Computational Fluid Dynamics
http://www.nd.edu/~gtryggva/CFD-Course/

Computational Fluid Dynamics

Lecture 8
February 6, 2017

Grétar Tryggvason

Computational Fluid Dynamics
http://www.nd.edu/~gtryggva/CFD-Course/

Diffusion versus dispersion

Godunov Theorem

Godunov Method

Upwind and Flux Splitting

Higher order schemes

Artificial viscosity

Computational Fluid Dynamics

Modified Equation

Computational Fluid Dynamics

Looking at the structure of the error terms—by deriving the Modified Equation—can often lead to insight into how the approximate solution behaves

Computational Fluid Dynamics

Derive modified equation for upwind difference method:

$$\frac{f_j^{n+1}-f_j^n}{\Delta t}+\frac{U}{h}(f_j^n-f_{j-1}^n)=0$$

Using Taylor expansion:

$$f_j^{n+1}=f_j^n+\frac{\partial f}{\partial t}\Delta t+\frac{\partial^2 f}{\partial t^2}\frac{\Delta t^2}{2}+\frac{\partial^3 f}{\partial t^3}\frac{\Delta t^3}{6}+\cdots$$

$$f_{j-1}^n=f_j^n-\frac{\partial f}{\partial x}h+\frac{\partial^2 f}{\partial x^2}\frac{h^2}{2}-\frac{\partial^3 f}{\partial x^3}\frac{h^3}{6}+\cdots$$

Computational Fluid Dynamics

Substituting

$$\frac{1}{\Delta t}\left\{\left[\cancel{f_j^n}+\frac{\partial f}{\partial t}\Delta t+\frac{\partial^2 f}{\partial t^2}\frac{\Delta t^2}{2}+\frac{\partial^3 f}{\partial t^3}\frac{\Delta t^3}{6}+\cdots\right]-\cancel{f_j^n}\right\}$$
$$+\frac{U}{h}\left\{\cancel{f_j^n}-\left[\cancel{f_j^n}-\frac{\partial f}{\partial x}h+\frac{\partial^2 f}{\partial x^2}\frac{h^2}{2}-\frac{\partial^3 f}{\partial x^3}\frac{h^3}{6}+\cdots\right]\right\}=0$$

Therefore,

$$\frac{\partial f}{\partial t}+U\frac{\partial f}{\partial x}=-\frac{\Delta t}{2}f_{tt}+\frac{Uh}{2}f_{xx}-\frac{\Delta t^2}{6}f_{ttt}-\frac{Uh^2}{6}f_{xxx}+\cdots$$

It helps the interpretation if all terms are written in f_{xx}, f_{xxx}

Computational Fluid Dynamics

Taking further derivatives (first in time, then in space):

$$f_{tt} + Uf_{xt} = -\frac{\Delta t}{2} f_{ttt} + \frac{Uh}{2} f_{xxt} - \frac{\Delta t^2}{6} f_{tttt} - \frac{Uh^2}{6} f_{xxxt} + \cdots$$

$$+ \quad -Uf_{tx} - U^2 f_{xx} = \frac{U\Delta t}{2} f_{ttx} - \frac{U^2 h}{2} f_{xxx} + \frac{U\Delta t^2}{6} f_{tttx} + \frac{U^2 h^2}{6} f_{xxxx} + \cdots$$

$$f_{tt} = U^2 f_{xx} + \Delta t\left(\frac{-f_{ttt}}{2} + \frac{U}{2} f_{ttx} + O(\Delta t)\right)$$
$$+\Delta x\left(\frac{U}{2} f_{xxt} - \frac{U^2}{2} f_{xxx} + O(h)\right)$$

Computational Fluid Dynamics

Similarly, we get

$$f_{ttt} = -U^3 f_{xxx} + O(\Delta t, h)$$
$$f_{ttx} = U^2 f_{xxx} + O(\Delta t, h)$$
$$f_{xxt} = -Uf_{xxx} + O(\Delta t, h)$$

Final form of the modified equation: $\lambda = \frac{U\Delta t}{h}$

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = \frac{Uh}{2}(1-\lambda) f_{xx} - \frac{Uh^2}{6}\left(2\lambda^2 - 3\lambda + 1\right) f_{xxx}$$
$$+O\left[h^3, h^2\Delta t, h\Delta t^2, \Delta t^3\right]$$

Computational Fluid Dynamics

By applying upwind differencing, we are effectively solving:

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = \frac{Uh}{2}(1-\lambda) f_{xx} - \frac{Uh^2}{6}\left(2\lambda^2 - 3\lambda + 1\right) f_{xxx} + \cdots$$

Numerical dissipation (diffusion)

Also note that the CFL condition $\lambda = \frac{U\Delta t}{h} < 1$

ensures a positive diffusion coefficient

$(\cdots) f_{xx}$ Dissipation

$(\cdots) f_{xxx}$ Dispersion

Computational Fluid Dynamics

Consider: $\frac{\partial f}{\partial t} = \alpha \frac{\partial^n f}{\partial x^n}$

Look for solutions of the form:

$$f(x,t) = a(t) e^{ikx}$$

Substitute to get

$$\frac{da(t)}{dt} = (ik)^n \alpha a(t)$$

solve

$$a(t) = a_o e^{(ik)^n \alpha t}$$

For even n we get diffusion, for odd n we get dispersion

Computational Fluid Dynamics

First-Order Methods and Diffusion $\lambda = \frac{U\Delta t}{h}$

Upwind:

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = \frac{Uh}{2}(1-\lambda) f_{xx} - \frac{Uh^2}{6}\left(2\lambda^2 - 3\lambda + 1\right) f_{xxx}$$

Lax-Friedrichs:

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = \frac{Uh}{2}\left(\frac{1}{\lambda} - \lambda\right) f_{xx} + \frac{Uh^2}{3}\left(1-\lambda^2\right) f_{xxx}$$

Dissipative = Smearing

Computational Fluid Dynamics

Second-Order Methods and Dispersion $\lambda = \frac{U\Delta t}{h}$

Lax-Wendroff:

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = -\frac{Uh^2}{2}\left(1-\lambda^2\right) f_{xxx} - \frac{Uh^3}{8}\lambda\left(1-\lambda^2\right) f_{xxxx}$$

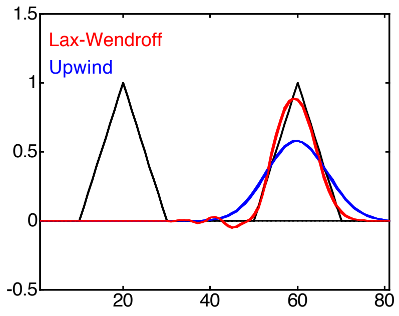
Beam-Warming:

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = \frac{Uh^2}{6}(1-\lambda)(2-\lambda) f_{xxx} - \frac{Uh^3}{8}(1-\lambda)^2(2-\lambda) f_{xxxx}$$

Dispersive = Wiggles

Computational Fluid Dynamics

dt=0.25*h



Computational Fluid Dynamics

First-Order Methods and Diffusion

$$\lambda = \frac{U\Delta t}{h}$$

Upwind:

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = \frac{Uh}{2}(1-\lambda)f_{xx} - \frac{Uh^2}{6}\left(2\lambda^2 - 3\lambda + 1\right)f_{xxx}$$

Dissipative = Smearing

Lax-Wendroff:

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = -\frac{Uh^2}{2}\left(1-\lambda^2\right)f_{xxx} - \frac{Uh^3}{8}\lambda\left(1-\lambda^2\right)f_{xxxx}$$

Dispersive = Wiggles

Computational Fluid Dynamics

Godunov Theorem

Computational Fluid Dynamics
Godunov's Theorem

Question: How can we avoid oscillatory behavior?

(How can we preserve monotonicity)

Godunov Theorem (1959):

"Monotone behavior of a numerical solution cannot be assured for linear finite-difference methods with more than first-order accuracy."

Computational Fluid Dynamics
Godunov Theorem - 1

For linear equation

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = 0$$

we have: $\frac{\partial f}{\partial t} = -U\frac{\partial f}{\partial x}; \quad \frac{\partial^2 f}{\partial t^2} = U^2\frac{\partial^2 f}{\partial x^2}$

A general linear scheme can be written as:

$$f_j^{n+1} = \sum_k c_k f_{j+k}^n$$

Expanding f_{j+k}^n in Taylor series around f_j^n

$$f_{j+k}^n = f_j^n + kh\frac{\partial f}{\partial x} + \frac{k^2h^2}{2}\frac{\partial^2 f}{\partial x^2} + O(h^3)$$

Computational Fluid Dynamics
Godunov Theorem - 2

Substituting

$$f_j^{n+1} = f_j^n\sum_k c_k + h\frac{\partial f}{\partial x}\sum_k kc_k + \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2}\sum_k k^2c_k + O(h^3) \qquad \text{(a)}$$

Also, f_j^{n+1} can be Taylor expanded in time around f_j^n

$$f_j^{n+1} = f_j^n + \Delta t\frac{\partial f}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 f}{\partial t^2} + O(\Delta t^3)$$

$$= f_j^n - U\Delta t\frac{\partial f}{\partial x} + U^2\frac{\Delta t^2}{2}\frac{\partial^2 f}{\partial x^2} + O(\Delta t^3) \qquad \text{(b)}$$

(a)=(b) (2nd order accurate):

$$\sum_k c_k = 1; \quad \sum_k kc_k = -\frac{U\Delta t}{h}; \quad \sum_k k^2c_k = \left(\frac{U\Delta t}{h}\right)^2$$

Computational Fluid Dynamics
Godunov Theorem - 3

Condition for monotonicity:

If $f^n_{j+1} > f^n_j$, then $f^{n+1}_{j+1} > f^{n+1}_j$

Since $f_j^{n+1} = \sum_k c_k f^n_{j+k}$

$$f^{n+1}_{j+1} - f^{n+1}_j = \sum_k c_k \underbrace{\left(f^n_{j+k+1} - f^n_{j+k}\right)}_{>0}$$

The above should be valid for arbitrary f^n_j

$$c_k > 0 \iff f^{n+1}_{j+1} - f^{n+1}_j > 0$$

Is it possible?

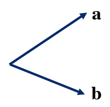
Computational Fluid Dynamics
Godunov Theorem - 4

$$\sum_k c_k = 1; \quad \sum_k k c_k = -\frac{U\Delta t}{h}; \quad \sum_k k^2 c_k = \left(\frac{U\Delta t}{h}\right)^2$$

Hence we get: $\left(\sum_k c_k\right)\left(\sum_k k^2 c_k\right) = \left(\sum_k k c_k\right)^2$

Define: $c_k = e_k^2 \;\; (c_k > 0)$

$$\left(\sum_k e_k^2\right)\left(\sum_k k^2 e_k^2\right) = \left(\sum_k k e_k^2\right)^2$$



or $(\mathbf{a}\cdot\mathbf{a})(\mathbf{b}\cdot\mathbf{b}) = (\mathbf{a}\cdot\mathbf{b})^2$

where $\mathbf{a} = (e_1, e_2, e_3, \ldots, e_k, \ldots); \quad \mathbf{b} = (1e_1, 2e_2, 3e_3, \ldots, ke_k, \ldots)$

This violates Cauchy inequality $(\mathbf{a}\cdot\mathbf{a})(\mathbf{b}\cdot\mathbf{b}) \geq (\mathbf{a}\cdot\mathbf{b})^2$

Monotone 2nd Order scheme is impossible!

Computational Fluid Dynamics

The key word in Godunov's theorem is linear. To overcome its limitations, look for nonlinear scheames:

1. Use central differencing and introduce numerical viscosity where needed (Artificial viscosity)

2. Limit the fluxes in such a way that oscillations are avoided (high-order Godunov methods, flux-corrected transport, TVD shemes, etc.)

Computational Fluid Dynamics

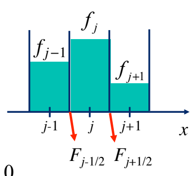
Godunov's Method

Computational Fluid Dynamics
Godunov Method

Godunov Method – Finite Volume Method + Riemann solver

Consider a 1-D Conservation Eq:

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial x}\left[F(f)\right] = 0$$



Integrating across the *j*-th cell:

$$\frac{\partial}{\partial t}\int_{x_{j-1/2}}^{x_{j+1/2}} f dx + F(x_{j+1/2}) - F(x_{j-1/2}) = 0$$

$$\Delta x_j = x_{j+1/2} - x_{j-1/2}$$

Define the cell-average: $f_{av} = \frac{1}{\Delta x_j}\int_{x_{j-1/2}}^{x_{j+1/2}} f\, dx$

$$\Delta x_j \frac{\partial f_{av}}{\partial t} + F(x_{j+1/2}) - F(x_{j-1/2}) = 0$$

Computational Fluid Dynamics
Godunov Method

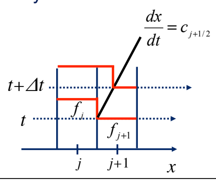
Integrating over time Δt

$$f_{av}^{n+1} = f_{av}^n - \frac{\Delta t}{\Delta x_j}\left[\frac{1}{\Delta t}\int_t^{t+\Delta t} F_{j+1/2}\, dt - \frac{1}{\Delta t}\int_t^{t+\Delta t} F_{j-1/2}\, dt\right]$$

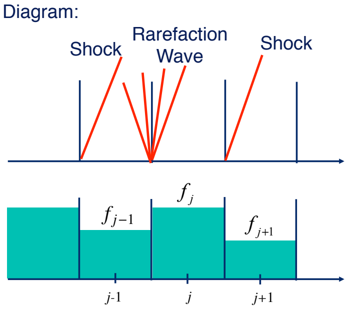
where the time integration is done by solving Riemann Problem for each cell boundary:

$$\frac{\partial f}{\partial t} + \frac{\partial}{\partial x}\left[F(f)\right] = 0$$

$$f(x,0) = \begin{cases} f_j & x \leq x_{j+1/2} \\ f_{j+1} & x > x_{j+1/2} \end{cases}$$



Wave Diagram:



Godunov Method: The Procedure

1. Construct a cell average value

$$f_{av} = \frac{1}{\Delta x_j} \int_{x_{j-1/2}}^{x_{j+1/2}} f\, dx$$

2. Solve a Riemann problem to find the time integration of fluxes

3. Construct new f_{av}

4. $t = t + \Delta t$ Go to 1.

Special Case: Linear Advection Equation

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = 0, \qquad U > 0$$

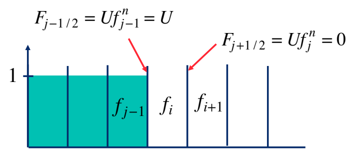
for which the Riemann problem is trivial and we get

$$F_{j+1/2} = Uf_j$$

$$f_{av,j}^{n+1} = f_{av,j}^{n} - \frac{U\Delta t}{h}\left(f_j^n - f_{j-1}^n\right)$$

1st Order Upwind Scheme

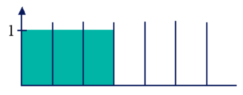
Consider the following initial conditions:



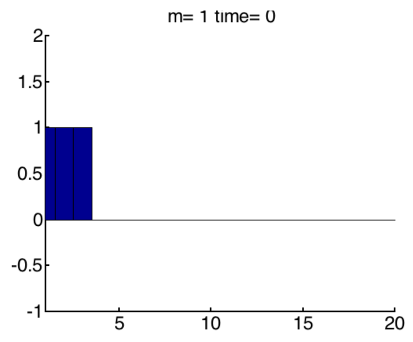
$$f_j^{n+1} = f_j^n - \frac{\Delta t}{h}(F_{j+1/2}^n - F_{j-1/2}^n)$$

Advect—Average—Advect—Average—etc

The averaging leads to numerical diffusion, even though the advection is exact!



Numerical versus exact solution



Computational Fluid Dynamics
Godunov Method

For nonlinear equations and systems the full Rieman problem must be solved
Since only the fluxes are needed, often approximate solvers are used

Computational Fluid Dynamics

Flux Vector Splitting

Computational Fluid Dynamics

For upwind schemes, it is necessary to determine the upstream direction. For systems with many characteristics running both left and right, there is not one “upstream” direction

Computational Fluid Dynamics
Upwind Scheme - Revisited

Generalized Upwind Scheme (for both $U > 0$ and $U < 0$)

$$f_j^{n+1} = f_j^n - \frac{U\Delta t}{h}(f_j^n - f_{j-1}^n), \;\; U > 0$$

$$f_j^{n+1} = f_j^n - \frac{U\Delta t}{h}(f_{j+1}^n - f_j^n), \;\; U < 0$$

Define

$$U^+ = \frac{1}{2}\left(U + |U|\right), \;\; U^- = \frac{1}{2}\left(U - |U|\right)$$

The two cases can be combined into a single expression:

$$f_j^{n+1} = f_j^n - \frac{\Delta t}{h}\left[U^+(f_j^n - f_{j-1}^n) + U^-(f_{j+1}^n - f_j^n)\right]$$

Where we have split the flux in an “upwind” and “downwind” part

Computational Fluid Dynamics

For a system of equations there are generally waves running in both directions. To apply upwinding, the fluxes must be decomposed into left and right running waves

Computational Fluid Dynamics
Flux Splitting

A system of hyperbolic equations

$$\frac{\partial \mathbf{f}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} = 0$$

Steger-Warming (1979)

can be written in the form

$$\frac{\partial \mathbf{f}}{\partial t} + [A]\frac{\partial \mathbf{f}}{\partial x} = 0; \qquad [A] = \frac{\partial \mathbf{F}}{\partial \mathbf{f}}$$

The system is hyperbolic if

$$[T]^{-1}[A][T] = [\lambda]; \quad [T]^{-1} = \begin{bmatrix} \mathbf{q}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{q}_N^{\mathrm{T}} \end{bmatrix}$$

Computational Fluid Dynamics
Flux Splitting

$$\mathbf{F} = [A]\mathbf{f} = [T][\lambda][T]^{-1}\mathbf{f}$$

The matrix of eigenvalues $[\lambda]$ is divided into two matrices

$$[\lambda] = [\lambda^+] + [\lambda^-]$$

Hence $[A] = [A^+] + [A^-] = [T][\lambda^+][T]^{-1} + [T][\lambda^-][T]^{-1}$

Define $\mathbf{F} = \mathbf{F}^+ + \mathbf{F}^-$ $\quad \left(\mathbf{F}^+ = [A^+]\mathbf{f},\ \ \mathbf{F}^- = [A^-]\mathbf{f}\right)$

Conservation law becomes

$$\frac{\partial \mathbf{f}}{\partial t} + \frac{\partial \mathbf{F}^+}{\partial x} + \frac{\partial \mathbf{F}^-}{\partial x} = 0$$

Computational Fluid Dynamics
Flux Splitting

Example: 1-D Hyperbolic Equation

$$\frac{\partial^2 f}{\partial t^2} - c^2\frac{\partial^2 f}{\partial x^2} = 0$$

Leading to

$$\begin{pmatrix} v_t \\ w_t \end{pmatrix} + \begin{pmatrix} 0 & -c^2 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} v_x \\ w_x \end{pmatrix} = 0 \qquad v = \frac{\partial f}{\partial t};\ \ w = \frac{\partial f}{\partial x}$$

$$\mathbf{f} = \begin{pmatrix} v \\ w \end{pmatrix};\ \ \mathbf{F} = \begin{pmatrix} -c^2 w \\ -v \end{pmatrix};\ \ [A] = \left[\frac{\partial \mathbf{F}}{\partial \mathbf{f}}\right] = \begin{bmatrix} 0 & -c^2 \\ -1 & 0 \end{bmatrix}$$

Computational Fluid Dynamics
Flux Splitting

$$[T]^{-1} = \begin{bmatrix} \mathbf{q}_1^{\mathrm{T}} \\ \mathbf{q}_2^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} 1 & -c \\ 1 & c \end{bmatrix} \qquad [\lambda^+] = \begin{bmatrix} c & 0 \\ 0 & 0 \end{bmatrix};\ \ [\lambda^-] = \begin{bmatrix} 0 & 0 \\ 0 & -c \end{bmatrix}$$

$$[A^+] = [T][\lambda^+][T]^{-1};\ \ [A^-] = [T][\lambda^-][T]^{-1}$$

$$\mathbf{F}^+ = [A^+]\mathbf{f};\ \ \mathbf{F}^- = [A^-]\mathbf{f}$$

Leading to: $\mathbf{F}^+ = \dfrac{1}{2}\begin{bmatrix} cv - c^2 w \\ -v + cw \end{bmatrix};\ \ \mathbf{F}^- = \dfrac{1}{2}\begin{bmatrix} -cv - c^2 w \\ -v - cw \end{bmatrix}$

For a nonlinear system of equations such as the Euler equations, there is some arbitrariness in how the system is split

Computational Fluid Dynamics

Solve using first order upwinding with flux splitting

$$\frac{\partial \mathbf{f}}{\partial t} + \frac{\partial \mathbf{F}^+}{\partial x} + \frac{\partial \mathbf{F}^-}{\partial x} = 0 \qquad \mathbf{F}^+ = \frac{1}{2}\begin{bmatrix} cv - c^2 w \\ -v + cw \end{bmatrix};\ \ \mathbf{F}^- = \frac{1}{2}\begin{bmatrix} -cv - c^2 w \\ -v - cw \end{bmatrix}$$

$$\begin{pmatrix} v_t \\ w_t \end{pmatrix} + \begin{pmatrix} 0 & -c^2 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} v_x \\ w_x \end{pmatrix} = 0 \qquad v = \frac{\partial f}{\partial t};\ \ w = \frac{\partial f}{\partial x}$$

Finite volume upwind approximation:

$$\begin{bmatrix} v \\ w \end{bmatrix}_j^{n+1} = \begin{bmatrix} v \\ w \end{bmatrix}_j^n - \frac{\Delta t}{h}\frac{1}{2}\left(\begin{bmatrix} cv - c^2 w \\ -v + cw \end{bmatrix}_j^n - \begin{bmatrix} cv - c^2 w \\ -v + cw \end{bmatrix}_{j-1}^n + \begin{bmatrix} -cv - c^2 w \\ -v - cw \end{bmatrix}_{j+1}^n - \begin{bmatrix} -cv - c^2 w \\ -v - cw \end{bmatrix}_j^n\right)$$

Computational Fluid Dynamics

Higher Order Schemes

Computational Fluid Dynamics
Higher Order Methods

Even though the fluxes were computed EXACTLY, the solution was very inaccurate due to numerical diffusion. Since the advection is exact, the problem must lie with the averaging of the solution and the assumption of a constant state in each cell

To generate more accurate schemes, we need to reconstruct a more accurate representation of the function in each cell.

Computational Fluid Dynamics
Higher Order Methods

For the first order spatial fluxes, the value in each cell was assumed to be a constant. For higher order computations of the spatial fluxes, the slope inside each cell is found

Once the value at the cell boundary has been found, the flux is found using these values

$$F_{j+1/2}^{n+1/2} = F\left(\left(f^L\right)_{j+1/2}^{n+1/2}, \left(f^R\right)_{j+1/2}^{n+1/2}\right)$$

Computational Fluid Dynamics
Higher Order Methods

For the Lax-Wendroff scheme the time and space discretization are closely linked.

In the Beam-Warming method, on the other hand, the time integration is independent of the spatial discretization

Computational Fluid Dynamics
Higher Order Methods

The separation of space and time discretization is generalized in the Method of Lines, where we convert the PDE into a set of ODEs for each grid point by writing:

$$\frac{df_j}{dt} = \frac{-1}{\Delta x}\left(F_{j+1/2}^n - F_{j-1/2}^n\right)$$

The time integration can, in principle, be done by any standard ODE solver, although in practice we often use second order Runge-Kutta methods

Computational Fluid Dynamics
Higher Order Methods

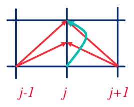
To get a second order time integration, use a half-step predictor to give second order accuracy in time:

$$f_j^{n+1/2} = f_j^n - \frac{\Delta t}{2\Delta x}\left(F_{j+1/2}^n - F_{j-1/2}^n\right)$$

Then correct by a midpoint rule:

$$f_j^{n+1} = f_j^n - \frac{\Delta t}{\Delta x}\left(F_{j+1/2}^{n+1/2} - F_{j-1/2}^{n+1/2}\right)$$

where the fluxes are found at the half step



Computational Fluid Dynamics
Higher Order Methods

For a method that is second order in time we therefore must find the fluxes twice. For the method on the last slide we find

$$F_{j+1/2}^n = F\left(f_j^n, f_{j+1}^n\right)$$

at the beginning of the step.

And

$$F_{j+1/2}^{n+1/2} = F\left(\left(f^L\right)_{j+1/2}^{n+1/2}, \left(f^R\right)_{j+1/2}^{n+1/2}\right)$$

at the half step. For a first order in space methods we take the values on either side to be the cell averages but for higher order methods the variables are extrapolated to the cell boundaries to give higher order fluxes.

Computational Fluid Dynamics

$$\frac{\partial f}{\partial t} + \frac{\partial F}{\partial x} = 0 \qquad \bar{f}_i(t) = \frac{1}{\Delta x}\int_{V_i} f(x,t)\,dx$$

$$\frac{d}{dt}\bar{f}_i(t) = \frac{1}{\Delta x}\Big(F(f(x_{i+1/2},t)) - F(f(x_{i+1/2},t))\Big)$$

$$= \frac{1}{\Delta x}\left(F_{i+1/2} - F_{i-1/2}\right) = L(\bar{f})_i$$

For high order methods the time integration is often done using high order Runge-Kutta methods, such the following third order method:

$$\bar{f}^{(1)} = \bar{f}^n + \Delta t\, L(\bar{f}^n)$$

$$\bar{f}^{(2)} = \frac{3}{4}\bar{f}^n + \frac{1}{4}\bar{f}^{(1)} + \frac{1}{4}\Delta t\, L(\bar{f}^{(1)})$$

$$\bar{f}^{n+1} = \frac{1}{3}\bar{f}^n + \frac{2}{3}\bar{f}^{(2)} + \frac{2}{3}\Delta t\, L(\bar{f}^{(2)})$$

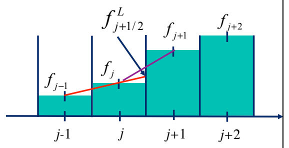
Computational Fluid Dynamics
Higher Order Methods

A fairly general family of higher order methods can be constructed by weighing the slopes in each cell in different ways:

Computational Fluid Dynamics
Higher Order Methods

Higher order fluxes

$$f^L_{j+1/2} = f_j + \frac{h}{2}S$$



Define slopes:

$$S^+ = \frac{f_{j+1} - f_j}{h} \qquad S^- = \frac{f_j - f_{j-1}}{h}$$

Define a weighted average:

$$f^L_{j+1/2} = f_j + \frac{h}{2}\left[\left(\frac{1-\kappa}{2}\right)S^- + \left(\frac{1+\kappa}{2}\right)S^+\right]$$

$k=0 \quad use \quad \tfrac{1}{2}\left(S^+ + S^-\right)$

$k=1 \quad use \quad S^+ \; only$

$k=-1 \quad use \quad S^- \; only$

Computational Fluid Dynamics
Higher Order Methods

Using

$$S^+ = \frac{f_{j+1} - f_j}{h} \qquad S^- = \frac{f_j - f_{j-1}}{h}$$

We find

$$f^L_{j+1/2} = f_j + \frac{h}{2}\left[\left(\frac{1-\kappa}{2}\right)S^- + \left(\frac{1+\kappa}{2}\right)S^+\right]$$

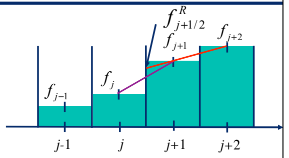
$$= f_j + \frac{h}{2}\left[\left(\frac{1-\kappa}{2}\right)\left(\frac{f_j - f_{j-1}}{h}\right) + \left(\frac{1+\kappa}{2}\right)\left(\frac{f_{j+1} - f_j}{h}\right)\right]$$

$$= f_j + \frac{1-\kappa}{4}\left(f_j - f_{j-1}\right) + \frac{1+\kappa}{4}\left(f_{j+1} - f_j\right)$$

Computational Fluid Dynamics
Higher Order Methods

Similarly

$$f^R_{j+1/2} = f_{j+1} - \frac{h}{2}S$$



$$S^+ = \frac{f_{j+2} - f_{j+1}}{h} \qquad S^- = \frac{f_{j+1} - f_j}{h}$$

$$f^R_{j+1/2} = f_{j+1} - \frac{1-\kappa}{4}\left(f_{j+1} - f_j\right) - \frac{1+\kappa}{4}\left(f_{j+2} - f_{j+1}\right)$$

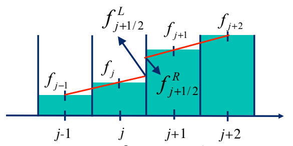
Computational Fluid Dynamics
Higher Order Methods

Evaluation of the function at the cell boundary:

$$f^L_{j+1/2} = f_j + \frac{1-\kappa}{4}\left(f_j - f_{j-1}\right) + \frac{1+\kappa}{4}\left(f_{j+1} - f_j\right)$$

$$f^R_{j+1/2} = f_{j+1} - \frac{1-\kappa}{4}\left(f_{j+1} - f_j\right) - \frac{1+\kappa}{4}\left(f_{j+2} - f_{j+1}\right)$$

Different values of k give different second order schemes



for $\kappa = -1$

Computational Fluid Dynamics
Higher Order Methods

$$f^L_{j+1/2} = f_j + \frac{1-\kappa}{4}\left(f_j - f_{j-1}\right) + \frac{1+\kappa}{4}\left(f_{j+1} - f_j\right)$$

Selecting k differently allows us to recover a number of standard schemes:

$\kappa = -1$	Second order upwind
$\kappa = 0$	Fromm’s scheme
$\kappa = 1/2$	QUICK
$\kappa = 1/3$	Third order upwind
$\kappa = 1$	Centered scheme

Computational Fluid Dynamics
Higher Order Methods

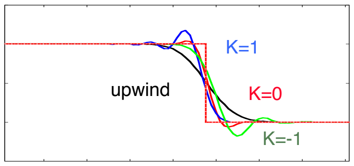
Predictor step $f_j^{n+1/2} = f_j^n - \frac{\Delta t}{2h}\left(F_{j+1/2}^n - F_{j-1/2}^n\right)$

Final step $f_j^{n+1} = f_j^n - \frac{\Delta t}{h}\left(F_{j+1/2}^{n+1/2} - F_{j-1/2}^{n+1/2}\right)$

Can use first order fluxes for the predictor step

where $F_{j+1/2}^{n+1/2} = F\left(\left(f^L\right)_{j+1/2}^{n+1/2}, \left(f^R\right)_{j+1/2}^{n+1/2}\right)$ is a higher order flux

Example: Linear Advection



Computational Fluid Dynamics
Higher Order Methods

$$f_{j+1/2}^L = f_j + \frac{1-\kappa}{4}\left(f_j - f_{j-1}\right) + \frac{1+\kappa}{4}\left(f_{j+1} - f_j\right)$$

Notice that we can also write the flux as

$$f_{j+1/2}^L = \frac{1}{2}\left(f_j + f_{j+1}\right) - \frac{1-\kappa}{4}\left(f_{j+1} - 2f_j + f_{j-1}\right)$$

Where the fluxes appear as centered approximations with a stabilizing artificial dissipation

Computational Fluid Dynamics
Higher Order Methods

An even more general family of schemes can be constructed by writing

$$f_{j+1/2}^L = f_j + \frac{1}{2}B\left(f_j - f_{j-1}, f_{j+1} - f_j\right)$$

Where B is a function of the slopes to the left and the right. Often we take

$$B\left(f_j - f_{j-1}, f_{j+1} - f_j\right) = \Psi(r)\left(f_j - f_{j-1}\right)$$

Where $r = \frac{f_{j+1} - f_j}{f_j - f_{j-1}}$

Later we will consider adjusting this function. Sometimes called limiter

Giving

$$f_{j+1/2}^L = f_j + \frac{1}{2}\Psi(r)\left(f_j - f_{j-1}\right)$$

Computational Fluid Dynamics
Higher Order Methods

The k scheme can be written in terms of the limiter

$$\Psi(r) = \frac{1+\kappa}{2}r + \frac{1-\kappa}{2} \qquad r = \frac{f_{j+1} - f_j}{f_j - f_{j-1}}$$

As can be seen by substitution:

$$f_{j+1/2}^L = f_j + \frac{1}{2}\Psi(r)\left(f_j - f_{j-1}\right)$$
$$= f_j + \frac{1}{2}\left(\frac{1+\kappa}{2}r + \frac{1-\kappa}{2}\right)\left(f_j - f_{j-1}\right)$$
$$= f_j + \frac{1}{2}\left(\frac{1+\kappa}{2}\left(\frac{f_{j+1} - f_j}{f_j - f_{j-1}}\right) + \frac{1-\kappa}{2}\right)\left(f_j - f_{j-1}\right)$$
$$= f_j + \frac{1+\kappa}{4}\left(f_{j+1} - f_j\right) + \frac{1-\kappa}{4}\left(f_j - f_{j-1}\right)$$

Computational Fluid Dynamics
Higher Order Methods

For the k scheme we have:

$$f_{j+1/2}^L = f_j + \frac{1}{2}\Psi(r)\left(f_j - f_{j-1}\right) \qquad r = \frac{f_{j+1} - f_j}{f_j - f_{j-1}}$$

$$\Psi(r) = \frac{1+\kappa}{2}r + \frac{1-\kappa}{2}$$

$\Psi(r) = \frac{1}{2};$	$\kappa = -1$	Second order upwind
$\Psi(r) = \frac{r}{2} + \frac{1}{2};$	$\kappa = 0$	Fromm’s scheme
$\Psi(r) = r;$	$\kappa = 1$	Centered scheme

Computational Fluid Dynamics
Higher Order Methods

$$f_{j+1/2}^L = f_j + \frac{1}{2}\Psi(r)\left(f_j - f_{j-1}\right) \qquad r = \frac{f_{j+1} - f_j}{f_j - f_{j-1}}$$

We can also write other schemes in this way. For example:

$\Psi(r) = 0$	First order upwind
$\Psi(r) = 1$	Lax-Wendroff
$\Psi(r) = r$	Beam-Warming

Computational Fluid Dynamics

But what about the oscillations?

Computational Fluid Dynamics

Artificial Viscosity

Computational Fluid Dynamics
Artificial Viscosity

Use centered difference method (e.g. Lax-Wendroff)
- Second order accuracy
- Oscillation near discontinuity

In order to “damp out” oscillation, we can either
- Use implicit numerical viscosity (upwind) or
- Add an explicit numerical viscosity

Computational Fluid Dynamics
Artificial Viscosity

Von Neumann and Richtmyer (1950)

Consider: $$\frac{\partial f}{\partial t}+\frac{\partial F}{\partial x}=0$$

O(1) coefficient

Replace the flux by: $$F' = F-\alpha\frac{\partial f}{\partial x};\quad \alpha = \boxed{D}h^2\left|\frac{\partial f}{\partial x}\right|$$

Giving: $$\frac{\partial f}{\partial t}+\frac{\partial F}{\partial x}=-\frac{\partial}{\partial x}\left(-\alpha\frac{\partial f}{\partial x}\right)=\underbrace{\frac{\partial}{\partial x}\left(Dh^2\left|\frac{\partial f}{\partial x}\right|\frac{\partial f}{\partial x}\right)}_{\to 0 \text{ as } h\to 0}$$

- Simulates the effect of the physical viscosity on the grid scale, concentrated around discontinuity and negligible elsewhere.
- h^2 is necessary to keep the viscous term of higher order.

Computational Fluid Dynamics
Artificial Viscosity

Example: Linear Advection Equation:

$$\frac{\partial f}{\partial t}+U\frac{\partial f}{\partial x}=\frac{\partial}{\partial x}\left(Dh^2\left|\frac{\partial f}{\partial x}\right|\frac{\partial f}{\partial x}\right)$$

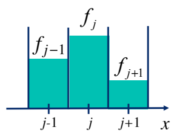
Discretization by Lax-Wendroff

$$f_j^{n+1}=f_j^n-\frac{U\Delta t}{2h}\left(f_{j+1}^n-f_{j-1}^n\right)+\frac{1}{2}\left(\frac{U\Delta t}{h}\right)^2\left(f_{j+1}^n-2f_j^n+f_{j-1}^n\right)$$
$$+\frac{\Delta t}{h}\left[\left(Dh^2\left|\frac{\partial f}{\partial x}\right|\frac{\partial f}{\partial x}\right)_{j+1/2}-\left(Dh^2\left|\frac{\partial f}{\partial x}\right|\frac{\partial f}{\partial x}\right)_{j-1/2}\right]$$

Computational Fluid Dynamics
Artificial Viscosity

Approximate:

$$\left(Dh^2\left|\frac{\partial f}{\partial x}\right|\frac{\partial f}{\partial x}\right)_{j+1/2}=D\left|f_{j+1}^n-f_j^n\right|\left(f_{j+1}^n-f_j^n\right)$$



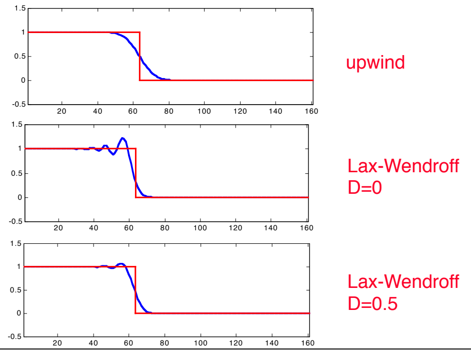
gives

$$f_j^{n+1}=f_j^n-\frac{U\Delta t}{2h}\left(f_{j+1}^n-f_{j-1}^n\right)+\frac{1}{2}\left(\frac{U\Delta t}{h}\right)^2\left(f_{j+1}^n-2f_j^n+f_{j-1}^n\right)$$
$$+D\frac{\Delta t}{h}\left[\left|f_{j+1}^n-f_j^n\right|\left(f_{j+1}^n-f_j^n\right)-\left|f_j^n-f_{j-1}^n\right|\left(f_j^n-f_{j-1}^n\right)\right]$$

Computational Fluid Dynamics
Artificial Viscosity

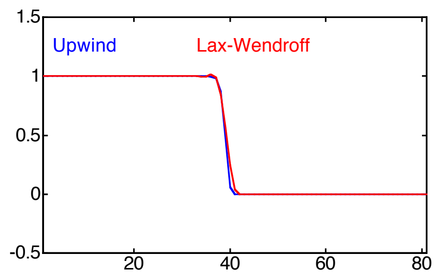
```
% Artificial Viscosity by LaxWendroff
n=161; nstep=250; length=4.0;h=length/(n-1);dt=0.25*h;
y=zeros(n,1);f=zeros(n,1);f(1)=1;time=0;

for m=1:nstep,m
  hold off;plot(f,'linewidt',2);  axis([1 n -0.5, 1.5]);hold on;
  plot([1,dt*(m-1)/h+1.5,dt*(m-1)/h+1.5,n],[1,1,0,0],'r','linewidt',2);pause;
  y=f; time=time+dt;
  for i=2:n-1,
    f(i)=y(i)-(0.5*dt/h)*(y(i+1)-y(i-1))+...
       (0.5*dt*dt/h/h)*(y(i+1)-2.0*y(i)+y(i-1))+...
                +0.5*(dt/h)*(abs(y(i+1)-y(i))*(y(i+1)-y(i))-...
                          abs(y(i)-y(i-1))*(y(i)-y(i-1)) );
  end;
end;
```

Computational Fluid Dynamics
Artificial Viscosity



Computational Fluid Dynamics

Nonlinear advection equation



Computational Fluid Dynamics

Artificial viscosity can be used with most other centered difference schemes and was, for a while, THE way aeronautical computations were done. Other types, including higher order, have been used.

Computational Fluid Dynamics

What we know so far:

First-order methods result in a low accuracy and dissipative solution, particularly at a shock. The solution does, however, not oscillate (preserves monotonicity).

Second-order methods capture shocks better but produce wiggly solutions.

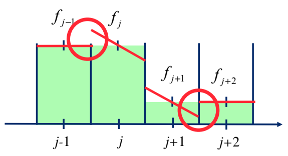
For the nonlinear advection equation, the first order captures the solution reasonably well, since the characteristics terminate in the shock

Computational Fluid Dynamics

The “Modern” view

Computational Fluid Dynamics
Oscillations

The formulation of second order schemes in terms of slopes allows us to understand the reason for oscillations near discontinuity. The construction of slopes leads to overshoots:



Computational Fluid Dynamics
Oscillations

To prevent oscillations near shocks when using high order schemes, we have already talked about artificial viscosity where we attempt to smooth out oscillations around the shocks.

The more modern approach is to prevent the apparence of oscillations by either:

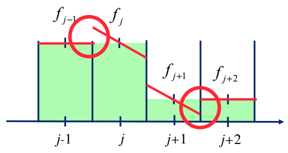
Limit the slopes when the variables are extrapolated to the cell boundaries

Limit the fluxes near shocks to prevent under or over shoot

Computational Fluid Dynamics
Oscillations

Near shocks the linear reconstruction leads to over and undershoots.

To prevent oscillations, apply LIMITERS to reduce the slopes where they will cause oscillations but leave then where they do not



Many possible limiters can be designed

Computational Fluid Dynamics
Oscillations

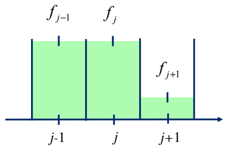
Example: Linear Advection Equation

$$\frac{\partial f}{\partial t} + U\frac{\partial f}{\partial x} = 0 \qquad F = Uf \qquad F_{j+1/2} = Uf^L_{j+1/2}$$

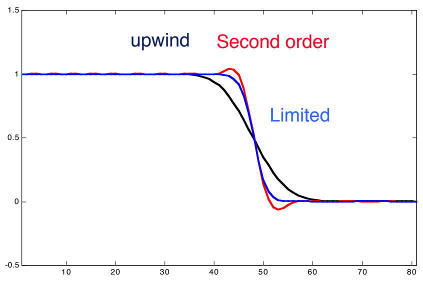
$$r_{j+1/2} = \frac{f_j - f_{j-1}}{f_{j+1} - f_j}$$

$$\Psi(r) = \frac{r + |r|}{1 + r}$$

$$\Psi = 2 \text{ if } f_{j+1} - f_j = 0$$



Computational Fluid Dynamics
Oscillations



Computational Fluid Dynamics
Oscillations

The realization that we could enforce monotonicity in otherwise high order schemes using nonlinear limiters revolutionized the computation of flows with shocks. The idea originated with Bram van Leer and Jay Boris but has since been taken further by a large number of researchers. It is still an active area of research.

A large number of limiters have been proposed and we will examine a few of those in more detail.

Diffusion versus dispersion

Godunov Theorem

Godunov Method

Upwind and Flux Splitting

Higher order schemes

Artificial viscosity