Universidade de Brasília

Equação da Advecção

Disciplina: Métodos Numéricos em Termofluidos
Professor: Adriano Possebon Rosa

Departamento de Engenharia Mecânica
Faculdade de Tecnologia
Universidade de Brasília

Sumário

A advecção é o transporte de uma propriedade (energia, poluente, temperatura, entropia, massa, etc) por um fluido, devido ao seu movimento. O problema da advecção é fundamental na Mecânica dos Fluidos e possui aplicações em diversas áreas.

1 Equação

A equação da advecção pura, em uma dimensão, é dada por

$$\frac{\partial \phi}{\partial t} + v\frac{\partial \phi}{\partial x} = 0 \ , \tag{1}$$

em que ϕ é a propriedade que está sendo transportada (ϕ pode ser a concentração de uma substância, a temperatura, a massa específica, a energia, a entropia ou a própria velocidade) e v é a velocidade com que a propriedade é transportada. Vamos considerar v como sendo uma constante positiva, por enquanto.

Essa equação é uma equação diferencial parcial hiperbólica. Esse tipo de equação possui uma condição inicial que é avançada no tempo e representa uma propagação no espaço, com uma velocidade finita.

Queremos encontrar a solução $\phi(x,t)$. Como condição inicial temos

$$\phi(x,0) = f(x) \ . \tag{2}$$

A solução analítica para esse problema é uma função do tipo

$$\phi(x,t) = f(x - vt) \ , \tag{3}$$

ou seja, o formato da solução inicial é mantido ao longo do tempo, com um deslocamento para a direita proporcional à velocidade de propagação.

Vamos considerar, nesta aula, a condição inicial dada por

$$\begin{array}{rl} 0, & -\infty < x < 0.5 \\ \sin(2x-1)), & 0.5 \leq x \leq \pi/2 + 0.5 \\ 0, & x > \pi/2 + 0.5 \ , \end{array} \tag{4}$$

como mostra a figura (1), e uma velocidade $v = 1$. A curva azul à esquerda, na figura, representa a condição inicial, ou seja, em $t = 0$. A curva do meio representa a solução analítica em $t = 3.5$ e a curva da direita em $t = 7.0$. Note que a equação da advecção transporta a condição inicial para a direita, sem alterar a sua forma. Ao longo desta aula, vamos utilizar diferentes métodos numéricos para resolver o problema da advecção com essa condição inicial.

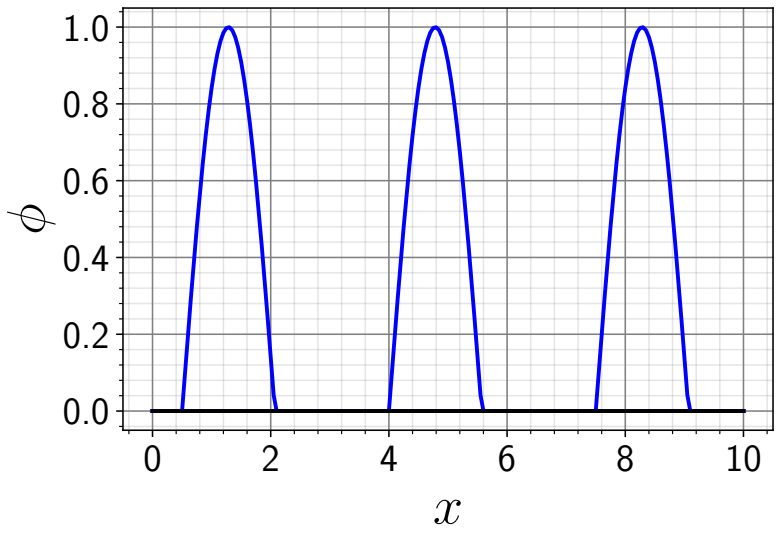


Figura 1: Solução analítica da equação da advecção pura.

A parte advectiva é um dos maiores desafios da Dinâmica dos Fluidos Computacional: muitos pesquisadores desenvolvem e testam métodos para conseguir resolver essa equação de forma precisa.

2 Solução Numérica

Temos a seguinte equação:

$$\frac{\partial \phi}{\partial t} = -v \frac{\partial \phi}{\partial x} \ . \tag{5}$$

Vamos começar com um método de Euler explícito para o tempo e um método de diferenças centradas para a derivada espacial. A partir de uma expansão em série de Taylor, idêntica à que fizemos na parte da equação do calor, temos

$$\frac{\phi_i^{k+1} - \phi_i^k}{\Delta t} = -v\left(\frac{\phi_{i+1}^k - \phi_{i-1}^k}{2\Delta x}\right) + O(\Delta t, \Delta x^2) \ . \tag{6}$$

Esse método é de primeira ordem no tempo e de segunda ordem no espaço. Truncando a aproximação e isolando ϕ_i^{k+1}, resulta

$$\phi_i^{k+1} = \phi_i^k - \frac{v\Delta t}{2\Delta x}\left(\phi_{i+1}^k - \phi_{i-1}^k\right) \ . \tag{7}$$

Para implementar esse método, eu utilizei um domínio de tamanho 10, com $0 \leq x \leq 10$. O domínio foi dividido em 200 pontos, com $\Delta x = 0.05$. Foi utilizado um $\Delta t = 0.025$. O código completo pode ser encontrado em anexo. A figura (2) mostra o resultado para $t = 0.5$. A curva azul, à esquerda, representa a condição inicial, em $t = 0$, a curva azul à direita representa a solução analítica em $t = 0.5$ e a curva vermelha representa a solução numérica em $t = 0.5$. Nota-se que o resultado numérico não é bom, pois aparecem várias oscilações que não têm significado físico, e a onda já está bastante distorcida. Mas calma que vai piorar! A figura (3) mostra o resultado para $t = 1$ e a figura (4) para $t = 1.5$. A solução piora com o avanço do tempo: o valor de ϕ cresce até chegar no limite do computador, e nesse momento o seu código dá um erro. Isso é "explodir".

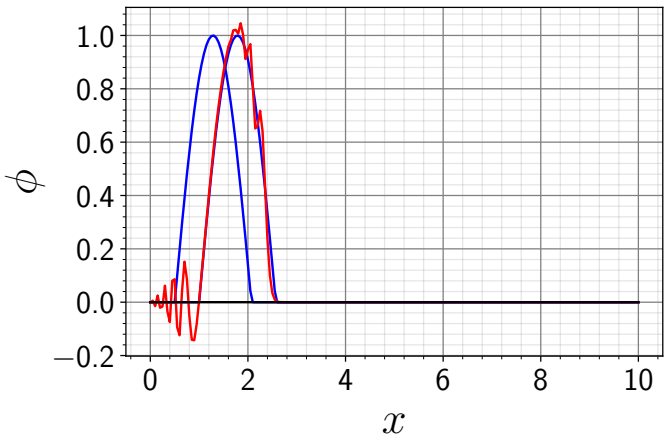


Figura 2: Método de diferenças centradas em $t = 0.5$.

Portanto, os resultados gerados por esse método não são bons. Uma análise de estabilidade de von Neumann mostra que o método é incondicionalmente instável, ou seja, ele sempre gera respostas que irão crescer indefinidamente no tempo.

Fisicamente, a equação da advecção transporta uma propriedade em uma direção. Podemos concluir que o valor da propriedade será afetado mais pelos valores que estão a montante. Por exemplo: a temperatura em uma dada região de um rio com correnteza é influenciada mais pela temperatura da água que está antes (montante) do que pela temperatura da água que está depois (jusante) [1].

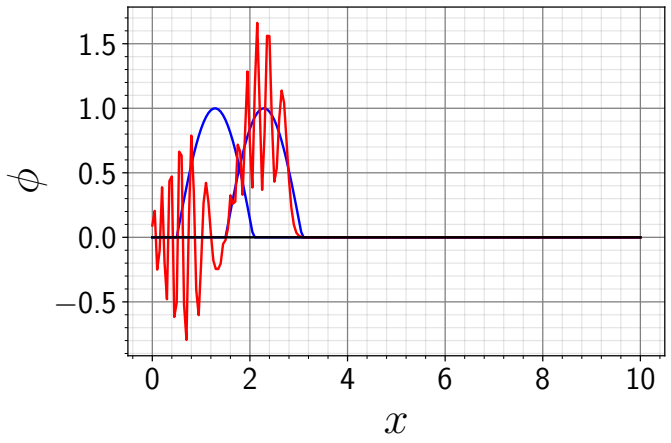


Figura 3: Método de diferenças centradas em $t = 1$.

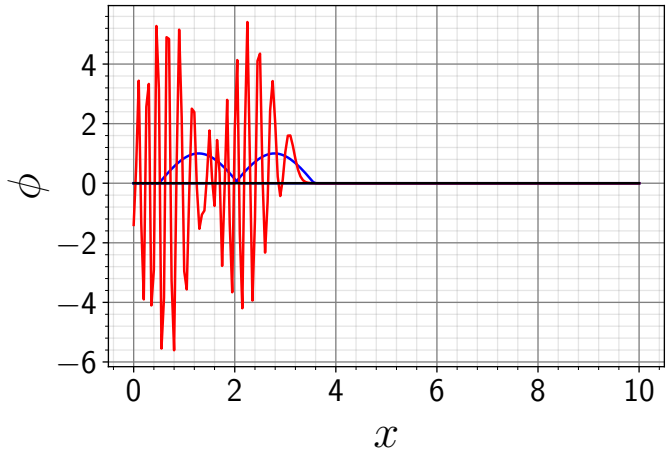


Figura 4: Método de diferenças centradas em $t = 1.5$.

Métodos que levam em consideração o sentido de propagação local da propriedade em estudo são chamada de métodos ***upwind***.

Uma aproximação de *upwind* de primeira ordem é dada por

$$v\frac{\partial \phi}{\partial x} \approx v\left(\frac{\phi_i - \phi_{i-1}}{\Delta x}\right) \tag{8}$$

se $v > 0$ ou

$$v\frac{\partial \phi}{\partial x} \approx v\left(\frac{\phi_{i+1} - \phi_i}{\Delta x}\right) \tag{9}$$

se $v < 0$. Note que agora as aproximações não são centradas, mas tendem mais para um dos dois lados. Como no nosso caso de estudo a velocidade v é constante e positiva, vamos utilizar a forma dada na equação (8).

A discretização da equação da advecção se torna

$$\frac{\phi_i^{k+1} - \phi_i^k}{\Delta t} + v\left(\frac{\phi_i^k - \phi_{i-1}^k}{\Delta x}\right) = 0\ . \tag{10}$$

Isolando ϕ_i^{k+1}, temos

$$\phi_i^{k+1} = \phi_i^k - \frac{v\Delta t}{\Delta x}\left(\phi_i^k - \phi_{i-1}^k\right) \ . \tag{11}$$

A figura (5) mostra o resultado do problema que estamos resolvendo quando aplicamos esse método *upwind* de primeira ordem. Temos aqui $\Delta x = 0.05$ e $\Delta t = 0.025$. Note que a solução numérica não diverge, o que já é uma vitória. No entanto, percebemos agora que há uma grande difusão de ϕ ao longo do tempo. Veremos o porquê disso daqui a pouco. Antes, Courant e CFL.

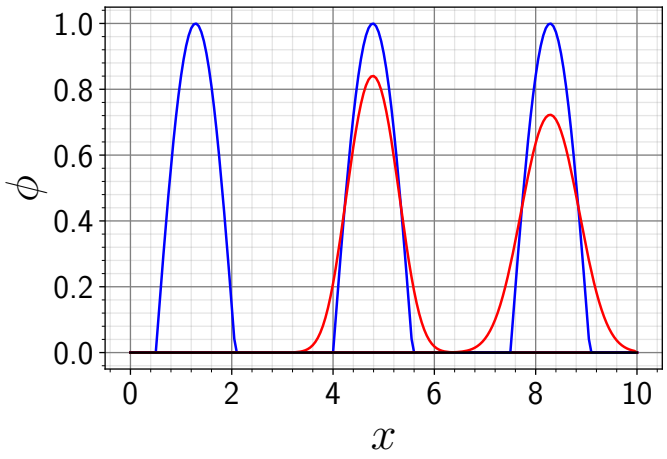


Figura 5: Método *upwind* de primeira ordem, para $t = 0$, $t = 3.5$ e $t = 7$.

Uma análise de estabilidade de von Neumann aplicada ao método de *upwind* de primeira ordem mostra que o método é estável se

$$\Delta \leq \frac{\Delta x}{v} \ . \tag{12}$$

A partir disso foi definido o número de Courant

$$C = \frac{v\Delta t}{\Delta x} \ . \tag{13}$$

Assim, para que o método seja estável devemos ter

$$C \leq 1 \ . \tag{14}$$

Essa condição é chamada de condição CFL (Courant-Friedrichs-Lewy) [2].

No resultado da figura (5), temos $C = 0.5$. A figura (6) mostra o que acontece quando $\Delta t = 0.06$, resultando em $C = 1.2$, o que não respeito a condição de CFL.

Uma interpretação da condição de CFL é que a velocidade de propagação numérica $\Delta x/\Delta t$ deve ser menor que a velocidade de propagação física v. Quando isso não ocorre, vão surgindo pequenos erros nas contas que acabam por destruir a simulação.

E por que há difusão ou amortecimento da condição inicial com esse método? A difusão observada no esquema de *upwind* de primeira ordem pode ser explicada a partir de uma análise mais cuidadosa da equação de diferenças finitas.

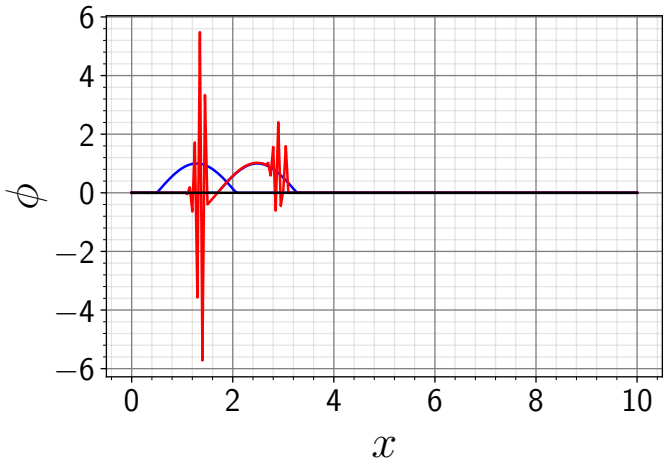


Figura 6: Método *upwind* de primeira ordem com $C = 1.2$, para $t = 1.2$.

Considerando uma velocidade positiva, vimos que o método é dado por

$$\frac{\phi_i^{k+1} - \phi_i^k}{\Delta t} + v\left(\frac{\phi_i^k - \phi_{i-1}^k}{\Delta x}\right) = 0\ . \tag{15}$$

Vamos escrever os termos usando séries de Taylor, ou seja,

$$\phi_i^{k+1} = \phi_i^k + \Delta t \frac{\partial \phi}{\partial t}\bigg|_{x_i,t_k} + \frac{\Delta t^2}{2}\ \frac{\partial^2 \phi}{\partial t^2}\bigg|_{x_i,t_k} + \frac{\Delta t^3}{3!}\ \frac{\partial^3 \phi}{\partial t^3}\bigg|_{x_i,t_k} + \ \cdots \tag{16}$$

$$\phi_{i-1}^k = \phi_i^k - \Delta x \frac{\partial \phi}{\partial x}\bigg|_{x_i,t_k} + \frac{\Delta x^2}{2}\ \frac{\partial^2 \phi}{\partial x^2}\bigg|_{x_i,t_k} - \frac{\Delta x^3}{3!}\ \frac{\partial^3 \phi}{\partial x^3}\bigg|_{x_i,t_k} + \ \cdots \tag{17}$$

Substituindo essas séries de volta na equação (15), resulta

$$\begin{aligned} &\frac{\partial \phi}{\partial t}\bigg|_{x_i,t_k} + \frac{\Delta t}{2}\ \frac{\partial^2 \phi}{\partial t^2}\bigg|_{x_i,t_k} + \frac{\Delta t^2}{3!}\ \frac{\partial^3 \phi}{\partial t^3}\bigg|_{x_i,t_k} + \ \cdots \\ &+ v\left(\frac{\partial \phi}{\partial x}\bigg|_{x_i,t_k} - \frac{\Delta x}{2}\ \frac{\partial^2 \phi}{\partial x^2}\bigg|_{x_i,t_k} + \frac{\Delta x^2}{3!}\ \frac{\partial^3 \phi}{\partial x^3}\bigg|_{x_i,t_k} + \ \cdots\right) = 0\ . \end{aligned} \tag{18}$$

Assim, recuperamos a equação original

$$\begin{aligned} &\frac{\partial \phi}{\partial t}\bigg|_{x_i,t_k} + v\ \frac{\partial \phi}{\partial x}\bigg|_{x_i,t_k} = v\frac{\Delta x}{2}\ \frac{\partial^2 \phi}{\partial x^2}\bigg|_{x_i,t_k} - \frac{\Delta t}{2}\ \frac{\partial^2 \phi}{\partial t^2}\bigg|_{x_i,t_k} - \frac{\Delta t^2}{3!}\ \frac{\partial^3 \phi}{\partial t^3}\bigg|_{x_i,t_k} + \ \cdots \\ &- v\frac{\Delta x^2}{3!}\ \frac{\partial^3 \phi}{\partial x^3}\bigg|_{x_i,t_k} + \ \cdots = 0\ . \end{aligned} \tag{19}$$

Note que se fizermos $\Delta x \to 0$ e $\Delta t \to 0$, recuperamos a equação original da advecção pura, como esperado. Isso nos mostra que a aproximação de diferenças finitas é consistente com a equação diferencial parcial. No entanto, os termos $O(\Delta x)$ e $O(\Delta t)$

podem ter grande influência na solução. Para ver isso, primeiro, podemos derivar a equação original, com relação ao tempo, para ver que

$$\frac{\partial^2 \phi}{\partial t^2} = -v \frac{\partial}{\partial x}\left(\frac{\partial \phi}{\partial t}\right) = v^2 \frac{\partial^2 \phi}{\partial x^2} \ . \tag{20}$$

Ao substituirmos essa relação de volta na equação (19), obtemos

$$\left.\frac{\partial \phi}{\partial t}\right|_{x_i,t_k} + v \left.\frac{\partial \phi}{\partial x}\right|_{x_i,t_k} = \left(\frac{v\Delta x}{2} - \frac{v^2 \Delta t}{2}\right) \left.\frac{\partial^2 \phi}{\partial x^2}\right|_{x_i,t_k} + O(\Delta x^2, \Delta t^2) \ . \tag{21}$$

Ou seja, também estamos aproximando a equação diferencial parcial da advecção com difusão, dada por

$$\frac{\partial \phi}{\partial t} + v \frac{\partial \phi}{\partial x} = \nu_{art} \frac{\partial^2 \phi}{\partial x^2} \ , \tag{22}$$

em que

$$\nu_{art} = v \frac{\Delta x}{2} (1 - C) \tag{23}$$

é chamada de viscosidade artificial ou numérica. Essa viscosidade é indesejada, no presente caso. Mas em outras situações, na dinâmica dos fluidos computacional, ela é adicionada propositalmente, para ajudar na estabilidade da solução numérica (mesmo com uma penalidade na precisão da resposta final).

Por isso esse método é tão difusivo. Para diminuir a difusão temos que deixar Δx pequeno ou $C \approx 1$. A figura (7) mostra o resultado do método com $C = 1$. Neste caso, o método numérico recupera a solução exata. Isso pode ser visto a partir da equação de diferenças, dada por

$$\phi_i^{k+1} = \phi_i^k - C\left(\phi_i^k - \phi_{i-1}^k\right) \ . \tag{24}$$

Com $C = 1$, temos $\phi_i^{k+1} = \phi_{i-1}^k$, mostrando que a solução numérica acompanha a solução analítica. Já a figura (8) mostra o resultado quando $C = 0.1$. Com Δx fixo, isso leva a um aumento da viscosidade artificial, resultando em uma maior difusão da solução inicial ao longo do tempo.

Na prática é muito difícil conseguirmos manter $C \approx 1$ nas simulações, especialmente em casos em duas e três dimensões. Então as opções que nos restam são diminuir o Δx ou escolher um outro método que não seja tão difusivo.

Um método mais preciso, de $O(\Delta t^2, \Delta x^2)$, é o método de Lax-Wendroff, escrito como [1]

$$\phi_i^{k+1} = \phi_i^k - \frac{C}{2}\left(\phi_{i+1}^k - \phi_{i-1}^k\right) + \frac{C^2}{2}\left(\phi_{i+1}^k - 2\phi_i^k + \phi_{i-1}^k\right) \ . \tag{25}$$

Esse método se baseia no método de Euler com diferenças centradas. Temos, agora, a adição de um termo de difusão artificial, que faz com que o método original de diferenças centradas se torne estável. A adição de difusão artificial puramente numérica, como o objetivo de estabilizar uma solução, é um artifício muito usado em problemas de mecânica dos fluidos [3]. O método de Lax-Wendroff é condicionalmente estável, devendo ter $C \leq 1$.

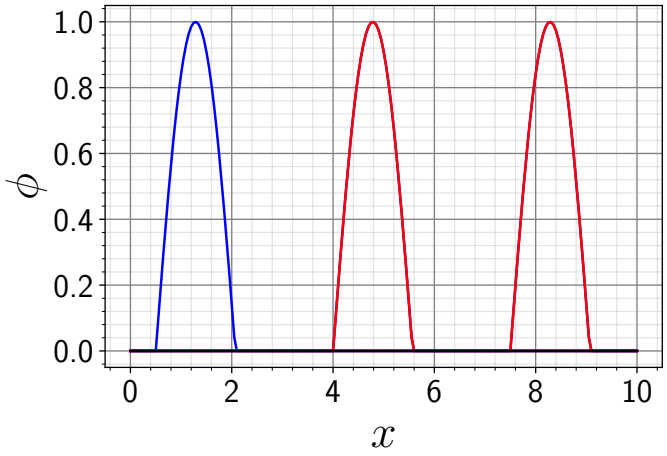


Figura 7: Método *upwind* de primeira ordem com $C = 1$, para $t = 0$, $t = 3.5$ e $t = 7$.

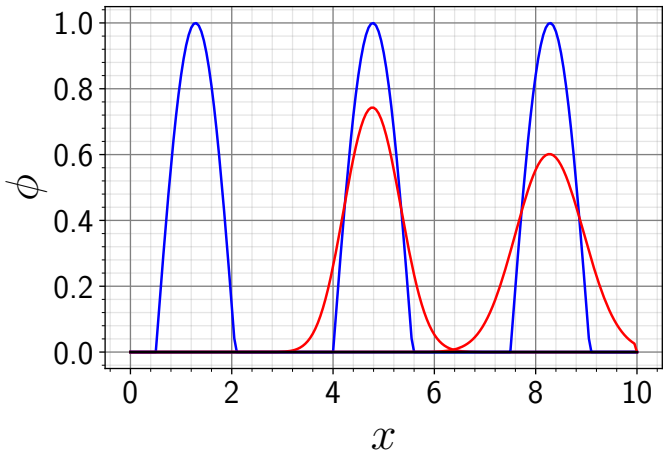


Figura 8: Método *upwind* de primeira ordem com $C = 0.1$, para $t = 0$, $t = 3.5$ e $t = 7$.

Fazendo uma expansão dos termos da séria (25), como fizemos para o método de *upwind* de primeira ordem, temos

$$\frac{\partial \phi}{\partial t} + v\frac{\partial \phi}{\partial x} = -\frac{v\Delta x^2}{6}(1 - C^2)\frac{\partial^3 \phi}{\partial x^3} + O(\Delta^3) \ , \tag{26}$$

Note que o primeiro termo de erro que aparece é uma derivada de terceira ordem. Derivadas de ordem par, quando truncadas, contribuem com uma difusão numérica artificial. Já derivadas de ordem ímpar contribuem com oscilações indesejadas na solução. Essas oscilações são chamadas de dispersão numérica. Elas aparecem quando o método numérico faz com que ondas de diferentes comprimentos s propaguem com velocidades diferentes. A figura (9) mostra o resultado do método de Lax-Wendroff. Observe as oscilações que surgem à esquerda da solução.

Por fim, um método *upwind* de segunda ordem no tempo e no espaço, conside-

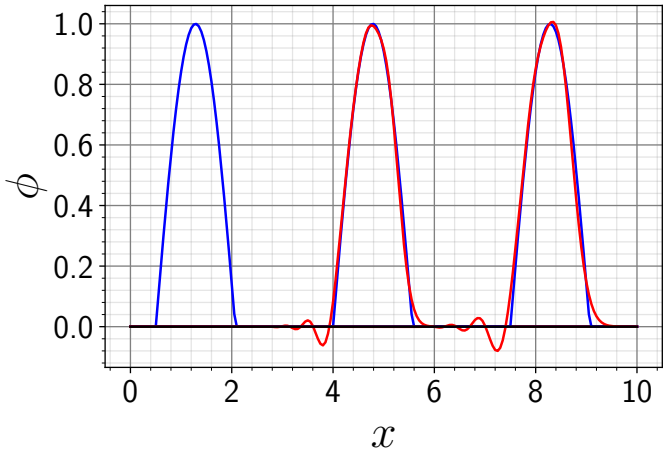


Figura 9: Método de Lax-Wendroff, com $C = 0.5$, para $t = 0$, $t = 3.5$ e $t = 7$.

rando $v > 0$, é dado por [4]:

$$\phi_i^{k+1} = \phi_i^k - C\left(\phi_i^k - \phi_{i-1}^k\right) - C\frac{(1-C)}{2}\left(\phi_i^k - 2\phi_{i-1}^k + \phi_{i-2}^k\right) \ . \tag{27}$$

A figura (10) mostra o resultado quando esse método é utilizado. Há pouca difusão numérica e a onda mantém a sua forma original. No entanto, é possível observar que há dispersão, com a formação de uma pequena onda que viaja à frente da solução.

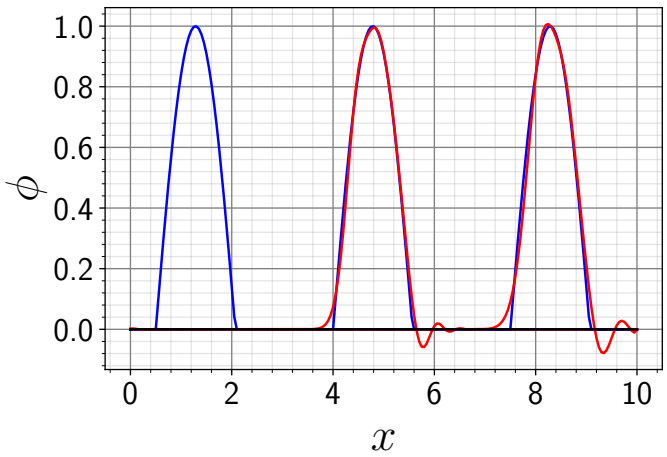


Figura 10: Método de *upwind* de segunda ordem no tempo e no espaço, com $C = 0.5$, para $t = 0$, $t = 3.5$ e $t = 7$.

Existem diversas outras aproximações para resolver a equação da advecção. Para uma lista mais completa de métodos, recomendo o livro do Hirsch [5], além dos livros do Hoffman e do Fortuna, já citados no texto.

Quando a equação é não linear, como no caso da equação de Burgers,

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = 0 \ , \tag{28}$$

ou na própria equação de Navier-Stokes, o problema fica bem mais complicado e devemos tomar mais cuidado ainda.

3 Advecção na Equação de Navier-Stokes

Na equação de Navier-Stokes incompressível em duas dimensões,

$$\rho\left[\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right]=-\frac{\partial p}{\partial x}+\mu\left[\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right] \tag{29}$$

e

$$\rho\left[\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right]=-\frac{\partial p}{\partial y}+\mu\left[\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}\right], \tag{30}$$

observamos a presença da advecção, nos termos destacados em azul. Nesse caso, a própria velocidade (ou melhor, a quantidade de movimento) está sendo transportada pela velocidade local do escoamento. Os termos advectivos são altamente não lineares e trazem grande complexidade à equação de Navier-Stokes. Os termos em vermelho nas equações (29) e (30) são os termos difusivos. O comportamento desses termos foi estudado nas últimas aulas.

Nosso próximo objetivo é resolver a equação de Navier-Stokes cheia, com a parte advectiva, a parte difusiva e a pressão. Esse é o assunto das próximas aulas.

Referências

[1] A. O. Fortuna. *Técnicas Computacionais para a Dinâmica dos Fluidos: Conceitos Básicos e Aplicações*. Edusp, São Paulo, 2nd edition, 2012.

[2] R. Courant, K. O. Friedrichs, and H. Lewy. On the Partial Difference Equations of Mathematical Physics. *IBM Journal of Research and Development*, 11:215–234, 1967.

[3] H. P. Langtangen and S. Linge. *Finite Difference Computing with PDEs - A Modern Software Approach*. 2016.

[4] J. D. Hoffman. *Numerical Methods for Engineers and Scientists*. Marcel Dekker, Inc., New York, 2nd edition, 2001.

[5] C. Hirsch. *Numerical Computation of Internal and External FLows*. Elsevier, New York, 2nd edition, 2007.

adveccao

March 23, 2023

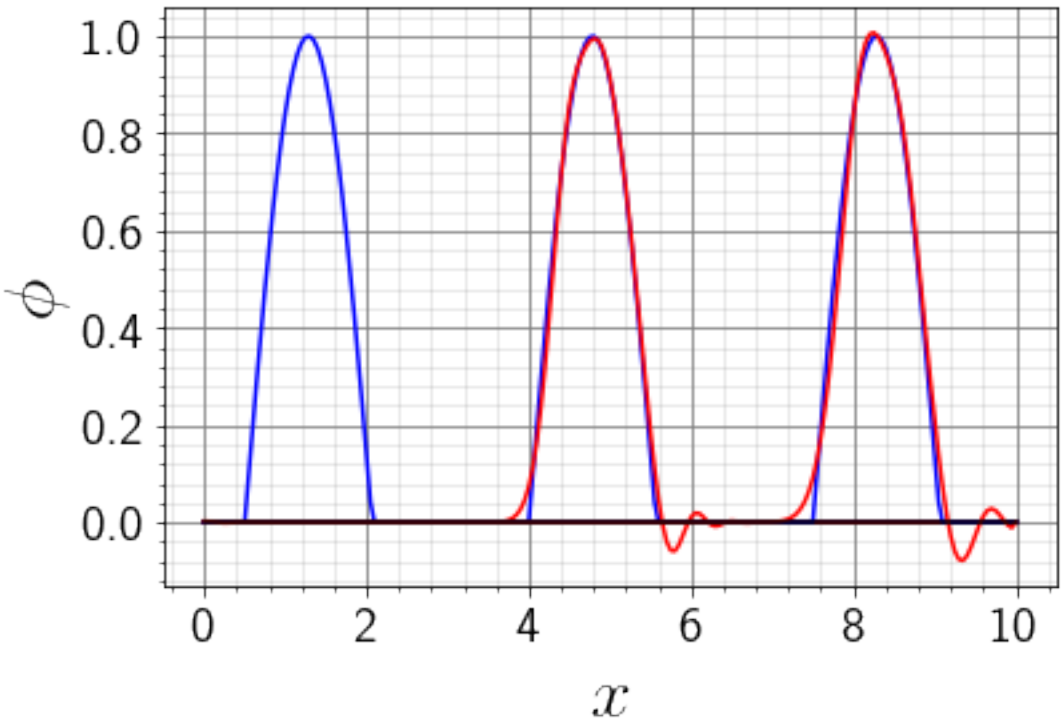
[1]:
```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import AutoMinorLocator
```

[4]:
```
v = 1.0    # Velocidade de advecção.
L = 10.0    # Tamanho do domínio.
tempo_final = 7.0    # Tempo final. Use 3.5 e 7.0.
N = 200    # Número de pontos no domínio espacial.

dt = 0.025   # Passo de tempo.
dx = L/N    # Distância entre os pontos.

C = v*dt/dx    # Número de Courant.

# Printando dx na tela.
print(f"dx: {dx:.2e}")
# Printando o Número de Courant na tela.
print(f"Número de Courant: {C:.2e}")

x = np.linspace(0.0, L, N+1)    # Vetor x, usado para plotar.

tempo = 0.0    # Tempo.

phi = np.zeros(N+1, float)    # Criando o vetor principal, phi.

# Implementando a condição inicial.
tipo = 0    # Tipo da função (0: seno, 1: normal, 2: degrau).
condicao_inicial(phi, x, tempo, v, tipo)

phi_novo = np.copy(phi)    # Vetor auxiliar, para avançar phi no tempo.

# Número de passos de tempo dados.
k = 0
# Número de passos de tempo final.
k_final = int(tempo_final/dt)
# Intervalo, em passos de tempo, para plotar.
# Mude o número do denominador para 2, 3, 4, ...,
```

```
# para ter mais curvas plotadas no gráfico final.
k_plot = int(k_final/2)


# Configurando o gráfico.
configurar_grafico()

# Plotando phi ao longo de x, no tempo 0.
plot_phi(x, phi, 'b')

# Começando o loop no tempo.
# Este é o loop principal do código.
while tempo <= tempo_final - 0.0001*dt:

    for i in range(N):

        # Euler explícito com diferenças centradas.
        #phi_novo[i] = diferencas_centradas(i, C, phi)

        # Euler explícito com upwind de primeira ordem.
        #phi_novo[i] = upwind_primeira_ordem(i, C, phi)

        # Euler explícito com upwind de terceira ordem misto.
        #phi_novo[i] = upwind_terceira_ordem_misto(i, C, phi)

        # Método de Lax-Wendroff.
        #phi_novo[i] = lax_wendroff(i, C, phi)

        # Método de segunda ordem no tempo e no espaço.
        phi_novo[i] = upwind_segunda_ordem(i, C, phi)

    # Atualizando o vetor phi.
    phi = np.copy(phi_novo)

    # Atualizando o tempo.
    tempo += dt

    # Atualizando a contagem de números de passos de tempo.
    k += 1

    # Plotando phi.
    if (k % k_plot == 0):
        phi_analitico = np.zeros(N+1, float)
        plot_phi(x, condicao_inicial(phi_analitico, x, tempo, v, tipo), 'b')
        plot_phi(x, phi, 'r', '-')

# Printando na tela o tempo final.
```

```
print(f'{tempo:.3f}')

# Plotar uma linha preta em zero.
# Apenas para melhorar a visualização.
phi = np.zeros(N+1, float)
plot_phi(x, phi, 'k')

# Exportando o gráfico em formato pdf.
plt.savefig('fig01.pdf', format='pdf', dpi=100, bbox_inches='tight')
```

```
dx: 5.00e-02
Número de Courant: 5.00e-01
7.000
```



[2]:
```
# Implementando a condição inicial.
def condicao_inicial(phi, x, t, v, tipo):
    """
    Retorna a condição inicial no tempo zero
    ou a solução analítica quanto o tempo
    é maior que zero.

    Args:
        i (int): posição no espaço.
```

```
        C (float): número de Courant.
        phi (numpy array): vetor da propriedade
            que está sendo advectada.
        x (numpy array): vetor com as posições.
        t (float): tempo.
        v (float): velocidade, constante.
        tipo (int): tipo da função. 0 é um seno,
            1 é uma distribuição normal e 2 é um
            degrau.

    Returns:
        phi (numpy array): vetor da propriedade.
    """

    # Função seno.
    for i in range(N):
        y = x[i] - v*t

        # Função seno.
        if tipo == 0:
            if y - 0.5 > 0.0 and y - 0.5 < np.pi/2:
                phi[i] = np.sin(2.0*(y-0.5))

        # Distribuição normal.
        if tipo == 1:
            phi[i] = np.exp(-(y - 1.0)**2.0/0.2)

        # Degrau.
        if tipo == 2:
            if y >= 1.0 and y <= 2.0:
                phi[i] = 1.0

    return phi


# Diferenças centradas.
def diferencas_centradas(i:int, C:float, phi:np.ndarray) -> float:
    """
    Avança phi no tempo, usando o método de Euler
    explícito e o método de diferenças centradas.

    Args:
        i (int): posição no espaço.
        C (float): número de Courant.
        phi (numpy array): vetor da propriedade
            que está sendo advectada.
```

```
    Returns:
        float: valor do novo phi no ponto i.
    """
    return phi[i] - 0.5*C*(phi[i+1] - phi[i-1])



# Upwind de primeira ordem.
def upwind_primeira_ordem(i:int, C:float, phi:np.ndarray) -> float:
    """
    Avança phi no tempo, usando o método de Euler
    explícito e o método de upwind de primeira odem.

    Args:
        i (int): posição no espaço.
        C (float): número de Courant.
        phi (numpy array): vetor da propriedade
            que está sendo advectada.

    Returns:
        float: valor do novo phi no ponto i.
    """

    return phi[i] - C*(phi[i] - phi[i-1])


# Upwind de terceira ordem misto.
def upwind_terceira_ordem_misto(i, C, phi):
    return phi[i] - C*(2.0*phi[i+1] + 3.0*phi[i] - 6.0*phi[i-1] + phi[i-2])/6.0


# Método de Lax-Wendroff.
def lax_wendroff(i, C, phi):
    return phi[i] - 0.5*C*(phi[i+1] - phi[i-1]) + 0.5*C*C*(
        phi[i+1] - 2.0*phi[i] + phi[i-1])


# Upwind de segunda ordem.
def upwind_segunda_ordem(i, C, phi):
    return phi[i] - C*(phi[i] - phi[i-1]) - 0.5*C*(1.0 - C)*(phi[i] - 2.
 ↪0*phi[i-1] + phi[i-2])
```

[3]:
```
def configurar_grafico():

    fig, ax = plt.subplots()

    #ax.legend()
```

```
    ax.set_xlabel(r'$x$', fontsize=28, usetex=True)
    ax.set_ylabel(r'$\phi$', fontsize=28, usetex=True)

    plt.xticks(size=20, usetex=True)
    plt.yticks(size=20, usetex=True)

    minor_locator = AutoMinorLocator(5)
    ax.xaxis.set_minor_locator(minor_locator)
    minor_locator = AutoMinorLocator(5)
    ax.yaxis.set_minor_locator(minor_locator)

    ax.grid(color='gray')
    ax.grid(which='minor', color='gray', alpha=0.2)


def plot_phi(x, phi, color, linha = '-'):

    plt.plot(x, phi, color + linha, lw = 1.5)
```