Instructional workshop on OpenFOAM programming
LECTURE # 7

Pavanakumar Mohanamuraly

May 10, 2014

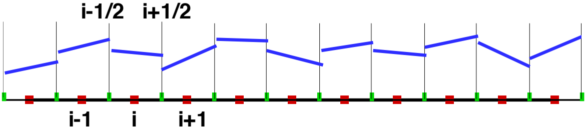
Outline

gaussLaplacianScheme - walk through

Introduction to flux limiters

Limiters in OpenFOAM

Conservation Laws [1]



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

- Q is the cell average value
- $F_{i+\frac{1}{2}}^n$ is approximation to average flux along $x = x_{i+\frac{1}{2}}$

$$F_{i+\frac{1}{2}}^n = \mathcal{F}\left(Q_i^n, Q_{i+1}^n\right) \tag{2}$$

[1] Reference: Finite-volume methods for hyperbolic problems, Randall J. Leveque, Cambridge press

Method of Godunov

- Reconstruct a piecewise polynomial function $\tilde{q}^n(x, t_n)$ defined for all x, from the cell averages Q_i^n.
- In the simplest case this is a piecewise constant function that takes the value Q_i^n in the i^{th} grid cell, i.e.,

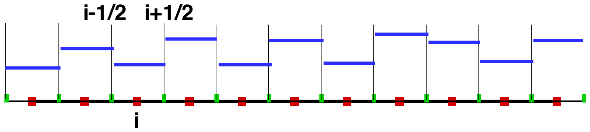
$$\tilde{q}^n(x, t_n) = Q_i^n \quad \textit{for all } x \in \mathcal{C}_i \tag{3}$$

- Evolve the hyperbolic equation exactly (or approximately) with this initial data to obtain $\tilde{q}^n(x, t_{n+1})$ a time Δt later.
- Average this function over each grid cell to obtain new cell averages

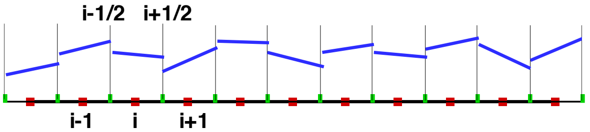
$$Q_i^{n+1} = \frac{1}{\Delta x} \int_{\mathcal{C}_i} \tilde{q}^n(x, t_{n+1}) dx \tag{4}$$

High resolution finite volume schemes

- Assuming constant cell variation leads to $O(\Delta x)$ error



- Linear variation of values in cell leads to $O(\Delta x^2)$ error



- Reconstruct slopes from cell centroid values
- Extrapolate to faces to get L/R states

Piecewise linear reconstruction

- To achieve better than first-order accuracy need better than piecewise constant function
- Construct a piecewise linear function using Q_i^n

$$\tilde{q}^n(x, t_{n+1}) = Q_i^n + \sigma_i^n(x - x_i) \quad \text{for } x_{i-\frac{1}{2}} \leq x \leq x_{i+\frac{1}{2}} \tag{5}$$

and

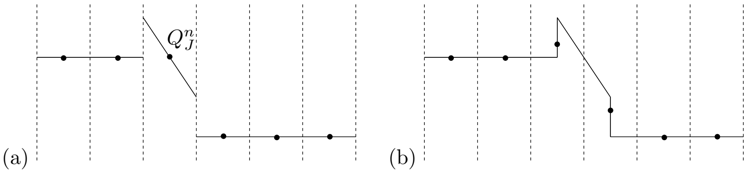
$$x_i = x_{i-\frac{1}{2}} + \frac{1}{2}\Delta x \tag{6}$$

- σ_i^n is the function slope of the i^{th} cell

Problems with linear reconstruction

- Consider a distribution of Q_i^n as shown below (J is some arbitrary cell)

$$Q_i^n = \begin{cases} 1 & if \;\; i \leq J \\ 0 & if \;\; i > J \end{cases} \quad (7)$$



(a) Linear reconstruction for $f(x)$ using cell averages Q_i^n
(b) Simple linear advection of reconstructed values to t_{n+1}

Under/Overshoot in solution

The remedy

- Introduce a function $\phi(\theta_i^n)$ to limit the slope of the function as shown below,

$$\tilde{q}^n(x, t_{n+1}) = Q_i^n + \underbrace{\phi(\theta_i^n)}_{limiter} \underbrace{\sigma_i^n}_{slope} (x - x_i) \quad for\ x_{i-\frac{1}{2}} \leq x \leq x_{i+\frac{1}{2}} \tag{8}$$

- θ_i^n is a measure of the variation of the function Q_i^n in cell i
- $\phi = 0$ is piecewise constant reconstruction
- $\phi = 1$ is piecewise linear reconstruction
- $0 < \phi < 1$ is piecewise linear reconstruction (with loss of accuracy)

Variation measure θ_i^n

- Many ways to obtain θ
- Implementation dependent function
- An example upwind version

$$\theta_i^n = \frac{\Delta Q^n_{I-\frac{1}{2}}}{\Delta Q^n_{i-\frac{1}{2}}} \tag{9}$$

where,

$$\Delta Q^n_{i-\frac{1}{2}} = Q_i^n - Q_{i-1}^n \tag{10}$$

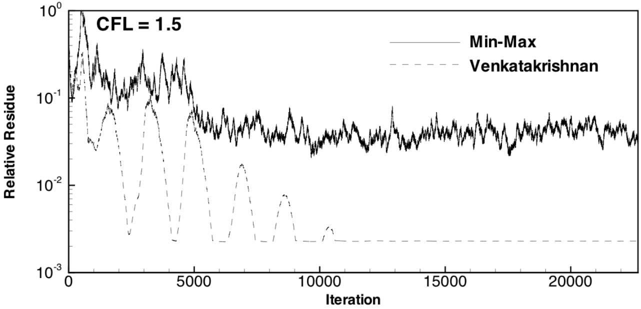
and

$$I = \begin{cases} i-1 \;\; if \;\; a > 0 \\ i+1 \;\; if \;\; a < 0 \end{cases} \tag{11}$$

where, a is the wave speed (advection)

Limiter function ϕ [2]

- Many choices available and no-universal choice
- Can be differentiable or non-differentiable
- Differentiable limiter = smoother convergence



- Min-Max is non-differentiable
- Venkatakishnan is differentiable

[2] Pavanakumar et al, NCAE 2012

Higher than 2^{nd} order

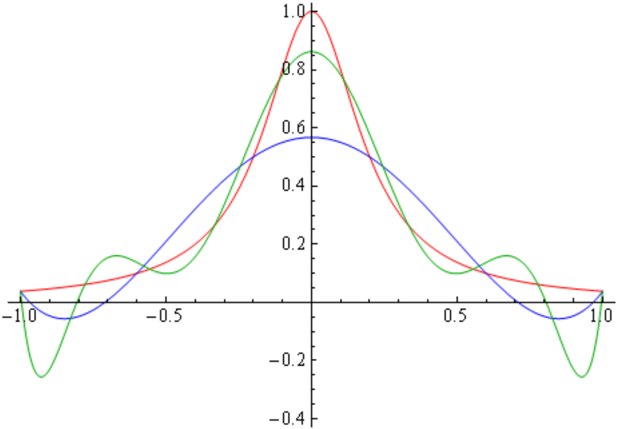
- Is it possible to go higher than second order ?
- Can we use higher-order (> 1) polynomials to reconstruct ?
- There are two main problems discussed in the next two slides

Curse of polynomial interpolation

Fit a polynomial over functions

- Runge phenomena

$$f(x) = \frac{1}{1 + x^2} \text{ for } -1 \leq x \leq 1 \qquad (12)$$

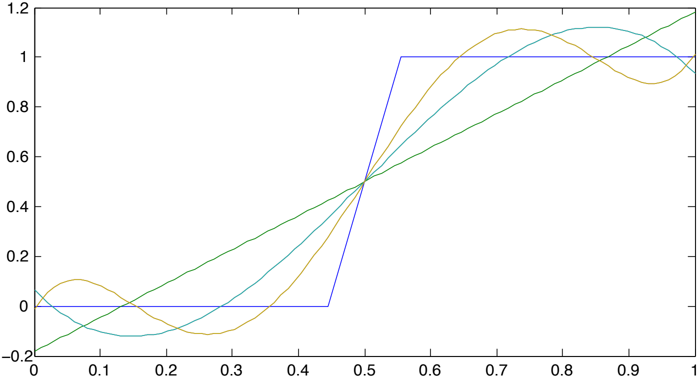


Curse of polynomial interpolation

Fit a polynomial over functions

- Gibbs oscillation

$$f(x) = \begin{cases} 0 & \text{if } x \leq 0.5 \\ 1 & \text{if } x > 0.5 \end{cases} \quad \text{for } 0 \leq x \leq 1 \tag{13}$$



Higher order polynomial

At steep gradients

- Suffer from under-shoot and over-shoot
- Violation of bounds
- Monotonic solution can become non-monotonic

Remedy

- Use orthogonal polynomial like Chebyshev, Legendre, etc
- ENO or WENO type limited polynomials

Beyond the scope of OpenFOAM

OpenFOAM: Limited gradient schemes

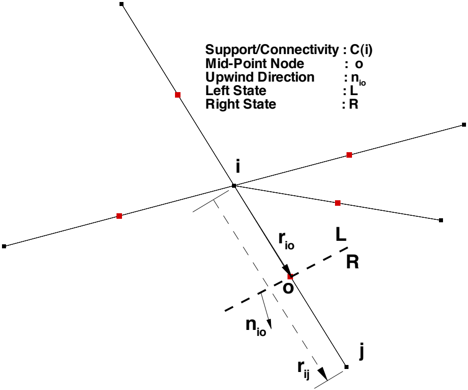
- cellLimited
- cellMDLimited
- faceLimited
- faceMDLimited

Source files located in

```
$FOAM_SRC
|__ /finiteVolume
   |__ /finiteVolume
      |__ /gradSchemes
         |__ /limitedGradSchemes
            |__ cellLimitedGrad
            |__ cellMDLimitedGrad
            |__ faceLimitedGrad
            |__ faceMDLimitedGrad
```

cellLimited gradient scheme - Some notations

- Let the face neighbouring cells of cell i be $j \in C_{ij}$
- Let the faces of cell i be $o \in C_{io}$
- r_{io} is the line joining cell i's centroid to its face centroids
- r_{ij} is the line joining cell i's centroid to its face neighbouring cell j's centroids



cellLimited gradient scheme

- Find maximum and minimum values of all cell (face) neighbours (C_{ij})

$$q_i^{max} = \max_{j \in C_{ij}} q_{ij} \tag{14}$$

$$q_i^{min} = \min_{j \in C_{ij}} q_{ij} \tag{15}$$

- ij is the cell i's j^{th} face-cell neighbour
- io is the cell i's o^{th} face

cellLimited gradient scheme

- Subtract the max/min values from the actual cell values

$$\Delta q_i^{max} = q_i - q_i^{max} \;\; for \;\; i \in \mathcal{C}_i \tag{16}$$
$$\Delta q_i^{min} = q_i - q_i^{min} \;\; for \;\; i \in \mathcal{C}_i \tag{17}$$

- An input parameter κ is used to adjust the Δq as follows,

$$\Delta \tilde{q}_i^{max} = \Delta q_i^{max} + \left(\frac{1-\kappa}{\kappa}\right)\left(\Delta q_i^{max} - \Delta q_i^{min}\right) \tag{18}$$
$$\Delta \tilde{q}_i^{min} = \Delta q_i^{min} - \left(\frac{1-\kappa}{\kappa}\right)\left(\Delta q_i^{max} - \Delta q_i^{min}\right) \tag{19}$$
$$\tag{20}$$

- $\kappa = 0$ is no-limiting
- $\kappa = 1$ is full limiting

cellLimited gradient scheme

- Calculate gradient ∇q_i using a suitable *gradScheme*
- Set the cell limiter values ϕ_i as follows, (loop over all faces of cell)

$$\phi_i = \begin{cases} \min\left[1, \min\limits_{o \in C_{io}} \left(\frac{\Delta \tilde{q}_i^{max}}{\nabla q_i \cdot r_{io}}\right)\right] & \text{if } \Delta \tilde{q}_i^{max} < \nabla q_i \cdot r_{io} \\ \min\left[1, \min\limits_{o \in C_{io}} \left(\frac{\Delta \tilde{q}_i^{min}}{\nabla q_i \cdot r_{io}}\right)\right] & \text{if } \Delta \tilde{q}_i^{min} > \nabla q_i \cdot r_{io} \end{cases} \tag{21}$$

- Now the limited gradient value is $\phi_i \nabla q_i$

This is the min/max limiter and remember the same ϕ_i value is used for all components of ∇q_i

cellMDLimited gradient scheme

- Exactly similar to the *cellLimited* version with the exception that we now calculate the limiter value for each row of a gradient tensor

$$\nabla q_i[n] = \nabla q_i[n] + \sum_{o \in C_{io}} \frac{\mathbf{r}_{io}}{|\mathbf{r}_{io}|} \begin{bmatrix} \Delta \tilde{q}_i^{max} - \nabla q_i[n] \cdot r_{io} & if \;\; \Delta \tilde{q}_i^{max} < \nabla q_i[n] \cdot r_{io} \\ \Delta \tilde{q}_i^{min} - \nabla q_i[n] \cdot r_{io} & if \;\; \Delta \tilde{q}_i^{min} > \nabla q_i[n] \cdot r_{io} \end{bmatrix} \quad for \;\; rows \;\; n = 1, 2, 3 \tag{22}$$

- Now the limited gradient value is calculated in-place in ∇q_i

faceLimited gradient scheme

- Max/min values calculated using local face owner/neighbour cell values

$$q_{io}^{max} = \max(q_{io}^{own}, q_{io}^{nei}) \tag{23}$$

$$q_{io}^{min} = \min(q_{io}^{own}, q_{io}^{nei}) \tag{24}$$

- Correct max/min face values using input parameter κ

$$\tilde{q}_{io}^{max} = q_{io}^{max} + \left(\frac{1-\kappa}{\kappa}\right)(q_{io}^{max} - q_{io}^{min}) \tag{25}$$

$$\tilde{q}_{io}^{min} = q_{io}^{min} - \left(\frac{1-\kappa}{\kappa}\right)(q_{io}^{max} - q_{io}^{min}) \tag{26}$$

faceLimited gradient scheme

- Let ∇q_i be the gradient obtained from *gradScheme*
- Limiter function ϕ_i is calculated as follows,

$$\phi_i = \begin{cases} \min\left[1, \min\limits_{o \in C_{io}} \left(\frac{\Delta q_{io}^{max}}{\nabla q_i \cdot r_{io}}\right)\right] & \text{if } \Delta q_{io}^{max} < \nabla q_i \cdot r_{io} \\ \min\left[1, \min\limits_{o \in C_{io}} \left(\frac{\Delta q_{io}^{min}}{\nabla q_i \cdot r_{io}}\right)\right] & \text{if } \Delta q_{io}^{min} > \nabla q_i \cdot r_{io} \end{cases} \tag{27}$$

where, $\Delta q_{io}^{max/min} = q_i - \tilde{q}_{io}^{max/min}$

- Now the limited gradient value is $\phi_i \nabla q_i$

More dissipative than cellLimited version but cheaper computationally (2X faster)

faceMDLimited gradient scheme

- Multidimensional version limiting is for gradient tensors

$$\nabla q_i[n] = \nabla q_i[n] + \sum_{o \in C_{io}} \frac{\mathbf{r}_{io}}{|\mathbf{r}_{io}|} \left[\begin{array}{ll} \Delta q_{io}^{max} - \nabla q_i[n] \cdot r_{io} & if \;\; \Delta q_{io}^{max} < \nabla q_i[n] \cdot r_{io} \\ \Delta q_{io}^{min} - \nabla q_i[n] \cdot r_{io} & if \;\; \Delta q_{io}^{min} > \nabla q_i[n] \cdot r_{io} \end{array} \right] \quad for \;\; rows \;\; n = 1, 2, 3 \tag{28}$$

- Limited gradient value calculated in-place ∇q_i

End of Week 3 Day 2