

```

1  !-----!
2  ! 6Dof彈道模擬(6 Degreeed of freedom trajectory simulation) !
3  ! 受控體(plant):無控火箭 !
4  ! 作者:陳維霖 台大機械博士班 學號:D09522006 !
5  ! date:2021/1/16 !
6  ! version:1 !
7  ! 輸入檔:(1)input_1_thrust !
8  !          (2)input_2_physical !
9  !          (3)input_3_aero !
10 !          (3-1-1)input_3_aero_1_1_CA_ON !
11 !          (3-1-2)input_3_aero_1_2_CA_OFF !
12 !          (3-1-3)input_3_aero_1_3_CN !
13 !          (3-1-4)input_3_aero_1_4_XCP !
14 !          (3-1-5)input_3_aero_1_5_CLFI !
15 !          (3-1-6)input_3_aero_1_6_CHMAL !
16 !          (3-1-7)input_3_aero_1_7_CHMAW !
17 !          (3-1-8)input_3_aero_1_8_CNFAL !
18 !          (3-1-9)input_3_aero_1_9_CNFAW !
19 !          (3-2)input_3_aero_2_reference !
20 !          (3-3)input_3_aero_3_other !
21 !          (4)input_4_atmosphere !
22 ! 輸出檔:(1)output_6D_1 !
23 !          (2)output_6D_2 !
24 !          (3)output_6D_3 !
25 !          (4)output_6D_4 !
26 !          (5)output_6D_5 !
27 !          (6)output_check_1_thrust !
28 !          (7)output_check_2_physical !
29 !          (8)output_check_3_aero !
30 !          (9)output_check_4_atmospher !
31 !          (10)output_debug_1_atmosphere !
32 !          (11)output_debug_2_physical !
33 !          (12)output_debug_3_thrust !
34 !          (13)output_debug_4_airforce_1 !
35 !          (14)output_debug_5_airforce_2 !
36 !          (15)output_debug_6_FM_ !
37 !-----!
38
39 !-----!
40 ! 建模 for 常數、變數 !
41 !-----!
42 module constant
43     implicit none
44     real*8, parameter :: PI=3.1415927410125732421875
45     real*8, parameter :: g=9.80665
46 endmodule
47
48 module global
49     implicit none
50     real*8, parameter :: dt=0.01
51     real*8, save :: t,factor
52     integer, save :: counter,i_main,call_times
53 endmodule
54
55 module module_atmosphere
56     implicit none
57     real*8, save :: P_air,rho,Cs
58 endmodule
59
60 module solution
61     implicit none
62     real*8, save :: Y(13),dY_dt(13)
63     real*8, save :: position(3),velocity(3),angular_velocity(3),Euler_angle_GD(3),mass
64     real*8, save :: angular_velocity_rate(3),Euler_angle_rate_GD(3)
65     real*8, save :: velocity_D(3)
66     real*8, save :: AF,phi_c,apha,V,Ma
67 endmodule
68
69 module input_data
70     implicit none
71     real*8 input_1_thrust(100,100)
72     real*8 input_2_physical(10,10)
73     real*8 input_3_aero_1(11,10,9),input_3_aero_2(4),input_3_aero_3(13,9)

```

```

74     real*8 input_4_atmosphere(22,6)
75 endmodule
76
77 module unit
78     implicit none
79     !副程式:讀取輸入資料(input)
80     integer, parameter :: unit_thrust=10,unit_thrust_check=15
81     integer, parameter :: unit_physical=20,unit_physical_check=25
82     integer, parameter :: unit_aero_3_1=30,unit_aero_3_2=45,unit_aero_3_3=50
83     integer, parameter :: unit_aero_check=55
84     !副程式:初始資料計算(initial)
85     integer, parameter :: unit_thrust_interpolation_check=65
86     integer, parameter :: unit_physical_interpolation_check=70
87     !副程式:標準大氣程式(atmosphere)
88     integer, parameter ::
unit atmosphere=75,unit_atmosphere_check=80,unit_atmosphere_debug=85
89     !副程式:物性、推力參數
90     integer, parameter :: unit_out_physical_debug=90
91     !副程式:推力(彈體座標系D)
92     integer, parameter :: unit_output_thrust_debug=95
93     !副程式:總外力及力矩(彈體座標系D->大地座標系G)
94     integer, parameter :: unit_F_M_debug=100
95     !副程式:輸出計算結果
96     integer, parameter :: unit_output_6D_1=150,unit_output_6D_2=200,&
97                             &unit_output_6D_3=250,unit_output_6D_4=300,unit_output_6D_5=350
98 endmodule
99 !-----!
100 !                                     主程式                                     !
101 !-----!
102 Program main
103     use global
104     use input_data
105     use solution
106     implicit none
107     integer i
108     real*8 timepoint(5,5)
109     write(*,*) "-----Main Program Start-----"
110
111     !-----讀取輸入資料(input)-----!
112     !呼叫副程式:讀取輸入資料(input)
113     call sub_input()
114     !-----初始資料計算(initial)-----!
115     !呼叫副程式:初始資料計算(initial)
116     call sub_initial()
117
118     !各種計數器
119     call_times = 0
120     i_main = 0
121
122     !-----6Dof計算迴圈-----!
123     do i_main=1,100000
124         t = (i_main-1)*dt
125         !-----螢幕輸出-----!
126         if(mod(i_main-1,100)==0 .or. i_main==1 .or. i_main==counter) then
127             if(i_main==1) then
128                 write(*,*) "-----6Dof iteration start-----"
129
130                 write(*,200) "t", "X(m)", "Y(m)", "Z(m)"
131                 endif
132                 write(*,210) t, (Y(i), i=4,6)
133                 format(a16,3a16)
134                 format(f16.4,3f16.2)
135             endif
136             !-----輸出計算結果-----!
137             !呼叫副程式:輸出計算結果
138             call sub_output(timepoint)
139
140             !-----6Dof方程式-----!
141             !呼叫副程式:6Dof方程式(6Dof)
142             call sub_6Dof()
143
144             !-----數值積分(Runge Kutta Method)-----!
145             !呼叫副程式:數值積分(Runge Kutta Method)

```

```

146     call sub_integral()
147
148     if(-Y(6)<0.0) then
149         write(*,*) "Height(m)<0:", -Y(6)
150         exit
151     endif
152 enddo
153 write(*,*) "-----Main Program End-----"
154
155 End Program
156 !-----
157 ! 副程式:讀取輸入資料(input)
158 ! 輸入:
159 ! 輸出:input_1_thrust
160 !       input_2_physical
161 !       input_3_aero_1
162 !       input_3_aero_2
163 !       input_3_aero_3
164 ! 定義:input_1_thrust:推力
165 !       input_2_physical:物性
166 !       input_3_aero_1:氣動力係數(二維)
167 !       input_3_aero_2:氣動力參考長度,面積
168 !       input_3_aero_3:氣動力係數(一維)
169 !       D3:矩陣個數
170 !       row:矩陣列數
171 !       column:矩陣行數
172 !-----
173 subroutine sub_input()
174     use input_data
175     use unit
176     integer i,j,k
177     integer row,column,d3
178     integer status
179
180     write(*,*) "-----subroutine:read input data-----"
181
182     ! (1) 讀取資料-推力
183     row=26
184     column=5
185     Open(unit=unit_thrust,file='input_1_thrust.txt',status='old')
186     read(unit_thrust,*)
187     read(unit_thrust,*,iostat=status) ((input_1_thrust(i,j),j=1,column),i=1,row)
188     write(*,*) "thrust data,read result(0:success,other:fail):",status
189     close(unit_thrust)
190
191     Open(unit=unit_thrust_check,file='output_check_1_thrust.txt',status='unknown')
192     write(unit_thrust_check,"(5A16)")
193     write(unit_thrust_check,"(5f16.2)") ((input_1_thrust(i,j),j=1,column),i=1,row)
194     close(unit_thrust_check)
195
196     ! (2) 讀取資料-物性
197     row=2
198     column=4
199     Open(unit=unit_physical,file='input_2_physical.txt',status='old')
200     read(unit_physical,*)
201     read(unit_physical,*,iostat=status) ((input_2_physical(i,j),j=1,column),i=1,row)
202     write(*,*) "physical data,read result(0:success,other:fail):",status
203     close(unit_physical)
204
205     Open(unit=unit_physical_check,file='output_check_2_physical.txt',status='unknown')
206     write(unit_physical_check,"(5A16)") "m(kg)", "x_cg(m)", "Ixx(kg-m^2)", "Iyy(kg-m^2)"
207     write(unit_physical_check,"(4f16.2)") ((input_2_physical(i,j),j=1,column),i=1,row)
208     close(unit_physical_check)
209
210     ! (3) 讀取資料-氣動力係數
211     row=11
212     column=10
213     d3=9
214     !3-1
215     !1-1:CA(ON)
216     k=1
217     Open(unit=unit_aero_3_1,file='input_3_aero_1_1.txt',status='old')

```

```

218     read(unit_aero_3_1,*)
219     read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
220     write(*,*)"aero data=",k,"",read result(0:success,other:fail):",status
221 close(unit_aero_3_1)
222
223 !1-2:CA(OFF)
224 k=2
225 Open(unit=unit_aero_3_1,file='input_3_aero_1_2.txt',status='old')
226     read(unit_aero_3_1,*)
227     read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
228     write(*,*)"aero data=",k,"",read result(0:success,other:fail):",status
229 close(unit_aero_3_1)
230
231 !1-3:CN
232 k=3
233 Open(unit=unit_aero_3_1,file='input_3_aero_1_3.txt',status='old')
234     read(unit_aero_3_1,*)
235     read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
236     write(*,*)"aero data=",k,"",read result(0:success,other:fail):",status
237 close(unit_aero_3_1)
238
239 !1-4:XCP
240 k=4
241 Open(unit=unit_aero_3_1,file='input_3_aero_1_4.txt',status='old')
242     read(unit_aero_3_1,*)
243     read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
244     write(*,*)"aero data=",k,"",read result(0:success,other:fail):",status
245 close(unit_aero_3_1)
246
247 !1-5:CLFI(22.5)
248 k=5
249 Open(unit=unit_aero_3_1,file='input_3_aero_1_5.txt',status='old')
250     read(unit_aero_3_1,*)
251     read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
252     write(*,*)"aero data=",k,"",read result(0:success,other:fail):",status
253 close(unit_aero_3_1)
254
255 !1-6:CHMAL
256 k=6
257 Open(unit=unit_aero_3_1,file='input_3_aero_1_6.txt',status='old')
258     read(unit_aero_3_1,*)
259     read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
260     write(*,*)"aero data=",k,"",read result(0:success,other:fail):",status
261 close(unit_aero_3_1)
262
263 !1-7:CHMAW
264 k=7
265 Open(unit=unit_aero_3_1,file='input_3_aero_1_7.txt',status='old')
266     read(unit_aero_3_1,*)
267     read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
268     write(*,*)"aero data=",k,"",read result(0:success,other:fail):",status
269 close(unit_aero_3_1)
270
271 !1-8:CNFAL
272 k=8
273 Open(unit=unit_aero_3_1,file='input_3_aero_1_8.txt',status='old')
274     read(unit_aero_3_1,*)
275     read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
276     write(*,*)"aero data=",k,"",read result(0:success,other:fail):",status
277 close(unit_aero_3_1)
278
279 !1-9:CNFAW
280 k=9
281 Open(unit=unit_aero_3_1,file='input_3_aero_1_9.txt',status='old')
282     read(unit_aero_3_1,*)
283     read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
284     write(*,*)"aero data=",k,"",read result(0:success,other:fail):",status
285 close(unit_aero_3_1)
286
287 !3-2
288 k=10
289 Open(unit=unit_aero_3_2,file='input_3_aero_2.txt',status='old')
290     read(unit_aero_3_2,"(9x,f16.8)",iostat=status) (input_3_aero_2(i),i=1,4)

```

```

291     write(*,*) "aero data=",k,"",read result(0:success,other:fail):",status
292 close(unit_aero_3_2)
293
294 !3-3
295 row=13
296 column=9
297 k=11
298 Open(unit=unit_aero_3_3,file='input_3_aero_3.txt',status='old')
299 read(unit_aero_3_3,*,iostat=status) ((input_3_aero_3(i,j),j=1,column),i=1,row)
300 write(*,*) "aero data=",k,"",read result(0:success,other:fail):",status
301 close(unit_aero_3_3)
302
303 !檢查氣動力輸入資料
304 Open(unit=unit_aero_check,file='output_check_3_aero.txt',status='unknown')
305 !3-1
306 row=11
307 column=10
308 d3=9
309 do k=1,9
310     if(k==1) then
311         write(unit_aero_check,*) k,":CA(ON) "
312     elseif(k==2) then
313         write(unit_aero_check,*) k,":CA(OFF) "
314     elseif(k==3) then
315         write(unit_aero_check,*) k,":CN"
316     elseif(k==4) then
317         write(unit_aero_check,*) k,":XCP"
318     elseif(k==5) then
319         write(unit_aero_check,*) k,":CLFI"
320     elseif(k==6) then
321         write(unit_aero_check,*) k,":CHMAL"
322     elseif(k==7) then
323         write(unit_aero_check,*) k,":CHMAW"
324     elseif(k==8) then
325         write(unit_aero_check,*) k,":CNFAL"
326     elseif(k==9) then
327         write(unit_aero_check,*) k,":CNFAW"
328     endif
329     write(unit_aero_check,100) ("MACH=", (input_3_aero_1(i,j,k),j=1,column),i=1,1)
330     write(unit_aero_check,100) ("AF=", (input_3_aero_1(i,j,k),j=1,column),i=2,row)
331     write(unit_aero_check,"(/)")
332 enddo
333 !3-2
334 write(unit_aero_check,101) "RL(M)=",input_3_aero_2(1)
335 write(unit_aero_check,101) "RA(M^2)=",input_3_aero_2(2)
336 write(unit_aero_check,101) "ALT(KM)=",input_3_aero_2(3)
337 write(unit_aero_check,"(a9,e16.4)") "RE=",input_3_aero_2(4)
338 write(unit_aero_check,"(/)")
339 !3-3
340 row=13
341 column=9
342
343 i=1
344 write(unit_aero_check,102) i,"MACH=", (input_3_aero_3(i,j),j=1,column)
345 i=i+1
346 write(unit_aero_check,102) i,"CNQ=", (input_3_aero_3(i,j),j=1,column)
347 i=i+1
348 write(unit_aero_check,102) i,"CMQ=", (input_3_aero_3(i,j),j=1,column)
349 i=i+1
350 write(unit_aero_check,102) i,"CLP=", (input_3_aero_3(i,j),j=1,column)
351 i=i+1
352 write(unit_aero_check,102) i,"CND=", (input_3_aero_3(i,j),j=1,column)
353 i=i+1
354 write(unit_aero_check,102) i,"XCPD=", (input_3_aero_3(i,j),j=1,column)
355 i=i+1
356 write(unit_aero_check,102) i,"CLDP=", (input_3_aero_3(i,j),j=1,column)
357 i=i+1
358 write(unit_aero_check,102) i,"CHMD=", (input_3_aero_3(i,j),j=1,column)
359 i=i+1
360 write(unit_aero_check,102) i,"CNFD=", (input_3_aero_3(i,j),j=1,column)
361 i=i+1
362 write(unit_aero_check,102) i,"XCPFD=", (input_3_aero_3(i,j),j=1,column)
363 i=i+1

```

```

364 write(unit_aero_check,102) i,"CND2=", (input_3_aero_3(i,j),j=1,column)
365 i=i+1
366 write(unit_aero_check,102) i,"XCPD2=", (input_3_aero_3(i,j),j=1,column)
367 i=i+1
368 write(unit_aero_check,102) i,"CLDP2=", (input_3_aero_3(i,j),j=1,column)
369
370 format(a8,10f8.2)
371 format(a9,f16.8)
372 format(i3,a8,9f8.2)
373
374 close(unit_aero_check)
375 write(*,*) "-----"
376 return
377 end
378 !-----!
379 ! 副程式:初始資料計算(initial) !
380 ! 輸入:input_1_thrust !
381 ! input_2_physical !
382 ! 輸出:position !
383 ! thrust !
384 ! physical !
385 ! dt !
386 ! count !
387 ! 定義:position(1)=x=飛彈在地面座標系g之x軸位置(km) !
388 ! position(2)=y=飛彈在地面座標系g之y軸位置(km) !
389 ! position(3)=z=飛彈在地面座標系g之z軸位置(km) !
390 ! velocity(1)=u=飛彈在地面座標系g之x軸速度(m/s) !
391 ! velocity(2)=v=飛彈在地面座標系g之y軸速度(m/s) !
392 ! velocity(3)=w=飛彈在地面座標系g之z軸速度(m/s) !
393 ! angular_velocity(1)=p=飛彈在地面座標系g之x角速度(rad/s) !
394 ! angular_velocity(2)=q=飛彈在地面座標系g之y角速度(rad/s) !
395 ! angular_velocity(3)=r=飛彈在地面座標系g之z角速度(rad/s) !
396 ! Euler_angle_GD(1)=theta=俯仰角(rad) !
397 ! Euler_angle_GD(2)=phi=偏航角(rad) !
398 ! Euler_angle_GD(3)=gamma=傾斜角(rad) !
399 ! thrust=內插完的推力 !
400 ! physical=內插完的物性 !
401 ! dt=時間步長(s) !
402 ! counter=到燃畢時的計算次數 !
403 !-----!
404 subroutine sub_initial()
405 use global
406 use constant
407 use input_data
408 use solution
409 use unit
410 implicit none
411 integer i,j,k
412 integer row,column
413 real*8 thrust_test(50000,5)
414 real*8 physical_test(50000,4)
415 real*8 t_start,t_end
416 real*8 A_up,A_mid,A_down
417 real*8 B_up,B_mid,B_down
418 real*8 mass_up,mass_down,mass_loss,mass_loss_rate
419 real*8 mass_start,mass_end
420 real*8 L_body,X_CG
421 real*8 t_up,t_down
422
423 write(*,*) "-----subroutine:initial data-----"
424
425 !L_body=彈體長度(m)
426 L_body=3.873
427 !X_CG=彈體重心位置(m)
428 X_CG=2.69
429
430 !初始俯仰角=25度
431 Euler_angle_GD(1)=25.0*PI/180.
432 Euler_angle_GD(2)=0.0
433 Euler_angle_GD(3)=0.0
434
435 position(1)=0.0
436 position(2)=0.0

```

```

437 position(3)=-0.8
438
439 velocity(1)=0.0
440 velocity(2)=0.0
441 velocity(3)=0.0
442
443 angular_velocity(1)=0.0
444 angular_velocity(2)=0.0
445 angular_velocity(3)=0.0
446
447 write(*,"(a4,f10.4,a4)" "dt=",dt,"(s)"
448 write(*,300) "L_body(m)", "X.CG(m)", "L_body-X.CG(m)"
449 write(*,310) L_body,X.CG,L_body-X.CG
450 write(*,300) "theta(deg)", "phi(deg)", "gamma(deg)"
451 write(*,310) (Euler_angle_GD(i),i=1,3)
452 write(*,300) "x(m)", "y(m)", "z(m)"
453 write(*,310) (position(i),i=1,3)
454 write(*,300) "Vx(m/s)", "Vy(m/s)", "Vz(m/s)"
455 write(*,310) (velocity(i),i=1,3)
456 write(*,300) "p(deg/s)", "q(deg/s)", "r(deg/s)"
457 write(*,310) (angular_velocity(i)*PI/180.,i=1,3)
458
459 format(3A20)
460 format(3F20.4)
461 !-----内插-----!
462 write(*,*) "-----interpolation-----"
463 !1.推力内插
464 counter=(input_1_thrust(26,1)-input_1_thrust(1,1))/dt+1
465 write(*,*) "counter:",counter
466 t_start=input_1_thrust(1,1)
467 t_end=input_1_thrust(26,1)
468 row=counter
469 column=5
470 do j=1,5
471     do i=1,row
472         t=t_start+(i-1)*dt
473         if((t-t_start)<1e-5) then
474             thrust_test(i,j)=input_1_thrust(1,j)
475         elseif(abs(t-t_end)<=1e-5) then
476             thrust_test(i,j)=input_1_thrust(26,j)
477         else
478             do k=2,26
479                 t_up=input_1_thrust(k-1,1)
480                 t_down=input_1_thrust(k,1)
481                 if(t>=t_up .and. t<=t_down) then
482                     A_up = t_up
483                     A_mid = t
484                     A_down = t_down
485                     B_up = input_1_thrust(k-1,j)
486                     B_down = input_1_thrust(k,j)
487                     B_mid = B_up + ( (A_mid-A_up)/(A_down-A_up) )*(B_down-B_up)
488                     thrust_test(i,j) = B_mid
489                 endif
490             enddo
491         endif
492     end do
493 end do
494 !輸出確認
495 open(unit=unit_thrust_interpolation_check,file="output_thrust_interpolation.txt")
496 write(unit_thrust_interpolation_check,100)
497 "t(sec)", "p(kgf/cm^2)", "TF_sea(kgf)", "TF_vacuum(kgf)", "dm/dt(kg/sec)"
498 write(unit_thrust_interpolation_check,110) ((thrust_test(i,j),j=1,column),i=1,row)
499 format(5a16)
500 format(5f16.2)
501 close(unit_thrust_interpolation_check)
502
503 !2.物性内插
504 row = counter
505 column = 4
506
507 factor = 1.0
508 do while(.true.)
509     mass_start = input_2_physical(1,1)

```



```

509     mass_end = input_2_physical(2,1)
510     mass_loss = 0.0
511     do i=1,row
512         mass_loss_rate = factor*thrust_test(i,5)
513         mass_loss = mass_loss + mass_loss_rate*dt
514         mass = mass_start - mass_loss
515         physical_test(i,1) = mass
516     end do
517     if( abs(physical_test(row,1)-mass_end) <= 1e-2) then
518         write(*,"(a10,f10.4)") "factor=",factor
519         exit
520     elseif((physical_test(row,1)-mass_end)>0) then
521         factor = factor + (physical_test(row,1)-mass_end)/mass_end
522     else
523         factor = factor + (physical_test(row,1)-mass_end)/mass_end
524     endif
525 end do
526 write(*,300) "mass_start(kg) ", "mass_end_compute", "mass_end"
527 write(*,310) mass_start, mass, mass_end
528 write(*,"(a40,f20.4)") "mass_end_compute-mass_end(kg)=", physical_test(row,1)-mass_end
529
530 do j=2,4
531     physical_test(1,j) = input_2_physical(1,j)
532     physical_test(row,j) = input_2_physical(2,j)
533     mass_up = physical_test(1,1)
534     mass_down = physical_test(row,1)
535     A_up = mass_up
536     A_down = mass_down
537     B_up = input_2_physical(1,j)
538     B_down = input_2_physical(2,j)
539     do i=1,row-1
540         A_mid = physical_test(i,1)
541         B_mid = B_up + ( (A_mid-A_up)/(A_down-A_up) )*(B_down-B_up)
542         physical_test(i,j) = B_mid
543     end do
544 end do
545 !輸出確認
546
547 open(unit=unit_physical_interpolation_check,file="output_physical_interpolation.txt"
548 )
549     write(unit_physical_interpolation_check,200)
550     "m(kg) ", "x_cg(m) ", "Ixx(kg-m^2) ", "Iyy(kg-m^2) "
551
552     write(unit_physical_interpolation_check,210) ((physical_test(i,j),j=1,column),i=1,r
553 ow)
554     format(4a16)
555     format(4f16.2)
556 close(unit_physical_interpolation_check)
557
558 write(*,*) "-----Y initial-----"
559 !設定初始值
560 !積分的變數(dY)共13個:1~3:速度、4~6:位置、7~9:角速度、10~12:尤拉角、13:質量
561 do i=1,3
562     Y(i) = velocity(i)
563     Y(i+3) = position(i)
564     Y(i+6) = angular_velocity(i)
565     Y(i+9) = Euler_angle_GD(i)
566 enddo
567 Y(13) = input_2_physical(1,1)
568 mass = Y(13)
569 write(*,"(i16,f16.2)") (i,Y(i),i=1,13)
570 write(*,*) "-----"
571
572 return
573 end
574 !-----!
575 ! 副程式:6Dof方程式(6Dof) !
576 ! 輸入: !
577 ! 輸出: !
578 ! 定義: !
579 !-----!
580
581 subroutine sub_6Dof()
582     use global

```



```

577 use input_data
578 use solution
579 implicit none
580 integer i
581 real*8 F_gravity(3)
582 real*8 F_thrust(3)
583 real*8 F_airforce(3)
584 real*8 F_total(3)
585 real*8 M_air(3)
586 real*8 M_total(3)
587 real*8 thrust(5)
588 real*8 physical(4)
589
590 M_air=0
591 M_total=0
592 F_airforce=0
593
594 !進入次數
595 call_times = call_times+1
596
597 !積分的變數(dY)共13個:1~3:速度、4~6:位置、7~9:角速度、10~12:尤拉角、13:質量
598 !變數更新
599 do i=1,3
600     velocity(i)          = Y(i)
601     position(i)          = Y(i+3)
602     angular_velocity(i)  = Y(i+6)
603     Euler_angle_GD(i)   = Y(i+9)
604     mass                 = Y(13)
605 end do
606
607 !-----標準大氣模式-----!
608 !呼叫副程式:標準大氣模式
609 call sub_atmosphere()
610
611 !-----物性、推力參數-----!
612 !呼叫副程式:物性、推力參數
613 call sub_parameter(thrust,physical)
614
615 !-----力及力矩-----!
616 !呼叫副程式:重力(彈體座標系D)
617 call sub_F_gravity(F_gravity)
618
619 !呼叫副程式:推力(彈體座標系D)
620 call sub_F_thrust(F_thrust,thrust)
621
622 !呼叫副程式:氣動力及氣動力矩(彈體座標系D)
623 call sub_F_airforce(F_airforce,M_air,physical)
624
625 !呼叫副程式:總外力及力矩(彈體座標系D->大地座標系G)
626 call sub_F_total(F_total,M_total,F_gravity,F_thrust,F_airforce,M_air)
627
628 !呼叫副程式:尤拉角(大地座標系G與彈體座標系D)
629 call sub_Euler_angle()
630
631 !呼叫副程式:角加速度(彈體座標系D)
632 call sub_angular_velocity(M_total,physical)
633
634 !呼叫副程式:微分方程(大地座標系G)
635 call sub_eqs(F_total)
636
637 return
638 end
639 !-----!
640 ! 副程式:標準大氣程式(atmosphere) !
641 ! 輸入:position !
642 ! 輸出:P_air !
643 ! rho !
644 ! Cs !
645 ! 定義:g=9.80665=重力加速度(m/s^2) !
646 ! R_earth=6356.766=地球半徑(km) !
647 ! R=287.05287=比氣體常數(J/(K*kg)) !
648 ! M=28.96442=分子量(kg/kmol) !
649 ! R_start=8314.32=通用氣體常數(J/(K*kg)) !

```

```

650 !      Z=地理高度(geometric altitude) (km) !
651 !      H=位勢高度(geopotential altitude) (km) !
652 !      H_b=相應層底部的位勢高度(km) !
653 !      P_b=相應層底部的大氣壓力(Pa或N/m^2) !
654 !      T_b=相應層底部的大氣溫度(K) !
655 !      M_b=相應層底部的分子量(kg/kmol) !
656 !      Temperature=溫度(K) !
657 !      beta=相應層沿位勢高度的溫度梯度(K/km) !
658 !      P_air=大氣壓力(Pa或N/m^2) !
659 !      rho=大氣密度(kg/m^3) !
660 !      Cs=大氣音速(m/s) !
661 !      position(1)=x=飛彈在地面座標系g之x軸位置(km) !
662 !      position(2)=y=飛彈在地面座標系g之y軸位置(km) !
663 !      position(3)=z=飛彈在地面座標系g之z軸位置(km) !
664 !      備註:參考"U.S. STANDARD ATMOSPHERE, 1962" !
665 !-----!
666 subroutine sub_atmosphere()
667     use global
668     use input_data
669     use module_atmosphere
670     use solution
671     use unit
672     implicit none
673     real*8, parameter ::
674     g=9.80665,R=287.05287,R_start=8314.32,R_earth=6356.766,M=28.96442
675     real*8 Z,H
676     real*8 Temperature
677     real*8 input_Z_b(30),input_H_b(30),input_T_b(30),input_beta(30),input_P_b(30),input_M_b(30)
678     real*8 H_b,T_b,beta,P_b,M_b
679     real*8 beta_m,H_m,H_b_m
680     integer i,j
681     integer row,column
682     integer status
683     ! (1) 讀取大氣溫度參數
684     row = 22
685     column = 6
686     if(i_main==1) then
687         open(unit=unit_atmosphere, file="input_4_atmosphere.txt")
688
689         read(unit_atmosphere,*,iostat=status)((input_4_atmosphere(i,j),j=1,column),i=1,row)
690     ! write(*,*)"大氣資料=",", 讀取結果(0:成功,other:失敗):",status
691     close(unit_atmosphere)
692     endif
693     do i=1,row
694         input_Z_b(i) = input_4_atmosphere(i,1)
695         input_H_b(i) = input_4_atmosphere(i,2)
696         input_T_b(i) = input_4_atmosphere(i,3)
697         input_beta(i) = input_4_atmosphere(i,4)
698         input_P_b(i) = input_4_atmosphere(i,5)
699         input_M_b(i) = input_4_atmosphere(i,6)
700     end do
701     !輸出大氣表
702     if(i_main==1) then
703         open(unit=unit_atmosphere_check,file="output_check_4_atmospher.txt")
704
705         write(unit_atmosphere_check,100)"Z_b(km) ","H_b(km) ","T_b(K) ","beta (K/km) ","P_b(pa)
706         ", "M_b(kg/kmol) "
707
708         write(unit_atmosphere_check,110)(input_Z_b(i),input_H_b(i),input_T_b(i),input_beta
709         (i),input_P_b(i),input_M_b(i),i=1,row)
710         format(6a14)
711         format(4f14.4,e14.4,f14.4)
712         close(unit_atmosphere_check)
713     endif
714
715     ! (2) 計算高度位勢(z為高度，向下為正，因此加-號)
716     Z = -position(3)/1000.0
717     H = (R_earth*Z)/(R_earth+Z)

```

```

715 ! (3) 依高度判斷在哪一層
716 do i=1,row-1
717     if (H>=input_H_b(i) .and. H<input_H_b(i+1)) then
718         H_b = input_H_b(i)
719         T_b = input_T_b(i)
720         beta = input_beta(i)
721         P_b = input_P_b(i)
722         M_b = input_M_b(i)
723     elseif (i==(row-1) .and. H>=input_H_b(row)) then
724         H_b = input_H_b(row)
725         T_b = input_T_b(row)
726         beta = input_beta(row)
727         P_b = input_P_b(row)
728         M_b = input_M_b(row)
729     elseif (H<input_H_b(1)) then
730         ! write(*,*) "位勢高度:", H, "超出範圍 (H<-2km) "
731     endif
732 end do
733
734 ! (4) 溫度
735 Temperature = T_b + beta*(H-H_b)
736
737 ! (5) 壓力
738 ! 將單位由km轉成m
739 beta_m = beta/1000.0
740 H_m = H*1000.0
741 H_b_m = H_b*1000.0
742 if (beta==0.0) then
743     P_air = P_b*exp( (-g/(R*T)) *(H_m-H_b_m) )
744 else
745     P_air = P_b*( 1.0+(beta_m/T_b)*(H_m-H_b_m) )**(-g/(R*beta_m) )
746 endif
747
748 ! (6) 密度
749 rho = P_air/(R*Temperature)
750
751 ! (7) 音速
752 ! k=1.4 為比熱常數 k=Cp/Cv
753 Cs = ( 1.4*(R_start/M)*Temperature )**0.5
754
755 ! (8) 馬赫數
756 V = ( velocity(1)**2 + velocity(2)**2 + velocity(3)**2 )**0.5
757 Ma = V/Cs
758
759 ! check
760 if (.true.) then
761     if (mod(call_times-1,4)==0.0) then
762         if (i_main==1) then
763             open(unit=unit_atmosphere_debug, file="output_debug_1_atmosphere.txt")
764             write(unit_atmosphere_debug,200) "t", "Z(km)", "H(km)", "T(K)", "P_air(pa)" &
765                 &,"rho(kg/m^3)", "Cs(m/s)", "Ma"
766             endif
767             write(unit_atmosphere_debug,210) (i_main-1)*dt, Z, H, Temperature, P_air, rho, Cs, Ma
768             format(a6,7a12)
769             format(f6.2,3f12.2,2e12.2,2f12.2)
770         endif
771     endif
772     return
773 end
774
775 !-----!
776 ! 副程式: 物性、推力參數 !
777 ! 輸入: input_1_thrust !
778 !       input_2_physical !
779 !       t !
780 !       dt !
781 !       i_main !
782 !       counter !
783 ! 輸出: thrust !
784 !       physical !
785 ! 定義: !
786 !-----!
787
788 subroutine sub_parameter(thrust, physical)
789     use global

```

```

788 use input_data
789 use solution
790 use unit
791 implicit none
792 real*8 thrust(5)
793 real*8 physical(4)
794 real*8 A,A1,A2
795 real*8 B,B1,B2
796 integer i,j
797
798 !燃畢前
799 if(i_main<counter) then
800     !推力資料的內插
801     do j=1,5
802         do i=1,26-1
803             if(input_1_thrust(i,1)<=t .and. t<=input_1_thrust(i+1,1)) then
804                 A = t
805                 A1 = input_1_thrust(i,1)
806                 A2 = input_1_thrust(i+1,1)
807                 B1 = input_1_thrust(i,j)
808                 B2 = input_1_thrust(i+1,j)
809                 !呼叫副程式:1維內插
810                 call sub_interpolation_1D(A,A1,A2,B,B1,B2)
811                 thrust(j) = B
812             endif
813         enddo
814     enddo
815     !物性資料的內插
816     do j=1,4
817         do i=1,1
818             if(mass<=input_2_physical(i,1) .and. mass>=input_2_physical(i+1,1)) then
819                 A = mass
820                 A1 = input_2_physical(i,1)
821                 A2 = input_2_physical(i+1,1)
822                 B1 = input_2_physical(i,j)
823                 B2 = input_2_physical(i+1,j)
824                 !呼叫副程式:1維內插
825                 call sub_interpolation_1D(A,A1,A2,B,B1,B2)
826                 physical(j) = B
827             endif
828         enddo
829     enddo
830 !燃畢
831 else
832     do i=1,5
833         thrust(i) = input_1_thrust(26,i)
834     enddo
835     do i=1,4
836         physical(i) = input_2_physical(2,i)
837     enddo
838 endif
839 if(i_main<counter) then
840     dY_dt(13) = -factor*thrust(5)
841 else
842     dY_dt(13) = 0.0
843     Y(13) = input_2_physical(2,1)
844     mass = input_2_physical(2,1)
845 endif
846
847 !check
848 if(mod(call_times-1,4)==0.0) then
849     if(i_main==1) then
850         open(unit=unit_out_physical_debug,file="output_debug_2_physical.txt")
851         write(unit_out_physical_debug,200)"t", "mass", "XCG", "Ixx", "Iyy", &
852         &"p", "TF_sea", "TF_vacuum", "dm/dt"
853         write(unit_out_physical_debug,200)"s", "kg", "m", "kg-m^2", "kg-m^2", &
854         &"kgf/cm^2", "kgf", "kgf", "kg/sec"
855     endif
856     write(unit_out_physical_debug,210)t,physical(1),physical(2),physical(3),&
857     physical(4),thrust(2),thrust(3),thrust(4),thrust(5)
858     format(a6,8a10)
859     format(f6.2,8f10.2)
860 endif

```

```

861     return
862 end
863 !-----!
864 ! 副程式:重力 (彈體座標系D)
865 ! 輸入:mass
866 ! 輸出:F_gravity
867 ! 定義:g=9.80665=重力加速度 (m/s^2)
868 !     F_gravity=重力 (N)
869 !     mass=重量 (kg)
870 ! 備註:z軸是朝下為正，因此重力加速度為正值(>0)
871 !-----!
872 subroutine sub_F_gravity(F_gravity)
873     use solution
874     use global
875     use constant
876     implicit none
877     integer i
878     real*8 X_G(3),X_D(3)
879     real*8 F_gravity(3)
880
881     !大地座標系G下的重力
882     X_G(1)=0.0
883     X_G(2)=0.0
884     X_G(3)=mass*g
885
886     !呼叫副程式:座標轉換 (地面座標系G->彈體座標系D)
887     call sub_Coordinate_Trans_GD(Euler_angle_GD,X_G,X_D)
888
889     !彈體座標系D下的重力
890     do i=1,3
891         F_gravity(i)=X_D(i)
892     enddo
893
894     !check
895     if(.false.) then
896         write(*,200) "mass","g"
897         write(*,210) mass,g
898         write(*,200) "X_G(1) ","X_G(2) ","X_G(3) "
899         write(*,210) (X_G(i),i=1,3)
900         write(*,200) "F_gravity(1) ","F_gravity(2) ","F_gravity(3) "
901         write(*,210) (F_gravity(i),i=1,3)
902         write(*,210) sqrt(F_gravity(1)**2+F_gravity(2)**2+F_gravity(3)**2)
903         format(3a16)
904         format(3f16.2)
905     endif
906
907     return
908 end
909 !-----!
910 ! 副程式:推力 (彈體座標系D)
911 ! 輸入:thrust
912 !     position
913 ! 輸出:F_thrust
914 ! 定義:F_thrust=推力 (N)
915 !     pressure_air=空氣壓力 (pa)
916 !     Area_exit=噴嘴面積 (m^2) (噴嘴直徑為0.182m)
917 ! 備註:當推力<0或燃畢後，推力為0
918 !     推力單位為牛頓 (N)，真空推力單位為 (kgf)
919 !-----!
920 subroutine sub_F_thrust(F_thrust,thrust)
921     use global
922     use module_atmosphere
923     use constant
924     use unit
925     implicit none
926     real*8, parameter :: Area_exit=(PI*(0.182**2))/4.0
927     real*8 F_thrust(3)
928     real*8 thrust(5)
929     real*8 F_T
930     integer i
931
932     !燃畢後推力=0，推力單位是kgf要轉乘N
933     if(i_main<counter) then

```

```

934     F_T=thrust(4)*g-P_air*Area_exit
935 else
936     F_T=0.0
937 endif
938
939 !推力<0.0
940 if(F_T<0.0) then
941     F_T=0.0
942 endif
943
944 !彈體座標系D下的推力
945 F_thrust(1)=F_T
946 F_thrust(2)=0.0
947 F_thrust(3)=0.0
948
949 !check
950 if(.true.) then
951     if(mod(call_times-1,4)==0.0) then
952         if(i_main==1) then
953             open(unit=output_thrust_debug,file="output_debug_3_thrust.txt")
954             write(unit=output_thrust_debug,200)"t",F_thrust(1),"F_thrust(2)",&
955                 &"F_thrust(3)","thrust(4)","P_air","Area_exit"
956             endif
957
958             write(unit=output_thrust_debug,210)t,(F_thrust(i),i=1,3),thrust(4),P_air,Area_ex
959             it
960             format(a6,6a12)
961             format(f6.2,4f12.2,e12.2,f12.4)
962         endif
963     endif
964     return
965 end
966
967 !-----!
968 ! 副程式:氣動力及氣動力矩(彈體座標系D)
969 ! 輸入:i_main
970 !     counter
971 !     input_3_aero_1
972 !     input_3_aero_2
973 !     input_3_aero_3
974 !     physical
975 !     velocity
976 !     angular_velocity
977 !     rho
978 !     Cs
979 ! 輸出:F_airforce
980 !     M_air
981 ! 定義:F_airforce(1)=彈體座標系x軸的氣動力(N)
982 !     F_airforce(2)=彈體座標系y軸的氣動力(N)
983 !     F_airforce(3)=彈體座標系z軸的氣動力(N)
984 !     M_air(1)=彈體座標系x軸的氣動力矩(N-m)
985 !     M_air(2)=彈體座標系y軸的氣動力矩(N-m)
986 !     M_air(3)=彈體座標系z軸的氣動力矩(N-m)
987 !     Cf(i)=合成氣動力係數
988 !     Cm(i)=合成氣動力矩係數
989 !     L_ref=參考長度(m)
990 !     S_ref=參考面積(m^2)
991 !     Q=動壓=0.5*rho*V^2
992 !     Ma=馬赫數
993 !     AF=總攻角(deg)
994 !     phi_c=風向角(deg)
995 !     XCG=彈體重心在彈體座標系x軸的位置(m)
996 !     velocity_D=彈體座標系下的速度(m/s)
997 !     Cf(i)=合成氣動力係數
998 !     Cm(i)=合成氣動力矩係數
999 !-----!
1000 subroutine sub_F_airforce(F_airforce,M_air,physical)
1001 use global
1002 use input_data
1003 use module_atmosphere
1004 use solution
1005 implicit none
1006 integer, parameter :: unit_output_air_1_debug=121,unit_output_air_2_debug=122
1007 integer i,j,k

```

```

1005 integer row,column,D3
1006 real*8 F_airforce(3)
1007 real*8 M_air(3)
1008 real*8 physical(4)
1009 real*8 X_G(3),X_D(3)
1010 real*8 AF_range(11),Ma_range(11)
1011 real*8 A,A1,A2
1012 real*8 B,B1,B2
1013 real*8 C,C1,C2
1014 real*8 AF1,AF2
1015 real*8 Ma1,Ma2
1016 real*8 CA,CN,XCP,CLFI,CHMAL,CHMAW,CNFAL,CNFAW
1017 real*8 CNQ,CMQ,CLP,CND,XCPD,CLDP,CHMD,CNFD,XCPFD,CND2,XCPD2,CLDP2
1018 real*8 XCG
1019 real*8 L_ref,S_ref
1020 real*8 Q
1021 real*8 Cf(3),Cm(3)
1022
1023 !大地座標系G下的速度
1024 do i=1,3
1025     X_G(i)=velocity(i)
1026 enddo
1027
1028 !呼叫副程式:座標轉換(地面座標系G->彈體座標系D)
1029 call sub_Coordinate_Trans_GD(Euler_angle_GD,X_G,X_D)
1030
1031 !彈體座標系D下的速度
1032 do i=1,3
1033     velocity_D(i)=X_D(i)
1034 enddo
1035
1036 L_ref=input_3_aero_2(1)
1037 S_ref=input_3_aero_2(2)
1038 Q=0.5*rho*(V**2)
1039
1040 XCG=physical(2)/L_ref
1041
1042 !計算總攻角AF,風向角phi_c
1043 if(V==0.0) then
1044     AF=0.0
1045 else
1046     AF=acosd(velocity_D(1)/V)
1047 endif
1048
1049 if(velocity_D(3)==0.0) then
1050     phi_c=0.0
1051 else
1052     phi_c=atan2(velocity_D(2),velocity_D(3))
1053 endif
1054
1055 if(velocity_D(1)==0.0) then
1056     apha=0.0
1057 else
1058     apha=atan2(velocity_D(3),velocity_D(1))
1059 endif
1060
1061 ! (3-1) 氣動力係數-1
1062 row=11
1063 column=10
1064 D3=9
1065
1066 !依據AF,Ma決定內插位置
1067 do i=1,row
1068     AF_range(i)=input_3_aero_1(i,1,1)
1069 enddo
1070 do j=1,column
1071     Ma_range(j)=input_3_aero_1(1,j,1)
1072 enddo
1073
1074 do i=2,row-1
1075     if(AF>=AF_range(i) .and. AF<=AF_range(i+1)) then
1076         exit
1077     endif

```



```

1078     enddo
1079     if (AF < AF_range(2)) then
1080         i=2
1081         AF=AF_range(2)
1082     elseif (AF > AF_range(row)) then
1083         i=row-1
1084         AF=AF_range(row)
1085     endif
1086 !   write(*,*) "AF",AF_range(2),AF_range(row),row-2+1
1087
1088     do j=2,column-1
1089         if (Ma >= Ma_range(j) .and. Ma <= Ma_range(j+1)) then
1090             exit
1091         endif
1092     enddo
1093     if (Ma < Ma_range(2)) then
1094         j=2
1095         Ma=Ma_range(2)
1096     elseif (Ma > Ma_range(column)) then
1097         j=column-1
1098         Ma=Ma_range(column)
1099     endif
1100 !   write(*,*) "Ma",Ma_range(2),Ma_range(column),column-2+1
1101
1102     AF1=AF_range(i)
1103     AF2=AF_range(i+1)
1104     Ma1=Ma_range(j)
1105     Ma2=Ma_range(j+1)
1106
1107     do k=1,D3
1108         A1=input_3_aero_1(i,j,k)
1109         A2=input_3_aero_1(i+1,j,k)
1110         B1=input_3_aero_1(i,j+1,k)
1111         B2=input_3_aero_1(i+1,j+1,k)
1112         !呼叫副程式:2維內插
1113         call sub_interpolation_2D(A,A1,A2,B,B1,B2,C,C1,C2,AF1,AF2,Ma1,Ma2)
1114         if (k==1 .and. i_main <= counter) then
1115             CA=C
1116         elseif (k==2 .and. i_main > counter) then
1117             CA=C
1118         elseif (k==3) then
1119             CN=C
1120         elseif (k==4) then
1121             XCP=C
1122         elseif (k==5) then
1123             CLFI=C
1124         elseif (k==6) then
1125             CHMAL=C
1126         elseif (k==7) then
1127             CHMAW=C
1128         elseif (k==8) then
1129             CNFAL=C
1130         elseif (k==9) then
1131             CNFAW=C
1132         endif
1133     enddo
1134
1135 ! (3-3) 氣動力係數-3
1136 row=13
1137 column=9
1138
1139 A=Ma
1140 A1=input_3_aero_3(1,j)
1141 A2=input_3_aero_3(1,j+1)
1142
1143 do i=2,row
1144     B1=input_3_aero_3(i,j)
1145     B2=input_3_aero_3(i,j+1)
1146     !呼叫副程式:1維內插
1147     call sub_interpolation_1D(A,A1,A2,B,B1,B2)
1148     if (i==2) then
1149         CNQ=B
1150     elseif (i==3) then

```

```

1151         CMQ=B
1152     elseif(i==4) then
1153         CLP=B
1154     elseif(i==5) then
1155         CND=B
1156     elseif(i==6) then
1157         XCPD=B
1158     elseif(i==7) then
1159         CLDP=B
1160     elseif(i==8) then
1161         CHMD=B
1162     elseif(i==9) then
1163         CNFD=B
1164     elseif(i==10) then
1165         XCPFD=B
1166     elseif(i==11) then
1167         CND2=B
1168     elseif(i==12) then
1169         XCPD2=B
1170     elseif(i==13) then
1171         CLDP2=B
1172     endif
1173 enddo
1174
1175 !計算氣動力和力矩
1176 if(V==0.0) then
1177     Cf(1)=-CA
1178     Cf(2)=-CN*sin(phi_c)
1179     Cf(3)=-CN*cos(phi_c)
1180
1181     Cm(1)=0.0
1182     Cm(2)=CN*(XCG-XCP)*cos(phi_c)
1183     Cm(3)=-CN*(XCG-XCP)*sin(phi_c)
1184 else
1185     Cf(1)=-CA
1186     Cf(2)=-CN*sin(phi_c)+CNQ*(0.5*angular_velocity(3)*L_ref/V)
1187     Cf(3)=-CN*cos(phi_c)-CNQ*(0.5*angular_velocity(2)*L_ref/V)
1188
1189     Cm(1)=CLP*(0.5*angular_velocity(1)*L_ref/V)
1190     Cm(2)=CN*(XCG-XCP)*cos(phi_c)+CMQ*(0.5*angular_velocity(2)*L_ref/V)
1191     Cm(3)=-CN*(XCG-XCP)*sin(phi_c)+CMQ*(0.5*angular_velocity(3)*L_ref/V)
1192 endif
1193 do i=1,3
1194     F_airforce(i)=Cf(i)*Q*S_ref
1195     M_air(i)=Cm(i)*Q*S_ref*L_ref
1196 enddo
1197
1198 !check
1199 if(.true.) then
1200     if(mod(call_times-1,4)==0.0) then
1201         if(i_main==1) then
1202             open(unit=unit_output_air_1_debug,file="output_debug_4_airforce_1.txt")
1203             write(unit_output_air_1_debug,100)"t","Cf(1)","Cf(2)","Cf(3)",&
1204                 &"Cm(1)","Cm(2)","Cm(3)"
1205
1206             open(unit=unit_output_air_2_debug,file="output_debug_5_airforce_2.txt")
1207             write(unit_output_air_2_debug,300)"t","phi_c","AF","MA","CA","CN","CNQ",&
1208                 &"CLP","CMQ","XCP","XCG"
1209         endif
1210         write(unit_output_air_1_debug,200)t,(Cf(i),i=1,3),(Cm(i),i=1,3)
1211         write(unit_output_air_2_debug,400)t,phi_c,AF,MA,CA,CN,CNQ,CLP,CMQ,XCP,XCG
1212
1213         format(a6,6a12)
1214         format(f6.2,6f12.2)
1215         format(a6,10a12)
1216         format(f6.2,10f12.2)
1217     endif
1218 endif
1219 return
1220 end
1221
1222 !-----!
1223 ! 副程式:總外力及力矩(彈體座標系D->大地座標系G) !
1224 ! 輸入:F_gravity(i) !

```

```

1224 !      F_thrust(i) !
1225 !      F_airforce(i) !
1226 !      M_air(i) !
1227 !      Euler_angle_GD(i) !
1228 !      i_main !
1229 !      counter !
1230 ! 輸出:F_total !
1231 !      M_total !
1232 ! 定義:F_total(1)=大地座標系X軸之總外力(N) !
1233 !      F_total(2)=大地座標系Y軸之總外力(N) !
1234 !      F_total(3)=大地座標系Z軸之總外力(N) !
1235 !      M_total(1)=大地座標系X軸之總外力矩(N-m) !
1236 !      M_total(2)=大地座標系Y軸之總外力矩(N-m) !
1237 !      M_total(3)=大地座標系Z軸之總外力矩(N-m) !
1238 !      F_gravity(i)=彈體座標系下之重力(N) !
1239 !      F_thrust(i)=彈體座標系下之推力(N) !
1240 !      F_airforce(i)=彈體座標系下之氣動力(N) !
1241 !      M_air(i)=彈體座標系下之氣動力矩(N-m) !
1242 !      Euler_angle_GD(i)=彈體座標系D與大地座標系G之尤拉角 !
1243 !      i_main=目前的計算次數 !
1244 !      counter=到燃畢時的計算次數 !
1245 !-----!
1246 subroutine sub_F_total(F_total,M_total,F_gravity,F_thrust,F_airforce,M_air)
1247 use global
1248 use solution
1249 use unit
1250 implicit none
1251 integer i
1252 real*8 F_total(3)
1253 real*8 M_total(3)
1254 real*8 F_gravity(3)
1255 real*8 F_thrust(3)
1256 real*8 F_airforce(3)
1257 real*8 M_air(3)
1258 real*8 X_D(3),X_G(3)
1259
1260 !彈體座標系D下的總外力
1261 do i=1,3
1262     X_D(i)=F_gravity(i)+F_thrust(i)+F_airforce(i)
1263 enddo
1264
1265 !呼叫副程式:座標轉換(彈體座標系D->地面座標系G)
1266 call sub_Coordinate_Trans_DG(Euler_angle_GD,X_G,X_D)
1267
1268 !彈體座標系G下的總外力
1269 do i=1,3
1270     F_total(i)=X_G(i)
1271 enddo
1272
1273 !彈體座標系D下的總外力矩
1274 do i=1,3
1275     M_total(i)=M_air(i)
1276 enddo
1277
1278 if(.false.) then
1279 write(*,220)t,(F_total(i),i=1,3)
1280 write(*,220)t,(F_gravity(i),i=1,3)
1281 write(*,220)t,(F_thrust(i),i=1,3)
1282 write(*,220)t,(F_airforce(i),i=1,3)
1283 write(*,220)t,(M_total(i),i=1,3)
1284     format(f6.2,3f12.2)
1285 endif
1286
1287 !check
1288 if(.true.) then
1289     if(mod(call_times-1,4)==0.0) then
1290         if(i_main==1) then
1291             open(unit=F_M_debug,file="output_debug_6_FM_.txt")
1292             write(unit_F_M_debug,200)"t","F_total(1)","F_total(2)","F_total(3)",&
1293             &"F_gravity(1)","F_gravity(2)","F_gravity(3)",&
1294             &"F_thrust(1)","F_thrust(2)","F_thrust(3)",&
1295             &"F_airforce(1)","F_airforce(2)","F_airforce(3)",&
1296             &"M_total(1)","M_total(2)","M_total(3)"

```

```

1297         endif
1298         write(unit_F_M debug,210) t, (F_total(i), i=1,3), (F_gravity(i), i=1,3), &
1299         & (F_thrust(i), i=1,3), (F_airforce(i), i=1,3), (M_total(i), i=1,3)
1300         format(a6,15a11)
1301         format(f6.2,15f11.2)
1302     endif
1303 endif
1304 return
1305 end
1306 !-----!
1307 ! 副程式:尤拉角(大地座標系G與彈體座標系D) !
1308 ! 輸入:angular_velocity(i) !
1309 ! Euler_angle_GD(i) !
1310 ! 輸出:Euler_angle_rate_GD !
1311 ! 定義:angular_velocity(1)=p=飛彈在地面座標系g之x角速度(rad/s) !
1312 ! angular_velocity(2)=q=飛彈在地面座標系g之y角速度(rad/s) !
1313 ! angular_velocity(3)=r=飛彈在地面座標系g之z角速度(rad/s) !
1314 ! Euler_angle_GD(1)=theta=俯仰角(rad) !
1315 ! Euler_angle_GD(2)=phi=偏航角(rad) !
1316 ! Euler_angle_GD(3)=gamma=傾斜角(rad) !
1317 !-----!
1318 subroutine sub_Euler_angle()
1319     use solution
1320     implicit none
1321     real*8 theta, phi, gamma
1322     real*8 P, Q, R
1323
1324     theta=Euler_angle_GD(1)
1325     phi=Euler_angle_GD(2)
1326     gamma=Euler_angle_GD(3)
1327
1328     P=angular_velocity(1)
1329     Q=angular_velocity(2)
1330     R=angular_velocity(3)
1331
1332     Euler_angle_rate_GD(1)=Q*cos(gamma)-R*sin(gamma)
1333     Euler_angle_rate_GD(2)=( Q*sin(gamma)+R*cos(gamma) )/cos(theta)
1334     Euler_angle_rate_GD(3)=P+( Q*sin(gamma)+R*cos(gamma) )*tan(theta)
1335
1336     return
1337 end
1338 !-----!
1339 ! 副程式:角加速度(彈體座標系D) !
1340 ! 輸入:angular_velocity(i) !
1341 ! Euler_angle_GD(i) !
1342 ! 輸出:Euler_angle_rate_GD !
1343 ! 定義:angular_velocity(1)=p=飛彈在地面座標系g之x角速度(rad/s) !
1344 ! angular_velocity(2)=q=飛彈在地面座標系g之y角速度(rad/s) !
1345 ! angular_velocity(3)=r=飛彈在地面座標系g之z角速度(rad/s) !
1346 ! Euler_angle_GD(1)=theta=俯仰角(rad) !
1347 ! Euler_angle_GD(2)=phi=偏航角(rad) !
1348 ! Euler_angle_GD(3)=gamma=傾斜角(rad) !
1349 !-----!
1350 subroutine sub_angular_velocity(M_total, physical)
1351     use solution
1352     use global
1353     implicit none
1354     integer i, j
1355     real*8 M_total(3)
1356     real*8 physical(4)
1357     real*8 I_moment(3,3), I_moment_inv(3,3)
1358     real*8 Ixx, Iyy, Izz, Ixy, Iyz, Ixz
1359     real*8 P, Q, R
1360     real*8 A(3), B(3,3)
1361
1362     Ixx=physical(3)
1363     Iyy=physical(4)
1364     Izz=physical(4)
1365     Ixy=0.0
1366     Iyz=0.0
1367     Ixz=0.0
1368
1369     I_moment(1,1)=Ixx

```

```

1370 I_moment(1,2)=-Ixy
1371 I_moment(1,3)=-Ixz
1372 I_moment(2,1)=I_moment(1,2)
1373 I_moment(2,2)=Iyy
1374 I_moment(2,3)=-Iyz
1375 I_moment(3,1)=I_moment(1,3)
1376 I_moment(3,2)=I_moment(2,3)
1377 I_moment(3,3)=Izz
1378
1379 P=angular_velocity(1)
1380 Q=angular_velocity(2)
1381 R=angular_velocity(3)
1382
1383 A(1)=M_total(1)-(Izz-Iyy)*Q*R+Iyz*(Q**2-R**2)+Ixz*P*Q-Ixy*P*R
1384 A(2)=M_total(2)-(Ixx-Izz)*P*R+Ixz*(R**2-P**2)+Ixy*Q*R-Iyz*P*Q
1385 A(3)=M_total(3)-(Iyy-Ixx)*P*Q+Ixy*(P**2-Q**2)+Iyz*P*R-Ixz*Q*R
1386
1387 !慣性矩反矩陣
1388 !呼叫副程式:反矩陣
1389 call inverse_matrix(I_moment,I_moment_inv)
1390 if(.false.) then
1391     write(*,*) "I_moment="
1392     write(*,200)((I_moment(i,j),j=1,3),i=1,3)
1393     write(*,*) "I_moment_inv="
1394     write(*,200)((I_moment_inv(i,j),j=1,3),i=1,3)
1395     B=matmul(I_moment,I_moment_inv)
1396     write(*,*) "I_moment*I_moment_inv="
1397     write(*,200)((B(i,j),j=1,3),i=1,3)
1398     format(3f12.4)
1399 endif
1400
1401 !計算角速度(彈體座標系D)
1402 angular_velocity_rate=matmul(I_moment_inv,A)
1403
1404 return
1405 end
1406 !-----!
1407 ! 副程式:微分方程(大地座標系G) !
1408 ! 輸入:velocity !
1409 !     position !
1410 !     F_total !
1411 !     mass !
1412 !     angular_velocity !
1413 !     angular_velocity_rate !
1414 !     Euler_angle_GD !
1415 !     Euler_angle_rate_GD !
1416 !     physicals !
1417 ! 輸出:Y !
1418 !     dY_dt !
1419 ! 定義:Y=積分前的物理量 !
1420 !     Y(1)=大地座標系下x軸方向之速度 !
1421 !     Y(2)=大地座標系下y軸方向之速度 !
1422 !     Y(3)=大地座標系下z軸方向之速度 !
1423 !     Y(4)=大地座標系下x軸方向之位置 !
1424 !     Y(5)=大地座標系下y軸方向之位置 !
1425 !     Y(6)=大地座標系下z軸方向之位置 !
1426 !     Y(7)=大地座標系下x軸方向之角速度 !
1427 !     Y(8)=大地座標系下y軸方向之角速度 !
1428 !     Y(9)=大地座標系下z軸方向之角速度 !
1429 !     Y(10)=尤拉角(1)俯仰角(theta) !
1430 !     Y(11)=尤拉角(2)偏航角(phi) !
1431 !     Y(12)=尤拉角(3)傾斜角(gamma) !
1432 !     Y(13)=重量(kg) !
1433 !     dY_dt=積分前的物理量的斜率 !
1434 !     dY_dt(1)=大地座標系下x軸方向之加速度 !
1435 !     dY_dt(2)=大地座標系下y軸方向之加速度 !
1436 !     dY_dt(3)=大地座標系下z軸方向之加速度 !
1437 !     dY_dt(4)=大地座標系下x軸方向之速度 !
1438 !     dY_dt(5)=大地座標系下y軸方向之速度 !
1439 !     dY_dt(6)=大地座標系下z軸方向之速度 !
1440 !     dY_dt(7)=大地座標系下x軸方向之角加速度 !
1441 !     dY_dt(8)=大地座標系下y軸方向之角加速度 !
1442 !     dY_dt(9)=大地座標系下z軸方向之角加速度 !

```

```

1443 !      dY_dt(10)=尤拉角(1)俯仰角(theta)斜率      !
1444 !      dY_dt(11)=尤拉角(2)偏航角(phi)斜率      !
1445 !      dY_dt(12)=尤拉角(3)傾斜角(gamma)斜率      !
1446 !      dY_dt(13)=重量斜率(kg/s)      !
1447 !      I_moment=慣性矩(3*3)      !
1448 !      I_moment_inv=慣性矩反矩陣(3*3)      !
1449 !-----!
1450 subroutine sub_eqs(F_total)
1451     use solution
1452     implicit none
1453     integer i
1454     real*8 F_total(3)
1455
1456     !積分的變數(dY)共13個:1~3:速度、4~6:位置、7~9:角速度、10~12:尤拉角、13:質量
1457
1458     !積分的變數(dY/dt)共13個:1~3:加速度、4~6:速度、7~9:角加速度、10~12:尤拉角斜率、13:質量斜率
1459     do i=1,3
1460         Y(i)=velocity(i)
1461         Y(i+3)=position(i)
1462         Y(i+6)=angular_velocity(i)
1463         Y(i+9)=Euler_angle_GD(i)
1464
1465         dY_dt(i)=F_total(i)/mass
1466         dY_dt(i+3)=velocity(i)
1467         dY_dt(i+6)=angular_velocity_rate(i)
1468         dY_dt(i+9)=Euler_angle_rate_GD(i)
1469     enddo
1470     Y(13)=mass
1471
1472     !check
1473     if(.false.) then
1474         write(*,200) (Y(i),i=1,13)
1475         write(*,200) (dY_dt(i),i=1,13)
1476
1477         format(13f8.2)
1478     endif
1479
1480     return
1481 end
1482 !-----!
1483 ! 副程式:數值積分(Runge Kutta Method,4階)      !
1484 ! 輸入:Y      !
1485 !      dY_dt      !
1486 !      dt      !
1487 ! 輸出:Y      !
1488 ! 定義:Y_old=Y(t(j))=積分前的Y      !
1489 !      Y=Y(t(j+1))=積分後的Y      !
1490 !      dY_dt=Y的斜率      !
1491 !      t=時間(s)      !
1492 !      dt=時間步長(s)      !
1493 !      Slope_1=Runge Kutta的係數(斜率)      !
1494 !      Slope_2=Runge Kutta的係數(斜率)      !
1495 !      Slope_3=Runge Kutta的係數(斜率)      !
1496 !      Slope_4=Runge Kutta的係數(斜率)      !
1497 !-----!
1498 subroutine sub_integral()
1499     use global
1500     use solution
1501     implicit none
1502     integer i,N
1503     real*8 Y_old(13)
1504     real*8 Slope_1(13),Slope_2(13),Slope_3(13),Slope_4(13)
1505     real*8 t_old
1506
1507     !需積分的變數個數
1508     N=13
1509     !儲存上一個時間點的函數值與時間(共N個需積分的函數)
1510     do i=1,N
1511         Y_old(i)=Y(i)
1512         t_old=t
1513     enddo

```

```

1514 !計算Slope_1
1515 do i=1,N
1516     Slope_1(i)=dY_dt(i)
1517 enddo
1518
1519 !計算Slope_2(前進0.5dt)
1520 t=t_old+0.5*dt
1521 do i=1,N
1522     !計算新函數值
1523     Y(i)=Y_old(i)+0.5*dt*Slope_1(i)
1524 enddo
1525 !呼叫副程式:6Dof方程式
1526 call sub_6Dof()
1527 do i=1,N
1528     Slope_2(i)=dY_dt(i)
1529 enddo
1530
1531 !計算Slope_3(前進0.5dt)
1532 t=t_old+0.5*dt
1533 do i=1,N
1534     !計算新函數值
1535     Y(i)=Y_old(i)+0.5*dt*Slope_2(i)
1536 enddo
1537 !呼叫副程式:6Dof方程式
1538 call sub_6Dof()
1539 do i=1,N
1540     Slope_3(i)=dY_dt(i)
1541 enddo
1542
1543 !計算Slope_4(前進dt)
1544 t=t_old+dt
1545 do i=1,N
1546     !計算新函數值
1547     Y(i)=Y_old(i)+dt*Slope_3(i)
1548 enddo
1549 !呼叫副程式:6Dof方程式
1550 call sub_6Dof()
1551 do i=1,N
1552     Slope_4(i)=dY_dt(i)
1553 enddo
1554
1555 !計算下一個時間點的Y
1556 do i=1,N
1557     Y(i)=Y_old(i)+(dt/6.0)*(Slope_1(i)+2.*Slope_2(i)+2.*Slope_3(i)+Slope_4(i))
1558 enddo
1559
1560 return
1561 end
1562 !-----!
1563 ! 副程式:座標轉換(地面座標系G->彈體座標系D) !
1564 ! 輸入:Euler_angle_GD !
1565 !     X_G !
1566 ! 輸出:X_D !
1567 ! 定義:Euler_angle_GD(1)=theta=俯仰角(rad) !
1568 !     Euler_angle_GD(2)=phi=偏航角(rad) !
1569 !     Euler_angle_GD(3)=gamma=傾斜角(rad) !
1570 !     Trans_GD(3,3)=地面座標系G->彈體座標系D之轉換矩陣 !
1571 !     X_G(1)=變數在地面座標系G之X軸方向投影量 !
1572 !     X_G(2)=變數在地面座標系G之Y軸方向投影量 !
1573 !     X_G(3)=變數在地面座標系G之Z軸方向投影量 !
1574 !     X_D(1)=變數在彈體座標系D之X軸方向投影量 !
1575 !     X_D(2)=變數在彈體座標系D之Y軸方向投影量 !
1576 !     X_D(3)=變數在彈體座標系D之Z軸方向投影量 !
1577 ! 備註:大地座標系OXgYgZg固定於地球上，座標系原點O為發射點 !
1578 !     OXg軸朝北為正，OYg軸朝東為正，OZg軸朝下為正。 !
1579 ! 備註:彈體座標系OXyz固定於彈體，座標系原點O為彈體重心 !
1580 !     OX軸與彈體縱軸重合，朝彈尖為正，OY軸朝右為正，OZ軸朝下為正。 !
1581 !-----!
1582 subroutine sub_Coordinate_Trans_GD(Euler_angle_GD,X_G,X_D)
1583 implicit none
1584 integer i,j
1585 real*8 Euler_angle_GD(3)
1586 real*8 Trans_GD(3,3)

```



```

1587     real*8    X_G(3),X_D(3)
1588     real*8    theta,phi,gamma
1589
1590     theta=Euler_angle_GD(1)
1591     phi=Euler_angle_GD(2)
1592     gamma=Euler_angle_GD(3)
1593
1594     !轉換矩陣(G->D)
1595     Trans_GD(1,1)=cos(phi)*cos(theta)
1596     Trans_GD(1,2)=cos(theta)*sin(phi)
1597     Trans_GD(1,3)=-sin(theta)
1598     Trans_GD(2,1)=-cos(gamma)*sin(phi)+sin(gamma)*sin(theta)*cos(phi)
1599     Trans_GD(2,2)=cos(gamma)*cos(phi)+sin(gamma)*sin(theta)*sin(phi)
1600     Trans_GD(2,3)=sin(gamma)*cos(theta)
1601     Trans_GD(3,1)=sin(gamma)*sin(phi)+cos(gamma)*sin(theta)*cos(phi)
1602     Trans_GD(3,2)=-sin(gamma)*cos(phi)+cos(gamma)*sin(theta)*sin(phi)
1603     Trans_GD(3,3)=cos(gamma)*cos(theta)
1604
1605     do i=1,3
1606         X_D(i)=0.0
1607         do j=1,3
1608             X_D(i)=X_D(i)+Trans_GD(i,j)*X_G(j)
1609         enddo
1610     enddo
1611
1612     !check
1613     if(.false.) then
1614         write(*,"(3a16)") "theta","phi","gamma"
1615         write(*,"(3f16.2)") theta,phi,gamma
1616         write(*,"(a16)") "Trans Matrix(GD)"
1617         write(*,"(3f16.2)") ((Trans_GD(i,j),j=1,3),i=1,3)
1618     endif
1619     return
1620 end
1621
1622 !-----!
1623 ! 副程式:座標轉換(彈體座標系D->地面座標系G)
1624 ! 輸入:Euler_angle_GD
1625 ! 輸出:X_G
1626 ! 定義:Euler_angle_GD(1)=theta=俯仰角(rad)
1627 !       Euler_angle_GD(2)=phi=偏航角(rad)
1628 !       Euler_angle_GD(3)=gamma=傾斜角(rad)
1629 !       Trans_DG(3,3)=彈體座標系D->地面座標系G之轉換矩陣
1630 !       X_G(1)=變數在地面座標系G之X軸方向投影量
1631 !       X_G(2)=變數在地面座標系G之Y軸方向投影量
1632 !       X_G(3)=變數在地面座標系G之Z軸方向投影量
1633 !       X_D(1)=變數在彈體座標系D之X軸方向投影量
1634 !       X_D(2)=變數在彈體座標系D之Y軸方向投影量
1635 !       X_D(3)=變數在彈體座標系D之Z軸方向投影量
1636 ! 備註:大地座標系OXgYgZg固定於地球上，座標系原點O為發射點
1637 !       OXg軸朝北為正，OYg軸朝東為正，OZg軸朝下為正。
1638 ! 備註:彈體座標系OXYZ固定於彈體，座標系原點O為彈體重心
1639 !       OX軸與彈體縱軸重合，朝彈尖為正，OY軸朝右為正，OZ軸朝下為正。
1640 !-----!
1641 subroutine sub_Coordinate_Trans_DG(Euler_angle_GD,X_G,X_D)
1642     implicit none
1643     integer i,j
1644     real*8 Euler_angle_GD(3)
1645     real*8 Trans_DG(3,3)
1646     real*8 X_G(3),X_D(3)
1647     real*8 theta,phi,gamma
1648
1649     theta=Euler_angle_GD(1)
1650     phi=Euler_angle_GD(2)
1651     gamma=Euler_angle_GD(3)
1652
1653     !轉換矩陣(D->G)
1654     Trans_DG(1,1)=cos(phi)*cos(theta)
1655     Trans_DG(2,1)=cos(theta)*sin(phi)
1656     Trans_DG(3,1)=-sin(theta)
1657     Trans_DG(1,2)=-cos(gamma)*sin(phi)+sin(gamma)*sin(theta)*cos(phi)
1658     Trans_DG(2,2)=cos(gamma)*cos(phi)+sin(gamma)*sin(theta)*sin(phi)
1659     Trans_DG(3,2)=sin(gamma)*cos(theta)

```

```

1660 Trans_DG(1,3)=sin(gamma)*sin(phi)+cos(gamma)*sin(theta)*cos(phi)
1661 Trans_DG(2,3)=-sin(gamma)*cos(phi)+cos(gamma)*sin(theta)*sin(phi)
1662 Trans_DG(3,3)=cos(gamma)*cos(theta)
1663
1664 do i=1,3
1665   X_G(i)=0.0
1666   do j=1,3
1667     X_G(i)=X_G(i)+Trans_DG(i,j)*X_D(j)
1668   enddo
1669 enddo
1670
1671 !check
1672 if(.false.)then
1673   write(*,"(3a16)") "theta","phi","gamma"
1674   write(*,"(3f16.2)") theta,phi,gamma
1675   write(*,"(a16)") "Trans Matrix(DG)"
1676   write(*,"(3f16.2)") ((Trans_DG(i,j),j=1,3),i=1,3)
1677 endif
1678 return
1679 end
1680 !-----!
1681 ! 副程式:1維內插 !
1682 ! 輸入:A !
1683 !     A1 !
1684 !     A2 !
1685 !     B1 !
1686 !     B2 !
1687 ! 輸出:B !
1688 ! 定義: !
1689 !-----!
1690 subroutine sub_interpolation_1D(A,A1,A2,B,B1,B2)
1691   implicit none
1692   real*8 A,A1,A2
1693   real*8 B,B1,B2
1694   B = B1 + (B2-B1)*( (A-A1)/(A2-A1) )
1695   return
1696 end
1697 !-----!
1698 ! 副程式:2維內插 !
1699 ! 輸入:A !
1700 !     A1 !
1701 !     A2 !
1702 !     B !
1703 !     B1 !
1704 !     B2 !
1705 !     C1 !
1706 !     C2 !
1707 !     AF !
1708 !     AF1 !
1709 !     AF2 !
1710 !     Ma !
1711 !     Ma1 !
1712 !     Ma2 !
1713 ! 輸出:C !
1714 ! 定義: !
1715 !-----!
1716 subroutine sub_interpolation_2D(A,A1,A2,B,B1,B2,C,C1,C2,AF1,AF2,Ma1,Ma2)
1717   use solution
1718   implicit none
1719   real*8 A,A1,A2
1720   real*8 B,B1,B2
1721   real*8 C,C1,C2
1722   real*8 AF1,AF2
1723   real*8 Ma1,Ma2
1724
1725   A = A1 + (A2-A1)*( (AF-AF1)/(AF2-AF1) )
1726   B = B1 + (B2-B1)*( (AF-AF1)/(AF2-AF1) )
1727
1728   C1=A
1729   C2=B
1730   C = C1 + (C2-C1)*( (Ma-Ma1)/(Ma2-Ma1) )
1731   return
1732 end

```

```

1733
1734
1735 !-----!
1736 ! 副程式:反矩陣 !
1737 ! 輸入:A !
1738 ! 輸出:inv_A !
1739 ! 定義: !
1740 !-----!
1741 subroutine inverse_matrix(A,inv_A)
1742   implicit none
1743   integer i,j
1744   !real*8 A(3,3),inv_A(3,3),I_matrix(3,3),save(3,3)
1745   real*8 A(3,3),inv_A(3,3),save(3,3)
1746   real*8 f
1747   ! data I_matrix / 1,0,0,0,1,0,0,0,1 /
1748   real*8 I_matrix(3,3)
1749
1750   do i=1,3
1751     do j=1,3
1752       if(i==j) then
1753         I_matrix(i,j)=1.0
1754       else
1755         I_matrix(i,j)=0.0
1756       endif
1757     end do
1758   end do
1759   !先清空矩陣
1760   inv_A=0.0
1761
1762   if(.false.) then
1763     write(*,*) "A="
1764     write(*,200)((A(i,j),j=1,3),i=1,3)
1765     write(*,*) "inv_A="
1766     write(*,200)((I_matrix(i,j),j=1,3),i=1,3)
1767     format(3f12.4)
1768   endif
1769   !先將A存入save中
1770   save=A
1771
1772   !計算反矩陣
1773   !1
1774   if(A(1,1) /= 1) then
1775     f=1.0/A(1,1)
1776     do j=1,3
1777       A(1,j)=f*A(1,j)
1778       I_matrix(1,j)=f*I_matrix(1,j)
1779     enddo
1780   endif
1781   !2
1782   if(A(2,1) /= 0) then
1783     f=-A(2,1)/A(1,1)
1784     do j=1,3
1785       A(2,j)=A(2,j)+f*A(1,j)
1786       I_matrix(2,j)=I_matrix(2,j)+f*I_matrix(1,j)
1787     enddo
1788   endif
1789   !3
1790   if(A(3,1) /= 0) then
1791     f=-A(3,1)/A(1,1)
1792     do j=1,3
1793       A(3,j)=A(3,j)+f*A(1,j)
1794       I_matrix(3,j)=I_matrix(3,j)+f*I_matrix(1,j)
1795     enddo
1796   endif
1797   !4
1798   if(A(2,2) /= 1) then
1799     f=1.0/A(2,2)
1800     do j=1,3
1801       A(2,j)=f*A(2,j)
1802       I_matrix(2,j)=f*I_matrix(2,j)
1803     enddo
1804   endif
1805   !5

```

```

1806   if(A(3,2) /= 0) then
1807       f=-A(3,2)/A(2,2)
1808       do j=1,3
1809           A(3,j)=A(3,j)+f*A(2,j)
1810           I_matrix(3,j)=I_matrix(3,j)+f*I_matrix(2,j)
1811       enddo
1812   endif
1813   !6
1814   if(A(3,3) /= 1) then
1815       f=1.0/A(3,3)
1816       do j=1,3
1817           A(3,j)=f*A(3,j)
1818           I_matrix(3,j)=f*I_matrix(3,j)
1819       enddo
1820   endif
1821   !7
1822   if(A(2,3) /= 0) then
1823       f=-A(2,3)/A(3,3)
1824       do j=1,3
1825           A(2,j)=A(2,j)+f*A(3,j)
1826           I_matrix(2,j)=I_matrix(2,j)+f*I_matrix(3,j)
1827       enddo
1828   endif
1829   !8
1830   if(A(1,3) /= 0) then
1831       f=-A(1,3)/A(3,3)
1832       do j=1,3
1833           A(1,j)=A(1,j)+f*A(3,j)
1834           I_matrix(1,j)=I_matrix(1,j)+f*I_matrix(3,j)
1835       enddo
1836   endif
1837   !9
1838   if(A(1,2) /= 0) then
1839       f=-A(1,2)/A(2,2)
1840       do j=1,3
1841           A(1,j)=A(1,j)+f*A(2,j)
1842           I_matrix(1,j)=I_matrix(1,j)+f*I_matrix(2,j)
1843       enddo
1844   endif
1845   inv_A=I_matrix
1846   do i=1,3
1847       do j=1,3
1848           if(abs(inv_A(i,j))>1e10) then
1849               write(*,*) "inverse matrix dosen't exit!"
1850               stop
1851           endif
1852       enddo
1853   enddo
1854
1855   !將save取回重新放入A中
1856   A=save
1857
1858   return
1859 end
1860 !-----!
1861 ! 副程式:輸出計算結果!
1862 ! 輸入: !
1863 ! 輸出: !
1864 ! 定義: !
1865 !-----!
1866 subroutine sub_output(timepoint)
1867     use global
1868     use constant
1869     use module_atmosphere
1870     use solution
1871     use unit
1872     implicit none
1873     integer i,j
1874     real*8 timepoint(5,5)
1875     real*8 R,H
1876
1877     !R:水平距離(km)
1878     R=sqrt(Y(4)**2+Y(5)**2)/1000.

```

```

1879 !H:高度(km)
1880 H=-Y(6)/1000.
1881 !i=1:離軌,i=2:最大速度,i=3:燃畢,i=4:最高點,i=5:落海
1882 !j=1:時間,j=2:高度,j=3:射程,j=4:速度,j=5:馬赫數
1883
1884 !i=2:最大速度
1885 i=2
1886 if(V>timepoint(2,4)) then
1887     timepoint(i,1)=t
1888     timepoint(i,2)=-Y(6)/1000.
1889     timepoint(i,3)=sqrt(Y(4)**2+Y(5)**2)/1000.
1890     timepoint(i,4)=V
1891     timepoint(i,5)=Ma
1892 endif
1893
1894 !i=3:燃畢
1895 i=3
1896 if(i_main==counter) then
1897     timepoint(i,1)=t
1898     timepoint(i,2)=-Y(6)/1000.
1899     timepoint(i,3)=sqrt(Y(4)**2+Y(5)**2)/1000.
1900     timepoint(i,4)=V
1901     timepoint(i,5)=Ma
1902 endif
1903
1904 !i=4:最高點
1905 i=4
1906 if(-Y(6)/1000.>timepoint(4,2)) then
1907     timepoint(i,1)=t
1908     timepoint(i,2)=-Y(6)/1000.
1909     timepoint(i,3)=sqrt(Y(4)**2+Y(5)**2)/1000.
1910     timepoint(i,4)=V
1911     timepoint(i,5)=Ma
1912 endif
1913
1914 !i=5:落海
1915 i=5
1916 if(-Y(6)<10.) then
1917     timepoint(i,1)=t
1918     timepoint(i,2)=-Y(6)/1000.
1919     timepoint(i,3)=sqrt(Y(4)**2+Y(5)**2)/1000.
1920     timepoint(i,4)=V
1921     timepoint(i,5)=Ma
1922 endif
1923
1924 !積分的變數(dY)共13個:1~3:速度、4~6:位置、7~9:角速度、10~12:尤拉角、13:質量
1925
1926 !積分的變數(dY/dt)共13個:1~3:加速度、4~6:速度、7~9:角加速度、10~12:尤拉角斜率、13:質量斜率
1927 if(i_main==1) then
1928     open(unit=unit_output_6D_1,file="output_6D_1.txt")
1929     write(unit_output_6D_1,100)"t(s) ","Ax (m/s^2) ","Ay (m/s^2) ","Az (m/s^2) ",&
1930         &"Vx (m/s) ","Vy (m/s) ","Vz (m/s) ","X (m) ","Y (m) ","Z (m) "
1931     open(unit=unit_output_6D_2,file="output_6D_2.txt")
1932     write(unit_output_6D_2,100)"t(s) ","P(deg) ","Q(deg) ","R(deg) ",&
1933         &"theta(deg) ","phi(deg) ","gamma(deg) ","dP/dt(deg/s) ",&
1934         &"dQ/dt(deg/s) ","dR/dt(deg/s) "
1935     open(unit=unit_output_6D_3,file="output_6D_3.txt")
1936     write(unit_output_6D_3,110)"t(s) ","AF(deg) ","apha(deg) ","phi_c(deg) ",&
1937         &"V (m/s) ","Cs (m/s) ","Ma","mass (kg) "
1938     open(unit=unit_output_6D_4,file="output_6D_4.txt")
1939     write(unit_output_6D_4,130)"t(s) ","Range(km) ","H(km) ","AF(deg) ",&
1940         &"V(m/s) ","Ma","mass(kg) "
1941 endif
1942 write(unit_output_6D_1,200)t,(dY_dt(i),i=1,3),(Y(i),i=1,6)
1943 write(unit_output_6D_2,200)t,(Y(i)*180./PI,i=7,12),(dY_dt(i),i=7,9)
1944 write(unit_output_6D_3,210)t,AF,apha*PI/180.,phi_c*PI/180.,V,Cs,Ma,Y(13)
1945 write(unit_output_6D_4,230)t,R,H,AF,V,Ma,Y(13)
1946
1947 format(a6,9a14)
1948 format(f6.2,9f14.2)
1949 format(a6,7a14)
1950 format(f6.2,7f14.2)

```

```
1950         format(a6,6a14)
1951         format(f6.2,6f14.2)
1952
1953     if(-Y(6)/1000.<0.1) then
1954         open(unit=unit_output_6D_5,file="output_6D_5.txt")
1955         write(unit_output_6D_5,120)"key point","t(s)","H(km)","Range(km)","V(m/s)","Ma"
1956         write(unit_output_6D_5,220)"leave launcher",(timepoint(1,j),j=1,5)
1957         write(unit_output_6D_5,220)"Max. V",(timepoint(2,j),j=1,5)
1958         write(unit_output_6D_5,220)"motor burn out",(timepoint(3,j),j=1,5)
1959         write(unit_output_6D_5,220)"Max. H",(timepoint(4,j),j=1,5)
1960         write(unit_output_6D_5,220)"fall into sea",(timepoint(5,j),j=1,5)
1961         close(unit_output_6D_5)
1962     endif
1963         format(6a14)
1964         format(a14,5f14.2)
1965
1966     return
1967 end
1968
```