



Ciências
ULisboa

Modelação Numérica

Aula 16

Modelo de ondas gravíticas em águas pouco profundas (shallow-water)

O modelo shallow water

A partir da equação de Navier-Stokes para um **fluido incompressível**, pode obter-se esta equação relacionando a altura da superfície livre do fluido (h) com a velocidade horizontal media vertical (u, v), no caso de ondas **com comprimento de onda muito maior que a profundidade**

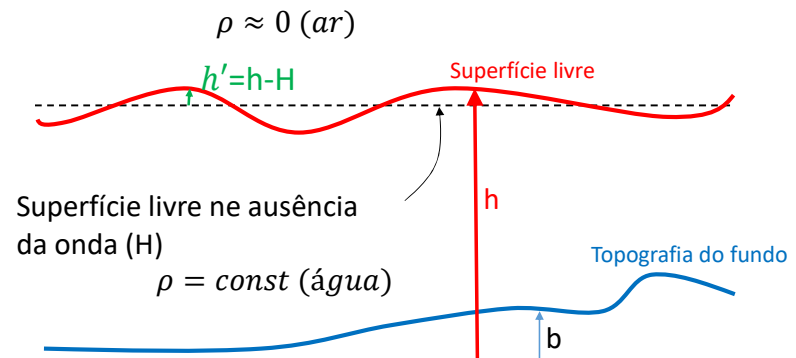
$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}(u^2) - \frac{\partial}{\partial y}(uv) - g \frac{\partial h}{\partial x}$$

$$\frac{\partial v}{\partial t} = -\frac{\partial}{\partial x}(uv) - \frac{\partial}{\partial y}(v^2) - g \frac{\partial h}{\partial y}$$

$$\frac{\partial h}{\partial t} = -\frac{\partial}{\partial x}((h-b)u) - \frac{\partial}{\partial y}((h-b)v)$$

- ✓ 3 equações
- ✓ 3 incógnitas (u, v, h)
- ✓ Não linear

Velocidade de fase das ondas:
 $c = \sqrt{gH} \gg u, v$
 Independente do comprimento de onda
 (ondas **não dispersivas**).



No oceano aberto $H \approx 5000 \text{ m}$:

$$c = \sqrt{gH} \approx 223 \text{ ms}^{-1} \approx 800 \text{ km h}^{-1}$$

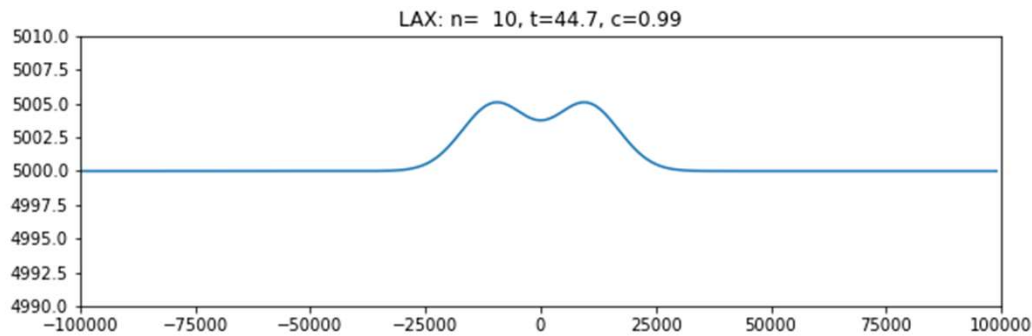
Modelo para ondas longas ($\lambda \gg 5000\text{m}$), e.g. maré, tsunamis

O modelo shallow water

A 1 dimensão:

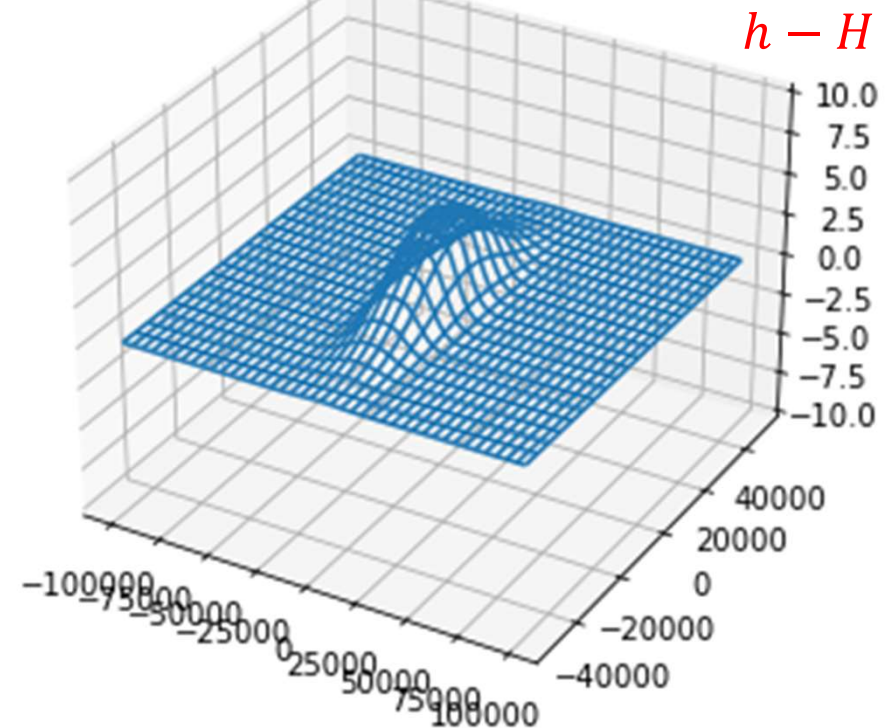
$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}(u^2) - g \frac{\partial h}{\partial x}$$

$$\frac{\partial h}{\partial t} = -\frac{\partial}{\partial x}((h-b)u)$$

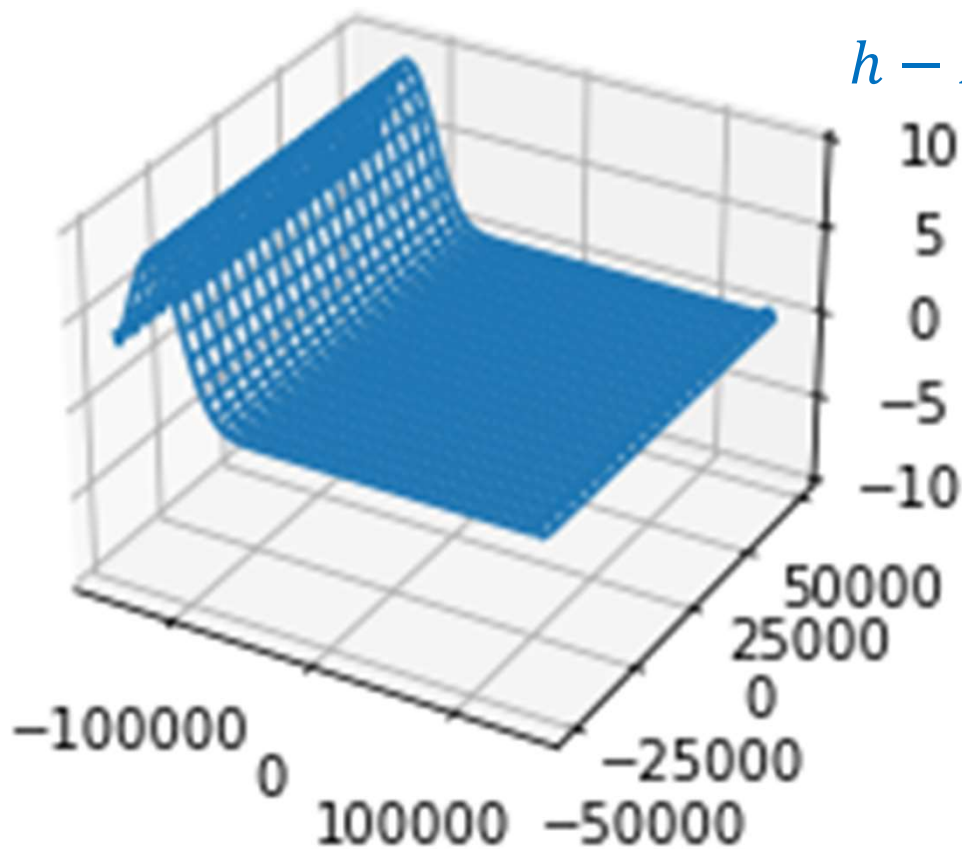


Condições fronteira **cíclicas** (ou
periódicas)

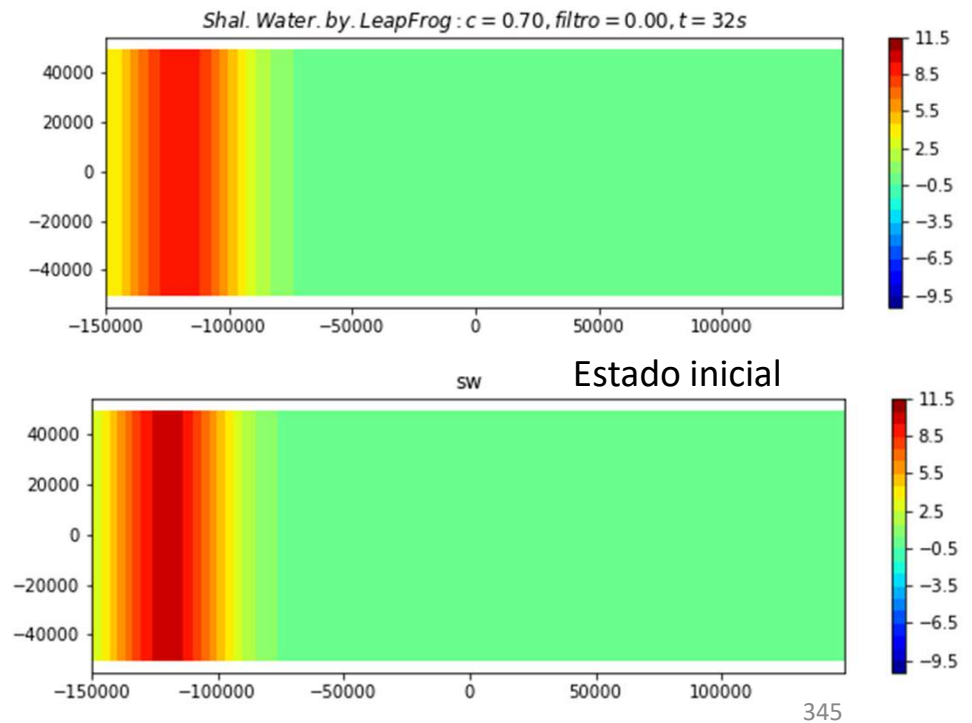
2D



Condição fronteira fechada (refletora) em x_{min}, x_{max}



em $x = 0, L_x: u = 0, \frac{\partial v}{\partial x} = 0, \frac{\partial h}{\partial x} = 0$

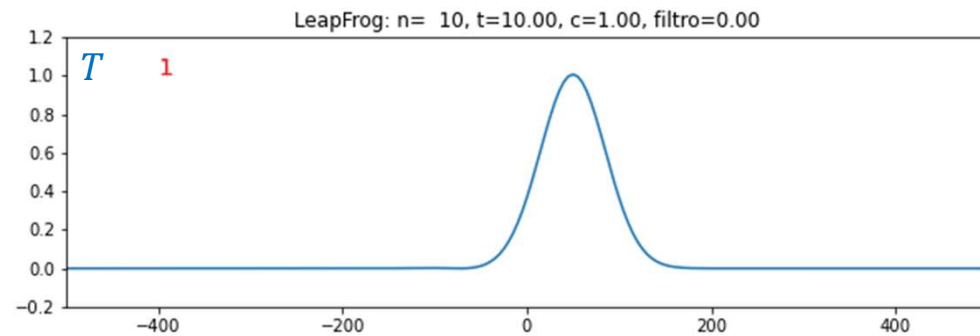


Recapitulando

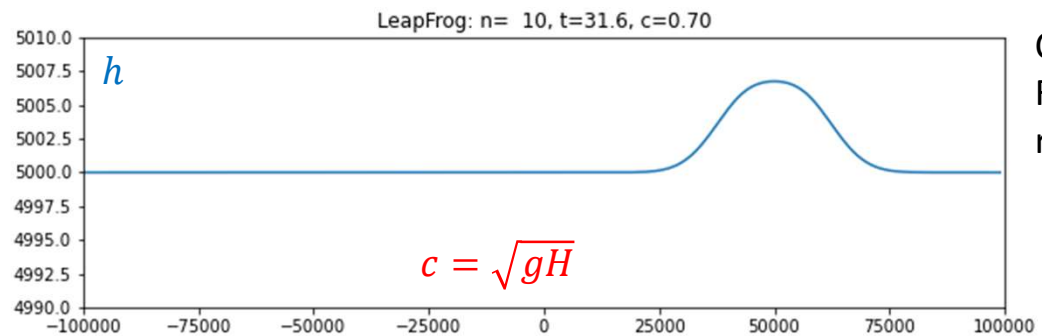
$$\frac{\partial T}{\partial t} = -u \frac{\partial T}{\partial x}$$

$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}(u^2) - g \frac{\partial h}{\partial x}$$

$$\frac{\partial h}{\partial t} = -\frac{\partial}{\partial x}((h-b)u)$$

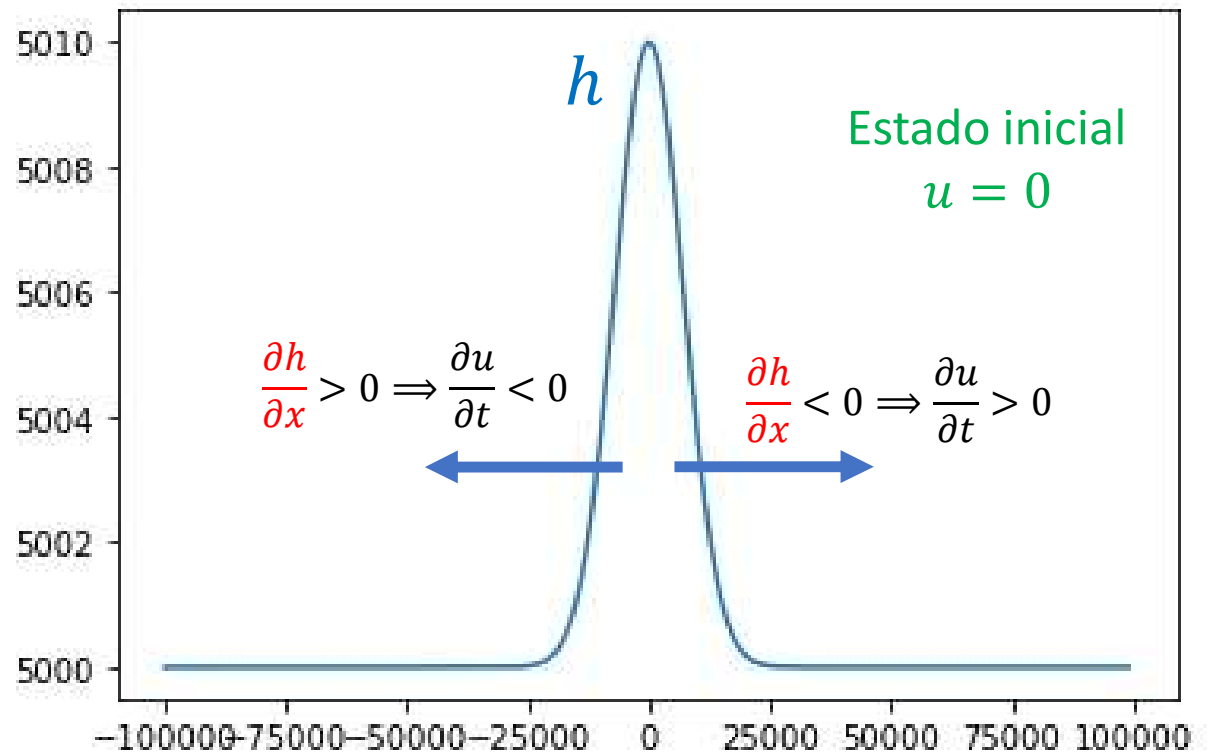
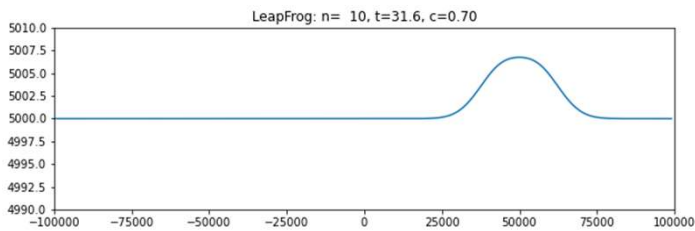


Cond.f
Fronteira
cíclicas



Cond.f
Fronteira
reflexivas

Shallow-water 1D: $\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x} (u^2) - g \frac{\partial h}{\partial x}$



1D shallow water, equação para U (leapfrog)

$$\frac{\partial u}{\partial t} = -\frac{\partial}{\partial x}(u^2) - g \frac{\partial h}{\partial x}$$

$$uu = u^2$$

$$\frac{(u_i^{n+1} - u_i^{n-1})}{2\Delta t} = -\frac{uu_{i+1}^n - uu_{i-1}^n}{2\Delta x} - g \frac{h_{i+1}^n - h_{i-1}^n}{2\Delta x}$$

$$U \equiv u_i^n; \quad UM \equiv u_i^{n-1}; \quad UP \equiv u_i^{n+1}; \quad UU = U^{**}2$$

$$UP[i] = UM[i] - dt/dx * (UU[i+1] - UU[i-1]) - g * dt/dx * (H[i+1] - H[i-1])$$

o resto é semelhante à equação de advecção linear, mas atenção à condição fronteira.

Shallow-water 1D

```
import numpy as np;import matplotlib.pyplot as plt
import os;import imageio
path=r'd:/'
g=9.81;h0=5000
nx=200;dx=1000;nt=2000;passo=20;snap=1200
closex=True
wavespeed=np.sqrt(g*h0)
courant=0.7; dt=courant*dx/wavespeed
Lx=nx/2*dx;x=np.arange(-Lx,Lx,dx)
h=np.ones(x.shape)*h0
hJUMP=10;xJUMP=-50000;LxJUMP=10000
h=h+hJUMP*np.exp(-(x-xJUMP)/LxJUMP)**2) #nivel inicial
b=np.zeros(h.shape)
b=0*np.exp(-(x-50000)/20000)**2) #batimetria
plt.figure(1,figsize=(10,10))
plt.subplot(2,1,1);plt.plot(x,h,color='red',label='0s');
plt.title('ShallowWater, c=%3.2f' % courant);plt.xlim(-Lx,Lx)
plt.subplot(2,1,2);plt.plot(x,b-h0);
plt.plot(x,np.ones(x.size)*0,color='red');plt.title('Batimetria')
plt.xlim(-Lx,Lx)
```



```

t=np.arange(0,dt*nt,dt)
frames=[]
u=np.zeros(h.shape)
uP=np.copy(u);uM=np.copy(u)
hP=np.copy(h);hM=np.copy(h)
dtdx2=0.5*dt/dx
dtdx=dt/dx
#1st step Euler
hmb=h-b;hmu=hmb*u;uu=u*u
for ix in range(nx):
    ixm=ix-1;ixp=ix+1
    if ix==0: #cíclico
        ixm=nx-1
    elif ix==nx-1:
        ixp=0
    uP[ix]=u[ix]-dtdx2*(uu[ixp]-uu[ixm])-dtdx2*g*(h[ixp]-h[ixm])
    hP[ix]=h[ix]-dtdx2*(hmu[ixp]-hmu[ixm])
if closex: #fronteira fechada
    uP[0]=0
    uP[nx-1]=0
    hP[0]=hP[1]
    hP[nx-1]=hP[nx-1]

```

```

#LeapFrog
for it in range(2,nt):
    print(it,nt)
    hmb=h-b;hmu=hmb*u;uu=u*u
    for ix in range(nx):
        ixm=ix-1;ixp=ix+1
        if ix==0: #cíclico
            ixm=nx-1
        elif ix==nx-1:
            ixp=0
        uP[ix]=uM[ix]-dtdx*(uu[ixp]-uu[ixm])-dtdx*g*(h[ixp]-h[ixm])
        hP[ix]=hM[ix]-dtdx*(hmu[ixp]-hmu[ixm])
    if closex: #fronteira fechada
        uP[0]=0
        uP[nx-1]=0
        hP[0]=hP[1]
        hP[nx-1]=hP[nx-2]
    uM=np.copy(u);u=np.copy(uP);hM=np.copy(h);h=np.copy(hP)

```

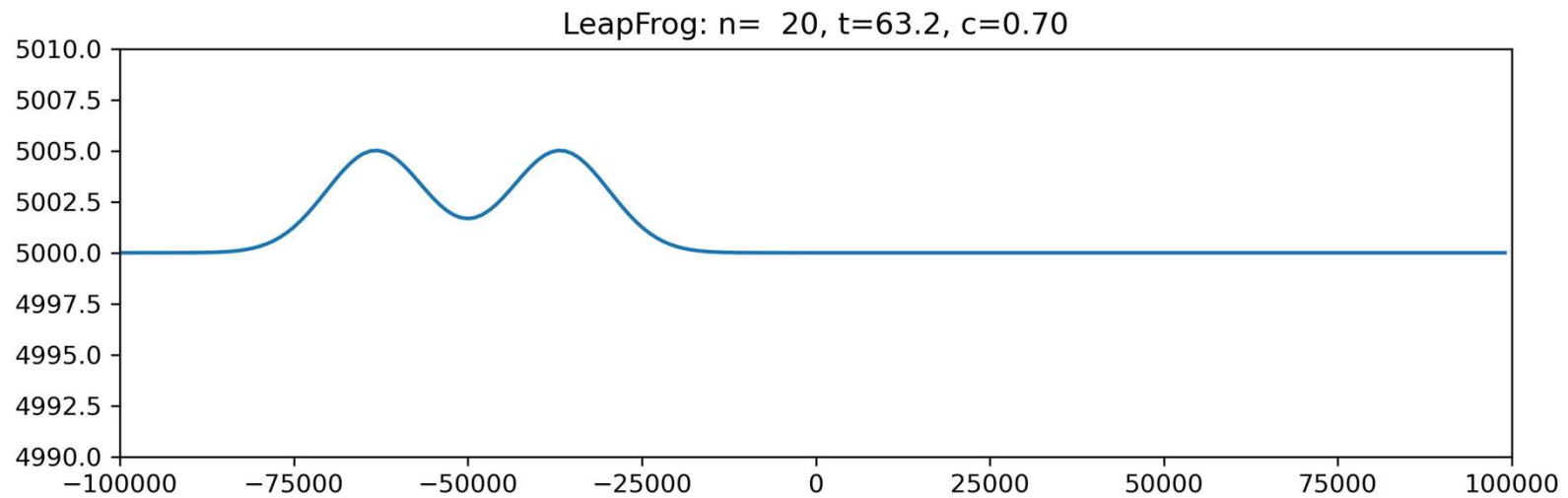
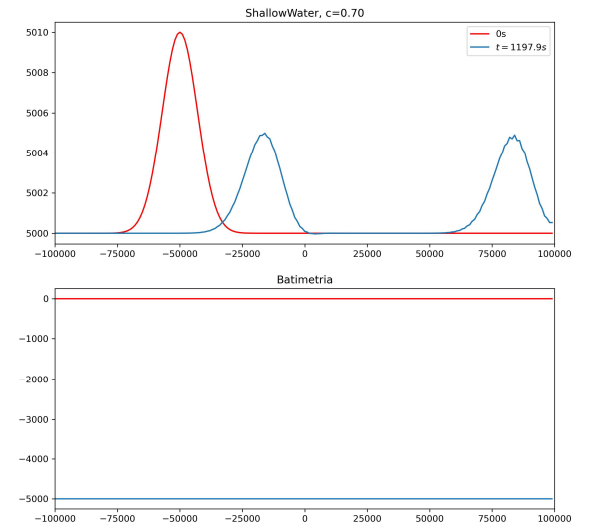
```

if it%passo==0:
    plt.figure(2,figsize=(10,3))
    plt.plot(x,h)
    plt.plot(x,b,color='black')
    plt.axis([-Lx,Lx,h0-hJUMP,h0+hJUMP])
    plt.title('LeapFrog: n=%4i, t=%4.1f, c=%3.2f' %(it,dt*it,courant))
    fn=path+'mov'+str(it)+'.png'
    plt.savefig(fn)
    plt.clf() #close figure
    frames.append(fn)
if snap>=it*dt and snap<=(it+1)*dt:
    plt.figure(1)
    plt.subplot(2,1,1);plt.plot(x,h,label=r'$t=%4.1f s$' %(it*dt))
plt.figure(1);plt.legend()

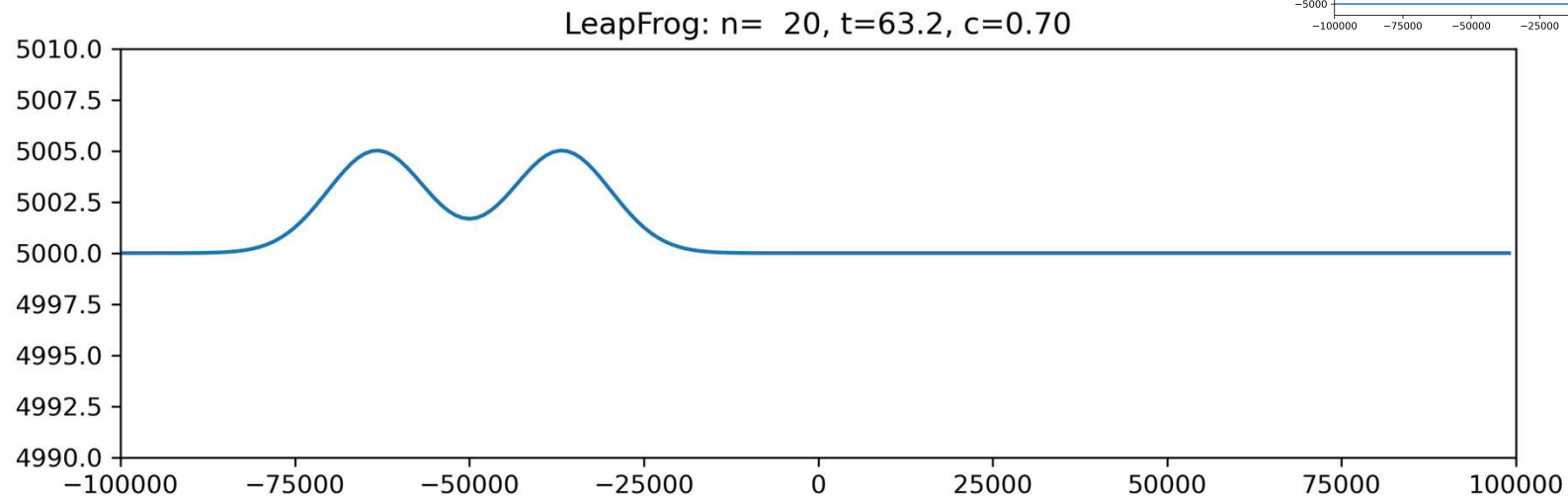
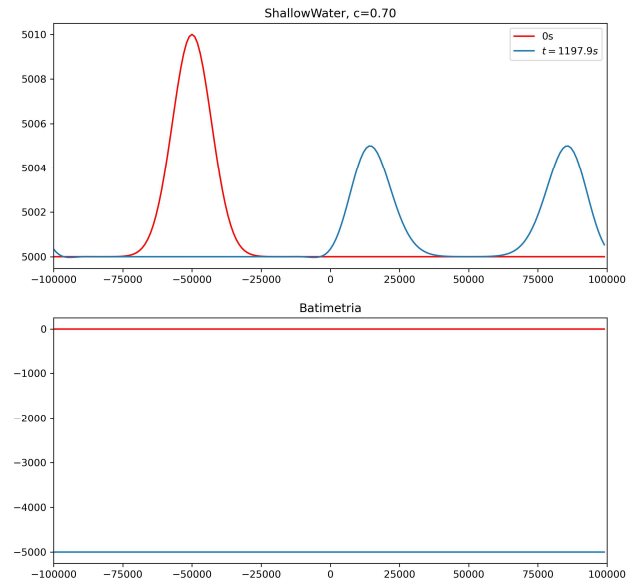
images=[] #frames para filme
for frame in frames:
    images.append(imageio.imread(frame))
    os.remove(frame)
imageio.mimsave(path+'SHA1dLFR'+'.gif', images,duration=0.1)

```

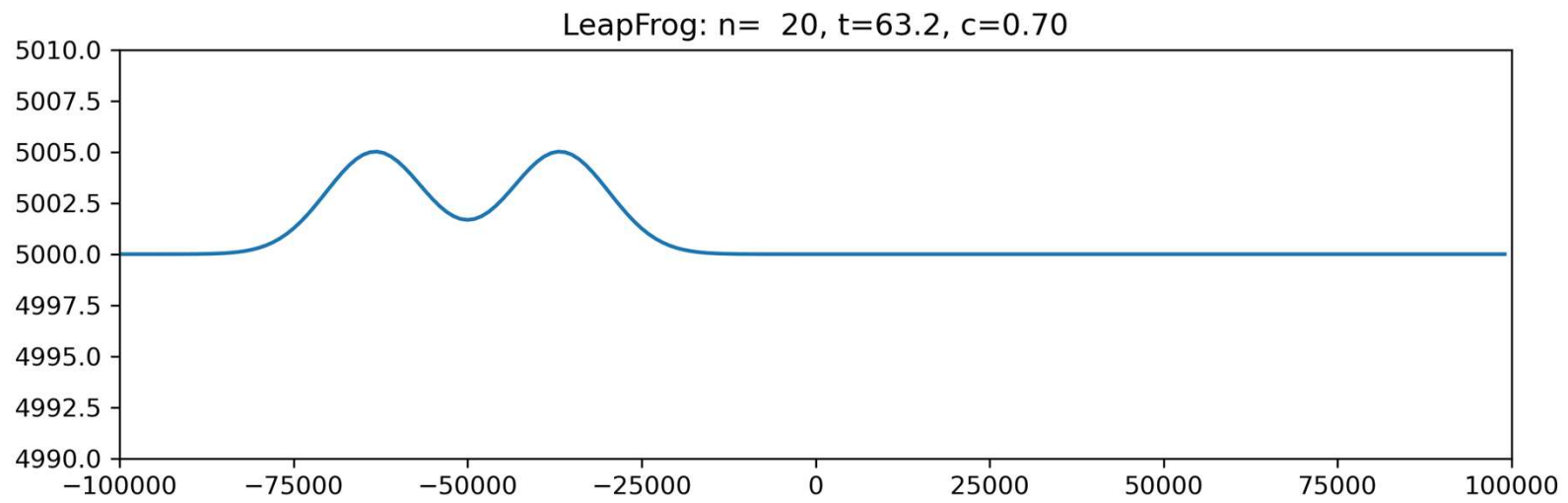
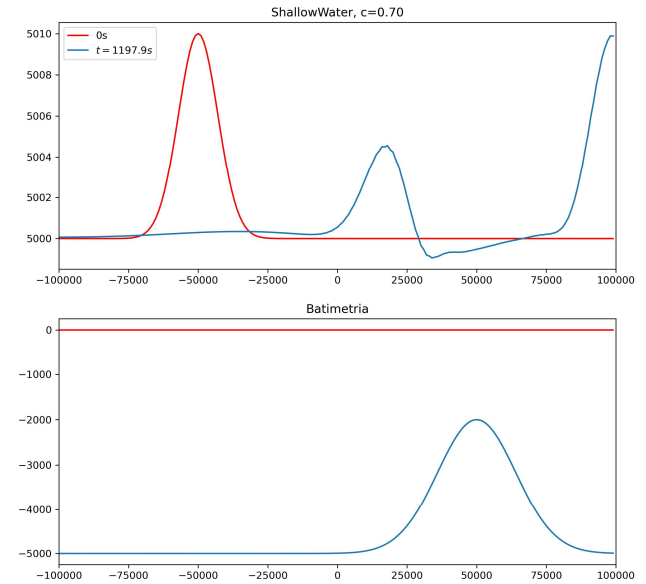
Batimetria FLAT, closeX=True



Batimetria FLAT, closeX=False



Fundo com montanha submarina
`b=3000*np.exp(-(x-50000)/20000)**2)`



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**:

$$\nabla^2 T = 0$$

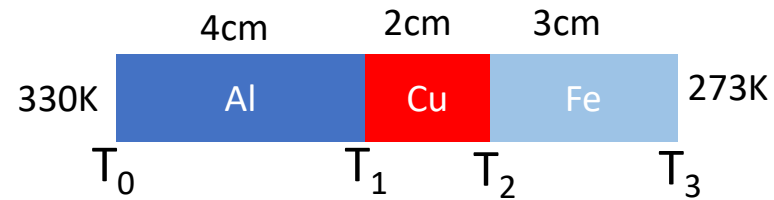
No caso unidimensional (com k variável):

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

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

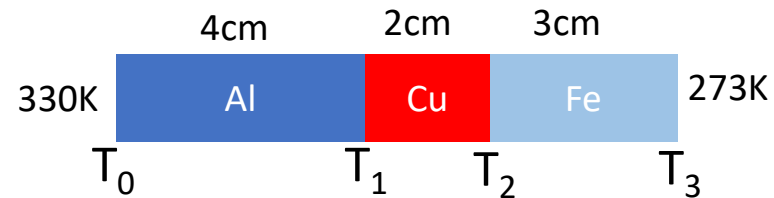


$$\text{Fluxo de calor} = k_{Al} \left(-\frac{\partial T}{\partial x} \right) \equiv k_{Al} \frac{T_0 - T_1}{x_1 - x_0} = k_{Cu} \frac{T_1 - T_2}{x_2 - x_1} = k_{Fe} \frac{T_2 - T_3}{x_3 - x_2} = \text{const}$$

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



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

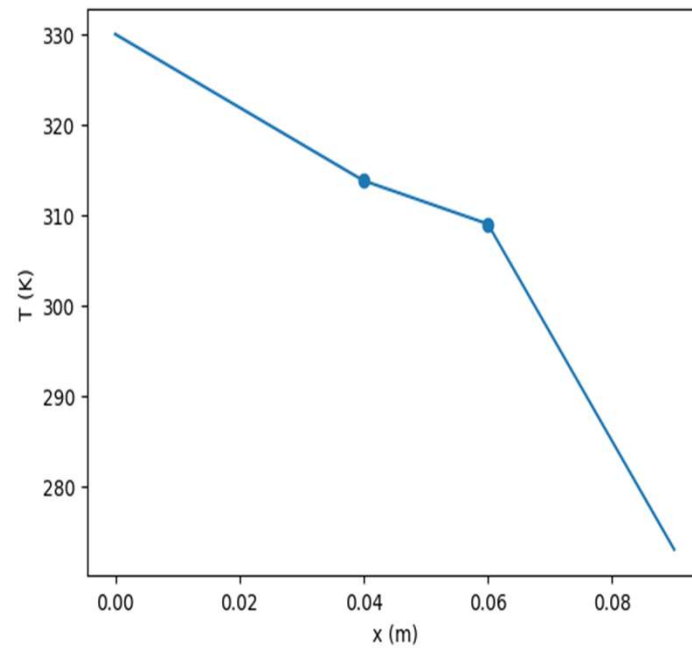
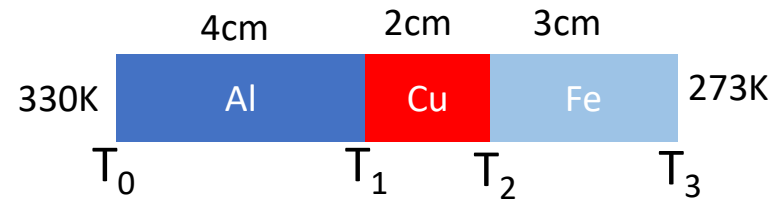
$$\begin{bmatrix} -\frac{k_{Al}}{x_1 - x_0} & -\frac{k_{Cu}}{x_2 - x_1} & \frac{k_{Cu}}{x_2 - x_1} \\ \frac{k_{Cu}}{x_2 - x_1} & -\frac{k_{Cu}}{x_2 - x_1} & -\frac{k_{Fe}}{x_3 - x_2} \end{bmatrix} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \begin{bmatrix} -\frac{k_{Al}}{x_1 - x_0} T_0 \\ -\frac{k_{Fe}}{x_3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

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)')
```

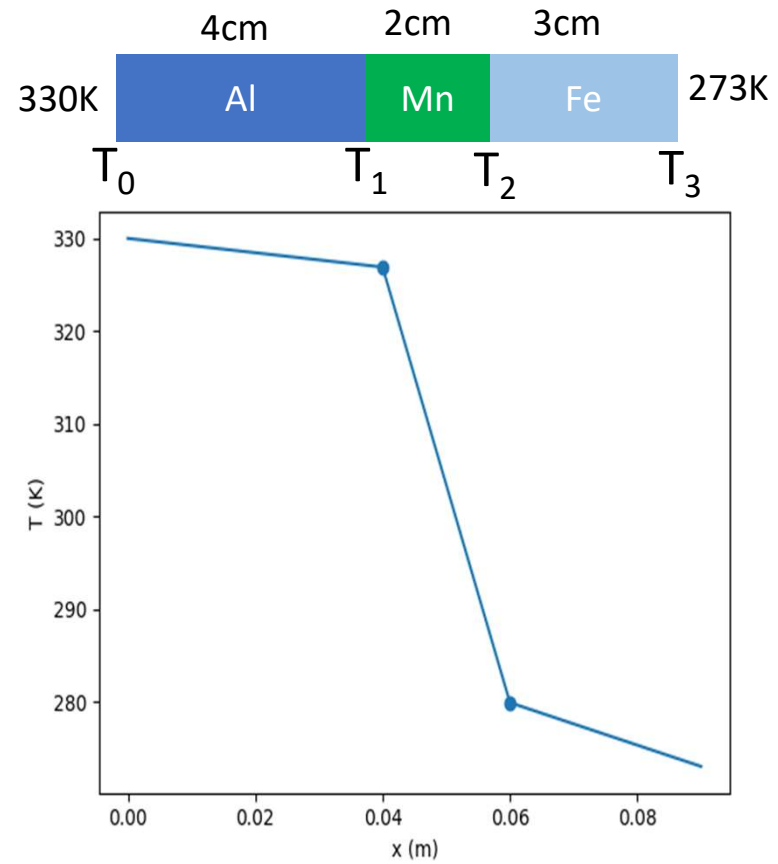
Solução

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:

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