Ciências ULisboa

Modelação Numérica
Aula 14

Equação de adveção-difusão.

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

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

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

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

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

Estabilidade oo Leapfrog (2)

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

Resolvendo:

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

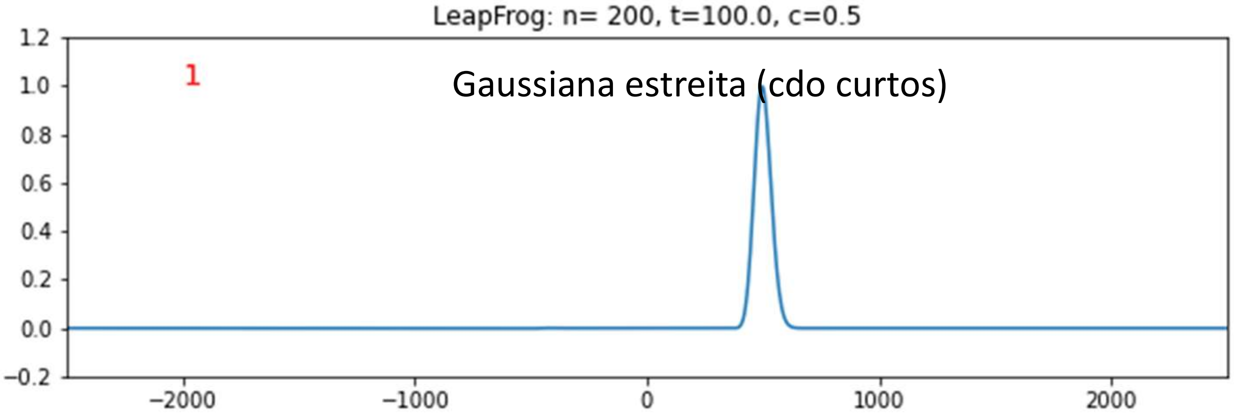
$$= -i\sigma \sin(k\Delta x) \pm \sqrt{1 - \sigma^2 \sin^2(k\Delta x)}$$

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

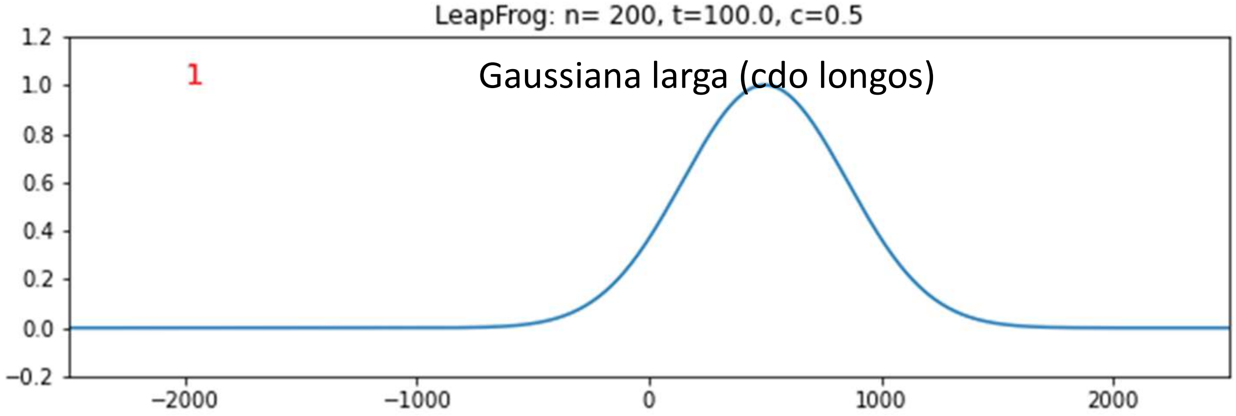
$$|B| = \sigma^2 \sin^2(k\Delta x) + 1 - \sigma^2 \sin^2(\mathrm{k}\Delta x) = 1$$

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

Advecção linear LeapFrog

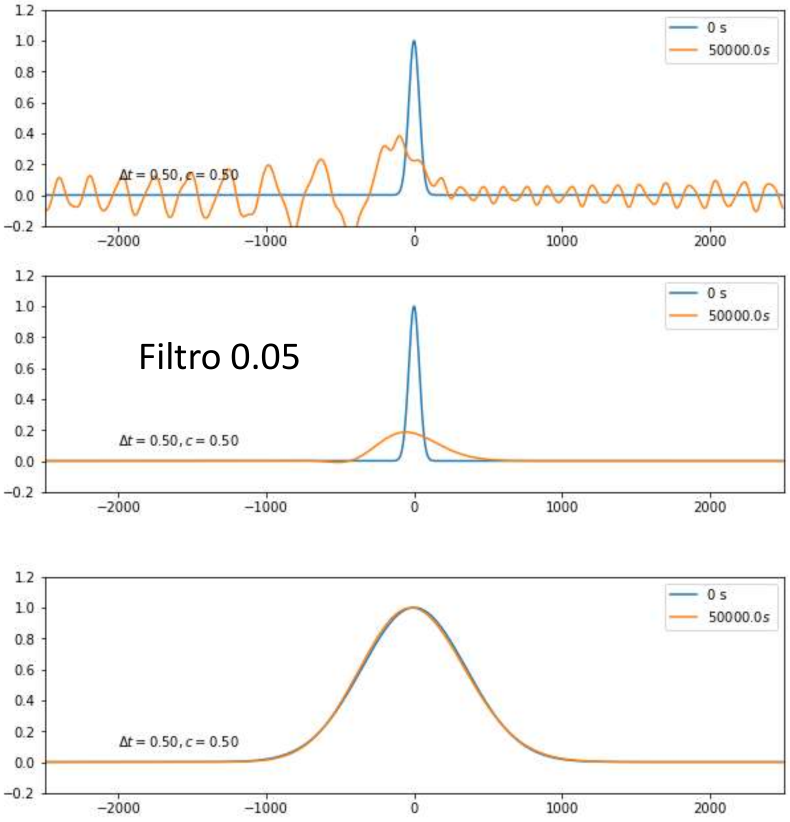

LeapFrog: n= 200, t=100.0, c=0.5

Gaussiana estreita (cdo curtos)


LeapFrog: n= 200, t=100.0, c=0.5

Gaussiana larga (cdo longos)

Após 50 voltas


Filtro 0.05

Equação de advecção-difusão (linear, 2D)

$$\frac{\partial T}{\partial t} = -u\frac{\partial T}{\partial x} - v\frac{\partial T}{\partial y} + K_D\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

$$u, v = const$$

$$K_D = const$$

É uma simplificação (2D, linear) da equação (transferência de calor num fluido sem fontes de calor):

$$\frac{\partial \theta}{\partial t} = -(\vec{v}.\nabla)\theta + \kappa_D \nabla^2 \theta$$

Casos particulares: $\vec{v} = 0$ (equação da difusão/condução); $\kappa_D = 0$ (equação da advecção)

Usando diferenças centradas

$$\frac{\partial T}{\partial t} = -u\frac{\partial T}{\partial x} - v\frac{\partial T}{\partial y} + K_D\left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}\right)$$

$$\frac{T_{i,j}^{n+1} - T_{i,j}^{n-1}}{2\Delta t} = -u\frac{T_{i+1,j}^{n} - T_{i-1,j}^{n}}{2\mathit{\Delta} x} - u\frac{T_{i,j+1}^{n} - T_{i,j-1}^{n}}{2\mathit{\Delta} x} +$$
$$K_D\left(\frac{T_{i-1,j}^{n} + T_{i+1,j}^{n} - 2T_{i,j}^{n}}{\mathit{\Delta} x^2} + \frac{T_{i,j-1}^{n} + T_{i,j+1}^{n} - 2T_{i,j}^{n}}{\mathit{\Delta} y^2}\right)$$

Não funciona. É instável.

Usando diferenças centradas (Leapfrog explicito com difusão em $\boldsymbol{n-1}$)

$$T_{i,j}^{n+1} = T_{i,j}^{n-1} +$$

$$+2\Delta t \left(-u \frac{T_{i+1,j}^{n} - T_{i-1,j}^{n}}{2\Delta x} - u \frac{T_{i,j+1}^{n} - T_{i,j-1}^{n}}{2\Delta x} \right.$$

$$\left. + \; K_D \left(\frac{T_{i-1,j}^{n-1} + T_{i+1,j}^{n-1} - 2T_{i,j}^{n-1}}{\Delta x^2} + \frac{T_{i,j-1}^{n-1} + T_{i,j+1}^{n-1} - 2T_{i,j}^{n-1}}{\Delta y^2} \right) \right)$$

É condicionalmente estável.

FTCS (explicito com difusão em $\boldsymbol{n}$)

$$T_{i,j}^{n+1} = T_{i,j} +$$

$$+\Delta t \left(-u \frac{T_{i+1,j}^{n} - T_{i-1,j}^{n}}{2\Delta x} - u \frac{T_{i,j+1}^{n} - T_{i,j-1}^{n}}{2\Delta x} \right.$$

$$\left. + \ K_D \left(\frac{T_{i-1,j}^{n} + T_{i+1,j}^{n} - 2T_{i,j}^{n}}{\Delta x^2} + \frac{T_{i,j-1}^{n} + T_{i,j+1}^{n} - 2T_{i,j}^{n}}{\Delta y^2} \right) \right)$$

É condicionalmente estável.

Usando diferenças centradas (implícito com difusão em $\boldsymbol{n+1}$)

$$T_{i,j}^{n+1} - 2\Delta t\, K_D \left(\frac{T_{i-1,j}^{n+1} + T_{i+1,j}^{n+1} - 2T_{i,j}^{n+1}}{\mathit{\Delta} x^2} + \frac{T_{i,j-1}^{n+1} + T_{i,j+1}^{n+1} - 2T_{i,j}^{n+1}}{\mathit{\Delta} y^2} \right)$$

$$= T_{i,j}^{n-1} + 2\Delta t \left(-u \frac{-T_{i+1,j}^{n} T_{i-1,j}^{n}}{2\mathit{\Delta} x} - v \frac{T_{i,j+1}^{n} - T_{i,j-1}^{n}}{2\mathit{\Delta} y} \right)$$

É absolutamente estável, mas requere a solução de um sistema de (muitas) equações simultâneas.

Adveção-difusão

```
import numpy as np
import matplotlib.pyplot as plt
def advdif(imethod,asel,u,kD,nt,dt,passo):
    methods=['FTCS','LF']
    method=methods[imethod]
    timefil=[asel,1-2*asel,asel]
    nx=500;dx=1;Lx=nx/2*dx
    c0=u*dt/dx
    c1=u*dt/(2*dx)
    c2=kD*dt/dx**2
    c2LF=kD*2*dt/dx**2
    x=np.arange(-Lx,Lx,dx)
    …
```

…

```
if u>0:
    nvolta=int(2*Lx/u/dt)
    nt=nvolta*2+1
    passo=nvolta
Wx=10;x0=0
TI=np.exp(-((x-x0)/Wx)**2)
T=np.copy(TI);TP=np.copy(T);TM=np.copy(T)
plt.plot(x,T,label='0')
meanT=np.mean(T)
plt.plot([-Lx,Lx],[meanT,meanT],color='gray',linestyle='dashed',alpha=0.5,label='Mean')
if method=='LF':
    plt.title(r'$%5s,u=%3.1f,k_D=%3.2f,\sigma=%3.2f,\sigma_k=%3.2f$' % (method,u,kD,c0,c2LF))
else:
    plt.title(r'$%5s,u=%3.1f,k_D=%3.2f,\sigma=%3.2f,\sigma_k=%3.2f$' % (method,u,kD,c1,c2))
it0=1
if method=='LF':
    it0=2
    for ix in range(nx):
        ixm=ix-1;ixp=ix+1
        if ixm<0:
            ixm=nx-1
        elif ixp>nx-1:
            ixp=0
        T[ix]=0.5*(TM[ixm]+TM[ixp])-u*dt/dx*(TM[ixp]-TM[ixm])\
                   +kD*dt*((TM[ixm]+TM[ixp]-2*TM[ix])/dx**2)
```

```
 for it in range(it0,nt):
     for ix in range(nx):
        ixm=ix-1;ixp=ix+1
        if ixm<0: #fronteira cíclica
            ixm=nx-1
        elif ixp>nx-1:
            ixp=0
        if method=='FTCS':
            TP[ix]=T[ix]-u*dt/(2*dx)*(T[ixp]-T[ixm])\
                        +kD*dt*((T[ixm]+T[ixp]-2*T[ix])/dx**2)
        elif method=='LF':
            TP[ix]=TM[ix]-u*dt/dx*(T[ixp]-T[ixm])\
                         +kD*2*dt*((TM[ixm]+TM[ixp]-2*TM[ix])/dx**2)
    if method=='LF':
        T=timefil[0]*TM+timefil[1]*T+timefil[2]*TP #filtro temporal
    TM=np.copy(T);T=np.copy(TP) #update temporal
    if it%passo==0:
        plt.plot(x,T,label=str(it))
plt.legend()
return c0,c1,c2
```

Teste de sensibilidade

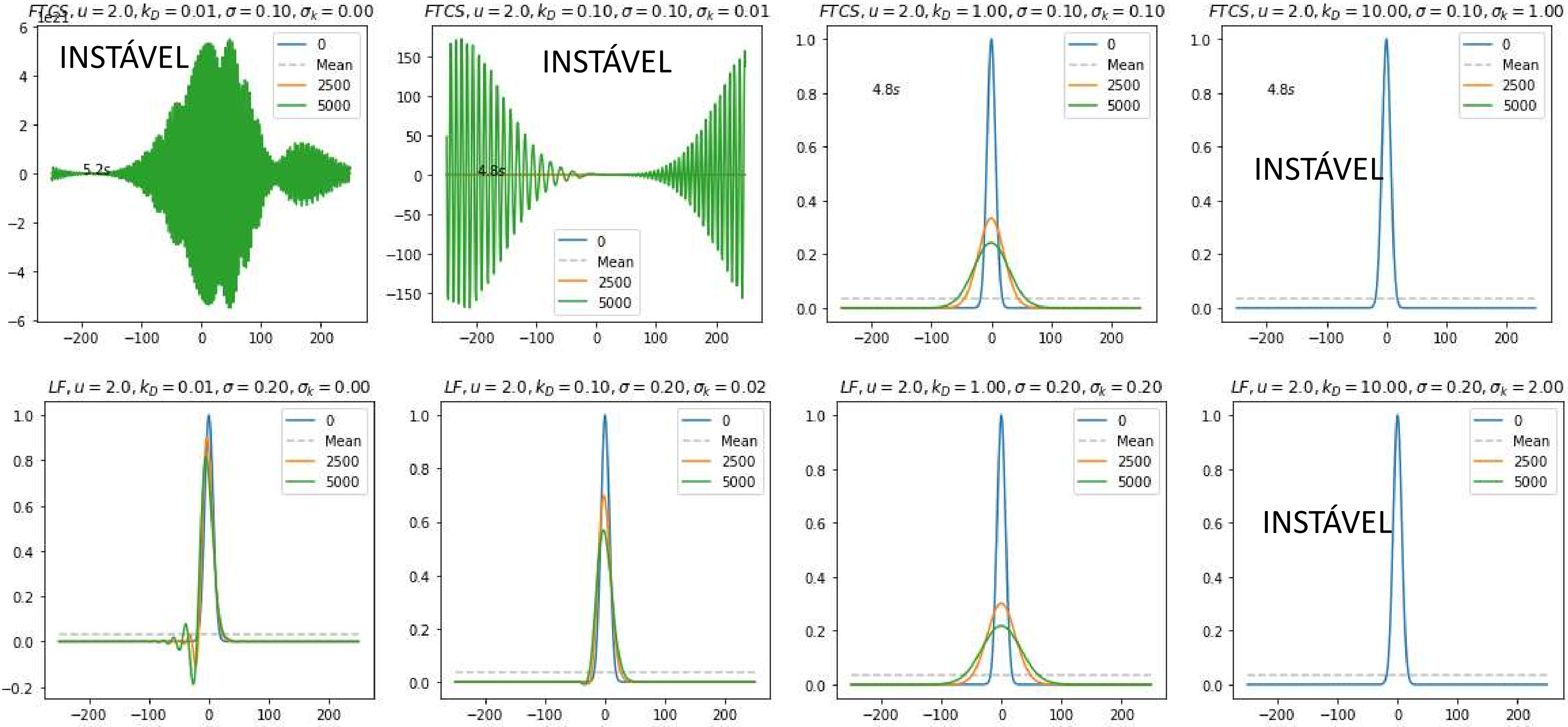
```
asel=0.05; #asel=0.
nt=5001;passo=2500;dt=0.1
for u in[0,2]:
    for imethod in[0,1]:
        kP=0
        plt.figure(figsize= (20,4))
        for kD in[0.01,0.1,1,10]:
            kP=kP+1
            plt.subplot(1,4,kP)
            c0,c1,c2=advdif(imethod,asel,u,kD,nt,dt,passo)
```

$u = 2, \Delta x = 1, \Delta t = 0.1$

$k_D \rightarrow$

FTCS

Leapfrog s/filtro

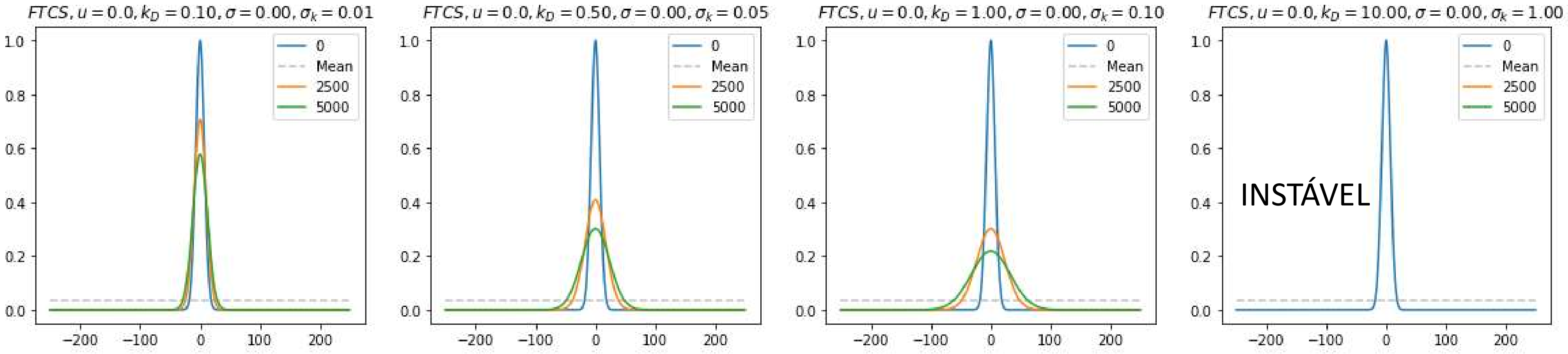


Alguma difusão estabiliza o FCTS e reduz a dispersão no leapfrog. **Muita difusão**: instável

$k_D \rightarrow$

$u = 0$

FTCS



$FTCS, u = 0.0, k_D = 0.10, \sigma = 0.00, \sigma_k = 0.01$

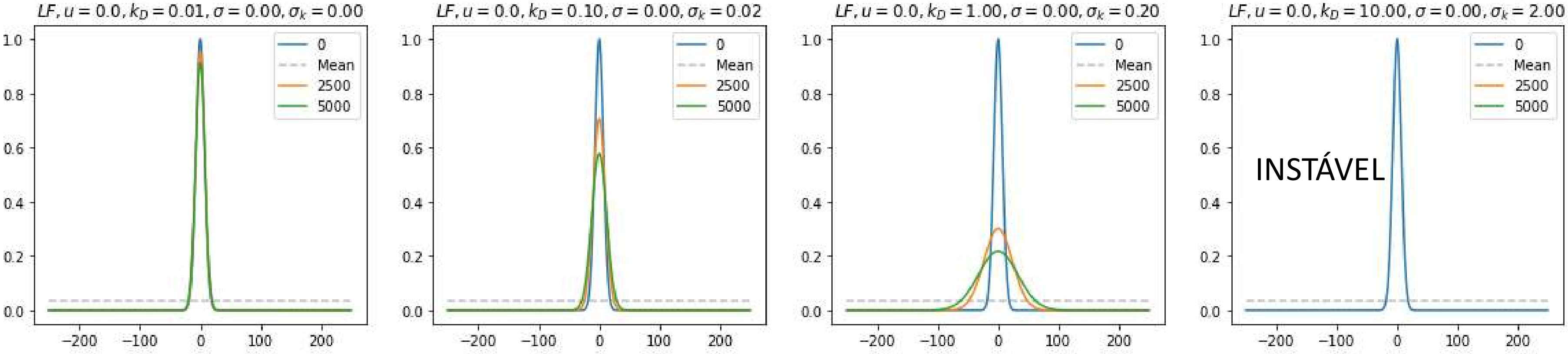
$FTCS, u = 0.0, k_D = 0.50, \sigma = 0.00, \sigma_k = 0.05$

$FTCS, u = 0.0, k_D = 1.00, \sigma = 0.00, \sigma_k = 0.10$

$FTCS, u = 0.0, k_D = 10.00, \sigma = 0.00, \sigma_k = 1.00$

INSTÁVEL

Leapfrog s/ filtro



$LF, u = 0.0, k_D = 0.01, \sigma = 0.00, \sigma_k = 0.00$

$LF, u = 0.0, k_D = 0.10, \sigma = 0.00, \sigma_k = 0.02$

$LF, u = 0.0, k_D = 1.00, \sigma = 0.00, \sigma_k = 0.20$

$LF, u = 0.0, k_D = 10.00, \sigma = 0.00, \sigma_k = 2.00$

INSTÁVEL

Estabilidade da solução da equação de adveção difusão

Leapfrog:

$$T_{i,j}^{n+1} = T_{i,j}^{n-1} - \frac{u\Delta t}{\Delta x}\left(T_{i+1,j}^{n} - T_{i-1,j}^{n}\right) + \frac{K_D 2\Delta t}{\Delta x^2}\left(T_{i-1,j}^{n-1} + T_{i+1,j}^{n-1} - 2T_{i,j}^{n-1}\right)$$

FCTS:

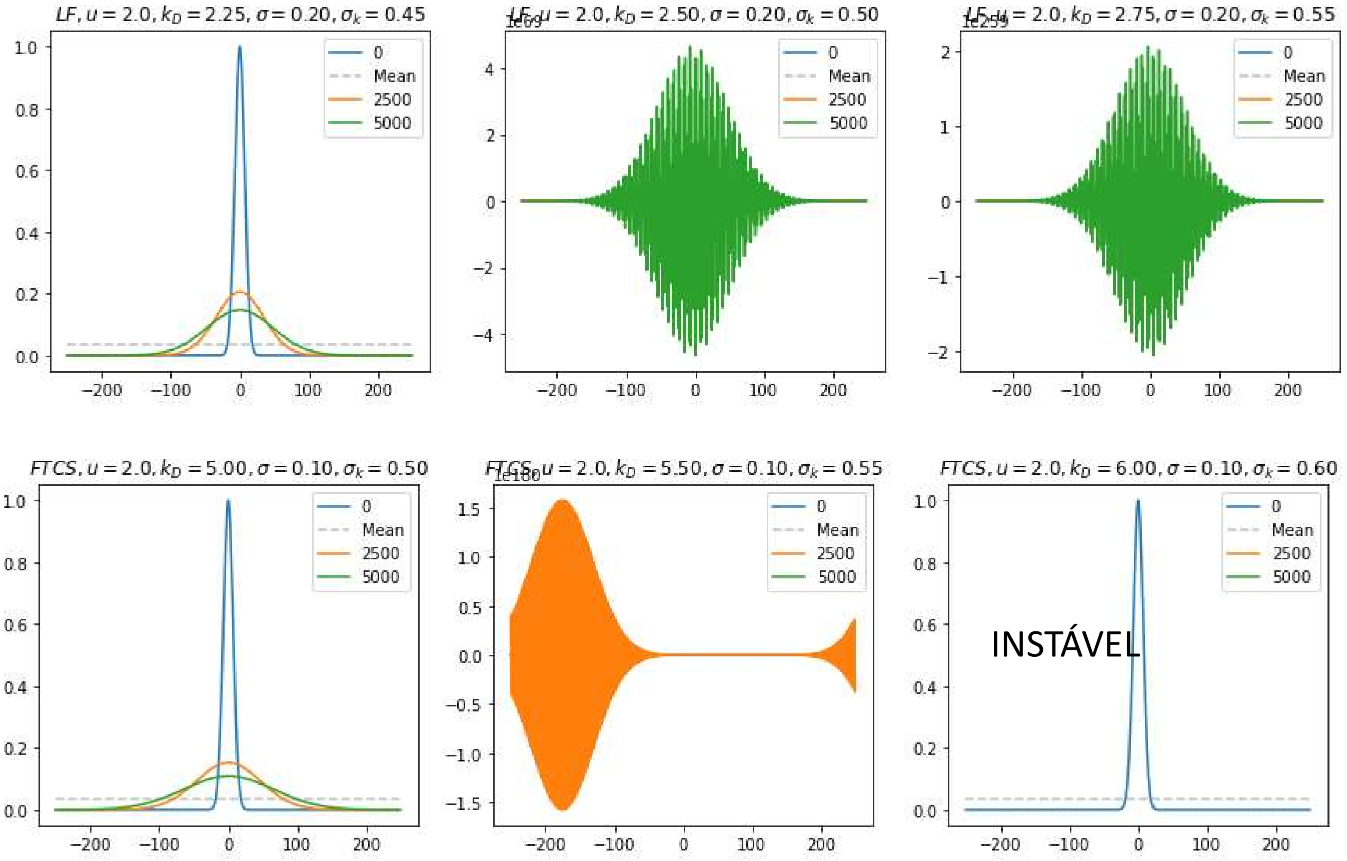
$$T_{i,j}^{n+1} = T_{i,j}^{n} - \frac{u\Delta t}{2\Delta x}\left(T_{i+1,j}^{n} - T_{i-1,j}^{n}\right) + \frac{K_D \Delta t}{\Delta x^2}\left(T_{i-1,j}^{n} + T_{i+1,j}^{n} - 2T_{i,j}^{n}\right)$$

A estabilidade depende dos dois números adimensionais

$\sigma = \frac{u\Delta t}{\Delta x} \leq 1$ *(Número de Courant),*$\sigma_k = \frac{K_D 2\Delta t}{\Delta x^2} \leq \frac{1}{2}$ (Leapfrog)

$\sigma = \frac{u\Delta t}{2\Delta x} \leq 1$, $\sigma_k = \frac{K_D \Delta t}{\Delta x^2} \leq \frac{1}{2}$ *(FTCS)*

Teste de sensibilidade a k_D: condição $\sigma_k \leq 0.5$



LF, $u = 2.0, k_D = 2.25, \sigma = 0.20, \sigma_k = 0.45$

LF, $u = 2.0, k_D = 2.50, \sigma = 0.20, \sigma_k = 0.50$

LF, $u = 2.0, k_D = 2.75, \sigma = 0.20, \sigma_k = 0.55$

FTCS, $u = 2.0, k_D = 5.00, \sigma = 0.10, \sigma_k = 0.50$

FTCS, $u = 2.0, k_D = 5.50, \sigma = 0.10, \sigma_k = 0.55$

FTCS, $u = 2.0, k_D = 6.00, \sigma = 0.10, \sigma_k = 0.60$

INSTÁVEL

Leapfrog

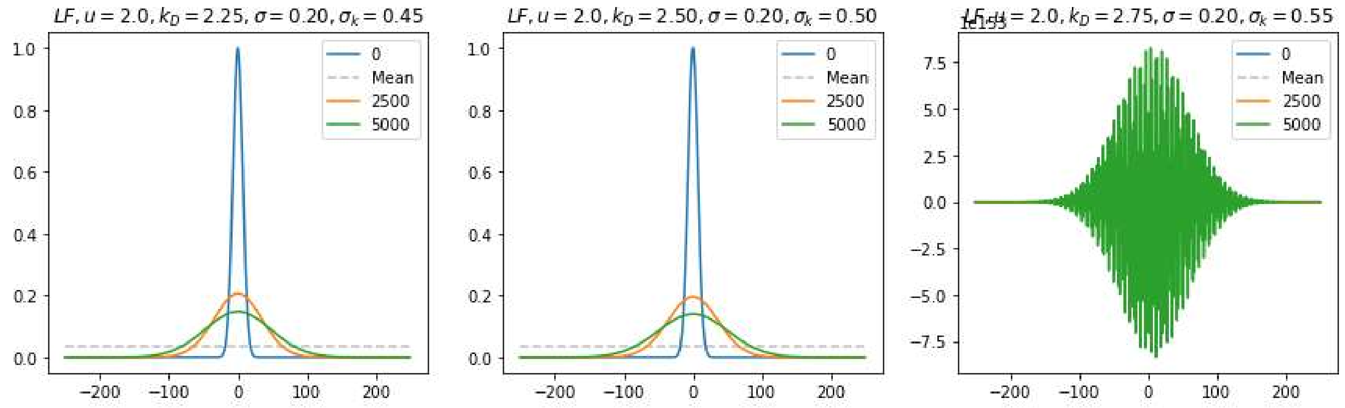
$$\sigma_k = \frac{K_D 2\Delta t}{\Delta x^2}$$

FTCS

$$\sigma_k = \frac{K_D \Delta t}{\Delta x^2}$$

Teste de sensibilidade a k_D (leapfrog)

$$\sigma_k = \frac{K_D 2\Delta t}{\Delta x^2}$$



Leapfrog com filtro temporal Robert-Aselin (0.05)

Não melhora muito…

Advecção difusão em 2D (leapfrog)

$$T_{i,j}^{n+1} = T_{i,j}^{n-1} +$$

```
…
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
…
```

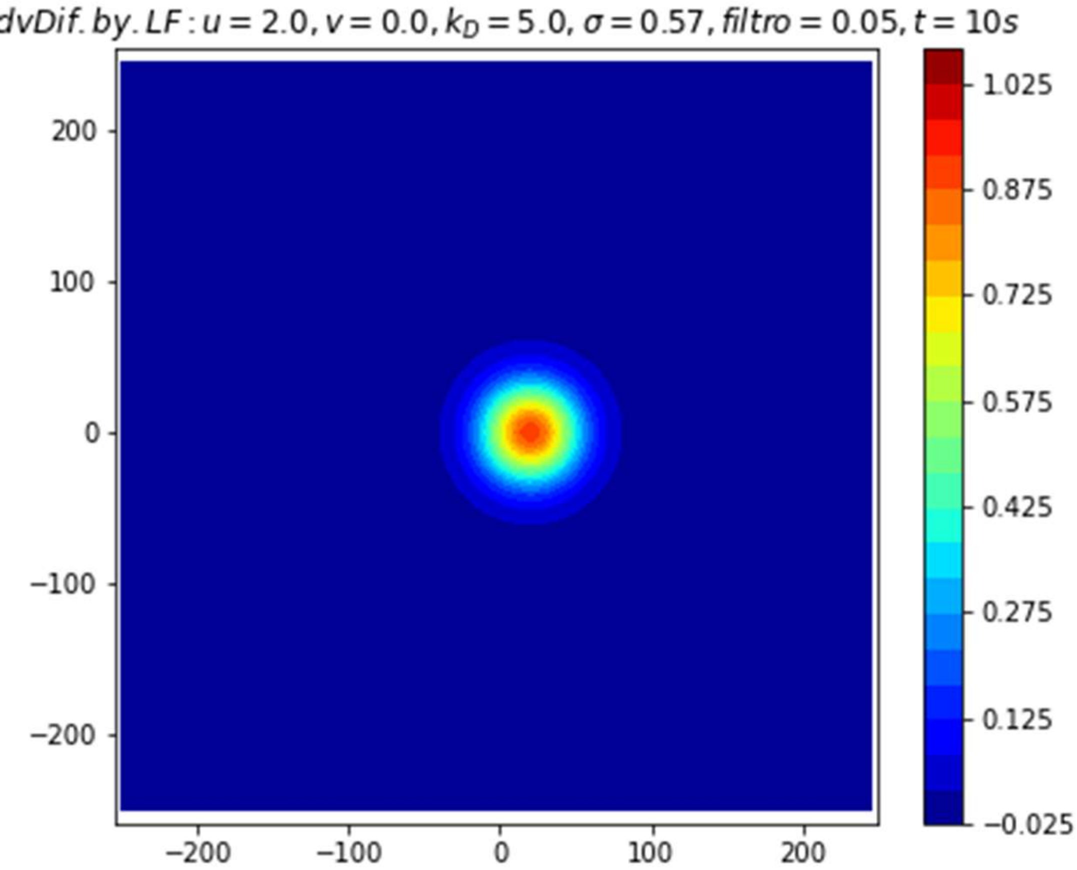
$$+2\Delta t\left(-u\frac{T_{i+1,j}^{n}-T_{i-1,j}^{n}}{2\Delta x}-u\frac{T_{i,j+1}^{n}-T_{i,j-1}^{n}}{2\Delta x}\right.$$

$$\left.+\,K_D\left(\frac{T_{i-1,j}^{n-1}+T_{i+1,j}^{n-1}-2T_{i,j}^{n-1}}{\Delta x^2}+\frac{T_{i,j-1}^{n-1}+T_{i,j+1}^{n-1}-2T_{i,j}^{n-1}}{\Delta y^2}\right)\right)$$

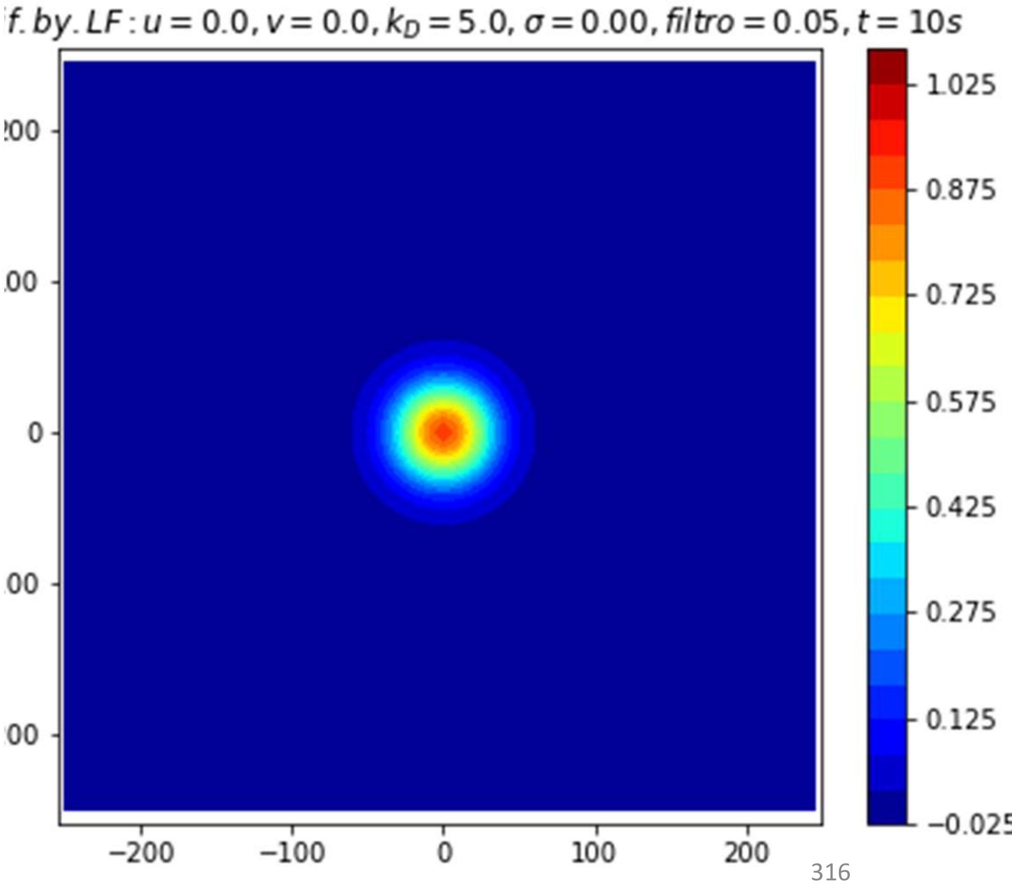
```
…
 TP[ix,iy]=TM[ix,iy]-u*dt/dx*(T[ixp,iy]-T[ixm,iy])-v*dt/dy*(T[ix,iyp]-T[ix,iym])\
+kD*2*dt*((TM[ixm,iy]+TM[ixp,iy]-2*TM[ix,iy])/dx**2+(TM[ix,iym]+TM[ix,iyp]-2*TM[ix,iy])/dy**2)
 …
```

Caso 2D Leapfrog

$u = 0$

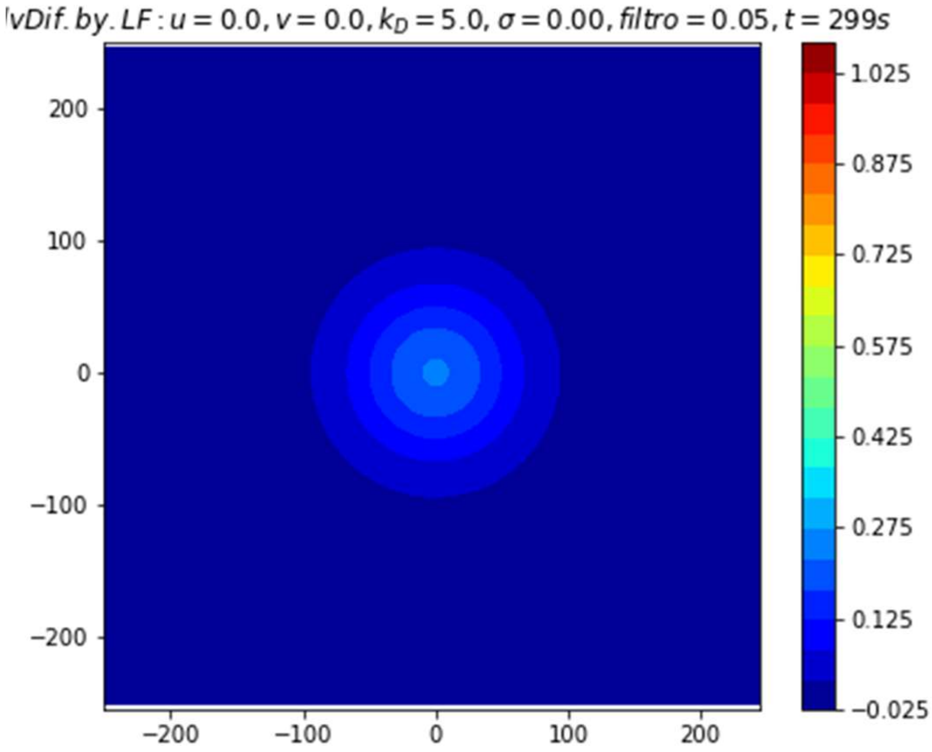


dvDif. by. LF : $u = 2.0, v = 0.0, k_D = 5.0, \sigma = 0.57, filtro = 0.05, t = 10s$

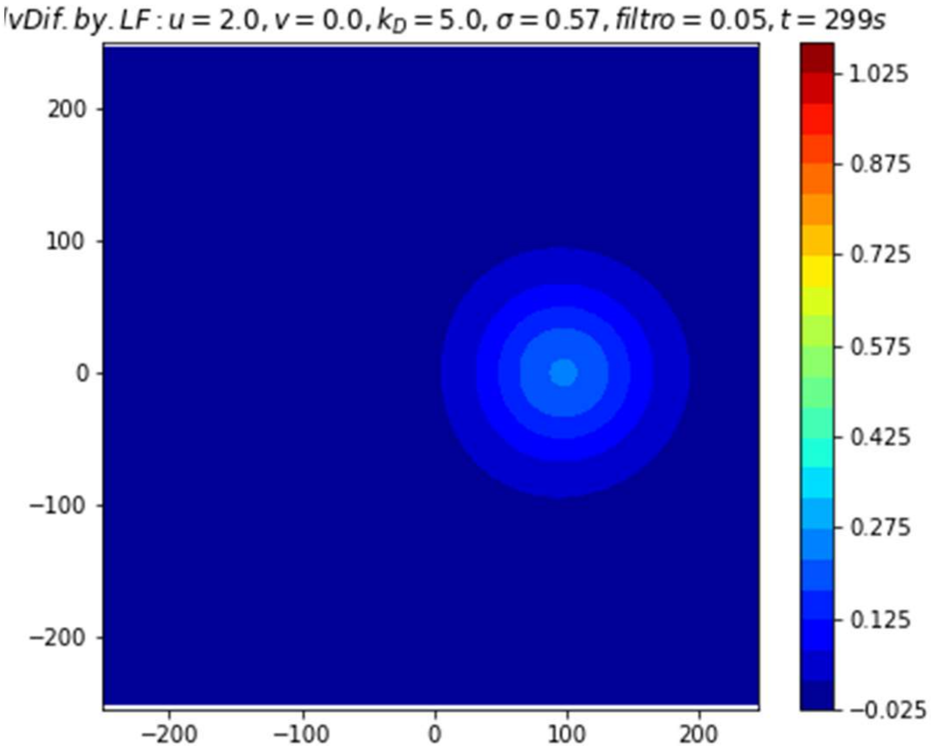


f. by. LF : $u = 0.0, v = 0.0, k_D = 5.0, \sigma = 0.00, filtro = 0.05, t = 10s$

$u = 0$



$lvDif.by.LF : u = 0.0, v = 0.0, k_D = 5.0, \sigma = 0.00, filtro = 0.05, t = 299s$
200
100
0
−100
−200
−200
−100
0
100
200
1.025
0.875
0.725
0.575
0.425
0.275
0.125
−0.025



$lvDif.by.LF : u = 2.0, v = 0.0, k_D = 5.0, \sigma = 0.57, filtro = 0.05, t = 299s$
200
100
0
−100
−200
−200
−100
0
100
200
1.025
0.875
0.725
0.575
0.425
0.275
0.125
−0.025

É possível estabelecer um método absolutamente estável

Vamos recordar o teorema do ponto médio

$$\int_a^b f(x)\,dx = f(\bar{x})(b-a)$$

$$\frac{\partial T}{\partial t} = -u\frac{\partial T}{\partial x} + K_D\frac{\partial^2 T}{\partial x^2}$$

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

$$= -u(1-\alpha)\left(\frac{T_{k+1}^{n} - T_{k-1}^{n}}{2\Delta x}\right) - u\alpha\left(\frac{T_{k+1}^{n+1} - T_{k-1}^{n+1}}{2\Delta x}\right)$$

$$+ K_D\left((1-\alpha)\frac{T_{k-1}^{n} + T_{k+1}^{n} - 2T_k^{n}}{\Delta x^2} + \alpha\frac{T_{k-1}^{n+1} + T_{k+1}^{n+1} - 2T_k^{n+1}}{\Delta x^2}\right)$$

$$\frac{\partial T}{\partial t} = -u\frac{\partial T}{\partial x} + K_D\frac{\partial^2 T}{\partial x^2}$$

$$T_k^{n+1} + \Delta t\, u\alpha\left(\frac{T_{k+1}^{n+1} - T_{k-1}^{n+1}}{2\Delta x}\right) - \Delta t\, K_D\, \alpha\frac{T_{k-1}^{n+1} + T_{k+1}^{n+1} - 2T_k^{n+1}}{\Delta x^2}$$

$$= T_k^n - \Delta t\, u(1-\alpha)\left(\frac{T_{k+1}^{n} - T_{k-1}^{n}}{2\Delta x}\right) + \Delta t\, K_D(1-\alpha)\frac{T_{k-1}^{n} + T_{k+1}^{n} - 2T_k^{n}}{\Delta x^2}$$

$\alpha = 0$ (FTCS), $\alpha = 1$ (Implícito), $\alpha = 0.5$ (semi-implícito, Crank Nicholson)

Na forma matricial, temos um Sistema de equações lineares para calcular a distribuição futura de $T_k^{n+1}(k = 0, \dots, N_x - 1) \equiv \vec{T}^{n+1}$:

$$M\vec{T}^{n+1} = \vec{b}^n$$

$$\frac{\partial T}{\partial t} = -u\frac{\partial T}{\partial x} + K_D\frac{\partial^2 T}{\partial x^2}$$

$$T_k^{n+1} + \Delta t\ u\alpha\left(\frac{T_{k+1}^{n+1} - T_{k-1}^{n+1}}{2\Delta x}\right) - \Delta t\ K_D\ \alpha\,\frac{T_{k-1}^{n+1} + T_{k+1}^{n+1} - 2T_k^{n+1}}{\Delta x^2}$$
$$= T_k^n - \Delta t\ u(1-\alpha)\left(\frac{T_{k+1}^n - T_{k-1}^n}{2\Delta x}\right) + \Delta t\ K_D(1-\alpha)\,\frac{T_{k-1}^n + T_{k+1}^n - 2T_k^n}{\Delta x^2}$$

Ou:

$$\alpha\Delta t\left(-\frac{u}{2\Delta x} - \frac{K_D}{\Delta x^2}\right)T_{k-1}^{n+1} + \left(1 + \frac{2\alpha\Delta t\ K_D}{\Delta x^2}\right)T_k^{n+1} + \alpha\Delta t\left(\frac{u}{2\Delta x} - \frac{K_D}{\Delta x^2}\right)T_{k+1}^{n+1}$$
$$= T_k^n - \Delta t\ u(1-\alpha)\left(\frac{T_{k+1}^n - T_{k-1}^n}{2\Delta x}\right) + \Delta t\ K_D(1-\alpha)\,\frac{T_{k-1}^n + T_{k+1}^n - 2T_k^n}{\Delta x^2}$$

$$M\vec{T}^{n+1} = \vec{b}^{n}$$

$$\alpha\Delta t\left(-\frac{u}{2\Delta x}-\frac{K_D}{\Delta x^2}\right)T_{k-1}^{n+1}+\left(1+\frac{2\alpha\Delta t\,K_D}{\Delta x^2}\right)T_k^{n+1}+\alpha\Delta t\left(\frac{u}{2\Delta x}-\frac{K_D}{\Delta x^2}\right)T_{k+1}^{n+1}$$

$$= T_k^n-\Delta t\,u(1-\alpha)\left(\frac{T_{k+1}^n-T_{k-1}^n}{2\Delta x}\right)+\Delta t\,K_D(1-\alpha)\,\frac{T_{k-1}^n+T_{k+1}^n-2T_k^n}{\Delta x^2}$$

$$\alpha = 0\ (FTCS), \alpha = 1\ (Implícito), \alpha = 0.5\ (Crank - Nicholson)$$

$$M\vec{T}^{n+1} = \vec{b}^{n}$$

$$\begin{bmatrix} \left(1+\frac{2\alpha\lambda\Delta t}{\Delta z^2}\right) & \left(\frac{u\Delta t}{2\Delta \mathrm{x}}-\frac{\alpha\lambda\Delta \mathrm{t}}{\Delta \mathrm{z}^2}\right) & \cdots & \left(-\frac{u\Delta t}{2\Delta \mathrm{x}}-\frac{\alpha\lambda\Delta t}{\Delta z^2}\right) \\ \left(-\frac{u\Delta t}{2\Delta \mathrm{x}}-\frac{\alpha\lambda\Delta t}{\Delta z^2}\right) & \left(1+\frac{2\alpha\lambda\Delta t}{\Delta z^2}\right) & \left(\frac{u\Delta t}{2\Delta \mathrm{x}}-\frac{\alpha\lambda\Delta \mathrm{t}}{\Delta \mathrm{z}^2}\right) & \\ \vdots & & \ddots & \vdots \\ & & \cdots & \\ \left(\frac{u\Delta t}{2\Delta \mathrm{x}}-\frac{\alpha\lambda\Delta \mathrm{t}}{\Delta \mathrm{z}^2}\right) & & \left(-\frac{u\Delta t}{2\Delta \mathrm{x}}-\frac{\alpha\lambda\Delta t}{\Delta z^2}\right) & \left(1+\frac{2\alpha\lambda\Delta t}{\Delta z^2}\right) \end{bmatrix} \begin{bmatrix} T_0^{n+1} \\ T_1^{n+1} \\ \\ \\ \\ T_{N_z-1}^{n+1} \end{bmatrix} = \vec{b}$$

A matriz M é esparsa (só não são ‘0’ a diagonal, subdiagonais e os cantos).

É uma matriz de Toeplitz (os termos de cada diagonal são todos iguais)

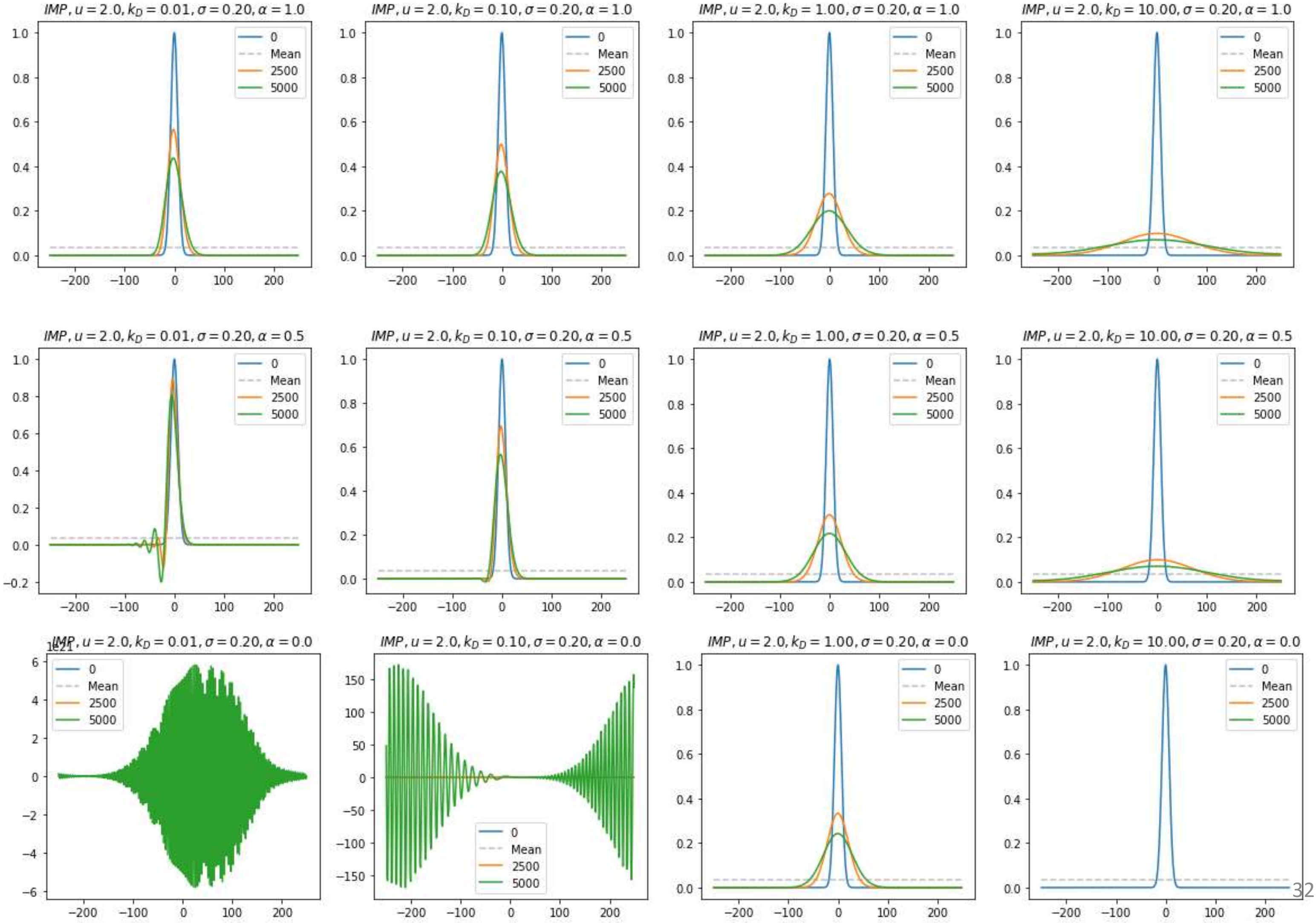
Os cantos impõem uma condição fronteira cíclica.

$k_D \rightarrow$

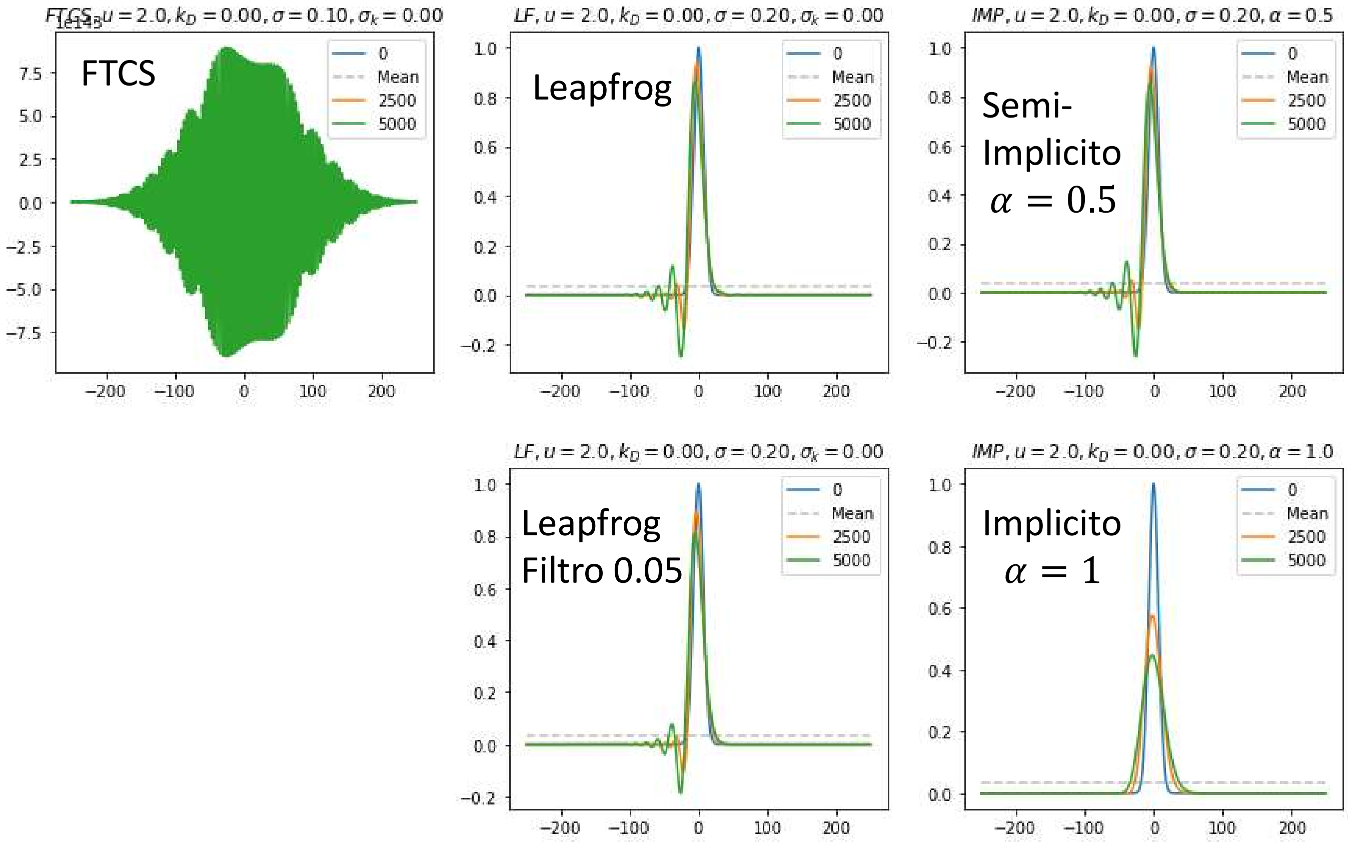
$\alpha = 1$
Implicito

$\alpha = 0.5$

$\alpha = 0$
FTCS



$k_D = 0$



FTCS

Leapfrog

Semi-Implicito $\alpha = 0.5$

Leapfrog Filtro 0.05

Implicito $\alpha = 1$

Método implícito

…

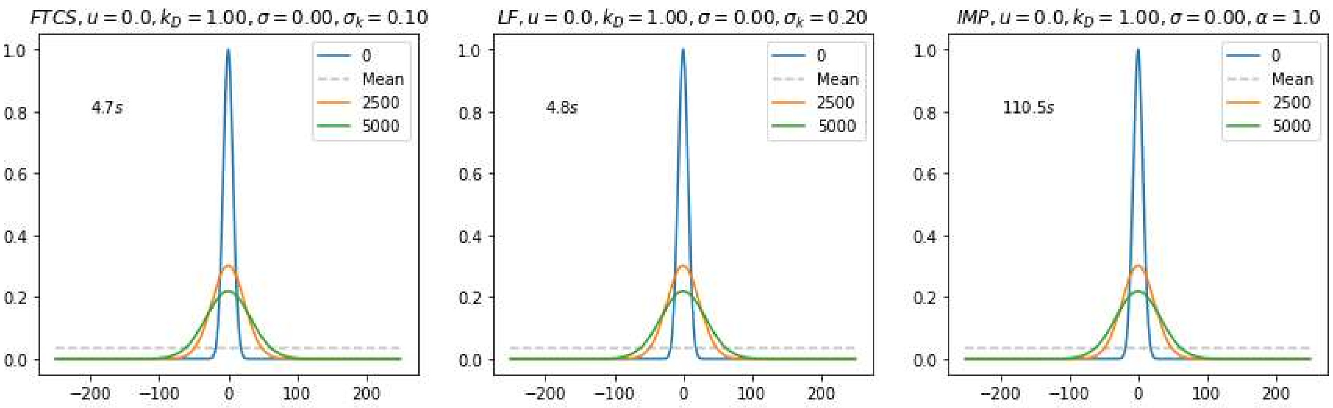
```
Wx=10;x0=0;TI=np.exp(-((x-x0)/Wx)**2); B=np.zeros(TI.shape)
T=np.copy(TI);TP=np.copy(T)
start_time=time.process_time()
M=np.zeros((nx,nx))
left=alpha*dt*(-u/(2*dx)-kD/dx**2);    right=alpha*dt*(u/(2*dx)-kD/dx**2)
center=(1+2*alpha*dt*kD/dx**2)
for ix in range(nx):
    M[ix,ix]=center
for ix in range(nx-1):
    M[ix,ix+1]=right
for ix in range(0,nx-1):
    M[ix+1,ix]=left
M[0,nx-1]=left;    M[nx-1,0]=right
```

…

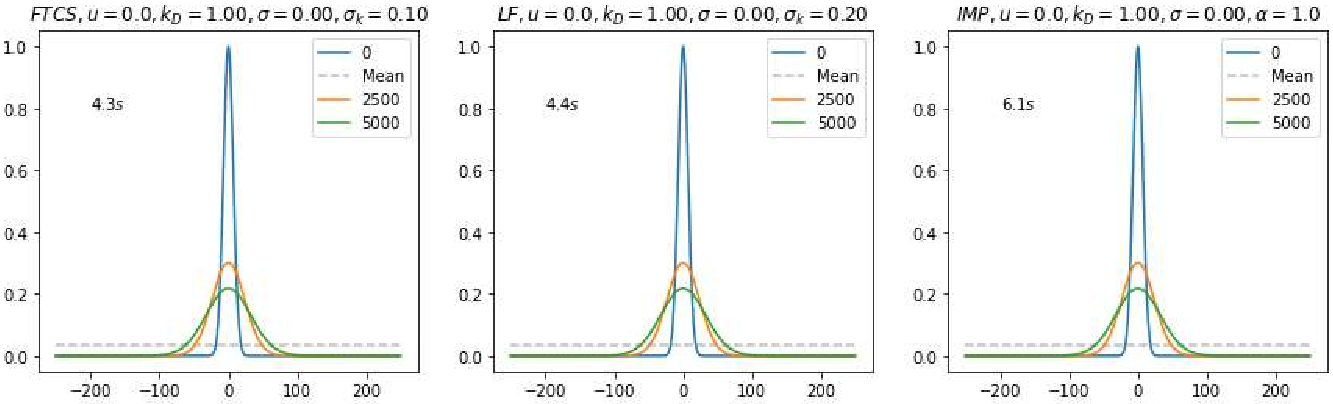
```
for it in range(1,nt):
    for ix in range(nx):
        ixm=ix-1;ixp=ix+1
        if ixm<0:
            ixm=nx-1
        elif ixp>nx-1:
            ixp=0
        B[ix]=T[ix]+dt*(1-alpha)*(-u*(T[ixp]-T[ixm])/(2*dx)+kD/dx**2*(T[ixm]+T[ixp]-2*T[ix]))
     TP=np.linalg.solve(M,B)
timespent=time.process_time()-start_time
```

$u = 0, k_D = 1$

```
M=np.zeros((nx,nx))
…
TP=np.linalg.solve(M,B)
```



```
from scipy.sparse import lil_matrix
from scipy.sparse.linalg import spsolve
…
M=lil_matrix((nx,nx))
…
M=M.tocsr()
…
TP=spsolve(M,B)
```



```
…   from scipy.sparse import lil_matrix; from scipy.sparse.linalg import spsolve
    Wx=10;x0=0;TI=np.exp(-((x-x0)/Wx)**2); B=np.zeros(TI.shape)
    T=np.copy(TI);TP=np.copy(T)
    start_time=time.process_time()
    M=lil_matrix((nx,nx)) #M=np.zeros((nx,nx))
    left=alpha*dt*(-u/(2*dx)-kD/dx**2);    right=alpha*dt*(u/(2*dx)-kD/dx**2)
    center=(1+2*alpha*dt*kD/dx**2)
    for ix in range(nx):
        M[ix,ix]=center
    for ix in range(nx-1):
        M[ix,ix+1]=right
    for ix in range(0,nx-1):
        M[ix+1,ix]=left
    M[0,nx-1]=left;    M[nx-1,0]=right
    M=M.toscr(M)
…
    for it in range(1,nt):
        for ix in range(nx):
            ixm=ix-1;ixp=ix+1
            if ixm<0:
                ixm=nx-1
            elif ixp>nx-1:
                ixp=0
            B[ix]=T[ix]+dt*(1-alpha)*(-u*(T[ixp]-T[ixm])/(2*dx)+kD/dx**2*(T[ixm]+T[ixp]-2*T[ix]))
         TP=spsolve(M,B) #np.linalg.solve(M,B)
    timespent=time.process_time()-start_time
```

Método implícito *sparse*

O método implícito

É absolutamente estável ($\alpha = 1$)

Obriga à solução de um Sistema de N_x equações, no caso 1D.

No caso multidimensional isso pode ser proibitivo.

Sendo a matriz dos coeficientes esparsa, existem métodos para reduzir drasticamente o custo da solução.