Ejercicio 2: Decaimiento radiocativo

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

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

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

- **Aproximar el decaimiento radioctivo usando los métodos *forward Euler* y *backward Euler*.** (a) Escribir y analizar las fórmulas de los métodos, (b) implementar los métodos en Python y (c) mostrar el comportamiento de ambos métodos para $\lambda = 1.5$, $y_0 = 20$, $T_{max} = 10$ y $N_t = 7, 8, 9, 10, 20$, (d) ¿Cuántos puntos se necesitan para que el error sea menor que 1.0 en cada método?

A) Escribir y analizar las fórmulas de los métodos

Forward Euler:

$$\begin{aligned} y_{n+1} &= y_n - h_t \lambda y_n \\ \Longrightarrow y_{n+1} &= (1 - h_t \lambda)\, y_n \end{aligned}$$

$$\begin{aligned} \text{Paso 1: } y_1 &= A\, y_0 \\ \text{Paso 1: } y_2 &= A^2\, y_0 \\ \vdots \quad & \vdots \quad \vdots \\ \text{Paso } N_t: y_{n+1} &= A^{N_t}\, y_0 \end{aligned}$$

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:

$$\begin{aligned} y_{n+1} &= y_n - h_t \lambda y_{n+1} \\ \Longrightarrow y_{n+1} &= (1 + h_t \lambda)^{-1}\, y_n \end{aligned}$$

$$\begin{aligned} \text{Paso 1: } y_1 &= B^{-1}\, y_0 \\ \text{Paso 1: } y_2 &= B^{-2}\, y_0 \\ \vdots \quad & \vdots \quad \vdots \\ \text{Paso } N_t: y_{n+1} &= B^{-N_t}\, y_0 \end{aligned}$$

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.

B) Implementar los métodos en Python

Spyder (Python 3.8)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\andy_\SpyHeat\untitled0.py

untitled0.py* | Conveccion_NoEstacionario1D_Cond_Dirichlet.py | Funciones_No_estacionario.py | 1D_Heat_05.py | Funcio

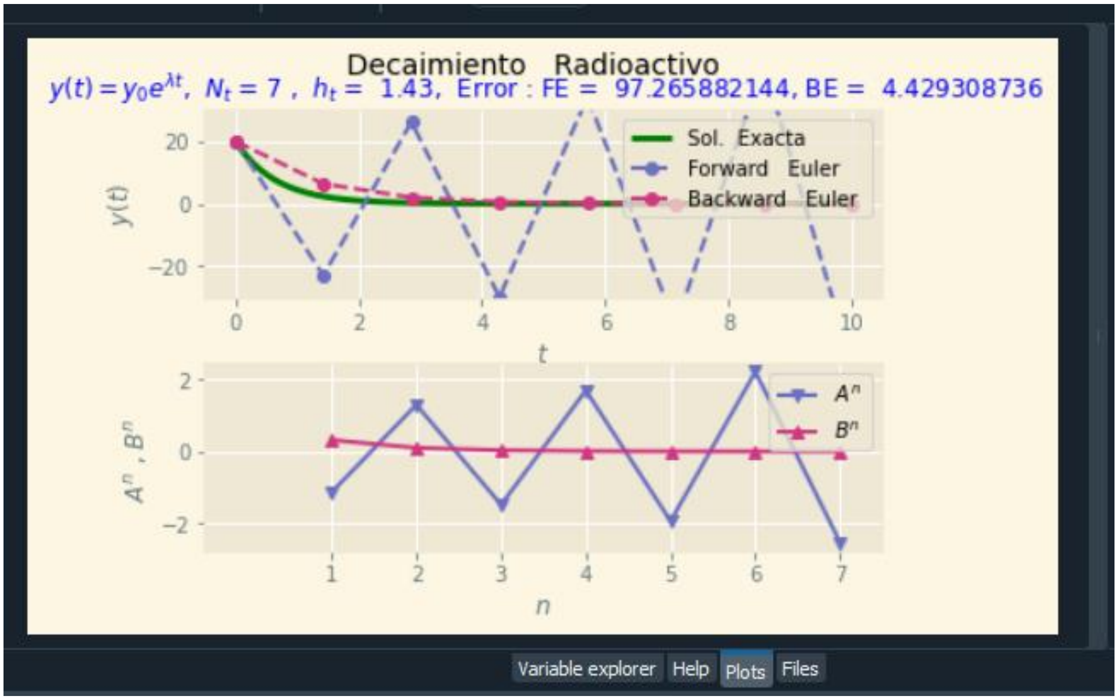
```python
# -*- coding: utf-8 -*-
"""
Created on Mon Dec 14 00:43:10 2020

@author: andy_
"""
import numpy as np
import matplotlib.pyplot as plt

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 = 68
Tmax = 10
ht = mesh (0, Tmax , Nt)
```

```python
tl = np. linspace (0, Tmax , 100)
y_exacta = exactSolution (tl , y0 , lam )
y_exac_p = exactSolution (t, y0 , lam)
norma_error_f = np.linalg.norm(yf - y_exac_p ,2)
norma_error_b = np.linalg.norm(yb - y_exac_p ,2)

Ecuacion = ' $y(t) = y_0 e^{\lambda t}$,' + '  $N_t$ ' + '= {} '. format (Nt) + ',  $h_t$ ' +
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
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 ()
```

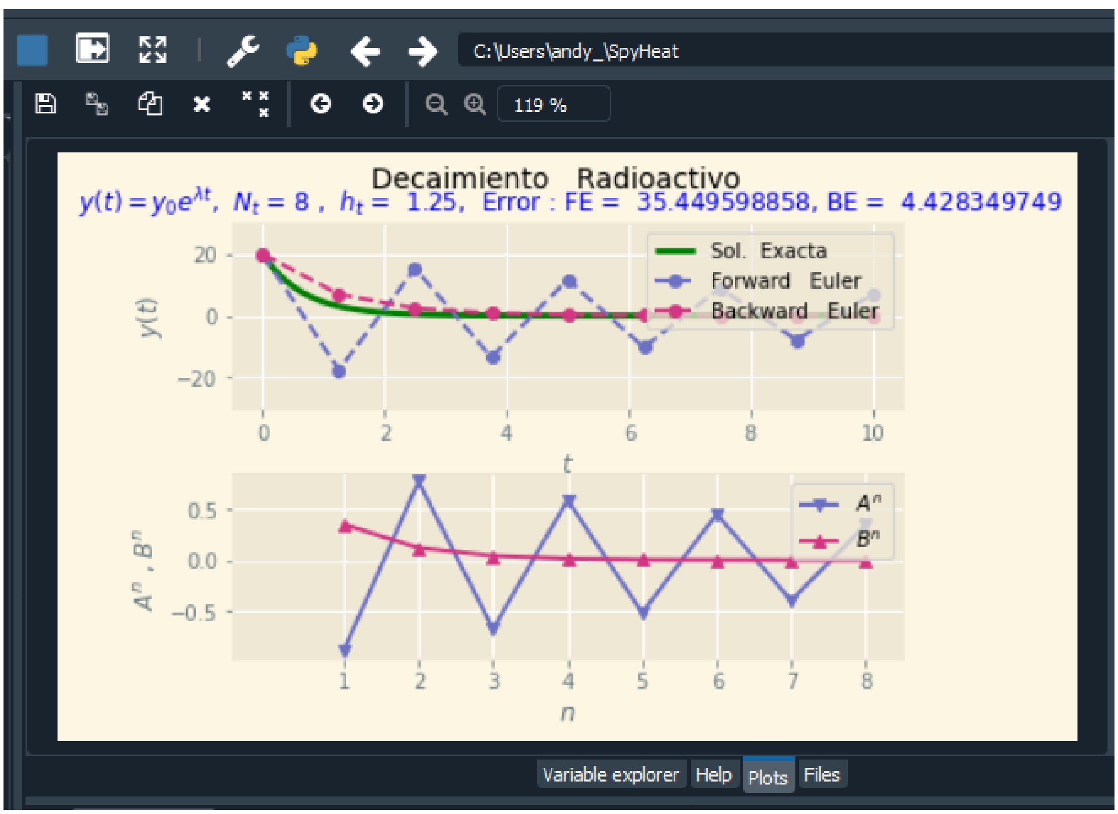
No documentation available

Click anywhere in this tooltip for additional help

C) Mostrar el comportamiento de ambos métodos para $\lambda = 1.5, y_0 = 20, T_{max} = 10$ **y** $Nt = 7$



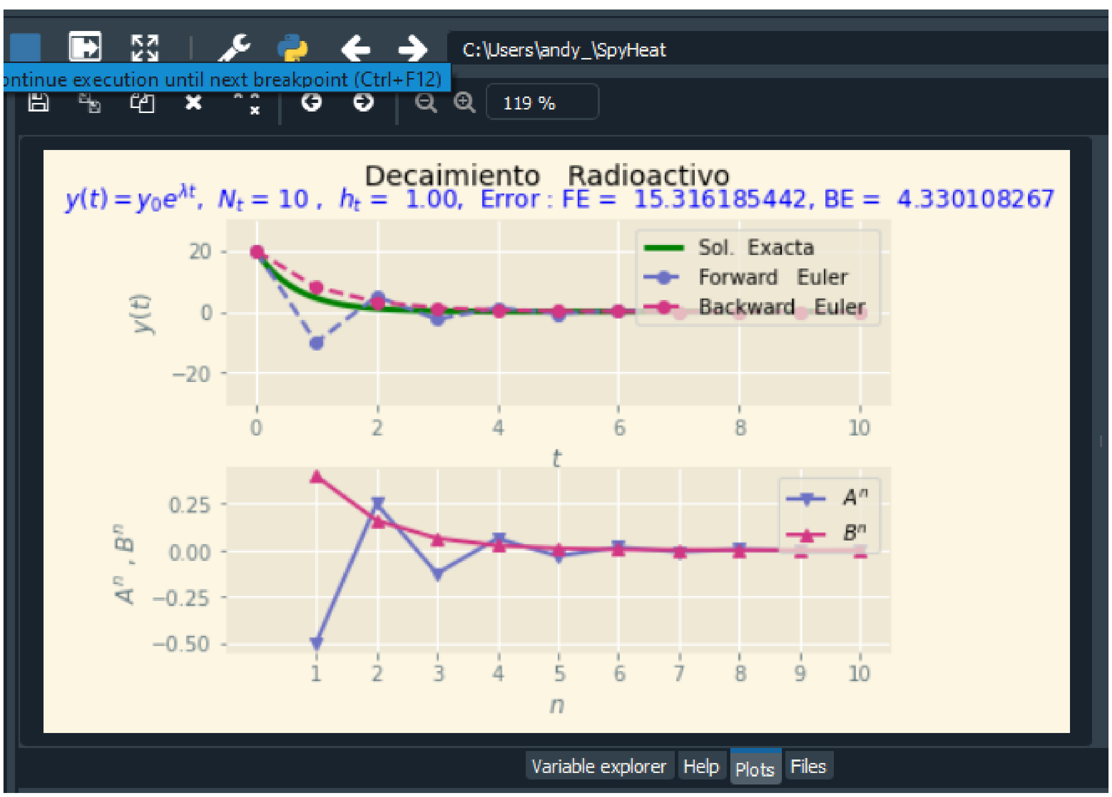
$Nt = 8$



Nt=9

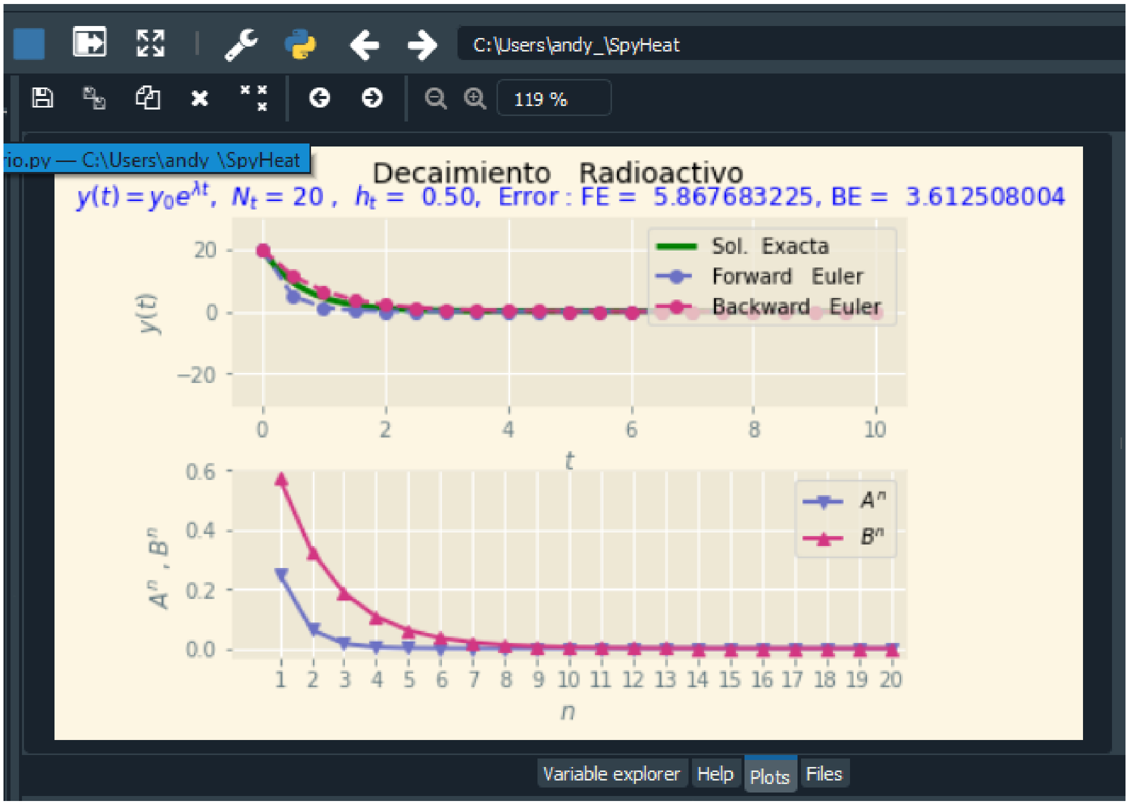
C:\Users\andy_\SpyHeat
119 %
ptions
Decaimiento Radioactivo
$y(t) = y_0 e^{\lambda t}$, N_t = 9 , h_t = 1.11, Error : FE = 20.592451368, BE = 4.389771777
Sol. Exacta
Forward Euler
Backward Euler
y(t)
t
A^n
B^n
A^n , B^n
n
Variable explorer
Help
Plots
Files

Nt = 10



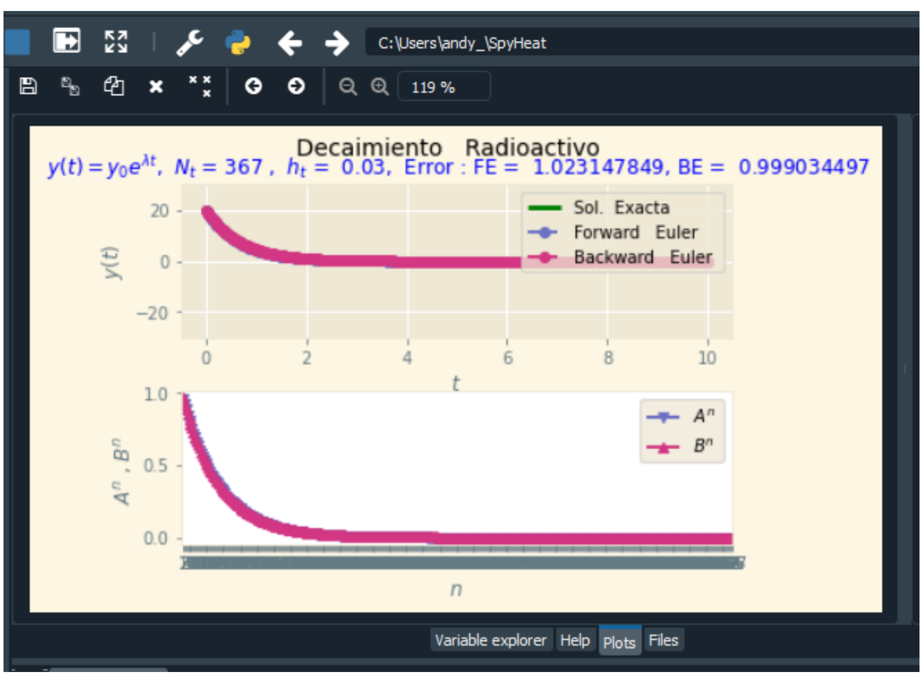
C:\Users\andy_\SpyHeat
ontinue execution until next breakpoint (Ctrl+F12)
119 %
Decaimiento Radioactivo
$y(t) = y_0 e^{\lambda t}$, N_t = 10 , h_t = 1.00, Error : FE = 15.316185442, BE = 4.330108267
Sol. Exacta
Forward Euler
Backward Euler
y(t)
t
A^n
B^n
A^n , B^n
n
Variable explorer
Help
Plots
Files

Nt = 20

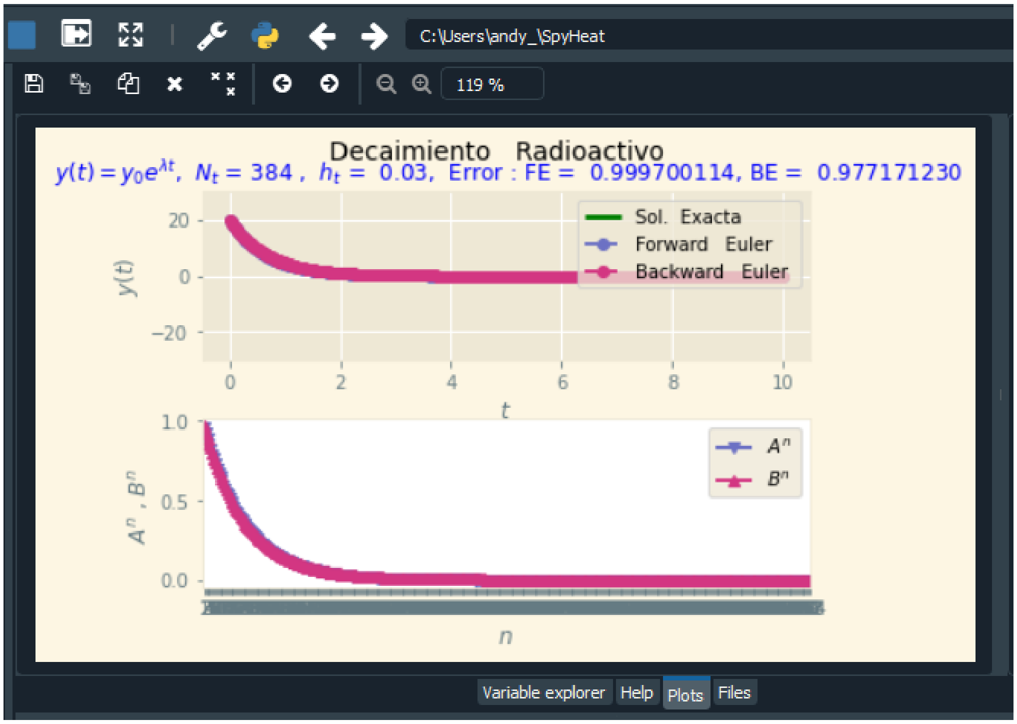


D) ¿Cuántos puntos se necesitan para que el error sea menor que 1 en cada método?

- Se necesitaron Se necesitaron $N_t = 367$ para que el método BE tenga una error menor que 1.

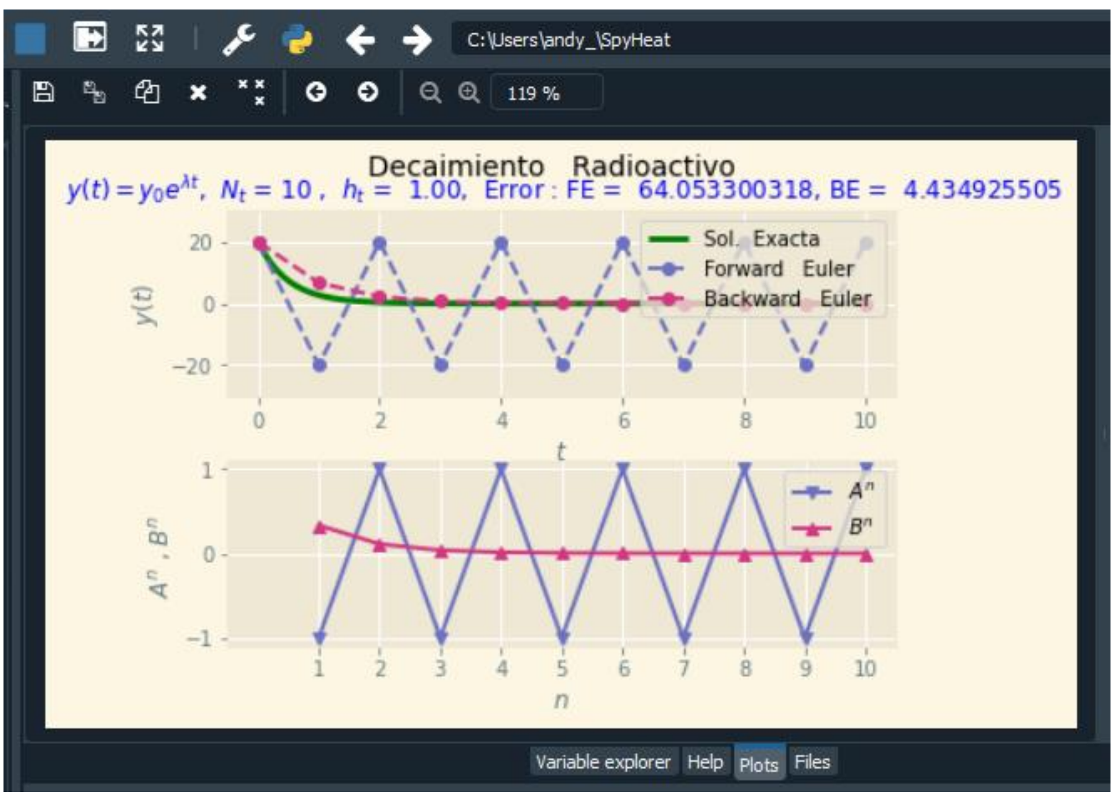


- Se necesitaron $N_t = 384$ para que el método FE tenga una error menor que 1.

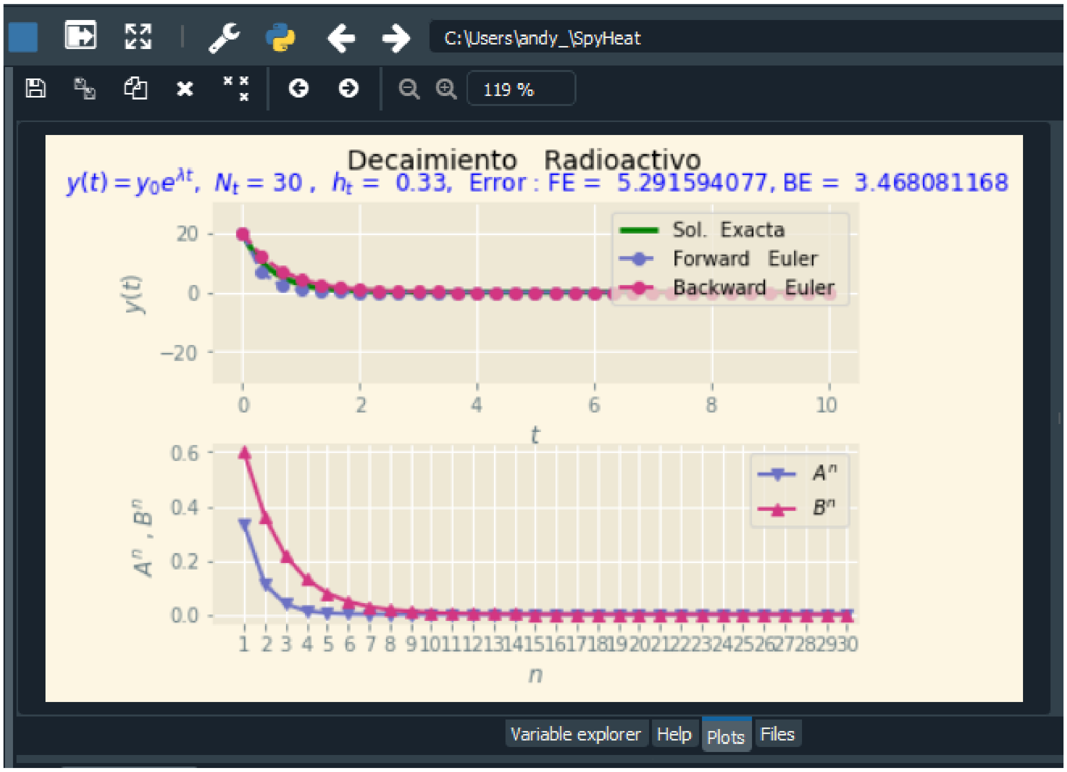


E) Ahora use $\lambda = 2$ y $conteste$

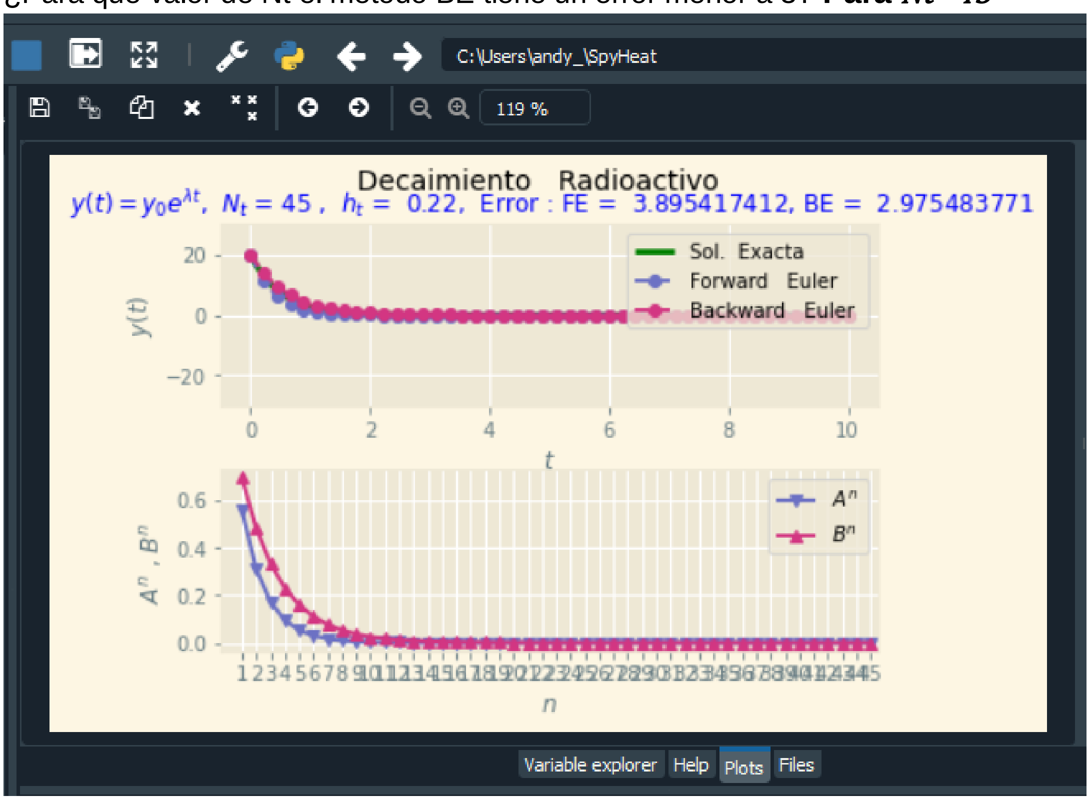
- ¿Para qué valor de Nt el método FE converge? **Para *Nt=10***



- ¿Para qué valor de Nt el método FE deja de oscilar? **Para *Nt=30***



- ¿Para qué valor de Nt el método BE tiene un error menor a 3? **Para *Nt=45***



- ¿Para qué valor de Nt el método FE tiene un error menor a 3? **Para *Nt=68***