Primera tarea

Modelaje matemático (CO4611)

Jhonny Lanzuisi

8 de Junio de 2022

Índice

La configuración inicial de python: importar librerías necesarias, usar el estilo `seaborn` de los gráficos y el comando `lp` que imprime directamente en LaTeX.

```
import sympy as sym
import numpy as np
from scipy.integrate import solve_ivp
import matplotlib.style
import matplotlib.pyplot as plt
import matplotlib

matplotlib.style.use('seaborn')

matplotlib.rcParams.update({
    'figure.autolayout': True,
})

def lp(input_string):
   print('$$' + sym.latex(input_string) + '$$')
```

1.a)

Como es una ecuación de primer orden podemos hallar la solución con el factor integrante:

$$\exp\left(\int \frac{2t}{1+t^2}\, dt\right).$$

Usaremos python (sympy) para sacar las cuentas. La solución de la integral anterior es:

```
p_t, t = sym.symbols('p_t, t')
p_t = 2*t/(1+t**2)
int_p_t = sym.Integral(p_t,t)
lp(sym.Eq(int_p_t,int_p_t.doit()))
```

$$\int \frac{2t}{t^2+1}\, dt = \log\left(t^2+1\right)$$

Para obtener la solución general solo hace falta calcular la siguiente integral.

$$\int \frac{1}{t^2+1} e^{\log(t^2+1)}\, dt = \int dt = t.$$

La solución general viene dada por:

$$y = \frac{C_1 + t}{t^2+1}.$$

Podemos ver el gráfico de la solución en la figura 1.

```
p1 = sym.plot(show=False, legend=True)
for i in range(-5,6):
    p = sym.plot((t+i)/(t**2+1), show=False, label='C_1= ' + str(i))
    p1.append(p[0])

p1.save('p1.pdf')
```

1.b)

En esta ecuación podemos separar variables para obtener:

$$\int (1-y^2)\, dy = \int x^2\, dx,$$

de donde se sigue:

$$y - \frac{y^3}{3} = \frac{x^3}{3} + C_1$$

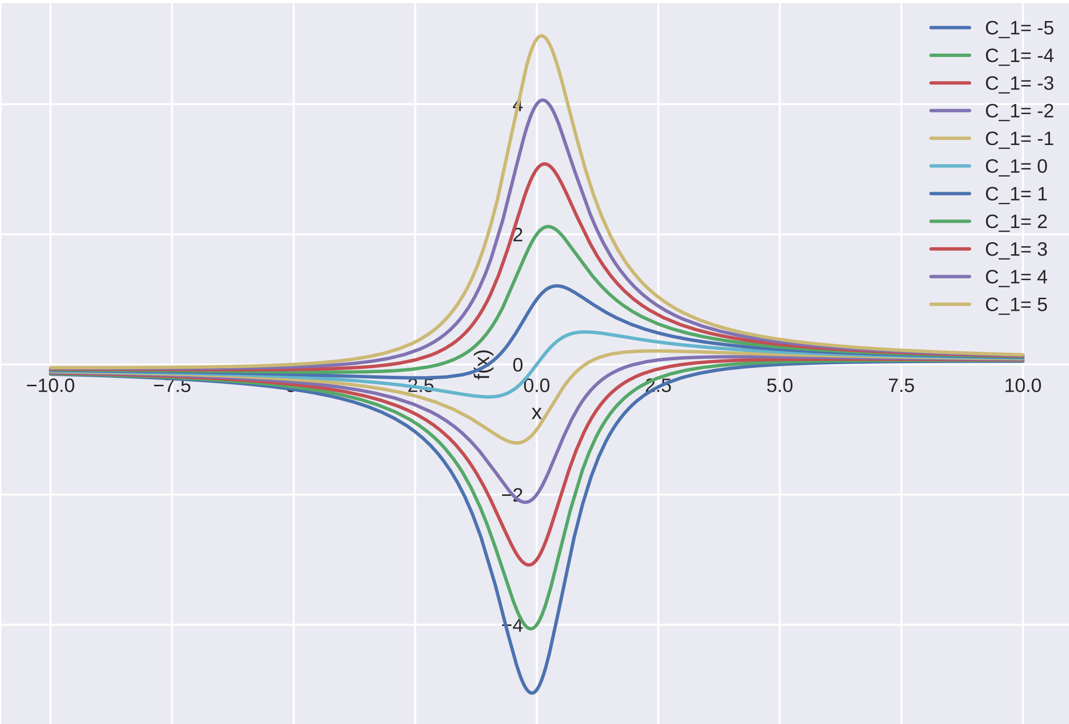

Figura 1: $y = \frac{C_1+t}{t^2+1}$

al multiplicar ambos lados por 3

$$3y - y^3 = x^3 + C_2.$$

Para hallar la solución en $(1, 1/2)$ sustituimos:

$$2 = \frac{1}{8} + C_2$$

de donde $C_2 = 15/8$ y la solución particular es:

$$y - y^3 - x^3 = \frac{15}{8}.$$

Podemos ver el gráfico de la solución particular en la figura 2.

```
x,y = sym.symbols('x,y')
p2 = sym.plot_implicit(y - y**3 - x**3 - 15/8,show=False)
p2.save('p2.pdf')
```

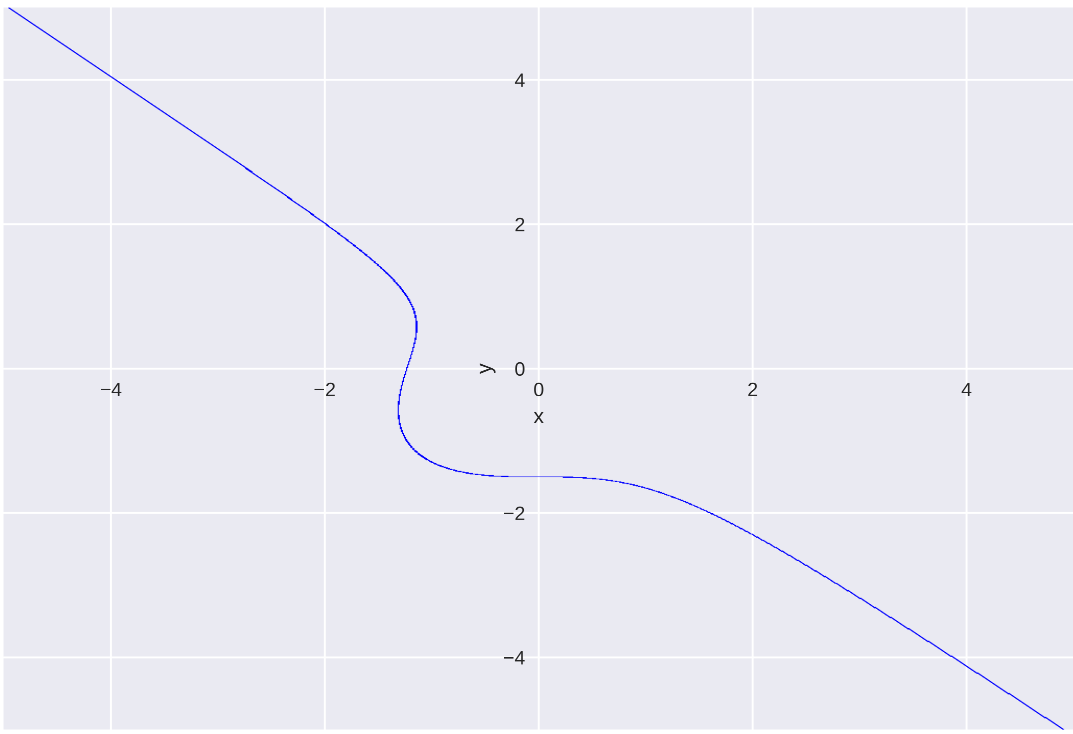

Figura 2: $y - y^3 - x^3 = \frac{15}{8}$

En general las soluciones lucen como en la figura 3.

```python
p3 = sym.plot(show=False)
for i in range(-5,6):
    p = sym.plot_implicit(y - y**3 - x**3 - i,show=False)
    p3.append(p[0])

p3.save('p3.pdf')
```

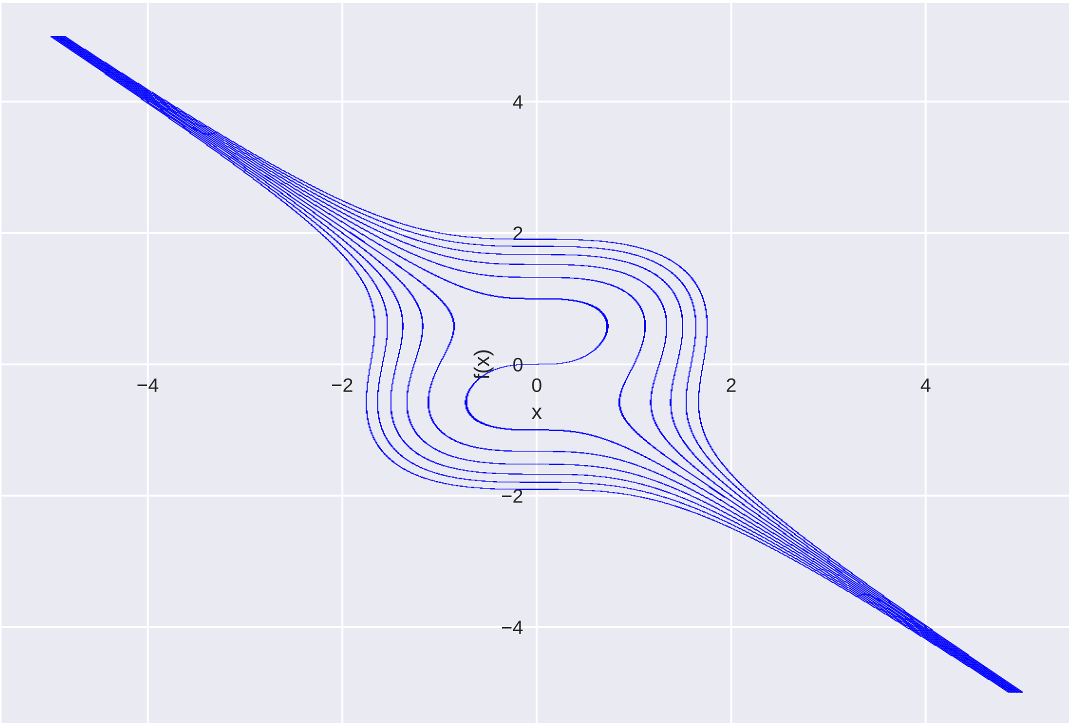

Figura 3: $3y - y^3 = x^3 + C_2$

1.c)

Podemos separar las variables de la ecuación para obtener:

$$\int e^y \, dy = \int t^3 + t \, dt,$$

de donde se sigue que

$$e^y = \frac{t^4}{4} + \frac{t^2}{2} + C_1$$

y la solución viene dada por:

$$y = \log(\frac{t^4}{4} + \frac{t^2}{2} + C_1)$$

Para encontrar la solución particular, sustituimos:

$$1 = \log(\frac{1}{4} + \frac{1}{2} + C_1)$$

de donde $C_1 = e - \frac{3}{4}$ y la solución buscada es:

$$y = \log(\frac{t^4}{4} + \frac{t^2}{2} + e - \frac{3}{4})$$

Podemos ver el gráfico de la solución en la figura 4.

```
p4 = sym.plot(sym.log(t**4/4 + t**2/2 + sym.E - 3/4), (t,-5,5),show=False)

p4.save('p4.pdf')
```

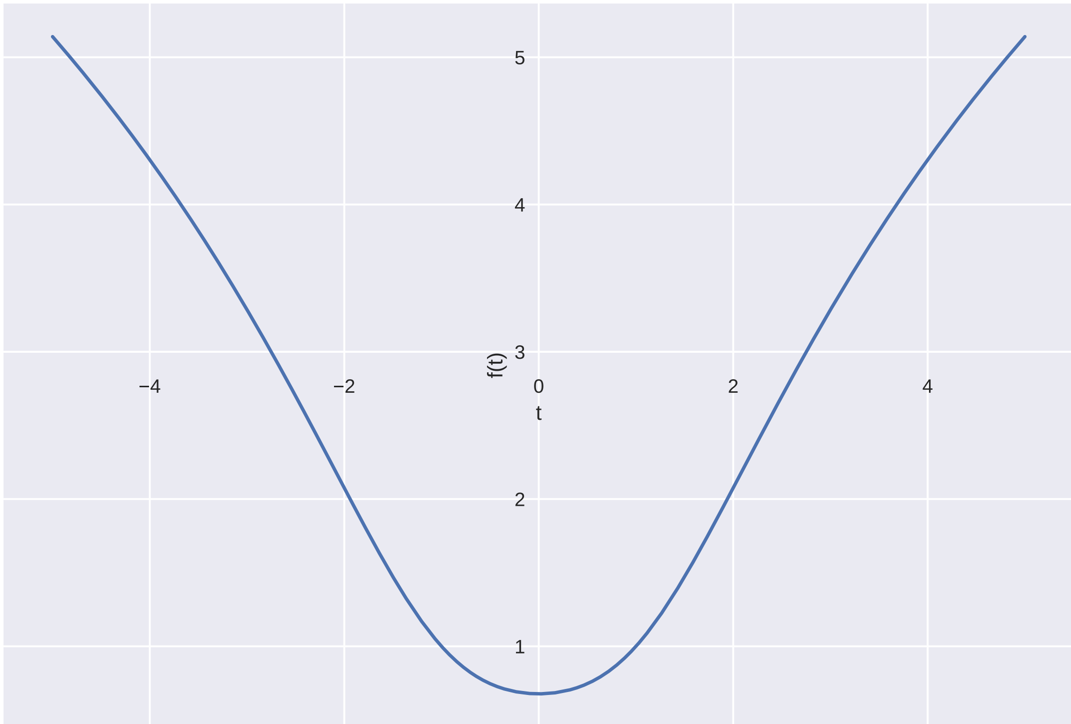

Figura 4: $y = \log(\frac{t^4}{4} + \frac{t^2}{2} + e - \frac{3}{4})$

En general las soluciones lucen así como en la figura 5.

```
p5 = sym.plot(show=False, legend=True)
for i in range(-5,6):
    p = sym.plot(
```

```
        sym.log(t**4/4 + t**2/2 + i),
        (t,-5,5),
        show=False,
        label='C_1= ' + str(i)
    )
    p5.append(p[0])

p5.save('p5.pdf')
```

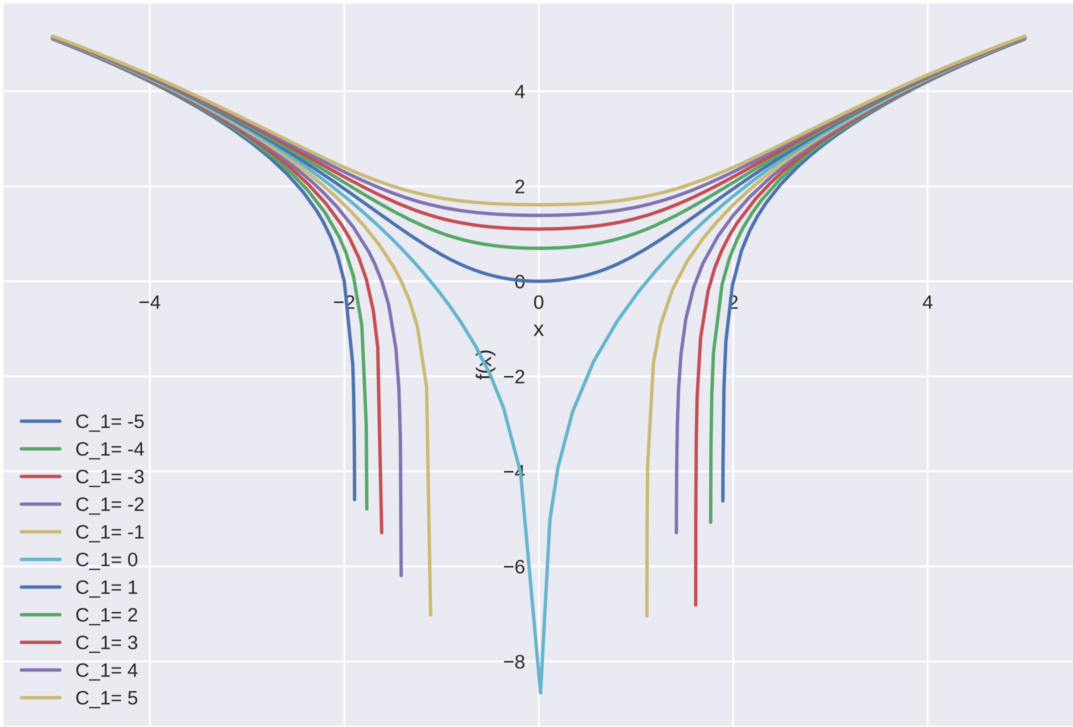

Figura 5: $y = \log(\frac{t^4}{4} + \frac{t^2}{2} + C_1)$

2

Podemos separar las variables para integrar:

$$\int \frac{1}{\sqrt{1-y^2}}\, dy = \int t\, dt.$$

Calculamos la primera integral:

```
lp(sym.integrate(1/sym.sqrt(1-y**2), y))
```

$$\operatorname{asin}(y)$$

Tenemos entonces que:

$$\sin^{-1}(y) = \frac{t^2}{2} + C_1$$

de donde se sigue que

$$y = \sin(\frac{t^2}{2} + C_1).$$

Para la solución particular sustituimos $y(0) = 1$

$$1 = \sin(C_1)$$

de donde $C_1 = \sin^{-1}(1) = \pi/2$.

La solución buscada es

$$y = \sin(\frac{t^2}{2} + \frac{\pi}{2}).$$

Podemos ver el gráfico de la solución en la figura 6.

```
p6 = sym.plot(sym.sin(t**2/2 + sym.pi/2), (t,-3,3),show=False)

p6.save('p6.pdf')
```

En general las soluciones lucen así como el la figura 7.

```
p7 = sym.plot(show=False, legend=True)
for i in range(-2,3):
    p = sym.plot(
        sym.sin(t**2/2 + i),
        (t,-3,3),
        show=False,
        label='C_1= ' + str(i)
    )
    p7.append(p[0])

p7.save('p7.pdf')
```

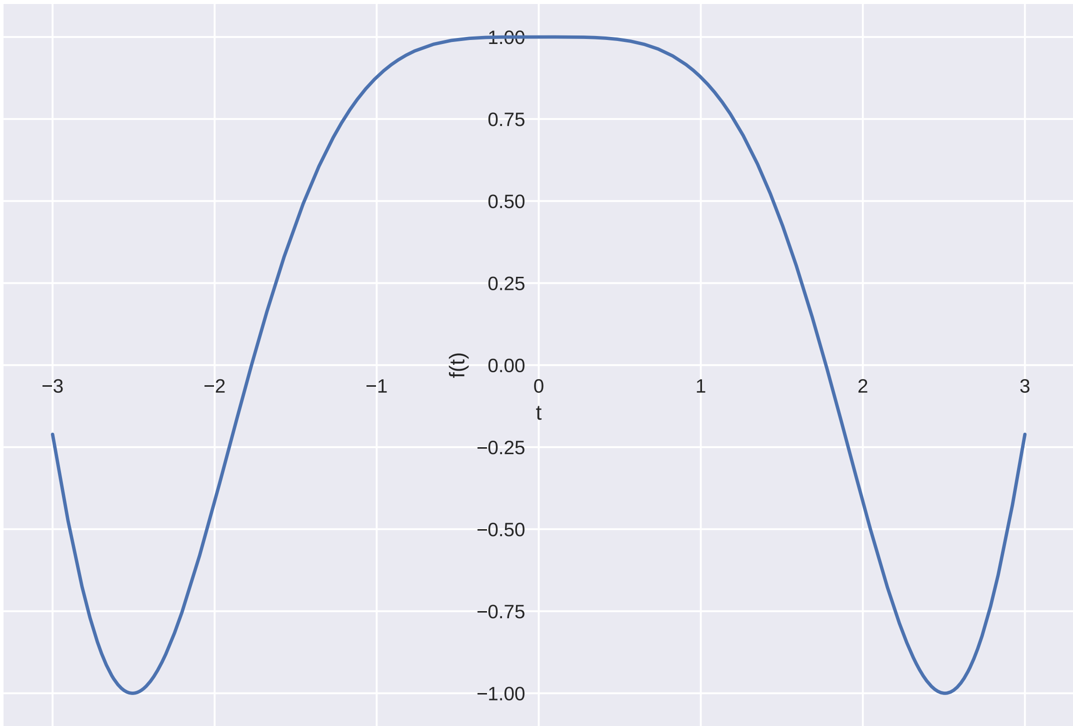

Figura 6: $y = \sin(\frac{t^2}{2} + \frac{\pi}{2})$

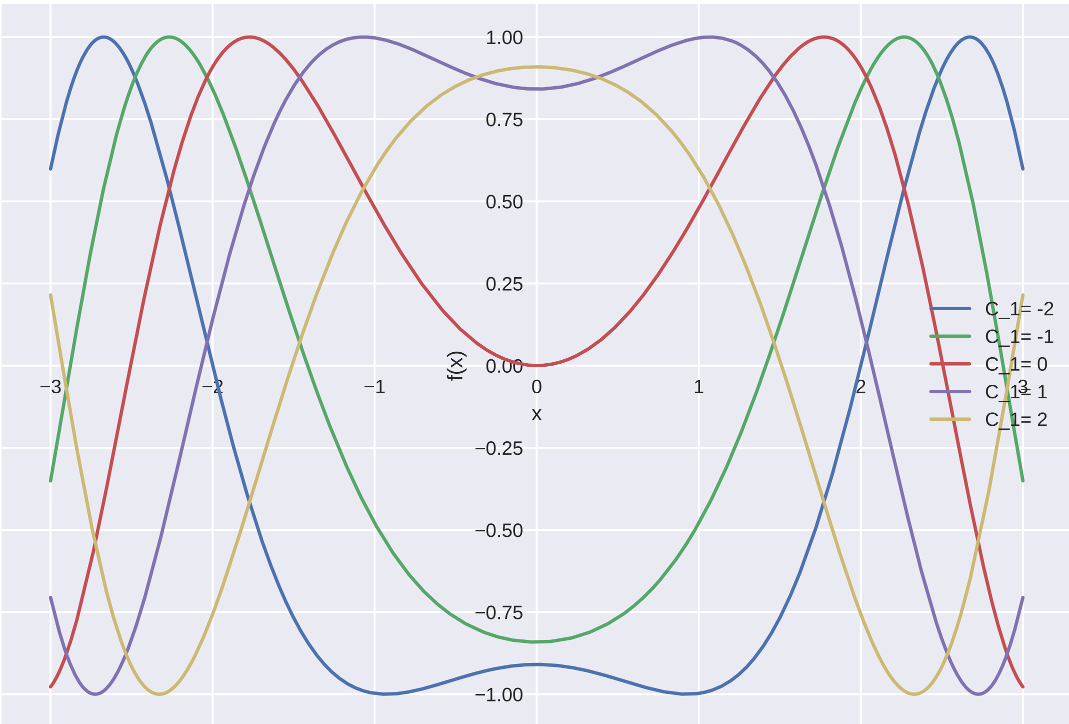

Figura 7: $y = \sin(\frac{t^2}{2} + C_1)$

3

Como es una ecuación de primer orden podemos hallar la solución con el factor integrante:

$$e^{\int 2tdt} = e^{t^2}$$

Hace falta también la siguiente integral:

$$\int 2te^{t^2}$$

cuyo valor es:

```
inte = sym.Integral(2*t*sym.exp(t**2),t)
lp(sym.Eq(inte, inte.doit()))
```

$$\int 2te^{t^2}\,dt = e^{t^2}$$

La solución general es entonces:

$$y = \frac{e^{t^2} + C_1}{e^{t^2}} = 1 + C_1 e^{-t^2}$$

Para la solución particular sustituimos $y(0) = 2$:

$$2 = 1 + \frac{C_1}{e^0} \implies C_1 = 1$$

Y obtenemos, finalmente:

$$y = 1 + e^{-t^2},$$

donde $t \to \infty$ implica que $y \to 1$.

El gráfico de la solución esta en la figura 8.

```
p8 = sym.plot(1 + sym.exp(-(t**2)), (t,0,5), show=False)

p8.save('p8.pdf')
```

Y el gráfico de la solución general en la figura 9.

```
p9 = sym.plot(show=False,legend=True)
for i in range(-5,6):
    p = sym.plot(
        1+i*sym.exp(-(t**2)),
        (t,0,5),
        show=False,
```

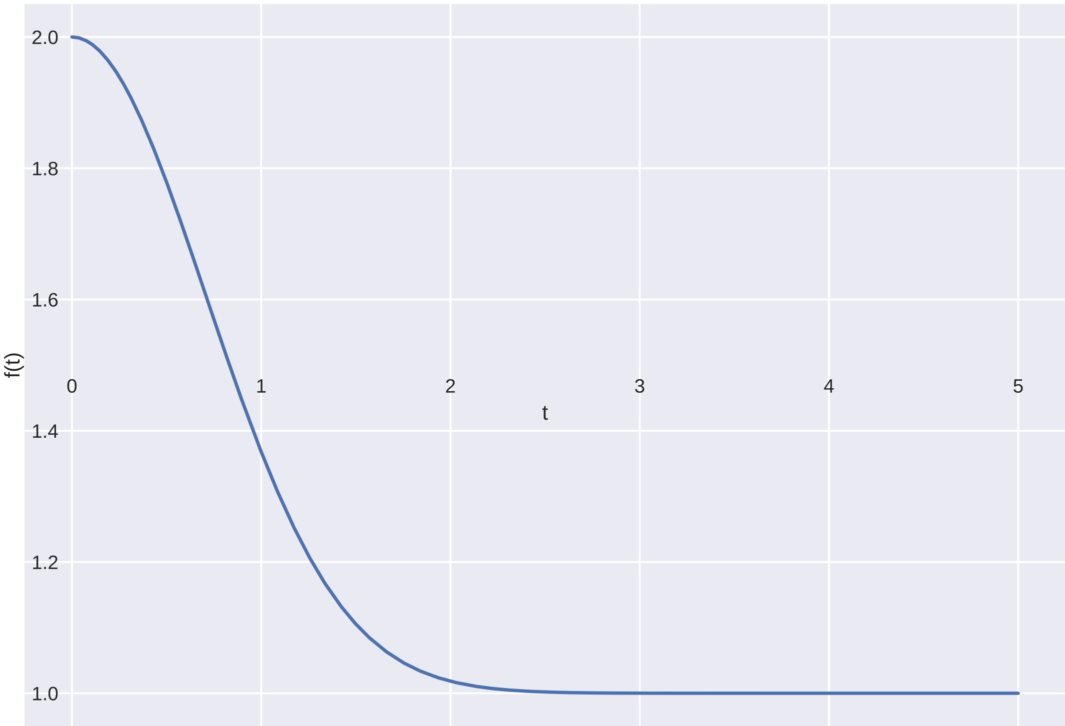

Figura 8: $y = 1 + e^{-t^2}$

```
        label='C_1= ' + str(i)
    )
    p9.append(p[0])

p9.save('p9.pdf')
```

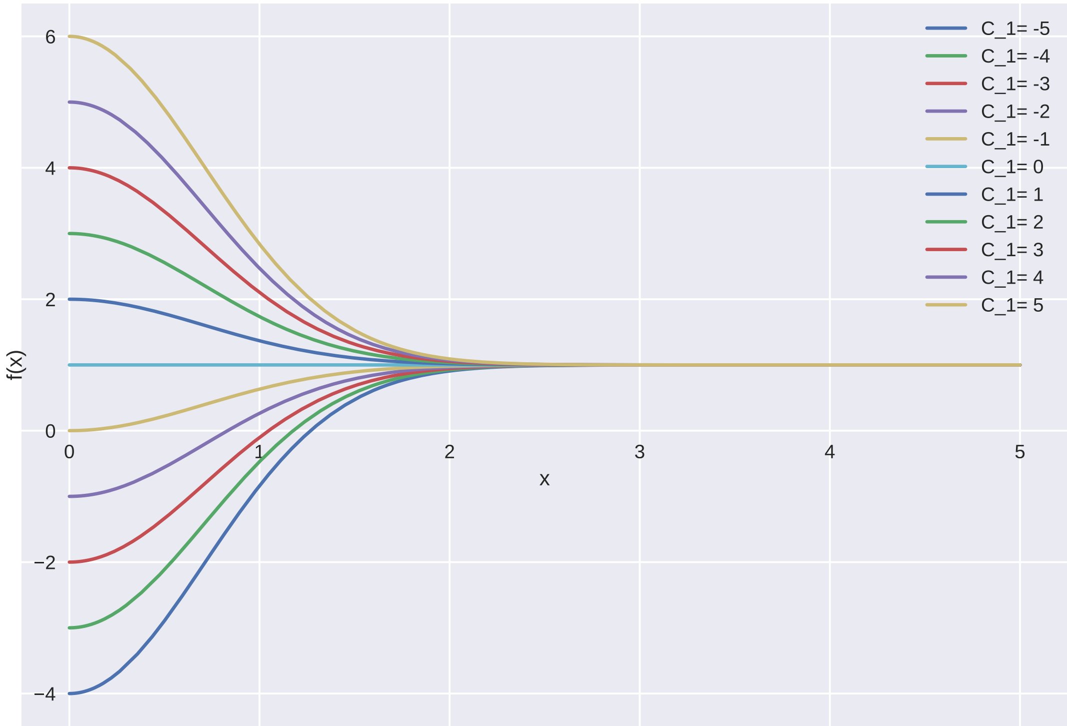

Figura 9: $y = \frac{e^{t^2}+C_1}{e^{t^2}} = 1 + C_1 e^{-t^2}$

Usaremos la librería `scypy` para resolver la ecuación de forma numérica:

```
from sympy.plotting.plot import List2DSeries
def fun(t,y):
    return 2*t - 2*t*y

sol = solve_ivp(fun, (0,5), [2])

p8.append(List2DSeries(sol.t,sol.y[0]))
p8.save('p10.pdf')
```

El siguiente gráfico 10 compara la solución exacta (azul) con la solución numérica (verde).

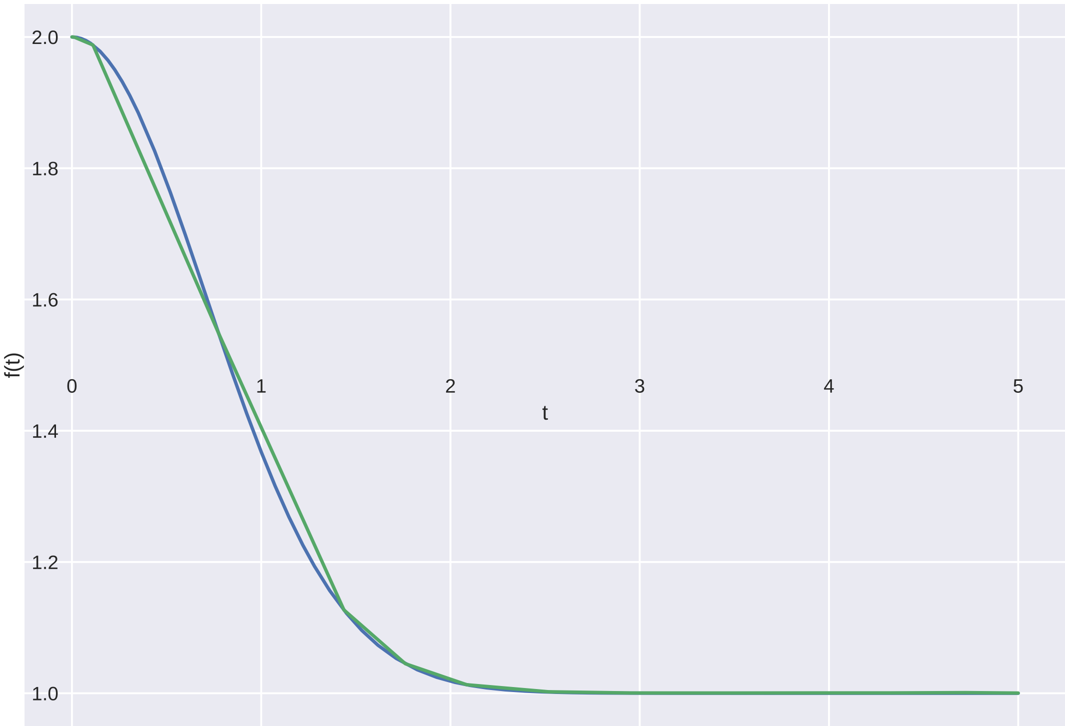

Figura 10:

La función `solve_ivp` usa por defecto el método de Runge-Kutta de orden 5, directamente de la documentación:

> Integration method to use:
>
> 'RK45' (default): Explicit Runge-Kutta method of order 5(4) [1]. The error is controlled assuming accuracy of the fourth-order method, but steps are taken using the fifth-order accurate formula (local extrapolation is done). A quartic interpolation polynomial is used for the dense output 2. Can be applied in the complex domain.

En cuanto al error, la función `solve_ivp` utiliza dos parámetros: `atol,rtol` que sol las tolerancias absolutas y relativas, respectivamente. Sus valores son `rtol=1e-3` y `atol=1e-6` y el error es siempre menor `atol + rtol * abs(y)`.

4

De acuerdo con el método de euler mejorado,

$$y_1 = y_0 + \frac{h}{2}(f(t_0, y_0) + f(t_1, y_0 + hf(t_0, y_0))).$$

Con $h = 0.1$ se tiene $t_0 = 0$, $t_1 = 0.1$ y $y_0 = 1$. Además $f(t_0, y_0) = 0$. La solución es entonces:

$$y_1 = 1 + \frac{0.1}{2} f(0.1, 1),$$

y al colocar esa expresión en la calculadora:

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

print(1 + (0.1/2) * f(0.1, 1))
```

```
## 1.009014802470346
```

mientras que la solución exacta es $y(1) = 1 + 1^2 = 2$.

Con $h = 0.025$ se tiene $t_0 = 0$, $t_1 = 0.025$ y $y_0 = 1$. Además $f(t_0, y_0) = 0$. La solución es entonces:

$$y_1 = 1 + \frac{0.025}{2} f(0.025, 1),$$

y al colocar esa expresión en la calculadora:

```
print(1 + (0.025/2) * f(0.025, 1))
```

```
## 1.00060938963624
```

mientras que la solución exacta es $y(1) = 1 + 1^2 = 2$.