



Ciências
ULisboa

Modelação Numérica

Aula 19

+Transferência de calor

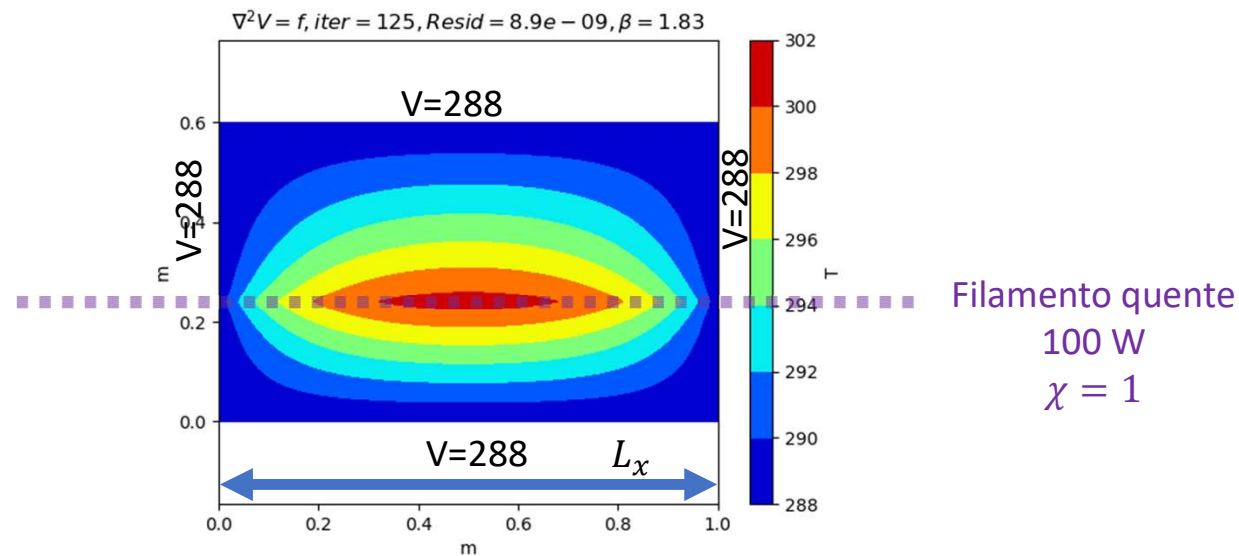
poisson.py v2 com condições fronteira opcionais (von Neumann)

```
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
```

Introduzindo fontes de calor (internas)

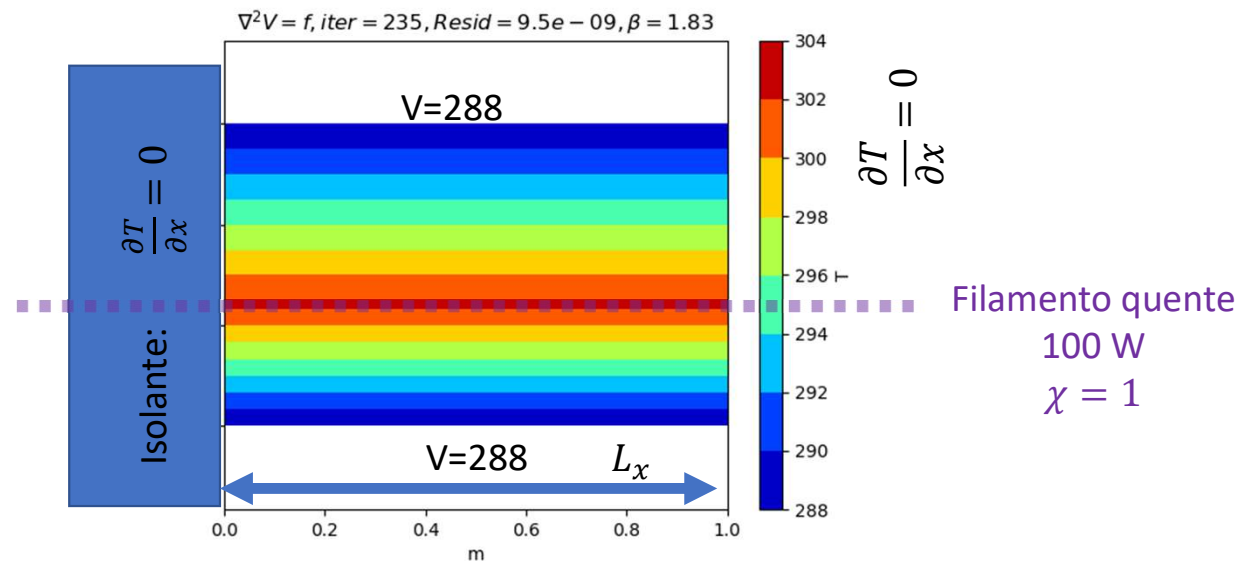
```
f=np.zeros ( (M,N) )  
for k in range (0,M) :  
    f[k,12]=-100 / (Lx*delta)  
[V,niter,res,beta]=poisson (f,V0,X,Y,maxiter,maxres)
```

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



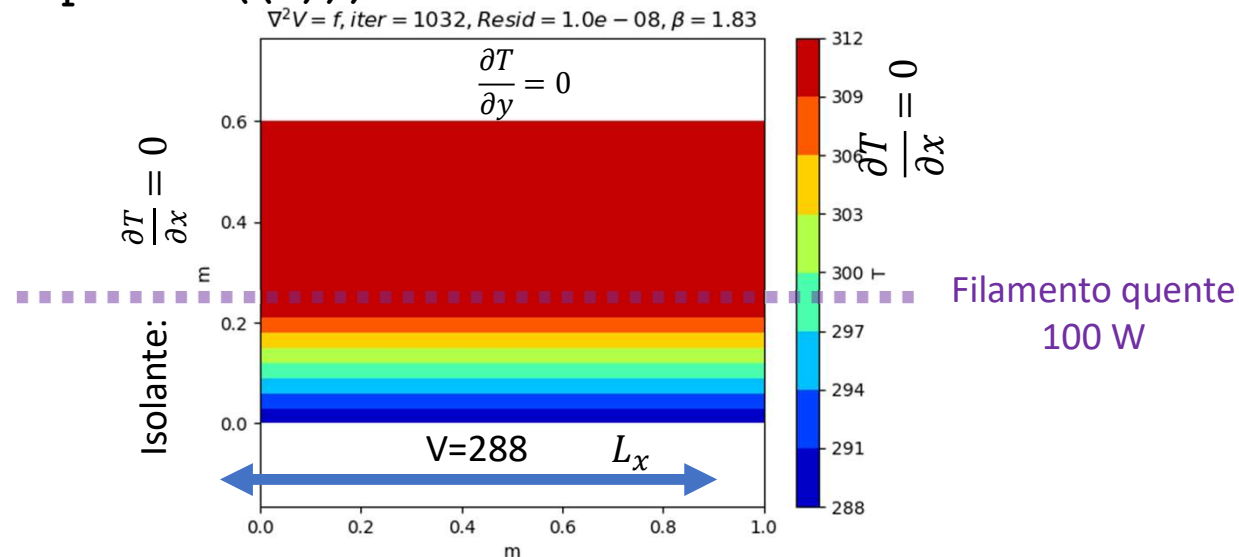
Mesmo, forçamento, outras condições fronteira

```
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) ) )
```

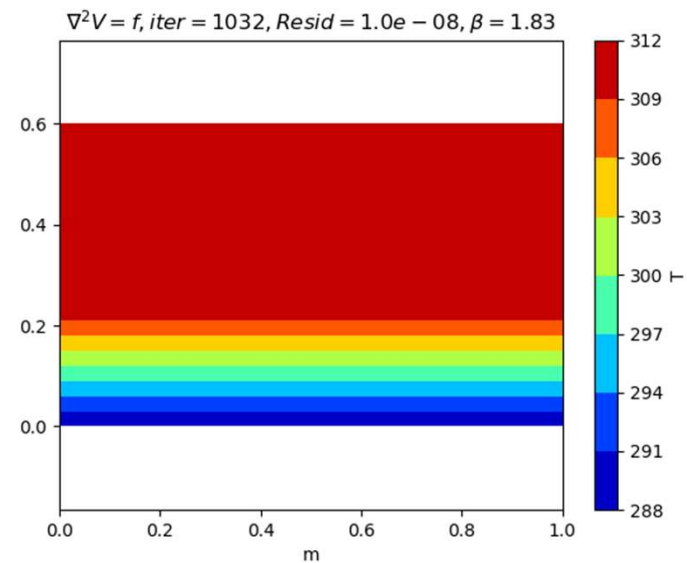
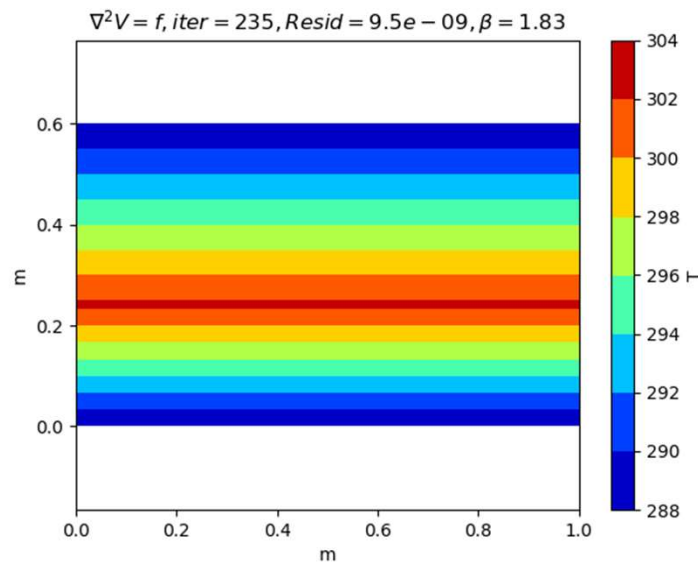


Mesmo, forçamento, outras condições fronteira

```
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) ) )
```



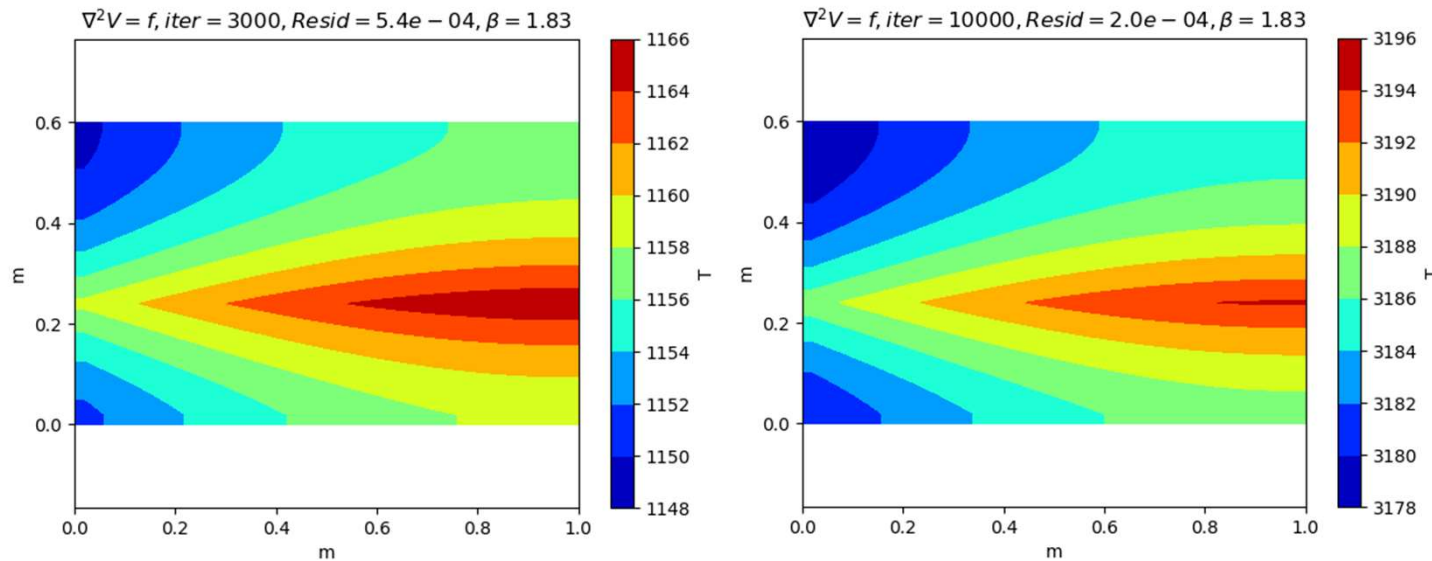
Porque fica mais quente quando isolado
 $y=L$?



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

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

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

```
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)

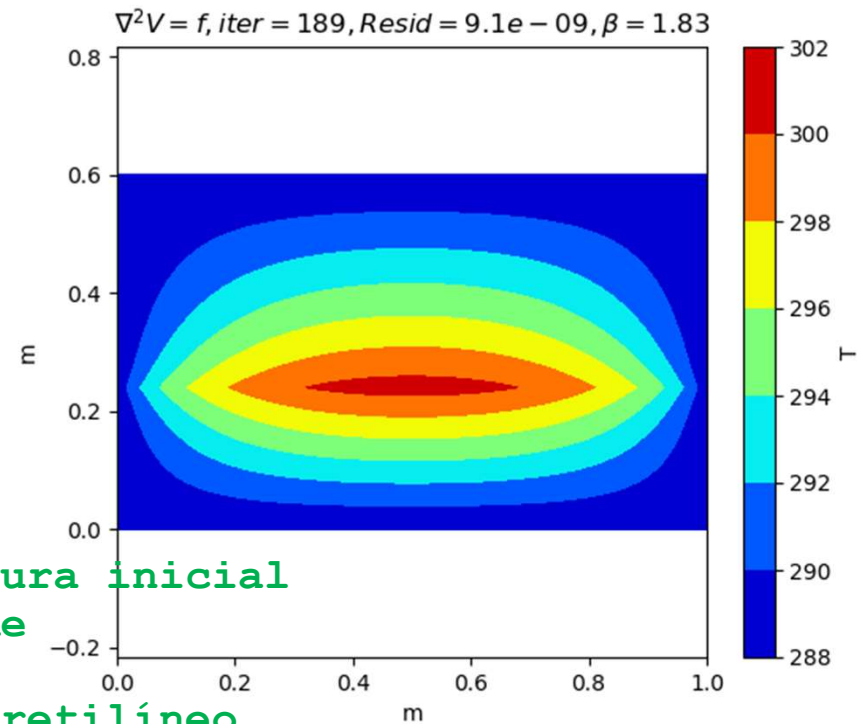
(notar que as derivadas são perpendiculares à fronteira de dentro para fora)

```
#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 retilíneo
    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)
```



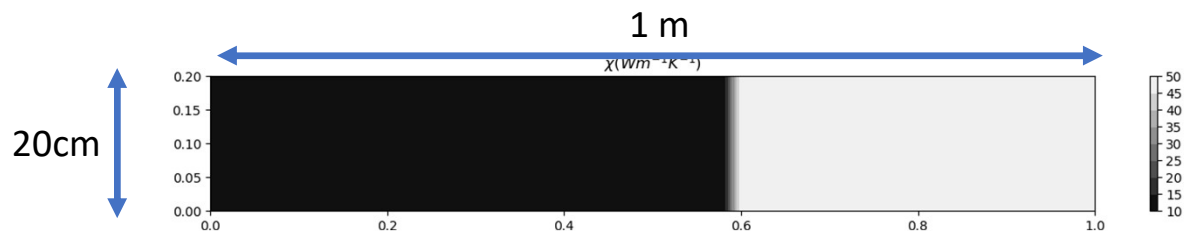
Notas

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

