



Computational Fluid Dynamics

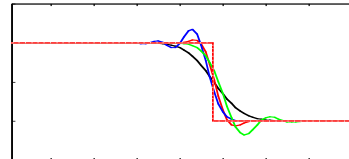
Lecture 9
February 8, 2017

Grétar Tryggvason



Godunov Theorem (1959):

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



Flux Limiters Review



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:

$$\frac{df_j}{dt} = \frac{-1}{h} (F_{j+1/2}^n - F_{j-1/2}^n)$$

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



For the Linear Advection equation we have:

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

$$\Psi(r) = \frac{1}{2}; \quad \text{Second order upwind}$$

$$\Psi(r) = \frac{r}{2} + \frac{1}{2}; \quad \text{Fromm's scheme}$$

$$\Psi(r) = r; \quad \text{Centered scheme (Beam-Warming)}$$

$$\Psi(r) = 1 \quad \text{Lax-Wendroff}$$

$$\Psi(r) = 0 \quad \text{First order upwind}$$



Designing Limiters: The Sweby diagram



Computational Fluid Dynamics Limiters

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

$r < 0$: local change of sign of the slopes
 $r < 1$: slope change
 $r = 1$: same slopes (linear f)
 $r > 1$: slope change

$r < 0$: revert to low order fluxes by taking the limiter to be zero
 $r = 1$: take the limiter to be 1



Computational Fluid Dynamics Limiters

To design schemes that prevent the emergence of unphysical oscillations, we need a precise definition of what we mean by no oscillations. A few such criteria have been proposed but the most widely used one is based on the Total Variation, defined by

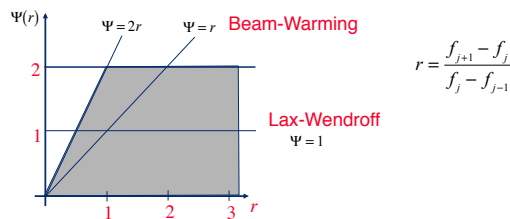
$$TV = \int \left| \frac{\partial f}{\partial x} \right| dx$$

It can be shown that many equations have the property that the Total Variation of the solution cannot grow in time. Schemes that preserve that behavior are generally referred to as Total Variation Diminishing (TVD) schemes.



Computational Fluid Dynamics Limiters

It can be shown that for a scheme to be second order and TVD, the limiter must lie in the shaded region.



Computational Fluid Dynamics Limiters

It has been found that it is also best to take the limiter to be between Lax-Wendroff and Beam-Warming

$$0 \leq \Psi(r) \leq r \quad r \geq 0$$

Generally we also require the limiters to be symmetric

$$\frac{\Psi(r)}{r} = \Psi\left(\frac{1}{r}\right)$$

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

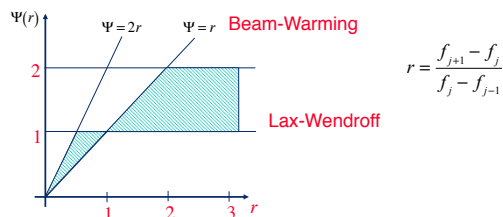
In many papers, r is defined as the inverse of the one used here



Computational Fluid Dynamics Limiters

Using the limitations given by second order accuracy, TVD and the requirement that the limiters lie between Lax-Wendroff and Beam-Warming, gives the Sweby-Diagram.

The limiters must lie in the shaded region

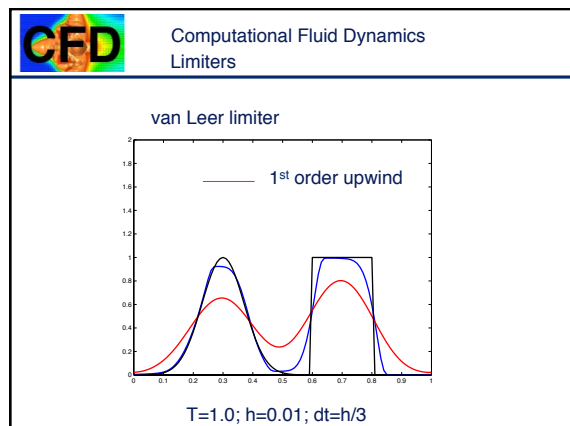
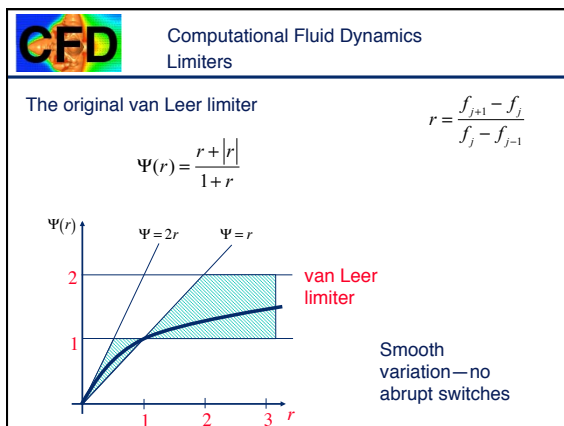
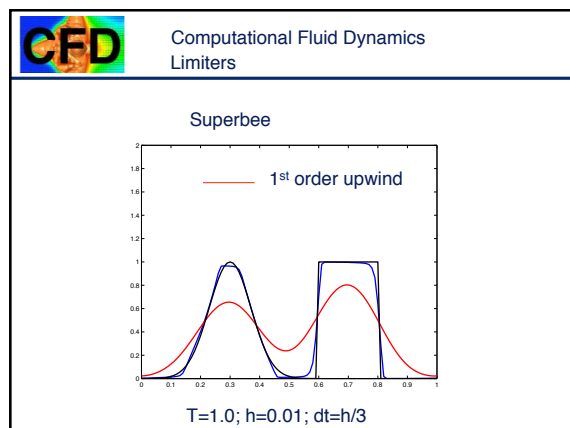
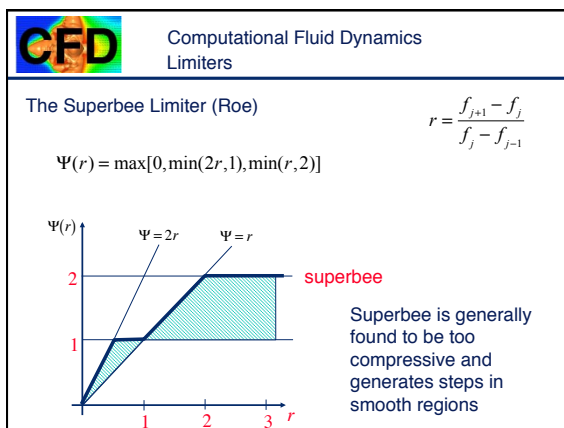
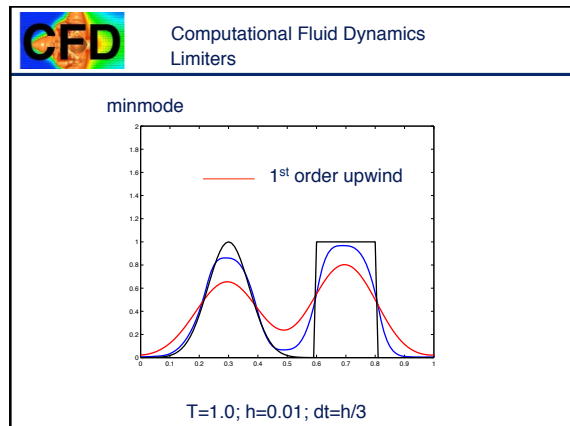
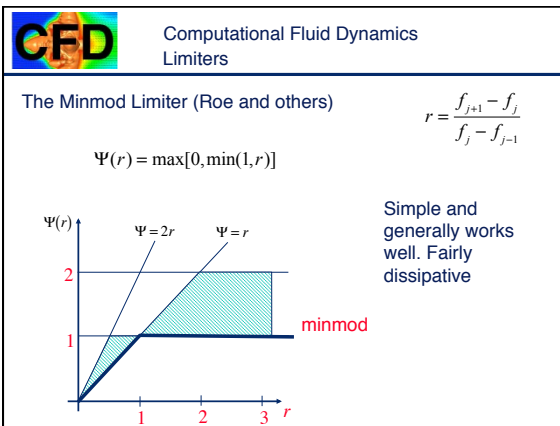


Computational Fluid Dynamics Limiters

The Sweby region is not the only one that is used to design limiters, but it is by far the most widely used. A very large number of limiters have been proposed that fall within this region. See, for example:

N. P. Waterson and H. Deconinck. Design Principles for bounded high-order convection schemes—a unified approach. JCP 224 (2007), 182-207

This paper treats only steady state problems.

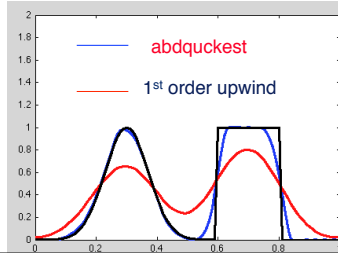


[illegible]



And many, many more!

V. G. Ferreira, F. A. Kurokawa, R.A.B. Queiroz, M.K. Kaibara, C.M. Oishi, J. A. Cuminato, A. Castelo, M.F. Tome and S. McKee. Assessment of a high-order finite difference upwind scheme for the simulation of convection-diffusion problems. Int. J. Numer. Meth. Fluids 2009; 60:1–26



Limiting the variables:

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

$$\begin{aligned} \text{Variables } f_{j+1/2}^L &= f_j^{n+1/2} + \frac{1}{2} \Psi^L(f_j^{n+1/2} - f_{j-1}^{n+1/2}) \\ f_{j+1/2}^R &= f_{j+1}^{n+1/2} - \frac{1}{2} \Psi^R(f_{j+1}^{n+1/2} - f_j^{n+1/2}) \end{aligned} \quad \kappa = -1$$

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

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



Limiting the Fluxes

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

$$\begin{aligned} \text{Variables } f_{j+1/2}^L &= f_j^{n+1/2} + \frac{1}{2} (f_j^{n+1/2} - f_{j-1}^{n+1/2}) \\ f_{j+1/2}^R &= f_{j+1}^{n+1/2} - \frac{1}{2} (f_{j+1}^{n+1/2} - f_j^{n+1/2}) \end{aligned} \quad \kappa = -1$$

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

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



Higher Order and More Recent Methods



In many cases we have solutions that require a high order method away from the discontinuity to represent a rapidly varying but smooth solution.

Beyond linear: Reconstruction of higher order approximations for the function in each cell (ENO and WENO).

The critical step in the methods discussed so far is the construction of a linear slope in each cell and the limitation of this slope to prevent oscillations. For higher order methods, a higher order profile needs to be constructed



ENO

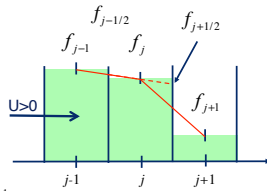
Essential Non-Oscillatory

Introduced in: A. Harten, B. Engquist, S. Osher, S.R. Chakravarty, Some results on high-order accurate essentially non-oscillatory schemes, Appl. Numer. Math. 2, 347–377 (1986).

A. Harten, B. Engquist, S. Osher, S.R. Chakravarty, Uniformly high order accurate essentially non-oscillatory schemes, J. Comput. Phys. 71(2), 231–303 (1987).



1. Construct left and right slopes by connecting the average values in adjacent cells
2. Select the downstream flux by using the smaller slope



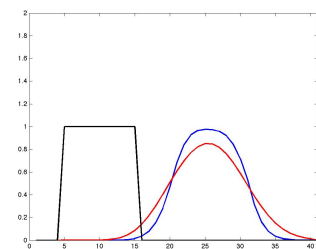
$$\Delta f_j^+ = f_{j+1} - f_j \quad \Delta f_j^- = f_j - f_{j-1} \quad \begin{matrix} j-1 & j & j+1 \end{matrix}$$

$$f_{j+1/2} = f_j + \frac{1}{2} \text{amin}(\Delta f_j^+, \Delta f_j^-) \quad \text{amin}(a, b) = \begin{cases} a, & |a| < |b| \\ b, & |b| \leq |a| \end{cases}$$

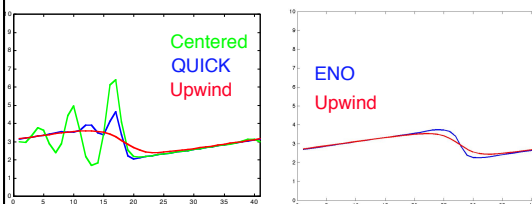

$$\begin{aligned} \frac{\partial f}{\partial t} + u \frac{\partial f}{\partial x} &= 0 \\ \text{amin}(a, b) &= \begin{cases} a, & |a| < |b| \\ b, & |b| \leq |a| \end{cases} \\ f_j^* &= f_j^n - \frac{\Delta t}{h} u_j^n (f_{j+1/2}^n - f_{j-1/2}^n) \\ f_j^{n+1} &= f_j^n - \frac{\Delta t}{h} \frac{1}{2} \left(u_j^n (f_{j+1/2}^n - f_{j-1/2}^n) + u_j^* (f_{j+1/2}^* - f_{j-1/2}^*) \right) \end{aligned}$$

$$f_{j+1/2} = \begin{cases} f_j + \frac{1}{2} \text{amin}(\Delta f_j^+, \Delta f_j^-), & \text{if } \frac{1}{2}(u_j + u_{j+1}) > 0 \\ f_j - \frac{1}{2} \text{amin}(\Delta f_{j+1}^+, \Delta f_{j+1}^-), & \text{if } \frac{1}{2}(u_j + u_{j+1}) < 0 \end{cases}$$

$$\Delta f_j^+ = f_{j+1} - f_j \quad \Delta f_j^- = f_j - f_{j-1}$$

[illegible]
$$\frac{\partial f}{\partial t} + u \frac{\partial f}{\partial x} = 0$$
Red: 1st Upwind
$$\frac{\partial f}{\partial t} + \frac{1}{2} \frac{\partial f^2}{\partial x} = D \frac{\partial^2 f}{\partial x^2}$$

Re cell=20



WENO

Review: C-W Shu. High Order Weighted Essential Nonoscillatory Schemes for Convection Dominated Problems. SIAM Review, Vol. 51 (2009), 82-126.



Computational Fluid Dynamics

$$\frac{df_j}{dt} + \frac{1}{\Delta x} (F_{j+1/2} - F_{j-1/2}) = 0$$

The time integration is done by a third order Runge-Kutta

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

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

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

where $L(f, t) = -\frac{\partial F}{\partial x}$



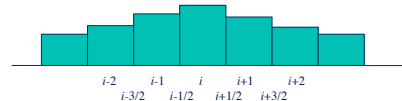
Computational Fluid Dynamics ENO/WENO

Constructing an interpolation polynomial from the cell averages: For anything higher than second order (linear) the problem is that the average value in the cell is not equal to the value at the center.

To get around this we look at the primitive function:

$$V(x) = \int_{-\infty}^x f(\xi) d\xi$$

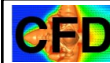
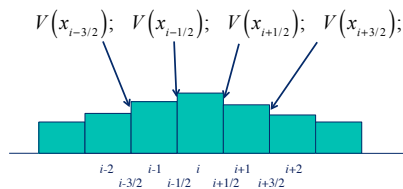
The lower bound is arbitrary and can be replaced



Computational Fluid Dynamics ENO/WENO

Since this is the integral over the cells, the discrete version is exact at the cell boundaries

$$V(x_{i+1/2}) = \sum_{j=-\infty}^i \int_{x_{j-1/2}}^{x_{j+1/2}} f(\xi) d\xi = \sum_{j=-\infty}^i \bar{f}_j \Delta x$$



Computational Fluid Dynamics ENO/WENO

A polynomial interpolating the edge values is given by $p(x)$ and we denote its derivative by $p'(x)$

$$p(x) = P'(x)$$

Then it can be shown that

$$\begin{aligned} \int_{x_{i-1/2}}^{x_{i+1/2}} p(\xi) d\xi &= \int_{x_{i-1/2}}^{x_{i+1/2}} P'(\xi) d\xi = P(x_{i+1/2}) - P(x_{i-1/2}) \\ &= V(x_{i+1/2}) - V(x_{i-1/2}) = \int_{x_{i-1/2}}^{x_{i+1/2}} f(\xi) d\xi = \bar{f}_i \Delta x \end{aligned}$$

That is, the integral of $p(x)$ over the cell is equal to the cell average \bar{f}_i



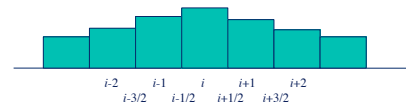
Computational Fluid Dynamics ENO/WENO

Thus, $p(x)$ gives the correct average value in each cell and the integrated value gives the exact values of the primitive function at the cell boundaries.

We need to write down a polynomial $P(x)$ that interpolates the values of the primitive function of the cell boundaries and then differentiate this polynomial to get $p(x)$, which lets us compute the variables at the cell boundary



Computational Fluid Dynamics ENO/WENO



The interpolation polynomial is often taken to be the Lagrangian Polynomial

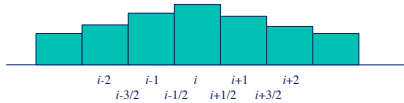
$$P(x) = \sum_{m=0}^k V(x_{i-r+m-1/2}) \prod_{\substack{l=0 \\ l \neq m}}^k \frac{x - x_{i-r+l-1/2}}{x_{i-r+m-1/2} - x_{i-r+l-1/2}}$$

Where r determines where we start and k is the order



Computational Fluid Dynamics ENO/WENO

ENO: Essential Non-Oscillatory



The question is now which point we select. We start by interpolating over one cell (linear). To add one point we can add either the point to the left or the right. In ENO we select the points based on the minimum absolute value of the divided differences of the function values



Computational Fluid Dynamics ENO/WENO

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

$$\bar{f}_i(t) = \frac{1}{\Delta x} \int_{V_i} f(x, t) dx$$

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

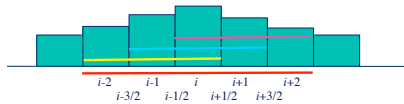
Find the cell average by integrating polynomial representation of the function, first over a subset of the points and then over all the points

$$\bar{f}_{i+l} = \frac{1}{\Delta x} \int_{V_{i+l}} p_j(x) dx; \quad l = -k + j, \dots, j$$

$$\bar{f}_{i+l} = \frac{1}{\Delta x} \int_{V_{i+l}} Q(x) dx; \quad l = -k, \dots, k$$



Computational Fluid Dynamics ENO/WENO



For a polynomial of order three, over the three intervals indicated we get

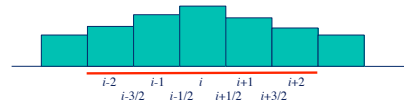
$$p_1(x_{i+1/2}) = \frac{1}{3} \bar{f}_{i-2} - \frac{7}{6} \bar{f}_{i-1} + \frac{11}{6} \bar{f}_i$$

$$p_2(x_{i+1/2}) = -\frac{1}{6} \bar{f}_{i-1} + \frac{5}{6} \bar{f}_i + \frac{1}{3} \bar{f}_{i+1}$$

$$p_3(x_{i+1/2}) = \frac{1}{3} \bar{f}_i + \frac{5}{6} \bar{f}_{i+1} - \frac{1}{6} \bar{f}_{i+2}$$



Computational Fluid Dynamics ENO/WENO



A polynomial for the whole interval is given by

$$Q(x_{i+1/2}) = \frac{1}{30} \bar{f}_{i-2} - \frac{13}{60} \bar{f}_{i-1} + \frac{47}{60} \bar{f}_i + \frac{9}{20} \bar{f}_{i+1} - \frac{1}{20} \bar{f}_{i+2}$$

Which is a weighted average of the lower order polynomials

$$Q(x_{i+1/2}) = \sum_{j=1}^k \gamma_j p_j(x_{i+1/2}) \quad \gamma_1 = \frac{1}{10}; \quad \gamma_2 = \frac{6}{10}; \quad \gamma_3 = \frac{3}{10};$$



Computational Fluid Dynamics ENO/WENO

Introduce a smoothness measure

$$\beta_j = \sum_{l=1}^k \int_{V_l} \Delta x^{2l-1} \left(\frac{d^l}{dx^l} p_j(x) \right)^2 dx$$

For our case this gives:

$$\beta_1 = \frac{13}{12} (\bar{f}_{j-2} - 2\bar{f}_{j-1} + \bar{f}_j)^2 + \frac{1}{4} (\bar{f}_{j-2} - 4\bar{f}_{j-1} + 3\bar{f}_j)^2$$

$$\beta_2 = \frac{13}{12} (\bar{f}_{j-1} - 2\bar{f}_j + \bar{f}_{j+1})^2 + \frac{1}{4} (\bar{f}_{j-1} - \bar{f}_{j+1})^2$$

$$\beta_3 = \frac{13}{12} (\bar{f}_j - 2\bar{f}_{j+1} + \bar{f}_{j+2})^2 + \frac{1}{4} (3\bar{f}_j - 4\bar{f}_{j+1} + \bar{f}_{j+2})^2$$



Computational Fluid Dynamics ENO/WENO

Then compute weights to find the smoothest approximation to the value of f at the cell boundary.

First find:

$$\tilde{\omega}_j = \frac{\gamma_j}{(\epsilon + \beta_j)^2}; \quad \omega_j = \frac{\tilde{\omega}_j}{\sum_j \tilde{\omega}_j}$$

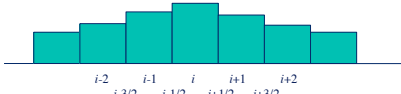
Then compute:

$$f_{i+1/2}^- \approx \sum_{j=0}^k \omega_j p_j(x_{i+1/2})$$

The value on the other side is found in the same way



Computational Fluid Dynamics ENO/WENO



In the WENO (weighted essentially non-oscillating) scheme we use all the points but weigh the contribution of each according to a smoothness criteria. High-order WENO represents the current state-of-the-art in computing of flows with sharp interfaces

Other smoothness criteria, weights, and interpolation functions have been studied, as well as how to implement the method on non-structured grids.



Computational Fluid Dynamics

Example: Third order WENO for the advection equation

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

The weights are

$$\tilde{\omega}_l = \frac{\gamma_l}{(\varepsilon + \beta_l)^2}$$

The semi-discrete equation is

$$\frac{df_j}{dt} + \frac{1}{\Delta x} (F_{j+1/2} - F_{j-1/2}) = 0$$

The fluxes are the weighted sum

$$F_{j+1/2} = \omega_1 F_{j+1/2}^{(1)} + \omega_2 F_{j+1/2}^{(2)}$$

where

$$F_{j+1/2}^{(1)} = -\frac{1}{2} F_{j-1} + \frac{3}{2} F_j$$

2nd order upwind

$$F_{j+1/2}^{(2)} = \frac{1}{2} F_j + \frac{1}{2} F_{j+1}$$

2nd order central

$$\beta_1 = (f_j - f_{j-1})^2$$

$$\beta_2 = (f_{j+1} - f_j)^2$$

$$\gamma_1 = \frac{1}{3}; \quad \gamma_2 = \frac{2}{3}$$

$$\varepsilon = 10^{-6}$$



Computational Fluid Dynamics

Why are the weights selected the way they are?

1. The linear weights $\gamma_1 = 1/3$ & $\gamma_2 = 2/3$ give a third order approximation in smooth regions. For smooth flows we want to recover the linear weights gamma $\omega_1 = \gamma_1$; $\omega_2 = \gamma_2$
2. Require $\omega_m > 0$; $\sum \omega_m = 1$
3. For a shock we want to get one of the lower order fluxes, so that $\omega_1 = 1$; $\omega_2 = 0$ or $\omega_1 = 0$; $\omega_2 = 1$

$$\omega_m = \gamma_m \quad \text{if } f(x) \text{ is smooth everywhere}$$

$$\omega_m = 0 \quad \text{if } f(x) \text{ has a discontinuity in the stencil spanned by } m$$



Computational Fluid Dynamics

$$\gamma_1 = \frac{1}{3}; \quad \gamma_2 = \frac{2}{3}$$

Linear weights, give a third order solution

$$\beta_1 = (f_j - f_{j-1})^2$$

$$\beta_2 = (f_{j+1} - f_j)^2$$

Smoothness indicators
(0 if fluxes are constant)

$$\tilde{\omega}_l = \frac{\gamma_l}{(\varepsilon + \beta_l)^2}$$

Modify the linear weights

$$\omega_m = \frac{\tilde{\omega}_m}{\sum_{l=1}^2 \tilde{\omega}_l}$$

Normalize the nonlinear weights

$$\varepsilon = 10^{-6}$$

A small number to avoid dividing by zero



Computational Fluid Dynamics

Fifth Order WENO or WENO5

The fluxes are the weighted sum

$$F_{j+1/2} = \omega_1 F_{j+1/2}^{(1)} + \omega_2 F_{j+1/2}^{(2)} + \omega_3 F_{j+1/2}^{(3)}$$

$$\gamma_1 = \frac{1}{16}; \quad \gamma_2 = \frac{5}{8}; \quad \gamma_3 = \frac{5}{16}$$

Smoothness Indicators

$$\beta_1 = \frac{1}{3} (4f_{j+2}^2 - 19f_{j+2}f_{j+1} + 25f_{j+1}^2 + 11f_{j+2}f_j - 31f_{j+1}f_j + 10f_j^2)$$

$$\beta_2 = \frac{1}{3} (4f_{j+1}^2 - 13f_{j+1}f_j + 13f_j^2 + 5f_{j+1}f_{j+1} - 13f_jf_{j+1} + 4f_{j+1}^2)$$

$$\beta_3 = \frac{1}{3} (10f_j^2 - 31f_jf_{j+1} + 25f_{j+1}^2 + 11f_jf_{j+2} - 19f_{j+1}f_{j+2} + 4f_{j+2}^2)$$

Fluxes

$$F_{j+1/2}^{(1)} = \frac{3}{8} F_{j-2} - \frac{5}{4} F_{j-1} + \frac{15}{8} F_j$$

$$F_{j+1/2}^{(2)} = -\frac{1}{8} F_{j-1} + \frac{3}{4} F_j + \frac{3}{8} F_{j+1}$$

$$F_{j+1/2}^{(3)} = \frac{3}{8} F_j + \frac{3}{4} F_{j+1} - \frac{1}{8} F_{j+2}$$

$$\tilde{\omega}_l = \frac{\gamma_l}{(\varepsilon + \beta_l)^2}$$

$$\omega_m = \frac{\tilde{\omega}_m}{\sum_{l=1}^3 \tilde{\omega}_l}$$



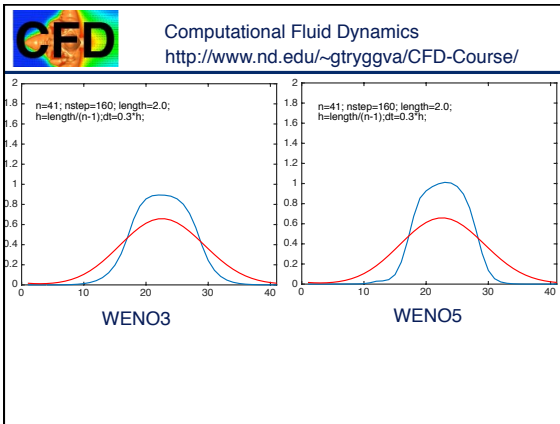
Computational Fluid Dynamics

<http://www.nd.edu/~gtryggva/CFD-Course/>

```
% one-dimensional linear advection by WENO
% 5th order scheme
N=100; length=2.0; n=length(N-1);
dx=length/N;
time=0.0;
```

```
gamma=1/3; gamma2=5/8; eps=0.000001;
```

```
% zeroth order
f=zeros(1,N); f(5)=zeros(N+1,1);
f(1)=zeros(N+1,1); f(2)=zeros(N+1,1);
f(N)=zeros(N+1,1); f(N)=zeros(N+1,1);
% first order
% for i=1:N-1, f(i)=1+0.5*sin(2*pi*i/N); end; % initial conditions
for i=1:N-1, f(i)=1; end
% Second Step
for j=2:N;
    beta1=(f(j)-f(j-1))^2; beta2=(f(j)-f(j-2))^2;
    omega1=gamma/(eps+beta1)^2; omega2=gamma2/(eps+beta2)^2;
    omega3=gamma/(eps+beta3)^2; omega4=gamma/(eps+beta4)^2;
    omega5=gamma/(eps+beta5)^2; omega6=gamma/(eps+beta6)^2;
    omega7=gamma/(eps+beta7)^2; omega8=gamma/(eps+beta8)^2;
    omega9=gamma/(eps+beta9)^2; omega10=gamma/(eps+beta10)^2;
    omega11=gamma/(eps+beta11)^2; omega12=gamma/(eps+beta12)^2;
    omega13=gamma/(eps+beta13)^2; omega14=gamma/(eps+beta14)^2;
    omega15=gamma/(eps+beta15)^2; omega16=gamma/(eps+beta16)^2;
    omega17=gamma/(eps+beta17)^2; omega18=gamma/(eps+beta18)^2;
    omega19=gamma/(eps+beta19)^2; omega20=gamma/(eps+beta20)^2;
    omega21=gamma/(eps+beta21)^2; omega22=gamma/(eps+beta22)^2;
    omega23=gamma/(eps+beta23)^2; omega24=gamma/(eps+beta24)^2;
    omega25=gamma/(eps+beta25)^2; omega26=gamma/(eps+beta26)^2;
    omega27=gamma/(eps+beta27)^2; omega28=gamma/(eps+beta28)^2;
    omega29=gamma/(eps+beta29)^2; omega30=gamma/(eps+beta30)^2;
    omega31=gamma/(eps+beta31)^2; omega32=gamma/(eps+beta32)^2;
    omega33=gamma/(eps+beta33)^2; omega34=gamma/(eps+beta34)^2;
    omega35=gamma/(eps+beta35)^2; omega36=gamma/(eps+beta36)^2;
    omega37=gamma/(eps+beta37)^2; omega38=gamma/(eps+beta38)^2;
    omega39=gamma/(eps+beta39)^2; omega40=gamma/(eps+beta40)^2;
    omega41=gamma/(eps+beta41)^2; omega42=gamma/(eps+beta42)^2;
    omega43=gamma/(eps+beta43)^2; omega44=gamma/(eps+beta44)^2;
    omega45=gamma/(eps+beta45)^2; omega46=gamma/(eps+beta46)^2;
    omega47=gamma/(eps+beta47)^2; omega48=gamma/(eps+beta48)^2;
    omega49=gamma/(eps+beta49)^2; omega50=gamma/(eps+beta50)^2;
    omega51=gamma/(eps+beta51)^2; omega52=gamma/(eps+beta52)^2;
    omega53=gamma/(eps+beta53)^2; omega54=gamma/(eps+beta54)^2;
    omega55=gamma/(eps+beta55)^2; omega56=gamma/(eps+beta56)^2;
    omega57=gamma/(eps+beta57)^2; omega58=gamma/(eps+beta58)^2;
    omega59=gamma/(eps+beta59)^2; omega60=gamma/(eps+beta60)^2;
    omega61=gamma/(eps+beta61)^2; omega62=gamma/(eps+beta62)^2;
    omega63=gamma/(eps+beta63)^2; omega64=gamma/(eps+beta64)^2;
    omega65=gamma/(eps+beta65)^2; omega66=gamma/(eps+beta66)^2;
    omega67=gamma/(eps+beta67)^2; omega68=gamma/(eps+beta68)^2;
    omega69=gamma/(eps+beta69)^2; omega70=gamma/(eps+beta70)^2;
    omega71=gamma/(eps+beta71)^2; omega72=gamma/(eps+beta72)^2;
    omega73=gamma/(eps+beta73)^2; omega74=gamma/(eps+beta74)^2;
    omega75=gamma/(eps+beta75)^2; omega76=gamma/(eps+beta76)^2;
    omega77=gamma/(eps+beta77)^2; omega78=gamma/(eps+beta78)^2;
    omega79=gamma/(eps+beta79)^2; omega80=gamma/(eps+beta80)^2;
    omega81=gamma/(eps+beta81)^2; omega82=gamma/(eps+beta82)^2;
    omega83=gamma/(eps+beta83)^2; omega84=gamma/(eps+beta84)^2;
    omega85=gamma/(eps+beta85)^2; omega86=gamma/(eps+beta86)^2;
    omega87=gamma/(eps+beta87)^2; omega88=gamma/(eps+beta88)^2;
    omega89=gamma/(eps+beta89)^2; omega90=gamma/(eps+beta90)^2;
    omega91=gamma/(eps+beta91)^2; omega92=gamma/(eps+beta92)^2;
    omega93=gamma/(eps+beta93)^2; omega94=gamma/(eps+beta94)^2;
    omega95=gamma/(eps+beta95)^2; omega96=gamma/(eps+beta96)^2;
    omega97=gamma/(eps+beta97)^2; omega98=gamma/(eps+beta98)^2;
    omega99=gamma/(eps+beta99)^2; omega100=gamma/(eps+beta100)^2;
    omega101=gamma/(eps+beta101)^2; omega102=gamma/(eps+beta102)^2;
    omega103=gamma/(eps+beta103)^2; omega104=gamma/(eps+beta104)^2;
    omega105=gamma/(eps+beta105)^2; omega106=gamma/(eps+beta106)^2;
    omega107=gamma/(eps+beta107)^2; omega108=gamma/(eps+beta108)^2;
    omega109=gamma/(eps+beta109)^2; omega110=gamma/(eps+beta110)^2;
    omega111=gamma/(eps+beta111)^2; omega112=gamma/(eps+beta112)^2;
    omega113=gamma/(eps+beta113)^2; omega114=gamma/(eps+beta114)^2;
    omega115=gamma/(eps+beta115)^2; omega116=gamma/(eps+beta116)^2;
    omega117=gamma/(eps+beta117)^2; omega118=gamma/(eps+beta118)^2;
    omega119=gamma/(eps+beta119)^2; omega120=gamma/(eps+beta120)^2;
    omega121=gamma/(eps+beta121)^2; omega122=gamma/(eps+beta122)^2;
    omega123=gamma/(eps+beta123)^2; omega124=gamma/(eps+beta124)^2;
    omega125=gamma/(eps+beta125)^2; omega126=gamma/(eps+beta126)^2;
    omega127=gamma/(eps+beta127)^2; omega128=gamma/(eps+beta128)^2;
    omega129=gamma/(eps+beta129)^2; omega130=gamma/(eps+beta130)^2;
    omega131=gamma/(eps+beta131)^2; omega132=gamma/(eps+beta132)^2;
    omega133=gamma/(eps+beta133)^2; omega134=gamma/(eps+beta134)^2;
    omega135=gamma/(eps+beta135)^2; omega136=gamma/(eps+beta136)^2;
    omega137=gamma/(eps+beta137)^2; omega138=gamma/(eps+beta138)^2;
    omega139=gamma/(eps+beta139)^2; omega140=gamma/(eps+beta140)^2;
    omega141=gamma/(eps+beta141)^2; omega142=gamma/(eps+beta142)^2;
    omega143=gamma/(eps+beta143)^2; omega144=gamma/(eps+beta144)^2;
    omega145=gamma/(eps+beta145)^2; omega146=gamma/(eps+beta146)^2;
    omega147=gamma/(eps+beta147)^2; omega148=gamma/(eps+beta148)^2;
    omega149=gamma/(eps+beta149)^2; omega150=gamma/(eps+beta150)^2;
    omega151=gamma/(eps+beta151)^2; omega152=gamma/(eps+beta152)^2;
    omega153=gamma/(eps+beta153)^2; omega154=gamma/(eps+beta154)^2;
    omega155=gamma/(eps+beta155)^2; omega156=gamma/(eps+beta156)^2;
    omega157=gamma/(eps+beta157)^2; omega158=gamma/(eps+beta158)^2;
    omega159=gamma/(eps+beta159)^2; omega160=gamma/(eps+beta160)^2;
    omega161=gamma/(eps+beta161)^2; omega162=gamma/(eps+beta162)^2;
    omega163=gamma/(eps+beta163)^2; omega164=gamma/(eps+beta164)^2;
    omega165=gamma/(eps+beta165)^2; omega166=gamma/(eps+beta166)^2;
    omega167=gamma/(eps+beta167)^2; omega168=gamma/(eps+beta168)^2;
    omega169=gamma/(eps+beta169)^2; omega170=gamma/(eps+beta170)^2;
    omega171=gamma/(eps+beta171)^2; omega172=gamma/(eps+beta172)^2;
    omega173=gamma/(eps+beta173)^2; omega174=gamma/(eps+beta174)^2;
    omega175=gamma/(eps+beta175)^2; omega176=gamma/(eps+beta176)^2;
    omega177=gamma/(eps+beta177)^2; omega178=gamma/(eps+beta178)^2;
    omega179=gamma/(eps+beta179)^2; omega180=gamma/(eps+beta180)^2;
    omega181=gamma/(eps+beta181)^2; omega182=gamma/(eps+beta182)^2;
    omega183=gamma/(eps+beta183)^2; omega184=gamma/(eps+beta184)^2;
    omega185=gamma/(eps+beta185)^2; omega186=gamma/(eps+beta186)^2;
    omega187=gamma/(eps+beta187)^2; omega188=gamma/(eps+beta188)^2;
    omega189=gamma/(eps+beta189)^2; omega190=gamma/(eps+beta190)^2;
    omega191=gamma/(eps+beta191)^2; omega192=gamma/(eps+beta192)^2;
    omega193=gamma/(eps+beta193)^2; omega194=gamma/(eps+beta194)^2;
    omega195=gamma/(eps+beta195)^2; omega196=gamma/(eps+beta196)^2;
    omega197=gamma/(eps+beta197)^2; omega198=gamma/(eps+beta198)^2;
    omega199=gamma/(eps+beta199)^2; omega200=gamma/(eps+beta200)^2;
    omega201=gamma/(eps+beta201)^2; omega202=gamma/(eps+beta202)^2;
    omega203=gamma/(eps+beta203)^2; omega204=gamma/(eps+beta204)^2;
    omega205=gamma/(eps+beta205)^2; omega206=gamma/(eps+beta206)^2;
    omega207=gamma/(eps+beta207)^2; omega208=gamma/(eps+beta208)^2;
    omega209=gamma/(eps+beta209)^2; omega210=gamma/(eps+beta210)^2;
    omega211=gamma/(eps+beta211)^2; omega212=gamma/(eps+beta212)^2;
    omega213=gamma/(eps+beta213)^2; omega214=gamma/(eps+beta214)^2;
    omega215=gamma/(eps+beta215)^2; omega216=gamma/(eps+beta216)^2;
    omega217=gamma/(eps+beta217)^2; omega218=gamma/(eps+beta218)^2;
    omega219=gamma/(eps+beta219)^2; omega220=gamma/(eps+beta220)^2;
    omega221=gamma/(eps+beta221)^2; omega222=gamma/(eps+beta222)^2;
    omega223=gamma/(eps+beta223)^2; omega224=gamma/(eps+beta224)^2;
    omega225=gamma/(eps+beta225)^2; omega226=gamma/(eps+beta226)^2;
    omega227=gamma/(eps+beta227)^2; omega228=gamma/(eps+beta228)^2;
    omega229=gamma/(eps+beta229)^2; omega230=gamma/(eps+beta230)^2;
    omega231=gamma/(eps+beta231)^2; omega232=gamma/(eps+beta232)^2;
    omega233=gamma/(eps+beta233)^2; omega234=gamma/(eps+beta234)^2;
    omega235=gamma/(eps+beta235)^2; omega236=gamma/(eps+beta236)^2;
    omega237=gamma/(eps+beta237)^2; omega238=gamma/(eps+beta238)^2;
    omega239=gamma/(eps+beta239)^2; omega240=gamma/(eps+beta240)^2;
    omega241=gamma/(eps+beta241)^2; omega242=gamma/(eps+beta242)^2;
    omega243=gamma/(eps+beta243)^2; omega244=gamma/(eps+beta244)^2;
    omega245=gamma/(eps+beta245)^2; omega246=gamma/(eps+beta246)^2;
    omega247=gamma/(eps+beta247)^2; omega248=gamma/(eps+beta248)^2;
    omega249=gamma/(eps+beta249)^2; omega250=gamma/(eps+beta250)^2;
    omega251=gamma/(eps+beta251)^2; omega252=gamma/(eps+beta252)^2;
    omega253=gamma/(eps+beta253)^2; omega254=gamma/(eps+beta254)^2;
    omega255=gamma/(eps+beta255)^2; omega256=gamma/(eps+beta256)^2;
    omega257=gamma/(eps+beta257)^2; omega258=gamma/(eps+beta258)^2;
    omega259=gamma/(eps+beta259)^2; omega260=gamma/(eps+beta260)^2;
    omega261=gamma/(eps+beta261)^2; omega262=gamma/(eps+beta262)^2;
    omega263=gamma/(eps+beta263)^2; omega264=gamma/(eps+beta264)^2;
    omega265=gamma/(eps+beta265)^2; omega266=gamma/(eps+beta266)^2;
    omega267=gamma/(eps+beta267)^2; omega268=gamma/(eps+beta268)^2;
    omega269=gamma/(eps+beta269)^2; omega270=gamma/(eps+beta270)^2;
    omega271=gamma/(eps+beta271)^2; omega272=gamma/(eps+beta272)^2;
    omega273=gamma/(eps+beta273)^2; omega274=gamma/(eps+beta274)^2;
    omega275=gamma/(eps+beta275)^2; omega276=gamma/(eps+beta276)^2;
    omega277=gamma/(eps+beta277)^2; omega278=gamma/(eps+beta278)^2;
    omega279=gamma/(eps+beta279)^2; omega280=gamma/(eps+beta280)^2;
    omega281=gamma/(eps+beta281)^2; omega282=gamma/(eps+beta282)^2;
    omega283=gamma/(eps+beta283)^2; omega284=gamma/(eps+beta284)^2;
    omega285=gamma/(eps+beta285)^2; omega286=gamma/(eps+beta286)^2;
    omega287=gamma/(eps+beta287)^2; omega288=gamma/(eps+beta288)^2;
    omega289=gamma/(eps+beta289)^2; omega290=gamma/(eps+beta290)^2;
    omega291=gamma/(eps+beta291)^2; omega292=gamma/(eps+beta292)^2;
    omega293=gamma/(eps+beta293)^2; omega294=gamma/(eps+beta294)^2;
    omega295=gamma/(eps+beta295)^2; omega296=gamma/(eps+beta296)^2;
    omega297=gamma/(eps+beta297)^2; omega298=gamma/(eps+beta298)^2;
    omega299=gamma/(eps+beta299)^2; omega300=gamma/(eps+beta300)^2;
    omega301=gamma/(eps+beta301)^2; omega302=gamma/(eps+beta302)^2;
    omega303=gamma/(eps+beta303)^2; omega304=gamma/(eps+beta304)^2;
    omega305=gamma/(eps+beta305)^2; omega306=gamma/(eps+beta306)^2;
    omega307=gamma/(eps+beta307)^2; omega308=gamma/(eps+beta308)^2;
    omega309=gamma/(eps+beta309)^2; omega310=gamma/(eps+beta310)^2;
    omega311=gamma/(eps+beta311)^2; omega312=gamma/(eps+beta312)^2;
    omega313=gamma/(eps+beta313)^2; omega314=gamma/(eps+beta314)^2;
    omega315=gamma/(eps+beta315)^2; omega316=gamma/(eps+beta316)^2;
    omega317=gamma/(eps+beta317)^2; omega318=gamma/(eps+beta318)^2;
    omega319=gamma/(eps+beta319)^2; omega320=gamma/(eps+beta320)^2;
    omega321=gamma/(eps+beta321)^2; omega322=gamma/(eps+beta322)^2;
    omega323=gamma/(eps+beta323)^2; omega324=gamma/(eps+beta324)^2;
    omega325=gamma/(eps+beta325)^2; omega326=gamma/(eps+beta326)^2;
    omega327=gamma/(eps+beta327)^2; omega328=gamma/(eps+beta328)^2;
    omega329=gamma/(eps+beta329)^2; omega330=gamma/(eps+beta330)^2;
    omega331=gamma/(eps+beta331)^2; omega332=gamma/(eps+beta332)^2;
    omega333=gamma/(eps+beta333)^2; omega334=gamma/(eps+beta334)^2;
    omega335=gamma/(eps+beta335)^2; omega336=gamma/(eps+beta336)^2;
    omega337=gamma/(eps+beta337)^2; omega338=gamma/(eps+beta338)^2;
    omega339=gamma/(eps+beta339)^2; omega340=gamma/(eps+beta340)^2;
    omega341=gamma/(eps+beta341)^2; omega342=gamma/(eps+beta342)^2;
    omega343=gamma/(eps+beta343)^2; omega344=gamma/(eps+beta344)^2;
    omega345=gamma/(eps+beta345)^2; omega346=gamma/(eps+beta346)^2;
    omega347=gamma/(eps+beta347)^2; omega348=gamma/(eps+beta348)^2;
    omega349=gamma/(eps+beta349)^2; omega350=gamma/(eps+beta350)^2;
    omega351=gamma/(eps+beta351)^2; omega352=gamma/(eps+beta352)^2;
    omega353=gamma/(eps+beta353)^2; omega354=gamma/(eps+beta354)^2;
    omega355=gamma/(eps+beta355)^2; omega356=gamma/(eps+beta356)^2;
    omega357=gamma/(eps+beta357)^2; omega358=gamma/(eps+beta358)^2;
    omega359=gamma/(eps+beta359)^2; omega360=gamma/(eps+beta360)^2;
    omega361=gamma/(eps+beta361)^2; omega362=gamma/(eps+beta362)^2;
    omega363=gamma/(eps+beta363)^2; omega364=gamma/(eps+beta364)^2;
    omega365=gamma/(eps+beta365)^2; omega366=gamma/(eps+beta366)^2;
    omega367=gamma/(eps+beta367)^2; omega368=gamma/(eps+beta368)^2;
    omega369=gamma/(eps+beta369)^2; omega370=gamma/(eps+beta370)^2;
    omega371=gamma/(eps+beta371)^2; omega372=gamma/(eps+beta372)^2;
    omega373=gamma/(eps+beta373)^2; omega374=gamma/(eps+beta374)^2;
    omega375=gamma/(eps+beta375)^2; omega376=gamma/(eps+beta376)^2;
    omega377=gamma/(eps+beta377)^2; omega378=gamma/(eps+beta378)^2;
    omega379=gamma/(eps+beta379)^2; omega380=gamma/(eps+beta380)^2;
    omega381=gamma/(eps+beta381)^2; omega382=gamma/(eps+beta382)^2;
    omega383=gamma/(eps+beta383)^2; omega384=gamma/(eps+beta384)^2;
    omega385=gamma/(eps+beta385)^2; omega386=gamma/(eps+beta386)^2;
    omega387=gamma/(eps+beta387)^2; omega388=gamma/(eps+beta388)^2;
    omega389=gamma/(eps+beta389)^2; omega390=gamma/(eps+beta390)^2;
    omega391=gamma/(eps+beta391)^2; omega392=gamma/(eps+beta392)^2;
    omega393=gamma/(eps+beta393)^2; omega394=gamma/(eps+beta394)^2;
    omega395=gamma/(eps+beta395)^2; omega396=gamma/(eps+beta396)^2;
    omega397=gamma/(eps+beta397)^2; omega398=gamma/(eps+beta398)^2;
    omega399=gamma/(eps+beta399)^2; omega400=gamma/(eps+beta400)^2;
    omega401=gamma/(eps+beta401)^2; omega402=gamma/(eps+beta402)^2;
    omega403=gamma/(eps+beta403)^2; omega404=gamma/(eps+beta404)^2;
    omega405=gamma/(eps+beta405)^2; omega406=gamma/(eps+beta406)^2;
    omega407=gamma/(eps+beta407)^2; omega408=gamma/(eps+beta408)^2;
    omega409=gamma/(eps+beta409)^2; omega410=gamma/(eps+beta410)^2;
    omega411=gamma/(eps+beta411)^2; omega412=gamma/(eps+beta412)^2;
    omega413=gamma/(eps+beta413)^2; omega414=gamma/(eps+beta414)^2;
    omega415=gamma/(eps+beta415)^2; omega416=gamma/(eps+beta416)^2;
    omega417=gamma/(eps+beta417)^2; omega418=gamma/(eps+beta418)^2;
    omega419=gamma/(eps+beta419)^2; omega420=gamma/(eps+beta420)^2;
    omega421=gamma/(eps+beta421)^2; omega422=gamma/(eps+beta422)^2;
    omega423=gamma/(eps+beta423)^2; omega424=gamma/(eps+beta424)^2;
    omega425=gamma/(eps+beta425)^2; omega426=gamma/(eps+beta426)^2;
    omega427=gamma/(eps+beta427)^2; omega428=gamma/(eps+beta428)^2;
    omega429=gamma/(eps+beta429)^2; omega430=gamma/(eps+beta430)^2;
    omega431=gamma/(eps+beta431)^2; omega432=gamma/(eps+beta432)^2;
    omega433=gamma/(eps+beta433)^2; omega434=gamma/(eps+beta434)^2;
    omega435=gamma/(eps+beta435)^2; omega436=gamma/(eps+beta436)^2;
    omega437=gamma/(eps+beta437)^2; omega438=gamma/(eps+beta438)^2;
    omega439=gamma/(eps+beta439)^2; omega440=gamma/(eps+beta440)^2;
    omega441=gamma/(eps+beta441)^2; omega442=gamma/(eps+beta442)^2;
    omega443=gamma/(eps+beta443)^2; omega444=gamma/(eps+beta444)^2;
    omega445=gamma/(eps+beta445)^2; omega446=gamma/(eps+beta446)^2;
    omega447=gamma/(eps+beta447)^2; omega448=gamma/(eps+beta448)^2;
    omega449=gamma/(eps+beta449)^2; omega450=gamma/(eps+beta450)^2;
    omega451=gamma/(eps+beta451)^2; omega452=gamma/(eps+beta452)^2;
    omega453=gamma/(eps+beta453)^2; omega454=gamma/(eps+beta454)^2;
    omega455=gamma/(eps+beta455)^2; omega456=gamma/(eps+beta456)^2;
    omega457=gamma/(eps+beta457)^2; omega458=gamma/(eps+beta458)^2;
    omega459=gamma/(eps+beta459)^2; omega460=gamma/(eps+beta460)^2;
    omega461=gamma/(eps+beta461)^2; omega462=gamma/(eps+beta462)^2;
    omega463=gamma/(eps+beta463)^2; omega464=gamma/(eps+beta464)^2;
    omega465=gamma/(eps+beta465)^2; omega466=gamma/(eps+beta466)^2;
    omega467=gamma/(eps+beta467)^2; omega468=gamma/(eps+beta468)^2;
    omega469=gamma/(eps+beta469)^2; omega470=gamma/(eps+beta470)^2;
    omega471=gamma/(eps+beta471)^2; omega472=gamma/(eps+beta472)^2;
    omega473=gamma/(eps+beta473)^2; omega474=gamma/(eps+beta474)^2;
    omega475=gamma/(eps+beta475)^2; omega476=gamma/(eps+beta476)^2;
    omega477=gamma/(eps+beta477)^2; omega478=gamma/(eps+beta478)^2;
    omega479=gamma/(eps+beta479)^2; omega480=gamma/(eps+beta480)^2;
    omega481=gamma/(eps+beta481)^2; omega482=gamma/(eps+beta482)^2;
    omega483=gamma/(eps+beta483)^2; omega484=gamma/(eps+beta484)^2;
    omega485=gamma/(eps+beta485)^2; omega48
```



CFD Computational Fluid Dynamics
ENO/WENO

WENO-Z is designed to be high order at points where the derivative of the function vanishes. The weights are modified

$$\tau_s = |\beta_1 - \beta_3|$$

$$\beta_j^z = \frac{\beta_j + \varepsilon}{\beta_j + \tau_s + \varepsilon}, \quad j = 1, 2, 3;$$

$$\tilde{\omega}_j = \frac{\gamma_j}{\beta_j^z} = \gamma_j \left(1 + \frac{\tau_s}{\beta_j + \varepsilon} \right);$$

$$\omega_j = \frac{\tilde{\omega}_j}{\sum_j \tilde{\omega}_j}$$

$$\varepsilon = 10^{-40}$$

$$f_{i+1/2}^- \approx \sum_{j=0}^k \omega_j p_j(x_{i+1/2})$$

Reference: R. Borges, M. Carmona, B. Costa, W. S. Don. An improved weighted essentially non-oscillatory scheme for hyperbolic conservation laws. Journal of Computational Physics 227 (2008) 3191–3211

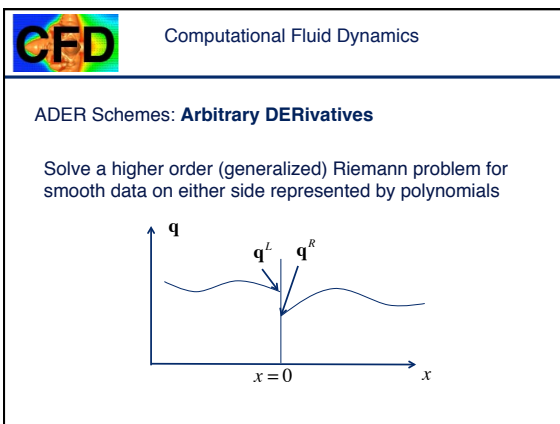
CFD Computational Fluid Dynamics

Other Approaches

CFD Computational Fluid Dynamics

For the advection terms, the methods described for hyperbolic equations, including ENO, can all be applied, yielding stable and robust methods that can be “forgiving” for low resolution.

Several other approaches have also been tried



CFD Computational Fluid Dynamics
CIP-gradient augmentation

The CIP (Constrained Interpolation Polynomial) Method (Yabe)

In addition to advecting the marker function f , its derivative is advected by fitting a third order polynomial through the function and its derivatives.

Start with $\frac{\partial f}{\partial t} + u \frac{\partial f}{\partial x} = 0$

Introduce $g = \partial f / \partial x$.

In 1D, the advection of the derivative is given by $\frac{\partial g}{\partial t} + u \frac{\partial g}{\partial x} = 0$

Therefore, the derivative is translated with velocity u , just as the function. In 2D splitting is used to separate translation and deformation

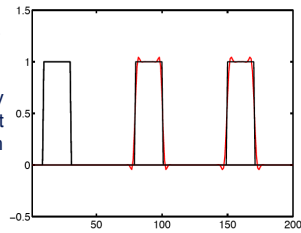
$\xi = u \Delta t$

f and g given

New f and g



The CIP method results in very accurate advection and for a sharp interface it greatly reduces overshoots, but does not eliminate them completely



Although most high-order finite volume methods are based on updating the average value and reconstruct a higher order approximation, advecting more information is gaining some popularity



Compact schemes



Compact Schemes

The standard way to obtain higher order approximations to derivatives is to include more points. This can lead to very wide stencils and near boundaries this requires a large number of “ghost” points outside the boundary. This can be overcome by “compact” schemes, where we derive expressions relating the derivatives at neighboring points to each other and the function values.



By a Taylor series expansion the following forth order relations between the values of f and the derivatives of f can be derived

$$f_{i+1} = f_i + \frac{\partial f_i}{\partial x} \Delta x + \frac{\partial^2 f_i}{\partial x^2} \frac{\Delta x^2}{2} + \frac{\partial^3 f_i}{\partial x^3} \frac{\Delta x^3}{6} + \frac{\partial^4 f_i}{\partial x^4} \frac{\Delta x^4}{24} + O(\Delta x^5) \quad (1)$$

$$f_{i-1} = f_i - \frac{\partial f_i}{\partial x} \Delta x + \frac{\partial^2 f_i}{\partial x^2} \frac{\Delta x^2}{2} - \frac{\partial^3 f_i}{\partial x^3} \frac{\Delta x^3}{6} + \frac{\partial^4 f_i}{\partial x^4} \frac{\Delta x^4}{24} + O(\Delta x^5) \quad (2)$$

Adding

$$f_{i+1} + f_{i-1} = 2f_i + f_i'' \Delta x^2 + f_i'''' \frac{\Delta x^4}{12} + O(\Delta x^6)$$

Taking the second derivative:

$$f_{i+1}'' + f_{i-1}'' = 2f_i'' + f_i'''' \Delta x^2 + f_i'''''' \frac{\Delta x^4}{12} + O(\Delta x^6) \quad (3)$$

Eliminating the fourth derivative

$$f_{i+1}'' + 10f_i'' + f_{i-1}'' = \frac{12}{\Delta x^2} (f_{i+1} - 2f_i + f_{i-1}) + O(\Delta x^4)$$



By a Taylor series expansion the following forth order relations between the values of f and the derivatives of f can be derived

$$f_{i+1} = f_i + \frac{\partial f_i}{\partial x} \Delta x + \frac{\partial^2 f_i}{\partial x^2} \frac{\Delta x^2}{2} + \frac{\partial^3 f_i}{\partial x^3} \frac{\Delta x^3}{6} + \frac{\partial^4 f_i}{\partial x^4} \frac{\Delta x^4}{24} + O(\Delta x^5) \quad (1)$$

$$f_{i-1} = f_i - \frac{\partial f_i}{\partial x} \Delta x + \frac{\partial^2 f_i}{\partial x^2} \frac{\Delta x^2}{2} - \frac{\partial^3 f_i}{\partial x^3} \frac{\Delta x^3}{6} + \frac{\partial^4 f_i}{\partial x^4} \frac{\Delta x^4}{24} + O(\Delta x^5) \quad (2)$$

Adding and taking the first derivative:

$$f_{i+1}' + f_{i-1}' = 2f_i' + f_i''' \Delta x^2 + f_i'''' \frac{\Delta x^4}{12} + O(\Delta x^6) \quad (3)$$

Subtracting 2 from 1

$$f_{i+1} - f_{i-1} = 2f_i' \Delta x + f_i''' \frac{\Delta x^3}{3} + O(\Delta x^5) \quad (4)$$

Eliminating the third derivative

$$f_{i+1}' + 4f_i' + f_{i-1}' = \frac{3}{\Delta x} (f_{i+1} - f_{i-1}) + O(\Delta x^4)$$



To solve the nonlinear advection-diffusion equation

$$\frac{\partial f}{\partial t} = -f \frac{\partial f}{\partial x} + D \frac{\partial^2 f}{\partial x^2}$$

we first find the first and second derivatives using the expressions derived above:

$$\frac{\partial f_{i+1}}{\partial x} + 4 \frac{\partial f_i}{\partial x} + \frac{\partial f_{i-1}}{\partial x} = \frac{3}{\Delta x} (f_{i+1} - f_{i-1}) + O(\Delta x^2)$$

$$\frac{\partial^2 f_{i+1}}{\partial x^2} + 10 \frac{\partial^2 f_i}{\partial x^2} + \frac{\partial^2 f_{i-1}}{\partial x^2} = \frac{12}{\Delta x^2} (f_{i+1} - 2f_i + f_{i-1}) + O(\Delta x^4)$$

And use the values to compute the RHS. The time integration is then done using a high order time integration method.

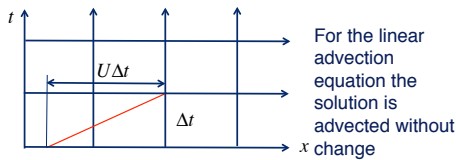


Semi-Lagrangian Schemes



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

$$\frac{df}{dt} = 0 \quad \text{on} \quad \frac{dx}{dt} = U \quad f(t, x) = f(t - \Delta t, x - U\Delta t)$$

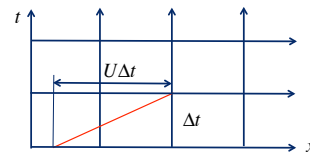


$$f(t, x) = f(t - \Delta t, x - U\Delta t)$$

$$f_j^{n+1} = \text{Intp}(f^n(x_j^o))$$

$$x_j^o = x_j - U\Delta t$$

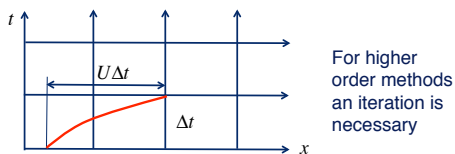
Interpolate the solution at the old time level, at a point given by tracing back the characteristic



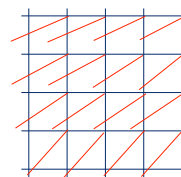
$$\frac{\partial f}{\partial t} + u(x, t) \frac{\partial f}{\partial x} = 0 \quad x_j^o = x_j - \int_t^{t+\Delta t} u(x(t), t) dt$$

$$\frac{df}{dt} = 0 \quad \text{on} \quad \frac{dx}{dt} = u \quad x_j^o = x_j - \frac{\Delta t}{2} (u(x_j, t) + u(x_j^o, t + \Delta t))$$

$$f_j^{n+1} = \text{Intp}(f^n(x_j^o))$$



For two and three-dimensional flows a multidimensional interpolation is necessary



Linear interpolation is usually too diffusive but several higher order ones have been used



Computational Fluid Dynamics

Semi-Lagrangian schemes are widely used in weather forecasting, simulations of plasma, and computer animations, for example



Computational Fluid Dynamics

Enormous progress has been made in solution techniques for hyperbolic systems with shocks in the last twenty years. Advanced methods are now able to resolve complex shocks within a grid space or two, even in multidimensional situations for a large range of governing parameters and physical complexity.

Increasingly we see methods developed for the inviscid Euler equation with shocks being used for the advection part of the Navier-Stokes solvers.