

Aula 19

Equações diferenciais independentes do tempo: problemas de condições fronteira (espacial): PARTE 2

Transferência de calor

Método da relaxação

A solução satisfaz: (caso $\Delta x = \Delta y = \Delta$)

$$\phi_{i-1,j} + \phi_{i+1,j} + \phi_{i,j-1} + \phi_{i,j+1} - 4\phi_{i,j} = \Delta^2 f_{i,j}$$

Iteração: $\phi_{i,j}^0 = 0, \phi_{i,j}^1, \dots, \phi_{i,j}^n, \dots$

Dada uma estimativa do campo ϕ , na iteração n existe um erro (resíduo R):

$$\phi_{i-1,j}^n + \phi_{i+1,j}^n + \phi_{i,j-1}^n + \phi_{i,j+1}^n - 4\phi_{i,j}^n - \Delta^2 f_{i,j} = R_{i,j}$$

Se se corrigir:

$$\phi_{i,j}^{n+1} = \phi_{i,j}^n + \frac{R_{i,j}}{4}$$

O erro será anulado (mas só nesse ponto!)

$$\phi_{i-1,j}^n + \phi_{i+1,j}^n + \phi_{i,j-1}^n + \phi_{i,j+1}^n - 4 \left(\phi_{i,j}^n + \frac{R_{i,j}}{4} \right) - \Delta^2 f_{i,j} = 0$$

Sobre-relaxação **simultânea**

Só se mantém um array de ϕ . Faz-se:

$$\phi_{i,j} = \phi_{i,j} + \beta \frac{R_{i,j}}{4}$$

i.e., à medida que se altera um ponto de grelha o novo valor já é utilizado no cálculo do resíduo dos pontos adjacentes.

$$1 \leq \beta < 2$$

β é o parâmetro de **sobre-relaxação**. Pode mostrar-se que o método converge mais rapidamente com:

$$\beta_{opt} = 2 - \pi\sqrt{2} \left(\frac{1}{M^2} + \frac{1}{N^2} \right)^{1/2}$$

Python: resolve $\nabla^2 V = f$

```
def poisson(f,V0,X,Y,maxiter,maxres,beta0):
    [M,N]=f.shape; #determina a dimensão das matrizes
    if f.shape!=X.shape or X.shape!=Y.shape \
        or V0.shape!=X.shape:
        print('Error in matrix size')
        return V0,0,1e30,beta0
    dx=X[2,1]-X[1,1];dy=Y[1,2]-Y[1,1]
    if dx!=dy: #Admite-se espaçamento regular
        print('Error in dx,dy')
        return V0,0,1e30,beta0
    delta=dx
    if (beta0<1) or (beta0>2): #parametro de sobre-relaxação
        beta=2-np.pi*np.sqrt(2.)*np.sqrt(1./M**2+1./N**2)
    else:
        beta=beta0
    V=V0 #inicializa a matriz solução
    iter=0
    resid=2*maxres #garante a primeira iteração
    pngs=[]
```

python

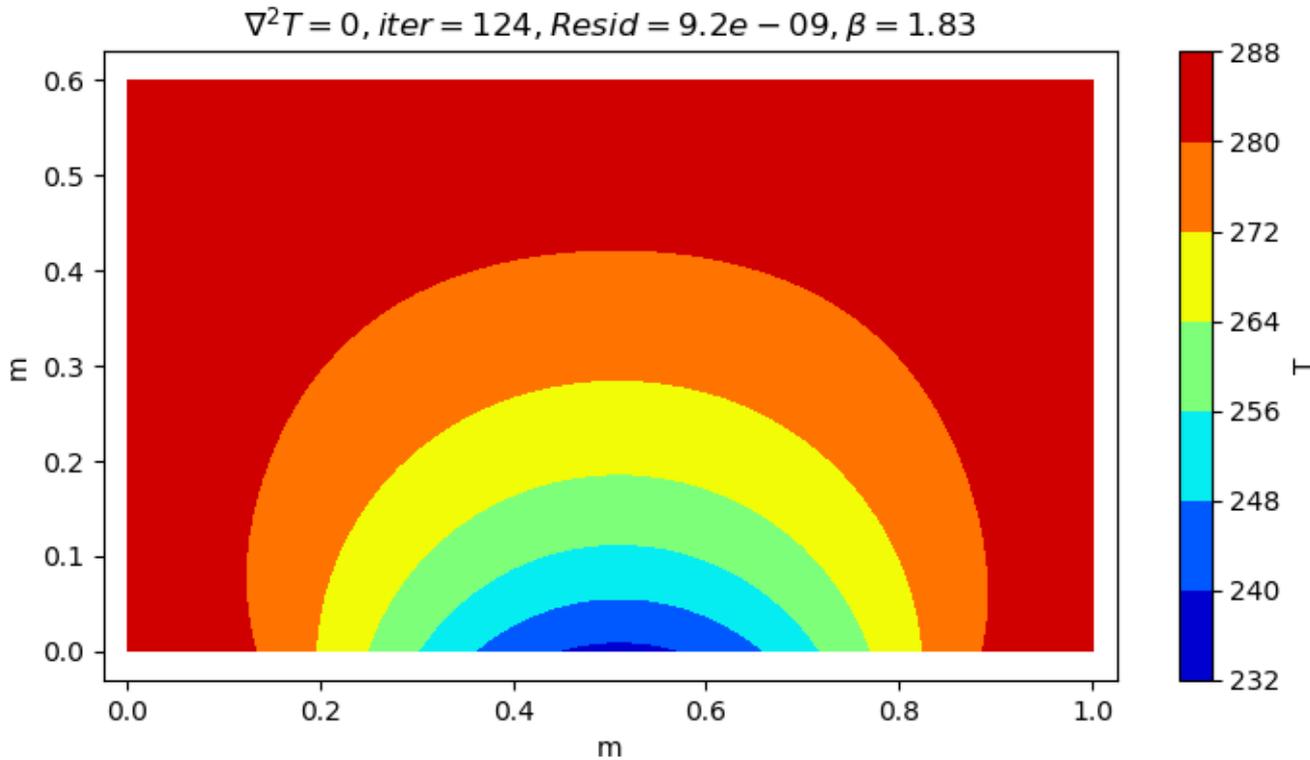
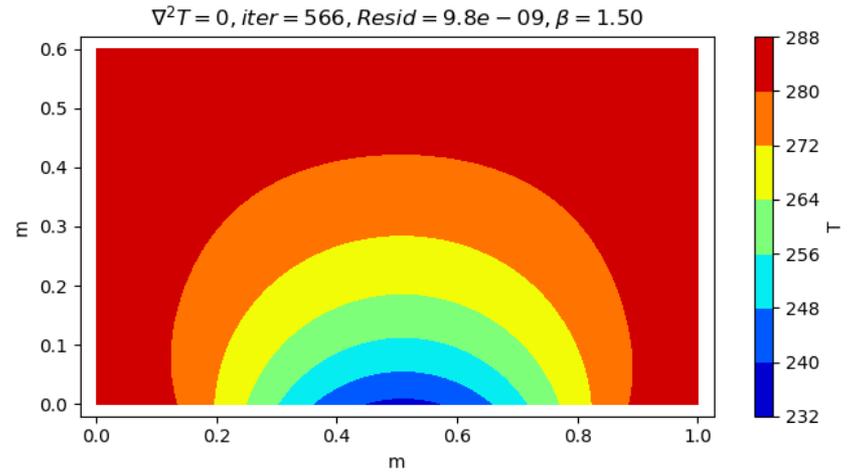
```
while resid>maxres and iter<maxiter: #iterações
iter=iter+1
resid=0; vmax=0
for i in range(1,M-1): #vai de 1 a M-2
    for j in range(1,N-1):
        R=V[i,j-1]+V[i,j+1]+V[i-1,j]+\
            V[i+1,j]-4*V[i,j]-delta**2*f[i,j]
        V[i,j]=V[i,j]+beta*R/4;
        resid=max(resid,abs(R))
#condições fronteira: entram aqui!
#se não se fizer nada V na fronteira fica sempre o mesmo
#o que é uma condição fronteira possível
vmax=np.max(np.abs(V))
resid=resid/vmax #residuo relativo
return V,iter,resid,beta
```

O Código anterior (tal como está...)

Resolve a equação $\nabla^2 V = f$ com a condição fronteira imposta (previamente) nos pontos de fronteira da matriz V

Main $\nabla^2 T = 0$

```
Lx=1.;M=51;N=31;delta=Lx/(M-1) #deltax=deltay=delta
X=np.zeros((M,N));Y=np.zeros((M,N));ro=np.zeros((M,N))
f=np.zeros((M,N))
for i in range(M):
    for j in range(N):
        X[i,j]=i*delta; Y[i,j]=j*delta
maxiter=3000;maxres=1.e-8 # erro relativo
V0=288.*np.ones((M,N)) #temperatura inicial inclui cond front
Dirichlet
for ix in range(M):
    V0[ix,0]=288-50*np.sin(ix*np.pi/M)**2
V,niter,res,beta=poisson(f,V0,X,Y,maxiter,maxres,0)
plt.figure()
map=plt.contourf(X,Y,V,cmap='jet') # gráfico de isolinhas
plt.colorbar(map,label='T')
plt.xlabel('m');plt.ylabel('m');plt.axis('equal');
plt.title(r'$\nabla^2 T=0,iter=%6i,Resid=%5.1e,\beta=%5.2f $' \
        % (niter,res,beta))
```



Com sobre-
relaxação
óptima

Vamos repensar a condição fronteira

#condições fronteira: entram aqui!

#se não se fizer nada V na fronteira fica sempre o mesmo

#o que é uma condição fronteira possível

$\left(\frac{\partial T}{\partial n}\right)_{\text{fronteira}} = 0 \Leftrightarrow$ Fronteira isolante (fluxo de calor zero)

```
for ix in range(M):
```

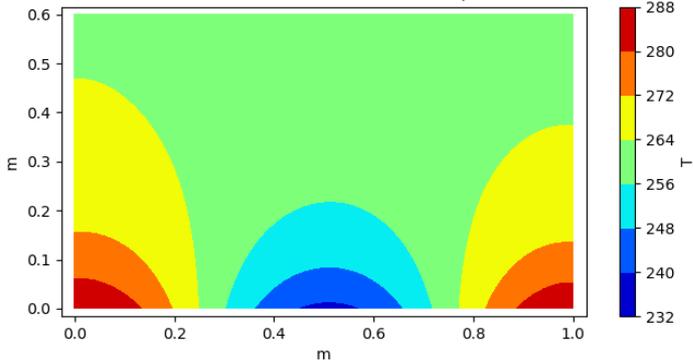
```
    V[ix,N-1]=V[ix,N-2] #topo
```

```
for iy in range(2,N):
```

```
    V[0,iy]=V[1,iy] #esquerda
```

```
    V[M-1,iy]=V[M-2,iy] #direita
```

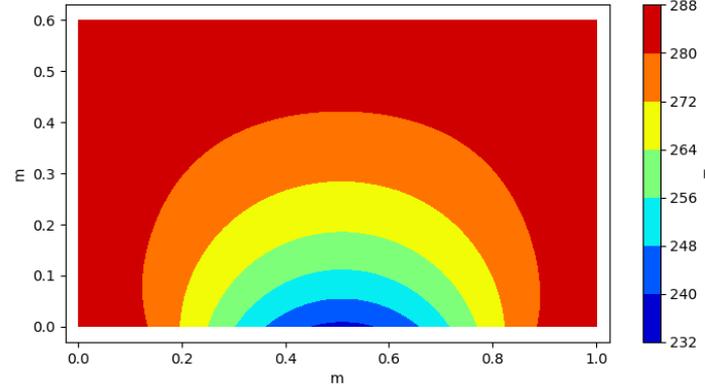
$\nabla^2 T = 0$, iter = 1041, Resid = $1.0e-08$, $\beta = 1.83$



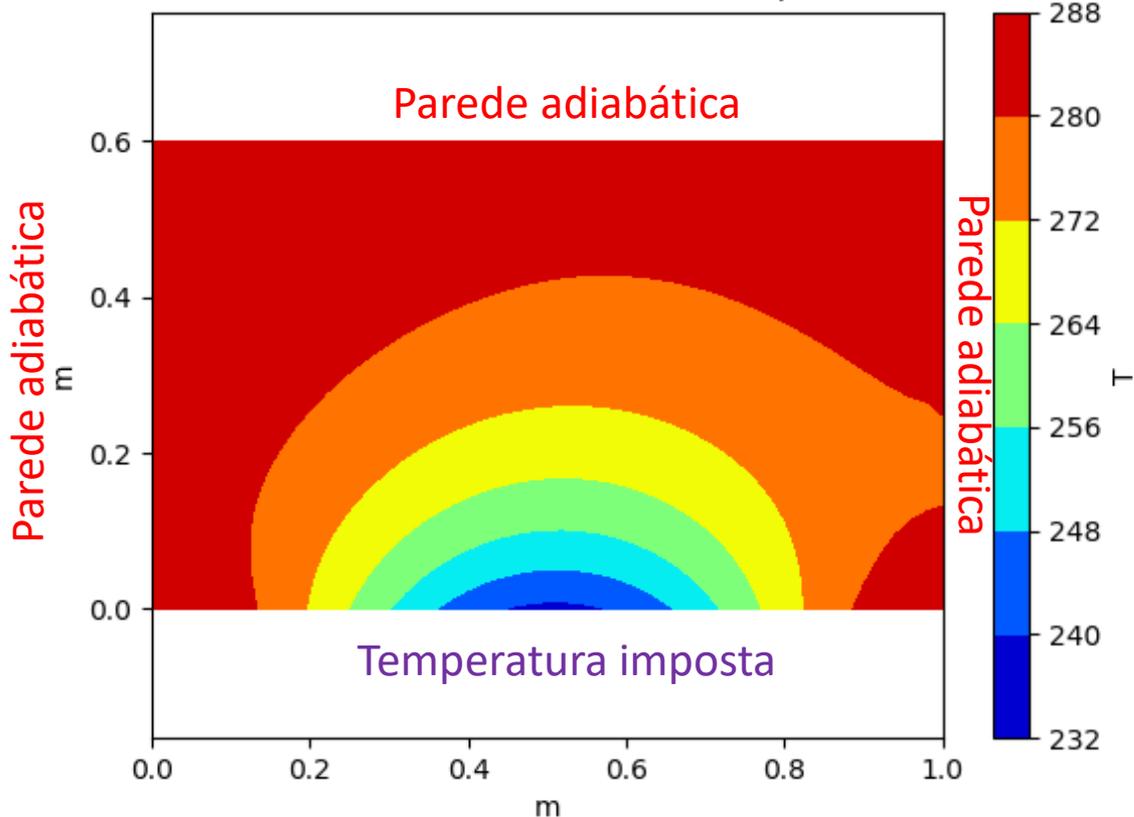
Paredes topo,
esquerda e direita
adiabáticas



$\nabla^2 T = 0$, iter = 124, Resid = $9.2e-09$, $\beta = 1.83$



$\nabla^2 T = 0$, iter = 20, Resid = $6.3e-01$, $\beta = 1.83$



Impõe-se a derivada normal:
Condição de von Neumann
(converge mais lentamente)



Impõe-se a função:
Condição de Dirichlet

Lei de Fourier da condução

Num meio conductor, o fluxo de calor (Wm^{-2}) é proporcional ao gradiente de temperatura, transportando calor em direção às regiões mais frias:

$$\vec{q} = -\chi \nabla T = -\chi \left(\frac{\partial T}{\partial x} \vec{i} + \frac{\partial T}{\partial y} \vec{j} + \frac{\partial T}{\partial z} \vec{k} \right)$$

χ ($Wm^{-1}K^{-1}$) é a **condutividade térmica** do meio, depende da sua composição e da temperatura.

A taxa de variação da temperatura num ponto depende da distribuição do fluxo de calor:

$$\rho c_p \frac{\partial T}{\partial t} = -\nabla \cdot (-\chi \nabla T)$$

ρ é a densidade, c_p é o calor específico do meio.

Na presença de fontes internas de calor...

Incluindo **fontes internas de calor** (reações químicas, decaimento radioativo, ou outras) dadas por \dot{q}_V ($W m^{-3}$), obtém-se a forma mais geral da **lei de Fourier**:

$$\rho c_p \frac{\partial T}{\partial t} = -\nabla \cdot (-\chi \nabla T) + \dot{q}_V$$

Em **equilíbrio** $T = const$, tem-se:

$$-\nabla \cdot (-\chi \nabla T) + \dot{q}_V = 0$$

No caso $\chi = const$ (**material homogêneo com pouco gradiente de temperatura**), fica a **equação de Poisson**:

$$\nabla^2 T = -\frac{1}{\chi} \dot{q}_V$$

Na ausência de fontes internas, temos a **equação de Laplace**:

$$\nabla^2 T = 0$$

poisson.py v2 com condições fronteira opcionais (von Neumann)

```
def poisson(f,V0,X,Y,maxiter,maxres,\ #posicionais (obrigatórias)
          beta0=0.,BxL=[],BxH=[],ByL=[],ByH=[],movie='',passo=1): #kwargs
    [M,N]=f.shape; #determina a dimensão das matrizes
    ...
    V=V0 #inicializa a matriz solução
    iter=0; resid=2*maxres
    while resid>maxres and iter<maxiter: #iterações
        iter=iter+1;resid=0; vmax=0
        for i in range(1,M-1):
            for j in range(1,N-1):
                R=V[i,j-1]+V[i,j+1]+V[i-1,j]+V[i+1,j]-4*V[i,j]-delta**2*f[i,j]
                V[i,j]=V[i,j]+beta*R/4;
                resid=max(resid,abs(R))
    if len(BxL)!=0: #caso contrário: Dirichlet, mantem V
        V[0,:]=V[1,:]+BxL*dx #BxL=(dT/dn) em x=0 (n normal à parede)
    if len(BxH)!=0:
        V[M-1,:]=V[M-2,:]+BxH*dx #BxH=(dT/dn) em x=Lx
    if len(ByL)!=0:
        V[:,0]=V[:,1]+ByL*dy
    if len(ByH)!=0:
        V[:,N-1]=V[:,N-2]+ByH*dy
    vmax=np.max(np.abs(V)); resid=resid/vmax #residuo relative
    return V,iter,resid,beta
```

main

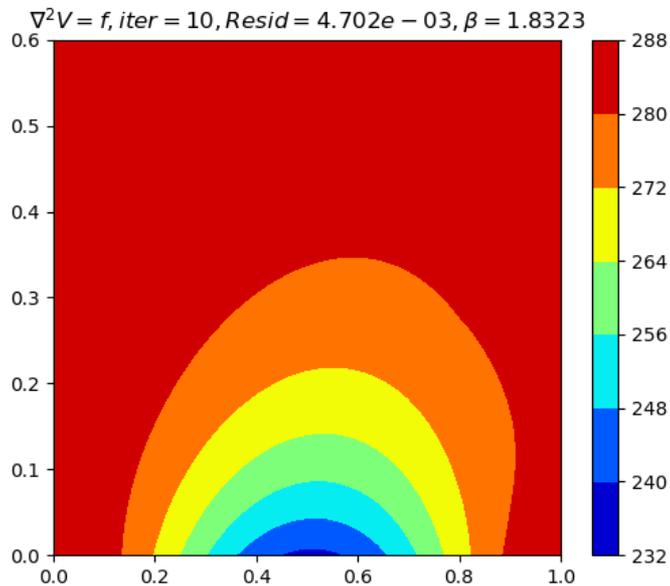
```
import numpy as np;import matplotlib.pyplot as plt
Lx=1.;Ly=1.2;M=51;N=31;delta=Lx/(M-1)
X=np.zeros((M,N));Y=np.zeros((M,N));ro=np.zeros((M,N));f=np.zeros((M,N))
for i in range(M):
    for j in range(N):
        X[i,j]=i*delta; Y[i,j]=j*delta
maxiter=3000;maxres=1.e-8 # erro relativo
V0=288.*np.ones((M,N)) #temperatura inicial inclui cond front Dirichlet
for ix in range(M):
    V0[ix,0]=288-50*np.sin(ix*np.pi/M)**2
BxH=np.zeros((N)); ByH=-np.ones((M))*10 #cond front von Neumann
[V,niter,res,beta]=poisson(f,V0,X,Y,maxiter,maxres,passo=10,\
    movie='T3',BxH=BxH,ByH=ByH)
plt.figure(2)
map=plt.contourf(X,Y,V,cmap='jet') # gráfico de isolinhas
plt.colorbar(map,label='T')
plt.xlabel('m');plt.ylabel('m');plt.axis('equal');
plt.title(r'\nabla^{2} V=f,iter=%6i,Resid=%5.1e,\beta=%5.2f $' % \
    (niter,res,beta))
```

for ix in range(M) :

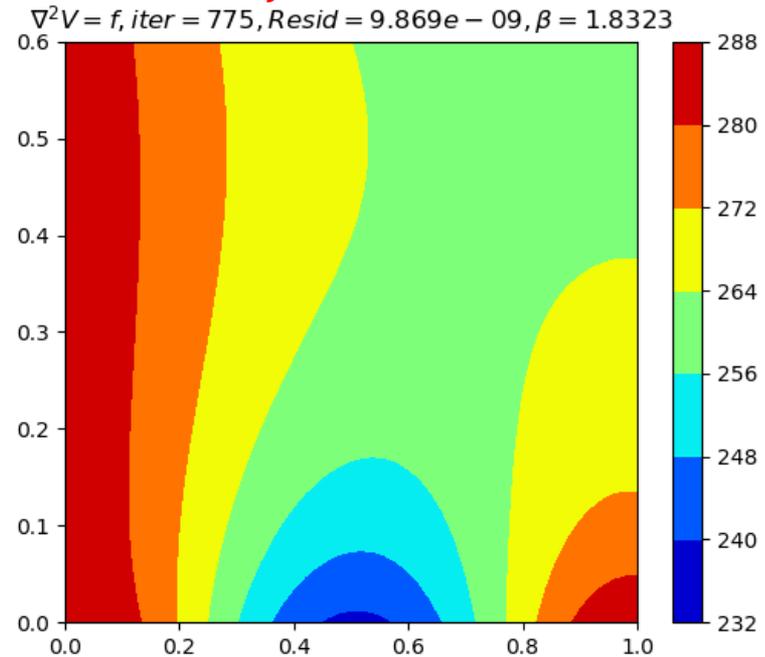
$V0[ix,0]=288-50*\text{np.sin}(ix*\text{np.pi}/M)**2$

$BxH=\text{np.zeros}(N)$; $ByH=-\text{np.ones}(M)*10$

$$\frac{\partial V}{\partial y} = 10 \text{ K/m}$$



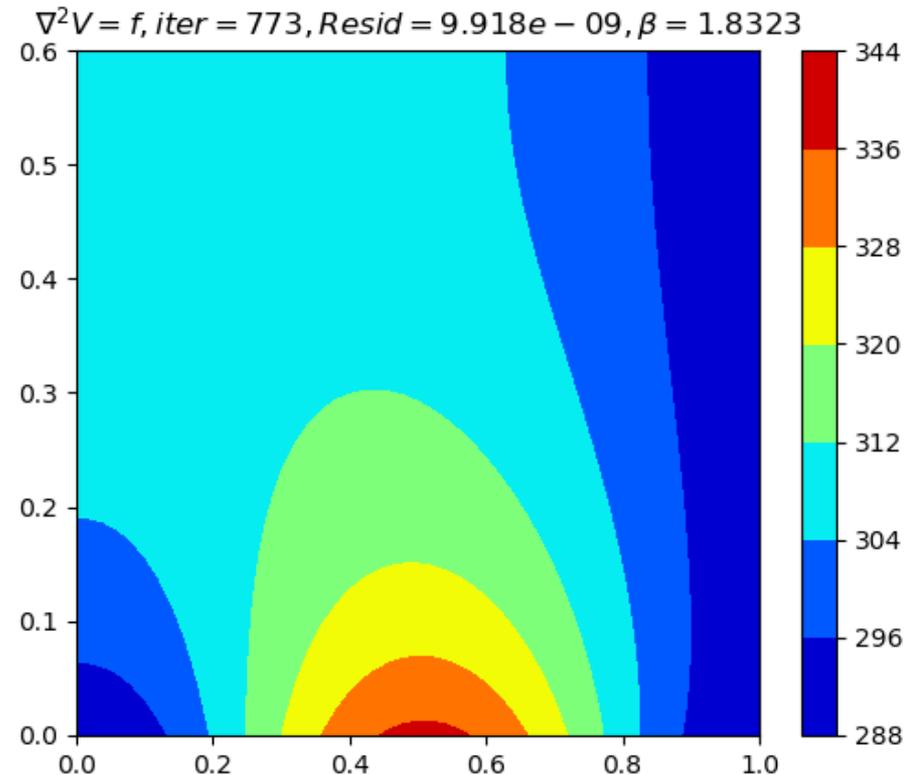
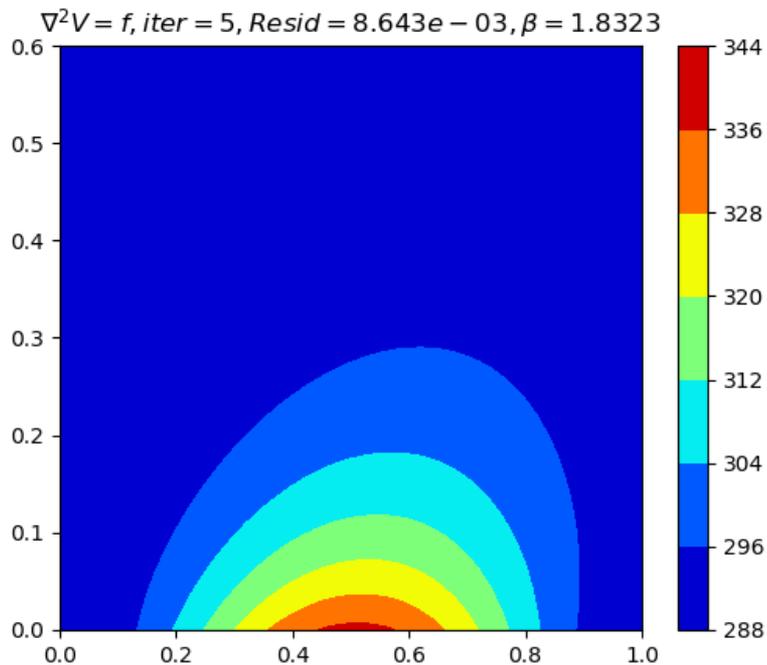
$V = 288$



$\frac{\partial V}{\partial x} = 0$

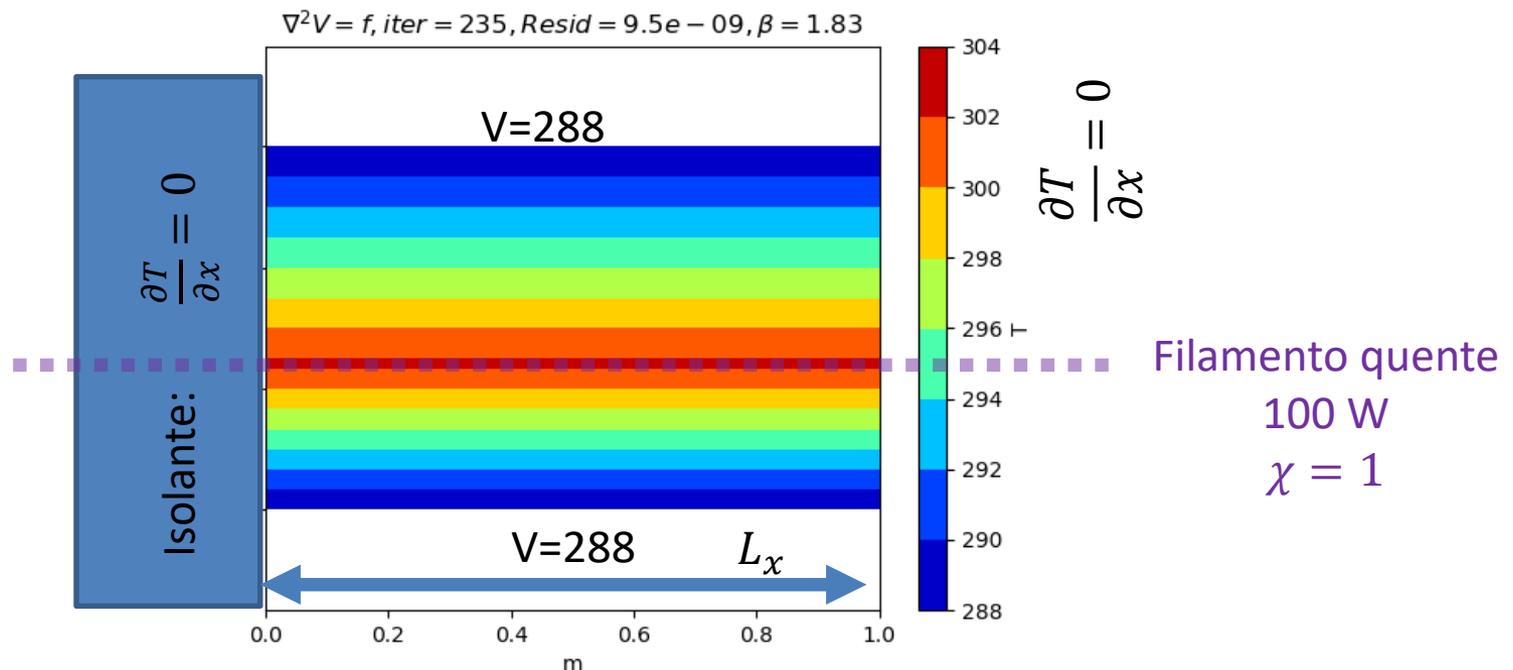
$$V = 288 - 50 * \sin\left(\frac{x\pi}{M}\right)^2$$

```
BxL=np.zeros ( (N) ) ;ByH=np.zeros ( (M) )  
[V,niter,res,beta]=poisson (f,V0,X,Y,maxiter,  
maxres,beta0,BxL=BxL,ByH=ByH)
```



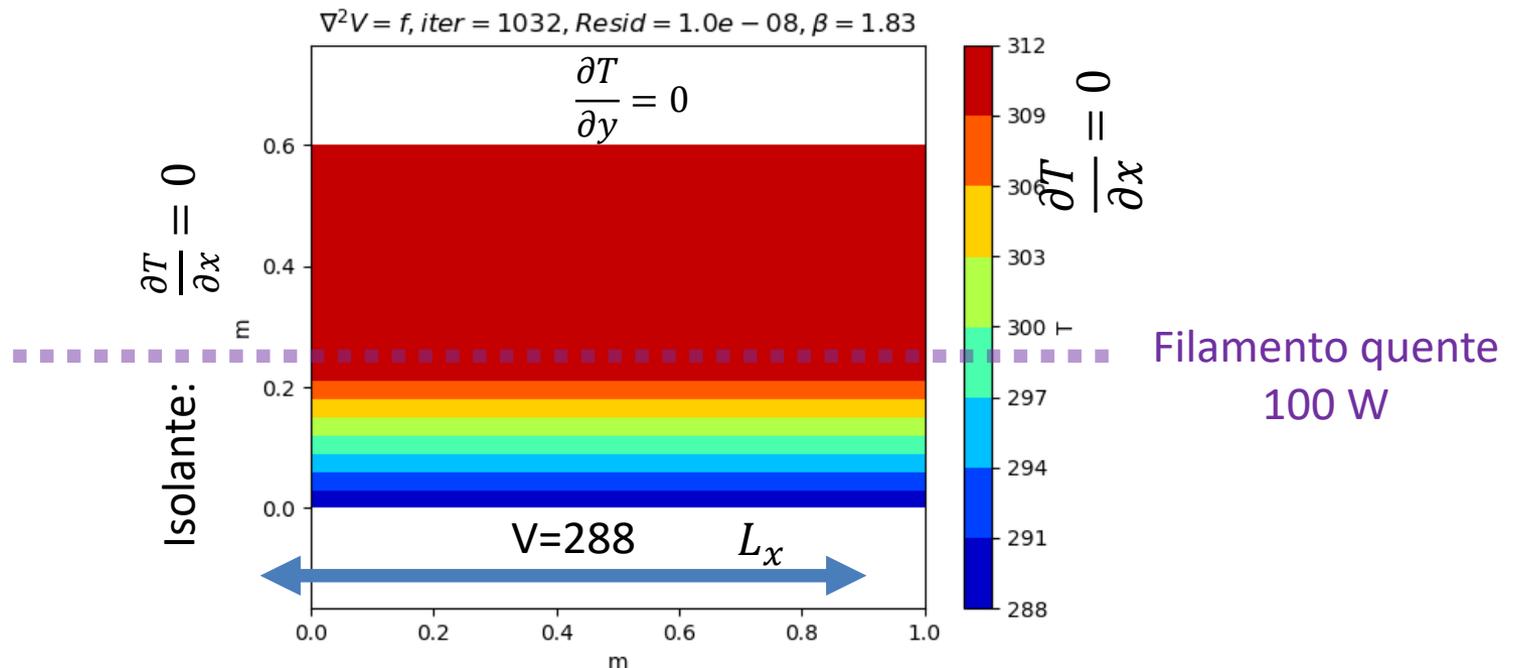
Mesmo, forçamento, outras condições fronteira

```
f=np.zeros( (M,N) )  
for k in range(0,M):  
    f[k,12]=-100/(Lx*delta)  
[V,niter,res,beta]=poisson(f,V0,X,Y,maxiter,maxres,BxL=np.  
zeros( (N) ),BxH=np.zeros( (N) ))
```

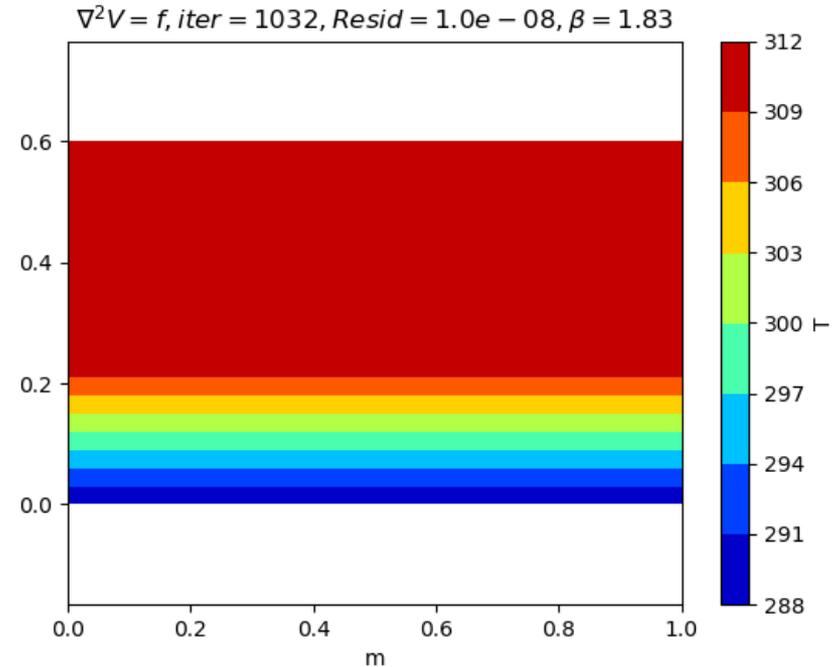
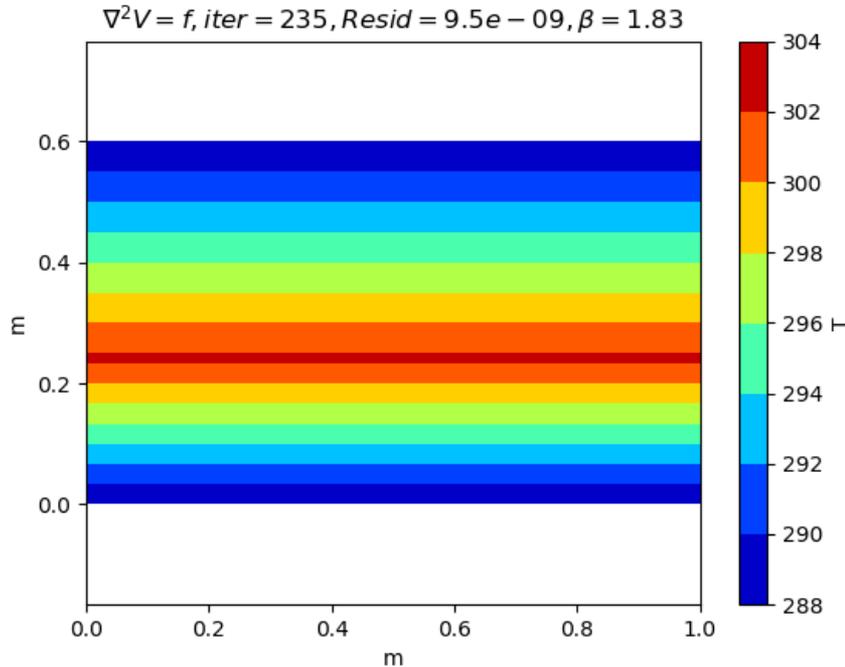


Mesmo, forçamento, outras condições fronteira

```
f=np.zeros ( (M,N) )  
for k in range (0,M) :  
    f[k,12]=-100/ (Lx*delta)  
[V,niter,res,beta]=poisson (f,V0,X,Y,maxiter,maxres,BxL=np.  
zeros ( (N) ) ,BxH=np.zeros ( (N) ) ,ByH=np.zeros ( (M) ) )
```



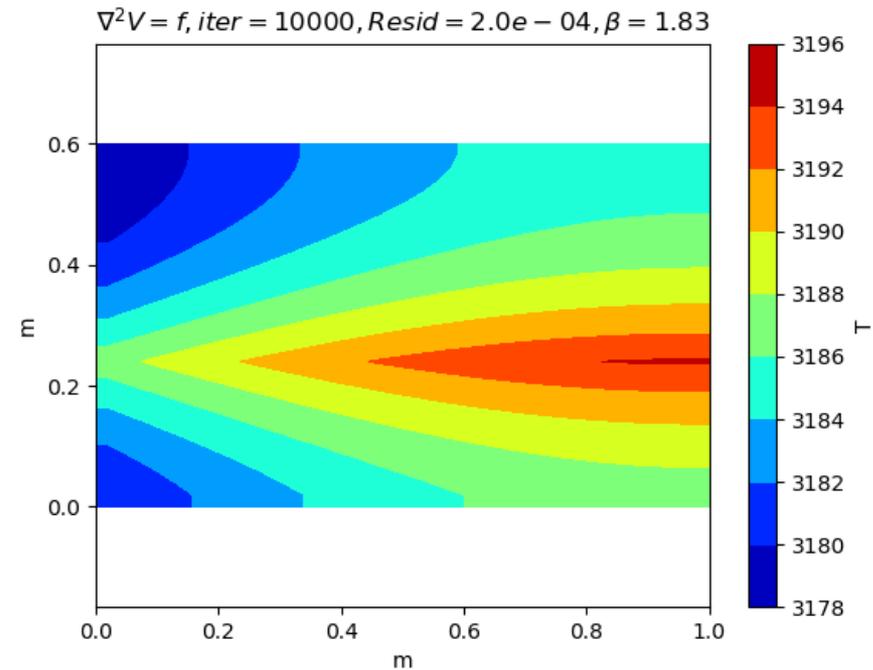
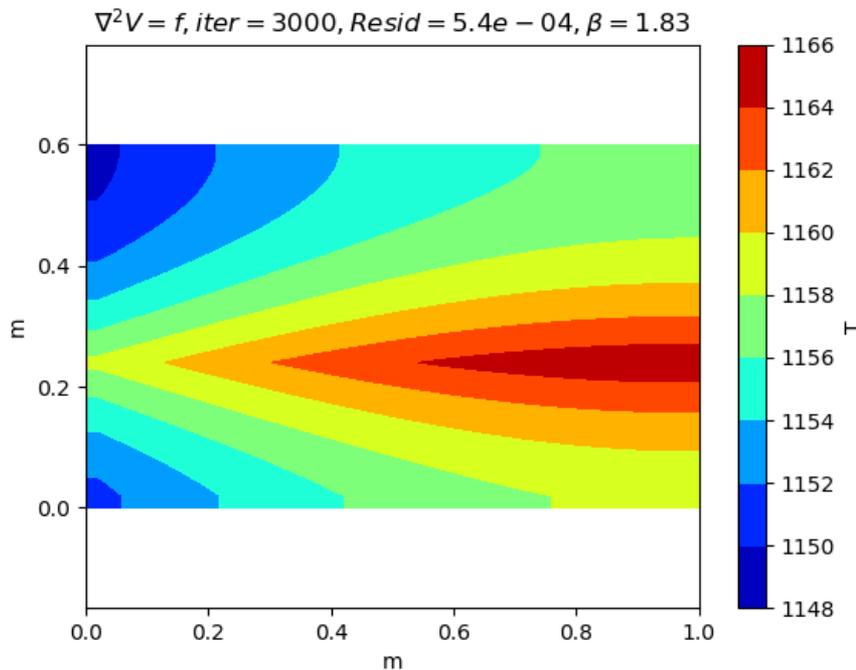
Porque fica mais quente quando isolado $y=L_y$?



Porque todo o calor fornecido (100W) tem que ser conduzido através da fronteira $y=0$ (é preciso o dobro do gradiente $\vec{q} = -\kappa \nabla T$)

```
[V,niter,res,beta]=poisson(f,V0,X,Y,maxiter,
maxres,BxL=np.zeros(N),BxH=np.zeros(N),ByH=np.zeros(M),ByL=np.zeros(M))
```

Não converge: a temperatura subiria sempre porque todas as paredes são não condutoras. A assimetria resulta da ordem de varrimento...



Condução em corpos heterogêneos (κ variável)

$$-\nabla \cdot (-\chi \nabla T) = -\dot{q}_V$$

Caso 2D: \dot{q}_S em Wm^{-2}

$$-\frac{\partial}{\partial x} \left(-\chi \frac{\partial T}{\partial x} \right) - \frac{\partial}{\partial y} \left(-\chi \frac{\partial T}{\partial y} \right) = -\dot{q}_S$$

Numa malha discreta (idêntico para y), em **diferenças centradas**:

$$-\frac{\partial}{\partial x} \left(-\chi \frac{\partial T}{\partial x} \right) \approx 1/\Delta x \left(\chi_{i+1/2} \frac{T_{i+1} - T_i}{\Delta x} - \chi_{i-1/2} \frac{T_i - T_{i-1}}{\Delta x} \right)$$

Com

$$\chi_{i+1/2} = \frac{\chi_i + \chi_{i+1}}{2}, \chi_{j-1/2} = \frac{\chi_{i-1} + \chi_i}{2}$$

$$\begin{aligned}
& \frac{1}{\Delta x} \left(\chi_{i+\frac{1}{2},j} \frac{T_{i+1,j} - T_{i,j}}{\Delta x} - \chi_{i-\frac{1}{2},j} \frac{T_{i,j} - T_{i-1,j}}{\Delta x} \right) \\
& + \frac{1}{\Delta y} \left(\chi_{i,j+\frac{1}{2}} \frac{T_{i,j+1} - T_{i,j}}{\Delta y} - \chi_{i,j-\frac{1}{2}} \frac{T_{i,j} - T_{i,j-1}}{\Delta y} \right) + f_{i,j} = R_{i,j} \\
& \frac{1}{\Delta x^2} \left(\chi_{i+\frac{1}{2},j} T_{i+1,j} - \chi_{i+\frac{1}{2},j} T_{i,j} - \chi_{i-\frac{1}{2},j} T_{i,j} + \chi_{i-\frac{1}{2},j} T_{i-1,j} \right) \\
& + \frac{1}{\Delta y^2} \left(\chi_{i,j+\frac{1}{2}} T_{i,j+1} - \chi_{i,j+\frac{1}{2}} T_{i,j} - \chi_{i,j-\frac{1}{2}} T_{i,j} + \chi_{i,j-\frac{1}{2}} T_{i,j-1} \right) \\
& + f_{i,j} = R_{i,j}
\end{aligned}$$

Sobre-relaxação simultânea

$$\begin{aligned} & \frac{\chi_{i-\frac{1}{2},j}}{\Delta x^2} T_{i-1,j} + \frac{\chi_{i+\frac{1}{2},j}}{\Delta x^2} T_{i+1,j} + \frac{\chi_{i,j-\frac{1}{2}}}{\Delta y^2} T_{i,j-1} + \frac{\chi_{i,j+\frac{1}{2}}}{\Delta y^2} T_{i,j+1} \\ & - \left(\frac{\chi_{i-\frac{1}{2},j}}{\Delta x^2} + \frac{\chi_{i+\frac{1}{2},j}}{\Delta x^2} + \frac{\chi_{i,j-\frac{1}{2}}}{\Delta y^2} + \frac{\chi_{i,j+\frac{1}{2}}}{\Delta y^2} \right) T_{i,j} + f_{i,j} = R_{i,j} \\ T_{i,j}^{novo} &= T_{i,j} + \beta \left(\frac{\chi_{i-\frac{1}{2},j}}{\Delta x^2} + \frac{\chi_{i+\frac{1}{2},j}}{\Delta x^2} + \frac{\chi_{i,j-\frac{1}{2}}}{\Delta y^2} + \frac{\chi_{i,j+\frac{1}{2}}}{\Delta y^2} \right)^{-1} R_{j,k} \end{aligned}$$

χ variável: $-\nabla \cdot (-\chi \nabla T) = -\dot{q}_V$

```
def poissonT(f,V0,chi,X,Y,maxiter,maxres,\
            beta0=0.,BxL=[],BxH=[],ByL=[],ByH=[]):
    ...
    V=V0
    chix=np.zeros((M,N))
    chiy=np.zeros((M,N))
    for i in range(1,M):
        for j in range(N):
            chix[i,j]=0.5*(chi[i,j]+chi[i-1,j])
    for i in range(M):
        for j in range(1,N):
            chiy[i,j]=0.5*(chi[i,j]+chi[i,j-1])
    chix=chix/dx**2
    chiy=chiy/dy**2
```

relaxação $T_{i,j}^{novo} = T_{i,j} + \beta \left(\frac{\chi_{i-\frac{1}{2},j}}{\Delta x^2} + \frac{\chi_{i+\frac{1}{2},j}}{\Delta x^2} + \frac{\chi_{i,j-\frac{1}{2}}}{\Delta y^2} + \frac{\chi_{i,j+\frac{1}{2}}}{\Delta y^2} \right)^{-1} R_{j,k}$

```

itera=0
resid=2*maxres #garante a primeira iteração
pngs=[]
while resid>maxres and itera<maxiter: #iterações
    itera=itera+1
    resid=0; vmax=0
    for i in range(1,M-1):
        for j in range(1,N-1):
            chi4=(chix[i,j]+chix[i+1,j]\
                +chiy[i,j]+chiy[i,j+1])
            R=chiy[i,j]*V[i,j-1]+ chiy[i,j+1]*V[i,j+1]+\
                chix[i,j]*V[i-1,j]+chix[i+1,j]*V[i+1,j]\
                -chi4*V[i,j]-f[i,j]
            V[i,j]=V[i,j]+beta*R/chi4;
            resid=max(resid,abs(R))

```

Condições fronteira (de von Neumann)

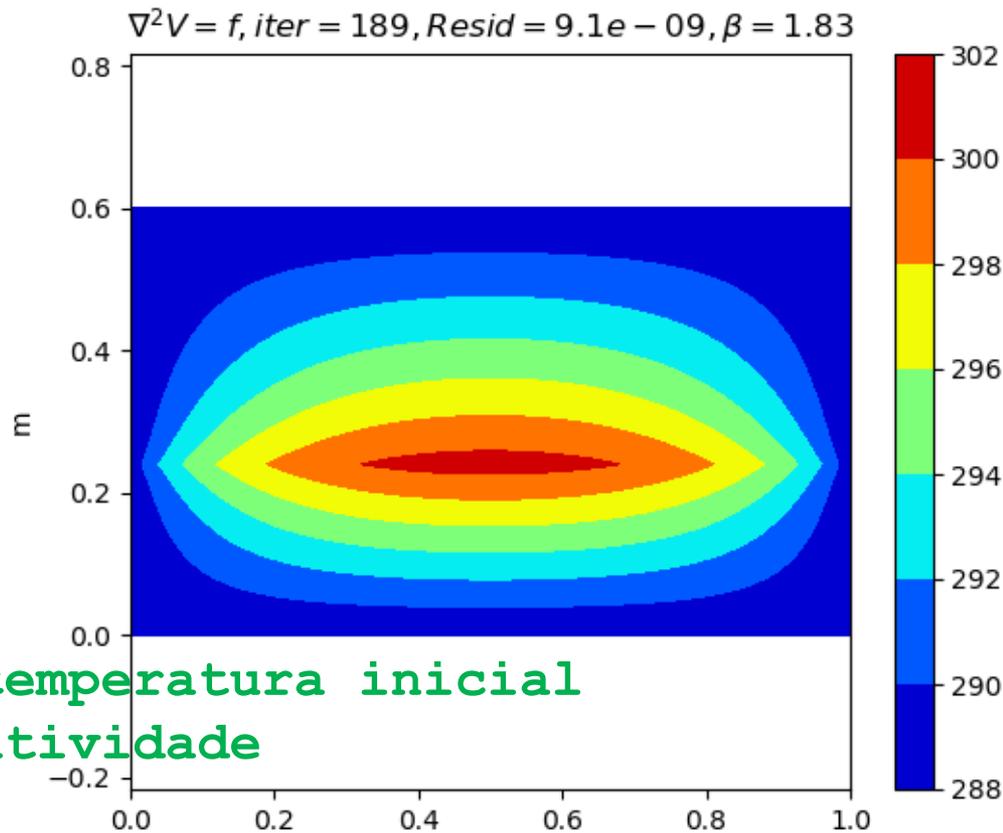
(notar que as derivadas são perpendiculares à fronteira de dentro para fora)

```
#Neumann X (caso contrário, mantém V fixo na fronteira)
    if len(BxL) !=0:
        V[0, :] = V[1, :] + BxL*dx
    if len(BxH) !=0:
        V[M-1, :] = V[M-2, :] + BxH*dx
#Neumann y (caso contrário, mantém V fixo na fronteira)
    if len(ByL) !=0:
        V[:, 0] = V[:, 1] + ByL*dy
    if len(ByH) !=0:
        V[:, N-1] = V[:, N-2] + ByH*dy

vmax=np.max(np.abs(V))
resid=resid/vmax #residuo relativo
```

teste

```
V0=288.*np.ones((M,N)) #temperatura inicial
chi=np.ones((M,N)) #condutividade
f=np.zeros((M,N))
for k in range(0,M): #filamento retilíneom
    f[k,12]=-100/(Lx*delta)
BxH=np.zeros((N))
ByH=-np.ones((M))*10
ByL=np.zeros((M))
[V,niter,res,beta]=poissonT(f,V0,chi,X,Y,maxiter,maxres)
```



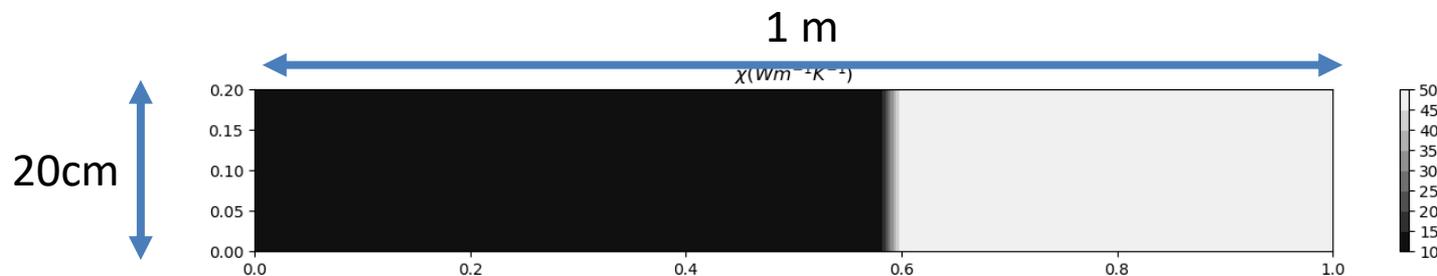
Notas

No caso em que $\chi = \text{const}$ a nova função dá os resultados obtidos anteriormente.

No entanto agora $f = -\dot{q}_S$, e no caso anterior era $f = -\frac{\dot{q}_S}{\chi}$.

Condução de calor através de uma parede heterogénea

```
V0=273.*np.ones((M,N)) #potencial inicial e cf em y=Ly
V0[:,0]=300 #condição fronteira em y=0
chi=np.ones((M,N))*50 #condutividade
chi[0:30,:]=10 #condutividade
f=np.zeros((M,N)) #não há fontes de calor internas
[T,niter,res,beta]=poissonT(f,V0,chi,X,Y,maxiter,maxres,\
    BxL=np.zeros((N)),BxH=np.zeros((N)))
```



Fluxos de calor em cada fronteira

