MAST90026 Computational Differential Equations: Weeks 8-10

Jesse Collis
Modified from Hailong Guo (2022)

Semester 1 2024

The University of Melbourne

Evolution PDEs

Classical examples:

$u_t = u_{xx}$,	diffusion equation,
$u_t + cu_x = 0$,	transport equation,
$u_{tt} = c^2 u_{xx}$,	wave equation,

Focus on 1+1 D (i.e. time + x) problems. 1+2 D and 1+3 D problems combine the techniques of $1+1$ D with techniques for 2D and 3D elliptic problems that we have looked at before.

We can think of 1+1 D PDEs as adding time to a BVP or adding space to an IVP. To be well posed, typically we will need both boundary conditions (like a BVP) and initial conditions (like an IVP).

Parabolic equation

Consider

$$u_t - (D(x)u')' + q(x)u = r(x),$$

with

$$\begin{aligned} \text{BCs} : u(a,t) &= \alpha(t), \\ u(b,t) &= \beta(t), \\ \text{IC} : u(x,0) &= u_0(x). \end{aligned}$$

If $D = 1$, $q = 0$, $r = 0$ then this is the heat/diffusion equation.

Two classes of numerical methods:

- Semi-discretisation
- Full discrete

Semi-discretisation (Method of line)

Idea: discretise in space only, e.g. FD, FEM, finite volume, spectral, collocation .etc. as before. This produces a system of ODEs that we solve with an IVP package e.g. *ode*45 in **Matlab**.

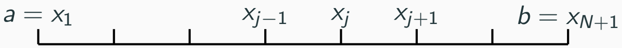
For larger problems, we need to discretise in time as well, e.g. if our IVP solver cannot handle the system of ODEs. This gives *fully discrete methods*.

But for modest problems the Method of Lines (MoL) is a good approach.

Semi-discretisation: Finite difference method for space

Consider the problem

$$u_t - (D(x)u_x)_x = r(x,t).$$



When we have an equation in flux form (or conservative form) e.g. from a conservation law, it's good practice to discretize each derivative separately, rather than expand derivatives and then discretize, i.e. $(D(x)u_x)_x$ rather than $D'(x)u_x + D(x)u_{xx}$.

Semi-discretisation: Finite difference scheme

Discretise $-(D(x)u_x)_x$ using central differences. We can use FD on $-D(x)u_x$ directly without expanding it.

$$
\begin{aligned}
D(x)u_x\big|_{x_j} &\approx \frac{D(x_j)(u_{j+1/2} - u_{j-1/2})}{h}, \\
(D(x)u_x)_x\big|_{x_j} &\approx \frac{D(x)u_x\big|_{x_{j+1/2}} - D(x)u_x\big|_{x_{j-1/2}}}{h} \\
&= \frac{1}{h}\left(\frac{D(x_{j+1/2})(u_{j+1} - u_j)}{h} - \frac{D(x_{j-1/2})(u_j - u_{j-1})}{h}\right) \\
&= \frac{1}{h^2}\left(D(x_{j+1/2})u_{j+1} - (D(x_{j+1/2}) + D(x_{j-1/2}))u_j + D(x_{j-1/2})u_{j-}\right.
\end{aligned}
$$

Semi-discretisation: Dirichlet boundary condition

Add Dirichlet boundary conditions:

$$u_1(t) = \alpha(t), \qquad u_{N+1}(t) = \beta(t).$$

Therefore at positions $j = 2, \ldots, N$ we have the following equations:

$$\dot{u}_j(t) - \frac{1}{h^2}\left(D(x_{j+1/2})u_{j+1} - (D(x_{j+1/2}) + D(x_{j-1/2}))u_j + D(x_{j-1/2})u_{j-1}\right) = r(x_j,$$

This is a complete system of $N-1$ ODEs. Use the BCs to modify the equations for $\dot{u}_2$ and $\dot{u}_N$. The initial condition becomes $u_j(0) = u_0(x_j)$.

Question: how big is the spatial discretisation error?
Answer: $O(h^2)$ because it's central differences

Semi-discretisation: Finite element method for space

Integrate by parts to get the weak formulation of the problem:

$$\int_a^b u_t\, v\, dx + \int_a^b D(x)\, u_x\, v_x\, dx = \int_a^b r\, v\, dx + \left.(D(x)\, u_x\, v)\right|_a^b, \forall v \in H_0^1(a,b).$$

Choose

$$\begin{aligned} u &= U_0(x) + \bar{u} \quad \in H_0^1 \times \mathbb{R}^+ \\ &= U_0(x) + \sum u_j(t) \underbrace{\phi_j(x)}_{\text{nodal basis}} \quad . \end{aligned}$$

Note: We need U_0 only for theoretical purpose. In the implementation, we don't need construct U_0.

Semi-discretisation: Finite element method for space

Now we get the Galerkin equations. Replace v above with ϕ_k

$$\int_a^b \left(\sum_j \dot{u}_j\phi_j\right) \phi_k \, dx + \int_a^b D(x) \left[U_0' + \sum_j u_j(t)\phi_j'\right] \phi_k' \, dx = \int_a^b r \, \phi_k \, dx.$$

Swapping the order of the sum and the integral:

$$\sum_j \underbrace{\left(\int_a^b \phi_j\phi_k \, dx\right)}_{\mathbf{M}} \dot{u}_j + \sum_j \underbrace{\left(\int_a^b D(x)\phi_j'\phi_k' \, dx\right)}_{\mathbf{K}} u_j$$

$$= \underbrace{\int_a^b r\phi_k \, dx - \int_a^b D(x)U_0'\phi_k' \, dx}_{\mathbf{f}}.$$

In matrix and vector notation:

$$\mathbf{M}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}.$$

Recap of ODE solvers

Four classes of modern methods used to solve this:

1. **Explicit Runge-Kutta methods (1895–1905)**: Simple to code, one-step methods. Matlab's $ode23$, $ode45$ are based on this.
2. **Explicit multi-step methods:** The most popular is Adams (1855). Very efficient, cheap higher-order methods (requiring one function evaluation per step, vs 3 or 6 for RK). Matlab: $ode113$.
3. **Implicit multi-step methods:** Most famous is Gear's Backward Differentiation Formula (1971). Matlab: $ode15s$.
4. **Implicit Runge-Kutta methods:** Have nice mathematical properties but are computationally expensive. Matlab: $ode23t$, $ode23tb$. There is also the TRBDF2 algorithm (1985) used for circuit simulation.

Note: Need special solvers (e.g. $ode15s$) for Stiff problems: slow and fast time scales present in the same problem

Recap of ODE solvers: Linear stability analysis

Consider the autonomous case (i.e., where f has no dependence on t):

$$\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}).$$

We travel along a solution $\bar{\mathbf{u}}$, varying slowly. In each step, we make an error $\mathbf{z}$. How does this error propagate? Write $\mathbf{y}(t) = \bar{\mathbf{u}} + \mathbf{z}(t)$, so:

$$\begin{aligned} & \dot{\mathbf{y}} = \mathbf{f}(\mathbf{y}), \\ \implies \quad & \dot{\mathbf{z}} = \mathbf{f}(\bar{\mathbf{u}} + \mathbf{z}) \\ & \quad \approx \mathbf{f}(\bar{\mathbf{u}}) + \left.\frac{\partial \mathbf{f}}{\partial \bar{\mathbf{u}}}\right|_{\bar{\mathbf{u}}} \mathbf{z} + O(||\mathbf{z}||^2). \end{aligned}$$

So locally this behaves like $\dot{\mathbf{z}} = J|_{\bar{\mathbf{u}}}\mathbf{z} + \mathbf{b}$

Recap of ODE solvers: Linear stability analysis

If J is diagonalisable (has a full set of linearly independent eigenvectors), we can write $J = T\Lambda T^{-1}$ (by the spectral theorem) where Λ is a diagonal matrix of eigenvalues and T is a matrix of eigenvectors. So:

$$\dot{\mathbf{z}} = T\Lambda T^{-1}\mathbf{z} + \mathbf{b},$$

$$\text{Define } \mathbf{w} = T^{-1}\mathbf{z}, \text{ or } \mathbf{z} = T\mathbf{w},$$

$$\text{So } \dot{\mathbf{w}} = \Lambda\mathbf{w} + T^{-1}\mathbf{b}.$$

This is a system of uncoupled scalar ODEs, $\dot{w}_i = \lambda_i w_i + b_i$. Since a real matrix can have complex eigenvalues, $\lambda \in \mathbb{C}$.

Recap of ODE solvers: Linear stability analysis

Note we use superscript to denote time, i.e. $w^n \approx w(t_n)$ and k denote time step size.

Region of Absolute Stability (RAS):

$$\{\lambda \Delta t \in \mathbb{C} : |w^n| \text{ stays bounded as } n \to \infty\}.$$

A-stability: $|w^n|$ bounded whenever $\mathbb{R}(\lambda) < 0$.

L-stability: $\left|\frac{w^{n+1}}{w^n}\right| \to 0$ as $\lambda \to -\infty$.

Recap of ODE solvers: Euler method

RAS:

$$\begin{aligned}
w^{n+1} &= w^n + kf(t_n, w^n) \\
&= w^n + k\lambda w^n \\
&= w^n(1 + k\lambda), \\
\left|w^{n+1}\right| < \infty \text{ as } n \to \infty &\Rightarrow |1 + h\lambda| \leq 1, \\
&\Rightarrow 0 \leq h \leq 2/|\lambda|.
\end{aligned}$$

If $\lambda \approx -1$ stability is not an issue, but still need Δt smaller for accuracy reasons. But what if $\lambda \ll -1$? Then we need $\Delta t < \left|\frac{2}{\lambda_{\max}}\right| \ll 1$. Now stability requirements force a much smaller Δt than accuracy requirements.

Heat equation

Consider $u_t = u_{xx}$ using finite difference

$$\begin{bmatrix} \frac{du_2}{dt} \\ \frac{du_3}{dt} \\ \vdots \\ \frac{du_N}{dt} \end{bmatrix} = \underbrace{\frac{1}{h^2} \begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & & & 1 & -2 \end{bmatrix}}_{A} \begin{bmatrix} u_2 \\ u_3 \\ \vdots \\ u_N \end{bmatrix} + \begin{bmatrix} \frac{\alpha(t)}{h^2} \\ 0 \\ \vdots \\ \frac{\beta(t)}{h^2} \end{bmatrix}$$

or

$$\dot{\mathbf{u}} = A\mathbf{u} + \mathbf{b}$$

where $\mathbf{u} = \begin{bmatrix} u_2 & u_3 & \dots & u_N \end{bmatrix}^T$ and $\mathbf{b} = \begin{bmatrix} \frac{\alpha(t)}{h^2} & 0 & \dots & \frac{\beta(t)}{h^2} \end{bmatrix}^T$

Heat equation: eigen analysis

A has eigenvalues $\frac{2}{h^2}(\cos(\pi kh) - 1)$, for $k = 1, \dots, N-1$.

$$\begin{aligned}
\lambda_1 &= \tfrac{2}{h^2}\left(\cos(\pi h) - 1\right) \\
&= \tfrac{2}{h^2}(1 - \tfrac{1}{2}\pi^2 h^2 + \cdots - 1) \\
&= -\pi^2 + O(h^2), \\
\lambda_{N-1} &= \tfrac{2}{h^2}\left(\cos(\pi(N-1)h) - 1\right) \\
&\approx \tfrac{2}{h^2}(-1 - 1) \\
&= -4/h^2.
\end{aligned}$$

λ_1 is bounded away from zero, so $\|A^{-1}\|_2 \le 1/\pi^2$; whereas $\lambda_{N-1} \to -\infty$ as $h \to 0$. Thus:

$$\kappa_2(A) = \left|\frac{\lambda_{\max}}{\lambda_{\min}}\right| \approx \frac{4}{\pi^2 h^2}.$$

Fully discrete: Parabolic PDEs

Idea: discretise the temporal derivative we get a set of fully discrete methods.

Semi-discretisation of $u_t = u_{xx}$ using finite difference: $\dot{\mathbf{u}} = A\mathbf{u} + \mathbf{b}$.

FTCS(Forward time central space):

$$\mathbf{u}^{n+1} = \mathbf{u}^n + \underbrace{\frac{k}{h^2}}_{=\ r} h^2 A\mathbf{u}^n + \bar{\mathbf{b}}^n,$$

where $\bar{\mathbf{b}}^n = k\mathbf{b}^n$.

Discretisation error is LTE $= \mathcal{O}(k + h^2)$, i.e. the method is first order in time and second order in space.

Fully discrete: Stability analysis

Eigenvalues of A lie in $[-4/h^2, -\pi^2] \Rightarrow |\lambda_i|_{\max} \approx 4/h^2$.

For stability we need $k\lambda_i$ to lie within the RAS for the time-stepping scheme. Euler's method has RAS $|1 + k\lambda| < 1$.

Substitute λ to get the stability of FTCS:

$$\begin{aligned} \left|1 + k\left(\tfrac{-4}{h^2}\right)\right| &< 1, \\ \left|1 - \tfrac{4k}{h^2}\right| &< 1, \\ -1 < 1 - \tfrac{4k}{h^2} &< 1, \\ \implies \quad -\tfrac{4k}{h^2} &< 0, \qquad \text{tells us nothing}, \\ \text{and} \qquad \tfrac{k}{h^2} &< 1/2, \qquad \text{i.e. } r < 1/2. \end{aligned}$$

$\Rightarrow$ FTCS is conditionally stable.

This is a severe restriction on the time step. If $h = 0.01$ and diffusion constant $= 1$, $k < 5 \times 10^{-5}$. That is a very small time step!

Backward Euler Method

Backward Euler method:

$$\begin{aligned} \dot{w}\big|_{t_n} &= \frac{w^n - w^{n-1}}{k}, \\ \implies \quad w^n &= w^{n-1} + k\, f(t_n, w^n). \end{aligned}$$

RAS:

$$\begin{gathered} \dot{w} = \lambda w \quad \implies \quad w^{n+1} = w^n + k\lambda w^{n+1}, \\ w^{n+1}(1 - k\lambda) = w^n, \\ \left|\tfrac{w^{n+1}}{w^n}\right| \text{ bounded if } \left|\tfrac{1}{1-k\lambda}\right| < 1, \\ \text{i.e. } |1 - k\lambda| > 1. \end{gathered}$$

$\Rightarrow$ A-stable and L-stable (as $\lambda \to \infty$, $\left|\frac{w^{n+1}}{w^n}\right| = \left|\frac{1}{1-k\lambda}\right| \to 0$).

Fully discrete: BTCS

Semi-discretisation of $u_t = u_{xx}$ using finite difference: $\dot{\mathbf{u}} = A\mathbf{u} + \mathbf{b}$.

BTCS:

$$(I - r\bar{A})\mathbf{u}^{n+1} = \mathbf{u}^n + \bar{\mathbf{b}}^{n+1}.$$

where $A = \frac{1}{h^2}\bar{A}$.

The matrix $I - r\bar{A}$ is tridiagonal so solving is cheap.

Summary of BTCS:

- all e-values of A are in RAS
- BTCS unconditional stable
- Choose k, h for accuracy reason
- LTE $= \mathcal{O}(k + h^2) \Rightarrow k \ll h$ for accuracy reason.

Fully discrete: Crank-Nicolson Method

Trapezoid rule for solving an ODE:

$$w^{n+1} = w^n + \tfrac{1}{2}k(f(t_n, w^n) + f(t_{n+1}, w^{n+1})).$$

This has $O(k^2)$ time step error.

Crank-Nicolson Method:

$$(I - \frac{r}{2}\bar{A})\mathbf{u}^{n+1} = (I + \frac{r}{2}\bar{A})\mathbf{u}^n + \frac{1}{2}(\bar{\mathbf{b}}^n + \bar{\mathbf{b}}^{n+1}).$$

Unconditional stable and LTE $= \mathcal{O}(k^2 + h^2) \Rightarrow$ Default method for diffusion equation.

Von Neumann analysis

Another way to get stability restriction without knowing e-values of matrix A.

Von Neumann analysis is based Fourier analysis and hence is generally limited to constant coefficient PDEs. For simplicity, consider Cauchy problems, i.e. no boundaries.

Use a Fourier transform to solve a linear PDE on $\mathbb{R}$:

$$\frac{\partial}{\partial x} e^{iqx} = iqe^{iqx}.$$

$\Rightarrow w(x) = e^{iqx}$ is an eigenfunction of $\frac{\partial}{\partial x}$ operator.

For any difference operator (FD, BD, CD), we get a grid function $W_j = e^{iqx_j} = e^{iqjh}$ which is an eigenfunction of the difference operator.

Von Neumann analysis: eigenfunctions

Forward differences:

$$\frac{W_{j+1} - W_j}{h} = \frac{e^{iq(j+1)h} - e^{iqjh}}{h} = \frac{1}{h} e^{iqjh}(e^{iqh} - 1)$$
$$= e^{iqjh} \underbrace{\left(\frac{e^{iqh} - 1}{h} \right)}_{\text{eigenvalue}},$$

Relationship between eigenfunction of forward difference and eigenfunction of $\frac{\partial}{\partial x}$.

Von Neumann analysis: discrete Fourier transform

Suppose we have a grid function V_j defined at grid points $x_j = jh$ for $j = 0, \pm 1, \pm 2, \ldots$, which is an l_2 function in the sense that the 2 -norm

$$\|U\|_2 = \left(h \sum_{j=-\infty}^{\infty} |U_j|^2 \right)^{1/2}$$

Express V_j as a linear combination of the grid funcitons e^{ijhq}.

$$V_j = \frac{1}{2\pi} \int_{-\pi/h}^{\pi/h} \hat{V}(q)\, e^{iqhj}\, dq,$$

where

$$\hat{V}(q) = h \sum_{j=-\infty}^{\infty} e^{-iqhj}\, V_j.$$

Von Neumann analysis: Parseval's relation

Parseval's Relation: Using the grid function 2-norm,

$$\|u\|_{\ell^2} = \left(h \sum_j |u_j|^2 \right)^{1/2} = \|\hat{U}\|_2 = \left(\int_{-\pi/h}^{\pi/h} |\hat{U}(q)|^2 \, dq \right)^{1/2}.$$

For strong stability, we want to show that

$$\|u^{n+1}\|_{\ell^2} \leq \|u^n\|_{\ell^2}, \qquad \text{i.e.} \quad \|\hat{U}^{n+1}\|_2 \leq \|\hat{U}^n\|_2.$$

Von Neumann analysis: FTCS

Look for $\hat{U}^{n+1}(q) = g(q)\hat{U}^n(q)$, where $g(q)$ is an "amplification factor".

To find $g(q)$ we use the eigenfunction e^{iqjh}:

$$\begin{aligned} u_j^{n+1} &= u_j^n + r(u_{j+1}^n - 2u_j^n + u_{j-1}^n), \\ u_j^{n+1} &= e^{iqhj} + r(e^{iqh(j+1)} - 2e^{iqhj} + e^{iqh(j-1)}) \\ &= e^{iqjh}(1 + re^{iqh} - 2 + e^{-iqh}) \\ &= u_j^n \underbrace{(1 + 2r(\cos(qh) - 1))}_{= \, g(qh)}. \end{aligned}$$

For strong stability, we want $|g(qh)| < 1$ for $qh \in [-\pi, \pi]$. So we need:

$$\begin{aligned} 1 - 4r < g = 1 + 2r(\cos(qh) - 1) < 1, \\ 1 - 4r > -1, \\ r < 1/2. \end{aligned}$$

Use Von Neumann stability analysis to show BTCS and Crank-Nisolson method are unconditionally stable.

$1+2$ D parabolic problem

Consider 1+2D parabolic problem:

$$u_t = u_{xx} + u_{yy}.$$

Using Crank-Nicolson plus a five-point stencil, we end up with equations like:

$$\begin{aligned} u_{ij}^{n+1} &= u_{ij}^n + \tfrac{k}{2}\left(\nabla_h^2 u_{ij}^n + \nabla_h^2 u_{ij}^{n+1}\right), \\ \implies \quad \underbrace{(I - \tfrac{k}{2}\nabla_h^2)}_{=\,A} u_{ij}^{n+1} &= (I + \tfrac{k}{2}\nabla_h^2) u_{ij}^n. \end{aligned}$$

condition number $\kappa_2(A) = O(k/h^2) \Rightarrow$ better conditioned than Poisson equation $\Rightarrow$ iterative methods, e.g. CG, converge faster

$1+2$ D parabolic problem: locally one-dimensional(LOD)

Idea: Use the fact that $\nabla_h^2 = D_x^2 + D_y^2$ to break up the computation: solve in x direction, then in y.

$$\begin{aligned}
u_{ij}^{n+1} &= u_{ij}^n + \tfrac{k}{2}(D_x^2 u_{ij}^n + D_x^2 u_{ij}^{n+1} + D_y^2 u_{ij}^n + D_y^2 u_{ij}^{n+1}),\\
\implies \quad u_{ij}^* &= u_{ij}^n + \tfrac{k}{2}(D_x^2 u_{ij}^n + D_x^2 u_{ij}^*),\\
u_{ij}^{n+1} &= u_{ij}^* + \tfrac{k}{2}(D_y^2 u_{ij}^* + D_y^2 u_{ij}^{n+1}),
\end{aligned}$$

$$\implies \quad (I - \tfrac{k}{2}D_x^2)u^* = (I + \tfrac{k}{2}D_x^2)u^n, \qquad \text{C–N in } x, \tag{1}$$

$$(I - \tfrac{k}{2}D_y^2)u^{n+1} = (I + \tfrac{k}{2}D_y^2)u^*, \qquad \text{C–N in } y. \tag{2}$$

In (1), u^* is compiled over rows through D_x^2. Each equation for j is independent, so there are $j = 0, \dots, m+1$ systems, and each system has a tridiagonal matrix for u_{ij}^*. So it's $O(m^2)$ operations to solve for u^*.

In (2), u^{n+1} is compiles over columns through D_y^2, giving m tridiagonal systems. Total work is $2m+2$ tridiagonal systems of size m, for $O(m^2)$ operations.

$1+2$ D parabolic problem: alternative direction implicit(ADI)

In the first step, the y diffusion is explicit, x diffusion is implicit. In the second step, it is reversed.

$$\begin{aligned} u_{ij}^{*} &= \tfrac{k}{2}(D_y^2 u_{ij}^{n} + D_x^2 u_{ij}^{*}), \\ u_{ij}^{n+1} &= u_{ij}^{*} + \tfrac{k}{2}(D_x^2 u_{ij}^{*} + D_y^2 u_{ij}^{n+1}). \end{aligned}$$

$\mathcal{O}(k^2+h^2)$ error, unconditionally stable, $O(m^2)$ operations per time step.

Wave equation

Consider $u_{tt} = c^2 u_{xx}$. Define $v = u_t, w = -cu_x$ then

$$v_t = u_{tt} = c^2 u_{xx} = -cw_x,$$

and

$$v_t + cw_x = 0, \qquad w_t + cv_x = 0.$$

So if we let

$$\mathbf{u} = \begin{bmatrix} v \\ w \end{bmatrix},$$

$$\mathbf{u}_t + \begin{bmatrix} 0 & c \\ c & 0 \end{bmatrix} \mathbf{u}_x = 0,$$

$$\mathbf{u}_t + A\mathbf{u}_x = 0, \qquad \text{(both linear)}.$$

The system is hyperbolic if A is diagonalisable with real eigenvalues, $\det(A - I\lambda) = 0$. e.g.:

$$A = \begin{bmatrix} 0 & c \\ c & 0 \end{bmatrix}, \quad \implies \quad \lambda = \pm c.$$

Advection equation

Initial boundary value problem(IBVP):

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = 0, \quad x \in (0,1)$$

Initial condition: $u(x,0) = u^0(x)$

Boundary conditions: $\begin{cases} u(0,t) = g_0(t) \text{ if } c > 0 \\ u(1,t) = g_1(t) \text{ if } c < 0 \end{cases}$

Advection equation: solution

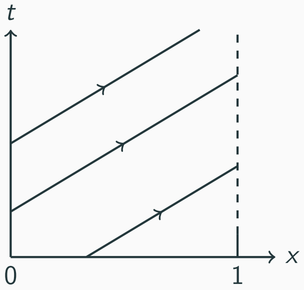
$$du = \frac{\partial u}{\partial t} dt + \frac{\partial u}{\partial x} dx = \left(\frac{\partial u}{\partial t} + \frac{dx}{dt}\frac{\partial u}{\partial x} \right) dt$$

If $\frac{dx}{dt} = c \Rightarrow x = ct + \xi$ Characteristics

$$\Downarrow$$

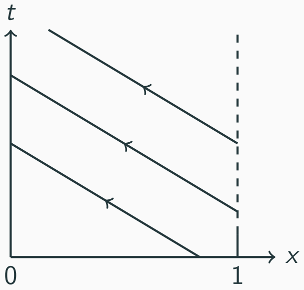
$$du = 0, \Rightarrow \underbrace{u(x,t) = f(\xi) = f(x-ct)}_{\text{General solution}}$$

Advection equation: solution when $c > 0$



$$u(x,t) = \begin{cases} u^0(x - ct), & \text{if } x - ct > 0 \\ g_0(t - x/c), & \text{if } x - ct < 0 \end{cases}$$

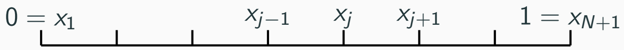
Advection equation: solution when $c < 0$



$$u(x,t) = \begin{cases} u^0(x-ct), & \text{if } x-ct<1 \\ g_1(t-x/c), & \text{if } x-ct>1 \end{cases}$$

Advection equation: Method of line

Consider $u_t + cu_x = 0$ with periodic BCs $u(0,t) = u(1,t)$.



BCs are $u_{N+1} = u_1$, and we need to solve for $u_2, \dots, u_{N+1}$. Then using our MoL equations from before:

$$\begin{aligned} \dot{u}_j(t) &= -\tfrac{c}{2h}(u_{j+1} - u_{j-1}), \\ \dot{u}_2(t) &= -\tfrac{c}{2h}(u_3 - u_1) = -\tfrac{c}{2h}(u_3 - u_{N+1}), && \text{(not solving for } u_1\text{)}, \\ \dot{u}_{N+1}(t) &= -\tfrac{c}{2h}(u_{N+2} - u_N) = -\tfrac{c}{2h}(u_2 - u_N), && \text{(periodic BCs)}. \end{aligned}$$

Advection equation: Method of line in matrix form

Matrix form:

$$\dot{\mathbf{u}} = A\mathbf{u},$$

where the matrix A is given by:

$$A = -\frac{c}{2h}\begin{bmatrix} 0 & 1 & \cdots & & & 0 & -1 \\ -1 & 0 & 1 & \cdots & & 0 & 0 \\ 0 & -1 & 0 & 1 & \cdots & 0 & 0 \\ \vdots & & & & & & \\ 1 & 0 & 0 & 0 & \cdots & -1 & 0 \end{bmatrix}.$$

Advection equation: eigenvalue of matrix

A is skew-symmetric: $A^T = -A$! Its eigenvalues are imaginary.

$$\lambda_p = -\frac{ic}{h}\sin(2\pi ph), \qquad p = 1, \dots, N.$$

Starts small, goes up and down. $\lambda_p \in [-ic/h, ic/h]$.

For stability, we need RAS for time stepper $\supset \Lambda(A)$.

So we try FTCS. But its RAS never includes λ_p. FTCS is *never stable for the advection equation*.

Advection equation: leap frog method

Leap frog method:

$$\dot{u} = \frac{u^{n+1} - u^{n-1}}{2k}.$$

Its RAS is $|\mathfrak{Im}(z)| \leq 1$. Leapfrog will be stable provided

$$\lambda_k \in [-ick/h, ick/h] \subset [-i, i],$$

i.e. $\left|\frac{ck}{h}\right| < 1$.

Courant Number: $\frac{ck}{h} = \nu$. Condition on stability is $\nu < 1$.

Fully discrete leapfrog method (CTCS).

$$u_j^{n+1} = u_j^{n-1} - \tfrac{ck}{h}(u_{j+1}^n - u_{j-1}^n).$$

This is an explicit method, $O(h^2 + k^2)$. Three time levels means we need a starting procedure and more memory.

Advection equation: Lax-Friedrichs Method (LxF)

For FTCS we had:

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

If we replace u_j^n with the average $\frac{1}{2}(u_{j+1}^n + u_{j-1}^n)$ we get LxF:

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

and CLAW (for nonlinear conservation laws):

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

Advection equation: Lax-Friedrichs Method (LxF)

$\frac{1}{2}(u_{j+1}^n + u_{j-1}^n)$ can be rewritten as $u_j^n + \frac{1}{2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n)$. So you've added dissipation (using central differences of u_{xx}). Adding numerical diffusion to stabilise the solution.

Approximation of time derivative at j:

$$\frac{u_j^{n+1} - u_j^n}{k} = -\tfrac{c}{2h} A_- \mathbf{u} + \tfrac{1}{2k} A_+ \mathbf{u}.$$

Approximation of time derivative at j:

$$\frac{u_j^{n+1} - u_j^n}{k} = -\tfrac{c}{2h} A_- \mathbf{u} + \tfrac{1}{2k} A_+ \mathbf{u}.$$

Splitting A into skew-symmetric and symmetric parts.

$$\dot{u}_j = -\tfrac{c}{2h} A_- \mathbf{u} + \tfrac{1}{2k} A_+ \vec{u} = B\mathbf{u}.$$

Advection equation: Lax-Friedrichs Method (LxF)

Eigenvalues of B: $-\frac{ic}{h}\sin(2\pi ph) - \frac{1}{k}(1 - cos(2\pi ph)) = \mu$.

At time step k, $k\mu = -i\nu\sin(2\pi ph) - (1 - cos(2\pi ph)$ which lie on an ellipse.

If p is a parameter then

$$k\mu = -\underbrace{i\nu\sin(2\pi ph)}_{\text{imaginary}} - \underbrace{(1 - cos(2\pi ph))}_{\text{real}}.$$

So $k\mu$ in the RAS if $|k\mu + 1| < 1$, i.e. $\nu^2\sin^2(2\pi ph) + \cos^2(2\pi ph) < 1$ $\Rightarrow |\nu| < 1$.

LxF is stable if FT RAS $\subset k\mu$ of B, which happens iff $|\nu| < 1$. But it is 1st order (LTE $= \mathcal{O}(k + h^2)$).

Advection equation: Lax-Wendroff Method (LxW)

To get a 2nd order method, use Taylor series expansion on PDE

$$u(x,t+k) = u(x,t) + k u_t(x,t) + \tfrac{1}{2}k^2 u_{tt}(x,t) + O(k^3),$$

$$u_t = -c u_x, \qquad u_{tt} = c^2 u_{xx},$$

$$\implies \quad u(x,t+k) = u(x,t) - kc \underbrace{u_x}_{\text{CD}} + \frac{k^2c^2}{2}\underbrace{u_{xx}}_{\text{CD}} + O(k^3),$$

$$\implies \quad u_j^{n+1} = u_j^n - \tfrac{ck}{2h}(u_{j+1}^n - u_{j-1}^n) + \tfrac{c^2k^2}{2h^2}(u_{j+1}^n - 2u_j^n + u_{j-1}^n),$$

$\dot{\mathbf{u}} = B\mathbf{u}$ evolves the system and:

$$k\mu = -i\left(\tfrac{ck}{h}\right)\sin(\pi p h) + \left(\tfrac{ck}{h}\right)^2(\cos(\pi p h) - 1), \qquad p = 1, \dots, N.$$

It is stable if $|\nu| \leq 1$

Advection equation: Lax-Wendroff Method (LxW)

Von Neumann analysis:

$$\begin{aligned}
u_j^n e^{iqhj} &= e^{iqhj} - \tfrac{1}{2}\nu(e^{iqh(j+1)} - e^{iqh(j-1)}) + \tfrac{1}{2}\nu^2(e^{iqh(j+1)} - 2e^{iqhj} + e^{iqh(j-1)})\\
&= e^{iqhj}(1 - \tfrac{1}{2}\nu 2i\sin(qh) + \tfrac{1}{2}\nu^2(2\cos(qh) - 2))\\
&= e^{iqhj}(1 - \tfrac{1}{2}\nu 2i\sin(qh) + \nu^2(\cos(qh) - 1)),\\
g(qh) &= 1 - i\nu 2\sin(qh/2)\cos(qh/2) - \nu^2 2\sin^2(qh/2),\\
|g|^2 &= (1 - 2\nu^2\sin^2(qh/2))^2 + 4\nu^2\sin^2(qh/2)\cos^2(qh/2)\\
&= 1 - 4\nu^2\sin^2(qh/2) + 4\nu^4\sin^4(qh/2) + 4\nu^2\sin^2(qh/2)\cos^2(qh/2)\\
&= 1 - 4\nu^2\sin^2(qh/2)(1 - \cos^2(qh/2)) + 4\nu^4\sin^4(qh/2)\\
&= 1 - 4\nu^2(1 - \nu^2)\underbrace{\sin^4(qh/2)}_{\in\ [0,1]}.
\end{aligned}$$

$$\implies \quad |g|^2 \leq 1, \quad \text{if } |\nu| \leq 1.$$

Advection equation: upwind method

The advection equation has a direction:

- If $c > 0$, information comes from the left.
- If $c < 0$, information comes from the right.

Use 1-sided FD, depending on sign of c. This gives us "upwind methods": use FTBS for $c > 0$ and use FTFS for $c < 0$.

$$\begin{aligned} u_j^{n+1} &= \begin{cases} u_j^n - \nu(u_j^n - u_{j-1}^n), & \text{if } c > 0 \\ u_j^n - \nu(u_{j+1}^n - u_j^n), & \text{if } c < 0 \end{cases} \\ &= u_j^n - \frac{\nu}{2}(u_{j+1}^n - u_{j-1}^n) + \frac{\nu}{2}(\text{sign}c)(u_{j+1}^n - 2u_j^n + u_{j-1}^n). \end{aligned}$$

Advection equation: stability of upwind method

Von Neumann analysis:

- FTBS stable for $0 < \nu < 1$ ($c > 0$ only).
- FTFS stable for $-1 < \nu < 0$ ($c < 0$ only).

Advection equation: Beam-Warning method

But upwind method is only first-order accurate. To get 2nd order method, use 1-sided 2nd order FD in Lax-Wendroff method,

$$u_j^{n+1} = \begin{cases} u_j^n - \frac{\nu}{2}(3u_j^n - 4u_{j-1}^n + u_{j-2}^n) + \frac{\nu^2}{2}(u_j^n - 2u_{j-1}^n + u_{j-2}^n), & \text{if } c > 0 \\ u_j^n - \frac{\nu}{2}(-3u_j^n + 4u_{j-1}^n - u_{j-2}^n) + \frac{\nu^2}{2}(u_j^n - 2u_{j-1}^n + u_{j-2}^n), & \text{if } c < 0 \end{cases}$$

Stable when $0 \leq \nu \leq 2$ if $c > 0$ and $-2 \leq \nu \leq 0$ if $c \leq 0$.

Advection equation: alternative derivation

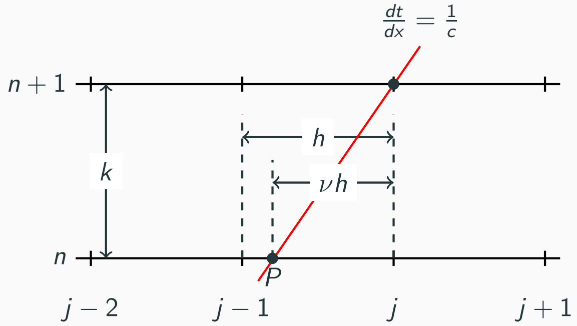
Characteristics:

- Straight lines with slope c:
- the exact solution satisfies: $u(x_j, t_{n+1}) = u(x_j - ck, t_n)$.

If $0 < \frac{ck}{h} < 1$, then $ck < h$ so $x_j - ck \in (x_{j-1}, x_j)$. If we choose $\frac{ck}{h} = \nu = 1$, then $x_j - ck = x_j - h = x_{j-1}$. Thus $u_j^{n+1} = u_{j-1}^n$ exactly. But it is unstable.

Exercise: FTCS, LxF, LxW, UW satisfies this equation when $\nu = 1$.

Advection equation: alternative derivation of UW scheme



$\boxed{u_j^{n+1} = u_P}$

Use linear interpolation $j-1, j$

$$u_P \approx \nu u_{j-1}^n + (1-\nu) u_j^n.$$

Advection equation: alternative derivation

If $0 < \nu < 1$ then $x_j - ck$ lies in (x_{j-1}, x_j). We have numerical values for $u(x_j) \approx u_j^n$ and $u(x_{j-1}) \approx u_{j-1}^n$.
$\Rightarrow u(x_j - ct, t_n)$ by interpolating numerical data!

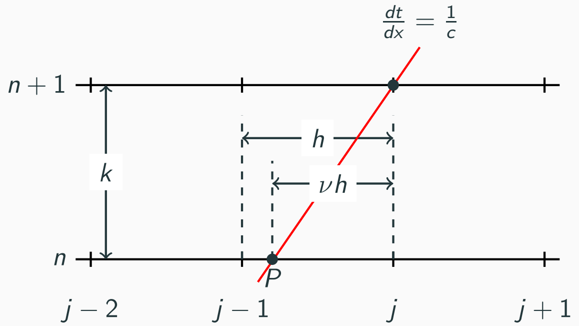
At t_n we know u_j, u_{j-1} so we form the linear interpolant p_1:

$$\begin{aligned} p_1(x) &= u_j^n + (x - x_j)\tfrac{1}{h}(u_j^n - u_{j-1}^n), \\ p_1(x_j - ck) &= u_j^n + (-ck)\tfrac{1}{h}(u_j^n - u_{j-1}^n), \\ &= u_j^n - \nu(u_j^n - u_{j-1}^n). \end{aligned}$$

This is just the upwind method, for $c > 0$. Similarly if $c < 0$ we get $p_1(x_j - ck) = u_j^n - \nu(u_{j+1}^n - u_j^n)$ which is the UW method for $c < 0$.

If $\frac{ck}{h} > 1$ we would be extrapolating in a region outside (x_{j-1}, x_j). This would be bad.

Advection equation: alternative derivation of LxF scheme

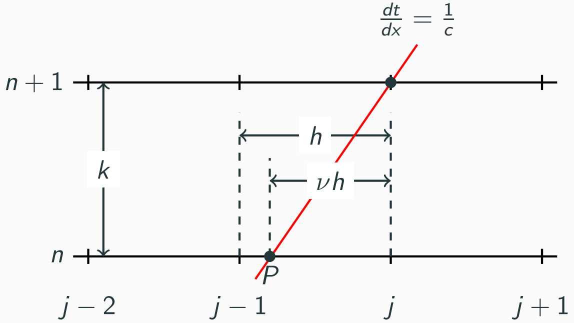


$$\boxed{u_j^{n+1} = u_P}$$

Use quadratic interpolation $j-1, j, j+1$

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

Advection equation: alternative derivation of BM scheme



$$\boxed{u_j^{n+1} = u_P}$$

Use quadratic interpolation $j-2, j-1, j$

$$u_P \approx -\frac{\nu}{2}(1-\nu)u_{j-2}^n + \nu(2-\nu)u_{j-1}^n + \frac{\nu}{2}(2-\nu)u_{j+1}^n$$

Advection equation: alternative derivation

If we use a quadratic interpolant p_2 based on $u_{j-1}^n, u_j^n, u_{j+1}^n$. This results in the LxW method.

Using p_2 based interpolation on $u_{j-2}^n, u_{j-1}^n, u_j^n$ gives the Beam-Warming equation. This is a second order upwind method.

For a method in which u_j^{n+1} depends on $u_{j+p}^n, \dots, u_{j+q}^n$ $(p < q)$, we must have

$$
\begin{aligned}
x_j + ph = x_{j+p} &\leq x_j - ck \leq x_{j+q} = x_j + qh, \\
\implies -q &\leq \underbrace{\tfrac{ck}{h}}_{=\,\nu} \leq -p, \text{ in order to be interpolating, not extrapolating.}
\end{aligned}
$$

Advection equation: domain of dependence

For the exact solution: the domain of dependence of (X, T) is the point $X - cT$.

In addition, we can consider

$$\begin{aligned} u_{tt} &= c^2 u_{xx} : (X - ct, X + ct), \\ u_t &= D u_{xx} : \mathbb{R}. \end{aligned}$$

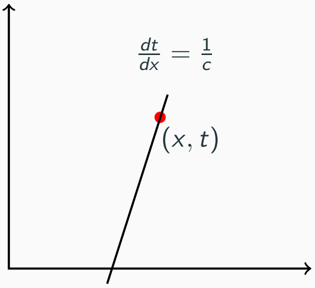
Numerical method also has a domain of dependence:

$$u_j^n \text{ depends on } \begin{cases} u_{j+p}^{n-1} & \dots & u_{j+q}^{n-1}, \\ \vdots & & \vdots \\ u_{j+np}^{0} & \dots & u_{j+nq}^{0}, \end{cases}$$

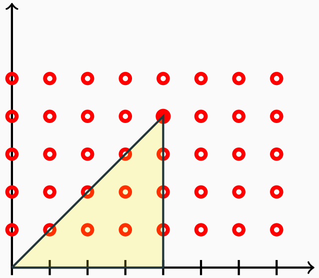
Define $T = nk$, then $nh = T\frac{h}{r}$ and $n = \frac{T}{r}$ (mesh ratio), where $r = k/h$. So the domain of dependence for u_j^n will be

$$(u((j+np)h, 0), u((j+nq)h, 0)) = \left(u^0\left(X + p\frac{T}{r}\right), u^0\left(X + q\frac{T}{r}\right)\right).$$

Advection equation: domain of dependence for UW scheme

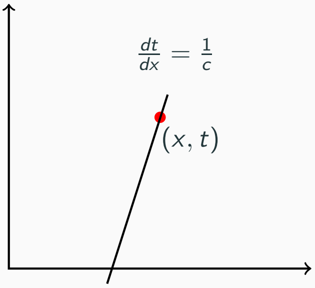


Analytic

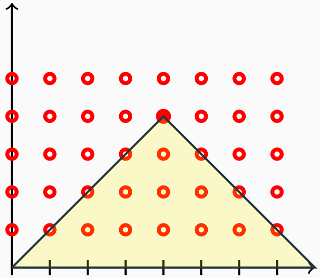


Numerical

Advection equation: domain of dependence for LxW scheme



Analytic



Numerical

Advection equation: domain of dependence

The numerical domain of dependence will fill the interval

$$\left(X + p\frac{T}{r}, X + q\frac{T}{r}\right).$$

LxW has $p = -1, q = 1$ so domain of dependence is

$$\left(X - \frac{T}{r}, X + \frac{T}{r}\right).$$

UW $(c > 0)$ has $p = -1, q = 0$ so domain of dependence is

$$\left(X - \frac{T}{r}, X\right).$$

For the numerical solution to converge to exact solution as $k, h \to 0$, must have the domain of dependence $X - cT \subseteq$ numerical domain of dependence $[X + \frac{pT}{r}, X + \frac{qT}{r}]$. Otherwise, can change $u^0(X - cT)$ which should change $u(X, T)$ but that change doesn't affect numerical solution.

Advection equation: CFL condition

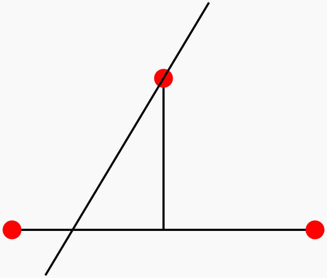
CFL condition: For a (FD) method to converge for a time dependent PDE must have exact domain of dependence $\subset$ numerical domain of dependence.

It often give the same stability condition as we seen before. A necessary condition for convergence.

For advection equation, we have

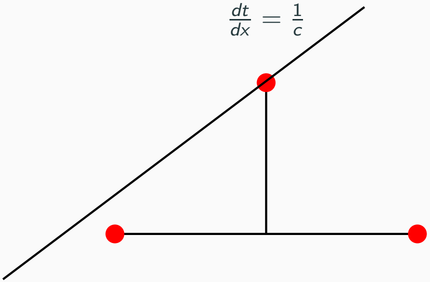
$$\begin{aligned} & X - cT \leq [X + pT/r, X + qT/r] \\ \Rightarrow \quad & pT/r \leq -cT \leq qT/r \\ \Rightarrow \quad & -q \leq cr = \nu \leq -p \\ & \boxed{\text{CFL condition}} \end{aligned}$$

Advection equation: CFL condition for LxW scheme



$\nu < 1$

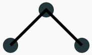
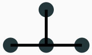
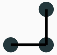
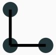
Stable



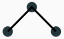
$\nu > 1$

Unstable

Advection equation: examples of CFL condition

LxF		$p = -1, q = 1$	$-1 \leq \nu \leq 1$
LxW		$p = -1, q = 1$	$-1 \leq \nu \leq 1$
UW		$p = -1, q = 0$	$0 \leq \nu \leq 1$
UW		$p = -1, 1 = 0$	$-1 \leq \nu \leq 0$

But not sufficient

FTCS  $\Rightarrow |\nu| \leq 1$ from CFL condition   but not stable for any ν

Advection equation: CFL condition for heat equation

Heat equation on $[a, b]$ (or $\mathbb{R}$): domain dependance $= [a, b]$ (or $\mathbb{R}$)

FTCS: u_j^{n+1} depends on $u_{j-1}^n, u_j^n, u_{j+1}^n$.

But we don't refine at fixed $r = \frac{k}{h}$ instead refine at fixed $\mu = \frac{k}{h^2}$, so u_j^n depends on

$$\begin{aligned} &u_{j-n}^0, \cdots, u_{j+n}^0 \\ =&u(jh - nh, jh + nh) \\ =&u(X - nh, X + nh) \\ =&u(X - T\frac{h}{k}, X + T\frac{h}{k}) \end{aligned}$$

$T = nk$. Refine at fixed $\mu = \frac{k}{h^2} \Rightarrow \frac{h}{k} = \frac{1}{\mu h}$.

Numerical domain of dependence is $(X - \frac{T}{\mu h}, X + \frac{T}{\mu h}) \Rightarrow \mathbb{R}$ as $h \Rightarrow 0$ at fixed $\mu \Rightarrow$ need to refine at fixed μ for FTCS but only stable for $0 \leq \mu \leq \frac{1}{2}$.

Advection equation: CFL condition for heat equation

BTCS/CN: they are implicit and use inverse matrix

$$(I - rA)^{-1} \text{ or } (I - \frac{r}{2}A)^{-1}$$

$\Rightarrow$ u_j^{n+1} depends on all u_i^n

$\Rightarrow$ numerical domain of dependence = whole interval $[a, b]$

$\Rightarrow$ always satisfy CFL condition

Modified equation: motivation

So far have looked at LTE to tell us how accurate a method is

A different way to look at it: Is there a PDE for which the numerical solution produces the exact solution?

Less ambitions: can we find a PDE that is better satisfied by our numerical method than the original problem?

Modified equation for upwind scheme

$u_t + cu_x = 0 \Rightarrow \quad u_j^{n+1} = u_j^n - \frac{ck}{h}\left(u_j^n - u_{j-i}^n\right)$

v(x,t): satisfy the difference equation exactly

$$v(x, t+k) = v(x,t) - \frac{ck}{h}\left(v(x,t) - v(x-h,t)\right)$$

Taylor series:

$$\left(v_t + \frac{1}{2}kv_{tt} + \frac{1}{6}k^2 v_{ttt} + \cdots\right) + c\left(v_x - \frac{1}{2}hv_{xx} + \frac{1}{6}h^2 v_{xxx} + \cdots\right) = 0$$

$$\Rightarrow v_t + cv_x = \frac{1}{2}(chv_{xx} - kv_{tt}) + \mathcal{O}\left(k^2 + h^2\right)$$

$$\partial_t : v_{tt} + cv_{xt} = \frac{1}{2}(chv_{xxt} - kv_{ttt}) + \mathcal{O}\left(k^2 + h^2\right)$$

$$\partial_x : v_{tx} + cv_{xx} = \frac{1}{2}(chv_{xxx} - kv_{ttx}) + \mathcal{O}\left(k^2 + h^2\right)$$

$$\Rightarrow v_{tt} = -cv_{xt} + \mathcal{O}(h+k) = c^2 v_{xx} + \mathcal{O}(k+h)$$

$$\Rightarrow v_t + cv_x = \frac{c}{2}(hv_{xx} - ckv_{xx}) + \mathcal{O}\left(k^2 + h^2\right)$$

Modified Equation: Dissipation and Dispersion

Model Problem:

$$\frac{\partial u}{\partial t} + c\frac{\partial u}{\partial x} = a\frac{\partial^2 u}{\partial x^2} - b\frac{\partial^3 u}{\partial x^3}, \quad x \in (0,1)$$

with $u(x,0) = u^0(x)$ and periodic boundary conditions.

Solution

$$u(x,t) = \sum_{q=-\infty}^{k=\infty} \mathbb{U}_q^0 e^{-4\pi^2\sigma(q)t} e^{i2\pi(qx-\omega(q)t)}$$

$$\sigma(q) = aq^2, \quad \omega(q) = cq - b4\pi^2 q^3$$

Modified Equation: Dissipation and Dispersion

$e^{-4\pi^2\sigma(q)t}$ represents Decay

$\sigma(q)$ dissipation relation

$e^{i2\pi(qx-\omega(q)t)}$ represents Propagation

$\omega(q)$ dispersion relation

For the exact solution of $u_t + cu_x = 0$:

$\sigma = 0$ no dissipation

$\omega = qc$, or $\omega/q = c$ (constant) no dispersion

Modified Equation: Dissipation and Dispersion

First Order Upwind

$$u_t + cu_x = \frac{ch}{2}(1-\nu)u_{xx} - \frac{ch^2}{6}\left(1-\nu^2\right)u_{xxx}.$$

Lax-Wendroff

$$u_t + cu_x = -\frac{ch^2}{6}\left(1-\nu^2\right)u_{xxx}.$$

Beam-Warming

$$u_t + cu_x = \frac{ch^2}{6}(2-\nu)(1-\nu)u_{xxx}.$$

Modified Equation: Dissipation and Dispersion

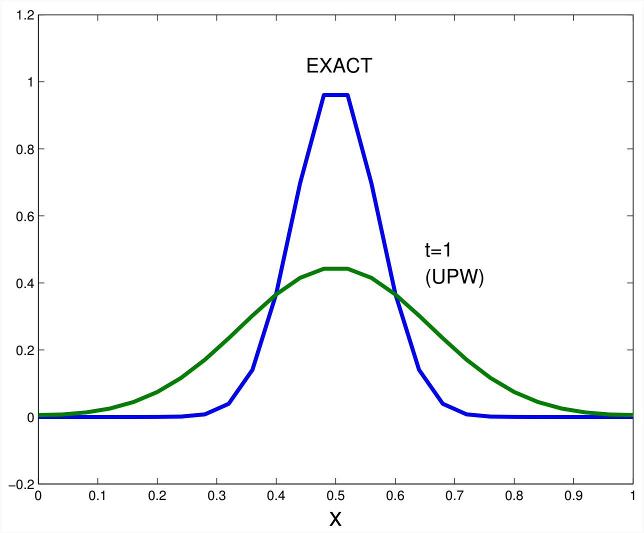
For the upwind scheme dissipation dominates over dispersion $\Rightarrow$ Smooth solutions.

For Lax-Wendroff and Beam-Warming dispersion is the leading error effect $\Rightarrow$ Oscillatory solutions (if not well resolved).

Lax-Wendroff has a negative phase error.

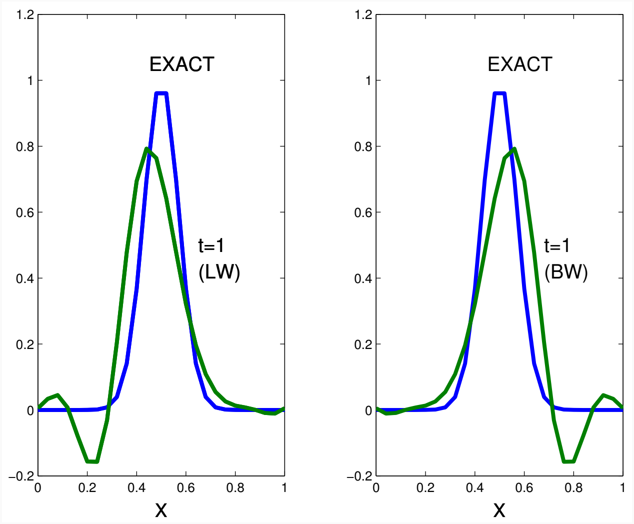
Beam-Warming has (for $\nu < 1$) a positive phase error.

Numerical example of upwind scheme



$h = 1/25 \quad \nu = 0.5$

Numerical example of LxW and BW scheme



$h = 1/25 \quad \nu = 0.5$

End of week 10!