

Modelação Numérica 2017

Aula 13, 29/Mar

- Advecção 2D
- Projecto #2

<http://modnum.ucs.ciencias.ulisboa.pt>

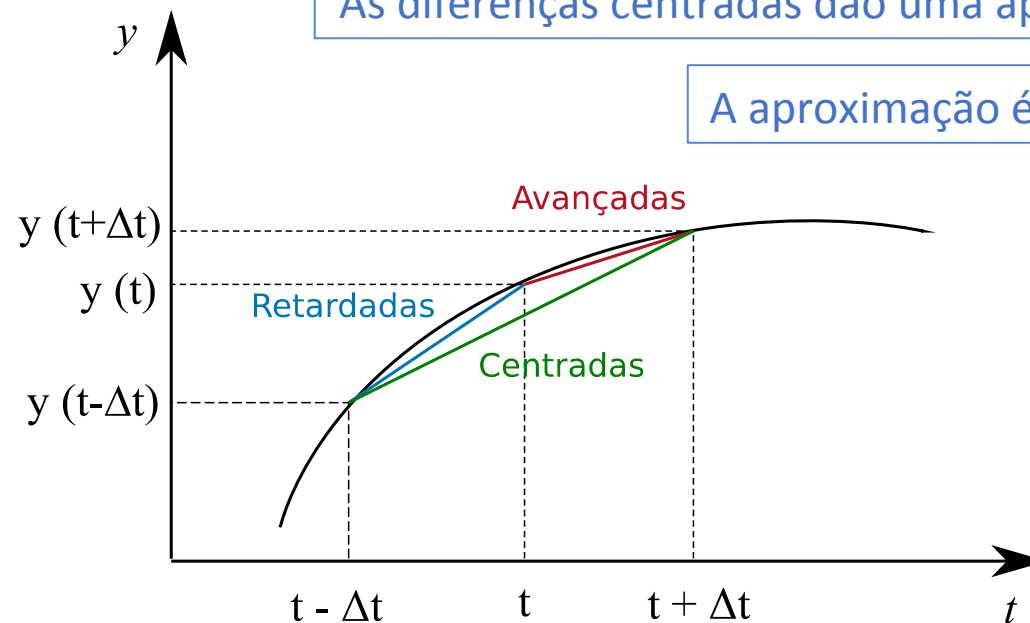
2a-f, Entregas	3a-f, T21	4a-f, T21	5a-f, PL22	5a-f, PL24	5a-f, PL23	6a-f, PL25	6a-f, PL21	
	14/Fev T1	15/Fev T2	16/Fev Intro-1	16/Fev Intro-1	16/Fev Intro-1	17/Fev Intro-1	17/Fev Intro-1	
	21/Fev T3	22/Fev T4	23/Fev Intro-2	23/Fev Intro-2	23/Fev Intro-2	24/Fev Intro-2	24/Fev Intro-2	
	28/Fev Carnaval	1/Março T5, Protoc.	2/Março Ex1-1	2/Março Ex1-1	2/Março Ex1-1	3/Março Ex1-1	3/Março Ex1-1	
	7/Março T6	8/Março T7	9/Março Ex1-2	9/Março Ex1-2	9/Março Ex1-2	10/Março Ex1-2	10/Março Ex1-2	
	14/Março T8	15/Março T9	16/Março Ex1-3	16/Março Ex1-3	16/Março Ex1-3	17/Março Ex1-3	17/Março Ex1-3	
20/Março, 17:00 Entrega Ex1	21/Março T10	22/Março T11	23/Março Ex1-4	23/Março Ex1-4	23/Março Ex1-4	24/Março Ex1-4	24/Março Ex1-4	Apresentações
	28/Março T12	29/Março T13, Protoc.	30/Março Ex2-1	30/Março Ex2-1	30/Março Ex2-1	31/Março Ex2-1	31/Março Ex2-1	
	4/Abr T14	5/Abr T15	6/Abr Ex2-2	6/Abr Ex2-2	6/Abr Ex2-2	7/Abr Ex2-2	7/Abr Ex2-2	
	11/Abr T16	12/Abr Páscoa	13/Abr Páscoa	13/Abr Páscoa	13/Abr Páscoa	14/Abr Páscoa	14/Abr Páscoa	
	18/Abr Páscoa	19/Abr Dia Ciências	20/Abr Ex2-3	20/Abr Ex2-3	20/Abr Ex2-3	21/Abr Ex2-3	21/Abr Ex2-3	
24/Abr, 17:00 Entrega Ex2	25/Abr Feriado	26/Abr T18	27/Abr Ex2-4	27/Abr Ex2-4	27/Abr Ex2-4	28/Abr Ex2-4	28/Abr Ex2-4	Apresentações
	2/Mai T19	3/Mai T20, Protoc.	4/Mai Ex3-1	4/Mai Ex3-1	4/Mai Ex3-1	5/Mai Ex3-1	5/Mai Ex3-1	
	9/Mai T21	10/Mai T22	11/Mai Ex3-2	11/Mai Ex3-2	11/Mai Ex3-2	12/Mai Ex3-2	12/Mai Ex3-2	
	16/Mai T23	17/Mai T24	18/Mai Ex3-3	18/Mai Ex3-3	18/Mai Ex3-3	19/Mai Ex3-3	19/Mai Ex3-3	
22/Mai, 17:00 Entrega Ex3	23/Mai T25	24/Mai T26	25/Mai Ex3-4	25/Mai Ex3-4	25/Mai Ex3-4	26/Mai Ex3-4	26/Mai Ex3-4	Apresentações
	30/Mai T27							

Diferenças finitas

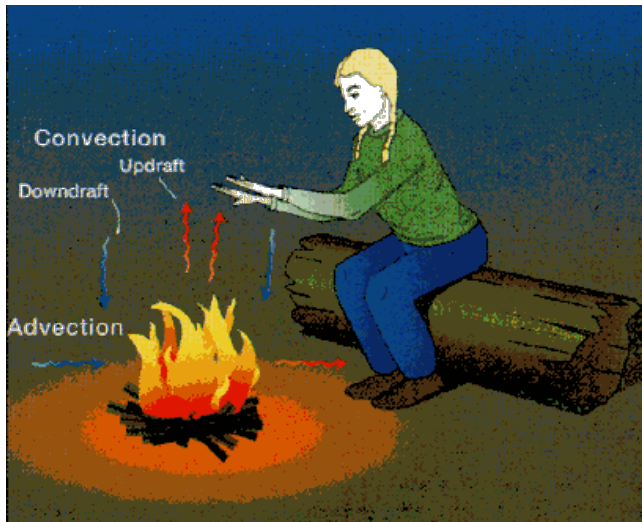
- Diferenças avançadas: $\left(\frac{\partial f}{\partial x}\right)_{x=x_0} = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} + \mathcal{O}(\Delta x)$
- Diferenças retardadas: $\left(\frac{\partial f}{\partial x}\right)_{x=x_0} = \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x} + \mathcal{O}(\Delta x)$
- Diferenças centradas: $\left(\frac{\partial f}{\partial x}\right)_{x=x_0} = \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x} + \mathcal{O}(\Delta x^2)$

As diferenças centradas dão uma aproximação mais exacta

A aproximação é melhor quando $\Delta x \rightarrow 0$



Advecção 2D



$$1D: \quad \frac{\partial h}{\partial t} = -u \frac{\partial h}{\partial x}$$

Outra notação:

$$2D: \quad \frac{\partial h}{\partial t} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y}$$

Advecção 2D

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

$$\frac{\partial h}{\partial t} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} \implies \frac{\partial h}{\partial t} = -\frac{\partial uh}{\partial x} - \frac{\partial vh}{\partial y}$$

$$\frac{\partial h}{\partial t} = -\nabla(\vec{u}h), \quad \vec{v}h \text{ é o fluxo de } h$$

Esta forma é usável em duas condições:

- $u, v = \text{constante}$ (advecção linear)
- $\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$ (fluido incompressível)

Advecção 2D

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

$$\frac{\partial h}{\partial t} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} \implies \boxed{\frac{\partial h}{\partial t} = -\frac{\partial uh}{\partial x} - \frac{\partial vh}{\partial y}}$$

Discretização no esquema de Lax:

$$\frac{h_{k,j}^{n+1} - h_{k,j}^n}{\Delta t} = -\frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

Diferença
avançada no
tempo

Diferença
centrada no
espaço em x

Diferença
centrada no
espaço em y

Advecção 2D

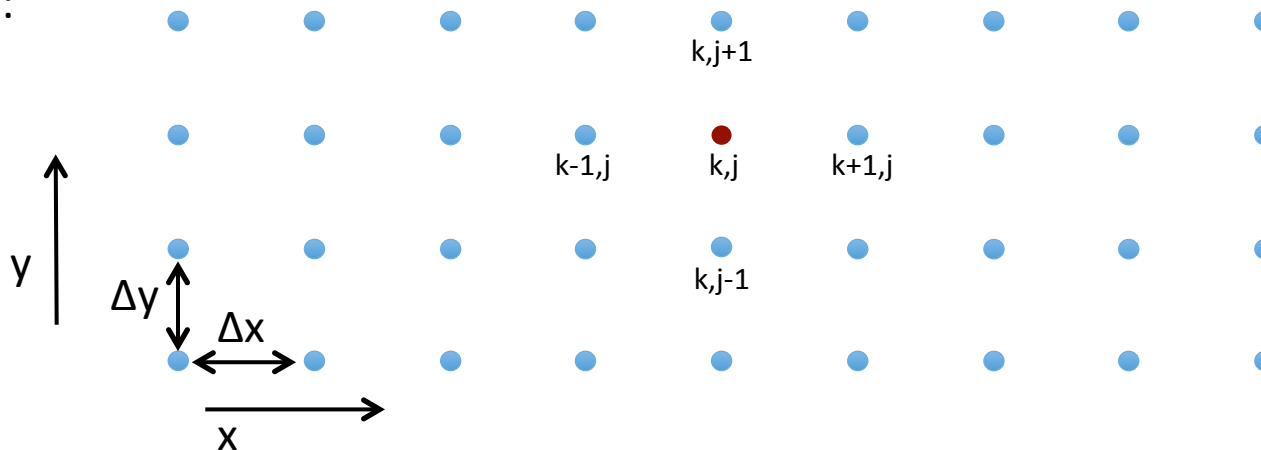
$$\frac{\partial T}{\partial t} = -u \frac{\partial T}{\partial x} - v \frac{\partial T}{\partial y}$$

$$\frac{\partial h}{\partial t} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} \implies \frac{\partial h}{\partial t} = -\frac{\partial uh}{\partial x} - \frac{\partial vh}{\partial y}$$

Discretização no esquema de Lax:

$$\frac{h_{k,j}^{n+1} - h_{k,j}^n}{\Delta t} = -\frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

t=n:



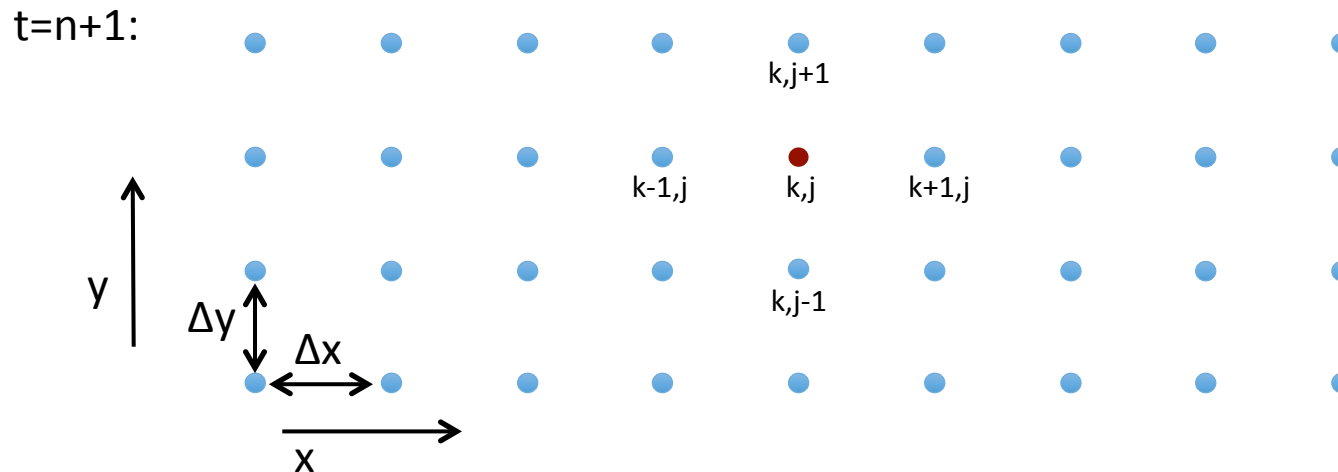
Advecção 2D

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

$$\frac{\partial h}{\partial t} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} \implies \frac{\partial h}{\partial t} = -\frac{\partial uh}{\partial x} - \frac{\partial vh}{\partial y}$$

Discretização no esquema de Lax:

$$\frac{h_{k,j}^{n+1} - h_{k,j}^n}{\Delta t} = -\frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$



Advecção 2D

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

$$\frac{\partial h}{\partial t} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} \implies \frac{\partial h}{\partial t} = -\frac{\partial uh}{\partial x} - \frac{\partial vh}{\partial y}$$

Discretização no esquema de Lax:

$$\frac{h_{k,j}^{n+1} - h_{k,j}^n}{\Delta t} = -\frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

$$h_{k,j}^{n+1} - h_{k,j}^n = -\Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

$$h_{k,j}^{n+1} = h_{k,j}^n - \Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

$$h_{k,j}^{n+1} = \frac{1}{4}(h_{k-1,j}^n + h_{k+1,j}^n + h_{k,j-1}^n + h_{k,j+1}^n) - \Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

Advecção 2D

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
```

```
plt.rcParams['figure.figsize'] = 10, 6
```

```
## Parâmetros
```

```
# discretização
```

```
nt=2000; nx=101; ny=101
passo=10;
dx=1000.; dy=1000.;
x = np.arange(0,nx)*dx
y = np.arange(0,ny)*dy
```

```
xmin=min(x); xmax=max(x)
ymin=min(y); ymax=max(y)
```

```
xx=np.zeros([nx,ny])
yy=np.zeros([nx,ny])
for ix in range(nx):
    for iy in range(ny):
        xx[ix,iy] = x[ix]
        yy[ix,iy] = y[iy]
```

```
# estação
```

```
ixStation = nx-2
iyStation = ny/2
hStation = np.zeros(nt)
```

```
# número de pontos no tempo e espaço (x e y)
# intervalo ebtre figuras
# espaçamento dos pontos no espaço (x e y)
# vector de pontos no espaço (x)
# vector de pontos no espaço (y)
```

```
# valor mínimo e máximo do vector x
# valor mínimo e máximo do vector y
```

```
# matriz 2D de valores x (posição em x)
# matriz 2D de valores y (posição em y)
```

```
# posição da estação, coordenada x
# posição da estação, coordenada y
# valores da variável h na estação
```

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

Advecção 2D

Aula passada

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

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
from matplotlib import cm
```

```
plt.rcParams['figure.figsize'] = 10, 6
```

```
#%% Parâmetros
```

```
# discretização
```

```
nt=2000; nx=101; ny=101
passo=10;
dx=1000.; dy=1000.;
x = np.arange(0,nx)*dx
y = np.arange(0,ny)*dy
```

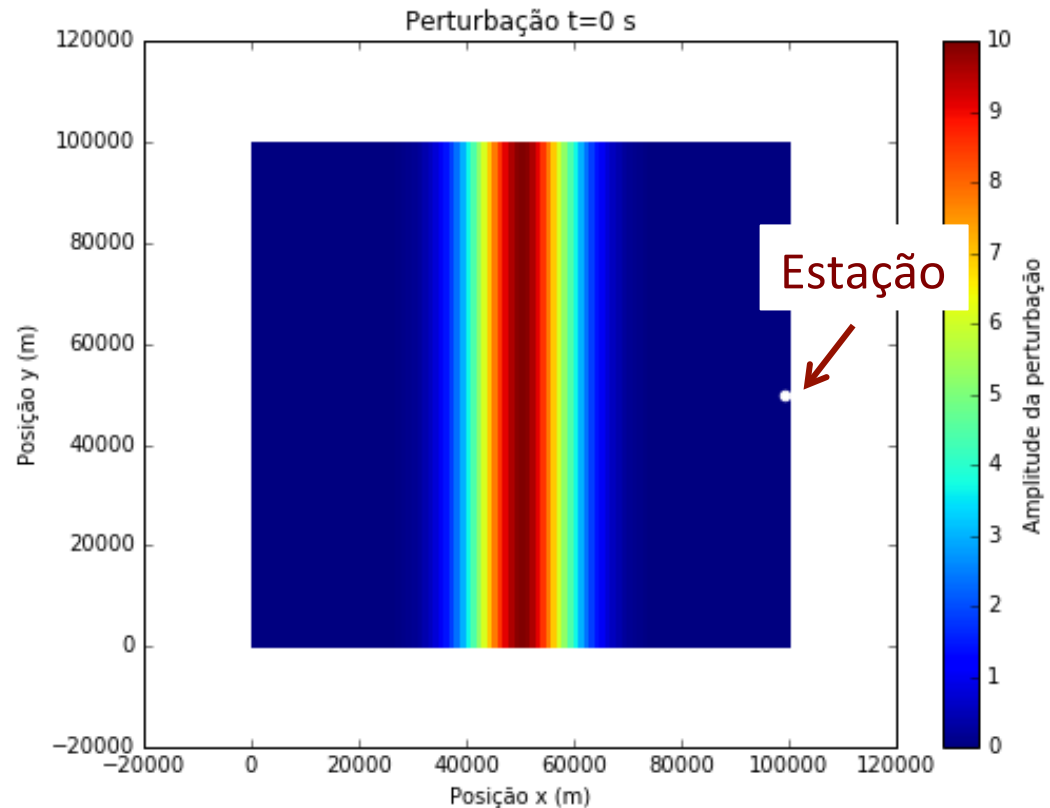
```
xmin=min(x); xmax=max(x)
ymin=min(y); ymax=max(y)
```

```
xx=np.zeros([nx,ny])
yy=np.zeros([nx,ny])
for ix in range(nx):
    for iy in range(ny):
        xx[ix,iy] = x[ix]
        yy[ix,iy] = y[iy]
```

```
# estação
```

```
ixStation = nx-2
iyStation = ny/2
hStation = np.zeros(nt)
```

← Sinal na estação



Advecção 2D

Aula passada

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

```
# velocidades
uspeed=10
vspeed=0
ws = np.sqrt(uspeed**2 + vspeed**2)
```

```
dt = 0.68 * dx / ws
dt2dx = dt/(2*dx)
dt2dy = dt/(2*dy)
```

→ courant = ws*dt/dx

```
u = uspeed * np.ones([nx,ny])
uP = np.zeros([nx,ny])
uP[:] = u[:]
```

```
v = vspeed * np.ones([nx,ny])
vP = np.zeros([nx,ny])
vP[:] = v[:]
```

```
# velocidade de propagação da direcção x
# velocidade de propagação da direcção y
# velocidade de propagação total
```

```
# espaçamento entre pontos no tempo
# dt/(2 dx)
# dt/(2 dy)
```

```
# número de courant
```

```
# matriz de velocidades (x)
# matriz de velocidades futuras (x)
# inicialização da matriz de velocidades futuras (x)
```

```
# matriz de velocidades (y)
# matriz de velocidades futuras (y)
# inicialização da matriz de velocidades futuras (y)
```

$$h_{k,j}^{n+1} = \frac{1}{4} (h_{k-1,j}^n + h_{k+1,j}^n + h_{k,j-1}^n + h_{k,j+1}^n) - \frac{\Delta t}{2\Delta x} (u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n) - \frac{\Delta t}{2\Delta y} (v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n)$$

Advecção 2D

$$h = h_0 e^{-\left(\frac{x-x_0}{L_x}\right)^2 - \left(\frac{y-y_0}{L_y}\right)^2}$$

```
%% Definição de h inicial (perturbação/sinal a propagar/advector)
```

```
hJUMP=10 # amplitude inicial do sinal
xJUMP=(xmax+xmin)/2. # localização inicial (em x) do sinal (= valor médio/central do vector)
LxJUMP=10.*dx # largura inicial do sinal (x)
yJUMP=(ymax+ymin)/2. # localização inicial (em y) do sinal (= valor médio/central do vector)
LyJUMP=2000.*dy # largura inicial do sinal (y)
```

```
# perturbação/sinal inicial:
```

```
h = hJUMP*np.exp(-((xx-xJUMP)/LxJUMP)**2 - ((yy-yJUMP)/LyJUMP)**2)
hP = np.zeros([nx,ny]) # sinal no passo temporal seguinte
hStation[0] = h[ixStation, iyStation] # localização da estação (x,y)
```

```
# plot
```

```
plt.close()
plt.rcParams['figure.figsize'] = 8, 6
```

```
# plt.contourf(xx,yy,h)
```

```
plt.pcolor(xx,yy,h) # figura 2D do sinal inicial
plt.clim(0,10) # limites da barra de cores
cb=plt.colorbar()
cb.set_label(u'Amplitude da perturbação') # legenda da barra de cores
```

```
# marcar a estação com um ponto branco
```

```
plt.scatter(xx[ixStation, iyStation], yy[ixStation, iyStation], c='w', edgecolors='w')
```

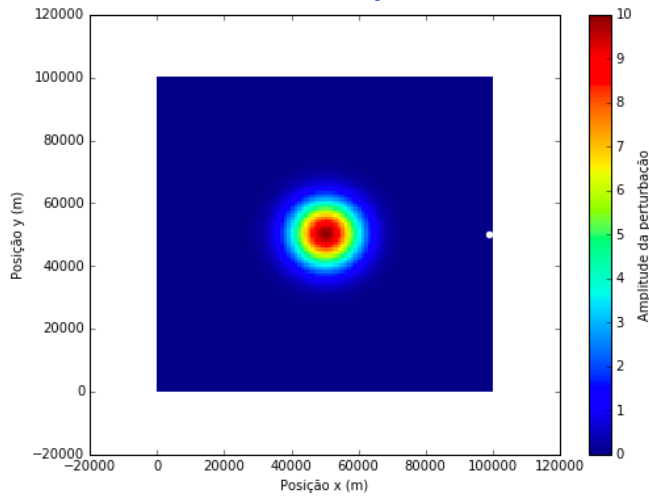
```
plt.xlabel(u'Posição x (m)')
plt.ylabel(u'Posição y (m)')
plt.title(u'Perturbação t=0 s')
```

```
plt.savefig('fig/advection2d-lax-t0.png')
```

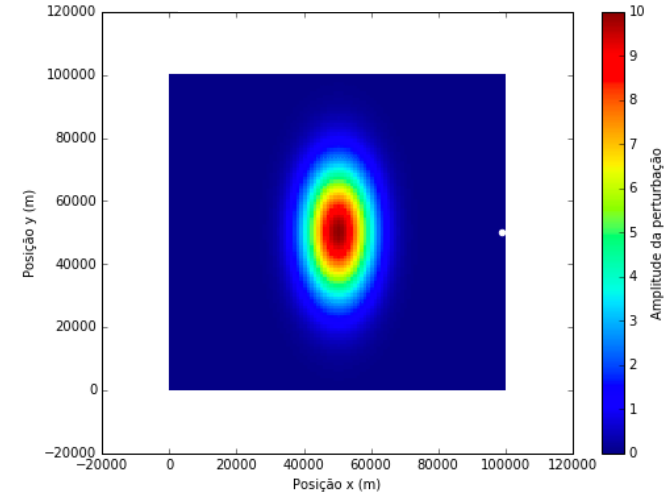
Advecção 2D

$$h = h_0 e^{-\left(\frac{x-x_0}{L_x}\right)^2 - \left(\frac{y-y_0}{L_y}\right)^2}$$

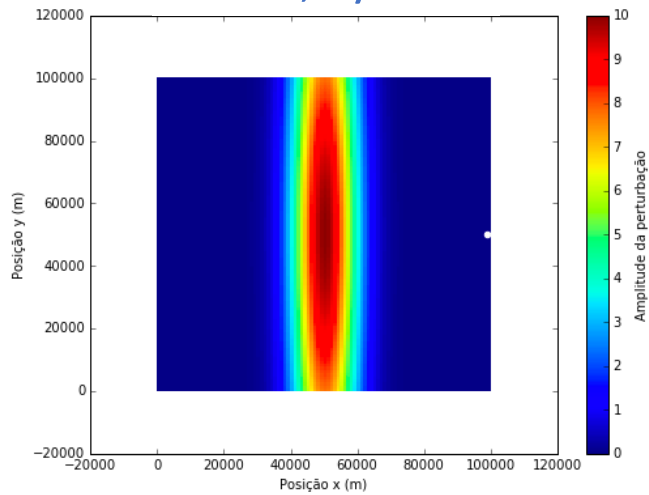
$L_x=10, L_y=10$



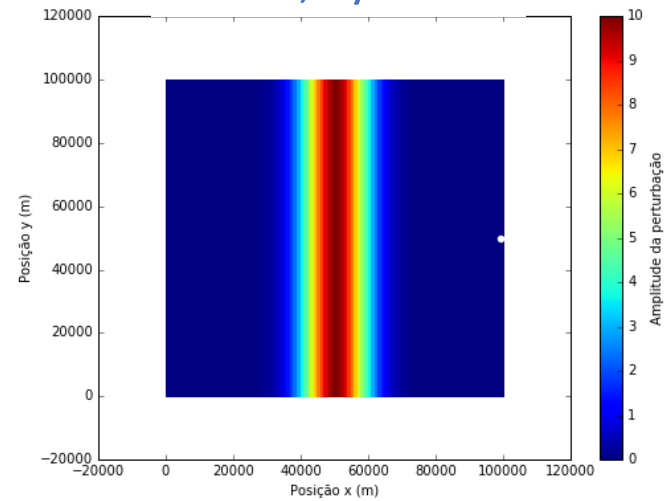
$L_x=10, L_y=20$



$L_x=10, L_y=100$



$L_x=10, L_y=1000$



Advecção 2D

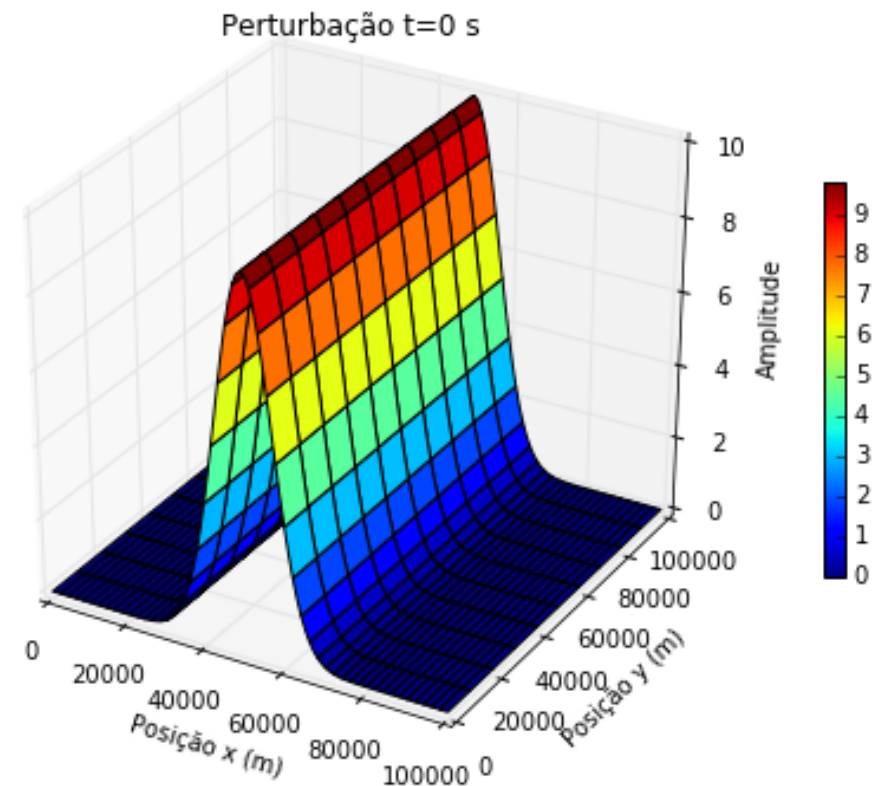
```
#%% plot3D
```

```
fig = plt.figure()  
ax = plt.axes(projection='3d')
```

```
surf=ax.plot_surface(xx,yy,h, rstride=2, cstride=10, cmap=cm.jet)  
ax.set_zlim(0,10)  
fig.colorbar(surf, shrink=0.5)
```

```
ax.set_xlabel(u'Posição x (m)')  
ax.set_ylabel(u'Posição y (m)')  
ax.set_zlabel(u'Amplitude')  
ax.set_title(u'Perturbação t=0 s')
```

```
plt.savefig('fig3d/advection2d-lax-t0.png')
```



$$\frac{\partial h}{\partial t} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} \implies \boxed{\frac{\partial h}{\partial t} = -\frac{\partial uh}{\partial x} - \frac{\partial vh}{\partial y}}$$

$$h_{k,j}^{n+1} = \frac{1}{4}(h_{k-1,j}^n + h_{k+1,j}^n + h_{k,j-1}^n + h_{k,j+1}^n) - \Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

%% Implementação do método LAX, propagar o sinal no tempo

```
for it in range(1,nt):
```

```
    hu = h*u
```

```
    hv = h*v
```

Avançar o sinal em todo o domínio excepto nas fronteiras

```
for ix in range(1,nx-1):
```

```
    for iy in range(1,ny-1):
```

```
        hP[ix,iy] = (h[ix-1,iy] + h[ix+1,iy] + h[ix,iy-1] + h[ix,iy+1])/4. \
```

```
                - dt2dx * (hu[ix+1,iy] - hu[ix-1,iy]) \
```

```
                - dt2dy * (hv[ix,iy+1] - hv[ix,iy-1])
```

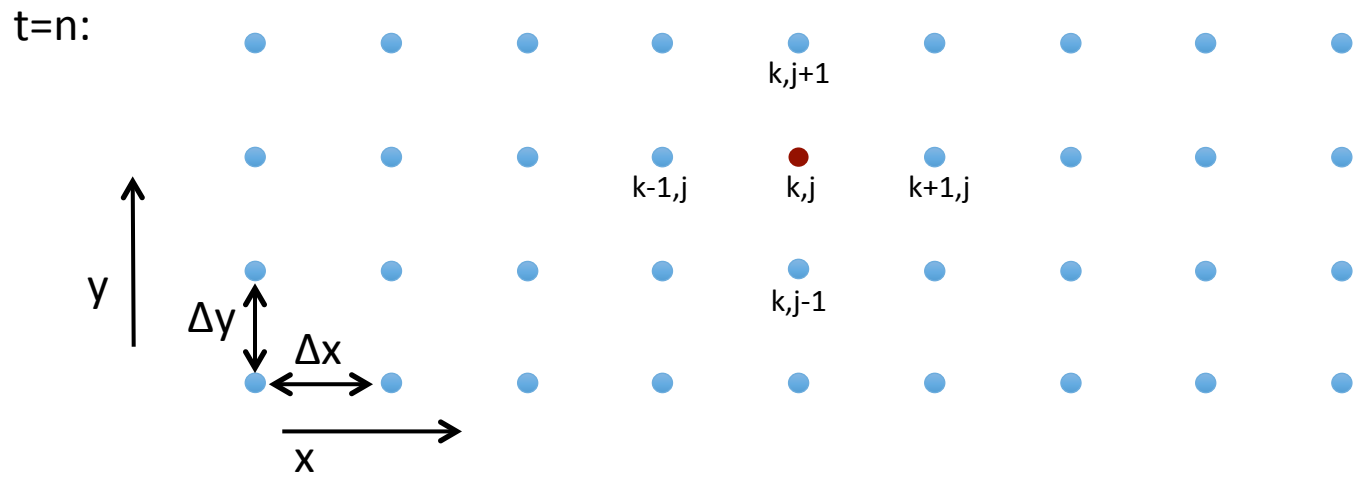

Advecção 2D

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

$$\frac{\partial h}{\partial t} = -u \frac{\partial h}{\partial x} - v \frac{\partial h}{\partial y} \implies \boxed{\frac{\partial h}{\partial t} = -\frac{\partial uh}{\partial x} - \frac{\partial vh}{\partial y}}$$

Discretização no esquema de Lax:

$$\frac{h_{k,j}^{n+1} - h_{k,j}^n}{\Delta t} = -\frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$



$$h_{k,j}^{n+1} = \frac{1}{4}(h_{k-1,j}^n + h_{k+1,j}^n + h_{k,j-1}^n + h_{k,j+1}^n) - \Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

Condições fronteira cíclicas em x

for iy in range(1,ny-1):

ix=0

hP[ix,iy] = (h[nx-1,iy] + h[ix+1,iy] + h[ix,iy-1] + h[ix,iy+1])/4. \
- dt2dx * (hu[ix+1,iy] - hu[nx-1,iy]) \
- dt2dy * (hv[ix,iy+1] - hv[ix,iy-1])

ix=nx-1

hP[ix,iy] = (h[ix-1,iy] + h[0,iy] + h[ix,iy-1] + h[ix,iy+1])/4. \
- dt2dx * (hu[0,iy] - hu[ix-1,iy]) \
- dt2dy * (hv[ix,iy+1] - hv[ix,iy-1])

Condições fronteira cíclicas em y

for ix in range(1,nx-1):

iy=0

hP[ix,iy] = (h[ix-1,iy] + h[ix+1,iy] + h[ix,ny-1] + h[ix,iy+1])/4. \
- dt2dx * (hu[ix+1,iy] - hu[ix-1,iy]) \
- dt2dy * (hv[ix,iy+1] - hv[ix,ny-1])

iy=ny-1

hP[ix,iy] = (h[ix-1,iy] + h[ix+1,iy] + h[ix,iy-1] + h[ix,0])/4. \
- dt2dx * (hu[ix+1,iy] - hu[ix-1,iy]) \
- dt2dy * (hv[ix,0] - hv[ix,iy-1])

$$h_{k,j}^{n+1} = \frac{1}{4}(h_{k-1,j}^n + h_{k+1,j}^n + h_{k,j-1}^n + h_{k,j+1}^n) - \Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

Condições fronteira nos cantos

`ix=0; ixm=nx-1; ixp=1`

`iy=0; iym=ny-1; iyp=1`

`hP[ix,iy] = 1./4. * (h[ixm,iy] + h[ixp,iy] + h[ix,iym] + h[ix,iyp]) \`
`- dt2dx * (hu[ixp,iy] - hu[ixm,iy]) \`
`- dt2dy * (hv[ix,iyp] - hv[ix,iym])`

`ix=nx-1; ixm=nx-2; ixp=0`

`iy=0; iym=ny-1; iyp=1`

`hP[ix,iy] = 1./4. * (h[ixm,iy] + h[ixp,iy] + h[ix,iym] + h[ix,iyp]) \`
`- dt2dx * (hu[ixp,iy] - hu[ixm,iy]) \`
`- dt2dy * (hv[ix,iyp] - hv[ix,iym])`

`ix=0; ixm=nx-1; ixp=1`

`iy=ny-1; iym=ny-2; iyp=0`

`hP[ix,iy] = 1./4. * (h[ixm,iy] + h[ixp,iy] + h[ix,iym] + h[ix,iyp]) \`
`- dt2dx * (hu[ixp,iy] - hu[ixm,iy]) \`
`- dt2dy * (hv[ix,iyp] - hv[ix,iym])`

`ix=nx-1; ixm=nx-2; ixp=0`

`iy=ny-1; iym=ny-2; iyp=0`

`hP[ix,iy] = 1./4. * (h[ixm,iy] + h[ixp,iy] + h[ix,iym] + h[ix,iyp]) \`
`- dt2dx * (hu[ixp,iy] - hu[ixm,iy]) \`
`- dt2dy * (hv[ix,iyp] - hv[ix,iym])`

$$h_{k,j}^{n+1} = \frac{1}{4}(h_{k-1,j}^n + h_{k+1,j}^n + h_{k,j-1}^n + h_{k,j+1}^n) - \Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

```

# atualizar o sinal
h[:] = hP[:]

# Fazer figuras com as soluções, de 10 em 10 passos no tempo
if (it+1)%passo==0:
#   if it<3*passo:
#       plt.close()
#       plt.rcParams['figure.figsize'] = 8, 6

#       plt.contourf(xx,yy,h)
#       plt.pcolor(xx,yy,h)
#       plt.clim(0,10)
#       cb=plt.colorbar()
#       cb.set_label(u'Amplitude da perturbação')
#       plt.scatter(xx[ixStation, iyStation], yy[ixStation, iyStation], c='w', edgecolors='w')

#       plt.xlabel(u'Posição x (m)')
#       plt.ylabel(u'Posição y (m)')
#       plt.title(u'Perturbação t='+ str(it*dt) + ' s')

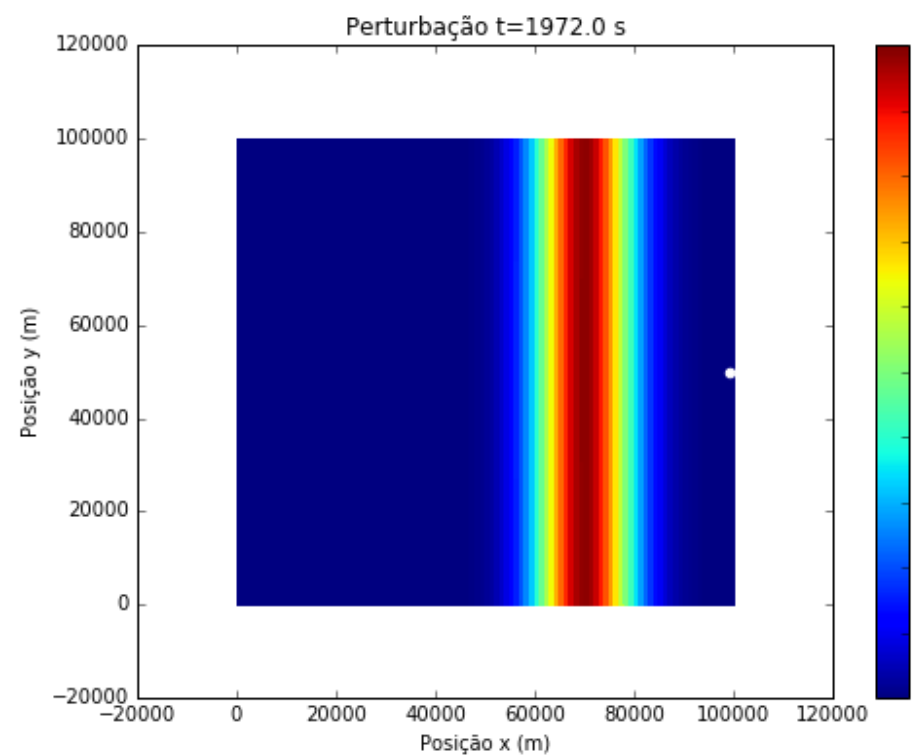
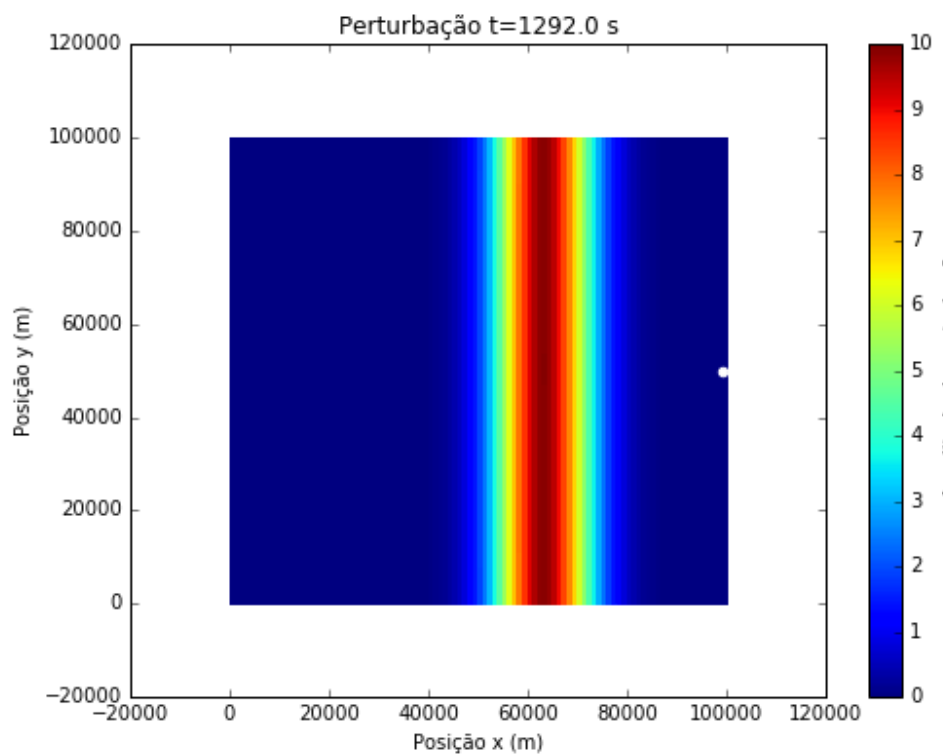
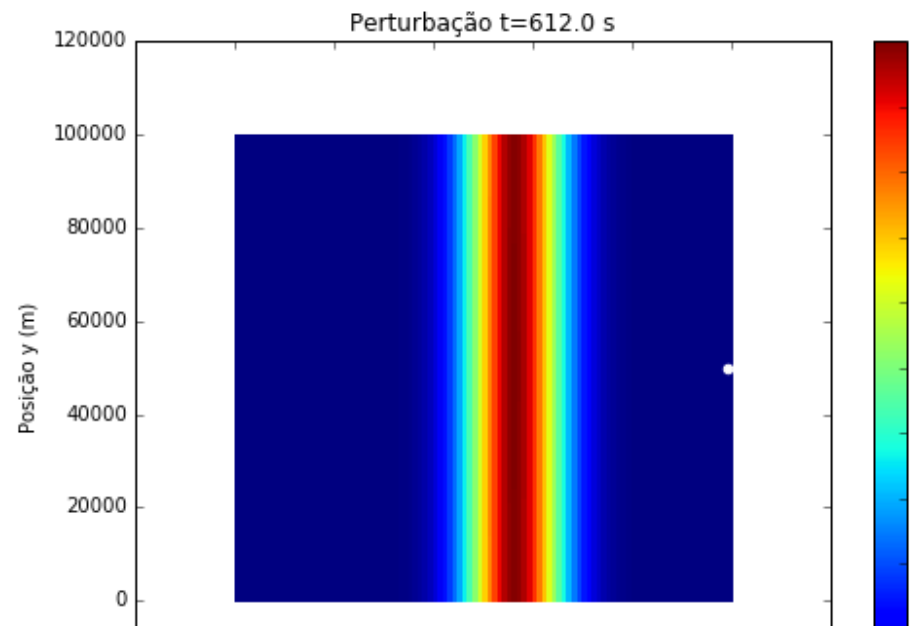
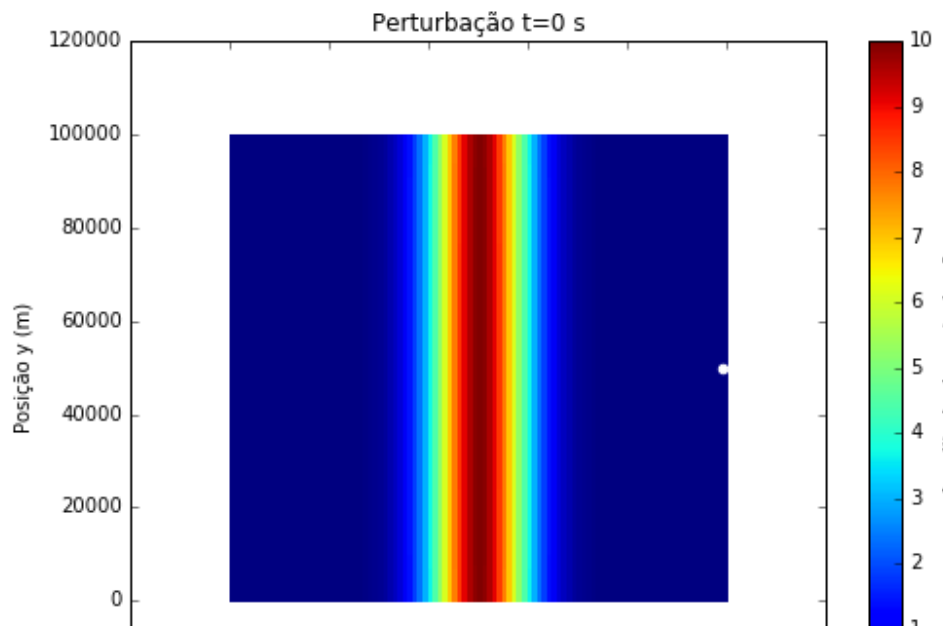
#       plt.savefig('fig/advection2d-lax-t'+ str(it) + '.png')

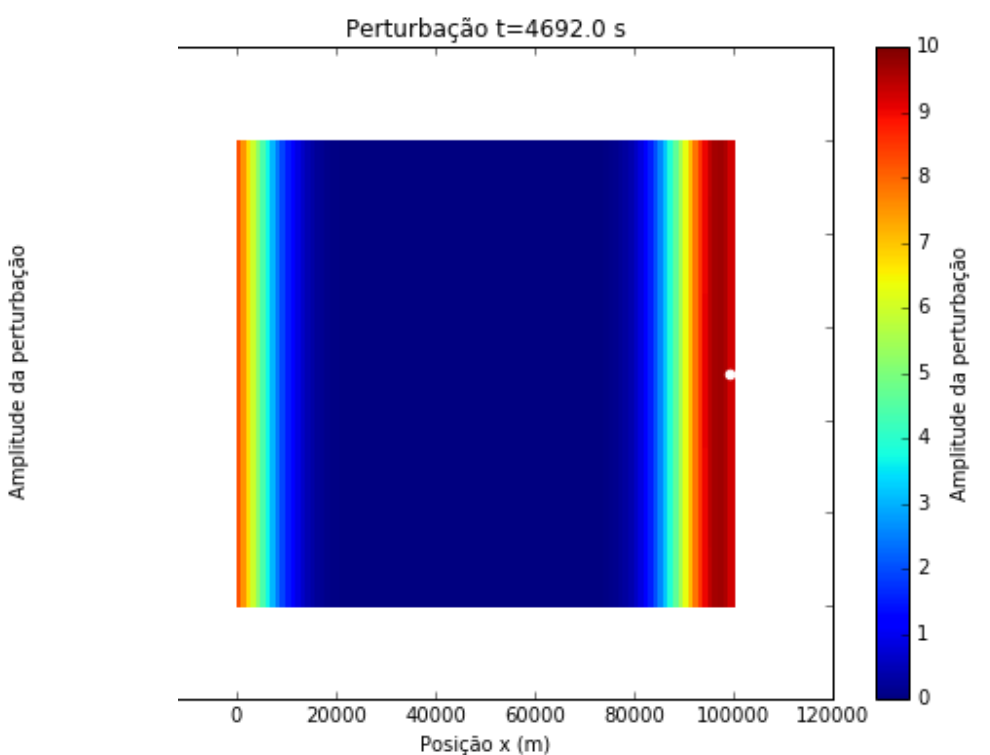
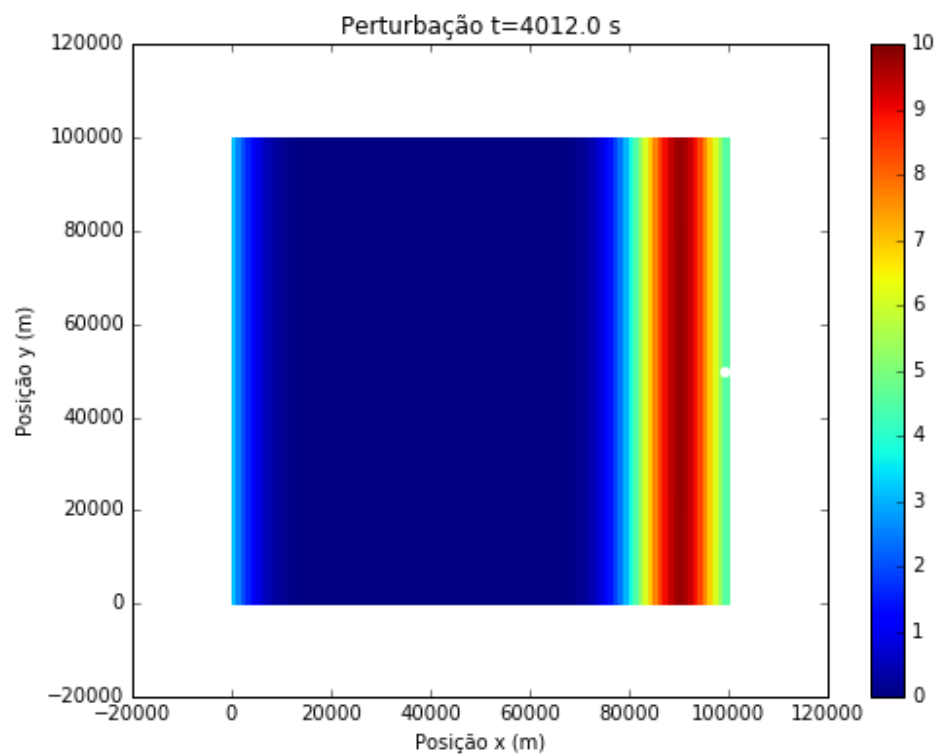
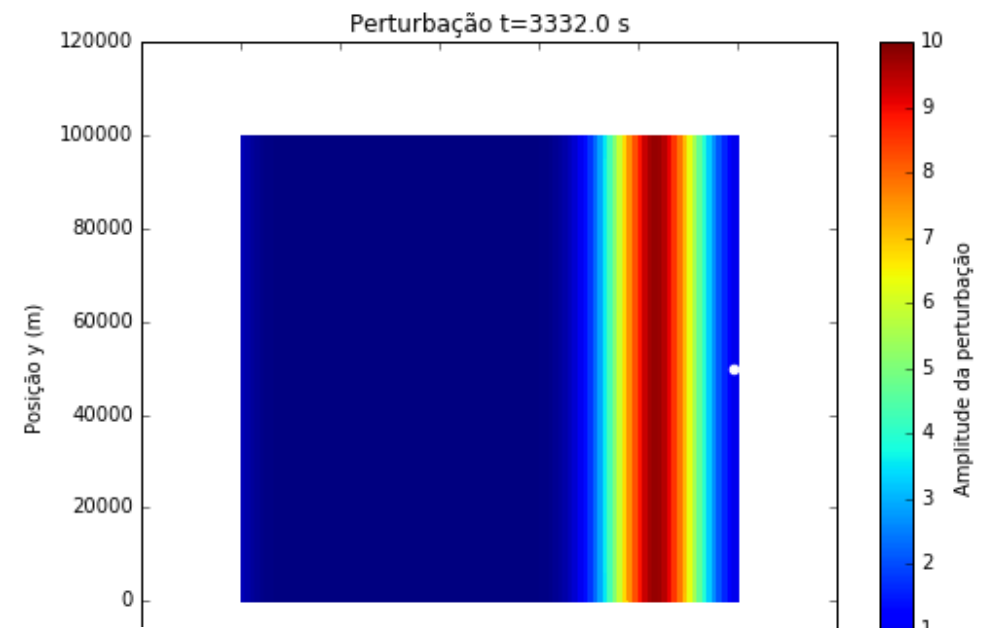
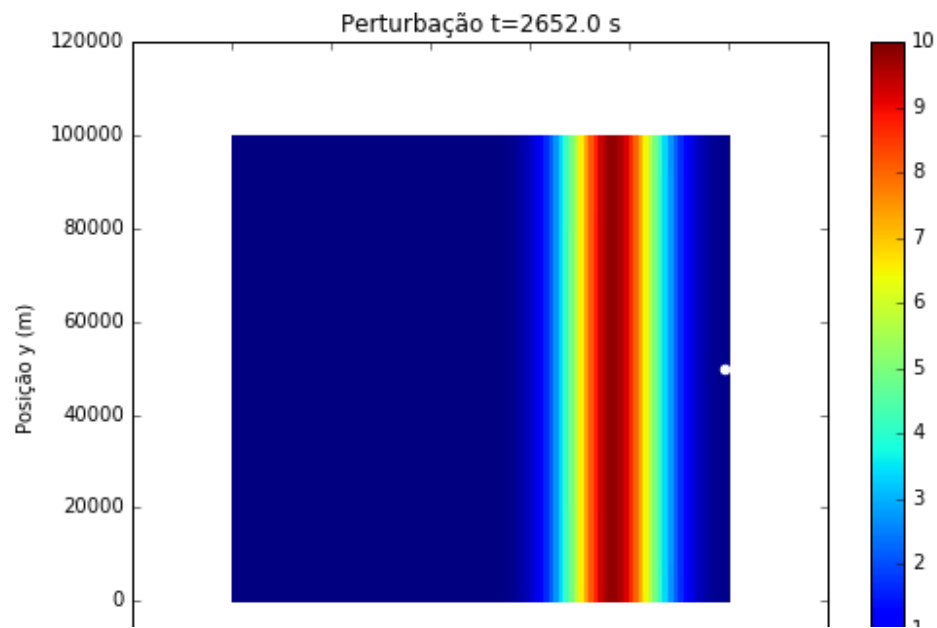
# guardar o sinal na estação
hStation[it] = h[ixStation, iyStation]

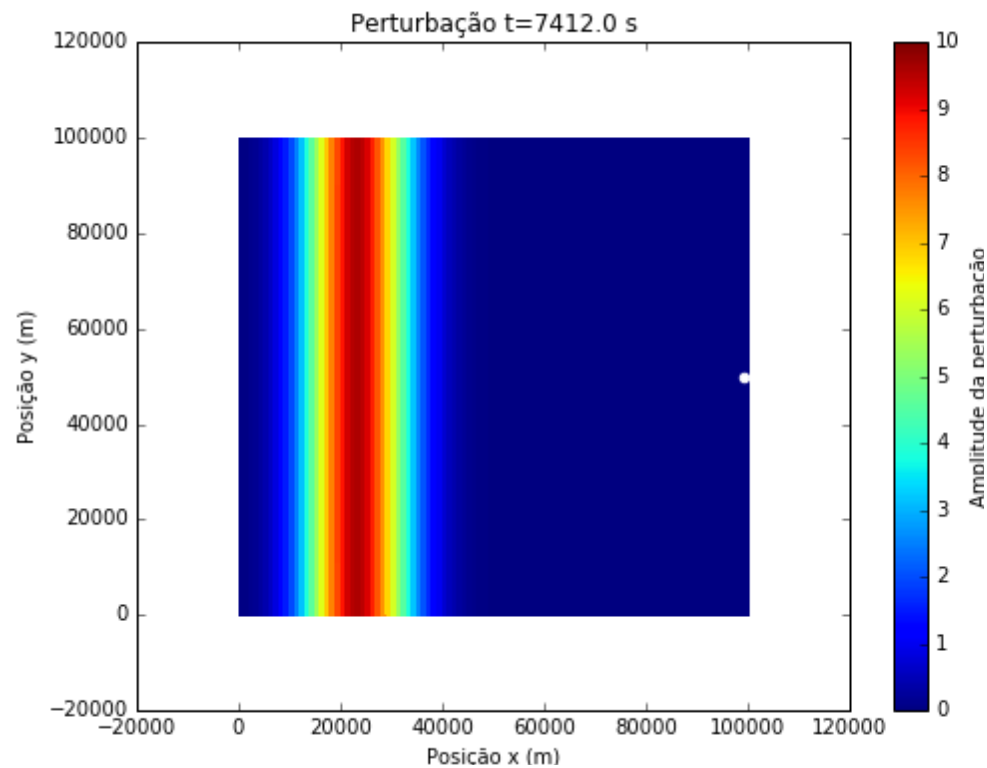
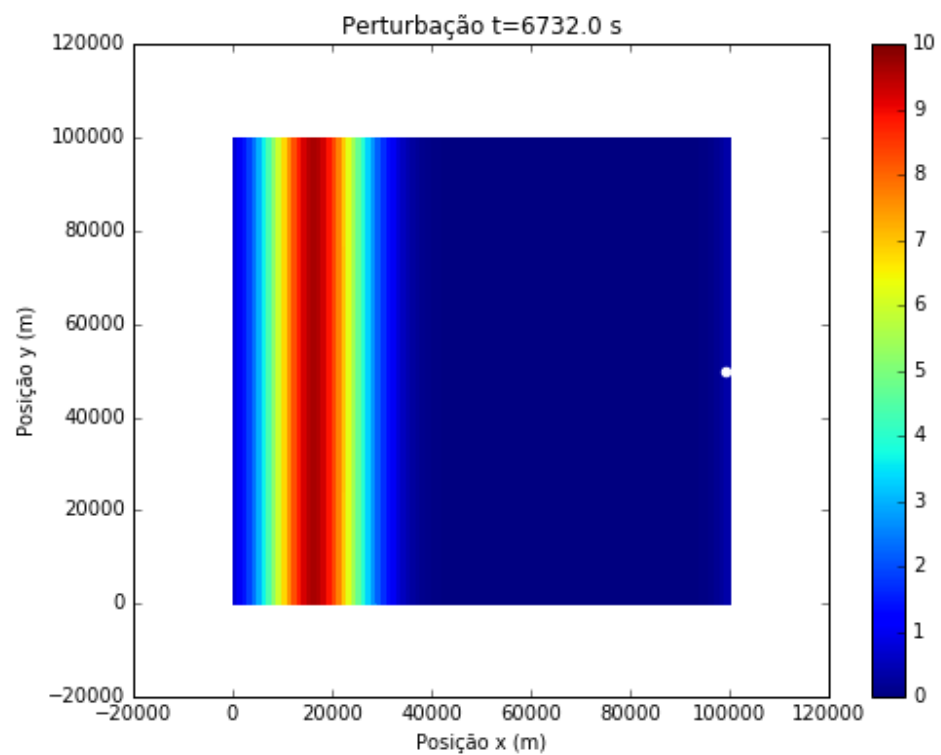
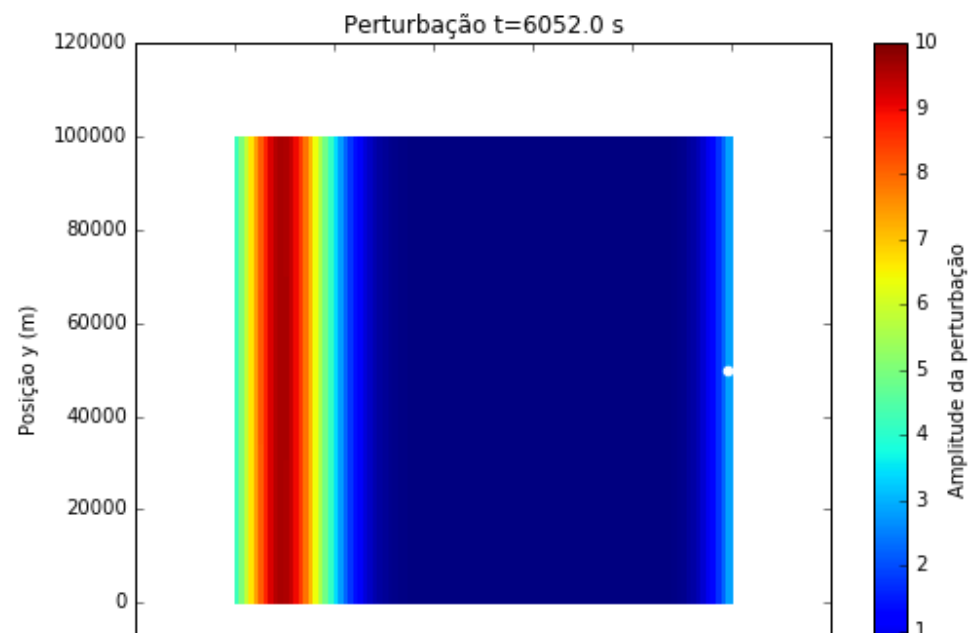
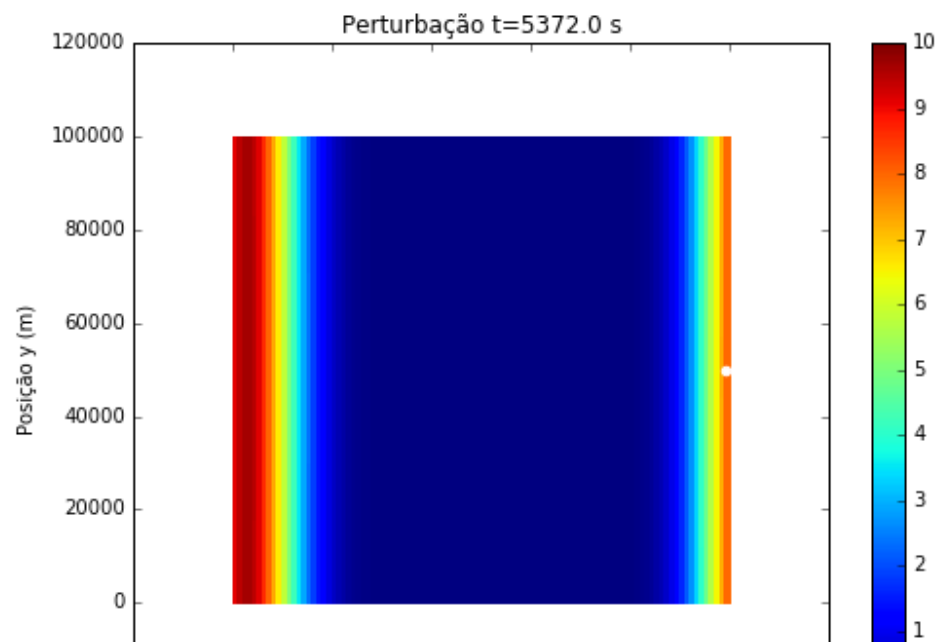
# figura com a evolução do sinal na estação
plt.close()
plt.rcParams['figure.figsize'] = 8, 6
plt.plot(np.arange(0,nt)*dt/60., hStation)

plt.ylabel(u'Amplitude')
plt.xlabel(u'Tempo (min)')
plt.title(u'Estação ix='+ str(ixStation) + ', iy='+ str(iyStation) + ', courant='+ str(courant))
plt.savefig('advection2d-lax-station-courant'+ str(courant) + '.png')

```



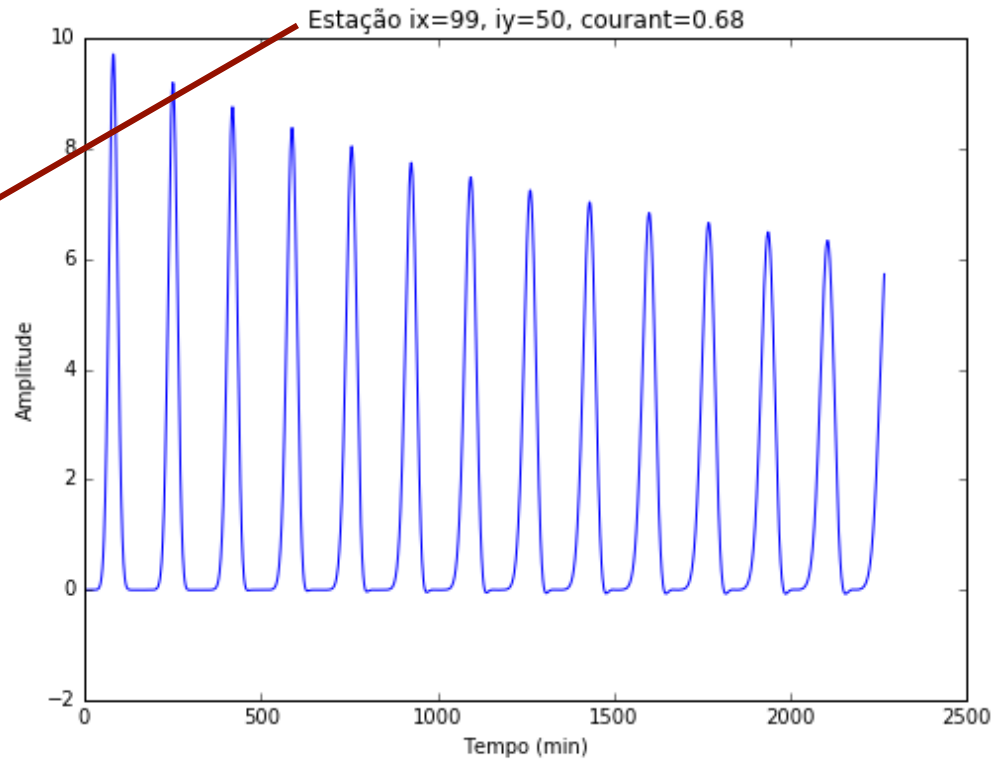
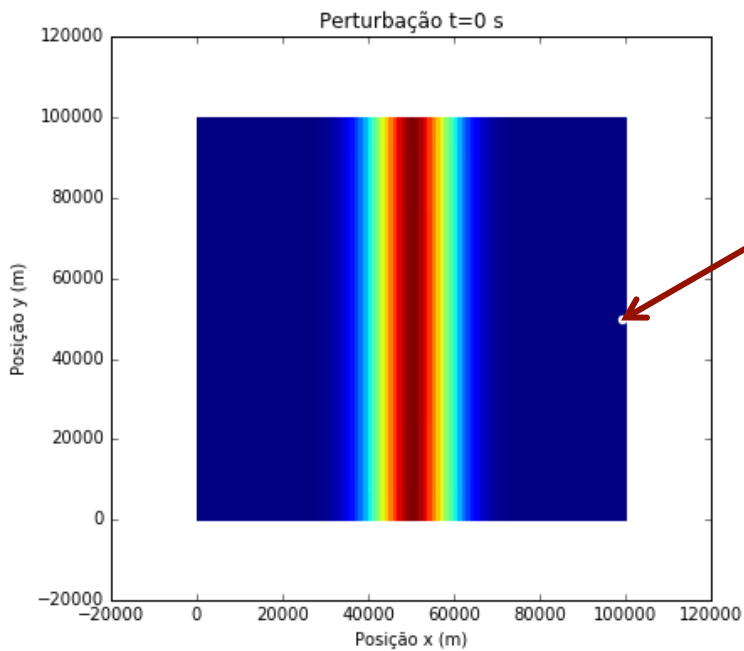




Advecção 2D, Lax

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

$$h_{k,j}^{n+1} = \frac{1}{4}(h_{k-1,j}^n + h_{k+1,j}^n + h_{k,j-1}^n + h_{k,j+1}^n) - \Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$



Equação de advecção (linear, 1D)

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

$$T_k^{n+1} = T_k^n - u \Delta t \frac{T_{k+1}^n - T_{k-1}^n}{2\Delta x}$$

- Trata-se de um método de 1ª ordem no tempo e 2ª ordem no espaço.
- A solução depende de condições fronteira espaciais. Vamos definir a solução num domínio espacial finito:

$$x \in [0, L_x] \implies x_k = (k - 1)\Delta x, k = 1, \dots, N$$

- Vamos considerar dois casos:
 - Condições cíclicas (periódicas): $x_0 = x_N, x_{N+1} = x_1$
 - Condições “abertas”: $\frac{\partial T}{\partial x} = 0$, em $x = x_1, x = x_N$

$$h_{k,j}^{n+1} = \frac{1}{4}(h_{k-1,j}^n + h_{k+1,j}^n + h_{k,j-1}^n + h_{k,j+1}^n) - \Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

```

# Condições fronteira cíclicas em x
for iy in range(1,ny-1):
    ix=0
    hP[ix,iy] = (h[nx-1,iy] + h[ix+1,iy] + h[ix,iy-1] + h[ix,iy+1])/4. \
        - dt2dx * (hu[ix+1,iy] - hu[nx-1,iy]) \
        - dt2dy * (hv[ix,iy+1] - hv[ix,iy-1])

    ix=nx-1
    hP[ix,iy] = (h[ix-1,iy] + h[0,iy] + h[ix,iy-1] + h[ix,iy+1])/4. \
        - dt2dx * (hu[0,iy] - hu[ix-1,iy]) \
        - dt2dy * (hv[ix,iy+1] - hv[ix,iy-1])

# # Condições fronteira abertas em x
# hP[0,iy]=hP[1,iy]
# hP[nx-1,iy]=hP[nx-2,iy]

# Condições fronteira cíclicas em y
for ix in range(1,nx-1):
    iy=0
    hP[ix,iy] = (h[ix-1,iy] + h[ix+1,iy] + h[ix,ny-1] + h[ix,iy+1])/4. \
        - dt2dx * (hu[ix+1,iy] - hu[ix-1,iy]) \
        - dt2dy * (hv[ix,iy+1] - hv[ix,ny-1])

    iy=ny-1
    hP[ix,iy] = (h[ix-1,iy] + h[ix+1,iy] + h[ix,iy-1] + h[ix,0])/4. \
        - dt2dx * (hu[ix+1,iy] - hu[ix-1,iy]) \
        - dt2dy * (hv[ix,0] - hv[ix,iy-1])

# # Condições fronteira abertas em y
# hP[ix,0]=hP[ix,1]
# hP[ix,ny-1]=hP[ix,ny-2]

```

$$h_{k,j}^{n+1} = \frac{1}{4}(h_{k-1,j}^n + h_{k+1,j}^n + h_{k,j-1}^n + h_{k,j+1}^n) - \Delta t \frac{u_{k+1,j}^n h_{k+1,j}^n - u_{k-1,j}^n h_{k-1,j}^n}{2\Delta x} - \Delta t \frac{v_{k,j+1}^n h_{k,j+1}^n - v_{k,j-1}^n h_{k,j-1}^n}{2\Delta y}$$

Condições fronteira nos cantos

`ix=0; ixm=nx-1; ixp=1`

`iy=0; iym=ny-1; iyp=1`

`hP[ix,iy] = 1./4. * (h[ixm,iy] + h[ixp,iy] + h[ix,iym] + h[ix,iyp]) \`
`- dt2dx * (hu[ixp,iy] - hu[ixm,iy]) \`
`- dt2dy * (hv[ix,iyp] - hv[ix,iym])`

`ix=nx-1; ixm=nx-2; ixp=0`

`iy=0; iym=ny-1; iyp=1`

`hP[ix,iy] = 1./4. * (h[ixm,iy] + h[ixp,iy] + h[ix,iym] + h[ix,iyp]) \`
`- dt2dx * (hu[ixp,iy] - hu[ixm,iy]) \`
`- dt2dy * (hv[ix,iyp] - hv[ix,iym])`

`ix=0; ixm=nx-1; ixp=1`

`iy=ny-1; iym=ny-2; iyp=0`

`hP[ix,iy] = 1./4. * (h[ixm,iy] + h[ixp,iy] + h[ix,iym] + h[ix,iyp]) \`
`- dt2dx * (hu[ixp,iy] - hu[ixm,iy]) \`
`- dt2dy * (hv[ix,iyp] - hv[ix,iym])`

`ix=nx-1; ixm=nx-2; ixp=0`

`iy=ny-1; iym=ny-2; iyp=0`

`hP[ix,iy] = 1./4. * (h[ixm,iy] + h[ixp,iy] + h[ix,iym] + h[ix,iyp]) \`
`- dt2dx * (hu[ixp,iy] - hu[ixm,iy]) \`
`- dt2dy * (hv[ix,iyp] - hv[ix,iym])`

Condições fronteira abertas nos cantos

`# hP[0,0]=hP[1,1]`

`# hP[0,ny-1]=hP[1,ny-2]`

`# hP[nx-1,0]=hP[nx-2,1]`

`# hP[nx-1,ny-1]=hP[nx-2,ny-2]`