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

Computational Fluid Dynamics

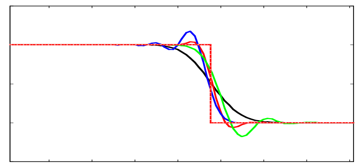
Lecture 9
February 8, 2017

Grétar Tryggvason

Computational Fluid Dynamics
Godunov's Theorem

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

Flux Limiters Review

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:

$$\frac{df_j}{dt} = \frac{-1}{h}\left(F^n_{j+1/2} - F^n_{j-1/2}\right)$$

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

For the Linear Advection equation we have:

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

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

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

$\Psi(r) = r;$ Centered scheme (Beam-Warming)

$\Psi(r) = 1$ Lax-Wendroff

$\Psi(r) = 0$ First order upwind

Computational Fluid Dynamics

Designing Limiters: The Sweby diagram

Computational Fluid Dynamics
Limiters

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

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

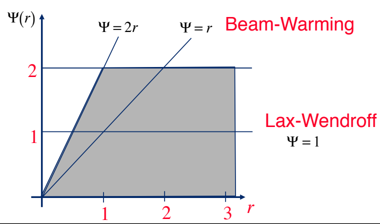
To design schemes that prevent the emergence of unphysical oscillations, we need a precise definition of what we mean by no oscillations. A few such criteria have been proposed but the most widely used on is based on the Total Variation, defined by

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

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.



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

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

$$0 \le \Psi(r) \le r \quad r \ge 0$$

Generally we also require the limiters to be symmetric

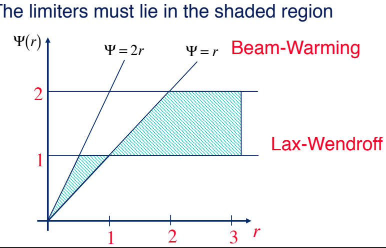
$$\frac{\Psi(r)}{r} = \Psi\left(\frac{1}{r}\right) \qquad r = \frac{f_{j+1} - f_j}{f_j - f_{j-1}}$$

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



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

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 approac. JCP 224 (2007), 182-207

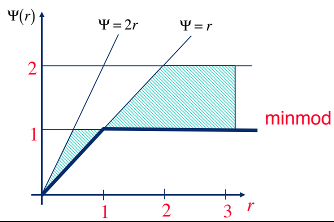
This paper treats only steady state problems.

Computational Fluid Dynamics
Limiters

The Minmod Limiter (Roe and others)

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

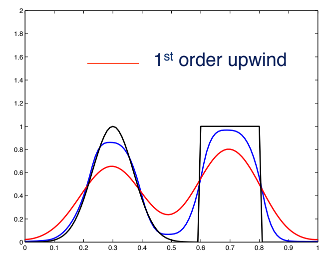
$$\Psi(r) = \max[0, \min(1, r)]$$



Simple and generally works well. Fairly dissipative

Computational Fluid Dynamics
Limiters

minmode



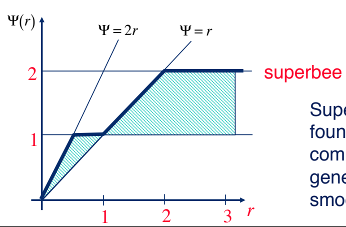
T=1.0; h=0.01; dt=h/3

Computational Fluid Dynamics
Limiters

The Superbee Limiter (Roe)

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

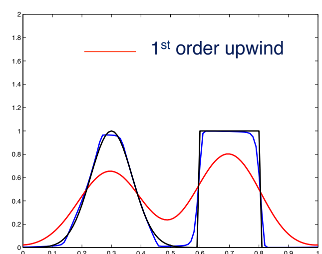
$$\Psi(r) = \max[0, \min(2r, 1), \min(r, 2)]$$



Superbee is generally found to be too compressive and generates steps in smooth regions

Computational Fluid Dynamics
Limiters

Superbee



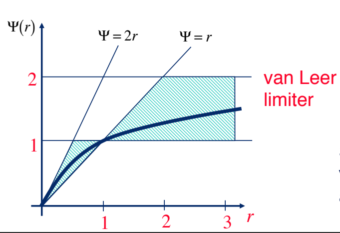
T=1.0; h=0.01; dt=h/3

Computational Fluid Dynamics
Limiters

The original van Leer limiter

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

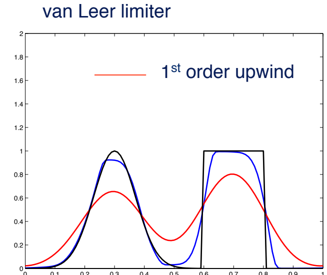
$$\Psi(r) = \frac{r + |r|}{1 + r}$$



Smooth variation—no abrupt switches

Computational Fluid Dynamics
Limiters

van Leer limiter



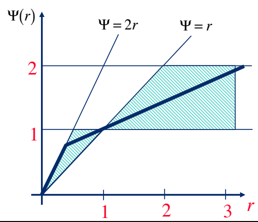
T=1.0; h=0.01; dt=h/3

Computational Fluid Dynamics
Limiters

The MUSCL Limiter
MUSCL: Monotonic Upstream-Centered Scheme for Conservation Laws

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

$$\Psi(r) = \max[0, \min(2r, (r+1)/2, 2)]$$

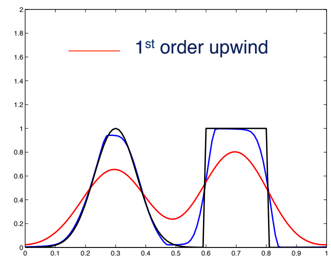


MUSCL

One of the original suggestions by van Leer

Computational Fluid Dynamics
Limiters

The MUSCL Limiter



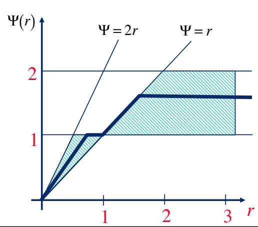
T=1.0; h=0.01; dt=h/3

Computational Fluid Dynamics
Limiters

The Sweby Limiter (Sweby)

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

$$\Psi(r) = \max[0, \min(\beta r, 1), \min(r, \beta)]; \quad 1 \le \beta \le 2$$

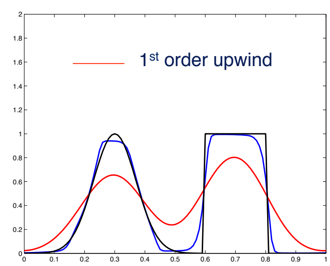


Sweby, beta=1.5

An example of a limiter with abrupt switching that traverses the interior of the allowable region

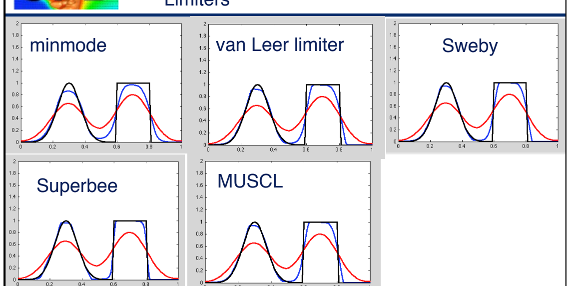
Computational Fluid Dynamics
Limiters

The Sweby Limiter (Sweby)



T=1.0; h=0.01; dt=h/3

Computational Fluid Dynamics
Limiters



Computational Fluid Dynamics
Limiters

```
% one-dimensional linear advection using flux limiters
% This looks like the final and complete version of the program
% Feb. 21, 2011
%-----------------------------------------------------------
n=101; nstep=300; length=1.0;h=length/(n-1);
dt=0.333333333*h; time=0.0;  for i=1:n, x(i)=h*(i-1);end

f=zeros(n,1);fo=zeros(n,1);fh=zeros(n,1);foo=zeros(n,1);
fu=zeros(n,1);fuo=zeros(n,1);
for i=1:n, if (x(i)>=0.0 & x(i)<=0.6), f(i)=exp(-100*(x(i)-0.3)^2
for i=1:n, if (x(i)>=0.6 & x(i)<=0.8), f(i)=1.0; end; end; %ini
fu=f;
```

SEE CODE ON RIGHT

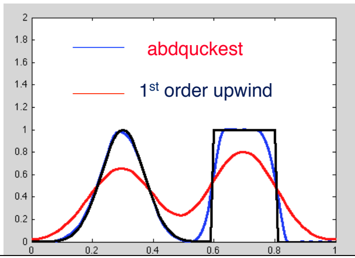
```
for m=1:nstep,m,time
  hold off;plot(x,f,'linewidt',2);  %axis([1 n 2.0, 4.5]); %plot solution
  hold on;plot(x,fu,'r','linewidt',2);axis([0 length 0 2]); pause(0.1)

  foo=f;
  for is=1:2    %two steps per timestep
   fo=f;
   for i=2:n-1,
     bot=(fo(i)-fo(i-1));top=(fo(i+1)-fo(i)); r=top*bot/(bot^2+0.00001);
     % Limiters
     % psi=max([0, min([2*r,0.5*(r+1),2])]);        % monotonized central (MC)
      psi=max([0, min([1.5*r,1]),min([r,1.5]) ]);  % Sweby
     % psi=max([0, min([2*r,1]),min([r,2])]);       % superbee  gtext('superbee')
     % psi=max([0, min([r,1])]);                    % minmod   gtext('MINMOD')
     % psi=(r+abs(r))/(r+1);                        % van Leer
     % psi=0;                                       % upwind
     % psi=r;
     fh(i)=fo(i)+0.5*psi*(fo(i)-fo(i-1));
   end;

   bot=(fo(1)-fo(n-1));top=(fo(2)-fo(1)); r=top*bot/(bot^2+0.00001);
   % psi=max([0, min([2*r,0.5*(r+1),2])]);        % monotonized central (MC)
   psi=max([0, min([1.5*r,1]),min([r,1.5]) ]);  % Sweby
   % psi=max([0, min([2*r,1]),min([r,2])]);       % superbee
   % psi=max([0, min([r,1])]);                    % minmod
   % psi=(r+abs(r))/(r+1);                        %van Leer
   % psi=0;                                       % upwind
   % psi=r;
   fh(1)=fo(1)+0.5*psi*(fo(1)-fo(n-1)); fh(n)=fh(1);

   for i=2:n-1,
     f(i)=fo(i)-(dt/h)*(fh(i)-fh(i-1) );
   end;
   f(1)=fo(1)-(dt/h)*(fh(1)-fh(n-1) );f(n)=f(1);
  end

  for i=1:n,f(i)=0.5*(f(i)+foo(i)); end % average the solution for second oder time step
```

```
%-----------------------
  % first order upwind
  fuo=fu; for i=2:n-1,fu(i)=fuo(i)-(dt/h)*(fuo(i)-fuo(i-1)); end
  fu(1)=fuo(1)-(dt/h)*(fuo(1)-fuo(n-1));fu(n)=fu(1);
  time=time+dt;
end;

f=zeros(n,1);
for i=1:n, if (x(i)>=0.0 & x(i)<=0.6), f(i)=exp(-100*(x(i)-0.3)^2); enc
for i=1:n, if (x(i)>=0.6 & x(i)<=0.8), f(i)=1.0; end; end; %initial cond.
hold on;plot(x,f,'k','linewidt',2);
% print -depsc tempfig
```

Computational Fluid Dynamics
Limiters

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 schemefor the simulation of convection–diffusion problems. Int. J. Numer. Meth. Fluids 2009; 60:1–26



Computational Fluid Dynamics
Limiters

Limiting the variables:

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

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

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

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

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

Computational Fluid Dynamics
Limiters

Limiting the Fluxes

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

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

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

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

Final step
$$f_j^{n+1} = f_j^n - \frac{\Delta t}{h}\Big(\left[F^n_{j+1/2} + \Psi^L_{j+1/2}\left(F^{n+1/2}_{j+1/2} - F^n_{j+1/2}\right)\right] - \left[F^n_{j-1/2} + \Psi^L_{j-1/2}\left(F^{n+1/2}_{j-1/2} - F^n_{j-1/2}\right)\right]\Big)$$

260
Computational Fluid Dynamics

Higher Order and More Recent Methods

261
Computational Fluid Dynamics
ENO/WENO

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

Computational Fluid Dynamics
ENO/WENO

ENO

Essential Non-Oscillatory

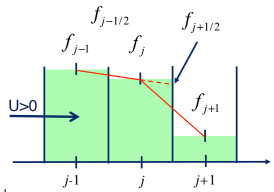
Introduced in: A. Harten, B. Engquist, S. Osher, S.R. Chakravarty, Some results on high-order accurate essentially non-oscillatory schemes, Appl. Numer. Math. 2, 347–377 (1986).

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).

Computational Fluid Dynamics
ENO/WENO

Example: Second order ENO

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



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

$$f_{j+1/2} = f_j + \tfrac{1}{2}\,\mathrm{amin}\left(\Delta f_j^+, \Delta f_j^-\right) \qquad \mathrm{amin}(a,b) = \begin{cases} a, & |a| < |b| \\ b, & |b| \le |a| \end{cases}$$

Computational Fluid Dynamics
ENO/WENO

Second order ENO scheme for the linear advection equation

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} = 0 \qquad \mathrm{amin}(a,b) = \begin{cases} a, & |a| < |b| \\ b, & |b| \le |a| \end{cases}$$

$$f_j^* = f_j^n - \frac{\Delta t}{h} u_j^n \left(f_{j+1/2}^n - f_{j-1/2}^n\right)$$

$$f_j^{n+1} = f_j^n - \frac{\Delta t}{h}\frac{1}{2}\left(u_j^n\left(f_{j+1/2}^n - f_{j-1/2}^n\right) + u_j^*\left(f_{j+1/2}^* - f_{j-1/2}^*\right)\right)$$

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

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

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

```
% one-dimensional linear advection by ENO
%------------------------------------------------------------
n=41; nstep=60; length=2.0;h=length/(n-1);
dt=0.3*h;
time=0.0;

f=zeros(n,1);fo=zeros(n,1);fh=zeros(n,1);
dfp=zeros(n,1);dfm=zeros(n,1);df=zeros(n,1);
fhs=zeros(n,1);fs=zeros(n,1);

for i=1:n, f(i)=1+0.5*sin(2*pi*h*(i-1)); end; %initial conditions
%f(5:15)=1;

fu=f;
for m=1:nstep,m,time

  hold off;plot(f,'linewidt',2);  %axis([1 n 2.0, 4.5]); %plot solution
  hold on;plot(fu,'r','linewidt',2);
set(gcf,'Position',[343 385 500 350]);axis([0 n 0 2]);
pause(0.1)

%-----------------------
fo=f;fh=f;fs=f;
  for i=2:n-1,
%     dfp(i)=fo(i+1)-fo(i);  dfm(i)=fo(i)-fo(i-1);
%     if (abs(dfp(i)) < abs(dfm(i))), df(i)=dfp(i);else df(i)=dfm(i); end
     df(i)=minabs(fo(i+1)-fo(i),fo(i)-fo(i-1));
  end;
     dfp(1)=fo(2)-fo(1);  dfm(i)=fo(1)-fo(n-1);
     if (abs(dfp(1)) < abs(dfm(1))), df(1)=dfp(1);else df(1)=dfm(1); end
    df(n)=df(1);

  for i=1:n,
     fh(i)=fo(i)+0.5*df(i);
  end;
   for i=2:n,
     fs(i)=fo(i)-(dt/h)*(fh(i)-fh(i-1));
  end;
     fs(1)=fs(n);

%Correction step
  for i=2:n-1,
     dfp(i)=fs(i+1)-fs(i);  dfm(i)=fs(i)-fs(i-1);
     if (abs(dfp(i)) < abs(dfm(i))), df(i)=dfp(i);else df(i)=dfm(i); end
  end;
       dfp(1)=fo(2)-fo(1);  dfm(i)=fo(1)-fo(n-1);
     if (abs(dfp(1)) < abs(dfm(1))), df(1)=dfp(1);else df(1)=dfm(1); end
    df(n)=df(1);

  for i=1:n,
     fhs(i)=fs(i)+0.5*df(i);
  end;
   for i=2:n-1,
     f(i)=fo(i)-0.5*(dt/h)*((fh(i)-fh(i-1))+(fhs(i)-fhs(i-1)) );
  end;
  f(1)=fo(1)-0.5*(dt/h)*((fh(1)-fh(n-1))+(fhs(1)-fhs(n-1)) );f(n)=f(1);
%-----------------------

fuo=fu; for i=2:n-1,fu(i)=fuo(i)-(dt/h)*(fuo(i)-fuo(i-1)); end
fu(1)=fuo(1)-(dt/h)*(fuo(1)-fuo(n-1));fu(n)=fu(1);
  time=time+dt;
end;
```

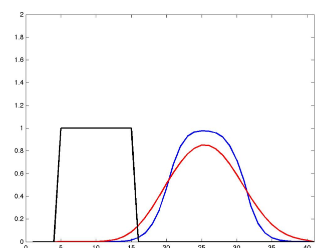
Computational Fluid Dynamics
ENO/WENO

Second order ENO scheme for the linear advection equation

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

Blue: 2nd order ENO

Red: 1st Upwind

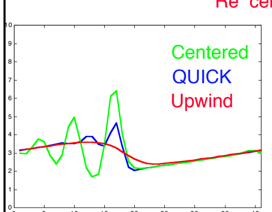


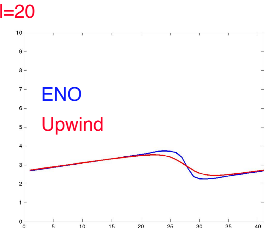
Computational Fluid Dynamics

Second order ENO

$$\frac{\partial f}{\partial t} + \frac{1}{2}\frac{\partial f^2}{\partial x} = D\frac{\partial^2 f}{\partial x^2}$$

Re cell=20





Computational Fluid Dynamics
ENO/WENO

WENO

Weighted Essential Non-Oscillatory

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

Computational Fluid Dynamics

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

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

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

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

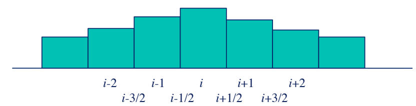
Computational Fluid Dynamics
ENO/WENO

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:

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

The lower bound is arbitrary and can be replaced

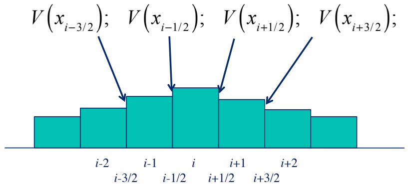


Computational Fluid Dynamics
ENO/WENO

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

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

$$V\left(x_{i-3/2}\right); \quad V\left(x_{i-1/2}\right); \quad V\left(x_{i+1/2}\right); \quad V\left(x_{i+3/2}\right);$$



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

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

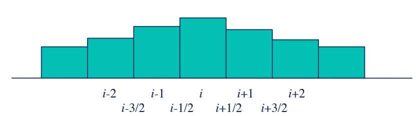
That is, the integral of $p(x)$ over the cell is equal to the cell average f_i

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



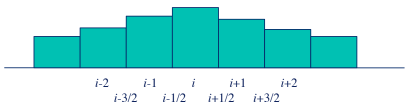
The interpolation polynomial is often taken to be the Lagrangian Polynomial

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

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

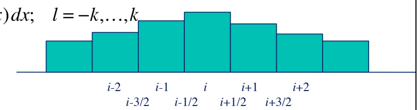
$$\frac{\partial f}{\partial t} + \frac{\partial F}{\partial x} = 0 \qquad \bar{f}_i(t) = \frac{1}{\Delta x}\int_{V_i} f(x,t)\,dx$$

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

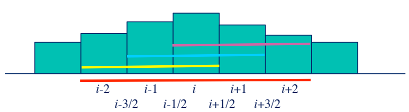
Find the cell average by integrating polynomial representation of the function, first over a subset of the points and then over all the points

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

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



Computational Fluid Dynamics
ENO/WENO



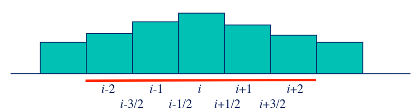
For a polynomial of order three, over the three intervals indicated we get

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

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

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

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

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

Computational Fluid Dynamics
ENO/WENO

Introduce a smoothness measure

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

For our case this gives:

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

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

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

Computational Fluid Dynamics
ENO/WENO

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

First find:

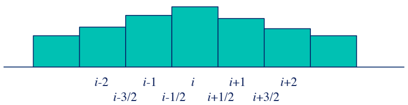
$$\tilde{\omega}_j = \frac{\gamma_j}{(\varepsilon + \beta_j)^2}; \quad \omega_j = \frac{\tilde{\omega}_j}{\sum_j \tilde{\omega}_j}$$

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 studies, 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 semi-discrete equation is

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

The fluxes are the weighted sum

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

where

$$F^{(1)}_{j+1/2}=-\frac{1}{2}F_{j-1}+\frac{3}{2}F_j$$ 2nd order upwind

$$F^{(2)}_{j+1/2}=\frac{1}{2}F_j+\frac{1}{2}F_{j+1}$$ 2nd order central

The weights are

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

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

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

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

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

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

Computational Fluid Dynamics

Why are the weights selected the way the 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 $\omega_1=\gamma_1;\quad \omega_2=\gamma_2$
2. Require $\omega_m>0;\quad \sum\omega_m=1$
3. For a shock we want to get one of the lower order fluxes, so that $\omega_1=1;\quad \omega_2=0$ or $\omega_1=0;\quad \omega_2=1$

$\omega_m\approx\gamma_m \quad if \quad f(x)$ is smooth everywhere

$\omega_m\approx 0 \quad if \quad f(x)$ 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

$\beta_1=\left(f_j-f_{j-1}\right)^2$, $\beta_2=\left(f_{j+1}-f_j\right)^2$ Smoothness indicators (0 if fluxes are constant)

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

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

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

Fluxes

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

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

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

Smoothness Indicators

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

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

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

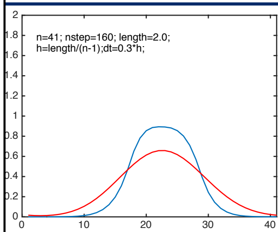
$$\tilde{\omega}_l=\frac{\gamma_l}{\left(\varepsilon+\beta_l\right)^2}$$

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

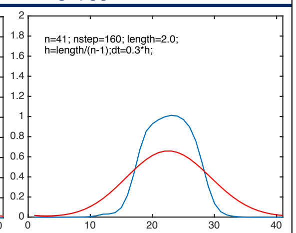
Computational Fluid Dynamics
http://www.nd.edu/~gtryggva/CFD-Course/

```
% one-dimensional linear advection by WENO
% third order scheme
%-----------------------------------------------------------
n=41; nstep=160; length=2.0;h=length/(n-1);
dt=0.3*h;
time=0.0;

gam1=1/3;gam2=2/3; eps=0.000001;

f=zeros(n+1,1);fo=zeros(n+1,1);
f1=zeros(n+1,1);f2=zeros(n+1,1);
flux=zeros(n,1);
fu=zeros(n+1,1);fuo=zeros(n+1,1);

%for i=1:n+1, f(i)=1+0.5*sin(2*pi*h*(i-1)); end; %initial conditions
for i=10:20, f(i)=1; end

fu=f;
for m=1:nstep,m,time

  hold off;plot(f,'linewidt',2);  %axis([1 n 2.0, 4.5]); %plot solution
  hold on;plot(fu,'r','linewidt',2);
set(gcf,'Position',[343 385 500 350]);axis([0 n 0 2]);
pause(0.1)

%-----------------------
fo=f;
  for j=2:n,
    beta1=(fo(j)-fo(j-1))^2;  beta2=(fo(j+1)-fo(j))^2;
    omt1=(gam1)/(eps+beta1)^2; omt2=(gam2)/(eps+beta2)^2;
    omtsum=omt1+omt2; om1=omt1/omtsum; om2=omt2/omtsum;
    flux(j)=om1*(-0.5*fo(j-1)+1.5*fo(j))+om2*(0.5*fo(j)+0.5*fo(j+1));
  end
  flux(1)=flux(n);

  for j=2:n,
     f(j)=fo(j)-(dt/h)*(flux(j)-flux(j-1));
  end;
  f(1)=f(n);f(n+1)=f(2);

% Second Step
f1=f;
  for j=2:n,
    beta1=(f1(j)-f1(j-1))^2;  beta2=(f1(j+1)-f1(j))^2;
    omt1=(gam1)/(eps+beta1)^2; omt2=(gam2)/(eps+beta2)^2;
    omtsum=omt1+omt2; om1=omt1/omtsum; om2=omt2/omtsum;
    flux(j)=om1*(-0.5*f1(j-1)+1.5*f1(j))+om2*(0.5*f1(j)+0.5*f1(j+1));
  end
  flux(1)=flux(n);

   for j=2:n,
     f(j)=0.75*fo(j)+0.25*f1(j)-0.25*(dt/h)*(flux(j)-flux(j-1));
  end;
  f(1)=f(n);f(n+1)=f(2);

% Third Step
f2=f;
  for j=2:n,
    beta1=(f2(j)-f2(j-1))^2;  beta2=(f2(j+1)-f2(j))^2;
    omt1=(gam1)/(eps+beta1)^2; omt2=(gam2)/(eps+beta2)^2;
    omtsum=omt1+omt2; om1=omt1/omtsum; om2=omt2/omtsum;
    flux(j)=om1*(-0.5*f2(j-1)+1.5*f2(j))+om2*(0.5*f2(j)+0.5*f2(j+1));
  end
  flux(1)=flux(n);

   for j=2:n,
     f(j)=(1/3)*fo(j)+(2/3)*f2(j)-(2/3)*(dt/h)*(flux(j)-flux(j-1));
  end;
  f(1)=f(n);f(n+1)=f(2);

%----------------------
% upwind
  fuo=fu; for i=2:n-1,fu(i)=fuo(i)-(dt/h)*(fuo(i)-fuo(i-1)); end
  fu(1)=fuo(1)-(dt/h)*(fuo(1)-fuo(n-1));fu(n)=fu(1);
  time=time+dt;
end;
```

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



WENO3



WENO5

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

$$\tau_5 = |\beta_1 - \beta_3|$$

$$\beta_j^z = \frac{\beta_j + \varepsilon}{\beta_j + \tau_5 + \varepsilon}, \quad j = 1,2,3;$$

$$\tilde{\omega}_j = \frac{\gamma_j}{\beta_j^z} = \gamma_j\left(1 + \frac{\tau_5}{\beta_j + \varepsilon}\right);$$

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

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

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

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

Computational Fluid Dynamics

Other Approaches

Computational Fluid Dynamics

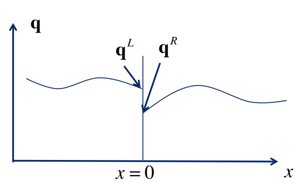
For the advection terms, the methods described for hyperbolic equations, including ENO, can all be applied, yielding stable and robust methods that can be "forgiving" for low resolution.

Several other approaches have also been tried

Computational Fluid Dynamics

ADER Schemes: **Arbitrary DERivatives**

Solve a higher order (generalized) Riemann problem for smooth data on either side represented by polynomials



Computational Fluid Dynamics
CIP-gradient augmentation

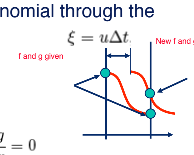
The CIP (Constrained Interpolation Polynomial) Method (Yabe)

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

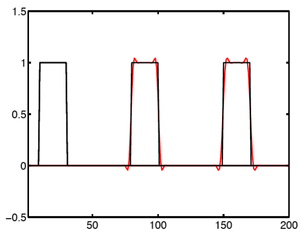
In 1D, the advection of the derivative is given by $\frac{\partial g}{\partial t} + u\frac{\partial g}{\partial x} = 0$



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

Computational Fluid Dynamics
CIP-gradient augmentation

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



Computational Fluid Dynamics

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

Computational Fluid Dynamics

Compact schemes

Computational Fluid Dynamics
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.

Computational Fluid Dynamics
Compact Schemes

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

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

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

Adding

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

Taking the second derivative: (3)

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

Eliminating the fourth derivative

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

Computational Fluid Dynamics
Compact Schemes

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

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

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

Adding and taking the first derivative:

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

Subtracting 2 from 1 (3)

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

Eliminating the third derivative

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

Computational Fluid Dynamics
Compact Schemes

To solve the nonlinear advection-diffusion equation

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

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

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

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

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

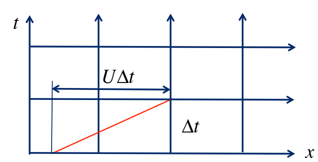
Computational Fluid Dynamics

Semi-Lagrangian Schemes

Computational Fluid Dynamics

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

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



For the linear advection equation the solution is advected without change

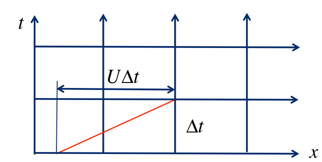
Computational Fluid Dynamics

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

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

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

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



Computational Fluid Dynamics

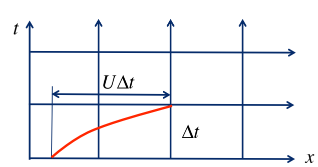
$$\frac{\partial f}{\partial t} + u(x,t)\frac{\partial f}{\partial x} = 0$$

$$\frac{df}{dt} = 0 \quad \text{on} \quad \frac{dx}{dt} = u$$

$$x_j^o = x_j - \int_t^{t+\Delta t} u(x(t),t)\,dt$$

$$x_j^o = x_j - \frac{\Delta t}{2}\left(u(x_j,t) + u(x_j^o, t + \Delta t)\right)$$

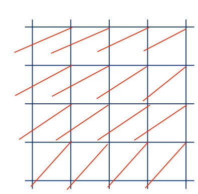
$$f_j^{n+1} = \text{Intp}(f^n(x_j^o))$$



For higher order methods an iteration is necessary

Computational Fluid Dynamics

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

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.