

DIFERENCIAS FINITAS

PROBLEMAS DE VALOR INICIAL

Modelación computacional en las ciencias y las ingenierías como
apoyo en el proceso enseñanza-aprendizaje
(PAPIME-PE101019)

Instituto de Geofísica
Universidad Nacional Autónoma de México



Esta obra está bajo una [Licencia Creative Commons Atribución-NoComercial-CompartirIgual 4.0 Internacional](https://creativecommons.org/licenses/by-nc-sa/4.0/).



- 1 PROBLEMAS DE VALOR INICIAL
- 2 MÉTODO DE EULER
 - Ejercicio 6.
 - Ejercicio 7.
 - Ejercicio 8.
- 3 MÉTODOS DE RUNGE-KUTTA
 - Ejercicio 9.
- 4 REFERENCIAS
- 5 CRÉDITOS

CONTENIDO

- ① PROBLEMAS DE VALOR INICIAL
- ② MÉTODO DE EULER
 - Ejercicio 6.
 - Ejercicio 7.
 - Ejercicio 8.
- ③ MÉTODOS DE RUNGE-KUTTA
 - Ejercicio 9.
- ④ REFERENCIAS
- ⑤ CRÉDITOS

PROBLEMAS DE VALOR INICIAL

Las ecuaciones diferenciales que modelan problemas de la ciencia y la ingeniería involucran el cambio de alguna variable con respecto a otra. La mayoría de estos problemas requieren de la solución de un problema de valor inicial, es decir, la solución de una ecuación diferencial que satisface una condición inicial dada:

IVP (INITIAL VALUE PROBLEM)

Aproximar la solución $y(t)$ al problema:

$$\frac{dy(t)}{dt} = f(t, y), \text{ para } a \leq t \leq b$$

sujeto a la condición inicial $y(a) = \alpha$

En muchos casos las soluciones analíticas a los problemas de valor inicial no pueden encontrarse, por lo que se usan métodos numéricos para aproximar sus soluciones.

CONTENIDO

- ① PROBLEMAS DE VALOR INICIAL
- ② MÉTODO DE EULER
 - Ejercicio 6.
 - Ejercicio 7.
 - Ejercicio 8.
- ③ MÉTODOS DE RUNGE-KUTTA
 - Ejercicio 9.
- ④ REFERENCIAS
- ⑤ CRÉDITOS

MÉTODO DE EULER

El método de Euler es el más sencillo para obtener soluciones aproximadas al problema de valor inicial bien planteado que se escribe como:

$$\frac{dy}{dt} = f(t, y), \quad a \leq t \leq b, \quad y(a) = \alpha \quad (1)$$

Podemos aproximar la solución en N_t pasos de tiempo igualmente espaciados en $[a, b]$, estos puntos se definen como sigue: $t_n = a + n * h_t$ para $n = 0, 1, 2, \dots, N_t$, donde $h_t = (b - a)/N_t$ es el tamaño del paso (*stepsize*).

Supongamos que el problema (1) tiene una solución única $y(t)$ y que además tiene dos derivadas continuas en $[a, b]$, de tal manera que para cada $n = 0, 1, 2, \dots, N_t - 1$ tenemos la siguiente expansión en series de Taylor:

$$y(t_{n+1}) = y(t_n) + (t_{n+1} - t_n) \frac{dy}{dt}(t_n) + \frac{(t_{n+1} - t_n)^2}{2} \frac{d^2y}{dt^2}(\xi_n)$$

para algún número $\xi_n \in (t_n - t_{n+1})$. Dado que $y(t)$ satisface la ecuación diferencial (1) y como $h_t = (t_{n+1} - t_n)$ entonces podemos escribir:

$$y(t_{n+1}) = y(t_n) + h_t f(t_n, y(t_n)) + \frac{h_t^2}{2} y''(\xi_n) \quad (2)$$

FORWARD EULER

El método de Euler hacia adelante (*forward*) consiste en eliminar el último término de la ecuación (2) de tal manera que, definiendo $y_n = y(t_n)$ tenemos lo siguiente

$$\begin{aligned}y_0 &= y(a) = \alpha \\y_{n+1} &= y_n + h_t f(t_n, y_n), \text{ para } n = 0, 1, 2, \dots, N_t - 1\end{aligned}\quad (3)$$

La ecuación (3) proporciona una aproximación a la solución del problema (1) en el paso t_{n+1} y se conoce como la ecuación en diferencias. Este método:

- Es explícito.
- Es barato.
- Es fácil de implementar.
- Es condicionalmente estable

BACKWARD EULER

Es posible derivar un método de Euler hacia atrás (Backward Euler) el cual se escribe como:

$$\begin{aligned}y_0 &= y(a) = \alpha \\y_{n+1} &= y_n + h_t f(t_{n+1}, y_{n+1}), \text{ para } n = 1, 2, \dots, N_t - 1\end{aligned}\quad (4)$$

Obsérvese que:

- Esta fórmula es implícita.
- El costo por paso puede ser mayor que en el caso de Forward Euler.
- En general es más difícil de implementar.
- Este método es más estable que el Forward Euler.

EJERCICIO 6: DECAIMIENTO RADIOCATIVO

La ley de decaimiento radioactivo dice que la masa de una sustancia radioactiva decae a una razón que es proporcional a la cantidad de masa que está presente. Si $y(t)$ expresa la cantidad de sustancia en el tiempo t , entonces la ley de decaimiento se expresa como:

$$\begin{aligned}\frac{dy(t)}{dt} &= -\lambda y(t), & \text{para } 0 < t < T_{max} \\ y(0) &= y_0 & \text{(condición inicial)}\end{aligned}$$

donde y_0 representa la cantidad de sustancia inicial, $T_{max} = h_t * N_t$ y $\lambda > 0$. La solución exacta es: $y(t) = y_0 e^{-\lambda t}$.

- ① **Aproximar el decaimiento radioactivo usando los métodos *forward Euler* y *backward Euler*:**
 - Ⓐ Escribir y analizar las fórmulas de los métodos.
 - Ⓑ Implementar los métodos en Python.
 - Ⓒ Mostrar el comportamiento de ambos métodos para $\lambda = 1.5$, $y_0 = 20$, $T_{max} = 10$ y $N_t = 7, 8, 9, 10, 20$.
 - Ⓓ ¿Cuántos pasos de tiempo (N_t) se necesitan para que el error sea menor que 1.0 en cada método?

Entregue su código documentado en una notebook de nombre `E06.decaimiento.ipynb`.

EJERCICIO 6: SOLUCIÓN 1.A

Forward Euler:

$$y_{n+1} = y_n - h_t \lambda y_n$$

$$\Rightarrow y_{n+1} = (1 - h_t \lambda) y_n$$

$$\text{Paso 1: } y_1 = A y_0$$

$$\text{Paso 1: } y_2 = A^2 y_0$$

$$\vdots \quad \vdots \quad \vdots$$

$$\text{Paso } N_t : y_{n+1} = A^{N_t} y_0$$

Donde: $A = (1 - h_t \lambda)$. Obsérvese que si existe un error en y_i , ese error se multiplica por A al calcular y_{i+1} . Si $|A| > 1$ dicho error podría llevar a la no convergencia del método.

Backward Euler:

$$y_{n+1} = y_n - h_t \lambda y_{n+1}$$

$$\Rightarrow y_{n+1} = (1 + h_t \lambda)^{-1} y_n$$

$$\text{Paso 1: } y_1 = B y_0$$

$$\text{Paso 1: } y_2 = B^2 y_0$$

$$\vdots \quad \vdots \quad \vdots$$

$$\text{Paso } N_t : y_{n+1} = B^{N_t} y_0$$

Donde: $B = (1 + h_t \lambda)^{-1}$. Obsérvese que si existe un error en y_i , ese error se multiplica por B al calcular y_{i+1} , pero $B < 1$ en todos los casos, por lo que el método es estable.

EJERCICIO 6: SOLUCIÓN 1.B

Nota: implemente el siguiente código y documente usando *docstring*. Agregue el código correspondiente para realizar las gráficas que se muestran en las siguientes páginas.

```
def mesh(a, b, Nt):
    ht = (b-a) / Nt
    return ht

def exactSolution(t, y0, lam):
    return y0 * np.exp(-lam * t)

def forwardEuler(y, ht, lam):
    A = 1 - ht*lam
    An = [A]
    for i, val in enumerate(y[0:-1]):
        y[i+1] = A * y[i]
        An.append(An[i] * A)
    return An

def backwardEuler(y, ht, lam):
    B = 1 / (1 + ht*lam)
    Bn = [B]
    for i, val in enumerate(y[0:-1]):
        y[i+1] = B * y[i]
        Bn.append(Bn[i] * B)
    return Bn
```

```
Nt = 6
Tmax = 10
ht = mesh(0, Tmax, Nt)
y0 = 20
lam = 1.5

t = np.linspace(0, Tmax, Nt+1)
yf = np.zeros(Nt+1)
yb = np.zeros(Nt+1)

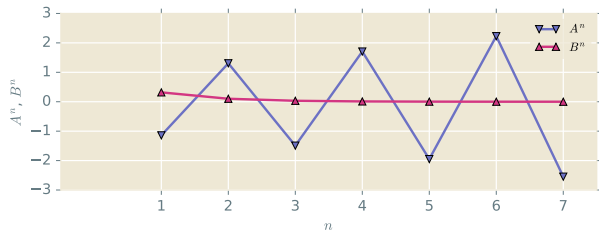
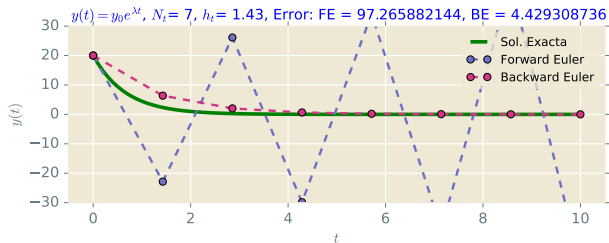
yf[0] = y0
yb[0] = y0
error_f = forwardEuler(yf, ht, lam)
error_b = backwardEuler(yb, ht, lam)

t1 = np.linspace(0, Tmax, 100)
y_exacta = exactSolution(t1, y0, lam)
y_exac_p = exactSolution(t, y0, lam)

error_f = np.linalg.norm(yf - y_exac_p, 2)
error_b = np.linalg.norm(yb - y_exac_p, 2)
```

EJERCICIO 6: SOLUCIÓN 1.C

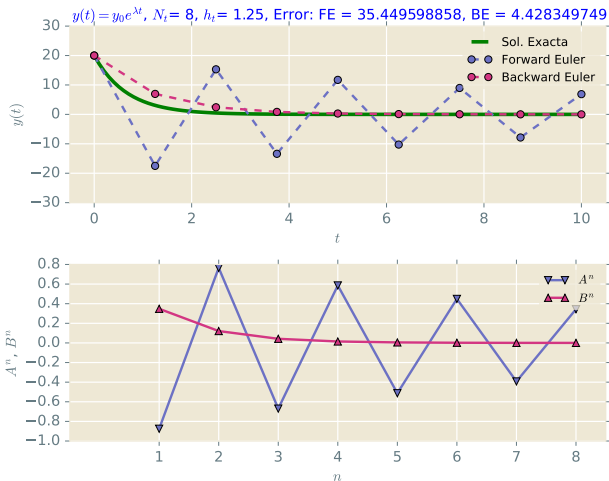
Decaimiento Radioactivo



El método forward Euler (FE) es inestable y diverge. El método backward Euler (BE) es muy estable y se obtiene una buena solución aunque imprecisa con un error de ~ 4.43 . Las gráficas de los coeficientes de cada método, A^n y B^n , reflejan el comportamiento de cada solución.

EJERCICIO 6: SOLUCIÓN 1.C

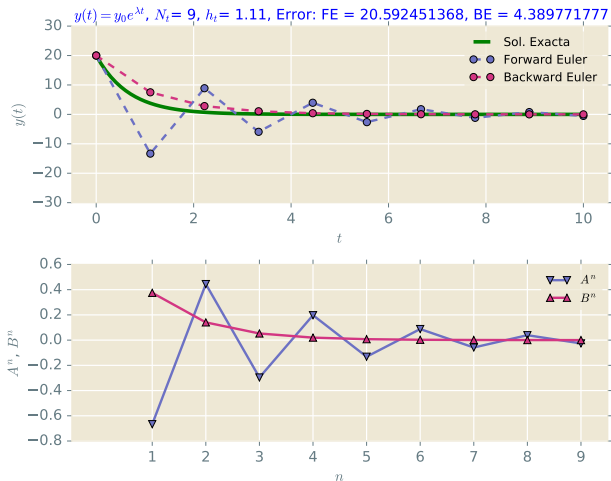
Decaimiento Radioactivo



La h_t se reduce, por lo que el método FE ya no diverge, pero es inestable de tal manera que oscila. El método BE sigue estable, observe que el error es similar al del caso anterior. Las gráficas de los coeficientes de cada método, A^n y B^n , se mantienen entre -1 y 1 , es decir $|A^n| < 1$ y $|B^n| < 1$.

EJERCICIO 6: SOLUCIÓN 1.C

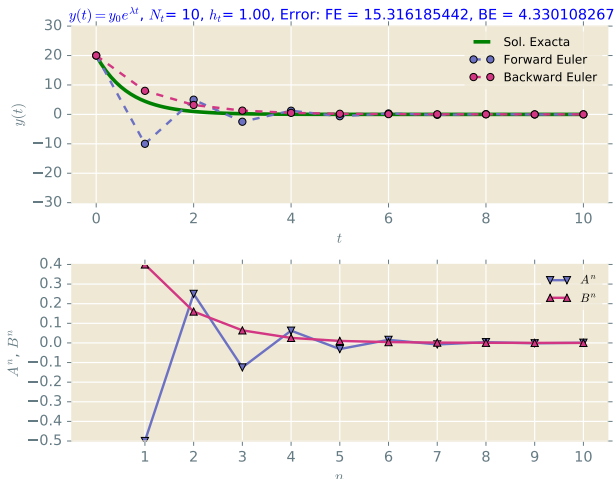
Decaimiento Radioactivo



La h_t es aún menor, pero el método FE sigue oscilando, aunque menos que en el caso anterior. El método BE sigue estable, observe que el error es del mismo orden que el caso anterior. Las gráficas de los coeficientes de cada método, A^n y B^n , muestran una reducción y se sigue cumpliendo $|A^n| < 1$ y $|B^n| < 1$.

EJERCICIO 6: SOLUCIÓN 1.C

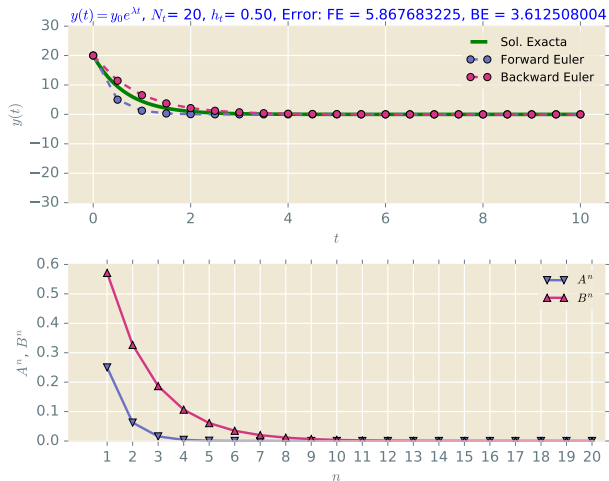
Decaimiento Radioactivo



La h_t es aún menor, el método FE oscila menos que en el caso anterior y el error se ha reducido. El método BE sigue estable, observe que el error es del mismo orden que el caso anterior. Las gráficas de los coeficientes de cada método, A^n y B^n , muestran una reducción y observe ahora que $|A^n| < 0.5$ y $|B^n| < 0.5$.

EJERCICIO 6: SOLUCIÓN 1.C

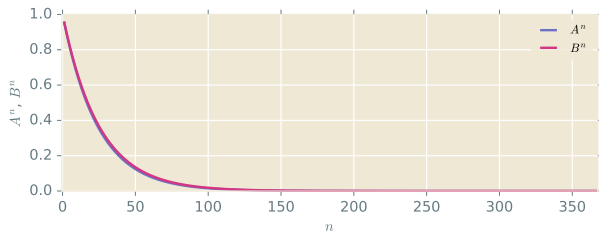
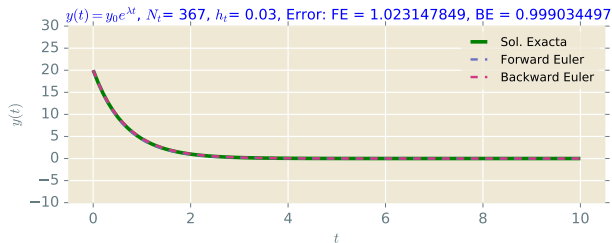
Decaimiento Radioactivo



Finalmente el método FE ya no oscila y su comportamiento es similar al método BE, a partir de ahora parece que los errores comienzan a reducirse en ambos casos conforme se aumenta N_t . De igual manera las gráficas de los coeficientes de cada método, A^n y B^n tienen un comportamiento similar y se mantienen en $|A^n| < 1$ y $|B^n| < 1$.

EJERCICIO 6: SOLUCIÓN 1.D

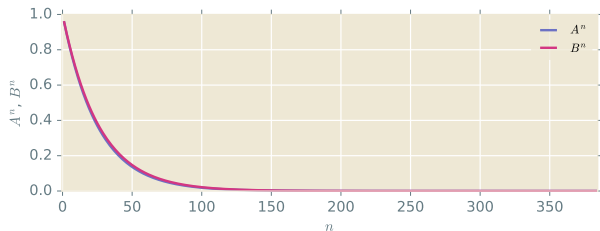
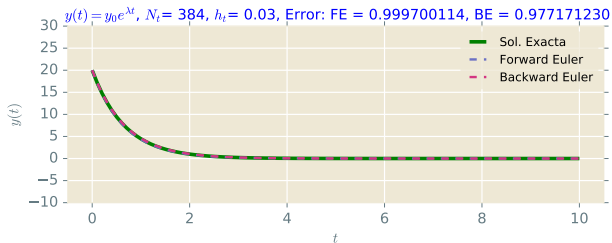
Decaimiento Radioactivo



Se requiere de $N_t = 367$ para que el método BE tenga un error menor que 1. Sin embargo ambos métodos presentan un compartamiento similar con un error muy parecido. Las gráficas de los coeficientes de cada método, A^n y B^n , se mantienen en $|A^n| < 1$ y $|B^n| < 1$.

EJERCICIO 6: SOLUCIÓN 1.D

Decaimiento Radioactivo



Se requiere de $N_t = 384$ para que el método FE tenga un error menor que 1. Sin embargo ambos métodos presentan un compartamiento similar con un error muy parecido. Las gráficas de los coeficientes de cada método, A^n y B^n , se mantienen en $|A^n| < 1$ y $|B^n| < 1$.

EJERCICIO 6: CÓDIGO PARA LOS GRÁFICOS

```

Ecuacion = 'y(t) = y_0 e^{lambda t},' + 'N_t$' + '='.format(Nt) + \
           'h_t$' + '='.format(ht)
Error = ', Error: FE = { :10.9f}, BE = { :10.9f}'.format(norma_error_f, norma_error_b)

plt.style.use(['Solarize_Light2'])
fig, (ax1, ax2) = plt.subplots(2,1)
fig.suptitle('Decaimiento Radioactivo', fontsize=14)
ax1.plot(tl, y_exacta, 'g-', lw=3, label='Sol. Exacta')
ax1.plot(t, yf, 'C7o--', label='Forward Euler')
ax1.plot(t, yb, 'C6o--', label='Backward Euler')
ax1.set_title(Ecuacion + Error, fontsize=12, color='blue')
ax1.set_xlim(-0.5, t[-1]+0.5)
ax1.set_ylim(-30,30)
ax1.set_xlabel('$t$')
ax1.set_ylabel('$y(t)$')
ax1.legend(loc='upper right', ncol=1, framealpha=0.75, fancybox=True, fontsize=10)
ax1.grid(color='w')
nticks = np.arange(1, Nt+1, 1)
ax2.plot(nticks, An[:-1], 'C7v-', label='$A^n$')
ax2.plot(nticks, Bn[:-1], 'C6^-', label='$B^n$')
ax2.set_xlim(-0.5, Nt+0.5)
ax2.set_xticks(nticks)
ax2.set_xlabel('$n$')
ax2.set_ylabel('$A^n$, $B^n$')
ax2.legend(loc='upper right', ncol=1, framealpha=0.75, fancybox=True, fontsize=10)
ax2.grid(color='w')
plt.subplots_adjust(hspace=0.35)
plt.savefig('decaimiento_Nt_{}.pdf'.format(Nt))
plt.show()

```

EJERCICIO 7: DECAIMIENTO RADIOACTIVO CON $\lambda = 2.0$

Resuelva el mismo problema del ejercicio 6, pero ahora use $\lambda = 2.0$ y conteste lo siguiente:

- 1 ¿Para qué valor de N_t el método FE converge?
- 2 ¿Para qué valor de N_t el método FE deja de oscilar?
- 3 ¿Para qué valor de N_t el método BE tiene un error menor a 3.0?
- 4 ¿Para qué valor de N_t el método FE tiene un error menor a 3.0?

Entregue su código documentado en una notebook de nombre `E07_decaimiento.ipynb`.

EJERCICIO 8: ECUACIÓN LOGÍSTICA

En el estudio de crecimiento de poblaciones, limitadas por competencia por alimentos, se tiene un modelo matemático conocido como la ecuación logística:

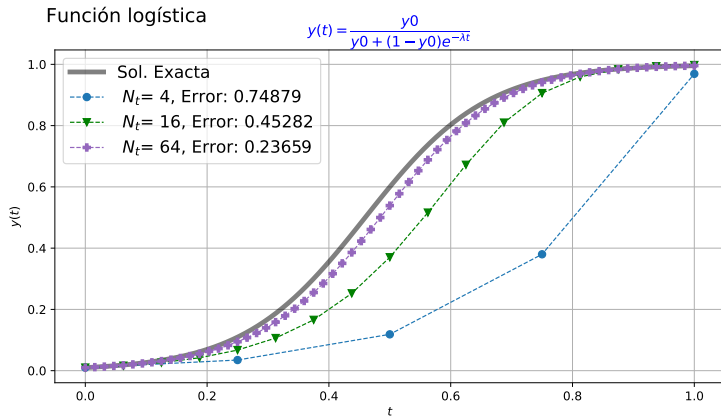
$$\begin{aligned}\frac{dy(t)}{dt} &= \lambda y(1 - y), & \text{para } 0 < t < T_{max} \\ y(0) &= y_0 & \text{(condición inicial)}\end{aligned}$$

donde y_0 es la población inicial, $\lambda > 0$ y $T_{max} = h_t * N_t$. Obsérvese que en este caso la ecuación es no lineal, pues del lado derecho aparece un término con y^2 . La solución exacta es: $y(t) = \frac{y_0}{y_0 + (1 - y_0)e^{-\lambda t}}$.

- 1 Aproximar el crecimiento de la población usando el método *forward Euler*.
 - 1 Escribir la fórmula del método para este caso e implementarla en Python.
 - 2 Reproducir la gráfica que se muestra en la siguiente lámina para $\lambda = 10$, $y_0 = 0.01$ y $N_t = 4, 16$ y 64 .

Entregue su código documentado en una notebook de nombre `E08_logistica.ipynb`.

EJERCICIO 8: ECUACIÓN LOGÍSTICA



CONTENIDO

- ① PROBLEMAS DE VALOR INICIAL
- ② MÉTODO DE EULER
 - Ejercicio 6.
 - Ejercicio 7.
 - Ejercicio 8.
- ③ MÉTODOS DE RUNGE-KUTTA
 - Ejercicio 9.
- ④ REFERENCIAS
- ⑤ CRÉDITOS

MÉTODOS DE RUNGE-KUTTA

El método de Euler es de orden $\mathcal{O}(h_t)$. Es posible obtener métodos de órdenes mayores, varios de ellos son conocidos como métodos de Runge-Kutta.

MÉTODO DE RUNGE-KUTTA DE ORDEN DOS (PUNTO MEDIO)

$$\begin{aligned}
 w_0 &= \alpha && \text{(condición inicial)} \\
 k_1 &= h_t f(t_n, w_n) \\
 w_{n+1} &= w_n + h_t f(t_n + h_t/2, w_n + k_1/2) \\
 &\text{para } n = 0, 1, \dots, N_t - 1
 \end{aligned}$$

Este método es de orden $\mathcal{O}(h_t^2)$.

MÉTODOS DE RUNGE-KUTTA

MÉTODO DE RUNGE-KUTTA DE ORDEN TRES (HEUN)

$$w_0 = \alpha \quad (\text{condición inicial})$$

$$k_1 = h_t f(t_n, w_n)$$

$$k_2 = h_t f(t_n + h_t/3, w_n + k_1/3)$$

$$k_3 = h_t f(t_n + 2h_t/3, w_n + 2k_2/3)$$

$$w_{n+1} = w_n + (k_1 + 3k_3)/4$$

$$\text{para } n = 0, 1, \dots, N_t - 1$$

Este método es de orden $\mathcal{O}(h_t^3)$.

MÉTODOS DE RUNGE-KUTTA

MÉTODO DE RUNGE-KUTTA DE CUARTO ORDEN

$$w_0 = \alpha \quad (\text{condición inicial})$$

$$k_1 = h_t f(t_n, w_n)$$

$$k_2 = h_t f\left(t_n + \frac{h_t}{2}, w_n + \frac{1}{2}k_1\right)$$

$$k_3 = h_t f\left(t_n + \frac{h_t}{2}, w_n + \frac{1}{2}k_2\right)$$

$$k_4 = h_t f(t_{n+1}, w_n + k_3)$$

$$w_{n+1} = w_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \text{ para } n = 0, 1, \dots, N_t - 1$$

Este método es de orden $\mathcal{O}(h_t^4)$.

EJERCICIO 9: MÉTODOS DE RUNGE-KUTTA

Considere el siguiente problema:

$$\begin{aligned}\frac{dy}{dt} &= y - t^2 + 1 \quad \text{en } 0 \leq t \leq 4 \\ y(0) &= 0.5\end{aligned}$$

Cuya solución exacta es: $y(t) = (t + 1)^2 - 0.5e^t$.

Para $h_t = 1.0, 0.5, 0.25, 0.125$ ($N_t = 4, 8, 16, 32$), realice lo siguiente:

- 1 Aproximar la solución del problema con:
 - Método de Euler.
 - Método de Runge-Kutta de orden 2.
 - Método de Runge-Kutta de orden 3.
 - Método de Runge-Kutta de orden 4.
- 2 En una figura, grafique la solución exacta y compárela con las soluciones numéricas obtenidas con cada método.
- 3 En otra figura, grafique el error y explique el comportamiento de cada método.

Entregue su código documentado en una notebook de nombre `E09_Runge-Kutta.ipynb`.

EJERCICIO 9: SOLUCIÓN (1)

Nota: implemente el siguiente código y documente usando *docstring*.

```
def mesh(a, b, Nt):
    ht = (b-a) / Nt
    return ht

def f(t,y):
    return y - t**2 + 1

def Exacta(t):
    return (t+1)**2 - 0.5 * np.exp(t)

def Euler(f, t, w, ht):
    for i, val in enumerate(w[0:-1]):
        w[i+1] = w[i] + ht * f(t[i], w[i])
        t[i+1] = t[0] + (i+1) * ht

def RK2(f, t, w, ht):
    for i, val in enumerate(w[0:-1]):
        k1 = ht * f(t[i], w[i])
        w[i+1] = w[i] +
            ht * f(t[i] + ht * 0.5,
                w[i] + k1 * 0.5)
        t[i+1] = a + (i+1) * ht
```

```
def RK3(f, t, w, ht):
    for i, val in enumerate(w[0:-1]):
        k1 = ht * f(t[i], w[i])
        k2 = ht * f(t[i] + ht/3,
                    w[i] + k1 / 3)
        k3 = ht * f(t[i] + 2 * ht / 3,
                    w[i] + 2 * k2 / 3)
        w[i+1] = w[i] + (k1 + 3 * k3) / 4
        t[i+1] = a + (i+1) * ht

def RK4(f, t, w, ht):
    for i, val in enumerate(w[0:-1]):
        k1 = ht * f(t[i], w[i])
        k2 = ht * f(t[i] + ht/2,
                    w[i] + k1 / 2)
        k3 = ht * f(t[i] + ht/2,
                    w[i] + k2 / 2)
        k4 = ht * f(t[i] + ht, w[i] + k3)
        w[i+1] = w[i] + (k1 + 2*k2 +
                        2*k3 + k4) / 6
        t[i+1] = a + (i+1) * ht
```

EJERCICIO 9: SOLUCIÓN (1)

Nota: implemente el siguiente código y documente usando *docstring*. Agregue el código correspondiente para reproducir las gráficas que se muestran en las siguientes páginas (en azul se muestran los errores obtenidos con cada método).

```
Nt = 8 # 4, 8, 16, 32
a = 0
b = 4
ht = mesh(a, b, Nt)
y0 = 0.5
```

```
t = np.linspace(a, b, Nt+1)
y_eul = np.zeros(Nt+1);
y_rk2 = np.zeros(Nt+1)
y_rk3 = np.zeros(Nt+1)
y_rk4 = np.zeros(Nt+1)
```

```
y_eul[0]=y0
y_rk2[0]=y0
y_rk3[0]=y0
y_rk4[0]=y0
```

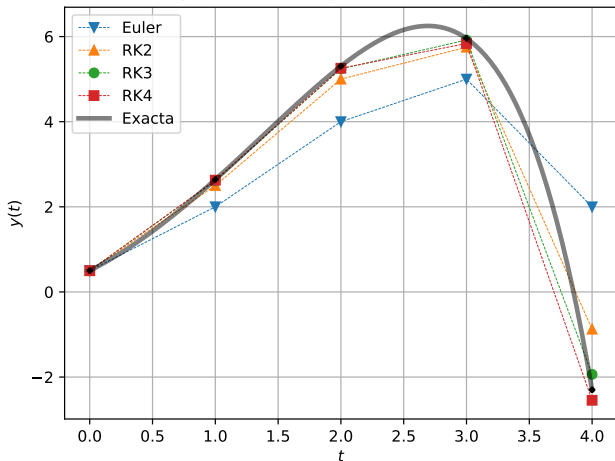
```
Euler(f, t, y_eul, ht)
RK2(f, t, y_rk2, ht)
RK3(f, t, y_rk3, ht)
RK4(f, t, y_rk4, ht)
```

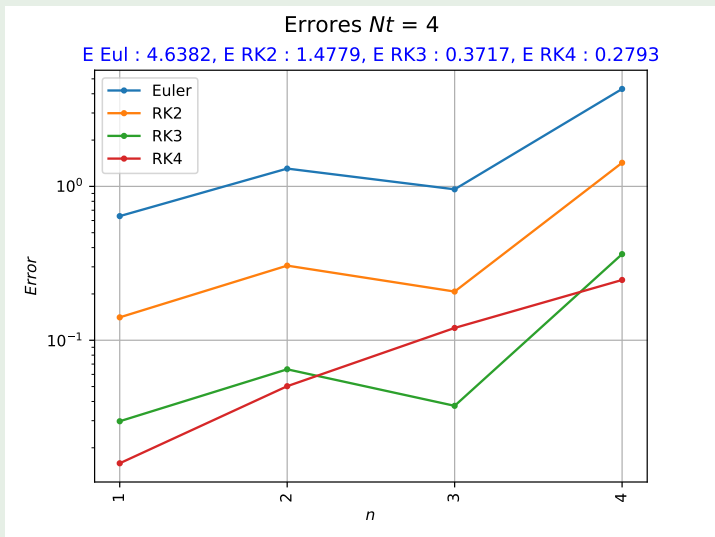
```
yp = Exacta(t)
e_eul = np.abs(yp - y_eul)
e_rk2 = np.abs(yp - y_rk2)
e_rk3 = np.abs(yp - y_rk3)
e_rk4 = np.abs(yp - y_rk4)
```

```
n_error_eul = np.linalg.norm(e_eul, 2)
n_error_rk2 = np.linalg.norm(e_rk2, 2)
n_error_rk3 = np.linalg.norm(e_rk3, 2)
n_error_rk4 = np.linalg.norm(e_rk4, 2)
```

EJERCICIO 9: COMPARACIÓN DE MÉTODOS CON $N_t = 4$ Solución y aproximación $Nt = 4$

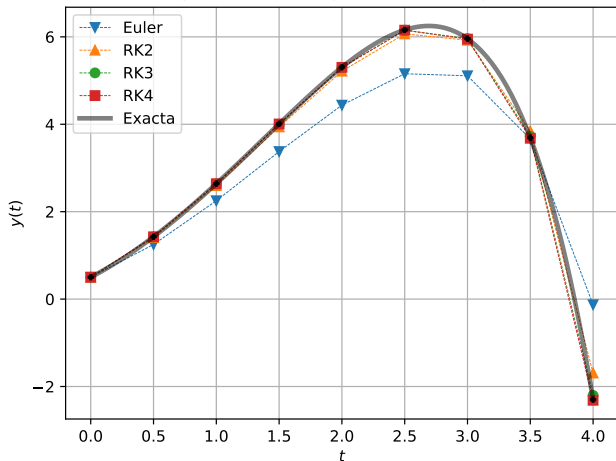
E Eul : 4.6382, E RK2 : 1.4779, E RK3 : 0.3717, E RK4 : 0.2793



EJERCICIO 9: COMPARACIÓN DE MÉTODOS CON $N_t = 4$ 

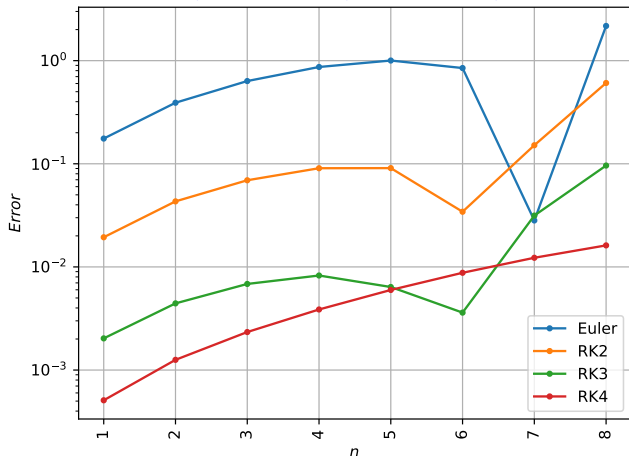
EJERCICIO 9: COMPARACIÓN DE MÉTODOS CON $N_t = 8$ Solución y aproximación $Nt = 8$

E Eul : 2.7881, E RK2 : 0.6453, E RK3 : 0.1021, E RK4 : 0.0234



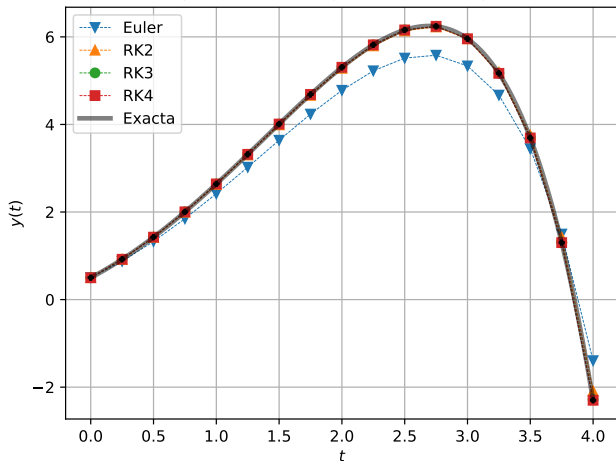
EJERCICIO 9: COMPARACIÓN DE MÉTODOS CON $N_t = 8$ Errores $N_t = 8$

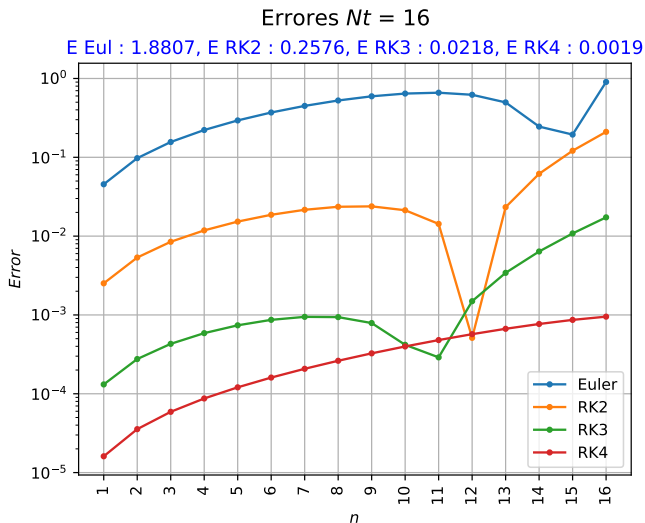
E Eul : 2.7881, E RK2 : 0.6453, E RK3 : 0.1021, E RK4 : 0.0234



EJERCICIO 9: COMPARACIÓN DE MÉTODOS CON $N_t = 16$ Solución y aproximación $N_t = 16$

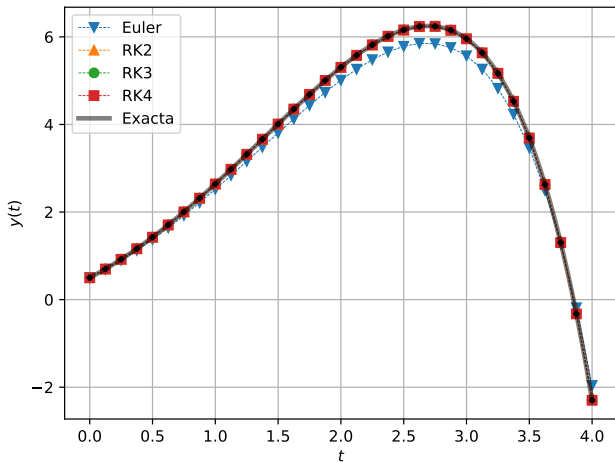
E Eul : 1.8807, E RK2 : 0.2576, E RK3 : 0.0218, E RK4 : 0.0019



EJERCICIO 9: COMPARACIÓN DE MÉTODOS CON $N_t = 16$ 

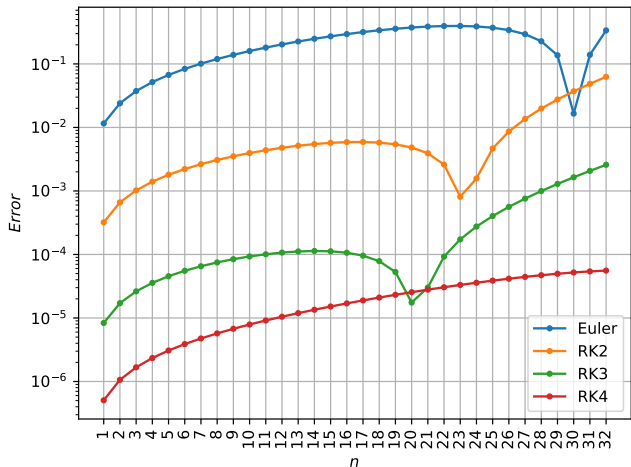
EJERCICIO 9: COMPARACIÓN DE MÉTODOS CON $N_t = 32$ Solución y aproximación $N_t = 32$

E Eul : 1.4408, E RK2 : 0.0973, E RK3 : 0.0042, E RK4 : 0.0002



EJERCICIO 9: COMPARACIÓN DE MÉTODOS CON $N_t = 32$ Errores $N_t = 32$


E Eul : 1.4408, E RK2 : 0.0973, E RK3 : 0.0042, E RK4 : 0.0002



CONTENIDO

- 1 PROBLEMAS DE VALOR INICIAL
- 2 MÉTODO DE EULER
 - Ejercicio 6.
 - Ejercicio 7.
 - Ejercicio 8.
- 3 MÉTODOS DE RUNGE-KUTTA
 - Ejercicio 9.
- 4 REFERENCIAS
- 5 CRÉDITOS

 [1] Richard Burden and J. Douglas Faires
Numerical Analysis
Ninth Edition, 2011 Brooks/Cole, Cengage Learning

 [2] R.J. Leveque,
*Finite Difference Method for Ordinary and Partial Differential Equations:
Steady State and Time-Dependent Problems* ,
Society for Industrial and Applied Mathematics (SIAM), Philadelphia, 2007.

CONTENIDO

- 1 PROBLEMAS DE VALOR INICIAL
- 2 MÉTODO DE EULER
 - Ejercicio 6.
 - Ejercicio 7.
 - Ejercicio 8.
- 3 MÉTODOS DE RUNGE-KUTTA
 - Ejercicio 9.
- 4 REFERENCIAS
- 5 CRÉDITOS

Dr. Luis M. de la Cruz Salas

Departamento de Recursos Naturales

Instituto de Geofísica

Universidad Nacional Autónoma de México



Trabajo realizado con el apoyo del Programa UNAM-DGAPA-PAPIME PE101019

