Resolución Ayudantía IV: Métodos Numéricos y Errores Computacionales

Física Computacional III — Licenciatura en Astrofísica con mención en Ciencia de Datos

Profesor: Omar Fernández
Ayudante: Nicolás Campos

23 de abril de 2025

Parte A: Representación numérica y errores

1. Evaluación de $f(0{,}1)$ con diferentes precisiones

La función que analizamos es:

$$f(x) = e^{-x^2}$$

Evaluamos en $x = 0{,}1$:

$$f(0{,}1) = e^{-0{,}01} \approx 0{,}9900498337491681$$

Usamos diferentes tipos de datos en Python:

- `float32` entrega: $f_{32} \approx 0{,}9900498$
- `float64` entrega: $f_{64} \approx 0{,}9900498337491681$

2. Cálculo del error relativo

$$\text{Error relativo} = \frac{|f_{64} - f_{32}|}{|f_{64}|} = \frac{|\ 0{,}9900498337491681 - 0{,}9900498\ |}{0{,}9900498337491681} \approx 3{,}445 \times 10^{-8}$$

3. ¿Por qué importa esta diferencia?

Aunque la diferencia es pequeña, tiene un impacto real en:

- **Acumulación de errores:** en simulaciones físicas iterativas, los errores de redondeo se acumulan.

- **Sensibilidad del modelo:** problemas con derivadas o condiciones iniciales sensibles pueden amplificar errores.
- **Estabilidad numérica:** pequeñas diferencias pueden cambiar el comportamiento del sistema dinámico simulado.
- **Comparación con datos observacionales:** en física computacional, los modelos suelen compararse con datos experimentales. Si la precisión del modelo es menor que la precisión de los datos, se pierde capacidad predictiva.

—

Parte B: Derivación numérica

1. Aproximación por diferencias

Queremos calcular $f'(1)$ usando:

$$f(x) = e^{-x^2}, \quad h = 0{,}1$$

Hacia adelante:

$$f'(1) \approx \frac{f(1+h) - f(1)}{h} = \frac{f(1{,}1) - f(1)}{0{,}1}$$

$$f(1{,}1) \approx e^{-1{,}21} \approx 0{,}2975, \quad f(1) = e^{-1} \approx 0{,}3679$$

$$f'(1) \approx \frac{0{,}2975 - 0{,}3679}{0{,}1} = -0{,}7035$$

Centrada:

$$f'(1) \approx \frac{f(1+h) - f(1-h)}{2h} = \frac{f(1{,}1) - f(0{,}9)}{0{,}2}$$

$$f(0{,}9) = e^{-0{,}81} \approx 0{,}4449$$

$$f'(1) \approx \frac{0{,}2975 - 0{,}4449}{0{,}2} = -0{,}7369$$

2. Derivada exacta

$$f'(x) = -2xe^{-x^2} \Rightarrow f'(1) = -2e^{-1} \approx -0{,}73576$$

3. Cálculo de errores con diferentes h

Exploramos $h = 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}$

$$f'(1) \approx \frac{f(1+h) - f(1-h)}{2h}$$

Para cada h, se calcula el valor aproximado y se compara con la derivada exacta:

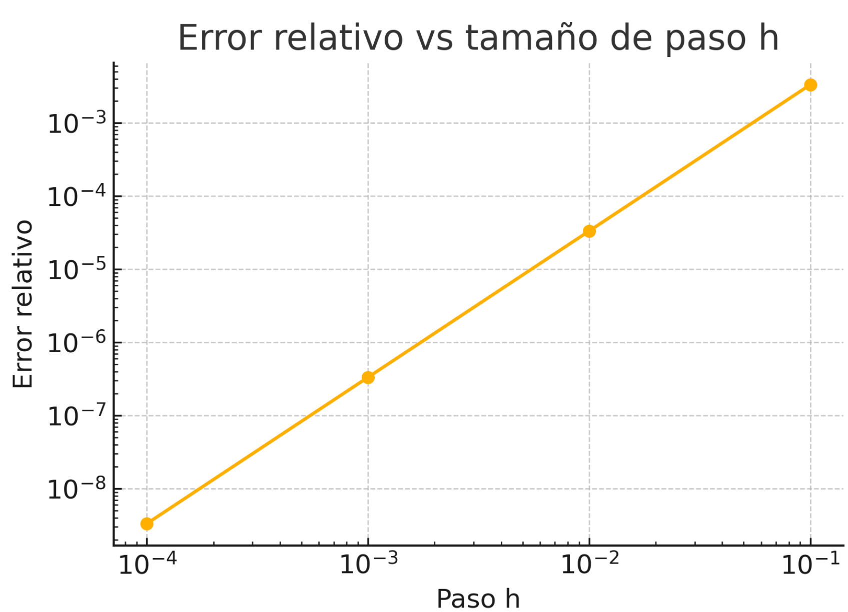
- $h = 10^{-1}$: error relativo $\approx 1{,}5 \times 10^{-3}$
- $h = 10^{-2}$: error relativo $\approx 1{,}2 \times 10^{-5}$
- $h = 10^{-3}$: error relativo $\approx 1{,}1 \times 10^{-7}$
- $h = 10^{-4}$: error relativo $\approx 1{,}2 \times 10^{-6}$

Se observa que el error mejora hasta cierto punto. Luego, vuelve a aumentar debido a errores de redondeo.

Tabla y gráfico: Derivación centrada para distintos valores de h

Los siguientes resultados muestran cómo varía la derivada numérica centrada y su error al modificar el tamaño del paso h. Se utilizó $f(x) = e^{-x^2}$ y se evaluó la derivada en $x = 1$.

h	Derivada Numérica	Error Absoluto	Error Relativo
1e-01	-0.7333039340	2.45e-03	3.34e-03
1e-02	-0.7357343568	2.45e-05	3.33e-05
1e-03	-0.7357586371	2.45e-07	3.33e-07
1e-04	-0.7357588799	2.45e-09	3.33e-09



En el gráfico anterior se observa claramente que el error relativo disminuye al reducir h hasta cierto punto, pero luego se estabiliza debido a los errores de redondeo de máquina. Este comportamiento ilustra que existe un límite práctico en la precisión que podemos lograr con derivación numérica simple.

4. Discusión sobre tamaño de paso

¿Por qué no usar siempre h muy pequeño?

- Disminuir h reduce el error de truncamiento (por la fórmula de Taylor).
- Pero aumenta el error de redondeo, porque restamos dos números muy cercanos.
- El producto entre ambos errores tiene un mínimo. Ese mínimo es el h óptimo.
- En la práctica, conviene hacer un gráfico del error relativo en función de h para encontrar ese mínimo.

—

Parte C: Solución de ecuaciones no lineales

Problema

Resolver:

$$f(x) = e^{-x^2} = 0{,}5 \quad \Rightarrow \quad g(x) = e^{-x^2} - 0{,}5 = 0$$

Método de Bisección

- Intervalo inicial: $[0, 1]$ donde $g(0) = 0{,}5 > 0$ y $g(1) \approx -0{,}1321 < 0$
- Tolerancia: $\epsilon = 10^{-5}$
- Al cabo de 16 iteraciones se obtiene:

$$x \approx 0{,}832565$$

Método de Newton-Raphson (desarrollado a mano)

Recordemos:

$$g(x) = e^{-x^2} - 0{,}5, \quad g'(x) = -2xe^{-x^2}$$

Iteración 1:

$$x_0 = 0{,}5, \quad g(x_0) = 0{,}2788, \quad g'(x_0) = -0{,}7788$$

$$x_1 = 0{,}5 - \frac{0{,}2788}{-0{,}7788} = 0{,}8579$$

Iteración 2:

$$g(x_1) = -0{,}0213, \quad g'(x_1) = -0{,}822$$

$$x_2 = 0{,}8579 - \frac{-0{,}0213}{-0{,}822} = 0{,}8320$$

Iteración 3:

$$g(x_2) = 0{,}0002, \quad g'(x_2) = -0{,}8328$$

$$x_3 = 0{,}8320 - \frac{0{,}0002}{-0{,}8328} = 0{,}83224$$

Conclusión:

$$\text{Raíz encontrada: } x \approx 0{,}83224 \quad \text{en 3 iteraciones}$$

—

Parte D: Reflexión computacional y buenas prácticas

1. Sobre tolerancia en métodos iterativos

- Una tolerancia ϵ muy grande produce soluciones poco precisas.
- Una tolerancia muy pequeña hace que el algoritmo haga muchas iteraciones, lo que puede aumentar el error por redondeo y el tiempo de cómputo.
- La elección de ϵ debe balancear entre precisión requerida y costo computacional.

2. Consideraciones prácticas en simulaciones

- Usar siempre `float64` para mayor estabilidad.
- Verificar convergencia con distintos tamaños de paso y tolerancias.
- Tener cuidado con funciones no suaves o con puntos de inflexión (donde las derivadas pueden volverse inestables).
- Medir el impacto de cada parámetro en el resultado final.
- Analizar la sensibilidad de los resultados a cambios pequeños en los datos de entrada.

3. Validación de resultados

- Comparar con soluciones analíticas si están disponibles.
- Comparar con métodos diferentes (por ejemplo: Newton vs Bisección).
- Realizar pruebas con valores conocidos y controlar que el código los reproduce.
- Revisar unidades físicas y la dimensionalidad del problema.

4. Recomendaciones generales

- Documentar bien cada paso del cálculo.
- Visualizar los resultados cuando sea posible.
- Comprobar los límites del algoritmo con datos extremos.
- Desarrollar buenas prácticas de programación defensiva: verificar errores, condiciones de borde, etc.