Ecuaciones diferenciales ordinarias

Curso de Física Computacional

M. en C. Gustavo Contreras Mayén

Facultad de Ciencias - UNAM

21 de marzo de 2018

Mejora en el método de Euler

Una forma práctica de superar la baja precisión del método de Euler es elevar el orden de la aproximación aplicando su variante llamada método predictor - corrector de Euler, que opera con dos estimaciones de solución en cada paso de propagación.

Método predictor - corrector

$$\overline{y}_{m+1} = y_m + h\, f(t_m, y_m), \quad m = 1, 2, 3, \ldots \tag{1}$$

$$y_{m+1} = y_m + \frac{h}{2}\left[f(t_m, y_m) + f(t_m + h, \overline{y}_{m+1})\right] \tag{2}$$

donde

1. $\overline{y}_{m+1}$ es el valor predictor.

Método predictor - corrector

$$\overline{y}_{m+1} = y_m + h\, f(t_m, y_m), \quad m = 1, 2, 3, \ldots \tag{1}$$

$$y_{m+1} = y_m + \frac{h}{2}\left[f(t_m, y_m) + f(t_m + h, \overline{y}_{m+1})\right] \tag{2}$$

donde

1. $\overline{y}_{m+1}$ es el valor predictor.
2. y_{m+1} es el valor corregido.

Valor predictor

El valor predictor de la solución $\overline{y}_{m+1}$ se obtiene del método de Euler, de la expresión predictora (ec. 1).

Se utiliza para estimar la propagación de la derivada $f(t_m + h, \overline{y}_{m+1})$

Valor corregido

El valor de la solución corregido y_{m+1} se obtiene de la fórmula de corrección (ec. 2), que utiliza el promedio de la derivada estimada

$$\frac{[f(t_m, y_m) + f(t_m + h, \overline{y}_{m+1})]}{2}$$

en el intervalo $[x_m, x_{m+1}]$

Casos especiales de Runge-Kutta

Veremos más adelante que los métodos de Euler básicos y el predictor - corrector son casos particulares de una familia completa de algoritmos, conocidos como métodos de Runge-Kutta.

Casos especiales de Runge-Kutta

El método básico de Euler es equivalente al método de primer orden de Runge-Kutta (RK1).

El algoritmo predictor - corrector de Euler es $O(h^2)$, es equivalente al método de Runge-Kutta de segundo orden (RK2).

Casos especiales de Runge-Kutta

La precisión superior del algoritmo predictor-corrector Euler obviamente se produce a expensas de un doble número de evaluaciones de las funciones del lado derecho $f(x, y)$ de las EDO.

Estabilidad y precisión de RK1

Para revisar la precisión y estabilidad de los métodos de Euler, consideremos el siguiente problema

$$y'' + y = 0 \tag{3}$$

$$y(0) = y_0, \quad y'(0) = y'_0 \tag{4}$$

Consideraciones

Obtén la solución a la EDO2 en el intervalo $0 \leq t \leq 100$, con un paso $ht = 0.05$.

Solución

La solución general de este problema es

$$y(t) = A\ \sin t + B\ \cos t$$

donde las constantes dependen de los valores iniciales $y(0), y_0'$.

Usando la notación

Usando la notación descrita anteriormente:

$$y_1 \equiv y, \quad y_2 \equiv y'$$

por lo que entonces el problema se expresa

$$F(x,y) = \begin{bmatrix} y_1' & = & y_2 \\ y_2' & = & -y1 \end{bmatrix} \tag{5}$$

Usando la notación

Para las condiciones iniciales:

$$\begin{bmatrix} y_1(0) & = & y_0 \\ y_2(0) & = & y_0' \end{bmatrix} \tag{6}$$

Solución exacta

La solución exacta del problema con los valores iniciales dados:

$$y_1(0) = 0 \quad y_2 = 1$$

es

$$y_1(t) = \sin t, \quad y_2(t) = \cos t$$

Código a utilizar

Usaremos el módulo **`moduloEuler`** que contiene las funciones **`Euler`** y **`EulerPC`**.

Vamos a comparar las dos soluciones.

Primera solución

Código 1: Método de Euler

```
1
2  def Euler(t, ht, y, n, Func):
3      f = [0] * (n + 1)
4
5      Func(t, y, f)
6
7      for i in range(1, n + 1):
8          y[i] += ht * f[i]
9
10     return y
```

Código I

Código 2: Código de Euler

```
1  from moduloEuler import Euler,
       EulerPC
2
3  def Func(t, y, f):
4      f[1] =  y[2]
5      f[2] = -y[1]
6
7  y0 = 0e0; dy0 = 1e0
8  tmax = 100e0
9  ht = 0.05e0
10
11 n = 2
```

Código II

```
12 y = [0]*(n + 1) #componentes de la
      solucion
13
14
15 t = 0e0
16 y[1] = y0
17 y[2] = dy0
18
19
20 salida1 = open("solucioneuler.txt","
      w")
21 salida1.write("      t          y1
          y2       check\n")
```

Código III

```
22
23 t = 0e0
24 y[1] = y0; y[2] = dy0
25
26 salida1.write(("{0:10.5f} {1:10.5f}
      {2:10.5f} {3:10.5f}\n"). \
27                format(t, y[1], y[2], y[1]
       * y[1] + y[2] * y[2]))
28
29 while (t + ht <= tmax):
30     Euler(t, ht, y, n, Func)
31     t += ht
32
```

Código IV

```
33     salida1.write(("{0:10.5f}{1:10.5f}{2:10.5f}{3:10.5f}\n"). \
34                    format(t,y[1], y[2], y[1] * y[1] + y[2] * y[2]))
35 salida1.close()
```

Código para graficar I

Cómo se genera un archivo de datos, tendremos que recuperar los mismos y luego graficar.

Código 3: Código para graficar

```
1
2 import matplotlib.pyplot as plt
3
4 with open('solucioneuler.txt') as f:
5     next(f)
6     lines = f.readlines()
7     t = [float(line.split()[0]) for
    line in lines]
```

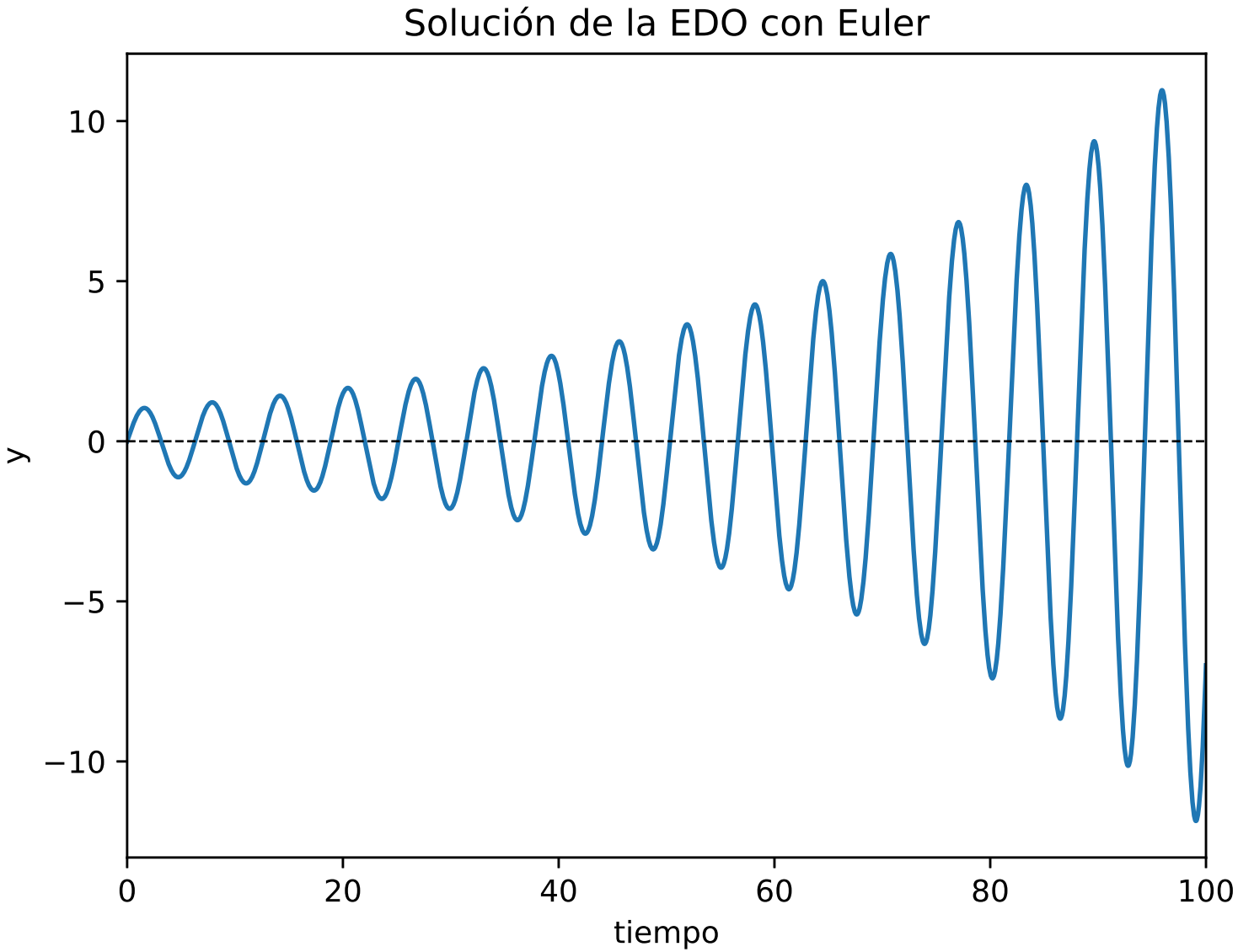
Código para graficar II

```
8     y = [float(line.split()[1]) for
      line in lines]
9     y2 = [float(line.split()[2]) for
      line in lines]
10
11 plt.figure(1)
12 plt.plot(t, y)
13 plt.axhline(y=0, lw=0.75, ls='dashed
      ', color='k')
14 plt.title('Solucion de la EDO con
      Euler')
15 plt.xlabel('tiempo')
16 plt.ylabel('y')
```

Código para graficar III

```
17 plt.xlim([0,100])
18 plt.show()
```

Gráfica de la solución



¿Qué está ocurriendo?

Sabemos que la solución de la EDO2 es periódica, no hay factores de amortiguamiento ni otro elemento que modifique la amplitud de la solución.

Pero vemos en la gráfica que aumenta la amplitud conforme transcurre el tiempo.

Diagrama espacio - fase

Otra manera de revisar si hay una inconsistencia con nuestros resultados, la encontraremos al graficar el estado fase.

Cómo debe de ser

El estado fase de la EDO2 supone que no hay pérdida de energía, por lo que tendríamos una gráfica cerrada y de trazo uniforme.

Veamos el resultado

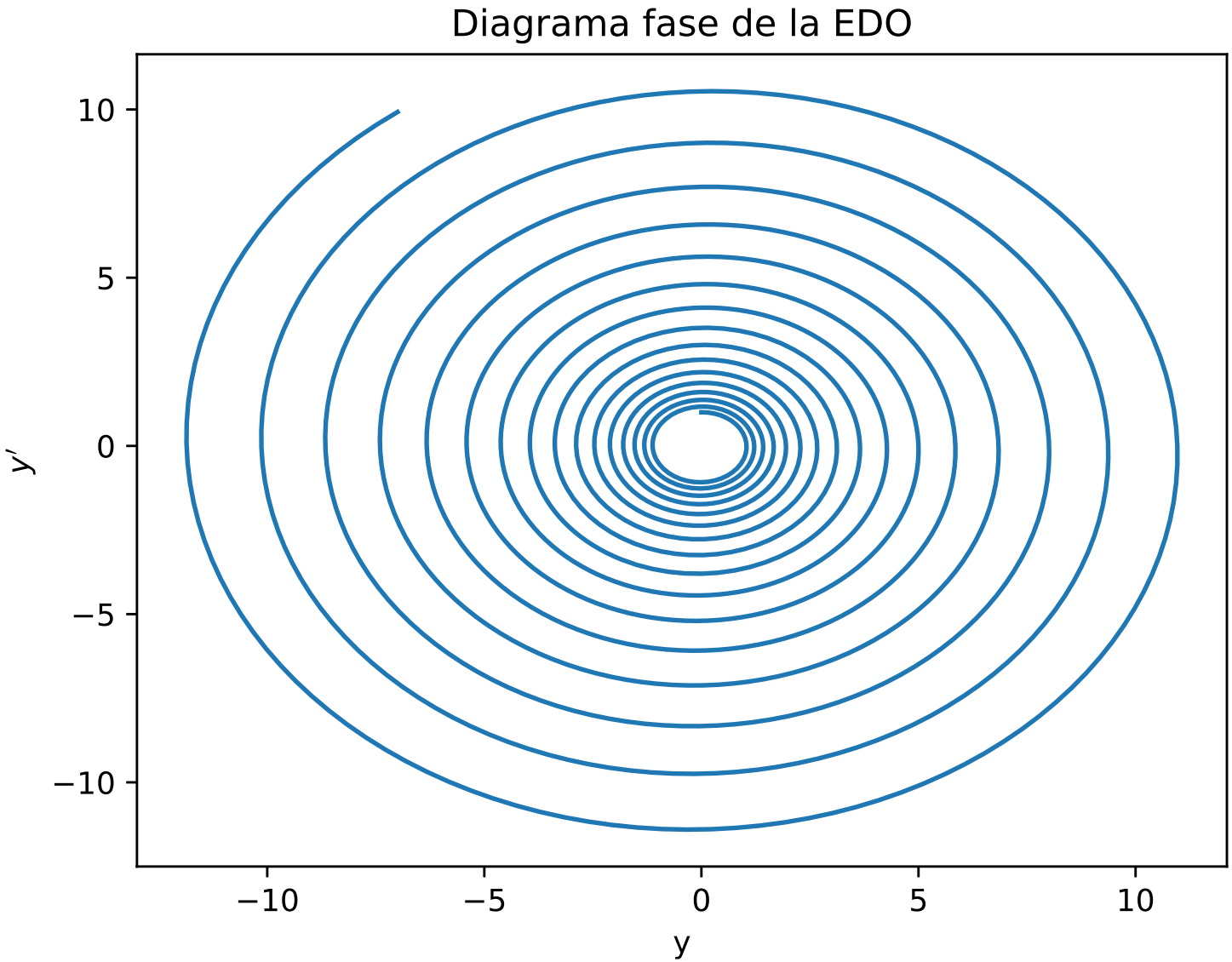
Código para graficar I

Cómo se genera un archivo de datos, tendremos que recuperar los mismos y luego graficar.

Código 4: Código para graficar

```
1 plt.figure(2)
2 plt.plot(y, y2)
3 plt.title('Diagrama fase de la EDO')
4 plt.xlabel('y')
5 plt.ylabel('$y^{\prime}$')
6
7 plt.show()
```

Gráfica del espacio fase



Resultado

El estado fase presenta una trayectoria abierta, por lo que no hay correspondencia con el fenómeno.

El método de Euler no funciona con EDO2 del tipo estudiado.

Usando el predictor - corrector I

Ahora usaremos el algoritmo de Euler predictor - corrector:

Código 5: Método predictor - corrector

```
1 def EulerPC(t, ht, y, n, Func):
2     f1 = [0] * (n + 1)
3     f2 = [0] * (n + 1)
4     yt = [0] * (n + 1)
5 
6     Func(t, y, f1)
7 
8     for i in range(1, n + 1):
9         yt[i] = y[i] + ht * f1[i]
```

Usando el predictor - corrector II

```
10
11      Func(t + ht, yt, f2)
12
13      ht2 = ht/2e0
14
15      for i in range(1, n + 1):
16          y[i] += ht2 * (f1[i] + f2[i])
17
18      return y
```

Cambio en el nombre de archivo

Ajusta en la línea del código donde se le da un nombre al archivo de datos por lo siguiente:

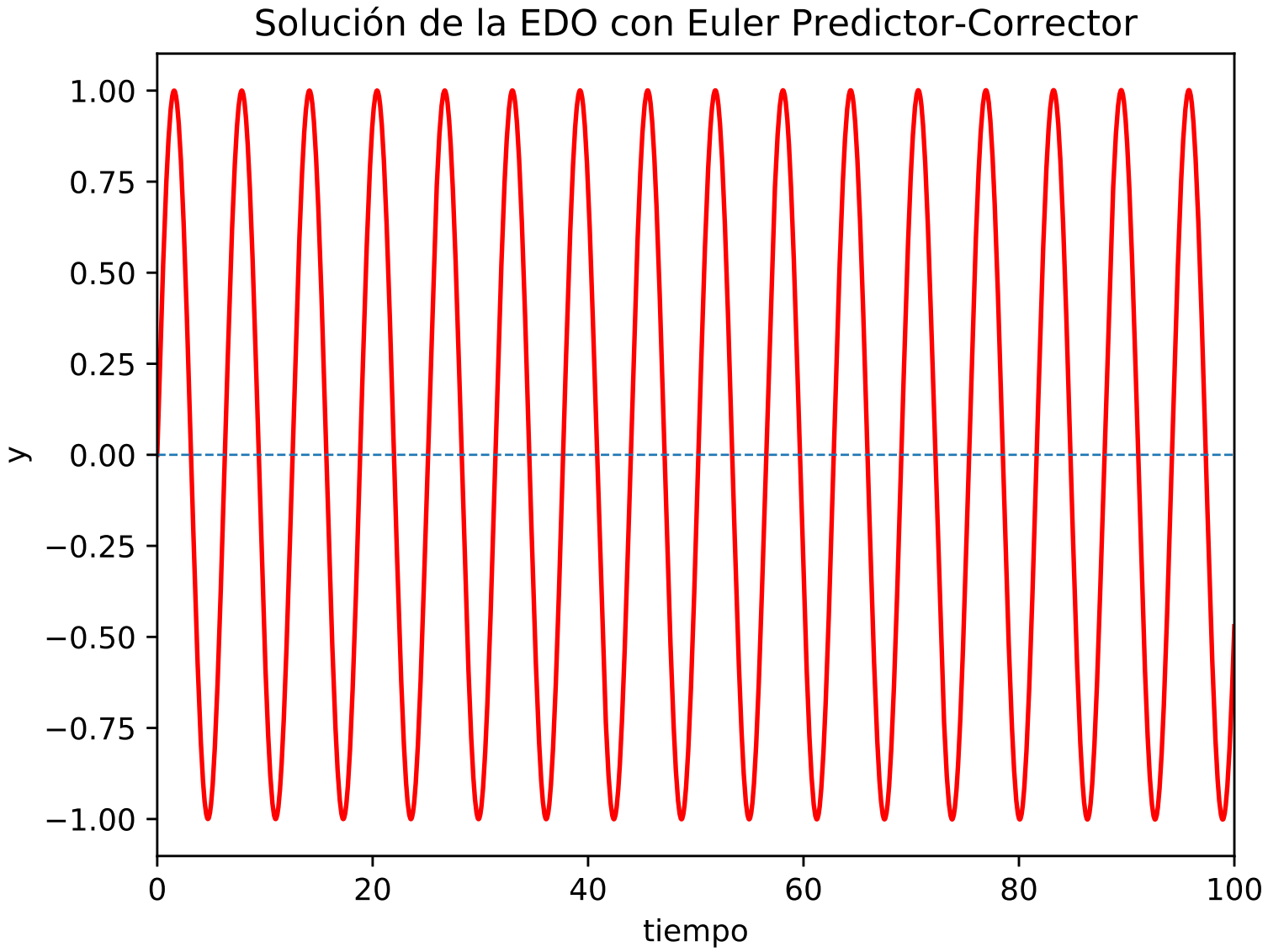
```
open("solucion_euler_pc.txt","w")
```

Ajuste en la graficación

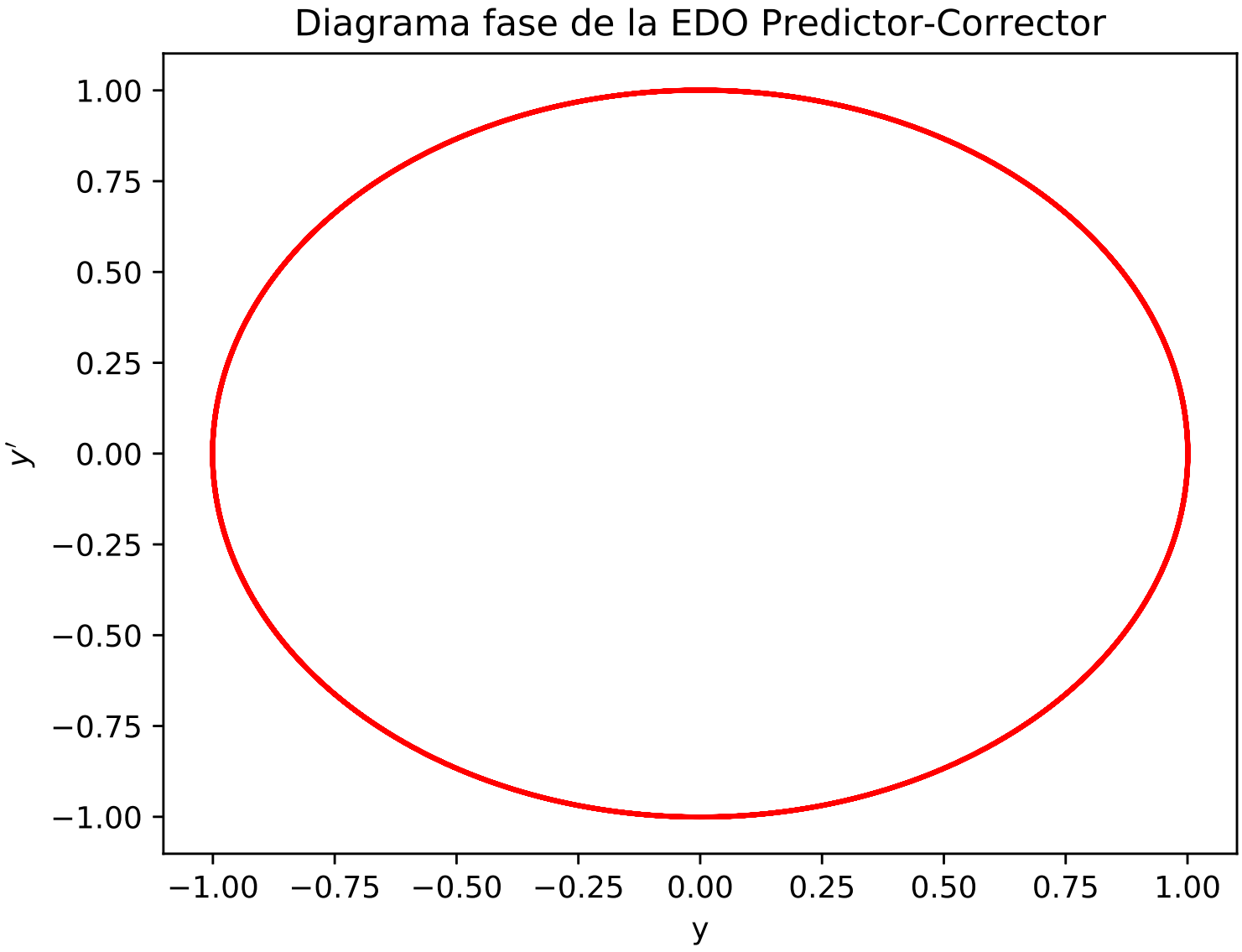
Realiza el mismo ajuste de nombre de archivo en la línea que abre el archivo de datos:

```
open("solucion_euler_pc.txt") as f:
```

Solución Euler predictor - corrector



Solución Euler predictor - corrector



Conclusión

Vemos que el método de Euler predictor - corrector, corrige la falla en la estabilidad de la solución.

Tendrás que revisar si el método permite una solución estable para la EDO.

Utilidad de los métodos RK

El objetivo de los métodos de Runge-Kutta (RK) es eliminar la necesidad de la diferenciación repetida de las ecuaciones diferenciales.

Utilidad de los métodos RK

Dado que la diferenciación repetida no está implicada en la fórmula de integración en la serie de Taylor de primer orden

$$\mathbf{y}(x+h) = \mathbf{y}(x) + \mathbf{y}'(x)\,h = \mathbf{y}(x) + \mathbf{F}(x,\mathbf{y})\,h$$

Método RK1

Se considera como el método Runge-Kutta de primer orden (RK1), también llamado, *método de Euler*.

Ya vimos que el error debido al truncamiento es bastante, no se usa en la práctica común.

Método de Runge-Kutta de segundo orden

Para obtener el método **RK2**, suponemos una fórmula de integración del tipo

$$\mathbf{y}(x+h) = \mathbf{y}(x) + c_0\,\mathbf{F}(x,\mathbf{y})\,h + c_1\mathbf{F}[x+p\,h, \mathbf{y}+q\,h\mathbf{F}(x,\mathbf{y})]\,h$$

Método de Runge-Kutta de segundo orden

Debemos de encontrar los parámetros c_0, c_1, p y q de tal forma que se parezca a la siguiente serie de Taylor

$$\begin{aligned}\mathbf{y}(x+h) &= \mathbf{y}(x) + \mathbf{y}'(x)\,h + \frac{1}{2!}\mathbf{y}''(x)\,h^2 + O(h^3) \\ &= \mathbf{y}(x) + \mathbf{F}(x,\mathbf{y})\,h + \frac{1}{2}\mathbf{F}'(x,\mathbf{y})\,h^2 + O(h^3)\end{aligned}$$

Método de Runge-Kutta de segundo orden

Notemos que

$$\mathbf{F}'(x,\mathbf{y}) = \frac{\partial \mathbf{F}}{\partial x} + \sum_{i=0}^{n-1} \frac{\partial \mathbf{F}}{\partial \mathbf{y}_i} \mathbf{y}_i' = \frac{\partial \mathbf{F}}{\partial x} + \sum_{i=0}^{n-1} \frac{\partial \mathbf{F}}{\partial \mathbf{y}_i} F_i(x,\mathbf{y})$$

donde n es el número de **1-EDO**.

Método de Runge-Kutta de segundo orden

Entonces podemos escribir para $\mathbf{y}(x+h)$ como

$$\mathbf{y}(x+h) = \mathbf{y}(x) + \mathbf{F}(x,\mathbf{y})\,h + \\ + \frac{1}{2}\left(\frac{\partial \mathbf{F}}{\partial x} + \sum_{i=0}^{n-1} \frac{\partial \mathbf{F}}{\partial y_i} F_i(x,\mathbf{y})\right) h^2 + O(h^3)$$

Método de Runge-Kutta de segundo orden

Regresando a la ecuación inicial, re-escribimos el último término mediante una serie de Taylor en varias variables:

$$\mathbf{F}[x + p\,h, \mathbf{y} + q\,h\mathbf{F}(x, \mathbf{y})] = \mathbf{F}(x, \mathbf{y}) + \frac{\partial \mathbf{F}}{\partial x}\,p\,h + \\ + q\,h \sum_{i=1}^{n-1} \frac{\partial \mathbf{F}}{\partial y_i} F_i(x, \mathbf{y}) + O(h^2)$$

Método de Runge-Kutta de segundo orden

Por lo que la ecuación inicial, toma la forma:

$$\mathbf{y}(x+h) = \mathbf{y}(x) + (c_0 + c_1)\,\mathbf{F}(x,\mathbf{y})\,h + \\ + c_1 \left[\frac{\partial \mathbf{F}}{\partial x}\,p\,h + q\,h \sum_{i=1}^{n-1} \frac{\partial \mathbf{F}}{\partial y_i} F_i(x,\mathbf{y}) \right] h + O(h^3)$$

Para que las expresiones sean idénticas, se necesita que:

$$c_0 + c_1 = 1 \qquad c_1\,p = \frac{1}{2} \qquad c_1\,q = \frac{1}{2}$$

Valores de las incógnitas

El conjunto anterior representa un sistema de tres ecuaciones y cuatro incógnitas, por lo que se asigna un valor a cualquiera de ellas.

Las opciones más comunes y sus nombres para los métodos son los siguientes:

Valores				Algoritmo
$c_0 = 0$	$c_1 = 1$	$p = \frac{1}{2}$	$q = \frac{1}{2}$	Euler modificado
$c_0 = \frac{1}{2}$	$c_1 = \frac{1}{2}$	$p = 1$	$q = 1$	Heun
$c_0 = \frac{1}{3}$	$c_1 = \frac{2}{3}$	$p = \frac{3}{4}$	$q = \frac{3}{4}$	Ralston

Todas estas fórmulas son del tipo **RK2**, ninguna tiene una superioridad numérica con respecto a las otras.

Método de Euler modificado

Sustituimos los valores de los parámetros en la ecuación general para obtener:

$$\mathbf{y}(x+h) = \mathbf{y}(x) + \mathbf{F}\left[x + \frac{h}{2}, \mathbf{y} + \frac{h}{2}\,\mathbf{F}(x, \mathbf{y})\right] h$$

Método de Euler modificado

Esta fórmula de integración puede evaluarse convenientemente, siguiendo la siguiente secuencia de operaciones:

$$\begin{aligned} \mathbf{K}_0 &= h\mathbf{F}(x, \mathbf{y}) \\ \mathbf{K}_1 &= h\mathbf{F}\left(x + \frac{h}{2}, \mathbf{y} + \frac{1}{2}\mathbf{K}_0\right) \\ \mathbf{y}(x+h) &= \mathbf{y}(x) + \mathbf{K}_1 \end{aligned}$$

Rep. del método de Euler modificado

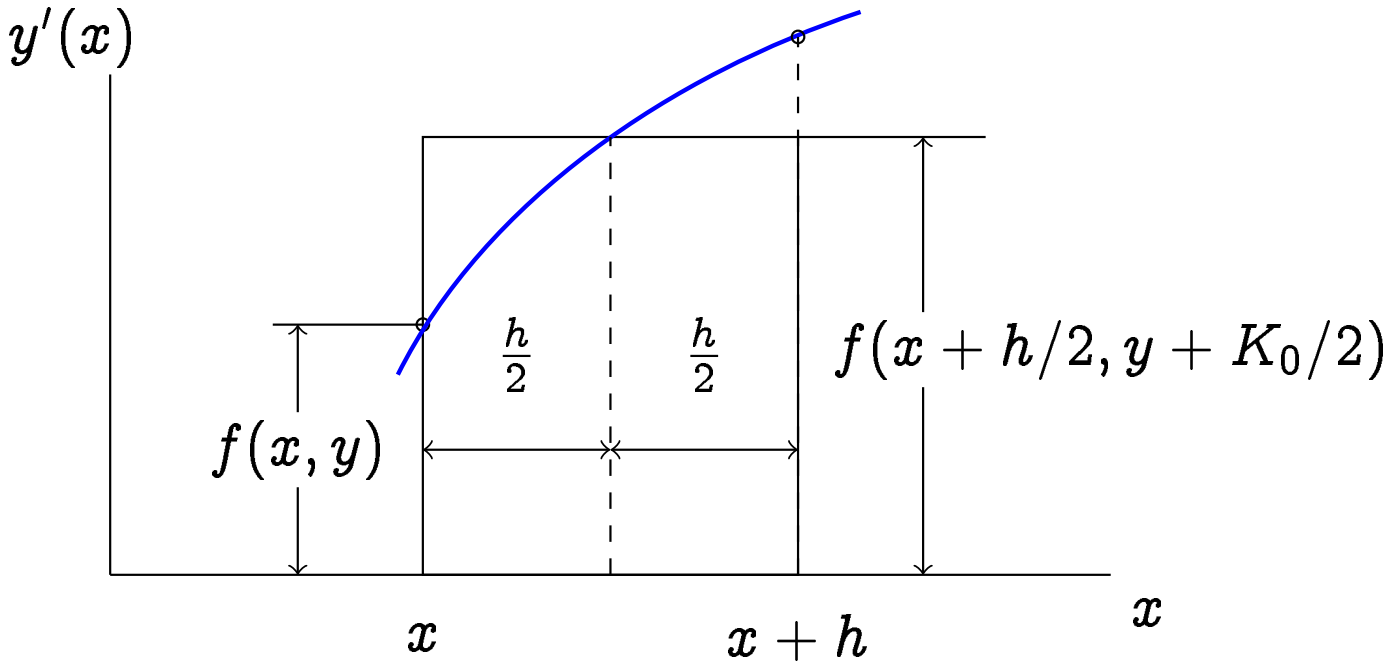


Figura 1: *Representación gráfica del método de Euler modificado para una EDO* $y' = f(x,y)$.

Método de Euler modificado

El valor de $\mathbf{K}_0 = h\mathbf{F}(x, \mathbf{y})$ devuelve un estimado de y en el punto central para la fórmula de Euler

$$y(x + \frac{h}{2}) = y(x) + f(x, y)\frac{h}{2} = y(x) + \frac{K_0}{2}$$

La segunda ecuación $\mathbf{K}_1$, aproxima el área del bloque por el área K_1 del rectángulo.

El error es proporcional a la curvatura y''' de la gráfica.

Ejemplo

Utiliza RK2 para integrar la siguiente EDO:

$$y' = \sin y \qquad\qquad y(0) = 1$$

de $x = 0$ a $x = 0.5$ en pasos de $h = 0.1$

Solución

Del problema tenemos que:

$$F(x, y) = \sin y$$

por lo que las fórmular canónicas de integración son

$$K_0 = h\, F(x, y) = 0.1 \sin y$$

$$K_1 = h\, F\left(x + \frac{h}{2}, y + \frac{1}{2}K_0\right) = 0.1 \sin\left(y + \frac{1}{2}K_0\right)$$

$$y(x + h) = y(x) + K_1$$

Solución

Como $y(0) = 1$, podemos integrar

$$K_0 = 0.1 \ \sin(1.0000) = 0.0841$$

$$K_1 = 0.1 \ \sin\left(1.0000 + \frac{0.0841}{2}\right) = 0.0863$$

$$y(0.1) = 1.0 + 0.0863 = 1.0863$$

Solución

En la siguiente evaluación

$$K_0 = 0.1\ \sin(1.0863) = 0.0885$$

$$K_1 = 0.1\ \sin\left(1.0.0863 + \frac{0.0885}{2}\right) = 0.0905$$

$$y(0.2) = 1.0863 + 0.0905 = 1.1768$$

y así, sucesivamente.

Solución

A manera de resumen, las cuentas se presentan en la siguiente tabla:

x	y	K_0	K_1
0.0	1.0000	0.0841	0.0863
0.1	1.0863	0.0885	0.0905
0.2	1.1768	0.0923	0.0940
0.3	1.2708	0.0955	0.0968
0.4	1.3676	0.0979	0.0988
0.5	1.4664		

Solución

La solución exacta (que podrían demostrar que cumple) es:

$$x(y) = ln(\csc y - \cot y) + 0.604582$$

que devuelve en $x(1.4664) = 0.5000$

Consideraciones

Sin embargo, es poco probable que esta precisión se mantenga mientras continuemos integrando, dado que los errores (tanto de truncamiento como de redondeo) se van acumulando, si se tiene un rango amplio de integración, se requiere entonces, de mejores fórmulas de integración.

Ejercicio

El circuito que se muestra, tiene una autoinductancia de $L = 50\,\text{H}$, una resistencia de $R = 20\,\Omega$, y una fuente de $V = 10\,\text{V}$.

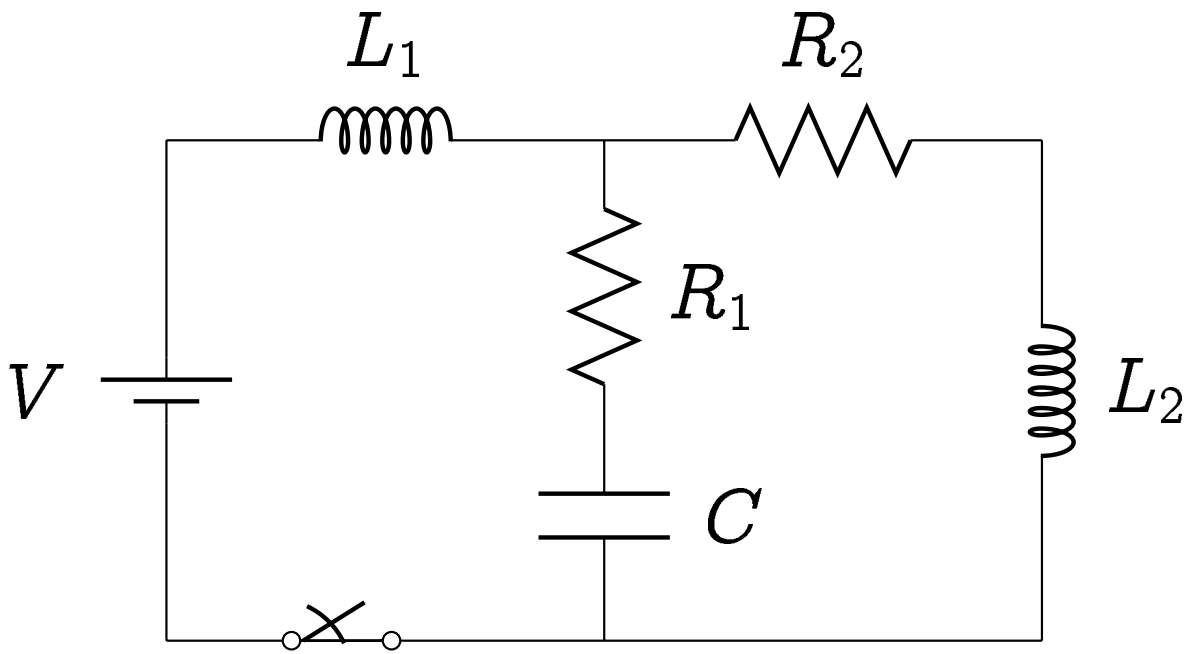


Figura 2: *Circuito RLC para el ejercicio.*

Condiciones iniciales del ejercicio

En el tiempo $t = 0$, la corriente $I(t)$ satisface

$$L\frac{d}{dt}I(t) + RI(t) = V, \qquad I(0) = 0$$

Usando el esquema de Runge-Kutta de segundo orden (RK2), calcula la corriente en el circuito para $0 \leq t \leq 10$ segundos, con $h = 0.1$

Ajustando la EDO

Se reescribe la ecuación como

$$\frac{d}{dt}I = -\frac{R}{L}I + \frac{V}{L} = F(I,t)$$

Aplicando el método RK2, tenemos

$$K_0 = h\left[-\frac{R}{L}\,I_n + \frac{V}{L}\right]$$

$$K_1 = h\left[-\frac{R}{L}\,(I_n + K_0) + \frac{V}{L}\right]$$

$$I_{n+1} = I_n + \frac{1}{2}\,(K_0 + K_1)$$

Código I

Código 6: Código para el circuito RLC

```
1  L = 50.0
2  R = 20.0
3  V = 10.0
4
5  h = 0.1
6
7  corriente = 0
8  I = []
9  I.append(0)
10
11 for i in range(99):
```

Código II

```
12      K0 = h * ((-R/L) * corriente + (V/L))
13      K1 = h *((-R/L) * (corriente + k0) + (V/L))
14      corriente = corriente + (K0 + K1) * 0.5
15      I.append(corriente)
```

Resultado gráfico de la solución

La rutina para la gráfica con **`matplotlib`** la pueden implementar sin mayor problema.

Resultado gráfico

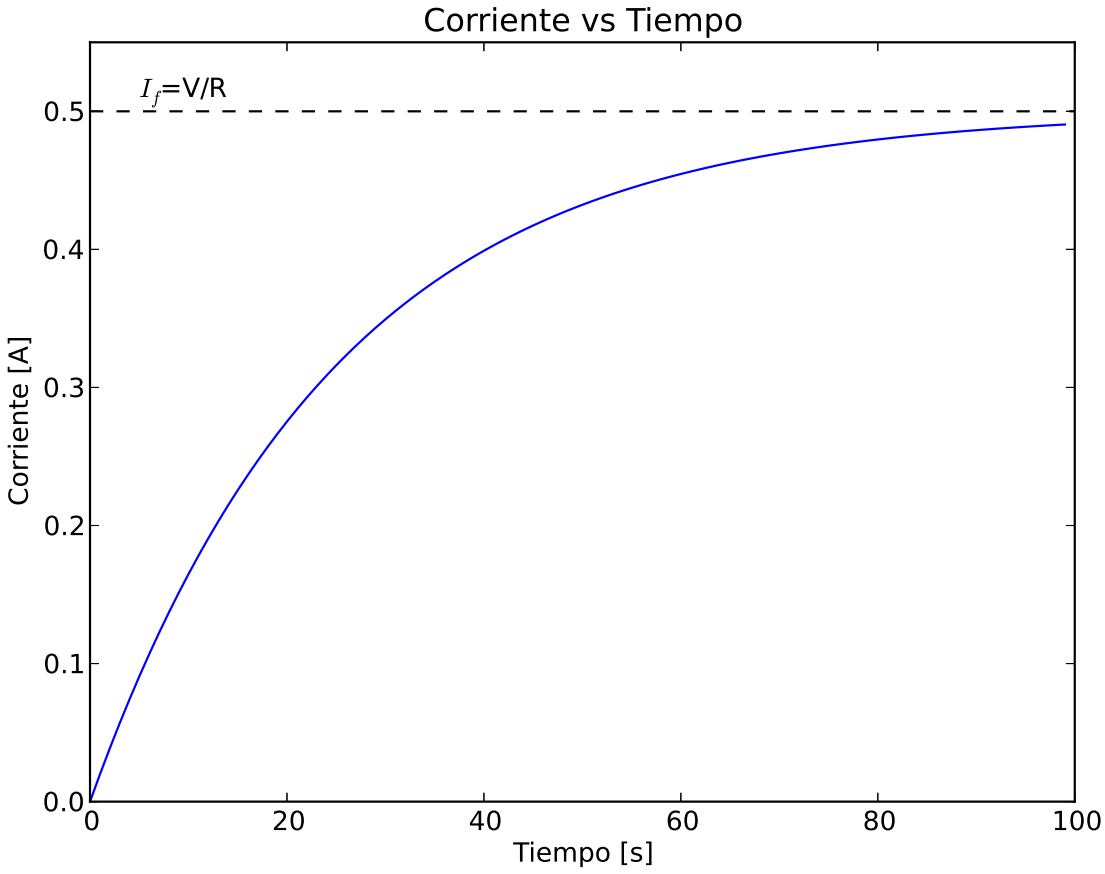


Figura 3: *Solución para el circuito, la corriente llega a un límite definido por el valor de V y de R.*

Conclusión del ejercicio

Nótese que el valor de corriente límite corresponde a $I_f = V/R$ que alcanzaría en un tiempo mucho mayor.

Método de Runge-Kutta de cuarto orden

El método de **RK4** se obtiene de la serie de Taylor, de la misma forma como se obtuvo RK2; considerando que la derivación es un proceso largo y no tan instructivo, entonces lo omitiremos.

Método de Runge-Kutta de cuarto orden

La expresión final de la fórmula de integración también depende de la elección de los parámetros, es decir, no hay una única fórmula para **RK4**.

Método RK4

La expresión más popular se le conoce como método **RK4**, que requiere de las siguientes operaciones:

$$K_0 = h\,\mathbf{F}(x, \mathrm{y})$$

$$K_1 = h\,\mathbf{F}(x + \frac{h}{2}, \mathrm{y} + \frac{\mathbf{K}_0}{2})$$

$$K_2 = h\,\mathbf{F}(x + \frac{h}{2}, \mathrm{y} + \frac{\mathbf{K}_1}{2})$$

$$K_3 = h\,\mathbf{F}(x + h, \mathrm{y} + \mathbf{K}_2)$$

$$\mathrm{y}(x + h) = \mathrm{y}(x) + \frac{1}{6}(\mathbf{K}_0 + 2\,\mathbf{K}_1 + 2\,\mathbf{K}_2 + \mathbf{K}_3)$$

Método RK4

El principal inconveniente de este método es que no se presta para una estimación del error de truncamiento. Por lo tanto, tenemos que "adivinar" el tamaño del paso de integración h, o determinarlo por ensayo y error.

Alternativa: métodos adaptativos

En contraste, los llamados métodos adaptativos pueden evaluar el error de truncamiento en cada paso de integración y ajustar el valor de h en consecuencia, pero con un gran costo, computacionalmente hablando.

Algoritmo RK4

La función **`integra`** en este módulo, implementa el método **RK4**.

El usuario deberá de proporcionar en **`integra`** la función `F(x,y)` que define el conjunto de 1-EDO $\mathbf{y}' = \mathbf{F}(x, \mathbf{y})$.

Código 7: Código para la función integra

```
1  def integra(F, x, y, xAlto, h):
2
3      def rk4(F, x, y, h):
4          K0 = h * F(x,y)
5          K1 = h * F(x + h/2.0, y + K
       0/2.0)
6          K2 = h * F(x + h/2.0, y + K
       1/2.0)
7          K3 = h * F(x + h, y + K2)
8
9          return (K0 + 2.0 * K1 + 2.0
       * K2 + K3) / 6.0
10
11     X = []
```

```
12      Y = []
13      X.append(x)
14      Y.append(y)
15
16      while x < xAlto:
17          h = min(h, xAlto - x)
18          y = y + rk4(F, x, y, h)
19          x = x + h
20          X.append(x)
21          Y.append(y)
22
23      return array(X), array(Y)
```

Ejemplo

Resolver

$$y'' = -0.1\,y' - x \qquad y(0) = 0 \qquad y'(0) = 1$$

de $x = 0$ a $x = 2$ con incrementos de $h = 0.25$ mediante **RK4**.

Solución

Usando la notación $y_0 = y$ junto con $y_1 = y'$, podemos escribir un conjunto de **1-EDO** como

$$\mathbf{y}' = \mathbf{F}(x, \mathbf{y}) = \begin{bmatrix} y_0' \\ y_1' \end{bmatrix} = \begin{bmatrix} y_1 \end{bmatrix}$$

Código completo I

Código 8: Código para el ejercicio

```
1  def F(x,y):
2      F = zeros((2), dtype='float64')
3      F[0] = y[1]
4      F[1] = -0.1 * y[1] - x
5      return F
6
7  x = 0.0
8  xAlto = 2.0
9  y = array([0.0,1.0])
10 h = 0.25
11 freq = 1
12
```

Código completo II

```
13 X,Y = integra(F, x, y, xAlto, h)
14 imprimeSoln(X, Y, freq)
```

Resultado

x	$y[0]$	$y[1]$
$0.0000e+00$	$0.0000e+00$	$1.0000e+00$
$2.5000e-01$	$2.4431e-01$	$9.4432e-01$
$5.0000e-01$	$4.6713e-01$	$8.2829e-01$
$7.5000e-01$	$6.5355e-01$	$6.5339e-01$
$1.0000e+00$	$7.8904e-01$	$4.2110e-01$
$1.0000e+00$	$7.8904e-01$	$4.2110e-01$
$1.0000e+00$	$7.8904e-01$	$4.2110e-01$
$1.2500e+00$	$8.5943e-01$	$1.3281e-01$
$1.5000e+00$	$8.5090e-01$	$2.1009e-01$
$1.7500e+00$	$7.4995e-01$	$6.0625e-01$
$2.0000e+00$	$5.4345e-01$	$-1.0543e+00$

Ejercicio 2

Usa RK4 para integrar

$$y' = 3\,y - 4\,e^{-x} \qquad\qquad y(0) = 1$$

desde $x = 0$ hasta $x = 10$ en pasos de $h = 0.1$.

Compara el resultado con la solución analítica $y = \exp(-x)$

Solución al ejercicio

Usaremos el programa anterior. Recordemos que la función `rk_4` supone que y es un arreglo, por lo que debemos de especificar la condición inicial como $y = \text{array}([1.0])$ y no $y = 1.0$

Código 9: Código para el ejercicio

```
1  def F(x,y):
2      F = zeros((1), dtype='float64')
3      F[0] = 3.0 * y[0] - 4.0 * exp(-x
     )
4      return F
5
6
7  x = 0.0
8  xAlto = 10.0
9  y = array([1.0])
10 h = 0.1
11 freq = 20
12
13 X,Y = integra(F, x, y, xAlto, h)
```

```
14 imprimeSoln(X, Y, freq)
```

Resultado

x	$y[0]$
$0.0000e+00$	$1.0000e+00$
$2.0000e+00$	$1.3250e-01$
$4.0000e+00$	$-1.1237e+00$
$6.0000e+00$	$-4.6056e+02$
$8.0000e+00$	$-1.8575e+05$
$1.0000e+01$	$-7.4912e+07$

Resultado

x	$y[0]$
$0.0000e+00$	$1.0000e+00$
$2.0000e+00$	$1.3250e-01$
$4.0000e+00$	$-1.1237e+00$
$6.0000e+00$	$-4.6056e+02$
$8.0000e+00$	$-1.8575e+05$
$1.0000e+01$	$-7.4912e+07$

Pero ¿es correcto esto?

¿Qué ocurrió?

De acuerdo a la solución numérica, y debería de acercarse a cero conforme se incrementa x, pero el resultado nos muestra lo contrario: después de un incremento inicial, la magnitud de y se incrementa súbitamente.

Podemos explicar esto mirando de cerca la solución analítica.

Solución general

La solución general de la EDO dada es

$$y = C\, e^{3\,x} + e^{-x}$$

que puede verificarse por sustitución.

La condición inicial $y(0) = 1$ hace que $C = 0$, que para la solución del problema, es precisamente $y = exp(-x)$.

Solución general

El problema en la solución numérica es el término dominante $C\ e^{3\,x}$

Supongamos que la condición inicial tiene un pequeño error ε, por lo que tenemos $y(0) = 1 + \varepsilon$.

Esto cambia la solución analítica por

$$y = \varepsilon e^{3\,x} + e^{-x}$$

Solución general

$$y = \varepsilon e^{3\,x} + e^{-x}$$

Vemos que el término que contiene el error ε, se hace dominante conforme x aumenta.

Sensibilidad de las condiciones iniciales

Dado que los errores son inherentes a las soluciones numéricas, tenemos el mismo efecto para cambios pequeños en las condiciones iniciales.

Concluimos que nuestra solución numérica es víctima de la *inestabilidad numérica*, debida a la sensibilidad de la solución a las condiciones iniciales.