飞行动力学及控制原理

大作业

班级：09011804

姓名：赵敏琨

学号：2018302068

成绩：

目录

一、符号说明

表格 1 符号说明

符号	说明	量纲
m	全机质量	kg
S_w	参考面积	m^2
c_A	参考弦长	m
b	参考展长	m
I_x 等	关于各轴的转动惯量	$kg \cdot m^2$
g	重力加速度 $g = 9.80665$ （分析过程中假设不随高度变化）	m / s^2
h 或 H	飞行高度	m
V	真空速	m / s
α	迎角 （空速矢量投影在机体纵轴下方时为正）	度
β	侧滑角 （空速矢量投影在机体纵轴右侧时为正）	度
p	滚转角速度（右滚转为正）	rad / s
q	俯仰角速度（抬头为正）	rad / s
r	偏航角速度（机头右偏为正）	rad / s
ϕ	滚转角（右机翼下，左机翼上为正）	度
θ	俯仰角（机头向上为正）	度
ψ	偏航角（右偏航为正）	度
u	速度在机体系纵轴的分量	m / s
v	速度在机体系横轴的分量	m / s
w	速度在机体系立轴的分量	m / s
C_L	升力系数	无量纲

C_{L_0}	零迎角升力系数	无量纲
C_L^{α}	升力系数随迎角变化的导数	迎角/度
$C_L^{\delta_e}$	升力升降舵操纵导数	平尾偏度/度
C_D	阻力系数	无量纲
C_{D_0}	零迎角阻力系数	无量纲
C_D^{α}	阻力系数随迎角变化的导数	迎角/度
$C_D^{\delta_e}$	阻力升降舵操纵导数	平尾偏度/度
C_m	俯仰力矩系数	无量纲
C_{m_0}	零升俯仰力矩系数	无量纲
C_m^{α}	俯仰力矩系数随迎角变化的导数	迎角/度
$C_m^{\delta_e}$	俯仰力矩升降舵操纵导数	平尾偏度/度
$C_m^{\bar{q}}$	俯仰角速率产生的纵向阻尼倒数	s/rad
$C_m^{\dot{\alpha}}$	洗流时差导数	s/rad
C_Y	侧力系数	无量纲
C_Y^{β}	侧力系数随侧滑角变化的导数	侧滑角/度
$C_Y^{\delta_r}$	侧力方向舵操纵导数	方向舵偏度/度
C_l	滚转力矩系数	无量纲
C_l^{β}	滚转力矩系数随侧滑角变化的导数	侧滑角/度
$C_l^{\delta_a}$	滚转力矩副翼操纵导数	副翼偏度/度
$C_l^{\delta_r}$	滚转力矩方向舵操纵交叉导数	方向舵偏度/度
C_l^{p}	滚转角速率产生的滚转阻尼导数	s/rad
C_l^{r}	偏航角速率产生的交叉动态导数	s/rad

C_n	偏航力矩系数	无量纲
C_n^{β}	偏航力矩系数随侧滑角变化的导数	侧滑角/度
$C_n^{\delta_a}$	偏航力矩副翼操纵交叉导数	副翼偏度/度
$C_n^{\delta_r}$	偏航力矩方向舵操纵导数	方向舵偏度/度
C_n^{p}	滚转角速率产生的交叉动态导数	s/rad
C_n^{r}	偏航角速率产生的偏航阻尼导数	s/rad
Q	动压	Pa
L	升力	N
D	阻力	N
Y	侧力	N
M	俯仰力矩	$N \cdot m$
L_M	滚转力矩	$N \cdot m$
N	偏航力矩	$N \cdot m$

二、解算步骤

1. 计算气动关键参数

首先从AOA_CL_CD_Cm.txt文件中加载全机系数表（CL、CD、Cm与AOA的关系表），设置插值间隔d = 0.01，范围x = -15:d:25，用MATLAB中的三次样条插值函数spline对三组数据进行插值计算，插值结果为yL、yD、ym，得到连续的函数关系曲线。计算完成后，绘制下图：

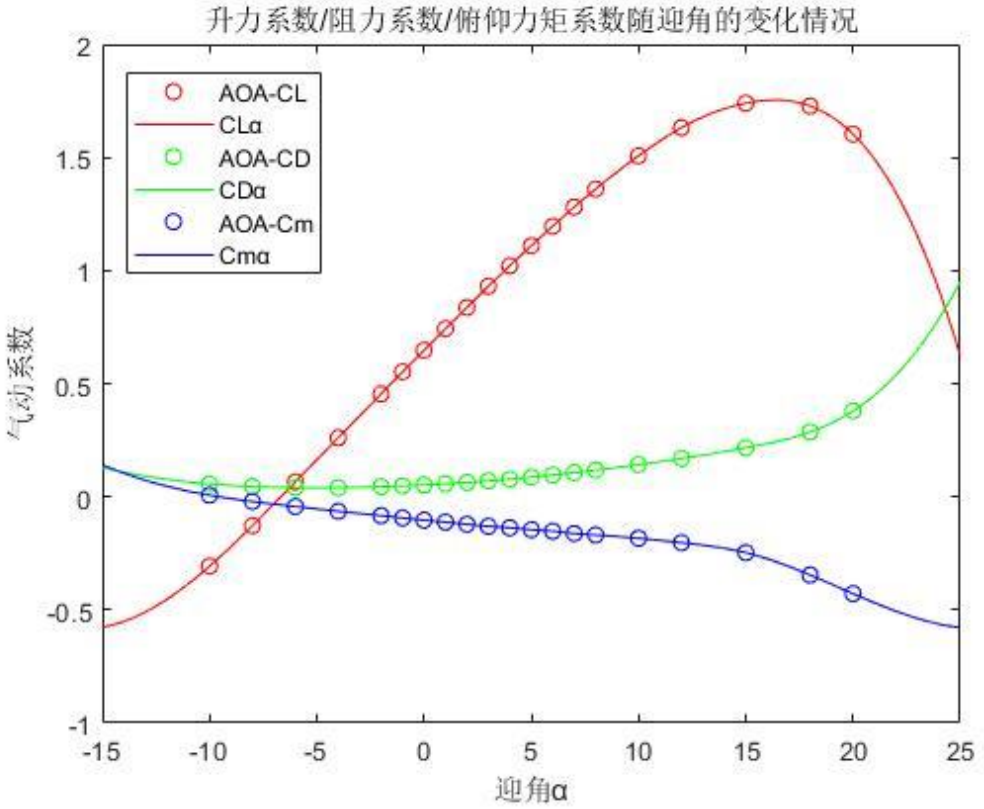


图 1 升力系数/阻力系数/俯仰力矩系数随迎角的变化关系曲线

之后，求关键参数：

- 查找yL=0时对应的x为零升迎角α_0；
- 查找yL最大时对应的x为临界迎角α_C；
- 查找x=0时对应的yL为零迎角升力系数C_{L_0}；
- x=0时对应的yD为零迎角阻力系数C_{D_0}；
- 查找yL=0时对应的ym为零升力矩系数C_{m_0}。

由图可知，在$\alpha_0 \sim (\alpha_C\text{-}5)$°范围内，三条曲线近似线性变化，在这个范围内用diff取差mean求均值的方法求气动静导数，得到C_L^α、C_D^α、C_m^α。再根据PDF数据文件中给出的其他系数（注：其中的升降舵、方向舵效为单侧，使用时须×2），得到如下表达式：

升力系数：

$$C_L = C_{L_0} + C_L^{\alpha}\alpha + C_L^{\delta_e}\delta_e$$

$$C_L = 0.647910 + 0.088485\alpha + 0.00656\delta_e$$

阻力系数：

$$C_D = C_{D_0} + C_D^{\alpha}\alpha + C_D^{\delta_e}\delta_e$$

$$C_D = 0.051832 + 0.006587\alpha + 0.00036\delta_e$$

俯仰力矩系数：

$$C_m = C_{m_0} + C_m^{\alpha}\alpha + C_m^{\delta_e}\delta_e + C_m^{\bar{q}}\bar{q} + C_m^{\bar{\dot{\alpha}}}\bar{\dot{\alpha}}$$

$$C_m = -0.036061 - 0.008902\alpha - 0.01684\delta_e - 7.58\bar{q} - 1.64\bar{\dot{\alpha}}$$

侧力系数：

$$C_Y = C_Y^{\beta}\beta + C_Y^{\delta_r}\delta_r$$

$$C_Y = -0.00668\beta + 0.00484\delta_r$$

滚转力矩系数：

$$C_l = C_l^{\beta}\beta + C_l^{\delta_a}\delta_a + C_l^{\delta_r}\delta_r + C_l^{p}\bar{p} + C_l^{r}\bar{r}$$

$$C_l = -0.00072\beta - 0.00393\delta_a - 0.00008\delta_r - 0.62\bar{p} - 0.01\bar{r}$$

偏航力矩系数：

$$C_n = C_n^{\beta}\beta + C_n^{\delta_a}\delta_a + C_n^{\delta_r}\delta_r + C_n^{p}\bar{p} + C_n^{r}\bar{r}$$

$$C_n = 0.00104\beta + 0.00034\delta_a - 0.00122\delta_r + 0.004\bar{p} - 0.04\bar{r}$$

然后根据升力、阻力、侧力，俯仰力矩、滚转力矩、偏航力矩的表达式计算各值：

$$Q = \frac{1}{2}\rho V^2$$

$$\begin{cases} L = C_L Q S_w \\ D = C_D Q S_w \\ Y = C_Y Q S_w \\ M = C_m Q S_w c_A \\ L_M = C_l Q S_w b \\ N = C_n Q S_w b \end{cases}$$

2. 配平计算

飞机机体系中质心的运动方程为：

$$\begin{cases} m(\dot{u}+wq-vr)=T-mg\sin\theta-D\cos\alpha\cos\beta+L\sin\alpha \\ m(\dot{v}+ur-wp)=Y+mg\sin\phi\cos\theta-D\sin\beta \\ m(\dot{w}+vp-uq)=mg\cos\phi\cos\theta-D\sin\alpha\cos\beta-L\cos\alpha \end{cases}$$

飞机力矩方程为：

$$\begin{cases} L_M=\dot{p}I_x-\dot{r}I_{xz}+qr(I_z-I_y)-pqI_{xz} \\ M=\dot{q}I_y+pr(I_x-I_z)+(p^2-r^2)I_{xz} \\ N=\dot{r}I_z-\dot{p}I_{xz}+pq(I_y-I_x)+qrI_{xz} \end{cases}$$

配平时，有

$$\begin{cases} F_x=T-mg\sin\theta-D\cos\alpha\cos\beta+L\sin\alpha=0 \\ F_y=Y+mg\sin\phi\cos\theta-D\sin\beta=0 \\ F_z=mg\cos\phi\cos\theta-D\sin\alpha\cos\beta-L\cos\alpha=0 \end{cases}$$

$$\begin{cases} L_M=\dot{p}I_x-\dot{r}I_{xz}+qr(I_z-I_y)-pqI_{xz}=0 \\ M=\dot{q}I_y+pr(I_x-I_z)+(p^2-r^2)I_{xz}=0 \\ N=\dot{r}I_z-\dot{p}I_{xz}+pq(I_y-I_x)+qrI_{xz}=0 \end{cases}$$

又因定常直线平飞时配平条件为

$$\dot{H}=0 \quad \dot{V}=0 \quad \phi=0 \quad \theta=\alpha \quad \beta=0$$

则配平时飞机运动方程为

$$\begin{cases} F_x=T-mg\sin\alpha-D\cos\alpha+L\sin\alpha=0 \\ F_y=Y=0 \\ F_z=mg\cos\alpha-D\sin\alpha-L\cos\alpha=0 \end{cases} \tag{1}$$

飞机角运动方程为：

$$\begin{cases} L_M=0 \\ M=0 \\ N=0 \end{cases} \tag{2}$$

配平目标是合外力、合外力矩均为 0，实质上是应用优化算法求解最小化问题。联立方程组(1)(2)，目标函数取为合外力与合外力矩的欧氏距离和：

$$\begin{aligned} TOBJ&=\sqrt{(\sqrt{F_x^2+F_y^2+F_z^2})^2+(\sqrt{L_M^2+M^2+N^2})^2} \\ &=\sqrt{F_x^2+F_y^2+F_z^2+L_M^2+M^2+N^2} \end{aligned}$$

同时，考虑到实际工程问题，需要满足以下约束：

$T>0$，推力大于0

$\alpha<\alpha_C$，迎角小于临界迎角

$|\delta_e|<\delta_{em}$，升降舵小于限制值

由于实际上进行纵向配平，故令方程组中与横侧向有关的状态量$\delta_r=\delta_a=\bar{p}=\bar{q}=\bar{r}=\dot{\alpha}=0$,由于0的约束过为严苛，算法中另外定义了约束容差，算法收敛情况较好时令其为 eps（MATLAB 中的极小量），局部收敛时适当增加容差，使得算法最终能够收敛到全局最优解。这部分条件相当于又附加了如下的约束：

$$|\delta_r|<eps,|\delta_a|<eps,$$
$$|\bar{p}|<eps,|\bar{q}|<eps,|\bar{r}|<eps,|\dot{\alpha}|<eps$$

所以，配平优化问题描述为：

$$\min TOBJ=\sqrt{F_x^2+F_y^2+F_z^2+L_M^2+M^2+N^2}$$
$$s.t.\begin{cases}T>0\\ \alpha<\alpha_C\\ |\delta_e|<\delta_{em}\\ |\delta_r|<eps\\ |\delta_a|<eps\\ |\bar{p}|<eps\\ |\bar{q}|<eps\\ |\bar{r}|<eps\\ |\dot{\alpha}|<eps\end{cases}$$

3. 配平算法简介

本报告采用了三种方法求解配平值，分别是带约束的粒子群优化算法，MATLAB 自带的 fmincon 优化函数，借助 Simulink 建模的 findop 方法。其中，粒子群算法为本人根据这学期另一门选修课（计算智能导论）学习的知识自行编写；fmincon 优化函数方法是由优化工具箱导出的代码改写而来；findop 方法附有直接导出的代码，步骤后文会详细说明。

(1) 带约束的粒子群优化算法（PSO）

粒子群优化（Particle Swarm Optimization, PSO）是基于鸟群社会行为（鸟群觅食）的模拟发展而来的一种随机优化技术，主要用于处理连续函数优化问题。

其主要思路是：让一群称为粒子的鸟在问题的搜索空间中飞翔，最优解被想象成食物所在的位置，而优化过程则看成是小鸟寻找食物的过程；在鸟群寻找食物的过程中，每一个个体的行为不但会受到其过去的经验和认知的影响，同时也会受到整体社会行为的影响；在 PSO 算法中，每一个粒子在搜索空间中各自有其方向和速度，并且根据自身过去的经验与群体行为进行概率搜索策略的调整。

其步骤如下：

1) 配平为最小化问题，**适应函数取为目标函数**，编写目标函数 Trans_Trim_Objective，由于配平要求合外力为 0，合外力矩为 0，取合外力与合外力矩的欧氏距离和作目标函数；
2) 注意到优化问题带有约束条件，所以需要用**带约束**的粒子群优化算法解决，**采用“惩罚值”的办法来处理约束**，约束条件写在算法主文件中；
3) **粒子群初始化：**

- **随机生成初始种群位置 pop_x(i,j)**：须满足位置限制 $-100 \le x_i \le 100, (i=1,2)$；
- **随机生成初始种群速度 pop_v(i,j)**：须满足速度限制 $-1 \le v_i \le 1, (i=1,2)$；
- **采用全局邻域拓扑（星状拓扑）**：每个粒子可以与其他所有粒子通信，所有其他粒子都是该粒子的近邻；
- **初始化个体历史最佳位置（个体极值点）pbest 和个体历史最佳适应度（个体极值）fitness_pbest**：随机初始化后不满足约束个体的 fitness_pbest 置为 $\frac{PV}{10^6}=10000$，这么做使得初始化种群时实际惩罚值较小，达到扩大搜索空间的目的；
- **初始化群体历史最佳位置（局部极值点）gbest 和群体历史最佳适应度（局部极值）fitness_gbest**：先在随机初始种群中比较一遍，得到随机初始种群的群体历史最佳位置；

4) **粒子群迭代更新：**

- **更新速度并对速度进行边界处理：**

速度更新公式为

$$v_i = c_1 \cdot \left(-\frac{1}{ger} \cdot iter + 1 \right) \cdot v_i + c_2 \cdot rand() \cdot (pbest_i - x_i) + c_3 \cdot rand() \cdot (gbest_i - x_i)$$

其中，c_2为自我学习因子，c_3为群体学习因子。c_1为惯性权重，*ger* 为最大迭代次数，*iter* 为当前迭代次数，相当于惯性权重因子叠加线性函数，使之随迭代次数增加而减小。因为初始阶段较大的惯性权重可以扩展算法在搜索空间中的探索范围；末尾阶段较小的惯性权重可以加强算法的局部搜索能力。

速度边界处理为

$$v_{ij} > V_{\max} \Rightarrow v_{ij} = V_{\max}$$
$$v_{ij} < V_{\min} \Rightarrow v_{ij} = V_{\min}$$

- **更新位置并对位置进行边界处理：**

位置更新公式为

$$x_i = x_i + v_i$$

位置边界处理为

$$x_{ij} > X_{\max} \Rightarrow x_{ij} = X_{\max}$$
$$x_{ij} < X_{\min} \Rightarrow x_{ij} = X_{\min}$$

- **进行约束条件判断并计算新种群各个个体的适应度：**

满足约束的个体适应度取为适应函数（目标函数）值，

不满足约束的个体适应度置为$\frac{PV}{1+1.05^{-iter+\frac{ger}{3}}}$，相当于进行适应值比例变换，迭代时原始惩罚值叠加类 Sigmoid 函数，使实际惩罚值随着代际增大而增大。因为迭代开始时，较小的惩罚值可以抑制竞争，扩大搜索范围；迭代后期时，较大的惩罚值可以鼓励竞争，加快收敛速度。

- **比较新适应度与个体历史最佳适应度**：最小化问题，如果更小则更新个体历史最佳适应度；
- **比较个体历史最佳适应度与种群历史最佳适应度**：最小化问题，如果更小则更新种群历史最佳适应度；
- **达到最大迭代次数，停止迭代**；

5) **迭代结果输出：**

- **输出收敛过程情况图**；
- **输出目标函数最优值与最优值点**；

算法流程图如下：

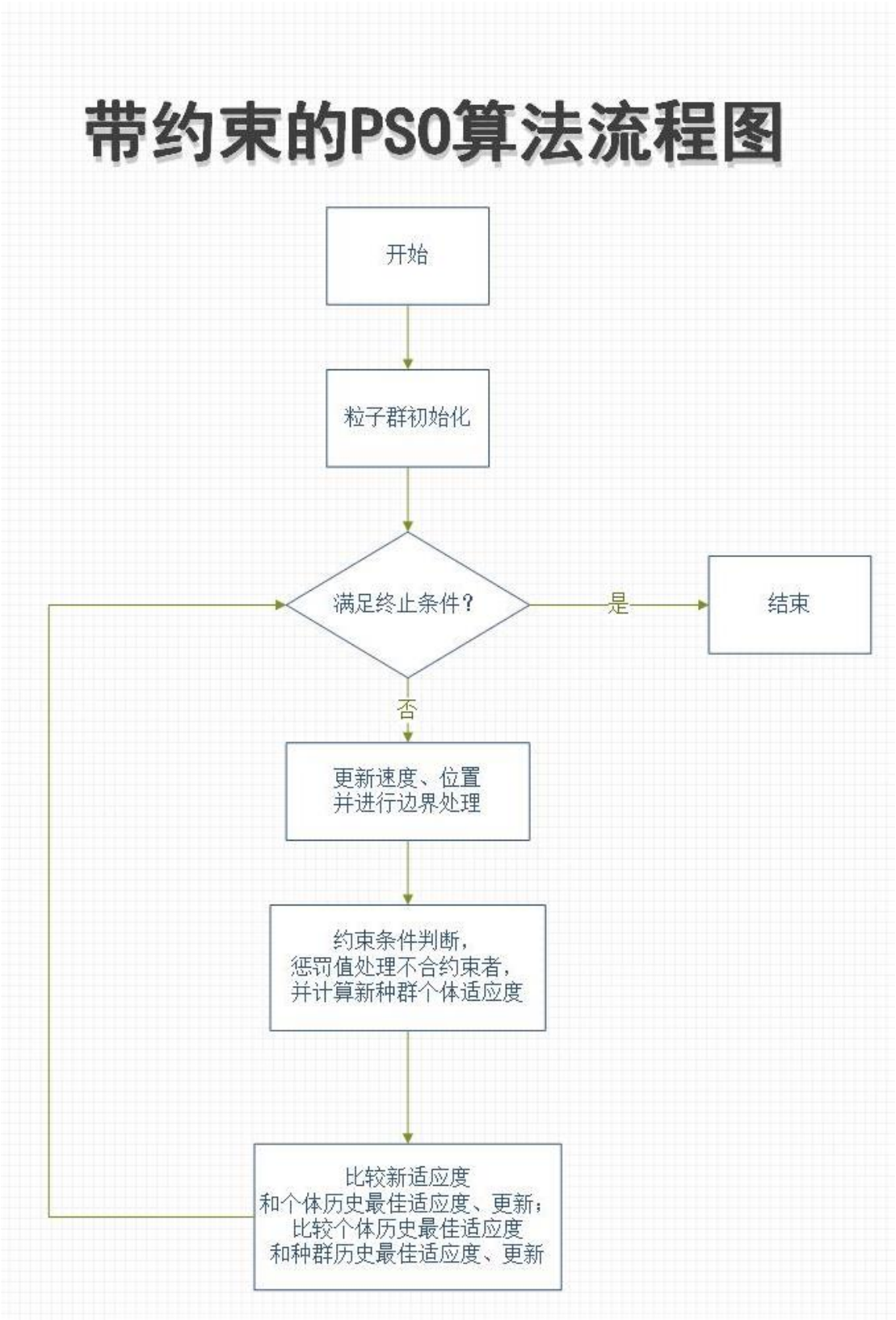


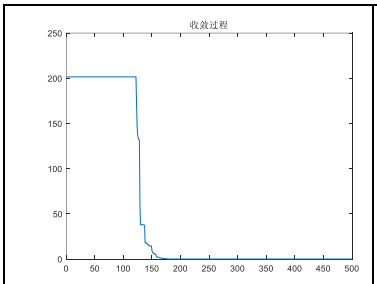
图 2 带约束的 PSO 算法流程图

本方法得到的配平状态点如下：

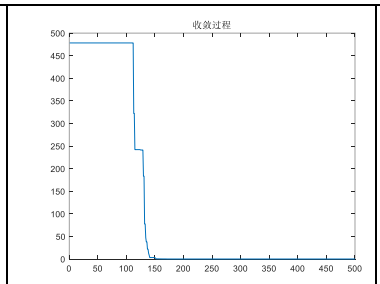
表格 2 PSO 算法得到的配平值

序号	高度 h(m)	速度 V(m/s)	推力值 T(N)	配平迎角 α(Deg)	升降舵偏度 δe(Deg)
1	50	25	31.683637	0.986192	-2.662713
2	50	50	34.212389	-5.35462	0.689182
3	50	75	28.080888	-6.391402	1.237248
4	1000	25	44.000284	1.191108	-2.771036
5	1000	50	39.755155	-5.227623	0.622049
6	1000	75	26.912804	-6.300557	1.189226
7	5000	25	25.162826	7.072536	-5.88009
8	5000	50	85.47294	-4.883925	0.440362
9	5000	75	26.095816	-5.800593	0.924933

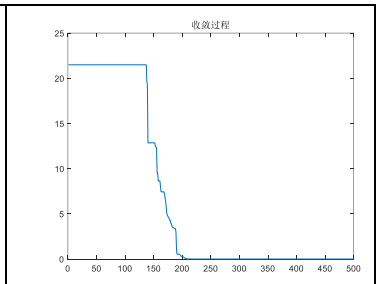
各次收敛情况图如下：



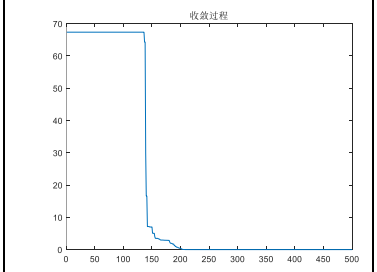
1



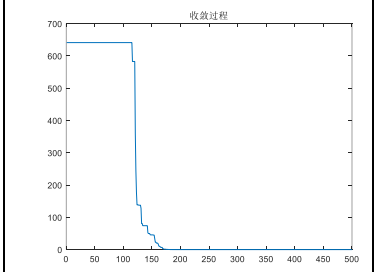
2



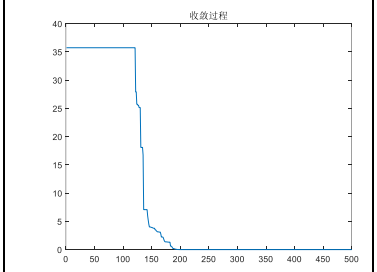
3



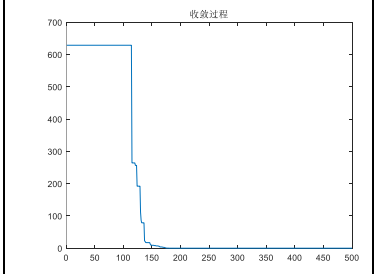
4



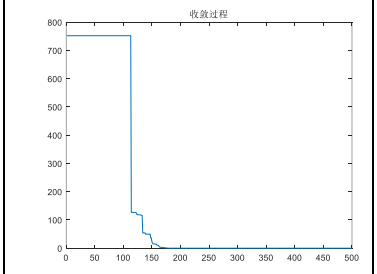
5



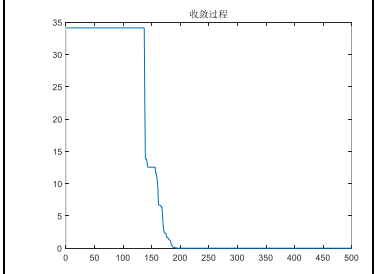
6



7



8



9

(2) Fmincon 优化函数法

在优化工具箱中使用的情况如下图：

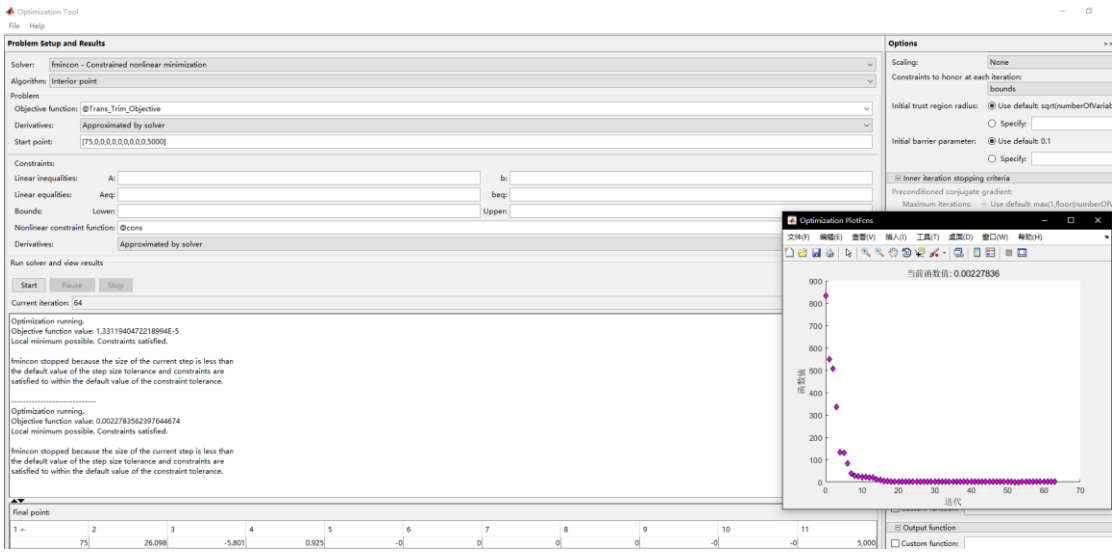


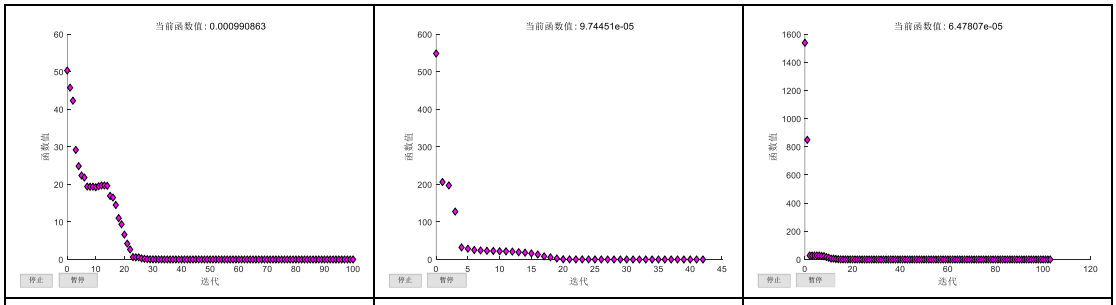
图 3 优化工具箱使用例图

由于频繁更改起始点较麻烦，导出代码，设置速度 V、高度 h 为全局变量，以运行代码 Trim_with_fmincon.m 的方式解算配平，本方法得到的配平状态点如下：

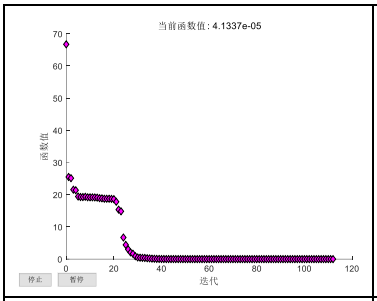
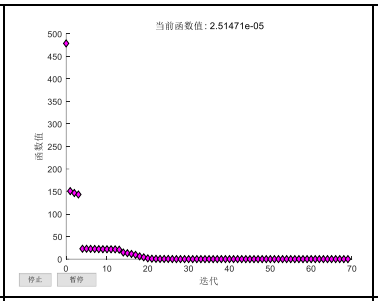
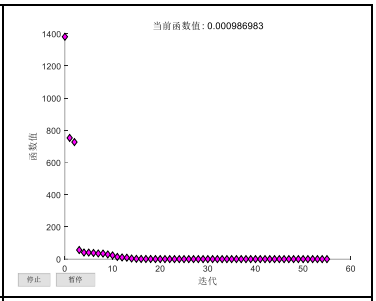
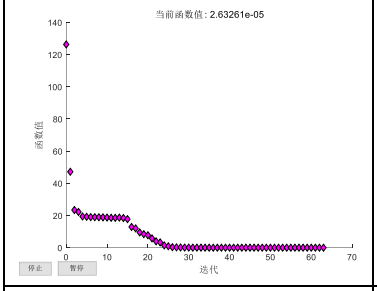
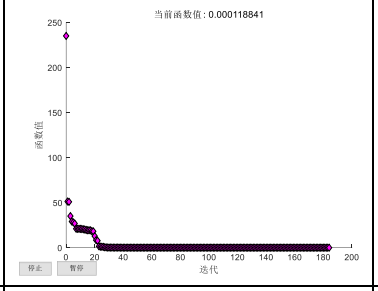
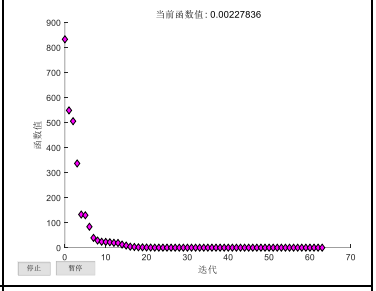
表格 3 fmincon 函数得到的配平值

序号	高度 h(m)	速度 V(m/s)	推力值 T(N)	配平迎角 α(Deg)	升降舵偏度 δe(Deg)
1	50	25	31.682558	0.986199	-2.662714
2	50	50	34.212472	-5.354621	0.689191
3	50	75	28.080944	-6.391402	1.237246
4	1000	25	31.72035	1.005508	7.456041
5	1000	50	39.755158	-5.227623	0.622044
6	1000	75	26.911821	-6.300557	1.189216
7	5000	25	25.162802	7.072536	-5.880096
8	5000	50	74.821004	-4.916274	3.143725
9	5000	75	26.098245	-5.800597	0.924897

各次收敛情况图如下：



1	2	3

		
4	5	6
		
7	8	9

(3) Simulink 建模 findop 方法

在 Simulink 中建模的情况如下：

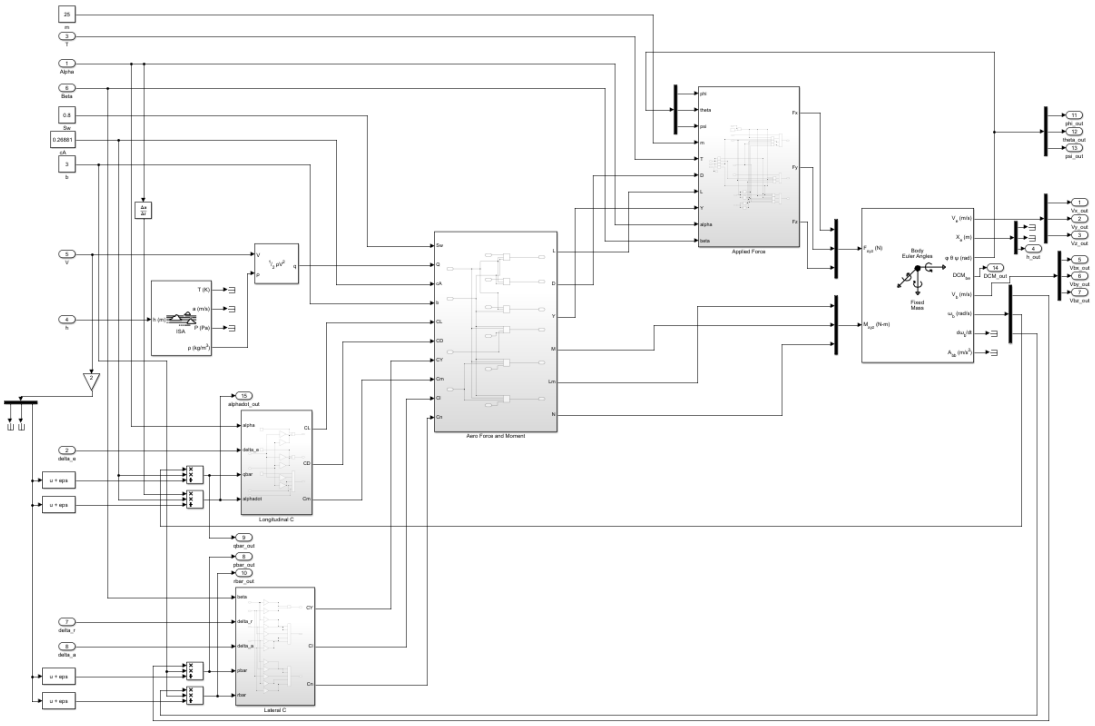


图 4 6DOF 气动建模——总系统

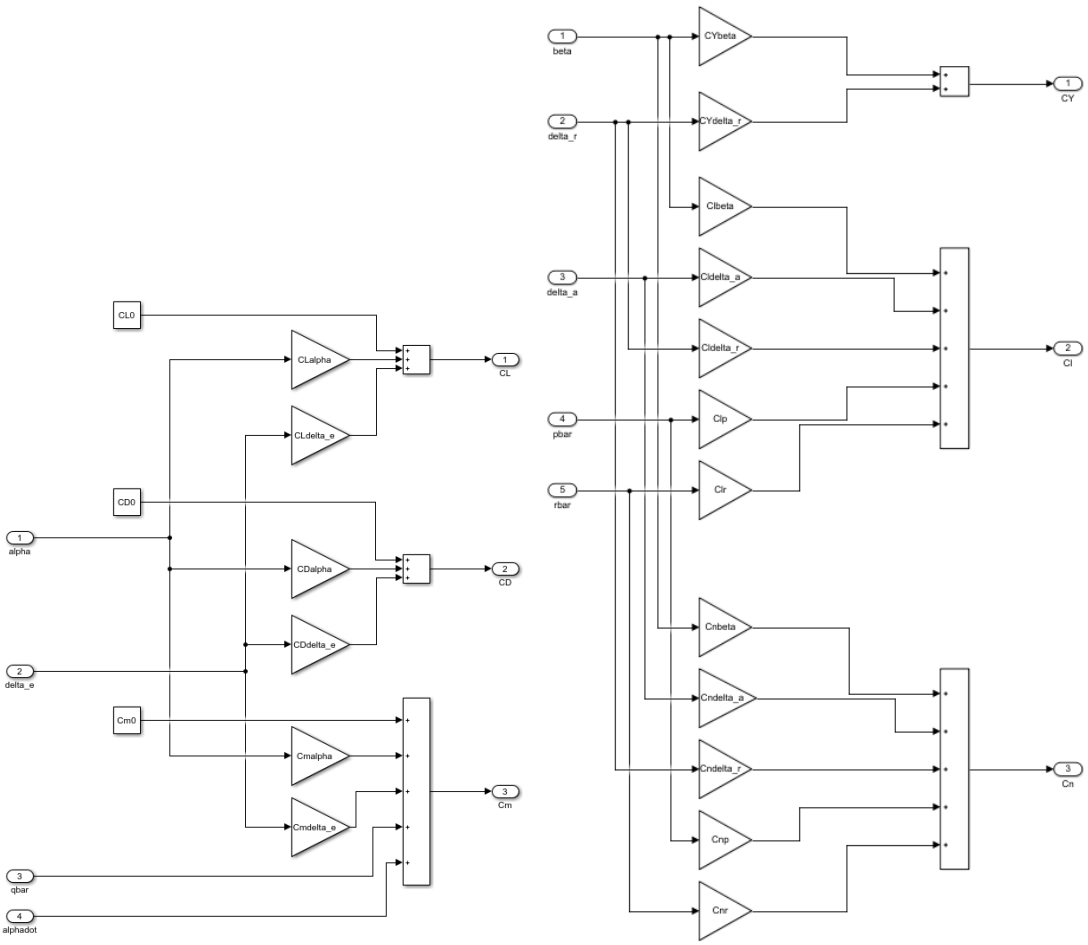


图 5 纵向气动系数&横侧向气动系数子模型

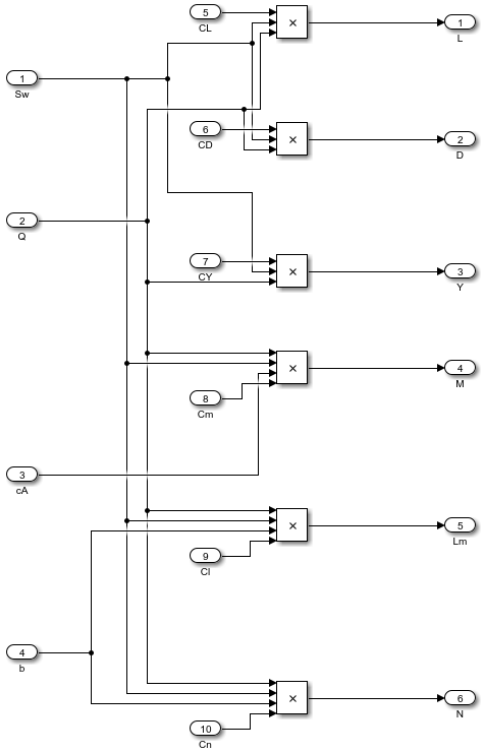


图 6 气动力子模型

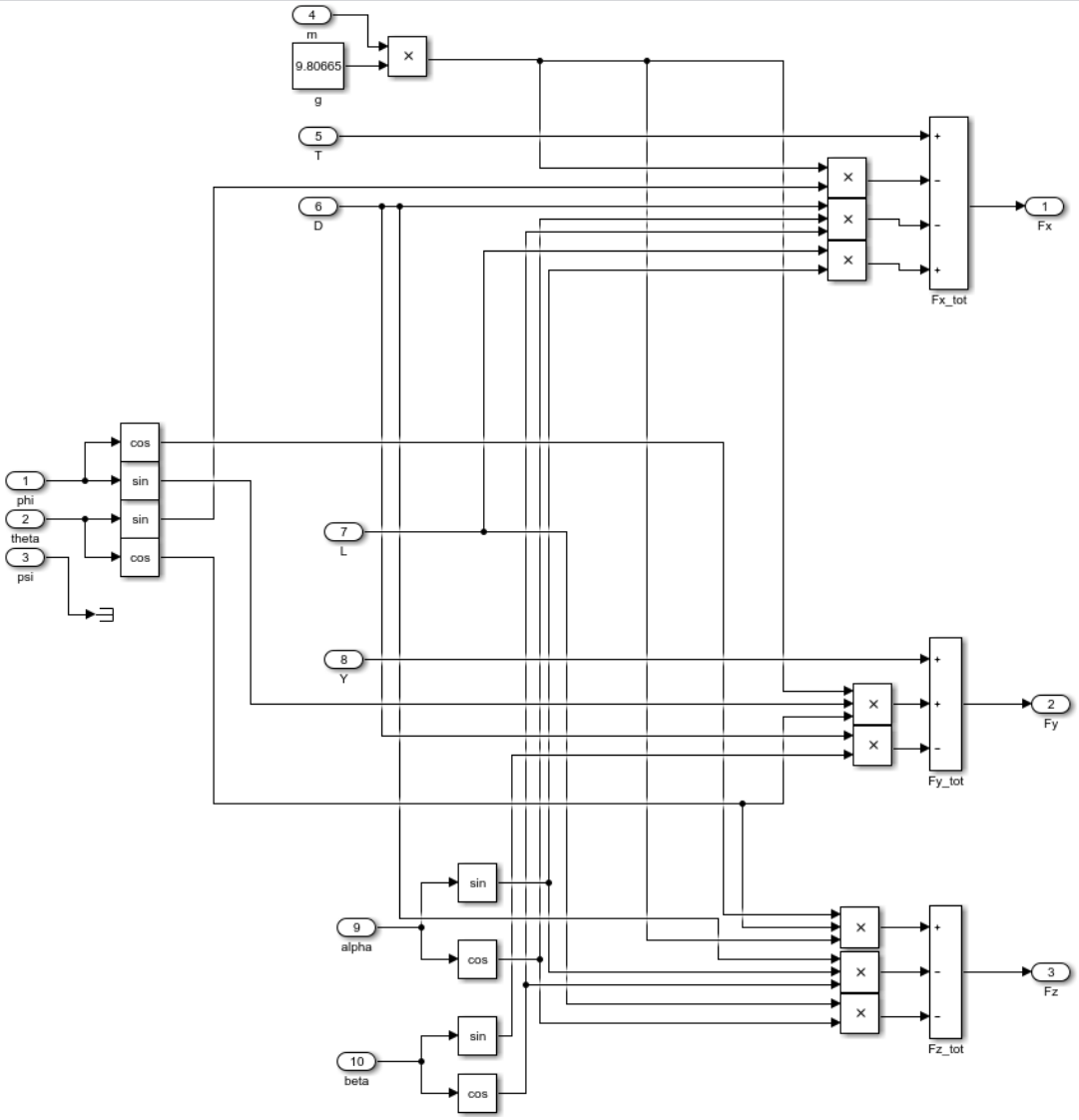


图 7 机体轴系受力子模型

使用模型线性化分析工具中查找工作点的情况如下图：

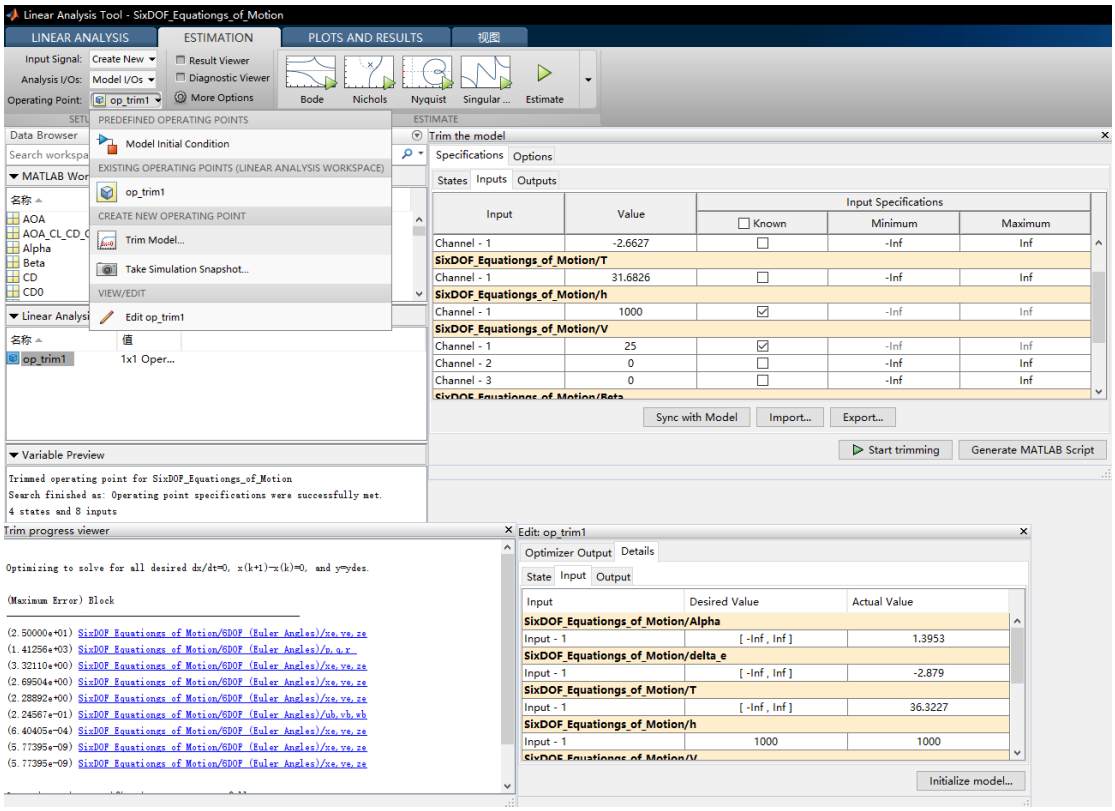


图 8 线性化分析工具中查找工作点例图

本方法得到的配平状态点如下：

表格 4 建模 findop 方法得到的配平值

序号	高度 h(m)	速度 V(m/s)	推力值 T(N)	配平迎角 α(Deg)	升降舵偏度 δe(Deg)
1	50	25	30.5527	1.0083	-2.6744
2	50	50	38.6297	-5.3826	0.70412
3	50	75	19.026	-7.6994	1.9289
4	1000	25	33.8586	1.4778	-2.9226
5	1000	50	37.52	-5.2104	0.61309
6	1000	75	11.5151	-6.9032	1.508
7	5000	25	95.6821	2.0982	-3.2506
8	5000	50	192.5851	-1.117	-1.5509
9	5000	75	59.2843	-1.8918	-1.1559

(4) 最优配平状态点评价选择

将三组各 9 次数据带入优化目标函数横向对比（代码见文件 Trim_Evaluation.m），得出选择矩阵 F：

表格 5 配平方法选择矩阵（1 表示选择）

序号	PSO 算法	fmincon 函数法	findop 法
1	1	0	0
2	1	0	0
3	0	1	0
4	1	0	0
5	1	0	0
6	1	0	0
7	1	0	0
8	1	0	0
9	1	0	0

根据选择矩阵 F，第三个工作点采用 fmincon 函数法解算的配平值，其余工作点采用 PSO 算法解算的配平值，输出最终配平结果 Final_Trim，最终配平状态点见本报告第三节。

4. 线性化、求解大导数

由小扰动线性化原理，**全量=基准量+增量**，其中
基准量：是配平后的状态值或操纵量；
小扰动：指在配平状态附近，幅值不大的运动；
由此可以推导得出（大导数系数表达形式的）飞机运动的纵向线性化方程和横侧向线性化方程如下：
飞机纵向线性化方程为：

$$\begin{cases} \Delta\dot{\bar{V}} + X_V\Delta\bar{V} + X_\alpha\Delta\alpha + X_\theta\Delta\theta = -X_{\delta_t}\Delta\delta_t \\ Z_V\Delta\bar{V} + \Delta\dot{\alpha} + Z_\alpha\Delta\alpha - \Delta\dot{\theta} = -Z_{\delta_e}\Delta\delta_e \\ M_V\Delta\bar{V} + M_{\dot{\alpha}}\Delta\dot{\alpha} + M_\alpha\Delta\alpha + \Delta\ddot{\theta} + M_q\Delta\dot{\theta} = -M_{\delta_e}\Delta\delta_e - M_{\delta_t}\Delta\delta_t \end{cases}$$

纵向运动方程的大导数及其表达式如下表所示：

表格 6 纵向线性化方程大导数计算公式

大导数	表达式	计算公式	含义
X_V	$\frac{D_V - T_V}{mV_0}$	$\frac{1}{mV_0}\{(\frac{1}{2}\rho V_0^2)S_w[2C_{D_0} + M_0 C_{D_M}] - V_0 T_V\}$	速度变化产生的切向力增量系数
X_α	$\frac{D_\alpha}{mV_0} - \frac{g}{V_0}$	$\frac{1}{mV_0}(C_{D_\alpha}\cdot\frac{1}{2}\rho V_0^2 S_w - mg)$	迎角变化产生的切向力增量系数
X_θ	$\frac{g}{V_0}$	$\frac{g}{V_0}$	俯仰角变化产生的切向力增量系数
X_{δ_t}	$-\frac{T_{\delta_t}}{mV_0}$	$-\frac{T_{\delta_t}}{mV_0}$	油门杆变化产生的切向力增量系数
Z_V	$\frac{L_V}{m}$	$\frac{1}{mV_0}(\frac{1}{2}\rho V_0^2)S_w(2C_{L_0} + M_0 C_{L_M})$	速度变化引起的法向力增量系数
Z_α	$\frac{L_\alpha}{mV_0}$	$\frac{1}{mV_0}C_{L_\alpha}\cdot\frac{1}{2}\rho V_0^2 S_w$	迎角变化产生的法向力增量系数
Z_{δ_e}	$\frac{L_{\delta_e}}{mV_0}$	$\frac{1}{mV_0}C_{\delta_e}\cdot\frac{1}{2}\rho V_0^2 S_w$	升降舵变化产生的法向力增量系数
M_V	$-\frac{V_0(M_V^a + T_V z_t)}{I_y}$	$-\frac{1}{I_y}[(\frac{1}{2}\rho V_0^2)c_A S_w(2C_{m_0} + M_0 C_{m_M}) + V_0 T_V z_t]$	速度变化引起的俯仰力矩增量系数

M_{α}	$-\frac{M_{\alpha}^{a}}{I_y}$	$-\frac{1}{I_y}(\frac{1}{2}\rho V_0^2)c_A S_w C_{m_{\alpha}}$	迎角变化产生的俯仰力矩增量系数
$M_{\dot{\alpha}}$	$-\frac{M_{\dot{\alpha}}^{a}}{I_y}$	$-\frac{1}{I_y}(\frac{1}{2}\rho V_0^2)\frac{c_A^2}{2V_0}S_w C_{m_{\dot{\alpha}}}$	迎角角速率（洗流时差）变化产生的俯仰力矩增量系数
M_q	$-\frac{M_q^{a}}{I_y}$	$-\frac{1}{I_y}(\frac{1}{2}\rho V_0^2)\frac{c_A^2}{2V_0}S_w C_{m_q}$	俯仰角速率变化产生的俯仰力矩增量系数
M_{δ_e}	$-\frac{M_{\delta_e}^{a}}{I_y}$	$-\frac{1}{I_y}(\frac{1}{2}\rho V_0^2)\frac{c_A^2}{2V_0}S_w C_{m_{\delta_e}}$	升降舵变化产生的俯仰力矩增量系数
M_{δ_t}	$-\frac{T_{\delta_t} z_t}{I_y}$	$-\frac{T_{\delta_t} z_t}{I_y}$	油门杆变化产生的俯仰力矩增量系数

飞机横侧向线性化方程为：

$$\begin{cases} mV_0(\frac{d\beta}{dt}+r)=Y_{\beta}^{s}\beta+Y_{\delta_r}^{s}\delta_r+G\phi \\ I_x\frac{dp}{dt}-I_{xz}\frac{dr}{dt}=L_{\beta}^{s}\beta+L_p^{s}p+L_r^{s}r+L_{\delta_a}^{s}\delta_a+L_{\delta_r}^{s}\delta_r \\ I_z\frac{dr}{dt}-I_{xz}\frac{dp}{dt}=N_{\beta}^{s}\beta+N_p^{s}p+N_r^{s}r+N_{\delta_a}^{s}\delta_a+N_{\delta_r}^{s}\delta_r \\ \frac{d\phi}{dt}=p \end{cases}$$

横侧向运动方程的大导数及其表达式如下表所示：

表格 7 横侧向线性化方程大导数计算公式

大导数	计算公式	含义
Y_{β}^{s}	$\frac{(\frac{1}{2}\rho V_0^2)S_w C_{Y_{\beta}}}{mV_0}$	侧滑角产生的侧力系数
Y_{ϕ}^{s}	$\frac{g}{V_0}$	滚转角产生的侧力系数
$Y_{\delta_r}^{s}$	$-\frac{(\frac{1}{2}\rho V_0^2)S_W C_{Y_{\delta_r}}}{mV_0}$	方向舵产生的侧力系数
L_{β}^{s}	$\frac{(\frac{1}{2}\rho V_0^2)S_w b C_{l_{\beta}}}{I_x}$	侧滑角产生的滚转力矩系数

L_p^s	$\frac{(\frac{1}{2}\rho V_0^2)S_w\frac{b^2}{2V_0}C_{l_p}}{I_x}$	滚转角速率产生的滚转力矩系数
L_r^s	$\frac{(\frac{1}{2}\rho V_0^2)S_w\frac{b^2}{2V_0}C_{l_r}}{I_x}$	偏航角速率产生的滚转力矩系数
$L_{\delta_a}^s$	$\frac{(\frac{1}{2}\rho V_0^2)S_wC_{Y_{\delta_a}}}{I_x}$	副翼产生的滚转力矩系数
$L_{\delta_r}^s$	$\frac{(\frac{1}{2}\rho V_0^2)S_wC_{Y_{\delta_r}}}{I_x}$	方向舵产生的滚转力矩系数
N_β^s	$\frac{(\frac{1}{2}\rho V_0^2)S_wC_{n_\beta}}{I_z}$	侧滑角产生的偏航力矩系数
N_r^s	$\frac{(\frac{1}{2}\rho V_0^2)S_w\frac{b^2}{2V_0}C_{n_r}}{I_z}$	偏航角速率产生的偏航力矩系数
N_p^s	$\frac{(\frac{1}{2}\rho V_0^2)S_w\frac{b^2}{2V_0}C_{n_p}}{I_z}$	滚转角速率产生的偏航力矩系数
$N_{\delta_a}^s$	$\frac{(\frac{1}{2}\rho V_0^2)S_wC_{n_{\delta_a}}}{I_z}$	副翼产生的偏航力矩系数
$N_{\delta_r}^s$	$\frac{(\frac{1}{2}\rho V_0^2)S_wC_{n_{\delta_r}}}{I_z}$	方向舵产生的偏航力矩系数

根据计算公式，编写相应的函数，以表格形式输出大导数计算结果，见本报告第三节。

三、最终计算结果

配平状态点计算结果：

序号	高度 h(m)	速度 V(m/s)	推力值 T(N)	配平迎角 α(Deg)	升降舵偏度 δ_e(Deg)
1	50	25	31.683637	0.986192	-2.662713
2	50	50	34.212389	-5.35462	0.689182
3	50	75	28.080944	-6.391402	1.237246
4	1000	25	44.000284	1.191108	-2.771036
5	1000	50	39.755155	-5.227623	0.622049
6	1000	75	26.912804	-6.300557	1.189226
7	5000	25	25.162826	7.072536	-5.88009
8	5000	50	85.47294	-4.883925	0.440362
9	5000	75	26.095816	-5.800593	0.924933

大导数计算结果：

序号	1	2	3	4	5	6	7	8	9
高度 h(m)	50	50	50	1000	1000	1000	5000	5000	5000
速度 V(m/s)	25	50	75	25	50	75	25	50	75
纵向线性化方程大导数									
X_V	0.0506	0.1011	0.1517	0.0461	0.0922	0.1383	0.0305	0.0610	0.0916
X_α	-0.3891	-0.1897	-0.1211	-0.3893	-0.1903	-0.1220	-0.3903	-0.1923	-0.1249
X_θ	0.3923	0.1961	0.1308	0.3923	0.1961	0.1308	0.3923	0.1961	0.1308
X_{δ_t}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
Z_V	0.6319	1.2638	1.8957	0.5762	1.1524	1.7286	0.3815	0.7631	1.1446
Z_α	0.0431	0.0863	0.1294	0.0393	0.0787	0.1180	0.0261	0.0521	0.0782
Z_{δ_e}	0.0016	0.0032	0.0048	0.0015	0.0029	0.0044	0.0010	0.0019	0.0029
M_V	1.7142	6.8568	15.4278	1.5631	6.2523	14.0676	1.0350	4.1402	9.3154
M_α	0.2116	0.8463	1.9043	0.1929	0.7717	1.7364	0.1278	0.5110	1.1498
$M_{\dot{\alpha}}$	0.0011	0.0023	0.0034	0.0010	0.0021	0.0031	0.0007	0.0014	0.0021

M_q	0.9686	1.9372	2.9058	0.8832	1.7664	2.6496	0.5848	1.1697	1.7545
M_{δ_e}	0.0011	0.0022	0.0032	0.0010	0.0020	0.0029	0.0006	0.0013	0.0019
M_{δ_t}	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
横侧向线性化方程大导数									
Y_β^s	-0.0033	-0.0065	-0.0098	-0.0030	-0.0059	-0.0089	-0.0020	-0.0039	-0.0059
Y_ϕ^s	0.3923	0.1961	0.1308	0.3923	0.1961	0.1308	0.3923	0.1961	0.1308
$Y_{\delta_r}^s$	-0.0012	-0.0024	-0.0035	-0.0011	-0.0022	-0.0032	-0.0007	-0.0014	-0.0021
L_β^s	-0.3315	-1.3259	-2.9834	-0.3023	-1.2090	-2.7203	-0.2002	-0.8006	-1.8014
L_p^s	-17.1268	-34.2535	-51.3803	-15.6167	-31.2335	-46.8502	-10.3412	-20.6824	-31.0236
L_r^s	-0.2762	-0.5525	-0.8287	-0.2519	-0.5038	-0.7556	-0.1668	-0.3336	-0.5004
$L_{\delta_a}^s$	0.0276	0.1105	0.2486	0.0252	0.1008	0.2267	0.0167	0.0667	0.1501
$L_{\delta_r}^s$	0.3714	1.4855	3.3425	0.3386	1.3546	3.0478	0.2242	0.8970	2.0182
N_β^s	0.0588	0.2351	0.5291	0.0536	0.2144	0.4824	0.0355	0.1420	0.3195
N_r^s	-0.4070	-0.8140	-1.2209	-0.3711	-0.7422	-1.1133	-0.2457	-0.4915	-0.7372
N_p^s	0.0407	0.0814	0.1221	0.0371	0.0742	0.1113	0.0246	0.0491	0.0737
$N_{\delta_a}^s$	0.0192	0.0769	0.1730	0.0175	0.0701	0.1577	0.0116	0.0464	0.1044
$N_{\delta_r}^s$	-0.0345	-0.1379	-0.3103	-0.0314	-0.1258	-0.2830	-0.0208	-0.0833	-0.1874

四、代码附录

全机系数数据：AOA_CL_CD_Cm.txt

-10	-0.30699	0.056399	0.006918
-8	-0.12795	0.045023	-0.02105
-6	0.065931	0.040349	-0.04321
-4	0.260457	0.040256	-0.06409
-2	0.45485	0.044127	-0.0835
-1	0.552691	0.047827	-0.09351
0	0.64791	0.051832	-0.10338
1	0.743208	0.057095	-0.11199
2	0.837028	0.063162	-0.12182
3	0.930372	0.070135	-0.13053
4	1.02106	0.077888	-0.1373
5	1.10971	0.086527	-0.14547
6	1.19606	0.096105	-0.15235
7	1.28065	0.106448	-0.16249
8	1.36008	0.117507	-0.16918
10	1.50686	0.141826	-0.18368
12	1.63177	0.169244	-0.20312
15	1.73971	0.216285	-0.24708
18	1.727	0.28649	-0.34593
20	1.60298	0.379494	-0.42825

①迎角相关静导数计算文件：AOA_CL_CD_Cm.m

```
%% 清空环境
clc,clear;
%% 加载迎角相关全机系数
load 'AOA_CL_CD_Cm.txt';
AOA = AOA_CL_CD_Cm(:,1);
CL = AOA_CL_CD_Cm(:,2);
CD = AOA_CL_CD_Cm(:,3);
Cm = AOA_CL_CD_Cm(:,4);
%% 用三次样条插值函数spline进行插值拟合
d = 0.01;                                    %插值间隔
x = -15:d:25;
yL = spline(AOA,CL,x);                       %三次样条插值
yD = spline(AOA,CD,x);
ym = spline(AOA,Cm,x);
%% 绘图
```

```
plot(AOA,CL,'ro',x,yL,'r');hold on;
plot(AOA,CD,'go',x,yD,'g');hold on;
plot(AOA,Cm,'bo',x,ym,'b');hold on;
title('升力系数/阻力系数/俯仰力矩系数随迎角的变化情况');
%% 求关键参数
[yL0,ZLI] = min(abs(yL));                          %ZLI:Zero-lift AOA Index
alpha0 = x(ZLI);                                   %零升迎角
fprintf('零升迎角alpha0:% .2f\n',alpha0);
[yLmax,CI] = max(yL);                              %CI:Critical AOA Index
alphaC = x(CI);                                    %临界迎角
fprintf('临界迎角alphaC:% .2f\n',alphaC);
CL0 = yL(find(~x));                                %零迎角升力系数
fprintf('零迎角升力系数CL0:% .6f\n',CL0);
CD0 = yD(find(~x));                                %零迎角阻力系数
fprintf('零迎角阻力系数CD0:% .6f\n',CD0);
Cm0 = ym(ZLI);                                     %零升力矩系数
fprintf('零升俯仰力矩系数Cm0:% .6f\n',Cm0);
%% 在线性化范围求纵向气动力、气动力矩系数静导数
yL_linear = yL(ZLI:CI-5/d);                        %升力系数-迎角曲线线性部分（零
升迎角至临界迎角-5° ）
CLalpha = mean(diff(yL_linear)/d);                 %升力系数随迎角变化的导数
fprintf('升力系数随迎角变化的导数CLalpha:% .6f\n',CLalpha);
yD_linear = yD(ZLI:CI-5/d);                        %阻力系数-迎角曲线线性部分（零
升迎角至临界迎角-5° ）
CDalpha = mean(diff(yD_linear)/d);                 %阻力系数随迎角变化的导数
fprintf('阻力系数随迎角变化的导数CDalpha:% .6f\n',CDalpha);
ym_linear = ym(ZLI:CI-5/d);                        %俯仰力矩系数-迎角曲线线性部分
（零升迎角至临界迎角-5° ）
Cmalpha = mean(diff(ym_linear)/d);                 %俯仰系数随迎角变化的导数
fprintf('俯仰力矩系数随迎角变化的导数Cmalpha:% .6f\n',Cmalpha);
%% 加载给定气动力、气动力矩系数，以便Simulink模型使用
CYbeta = -0.00668;  Cnbeta = 0.00104;   Clbeta = -0.00072;
CLdelta_e = 0.00656;    CDdelta_e = 0.00036;    Cmdelta_e = -0.01684;
CYdelta_r = 0.00484;    Cndelta_r = -0.00122;   Cldelta_r = -0.00008;
CYdelta_a = 0.00018;    Cndelta_a = 0.00034;    Cldelta_a = -0.00393;
Cmqbar = -7.58;     Cmalphadot = -1.64;
Cnr = -0.04;    Clp = -0.62;    Clr = -0.01;    Cnp = 0.004;
Ix = 1.986;     Iy = 3.447;     Iz = 5.392;     Ixy = 0;    Ixz = 0.011;    Iyz = 0;
%% Simulink模型仿真、线性化配平输入参数设置
h = 1000;       V = [25, 0, 0];
t = 10;     Alpha = 1.078295;   delta_e =   8.028008;      Beta = 0;
T = 35.488724;      delta_r = 0.000008;     delta_a = -0.181498 ;
```

飞机重力计算函数：fG.m

（假设重力加速度不随高度变化）

```
function G = fG(m)              %计算飞机所受重力
G = m * 9.80665;                %当前的重力加速度g = 9.80665
```

动压计算函数：fQ.m

```
function Q = fQ(h, V)
% 由ISA简表推出的当前飞行高度与空气密度的关系函数
rho0=1.225;T0=288.15;
if h<=11000
    T=T0-0.0065*h;
    rho=rho0*(T/T0)^4.25588;
elseif h>11000&&h<=20000
    T=216.65;
    rho=0.36392*exp((-h+11000)/6341.62);
else
    T=216.65+0.001*(h-20000);
    rho=0.088035*(T/216.65)^-35.1632;
end
Q = 0.5 * rho * V * V;
```

升力系数计算函数：fCL.m

```
function CL = fCL(Alpha, delta_e)
CL = 0.647910 + 0.088485 * Alpha + 0.00656 * delta_e;
```

阻力系数计算函数：fCD.m

```
function CD = fCD(Alpha, delta_e)
CD = 0.051832+0.006587 * Alpha + 0.00036 * delta_e;
```

侧力系数计算函数：fCY.m

```
function CY = fCY(Beta, delta_r)
CY = -0.00668 * Beta + 0.00484 * delta_r;
```

滚转力矩系数计算函数：fCLM.m

```
function CLM = fCLM(Beta, delta_a, delta_r, pbar, rbar)
```

```
CLM = -0.00072 * Beta - 0.00393 * delta_a - 0.00008 * delta_r - 0.62 * pbar - 0.01
* rbar;
```

俯仰力矩系数计算函数：fCM.m

```
function CM = fCM(Alpha, delta_e, qbar, alphadot)
CM = -0.036061 - 0.008902 * Alpha - 0.01684 * delta_e - 7.58 * qbar - 1.64 *
alphadot;
```

偏航力矩系数计算函数：fCN.m

```
function CN = fCN(Beta, delta_a, delta_r, pbar, rbar)
CN = 0.00104 * Beta + 0.00034 * delta_a - 0.00122 * delta_r + 0.004 * pbar - 0.04 *
rbar;
```

气动力计算函数：Aero_Force.m

```
function ADF = Aero_Force(Ci, Q, Sw)
ADF = Ci * Q * Sw;
```

气动力矩计算函数：Aero_Moment.m

```
function ADM = Aero_Moment(Ci, Q, Sw, s)          %s表示飞机结构参数，计算M时为气动弦
长cA，计算LM和N时为展长b
ADM = Ci * Q * Sw * s;
```

机体轴合外力计算函数：BodyAxisForce.m

```
function F = BodyAxisForce(T, G, L, D, Y, Alpha)
F = sqrt((T - G * sin(Alpha) - D * cos(Alpha) + L * sin(Alpha))^2 ...,
    + Y^2 ...,
    + (G * cos(Alpha) - D * sin(Alpha) - L * cos(Alpha))^2)
```

机体轴合外力矩计算函数：BodyAxisMoment.m

```
function M = BodyAxisMoment(LM, M, N)
M = sqrt(LM^2 + M^2 + N^2)
```

（原始）配平目标函数：Trim_Objective.m

（程序中并未使用，用作对照变量序号）

```
function OBJ = Trim_Objective (V, T, Alpha, delta_e, delta_r, delta_a, pbar, qbar,
rbar, alphadot, h)
OBJ = sqrt((BodyAxisForce(T, fG(25), Aero_Force(fCL(Alpha, delta_e), fQ(h, V),
0.8), ...,
        Aero_Force(fCD(Alpha, delta_e), fQ(h, V), 0.8), ...,
        Aero_Force(fCY(0, delta_r), fQ(h, V), 0.8), ...,
        Alpha))^2 + (BodyAxisMoment(Aero_Moment(fCLM(0, delta_a, delta_r, pbar,
rbar), fQ(h, V), 0.8, 3), ...,
        Aero_Moment(fCM(Alpha, delta_e, qbar, alphadot), fQ(h, V), 0.8,
0.26881), ...,
        Aero_Moment(fCN(0, delta_a, delta_r, pbar, rbar), fQ(h, V), 0.8, 3)))^2)
```

（转化）配平目标函数：Trans_Trim_Objective.m（输入为向量）

```
function TOBJ = Trans_Trim_Objective(x)
TOBJ = sqrt((BodyAxisForce(x(2), fG(25), Aero_Force(fCL(x(3), x(4)), fQ(x(11),
x(1)), 0.8), ...,
        Aero_Force(fCD(x(3), x(4)), fQ(x(11), x(1)), 0.8), ...,
        Aero_Force(fCY(0, x(5)), fQ(x(11), x(1)), 0.8), ...,
        x(3)))^2 + (BodyAxisMoment(Aero_Moment(fCLM(0, x(6), x(5), x(7), x(9)),
fQ(x(11), x(1)), 0.8, 3), ...,
        Aero_Moment(fCM(x(3), x(4), x(8), x(10)), fQ(x(11), x(1)), 0.8,
0.26881), ...,
        Aero_Moment(fCN(0, x(6), x(5), x(7), x(9)), fQ(x(11),x(1)), 0.8, 3)))^2)
```

②带约束的粒子群优化算法：PSO_with_Constraints.m

```
%% 清空环境
clc;clear;
%% 目标函数
%    TOBJ = Trans_Trim_Objective(x)
%    nvars = 11
%    约束条件为：
%    x(2) >= 0;              推力大于0
%    x(3) < 16.35;           迎角小于临界迎角
%    |x(4)| < 25;            升降舵偏度小于25°
%    x(1) = V;               飞行速度V
%    x(11) = h;              飞行高度h
%    x(5~10) = 0;            delta_r=delta_a=pbar=qbar=rbar=alphadot=0
h = 5000;          V = 75;
```

```
c1 = [0,1,0,0,0,0,0,0,0,0,0];
c2 = [0,0,1,0,0,0,0,0,0,0,0];
c3 = [0,0,0,1,0,0,0,0,0,0,0];
ceq1 = [1,0,0,0,0,0,0,0,0,0,0];
ceq2 = [0,0,0,0,0,0,0,0,0,0,1];
ceq3 = [0,0,0,0,1,0,0,0,0,0,0];
ceq4 = [0,0,0,0,0,1,0,0,0,0,0];
ceq5 = [0,0,0,0,0,0,1,0,0,0,0];
ceq6 = [0,0,0,0,0,0,0,1,0,0,0];
ceq7 = [0,0,0,0,0,0,0,0,1,0,0];
ceq8 = [0,0,0,0,0,0,0,0,0,1,0];
epsilon = eps;                  %等式约束容差
fun = @Trans_Trim_Objective;
cons1 = @(X)(c1*X>=0);
cons2 = @(X)(c2*X<16.35);
cons3 = @(X)(abs(c3*X)<25);
cons4 = @(X)(abs(ceq1*X-V)<=epsilon);
cons5 = @(X)(abs(ceq2*X-h)<=epsilon*h/10); %V,h量级不同，容差有转换系数
cons6 = @(X)(abs(ceq3*X)<=epsilon);
cons7 = @(X)(abs(ceq4*X)<=epsilon);
cons8 = @(X)(abs(ceq5*X)<=epsilon);
cons9 = @(X)(abs(ceq6*X)<=epsilon);
cons10 = @(X)(abs(ceq7*X)<=epsilon);
cons11 = @(X)(abs(ceq8*X)<=epsilon);
%% 设置种群参数
sizepop = 500;                          % 初始种群个数
dim = 11;                               % 空间维数/随nvars改变
ger = 500;                              % 最大迭代次数     (建议范围50~1000)
xlimit_max = 100*ones(dim,1);           % 设置位置参数限制
xlimit_min = -100*ones(dim,1);
vlimit_max = 1*ones(dim,1);             % 设置速度限制
vlimit_min = -1*ones(dim,1);
c_1 = 0.5;                              % 初始惯性权重
c_2 = 2;                                % 自我学习因子
c_3 = 2;                                % 群体学习因子
PV = 10^10;                             % 原始惩罚值
%% 生成初始种群
%  首先随机生成初始种群位置
%  然后随机生成初始种群速度
%  然后初始化个体历史最佳位置，以及个体历史最佳适应度
%  然后初始化群体历史最佳位置，以及群体历史最佳适应度
for i=1:dim
    for j=1:sizepop
        pop_x(i,j) = xlimit_min(i)+(xlimit_max(i) - xlimit_min(i))*rand;  % 初始种
```

```
群的位置
        pop_v(i,j) = vlimit_min(i)+(vlimit_max(i) - vlimit_min(i))*rand;  % 初始种
群的速度
        pop_x(1,j) = V;                                                   % 强制设
定初始种群的位置
        pop_x(11,j) = h;                                                  % 分别对
应V, delta_e, delta_r,
        pop_x(5,j) = 0;                                                   % pbar,
qbar, rbar, alphadot, h
        pop_x(6,j) = 0;
        pop_x(7,j) = 0;
        pop_x(8,j) = 0;
        pop_x(9,j) = 0;
        pop_x(10,j) = 0;
    end
end
pbest = pop_x;                                  % 每个个体的历史最佳位置
for j=1:sizepop
    if cons1(pop_x(:,j))
        if cons2(pop_x(:,j))
            if cons3(pop_x(:,j))
                if cons4(pop_x(:,j))
                    if cons5(pop_x(:,j))
                        if cons6(pop_x(:,j))
                            if cons7(pop_x(:,j))
                                if cons8(pop_x(:,j))
                                    if cons9(pop_x(:,j))
                                        if cons10(pop_x(:,j))
                                            if cons11(pop_x(:,j))
                                                fitness_pbest(j) =
fun(pop_x(:,j));     % 每个个体的历史最佳适应度
                                            else
                                                fitness_pbest(j) = PV/10^6;
                                            end
                                        else
                                            fitness_pbest(j) = PV/10^6;
                                        end
                                    else
                                        fitness_pbest(j) = PV/10^6;
                                    end

                                else
                                    fitness_pbest(j) = PV/10^6;
                                end
```

```
                        else
                            fitness_pbest(j) = PV/10^6;
                        end
                    else
                        fitness_pbest(j) = PV/10^6;
                    end
                else
                    fitness_pbest(j) = PV/10^6;
                end
            else
                fitness_pbest(j) = PV/10^6;
            end
        else
            fitness_pbest(j) = PV/10^6;
        end
    else
        fitness_pbest(j) = PV/10^6;
    end
end
% 初始化种群时实际惩罚值较小，达到扩大搜索空间的目的
gbest = pop_x(:,1);                             % 种群的历史最佳位置
fitness_gbest = fitness_pbest(1);               % 种群的历史最佳适应度
for j=1:sizepop
    if fitness_pbest(j) < fitness_gbest         % 如果求最小值，则为<; 如果求最大值，
则为>;
        gbest = pop_x(:,j);
        fitness_gbest=fitness_pbest(j);
    end
end


%% 粒子群迭代
%    更新速度并对速度进行边界处理
%    更新位置并对位置进行边界处理
%    进行自适应变异
%    进行约束条件判断并计算新种群各个个体位置的适应度
%    新适应度与个体历史最佳适应度做比较
%    个体历史最佳适应度与种群历史最佳适应度做比较
%    再次循环或结束

iter = 1;                           %迭代次数
```

```
record = zeros(ger, 1);           % 记录器
while iter <= ger
    for j=1:sizepop
        %    更新速度并对速度进行边界处理
        %    惯性权重因子叠加线性函数，使之随迭代次数增加而减小
        pop_v(:,j)= c_1*(-1/ger*iter+1) * pop_v(:,j) + c_2*rand*(pbest(:,j)-
pop_x(:,j))+c_3*rand*(gbest-pop_x(:,j));% 速度更新
        for i=2:dim-1
            if  pop_v(i,j) > vlimit_max(i)
                pop_v(i,j) = vlimit_max(i);
            end
            if  pop_v(i,j) < vlimit_min(i)
                pop_v(i,j) = vlimit_min(i);
            end
        end

        %    更新位置并对位置进行边界处理
        pop_x(:,j) = pop_x(:,j) + pop_v(:,j);% 位置更新
        for i=2:dim-7                                    %不更新种群第1个元素，第5-
11个元素的值
            if  pop_x(i,j) > xlimit_max(i)
                pop_x(i,j) = xlimit_max(i);
            end
            if  pop_x(i,j) < xlimit_min(i)
                pop_x(i,j) = xlimit_min(i);
            end
        end

        %    进行自适应变异
        if rand > 0.85
            i=ceil(dim*rand);
            if i > 1
                if i < 5                                 %不变异种群第1个元素，第5-
11个元素的值
                    pop_x(i,j)=xlimit_min(i) + (xlimit_max(i) - xlimit_min(i)) *
rand;
                end
            end
        end

        %    进行约束条件判断并计算新种群各个个体位置的适应度
        if cons1(pop_x(:,j))
            if cons2(pop_x(:,j))
                if cons3(pop_x(:,j))
```

```
                    if cons4(pop_x(:,j))
                        if cons5(pop_x(:,j))
                            if cons6(pop_x(:,j))
                                if cons7(pop_x(:,j))
                                    if cons8(pop_x(:,j))
                                        if cons9(pop_x(:,j))
                                            if cons10(pop_x(:,j))
                                                if cons11(pop_x(:,j))
                                                    fitness_pop(j) =
fun(pop_x(:,j));    % 当前个体的适应度
                                                else
                                                    fitness_pop(j) = PV/(1+1.05^(-
iter+ger/3));
                                                end
                                            else
                                                fitness_pop(j) = PV/(1+1.05^(-
iter+ger/3));
                                            end
                                        else
                                            fitness_pop(j) = PV/(1+1.05^(-
iter+ger/3));
                                        end

                                    else
                                        fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
                                    end
                                else
                                    fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
                                end
                            else
                                fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
                            end
                        else
                            fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
                        end
                    else
                        fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
                    end
                else
                    fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
                end
            else
                fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
            end
        else
            fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
        end
```

```
        else
            fitness_pop(j) = PV/(1+1.05^(-iter+ger/3));
        end
        % 进行适应值比例变换：
        % 迭代时惩罚系数叠加类Sigmoid函数，相当于实际惩罚值随着代际增大而增大。
        % 迭代开始时，实际惩罚值小，抑制竞争，扩大搜索范围；
        % 迭代后期时，实际惩罚值大，鼓励竞争，加快收敛速度。

        %    新适应度与个体历史最佳适应度做比较
        if fitness_pop(j) < fitness_pbest(j)        % 如果求最小值，则为<; 如果求最
大值，则为>;
            pbest(:,j) = pop_x(:,j);                % 更新个体历史最佳位置
            fitness_pbest(j) = fitness_pop(j);      % 更新个体历史最佳适应度
        end

        %    个体历史最佳适应度与种群历史最佳适应度做比较
        if fitness_pbest(j) < fitness_gbest         % 如果求最小值，则为<; 如果求最
大值，则为>;
            gbest = pbest(:,j);                     % 更新群体历史最佳位置
            fitness_gbest=fitness_pbest(j);         % 更新群体历史最佳适应度
        end
    end

    record(iter) = fitness_gbest;%最大值记录

    iter = iter+1;

end
%% 迭代结果输出

plot(record);title('收敛过程');
disp(['最优值：',num2str(fitness_gbest)]);
disp('变量取值：');
fprintf('%.6f\t',gbest);
```

粒子群优化解算得到的配平状态值：Trim by PSO.txt

25	31.683637	0.986192	-2.662713	0	0	0	0	0	0	50
50	34.212389	-5.35462	0.689182	0	0	0	0	0	0	50
75	28.080888	-6.391402	1.237248	0	0	0	0	0	0	50
25	44.000284	1.191108	-2.771036	0	0	0	0	0	0	1000
50	39.755155	-5.227623	0.622049	0	0	0	0	0	0	1000
75	26.912804	-6.300557	1.189226	0	0	0	0	0	0	1000
25	25.162826	7.072536	-5.88009	0	0	0	0	0	0	5000

50	85.47294	-4.883925	0.440362	0	0	0	0	0	0	5000
75	26.095816	-5.800593	0.924933	0	0	0	0	0	0	5000

③利用 MATLAB 优化工具箱自带的 fmincon 函数方法：

Trim_with_fmincon.m

```
%% 清空环境
clc,clear;
%% 设置高度值，速度值；设置起始点
global h V
h = 5000;       V = 75;
x0 = [V, 0, 0, 0, 0, 0, 0, 0, 0, 0, h];
Trim_Result = Trim_by_fmincon(x0);
disp('变量取值：');
fprintf('%.6f\t',Trim_Result);
%% 采用MATLAB内置fmincon优化函数，默认设置
function [x,fval,exitflag,output,lambda,grad,hessian] = Trim_by_fmincon(x0)
options = optimoptions('fmincon');
% 配平值输出与迭代图示
options = optimoptions(options,'Display', 'off');
options = optimoptions(options,'PlotFcn', { @optimplotfval });
[x,fval,exitflag,output,lambda,grad,hessian] = ...
fmincon(@Trans_Trim_Objective,x0,[],[],[],[],[],[],@cons,options);
end
```

③为 fmincon 优化函数编写的约束函数：cons.m

```
function [c,ceq] = cons(x)
global h V
c(1) = -x(2);                       %推力T大于0
c(2) = x(3) - 16.35;                %迎角Alpha小于临界迎角
c(3) = x(4) - 25;                   %升降舵偏度小于25°
c(4) = -(x(4) + 25);
epsilon = eps;                          %角速率容差epsilon
c(5) = x(5) - epsilon;                  %delta_r < epsilon
c(6) = -(x(5) + epsilon);
c(7) = x(6) - epsilon;                  %delta_a < epsilon
c(8) = -(x(6) + epsilon);
c(9) = x(7) - epsilon;                  %|pbar| < epsilon
c(10) = -(x(7) + epsilon);
c(11) = x(8) - epsilon;                  %|qbar| < epsilon
c(12) = -(x(8) + epsilon);
```

```
c(13) = x(9) - epsilon;                    %|rbar| < epsilon
c(14) = -(x(9) + epsilon);
c(15) = x(10) - epsilon;                   %|alphadot| < epsilon
c(16) = -(x(10) + epsilon);

ceq(1) = x(1) - V;                  %速度V
ceq(2) = x(11) - h;                 %高度h
```

fmincon 优化解算得到的配平状态值：Trim by fmincon.txt

25 31.682558 0.986199 -2.662714 0 0 0 0 0 0 50
50 34.212472 -5.354621 0.689191 0 0 0 0 0 0 50
75 28.080944 -6.391402 1.237246 0 0 0 0 0 0 50
25 31.72035 1.005508 7.456041 0.000031 -0.000267 0.000002 -0.019841 -0.000003 -0.012303 1000
50 39.755158 -5.227623 0.622044 0 0 0 0 0 0 1000
75 26.911821 -6.300557 1.189216 0 0 0 0 0 0 1000
25 25.162802 7.072536 -5.880096 0 0 0 0 0 0 5000
50 74.821004 -4.916274 3.143725 -0.000033 0.000268 -0.000002 -0.006044 0.000003 0.000353 5000
75 26.098245 -5.800597 0.924897 0 0 0 0 0 0 5000

④SIMULINK 建模，Trim Model 配平，利用 findop 函数方法：

Trim_with_SimulinkModel.m（自动生成的代码）

```
%% Search for a specified operating point for the model -
SixDOF_Equationgs_of_Motion.
%
% This MATLAB script is the command line equivalent of the trim model
% tab in linear analysis tool with current specifications and options.
% It produces the exact same operating points as hitting the Trim button.

% MATLAB(R) file generated by MATLAB(R) 9.4 and Simulink Control Design (TM) 5.1.
%
% Generated on: 08-Dec-2020 23:47:27

%% Specify the model name
model = 'SixDOF_Equationgs_of_Motion';

%% Create the operating point specification object.
opspec = operspec(model);
```

```
%% Set the constraints on the states in the model.
% - The defaults for all states are Known = false, SteadyState = true,
%   Min = -Inf, Max = Inf, dxMin = -Inf, and dxMax = Inf.

% State (1) - SixDOF_Equationgs_of_Motion/6DOF (Euler Angles)/Calculate DCM & Euler
Angles/phi theta psi
% - Default model initial conditions are used to initialize optimization.

% State (2) - SixDOF_Equationgs_of_Motion/6DOF (Euler Angles)/p,q,r
% - Default model initial conditions are used to initialize optimization.

% State (3) - SixDOF_Equationgs_of_Motion/6DOF (Euler Angles)/ub,vb,wb
% - Default model initial conditions are used to initialize optimization.

% State (4) - SixDOF_Equationgs_of_Motion/6DOF (Euler Angles)/xe,ye,ze
% - Default model initial conditions are used to initialize optimization.

%% Set the constraints on the inputs in the model.
% - The defaults for all inputs are Known = false, Min = -Inf, and
% Max = Inf.

% Input (1) - SixDOF_Equationgs_of_Motion/Alpha
opspec.Inputs(1).u = 0.9862;

% Input (2) - SixDOF_Equationgs_of_Motion/delta_e
opspec.Inputs(2).u = -2.6627;

% Input (3) - SixDOF_Equationgs_of_Motion/T
opspec.Inputs(3).u = 31.6826;
opspec.Inputs(3).Min = 0;

% Input (4) - SixDOF_Equationgs_of_Motion/h
opspec.Inputs(4).u = 5000;
opspec.Inputs(4).Known = true;

% Input (5) - SixDOF_Equationgs_of_Motion/V
% - Default model initial conditions are used to initialize optimization.
opspec.Inputs(5).Known = [true;false;false];

% Input (6) - SixDOF_Equationgs_of_Motion/Beta
opspec.Inputs(6).u = 2.2204e-16;
opspec.Inputs(6).Known = true;
```

```
% Input (7) - SixDOF_Equationgs_of_Motion/delta_r
opspec.Inputs(7).u = 2.2204e-16;

% Input (8) - SixDOF_Equationgs_of_Motion/delta_a
opspec.Inputs(8).u = 2.2204e-16;

%% Set the constraints on the outputs in the model.
% - The defaults for all outputs are Known = false, Min = -Inf, and
% Max = Inf.

% Output (1) - SixDOF_Equationgs_of_Motion/Vx_out
% - Default model initial conditions are used to initialize optimization.

% Output (2) - SixDOF_Equationgs_of_Motion/Vy_out
% - Default model initial conditions are used to initialize optimization.

% Output (3) - SixDOF_Equationgs_of_Motion/Vz_out
% - Default model initial conditions are used to initialize optimization.

% Output (4) - SixDOF_Equationgs_of_Motion/h_out
% - Default model initial conditions are used to initialize optimization.

% Output (5) - SixDOF_Equationgs_of_Motion/Vbx_out
% - Default model initial conditions are used to initialize optimization.

% Output (6) - SixDOF_Equationgs_of_Motion/Vby_out
% - Default model initial conditions are used to initialize optimization.

% Output (7) - SixDOF_Equationgs_of_Motion/Vbz_out
% - Default model initial conditions are used to initialize optimization.

% Output (8) - SixDOF_Equationgs_of_Motion/pbar_out
% - Default model initial conditions are used to initialize optimization.

% Output (9) - SixDOF_Equationgs_of_Motion/qbar_out
% - Default model initial conditions are used to initialize optimization.

% Output (10) - SixDOF_Equationgs_of_Motion/rbar_out
% - Default model initial conditions are used to initialize optimization.

% Output (11) - SixDOF_Equationgs_of_Motion/phi_out
% - Default model initial conditions are used to initialize optimization.

% Output (12) - SixDOF_Equationgs_of_Motion/theta_out
```

```
% - Default model initial conditions are used to initialize optimization.

% Output (13) - SixDOF_Equationgs_of_Motion/psi_out
% - Default model initial conditions are used to initialize optimization.

% Output (14) - SixDOF_Equationgs_of_Motion/DCM_out
% - Default model initial conditions are used to initialize optimization.

% Output (15) - SixDOF_Equationgs_of_Motion/alphadot_out
% - Default model initial conditions are used to initialize optimization.


%% Create the options
opt = findopOptions('DisplayReport','iter');

%% Perform the operating point search.
[op,opreport] = findop(model,opspec,opt);
```

findop 方法解算得到的配平状态值：Trim by findop.txt

```
25   30.5527 1.0083   -2.6744 0    0    0    0    0    0    50
50   38.6297 -5.3826  0.70412 0    0    0    0    0    0    50
75   19.026   -7.6994  1.9289  0    0    0    0    0    0    50
25   33.8586 1.4778   -2.9226 0    0    0    0    0    0    1000
50   37.52    -5.2104  0.61309 0    0    0    0    0    0    1000
75   11.5151 -6.9032  1.508    0    0    0    0    0    0    1000
25   95.6821 2.0982   -3.2506 0    0    0    0    0    0    5000
50   192.5851     -1.117    -1.5509 0    0    0    0    0    0    5000
75   59.2843 -1.8918  -1.1559 0    0    0    0    0    0    5000
```

⑤对三种配平方法评价：Trim_Evaluation.m

```
%% 清空环境
clc,clear;
%% 三种配平方法评价
load 'Trim by PSO.txt';
load 'Trim by fmincon.txt';
load 'Trim by findop.txt';
% 计算配平目标函数值
for i = 1:9
    f1(i,1) = Trans_Trim_Objective(Trim_by_PSO(i,:));        % 粒子群算法
    f2(i,1) = Trans_Trim_Objective(Trim_by_fmincon(i,:));    % fmincon函数
    f3(i,1) = Trans_Trim_Objective(Trim_by_findop(i,:));     % findop方法
```

```
end
%% 比较并生成选择值矩阵F（0-1矩阵）
F = zeros (9,3);
for i = 1:9
    if f1(i,1) < f2(i,1)
        if f1(i,1) < f3(i,1)
            F(i,1) = 1;
        else
            F(i,3) = 1;
        end
    else
        if f2(i,1) < f3(i,1)
            F(i,2) = 1;
        else
            F(i,3) = 1;
        end
    end
end
%% 生成最终配平状态点
Final_Trim = zeros(9,11);
for i = 1:9
    if F(i,1) == 1
        Final_Trim(i,:) = Trim_by_PSO(i,:);
    end
    if F(i,2) == 1
        Final_Trim(i,:) = Trim_by_fmincon(i,:);
    end
    if F(i,3) == 1
        Final_Trim(i,:) = Trim_by_findop(i,:);
    end
end
```

最终配平状态点：Final Trim Point.txt

25	31.683637	0.986192	-2.662713	0	0	0	0	0	0	50
50	34.212389	-5.35462	0.689182	0	0	0	0	0	0	50
75	28.080944	-6.391402	1.237246	0	0	0	0	0	0	50
25	44.000284	1.191108	-2.771036	0	0	0	0	0	0	1000
50	39.755155	-5.227623	0.622049	0	0	0	0	0	0	1000
75	26.912804	-6.300557	1.189226	0	0	0	0	0	0	1000
25	25.162826	7.072536	-5.88009	0	0	0	0	0	0	5000
50	85.47294	-4.883925	0.440362	0	0	0	0	0	0	5000
75	26.095816	-5.800593	0.924933	0	0	0	0	0	0	5000

工作点数据：Operating Point.txt

```
50   25   31.683637   0.986192    -2.662713
50   50   34.212389   -5.35462    0.689182
50   75   28.080944   -6.391402   1.237246
1000    25   44.000284   1.191108    -2.771036
1000    50   39.755155   -5.227623   0.622049
1000    75   26.912804   -6.300557   1.189226
5000    25   25.162826   7.072536    -5.88009
5000    50   85.47294    -4.883925   0.440362
5000    75   26.095816   -5.800593   0.924933
```

⑥计算大导数：Linear_Coefficients.m

```
%% 清空环境
clc,clear;
%% 加载工作点数据、飞机结构参数
load 'Operating Point.txt';
h0 = Operating_Point(:,1);      %高度h
V0 = Operating_Point(:,2);      %速度V
% 定义全局变量
global m Sw cA b g zt G L0 M0 CDM  CLM CmM TV
m = 25; Sw = 0.8;   cA = 0.26881; b = 3; g = 9.80665;   zt = 0;
T_delta_t = 0;
% 在基准运动和小扰动情况下，有T0 = D0、L0 = G，M0 = 0，有：
G = m * g;  L0 = G; M0 = 0;
% 低速飞行，各系数不随马赫数变化，有：
CDM = 0;    CLM = 0;    CmM = 0;    TV = 0;
%% 计算每个工作点的大导数
Longi = zeros(3,6,9);           % 纵向
Lati = zeros(3,5,9);            % 横侧向
for i = 1:9
    Longi(:, :, i) = [fX_V(V0(i), fQ(h0(i), V0(i))) fX_alpha(V0(i), fQ(h0(i),
V0(i))) fX(V0(i)) fX_delta_t(T_delta_t, V0(i)) eps eps; ...
                      fZ_V(V0(i), fQ(h0(i), V0(i))) fZ_alpha(V0(i), fQ(h0(i),
V0(i))) fZ_delta_e(V0(i), fQ(h0(i), V0(i))) eps eps eps; ...
                      fM_V(V0(i), fQ(h0(i), V0(i))) fM_alpha(fQ(h0(i), V0(i)))
fM_alphadot(V0(i), fQ(h0(i), V0(i))) ...
                      fM_q(V0(i), fQ(h0(i), V0(i))) fM_delta_e(V0(i), fQ(h0(i),
V0(i))) fM_delta_t(T_delta_t)];
    Lati(:, :, i) = [fY_beta(V0(i), fQ(h0(i), V0(i))) fY_phi(V0(i))
fY_delta_r(V0(i), fQ(h0(i), V0(i))) eps eps; ...
                     fL_beta(fQ(h0(i), V0(i))) fL_p(V0(i), fQ(h0(i), V0(i)))
```

```
fL_r(V0(i), fQ(h0(i), V0(i))) ...
                    fL_delta_a(fQ(h0(i), V0(i))) fL_delta_r(fQ(h0(i), V0(i))); ...
                    fN_beta(fQ(h0(i), V0(i))) fN_r(V0(i), fQ(h0(i), V0(i)))
fN_p(V0(i), fQ(h0(i), V0(i))) ...
                    fN_delta_a(fQ(h0(i), V0(i))) fN_delta_r(fQ(h0(i), V0(i)))];
end
%% 以表格形式输出
LongiCo = zeros(9, 13);
LatiCo = zeros(9, 13);
for k = 1:9
    %每个工作点的纵向线性化方程大导数
    for j = 1:4
        LongiCo(k, j) = Longi(1, j, k);
    end
    for j = 1:3
        LongiCo(k, j+4) = Longi(2, j, k);
    end
    for j = 1:6
        LongiCo(k, j+7) = Longi(3, j, k);
    end
    %每个工作点的横侧向线性化方程大导数
    for j =1:3
        LatiCo(k, j) = Lati(1, j, k);
    end
    for j =1:5
        LatiCo(k, j+3) = Lati(2, j, k);
    end
    for j =1:5
        LatiCo(k, j+8) = Lati(3, j, k);
    end
end
TransLongiCo = LongiCo';
TransLatiCo = LatiCo';
%% 大导数计算公式
function X_V = fX_V(V0, Q)
global m Sw M0 CDM TV
CD0 = 0.051832;
X_V = 1 / (m * V0) * (Q * Sw * (2*CD0 + M0 * CDM) - V0 * TV);
end

function X_alpha = fX_alpha(V0, Q)
global m Sw G
CDalpha = 0.006587;
X_alpha = 1 / (m * V0) * (CDalpha * Q * Sw - G);
```

```
end

function X_theta = fX(V0)
global g
X_theta = g / V0;
end

function X_delta_t = fX_delta_t(T_delta_t,V0)
global m
X_delta_t = -T_delta_t / (m * V0);
end

function Z_V = fZ_V(V0, Q)
global m Sw M0 CLM
CL0 = 0.647910;
Z_V = 1 / (m * V0) * Q * Sw * (2*CL0 + M0 * CLM);
end

function Z_alpha = fZ_alpha(V0, Q)
global m Sw
CLalpha = 0.088485;
Z_alpha = 1 / (m * V0) * CLalpha * Q * Sw;
end

function Z_delta_e = fZ_delta_e(V0, Q)
global m Sw
CLdelta_e = 0.00328;
Z_delta_e = 1 / (m * V0) * CLdelta_e * Q * Sw;
end

function M_V = fM_V(V0, Q)
global Sw cA zt M0 CmM TV
Cm0 = -0.036061;     Iy = 3.447;
M_V = -1 / Iy * (Q * cA * Sw * (2*Cm0 + M0 * CmM) + V0 * TV * zt);
end

function M_alpha = fM_alpha(Q)
global  Sw cA
Cmalpha = -0.008902;     Iy = 3.447;
M_alpha = -1 / Iy * Q * cA * Sw * Cmalpha;
end

function M_alphadot = fM_alphadot(V0, Q)
global Sw cA
```

```
Cmalpha = -0.008902;     Iy = 3.447;
M_alphadot = -1 / Iy * Q * cA^2 / (2*V0) * Sw * Cmalpha;
end

function M_q = fM_q(V0, Q)
global Sw cA
Cmqbar = -7.58;      Iy = 3.447;
M_q = -1 / Iy * Q * cA^2 / (2*V0) * Sw * Cmqbar;
end

function M_delta_e = fM_delta_e(V0, Q)
global Sw cA
Cmdelta_e = -0.00842;    Iy = 3.447;
M_delta_e = -1 / Iy * Q * cA^2 / (2*V0) * Sw * Cmdelta_e;
end

function M_delta_t = fM_delta_t(T_delta_t)
global zt
Iy = 3.447;
M_delta_t = -T_delta_t * zt / Iy;
end

function Y_beta = fY_beta(V0, Q)
global m Sw
CYbeta = -0.00668;
Y_beta = Q * Sw * CYbeta / (m * V0);
end

function Y_phi = fY_phi(V0)
global g
Y_phi = g / V0;
end

function Y_delta_r = fY_delta_r(V0, Q)
global m Sw
CYdelta_r = 0.00242;
Y_delta_r = -Q * Sw * CYdelta_r / (m * V0);
end

function L_beta = fL_beta(Q)
global Sw b
Clbeta = -0.00072;  Ix = 1.986;
L_beta = Q * Sw * b * Clbeta / Ix;
end
```

```
function L_p = fL_p(V0, Q)
global Sw b
Clp = -0.62;    Ix = 1.986;
L_p = Q * Sw * b^2 / (2*V0) * Clp / Ix;
end

function L_r = fL_r(V0, Q)
global Sw b
Clr = -0.01;    Ix = 1.986;
L_r = Q * Sw * b^2 / (2*V0) * Clr / Ix;
end

function L_delta_a = fL_delta_a(Q)
global Sw
CYdelta_a = 0.00018;    Ix = 1.986;
L_delta_a = Q * Sw * CYdelta_a / Ix;
end

function L_delta_r = fL_delta_r(Q)
global Sw
CYdelta_r = 0.00242;    Ix = 1.986;
L_delta_r = Q * Sw * CYdelta_r / Ix;
end

function N_beta = fN_beta(Q)
global Sw
Cnbeta = 0.00104;   Iz = 5.392;
N_beta = Q * Sw * Cnbeta / Iz;
end

function N_r = fN_r(V0, Q)
global Sw b
Cnr = -0.04;    Iz = 5.392;
N_r = Q * Sw * b^2 / (2*V0) * Cnr / Iz;
end

function N_p = fN_p(V0, Q)
global Sw b
Cnp = 0.004;    Iz = 5.392;
N_p = Q * Sw * b^2 / (2*V0) * Cnp / Iz;
end

function N_delta_a = fN_delta_a(Q)
```

```
global Sw
Cndelta_a = 0.00034;    Iz = 5.392;
N_delta_a = Q * Sw * Cndelta_a / Iz;
end

function N_delta_r = fN_delta_r(Q)
global Sw
Cndelta_r = -0.00061;   Iz = 5.392;
N_delta_r = Q * Sw * Cndelta_r / Iz;
end
```