Bitácora de la tesis

Pregrado de Física

Santiago Peña Martínez

spenam@unal.edu.co

Inicio 7 Febrero 2019

Índice general

Jueves, 7 Febrero 2019

1. Inicio de la carrera contra reloj

Hoy en la tarde estuve hablando con Óscar, le conté de mis ideas sobre terminar temprano la tésis y acabar la carrera pronto con el fin de irme a Japón. Él se mostró muy entusiasmado con mi idea, me dio consejos para posiblemente llevar a cabo esta tarea (incluyendo el de hacer una bitácora). Esta bitácora va a ser un poco narrativa almismo tiempo de técnica, creo que acá podría expresar mis pensamientos y quizá leerlos en un futuro cuando no tenga nada que hacer, recordar lo que pensaba cuando tenía 20 años y comparar con lo que esté haciendo ahora.
Como soy distraído, estaba estudiando para el parcial del martes de Ondas y recordé que tenía que hacer la bitácora, decidí buscar un template en internet y empezarla ya que de lo contrario puede que nunca la empiece.
Hoy no tengo pensado avanzar en cuanto al proyecto debido al parcial que tengo de Ondas, pero este ejercicio práctico me sirve de alguna manera para llevar cuenta de qué es lo que hago, que casos tomo, cómo hago mis códigos e incluso como práctica para usar latex así esté usando sólamente una template.
P.D. Estoy escuchando August de Sean Angus Watson, creo que esta canción me ha servido muchas veces para estudiar y relajarme. Vamos a ver como nos va!

Martes, 12 Febrero 2019

1. Ecuaciones de Hamilton del sistema y desarrollo para pequeñas oscilaciones

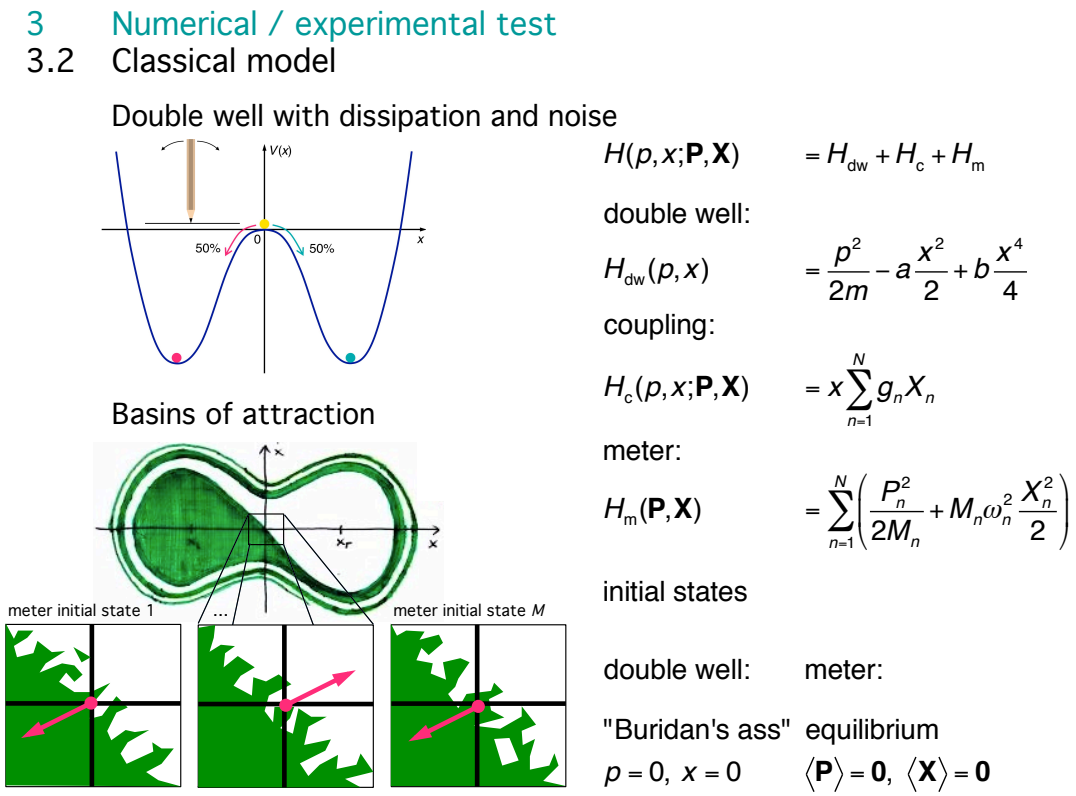

Figura 1: Hamiltonianos del sistema

Para este sistema se tiene el siguiente hamiltoniano:

$$H(p,x;\mathbf{P},\mathbf{X}) = H_{dw} + H_c + H_m \tag{0.1}$$

Con el Hamiltoniano del pozo doble (cuartico) dado por:

$$H_{dw} = \frac{p^2}{2m} - a\frac{x^2}{2} + b\frac{x^4}{4} \tag{0.2}$$

El del acoplamiento dado por:

$$H_c(p,x;\mathbf{P},\mathbf{X}) = x\sum_{n=1}^{N} g_n X_n \tag{0.3}$$

Y el del medidor:

$$H_m(\mathbf{P},\mathbf{X}) = \sum_{n=1}^{N}\Big(\frac{P_n^2}{2M_n} + M_n\omega_n^2\frac{X_n^2}{2}\Big) \tag{0.4}$$

Las ecuaciones de Hamilton para el pozo son:

$$\dot{p} = \frac{-\partial H}{\partial x} = ax - bx^3 - \sum_{n=1}^{N} g_n X_n \tag{0.5}$$

$$\dot{v} = \frac{-1}{m}\frac{\partial H}{\partial x} = \frac{ax}{m} - \frac{bx^3}{m} - \frac{\sum_{n=1}^{N} g_n X_n}{m} \tag{0.6}$$

$$\dot{x} = \frac{\partial H}{\partial p} = \frac{p}{m} \tag{0.7}$$

Las ecuaciones de Hamilton para los osciladores son:

$$\dot{P}_i = \frac{-\partial H}{\partial X_i} = -xg_i - M_i\omega_i^2 X_i \tag{0.8}$$

$$\dot{V}_i = \frac{-1}{M_i}\frac{\partial H}{\partial X_i} = \frac{-xg_i}{M_i} - \omega_i^2 X_i \tag{0.9}$$

$$\dot{X}_i = \frac{\partial H}{\partial P_i} = \frac{P_i}{M_i} \tag{0.10}$$

Ahora bien, para calcular las frecuencias de pequeñas oscilaciones del pozo doble se procede de la siguiente manera. El potencial del pozo está dado por:

$$V(x) = -\frac{1}{2}ax^2 + \frac{1}{4}bx^4 \tag{0.11}$$

Los puntos de equilibrio ocurren cuando el potencial V es máximo o mínimo, es decir:

$$\frac{d}{dx}V = 0 \to x(bx^2 - a) = 0 \tag{0.12}$$

Las raices de la ecuación anterior son $x = 0, \pm\sqrt{\frac{a}{b}}$, donde $x = 0$ es un punto de equilibrio inestable y $x = \pm\sqrt{\frac{a}{b}}$ son puntos de equilibrio estables.
Ahora bien, hacemos una expansión de Taylor alrededor del punto de equilibrio estable $x_0 = \sqrt{\frac{a}{b}}$:

$$U(x_0+\epsilon) = U(x_0) + U'(x_0)\epsilon + \frac{1}{2}U''(x_0)\epsilon^2 + ... \tag{0.13}$$

El primer término es una constante entonces se puede ignorar, el segundo término es cero ya que $U'(x_0) = 0$ y el tercer término será $\frac{1}{2}(-a + 3bx^2)|_{x=x_0}$, esto da como resultado:

$$U = \frac{1}{2}(2a)\epsilon^2 \tag{0.14}$$

Es claro que la constante elástica de este potencial es $2a$ (por comparación con $U = \frac{1}{2}kx^2$). Esto quiere decir que la frecuencia de pequeñas oscilaciones en estos puntos de equilibrio están dadas por:

$$\omega = \sqrt{\frac{k}{m}} = \sqrt{\frac{2a}{m}} \tag{0.15}$$

2. Códigos para las ecuaciones de Hamilton de para 1 oscilador acoplado

```
def A(m0, a0, b0, x0, g0, X0): #p punto
    return a0*x0/m0 - b0*x0*x0*x0/m0 -(g0*X0)/m0
def V(m0,p0): #equis punto
    return p0/m0
def Ai(Mi0,wi0,x0,gi0,Xi0): #P punto sub i
    return -x0*gi0/Mi0-wi0**2.*Xi0
def Vi(Mi0,Pi0): #Equis punto sub i
    return Pi0/Mi0
```

3. Código para el integrador de Ruth de 4to orden

```
def ruth4th(xx,pp,mm,XX01,PP01,MM1,ww1,gacople,dt,
iteraciones,a_poso,b_poso)#defino funcion con el integrador simplectico Ruth
    de 4to orden

dos=2.**(1./3.) #constante necesaria

aa=[a,(1./(2.-dos)),(-dos/(2.-dos)),(1./(2.-dos)),0] #constantes de la
    posicion del integrador
bb=[b,(1./(2.*(2.-dos))),(1.-dos)/(2.*(2.-dos)), (1.-dos)/(2.*(2.-dos))
    ,(1./(2.*(2.-dos)))] #constantes de la velocidad del integrador

    x = [xx] #posicion inicial
    p = [pp] #momento inicial
    X1 = [XX01] #posicion inicial del primer oscilador
    P1 = [PP01] #momento inicial del primer oscilador
    t  = [0.] #lista de tiempo

    xi , pi , X1i , P1i= xx, pp , XX01 , PP01
    for i in range (iteraciones):
        vini = pi/m
        vv = V(mm,pi)+bb[1]*A(mm,a_poso,b_poso,xi,gacople,X1i)*dt
        VV1 = Vi(MM1,P1i)+bb[1]*Ai(MM1,ww1,xi,gacople[1],X1i)*dt
        xx = xi + aa[1]*vv*dt
```

```
        XX1 =XX1i + aa[1]*VV1*dt
        for j in range(2,len(aa)):
            vv += bb[j]*A(mm,a_poso,b_poso,xx,gacople,XX1)*dt
    VV1 += bb[j]*Ai(MM1,ww1,xx,gacople[1],XX1)*dt
            xx += aa[j]*vv*dt
            XX1 += aa[j]*VV1*dt

        xi = xx
        pi = m*vv
        X1i = XX1
        P1i = MM1*VV1

        x.append(xi)
        p.append(pi)
        X1.append(X1i)
        P1.append(P1i)
        t.append(t[-1] + dt)
```

4. Código para integrador de Yoshida de 6to orden

```
#codigo con el integrador de yoshida de sexto orden, los parametros son xx=
    posicion inicial, pp=momento inicial, XX01=posicion inicial del primer
    oscilador, PP01=momento inicial del primer oscilador, MM1=masa del primer
     oscilador, ww1=frecuencia del primer oscilador, gacople= lista de los
    acoples para cada oscilador, dt= paso del integrador, iteraciones= numero
     de iteraciones del integrador, a_poso=parametro a del poso, b_poso=
    parametro b del poso,

def yoshida(xx,pp,mm,XX01,PP01,MM1,ww1,gacople,dt,iteraciones,a_poso,b_poso):
     #defino funcion con el integrador simplectico yoshida de 6to orden


aa = [ a, 0.78451361047756, 0.23557321335936,  -1.1776799841789,
    1.3151863206839,  -1.1776799841789,  0.23557321335936, 0.78451361047756,
    0]
bb = [ b, 0.39225680523878, 0.51004341191846, -0.47105338540976,
    0.068753168252520, 0.068753168252520, -0.47105338540976,
    0.51004341191846, 0.39225680523878]


    x = [xx] #posicion inicial
    p = [pp] #momento inicial
    X1 = [XX01] #posicion inicial del primer oscilador
    P1 = [PP01] #momento inicial del primer oscilador
    t  = [0.] #lista de tiempo

    xi , pi , X1i , P1i= xx, pp , XX01 , PP01
    for i in range (iteraciones):
        vini = pi/m
        vv = V(mm,pi)+bb[1]*A(mm,a_poso,b_poso,xi,gacople,X1i)*dt
        VV1 = Vi(MM1,P1i)+bb[1]*Ai(MM1,ww1,xi,gacople[1],X1i)*dt
        xx = xi + aa[1]*vv*dt
        XX1 =XX1i + aa[1]*VV1*dt
```

```
        for j in range(2,len(aa)):
            vv += bb[j]*A(mm,a_poso,b_poso,xx,gacople,XX1)*dt
    VV1 += bb[j]*Ai(MM1,ww1,xx,gacople[1],XX1)*dt
            xx += aa[j]*vv*dt
            XX1 += aa[j]*VV1*dt

        xi = xx
        pi = m*vv
        X1i = XX1
        P1i = MM1*VV1

        x.append(xi)
        p.append(pi)
        X1.append(X1i)
        P1.append(P1i)
        t.append(t[-1] + dt)

```

5. Resumen y lo que falta por hacer de hoy

Los desarrollos anteriores fueron hechos con el fin de tener a la mano las ecuaciones útiles a las cuales vuelvo constantemente. Los códigos por otra parte, son tentativos. Lo que debo hacer en la próxima sesión:

- Comprobar que los códigos propuestos funcionen, con casos de prueba
- Agregar un código de Ruth de 3er orden
- Una vez que los códigos funcionen, crear un código para el sistema mio que llame las funciones de los integradores
- Cuando lo anterior sea hecho, agregar los códigos al repositorio de github

6. This is another example experiment

Figura 2: Example figure.

7. Rápidamente como recuperar los archivos que estaba copiando en el ssh

```
rsync -P -e ssh spenam@168.176.92.190:mec_estadistica/lambda09.dat /home/
    santiago/Downloads/mec_estadistica/proycod/09
```

Sábado, 16 Febrero 2019

1. Código de Ruth de 3er orden

```
def ruth3rd(xx,pp,mm,XX01,PP01,MM1,ww1,gacople,dt,
iteraciones,a_poso,b_poso)#defino funcion con el integrador simplectico Ruth
    de 3er orden

aa=[a,-1./24.,3./4.,7./24.] #constantes de la posicion del integrador
bb=[b,1.,-2./3., 2./3.] #constantes de la velocidad del integrador

    x = [xx] #posicion inicial
    p = [pp] #momento inicial
    X1 = [XX01] #posicion inicial del primer oscilador
    P1 = [PP01] #momento inicial del primer oscilador
    t  = [0.] #lista de tiempo

    xi , pi , X1i , P1i= xx, pp , XX01 , PP01
    for i in range (iteraciones):
        vini = pi/m
        vv = V(mm,pi)+bb[1]*A(mm,a_poso,b_poso,xi,gacople,X1i)*dt
        VV1 = Vi(MM1,P1i)+bb[1]*Ai(MM1,ww1,xi,gacople[1],X1i)*dt
        xx = xi + aa[1]*vv*dt
        XX1 =XX1i + aa[1]*VV1*dt
        for j in range(2,len(aa)):
            vv += bb[j]*A(mm,a_poso,b_poso,xx,gacople,XX1)*dt
      VV1 += bb[j]*Ai(MM1,ww1,xx,gacople[1],XX1)*dt
            xx += aa[j]*vv*dt
            XX1 += aa[j]*VV1*dt

        xi = xx
        pi = m*vv
        X1i = XX1
        P1i = MM1*VV1

        x.append(xi)
        p.append(pi)
        X1.append(X1i)
        P1.append(P1i)
        t.append(t[-1] + dt)
```

2. más dilemas con los códigos!!!!

Hice los códigos usando los loops, para intentar hacer casos de prueba tomé el potencial caótico de Henon-Heiles, pero oh sorpresa, no obtenía la misma figura. Obtenía algo cercano pero no exactamente lo que necesitaba.

En un momento de desesperación, busqué alternativas. Estas posibles alternativas llegaron en Mathematica y voy a probar en Matlab(Octave).

Ocurre que busqué en internet posibles integradores simplécticos de Mathematica y obtengo resultados un poco distintos para los mismo parámetros a los que ya les había hecho 123081324796 gráficas! Algo bueno es que estas gráficas de Mathematica también dependen aparentemente del paso del integrador.

Estas imágenes muestran lo que quiero decir para los parámetros dados por: $m = 0{,}1$, $a = 0{,}25$, $b = 0{,}01$, $M = 0{,}01$, $\omega = 2{,}236067977$,$g = 0{,}01$ y condiciones iniciales dadas por $P(0) = 0{,}1, X(0) = 0{,}1$ para un tiempo t =300:

Todo esto usando integradores simplécticos de 4to orden.

- Para $dt = 0{,}001$:

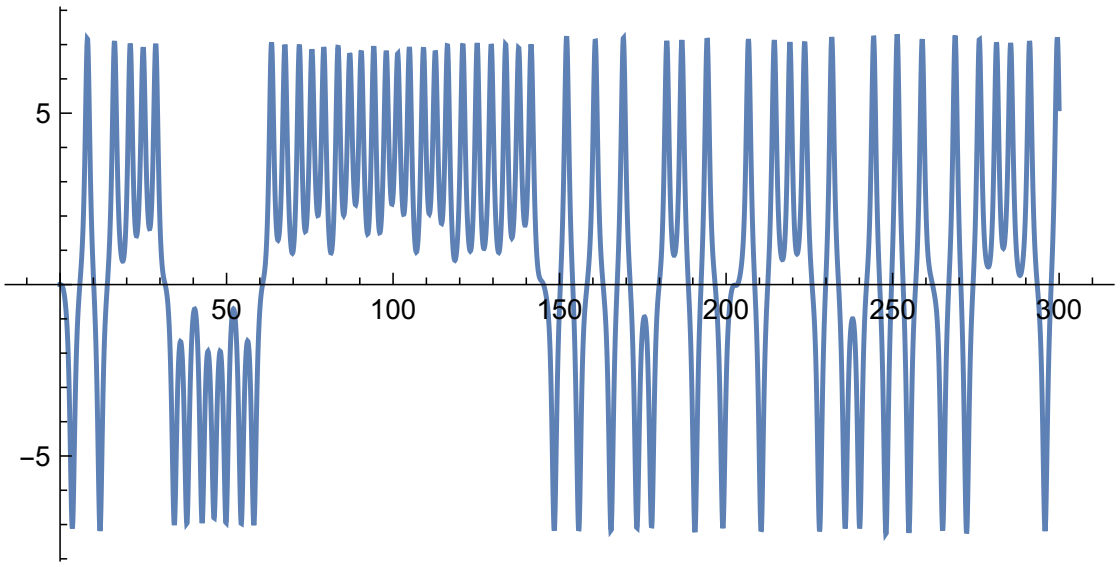

Figura 1: Gráfica x vs t en Mathematica para $dt = 0{,}001$

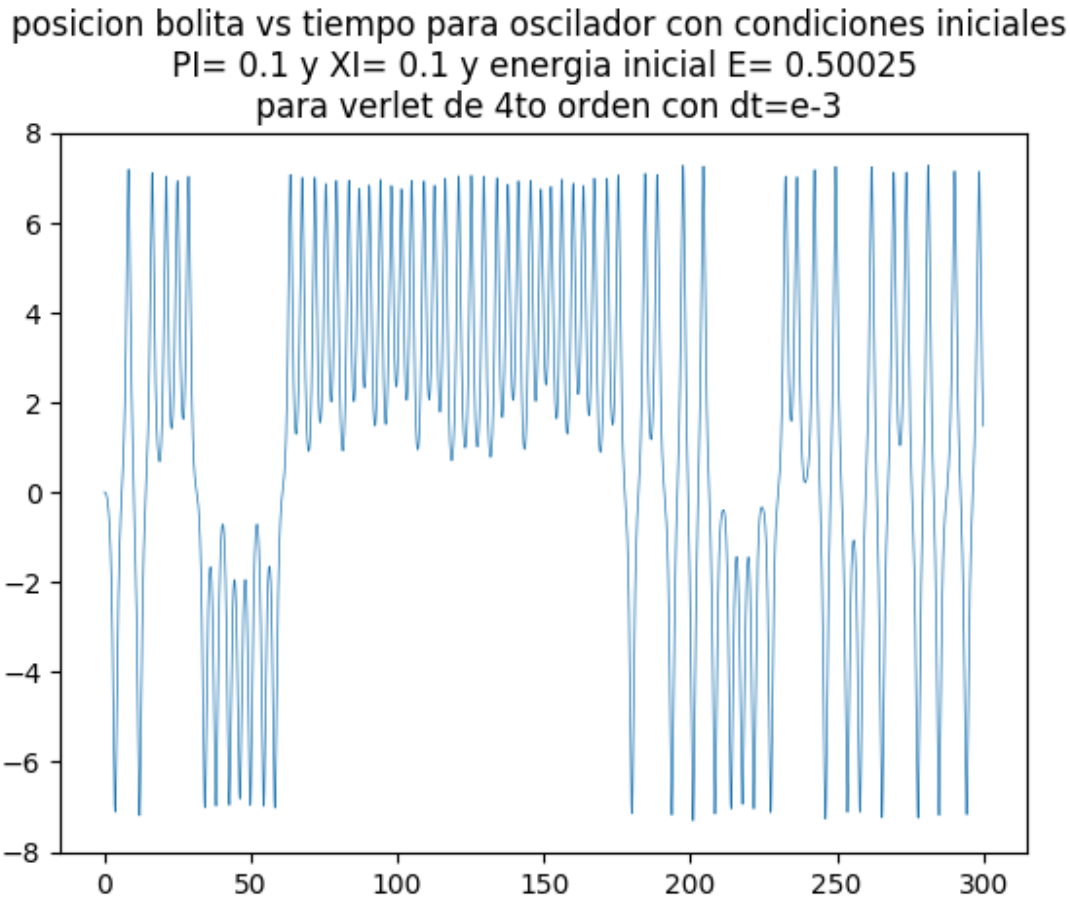

Figura 2: Gráfica x vs t en Python para $dt = 0{,}001$

- Para $dt = 0{,}0001$:

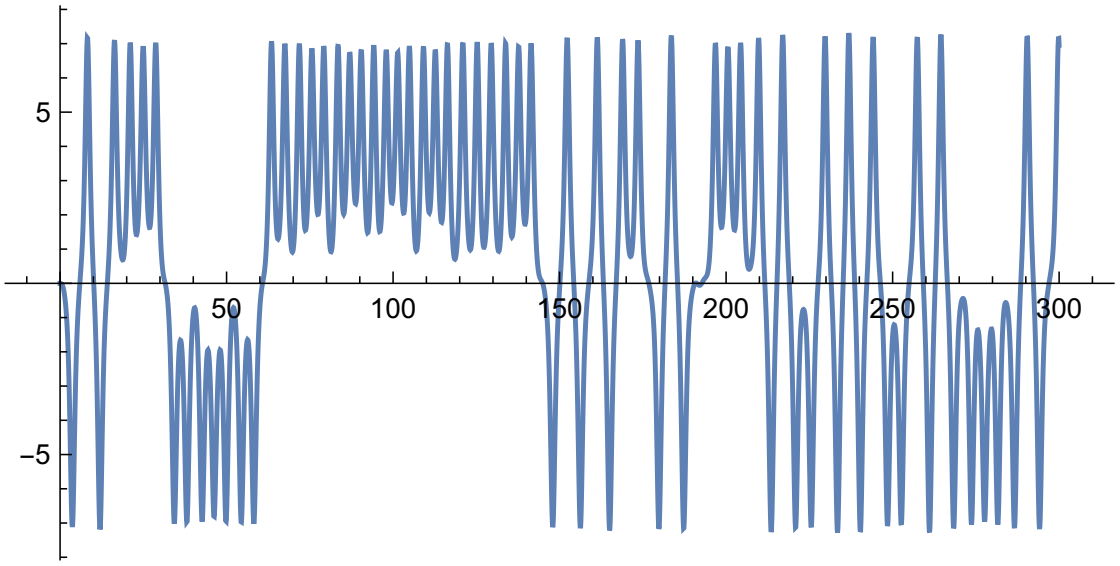

Figura 3: Gráfica x vs t en Mathematica para $dt = 0{,}0001$

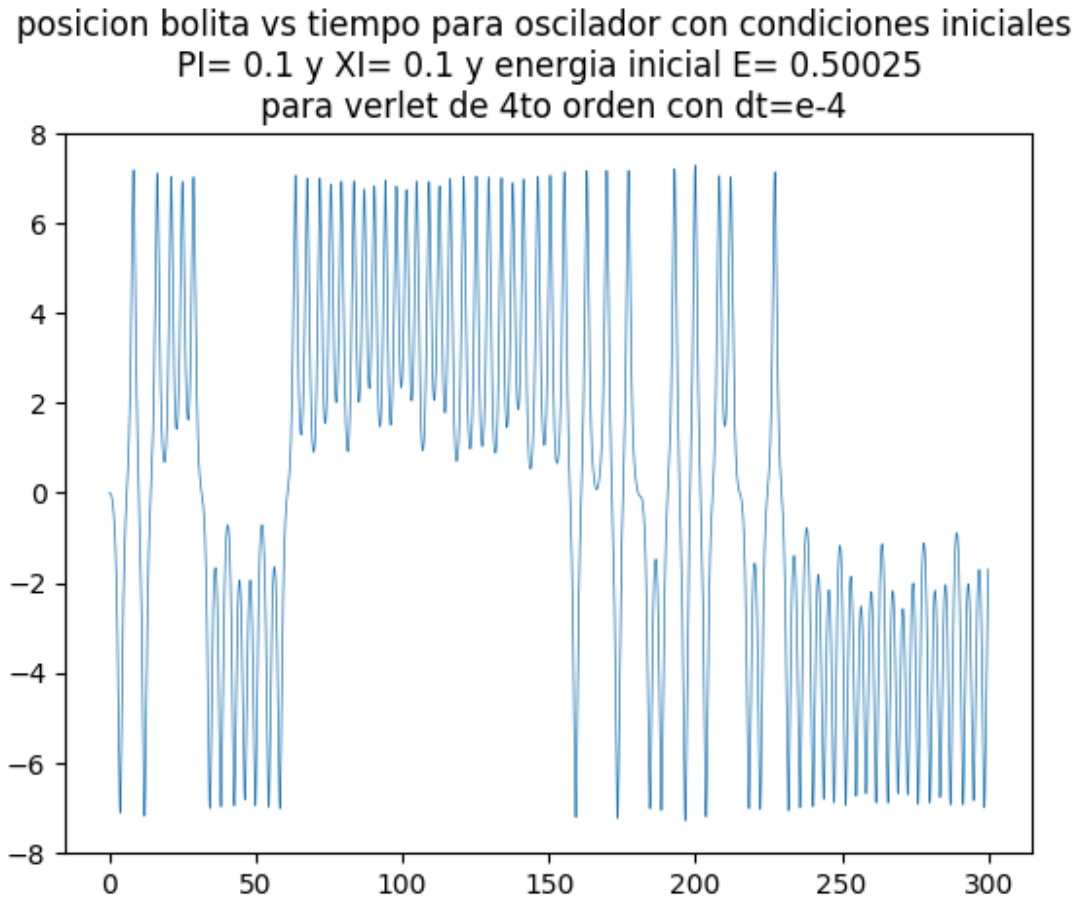

Figura 4: Gráfica x vs t en Python para $dt = 0{,}0001$

3. Resumen y lo que falta por hacer de hoy v2

La búsqueda de los integradores simplécticos que tengan buenos resultados para este sistema empieza a ser cada vez más complicada. Lo que debo hacer en la próxima sesión otra vez:

- Comprobar que los códigos propuestos funcionen, con casos de prueba
- Una vez que los códigos funcionen, crear un código para el sistema mio que llame las funciones de los integradores
- Cuando lo anterior sea hecho, agregar los códigos al repositorio de github
- Hablar con Óscar y el profesor para ver que opinan al respecto.

Lunes, 25 Febrero 2019

1. Secciones de poincaré para m=1a=0-25b=0-01M=0-1w=0-8g=0-1 y condiciones iniciales bolita quieta y oscilador P=X=0.1

Hice un gif usando 30 secciones de poincaré para el mismo problema, las secciones se crean a partir de tomar la frecuencia del oscilador ω aunque ahora que lo pienso, debí haber tomado el periodo del oscilador. Me pregunto si dará lo mismo.
Entonces esta primera parte será para la frecuencia, la otra parte será para el periodo:

- gif de sección de poincaré variando un poco el término de la frecuencia:

No pude agregar gifs a este archivo jajaja. Entonces me contento con haberle enviado al profesor los gifs de las secciones de poicaré para la frecuencia y periodo del oscilador.
Los otros objetivos siguen siendo los mismos:

- Comprobar que los códigos propuestos funcionen, con casos de prueba
- Una vez que los códigos funcionen, crear un código para el sistema mio que llame las funciones de los integradores
- Cuando lo anterior sea hecho, agregar los códigos al repositorio de github
- Hablar con Óscar y el profesor para ver que opinan al respecto.

Domingo, 17 Marzo 2019

1. Secciones de poincaré ya que las anteriores no sirvieron

Dado que las secciones de poincaré que intenté hacer anteriormente no funcionaron, entonces procedí a buscar bien cómo se hacían, en esto encontré un algoritmo que creo que funcionará:
Procedimiento computacional (Algoritmo para el problema de Henon-Heiles):
El procedimiento seguido en este trabajo para calcular las secciones de Poincaré en el sistema de Hénon-Heiles es el siguiente (se ha empleado el plano $x = 0$ por simplicidad, pero se podría emplear cualquier otro plano).

1. Se parten de unas condiciones iniciales tales que $x(0) = 0$, $y(0) = y_0$, $p_y(0) = p_{y0}$ y una energía $H = H_0$. A partir de estos valores, se calcula $p_x(0)$ a partir de la restricción $H = H_0$.

$$p_x(0) = \sqrt{2H_0 - p_{y0}^2 - y_0^2 + \frac{2}{3}y_0^3} \tag{0.16}$$

2. Se integra el sistema de ecuaciones diferenciales (4.6) con el integrador simpléctico de orden 4 empleado en todo el trabajo.

3. De la integración de las ecuaciones, el programa almacena los pares (y, p_y) que cumplen que $\epsilon < x < \epsilon$. Siendo ϵ un parámetro del programa que indica la precisión del mapa de Poincaré. Cuanto menor sea ϵ, menos puntos tendrá el mapa de Poincaré resultante.

4. Una vez se han integrado las ecuaciones y se han almacenado en un archivo todos los pares (y, p_y) para esta condición inicial, se vuelve al primer paso cambiando las condiciones iniciales con la misma energía y se repite todo el proceso.

5. Repitiendo este procedimiento varias veces se obtiene el mapa de Poincaré asociado al sistema dinámico (cuantas más veces repita, más rico será el mapa).

Con esto en mente, el proceso para el hamiltoniano de nuestro problema será usando la posición del primer oscilador en $X = 0$:

1. Se parten de unas condiciones iniciales tales que $X(0) = 0$, $p(0) = p_0$, $x(0) = x_0$ y una energía $H = H_0$. A partir de estos valores, se calcula $P(0)$ a partir de la restricción $H = H_0$.

$$P(0) = \sqrt{2M(H_0 - \frac{p_0^2}{2m} + a\frac{x_0^2}{2} - b\frac{x_0^4}{4})} \tag{0.17}$$

2. Se integra el sistema de ecuaciones diferenciales (4.6) con el integrador simpléctico de orden 4 empleado en todo el trabajo.

3. De la integración de las ecuaciones, el programa almacena los pares (y, p_y) que cumplen que $\epsilon < x < \epsilon$. Siendo ϵ un parámetro del programa que indica la precisión del mapa de Poincaré. Cuanto menor sea ϵ, menos puntos tendrá el mapa de Poincaré resultante.

4. Una vez se han integrado las ecuaciones y se han almacenado en un archivo todos los pares (y, p_y) para esta condición inicial, se vuelve al primer paso cambiando las condiciones iniciales con la misma energía y se repite todo el proceso.

5. Repitiendo este procedimiento varias veces se obtiene el mapa de Poincaré asociado al sistema dinámico (cuantas más veces repita, más rico será el mapa).

La manera como voy a proceder es haciendo una grilla de 25 puntos al rededor de la zona de equilibrio del oscilador y manteniendo una energía fija, en la primera prueba la energía impuesta es la resultante de las condiciones iniciales del oscilador dadas por $X = 0{,}1$ y $P = 0{,}1$.
Los otros objetivos siguen siendo los mismos:

- Comprobar que los códigos propuestos funcionen, con casos de prueba
- Una vez que los códigos funcionen, crear un código para el sistema mio que llame las funciones de los integradores
- Cuando lo anterior sea hecho, agregar los códigos al repositorio de github
- Hablar con Óscar y el profesor para ver que opinan al respecto.

Martes, 19 Marzo 2019

1. Comandos para conectarse por ssh

ssh spenam@168.176.92.190
la contraseña es el usuario: spenam
desde fuera del ssh copio los archivos que necesito por medio de:

```
scp route to file/file userid@168.176.92.190:Path where to save/
```

ejemplo de lo anterior:

```
scp ~/Desktop/tesis/poincare/compu_grupo/test_grupo.py spenam@168.176.92.190:
    poincare/
```

para traer archivos, en este caso fue una carpeta:

```
scp -r chaosg@168.176.92.190:poincare  ~/Desktop/tesis/poincare/
```

Tambien puedo conectarme al cluster de Delaware por medio de este comando:

```
ssh spenam@asterix-login.physics.udel.edu
```

La contrasena es la de todo.

Sabado, 13 Abril 2019

1. Parece que el integrador funciona pero a la ves no funciona

Definitivamente no entiendo esto de los integradores, puesto que decidí usar la base del que estaba generalizado, en donde solo tengo que insertar las constantes que definen el integrador. Decidí probar las secciones de Poincaré del potencial caotico de Henon-Heiles (El cual está más arriba) y usé un integrador de ruth de 3er orden. Luego de jugar con eso y buscar las aparentes condiciones iniciales que funcionaban, el resultado fueron gráficas bastante bonitas en donde creí que todo estaba empezando a dar:

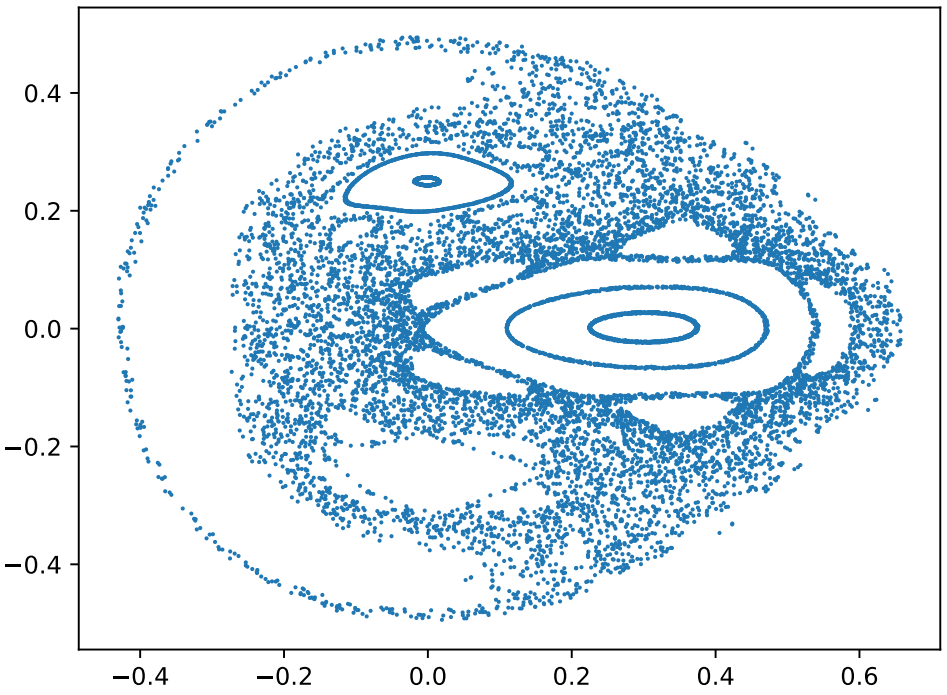

Figura 1: Secciones de Poincaré para el potencial de Henon-Heiles

Parecía que lo había logrado, este es el código que usé:

```
import math as m
import numpy as np
from matplotlib import pyplot as plt

def xpunto(vx0): #time derivative of x
    return vx0
def pxpunto(x0,y0,lambda0): #time derivative of px
    #print (x0,lambda0,y0)
    return -x0-(2.*lambda0*x0*y0)
def ypunto(vy0): #time derivative of y
    return vy0
def pypunto(x0,y0,lambda0): #time derivative of py
    return -y0-(lambda0*((x0*x0)-(y0*y0)))
def energia_poinc(y0,py0, H0): #initial condition for px using the
    constraints
    print (y0,py0,H0)
    print ((2.*H0)-(py0*py0)-(y0*y0)+((2./3.)*(y0*y0*y0)))
    return m.sqrt(abs((2.*H0)-(py0*py0)-(y0*y0)+((2./3.)*(y0*y0*y0))))



iteraciones=500000
dt=0.01
t=[0.]
x=[]
y=[]
px=[]
py=[]

def ruth3rd(xx,yy,ppx,ppy,lambda00):#define function with symplectic
    integrator

    aa=[1,-1./24.,3./4.,7./24.] #constants for position variable of the
    integrator
    bb=[1,1.,-2./3., 2./3.] #constants of velocity of the integrator

    x.append(xx)
    y.append(yy)
    px.append(ppx)
    py.append(ppy)



    xi , yi , pxi , pyi= xx, yy, ppx , ppy
    for i in range (iteraciones):
        #print (i*1.0*100.0/(iteraciones*1.0))
        vvx = pxi+bb[1]*pxpunto(xi,yi,lambda00)*dt
        VV1y = pyi+bb[1]*pypunto(xi,yi,lambda00)*dt
        xx = xi + aa[1]*vvx*dt
        yy =yi + aa[1]*VV1y*dt
        for j in range(2,len(aa)):
            vvx += bb[j]*pxpunto(xx,yy,lambda00)*dt
      VV1y += bb[j]*pxpunto(xx,yy,lambda00)*dt
            xx += aa[j]*vvx*dt
```

```
            yy += aa[j]*VV1y*dt

        xi = xx
        pxi = vvx
        yi = yy
        pyi = VV1y

        x.append(xi)
        px.append(pxi)
        y.append(yy)
        py.append(pyi)
        t.append(t[-1] + dt)

setsrange=4
sets=np.array(np.linspace(0., 4., 4),dtype=np.float64)
sets1=[1.,2.,3.,4.]

for I in range(1):

    xpoinc=[]
    ppoinc=[]

    for ii in range (setsrange):
        for jj in range (setsrange):

            Hinicial=1./8.
            x=[0.]
            y=[-0.15+0.3*(sets[jj]/(4.-1.))]
            py=[-0.15+0.3*(sets[ii]/(4.-1.))]
            #print(y[0],py[0],Hinicial)
            #print(energia_poinc(y[0],py[0],Hinicial))
            px=[energia_poinc(y[0],py[0],Hinicial)]

            ruth3rd(x[0],y[0],px[0],py[0],1.)


      crossings=[]
            crossingsindex=[]
            for idx, item in enumerate(x[:-1]): #intersections
                if x[idx] < 0 and x[idx+1] > 0:
                    crossings.append(x[idx])
                    crossingsindex.append(idx)


            for index in crossingsindex:
                xpoinc.append(y[index])
                ppoinc.append(py[index])

np.savetxt('xpoinc.out',xpoinc) #data of the poincare plot
np.savetxt('ppoinc.out',ppoinc)
```

Parecía que todo iba viento en popa, pero al intentar hacer lo mismo con un integrador de cuarto orden de ruth no daba lo mismo. No daba lo mismo!!!!!! A pesar de que solo cambié las constantes, el integrador fallaba y daba valores demasiado grandes, no

entiendo porqué:

```
import math as m
import numpy as np
from matplotlib import pyplot as plt

def xpunto(vx0): #time derivative of x
    return vx0
def pxpunto(x0,y0,lambda0): #time derivative of px
    print (x0,lambda0,y0)
    return -x0-(2.*lambda0*x0*y0)
def ypunto(vy0): #time derivative of y
    return vy0
def pypunto(x0,y0,lambda0): #time derivative of py
    return -y0-(lambda0*((x0*x0)-(y0*y0)))
def energia_poinc(y0,py0, H0): #initial condition for px using the
    constraints
    print (y0,py0,H0)
    print ((2.*H0)-(py0*py0)-(y0*y0)+((2./3.)*(y0*y0*y0)))
    return m.sqrt(abs((2.*H0)-(py0*py0)-(y0*y0)+((2./3.)*(y0*y0*y0))))



iteraciones=2000
dt=0.01
t=[0.]
x=[]
y=[]
px=[]
py=[]

def ruth3rd(xx,yy,ppx,ppy,lambda00):#define function with symplectic
    integrator

    dos=np.math.pow(2.0, 1.0/3.0)
    aa=[1,0.5/ (2.0 - dos), 0.5*(1.0-dos)/ (2.0 - dos), 0.5*(1.0-dos)/ (2.0 -
     dos), 0.5/ (2.0 - dos)] #constants for position variable of the
    integrator
    bb=[1,1./ (2.0 - dos),-dos/ (2.0 - dos),1./ (2.0 - dos),0.] #constants of
     velocity of the integrator

    x.append(xx)
    y.append(yy)
    px.append(ppx)
    py.append(ppy)



    xi , yi , pxi , pyi= xx, yy, ppx , ppy
    for i in range (iteraciones):
        #print (i*1.0*100.0/(iteraciones*1.0))
        vvx = pxi+bb[1]*pxpunto(xi,yi,lambda00)*dt
        VV1y = pyi+bb[1]*pypunto(xi,yi,lambda00)*dt
```

```
        xx = xi + aa[1]*vvx*dt
        yy =yi + aa[1]*VV1y*dt
        for j in range(2,len(aa)):
            vvx += bb[j]*pxpunto(xx,yy,lambda00)*dt
      VV1y += bb[j]*pxpunto(xx,yy,lambda00)*dt
            xx += aa[j]*vvx*dt
            yy += aa[j]*VV1y*dt

        xi = xx
        pxi = vvx
        yi = yy
        pyi = VV1y

        x.append(xi)
        px.append(pxi)
        y.append(yy)
        py.append(pyi)
        t.append(t[-1] + dt)

setsrange=2
sets=np.array(np.linspace(0., 4., 4),dtype=np.float64)
sets1=[1.,2.,3.,4.]

for I in range(1):

    xpoinc=[]
    ppoinc=[]

    for ii in range (setsrange):
        for jj in range (setsrange):

            Hinicial=1./8.
            x=[0.]
            y=[-0.15+0.3*(sets[jj]/(4.-1.))]
            py=[-0.15+0.3*(sets[ii]/(4.-1.))]
            #print(y[0],py[0],Hinicial)
            #print(energia_poinc(y[0],py[0],Hinicial))
            px=[energia_poinc(y[0],py[0],Hinicial)]

            ruth3rd(x[0],y[0],px[0],py[0],1.)


      crossings=[]
            crossingsindex=[]
            for idx, item in enumerate(x[:-1]): #intersections
                if x[idx] < 0 and x[idx+1] > 0:
                    crossings.append(x[idx])
                    crossingsindex.append(idx)


            for index in crossingsindex:
                xpoinc.append(y[index])
                ppoinc.append(py[index])

```

Sabado, 13 Abril 2019

```python
np.savetxt('xpoinc4th.out',xpoinc) #data of the poincare plot
np.savetxt('ppoinc4th.out',ppoinc)
```

Hice también varias pruebas numéricas de los integradores para el caso del péndulo y muestra que conserva bien el area, entonces no se que está pasando.
Por fin encontré el error, estaba en el algoritmo del integrador, estaba llamando pxpunto en el paso de evolución de VV1y!!

Viernes, 19 Abril 2019

1. Algoritmo terminado para los integradores

Luego de que por fin tuviera bien aparentemente los integradores los probé con el sistema y obtuve unos resultados aparentemente satisfactorios, en la carpeta de poincare que se llama pruba integrador hice los codigos llamados 3rdsistema y 4th sistema donde supuestamente los tengo en el modo del algoritmo y aparentemente funciona bastante bien. Ahora voy a estudiar los papers que me mandó Thomas donde implementan unos métodos de integradores simplecticos sin constantes negativas donde argumentan que las constantes negativas dan inestabilidad en la integracion. Algo interesante es que probé esto con el caso que tiene una frecuencia bastante alta y la grafica no coincidio con ninguna que habia hecho anteriormente lo cual me genera bastante curiosidad.

Lunes, 29 Abril 2019

1. Papers de nuevos integradores simplecticos que thomas me paso

En los papers que thomas me mando usan unos integradores simplecticos de segundo orden llamados $SABA_2$ y $SBAB_2$, estos algoritmos pueden ser generalizados para cualquier orden en Hamiltonianos de la forma:

$$H = A + \epsilon B \tag{0.18}$$

Dadas las ecuaciones de Hamilton definidas por:

$$\frac{dp_n}{dt} = -\frac{\partial H}{\partial q_n},\ \frac{dq_n}{dt} = \frac{\partial H}{\partial p_n},\ n = 1, 2, ...N \tag{0.19}$$

Tambien recordamos los corchetes de poisson para funciones $f(p,q)$ y $g(p,q)$ los cuales son dados por:

$$\{f, g\} = \sum_{n=1}^{n} \left(\frac{\partial f}{\partial p_n}\frac{\partial g}{\partial q_n} - \frac{\partial f}{\partial q_n}\frac{\partial g}{\partial p_n} \right) \tag{0.20}$$

Entonces las ecuaciones de Hamilton toman la forma de:

$$\frac{dx}{dt} = \{H, x\} = L_H x \tag{0.21}$$

Donde L_H es el operador diferencial definido por $L_\chi f = \{\chi, f\}$. La solucion a la ecuacion anterior para condiciones iniciales $x(0) = x_0$ esta dada por:

$$x(t) = \sum_{n\geq 0} \frac{t^n}{n!} L_H^n x_0 = e^{tL_H} x_0 \tag{0.22}$$

Entonces los integradores quedarían:

$$SABA_2 = e^{c_1 \tau L_A} e^{d_1 \tau L_{\epsilon B}} e^{c_2 \tau L_A} e^{d_1 \tau L_{\epsilon B}} e^{c_1 \tau L_A} \tag{0.23}$$

con $c_1 = \frac{1}{2}(1 - \frac{1}{\sqrt{3}})$, $c_2 = \frac{1}{\sqrt{3}}$ y $d_2 = \frac{1}{2}$, mientras que el otro integrador queda:

$$SBAB_2 = e^{d_1 \tau L_{\epsilon B}} e^{c_2 \tau L_A} e^{d_2 \tau L_{\epsilon B}} e^{c_2 \tau L_A} e^{d_1 \tau L_{\epsilon B}} \tag{0.24}$$

con $c_2 = \frac{1}{2}$, $d_1 = \frac{1}{6}$ y $d_2 = \frac{2}{3}$. Al usar estos integradores estamos aproximando el comportamiento dinamico del Hamiltoniano real $H = A + \epsilon B$ por el Hamiltoniano $\widetilde{H} = A + \epsilon B + \mathcal{O}(\tau^4\epsilon + \tau^2\epsilon^2)$.

La precision de los integradores puede ser mejorada al introducir el termino $C = \{\{A, B\}, B\}$ lo cual lleva a un sistema integrable. Esto agrega unos pequenos pasos que pueden ser agregados al integrador:

$$SABA_2C = e^{-(\tau^3\epsilon^2 g/2)L_C}(SABA_2)e^{-(\tau^3\epsilon^2 g/2)L_C} \tag{0.25}$$

Una expresion similar puede ser derivada para $SBAB_2$ donde el valor de g se escoge de tal manera que elimine la dependencia de $\tau^2\epsilon^2$ asi el integrador queda con el orden de $\mathcal{O}(\tau^4\epsilon + \tau^4\epsilon^2$. En particular tenemos que $g = (2-\sqrt{3})/24$ para $SABA_2$ y $g = 1/72$ para $SBAB_2$. Se enfatiza que estos integradores solo involucran pasos hacia adelante lo cual incrementa la estabilidad numerica.
Para nuestro sistema es necesario entonces calcular los corchetes de poisson de estos casos, esto tiene como resultado:

$$A = \frac{p^2}{2m} + \sum_{n=1}^{N} \frac{P_n^2}{2M_n} \tag{0.26}$$

$$B = -a\frac{x^2}{2} + b\frac{x^4}{4} + \sum_{n=1}^{N} M_n\omega_n^2\frac{X_n^2}{2} + x\sum_{n=1}^{N} g_n X_n \tag{0.27}$$

$$L_A x = \frac{p}{m} \tag{0.28}$$

$$L_A X_n = \frac{P_n}{M_n} \tag{0.29}$$

$$L_A p = L_A P = 0 \tag{0.30}$$

$$L_B x = L_B X = 0 \tag{0.31}$$

$$L_B p = ax - bx^3 - \sum_{n=1}^{N} g_n X_n \tag{0.32}$$

$$L_B P_n = -M_n\omega_n^2 X_n - x g_n \tag{0.33}$$

Con esto, obtengo que:

$$\{\{A, B\}, B\} = \frac{\big(-ax + bx^3 + \sum_{n=1}^{N} g_n X_n\big)^2}{m} + \sum_{n=1}^{N} \frac{(x g_n + M_n\omega_n^2 X_n)^2}{M_n} \tag{0.34}$$

Por lo tanto:

$$L_C p = \tfrac{-2}{m}(-ax + bx^3 + \textstyle\sum_{n=1}^{N} g_n X_n)(-a + 3bx^2) - \sum_{n=1}^{N} \tfrac{2}{M_n}(x g_n + M_n\omega_n^2 X_n)(g_n) =$$
$$= \tfrac{-2}{m}(a^2 x - abx^3 - a\textstyle\sum_{n=1}^{N} g_n X_n - 3abx^3 + 3b^2x^5 + 3bx^2\sum_{n=1}^{N} g_n X_n) - \sum_{n=1}^{N} \tfrac{2}{M_n}(x g_n^2 + M_n g_n\omega_n^2 X_n)$$

(0.35)

$$\begin{aligned} L_C P = \tfrac{-2}{m}(-ax + bx^3 + \textstyle\sum_{n'=1}^{N} g_{n'} X_{n'})(g_n) - 2(x g_n + M_n \omega_n^2 X_n)\omega_n^2 = \\ = \tfrac{-2}{m}(-a x g_n + b x^3 g_n + \textstyle\sum_{n'=1}^{N} g_{n'} X_{n'} g_n) - 2 x g_n \omega_n^2 - 2 M_n \omega_n^4 X_n \end{aligned} \tag{0.36}$$

Entonces tengo como resultado las evoluciones de la manera siguiente:

$$e^{\tau L_A} : \begin{cases} x' = \frac{p}{m}\tau + x \\ p' = p \\ X_n' = \frac{P_n}{M_n}\tau + X_n \\ P_n' = P_n \end{cases} \tag{0.37}$$

$$e^{\tau L_B} : \begin{cases} x' = x \\ p' = (ax - bx^3 - \sum_{n=1}^{N} g_n X_n)\tau + p \\ X_n' = X_n \\ P_n' = (-M_n \omega_n^2 X_n - x g_n)\tau + P_n \end{cases} \tag{0.38}$$

$$e^{-\tau L_C} : \begin{cases} x' = x \\ p' = \big(\frac{-2}{m}(a^2 x - abx^3 - a\sum_{n=1}^{N} g_n X_n - 3abx^3 + 3b^2 x^5 + 3bx^2 \sum_{n=1}^{N} g_n X_n) \\ - \sum_{n=1}^{N} \frac{2}{M_n}(x g_n^2 + M_n g_n \omega_n^2 X_n)\big)\tau + p \\ X_n' = X_n \\ P_n' = \big(\frac{-2}{m}(-a x g_n + b x^3 g_n + \sum_{n'=1}^{N} g_{n'} g_n X_n) - (2 x g_n \omega_n^2 + 2 M_n \omega_n^4 X_n)\big)\tau + P_n \end{cases} \tag{0.39}$$

Lunes, 10 Marzo 2020

1. La era de Julia

Mudé los integradores a Julia, en este lenguaje los pude generalizar facilmente y se me es mucho más fácil agregar osciladores.

2. Lo que he hecho en Julia

With Julia I have generalized the integrators using the OrdinaryDiffEq.jl package, this package has in its documentation symplectic integrators of different orders including the ones I was using before. This package is very suitable for what I am doing because I can easily include more modes into the bath without having to rewrite whole blocks of code.

3. Ensemble evolution and mutual information using BoxMuller

For only one oscillator. In order to track the evolution of ensembles of trajectories I used the Box-Muller transform [**golderBoxMullerMethodGenerating1976**] to generate gaussian distributions in the phase space of the oscillator and the system. Using this I solved the differential equations and evolved the ensembles on different initial positions for the oscillator phase space with the parameters:

$$a = 0{,}25 \tag{0.40}$$
$$b = 0{,}01 \tag{0.41}$$
$$time = 100. \tag{0.42}$$
$$dts = 0{,}001 * 2.\pi/w[2]w[2]_r effers_t o_f requency_o f_o scillator \tag{0.43}$$
$$m[1] = 1.Particle_m ass \tag{0.44}$$
$$m[2] = 0{,}1Oscilator_m ass \tag{0.45}$$
$$g[2] = 0{,}05Oscillator_c oupling_c onstant \tag{0.46}$$
$$w[2] = 0{,}7071Oscillator_f requency \tag{0.47}$$
$$\tag{0.48}$$

Todos estos ensayos se pueden hacer en el notebook ïnformacion". En ese mismo notebook hay unas subrutinas que calculan la entropia All of these test can be reproduced again on the notebook ïnformacion". On the same notebook there are some subroutines that

calculate the mutual information every 10 dts of the ensamble and plots it in order to obtain the dynamical evolution of the mutual information in the ensemble. This is done using the Julia package ÇhaosTools.jl"which contains a function that calculates the Information of an ensemble of points efficiently.

4. Attractors of initial side for the particle

Now that I evolved the ensembles of trajectories I plotted the attractors for different initial conditions for the oscillator by making a grid on the phase space of the oscillator and the particle on the unstable equilibrium point. If the particle falls to the right well I gave the color red (1 in number) to the plot and if it falls on the left well I gave the color white (0 in number). I used two types of integrators for the same conditions of the system mentioned above (VerletLeapfrog and CalvoSanz4), the attractors where the same for both integrators. I varied the coupling constat g to check for changes in the attractors, this is showed in the figures below:

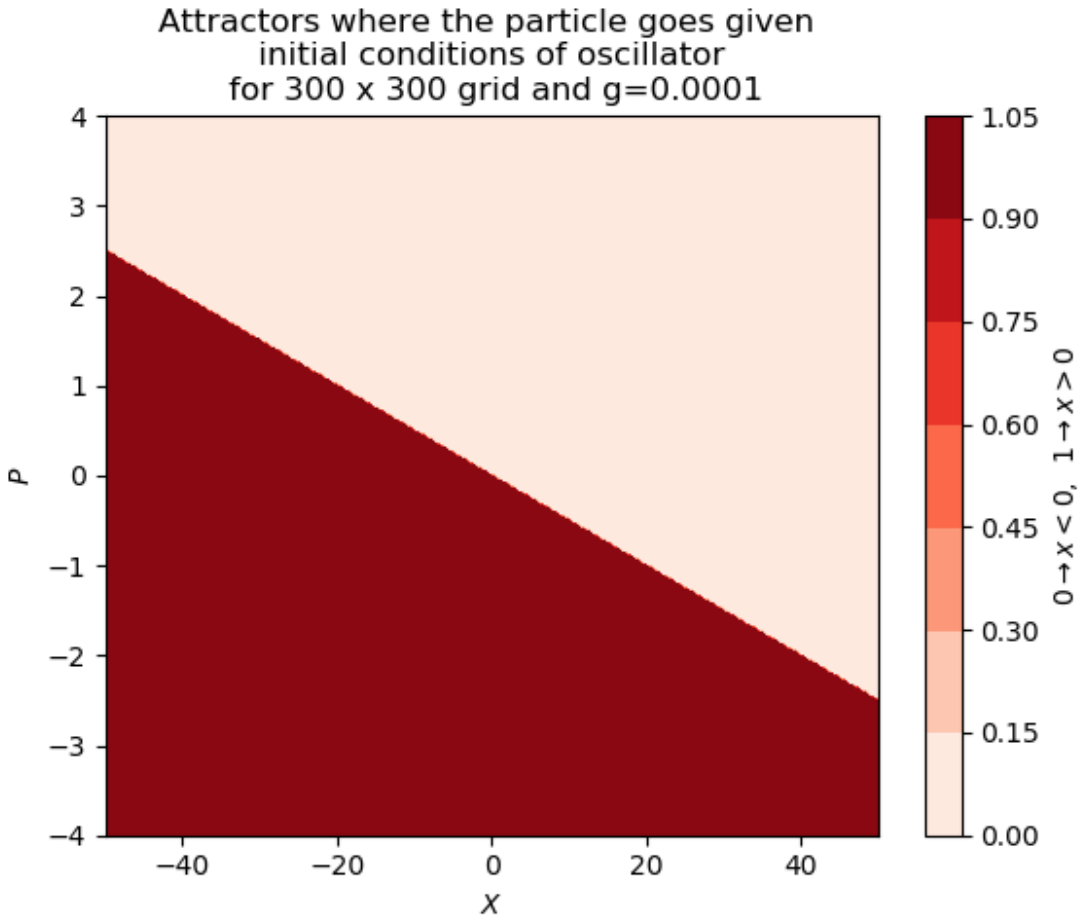

Figura 1: Attractor for g=0.0001

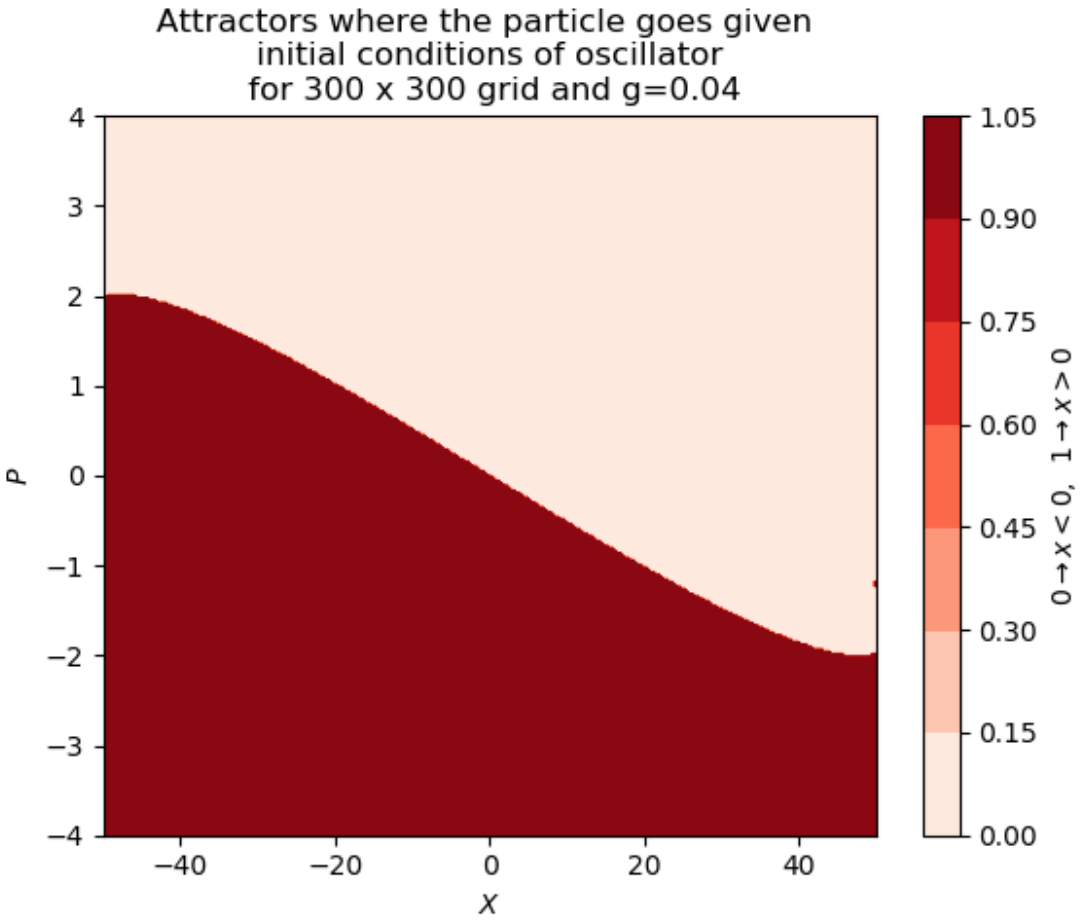

Figura 2: Attractor for g=0.04

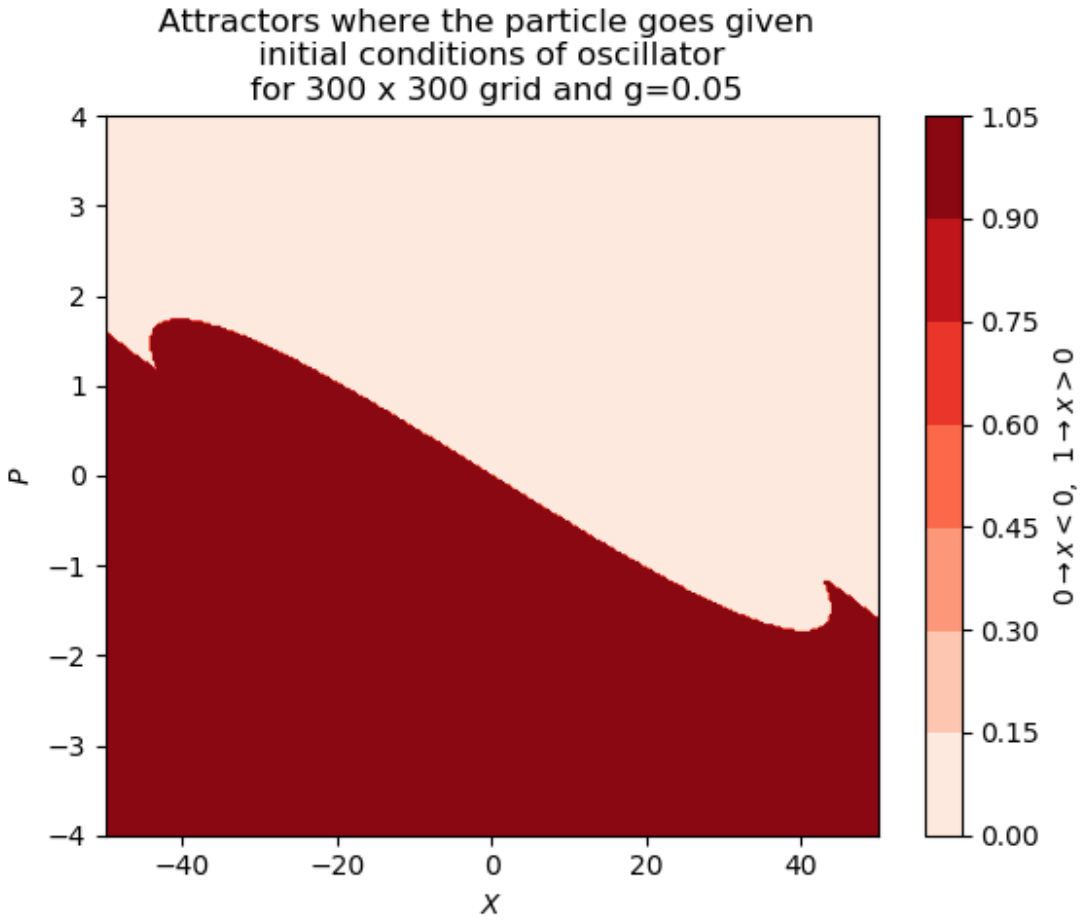

Figura 3: Attractor for g=0.05

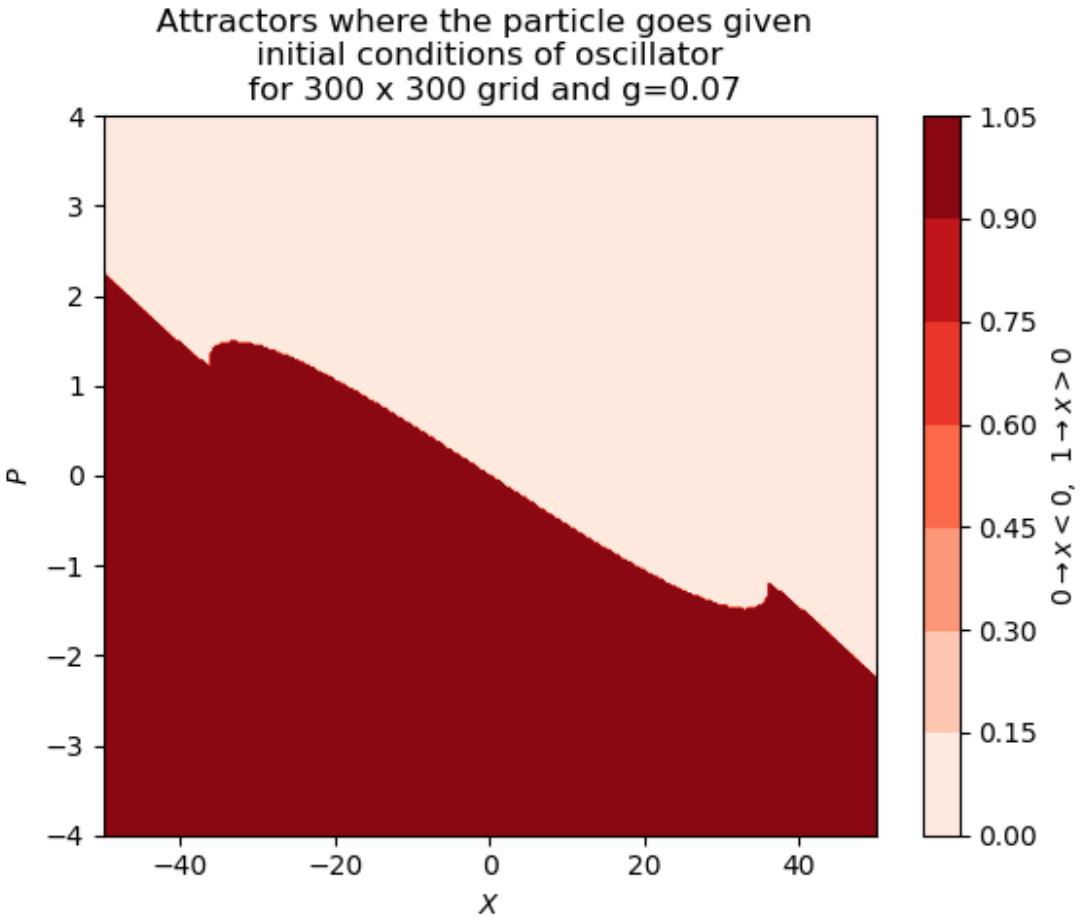

Figura 4: Attractor for g=0.07

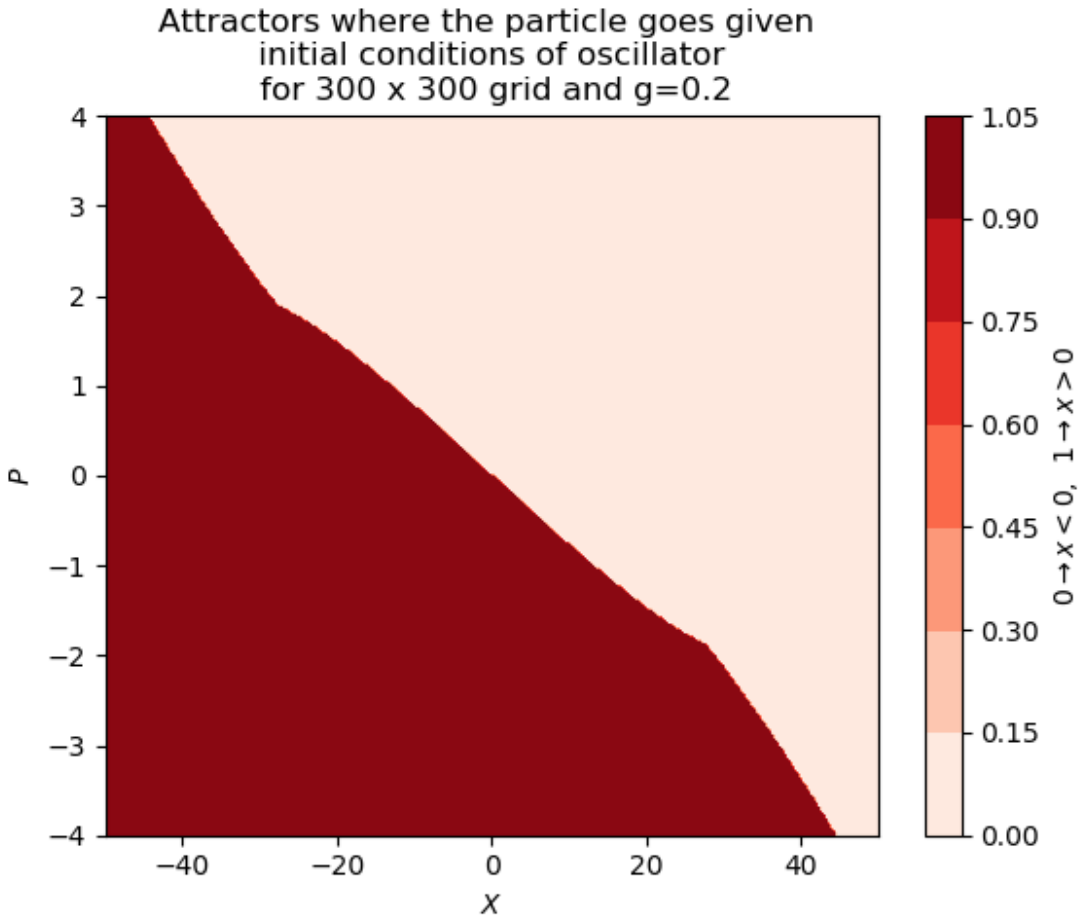

Figura 5: Attractor for g=0.2

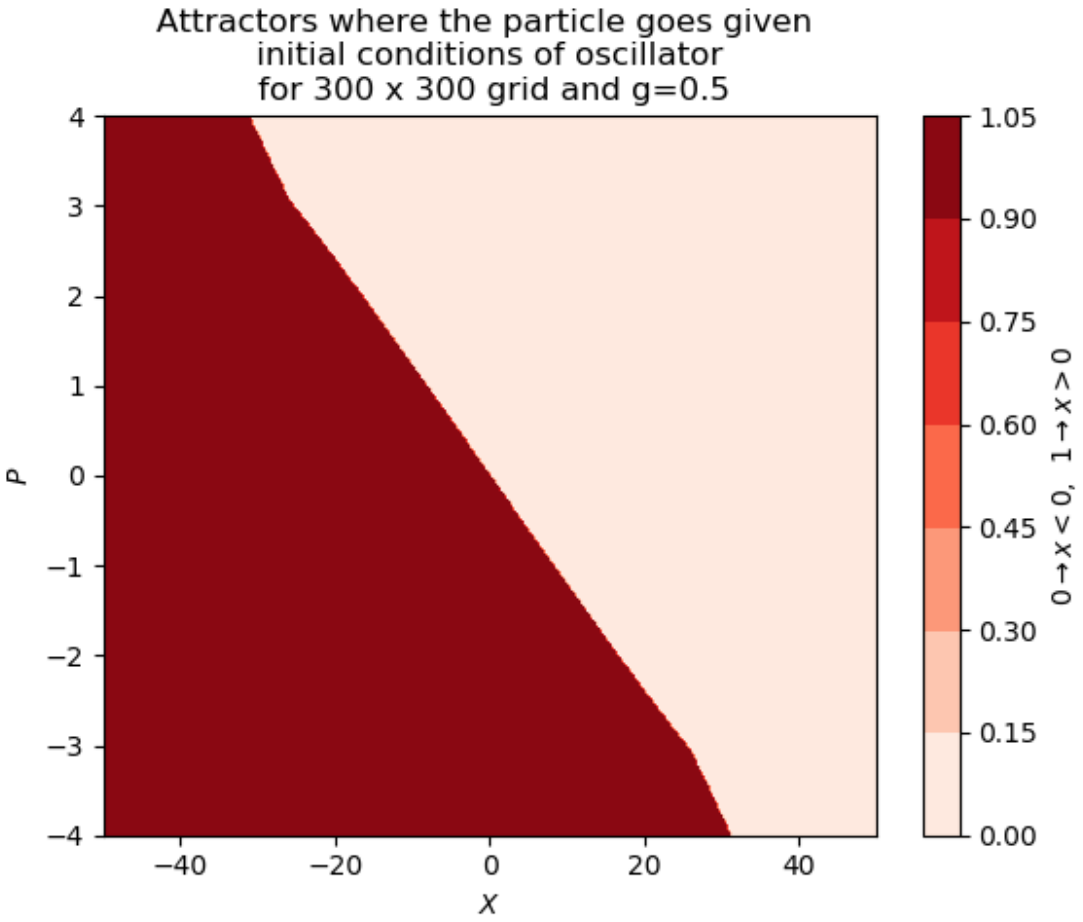

Figura 6: Attractor for g=0.5

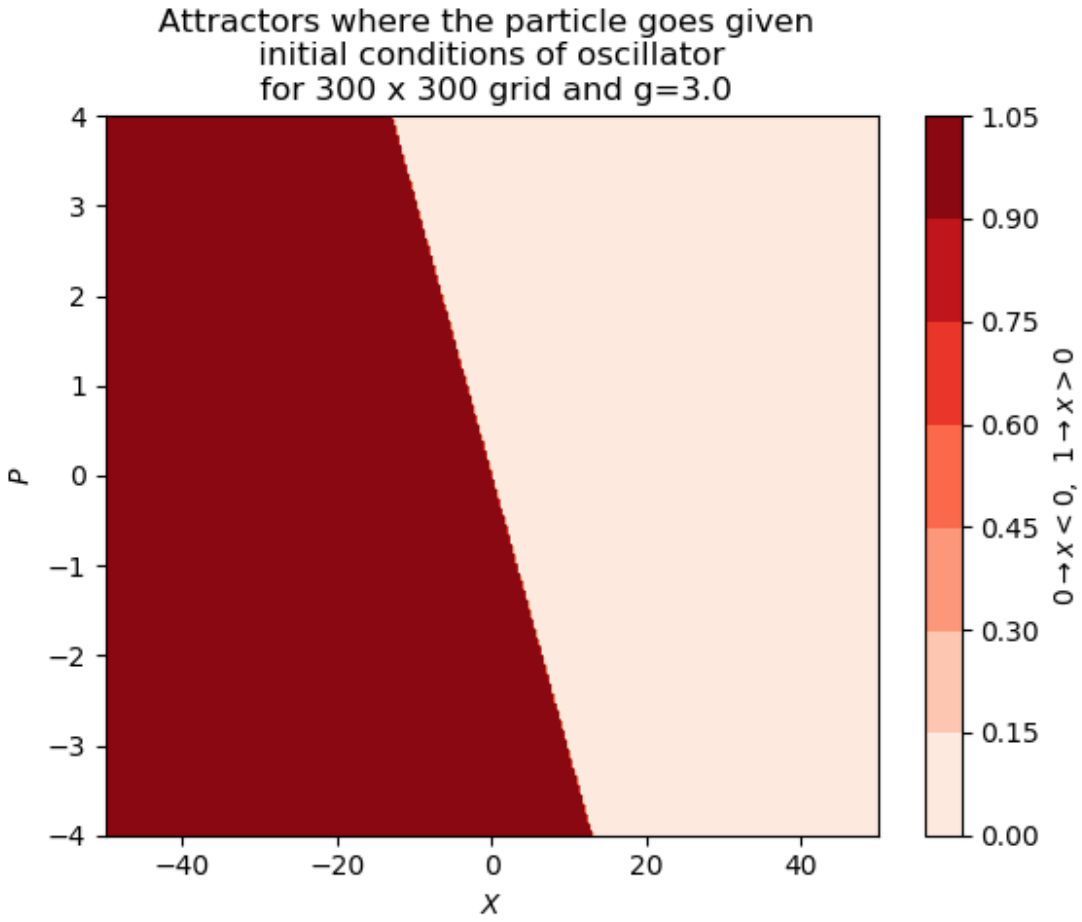

Figura 7: Attractor for g=3.0

It can be seen some weird structures of spikes for $g = 0{,}05$ and greater as seen in fig 3 and above. I made a little zoom of this structure for the case of $g = 0{,}05$ and the result is show below:

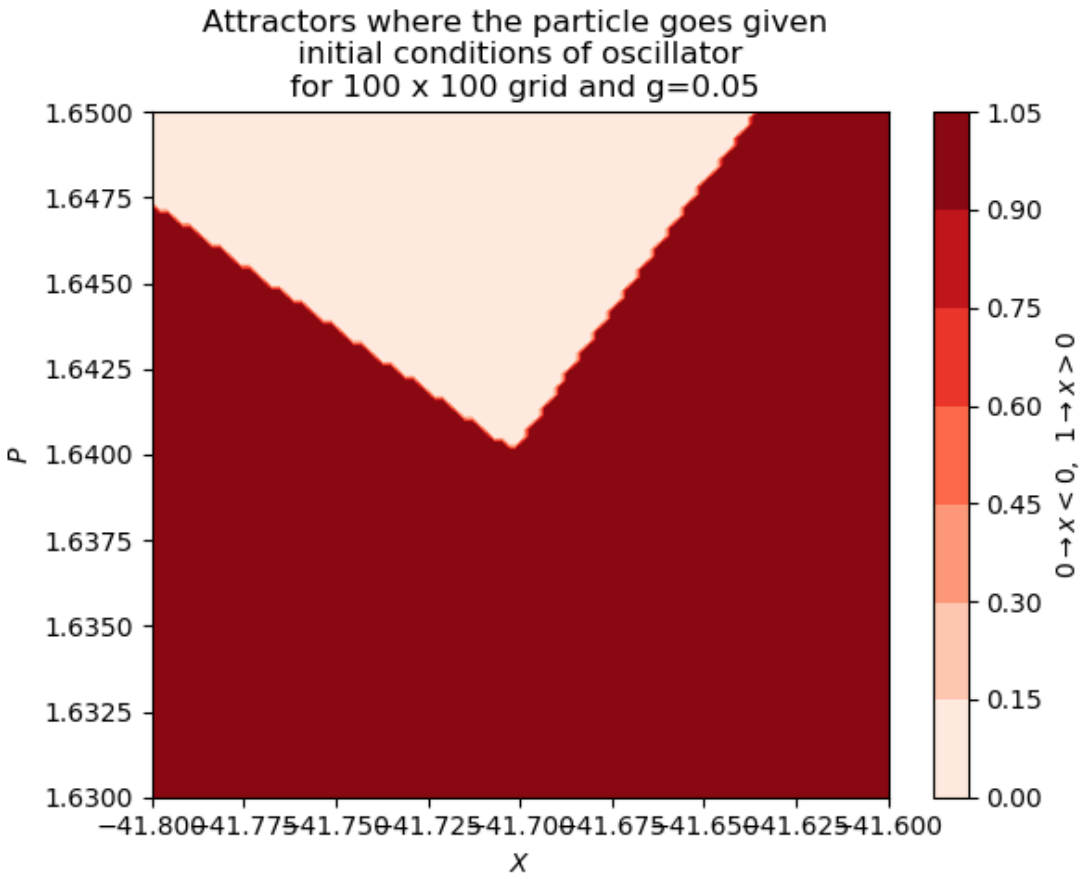

Figura 8: Weird attractor zone for g=0.05

This structure seems to have some physics into into it, but I will not come back until later.

I will evolve the ensemble on one of these spikes for the case of g=0.07 and check the result.

5. This is yet another example experiment

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna.

Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetuer.

Friday, 26 March 2010

1. This shows a sample table

Groups	Treatment X	Treatment Y
1	0.2	0.8
2	0.17	0.7
3	0.24	0.75
4	0.68	0.3

Cuadro 1: The effects of treatments X and Y on the four groups studied.

Table 1 shows that groups 1-3 reacted similarly to the two treatments but group 4 showed a reversed reaction.

Saturday, 27 March 2010

1. Bulleted list example

This is a bulleted list:

- Item 1
- Item 2
- ...and so on

2. This is an example experiment

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

3. This is another example experiment

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.

Formulae and Media Recipes

Media

Media 1

Compound	1L	0.5L
Compound 1	10g	5g
Compound 2	20g	10g

Cuadro 1: Ingredients in Media 1.

Formulae

Formula 1 - Pythagorean theorem

$a^2 + b^2 = c^2$ [**diaconisDynamicalBiasCoin2007**]

bib.bib