



UNIVERSIDAD POLÍTÉCNICA  
DE AGUASCALIENTES

Universidad Politécnica de Aguascalientes  
Ingeniería en Sistemas Computacionales

Materia: Matemáticas para ingeniería II  
Docente: Isaac Vázquez Mendoza

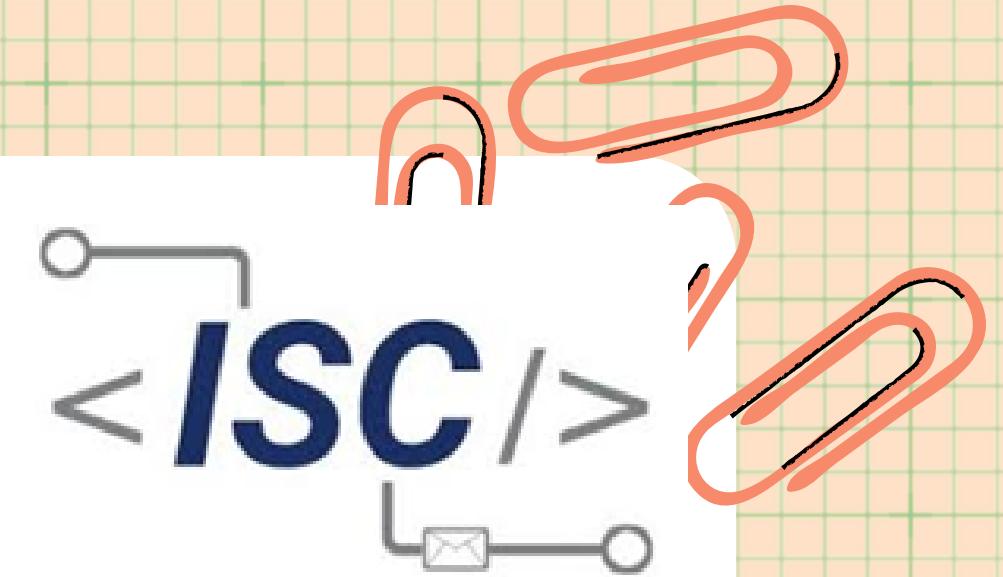
Equipo 4

## Método Runge-Kutta 4

Integrantes:

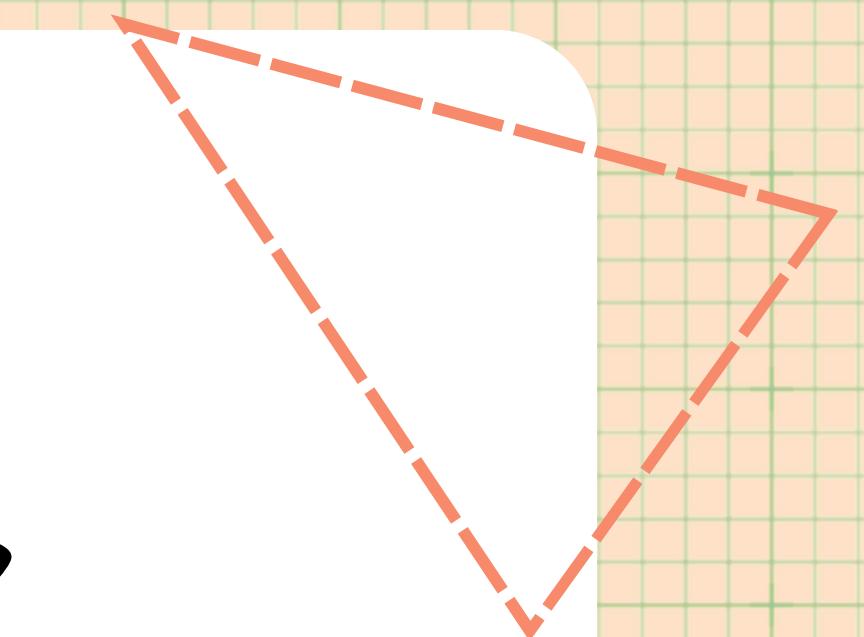
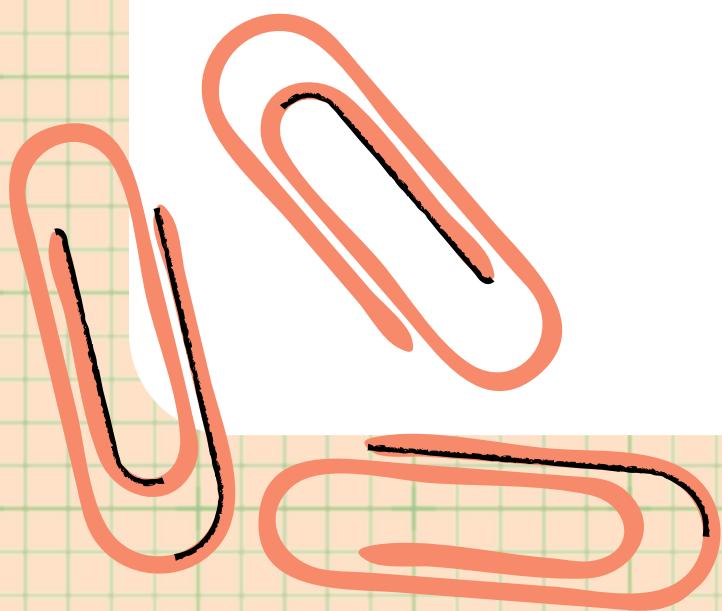
Vázquez Ríos Juan Ramon UP230190  
Olmedo Cruz Danna Giselle UP230183  
Surdez Espeleta Andrea Mariana UP230188  
Martin Cornejo Diego Ernesto UP230346  
López Nava Alberto Misael UP239756

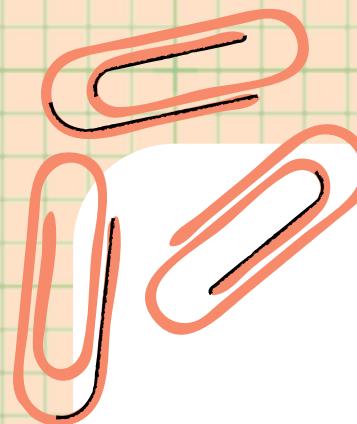
Grado y Grupo: ISC06A



# Introducción

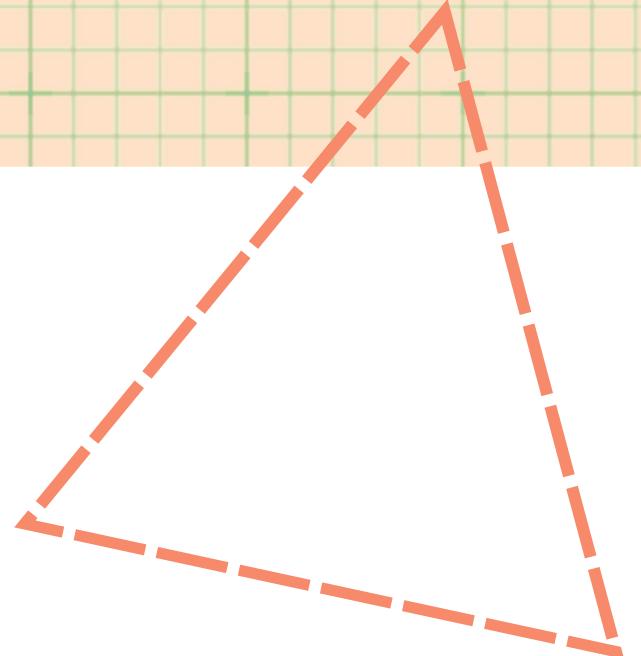
---



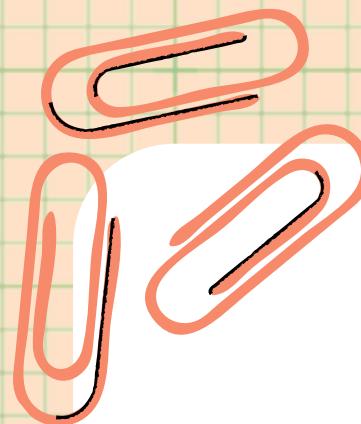


# ¿Por qué necesitamos métodos numéricos?

-----

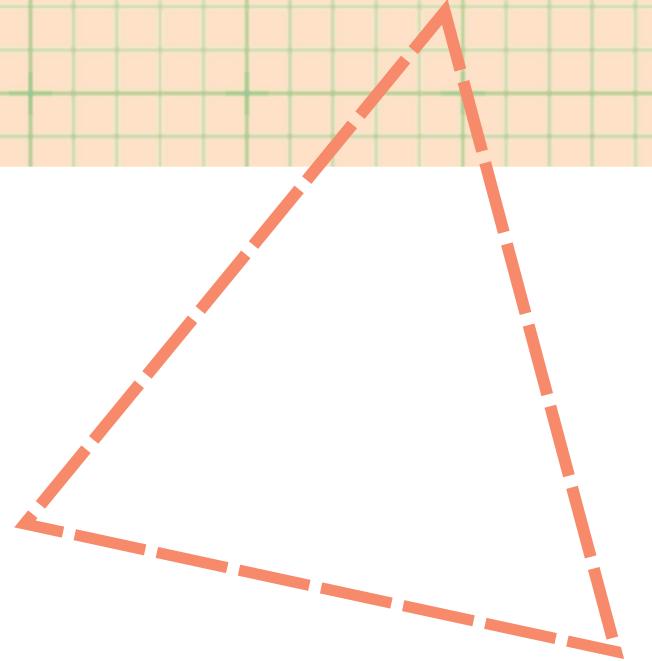


- Muchas ecuaciones diferenciales no tienen solución analítica.
- Incluso cuando la tienen, pueden ser muy complejas o costosas de evaluar.
- Los métodos numéricos nos permiten aproximar la solución en puntos discretos.
- RK4 es el más popular por su excelente equilibrio entre precisión y eficiencia.



## Formulación del problema

---

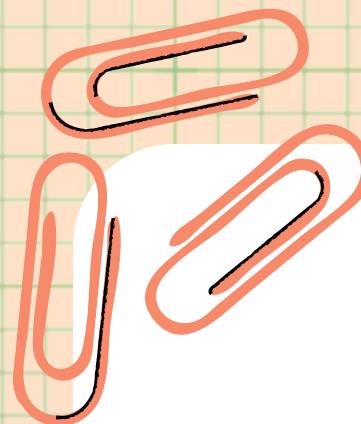


Problema de valor inicial (PVI):

$$\frac{dx}{dt} = f(t, x) \quad x(t_0) = x_0$$

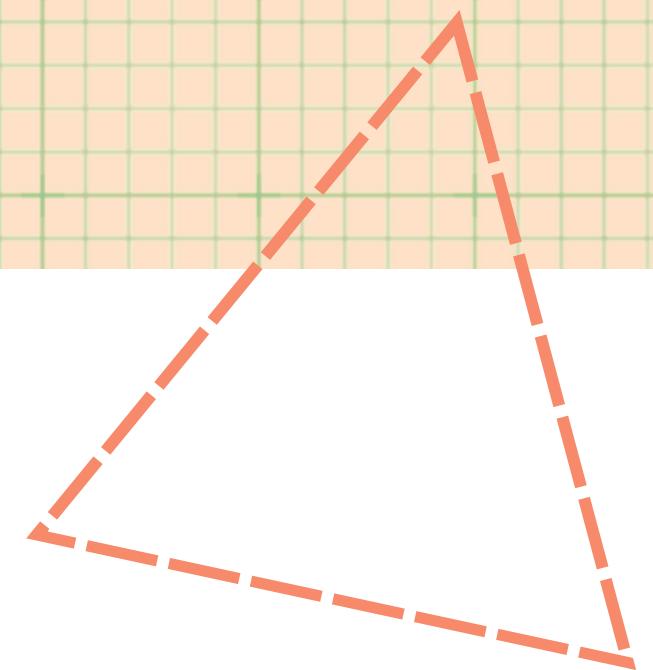
Objetivo:

Encontrar valores aproximados de  $x(t)$  en puntos  $t_0, t_1, t_2, \dots, t_n$  usando un paso  $h$  constante.



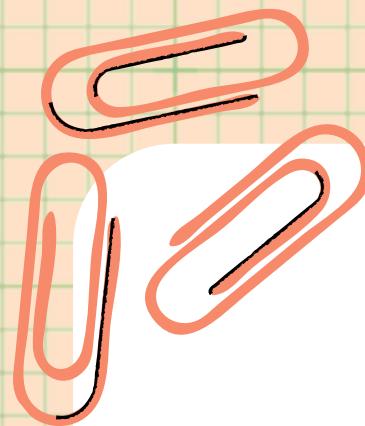
## Idea detrás de R-K4

---



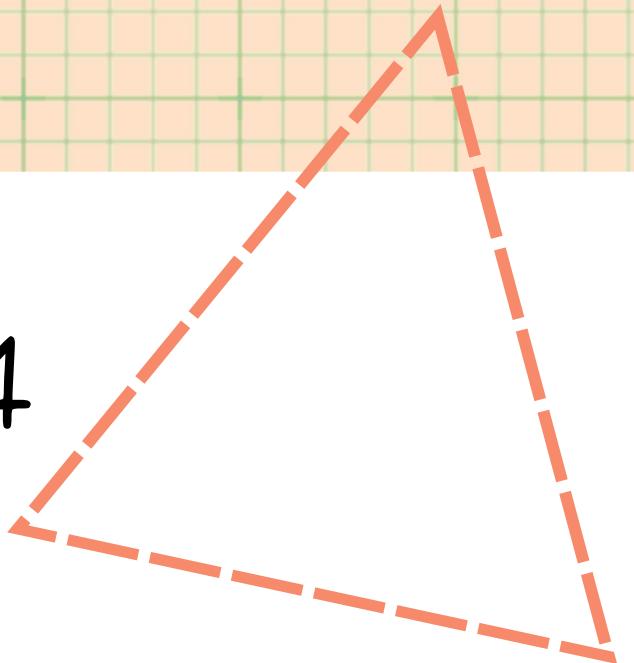
- El método de Euler usa una sola pendiente (al inicio del intervalo).
- RK4 mejora esto evaluando cuatro pendientes en distintos puntos del intervalo:
  - a. Al inicio.
  - b. En el punto medio (usando predicción de Euler).
  - c. Otra vez en el punto medio (con mejor predicción).
  - d. Al final del intervalo.
- Luego, combina estas pendientes en un promedio ponderado.

(Esto permite una aproximación muy cercana a la curva real).



## Fórmulas del método RK4

---



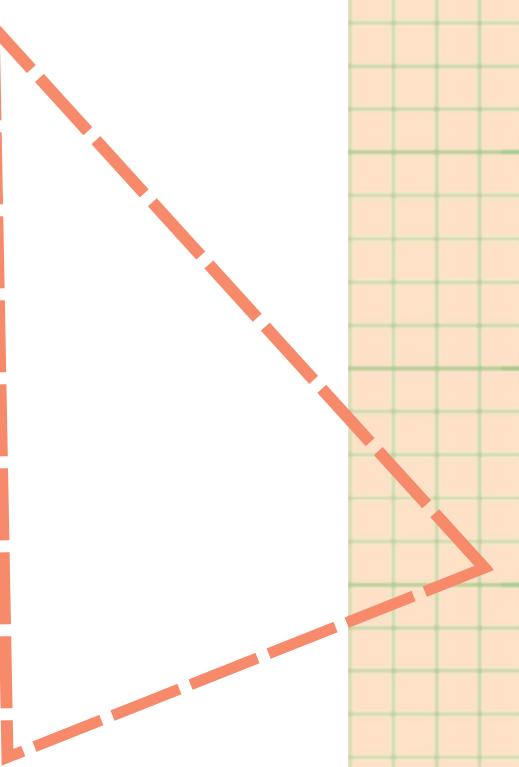
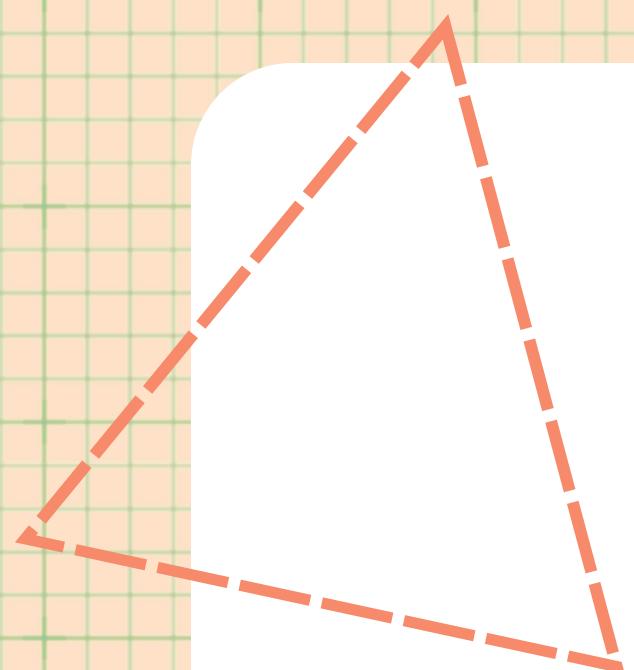
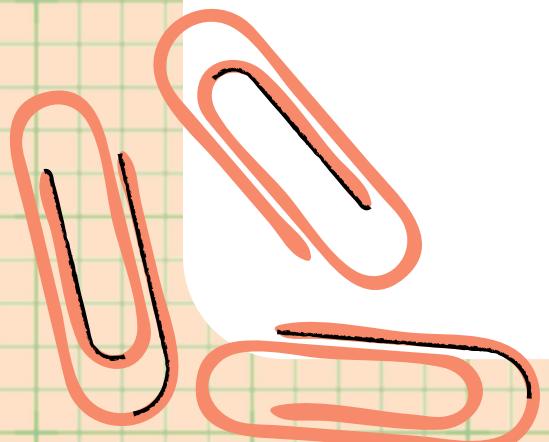
- $K_1 = h \cdot f(t_K, x_K)$
- $K_2 = h \cdot f(t_K + h/2, x_K + K_1/2)$
- $K_3 = h \cdot f(t_K + h/2, x_K + K_2/2)$
- $K_4 = h \cdot f(t_K + h, x_K + K_3)$
- $x_{K+1} = x_K + 1/6(K_1 + 2K_2 + 2K_3 + K_4)$
- $t_{K+1} = t_K + h$

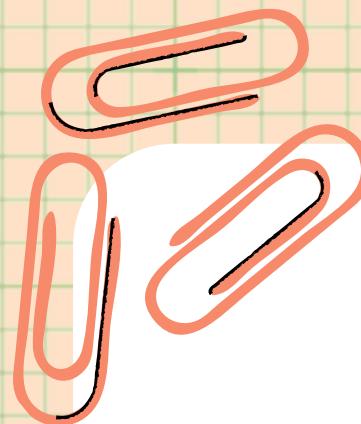
Orden del método: 4

Error global: proporcional a  $h^4 \rightarrow$  muy preciso incluso con pasos moderados.

# Ejemplo Práctico

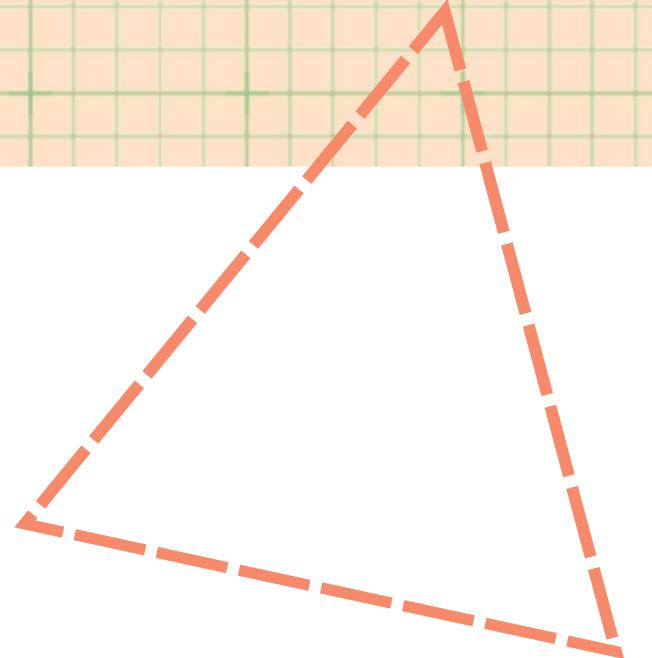
---





## Formulación del problema

---



Problema:

$$\frac{dx}{dt} = t - x, \quad x(0) = 1, \quad h = 0.1$$

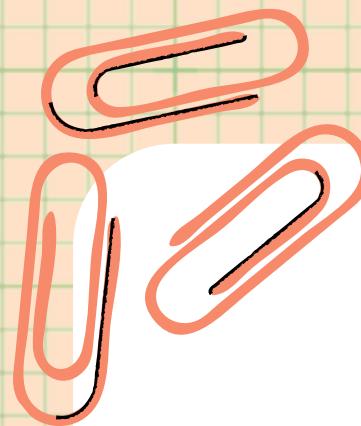
Calcular  $x_1$ :

$$1. K_1 = 0.1 \cdot f(0, 1) = 0.1 \cdot (0 - 1) = -0.1$$

$$2. K_2 = 0.1 \cdot f(0.05, 1 - 0.05) = 0.1 \cdot (0.05 - 0.95) = -0.09$$

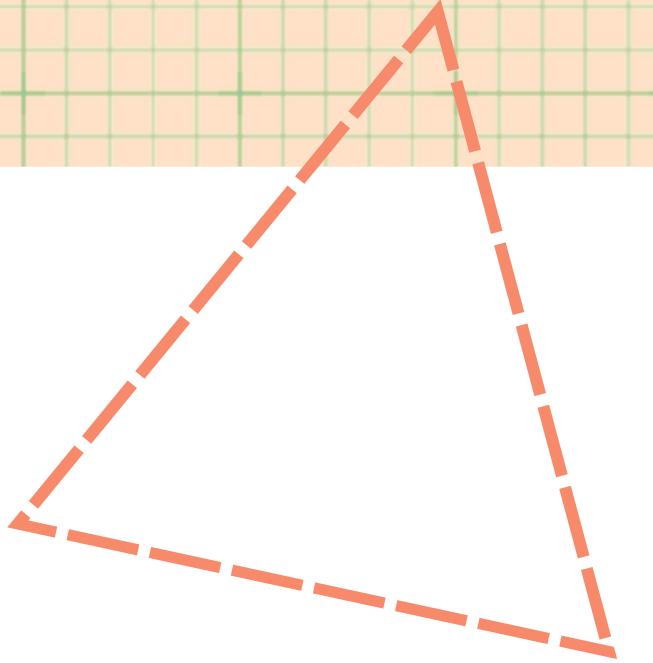
$$3. K_3 = 0.1 \cdot f(0.05, 1 - 0.045) = 0.1 \cdot (0.05 - 0.955) = -0.0905$$

$$4. K_4 = 0.1 \cdot f(0.1, 1 - 0.0905) = 0.1 \cdot (0.1 - 0.9095) = -0.08095$$



## Formulación del problema

-----

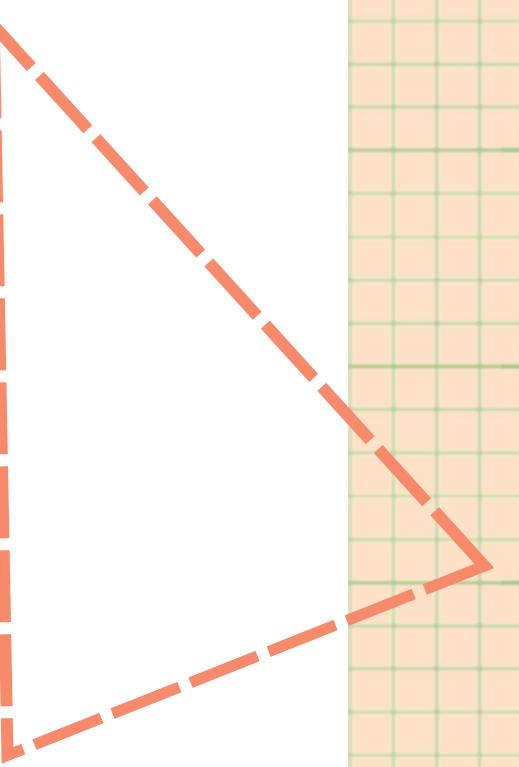
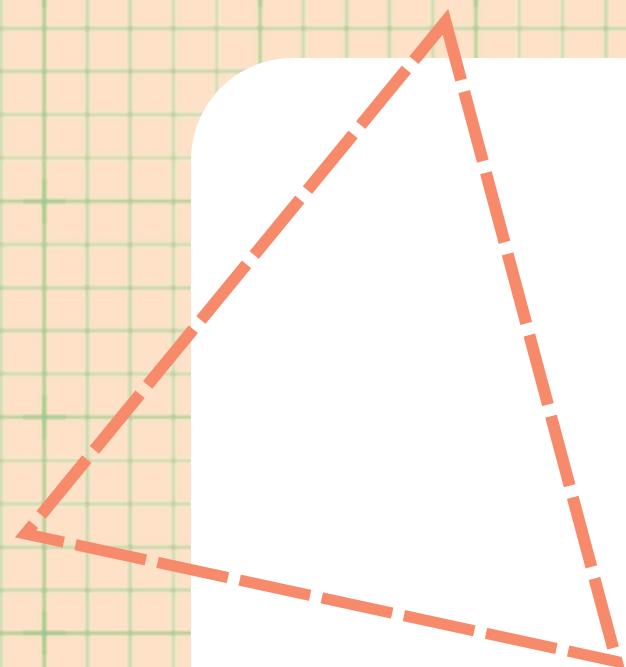
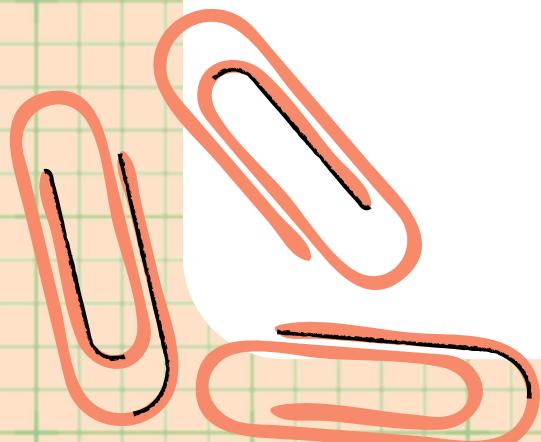


Resultado:

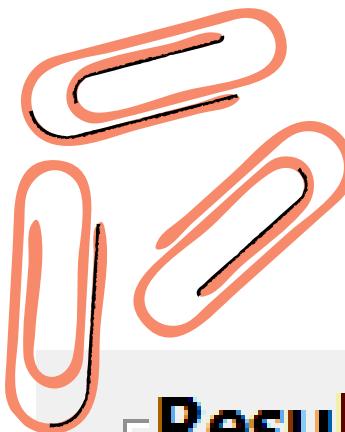
$$x_1 = 1 + 16(-0.1 + 2(-0.09) + 2(-0.0905) + (-0.08095)) \approx 0.9098$$

# Demostración Gráfica

---



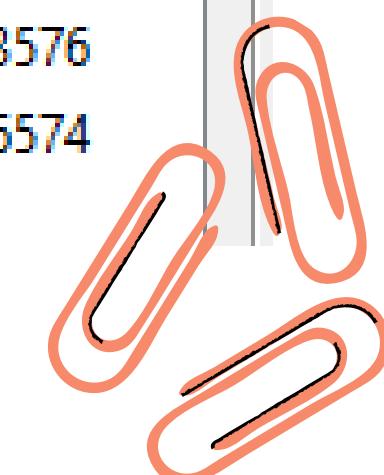
# Método Runge-Kutta 4° — Equipo 4



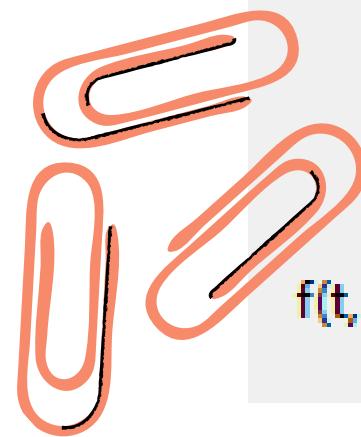
$f(t, x) =$   t<sub>0</sub>:  x<sub>0</sub>:  h:

## Resultados

Iter	t <sub>k</sub>	x <sub>k</sub>	k1	k2	k3	k4	x <sub>(k+1)</sub>
0	0.00000	1.00000	-0.10000	-0.09000	-0.09050	-0.08095	0.90968
1	0.10000	0.90968	-0.08097	-0.07192	-0.07237	-0.06373	0.83746
2	0.20000	0.83746	-0.06375	-0.05556	-0.05597	-0.04815	0.78164
3	0.30000	0.78164	-0.04816	-0.04076	-0.04113	-0.03405	0.74064
4	0.40000	0.74064	-0.03406	-0.02736	-0.02770	-0.02129	0.71306
5	0.50000	0.71306	-0.02131	-0.01524	-0.01554	-0.00975	0.69762
6	0.60000	0.69762	-0.00976	-0.00427	-0.00455	0.00069	0.69317
7	0.70000	0.69317	0.00068	0.00565	0.00540	0.01014	0.69866
8	0.80000	0.69866	0.01013	0.01463	0.01440	0.01869	0.71314
9	0.90000	0.71314	0.01869	0.02275	0.02255	0.02643	0.73576
10	1.00000	0.73576	0.02642	0.03010	0.02992	0.03343	0.76574



# Método Runge-Kutta 4º — Equipo 4



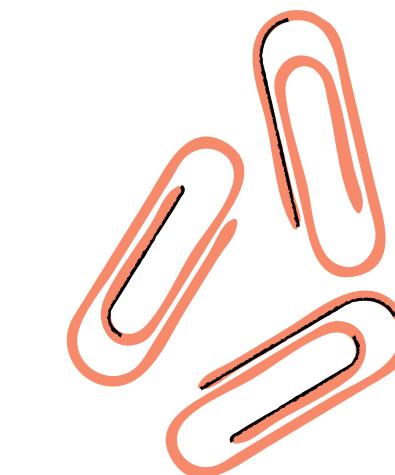
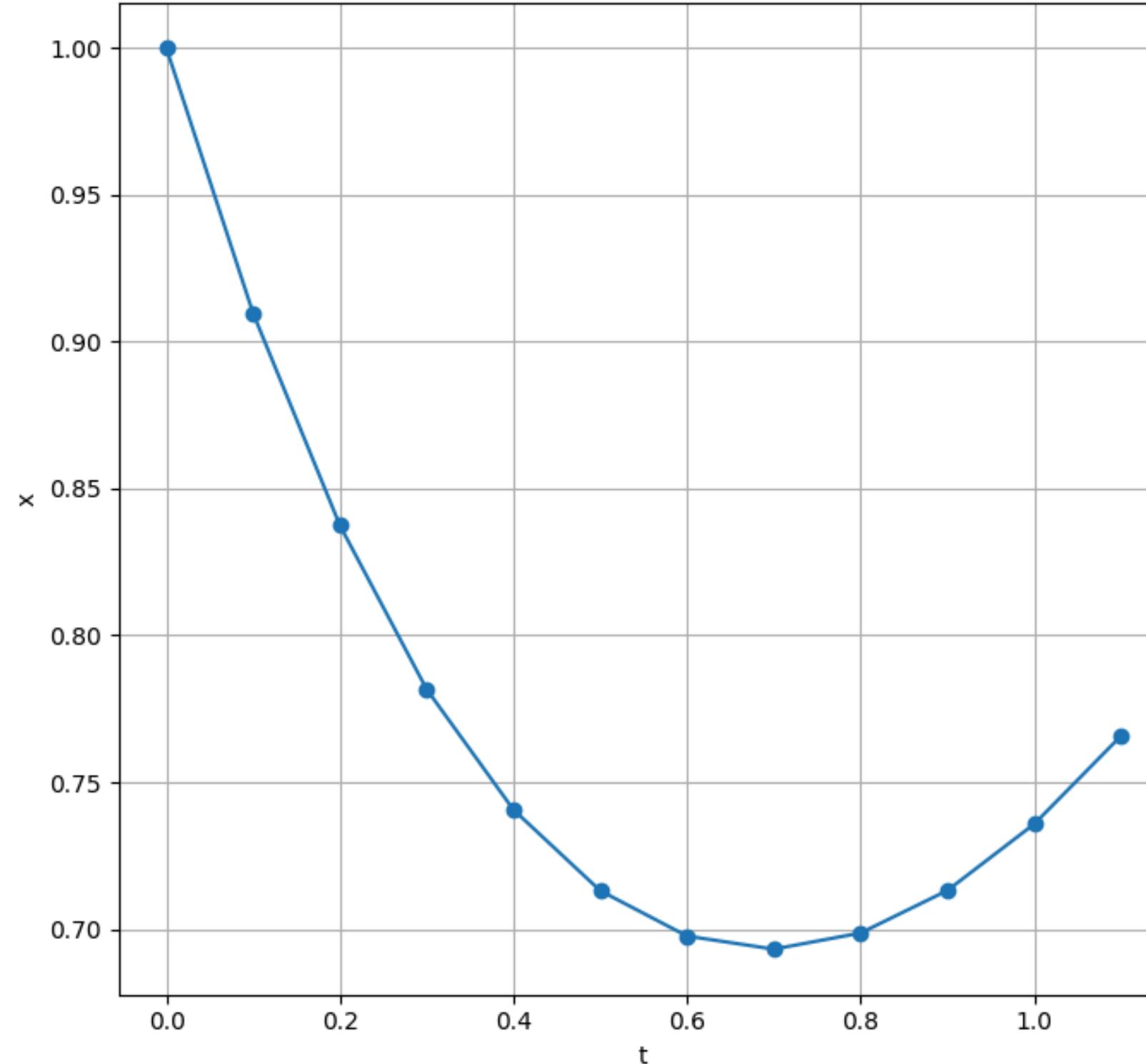
$f(t, x) =$

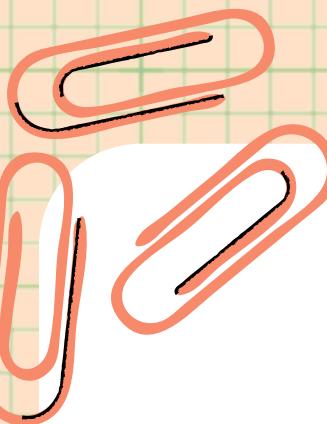
$t_0:$

$x_0:$

$h:$

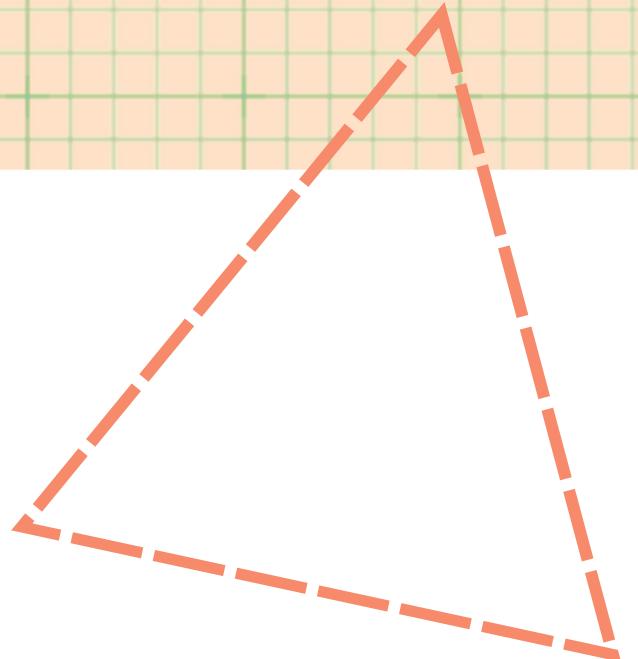
Runge-Kutta 4º Orden





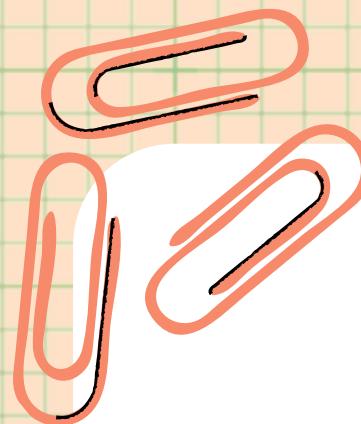
# Implementación en Python - Vista general

---



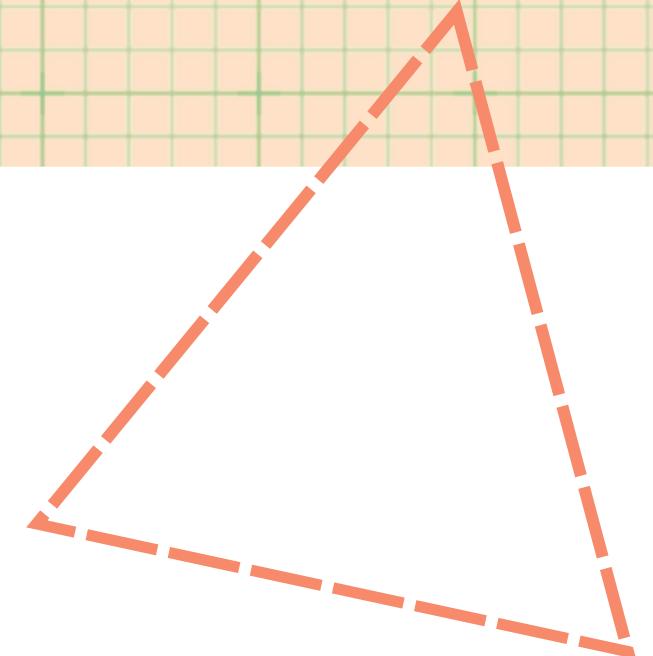
Nuestra aplicación permite:

- Ingresar cualquier función  $f(t,x)$  usando sintaxis matemática natural.
- Definir condiciones iniciales  $t_0, x_0$  y tamaño de paso  $h$ .
- Ver cada iteración paso a paso.
- Observar la gráfica de la solución inmediatamente.



# Cómo procesamos la función matemática

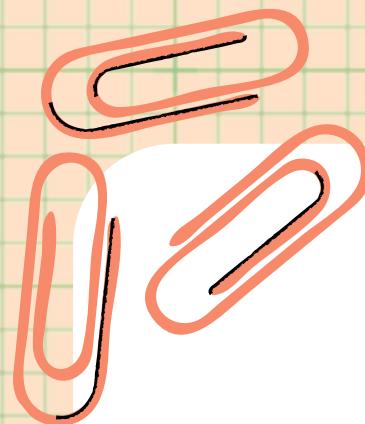
---



python

```
1 f_str = entry_funcion.get()          # Ej: "t - x"  
2 f_tx = sp.sympify(f_str)            # Convierte a expresión simbólica  
3 f = sp.lambdify((t, x), f_tx, "numpy") # → función ejecutable
```

- `sympify`: interpreta expresiones como  $\sin(t) + x^{**2}$ .
- `lambdify`: genera una función que Python puede evaluar numéricamente.
- Esto hace que el programa sea flexible y potente, sin necesidad de reescribir código.



# Código RK4

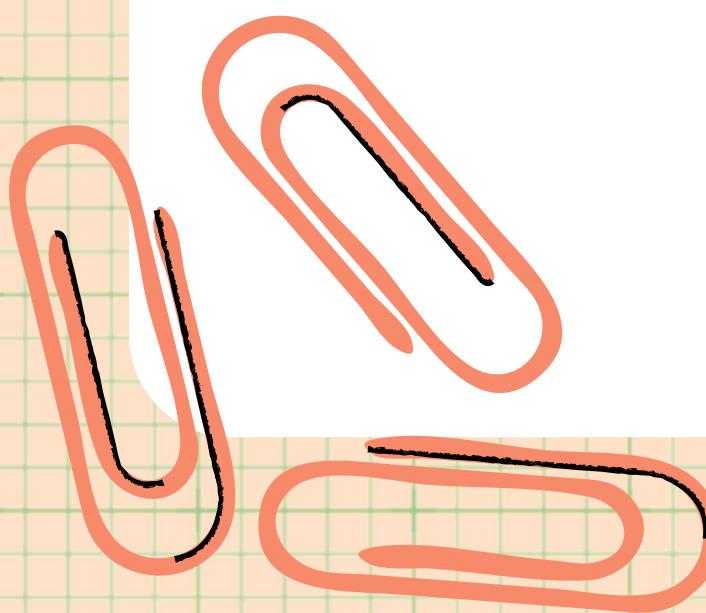
---

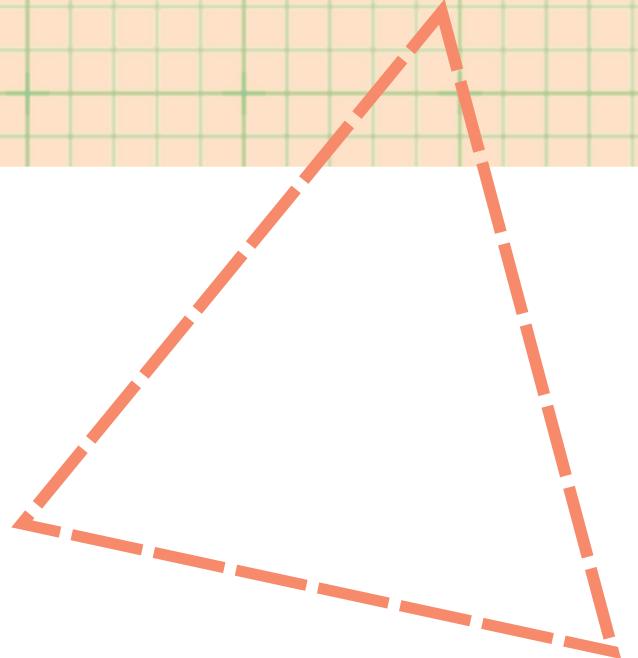
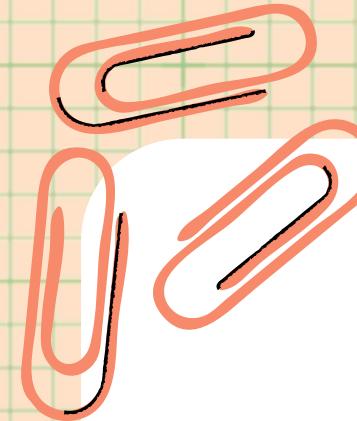
```
while (h > 0 and tk_ < tf) or (h < 0 and tk_ > tf):  
  
    k1 = h * f(tk_, xk)  
    k2 = h * f(tk_ + h/2, xk + k1/2)  
    k3 = h * f(tk_ + h/2, xk + k2/2)  
    k4 = h * f(tk_ + h, xk + k3)  
  
    x_next = xk + (k1 + 2*k2 + 2*k3 + k4) / 6  
    t_next = tk_ + h
```

- Cada línea del código corresponde exactamente a una fórmula matemática.
- El programa muestra todos los  $k_i$  en la tabla, para que el usuario entienda cómo se construye la solución.

# Conclusión

---





- El método de Runge-Kutta de cuarto orden es una herramienta poderosa y accesible para resolver EDOs.
- Nuestra aplicación traduce la teoría en experiencia visual e interactiva.
- Al mostrar cada paso del cálculo, refuerza la comprensión conceptual y numérica.