Métodos de diferenças finitas.

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)

Esquema implícito: FTBS

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.

Esquema implícito: FTBS

- ✓ 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.

Esquema implícito: FTBS

Esquema implícito: FTBS

Esquemas Implícitos e Semi-Implícitos

Esquema implícito: FTBS

O passo de tempo Δt permitido pelos esquemas explícitos básico (leapfrog), é duas vezes aquele que satisfaz o critério CFL $\left(0 \leq c\frac{\Delta t}{\Delta x} \leq 1\right)$, é consideravelmente muito longo em relação ao passo de tempo Δt necessário para a integração precisa das equações quasi-geostróficas, que não permitem ondas de oscilação rápida.

Assim, consideraremos aqui esquemas implícitos, que têm a agradável propriedade de serem estáveis para qualquer escolha de passo de tempo.

Esquema implícito: FTBS

Esquemas Implícitos versus Explícitos, um exemplo simples

Para esquemas implícitos, **os termos espaciais são avaliados**, pelo menos parcialmente, no nível de tempo desconhecido.

Vamos considerar um dos exemplos mais simples possíveis, examinando a equação de difusão unidimensional, também conhecida como equação do calor/momentum

$$\frac{\partial u}{\partial t} = A\frac{\partial^2 u}{\partial x^2} \qquad \frac{u_j^{n+1} - u_j^n}{\Delta t} = A\frac{u_{j-1}^n - 2u_j^n + u_{j+1}^n}{\Delta x^2}$$

Uma **solução formal desta equação diferencial parabólica** requer uma **condição inicial**, bem como **duas condições de contorno**, estas últimas, para não complicar nosso problema desnecessariamente, aqui são tomadas como condições de Dirichlet.

Esquema implícito: FTBS

Esquemas Implícitos versus Explícitos, um exemplo simples

"Esta equação é discretizada com diferenças finitas espaciais centradas e integrada no tempo com um esquema de Euler para frente. Na forma explícita tradicional, obtemos"

$$u_j^{n+1} = u_j^n + A\frac{\Delta t}{\Delta x^2}\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right)$$

$$\boxed{\nu = A\frac{\Delta t}{\Delta x^2}}$$

$$u_j^{n+1} = u_j^n + \nu\left(u_{j-1}^n - 2u_j^n + u_{j+1}^n\right)$$

$$u_j^{n+1} = u_j^n + \nu u_{j-1}^n - 2\nu u_j^n + \nu u_{j+1}^n$$

$$u_j^{n+1} = \nu u_{j-1}^n + u_j^n - 2\nu u_j^n + \nu u_{j+1}^n$$

que pode ser reescrito como

$$u_j^{n+1} = \nu u_{j-1}^n + (1 - 2\nu)u_j^n + \nu u_{j+1}^n$$

onde $\nu = A\frac{\Delta t}{\Delta x^2}$ é, como anteriormente, o número de von Neumann

Esquema implícito: FTBS

Esquemas Implícitos versus Explícitos, um exemplo simples

$$u_j^{n+1} = \nu u_{j-1}^n + (1-2\nu)u_j^n + \nu u_{j+1}^n$$

Esta equação é explícita em termos de u_j^{n+1} , que é o valor no nível de tempo desconhecido $n+1$, e, portanto, é possível de resolver.

Uma análise de estabilidade pode ser realizada e pode ser mostrada que é condicionalmente estável $(-1 \leq \lambda \leq 1)$ para $\nu \leq \frac{1}{2}$.

Uma condição mais restrita, com apenas uma raiz positiva $(0 \leq \lambda \leq 1)$ para a solução não oscilatória, é obtida para $\nu \leq \frac{1}{4}$.

Esquema implícito: FTBS

Esquemas Implícitos versus Explícitos, um exemplo simples

$$u_j^{n+1} = u_j^n + A\frac{\Delta t}{\Delta x^2}\left(u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}\right)$$

$$\nu = A\frac{\Delta t}{\Delta x^2}$$

$$u_j^{n+1} = u_j^n + \nu\left(u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+1}\right)$$

Umas aproximação similar, ***mas resolvendo o termo espacial*** no nível de tempo desconhecido n+1, produzirá uma ***Discretização completamente implícita***

$$u_j^{n+1} = u_j^n + \nu\left(u_{j-1}^{n+1} - 2u_j^{n+1} + u_{j+1}^{n+!}\right)$$

$$u_j^{n+1} = u_j^n + \ \nu u_{j-1}^{n+1} + 2\nu u_j^{n+1} + \nu u_{j+1}^{n+1}$$

$$-\nu u_{j-1}^{n+1} + u_j^{n+1} + 2\nu u_j^{n+1} - \nu u_{j+1}^{n+1} = u_j^n$$

$$-\nu u_{j-1}^{n+1} + (1 + 2\nu)u_j^{n+1} - \nu u_{j+1}^{n+1} = u_j^n$$

Esquema implícito: FTBS

Esquemas Implícitos versus Explícitos, um exemplo simples

$$-\nu u_{j-1}^{n+1} + (1+2\nu)u_j^{n+1} - \nu u_{j+1}^{n+1} = u_j^n$$

Esta **discretização implícita é incondicionalmente estável**. Para resolver a equação, é necessário considerar todos os pontos da grade j.

Esquema implícito: FTBS

Esquemas Implícitos versus Explícitos, um exemplo simples

$$-\nu u_{j-1}^{n+1} + (1+2\nu)u_j^{n+1} - \nu u_{j+1}^{n+1} = u_j^n$$

No presente caso, quando **estamos lidando com a equação do calor linearizada**, **o** **problema pode ser expresso como um sistema linear de equações** $A\vec{X} = \vec{B}$, onde A é uma matriz, $\vec{X}$ é um vetor dado pelos valores desconhecidos de u no tempo $n+1$, e $\vec{B}$ é um vetor dado pelos valores conhecidos de u:"

$$A\vec{X} = \vec{B},$$

$$\begin{bmatrix} (1+2\nu) & -\nu & & \\ -\nu & (1+2\nu) & -\nu & \\ \cdots & \cdots & \cdots & \\ & \cdots & \cdots & \cdots \\ & -\nu & (1+2\nu) & -\nu \\ & \cdots & \cdots & \cdots \\ & & \cdots & \cdots \\ & & -\nu & (1+2\nu) \end{bmatrix} \begin{bmatrix} u_2^{n+1} \\ u_3^{n+1} \\ \cdots \\ \cdots \\ u_i^{n+1} \\ \cdots \\ \cdots \\ u_{I-1}^{n+1} \end{bmatrix} = \begin{bmatrix} u_2^n + \nu u_1^{n+1} \\ u_3^n \\ \cdots \\ \cdots \\ u_i^n \\ \cdots \\ \cdots \\ u_{I-1}^n + \nu u_I^{n+1} \end{bmatrix}.$$

Esquema implícito: FTBS

Esquemas Implícitos versus Explícitos, um exemplo simples

Por razões didáticas, consideramos como u_1^{n+1} e u_j^{n+1} conhecidos a partir das **condições de contorno de Dirichlet**.

As *condições de Neumann e Cauchy também podem ser aplicadas*, mas são um pouco mais complicadas de implementar.

A ***solução no nível de tempo*** $\boldsymbol{n+1}$ é determinada resolvendo este ***sistema de equações***.

"O método implícito é, consequentemente, *muito exigente do ponto de vista computacional* em comparação com o método explícito, mas como é incondicionalmente estável, é possível usar passos de tempo maiores.

No caso presente, **a matriz é tridiagonal, o que é vantajoso do ponto de vista computacional**, uma vez que o problema pode ser resolvido usando, por exemplo, o algoritmo de Thomas, uma versão simplificada da eliminação de Gauss.

Esquema implícito: FTBS

Esquemas Semi Implícitos

Esquemas semi-implícitos avaliam a derivada espacial em uma média dos níveis de tempo n e $n + 1$, em vez de apenas em $n + 1$ como no caso totalmente implícito.

Se $F(x, y, t)$ é um termo que compreende derivadas espaciais de um dado escalar $T(x, y, t)$, pode-se considerar a expressão geral para uma versão discretizada da equação para a evolução temporal de u_j^n

$$\frac{\partial u}{\partial t} = F(x, y) \Rightarrow \frac{u_j^{n+1} - u_j^n}{\Delta t} = (1 - \beta)F_{i,j}^n + (\beta)F_{i,j}^{n+1}$$

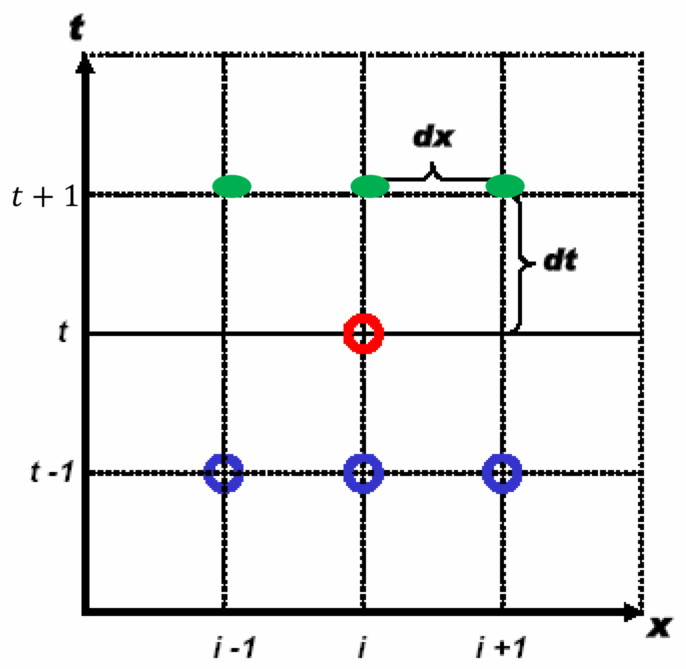
onde $\beta = 0$ resulta em um esquema explícito, $\beta = 1$ em um esquema totalmente implícito e 0< $\beta < 1$ em um esquema semi-implícito.

Esquema implícito: FTBS

Esquemas Semi Implícitos

Um método semi-implícito comumente usado é dado pelo esquema de Crank-Nicolson, no qual $\beta = 0.5$ e a derivada temporal é expressa com o esquema usual de Euler avançado. O termo que compreende as derivadas espaciais está, portanto, centrado no nível de tempo $\mathrm{n} + \frac{1}{2}$ o que, de fato, transforma este esquema em um esquema trapezoidal implícito no tempo.

$$\frac{\partial u}{\partial t} = F(x,y) \Rightarrow \frac{u_j^{n+1} - u_j^n}{\Delta t} = (1-\beta)F_{i,j}^n + (\beta)F_{i,j}^{n+1}$$



Ao realizar uma expansão de Taylor em torno de ($\mathrm{i}, \mathrm{n} + \frac{1}{2}$), pode-se verificar que este esquema implícito é caracterizado por uma precisão de segunda ordem no tempo, o que representa uma melhoria apreciável em relação à precisão de primeira ordem do esquema explícito de Euler avançado.

Esquema implícito: FTBS

Esquemas Semi Implícitos *A equação de difusão unidimensional (1D)*

A equação de difusão (ex: equação de difusão de calor) é geralmente associada à diferenciação centrada no espaço. Quando se usa um esquema de tempo semi-implícito, ela se torna

$$\frac{\partial u}{\partial t} = F(x,y) \Rightarrow \frac{u_j^{n+1} - u_j^n}{\Delta t} = (1-\beta)F_{i,j}^n + (\beta)F_{i,j}^{n+1}$$

$$\frac{u_j^{n+1} - u_j^n}{\Delta t} = (1-\beta)\left(A\frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\Delta x^2}\right) + (\beta)\left(A\frac{u_{j+1}^{n+1} - 2u_j^{n+1} + u_{j-1}^{n+1}}{\Delta x^2}\right)$$

O esquema semi-implícito de Crank-Nicolson ($\beta = 0.5$) resulta em uma precisão numérica de segunda ordem tanto no tempo quanto no espaço e, portanto, o erro de truncamento é de $O[(\Delta t)^2, (\Delta x)^2]$

Esquema implícito: FTBS

PDEs Parabólicas: ***Esquemas Explícitos*** para PDEs Parabólicas

Resumo: Solução de EDPs Parabólicas por Esquemas Explícitos

Vantagens:
- cálculos muito fáceis,
- simplesmente fornece um passo à frente n+1

Desvantagem:
– baixa precisão, O (Δt) em relação ao tempo
– sujeito a instabilidade; deve usar "pequenos" Δt’
- requer muitos passos !!!

Esquema implícito: FTBS

PDEs Parabólicas: ***Esquemas Implícitos*** para PDEs Parabólicas

-
Expresso T_i^{n+1} termos de T_j^{n+1} , T_i^n, e possivelmente também T_j^n (em que j = i – 1 e i+1)

- Representa o domínio espacial e temporal. Para cada novo tempo, escreve m (nº de nós interiores) equações e simultaneamente resolve para m valores desconhecidos (sistema com bandas).

Esquema implícito: FTBS

The 1-D Heat Equation: $$\frac{\partial T}{\partial t} = k\frac{\partial^2 T}{\partial x^2}$$

Simple Implicit Method. *Substituting:*

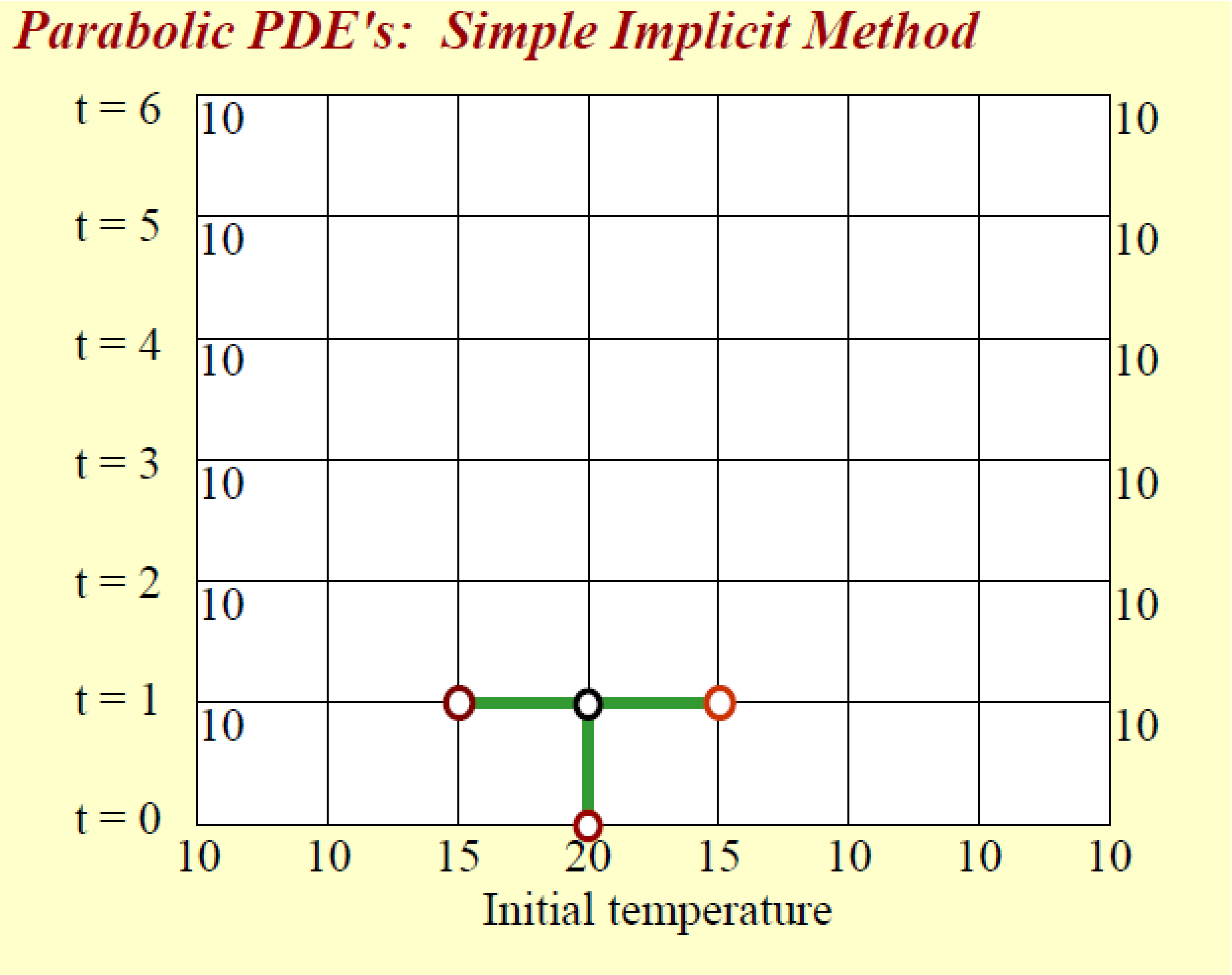
$$\frac{\partial^2 T}{\partial x^2} = \frac{T_{i-1}^{m+1} - 2T_i^{m+1} + T_{i+1}^{m+1}}{(\Delta x)^2} + O(\Delta x)^2 \qquad \text{Centered FDD}$$

$$\frac{\partial T}{\partial t} = \frac{T_i^{m+1} - T_i^m}{\Delta t} + O(\Delta t) \qquad \text{Backward FDD}$$

results in: $-\lambda T_{i-1}^{m+1} + (1+2\lambda)T_i^{m+1} - \lambda T_{i+1}^{m+1} = T_i^m$ with $\lambda = k\frac{\Delta t}{(\Delta x)^2}$

1. Requer I.C. para o caso em que m = 0: ou seja, T_i^m é dado para todo i.
2. Requer B.C.s para escrever n expressões.

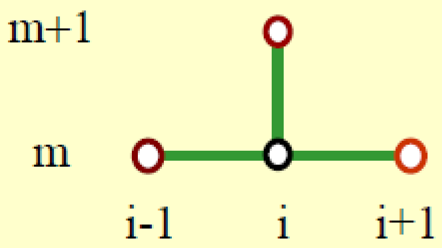
Esquema implícito: FTBS



1. Requer I.C. para o caso em que m = 0: ou seja, T_i^m é dado para todo i.
2. Requer B.C.s para escrever n expressões.

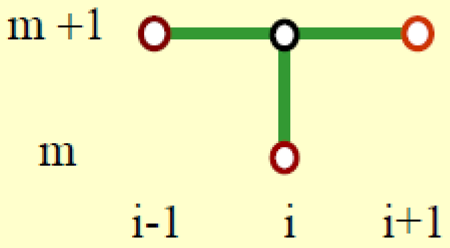
Esquema implícito: FTBS

Explicit Method



$$T_i^{m+1} = T_i^m + \lambda\left(T_{i+1}^m - 2T_i^m + T_{i-1}^m\right)$$

Simple Implicit Method



$$T_i^m = -\lambda T_{i-1}^{m+1} + (1+2\lambda)\, T_i^{m+1} - \lambda T_{i+1}^{m+1}$$

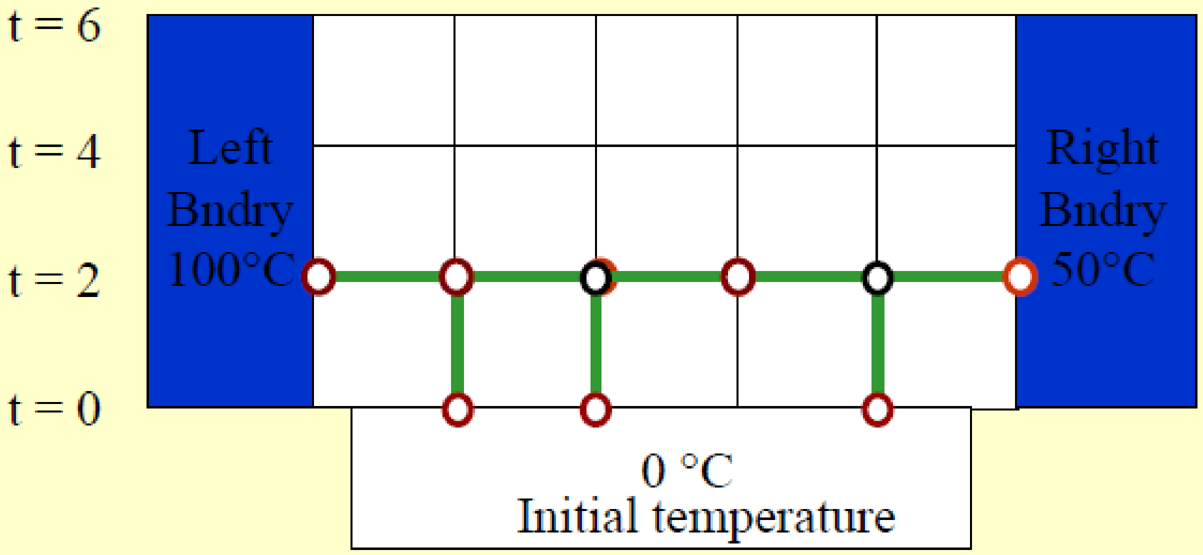
with $\lambda = k\dfrac{\Delta t}{(\Delta x)^2}$ for both

1. Requer I.C. para o caso em que m = 0: ou seja, T_i^m é dado para todo i.
2. Requer B.C.s para escrever n expressões.

Esquema implícito: FTBS

$$-\lambda T_{i-1}^{m+1} + (1+2\lambda)T_i^{m+1} - \lambda T_{i+1}^{m+1} = T_i^m \quad \text{with} \quad \lambda = k\frac{\Delta t}{(\Delta x)^2}$$

Parabolic PDE's: Simple Implicit Method



At the Left boundary: $(1+2\lambda)T_1^{m+1} - \lambda T_2^{m+1} = T_1^{m} + \lambda\, T_0^{m+1}$

Away from boundary: $-\lambda T_{i-1}^{m+1} + (1+2\lambda)T_i^{m+1} - \lambda T_{i+1}^{m+1} = T_i^{m}$

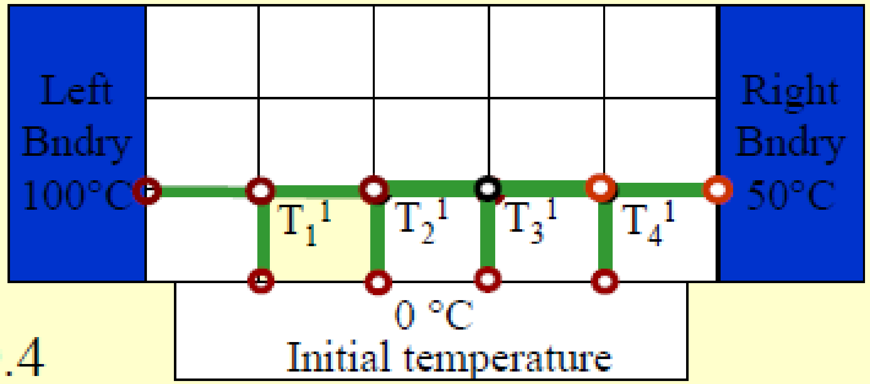
At the Right boundary: $(1+2\lambda)T_i^{m+1} - \lambda T_{i-1}^{m+1} = T_i^{m} + \lambda\, T_{i+1}^{m+1}$

m = 3; t = 6

m = 2; t = 4

m = 1; t = 2

m = 0; t = 0



Let $\lambda = 0.4$

$$\begin{bmatrix}1.8 & -0.4 & 0 & 0\\ -0.4 & 1.8 & -0.4 & 0\\ 0 & -0.4 & 1.8 & -0.4\\ 0 & 0 & -0.4 & 1.8\end{bmatrix}\begin{Bmatrix}T_1^1\\ T_2^1\\ T_3^1\\ T_4^1\end{Bmatrix}=\begin{Bmatrix}40\\ 0\\ 0\\ 20\end{Bmatrix}$$

$$\begin{Bmatrix}T_1^1\\ T_2^1\\ T_3^1\\ T_4^1\end{Bmatrix}=\begin{Bmatrix}27.6\\ 6.14\\ 4.03\\ 12.0\end{Bmatrix}$$

At the Left boundary: $(1+2\lambda)T_1^1 - \lambda T_2^1 = T_1^0 + \lambda T_0^1$

$$1.8\, T_1^1 - 0.4\, T_2^1 = 0 + 0.8*100 = \quad 40$$

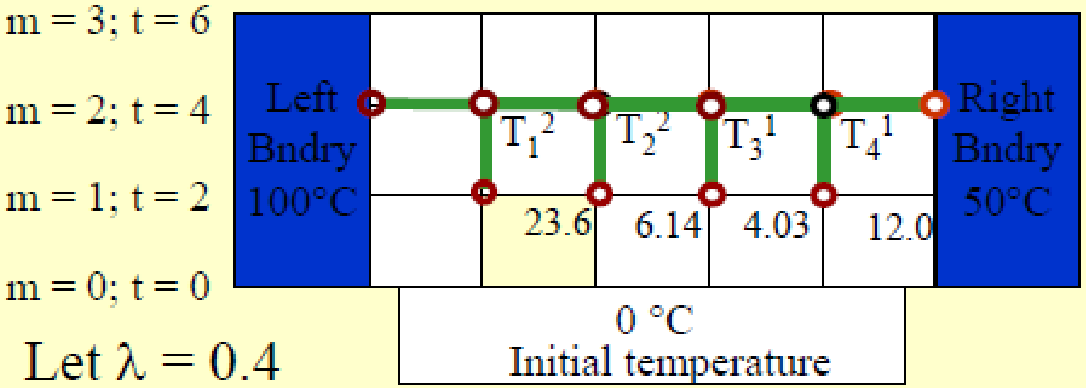
Away from boundary: $-\lambda T_{i-1}^1 + (1+2\lambda)T_i^1 - \lambda T_{i+1}^1 = T_i^0$

$$-0.4\, T_1^1 + 1.8\, T_2^1 - 0.4\, T_3^1 = 0 = \quad 0$$

$$-0.4\, T_2^1 + 1.8\, T_3^1 - 0.4\, T_4^1 = 0 = \quad 0$$

At the Right boundary: $(1+2\lambda)T_3^1 - \lambda T_2^1 = T_3^0 + \lambda T_4^1$

$$1.8\, T_i^{m+1} - 0.4\, T_{i-1}^{m+1} = 0 + 0.4*50) = \quad 20$$



Let $\lambda = 0.4$

$$\begin{bmatrix} 1.8 & -0.4 & 0 & 0 \\ -0.4 & 1.8 & -0.4 & 0 \\ 0 & -0.4 & 1.8 & -0.4 \\ 0 & 0 & -0.4 & 1.8 \end{bmatrix} \begin{Bmatrix} T_1^2 \\ T_2^2 \\ T_3^2 \\ T_4^2 \end{Bmatrix} = \begin{Bmatrix} 78.5 \\ 6.14 \\ 4.03 \\ 32.0 \end{Bmatrix}$$

$$\begin{Bmatrix} T_1^2 \\ T_2^2 \\ T_3^2 \\ T_4^2 \end{Bmatrix} = \begin{Bmatrix} 38.5 \\ 14.1 \\ 9.83 \\ 20.0 \end{Bmatrix}$$

At the Left boundary: $(1+2\lambda)T_1^2 - \lambda T_2^2 = T_1^1 + \lambda T_0^2$

$$1.8\ T_1^2 - 0.4\ T_2^2 = 23.6 + 0.4*100 = \qquad 78.5$$

Away from boundary: $-\lambda T_{i-1}^2 + (1+2\lambda)T_i^2 - \lambda T_{i+1}^2 = T_i^1$

$$-0.4*T_1^2 + 1.8*T_2^2 - 0.4*T_3^2 = 6.14 = \qquad 6.14$$

$$-0.4*T_2^2 + 1.8*T_3^2 - 0.4*T_4^2 = 4.03 = \qquad 4.03$$

At the Right boundary: $(1+2\lambda)T_3^2 - \lambda T_4^2 = T_3^1 + \lambda T_4^2$

$$1.8*T_3^2 - 0.4*T_4^2 = 12.0 + 0.4*50 = \qquad 32.0$$

Analise de estabilidade

Esquema implícito: FTBS

A equação de advecção linear com discretização Backward no tempo e backward no espaço pode ser escrita como:

$$\boxed{\frac{\emptyset_j^{n+1}-\emptyset_j^{n}}{\Delta t}+u\,\frac{\emptyset_j^{n+1}-\emptyset_{j-1}^{n+1}}{\Delta x}=0} \qquad (25)$$

$$\emptyset_j^{n+1}-\emptyset_j^{n}=-u\,\frac{\Delta t}{\Delta x}\left(\emptyset_j^{n+1}-\emptyset_{j-1}^{n+1}\right)$$

$$\emptyset_j^{n+1}=\emptyset_j^{n}-u\,\frac{\Delta t}{\Delta x}\left(\emptyset_j^{n+1}-\emptyset_{j-1}^{n-1}\right)$$

$$\emptyset_j^{n+1}=\emptyset_j^{n}-C\left(\emptyset_j^{n+1}-\emptyset_{j-1}^{n+1}\right)$$

$$\emptyset_j^{n+1}=\emptyset_j^{n}-\ C\emptyset_j^{n+1}+C\emptyset_{j-1}^{n+1}$$

$$\emptyset_j^{n+1}+\ C\emptyset_j^{n+1}=\emptyset_j^{n}+C\emptyset_{j-1}^{n+1}$$

$$(1+C)\emptyset_j^{n+1}=\emptyset_j^{n}+C\emptyset_{j-1}^{n+1}$$

$$\boxed{\emptyset_j^{n+1}=\frac{1}{(1+C)}\left(\emptyset_j^{n}+C\emptyset_{j-1}^{n+1}\right)} \qquad (26)$$

Exercício: mostrar que o fator de amplificação no esquema BTBS É incondicionalmente estável i.e.

$|A|^2 = [1 + 2c(1+c)(1 - cos(k\Delta x))]^{-1}$

$$\boxed{\emptyset_j^{n+1} = \frac{1}{(1+C)}\left(\emptyset_j^n + C\emptyset_{j-1}^{n+1}\right)} \qquad (27)$$

Substituindo $\phi(x_j, t_n) = A^n e^{ikj\Delta x}$ **Na eqn 26º temos.**

$$\boxed{A^{n+1}e^{ikj\Delta x} = \frac{1}{(1+C)}\left(A^n e^{ikj\Delta x} + CA^{n+1}e^{ik(j+1)\Delta x}\right)}$$

$$\boxed{A^n A e^{ikj\Delta x} = \frac{1}{(1+C)}\left(A^n e^{ikj\Delta x} + CA^n A e^{ikj\Delta x}e^{ik\Delta x}\right)}$$

Cancelando os termo $A^n e^{ikj\Delta x}$.

$$\boxed{\cancel{A^n} A \cancel{e^{ikj\Delta x}} = \frac{1}{(1+C)}\left(\cancel{A^n}\cancel{e^{ikj\Delta x}} + C\cancel{A^n} A \cancel{e^{ikj\Delta x}}e^{ik\Delta x}\right)}$$

$$\boxed{A = \frac{1}{(1+C)}\left(1 + CAe^{ik\Delta x}\right)}$$

$$A = \frac{1}{(1+C)}\left(1 + CAe^{ik\Delta x}\right)$$

$$(1+C)A = \left(1 + CAe^{ik\Delta x}\right)$$

$$(1+C)A - CAe^{ik\Delta x} = 1$$

$$A\left(1 + C - Ce^{ik\Delta x}\right) = 1$$

$$A = \frac{1}{\left(1 + C - Ce^{ik\Delta x}\right)}$$

$$|A|^2 = \frac{1}{\left(1 + C - Ce^{ik\Delta x}\right)}\frac{1}{\left(1 + C - Ce^{-ik\Delta x}\right)}$$

$$|A|^2 = \frac{1}{\left(1 + C - Ce^{-ik\Delta x} + C + C^2 - C^2e^{-ik\Delta x} - Ce^{ik\Delta x} - C^2\, e^{ik\Delta x} + C^2e^{ik\Delta x - ik\Delta x}\right)}$$

$$|A|^2 = \frac{1}{\left(1 + C - Ce^{-ik\Delta x} + C + C^2 - C^2e^{-ik\Delta x} - Ce^{ik\Delta x} - C^2\, e^{ik\Delta x} + C^2\right)}$$

$$|A|^2 = \frac{1}{\left(1 + 2C + 2C^2 - (C + C^2)e^{-ik\Delta x} - (C + C^2)e^{ik\Delta x}\right)}$$

Paulo Yoshio Kubota

$$|A|^2 = \frac{1}{\left(1+2C+2C^2-(C+C^2)e^{-ik\Delta x}-(C+C^2)e^{ik\Delta x}\right)}$$

$$|A|^2 = \frac{1}{\left(1+2C+2C^2-(C+C^2)(e^{-ik\Delta x}+e^{ik\Delta x})\right)}$$

$$|A|^2 = \frac{1}{\left(1+2C+2C^2-2(C+C^2)\dfrac{(e^{-ik\Delta x}+e^{ik\Delta x})}{2}\right)}$$

$$|A|^2 = \frac{1}{\left(1+2C+2C^2-2(C+C^2)\cos(k\Delta x)\right)}$$

$$|A|^2 = \frac{1}{\left(1+2(C+C^2)-2(C+C^2)\cos(k\Delta x)\right)}$$

$$|A|^2 = \frac{1}{\left(1+2(C+C^2)(1-\cos(k\Delta x))\right)}$$

$$|A|^2 = \frac{1}{\left(1+2C(1+C)(1-\cos(k\Delta x))\right)}$$

$$|A|^2 = \frac{1}{(1 + 2C(1 + C)(1 - \cos(k\Delta x)))}$$

Para qualquer numero de onda exceto para k=0, $(1 - \cos(k\Delta x) > 0$

Então $2C(1 + C)$ **>0**

Portanto

$$|A|^2 = \frac{1}{(1 + 2C(1 + C)(1 - \cos(k\Delta x)))} \leq 1$$

A discretização no tempo realizada pelo método implícito implícita, pode ser escrito na forma: Aqui o nosso método upwind torna-se :

$$\frac{\emptyset_j^{n+1} - \emptyset_j^n}{\Delta t} + u\frac{\emptyset_j^{n+1} - \emptyset_{j-1}^{n+1}}{\Delta x} = 0$$

Nós podemos escrever a equação como um sistema linear de equações acopladas:

$$\emptyset_j^{n+1} - \emptyset_j^n = -u\frac{\Delta t}{\Delta x}\left(\emptyset_j^{n+1} - \emptyset_{j-1}^{n+1}\right)$$

Defini-se $C = u\frac{\Delta t}{\Delta x}$

$$\emptyset_j^{n+1} - \emptyset_j^n = -C\left(\emptyset_j^{n+1} - \emptyset_{j-1}^{n+1}\right)$$

$$\emptyset_j^{n+1} - \emptyset_j^n = -C\emptyset_j^{n+1} + C\emptyset_{j-1}^{n+1}$$

$$\emptyset_j^{n+1} + C\emptyset_j^{n+1} - C\emptyset_{j-1}^{n+1} = \emptyset_j^n$$

$$-C\emptyset_{j-1}^{n+1} + (1 + C)\emptyset_j^{n+1} = \emptyset_j^n$$

Em forma matricial, resolve-se para os pontos 1,. . . , j-1, isto é:

$$\begin{pmatrix} 1+C & 0 & 0 & 0 & 0 & 0 & -C \\ -C & 1+C & 0 & 0 & 0 & 0 & 0 \\ 0 & -C & 1+C & 0 & 0 & 0 & 0 \\ 0 & 0 & -C & 1+C & 0 & 0 & 0 \\ \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & 0 & -C & 1+C & 0 \\ 0 & 0 & 0 & 0 & 0 & -C & 1+C \end{pmatrix} \begin{pmatrix} \emptyset_1^{n+1} \\ \emptyset_2^{n+1} \\ \emptyset_3^{n+1} \\ \emptyset_4^{n+1} \\ \vdots \\ \emptyset_{j-2}^{n+1} \\ \emptyset_{j-1}^{n+1} \end{pmatrix} = \begin{pmatrix} \emptyset_1^{n} \\ \emptyset_2^{n} \\ \emptyset_3^{n} \\ \emptyset_4^{n} \\ \vdots \\ \emptyset_{j-2}^{n} \\ \emptyset_{j-1}^{n} \end{pmatrix}$$

Isto requer a resolução de uma matriz isto torna o métodos implícitos geralmente mais caros do que os métodos explícitos. No entanto, a análise de estabilidade mostraria que esta discretização implícita é estável para qualquer escolha de C. (Mas não se deve confundir estabilidade com precisão. As soluções mais precisas com este método ainda deverá ter um C pequeno). Observe também que a forma da matriz vai mudar dependendo da escolha das condições de contorno.

Exercício

Resolver a equação advecção 10 numericamente no domínio 0 ≤ X ≤ 1000M. Deixe Δx = 0,5 M. Presume-se que, para a velocidade de Advecção U = 1 m/s. Deixe o estado inicial ser uma função passo . Para a condição limite use phi(0,t) = 0.

$$\phi(x,0) = \begin{matrix} 0 & \text{for} & x < 40 \\ 10 & \text{for} & 40 \leq x \leq 200 \\ 0 & \text{for} & x > 200 \end{matrix} \ .$$

Usando o esquema BTBS e mostrar soluções para T = 0S T = 100 º -s, T = 200S T = 300S , T = 800S.

1. Compare as soluções de 26º com c=1,0 e c=3,0

Paulo Yoshio Kubota

```fortran
MODULE LinearSolve
 IMPLICIT NONE
 PRIVATE
 INTEGER, PARAMETER :: r8 = selected_real_kind(15, 307)
 INTEGER, PARAMETER :: r4 = selected_real_kind(6, 37)
 PUBLIC :: solve_tridiag
CONTAINS
subroutine solve_tridiag(a,b,c,d,x,n)
      implicit none
!a - sub-diagonal (diagonal abaixo da diagonal principal)
!b - diagonal principal
!c - sup-diagonal (diagonal acima da diagonal principal)
!d - parte à direita
!x - resposta
!n - número de equações
        integer, intent(in) :: n
        real (r8), dimension (n),intent (in   ) :: a,b,c,d
        real (r8), dimension (n),intent (out) :: x
        real (r8), dimension (n) :: cp, dp
        real (r8) :: m
        integer :: i
! inicializar c-primo e d-primo
        cp(1) = c(1)/b(1)
        dp(1) = d(1)/b(1)
! resolver para vetores c-primo e d-primo
         do i = 2,n
           m = b(i)-cp(i-1)*a(i)
           cp(i) = c(i)/m
           dp(i) = (d(i)-dp(i-1)*a(i))/m
         enddo
! inicializar x
         x(n) = dp(n)
! resolver para x a partir de vetores c-primo e d-primo
        do i = n-1, 1, -1
          x(i) = dp(i)-cp(i)*x(i+1)
        end do
    end subroutine solve_tridiag
END MODULE LinearSolve
```

```fortran
MODULE Class_Fields
 IMPLICIT NONE
 PRIVATE
 INTEGER, PARAMETER :: r8 = selected_real_kind(15, 307)
 INTEGER, PARAMETER :: r4 = selected_real_kind(6, 37)
 REAL (KIND=r8), PUBLIC, ALLOCATABLE :: A0  (:)
 REAL (KIND=r8), PUBLIC, ALLOCATABLE :: A   (:)
 REAL (KIND=r8), PUBLIC, ALLOCATABLE :: Anew(:)
 REAL (KIND=r8), PUBLIC, ALLOCATABLE :: AA  (:,:)
 REAL (KIND=r8), PUBLIC, ALLOCATABLE :: B   (:)
 REAL (KIND=r8), PUBLIC, ALLOCATABLE :: X   (:)
 INTEGER     , PUBLIC            :: ilo
 INTEGER     , PUBLIC            :: ihi
 INTEGER     , PUBLIC            :: iMax
 PUBLIC ::  Init_Class_Fields

 CONTAINS

SUBROUTINE Init_Class_Fields(nx,dx, coef_C )
 IMPLICIT NONE
 INTEGER        , INTENT (IN   ) :: nx
 REAL (KIND=r8) , INTENT (IN   ) :: dx
 REAL (KIND=r8) , INTENT (IN   ) :: coef_C
 REAL (KIND=r8) :: coord_X(0:nx)
 INTEGER        :: i
 iMax=nx
 ALLOCATE (A0   (0:iMax)      )
 ALLOCATE (A    (0:iMax)      )
 ALLOCATE (Anew (0:iMax)      )
 ALLOCATE (AA   (0:iMax+1,0:iMax+1))
 ALLOCATE (B    (0:iMax)      )
 ALLOCATE (X    (0:iMax)      )

!# python is zero-based.  We are assuming periodic BCs, so
!# points 0 and N-1 are the same.  Set some integer indices to
!# allow us to easily access points 1 through N-1.  Point 0
!# won't be explicitly updated, but rather filled by the BC
!# routine.
```

Paulo Yoshio Kubota

```
ilo = 1
 ihi = iMax-1
 DO i=0,iMax
   coord_X(i) = REAL(i,KIND=r8)*dx
 END DO
 !
 ! initialize the data -- tophat
 !
 DO i=0,iMax
   IF(coord_X(i) >= 0.333_r8 .and. coord_X(i) <= 0.666_r8)
THEN
      A0(i) = COS(0.50_r8 - coord_X(i))
   ELSE
      A0(i) = 0.0_r8
   END IF
 END DO
 A=A0
 AA=0.0_r8
 !
 ! """ we don't explicitly update point 0, since it is identical
 !          to N-1, so fill it here """
 A(0) = A(ihi)

 !
 !  a(t+1,i)  -   a(t,i)               a(t+1,i) - a(t+1,i-1)
 !  ----------------------  = -  U ---------------------------------
 !        Dt                                    Dx
 !
 !                                           -                                 -
 !                                      Dt  |                                   |
 !  a(t+1,i)  -  a(t,i)  =  - U ---- | a(t+1,i) - a(t+1,i-1) |
 !                                      Dx |                                   |
 !                                           -                                 -

 !                                      -                                 -
 !                                      |                                  |
 !  a(t+1,i)  -  a(t,i)  =  - C  | a(t+1,i) - a(t+1,i-1) |
 !                                      |                                  |
 !                                      -                                 -
```

```
!
 !  a(t+1,i)  -  a(t,i)  =  - Ca(t+1,i) + Ca(t+1,i-1)
 !
 !  a(t+1,i) +  Ca(t+1,i) - Ca(t+1,i-1)   =   a(t,i)

 !  -C a(t+1,i-1)  + (1 + C) a(t+1,i)   =   a(t,i)

 !  -C a(t+1,  0)  + (1 + C) a(t+1,1)   =   a(t,1)
 !  -C a(t+1,  1)  + (1 + C) a(t+1,2)   =   a(t,2)
 !  -C a(t+1,  2)  + (1 + C) a(t+1,3)   =   a(t,3)
 !  -C a(t+1,  3)  + (1 + C) a(t+1,4)   =   a(t,4)
 !  -C a(t+1,i-1)  + (1 + C) a(t+1,i)   =   a(t,i)
 !  ____                    ____   __        __
 ! |                              | |              |
 ! | (1 + C)      0     -C   | | a(t+1,1)   | =   a(t,1)
 ! | -C       (1 + C)    0   | | a(t+1,2)   | =   a(t,i)
 ! |  0          -C   (1 + C)| | a(t+1,i-1)| =   a(t,i)
 ! |____                 ____| |__        __|
 !
 !            A                            X        =    B
!
 !# create the matrix
 !# loop over rows [ilo,ihi] and construct the matrix.  This will
 !# be almost bidiagonal, but with the upper right entry also
 !# nonzero.
 !
  AA(0,ihi   ) =  1.0_r8 + coef_C
  AA(0,ilo   ) = -coef_C
  DO i =ilo,ihi
     AA(i,i+1)  = 0.0_r8
     AA(i,i   ) = 1.0_r8 + coef_C
     AA(i,i-1)  = -coef_C
  END DO
  AA(nx,ihi   ) =   1.0_r8 + coef_C
  AA(nx,ilo   ) =  -coef_C
END SUBROUTINE Init_Class_Fields
END MODULE Class_Fields
```

```fortran
MODULE Class_WritetoGrads
 USE Class_Fields, Only: Anew,A,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='ImplicitLinearAdvection1D'
  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)
   CtrlWriteDataFile=.FALSE.
   Yout=REAL(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

   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 1'
   WRITE (UnitCtl,'(A          )')'A   0 99 resultado da edol yc'
   WRITE (UnitCtl,'(A          )')'endvars'
   CLOSE (UnitCtl,STATUS='KEEP')
   CLOSE (UnitData,STATUS='KEEP')
   ok=0
 END FUNCTION SchemeWriteCtl


END MODULE Class_WritetoGrads
```

Paulo Yoshio Kubota

```fortran
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
PROGRAM Main
 USE  Class_Fields, Only : Init_Class_Fields,B,A,AA,ilo,ihi,Anew
 USE  LinearSolve, OnLy : solve_tridiag
 USE  Class_WritetoGrads, OnLy :schemeWriteCtl,
schemeWriteData,initClass_WritetoGrads

 ! main program to check the Tridiagonal system solver
 INTEGER, PARAMETER :: r8 = selected_real_kind(15, 307)
 INTEGER, PARAMETER :: r4 = selected_real_kind(6, 37)
 INTEGER      , PARAMETER :: nn=200
 INTEGER      , PARAMETER :: nx=nn
 REAL (KIND=8) , PARAMETER :: xmin=0.0
 REAL (KIND=8) , PARAMETER :: xmax=1.0
 REAL (KIND=8) , PARAMETER :: C=0.5        ! # CFL number
                                          !    [0.5, 1.0, 10.0]
! REAL(KIND=8) , PARAMETER :: u = 10.0
 REAL (KIND=8) , PARAMETER :: Dx=(xmax - xmin)/(nx-1)
! REAL(KIND=8) , PARAMETER :: Dt=C*dx/u
 INTEGER      , PARAMETER :: ninteraction=400
 INTEGER :: i,j

 CALL Init()
 CALL run()
 STOP

 CONTAINS

 SUBROUTINE Init()
  IMPLICIT NONE
    CALL Init_Class_Fields(nx,dx, C)
    CALL initClass_WritetoGrads()
 END SUBROUTINE Init
```

```fortran
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 SUBROUTINE Run()
  IMPLICIT NONE
  REAL (KIND=r8) :: a_sub_diagonal   (0:nx)
  REAL (KIND=r8) :: b_pri_diagonal   (0:nx)
  REAL (KIND=r8) :: c_sup_diagonal   (0:nx)
  INTEGER :: test,irec

  DO i =ilo,ihi
!         cc(i)= 0
!         b(i)= 1.0 + C
!         a(i)= - C

      a_sub_diagonal(i) =AA(i,i-1)   ! -C         !AA(i,i-1)
      b_pri_diagonal(i) =AA(i,i  )     ! 1.0_r8 + C !AA(i,i  )
      c_sup_diagonal(i) =AA(i,i+1)  ! 0.0_r8     !AA(i,i+1)
  END DO

  DO i =ilo,ihi
    WRITE(*,"(3F6.2)")a_sub_diagonal(i),b_pri_diagonal(i), &
                                  c_sup_diagonal(i)
  END DO
   irec=0
```

```
 DO i=1,ninteraction

! create the RHS -- this holds all entries except for a[0]

        B    (ilo:ihi)  = A(ilo:ihi)
                       ! tridag(a,b,c,d,nn)
                      ! PRINT*,A
                      ! PRINT*," ! tridag(a,b,c,d,nn)tridag(a,b,c,d,nn)"
                       Anew (ilo:ihi)  = 0.0_r8
        test=SchemeWriteData(irec)

      CALL solve_tridiag( a_sub_diagonal(ilo:ihi), &
                                  b_pri_diagonal(ilo:ihi), &
                                   c_sup_diagonal(ilo:ihi), &
                                   B                       (ilo:ihi), &
                                  Anew                  (ilo:ihi), &
                                   nx-1                  )
        Anew(ihi+1)= Anew(ihi)
       A(ilo:ihi+1) = Anew(ilo:ihi+1)

       A(ilo+2) = A(ihi  )
       A(ilo+1) = A(ihi-1)
       A(ilo   ) = A(ihi-2)

 END DO    !   t += dt
   test=SchemeWriteCtl(ninteraction)

 END SUBROUTINE Run
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
 SUBROUTINE Finalize()

 END SUBROUTINE Finalize
END PROGRAM Main
```

Exercício de difusão

Resolver numericamente o problema de difusão que tem sido discutido na seção anterior usando o esquema de diferença finitas para a derivada temporal. Use uma resolução espacial de $\Delta x = 10^{-2}$ m e Coeficiente de difusão $K = 2{,}9 \times 10^{-5}$. Integrar para pelo menos

Para 6 horas (cerca de 25000 segundos), e mostrar as soluções para T = 1 Hora, T = 2 Horas, T = 3 Horas, T = 4 Horas, T = 5 Horas, e T = 6 Horas. Comparar a solução com o Uma análise Fourier (retenção 1000 componentes). Escolha o passo de tempo sendo de tal ordem que o sistema seja estável. Deixe o primeiro setup em temperatura de 1m haste longa dada.

```
x=0.0
 DO i=1,iMax
   IF(x >= 0.0 .and. x < 0.5) THEN
     PHI_C(i)= 273.15 + 2*X
   ELSE IF(x > 0.5 .and. x < 1.0) THEN
     PHI_C(i)= 273.15 + 2.0 - 2*X
   END IF
    x =  (i)* DX
 END DO
```

Ambas as extremidades são mantidos na mesma temperatura $T_0 = 273.15$K.

Esquemas Implícitos para PDEs Parabólicas

O Esquema Crank-Nicholson é centrado no tempo e no espaço.

Método Crank-Nicolson (CN) (Método Implícito) Fornece precisão de 2ª ordem no espaço e no tempo. Média da 2ª derivada no espaço para t^{m+1} e t^{m+1} .

Tem boas propriedades de estabilidade e precisão.

Tem precisão de segunda ordem e incondicionalmente estável.

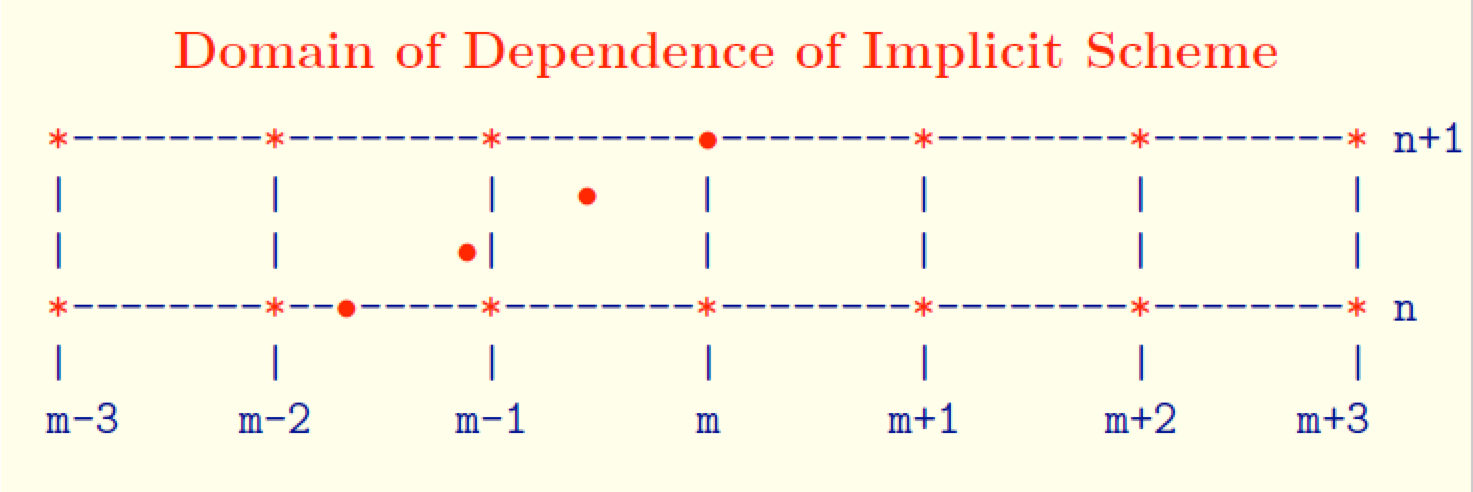
Duas formas do esquema de Crank-Nicholson para o esquema de advecção são comumente usadas:

O Esquema C-N de quatro pontos:

$$\frac{1}{2}\left[\frac{U_m^{n+1}-U_m^n}{\Delta t}+\frac{U_{m+1}^{n+1}-U_{m-1}^n}{\Delta t}\right]+\frac{c}{2}\left[\frac{U_{m+1}^{n+1}-U_m^{n+1}}{\Delta x}+\frac{U_{m+1}^n-U_m^n}{\Delta x}\right]=0$$

O Esquema C-N de seis pontos:

$$\frac{U_m^{n+1}-U_m^n}{\Delta t}+\frac{c}{2}\left[\frac{U_{m+1}^{n+1}-U_{m-1}^{n+1}}{2\Delta x}+\frac{U_{m+1}^n-U_m^n}{2\Delta x}\right]=0$$



A linha com bolinhas (●) representa uma trajetória de parcela.

O valor no ponto $m\Delta x$ no tempo $(n+1)\Delta t$ depende de todos os pontos indicados por asteriscos vermelhos (*).

Assim, o domínio computacional de dependência envolve o domínio físico de dependência.

Esta é uma <u>condição necessária para um esquema estável</u>.

Todos os esquemas implícitos também têm uma desvantagem significativa.

Como U_m^{n+1} aparece nos lados esquerdo e direito, a solução para U_m^{n+1} requer a solução de um sistema de equações.

Se envolver apenas ***sistemas tridiagonais***, isso não é um obstáculo, pois existem métodos rápidos para resolvê-los.

Existem também métodos, como passos fracionários (com cada direção espacial resolvida sucessivamente), onde uma dimensão do espaço é considerada de cada vez.

Esses esquemas chamados ADI (alternating direction implicit) permitem grandes passos de tempo sem um grande custo computacional adicional.

Esquemas Implícitos para PDEs Parabólicas

Método Crank-Nicolson (CN) (Método Implícito) Fornece precisão de 2ª ordem no espaço e no tempo. Média da 2ª derivada no espaço para t^{m+1} e t^{m+1} .

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{2}\left[\frac{T_{i-1}^{m} - 2T_{i}^{m} + T_{i+1}^{m}}{(\Delta x)^2} + \frac{T_{i-1}^{m+1} - 2T_{i}^{m+1} + T_{i+1}^{m+1}}{(\Delta x)^2}\right] + O(\Delta x)^2$$

$$\frac{\partial T}{\partial t} = \frac{T_{i}^{m+1} - T_{i}^{m}}{\Delta t} + O(\Delta t^2) \quad \text{(central difference in time now)}$$

$$-\lambda T_{i-1}^{m+1} + 2(1+\lambda)T_{i}^{m+1} - \lambda T_{i+1}^{m+1} = \lambda T_{i-1}^{m} + 2(1-\lambda)T_{i}^{m} - \lambda T_{i+1}^{m}$$

Requer I.C. para o caso em que m = 0: T_i^0 valor dado, f(x)

Requer BC's para escrever a expressão para T_0^{m+1} & T_{i+1}^{m+1}

Exercicio: Método Crank-Nicolson (CN).

$$\frac{\partial^2 T}{\partial x^2} = \frac{1}{2}\left[\frac{T_{i-1}^m - 2T_i^m + T_{i+1}^m}{(\Delta x)^2} + \frac{T_{i-1}^{m+1} - 2T_i^{m+1} + T_{i+1}^{m+1}}{(\Delta x)^2}\right] + O(\Delta x)^2$$

$$\frac{\partial T}{\partial t} = \frac{T_i^{m+1} - T_i^m}{\Delta t} + O(\Delta t^2) \quad \text{(central difference in time now)}$$

$$-\lambda T_{i-1}^{m+1} + 2(1+\lambda)T_i^{m+1} - \lambda T_{i+1}^{m+1} = \lambda T_{i-1}^m + 2(1-\lambda)T_i^m - \lambda T_{i+1}^m$$

Requer I.C. para o caso em que m = 0: T_i^0 valor dado, f(x)
Requer BC's para escrever a expressão para T_0^{m+1} & T_{i+1}^{m+1}

$$2T_i^{n+1} - 2T_i^n = \frac{\Delta t}{\Delta x \Delta x}\left(T_{i-1}^n - 2T_i^n + T_{i+1}^n + T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right)$$

$$2T_i^{n+1} - \frac{\Delta t}{\Delta x \Delta x}\left(T_{i-1}^{n+1} - 2T_i^{n+1} + T_{i+1}^{n+1}\right) = \frac{\Delta t}{\Delta x \Delta x}\left(T_{i-1}^n - 2T_i^n + T_{i+1}^n\right) + 2T_i^n$$

$$2T_i^{n+1} + 2\frac{\Delta t}{\Delta x \Delta x}T_i^{n+1} - \frac{\Delta t}{\Delta x \Delta x}T_{i-1}^{n+1} - \frac{\Delta t}{\Delta x \Delta x}T_{i+1}^{n+1} = 2T_i^n - 2\frac{\Delta t}{\Delta x \Delta x}T_i^n + \frac{\Delta t}{\Delta x \Delta x}T_{i-1}^n + \frac{\Delta t}{\Delta x \Delta x}T_{i+1}^n$$

$$-\frac{\Delta t}{\Delta x \Delta x}T_{i-1}^{n+1} + \left(2 + 2\frac{\Delta t}{\Delta x \Delta x}\right)T_i^{n+1} - \frac{\Delta t}{\Delta x \Delta x}T_{i+1}^{n+1} = \frac{\Delta t}{\Delta x \Delta x}T_{i-1}^n + \left(2 - 2\frac{\Delta t}{\Delta x \Delta x}\right)T_i^n + \frac{\Delta t}{\Delta x \Delta x}T_{i+1}^n$$

$$-\frac{\Delta t}{\Delta x \Delta x}T_{i-1}^{n+1}+\left(2+2\frac{\Delta t}{\Delta x \Delta x}\right)T_{i}^{n+1}-\frac{\Delta t}{\Delta x \Delta x}T_{i+1}^{n+1}=\frac{\Delta t}{\Delta x \Delta x}T_{i-1}^{n}+\left(2-2\frac{\Delta t}{\Delta x \Delta x}\right)T_{i}^{n}+\frac{\Delta t}{\Delta x \Delta x}T_{i+1}^{n}$$

$$-\lambda T_{i-1}^{n+1}+2(1+\lambda)T_{i}^{n+1}-\lambda T_{i+1}^{n+1}=\lambda T_{i-1}^{n}+2(1-\lambda)T_{i}^{n}+\lambda T_{i+1}^{n}$$

Exercicio:
Método Crank-Nicolson (CN).

$$-\lambda T_{i-1}^{n+1}+2(1+\lambda)T_{i}^{n+1}-\lambda T_{i+1}^{n+1}=\lambda T_{i-1}^{n}+2(1-\lambda)T_{i}^{n}+\lambda T_{i+1}^{n}$$

$$i=1 \Rightarrow -\lambda T_{0}^{n+1}+2(1+\lambda)T_{1}^{n+1}-\lambda T_{2}^{n+1}=\lambda T_{0}^{n}+2(1-\lambda)T_{1}^{n}+\lambda T_{2}^{n}$$

$$i=2 \Rightarrow -\lambda T_{1}^{n+1}+2(1+\lambda)T_{2}^{n+1}-\lambda T_{3}^{n+1}=\lambda T_{1}^{n}+2(1-\lambda)T_{2}^{n}+\lambda T_{3}^{n}$$

$$i=3 \Rightarrow -\lambda T_{2}^{n+1}+2(1+\lambda)T_{3}^{n+1}-\lambda T_{4}^{n+1}=\lambda T_{2}^{n}+2(1-\lambda)T_{3}^{n}+\lambda T_{4}^{n}$$

$$i=4 \Rightarrow -\lambda T_{3}^{n+1}+2(1+\lambda)T_{4}^{n+1}-\lambda T_{5}^{n+1}=\lambda T_{3}^{n}+2(1-\lambda)T_{4}^{n}+\lambda T_{5}^{n}$$

$$i=5 \Rightarrow -\lambda T_{4}^{n+1}+2(1+\lambda)T_{5}^{n+1}-\lambda T_{6}^{n+1}=\lambda T_{4}^{n}+2(1-\lambda)T_{5}^{n}+\lambda T_{6}^{n}$$

$$[A][X]=[B]$$

Exercicio:
Método Crank-Nicolson (CN).

$$-\lambda T_{i-1}^{n+1} + 2(1+\lambda)T_i^{n+1} - \lambda T_{i+1}^{n+1} = \lambda T_{i-1}^{n} + 2(1-\lambda)T_i^{n} + \lambda T_{i+1}^{n}$$

$$i = 1 \Rightarrow -\lambda T_0^{n+1} + 2(1+\lambda)T_1^{n+1} - \lambda T_2^{n+1} = \lambda T_0^{n} + 2(1-\lambda)T_1^{n} + \lambda T_2^{n}$$

$$i = 2 \Rightarrow -\lambda T_1^{n+1} + 2(1+\lambda)T_2^{n+1} - \lambda T_3^{n+1} = \lambda T_1^{n} + 2(1-\lambda)T_2^{n} + \lambda T_3^{n}$$

$$i = 3 \Rightarrow -\lambda T_2^{n+1} + 2(1+\lambda)T_3^{n+1} - \lambda T_4^{n+1} = \lambda T_2^{n} + 2(1-\lambda)T_3^{n} + \lambda T_4^{n}$$

$$i = 4 \Rightarrow -\lambda T_3^{n+1} + 2(1+\lambda)T_4^{n+1} - \lambda T_5^{n+1} = \lambda T_3^{n} + 2(1-\lambda)T_4^{n} + \lambda T_5^{n}$$

$$i = 5 \Rightarrow -\lambda T_4^{n+1} + 2(1+\lambda)T_5^{n+1} - \lambda T_6^{n+1} = \lambda T_4^{n} + 2(1-\lambda)T_5^{n} + \lambda T_6^{n}$$

$$\begin{bmatrix} 2(1+\lambda) & -\lambda & 0 & 0 & 0 & -\lambda \\ -\lambda & 2(1+\lambda) & -\lambda & 0 & 0 & 0 \\ 0 & -\lambda & 2(1+\lambda) & -\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 2(1+\lambda) & -\lambda & 0 \\ 0 & & 0 & -\lambda & 2(1+\lambda) & -\lambda \\ -\lambda & 0 & 0 & 0 & -\lambda & 2(1+\lambda) \end{bmatrix} \begin{bmatrix} T_0^{n+1} \\ T_1^{n+1} \\ T_2^{n+1} \\ T_3^{n+1} \\ T_4^{n+1} \\ T_{i+1}^{n+1} \end{bmatrix} = \begin{bmatrix} \lambda T_{-1}^{n} + 2(1-\lambda)T_0^{n} + \lambda T_1^{n} \\ \lambda T_0^{n} + 2(1-\lambda)T_1^{n} + \lambda T_2^{n} \\ \lambda T_1^{n} + 2(1-\lambda)T_2^{n} + \lambda T_3^{n} \\ \lambda T_2^{n} + 2(1-\lambda)T_3^{n} + \lambda T_4^{n} \\ \lambda T_3^{n} + 2(1-\lambda)T_4^{n} + \lambda T_5^{n} \\ \lambda T_4^{n} + 2(1-\lambda)T_5^{n} + \lambda T_6^{n} \\ \lambda T_{i-1}^{n} + 2(1-\lambda)T_i^{n} + \lambda T_{i+1}^{n} \end{bmatrix}$$

Exercicio: Método Crank-Nicolson (CN).

$$\begin{bmatrix} 2(1+\lambda) & -\lambda & 0 & 0 & 0 & -\lambda \\ -\lambda & 2(1+\lambda) & -\lambda & 0 & 0 & 0 \\ 0 & -\lambda & 2(1+\lambda) & -\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 2(1+\lambda) & -\lambda & 0 \\ 0 & & 0 & -\lambda & 2(1+\lambda) & -\lambda \\ -\lambda & 0 & 0 & 0 & -\lambda & 2(1+\lambda) \end{bmatrix} \begin{bmatrix} T_0^{n+1} \\ T_1^{n+1} \\ T_2^{n+1} \\ T_3^{n+1} \\ T_4^{n+1} \\ T_{i+1}^{n+1} \end{bmatrix} = \begin{bmatrix} \lambda T_{-1}^n + 2(1-\lambda)T_0^n + \lambda T_1^n \\ \lambda T_0^n + 2(1-\lambda)T_1^n + \lambda T_2^n \\ \lambda T_1^n + 2(1-\lambda)T_2^n + \lambda T_3^n \\ \lambda T_2^n + 2(1-\lambda)T_3^n + \lambda T_4^n \\ \lambda T_3^n + 2(1-\lambda)T_4^n + \lambda T_5^n \\ \lambda T_4^n + 2(1-\lambda)T_5^n + \lambda T_6^n \\ \lambda T_{i-1}^n + 2(1-\lambda)T_i^n + \lambda T_{i+1}^n \end{bmatrix}$$

$$\begin{bmatrix} 2(1+\lambda) & -\lambda & 0 & 0 & 0 & -\lambda \\ -\lambda & 2(1+\lambda) & -\lambda & 0 & 0 & 0 \\ 0 & -\lambda & 2(1+\lambda) & -\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 2(1+\lambda) & -\lambda & 0 \\ 0 & & 0 & -\lambda & 2(1+\lambda) & -\lambda \\ -\lambda & 0 & 0 & 0 & -\lambda & 2(1+\lambda) \end{bmatrix} \begin{bmatrix} T_0^{n+1} \\ T_1^{n+1} \\ T_2^{n+1} \\ T_3^{n+1} \\ T_4^{n+1} \\ T_{i+1}^{n+1} \end{bmatrix} = \begin{bmatrix} 2(1-\lambda) & \lambda & 0 & 0 & 0 & \lambda \\ \lambda & 2(1-\lambda) & \lambda & 0 & 0 & 0 \\ 0 & \lambda & 2(1-\lambda) & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 2(1-\lambda) & \lambda & 0 \\ 0 & & 0 & \lambda & 2(1-\lambda) & \lambda \\ \lambda & 0 & 0 & 0 & \lambda & 2(1-\lambda) \end{bmatrix} \begin{bmatrix} T_0^n \\ T_1^n \\ T_2^n \\ T_3^n \\ T_4^n \\ T_{i+1}^n \end{bmatrix}$$

Simbolicamente, a equação pode ser escrita

$$M_1 U_i^{n+1} = M_2 U_i^n$$

Exercicio: Método Crank-Nicolson (CN).

Simbolicamente, a equação pode ser escrita

$$M_1 U_i^{n+1} = M_2 U_i^n$$

A solução formal disso é trivial:

$$U_i^{n+1} = {M_1}^{-1} M_2 U_i^n$$

No entanto, isso requer a inversão de uma matriz M × M. Existem maneiras muito melhores de resolver isso.

A matriz M_1 é tri-diagonal periódica. Existem muito métodos numéricos eficientes de inverter um sistema com tal matriz.

Exercicio: Método Crank-Nicolson (CN).

O problema não periódico, com dados U_0^n e U_M^n, resulta em uma matriz ligeiramente diferente, mas também tri-diagonal.

Se os termos não lineares são tratados implicitamente, devemos resolver um sistema algébrico não linear a cada passo de tempo.

Isso normalmente é impraticável.

A possibilidade de usar um passo de tempo com um número de Courant muito maior que 1 em um esquema implícito não garante que obteremos resultados precisos e econômico computacionalmente.

O esquema implícito mantém a estabilidade retardando as soluções, de modo que as ondas satisfaçam a condição CFL.

Exercicio: Método Crank-Nicolson (CN).

Por esta razão, ***esquemas implícitos são úteis para aqueles modos que são muito rápidos***, mas de pouca importância meteorológica.

Em Estudos futuros, consideraremos esquemas nos quais ***os termos da onda gravitacional são implícitos*** enquanto os termos restantes são explícitos.

Esses esquemas semi-implícitos são de importância crucial na NWP moderna.

Exercicio: Método Crank-Nicolson (CN).

$$\begin{bmatrix} 2(1+\lambda) & -\lambda & 0 & 0 & 0 & -\lambda \\ -\lambda & 2(1+\lambda) & -\lambda & 0 & 0 & 0 \\ 0 & -\lambda & 2(1+\lambda) & -\lambda & 0 & 0 \\ 0 & 0 & -\lambda & 2(1+\lambda) & -\lambda & 0 \\ 0 & & 0 & -\lambda & 2(1+\lambda) & -\lambda \\ -\lambda & 0 & 0 & 0 & -\lambda & 2(1+\lambda) \end{bmatrix} \begin{bmatrix} T_0^{n+1} \\ T_1^{n+1} \\ T_2^{n+1} \\ T_3^{n+1} \\ T_4^{n+1} \\ T_{i+1}^{n+1} \end{bmatrix} = \begin{bmatrix} 2(1-\lambda) & \lambda & 0 & 0 & 0 & \lambda \\ \lambda & 2(1-\lambda) & \lambda & 0 & 0 & 0 \\ 0 & \lambda & 2(1-\lambda) & \lambda & 0 & 0 \\ 0 & 0 & \lambda & 2(1-\lambda) & \lambda & 0 \\ 0 & & 0 & \lambda & 2(1-\lambda) & \lambda \\ \lambda & 0 & 0 & 0 & \lambda & 2(1-\lambda) \end{bmatrix} \begin{bmatrix} T_0^{n} \\ T_1^{n} \\ T_2^{n} \\ T_3^{n} \\ T_4^{n} \\ T_{i+1}^{n} \end{bmatrix}$$

$$M_1 U_i^{n+1} = M_2 U_i^n$$

$$U_i^{n+1} = {M_1}^{-1} M_2 U_i^n$$

Esquema implícito: FTBS

Ondas de gravidade puras bidimensionais (2D)

Vamos agora considerar um dos subconjuntos mais simples possíveis das equações de movimento na atmosfera ou no oceano, ou seja, as equações de águas rasas linearizadas (frequentemente denominadas equações de ondas de inércia-gravidade) em duas dimensões

$$\frac{\partial u}{\partial t} - fv = -g\frac{\partial h}{\partial x}$$

$$\frac{\partial v}{\partial t} + fu = -g\frac{\partial h}{\partial y}$$

$$\frac{\partial h}{\partial t} = -H\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)$$

$f \equiv 2\Omega \sin\varphi$ é a aceleração de Coriolis, onde Ω é a frequência angular da rotação da Terra e φ a latitude. Entretanto, f é definido como uma constante. As equação suportam as soluções do tipo onda.

$$u, v, h = (u_0, v_0, h_0)e^{i(kx+ly+\omega t)}$$

Incluindo nas equações acima, obtém-se

$$\omega^2 = f^2 + gH(k^2 + l^2)$$

que descreve a relação de dispersão para as ondas de Poincaré (ondas de inércia-gravidade).

Esquema implícito: FTBS

Ondas de gravidade puras bidimensionais (2D)

Vamos agora discretizar as equações para ondas de gravidade em águas rasas bidimensionais, sem os termos de Coriolis, usando o esquema de Crank-Nicolson em uma grade C e uma integração temporal de Euler avançado

$$u_{i,j}^{n+1} = u_{i,j}^{n} - g\Delta t\left([\beta]\frac{h_{i+1,j}^{n+1} - h_{i,j}^{n+1}}{\Delta x} + [1-\beta]\frac{h_{i+1,j}^{n} - h_{i,j}^{n}}{\Delta x}\right)$$

$$v_{i,j}^{n+1} = v_{i,j}^{n} - g\Delta t\left([\beta]\frac{h_{i,j+1}^{n+1} - h_{i,j}^{n+1}}{\Delta y} + [1-\beta]\frac{h_{i,j+1}^{n} - h_{i,j}^{n}}{\Delta y}\right)$$

$$h_{i,j}^{n+1} = h_{i,j}^{n} - H\Delta t\left[\left([\beta]\frac{u_{i,j}^{n+1} - u_{i-1,j}^{n+1}}{\Delta x} + [1-\beta]\frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta x}\right) + \left([\beta]\frac{v_{i,j}^{n+1} - v_{i,j-1}^{n+1}}{\Delta y} + [1-\beta]\frac{v_{i,j}^{n} - v_{i,j-1}^{n}}{\Delta y}\right)\right]$$

$$\beta = \frac{1}{2}$$

$$u_{i,j}^{n+1} = u_{i,j}^{n} - g\Delta t\left(\frac{1}{2}\frac{h_{i+1,j}^{n+1} - h_{i,j}^{n+1}}{\Delta x} + \frac{1}{2}\frac{h_{i+1,j}^{n} - h_{i,j}^{n}}{\Delta x}\right)$$

$$v_{i,j}^{n+1} = v_{i,j}^{n} - g\Delta t\left(\frac{1}{2}\frac{h_{i,j+1}^{n+1} - h_{i,j}^{n+1}}{\Delta y} + \frac{1}{2}\frac{h_{i,j+1}^{n} - h_{i,j}^{n}}{\Delta y}\right)$$

$$h_{i,j}^{n+1} = h_{i,j}^{n} - H\Delta t\left[\left(\frac{1}{2}\frac{u_{i,j}^{n+1} - u_{i-1,j}^{n+1}}{\Delta x} + \frac{1}{2}\frac{u_{i,j}^{n} - u_{i-1,j}^{n}}{\Delta x}\right) + \left(\frac{1}{2}\frac{v_{i,j}^{n+1} - v_{i,j-1}^{n+1}}{\Delta y} + \frac{1}{2}\frac{v_{i,j}^{n} - v_{i,j-1}^{n}}{\Delta y}\right)\right]$$

Esquema implícito: FTBS

Ondas de gravidade puras bidimensionais (2D)

Vamos agora discretizar as equações para ondas de gravidade em águas rasas bidimensionais, sem os termos de Coriolis, usando o esquema de Crank-Nicolson em uma grade C e uma integração temporal de Euler avançado

$$u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{g\Delta t}{2\Delta x}\left(\left(h_{i+1,j}^{n+1} - h_{i,j}^{n+1}\right) + \left(h_{i+1,j}^{n} - h_{i,j}^{n}\right)\right)$$

$$u_{i-1,j}^{n+1} = u_{i,j}^{n} + \frac{g\Delta t}{2\Delta x}\left(\left(h_{i,j}^{n+1} - h_{i-1,j}^{n+1}\right) + \left(h_{i,j}^{n} - h_{i,-1j}^{n}\right)\right)$$

$$v_{i,j}^{n+1} = v_{i,j}^{n} - \frac{g\Delta t}{2\Delta y}\left(\left(h_{i,j+1}^{n+1} - h_{i,j}^{n+1}\right) + \left(h_{i,j+1}^{n} - h_{i,j}^{n}\right)\right)$$

$$v_{i,j-1}^{n+1} = v_{i,j}^{n} + \frac{g\Delta t}{2\Delta y}\left(\left(h_{i,j}^{n+1} - h_{i,j-1}^{n+1}\right) + \left(h_{i,j}^{n} - h_{i,j-1}^{n}\right)\right)$$

$$h_{i,j}^{n+1} = h_{i,j}^{n} - H\Delta t\left[\left(\frac{u_{i,j}^{n+1} - u_{i-1,j}^{n+1} + u_{i,j}^{n} - u_{i-1,j}^{n}}{2\Delta x}\right) + \left(\frac{v_{i,j}^{n+1} - v_{i,j-1}^{n+1} + v_{i,j}^{n} - v_{i,j-1}^{n}}{2\Delta y}\right)\right]$$

Uma análise de estabilidade dessas equações pode ser realizada inserindo as soluções de onda

$$(u^n, v^n, h^n) = (u_0, v_0, h_0)\lambda^n e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$(u^n) = (u_0)\lambda^n e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$(v^n) = (v_0)\lambda^n e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$(h^n) = (h_0)\lambda^n e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$(u^{n+1}) = (u_0)\lambda^{n+1} e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$(u^{n+1}) = (u_0)\lambda^n \lambda^1 e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$(u^{n+1}) = \lambda^1 (u_0)\lambda^n e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$u^{n+1} - u^n = (\lambda - 1)\,(u_0)\,\lambda^n e^{i(k(m)\Delta x + l(j)\Delta y)}$$

Paulo Yoshio Kubota

Esquema implícito: FTBS

Ondas de gravidade puras bidimensionais (2D)

$$u_{i,j}^{n+1} = u_{i,j}^{n} - \frac{g\Delta t}{2\Delta x}\left(h_{i+1,j}^{n+1} - h_{i,j}^{n+1} + h_{i+1,j}^{n} - h_{i,j}^{n}\right)$$

Obtém-se:

$$(h_{m+1}^{n+1}) = (h_0)\lambda^{n+1}e^{i(k(m+1)\Delta x+l(j)\Delta y)}$$

$$(h_{m}^{n+1}) = (h_0)\lambda^{n+1}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

$$(h_{m+1}^{n}) = (h_0)\lambda^{n}e^{i(k(m+1)\Delta x+l(j)\Delta y)}$$

$$(h_{m+1}^{n+1}) = \lambda^1 e^{i(k(1)\Delta x)}(h_0)\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

$$(h_{m}^{n+1}) = \lambda^1(h_0)\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

$$(h_{m+1}^{n}) = e^{i(k(1)\Delta x)}(h_0)\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

$$u^{n+1} - u^{n} = (\lambda - 1)\,(u_0)\,\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

$$(h_{m}^{n}) = (h_0)\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

$$\left(h_{i+1,j}^{n+1} - h_{i,j}^{n+1} + h_{i+1,j}^{n} - h_{i,j}^{n}\right) = \left(\lambda^1 e^{i(k(1)\Delta x)} - \lambda^1\right)(h_0)\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)} + \left(e^{i(k(1)\Delta x)} - 1\right)(h_0)\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

$$\left(h_{i+1,j}^{n+1} - h_{i,j}^{n+1} + h_{i+1,j}^{n} - h_{i,j}^{n}\right) = \left[\left(\lambda^1 e^{i(k(1)\Delta x)} - \lambda^1\right) + \left(e^{i(k(1)\Delta x)} - 1\right)\right](h_0)\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

$$\left(h_{i+1,j}^{n+1} - h_{i,j}^{n+1} + h_{i+1,j}^{n} - h_{i,j}^{n}\right) = \left[\lambda^1\left(e^{i(k(1)\Delta x)} - 1\right) + \left(e^{i(k(1)\Delta x)} - 1\right)\right](h_0)\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

$$\left(h_{i+1,j}^{n+1} - h_{i,j}^{n+1} + h_{i+1,j}^{n} - h_{i,j}^{n}\right) = \left[(\lambda + 1)\left(e^{i(k(1)\Delta x)} - 1\right)\right](h_0)\lambda^{n}e^{i(k(m)\Delta x+l(j)\Delta y)}$$

Esquema implícito: FTBS

Ondas de gravidade puras bidimensionais (2D)

$$u_{i,j}^{n+1} - u_{i,j}^{n} = -\frac{g\Delta t}{2\Delta x}\left(h_{i+1,j}^{n+1} - h_{i,j}^{n+1} + h_{i+1,j}^{n} - h_{i,j}^{n}\right)$$

Obtém-se:

$$u^{n+1} - u^{n} = (\lambda - 1)\,(u_0)\,\lambda^{n} e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$\left(h_{i+1,j}^{n+1} - h_{i,j}^{n+1} + h_{i+1,j}^{n} - h_{i,j}^{n}\right) = \left[(\lambda + 1)\left(e^{i(k(1)\Delta x)} - 1\right)\right](h_0)\lambda^{n} e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$(\lambda - 1)\,(u_0)\,\lambda^{n} e^{i(k(m)\Delta x + l(j)\Delta y)} = -\frac{g\Delta t}{2\Delta x}\left[(\lambda + 1)\left(e^{i(k(1)\Delta x)} - 1\right)\right](h_0)\lambda^{n} e^{i(k(m)\Delta x + l(j)\Delta y)}$$

$$(\lambda - 1)\,u_0 = -\frac{g\Delta t}{2\Delta x}\left[(\lambda + 1)\left(e^{i(k(1)\Delta x)} - 1\right)\right]h_0$$

Esquema implícito: FTBS

Ondas de gravidade puras bidimensionais (2D)

Obtém-se:

$$u_0(1-\lambda) = \frac{g\Delta t}{2\Delta x}(1+\lambda)\left(e^{ik\Delta x}-1\right)h_0$$

$$v_0(1-\lambda) = \frac{g\Delta t}{2\Delta y}(1+\lambda)\left(e^{il\Delta y}-1\right)h_0$$

$$h_0(1-\lambda) = H\Delta t(1+\lambda)\left(\frac{1-e^{ik\Delta x}}{\Delta x}u_0 + \frac{1-e^{il\Delta y}}{\Delta y}v_0\right)$$

$$h_0(1-\lambda) = H\Delta t\frac{(1+\lambda)}{1}\left(\frac{1-e^{ik\Delta x}}{\Delta x}\left[\frac{g\Delta t}{2\Delta x}\frac{(1+\lambda)}{(1-\lambda)}\left(e^{ik\Delta x}-1\right)h_0\right] + \frac{1-e^{il\Delta y}}{\Delta y}\left[\frac{g\Delta t}{2\Delta y}\frac{(1+\lambda)}{(1-\lambda)}\left(e^{il\Delta y}-1\right)h_0\right]\right)$$

$$h_0(1-\lambda) = g\Delta tH\Delta t\frac{(1+\lambda)(1+\lambda)}{(1-\lambda)}\left(\frac{-\left(e^{ik\Delta x}-1\right)\left(e^{ik\Delta x}-1\right)}{2\Delta x^2}[h_0] + \frac{-\left(e^{il\Delta y}-1\right)\left(e^{il\Delta y}-1\right)}{2\Delta y^2}[h_0]\right)$$

$$h_0(1-\lambda) = g\Delta tH\Delta t\frac{(1+\lambda)(1+\lambda)}{(1-\lambda)}\left(\frac{-\left(e^{ik\Delta x}-1\right)^2}{2\Delta x^2}[h_0] + \frac{-\left(e^{il\Delta y}-1\right)^2}{2\Delta y^2}[h_0]\right)$$

Paulo Yoshio Kubota

Esquema implícito: FTBS

Ondas de gravidade puras bidimensionais (2D)

$$1 - e^{ik\Delta x}$$

Obtém-se:

$$u_0(1-\lambda) = \frac{g\Delta t}{2\Delta x}(1+\lambda)\left(e^{ik\Delta x}-1\right)h_0$$

$$v_0(1-\lambda) = \frac{g\Delta t}{2\Delta y}(1+\lambda)\left(e^{il\Delta y}-1\right)h_0$$

$$h_0(1-\lambda) = H\Delta t(1+\lambda)\left(\frac{1-e^{ik\Delta x}}{\Delta x}u_0 + \frac{1-e^{il\Delta y}}{\Delta y}v_0\right)$$

$$h_0(1-\lambda) = \frac{(1+\lambda)(1+\lambda)}{(1-\lambda)}gH\Delta t^2\left(\frac{\left(1-e^{ik\Delta x}\right)^2}{2\Delta x^2}[h_0] + \frac{\left(1-e^{il\Delta y}\right)^2}{2\Delta y^2}[h_0]\right)$$

IDENTIDADES TRIGONOMÉTRICAS

1) $\operatorname{sen}^2 x + \cos^2 x = 1$
2) $\operatorname{sen}^2 x = 1 - \cos^2 x$
3) $\cos^2 x = 1 - \operatorname{sen}^2 x$
4) $\operatorname{sen}^2 nx = \frac{1}{2} - \frac{1}{2}\cos 2nx$
5) $\cos^2 nx = \frac{1}{2} + \frac{1}{2}\cos 2nx$
6) $2\operatorname{sen}^2 \frac{1}{2}x = 1 - \cos x$
7) $2\cos^2 \frac{1}{2}x = 1 + \cos x$
8) $\operatorname{sen} 2x = 2\operatorname{sen} x \cdot \cos x$
9) $\operatorname{sen} x \cdot \cos x = \frac{1}{2}\operatorname{sen} 2x$
10) $\operatorname{cosec}^2 x = \operatorname{cotg}^2 x + 1$
11) $\operatorname{tg} 2x = \frac{2\operatorname{tg} x}{1-\operatorname{tg}^2 x}$
12) $\cos 2x = \cos^2 x - \operatorname{sen}^2 x$
13) $\operatorname{cotg} x = \frac{1}{\operatorname{tg} x}$
14) $\sec x = \frac{1}{\cos x}$
15) $\operatorname{cosec} x = \frac{1}{\operatorname{sen} x}$
16) $\operatorname{tg} x = \frac{\operatorname{sen} x}{\cos x}$
17) $\sec^2 x = \operatorname{tg}^2 x + 1$
18) $\operatorname{cotg} x = \frac{\cos x}{\operatorname{sen} x}$
19) $\operatorname{sen}^2 x = \frac{1}{2}(1 - \cos 2x)$
20) $\cos^2 x = \frac{1}{2}(1 + \cos 2x)$
21) $\operatorname{sen} x \cdot \cos y = \frac{1}{2}[\operatorname{sen}(x-y) + \operatorname{sen}(x+y)]$
22) $\operatorname{sen} x \cdot \operatorname{sen} y = \frac{1}{2}[\cos(x-y) - \cos(x+y)]$
23) $\cos x \cdot \cos y = \frac{1}{2}[\cos(x-y) + \cos(x+y)]$
24) $1 \pm \operatorname{sen} x = 1 \pm \cos(\pi/2 - x)$

$$2sin^2\left(\frac{k\Delta x}{2}\right) = (1-\cos(k\Delta x))$$

$$e^{ik\Delta x} = \cos(k\Delta x) + i\sin(k\Delta x)$$

$$Real(e^{ik\Delta x}) = \cos(k\Delta x)$$

$$B = 2gH\Delta t^2\left[\frac{sin^4\left(\frac{k\Delta x}{2}\right)}{\Delta x^2} + \frac{sin^4\left(\frac{l\Delta y}{2}\right)}{\Delta y^2}\right]$$

Eliminando u_0 , v_0 e h_0 , obtemos a seguinte equação quadrática para λ

$$\lambda^2 - 2\lambda\frac{1-B}{1+B} + 1 = 0$$

Esquema implícito: FTBS

Ondas de gravidade puras bidimensionais (2D)

$$\lambda^2 - 2\frac{1-B}{1+B}\lambda + 1 = 0$$

$$\lambda_{1,2} = \frac{1 - B \pm 2i\sqrt{B}}{1+B}$$

Esses dois fatores de amplificação têm o valor absoluto

$$\left|\lambda_{1,2}\right|^2 = \left(\frac{1-B}{1+B}\right)^2 + \left(\frac{2\sqrt{B}}{1+B}\right)^2 = 1$$

e, portanto, o esquema é incondicionalmente estável.
O esquema também é considerado 'neutro estável', pois $\left|\lambda_{1,2}\right| = 1$ está justamente no limite da estabilidade.
Este exemplo de aplicação do esquema de Crank-Nicolson mostra o poder dos métodos semi-implícitos, pois estes tanto diminuem o erro de truncamento temporal de $0(Dt^1)$ para $0(Dt^2)$ quanto tornam o esquema incondicionalmente estável.