UERJ - Universidade do Estado do Rio de Janeiro

IPRJ - Instituto Politécnico do Rio de Janeiro

Graduação em Engenharia de Computação

Métodos Numéricos para Equações Diferenciais II
Trabalho I
A Equação de Advecção

André Savioli Martins

Relatório apresentado ao curso de Engenharia de Computação, Universidade do Estado do Rio de Janeiro

Professor: Grazione de Souza Boy

Nova Friburgo

15/07/2018

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

EDP	Equação Diferencial Parcial
FTCS	Forward Time Centred Space
CFL	Courant–Friedrichs–Lewy

Sumário

1 Introdução

A Modelagem Computacional é a área da ciência que utiliza modelos matemáticos para a análise, solução e simulação de problemas científicos através de algoritmos e o emprego de computadores.

Quando se trata das leis naturais fundamentais, por exemplo, a Lei da Conservação de Massa, grande parte delas são governadas por Equações Diferenciais Parciais (EDP), e para certos problemas, dependendo das suas condições iniciais, de contorno ou, de ambas é díficil encontrar uma solução analítica.

Tendo isto em vista é possível encontrar uma aproximação do problema usando técnicas para aproximação de equações diferenciais parciais, onde nos dias atuais, encontraremos uma variedade de métodos à disposição, cada qual com seus pontos positivos e negativos levando em consideração o problema ao qual serão aplicados, tornando-se imprescindível a avaliação de cada um.

Em se tratando de métodos para resolução de problemas que envolvem equações diferenciais parciais podemos inicialmente lançar mão de discretizações a partir do método de elementos finitos usando Diferenças Finitas que é guiado por aproximação de derivadas ou do método de Volumes Finitos, que contempla as aproximações de integrais.

1.1 Objetivo

Este trabalho tem como objetivo comparar alguns dos métodos que podem ser empregados para a solução de problemas que envolvem EDPs, de forma mais específica, trataremos da EDP que governa o fenômeno de advecção unidimensional, que é dada por:

$$\frac{\partial c}{\partial t} + \overline{u}\frac{\partial c}{\partial x} = 0 \tag{1.1}$$

Onde c indica a variável dependente e $\overline{u}$ a velocidade de advecção, que para este trabalho é constante.

A condição inicial para este problema é dada por:

$$c(x,0) = e^{-200(x-0,3)^2} + s(x) \tag{1.2}$$

onde:

$$s(x) = \begin{cases} 1{,}0, & \text{se} \quad \text{a} \leq x \leq b \\ \\ 0{,}0, & \text{caso contrário.} \end{cases} \tag{1.3}$$

e a e b delimitam uma região no interior do domínio de comprimento L.

Sobre este problema serão aplicados quatro métodos, o primeiro, o método de *Lax-Friedrichs*, o segundo, um método com abordagem *UpWind* para os fluxos, que neste trabalho chamaremos apenas de método *UpWind*, o terceiro, o método *Lax-Wendroff* e o quarto e último, o método *LeapFrog*.

2 Metodologia

Neste capítulo serão descritos os procedimentos utilizados na obtenção dos resultados do Capítulo 3. Serão descritos os cuidados que foram tomados na implementação de cada método, discutida a necessidade de uma forma para comparar o desempenho das soluções numéricas em relação ao problema de advecção unidimensional.

Ainda neste capítulo avaliaremos as principais características dos métodos que serão analisados. Na subseção 2.1.2.1 sobre o método Lax-Friedrichs, na subseção 2.1.2.2 o método UpWind, na subseção 2.1.2.3 o método Lax-Wendroff e na subseção 2.1.2.4 o método LeapFrog.

2.1 Fundamentos Teóricos

Nesta seção serão abordados os fundamentos teóricos necessários para avaliação dos métodos Lax-Friedrichs, UpWind, Lax-Wendroff e LeapFrog em relação a solução analítica e as soluções numéricas.

2.1.1 Solução Analítica

A equação de advecção possui uma solução exata, o que é algo muito importante para avaliação do desempenho dos métodos utilizados neste trabalho. É possível demonstrar que a solução para a EDP de advecção unidimensional deste problema é:

$$c(x,t) = e^{-200(x-\overline{u}t-0,3)^2} + s(x-\overline{u}t) \tag{2.1}$$

No entanto neste trabalho apenas usaremos esse resultado e o aplicaremos com o intuito ter uma referência precisa para comparação no Capítulo 3.

2.1.2 Solução Numérica

Para a implementação dos métodos de solução numérica foram utilizadas: a linguagem de programação *Python*[1] em sua versão 3.5, a biblioteca de computação científica *NumPy*[2], e, para a geração de gráficos, a biblioteca *Matplotlib*[3]..

Na implementação dos métodos de solução numérica alguns cuidados foram tomados no que diz respeito ao valor das fronteiras do domínio. Adotar uma escolha equivocada

[1] <https://www.python.org>
[2] <http://www.numpy.org>
[3] <https://matplotlib.org>

poderia levar o método a uma solução não esperada para o problema, sendo assim, nos considerados *ghost values* definimos o uso da solução analítica como escolha de valores para tais pontos de cálculo. Uma outra possibilidade seria o uso de valores vizinhos das extremidades à esquerda e à direita do domínio computacional, mas a perturbação da solução era um pouco maior em testes prévios então foi desconsiderada.

De imediato definiremos como C a seguinte expressão

$$C = \overline{u}\frac{\Delta t}{\Delta x} \tag{2.2}$$

onde C é o número de *Courant* ou condição de *Courant–Friedrichs–Lewy* (CFL), que será importante para as discussão e conclusão, no Capítulo 4 e Capítulo 5 respectivamente.

Essa expressão será encontrada pelas próximas subseções quando forem abordadas algumas características dos métodos UpWind, Lax-Wendroff e LeapFrog. A implementação dos 4 algoritmos de solução numérica encontram-se nos anexos deste trabalho.

2.1.2.1 O Método Lax-Friedrichs

O método Lax-Friedrichs é dado pela seguinte formulação:

$$Q_i^{n+1} = \frac{Q_{i-1}^n + Q_{i+1}^n}{2} - \frac{\Delta t}{2\Delta x}\{f(Q_{i+1}^n) - f(Q_{i-1}^n)\} \tag{2.3}$$

Este é um método de primeira ordem bastante similiar ao esquema conhecido como Forward Time Centred Space (FTCS),

$$Q_i^{n+1} = Q_i^n - \frac{\overline{u}\Delta t}{2\Delta x}(Q_{i+1}^n - Q_{i-1}^n) \tag{2.4}$$

no entanto apesar da similaridade com o método Lax-Friedrichs, suas performances segundo Causon e Mingham (2010) são bastante diferentes.

Em relação os fluxos, substituímos conforme o caráter advectivo da equação, e portanto a formulação torna-se:

$$Q_i^{n+1} = \frac{Q_{i-1}^n + Q_{i+1}^n}{2} - \frac{C}{2}(Q_{i+1}^n - Q_{i-1}^n) \tag{2.5}$$

Conforme foi dito na subseção 2.1.2, ao implementar esse método teve-se o cuidado de observar os *ghost values* existentes nas extremidades do domínio computacional.

2.1.2.2 O Método UpWind

O método UpWind é um método de primeira ordem dado pela seguinte formulação:

$$Q_i^{n+1} = Q_i^n - C(Q_i^n - Q_{i-1}^n) \tag{2.6}$$

onde C é o número de *Courant* definido pela Equação 2.2.

Este método aproveita o fato de que em problemas hiperbólicos a informação se propaga como ondas em direções características, portanto leva em consideração a velocidade e a direção de propagação dessas ondas (SOUTO, 2017).

Ao contrário do método Lax-Friedrichs, só existem *ghost values* à esquerda do domínio computacional.

2.1.2.3 O Método Lax-Wendroff

O método Lax-Wendroff diferente dos métodos anteriores é um método de segunda ordem dado pela seguinte formulação:

$$Q_i^{n+1} = Q_i^n - \frac{C}{2}\{(Q_{i+1}^n - Q_{i-1}^n) - C(Q_{i+1}^n - 2Q_{i-1}^n + Q_{i-1}^n)\} \tag{2.7}$$

Conforme foi dito na subseção 2.1.2, ao implementar esse método teve-se o cuidado de observar os *ghost values* existentes nas extremidades do domínio computacional.

2.1.2.4 O Método LeapFrog

O método LeapFrog é também, um método de segunda ordem, ele é dado pela seguinte formulação:

$$Q_i^{n+1} = Q_i^{n-1} - C(Q_{i+1}^n - Q_{i-1}^n) \tag{2.8}$$

Este método exige que algumas escolhas sejam tomadas para a sua implementação. A primeira é em relação aos *ghost values* existentes nas extremidades do domínio computacional. A segunda é em relação ao seu termo Q_i^{n-1}, que é necessário para iniciar o método e seguir com as próximas iterações, visto que apenas a condição inicial do problema não satisfaz o método.

A solução neste caso é realizar a primeira iteração com um método que não depende do termo Q_i^{n-1} e só partir de então iniciar as iterações com o método LeapFrog, quando haverá finalmente o termo Q_i^{n-1}.

Neste trabalho poderíamos escolher entre os outros 3 demais métodos para serem utilizados como a primeira iteração necessária ao método LeapFrog. Porém dentre eles escolhemos o método Lax-Wendroff pois é um método de segunda ordem e portanto possui um erro menor.

3 Resultados

Neste Capítulo apresentamos graficamente os resultados obtidos através da execução dos métodos implementados computacionalmente. Ao todo foram realizados 16 experimentos, 4 para cada método variando os parâmetros Δt, Δx, u e consequentemente o número de *Courant*, conforme a Tabela 1.

Núm. do Experimento	**L**	**a**	**b**	**Δt**	**Δx**	**ū**	**Núm. de Courant**
1	3	0.556	0.878	0.01	0.02	0.4	0.2
2	3	0.556	0.878	0.001	0.001	0.6	0.6
3	3	0.556	0.878	0.0005	0.0005	0.8	0.8
4	3	0.556	0.878	0.01	0.01	1.01	1.01

Tabela 1 – Definições de Variáveis para os Experimentos sobre cada Método Numérico

Nos gráficos deste capítulo a solução analítica é apresentada em linha tracejada vermelha e a solução numérica é apresentada linha contínua preta.

Abaixo na Figura 1 temos o gráfico da condição inicial em t = 0, que é igual para todos os experimentos.

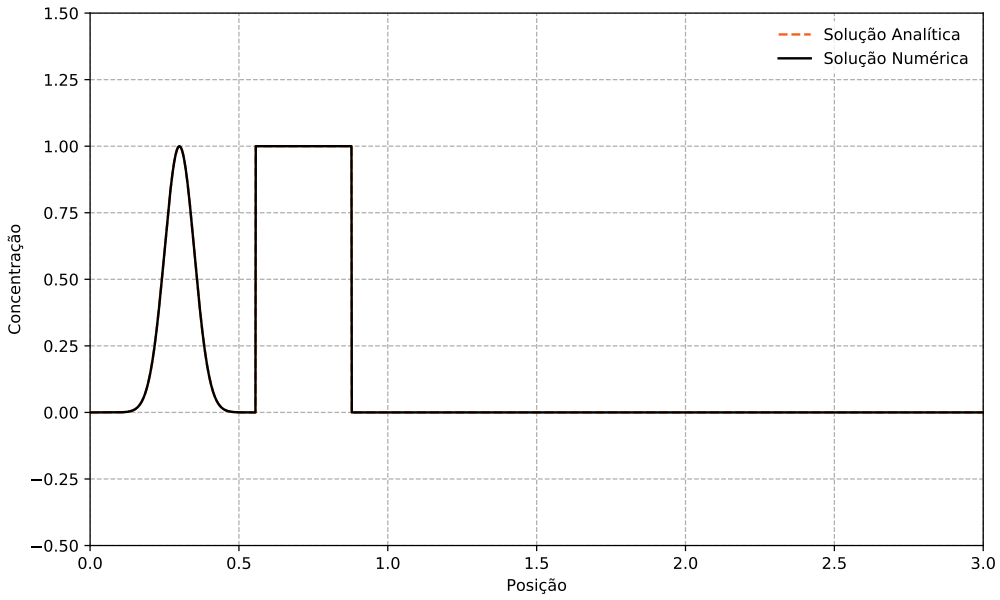

Figura 1 – Condição Inicial para o problema em questão em t = 0 e perfil da solução analítica

3.1 Resultados para o Método Lax-Friedrichs

Na subseção 3.1.1, subseção 3.1.2, subseção 3.1.3 e subseção 3.1.4 são apresentados os resultados para os 4 experimentos realizados tendo em vista os dados da Tabela 1.

3.1.1 Primeiro Experimento com o Método Lax-Friedrichs

Abaixo apresentamos os resultados do primeiro experimento com o método Lax-Friedrichs.

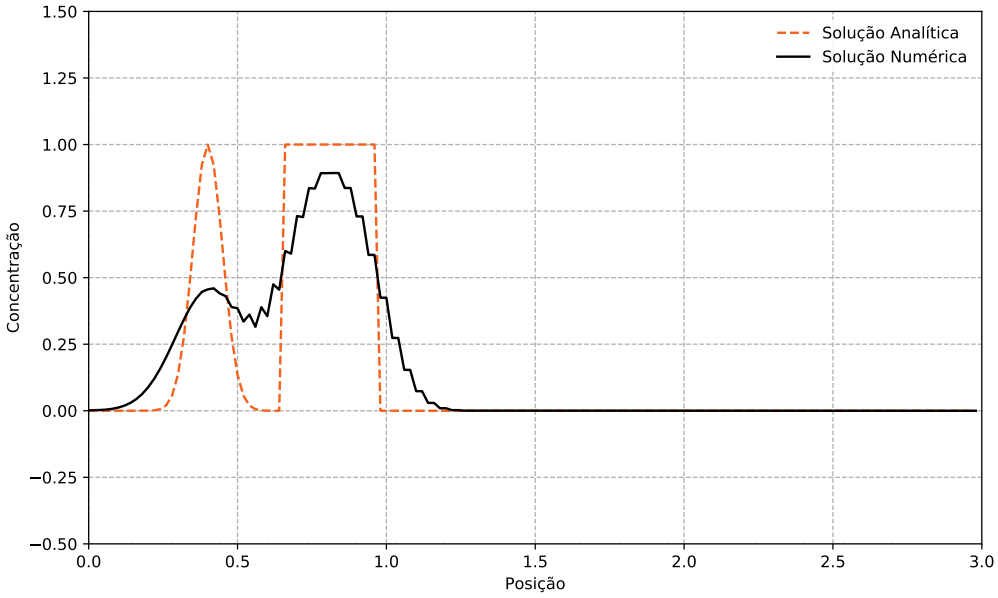

Figura 2 – Lax-Friedrichs - Primeiro Experimento em t = 0,25

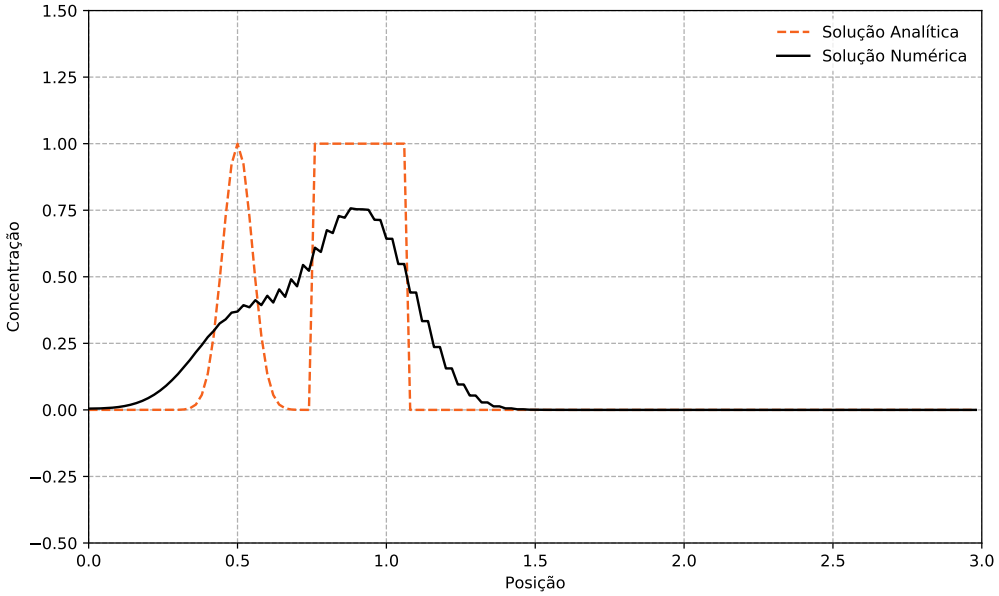

Figura 3 – Lax-Friedrichs - Primeiro Experimento em t = 0,5

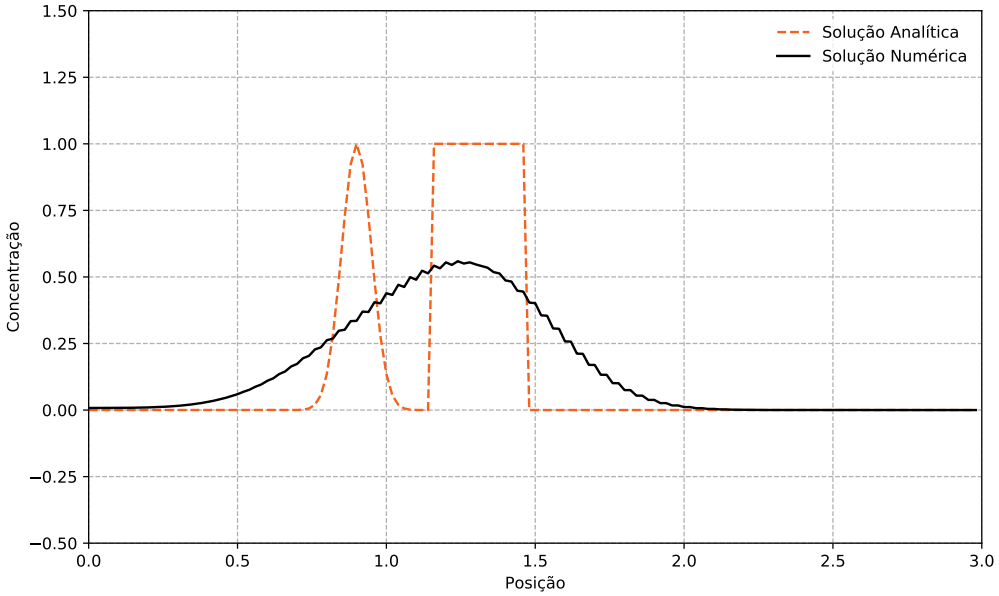

Figura 4 – Lax-Friedrichs - Primeiro Experimento em t = 1,5

3.1.2 Segundo Experimento com o Método Lax-Friedrichs

Abaixo apresentamos os resultados do segundo experimento com o método Lax-Friedrichs.

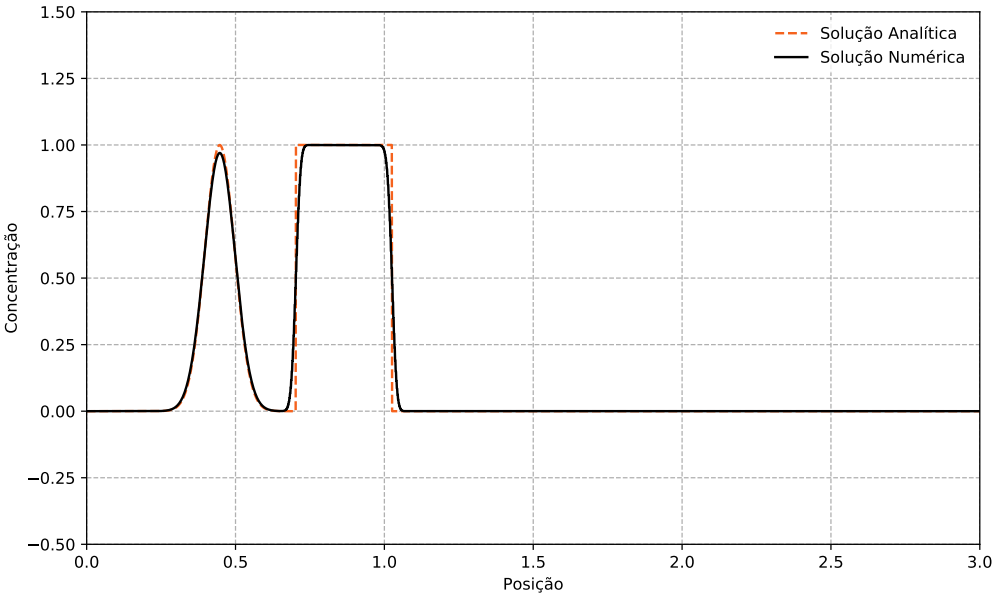

Figura 5 – Lax-Friedrichs - Segundo Experimento em t = 0,25

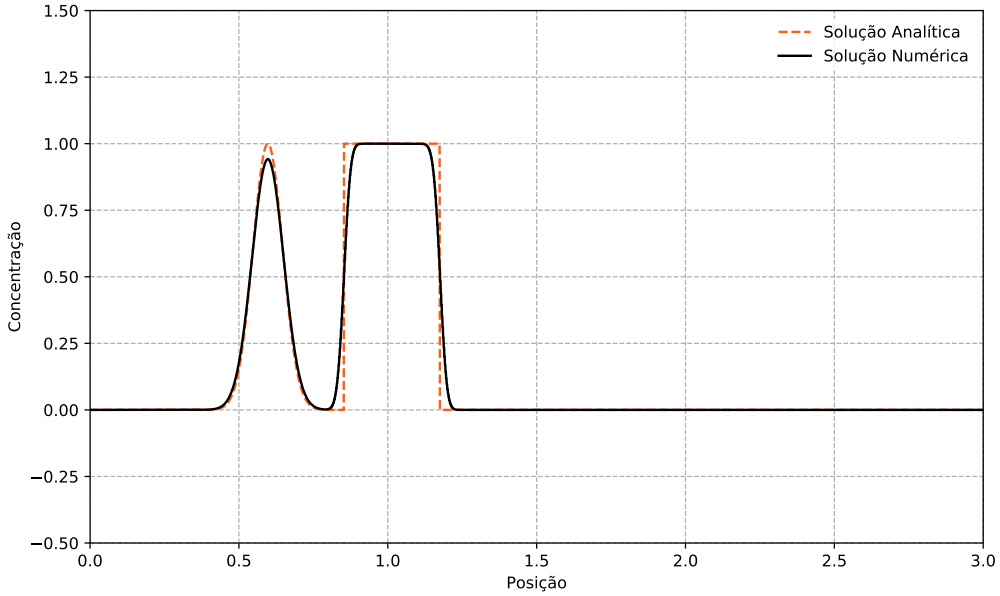

Figura 6 – Lax-Friedrichs - Segundo Experimento em t = 0,5

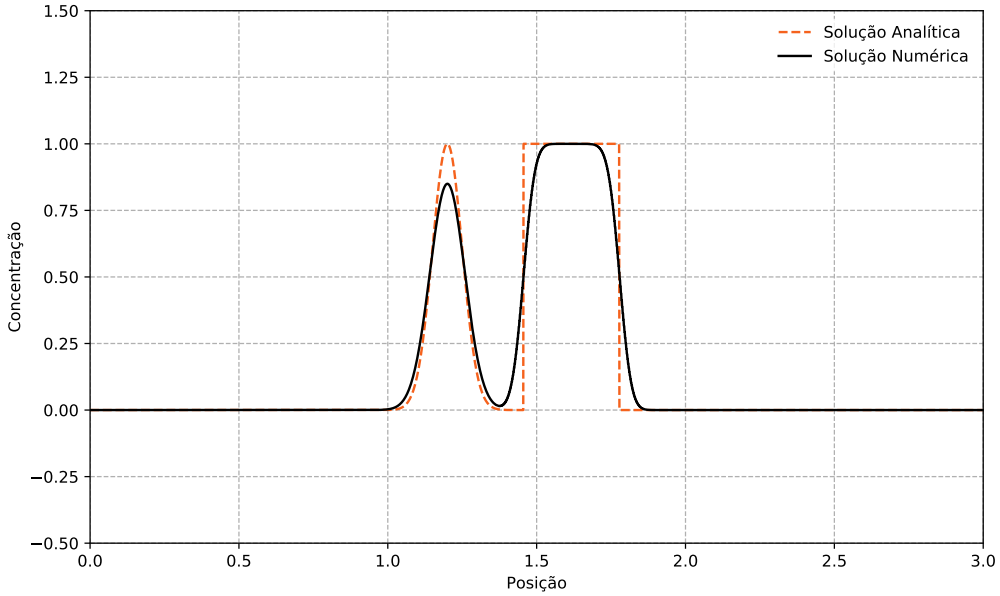

Figura 7 – Lax-Friedrichs - Segundo Experimento em t = 1,5

3.1.3 Terceiro Experimento com o Método Lax-Friedrichs

Abaixo apresentamos os resultados do terceiro experimento com o método Lax-Friedrichs.

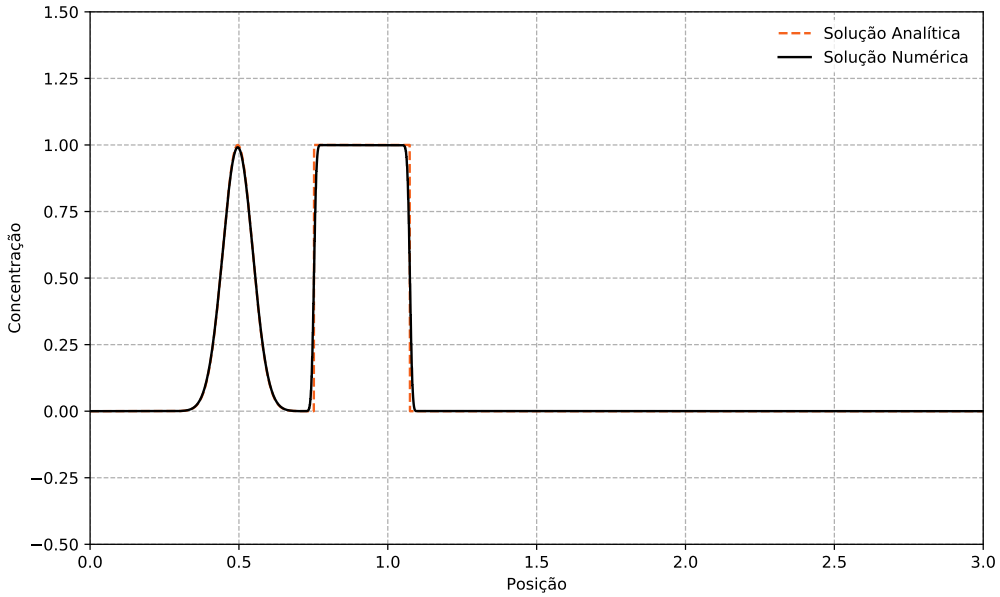

Figura 8 – Lax-Friedrichs - Terceiro Experimento em t = 0,25

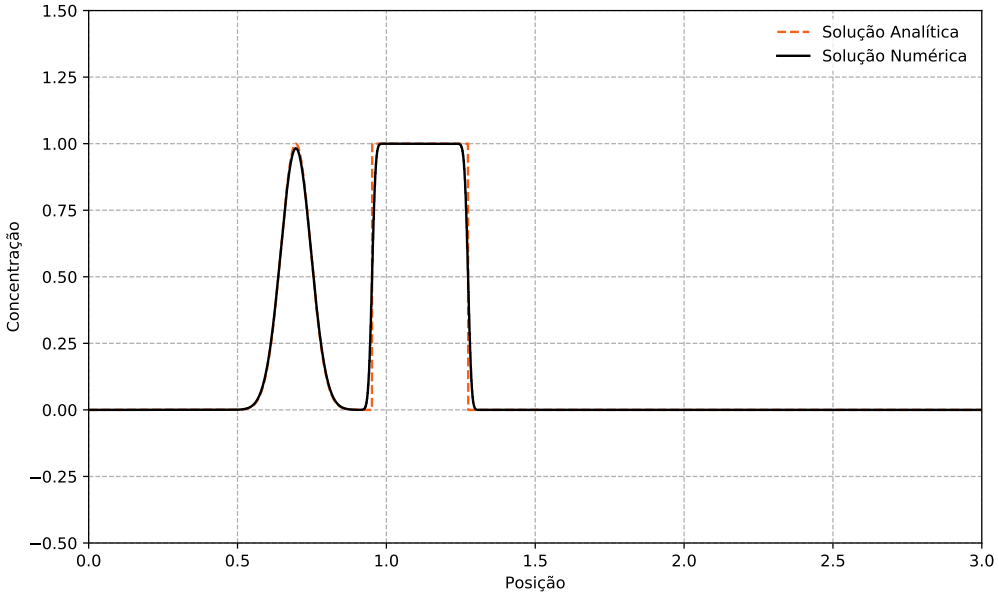

Figura 9 – Lax-Friedrichs - Terceiro Experimento em t = 0,5

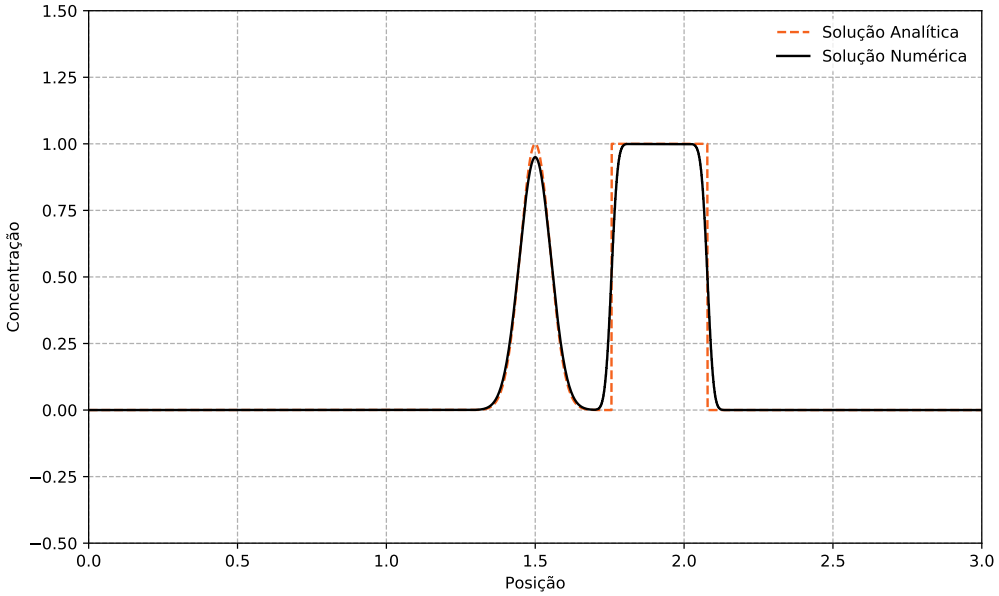

Figura 10 – Lax-Friedrichs - Terceiro Experimento em t = 1,5

3.1.4 Quarto Experimento com o Método Lax-Friedrichs

Abaixo apresentamos os resultados do quarto experimento com o método Lax-Friedrichs.

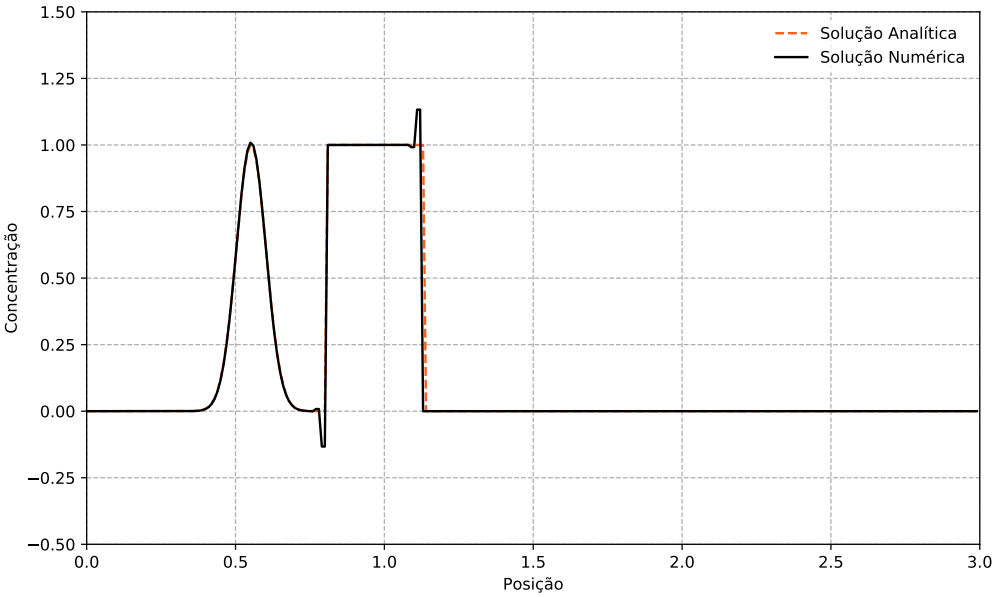

Figura 11 – Lax-Friedrichs - Quarto Experimento em t = 0,25

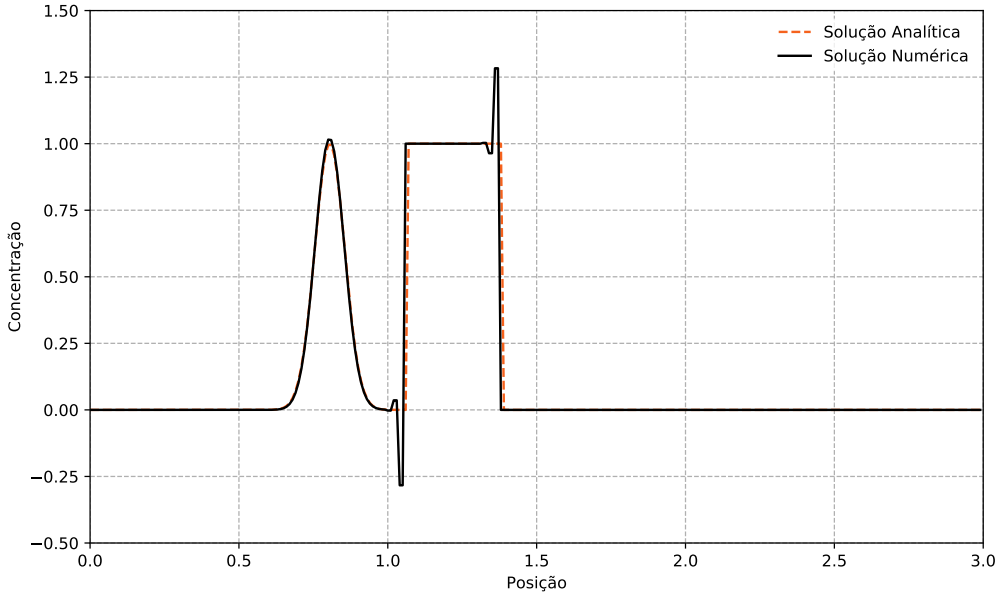

Figura 12 – Lax-Friedrichs - Quarto Experimento em t = 0,5

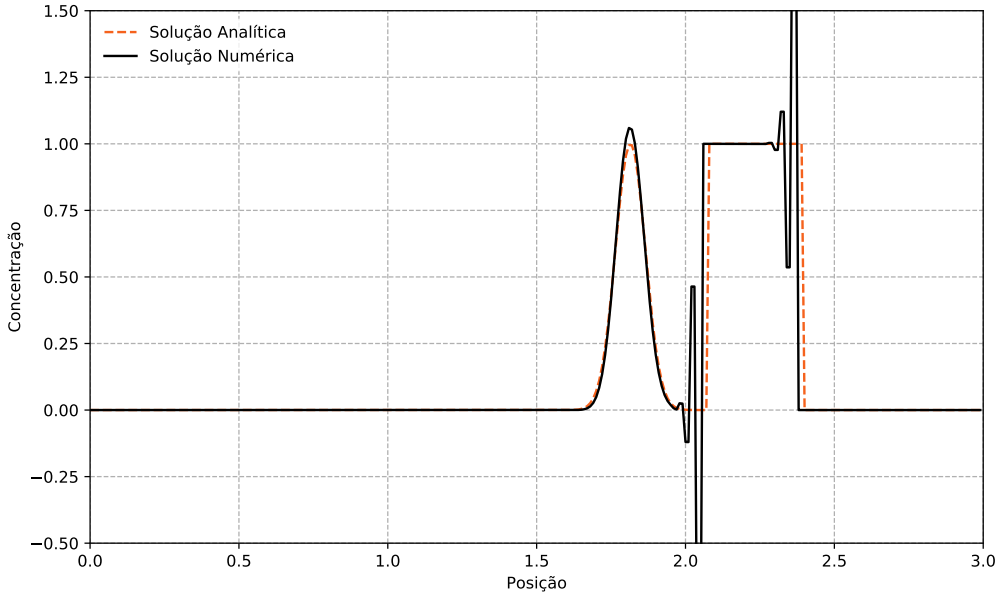

Figura 13 – Lax-Friedrichs - Quarto Experimento em t = 1,5

3.2 Resultados para o Método UpWind

Na subseção 3.2.1, subseção 3.1.2, subseção 3.2.3 e subseção 3.2.4 são apresentados os resultados para os 4 experimentos realizados tendo em vista os dados da Tabela 1.

3.2.1 Primeiro Experimento com o Método UpWind

Abaixo apresentamos os resultados do primeiro experimento com o método UpWind.

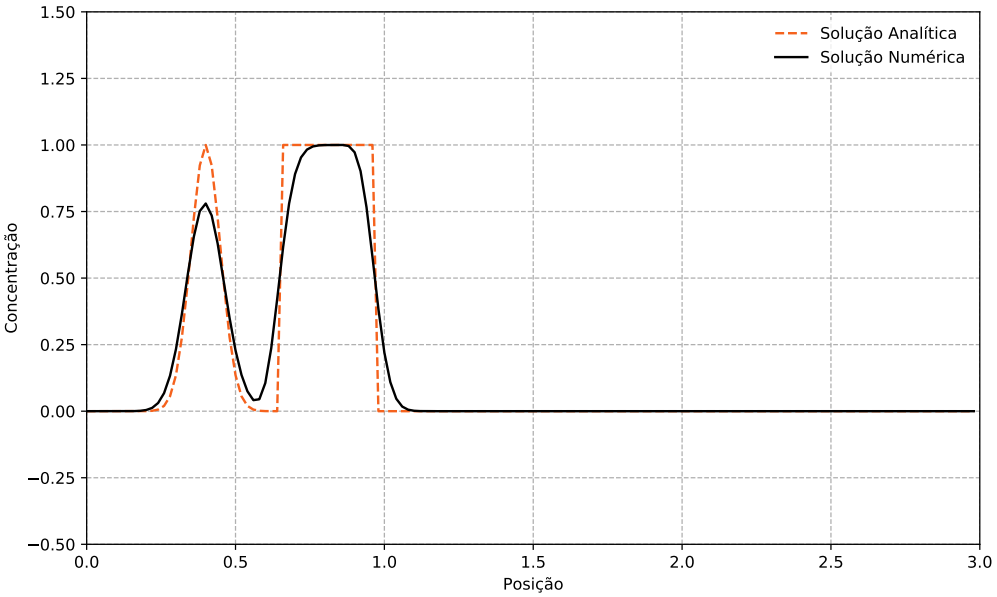

Figura 14 – UpWind - Primeiro Experimento em t = 0,25

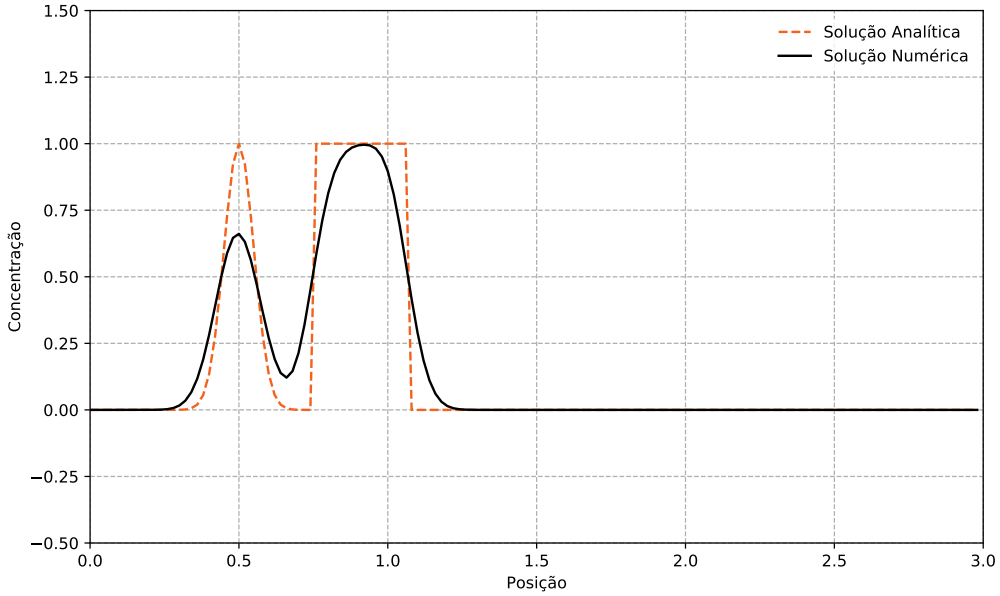

Figura 15 – UpWind - Primeiro Experimento em t = 0,5

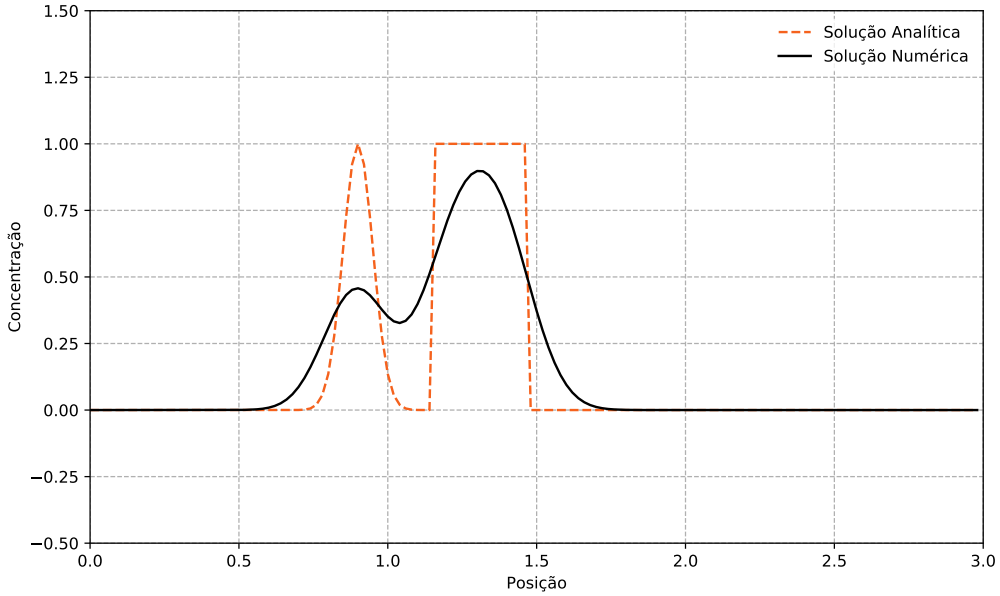

Figura 16 – UpWind - Primeiro Experimento em t = 1,5

3.2.2 Segundo Experimento com o Método UpWind

Abaixo apresentamos os resultados do segundo experimento com o método UpWind.

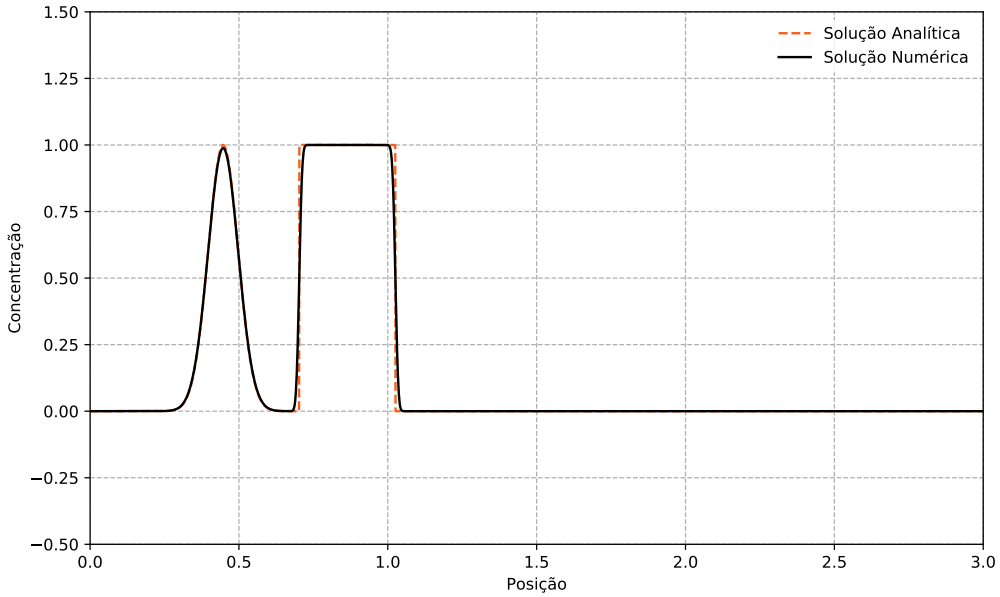

Figura 17 – UpWind - Segundo Experimento em t = 0,25

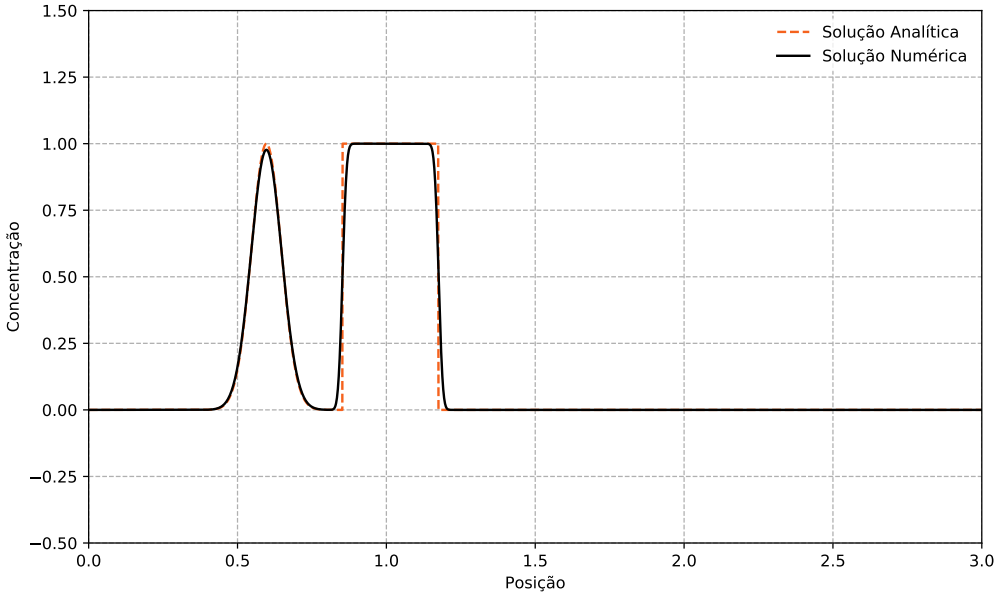

Figura 18 – UpWind - Segundo Experimento em t = 0,5

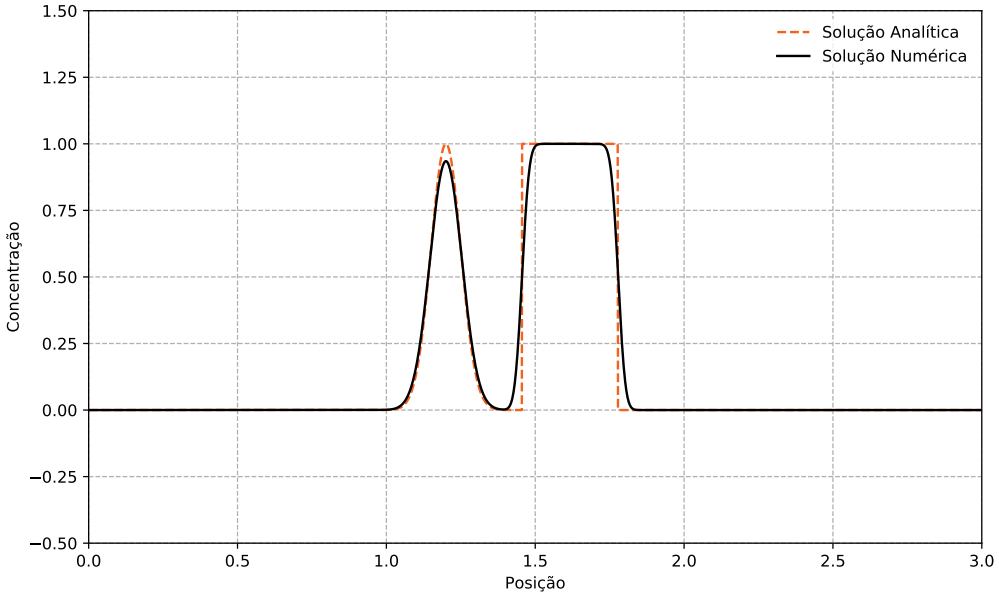

Figura 19 – UpWind - Segundo Experimento em t = 1,5

3.2.3 Terceiro Experimento com o Método UpWind

Abaixo apresentamos os resultados do terceiro experimento com o método UpWind.

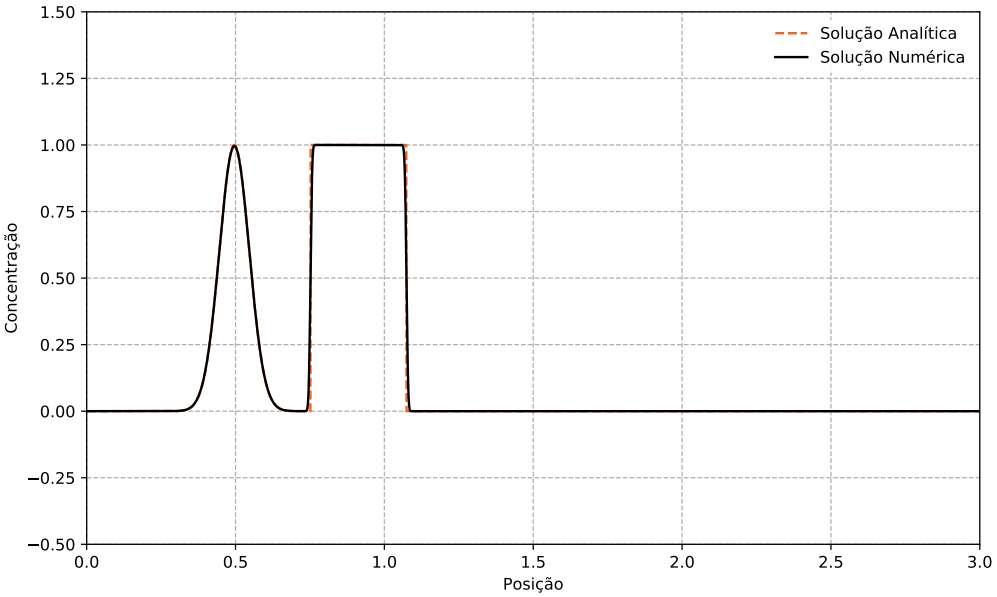

Figura 20 – UpWind - Terceiro Experimento em t = 0,25

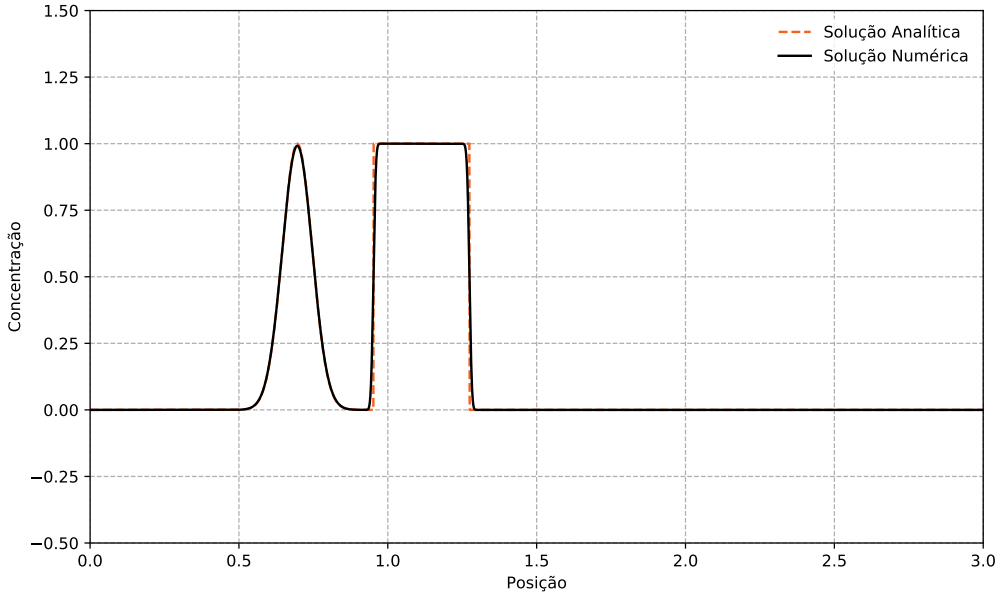

Figura 21 – UpWind - Terceiro Experimento em t = 0,5

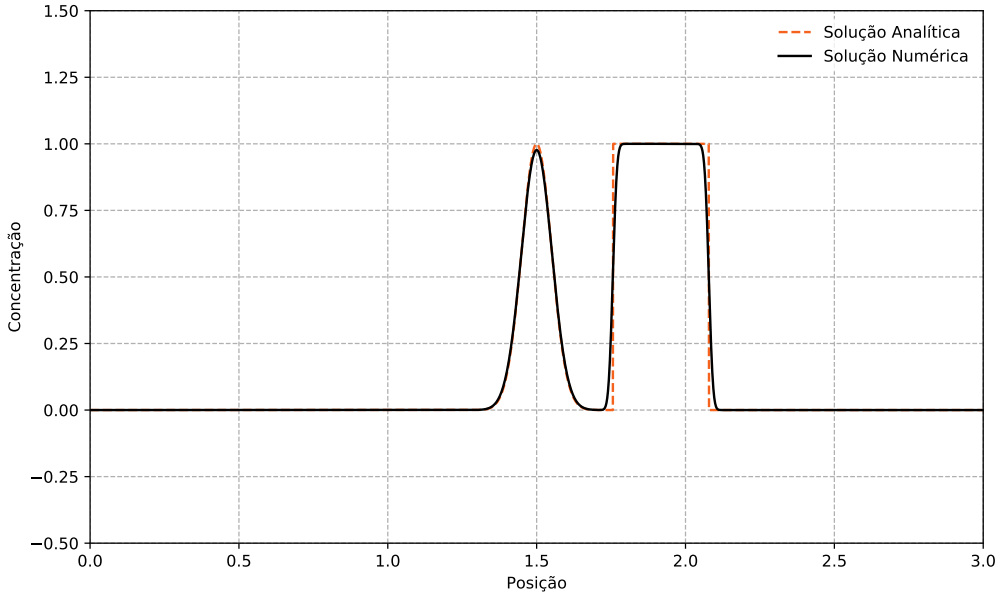

Figura 22 – UpWind - Terceiro Experimento em t = 1,5

3.2.4 Quarto Experimento com o Método UpWind

Abaixo apresentamos os resultados do quarto experimento com o método UpWind.

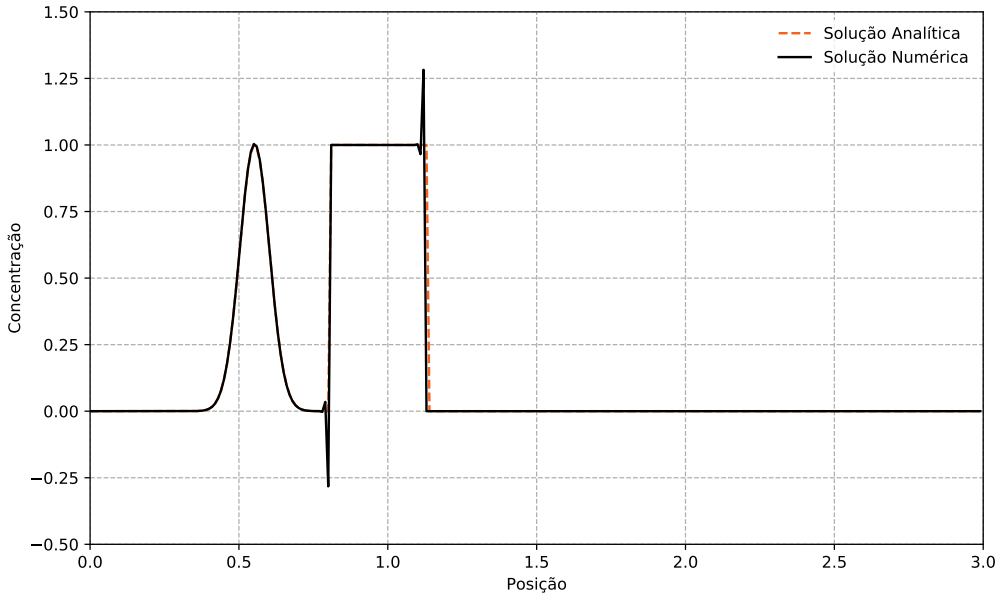

Figura 23 – UpWind - Quarto Experimento em t = 0,25

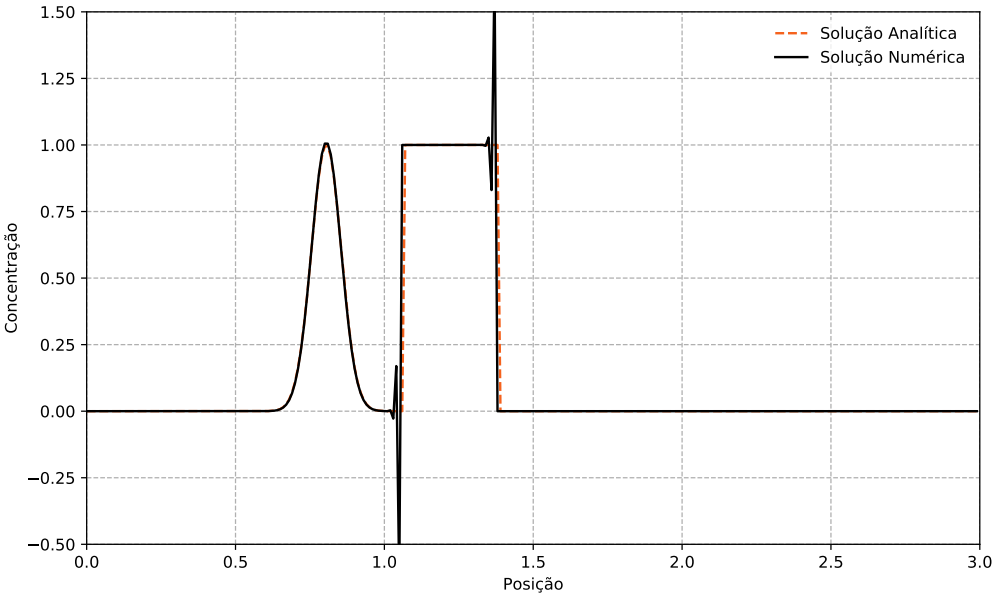

Figura 24 – UpWind - Quarto Experimento em t = 0,5

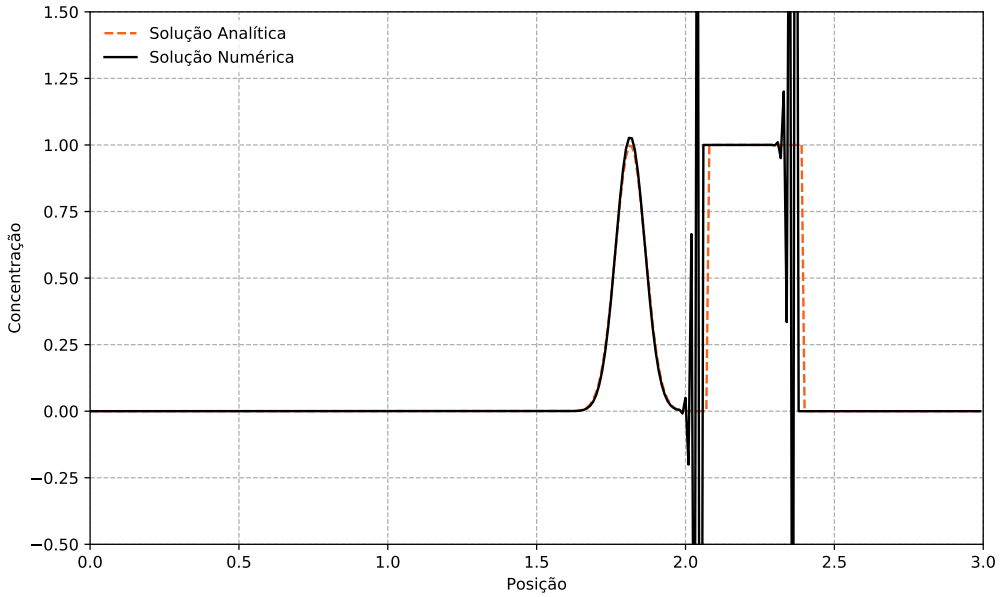

Figura 25 – UpWind - Quarto Experimento em t = 1,5

3.3 Resultados para o Método Lax-Wendroff

Na subseção 3.3.1, subseção 3.3.2, subseção 3.3.3 e subseção 3.3.4 são apresentados os resultados para os 4 experimentos realizados tendo em vista os dados da Tabela 1.

3.3.1 Primeiro Experimento com o Método Lax-Wendroff

Abaixo apresentamos os resultados do primeiro experimento com o método Lax-Wendroff.

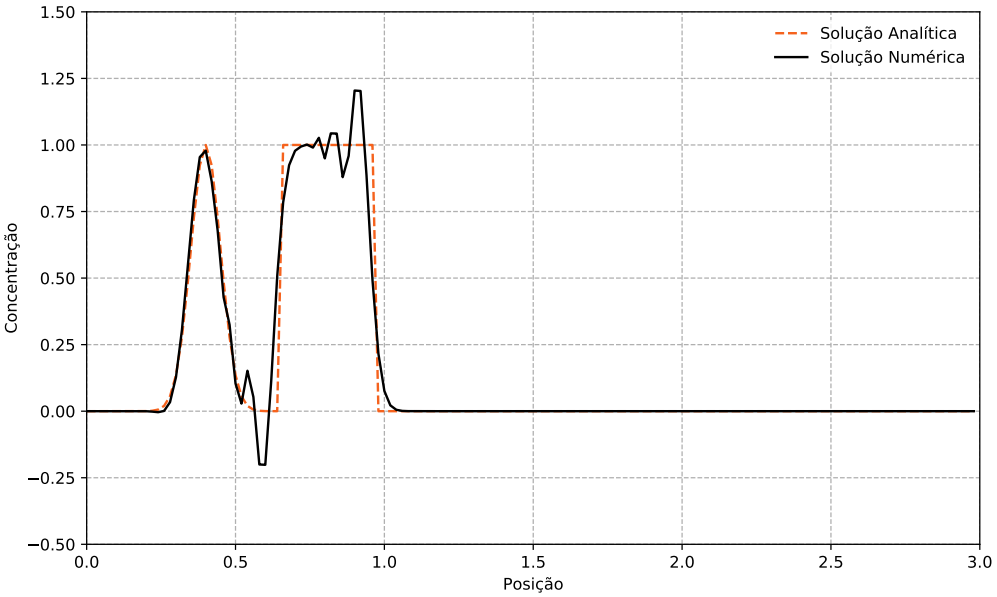

Figura 26 – Lax-Wendroff - Primeiro Experimento em t = 0,25

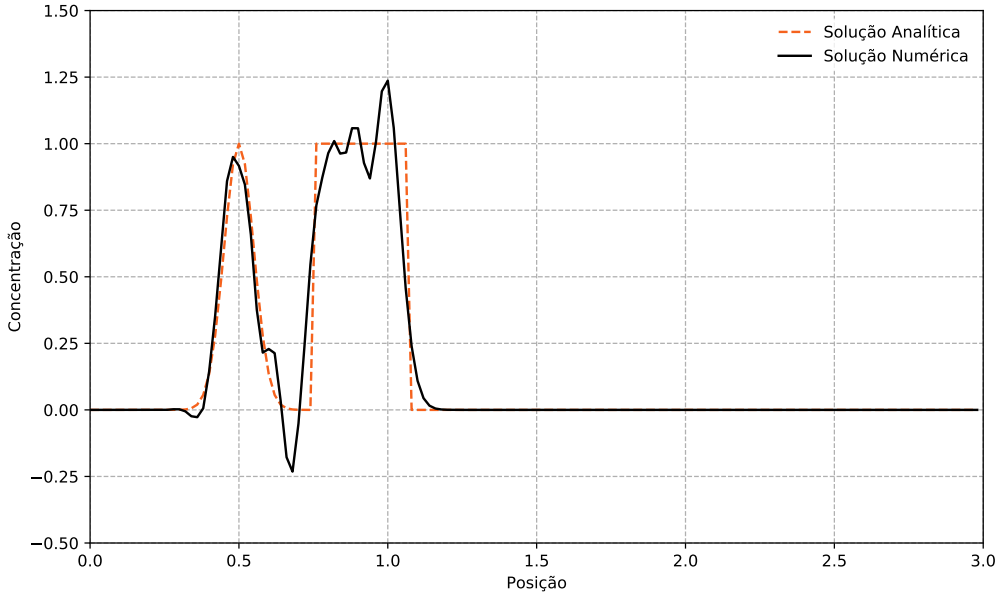

Figura 27 – Lax-Wendroff - Primeiro Experimento em t = 0,5

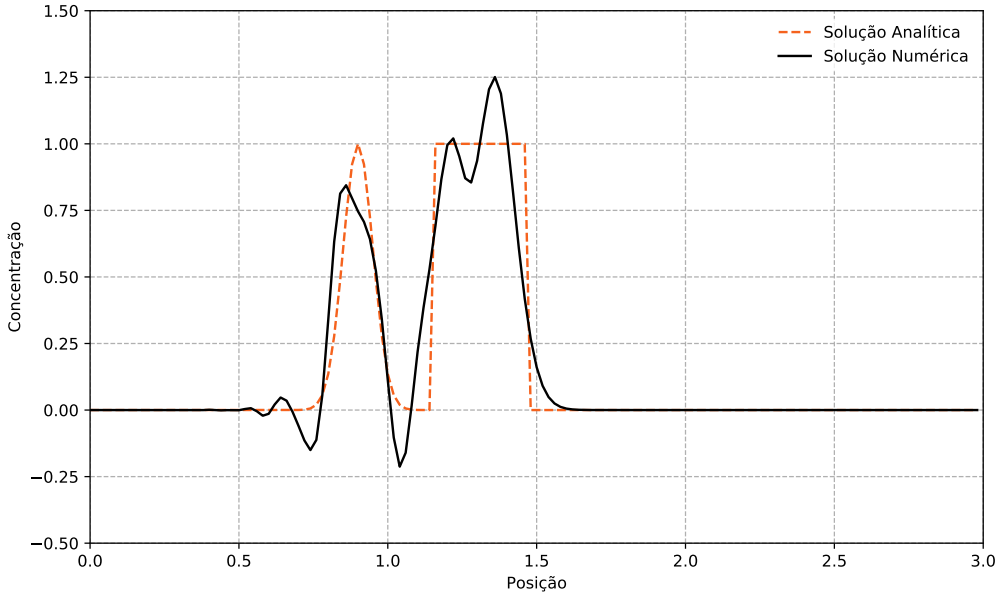

Figura 28 – Lax-Wendroff - Primeiro Experimento em t = 1,5

3.3.2 Segundo Experimento com o Método Lax-Wendroff

Abaixo apresentamos os resultados do segundo experimento com o método Lax-Wendroff.

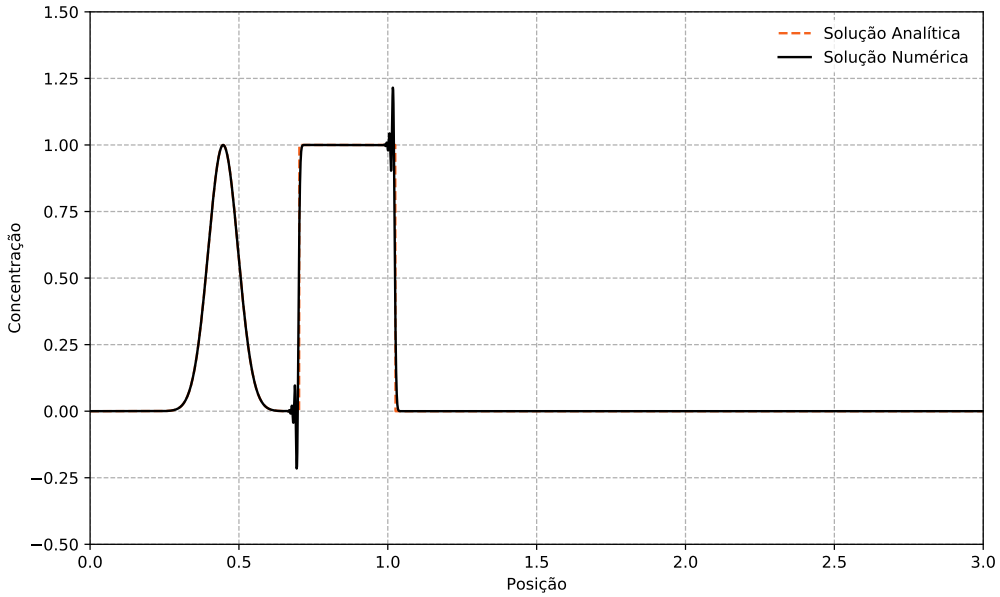

Figura 29 – Lax-Wendroff - Segundo Experimento em t = 0,25

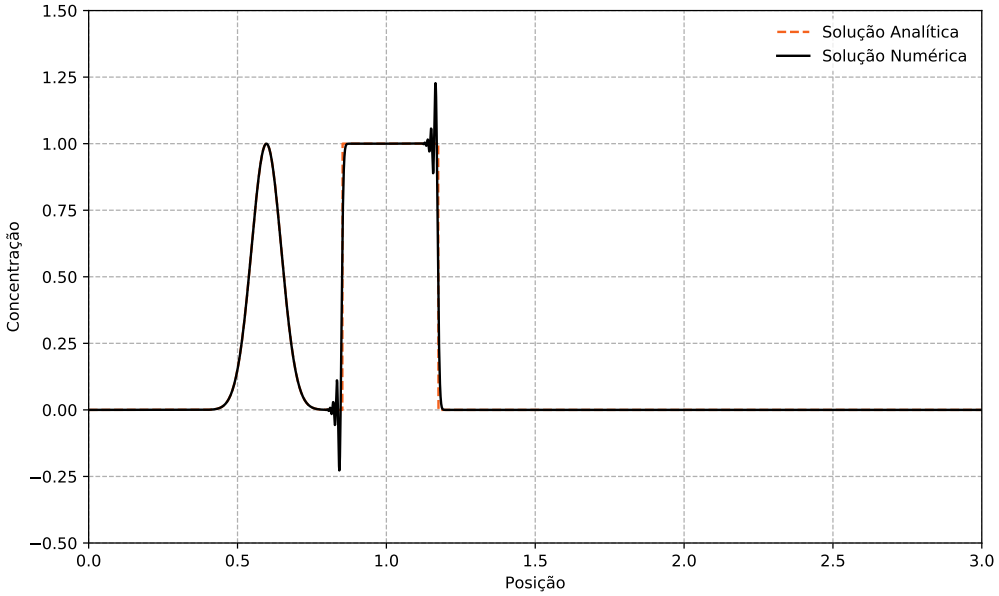

Figura 30 – Lax-Wendroff - Segundo Experimento em t = 0,5

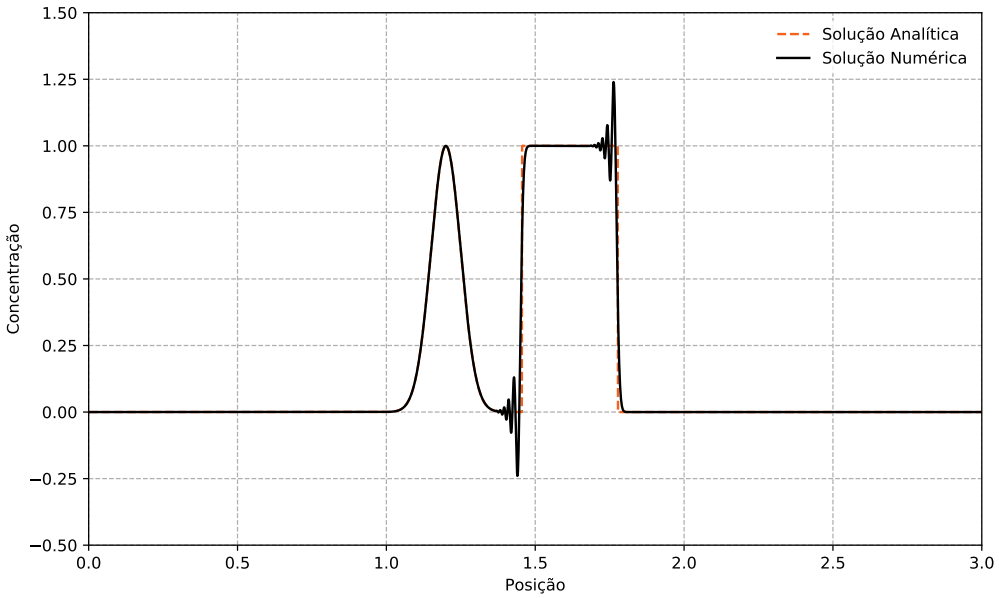

Figura 31 – Lax-Wendroff - Segundo Experimento em t = 1,5

3.3.3 Terceiro Experimento com Método Lax-Wendroff

Abaixo apresentamos os resultados do terceiro experimento com o método Lax-Wendroff.

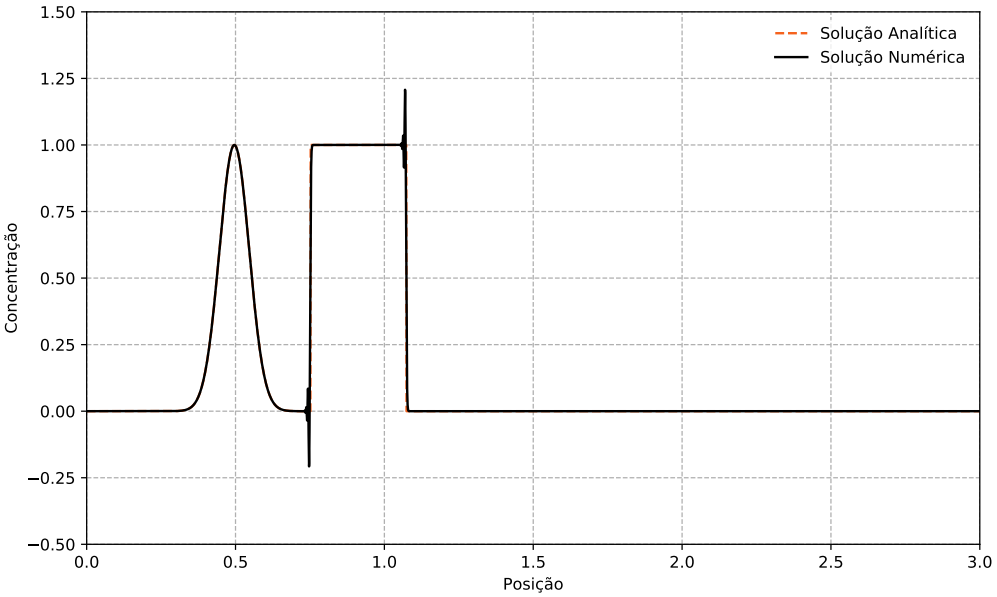

Figura 32 – Lax-Wendroff - Terceiro Experimento em t = 0,25

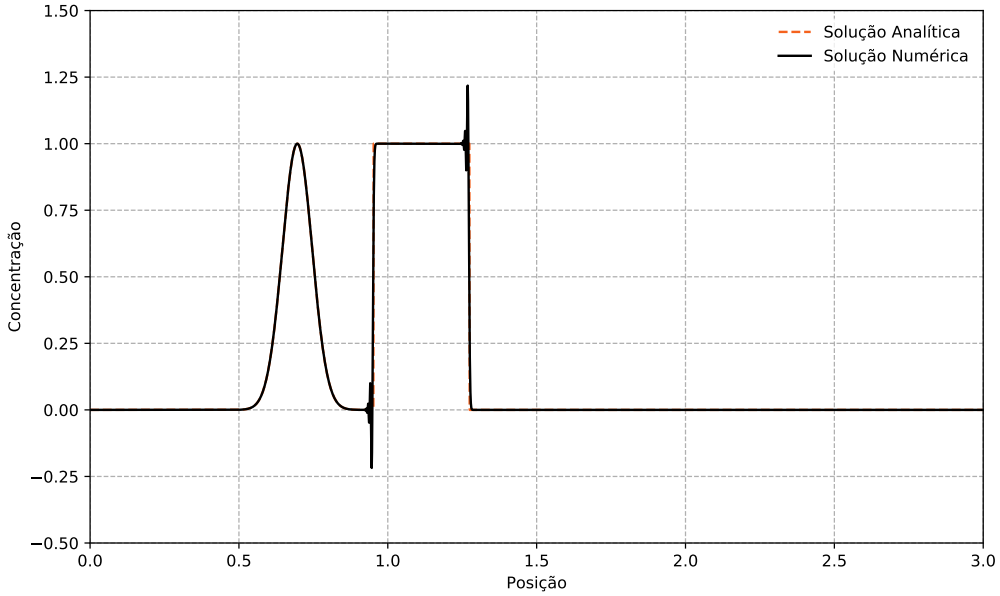

Figura 33 – Lax-Wendroff - Terceiro Experimento em t = 0,5

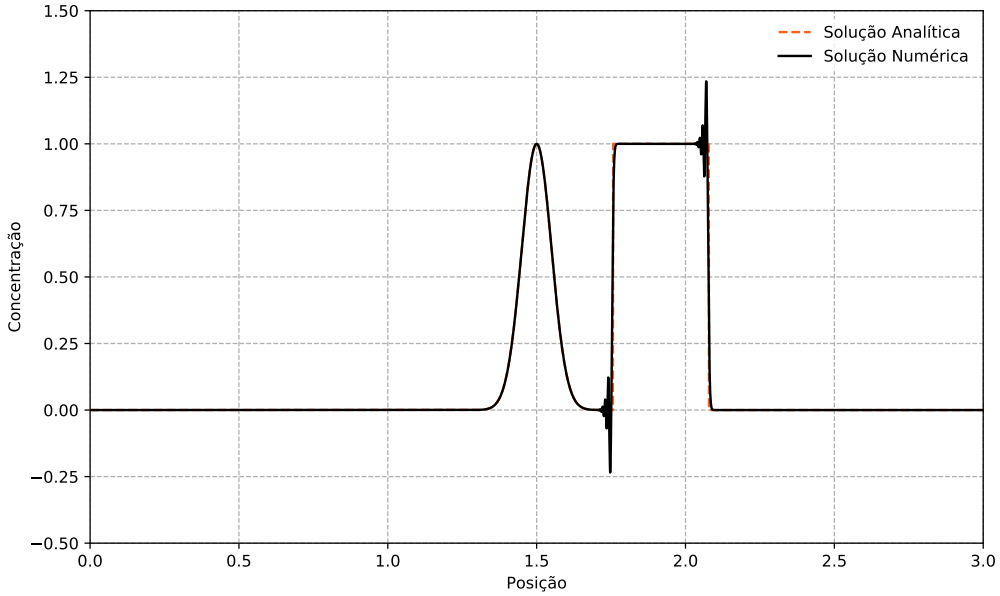

Figura 34 – Lax-Wendroff - Terceiro Experimento em t = 1,5

3.3.4 Quarto Experimento com o Método Lax-Wendroff

Abaixo apresentamos os resultados do quarto experimento com o método Lax-Wendroff.

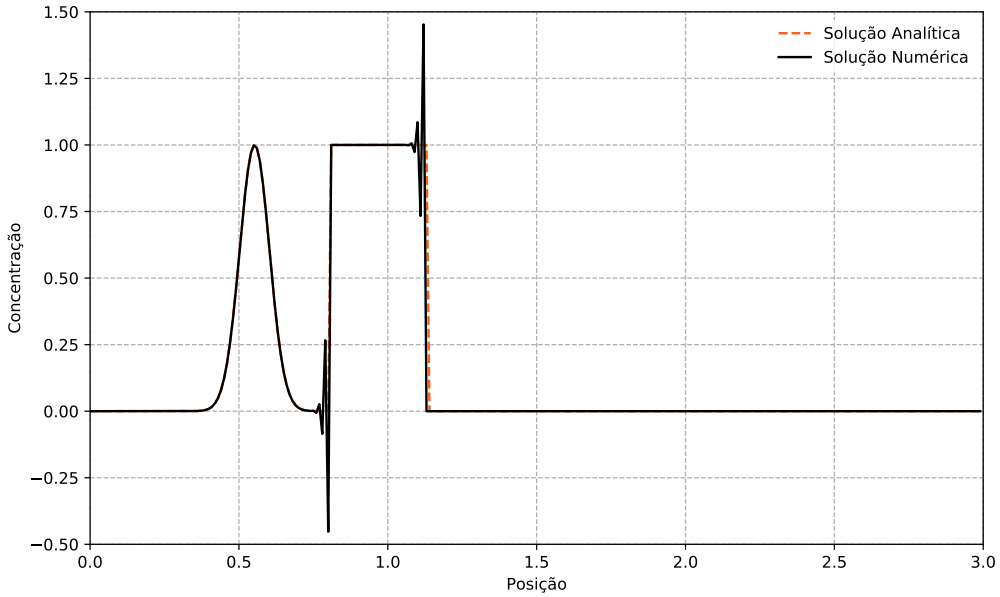

Figura 35 – Lax-Wendroff - Quarto Experimento em t = 0,25

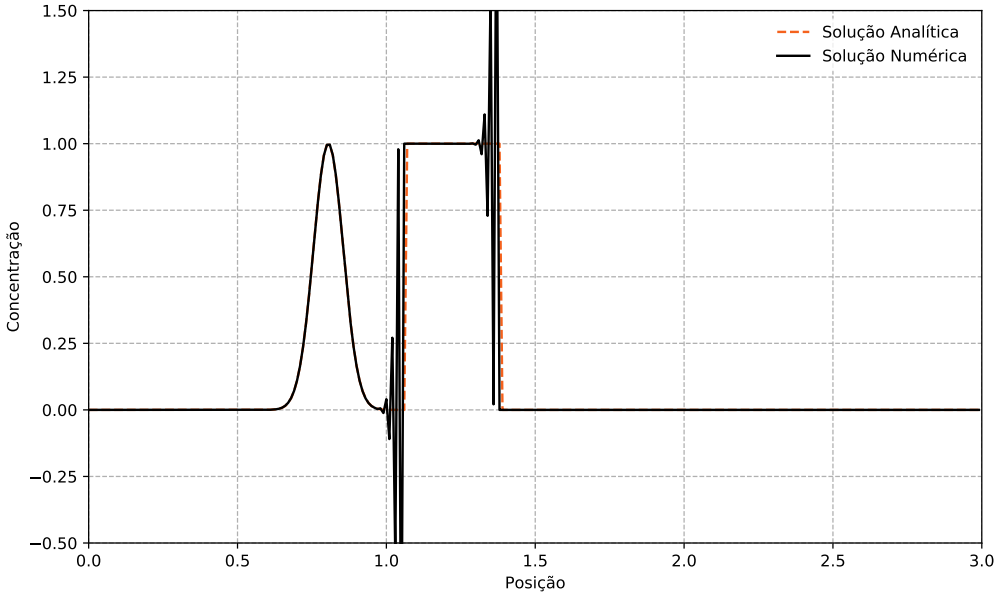

Figura 36 – Lax-Wendroff - Quarto Experimento em t = 0,5

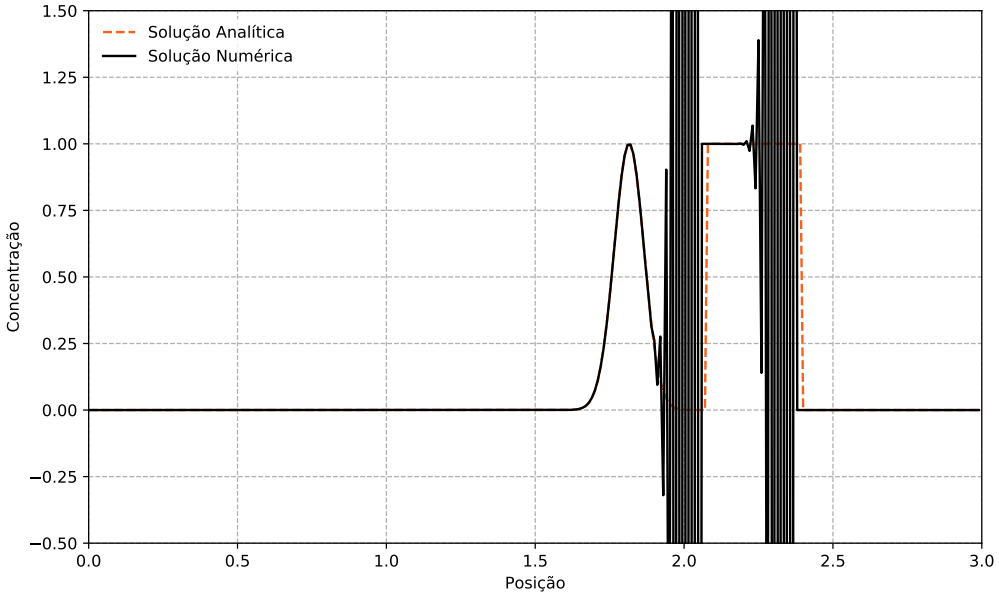

Figura 37 – Lax-Wendroff - Quarto Experimento em t = 1,5

3.4 Resultados para o Método LeapFrog

Na subseção 3.4.1, subseção 3.4.2, subseção 3.4.3 e subseção 3.4.4 são apresentados os resultados para os 4 experimentos realizados tendo em vista os dados da Tabela 1.

3.4.1 Primeiro Experimento com o Método LeapFrog

Abaixo apresentamos os resultados do primeiro experimento com o método Leap-Frog.

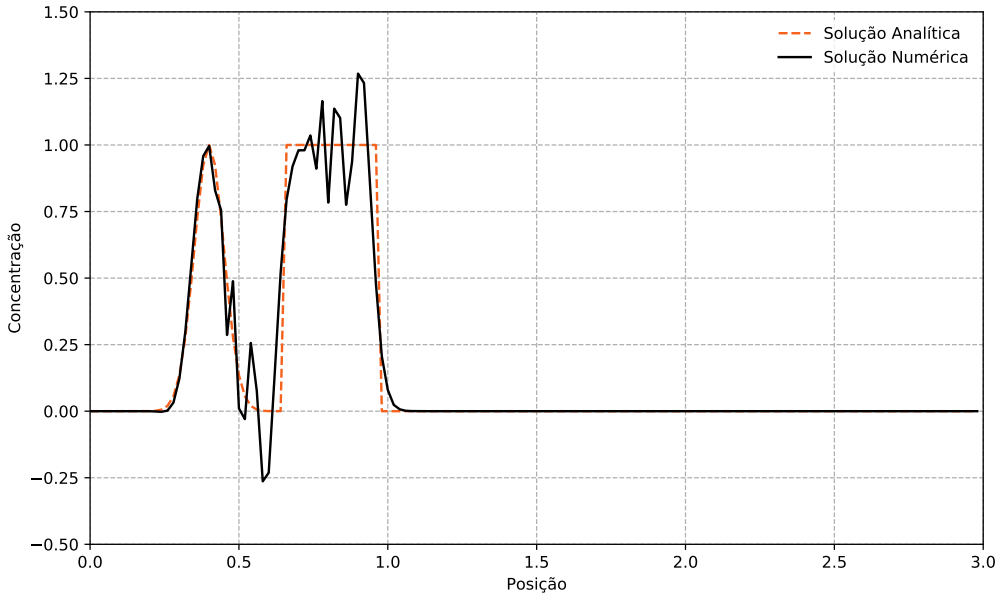

Figura 38 – LeapFrog - Primeiro Experimento em t = 0,25

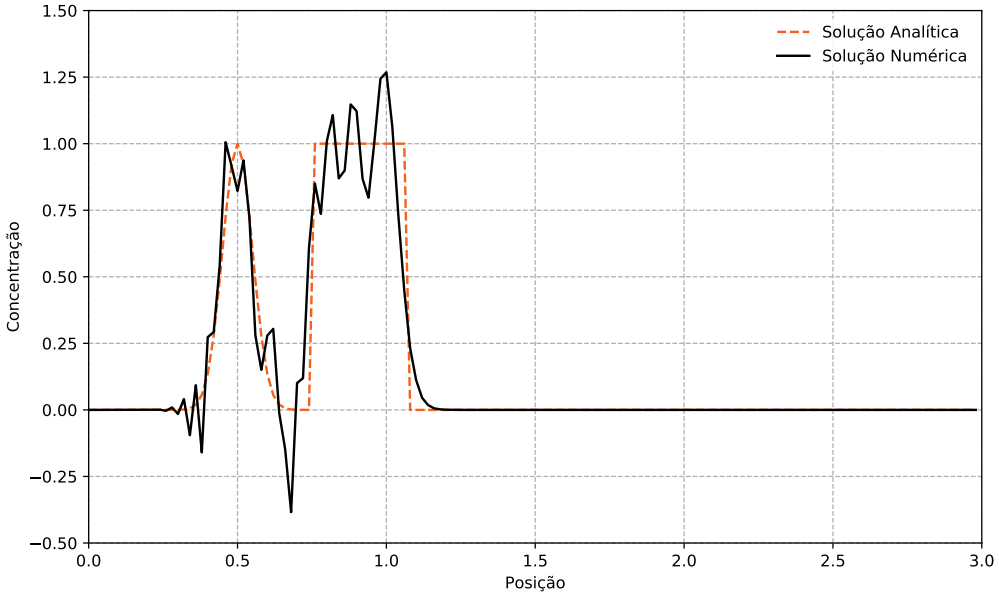

Figura 39 – LeapFrog - Primeiro Experimento em t = 0,5

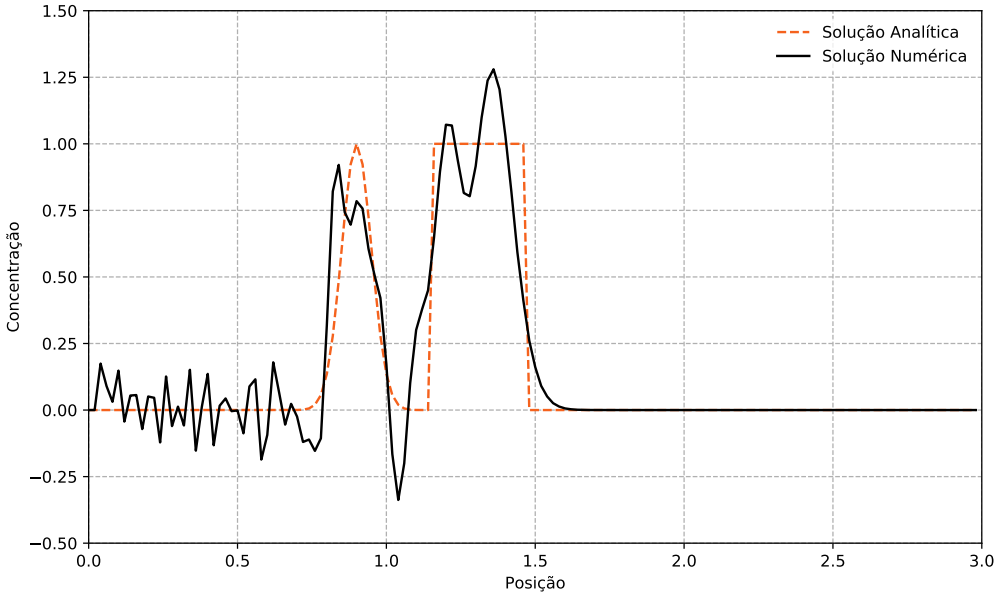

Figura 40 – LeapFrog - Primeiro Experimento em t = 1,5

3.4.2 Segundo Experimento com o Método LeapFrog

Abaixo apresentamos os resultados do segundo experimento com o método Leap-Frog.

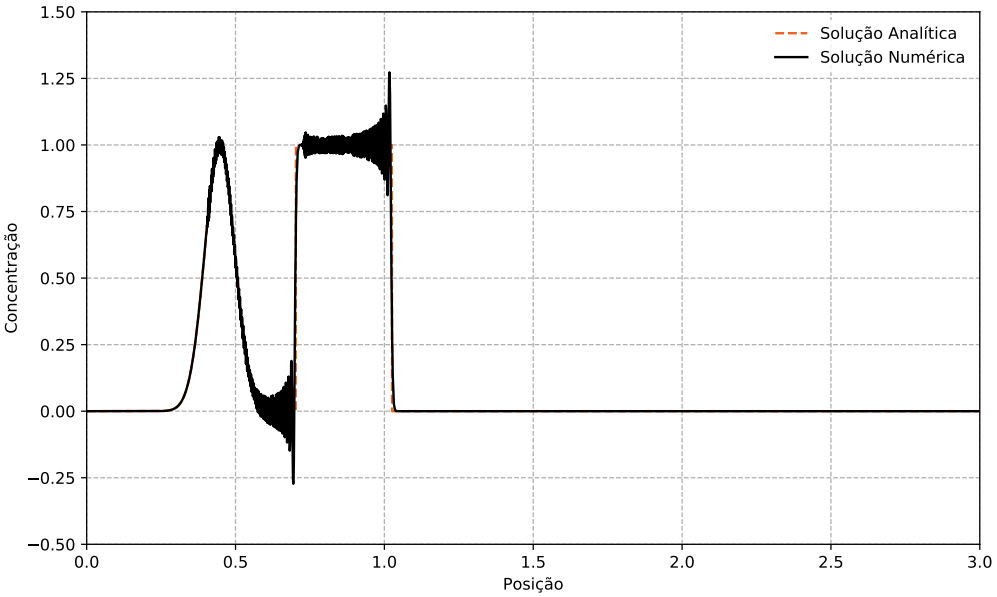

Figura 41 – LeapFrog - Segundo Experimento em t = 0,25

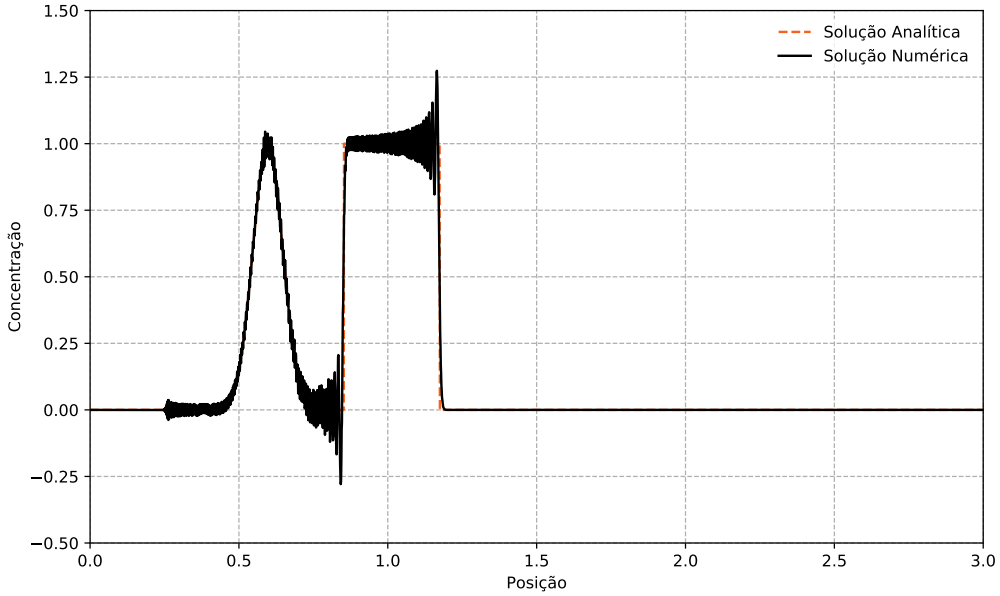

Figura 42 – LeapFrog - Segundo Experimento em t = 0,5

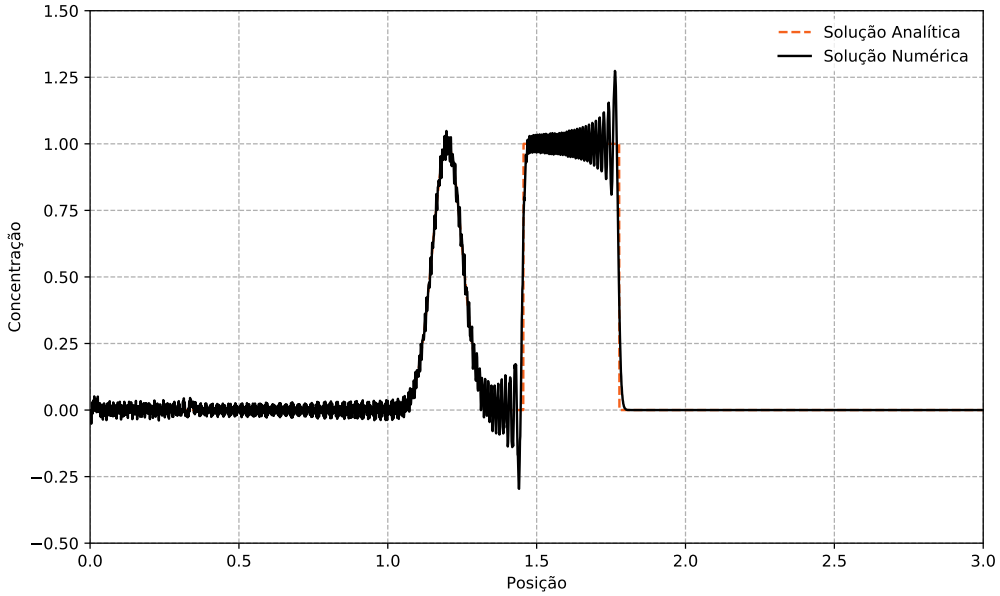

Figura 43 – LeapFrog - Segundo Experimento em t = 1,5

3.4.3 Terceiro Experimento com o Método LeapFrog

Abaixo apresentamos os resultados do terceiro experimento com o método LeapFrog.

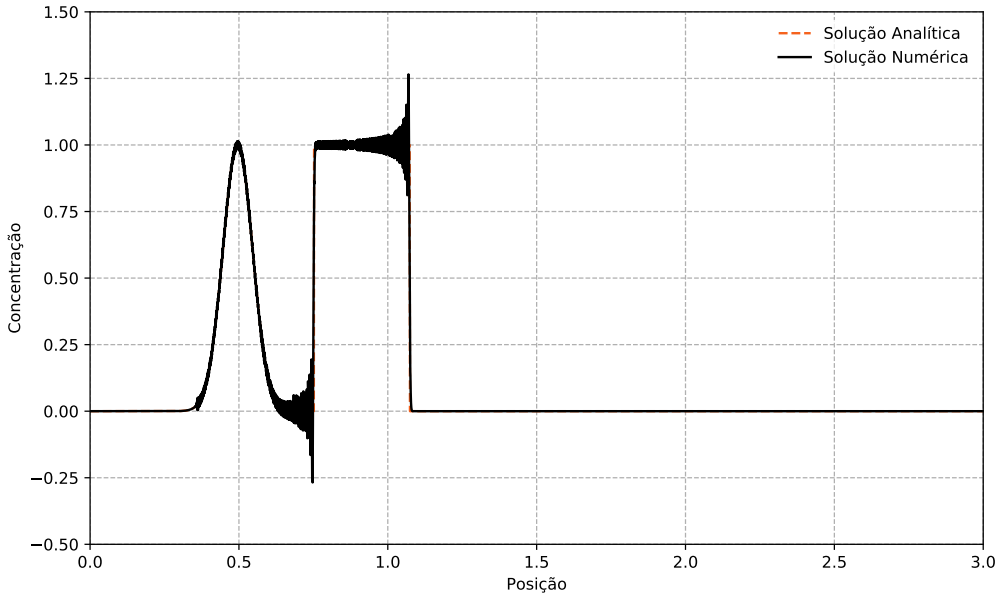

Figura 44 – LeapFrog - Terceiro Experimento em t = 0,25

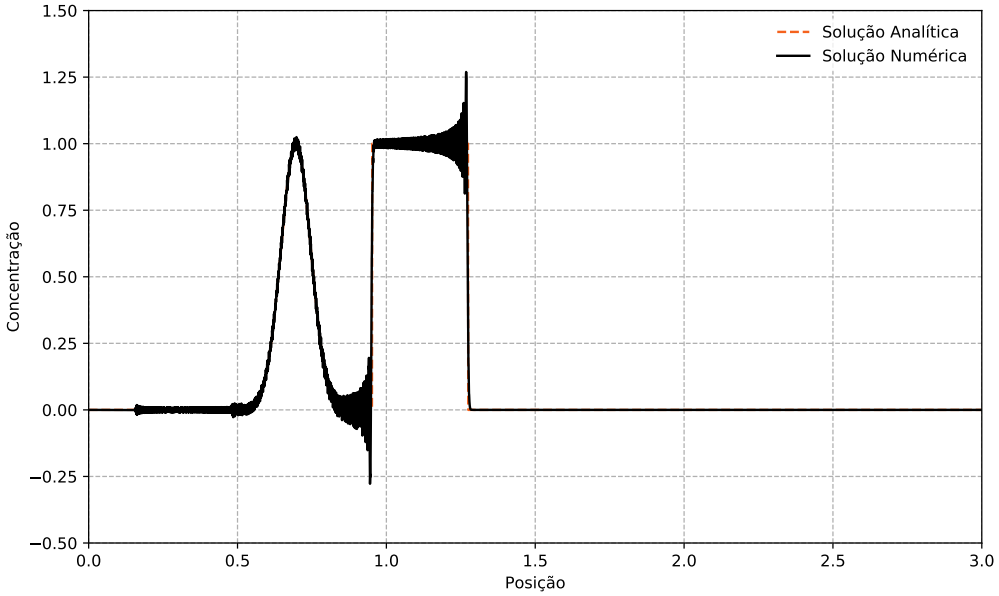

Figura 45 – LeapFrog - Terceiro Experimento em t = 0,5

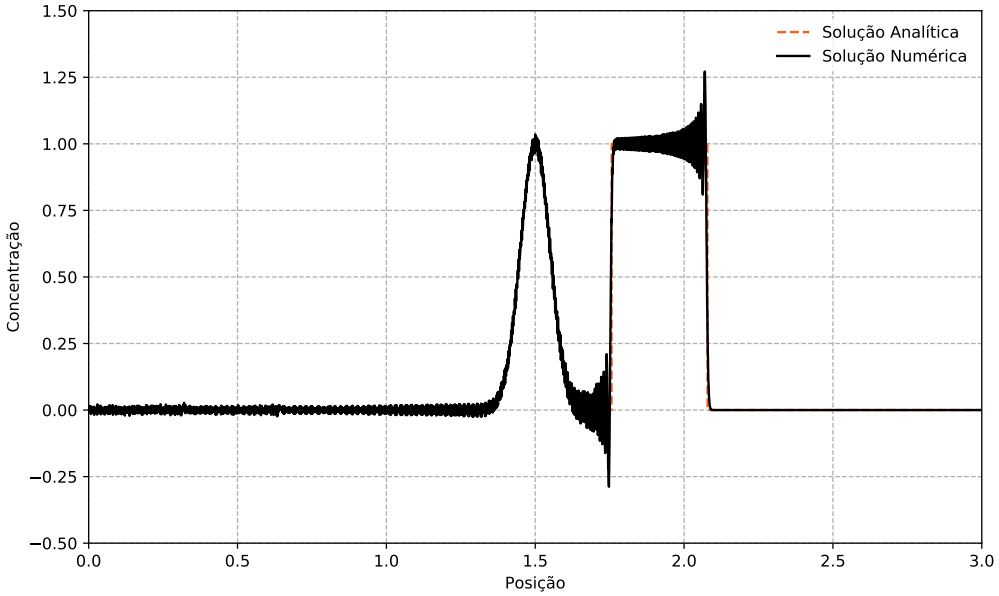

Figura 46 – LeapFrog - Terceiro Experimento em t = 1,5

3.4.4 Quarto Experimento com o Método LeapFrog

Abaixo apresentamos os resultados do quarto experimento com o método LeapFrog.

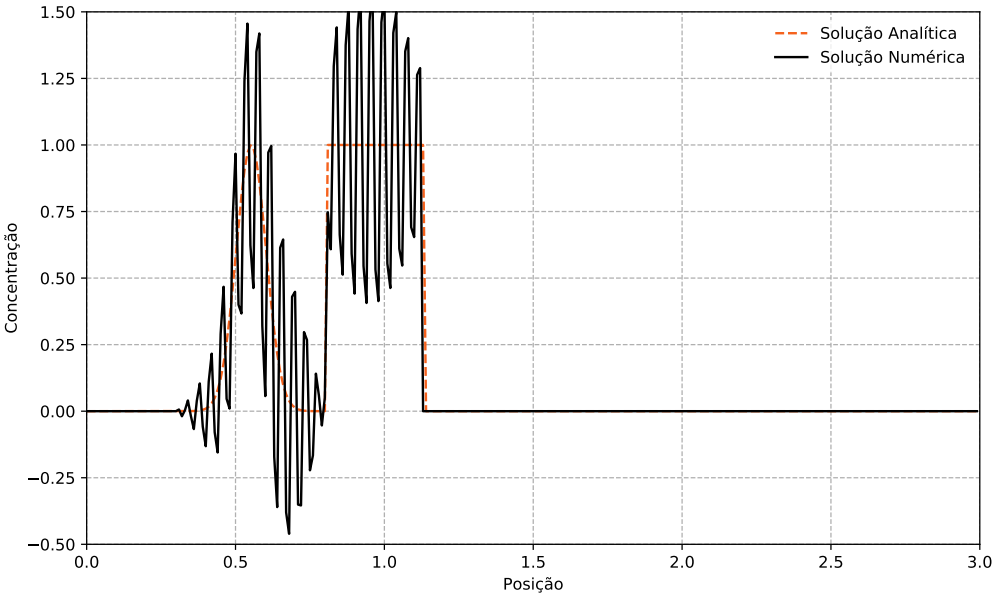

Figura 47 – LeapFrog - Quarto Experimento em t = 0,25

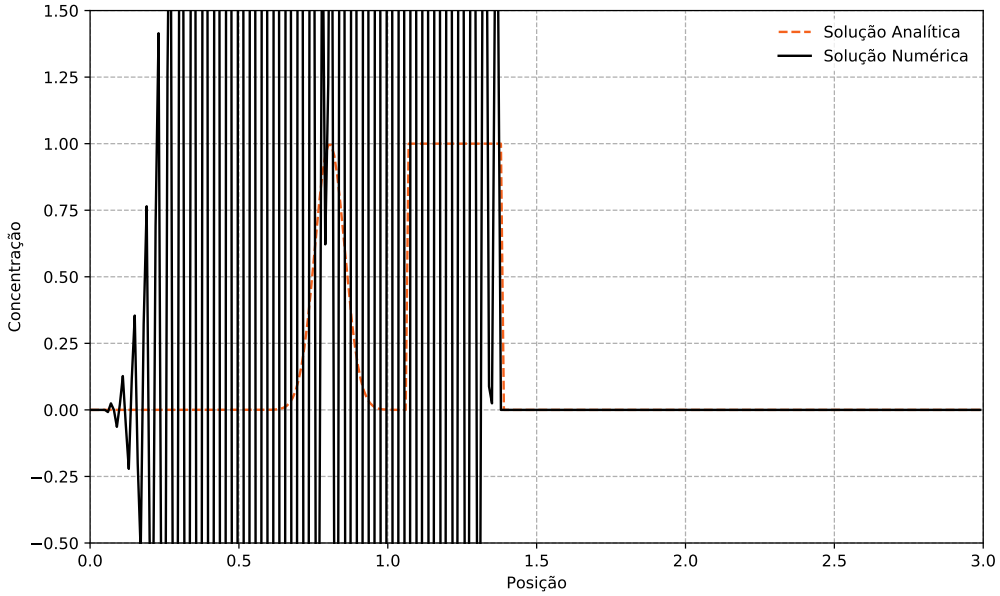

Figura 48 – LeapFrog - Quarto Experimento em t = 0,5

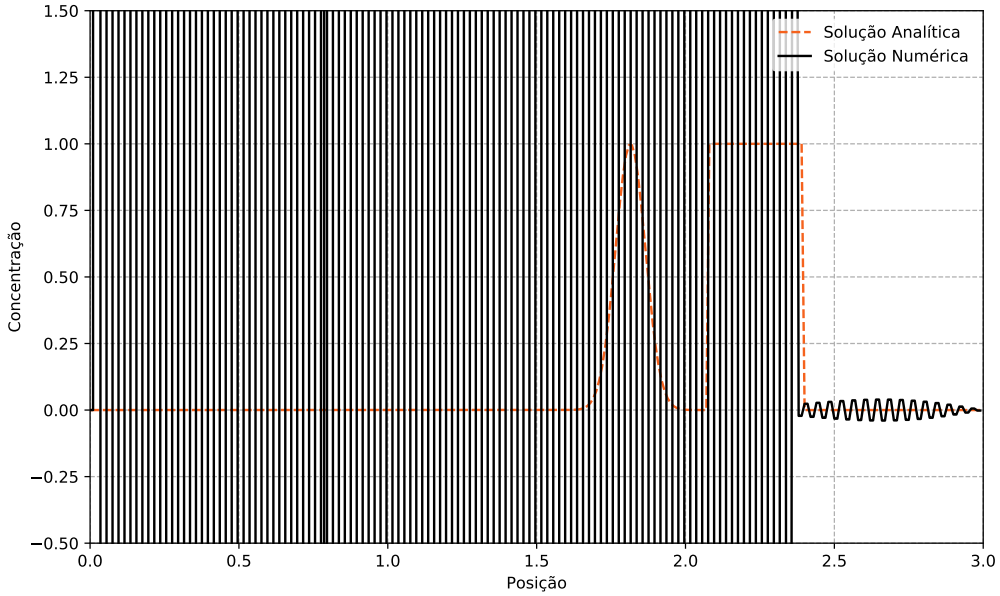

Figura 49 – LeapFrog - Quarto Experimento em t = 1,5

4 Discussão

Conforme podemos observar no Capítulo 3 os métodos de primeira ordem garantiram uma boa aproximação quando houve um refinamento da malha espacial, inclusive na presença das descontinuidades em a e b como por exemplo o método Lax-Friedrichs no experimento da subseção 3.1.3 e também garantiram uma propagação suave dos erros gerados pela violação da condição de estabilidade como é possível identificar na subseção 3.2.4, além disso, como era já esperado, pois um erro maior é característica desses métodos, para malhas menos refinadas a aproximação ficou visivelmente longe da solução analítica.

Em relação aos métodos de segunda ordem, em ambos os casos, com malhas muito ou pouco refinadas obteve-se bons resultados. Em malhas espaciais bastante refinadas as soluções obtidas pelos métodos Lax-Wendroff e LeapFrog foram muito próximas da curva analítica. Entretanto na presença de descontinuidades o método LeapFrog gerou muitas oscilações e uma propagação de erros muito rápida na presença destas.

Já o método Lax-Wendroff mesmo impactado pelas descontinuidades do domínio, se manteve mais estável na presença dessas oscilações como pode ser notado na subseção 3.3.3. Esse mesmo comportamento foi observado nas condições do quarto experimento onde a solução diverge mais rapidamente para o LeapFrog do que para Lax-Wendroff.

5 Conclusão

De acordo com os resultados obtidos e através da comparação dos mesmos com a curva analítica, podemos confirmar algumas das características teoricamente conhecidas de cada método, como o impacto da ordem do método numérico aplicado a uma determinada solução e como o refinamento da malha é importante quando o método aplicado tem um erro maior. Além disso ao se empregar o métodos numéricos na resolução de problemas de engenharia é necessário garantir que as condições de estabilidade sejam satisfeitas e analisar as condições iniciais, de contorno ou ambas, para garantir que o problema seja bem-posto e a solução seja adequada.

Referências

CAUSON, D. M.; MINGHAM, P. C. G. *INTRODUCTORY FINITE DIFFERENCE METHODS FOR PDES*. England: Professor D. M. Causon, Professor C. G. Mingham Ventus Publishing ApS, 2010. Acesso em: 15 jul 2018. Citado na página 12.

SOUTO, H. P. A. *Métodos Numéricos para Equações Diferenciais II*. Nova Friburgo - Rio de Janeiro: Helio Pedro Amaral Souto, 2017. Citado na página 13.

Anexos

ANEXO A – Código Python para o Método Lax-Friedrichs

```
#!/usr/bin/python

import time;
import math;
import numpy as np;
import matplotlib.pyplot as plt
import pylab


A = 0.556;
B = 0.878;

L = 3

DX = 0.0005;

DT = 0.0005;

U = 0.8;

shots = [ 0.245, 0.495, 1.5 ];

maxTime = 2;

totalSizeParts = int( L / DX );

totalTimeParts = int( maxTime / DT );

Qs = np.zeros( shape = ( totalTimeParts, totalSizeParts ) );

C = 1;

def s( x ):
```

```
    if( ( x >= A ) and ( x <= B ) ):
        return 1.;
    else:
        return 0.;
def initialCondition( x ):
    result = math.exp( -200 * pow( ( x - 0.3 ), 2 ) ) + s( x );
    return result;

def analyticSolution( x, t ):
    result = initialCondition( x -U*t );
    return result;
def laxFriedrichs( i, x ):
    if( x - 1 < 1 ):
        xPrevious = 0
    else:
        xPrevious = ( x - 1 )
    if( x + 1 >= totalSizeParts ):
        xNext = totalSizeParts - 2
    else:
        xNext = ( x + 1 )
```

```
    if( i + 1 > 1 ):

        iPrevious = ( i - 1 ) + totalTimeParts -1
        iPrevious = 0

    else:

        iPrevious = ( i - 1 )

    Q = ( ( Qs[ i ][ xPrevious ] + Qs[ i ][ xNext ] ) / 2 ) - ( ( U * ( DT/( 2 * DX

    return Q;

startTime = time.time()

for x in range( 0, totalSizeParts ):

    Qs[ 0 ][ x ] = initialCondition( x * DX );

for i in range( 0, totalTimeParts ):

    if( i == ( totalTimeParts - 1 ) ):
        break;

    for x in range( 0, totalSizeParts ):

        Qs[ i + 1 ][ x ] = laxFriedrichs( i, x );

xArray = [ DX*i for i in range( 0, totalSizeParts ) ]

endTime = time.time();

executionTime = endTime - startTime;

print('Total Execution Time: {0:.3f}'.format( executionTime ), 's' );

for p in shots:
```

```
p = int(math.ceil(p/DT));

if( p > totalTimeParts ):
    p = totalTimeParts-1;

analyticSolutionData = [ analyticSolution( i * DX, p*DT ) for i in range( 0, totalSi

numericalData = Qs[ p ];

fig, ax1 = plt.subplots()

plt.axis([0, L, -0.5, 1.5])

color = (245/255, 98/255, 29/255, 1)
ax1.set_xlabel('Posição')
ax1.set_ylabel('exp', color=color)
ax1.plot(xArray, analyticSolutionData, linestyle='dashed', color=color, label= 'Solu
ax1.tick_params(axis='y', labelcolor=color)


color = (0,0,0,1)
ax1.set_ylabel( 'Concentração' , color=color)
ax1.plot(xArray, numericalData, linestyle='solid', color=color, label = 'Solução Num
ax1.tick_params(axis='y', labelcolor=color)

ax1.legend( [ 'Solução Analítica', 'Solução Numérica' ], loc='upper right', frameon
ax1.grid( linestyle = 'dashed' )
ax1.legend().get_frame().set_linewidth(0.0)

F = pylab.gcf()

F.set_size_inches( ( 10, 6 ) )

plt.savefig("lax-friedrichs-plots/Lax-Friedrichs - SHOT {0} - Courant Number = {1} -

plt.show()
```

ANEXO B – Código Python para o Método UpWind

```
    #!/usr/bin/python

import time;
import math;
import numpy as np;
import matplotlib.pyplot as plt
import pylab

A = 0.556;
B = 0.878;

L = 3

DX = 0.0005;

DT = 0.0005;

U = 0.8;

shots = [ 0 ];

maxTime = 0.5;

totalSizeParts = int( L / DX );

totalTimeParts = int( maxTime / DT );

Qs = np.zeros( shape = ( totalTimeParts, totalSizeParts ), dtype = "float64" );

C = 1;

def s( x ):
```

```
    if( ( x >= A ) and ( x <= B ) ):

        return 1.;

    else:

        return 0.;

def initialCondition( x ):

    result = math.exp( -200 * pow( ( x - 0.3 ), 2 ) ) + s( x );

    return result;

def analyticSolution( x, t ):

    result = initialCondition( x -U*t );

    return result;

def upWind( i, x ):

    if( x - 1 < 0 ):

        xPrevious = 0

    else:

        xPrevious = x - 1

    F1 = Qs[ i ][ x ];
    F2 = Qs[ i ][ xPrevious ];

    Q = Qs[ i ] [ x ] - ( U * ( DT / DX ) * (F1 - F2) );

    return Q;


startTime = time.time()
```

```
for x in range( 0, totalSizeParts ):

    Qs[ 0 ][ x ] = initialCondition( x * DX );

for i in range( 0, totalTimeParts ):

    if( i == ( totalTimeParts - 1 ) ):
        break;

    for x in range( 0, totalSizeParts ):

        Qs[ i + 1 ][ x ] = upWind( i, x );

xArray = [ DX*i for i in range( 0, totalSizeParts ) ]

endTime = time.time();

executionTime = endTime - startTime;

print('Total Execution Time: {0:.3f}'.format( executionTime ), 's' );

for p in shots:

    p = int(math.ceil(p/DT));

    if( p > totalTimeParts ):
        p = totalTimeParts-1;

    analyticSolutionData = [ analyticSolution( i * DX, p*DT ) for i in range( 0, to

    numericalData = Qs[ p ];

    fig, ax1 = plt.subplots()

    plt.axis([0, L, -0.5, 1.5])

    color = (245/255, 98/255, 29/255, 1)
    ax1.set_xlabel('Posição')
```

```
    ax1.set_ylabel('exp', color=color)
    ax1.plot(xArray, analyticSolutionData, linestyle='dashed', color=color, label= 'Solu
    ax1.tick_params(axis='y', labelcolor=color)



    color = (0,0,0,1)
    ax1.set_ylabel( 'Concentração' , color=color)
    ax1.plot(xArray, numericalData, linestyle='solid', color=color, label = 'Solução Num
    ax1.tick_params(axis='y', labelcolor=color)

    ax1.legend( [ 'Solução Analítica', 'Solução Numérica' ], loc='upper right', frameon
    ax1.grid( linestyle = 'dashed' )
    ax1.legend().get_frame().set_linewidth(0.0)

    F = pylab.gcf()

    F.set_size_inches( ( 10,6 ) )

    plt.savefig("upwind-plots/Lax-Friedrichs - SHOT {0} - Courant Number = {1} - dT = {2

    plt.show()

endTime = time.time();

executionTime = endTime - startTime;

print('Total Execution Time: {0:.3f}'.format( executionTime ), 's' );
```

ANEXO C – Código Python para o Método Lax-Wendroff

```
    #!/usr/bin/python
import time;
import math;
import numpy as np;
import matplotlib.pyplot as plt
import pylab

A = 0.556;
B = 0.878;

L = 3

DX = 0.01;

DT = 0.01;

U = 0.8;

shots = [ 0.245, 0.495, 1.5 ];

maxTime = 2;

totalSizeParts = int( L / DX );

totalTimeParts = int( maxTime / DT );

Qs = np.zeros( shape = ( totalTimeParts, totalSizeParts ) );

C = U*(DT/DX);

def s( x ):

    if( ( x >= A ) and ( x <= B ) ):
```

```
        return 1.;

    else:

        return 0.;

def initialCondition( x ):

    result = math.exp( -200 * pow( ( x - 0.3 ), 2 ) ) + s( x );

    return result;


def analyticSolution( x, t ):

    result = initialCondition( x -U*t );

    return result;

def laxWendroff( i, x ):

    if( x - 1 < 0 ):

        xPrevious = 0

        Q = analyticSolution( 0, 0 );

        return Q;

    else:

        xPrevious = ( x - 1 )

    if( x + 1 >= totalSizeParts ):

        Q = analyticSolution( L, 0 );

        return Q;
```

```
        xNext = totalSizeParts - 2

    else:

        xNext = ( x + 1 )

    Q = Qs[ i ] [ x ] - ( C ) * ( ( Qs[ i ][ xNext ] - Qs[ i ][ xPrevious ] ) - ( C

    return Q;

startTime = time.time()

for x in range( 0, totalSizeParts ):

    Qs[ 0 ][ x ] = initialCondition( x * DX );

for i in range( 0, totalTimeParts ):

    if( i == ( totalTimeParts - 1 ) ):
        break;

    for x in range( 0, totalSizeParts ):

        Qs[ i + 1 ][ x ] = laxWendroff( i, x );

xArray = [ DX*i for i in range( 0, totalSizeParts ) ]

endTime = time.time();

executionTime = endTime - startTime;

print('Total Execution Time: {0:.3f}'.format( executionTime ), 's' );

for p in shots:

    p = int(math.ceil(p/DT));

    if( p > totalTimeParts ):
```

```
    p = totalTimeParts-1;

analyticSolutionData = [ analyticSolution( i * DX, p*DT ) for i in range( 0, totalSi

numericalData = Qs[ p ];

fig, ax1 = plt.subplots()

plt.axis([0, L, -0.5, 1.5])

color = (245/255, 98/255, 29/255, 1)
ax1.set_xlabel('Posição')
ax1.set_ylabel('exp', color=color)
ax1.plot(xArray, analyticSolutionData, linestyle='dashed', color=color, label= 'Solu
ax1.tick_params(axis='y', labelcolor=color)


color = (0,0,0,1)
ax1.set_ylabel( 'Concentração' , color=color)
ax1.plot(xArray, numericalData, linestyle='solid', color=color, label = 'Solução Num
ax1.tick_params(axis='y', labelcolor=color)

ax1.legend( [ 'Solução Analítica', 'Solução Numérica' ], loc='upper right', frameon
ax1.grid( linestyle = 'dashed' )
ax1.legend().get_frame().set_linewidth(0.0)

F = pylab.gcf()

F.set_size_inches( ( 10, 6 ) )

plt.savefig("lax-wendroff-plots/Lax-Wendroff - SHOT {0} - Courant Number = {1} - dT

plt.show()
```

ANEXO D – Código Python para o Método LeapFrog

```
    #!/usr/bin/python

import time;
import math;
import numpy as np;
import matplotlib.pyplot as plt
import pylab

A = 0.556;
B = 0.878;

L = 3

DX = 0.01;

DT = 0.01;

U = 1.01;

shots = [ 0.245, 0.495, 1.5 ];

maxTime = 2;

totalSizeParts = int( L / DX );

totalTimeParts = int( maxTime / DT );

Qs = np.zeros( shape = ( totalTimeParts, totalSizeParts ), dtype = np.float64 );

C = 1;

def s( x ):
```

```
    if( ( x >= A ) and ( x <= B ) ):

        return 1.;

    else:

        return 0.;

def initialCondition( x ):

    result = math.exp( -200 * pow( ( x - 0.3 ), 2 ) ) + s( x );

    return result;


def analyticSolution( x, t ):

    result = initialCondition( x -U*t );

    return result;

def leapFrog( i, x ):

    if( x - 1 < 0 ):

        xPrevious = 0

    else:

        xPrevious = x - 1

    if( x == ( totalSizeParts - 1 ) ):

        xNext = totalSizeParts - 1

    else:
        xNext = x + 1

    if( xNext == (x + 1) and xPrevious == 0 ):
```

```
        Q = analyticSolution(  xNext * DX, xPrevious * DT );
    else:
        Q = Qs[ i - 1 ] [ x ] - ( U * ( DT / DX ) * ( Qs[ i ][ xNext ] - Qs[ i ][ x

    return Q;

def laxWendroff( i, x ):

    if( x - 1 < 0 ):

        xPrevious = 0

        Q = analyticSolution( 0, 0 );

        return Q;

    else:

        xPrevious = ( x - 1 )

    if( x + 1 >= totalSizeParts ):

        Q = analyticSolution( 0, 0 );

        return Q;

        xNext = totalSizeParts - 2

    else:

        xNext = ( x + 1 )

    Q = Qs[ i ] [ x ] - ( U * ( DT / DX ) ) * ( ( Qs[ i ][ xNext ] - Qs[ i ][ xPrev

    return Q;

startTime = time.time()

for x in range( 0, totalSizeParts ):
```

```
    Qs[ 0 ][ x ] = initialCondition( x * DX );

for x in range( 0, totalSizeParts ):

    Qs[ 1 ][ x ] = laxWendroff( 0, x );

for i in range( 1, totalTimeParts ):

    if( i == ( totalTimeParts - 1 ) ):
        break;

    for x in range( 0, totalSizeParts ):

        Qs[ i + 1 ][ x ] = leapFrog( i, x );

xArray = [ DX*i for i in range( 0, totalSizeParts ) ]

endTime = time.time();

executionTime = endTime - startTime;

print('Total Execution Time: {0:.3f}'.format( executionTime ), 's' );

for p in shots:

    p = int(math.ceil(p/DT));

    if( p > totalTimeParts ):
        p = totalTimeParts-1;

    analyticSolutionData = [ analyticSolution( i * DX, p*DT ) for i in range( 0, totalSi

    numericalData = Qs[ p ];

    fig, ax1 = plt.subplots()

    plt.axis([0, L, -0.5, 1.5])
```

```
color = (245/255, 98/255, 29/255, 1)
ax1.set_xlabel('Posição')
ax1.set_ylabel('exp', color=color)
ax1.plot(xArray, analyticSolutionData, linestyle='dashed', color=color, label=
ax1.tick_params(axis='y', labelcolor=color)


color = (0,0,0,1)
ax1.set_ylabel( 'Concentração' , color=color)  # we already handled the x-label
ax1.plot(xArray, numericalData, linestyle='solid', color=color, label = 'Soluçã
ax1.tick_params(axis='y', labelcolor=color)

ax1.legend( [ 'Solução Analítica', 'Solução Numérica' ], loc='upper right', fra
ax1.grid( linestyle = 'dashed' )
ax1.legend().get_frame().set_linewidth(0.0)

F = pylab.gcf()

F.set_size_inches( (10,6 ) )# resetthe size


plt.savefig("leapfrog-plots/LeapFrog - SHOT {0} - Courant Number = {1} - dT = {

plt.show()
```