```fortran
!---------------------------------------------------------------------------!
!  6Dof彈道模擬(6 Degreed of freedom trajectory simulation)                 !
!  受控體(plant):無控火箭                                                   !
!  作者:陳維霖 台大機械博士班 學號:D09522006                                !
!  date:2021/1/16                                                           !
!  version:1                                                                !
!  輸入檔:(1)input_1_thrust                                                 !
!         (2)input_2_physical                                               !
!         (3)input_3_aero                                                   !
!         (3-1-1)input_3_aero_1_1_CA_ON                                     !
!         (3-1-2)input_3_aero_1_2_CA_OFF                                    !
!         (3-1-3)input_3_aero_1_3_CN                                        !
!         (3-1-4)input_3_aero_1_4_XCP                                       !
!         (3-1-5)input_3_aero_1_5_CLFI                                      !
!         (3-1-6)input_3_aero_1_6_CHMAL                                     !
!         (3-1-7)input_3_aero_1_7_CHMAW                                     !
!         (3-1-8)input_3_aero_1_8_CNFAL                                     !
!         (3-1-9)input_3_aero_1_9_CNFAW                                     !
!         (3-2)input_3_aero_2_reference                                     !
!         (3-3)input_3_aero_3_other                                         !
!         (4)input_4_atmosphere                                             !
!  輸出檔:(1)output_6D_1                                                    !
!         (2)output_6D_2                                                    !
!         (3)output_6D_3                                                    !
!         (4)output_6D_4                                                    !
!         (5)output_6D_5                                                    !
!         (6)output_check_1_thrust                                          !
!         (7)output_check_2_physical                                        !
!         (8)output_check_3_aero                                            !
!         (9)output_check_4_atmospher                                       !
!         (10)output_debug_1_atmosphere                                     !
!         (11)output_debug_2_physical                                       !
!         (12)output_debug_3_thrust                                         !
!         (13)output_debug_4_airforce_1                                     !
!         (14)output_debug_5_airforce_2                                     !
!         (15)output_debug_6_FM_                                            !
!---------------------------------------------------------------------------!

!---------------------------------------------------------------------------!
!                       建模 for 常數、變數                                 !
!---------------------------------------------------------------------------!
module constant
  implicit none
  real*8, parameter :: PI=3.1415927410125732421875
  real*8, parameter :: g=9.80665
endmodule

module global
  implicit none
  real*8, parameter :: dt=0.01
  real*8, save :: t,factor
  integer, save :: counter,i_main,call_times
endmodule

module module_atmosphere
  implicit none
  real*8, save :: P_air,rho,Cs
endmodule

module solution
  implicit none
  real*8, save :: Y(13),dY_dt(13)
  real*8, save :: position(3),velocity(3),angular_velocity(3),Euler_angle_GD(3),mass
  real*8, save :: angular_velocity_rate(3),Euler_angle_rate_GD(3)
  real*8, save :: velocity_D(3)
  real*8, save :: AF,phi_c,apha,V,Ma
endmodule

module input_data
  implicit none
  real*8  input_1_thrust(100,100)
  real*8  input_2_physical(10,10)
  real*8  input_3_aero_1(11,10,9),input_3_aero_2(4),input_3_aero_3(13,9)
  real*8  input_4_atmosphere(22,6)
endmodule

module unit
  implicit none
  !副程式:讀取輸入資料(input)
  integer, parameter :: unit_thrust=10,unit_thrust_check=15
  integer, parameter :: unit_physical=20,unit_physical_check=25
  integer, parameter :: unit_aero_3_1=30,unit_aero_3_2=45,unit_aero_3_3=50
  integer, parameter :: unit_aero_check=55
  !副程式:初始資料計算(initial)
  integer, parameter :: unit_thrust_interpolation_check=65
  integer, parameter :: unit_physical_interpolation_check=70
  !副程式:標準大氣程式(atmosphere)
  integer, parameter :: unit_atmosphere=75,unit_atmosphere_check=80,unit_atmosphere_debug=85
  !副程式:物性、推力參數
  integer, parameter :: unit_out_physical_debug=90
  !副程式:推力(彈體座標系D)
  integer, parameter :: unit_output_thrust_debug=95
  !副程式:總外力及力矩(彈體座標系D->大地座標系G)
  integer, parameter :: unit_F_M_debug=100
  !副程式:輸出計算結果
  integer, parameter :: unit_output_6D_1=150,unit_output_6D_2=200,&
                       &unit_output_6D_3=250,unit_output_6D_4=300,unit_output_6D_5=350
endmodule
!--------------------------------------------------------------------------!
!                              主程式                                      !
!--------------------------------------------------------------------------!
Program main
  use global
  use input_data
  use solution
  implicit none
  integer i
  real*8 timepoint(5,5)
  write(*,*)"-------------------------Main Program Start-------------------------"

  !-------------------讀取輸入資料(input)-------------------!
  !呼叫副程式:讀取輸入資料(input)
  call sub_input()
  !-------------------初始資料計算(initial)-------------------!
  !呼叫副程式:初始資料計算(initial)
  call sub_initial()

  !各種計數器
  call_times = 0
  i_main = 0

  !------------------------6Dof計算迴圈------------------------!
  do i_main=1,100000
    t = (i_main-1)*dt
    !------------------------螢幕輸出------------------------!
    if(mod(i_main-1,100)==0 .or. i_main==1 .or. i_main==counter)then
      if(i_main==1)then
        write(*,*)"----------------------6Dof interation start---------------------"

        write(*,200)  "t","X(m)","Y(m)","Z(m)"
      endif
      write(*,210)  t,(Y(i),i=4,6)
          format(a16,3a16)
          format(f16.4,3f16.2)
    endif
    !------------------------輸出計算結果------------------------!
    !呼叫副程式:輸出計算結果
    call sub_output(timepoint)

    !------------------------6Dof方程式------------------------!
    !呼叫副程式:6Dof方程式(6Dof)
    call sub_6Dof()

    !------------------------數值積分(Runge Kutta Method)------------------------!
    !呼叫副程式:數值積分(Runge Kutta Method)
    call sub_integral()

    if(-Y(6)<0.0)then
      write(*,*)"Height(m)<0:",-Y(6)
      exit
    endif
  enddo
  write(*,*)"--------------------------Main Program End-----------------------------"

End Program
!-----------------------------------------------------------------------------!
!  副程式:讀取輸入資料(input)                                                 !
!  輸入:                                                                      !
!  輸出:input_1_thrust                                                        !
!       input_2_physical                                                      !
!       input_3_aero_1                                                        !
!       input_3_aero_2                                                        !
!       input_3_aero_3                                                        !
!  定義:input_1_thrust:推力                                                   !
!       input_2_physical:物性                                                 !
!       input_3_aero_1:氣動力係數(二維)                                       !
!       input_3_aero_2:氣動力參考長度,面積                                    !
!       input_3_aero_3:氣動力係數(一維)                                       !
!       D3:矩陣個數                                                           !
!       row:矩陣列數                                                          !
!       column:矩陣行數                                                       !
!-----------------------------------------------------------------------------!
subroutine sub_input()
  use input_data
  use unit
  integer i,j,k
  integer row,column,d3
  integer status

  write(*,*)"------------------subroutine:read input data-------------------"

  !(1)讀取資料-推力
  row=26
  column=5
  Open(unit=unit_thrust,file='input_1_thrust.txt',status='old')
    read(unit_thrust,*)
    read(unit_thrust,*,iostat=status) ((input_1_thrust(i,j),j=1,column),i=1,row)
    write(*,*)"thrust data,read result(0:success,other:fail):",status
  close(unit_thrust)

  Open(unit=unit_thrust_check,file='output_check_1_thrust.txt',status='unknown')
    write(unit_thrust_check,"(5A16)")
    "t(sec)","p(kgf/cm^2)","TF_sea(kgf)","TF_vacuum(kgf)","dm/dt(kg/sec)"
    write(unit_thrust_check,"(5f16.2)") ((input_1_thrust(i,j),j=1,column),i=1,row)
  close(unit_thrust_check)

  !(2)讀取資料-物性
  row=2
  column=4
  Open(unit=unit_physical,file='input_2_physical.txt',status='old')
    read(unit_physical,*)
    read(unit_physical,*,iostat=status) ((input_2_physical(i,j),j=1,column),i=1,row)
    write(*,*)"physical data,read result(0:success,other:fail):",status
  close(unit_physical)

  Open(unit=unit_physical_check,file='output_check_2_physical.txt',status='unknown')
    write(unit_physical_check,"(5A16)") "m(kg)","x_cg(m)","Ixx(kg-m^2)","Iyy(kg-m^2)"
    write(unit_physical_check,"(4f16.2)") ((input_2_physical(i,j),j=1,column),i=1,row)
  close(unit_physical_check)

  !(3)讀取資料-氣動力係數
  row=11
  column=10
  d3=9
  !3-1
  !1-1:CA(ON)
  k=1
  Open(unit=unit_aero_3_1,file='input_3_aero_1_1.txt',status='old')
```

```
        read(unit_aero_3_1,*)
        read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_1)

      !1-2:CA(OFF)
      k=2
      Open(unit=unit_aero_3_1,file='input_3_aero_1_2.txt',status='old')
        read(unit_aero_3_1,*)
        read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_1)

      !1-3:CN
      k=3
      Open(unit=unit_aero_3_1,file='input_3_aero_1_3.txt',status='old')
        read(unit_aero_3_1,*)
        read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_1)

      !1-4:XCP
      k=4
      Open(unit=unit_aero_3_1,file='input_3_aero_1_4.txt',status='old')
        read(unit_aero_3_1,*)
        read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_1)

      !1-5:CLFI(22.5)
      k=5
      Open(unit=unit_aero_3_1,file='input_3_aero_1_5.txt',status='old')
        read(unit_aero_3_1,*)
        read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_1)

      !1-6:CHMAL
      k=6
      Open(unit=unit_aero_3_1,file='input_3_aero_1_6.txt',status='old')
        read(unit_aero_3_1,*)
        read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_1)

      !1-7:CHMAW
      k=7
      Open(unit=unit_aero_3_1,file='input_3_aero_1_7.txt',status='old')
        read(unit_aero_3_1,*)
        read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_1)

      !1-8:CNFAL
      k=8
      Open(unit=unit_aero_3_1,file='input_3_aero_1_8.txt',status='old')
        read(unit_aero_3_1,*)
        read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_1)

      !1-9:CNFAW
      k=9
      Open(unit=unit_aero_3_1,file='input_3_aero_1_9.txt',status='old')
        read(unit_aero_3_1,*)
        read(unit_aero_3_1,*,iostat=status) ((input_3_aero_1(i,j,k),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_1)

      !3-2
      k=10
      Open(unit=unit_aero_3_2,file='input_3_aero_2.txt',status='old')
        read(unit_aero_3_2,"(9x,f16.8)",iostat=status) (input_3_aero_2(i),i=1,4)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_2)

      !3-3
      row=13
      column=9
      k=11
      Open(unit=unit_aero_3_3,file='input_3_aero_3.txt',status='old')
        read(unit_aero_3_3,*,iostat=status) ((input_3_aero_3(i,j),j=1,column),i=1,row)
        write(*,*)"aero data=",k,",read result(0:success,other:fail):",status
      close(unit_aero_3_3)

      !檢查氣動力輸入資料
      Open(unit=unit_aero_check,file='output_check_3_aero.txt',status='unknown')
      !3-1
      row=11
      column=10
      d3=9
      do k=1,9
          if(k==1)then
            write(unit_aero_check,*)k,":CA(ON)"
          elseif(k==2)then
            write(unit_aero_check,*)k,":CA(OFF)"
          elseif(k==3)then
            write(unit_aero_check,*)k,":CN"
          elseif(k==4)then
            write(unit_aero_check,*)k,":XCP"
          elseif(k==5)then
            write(unit_aero_check,*)k,":CLFI"
          elseif(k==6)then
            write(unit_aero_check,*)k,":CHMAL"
          elseif(k==7)then
            write(unit_aero_check,*)k,":CHMAW"
          elseif(k==8)then
            write(unit_aero_check,*)k,":CNFAL"
          elseif(k==9)then
            write(unit_aero_check,*)k,":CNFAW"
          endif
          write(unit_aero_check,100) ("MACH=",(input_3_aero_1(i,j,k),j=1,column),i=1,1)
          write(unit_aero_check,100) ("AF=",(input_3_aero_1(i,j,k),j=1,column),i=2,row)
          write(unit_aero_check,"(/)")
      enddo
      !3-2
      write(unit_aero_check,101) "RL(M)=",input_3_aero_2(1)
      write(unit_aero_check,101) "RA(M^2)=",input_3_aero_2(2)
      write(unit_aero_check,101) "ALT(KM)=",input_3_aero_2(3)
      write(unit_aero_check,"(a9,e16.4)") "RE=",input_3_aero_2(4)
      write(unit_aero_check,"(/)")
      !3-3
      row=13
      column=9

      i=1
      write(unit_aero_check,102) i,"MACH=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
      write(unit_aero_check,102) i,"CNQ=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
      write(unit_aero_check,102) i,"CMQ=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
      write(unit_aero_check,102) i,"CLP=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
      write(unit_aero_check,102) i,"CND=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
      write(unit_aero_check,102) i,"XCPD=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
      write(unit_aero_check,102) i,"CLDP=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
      write(unit_aero_check,102) i,"CHMD=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
      write(unit_aero_check,102) i,"CNFD=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
      write(unit_aero_check,102) i,"XCPFD=",(input_3_aero_3(i,j),j=1,column)
      i=i+1
```

```
  write(unit_aero_check,102) i,"CND2=",(input_3_aero_3(i,j),j=1,column)
  i=i+1
  write(unit_aero_check,102) i,"XCPD2=",(input_3_aero_3(i,j),j=1,column)
  i=i+1
  write(unit_aero_check,102) i,"CLDP2=",(input_3_aero_3(i,j),j=1,column)

      format(a8,10f8.2)
      format(a9,f16.8)
      format(i3,a8,9f8.2)

  close(unit_aero_check)
  write(*,*)"---------------------------------------------------------------"
  return
end
!----------------------------------------------------------------------------!
!  副程式:初始資料計算(initial)                                              !
!  輸入:input_1_thrust                                                       !
!       input_2_physical                                                     !
!  輸出:position                                                             !
!       thrust                                                               !
!       physical                                                             !
!       dt                                                                   !
!       count                                                                !
!  定義:position(1)=x=飛彈在地面座標系g之x軸位置(km)                         !
!       position(2)=y=飛彈在地面座標系g之y軸位置(km)                         !
!       position(3)=z=飛彈在地面座標系g之z軸位置(km)                         !
!       velocity(1)=u=飛彈在地面座標系g之x軸速度(m/s)                        !
!       velocity(2)=v=飛彈在地面座標系g之y軸速度(m/s)                        !
!       velocity(3)=w=飛彈在地面座標系g之z軸速度(m/s)                        !
!       angular_velocity(1)=p=飛彈在地面座標系g之x角速度(rad/s)              !
!       angular_velocity(2)=q=飛彈在地面座標系g之y角速度(rad/s)              !
!       angular_velocity(3)=r=飛彈在地面座標系g之z角速度(rad/s)              !
!       Euler_angle_GD(1)=theta=俯仰角(rad)                                  !
!       Euler_angle_GD(2)=phi=偏航角(rad)                                    !
!       Euler_angle_GD(3)=gamma=傾斜角(rad)                                  !
!       thrust=內插完的推力                                                  !
!       physical=內插完的物性                                                !
!       dt=時間步長(s)                                                       !
!       counter=到燃畢時的計算次數                                           !
!----------------------------------------------------------------------------!
subroutine sub_initial()
  use global
  use constant
  use input_data
  use solution
  use unit
  implicit none
  integer i,j,k
  integer row,column
  real*8 thrust_test(50000,5)
  real*8 physical_test(50000,4)
  real*8 t_start,t_end
  real*8 A_up,A_mid,A_down
  real*8 B_up,B_mid,B_down
  real*8 mass_up,mass_down,mass_loss,mass_loss_rate
  real*8 mass_start,mass_end
  real*8 L_body,X_CG
  real*8 t_up,t_down

  write(*,*)"--------------------subroutine:initial data--------------------"

  !L_body=彈體長度(m)
  L_body=3.873
  !X_CG=彈體重心位置(m)
  X_CG=2.69

  !初始俯仰角=25度
  Euler_angle_GD(1)=25.0*PI/180.
  Euler_angle_GD(2)=0.0
  Euler_angle_GD(3)=0.0

  position(1)=0.0
  position(2)=0.0
```

```fortran
position(3)=-0.8

velocity(1)=0.0
velocity(2)=0.0
velocity(3)=0.0

angular_velocity(1)=0.0
angular_velocity(2)=0.0
angular_velocity(3)=0.0

write(*,"(a4,f10.4,a4)")"dt=",dt,"(s)"
write(*,300)"L_body(m)","X_CG(m)","L_body-X_CG(m)"
write(*,310)L_body,X_CG,L_body-X_CG
write(*,300)"theta(deg)","phi(deg)","gamma(deg)"
write(*,310)(Euler_angle_GD(i),i=1,3)
write(*,300)"x(m)","y(m)","z(m)"
write(*,310)(position(i),i=1,3)
write(*,300)"Vx(m/s)","Vy(m/s)","Vz(m/s)"
write(*,310)(velocity(i),i=1,3)
write(*,300)"p(deg/s)","q(deg/s)","r(deg/s)"
write(*,310)(angular_velocity(i)*PI/180.,i=1,3)

    format(3A20)
    format(3F20.4)
!--------------------內插--------------------!
write(*,*)"--------------------interpolation--------------------"
!1.推力內插
counter=(input_1_thrust(26,1)-input_1_thrust(1,1))/dt+1
write(*,*)"counter:",counter
t_start=input_1_thrust(1,1)
t_end=input_1_thrust(26,1)
row=counter
column=5
do j=1,5
  do i=1,row
    t=t_start+(i-1)*dt
    if((t-t_start)<1e-5)then
      thrust_test(i,j)=input_1_thrust(1,j)
    elseif(abs(t-t_end)<=1e-5)then
      thrust_test(i,j)=input_1_thrust(26,j)
    else
      do k=2,26
        t_up=input_1_thrust(k-1,1)
        t_down=input_1_thrust(k,1)
        if(t>=t_up .and. t<=t_down)then
          A_up   = t_up
          A_mid  = t
          A_down = t_down
          B_up   = input_1_thrust(k-1,j)
          B_down = input_1_thrust(k,j)
          B_mid  = B_up + ( (A_mid-A_up)/(A_down-A_up) )*(B_down-B_up)
          thrust_test(i,j) = B_mid
        endif
      enddo
    endif
  end do
end do
!輸出確認
open(unit=unit_thrust_interpolation_check,file="output_thrust_interpolation.txt")
  write(unit_thrust_interpolation_check,100) "t(sec)","p(kgf/cm^2)","TF_sea(kgf)","TF_vacuum(kgf)","dm/dt(kg/sec)"
  write(unit_thrust_interpolation_check,110)((thrust_test(i,j),j=1,column),i=1,row)
      format(5a16)
      format(5f16.2)
close(unit_thrust_interpolation_check)

!2.物性內插
row = counter
column = 4

factor = 1.0
do while(.true.)
  mass_start = input_2_physical(1,1)
```

```
    mass_end = input_2_physical(2,1)
    mass_loss = 0.0
    do i=1,row
      mass_loss_rate     = factor*thrust_test(i,5)
      mass_loss          = mass_loss + mass_loss_rate*dt
      mass               = mass_start - mass_loss
      physical_test(i,1) = mass
    end do
    if( abs(physical_test(row,1)-mass_end) <= 1e-2)then
      write(*,"(a10,f10.4)")"factor=",factor
      exit
    elseif((physical_test(row,1)-mass_end)>0)then
      factor = factor + (physical_test(row,1)-mass_end)/mass_end
    else
      factor = factor + (physical_test(row,1)-mass_end)/mass_end
    endif
  end do
  write(*,300)"mass_start(kg)","mass_end_compute","mass_end"
  write(*,310)mass_start,mass,mass_end
  write(*,"(a40,f20.4)")"mass_end_compute-mass_end(kg)=",physical_test(row,1)-mass_end

  do j=2,4
    physical_test(1,j)   = input_2_physical(1,j)
    physical_test(row,j) = input_2_physical(2,j)
    mass_up   = physical_test(1,1)
    mass_down = physical_test(row,1)
    A_up      = mass_up
    A_down    = mass_down
    B_up      = input_2_physical(1,j)
    B_down    = input_2_physical(2,j)
    do i=1,row-1
      A_mid = physical_test(i,1)
      B_mid = B_up + ( (A_mid-A_up)/(A_down-A_up) )*(B_down-B_up)
      physical_test(i,j) = B_mid
    end do
  end do
  !輸出確認

  open(unit=unit_physical_interpolation_check,file="output_physical_interpolation.txt")
    write(unit_physical_interpolation_check,200) "m(kg)","x_cg(m)","Ixx(kg-m^2)","Iyy(kg-m^2)"

    write(unit_physical_interpolation_check,210)((physical_test(i,j),j=1,column),i=1,row)
        format(4a16)
        format(4f16.2)
  close(unit_physical_interpolation_check)

  write(*,*)"--------------------Y initial--------------------"
  !設定初始值
  !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
  do i=1,3
    Y(i)   = velocity(i)
    Y(i+3) = position(i)
    Y(i+6) = angular_velocity(i)
    Y(i+9) = Euler_angle_GD(i)
  enddo
  Y(13) = input_2_physical(1,1)
  mass  = Y(13)
  write(*,"(i16,f16.2)")(i,Y(i),i=1,13)
  write(*,*)"--------------------------------------------------------------"

  return
end
!---------------------------------------------------------------------------!
!  副程式:6Dof方程式(6Dof)                                                   !
!  輸入:                                                                    !
!  輸出:                                                                    !
!  定義:                                                                    !
!---------------------------------------------------------------------------!
subroutine sub_6Dof()
  use global
```

```
    use input_data
    use solution
    implicit none
    integer i
    real*8  F_gravity(3)
    real*8  F_thrust(3)
    real*8  F_airforce(3)
    real*8  F_total(3)
    real*8  M_air(3)
    real*8  M_total(3)
    real*8  thrust(5)
    real*8  physical(4)

    M_air=0
    M_total=0
    F_airforce=0

    !進入次數
    call_times = call_times+1

    !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
    !變數更新
    do i=1,3
      velocity(i)         = Y(i)
      position(i)         = Y(i+3)
      angular_velocity(i) = Y(i+6)
      Euler_angle_GD(i)   = Y(i+9)
      mass                = Y(13)
    end do

    !--------------------標準大氣模式--------------------!
    !呼叫副程式:標準大氣模式
    call sub_atmosphere()

    !--------------------物性、推力參數--------------------!
    !呼叫副程式:物性、推力參數
    call sub_parameter(thrust,physical)

    !--------------------力及力矩--------------------!
    !呼叫副程式:重力(彈體座標系D)
    call sub_F_gravity(F_gravity)

    !呼叫副程式:推力(彈體座標系D)
    call sub_F_thrust(F_thrust,thrust)

    !呼叫副程式:氣動力及氣動力矩(彈體座標系D)
    call sub_F_airforce(F_airforce,M_air,physical)

    !呼叫副程式:總外力及力矩(彈體座標系D->大地座標系G)
    call sub_F_total(F_total,M_total,F_gravity,F_thrust,F_airforce,M_air)

    !呼叫副程式:尤拉角(大地座標系G與彈體座標系D)
    call sub_Euler_angle()

    !呼叫副程式:角加速度(彈體座標系D)
    call sub_angular_velocity(M_total,physical)

    !呼叫副程式:微分方程(大地座標系G)
    call sub_eqs(F_total)

    return
  end
  !---------------------------------------------------------------------------!
  !  副程式:標準大氣程式(atmosphere)                                          !
  !  輸入:position                                                            !
  !  輸出:P_air                                                               !
  !       rho                                                                 !
  !       Cs                                                                  !
  !  定義:g=9.80665=重力加速度(m/s^2)                                         !
  !       R_earth=6356.766=地球半徑(km)                                       !
  !       R=287.05287=比氣體常數(J/(K*kg))                                    !
  !       M=28.96442=分子量(kg/kmol)                                          !
  !       R_start=8314.32=通用氣體常數(J/(K*kg))                              !
```

```fortran
!       Z=地理高度(geometric altitude)(km)                                  !
!       H=位勢高度(geopotential altitude)(km)                               !
!       H_b=相應層底部的位勢高度(km)                                        !
!       P_b=相應層底部的大氣壓力(Pa或N/m^2)                                 !
!       T_b=相應層底部的大氣溫度(K)                                         !
!       M_b=相應層底部的分子量(kg/kmol)                                     !
!       Temperature=溫度(K)                                                 !
!       beta=相應層沿位勢高度的溫度梯度(K/km)                               !
!       P_air=大氣壓力(Pa或N/m^2)                                           !
!       rho=大氣密度(kg/m^3)                                                !
!       Cs=大氣音速(m/s)                                                    !
!       position(1)=x=飛彈在地面座標系g之x軸位置(km)                        !
!       position(2)=y=飛彈在地面座標系g之y軸位置(km)                        !
!       position(3)=z=飛彈在地面座標系g之z軸位置(km)                        !
!  備註:參考"U.S. STANDARD ATMOSPHERE,1962"                                 !
!--------------------------------------------------------------------------!
subroutine sub_atmosphere()
  use global
  use input_data
  use module_atmosphere
  use solution
  use unit
  implicit none
  real*8, parameter :: g=9.80665,R=287.05287,R_start=8314.32,R_earth=6356.766,M=28.96442
  real*8  Z,H
  real*8  Temperature
  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)
  real*8  H_b,T_b,beta,P_b,M_b
  real*8  beta_m,H_m,H_b_m
  integer i,j
  integer row,column
  integer status

  !(1)讀取大氣溫度參數
  row = 22
  column = 6
  if(i_main==1)then
    open(unit=unit_atmosphere, file="input_4_atmosphere.txt")
    read(unit_atmosphere,*,iostat=status)((input_4_atmosphere(i,j),j=1,column),i=1,row)
  !  write(*,*)"大氣資料=",",讀取結果(0:成功,other:失敗):",status
    close(unit_atmosphere)
  endif
  do i=1,row
    input_Z_b(i)  = input_4_atmosphere(i,1)
    input_H_b(i)  = input_4_atmosphere(i,2)
    input_T_b(i)  = input_4_atmosphere(i,3)
    input_beta(i) = input_4_atmosphere(i,4)
    input_P_b(i)  = input_4_atmosphere(i,5)
    input_M_b(i)  = input_4_atmosphere(i,6)
  end do

  !輸出大氣表
  if(i_main==1)then
    open(unit=unit_atmosphere_check,file="output_check_4_atmospher.txt")
    write(unit_atmosphere_check,100)"Z_b(km)","H_b(km)","T_b(K)","beta(K/km)","P_b(pa)","M_b(kg/kmol)"
    write(unit_atmosphere_check,110)(input_Z_b(i),input_H_b(i),input_T_b(i),input_beta(i),input_P_b(i),input_M_b(i),i=1,row)
        format(6a14)
        format(4f14.4,e14.4,f14.4)
    close(unit_atmosphere_check)
  endif

  !(2)計算高度位勢(Z為高度，向下為正，因此加-號)
  Z = -position(3)/1000.0
  H = (R_earth*Z)/(R_earth+Z)

  !(3)依高度判斷在哪一層
  do i=1,row-1
    if(H>=input_H_b(i) .and. H<input_H_b(i+1))then
      H_b  = input_H_b(i)
      T_b  = input_T_b(i)
      beta = input_beta(i)
      P_b  = input_P_b(i)
      M_b  = input_M_b(i)
    elseif(i==(row-1) .and. H>=input_H_b(row))then
      H_b  = input_H_b(row)
      T_b  = input_T_b(row)
      beta = input_beta(row)
      P_b  = input_P_b(row)
      M_b  = input_M_b(row)
    elseif(H<input_H_b(1))then
  !    write(*,*)"位勢高度:",H,"超出範圍(H<-2km)"
    endif
  end do

  !(4)溫度
  Temperature = T_b + beta*(H-H_b)

  !(5)壓力
  !將單位由km轉成m
  beta_m = beta/1000.0
  H_m = H*1000.0
  H_b_m = H_b*1000.0
  if(beta==0.0)then
    P_air = P_b*exp( (-g/(R*T))*(H_m-H_b_m) )
  else
    P_air = P_b*( 1.0+(beta_m/T_b)*(H_m-H_b_m) )**( -g/(R*beta_m) )
  endif

  !(6)密度
  rho = P_air/(R*Temperature)

  !(7)音速
  !k=1.4為比熱常數k=Cp/Cv
  Cs = ( 1.4*(R_start/M)*Temperature )**0.5

  !(8)馬赫數
  V = ( velocity(1)**2 + velocity(2)**2 + velocity(3)**2 )**0.5
  Ma = V/Cs

  !check
  if(.true.)then
    if(mod(call_times-1,4)==0.0)then
      if(i_main==1)then
        open(unit=unit_atmosphere_debug,file="output_debug_1_atmosphere.txt")
        write(unit_atmosphere_debug,200)"t","Z(km)","H(km)","T(K)","P_air(pa)"&
        &,"rho(kg/m^3)","Cs(m/s)","Ma"
      endif
      write(unit_atmosphere_debug,210)(i_main-1)*dt,Z,H,Temperature,P_air,rho,Cs,Ma
          format(a6,7a12)
          format(f6.2,3f12.2,2e12.2,2f12.2)
    endif
  endif
  return
end
!---------------------------------------------------------------------------!
!  副程式:物性、推力參數                                                    !
!  輸入:input_1_thrust                                                      !
!       input_2_physical                                                    !
!       t                                                                   !
!       dt                                                                  !
!       i_main                                                              !
!       counter                                                             !
!  輸出:thrust                                                              !
!       physical                                                            !
!  定義:                                                                    !
!---------------------------------------------------------------------------!
subroutine sub_parameter(thrust,physical)
  use global
  use input_data
  use solution
  use unit
  implicit none
  real*8  thrust(5)
  real*8  physical(4)
  real*8  A,A1,A2
  real*8  B,B1,B2
  integer i,j

  !燃畢前
  if(i_main<counter)then
    !推力資料的內插
    do j=1,5
      do i=1,26-1
        if(input_1_thrust(i,1)<=t .and. t<=input_1_thrust(i+1,1))then
          A  = t
          A1 = input_1_thrust(i,1)
          A2 = input_1_thrust(i+1,1)
          B1 = input_1_thrust(i,j)
          B2 = input_1_thrust(i+1,j)
          !呼叫副程式:1維內插
          call sub_interpolation_1D(A,A1,A2,B,B1,B2)
          thrust(j) = B
        endif
      enddo
    enddo
    !物性資料的內插
    do j=1,4
      do i=1,1
        if(mass<=input_2_physical(i,1) .and. mass>=input_2_physical(i+1,1))then
          A  = mass
          A1 = input_2_physical(i,1)
          A2 = input_2_physical(i+1,1)
          B1 = input_2_physical(i,j)
          B2 = input_2_physical(i+1,j)
          !呼叫副程式:1維內插
          call sub_interpolation_1D(A,A1,A2,B,B1,B2)
          physical(j) = B
        endif
      enddo
    enddo
  !燃畢
  else
    do i=1,5
        thrust(i) = input_1_thrust(26,i)
    enddo
    do i=1,4
        physical(i) = input_2_physical(2,i)
    enddo
  endif
  if(i_main<counter)then
    dY_dt(13) = -factor*thrust(5)
  else
    dY_dt(13) = 0.0
    Y(13) = input_2_physical(2,1)
    mass = input_2_physical(2,1)
  endif

  !check
  if(mod(call_times-1,4)==0.0)then
    if(i_main==1)then
      open(unit=unit_out_physical_debug,file="output_debug_2_physical.txt")
      write(unit_out_physical_debug,200)"t","mass","XCG","Ixx","Iyy",&
      &"p","TF_sea","TF_vacuum","dm/dt"
      write(unit_out_physical_debug,200)"s","kg","m","kg-m^2","kg-m^2",&
      &"kgf/cm^2","kgf","kgf","kg/sec"
    endif
    write(unit_out_physical_debug,210)t,physical(1),physical(2),physical(3),&
    physical(4),thrust(2),thrust(3),thrust(4),thrust(5)
        format(a6,8a10)
        format(f6.2,8f10.2)
  endif
```

```
  return
end
!----------------------------------------------------------------------------!
!  副程式:重力(彈體座標系D)                                                  !
!  輸入:mass                                                                 !
!  輸出:F_gravity                                                            !
!  定義:g=9.80665=重力加速度(m/s^2)                                          !
!        F_gravity=重力(N)                                                   !
!        mass=重量(kg)                                                       !
!  備註:Z軸是朝下為正，因此重力加速度為正值(>0)                              !
!----------------------------------------------------------------------------!
subroutine sub_F_gravity(F_gravity)
  use solution
  use global
  use constant
  implicit none
  integer i
  real*8  X_G(3),X_D(3)
  real*8  F_gravity(3)

  !大地座標系G下的重力
  X_G(1)=0.0
  X_G(2)=0.0
  X_G(3)=mass*g

  !呼叫副程式:座標轉換(地面座標系G->彈體座標系D)
  call sub_Coordinate_Trans_GD(Euler_angle_GD,X_G,X_D)

  !彈體座標系D下的重力
  do i=1,3
    F_gravity(i)=X_D(i)
  enddo

  !check
  if(.false.)then
    write(*,200)"mass","g"
    write(*,210)mass,g
    write(*,200)"X_G(1)","X_G(2)","X_G(3)"
    write(*,210)(X_G(i),i=1,3)
    write(*,200)"F_gravity(1)","F_gravity(2)","F_gravity(3)"
    write(*,210)(F_gravity(i),i=1,3)
    write(*,210)sqrt(F_gravity(1)**2+F_gravity(2)**2+F_gravity(3)**2)
        format(3a16)
        format(3f16.2)
  endif

  return
end
!----------------------------------------------------------------------------!
!  副程式:推力(彈體座標系D)                                                  !
!  輸入:thrust                                                               !
!        position                                                            !
!  輸出:F_thrust                                                             !
!  定義:F_thrust=推力(N)                                                     !
!        pressure_air=空氣壓力(pa)                                           !
!        Area_exit=噴嘴面積(m^2)(噴嘴直徑為0.182m)                           !
!  備註:當推力<0或燃畢後，推力為0                                            !
!        推力單位為牛頓(N)，真空推力單位為(kgf)                              !
!----------------------------------------------------------------------------!
subroutine sub_F_thrust(F_thrust,thrust)
  use global
  use module_atmosphere
  use constant
  use unit
  implicit none
  real*8, parameter :: Area_exit=(PI*(0.182**2))/4.0
  real*8  F_thrust(3)
  real*8  thrust(5)
  real*8  F_T
  integer i

  !燃畢後推力=0，推力單位是kgf要轉乘N
  if(i_main<counter)then
```

```fortran
    F_T=thrust(4)*g-P_air*Area_exit
  else
    F_T=0.0
  endif

  !推力<0.0
  if(F_T<0.0)then
    F_T=0.0
  endif

  !彈體座標系D下的推力
  F_thrust(1)=F_T
  F_thrust(2)=0.0
  F_thrust(3)=0.0

  !check
  if(.true.)then
    if(mod(call_times-1,4)==0.0)then
      if(i_main==1)then
        open(unit=unit_output_thrust_debug,file="output_debug_3_thrust.txt")
        write(unit_output_thrust_debug,200)"t","F_thrust(1)","F_thrust(2)",&
        &"F_thrust(3)","thrust(4)","P_air","Area_exit"
      endif

      write(unit_output_thrust_debug,210)t,(F_thrust(i),i=1,3),thrust(4),P_air,Area_exit
          format(a6,6a12)
          format(f6.2,4f12.2,e12.2,f12.4)
    endif
  endif
  return
end
!---------------------------------------------------------------------------!
!  副程式:氣動力及氣動力矩(彈體座標系D)                                     !
!  輸入:i_main                                                              !
!       counter                                                             !
!       input_3_aero_1                                                      !
!       input_3_aero_2                                                      !
!       input_3_aero_3                                                      !
!       physical                                                            !
!       velocity                                                            !
!       angular_velocity                                                    !
!       rho                                                                 !
!       Cs                                                                  !
!  輸出:F_airforce                                                          !
!       M_air                                                               !
!  定義:F_airforce(1)=彈體座標系x軸的氣動力(N)                              !
!       F_airforce(2)=彈體座標系y軸的氣動力(N)                              !
!       F_airforce(3)=彈體座標系z軸的氣動力(N)                              !
!       M_air(1)=彈體座標系x軸的氣動力矩(N-m)                               !
!       M_air(2)=彈體座標系y軸的氣動力矩(N-m)                               !
!       M_air(3)=彈體座標系z軸的氣動力矩(N-m)                               !
!       Cf(i)=合成氣動力係數                                                !
!       Cm(i)=合成氣動力矩係數                                              !
!       L_ref=參考長度(m)                                                   !
!       S_ref=參考面積(m^2)                                                 !
!       Q=動壓=0.5*rho*V^2                                                  !
!       Ma=馬赫數                                                           !
!       AF=總攻角(deg)                                                      !
!       phi_c=風向角(deg)                                                   !
!       XCG=彈體重心在彈體座標系x軸的位置(m)                                !
!       velocity_D=彈體座標系下的速度(m/s)                                  !
!       Cf(i)=合成氣動力係數                                                !
!       Cm(i)=合成氣動力矩係數                                              !
!---------------------------------------------------------------------------!
subroutine sub_F_airforce(F_airforce,M_air,physical)
  use global
  use input_data
  use module_atmosphere
  use solution
  implicit none
  integer, parameter :: unit_output_air_1_debug=121,unit_output_air_2_debug=122
  integer i,j,k
```

```
      integer row,column,D3
      real*8  F_airforce(3)
      real*8  M_air(3)
      real*8  physical(4)
      real*8  X_G(3),X_D(3)
      real*8  AF_range(11),Ma_range(11)
      real*8  A,A1,A2
      real*8  B,B1,B2
      real*8  C,C1,C2
      real*8  AF1,AF2
      real*8  Ma1,Ma2
      real*8  CA,CN,XCP,CLFI,CHMAL,CHMAW,CNFAL,CNFAW
      real*8  CNQ,CMQ,CLP,CND,XCPD,CLDP,CHMD,CNFD,XCPFD,CND2,XCPD2,CLDP2
      real*8  XCG
      real*8  L_ref,S_ref
      real*8  Q
      real*8  Cf(3),Cm(3)

      !大地座標系G下的速度
      do i=1,3
        X_G(i)=velocity(i)
      enddo

      !呼叫副程式:座標轉換(地面座標系G->彈體座標系D)
      call sub_Coordinate_Trans_GD(Euler_angle_GD,X_G,X_D)

      !彈體座標系D下的速度
      do i=1,3
        velocity_D(i)=X_D(i)
      enddo

      L_ref=input_3_aero_2(1)
      S_ref=input_3_aero_2(2)
      Q=0.5*rho*(V**2)

      XCG=physical(2)/L_ref

      !計算總攻角AF,風向角phi_c
      if(V==0.0)then
        AF=0.0
      else
        AF=acosd(velocity_D(1)/V)
      endif

      if(velocity_D(3)==0.0)then
        phi_c=0.0
      else
        phi_c=atan2(velocity_D(2),velocity_D(3))
      endif

      if(velocity_D(1)==0.0)then
        apha=0.0
      else
        apha=atan2(velocity_D(3),velocity_D(1))
      endif

      !(3-1)氣動力係數-1
      row=11
      column=10
      D3=9

      !依據AF,Ma決定內插位置
      do i=1,row
        AF_range(i)=input_3_aero_1(i,1,1)
      enddo
      do j=1,column
        Ma_range(j)=input_3_aero_1(1,j,1)
      enddo

      do i=2,row-1
        if(AF>=AF_range(i) .and. AF<=AF_range(i+1))then
          exit
        endif
```

```fortran
  enddo
  if(AF<AF_range(2))then
    i=2
    AF=AF_range(2)
  elseif(AF>AF_range(row))then
    i=row-1
    AF=AF_range(row)
  endif
!  write(*,*)"AF",AF_range(2),AF_range(row),row-2+1

  do j=2,column-1
    if(Ma>=Ma_range(j) .and. Ma<=Ma_range(j+1))then
      exit
    endif
  enddo
  if(Ma<Ma_range(2))then
    j=2
    Ma=Ma_range(2)
  elseif(Ma>Ma_range(column))then
    j=column-1
    Ma=Ma_range(column)
  endif
!  write(*,*)"Ma",Ma_range(2),Ma_range(column),column-2+1

  AF1=AF_range(i)
  AF2=AF_range(i+1)
  Ma1=Ma_range(j)
  Ma2=Ma_range(j+1)

  do k=1,D3
    A1=input_3_aero_1(i,j,k)
    A2=input_3_aero_1(i+1,j,k)
    B1=input_3_aero_1(i,j+1,k)
    B2=input_3_aero_1(i+1,j+1,k)
    !呼叫副程式:2維內插
    call sub_interpolation_2D(A,A1,A2,B,B1,B2,C,C1,C2,AF1,AF2,Ma1,Ma2)
    if(k==1 .and. i_main<=counter)then
      CA=C
    elseif(k==2 .and. i_main>counter)then
      CA=C
    elseif(k==3)then
      CN=C
    elseif(k==4)then
      XCP=C
    elseif(k==5)then
      CLFI=C
    elseif(k==6)then
      CHMAL=C
    elseif(k==7)then
      CHMAW=C
    elseif(k==8)then
      CNFAL=C
    elseif(k==9)then
      CNFAW=C
    endif
  enddo

  !(3-3)氣動力係數-3
  row=13
  column=9

  A=Ma
  A1=input_3_aero_3(1,j)
  A2=input_3_aero_3(1,j+1)

  do i=2,row
    B1=input_3_aero_3(i,j)
    B2=input_3_aero_3(i,j+1)
    !呼叫副程式:1維內插
    call sub_interpolation_1D(A,A1,A2,B,B1,B2)
    if(i==2)then
      CNQ=B
    elseif(i==3)then
```

```fortran
      CMQ=B
    elseif(i==4)then
      CLP=B
    elseif(i==5)then
      CND=B
    elseif(i==6)then
      XCPD=B
    elseif(i==7)then
      CLDP=B
    elseif(i==8)then
      CHMD=B
    elseif(i==9)then
      CNFD=B
    elseif(i==10)then
      XCPFD=B
    elseif(i==11)then
      CND2=B
    elseif(i==12)then
      XCPD2=B
    elseif(i==13)then
      CLDP2=B
    endif
  enddo

  !計算氣動力和力矩
  if(V==0.0)then
    Cf(1)=-CA
    Cf(2)=-CN*sin(phi_c)
    Cf(3)=-CN*cos(phi_c)

    Cm(1)=0.0
    Cm(2)=CN*(XCG-XCP)*cos(phi_c)
    Cm(3)=-CN*(XCG-XCP)*sin(phi_c)
  else
    Cf(1)=-CA
    Cf(2)=-CN*sin(phi_c)+CNQ*(0.5*angular_velocity(3)*L_ref/V)
    Cf(3)=-CN*cos(phi_c)-CNQ*(0.5*angular_velocity(2)*L_ref/V)

    Cm(1)=CLP*(0.5*angular_velocity(1)*L_ref/V)
    Cm(2)=CN*(XCG-XCP)*cos(phi_c)+CMQ*(0.5*angular_velocity(2)*L_ref/V)
    Cm(3)=-CN*(XCG-XCP)*sin(phi_c)+CMQ*(0.5*angular_velocity(3)*L_ref/V)
  endif
  do i=1,3
    F_airforce(i)=Cf(i)*Q*S_ref
    M_air(i)=Cm(i)*Q*S_ref*L_ref
  enddo

  !check
  if(.true.)then
    if(mod(call_times-1,4)==0.0)then
      if(i_main==1)then
        open(unit=unit_output_air_1_debug,file="output_debug_4_airforce_1.txt")
        write(unit_output_air_1_debug,100)"t","Cf(1)","Cf(2)","Cf(3)",&
                                         &"Cm(1)","Cm(2)","Cm(3)"

        open(unit=unit_output_air_2_debug,file="output_debug_5_airforce_2.txt")
        write(unit_output_air_2_debug,300)"t","phi_c","AF","MA","CA","CN","CNQ",&
                                         &"CLP","CMQ","XCP","XCG"
      endif
      write(unit_output_air_1_debug,200)t,(Cf(i),i=1,3),(Cm(i),i=1,3)
      write(unit_output_air_2_debug,400)t,phi_c,AF,Ma,CA,CN,CNQ,CLP,CMQ,XCP,XCG

          format(a6,6a12)
          format(f6.2,6f12.2)
          format(a6,10a12)
          format(f6.2,10f12.2)
    endif
  endif
  return
end
!---------------------------------------------------------------------------!
!  副程式:總外力及力矩(彈體座標系D->大地座標系G)                            !
!  輸入:F_gravity(i)                                                        !
!       F_thrust(i)                                                           !
!       F_airforce(i)                                                         !
!       M_air(i)                                                              !
!       Euler_angle_GD(i)                                                     !
!       i_main                                                                !
!       counter                                                               !
!  輸出:F_total                                                               !
!       M_total                                                               !
!  定義:F_total(1)=大地座標系X軸之總外力(N)                                   !
!       F_total(2)=大地座標系Y軸之總外力(N)                                   !
!       F_total(3)=大地座標系Z軸之總外力(N)                                   !
!       M_total(1)=大地座標系X軸之總外力矩(N-m)                               !
!       M_total(2)=大地座標系Y軸之總外力矩(N-m)                               !
!       M_total(3)=大地座標系Z軸之總外力矩(N-m)                               !
!       F_gravity(i)=彈體座標系下之重力(N)                                    !
!       F_thrust(i)=彈體座標系下之推力(N)                                     !
!       F_airforce(i)=彈體座標系下之氣動力(N)                                 !
!       M_air(i)=彈體座標系下之氣動力矩(N-m)                                  !
!       Euler_angle_GD(i)=彈體座標系D與大地座標系G之尤拉角                    !
!       i_main=目前的計算次數                                                 !
!       counter=到燃畢時的計算次數                                            !
!-----------------------------------------------------------------------------!
subroutine sub_F_total(F_total,M_total,F_gravity,F_thrust,F_airforce,M_air)
  use global
  use solution
  use unit
  implicit none
  integer i
  real*8  F_total(3)
  real*8  M_total(3)
  real*8  F_gravity(3)
  real*8  F_thrust(3)
  real*8  F_airforce(3)
  real*8  M_air(3)
  real*8  X_D(3),X_G(3)

  !彈體座標系D下的總外力
  do i=1,3
    X_D(i)=F_gravity(i)+F_thrust(i)+F_airforce(i)
  enddo

  !呼叫副程式:座標轉換(彈體座標系D->地面座標系G)
  call sub_Coordinate_Trans_DG(Euler_angle_GD,X_G,X_D)

  !彈體座標系G下的總外力
  do i=1,3
    F_total(i)=X_G(i)
  enddo

  !彈體座標系D下的總外力矩
  do i=1,3
    M_total(i)=M_air(i)
  enddo

  if(.false.)then
  write(*,220)t,(F_total(i),i=1,3)
  write(*,220)t,(F_gravity(i),i=1,3)
  write(*,220)t,(F_thrust(i),i=1,3)
  write(*,220)t,(F_airforce(i),i=1,3)
  write(*,220)t,(M_total(i),i=1,3)
      format(f6.2,3f12.2)
  endif

  !check
  if(.true.)then
    if(mod(call_times-1,4)==0.0)then
      if(i_main==1)then
        open(unit=unit_F_M_debug,file="output_debug_6_FM_.txt")
        write(unit_F_M_debug,200)"t","F_total(1)","F_total(2)","F_total(3)",&
        &"F_gravity(1)","F_gravity(2)","F_gravity(3)",&
        &"F_thrust(1)","F_thrust(2)","F_thrust(3)",&
        &"F_airforce(1)","F_airforce(2)","F_airforce(3)",&
        &"M_total(1)","M_total(2)","M_total(3)"
      endif
      write(unit_F_M_debug,210)t,(F_total(i),i=1,3),(F_gravity(i),i=1,3),&
      &(F_thrust(i),i=1,3),(F_airforce(i),i=1,3),(M_total(i),i=1,3)
          format(a6,15a11)
          format(f6.2,15f11.2)
    endif
  endif
  return
end
!----------------------------------------------------------------------------!
!  副程式:尤拉角(大地座標系G與彈體座標系D)                                   !
!  輸入:angular_velocity(i)                                                  !
!       Euler_angle_GD(i)                                                    !
!  輸出:Euler_angle_rate_GD                                                  !
!  定義:angular_velocity(1)=p=飛彈在地面座標系g之x角速度(rad/s)              !
!       angular_velocity(2)=q=飛彈在地面座標系g之y角速度(rad/s)              !
!       angular_velocity(3)=r=飛彈在地面座標系g之z角速度(rad/s)              !
!       Euler_angle_GD(1)=theta=俯仰角(rad)                                  !
!       Euler_angle_GD(2)=phi=偏航角(rad)                                    !
!       Euler_angle_GD(3)=gamma=傾斜角(rad)                                  !
!----------------------------------------------------------------------------!
subroutine sub_Euler_angle()
  use solution
  implicit none
  real*8  theta,phi,gamma
  real*8  P,Q,R

  theta=Euler_angle_GD(1)
  phi=Euler_angle_GD(2)
  gamma=Euler_angle_GD(3)

  P=angular_velocity(1)
  Q=angular_velocity(2)
  R=angular_velocity(3)

  Euler_angle_rate_GD(1)=Q*cos(gamma)-R*sin(gamma)
  Euler_angle_rate_GD(2)=( Q*sin(gamma)+R*cos(gamma) )/cos(theta)
  Euler_angle_rate_GD(3)=P+( Q*sin(gamma)+R*cos(gamma) )*tan(theta)

  return
end
!----------------------------------------------------------------------------!
!  副程式:角加速度(彈體座標系D)                                              !
!  輸入:angular_velocity(i)                                                  !
!       Euler_angle_GD(i)                                                    !
!  輸出:Euler_angle_rate_GD                                                  !
!  定義:angular_velocity(1)=p=飛彈在地面座標系g之x角速度(rad/s)              !
!       angular_velocity(2)=q=飛彈在地面座標系g之y角速度(rad/s)              !
!       angular_velocity(3)=r=飛彈在地面座標系g之z角速度(rad/s)              !
!       Euler_angle_GD(1)=theta=俯仰角(rad)                                  !
!       Euler_angle_GD(2)=phi=偏航角(rad)                                    !
!       Euler_angle_GD(3)=gamma=傾斜角(rad)                                  !
!----------------------------------------------------------------------------!
subroutine sub_angular_velocity(M_total,physical)
  use solution
  use global
  implicit none
  integer i,j
  real*8  M_total(3)
  real*8  physical(4)
  real*8  I_moment(3,3),I_moment_inv(3,3)
  real*8  Ixx,Iyy,Izz,Ixy,Iyz,Ixz
  real*8  P,Q,R
  real*8  A(3),B(3,3)

  Ixx=physical(3)
  Iyy=physical(4)
  Izz=physical(4)
  Ixy=0.0
  Iyz=0.0
  Ixz=0.0

  I_moment(1,1)=Ixx
```

```fortran
    I_moment(1,2)=-Ixy
    I_moment(1,3)=-Ixz
    I_moment(2,1)=I_moment(1,2)
    I_moment(2,2)=Iyy
    I_moment(2,3)=-Iyz
    I_moment(3,1)=I_moment(1,3)
    I_moment(3,2)=I_moment(2,3)
    I_moment(3,3)=Izz

    P=angular_velocity(1)
    Q=angular_velocity(2)
    R=angular_velocity(3)

    A(1)=M_total(1)-(Izz-Iyy)*Q*R+Iyz*(Q**2-R**2)+Ixz*P*Q-Ixy*P*R
    A(2)=M_total(2)-(Ixx-Izz)*P*R+Ixz*(R**2-P**2)+Ixy*Q*R-Iyz*P*Q
    A(3)=M_total(3)-(Iyy-Ixx)*P*Q+Ixy*(P**2-Q**2)+Iyz*P*R-Ixz*Q*R

    !慣性矩反矩陣
    !呼叫副程式:反矩陣
    call inverse_matrix(I_moment,I_moment_inv)
    if(.false.)then
      write(*,*)  "I_moment="
      write(*,200)((I_moment(i,j),j=1,3),i=1,3)
      write(*,*)  "I_moment_inv="
      write(*,200)((I_moment_inv(i,j),j=1,3),i=1,3)
      B=matmul(I_moment,I_moment_inv)
      write(*,*)  "I_moment*I_moment_inv="
      write(*,200)((B(i,j),j=1,3),i=1,3)
          format(3f12.4)
    endif

    !計算角速度(彈體座標系D)
    angular_velocity_rate=matmul(I_moment_inv,A)

    return
  end
  !----------------------------------------------------------------------------!
  !  副程式:微分方程(大地座標系G)                                              !
  !  輸入:velocity                                                             !
  !       position                                                             !
  !       F_total                                                              !
  !       mass                                                                 !
  !       angular_velocity                                                     !
  !       angular_velocity_rate                                                !
  !       Euler_angle_GD                                                       !
  !       Euler_angle_rate_GD                                                  !
  !       physicals                                                            !
  !  輸出:Y                                                                    !
  !       dY_dt                                                                !
  !  定義:Y=積分前的物理量                                                     !
  !       Y(1)=大地座標系下x軸方向之速度                                       !
  !       Y(2)=大地座標系下y軸方向之速度                                       !
  !       Y(3)=大地座標系下z軸方向之速度                                       !
  !       Y(4)=大地座標系下x軸方向之位置                                       !
  !       Y(5)=大地座標系下y軸方向之位置                                       !
  !       Y(6)=大地座標系下z軸方向之位置                                       !
  !       Y(7)=大地座標系下x軸方向之角速度                                     !
  !       Y(8)=大地座標系下y軸方向之角速度                                     !
  !       Y(9)=大地座標系下z軸方向之角速度                                     !
  !       Y(10)=尤拉角(1)俯仰角(theta)                                         !
  !       Y(11)=尤拉角(2)偏航角(phi)                                           !
  !       Y(12)=尤拉角(3)傾斜角(gamma)                                         !
  !       Y(13)=重量(kg)                                                       !
  !       dY_dt=積分前的物理量的斜率                                           !
  !       dY_dt(1)=大地座標系下x軸方向之加速度                                 !
  !       dY_dt(2)=大地座標系下y軸方向之加速度                                 !
  !       dY_dt(3)=大地座標系下z軸方向之加速度                                 !
  !       dY_dt(4)=大地座標系下x軸方向之速度                                   !
  !       dY_dt(5)=大地座標系下y軸方向之速度                                   !
  !       dY_dt(6)=大地座標系下z軸方向之速度                                   !
  !       dY_dt(7)=大地座標系下x軸方向之角加速度                               !
  !       dY_dt(8)=大地座標系下y軸方向之角加速度                               !
  !       dY_dt(9)=大地座標系下z軸方向之角加速度                               !
```

```
!       dY_dt(10)=尤拉角(1)俯仰角(theta)斜率                                 !
!       dY_dt(11)=尤拉角(2)偏航角(phi)斜率                                   !
!       dY_dt(12)=尤拉角(3)傾斜角(gamma)斜率                                 !
!       dY_dt(13)=重量斜率(kg/s)                                             !
!       I_moment=慣性矩(3*3)                                                 !
!       I_moment_inv=慣性矩反矩陣(3*3)                                       !
!----------------------------------------------------------------------------!
subroutine sub_eqs(F_total)
  use solution
  implicit none
  integer i
  real*8  F_total(3)

  !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

  !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
  do i=1,3
    Y(i)=velocity(i)
    Y(i+3)=position(i)
    Y(i+6)=angular_velocity(i)
    Y(i+9)=Euler_angle_GD(i)

    dY_dt(i)=F_total(i)/mass
    dY_dt(i+3)=velocity(i)
    dY_dt(i+6)=angular_velocity_rate(i)
    dY_dt(i+9)=Euler_angle_rate_GD(i)
  enddo
  Y(13)=mass

  !check
  if(.false.)then
    write(*,200)(Y(i),i=1,13)
    write(*,200)(dY_dt(i),i=1,13)

        format(13f8.2)
  endif

  return
end
!----------------------------------------------------------------------------!
!  副程式:數值積分(Runge Kutta Method,4階)                                   !
!  輸入:Y                                                                    !
!       dY_dt                                                                !
!       dt                                                                   !
!  輸出:Y                                                                    !
!  定義:Y_old=Y(t(j))=積分前的Y                                              !
!       Y=Y(t(j+1))=積分後的Y                                                !
!       dY_dt=Y的斜率                                                        !
!       t=時間(s)                                                            !
!       dt=時間步長(s)                                                       !
!       Slope_1=Runge Kutta的係數(斜率)                                      !
!       Slope_2=Runge Kutta的係數(斜率)                                      !
!       Slope_3=Runge Kutta的係數(斜率)                                      !
!       Slope_4=Runge Kutta的係數(斜率)                                      !
!----------------------------------------------------------------------------!
subroutine sub_integral()
  use global
  use solution
  implicit none
  integer i,N
  real*8  Y_old(13)
  real*8  Slope_1(13),Slope_2(13),Slope_3(13),Slope_4(13)
  real*8  t_old

  !需積分的變數個數
  N=13
  !儲存上一個時間點的函數值與時間(共N個需積分的函數)
  do i=1,N
    Y_old(i)=Y(i)
    t_old=t
  enddo

  !計算Slope_1
  do i=1,N
    Slope_1(i)=dY_dt(i)
  enddo

  !計算Slope_2(前進0.5dt)
  t=t_old+0.5*dt
  do i=1,N
    !計算新函數值
    Y(i)=Y_old(i)+0.5*dt*Slope_1(i)
  enddo
  !呼叫副程式:6Dof方程式
  call sub_6Dof()
  do i=1,N
    Slope_2(i)=dY_dt(i)
  enddo

  !計算Slope_3(前進0.5dt)
  t=t_old+0.5*dt
  do i=1,N
    !計算新函數值
    Y(i)=Y_old(i)+0.5*dt*Slope_2(i)
  enddo
  !呼叫副程式:6Dof方程式
  call sub_6Dof()
  do i=1,N
    Slope_3(i)=dY_dt(i)
  enddo

  !計算Slope_4(前進dt)
  t=t_old+dt
  do i=1,N
    !計算新函數值
    Y(i)=Y_old(i)+dt*Slope_3(i)
  enddo
  !呼叫副程式:6Dof方程式
  call sub_6Dof()
  do i=1,N
    Slope_4(i)=dY_dt(i)
  enddo

  !計算下一個時間點的Y
  do i=1,N
    Y(i)=Y_old(i)+(dt/6.0)*(Slope_1(i)+2.*Slope_2(i)+2.*Slope_3(i)+Slope_4(i))
  enddo

  return
end
!---------------------------------------------------------------------------!
!  副程式:座標轉換(地面座標系G->彈體座標系D)                                !
!  輸入:Euler_angle_GD                                                      !
!       X_G                                                                 !
!  輸出:X_D                                                                 !
!  定義:Euler_angle_GD(1)=theta=俯仰角(rad)                                 !
!       Euler_angle_GD(2)=phi=偏航角(rad)                                   !
!       Euler_angle_GD(3)=gamma=傾斜角(rad)                                 !
!       Trans_GD(3,3)=地面座標系G->彈體座標系D之轉換矩陣                    !
!       X_G(1)=變數在地面座標系G之X軸方向投影量                             !
!       X_G(2)=變數在地面座標系G之Y軸方向投影量                             !
!       X_G(3)=變數在地面座標系G之Z軸方向投影量                             !
!       X_D(1)=變數在彈體座標系D之X軸方向投影量                             !
!       X_D(2)=變數在彈體座標系D之Y軸方向投影量                             !
!       X_D(3)=變數在彈體座標系D之Z軸方向投影量                             !
!  備註:大地座標系OXgYgZg固定於地球上，座標系原點O為發射點                  !
!       OXg軸朝北為正，OYg軸朝東為正，OZg軸朝下為正。                       !
!  備註:彈體座標系OXYZ固定於彈體，座標系原點O為彈體重心                     !
!       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。        !
!---------------------------------------------------------------------------!
subroutine sub_Coordinate_Trans_GD(Euler_angle_GD,X_G,X_D)
  implicit none
  integer i,j
  real*8  Euler_angle_GD(3)
  real*8  Trans_GD(3,3)
  real*8  X_G(3),X_D(3)
  real*8  theta,phi,gamma

  theta=Euler_angle_GD(1)
  phi=Euler_angle_GD(2)
  gamma=Euler_angle_GD(3)

  !轉換矩陣(G->D)
  Trans_GD(1,1)=cos(phi)*cos(theta)
  Trans_GD(1,2)=cos(theta)*sin(phi)
  Trans_GD(1,3)=-sin(theta)
  Trans_GD(2,1)=-cos(gamma)*sin(phi)+sin(gamma)*sin(theta)*cos(phi)
  Trans_GD(2,2)=cos(gamma)*cos(phi)+sin(gamma)*sin(theta)*sin(phi)
  Trans_GD(2,3)=sin(gamma)*cos(theta)
  Trans_GD(3,1)=sin(gamma)*sin(phi)+cos(gamma)*sin(theta)*cos(phi)
  Trans_GD(3,2)=-sin(gamma)*cos(phi)+cos(gamma)*sin(theta)*sin(phi)
  Trans_GD(3,3)=cos(gamma)*cos(theta)

  do i=1,3
    X_D(i)=0.0
    do j=1,3
      X_D(i)=X_D(i)+Trans_GD(i,j)*X_G(j)
    enddo
  enddo

  !check
  if(.false.)then
    write(*,"(3a16)")"theta","phi","gamma"
    write(*,"(3f16.2)")theta,phi,gamma
    write(*,"(a16)")"Trans Matrix(GD)"
    write(*,"(3f16.2)")((Trans_GD(i,j),j=1,3),i=1,3)
  endif
  return
end
!----------------------------------------------------------------------------!
!  副程式:座標轉換(彈體座標系D->地面座標系G)                                 !
!  輸入:Euler_angle_GD                                                       !
!       X_D                                                                  !
!  輸出:X_G                                                                  !
!  定義:Euler_angle_GD(1)=theta=俯仰角(rad)                                  !
!       Euler_angle_GD(2)=phi=偏航角(rad)                                    !
!       Euler_angle_GD(3)=gamma=傾斜角(rad)                                  !
!       Trans_DG(3,3)=彈體座標系D->地面座標系G之轉換矩陣                     !
!       X_G(1)=變數在地面座標系G之X軸方向投影量                              !
!       X_G(2)=變數在地面座標系G之Y軸方向投影量                              !
!       X_G(3)=變數在地面座標系G之Z軸方向投影量                              !
!       X_D(1)=變數在彈體座標系D之X軸方向投影量                              !
!       X_D(2)=變數在彈體座標系D之Y軸方向投影量                              !
!       X_D(3)=變數在彈體座標系D之Z軸方向投影量                              !
!  備註:大地座標系OXgYgZg固定於地球上，座標系原點O為發射點                   !
!       OXg軸朝北為正，OYg軸朝東為正，OZg軸朝下為正。                        !
!  備註:彈體座標系OXYZ固定於彈體，座標系原點O為彈體重心                      !
!       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。         !
!----------------------------------------------------------------------------!
subroutine sub_Coordinate_Trans_DG(Euler_angle_GD,X_G,X_D)
  implicit none
  integer i,j
  real*8  Euler_angle_GD(3)
  real*8  Trans_DG(3,3)
  real*8  X_G(3),X_D(3)
  real*8  theta,phi,gamma

  theta=Euler_angle_GD(1)
  phi=Euler_angle_GD(2)
  gamma=Euler_angle_GD(3)

  !轉換矩陣(D->G)
  Trans_DG(1,1)=cos(phi)*cos(theta)
  Trans_DG(2,1)=cos(theta)*sin(phi)
  Trans_DG(3,1)=-sin(theta)
  Trans_DG(1,2)=-cos(gamma)*sin(phi)+sin(gamma)*sin(theta)*cos(phi)
  Trans_DG(2,2)=cos(gamma)*cos(phi)+sin(gamma)*sin(theta)*sin(phi)
  Trans_DG(3,2)=sin(gamma)*cos(theta)
```

```fortran
    Trans_DG(1,3)=sin(gamma)*sin(phi)+cos(gamma)*sin(theta)*cos(phi)
    Trans_DG(2,3)=-sin(gamma)*cos(phi)+cos(gamma)*sin(theta)*sin(phi)
    Trans_DG(3,3)=cos(gamma)*cos(theta)

    do i=1,3
      X_G(i)=0.0
      do j=1,3
        X_G(i)=X_G(i)+Trans_DG(i,j)*X_D(j)
      enddo
    enddo

    !check
    if(.false.)then
      write(*,"(3a16)")"theta","phi","gamma"
      write(*,"(3f16.2)")theta,phi,gamma
      write(*,"(a16)")"Trans Matrix(DG)"
      write(*,"(3f16.2)")((Trans_DG(i,j),j=1,3),i=1,3)
    endif
    return
  end
  !---------------------------------------------------------------------------!
  !  副程式:1維內插                                                           !
  !  輸入:A                                                                   !
  !       A1                                                                  !
  !       A2                                                                  !
  !       B1                                                                  !
  !       B2                                                                  !
  !  輸出:B                                                                   !
  !  定義:                                                                    !
  !---------------------------------------------------------------------------!
  subroutine sub_interpolation_1D(A,A1,A2,B,B1,B2)
    implicit none
    real*8  A,A1,A2
    real*8  B,B1,B2
    B = B1 + (B2-B1)*( (A-A1)/(A2-A1) )
    return
  end
  !---------------------------------------------------------------------------!
  !  副程式:2維內插                                                           !
  !  輸入:A                                                                   !
  !       A1                                                                  !
  !       A2                                                                  !
  !       B                                                                   !
  !       B1                                                                  !
  !       B2                                                                  !
  !       C1                                                                  !
  !       C2                                                                  !
  !       AF                                                                  !
  !       AF1                                                                 !
  !       AF2                                                                 !
  !       Ma                                                                  !
  !       Ma1                                                                 !
  !       Ma2                                                                 !
  !  輸出:C                                                                   !
  !  定義:                                                                    !
  !---------------------------------------------------------------------------!
  subroutine sub_interpolation_2D(A,A1,A2,B,B1,B2,C,C1,C2,AF1,AF2,Ma1,Ma2)
    use solution
    implicit none
    real*8  A,A1,A2
    real*8  B,B1,B2
    real*8  C,C1,C2
    real*8  AF1,AF2
    real*8  Ma1,Ma2

    A = A1 + (A2-A1)*( (AF-AF1)/(AF2-AF1) )
    B = B1 + (B2-B1)*( (AF-AF1)/(AF2-AF1) )

    C1=A
    C2=B
    C = C1 + (C2-C1)*( (Ma-Ma1)/(Ma2-Ma1) )
    return
  end
```

```fortran

!-----------------------------------------------------------------------------!
!  副程式:反矩陣                                                              !
!  輸入:A                                                                     !
!  輸出:inv_A                                                                 !
!  定義:                                                                      !
!-----------------------------------------------------------------------------!
subroutine inverse_matrix(A,inv_A)
  implicit none
  integer i,j
  !real*8  A(3,3),inv_A(3,3),I_matrix(3,3),save(3,3)
  real*8  A(3,3),inv_A(3,3),save(3,3)
  real*8  f
!  data I_matrix / 1,0,0,0,1,0,0,0,1 /
  real*8 I_matrix(3,3)

  do i=1,3
    do j=1,3
      if(i==j)then
        I_matrix(i,j)=1.0
      else
        I_matrix(i,j)=0.0
      endif
    end do
  end do
  !先清空矩陣
  inv_A=0.0

  if(.false.)then
    write(*,*)  "A="
    write(*,200)((A(i,j),j=1,3),i=1,3)
    write(*,*)  "inv_A="
    write(*,200)((I_matrix(i,j),j=1,3),i=1,3)
        format(3f12.4)
  endif
  !先將A存入save中
  save=A

  !計算反矩陣
  !1
  if(A(1,1) /= 1)then
    f=1.0/A(1,1)
    do j=1,3
      A(1,j)=f*A(1,j)
      I_matrix(1,j)=f*I_matrix(1,j)
    enddo
  endif
  !2
  if(A(2,1) /= 0)then
    f=-A(2,1)/A(1,1)
    do j=1,3
      A(2,j)=A(2,j)+f*A(1,j)
      I_matrix(2,j)=I_matrix(2,j)+f*I_matrix(1,j)
    enddo
  endif
  !3
  if(A(3,1) /= 0)then
    f=-A(3,1)/A(1,1)
    do j=1,3
      A(3,j)=A(3,j)+f*A(1,j)
      I_matrix(3,j)=I_matrix(3,j)+f*I_matrix(1,j)
    enddo
  endif
  !4
  if(A(2,2) /= 1)then
    f=1.0/A(2,2)
    do j=1,3
      A(2,j)=f*A(2,j)
      I_matrix(2,j)=f*I_matrix(2,j)
    enddo
  endif
  !5
```

```fortran
    if(A(3,2) /= 0)then
      f=-A(3,2)/A(2,2)
      do j=1,3
        A(3,j)=A(3,j)+f*A(2,j)
        I_matrix(3,j)=I_matrix(3,j)+f*I_matrix(2,j)
      enddo
    endif
    !6
    if(A(3,3) /= 1)then
      f=1.0/A(3,3)
      do j=1,3
        A(3,j)=f*A(3,j)
        I_matrix(3,j)=f*I_matrix(3,j)
      enddo
    endif
    !7
    if(A(2,3) /= 0)then
      f=-A(2,3)/A(3,3)
      do j=1,3
        A(2,j)=A(2,j)+f*A(3,j)
        I_matrix(2,j)=I_matrix(2,j)+f*I_matrix(3,j)
      enddo
    endif
    !8
    if(A(1,3) /= 0)then
      f=-A(1,3)/A(3,3)
      do j=1,3
        A(1,j)=A(1,j)+f*A(3,j)
        I_matrix(1,j)=I_matrix(1,j)+f*I_matrix(3,j)
      enddo
    endif
    !9
    if(A(1,2) /= 0)then
      f=-A(1,2)/A(2,2)
      do j=1,3
        A(1,j)=A(1,j)+f*A(2,j)
        I_matrix(1,j)=I_matrix(1,j)+f*I_matrix(2,j)
      enddo
    endif
    inv_A=I_matrix
    do i=1,3
      do j=1,3
        if(abs(inv_A(i,j))>1e10)then
          write(*,*)"inverse matrix dosen't exit!"
          stop
        endif
      enddo
    enddo

    !將save取回重新放入A中
    A=save

    return
  end
  !--------------------------------------------------------------------------!
  !  副程式:輸出計算結果                                                     !
  !  輸入:                                                                   !
  !  輸出:                                                                   !
  !  定義:                                                                   !
  !--------------------------------------------------------------------------!
  subroutine sub_output(timepoint)
    use global
    use constant
    use module_atmosphere
    use solution
    use unit
    implicit none
    integer i,j
    real*8  timepoint(5,5)
    real*8  R,H

    !R:水平距離(km)
    R=sqrt(Y(4)**2+Y(5)**2)/1000.
```

```
      !H:高度(km)
      H=-Y(6)/1000.
      !i=1:離軌,i=2:最大速度,i=3:燃畢,i=4:最高點,i=5:落海
      !j=1:時間,j=2:高度,j=3:射程,j=4:速度,j=5:馬赫數

      !i=2:最大速度
      i=2
      if(V>timepoint(2,4))then
        timepoint(i,1)=t
        timepoint(i,2)=-Y(6)/1000.
        timepoint(i,3)=sqrt(Y(4)**2+Y(5)**2)/1000.
        timepoint(i,4)=V
        timepoint(i,5)=Ma
      endif

      !i=3:燃畢
      i=3
      if(i_main==counter)then
        timepoint(i,1)=t
        timepoint(i,2)=-Y(6)/1000.
        timepoint(i,3)=sqrt(Y(4)**2+Y(5)**2)/1000.
        timepoint(i,4)=V
        timepoint(i,5)=Ma
      endif

      !i=4:最高點
      i=4
      if(-Y(6)/1000.>timepoint(4,2))then
        timepoint(i,1)=t
        timepoint(i,2)=-Y(6)/1000.
        timepoint(i,3)=sqrt(Y(4)**2+Y(5)**2)/1000.
        timepoint(i,4)=V
        timepoint(i,5)=Ma
      endif

      !i=5:落海
      i=5
      if(-Y(6)<10.)then
        timepoint(i,1)=t
        timepoint(i,2)=-Y(6)/1000.
        timepoint(i,3)=sqrt(Y(4)**2+Y(5)**2)/1000.
        timepoint(i,4)=V
        timepoint(i,5)=Ma
      endif

      !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

      !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
      if(i_main==1)then
        open(unit=unit_output_6D_1,file="output_6D_1.txt")
          write(unit_output_6D_1,100)"t(s)","Ax(m/s^2)","Ay(m/s^2)","Az(m/s^2)",&
            &"Vx(m/s)","Vy(m/s)","Vz(m/s)","X(m)","Y(m)","Z(m)"
        open(unit=unit_output_6D_2,file="output_6D_2.txt")
          write(unit_output_6D_2,100)"t(s)","P(deg)","Q(deg)","R(deg)",&
            &"theta(deg)","phi(deg)","gamma(deg)","dP/dt(deg/s)",&
            &"dQ/dt(deg/s)","dR/dt(deg/s)"
        open(unit=unit_output_6D_3,file="output_6D_3.txt")
          write(unit_output_6D_3,110)"t(s)","AF(deg)","apha(deg)","phi_c(deg)",&
            &"V(m/s)","Cs(m/s)","Ma","mass(kg)"
        open(unit=unit_output_6D_4,file="output_6D_4.txt")
          write(unit_output_6D_4,130)"t(s)","Range(km)","H(km)","AF(deg)",&
            &"V(m/s)","Ma","mass(kg)"
      endif
      write(unit_output_6D_1,200)t,(dY_dt(i),i=1,3),(Y(i),i=1,6)
      write(unit_output_6D_2,200)t,(Y(i)*180./PI,i=7,12),(dY_dt(i),i=7,9)
      write(unit_output_6D_3,210)t,AF,apha*PI/180.,phi_c*PI/180,V,Cs,Ma,Y(13)
      write(unit_output_6D_4,230)t,R,H,AF,V,Ma,Y(13)

          format(a6,9a14)
          format(f6.2,9f14.2)
          format(a6,7a14)
          format(f6.2,7f14.2)
          format(a6,6a14)
          format(f6.2,6f14.2)

      if(-Y(6)/1000.<0.1)then
        open(unit=unit_output_6D_5,file="output_6D_5.txt")
          write(unit_output_6D_5,120)"key point","t(s)","H(km)","Range(km)","V(m/s)","Ma"
          write(unit_output_6D_5,220)"leave luncher",(timepoint(1,j),j=1,5)
          write(unit_output_6D_5,220)"Max. V",(timepoint(2,j),j=1,5)
          write(unit_output_6D_5,220)"motor burn out",(timepoint(3,j),j=1,5)
          write(unit_output_6D_5,220)"Max. H",(timepoint(4,j),j=1,5)
          write(unit_output_6D_5,220)"fall into sea",(timepoint(5,j),j=1,5)
        close(unit_output_6D_5)
      endif
          format(6a14)
          format(a14,5f14.2)

      return
    end
```