Simulación de Péndulos Acoplados con Interacción No Lineal Cuadrática

Camilo R. Rodríguez

Octubre 2025
Física Computacional II — Profesor John Hernán Díaz

Resumen

Presentamos una implementación computacional en C++ que simula dos péndulos acoplados mediante un término cuadrático no lineal. Se emplea un integrador de Runge–Kutta de cuarto orden (RK4) y se realizan estudios comparativos variando el coeficiente de acoplamiento κ. El código es modular y produce salidas en formato CSV para su posterior análisis.

1. Introducción

Breve motivación y objetivos: modelar la dinámica no lineal, analizar sincronización y estudiar conservación de magnitudes.

2. Modelo físico

Las ecuaciones de movimiento utilizadas son:

$$\ddot{\theta}_1 + \frac{g}{l}\sin\theta_1 + \kappa(\theta_1 - \theta_2)^2 = 0, \qquad \ddot{\theta}_2 + \frac{g}{l}\sin\theta_2 + \kappa(\theta_2 - \theta_1)^2 = 0.$$

Aquí κ es el coeficiente de acoplamiento no lineal y l la longitud.

3. Método numérico

Se reescriben como un sistema de 4 ecuaciones de primer orden y se integra con RK4:

$$\mathbf{s} = (\theta_1,\ \omega_1,\ \theta_2,\ \omega_2)$$

El paso temporal se denota Δt; la implementación usa un paso fijo y guarda resultados a intervalos para reducir I/O excesivo.

4. Diseño del software

El código está organizado en POO:

- `include/Pendulo.h`: declaración de la clase `Pendulo`.
- `src/Pendulo.cpp`: implementación de aceleración y energías.
- `src/main.cpp`: lógica de ejecución, lectura de valores de κ, bucle sobre distintas corridas y guardado de resultados.
- `scripts/`: scripts en Python para graficar resultados individuales y comparar varias corridas.

5. Validación y consideraciones numéricas

Se recomienda:

- Probar varios valores de Δt para verificar convergencia.
- Comparar con la versión lineal (reemplazar $\sin\theta \to \theta$) para obtener modos normales en regime pequeño.
- Controlar la energía total: debido al acoplamiento no lineal, la energía definida con el potencial asociado al término de acoplamiento puede no ser estrictamente conservada; usar Δt pequeño para limitaciones numéricas.

6. Resultados (instrucciones)

Ejecutar el programa para varios κ y luego:

```
python3 scripts/plot_compare.py
```

7. Resultados y comparación de simulaciones

A continuación se muestran las comparaciones obtenidas al variar el parámetro de acoplamiento κ. Cada gráfica fue generada con el script `plot_compare.py` a partir de los archivos de resultados `results_kappaX.csv`.

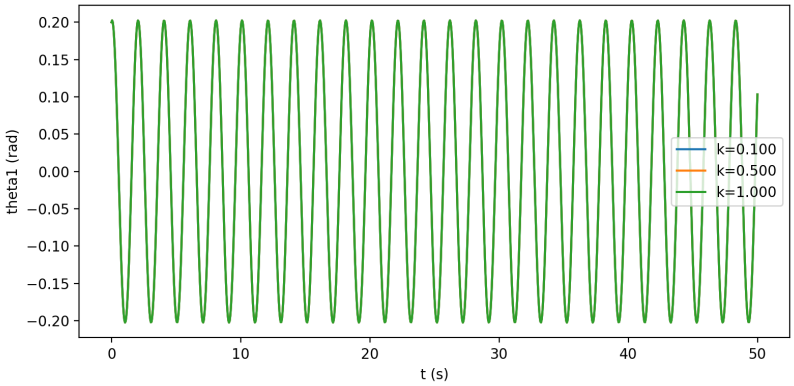

Figura 1: Evolución temporal de $\theta_1(t)$ para distintos valores de κ. Se observa cómo la frecuencia y la amplitud cambian al aumentar el acoplamiento, mostrando un intercambio de energía más rápido entre los péndulos.

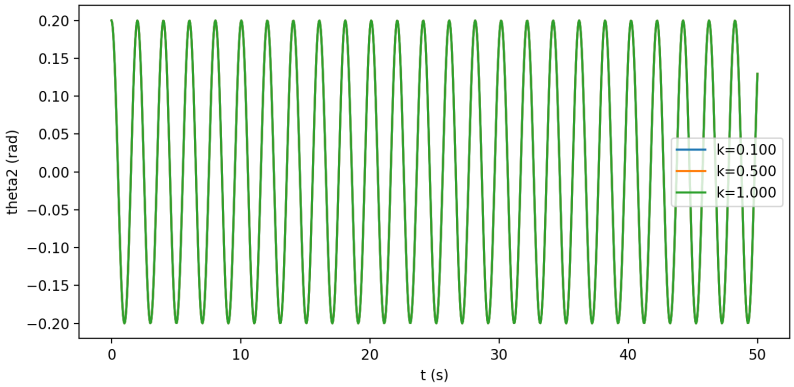

Figura 2: Evolución temporal de $\theta_2(t)$ para distintos valores de κ. Las curvas presentan una clara correlación con las oscilaciones del primer péndulo, evidenciando la sincronización parcial del sistema.

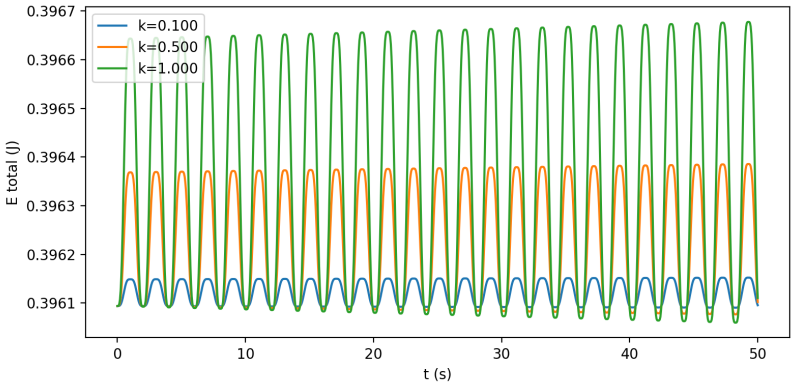

Figura 3: Energía total del sistema en función del tiempo. Debido a la naturaleza no lineal del acoplamiento, la energía no se conserva estrictamente, generando oscilaciones asociadas al intercambio entre modos.

8. Conclusiones

Resumen de hallazgos, discusión sobre sincronización y transferencia de energía, posibles extensiones (mapa de Poincaré, exponentes de Lyapunov).

A. Fragmento de código

Listing 1: Función de aceleración no lineal.

```
double Pendulo::accel(double theta_other, const Params& p) const {
    double grav = -(p.g / p.l) * std::sin(theta);
    double coupling = - p.kappa * (theta - theta_other) * (theta - theta_
    return grav + coupling;
}
```

B. Instrucciones para reproducir

Compilar con `make`, ejecutar `./bin/pendulos` y generar gráficas con `python3 scripts/plot`$_{c}$*ompare.py.Pararepro*$_{d}$*ucibilidad*, *incluirparmetrosycondicionesinicialesus*