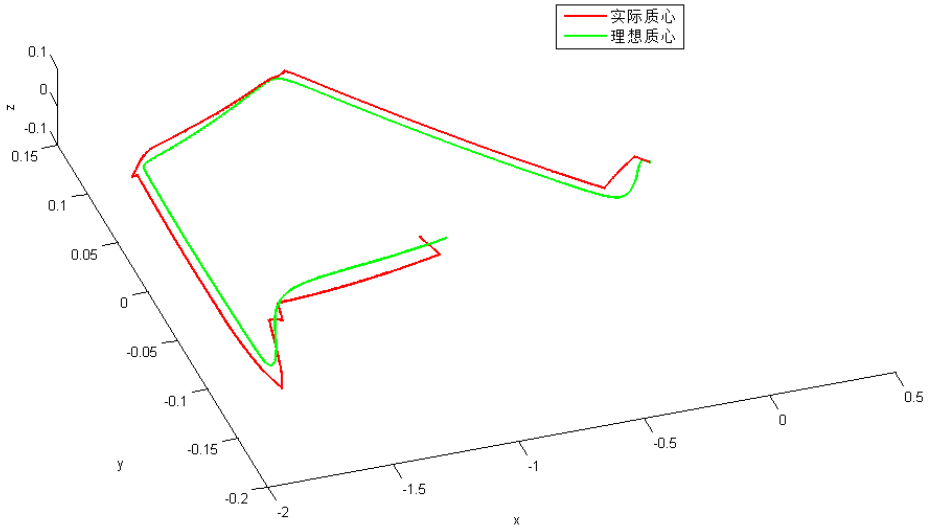


图 31 飞行器瞬时质心与理想质心三维图

6.4 结果分析

我们假设油箱向发动机供油的过程中，供油和耗油相等，供油不产生多余的燃油，所以四个主油箱的总供油量等于飞行器耗油 6441.524212kg。

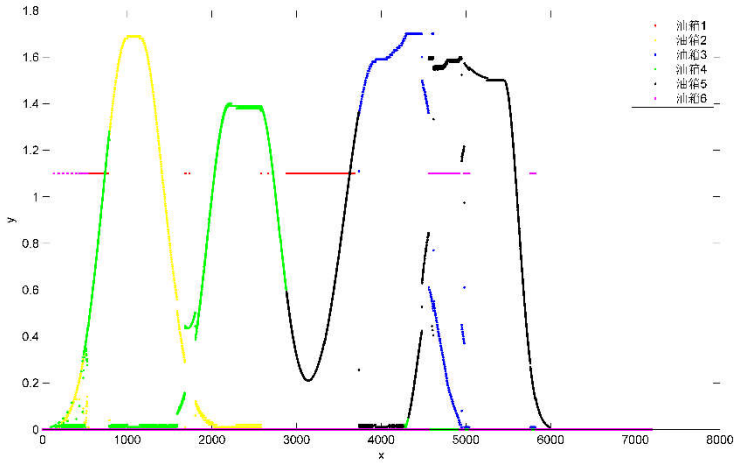


图 32 6 个油箱供油速度曲线

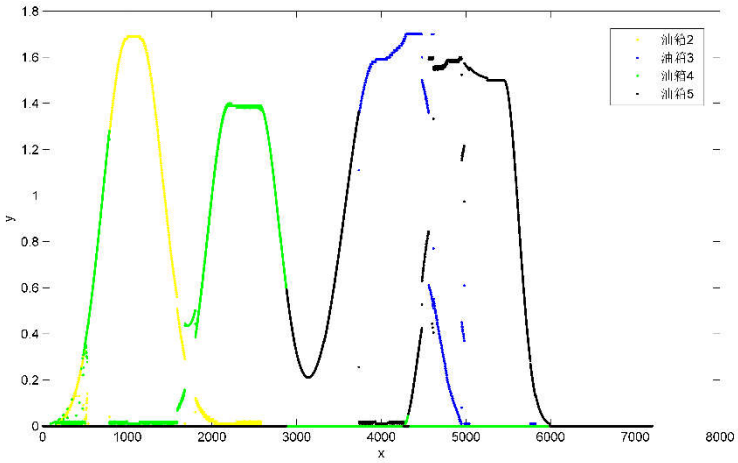


图 33 4 个油箱供油速度曲线

有 N 个时间点数据，需要 N 长度的数组存储发动机的耗油量，需要 3*N 长度的数组存储飞行器理想质心数据。为了生成有效的方案，需要计算每个油桶投影在“理想向量”模的长度。时间复杂度 O(M*6*N)，M 为每次为了计算投影向量的平均算法步骤。每个方案要进行平均 1000 次的最优值搜索，时间复杂度为 O(1000N)。空间复杂度 O(4*N)，总的时间复杂度是 O(M*6*N+1000N)。程序的运行总时间 464.3s。实验结果满足约束条件，在此情况下得到的瞬时质心与理想质心的最大值为 0.068697722m，在误差允许范围内，近似最优解，表明算法的有效性。

七、 问题三模型建立和求解

7.1 飞行器水平飞行-质心模型建立

7.1.1 约束条件

油箱供油速度约束：

$$O_i(t) \le U_i \tag{7-1}$$

其中$O_i(t)$表示表示t时刻油箱i的供油速度，U_i表示油箱i的最大供油速度。

总供油量的约束：每个油箱的供油量加在一起大于飞机总共需要的供油量。

$$\sum_{i=2}^{5} O_i(t) \ge O(t) \tag{7-2}$$

其中$O(t)$表示t时刻战斗机总共需要的供油量。

油箱供油时间约束：

每个油箱一次供油的持续时间不少于 60 秒。

$$O_i(t) \sim O_i(t+59) > 0, O_i(t) > 0 \tag{7-3}$$

油箱选择约束：

$D_i(t)$表示t时刻是否使用油箱i进行供油，表达如下：

$$D_i(t) = \begin{cases} 1, O_i(t) > 0 \\ 0, O_i(t) = 0 \end{cases} \tag{7-4}$$

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。

$$\begin{cases} 1 \le \sum_{i=2}^{5} D_i(t) \le 2 \\ 1 \le \sum_{i=1}^{6} D_i(t) \le 3 \end{cases} \tag{7-5}$$

剩余油量约束条件：

$$\sum_{i=1}^{6} v_i(t_{end}) \ge 1 \tag{7-6}$$

在飞行结束时刻t_{end} 6 个油箱剩余的总油量大于 1。

7.1.2 飞行器质心求解

飞行器质心的求解和问题二求解方法相同。

求解t时刻飞行器质心$c_1(t)$前先要求解飞行器质心需要先求解飞行器中每一个油箱中燃油的质心，方法同问题一中求解第五种形状下i个油箱中燃油的质心坐标$\left(x_{ci}(t),y_{ci}(t),z_{ci}(t)\right)$：

$$\begin{cases} x_{ci}(t)=\dfrac{H_i}{2}=\dfrac{A_i(t)}{2L_i} \\ y_{ci}(t)=\dfrac{W_i}{2} \\ z_{ci}(t)=\dfrac{L_i}{2} \end{cases} ,i=1,2,3,\cdots,6 \tag{7-7}$$

其中$A_i=\dfrac{v_i(t)}{W_i}$，其中L_j，H_j，W_j分别为油箱j的长宽高。

同问题一解法一样求得t时刻油箱i中的燃油质心在飞行器坐标系下的坐标$\left(\mathrm{x}_i(t),\mathrm{y}_i(t),\mathrm{z}_i(t)\right)$为：

$$\begin{cases} x_i(t)=x_{0i}(t)-\dfrac{\mathrm{L}_i}{2}+x_{ci}(t) \\ y_i(t)=\mathrm{y}_{0i}(t) \\ z_i(t)=z_{0i}(t)-\dfrac{H_j}{2}+z_{ci}(t) \end{cases} ，i=1,2,3,\cdots,6 \tag{7-8}$$

同问题一解法一样求得t时刻飞行器在飞行器坐标系下的质点坐标$\left(c_{1x}(t),c_{1y}(t),c_{1z}(t)\right)$为：

$$\begin{cases} c_{1x}(t)=\dfrac{\sum_{i=1}^{N}m_i(t)x_c(t)}{\sum_{i=1}^{N}m_i(t)} \\ c_{1y}(t)=\dfrac{\sum_{i=1}^{N}m_i(t)y_c(t)}{\sum_{i=1}^{N}m_i(t)} \\ c_{1z}(t)=\dfrac{\sum_{i=1}^{N}m_i(t)z_c(t)}{\sum_{i=1}^{N}m_i(t)} \end{cases} \tag{7-9}$$

其中$v_i(t)$为t时刻i号油箱内剩余的油体积，定义为：

$$\begin{cases} v_i(t) = v_i(t-1) - O_i(t), i \neq 2,5 \\ v_2(t) = v_2(t-1) - O_2(t) + O_1(t) \\ v_5(t) = v_5(t-1) - O_5(t) + O_6(t) \end{cases} \tag{7-10}$$

其中与副油箱连接的 2，5 号油箱内燃油体积不能超过其油箱的体积：

$$v_i(t) \leq v_i, i = 2,5 \tag{7-11}$$

其中 v_i 代表油箱 i 的体积。

把每个油箱看作质点，可以求解飞行器的质点 $c_1(t) = \left[c_{1x}(t), c_{1y}(t), c_{1z}(t)\right]$ 为：

其中 $m_i(t)$ 为 t 时刻 i 号油箱油的重量。

$$m_i(t) = v_i(t) \cdot \rho \tag{7-12}$$

其中 ρ 为油的密度。

7.1.3 目标函数

为本次任务制定油箱供油策略，使得飞行器瞬时质心$\vec{c}_1(t)$与飞行器（不载油）质心$\vec{c}_0$的最大距离达到最小，即：

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

7.1.4 模型展示

根据以上变量、目标及约束条件的分析，建立飞机供油的动态模型：

其中 $\vec{c}_i(t) = \left[c_i(1), c_i(2), ..., c_i(t_{end})\right], i = 1,2$ 。

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

$$
s.t.\begin{cases}
O_i(t)\le U_i;\sum_{i=2}^{5}O_i(t)\ge O(t);O_i(t)\sim O_i(t+59)>0,O_i(t)>0\\
D_i(t)=\begin{cases}1,O_i(t)>0\\0,O_i(t)=0\end{cases}\\
\begin{cases}1\le\sum_{i=2}^{5}D_i(t)\le 2\\1\le\sum_{i=1}^{6}D_i(t)\le 3\end{cases}\\
v_i(t)\le v_i,i=2,5;v_i(t)=v_i(t-1)-O_i(t)\\
\begin{cases}x_{ci}(t)=\dfrac{H_i}{2}=\dfrac{A_i(t)}{2L_i}\\ y_{ci}(t)=\dfrac{W_i}{2}\\ z_{ci}(t)=\dfrac{L_i}{2}\end{cases}\quad,i=1,2,3,\cdots,6\\
A_i=\dfrac{v_i(t)}{W_i(t)}\\
\begin{cases}x_i(t)=x_{0i}(t)-\dfrac{\mathrm{L}_i(t)}{2}+x_{ci}(t)\\ y_i(t)=\mathrm{y}_{0i}(t)\\ z_i(t)=z_{0i}(t)-\dfrac{H_j(t)}{2}+z_{ci}(t)\end{cases}\quad,\ i=1,2,3,\cdots,6\\
m_i(t)=v_i(t)\cdot\rho\\
c_{1x}(t)=\dfrac{\sum_{j=1}^{6}m_j(t)x_j(t)}{\sum_{k=1}^{6}m_j(t)};c_{1y}(t)=\dfrac{\sum_{j=1}^{6}m_j(t)y_j(t)}{\sum_{k=1}^{6}m_j(t)};c_{1z}(t)=\dfrac{\sum_{j=1}^{6}m_j(t)z_j(t)}{\sum_{k=1}^{6}m_j(t)}\\
\sum_{i=1}^{6}v_i(t_{end})\ge 1
\end{cases}
\tag{7-13}
$$

7.2 基于粒子群算法以及倒序质心偏移补偿算法模型求解

7.2.1 求解分析

在问题三中，由于未给每个油箱中的定初始油量$v_i(0)$，假设飞行器油箱中的燃油都只供给飞行器发动机，不会把燃油直接排出飞行器外。

这样可以根据飞行结束后飞行器中油箱中燃油剩余的体积$v_i\left(t_{end}\right)$，以及飞行器t时刻的油耗$O(t)$。根据问题二的方法使用倒序偏移补偿算法求得从t_{end}时刻到t_0时刻的油箱i的供油速度$\vec{O}_i(t)$，以及使得飞行器瞬时质心$\vec{c}_1(t)$与飞行器(不载油)质心$\vec{c}_0$的最大距离达到最小时$d_{\min}=\min\max_t\|\vec{c}_1(t)-\vec{c}_2(t)\|_2$。

本问题的关键在于求解飞行结束后飞行器油箱中燃油剩余的体积$v_i\left(t_{end}\right)$。使得飞行器瞬时质心$\vec{c}_1(t)$与飞行器(不载油)质心$\vec{c}_0$的最大距离达到最小$d_{\min}=\min\max_t\|\vec{c}_1(t)-\vec{c}_2(t)\|_2$。

倒序质心偏移补偿算法：

在问题二中以最后时刻为起始点，首先计算t到$t-1$时刻相邻两个理想质心坐标的偏移向量$\vec{p}(t)$：

$$\vec{P}(t)=(x(t)-x(t-1),y(t)-y(t-1),z(t)-z(t-1)) \tag{7-14}$$

同问题二一样设计$t\sim t-1$时刻的供油方案，用$t\sim t-1$时刻油箱供油产生的质心偏移对$t\sim t-1$到理想质心质心偏移进行补偿，可以使得补偿以后得到的距离达到最小。

在已知飞行结束后飞行器油箱中燃油剩余体积，在每个时刻的理想质心位置，以及在每个时刻下的飞行器需要的总供油，利用上述算法可以计算出对应的供油策略。

7.2.2 模型求解

根据求解分析，选择采用粒子群算法求解：

粒子群记为$Q=[X_1,X_2,X_3,\cdots,X_N]$，其中$X_i=\left[x_{i1},x_{i2},x_{i3},\cdots,x_{iD}\right],i=1,2,3,\cdots,N$，其中$N$为粒子的个数。

第i个粒子的“飞行”速度，记为：$V_i=[v_{i1},v_{i2},v_{i3},\cdots,v_{iD}]$。

第i个粒子迄今为止搜索到的最优位置称为个体最优值，记为：$P_i=[P_{i1},P_{i2},P_{i3},\cdots,P_{iD}]$。

整个粒子群迄今为止搜索到的最优位置为全局最优值，记为：$P_g=[P_{g1},P_{g2},P_{g3},\cdots,P_{gD}]$。

粒子的速度更新公式为：

$$V_{id}^{k+1}=\omega V_{id}^{k}+b_1r_1\left(P_{id}^{k}-X_{jd}^{k}\right)+b_2r_2\left(P_{gd}^{k}-X_{id}^{k}\right) \tag{7-15}$$

粒子的位置更新公式为：

$$X_{id}^{k+1}=X_{id}^{k}+V_{id} \tag{7-16}$$

其中k代表迭代的次数，ω称为惯性因子其值为非负，较大时，全局寻优能力强，局部寻优能力强；较小时，全局寻优能力弱，局部寻优能力强。通过调整ω的大小，可

以对全局寻优性能和局部寻优性能进行调整b_1和b_2称为加速常数，前者为每个粒子的个体学习因子，后者为每个粒子的社会学习因子。r_1和r_2是$[0,1]$区间内均匀分布的随机数。

结合本题需要求解飞行结束后飞行器油箱中燃油剩余的体积，令粒子群为：

$$X_i=[x_{i1},x_{i2},x_{i3},x_{i4},x_{i5},x_{i6}],i=1,2,3,\cdots,N \tag{7-17}$$

其中x_{ij}代表粒子i中飞行结束后飞行器油箱j中燃油剩余的体积。且由于在飞行结束时刻6个油箱剩余的总油量要大于1，则粒子的坐标应满足：

$$\sum_{j=1}^{6}x_{ij}\geq 1 \tag{7-18}$$

由前面分析可知将粒子X_i中的值带入问题二中可以得到此时使得飞行器瞬时质心$\vec{c}_1(t)$与飞行器（不载油）质心$\vec{c}_0$的最大距离达到最小时对应的距离$d_{\min}=\min\limits_{t}\max\|\vec{c}_1(t)-\vec{c}_2(t)\|_2$，所以令粒子$X_i$的适应值为：

$$F_i=d_{\min}=\min\limits_{t}\max\|\vec{c}_1(t)-\vec{c}_2(t)\|_2 \tag{7-19}$$

算法的步骤如下：

Step1:初始化粒子群，包括群体的个数N，每个粒子的位置X_i和速度V_i，且X_i应满足式（7-17）。

Step2:计算每个粒子的适应度值F_i；

Step3: 对每个粒子，用它的适应度值F_i和个体最优值P_i比较，如果$F_i<P_i$，则用F_i替换掉P_i；

Step4: 对每个粒子，用它的适应度值F_i和全局极值P_g比较，如果$F_i<P_i$则用F_i替换掉P_g；

Step5: 根据公式（7-15），（7-16）更新粒子的速度V_i和位置X_i；

Step6: 如果满足结束条件(误差足够好或到达最大循环次数)，输出最优的适应值F_{opt}，以及对应的粒子坐标X_{opt}。否则返回Step2。

根据算法输出的适应值F_{opt}，X_{opt}代入问题二中就可以得到在飞行结束后飞行器各个油箱中燃油剩余的体积为x_{opt1}，x_{opt2}，x_{opt3}，x_{opt4}，x_{opt5}，x_{opt6}的情况下飞行器瞬时质心$\vec{c}_1(t)$与飞行器(不载油)质心$\vec{c}_0$的最大距离达到最小时对应的距离为F_{opt}，以及相应的

供油策略。

求解步骤如下图所示：

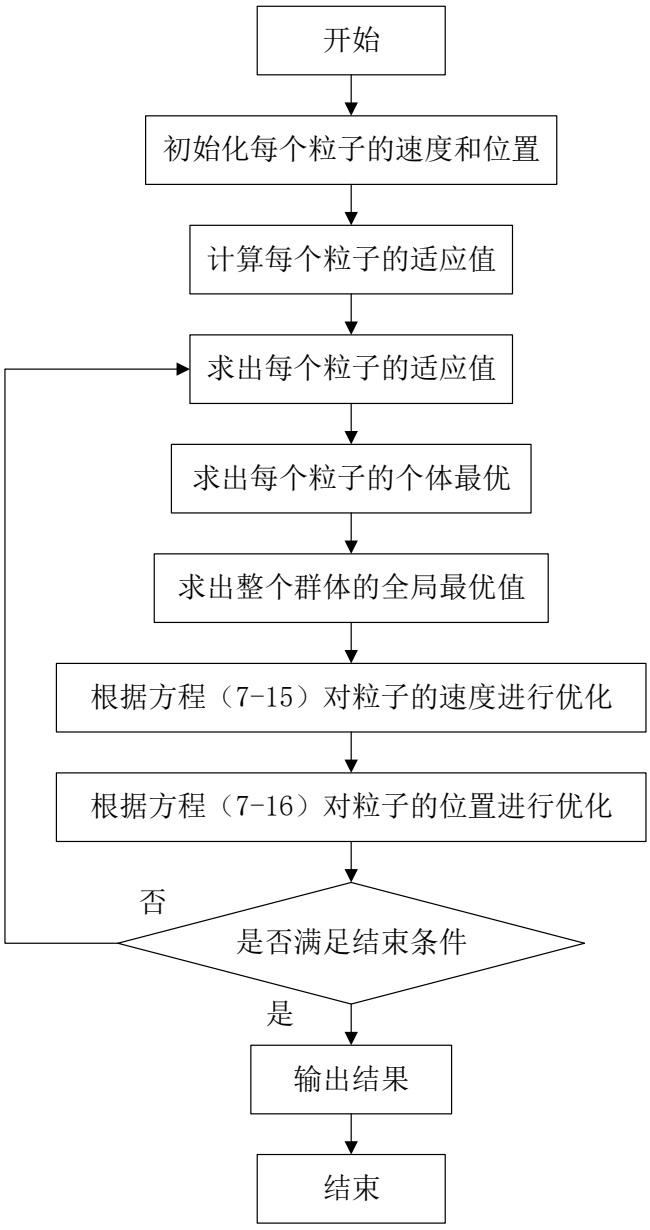


图 34 粒子群算法流程图

7.2.3 求解结果

初始化粒子群数量为 2000，初始化坐标为粒子在满足 $\sum_{j=1}^{6} x_{ij} = 1$ 的空间内的均匀分布的坐标，初始化速度为 0，令参数 $\omega = 0.6, b_1 = b_2 = 1.7$，粒子群算法收敛。

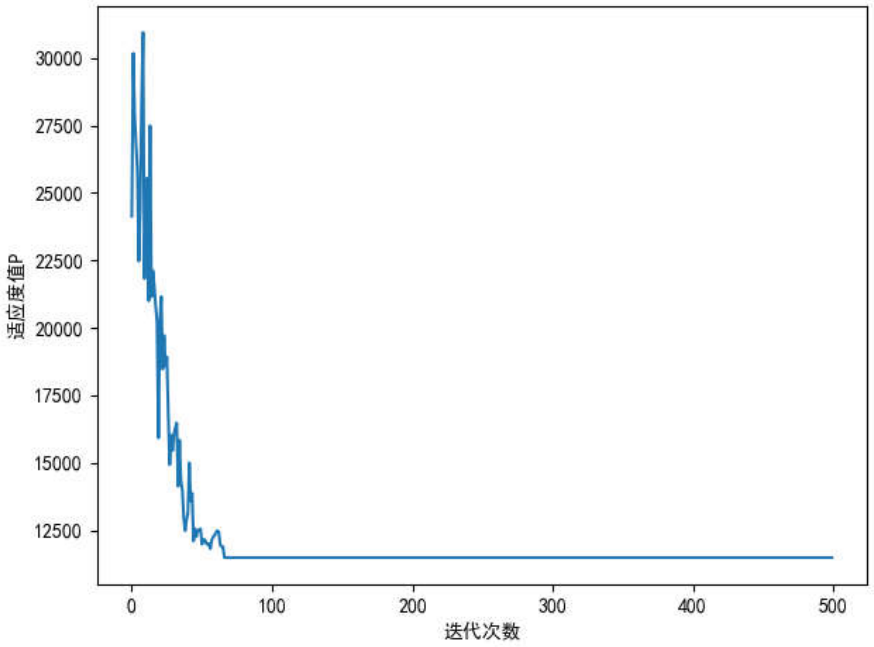


图 35 6 个油箱各自的供油速度曲线

通过编写 python 和 matlab 程序在约束条件下得到：

初始油量（kg）分别为：268.5, 1025.49, 1315.8, 2247.43269, 1778.151979, 1019.8

结束油量（kg）分别为：10， 10.79， 220.36， 84.6， 0.55， 523.7

飞行器飞行过程中 **6 个油箱各自的供油速度曲线和 4 个主油箱的总供油速度曲线**(时间间隔为 1s)如下：

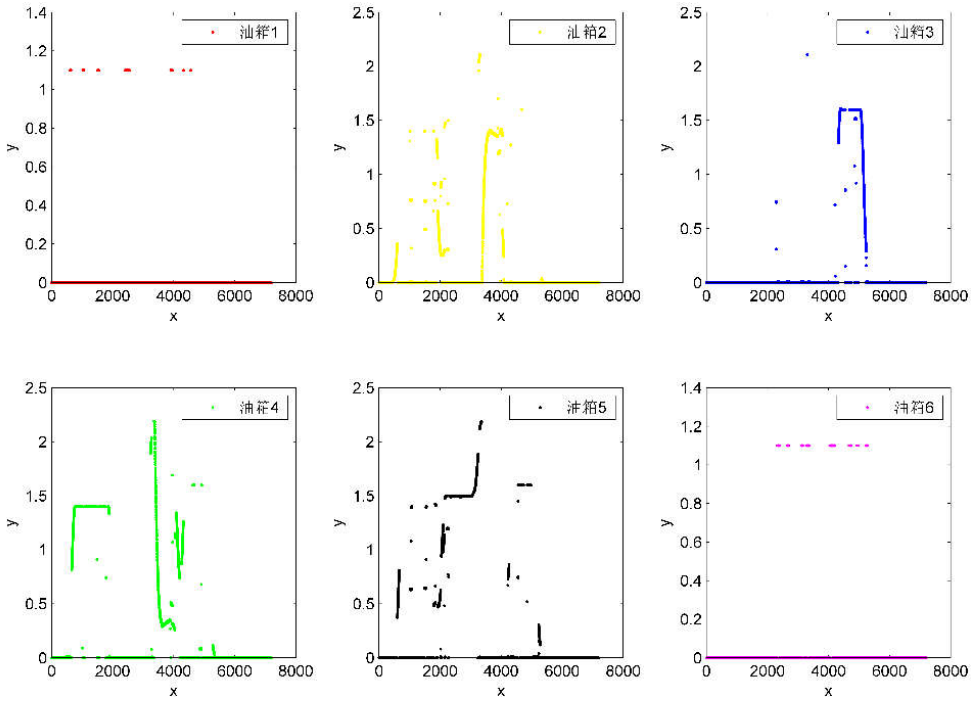


图 36 6 个油箱各自的供油速度曲线

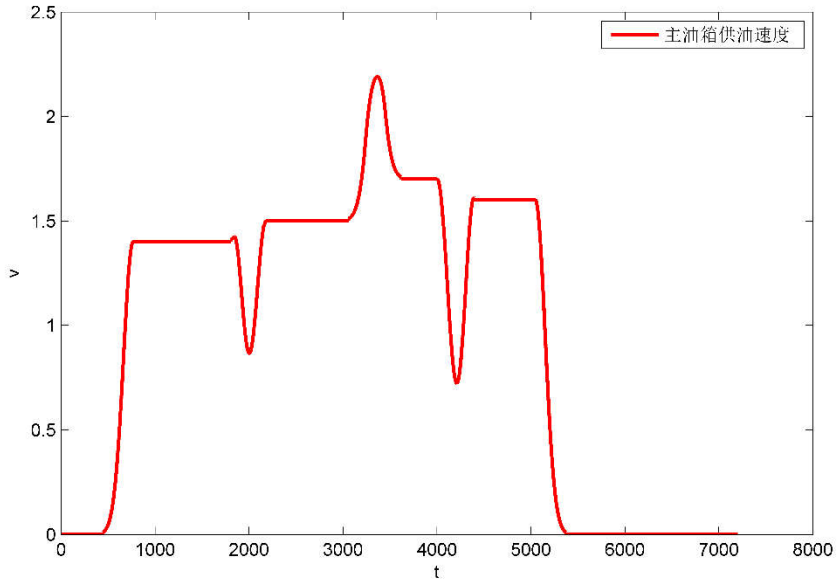


图 37 4 个主箱油总的总供油速度曲线

得到飞行器瞬时质心与理想质心的距离如图 38，**最大距离最小值为：0.120308m，4 个主油箱的总供油量为 6805.174669kg。**

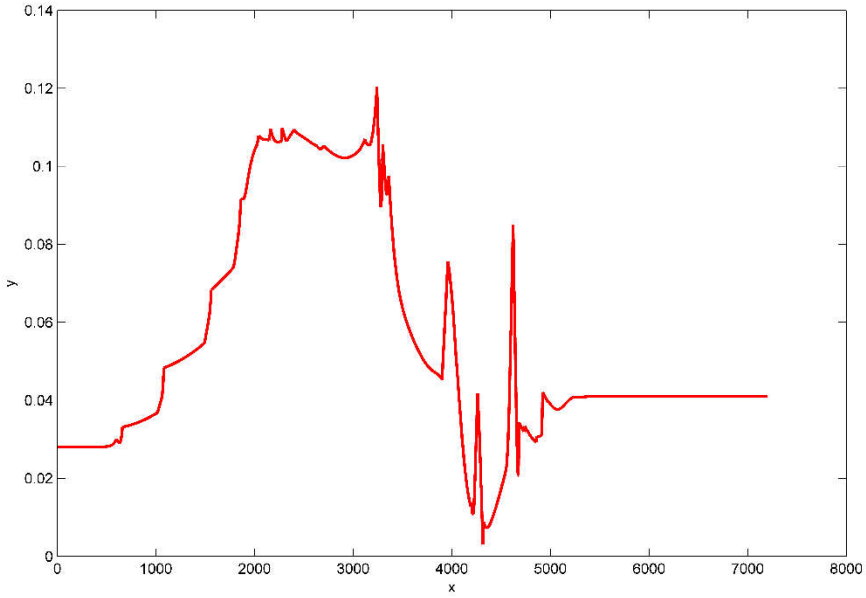


图 38 飞行器瞬时质心与理想质心的距离

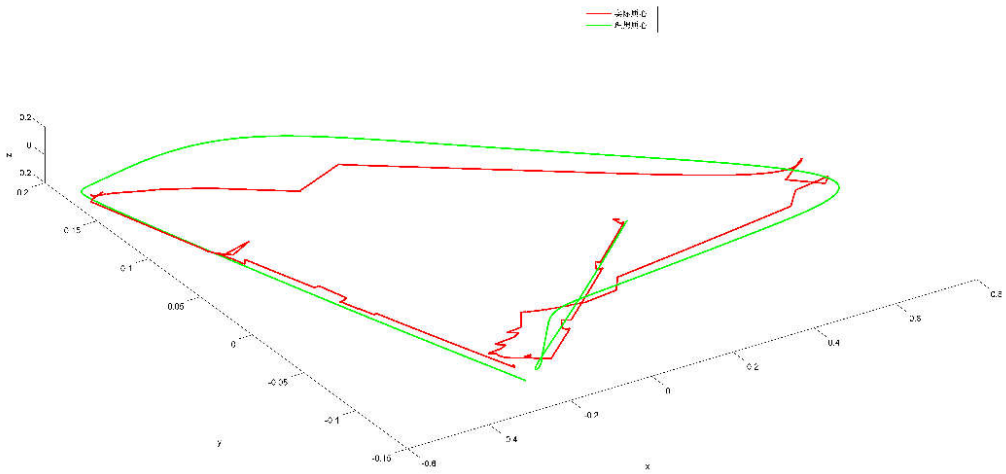


图 39 飞行器瞬时质心与理想质心三维图

7.3 结果分析

我们假设油箱向发动机供油的过程中，供油和耗油相等，供油不产生多余的燃油，所以四个主油箱的总供油量等于飞行器耗油 6805.174669kg。

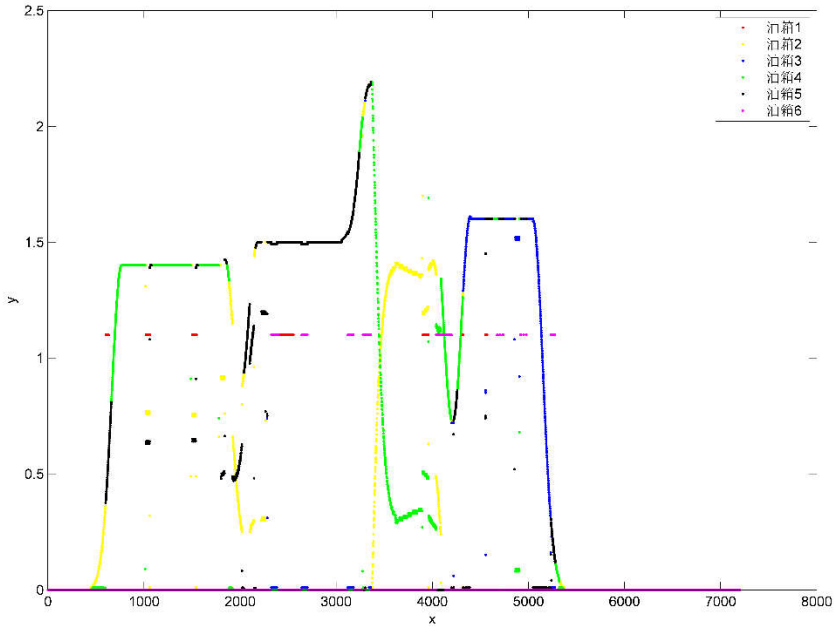


图 40 6 个油箱供油速度曲线

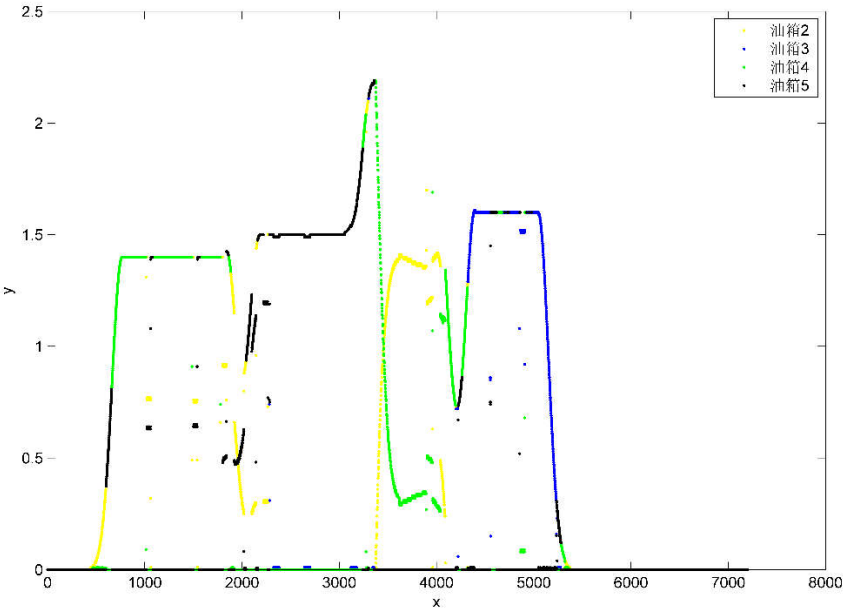


图 41 4 个油箱供油速度曲线

对于粒子点数为 N，维度为 D 的粒子群，需要 $N\times D$ 的矩阵进行存储，在每个粒子下的供油方案为需要 $T\times B$ 矩阵进行存储，其中 T 为时间的维度，B 为油箱的个数。算法的空间复杂度为 $O(N\times D\times T\times B)$。设迭代次数为 K,则倒序质心偏移补偿算法的时间复杂度为 $O(T)$，遗传算法的时间复杂度为 $O(K)$，则整个算法的时间复杂度为 $O(T+K)$，算法实际运行时间为 758.6s。本文算法求得满足约束条件下的油箱的初始油量（kg）分别为：268.5, 1025.49, 1315.8, 2247.43269, 1778.151979, 1019.8，结束油量（kg）分别为：10， 10.79， 220.36， 84.6， 0.55， 523.7，在此情况下得到的瞬时质心与理想质心的最大值为 0.120308m，在误差允许范围内，近似最优解，表明算法的有效性。

八、 问题四模型建立和求解

8.1 飞行器俯仰飞行-质心模型建立

8.1.1 约束条件

油箱供油速度约束：

$$O_i(t) \le U_i \tag{8-1}$$

其中$O_i(t)$表示表示t时刻油箱i的供油速度，U_i表示油箱i的最大供油速度。

总供油量的约束：每个油箱的供油量加在一起大于飞机总共需要的供油量。

$$\sum_{i=2}^{5} O_i(t) \ge O(t) \tag{8-2}$$

其中$O(t)$表示t时刻战斗机总共需要的供油量。

油箱供油时间约束：

每个油箱一次供油的持续时间不少于 60 秒。

$$O_i(t) \sim O_i(t+59) > 0, O_i(t) > 0 \tag{8-3}$$

油箱选择约束：

$D_i(t)$表示t时刻是否使用油箱i进行供油，表达如下：

$$D_i(t) = \begin{cases} 1, O_i(t) > 0 \\ 0, O_i(t) = 0 \end{cases} \tag{8-4}$$

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。

$$\begin{cases} 1 \le \sum_{i=2}^{5} D_i(t) \le 2 \\ 1 \le \sum_{i=1}^{6} D_i(t) \le 3 \end{cases} \tag{8-5}$$

8.1.2 飞行器质心求解

求解t时刻飞行器质心$c_1(t)$前先要求解飞行器质心需要先求解飞行器中每一个油箱中燃油的质心，将t时刻飞行器的俯仰角$\alpha(t)$，油箱i内的燃油的体积$v_i(t)$以及油箱i的尺寸$H_{ii} \times H_i \times W_i$，代入问题一中的模型求解得第$i$个油箱中燃油的质心坐标

$\left(x_{ci}(t), y_{ci}(t), z_{ci}(t)\right)$。

同问题一解法一样求得 t 时刻油箱 i 中的燃油质心在飞行器坐标系下的坐标 $\left(\mathrm{x}_i(t), \mathrm{y}_i(t), \mathrm{z}_i(t)\right)$ 为：

$$\begin{cases} x_i(t) = x_{0i}(t) - \dfrac{\mathrm{L}_i}{2} + x_{ci}(t) \\ y_i(t) = \mathrm{y}_{0i}(t) \qquad\qquad ，\ i = 1,2,3,\cdots,6 \\ z_i(t) = z_{0i}(t) - \dfrac{H_j}{2} + z_{ci}(t) \end{cases} \tag{8-6}$$

同问题一解法一样求得 t 时刻飞行器在飞行器坐标系下的质点坐标 $\left(c_{1x}(t), c_{1y}(t), c_{1z}(t)\right)$ 为：

$$\begin{cases} c_{1x}(t) = \dfrac{\sum_{i=1}^{N} m_i(t)\, x_c(t)}{\sum_{i=1}^{N} m_i(t)} \\ c_{1y}(t) = \dfrac{\sum_{i=1}^{N} m_i(t)\, y_c(t)}{\sum_{i=1}^{N} m_i(t)} \\ c_{1z}(t) = \dfrac{\sum_{i=1}^{N} m_i(t)\, z_c(t)}{\sum_{i=1}^{N} m_i(t)} \end{cases} \tag{8-7}$$

其中 $v_i(t)$ 为 t 时刻 i 号油箱内剩余的油体积，定义为：

$$\begin{cases} v_i(t) = v_i(t-1) - O_i(t), i \neq 2,5 \\ v_2(t) = v_2(t-1) - O_2(t) + O_1(t) \\ v_5(t) = v_5(t-1) - O_5(t) + O_6(t) \end{cases} \tag{8-8}$$

其中与副油箱连接的 2，5 号油箱内燃油体积不能超过其油箱的体积：

$$v_i(t) \leq v_i, i = 2,5 \tag{8-9}$$

其中 v_i 代表油箱 i 的体积。

把每个油箱看作质点，可以求解飞行器的质点 $c_1(t) = \left[c_{1x}(t), c_{1y}(t), c_{1z}(t)\right]$ 为：

$$\begin{cases} x_{ci}(t)=\dfrac{H_i}{2}=\dfrac{A_i(t)}{2L_i} \\ y_{ci}(t)=\dfrac{W_i}{2} \\ z_{ci}(t)=\dfrac{L_i}{2} \end{cases} ,i=1,2,3,\cdots,6 \tag{8-10}$$

其中$m_i(t)$为t时刻i号油箱油的重量。

$$m_i(t)=v_i(t)\cdot\rho \tag{8-11}$$

其中ρ为油的密度。

8.1.3 目标函数

为本次任务制定油箱供油策略，使得飞行器瞬时质心$\vec{c}_1(t)$与飞行器（不载油）质心$\vec{c}_0$的最大距离达到最小，即：

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

8.1.4 模型展示

根据以上变量、目标及约束条件的分析，建立飞机供油的动态模型：

其中$\vec{c}_i(t)=\left[c_i(1),c_i(2),...,c_i(t_{end})\right],i=1,2$。

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

$$
s.t.\begin{cases}
O_i(t) \le U_i; \sum_{i=2}^{5} O_i(t) \ge O(t) \\
O_i(t) \sim O_i(t+59) > 0, O_i(t) > 0 \\
D_i(t) = \begin{cases} 1, O_i(t) > 0 \\ 0, O_i(t) = 0 \end{cases} \\
1 \le \sum_{i=2}^{5} D_i(t) \le 2; 1 \le \sum_{i=1}^{6} D_i(t) \le 3 \\
v_i(t) \le v_i, i = 2,5; v_i(t) = v_i(t-1) - O_i(t) \\
x_{ci}(t) = \frac{H_i}{2} = \frac{A_i(t)}{2L_i}, y_{ci}(t) = \frac{W_i}{2}, z_{ci}(t) = \frac{L_i}{2}, i = 1,2,3,\cdots,6 \\
\begin{cases}
x_i(t) = x_{0i}(t) - \frac{\mathrm{L}_i(t)}{2} + x_{ci}(t) \\
y_i(t) = \mathrm{y}_{0i}(t) \qquad , \quad i = 1,2,3,\cdots,6 \\
z_i(t) = z_{0i}(t) - \frac{H_j(t)}{2} + z_{ci}(t)
\end{cases} \\
m_i(t) = v_i(t) \cdot \rho \\
c_{1x}(t) = \frac{\sum_{j=1}^{6} m_j(t) x_j(t)}{\sum_{k=1}^{6} m_j(t)}, c_{1y}(t) = \frac{\sum_{j=1}^{6} m_j(t) y_j(t)}{\sum_{k=1}^{6} m_j(t)}, c_{1z}(t) = \frac{\sum_{j=1}^{6} m_j(t) z_j(t)}{\sum_{k=1}^{6} m_j(t)}
\end{cases} \tag{8-12}
$$

8.2 基于不同角度的质心偏移补偿算法模型求解

8.2.1 模型求解

算法的实施步骤如下：

Step1：得到$t+59$时刻的角度$\alpha(t+59)$。

Step2：得到t时刻飞行器各个油桶的质量$v_i(t)$。

Step3：计算t时刻飞行器各个油桶的质量$m_i(t)$，以及飞行器对于空载时的质心的偏移向量$P_i(t)$。

Step4: 以 Step3 得到偏移向量$P_i(t)$作为“理想向量”，飞行器供油产生的质心变化向量对“理想向量”进行补偿使得两个向量的差值最小。

Step5: 计算每个单独的油桶，每减少 1 个单位的质量，对当前质心造成的偏转向量。

Step6: 生成所有组合的油桶方案，然后计算每个方案在投影在“理想向量”上的模的长度。

Step7: 选取投影在“理想向量”模长最长的方案对质心偏移进行部长，作为$(t+59)$分段的主方案。

Step8: 对分段的每个时刻进行搜索解空间，选取生成的新质心距离原点最近的油桶供油数据。

8.2.3 求解结果

通过编写 python 和 matlab 程序在约束条件下得到飞行器飞行过程中 **6 个油箱各自的供油速度曲线和 4 个主油箱的总供油速度曲线**(时间间隔为 1s)如下：

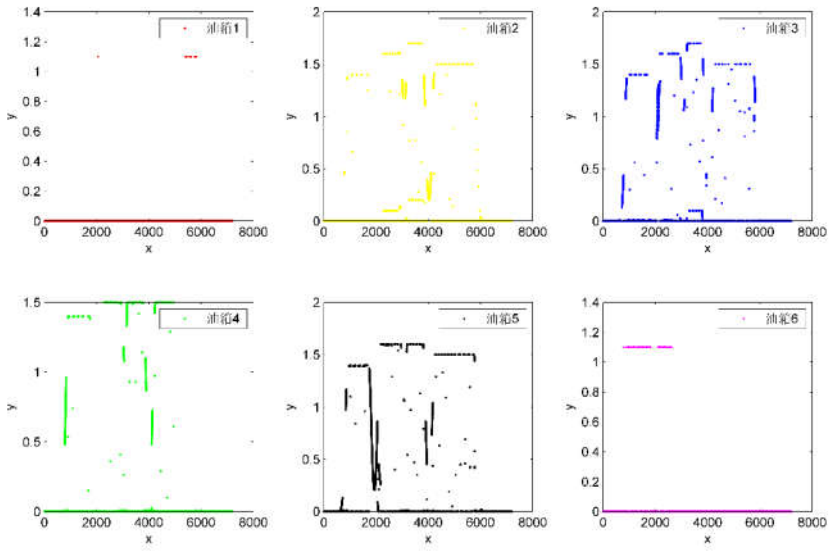


图 42 6 个油箱各自的供油速度曲线

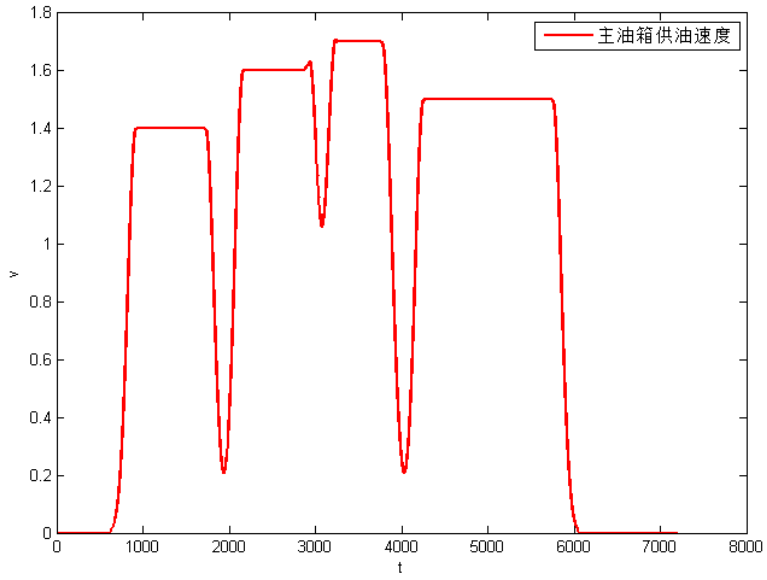


图 43 4 个主箱油总的总供油速度曲线

得到飞行器瞬时质心与理想质心的距离如图 44，**最大距离最小值为：0.14345552m，4 个主油箱的总供油量为 7035.545kg。**

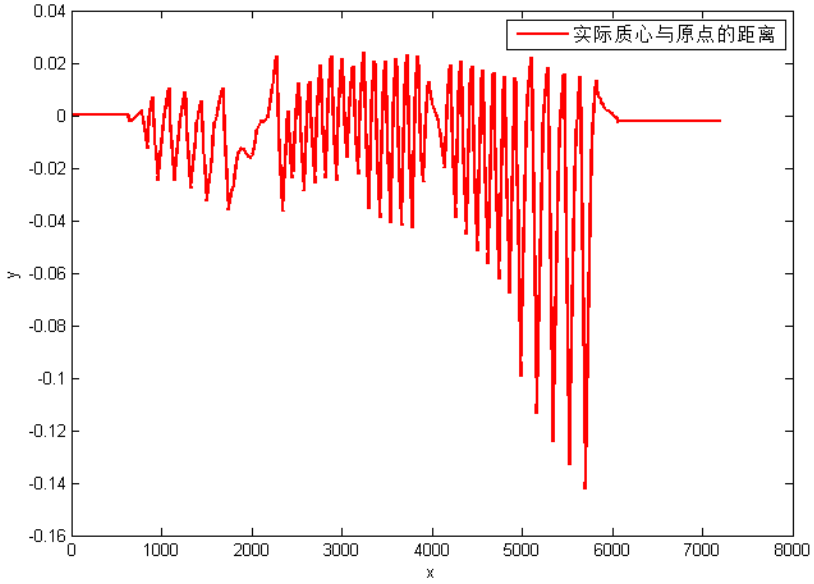


图 44 飞行器瞬时质心与原点的距离

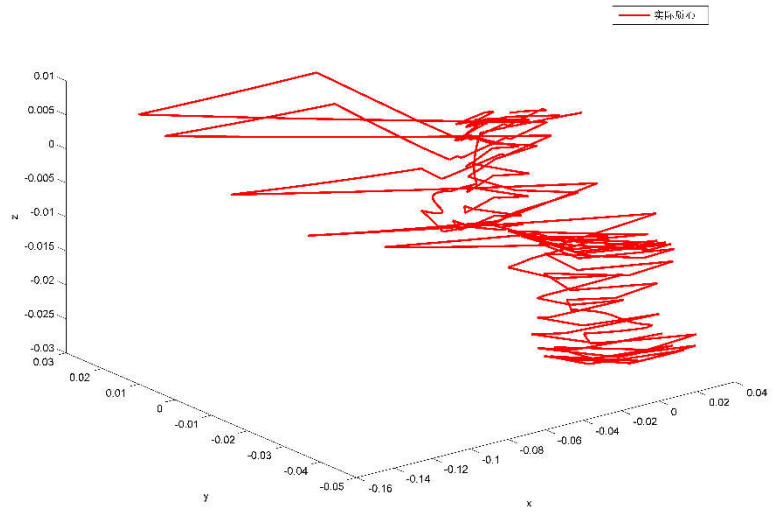


图 45 飞行器瞬时质心图

8.3 结果分析

我们假设油箱向发动机供油的过程中，供油和耗油相等，供油不产生多余的燃油，所以四个主油箱的总供油量等于飞行器耗油 7035.545kg。

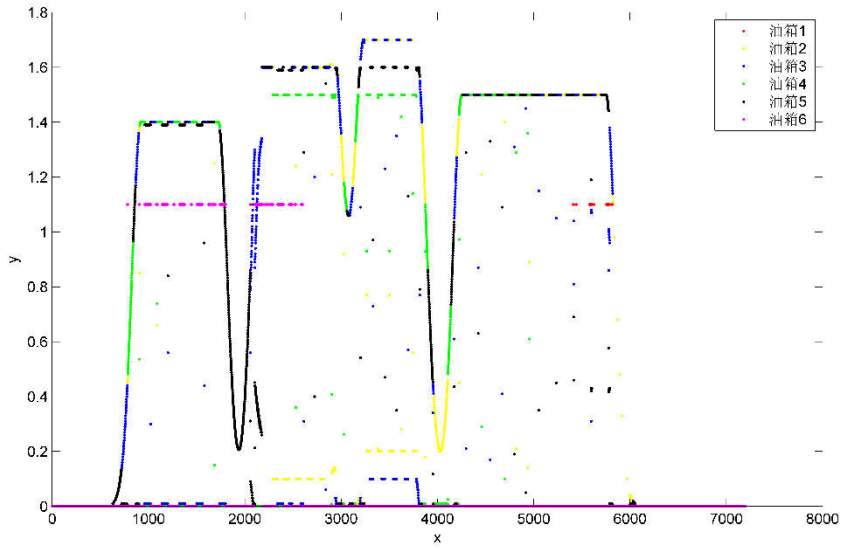


图 46 6 个油箱供油速度曲线

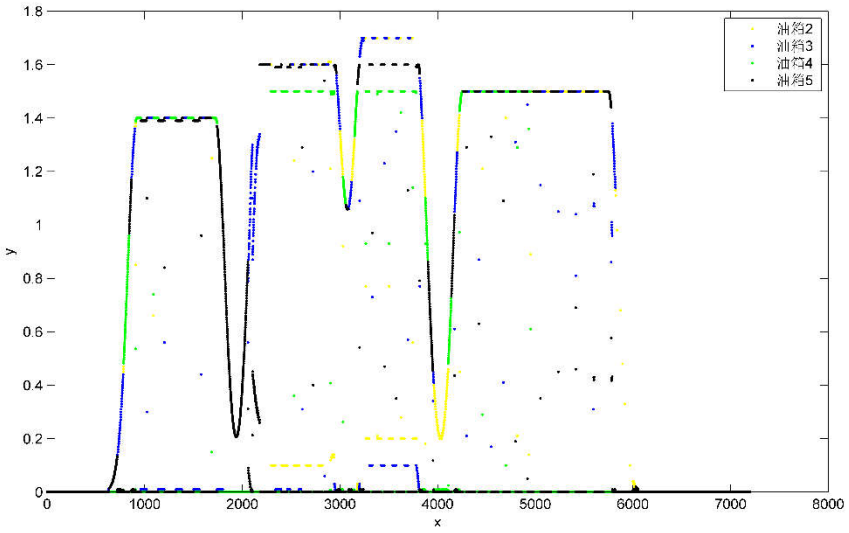


图 47 4 个油箱供油速度曲线

有 N 个时间点数据，需要 N 长度的数组存储发动机的耗油量，需要 N 长度的数组存储飞行器俯仰角数据。为了生成有效的方案，需要计算每个油桶投影在“理想向量”模的长度。时间复杂度 O(M*6*N),M 为每次为了计算投影向量的平均算法步骤。每个方案要进行平均 1000 次的最优值搜索，时间复杂度为 $O(1000N)$。空间复杂度 O(2*N), 总的时间复杂度是 O(M*6*N+1000N)，算法的实际运行时间为 390.2s。实验结果表明，在此情况下得到的瞬时质心与理想质心的最大值为 0.14345552m，在误差允许范围内，近似最优解，表明算法的有效性。

九、 模型评价

9.1 模型的优点

1、 飞行器-燃油组合体质心模型中考虑到了油箱中燃油液体形状的所有可能，并求得在不同形状下的质心坐标。
2、 理想质心偏移补偿法只考虑了前后时刻的关系，求解速度快。

9.2 模型的缺点

1、 在求飞行器质心时把油箱中的燃油看作质点，在计算时会产生误差。
2、 在理想质心偏移补偿法中每一时刻的供油方案只考虑了前后两个时刻的理想质心偏移，没有考虑整体之间的相关性。
3、 粒子群算法中的参数较多，且对结果影响较大，需要多次实验进行确定。

十、 参考文献

[1] 陈铭年,雷治国,徐建全.液罐车坡道上停驻时液体质心坐标的计算——液罐车纵向稳定性计算分析研究之一[C].//中国汽车工程学会.中国汽车工程学会 2003 学术年会论文集(下册).2003:847-853.

[2] 金先龙, 张淑敏. 非满载液罐汽车坡道行驶时液体质心位置的计算及分析[J]. 专用汽车, 1990, 000(004):13-17.

[3] 常胜利. 多边形重心坐标的求法[J]. 高等数学研究, 2005, 8(002):21-23.

[4] 张利彪, 周春光, 马铭, et al. 基于粒子群算法求解多目标优化问题[J]. 计算机研究与发展, 2004(07):1286-1291.

[5] Kennedy J, Eberhart R. Particle swarm optimization[C]//Proceedings of ICNN'95-International Conference on Neural Networks. IEEE, 1995, 4: 1942-1948.

十一、 附录

主要代码如下

```
# main.py
# coding=utf-8
from solution import solution
import winsound
import xlwt
import time
def main1():
    s = solution()
    s.readForQuestion_one('C:/Users/lee/Desktop/华为杯/2020 年中国研究生数学建模竞赛赛题/2020 年 F 题/2020 年 F 题--飞行器质心平衡供油策略优化/附件 1-飞行器参数.xlsx', '飞行器结构参数')
    s.readForQuestion_one_question('C:/Users/lee/Desktop/华为杯/2020 年中国研究生数学建模竞赛赛题/2020 年 F 题/2020 年 F 题--飞行器质心平衡供油策略优化/附件 2-问题 1 数据.xlsx', '油箱供油曲线', '飞行器俯仰角')
    s.write_one('C:/Users/lee/Desktop/question_one.xls', '飞行器坐标')

def main2():
    s = solution()
    s.readForQuestion_one('C:/Users/lee/Desktop/华为杯/2020 年中国研究生数学建模竞赛赛题/2020 年 F 题/2020 年 F 题--飞行器质心平衡供油策略优化/附件 1-飞行器参数.xlsx', '飞行器结构参数')
    s.readForQuestion_two('C:/Users/lee/Desktop/华为杯/2020 年中国研究生数学建模竞赛赛题/2020 年 F 题/2020 年 F 题--飞行器质心平衡供油策略优化/附件 3-问题 2 数据.xlsx', '发动机耗油速度', '飞行器理想质心数据')
    s.write_two('C:/Users/lee/Desktop/question_two.xls', '飞行器质心距离', '供油', '飞行器质心')

def main3():
    s = solution()
    s.readForQuestion_one('C:/Users/lee/Desktop/华为杯/2020 年中国研究生数学建模竞赛赛题/2020 年 F 题/2020 年 F 题--飞行器质心平衡供油策略优化/附件 1-飞行器参数.xlsx', '飞行器结构参数')
    # 不知道为什么表格的名字还不一样
    s.readForQuestion_three('C:/Users/lee/Desktop/华为杯/2020 年中国研究生数学建模竞赛赛题/2020 年 F 题/2020 年 F 题--飞行器质心平衡供油策略优化/附件 4-问题 3 数据.xlsx', '发动机耗油数据', '飞行器理想质心')
```

```
    s.write_two('C:/Users/lee/Desktop/question_three.xls', '飞行器质心距离', '供油', '飞行器质心')

def main4():
    s = solution()
    s.readForQuestion_one('C:/Users/lee/Desktop/华为杯/2020 年中国研究生数学建模竞赛赛题/2020 年 F 题/2020 年 F 题--飞行器质心平衡供油策略优化/附件 1-飞行器参数.xlsx','飞行器结构参数')
    s.readForQuestion_four('C:/Users/lee/Desktop/华为杯/2020 年中国研究生数学建模竞赛赛题/2020年F题/2020年F题--飞行器质心平衡供油策略优化/附件5-问题4数据.xlsx', '发动机耗油数据', '飞行器俯仰角')
    s.write_two('C:/Users/lee/Desktop/question_four.xls', '飞行器质心距离', '供油', '飞行器质心')
if __name__ == "__main__":
    start = time.time()
    main1()
    end = time.time()
    print(end - start)
    # winsound.Beep(600, 2000)

    # utils.py
    #coding=utf-8
import math
'''
功能函数   计算液体的质心
@:param   三个点坐标值
'''
class point:
    def __init__(self,x,y,z=0):
        self.x = x
        self.y = y
        self.z = z
    def getX(self):
        return self.x
    def getY(self):
        return self.y
    def getZ(self):
        return self.z
    def setX(self, x):
        self.x = x
```

```
    def setY(self, y):
        self.y = y
    def setZ(self, z):
        self.z = z
    def norm(self):
        no = math.sqrt(math.pow(self.x, 2) + math.pow(self.y, 2) + math.pow(self.z, 2))
        self.x = self.x / no
        self.y = self.y / no
        self.z = self.z / no

class triangle:
    def __init__(self, a, b, c):
        self.a = a
        self.b = b
        self.c = c
        self.centroid_x = 0
        self.centroid_y = 0
        self.centroid_z = 0
        self.centroid = 0
        self.area = 0
        '''
        构造好三角形之后直接计算面积和质心
        '''
        self.get_centroid_of_triangle()
        self.get_area_of_triangle()
    def getA(self):
        return self.a
    def getB(self):
        return self.b
    def getC(self):
        return self.c
    def get_centroid_of_triangle(self):
        self.centroid_x = (self.a.getX() + self.b.getX() + self.c.getX()) / 3.0
        self.centroid_y = (self.a.getY() + self.b.getY() + self.c.getY()) / 3.0
        self.centroid_z = (self.a.getZ() + self.b.getZ() + self.c.getZ()) / 3.0
        self.centroid = point(self.centroid_x, self.centroid_y, self.centroid_z)
        return self.centroid
    def get_area_of_triangle(self):
        '''
        得到三角形的面积   使用海伦公式
```

```
        '''
        lenA = math.sqrt(math.pow(self.b.getX() - self.c.getX(), 2) + math.pow(self.b.getY()
- self.c.getY(), 2) + math.pow(self.b.getZ() - self.c.getZ(), 2))
        lenB = math.sqrt(math.pow(self.a.getX() - self.c.getX(), 2) + math.pow(self.a.getY()
- self.c.getY(), 2) + math.pow(self.a.getZ() - self.c.getZ(), 2))
        lenC = math.sqrt(math.pow(self.b.getX() - self.a.getX(), 2) + math.pow(self.b.getY()
- self.a.getY(), 2) + math.pow(self.b.getZ() - self.a.getZ(), 2))
        self.area = 1.0 / 4.0 * math.sqrt((lenA + lenB + lenC) * (lenA + lenB - lenC) * (lenA
+ lenC - lenB) * (lenB + lenC - lenA))
        return self.area
    def get_centroid_x(self):
        '''
            一定要求解完质心坐标再使用
        '''
        return self.centroid_x
    def get_centroid_y(self):
        return self.centroid_y
    def get_centroid_z(self):
        return self.centroid_z
    def get_area(self):
        return self.area
    def get_centroid(self):
        return self.centroid

'''
功能函数   求多边形的质心
@:param v  是多个点的坐标  list point  类型
参考链接  https://www.cnblogs.com/ningmouming/p/9803783.html
'''
def centroid_of_convex_polygon(v):
    if(len(v) < 3):
        return -1
    if(len(v) == 3):
        t = triangle(v[0], v[1], v[2])
        return t.get_centroid_of_triangle()
    sum_Area = 0 #  用面积替代质量?
    #  多个三角形的类对象
    tmp = [] # tmp  似乎没有用
    centroid = point(0,0,0)
    for i in range(len(v) - 2):
```

```
            t = triangle(v[0], v[i+1], v[i+2])
            tmp.append(t)
            sum_Area += t.get_area()
            centroid.x += t.get_centroid_x() * t.get_area()
            centroid.y += t.get_centroid_y() * t.get_area()
            centroid.z += t.get_centroid_z() * t.get_area()
    if(sum_Area == 0):
            return centroid
    centroid.x = centroid.x / sum_Area
    centroid.y = centroid.y / sum_Area
    centroid.z = centroid.z / sum_Area
    return centroid

'''
求多个质心坐标的到一个质心坐标  v point  质心坐标 list    t  质量    返回 point
'''
def centroid_of_centorids(v, t):
    # 多个质心
    tmp = []
    centroid = point(0,0,0)
    sumWeight = 0
    for i in range(len(v)):
            sumWeight += t[i]
            centroid.x += t[i] * v[i].getX()
            centroid.y += t[i] * v[i].getY()
            centroid.z += t[i] * v[i].getZ()
    centroid.x = centroid.x / sumWeight
    centroid.y = centroid.y / sumWeight
    centroid.z = centroid.z / sumWeight
    return centroid

'''
体积转质量
'''
def volume2weight(v, roh):
    '''
    :param v: 体积值
    :param roh: 密度值
    :return: 质量值
    '''
```

```
    return roh * v

'''
质量转体积
'''
def weight2volume(m, roh):
    '''
    :param m: 质量
    :param roh: 密度
    :return: 体积值
    '''
    return m / roh

'''
多边形重心 采用 周同学的 list
'''
def centroid_of_convex_polygon_list(para):
    v = []
    for i in range(len(para)):
        v.append(point(para[i][0], para[i][1], 0))

    if(len(v) < 3):
        return -1
    if(len(v) == 3):
        t = triangle(v[0], v[1], v[2])
        t = t.get_centroid_of_triangle()
        centroid_list = []
        centroid_list.append(t.x)
        centroid_list.append(t.y)
        return centroid_list
    sum_Area = 0 # 用面积替代质量?
    # 多个三角形的类对象
    tmp = [] # tmp 似乎没有用
    centroid = point(0,0,0)
    for i in range(len(v) - 2):
        t = triangle(v[i], v[i+1], v[i+2])
        tmp.append(t)
        sum_Area += t.get_area()
        centroid.x += t.get_centroid_x() * t.get_area()
        centroid.y += t.get_centroid_y() * t.get_area()
```

```
            centroid.z += t.get_centroid_z() * t.get_area()
        centroid.x = centroid.x / sum_Area
        centroid.y = centroid.y / sum_Area
        centroid.z = centroid.z / sum_Area
        centroid_list = []
        centroid_list.append(centroid.x)
        centroid_list.append(centroid.y)
        return centroid_list

'''
根据得到的  平面的重心  计算 三维的中心
'''
def twoDim2threeDim(v, oilPoint, lwh):
        '''
        :param v: list  输入的  x z list
                  oilPoint: list (油桶的中心坐标)
                  lwh:  油桶的长宽高
        :return:  以飞机质心为原点的坐标系
        '''
        # 现在假设得到的坐标是（x_c, z_c）
        length = lwh[0]
        width = lwh[1]
        height = lwh[2]
        v_local = [v[0] - 0.5 * length + oilPoint[0], oilPoint[1], v[1] - 0.5 * height + oilPoint[2]]
        return v_local


def threeDim2threeDim(v, oilPoint, lwh):

        # 现在假设得到的坐标是（x_c, z_c）
        length = lwh[0]
        width = lwh[1]
        height = lwh[2]
        # v_local = [v[0] - 0.5 * length + oilPoint[0], oilPoint[1], v[1] - 0.5 * height +oilPoint[2]]
        v_local = [oilPoint[0] - 0.5 * length + v.getX(), oilPoint[1], v.getZ() + oilPoint[2] - 0.5 *
height]
        return v_local

def cal_included_angle(a, b):
        '''
```

```
    余弦定理
    :param a:   list [x, y, z]
    :param b: == a
    :return:   angle
    '''
    cos_theta = (a[0] * b[0] + a[1] * b[1] + a[2] * b[2]) / (math.sqrt(math.pow(a[0], 2) +
math.pow(a[1], 2) + math.pow(a[2], 2)) *

math.sqrt(math.pow(b[0], 2) + math.pow(b[1], 2) + math.pow(b[2], 2)))
    theta = math.degrees(math.acos(cos_theta))
    return theta

# 组合字符串代码
def perm(s=''):
    # 这里是递归函数的出口，为什么呢，因为这里表示：一个长度为 1 的字符串，它的排列组合就是它自己。
    if len(s) <= 1:
        return [s]
    sl = []   # 保存字符串的所有可能排列组合
    for i in range(len(s)):   # 这个循环，对应 解题思路 1）确定字符串的第一个字母是谁，有 n 种可能（n 为字符串 s 的长度
        for j in perm(s[0:i] + s[i + 1:]):   # 这个循环，对应 解题思路 2）进入递归，s[0:i]+s[i+1:]的意思就是把 s 中的 s[i]给去掉
            sl.append(s[i] + j)   # 对应 解题思路 2）问题就从“返回字符串中的字母排列组合” **变成了** “返回 第一个字母+除去第一个字母外的字符串的排列组合”
    return sl

def perm_main(s=''):
    perm_nums = perm(s)   # 有可能存在字母相同的情况
    no_repeat_nums = list(set(perm_nums))   # 去重，挺牛的，这个代码
    # print('perm_nums', len(perm_nums), perm_nums)
    # print('no_repeat_nums', len(no_repeat_nums), no_repeat_nums)
    return no_repeat_nums

def distance(a, b):
    return math.sqrt(math.pow(a.getX() - b.getX(), 2) + math.pow(a.getY() - b.getY(), 2) +
math.pow(a.getZ() - b.getZ(), 2))

if __name__=='__main__':
    pass
```

```
'''
    校验质心坐标没有问题
    v = []
    v.append(point(1,1,1))
    v.append(point(0,0,0))
    v.append(point(1,1,0))
    v.append(point(0,0,1))
    v.append(point(0,1,0))
    v.append(point(1,0,0))
    v.append(point(0,1,1))
    v.append(point(1,0,1))
    w = []
    for i in range(8):
        w.append(1)
    t = centroid_of_centorids(v, w)
    print(t.getX(),t.getY(), t.getZ())
'''
    # oildrum.py
    #coding=utf-8
from utils import centroid_of_centorids,point
class oildrum:
    def __init__(self, x, y, z, weight, length, width, height, name):
        self.ari_x = x #  相对于  飞行器的坐标
        self.air_y = y
        self.air_z = z
        self.weight = weight
        self.length = length
        self.width = width
        self.height = height
        self.name = name
        self.theta = 0
        self.power_weight = 0
        self.state = -1 # -1  状态表示不在工作
        self.value = [] #  供油值
    def set_other_para(self, U, roh):
        self.U = U    #  供油上限  kg/s
        self.roh = roh    #  液体的密度  kg/m^2
        self.maxWeight = (self.length * self.width * self.height) * self.roh
        self.volume = self.weight / self.roh    #  带有的液体体积
```

```
def get_lwh(self):
    return [self.length, self.width, self.height]
def get_lhw(self):
    return [self.length, self.height, self.width]
def get_xyz(self):
    return [self.ari_x, self.air_y, self.air_z]

def get_U(self):
    return self.U
def get_state(self):
    return self.state
def turn_on(self):
    if self.state == -1:
        self.state = 0
    else:
        self.state += 1
def turn_off(self):
    if self.state != -1:
        self.state = -1

def set_theta(self, theta):
    self.theta = theta

def consume_one_second(self, con):
    '''
    消耗的体积 TODO -1 -2 判断
    :param con: kg
    :return:
    '''
    if con > self.U:
        # print('[ERROR] con bigger than self.U', con, self.U)
        return -1
    if self.weight - con < 0:
        #print('[ERROR] more than enpty()', self.name)
        return -2

    self.weight -= con
    self.volume = self.weight / self.roh
    return 1
def check_can_consume(self, con):
```

```
            if con > self.U:
                return False
            if self.weight - con < 0:
                return False
            return True
        def check_can_add(self, other_con):
            if other_con + self.weight > self.maxWeight:
                return False
            return True

        def set_weight(self, weight):
            self.weight = weight
            self.volume = self.weight / self.roh

        def get_weight(self):
            return self.weight

        def add_one_second(self, other_con):
            '''
            :param other_con: 别的油桶的消耗 kg
            :return:
            '''
            if other_con + self.weight > self.maxWeight:
                # print('[ERROR] other_con bigger than self.maxWeight', other_con,
self.weight, self.maxWeight)
                return -1
            self.weight += other_con
            self.volume = self.weight / self.roh
            return 1
        def cal_one_kg_pw(self):
            # 计算减少单位质量造成的飞行器质心偏移
            v = []
            v.append(point(self.ari_x, self.air_y, self.air_z))
            v.append(point(0,0,0))
            w = []
            w.append(100) # 表示这个点增加 100 的向量的反方向
            w.append(3000)
            p = centroid_of_centorids(v, w)
            p = point(-p.getX(), -p.getY(), -p.getZ())
            # p.norm()
```

```
            self.power_weight = p
            # print(self.name, p.getX(), p.getY(), p.getZ())
        def get_pw(self):
            return self.power_weight

        def cal_one_kg_pw_cur_weight_point(self, p):

            pass

        # core_question.py
        #coding=utf-8
import math
import utils
class centroid_t_to_f:
        def __init__(self, length, width, height, volume, theta):
            self.length = length
            self.width = width
            self.height = height
            self.volume = volume
            self.theta = theta
            self.area = self.area()
            self.a0 = self.extremeAngle() #  计算临界角度

            self.extreme_small_volume_small_angle_left_height = 0
            self.extreme_small_volume_small_angle_area                                    =
self.cal_extreme_small_volume_small_angle() #  计算小体积小角度的临界面积

            self.extreme_big_volume_small_angle_right_height = 0
            self.extreme_big_volume_small_angle_area                                      =
self.cal_extreme_big_volume_small_angle() #  计算大体积小角度的临界面积

            self.extreme_small_volume_big_angle_bottom_point = 0
            self.extreme_small_volume_big_angle_area                                      =
self.cal_extreme_small_volume_big_angle()   #  计算小体积大角度的临界面积


            self.extreme_big_volume_big_angle_area                                        =
self.cal_extreme_big_volume_big_angle() # j 计算大体积大角度的临界面积
        def area(self):
            self.area = self.volume / self.width
```

```
        return self.area
    def extremeAngle(self):
        self.a0 = math.degrees(math.atan2(self.height,self.length))
        return self.a0
    def cal_extreme_small_volume_small_angle(self):
        # 小体积小角度极限面积
        left_height = self.length * math.tan(math.radians(self.theta))
        self.extreme_small_volume_small_angle_left_height = left_height
        return left_height * self.length * 0.5
    def cal_extreme_big_volume_small_angle(self):
        # 大体积 小角度的极限面积
        rect_area = self.height * self.length
        right_height = self.length * math.tan(math.radians(self.theta))
        self.extreme_big_volume_small_angle_right_height = right_height
        return rect_area - right_height * self.length * 0.5
    def cal_extreme_small_volume_big_angle(self):
        # 计算小体积大角度的极限值
        left_bottom = self.height * math.tan(math.radians(90 - self.theta))
        self.extreme_small_volume_big_angle_bottom_point = left_bottom
        return left_bottom * self.height * 0.5
    def cal_extreme_big_volume_big_angle(self):
        # 计算大体积大角度的极限值
        rect_area = self.height * self.length
        right_upper = self.height * math.tan(math.radians(90 - self.theta))
        return rect_area - right_upper * self.height * 0.5

    def solve(self):
        if(self.theta == 0):
            height_left = self.area / self.length
            v = []
            v.append(utils.point(0, 0, 0))
            v.append(utils.point(self.length, 0, 0))
            v.append(utils.point(self.length, 0, height_left))
            v.append(utils.point(0, 0, height_left))
            return utils.centroid_of_convex_polygon(v)
        if(self.theta < self.a0):
            # 此时分 3 中情况讨论
            if(self.area <= self.extreme_small_volume_small_angle_area):
                # 此时三角形的两个点落在直角的两条边上   x * x * tan(theta) = 2 * s
                bottom = math.sqrt(2 * self.area    / math.tan(math.radians(self.theta)))
```

```
                left = bottom * math.tan(math.radians(self.theta))
                v = []
                v.append(utils.point(0,0,0))
                v.append(utils.point(0,0,left))
                v.append(utils.point(bottom,0,0))
                return utils.centroid_of_convex_polygon(v)
            elif self.area > self.extreme_small_volume_small_angle_area and self.area <=
self.extreme_big_volume_small_angle_area:
                # 此时两个点落在两条高线上 一共四个点
                area_s1 = self.area - self.extreme_small_volume_small_angle_area
                delta_left = area_s1 / self.length
                v = []
                v.append(utils.point(0,                                                    0,
self.extreme_small_volume_small_angle_left_height + delta_left))
                v.append(utils.point(self.length,0,delta_left))
                v.append(utils.point(self.length, 0 , 0))
                v.append(utils.point(0,0,0))
                return utils.centroid_of_convex_polygon(v)
            elif self.area > self.extreme_big_volume_small_angle_area:
                rect_area = self.length * self.height
                delta_s = rect_area - self.area
                # x * x * tan(theta) = 2 * delta_s
                upper = math.sqrt(2 * delta_s / math.tan(math.radians(self.theta)))
                right = upper * math.tan(math.radians(self.theta))
                v = []
                v.append(utils.point(self.length - upper,0,self.height))
                v.append(utils.point(0,0,self.height))
                v.append(utils.point(0,0,0))
                v.append(utils.point(self.length, 0, 0))
                v.append(utils.point(self.length, 0, self.height - right))
                return utils.centroid_of_convex_polygon(v)
            else:
                print('ERROR')
        else:
            # 角度 >= a0
            # 此时分 3 种情况讨论
            if self.area < self.extreme_small_volume_big_angle_area:
                # 此时三角形的两个点落在直角的两条边上
                bottom = math.sqrt(2 * self.area / math.tan(math.radians(self.theta)))
                left = bottom * math.tan(math.radians(self.theta))
```

```
                    v = []
                    v.append(utils.point(0, 0, 0))
                    v.append(utils.point(0, 0, left))
                    v.append(utils.point(bottom, 0, 0))
                    return utils.centroid_of_convex_polygon(v)
                elif self.area >= self.extreme_small_volume_big_angle_area and self.area < self.extreme_big_volume_big_angle_area:
                    # 此时两个点落在两条高线上 一共四个点
                    area_s1 = self.area - self.extreme_small_volume_small_angle_area
                    delta_bottom = area_s1 / self.height
                    v = []
                    v.append(utils.point(self.extreme_small_volume_big_angle_bottom_point + delta_bottom, 0, 0))
                    v.append(utils.point(delta_bottom, 0, self.height))
                    v.append(utils.point(0, 0, self.height))
                    v.append(utils.point(0, 0, 0))
                    return utils.centroid_of_convex_polygon(v)
                elif self.area > self.extreme_big_volume_big_angle_area:
                    rect_area = self.length * self.height
                    delta_s = rect_area - self.area
                    # x * x * tan(theta) = 2 * delta_s
                    upper = math.sqrt(2 * delta_s / math.tan(math.radians(self.theta)))
                    right = upper * math.tan(math.radians(self.theta))
                    v = []
                    v.append(utils.point(self.length - upper, 0, self.height))
                    v.append(utils.point(0, 0, self.height))
                    v.append(utils.point(0, 0, 0))
                    v.append(utils.point(self.length, 0, 0))
                    v.append(utils.point(self.length, 0, self.height - right))
                    return utils.centroid_of_convex_polygon(v)
                else:
                    print('ERROR', self.area)

if __name__ == '__main__':
    t = centroid_t_to_f(10, 10, 10, 125, 10)
    tmp = t.solve()
    print(tmp.getX(), tmp.getY(), tmp.getZ())
```