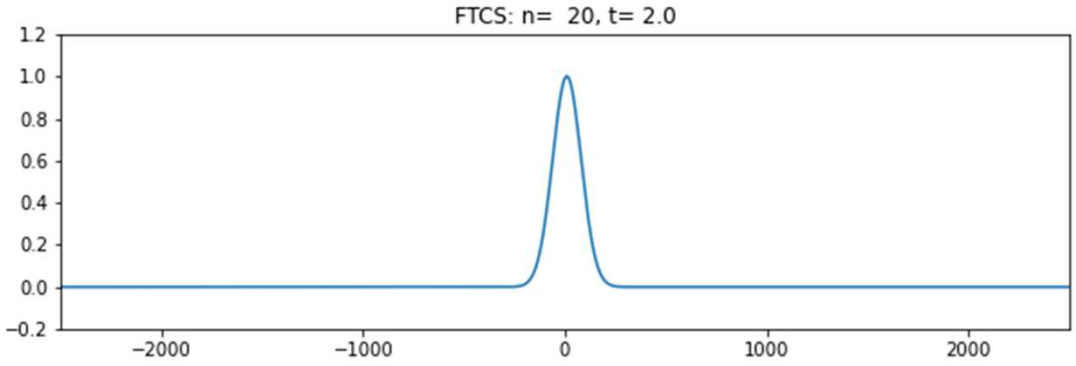
Ciências ULisboa

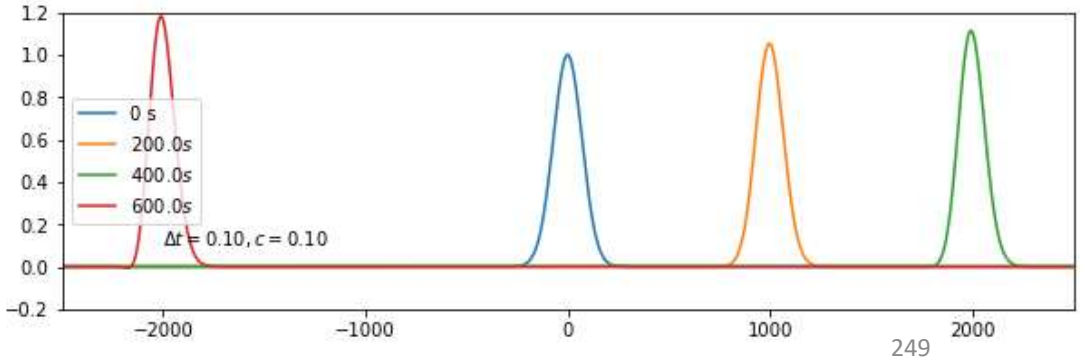
Modelação Numérica
Aula 13

Equação de advecção linear. Método de Lax

Agora com um filme...

```
…
import imageio
import os
…
movie=20
frames=[]
for it in range(1,nt):
      …
    if it%movie==0:
        plt.figure(2,figsize=(10,3))
        plt.plot(x,T);
        plt.axis([-Lx,Lx,-0.2,1.2])
        plt.title('FTCS: n=%4i, t=%4.1f' %(it,dt*it))
        fn='mov'+str(it)+'.png'
        plt.savefig(fn); plt.clf()
        frames.append(fn)
…
images=[]
for frame in frames:
   images.append(imageio.imread(frame))
    os.remove(frame)
imageio.mimsave('FTCS'+'.gif',images,duration=0.1)
```





O método de Euler (FTCS) é incondicionalmente instável

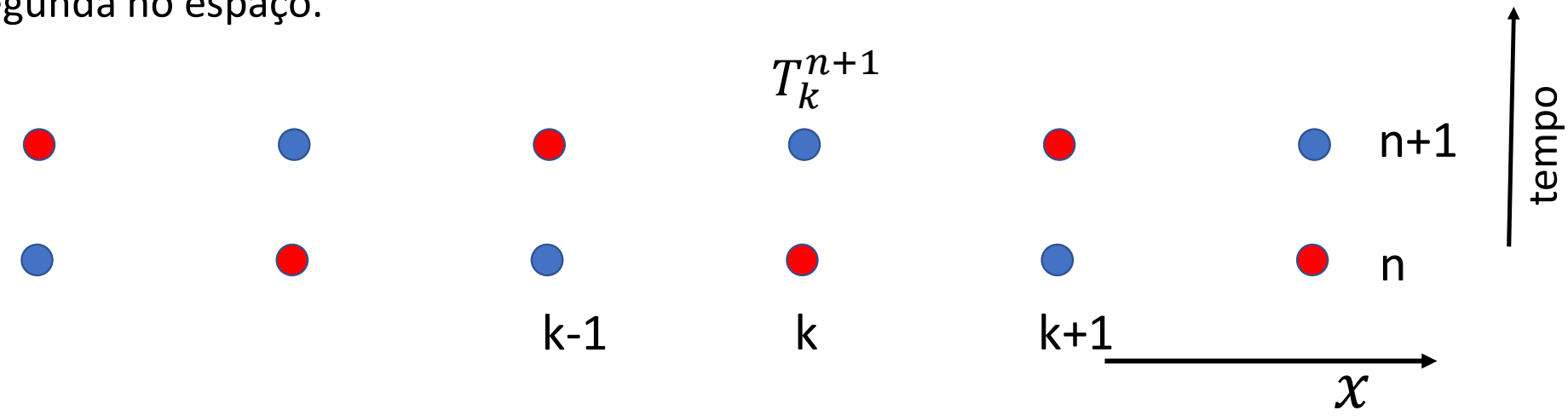
A instabilidade é neste caso **independente** da escolha dos parâmetros de discretização $(\Delta t, \Delta x)$ e não é uma consequência da ordem da aproximação. É possível definir esquemas de 1ª ordem (ou ordem mais elevada) condicionalmente estáveis.

Aproximação de Lax

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

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

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



LAX

```
…
for it in range(1,nt):
    for ix in range(1,nx-1):
        TP[ix]=0.5*(T[ix-1]+T[ix+1])-u*dt/(dx*2)*(T[ix+1]-T[ix-1])
    TP[nx-1]=0.5*(T[nx-2]+T[0])-u*dt/(dx*2)*(T[0]-T[nx-2]) #fronteira cíclica
    TP[0]=0.5*(T[1]+T[nx-1])-u*dt/(dx*2)*(T[1]-T[nx-1]) #fronteira cíclica
    T=np.copy(TP) #update temporal
…
```

FTCS

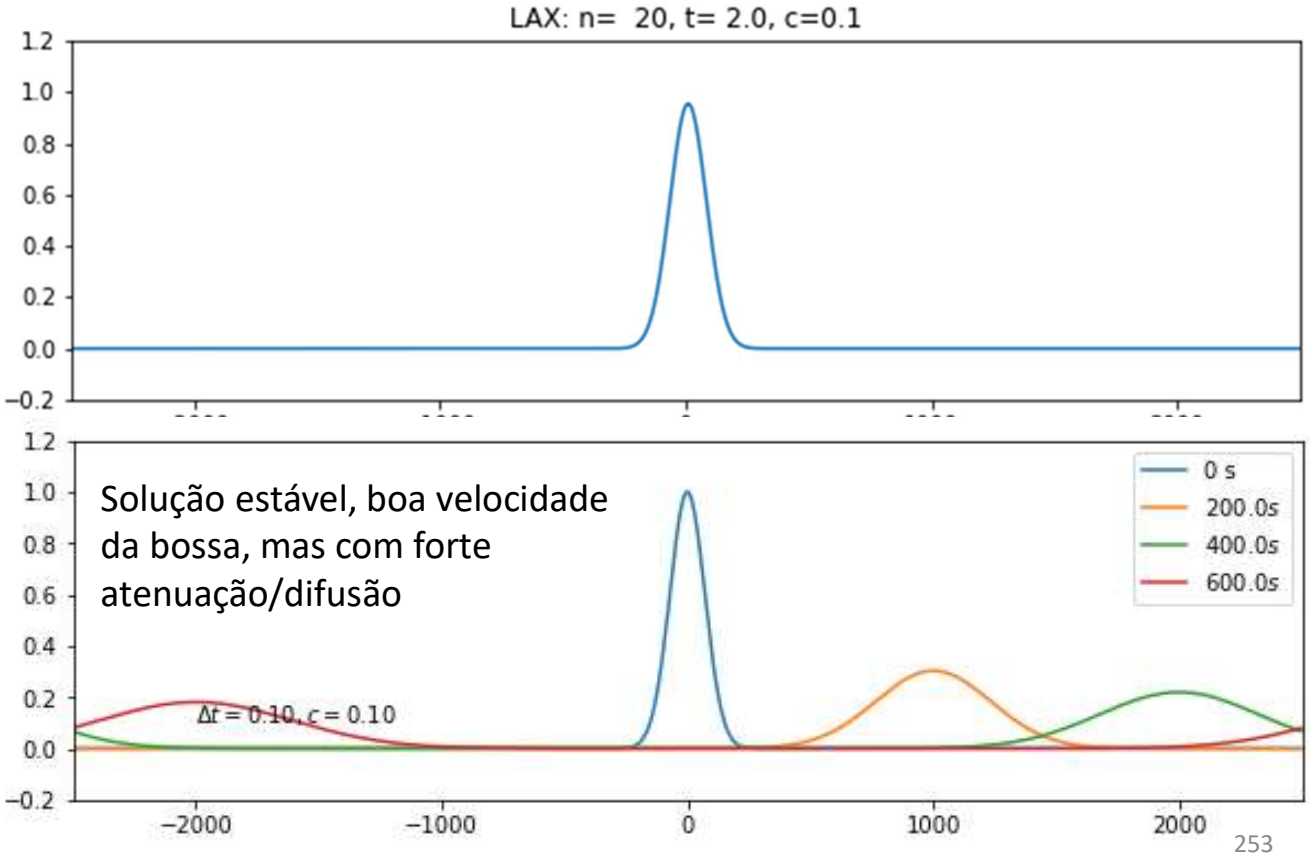
```
for it in range(1,nt):
    for ix in range(1,nx-1):
        TP[ix]=T[ix]-u*dt/(dx*2)*(T[ix+1]-T[ix-1])
    TP[nx-1]=T[nx-1]-u*dt/(dx*2)*(T[0]-T[nx-2]) #fronteira cíclica
    TP[0]=T[0]-u*dt/(dx*2)*(T[1]-T[nx-1]) #fronteira cíclica
    T=np.copy(TP) #update temporal
```

LAX Courant=0.1: difusão numérica

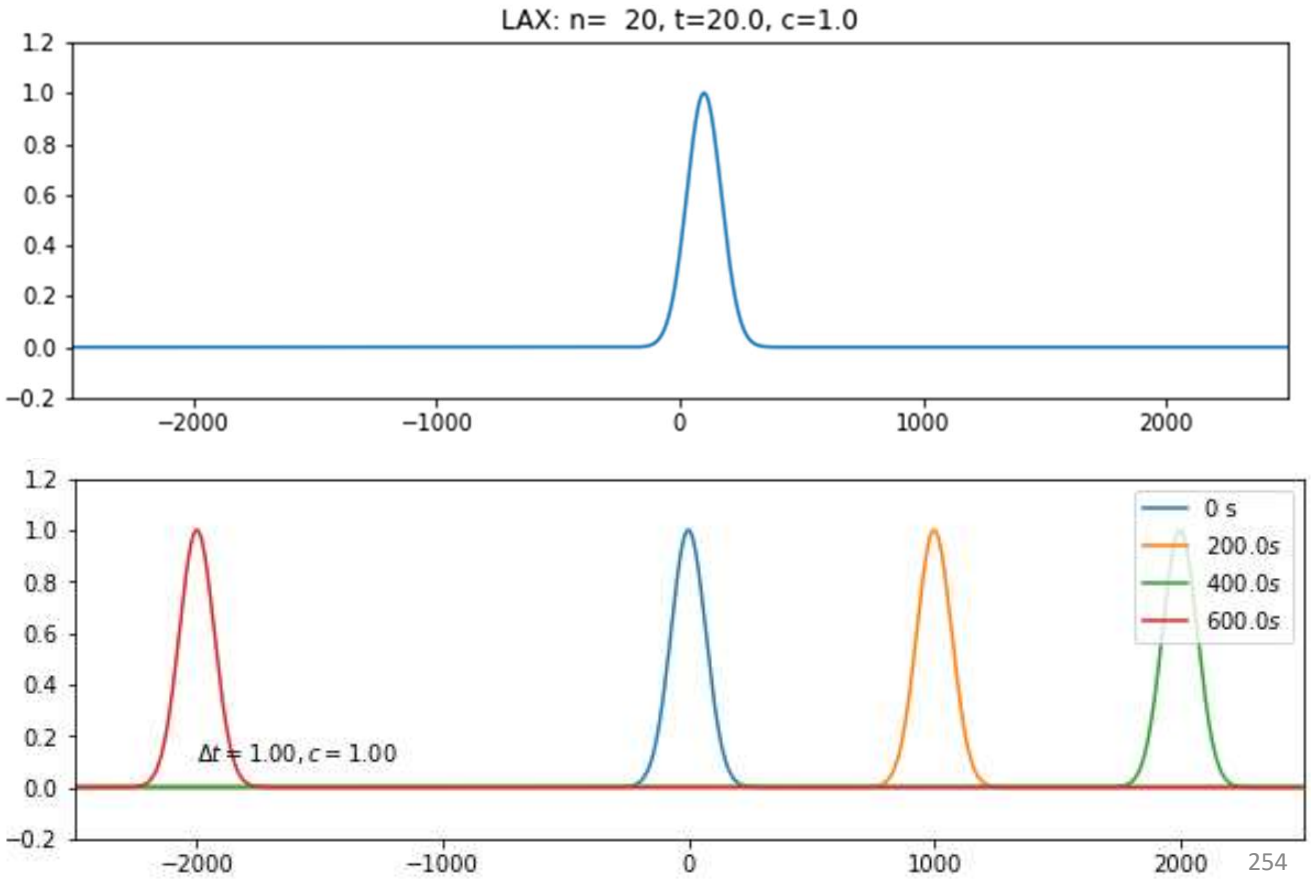
O comportamento do método depende do Número de Courant (of de Courant−Friedrichs−Lewy, CFL)

$$c = \frac{u\Delta t}{\Delta x}$$

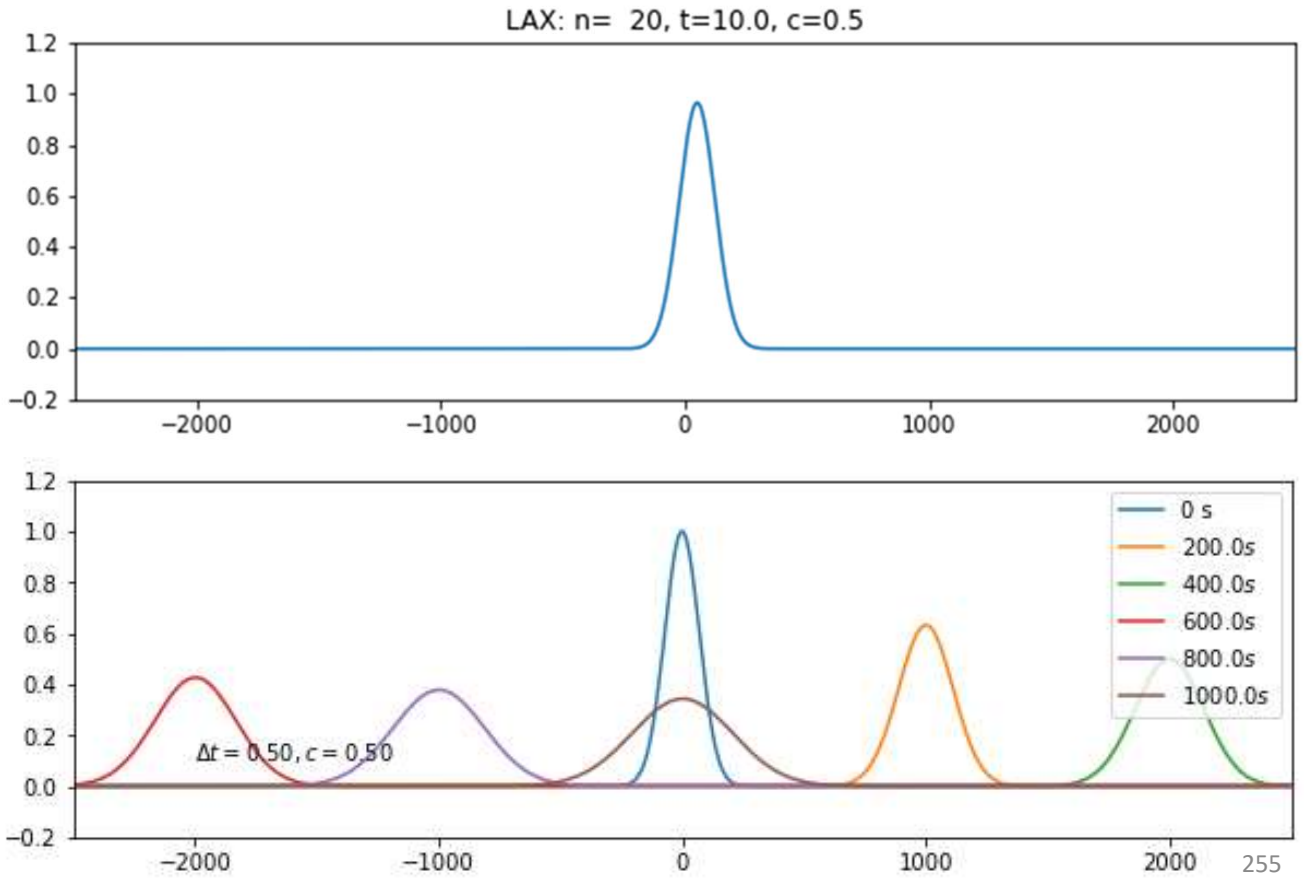
Notar que c é adimensional.



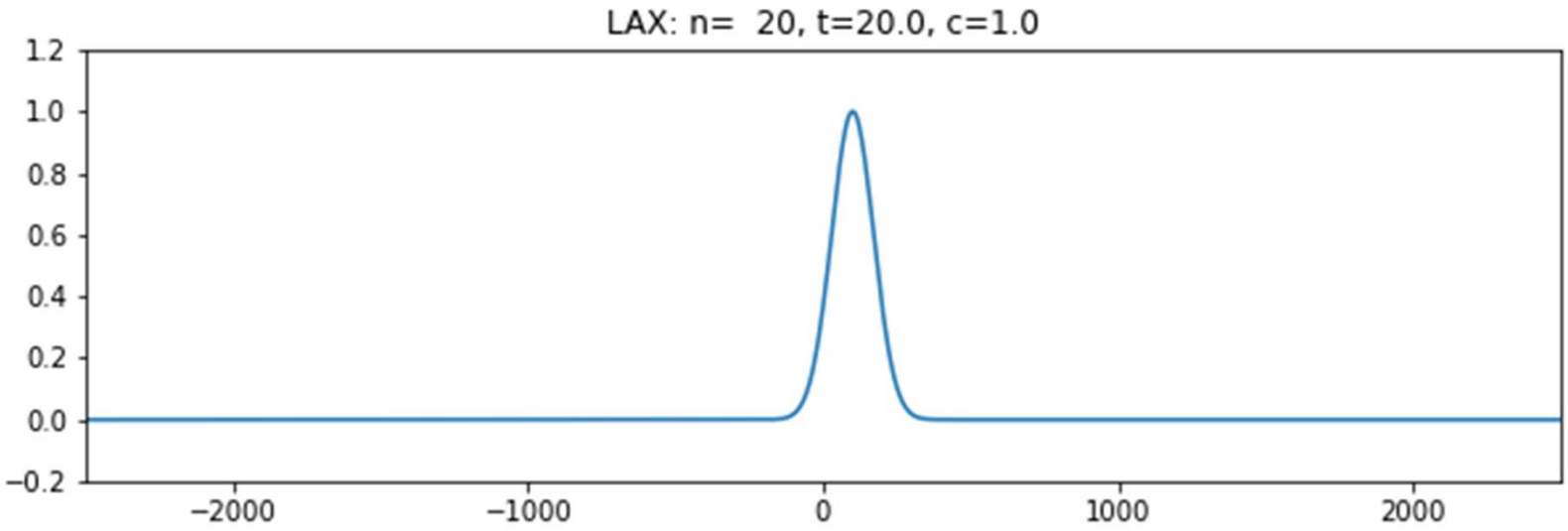
LAX Courant=1.0: sem atenuação



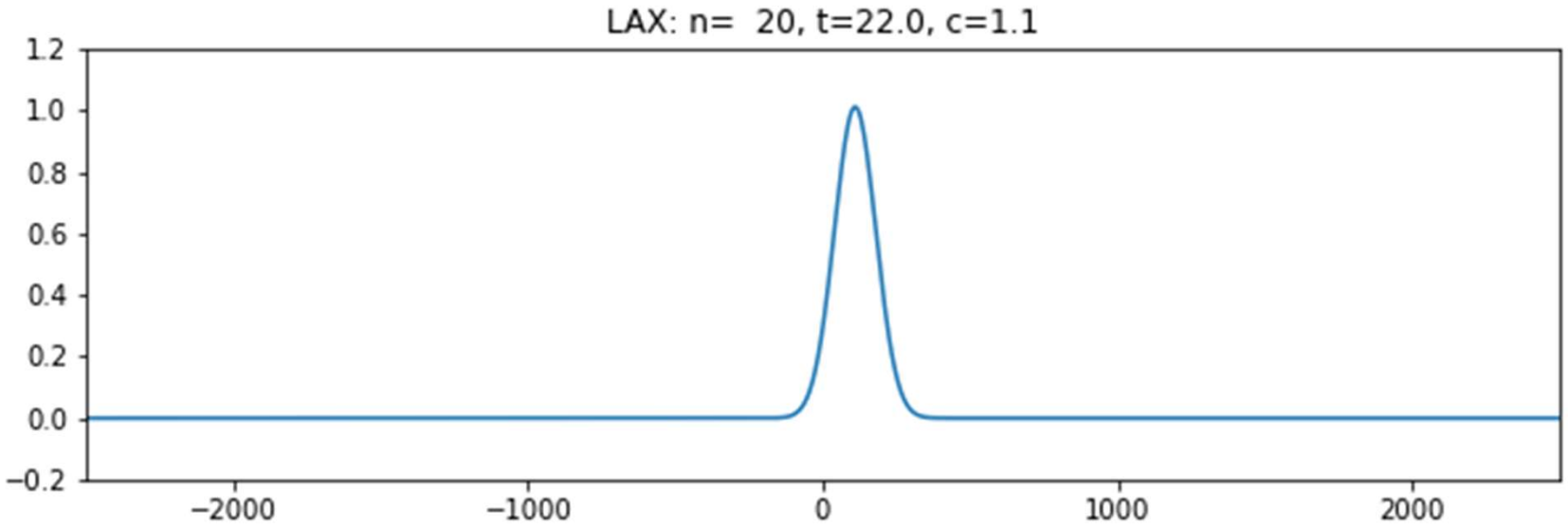
LAX Courant=0.5



LAX Courant=1.0: 5 voltas... perfeito?



LAX Courant=1.1: instável



O método de LAX

É **condicionalmente estável**: **se** $c \leq 1$ $\left(c = \frac{u\Delta t}{\Delta x}\right)$ c é o **número de Courant** (ou CFL)

É **exato** para $c = 1$. Mas **atenua as pequenas escalas se** $\boldsymbol{c < 1}$. No caso da equação de adveção linear $(u = const)$ é fácil escolher $c = 1$.

Em geral não é possível fazer isso, mas deve evitar-se um c pequeno pois a difusão (numérica) será elevada.

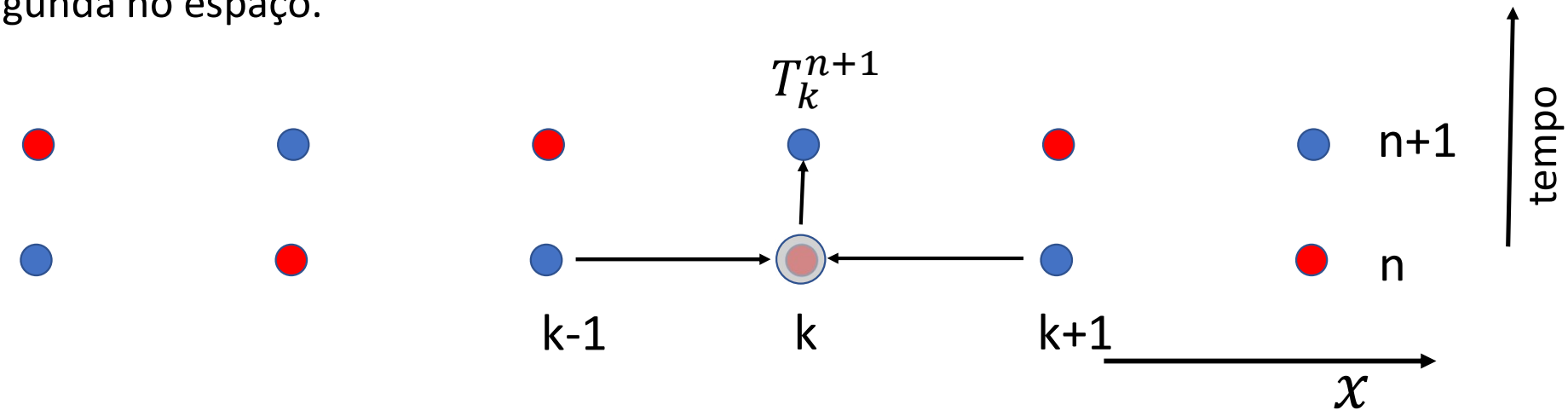
Notem que o $c \leq 1$ pode ser lida como a condição de que, em cada passo de tempo (Δt), a propagação da informação $(l = u\Delta t)$ não pode exceeder um passo de grelha espacial $(l \leq \Delta x)$. No caso da equação de adveção linear a informação propaga-se à velocidade do fluido (u). Noutros casos pode propagar-se mais rapidamente, nomeadamente na forma de uma onda.

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

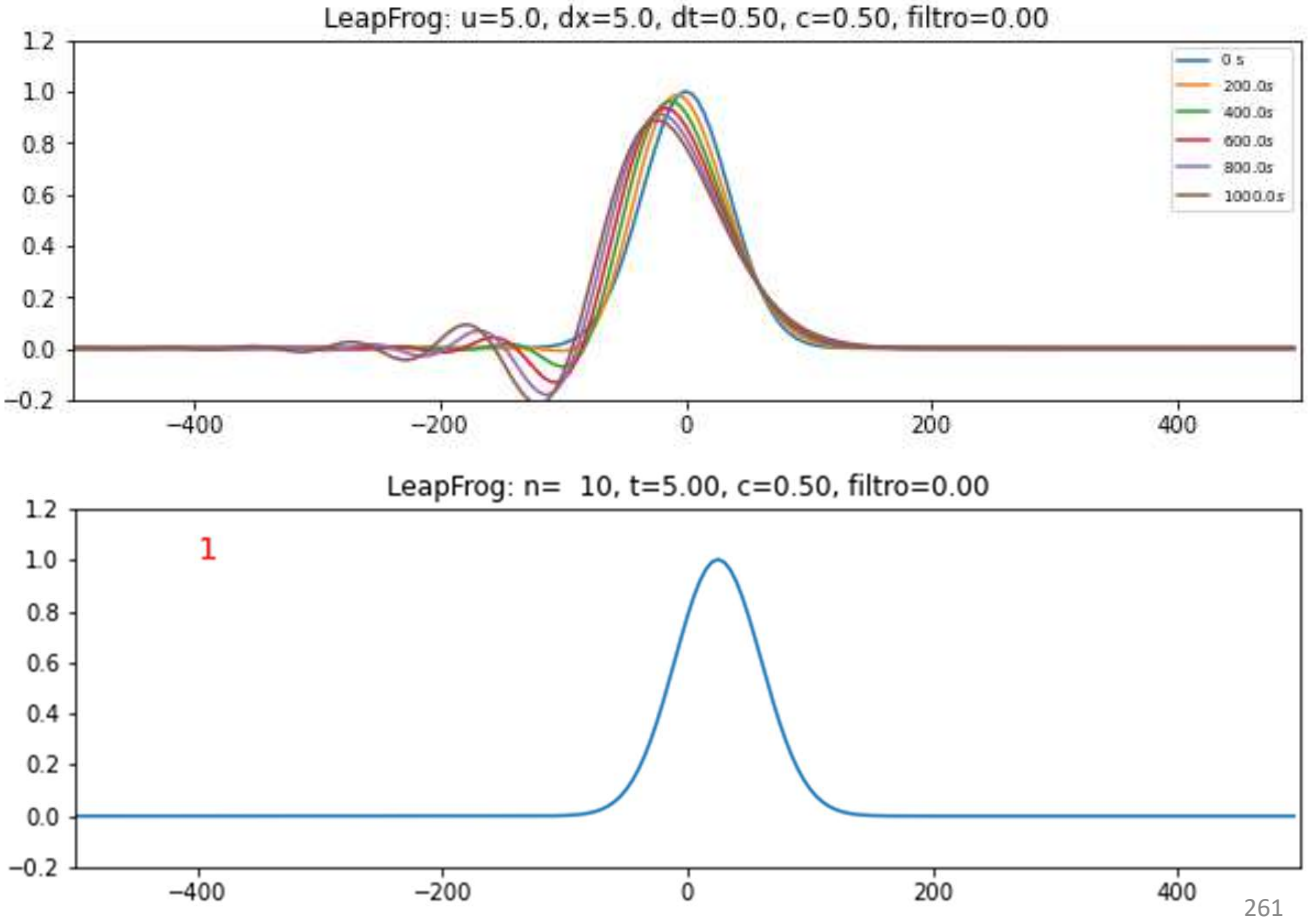
LeapFrog: $T_k^{n+1} = T_k^{n-1} - u2\Delta t \frac{T_{k+1}^n - T_{k-1}^n}{2\Delta x}$

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

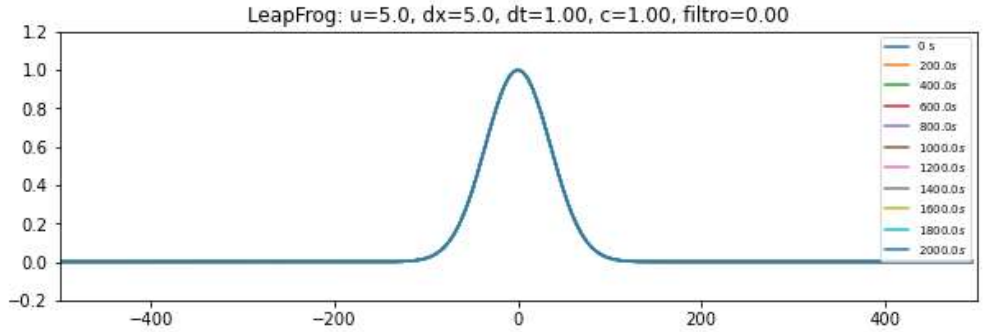
Leapfrog $c = 0.5$ dispersão numérica

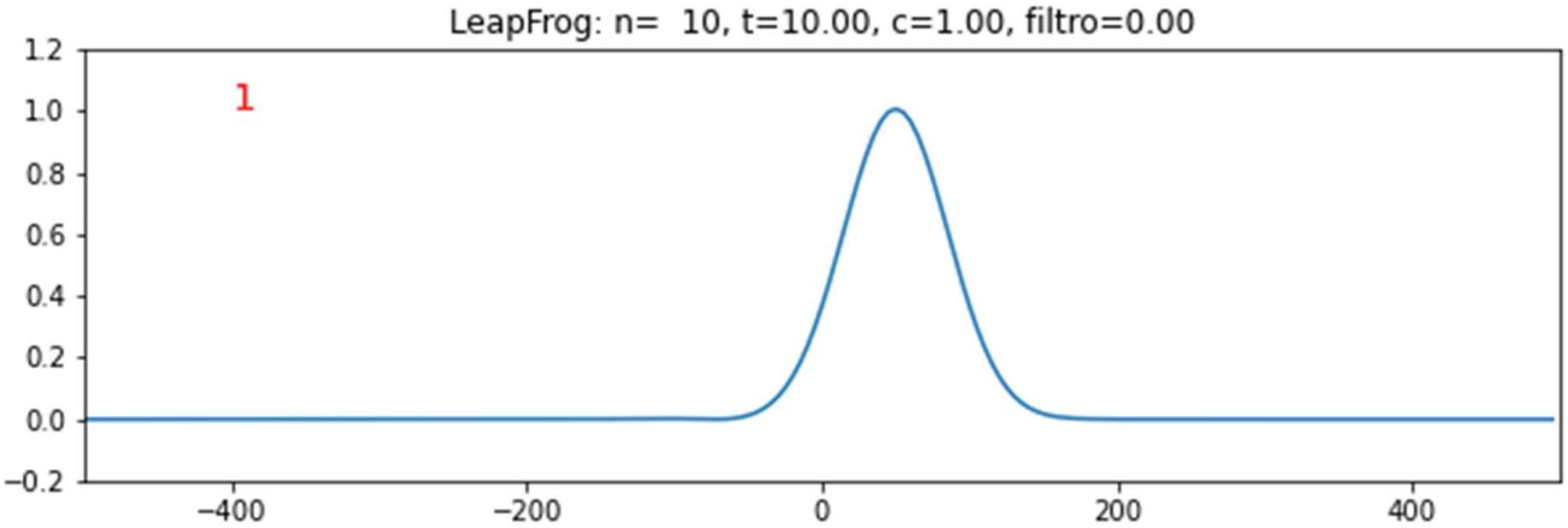
Ao longo do tempo a bossa vai-se allargando, com oscilações de pequeno comprimento de onda a deslocarem-se mais lentamente: a isto chama-se **dispersão** (numérica).

Num sistema dispersivo a velocidade depende do comprimento de onda.



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

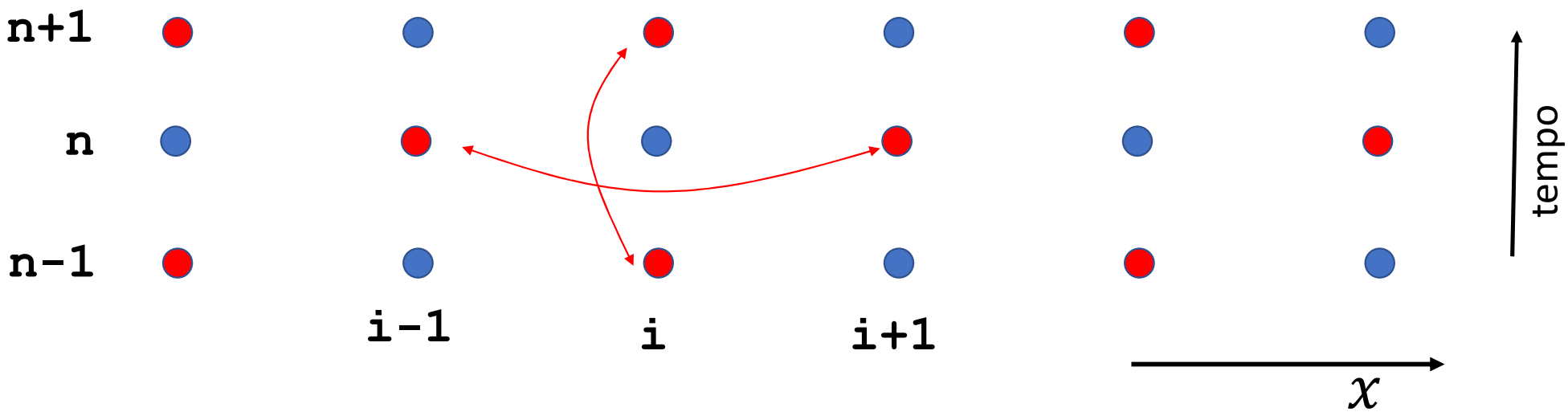




LeapFrog

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

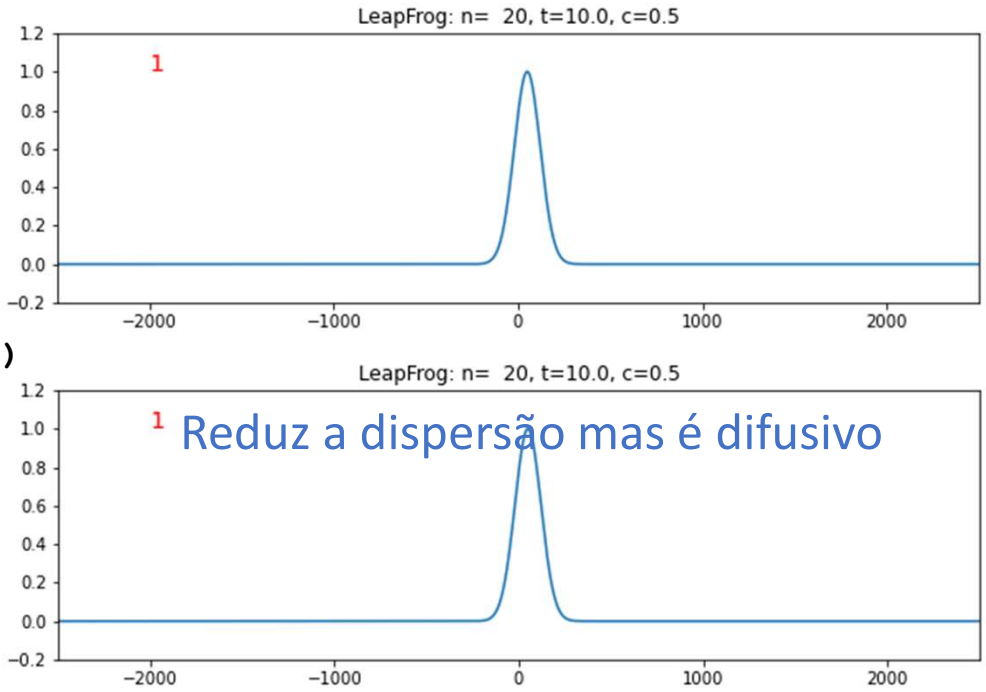
A malha computacional divide-se em conjuntos (pontos vermelhos e azuis) que não comunicam entre si. As malhas tendem a desacoplar, criando ruído.



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

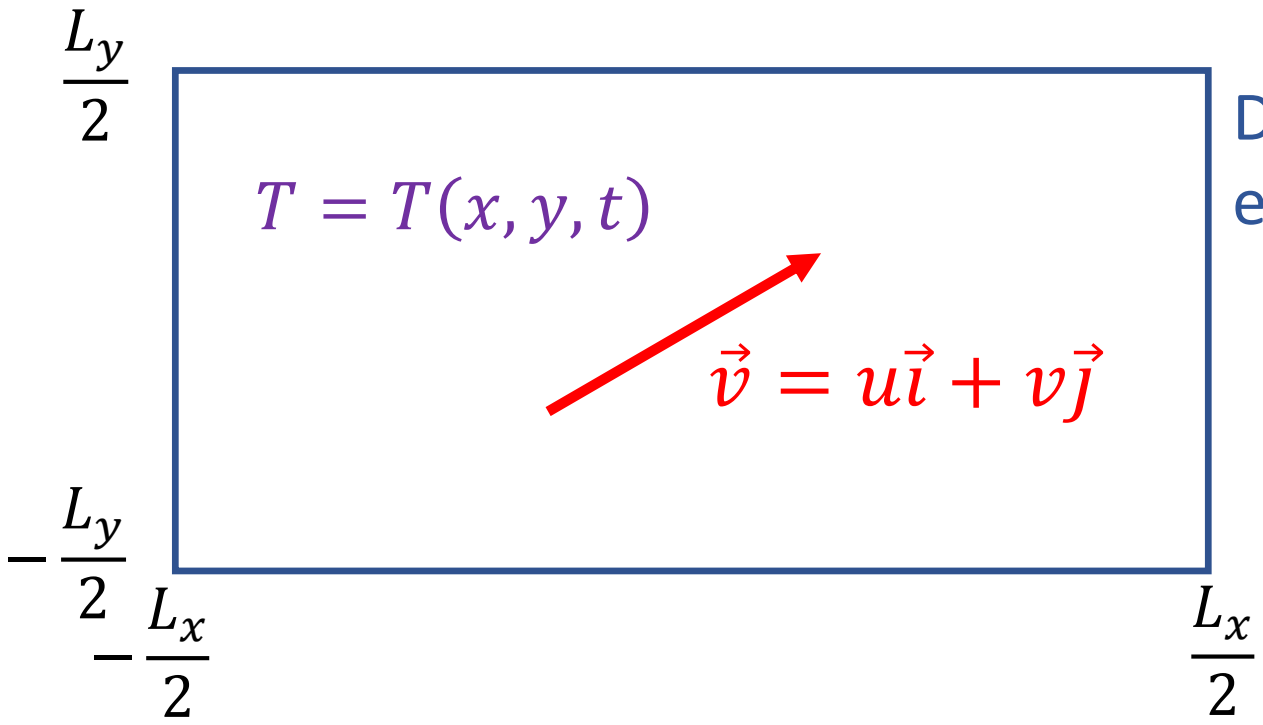
$$\frac{\partial T}{\partial t} = -u\frac{\partial T}{\partial x} - v\frac{\partial T}{\partial y} \Rightarrow \frac{\partial T}{\partial t} = -\frac{\partial uT}{\partial x} - \frac{\partial vT}{\partial y}.$$

A forma de fluxo é exata em duas condições:

$u, v = const$ (advecção linear)

ou

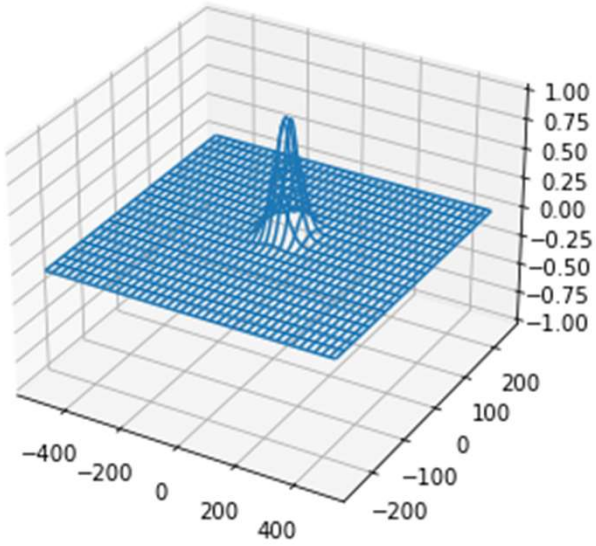
$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0$ (fluido incompressível)



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)

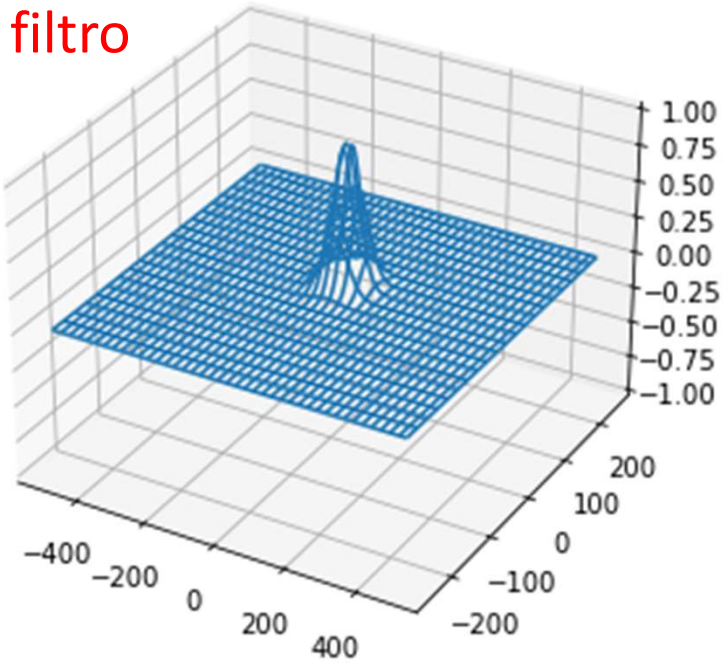
```
…
    if it%movie==0:
        tit=r'$Adv.Lin. by. LeapFrog: u=%3.1f, v=%3.1f, c=%3.2f, \
              filtro=%3.2f, t=%4.0f s $' % (u,v,courant,asel,it*dt)
        fn=graf(xis,yps,T,it,tit)
        frames.append(fn)

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

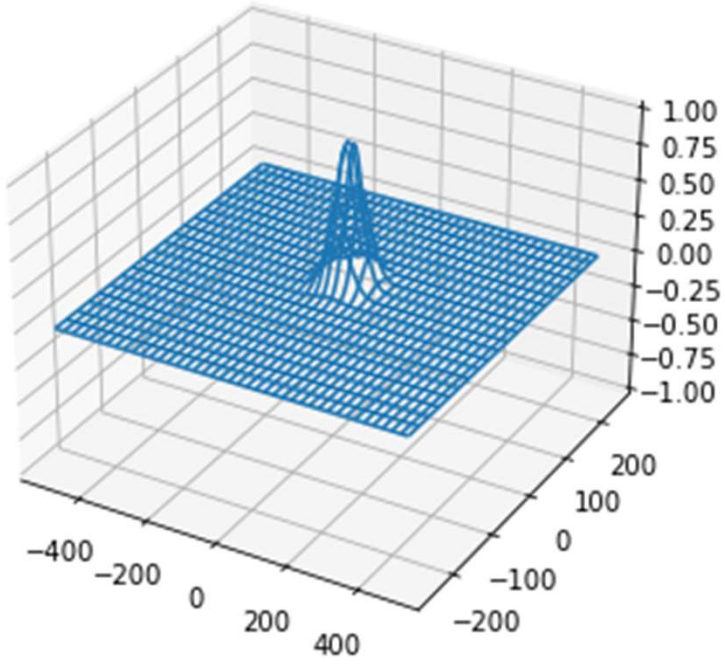
Condições cíclicas

$Adv.Lin.by.LeapFrog: u = 5.0, v = 10.0, c = 0.95, filtro = 0.0$ $Adv.Lin.by.LeapFrog: u = 5.0, v = 10.0, c = 0.95, filtro = 0.00, t = 3s$

Com filtro

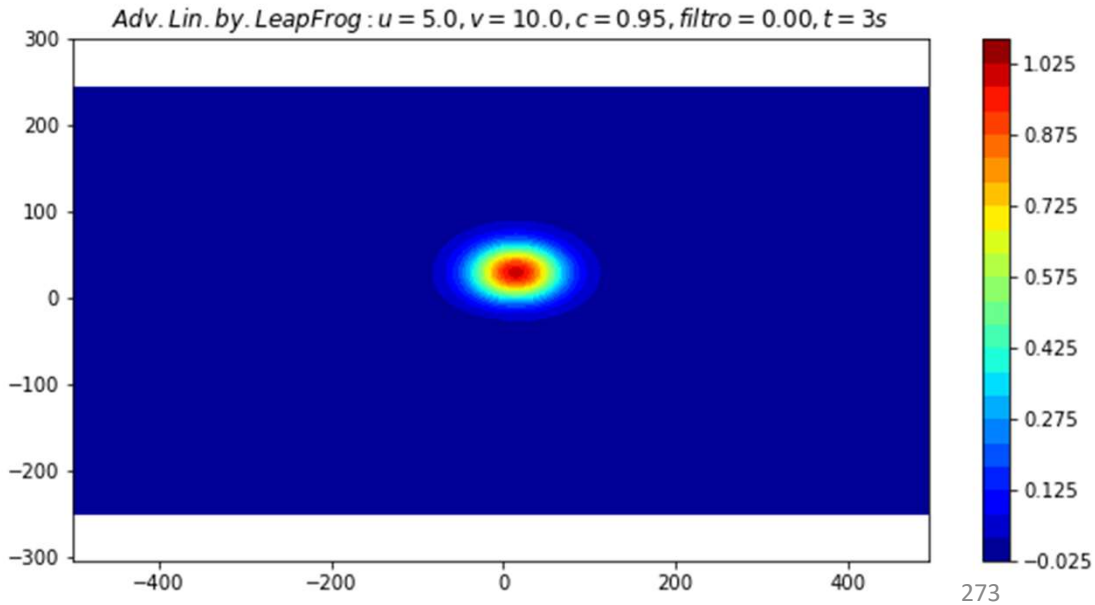


Sem filtro



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:

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

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

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

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

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

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.

Ciências ULisboa

Modelação Numérica
Aula 13a (NA)

Análise de estabilidade de esquemas numéricos pelo método de von Neumann (ou de Fourier)

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

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

Estabilidade do método de Lax

Método de Lax

$$T_m^{n+1} = \frac{1}{2}(T_{m-1}^n + T_{m+1}^n) - U\Delta t \frac{T_{m+1}^n - T_{m-1}^n}{2\Delta x}$$

Substituindo

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

Simplificando

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

$$B = \cos(k\Delta x) - i\frac{U\Delta t}{\Delta x}\sin(k\Delta x) \Longrightarrow |B| = \sqrt{\cos^2(k\Delta x) + \frac{U^2\Delta t^2}{\Delta x^2}\sin^2(k\Delta x)}$$

Logo, se

$$\frac{U^2\Delta t^2}{\Delta x^2} \leq 1 \Longrightarrow |B| \leq 1$$

O método é condicionalmente estável, será estável se satisfizer a condição CFL (Courant-Friederichs-Levy): $\frac{U\Delta t}{\Delta x} \leq 1$ (Número de Courant)

Se $\frac{U\Delta t}{\Delta x} = 1$ o método conserva a amplitude original, se $\frac{U\Delta t}{\Delta x} < 1$ o método sofre de atenuação, se $\frac{U\Delta t}{\Delta x} > 1$ o método é instável (amplitude cresce exponencialmente).

Dispersão no método de Lax

A estabilidade é uma condição necessária para um método ser usável, mas não é suficente para o caracterizar. Para recuperar a solução analítica exata, seria necessário que

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

No método de Lax temos:

$$B = \cos(k\Delta x) - i\frac{U\Delta t}{\Delta x}\sin(k\Delta x) = |B|e^{-i\frac{U\Delta t}{\Delta x}\tan(k\Delta x)}$$

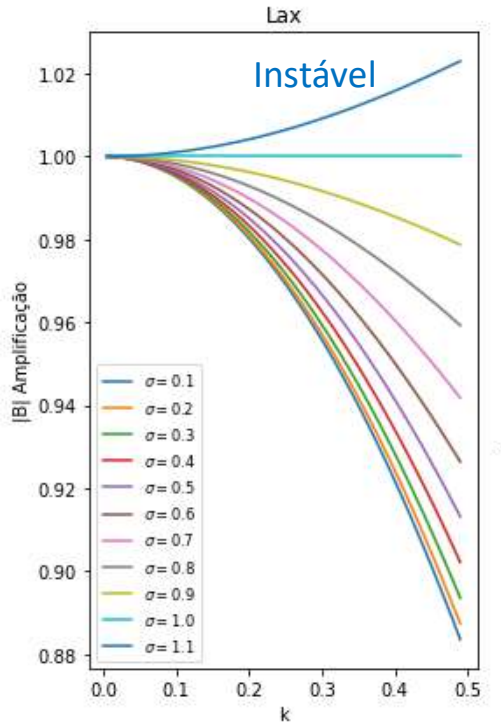
Logo, a velocidade de fase numérica será

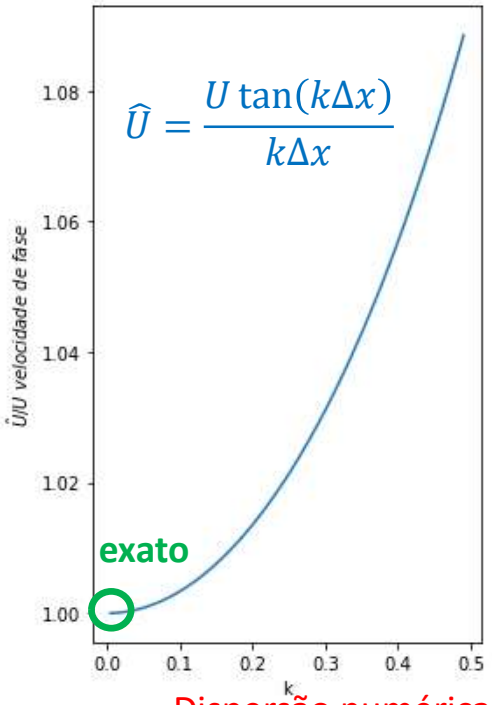
$$\widehat{U} = \frac{U\tan(k\Delta x)}{k\Delta x}$$

No $\lim_{k\Delta x\to 0} \widehat{U} = U$: o modelo aproxima-se da solução analítica nos pequenos números de onda k, i.e. nos grandes comprimentos de onda. Fora desse limite a velocidade de fase depende de k e temos dispersão numérica.

Comportamento do método de Lax

```
import numpy as np
import matplotlib.pyplot as plt
U=1;dt=1;dx=1;sigma=U*dt/dx
kNyq=1/(2*dx)
Nk=100;
k=np.arange(1,Nk-1)/Nk*kNyq
plt.figure(figsize=(8,6))
for sigma in np.arange(0.1,1.2,0.1):
Bmod=np.sqrt(np.cos(k*dx)**2+\
     sigma**2*np.sin(k*dx)**2)
plt.subplot(1,2,1);plt.plot(k,Bmod,\
     label=r'$\sigma=%2.1f$' % sigma)
plt.ylabel('|B| Amplificação');
plt.xlabel('k')
plt.legend(loc='lower left')
plt.title('Lax')
plt.subplot(1,2,2)
Urel=np.tan(k*dx)/(k*dx);plt.plot(k,Urel)
plt.ylabel(r'$\hat{U}/U$');plt.xlabel('k')
plt.tight_layout()
```





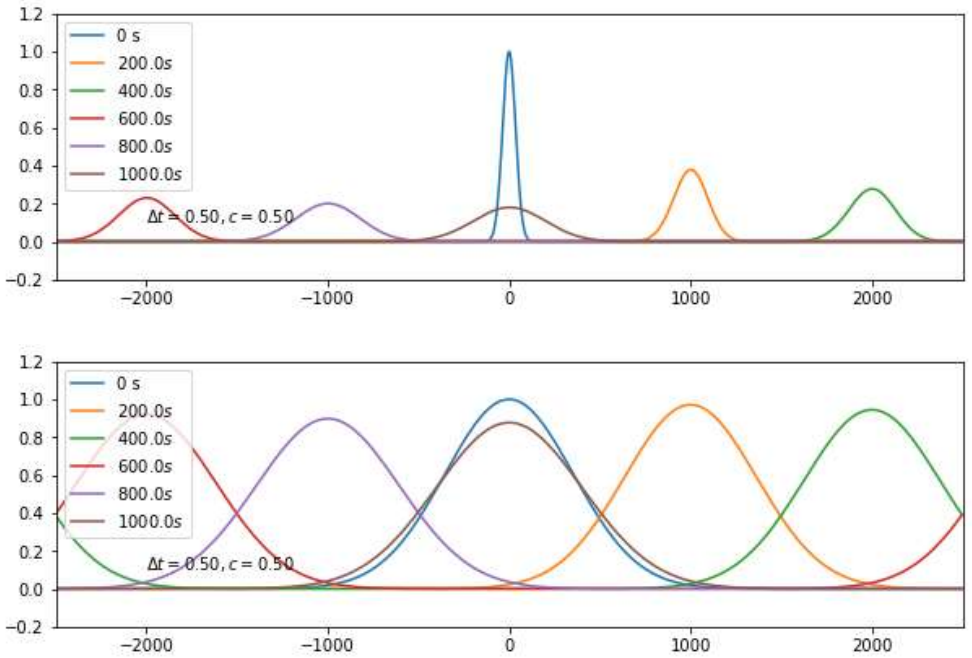
Dispersão numérica →

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

Advecção linear pelo método de Lax

No método de Lax os pequenos comprimento de onda são rapidamente atenuados, escondendo a dispersão. Os grandes c.d.o. são advectados com a velocidade de fase correta.

Integração de 1 volta ao domínio



Estabilidade do Leapfrog (análise pelo método de von Neumann)

Leapfrog

$$\frac{T_m^{n+1} - T_m^{n-1}}{2\Delta t} = -U\frac{T_{m+1}^{n} - T_{m-1}^{n}}{2\Delta x}$$

Substituindo $T_m^n = B^n e^{ikm\Delta}$:

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

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

Obtemos a equação do 2º grau para B ($\sigma = U\Delta t/\Delta x$):

$$B^2 + 2i\sigma\sin(k\Delta x)\,B - 1 = 0$$

Aplicação ao Leapfrog (2)

$$B^2 + 2i\sigma \sin(k\Delta x)\, B - 1 = 0$$

Resolvendo:

$$B = \frac{\left(-2i\sigma \sin(k\Delta x) \pm \sqrt{-4\sigma^2 \sin^2(k\Delta x) + 4}\right)}{2}$$

$$= -i\sigma \sin(k\Delta x) \pm \sqrt{1 - \sigma^2 \sin^2(k\Delta x)}$$

Se $\sigma \leq 1$ (Nºcourant) o radicando é real e:

$$|B| = \sigma^2 \sin^2(k\Delta x) + 1 - \sigma^2 \sin^2(\mathrm{k}\Delta x) = 1$$

E a amplitude é conservada (não há amplificação nem atenuação, $\forall k$).

Aplicação ao Leapfrog (3)

$$B = -i\sigma \sin(k\Delta x) \pm \sqrt{1 - \sigma^2 \sin^2(k\Delta x)}$$

com $\sigma \leq 1$ (radicando é real) $|B_{1,2}| = 1$, logo as duas raízes são:

$$B_1 = e^{-i\theta_k}$$

$$B_2 = -e^{i\theta_k}$$

onde: $\sin\theta_k = \sigma \sin(k\Delta x)$.

A fase tem um comportamento complicado, devido á existência de duas raízes na equação do 2º grau.

A solução geral será uma combinação linear das soluções parcelares, com duas constantes a determinar pelas condições iniciais: (n é o passo de tempo)

$$T_m^n = (CB_1^n + DB_2^n)e^{ikm\Delta x} = \left(Ce^{-in\theta_k} + D\left(-e^{i\theta_k}\right)^n\right)e^{ikm\Delta x}$$

Atenção aos () !
n PAR vs n ÍMPAR

Aplicação ao Leapfrog: solução geral

$$T_m^n = (CB_1^n + DB_2^n)e^{ikm\Delta} = \left(Ce^{-in\theta_k} + D\left(-e^{i\theta_k}\right)^n\right)e^{ikm\Delta x}$$
$$= Ce^{i(km\Delta x - n\theta_k)} + D(-1)^n e^{i(km\Delta x + n\ \ _k)}$$

Onda progressiva — Onda regressiva, com amplitude a mudar de sinal entre passos de tempo

MODO FÍSICO — **MODO Computacional**

A solução é estável quando $n \to \infty$, se $\theta_k \in \mathbb{R}$, i.e. $\sigma \leq 1$ (Número de Courant)

Solução exata seria:

$$T_m^n = e^{ik(m\Delta x - Un\Delta t)}$$

Aplicação ao Leapfrog: condições iniciais

$$T_m^n = (CB_1^n + DB_2^n)e^{ikm\Delta} = \left(Ce^{-in\theta_k} + D\left(-e^{i\theta_k}\right)^n\right)e^{ikm\Delta x}$$

Condição incial:

$$T(x = m\Delta x, t = 0) \approx T_m^0 = e^{ikm\Delta x} \Longrightarrow (C + D) = 1$$

Falta uma condição que terá de ser determinada pelo método usado na primeira iteração, visto que o leapfrog só pode ser usado na segunda… Se a primeira iteração usar o método de Euler:

$$\frac{T_m^{n+1} - T_m^n}{\Delta t} = -U\frac{T_{m+1}^n - T_{m-1}^n}{2\Delta x} \Longrightarrow T_m^1 = T_m^0 - \frac{U\Delta t}{2\Delta x}(\mathrm{T_{m+1}^0 - T_{m-1}^0})$$

$$(CB_1 + DB_2)e^{ikm\Delta} = e^{ikm\Delta x} - \frac{U\Delta t}{2\Delta x}\left(e^{ik(m+1)\Delta x} - e^{-ik(\mathrm{m-1})\Delta x}\right)$$

Aplicação ao Leapfrog (6)

$(C + D) = 1,$

$$\left(Ce^{-i\theta_k} - De^{i\theta_k}\right)e^{ikm\Delta x} = e^{ikm\Delta x} - \frac{U\Delta t}{2\Delta x}\left(e^{ik(m+1)\Delta x} - e^{ik(\mathrm{m}-1)\Delta x}\right)$$

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

Logo:

$$\left(Ce^{-i\theta_k} - (1 - C)e^{i\theta_k}\right) = 1 - i\sin\theta_k$$

$$C\left(e^{-i\theta_k} + e^{i\theta_k}\right) - e^{i\theta_k} = 1 - i\sin\theta_k$$

$$2C\cos\theta_k = \cos\theta_k + i\sin\theta_k + 1 - i\sin\theta_k$$

$$C = \frac{1 + \cos\theta_k}{2\cos\theta_k}$$

Leapfrog: Solução completa

$$T_m^n = \frac{1+\cos\theta_k}{2\cos\theta_k} e^{i(km\Delta x - n\theta_k)} - \frac{1-\cos\theta_k}{2\cos\theta_k}(-1)^n e^{i(km\Delta x + n\theta_k)}$$

Com $\theta_k = \sin^{-1}(\sigma \sin(k\Delta x))$, $\sigma = \frac{U\Delta t}{\Delta x} \leq 1$ (condição CFL)

Se $\sigma = 1$, $\theta_k = k\Delta x = kU\Delta t$, e o modo físico propaga-se à velocidade da solução exata, e o modo numérico propaga-se à mesma velocidade mas na direção oposta. A amplitude relativa dos dois modos depende do número de onda.

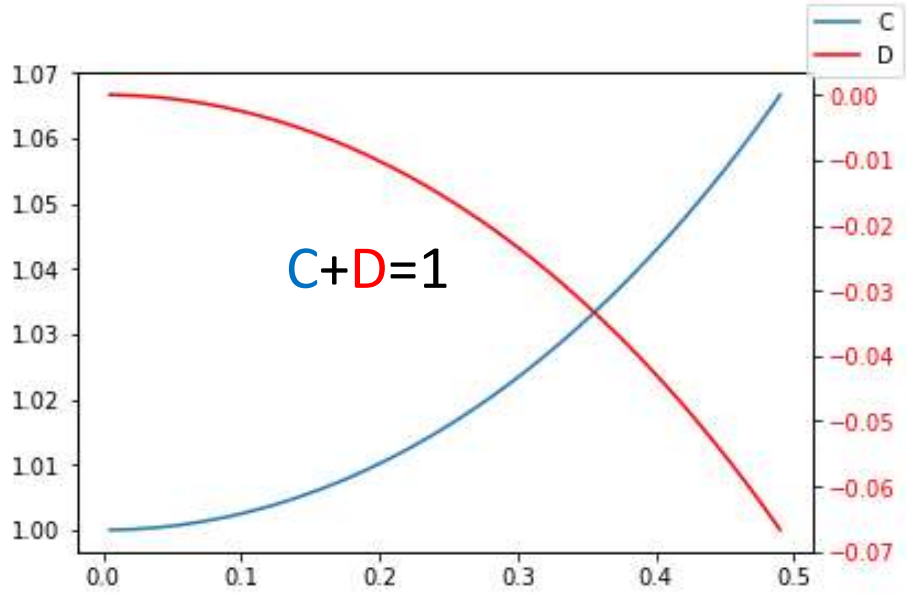
Os pequenos k (grandes cdo) são melhor representados.

O método é dispersivo: a velocidade de fase (que devia ser constante) depende de k.

Cálculo de C,D

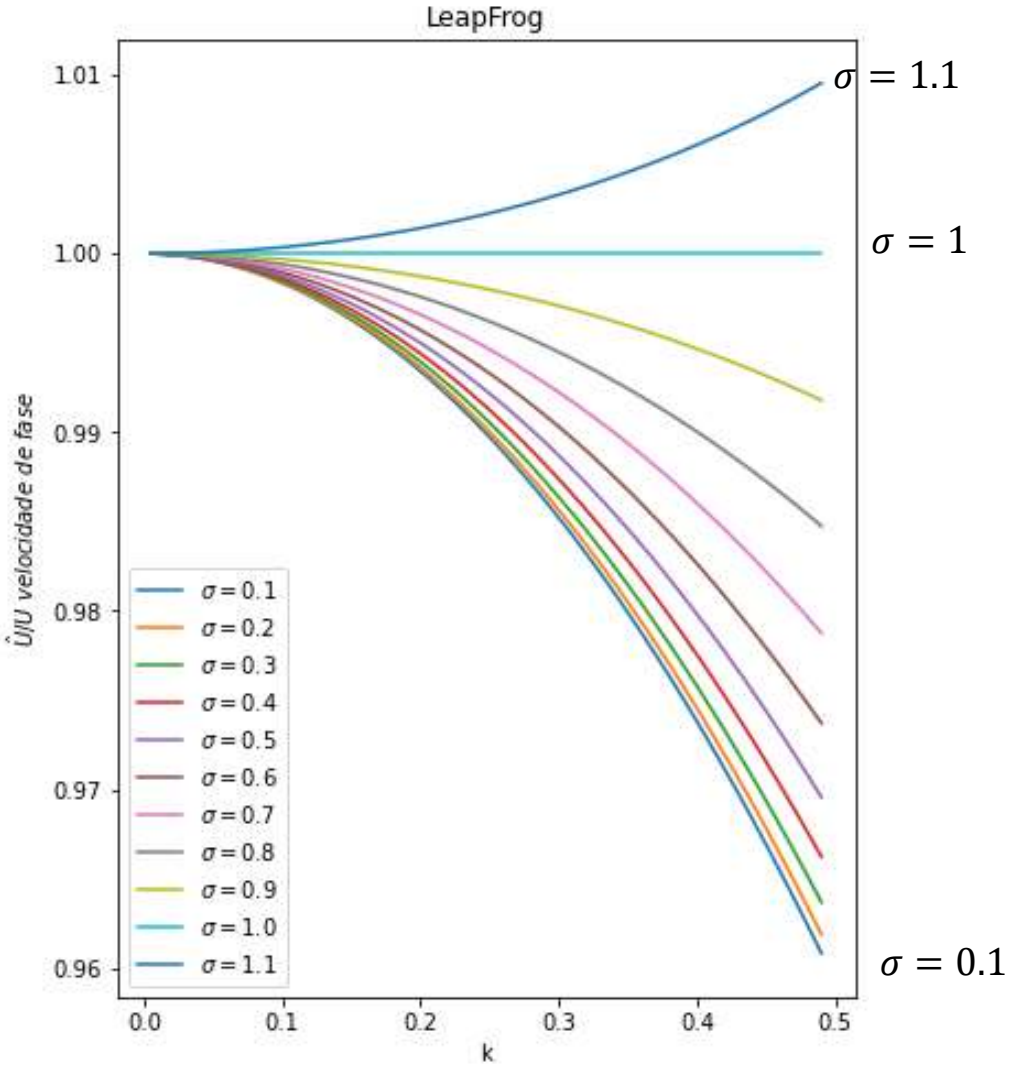
```
fig,ax1=plt.subplots()
U=1;dt=1;dx=1;sigma=U*dt/dx
kNyq=1/(2*dx)
Nk=100
k=np.arange(1,Nk-1)/Nk*kNyq
thetak=np.arcsin(sigma*np.sin(k*dx));
C=(1+np.cos(thetak))/(2*np.cos(thetak));
D=1-C
ax1.plot(k,C,label='C')
ax2=ax1.twinx()
ax2.plot(k,D,label='D',color='red')
ax2.set_xlabel('k')

ax2.tick_params(axis='y', labelcolor='red')
fig.legend()
```



Velocidade de fase (Û/U)

```
plt.figure(figsize=(6,8))
for sigma in np.arange(0.1,1.2,0.1):
    dt=sigma*dx/U;
    thetak=np.arcsin(sigma*np.sin(k*dx));
    Uhat=thetak/(k*dt);
    plt.plot(k,Uhat,label=\
             r'$\sigma=%2.1f$' % sigma)
plt.ylabel(r'$\hat{U}/U\ velocidade\ de\
fase$');plt.xlabel('k')
plt.title('LeapFrog')
plt.legend()
```



Comentário final: leapfrog

A existência dos 2 “modos” resulta do desacoplamento entre as malhas pares/ímpares: os pontos vermelhos e azuis não interagem (excepto talvez na fronteira): essa a razão da necessidade do filtro temporal de Robert-Aselin.