Discontinuous Higher Order Discretization Methods

Willem Deconinck

ECMWF

ECMWF Training Course in Advanced Numerical Methods

March 2017

What are Higher Order Methods?

Definition

Higher order methods have truncation errors exceeding 2

What are Higher Order Methods?

Definition

Higher order methods have truncation errors exceeding 2

- Fourth order finite-difference:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x} + \frac{\Delta x^4}{30}\left.\frac{\partial^5 u}{\partial x^5}\right|_{x_i} + \ldots$$

What are Higher Order Methods?

Definition

Higher order methods have truncation errors exceeding 2

- Fourth order finite-difference:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x} + \frac{\Delta x^4}{30} \left.\frac{\partial^5 u}{\partial x^5}\right|_{x_i} + \ldots$$

- The spectral method is of an "infinite order"

$$||u - u_{\text{exact}}|| = O(N^{-p}), \quad N \to \infty, \quad \text{non-continuous in } \frac{\partial^p u}{\partial x^p}$$

What are Higher Order Methods?

Definition

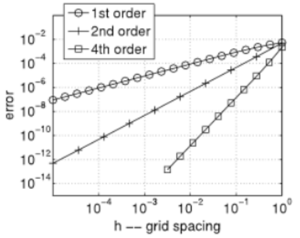
Higher order methods have truncation errors exceeding 2

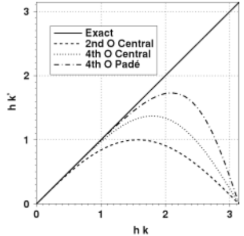
- Fourth order finite-difference:

$$\left.\frac{\partial u}{\partial x}\right|_{x_i} = \frac{-u_{i+2} + 8u_{i+1} - 8u_{i-1} + u_{i-2}}{12\Delta x} + \frac{\Delta x^4}{30}\left.\frac{\partial^5 u}{\partial x^5}\right|_{x_i} + \ldots$$

- The spectral method is of an "infinite order"

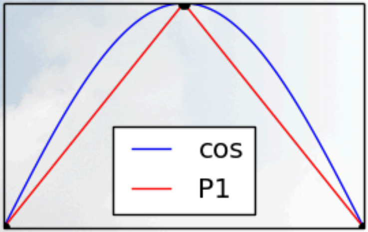
$$||u - u_{\text{exact}}|| = O(N^{-p}), \quad N \to \infty, \quad \text{non-continuous in } \frac{\partial^p u}{\partial x^p}$$

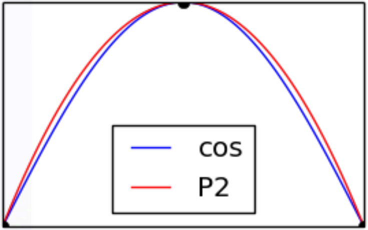




Why Higher Order Methods?

Compare cos function, approximated by 3 points





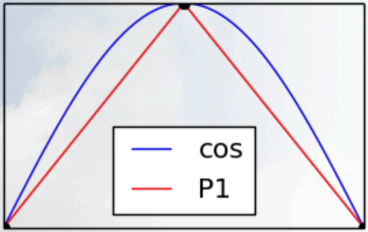
Can we not just add more points?

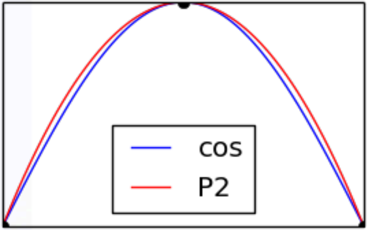
Higher-order methods when:

- High accuracy is required (increasingly so)

Why Higher Order Methods?

Compare cos function, approximated by 3 points





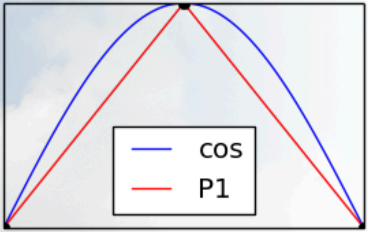
Can we not just add more points?

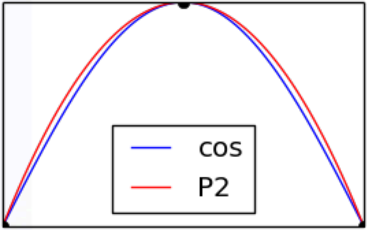
Higher-order methods when:

- High accuracy is required (increasingly so)
- Long time integration is required

Why Higher Order Methods?

Compare cos function, approximated by 3 points





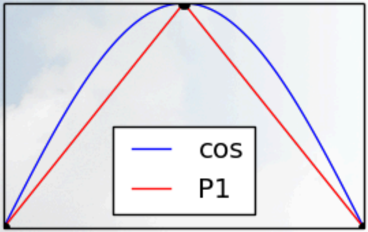
Can we not just add more points?

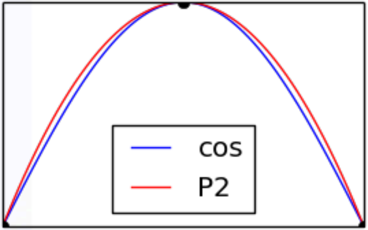
Higher-order methods when:

- High accuracy is required (increasingly so)
- Long time integration is required
- Memory becomes a bottleneck

Why Higher Order Methods?

Compare cos function, approximated by 3 points





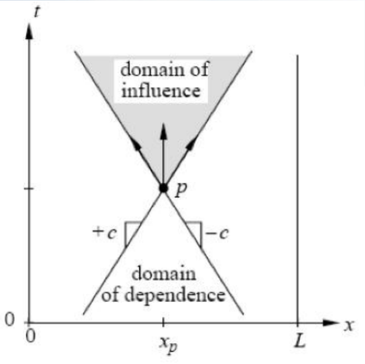
Can we not just add more points?

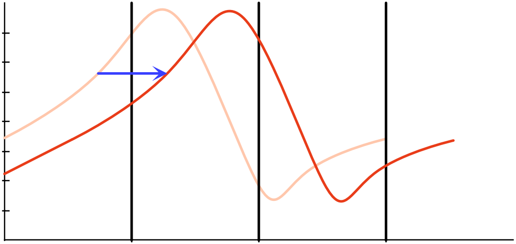
Higher-order methods when:

- High accuracy is required (increasingly so)
- Long time integration is required
- Memory becomes a bottleneck
- Scalability on parallel computers is important

Hyperbolic Conservation Laws

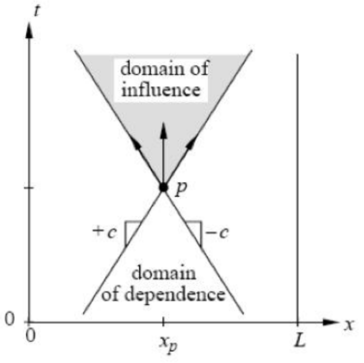
$$\frac{\partial u}{\partial t} + \frac{\partial f}{\partial x} - g = 0$$

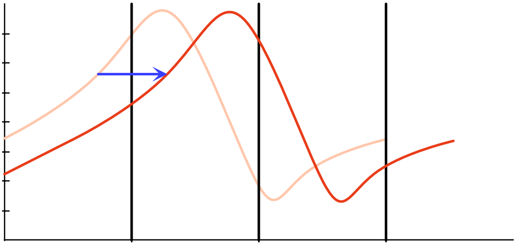




Hyperbolic Conservation Laws

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





Conservation:

Flux going out of one cell = Flux entering the next

Higher-Order Finite Difference

Fourth order Finite difference:

$$\frac{\partial u_i}{\partial t} = -\left(\frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12\Delta x}\right) + g_i$$

Higher-Order Finite Difference

Fourth order Finite difference:

$$\frac{\partial u_i}{\partial t} = -\left(\frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12\Delta x}\right) + g_i$$

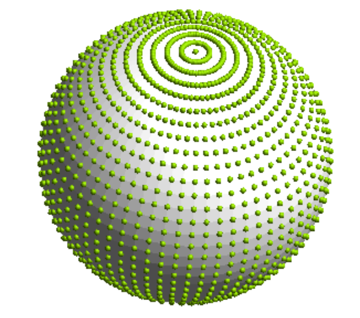
- Very fast, 5-point stencils
- Decoupling of domain in subdomains

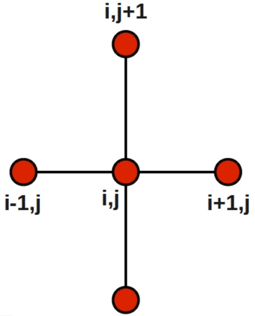
Higher-Order Finite Difference

Fourth order Finite difference:

$$\frac{\partial u_i}{\partial t} = -\left(\frac{-f_{i+2} + 8f_{i+1} - 8f_{i-1} + f_{i-2}}{12\Delta x}\right) + g_i$$

- Very fast, 5-point stencils
- Decoupling of domain in subdomains
- Structured grids
- Unnecessary small grid-spacing at higher latitudes





Finite Volume method

Integrated equation

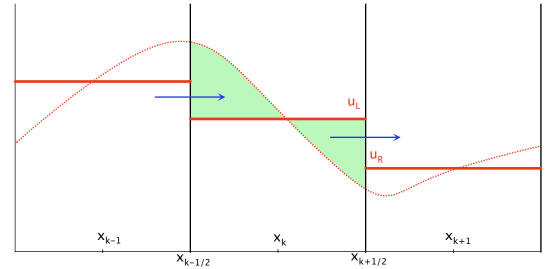
$$\int \left(\frac{\partial u}{\partial t} + \frac{\partial f(u,x)}{\partial x} - g \right) dx = 0$$

Finite Volume method

Integrated equation

$$\int \left(\frac{\partial u}{\partial t} + \frac{\partial f(u,x)}{\partial x} - g \right) dx = 0$$

$$\frac{\partial \langle u \rangle_k}{\partial t} + \frac{1}{\Delta x_k} \left[f^* \right]_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} - \langle g \rangle = 0$$

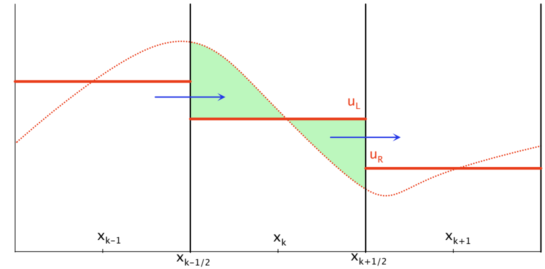


Finite Volume method

Integrated equation

$$\int \left(\frac{\partial u}{\partial t} + \frac{\partial f(u,x)}{\partial x} - g \right) dx = 0$$

$$\frac{\partial \langle u \rangle_k}{\partial t} + \frac{1}{\Delta x_k} \left[f^* \right]_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} - \langle g \rangle = 0$$



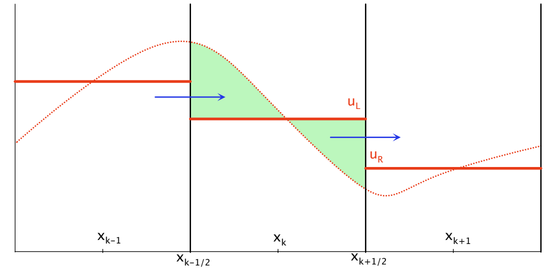
- Discontinuous solution
- Conservative:
  Flux = continuous

Finite Volume method

Integrated equation

$$\int \left(\frac{\partial u}{\partial t} + \frac{\partial f(u,x)}{\partial x} - g \right) dx = 0$$

$$\frac{\partial \langle u \rangle_k}{\partial t} + \frac{1}{\Delta x_k} \left[f^* \right]_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} - \langle g \rangle = 0$$



- Discontinuous solution
- Conservative:
  Flux = continuous

Jump condition at $x_{k+\frac{1}{2}}$: $f(\langle u_L \rangle) \neq f(\langle u_R \rangle)$

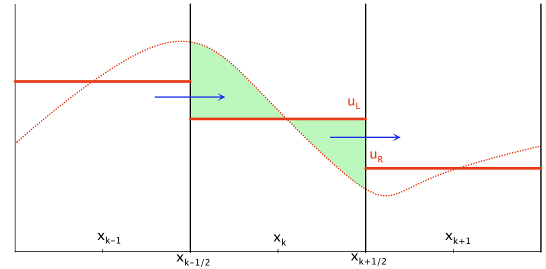
Riemann problem: $f^* = \mathcal{H}(u_L, u_R)$

Finite Volume method

Integrated equation

$$\int \left(\frac{\partial u}{\partial t} + \frac{\partial f(u,x)}{\partial x} - g \right) dx = 0$$

$$\frac{\partial \langle u \rangle_k}{\partial t} + \frac{1}{\Delta x_k} \left[f^* \right]_{x_{k-\frac{1}{2}}}^{x_{k+\frac{1}{2}}} - \langle g \rangle = 0$$

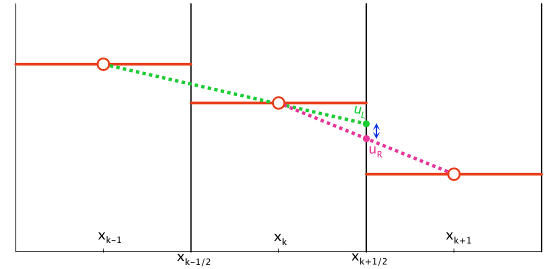


- Discontinuous solution
- Conservative:
  Flux = continuous

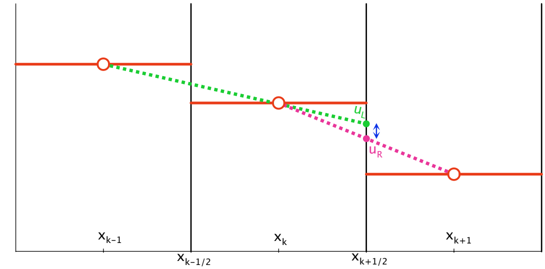
Jump condition at $x_{k+\frac{1}{2}}$: $f(\langle u_L \rangle) \neq f(\langle u_R \rangle)$

Riemann problem: $f^* = \mathcal{H}(u_L, u_R)$ $\rightarrow$ Provides upwinding

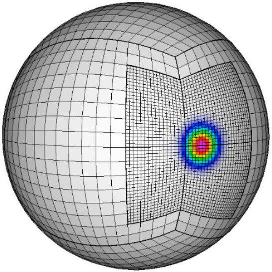
Second order Finite Volume Method



Second order Finite Volume Method



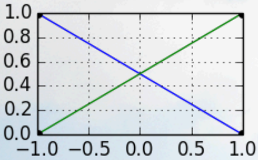
- Complex geometries on unstructured meshes
- Nested adaptive meshes
- Solution is defined in local manner
- Decoupling of domain in subdomains
- Natural upwinding couples cells
- Higher-order (>2) tedious and costly (extended stencils)
- Grid smoothness requirements

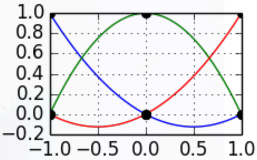


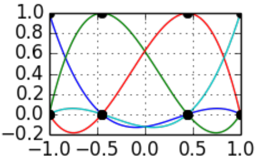
Finite Element Method – Continuous Galerkin

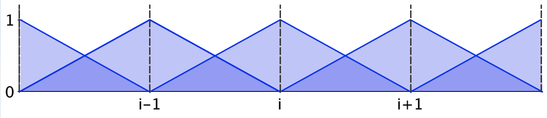
Equation is satisfied in global sense with solution defined nonlocally

$$\int \left(\frac{\partial u_h}{\partial t} + \frac{\partial f_h}{\partial x} - g \right) L_j(x) dx = 0, \qquad u_h(x) = \sum_k^N u_k \; L_k(x)$$









- Continuity imposed
- L_j has Value 1 in point j, Value 0 everywhere else

Global system of equations: $$M \cdot \frac{d\mathbf{u}_h}{dt} + S \cdot \mathbf{f}_h = M \cdot \mathbf{g}_h$$

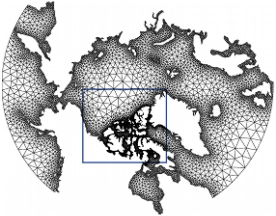
Mass matrix M : $$M_{ij} = \int_\Omega L_i(x) L_j(x) dx$$

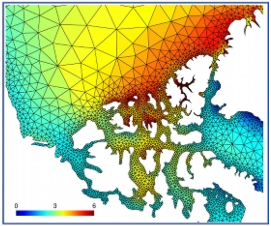
Stiffness matrix S : $$S_{ij} = \int_\Omega L_i(x) \frac{dL_j(x)}{dx} dx$$

Global system of equations: $$M \cdot \frac{d\mathbf{u}_h}{dt} + S \cdot \mathbf{f}_h = M \cdot \mathbf{g}_h$$

Mass matrix M : $$M_{ij} = \int_\Omega L_i(x) L_j(x) dx$$

Stiffness matrix S : $$S_{ij} = \int_\Omega L_i(x) \frac{dL_j(x)}{dx} dx$$





- High-order accuracy with compact flexible elements
- Complex geometries on unstructured meshes
- Implicit in time (Linear System Solver)
- Not well suited for problems with direction
- Everything is coupled through Mass matrix

Discontinuous Higher-Order Methods

We want a method that combines

- the flexibility of high-order elements of FEM
- the locality and scalability of FVM

Discontinuous Higher-Order Methods

We want a method that combines

- the flexibility of high-order elements of FEM
- the locality and scalability of FVM

There exists a "family" of discontinuous higher-order methods with exactly these components

- Discontinuous Galerkin Method
- Spectral Volume Method
- Spectral Difference Method
- Flux Reconstruction Method

Idea behind Discontinuous Higher-Order Methods

Idea behind Discontinuous Higher-Order Methods

- Solution is described within one element as a high-order function (borrowed from Finite Element Method)
  - Polynomial of order P
  - Fourier series
  - Taylor series: $\langle u \rangle, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots$

Idea behind Discontinuous Higher-Order Methods

- Solution is described within one element as a high-order function (borrowed from Finite Element Method)
  - Polynomial of order P
  - Fourier series
  - Taylor series: $\langle u \rangle, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots$
- Solution is defined locally on a per element basis

Idea behind Discontinuous Higher-Order Methods

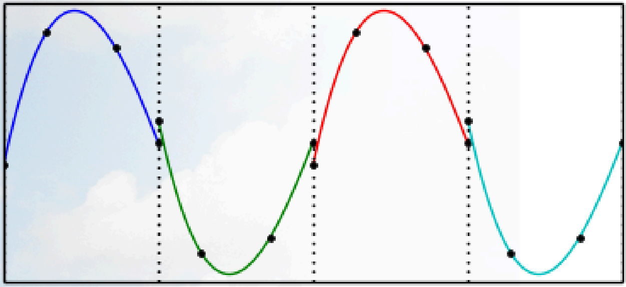
- Solution is described within one element as a high-order function (borrowed from Finite Element Method)
  - Polynomial of order P
  - Fourier series
  - Taylor series: $\langle u \rangle, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots$
- Solution is defined locally on a per element basis
- Solution is not continuous across elements

Idea behind Discontinuous Higher-Order Methods

- Solution is described within one element as a high-order function (borrowed from Finite Element Method)
  - Polynomial of order P
  - Fourier series
  - Taylor series: $\langle u \rangle, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots$
- Solution is defined locally on a per element basis
- Solution is not continuous across elements
- Flux is to be made continuous with Riemann solver (borrowed from Finite Volume Method)

Idea behind Discontinuous Higher-Order Methods

- Solution is described within one element as a high-order function (borrowed from Finite Element Method)
  - Polynomial of order P
  - Fourier series
  - Taylor series: $\langle u \rangle, \frac{\partial u}{\partial x}, \frac{\partial^2 u}{\partial x^2}, \ldots$
- Solution is defined locally on a per element basis
- Solution is not continuous across elements
- Flux is to be made continuous with Riemann solver (borrowed from Finite Volume Method)



P3 basis functions

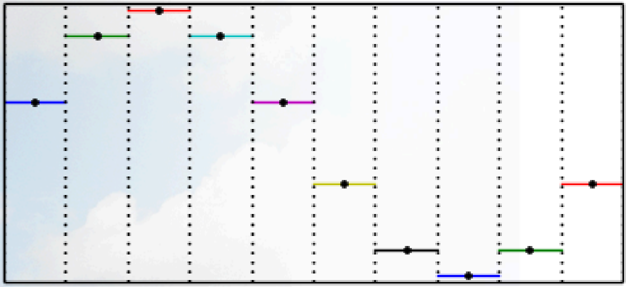
Observations

- Duplicated points at element interfaces (= more work)
- Solution does not look too nice as it is discontinuous
- Discontinuity does not affect high-order accuracy
- Discontinuity decouples elements (boundary conditions)
- Parallel efficiency outweighs more work

Observations

- Duplicated points at element interfaces (= more work)
- Solution does not look too nice as it is discontinuous
- Discontinuity does not affect high-order accuracy
- Discontinuity decouples elements (boundary conditions)
- Parallel efficiency outweighs more work

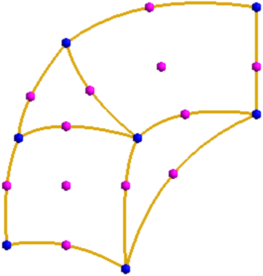
Interestingly: 1^{st} order corresponds to Standard Finite Volume



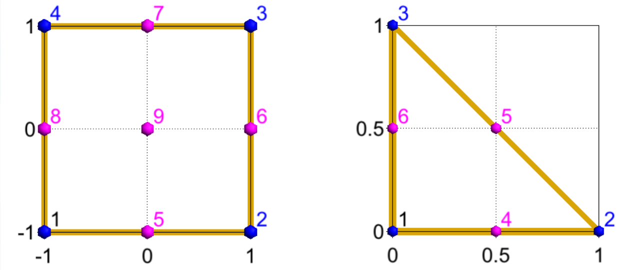
P0 basis functions

High-Order elements

- Extra points inside an element
- Effective increase in resolution
- Curved elements can align with coast lines
- Local mapping to standard element in parametric coördinates



Parametric coördinates



$$\bar{\bar{J}} = \frac{\partial(x,y)}{\partial(\xi,\eta)} = \begin{bmatrix} x_\xi & y_\xi \\ x_\eta & y_\eta \end{bmatrix}$$

$$\text{Volume} \propto \det(\bar{\bar{J}})$$

Mathematical element operations

Lagrange polynomials

Interpolation

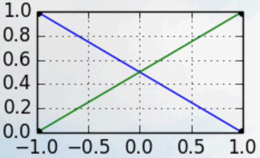
$$q(\xi) = \sum_{j=1}^{N} Q_j \; L_j(\xi) \qquad \text{with} \qquad L_j(\xi) = \prod_{k \neq j}^{N} \frac{\xi - \xi_k}{\xi_j - \xi_k}$$

Lagrange polynomials

Interpolation

$$q(\xi) = \sum_{j=1}^{N} Q_j \; L_j(\xi) \qquad \text{with} \qquad L_j(\xi) = \prod_{k \neq j}^{N} \frac{\xi - \xi_k}{\xi_j - \xi_k}$$

P1 (N=2)

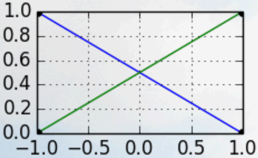


Lagrange polynomials

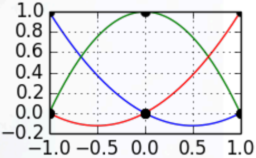
Interpolation

$$q(\xi) = \sum_{j=1}^{N} Q_j \; L_j(\xi) \qquad \text{with} \qquad L_j(\xi) = \prod_{k \neq j}^{N} \frac{\xi - \xi_k}{\xi_j - \xi_k}$$

P1 (N=2)



P2 (N=3)

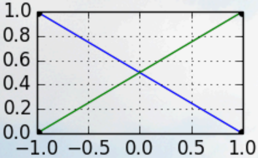


Lagrange polynomials

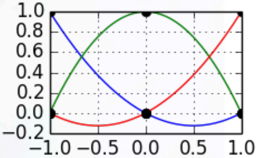
Interpolation

$$q(\xi) = \sum_{j=1}^{N} Q_j \; L_j(\xi) \qquad \text{with} \qquad L_j(\xi) = \prod_{k \neq j}^{N} \frac{\xi - \xi_k}{\xi_j - \xi_k}$$

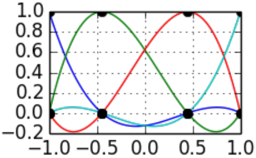
P1 (N=2)



P2 (N=3)



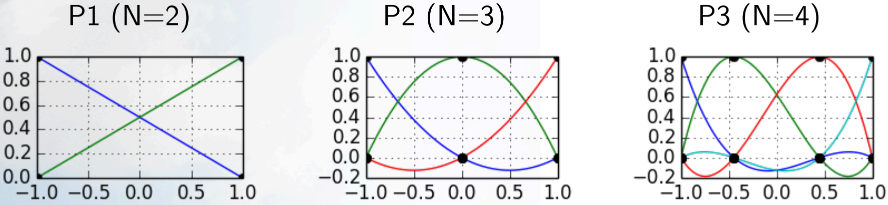
P3 (N=4)



Lagrange polynomials

Interpolation

$$q(\xi) = \sum_{j=1}^{N} Q_j \; L_j(\xi) \qquad \text{with} \qquad L_j(\xi) = \prod_{k \neq j}^{N} \frac{\xi - \xi_k}{\xi_j - \xi_k}$$

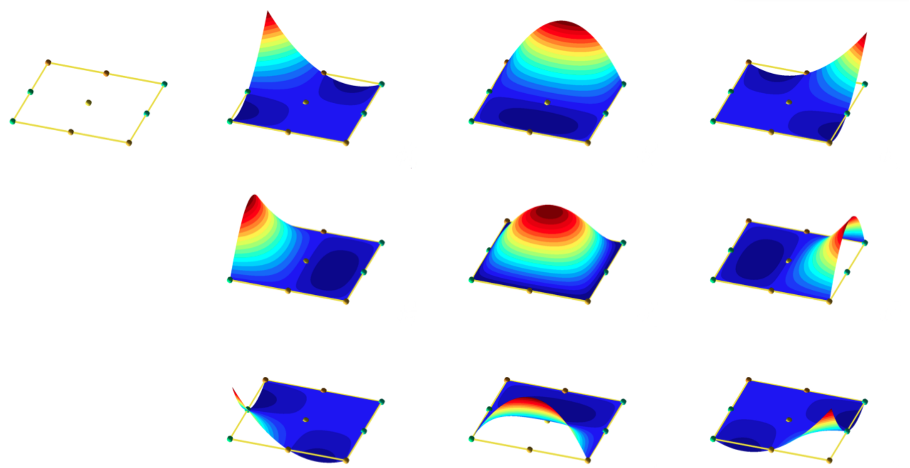


Differentiation

$$\frac{\partial q}{\partial \xi}(\xi) = \sum_{j=1}^{N} Q_j \; \frac{\partial L_j}{\partial \xi}(\xi) \qquad \text{with} \qquad \frac{\partial L_j}{\partial \xi}(\xi) = \sum_{i \neq j}^{N} \frac{1}{\xi_j - \xi_i} \prod_{\substack{k \neq i \\ k \neq j}}^{N} \frac{\xi - \xi_k}{\xi_j - \xi_k}$$

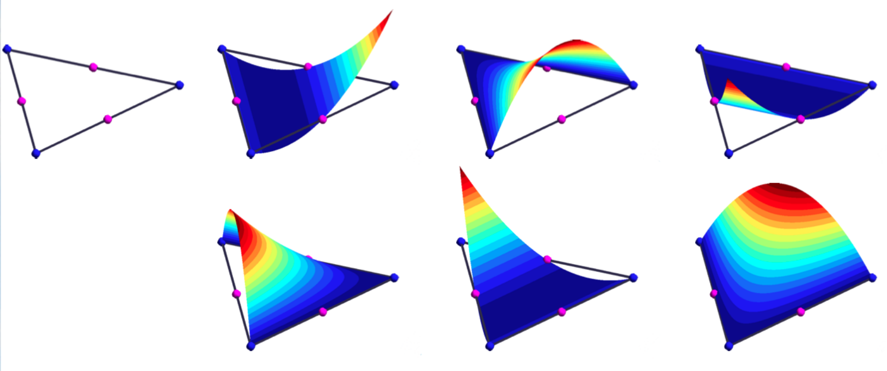
Quadrilateral P2 (N=9)

$$q(\xi, \eta) = \sum_{j=1}^{N} Q_j \; L_j(\xi, \eta)$$



Triangle P2 (N=6)

$$q(\xi, \eta) = \sum_{j=1}^{N} Q_j \; L_j(\xi, \eta)$$



Element Integrals using quadrature

A quadrature rule approximates an integral using a weighted sum:

$$\int_{-1}^{+1} f(x)dx \approx \sum_{k=1}^{n} w_k^{\text{quad}} \; f(x_k^{\text{quad}})$$

- x_k^{quad} are quadrature points
- w_k^{quad} are quadrature weights

Element Integrals using quadrature

A quadrature rule approximates an integral using a weighted sum:

$$\int_{-1}^{+1} f(x)dx \approx \sum_{k=1}^{n} w_k^{\mathrm{quad}} \; f(x_k^{\mathrm{quad}})$$

- x_k^{quad} are quadrature points
- w_k^{quad} are quadrature weights

One of the most widely used families of quadrature rules is Gauss-Legendre quadrature:

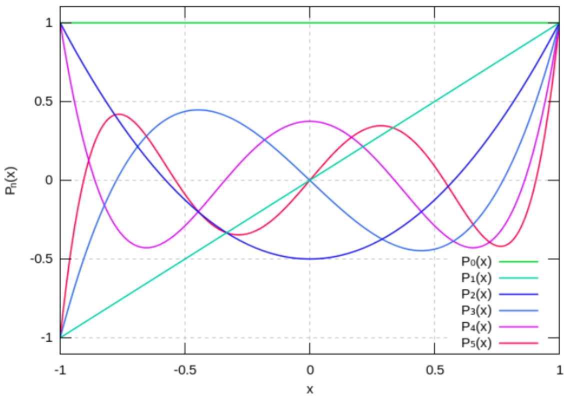
- Gauss-Legendre quadrature rule with n points and n weights can integrate a polynomial of degree $2n-1$ exactly!
- x_k are distributed like the roots of the Legendre polynomial $P_n(x)$
- w_k are then: $w_k = \frac{2}{(1-x_k^2)P_n'(x_k)^2}$

Legendre polynomials:

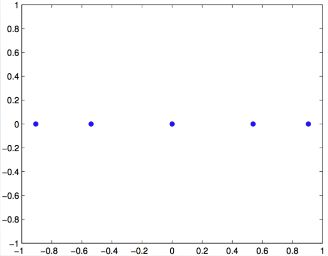
$$P_0(x) = 1, \qquad P_1(x) = x,$$

$$P_n(x) = \frac{2n-1}{n}\; P_{n-1}(x)\; x - \frac{n-1}{n} P_{n-2}(x)$$

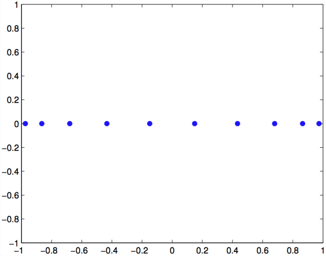
$$P_n'(x) = \frac{n}{1-x^2}\left(P_{n-1}(x) - P_n(x)\; x\right)$$



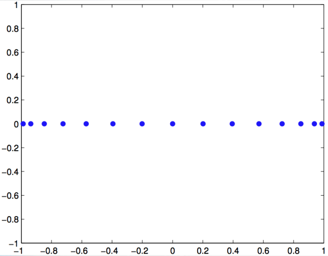
Gauss-Legendre quadrature points clustered towards ± 1:



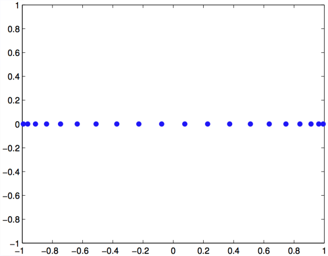
5 points



10 points



15 points



20 points

The Gauss-Legendre quadrature points and weights have been extensively tabulated for $x \in [-1, 1]$

Number of points (n)	Quadrature points	Quadrature weights
1	0	2
2	$-1/\sqrt{3}, 1/\sqrt{3}$	1, 1
3	$-\sqrt{3/5}, 0, \sqrt{3/5}$	5/9, 8/9, 5/9
⋮	⋮	⋮

`quadrature.py`: python-program provides points/weights with $x \in [-1, 1]$ for any n

The Gauss-Legendre quadrature points and weights have been extensively tabulated for $x \in [-1, 1]$

Number of points (n)	Quadrature points	Quadrature weights
1	0	2
2	$-1/\sqrt{3}, 1/\sqrt{3}$	1, 1
3	$-\sqrt{3/5}, 0, \sqrt{3/5}$	5/9, 8/9, 5/9
⋮	⋮	⋮

`quadrature.py`: python-program provides points/weights with $x \in [-1, 1]$ for any n

Useful for exact integration of Lagrange polynomials.

The Gauss-Legendre quadrature points and weights have been extensively tabulated for $x \in [-1, 1]$

Number of points (n)	Quadrature points	Quadrature weights
1	0	2
2	$-1/\sqrt{3}, 1/\sqrt{3}$	1, 1
3	$-\sqrt{3/5}, 0, \sqrt{3/5}$	5/9, 8/9, 5/9
$\vdots$	$\vdots$	$\vdots$

`quadrature.py`: python-program provides points/weights with $x \in [-1, 1]$ for any n

Useful for exact integration of Lagrange polynomials.

First interpolate to quadrature points!

The Gauss-Legendre quadrature points and weights have been extensively tabulated for $x \in [-1, 1]$

Number of points (n)	Quadrature points	Quadrature weights
1	0	2
2	$-1/\sqrt{3}, 1/\sqrt{3}$	1, 1
3	$-\sqrt{3/5}, 0, \sqrt{3/5}$	5/9, 8/9, 5/9
$\vdots$	$\vdots$	$\vdots$

`quadrature.py`: python-program provides points/weights with $x \in [-1, 1]$ for any n

Useful for exact integration of Lagrange polynomials.
First interpolate to quadrature points!

n	$2n-1$
1	$\leq$ P1
2	$\leq$ P3
3	$\leq$ P5

Discontinuous Galerkin method

A Discontinuous Galerkin scheme

Deriving the DG formulation

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{f} = 0$$

A Discontinuous Galerkin scheme

Deriving the DG formulation

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{f} = 0$$

Integrate over entire domain Ω:

$$\int_{\Omega} \left(\frac{\partial u_h}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{f}_h \right) L_i(\boldsymbol{x}) d\boldsymbol{x} = 0$$

A Discontinuous Galerkin scheme

Deriving the DG formulation

$$\frac{\partial u}{\partial t} + \nabla \cdot \boldsymbol{f} = 0$$

Integrate over entire domain Ω:

$$\int_\Omega \left(\frac{\partial u_h}{\partial t} + \nabla \cdot \boldsymbol{f}_h \right) L_i(\boldsymbol{x}) d\boldsymbol{x} = 0$$

Rewrite as sum of element integration:

$$\sum_e \left(\int_e \frac{\partial u_h}{\partial t} \; L_i(\boldsymbol{x}) \; d\boldsymbol{x} + \int_e \nabla \cdot \boldsymbol{f}_h \; L_i(\boldsymbol{x}) \; d\boldsymbol{x} \right) = 0$$

A Discontinuous Galerkin scheme

Deriving the DG formulation

$$\frac{\partial u}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{f} = 0$$

Integrate over entire domain Ω:

$$\int_\Omega \left(\frac{\partial u_h}{\partial t} + \boldsymbol{\nabla} \cdot \boldsymbol{f}_h \right) L_i(\boldsymbol{x}) d\boldsymbol{x} = 0$$

Rewrite as sum of element integration:

$$\sum_e \left(\int_e \frac{\partial u_h}{\partial t} \; L_i(\boldsymbol{x}) \; d\boldsymbol{x} + \int_e \boldsymbol{\nabla} \cdot \boldsymbol{f}_h \; L_i(\boldsymbol{x}) \; d\boldsymbol{x} \right) = 0$$

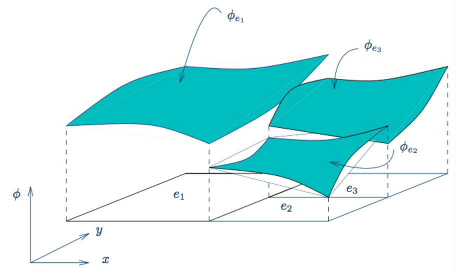
Integrating by parts:

$$\sum_e \left(\int_e \frac{\partial u_h}{\partial t} \; L_i(\boldsymbol{x}) \; d\boldsymbol{x} + \oint_{\partial e} L_i(\boldsymbol{x}) \; \boldsymbol{f}^* \cdot \boldsymbol{n} \; d\boldsymbol{x} - \int_e \boldsymbol{\nabla} L_i(\boldsymbol{x}) \cdot \boldsymbol{f} \; d\boldsymbol{x} \right) = 0$$

$$\sum_e \left(\int_e \frac{\partial u_h}{\partial t} \, L_i(\boldsymbol{x}) \; d\boldsymbol{x} + \oint_{\partial e} L_i(\boldsymbol{x}) \; \boldsymbol{f}^* \cdot \boldsymbol{n} \; d\boldsymbol{x} - \int_e \nabla L_i(\boldsymbol{x}) \cdot \boldsymbol{f} \; d\boldsymbol{x} \right) = 0$$

This can be satisfied for each element locally:

$$\int_e \frac{\partial u_h}{\partial t} \, L_i(\boldsymbol{x}) \; d\boldsymbol{x} + \oint_{\partial e} L_i(\boldsymbol{x}) \; \boldsymbol{f}^* \cdot \boldsymbol{n} \; d\boldsymbol{x} - \int_e \nabla L_i(\boldsymbol{x}) \cdot \boldsymbol{f} \; d\boldsymbol{x} = 0$$



Riemann problem:
u_h is discontinuous at interfaces.
We need conservation.
Numerical flux function $\boldsymbol{f}^*$ must be unique and provides element coupling

Numerical Flux

Question: How should we choose $\boldsymbol{f}^*$ on the “faces” of an element?

Numerical Flux

Question: How should we choose $\boldsymbol{f}^*$ on the "faces" of an element?

Answer: Just like in FV, numerical flux on a face should depend on data in the two neighbouring elements.

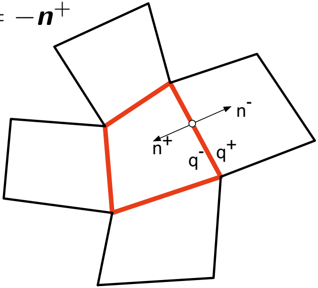
Numerical Flux

Question: How should we choose $\boldsymbol{f}^*$ on the "faces" of an element?

Answer: Just like in FV, numerical flux on a face should depend on data in the two neighbouring elements.

Let q^- (resp. q^+) denote the value of q on the interior (resp. exterior) face of an element

Let $\boldsymbol{n}^-$ (resp. $\boldsymbol{n}^+$) denote the outward normal vector on the face of the "local" (resp. "neighbour") element. Hence $\boldsymbol{n}^- = -\boldsymbol{n}^+$



Define "average" and "jump" operators:

$$\{\{q\}\} \equiv \frac{q^- + q^+}{2}$$

$$[\![q]\!] \equiv \boldsymbol{n}^- q^- + \boldsymbol{n}^+ q^+ = \boldsymbol{n}^-(q^- - q^+)$$

Numerical Flux

Roe scheme:

$$\boldsymbol{f}^* = \{\{\boldsymbol{f}\}\} + \frac{1}{2}|A|\ [\![u]\!] \qquad \text{with} \quad A \equiv \frac{\partial \boldsymbol{f}}{\partial u}$$

Rusanov scheme:

$$\boldsymbol{f}^* = \{\{\boldsymbol{f}\}\} + \frac{1}{2}\lambda_{\mathsf{max}}\ [\![u]\!] \qquad \text{with} \quad \lambda_{\mathsf{max}} \equiv \text{max wave speed}$$

Consider 1D linear advection: $\boldsymbol{f} = au$, and a is advection speed

$$\begin{aligned}\boldsymbol{f}^* &= \frac{1}{2}(au^- + au^+) + \frac{|a|}{2}(u^- - u^+)\\ &= u^-(\frac{a}{2} + \frac{|a|}{2}) + u^+(\frac{a}{2} - \frac{|a|}{2})\\ &= \begin{cases} au^- & \text{if } a > 0\\ au^+ & \text{if } a < 0\end{cases}\end{aligned}$$

Numerical Flux Properties

- Stability: upwind according to flow direction
- Conservative: $\boldsymbol{f}^*$ is same computed when computed from the perspective of the neighbour element
- Consistent: $\boldsymbol{f}^* \rightarrow \boldsymbol{f}$ when $[\![u]\!] \rightarrow 0$
- Rusanov scheme is much more dissipative than Roe scheme

Finite Volume

The jump $[\![u]\!]$ is usually large, and Roe scheme is preferred.

Discontinuous Higher-Order methods

The jump $[\![u]\!]$ can be very small, making the cheaper Rusanov scheme an attractive choice.

Back to the DG scheme

$$\int_e \frac{\partial u_h}{\partial t} \, L_i(\boldsymbol{x}) \; dx + \oint_{\partial e} L_i(\boldsymbol{x}) \; \boldsymbol{f}^* \cdot \boldsymbol{n} \; d\boldsymbol{x} - \int_e \boldsymbol{\nabla} L_i(\boldsymbol{x}) \cdot \boldsymbol{f} \;\; d\boldsymbol{x} = 0$$

Consider the special P0 case where $L_i(\boldsymbol{x}) = 1$, and thus $\boldsymbol{\nabla} L_i(\boldsymbol{x}) = 0$:

$$\int_e \frac{\partial u_h}{\partial t} \; d\boldsymbol{x} + \oint_{\partial e} \boldsymbol{f}^* \cdot \boldsymbol{n} \; d\boldsymbol{x} = 0$$

This is the definition of the Finite Volume scheme!

Although we started from the variational formulation like the Finite Element Method, the Discontinuous Galerkin Method can be reinterpreted as an extention of the Finite Volume Method

Implementing a DG scheme

$$\int_e \frac{\partial u_h}{\partial t} L_i(\boldsymbol{x})\, dx = \int_e \nabla L_i(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x})\, dx - \oint_{\partial e} L_i(\boldsymbol{x})\, \boldsymbol{f}^*(\boldsymbol{x}) \cdot \boldsymbol{n}\, dx$$

$$u_h(\boldsymbol{x}, t) = \sum_{j=1}^{N} u_j(t)\, L_j(\boldsymbol{x})$$

$$\sum_{j=1}^{N} \underbrace{\int_e L_i(\boldsymbol{x})\, L_j(\boldsymbol{x})\, dx}_{M_{ij}} \frac{\partial u_j}{\partial t} = \underbrace{\int_e \nabla L_i(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x})\, dx}_{\mathrm{RHS}_i^I} - \underbrace{\oint_{\partial e} L_i(\boldsymbol{x})\, \boldsymbol{f}^*(\boldsymbol{x}) \cdot \boldsymbol{n}\, dx}_{\mathrm{RHS}_i^{II}}$$

$$\mathbf{M}_e \frac{\partial \mathbf{U}_e}{\partial t} = \mathbf{RHS}_e^I - \mathbf{RHS}_e^{II}$$

$$\mathbf{M}_e \frac{\partial \mathbf{U}_e}{\partial t} = \mathbf{RHS}_e^{I} - \mathbf{RHS}_e^{II}$$

Mass matrix

$$M_{ij}^e = \int_e L_i(\boldsymbol{x})\ L_j(\boldsymbol{x})\ d\boldsymbol{x}$$

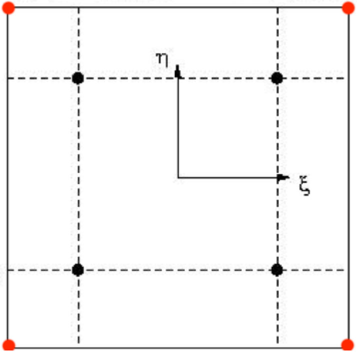
Computation in practice

Transform to parametric coordinates

$$M_{ij}^e = \int_e L_i(\xi)\ L_j(\xi)\ \left|\frac{d\boldsymbol{x}}{d\xi}\right|\ d\xi$$

Gaussian Quadrature

$$M_{ij}^e = \sum_q^{N_q} w_q\ L_i(\xi_q)\ L_j(\xi_q)\ |J_q^e|$$



$$\mathbf{M}_e \frac{\partial \mathbf{U}_e}{\partial t} = \mathbf{RHS}_e^I - \mathbf{RHS}_e^{II}$$

First RHS term

$$\mathrm{RHS}_i^I = \int_e \nabla L_i(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x})\ d\boldsymbol{x}$$

Computation in practice

Transform to parametric coordinates

$$\mathrm{RHS}_i^I = \int_e \nabla L_i(\boldsymbol{\xi}) \overline{\overline{J}}^{-1} \cdot \boldsymbol{f}(\boldsymbol{\xi})\ |J| d\boldsymbol{\xi}$$

Gaussian Quadrature

$$\mathrm{RHS}_i^I = \sum_q^{N_q} w_q\ \nabla L_i(\boldsymbol{\xi}_q)\, |J_q|\, \overline{\overline{J}}_q^{-1} \cdot \boldsymbol{f}(\boldsymbol{\xi}_q)$$

Approximation of order of scheme: $\boldsymbol{f}(\boldsymbol{\xi}) \approx \sum_{j=1}^{N} L_j(\boldsymbol{\xi})\ \boldsymbol{f}(u_j)$

$$\mathrm{RHS}_i^I \approx \sum_{j=1}^{N} \underbrace{\sum_q^{N_q} w_q\ L_i(\boldsymbol{\xi}_q) \nabla L_i(\boldsymbol{\xi}_q)\, |J_q|\, \overline{\overline{J}}_q^{-1}}_{S_{ij}} \cdot \boldsymbol{f}_j$$

Stiffness or Advection matrix $\qquad \mathbf{RHS}_e^I \approx \mathbf{S}_e\ \mathbf{F}_e$

$$\mathbf{M}_e \frac{\partial \mathbf{U}_e}{\partial t} = \mathbf{RHS}_e^{I} - \mathbf{RHS}_e^{II}$$

Second RHS term

$$\mathrm{RHS}_i^{II} = \oint_{\partial e} L_i(\boldsymbol{x})\ \boldsymbol{f}^*(\boldsymbol{x}) \cdot \boldsymbol{n}\ d\boldsymbol{x}$$

Computation in practice

Transform to parametric coordinates

$$\mathrm{RHS}_i^{II} = \sum_{f=1}^{N_f} \int_{\partial e_f} L_i(\boldsymbol{\xi})\ \boldsymbol{f}^*(\boldsymbol{\xi}) \cdot \boldsymbol{n}\ |J_f| d\boldsymbol{\xi}$$

Example in 1D

$$\mathrm{RHS}_i^{II} = L_i(\xi_L)\ \boldsymbol{f}^*(\xi_L) \cdot (-1)\ +\ L_i(\xi_R)\ \boldsymbol{f}^*(\xi_R) \cdot (+1)$$

$$\mathrm{RHS}_i^{II} = [L_i(\xi_L) \quad L_i(\xi_R)] \cdot \begin{bmatrix} -\boldsymbol{f}^*(\xi_L) \\ +\boldsymbol{f}^*(\xi_R) \end{bmatrix}$$

$$\mathbf{RHS}_e^{II} = \mathbf{H}_e\ \mathbf{Fn}_e^*$$

Collecting the pieces

$$\mathbf{M}_e \frac{\partial \mathbf{U}_e}{\partial t} = \mathbf{RHS}_e^I - \mathbf{RHS}_e^{II}$$

$$\int_e \frac{\partial u_h}{\partial t} \, L_i(\boldsymbol{x}) \, d\boldsymbol{x} = \int_e \nabla L_i(\boldsymbol{x}) \cdot \boldsymbol{f}(\boldsymbol{x}) \, d\boldsymbol{x} - \oint_{\partial e} L_i(\boldsymbol{x}) \, \boldsymbol{f}^*(\boldsymbol{x}) \cdot \boldsymbol{n} \, d\boldsymbol{x}$$

Implemented as matrix products

$$\mathbf{M}_e \frac{\partial \mathbf{U}_e}{\partial t} = \mathbf{S}_e \, \mathbf{F}_e - \mathbf{H}_e \, \mathbf{Fn}_e^*$$

$$\frac{\partial \mathbf{U}_e}{\partial t} = \mathbf{M}_e^{-1} \, \mathbf{S}_e \, \mathbf{F}_e - \mathbf{M}_e^{-1} \, \mathbf{H}_e \, \mathbf{Fn}_e^*$$

$$\frac{\partial \mathbf{U}_e}{\partial t} = \mathbf{Ds}_e \, \mathbf{F}_e - \mathbf{Dh}_e \, \mathbf{Fn}_e^*$$

$$\mathbf{U}_e = [u_1, u_2, u_3, \dots, u_N] \qquad \mathbf{F}_e = [\boldsymbol{f}(u_1), \boldsymbol{f}(u_2), \boldsymbol{f}(u_3), \dots, \boldsymbol{f}(u_N)]$$

Demonstration 1D DGM

Spectral Difference Method

Spectral Difference Method

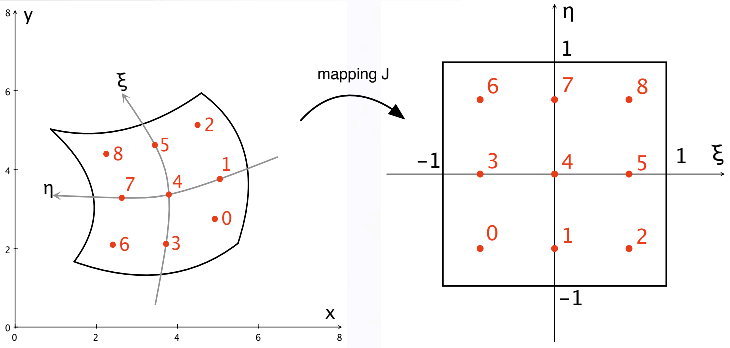
Why this name?

- Spectral: Higher-Order solution can be described as a Fourier Series
- Difference: Equations are solved in differential form, like Finite Difference

Some properties

- Differential form of equations $\rightarrow$ no quadrature necessary
- Unstructured grids / Complex geometries
- Compact stencil
- Shape functions provide higher order
- Upwinding between cells through Riemann solver
- Very intuitive approach

Spectral Difference method



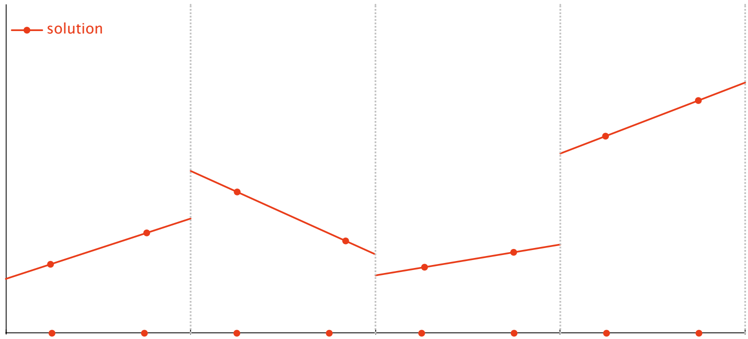
$$\frac{\partial q}{\partial t} + \nabla \cdot \mathbf{f} = 0 \qquad \frac{\partial \tilde{q}}{\partial t} + \tilde{\nabla} \cdot \tilde{\mathbf{f}} = 0$$

with mapping $\overline{\overline{J}} = {}^{\partial \vec{x}}/_{\partial \vec{\xi}}$

$$\tilde{q} = |J| \; q \qquad \tilde{\mathbf{f}} = \begin{bmatrix} \tilde{f}_\xi \\ \tilde{f}_\eta \\ \tilde{f}_\zeta \end{bmatrix} = |J| \; \overline{\overline{J}}^{-1} \; \mathbf{f}$$

Spectral Difference method

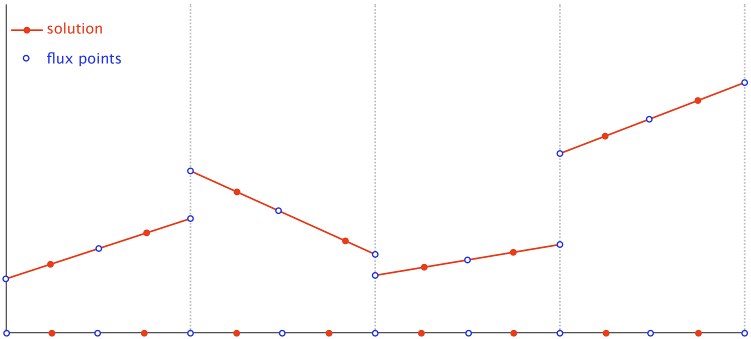
Example: 1D 2^{nd} order scheme $\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} = 0$ with $f = q$



- Solution $q(\xi)$ is discontinuous and linear
- Goal is to get $\frac{\partial}{\partial \xi}$ to 2^{nd} order accuracy

Spectral Difference method

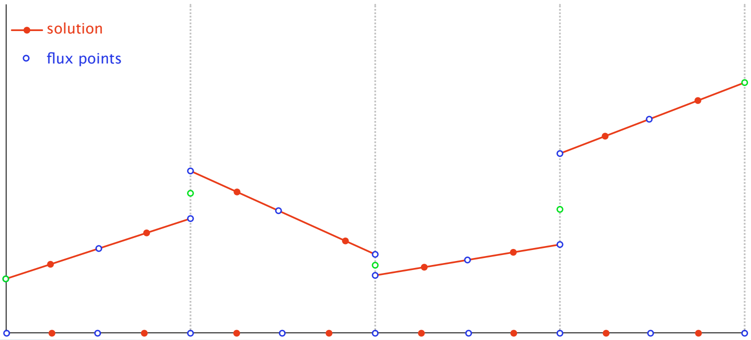
Example: 1D 2^{nd} order scheme $\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} = 0$ with $f = q$



- Extrapolate solution $q(\xi)$ to "flux points"
- Compute flux

Spectral Difference method

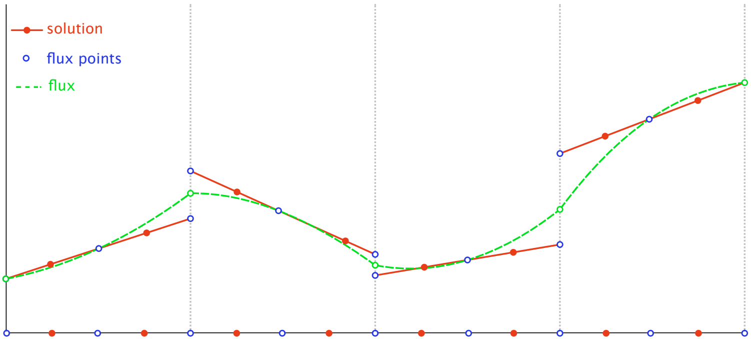
Example: 1D 2$^{\text{nd}}$ order scheme $\quad \frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} = 0 \quad$ with $f = q$



- Extrapolate solution $q(\xi)$ to “flux points”
- Compute flux
- Compute Riemann flux for conservation and upwinding between cells

Spectral Difference method

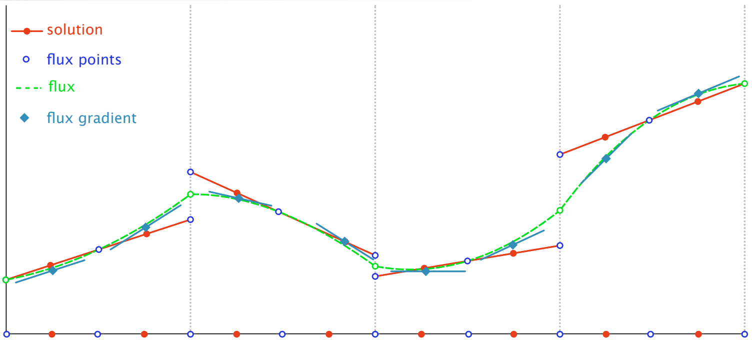
Example: 1D 2^{nd} order scheme $\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} = 0$ with $f = q$



- Flux is now a parabolic function

Spectral Difference method

Example: 1D 2^{nd} order scheme $\frac{\partial q}{\partial t} + \frac{\partial f}{\partial x} = 0$ with $f = q$



- Flux is now a parabolic function
- Compute gradient of parabolic function in "solution points"

Stability of the Spectral Difference Method

Question:

- Where should we put the solution points?
- Where should we put the flux points?

Stability of the Spectral Difference Method

Question:

- Where should we put the solution points?
- Where should we put the flux points?

Answer:

- Free to choose location of solution points
- Flux points however not:
  - Points on interface for element coupling
  - Stability analysis required (not covered)
  - One more point than solution points (in 1D)

Stability of the Spectral Difference Method

Question:

- Where should we put the solution points?
- Where should we put the flux points?

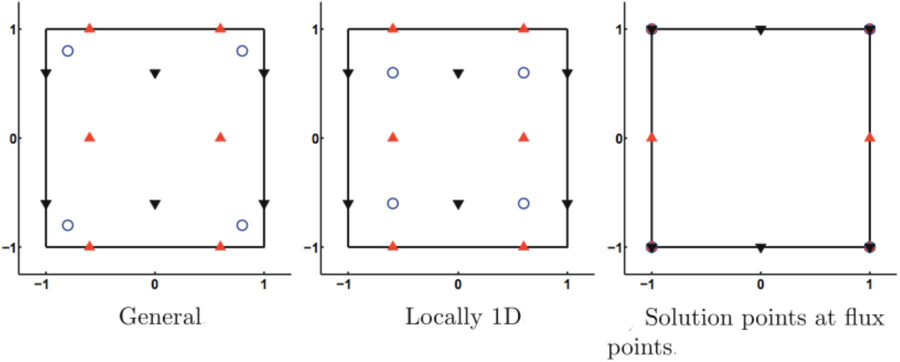
Answer:

- Free to choose location of solution points
- Flux points however not:
  - Points on interface for element coupling
  - Stability analysis required (not covered)
  - One more point than solution points (in 1D)

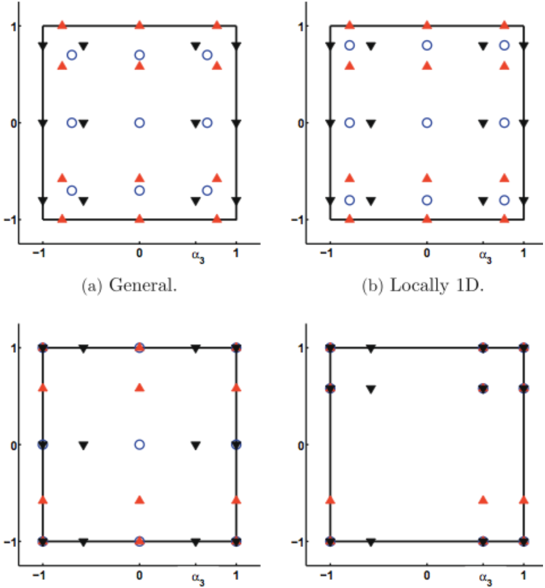
Flux point location

Roots of Legendre polynomial plus $[\xi = -1, \;\; \xi = +1]$

Solution / Flux Point distributions for SD



Second-order quadrilateral SD cells. Solution points (○) and ξ_1- (▼) and ξ_2-flux points (▲).



(a) General.

(b) Locally 1D.

(c) Most solution points at flux points, symmetrical.

(d) All solution points at flux points, asymmetrical.

Implementing a SD method

1. Interpolate solution to all flux points (could be optimized depending on solution point distribution)

$$q_f = \sum_{j=1}^{N} q_j \; L_j^{\mathrm{sol}}(\xi_f) \qquad \rightarrow Q^e_{\mathrm{flxpts}} = \mathbf{I}_f \, Q^e$$

2. Compute numerical flux in interface flux points

$$\tilde{\mathbf{F}}^e_{\mathrm{interface}} = |J| \; \overline{\overline{J}}^{-1} \; \boldsymbol{f}^*$$

3. Compute fluxes in internal flux points

$$\tilde{\mathbf{F}}^e_{\mathrm{internal}} = |J| \; \overline{\overline{J}}^{-1} \; \boldsymbol{f}(Q^e_{\mathrm{flxpts}})$$

4. Compute flux divergence

$$\frac{\partial \tilde{\boldsymbol{f}}}{\partial \xi} = \sum_{j=1}^{N_f} \tilde{\boldsymbol{f}}_j \; \frac{\partial L_j^{\mathrm{flx}}(\xi_f)}{\partial \xi} \qquad \rightarrow \tilde{\boldsymbol{\nabla}} \cdot \tilde{\mathbf{F}}^e = \tilde{\mathbf{D}} \; \tilde{\mathbf{F}}^e$$

5. Update solution:

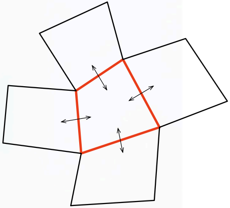
$$\frac{\partial Q^e}{\partial t} = \frac{1}{|J|} \tilde{\mathbf{D}} \; \tilde{\mathbf{F}}^e$$

Demonstration 1D SDM

Parallel efficiency

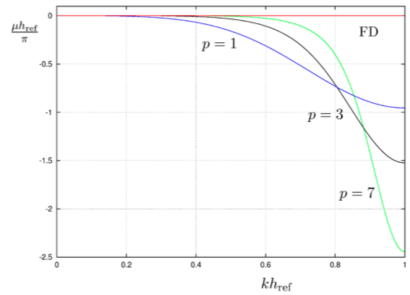
Discontinuous Higher-Order methods offer huge potential

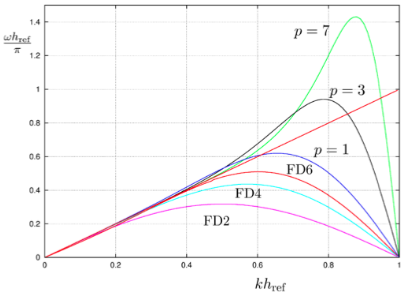
- Matrix multiplications for near peak FLOP-rates
  I have shown you can write the entire method in matrix-vector notation
- Avoiding global communication
  Every element acts as a standalone domain with boundary conditions

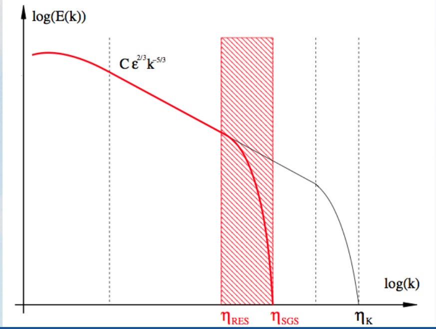


- Mapping of problem to nested hierarchical computer architectures
  - MPI – distributed memory
  - OpenMP – shared memory
  - Accelerators (e.g. GPU) – matrix multiplications

Spectral properties







Properties depend on choices for numerical flux, shape functions

Numerical damping of high wave numbers could make the method suitable for Implicit LES!

Concluding

- Introduction to Higher-Order accuracy on unstructured meshes
- Implementations for hyperbolic conservation laws
- There is lots more to consider (diffusion terms, monotonicity, time stepping, curved elements)
- Parallel efficiency as main driving force
- Implicit LES properties are to be examined

References

- Hesthaven, J.S., Warburton, T.: Nodal Discontinuous Galerkin Methods — Algorithms, Analysis, and Applications
- Liu, Y., Vinokur, M., & Wang, Z. (2006). Spectral difference method for unstructured grids I: Basic formulation. Journal of Computational Physics.
- Dhatt, G., Touzot, G.: The Finite Element Method Displayed