entrega2

March 22, 2023

[1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

1 2. Introducción

1.1 a)

Sea $\tilde{x} := x\sqrt{k_1}$ and $\tilde{y} := y\sqrt{k_2}$, entonces

$$h(x,y) := \begin{pmatrix} \sqrt{k_1} & 0 \\ 0 & \sqrt{k_2} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{1}$$

Sea $\tilde{u}(x,y,t) := u(h(x,y),t) = u(\tilde{x},\tilde{y},t)$

Por la regla de la cadena obtenemos:

$$\tilde{u}_t(x,y,t) = u_t(\tilde{x},\tilde{y},t) \tag{2}$$

$$\tilde{u}_x(x,y,t) = \frac{\partial \tilde{x}}{\partial x} u_x(\tilde{x},\tilde{y},t) + \frac{\partial \tilde{y}}{\partial x} u_y(\tilde{x},\tilde{y},t) = \sqrt{k_1} u_x(\tilde{x},\tilde{y},t) \tag{3}$$

$$\tilde{u}_{xx}(x,y,t) = \frac{\partial}{\partial x}\sqrt{k_1} u_x(\tilde{x},\tilde{y},t) = k_1 u_{xx}(\tilde{x},\tilde{y},t) \tag{4}$$

Análogamente para y:

$$\tilde{u}_{yy}(x,y,t) = k_2 u_{yy}(\tilde{x},\tilde{y},t) \tag{5}$$

Veamos que está función es solución de (1):

$$\tilde{u}_t(x,y,t) - \Delta u(x,y,t) = f(x,y,t) \Leftrightarrow u_t(\tilde{x},\tilde{y},t) - k_1 u_{xx}(\tilde{x},\tilde{y},t) - k_2 u_{yy}(\tilde{x},\tilde{y},t) = f(x,y,t) \tag{6}$$

Ahora el nuevo dominio es:

$$\Omega' := \{(x,y) : (x\sqrt{k_1}, y\sqrt{k_2}) \in \Omega\} = \{(x/\sqrt{k_1}, y/\sqrt{k_2}) : (x,y) \in \Omega\} \tag{7}$$

Por tanto, comprobamos que escalamos el nuevo dominio con $(1/\sqrt{k_1}, 1/\sqrt{k_2})$ para cada uno de los axis.

1.2 b)

Sea $\tilde{u}(x,t) := u(x\sqrt{k},t)$, donde sabemos la solución:

$$\tilde{u}(x,t) = \frac{1}{\sqrt{4\pi t}} \int_{\mathbb{R}} e^{-\frac{(x-y)^2}{4t}} \tilde{u}_0(y) dy \tag{8}$$

Ahora, sabemos que $\tilde{u}(x/\sqrt{k},t) = u(x,t)$. Entonces

$$u(x,t) = \frac{1}{\sqrt{4\pi t}} \int_{\mathbb{R}} e^{-\frac{(x/\sqrt{k}-y)^2}{4t}} \tilde{u}_0(y) dy \qquad (a) \tag{9}$$

Con el siguiente cambio de $u := \sqrt{k}y$:

$$u(x,t) = \frac{1}{\sqrt{4\pi tk}} \int_{\mathbb{R}} e^{-\frac{(x-y)^2}{4tk}} \tilde{u}_0(u/\sqrt{k}) du = \frac{1}{\sqrt{4\pi tk}} \int_{\mathbb{R}} e^{-\frac{(x-y)^2}{4tk}} u_0(u) du \tag{10}$$

Ahora calculamos la derivada con respecto a k sobre la ecuación (a):

$$\frac{\partial}{\partial k} u_k(x,t) = \frac{1}{\sqrt{4\pi t}} \int_{\mathbb{R}} (-xk^{-3/2}/2) - \frac{2(x/\sqrt{k}-y)}{4t} e^{-\frac{(x/\sqrt{k}-y)^2}{4t}} \tilde{u}_0(y) dy \tag{11}$$

Observamos que el interior de la integral es positivo, entonces la derivada es positiva

2 3. Principios de Comparación

2.1 c)

La primera desigualdad la tenemos inmediata de aplicar el teorema del principio de comparación con la solución $u \equiv 0$. Entonces $0 \leq u_D$ y $0 \leq u_N$.

Ahora, vemos para la condición de Neumann:

$$\frac{\partial u_N}{\partial n} = 0 \Rightarrow u_N \equiv \text{cte} \geq 0 \tag{12}$$

Entonces tenemos por el teorema del principio de comparación que $u_D \leq u_N$

Una vez hemos demostrado que $u_D \leq u_N$. Podemos demostrar fácilmente que u_R se encuentar entre ambos. Cómo $g^N \equiv 0$ entonces tenemos tres opciones: 1) Si $\alpha, \beta > 0$: entonces se cumple la condición de Neumann y Dirichlet. Por tanto $u_R \equiv u_D$ y $u_R \equiv u_D$ 2) Si $\alpha = 0, \beta > 0$: entonces tenemos la misma condición que Dirichlet. Por tanto $u_R \equiv u_D$. 3) Si $\alpha > 0, \beta = 0$: entonces tenemos la misma condición que Neummann. Por tanto $u_R \equiv u_N$.

De cualquier modo, se cumple la desigualdad

Una interpretación física de esta desigualdad podría venir de que la condición de Dirichlet pone por frontera una pared a "temperatura 0", mientras que Neumann pone un aislamiento (no infiltración de frío por la paredes). Por tanto si $f \geq 0$, entonces siempre se va a cumplir las condiciones.

En cambio, para el apartado siguiente si tenemos una condición frontera positiva para Dirichlet, estamos introduciendo calor por los bordes y entonces la solución de Dirichlet será mayor al principio (como podemos ver en el siguiente apartado).

2.2 d)

Por ejemplo si cogemos $g^D = 1, g^N = 0$ y además $f = 0$ entonces $u_D \xrightarrow{t\to\infty} 1$ mientras que $u_N \equiv 0$

3 4. La masa total

3.1 e)

Integramos la ecuación respecto de x:

$$\underbrace{\int u_t(x,t)dx}_{(a)} - \underbrace{\int u_{xx}(x,t)dx}_{(b)} = \underbrace{\int f(x,t)dx}_{(c)} \tag{13}$$

$$M'(t) = \int_\Omega \frac{\partial}{\partial t} u(x,t)dx = \underbrace{\int_\Omega \Delta u(x,t)dx}_{(a)} + \underbrace{\int_\Omega f(x,t)dx}_{(b)} \tag{14}$$

$$(a) =_{Th.Div} \int_{\partial\Omega} \nabla u \cdot n = \int_{\partial\Omega} \frac{\partial u}{\partial n} = 0 \tag{15}$$

$$(b) = \begin{cases} \int_0^1 (1+\cos t)x(1-x) = \left[(1+\cos t)(x^2/2 - x^3/3)\right]_0^1 = (1+\cos t)/6 & \text{If } t < \pi \\ 0 & \text{If } t > \pi \end{cases} \tag{16}$$

Ahora integramos M:

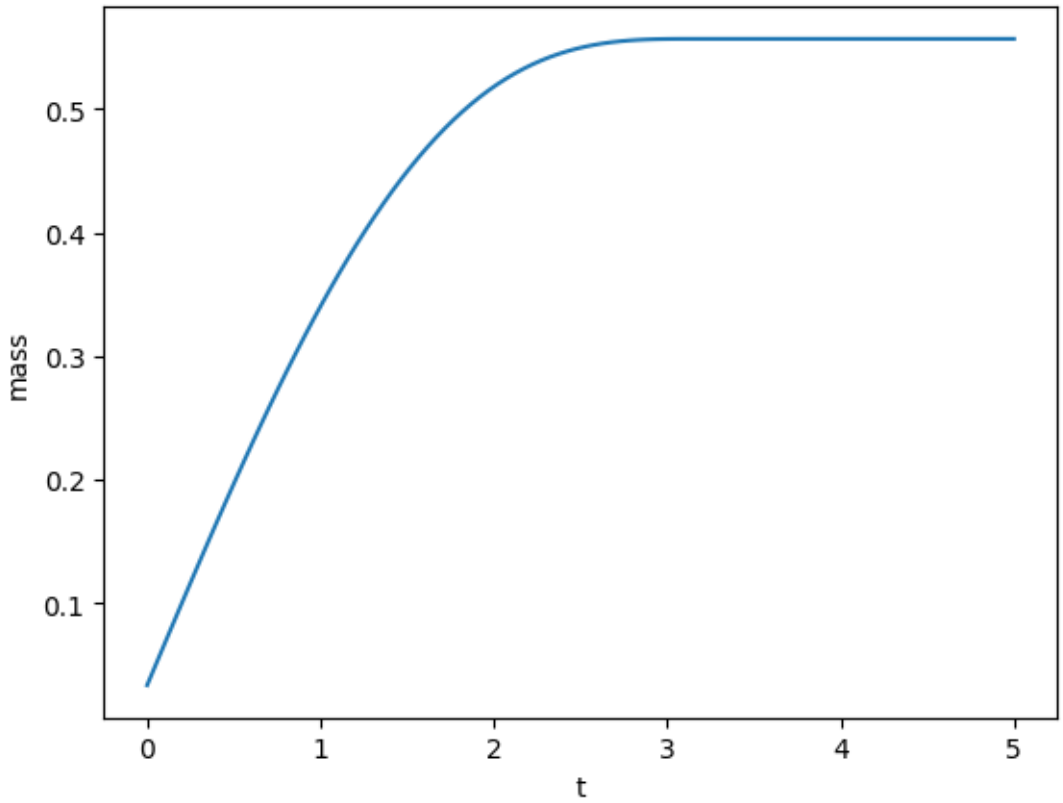
$$M(t) = M(0) + \int_0^t M'(t) = \underbrace{\int_0^1 x^2(1-x)^2 dx}_{(1)} + \underbrace{\int_0^{\min\{\pi,t\}} M'(t)dt}_{(2)} \tag{17}$$

$$(1) = x^3/3 + x^5/5 - 2x^4/4]_0^1 = 1/3 + 1/5 - 1/2 = 1/30 \tag{18}$$

$$(2) = \left[(t+\sin t)/6\right]_0^{\min\{\pi,t\}} = \begin{cases} (t+\sin t)/6 & \text{If } t < \pi \\ \pi/6 & \text{If } t > \pi \end{cases} \tag{19}$$

[2]:
```
mass_t = lambda x: 1/30 + np.minimum((x+np.sin(x))/6, np.pi/6)
t_m = np.linspace(0, 5, 100)
```

[3]:
```
plt.plot(t_m, mass_t(t_m))
plt.xlabel('t')
plt.ylabel('mass')
plt.show()
```



4 5. Planteamiento numérico

4.1 f)

$$\frac{u_i^{j+1} - u_i^j}{\Delta t} = \theta \frac{u_{i+1}^{j+1} - 2u_i^{j+1} + u_{i-1}^{j+1}}{\Delta x^2} + (1-\theta)\frac{u_{i+1}^{j} - 2u_i^{j} + u_{i-1}^{j}}{\Delta x^2} + \theta f_i^{j+1} + (1-\theta) f_i^j \tag{20}$$

Si $\theta = 0$ tenemos el método explícito (Forward), si $\theta = 1$ tenemos el método implicito (Backward). Ambos son métodos de orden $\mathcal{O}(\Delta t, \Delta x^2)$. Tienen de ventaja que hacen falta menos operaciones y para el caso explícito, no haría falta resolver ninguna matriz.

Si $\theta = 1/2$ tenemos un método de orden $\mathcal{O}(\Delta t^2, \Delta x^2)$, por tanto, será ventajoso utilizar este último si nos enfocamos en disminuir el error.

4.2 g)

$$Au = b \Leftrightarrow \begin{pmatrix} (1+2\theta\lambda) & -\lambda\theta & 0 & \dots & 0 \\ -\lambda\theta & (1+2\lambda\theta) & -\lambda\theta & \dots & \vdots \\ 0 & \lambda\theta & \ddots & \ddots & \vdots \\ \vdots & & \ddots & (1+2\theta\lambda) & -\lambda\theta \\ 0 & \dots & & \lambda\theta & (1+2\theta\lambda) \end{pmatrix} \begin{pmatrix} u_1^{j+1} \\ u_2^{j+1} \\ \vdots \\ u_{N-2}^{j+1} \\ u_{N-1}^{j+1} \end{pmatrix} = \begin{pmatrix} b_1^j \\ b_2^j \\ \vdots \\ b_{N-2}^j \\ b_{N-1}^j \end{pmatrix} \tag{21}$$

Ahora, para el nuevo sistema con las condiciones de Neumann se incorpora dos condiciones al sistema para satisfacer la siguiente igualdad:

$$\frac{\partial u}{\partial x}(0,t) = \frac{\partial u}{\partial x}(1,t) = 0 \tag{22}$$

Las escribiriamos así la discretización de las condiciones frontera con diferencias centradas:

$$\begin{cases} u_1^{j+1} - u_{-1}^{j+1} = 0 \\ u_{N+1}^{j+1} - u_{N-1}^{j+1} = 0 \end{cases} \Rightarrow \begin{cases} u_{-1}^{j+1} = u_1^{j+1} \\ u_{N+1}^{j+1} = u_{N-1}^{j+1} \end{cases} \tag{23}$$

Vamos a comprobar en los bordes cómo quedaría nuestro esquema:

$$(1+2\theta\lambda)u_i^{j+1} - \lambda\theta(u_{i+1}^{j+1} + u_{i-1}^{j+1}) = \lambda(1-\theta)(u_{i+1}^{j} + u_{i-1}^{j}) + u_i^j(1-2\theta(1-\theta)) + \Delta t(\theta f_i^{j+1} + (1-\theta)f_i^j) \tag{24}$$

Ahora sustituimos para $i = \{0, N\}$

Para $i = 0$:

$$(1+2\theta\lambda)u_0^{j+1} - 2\lambda\theta u_1^{j+1} = 2\lambda(1-\theta)u_1^j + u_0^j(1-2\theta(1-\theta)) + \Delta t(\theta f_0^{j+1} + (1-\theta)f_0^j) \tag{25}$$

Entonces tenemos los siguentes valor de

$$b_0^j = 2\lambda(1-\theta)u_1^j + u_0^j(1-2\theta(1-\theta)) + \Delta t(\theta f_0^{j+1} + (1-\theta)f_0^j) \tag{26}$$

Seguimos un proceso análogo para $i = N$

Entonces el esquema matricial nos quedaría así:

$$Au = b \Leftrightarrow \begin{pmatrix} 1+2\theta\lambda & -2\lambda\theta & 0 & ... & & 0 & 0 \\ -\lambda\theta & (1+2\theta\lambda) & -\lambda\theta & 0 & ... & 0 & 0 \\ 0 & -\lambda\theta & (1+2\lambda\theta) & -\lambda\theta & ... & \vdots & \vdots \\ \vdots & 0 & -\lambda\theta & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & (1+2\theta\lambda) & -\lambda\theta & \vdots \\ 0 & 0 & ... & & -\lambda\theta & (1+2\theta\lambda) & \vdots \\ 0 & \vdots & 0 & ... & 0 & -2\lambda\theta & 1+2\theta\lambda \end{pmatrix} \begin{pmatrix} u_0^{j+1} \\ u_1^{j+1} \\ u_2^{j+1} \\ \vdots \\ u_{N-2}^{j+1} \\ u_{N-1}^{j+1} \\ u_N^{j+1} \end{pmatrix} = \begin{pmatrix} b_0^j \\ b_1^j \\ b_2^j \\ \vdots \\ b_{N-2}^j \\ b_{N-1}^j \\ b_N^j \end{pmatrix} \tag{27}$$

4.3 h)

Para la condición de Dirichlet es más sencillo, ya que directamente podemos dar el valor 0 a u_0^{j+1} y u_N^{j+1}

Entonces el esquema matricial nos quedaría tal que así:

$$Au = b \Leftrightarrow \begin{pmatrix} 1 & 0 & 0 & \dots & & 0 & 0 \\ 0 & (1+2\theta\lambda) & -\lambda\theta & 0 & \dots & 0 & 0 \\ 0 & -\lambda\theta & (1+2\lambda\theta) & -\lambda\theta & \dots & \vdots & \vdots \\ \vdots & 0 & \lambda\theta & \ddots & \ddots & \vdots & \vdots \\ \vdots & \vdots & & \ddots & (1+2\theta\lambda) & -\lambda\theta & \vdots \\ 0 & 0 & \dots & & \lambda\theta & (1+2\theta\lambda) & \vdots \\ 0 & \vdots & 0 & \dots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u_0^{j+1} \\ u_1^{j+1} \\ u_2^{j+1} \\ \vdots \\ u_{N-2}^{j+1} \\ u_{N-1}^{j+1} \\ u_N^{j+1} \end{pmatrix} = \begin{pmatrix} 0 \\ b_1^j \\ b_2^j \\ \vdots \\ b_{N-2}^j \\ b_{N-1}^j \\ 0 \end{pmatrix} \tag{28}$$

5 6. Resultados Numéricos

5.1 i)

[4]:
```
N = 30
M = 4500
T = 5
theta = 0.5

dx = 1/N
dt = T/M

lambd = dt/(dx*dx)

x = np.linspace(0,1,N+1)
t = np.linspace(0,T,M+1)
X,T = np.meshgrid(x,t)
```

[5]:
```
# Comprobamos que el método es estable y convergente
assert lambd*(1-theta)<=1/2
```

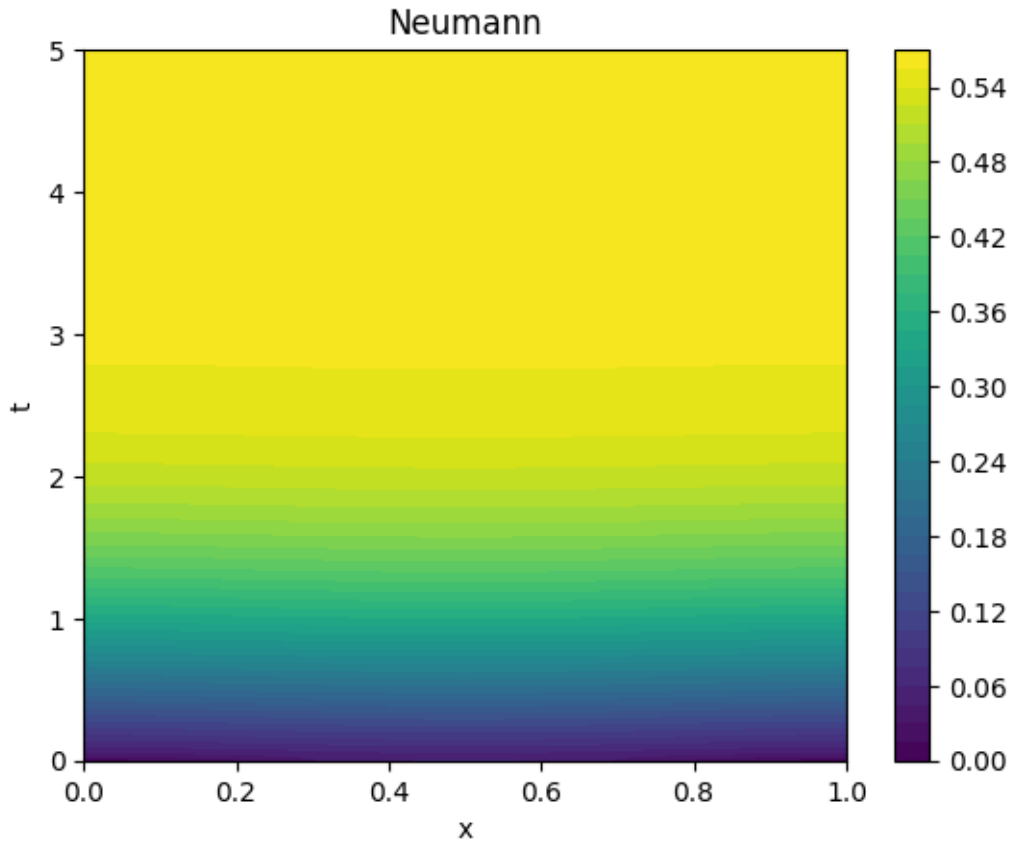
[6]:
```
f = lambda t,x: np.where(t<np.pi, (1+np.cos(t))*x*(1-x) ,0)
u_0 = lambda x: x**2*(1-x)**2
```

[7]:
```
def numerical_neumann(u_0, theta, lambd):
    # initialize coeff matrix with the values specified in the scheme
    A = np.diag([1+2*theta*lambd]*(N+1),0) + np.diag([-theta*lambd]*(N),1) + np.
 ↪diag([-theta*lambd]*(N),-1)
    A[0,1]=-2*lambd*theta; A[-1,-2]=-2*lambd*theta;
    A_inv = np.linalg.inv(A)

    # inialize solution with initial values
    u = np.zeros((M+1,N+1))
    u[0] = u_0(x)

    for j in range(len(t)-1):
        b = np.zeros(N+1)
```

```
        for i in range(1,N):
            b[i]=lambd*(1-theta)*(u[j][i+1]+u[j][i-1]) +␣
↪u[j][i]*(1-2*lambd*(1-theta)) + dt*(theta*f(t[j+1],x[i]) +␣
↪(1-theta)*f(t[j],x[i]))

        b[0] = 2*lambd*(1-theta)*(u[j][1])   + u[j][0]*(1-2*lambd*(1-theta)) +␣
↪dt*(theta*f(t[j+1],x[0]) + (1-theta)*f(t[j],x[0]))
        b[N] = 2*lambd*(1-theta)*(u[j][N-1]) + u[j][N]*(1-2*lambd*(1-theta)) +␣
↪dt*(theta*f(t[j+1],x[N]) + (1-theta)*f(t[j],x[N]))

        u[j+1] = np.dot(A_inv,b)

    return u
```

[8]:
```
u_N = numerical_neumann(u_0, theta, lambd)
```

[9]:
```
plt.contourf(X,T,u_N, levels=40)
plt.colorbar()
plt.title('Neumann')
plt.xlabel('x')
plt.ylabel('t')
plt.show()
```



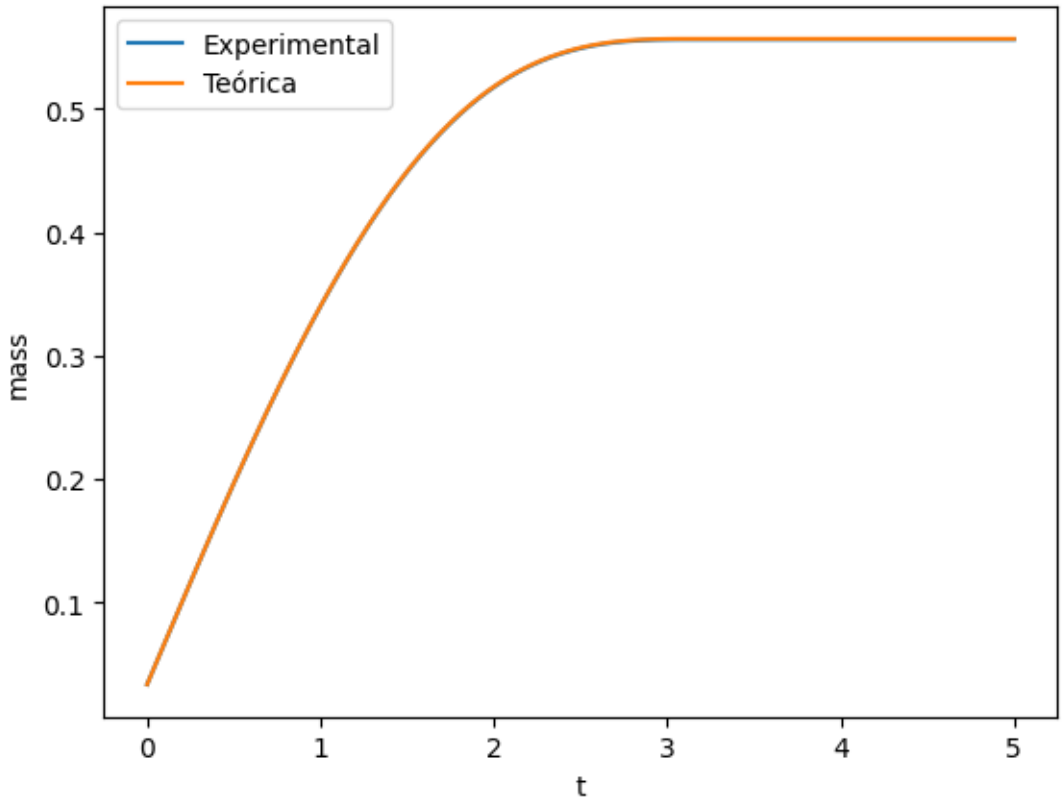
5.2 j)

We integrate the discretization of the solution u_N:

$$M(t) = \int_0^1 u_N(x,t)dx \tag{29}$$

[10]:
```
mass_e = np.trapz(u_N, dx=dx, axis=1) # regla del trapecio
```

[11]:
```
plt.plot(t,mass_e, label='Experimental')
plt.plot(t_m,mass_t(t_m), label='Teórica')
plt.legend()
plt.xlabel('t')
plt.ylabel('mass')
plt.show()
```

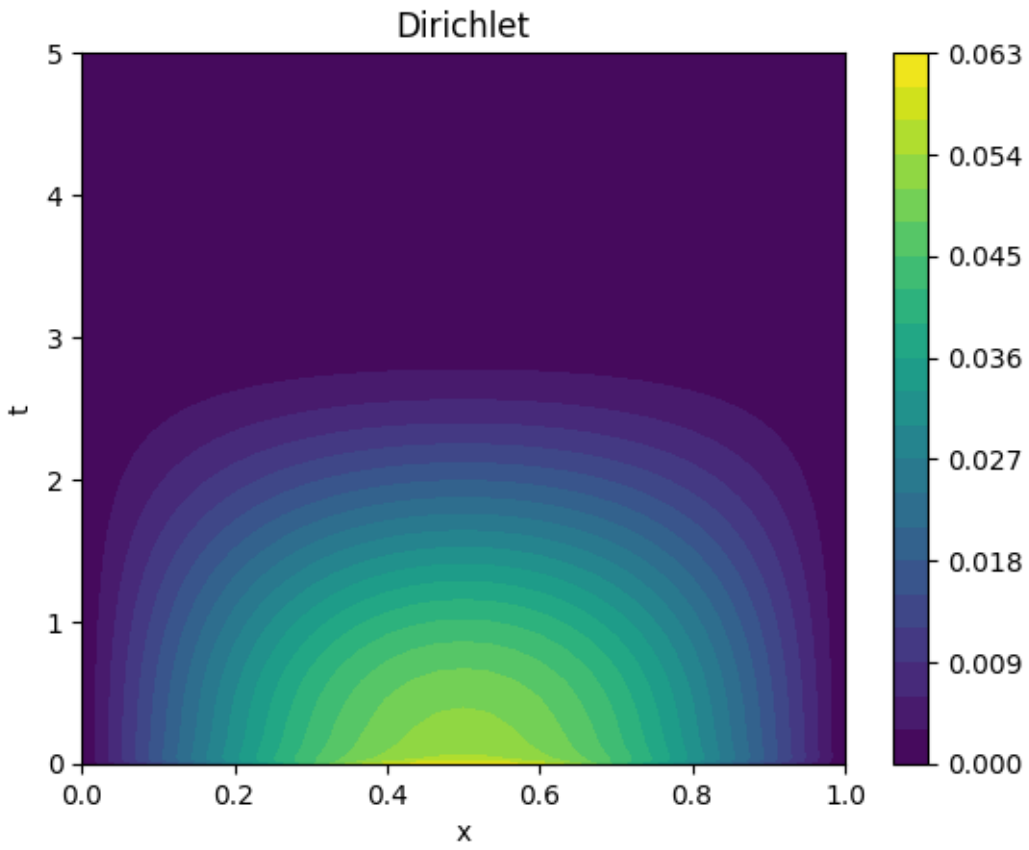


5.3 k)

[12]:
```
def numerical_dirichlet(u_0, theta, lambd):
    # create coeff matrix and calculate the inverse
    A = np.diag([1+2*theta*lambd]*(N-1),0) + np.diag([-theta*lambd]*(N-2),1) +␣
 ↪np.diag([-theta*lambd]*(N-2),-1)
    A_inv = np.linalg.inv(A)

    # initialize solution with initial values
    u = np.zeros((M+1,N+1))
    u[0] = u_0(x)

    for j in range(len(t)-1):
        b = np.zeros(N-1)
        for i in range(1,len(x)-1):
            b[i-1]=lambd*(1-theta)*(u[j][i+1]+u[j][i-1]) +␣
 ↪u[j][i]*(1-2*lambd*(1-theta)) + dt*(theta*f(t[j+1],x[i]) +␣
 ↪(1-theta)*f(t[j],x[i]))

        u[j+1][1:-1] = np.dot(A_inv,b)

    return u
```

[13]:
```
u_D = numerical_dirichlet(u_0, theta, lambd)
```

[14]:
```
plt.contourf(X,T,u_D, levels=20)
plt.title('Dirichlet')
plt.xlabel('x')
plt.ylabel('t')
plt.colorbar()
plt.show()
```



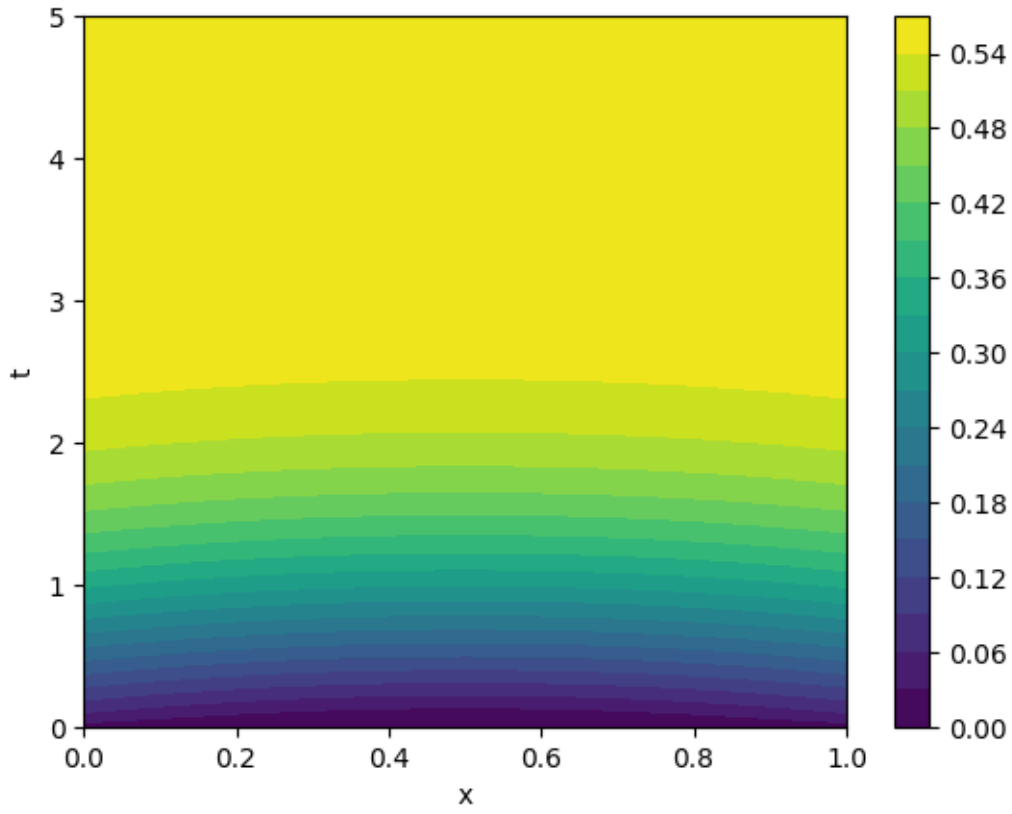
Podemos ver que $u_N \geq u_D$ para nuestra solución numérica como se demostró en el ejercició teórico.

[15]:
```
np.min(u_N-u_D)
```

[15]: 0.0

Y también vemos ploteado la diferencia $u_N - u_D$, como se puede ver, hay mayor diferecia en los bordes:

[16]:
```
plt.contourf(X,T,u_N-u_D, levels=20)
plt.colorbar()
plt.xlabel('x')
plt.ylabel('t')
plt.show()
```



Cambiamos las condiciones iniciales para valores negativos de x_0 (en este caso -1)

[17]:
```
v_0 = lambda x: -1
```

[18]:
```
v_N = numerical_neumann(v_0, theta, lambd)
v_D = numerical_dirichlet(v_0, theta, lambd)
```

Comprobamos que $v_N \not\geq v_D$

[19]:
```
np.min(v_N-v_D)
```

[19]:
```
-0.9999597024554147
```

Una interpretación física sería que en este caso al ser la solución menor que 0, las condiciones frontera (para Dirichlet) suman energía al sistema en vez de restar como en el caso anterior.

[ ]: