基于微分方程和非线性规划的失联飞机搜寻方案模型

摘要

本文通过分析飞机失去动力后的受力情况，建立数学模型，通过解微分方程组，得到飞机的大致坠落轨迹，确定飞机残骸坠落在海面的范围。

考虑飞机失事时受重力和斜向上的气流阻力，气流阻力与空气密度有关，且 10000 米高度落差内密度相差极大，故考虑大气密度为垂直方向上分布函数。将气流阻力分解为水平方向的阻力及竖直方向的升力。将飞机的加速度分解为水平方向的加速度及竖直方向的加速度，建立平面坐标系上的微分动力学方程，用 Matlab 数值解法求解出微分方程的解曲线，即飞机坠落轨迹，即可计算出飞机的理论落点，然后再确定飞机残骸可能的分布区域，最后利用平行扫视搜寻法对所确定的区域进行搜索。

我们对常见的“平行线扫视搜寻法”做出改进，通过将搜救船与搜救飞机一字排开，并平均分为两批，进行横向搜索，形成一条搜索带。再根据实际搜寻半径，我们将搜救区域划分为若干个小正方形，并绘制出相应的网格图，建立了以所行哈密尔顿图为目标的图论模型。另外我们考虑到在搜索区域的中间搜寻到目标的概率最大，所以我们首先让两批船同时从搜索区域中间行进，到达终点之后分别向左右两侧转弯，以最快的速度覆盖完所有的小正方形。

简化假设该题，计算飞机和轮船的平均搜索能力，以成本当成目标函数，以搜救工具的数量、搜救面积、搜寻时间等因素当做约束条件，构建非线规划模型，运用 lingo 软件求得最优搜寻方案。

关键词： 微分方程 受力分析 平行扫视搜寻法 哈密尔顿图 非线性规划 图论 MATLAB 计算

一　问题重述

回忆失联的的马航 MH370，建立一个通用的数学模型，可以帮助“搜索者”规划一个有用方案，寻找从 A 点飞到 B 点可能坠毁在开放水域如大西洋、太平洋、印度洋，南大洋，或北冰洋的失联飞机。假设坠落的飞机没有信号。你的模型应该认识到,因为有许多不同类型的飞机,所以我们可能会有许多不同类型的搜索和通常使用不同电子或传感器搜索飞机。另外，准备 1-2 页的非技术的发言稿，为航空公司在他们的新闻发布会上关于他们的未来搜索计划进行演讲。

二　问题分析

飞机失去动力后，会受到重力、空气对机翼的升力和飞机下降时的阻力的影响，考虑到空气的阻力与空气的密度有关，为了方便计算，选取空气密度的平均值作为空气的密度。飞机在坠落过程中，阻力和升力的大小仅与飞机下落速度的二次方有关，在水平与竖直方向上受力分解，求出水平位移和竖直位移与时间的关系，用 Matlab 就能绘制出飞机的坠落轨迹，能够比较精确地确定飞机的落水点。

确定落水区域之后，通常将搜救船与搜救飞机一字排开，进行横向搜索，形成一条搜索带。我们根据实际搜寻半径，将搜救区域划分为若干个小正方形，并绘制出相应的网格图，建立了以哈密尔顿图为目标的图论模型。另外我们考虑到在搜索区域的中间搜寻到目标的概率最大，所以我们让两批搜救队伍同时从搜索区域中间行进，到达搜索终点之后分别向左右两侧转弯，以最快的速度覆盖完所有的区域。

简化假设该题，计算飞机和轮船的平均搜索能力，以成本当成目标函数，以搜救工具的数量、搜救面积、搜寻时间等因素当做约束条件，构建非线规划模型。

三　模型假设

1、假设飞机在坠落过程中保持完整，不发生解体。

2、飞机的坠落方向与其原飞行方向保持一致，不发生偏移。

3、假设气流与洋流是均匀的。

4、假设飞机在坠落过程中重力加速度不变。

5、飞机在坠落过程中不考虑风力。

四　符号说明

F_w	飞机所受的空气的总作用力

F_u	飞机所受空气作用力竖直向上的分力，即升力
f	飞机所受空气作用力水平的分力，即阻力
Mg	飞机的重力
ρ_H	海拔高度为 H 时的空气密度
ρ_0	标准状态下的空气密度
α	空气温度梯度
H	海拔高度
T_0	绝对温度，为 273K
ρ	大气的密度的平均值
C_D	阻力系数
C_u	空气升力系数
C_w	空气阻力系数
S_1	飞机翼展面积
S_2	飞机底面面积
a_x	飞机坠落时的水平加速度
a_y	飞机坠落时的竖直加速度
v_x	飞机坠落海面时的水平分速度
v_y	飞机坠落海面时的竖直分速度
t	飞机从失事到坠落海面的时间
x	飞机在坠落海面时的横坐标位置
y	飞机在坠落海面时的纵坐标位置

五　模型建立与求解

5.1 飞机落地区域的模型的建立与求解

飞机在发动机失效之后，在初速度的影响下，受惯性的影响坠落。在此时对飞机进行受力分析。这时飞机的受力有重力 Mg,空气对飞机的作用力F_w,其中F_w又可分解为竖直向上的升力F_w和水平向右的阻力 f。如图 5-1 所示

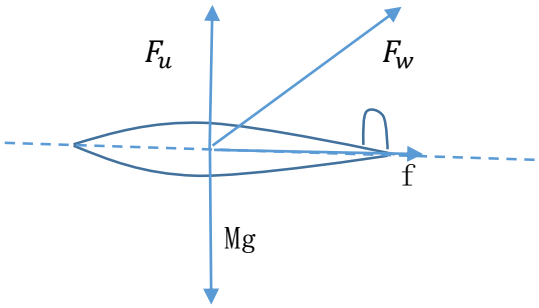

图 5-1

其中，飞机的升力F_u计算公式

$$F_u = \frac{1}{2}\rho C_u S_1 V^2 \qquad (5\text{-}1)$$

其中ρ为大气的密度，C_u为空气升力系数，S_1为飞机翼展面积，V 为飞机初速度。

飞机的阻力 f 的计算公式为

$$\mathrm{f} = \frac{1}{2}\rho C_w S_2 \mathrm{V}^2 \qquad (5\text{-}2)$$

其中C_W为空气阻力系数，S_2为飞机底面面积。

为了计算大气密度ρ，我们查阅相关文献得到了以下数据

海拔高度(m)	0	1000	2000	2500	3000	4000	5000
相对空气密度	1	0.903	0.813	0.770	0.730	0.653	0.583

表 5-1

由表 5-1 可知，在标准状态下大气压力为 1，相对空气密度为 1，一般标准状况（273K,101Kpa）下，空气密度为 1.293g/L，根据气体状态方程式求得空气密度与海拔的高度的关系为

$$\rho_H = (1 - \alpha H/T_0)^{4.26} \qquad (5\text{-}3)$$

（5-3）式中ρ_H是海拔高度为 H 时的空气密度；ρ_0标准状态下的空气密度；α是空气温度梯度，约为 0.0065K/M；H 海拔高度，单位 m；T_0是绝对温度，为 273K。

将相关数据代入，可以得到空气密度与高度的关系曲线如下图

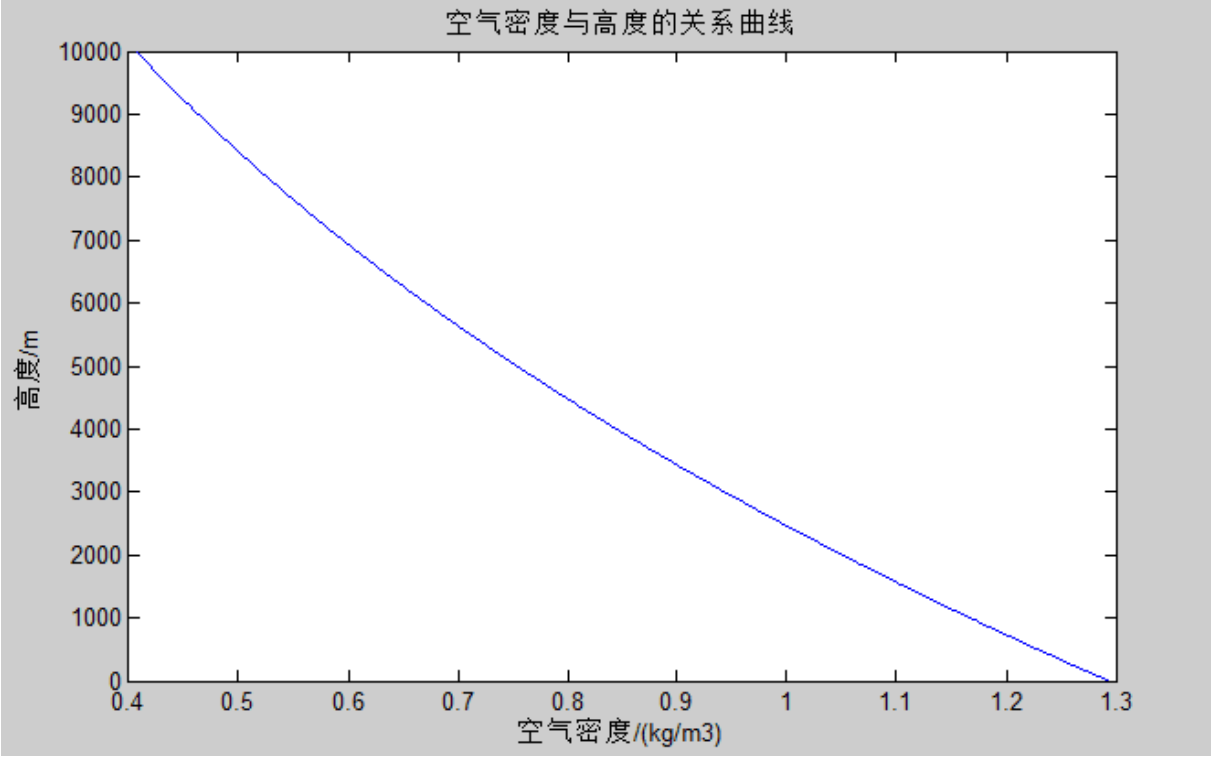

图 5-2

通过计算可得到当高度为 10000m 时，空气的密度为 574.68g/m^3。

由于该曲线近似可看成线性关系，所以我们取平均值作为大气的密度

$$\rho=\frac{1293+574.68}{2}=849\,g/m^3=0.849kg/m^3 \qquad (5\text{-}4)$$

建立平面直角坐标系，以飞机正对的海平面的一点作为坐标系的原点，以飞机的飞行方向作为 X 轴，以原点的竖直方向作为 Y 轴，可将将飞机的加速度分解为 X 方向的加速度a_x和 Y 方向的加速度a_y，根据图 5-1 的飞机的受力分析图可以得到如下等式：

$$f=Ma_x \qquad (5\text{-}5)$$

$$Mg-F_u=Ma_y \qquad (5\text{-}6)$$

其中根据微分方程可以得到a_x、a_y和 V 的表达式，如下所示：

$$a_x=\frac{d^2x}{dt^2} \qquad (5\text{-}7)$$

$$a_y=\frac{d^2y}{dt^2} \qquad (5\text{-}8)$$

综合（5-1）、（5-2）、（5-5）、（5-6）、（5-7）、（5-8）等式可以得到如下公式：

$$\frac{1}{2}\rho C_w S_1\left(\frac{dx}{dt}\right)^2=M\frac{d^2x}{dt^2} \qquad (5\text{-}9)$$

$$\frac{1}{2}\rho C_u S_2(\frac{dy}{dt})^2 = Mg - M\frac{d^2y}{dt^2} \qquad (5\text{-}10)$$

我们查阅马航 MH370 的型号波音 777 的相关资料，估计飞机的重量约为 200000kg，翼展面积 s1 约为 130m^2，机翼受力面积 s2 为 200m^2，飞机空气阻力系数 C_w为 0.08，空气升力系数C_u为 1.2。将初始条件 $\frac{dx}{dt}|_{x=0}$=240m/s、$\frac{dy}{dt}|_{x=0}$=0、y(0)=10000、x(0)=0 代入（5-9）、（5-10）之中，使用 Matlab 进行计算，最终可以得到飞机坠落的轨迹，如下图所示：

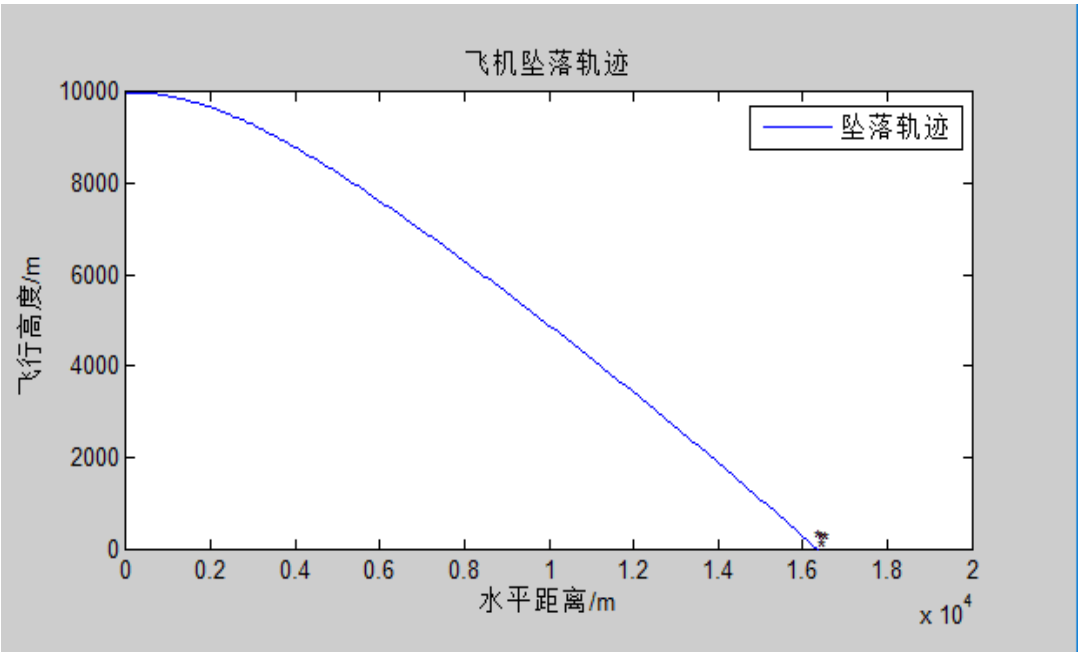

图 5-3

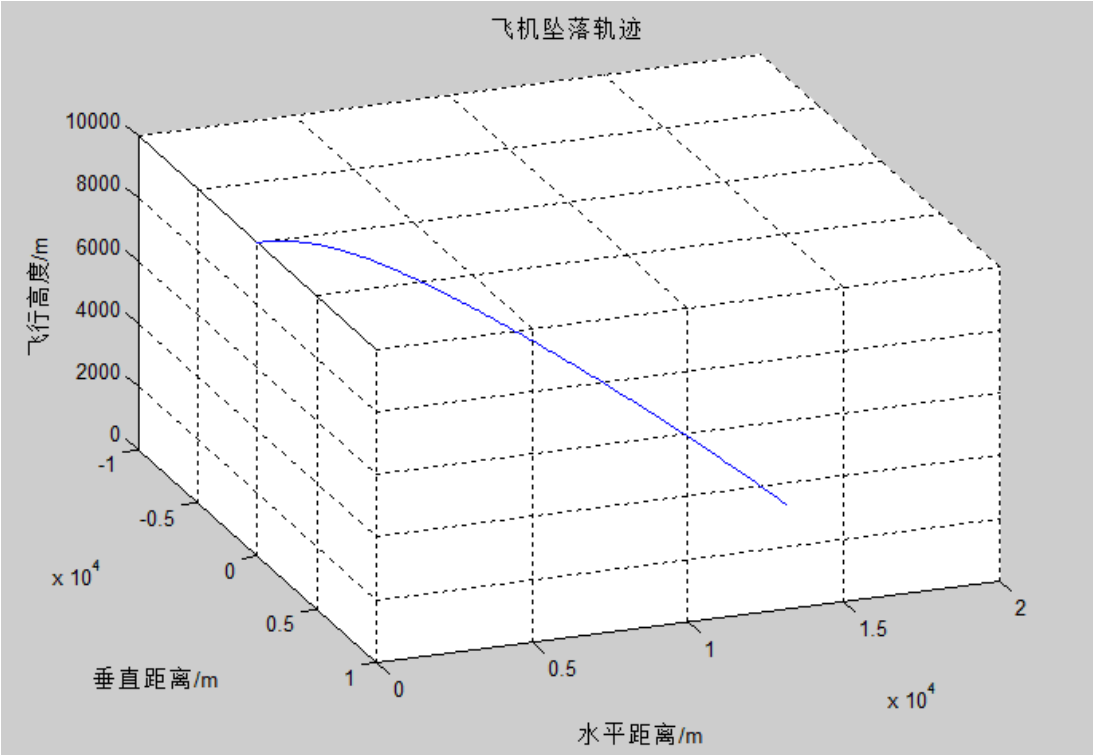

图 5-4

其中飞机从失事到坠落海面的时间 t 为 81.9071s，飞机的位置 x=16328m, y =0m，则坐标（16328,0）为飞机的理论落点，飞机坠落海面时水平速度v_x 为 167.3729m/s，竖直速度v_y为 138.6997m/s，其中v_y的方向沿着纵坐标负方向。经查阅相关资料可以得到，飞机坠落海面的速度不会使飞机发生解体，假设 1 成立。

5.2 搜救方案的模型的建立与求解

在 5.1 中我们确定了飞机的理论落点，但是在飞机失联之后仍有可能向前飞行一段距离，所以其理论落点就有可能沿着飞机的原飞行方向向前平移，取搜索区域为以理论落点为圆心且半径为 20km 的圆，则其可能的搜索区域为圆心在一条直线上的圆的集合，示意图如下:

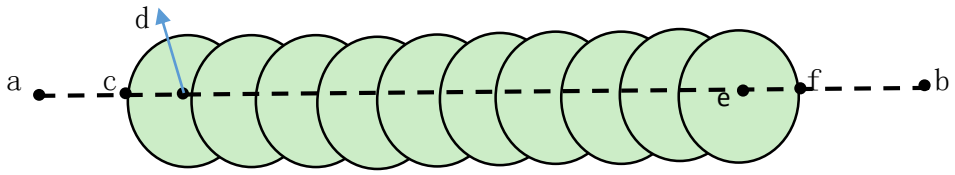

图 5-5

图 5-5 中，a 到 b 为飞机的飞行方向，c 为飞机的失联地点，d 为飞机理论落点，e 为理论落点最远距离，f 为搜救队所要搜寻在最远的距离。

为了问题方便解决，我们近似地将此区域处理为一块为长为 240km，宽为 40km 的长方形区域,如下图所示:

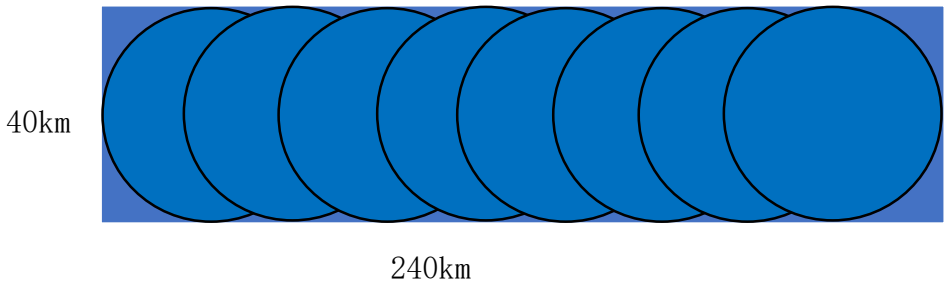

图 5-6

为了使搜救方案更科学合理，通过查找了相关海上搜救方法的资料，比较之下，我们选择了平行扫视搜寻法，因为这种搜救方法适用于搜寻目标位置很不确定并要求均匀覆盖一个广大区域的情况，而本题目中失联飞机的具体坠落地点不确定而且范围广，与平行扫视搜寻法的适用范围较吻合。

平行搜寻法的具体实施方法如下：

当执行平行扫视搜寻时，先设定一搜寻区域，并根据现场情况确定搜寻线间距，搜寻设施把搜寻区域的一角作为搜寻起始点，然后沿矩形长边来回保持间距搜寻。示意图如下：

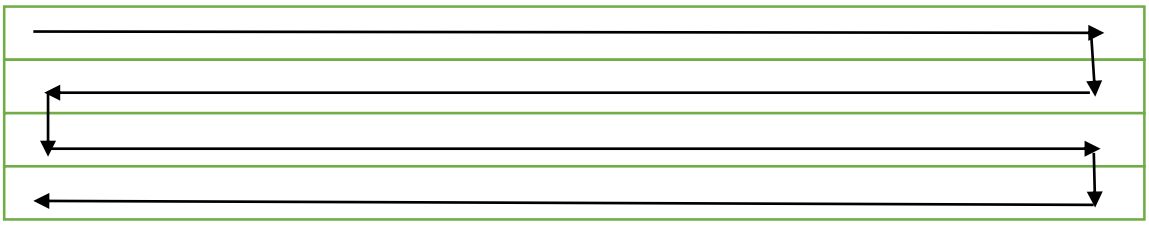

图 5-7

图 5-6 之中，方框代表搜寻区域，箭头代表搜救船或飞机的搜寻方向。

经仔细分析，我们发现“平行扫视搜寻法”并不能直接解决问题，而且不能保证其能在最快的时间内找到失联飞机，故对搜寻方法进行改进，具体改进过程如下：

我们知道搜索区域一定，船和飞机的速度一定，故要使搜救时间最短，就要扩大搜索半径，减少搜救路程。关于扩大搜索半径，我们想到了将 n 条搜救船或飞机一字排开，并且平均分为两批。则其在单位时间内扫过的面积最大。接下来考虑减少路

程，根据搜寻船的排列方式，我们可以将矩形搜救区域划分为若干个小正方形，这样，搜寻问题就转化为怎么在最短的时间内把所有的小正方形遍历完毕。

由于搜寻船和飞机的速度是确定的，所以只要使搜救路线最短即可。运用图论的知识，我们将每个小正方形的中心看作一个点，最短搜救路线就转化为了求经过所有点的最小哈密尔顿图。哈密尔顿图就是通过图 G 的每个结点一次且仅一次的通路。但考虑的每个小正方大小一样，故得到的所有哈密尔顿图大小也一样，即路程一样。通过进一步分析和查找相关资料，为使时间最短，那就要减少搜救路线的拐角数，这就要求搜救船以直线行走。由于失联飞机在搜寻区域中间的概率最大，所以两批搜救队伍最开始从搜救区域中间同时开始行进，等搜救队伍到终点时再分头行动，分别向左右两侧转向，继续搜寻。根据上述原则，我们得到了最终的搜救路线图，如下图所示：

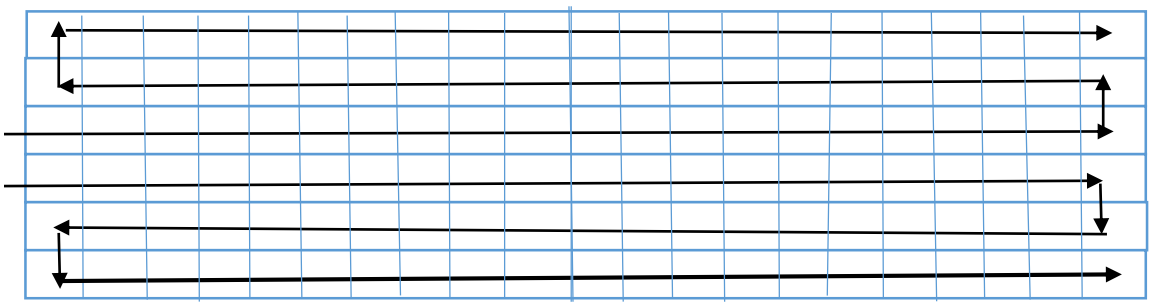

图 5-8

5.3 搜救成本的模型建立与求解

由于本模型需要求出搜救成本的最小值，而对于搜救成本又有多种因素的制约，所以想到使用非线性规划对搜救成本进行求解。

我们仅假设现有的搜救力量只有船和飞机，且飞机和船都可统一调度。经查阅资料，我们假设能提供的飞机有 10 架，船有 40 艘，搜救面积为 9600km^2,以它们作为约束变量，建立以总成本为目标函数的非线性规划模型，求最小成本。

为便于解决问题，我们设飞机的搜寻能力为 600km^2/h，船搜寻能力 150km^2/h，由于黑匣子的信号发射时间是 30 天，所以搜索时间小于 10 天，为扩大搜索提供时间，飞机需要补给，所以每 3 架飞机至少配备一艘船，将其作为标准。设需要调用的船 x 艘，飞机 y 架，总成本为 Z，由非线性规划的相关知识可建立数学模型：

目标函数：　Z=16000*x*c+6000*y*c

约束条件：
$$\begin{cases} 0 \le x \le 10 \\ 0 \le y \le 40 \\ a*c*x+b*c*y \ge S \\ c \le 10*24 \\ y \ge x/3 \end{cases}$$

用 lingo 软件进行求解，运行结果如下：（具体程序见附录）

```
Local optimal solution found.
Objective value:                              270000.0
Extended solver steps:                             31
Total solver iterations:                         1043

                     Variable           Value        Reduced Cost
                            C        5.000000            53999.98
                            X        3.000000            89999.98
                            Y        1.000000            0.000000
                            A        600.0000            0.000000
                            B        150.0000            0.000000
                            S        9600.000            0.000000

                          Row    Slack or Surplus      Dual Price
                            1        270000.0           -1.000000
                            2        3.000000            0.000000
                            3        7.000000            0.000000
                            4        1.000000            0.000000
                            5        39.00000            0.000000
                            6        150.0000            0.000000
                            7        235.0000            0.000000
                            8        0.000000           -29999.99
                            9        0.000000            0.000000
```

图 5-9

由图 5-9 得出用 3 艘船，1 架飞机，搜救时间为 5 小时，搜救的最小成本为 270000 美元。

六　模型检验与评价

6.1 搜寻区域确立模型的优缺点

6.1.1 搜寻区域确立模型的优点

本模型简单易于分析，通过合理的假设，关注主要因素，忽略不必要因素的影响，能够大致推测出飞机失去动力后的运动轨迹以及坠落水中的运动轨迹，基本确定飞机残骸的最后坠落位置，较为符合实际，能够应用于实际搜索。

6.1.2 搜寻区域确立模型的缺点

缺点是忽略了实际飞机在下降过程中，风速、天气的影响，考虑的因素较少，与实际坠落位置会有偏差。本模型对于飞机的坠落过程也过于理想化，如飞机在坠落过程中始终保持在一个平面内，不发生偏移，而在实际情况下，这是很难做到的。

6.2 搜寻方式模型的优缺点

6.2.1 搜寻方式模型的优点

在该模型中，我们在理论和细节方面把握的非常到位，在理论方面我们运用了哈密尔顿回路的原理，将飞机搜寻问题简化为求哈密尔顿圈的最优解，并联合实际求出在将区域全部扫描完的基础上的最短路线。在细节方面我们考虑到从搜寻区域的中间搜寻到飞机的概率最大，所以我们让船与飞机分为两批，先从中间搜寻，接着分别向左右两侧搜寻。

6.2.2 搜寻方式模型的缺点

在考虑问题时我们并未考虑搜救中心到搜救地点的距离以及各搜救船只和飞机的搜索能力的不同等现实因素。

6.3 搜救成本非线性规划模型的优缺点

6.3.1 搜救成本非线性规划模型的优点

利用已知在条件可以计算出在能解决问题在前提下所使用在最小代价，为国家节省大量的人力与物力。

6.3.2 搜救成本非线性规划模型的缺点

在很多情况下，搜救失事飞机是不计成本的，一味的追求低成本可能会导致搜救时间上的浪费。

七　模型改进与推广

由于在上述模型中，在飞机坠落海面时，把它视为静止的，没有考虑风力和洋流对于飞机的作用力，而在实际情况下，是存在这些作用力的，这些作用力会导致飞机在海面上发生漂移作用。

为了计算飞机的漂移距离，我们想到在仅考虑风和洋流对漂移影响的前提下，首先使用 FVCOM 模型对流场进行数值预报，利用经验公式的方法研究风与风致漂流的关系，利用蒙特卡洛算法经过多次模拟计算得出搜寻区域以及搜寻区域随时间的演变。实际案例验证表明，该模型计算快速、准确、操作性较好，能够为搜救工作提供更加准确可信的依据。

八　参考资料

[1] 姜启源，谢金星，叶俊 《数学建模》 北京 高等教育出版社 2003.08

[2]《优化建模 LINDO/LINGO 软件》，谢金星　薛毅，清华大学出版社 2006.09

[3]《图论》，作者王树禾，科学出版社　2010.08

[4]《网络优化》，谢金星 邢文训，清华大学出版社　2007.03

[5]《数学模型》，蒋明 谢金星，高等教育出版社　2006.05

[6]《matlab 与数学实验》王兵团 李桂婷　中国铁道出版社　2014.08

附录：

附录 1：空气密度与高度关系图以及飞机坠落海面的速度的 MATLAB 代码

```
y=dsolve('200000*9.8-101.88*(Dy)^2=200000*D2y','Dy(0)=0','y(0)=10000','t')
y =
(pi*2500000*i)/2547 - (2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (2500000*log(2547/49000000))/2547 + 10000
x=dsolve('200000*D2x+4.4148*(Dx)^2=0','x(0)=0','Dx(0)=240','t')
x =
(500000000*log((33111*t)/6250000 + 1))/11037
syms t;
f=sym((500000000*log((33111*t)/6250000 + 1))/11037);
finverse(f)
ans =
(6250000*exp((11037*t)/500000000))/33111 - 6250000/33111
ezplot('(500000000*log((33111*t)/6250000 + 1))/11037','(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000',[0,86])
 axis equal;
 axis([0,20000,0,10000])
 y=-(43980465111040000*log((2*exp(44439596277231 2751/2748779069440000000 - (863554413089409201*i)/549755813888000000) - ((2547*exp((2547*0)/2500000))/12250000 +
```

(673450872012799983*exp(444395962772312751/1374389534720000000 - (863554413089409201*i)/274877906944000000))/168362718003200000)^(1/2))/(2*exp(444395962772312751/2748779069440000000 - (863554413089409201*i)/549755813888000000) + ((2547*exp((2547*0)/2500000))/12250000 + (673450872012799983*exp(444395962772312751/1374389534720000000 - (863554413089409201*i)/274877906944000000))/168362718003200000)^(1/2))))/6214873907066283

```
y =

   81.9071 + 0.0000i

 x=(500000000*log((33111*81.9071)/6250000 + 1))/11037

x =

    1.6328e+04

 syms x;

 y=(500000000*log((33111*x)/6250000 + 1))/11037;

 diff(y)

ans =

240/((33111*x)/6250000 + 1)

 240/((33111*16328)/6250000 + 1)

ans =

     2.7428

 syms t;

 y=(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000;

 diff(y)

ans =

-(245000000000000*tanh((6214873907066283*t)/87960930222080000)*((6214873907066283*tanh((6214873907066283*t)/87960930222080000)^2)/87960930222080000 - 6214873907066283/87960930222080000))/(6487209*((49000000*tanh((6214873907066283*t)/8796093022208000)^2)/2547 - 1322047255619945/68719476736))
```

-

(245000000000000*tanh((6214873907066283*81.9071)/87960930222080000)*((6214873907066283*tanh((6214873907066283*81.9071)/87960930222080000)^2)/87960930222080000 - 6214873907066283/87960930222080000))/(6487209*((49000000*tanh((6214873907066283*81.9071)/87960930222080000)^2)/2547 - 1322047255619945/68719476736))

ans =

-138.6997

240/((33111*81.9071)/6250000 + 1)

ans =

167.3729

附录二：搜救最小成本的 LINGO 计算代码

```
min=(16000*c*x+6000*2*c*y);
        x>=0;
        x<=10;
        y>=0;
        y<=40;
        c*a*x+b*2*c*y>=s;
        c<=120;
        y>=x/3;
        a=600;
        b=150;
        s=9600;
        @gin(x);
        @gin(y);
        @gin(c);
```

附录三：飞机坠落轨迹的 MATLAB 代码

```
ezplot('(500000000*log((33111*t)/6250000 + 1))/11037','(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000',[0,86])
ezplot3('(500000000*log((33111*t)/6250000 + 1))/11037','(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000','0',[0,86])
ezplot3('(500000000*log((33111*t)/6250000 + 1))/11037','(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000','0',[0,86])
ezplot3('0','(500000000*log((33111*t)/6250000 + 1))/11037','(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000',[0,86])
ezplot3('0','(500000000*log((33111*t)/6250000 + 1))/11037','(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000')
ezplot3('(500000000*log((33111*t)/6250000 + 1))/11037','0','(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000')
ezplot3('(500000000*log((33111*t)/6250000 + 1))/11037','0','(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000')
ezplot3('(500000000*log((33111*t)/6250000 + 1))/11037','0','(2500000*log((49000000*tanh((21*283^(1/2)*t)/5000)^2)/2547 - 49000000/2547))/2547 - (pi*2500000*i)/2547 + (2500000*log(2547/49000000))/2547 + 10000',[0,86])
axis equal;
axis([0,20000,0,10000])
axis([0,20000,0,10000,-100,100])
axis([0,20000,0,10000,-10000,10000])
axis([0,20000,-10000,10000,0,10000,])
```