Physics 562 - Computational Physics

Assignment 3: Chaotic Double Pendulum

Josh Fernandes
Department of Physics & Astronomy
California State University Long Beach

March 13, 2014

Abstract

This paper examine emergent chaos by analyzing a double pendulum. A double pendulum is a chaotic system that is nonlinear, deterministic, and highly sensitive to initial conditions. There are three states for the double pendulum to be in: no chaos, emerging chaos, and full chaos. These states correspond to a Lyapunov exponent that is negative, zero, and positive, respectively

1 Single Pendulum

A single pendulum is a very predictable system that undergoes simple harmonic motion. The kinetic energy of a pendulum is given by

$$T = \frac{1}{2}mv^2 = \frac{1}{2}m(l\dot{\theta})^2 \tag{1}$$

and the potential energy is given by

$$V = mgh = mgl(1 - cos(\theta)) \tag{2}$$

where l is the length of the pendulum, and m is the mass of the bob. The lagrangian is given by

$$\mathcal{L} = T - V = \frac{1}{2}m(l\dot{\theta})^2 - mgl(1 - cos(\theta)). \tag{3}$$

The euler-lagrange equations can be applied to the lagrangian in order to get the equations of motions, but instead we will use the hamiltonian. The hamiltonian is given by

$$\mathcal{H} = \Sigma p_{q_i}\dot{q}_i - \mathcal{L} = p_\theta\dot{\theta} - \frac{1}{2}m(l\dot{\theta})^2 + mgl(1 - cos(\theta)). \tag{4}$$

Furthurmore we know that

$$p_\theta = \frac{\partial \mathcal{L}}{\partial \theta} = ml^2\dot{\theta} \tag{5}$$

which can be rewritten

$$\dot{\theta} = \frac{p_\theta}{ml^2} \tag{6}$$

so we can substitute into the hamiltonian to get it of the form

$$\mathcal{H} = \frac{p_\theta^2}{ml^2} - \frac{1}{2}ml^2(\frac{p_\theta}{ml^2})^2 + mgl(1 - cos(\theta)). \tag{7}$$

$$\mathcal{H} = \frac{p_\theta^2}{ml^2} - \frac{1}{2}\frac{p_\theta^2}{ml^2} + mgl(1 - cos(\theta)). \tag{8}$$

$$\mathcal{H} = \frac{1}{2}\frac{p_\theta^2}{ml^2} + mgl(1 - cos(\theta)). \tag{9}$$

We can then solve the hamiltonian equations of motion which are

$$\dot{\theta} = \frac{\partial \mathcal{H}}{\partial p_\theta} = \frac{p_\theta}{ml^2} \tag{10}$$

$$\dot{p_\theta} = -\frac{\partial \mathcal{H}}{\partial \theta} = -mglsin\theta \tag{11}$$

2 Double Pendulum

The double pendulum is more complex. The equations of motion for the double pendulum is given by

$$\dot{\theta_1} = \frac{\partial \mathcal{H}}{\partial p_1} = \frac{lp_1 - lp_2 cos(\theta_1 - \theta_2)}{l^3[m_1 + m_2 sin^2(\theta_1 - \theta_2)]} \tag{12}$$

$$\dot{\theta_2} = \frac{\partial \mathcal{H}}{\partial p_2} = \frac{l(m_1 + m_2)p_2 - lm_2 p_1 cos(\theta_1 - \theta_2)}{l^3 m_2[m_1 + m_2 sin^2(\theta_1 - \theta_2)]} \tag{13}$$

$$\dot{p_1} = -\frac{\partial \mathcal{H}}{\partial \theta_1} = -(m_1 + m_2)glsin(\theta_1) - C_1 + C_2 \tag{14}$$

$$\dot{p_2} = -\frac{\partial \mathcal{H}}{\partial \theta_2} = -m_2 glsin(\theta_2) + C_1 - C_2 \tag{15}$$

where

$$C_1 \equiv \frac{p_1 p_2 sin(\theta_1 - \theta_2)}{l^2[m_1 + m_2 sin^2(\theta_1 - \theta_2)]} \tag{16}$$

$$C_2 \equiv \frac{l^2 m_2 p_1^2 + l^2(m_1 + m_2)p_2^2 - l^2 m_2 p_1 p_2 cos(\theta_1 - \theta_2)}{2l^4[m_1 + m_2 sin^2(\theta_1 - \theta_2)]^2} sin[2(\theta_1 - \theta_2)] \tag{17}$$

In order to understand how the double pendulum behaves, the momentum will be plotted against the angle for each pendulum. In addition, the lyapunov exponent will be calculated for three different cases. These cases are

Listing 1: Module `Cases`

```
1
2     !!Low Energy - No Chaos
3     y(1) = 1._dp*(pi/180)            ! initial angle for 1st pendulum
4     y(2) = 0._dp                  ! initial momentum for 1st pendulum
5     y(3) = 1._dp*(pi/180)            ! initial angle for 2nd pendulum
6     y(4) = 0._dp                  ! initial momentum for 2nd pendulum
7
8     !!Medium Energy - Emerging Chaos But Stable Orbit
9     y(1) = 1._dp*(pi/180)            ! initial angle for 1st pendulum
10    y(2) = 1._dp                  ! initial momentum for 1st pendulum
11    y(3) = 1._dp*(pi/180)            ! initial angle for 2nd pendulum
12    y(4) = 1.01_dp                ! initial momentum for 2nd pendulum
```

```
13
14      !!High Energy - Full Chaos
15      y(1) = 1._dp*(pi/180)              ! initial angle for 1st pendulum
16      y(2) = 1._dp                    ! initial momentum for 1st pendulum
17      y(3) = 1._dp*(pi/180)              ! initial angle for 2nd pendulum
18      y(4) = 1.3_dp                   ! initial momentum for 2nd pendulum
```

The mass for the first bob is 1 kg and the mass of the second bob is 2 kg. The length of both pendulums is 1 meter. The lyapunov exponents are given by

$$|p_1 - p_2| = e^{\lambda \cdot t} \tag{18}$$

where P_1 and p_2 are the momenta for the two pendulums, and λ is the lyapunov exponent.

3 The Fortran95 code

The code solves the equation of motion using the Runga-Kutta method. First a module called `NumType` is created to store all my global parameters.

Listing 2: Module `NumType`

```
1
2   module NumType
3
4       save
5       integer, parameter :: dp = kind(1.d0)
6       real(dp), parameter :: pi = 4*atan(1._dp), &
7       e = exp(1._dp)
8       complex(dp), parameter :: iic = (0._dp,1._dp)
9
10  end module NumType
```

Listing 3: `rkf45.f95`

```
1
2   subroutine rkf45step(t,y,h)  ! 4-th order Runge-Kutta step
3
4       use setup, only : dp, n_eq
5       implicit none
```

```
      real(dp), intent(inout) :: t, h
      real(dp), dimension(n_eq), intent(inout) :: y
      real(dp), dimension(n_eq) :: k1, k2, k3, k4, k5, k6, y1, y2
      real(dp), parameter :: epsilon = 1.e-6_dp, tiny = 1.e-20_dp
      real(dp) :: rr, delta

      call deriv(t,         h,  y,    k1)
      call deriv(t+h/4,     h,  y+ k1/4,   k2)
      call deriv(t+3*h/8,   h,  y+ (3*k1+9*k2)/32,  k3)
      call deriv(t+12*h/13,h,  y+ (1932*k1-7200*k2+7296*k3) &
      /2197 , k4)
      call deriv(t+h,       h,  y+ (439*k1/216-8*k2+3680*k3/513 &
      -845*k4/4104),    k5)
      call deriv(t+h/2,     h,  y+ (-8*k1/27 +2*k2-3544*k3/2565 + &
      1859*k4/4104 -11*k5/40),    k6)

      y1 = y + 25*k1/216 + 1408*k3/2565 + 2197*k4/4104 &
       - k5/5
      y2 = y + 16*k1/135 + 6656*k3/12825 + 28561*k4/ &
      56430 - 9*k5/50 + 2*k6/55

    rr = sqrt(dot_product(y1-y2,y1-y2))/h + tiny

      if ( rr < epsilon ) then
          t = t + h
          y = y1
          delta = 0.92_dp * (epsilon/rr)**(0.2_dp)
          h = delta*h
          write (unit = 3,fmt='(3f20.10)') t, y(1)
          write (unit = 4,fmt='(3f20.10)') t, y(2)
          write (unit = 5,fmt='(3f20.10)') t, y(3)
          write (unit = 6,fmt='(3f20.10)') t, y(4)
          write (unit = 7,fmt='(3f20.10)') y(1), y(2)
          write (unit = 8,fmt='(3f20.10)') y(3), y(4)
          write (unit = 9,fmt='(3f20.10)') -sin(y(1)),-cos(y(1))
          write (unit = 10,fmt='(3f20.10)') -sin(y(1))-sin(y(3)) &
          ,-cos(y(1))-cos(y(3))
      else
          delta = 0.92_dp * (epsilon/rr)**(0.25_dp)
          h = delta*h
```

```
46      end if
47
48
49      contains
50
51          subroutine deriv(t,h,y,k)    ! derivative
52
53              use setup, only : dp, n_eq, g, length, mass1, mass2
54              implicit none
55              real(dp), intent(in) :: t, h
56              real(dp), dimension(n_eq), intent(in) :: y
57              real(dp), dimension(n_eq) :: f, k
58              real(dp) :: c1, c2
59
60              c1 = (y(2)*y(4)*sin(y(1)-y(3)))/ &
61                  (length*length*(mass1+mass2*(sin(y(1)-y(3)))**2))
62              c2 = (length**2*mass2*y(2)**2 + &
63                  length**2*(mass1+mass2)*y(4)**2- &
64                  length*length*mass2*y(2)*y(4)*cos(y(1)-y(3)))/ &
65                  (2*length**4*(mass1+mass2*(sin(y(1)-y(3)))**2)**2)* &
66                  sin(2*(y(1)-y(3)))
67
68              f(1) = (length*y(2) - length*y(4)*cos(y(1)-y(2)))/ &
69                      (length**3*(mass1+mass2*(sin(y(1)-y(2)))**2))
70              f(2) = -(mass1+mass2)*g*length*sin(y(1)) - c1 + c2
71              f(3) = (length*(mass1+mass2)*y(4) - length*&
72                      mass2*y(2)*cos(y(1)-y(2)))/ &
73                      (length**3*mass2*(mass1+mass2*(sin(y(1)-y(2)))**2))
74              f(4) = -mass2*g*length*sin(y(3)) + c1 - c2
75
76              k(1:n_eq) = h*f(1:n_eq)
77
78          end subroutine deriv
```

The main program is adpend and it begins with its own module.

Listing 4: adpend.f95

```
1
2
3   module setup
4
```

```
5       use NumType
6       implicit none
7       integer, parameter :: n_eq = 4
8       real(dp), parameter:: g = 10.0_dp, length = 1.0_dp, &
9        mass1 = 1.0_dp, mass2 = 2.0_dp
10      real(dp) :: t, tmax, dt, lambda
11      real(dp), dimension(n_eq) :: y
12
13  end module setup
14
15  program pendulum
16
17      use setup
18      implicit none
19
20      !!initial conditions!!
21
22      t = 0._dp                       ! time to start
23      tmax = 100._dp                  ! time to exit
24      dt = 0.001_dp                   ! time step
25      lambda = 0._dp                  !intiate a value for lyapanov exponent
26
27
28      !!Below are four cases - Make sure only one set of y is uncommented!!
29
30      !!Low Energy - No Chaos
31      y(1) = 1._dp*(pi/180)           ! initial angle 1
32      y(2) = 0._dp                    ! initial momentum 1
33      y(3) = 1._dp*(pi/180)           ! initial angle 2
34      y(4) = 0._dp                    ! initial momentum 2
35
36      !!Medium Energy - Emerging Chaos But Stable Orbit
37  !   y(1) = 1._dp*(pi/180)           ! initial angle 1
38  !   y(2) = 1._dp                    ! initial momentum 1
39  !   y(3) = 1._dp*(pi/180)           ! initial angle 2
40  !   y(4) = 1.01_dp                  ! initial momentum 2
41
42      !!High Energy - Full Chaos
43  !   y(1) = 1._dp*(pi/180)           ! initial angle 1
44  !   y(2) = 1._dp                    ! initial momentum 1
```

```
!    y(3) = 1._dp*(pi/180)          ! initial angle 2
!    y(4) = 1.3_dp                  ! initial momentum 2

     !!High Energy - Full Chaos
!    y(1) = 60._dp*(pi/180)         ! initial angle 1
!    y(2) = 1._dp                   ! initial momentum 1
!    y(3) = 60._dp*(pi/180)         ! initial angle 2
!    y(4) = 2._dp                   ! initial momentum 2


     !!open all the files that data will be written to!!

     open(unit = 3, file = 'pend_angle1.data', &
          action = 'write', status = 'replace')
     open(unit = 4, file = 'pend_mom1.data', &
          action = 'write', status = 'replace')
     open(unit = 5, file = 'pend_angle2.data', &
          action = 'write', status = 'replace')
     open(unit = 6, file = 'pend_mom2.data', &
          action = 'write', status = 'replace')
     open(unit = 7, file = 'angle_mom1.data', &
          action = 'write', status = 'replace')
     open(unit = 8, file = 'angle_mom2.data', &
          action = 'write', status = 'replace')
     open(unit = 9, file = 'pend_xy1.data', &
          action = 'write', status = 'replace')
     open(unit = 10, file = 'pend_xy2.data', &
          action = 'write', status = 'replace')
     open(unit = 11, file = 'lambda.data', &
          action = 'write', status = 'replace')

     !!calculate the momenta and angle of the pendulums!!

     do while ( t < tmax )
          if ( t + dt > tmax) dt = tmax -t
          call rkf45step(t,y,dt)
          !use the runga kutta method to calculate for theta and momentum
          lambda = lambda + log(abs(y(4)-y(2)))/t
          !update the summation of the lyapanov exponent
          write (unit = 11,fmt='(3f20.10)') lambda
```

```
85          !write lamba to file
86      end do
87
88  end program pendulum
```

The code is run by typing `./pend`. Various data sets are plotted to different files for easy graphing.

4 Results

A single pendulum is very ordered and predictable.

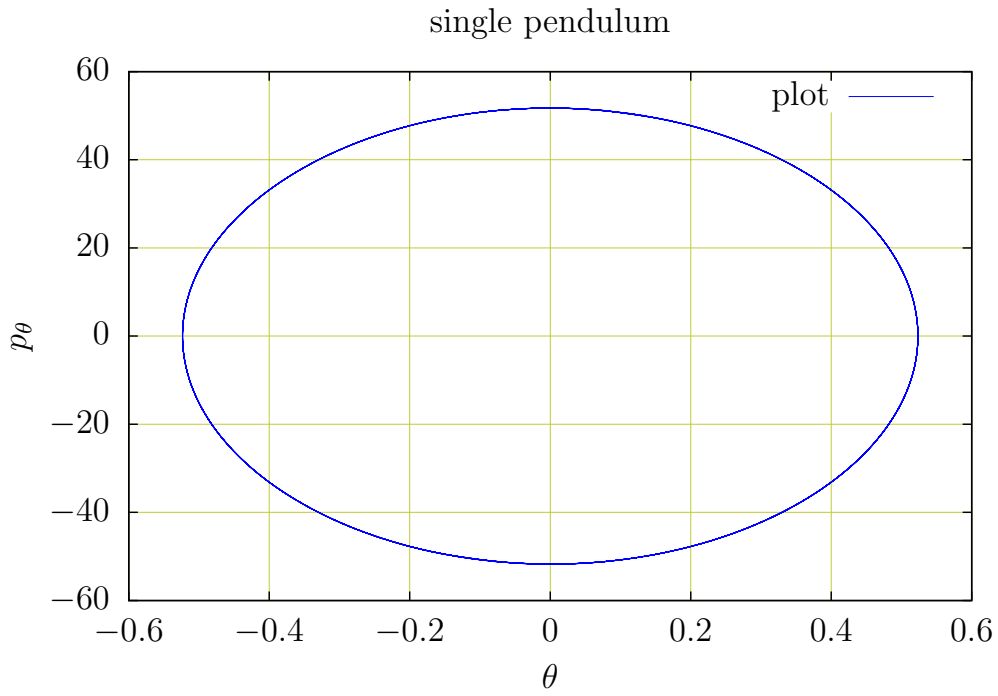

Figure 1: A simple pendulum.

Figure 1 shows that you can predict the values of the momentum for any time t. The momentum is maximum when the pendulum is at the bottom of the arc and zero at the ends of the swing. Because of this, there is no chaos in the pendulums movement.

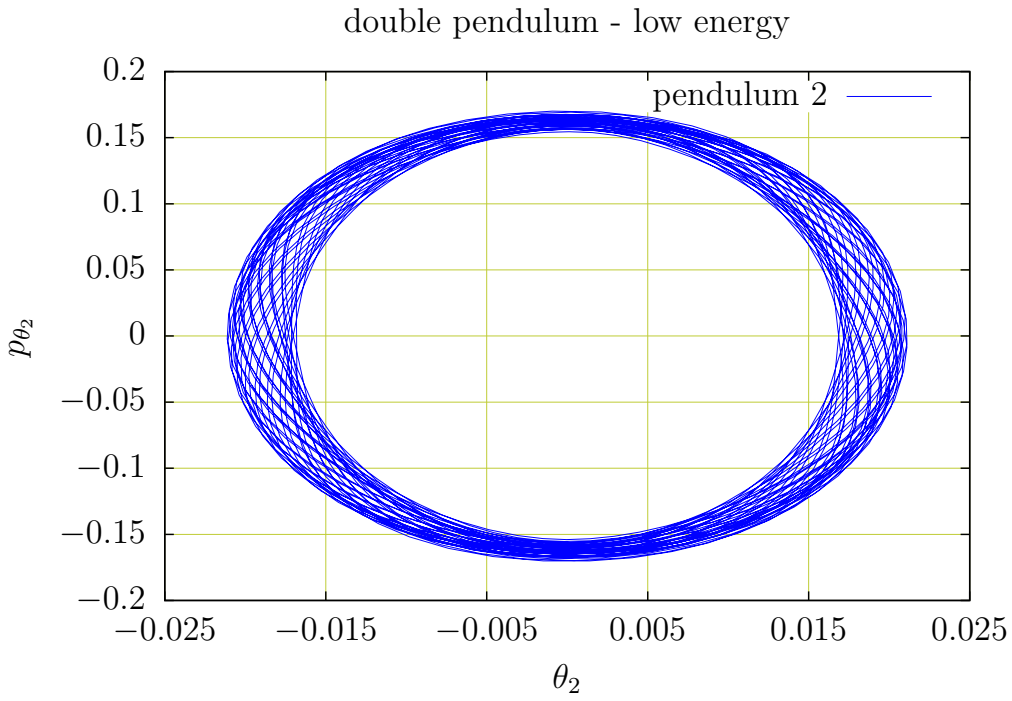

Figure 2: Plot for the second bob.

Figure 2 is similar to the single pendulum. The orbit has slight variation but it is a stable orbit. The calculated lyapanov exponent is a negative number, which confirms that the orbit is closed.

Figure 3

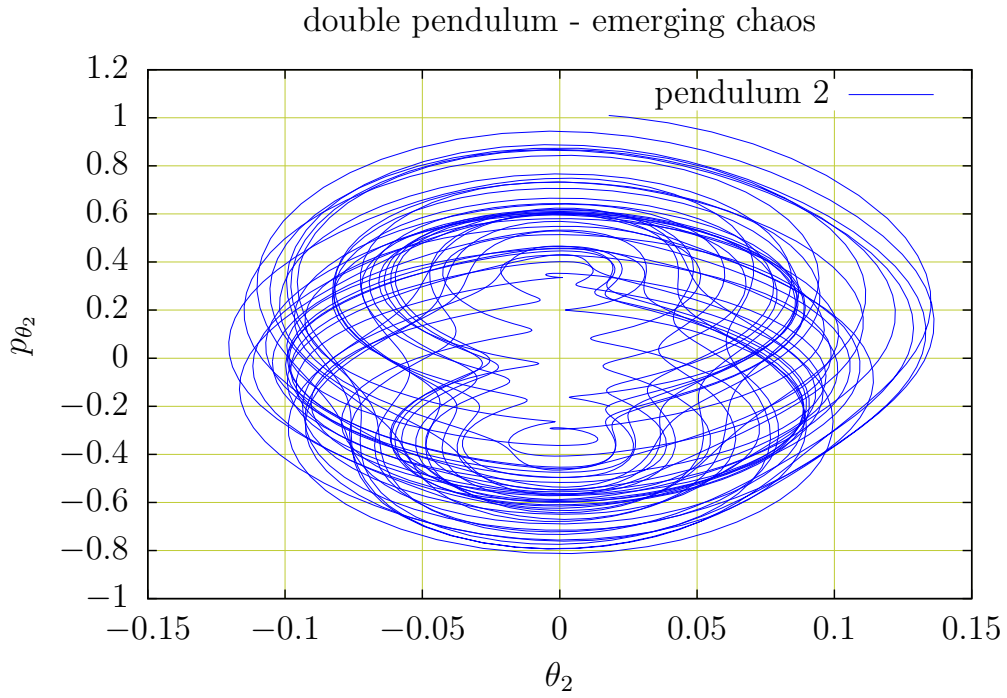

Figure 3: The second bob begins to show chaotic motion, but resumes a stable orbit

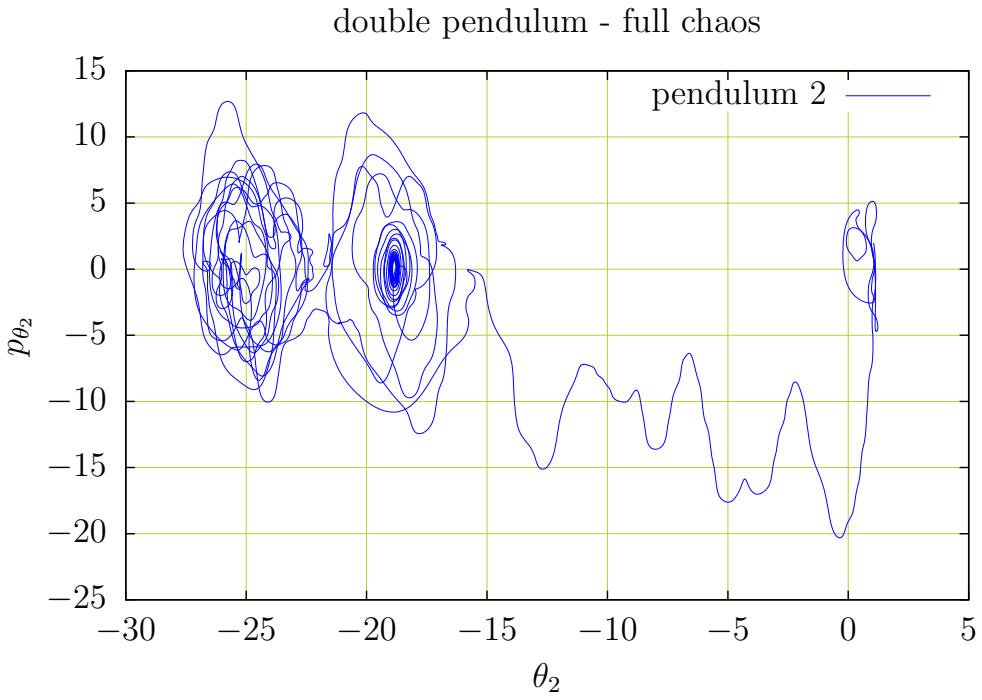

Figure 4: The second bob does not have a stable orbit

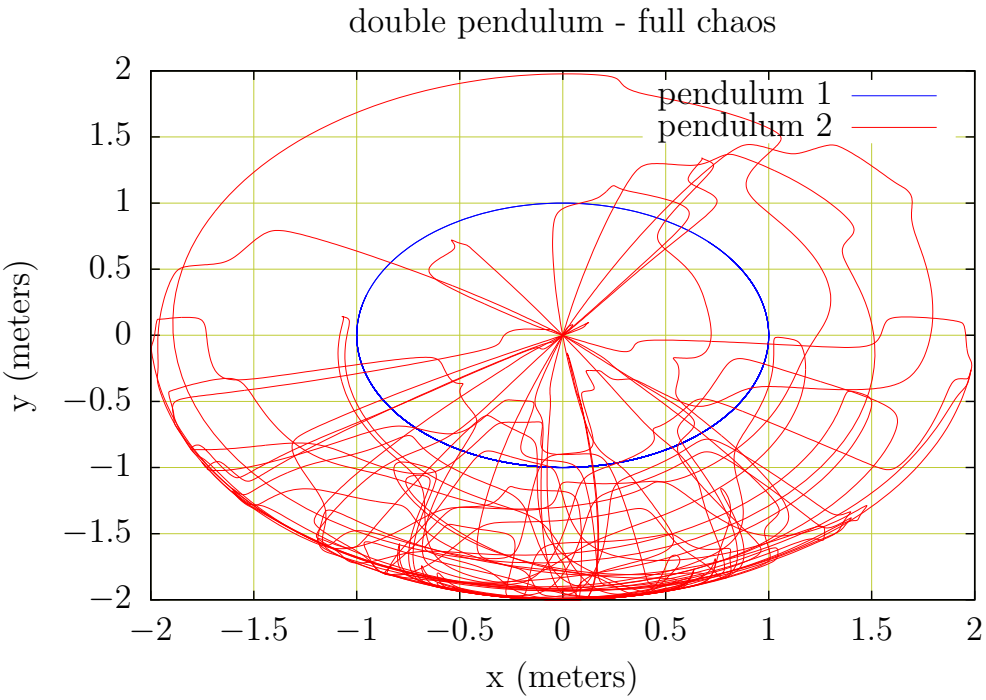

Figure 5: Notice that the first pendulum spins in a full circle at times.

5 Summary and conclusions

The double pendulum is a chaotic system.

References

[1] M. Metcalf, J. Reid and M. Cohen, *Fortran 95/2003 explained*. Oxford University Press, 2004.