

# Análisis Numérico para Ingeniería

Clase Nro. 4

*“Ver lo que está delante  
de nuestros ojos requiere  
un esfuerzo constante.”*

*Eric Arthur Blair*

# Temas a tratar en esta clase :

- Sistemas de EDOs de 1er. Orden.
- Transformación de EDOs de orden  $n$ .
- Resolución de Sistemas de 1er. Orden.
- Estrategias de ajuste del paso  $h$ .



# Sistemas de EDOs de 1er. orden

- Infinidad de problemas de diversas áreas del conocimiento como la *física*, la *química*, la *biología*, la *economía*, la *ingeniería*, etc. se expresan a menudo en términos de **ecuaciones diferenciales ordinarias**.
- Estas ecuaciones pueden estar relacionadas entre sí, por lo que conforman **sistemas de ecuaciones diferenciales**.
- Las ecuaciones diferenciales de **orden  $n$** , pueden transformarse en sistemas de ecuaciones de **primer orden**, para su resolución.

# Ejemplo de Sistemas de EDOs

El modelo **SIR**, desarrollado por el químico escocés **William Ogilvy Kernack** y el matemático **A.G. McKendrick**, modela la evolución de una enfermedad infecto-contagiosa.

El modelo divide a la población en individuos **susceptibles** de ser infectados, individuos **infectados** e individuos **recuperados** de la enfermedad.



## MODELO SIR

$$\frac{dS}{dt} = -\beta \cdot S(t) \cdot I(t)$$

$$\frac{dI}{dt} = \beta \cdot S(t) \cdot I(t) - \gamma \cdot I(t)$$

$$\frac{dR}{dt} = \gamma \cdot I(t)$$



# Ejemplo de Sistemas de EDOs

En el año 1978 en la revista científica British Medical Journal se publicaron los datos de un brote de gripe en un internado del norte de Inglaterra que se extendió del 22 de enero al 4 de febrero, infectando a 512 de las 763 personas que allí estudiaban.

Debido a que en este caso concreto se cuenta con datos reales y la cantidad de individuos permanece constante, es un escenario ideal para corroborar la exactitud del modelo SIR para predecir la evolución de la infección.

Precisamente para este caso en particular se han determinado los siguientes valores:  $\gamma = 0.4477$  y  $\beta = 0.0022$ .

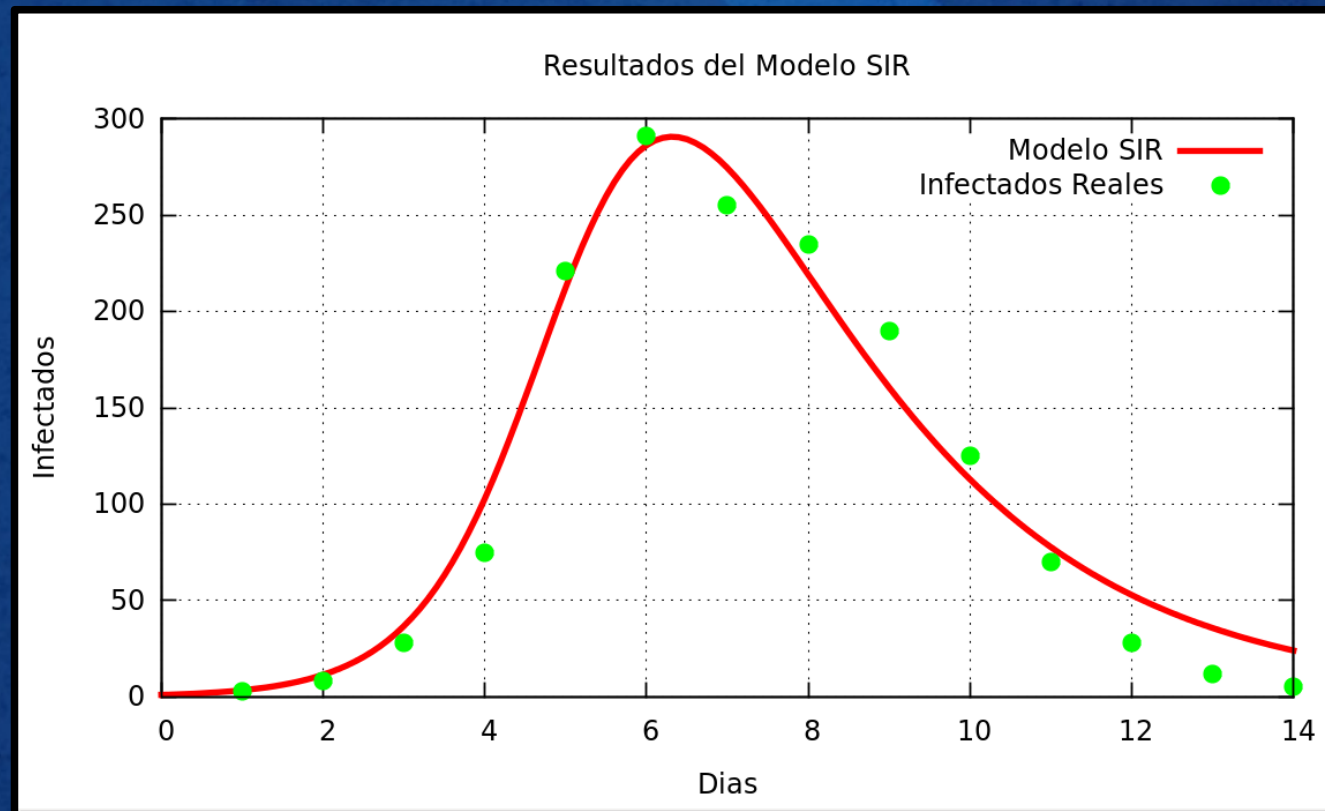
# Ejemplo de Sistemas de EDOs

En este caso los valores iniciales son:

Individuos Suceptibles:  $S_0 = 762$ ,

Individuos Infectados:  $I_0 = 1$ ,

Individuos Recuperados:  $R_0 = 0$

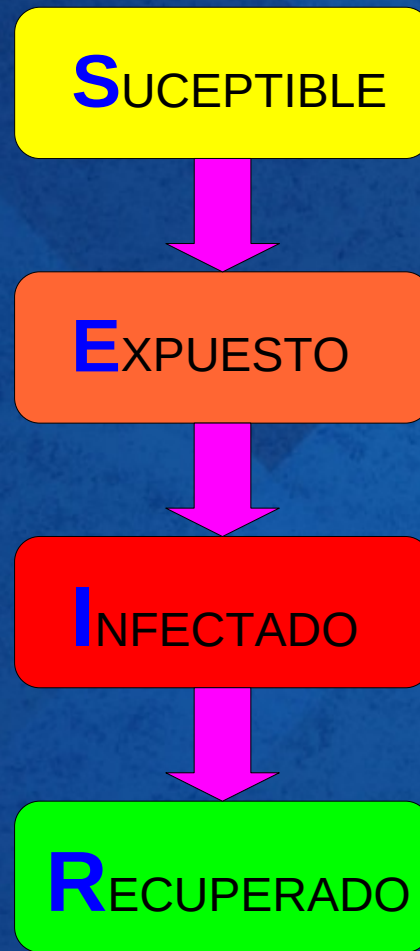




# Modelo SEIR

El modelo **SEIR**, es una adaptación del modelo **SIR** explicado anteriormente

El modelo divide a la población en individuos **susceptibles** de ser infectados, individuos **expuestos**, individuos **infectados** e individuos **recuperados** de la enfermedad.



## MODELO SEIR

$$\frac{dS}{dt} = -\beta \cdot S(t) \cdot I(t) / N$$

$$\frac{dE}{dt} = \beta \cdot S(t) \cdot I(t) / N - \sigma \cdot E(t)$$

$$\frac{dI}{dt} = \sigma \cdot E(t) - \gamma \cdot I(t)$$

$$\frac{dR}{dt} = \gamma \cdot I(t)$$

# Modelo SEIR para el COVID-19

Para los que deseen saber más sobre el tema y quieran experimentar con estos modelos, les dejo el enlace a este Blog.



BLOG DEL INSTITUTO DE MATEMÁTICAS DE LA UNIVERSIDAD DE SEVILLA



# EDOs de orden n

Una **EDO** de orden **n** puede transformarse en un **sistema** de **n** ecuaciones de **primer orden**.

$$\frac{d^n y}{dx^n} = f\left(x, y, \frac{dy}{dx}, \frac{d^{(2)} y}{dx^2}, \dots, \frac{d^{(n-1)} y}{dx^{n-1}}\right)$$

$$y|_{x=x_0} = y_0$$

$$\left. \frac{dy}{dx} \right|_{x=x_0} = y'_0$$

$$\left. \frac{d^{(2)} y}{dx^2} \right|_{x=x_0} = y''_0$$

.....

$$\left. \frac{d^{(n-1)} y}{dx^{n-1}} \right|_{x=x_0} = y_0^{(n-1)}$$

**CONDICIONES  
INICIALES**



# Algoritmo de transformación

Las EDOs de orden  $n$  se transforman en sistemas de  $n$  ecuaciones de primer orden para poder resolverlas con los métodos ya vistos.

$$\frac{d^n y}{dx^n} = f\left(x, y, \frac{dy}{dx}, \frac{d^{(2)}y}{dx^2}, \dots, \frac{d^{(n-1)}y}{dx^{n-1}}\right) = f(x, y, p, q, \dots, w)$$

$$\frac{dy}{dx} = y' = p$$

$$\frac{d^{(2)}y}{dx^2} = y'' = p' = q$$

.....

$$\frac{d^{(n-1)}y}{dx^{n-1}} = z' = w$$

Sustituimos cada derivada por una nueva variable.



# Ejemplo (I)

Ecuación diferencial de **orden 3**

$$y''' + x \cdot y'' - x \cdot y' - 2 \cdot y = x$$


$$x_0 = 0 \quad ; \quad y_{x_0} = y''_{x_0} = 0 \quad ; \quad y'_{x_0} = 1$$

Sistema de **3 ecuaciones** diferenciales de **1er. orden**

$$y' = p$$

$$p' = q$$

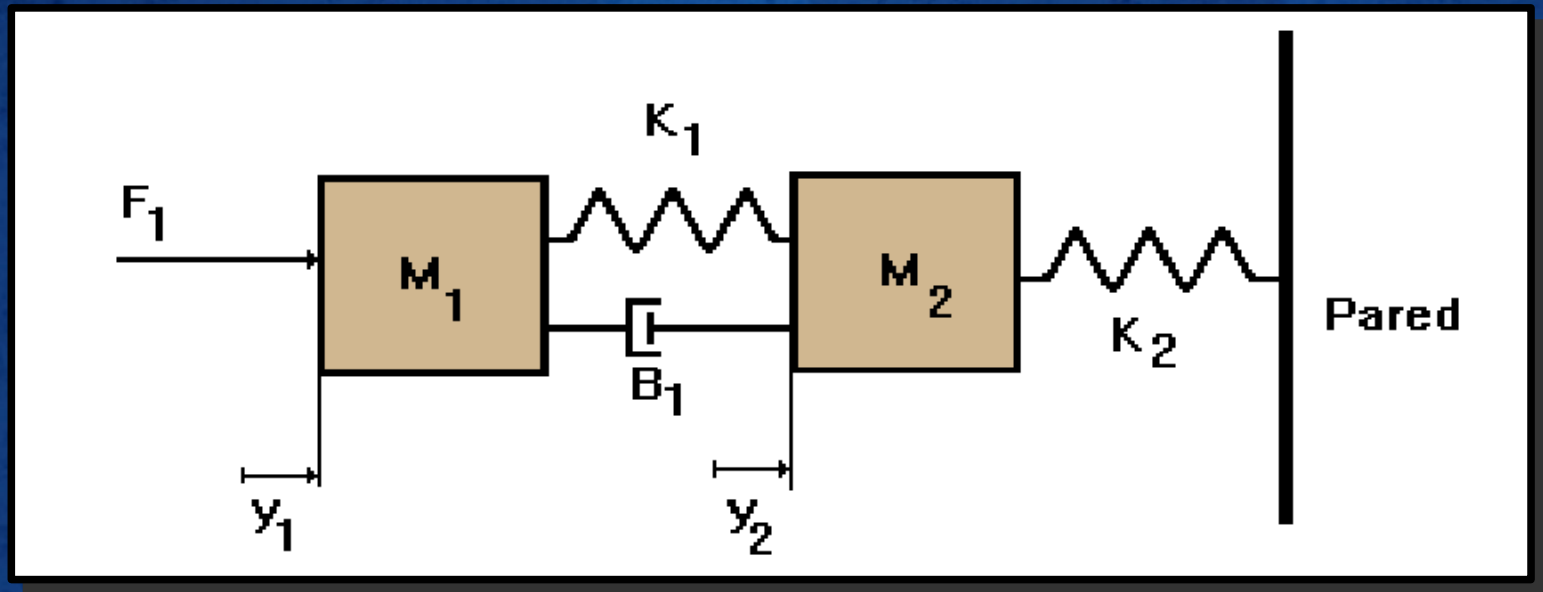
$$q' = x - x \cdot q + x \cdot p + 2 \cdot y$$



Sustituimos cada derivada por una nueva variable.

# Resolución completa de un problema

## Sistema Masa – Resorte - Amortiguador



Sistema de 2 ecuaciones diferenciales de orden 2

$$\begin{aligned} M_1 \cdot y''_1 + B_1 \cdot y'_1 + k_1 \cdot y_1 - B_1 \cdot y'_2 - k_2 \cdot y_2 &= F_1(t) \\ -B_1 \cdot y'_1 - k_1 \cdot y_1 + M_2 \cdot y''_2 + B_1 \cdot y'_2 + (k_1 + k_2) \cdot y_2 &= 0 \end{aligned}$$



# Transformación del Ejemplo (II)

Sistema de 2 ecuaciones diferenciales de orden 2

$$\begin{aligned} M_1 \cdot y''_1 + B_1 \cdot y'_1 + k_1 \cdot y_1 - B_1 \cdot y'_2 - k_2 \cdot y_2 &= F_1(t) \\ -B_1 \cdot y'_1 - k_1 \cdot y_1 + M_2 \cdot y''_2 + B_1 \cdot y'_2 + (k_1 + k_2) \cdot y_2 &= 0 \end{aligned}$$

Sistema de 4 ecuaciones diferenciales de 1er. orden

$$\begin{aligned} y_1' &= v_1 \\ v_1' &= \frac{F_1(t) - B_1 \cdot v_1 + B_1 \cdot v_2 - k_1 \cdot y_1 + k_2 \cdot y_2}{M_1} \\ y_2' &= v_2 \\ v_2' &= \frac{B_1 \cdot v_1 + k_1 \cdot y_1 - B_1 \cdot v_2 - (k_1 + k_2) \cdot y_2}{M_2} \end{aligned}$$

# Implementación con vectores

La utilización de vectores simplifica notablemente la implementación de los métodos para resolver sistemas de ecuaciones diferenciales.

$$v = \begin{array}{|c|c|c|c|c|} \hline t & y_1 & v_1 & y_2 & v_2 \\ \hline \end{array}$$

$v(0) \quad v(1) \quad v(2) \quad v(3) \quad v(4)$

← Vector de variables

$$vp = \begin{array}{|c|c|c|c|c|} \hline 1 & y'_1 & v'_1 & y'_2 & v'_2 \\ \hline \end{array}$$

$vp(0) \quad vp(1) \quad vp(2) \quad vp(3) \quad vp(4)$

← Vector de derivadas



# Implementación con vectores (II)

$$y_1' = v_1$$

$$v_1' = \frac{F_1(t) - B_1 \cdot v_1 + B_1 \cdot v_2 - k_1 \cdot y_1 + k_2 \cdot y_2}{M_1}$$

$$y_2' = v_2$$

$$v_2' = \frac{B_1 \cdot v_1 + k_1 \cdot y_1 - B_1 \cdot v_2 - (k_1 + k_2) \cdot y_2}{M_2}$$

$$v = \begin{array}{|c|c|c|c|c|} \hline t & y_1 & v_1 & y_2 & v_2 \\ \hline \end{array}$$

$v(0) \quad v(1) \quad v(2) \quad v(3) \quad v(4)$

← Vector de variables

$$vp = \begin{array}{|c|c|c|c|c|} \hline 1 & y_1' & v_1' & y_2' & v_2' \\ \hline \end{array}$$

$vp(0) \quad vp(1) \quad vp(2) \quad vp(3) \quad vp(4)$

← Vector de derivadas

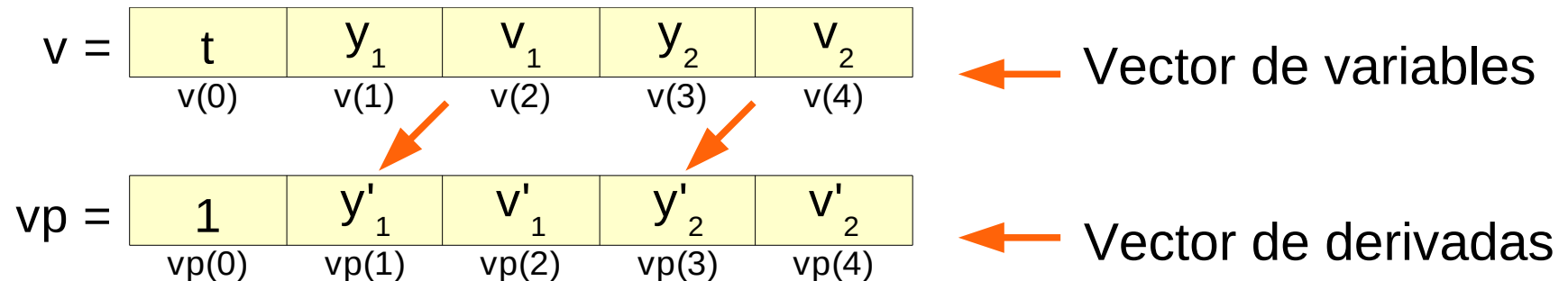
# Implementación con vectores (III)

$$vp(1) = v(2)$$

$$vp(2) = \frac{F_1(v(0)) - B_1 \cdot v(2) + B_1 \cdot v(4) - k_1 \cdot v(1) + k_2 \cdot v(3)}{M_1}$$

$$vp(3) = v(4)$$

$$vp(4) = \frac{B_1 \cdot v(2) + k_1 \cdot v(1) - B_1 \cdot v(4) - (k_1 + k_2) \cdot v(3)}{M_2}$$





# Definición de la Función Derivada

```
MODULE sistema_MRA
! Cantidad de Ecuaciones del Sistema
INTEGER, PARAMETER :: cant_ec = 4

CONTAINS

FUNCTION v_prima(v)
! Definición del Sistema de Ecuaciones Diferenciales
REAL(8), PARAMETER :: m1=1.0, m2=0.05, k1=5.0, k2=5.0, b1=1.0
REAL(8), DIMENSION(0:cant_ec) :: v, v_prima

v_prima(0) = 1.0
v_prima(1) = v(2)
v_prima(2) = (-b1*v(2)+b1*v(4)-k1*v(1)+k2*v(3))/m1
v_prima(3) = v(4)
v_prima(4) = (-b1*v(4)+b1*v(2)-k1*v(3)+k1*v(1)-k2*v(3))/m2

END FUNCTION v_prima

END MODULE
```

# Euler Simple (Implementación)

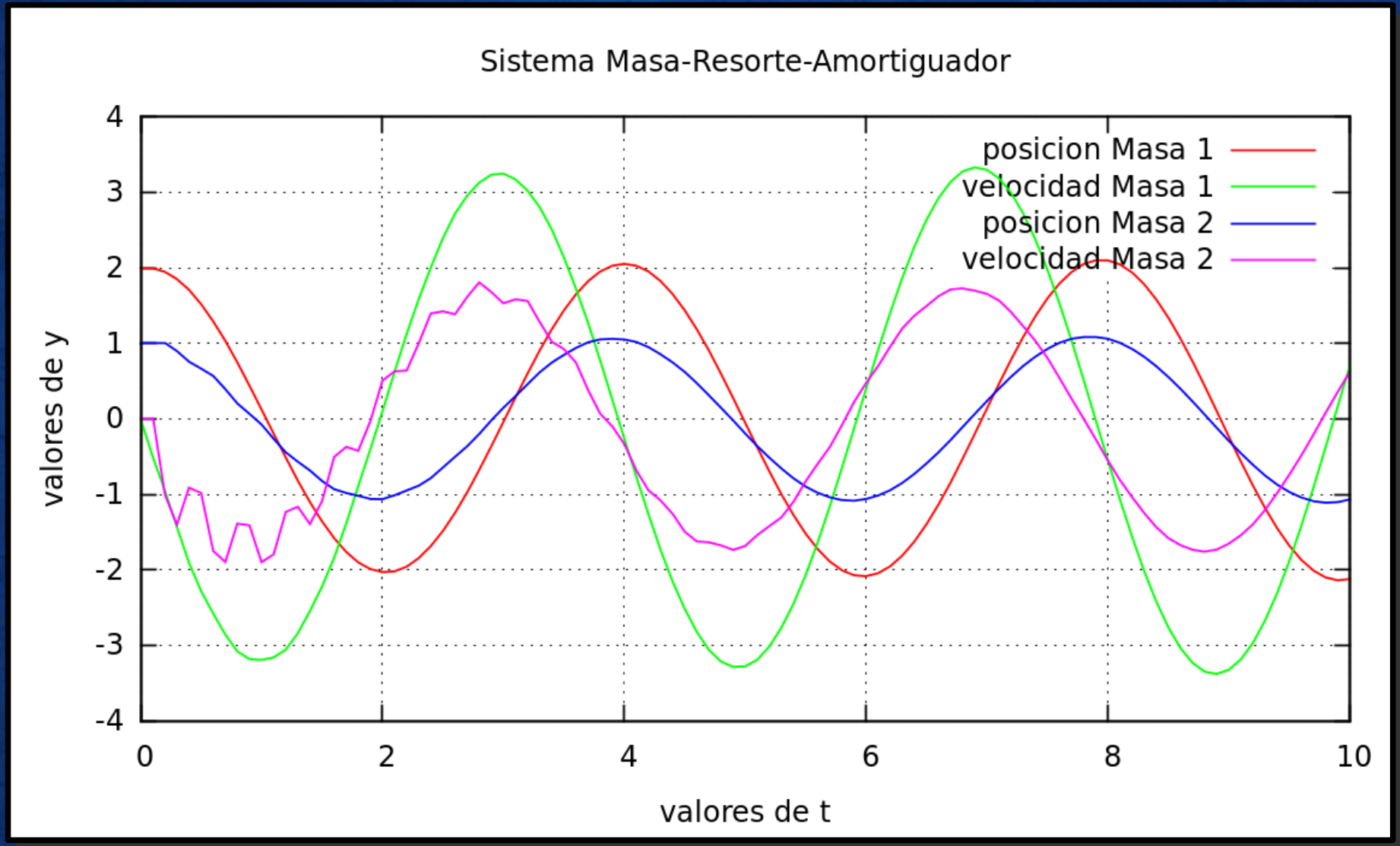
```
SUBROUTINE EulerSimple(vi, cant_ec, max_iter, h, formato)
!Metodo de Euler Simple
INTEGER, INTENT(IN) :: cant_ec, max_iter
REAL(8), INTENT(IN), DIMENSION(0:cant_ec) :: vi
REAL(8), INTENT(IN) :: h
CHAR*15, INTENT(IN) :: formato
INTEGER iter
REAL(8), DIMENSION(0:cant_ec) :: v

WRITE (*, formato) 0, vi
v = vi
DO iter = 1, max_iter
    v = v + h*v_prima(v)
    WRITE (*, formato) iter, v
END DO

END SUBROUTINE EulerSimple
```



# Solución Euler Simple ( $h=0.1$ )



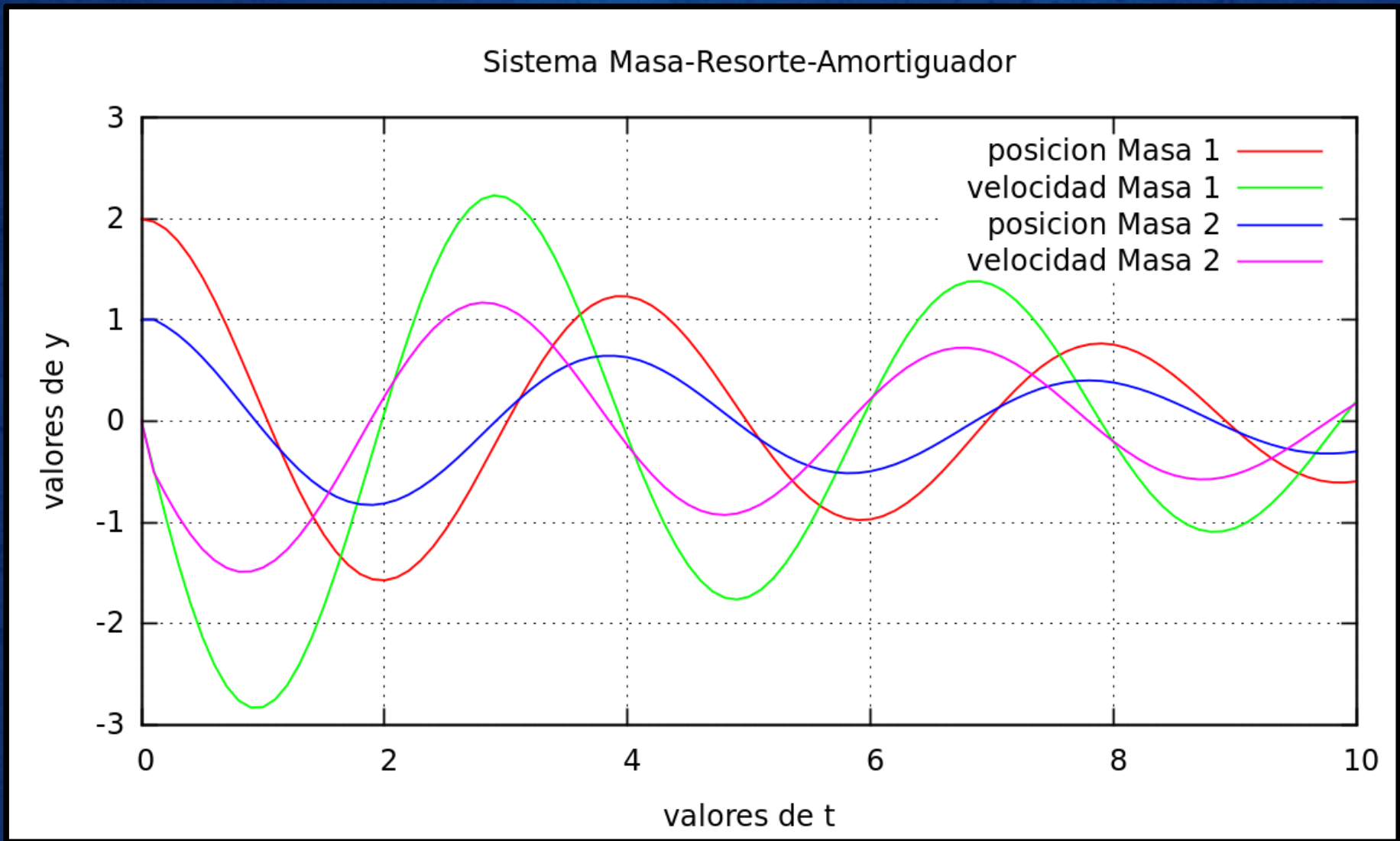
# Euler Modificado (Implementación)

```
SUBROUTINE EulerModificado(vi, cant_ec, max_iter, h, formato)
!Metodo de Euler Modificado
INTEGER, INTENT(IN) :: cant_ec, max_iter
REAL(8), INTENT(IN), DIMENSION(0:cant_ec) :: vi
REAL(8), INTENT(IN) :: h
CHAR*15, INTENT(IN) :: formato
INTEGER iter
REAL(8), DIMENSION(0:cant_ec) :: v
WRITE (*, formato) 0, vi
v = vi
DO iter = 1, max_iter
    vp = v_prima(v)
    v = v + h*(vp + v_prima(v + h*vp))/2.0
    WRITE (*, '(I5, formato) iter, v
END DO

END SUBROUTINE EulerModificado
```



# Solución Euler Modificado ( $h=0.1$ )

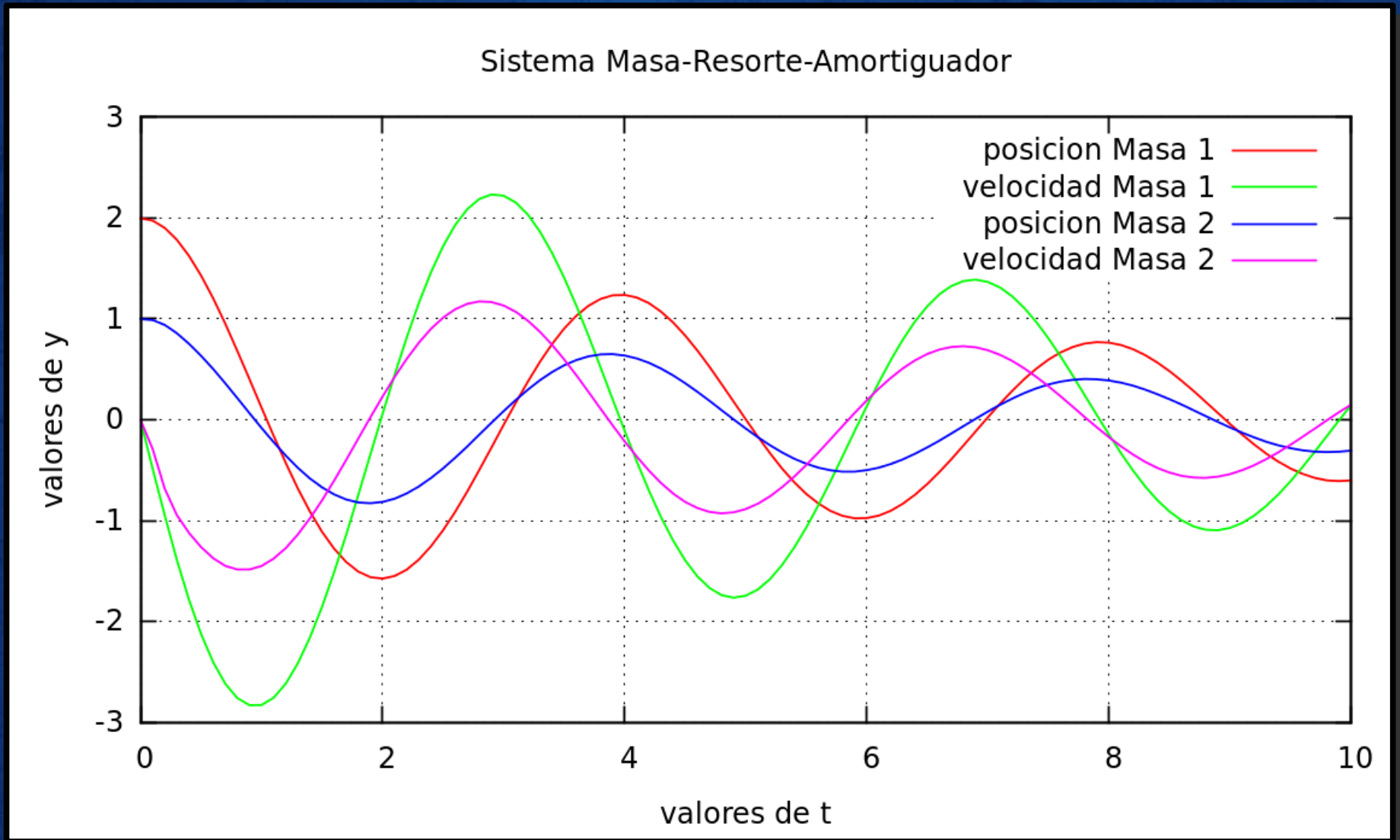


# Runge-Kutta (4to. Orden)

```
SUBROUTINE RungeKutta(vi, cant_ec, max_iter, h, formato)
!Metodo de Runge-Kutta (de 4to. orden)
REAL(8), INTENT(IN), DIMENSION(0:cant_ec) :: vi
REAL(8), INTENT(IN) :: h
INTEGER, INTENT(IN) :: cant_ec, max_iter
CHAR*15, INTENT(IN) :: formato
INTEGER iter
REAL(8), DIMENSION(0:cant_ec) :: v, k1, k2, k3, k4
WRITE (*, formato) 0, vi
v = vi
DO iter = 1, max_iter
    k1 = h*v_prima(v)
    k2 = h*v_prima(v+k1/2.0)
    k3 = h*v_prima(v+k2/2.0)
    k4 = h*v_prima(v+k3)
    v = v + (k1 + 2.0*k2 + 2.0*k3 +k4)/6.0
    WRITE (*, formato) iter, v
END DO
END SUBROUTINE RungeKutta
```



# Solución Runge-Kutta 4to. Orden ( $h=0.1$ )



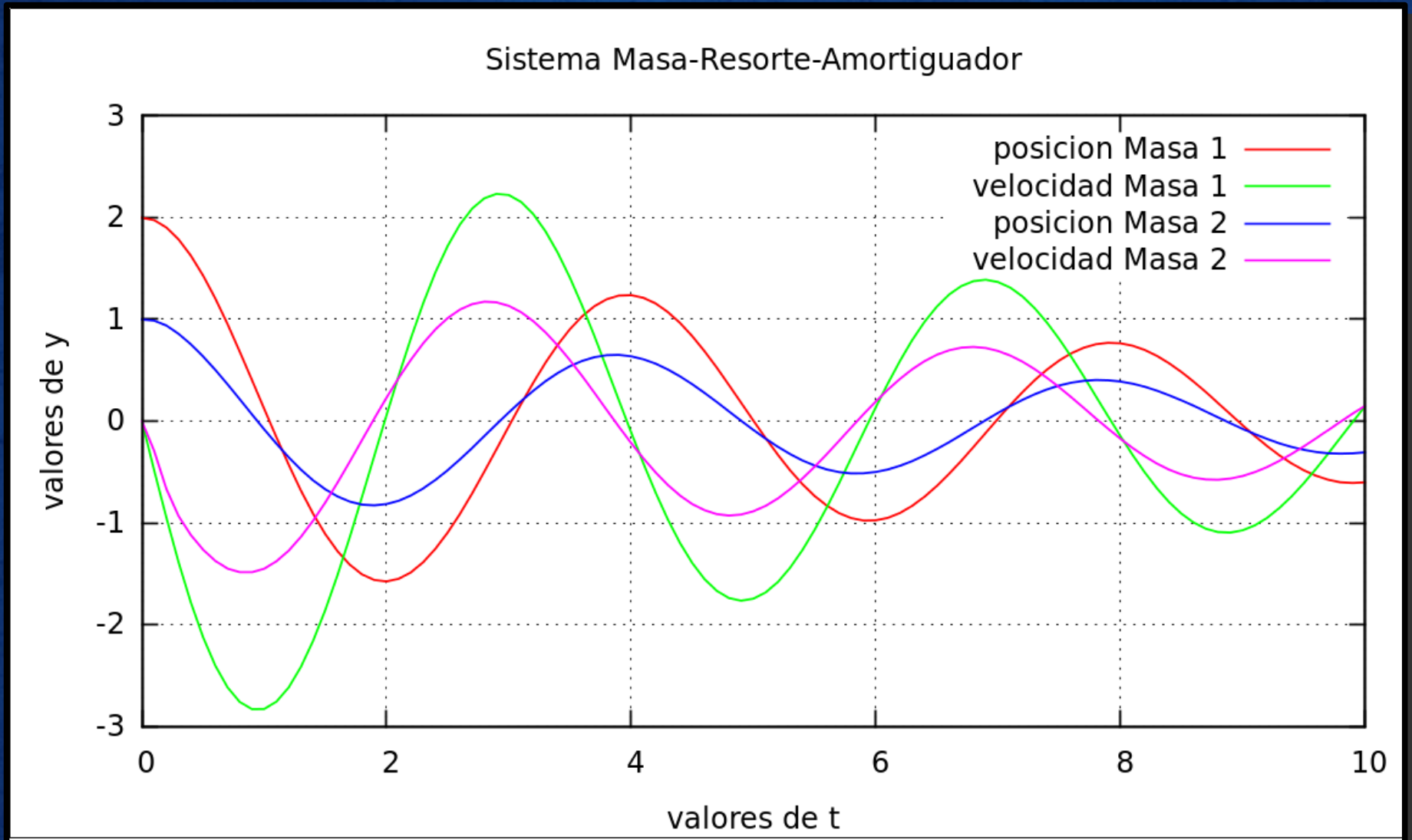
# Runge-Kutta-Fehlberg

```
SUBROUTINE RungeKuttaFehlberg(vi, cant_ec, max_iter, h, formato)
!Metodo de Runge-Kutta-Fehlberg (de 6to. orden)
REAL(8), INTENT(IN), DIMENSION(0:cant_ec) :: vi
REAL(8), INTENT(IN) :: h
INTEGER, INTENT(IN) :: cant_ec, max_iter
CHAR*15, INTENT(IN) :: formato
INTEGER iter
REAL(8), DIMENSION(0:cant_ec) :: v, k1, k2, k3, k4, k5, k6, e

WRITE (*, formato) 0, vi, 0
v = vi
DO iter = 1, max_iter
    k1 = h*v_prima(v)
    k2 = h*v_prima(v + k1/4.0)
    k3 = h*v_prima(v + (3.0*k1 + 9.0*k2)/32.0)
    k4 = h*v_prima(v + (1932.0*k1 - 7200.0*k2 + 7296.0*k3)/2197.0)
    k5 = h*v_prima(v + 439.0*k1/216.0 - 8.0*k2 + 3680.0*k3/513.0 - 845.0*k4/4104.0)
    k6 = h*v_prima(v - 8.0*k1/27.0 + 2.0*k2 - 3544.0*k3/2565.0 + 1859.0*k4/4104.0 - 11.0*k5/40.0)
    v = v + (25.0*k1/216.0 + 1408.0*k3/2565.0 + 2197.0*k4/4104.0 - k5/5.0)
    e = k1/360.0 - 128.0*k3/4275.0 - 2197.0*k4/75240.0 + k5/50.0 + 2.0*k6/55.0
    WRITE (*, formato) iter, v, e
END DO
END SUBROUTINE RungeKuttaFehlberg
```



# Runge-Kutta-Fehlberg (h=0.1)



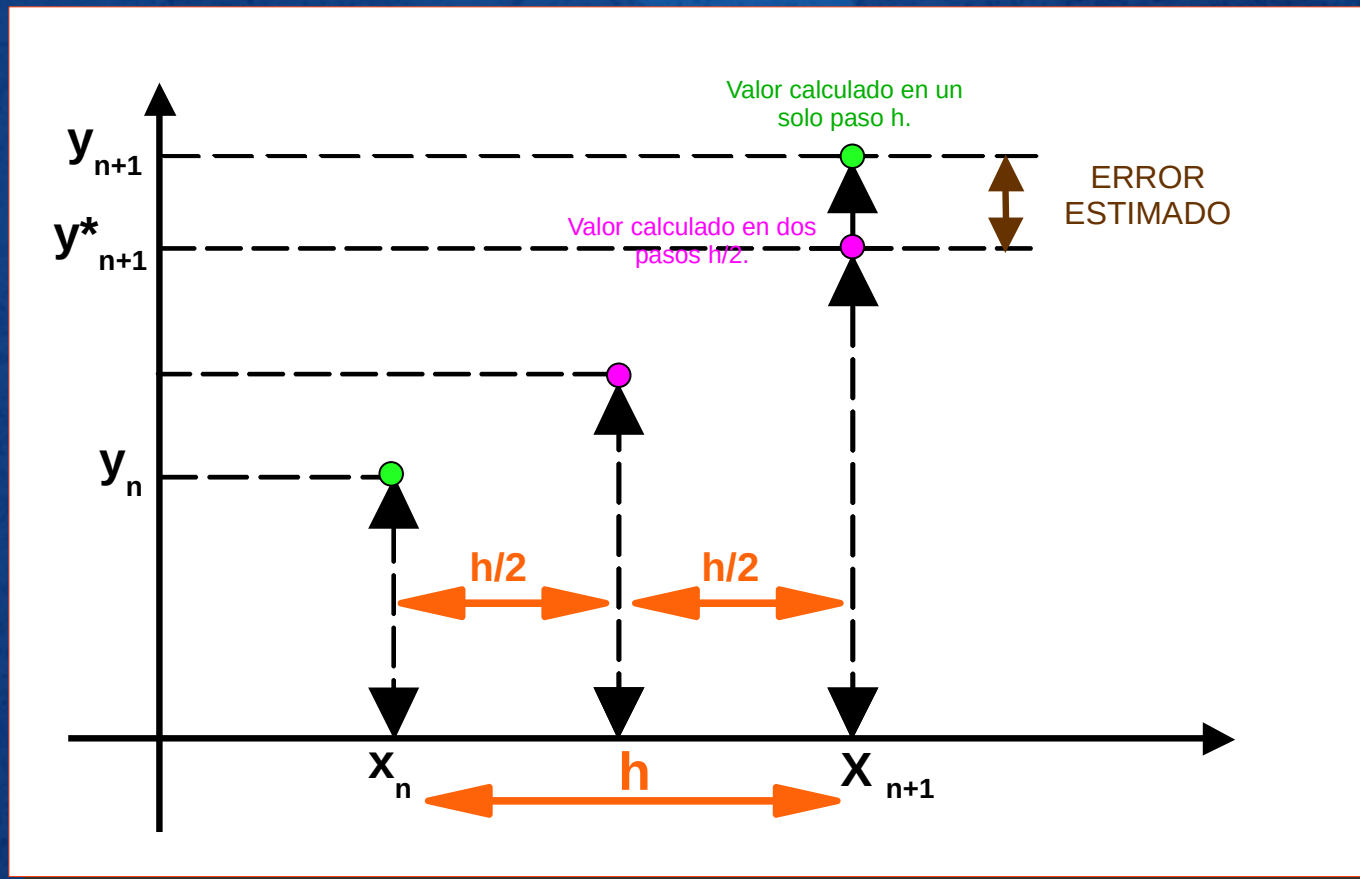
# Estrategias de ajuste del paso $h$

- La exactitud de los métodos numéricos de resolución de ecuaciones diferenciales depende fuertemente del **paso  $h$**  elegido.
- Existen diferentes estrategias para ajustar automáticamente el **paso  $h$**  y de esta forma mantener el error de la solución por debajo de una cierta tolerancia establecida.



# Estimación del Error

En aquellos métodos que **no tienen** una fórmula específica para la **estimación del error**, es posible estimarlo por medio de la diferencia del valor calculados con **un paso de avance  $h$**  y del obtenido al avanzar **dos pasos  $h/2$** .

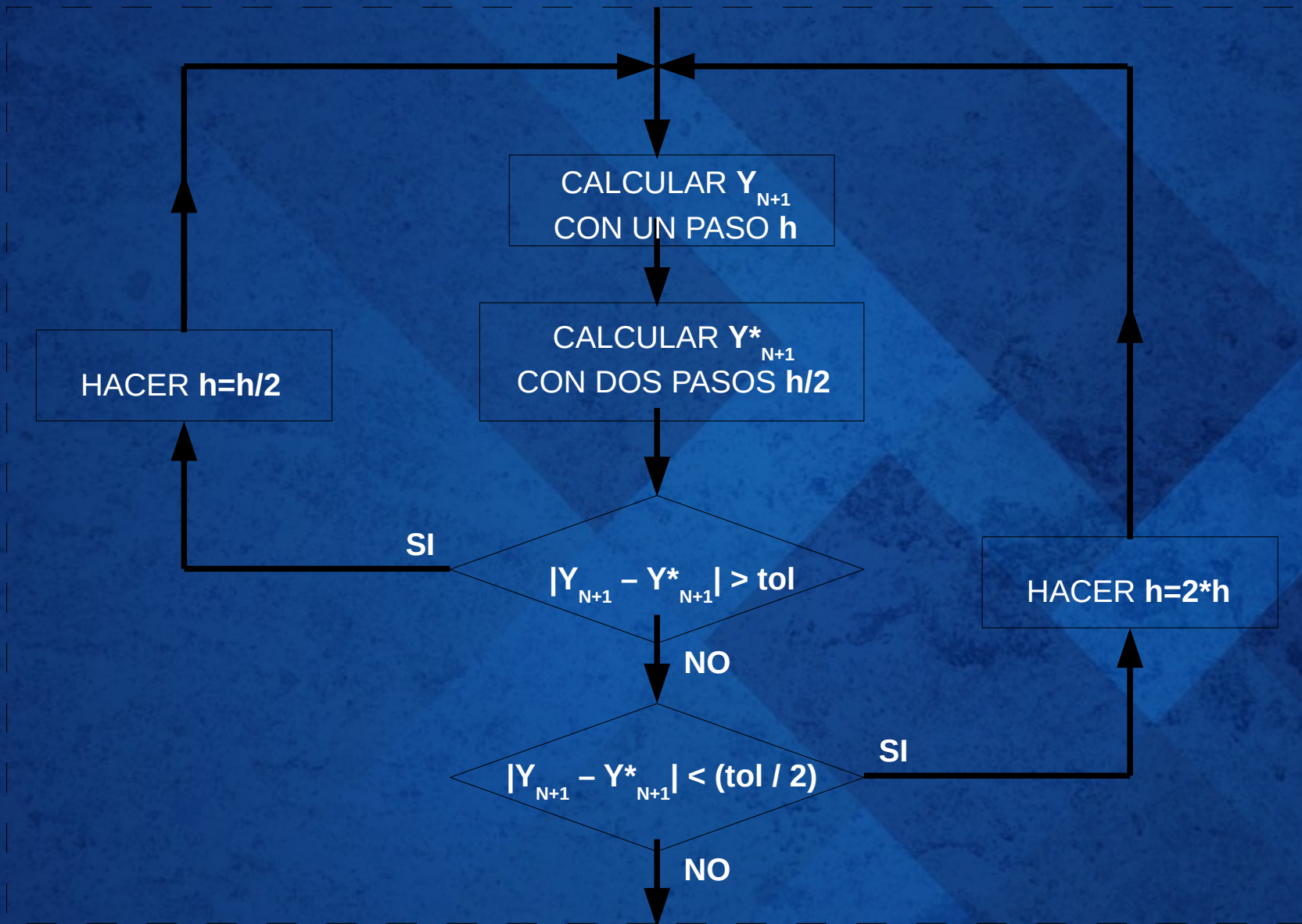


# Estrategia de ajuste 1

- Se calcula el valor  $y_{n+1}$  utilizando un paso de avance  $h$ .
- Se calcula el valor  $y_{n+1}^*$  utilizando dos pasos de avance  $h/2$ .
- Si el valor absoluto de la diferencia entre ambos valores es **mayor** a la tolerancia establecida, entonces se establece  $h/2$  como nuevo valor de  $h$  y se vuelve al **paso 1**.
- Si el valor absoluto de la diferencia entre ambos valores es **menor a la mitad** de la tolerancia establecida, entonces se establece  $2*h$  como nuevo valor de  $h$  y se vuelve al **paso 1**.

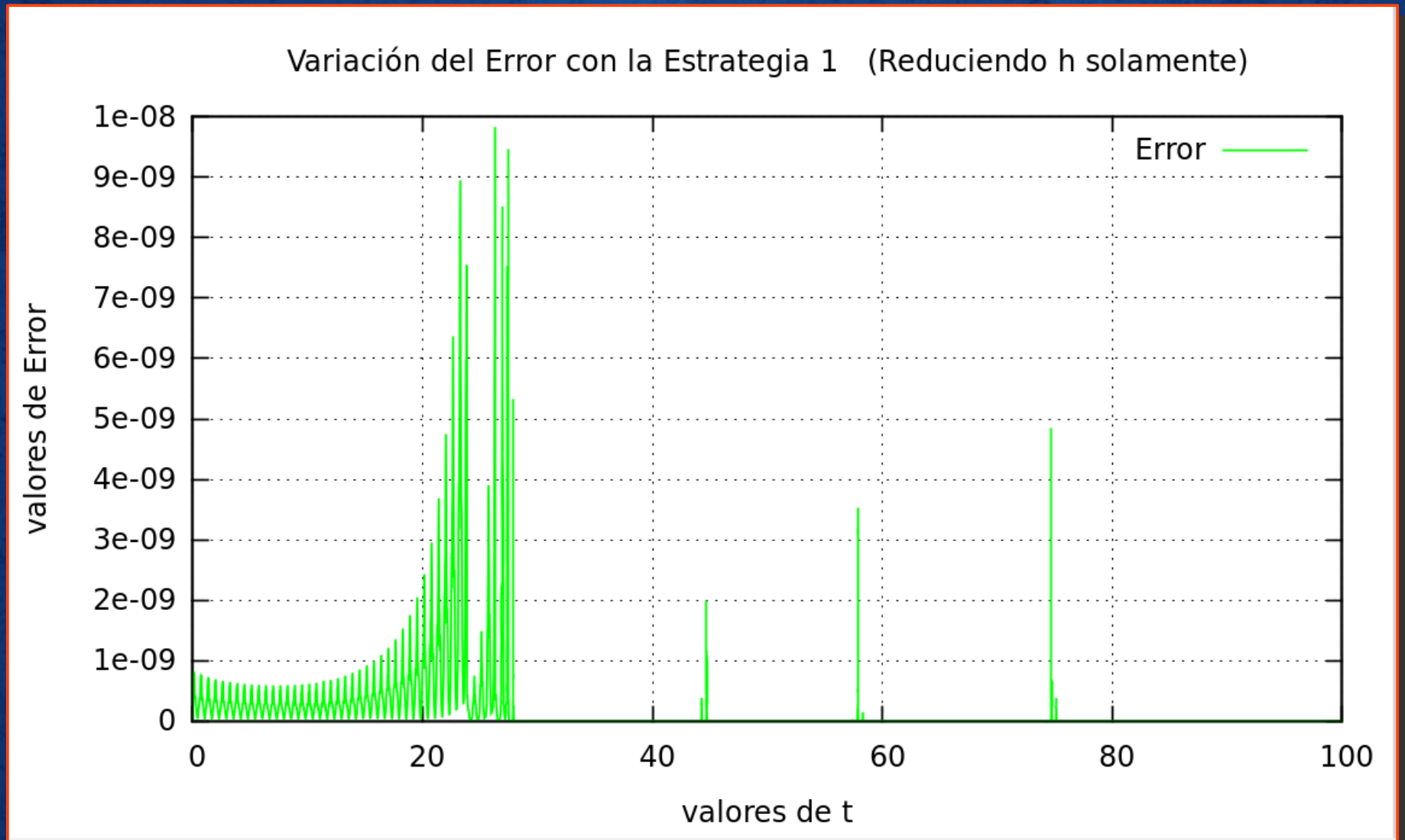


# Diagrama de Flujo de la estrategia 1



# Variación del error con la Estrategia 1

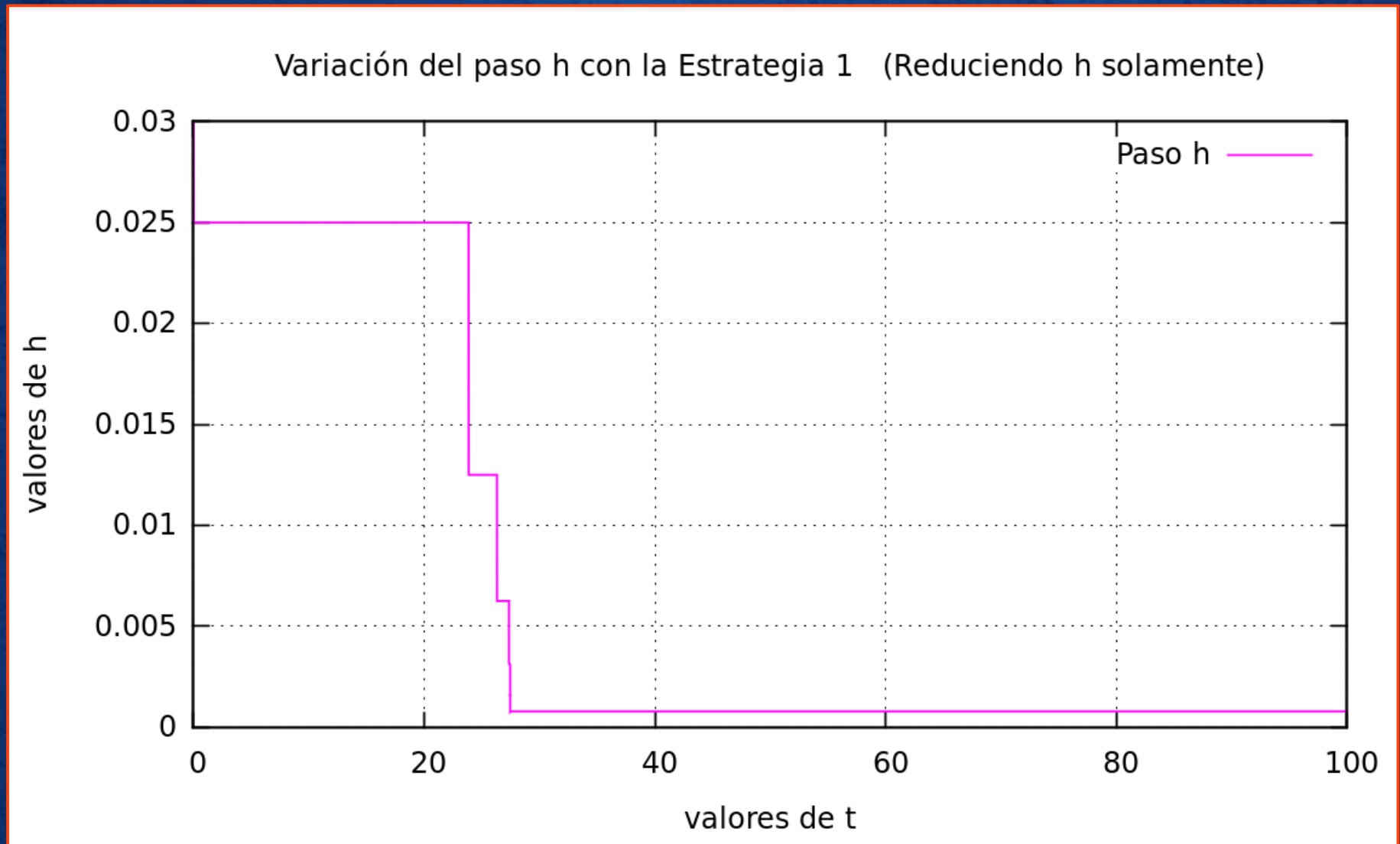
El paso se reduce de ser necesario, pero nunca se aumenta.





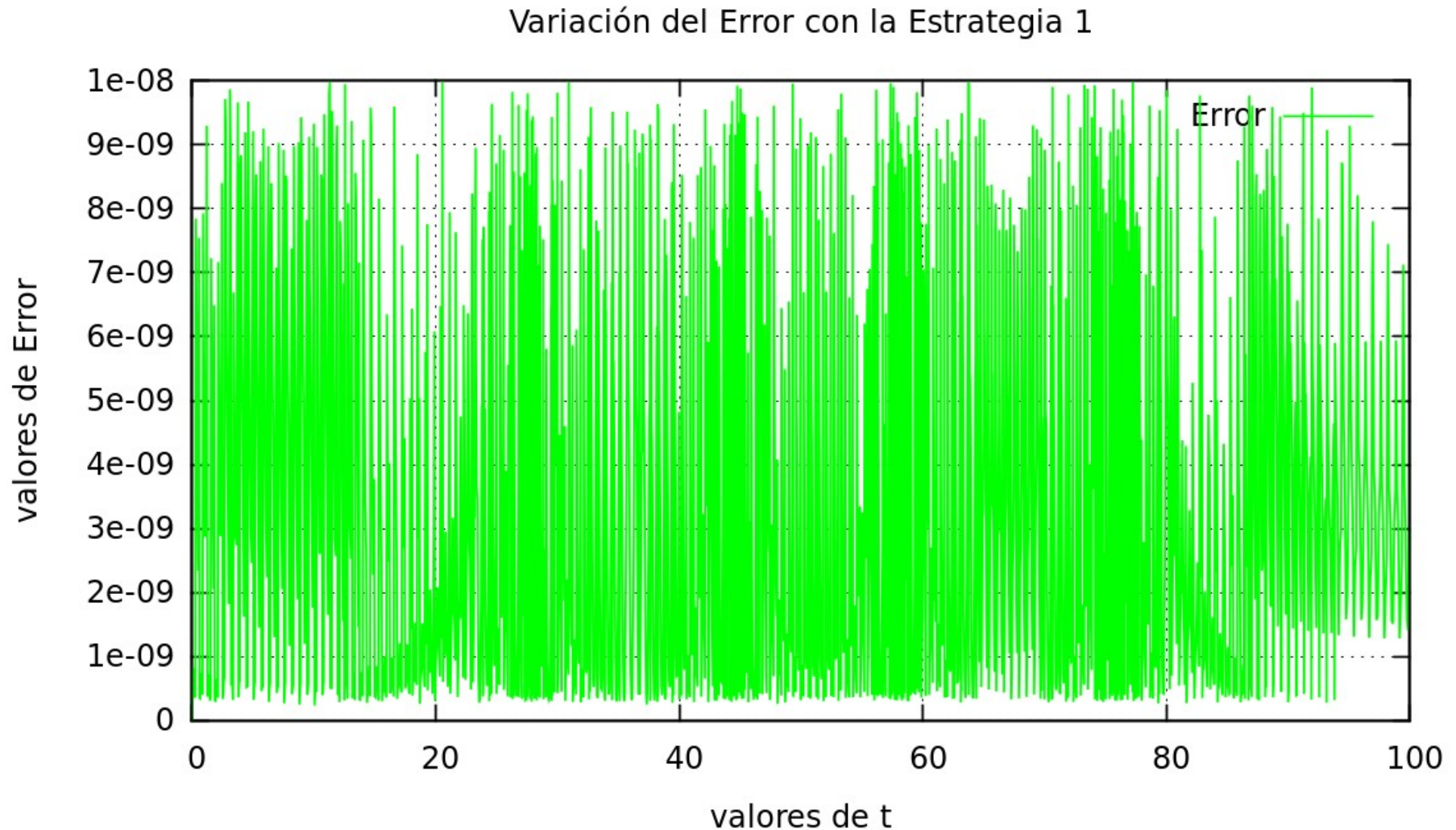
# Variación del paso $h$ con la Estrategia 1

El paso se reduce de ser necesario, pero nunca se aumenta.



# Variación del error con la Estrategia 1

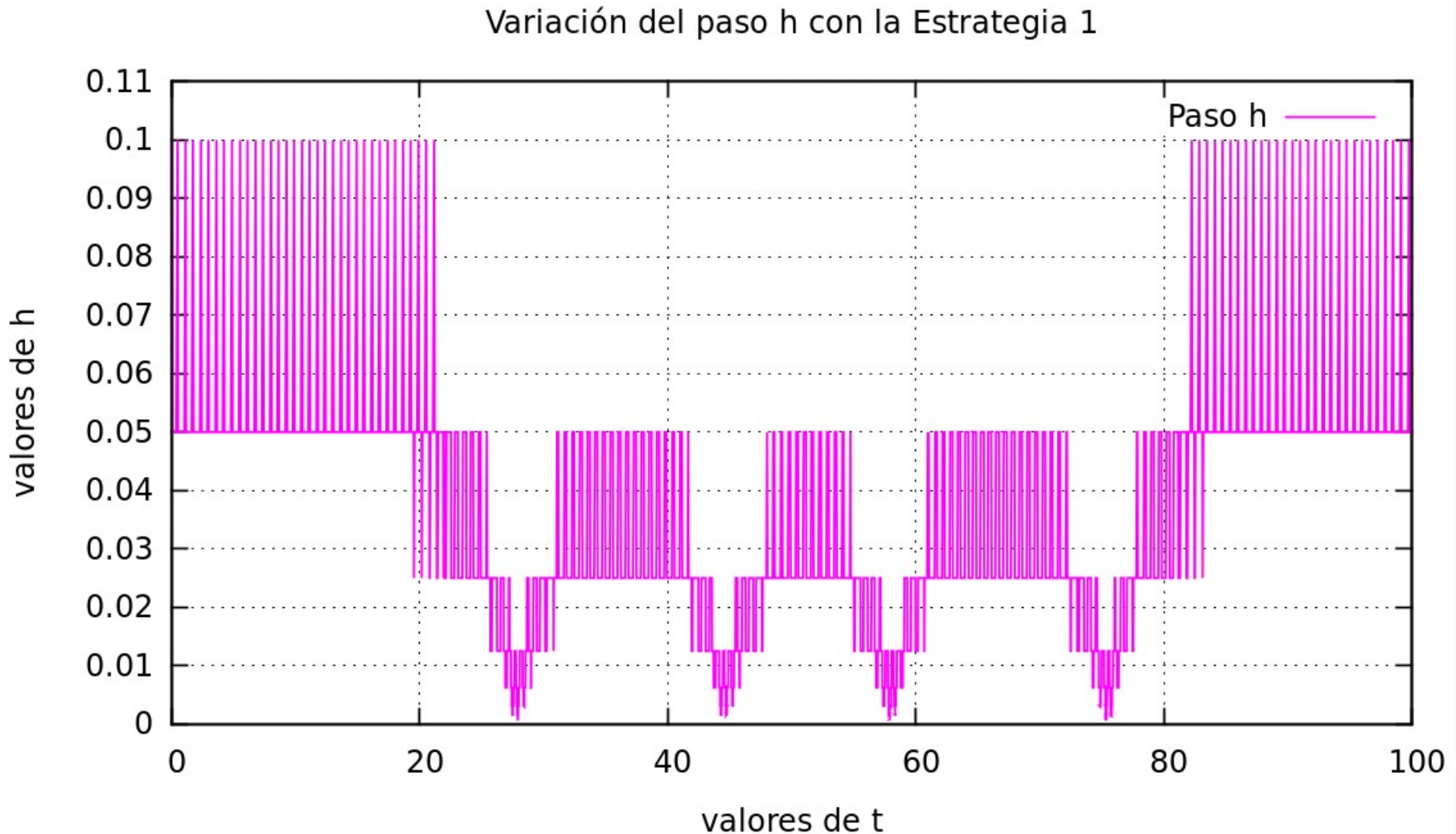
El paso se reduce o se aumenta para lograr la cota de error deseada.





# Variación del paso $h$ con la Estrategia 1

El paso se reduce o se aumenta para lograr la cota de error deseada.



# Estrategia de ajuste 2

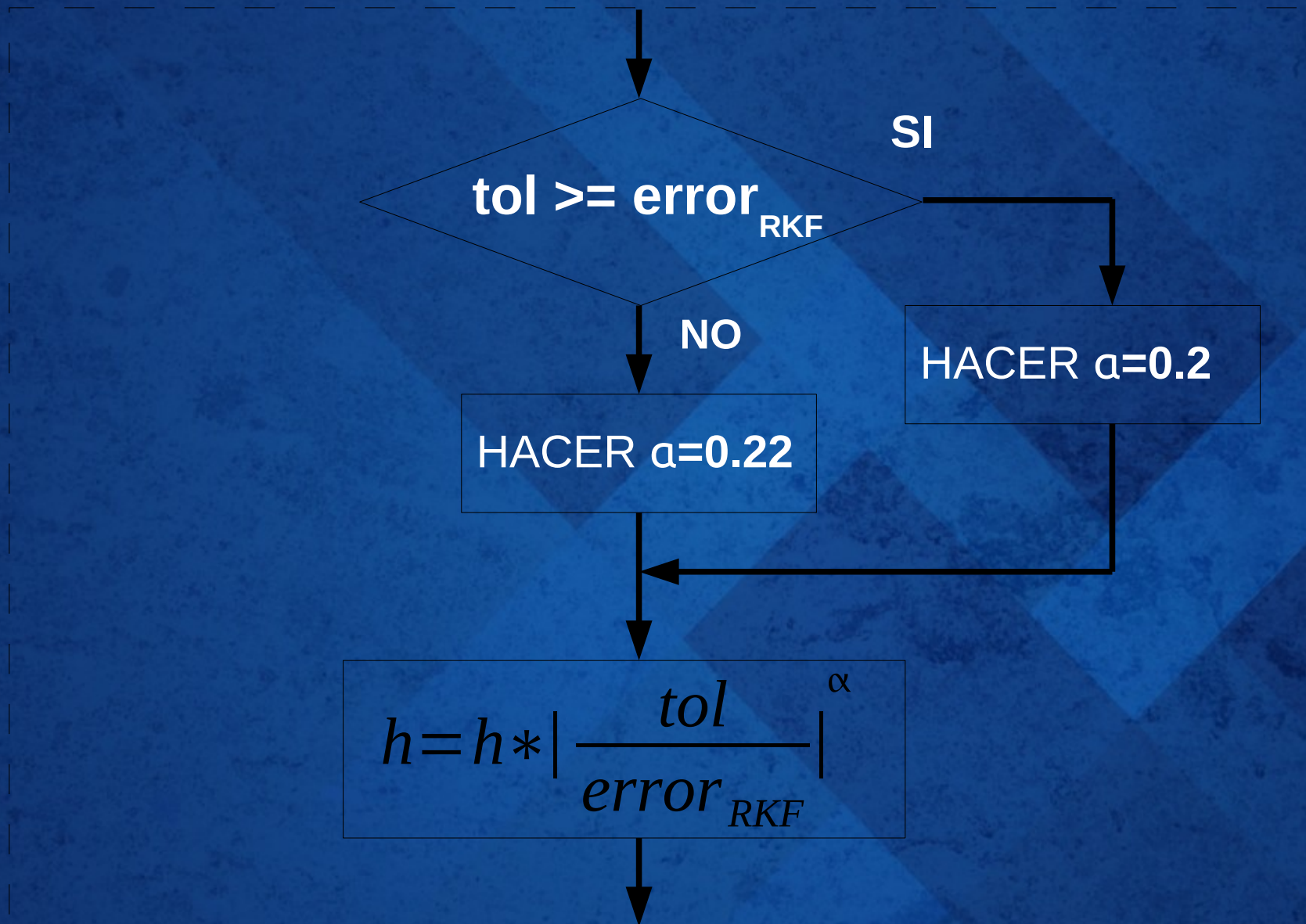
A partir de una estimación del error obtenida por medio de un método como por ejemplo **Runge-Kutta-Fehlberg**, podemos ajustar el **valor de h**, en cada paso.

$$h_{nuevo} = h_{actual} \cdot \left| \frac{tol_{problema}}{\epsilon_{RKF}} \right|^{\alpha}$$

El valor de  $\alpha$  se ajusta en base a la relación entre el valor estimado del error y la tolerancia establecida.

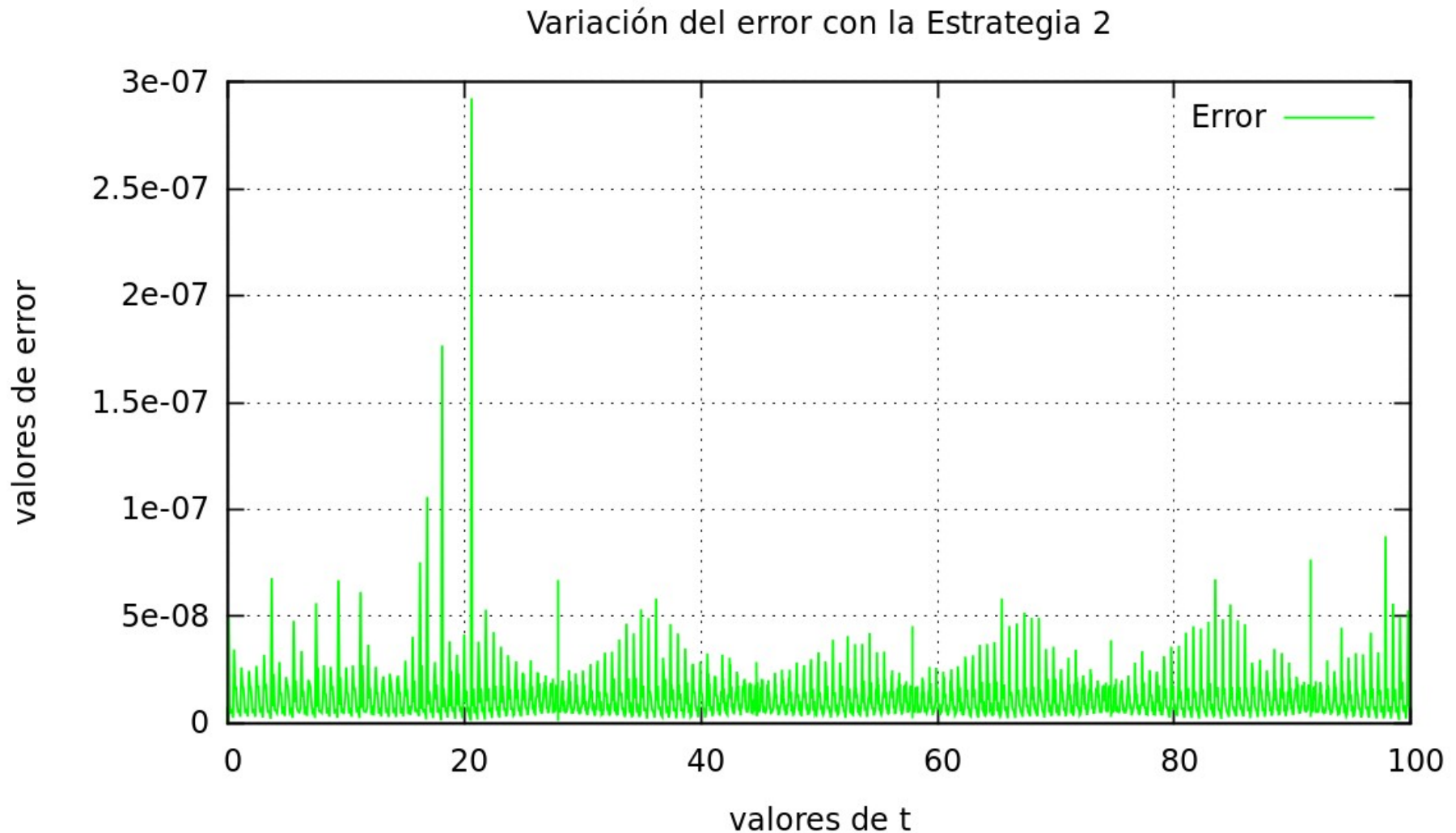


# Diagrama de la estrategia 2



# Variación del error con la Estrategia 2

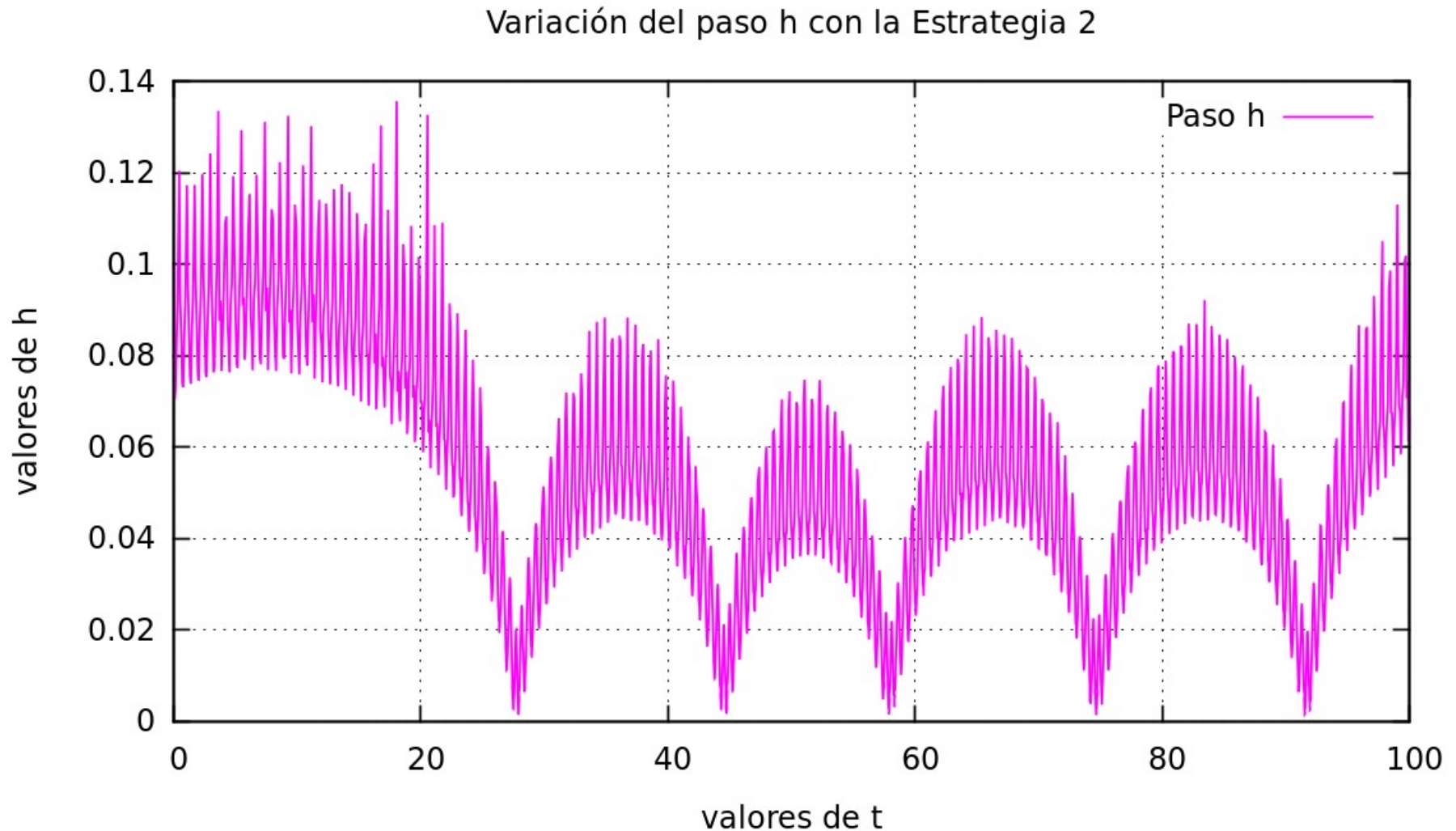
El paso se ajusta de acuerdo al valor de error calculado.





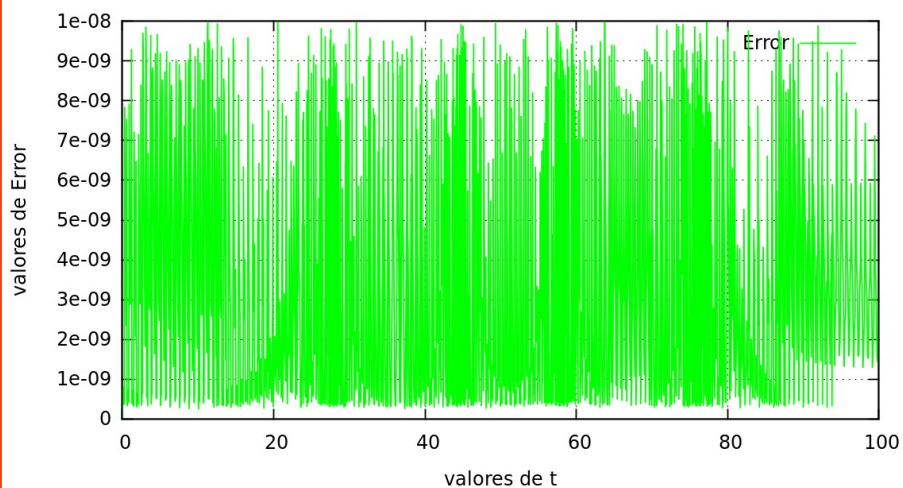
# Variación del paso $h$ con la Estrategia 2

El paso se ajusta de acuerdo al valor de error calculado.

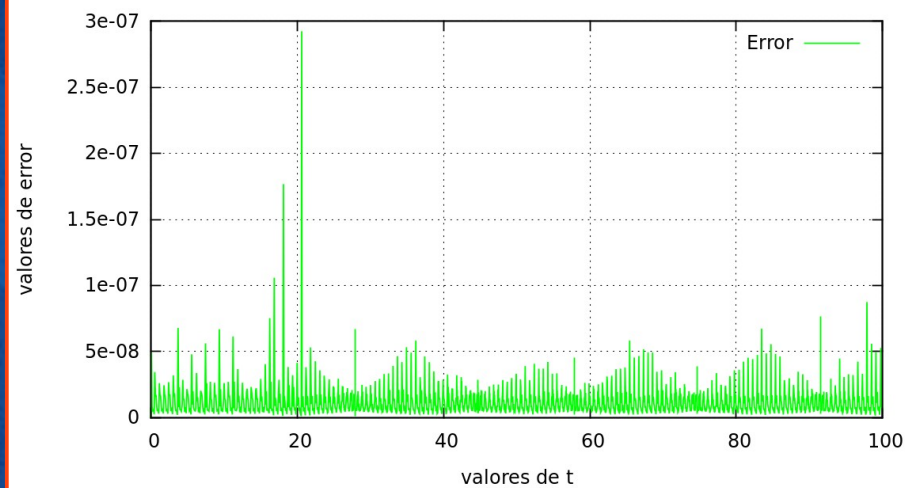


# Comparación de ambas estrategias

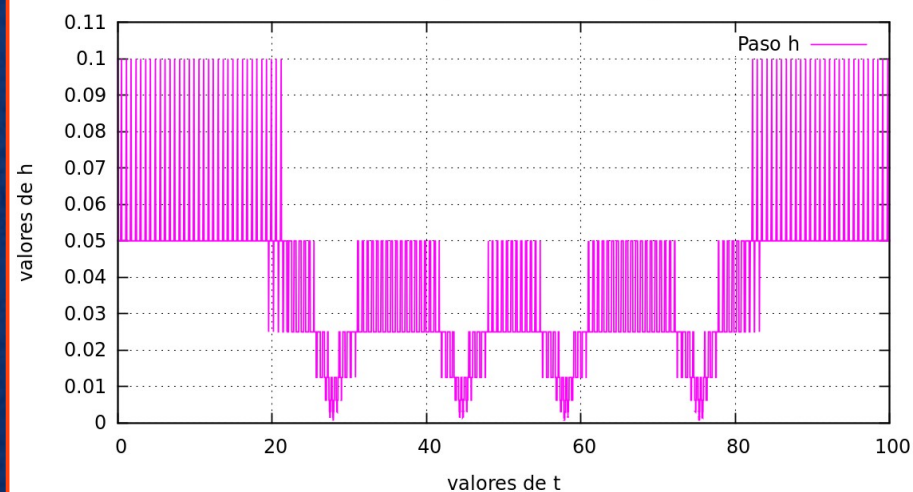
Variación del Error con la Estrategia 1



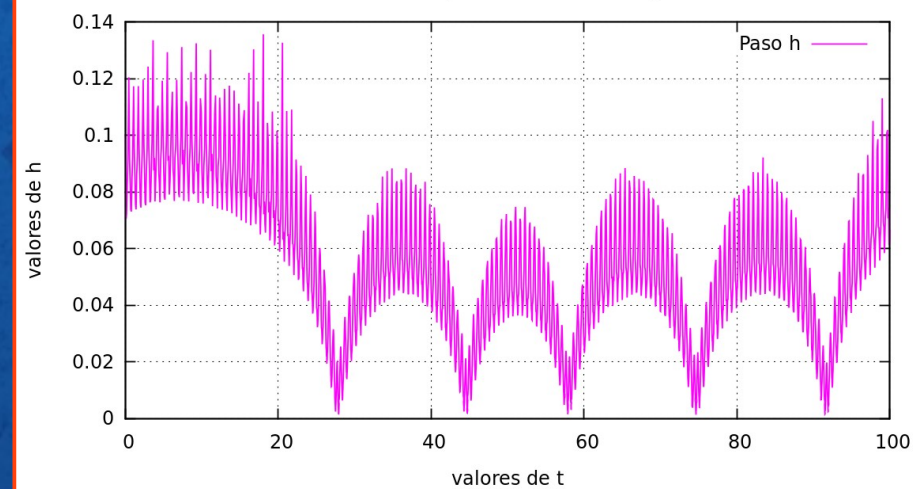
Variación del error con la Estrategia 2



Variación del paso h con la Estrategia 1



Variación del paso h con la Estrategia 2





# Resumen de la Estrategia de ajuste 1

- **Ventajas**

- No se necesita de un método que calcule explícitamente el error en cada paso.
- Los valores de  $h$  obtenidos **son siempre múltiplos** del  $h$  original.

- **Desventajas**

- La implementación es un poco más complicada que la de la **estrategia (II)**.

# Estrategia de ajuste 2

- **Ventajas**

- Su implementación es muy sencilla.

- **Desventajas**

- El valor de  $h$  que se obtiene generalmente **no es un múltiplo** del  $h$  original.



# Preguntas . . .

