Simulación numérica del péndulo de Foucault con Octave

Marco Reyes, Nestor Pineda*

Universidad Nacional Autónoma de Honduras en el Valle de Sula

Resumen

A review of the Foucault pendulum is performed, considering the numerical approach, the study of many problems in classical mechanics can focus naturally using computational tools, and the problem of Foucault pendulum is an example of this. The Foucault pendulum is a simple pendulum basically set in a non-inertial reference frame. By taking into account the rotation of the Earth the pendulum makes a slight movement of precession, we can predict the movement to formulate and solve the dynamic equations associated with the pendulum, these equations are derived from Newton's laws considering the pseudo force arises by including the rotation of the Earth. To solve the coupled differential equations of the Foucault pendulum, we used the algorithms of Euler-Cromer , Runge Kutta second order Runge Kutta fourth order, all these results are similar algorithms are used to plot the trajectory of the pendulum in polar coordinates , closed paths precess features are observed precession motion.

Keywords: Pendulum, rotation, precession.

Se realiza una revisión del péndulo de Foucault, considerando la programación numérica, el estudio de muchos problemas que surgen en la mecánica clásica se pueden enfocar de forma natural utilizando herramientas computacionales, y el problema del péndulo de Foucault es un ejemplo de esto. El péndulo de Foucault es básicamente un péndulo simple considerando que esta ubicado en un marco de referencia no inercial. Al tomar en cuenta la rotación de la tierra, el péndulo realiza un movimiento leve de precesión, este movimiento lo podemos predecir al formular y resolver las ecuaciones dinámicas asociadas al péndulo, se deducen estas ecuaciones a partir de la segunda ley de Newton considerando la pseudo fuerza que surge al incluir la rotación de Tierra. Para resolver las ecuaciones diferenciales acopladas del péndulo de Foucault, se utilizan los algoritmos de Euler-Cromer, Runge Kutta de segundo orden y Runge Kutta de orden cuatro, con todos estos algoritmos los resultados son similares, al graficar la trayectoria del péndulo en coordenadas polares, se observan trayectorias cerradas características del movimiento de precesión.

Palabras claves: Péndulo, rotación de la tierra, precesión.

I. Introducción

Hoy en día, la enseñanza de la física esta muy ligada al uso de herramientas computacionales, especialmente en los cursos de nivel intermedio y avanzado surgen problemas que no tienen soluciones analíticas, los cuales requieren hacer uso de algoritmos numéricos, para tener una solución aproximada. Estos problemas pueden ser programados en diversos programas, como Fortran, C, C++, Java, Python y Octave, todos son software libre y es cuestión de gustos cual seleccionar, en este trabajo utilizamos Octave.

Los algoritmos que escogimos para integrar las ecuaciones diferenciales son: Euler-Cromer (EC), Runge-Kutta de segundo orden (RK2)y Runge Kutta de orden 4 (RK4). La eficiencia de estos algoritmos va en orden creciente.

II. El péndulo de Foucault

Este es un péndulo hecho de una masa muy pesada suspendida de un alambre muy largo de un techo alto, esto le permite al péndulo oscilar de norte a sur y de este a oeste. Visto desde un marco inercial, solo actúan dos fuerzas sobre el péndulo, el peso y la tensión de la cuerda. En el marco rotatorio de la tierra, están además la fuerza centrifuga y la fuerza de Coriolis, así la ecuación de movimiento queda

$$m\frac{d^2\vec{r}}{dt^2} = \vec{T} + m\vec{g_0} + m\left(\vec{\omega}\times\vec{r}\right)\times\vec{\omega} + 2m\frac{d\vec{r}}{dt}\times\vec{\omega} \tag{1}$$

El segundo y tercer término combinado en el lado derecho de la ecuación 1, nos da $m\vec{g}$, donde $\vec{g}$, es la aceleración de caída libre observada, la ecuación de movimiento se convierte

$$m\frac{d^2\vec{r}}{dt^2} = \vec{T} + m\vec{g} + 2m\frac{d\vec{r}}{dt}\times\vec{\omega} \tag{2}$$

Escogemos nuestros ejes, como se muestra en la figura 1 y restringimos nuestra discusión al caso de oscilaciones pequeñas, así el ángulo α entre el péndulo y la vertical siempre es pequeño [4], [5], [6]. Esto nos permite hacer dos aproximaciones, primero la componente z de la tensión es bien aproximada por su magnitud $T_z = Tcos\alpha \approx T$, segundo para oscilaciones pequeñas $T_z = mg$, poniendo estas dos aproximaciones juntas, podemos escribir

$$T \approx mg \tag{3}$$

De la figura 1, se deduce que las componentes de la tensión en el plano xy son

$$T_x = -T\frac{x}{L}, \qquad T_y = -T\frac{y}{L}, \qquad T_z \approx T$$

La velocidad angular en función de la latitud se puede escribir así

$$\vec{\omega} = -\hat{x}\left(cos\theta\right) + \hat{z}\left(\omega sen\theta\right) \tag{4}$$

y las componentes de velocidad se pueden escribir en la notación conveniente de punto

$$v_x = \dot{x} \qquad v_y = \dot{y}, \qquad v_z = 0$$

al realizar el producto vectorial $\vec{\omega}\times\vec{v}$ se obtienen las siguientes componentes

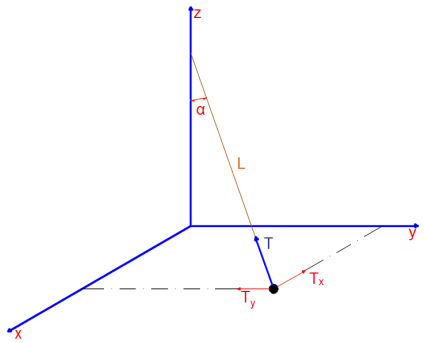

Figura 1: *Sistema coordenado utilizado.*

$$\left(\vec{\omega}\times\vec{v}\right)_x = -\dot{y}\ \omega\ sen\theta$$

$$\left(\vec{\omega}\times\vec{v}\right)_y = \dot{x}\ \omega\ sen\theta$$

$$\left(\vec{\omega}\times\vec{v}\right)_z = -\dot{y}\ \omega\ cos\theta$$

Al reemplazar lo anterior, se llega a las siguientes ecuaciones

$$a_x = \ddot{x} = -\frac{T}{m}\frac{x}{L} + 2\ \dot{y}\ \omega\ sen\theta \tag{5}$$

$$a_y = \ddot{y} = -\frac{T}{m}\frac{y}{L} - 2\ \dot{x}\ \omega\ sen\theta \tag{6}$$

para pequeñas oscilaciones recordemos que $T \approx mg$ con lo que finalmente obtenemos

$$\ddot{x} = -\frac{g}{L}x + 2\ \dot{y}\ \omega\ sen\theta \tag{7}$$

$$\ddot{y} = -\frac{g}{L}y - 2\ \dot{x}\ \omega\ sen\theta \tag{8}$$

Las ecuaciones 7 y 8 son ecuaciones diferenciales acopladas, al integrar numéricamente estas ecuaciones vamos a obtener la trayectoria del péndulo en el plano xy.

III. Algoritmos

Vamos a describir los algoritmos utilizados en este trabajo.

I. Euler Cromer

Tenemos un problema con valores iniciales, que no se puede resolver de forma analítica, en consecuencia, es necesario utilizar los métodos numéricos [3].
Los métodos de Euler no se suelen utilizar en

la práctica debido a que la solución que proporciona acumula errores apreciables a lo largo del proceso; sin embargo es un buen punto de partida por su sencillez.
El algoritmo es el siguiente

Condiciones iniciales:

$x\left(0\right)=x_0 \qquad y\left(0\right)y_0 \qquad v_x\left(0\right)=v_0$

Calculo del paso: intervalo de tiempo $\left[0,t_f\right]$
$h=\left(t_f-0\right)/M$

Ciclo

para $n=0 \;\; a \;\; M$

$$v_x\left(n+1\right)=v_x\left(n\right)+ha_x\left(n\right) \tag{9}$$

$$v_y\left(n+1\right)=v_y\left(n\right)+ha_y\left(n\right) \tag{10}$$

$$x\left(n+1\right)=x\left(n\right)+hv_x\left(n+1\right) \tag{11}$$

$$y\left(n+1\right)=y\left(n\right)+hv_y\left(n+1\right) \tag{12}$$

II. Script de EC en Octave

A continuación se muestra la implementación en Octave.

```
clear;
npts=5000;
l=200;
g=9.80;
w=0.00007272205;
angulo=50;
x0=15; y0=15; vx0=0; vy0=0;
dt=300/npts;
nt=npts/10;
fprintf('Paso de
tiempo usado dt=tmax/npts=
%7.4f\n',dt);
ay0=-g*(y0/l); ax0=-g*(x0/l);
printf('t          X
Y  \n')
t(1)=0;
x(1)=x0;
y(1)=y0;
vx(1)=vx0;
vy(1)=vy0;
ax(1)=ax0;
ay(1)=ay0;
fprintf('%3.1f   %7.2f
%11.2f \n',
t(1),x(1),y(1));

for i=1:npts
  vx(i+1)=vx(i)+ax(i)*dt;
  vy(i+1)=vy(i)+ay(i)*dt;
  x(i+1)=x(i)+vx(i+1)*dt;
  y(i+1)=y(i)+vy(i+1)*dt;
  t(i+1)=t(i)+dt;
  ax(i+1)=-(g/l)*x(i+1)+
  2*w*sind(angulo)*vy(i+1);
  ay(i+1)=-(g/l)*y(i+1)
  -2*w*sind(angulo)*vx(i+1);
  if(mod(i,nt)==0)
    fprintf('%3.1f
    %6.2f    %9.2f \n',t(i+1),
    x(i+1),y(i+1));
  end;
  if (x(i)>0 & y(i)>0)
   theta(i)=atan(y(i)/x(i));
  elseif (x(i)<0 & y(i)>0)
   theta(i)=pi+atan(y(i)/x(i));
  elseif ((x(i)<0 & y(i)<0))
   theta(i)=pi+atan(y(i)/x(i));
  else
   theta(i)=2*pi+atan(y(i)/x(i));
  end;
  r(i)=sqrt(y(i)^2+x(i)^2);
end;

%subplot(2,1,1)
%plot(t,x);
%plot(t,x,'b-','MarkerSize',5)
%ylabel('X(m)');
%xlabel('t(s)');
%legend('X(t)');
%subplot(2,1,2)
%plot(t,y);
%ylabel('Y(m)')
%xlabel('t(s)')
%legend('Y=f(t)');
%subplot(3,1,3)
polar(theta,r,'-b');
%xlabel('X(m)')
%ylabel('Y(m)');
```

IV. Método de Runge Kutta de orden 2

Los métodos de Runge Kutta son generalizaciones de los métodos de Euler, veamos RK2 escrita en forma vectorial ya que se trata de un sistema de ecuaciones diferenciales [1], [3] .

$$\vec{x}_{n+1} = \vec{x}_n + \frac{h}{2}\left(\vec{k}_1 + \vec{k}_2\right) \tag{13}$$

donde

$$\vec{k}_1 = \vec{F}\left(x_n, y_n\right) \tag{14}$$

$$\vec{k}_2 = \vec{F}\left(x_n + h, y_n + h\vec{k_1}\right) \tag{15}$$

I. Script de RK2 con Octave

A continuación se muestra la implementación en Octave.

```
clear all;
global npts=1000;
global l=15;
global g=9.80;
global w=0.007272205;
global angulo=50;
global x0=15;
global y0=15;
global vx0=0;
global vy0=0;
dt=500/npts;
nt=npts/10;
fprintf('Paso de
tiempo usado
dt=tmax/npts=
%7.4f\n',dt);
ay0=-g*(y0/l); ax0=-g*(x0/l);
printf('t          X
Y  \n')
t(1)=0;
x(1)=x0;
y(1)=y0;
vx(1)=vx0;
vy(1)=vy0;

fprintf('%3.1f   %7.2f
%11.2f \n',t(1),
x(1),y(1));

for i=1:npts
f(i)=func1(x(i),vy(i));
h(i)=func2(y(i),vx(i));
t(i+1)=t(i)+dt;
m1=vx(i)*dt;
n1=vy(i)*dt;
k1=dt*f(i);
L1=dt*h(i);
m2=dt*(vx(i)+k1);
n2=dt*(vy(i)+L1);
f2(i)=func1(x(i)+m1,vy(i)+L1);
h2(i)=func2(y(i)+n1,vx(i)+k1);
k2=dt*f2(i);
L2=dt*h2(i);
vx(i+1)=vx(i)+(k1+k2)/2;
vy(i+1)=vy(i)+(L1+L2)/2;
x(i+1)=x(i)+(m1+m2)/2;
y(i+1)=y(i)+(n1+n2)/2;
%x(i+1)=x(i)+dt*vx(i+1);
%y(i+1)=y(i)+dt*vy(i+1);
r(i)=sqrt(y(i)^2+x(i)^2);

if(mod(i,nt)==0)
    fprintf('%3.1f
    %6.2f    %9.2f \n',t(i+1),x(i+1),y(i+1));
end;

if (x(i)>0 & y(i)>0)
   theta(i)=atan(y(i)/x(i));
elseif (x(i)<0 & y(i)>0)
   theta(i)=pi+atan(y(i)/x(i));
elseif ((x(i)<0 & y(i)<0))
   theta(i)=pi+atan(y(i)/x(i));
else
   theta(i)=2*pi+atan(y(i)/x(i));
end;

end;

%subplot(2,1,1)
%plot(t,x);
%plot(t,x,'MarkerSize',5)
%ylabel('X(m)');
%xlabel('t(s)');
%legend('X(t)');
%subplot(2,1,2)
%plot(t,y);
%ylabel('Y(m)')
%xlabel('t(s)')
%legend('Y=f(t)');
%subplot(3,1,3)
polar(theta,r,'-b');
```

```
%xlabel('X(m)')
%ylabel('Y(m)');
```

V. Método de Runge Kutta de orden 4

El método de RK4 es uno de los mas populares, para resolver ecuaciones diferenciales por su alta precisión, el algoritmo en forma vectorial es [1], [2]

$$\vec{x}_{t+h} = \vec{x} + \frac{1}{6}\left(\vec{k}_1 + 2\vec{k}_2 + 2\vec{k}_3 + \vec{k}_4\right) \quad (16)$$

donde

$$\vec{k}_1 = \vec{F}\left(t+h\right) \quad (17)$$

$$\vec{k}_2 = \vec{F}\left(t+\frac{1}{2}h,\ \vec{x}+\frac{1}{2}h\vec{k}_1\right) \quad (18)$$

$$\vec{k}_3 = \vec{F}\left(t+\frac{1}{2}h,\ \vec{x}+\frac{1}{2}h\vec{k}_2\right) \quad (19)$$

$$\vec{k}_4 = \vec{F}\left(t+h,\ \vec{x}+\frac{1}{2}h\vec{k}_3\right) \quad (20)$$

I. Programa de RK4 en Octave

A continuación se muestra la implementación en Octave.

```
clear all;
global npts=1000;
global l=200;
global g=9.80;
global
w=0.010007272205;
global angulo=50;
global x0=15;
global y0=15;
global vx0=0;
global vy0=0;
dt=1000/npts;
nt=npts/10;
fprintf('Paso de
tiempo usado dt=tmax/npts=%7.4f\n',dt);
ay0=-g*(y0/l);
ax0=-g*(x0/l);
printf('t
X           Y  \n')
t(1)=0;
x(1)=x0;
y(1)=y0;
vx(1)=vx0;
vy(1)=vy0;

fprintf('%8.3f   %8.3f
%8.3f \n',t(1),x(1),y(1));

for i=1:npts
k1=dt*vx(i);
l1=dt*func1(x(i),vy(i));
q1=dt*vy(i);
m1=dt*func2(y(i),vx(i));

k2=dt*(vx(i)+l1/2);
l2=dt*func1(x(i)+k1/2,vy(i)+m1/2);
q2=dt*(vy(i)+m1/2);
m2=dt*func2(y(i)+q1/2,vx(i)+l1/2);

k3=dt*(vx(i)+l2/2);
l3=dt*func1(x(i)+k2/2,vy(i)+m2/2);
q3=dt*(vy(i)+m2/2);
m3=dt*func2(y(i)+q2/2,vx(i)+l2/2);

k4=dt*(vx(i)+l3);
l4=dt*func1(x(i)+k3,vy(i)+m3);
q4=dt*(vy(i)+m3);
m4=dt*func2(y(i)+q3,vx(i)+l3);

x(i+1)=x(i)+(k1+2*k2+2*k3+k4)/6;
vx(i+1)=vx(i)+(l1+2*l2+2*l3+l4)/6;
y(i+1)=y(i)+(q1+2*q2+2*q3+q4)/6;
vy(i+1)=vy(i)+(m1+2*m2+2*m3+m4)/6;

t(i+1)=t(i)+dt;

r(i)=sqrt(y(i)^2+x(i)^2);

if (x(i)>0 & y(i)>0)
   theta(i)=atan(y(i)/x(i));
elseif (x(i)<0 & y(i)>0)
   theta(i)=pi+atan(y(i)/x(i));
elseif ((x(i)<0 & y(i)<0))
   theta(i)=pi+atan(y(i)/x(i));
else
   theta(i)=2*pi+atan(y(i)/x(i));
end;


if(mod(i,nt)==0)
```

```
    fprintf('%8.3f
    %8.3f    %8.3f \n',t(i+1),x(i+1),
    y(i+1));
end;

end;



%subplot(2,1,1)
%plot(t,x);
%plot(t,x,'MarkerSize',5)
%ylabel('X(m)');
%xlabel('t(s)');
%legend('X(t)');
%subplot(2,1,2)
%plot(t,y);
%ylabel('Y(m)')
%xlabel('t(s)')
%legend('Y=f(t)');
%subplot(3,1,3)
polar(theta,r,'-b');
%xlabel('X(m)')
%ylabel('Y(m)');
```

Las funciones func1 y func2 son las siguientes.

```
function f1=func1(x,vy)
global g;
global l;
global w;
global angulo;
f1=-(g/l)*x+2*w*sind(angulo)*vy;
endfunction

function f2=func2(y,vx)
global g;
global l;
global w;
global angulo;
f2=-(g/l)*y-2*w*sind(angulo)*vx;
endfunction
```

VI. Resultados

Consideremos un péndulo de 20 metros de longitud que esta oscilando en un marco referencia giratorio con la frecuencia angular de la Tierra $0.000072 rad/s$, resolviendo con RK4 se obtiene la trayectoria del péndulo proyectada en el plano xy, como se muestra en la figura 2

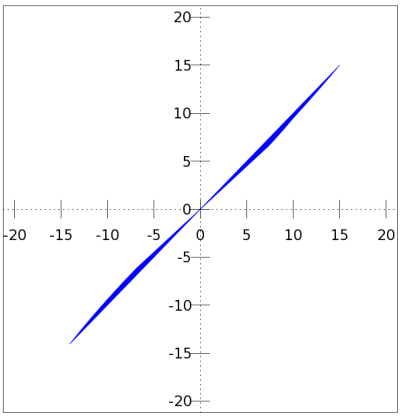

Figura 2: *Péndulo de Foucault*

Claramente las lineas superpuestas demuestran que el péndulo tiene un movimiento lateral. Aumentando la frecuencia angular se observa la siguiente trayectoria, figura 3.

El efecto de precesión se vuelve más notable si

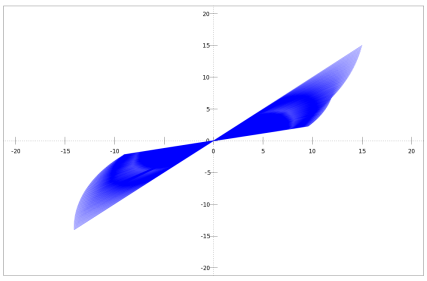

Figura 3: *Péndulo de Foucault con mayor frecuencia*

aumentamos la longitud del péndulo, observe la figura 4.

Los gráficos anteriores muestran claramente

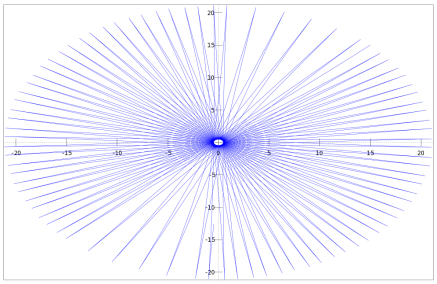

Figura 4: *Péndulo con mayor longitud*

que la trayectoria que realiza el péndulo tiene movimiento de precesión, debido a que esta en un marco de referencia giratorio.

Referencias

[1] Ward Cheney. *Numerical Mathematics and computing*. Thompson, 2008.

[2] Mark Newman. *Computational Physics*. Create Space Independet Publishing Platform, 2012.

[3] Hisao Nakanishi Nicholas Giordano. *Computational Physics*. Addison Wesley, 2005.

[4] John Taylor. *Classical Mechanics*. University Science Books, 2005.

[5] Stephen T. Thornton. *Classical dynamics of particles and systems*. Cengage Learning, 5 edition, 2003.

[6] Frank Berkshire Tom Kibble. *Classical Mechanics*. Imperial College Press, 2004.