Ciências ULisboa

Modelação Numérica au

Equação de Poisson

Lei de Fourier da condução em 3 dimensoões

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

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

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

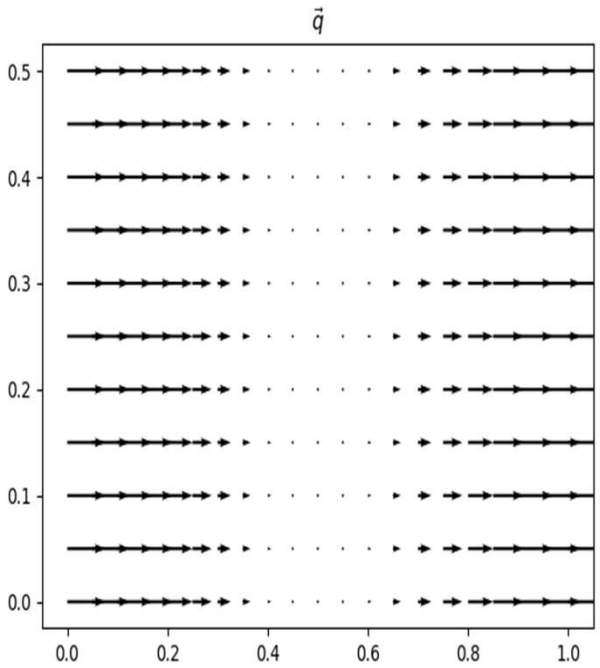
A taxa de variação da temperatura num ponto depende da distribuição do fluxo de calor:

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

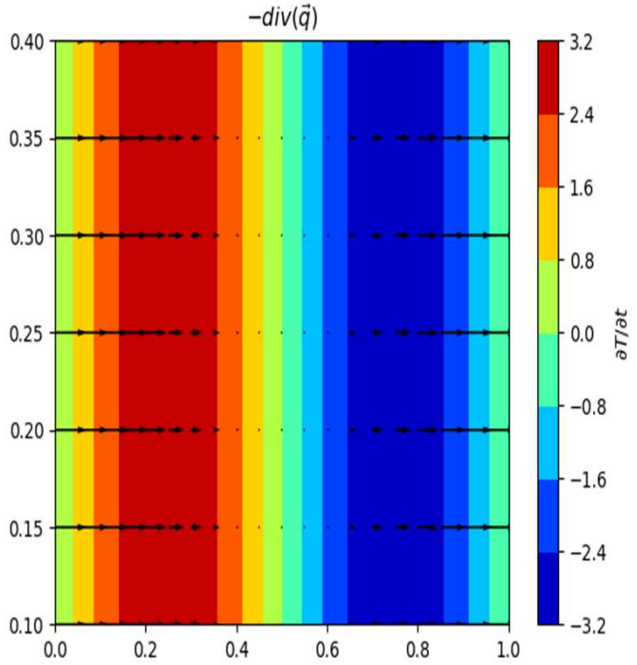
ρ é a densidade, c_p é o calor específico do meio.

$$-\nabla \cdot (-\chi \nabla T) = -\nabla \cdot (\vec{q}) = -\text{div}\, \vec{q}$$

Fluxo de calor



Taxa de aquecimento



Zona de convergência

Zona de divergência

Na presença de fontes internas de calor…

Incluindo fontes internas de calor (reações químicas, decaimento radioativo, ou outras) dadas por $\dot{q}_V \left(Wm^{-3}\right)$, 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 homogéneo, 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:

$$\nabla^2 T = 0$$

Equação de Poisson da electrostática

o potencial gerado por uma distribuição contínua de carga elétrica numa placa satisfaz à equação:

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = -\frac{\rho}{\varepsilon_0}$$

em que V é o potencial elétrico, $\rho(x, y)$ a densidade volúmica de carga e ε_0 a permitividade elétrica do meio.

Em três dimensões, seria:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{\rho}{\varepsilon_0}$$

Equação de Poisson

Eq. Poisson da electrostática:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{\rho}{\varepsilon_0}$$

Eq. de Fourier da condução de calor em meio homogéneo:

$$\nabla^2 T = \frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} + \frac{\partial^2 T}{\partial z^2} = -\frac{1}{\chi}\,\dot{q}_V$$

Forma geral:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = f(x, y, z)$$

Caso particular (equação de Laplace): $f(x, y, z)$=0

Problemas de condição fronteira

A equação de Poisson não pode ser resolvida progressivamente a partir de um ponto, como fizemos com as trajetórias. A solução é sempre global. Vamos considerar o caso 2D. Na forma discreta, a equação pode escrever-se, usando diferenças finitas centradas:

$$\begin{cases} \dfrac{\partial^2 \phi}{\partial x^2} \approx \dfrac{\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}}{\Delta x^2} \\ \dfrac{\partial^2 \phi}{\partial y^2} \approx \dfrac{\phi_{i,j-1} - 2\phi_{i,j} + \phi_{i,j+1}}{\Delta y^2} \end{cases}$$

Onde:

$$\begin{cases} \phi_{i,j} = \phi(x_i, y_j) \\ x_i = (i-1)\Delta x; i = 0, \dots, M-1 \\ y_j = (j-1)\Delta y; j = 0, \dots, N-1 \end{cases}$$

Diferenças centradas

A diferença finita centrada, constitui uma aproximação de segunda ordem, i.e.:

$$\frac{\partial^2 \phi}{\partial x^2} \approx \frac{\phi_{i-1,j} - 2\phi_{i,j} + \phi_{i+1,j}}{\Delta x^2} + E\left(\Delta x^2\right)$$

Resultando da soma das duas séries de Taylor:

$$\phi(x + \Delta x) = \phi(x) + \frac{d\phi}{dx}\Delta x + \frac{1}{2}\frac{d^2\phi}{dx^2}\Delta x^2 + \frac{1}{3!}\frac{d^3\phi}{dx^3}\Delta x^3 + \cdots$$

$$\phi(x - \Delta x) = \phi(x) - \frac{d\phi}{dx}\Delta x + \frac{1}{2}\frac{d^2\phi}{dx^2}\Delta x^2 - \frac{1}{3!}\frac{d^3\phi}{dx^3}\Delta x^3 + \cdots$$

$$\phi(x + \Delta x) + \phi(x - \Delta x) = 2\phi(x) + \frac{d^2\phi}{dx^2}\Delta x^2 + \frac{2}{4!}\frac{d^4\phi}{dx^4}\Delta x^4 + \cdots$$

etc.

Equação de Poisson discreta, em diferenças centradas

Se for $\Delta x = \Delta y = \Delta$, fica

$$\phi_{i-1,j} + \phi_{i+1,j} + \phi_{i,j-1} + \phi_{i.j+1} - 4\phi_{i,j} = \Delta^2 f_{i,j}$$
$$\{i = 1, \dots, M-2; j = 1, \dots, N-2\}$$

onde se notou que as diferenças centradas só se podem calcular nos pontos interiores do domínio. Na fronteira, os valores $(i = 0, M-1; j = 0, N-1)$, têm que ser impostos.

A solução depende da fronteira!

Método da relaxação

A solução satisfaz:

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

Começamos por aribtrar uma distribuição para ϕ, por exemplo $\phi = 0$, e vamos melhorar essa estimativa, de forma iterativa:

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

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

Se se corrigir:

$$\phi^{n+1}_{i,j} = \phi^n_{i,j} + \frac{R_{i,j}}{4}$$

O erro será anulado (mas só nesse ponto!)

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 sobrerrelaxação
        beta=2-np.pi*np.sqrt(2.)*np.sqrt(1./M**2+1./N**2)
    else:
        beta=beta0
    V=V0 #inicializa a matriz soluçáo
    iter=0
    resid=2*maxres #garante a primeira iteração
    pngs=[]
```

python

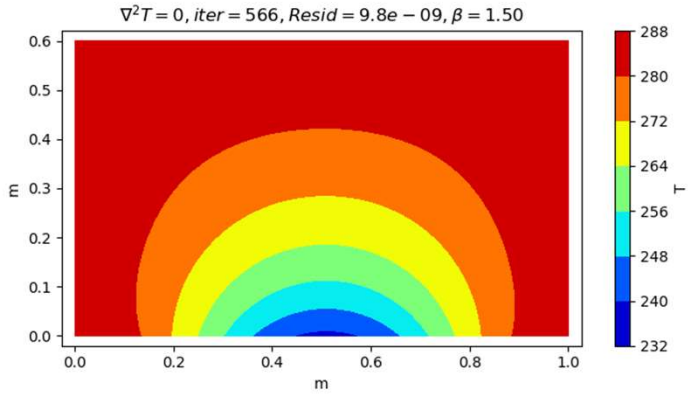
```
        while resid>maxres and iter<maxiter: #iterações
        iter=iter+1
        resid=0; vmax=0
        for i in range(1,M-1): #vai de 1 a M-2
            for j in range(1,N-1):
                R=V[i,j-1]+V[i,j+1]+V[i-1,j]+\
                  V[i+1,j]-4*V[i,j]-delta**2*f[i,j]
                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
```

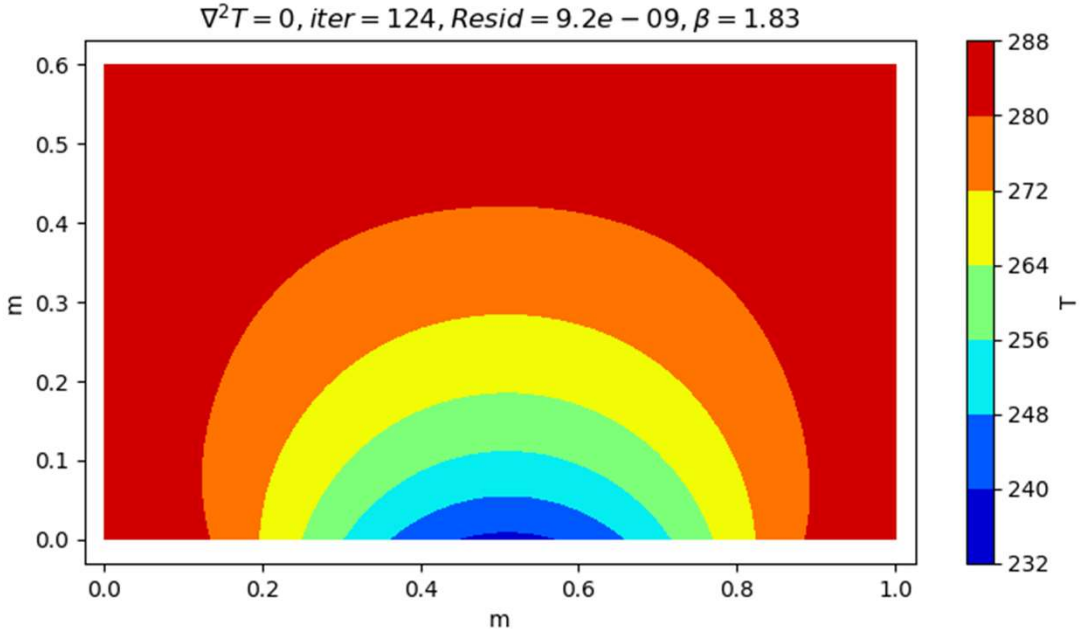
Main $\nabla^2 T = 0$

```
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))
```

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





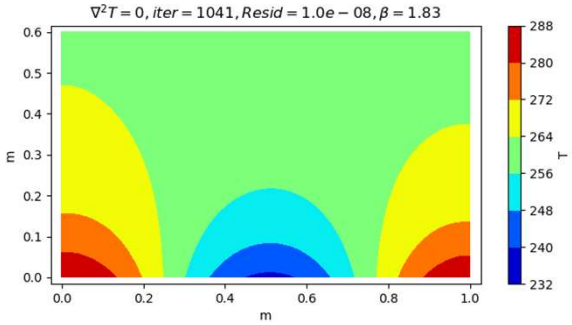
Com sobre-relaxação óptima

Vamos repensar a condição fronteira

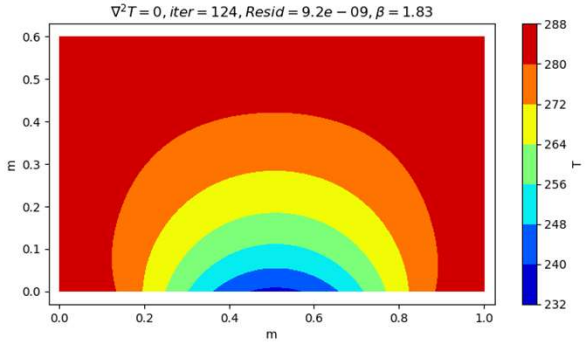
```
#condições fronteira: entram aqui!
#se não se fizer nada V na fronteira fica sempre o mesmo
#o que é uma condição fronteira possível
```

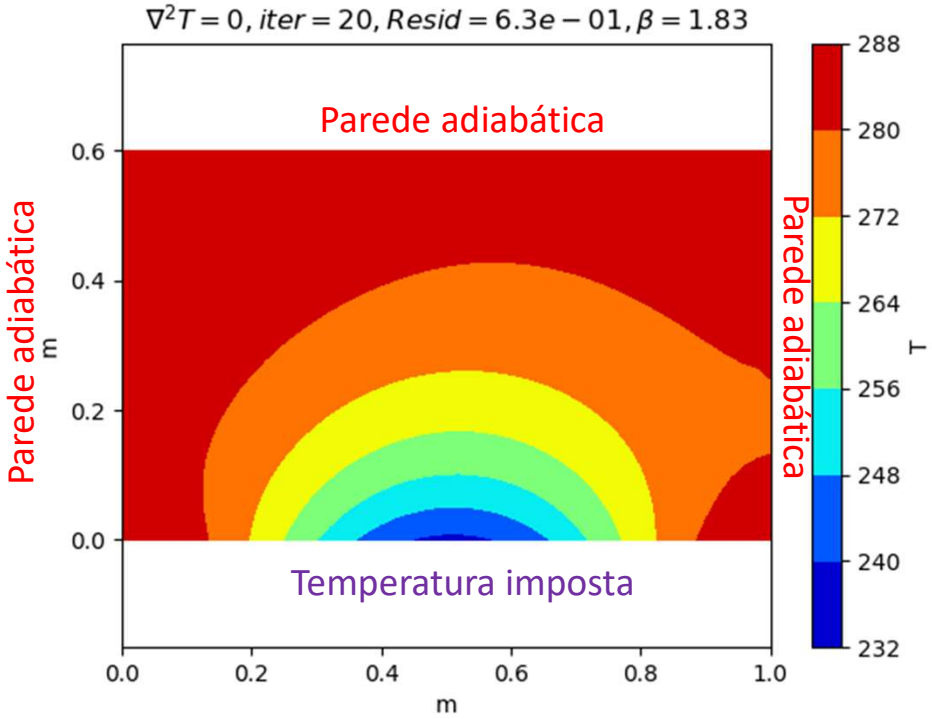
$\left(\frac{\partial T}{\partial n}\right)_{\text{fronteira}} = 0 \Leftrightarrow$ Fronteira isolante (fluxo de calor zero)

```
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
```



Paredes topo, esquerda e direita adiabáticas
←





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 $\left(Wm^{-2}\right)$ é proporcional ao gradiente de temperatura, transportando calor em direção às regiões mais frias:

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

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 $\dot{q}_V \left(Wm^{-3}\right)$, 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 homogéneo 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)

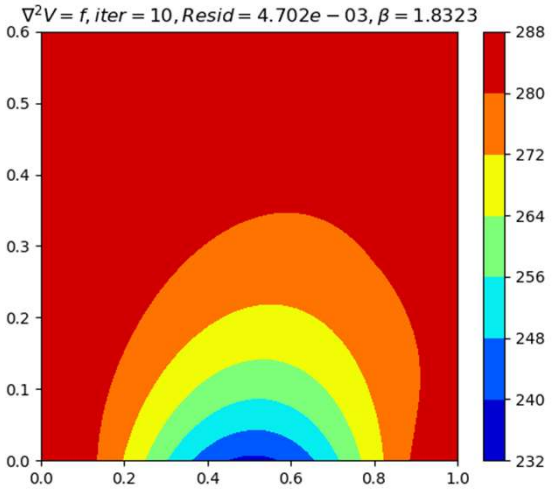
```
def poisson(f,V0,X,Y,maxiter,maxres,\  #posicionais (obrigatórias)
            beta0=0.,BxL=[],BxH=[],ByL=[],ByH=[],movie='',passo=1):    #kwargs
    [M,N]=f.shape; #determina a dimensão das matrizes
    …
    V=V0 #inicializa a matriz soluçáo
    iter=0; resid=2*maxres
    while resid>maxres and 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

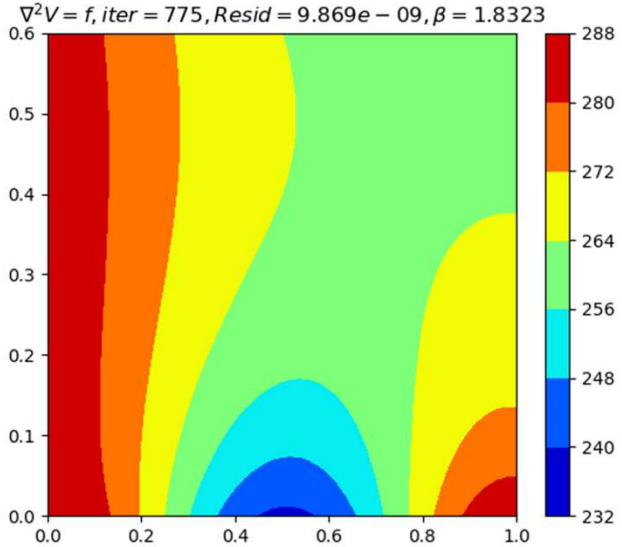
```
import numpy as np;import matplotlib.pyplot as plt
Lx=1.;Ly=1.2;M=51;N=31;delta=Lx/(M-1)
X=np.zeros((M,N));Y=np.zeros((M,N));ro=np.zeros((M,N));f=np.zeros((M,N))
for i in range(M):
    for j in range(N):
        X[i,j]=i*delta; Y[i,j]=j*delta
maxiter=3000;maxres=1.e-8 # erro relativo
V0=288.*np.ones((M,N)) #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*np.sin(ix*np.pi/M)**2
BxH=np.zeros((N));ByH=-np.ones((M))*10
```

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



$V = 288$



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

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

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