NEW TOOL FOR FINDING PERIODIC HALO ORBITS: THE SOLVER OF A SPACECRAFT SIMULATOR (ESPSS -ECOSIMPRO® EUROPEAN SPACE PROPULSION SYSTEM SIMULATION)

Christophe R. Koppel

KopooS Consulting Ind., 57 rue d'Amsterdam-75008 Paris

ABSTRACT

Halo orbits and other periodic orbits in the restricted circular 3 body problem has always been very well explained in the literature since their discovery by Farquhar in the 60's of last century. However for finding the numerical values of such orbits, the availability of the tools dedicated for such tasks is not obvious. Due to the fact that the differential equations are quite simple, these days most of the time tools used are based on some computer listings written within an US based mathematical framework, which is clearly dedicated for people highly involved in computer and computer language rather than for general purpose Engineers.

Hence the approach chosen in the paper is to rely on a tool largely used by Engineers (and not computer guys) for taking advantage of the capabilities of solving dynamic problems: the tool used is a European tool which is an object oriented solver of differential equations which is the cornerstone of the Spacecraft Simulator (ESPSS - EcosimPro® European Space Propulsion System Simulation) largely used by engineers.

The paper presents the mathematical problem in simple words and the method used to solve it.

The major advantages of the approach proposed and used successfully is to benefit of a real simulation framework based on models and on experiments where there are no mixing between the inputs\outputs needs and the real problem being to be solved. Hence the full model can be clearly and explicitly described while the results coming from the experiments can be extensively assessed and analysed with simple monitor outputs.

Index Terms— Solver, Halo, EcosimPro, ESPSS

1. INTRODUCTION

The use of an engineering simulation tool not dedicated to computer people neither to mathematicians or numerical analysts can be considered as a great improvement of the current engineering practices. The paper describes the integration of the equations into EcosimPro® which is a Physical Simulation Modelling tool that is an object-oriented tool dedicated for system analysis. That is a visual simulation tool capable of solving various kinds of dynamic systems represented by writing equations and discrete events. It can be used to study both transients and steady states. The object oriented tool, with the propulsion libraries *ESPSS* (European Space Propulsion System Simulation) from ESA for example, allows the user to draw (and to design at the same time) the propulsion system with components of that specific library with tanks, lines, orifices, thrusters, tees. The user enhances the design with components from the thermal library (heaters, thermal conductance, radiators), from the control library (analogue/digital devices), from the electrical library, etc.

Because this tool, developed for ESA is available freely for every of its engineers, and available too at all prime satellite corporations, and in many other companies, it has been found interesting to assess its use for deeper mathematical problems like finding halo orbits around Earth-Moon L2 point as it is needed to know for some preliminary projects.

Most of the well known problems take as starting point the circular restricted three body problem, which models the motion of a massless particle under the gravitational attraction of two punctual primaries revolving in circular orbits around their centre of mass. In the rotating frame, the two bodies remain fixed, and the point L2 is also fixed (as well as 4 other well known points). It is rather easy to find the characteristic point L2, but orbits around such point must be solved with a so high accuracy that the problem is more to find initial condition with the same accuracy. This problem is thus presented below, first with the equations which are not very complex, then the method used within EcosimPro to iterate to a good enough solution.

2. THE DIFFERENTIAL SYSTEM

First, one presents directly the differential system needed to be solved for finding any trajectory (without thrust) in the circular restricted three body problem (CR3BP), one has:

$$\dot{X} = f(X) \tag{1}$$

where the vector X is a 6 dimensions vector and the vector function f contains equations according to the following one column matrixes.

$$X=\begin{bmatrix}x\\y\\z\\\dot{x}\\\dot{y}\\\dot{z}\end{bmatrix}\quad f(X)=\begin{bmatrix}\dot{x}\\\dot{y}\\\dot{z}\\x+2\dot{y}-\frac{(1-\mu)(x+\mu)}{r_1^3}-\mu\frac{(x-(1-\mu))}{r_2^3}\\y-2\dot{x}-\frac{(1-\mu)y}{r_1^3}\quad-\mu\frac{y}{r_2^3}\\-\frac{(1-\mu)z}{r_1^3}\quad-\mu\frac{z}{r_2^3}\end{bmatrix}$$

with

$$r_1=\begin{bmatrix}x-(-\mu)\\y\\z\end{bmatrix}\qquad \|r_1\|=\sqrt{(x-(-\mu))^2+y^2+z^2}$$

$$r_2=\begin{bmatrix}x-(1-\mu)\\y\\z\end{bmatrix}\qquad \|r_2\|=\sqrt{(x-(1-\mu))^2+y^2+z^2}$$

This system of differential equations (1) comes from the fact that one considers the non dimensional rotating two main body system centered at the system center of mass.

First, the relative position of the third body $\vec{r}$ having coordinates x,y,z is sketched in Figure 1.

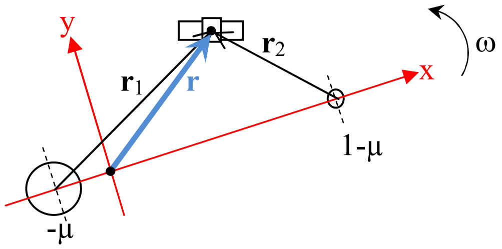

Figure 1: Non dimensional rotating frame considered

A very important simplification comes from the fact that the main body orbit is supposed here circular, so the angular rotation ω of the rotating system wrt inertial frame is constant. A standard convention for such problem in non inertial frame, is used: the total mass of the system is a unit mass ($m_1 + m_2 = 1$) and the constant separation between main body is a unit length. The normalized mass of both main body then follows as $m_1 = 1 - \mu$ and $m_2 = \mu$. The time unit is 2π for one period of the system, leading to ω=1 as for the gravitation constant G=1.Velocity derivative equation (i.e. acceleration) of the third body is thus quite easy to derive because in the rotating system, the well known expression of the relative acceleration $\ddot{\vec{r}}$ for a constant $\vec{\omega}$ under the two main body gravitational attractions stands as:

$$\ddot{\vec{r}}+\vec{\omega}\times(\vec{\omega}\times\vec{r})+2\vec{\omega}\times\dot{\vec{r}}=-\frac{(1-\mu)}{r_1^3}\vec{r}_1-\frac{\mu}{r_2^3}\vec{r}_2$$

With simply $\vec{\omega}$ =1 along the z axis, the cross products are obvious, giving straightforward the system (1).

3. SOME PERIODIC ORBITS

For some periodic orbit it is sufficient to find initial conditions of X (position and velocity) such that after half an orbit some values remain null as initially:

$$X(0)=\begin{bmatrix}x_0\\0\\z_0\\0\\\dot{y}_0\\0\end{bmatrix}\qquad X(T½)=\begin{bmatrix}x\\0\\z\\0\\\dot{y}\\0\end{bmatrix}$$

In clear, the starting point is in the x,z plane because y_0=0 with no velocity on x, z but with velocity on y: $\dot{y}_0 \neq 0$.

One notes that in the final condition there are 3 zeros.

4. SOLUTIONS OF PERIODIC ORBITS

To find a periodic orbit, one supposes that a first starting point not too far from the solution is given.

In order to reduce the number of guesses to do, one can fix the initial value x_0 (and further make a loop on it) .The new problem of the 3 zeros is for a given x_0, to find Y such that:

$$g(Y)=0 \quad\text{with}\quad Y=\begin{bmatrix}z_0\\\dot{y}_0\\T_{1/2}\end{bmatrix}\quad\text{and}\quad g(Y)=\begin{bmatrix}y(t)\\\dot{x}(t)\\\dot{z}(t)\end{bmatrix}_{t=T_{1/2}}$$

where the value $T_{½}$ is the time at which y is for the first time null after the integration 0 to t; i.e. the trajectory is back to the x, z plane whatever the values of velocities along x or z. Those two last values have to be nullified: iterations by Newton method can be used for finding right guesses for nullified all the lines of $g(Y)$:

$$Y_{n+1}=Y_n-\left[\frac{\partial g}{\partial Y}_{|Y=Y_n}\right]^{-1}.g(Y_n)$$

Note : This is like for a 1-D curve: finding a new guess u_1 after u_0 of a function $h(u)$=0, one follows the tangent of the curve $h(u)$ at point $u=u_0$ up crossing the line "0", $\frac{h(u_0)-0}{u_0-u_1}=\frac{dh}{du}_{|u=u_0}$.

Hence $u_0-u_1=\left[\frac{dh}{du}_{|u=u_0}\right]^{-1}.(h(u_0)-0)$ giving

$u_1=u_0-\left[\frac{dh}{du}_{|u=u_0}\right]^{-1}.h(u_0)$ in one dimension.

Here the tangent (or Jacobian differential) is a bit delicate because the definition of g is including integration *0* to $T_{½}$.

The derivative matrix $\frac{\partial g}{\partial Y}_{|Y=Y_n}$ is:

- for the sub set of variable *Y*: $\begin{bmatrix} z_0 \\ \dot{y}_0 \end{bmatrix}$, $\frac{\partial g}{\partial Y}_{|Y=Y_n}$ comprises integration of a subset of the derivative of the general function *f* in (1) $\dot{X} = f(X)$ for which one has the STM (state transition matrix): $M(t,t_0) = \frac{\partial X_{|t=t}}{\partial X_{|t=t_0}}$ defined by the differential equation with initial value (2):

$$\dot{M}(t,t_0) = \frac{d}{dt}\left[\frac{\partial X_{|t=t}}{\partial X_{|t=t_0}}\right] \quad M(t_0,t_0) = \frac{\partial X_{|t=t_0}}{\partial X_{|t=t_0}} = [Identity] \quad (2)$$

  That is an impressive system of 36 differential equations to be integrated simultaneously from *t*=0 to *t*, but some of the equations are trivial...

  Note: because the time is not explicitly appearing in the equations *f(X)* as the system is autonomous, one has:

$$\dot{M}(t,t_0) = \frac{d}{dt}\left[\frac{\partial X_{|t=t}}{\partial X_{|t=t_0}}\right] = \left[\frac{\partial \dot{X}_{|t=t}}{\partial X_{|t=t_0}}\right] = \left[\frac{\partial \dot{X}_{|t=t}}{\partial X_{|t=t}}\right] \cdot \left[\frac{\partial X_{|t=t}}{\partial X_{|t=t_0}}\right] = \left[\frac{\partial f}{\partial X}\right] \cdot M(t,t_0)$$

  Hence, it is obvious that (2) can better rely on the following differential equation (3) involving f :

$$\dot{M}(t,t_0) = \left[\frac{\partial f}{\partial X}\right] \cdot M(t,t_0). \quad (3)$$

- for the sub set of variable *Y*: $\left[T_{½} \right]$, $\frac{\partial g}{\partial Y}_{|Y=Y_n}$ is given by $\frac{dg}{dt}_{|Y=Y_n} = \dot{g}_{|Y=Y_n}$, this is directly a sub set of the *f* function itself in (1) $\dot{X} = f(X)$ which is to be integrated as well (6 additional differential equations).

A. PRACTICAL APPROACH

By numbering the variables of the first problem (1) with index 1 to 6 and numbering the variable *t* to index 7, one gets straight forward successively that *Y* involves variables index 3 5 7 and g(Y) involves variables index 2 4 6. Hence the derivative matrix can be written symbolically in short as in equation (3):

$$\frac{dg}{dY}_{|Y=Y_n} = [Col.3\,5\ of\ rows\ 2\,4\,6\ of\ M(t,t_0)]\ [rows\ 2\,4\,6\ of\ f(X)] \quad (3)$$

B. FINAL RESOLUTION

Finally the solution of periodic orbits is performed by the integration of a system of 36 + 6= 42 differential equations and that within a second sub-loop for finding the solution $g(Y) = 0$ with Newton, which leads to periodic orbit.

A primary loop on the fixed variable allows plotting many halo orbits.

With some tests, it was better to guess x_0 while keeping fixed z_0 so in the equations above it is just matter of replacing the index 3 by index 1.

Note: The model and experiment is a standalone model within EcosimPro "as-is" without real need of sophisticated libraries like ESPSS, even if for further applications this library would be highly recommended. In addition to the equations in EL (EcosimPro langage) shown below, a simple function so called "ODE113" has been implemented for the integration of the 6 and 42 differential equations (based on Runge-Kutta with possibility of error control and variable time steps) and also a matrix inversion routine with error quantification has been added. Such features could be as well added by the Ecosimpro team to EcosimPro "as-is"!

5. APPLICATION TO HALO ORBITS LAGRANGE POINT L2 OF THE SYSTEM EARTH+MOON

The system of equations (1) representing the CRTBP is used for the Earth+Moon system with μ = 0.0121506038.

For the following set of initial values:

x_0= 1.12
z_0=0.01 (fixed for each Halo)
$\dot{y}_0$ = 0.17
$T_{½}$= 1.7

we get the following Figure 2 of simulation plots of 20 Halo orbits with run time around of half second for each Halo orbit. After each halo the fixed value z_0 is increased by 1% each time.

Of course, the above plot is very useful for analysts, but for a first view showing the complexity of the periodic solution, using a 3D visualisation tool fed by the data from EcosimPro we can get a cubic view with projections of the orbits on the three reference planes as in Figure 3.

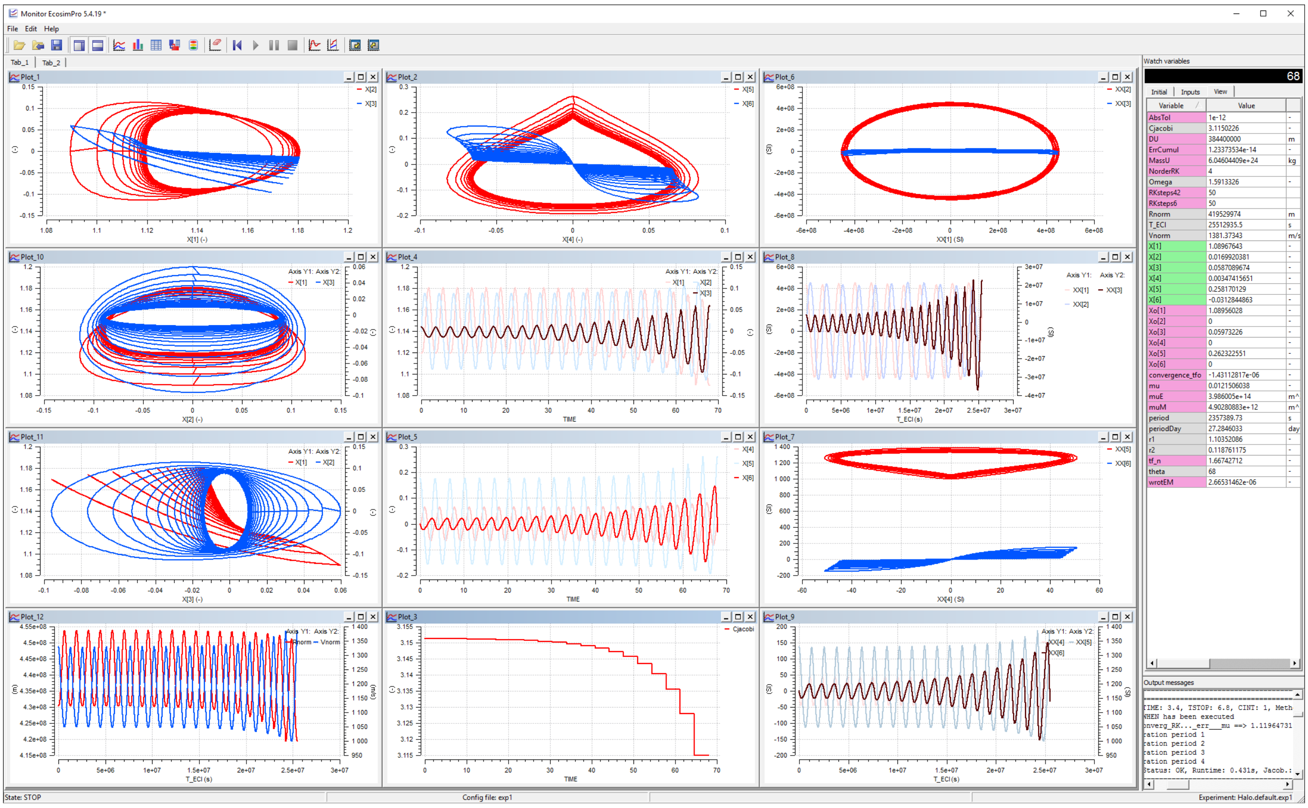

Figure 2: simulation plots of 20 Halo orbits around Earth+Moon L2

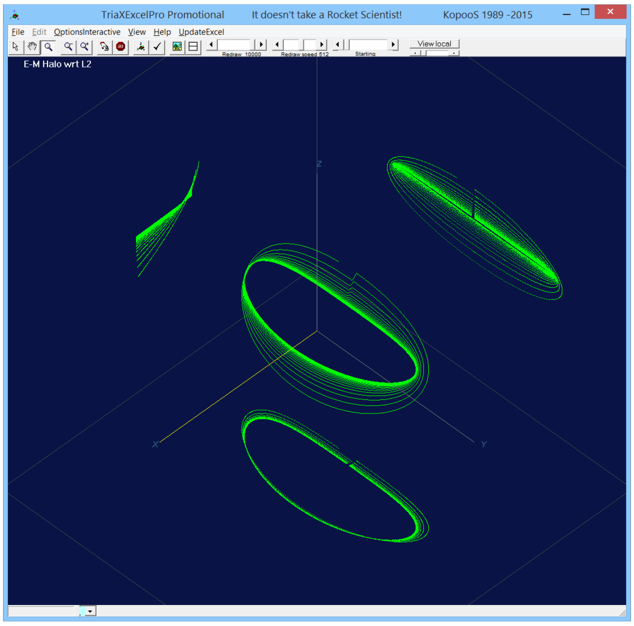

Figure 3: 3D visualisation of 20 Halo orbits around Earth+Moon L2 (3D cubic view)

6. CONCLUSIONS

The paper has presented in simple words the mathematical problem of finding some Halo orbits and the method implemented to solve it within the EcosimPro environment.
The major advantages of the approach and used successfully is to benefit of a real simulation framework based on models and on experiments where there are no mixing between the inputs\outputs needs and the real problem being to be solved.
Hence the full model can be clearly and explicitly described while the results coming from the experiments can be extensively assessed and analysed with simple EcosimPro monitor outputs.

7. REFERENCES

[1] Robert W. Farquhar "The Utilization of Halo Orbit in Advanced Lunar Operation," NASA-TN-D-6365, JULY 1971

[2] J.D. Mireles James, "Celestial Mechanics Notes Set 1: Introduction to the N-Body Problem," January 3, 2007

[3] Franco Bernelli Zazzera, Francesco Topputo, Mauro Massari, "Assessment of Mission Design Including Utilization of Libration Points and Weak Stability Boundaries," for Advanced Concepts Team (ESTEC), 2005

[4] JeffreyS. Parker, "Developing a Mission Design Architecture for the EarthMoon Three-Body System," thesis December 3, 2004

8. ANNEX: TRACEABILITY

For readers interested to run similar cases, the full listing of the experiment and the model in the EcosimPro environment is given below.

A. LISTING OF THE EXPERIMENT

```
/*-----------------------------------------------------------------------------------
LIBRARY: MY_SAT
COMPONENT: Halo
PARTITION: default
EXPERIMENT: exp1
TEMPLATE: TRANSIENT
-----------------------------------------------------------------------------------*/
EXPERIMENT exp1 ON Halo.default
DECLS
  REAL T_Halo
  STRING Filnam="Rep"
  INTEGER nbHalo=1
OBJECTS
INIT
BOUNDS
  MY_SAT.AbsTolM12 = 1e-012
  MY_SAT.NbSteps2000 = 50
  MY_SAT.NorderRK85 =4 -- 5
BODY
  GuessZ3notX1_o=1
  Xo1= 1.12 -- x_o
  Xo3=0.01 -- zo FIXED NOW
  Xo5= 0.17--ydot_o
  Thalfperiod_o=1.7-- t
  NloopNewtonHalo=15
  T_Halo=2*Thalfperiod_o
  nbHalo=20
  Filnam="Halo20.rpt"
  -- creates an ASCII file with the results in table format
  REPORT_TABLE(Filnam, " *X[*] *XX[*] *PHI* Cj* A* G* Halo* *12 *00 *85 *U *RK Om*
R* T_* Teco* V* conv* mu* per* r* wrot*] dE* L* ")
  DEBUG_LEVEL= 1
  IMETHOD= DASSL
  setStopWhenBadOperation(FALSE)
  REL_ERROR = MY_SAT.AbsTolM12
  ABS_ERROR = REL_ERROR
  TOLERANCE =REL_ERROR   REPORT_MODE=IS_STEP
  TIME = 0
  FOR (i IN 1, nbHalo)
    FlagSearchPeriodicOrbit=TRUE
    INTEG_TO(TIME+T_Halo,1)
    -- Case of series of Halo orbits (evolution of z)
    IF i!=nbHalo THEN --change but not for the last one to keep all results of the last case
      Xo[3]=Xo[3]+i*Xo[3]*0.01
    END IF
  END FOR
END EXPERIMENT
```

B. LISTING OF THE MODEL

```
COMPONENT Halo
DATA
    REAL Xo1=0.99197555537727 UNITS "DU" "xo"
    REAL Xo3=-0.00191718187218 UNITS "DU" "zo"
    REAL Xo5=-0.01102950210737 UNITS "DU/TU" "vyo"
    REAL Thalfperiod_o=1.52776735363559 UNITS "TU" "half period for
periodic orbit, initial guess"
    INTEGER NloopNewtonHalo=0 UNITS "-" " 0 --no convergence-- else up to 14 is
enough for convergennce"
    INTEGER GuessZ3notX1_o=3 UNITS "-" " flag=3 for xo fixed and zo guess
==>find a Lyapunov plan; flag=1 for zo fixed and xo guess ==>find Halo from a Lyapunov plan
with some small zo "
    --REAL RunCode=2 UNITS "-" "code=0: J.D. Mireles James 1 Nick Truesdale 2: Earth Moon
L2, 10: J.D. Mireles James L2 from Lyapunov, etc..."
DECLS
    BOOLEAN FlagSearchPeriodicOrbit=TRUE --directive for new search of
periodic orbits
    CONST INTEGER LDIM=6
    INTEGER NorderRK,NbSteps, RKsteps42,RKsteps6,
GuessZ3notX1,Function_ODE_IVP --info
    INTEGER i462[3]={4,6,2}
    INTEGER i357[3]={3,5,7}
    REAL X[LDIM] UNITS "-" --position then velocity in barycentric rotating frame addim
    REAL theta UNITS "-"
    REAL T_ECI,period UNITS "s"
    REAL periodDay UNITS "day"
    REAL r1,r2,Omega,Cjacobi UNITS "-"
    EXPL REAL wrotEM3D[3] , wrotEMCrossXXrot[3] UNITS "-" --dim
    EXPL REAL XX[6], XXrot[3] UNITS "SI" --dim
    EXPL REAL Rnorm UNITS "m"
    EXPL REAL Vnorm UNITS "m/s"
    DISCR REAL Xf_n[LDIM] UNITS "-" --point then velocity in barycentric rotating frame
addim
    DISCR REAL dX6_dt[LDIM] UNITS "-" --velocity then acceleration in barycentric
rotating frame addim
    DISCR REAL Xo_n[7+10], Xo[7] UNITS "-" -- 6+added more rowse for compact
information data
    DISCR REAL PHI[6,7] UNITS "-"
    DISCR REAL DF[3,3],D[3,3],XSo[3], XSo_star[3] ,Xff[3],ErrCumul
UNITS "-"
    DISCR REAL muE,muS,muM UNITS "m^3/s^2"
    DISCR REAL dEM,AU,DU UNITS "m"
    DISCR REAL MassU UNITS "kg"
    DISCR REAL wrotEM UNITS "-"
    DISCR REAL mu UNITS "-"
    DISCR REAL G = 6.67384E-11 UNITS "m^3/(kg.s^2)"--+- 0.00080 m^3.kg^-1.s^-2
    DISCR REAL convergence_tfo UNITS "-"
    DISCR REAL to_n,tf_n,Thalfperiod UNITS "-"
    DISCR REAL AbsTol UNITS "-"
    DISCR REAL L1, L2, L3 UNITS "DU" --for info
```

```
INIT
    FOR (i IN 1,6)
      Xo[i] = 0
    END FOR
    GuessZ3notX1=GuessZ3notX1_o
    muE = 1*3.986005E14
    muS = 328902.82113001*3.986005E14--; % was Relative to earth
    muM = 0.0123000569113856 *3.986005E14
    mu=muM/(muE+muM)
    dEM=384400e3
    Xo[1]=Xo1 --GuessZ3notX1=3 --guess Z User to choose or default =3
    Xo[3]=Xo3
    Xo[5]=Xo5
    Thalfperiod=Thalfperiod_o
    DU=dEM
    MassU=(muE+muM)/G
    wrotEM=sqrt(G*MassU/DU**3)
    --for info here only because mu in known and allow computation of L1 L2 L2
    L1=findLagrangePoints(0.83, mu)-- init value not too far from the wanted roots
    L2=findLagrangePoints(1.15 , mu)
    L3=findLagrangePoints(-1.0, mu)
    PRINT (" for_information:_L1,L2,L3_in DistanceUnitsEarthMoon= $L1 $L2 $L3 ")
    --Eco Normal Init of the derivatives
    FOR (i IN 1,6)
      X[i]=Xo[i]
    END FOR
    Xo[7]= Thalfperiod --variable added
    i357[1]=GuessZ3notX1
DISCRETE
    WHEN FlagSearchPeriodicOrbit THEN -- this is like a program to be run
  before starting integratons by EcosimPro depending on the directive FlagSearchPeriodicOrbit .
      --Inputs : Xo[i] (including Xo[7]= Thalfperiod), NloopNewtonHalo , mu OUT: X[i] initialized by
  Xo which is set to the last converged Xo_n[i] (for a good starting guess for other periodic orbits)
      --Iteration on the suited IVP fulfilling the goal (with xo fixed (index 1) )
      -- goal: after a half_period vx,vz and y shall be all null (index 4,6,2) with free variables to
  guess: initial values of zo, vyo, half_period (index 3,5 and variable tf_n)
      FlagSearchPeriodicOrbit=FALSE --clear the condition for running this routine
      to_n=0 --never modified here
      FOR (i IN 1,7)
        Xo_n[i]=Xo[i] --here we work with IVP Xo_n (including Thalfperiod) because Xo is
  never modified inside the next loop
      END FOR
      --@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
      FOR (k IN 1,NloopNewtonHalo)
        --call ODE integration for the final state Xf_n from the given IVP Xo_n to see how good
  are the guesses and process the iterations
        AbsTol=AbsTolM12--1E-12
        NbSteps=NbSteps2000
        NorderRK=NorderRK85
        Function_ODE_IVP=LDIM
        tf_n=Xo_n[7] -- tf is a condition final for the ODE but it is as Thalfperiod an initial
  condition for the process of finding a periodic solution by convergence Newton
        ODE113 (LDIM, to_n, tf_n, Xo_n, Xf_n, NorderRK, AbsTol,
  NbSteps, mu, Function_ODE_IVP, RKsteps6 )--out Xf_n
        --Zero search by Newton method iterations
        FOR (i IN 1,3)
          XSo[i]=Xo_n[i357[i]]
        END FOR
        FOR (i IN 1,3)--Array with the 3 components results of ODE integration to be
  nullified by converging the IVP XSo to XSo_star
          Xff[i]=Xf_n[i462[i]] -- i462[3]={4,6,2} i357[3]={3,5,7}
        END FOR
        --Jacobian at current final point tf_n=Xo_n[7] wrt IVP initial Xo_n given for to_n -- IT
  INCLUDES THE ODE113 SIZE 42
        STMatrixCR3BP ( to_n, tf_n , Xo_n, PHI, mu ,
  RKsteps42)-- out PHI = d FF / d xx = d xxdot_i / d xx_j
        --derivative of X6 wrt time at final point, needed for getting the time derivatives to fill the
  matrix DF (dFF/dxx)
        Function_ODE_IVP_6( 6, Xf_n, dX6_dt, mu )
        FOR (i IN 1,6)--extended PHI last column added with time derivatives d FF / d t = d
  xxdot_i / d t in column 7
          PHI[i,7] = dX6_dt[i]
        END FOR
        -- dFF/dxx Full derivative of XXf (to be nullified) wrt XXo (selected state variables and
  time) i462[3]={4,6,2} i357[3]={3,5,7}
        FOR (i IN 1,3)
          FOR (j IN 1,3)
            DF[i,j] =PHI[i462[i],i357[j]] -- i462[3]={4,6,2} i357[3]={3,5,7}
          END FOR
        END FOR
        InvMatrix( 3,DF, D , ErrCumul)
        --XSo_star The next solution guess : XSo_star = XSo-inv(dFF/dxx)*Xff
        FOR (i IN 1,3)--extended PHI with time derivatives
          XSo_star[i]=XSo[i]-SUM (m IN 1,3; D[i,m]*Xff[m])
        END FOR
        --New Xo_n = Xo_n+1 for iterations
        FOR (i IN 1,7)
          Xo_n[i]=Xo[i] --come back to the first init conditions before update of the selected
  ones
        END FOR
        FOR (i IN 1,3)
          Xo_n[i357[i]]=XSo_star[i]--update the selected ones with better guesses
        END FOR
        -- end for the new Xo_n, ready to go for iterations
        --PRINTa1 (3, XSo_star , "new guess")
        --convergence and for info
        convergence_tfo=XSo_star[3]-XSo[3]
        Xo_n[8]= convergence_tfo --for info only and printing
        Xo_n[9]= NorderRK --for info only and printing
        Xo_n[10]= RKsteps6 --for info only and printing
        Xo_n[11]= RKsteps42 --for info only and printing
        Xo_n[12]= ErrCumul --for info only and printing
        Xo_n[13]= mu --for info only and printing
      END FOR --k
      --@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
      PRINTa1 (13, Xo_n, "final_Xo_n--_tf_n_converg_RK..._err--_mu ")
      FOR (i IN 1,7)--Update Xo from last converged Xo_n, and also memorized for starting
  other periodic orbit search if any
        Xo[i]=Xo_n[i] --including the time tf_n
      END FOR
      --Update wrt Init: New init conditions for derivative variables for EcosimPro integration: the
  right one for a periodic orbit
      FOR (i IN 1,6) --only 6 for X
        X[i]=Xo[i]
      END FOR
    END WHEN
CONTINUOUS
      r1=((mu+X[1])**2+X[2]**2+X[3]**2)**(1/2)--distance point to body1
      r2=((mu+X[1]-1)**2+X[2]**2+X[3]**2)**(1/2)--distance point to body2
      EXPAND (i IN 1,3) X[i+3] = X[i]'
      --dynamic f=ma in barycentric rotating frame, see for example J.D. Mireles James and many
  others
      X[4]'=+X[1]+2*X[5]-(X[1]+mu)*(1-mu)/r1**3-(X[1]+mu-1)*mu/r2**3
      X[5]'=+X[2]-2*X[4]-X[2]*(1-mu)/r1**3-X[2]*mu/r2**3
      X[6]'=-X[3]*(1-mu)/r1**3-X[3]*mu/r2**3
      --for info
      Omega=0.5*(X[1]**2+X[2]**2)+(1-mu)/r1+mu/r2
      Cjacobi=2*Omega-(X[4]**2+X[5]**2+X[6]**2)
      --Geocentric results in ECI with vector XX
      T_ECI=TIME/wrotEM --TIME is addim = 6.28 for 1 period
      period=2*3.1415926535897932384626433832795 /wrotEM
      periodDay=period/86400
      EXPAND (i IN 1,2) wrotEM3D[i]=0 -- only 2 first coordinates
      wrotEM3D[3]=wrotEM -- the 3rd coordinate
      --cross product
      wrotEMCrossXXrot[3]=wrotEM3D[1]*XXrot[2]-wrotEM3D[2]*XXrot[1]
      wrotEMCrossXXrot[1]=wrotEM3D[2]*XXrot[3]-wrotEM3D[3]*XXrot[2]
      wrotEMCrossXXrot[2]=wrotEM3D[3]*XXrot[1]-wrotEM3D[1]*XXrot[3]
      EXPAND_BLOCK (i IN 1,3)
          XXrot[i] = X[i]*DU
          XX[i+3] = X[i+3]*DU*wrotEM+wrotEMCrossXXrot[i]
      END EXPAND_BLOCK
      theta=TIME --wrotEM*T_ECI
      XX[1] = XXrot[1]*cos(theta)-XXrot[2]*sin(theta)
      XX[2] = XXrot[1]*sin(theta)+XXrot[2]*cos(theta)
      XX[3] = XXrot[3]
      -- useful
      Rnorm=sqrt(SUM(i IN 1,3; XX[i]**2))
      Vnorm=sqrt(SUM(i IN 4,6; XX[i]**2))
END COMPONENT
```