Ciências
ULisboa

Modelação Numérica 2022
Aula 8

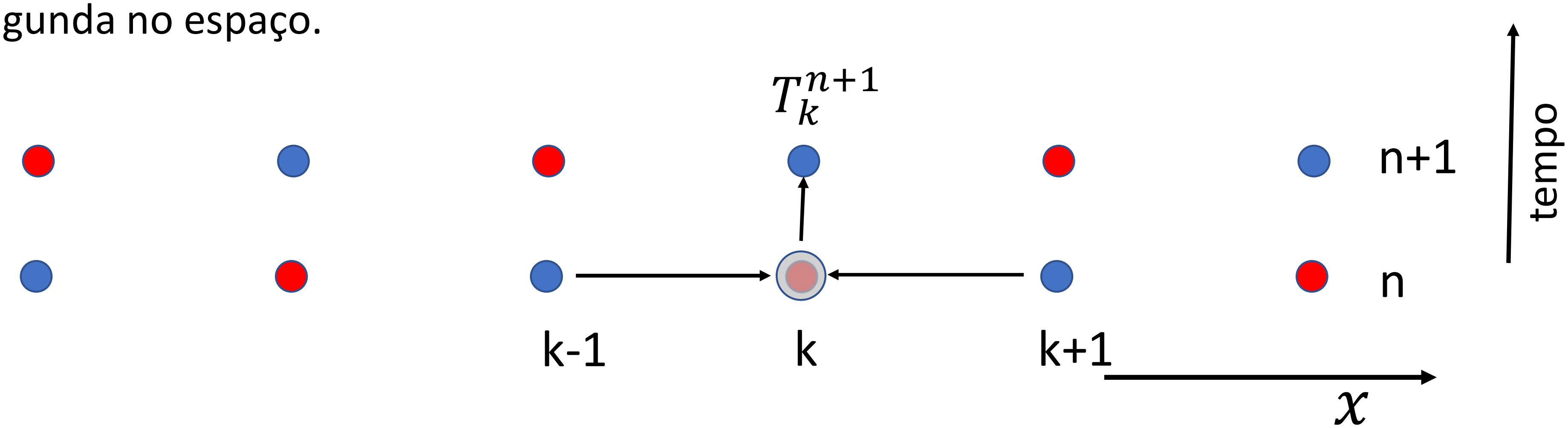
Resolução numérica da equação de adveção linear pelos métodos de Lax e LeapFrog.
Condição de estabilidade (Número de Courant).

Aproximações FCTS e Lax

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

Lax: $\boldsymbol{T_k^{n+1} = \frac{1}{2}(T_{k-1}^n + T_{k+1}^n) - u\Delta t \frac{T_{k+1}^n - T_{k-1}^n}{2\Delta x}}$

Continua a ser um método com 1 nível temporal e de primeira ordem no tempo e segunda no espaço.



O método de LeapFrog

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

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

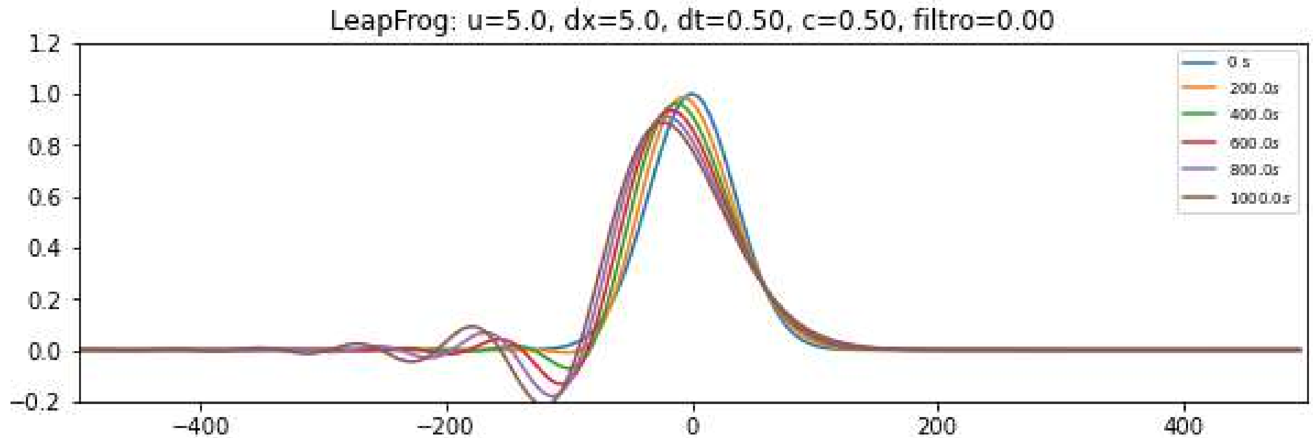
LeapFrog: $T_k^{n+1} = T_k^{n-1} - u2\Delta t \frac{T_{k+1}^n - T_{k-1}^n}{2\Delta x}$

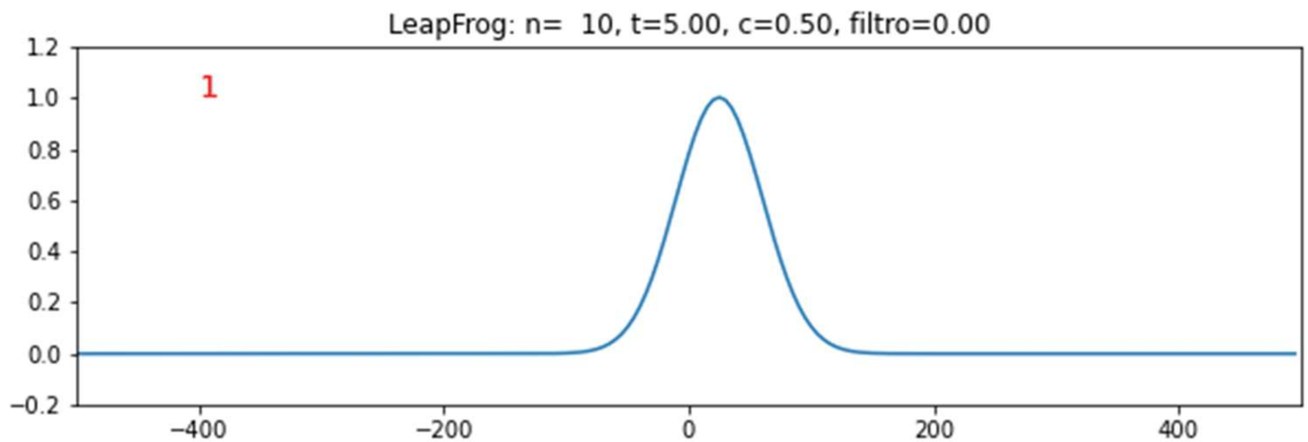
No caso do LeapFrog é preciso conhecer dois passos de tempo anteriores $(n - 1, n)$ para calcular o estado futuro $(n + 1)$.

Leapfrog $c = 0.5$ dispersão numérica

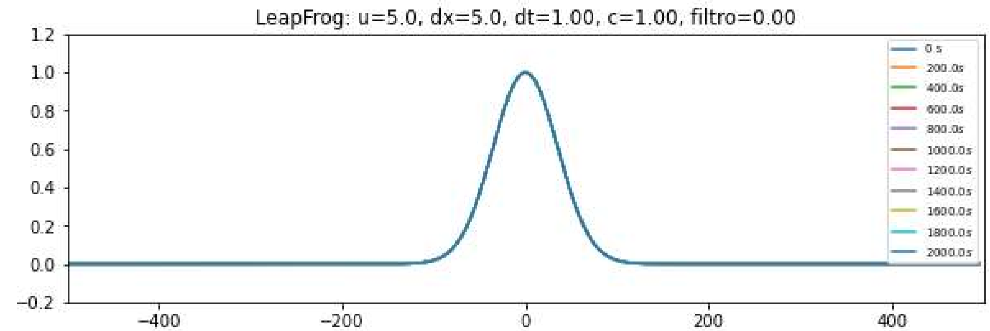
Ao longo do tempo a bossa vai-se allargando, com oscilações de pequeno comprimento de onda a deslocarem-se mais lentamente: a isto chama-se **dispersão** (numérica).

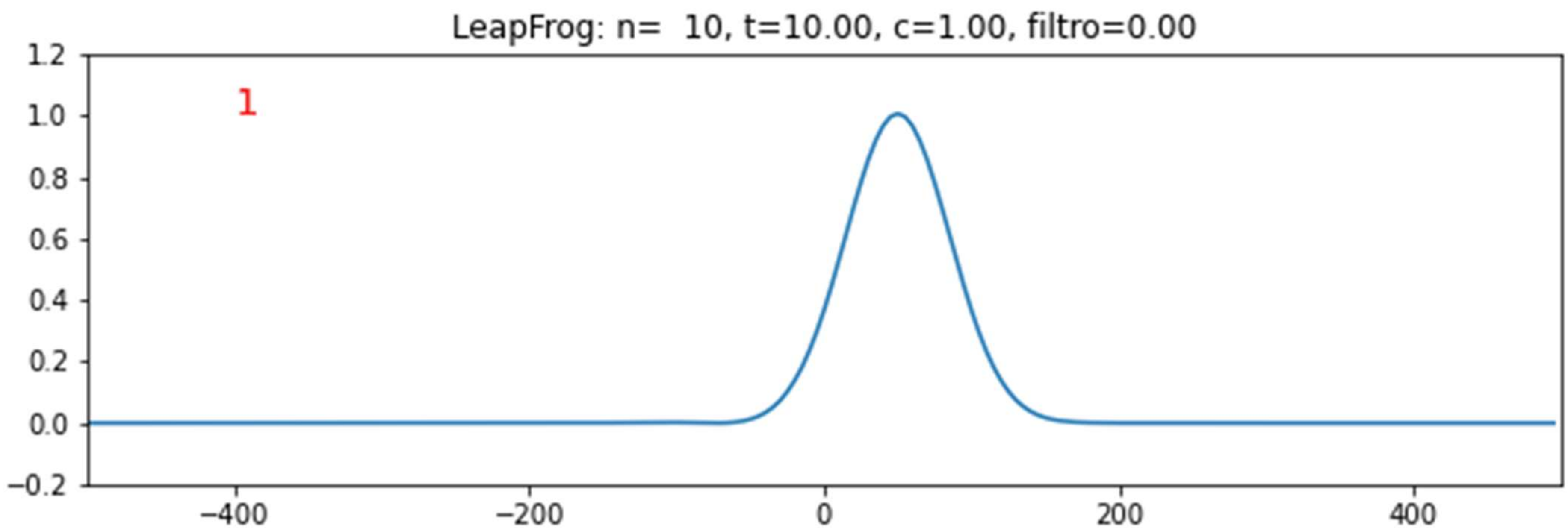
Num sistema dispersivo a velocidade depende do comprimento de onda.





LeapFrog $c = 1$ (10 voltas): perfeito!

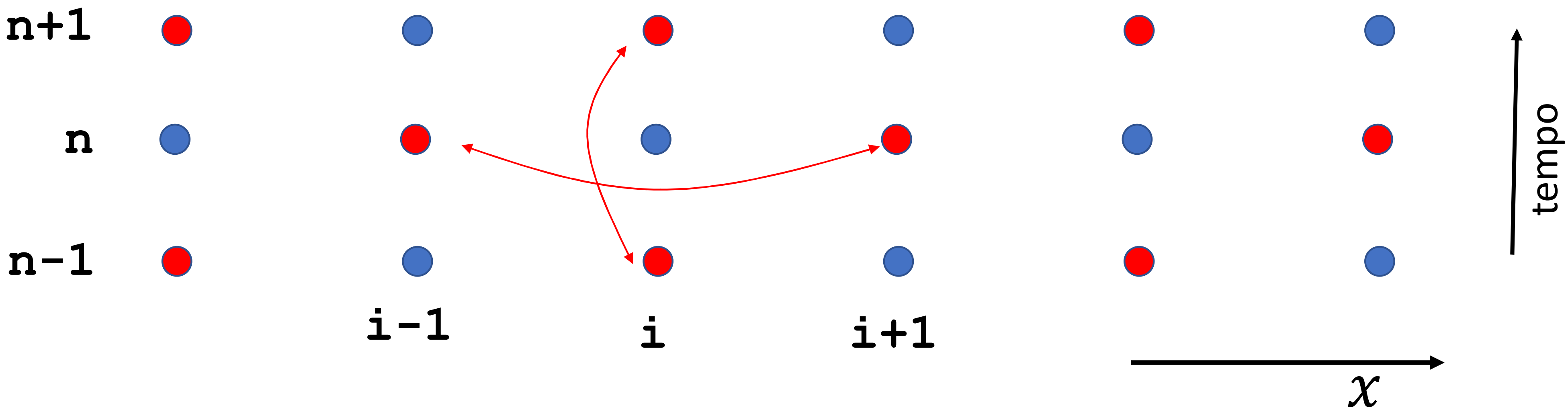




LeapFrog

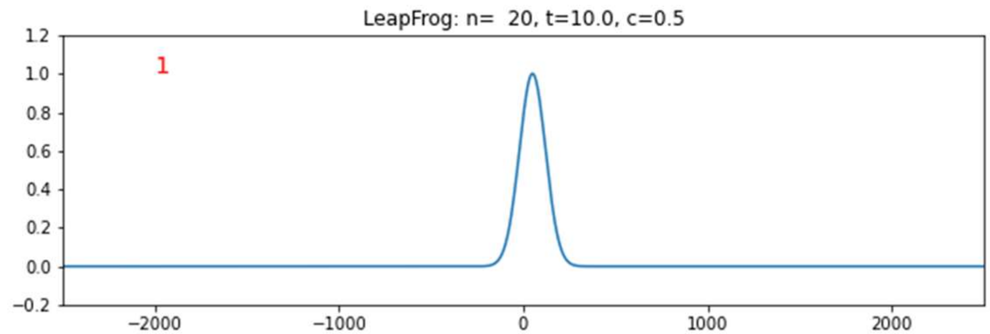
$$T_k^{n+1} = {\color{red}T_k^{n-1}} - u\Delta t \frac{T_{k+1}^{n} - T_{k-1}^{n}}{\Delta x}$$

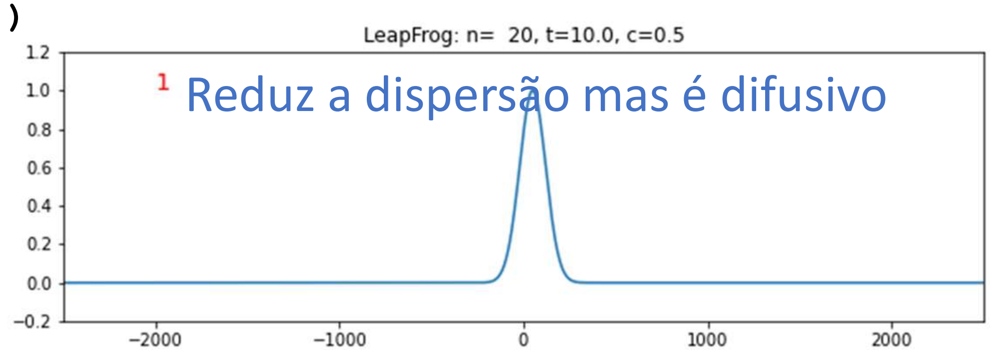
A malha computacional divide-se em conjuntos (pontos vermelhos e azuis) que não comunicam entre si. As malhas tendem a desacoplar, criando ruído.



LeapFrog c =0.5 (10 voltas): dispersivo

```
#Filtro temporal
asel=0.05 #asel=0 desliga o filtro
timefil=[asel,1-2*asel,asel]

for it in range(2,nt):
    for ix in range(1,nx-1):
        TP[ix]=TM[ix]-u*dt/(dx)*(T[ix+1]-T[ix-1])
    TP[nx-1]=TM[nx-1]-u*dt/(dx)*(T[0]-T[nx-2])
    TP[0]=TM[0]-u*dt/(dx)*(T[1]-T[nx-1])
    T=timefil[0]*TM+timefil[1]*T+timefil[2]*TP
    TM=np.copy(T);T=np.copy(TP)
```





Com o filtro temporal reduzem-se as oscilações de pequeno c.d.o (menos dispersão), mas a bossa perde amplitude a alarga (difusão)

Numa próxima aula

Faremos a discussão mais aprofundada das propriedades destes 2 esquemas (Lax e Leapfrog).

Antes disso vamos pensar na adveção a duas dimensões.

Advecção 2D

Equação na forma de fluxo

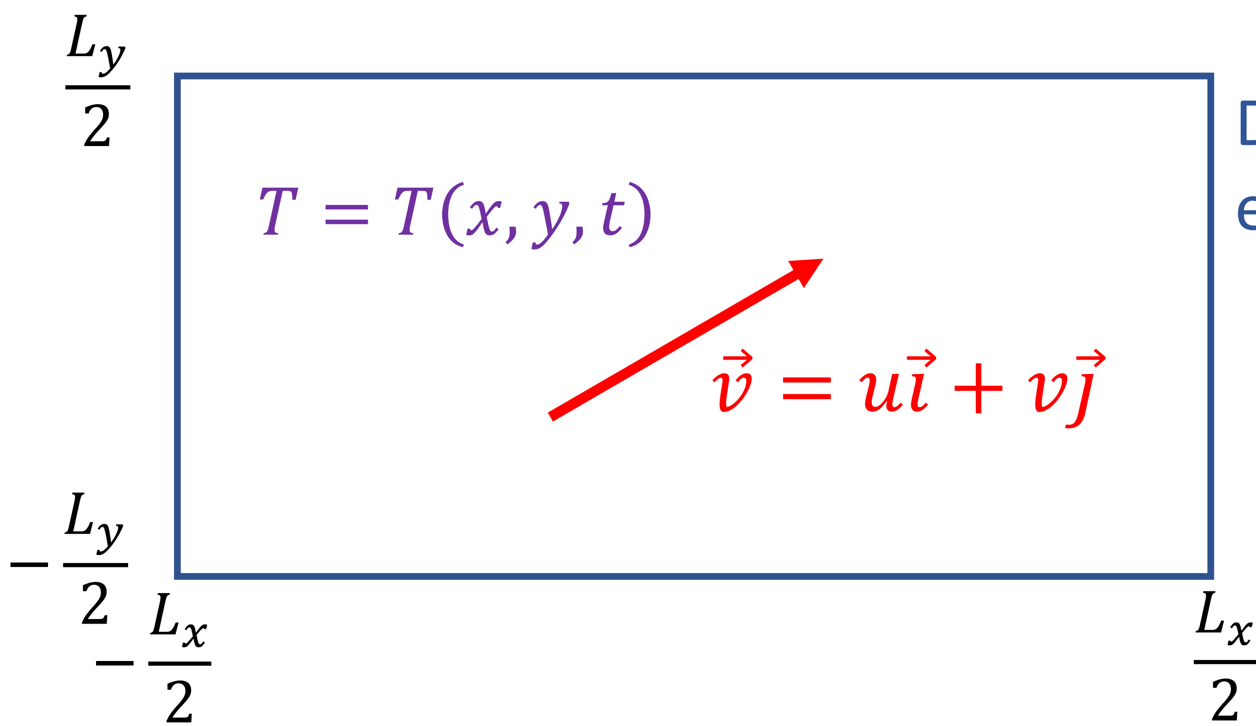
$$\frac{\partial T}{\partial t} = -u\frac{\partial T}{\partial x} - v\frac{\partial T}{\partial y} \Longrightarrow \frac{\partial T}{\partial t} = -\frac{\partial uT}{\partial x} - \frac{\partial vT}{\partial y}.$$

A forma de fluxo é exata em duas condições:

$u, v = const$ (advecção linear)

ou

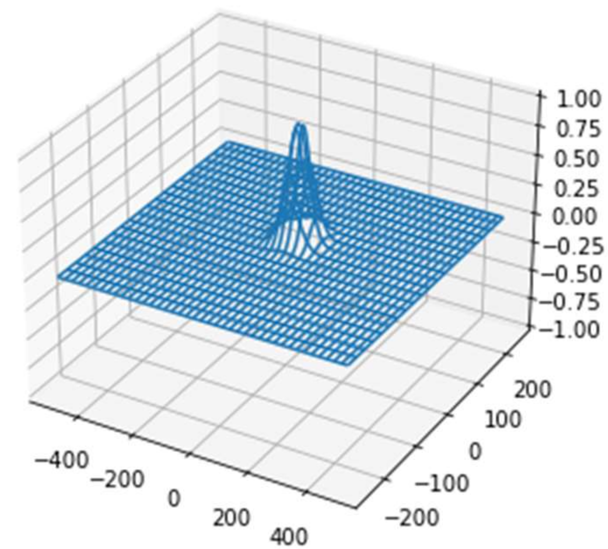
$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$ (fluido incompressível)



Domínio espacial

Advecção 2D Leapfrog (1, wireframe)

```
import numpy as np
import matplotlib.pyplot as plt
import imageio
import os
from mpl_toolkits.mplot3d import axes3d

def graf(xis,yps,zed,it,tit):
    fig=plt.figure(2,figsize=(10,10))
    ax = fig.add_subplot(111, projection='3d')
    ax.plot_wireframe(xis,yps,zed,rstride=5,cstride=5)
    ax.set_zlim3d(-1,1)
    plt.title(tit)
    fn='mov'+str(it)+'.png'
    plt.savefig(fn)
    plt.clf()
    return fn
```



Advecção 2D Leapfrog (2, preliminares)

```
…
nx=200;dx=5;Lx=nx/2*dx;ny=100;dy=dx;Ly=ny/2*dy #grelha
u=5;v=10;dt=0.3
nvolta=int(2*Lx/u/dt); nt=nvolta*1+1; passo=nvolta
asel=0.0; timefil=[asel,1-2*asel,asel]
x=np.arange(-Lx,Lx,dx);y=np.arange(-Ly,Ly,dy)
xis=np.zeros((nx,ny));yps=np.zeros((nx,ny))
for iy in range(ny):
    xis[:,iy]=x
for ix in range(nx):
    yps[ix,:]=y
movie=10; frames=[]
courant=np.sqrt(u**2+v**2)*dt/(np.sqrt(dx**2+dy**2)/2)
Wx=50;x0=0;Wy=30;y0=0
TI=np.exp(-((xis-x0)/Wx)**2-((yps-y0)/Wy)**2) #perturbação inicial
T=np.copy(TI);TP=np.copy(T);TM=np.copy(T)
```

#xis, yps são arrays 2d

Advecção 2D Leapfrog (3, 1º passo)

…

```
#1° passo Euler
for ix in range(nx):
    ixm=ix-1;ixp=ix+1
    if ixm<0:
        ixm=nx-1
    elif ixp>nx-1:
        ixp=0
    for iy in range(ny):
        iym=iy-1;iyp=iy+1
        if iym<0:
            iym=ny-1
        elif iyp>ny-1:
            iyp=0
        T[ix,iy]=TM[ix,iy]-u*dt/(2*dx)*(TM[ixp,iy]-TM[ixm,iy])- \
                 v*dt/(2*dx)*(TM[ix,iyp]-TM[ix,iym])
```

Advecção 2D Leapfrog (4, leapfrog)

…

```
for it in range(2,nt):
    for ix in range(nx): #percorre-se todos os pontos
        ixm=ix-1;ixp=ix+1
        if ixm<0:
            ixm=nx-1
        elif ixp>nx-1:
            ixp=0
        for iy in range(ny):
            iym=iy-1;iyp=iy+1
            if iym<0:
                iym=ny-1
            elif iyp>ny-1:
                iyp=0
            TP[ix,iy]=TM[ix,iy]-u*dt/dx*(T[ixp,iy]-T[ixm,iy])-\
                      v*dt/dy*(T[ix,iyp]-T[ix,iym])
    T=timefil[0]*TM+timefil[1]*T+timefil[2]*TP #filtro temporal
    TM=np.copy(T);T=np.copy(TP) #update temporal
```

Condição
fronteira cícilica

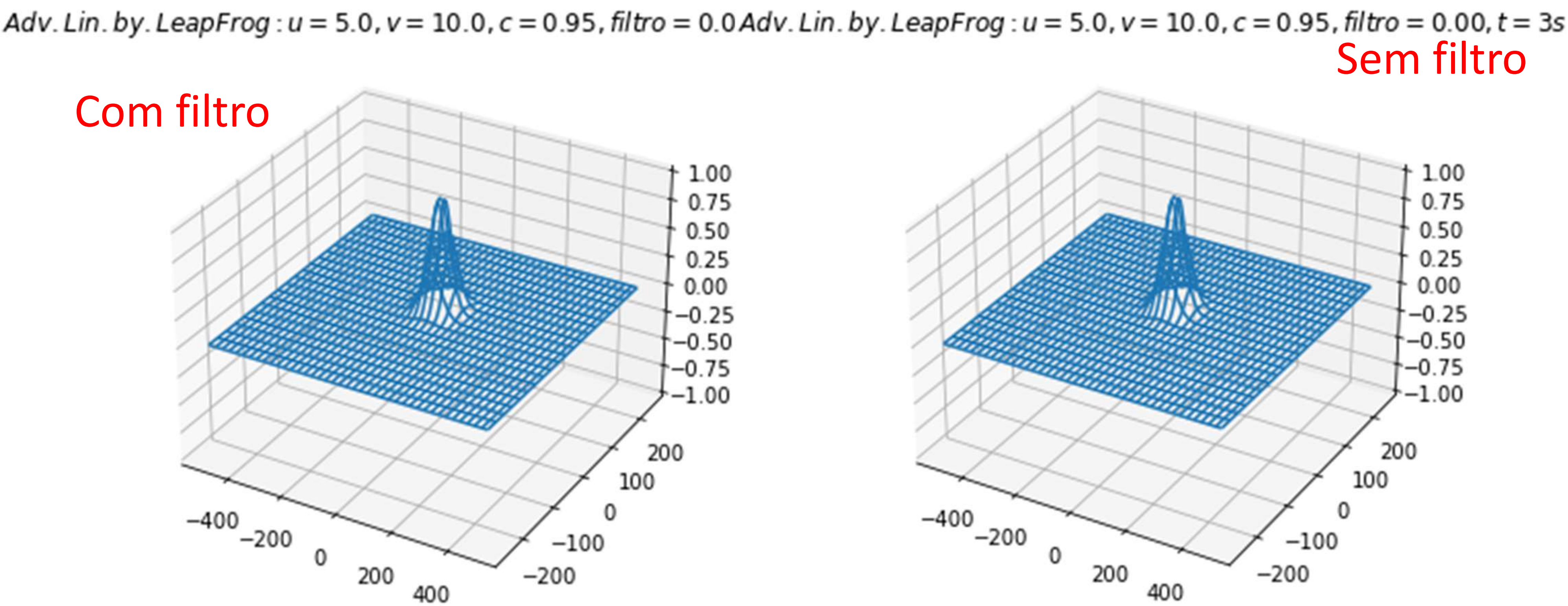
Condição fronteira
cícilica

Advecção 2D Leapfrog (5, movie)

…

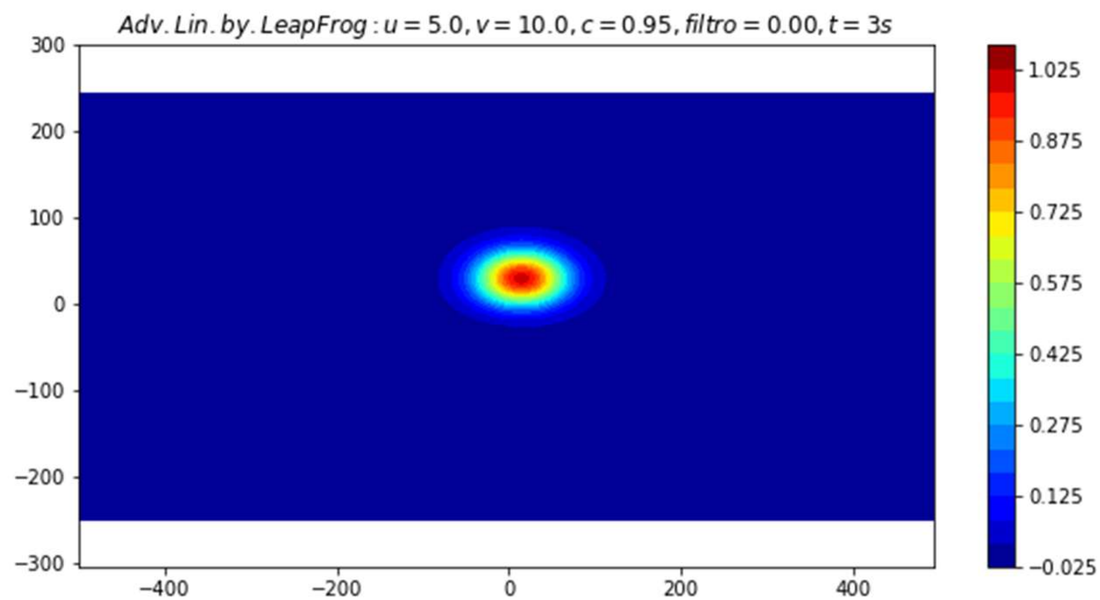
```
    if it%movie==0:
        tit=r'$Adv.Lin. by. LeapFrog: u=%3.1f, v=%3.1f, c=%3.2f, \
              filtro=%3.2f, t=%4.0f s $' % (u,v,courant,asel,it*dt)
        fn=graf(xis,yps,T,it,tit)
        frames.append(fn)

images=[] #frames para filme
for frame in frames:
    images.append(imageio.imread(frame))
    os.remove(frame)
imageio.mimsave('ADV2d'+'.gif', images,duration=0.1)
```

Condições cíclicas



O output gráfico podia ser diferente

```
def graf(xis,yps,zed,it,tit):
    plt.figure(2,figsize=(10,5))
    map=plt.contourf(xis,yps,zed,cmap='jet',levels=np.arange(-0.025,1.1,0.05))
    plt.colorbar(map)
    plt.axis('equal')
    plt.title(tit)
    fn='mov'+str(it)+'.png'
    plt.savefig(fn)
    plt.clf()
    return fn
```



Popriedades de um método numérico

Consistência: converge para a solução analítica $\lim_{\Delta x, \Delta t \to 0}$

Precisão: "velocidade" de convergência do erro para zero, e.g. $\mathcal{O}(\Delta t^2)$

Estabilidade/ atenuação/ difusão numérica

Frequentemente, a estabilidade é conseguida com atenuação de pequenos comprimentos de onda: comportamento difusivo.

Velocidade de fase/ Dispersão

Frequentemente, diferentes comprimentos de onda são propagados a diferente velocidade, apesar de isso não resultar de leis físicas: dispersão numérica

Equação de advecção (linear,1D) $\frac{\partial T}{\partial t} = -U \frac{\partial T}{\partial x}$

Em geral, se a condição inicial for $T(x,0) = f(x)$, a solução analítica será $T(x,t) = f(x - Ut)$. O comportamento da solução numérica depende de f.

Podemos estudar as propriedades de um método de solução a partir da análise do comportamento da solução no caso em que a condição inicial é uma sinusoide (k é o número de onda, $k = 2\pi/\lambda$)

$$T(x,0) = \cos(kx)$$

Com esta condição inicial a solução analítica é:

$$T(x,t) = \cos\big(k(x - Ut)\big)$$

i.e. a sinusoide propaga-se à velocidade U, sem deformação.

Análise de estabilidade: Método de von Neumann (ou de Fourier)

Em geral a solução numérica pode ser escrita na forma discreta (omitindo $Re\{\}$):

$$T(x = m\Delta x, t = 0) \approx T_m^0 = e^{ikm\Delta x}$$

No instante $t = n\Delta t$, vamos estudar a solução (numérica) por separação de variáveis:

$$T(x = m\Delta x, t = n\Delta t) \approx T_m^n = B^n e^{ikm\Delta x}$$

Em que B é uma constante complexa, cujo valor depende do método numérico utilizado. A solução analítica seria recuperada se

$$B^n = e^{-ikUn\Delta t} \Leftrightarrow B = e^{-ikU}$$

Sendo nesse caso B um complexo de módulo 1 e fase $-kU\Delta t$. No caso geral, se $|B| < 1$ a solução decairá no tempo com B^n, se $|B| > 1$ crescerá exponencialmente no tempo e o método é instável.

Aplicação ao método de Euler FTCS

$$T_m^{n+1} = T_m^n - \text{U}\Delta t \frac{T_{m+1}^n - T_{m-1}^n}{2\Delta x} \text{ com } T_m^n = B^n e^{ikm\Delta x}$$

substituindo

$$B^{n+1} e^{ikm\Delta x} = B^n e^{ikm\Delta x} - \frac{U\Delta t}{2\Delta x}\left(B^n e^{ik(m+1)\Delta x} - B^n e^{ik(m-1)\Delta x}\right)$$

$$B^{n+1} = B^n - \frac{U\Delta t}{2\Delta x}\left(B^n e^{ik\Delta x} - B^n e^{-ik\Delta x}\right)$$

$$B = 1 - \frac{U\Delta t}{2\Delta x}\left(e^{ik\Delta x} - e^{-ik\Delta x}\right) = 1 - i\frac{U\Delta t}{\Delta x}\sin(k\Delta x)$$

Logo

$$|B| = \sqrt{1 + \frac{U^2\Delta t^2}{\Delta x^2}\sin^2(k\Delta x)} > 1$$

$$\sin(k\Delta x) = \frac{e^{ik\Delta x} - e^{-ik\Delta x}}{2i}$$

E o método é absolutamente instável.