Examen 1 — Física Computacional 2

Profesor: John Hernán Díaz

Trabajo en grupos de **2** personas

Octubre de 2025

Indicaciones generales

- El examen se realiza en **grupos de 2 personas**.
- El **Problema 1 es obligatorio** para todos los grupos.
- En el **Problema 2** cada grupo debe **escoger uno** de los planteados más abajo. Si ninguna fenomenología les satisface, pueden **proponer una** *en los mismos términos* (no linealidad/caos/sincronización/transporte, etc.) y nivel de complejidad equivalente, previa aprobación del profesor.
- Todo **Examen 1** debe ser un **proyecto modular** en C++ con **Make o CMake**. La modularización interna es libre, pero debe existir el **mínimo** que se detalla en la Sección .
- **Entrega**: repositorio en GitHub o GitLab (público o con acceso al profesor), más defensa oral del proyecto (presentación breve).

Resultados de aprendizaje

Modelar, implementar y analizar sistemas físicos complejos con enfoques multiescala y no lineales; producir software científico reproducible (POO en C++, integración numérica robusta, scripts de posprocesado, documentación técnica y análisis físico en LaTeX).

Problema 1 (Obligatorio): Simulación de N partículas en una caja

Implemente un programa en C++ que simule el movimiento de N partículas esféricas de masa igual confinadas en una caja rectangular de dimensiones $W \times H$. Las partículas tienen radio $r \ll \text{mín}(W, H)$, interactúan por **colisiones elásticas** entre sí y con paredes *(rebote perfecto)*. El programa debe generar un archivo con posiciones y velocidades de todas las partículas en función del tiempo.

Requerimientos mínimos

a) **Diseño POO:** Clase(s) para partículas (`Bola`) y, opcionalmente, `Caja`. Métodos para avanzar en el tiempo, choques con paredes y entre partículas.

b) **Integración temporal:** Euler (válido para prototipo) y una opción *estable* recomendada (*Velocity-Verlet* o *Leapfrog*).

c) **Salida de datos:** columnas $t, (x_i, y_i, vx_i, vy_i)$ para $i = 1, \ldots, N$.

d) **Experimentos** (al menos 2): trayectorias; histograma de $|\mathbf{v}|$ y comparación cualitativa con Maxwell–Boltzmann; choques/tiempo y discusión de presión; conservación de energía total; contraste gas diluido vs. denso.

Análisis físico (documento LaTeX) Objetivos, método numérico, validación básica (convergencia cualitativa, conservación de magnitudes), resultados (gráficas/tablas), discusión y conclusiones.

Problema 2 (Elegir uno o proponer equivalente)

Implemente en C++ **uno** de los siguientes sistemas acoplados no lineales (o proponga uno equivalente con las mismas exigencias de complejidad). Use integración robusta (sugerido: *Runge–Kutta 4*) y provea **scripts** para generar y visualizar resultados.

Opción A: Osciladores de Duffing acoplados

Dos osciladores de Duffing acoplados por un resorte lineal:

$$m\ddot{x} + c\dot{x} + \alpha x + \beta x^3 = F_0 \cos(\omega t) - k(x - x_{\text{otro}}).$$

Tareas sugeridas: sincronización/desincronización al variar k; espacios fase (x, v); respuesta al forzamiento F_0; diagramas de bifurcación en un parámetro.

Opción B: Circuitos de Chua acoplados

Dos sistemas de Chua acoplados (acoplo en x):

$$\dot{x}_i = \alpha(y_i - x_i - f(x_i)) + \kappa(x_j - x_i), \quad \dot{y}_i = x_i - y_i + z_i, \quad \dot{z}_i = -\beta y_i,$$

$$f(x) = m_1 x + \tfrac{1}{2}(m_0 - m_1)(|x + 1| - |x - 1|).$$

Tareas sugeridas: atractores individuales y acoplados; sincronización caótica al variar κ; bifurcaciones vs. α, β.

Opción C: Van der Pol acoplados

Osciladores de Van der Pol con acoplamiento lineal:

$$\ddot{x}_i - \mu(1 - x_i^2)\dot{x}_i + \omega_0^2 x_i = k(x_j - x_i).$$

Tareas sugeridas: sincronización en fase/frecuencia; diagramas de Lissajous x_1 vs. x_2; variación de μ.

Opción D: Péndulos acoplados con interacción no lineal cuadrática

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

Tareas sugeridas: modos colectivos y transferencia de energía; trayectorias en espacio fase (θ, ω); comparación con el caso lineal.

Análisis físico (documento LaTeX) Definir régimen (parámetros), validar integración (paso de tiempo), caracterizar sincronización/caos (mapa de Poincaré, espectro de potencia, exponentes de Lyapunov si aplica), discutir resultados.

Requisitos mínimos de proyecto (estructura y reproducibilidad)

1. **Diagrama de flujo** del programa (PDF/PNG en `documents/`).
2. **Build**: `Makefile` *o* `CMakeLists.txt` funcional.
3. **README** claro: cómo compilar, correr, reproducir figuras y animaciones.
4. **Doxygen**: uso estándar de *Doxygen* con `Doxyfile`.
5. **Estructura de carpetas (mínima):**
   - `include/` (cabeceras y librerías propias).
   - `src/` (`.cpp` principales y auxiliares).
   - `scripts/` (Python, Gnuplot, Octave, SFML u otros para graficación/animación).
   - `results/` (gráficas, GIFs, listas de datos, etc.).
   - `analysis/` (notebooks o guiones de análisis numérico opcional).
   - `documents/` (análisis físico en LaTeX y PDF listo para compilar).
6. **Estilo de código C++**: seguir *Google C++ Style Guide*: `https://google.github.io/styleguide/cppguide.html`
7. **Repositorio**: GitHub o GitLab con historial de commits del equipo.
8. **Defensa**: presentación corta (5–8 diapositivas) con objetivos, diseño, validación, resultados y conclusiones.

Criterios de evaluación (sugeridos)

- Correctitud física y numérica (modelado, integración, validaciones): 35 %.
- Calidad del software (POO, modularización, build, estilo, Doxygen): 30 %.
- Resultados y análisis (figuras, tablas, discusión comparativa): 25 %.
- Reproducibilidad y presentación (repo, scripts, README, defensa): 10 %.