Análisis Numérico para Ingeniería

Clase Nro. 3

“No llega antes el que va más rápido, sino el que sabe a donde va.”

Séneca

Temas a tratar en esta clase :

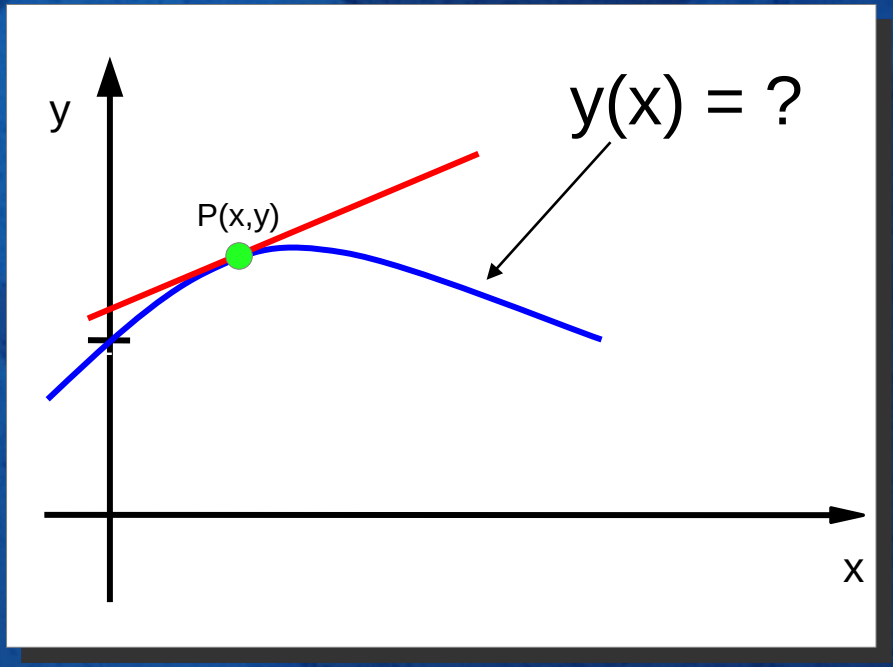
- Introducción
- Problemas de Valores Iniciales
- Método de la Serie de Taylor
- Método de Euler Simple
- Método de Euler Modificado
- Método de Runge-Kutta 4to.Orden
- Métodos de Runge-Kutta de Orden superior

Cuál es el problema a resolver ?

Hallar los valores de una determinada función y(x) desconocida.

Datos:

- Se sabe que la derivada de una función desconocida está representada por una determinada expresión.
- Sólo se conoce el valor de dicha función desconocida **y(x)**, en un punto determinado.



Ejemplo:

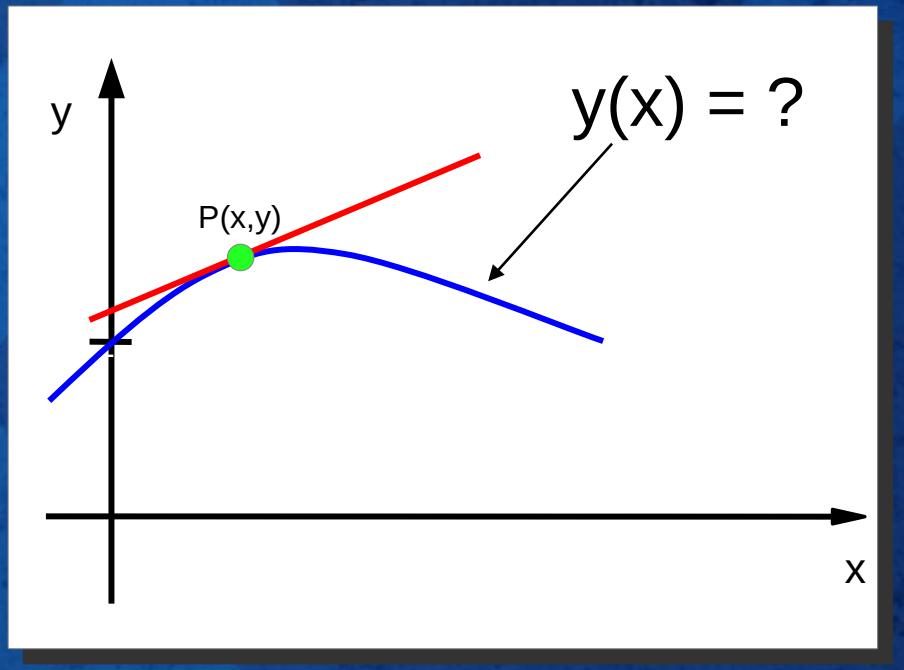
Datos del Problema:

Ecuación Diferencial

$$\frac{dy(x)}{dx} = x * y^{\frac{1}{3}}$$

Valores Iniciales

$$x_0 = 1$$
$$y_0 = y(x_0) = 1$$



Ecuación Diferencial Ordinaria

- **Una ecuación diferencial ordinaria,** es una ecuación que contiene derivadas con respecto a una sola variable independiente.

- Muchos problemas de Ingeniería se modelan con **E**cuaciones **D**iferenciales **O**rdinarias *(EDO)*, por lo que es de gran importancia hallar su solución.

- ***Solución Analítica:*** Algunos de los métodos generales de resolución de ecuaciones diferenciales ordinarias lineales que permiten encontrar soluciones analíticas utilizan **separación de variables**, **Series de Taylor**, etc.

- ***Solución Numérica:*** Algunos de los métodos para resolver numéricamente ecuaciones diferenciales, son los métodos de **Runge-Kutta**, los métodos **multipaso** y los métodos de **extrapolación.**

Ecuaciones Diferenciales Ordinarias

- Las **EDO**s son ecuaciones diferenciales en las cuales, todas las variables dependientes son funciones de una **sola** variable independiente.

- Una EDO de **orden n explícita**, donde **n** es el orden de la **máxima derivada**, con variable independiente **x** es:

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

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

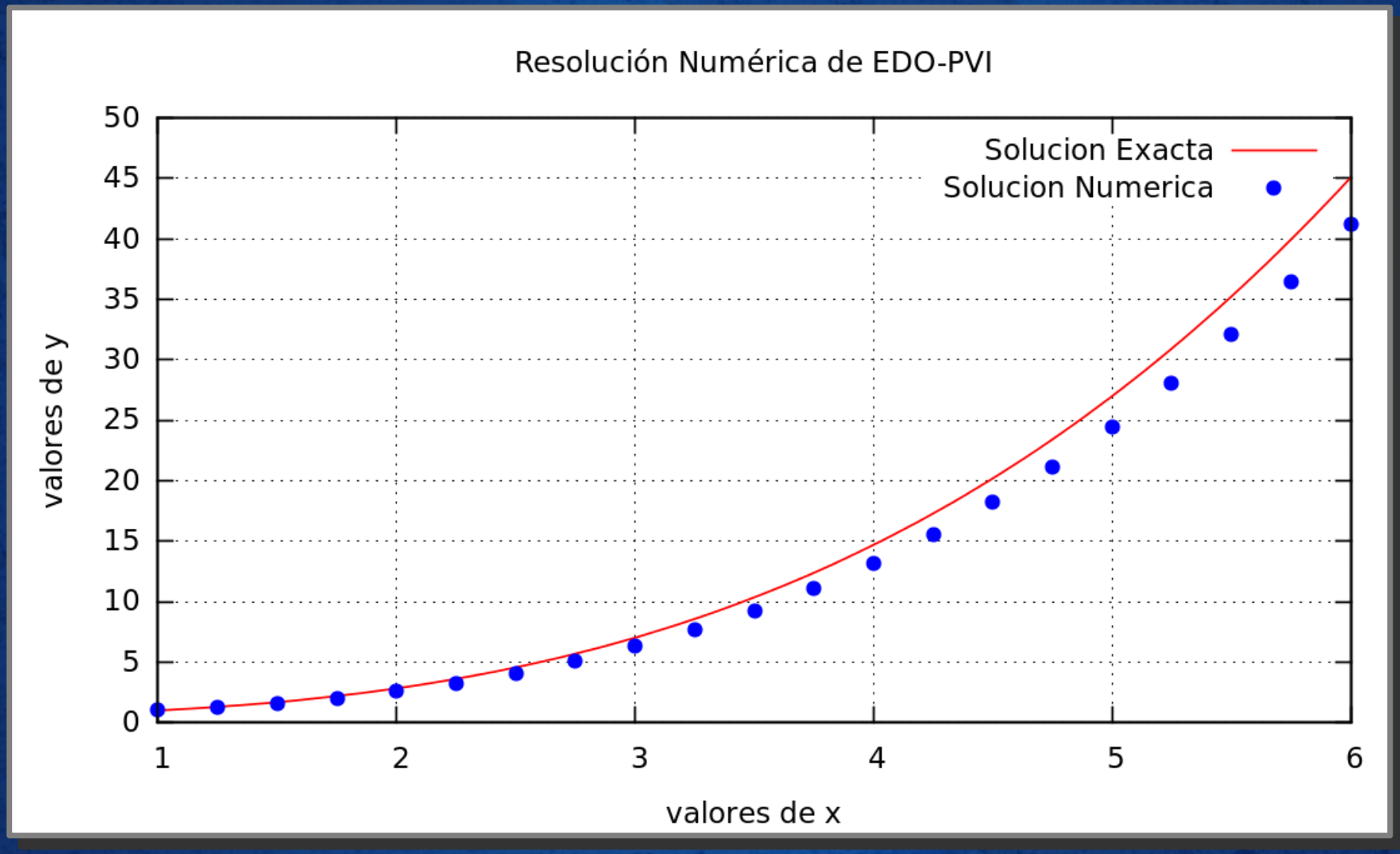
Solución Numérica de EDOs

- La solución numérica se utiliza cuando no es posible o es extremadamente difícil obtener la integral de **f(x,y)**.

- Encontrar la solución numérica de una ecuación diferencial implica encontrar **un conjunto de valores funcionales**.

- Cuando se halla la solución numérica, **NO** se obtiene una expresión funcional de la forma **y = f(x)**, sino un conjunto de valores numéricos.

Gráfico de una Solución Numérica



Método de la Serie de Taylor

Si bien no puede considerarse estrictamente como un método numérico, constituye la base del desarrollo de muchos métodos numéricos de resolución de **EDO PVI**, como los denominados métodos de **RUNGE-KUTTA**.

$$y(x)=y(x_0)+y'(x_0)(x-x_0)+\frac{y''(x_0)(x-x_0)^2}{2!}+\frac{y'''(x_0)(x-x_0)^3}{3!}+\cdots$$

Y si definimos $h = (x - x_0)$, nos queda:

$$y(x)=y(x_0)+y'(x_0)h+\frac{y''(x_0)h^2}{2!}+\frac{y'''(x_0)h^3}{3!}+\cdots$$

Método de la Serie de Taylor

$$y(x)=y(x_0)+y'(x_0)(x-x_0)+\frac{y''(x_0)(x-x_0)^2}{2!}+\frac{y'''(x_0)(x-x_0)^3}{3!}+\cdots$$

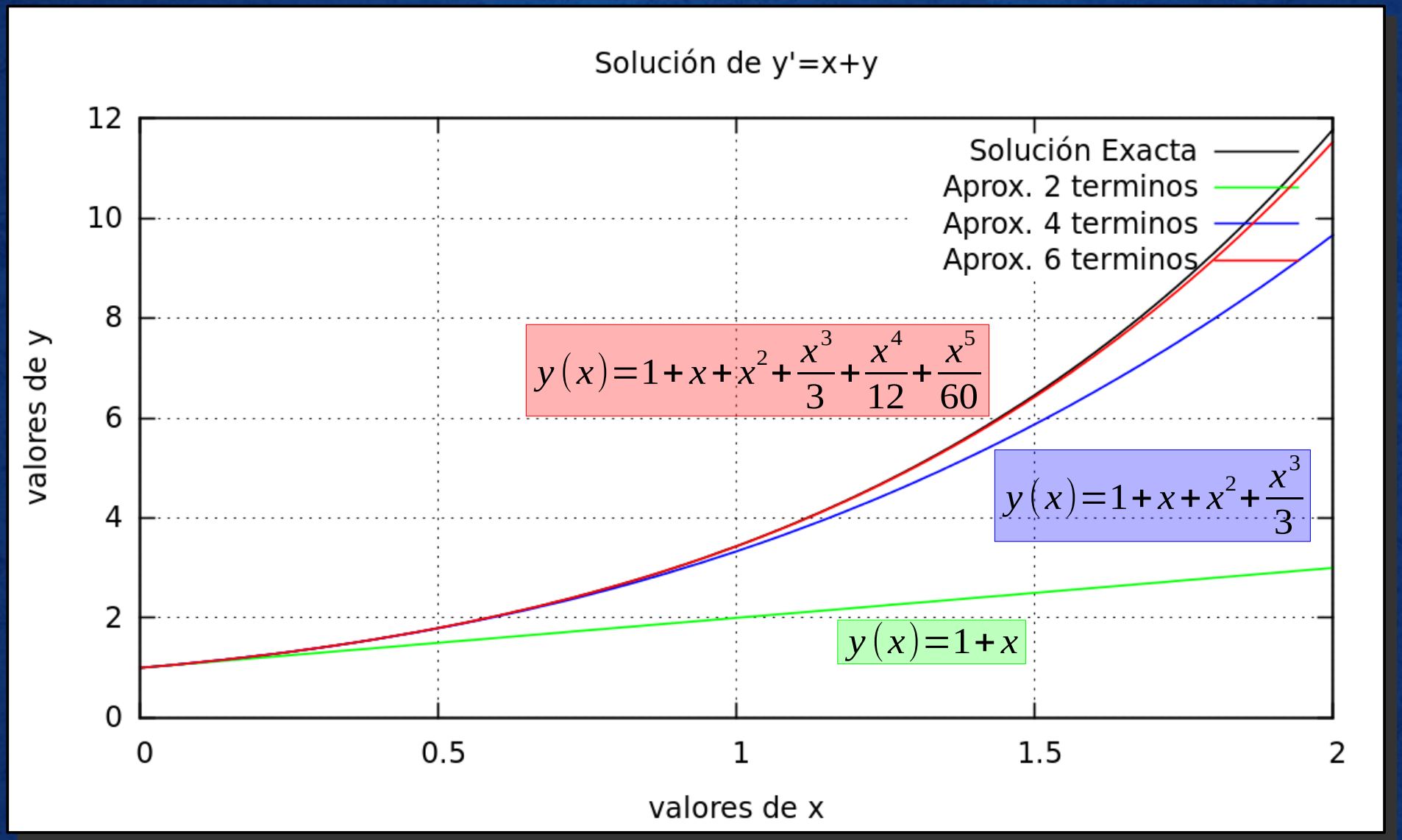
Ejemplo:

$$y'=x+y$$
$$y(0)=1 \Rightarrow x_0=0$$

$$\begin{aligned} y'=x+y &\Rightarrow y'(x_0)=0+1=1 \\ y''=1+y' &\Rightarrow y''(x_0)=1+1=2 \\ y'''=0+y'' &\Rightarrow y'''(x_0)=0+2=2 \\ \cdots &\quad \cdots \quad\quad \cdots \end{aligned}$$

$$y(x)=1+1(x-0)+\frac{2(x-0)^2}{2!}+\frac{2(x-0)^3}{3!}+\frac{2(x-0)^4}{4!}+ERROR=1+x+x^2+\frac{x^3}{3}+\frac{x^4}{12}+ERROR$$

Gráfico de la Solución obtenida



Error en la Serie de Taylor

Tomando como base el desarrollo de Taylor:

$$y(x)=y(x_0)+y'(x_0)(x-x_0)+\frac{y''(x_0)(x-x_0)^2}{2!}+\frac{y'''(x_0)(x-x_0)^3}{3!}+\cdots$$

Si en el caso del **ejemplo anterior**, tomamos sólo 5 términos:

$$y(x)=1+h+h^2+\frac{h^3}{3}+\frac{h^4}{12}+ERROR$$

Por lo tanto, podemos expresar el **error** como:

$$ERROR=\frac{y^{(V)}(\xi)(x-x_0)^5}{5!}=\frac{y^{(V)}(\xi)h^5}{5!} \qquad para \quad 0<\xi<h$$

Error de la Serie de Taylor

En forma general, tomando hasta el término de $\mathbf{h^n}$ de la serie:

$$ERROR = \frac{y^{(n+1)}(\xi)(x-x_0)^{n+1}}{n+1!} = \frac{y^{(n+1)}(\xi)h^{n+1}}{n+1!} \qquad para \quad 0<\xi<h$$

- El error **disminuye** al reducir el **h**.
- Sin embargo, no es posible calcular $\mathbf{y^{(n+1)}(\xi)}$ pues se **desconoce** el valor de **ξ** y no siempre es posible acotar la derivada en el intervalo **[0,h]**, pues sólo se conoce su valor en $\mathbf{x_0}$.
- La serie de Taylor puede **truncarse** cuando los términos restantes **no poseen cifras significativas** con respecto a la precisión establecida, pues **no contribuyen** al valor final.

Ejemplo de la Serie de Taylor

Ecuación Diferencial

$$y' = x \cdot y^{\frac{1}{3}}$$
$$y(1) = 1 \;\Rightarrow\; x_0 = 1$$

Solución Exacta

$$y = \left(\frac{x^2+2}{3}\right)^{\frac{3}{2}}$$

x	Taylor (5 T)	Exacto	Error
1,00	1	1	0,0000000000
1,10	1,10682	1,1068166063	-0,0000033937
1,20	1,22787394	1,2278795936	0,0000056536
1,50	1,68618	1,6861706012	-0,0000093988
2,00	2,82846	2,8284271247	-0,0000328753
2,50	4,56042	4,5603590867	-0,0000609133
3,00	7,02123	7,0211321235	-0,0000978765
4,00	14,6971	14,6969384567	-0,0001615433
5,00	27,00022	27	-0,0002200000

Métodos Numéricos para resolver EDO

- ***Métodos Unipaso:***
  - Euler Simple y Modificado
  - Métodos de Runge-Kutta

- ***Métodos Multipaso:*** ***(no los analizaremos en este curso)***
  - Métodos explícitos (Adams-Bashfort)
  - Métodos implícitos (Adams-Moulton)

Método de Euler Simple

Se obtiene a partir de los dos primeros términos de la Serie de Taylor

$$y(x_0+h)=y(x_0)+y'(x_0)h+\frac{y''(\xi)h^2}{2!} \qquad x_0<\xi<x_{0+h}$$

ERROR LOCAL

- Es un método **sencillo**.
- Es necesario utilizar un valor **pequeño** de **h** para lograr cierta **exactitud.** *Error local* ***O(h²)***.
- El **error global** aumenta a medida que nos **alejamos** del **valor inicial**.

Leonhard Paul Euler

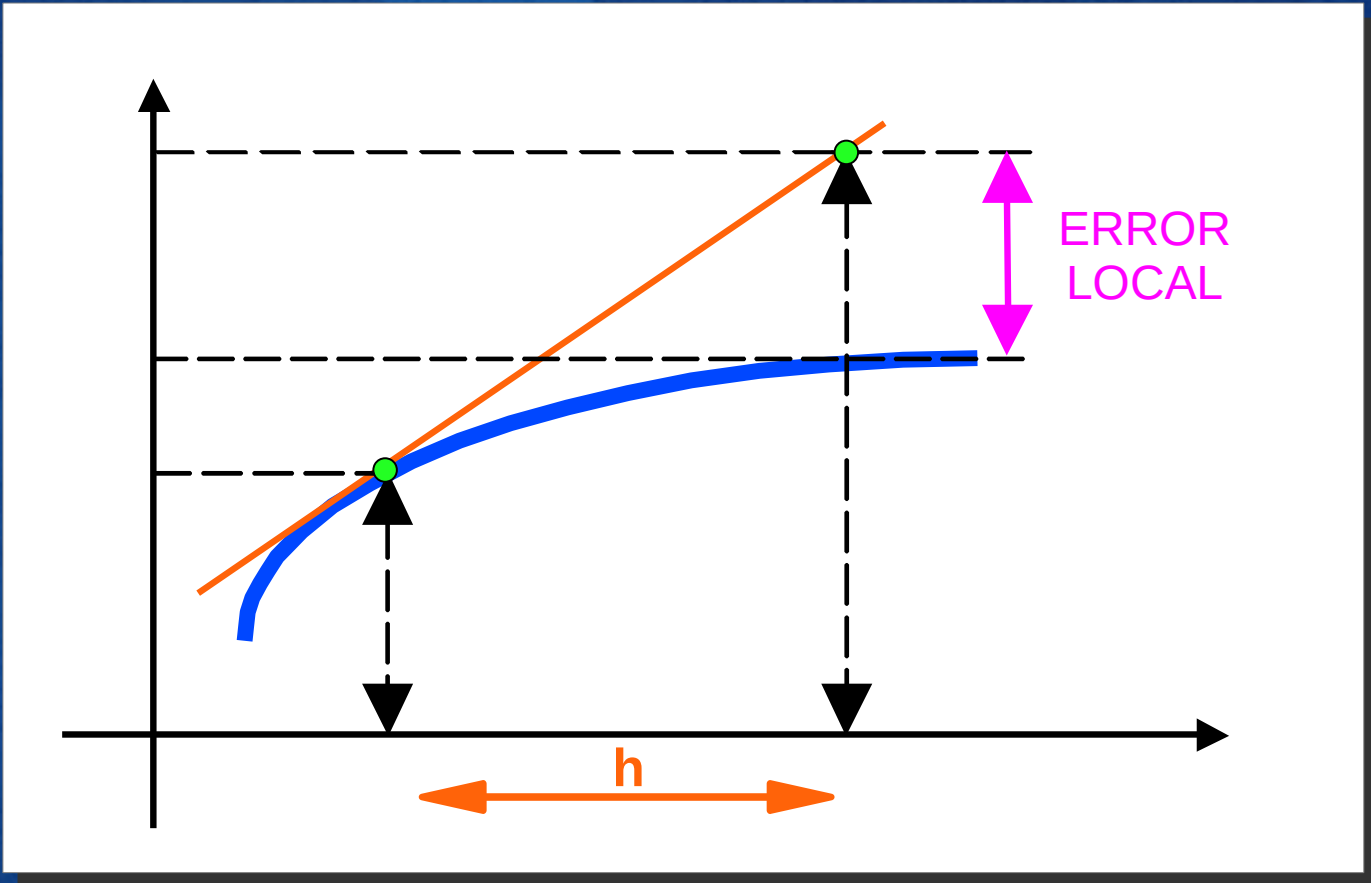
- Nació el 15 de abril de 1707 en Basilea, Suiza.
- Realizó importantes descubrimientos en áreas tan diversas como el cálculo o la teoría de grafos.
- Falleció el 18 de septiembre de 1783 en San Petesburgo, Rusia.
- En 2002 nombraron a un asteroide con su nombre, en su honor.

Representación gráfica

Esquema Iterativo

$$x_{n+1} = x_n + h$$
$$y_{n+1} = y_n + h * y'(x_n, y_n)$$

Se utiliza la pendiente que pasa por el extremo izquierdo del intervalo: $y'_n = f(x_n, y_n)$ para determinar el incremento de la función.



Implementación con vectores (I)

La implementación con vectores nos permite trabajar en una forma más compacta, simplificando su programación.

$$x_{n+1} = x_n + h$$
$$y_{n+1} = y_n + h * y'(x_n, y_n)$$

$$v_{n+1} = v_n + h \cdot v'(v_n)$$

v = | x | y |
v(0) v(1)

← Vector de variables

vp = | 1 | y' |
vp(0) vp(1)

← Vector de derivadas

Implementación con vectores (II)

$$v_{n+1} = v_n + h \cdot v'(v_n) \quad \Rightarrow \quad v = v + h \cdot vp$$

Vector de variables **Vector de derivadas**

v = | x | y |
v(0) v(1)

vp = | 1 | y' |
vp(0) vp(1)

v = | x | y | + | h | h*y' | = | x+h | y+h*y' |
v(0) v(1) vp(0) vp(1) v(0) v(1)

Definición de la Función Derivada

```
MODULE derivada

! Cantidad de Ecuaciones
INTEGER, PARAMETER :: cant_ec = 1

CONTAINS

FUNCTION v_prima(v)
!  Definición de la Ecuacion Diferencial
REAL(8), DIMENSION(0:cant_ec) :: v, v_prima

  v_prima(0) = 1.0
  v_prima(1) = v(0)*v(1)**(1/3.)

END FUNCTION v_prima

END MODULE
```

Ejemplo de implementación con la función:

$$v'(x,y) = x \cdot y^{\frac{1}{3}}$$

Euler Simple (Implementación)

```
SUBROUTINE EulerSimple(vi, cant_ec, max_iter, h)
!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
INTEGER iter
REAL(8), DIMENSION(0:cant_ec) :: v

  WRITE (*, '(I5, 2F10.6)') 0, vi

  v = vi
  DO iter = 1, max_iter
     v = v + h*v_prima(v)
     WRITE (*, '(I5, 2F10.6)') iter, v
  END DO

END SUBROUTINE EulerSimple
```

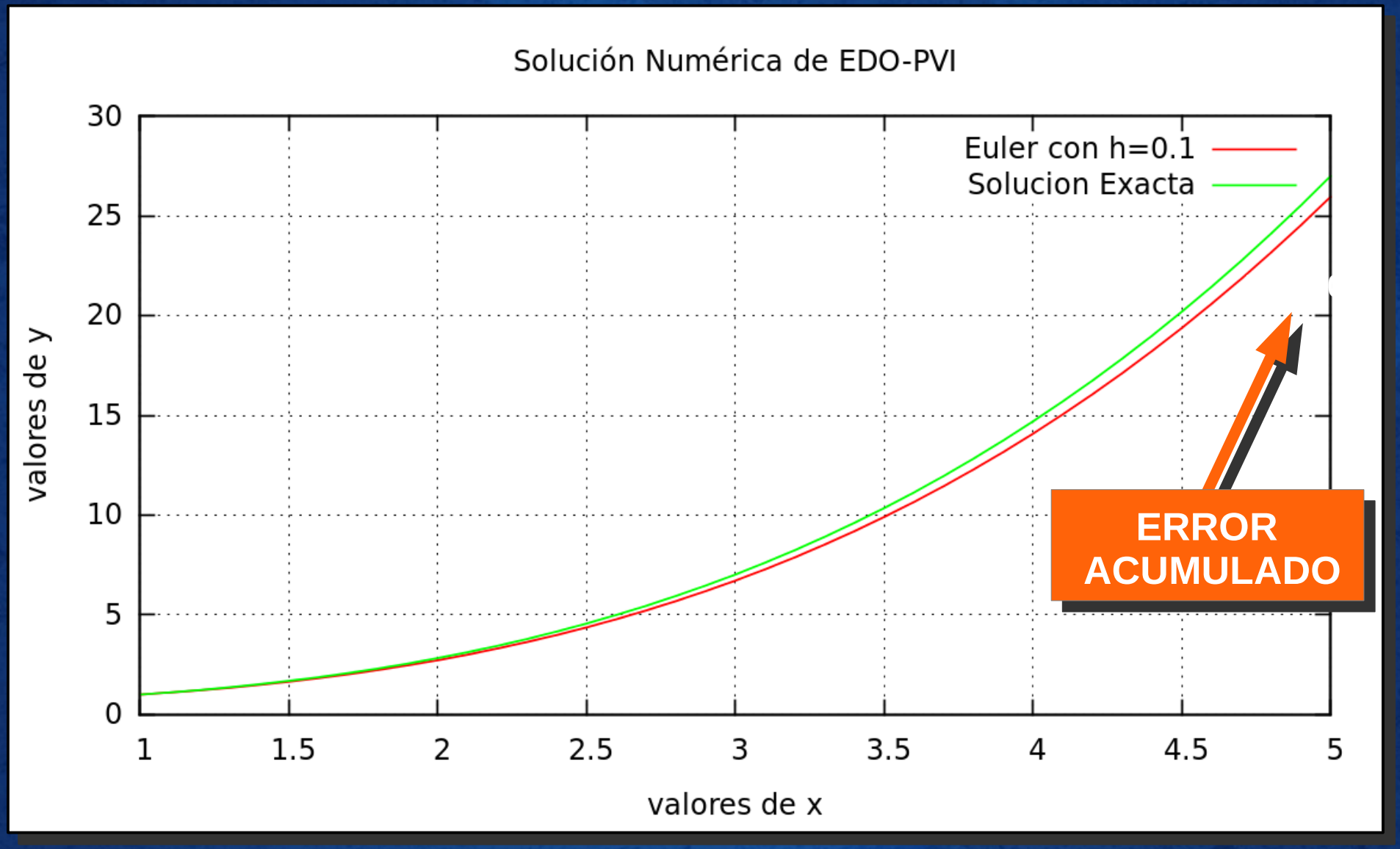
Influencia del valor de h

La **exactitud** de la solución numérica **aumenta** a medida que el paso **h disminuye**.

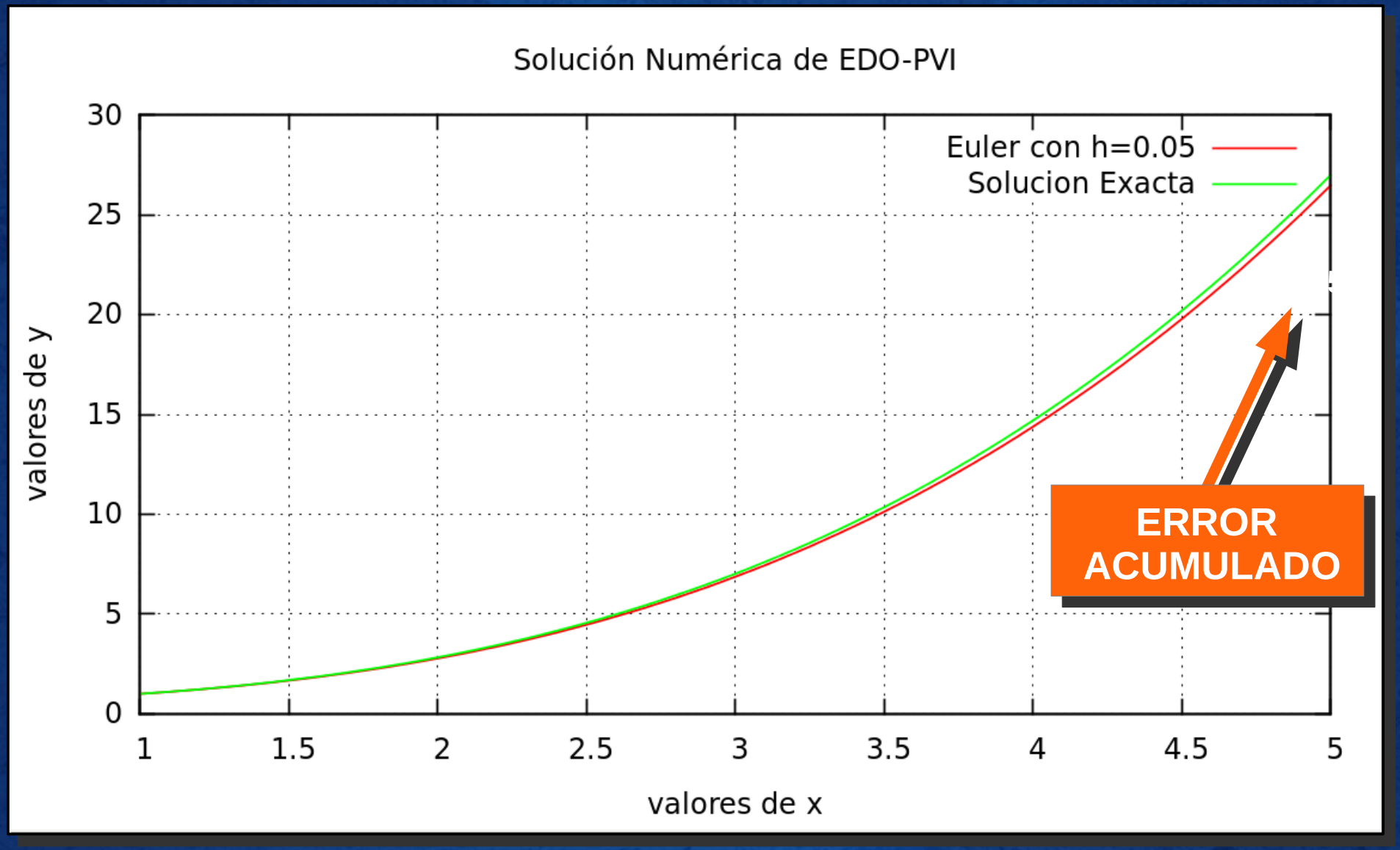
x	h=0,1	h=0,05	h=0,01	S.Exacta
1	1,00	1,00	1,00	1,00
2	2,72	2,78	2,82	2,83
3	6,71	6,87	6,99	7,02
4	14,08	14,39	14,63	14,70
5	25,96	26,48	26,89	27,00

Los métodos basados en la **fórmula de Taylor** usan una técnica que consiste en **dividir por 2 el paso h** (h = h/2) y comparar ambas soluciones obtenidas.

Solución obtenida por Euler con h= 0.1



Solución obtenida por Euler con h= 0.05



Método de Euler Modificado

Se obtiene a partir de los **tres** primeros términos de la **Serie de Taylor**.

$$y_{n+1} = y_n + h \cdot y'_n + h^2 \cdot \frac{y_n''}{2!} + h^3 \cdot \frac{y_n'''(\xi)}{3!} \qquad x_n < \xi < x_n + h$$

ERROR LOCAL

Sustituimos la **derivada segunda** por su aproximación en diferencias.

$$y_n'' \approx \frac{y'_{n+1} - y'_n}{h}$$

Error Local para Euler Modificado

Por lo tanto, nos queda la expresión:

$$y_{n+1} = y_n + h * y'_n + \frac{h^2}{2!} * \frac{(y'_{n+1} - y'_n)}{h} + O(h^3)$$

Simplificando:

$$y_{n+1} = y_n + h * y'_n + h * \frac{(y'_{n+1} - y'_n)}{2} + O(h^3)$$

Finalmente obtenemos:

$$y_{n+1} = y_n + h * \frac{(y'_{n+1} + y'_n)}{2} + O(h^3)$$

ERROR LOCAL

El **error local** es del orden de $\mathbf{h^3}$

Método de Euler Modificado (Heun)

Se obtiene una mejora en la exactitud con respecto de **Euler Simple**, considerando en el cálculo, el **promedio** de dos derivadas, en lugar de considerar una sola.

$$\tilde{y}_{n+1} = y_n + h \cdot y'(x_n, y_n)$$

$$y_{n+1} = y_n + h \cdot \frac{\left(y'(x_n, y_n) + y'(x_{n+1}, \tilde{y}_{n+1})\right)}{2}$$

La derivada en el **extremo derecho** del intervalo se calcula en función del valor **estimado** de $\mathbf{y_{n+1}}$. Este valor se estima por medio del método de **Euler Simple**.

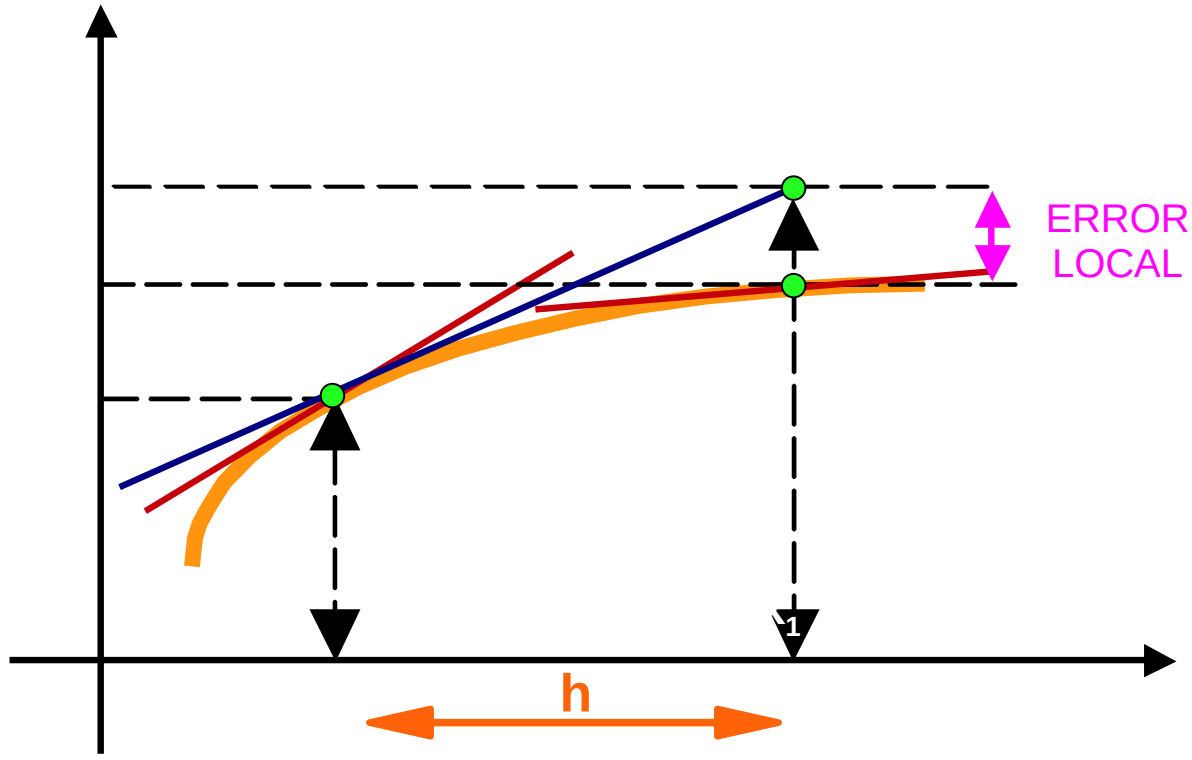
Representación gráfica

Esquema Iterativo

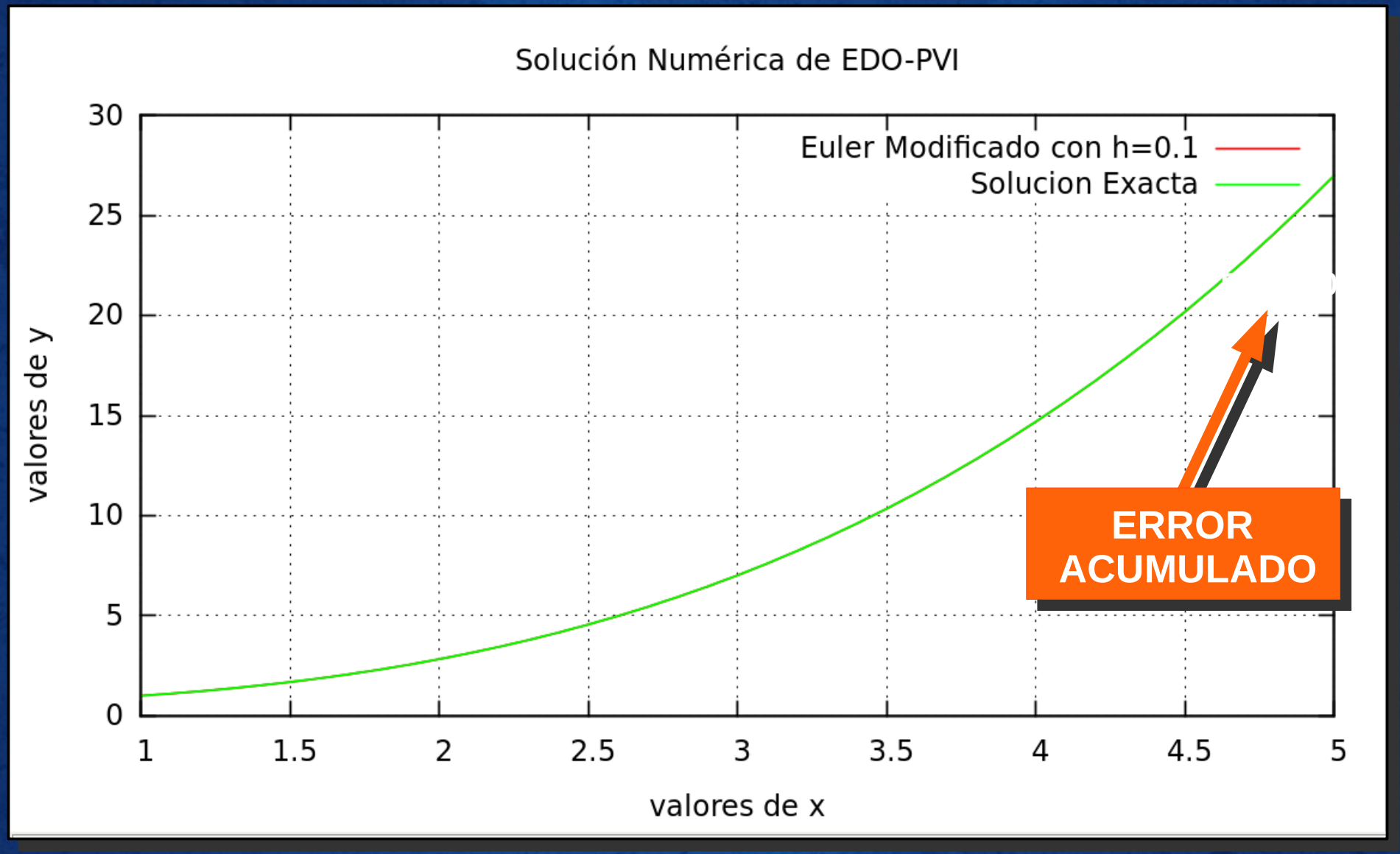
$$\tilde{y}_{n+1} = y_n + h \cdot y'(x_n, y_n)$$

$$y_{n+1} = y_n + h \cdot \frac{(y'(x_n, y_n) + y'(x_{n+1}, \tilde{y}_{n+1}))}{2}$$

Se utiliza el promedio de las pendientes que pasan por ambos extremos del intervalo para determinar el incremento de la función.



Solución obtenida por Euler Modificado con h = 0.1



Euler Modificado (Implementación)

```
SUBROUTINE EulerModificado(vi, cant_ec, max_iter, h)
!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
INTEGER iter
REAL(8), DIMENSION(0:cant_ec) :: v, vp

  WRITE (*, '(I5, 2F10.6)') 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, 2F10.6)') iter, v
  END DO

END SUBROUTINE EulerModificado
```

Error Local y Global

- **Error Local**

  Es el error que se produce en cada paso. Por lo tanto se considera que el valor previo es un valor exacto.

- **Error Global**

  Es el error acumulado en **n** pasos. En este caso se trata de la diferencia entre el valor calculado luego de **n** pasos y el valor exacto.

Orden de un Método Numérico

Tomemos como ejemplo al **Método de Euler Simple**

$$y(x_n+h) = y(x_n) + y'(x_n)\cdot h + \frac{y''(\xi)h^2}{2!}$$

$y(x_{n+1})$ y_{n+1} Error de truncamiento en un paso

- Si un método tiene un error de truncamiento **por paso** de la forma $e_{n+1}(h) = C \cdot h^{(n+1)} \cdot y^{(n+1)}(\xi_i)$ se puede demostrar que su error de truncamiento acumulado $E_n(h) = y(x_n) - y_n$ es igual a:

$$E_n(h) = \tilde{C} \cdot h^n \cdot y^{(n+1)}(\xi_i).$$

- Por lo tanto, diremos que un método numérico es de **orden n** si su error de **truncamiento acumulado** $E_n(h)$ es $O(h^n)$.

Métodos más eficientes

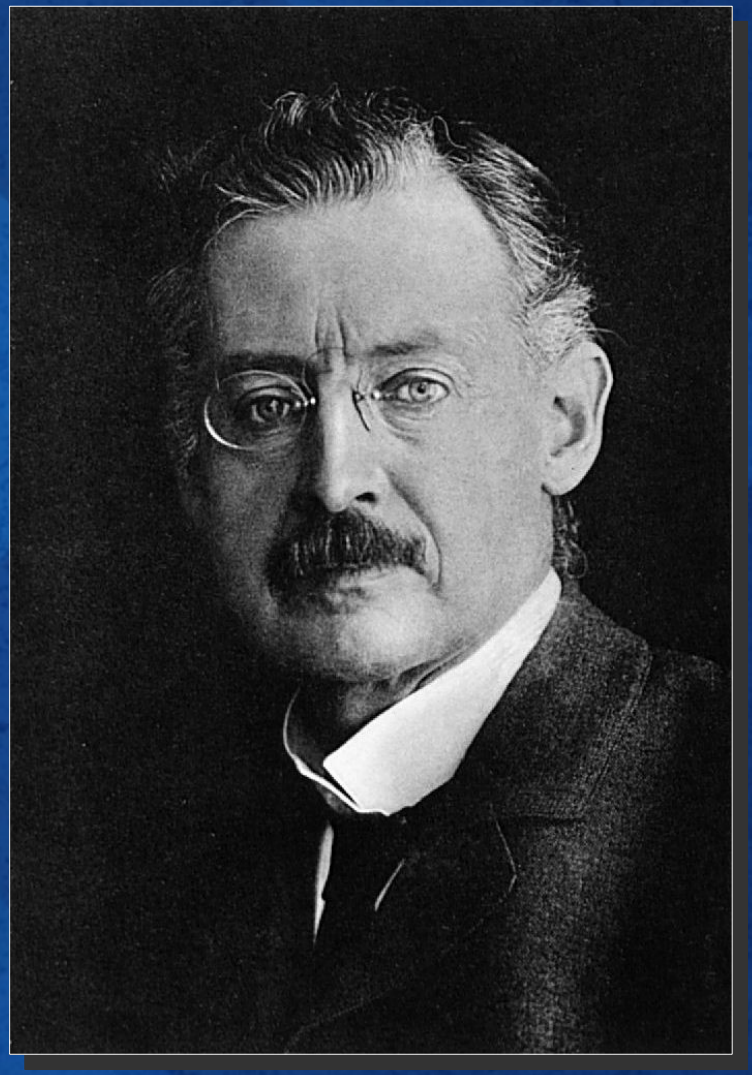
- **Runge** y **Kutta** desarrollaron un conjunto de métodos de resolución de ecuaciones diferenciales ordinarias, que mejoran notablemente su **eficiencia**.

- Uno de los más utilizados es el método de **Runge-Kutta de 4to. Orden**.

- Posteriormente se desarrollaron métodos de mayor orden, que incluyen la **estimación** del **error** producido en **cada paso**.

Carl David Tolmé Runge

- Nació el 30 de agosto de 1856 en Bremen, Alemania.
- En 1901 desarrolló, en colaboración con Martin Wilhelm Kutta, un método para resolver ecuaciones diferenciales ordinarias.
- Falleció el 3 de enero de 1927 en Göttingem, Alemania.
- Un cráter en la luna lleva su nombre.



Martin Wilhelm Kutta

- Nació el 3 de noviembre de 1867 en Pitschen, Alemania.
- En 1901 desarrolló, en colaboración con Karl Runge, un método para resolver ecuaciones diferenciales ordinarias.
- Falleció el 25 de diciembre de 1944 en Fürstenfeldbruck, Alemania.
- Fue profesor de la universidad de Stuttgart desde 1911 hasta su retiro en 1935.

Métodos de Runge-Kutta (2do. Orden)

Escribimos el incremento de **y** como un promedio ponderado de **dos** estimaciones de **Δy**, a las que denominamos **K1** y **K2**.

$$y_{n+1} = y_n + a \cdot K_1 + b \cdot K_2 \qquad \textbf{[1]}$$

Siendo:

$$K_1 = h \cdot f\left(x_n, y_n\right)$$
$$K_2 = h \cdot f\left(x_n + \alpha \cdot h, y_n + \beta \cdot K_1\right) \qquad \textbf{[2]}$$

El problema consiste en encontrar un esquema para determinar los valores de **a, b, α** y **β**.

Métodos de Runge-Kutta (2do. Orden)

$$y_{n+1} = y_n + h * f(x_n, y_n) + \frac{h^2 * f'(x_n, y_n)}{2!} + \cdots \qquad [3]$$

Sabiendo que:

$$f' = \frac{df}{dx} \approx f_x + f_y \frac{dy}{dx} = f_x + f_y f$$

$$y_{n+1} = y_n + h * f_n + \frac{h^2}{2}(f_x + f_y f)_n \qquad [4]$$

Métodos de Runge-Kutta (2do. Orden)

Si sustituimos las definiciones de K_1 y K_2 obtenemos:

$$y_{n+1} = y_n + a \cdot h \cdot f(x_n, y_n) + b \cdot h \cdot f(x_n + \alpha \cdot h;\ y_n + h \cdot \beta \cdot f(x_n, y_n)) \quad \textbf{[5]}$$

Expandiendo el último término en **Serie de Taylor** y conservando sólo los términos de la **1ra. derivada**, nos queda:

$$f(x_n + \alpha \cdot h;\ \ y_n + h \cdot \beta \cdot f(x_n,\ \ y_n)) \approx (f + \alpha \cdot h \cdot f_x + \beta \cdot h \cdot f_y \cdot f)_n \quad \textbf{[6]}$$

Por último sustituyendo y agrupando convenientemente:

$$y_{n+1} = y_n + a \cdot h \cdot f_n + b \cdot h \cdot (f + \alpha \cdot h \cdot f_x + \beta \cdot h \cdot f_y f)_n$$

$$y_{n+1} = y_n + (a+b) \cdot h \cdot f_n + h^2 \cdot (\alpha \cdot b \cdot f_x + \beta \cdot b \cdot f_y f)_n$$

Métodos de Runge-Kutta (2do. Orden)

Igualando la expresión anterior con la [4],

obtenemos el siguiente sistema de ecuaciones:

SISTEMA DE ECUACIONES

$$a+b=1$$

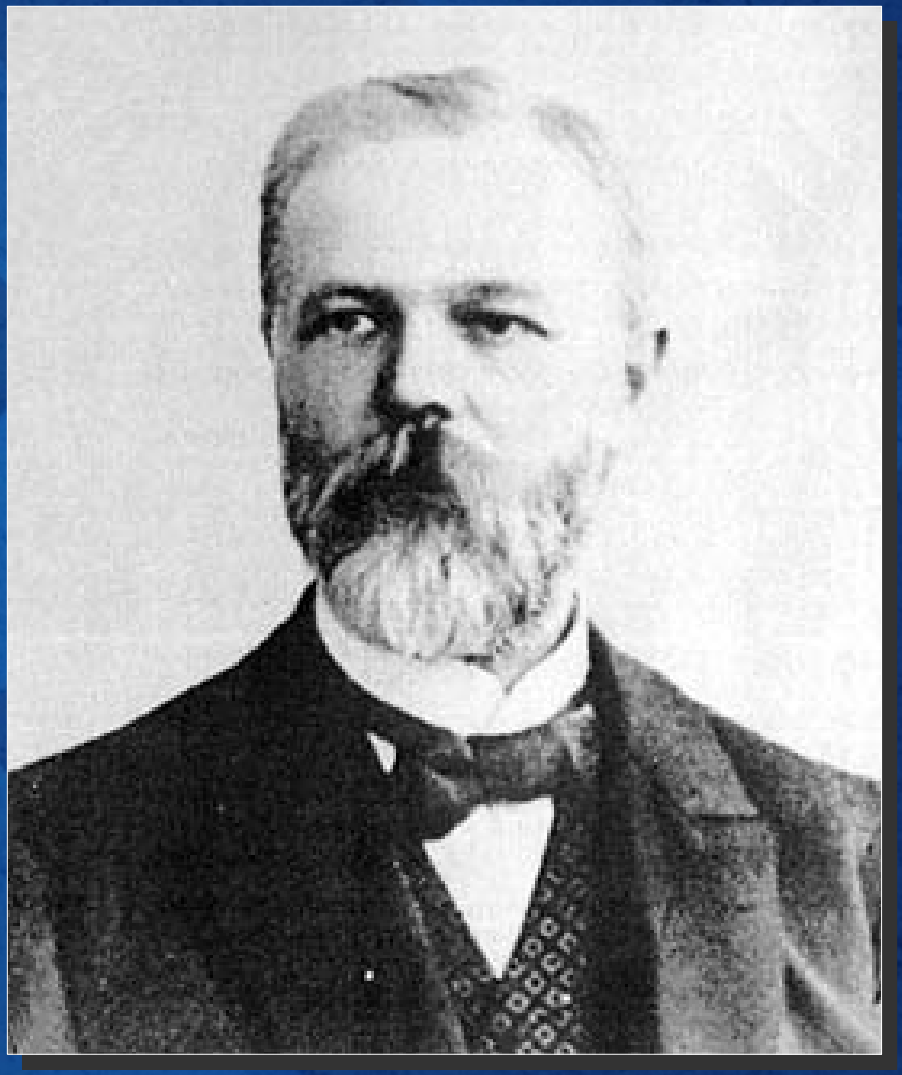
$$\alpha * b=\frac{1}{2}$$

$$\beta * b=\frac{1}{2}$$

Y como tenemos **3 ecuaciones** y **cuatro incógnitas**, podemos **elegir arbitrariamente** uno de los valores y calcular los restantes.

Karl L. W. Heun

- Nació el 3 de abril de 1859 en Wiesbaden, Alemania.
- En 1889 desarrolló, el método de resolución de ecuaciones diferenciales que lleva su nombre.
- Falleció el 10 de enero de 1929 en Karlsruhe, Alemania.



Método de Heun (Euler Modificado)

Por ejemplo, si elegimos **a=1/2** obtenemos: **b=1/2** y **α=β=1**.

Estos valores conducen a la fórmula del método de **Euler Modificado** ó **Método de Heun**:

$$\tilde{y}_{n+1} = y_n + h \cdot f(x_n, y_n)$$

$$y_{n+1} = y_n + h \cdot \frac{(f(x_n, y_n) + f(x_{n+1}, \tilde{y}_{n+1}))}{2}$$

(*) Cómo la derivada en $\mathbf{x_{n+1}}$ es desconocida, se utiliza una aproximación en su lugar.

Euler Mejorado

Otra variante se logra si elegimos el valor **b=1** y obtenemos: **a=0** y **α=β=1/2**.

Estos valores conducen a la fórmula del método de **Euler Mejorado**:

$$\tilde{y}_{n+\frac{1}{2}} = y_n + \frac{h}{2} * f(x_n, y_n)$$

$$y_{n+1} = y_n + h * f(x_n + \frac{h}{2}, \ \tilde{y}_{n+\frac{1}{2}})$$

(*) Cómo la derivada en $\mathbf{x_{n+1/2}}$ es desconocida, se utiliza una aproximación en su lugar.

Métodos de Runge-Kutta (4to. Orden)

- A medida que el **orden del método** aumenta, también aumenta la **complejidad matemática** de la deducción del mismo ya que deben considerarse más términos de la **Serie de Taylo**r.

- Por ejemplo, si consideramos los términos hasta (h^4) obtendremos un sistema de **11 ecuaciones** y **13 incógnitas**.

- De la misma forma que en el caso de ***2do. Orden***, si se tienen menos ecuaciones que incógnitas, es necesario elegir **arbitrariamente** dos valores, para poder determinar los valores de los coeficientes.

Métodos de Runge-Kutta (4to. Orden)

Por los motivos anteriormente expuestos, simplemente presentaremos la **fórmula final** de cálculo de los coeficientes y del **próximo valor**.

$$k_1 = h \cdot f(x_n, y_n)$$

$$k_2 = h \cdot f\left(x_n + \frac{h}{2},\ y_n + \frac{k_1}{2}\right)$$

$$k_3 = h \cdot f\left(x_n + \frac{h}{2},\ y_n + \frac{k_2}{2}\right)$$

$$k_4 = h \cdot f(x_n + h,\ y_n + k_3)$$

$$y_{n+1} = y_n + \frac{1}{6} \cdot (k_1 + 2 \cdot k_2 + 2 \cdot k_3 + k_4)$$

Métodos de Runge-Kutta (4to. Orden)

Orden de los Errores

- Este método tiene un ***Error Local*** del orden de (h^5) y un ***Error Global*** del orden de (h^4). Si comparamos con los métodos de ***2do. Orden***, se observa una importante mejora.

- De la misma forma que en el caso de ***2do. Orden***, como se tienen menos ecuaciones que incógnitas, es necesario elegir **arbitrariamente** dos valores, para poder determinar los valores de los coeficientes.

Métodos de Runge-Kutta (4to. Orden)

```
SUBROUTINE RungeKutta(vi, cant_ec, max_iter, h)
!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
INTEGER iter
REAL(8), DIMENSION(0:cant_ec) :: v, k1, k2, k3, k4

  WRITE (*, '(I5, 2F10.6)') 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 (*, '(I5, 2F10.6)') iter, v
  END DO
END SUBROUTINE RungeKutta
```

Mejoras a los Métodos de Runge-Kutta

- La idea básica consiste en calcular dos estimaciones de Runge-Kutta de diferente orden para calcular el nuevo valor de la función.

- La diferencia entre las dos estimaciones puede utilizarse para estimar el error en el método de orden menor.

- Por ejemplo, en Runge-Kutta-Merson se calcula la diferencia entre una estimación de 5to.orden y una de 4to.

- En cambio en Runge-Kutta-Fehlberg, se realizan 6 evaluaciones funcionales por iteración y se calcula una estimación del error.

Métodos de Runge-Kutta-Fehlberg

El método de Runge-Kutta de **orden 6to**. También conocido como método de **Runge-Kutta-Fehlberg**, incorpora una fórmula para la **estimación del error** en **cada paso**.

$$k_1 = h \cdot f(x_n, y_n)$$

$$k_2 = h \cdot f\left(x_n + \frac{h}{4},\ y_n + \frac{k_1}{4}\right)$$

$$k_3 = h \cdot f\left(x_n + \frac{3h}{8},\ y_n + \frac{3k_1}{32} + \frac{9k_2}{32}\right)$$

$$k_4 = h \cdot f\left(x_n + \frac{12h}{13},\ y_n + \frac{1932k_1}{2197} - \frac{7200k_2}{2197} + \frac{7296k_3}{2197}\right)$$

$$k_5 = h \cdot f\left(x_n + h,\ y_n + \frac{439k_1}{216} - 8k_2 + \frac{3680k_3}{513} - \frac{845k_4}{4104}\right)$$

$$k_6 = h \cdot f\left(x_n + \frac{h}{2},\ y_n - \frac{8k_1}{27} + 2k_2 - \frac{3544k_3}{2565} - \frac{1859k_4}{4104} - \frac{11k_5}{40}\right)$$

$$y_{n+1} = y_n + \frac{25k_1}{216} + \frac{1408k_3}{2565} + \frac{2197k_4}{4104} - \frac{k_5}{5}$$

$$E = \frac{k_1}{360} - \frac{128k_3}{4275} - \frac{2197k_4}{75240} + \frac{k_5}{50} + \frac{2k_6}{55}$$

Métodos de Runge-Kutta-Fehlberg

```
SUBROUTINE RungeKuttaFehlberg(vi, cant_ec, max_iter, h)
!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
INTEGER iter
REAL(8), DIMENSION(0:cant_ec) :: v, k1, k2, k3, k4, k5, k6, e

  WRITE (*, '(I5, 3F10.6)') 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 (*, '(I5, 3F10.6)') iter, v, e
  END DO
END SUBROUTINE RungeKuttaFehlberg
```

Preguntas . . .