

Métodos numéricos en astronomía dinámica: Trabajo Final

Due on Lunes, Julio 20, 2015

Fernando Roid 10:30am

Guido Granda Muñoz

Contents

| | |
|---|-----------|
| Problem 1 | 3 |
| Solución | 3 |
| Integrador de Euler | 4 |
| Integrador de Runge kutta | 4 |
| Integrador Simplicético de primer orden | 8 |
| Problem 2 | 12 |
| Solución | 12 |

Problem 1

Considere o Hamiltoniano de un péndulo simples:

$$H(\theta, p) = \frac{p^2}{2} - k\cos(\theta) \quad (1)$$

Siendo θ el ángulo, p el momentum, y k una constante. Para el siguiente conjunto de condiciones iniciales (θ_i, p_i) :

$$\theta_i = 0 \quad \forall i \quad p_i = \frac{i\sqrt{k}}{10} \quad i = 0, 1, 2, \dots, 30$$

Resolver numéricamente las ecuaciones de Hamilton del sistema utilizando:

1. Un integrador de Euler.
2. Un integrador simpléctico de primer orden, recordando que el Hamiltoniano puede ser separado de la forma: $H(\theta, p) = H_0(p) + H_1(\theta)$.
3. Un integrador Runge-Kutta de cuarto orden.

Escriba las ecuaciones de los integradores explícitamente y haga un programa para realizar la integración numérica considerando un valor de $k = 1$ y un valor de tamaño de paso de $\tau = 0.1$. Integre las soluciones por un tiempo total $T = 1000\tau$. Imprima el valor de cada solución a cada paso τ , y para cada integrador grafique todas las soluciones en un mismo gráfico p vs θ , con θ variando de 0 a 2π .

Solución

Las ecuaciones de Hamilton son:

$$\frac{dq_i}{dt} = \frac{\partial H}{\partial p_i} \quad (2)$$

$$\frac{dp_i}{dt} = -\frac{\partial H}{\partial q_i} \quad (3)$$

Remplazando el hamiltoniano de la ecuación (1), obtenemos:

$$\frac{d\theta_i}{dt} = p_i \quad (4)$$

$$\frac{dp_i}{dt} = k\sin(\theta_i) \quad (5)$$

Derivando la ecuación (4) y remplazando el resultado en la ecuación (5), obtenemos:

$$\ddot{\theta}_i - k\sin(\theta_i) = 0 \quad (6)$$

Esta ecuación es de segundo orden, para convertirla en una de primer orden hacemos el siguiente cambio de variables:

$$x1(t) = \theta(t) \quad (7)$$

$$x2(t) = \dot{\theta}(t) \quad (8)$$

Con ello obtenemos un sistema de dos ecuaciones diferenciales de primer orden:

$$\dot{x1}(t) = x2(t) \quad (9)$$

$$\dot{x2}(t) = k\sin(x1(t)) \quad (10)$$

Integrador de Euler

Para el caso del integrador de Euler el código es el siguiente:

Listing 1: Código para el integrador de Euler.

```

program euler
implicit none
real(kind=8):: a,b,t,x1,x2,h
real(kind=8), external ::f1,f2
5 integer :: i,imax,n

a=0.0      ! t_o
b=100.0    ! tmax
n=1000.0   ! number of steps
10 h=(b-a)/n ! step size
t=a
x1=0.0     ! initial condition for theta
x2=3.0     ! initial condition for dtheta/dt

15 open(unit=1,file='euler_30.txt',form='formatted',status='replace',
action='READWRITE')

print *, "           time           theta           dtheta/dt /n"

20 do i=1,n+1
    write(unit=1,fmt="(2x,g14.6,2x,g14.6,2x,g14.6)") t,x1,x2
    write(*,'(2x,g14.6,2x,g14.6,2x,g14.6)') t, x1, x2
    t=t+h
    x1=x1+h*f1(x2,t)
    x2=x2+h*f2(x1,t)
enddo
close(unit=1)
end program euler

30 real(kind=8) function f1(x,t)
implicit none
real(kind=8), intent(in) :: x,t
f1=x
end function f1

35 real(kind=8) function f2(x,t)
implicit none
real(kind=8), intent(in) :: x,t
f2=1.0*sin(x)
end function f2
40

```

Los resultados obtenidos se muestran en la figura siguiente:

Integrador de Runge kutta

Para el caso del integrador de Runge Kutta de cuarto orden el código es el siguiente:

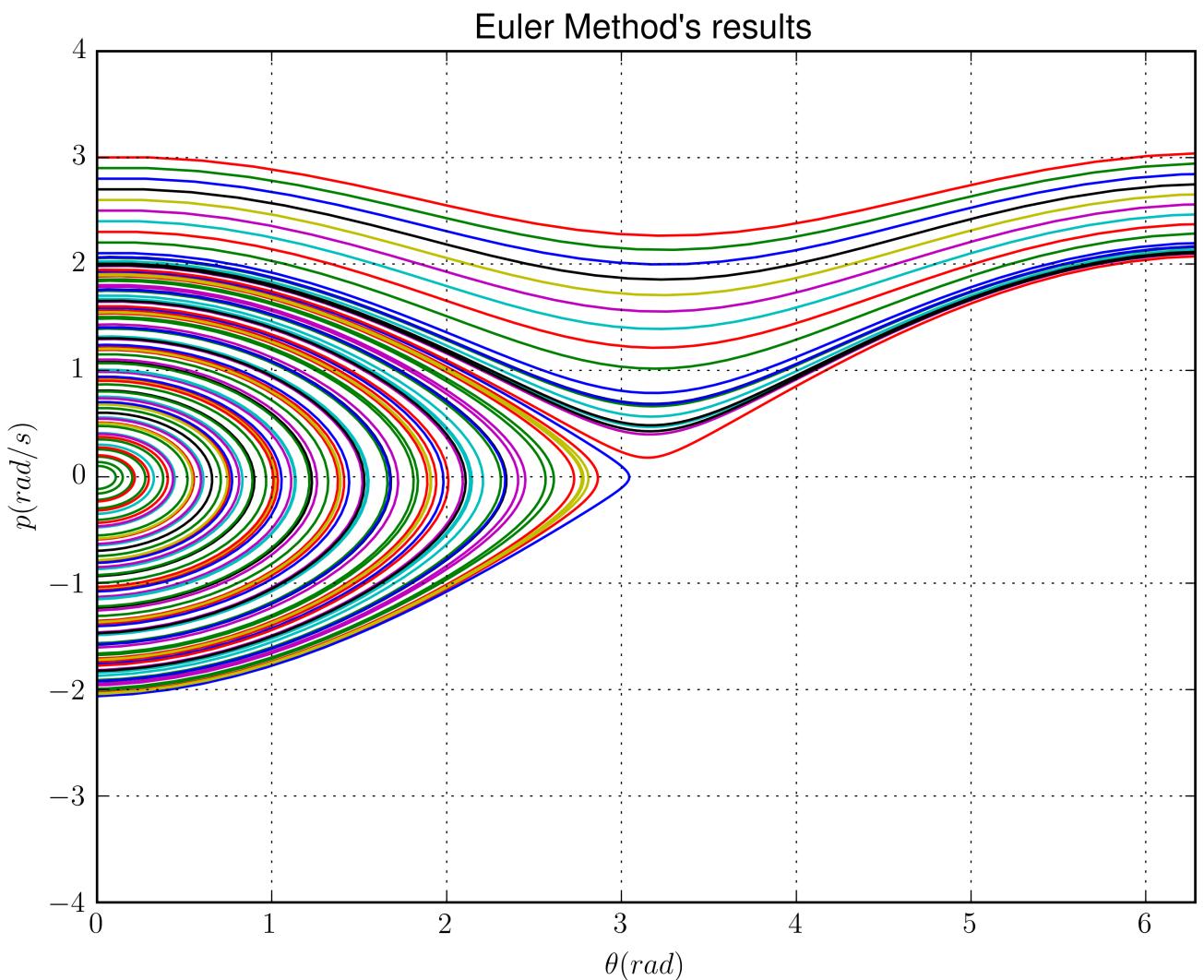


Figure 1: Resultados usando el integrador de Euler.

Listing 2: Código para el integrador de Runge Kutta de cuarto orden.

```

program problem2
implicit none
real(kind=8):: t0,tmax,dt,t1
real(kind=8), dimension(2):: u0,u1
5 integer :: n,n2
external :: f
t0=0.0
tmax=100.0
n=1000.0
10 dt=(tmax-t0)/n
u0(1)=0.0
u0(2)=3.0
n2=2
! k=1.0
15 ! t(1)=a
! t(2)=a

open(unit=1,file='runge_30.txt',form='formatted', status='replace',action='READWRITE' )
do
20   write(unit=1,fmt="(2x,g14.6,2x,g14.6,2x,g14.6)") t0,u0(1),u0(2)
   write(*,'(2x,g14.6,2x,g14.6,2x,g14.6)' ) t0, u0(1), u0(2)
   if (tmax <= t0) then
      exit
   end if
25   t1=t0+dt
   call rk4vec(t0,2,u0,dt,f,u1)
   t0=t1
   u0(1:n2)=u1(1:n2)
end do
30 close(unit=1)
return

end program problem2
subroutine rk4vec ( t0, m, u0, dt, f, u )
35 !*****80
!
!! RK4VEC takes one Runge-Kutta step for a vector ODE.
!
40 ! Licensing:
!
! This code is distributed under the GNU LGPL license.
!
! Modified:
45 !
! 08 October 2013
!
! Author:
!
50 ! John Burkardt
!
! Parameters:

```

```

!
!      Input, real ( kind = 8 ) T0, the current time.
55 !
!      Input, integer ( kind = 4 ) M, the dimension of the system.
!
!      Input, real ( kind = 8 ) U0(M), the solution estimate at the current time.
!
60 !      Input, real ( kind = 8 ) DT, the time step.
!
!      Input, external F, a subroutine of the form
!      subroutine f ( t, m, u, uprime )
!      which evaluates the derivative UPRIME(1:M) given the time T and
65 !      solution vector U(1:M).
!
!      Output, real ( kind = 8 ) U(M), the fourth-order Runge-Kutta solution
!      estimate at time T0+DT.
!
70 implicit none

integer ( kind = 4 ) m

real ( kind = 8 ) dt
75 external f
real ( kind = 8 ) f0(m)
real ( kind = 8 ) f1(m)
real ( kind = 8 ) f2(m)
real ( kind = 8 ) f3(m)
80 real ( kind = 8 ) t0
real ( kind = 8 ) t1
real ( kind = 8 ) t2
real ( kind = 8 ) t3
real ( kind = 8 ) u(m)
85 real ( kind = 8 ) u0(m)
real ( kind = 8 ) u1(m)
real ( kind = 8 ) u2(m)
real ( kind = 8 ) u3(m)
!
90 ! Get four sample values of the derivative.
!
call f ( t0, m, u0, f0 )

t1 = t0 + dt / 2.0D+00
95 u1(1:m) = u0(1:m) + dt * f0(1:m) / 2.0D+00
call f ( t1, m, u1, f1 )

t2 = t0 + dt / 2.0D+00
u2(1:m) = u0(1:m) + dt * f1(1:m) / 2.0D+00
100 call f ( t2, m, u2, f2 )

t3 = t0 + dt
u3(1:m) = u0(1:m) + dt * f2(1:m)
call f ( t1, m, u3, f3 )
105 !

```

```

!   Combine them to estimate the solution U at time T1.
!
110  u(1:m) = u0(1:m) + dt * ( f0(1:m) + 2.0D+00 * f1(1:m) + 2.0D+00 * f2(1:m) &
      + f3(1:m) ) / 6.0D+00
    return
end
subroutine f(t,n,u,uprime)
  implicit none
  integer (kind=4) :: n
115  real (kind=8) :: t,u(n),uprime(n)
  real (kind=8) :: k
  k=1.0
  uprime(1)=u(2)
  uprime(2)=-k*sin(u(1))
120  return
end subroutine f

```

Para el integrador de Runge Kutta de orden cuatro el resultado es el siguiente:

Integrador Simplicético de primer orden

En un integrador simplicético las aproximaciones obtenidas de las variables canónicas en un instante de tiempo y las correspondientes a un instante posterior están relacionadas por una transformación canónica. La evolución temporal del espacio de fases está dada por:

$$u(\tau) = \exp(\tau \hat{H})u(0) \quad (11)$$

Con $\exp(\tau \hat{H})$ de operador de evolución temporal, y \hat{H} dado por:

$$\hat{H} = \sum_{i=1}^n \left(\frac{\partial^2 H}{\partial q_i \partial p_i} - \frac{\partial^2 H}{\partial p_i \partial q_i} \right) \quad (12)$$

Si separamos el hamiltoniano en dos partes:

$$H = H_0 + H_1 \quad (13)$$

La ecuación (11) se puede escribir de la forma:

$$u(\tau) = \exp(\tau \hat{H}_0) \exp(\tau \hat{H}_1) u(0) \quad (14)$$

considerando una aproximación de primer orden. Debido a que el integrador es simplicético:

$$\tilde{u}(\tau) = \exp(\tau \hat{H}_1) u(0) \quad (15)$$

y,

$$u(\tau) = \exp(\tau \hat{H}_0) \tilde{u}(\tau) \quad (16)$$

pueden ser obtenidos por transformaciones canónicas. Como la composición de transformaciones canónicas es una transformación canónica se puede demostrar que :

$$H = H_0(p) + H_1(q) \quad (17)$$

$$q = q_0 + \tau \frac{\partial H_0(p)}{\partial p}(p_0) \quad (18)$$

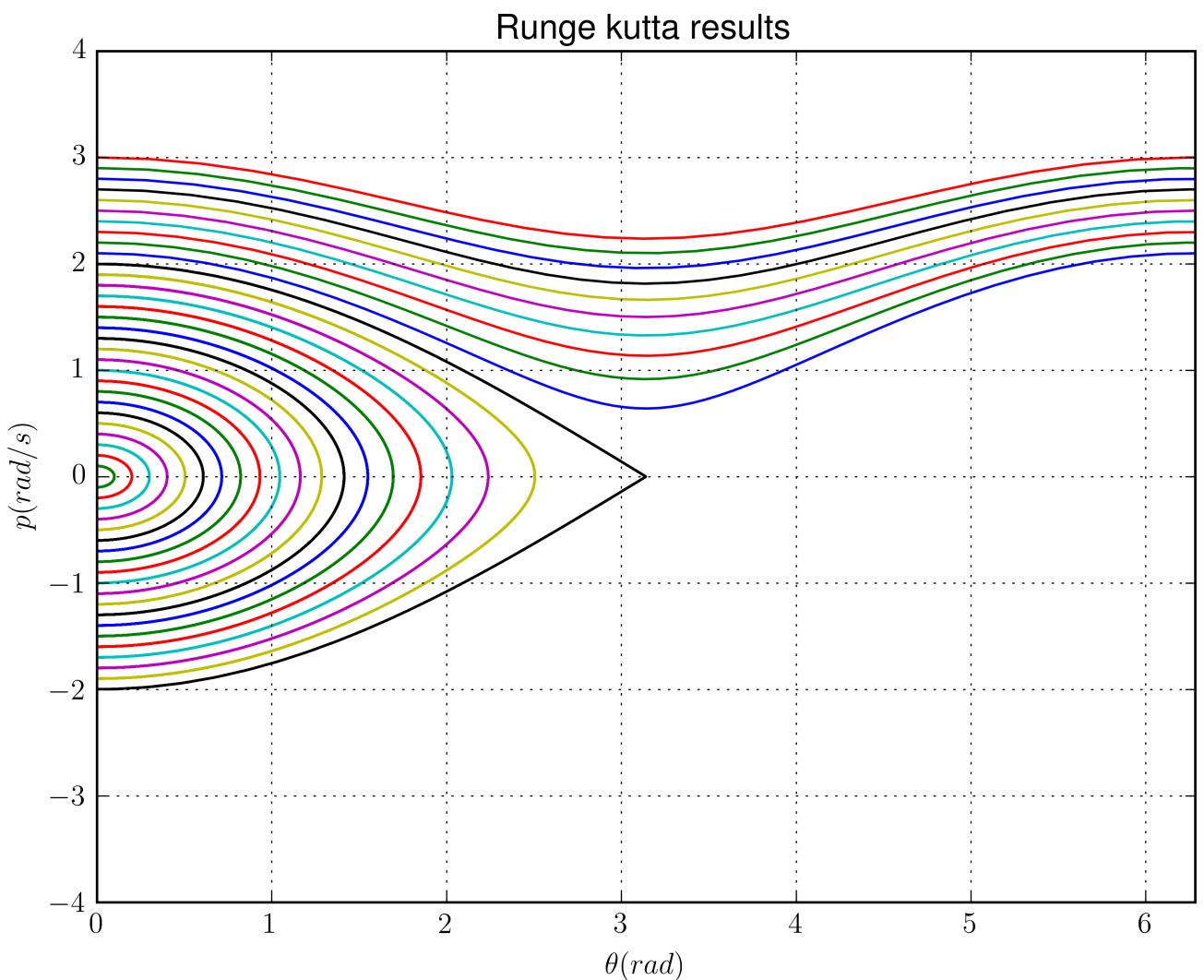


Figure 2: Resultados usando el integrador Runge Kutta.

$$p = p_0 - \tau \frac{\partial H_1(q)}{\partial q}(q) \quad (19)$$

De la ecuación (17) en (1), obtenemos:

$$H_0(p) = \frac{p^2}{2}, \quad (20)$$

$$H_1(\theta) = -k\cos(\theta), \quad (21)$$

con lo cual obtenemos:

$$\frac{\partial H_0}{\partial p} = p, \quad (22)$$

$$\frac{\partial H_1}{\partial \theta} = k\sin(\theta), \quad (23)$$

usando las ecuaciones (18) y (19) obtenemos :

$$\theta = \theta_0 + \tau p(\theta_0) \quad (24)$$

$$p = p_0 - \tau k\sin(\theta) \quad (25)$$

El código del integrador simpléctico es el siguiente:

Listing 3: Código para el integrador de simpléctico.

```

program symplectic
implicit none
real(kind=8):: a,b,t,x1,x2,h,T1,T2
real(kind=8), external ::f1,f2
integer :: i,imax,n

a=0.0      ! t_o
b=100.0    ! tmax
n=1000.0   ! number of steps
10 h=(b-a)/n ! step size
t=a
x1=0.0     ! initial condition for theta
x2=3.0     ! initial condition for dtheta/dt

15 open(unit=1,file='symplectic_30.txt',form='formatted', status='replace',action='REWRITE' )

print *, "           time           theta           dtheta/dt /n"

do i=1,n+1
  write(unit=1,fmt="(2x,g14.6,2x,g14.6,2x,g14.6)") t,x1,x2
  write(*,'(2x,g14.6,2x,g14.6,2x,g14.6)' ) t, x1, x2
  t=t+h
  x1=x1+h*f1(x2,t)
  x2=x2+h*f2(x1,t)
enddo
close(unit=1)
end program symplectic

real(kind=8) function f1(x,t)

```

```

30  implicit none
    real(kind=8), intent(in) :: x,t
    f1=x
end function f1

35  real(kind=8) function f2(x,t)
    implicit none
    real(kind=8), intent(in) :: x,t
    f2=-1.0*sin(x)
end function f2

```

Usando este integrador obtenemos los siguientes resultados:

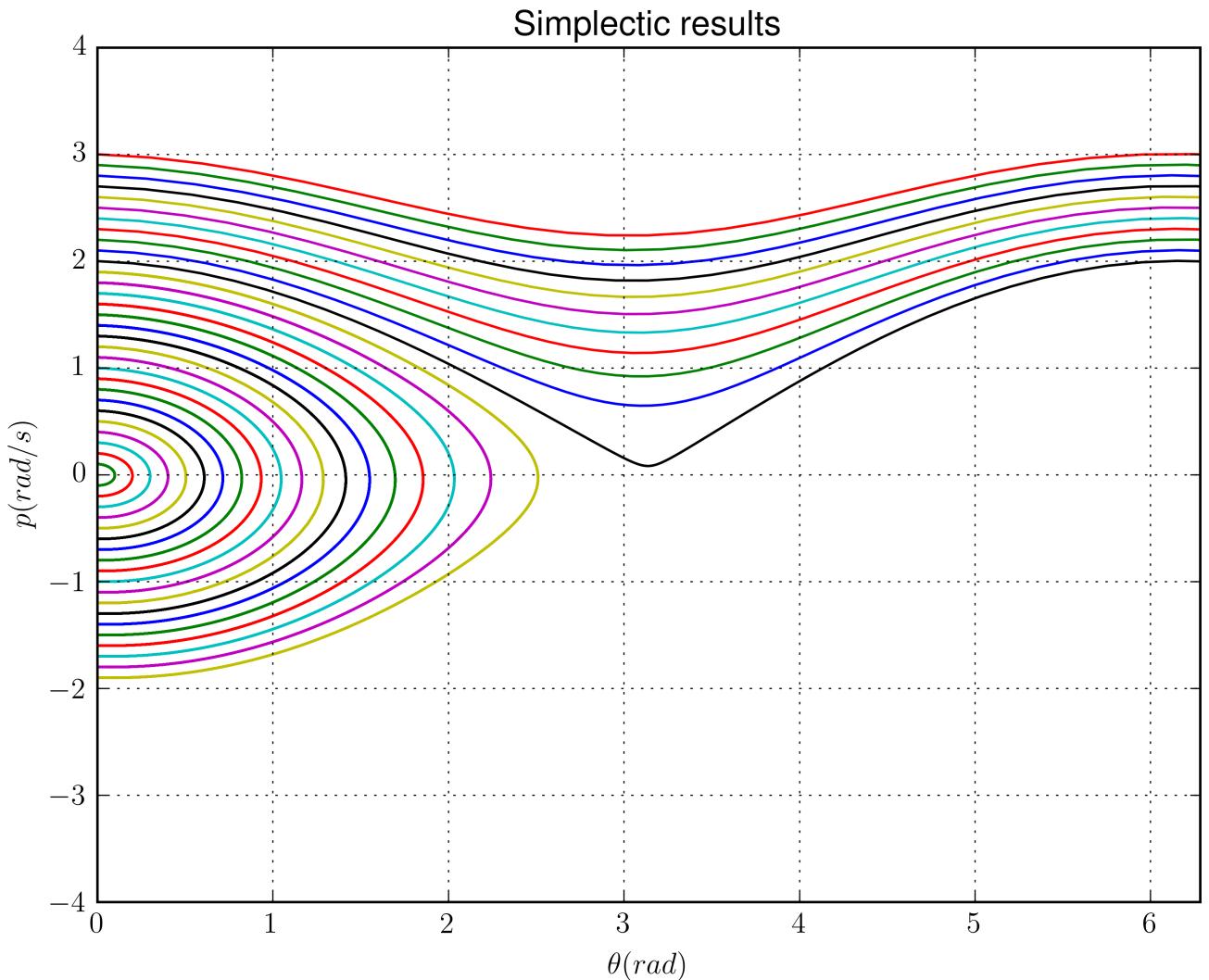


Figure 3: Resultados usando el integrador simpléctico.

Problem 2

Instale el integrador SWIFT, lea el manual del integrador, corra el ejemplo que viene con el, y grafique los resultados de la integración.

Solución

Las propiedades obtenidas luego de correr el ejemplo son las siguientes:

- Excentricidad (e): Parámetro que describe la forma de la elipse, nos dice que tan elongada es la elipse comparada con un círculo. La excentricidad para una órbita circular toma el valor de cero, para una órbita elíptica entre cero y uno, para una órbita parabólica 1, y para una órbita hiperbólica valores mayores a 1.
- Semi-eje mayor (a): Determina el tamaño de la órbita.
- Inclinación (i): Describe el plano de la órbita.
- Longitud de nodo ascendente (Ω): Es el ángulo con vértice en el Sol, que va desde el punto de Aries hasta el nodo ascendente de la órbita del objeto.
- Argumento del perihelio (ω): Es el ángulo que va desde el nodo ascendente hasta el periastro, medido en el plano orbital del objeto y en el sentido de movimiento.
- Anomalía media (m): Es la fracción de un período orbital que ha transcurrido expresada como ángulo.

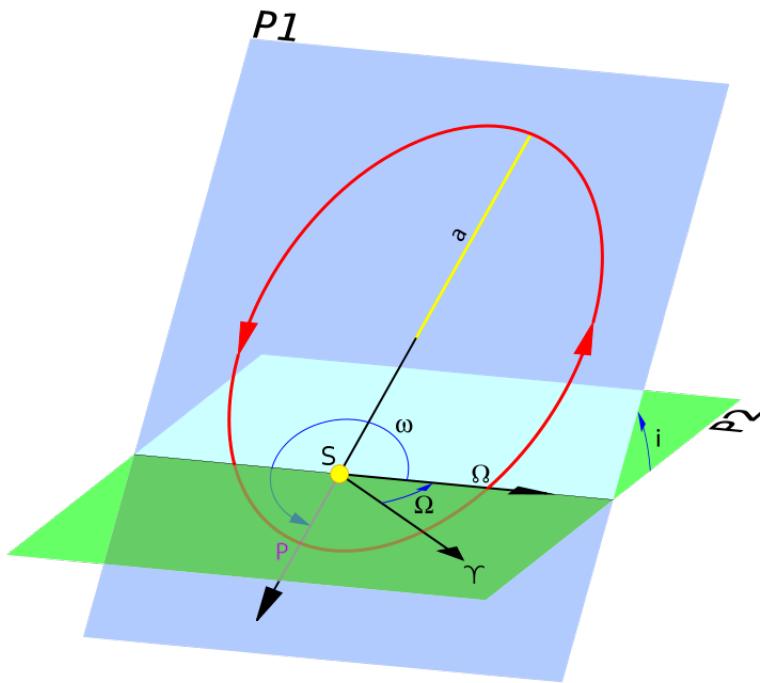
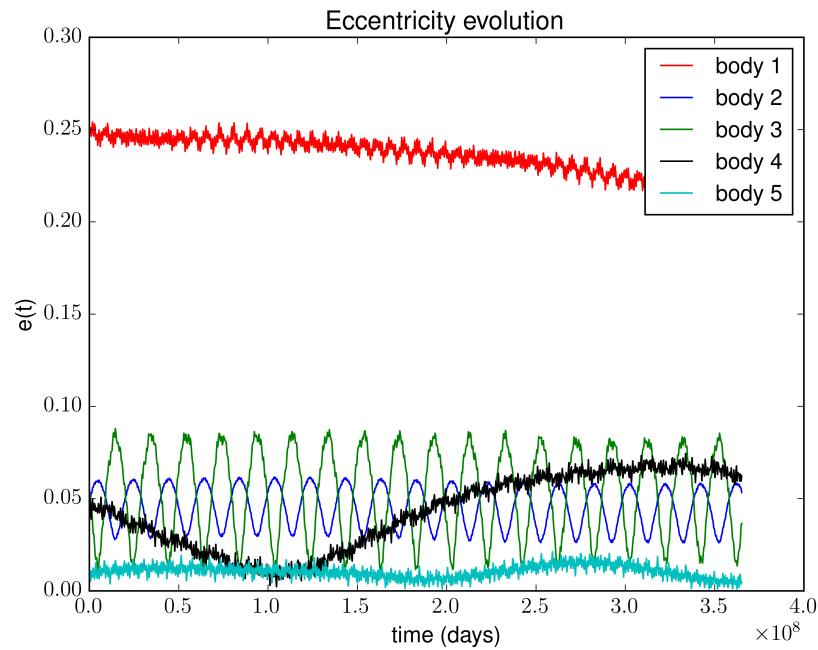
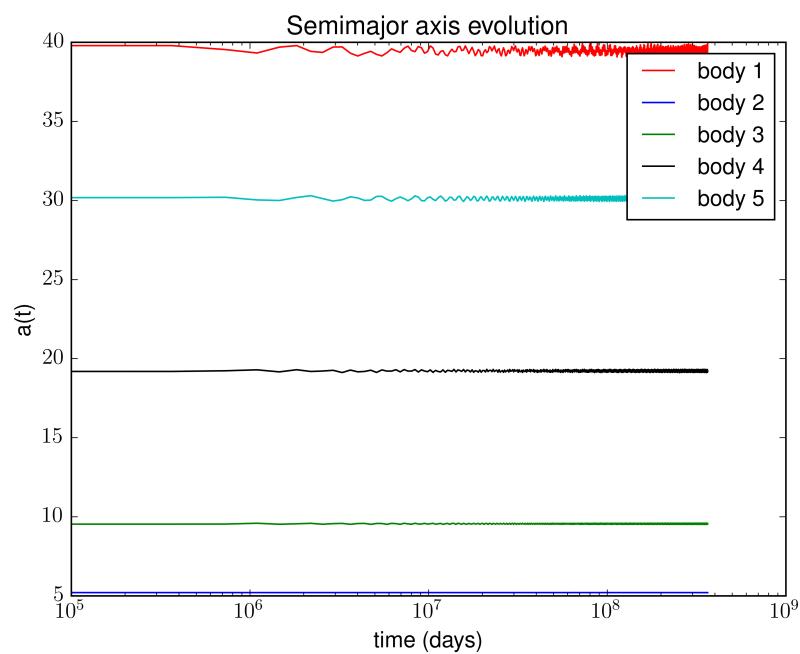
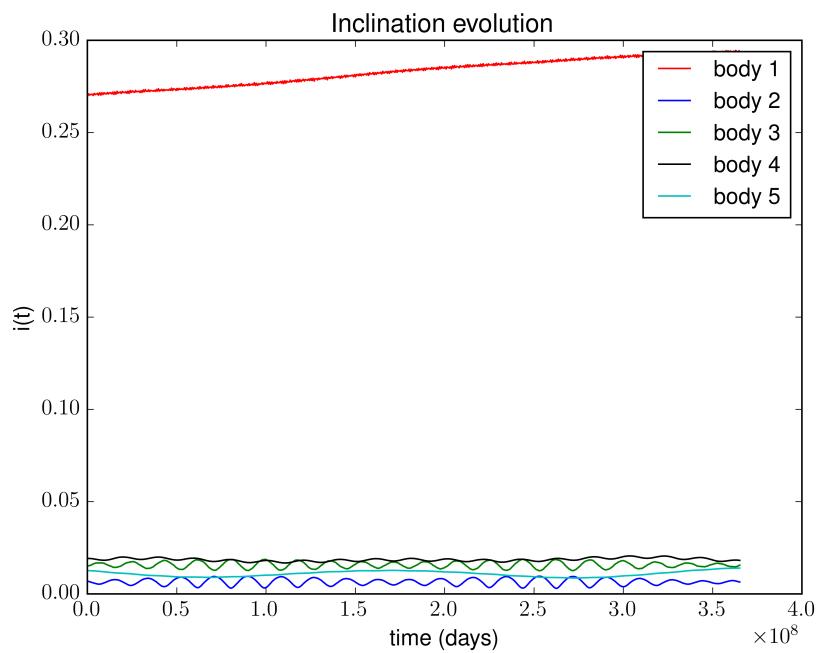
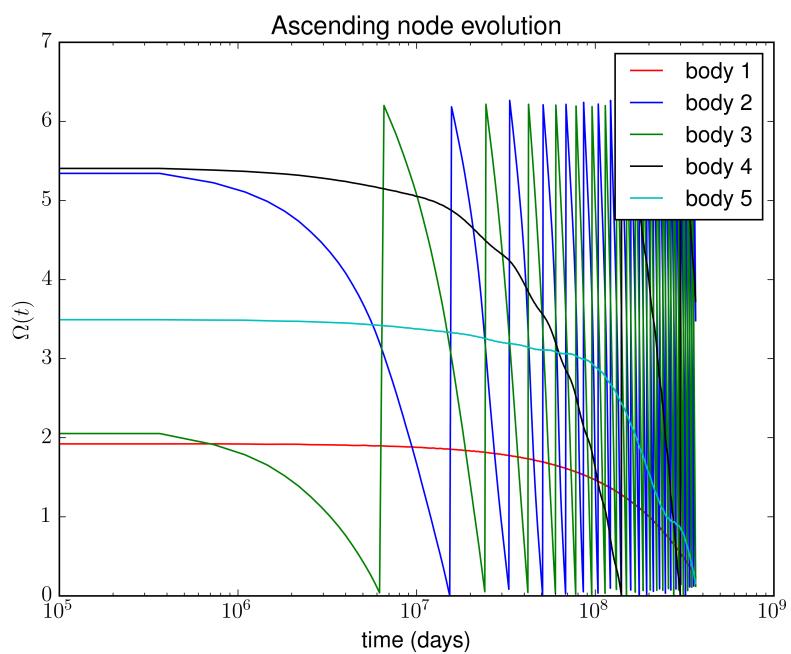


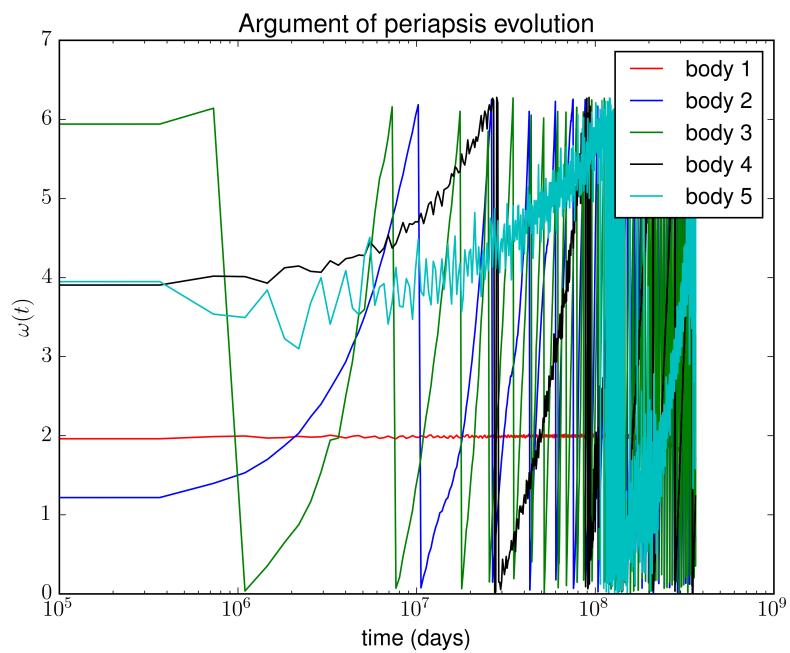
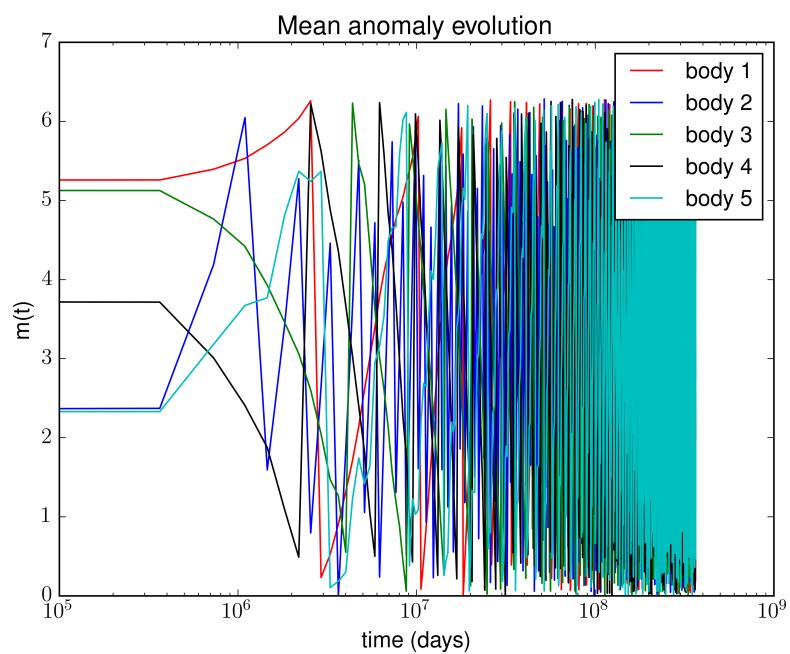
Figure 4: Elementos orbitales.

Los resultados obtenidos se muestran en las siguientes figuras:

De los resultados, es fácil darse cuenta que los cuerpos de color azul, verde, negro, cyan y rojo corresponden a los planetas Júpiter, Saturno, Urano, Neptuno y Plutón respectivamente.

Figure 5: Variación de $e(t)$.Figure 6: Variación de $a(t)$.

Figure 7: Variación de $i(t)$.Figure 8: Variación de $\Omega(t)$.

Figure 9: Variación de $\omega(t)$.Figure 10: Variación de $m(t)$.