Advecção com difusão e forçante para uma fonte do poluente com aplicação do operador *splitting*

Alejandro H. D. Peralta*

Instituto de Astronomia, Geofísica e Ciências Atmosféricas da Universidade de São Paulo

16 de outubro de 2022

Resumo

Neste trabalho apresentamos a solução numérica com o esquema leapfrog da equação de advecção com difusão e forçante como uma fonte pontual. Além de resolver numericamente, a aplicação do operador *splitting* à forçante é uma técnica com solução estável e que a emissão tem um comportamente independente ao campo de advecção e difusão. Excluindo a forçante, o esquema numérico pode ser avaliado com a condição de estabilidade de Von Neumann para as condições do exercício como também com as variações dos parâmetros como velocidade do vento e o fator de amplitude "K" do termo de difusão. Em conclusão, a difusão e a aplicação do método splitting à forçante permitem melhorar a estabilidade numérica da solução numérica com o método leapfrog em comparação dos resultados numéricos da equação de advecção e forçante do Ex. 2.

1. Introdução

Os modelos de transporte atmosférico resolvem os processos físicos (advecção, difusão, nuvens, sedimentação seca) e químicos (reações de gas para aerossóis, emissões e sumidouros) ao longo do tempo como mostra a eq. 1, onde c_i é a concentração da espécie e R_{gi} e E_i são a produção neta da reação da fase gasosa e a emissão, respetivamente (Seinfeld e Pandis, 2016). Na realidade, estes processos acontecem simultaneamente. No entanto, tentar resolver todo com uma só equação discretizada numericamente pode demandar muitos recursos computacionais e cada processo pode precisar de um esquema numérico diferente para obter uma solução numérica. Por tanto, temos um problema de acoplamento que precisa de alguma técnica para obter a solução da eq. 1. Para isso, a técnica de *splitting* ou também chamado *timestep splitting* resolve individualmente os processos de maneira estável para ser adicionados depois como um resultados final (Caya et al., 1998). Esta técnica é a mais usada na maioria de modelos atmosféricos de transporte químico (Seinfeld e Pandis, 2016).

$$\frac{\partial c_i}{\partial t} = \left(\frac{\partial c_i}{\partial t}\right)_{adv} + \left(\frac{\partial c_i}{\partial t}\right)_{diff} + \left(\frac{\partial c_i}{\partial t}\right)_{nuv} + \left(\frac{\partial c_i}{\partial t}\right)_{gas} + \left(\frac{\partial c_i}{\partial t}\right)_{aer} + R_{gi} + E_i \tag{1}$$

Conforme com Seinfeld e Pandis (2016), a concentração muda cada Δt como segue $\Delta c = c(t + \Delta t) - c(t)$. Se nós desacoplamos cada processo temos $\Delta c^A = A(\Delta t)c(t)$, $\Delta c^D = D(\Delta t)c(t)$, $\Delta c^{Nuv} = Nuv(\Delta t)c(t)$, $\Delta c^G = G(\Delta t)c(t)$, $\Delta c^{Aer} = Aer(\Delta t)c(t)$, $\Delta c^R = R(\Delta t)c(t)$, $\Delta c^E = E(\Delta t)c(t)$; então, eles podem ser adicionados em paralelo $\Delta c = \Delta c^A + \Delta c^D + \Delta c^{Nuv} + \Delta c^G + \Delta c^{Aer} + \Delta c^R + \Delta c^E$ para obter a nova concentração $c(t+\Delta t) = c(t) + \Delta c$, ou em série como mostra a eq. 2 (Seinfeld e Pandis, 2016). Nem todos os processos podem variar a concentração no mesmo Δt pelo que pode gerar erros de representação temporal segundo o processo.

*Estudante de doutorado, email aperalta@usp.br

$$
\begin{aligned}
c^1(t+\Delta t) &= A(\Delta t)c(t) \\
c^2(t+\Delta t) &= D(\Delta t)c^1(t+\Delta t) \\
c^3(t+\Delta t) &= Nuv(\Delta t)c^2(t+\Delta t) \\
c^4(t+\Delta t) &= Aer(\Delta t)c^3(t+\Delta t) \\
c^5(t+\Delta t) &= R(\Delta t)c^4(t+\Delta t) \\
c(t+\Delta t) &= E(\Delta t)c^5(t+\Delta t)
\end{aligned} \tag{2}
$$

Neste trabalho aplicamos o operador *splitting* à forçante F que é uma fonte senoidal que varia no tempo ($n\Delta t$) e que é parte da equação de advecção e difusão nas mesmas condições do exercício 2. Conforme com Döös et al. (2020), a difusão pode ser discretizada como a segunda ordem da derivada (eq. 3), o qual melhora a estabilidade do esquema numérico leapfrog aplicado neste trabalho se comparar com o Ex. 2. Finalmente, os resultados foram verificados experimentalmente com o critério discutido na Fig. 8.6 de Döös et al. (2020) para a estabilidade numérica do esquema, considerando variações do vento (U) e o termo da difusão (K ou kappa) para F = 0. Os resultados com e sem *splitting* também foram comparados e mostram diferenças pequenas na solução numérica quando varia o tempo (n+1) no caso da forçante.

$$
\left(\frac{d^2u}{dx^2}\right)_j \approx \left[\frac{d}{dx}\left(\frac{du}{dx}\right)\right]_j \approx \frac{\frac{u_{j+1}-u_j}{\Delta x} - \frac{u_j-u_{j-1}}{\Delta x}}{\Delta x} = \frac{u_{j+1}-2u_j+u_{j-1}}{(\Delta x)^2} \tag{3}
$$

2. Descrição da metodologia

A aproximação considerou as condições do exercício 2 com a adição do efeito da difusão. A equação que governa este problema é dada pela eq. 4, onde F é a mesma fonte periódica do Ex. 2, localizado na metade da grade com uma resolução horizontal $\Delta x = 2500$ metros e temporal $\Delta t = 100$ segundos.

$$
\frac{\partial C}{\partial t} + U\frac{\partial C}{\partial x} = K\frac{\partial^2 C}{\partial^2 x} + F \tag{4}
$$

O requerimento do exercício 3 é determinar o fator K de forma que o tempo de decaimento seja da ordem de 3 horas. Inicialmente F está no tempo n, devido que o esquema leapfrog precisa iniciar com o esquema Euler *forward-backward*. Logo a discretização segue o esquema leapfrog e é radiacional nas condições de fronteira. A eq. 4 foi discretizada para o esquema leapfrog (eq. 5), considerando a advecção e a difusão no tempo n-1 com a forçante no tempo n-1,

$$
C_j^{n+1} = C_j^{n-1} - 2\Delta t\, U\frac{C_{j+1}^n - C_{j-1}^n}{2\Delta x} + 2\Delta t\, K\frac{C_{j+1}^{n-1} - 2C_j^{n-1} + C_{j-1}^{n-1}}{(\Delta x)^2} + 2\Delta t F_j^{n-1} \tag{5}
$$

ou também expressado como,

$$
C_j^{n+1} = C_j^{n-1} - \mu(C_{j+1}^n - C_{j-1}^n) + 2\nu(C_{j+1}^{n-1} - 2C_j^{n-1} + C_{j-1}^{n-1}) + 2\Delta t F_j^{n-1},
$$

onde $\mu = \frac{U\,\Delta t}{\Delta x}$ como número de Courant e $\nu \approxeq K\,\Delta t/(\Delta x)^2$ número de difusão. A solução da discretização é definida como cenário "Base". A equação de difusão linear de segunda ordem tem um tempo de decaimento expressado como $te = \frac{1}{K\,k^2}$, onde k é o número de onda ($k = \frac{\omega}{U}$). Depois a forçante é introduzido com o método *splitting*, o passo 1 está expressado como

$$
\frac{C_j^* - C_j^{n-1}}{2\Delta t} + U\frac{C_{j+1}^n - C_{j-1}^n}{2\Delta x} - K\frac{C_{j+1}^{n-1} - 2C_j^{n-1} + C_{j-1}^{n-1}}{(\Delta x)^2} = 0,
$$

no passo 2 a forçante é inserida como

$$
\frac{C_j^{n+1} - C_j^*}{2\Delta t} = \frac{F_j^{n-1} + F_j^{n-2}}{2}.
$$

A verificação do critério de estabilidade discutido em Döös et al. (2020) considera avaliar λ^2 em função de ν e μ com $F = 0$, ou seja a equação é de advecção e difusão. A analise de estabilidade com o esquema leapfrog é expressada com a equação

$$\lambda^2 + 2ia\lambda + 8b - 1 = 0, \tag{6}$$

onde $a \approxeq \mu \sin(k\Delta x)$ e $b \approxeq \nu \sin^2(\frac{k\Delta x}{2})$. As duas soluções da equação são $\lambda = -ia \pm \sqrt{-a^2 + 1 - 8b}$. Se $1 - 8b - a^2 > 0$, logo $|\lambda|^2 = a^2 + 1 - 8b - a^2 = 1 - 8b < 1$ apresenta um esquema estável. Porém, se $1 - 8b - a^2 < 0$, logo $\lambda = -i(a \mp \sqrt{a^2 + 8b - 1})$, temos $\lambda^2 = -(2a^2 + 8b - 1 \mp 2a\sqrt{a^2 + 8b - 1})$; as duas ultimas soluções foram consideradas para achar a relação com ν e μ. A programação dos experimentos em código de Python é mostrado no Apêndice A, onde mostra-se a solução numérica, a mesma com splitting e a avaliação da estabilidade com F = 0.

3. Resultados

A Fig. 1 mostra a solução numérica com o esquema leapfrog sem *splitting* chamado "Base". Os resultados mostram diferenças entre as duas soluções (com e sem *splitting*), principalmente o quando a fonte é acoplada em tempos diferentes que o cenário Base (F^{n-1}) como é mostrado na Fig. 2 quando inserir a fonte (F) no tempo n+1. A verificação da estabilidade de von Neumann considera a comparação de λ^2 com os valores dos parâmetros μ e ν. Achar a solução da eq. 6 somente acontece quando F = 0, então foram considerados as duas soluções seguintes apresentados e discutidos em Döös et al. (2020):

$$|\lambda|^2 = 1 - 8b < 1$$
$$\lambda^2 = -(2a^2 + 8b - 1 + 2a\sqrt{a^2 + 8b - 1})$$

Este trabalho mostra o resultado de λ^2 para as condições do exercício (U=10 m/s, $\omega = 2\pi/1800$ e te = 3 h) e também com as variações da velocidade do vento, omega e tempo de decaimento. A Fig. 3 (lado esquerdo) mostra o ponto preto que corresponde à estabilidade para as condições do exercício e, em cores similares como mostra a Fig. 8.6 de Döös et al. (2020), os resultados de λ^2 baseado nas variações dea velocidade do vento, omega e tempo de decaimento.

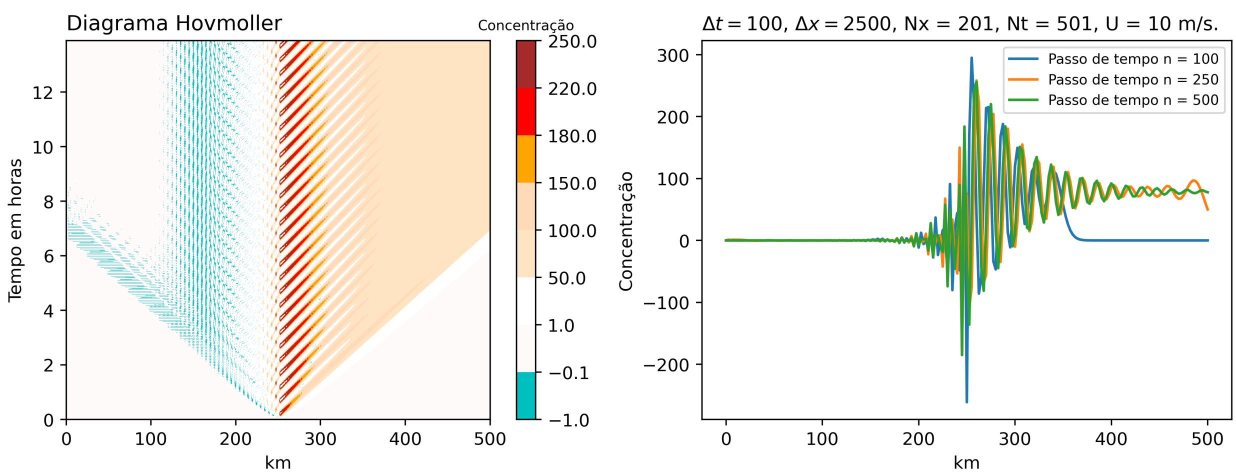

Figura 1: Solução numérica com o esquema leapfrog na advecção e a difusão no tempo n-1, forçante no tempo n-1, chamado cenário 'Base'.

4. Discussão dos resultados

A solução numérica da advecção e a difusão no tempo n-1 e a forcante no tempo n-1 com o esquema Leapfrog (Fig. 1) mostra que o termo da difusão melhora a estabilidade, reduzindo o modo computacional gerado quando somente acontece processos de advecção e a emissão (fonte pontual e senoidal com variação temporal). Por tanto, o efeito da

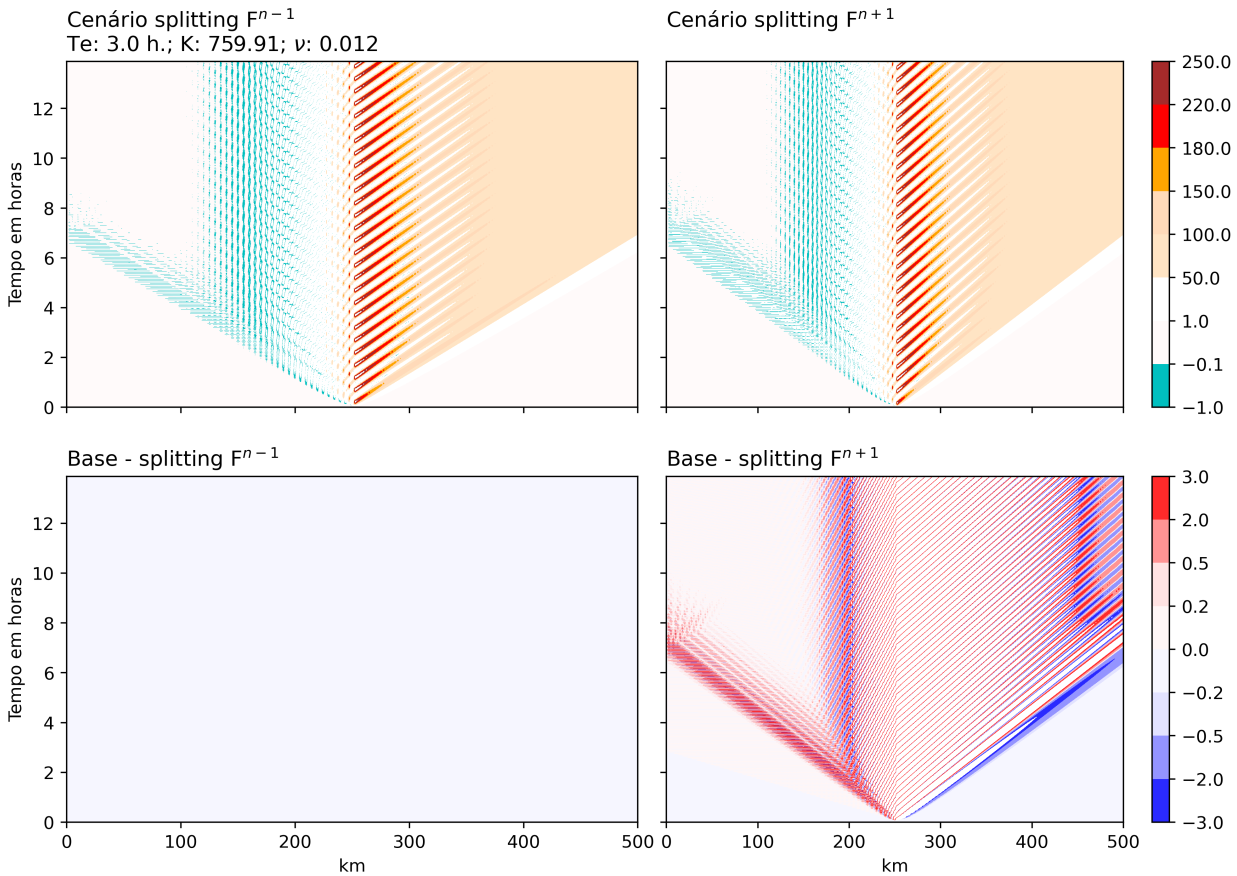

Figura 2: Cenários de soluções numéricas com a aplicação do operador *splitting* para a fonte inserida nos tempos F^{n-1} e F^{n+1}.

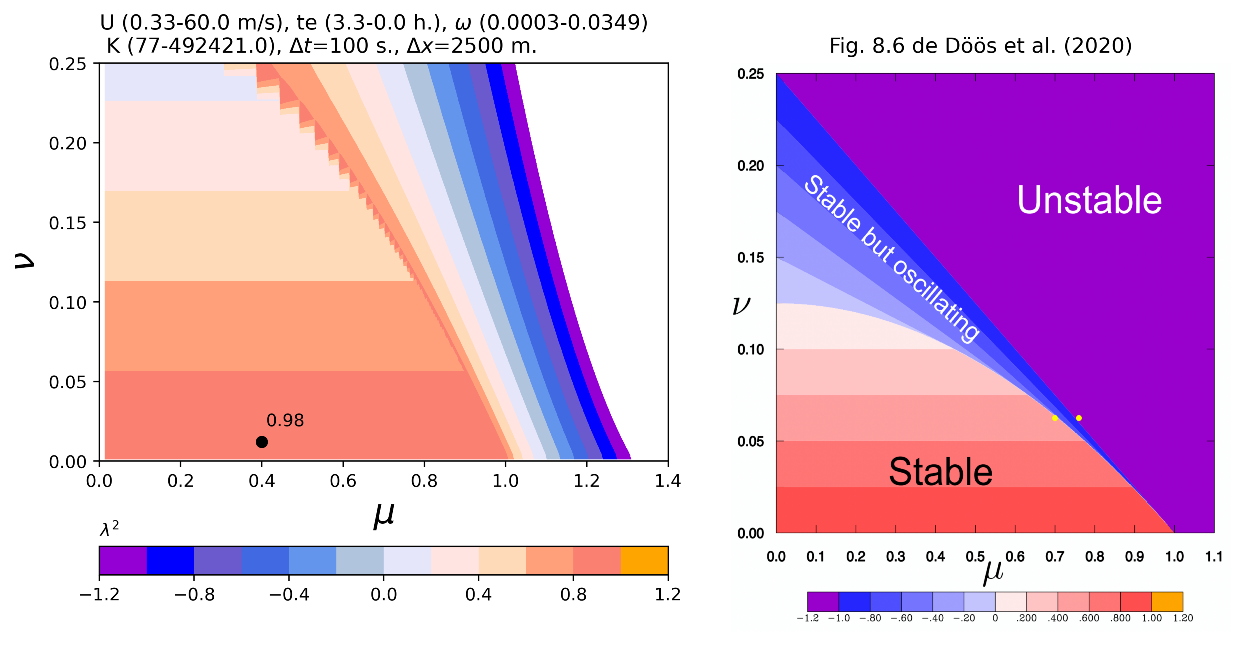

Figura 3: Verificação experimental com o critério discutido em Döös et al. (2020) (Figura 8.6), com variações de U e K.

difusão tem um comportamento similar à aplicação de filtros no espaço e tempo como aconteceu no Ex. 2. No entanto, a magnitude da fonte e a propagação é atenuado devido à difusão, algo similar como aconteceu com o esquema de ordem 1 Euler progressivo no tempo e regressivo no espaço aplicado às condições do Ex. 2.

A forçante (F) inserida com o método *splitting* no tempo F^{n-1} mostra resultados iguais na discretização com o resultado Base. Logo, a fonte inserida para o cenário F^{n+1} mostra diferenças negativas marcantes ao início da propagação comparado com o resultado Base, mostrado como Base - splitting F^{n+1} na Fig. 2. Por tanto, a forçante como processo de emissão presenta um comportamento independente do campo de advecção e difusão.

A avaliação da estabilidade de advecção-difusão com o esquema leapfrog, sem considerar a fonte, é estável (λ^2 = 0,98) para um μ de 0,4 e ν equal a 0,012. Nas condições de configuração do modelo leapfrog mostrado na Fig. 8.6 de Döös et al. (2020) o valor λ^2 estaria na faixa de 0,8 e 1, sendo estável (Fig. 3). Outras avaliações de estabilidade consideraram a variação de 1000 pontos para parâmetros como U (0,33 - 60 m/s), te (3,3 - 0.001 h), ω (0,0003 - 0,0349) e mantendo fixos $\Delta t = 100\,s$ e $\Delta x = 2500$ m, mostrados na Fig. 3 (esquerda). Podemos notar que temos instabilidades para valores de cor roxo, por exemplo quando μ > 1,1 e ν > 0,1; mas para μ < 1 podemos ter soluções numéricas estáveis com ν maiores até 0,20. O parâmetro K mostra uma variação de aumento devido à variação de decrescimento do tempo de decaimento. O aumento da velocidade do vento acrescenta a instabilidade numérica a partir de valores acima de 30 m/s. Em conclusão, o termo de difusão e a aplicação do método splitting à forçante permite melhorar a estabilidade numérica da solução numérica com o método leapfrog.

Bibliografia

Caya, A., Laprise, R., Zwack, P. (1998). Consequences of Using the Splitting Method for Implementing Physical Forcings in a Semi-Implicit Semi-Lagrangian Model. Monthly Weather Review 126, 1707–1713. https://doi.org/10.1175/1520-0493(1998)126%3C1707:COUTSM%3E2.0.CO;2

Döös, K., Lundberg, P., Campino, A.A. (2020). Basic Numerical Methods in Meteorology and Oceanography, 1.ª ed. Department of Meteorology, Stockholm University, Stockholm.

Seinfeld, J.H., Pandis, S.N. (2016). Atmospheric Chemistry and Physics: From Air Pollution to Climate Change, 3.ª ed. John Wiley & Sons, Inc.

Apêndice A

O código escrito em Python começou com resolver todo sem o operador *splitting* como segue

```python
def wave(t, n):
    om = 2*np.pi/1800
    res = np.sin(om*t[n])
    fonte = lambda res: res if res >= 0 else 0

    return fonte(res)

# Condições do exercício
# ----------------------
Nx, dx, Nt, dt, U  = 201, 2500, 501, 100, 10,
x, _ = np.linspace(0, (Nx-1)*dx, Nx, retstep = True)
t, _ = np.linspace(0, (Nt-1)*dt, Nt, retstep = True)
CFL     = U*dt/dx
C = np.zeros((Nx, Nt))
F = C.copy()
w = int(dx/(dt*U)) # w*dt*U/dx = 1; w*dt = dx
T, X = np.meshgrid(t, x)
om = 2*np.pi/1800
te = 3*3600 # seconds decaimento
K = 1/(te*(om/U)**2) # kappa
v = K*dt/(dx)**2  # nu
```

```
# Método leapfrog
for n in range(1, Nt-1):
    # Euler forward
    if n == 1:
        # Inicialmente F no tempo n, Euler forward-backward
        F[101, n] = 1/2*(wave(t, n) + wave(t, n-1))
        for j in range(1, Nx-1):
            C[j,n+1] = C[j,n] - CFL*(C[j,n]-C[j-1,n]) +
                       v*(C[j+1,n]-2*C[j,n]+C[j-1,n]) +
                       dt*(F[j,n])
    else: # Fonte e difusão como n-1
        F[101, n-1] = 1/2*(wave(t, n-1) + wave(t, n-2))
        for j in range(1, Nx-1):
            C[j,n+1] = C[j,n-1] - CFL*(C[j+1,n]-C[j-1,n]) +
                        2*v*(C[j+1,n-1]-2*C[j,n-1]+C[j-1,n-1]) + 2*dt*(F[j,n-1])

    # Radiacional
    C[-1, n+1] = C[-1, n] - CFL*(C[-1, n]- C[-2, n])
```

O segundo passo foi calcular com o operador *splitting* como segue:

```
C = np.zeros((Nx, Nt))
Cs = C.copy()
F = C.copy()
for n in range(1, Nt-1):
    # Euler forward
    if n == 1:
        # Inicialmente F no tempo n, Euler forward-backward
        F[101, n] = 1/2*(wave(t, n) + wave(t, n-1))
        for j in range(1, Nx-1):
            # Passo 1
            # -------
            Cs[j,n+1] = C[j,n] - CFL*(C[j,n]-C[j-1,n]) +
                        v*(C[j+1,n]-2*C[j,n]+C[j-1,n])
            # Passo 2
            # -------
            C[j,n+1] = Cs[j,n+1] + dt*(F[j,n])

    # Leapfrog
    else:
        if fonte=='n-1':
            F[101, n-1] = 1/2*(wave(t, n-1) + wave(t, n-2))
        elif fonte=='n+1':
            F[101, n+1] = 1/2*(wave(t, n+1) + wave(t, n))
        for j in range(1, Nx-1):
            # Passo 1
            # -------
            Cs[j,n+1] = C[j,n-1] - CFL*(C[j+1,n]-C[j-1,n]) +
                        2*v*(C[j+1,n-1]-2*C[j,n-1]+C[j-1,n-1])
            # Passo 2
            # -------
            if fonte =='n-1':
                C[j,n+1] = Cs[j,n+1] + 2*dt*F[j,n-1]
            elif fonte =='n+1':
```

```
                C[j,n+1] = Cs[j,n+1] + 2*dt*F[j,n+1]

    # Radiacional
    C[-1, n+1] = C[-1, n] - CFL*(C[-1, n]- C[-2, n])
```

Para o analises de estabilidade considerando o critério na Fig. 8.6 em Döös et al. (2020), primeiro considerou-se variar a velocidade do vento, omega e o tempo de decaimento, como segue

```
# Resultado do Exercício 3:
mu = U*dt/dx
k = om/U # numero de onda
K = 1/(te*k**2) # kappa
v = K*dt/(dx)**2  # nu
a = mu * np.sin(k*dx)
b = v * np.sin(k*dx/2)**2
lb = lambda a, b: 1 - 8*b if 1 - 8*b - a**2 > 0 else
                   -(2*a** + 8*b - 1 + 2*a*np.sqrt(a**2+8*b-1))
lb_ex3 = lb(a, b)

# Avaliação com o critério de estabilidade para gerar a figura
# ------------------------------------------------------------
# Variação da U, mantendo
Us = np.linspace(U/30, U*6, 1000)
oms = np.linspace(om/10, om*10, 1000)
mus = Us*dt/dx # CFLs
ks = oms/Us # números de onda

# Variação do K (kappa) podendo varia te
tes = np.linspace(te*1.1, te/1800, 1000)
Ks = 1/(tes*ks**2) # kappas
vs = Ks*dt/(dx)**2  # nu

MUS, VS = np.meshgrid(mus, vs)
lamb = np.zeros(MUS.shape)*np.nan

for i in range(len(vs)):
    for j in range(len(mus)):
        a = mus[j] * np.sin(ks[j]*dx)
        b = vs[i] * (np.sin(ks[i]*dx/2))**2
        key = 1 - 8*b - a**2
        if key > 0:
            lamb[i,j] = 1 - 8*b
        elif key < 0:
            lamb[i,j] = -(2*a** + 8*b - 1 + 2*a*np.sqrt(a**2+8*b-1))

fig, (ax, ax2) = plt.subplots(1,2, figsize=(12,6), gridspec_kw={'wspace':.05})
im = ax.contourf(MUS, VS, lamb, levels= np.arange(-1.2,1.4,.2), colors=colores)
cbar = fig.colorbar(im, ax=ax,orientation="horizontal")
cbar.ax.set_title('$\lambda ^2$', fontsize=10, loc='left')
ax.set_xlabel("$\mu$", fontsize=20, fontweight='bold')
ax.set_ylabel("$\\nu$", fontsize=20, fontweight='bold')
ax.set_xlim(0, 1.1)
ax.set_ylim(0, 0.25)
ax.scatter(mu, v, c="k")
ax.text(mu+.01, v+.01, str(round(lb_ex3,2)))
```

```
ax.set_title(f"U ({round(Us[0],2)}-{round(Us[-1],2)} m/s), \
             te ({round(tes[0]/3600,2)}-{round(tes[-1]/3600,2)} h.), \
             $\omega$ ({round(oms[0],4)}-{round(oms[-1],4)})\
             \n K ({round(Ks[0])}-{round(Ks[-1],0)}), \
             $\Delta t$={dt} s., $\Delta x$={dx} m.", loc='left')
im2 = plt.imread('fig/fig8_6_doos.png')
ax2.imshow(im2, extent=(-0.1, 1, 0.2, 1.45))
ax2.set_title("Fig. 8.6 de Döös et al. (2020)")
ax2.axis('off')
fig.savefig("fig/estabilidade.png", dpi = 400, bbox_inches='tight', facecolor='w')
```