Dinâmica 25/09/2023 a 25/09/2023

Modelo Numérico da Atmosfera

MET-576-4

Modelagem Numérica da Atmosfera

Dr. Paulo Yoshio Kubota

Os métodos numéricos, formulação e parametrizações utilizados nos modelos atmosféricos serão descritos em detalhe.

3 Meses
24 Aulas (2 horas cada)

Modelo Numérico da Atmosfera

Dinâmica:

Métodos numéricos amplamente utilizados na solução numérica das equações diferencias parciais que governam os movimentos na atmosfera serão o foco, mas também serão analisados os novos conceitos e novos métodos.

Modelo Numérico da Atmosfera

- ✓ Métodos de diferenças finitas.
- ✓ Acurácia.
- ✓ Consistência.
- ✓ Estabilidade.
- ✓ Convergência.
- ✓ Grades de Arakawa A, B, C e E.
- ✓ Domínio de influência e domínio de dependência.
- ✓ Dispersão numérica e dissipação.
- ✓ Definição de filtros monótono e positivo.
- ✓ Métodos espectrais.
- ✓ Métodos de volume finito.
- ✓ Métodos Semi-Lagrangeanos.
- ✓ Conservação de massa local.
- ✓ Esquemas explícitos versus semi-implícitos.
- ✓ Métodos semi-implícitos.

Modelo Numérico da Atmosfera

Consistency

O FDE é consistente com o PDE se, no limite $\Delta x \to 0$ e $\Delta t \to 0$, o FDE coincidir com o PDE.
Obviamente, esse é um requisito básico que o FDE deve atender se suas soluções forem boas aproximações das soluções do PDE.

A consistência é bastante simples de verificar:

- Substitua u em vez de U no FDE.
- Avalie todos os termos usando uma expansão da série de Taylor centrada no ponto $(x_j t_n)$.
- Subtraia o PDE do FDE.

Se a diferença (erro de truncamento local) for para zero como $\Delta x \to 0$ e $\Delta t \to 0$, então o FDE é consistente com o PDE.

Modelo Numérico da Atmosfera

Erro de truncamento e Consistência

Vamos verificar a consistência do esquema a montante para a equação de advecção.
Primeiro, considere a expansão da série Taylor:

$$\left.\begin{aligned} u_j^{n+1} &= \left(u + u_t\Delta t + \tfrac{1}{2}u_{tt}\Delta t^2 + \cdots\right)_j^n \\ u_{j-1}^n &= \left(u - u_x\Delta x + \tfrac{1}{2}u_{xx}\Delta x^2 - \cdots\right)_j^n \end{aligned}\right\}$$

Nós substituímos isso no FDE e obtemos

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + c\frac{U_j^n - U_{j-1}^n}{\Delta x} = 0$$

$$\left(u_t + \tfrac{1}{2}u_{tt}\Delta t + \cdots\right)_j^n + c\left(u_x - \tfrac{1}{2}u_{xx}\Delta x + \cdots\right)_j^n \simeq 0$$

Subtrair o PDE resulta no erro de truncamento local:

$$\tau = \left(\frac{u_{tt}}{2}\right)\Delta t - \left(\frac{cu_{xx}}{2}\right)\Delta x + \text{H.O.T.} = O(\Delta t) + O(\Delta x)$$

Modelo Numérico da Atmosfera

Erro de truncamento e Consistência

Novamente, o erro de truncamento local é:

$$\tau = \left(\frac{u_{tt}}{2}\right)\Delta t - \left(\frac{cu_{xx}}{2}\right)\Delta x + \text{H.O.T.} = O(\Delta t) + O(\Delta x)$$

Claramente, quando $\Delta x \to 0$ e $\Delta t \to 0$, temos $\tau \to 0$. Portanto, o FDE é consistente.

Observe que os erros de truncamento de tempo e espaço são de primeira ordem, porque as diferenças finitas não estão centradas tanto no espaço quanto no tempo.

Erros de truncamento para diferenças centralizadas são de segunda ordem.

Portanto, em geral, diferenças centralizadas são mais precisas do que diferenças não centradas.

# # #

Erros de truncamento são um fator crucial para determinar a precisão da previsão no NWP.

Modelo Numérico da Atmosfera

Convergência e Estabilidade

A segunda questão levantada acima foi **se a solução do FDE converge para a solução do PDE.**

Isto é, se deixarmos $\Delta x \to 0$ e $\Delta t \to 0$, de modo que $j\Delta x \to x$ e $n\Delta t \to t$, portanto, $U(j\Delta x, n\Delta t) \Rightarrow u(x,t)$?

Isso é claramente importante **e pode ser respondido considerando outro problema, o da estabilidade computacional.**

Considere novamente a equação de advecção

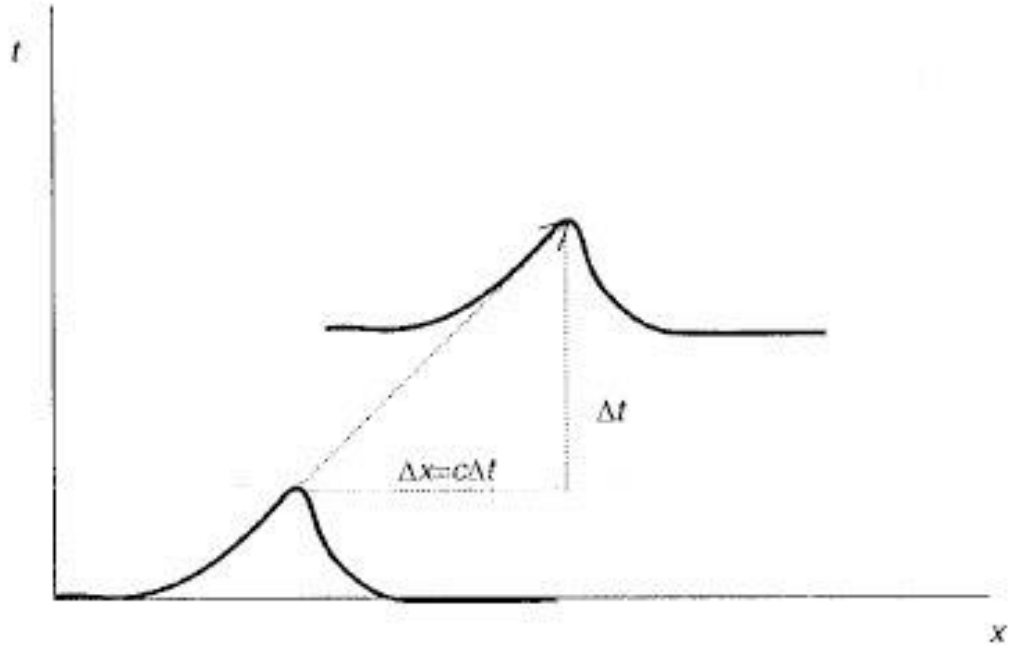
$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0\,,$$

que tem a solução $u(x,t) = u(x+ct, 0)$

A forma da solução u (x, 0) se translada ao longo do eixo x com velocidade c (veja a Figura abaixo).

Modelo Numérico da Atmosfera

Convergência e Estabilidade



Esquema da solução da equação de advecção (para c> 0).

Modelo Numérico da Atmosfera

Convergência e Estabilidade

O FDE para o esquema upstream pode ser escrito como

$$U_j^{n+1} = (1-\mu)U_j^n + \mu U_{j-1}^n$$

onde

$$\mu \equiv \frac{c\Delta t}{\Delta x}$$

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} + c\frac{U_j^n - U_{j-1}^n}{\Delta x} = 0$$

é o número Courant (ou número Lewy).

Vamos supor que $0 \leq \mu \leq 1$.

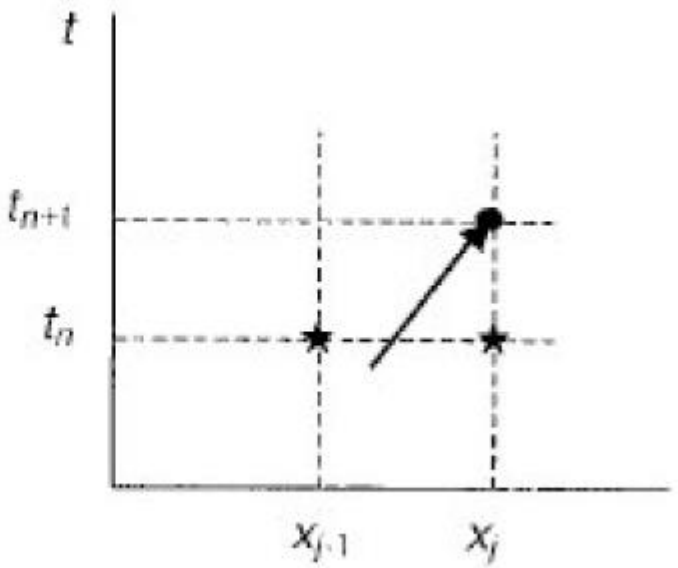
Então a solução FDE no novo nível de tempo U_j^{n+1} é **interpolada** entre os valores U_j^n e U_{j-1}^n.

Nesse caso, o esquema de advecção funciona da maneira que deveria, porque a verdadeira solução está entre esses valores (U_j^{n+1} e U_j^n).

Modelo Numérico da Atmosfera

Convergência e Estabilidade

(a) $0 \leq c \leq \dfrac{\Delta x}{\Delta t}$



Esquema da relação entre $\Delta x, \Delta t$ *e* c levando à interpolação da solução no nível de tempo $n + 1$.

$$0 < \mu \equiv \frac{c\Delta t}{\Delta x} < 1$$

Modelo Numérico da Atmosfera

Convergência e Estabilidade

$$0 < \mu \equiv \frac{c\Delta t}{\Delta x} < 1$$

No entanto, suponha que esta condição não seja satisfeita, de modo que

$$\mu = \frac{c\Delta t}{\Delta x} > 1$$

$$\mu = \frac{c\Delta t}{\Delta x} < 0\,.$$

Então, a parcela que chega ao ponto x_j no tempo t_{n+1} vem de algum lugar fora do intervalo $(x_j - 1, x_j)$ no tempo t_n.

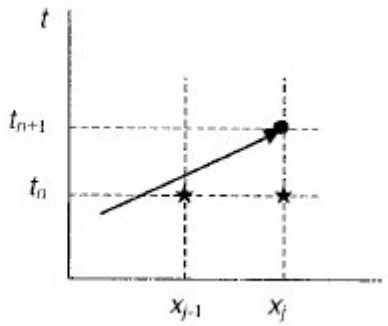
Lembre-se de que $\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0$ é uma aproximação linear de $\frac{du}{dt} = 0$

Então os valores de U_j^{n+1} devem ser **extrapolada** dos valores U_j^n e U_{j-1}^n.

Modelo Numérico da Atmosfera

Convergência e Estabilidade

(b) $0 \leq \frac{\Delta x}{\Delta t} \leq c$

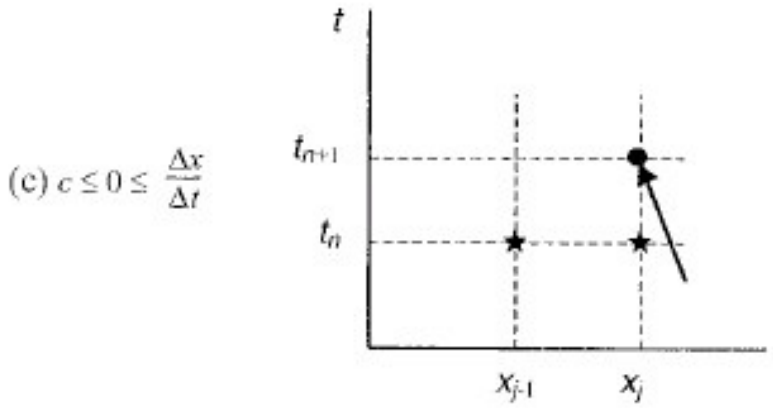


Esquema da relação entre $\Delta x, \Delta t$ *e* c liderando à extraploação da solução no nível de tempo $n + 1$.

$$\mu \equiv \frac{c\Delta t}{\Delta x} > 1$$

Modelo Numérico da Atmosfera

Convergência e Estabilidade



Esquema da relação entre $\Delta x, \Delta t$ *e* c liderando à extraploação da solução no nível de tempo $n + 1$.

$$\mu \equiv \frac{c\Delta t}{\Delta x} < 0$$

Convergência e Estabilidade

O problema com a extrapolação é que o máximo valor absoluto da solução U_j^n aumenta a cada intervalo de tempo.

Tomando valores absolutos do FDE, obtemos

$$\left|U_j^{n+1}\right| \leq \left|U_j^n\right||1-\mu| + \left|U_{j-1}^n\right||\mu|$$

Agora definindo $\Upsilon^{n+1} = max_j\left|U_j^n\right|$, $\Upsilon^{n+2} = max_j\left|U_j^{n+1}\right|$ tem-se:

$$max_j\left|U_j^{n+1}\right| \leq max_j\left|U_j^n\right||1-\mu| + max_j\left|U_{j-1}^n\right||\mu|$$

$$\Upsilon^{n+2} \leq \Upsilon^{n+1}|1-\mu| + \Upsilon^{n+1}|\mu|$$

$$\Upsilon^{n+1}\Upsilon^1 \leq \Upsilon^n\Upsilon^1|1-\mu| + \Upsilon^n\Upsilon^1|\mu|$$

$$\Upsilon^{n+1} \leq \Upsilon^n|1-\mu| + \Upsilon^n|\mu|$$

$$\Upsilon^{n+1} \leq \{|1-\mu| + |\mu|\}\Upsilon^n$$

Modelo Numérico da Atmosfera

Convergência e Estabilidade

$$\Upsilon^{n+1} \leq \{|1-\mu| + |\mu|\}\Upsilon^{n}$$

e $\Upsilon^{n+1} \leq \Upsilon^{n}$se e somente se $0 \leq \mu \leq 1$.

Se a condição $0 \leq \mu \leq 1$ não for satisfeita, a solução não é limitada e cresce com n.

Se deixarmos $\Delta x \to 0, \Delta t \to 0$ com μ = const., Isso só piorará as coisas, porque então $n \to \infty$.

Na prática, se a condição $0 \leq \mu \leq 1$ não for satisfeita, o FDE explode em alguns intervalos de tempo.

Convergência e Estabilidade

Exercício prático:

Use o modelo simples SLAM para explorar o fenômeno da instabilidade computacional.

• Escolha Δt pequeno e faça uma integração completa.

• Aumente Δt e observe a solução do modelo “explodir”.

• Para experimento numérico, determine aproximadamente o valor máximo de Δt que produz integrações estáveis.

• Relacione este máximo com o número Courant. O que isso implica sobre a velocidade máxima de fase do sistema?

Estabilidade Computacional

Agora definimos estabilidade computacional.

Definição: Um FDE é computacionalmente estável se a solução do FDE em um tempo fixo $t = n\Delta t$ é limitado como $\Delta t \to 0$.

Observe que com $n\Delta t$ fixo, Δt->0 implica $n \to \infty$

Iremos derivar uma condição de estabilidade que envolve o Courant Number.

A condição no número de Courant é geralmente conhecida como critério Courant-Friedrichs-Lewy ou simplesmente a condição CFL.

# # #

Lembre-se da história de Courant, Friedrichs e Lewy em Göttingen.

Modelo Numérico da Atmosfera

Estabilidade Computacional

Agora podemos afirmar o teorema fundamental de Lax-Richtmyer:

Dado um problema de valor inicial linear corretamente proposto e um esquema de diferenças finitas que satisfaça a condição de consistência, então a estabilidade do FDE é a condição necessária e suficiente para a convergência.

$$Estabilidade <=> Convergencia$$

Para sistemas consistentes

Este teorema nos permite estabelecer convergência examinando as questões mais fáceis de consistência e estabilidade.

Estamos interessados na convergência não porque queremos deixar $\Delta x, \Delta t \to 0$, mas porque queremos ter certeza de que os erros $u(j\Delta x, n\Delta t) - U_j^n$ sejam aceitavelmente pequenos.

Definição $u(j\Delta x, n\Delta t) - U_j^n$ é o erro de truncamento global.

Modelo Numérico da Atmosfera

Estabilidade Computacional

Exemplo: Usamos o critério do método máximo para estudar a condição de estabilidade da equação de difusão

$$\frac{\partial u}{\partial t} = \sigma \frac{\partial^2 u}{\partial x^2}$$

Uma aproximação FDE (esquema FTCS) é dada por

$$\frac{U_j^{n+1} - U_j^n}{\Delta t} = \sigma \frac{U_{j+1}^n - 2U_j^n + U_{j-1}^n}{\Delta x^2}$$

(a verificação da consistência deste FDE é imediata).

Nota: Uma vez que as diferenças são centradas no espaço, mas avançadas no tempo, o erro de truncamento é de primeira ordem no tempo e de segunda ordem no espaço

$$\tau = O(\Delta t) + O(\Delta x)^2 .$$

Podemos escrever o FDE na forma de equação

$$U_j^{n+1} = \mu U_{j+1}^n + (1 - 2\mu) U_j^n + \mu U_{j-1}^n$$

$$\mu = \sigma \Delta t / \Delta x^2 .$$

Modelo Numérico da Atmosfera

Estabilidade Computacional

Novamente

$$U_j^{n+1} = \mu U_{j+1}^n + (1-2\mu)U_j^n + \mu U_{j-1}^n$$

Se tomarmos valores absolutos, e deixarmos $\Upsilon^n = max_j \left|U_j^n\right|$

$$\Upsilon^{n+1} \leq \{|\mu| + |1-2\mu| + |\mu|\}\Upsilon^n$$

Assim, obtemos a condição

$$0 \leq \mu \leq 1/2$$

para garantir que a solução permaneça limitada como $n \to \infty$.
Essa é a condição necessária para a estabilidade do FDE.
# # #
Infelizmente, o critério do máximo só pode ser aplicado em poucos casos.

Na maioria dos FDEs, alguns coeficientes das equações são negativos e o critério não pode ser aplicado.

Precisamos de um método mais poderoso de estabelecer estabilidade.

Modelo Numérico da Atmosfera

O Metodo de von Neumann

Outro critério de estabilidade com uma aplicação muito mais ampla é o critério de estabilidade de von Neumann.
Assumimos que podemos expandir a solução do FDE em um conjunto apropriado de autofunções.
Para simplificar, assumimos uma expansão para a série Fourier:

$$U(x,t) = \sum_k Z_k(t) e^{ikx} \qquad U_j^n = \sum_k Z_k^n e^{ikj\Delta x}$$

A variável de espaço x e o número de onda k podem ser multidimensionais, $x = (x_1, x_2, x_3)$ e k $= (k_1, k_2, k_3)$

mas, para simplificar, consideraremos o caso escalar. Nós definimos x_j = jΔx , t_n = nΔt

Definimos o número de onda para a série Fourier: $p = k\Delta x$

Então a expansão de Fourier é

$$U_j^n = \sum_p Z_p^n e^{ipj} \qquad (\text{Note: } kx = kj\Delta x = pj)$$

Modelo Numérico da Atmosfera

O Método de von Neumann

Quando substituímos esta expansão de Fourier em um FDE linear, obtemos um sistema de equações

$$Z_p^{n+1} = \rho_p Z_p^n$$

Aqui ρ_P é um fator de amplificação que, aplicado ao p-ésimo componente de Fourier da solução no tempo $\mathrm{n}\Delta t$, o avança para o tempo $(\mathrm{n}+1)\Delta t$; ρ_P depende de p, Δt e Δx.

Se conhecermos as condições iniciais

$$U_j^0 = \sum_p Z_p^0 e^{ipj}$$

então a solução do FDE é (lembre-se do aviso sobre sobrescritos)

$$Z_p^n = \rho_p^n Z_p^0$$

Portanto, a estabilidade é garantida se o fator ρ^n for limitado para todo $\mathrm{p} = k\Delta x$ quando $\Delta t \to 0$ e $n \to \infty$

Portanto, devemos ter $|\rho_P| < \mu$ para todo $\mathrm{p} = k\Delta x$ como $n \to \infty$.

Métodos de diferenças finitas.

Consistência (?exatidão?) e Estabilidade da solução

- Se o erro de truncamento do esquema de diferença finita se aproxima de zero, quando $\Delta t \rightarrow 0$, então, o esquema é coerente.
- Para um problema de valor inicial, a coerência não é suficiente para garantir que um esquema numérico vá convergir para as soluções corretas quando $\Delta t \rightarrow 0$ (e $\Delta x \rightarrow 0$).

- O teorema equivalência de Lax, diz que para os estados ser consistente **no método linear**, a estabilidade é a **condição necessária e suficiente para a convergência**.

- Para resolver um problema de valor inicial o **erro round_off** e de **truncamento**, irá acumular com cada passo de tempo, e a solução aproximada às vezes vai **divergir da solução correta**. Se o passo de tempo Δt é pequeno o erro a cada intervalo de tempo deve ser pequeno e deve se acumular mais lentamente.

- No entanto, algumas vezes, a solução numérica irá rapidamente divergir da solução correta, caso em que o método numérico é dito ser Instável.

Análise de Estabilidade EDO linear

Assume-se que a solução para a eq. 7 tem a forma $y^n = A^n e^{ik}$ e substitua no Esquema Avançado Eq. 9

$$\frac{\partial y}{\partial t} = -\lambda y \qquad (7)$$

$$\frac{y^{n+1} - y^n}{\Delta t} = -\lambda y^n \qquad (7)$$

$$y^{n+1} = (1 - \lambda\Delta t) y^n \qquad (9)$$

$$A^{n+1} e^{ik} = (1 - \lambda\Delta t) A^n e^{ik} \qquad (10)$$

$$A^n A e^{ik} = A^n e^{ik} (1 - \lambda\Delta t) \qquad (11)$$

$$A = (1 - \lambda\Delta t) \qquad (12)$$

A é chamado de fator de amplificação. Então $|A| \leq 1$

$$|1 - \lambda\Delta t| \leq 1$$

$$1 - \lambda\Delta t \leq 1$$
$$-\lambda\Delta t \leq 1 - 1$$
$$-\lambda\Delta t \leq 0$$
$$\Delta t \geq 0$$

$$1 - \lambda\Delta t \geq -1$$
$$-\lambda\Delta t \geq -1 - 1$$
$$-\Delta t \geq \frac{-2}{\lambda}$$
$$\Delta t \leq \frac{2}{\lambda}$$

Análise de Estabilidade EDO linear

Assume-se que a solução para a eq. 7 tem a forma $y^n \approx A^n$ e substitua no Esquema Avançado Eq. 9

$$A = (1 - \lambda \Delta t) \qquad (12)$$

O sistema é estável, se

$\Delta t \geq 0$ ⟵ $|1 - \lambda \Delta t| \leq 1$ ⟶ $\Delta t \leq \frac{2}{\lambda}$

N.B. Para uma equação não linear, a análise de estabilidade da forma linearizada da equação não linear é uma condição necessária, mas não é suficiente.

Métodos de diferenças finitas.

```
MODULE Class_NumericalScheme
  IMPLICIT NONE
  PRIVATE
  INTEGER, PUBLIC     , PARAMETER :: r8=8
  INTEGER, PUBLIC     , PARAMETER :: r4=4

  REAL (KIND=r8)                  :: dt
  REAL (KIND=r8)                  :: lambda
  INTEGER           , PARAMETER :: UnitData=1
  INTEGER           , PARAMETER :: UnitCtl=2
  CHARACTER (LEN=400)             :: FileName
  LOGICAL                         :: CtrlWriteDataFile
  PUBLIC :: InitNumericalScheme
  PUBLIC :: SchemeForward
  PUBLIC :: SchemeUpdate
  PUBLIC :: SchemeWriteData
  PUBLIC :: SchemeWriteCtl
  PUBLIC :: AnaliticFunction

CONTAINS

 SUBROUTINE InitNumericalScheme(dt_in,lambda_in)
    IMPLICIT NONE
   REAL (KIND=r8) :: dt_in
   REAL (KIND=r8) :: lambda_in
   FileName=''
   dt=dt_in
   lambda=lambda_in
   FileName='EDOL'
   CtrlWriteDataFile=.TRUE.
 END SUBROUTINE InitNumericalScheme
```

```
FUNCTION SchemeForward(yc)  RESULT(yp)
   IMPLICIT NONE
   ! Utilizando a diferenciacao forward
   !
   ! y(n+1) - y(n)
   !--------------- = lambda * y(n) ; onde lambda > 0
   !       dt
   !
   REAL (KIND=r8), INTENT (IN   ) :: yc
   REAL (KIND=r8) :: yp

   yp =  yc -lambda*dt*yc

 END FUNCTION SchemeForward

 FUNCTION SchemeUpdate(y_in)  RESULT (y_out)
   IMPLICIT NONE
   REAL (KIND=r8), INTENT(IN) :: y_in
   REAL (KIND=r8)             :: y_out
   y_out=y_in
 END FUNCTION SchemeUpdate


 FUNCTION AnaliticFunction(y0,tn)  RESULT (y_out)
   IMPLICIT NONE
   REAL (KIND=r8), INTENT (IN) :: y0
   INTEGER      ,      INTENT (IN) :: tn

   REAL (KIND=r8)             :: y_out
   y_out=y0*exp(-lambda*tn*dt)
 END FUNCTION AnaliticFunction
```

Métodos de diferenças finitas.

```
FUNCTION SchemeWriteData(irec,y_in,ya)  RESULT (ok)
   IMPLICIT NONE
   INTEGER   , INTENT (INOUT) :: irec
   REAL (KIND=r8), INTENT (IN) :: y_in
   REAL (KIND=r8), INTENT (IN) :: ya
   INTEGER            :: ok
   INTEGER            :: lrec
   REAL (KIND=r4)       :: Yout
   INQUIRE (IOLENGTH=lrec) Yout

IF(CtrlWriteDataFile)OPEN(UnitData,FILE=TRIM(FileName)//'.bin',&
FORM='UNFORMATTED', ACCESS='DIRECT',
STATUS='UNKNOWN', &
ACTION='WRITE',RECL=lrec)
   CtrlWriteDataFile=.FALSE.
   Yout=REAL(y_in,KIND=r4)
   irec=irec+1
   WRITE(UnitData,rec=irec) Yout
   irec=irec+1
   WRITE(UnitData,rec=irec) REAL (ya,KIND=r4)
   ok=0
 END FUNCTION SchemeWriteData
```

```
 FUNCTION SchemeWriteCtl(nrec)  RESULT( ok)
   IMPLICIT NONE
   INTEGER, INTENT (IN) :: nrec
   INTEGER            :: ok

   OPEN(UnitCtl,FILE=TRIM(FileName)//'.ctl',FORM='FORMATTED',&
   ACCESS='SEQUENTIAL',STATUS='UNKNOWN',ACTION='WRITE')
   WRITE (UnitCtl,'(A6,A)')'dset ^',TRIM(FileName)//'.bin'
   WRITE (UnitCtl,'(A   )')'title  EDO'
   WRITE (UnitCtl,'(A   )')'undef  -9999.9'
   WRITE (UnitCtl,'(A   )')'xdef  1 linear -48.00 1'
   WRITE (UnitCtl,'(A   )')'ydef  1 linear  -1.27 1'
   WRITE (UnitCtl,'(A6,I6,A25)')'tdef  ',nrec,' linear  00z01jan0001 1hr'
   WRITE (UnitCtl,'(A20 )')'zdef  1 levels 1000 '
   WRITE (UnitCtl,'(A)')'vars 2'
   WRITE (UnitCtl,'(A)')'yc 0 99 resultado da edol yc'
   WRITE (UnitCtl,'(A)')'ya 0 99 funcao analitica'
   WRITE (UnitCtl,'(A)')'endvars'
   CLOSE (UnitCtl,STATUS='KEEP')
   CLOSE (UnitData,STATUS='KEEP')

   ok=0
 END FUNCTION SchemeWriteCtl

END MODULE Class_NumericalScheme
```

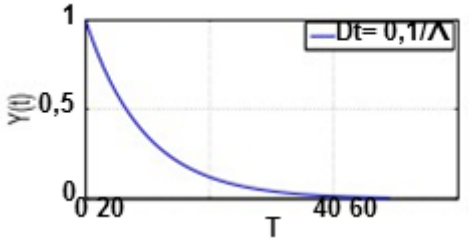
Métodos de diferenças finitas.

```
PROGRAM MAIN
 USE Class_NumericalScheme, Only :r8, InitNumericalScheme, SchemeForward, SchemeUpdate,&
                          SchemeWriteCtl, SchemeWriteData, AnaliticFunction
 IMPLICIT NONE
 REAL (KIND=r8) :: yn
 REAL (KIND=r8) :: yc
 REAL (KIND=r8) :: yp
 REAL (KIND=r8) :: ya

 REAL (KIND=r8), PARAMETER :: lambda=0.1 ![1/s]
 REAL (KIND=r8), PARAMETER ::  y0=1.0_r8
 INTEGER     , PARAMETER ::  nrec=200
 REAL(KIND=r8) :: dt=1.5/lambda
 INTEGER      :: test
 INTEGER      :: irec
 INTEGER      :: tn

 CALL Init()
 yc=y0
irec=0
 DO tn=0,nrec
   yp=SchemeForward(yc)
   ya=AnaliticFunction(y0,tn)
   test=SchemeWriteData(irec,yc,ya)
   yn=SchemeUpdate(yc)
   yc=SchemeUpdate(yp)
 END DO
 test=SchemeWriteCtl(nrec)

CONTAINS

  SUBROUTINE Init()
    IMPLICIT NONE
    CALL InitNumericalScheme(dt,lambda)
  END SUBROUTINE Init
END PROGRAM MAIN
```

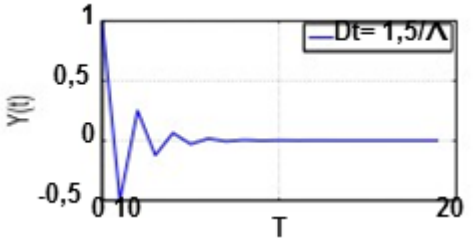
Métodos de diferenças finitas.

Há três possibilidade para a solução numérica ou seja $y^{n+1} = (1 - \lambda\Delta t)y^n$ depende a escolha de Δt

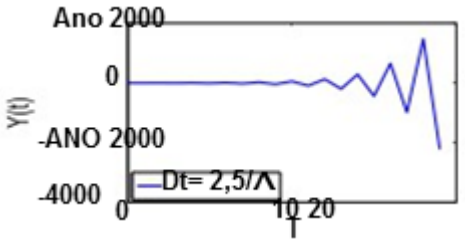
I) Se $\Delta t < \frac{1}{\lambda}$, então $0 < (1 - \lambda\Delta t) < 1$, A solução numérica é uma função crescente com o tempo



II) Se $\frac{1}{\lambda} < \Delta t < \frac{2}{\lambda}$, A solução numérica diminui a magnitude mas oscila no sinal O sistema é estável , mas não consistente.



(III) se $\Delta t > \frac{2}{\lambda}$ então, $(1 - \lambda\Delta t) < -1$,. A solução oscila no Sinal e aumenta em magnitude com o passar do tempo. O método é Instável para as grandes passo de tempo.



Métodos de diferenças finitas.

A restrição de Δt para garantir a estabilidade pode por vezes *ser removido* pela escolha do método numérico. ex. usando o esquema atrasado.

$$\frac{dy}{dt} = -\lambda y$$

$$\frac{y^n - y^{n-1}}{\Delta t} = -\lambda y^n$$

$$A^n - A^{n-1} = -\lambda \Delta t A^n$$

$$A^n - A^n A^{-1} = -\lambda \Delta t A^n$$

$$1 - A^{-1} = -\lambda \Delta t$$

$$A^{-1} = 1 + \lambda \Delta t$$

$$A = \frac{1}{1 + \lambda \Delta t}$$

O sistema é estável, se |A| ≤ 1

$$|A| = \left| \frac{1}{1 + \lambda \Delta t} \right| \leq 1$$

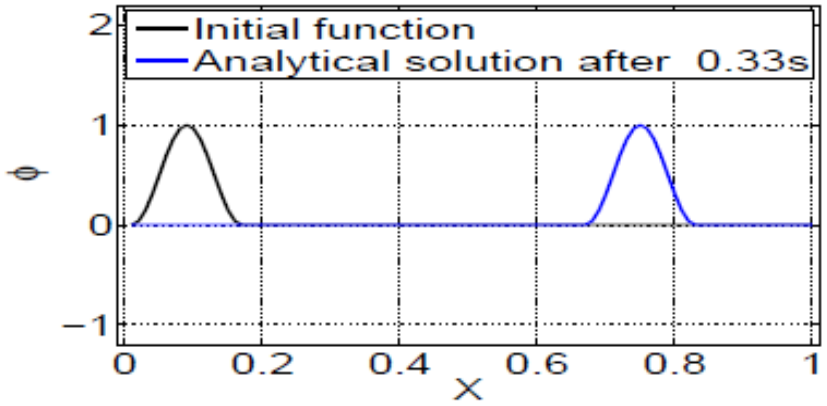
Métodos de diferenças finitas.

Exemplo: Equações diferenciais Parciais

Considere uma equação de Advecção Linear em uma dimensão

$$\frac{\partial \phi}{\partial t} + u\frac{\partial \phi}{\partial x} = 0 \qquad (10)$$

Onde u é a velocidade constante, o domínio é $0 \leq x \leq 1$, com condição de fronteira periódicas $\phi(x,0) = \phi(1,t)$, E a condição inicial é dada como $\phi(x,0) = F(x)$. Por exemplo $F(x) = \sin^2(x)$



A solução analítica é

$$\boldsymbol{\phi(x,0) = F(x - u * t).}$$

A função inicial é advectada com velocidade u, e a forma é preservada.

Métodos de diferenças finitas.

```
MODULE Class_Fields
  IMPLICIT NONE
  PRIVATE
  INTEGER, PUBLIC     , PARAMETER  :: r8=8
  INTEGER, PUBLIC     , PARAMETER  :: r4=4
  REAL (KIND=r8),PUBLIC ,ALLOCATABLE :: PHI_P(:)
  REAL (KIND=r8),PUBLIC ,ALLOCATABLE :: PHI_A(:)
  REAL (KIND=r8),PUBLIC ,ALLOCATABLE :: PHI_C(:)
  REAL (KIND=r8),PUBLIC ,ALLOCATABLE :: PHI_M(:)
  REAL (KIND=r8),PUBLIC              :: Uvel
  INTEGER     ,PUBLIC                :: iMax
  PUBLIC :: Init_Class_Fields
CONTAINS
!-------------------------------------------------------------------------------
  SUBROUTINE Init_Class_Fields(xdim,Uvel0)
    IMPLICIT NONE
    INTEGER     , INTENT (IN   ) :: xdim
    REAL (KIND=r8), INTENT (IN   ):: Uvel0
    iMax=xdim
    Uvel=Uvel0
    ALLOCATE (PHI_A(-1:iMax+2))
    ALLOCATE (PHI_P(-1:iMax+2))
    ALLOCATE (PHI_C(-1:iMax+2))
    ALLOCATE (PHI_M(-1:iMax+2))
  END SUBROUTINE Init_Class_Fields
!-------------------------------------------------------------------------------
END MODULE Class_Fields
```

```
MODULE Class_NumericalMethod
  USE Class_Fields, Only : PHI_A,PHI_P,PHI_C,PHI_M,Uvel,iMax
 IMPLICIT NONE
 PRIVATE
 INTEGER, PUBLIC     , PARAMETER :: r8=8
 INTEGER, PUBLIC     , PARAMETER :: r4=4
 REAL (KIND=r8) :: Dt
 REAL (KIND=r8) :: Dx

 PUBLIC :: InitNumericalScheme
 PUBLIC :: SchemeForward
 PUBLIC :: SchemeUpdate
 PUBLIC :: SchemeUpStream
 PUBLIC :: AnaliticFunction

CONTAINS
!-------------------------------------------------------------------------------
 SUBROUTINE InitNumericalScheme(dt_in,dx_in)
  IMPLICIT NONE
  REAL (KIND=r8), INTENT (IN   ) :: dt_in
  REAL (KIND=r8), INTENT (IN   ) :: dx_in
  INTEGER :: i
  REAL (KIND=r8) :: x
  Dt=dt_in
  Dx=dx_in
  x=0.0
  DO i=1,iMax
     PHI_C(i)= sin(x)*sin(x)
     x =  (6*i)* DX
  END DO
  PHI_M=PHI_C
  PHI_P=PHI_C
 END SUBROUTINE InitNumericalScheme
```

Métodos de diferenças finitas.

```
FUNCTION AnaliticFunction(tn)  RESULT (ok)
  IMPLICIT NONE
  INTEGER,      INTENT (IN) :: tn
  INTEGER      :: j,ok
  REAL (KIND=r8) :: x
  ok=1
  x=0.0
  DO j=1,iMax
    PHI_A(j)=(sin(x - Uvel*tn*(dt))**2)
    x =  (6*j)* DX
  END DO
  ok=0
END FUNCTION AnaliticFunction

!-------------------------------------------------------------------------------------
FUNCTION SchemeForward()  RESULT(ok)
  IMPLICIT NONE
  ! Utilizando a diferenciacao forward
  !
  ! F(j,n+1) - F(j,n)       F(j+1,n) - F(j,n)
  !------------------- + u ------------------- = 0
  !      dt                  dx
  !
  INTEGER :: ok
  INTEGER :: j
  DO j=1,iMax
    PHI_P(j) = PHI_C(j) - (Uvel*Dt/Dx)*(PHI_C(j+1)-PHI_C(j))
  END DO
  CALL UpdateBoundaryLayer()
END FUNCTION SchemeForward
```

```
!-------------------------------------------------------------------------------------
FUNCTION SchemeUpStream()  RESULT(ok)
  IMPLICIT NONE
  ! Utilizando a diferenciacao forward no tempo e
  ! backward no espaco (upstream)
  !
  ! F(j,n+1) - F(j,n)       F(j,n) - F(j-1,n)
  !------------------- + u ------------------- = 0
  !      dt                  dx
  !
  INTEGER :: ok
  INTEGER :: j
  DO j=1,iMax
    PHI_P(j) = PHI_C(j) - (Uvel*Dt/Dx)*(PHI_C(j)-PHI_C(j-1))
  END DO
  CALL UpdateBoundaryLayer()
END FUNCTION SchemeUpStream
!-------------------------------------------------------------------------------------
SUBROUTINE UpdateBoundaryLayer()
  IMPLICIT NONE
  PHI_P(0          )    =  PHI_P(iMax )
  PHI_P(-1         )    =  PHI_P(iMax-1)
  PHI_P(imax+1 )    =   PHI_P(1)
  PHI_P(iMax+2 )    =   PHI_P(2)
END  SUBROUTINE UpdateBoundaryLayer
!-------------------------------------------------------------------------------------
FUNCTION SchemeUpdate()  RESULT (ok)
  IMPLICIT NONE
  INTEGER :: ok
  PHI_M=PHI_C
  PHI_C=PHI_P
  ok=0
END FUNCTION SchemeUpdate
!-------------------------------------------------------------------------------------
END MODULE Class_NumericalMethod
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

Métodos de diferenças finitas.

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
MODULE Class_WritetoGrads
 USE Class_Fields, Only: PHI_A,PHI_P,PHI_C,PHI_M,Uvel,iMax
 IMPLICIT NONE
 PRIVATE
 INTEGER, PUBLIC      , PARAMETER  :: r8=8
 INTEGER, PUBLIC      , PARAMETER  :: r4=4
 INTEGER              , PARAMETER :: UnitData=1
 INTEGER              , PARAMETER :: UnitCtl=2
 CHARACTER (LEN=400)               :: FileName
 LOGICAL                           :: CtrlWriteDataFile
 PUBLIC :: SchemeWriteCtl
 PUBLIC :: SchemeWriteData
 PUBLIC :: InitClass_WritetoGrads
CONTAINS
 SUBROUTINE InitClass_WritetoGrads()
   IMPLICIT NONE
  FileName=''
  FileName='AdvecLinearConceitual1D'
  CtrlWriteDataFile=.TRUE.
 END SUBROUTINE InitClass_WritetoGrads

 FUNCTION SchemeWriteData(irec)  RESULT (ok)
   IMPLICIT NONE
   INTEGER   , INTENT (INOUT) :: irec
   INTEGER                    :: ok
   INTEGER                    :: lrec
   REAL (KIND=r4)             :: Yout(iMax)
   INQUIRE (IOLENGTH=lrec) Yout
   IF(CtrlWriteDataFile)OPEN(UnitData,FILE=TRIM(FileName)//'.bin',&
   FORM='UNFORMATTED', ACCESS='DIRECT',
STATUS='UNKNOWN', &
   ACTION='WRITE',RECL=lrec)
  ok=1
   CtrlWriteDataFile=.FALSE.
   Yout=REAL(PHI_C(1:iMax),KIND=r4)
   irec=irec+1
   WRITE(UnitData,rec=irec)Yout
```

```
  Yout=REAL(PHI_A(1:iMax),KIND=r4)
   irec=irec+1
   WRITE(UnitData,rec=irec)Yout
  ok=0
 END FUNCTION SchemeWriteData

 FUNCTION SchemeWriteCtl(nrec)  RESULT (ok)
   IMPLICIT NONE
   INTEGER, INTENT (IN) :: nrec
   INTEGER              :: ok
   ok=1
  OPEN(UnitCtl,FILE=TRIM(FileName)//'.ctl',FORM='FORMATTED', &
  ACCESS='SEQUENTIAL',STATUS='UNKNOWN',ACTION='WRITE')
   WRITE (UnitCtl,'(A6,A          )')'dset ^',TRIM(FileName)//'.bin'
   WRITE (UnitCtl,'(A             )')'title  EDO'
   WRITE (UnitCtl,'(A             )')'undef  -9999.9'
   WRITE (UnitCtl,'(A6,I8,A18   )')'xdef  ',iMax,' linear 0.00 0.001'
   WRITE (UnitCtl,'(A              )')'ydef  1 linear  -1.27 1'
   WRITE (UnitCtl,'(A6,I6,A25   )')'tdef  ',nrec,' linear  00z01jan0001 1hr'
   WRITE (UnitCtl,'(A20           )')'zdef  1 levels 1000 '
   WRITE (UnitCtl,'(A        )')'vars 2'
   WRITE (UnitCtl,'(A        )')'phic 0 99 resultado da edol yc'
   WRITE (UnitCtl,'(A        )')'phia 0 99 solucao analitica ya'
   WRITE (UnitCtl,'(A        )')'endvars'
   CLOSE (UnitCtl,STATUS='KEEP')
   CLOSE (UnitData,STATUS='KEEP')
   ok=0
 END FUNCTION SchemeWriteCtl
END MODULE Class_WritetoGrads
```

Métodos de diferenças finitas.

```
PROGRAM Main
 USE Class_Fields, Only:Init_Class_Fields
 USE Class_NumericalMethod, Only : InitNumericalScheme, &
       SchemeForward,  SchemeUpdate, SchemeUpStream,&
       AnaliticFunction
 USE Class_WritetoGrads, Only :InitClass_WritetoGrads, &
                        SchemeWriteData, SchemeWriteCtl
 IMPLICIT NONE
 INTEGER               , PARAMETER  :: r8=8
 INTEGER               , PARAMETER  :: r4=4
 INTEGER               , PARAMETER :: xdim=100
 REAL (KIND=r8)        , PARAMETER :: LX=1.0
 REAL (KIND=r8)        , PARAMETER :: Uvel0=10.0!m/s
 REAL (KIND=r8)        , PARAMETER :: dx=LX/xdim   !m
 REAL (KIND=r8)        , PARAMETER :: dt=0.1*dx/Uvel0   !s       !
c*Dt/Dx < 1
!                                                      ! =>  Dt <
dx/Uvel0
 INTEGER           , PARAMETER :: ninteraction=20000
 CALL Init()
 CALL run()

CONTAINS

 SUBROUTINE Init()
  IMPLICIT NONE
  CALL Init_Class_Fields(xdim,Uvel0)
  CALL InitNumericalScheme(dt,dx)
  CALL InitClass_WritetoGrads
 END SUBROUTINE Init
```

```
SUBROUTINE Run()
  IMPLICIT NONE
  INTEGER :: test,tn,irec
  irec=0
  DO tn=0,ninteraction
    test=SchemeUpStream()
    test=AnaliticFunction(tn)
    test=SchemeWriteData(irec)
    test=SchemeUpdate()
  END DO
  test=SchemeWriteCtl(ninteraction)
 END SUBROUTINE Run

 SUBROUTINE Finalize()
  IMPLICIT NONE

 END SUBROUTINE Finalize

END PROGRAM Main
```

Métodos de diferenças finitas.

Esquemas com diferenças centradas no espaço de quarta ordem:

$$\frac{u_{j+1}-u_{j-1}}{2\Delta x}=\frac{\partial u}{\partial x}+\frac{1}{3!}\frac{\partial^3 u}{\partial x^3}(\Delta x)^2+O[(\Delta x)^4]$$

O Quociente é de segunda ordem de acurácia. É formado considerando a diferença de valores de u_j em um ponto de grade distante do ponto central. Simularmente um quociente pode ser construído considerando a diferença de dois pontos distante do ponto central. Susbstituindo Δx por $2\Delta x$

$$\frac{u_{j+2}-u_{j-2}}{4\Delta x}=\frac{\partial u}{\partial x}+\frac{4}{3!}\frac{\partial^3 u}{\partial x^3}(\Delta x)^2+O[(\Delta x)^4]$$

O Quociente é de segunda ordem de acurácia. Mas os coeficientes são grandes. Outra aproximação consistente para a derivada espacial pode ser construída pelo combinação linear das duas equações acima. A combinação para o termo de segunda ordem no erro de truncamento é cancelada

$$\frac{4}{3}\frac{u_{j+1}-u_{j-1}}{2\Delta x}-\frac{1}{3}\frac{u_{j+2}-u_{j-2}}{4\Delta x}=\frac{\partial u}{\partial x}+O[(\Delta x)^4]$$

Métodos de diferenças finitas.

Esquemas com diferenças centradas no espaço de quarta ordem:

$$\frac{\partial u}{\partial x} = \frac{4}{3}\frac{u_{j+1} - u_{j-1}}{2\Delta x} - \frac{1}{3}\frac{u_{j+2} - u_{j-2}}{4\Delta x} - O[(\Delta x)^4]$$

Por Exemplo: Resolvendo a equação de advecção linear

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x}$$

Na solução analítica o estado inicial de propaga no espaço com velocidade de fase constante c sem mudança de forma. A solução numérica, entretanto, ficará atrasada em relação a analítica e dispersa. A alta ordem de acurácia o atraso será menor (a diferença de fase será menor).

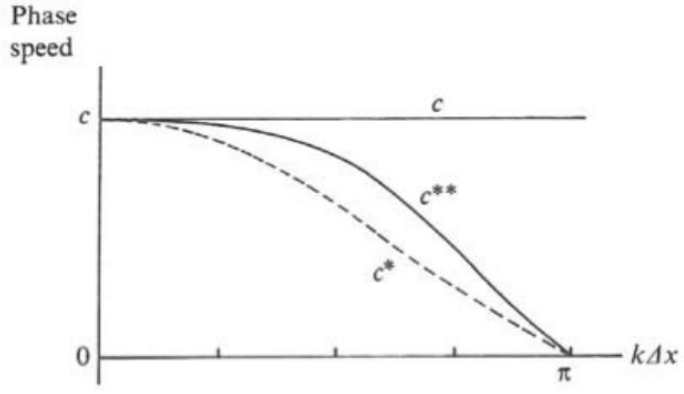
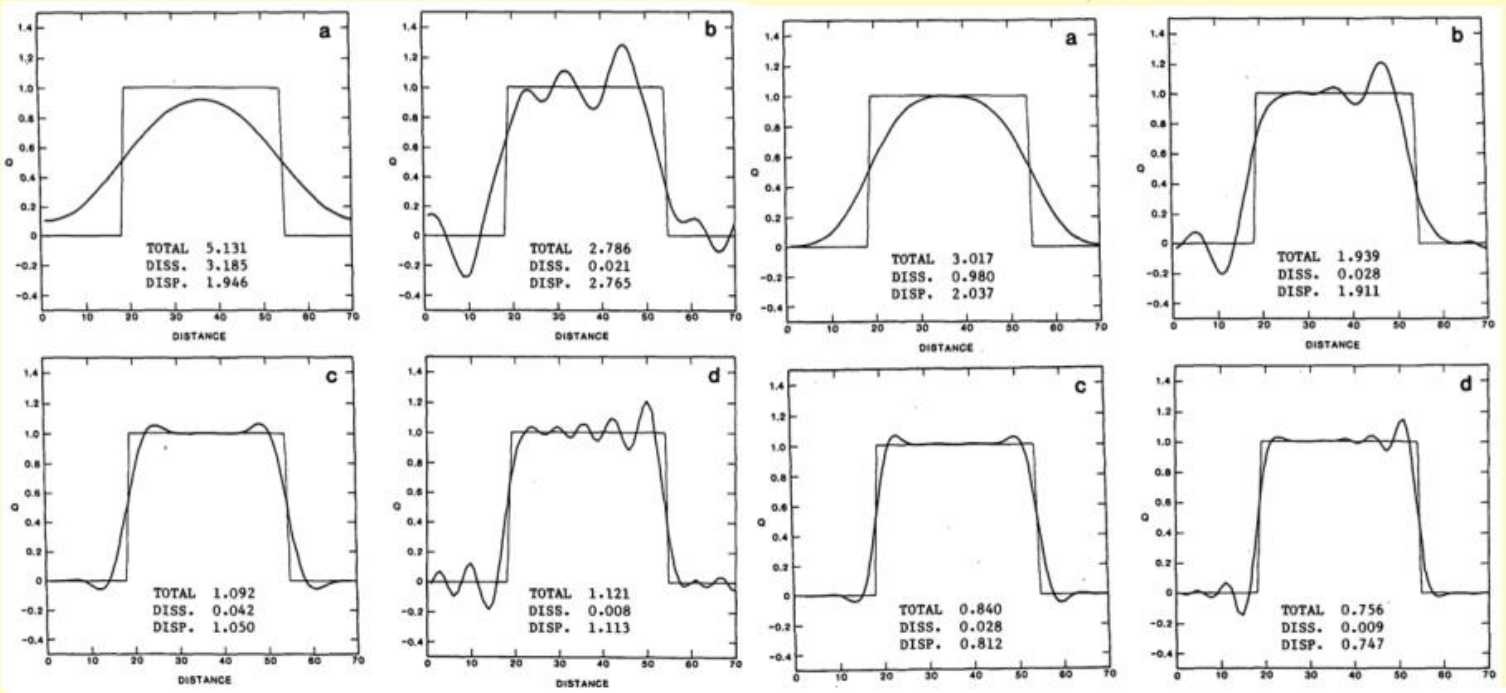


Figure 4.1 Phase speed for the linear advection equation, c, and for the corresponding differential-difference equations with second order (c^*) and with fourth order (c^{**}) centered space differencing.

Métodos de diferenças finitas.

O Aumento da ordem de acurácia não resolve o problema de dispersão que envolve a solução numérica (Takacs,1985)



Numerical solutions for (a) 1st-order scheme, (b) 2nd-order scheme, (c) 3rd-order scheme, and (d) 4th-order scheme, after two complete translations

Métodos de diferenças finitas.

Exercício 1

Resolva numericamente a equação de advecção utilizando a acurácia de quarta ordem na discretização espacial e compare com a solução de segunda ordem. Considere $\mu = \frac{c\Delta x}{\Delta t} = 0.2$ **e** $\mu = \frac{c\Delta x}{\Delta t} = 0.7$**. Como condição inicial use uma onda quadrada.**

$$\frac{\partial u}{\partial t} = -c\frac{\partial u}{\partial x}$$

$$\frac{\partial u}{\partial x} = \frac{4}{3}\frac{u_{j+1} - u_{j-1}}{2\Delta x} - \frac{1}{3}\frac{u_{j+2} - u_{j-2}}{4\Delta x} - O[(\Delta x)^4]$$