



Ciências
ULisboa

Modelação Numérica 2022

Aula 12

Equação de Poisson

Calcular a distribuição de temperatura numa barra composta 1D

Admite-se que não existe fluxo lateral de calor.

Na direção longitudinal vale a lei de Fourier:

$$\text{Fluxo de calor} = -k \frac{\partial T}{\partial x}$$

Metal	Condutividade $Wm^{-1}K^{-1}$
Aluminio	237
Cobre	401
Ferro	80



Admite-se que a barra atingiu o equilíbrio térmico ($T = \text{const}$ em cada ponto), logo Fluxo independente de x . Logo, em cada metal ($k = \text{const}$):

$$\frac{\partial T}{\partial x} = \text{const}$$

Isto é a temperatura varia linearmente, seguindo uma **linha quebrada**, com quebras nas transições entre materiais.

Discussão

A equação da condução de calor, de **Fourier**, é uma equação diferencial:

$$\frac{\partial T}{\partial t} = -\nabla \cdot (-k\nabla T)$$

reduzindo-se, o **caso estacionário**, com condutividade constante, à **Equação de Laplace**:

$$\nabla^2 T = 0$$

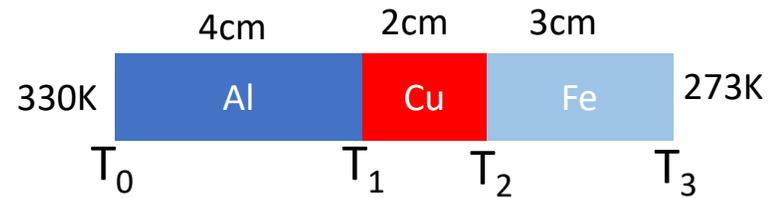
No caso unidimensional (com k variável):

$$-\frac{\partial}{\partial x} \left(-k \frac{\partial T}{\partial x} \right) = 0$$

No caso em que k é **constante por troços**, obtém-se (**de forma exata**) um sistema de **equações lineares algébricas**.

O caso geral, em que é preciso resolver de forma aproximada, a **equação diferencial**, será tratado mais tarde.

Condução

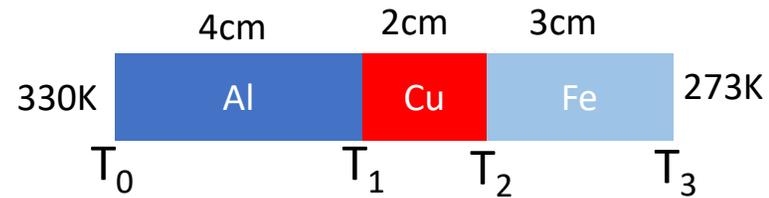


$$\text{Fluxo de calor} = k_{Al} \left(-\frac{\partial T}{\partial x} \right) \equiv k_{Al} \frac{T_0 - T_1}{x_1 - x_0} = k_{Cu} \frac{T_1 - T_2}{x_2 - x_1} = k_{Fe} \frac{T_2 - T_3}{x_3 - x_2} = \text{const}$$

Sistema de equações lineares algébricas:

$$\begin{cases} -\frac{k_{Al}}{x_1 - x_0} T_1 - \frac{k_{Cu}}{x_2 - x_1} T_1 + \frac{k_{Cu}}{x_2 - x_1} T_2 = -\frac{k_{Al}}{x_1 - x_0} T_0 \\ \frac{k_{Cu}}{x_2 - x_1} T_1 - \frac{k_{Cu}}{x_2 - x_1} T_2 - \frac{k_{Fe}}{x_3 - x_2} T_2 = -\frac{k_{Fe}}{x_3 - x_2} T_3 \end{cases}$$

Forma matricial



$$\begin{cases} -\frac{k_{Al}}{x_1 - x_0} T_1 - \frac{k_{Cu}}{x_2 - x_1} T_1 + \frac{k_{Cu}}{x_2 - x_1} T_2 = -\frac{k_{Al}}{x_1 - x_0} T_0 \\ \frac{k_{Cu}}{x_2 - x_1} T_1 - \frac{k_{Cu}}{x_2 - x_1} T_2 - \frac{k_{Fe}}{x_3 - x_2} T_2 = -\frac{k_{Fe}}{x_3 - x_2} T_3 \end{cases}$$

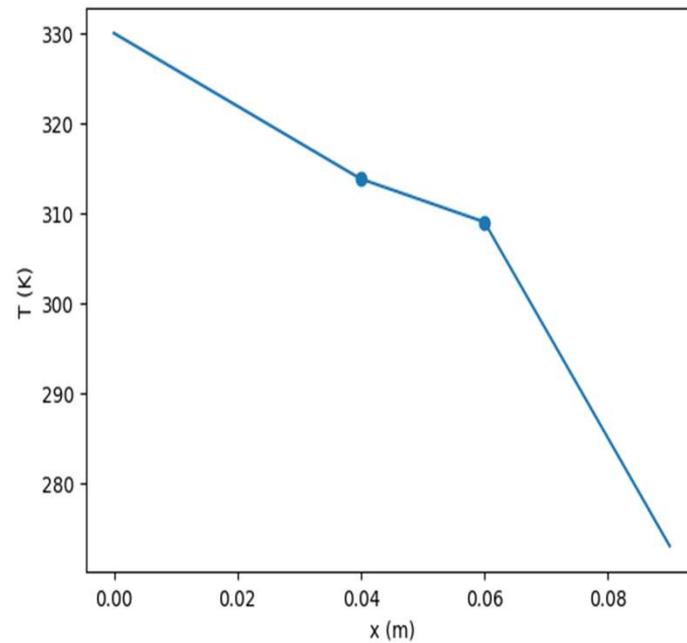
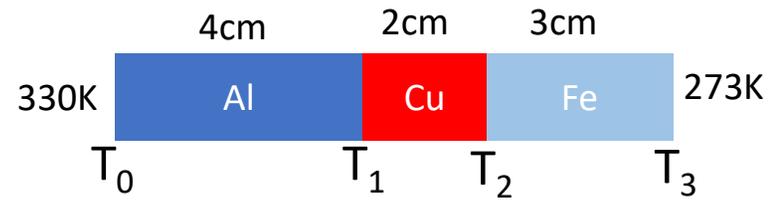
$$\begin{bmatrix} -\frac{k_{Al}}{x_1 - x_0} & -\frac{k_{Cu}}{x_2 - x_1} & \frac{k_{Cu}}{x_2 - x_1} \\ \frac{k_{Cu}}{x_2 - x_1} & -\frac{k_{Cu}}{x_2 - x_1} & -\frac{k_{Fe}}{x_3 - x_2} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} -\frac{k_{Al}}{x_1 - x_0} T_0 \\ -\frac{k_{Fe}}{x_3 - x_2} T_3 \end{bmatrix}$$

Equilíbrio térmico numa barra

```
import numpy as np
import matplotlib.pyplot as plt
kAl=237;kCu=401;kFe=80;T0=330;T3=273
x0=0;x1=0.04;x2=0.06;x3=0.09
A=np.array([[ -kAl/(x1-x0)-kCu/(x2-x1), kCu/(x2-x1)], \
            [kCu/(x2-x1), -kCu/(x2-x1)-kFe/(x3-x2)]], dtype=float)
b=np.array([-kAl/(x1-x0)*T0, -kFe/\
            (x3-x2)*T3], dtype=float)
T=np.linalg.solve(A,b)
print('T= ',T)
plt.plot([x0,x1,x2,x3], [T0,T[0],T[1],T3])
plt.scatter([x1,x2], T)
plt.xlabel('x (m)')
plt.ylabel('T (K)')
```

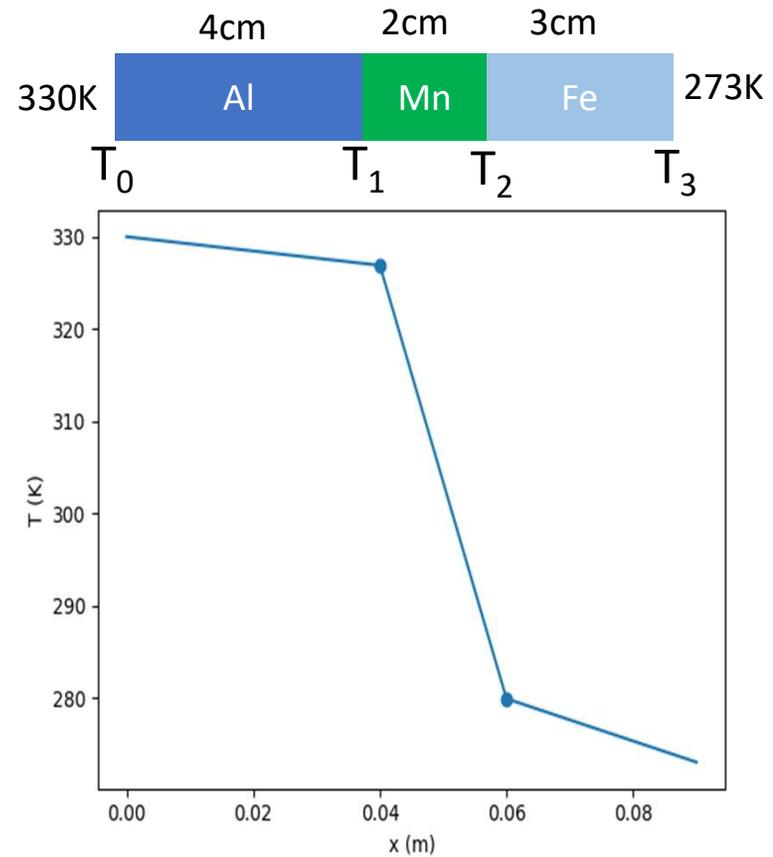
Solução

Metal	Condutividade $Wm^{-1}K^{-1}$
Aluminio	237
Cobre	401
Ferro	80



Solução 2

Metal	Condutividade $Wm^{-1}K^{-1}$
Aluminio	237
Manganésio	7.81
Ferro	80





Se a barra for constituída por n segmentos

$$\begin{bmatrix}
 -\frac{k_1}{\Delta x_1} - \frac{k_2}{\Delta x_2} & \frac{k_2}{\Delta x_2} & \dots & 0 & 0 \\
 \frac{k_2}{\Delta x_2} & -\frac{k_2}{\Delta x_2} - \frac{k_3}{\Delta x_3} & \dots & 0 & 0 \\
 \dots & \dots & \dots & \dots & \dots \\
 0 & 0 & \dots & -\frac{k_{n-2}}{\Delta x_{n-2}} - \frac{k_{n-1}}{\Delta x_{n-1}} & \frac{k_{n-1}}{\Delta x_{n-1}} \\
 0 & 0 & \dots & \frac{k_{n-1}}{\Delta x_{n-1}} & -\frac{k_{n-1}}{\Delta x_{n-1}} - \frac{k_n}{\Delta x_n}
 \end{bmatrix}
 \begin{bmatrix}
 T_1 \\
 T_2 \\
 \dots \\
 T_{n-2} \\
 T_{n-1}
 \end{bmatrix}
 =
 \begin{bmatrix}
 -\frac{k_1}{\Delta x_1} T_0 \\
 0 \\
 \dots \\
 0 \\
 \frac{k_n}{\Delta x_n} T_n
 \end{bmatrix}$$

Trata-se de um **sistema triadiagonal**, cuja solução são as temperaturas nas interfaces $[T_1, \dots, T_{n-1}]$, dadas as temperaturas na fronteira $[T_0, T_n]$ e as condutividades $[k_1, \dots, k_n]$, com:

$$\Delta x_m = x_m - x_{m-1}$$

Lei de Fourier da condução em 3 dimensões

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.

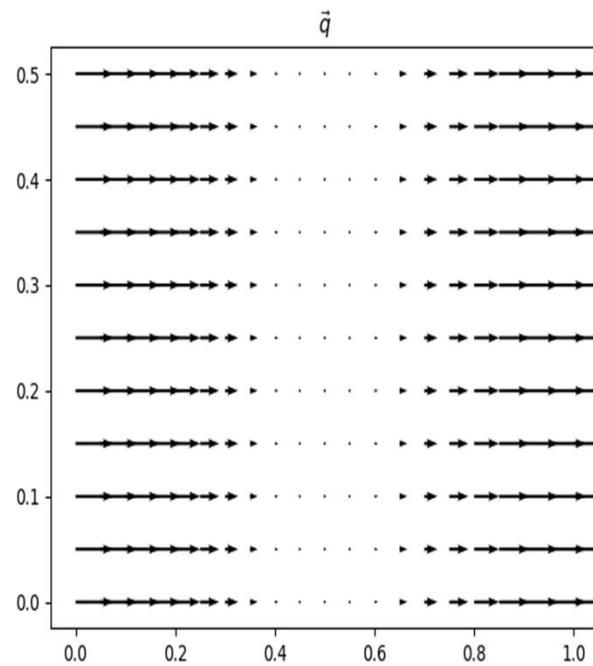
A taxa de variação da temperatura num ponto depende da distribuição do fluxo de calor:

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

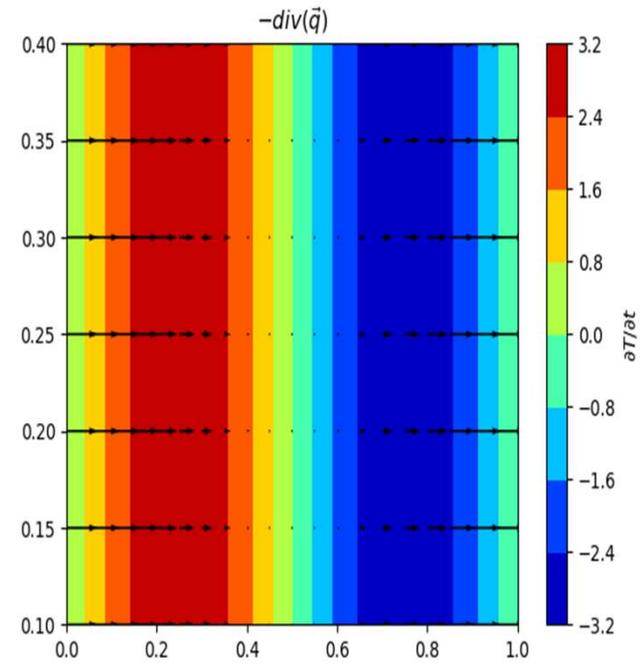
ρ é a densidade, c_p é o calor específico do meio.

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

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 (Wm^{-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**:

$$\nabla^2 T = 0$$

Equação de Poisson da electrostática

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

$$\frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} = -\frac{\rho}{\epsilon_0}$$

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

Em três dimensões, seria:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{\rho}{\epsilon_0}$$

Equação de Poisson

Eq. Poisson da electrostática:

$$\nabla^2 V = \frac{\partial^2 V}{\partial x^2} + \frac{\partial^2 V}{\partial y^2} + \frac{\partial^2 V}{\partial z^2} = -\frac{\rho}{\epsilon_0}$$

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:

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

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.:

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

Resultando da soma das duas séries de Taylor:

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

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

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

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 arbitrar 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):

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

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!**)

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

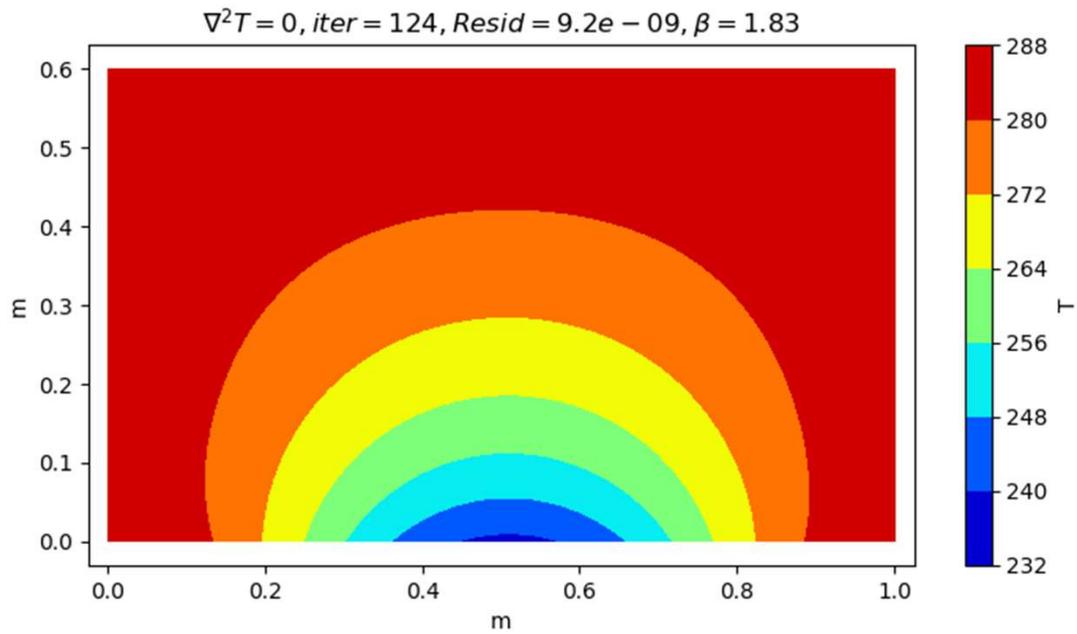
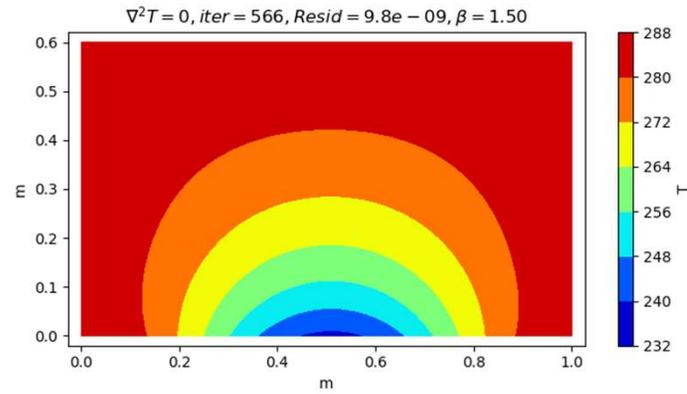
```
while resid>maxres and iter<maxiter: #iterações
iter=iter+1
resid=0; vmax=0
for i in range(1,M-1): #vai de 1 a M-2
    for j in range(1,N-1):
        R=V[i,j-1]+V[i,j+1]+V[i-1,j]+\
            V[i+1,j]-4*V[i,j]-delta**2*f[i,j]
        V[i,j]=V[i,j]+beta*R/4;
        resid=max(resid,abs(R))
#condições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

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...)

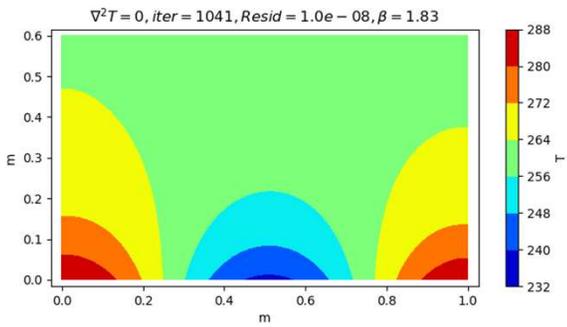
Resolve a equação $\nabla^2 V = f$ com a condição fronteira imposta (previamente) nos pontos de fronteira da matriz V



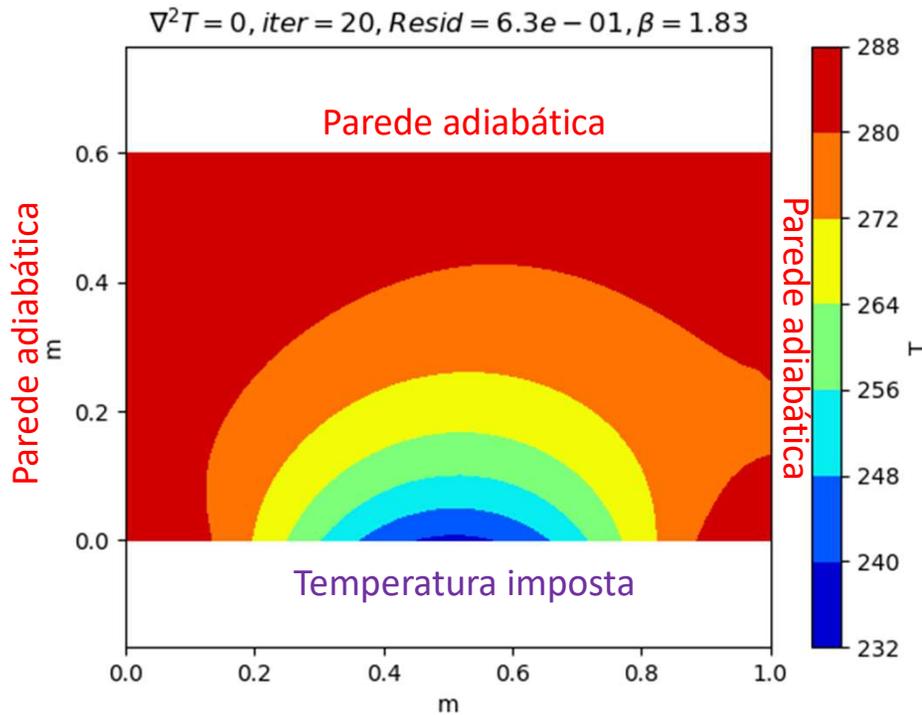
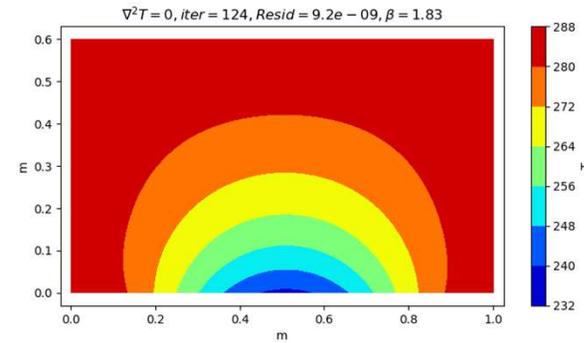
Com sobre-
relaxação
óptima

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 (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, mantém 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

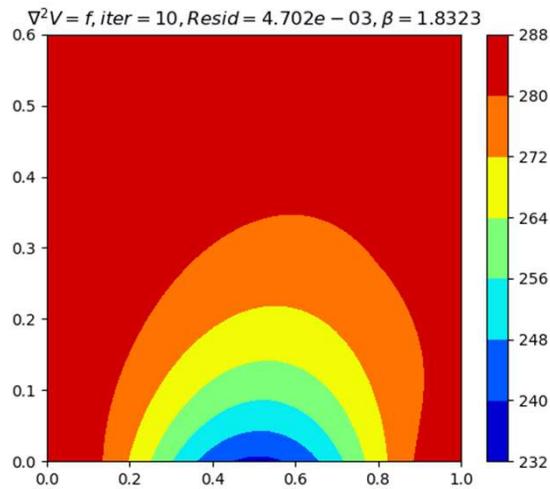
```
import numpy as np;import matplotlib.pyplot as plt
Lx=1.;Ly=1.2;M=51;N=31;delta=Lx/(M-1)
X=np.zeros((M,N));Y=np.zeros((M,N));ro=np.zeros((M,N));f=np.zeros((M,N))
for i in range(M):
    for j in range(N):
        X[i,j]=i*delta; Y[i,j]=j*delta
maxiter=3000;maxres=1.e-8 # erro relativo
V0=288.*np.ones((M,N))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

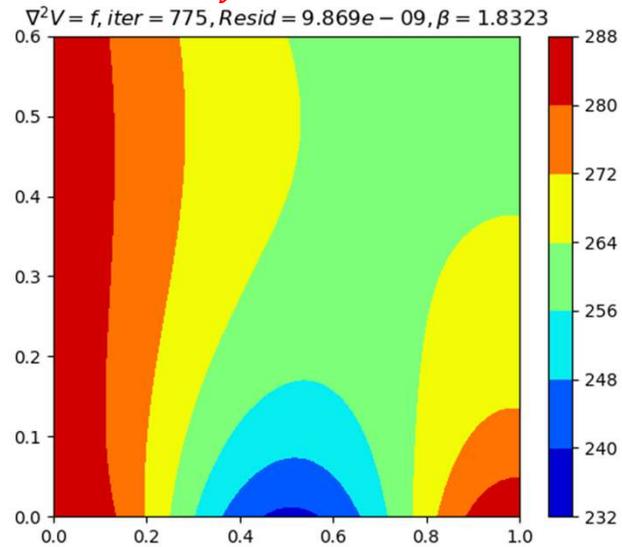
```
    V0[ix,0]=288-50*np.sin(ix*np.pi/M)**2
```

```
    BxH=np.zeros((N));ByH=-np.ones((M))*10
```

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



$$V = 288$$



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

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

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

