Transporte pela advecção de uma fonte de pulso senoidal e aplicação de filtros para a remoção do modo computacional

Alejandro H. D. Peralta*

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

2 de outubro de 2022

Resumo

A emissão de um poluente pode variar ao longo do tempo, como no caso de uma chaminé que emite o pulso senoidal no campo básico com velocidade do vento constante. Este trabalho mostra os cálculos para resultados analíticos e numéricos (Euler progressivo-regressivo, leapfrog (2ª e 4ª ordem) e implícito como o esquema Crank-Nicolson). Alguns métodos numéricos geraram oscilações do modo computacional pelo que foram filtrados. Outros experimentos foram considerados para o método implícito com a variação da resolução do tempo Δt para obter diferentes números de Courant (CFL) para valores de 1, 2 e 4. A aproximação da ordem 1 é um importante esquema que não precisa de filtros devido à simplicidade do método. No entanto, o esquema é difuso pelo que as concentrações são subestimadas se comparar com a solução analítica. Outros esquemas como leapfrog e Crank-Nicolson geram resultados com oscilações que contradizem o fenômeno físico, pelo que a aplicação de filtros é importante para preservar a monotonicidade. Os resultados dos experimentos são importantes a fim de representar a realidade do fenômeno do transporte dos poluentes na atmosfera, como no caso dos modelos de qualidade do ar.

1. Introdução

As fontes de emissão podem variar ao longo do tempo como acontecem em diferentes atividades humanas como industriais, residencial e transporte rodoviário. O exercício 2 considera analisar uma fonte de poluição pontual (p.e., chaminé) que emite em forma de pulso senoidal com o mesmo campo básico de velocidade do vento (U = 10 m/s) do exercício 1 (equação de advecção linearizada). Os cálculos consideram resultados da solução analítica e também dos métodos numéricos (Euler progressivo-regressivo, leapfrog 2ª ordem, leapfrog 4ª ordem no espaço e implícito como o esquema Crank-Nicolson). As condições iniciais do campo são nulas e somente a fonte pontual senoidal gera emissões na metade do domínio 1D. Alguns métodos numéricos geraram oscilações do modo computacional que foram filtradas. Outros experimentos foram considerados para o método implícito com a variação da resolução do tempo Δt para obter diferentes números de Courant (CFL) para valores de 1, 2 e 4. Os resultados dos experimentos são importantes para conhecer as aplicações dos filtros nos modelos eulerianos de qualidade do ar considerando que as unidades das concentrações dos poluentes sempre são positivas, pelo que é importante preservar a monotonicidade física do fenômeno.

$$\frac{dC}{dt} = F(C,t) \tag{1}$$

$$C(t_{n+1}) = C(t_n) + \int_{t_n}^{t_{n+1}} F(C,t)dt \tag{2}$$

A aplicação de equações diferenciais ordinárias (ODEs, siglas em inglês) que describe a evolução do poluente no tempo é mostrado na eq. 1 que pode ser resolvido com a interpretação geométrica como mostra a eq. 2, (Brasseur e

*Estudante de doutorado, email aperalta@usp.br

Jacob, 2017). Além disso, o desenvolvimento do exercício 2 precisa das equações do movimento da onda ao longo do tempo e espaço causado pela advecção desde a fonte. A forma geral para ondas transversais de propagação considera a $y = f(x - vt)$ onde v é a velocidade. As ondas harmônicas tem a forma de $\sin(kx - \omega t)$ ou $\cos(kx - \omega t)$ onde $\omega = 2\pi/T$ para T como período e o número de onda $k = \frac{2\pi}{\lambda}$ para λ como comprimento de onda em metros.

1.1 Filtros Robert-Asselin

Os filtros de Robert-Asselin podem remover oscilações do modo computacional para os esquemas como o método leapfrog (Doos et al., 2020). Neste trabalho foram considerados os filtros no espaço (eq. 3) e tempo (eq. 4).

$$C_j = C_j + \gamma(C_{j-1} - 2C_j + C_{j+1}) \tag{3}$$

$$C^n = C^n + \gamma(C^{n-1} - 2C^n + C^{n+1}) \tag{4}$$

2. Descrição da metodologia

O exercício 2 considera que a fonte tem um pulso senoidal de emissão na metade do campo de advecção, governado pela equação $\frac{\partial C}{\partial t} + U\frac{\partial C}{\partial x} = F$ onde $F = F(x,t)$, considerando $F(i = 100, n\Delta t) = sin(\omega.n\Delta t)$ para n = 0, ... N_{max} se $sin(\omega.n.\Delta t) > 0$ e $F(i = 100, n\Delta t) = 0$ caso $sin(\omega.n\Delta t) < 0$ como $\omega = \frac{2\pi}{1800}\ s^{-1}$. A equação com derivadas parciais foi transformada numa equação diferencial ordinária com solução geral (Brasseur e Jacob, 2017), com isso temos

$$\frac{dx}{dt} = U$$

,

$$\frac{dt}{ds} = 1 \tag{5}$$

$$\frac{dC}{ds} = F(x = 100, t). \tag{6}$$

Podemos encontrar a relação entre equações 5, 6 como segue

$$dC = F(x = 100, t)dt$$

$$\int dC = \int_{t_n}^{t_n+1} F(x = 100, t)\, dt$$

$$C^{n+1} = C^n - \frac{1}{\omega}[\cos(\omega\, t_{n+1}) - \cos(\omega\, t_n)]. \tag{7}$$

Esta solução analítica pode ser propagada no espaço como $f(x - U\, t)$; para nosso caso a propagação acontece desde a metade da grade. Este trabalho considerou representar a solução analítica com uma função de onda harmônica,

$$C_x^{n+1} = C_x^n - \frac{1}{\omega}[\cos(-x\, k + \omega\, t_{n+1}) - \cos(-x\, k + \omega\, t_n)]$$

com isso, os resultados negativos foram removidos e substituídos por zero para preservar a monotonicidade do fenômeno físico. A seção Apêndice A mostra um resumo das principais partes do código escrito em Python para resolver os diferentes esquemas numéricos. A discretização da fonte $F(x = 100, t)$ segue uma aproximação tipo *Euler forward* a partir da equação diferencial ordinária (Brasseur e Jacob, 2017),

$$\frac{dC}{dt} = F(C,t)$$

$$\frac{C^{n+1} - C^n}{\Delta t} = F(C^n, t_n)$$

onde $\Delta t = t_{n+1} - t_n$ o passo de tempo no método numérico, então nós temos

$$C^{n+1} = C^n + \Delta t\, F^n.$$

Também temos como outra opção o esquema implícito trapezoidal como segue

$$C^{n+1} = C^n + \frac{\Delta t}{2}(F^{n+1} + F^n).$$

No esquema de segundo ordem como leapfrog, a discretização é a seguinte

$$C^{n+1} = C^{n-1} + 2\,\Delta t\, F^n.$$

3. Resultados

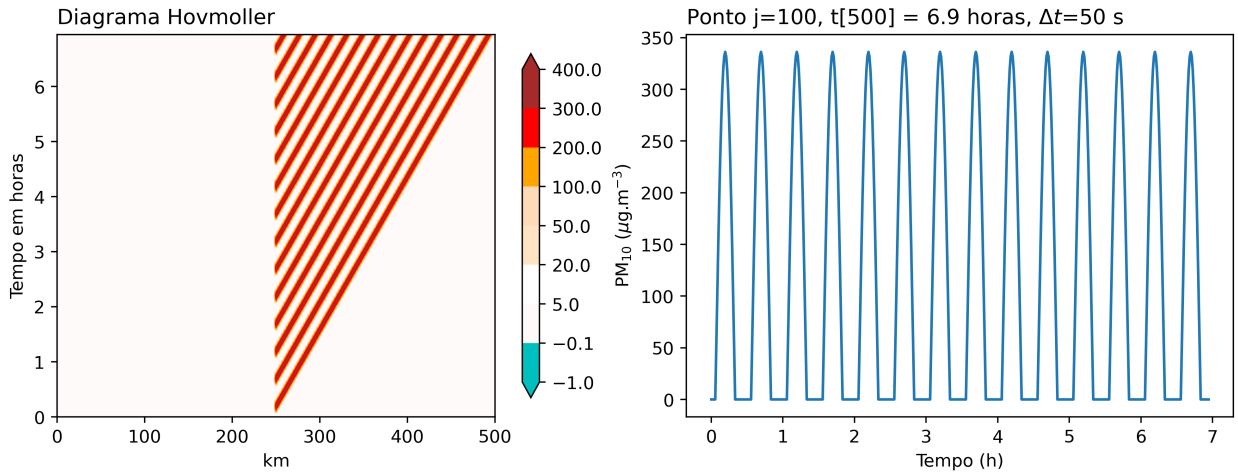

Figura 1: Diagrama Hovmoller da solução analítica (esquerda) e variação da fonte ao longo do tempo (direita).

A propagação da fonte como solução analítica é mostrada no diagrama Hovmoller da Fig. 1, onde também podemos notar (direita da figura) a variação senoidal da fonte ao longo do tempo. A advecção do lado direito desde o centro do domínio da grade é devido à velocidade do vento constante de sinal positivo. A magnitude da concentração é explicada pela integração mostrada na eq. 7. A solução analítica foi comparada com as aproximações numéricas mostradas na Fig. 2. As perturbações dos esquemas podem sair do domínio sem que aconteça reflexão devido à implementação do esquema *Euler forward-backward* como condição de contorno para os dois pontos finais à direita da grade. O esquema de ordem 1 da figura mostra concentrações menores de propagação se comparar com a solução analítica, pelo que o método é difuso. Os demais esquemas apresentam oscilações traseiras desde o centro que é contraditório ao comportamento físico do fenômeno (advecção à direita quando a velocidade do vento é positiva).

As aproximações do esquema de 4ª ordem mostram valores mais aproximados à solução analítica, no entanto com oscilações numéricas negativas. Portanto, para tornar o resultado mais realista, os filtros no tempo e espaço com as equações 3, 4 de Robert-Asselin foram aplicados somente para as oscilações do lado esquerdo desde o centro da grade. Os resultados filtrados são mostrados na Fig. 3 para os três métodos numéricos. Os resultados dos esquemas de leapfrog

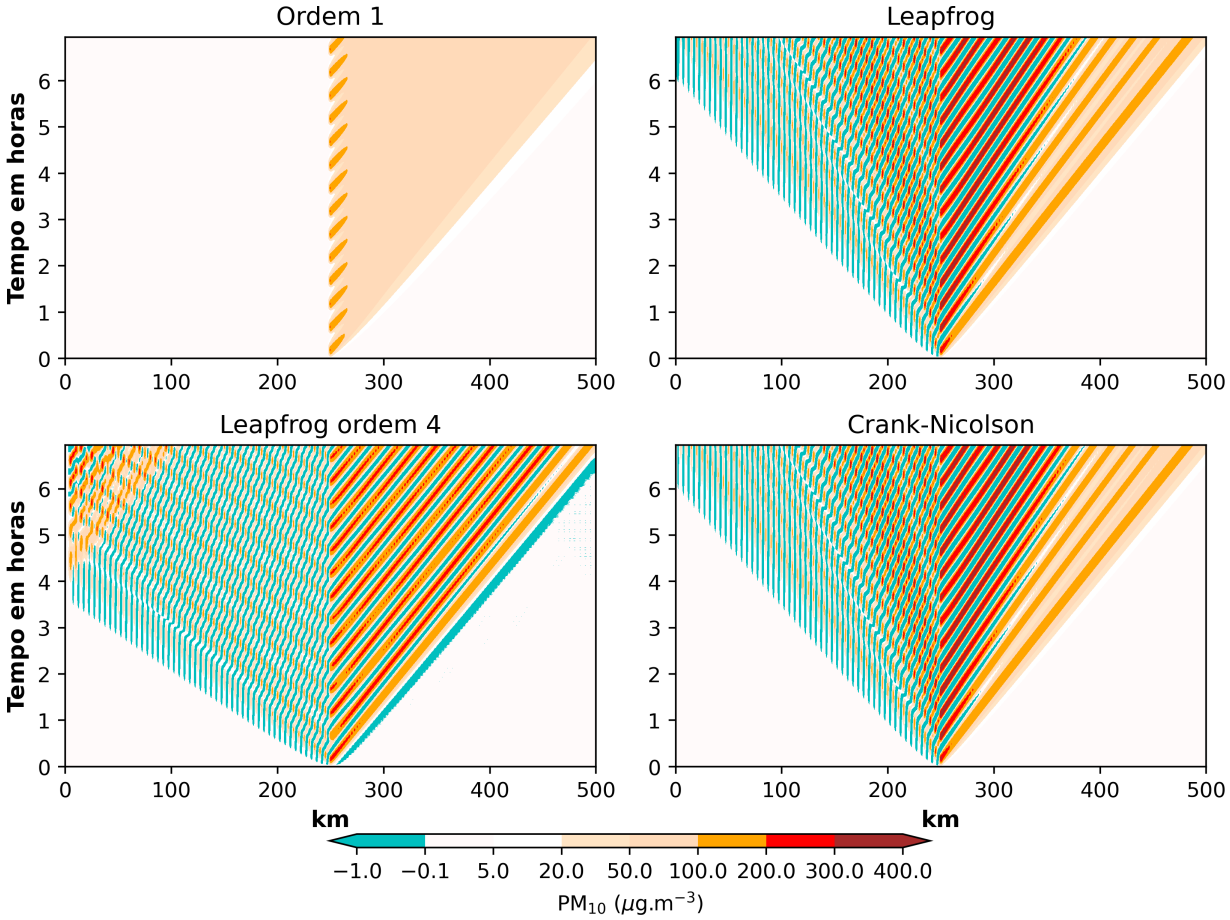

Figura 2: Soluções numéricas com diferentes esquemas para CFL = 0,2; Δt= 100 s.

e Crank-Nicolson tem semelhanças no modo da propagação depois da filtragem. No entanto, se a filtragem acontece para todos os pontos do domínio, as aproximações do lado direito também são afetadas apesar de usar um tipo de filtragem (espaço ou tempo) como é ilustrado na Fig. 4 que mostra resultados filtrados ((a) espaço, (b) tempo e (c) espaço-tempo) das aproximações do método leapfrog 2º ordem. Os resultados dos experimentos com o esquema Crank-Nicolson consideraram diferentes níveis de CFL ≥ 1 (Fig. 5). Os resultados com CFL = 1 tem um comportamento estável ao longo da integração. No entanto, as aproximações a partir de CFL ≥ 2 apresentam instabilidades depois de 20 horas de simulação como é o caso das aproximações com CFL = 4. A aplicação de filtros para remover as oscilações do lado esquerdo foi eficaz para CFL menores que 2; os resultados com CFL = 4 depois da aplicação dos filtros continuaram com as oscilações negativas.

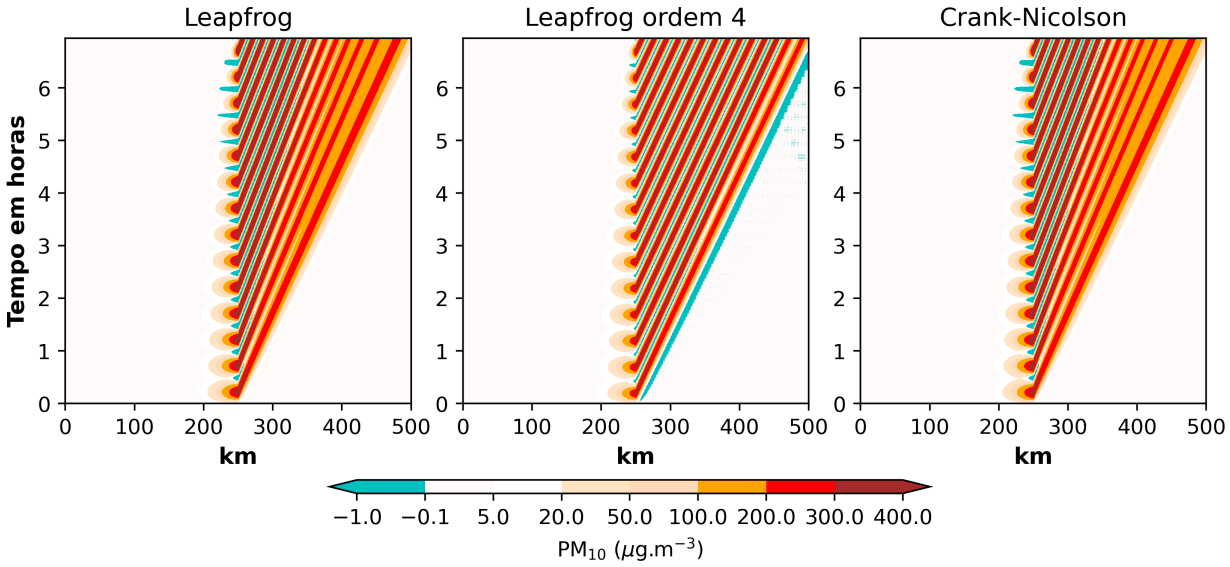

Figura 3: Soluções numéricas com aplicação de filtro no espaço e tempo aplicado somente para os resultados do lado esquerdo do centro do domínio.

4. Discussão dos resultados

A fonte sempre gera emissões instantáneas que são acumulativas se consideramos um Δt, pelo que é importante considerar o desenvolvimento da integral da eq. 7. O vento com magnitude positiva apresenta uma advecção da fonte da direcção direita desde o centro do domínio. A discretização da fonte foi considerada em cada método numérico onde é multiplicado por Δt se é de primeiro ordem no tempo ou por $2\Delta t$ se é de segunda ordem como no caso de leapfrog. A aproximação do ordem 1 progressivo no tempo e regressivo no espaço é um importante esquema que não precisa de filtros devido à simplicidade do método. No entanto, o esquema é difuso pelo que as concentrações são subestimadas se comparar com a solução analítica. Outros esquemas como leapfrog (2ª e 4ª ordem) e Crank-Nicolson geram resultados com oscilações que contradizem o fenômeno físico da solução analítica, pelo que a aplicação do filtros no espaço e tempo é importante para preservar a monotonicidade. No entanto, acontece atenuação das ondas se a filtragem acontece para todas as aproximações localizadas em todos os pontos do domínio. Em conclusão, a filtragem impacta a aproximação física, mas se somente é aplicada para as oscilações computacionais é efetiva em removê-los.

Bibliografia

Brasseur, G.P., Jacob, D.J. (2017). Modeling of Atmospheric Chemistry, First. ed. Cambridge University Press. https://doi.org/10.1017/9781316544754

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

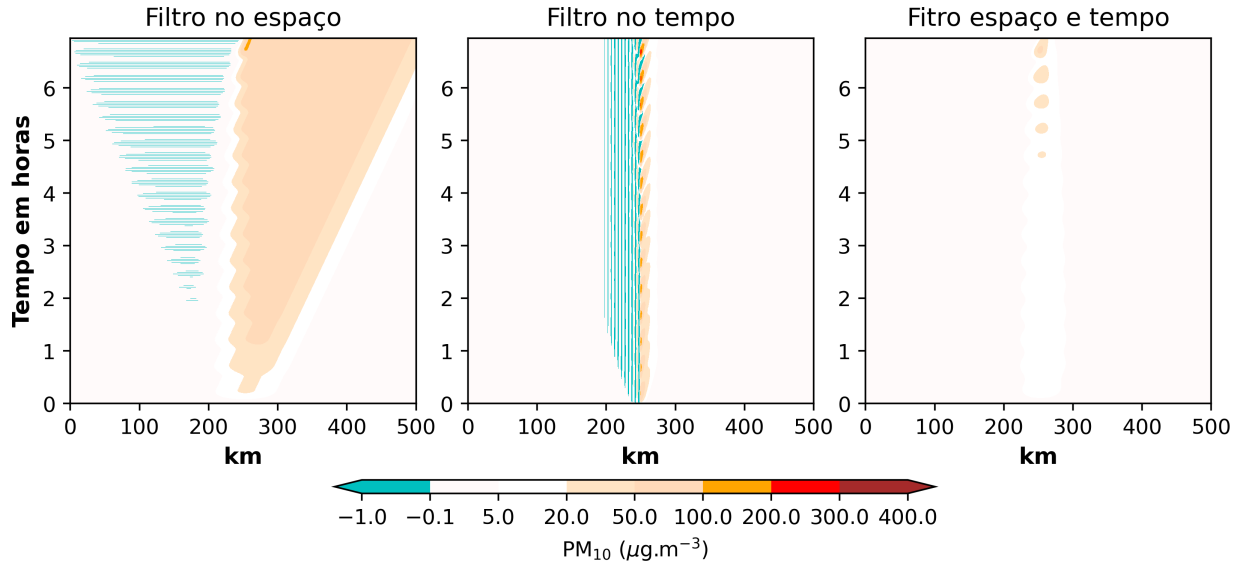

Figura 4: Filtragem com o método de Robert-Asselin no (a) espaço, (b) no tempo e (c) os dois para todos os resultados de leapfrog.

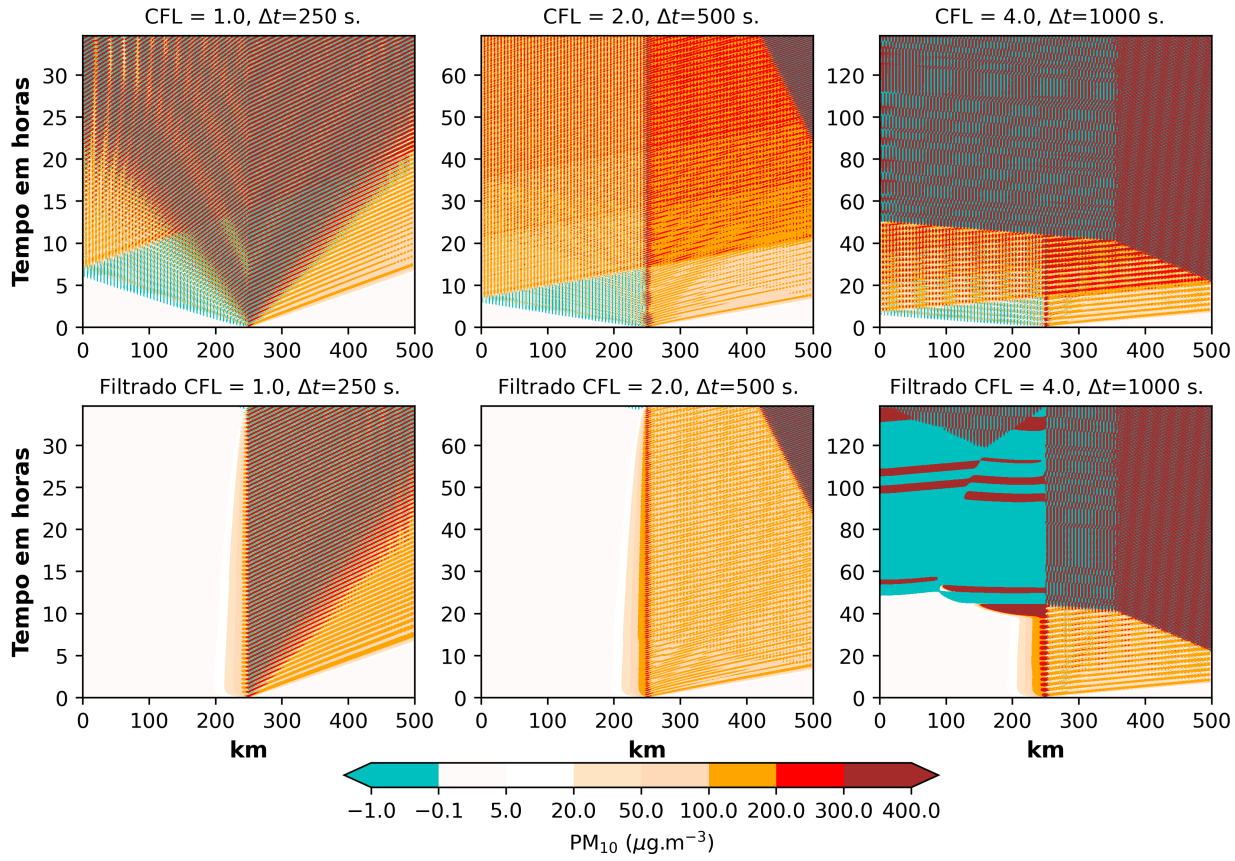

Figura 5: Variação do CFL ≥ 1 para o método implícito Crank-Nicolson. Os resultados sem filtragem são mostrados na parte superior da figura.

Apêndice A

A solução analítica considerou a aplicação da equação da onda harmônica, representado como

$$C^n = C^{n-1} - \frac{1}{\omega}\left[\cos(-x\,k + \omega t^n) - \cos(-x\,k + \omega t^{n-1})\right].$$

A equação está inserida no seguinte código em Python:

```
Nx, dx, Nt, dt, U  = 201, 2500, 501, 50, 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_s = np.zeros((Nx, Nt))
T, X = np.meshgrid(t, x)

for n in range(1, Nt):
    om=2*np.pi/1800
    k= om/U
    j = int(100 + U*t[n]/dx)
    C_s[100:j, n] = C_s[100:j, n-1] - (np.cos(-x[100:j]*k + om*t[n])/om - \
                      np.cos(-x[100:j]*k + om*t[n-1])/om)

C_s[C_s <= 0] = 0
```

A função do Euler progressivo no tempo e regressivo no espaço é mostrado em Python a continuação

```
C = np.zeros((Nx, Nt))
F = C.copy()

def euler_back(C, n, CFL, dt, F):
    C[1:, n] = C[1:, n-1] + F[1:, n-1]*dt - CFL*(C[1:, n-1]- C[:-1, n-1])
    return C

for n in range(1, Nt):
    F[100, n-1] = wave(t,n-1)
    # advecção ordem 1
    C = euler_back(C, n, CFL, dt, F)
```

para `C` como matriz de Nx e Nt pontos de grade no espaço e tempo, respetivamente. O exercício 1 considerou uma resolução espacial $\Delta x = 5000$; mas para representar o comprimento de onda, a resolução mudou para $\Delta x = 2500$ e um $\Delta t = 50$ segundos. A função leapfrog considera para o primer passo de tempo o esquema Euler progressivo-regressivo, depois o código considera o modo radiacional:

```
def leap2(C, n, CFL, dt, F, f_space=False, f_time = False, alfa=0.105):
    if n == 1:
    # Euler
        C = euler_back(C, n, CFL, dt, F)

    elif n > 1:
        C[1:-1,n] = C[1:-1,n-2]+2*dt*F[1:-1,n-1]-CFL*(C[2:, n-1]-C[:-2,n-1])
        # radiacional
        C[-1, n] = C[-1, n-1] + F[-1,n-1]*dt - CFL*(C[-1, n-1]- C[-2, n-1])

    # Filtro Robert-Asselin
    C = filtro(C, alfa, f_space, f_time)

    return C
```

```
C_l = np.zeros((Nx, Nt))
C_lf = C_l.copy()
F = C_l.copy()

for n in range(1, Nt):
    F[100,n-1] = wave(t,n-1)
    C_l = leap2(C_l, n, CFL, dt, F, f_space=False, f_time = False, alfa=0.105)

    C_lf = leap2(C_lf, n, CFL, dt, F, f_space=True, f_time=True)
```

A continuação a função de leapfrog 4ª ordem mostra os modos radiacionais com aplicação de Euler no primeiro passo de tempo:

```
def ordem4(C,n, CFL, dt, F, f_space=False, f_time = False, alfa=0.105):
    # Aprox leapfrog 4a ordem
    # --------------
    if n == 1:
        # Euler
        C = euler_back(C, n, CFL, dt, F)

    elif n > 1:
        C[2:-2, n] = C[2:-2,n-2] + F[2:-2,n-1]*2*dt - /
                     CFL/6*(C[:-4,n-1] - 8*C[1:-3,n-1] + 8*C[3:-1,n-1] - C[4:,n-1])
        # radiacional
        C[-2, n] = C[-2, n-1] + F[-2,n-1]*dt - CFL*(C[-2, n-1] - C[-3, n-1])
        C[-1, n] = C[-1, n-1] + F[-1,n-1]*dt - CFL*(C[-1, n-1] - C[-2, n-1])

    # Filtro Robert-Asselin
    C = filtro(C, alfa, f_space, f_time)

    return C

C_4o = np.zeros((Nx, Nt))
C_4of = C_4o.copy()
F = C_4o.copy()

for n in range(1, Nt):
    F[100,n-1] = wave(t,n-1)
    # Não filtrado
    C_4o = ordem4(C_4o,n, CFL, dt, F)
    # Filtrado
    C_4of = ordem4(C_4of,n, CFL, dt, F, f_space=True, f_time=True)
```

Finalmente, o método implícito escolhido é o esquema Crank-Nicolson com as seguintes funções:

```
def crank_matrix(CFL, x, uc=0.5):
    import scipy.sparse as sp
    uns   = np.ones(len(x))
    r     = uns*CFL/2
    diags = (-1, 0, 1)  # -1 low diagonal, 0 main diagonal, 1 upper diagonal
    A = sp.spdiags( [-uc*r, uns, uc*r], diags, len(x), len(x) )
    A = (sp.lil_matrix(A)).tocsr()
    B = sp.spdiags( [(1-uc)*r, uns, -(1-uc)*r], diags, len(x), len(x) )
    B = (sp.lil_matrix(B)).tocsr()
```

```
    return A, B

def crank(A, B, C, n, CFL, f_space=False, f_time = False, alfa=0.105):
    C[:, n] = spsolve(A, B*C[:, n-1])
    C[-1, n] = C[-1, n-1] - CFL*(C[-1, n-1] - C[-2, n-1])

    # Filtro Robert-Asselin
    C = filtro(C, alfa, f_space, f_time)

    return C
```

Com o fim de preservar a função de resolução do esquema Crank-Nicolson, a fonte discretizada foi somada ao campo nulo, escrito em Python como segue:

```
C_cr = np.zeros((Nx, Nt))
C_crf = C_cr.copy()
F = C_cr.copy()
A, B = crank_matrix(CFL, x)

for n in range(1, Nt):
    F[100,n-1] = wave(t, n-1)
    C_cr[:,n-1] = C_cr[:,n-1] + dt*F[:,n-1]
    C_cr = crank(A, B, C_cr, n, CFL, f_space=False, f_time = False)

    C_crf[:,n-1] = C_crf[:,n-1] + dt*F[:,n-1]
    C_crf = crank(A, B, C_crf, n, CFL, f_space=True, f_time = True)
```

Finalmente, o seguinte código em Python mostram os filtros Robert-Asselin que removem o modo computacional. Neste trabalho, somente foi aplicado até a metade da grade devido a que a filtragem afeta significativamente os resultados de advecção da fonte.

```
def filtro(C, alfa, f_space=False, f_time=False):
    """Filtro Robert-Asselin para remover o modo computacional

    Args:
        C (array): Campos nulos com a emissão na metade
        n (int): passo de tempo
        alfa (float, optional): Coeficiente.
        f_space (bool, optional): Filtro no espaço. Defaults to False.
        f_time (bool, optional): Filtro no tempo. Defaults to False.

    Returns:
        array: valores de C filtrados
    """
    # Filtro Asselin desde a mitade da grade
    if f_space == True:
        C[1:100,:] = C[1:100, :] + alfa*(C[:99,:] - 2*C[1:100, :] + C[2:101, :])

    if f_time == True:
        C[:100,1:-1] = C[:100,1:-1]+alfa*(C[:100,:-2] - 2*C[:100,1:-1] + C[:100,2:])

    else:
        pass

    return C
```