UNIVERSIDAD DISTRITAL
FRANCISCO JOSÉ DE CALDAS

Nonlinear Pendulum with Friction

Laura Sofía Cortés y Yerly Zaudi Gomez Contreras

Física Computacional 1, Programa académico de Física, Universidad Distrital Francisco José de Caldas

23 de Mayo de 2023

1. Introduction

When we want to know a system in greater detail, nature insists on being nonlinear.The nonlinear of the pendulum do that its equation will be difficult to solve analytically. For this reason, approximations are usually made for a way around, so, When we have a small angle $sin\theta \approx \theta$, for $\theta < 10°$. This converts into a linear system that is easy to solve.However, by constraining the system in this way, we lose some of the physics of the system. In physics, non-linearity is vital to the operation of a laser and the formation of turbulence in a fluid.

The aim of this work is to compare the behaviour of the non-linear pendulum by means of its analytical solution, the Runge Kutta method implemented in the language C++, with the results obtained experimentally.For this purpose, the solution of the elliptic integrals, the experimental set-up of the simple pendulum and finally the following steps are carried out for the development of the computational problem:

- Spreadsheet model
- Pseudocode
- Algorithm
- C++ code

2. Objectives

2.1. General Objective

- Solve the nonlinear pendulum differential equation computationally in c++ with the Runge-Kutta method.

2.2. Specific Objectives

- To develop the physical-mathematical model of the nonlinear pendulum.
- Experimental set-up of the nonlinear pendulum.

- Obtain experimental data and plots of angle vs. time.
- Apply Runge-Kutta to the differential equation of the physical-mathematical model in c++.
- Compare experimental and numerical solution.

3. Theoretical framework

3.1. Differential equation deduction

The nonlinear pendulum consists of a point mass m suspended from a fixed point by means of a wire of length L firm and with negligible mass. Suppose that the point mass is displaced from its equilibrium position, where the net force is equal to zero, if it is displaced from this position with an angle *theta* the mass will start to move in an arc length s and to oscillate, performing Newton we can decompose the forces acting on the mass. In addition, a frictional force proportional to a constant by the first derivative of the angle will be taken into account for experimental comparison. Figure 1.

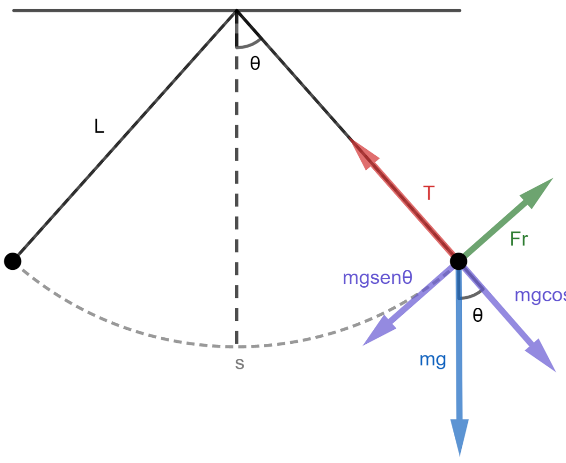


Figura 1: Physical Pendulum Model. **(Own Authorship, 2023)**

By summing the forces, we have that:

$$\sum F_x = -mg\,\mathrm{sen}(\theta) - b\frac{\mathrm{d}\theta}{\mathrm{d}t} = m\frac{\mathrm{d}^2 s}{\mathrm{d}t^2}$$

Since $s = L\theta$ and if we consider $L = mathrmcte$, it turns out:

$$-mg\,\mathrm{sen}(\theta) - b\frac{\mathrm{d}\theta}{\mathrm{d}t} = mL\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2}$$

By dividing by mL and reordering:

$$\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} + \frac{b}{mL}\frac{\mathrm{d}\theta}{\mathrm{d}t} + \frac{g}{L}\,\mathrm{sen}(\theta) = 0 \tag{1}$$

This is a nonlinear differential equation, its analytical solution corresponds to a series of infinite terms. And it is subject to the initial conditions $\theta(0) = \theta_{max}$ and $\frac{\mathrm{d}\theta}{\mathrm{d}t}(0) = 0$. Where θ_{max} is the maximum angular amplitude.

3.2. Analytical Solution

The solution of certain types of nonlinear systems can be represented in closed form by elliptic integrals. For example, a plane pendulum. We have:

Where θ is the angle and ω_0 is the natural frequency of oscillation. The period is given by

$$\tau \approx 2\pi\sqrt{\frac{l}{g}} \tag{2}$$

To use the energy instead of the equation of motion

$$T + U = E = constat$$

we consider that the potential energy is zero at the lowest point of the circular trajectory described by the mass of the pendulum. Then we can express the energy as:

$$T = \frac{1}{2}I\omega^2 = \frac{1}{2}ml^2\dot{\theta}^2$$
$$U = mgl(1 - cos\theta)$$

Where I is the angular momentum, ω the angular frequency and l the length of the pendulum. If we have:

$$T(\theta = \theta_0) = 0$$
$$U(\theta = \theta_0) = E = mgl(1 - cos\theta_{\theta 0})$$

Using the trigonometric identities we can rewrite as:

$$E = U = 2mglsin^2(\theta_0/2) \tag{3}$$

For the kinetic energy we have $T = E - U$,

$$\frac{1}{2}ml^2\dot{\theta}^2 = 2mgl[sin^2(\theta_0/2) - sin^2(\theta/2)]$$

Thus we have for the speed

$$\dot{\theta} = 2\frac{g}{l}[sin^2(\theta_0/2) - sin^2(\theta/2)]^{-\frac{1}{2}}d\theta \tag{4}$$

from which

$$dt = \frac{1}{2}\sqrt{\frac{l}{g}}[sin^2(\theta_0/2) - sin^2(\theta/2)]^{-\frac{1}{2}}d\theta$$

We can integrate this equation to obtain the period τ, the integral over $\theta = 0$ to $\theta = \theta_0$

$$\tau = 2\sqrt{\frac{1}{2}}\int_0^{\theta_0}[sen^2(\theta_0/2) - sen^2(\theta/2)]^{-}\frac{1}{2}$$

This is an elliptic integral of first order,can be clearly seen if we make the following substitutions:

$$z = \frac{cos(\theta/2)}{2sin(\theta_0/2)}$$
$$k = sin(\theta_0/2)$$
$$dz = \frac{cos(\theta/2)}{2sin(\theta_0/2)}d\theta = \frac{\sqrt{1 - k^2z^2}}{2k}d\theta$$

so

$$\tau = 4\sqrt{\frac{l}{g}}\int_0^{\theta_0}[(1-z^2)(1-k^2z^2)]^{-\frac{1}{2}}dz$$

For oscillatory motion to result. $|\theta_0| < pi$, therefore $-1 < k < +1$, we can replace in the integral by the following power series:

$$(1-k^2z^2)^{-\frac{1}{2}} = 1 + \frac{k^2z^2}{2} + \frac{3k^4z^4}{8} + ...$$

The expression for the period becomes

$$\tau = 2\pi\sqrt{\frac{l}{g}}[1 + \frac{k^2}{4} + \frac{9k^4}{64} + ...] \tag{5}$$

If $|k|$ is large, then we need many terms to produce an accurate result, but for a small k the expansion converges reasonably fast.
The equation for the angular velocity provides the necessary relationship $(\dot{(\theta)} = \dot{\theta}(\theta))$ to construct a phase diagram. The parameter θ_0 specifies the total energy.

3.3. The 4th Order Runge-Kutta Method

The fourth-order Runge-Kutta method is a numerical method for solving ordinary differential equations of the form:

$$\frac{dy}{dx} = f(x,y) \tag{6}$$

The method uses a weighted average of four estimates of the slope to update the value of y at each time step. The formula for the fourth-order Runge-Kutta method is:

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)h \tag{7}$$

where h is the step size, k_1, k_2, k_3, and k_4 are given by:

$$\begin{aligned} k_1 &= f(x_n, y_n) \\ k_2 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_1h) \\ k_3 &= f(x_n + \frac{1}{2}h, y_n + \frac{1}{2}k_2h) \\ k_4 &= f(x_n + h, y_n + k_3h) \end{aligned}$$

The fourth-order Runge-Kutta method is known for its high accuracy and stability, and is widely used in scientific and engineering applications.
In order to apply the Runge-Kutta method to equation 1, we have to make a reduction of order, since this equation is of order 2, for this we make the following change of variable:

$$\alpha = \theta' \Rightarrow \alpha' = \theta'' \tag{8}$$

Replacing in 1 and in the initial conditions:

$$\alpha' = -\frac{b}{mL}\alpha - \frac{g}{L}sen\theta \tag{9}$$

$$\theta(0) = \theta_{max} \quad , \quad \alpha(0) = 0 \tag{10}$$

We are now left with a system of two first order equations. We define a function $f(\theta, t)$ for each of the equations.

$$f_1(\theta, t) = \theta' = \alpha \tag{11}$$

$$f_2(\theta, t) = \alpha' = -\frac{b}{mL}\alpha - \frac{g}{L}sen\theta \tag{12}$$

Where the initial conditions are given by the following matrix:

$$CI = \begin{bmatrix} \theta_0 \\ \alpha_0 \end{bmatrix} = \begin{bmatrix} \theta_{max} \\ 0 \end{bmatrix} \tag{13}$$

Es decir, $\theta_0 = \theta_{max}$ y $\alpha_0 = 0$.

We will have then the k for each function, that is to say for f_1 we have $k_{11}, k_{21}, k_{31}, k_{41}$ and for f_2 we have $k_{12}, k_{22}, k_{32}, k_{42}$, we must find the set the 4 k for each iteration that is done, taking the definitions of the k as follows:

$$k_{11} = f_1(t, \theta_0) \quad , \quad k_{12} = f_2(\theta, t) \tag{14}$$

$$k_{21} = f_1\left(t_0 + \frac{h}{2}, \theta_0 + \frac{h}{2}k_{11}\right) \quad , \quad k_{22} = f_2\left(t_0 + \frac{h}{2}, \alpha_0 + \frac{h}{2}k_{12}\right) \tag{15}$$

$$k_{31} = f_1\left(t_0 + \frac{h}{2}, \theta_0 + \frac{h}{2}k_{21}\right) \quad , \quad k_{32} = f_2\left(t_0 + \frac{h}{2}, \alpha_0 + \frac{h}{2}k_{22}\right) \tag{16}$$

$$k_{41} = f_1\left(t_0 + h, \theta_0 + hk_{31}\right) \quad , \quad k_{42} = f_2\left(t_0 + h, \alpha_0 + hk_{32}\right) \tag{17}$$

3.4. Problem statement

The general problem is to solve the equation of the nonlinear pendulum taking into account a friction force which is proportional to a constant by the angular velocity in a computational way, using the programming language c++ and applying the Runge-Kutta fourth order method. This differential equation is presented in this way in order to make a comparison between the computational solution and the experimental solution, for which an assembly and data collection will be performed. Due to the fact that the assembly was carried out with a wire which, when displaced at an angle greater than 90°, moved in free fall, this alters the assumption that L is constant, for which reason the data is truncated up to 130°.

4. Methodology

4.1. Computational assembly

4.1.1. Pseudocode

See Algorithm 1.

Algorithm 1: Pseudocode

```
- Includes the necessary libraries
- Defines a class called "Pendulum simulation"
- Includes member variables (private)
    ▪ $b$: Damping constant
    ▪ $m$: Mass
    ▪ $L$: Pendulum length
    ▪ $\theta_{Max}$: Initial value of $\theta$
    ▪ $h$: Step size
    ▪ $t_f$: Final time
- Methods (public)
    ▪ We define the f2 that calculates the derivative of $\theta$ and $\alpha$ with respect to time
- Define the constructor. Initializes the private variables provided in the object.
- Define f2 ($f_2$ Runge Kutta's method), we set the dependence parameters $\theta$ y $\alpha$
    $f_2 = -(b/(m * L)) * \alpha - (9{,}8/L) * sin(\theta)$
- To solve Runge Kutta's method we set up a new function
$voidsolve()\{$
  1. Initialize local variables for initial time ($t$), initial angle ($\theta$), and initial angular velocity ($\alpha$).
  2. Calculate number of steps $h$
  3. Open the results output files: results1.txt and results2.txt, store the results.
  4. We start the loop for the iterations in this case For is used.
  5. In every step, calculate the coeficients $k_{11}$, $k_{12}$, $k_{21}$, $k_{22}$, $k_{31}$, $k_{32}$, $k_{41}$, $k_{42}$ using the fourth-order Runge-Kutta method.
  6. Update the values of $\theta$ and $\alpha$ with the calculated values for $k$.
  7. Update the time by adding the step $t + h$.
  8. Write the values in the result files and close.
  9. Close the function solve.
     - Define the function grapThetaVsTime: Receive expfile (path to file with experimental data)
     - Set commands for graphs using xmgrace and expfile.
     - Execute commands using system()
     - Define function graphAlphaVsTheta: Create a command string to graph the $alpha$ vs $theta$ data using xmgrace. Execute the command string using system()
     -Create object
     $intmain()\{$
       a) Define variables and constants.
       b) Create object with the values provided.
       c) Call solve.
       d) Call graphics.
```

4.1.2. Flowchart

4.1.3. Code

The following is the c++ code used to plot the experimental data, solve numerically the differential equation of the nonlinear pendulum and perform the respective plots.

```
#include <iostream>
#include <fstream>
#include <cmath>

class PendulumSimulation {

private:
    double b;        // Constante de amortiguamiento
    double m;        // Masa
    double L;        // Longitud del péndulo
    double thetaMax; // Valor inicial de theta
    double h;        // Tamaño del paso
    double tf;       // Tiempo final

public:
    // Constructor
    PendulumSimulation(double damping, double mass, double length,
                       double initialTheta, double stepSize, int finalTime)
        : b(damping), m(mass), L(length),
          thetaMax(initialTheta), h(stepSize), tf(finalTime) {}

    // Función f2
    double f2(double theta, double alpha) {
        return -(b / (m * L)) * alpha - (9.8 / L) * std::sin(theta);
    }

    // Resolver usando método de Runge-Kutta de cuarto orden
    void solve() {

        double t = 0.0;          // Tiempo inicial
        double theta = thetaMax; // Theta inicial
        double alpha = 0.0;      // Alpha inicial

        // Calcular número de pasos
        int numSteps = static_cast<int>(std::round(tf / h));

        // Abrir archivo de texto en modo escritura
        // Guardar tiempo vs theta
        std::ofstream outputFile1("results1.txt");
        // Abrir archivo de texto en modo escritura
        // Guardar alpha vs theta
        std::ofstream outputFile2("results2.txt");
```

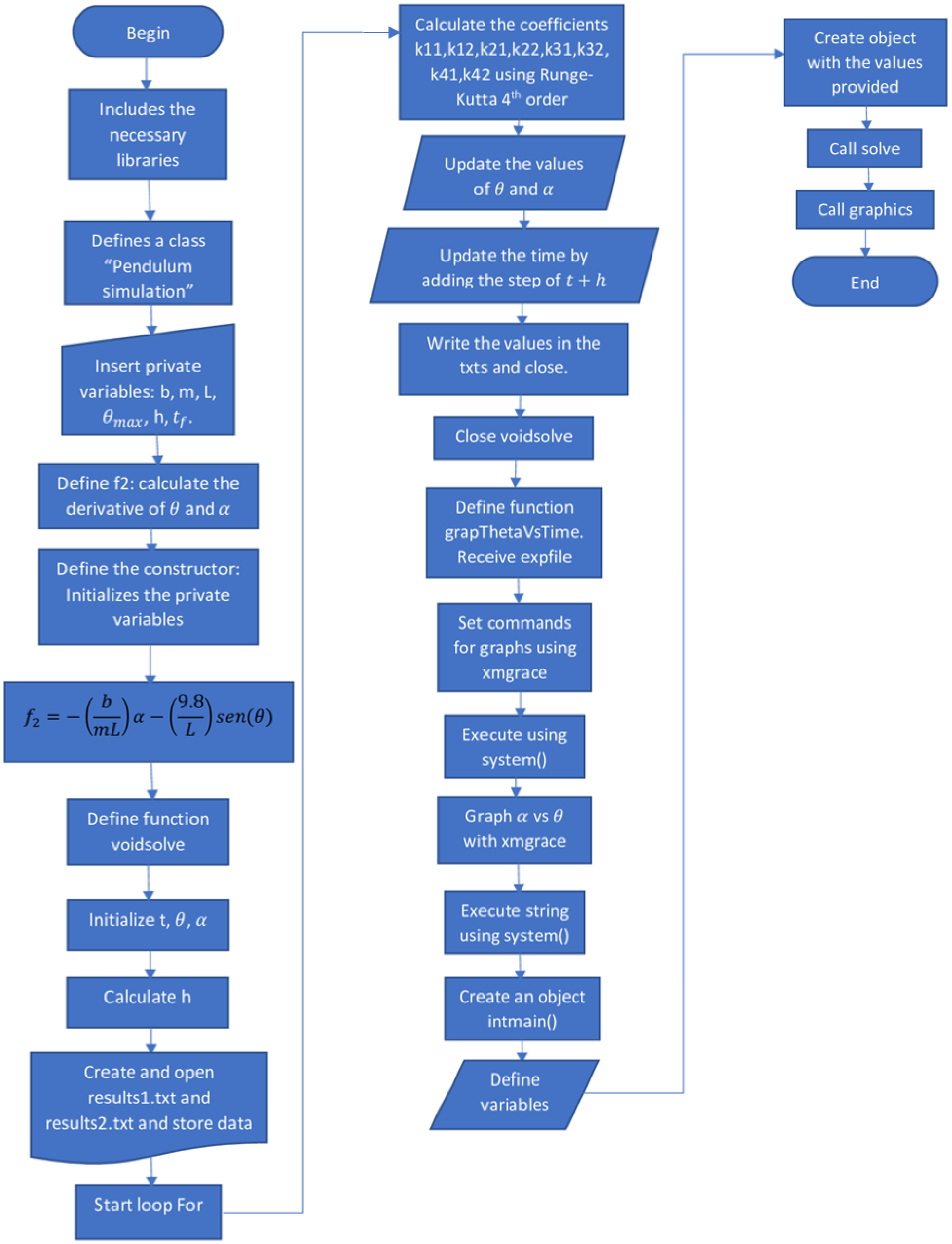

Begin
Includes the necessary libraries
Defines a class "Pendulum simulation"
Insert private variables: b, m, L, θ_{max}, h, t_f.
Define f2: calculate the derivative of θ and α
Define the constructor: Initializes the private variables
$f_2 = -\left(\frac{b}{mL}\right)\alpha - \left(\frac{9.8}{L}\right) sen(\theta)$
Define function voidsolve
Initialize t, θ, α
Calculate h
Create and open results1.txt and results2.txt and store data
Start loop For
Calculate the coefficients k11,k12,k21,k22,k31,k32, k41,k42 using Runge-Kutta 4th order
Update the values of θ and α
Update the time by adding the step of $t + h$
Write the values in the txts and close.
Close voidsolve
Define function grapThetaVsTime. Receive expfile
Set commands for graphs using xmgrace
Execute using system()
Graph α vs θ with xmgrace
Execute string using system()
Create an object intmain()
Define variables
Create object with the values provided
Call solve
Call graphics
End

Figura 2: Flowchart

```
    // Verificar si el archivo se abrió correctamente
    if (!outputFile1 || !outputFile2) {
        std::cerr << "Error al abrir el archivo de salida." << std::endl;
        return;
    }

    // Escribir al archivo los valores iniciales de t y theta
    outputFile1 << t << " " << theta * 180.0 / M_PI << "\n";

    // Escribir al archivo los valores iniciales de alpha y theta
    outputFile2 << alpha << " " << theta * 180.0 / M_PI << "\n";

    for (int i = 0; i < numSteps; i++) {

        // Calcular coeficientes k
        double k11 = alpha;
        double k12 = f2(theta, alpha);

        double k21 = alpha + (h / 2.0) * k12;
        double k22 = f2(theta + (h / 2.0) * k11, alpha + (h / 2.0) * k12);

        double k31 = alpha + (h / 2.0) * k22;
        double k32 = f2(theta + (h / 2.0) * k21, alpha + (h / 2.0) * k22);

        double k41 = alpha + h * k32;
        double k42 = f2(theta + h * k31, alpha + h * k32);

        // Actualizar los valores de theta y alpha
        theta = theta + (h / 6.0) * (k11 + 2 * k21 + 2 * k31 + k41);
        alpha = alpha + (h / 6.0) * (k12 + 2 * k22 + 2 * k32 + k42);

        // Actualizar tiempo
        t = t + h;

        // Escribir al archivo los valores de t y theta
        outputFile1 << t << " " << theta * 180.0 / M_PI << "\n";
        outputFile2 << alpha << " " << theta * 180.0 / M_PI << "\n";
    }

    // Cerrar el archivo de texto
    outputFile2.close();
    outputFile2.close();
}

// Graficar theta vs tiempo
void graphThetaVsTime(std::string expfile) {
    std::string command = "xmgrace -nxy results1.txt -nxy " + expfile + " -pexec \"";
    // Título
    command += "title \\\"Posicion vs tiempo\\\";";
    // Estilizar
```

```
        command += "legend loctype view;";
        command += "s0 legend \\\"Calculada\\\";";
        command += "s1 legend \\\"Experimental\\\";";
        command += "s0 line color 15;";
        command += "s1 line color 12;";
        command += "s1 line linestyle 2;";
        command += "s1 line linewidth 2;";
        // Etiqueta de eje x
        command += "xaxis label \\\"Tiempo (s)\\\";";
        // Etiqueta de eje y, con fuente cambiada para visualización
        command += "yaxis label font 12;";
        command += "yaxis label \\\"q\\\";";
        command += "\" ";
        system(command.c_str());
    }

    // Graficar alpha vs theta
    void graphAlphaVsTheta() {
        std::string command = "xmgrace -nxy results2.txt -pexec \"";
        // Título
        command += "title \\\"Diagrama de Fase\\\";";
        // Estilizar
        command += "legend loctype view;";
        command += "s0 legend \\\"Calculada\\\";";
        command += "s0 line color 15;";
        // Etiqueta de eje x
        command += "xaxis label font 12;";
        command += "xaxis label \\\"a\\\";";
        // Etiqueta de eje y, con fuente cambiada para visualización
        command += "yaxis label font 12;";
        command += "yaxis label \\\"q\\\";";
        command += "\" ";
        system(command.c_str());
    }
};

int main() {

    double b = 0.0002;                   // Constante de amortiguamiento
    double m = 0.0464;                   // Masa
    double L = 0.22;                     // Longitud del péndulo
    double thetaMax = 24 * M_PI / 180.0; // Valor inicial de theta
    double h = 0.01;                     // Tamaño del paso
    double tf = 30.0;                    // Tiempo final

    // Ruta al artchivo con datos experimentales para graficar theta vs tiempo
    std::string expfile = "1.txt";

    // Instanciar objeto
    PendulumSimulation simulation(b, m, L, thetaMax, h, tf);
    // Resolver
    simulation.solve();
    // Graficar theta vs tiempo en xmgrace
    simulation.graphThetaVsTime(expfile);

    simulation.graphAlphaVsTheta();

    return 0;
}
```

4.2. Experimental Assembly

Measuring tools

- Goniometer.
- Tape measure.

Materials

- Universal support.
- String.
- Sphere of constant density.

Software and equipment

- *Tracker Video Analyzer*
- Time recording and measuring device (Mobile phone)

The universal support was positioned on the floor in front of a white wall, one of the arms of the universal support was extended to hang the ball with the thread, the ball had a metal ring embedded in it, from which the thread was tied firmly. The other end of the thread was taken and tied firmly to the previously extended arm of the universal support; in this way the length would remain constant during the oscillations.

The goniometer was placed on the universal support in such a way that it was on the horizontal or vertical with center where the string was attached, in order to have some reference of the angle at which the ball will be displaced from its equilibrium. In another universal support located just in front of the pendulum and with some tweezers the mobile device was placed to record the assembly and to be able to analyze it correctly in Tracker.

A video was recorded for 10 different angular amplitudes from approximately 30° to 160° for the same length which was from approximately $L = 22cm$ to approximately 115° and from $L = 27cm$ from that angle to 160°.

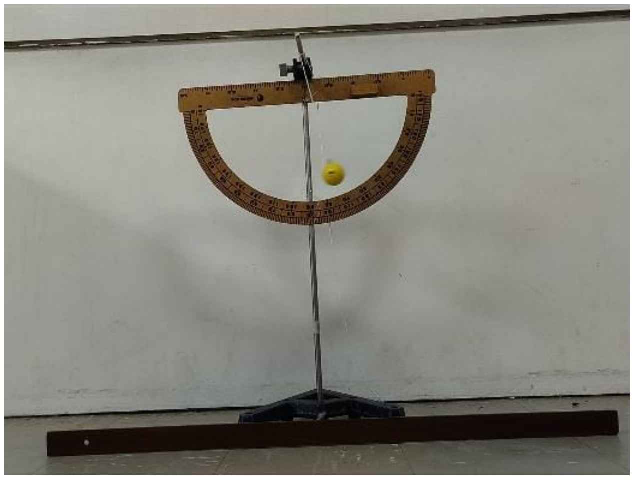
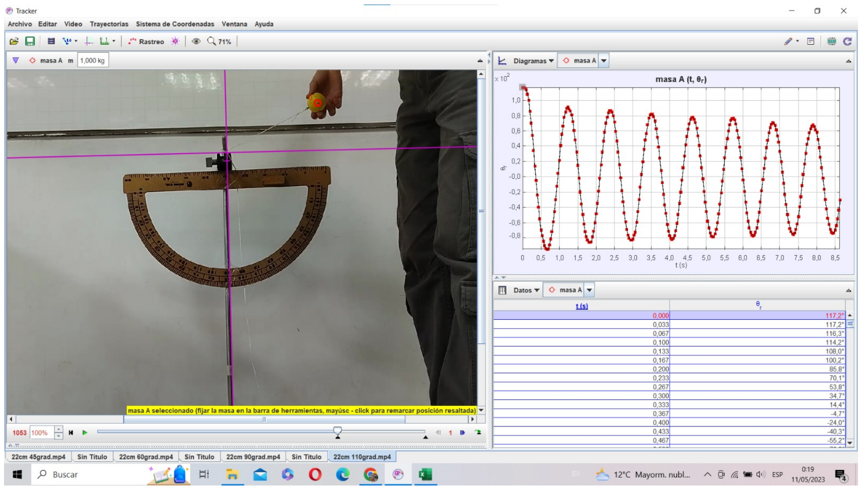


Figura 3: Experimental Assembly. **(Autoría propia, 2023)**


Tracker
Archivo Editar Video Trayectorias Sistema de Coordenadas Ventana Ayuda
Rastreo
masa A
Diagramas
masa A (t, θr)
t (s)
Datos
masa A (fijar la masa en la barra de herramientas, mayúsc - click para remarcar posición resaltada)
22cm 45grad.mp4
Sin Título
22cm 60grad.mp4
22cm 90grad.mp4
22cm 110grad.mp4

t (s)	θ_r
0,000	117,2°
0,033	117,2°
0,067	116,3°
0,100	114,2°
0,133	108,0°
0,167	100,2°
0,200	85,8°
0,233	70,1°
0,267	53,8°
0,300	34,7°
0,333	14,4°
0,367	-4,7°
0,400	-24,0°
0,433	-40,3°
0,467	-55,2°

Figura 4: Tracker interface in analysis of an angular amplitude of 117°. **(Own Authorship, 2023)**

In Tracker, for each angle, while keeping the radius constant, time, angle and angular velocity data were taken and placed in an Excel and txt document, also a txt was made for each amplitude with their respective initial conditions.

5. Results

5.1. Experimental Results

En el siguiente link se encuentra el documento Excel con los datos recolectados para el péndulo de 22cm de largo para diferentes ángulos con su gráfica de ángulo versus tiempo y diagrama de fase.
`https://udistritaleduco-my.sharepoint.com/:x:/g/personal/lscortesr_udistrital_edu_co/EYG_jqh30pdItV3jPTfPVc0B99WWusRmpAMywsEyzjN3pQ?e=69KeJ1`

Tomando 2 datos de tiempo y de ángulo se puede determinar el coeficiente b para nuestro montaje y este, usarlo como constante para determinar los resultados experimentales. Se tiene que para una amplitud angular aproximada de 89, $\theta_0 = 89{,}4°$ con $t_0 = 0{,}033s$ y $\theta_1 = 89{,}2°$ con $t_1 = 0{,}067s$. Tomando la función de la amplitud dependiendo del tiempo de un oscilador amortiguado se tiene:

$$\theta_1 = \theta_0 e^{-\frac{b}{2m}t}$$

Despejando b y tomando $t = t_1 - t_0 = 0{,}067s - 0{,}033s = 0{,}034s$ y sabiendo que la masa del objeto es $m = 0{,}0464$kg.

$$-\frac{b}{2m}t = ln\left(\frac{\theta_1}{\theta_0}\right)$$

$$b = -\frac{2m}{t}ln\left(\frac{\theta_1}{\theta_0}\right)$$

Reemplazando:

$$b = -\frac{2(0{,}0464kg)}{0{,}034s}ln\left(\frac{89{,}2°}{89{,}4°}\right)$$

Por lo tanto,

$$b = 0{,}006113$$

5.2. Computational Results

Computational results can be obtained by run the c++ code shown before. The code will ask us the constants b, m, L, g, h, $\theta(0)$, this can be obtained in the experimental assembly and b was calculated before.

An example of the obtained graphics with this code is:

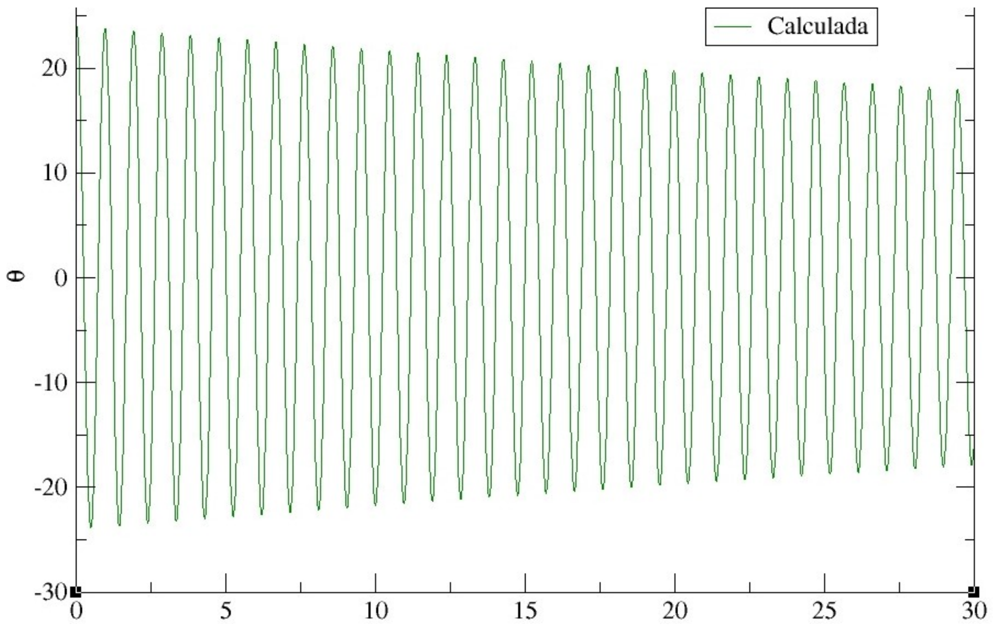


Figura 5: Computacional graphic of angle versus time. **(Own Authorship, 2023)**

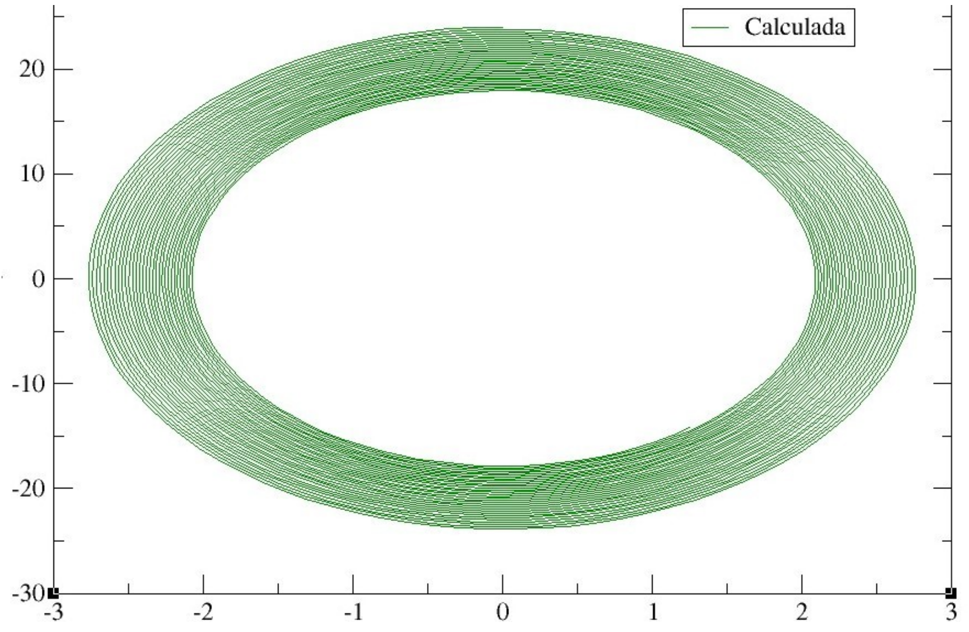


Figura 6: Computacional graphic of phase. **(Own Authorship, 2023)**

6. Análisis y Discusión

Using the constants obtained in the experimental model we can compare the computational and the experimental graphics, the result is the following.

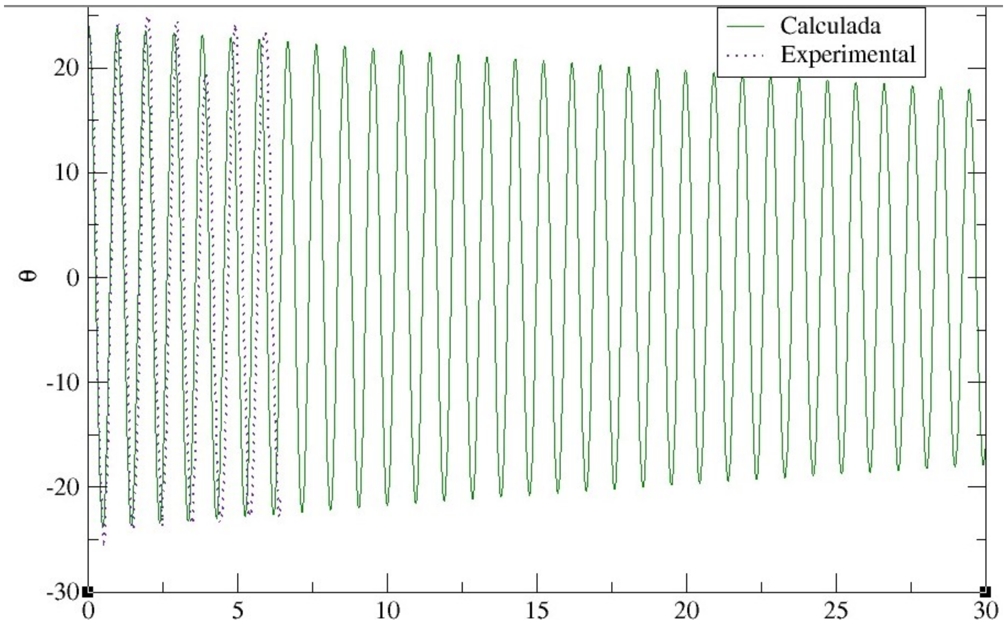


Figura 7: Computacional graphic and experimental of angle versus time. **(Own Authorship, 2023)**

We can see that the computational model fits very well to the experimental model.