

# Proyecto Final

**Implementación:** RK2 y RK4

**Lenguaje:** Python con NumPy y Matplotlib. Alternativa: MATLAB.

## 1) Métodos a implementar y lenguaje

- Métodos: Runge–Kutta de 2º orden (RK2 o Heun) y Runge–Kutta de 4º orden (RK4).
- Lenguaje: Python. Se entregará código modular con funciones de paso, un driver común y gráficos.

## 2) Conjunto de validación con solución analítica (con C.I.)

Todas las ecuaciones pueden tomarse del Zill 11ª edición. Se fijan condiciones iniciales explícitas para comparar.

### 2.1 Ecuaciones de primer orden

- 1) Logística

$$( = P(a - bP), P(0) = 0.20, a = 2, b = 1 )$$

- 2) Lineal de primer orden

$$( y' + 2xy = x^3, y(0) = 0 )$$

### 2.2 Ecuaciones de segundo orden no homogéneas

Trabajaremos cualquiera de estas dos opciones.

- 1) Opción A

$$( y'' - 4y = e^{\{-t\}}, y(0) = 1, y'(0) = 0 )$$

- 2) Opción B

$$( y'' - 2y' + 5y = t, y(0) = 0, y'(0) = 1 )$$

### 2.3 Sistemas lineales 2×2

- 1) Sistema 1

$$( x' = -x + 2y, y' = -3x - 4y, (x(0), y(0)) = (1, 0) )$$

- 2) Sistema 2

$$( x' = -x, y' = -2y, (x(0), y(0)) = (1, 1) )$$

### 3) Sistema 2×2 no lineal para resolver numéricamente

Van der Pol en forma de sistema

$$\begin{aligned} (x' &= y) \\ (y' &= (1 - x^2), y - x) \end{aligned}$$

Parámetro: ( $\mu = 1$ )

Condiciones iniciales: ( $x(0) = 2, y(0) = 0$ )

Se resolverá con RK2 y RK4. Se reportarán series temporales y trayectoria en el plano fase.

### 4) Uso de IA como copiloto y evidencia

- Generar y depurar funciones de paso para RK2 y RK4 y un driver común.
- Documentar con docstrings y anotar supuestos y limitaciones.
- Bitácora de uso: prompts clave, respuestas y decisiones aplicadas con capturas.
- Pruebas cortas mostrando que al reducir el paso ( $h$ ) el error baja acorde al orden del método.

### 5) Entregables inmediatos

- Código base de RK2 y RK4 con pruebas en los casos analíticos.
- Tablas de error para varios ( $h$ ) y gráficas comparativas contra la solución exacta.
- Caso Van der Pol con gráficas y una discusión breve.