para el caso no caótico se uso el siguiente código con ángulo inicial 0.2 y 0.201, aunque variando la subrutina de despliega para poder graficar $\ln|\theta_1 - \theta_2|$ y $|\theta_1 - \theta_2|$

```
    PROGRAM Euler_Cromer
!**********************************************************************
! Se resuelve el pendulo no-lineal, amortiguado y con forzamiento
!
!
!**********************************************************************

 REAL*8, DIMENSION(:), ALLOCATABLE :: theta1,theta2,omega1,omega2,t
 REAL*8 :: length,dt
 INTEGER*8 :: n=15000
!

! print*,"numero de pasos"
! read*, n   ! n=10000
 ALLOCATE (theta1(0:n),theta2(0:n),omega1(0:n),omega2(0:n),t(0:n))
!
!
 call inicializa(theta1, omega1, t, n, length, dt)
 call inicializa(theta2,omega2,t,n,length,dt)
 call calcula(theta1, omega1, t, n, length, dt)
 call calcula(theta2,omega2,t,n,length,dt)
 call despliega (theta1,theta2,t, n, length, dt)
!
END PROGRAM Euler_Cromer
!
!
SUBROUTINE inicializa(theta, omega, t, n ,length, dt)
 INTEGER*8, INTENT (IN) :: n
 REAL*8, DIMENSION(0:n) :: theta,omega,t
 REAL*8 :: length,dt
 print*,'Angulo inicial del pendulo (en radianes)'
 read*, theta(0)
 !print*,'Velocidad angular inicial del pendulo (en radianes/s)'
 !read*, omega(0)
 omega(0)=0
 t(0)=0.
 !print*,'Longitud del pendulo (in m)'
 !read*, length
 length=9.8d0
! print*, 'Tamaño de paso (en segundos)'
 !read*, dt
 dt = 0.004d0
END SUBROUTINE inicializa
!
```

```
!
SUBROUTINE calcula(theta, omega, t, n, length, dt)
 INTEGER*8, INTENT (IN) :: n
 REAL*8, DIMENSION(0:n) :: theta,omega,t
 REAL*8 :: length,dt,g,periodo
 INTEGER :: i
 PI= 4.*ATAN(1.)
 i= 0
 g= 9.80d0
 q=1/2.0d0
 !print*," Amplitud de la fuerza"
 !read*, df
 df=0.5d0
 dfr=2/3.d0
 DO
 omega(i+1) = omega(i) - (g/length) *sin(theta(i)) * dt - q * omega(i)*dt+df*sin(dfr*t
 theta(i+1) = theta(i) + omega(i+1) * dt ! Metodo de Cromer
 if (theta(i+1) > PI ) theta(i+1)=theta(i+1)-2.*PI
 if (theta(i+1) < -PI) theta(i+1)=theta(i+1)+2.*PI
 t(i+1) = t(i) + dt
 IF (i >= n-1) EXIT
 i=i+1
 ENDDO
END SUBROUTINE calcula
SUBROUTINE despliega(theta1,theta2, t, n, length, dt)
 INTEGER*8, INTENT (IN) :: n
 REAL*8, DIMENSION(0:n) :: theta,theta1,theta2,t
 REAL*8 :: length,dt
 INTEGER :: i
 CHARACTER(LEN=10), PARAMETER :: f1 = '(3ES16.6)'

 !print*," archivo de datos"
 !read*, archivo
 do i=1, n, 1
  theta(i)= ABS(theta1(i)-theta2(i))
 end do
 OPEN (UNIT=10,FILE="0p5.dat",STATUS='UNKNOWN')
 !
 do i=1 , n ,10
  WRITE(10,f1) theta(i),t(i)
 end do
 !
 CLOSE(10)
END SUBROUTINE despliega
```

para el caso caótico se uso el siguiente código con ángulo inicial 0.2 y 0.201, aunque variando la subrutina de despliega para poder graficar $\ln|\theta_1 - \theta_2|$ y $|\theta_1 - \theta_2|$

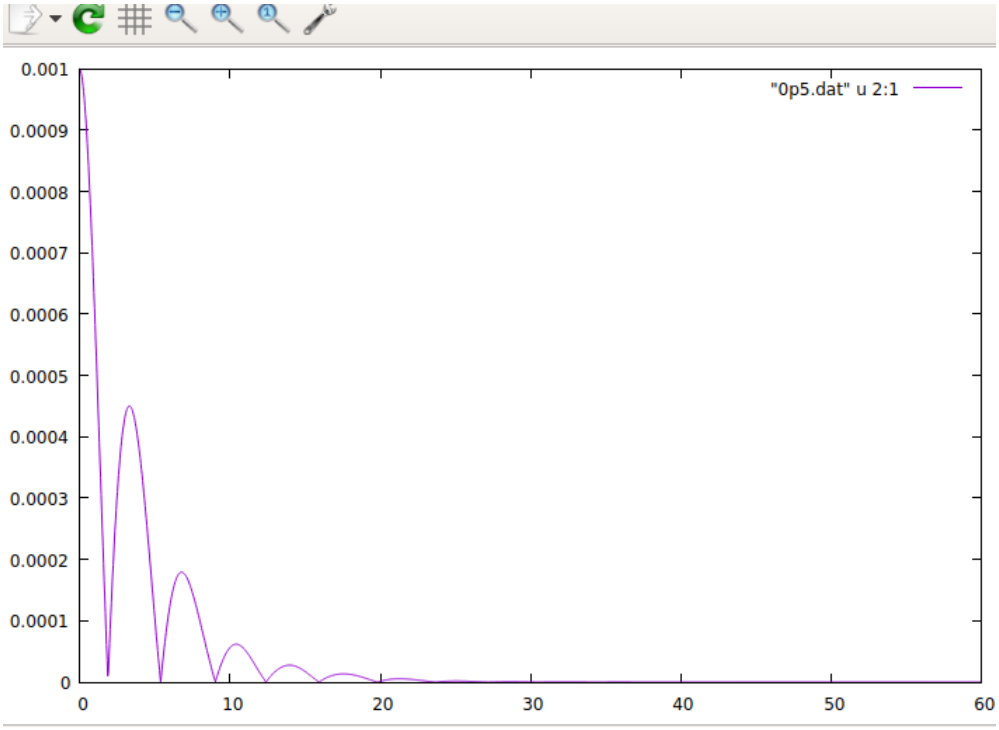

Figura 1: $|\theta_1 - \theta_2|$ vs t

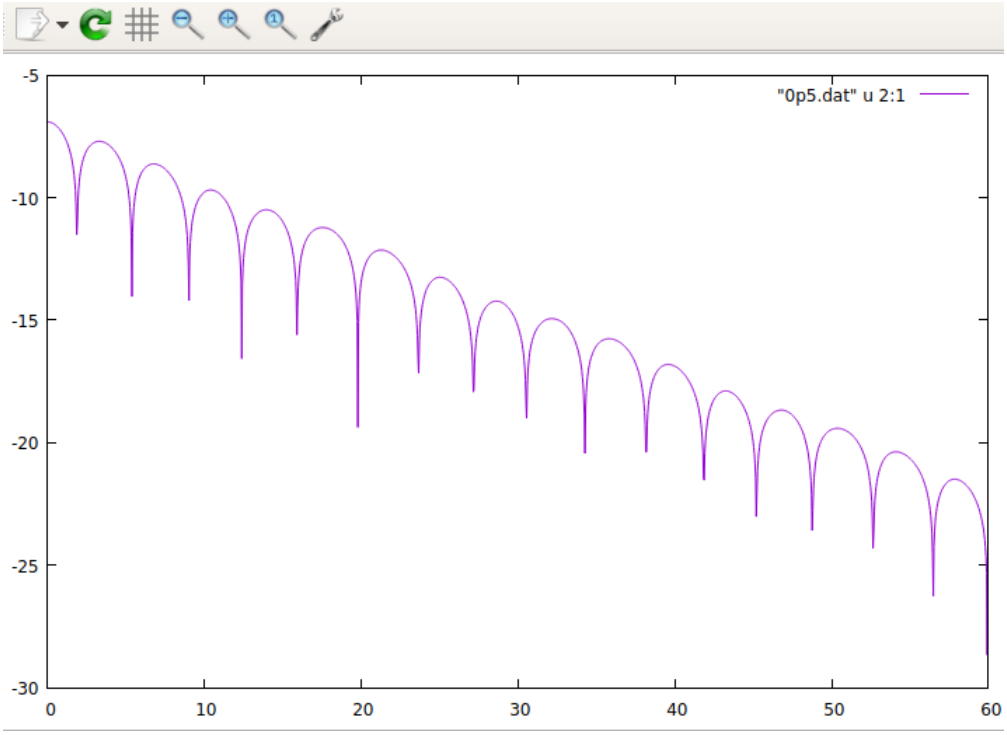

Figura 2: $\ln|\theta_1 - \theta_2|$ vs t

```
    PROGRAM Euler_Cromer
!**********************************************************************
! Se resuelve el pendulo no-lineal, amortiguado y con forzamiento
!
!
!**********************************************************************

 REAL*8, DIMENSION(:), ALLOCATABLE :: theta1,theta2,omega1,omega2,t
 REAL*8 :: length,dt
 INTEGER*8 :: n=40000
!

! print*,"numero de pasos"
! read*, n   ! n=10000
 ALLOCATE (theta1(0:n),theta2(0:n),omega1(0:n),omega2(0:n),t(0:n))
!
!
 call inicializa(theta1, omega1, t, n, length, dt)
 call inicializa(theta2,omega2,t,n,length,dt)
 call calcula(theta1, omega1, t, n, length, dt)
 call calcula(theta2,omega2,t,n,length,dt)
 call despliega (theta1,theta2,t, n, length, dt)
!
END PROGRAM Euler_Cromer
!
!
SUBROUTINE inicializa(theta, omega, t, n ,length, dt)
 INTEGER*8, INTENT (IN) :: n
 REAL*8, DIMENSION(0:n) :: theta,omega,t
 REAL*8 :: length,dt
 print*,'Angulo inicial del pendulo (en radianes)'
 read*, theta(0)
 !print*,'Velocidad angular inicial del pendulo (en radianes/s)'
 !read*, omega(0)
 omega(0)=0
 t(0)=0.
 !print*,'Longitud del pendulo (in m)'
 !read*, length
 length=9.8d0
 !print*, 'Tamaño de paso (en segundos)'
 !read*, dt
 dt=0.004d0
END SUBROUTINE inicializa
!
!
SUBROUTINE calcula(theta, omega, t, n, length, dt)
 INTEGER*8, INTENT (IN) :: n
```

```
 REAL*8, DIMENSION(0:n) :: theta,omega,t
 REAL*8 :: length,dt,g,periodo
 INTEGER :: i
 PI= 4.*ATAN(1.)
 i= 0
 g= 9.80d0
 q=1/2.0d0
 !print*," Amplitud de la fuerza"
 !read*, df
 df=1.20d0
 dfr=2/3.d0
 DO
 omega(i+1) = omega(i) - (g/length) *sin(theta(i)) * dt - q * omega(i)*dt+df*sin(dfr*t
 theta(i+1) = theta(i) + omega(i+1) * dt ! Metodo de Cromer
 if (theta(i+1) > PI ) theta(i+1)=theta(i+1)-2.*PI
 if (theta(i+1) < -PI) theta(i+1)=theta(i+1)+2.*PI
 t(i+1) = t(i) + dt
 IF (i >= n-1) EXIT
 i=i+1
 ENDDO
END SUBROUTINE calcula
SUBROUTINE despliega(theta1,theta2, t, n, length, dt)
 INTEGER*8, INTENT (IN) :: n
 REAL*8, DIMENSION(0:n) :: theta,theta1,theta2,t
 REAL*8 :: length,dt
 INTEGER :: i
 CHARACTER(LEN=10), PARAMETER :: f1 = '(3ES16.6)'

 !print*," archivo de datos"
 !read*, archivo
 do i=1, n, 1
  theta(i)=LOG(ABS(theta1(i)-theta2(i)))
 end do

 OPEN (UNIT=10,FILE="1p2.dat",STATUS='UNKNOWN')
 !
 do i=1, n, 10
  WRITE(10,f1) theta(i),t(i)
 end do
 !
 CLOSE(10)
END SUBROUTINE despliega
```

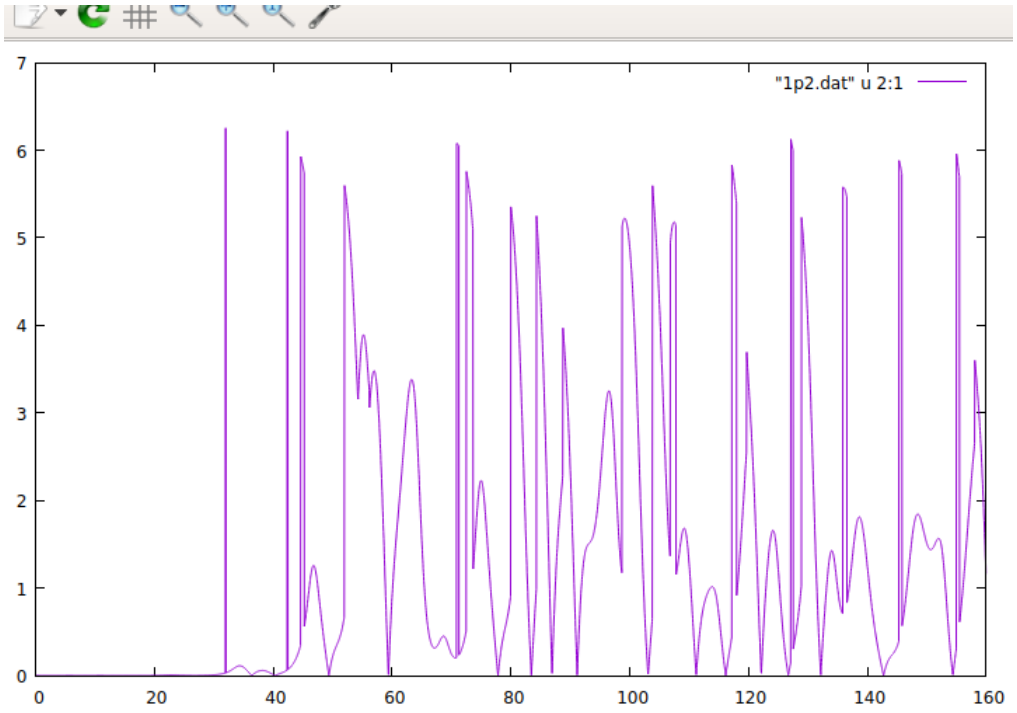

Figura 3: $|\theta_1 - \theta_2|$ vs t

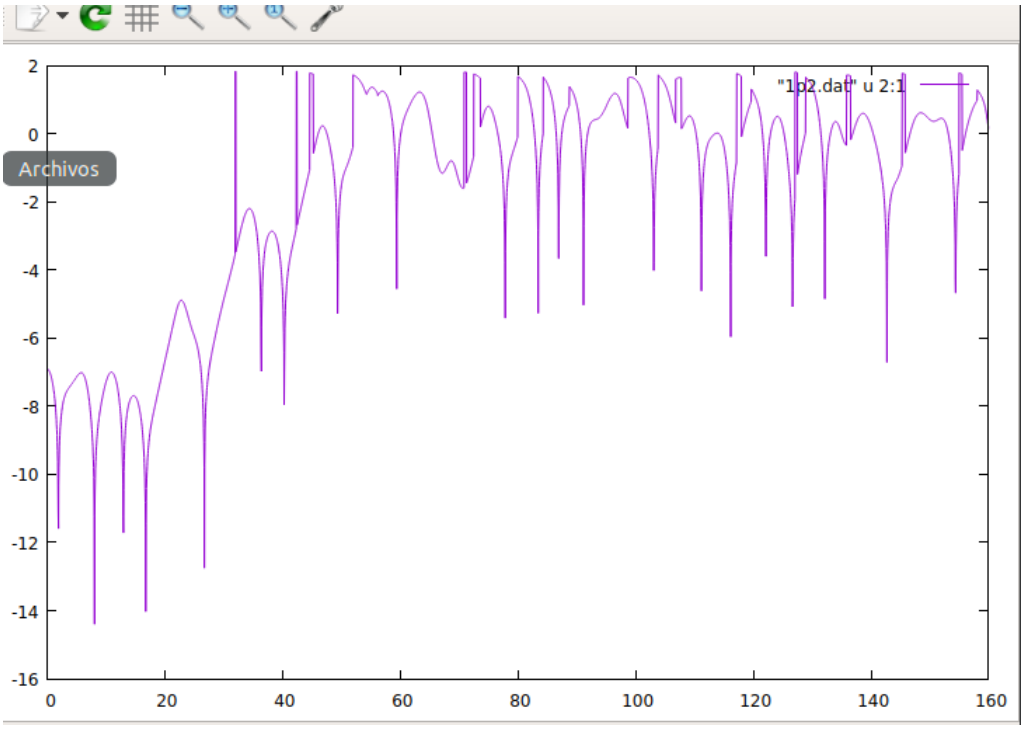

Figura 4: $\ln|\theta_1 - \theta_2|$ vs t