Tarea 2

Matemáticas para la Ciencia de Datos

Docente: **Briceyda B. Delgado**

Alumno: **Luis Fernando Izquierdo Berdugo**

Fecha: **26 de Agosto de 2024**

Instrucciones

Problema 1 (60 puntos) Considere el problema con valor inicial

$$r'(t)=r(t)+2t-t^2 \; r(0)=1 \; t \in [0,T)$$

1. Encuentre la solución analítica de la ecuación diferencial.
2. Derive una fórmula iterativa usando el `método de Euler hacia adelante (forwards)`, siguiendo la notación del libro, cada iteración la denotamos por y_n y suponemos que $y_0=r(0)=1$.
3. Repita el inciso anterior, usando la notación z_n para las iteraciones del `método de Euler hacia atrás (backwards)`, encuentre la fórumla explícita.

Problema 2 (40 puntos)

1. Tomando $\Delta t=\frac{1}{10}$ calcule y_1, y_2, y_3 y z_1, z_2, z_3. Compare estos valores obtenidos de la solución analítica del inciso `(1)`, $r(\Delta t)$, $r(2\Delta t)$, $r(3\Delta t)$ y calcule los errores relativos en cada caso.
2. Escriba un programa en Python que calcule ambos métodos de Euler.
3. Realice una gráfica comparativa entre las soluciones numéricas obtenidas por ambos métodos de Euler y la solución analítica obtenida en el inciso `(1)`.

Inciso (a) - Solución Analítica

Si tenemos la ecuación:

$$r'(t)=r(t)+2t-t^2$$

se puede observar que la solución a tener en cuenta es el método de `Factor Integrante`, siendo la forma para este caso:

$$y'+p(x)y=q(x)$$

Si se reescribe la ecuación original, se obtiene:

$$r'(t) - r(t) = 2t - t^2$$

Siendo:

$$p(t) = -1 \; q(t) = 2t - t^2$$

Para encontrar el factor integrante $\mu(t)$ se calcula con:

$$\mu(t) = e^{\int p(t)\, dt}$$

$$\mu(t) = e^{\int -1\, dt}$$

$$\mu(t) = e^{-t}$$

El siguiente paso será usar la fórmula de factor integrante para resolver la ecuación, siendo esta:

$$y\,\mu = \int q\,\mu\, dx$$

$$r(t)e^{-t} = \int (2t - t^2)e^{-t}\, dt$$

Se resuelve el lado derecho por integración por partes y se obtiene:

$$r(t)e^{-t} = t^2 e^{-t} + C$$

Simplificando:

$$r(t) = t^2 + C\, e^{t}$$

Sabiendo que $r_0 = 1$:

$$1 = 0^2 + C\, e^{0}$$

$$C = 1$$

Entonces la solución analítica del problema será

$$r(t) = t^2 + e^{t}$$

Evaluando con valores de 0 a 1 y saltos de 0.1

y_n	**Sustitución**	**Resultado**
y_0		1
y_1	$(0.1)^2 + e^{0.1}$	1.115170918
y_2	$(0.2)^2 + e^{0.2}$	1.261402758
y_3	$(0.3)^2 + e^{0.3}$	1.439858808
y_4	$(0.4)^2 + e^{0.4}$	1.651824698
y_5	$(0.5)^2 + e^{0.5}$	1.898721271

y_n	**Sustitución**	**Resultado**
y_6	$(0.6)^2+e^{0.6}$	2.1821188
y_7	$(0.7)^2+e^{0.7}$	2.503752707
y_8	$(0.8)^2+e^{0.8}$	2.865540928
y_9	$(0.9)^2+e^{0.9}$	3.269603111
y_{10}	$(1)^2+e^{1}$	3.718281828

Inciso (b) - Método de Euler hacia adelante

Usando la fórmula del libro para el Método de Euler para adelante (o metodo implícito), se puede obtener:

$$\frac{y_{n+1}-y_n}{\Delta t}=y_n+2t-t^2$$

Despejando

$$y_{n+1}=y_n+\left(y_n+2t-t^2\right)\Delta t$$

Con esta fórmula ya se puede iterar sabiendo que $y_0=1$

Sabiendo que $\Delta t=\frac{1}{10}$ y asignando arbitrariamente $T=1$, se usa la fórmula para calcular N

$$N=\frac{T}{h}$$

$$N=\frac{1}{\frac{1}{10}}=10$$

Se evaluará 10 veces, de 0 a 1, con saltos de 0.1 en t

y_n	**Sustitución**	**Resultado**
y_0		1
y_1	$1+\left(1+2(0)-(0)^2\right)0.1$	1.1
y_2	$1.1+\left(1.1+2(0.1)-(0.1)^2\right)0.1$	1.229
y_3	$1.229+\left(1.229+2(0.2)-(0.2)^2\right)0.1$	1.3879
y_4	$1.3879+\left(1.3879+2(0.3)-(0.3)^2\right)0.1$	1.57769
y_5	$1.57769+\left(1.57769+2(0.4)-(0.4)^2\right)0.1$	1.799459
y_6	$1.799459+\left(1.799459+2(0.5)-(0.5)^2\right)0.1$	2.0544049

y_n	**Sustitución**	**Resultado**
y_7	$2.0544049+\left(2.0544049+2\left(0.6\right)-\left(0.6\right)^2\right)0.1$	2.34384539
y_8	$2.34384539+\left(2.34384539+2\left(0.7\right)-\left(0.7\right)^2\right)0.1$	2.669229929
y_9	$2.669229929+\left(2.669229929+2\left(0.8\right)-\left(0.8\right)^2\right)0.1$	3.032152922
y_{10}	$3.032152922+\left(3.032152922+2\left(0.9\right)-\left(0.9\right)^2\right)0.1$	3.434368214

Inciso (c) - Método de Euler hacia atrás

Usando la fórmula del libro para el Método de Euler para adelante (o metodo implícito), se obtiene:

$$\frac{z_{n+1}-z_n}{\Delta t}=z_{n+1}+2t-t^2$$

Lo cual se puede simplificar a

$$z_{n+1}-z_n=\Delta t\, z_{n+1}+\Delta t\, 2t-\Delta t\, t^2$$

$$z_{n+1}=\Delta t\, z_{n+1}+\Delta t\, 2t-\Delta t\, t^2+z_n$$

$$z_{n+1}-\Delta t\, z_{n+1}=\Delta t\, 2t-\Delta t\, t^2+z_n$$

$$\left(1-\Delta t\right) z_{n+1}=\Delta t\, 2t-\Delta t\, t^2+z_n$$

$$z_{n+1}=\frac{\Delta t\, 2t-\Delta t\, t^2+z_n}{1-\Delta t}$$

Tomando en cuenta los datos previamente calculados ($\Delta t=\frac{1}{10}$, $N=10$ y $T=1$), se evalúa

z_n	**Sustitución**	**Resultado**
z_0		1
z_1	$\frac{\left(0.1\right)\left(2*0.1\right)-\left(0.1\right)\left(0.1^2\right)+1}{0.9}$	1.132222222
z_2	$\frac{\left(0.1\right)\left(2*0.2\right)-\left(0.1\right)\left(0.2^2\right)+1.132222222}{0.9}$	1.298024691
z_3	$\frac{\left(0.1\right)\left(2*0.3\right)-\left(0.1\right)\left(0.3^2\right)+1.298024691}{0.9}$	1.498916324
z_4	$\frac{\left(0.1\right)\left(2*0.4\right)-\left(0.1\right)\left(0.4^2\right)+1.498916324}{0.9}$	1.736573693
z_5	$\frac{\left(0.1\right)\left(2*0.5\right)-\left(0.1\right)\left(0.5^2\right)+1.736573693}{0.9}$	2.012859659

z_n	**Sustitución**	**Resultado**
z_6	$\frac{(0.1)(2*0.6)-(0.1)(0.6^2)+2.012859659}{0.9}$	2.329844065
z_7	$\frac{(0.1)(2*0.7)-(0.1)(0.7^2)+2.329844065}{0.9}$	2.689826739
z_8	$\frac{(0.1)(2*0.8)-(0.1)(0.8^2)+2.689826739}{0.9}$	3.095363044
z_9	$\frac{(0.1)(2*0.9)-(0.1)(0.9^2)+3.095363044}{0.9}$	3.549292271
z_{10}	$\frac{(0.1)(2*1)-(0.1)(1^2)+3.549292271}{0.9}$	4.05476919

Inciso (d) - Comparación

Con los datos calculados para la evaluación de las primeras 10 iteraciones, se hace la comparación entre los 3 métodos. En los cuales podemos observar que tienen valores bastante parecidos.

y_n	**Analítico**	**Euler hacia adelante**	**Euler hacia atrás**
y_0	1	1	1
y_1	1.115170918	1.1	1.132222222
y_2	1.261402758	1.229	1.298024691
y_3	1.439858808	1.3879	1.498916324

Se calcula el error relativo con la fórmula:

$$ErrorRelativo=\frac{Valor_{real}-Valor_{calculado}}{Valor_{real}}$$

En el caso de la solución analítica y Euler hacia adelante, se puede observar que Euler hacia adelante suele subestimar el valor de y_n, lo cual es característico del método.

y_n	**Analítico**	**Euler hacia adelante**	**Error relativo**
y_1	1.115170918	1.1	0.013604119
y_2	1.261402758	1.229	0.025687876
y_3	1.439858808	1.3879	0.036086044

Para la comparación entre la solución analítica y Euler hacia atrás, se tiene una sobreestimación del valoy de y_n po parte de Euler hacia atrás, lo cual también es característico del método, sin embargo, este suele dar resultados más estables que el método de Euler hacia adelante.

y_n	Analítico	Euler hacia atrás	Error relativo
y_1	1.115170918	1.132222222	− 0.015290305
y_2	1.261402758	1.298024691	− 0.029032704
y_3	1.439858808	1.498916324	− 0.041016186

Inciso (e) - Desarrollo en Python

Para desarrollar el programa se utilizarán las librerías numpy y sympy, de igual manera se importará de una vez la librería matplotlib, que se utilizará en el `inciso (f)`.

```
import numpy as np
import matplotlib.pyplot as plt
import sympy as sp
```

Lo primero será definir la ecuación diferencial en símbolos de sympy, con el objetivo de facilitar trabajar con estas variables. De iugal manera se definirá la ecuación

```
t = sp.symbols('t')
r = sp.Function('r')
```

Se establecen los parámetros conocidos para el problema

```
t0 = 0
y0 = 1
T = 1
h = 0.1
```

Se definirá la ecuación diferencial en forma numérica para poder utilizarla en el método de Euler hacia adelante.

```
def f(t, r):
    return r + 2*t - t**2
```

Lo siguiente será definir la función del método de euler hacia adelante. Esta función tomará los siguientes parámetros:

- `f`es la función que representa la ecuación diferencial a resolver
- `t0`es el valor inicial de la variable `t`
- `y0` es el valor inicial de la variable `y`
- `h`es el tamaño del salto o el incremento de `t` en cada iteración
- `T` es el tiempo final hasta el cual se calculará la solución. Con estos parámetros, definimos:
- `t`como un arreglo que contiene todos los valores de t, desde `t0` hasta `T`, con un incremento de `h`.
- `y`es el arreglo donde se guardarán los valores calculados de y_n en cada punto del tiempo.

Lo primero es definir al valor conocido de `y0` en el elemento 0 del arreglo `y`. A continuación se hará una iteración sobre todos los elementos de `t`, donde se asignará el valor de la siguiente y, siguiendo la fórmula:

$$y_{n+1} = y_n + \Delta t\left(f\left(t, r\right)\right)$$

Finalmente, se regresarán los valores de `t`y `y`calculados.

```
def euler_forward(f, t0, y0, h, T):
    t = np.arange(t0, T+h, h)
    y = np.zeros(len(t))
    y[0] = y0
    for i in range(len(t)-1):
        y[i+1] = y[i] + h*f(t[i], y[i])
    return t, y
```

Se evalúa la función

```
t_forward, r_forward = euler_forward(f, t0, y0, h, T)
print(r_forward)
```

```
[1.         1.1        1.229      1.3879     1.57769    1.799459
 2.0544049  2.34384539 2.66922993 3.03215292 3.43436821]
```

Euler hacia atrás será muy similar a Euler hacia adelante, sin embargo se creará otra función auxiliar, la cual servirá para obtener el resultao final. Esta función solamente devolverá la fórmula $2t - t^2$. La otra diferencia en la función de Euler hacia atrás será que la fórmula para calcular los reultados será:

$$y_{n+1} = y_n + \Delta t\left(f_{aux}\right)¿\frac{¿}{1 - \Delta t}$$

```
def f_aux(t):
    return 2*t - t**2

def euler_backward(f_aux, t0, y0, h, T):
    t = np.arange(t0, T+h, h)
    y = np.zeros(len(t))
    y[0] = y0
    for i in range(len(t)-1):
        y[i+1] = (y[i] + h*f_aux(t[i+1]))/(1-h)
    return t, y
```

Se ejecuta la función

```
t_backward, r_backward = euler_backward(f_aux, t0, y0, h, T)
print(r_backward)
```

```
[1.         1.13222222 1.29802469 1.49891632 1.73657369 2.01285966
 2.32984407 2.68982674 3.09536304 3.54929227 4.05476919]
```

Se aplica la solución analítica por medio de sympy. Primero se plantea como ecuación en la variable `eq`, luego se resuelve con la función `dsolve` y se simplifica con `simplify`, esto devuelve la expresión simbólica de la solución de la ecuación en la variable `r_t`.

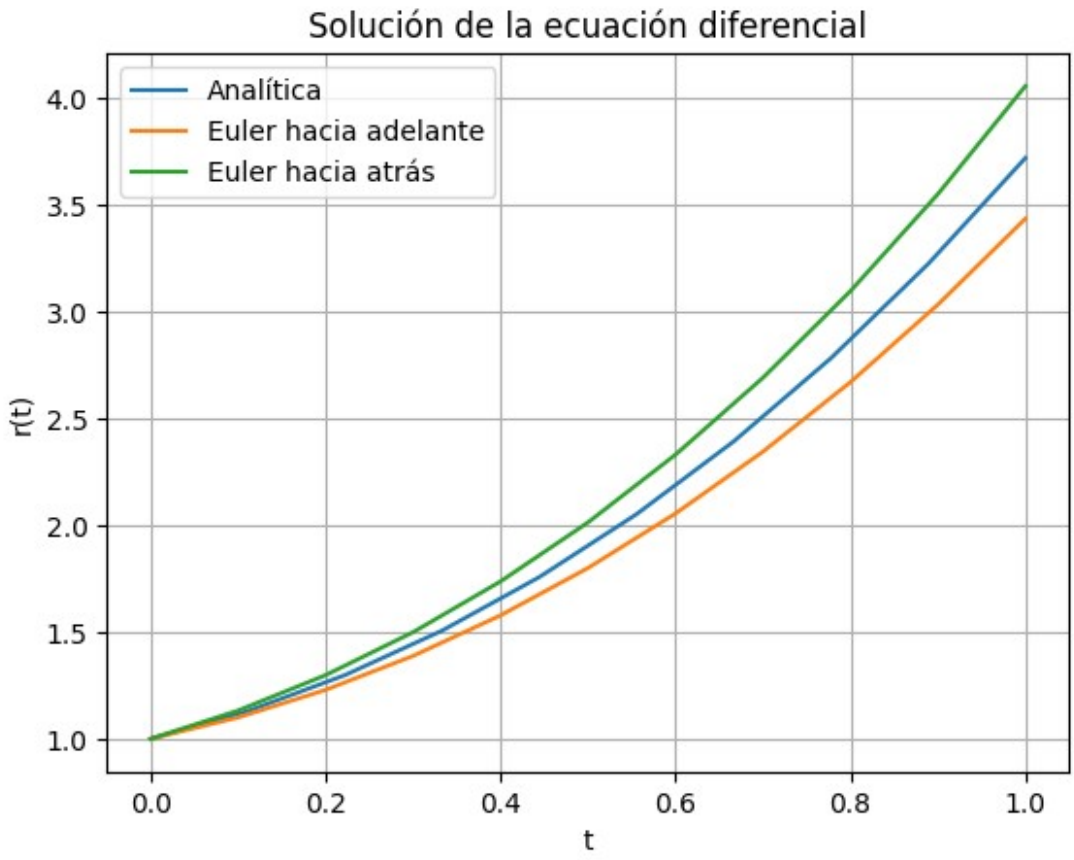
Se utiliza la función `linspace` de numpy para crear un arreglo con 10 valores que van desde `t0`hasta `T`, despues se hace un ciclo para cada valor `ti`, en el cual se sustituye la variable `t`de `r_t`por el valor numérico que está en `ti`, lo cual devuelve el resultado de la solución analítica.

```
eq = sp.Eq(r(t).diff(t), r(t) + 2*t - t**2)
sol = sp.dsolve(eq, ics={r(0): 1})
r_t = sp.simplify(sol.rhs)
t_analytical = np.linspace(t0, T, 10)
r_analytical = np.array([r_t.subs(t, ti) for ti in t_analytical])
```

Inciso (f) - Gráficas

Se utiliza el módulo matplotlib para generar una gráfica con las soluciones a los tres métodos de solución.

```
# Graficamos las soluciones
plt.plot(t_analytical, r_analytical,label='Analítica')
plt.plot(t_forward, r_forward, label='Euler hacia adelante')
plt.plot(t_backward, r_backward, label='Euler hacia atrás')
plt.legend()
plt.xlabel('t')
plt.ylabel('r(t)')
plt.title('Solución de la ecuación diferencial')
plt.grid(True)
plt.show()
```



En la gráfica se puede observar que los 3 modelos se aproximan a la misma solución. Si se toma la solución analítica como la real, se corrobora lo cmentado anteriormente, que el método de `Euler hacia adelante` tiende a subestimar los valores de `r(t)` y el método de `Euler hacia atrás` suele sobreestimarlos.

Sería interesante cambiar el tamaño del salto por uno más pequeño, para poder probar la teoría de que mientras menor el tamaño del salto, más precisos suelen ser los modelos de Euler.

Bibliografía

- Tveito et al. (n.d.) Elements of scientific computing. Springer Heidelberg Dordrecht London New York. https://doi.org/10.1007/978-3-642-11299-7
- Google. (2024). Gemini (Aug 10 version) [Large language model]. https://gemini.google.com/app