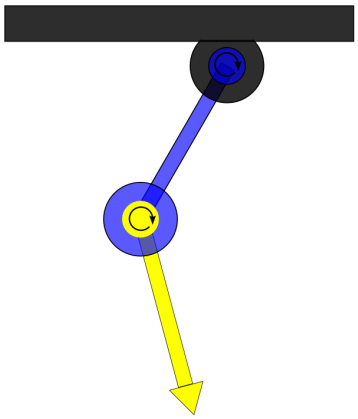
# Study the Dynamics of Double Pendulum

## 1 Introduction

The double pendulum is a classic example in physics that illustrates the complex and chaotic behavior of dynamic systems. It consists of two simple pendulums, where one pendulum is attached to the end of the other, creating a system with two degrees of freedom.
This setup leads to rich and intricate motion, highly sensitive to initial conditions, making it an ideal subject for studying nonlinear dynamics and chaos theory.



Unlike a simple pendulum, which exhibits periodic motion, the double pendulum's movement can vary unpredictably over time. Even with a slight difference in the initial conditions, the resulting paths can diverge significantly.

## 2 Parameters Required for the C++ Code

To simulate the dynamics of a double pendulum using C++, we need the following parameters:

| **Parameters Input by User** | **Symbol** |
|---|---|
| Mass of first pendulum | mass1 |
| Mass of second pendulum | mass2 |
| Length of first pendulum | length1 |
| Length of second pendulum | length2 |
| The angle of suspension of the first pendulum | angle1 |
| The angle of suspension of the second pendulum | angle2 |

Table 1

The acceleration due to gravity (g) is a pre-defined parameter in this problem.

## 3 Governing Equations of Motion

The system is governed by a set of coupled differential equations, which do not have a simple analytical solution. Instead, the behavior of the double pendulum is typically explored through numerical simulations and phase space analysis.
Assume the masses of the two pendulums are $m_1$ and $m_2$, and the lengths of the pendulums are $l_1$ and $l_2$. The angles that the pendulums make with the vertical are $\theta_1$ and $\theta_2$, respectively. The Lagrangian for the system is given by:

$$L = \frac{1}{2}m_1 l_1^2 \dot{\theta}_1^2 + \frac{1}{2}m_2 \left[ l_1^2 \dot{\theta}_1^2 + l_2^2 \dot{\theta}_2^2 + 2 l_1 l_2 \dot{\theta}_1 \dot{\theta}_2 \cos(\theta_1 - \theta_2) \right] - m_1 g l_1 \cos\theta_1 - m_2 g \left[ l_1 \cos\theta_1 + l_2 \cos\theta_2 \right], \quad (1)$$

The equations of motion derived from the Euler-Lagrange equations are:

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{\theta}_1}\right] - \frac{\partial L}{\partial \theta_1} = 0, \quad (2)$$

$$\frac{d}{dt}\left[\frac{\partial L}{\partial \dot{\theta}_2}\right] - \frac{\partial L}{\partial \theta_2} = 0. \quad (3)$$

These lead to the following second-order nonlinear differential equations:

$$(m_1 + m_2) l_1 \ddot{\theta}_1 + m_2 l_2 \ddot{\theta}_2 \cos(\theta_1 - \theta_2) + m_2 l_2 \dot{\theta}_2^2 \sin(\theta_1 - \theta_2) + (m_1 + m_2) g \sin\theta_1 = 0, \quad (4)$$

$$l_2 \ddot{\theta}_2 + l_1 \ddot{\theta}_1 \cos(\theta_1 - \theta_2) - l_1 \dot{\theta}_1^2 \sin(\theta_1 - \theta_2) + g \sin\theta_2 = 0. \quad (5)$$

Upon solving equations (4) and (5), we get the following equations:

$$\frac{d^2\theta_1}{dt^2} = \frac{-g(2m_1 + m_2)\sin(\theta_1) - m_2 g \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1 - \theta_2) m_2 \left( \frac{d^2\theta_2}{dt^2} + \frac{d^2\theta_1}{dt^2} \cos(\theta_1 - \theta_2) \right)}{l_1 \left(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)\right)}, \quad (6)$$

$$\frac{d^2\theta_2}{dt^2} = \frac{2\sin(\theta_1 - \theta_2) \left( \frac{d^2\theta_1}{dt^2}(m_1 + m_2) + g(m_1 + m_2)\cos(\theta_1) + \frac{d^2\theta_2}{dt^2} l_2 m_2 \cos(\theta_1 - \theta_2) \right)}{l_2 \left(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)\right)}. \quad (7)$$

If we substitute

$$\omega_1 = \frac{d\theta_1}{dt} \quad (8)$$

and,

$$\omega_2 = \frac{d\theta_2}{dt} \quad (9)$$

Finally, we get the equations for acceleration for the dynamics of a double pendulum.

$$\frac{d\omega_1}{dt} = \frac{-g(2m_1 + m_2)\sin(\theta_1) - m_2 g \sin(\theta_1 - 2\theta_2) - 2\sin(\theta_1 - \theta_2) m_2 \left( \omega_2^2 + \omega_1^2 \cos(\theta_1 - \theta_2) \right)}{l_1 \left(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)\right)}, \quad (10)$$

$$\frac{d\omega_2}{dt} = \frac{2\sin(\theta_1 - \theta_2) \left( \omega_1^2 (m_1 + m_2) + g(m_1 + m_2)\cos(\theta_1) + \omega_2^2 l_2 m_2 \cos(\theta_1 - \theta_2) \right)}{l_2 \left(2m_1 + m_2 - m_2 \cos(2\theta_1 - 2\theta_2)\right)} \quad (11)$$

Understanding the motion of a double pendulum has applications in various fields, such as mechanical engineering, robotics, and the study of chaotic systems in nature. This introduction to the double pendulum aims to provide a foundational understanding of its dynamics, mathematical modeling, and the implications of its chaotic behavior.

# 4 Algorithm

1. **Include all the required header files.**
2. **Define a class 'double_pendulum'** which has all the above-listed parameters in Table 1 and pre-defined parameters declared as private members.
3. **Declare public member functions** in the class that can be accessed from outside the class using an object or instance of the class defined in the main function (the compiler starts the execution of any C++ program from the main() function)
4. **Definition of Member function 1:** void input ( ){//}
   The function initializes the parameters, including the mass of the first & second pendulums, the length of the first & second pendulums, and the angle of suspension of the first & second pendulums.

- Return-type: `void`
- The function doesn't accept any parameters.
- Apart from taking input from the user, the function implements two other important tasks:
  - Conversion of angles input by the user in degrees to radians, as the formulas involved require the angle in radians only.
  - Initializes the value of the gravitational acceleration (g) to 9.8 $m/s^2$.

5. **Definition of Member Function 2:** double acceleration1 (double, double, double, double) {//} This function calculates the angular acceleration of the mass 1 using eq. (10).
   - Return type: `double`
     The function returns the calculated value of the angular acceleration of mass 1.
   - The function accepts four parameters
     - $\theta_1$: The angle of suspension of first pendulum.
     - $\theta_2$: The angle of suspension of second pendulum.
     - $\omega_1$: The angular velocity of mass 1.
     - $\omega_1$: The angular velocity of mass 2.
6. **Definition of Member Function 3:** double acceleration2 (double, double, double, double) {//} This function calculates the angular acceleration of the mass 2 using eq. (11).
   - Return type: `double`
     The function returns the calculated value of the angular acceleration of mass 2.
   - The function accepts four parameters
     - $\theta_1$: The angle of suspension of first pendulum.
     - $\theta_2$: The angle of suspension of second pendulum.
     - $\omega_1$: The angular velocity of mass 1.
     - $\omega_1$: The angular velocity of mass 2.
7. **Definition of Member Function 4:** void calc (void) {//}
   This function calculates the angular displacements for different time steps via the Ralston Method, a second-order numerical method (for more details, check out the documentation for the same).
   - Return data-type: void This function doesn't return any value.
   - The function doesn't accept any parameter
   - The function performs the following important computations required for the final result:
     - Initializes the step size (h), the number of iterations, the angular velocity of mass 1 (angular_velocity1), and the angular velocity of mass 2 (angular_velocity2)
     - A for-loop is used to implement the Ralston method for calculating the angular velocities of both masses at different time steps and furthering the angular displacements of both masses. The loop gets terminated depending on the value of the total number of iterations.
     - The function finally creates a '.dat' data file and enters the data computed to this file, which includes angle1, angular_velocity1, angle 2, and angular_velocity2.
8. **Define the main function:** `int main()`
   - Creates an instance/object of the class 'double_pendulum' and makes function calls to all necessary functions.
   - Return 0 to indicate successful execution.
9. The final step in the algorithm involves a crucial step i.e., plotting the data saved in the file by the calc() function using **Gnuplot software**.
   Gnuplot is a graphing utility for Linux, OS/2, MS Windows, OSX, VMS, and many other platforms. Its source code is freely distributed and is extensively used for data visualization.
   Follow the following steps to plot data on Gnuplot:
   (a) Download Gnuplot.

(b) Go to → Change directory.
Select the folder where the 'phase_trajectory.dat' data file is stored.

(c) Write the following command in Gnuplot:

```
plot "phase_trajectory.dat" u 1:2 w l \\or
plot "phase_trajectory.dat" u 3:4 w l
\* This command asks the Gnuplot to plot Column 2 with respect to Column 1
    with a line-type graph or plot Column 4 with respect to Column 3 with a
    line-type graph. There are many other variations to this command, you can
    check them out at the official website of Gnuplot or watch some free
    tutorials on YouTube
```