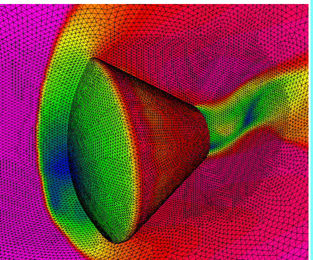
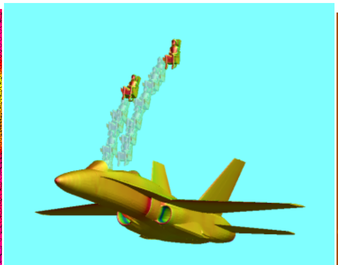
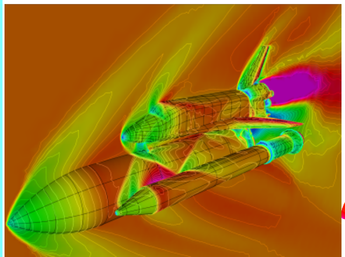
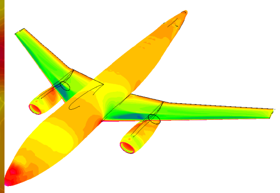
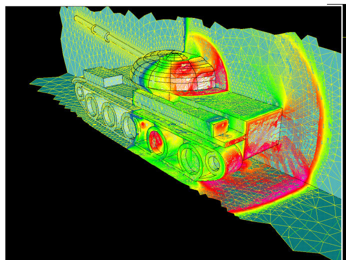
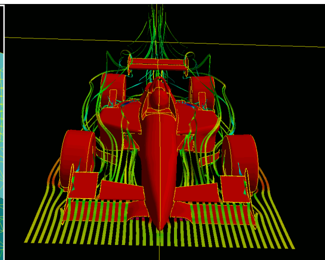
DEPARTMENT OF MECHANICAL AND AEROSPACE ENGINEERING

Reconstructed Discontinuous Galerkin Methods For Computational Fluid Dynamics

Hong Luo
Department of Mechanical and Aerospace Engineering
North Carolina State University

Presented at Beijing University of Aeronautics and Astronautics
Beijing, China
June 30, 2014

Outline

- Governing Equations
- Discontinuous Galerkin (DG) Finite Element Methods for the Compressible Navier-Stokes Equations
  - Nodal DG Methods
  - Modal DG Methods
- Discretization of Convective (Inviscid) Fluxes
- Reconstructed DG (RDG(PnPm)) Methods
  - K-exact reconstruction
  - Green-Gauss Reconstruction
  - Least-Squares Reconstruction
  - WENO Reconstruction
  - Hierarchical WENO Reconstruction
- Discretization of Diffusive (Viscous) Fluxes
- Temporal Discretizations
  - Explicit Methods
  - Implicit Methods

Governing Equations

Compressible Navier-Stokes Equations:

$$\frac{\partial \mathbf{U}(x,t)}{\partial t} + \frac{\partial \mathbf{F}_k(\mathbf{U}(x,t))}{\partial x_k} = \frac{\partial \mathbf{G}_k(\mathbf{U}(x,t))}{\partial x_k}$$

where the conservative state, inviscid flux, and viscous flux vectors

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ \rho e \end{pmatrix} \qquad \mathbf{F}_j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p\delta_{ij} \\ u_j(\rho e + p) \end{pmatrix} \qquad \mathbf{G}_j = \begin{pmatrix} 0 \\ \sigma_{ij} \\ u_l \sigma_{lj} + q_j \end{pmatrix}$$

Here ρ, *p*, and *e* denote the density, pressure, and specific total energy of the fluid, respectively, and u_i is the velocity of the flow in the coordinate direction x_i

δ_{ij} : Kronecker tensor.

Compressible Navier-Stokes Equations

The pressure, the viscous stress tensor σ_{ij} and the heat flux vector q_j are given by

$$p = (\gamma - 1)\rho(e - \frac{1}{2}u_j u_j) \quad \sigma_{ij} = \mu(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}) - \frac{2}{3}\mu\frac{\partial u_k}{\partial x_k}\delta_{ij} \quad q_j = k\frac{\partial T}{\partial x_j}$$

where $\gamma(=1.4)$ ratio of the specific heats, μ: molecular viscosity, k: thermal conductivity coefficient. The molecular viscosity can be determined through Sutherland's law

$$\frac{\mu}{\mu_0} = \left(\frac{T}{T_0}\right)^{\frac{3}{2}} \frac{T_0 + S}{T + S}$$

μ_0 denotes the viscosity at the reference temperature T_0, and S is a constant which for air assumes the value $S = 110^oK$. The temperature of the fluid T is determined by

$$T = \frac{p}{R\rho}$$

R: ideal gas constant.

Compressible Euler Equations

If we neglect the loss of the heat by thermal diffusion (k=0) and the effects of viscosity(μ=0), the Navier-Stokes equations become the Euler equations:

$$\frac{\partial \mathbf{U}(x,t)}{\partial t} + \frac{\partial \mathbf{F}_k(\mathbf{U}(x,t))}{\partial x_k} = 0$$

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho u_i \\ \rho e \end{pmatrix} \qquad \mathbf{F}_j = \begin{pmatrix} \rho u_j \\ \rho u_i u_j + p\delta_{ij} \\ u_j(\rho e + p) \end{pmatrix}$$

$$p = (\gamma - 1)\rho(e - \frac{1}{2}u_j u_j)$$

Boundary Conditions

- Dirichlet boundary conditions

$$\mathbf{U}_\Gamma = \mathbf{U}_b$$

- No-slip nall boundary conditions:

$$\text{Isothermal wall: } \mathbf{V}_\Gamma = \mathbf{0}, \text{ and } T_\Gamma = T_w$$

$$\text{Adiabatic wall : } \mathbf{V}_\Gamma = \mathbf{0}, \text{ and } \left.\frac{\partial T}{\partial n}\right|_\Gamma = 0$$

- Slip wall boundary conditions:

$$\mathbf{V}n|_\Gamma=0$$

- Inflow/Outflow boundary conditions: characteristic boundary conditions
- Periodic boundary conditions: no boundary conditions

Discontinuous Galerkin (DG) Finite Element Methods

- Advantages:
  - Several useful mathematical properties with respect to conservation, stability, and convergence.
  - Easy extension to higher-order (>2nd) schemes.
  - Well suited for complex geometries.
  - Easy adaptive strategies, allowing implementation of *hp*-refinement and hanging nodes.
  - Compact and highly parallelizable.
  - Accuracy for low Mach number flows.

- Disadvantages:
  - Require higher-order boundary representation
    - →Geometric modeling capability
    - →Curved boundary elements
  - High computing costs (more degrees of freedom)
    - → CPU time
    - → Storage Requirements
  - Treatment of discontinuities (like all other high-order methods)
    - → Sensitive to the implementation of limiters
    - → Lead to loss of high-order accuracy

Variational (Weak) formulation

Find $\mathbf{U} \in V$ such that

$$\int_{\Omega} \frac{\partial \mathbf{U}}{\partial t} W d\Omega + \int_{\Gamma} \mathbf{F}_k \mathbf{n}_k d\Gamma - \int_{\Omega} \mathbf{F}_k \frac{\partial W}{\partial x_k} d\Omega = \int_{\Gamma} \mathrm{G}_k \mathbf{n}_k d\Gamma - \int_{\Omega} \mathrm{G}_k \frac{\partial W}{\partial x_k} d\Omega, \quad \forall W \in V$$

V: Solution space

W: Test function

Semi-discrete form

$$\Omega = \bigcup \Omega_e \qquad V_h^{\mathrm{P_n}} = \left\{ v_h \in \left[L_2(\Omega) \right]^{\mathrm{m}} : v_h\big|_{\Omega_e} \in \left[V_{\mathrm{P_n}}^{\mathrm{m}} \right] \ \forall \Omega_e \in \Omega \right\},$$

$$V_{\mathrm{P_n}}^{\mathrm{m}} = \mathrm{span}\ \left\{ \prod_{i=1}^{d} x_i^{\alpha_i} : 0 \le \alpha_i \le \mathrm{p_n}, 0 \le i \le d \right\},$$

m: dimension of conservative state vector
d: number of spatial dimension

Find $\mathbf{U}_h \in V_h^{\mathrm{P_n}}$ such as

$$\frac{\mathrm{d}}{\mathrm{dt}} \int_{\Omega_e} \mathrm{U}_h W_h d\Omega + \int_{\Gamma_e} \mathbf{F}_k(\mathrm{U}_h) \mathbf{n}_k W_h d\Gamma - \int_{\Omega_e} \mathrm{F}_k(\mathbf{U}_h) \frac{\partial W_h}{\partial x_k} d\Omega = \int_{\Gamma_e} \mathbf{G}_k(\mathrm{U}_h) \mathbf{n}_k W_h d\Gamma - \int_{\Omega_e} \mathrm{G}_k(\mathrm{U}_h) \frac{\partial W_h}{\partial x_k} d\Omega, \ \forall W_h \in V_h^{\mathrm{P_n}}$$

$\mathrm{U_h}$: piecewise polynomial function of degree p_n, which is discontinuous between the cell interfaces.

DG(P$_n$) Method

$$\frac{d}{dt}\int_{\Omega_e}\mathbf{U}_h B_i d\Omega + \int_{\Gamma_e}\mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k B_i d\Gamma - \int_{\Omega_e}\mathbf{F}_k(\mathbf{U}_h)\frac{\partial B_i}{\partial x_k}d\Omega = \int_{\Gamma_e}\mathbf{G}_k(\mathbf{U}_h)\mathbf{n}_k B_i d\Gamma - \int_{\Omega_e}\mathbf{G}_k(\mathbf{U}_h)\frac{\partial B_i}{\partial x_k}d\Omega, \quad 1 \le i \le \mathrm{N}$$

B_i(x): basis functions of the polynomials of degree P$_n$, $1 \le i \le$N.

N: dimension of the polynomial space P$_n$.

$\uparrow$

Discontinuous Galerkin method of degree Pn (DG(Pn)) : O(h^{n+1})

$\mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k = \mathbf{H}_\mathrm{k}(\mathbf{U}^L{}_h, \mathbf{U}^R{}_h, \mathbf{n}_k)$ $\leftarrow$ Numerical Riemann flux function

$$\mathbf{G}_k(\mathbf{U}_h, \frac{\partial \mathbf{U}_h}{\partial x_i})\mathbf{n}_k = \mathbf{H}_\mathrm{v}(\mathbf{U}_h^L, \mathbf{U}_h^R, \frac{\partial \mathbf{U}_h^L}{\partial x_i}, \frac{\partial \mathbf{U}_h^R}{\partial x_i}, \mathbf{n})$$

The computation of the viscous fluxes has to properly resolve the discontinuities at the interfaces.

DG(P$_n$) Method

$$\mathbf{U}_h\big|_{\Omega_e} = \sum_{j=1}^{N} \mathbf{U}_j^{\Omega_e}(t) B_j^{\Omega_e}(x) = \sum_{j=1}^{N} \mathbf{U}_j(t) B_j(x)$$

$$\left(\int_{\Omega_e} B_j B_i \, d\Omega\right) \frac{dU_j}{dt} = \mathrm{R}_i \qquad 1 \le i \le \mathrm{N}$$

$$\mathrm{R}_i = -\int_{\Gamma_e} \mathbf{F}_k \mathbf{n}_k B_i \, d\Gamma + \int_{\Omega_e} \mathbf{F}_k \frac{\partial B_i}{\partial x_k} d\Omega + \int_{\Gamma_e} \mathbf{G}_k \mathbf{n}_k B_i \, d\Gamma - \int_{\Omega_e} \mathbf{G}_k \frac{\partial B_i}{\partial x_k} d\Omega$$

$$\mathbf{M}_{\mathrm{N}\times\mathrm{N}} \frac{\mathrm{d}}{\mathrm{dt}} \begin{pmatrix} \mathrm{U}_1 \\ \mathrm{U}_2 \\ \bullet \\ \bullet \\ \mathrm{U}_\mathrm{N} \end{pmatrix} = \begin{pmatrix} \mathrm{R}_1 \\ \mathrm{R}_2 \\ \bullet \\ \bullet \\ \mathrm{R}_\mathrm{N} \end{pmatrix}$$

- No global mass matrix needs to be inverted.
- Inter-element communications are minimal.

Computation of the domain integral

$$R_i = \int_{\Omega_e} \mathbf{F}_k(\mathbf{U}_h)\frac{\partial B_i}{\partial x_k}d\Omega, \qquad 1 \le i \le \mathrm{N}$$

Computer Implementation:

```
do ielem = 1, nelem                    ! Loop over the elements
   do igaus = 1, ngaus                 ! Loop over the Gauss points
      ! Get the solution at the Gauss points
      Unkno = …
      ! Compute the fluxes at the Gauss points
      Fluxes = …
      ! Scatter it  to the RHS vector
      rhsel = rhsel + …
   enddo
enddo
```

Computation of boundary integral

$$R_i = -\int_{\Gamma_e} \mathbf{F}_k(\mathrm{U}_h)\mathbf{n}_k B_i d\Gamma, \qquad 1 \le i \le \mathrm{N}$$

Computer implementation (element-based data structure)

```
do ielem = 1, nelem                        ! Loop over the elements
     do ifele = 1, nfele                   ! Loop over the face of this element
     jelem = elsuel(ifele,ielem)           ! Adjacent element number
        do igaus = 1, ngaus                ! Loop over the Gauss points
             ! Get the solutions at the Gauss points
             Unkno_ielem = …               ! Solution at the left of Gauss point
             unkno_jelem = …               ! Solution at the right of Gauss point
             ! Compute the fluxes at the Gauss points by a Riemann flux functions
             Fluxes = …
            ! Scatter it  to the RHS vector
            rhsel_ielem = rhsel _ielem + …
        enddo      ! End of the do-loop over the Gauss points
     enddo         ! End of the do-loop over the faces of this element
Enddo              ! End of the do-loop over the elements
```

Computation of boundary integral

$$R_i = -\int_{\Gamma_e} \mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k B_i d\Gamma, \qquad 1 \le i \le \mathrm{N}$$

Computer implementation (faced-based data structure)

```
do iface = 1, nface                    ! Loop over the faces
   ielem = intfac(1,iface)             ! Left element of this  face
   jelem = intfac(2,iface)             ! Right element of this face
   do igaus = 1, ngaus                 ! Loop over the Gauss points of this face
        ! Get the solutions at the Gauss points
        Unkno_ielem = …                    ! Solution at the left of Gauss point
        unkno_jelem = …                    ! Solution at the right of Gauss point
        ! Compute the fluxes at the Gauss points by a Riemann flux functions
        Fluxes = …
        ! Scatter it  to the RHS vector
        rhsel_ielem = rhsel_ielem + …
        rhsel_jelem = rhsel_jelem - …
   enddo      ! End of the do-loop over the Gauss points
Enddo        ! End of the do-loop over the faces
```

Numerical Integration

Quadrature order requirements for the 2D Euler equations using conservative state variables for triangle

	Boundary (1-D)			Domain (2-D)		
	q=1	q=2	q=3	q=1	q=2	q=3
P=0	1	3	5			
P=1	2	5	7	3	3	4
P=2	3	7	7	4	5	6
P=3	7	9	9	6	7	8

where p is interpolation order and q is element geometry order.

Numerical Integration

Quadrature order requirements for the 3D Euler equations using conservative state variables for tetrahedral

	Boundary (2-D)			Domain (3-D)		
	q=1	q=2	q=3	q=1	q=2	q=3
P=0	1					
P=1	3	5	6	4	5	6
P=2	6	7	8	5	6	7
P=3	8	9	10	7	7	8

where p is interpolation order and q is element geometry order.

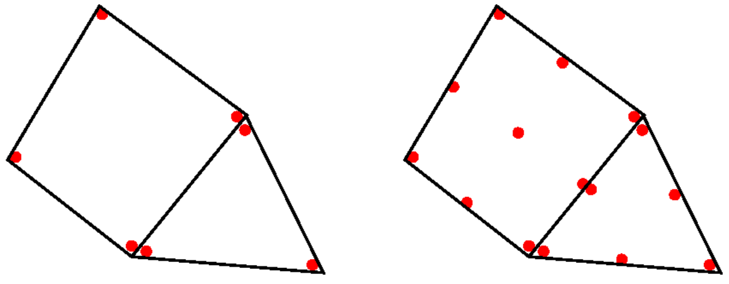
Nodal Discontinuous Galerkin Methods

The DG solutions in each element are represented using finite element shape functions:

$$\mathbf{U}_h = \sum_{j=1}^{N} \mathbf{U}_j(t) B_j(x),$$

where B_i: finite element shape functions.

$$\mathbf{M}_{N\times N} \frac{d}{dt}\begin{pmatrix} U_1 \\ U_2 \\ \bullet \\ \bullet \\ U_N \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ \bullet \\ \bullet \\ R_N \end{pmatrix}$$


Q1/P1 Q2/P2

The unknowns are the values of the conservative variables at the nodes.

The shape (trial, test) functions depend on the shape of elements.

Lagrange Basis Functions

- 1D
  - Introduce reference coordinate ξ over an element with nodes 1 and 2
    - $\zeta=(x-x_1)/(x_2-x_1)$
  - Linear basis functions over an element with nodes 1 and 2
    - $B_1=1-\zeta$, $B_2=\zeta$
  - Quadratic basis functions
    - $B_1 = (1-\zeta)(1-2\zeta)$, $B_2 = -\zeta(1-2\zeta)$, $B_3 = 4\zeta(1-\zeta)$
-

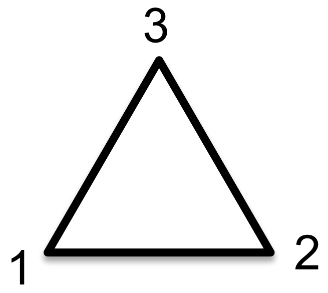
Finite Element Basis Functions for Triangles

- Introduce area (barycentric) coordinates ζ_i (i=1,2,3) for a point P(X) in a triangle with nodes 1, 2, and 3
  - ζ_1 = area of triangle P23/area of triangle 123
  - ζ_2 = area of triangle P31/area of triangle 123
  - ζ_3 = area of triangle P12/area of triangle 123

- Introduce reference coordinates (ξ, η)
  - $\xi = \zeta_2$, and $\eta = \zeta_3$
  - $\mathbf{X} = \zeta_i \mathbf{X}_i = (1-\xi-\eta)\mathbf{X}_1 + \xi \mathbf{X}_2 + \eta \mathbf{X}_3$

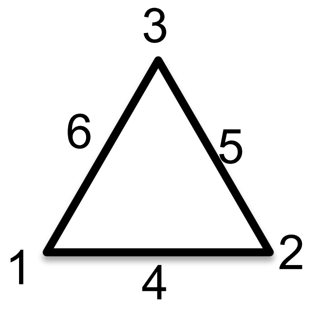
Shape Functions for Triangles

- Linear basis functions over a triangle with nodes 1, 2, and 3
  - $B_1 = \zeta_1 = 1-\xi-\eta$
  - $B_2 = \zeta_2 = \xi$
  - $B_3 = \zeta_3 = \eta$



Degree of freedom of linear triangle

- Quadratic basis functions
  - $B_1 = \zeta_1 (2\zeta_1-1) = (1-\xi-\eta)(1-2\xi-2\eta)$
  - $B_2 = \zeta_2 (2\zeta_2-1) = \xi(2\xi-1)$
  - $B_3 = \zeta_3 (2\zeta_3-1) = \eta(2\eta-1)$
  - $B_4 = 4\zeta_1 \zeta_2 = 4(1-\xi-\eta)\xi$
  - $B_5 = 4\zeta_2 \zeta_3 = 4\xi\eta$
  - $B_6 = 4\zeta_3 \zeta_1 = 4\eta (1-\xi-\eta)$



Degree of freedom of quadratic triangle

Shape Functions for Quads

- Bi-linear Quad
  - $B_1 = (1-\xi)(1-\eta)$
  - $B_2 = \xi(1-\eta)$
  - $B_3 = \xi\eta$
  - $B_4 = (1-\xi)\eta$

- Quadratic Serendipity Quads Bi-linear Quad
  - $B_1 = (1-\xi)(1-\eta)(1-2\xi-2\eta)$
  - $B_2 = -\xi(1-\eta)(1-2\xi+2\eta)$
  - $B_3 = -\xi(1-\eta)(3-2\xi-2\eta)$
  - $B_4 =-(1-\xi)\eta(1+2\xi-2\eta)$
  - $B_5 = 4\xi(1-\xi)(1-\eta)$
  - $B_6 = 4\xi\eta\ (1-\eta)$
  - $B_7 = 4(1-\xi)\xi\eta$
  - $B_8 = 4(1-\xi)(1-\eta)\eta$

Model Discontinuous Galerkin Methods

The DG solutions in each element are represented using model basis functions:

$$\mathbf{U}_h = \sum_{j=1}^{N} \mathbf{U}_j(t) B_j(x),$$

where B_i: model basis functions.

$$\mathbf{M}_{N\times N} \frac{d}{dt} \begin{pmatrix} U_1 \\ U_2 \\ \bullet \\ \bullet \\ U_N \end{pmatrix} = \begin{pmatrix} R_1 \\ R_2 \\ \bullet \\ \bullet \\ R_N \end{pmatrix}$$

The unknowns are the moments of the conservative variables in each elements.

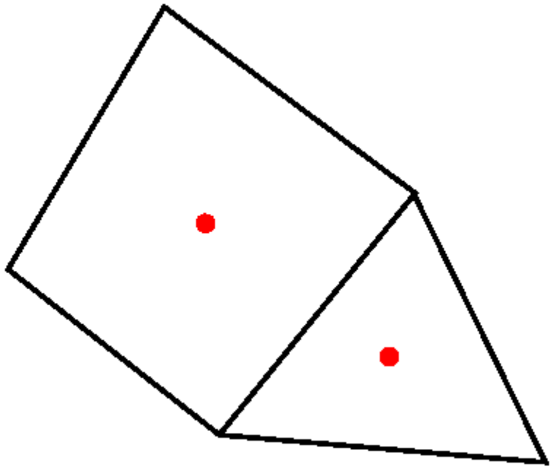
Legendre Basis Functions

- 1D Legendre Basis Function
  - Introduce reference coordinate ξ over an element with nodes 1 and 2
    - $\zeta=2(x-x_c)/(x_2-x_1)$ with $x_c= (x_2+x_1)/2$
  - Linear basis functions over an element with nodes 1 and 2
    - $B_1=1$, $B_2 = \zeta$
  - Quadratic basis functions
    - $B_1=1$, $B_2 = \zeta$, $B_3 = (3\zeta^2-1)/2$
- Bonnet's recursion formula: $(n+1)B_{n+2}(\zeta) = (2n+1)\,\zeta B_{n+1}(\zeta) - nB_n(\zeta)$
- Multi-dimensional Basis Function can be derived using tensor-product

Taylor Basis Functions

$$\mathbf{U}_h = \mathbf{U}_c(t) + \left.\frac{\partial \mathbf{U}(t)}{\partial x}\right|_c (\mathrm{x}-\mathrm{x}_c) + \left.\frac{\partial \mathbf{U}(t)}{\partial y}\right|_c (y-y_c) + \left.\frac{\partial^2 \mathbf{U}(t)}{\partial x^2}\right|_c \frac{(x-x_c)^2}{2} + \left.\frac{\partial^2 \mathbf{U}(t)}{\partial y^2}\right|_c \frac{(y-y_c)^2}{2} + \left.\frac{\partial^2 \mathbf{U}(t)}{\partial x \partial y}\right|_c (x-x_c)(y-y_c)$$

$$\mathbf{U}_h = \widetilde{\mathbf{U}} + \left.\frac{\partial \mathbf{U}}{\partial x}\right|_c (x-x_c) + \left.\frac{\partial \mathbf{U}}{\partial y}\right|_c (y-y_c)$$
$$+ \left.\frac{\partial^2 \mathbf{U}}{\partial x^2}\right|_c \left(\frac{(x-x_c)^2}{2} - \int_{\Omega_e} \frac{(x-x_c)^2}{2} d\Omega\right) + \left.\frac{\partial^2 \mathbf{U}}{\partial y^2}\right|_c \left(\frac{(y-y_c)^2}{2} - \int_{\Omega_e} \frac{(y-y_c)^2}{2} d\Omega\right)$$
$$+ \left.\frac{\partial^2 \mathbf{U}}{\partial x \partial y}\right|_c \left((x-x_c)(y-y_c) - \int_{\Omega_e} (x-x_c)(y-y_c) d\Omega\right)$$



The unknowns are the cell-averaged conservative variables and their derivatives at the center of the cells, regardless of element shapes.

Normalized Taylor basis functions

$$B_1 = 1 \qquad B_2 = \frac{\mathrm{x} - \mathrm{x}_\mathrm{c}}{\Delta x} \qquad B_3 = \frac{\mathrm{y} - \mathrm{y}_\mathrm{c}}{\Delta y} \qquad B_4 = \frac{(x - x_c)^2}{2\Delta x^2} - \frac{1}{\Omega_i}\int_{\Omega_i} \frac{(x - x_c)^2}{2\Delta x^2} d\Omega$$

$$B_5 = \frac{(y - y_c)^2}{2\Delta y^2} - \frac{1}{\Omega_i}\int_{\Omega_i} \frac{(y - y_c)^2}{2\Delta y^2} d\Omega \qquad B_6 = \frac{(x - x_c)(y - y_c)}{\Delta x \Delta y} - \frac{1}{\Omega_i}\int_{\Omega_i} \frac{(x - x_c)(y - y_c)}{\Delta x \Delta y} d\Omega$$

$$\mathbf{U}_h = \tilde{\mathbf{U}} B_1 + \mathbf{U}_x B_2 + \mathbf{U}_y B_3 + \mathbf{U}_{xx} B_4 + \mathbf{U}_{yy} B_5 + \mathbf{U}_{xy} B_6$$

$$\mathbf{U}_x = \left.\frac{\partial \mathbf{U}}{\partial x}\right|_c \Delta \mathrm{x}, \quad \mathbf{U}_y = \left.\frac{\partial \mathbf{U}}{\partial y}\right|_c \Delta \mathrm{y}, \quad \mathbf{U}_{xx} = \left.\frac{\partial^2 \mathbf{U}}{\partial x^2}\right|_c \Delta \mathrm{x}^2, \quad \mathbf{U}_{yy} = \left.\frac{\partial^2 \mathbf{U}}{\partial y^2}\right|_c \Delta \mathrm{y}^2, \quad \mathbf{U}_{xy} = \left.\frac{\partial^2 \mathbf{U}}{\partial x \partial y}\right|_c \Delta x \Delta \mathrm{y}$$

$$\Delta x = \frac{x_{\max} - x_{\min}}{2} \qquad \Delta y = \frac{y_{\max} - y_{\min}}{2}$$

Xmax, xmin, ymax, ymin are the maximum and minimum coordinates in the cell Ωi in x-, and y-directions, respectively.

Alleviate the stiffness of the system matrix for higher-order DG approximation

Features of Taylor Basis

- A finite volume code can be easily converted to a DG code.
- Same approximate polynomial solution for any shapes of elements:
  - Can be easily extended and implemented on arbitrary meshes.
- Cell-averaged variables and their derivatives are handily available:
  - Make implementation of WENO reconstruction easy and efficient
- Hierarchic basis
  - Make implementation of *p*-multigrid methods and *p*-refinement easy and efficient

DG(P$_0$) approximation

$$\mathbf{U}_h = \mathbf{U}(t) \qquad \mathrm{N}=1, \quad \mathrm{B}_1=1,$$

$$\frac{d}{dt}\int_{\Omega_e} \mathbf{U}_h B_i d\Omega + \int_{\Gamma_e} \mathbf{F}_k(\mathbf{U}_h)\mathbf{n}_k B_i d\Gamma - \int_{\Omega_e} \mathbf{F}_k(\mathbf{U}_h)\frac{\partial B_i}{\partial x_k} d\Omega = 0, \quad 1 \le i \le \mathrm{N}$$

$$\downarrow$$

$$\frac{d}{dt}\int_{\Omega_e} \mathbf{U}(t) d\Omega + \int_{\Gamma_e} \mathbf{F}_k \mathbf{n}_k d\Gamma = 0$$

- The classical first-order cell-centered finite volume scheme exactly corresponds to the DG(P$_0$) method.

- DG methods can be regarded as a natural generalization of finite volume methods to higher order methods.

- By simply increasing the degree o of polynomials DG methods of corresponding higher-orders are obtained.

DG(P_1) approximation

N=3,
$$\mathbf{U}_h = \widetilde{\mathrm{U}} B_1 + \mathbf{U}_x B_2 + \mathbf{U}_y B_3$$

$$\frac{d}{dt}\int_{\Omega_e} \widetilde{\mathrm{U}}\, d\Omega + \int_{\Gamma_e} \mathbf{F}_k(\mathrm{U}_h)\,\mathbf{n}_k\, d\Gamma = 0, \quad i = 1$$

$$\begin{pmatrix} \int B_2 B_2 d\Omega & \int B_2 B_3 d\Omega \\ \int B_3 B_2 d\Omega & \int B_3 B_3 d\Omega \end{pmatrix} \begin{pmatrix} \dfrac{d\mathbf{U}_x}{dt} \\ \dfrac{d\mathbf{U}_y}{dt} \end{pmatrix} + \begin{pmatrix} \int_{\Gamma_e} F_k n_k B_2 d\Gamma - \int_{\Omega_e} F_k \dfrac{\partial B_2}{\partial x_k} d\Omega \\ \int_{\Gamma_e} F_k n_k B_3 d\Gamma - \int_{\Omega_e} F_k \dfrac{\partial B_3}{\partial y_k} d\Omega \end{pmatrix} = 0$$

DG(P$_2$) approximation

N=6, $$\mathbf{U}_h = \tilde{\mathbf{U}}B_1 + \mathbf{U}_x B_2 + \mathbf{U}_y B_3 + \mathbf{U}_{xx} B_4 + \mathbf{U}_{yy} B_5 + \mathbf{U}_{xy} B_6$$

$$\frac{d}{dt}\int_{\Omega_e} \tilde{\mathbf{U}}\, d\Omega + \int_{\Gamma_e} \mathbf{F}_k(\mathbf{U}_h)\, \mathbf{n}_k\, d\Gamma = 0, \qquad i = 1$$

$$M_{5x5}\begin{pmatrix} \frac{d\mathbf{U}_x}{dt} \\ \frac{d\mathbf{U}_y}{dt} \\ \frac{d\mathbf{U}_{xx}}{dt} \\ \frac{d\mathbf{U}_{yy}}{dt} \\ \frac{d\mathbf{U}_{xy}}{dt} \end{pmatrix} + \begin{pmatrix} \int_{\Gamma_e} F_k n_k B_2 d\Gamma - \int_{\Omega_e} F_k \frac{\partial B_2}{\partial x_k} d\Omega \\ \int_{\Gamma_e} F_k n_k B_3 d\Gamma - \int_{\Omega_e} F_k \frac{\partial B_3}{\partial x_k} d\Omega \\ \int_{\Gamma_e} F_k n_k B_4 d\Gamma - \int_{\Omega_e} F_k \frac{\partial B_4}{\partial x_k} d\Omega \\ \int_{\Gamma_e} F_k n_k B_5 d\Gamma - \int_{\Omega_e} F_k \frac{\partial B_5}{\partial x_k} d\Omega \\ \int_{\Gamma_e} F_k n_k B_6 d\Gamma - \int_{\Omega_e} F_k \frac{\partial B_6}{\partial x_k} d\Omega \end{pmatrix} = 0$$

Similarity and difference between FV and DG

- The discretized governing equations for cell-averaged variables and the assumption of a polynomial solution on each cell are exactly the same for both FV and DG methods.

- The only difference between them is the way how to obtain the polynomial solution, i.e., how to compute the derivatives.

DG Methods

- The derivatives are computed in a manner similar to the mean variables, which is unique, compact, rigorous, and elegant mathematically.

- The higher order DG methods can be easily constructed by simply increasing the degree *p* of the polynomials locally, in contrast to the finite volume methods which use the extended stencils to achieve higher order of accuracy.

Reconstruction Methods

- The polynomial solutions are reconstructed from cell-averaged variables of neighbouring cells. The multi-dimensional reconstruction schemes based on the extension of 1D MUSCL approach have two serious flaws:
  - Uncertainty and arbitrariness in choosing the stencils and the methods to compute the derivatives;
    - Formal second order accuracy is hardly obtained in practice !!!
  - Extended stencils required for higher-order (>2nd) reconstruction.
    - The finite volume methods are not practical at higher order and have remained second-order !!!

Observation

Reconstruction and DG methods can be viewed as two ways to obtain higher accuracy of the first order finite volume methods

	DG	Reconstruction
Efficiency (computing costs and storage requirements)	Bad	Good
Robustness	Good	Bad
Accuracy	Good	Bad
Tolerance to grid irregularity	Good	Bad

In-cell Recovery/Reconstruction

- Objective:
  - Combine the advantages of both reconstruction and DG methods in order to improve the efficiency of the DG methods
    - PnPm schemes (Dumbser et al, 2008)
    - Reconstruction-based DG (Luo et al, 2009)
    - Hybrid DG/FV schemes (Zhang et al, 2010)
- How ?
  - Recover/reconstruct a higher-order polynomial solution from the underlying discontinuous DG polynomial solution

Background

To reduce high computing costs of the DG methods, Reconstructed DG (RDG(P_nP_m)) schemes were introduced by Dumbser et al.

- P_n indicates that a piecewise polynomial of degree of n is used to represent a DG solution.
- P_m represents a reconstructed polynomial solution of degree of m ($m \geq n$) that is used to compute the fluxes and source terms.
- Provide a unified formulation for both finite volume and DG methods, and contain both classical finite volume and standard DG methods as two special cases of RDG(P_nP_m) schemes.

Background

Classification of the RDG(PnPm) Schemes

Order of Accuracy	Schemes
O(1)	RDG(P_0P_0) (DG(P_0))
O(2)	RDG(P_0P_1) RDG(P_1P_1) (DG(P_1))
O(3)	RDG(P_0P_2) RDG(P_1P_2) RDG(P_2P_2)
O(4)	RDG(P_0P_3) RDG(P_1P_3) RDG(P_2P_3) RDG(P_3P_3)
	:
O(M+1)	RDG(P_0P_m) FV … RDG(P_nP_m) New Class … RDG(P_mP_m) DG

Reconstructed Discontinuous Galerkine Method: RDG(PnPm)

$$\frac{d}{dt}\int_{\Omega_e} \mathbf{U}_{P_n} B_i d\Omega + \int_{\Gamma_e} \mathbf{F}_k(\mathbf{U}^{R}_{P_m})\mathbf{n}_k B_i d\Gamma - \int_{\Omega_e} \mathbf{F}_k(\mathbf{U}^{R}_{P_m})\frac{\partial B_i}{\partial x_k} d\Omega = \int_{\Gamma_e} \mathbf{G}_k(\mathbf{U}^{R}_{P_m})\mathbf{n}_k B_i d\Gamma - \int_{\Omega_e} \mathbf{G}_k(\mathbf{U}^{R}_{P_m})\frac{\partial B_i}{\partial x_k} d\Omega, \quad 1 \le i \le N$$

$\mathbf{U}^{R}_{P_m}$: reconstructed polynomial solution of degree Pm

$B_i(x)$: basis functions of polynomials of degree Pn , $1 \le i \le N$

N: dimension of the polynomial space P_n

$\uparrow$

Reconstructed Discontinuous Galerkin method RDG(PnPm) : $O(h^{m+1})$

Background

- Observation: The construction of an accurate and efficient reconstruction operator is crucial to the success of the RDG(PnPm) schemes.

Reconstructed DG method RDG(P_1P_2)

- From a linear polynomial DG solution in any cell i

$$\mathbf{U}_i = \widetilde{\mathbf{U}}_i B_1 + \mathbf{U}_{xi} B_2 + \mathbf{U}_{yi} B_3$$

- Reconstruct a quadratic polynomial solution U^R

$$\mathbf{U}_i^R = \widetilde{\mathbf{U}}_i^R B_1 + \mathbf{U}_{xi}^R B_2 + \mathbf{U}_{yi}^R B_3 + \mathbf{U}_{xxi}^R B_4 + \mathbf{U}_{yyi}^R B_5 + \mathbf{U}_{xyi}^R B_6$$

- Six degrees of freedom

Requirements For Reconstruction

- Conservation
- Compactness
  - Maintain compactness of the underlying DG method
  - Necessary for unstructured arbitrary grids
  - Stencils involve only Von Neumann neighborhood (face-neighboring cells)

Reconstruction

- Requiring conservation and reconstructed first derivatives equal to the ones of the underlying DG solution leads

$$\widetilde{\mathbf{U}}_i^R = \widetilde{\mathbf{U}}_i^R \qquad \mathbf{U}_{xi}^R = \mathbf{U}_{xi} \qquad \mathbf{U}_{yi}^R = \mathbf{U}_{yi}$$

due to the judicious choice of Taylor-basis in the DG formulation

- Three second derivatives only need to be reconstructed.

1. Least-squares Recovery (P_1P_2(rc)) (Dumbser et al.)

- This is achieved using a so-called in-cell recovery where recovered equations are obtained using a L_2 projection for each face-neighboring cell j, i.e.,

$$\int_{\Omega_j} \mathbf{U}_i^R B_k^j \, d\Omega = \int_{\Omega_j} \mathbf{U}_j \; B_k^j \, d\Omega, \qquad k = 1,\ldots,3$$

- The over-determined system of linear equations (9x3 for a triangular cell and 12x3 for a quadrilateral cell) is solved using a least-squares method.

(weak interpolation)

P_1P_2(rc) (Dumbser et al)

- 2-exact reconstruction
  - Able to reconstruct a quadratic polynomial exactly.
- Complex and expensive
  - Require to compute the integral on both left and right-side of the recovered equations.
- Problematic
  - For a boundary cell, the number of the face neighbouring cells might not be enough to recover a polynomial solution of a desired order.
  - Use extended one-side stencils -> destroy the compactness.

2. Least-squares Reconstruction (P_1P_2(RC)) (Luo et al)

- The remaining three degrees of freedom can be determined by requiring that the reconstructed solution and its first derivatives are equal to the underlying DG solution and its first derivatives for all the face neighboring cells.

$$\mathbf{U}_j = \widetilde{\mathbf{U}}_i + \mathbf{U}_{xi} B_2^j + \mathbf{U}_{yi} B_3^j + \mathbf{U}_{xxi}^R B_4^j + \mathbf{U}_{yyi}^R B_5^j + \mathbf{U}_{xyi}^R B_6^j$$

$$\left.\frac{\partial U}{\partial x}\right|_j = U_{xi}\frac{1}{\Delta x_i} + U_{xxi}^R \frac{B_2^j}{\Delta x_i} + U_{xyi}^R \frac{B_3^j}{\Delta x_i}$$

$$\left.\frac{\partial U}{\partial y}\right|_j = U_{yi}\frac{1}{\Delta y_i} + U_{yyi}^R \frac{B_3^j}{\Delta y_i} + U_{xyi}^R \frac{B_2^j}{\Delta y_i}$$

Strong interpolation

Least-Squares Reconstruction

- Similar equations can be written for all face-neighboring cells, which leads to a non-square matrix.

- The number of face-neighboring cells for a triangular and quadrilateral cell is 3 and 4, respectively. As a result, the size of resulting non-square matrix is 9x3, and 12x3, respectively.

- This over-determined linear system of 9 or 12 equations for 3 unknowns can be solved in the least-squares sense.

P_1P_2(RC) method

- Simple and straightforward
- The boundary conditions are used to obtain the reconstructed equations for the boundary cells, thus ensuring existence of an over-determined system.
- 2-exact reconstruction

3. Green-Gauss Reconstruction (P_1P_2(GG)) (Zhang et al)

- Green-Gauss reconstruction is widely used to reconstruct a gradient from the cell-averaged values in the finite volume methods.

- Similarly, the second derivatives in a cell *i* can be reconstructed from the first derivatives using Green's theorem as follows,

$$\int_{\Omega_i} \left.\frac{\partial^2 U}{\partial x^2}\right|_i d\Omega = \Omega_i \left.\frac{\partial^2 U}{\partial x^2}\right|_i = \int_{\Gamma_i} \frac{\partial U}{\partial x} n_x d\Gamma$$

Green-Gauss Reconstruction (P_1P_2(GG)) (Zhang et al)

- Simple, efficient. and robust
  - No need to solve a least-squares problem.
- Less accurate
  - Only use the information on first derivatives.
- Not 2-exact reconstruction
  - Cannot reconstruct a quadratic polynomial exactly

Cost Analysis (Tetrahedral Grids)

	RDG(P_0P_1)	RDG(P_1P_1)	RDG(P_1P_2)	RDG(P_2P_2)
Number of quadrature points for boundary integrals	1	3	4	7
Number of quadrature points for domain integrals	0	4	5	11
Reconstruction	Yes	No	Yes	No
Order of Accuracy	$O(h^2)$	$O(h^2)$	$O(h^3)$	$O(h^3)$
Storage for Implicit Diagonal Matrix	25 words Per element	400	400	2500

Cost Analysis (Hexahedral Grid)

Spatial method	RDG(P1P1))	RDG(P1P2)	RDG(P2P2)
Nr. of quadrature points for boundary integrals	4	4	9
Nr. of quadrature points for domain integrals	8	8	27
Reconstruction	NO	YES	NO
Order of spatial accuracy	$O(h^2)$	$O(h^3)$	$O(h^3)$
Storage for the implicit diagonal matrix per element	400 words	400 words	2500 words

The memory requirement for RDG(P_1P_2) is much smaller than DG(P_2).

Storage Consideration

Consider the memory requirements for storing only a block diagonal matrix

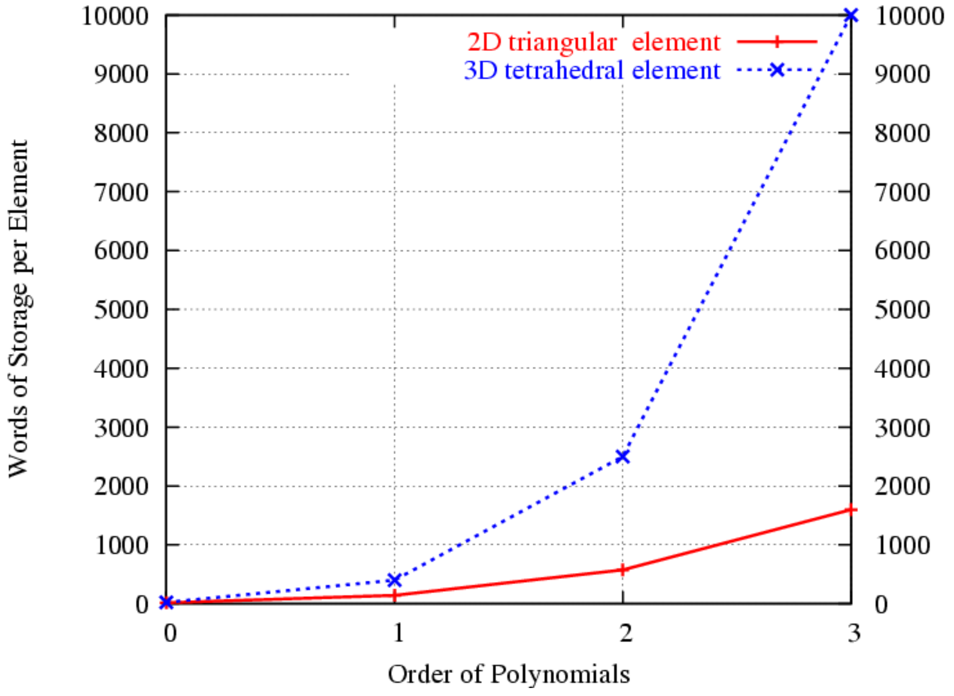
D(ndegr^2,neqns^2,nelem)

ndegr: degrees of freedom for the polynomial

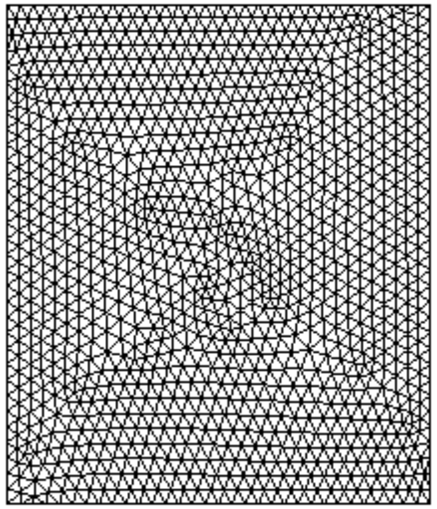
neqns: number of unknowns variables

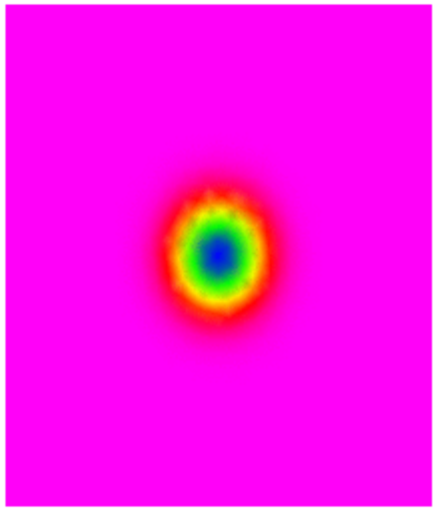
nelem: number of elements

Storage requirements for Implicit DG method are very demanding, especially for higher-order methods !!!



Example 1. Convection of an isentropic vortex

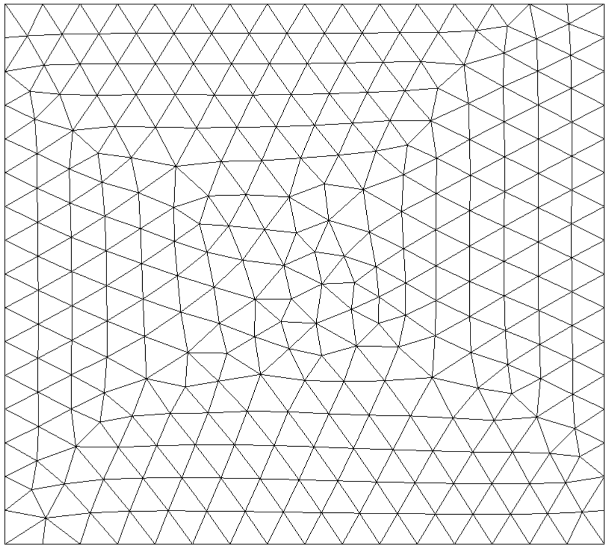

Time =0 Nelem =2216 Npoin =1173

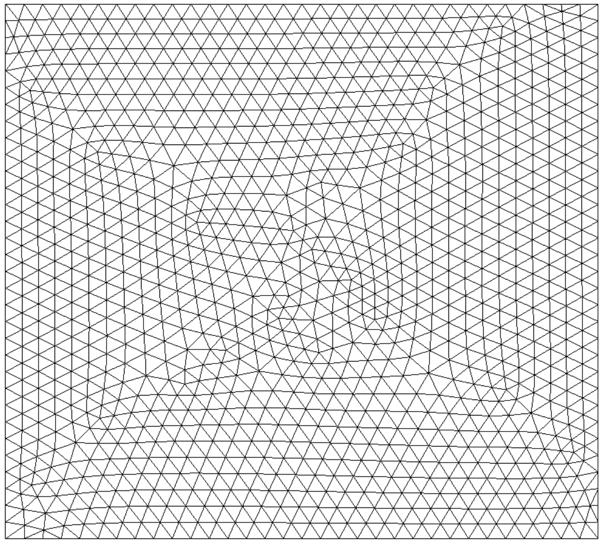

Time =0.000000 Min =0.490425 Max =1.000000

Example 1. isentropic vortex convection

Sequences of three successively globally refined meshes

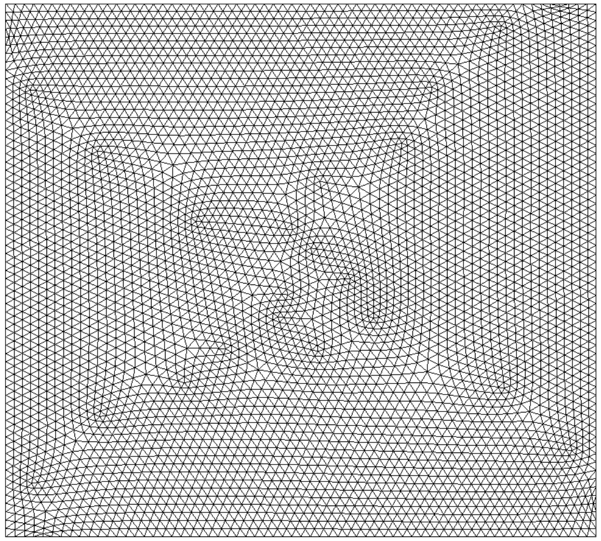
16x16



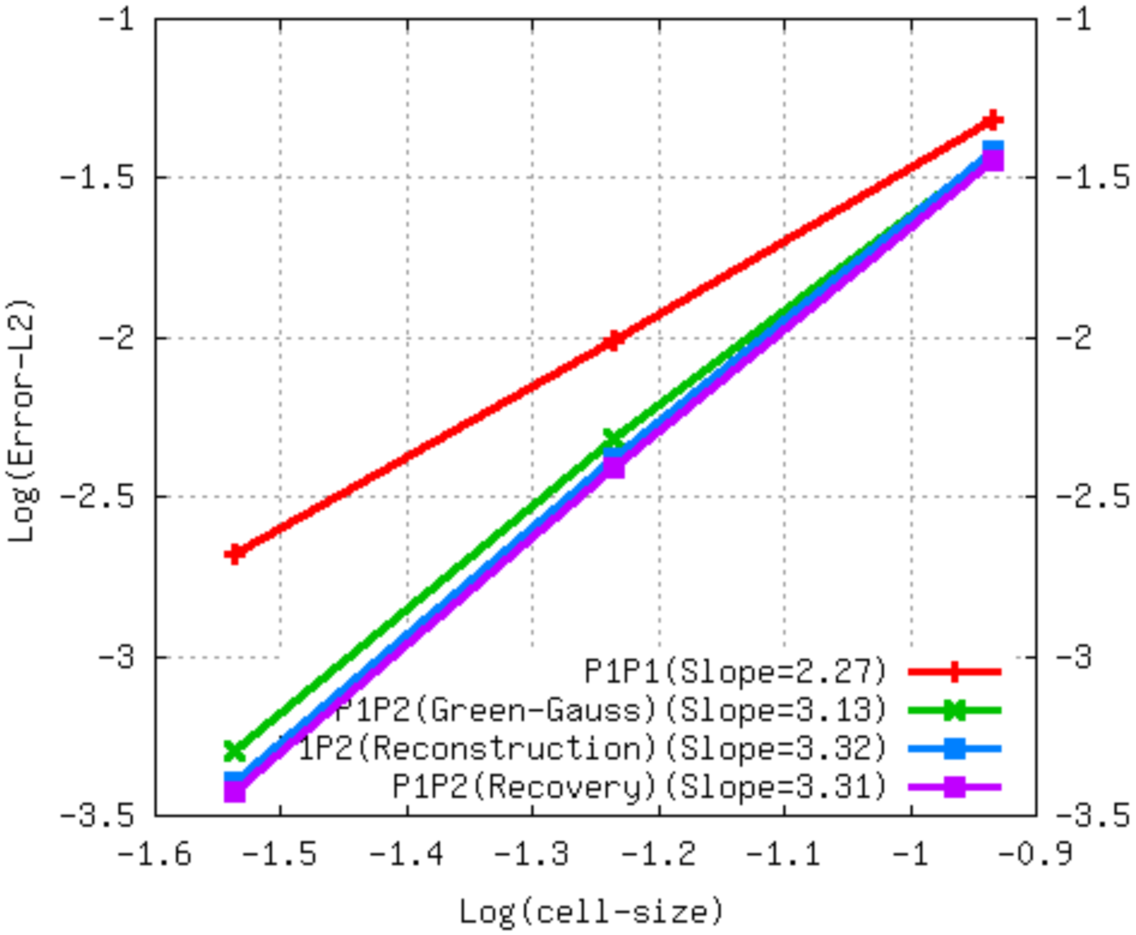


32x32

64x64



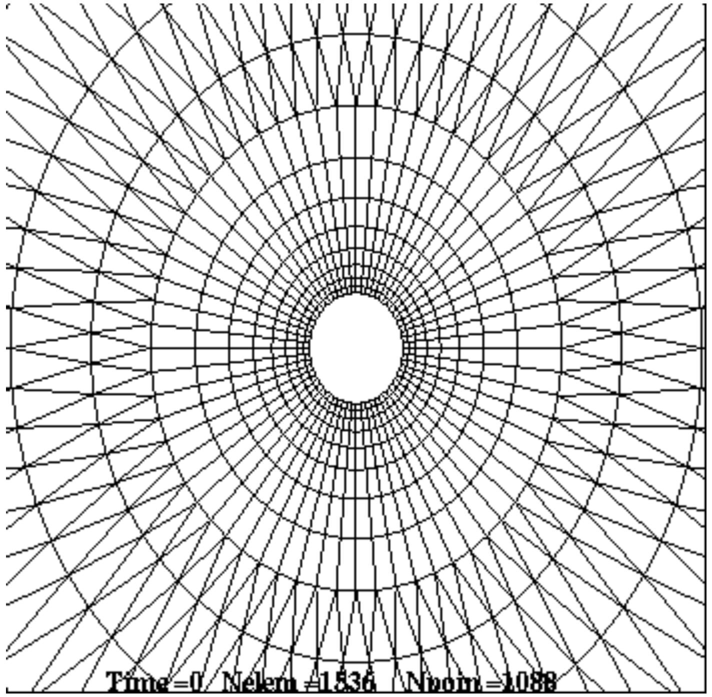
Solution Accuracy for different RDG methods

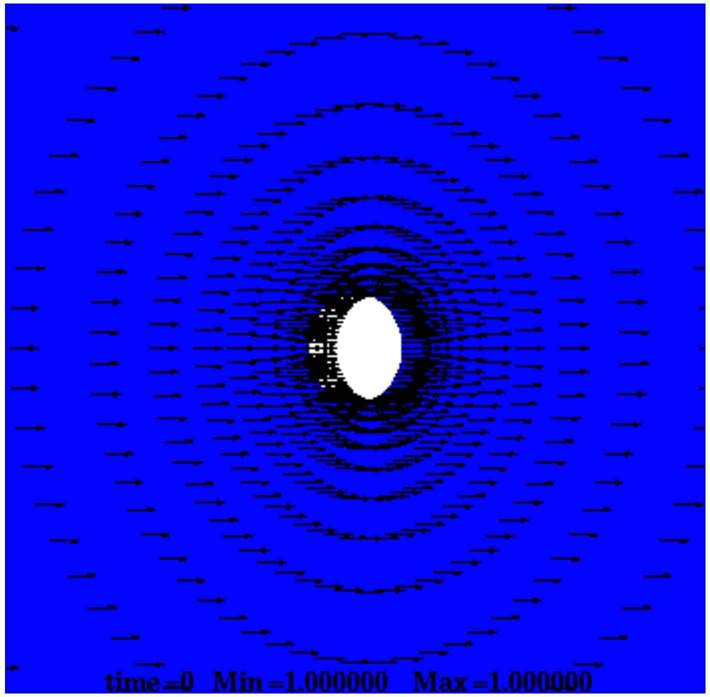


Example 2. Subsonic inviscid flow past a circular cylinder (M=0.38)

Assess the order of accuracy of the reconstructed discontinuous Galerkin methods.

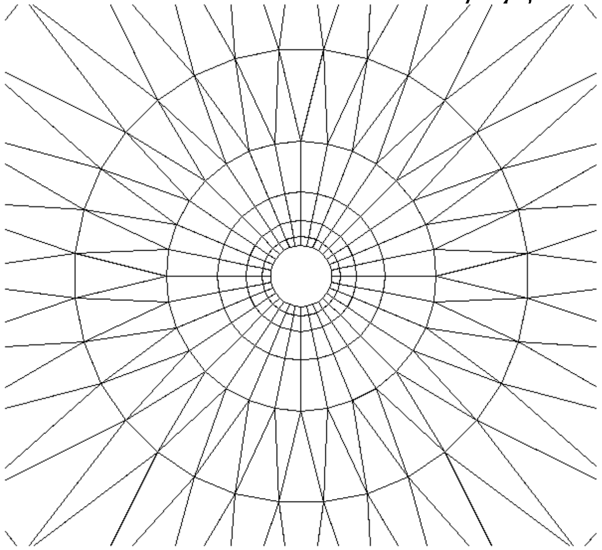
Entropy production is served as the error measurement.



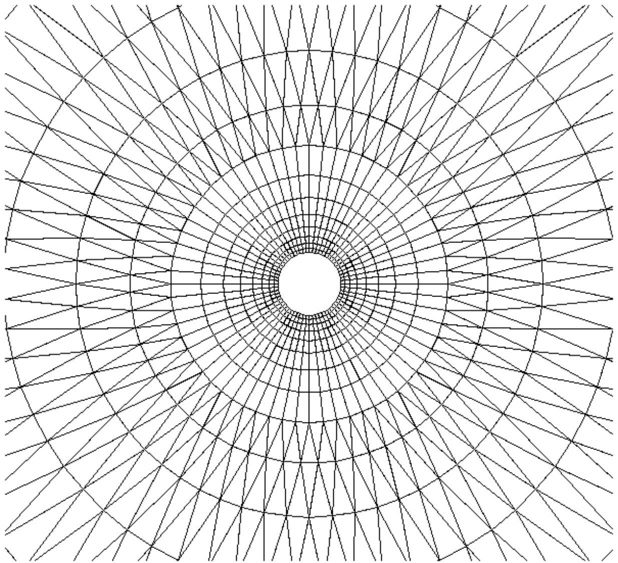


Example 2. Subsonic flow past a cylinder

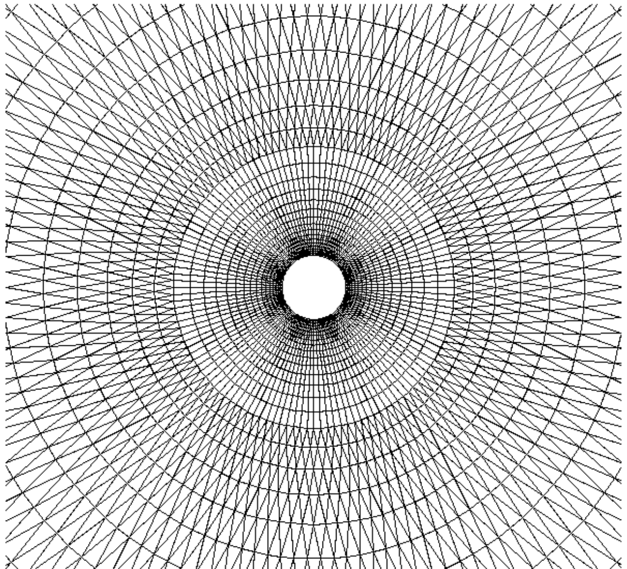
Sequences of three successively globally refined meshes



32x9



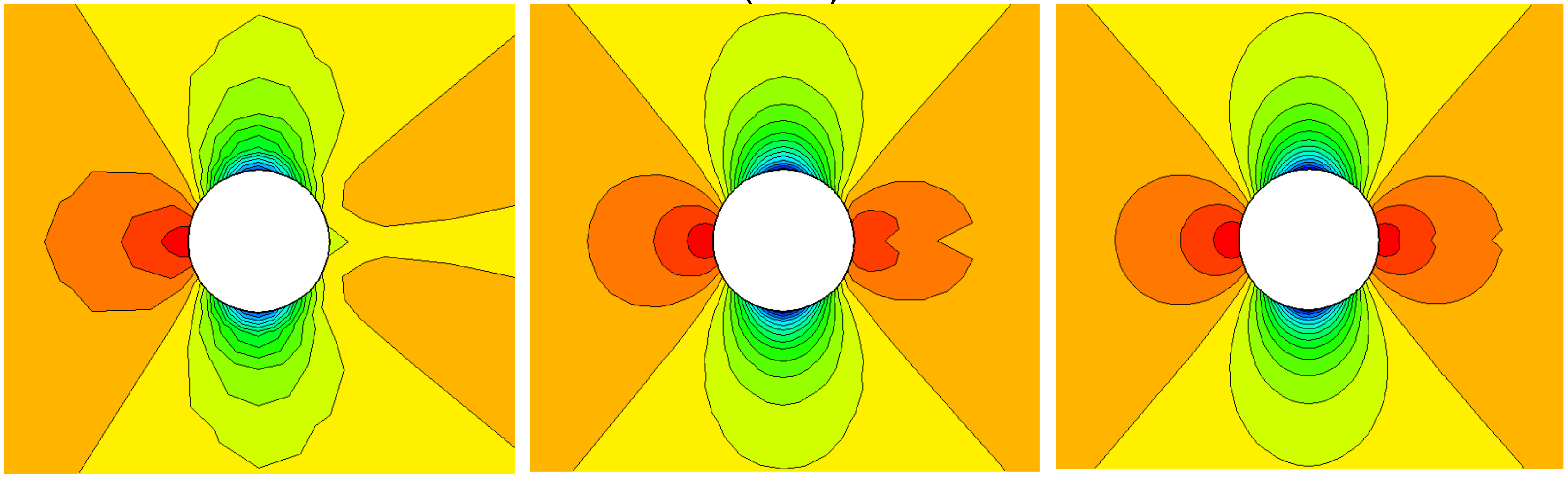
64x17



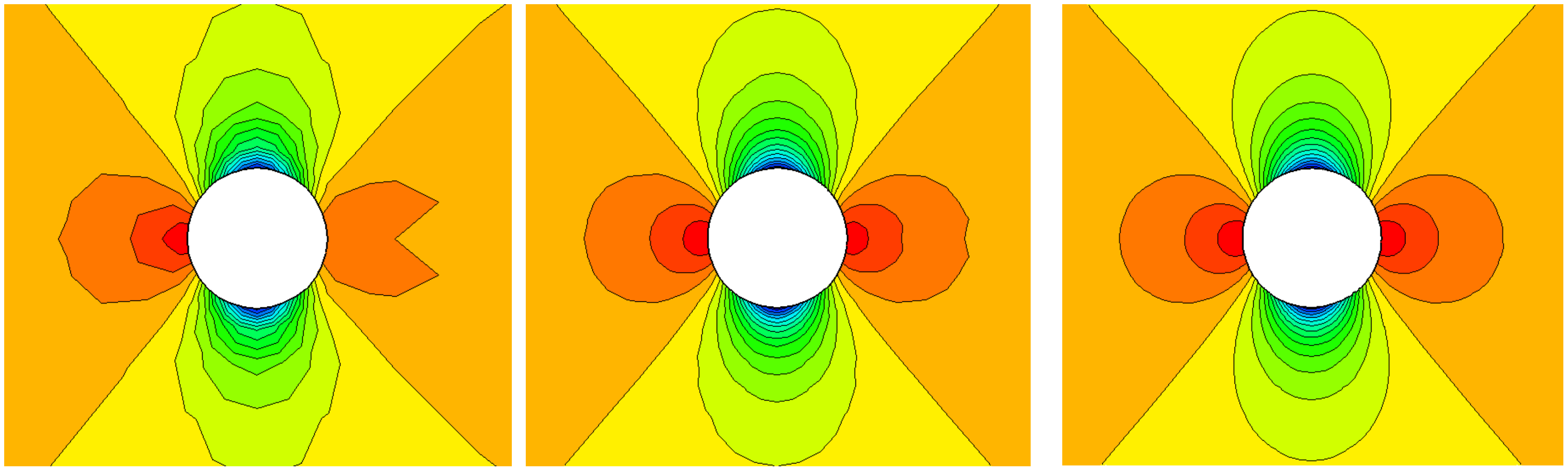
128x33

Computed density contours

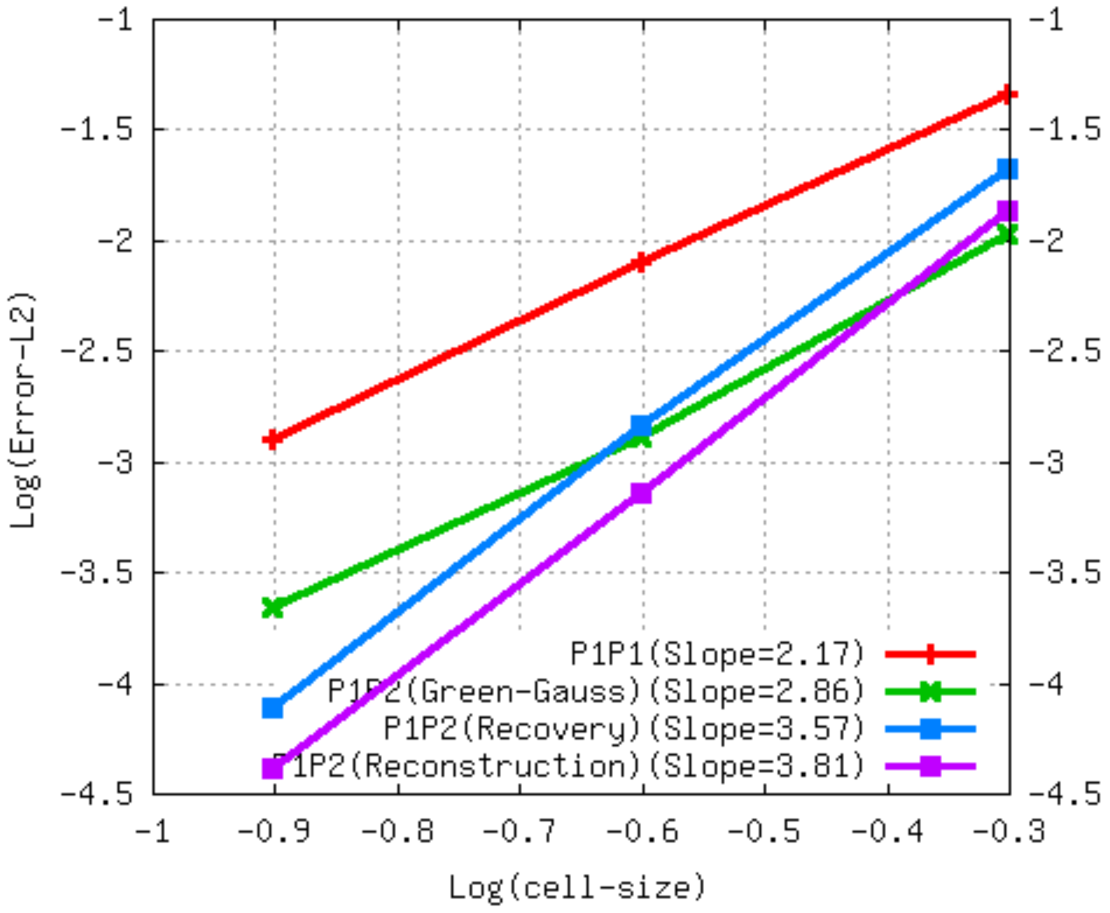
DG(P1)



P1P2(LS)



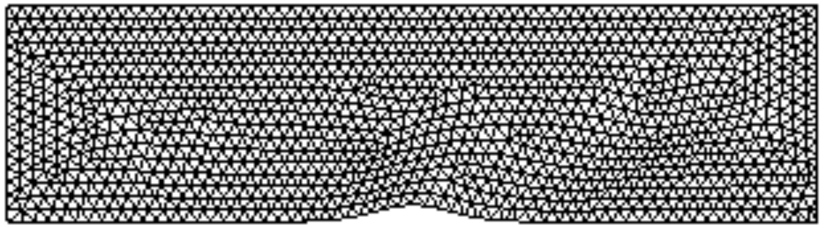
Solution Accuracy for different RDG methods



Example 3. Subsonic inviscid flow (M=0.5) through a channel with a smooth bump

Assess the order of accuracy of the reconstructed discontinuous Galerkin methods for internal flows

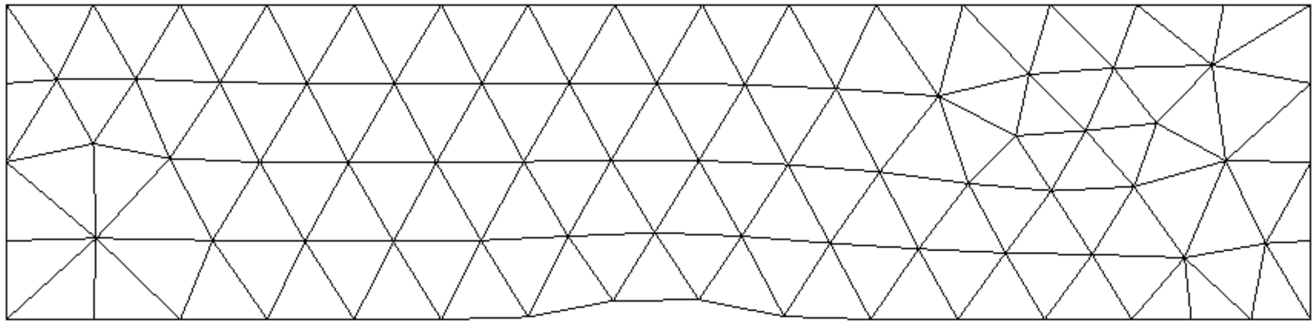
Entropy production is served as the error measurement.


Time =0 Nelem =2135 Npoin =1145

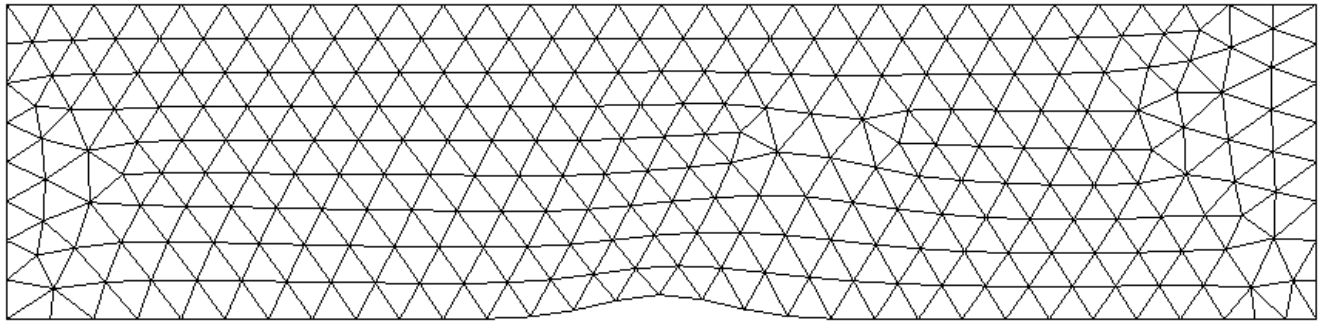
Time =0.000000 Min =1.000000 Max =1.000000

Example 3. Subsonic flow in a channel

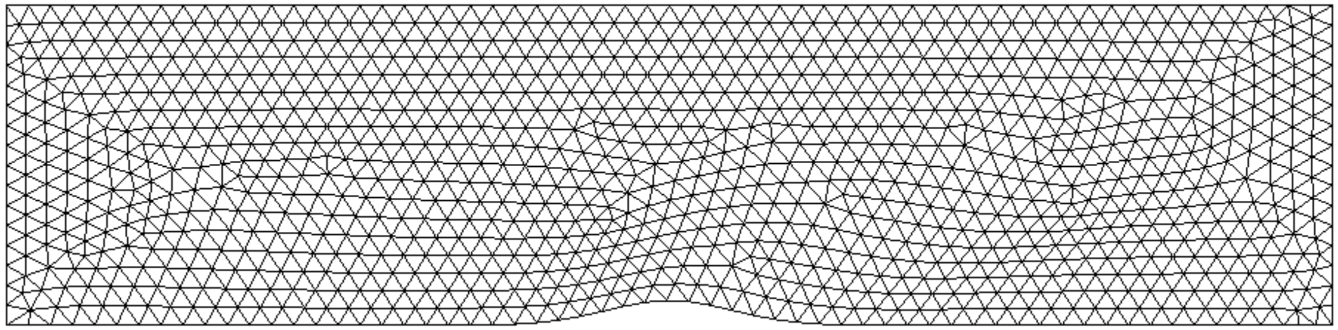
Sequences of three successively globally refined meshes



127 cells

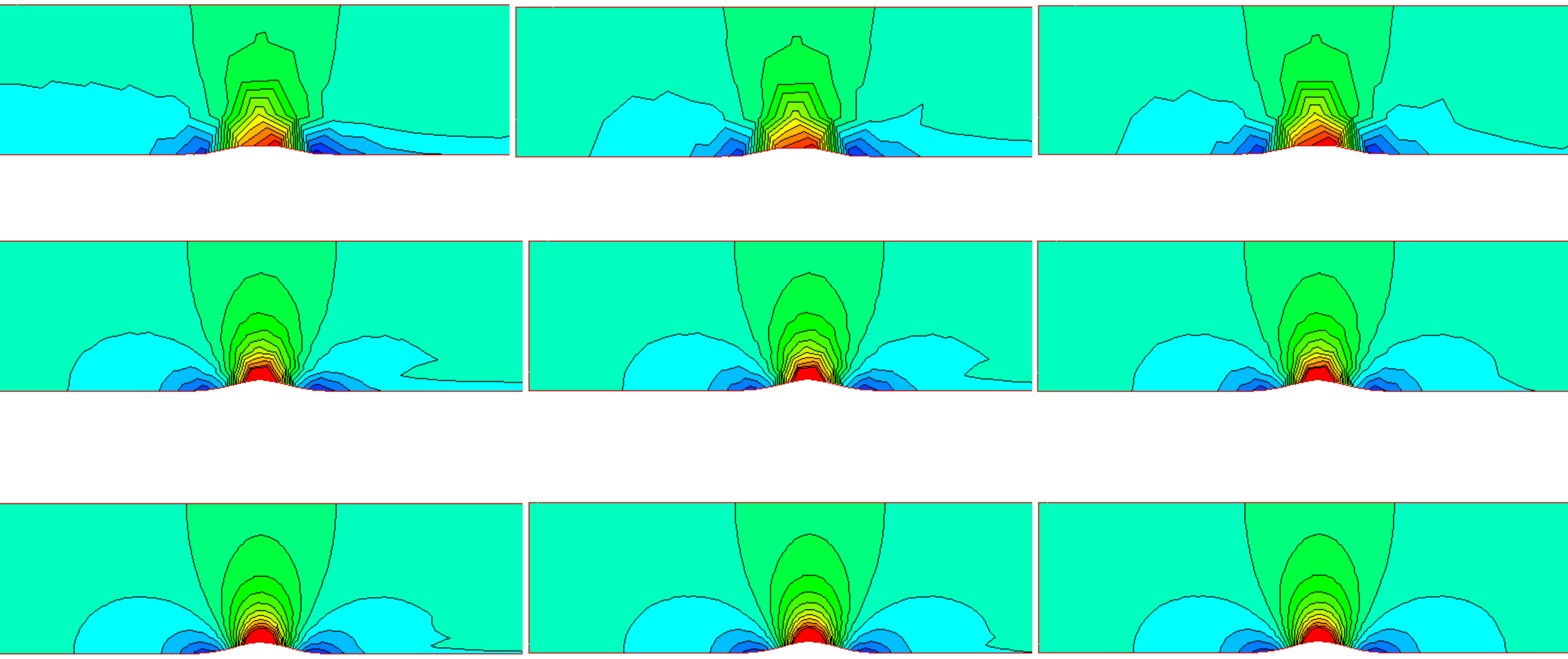


528 cells



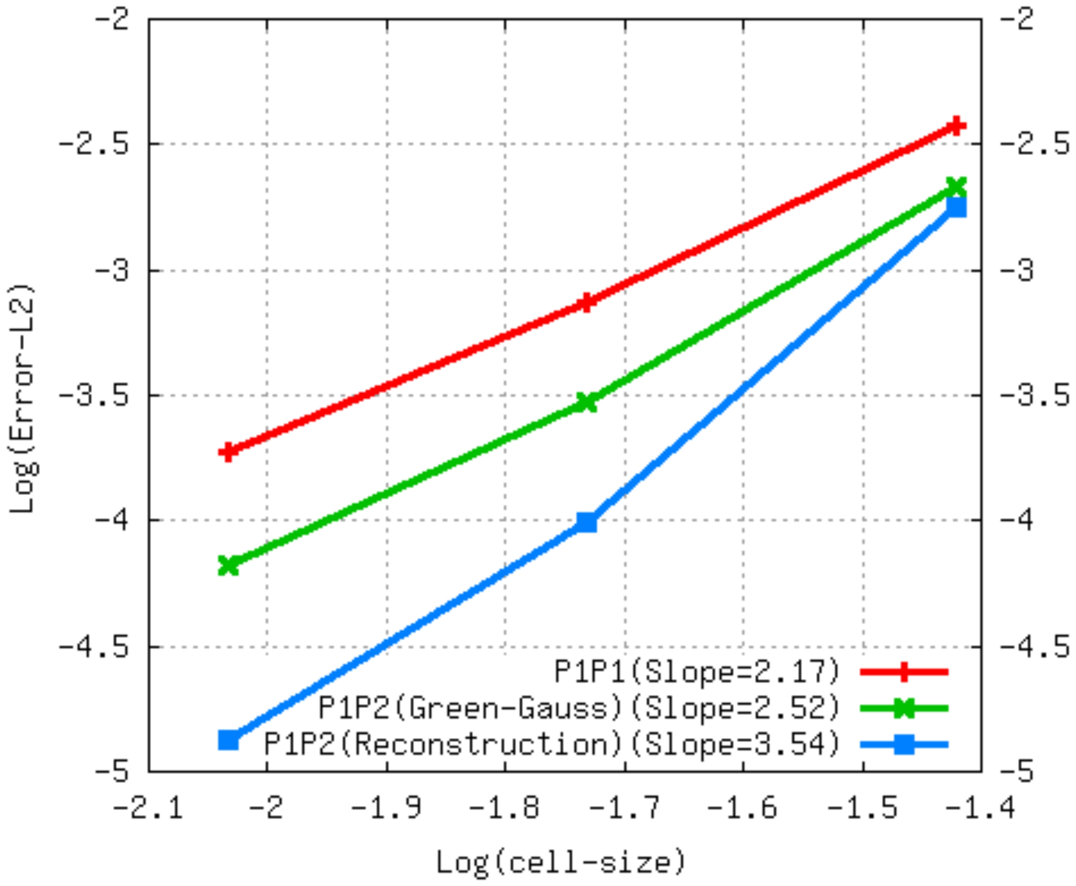
2,032 cells

Computed velocity contours



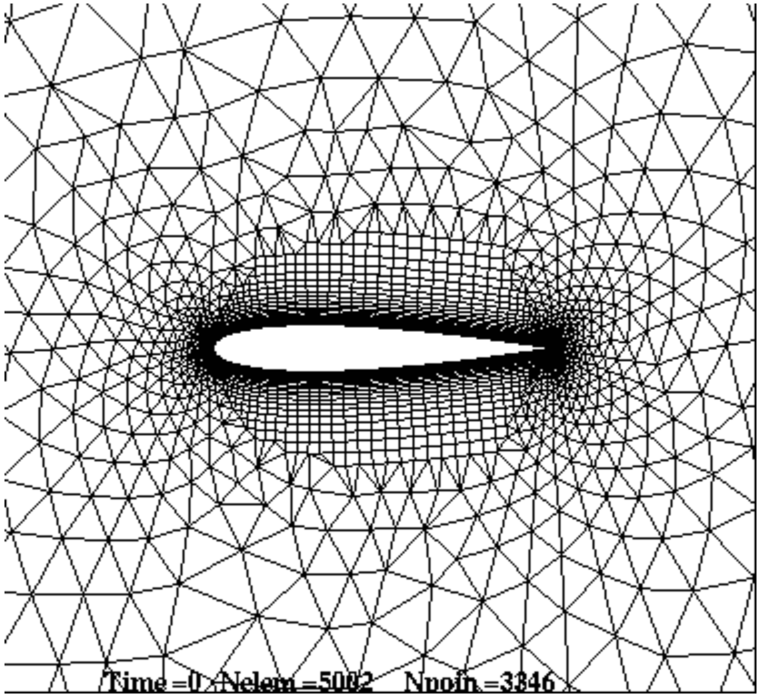
DG(P1) P1P2(GG) P1P2(LS)

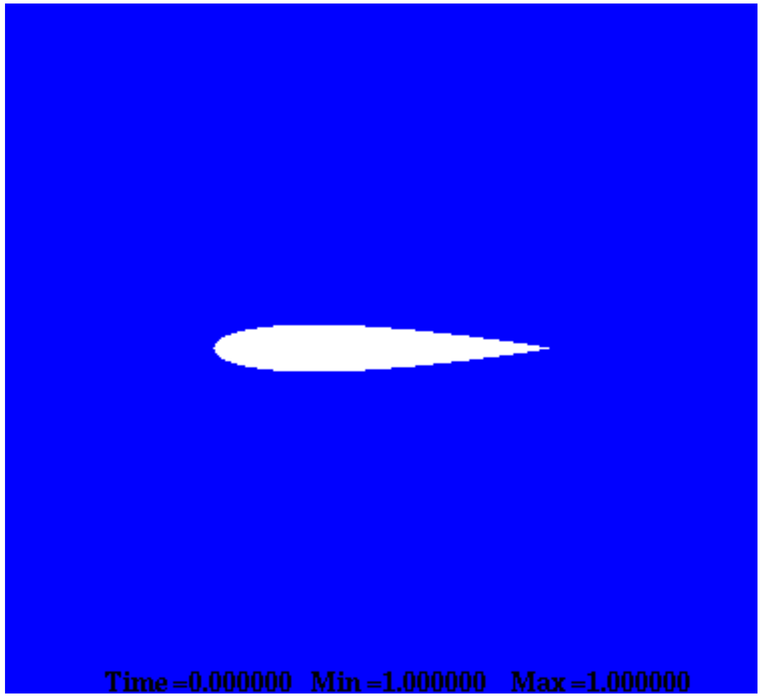
Solution Accuracy for different RDG methods



Example 4. Subsonic inviscid flow past an NACA0012 airfoil (M_∞=0.63, α=2°)

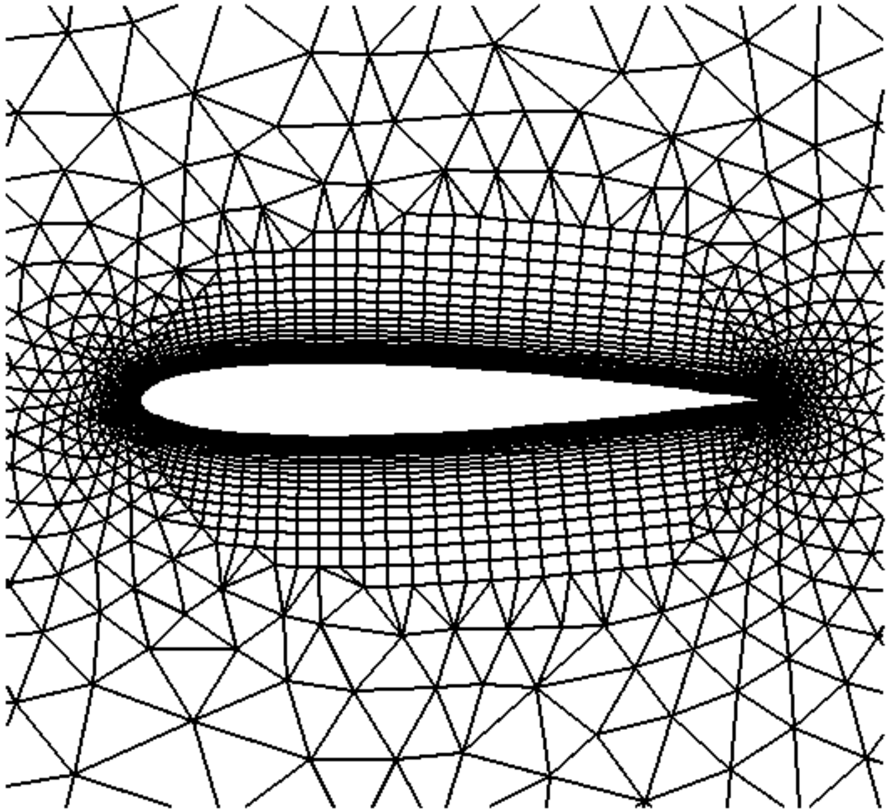
This test case is designed to assess the accuracy and robustness of the reconstructed discontinuous Galerkin methods for inviscid solutions on viscous type grids.





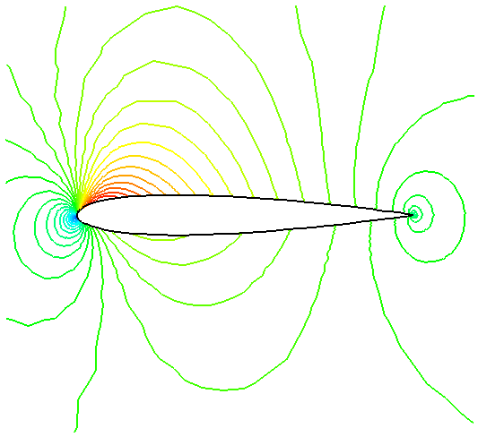
Example 4. Subsonic Flow past a NACA 0012 airfoil

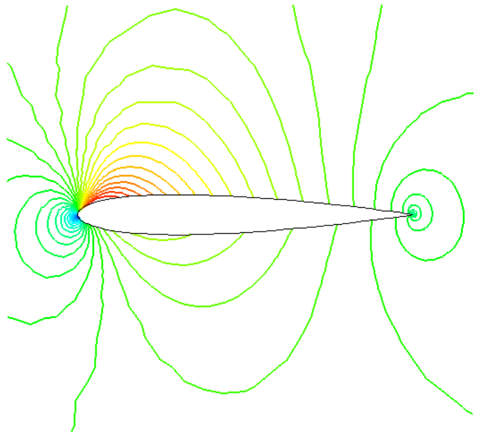
Nquad = 1,533, ntria=3,469, nbfac=157

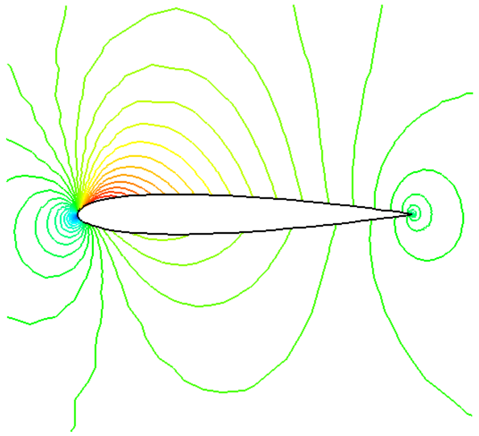


Subsonic flow past a NACA0012 Airfoil M_∞=0.63, α=2°

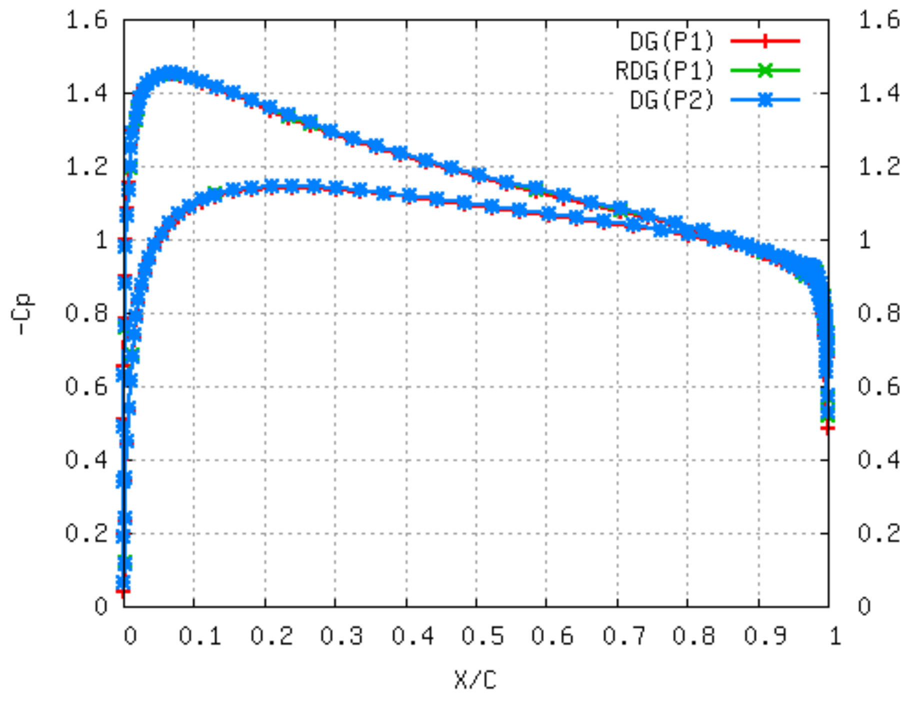
Computed Mach Number Contours

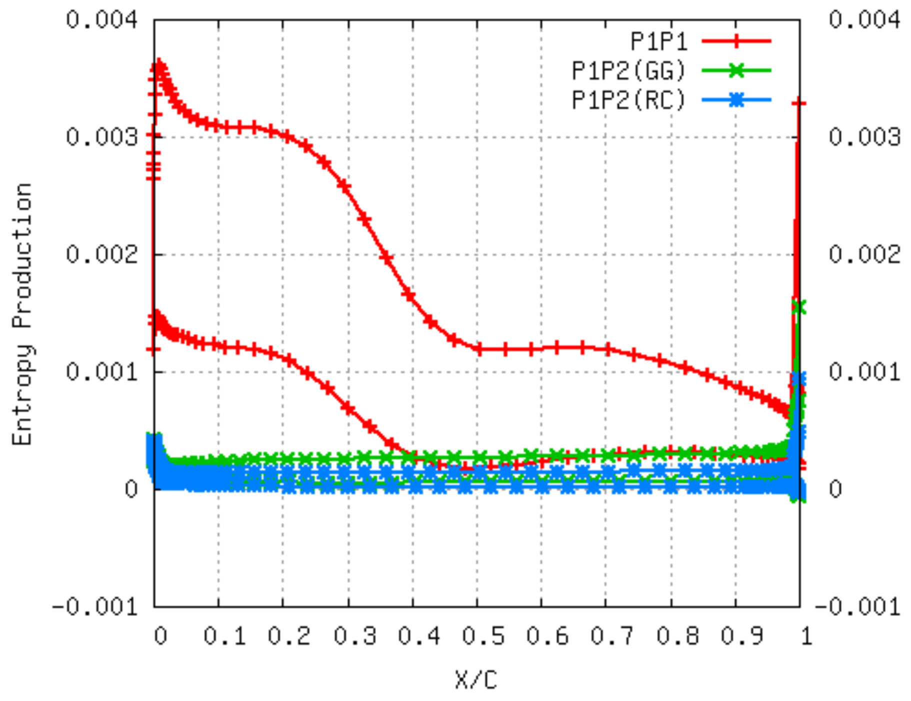

DG(P1)


P1P2(GG)


P1P2(LS)

Subsonic flow past a NACA0012 airfoil M_∞=0.63, α=2°





Comparison of computed pressure coefficient (left) and entropy production (right) distributions on the surface of airfoil obtained by the RDG methods

Concluding Remarks

- A class of RDG(P_1P_2) methods has been presented for solving the compressible flow problems.

- All three RDG methods are able to deliver the desired third order of accuracy and can significantly improve the accuracy of the underlying second-order method.

- The least-squares RDG provides the best performance in terms of accuracy, efficiency, and robustness.

Observation

- The extension of the RDG(P_1P_2) method to tetrahedral grids was unsuccessful.

- The RDG method suffers from linear instability, similar to RDG(P_0P_1)
  - Instability occurs even for linear equations and in smooth flows.
  - Reconstruction stencils only involve von Neumann neighborhood, i.e., adjacent face-neighboring cells

- To maintain the linear stability
  - Augment stencils in the reconstruction
    $\rightarrow$ Destroy the compactness of the underlying DGM.
  - Use non-linear stability enforcement to achieve linear stability
    $\rightarrow$ Limiters in the case of RDG(P_0P_1) method
    $\rightarrow$ ENO/WENO reconstructions.

Objective

- Develop an RDG(P_1P_2) method based Hermit WENO reconstruction for solving the Euler equations on tetrahedral grids.
  - enhance the accuracy, and therefore reduce the high computational costs of the underlying DG methods
  - avoid the spurious oscillations in the vicinity of strong discontinuities, and therefore maintain the non-linear stability, and naturally linear stability.
- Attempt to address the two weakest links of the DG methods.

Hermit WENO Reconstruction

- On a tetrahedral cell *i*, a convex combination of the least-squares reconstructed 2nd order derivatives at the cell itself and its four face-neighboring cells

$$\frac{\partial^2 U}{\partial x_i \partial x_j}\Big|_i = \sum_{k=1}^{5} w_k \frac{\partial^2 U}{\partial x_i \partial x_j}\Big|_k$$

w_k: weighting function

o_k : oscillation indicator

$$w_k = \frac{(\varepsilon + o_k)^{-\gamma}}{\sum_{i=1}^{5} (\varepsilon + o_i)^{-\gamma}}$$

$$o_k = [\int_{\Omega_i} (\frac{\partial^2 U}{\partial x_i \partial x_j}\Big|_k)^2 d\Omega]^2$$

ε → a small positive number

γ → an integer parameter

Hermit WENO Reconstruction

- Central stencil

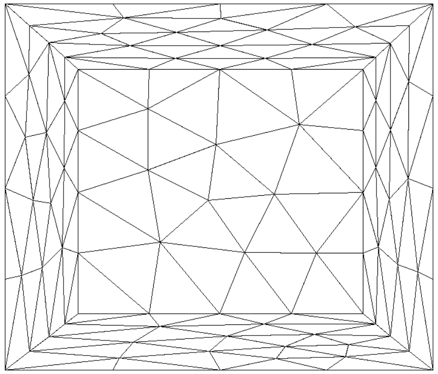
  → the least-squares reconstructed polynomial at the cell itself

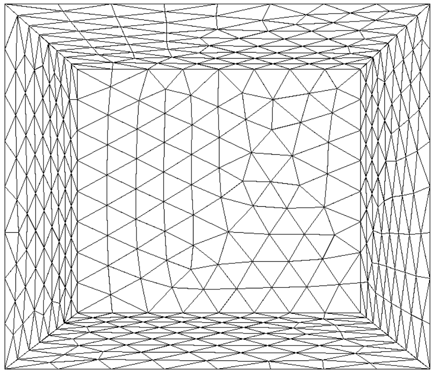
- Biased stencils

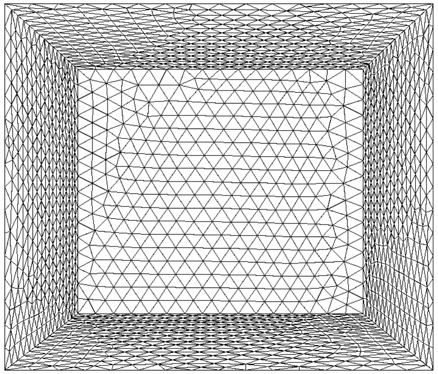
  → the least-squares reconstructed polynomials on its four face-neighboring cells

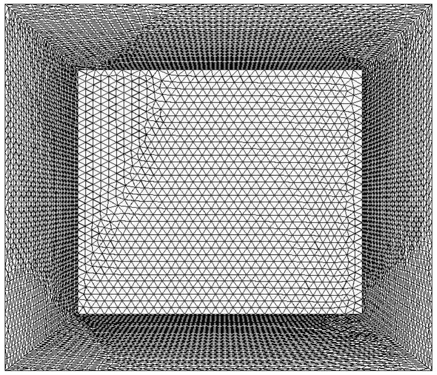
Example 1. Convergence Study for the Quadratic Hermit WENO Reconstruction

- Access the order of accuracy on tetrahedral grids.
- A smooth function f(x,y,z)=sin(πx)cos(2πy)sin(3πz)


547 elements
156 points
103 boundary pts


4406 elements
990 points
444 boundary pts


35697 elements
6973 points
1785 boundary pts


286702 elements
52093 points
7094 boundary pts

Solution Accuracy for the Quadratic Hermit WENO Reconstruction

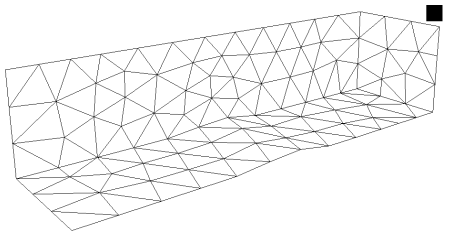
L_2-error and order of the convergence

Number of cells	L2-error	Order
547	3.44033E-2	-
4,406	4.24326E-03	3.01
35,697	4.46581E-04	3.23
286,705	4.93515E-05	3.17

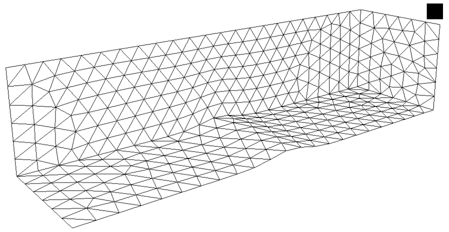
The Hermit WENO reconstruction delivers the designed 3rd order of convergence !!

Example 2. A Subsonic Flow through a Channel with a Smooth Bump (Ma=0.5, α=0°)

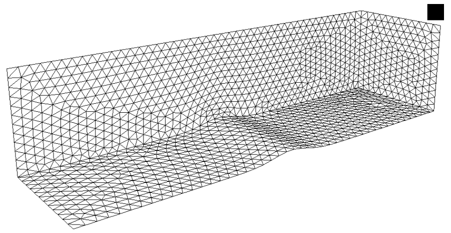
- Access the order of accuracy of the Hermit WENO RDG(P_0P_1, P_1P_1, P_1P_2) method for internal flows.
- Entropy production is served as the error measurement.



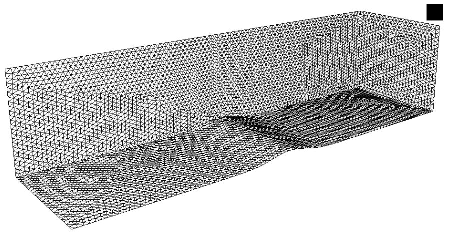
889 cells
254 pts
171 boundary pts



6986 cells
1555 pts
691 boundary pts

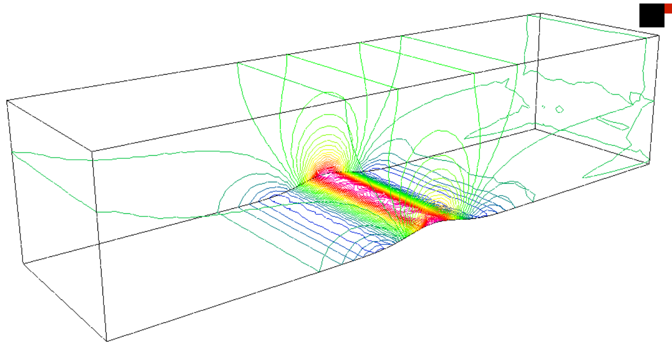


449522 cells
81567 pts
10999 boundary pts

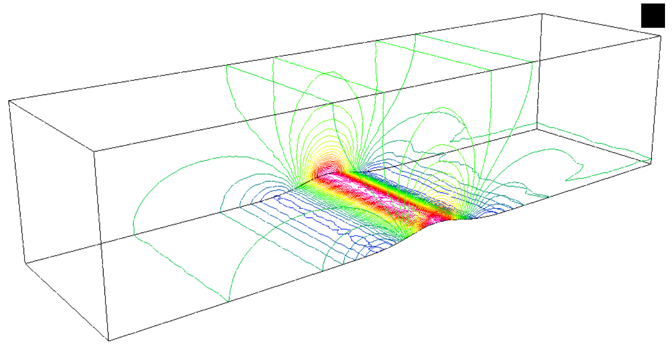


449522 cells
81567 pts
10999 boundary pts

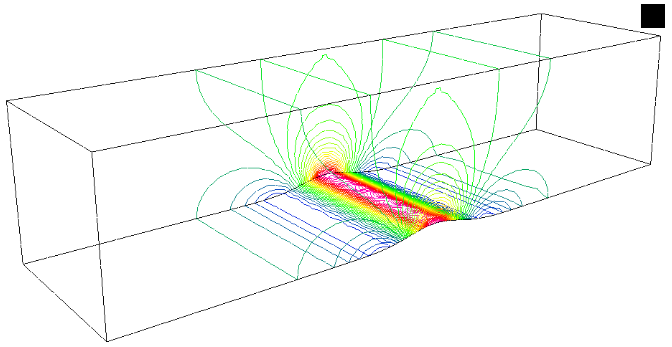
Computed Velocity Contours



Obtained by the RDG(P_0P_1) on the finest grid

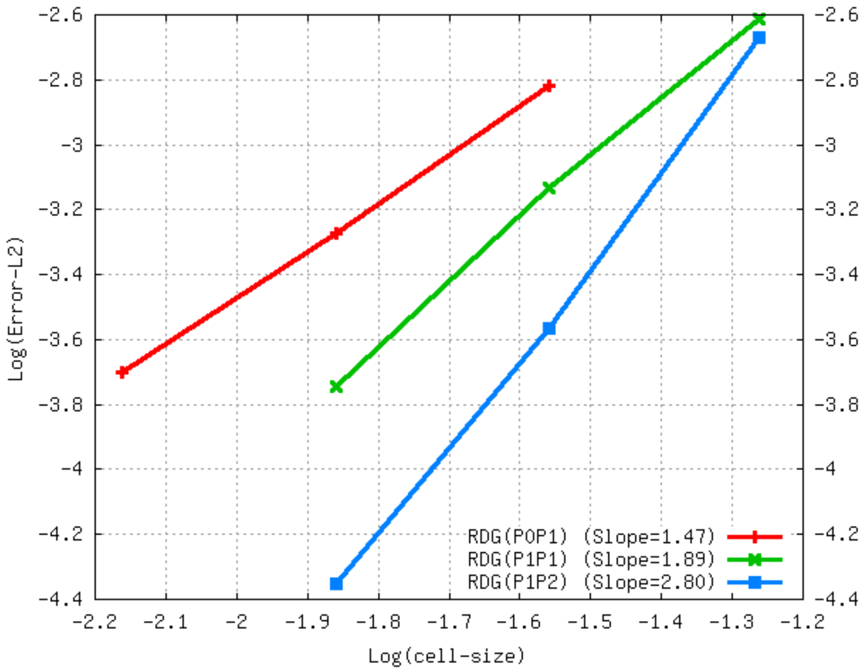


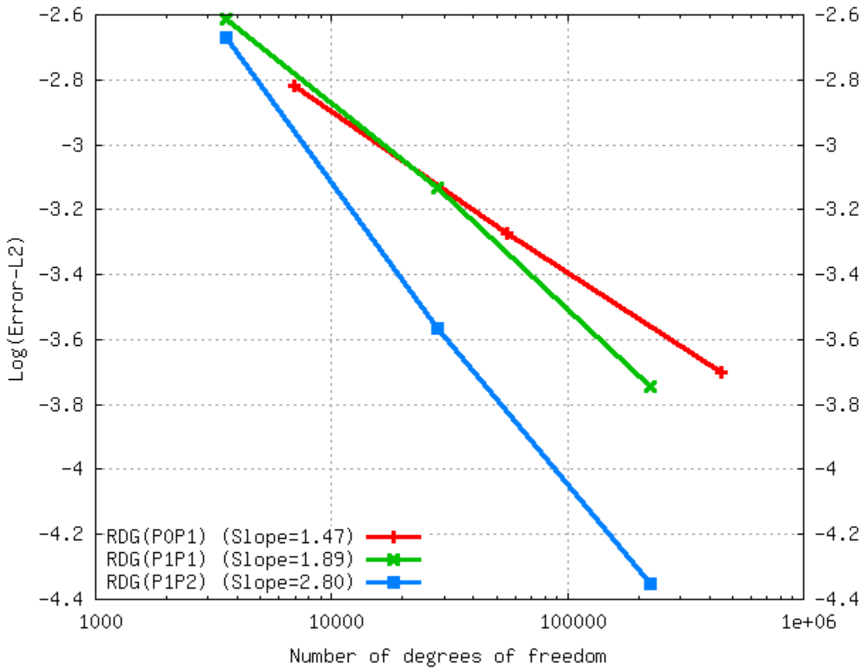
Obtained by the RDG(P_1P_1) on the fine grid



Obtained by the RDG(P_1P_2) on the fine grid

Solution accuracy for different RDG methods

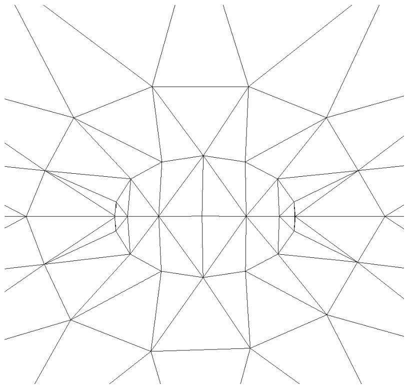




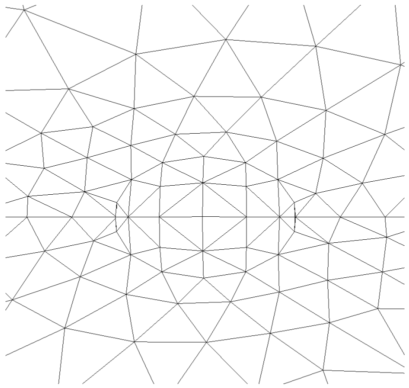
- Hermit WENO reconstruction-based RDG(P_1P_2) method
  - significantly increases the accuracy of the underlying DG method
  - greatly decreases its computational costs

Example 3. Subsonic Flow past a Sphere (Ma=0.5)

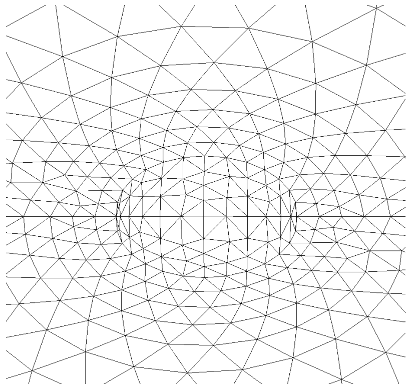
- Access the order of accuracy of the Hermit WENO RDG(P_0P_1, P_1P_1, P_1P_2) method for external flows.
- Entropy production is served as the error measurement.



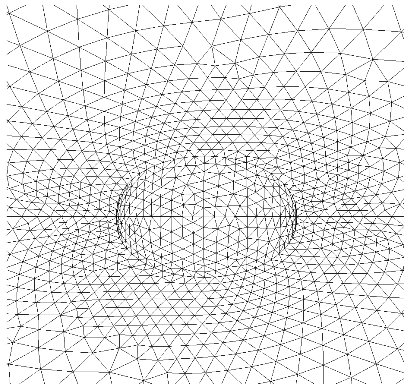
535 cells
167 points
124 boundary pts



2426 cells
598 points
322 boundary pts

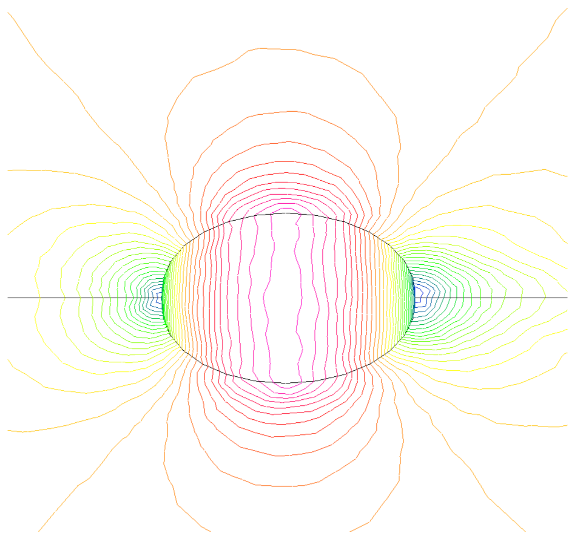


16467 cells
3425 points
1188 boundary pts

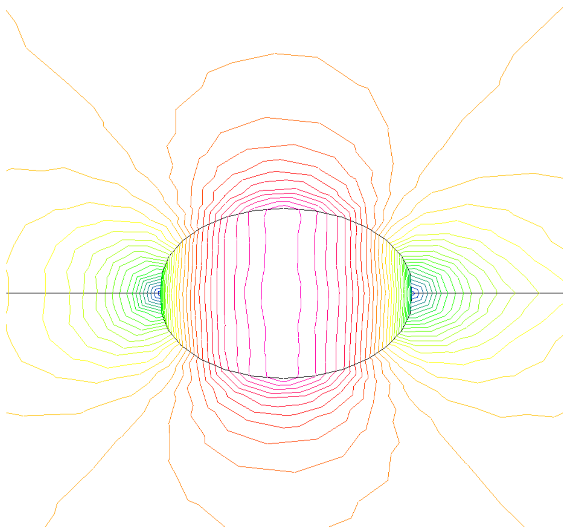


124706 cells
23462 points
4538 boundary pts

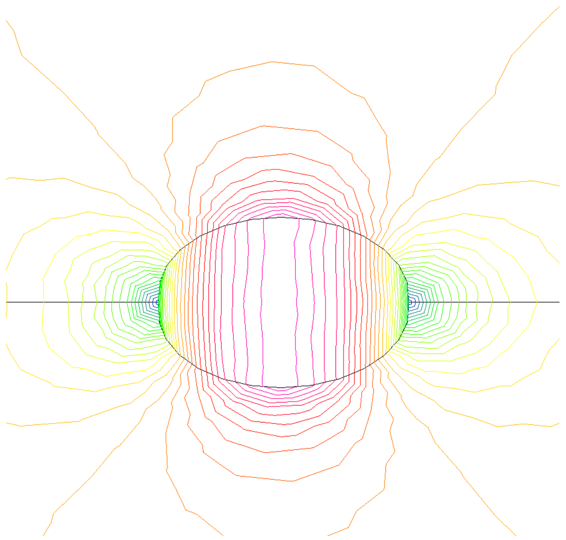
Computed Velocity Contours



Obtained by the RDG(P_0P_1) on the finest grid

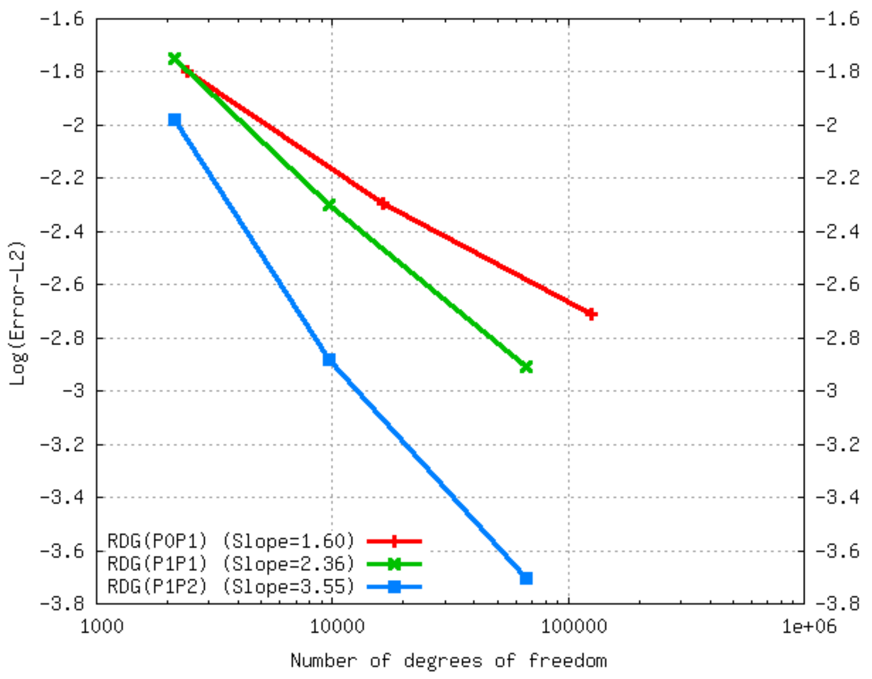


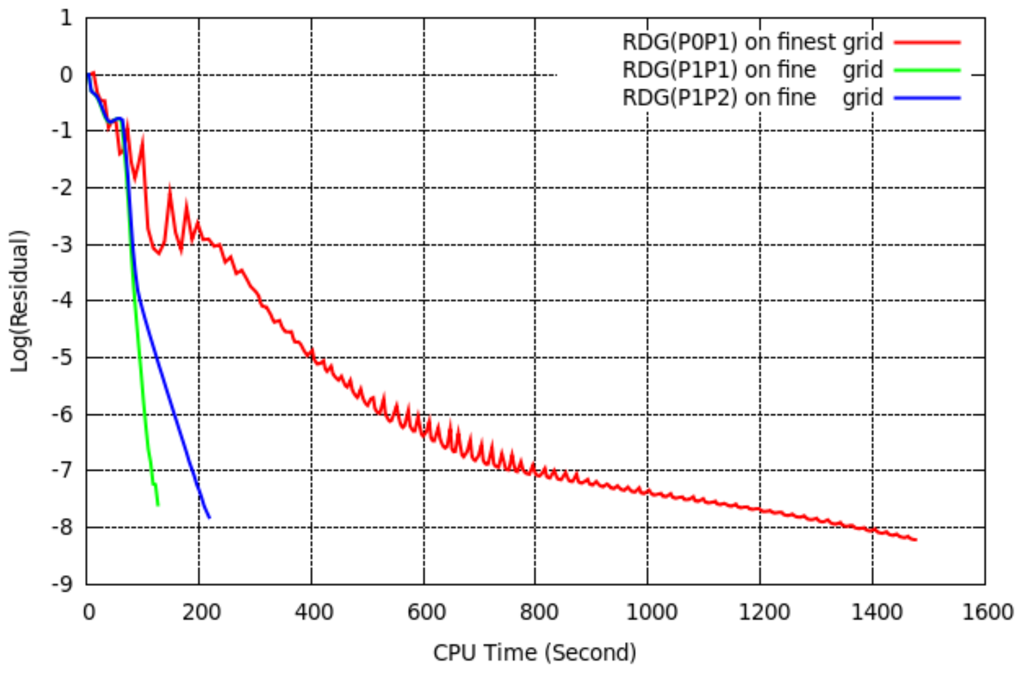
Obtained by the RDG(P_1P_1) on the fine grid



Obtained by the RDG(P_1P_2) on the fine grid

Efficiency Comparison for Different RDG Methods

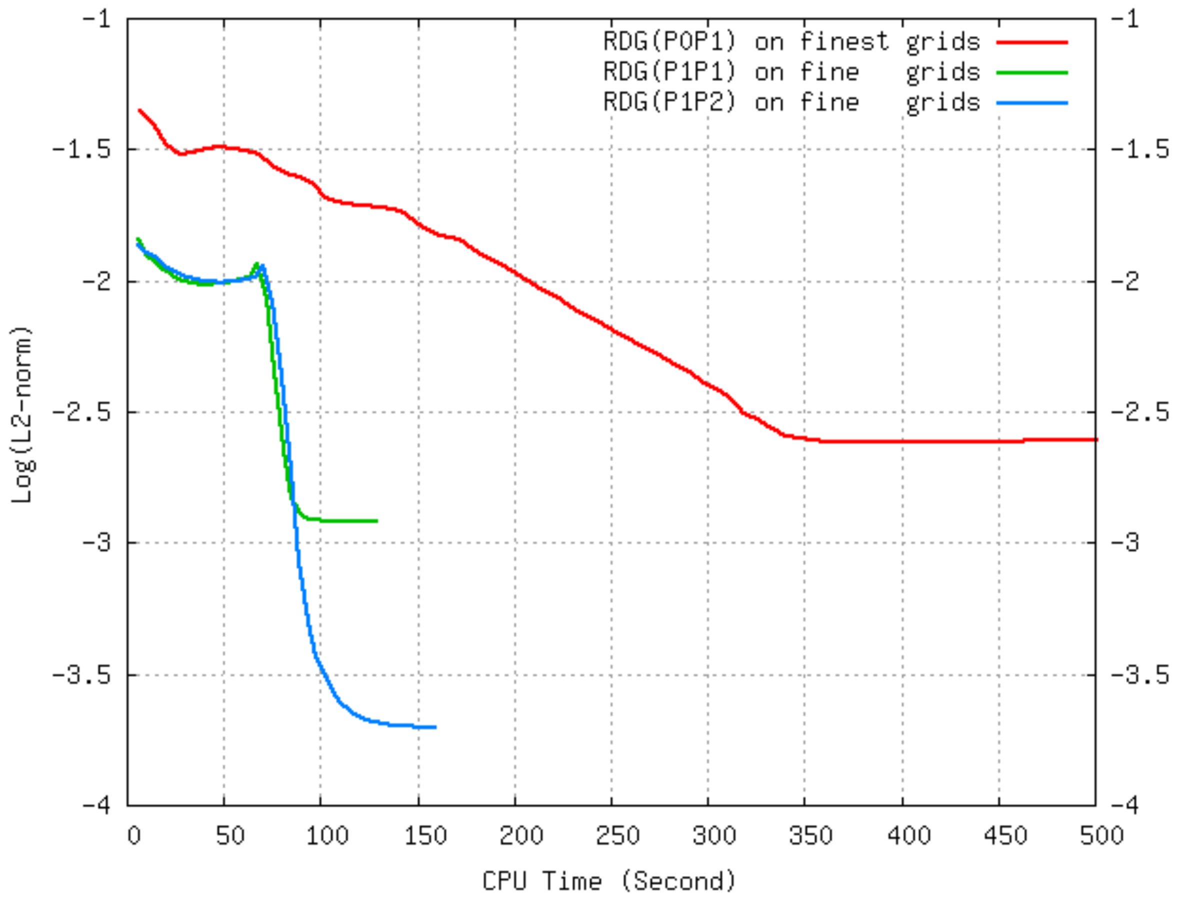




Convergence order
versus
number of degree of freedom

Convergence history
versus
CPU time (Second)

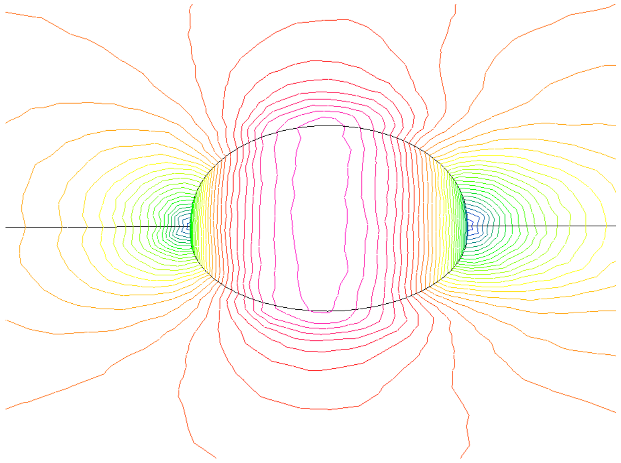
Efficiency Comparison for Different RDG Methods



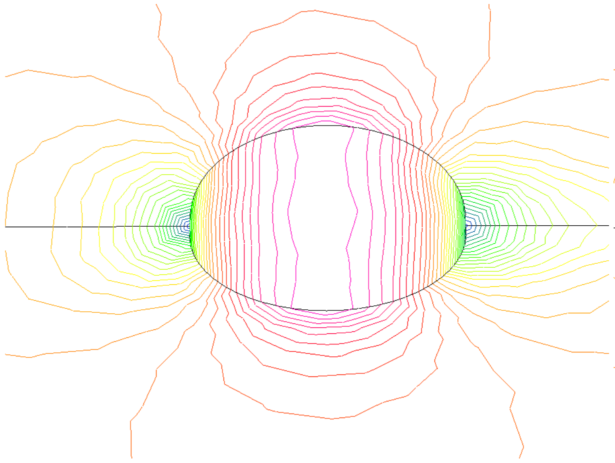
L2 norm versus CPU time

Example 4. Low Mach Number Flow past a Sphere (Ma=0.01)

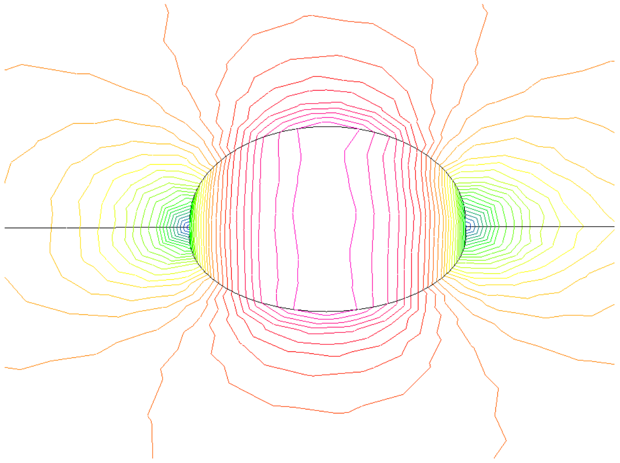
- Access the accuracy for solving low Mach number flow problems.



Velocity contours obtained by the RDG(P_0P_1) on the finest grid

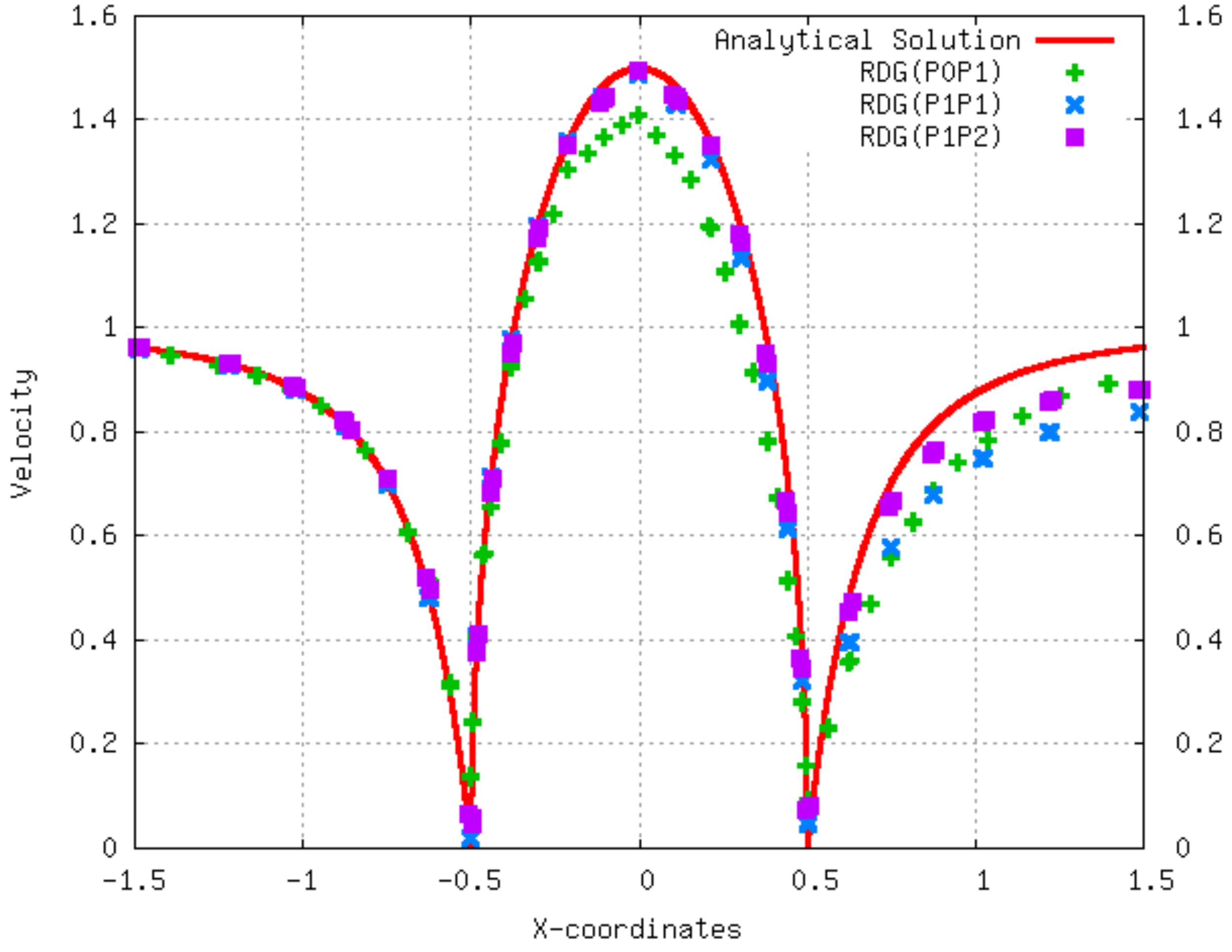


Velocity contours obtained by the RDG(P_1P_1) on the fine grid



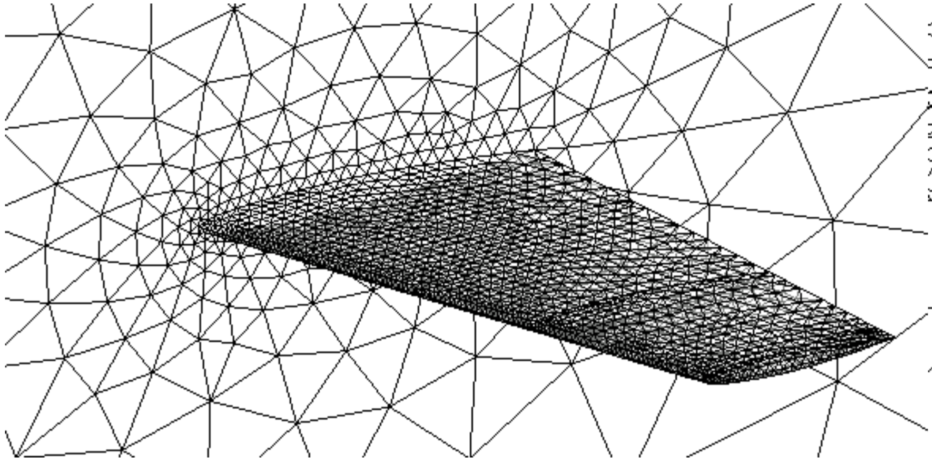
Velocity contours obtained by the RDG(P_1P_2) on the fine grid

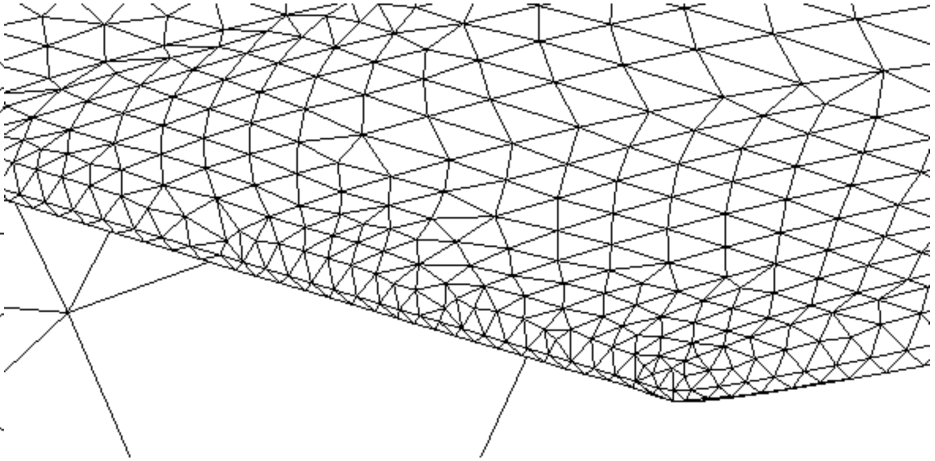
Comparison of the Computed Velocity Distributions on the Surface of the Sphere



Example 5. Transonic Flow past an ONERA M6 Wing (Ma=0.699, α=3.06°)

- Access the accuracy and robustness of the RDG(P_1P_2) method for transonic flow problems.

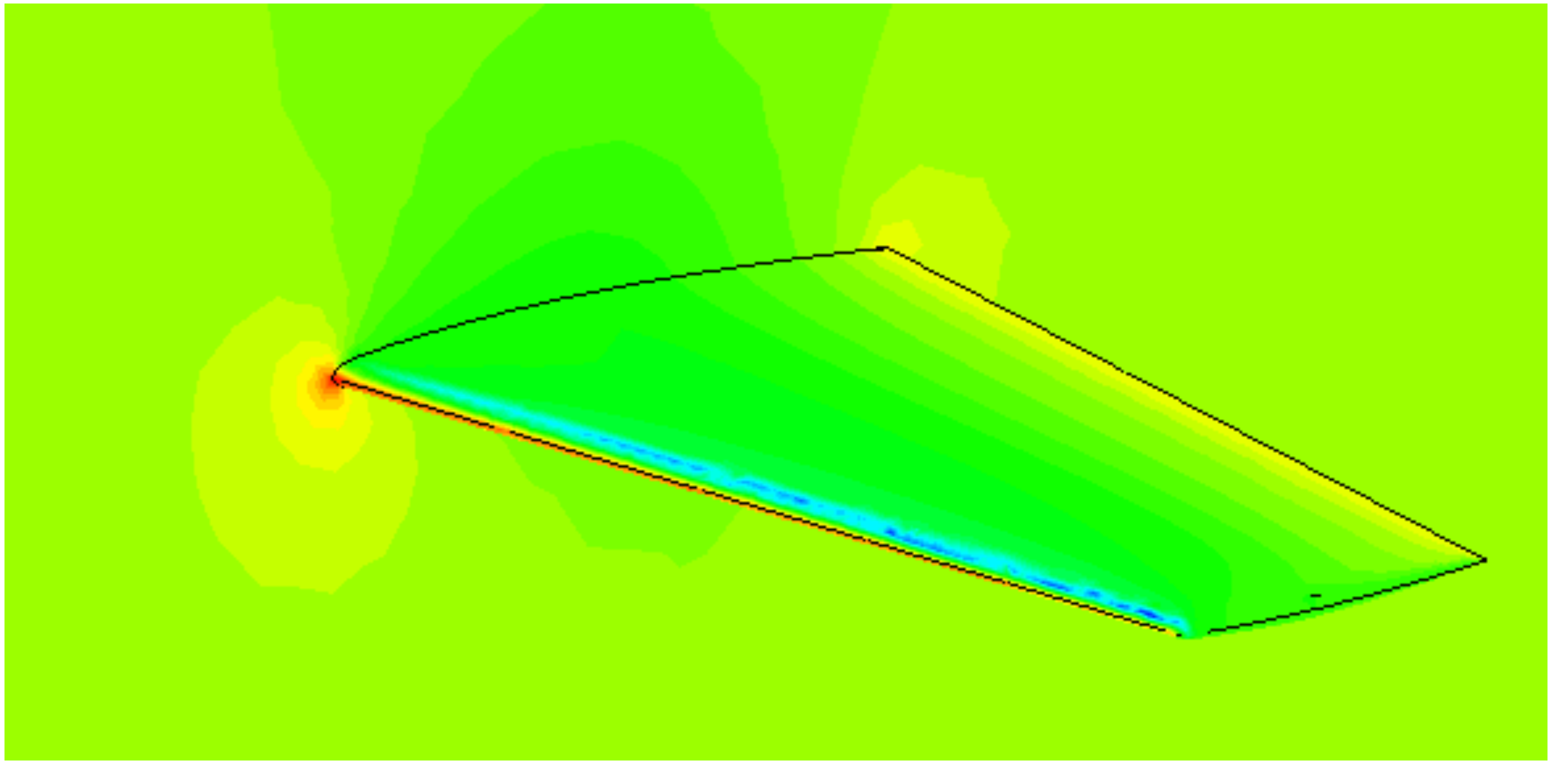




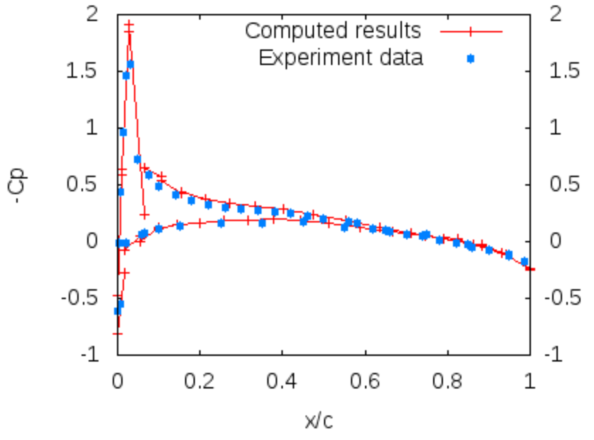
41,440 elements
8,325 grid points
2,575 boundary points

coarseness of grids
in the vicinity of
the leading edge

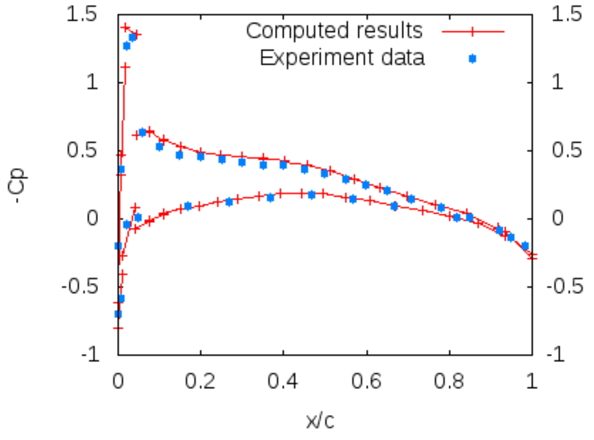
Computed Pressure Contours by RDG(P_1P_2)



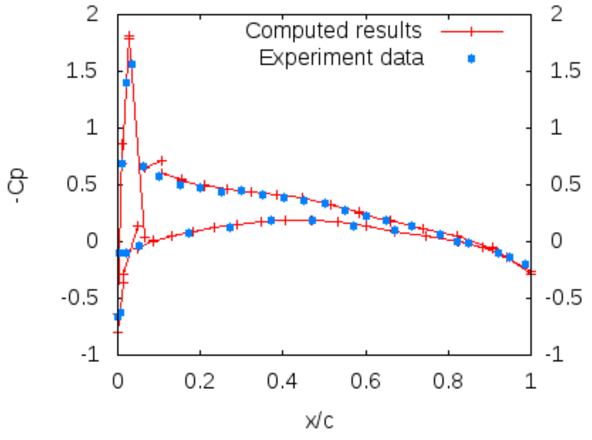
Computed Pressure Coefficient Distributions Compared with Experimental Data



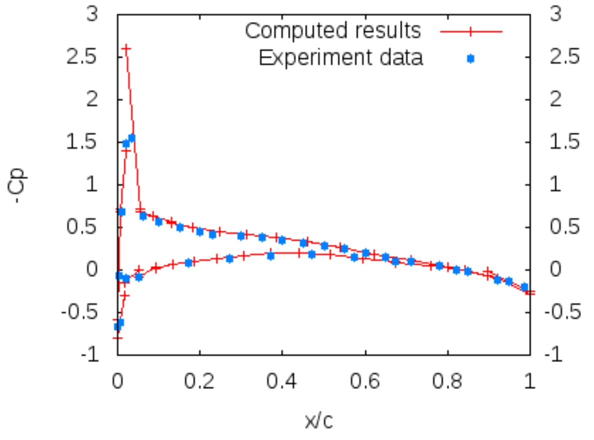
η=0.20



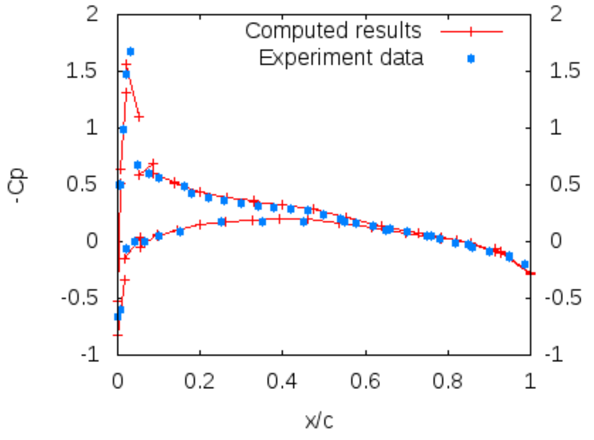
η=0.44



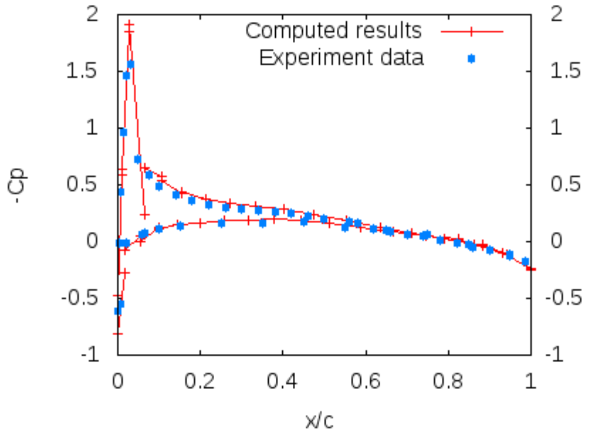
η=0.65



η=0.80



η=0.90

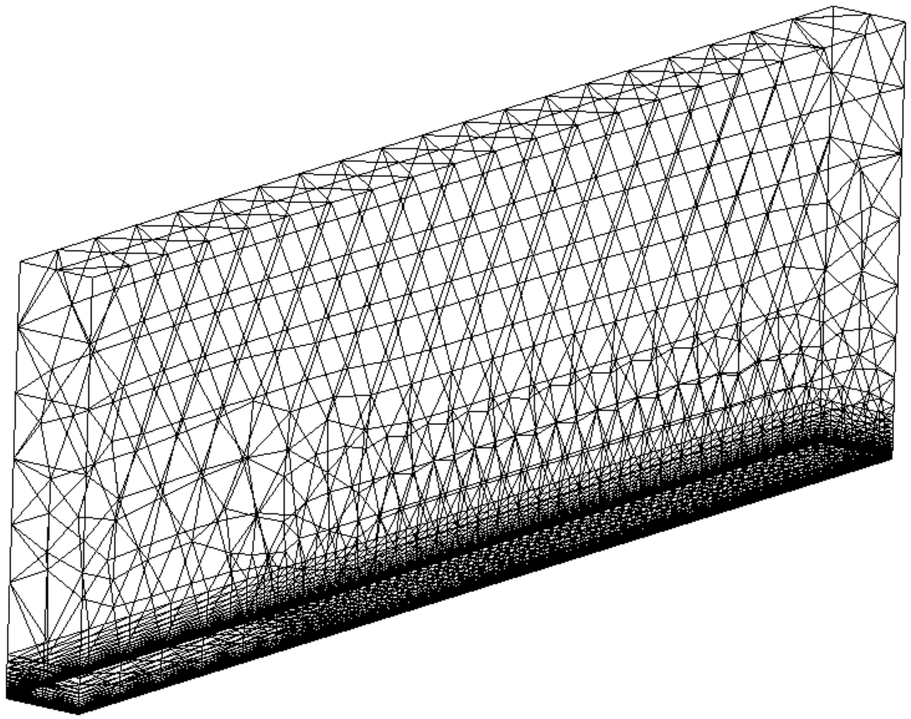


η=0.95

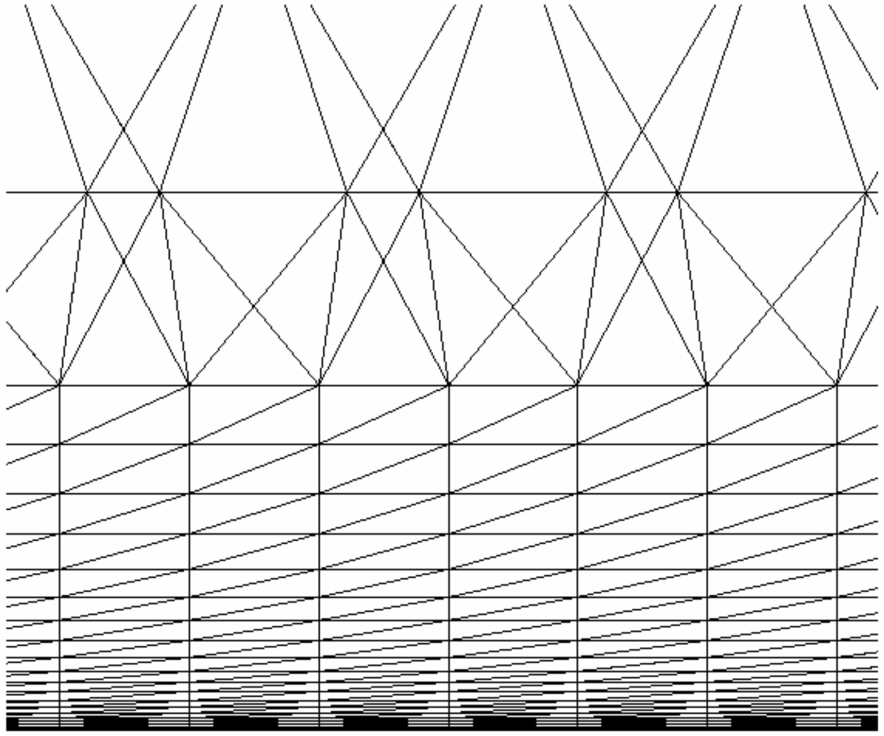
Example 6. Blasius Boundary Layer Solution

- Demonstrate that the RDG(P_1P_2) method is able to obtain the accurate solution for the viscous flow problems.
- Flow condition: Ma=0.5, Re=100,000

Grids Used for Computing the Blasius Solution

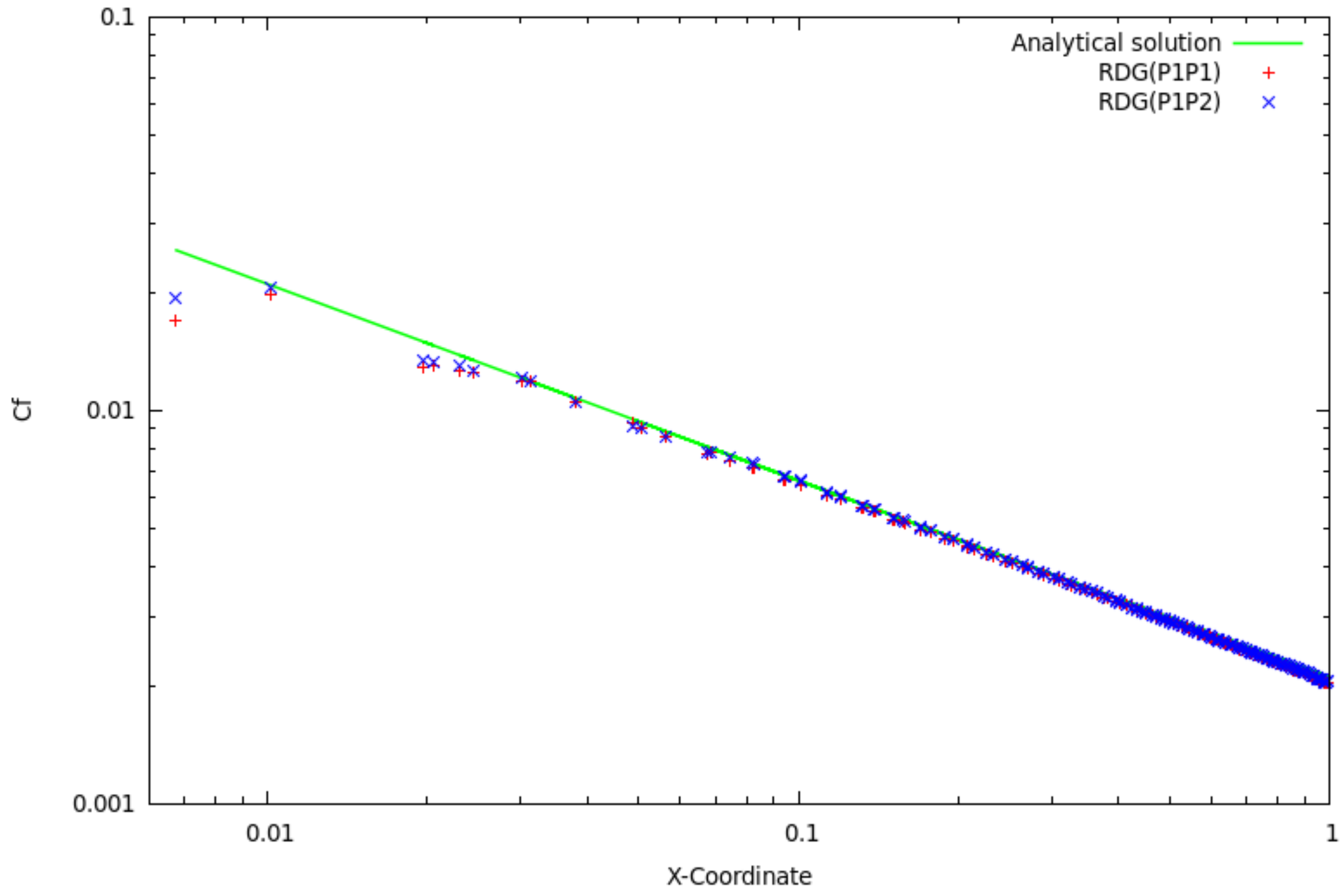


47,535 elements
9,828 points
3,631 boundary points



Grids in the boundary layer

Comparison of Computed skin friction coefficients between RDG(P_1P_2) and RDG(P_1P_1)



Concluding Remarks

- An RDG(P_1P_2) method based on a Hermit WENO reconstruction, designed not only to enhance the underlying DG method but also to maintain non-linear stability, has been presented for solving the compressible Euler equations on tetrahedral grids.

- This RDG(P_1P_2) method is able to deliver the designed 3rd order of accuracy, and outperforms the second-order finite volume method RDG(P_0P_1) by orders of magnitudes to achieve the same accuracy.

- This RDG(P_1P_2) has also successfully been extended to problems with strong discontinuities using a hierarchical reconstruction and viscous flows, which will be discussed next.

Background

- Objective: Develop a RDG method based on a hierarchical WENO reconstruction: HWENO(P_1P_2), for compressible flows with strong discontinuities on hybrid grids.
  - enhance the accuracy, and therefore reduce the high computational costs of the underlying DG methods
  - avoid the spurious oscillations in the vicinity of strong discontinuities, and therefore maintain the non-linear stability, and naturally linear stability.

- Attempt to address the two weakest links of the DG methods.

WENO reconstruction at P_2: WENO(P_1P_2)

- Objective:
  - Reconstruct a quadratic polynomial solution (P_2) from the underlying discontinuous linear polynomial DG polynomial solution (P_1) based on a WENO reconstruction.

2-exactness Least-squares Reconstruction

- From a linear polynomial DG solution in any cell i

$$\mathbf{U}_i = \widetilde{\mathbf{U}}_i + \mathbf{U}_{\mathbf{x}i} B_2 + \mathbf{U}_{\mathbf{yi}} B_3 + \mathbf{U}_{zi} B_4$$

- Reconstruct a quadratic polynomial solution U^{R}

$$\begin{aligned}\mathbf{U}_i^R = \widetilde{\mathbf{U}}_i^R &+ \mathbf{U}_{xi}^R B_2 + \mathbf{U}_{yi}^R B_3 + \mathbf{U}_{zi}^R B_4 \\ &+ \mathbf{U}_{xxi}^R B_5 + \mathbf{U}_{yyi}^R B_6 + \mathbf{U}_{zzi}^R B_7 \\ &+ \mathbf{U}_{xyi}^R B_8 + \mathbf{U}_{xzi}^R B_9 + \mathbf{U}_{yzi}^R B_{10}\end{aligned}$$

- 10 degrees of freedom

Requirements for Reconstruction

- Conservation
- Compactness
    - Maintain the compactness of the underlying DG method
    - Necessary for unstructured arbitrary grids
    - Stencils involve only Von Neumann neighborhood (adjacent face-neighboring cells)

Least-squares Reconstruction (P_1P_2)

- Requiring conservation and reconstructed first derivatives equal to the ones of the underlying DG solution leads

$$\widetilde{\mathbf{U}}_i^R = \widetilde{\mathbf{U}}_i\,, \quad \mathbf{U}_{xi}^R = \mathbf{U}_{xi}\,, \quad \mathbf{U}_{yi}^R = \mathbf{U}_{yi}\,, \quad \mathbf{U}_{zi}^R = \mathbf{U}_{zi}$$

due to the judicious choice of Taylor-basis in the DG formulation

- Six second derivatives only need to be reconstructed

Least-squares Reconstruction (P_1P_2)

For a face-neighboring cell j:

$$\mathbf{U}_j = \widetilde{\mathbf{U}}_i + \mathbf{U}_{xi} B_2 + \mathbf{U}_{yi} B_3 + \mathbf{U}_{zi} B_4$$

$$+ \mathbf{U}^R_{xxi} B_5 + \mathbf{U}^R_{yyi} B_6 + \mathbf{U}^R_{zzi} B_7 + \mathbf{U}^R_{xyi} B_8 + \mathbf{U}^R_{xzi} B_9 + \mathbf{U}^R_{yzi} B_{10}$$

$$\frac{\partial \mathbf{U}}{\partial x}\bigg|_j = \mathbf{U}_{xi} \frac{1}{\Delta x_i} + \mathbf{U}^R_{xxi} \frac{B_2}{\Delta x_i} + \mathbf{U}^R_{xyi} \frac{B_3}{\Delta x_i} + \mathbf{U}^R_{xzi} \frac{B_4}{\Delta x_i}$$

$$\frac{\partial \mathbf{U}}{\partial y}\bigg|_j = \mathbf{U}_{yi} \frac{1}{\Delta y_i} + \mathbf{U}^R_{yyi} \frac{B_3}{\Delta y_i} + \mathbf{U}^R_{xyi} \frac{B_2}{\Delta y_i} + \mathbf{U}^R_{yzi} \frac{B_4}{\Delta y_i}$$

$$\frac{\partial \mathbf{U}}{\partial z}\bigg|_j = \mathbf{U}_{zi} \frac{1}{\Delta z_i} + \mathbf{U}^R_{zzi} \frac{B_4}{\Delta z_i} + \mathbf{U}^R_{xzi} \frac{B_2}{\Delta z_i} + \mathbf{U}^R_{yzi} \frac{B_3}{\Delta z_i}$$

Least-squares Reconstruction (P_1P_2)

- Similar equations can be written for all face-neighboring cells, which leads to a non-square matrix.

- The size of resulting non-square matrix is (4xnface)×6, where nface is the number of face-neighboring cells.
  - nface=4 for a tetrahedral cell
  - nface=5 for a prismatic or pyramidal cell
  - nface=6 for a hexhedral cell

- This over-determined linear system of (4xnface) equations for 6 unknowns can be solved in the least-squares sense.

Least-squares Reconstruction (P_1P_2)

- Simple and straightforward
- The boundary conditions are used to obtain the reconstructed equations for the boundary cells, thus ensuring existence of an over-determined system.

Instability Issues of RDG Method in 3D

- The RDG method suffers from linear instability, similar to RDG(P_0P_1)
  - Instability occurs even for linear equations and in smooth flows.
  - Reconstruction stencils only involve von Neumann neighborhood, i.e., adjacent face-neighboring cells

- To maintain the linear stability
  - Augment stencils in the reconstruction

    → Destroy the compactness of the underlying DGM.
  - Use non-linear stability enforcement to achieve linear stability

    → Limiters in the case of RDG(P_0P_1) method

    → ENO/WENO reconstructions.

Hermite WENO Reconstruction

- On a tetrahedral cell *i*, a convex combination of the least-squares reconstructed 2nd order derivatives at the cell itself (k=0) and its face-neighboring cells (k=1,…,nface)

$$\left.\frac{\partial^2 U}{\partial x_i \partial x_j}\right|_i = \sum_{k=0}^{nface} w_k \left.\frac{\partial^2 U}{\partial x_i \partial x_j}\right|_k$$

w_k: weighting function

$$w_k = \frac{(\varepsilon + o_k)^{-\gamma}}{\sum_{i=0}^{nface} (\varepsilon + o_i)^{-\gamma}}$$

o_k : oscillation indicator

$$o_k = \left[\int_{\Omega_i} \left(\left.\frac{\partial^2 U}{\partial x_i \partial x_j}\right|_k\right)^2 d\Omega\right]^2$$

ε → a small positive number

γ → an integer parameter

Hermite WENO Reconstruction

- Central stencil

  → the least-squares reconstructed polynomial at the cell itself

- Biased stencils

  → the least-squares reconstructed polynomials on its face-neighboring cells

WENO reconstruction at P_1 : HWENO(P_1P_2)

- Observation:

  Although the WENO(P_1P_2) method does not introduce any new oscillatory behavior for the reconstructed curvature terms (second derivatives) due to the WENO reconstruction, it cannot remove inherent oscillations in the underlying DG(P1) solutions, leading to non-linear instability.

- Objective:

  Reconstruct and modify the linear part (first derivatives) of the resulting quadratic polynomial solution (P_2) in order to ensure non-linear instability for flows with strong discontinuities using WENO reconstruction.

WENO reconstruction at P_1 : HWENO(P_1P_2)

- The following nface stencils (i,j_1), (i,j_2), …, and (i,j_nface) are chosen to construct a Hermite polynomial such that

$$\frac{\partial \mathbf{U}}{\partial x}\Big|_j = \mathbf{U}^R_{xi}\frac{1}{\Delta x_i} + \mathbf{U}^R_{xxi}\frac{B_2}{\Delta x_i} + \mathbf{U}^R_{xyi}\frac{B_3}{\Delta x_i} + \mathbf{U}^R_{xzi}\frac{B_4}{\Delta x_i}$$

$$\frac{\partial \mathbf{U}}{\partial y}\Big|_j = \mathbf{U}^R_{yi}\frac{1}{\Delta y_i} + \mathbf{U}^R_{yyi}\frac{B_3}{\Delta y_i} + \mathbf{U}^R_{xyi}\frac{B_2}{\Delta y_i} + \mathbf{U}^R_{yzi}\frac{B_4}{\Delta y_i}$$

$$\frac{\partial \mathbf{U}}{\partial z}\Big|_j = \mathbf{U}^R_{zi}\frac{1}{\Delta z_i} + \mathbf{U}^R_{zzi}\frac{B_4}{\Delta z_i} + \mathbf{U}^R_{xzi}\frac{B_2}{\Delta z_i} + \mathbf{U}^R_{yzi}\frac{B_3}{\Delta z_i}$$

Hermite WENO Reconstruction

- On a cell *i*, a convex combination of these nface (k=1,2,…,nface) reconstructed 1st derivatives and the first derivatives at the cell itself (k=0) is used to modify the first derivatives

$$\left.\frac{\partial U}{\partial x_i}\right|_i = \sum_{k=0}^{nface} w_k \left.\frac{\partial U}{\partial x_i}\right|_k$$

w_k: weighting function

$$w_k = \frac{(\varepsilon + o_k)^{-\gamma}}{\sum_{i=0}^{nface} (\varepsilon + o_i)^{-\gamma}}$$

o_k : oscillation indicator

$$o_k = \left[\int_{\Omega_i} \left(\left.\frac{\partial U}{\partial x_i}\right|_k\right)^2 d\Omega\right]^{1/2}$$

ε → a small positive number

γ → an integer parameter

Hermite WENO Reconstruction

- Central stencil

  → the gradient from the DG solution itself at cell itself

- Biased stencils

  → the eight reconstructed gradients

Numerical Examples

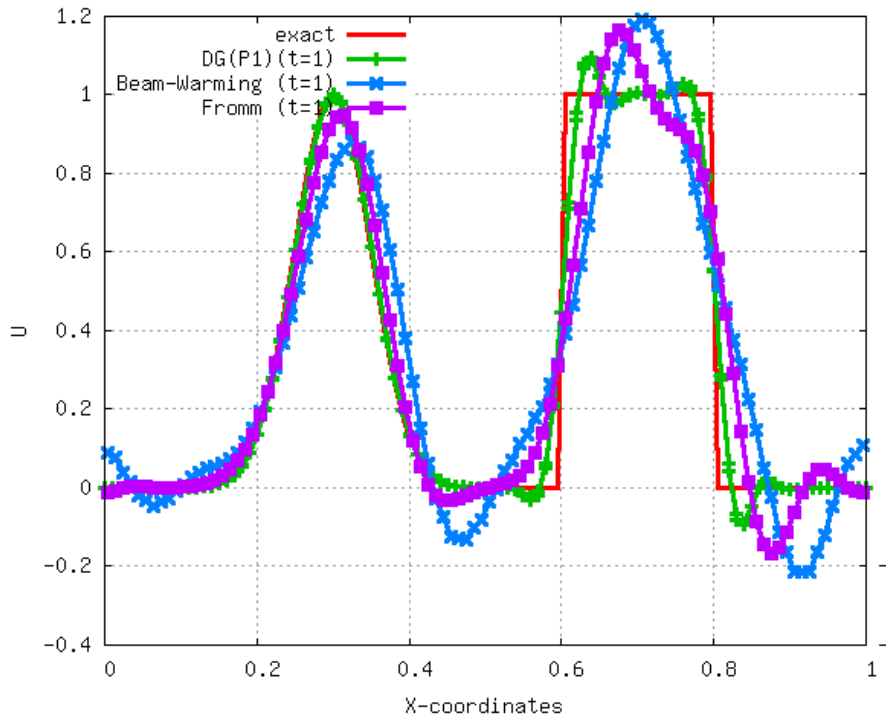
Numerical examples are presented to demonstrate

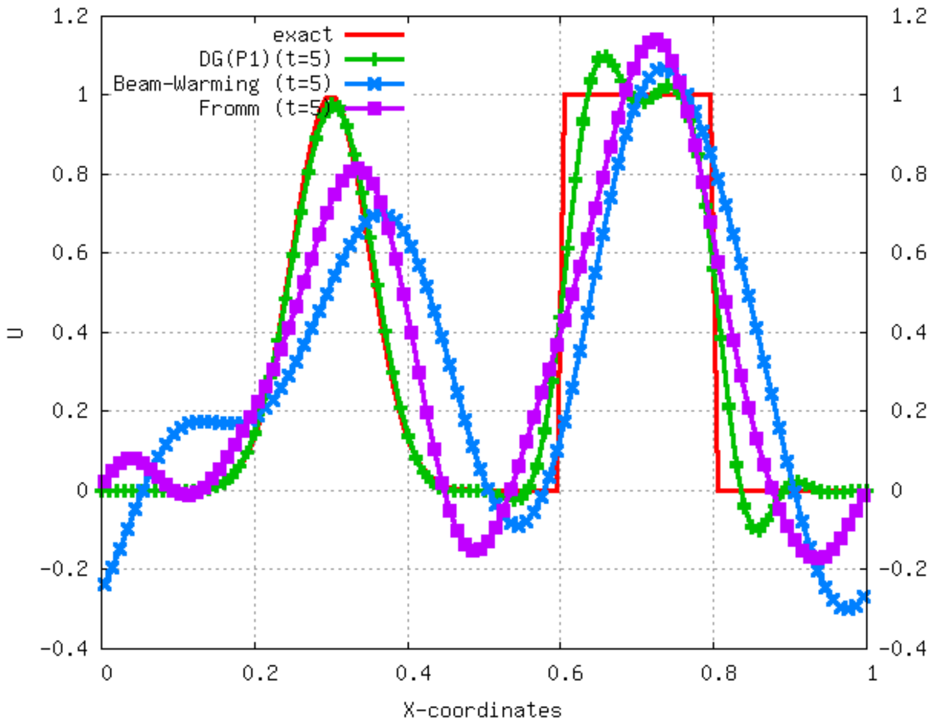
- Accuracy
- Robustness
- Essentially oscillation-free property

of the RDG method

Example 1. Convection of a Gaussian and a square wave

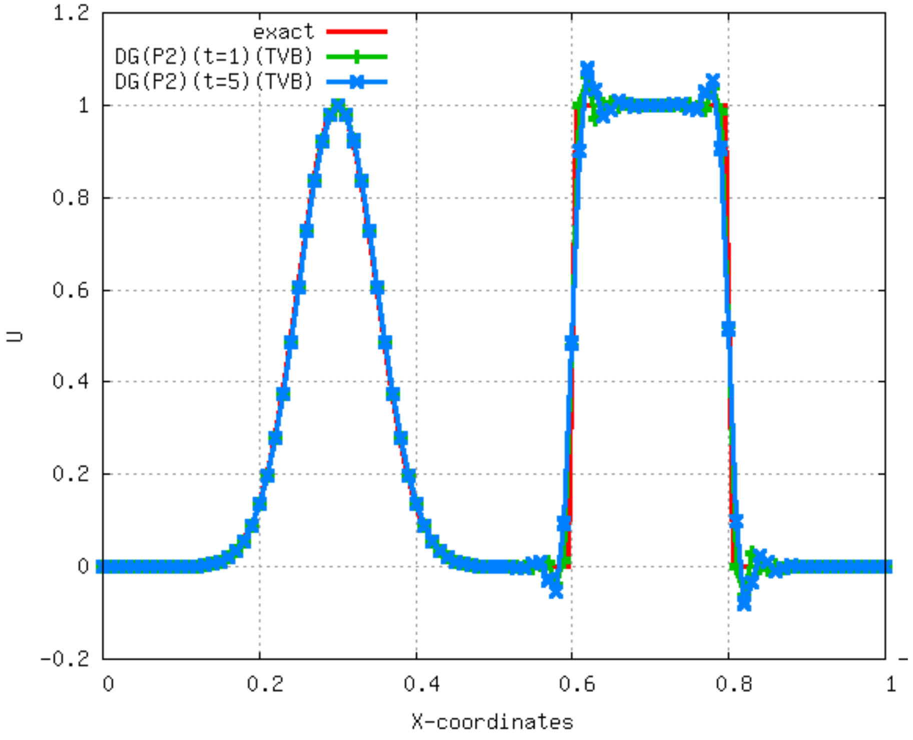
This simple test case is chosen to demonstrate the accuracy of the RDG method.

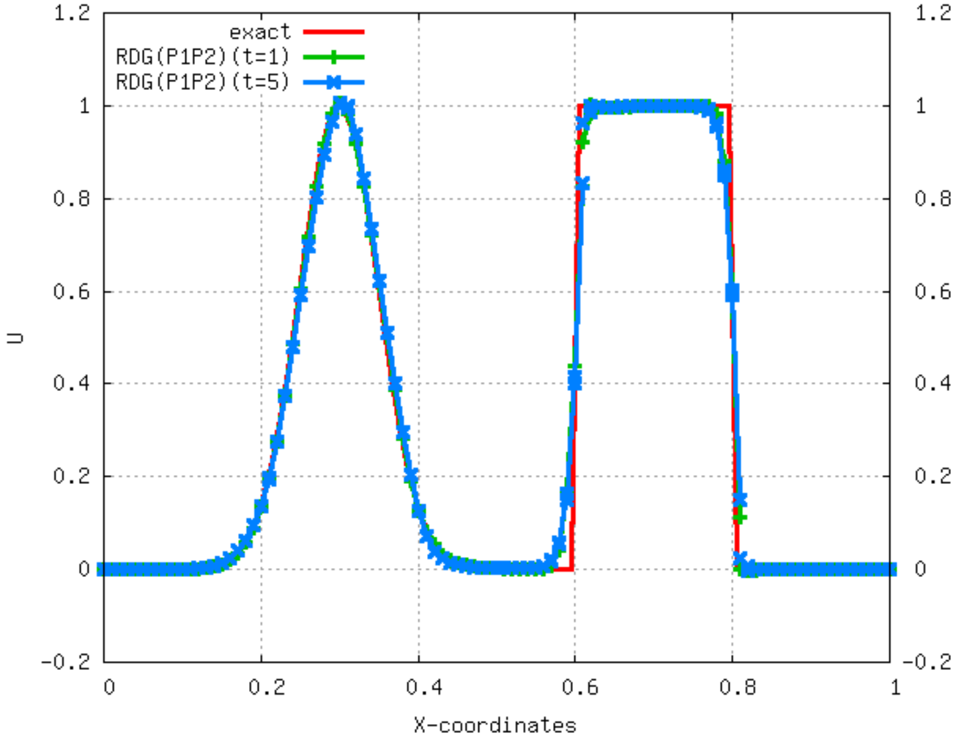




The superior dissipation and dispersion property of DG !

Convection of a Gaussian and a square wave





Note the high accuracy and oscillation-free of the RDG !

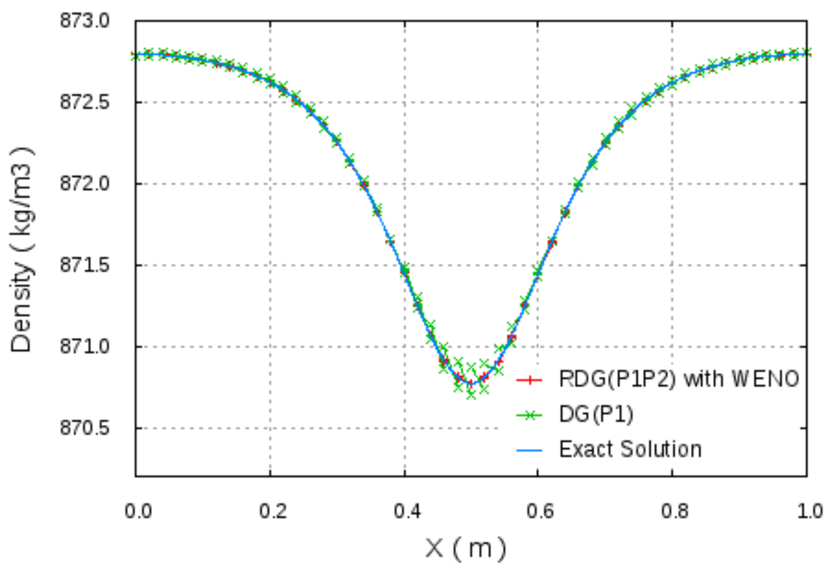
Example 2

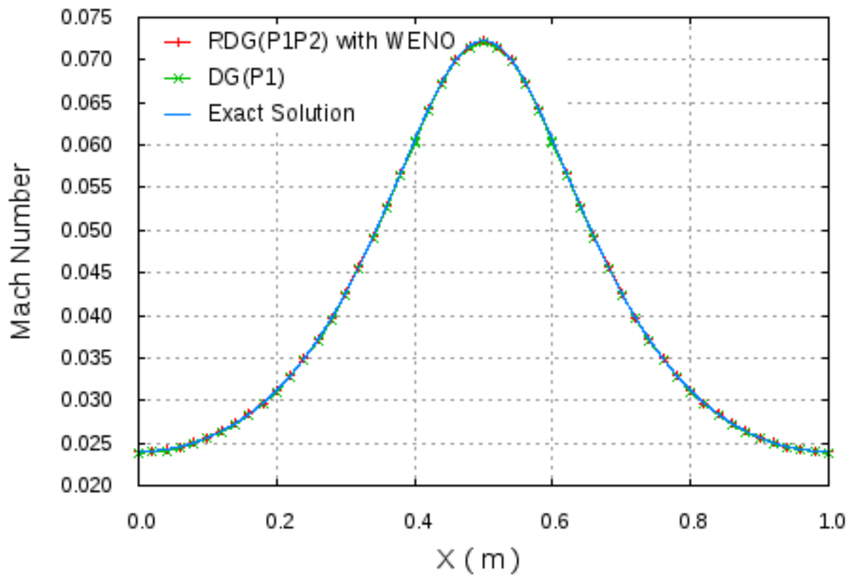
Water/vapor flow in a convergent-divergent nozzle

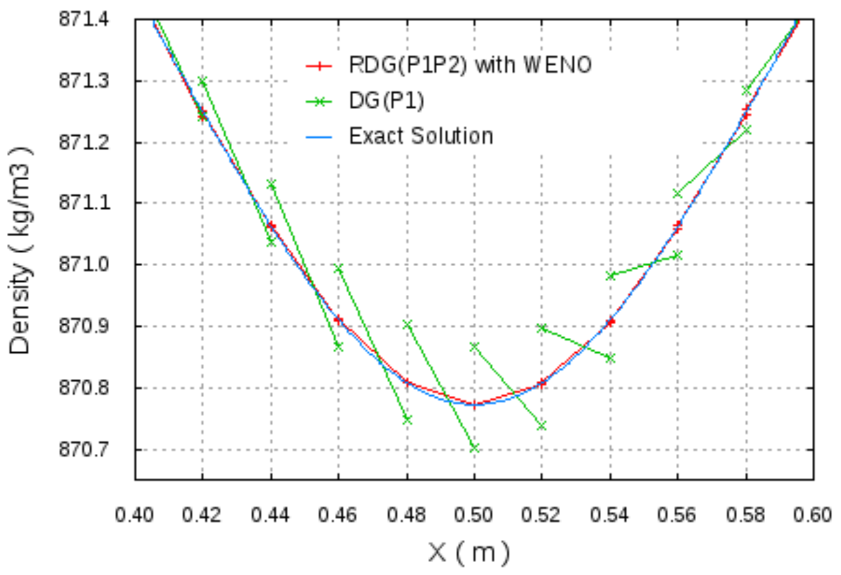
Stiffened EOS is used.

This example is chosen to demonstrate the robustness of the RDG method.

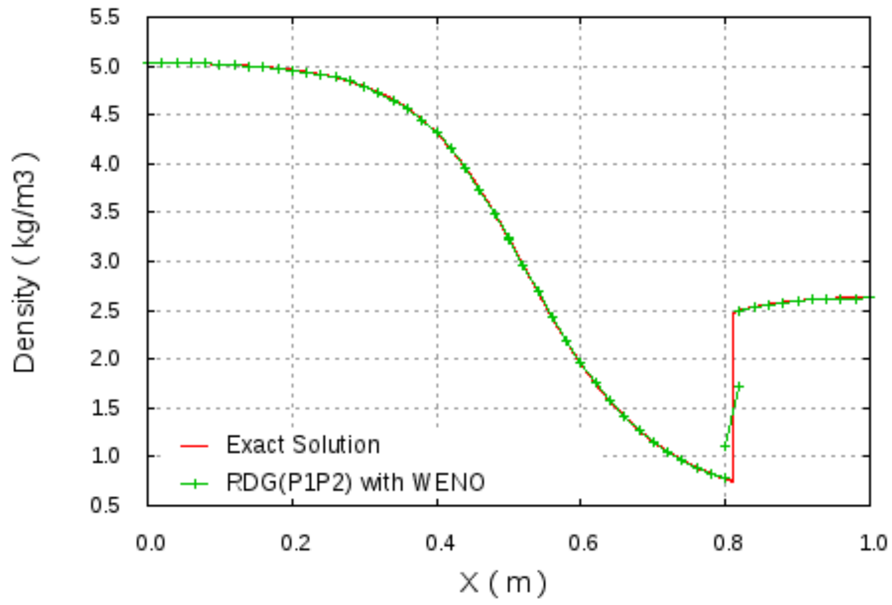
Water flow in a convergent-divergent nozzle

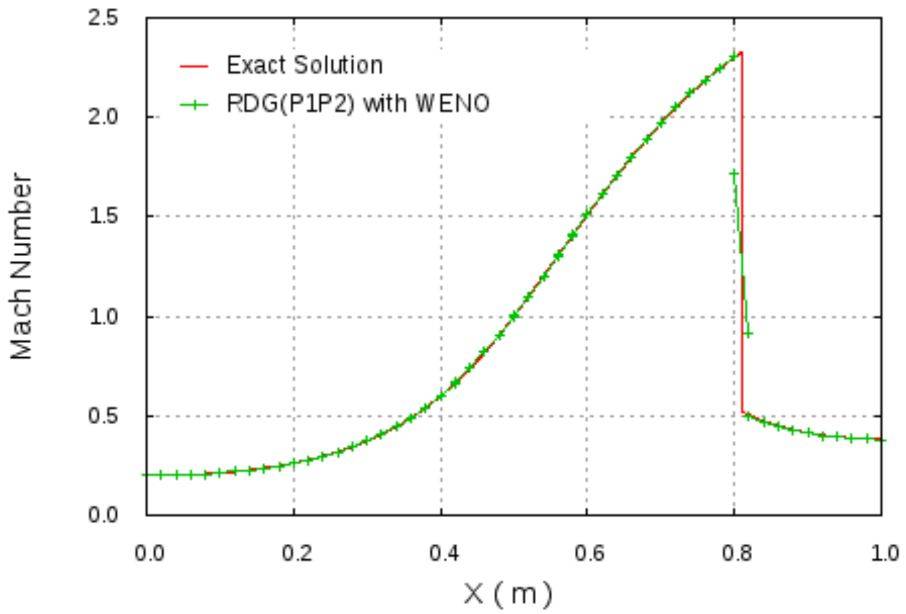






Vapor flow in a convergent-divergent nozzle

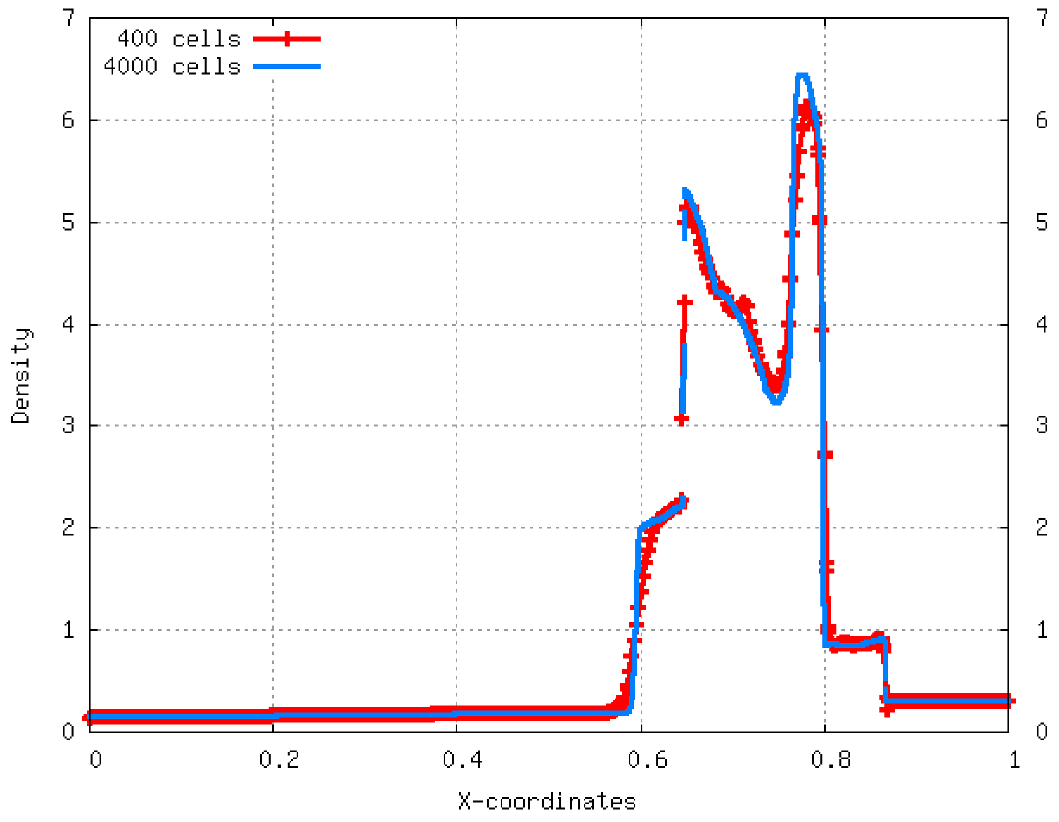




No single parameter is changed !!!

No time-derivative preconditioner is required !!!

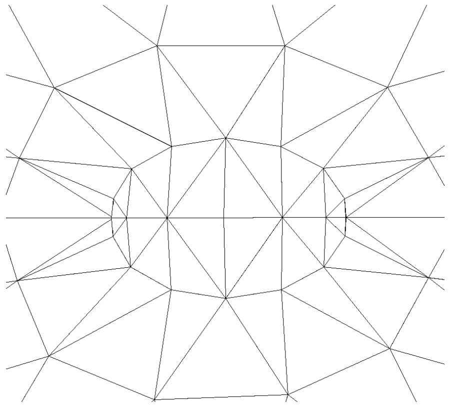
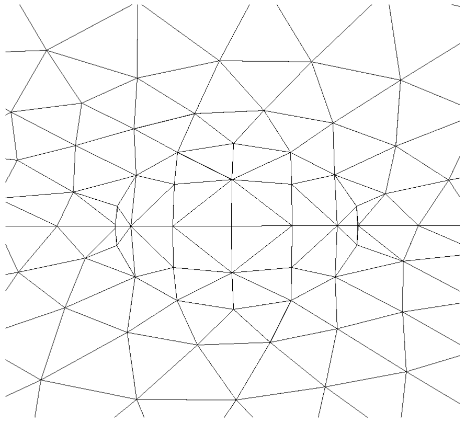
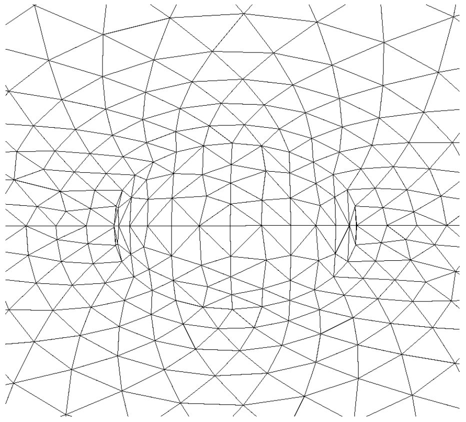
Example 3. Woodward-Collela blast wave problem



This example is chosen to demonstrate the essentially non-oscillatory property of the RDG method.

Example 4. A Subsonic Flow past a Sphere ($M_\infty=0.5$)

- Access the order of accuracy of the RDG(P_1P_1), WENO(P_1P_2) and HWENO(P_1P_2) methods for external flows.
- Entropy production is served as the error measurement.

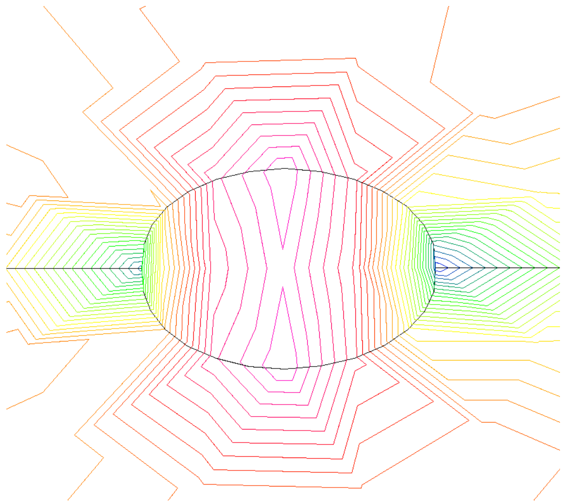




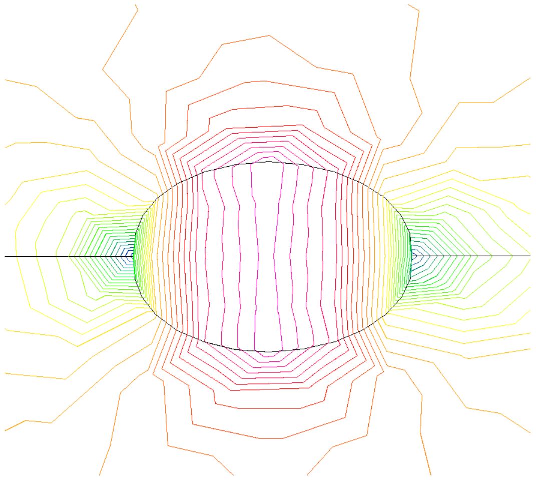
535 cells
167 points
124 boundary pts

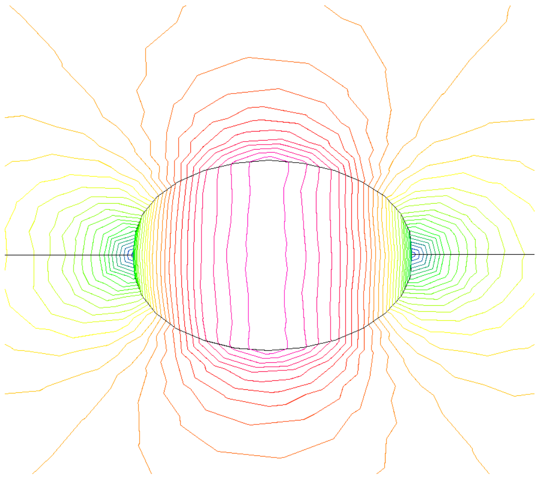
62426 cells
598 points
322 boundary pts

16467 cells
3425 points
1188 boundary pts

Computed Velocity Contours by HWENO(P_1P_2)


Coarse Grid


Medium Grid


Fine Grids

Convergence Study for different RDG methods

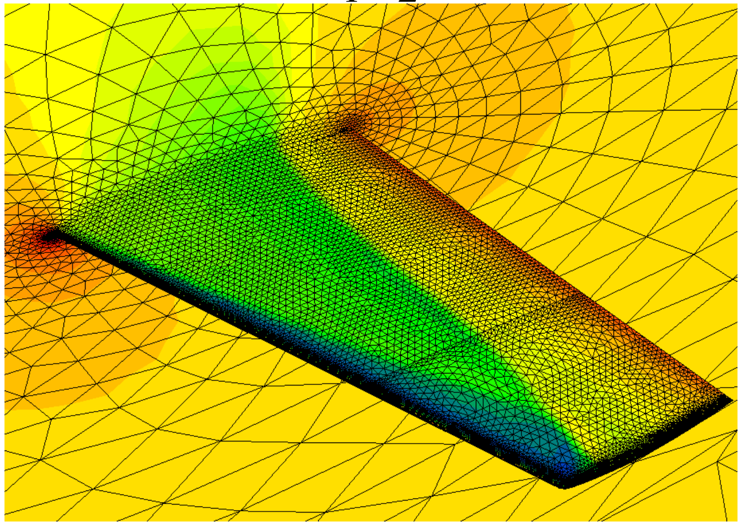
L2-error and order of convergence for the RDG(P_1P_1), WENO(P_1P_2), and HWENO(P_1P_2) methods

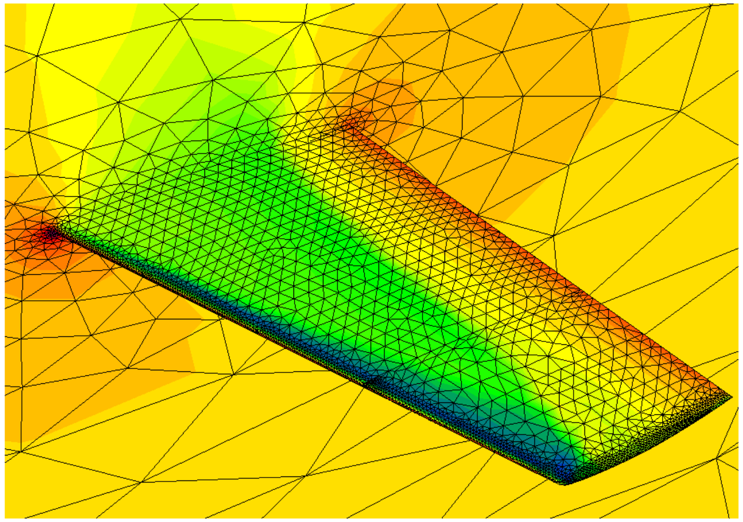
	RDG(P_1P_1)		WENO(P_1P_2)		HWENO(P_1P_2)	
Length scale	L^2-error	Order	L^2-error	Order	L^2-error	Order
7.760E-2	1.783E-2		1.052E-2		1.117E-2	
4.688E-2	5.010E-3	2.519	1.317E-3	4.124	1.503E-3	3.980
2.476E-2	1.232E-3	2.198	1.978E-4	2.964	2.201E-4	3.009

Both WENO(P_1P_2) and HWENO(P_1P_2) deliver the designed 3rd order of convergence !!

Example 5. Transonic Flow past an ONERA M6 Wing (M_∞=0.84, α=3.06°)

- Access the accuracy and non-oscillatory property of the HWENO(P_1P_2) method for flows with discontinuities.

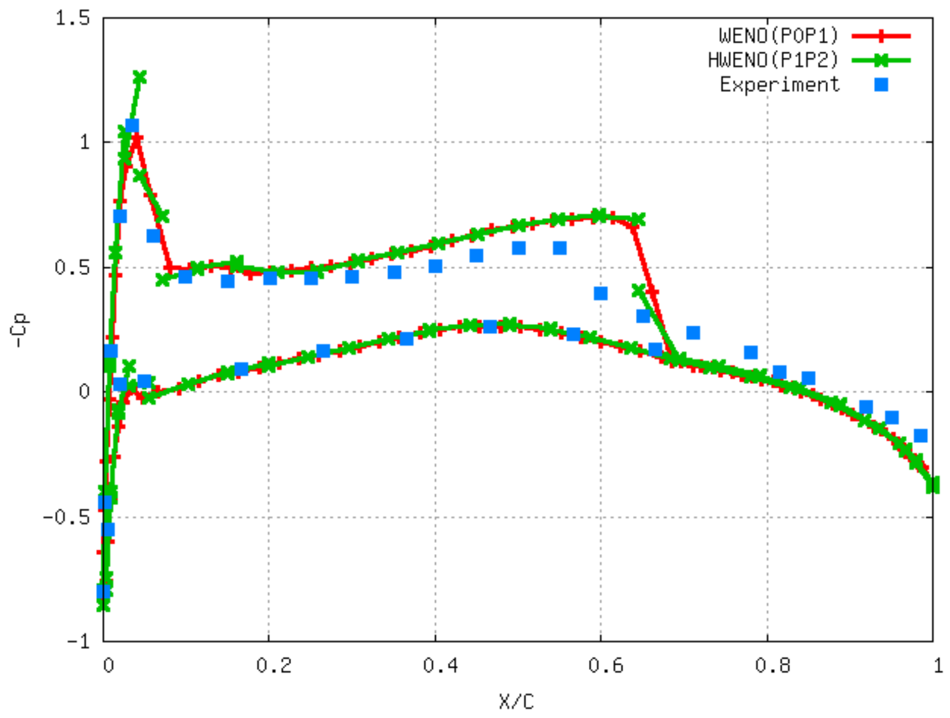


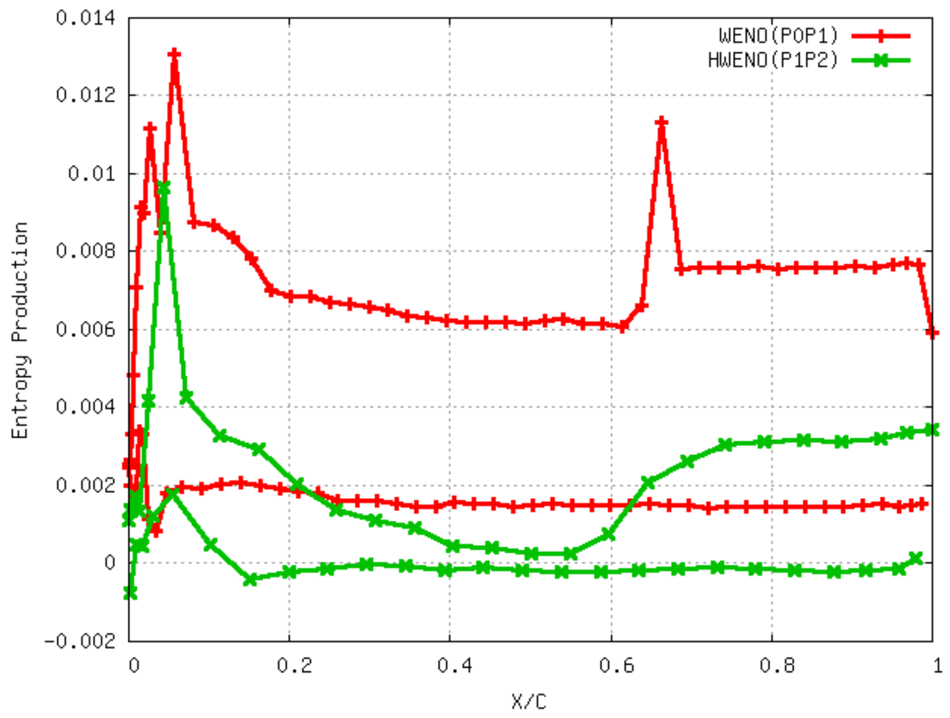


Computed Pressure Contours

WENO(P_0P_1)	HWENO(P_1P_2)
nelem = 593,169	nelem = 95,266
npoin = 110,282	npoin = 18,806
nboun = 19,887	nboun = 5,287

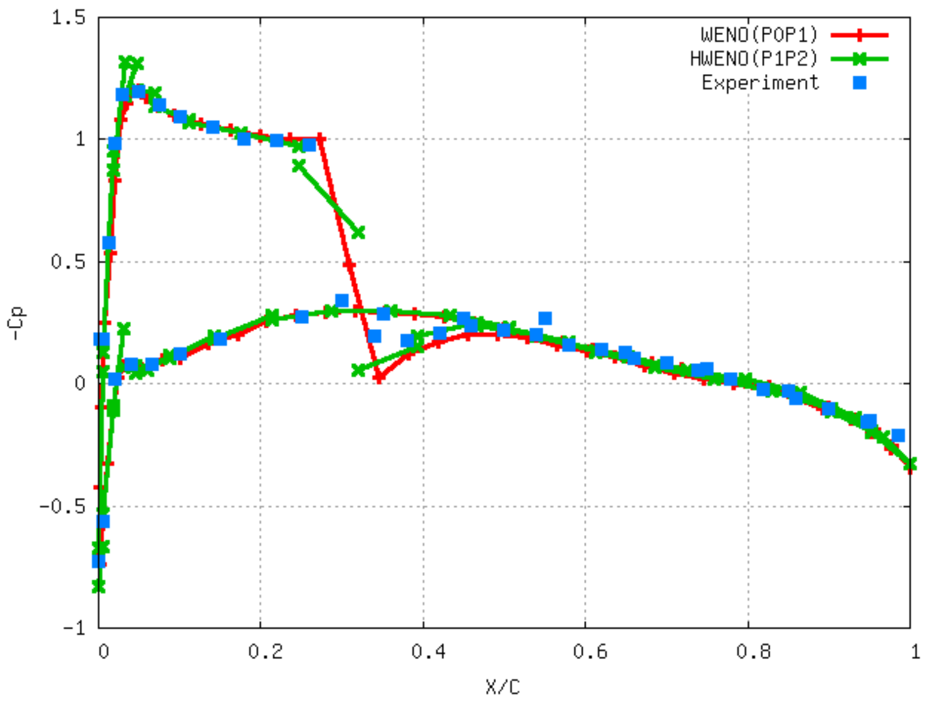
Computed Pressure Coefficient and Entropy Production Distributions at different spanwise locations

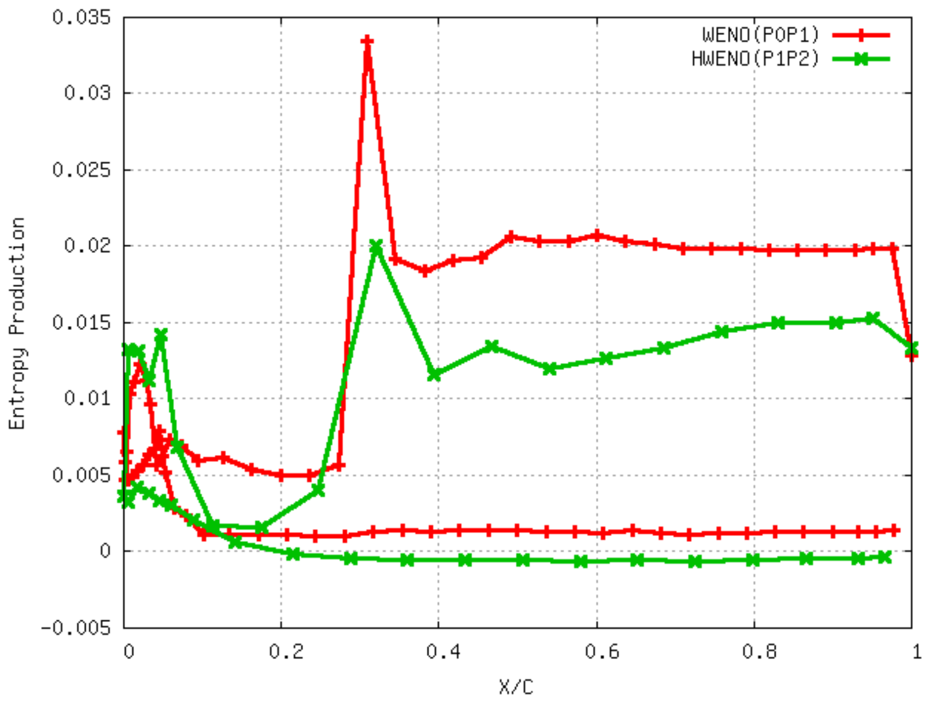




η=0.20

Computed Pressure Coefficient and Entropy Production Distributions at different spanwise locations

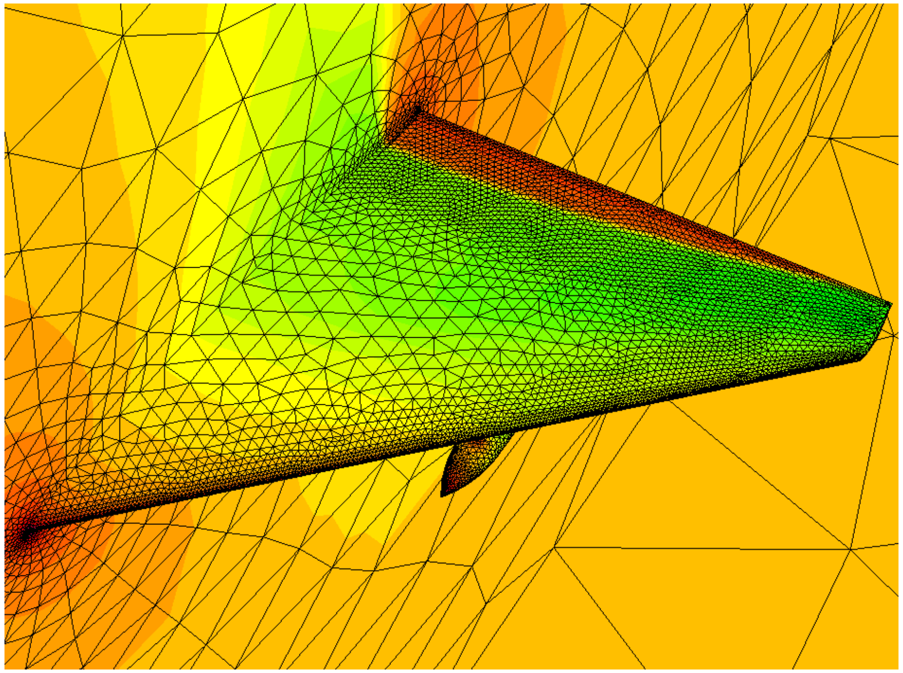
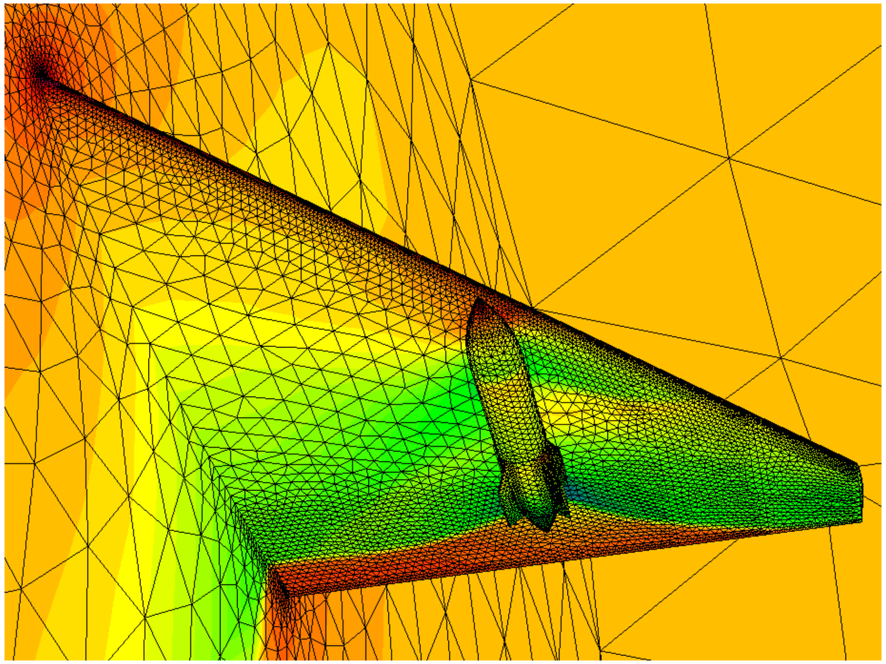




η=0.90

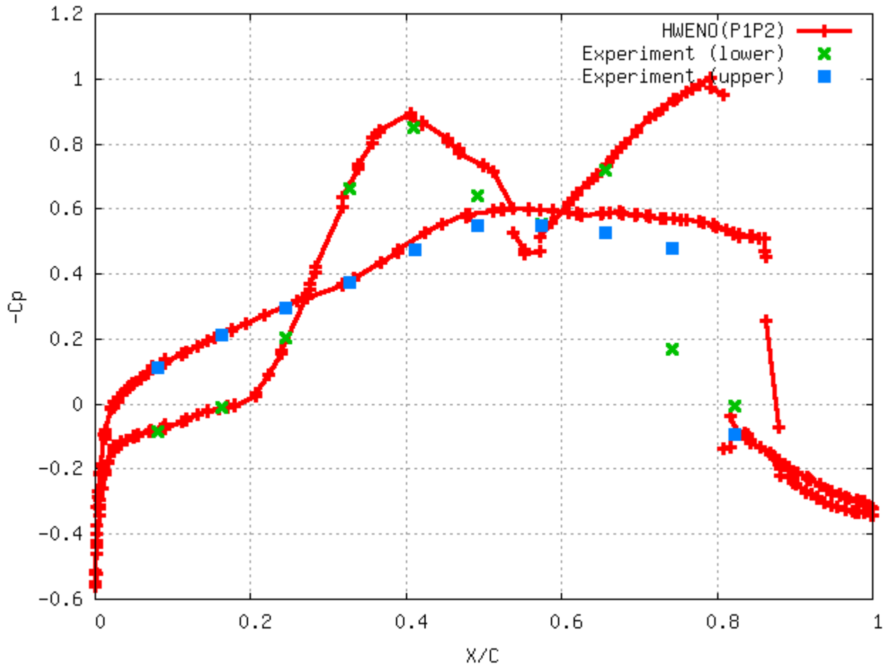
Example 6. Transonic Flow past past a Wing/Pylon/Finned-Store Configuration (M_∞=0.95, α=0°)

- Access the accuracy and non-oscillatory property of the HWENO(P_1P_2) method for flows with strong discontinuities.

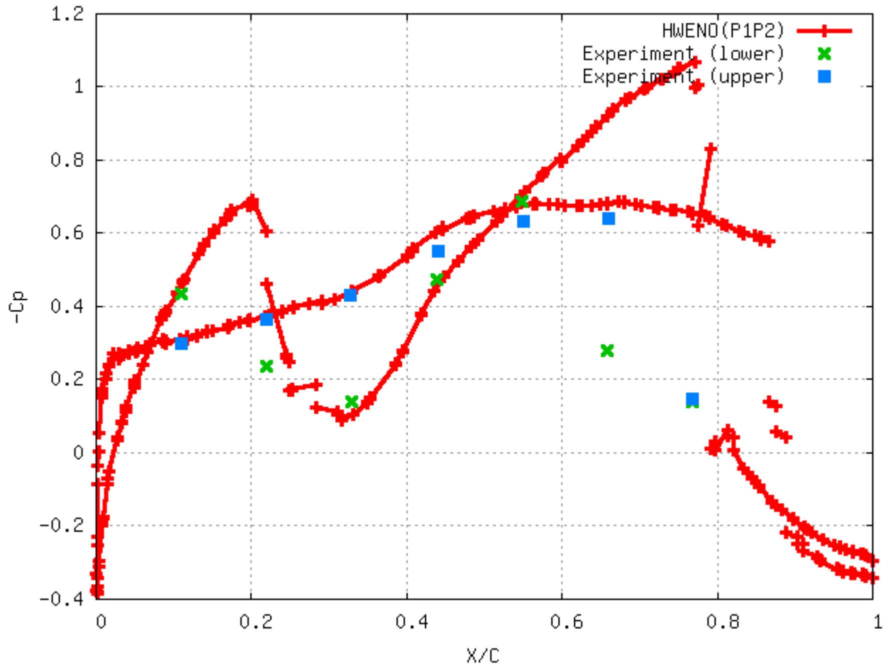



Computed Pressure Contours
(nelem=319,134, npoin=61,075, nboun=14,373)

Computed Pressure Coefficient Distributions at different spanwise locations

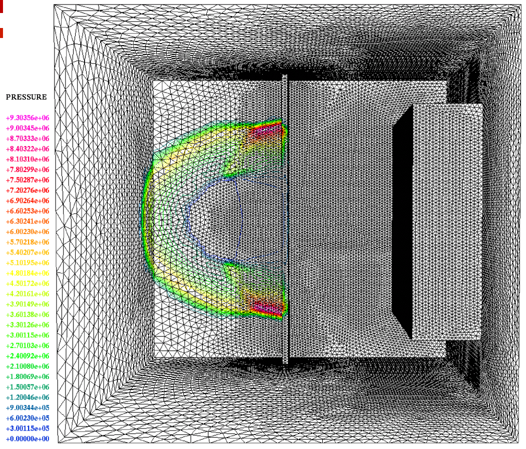


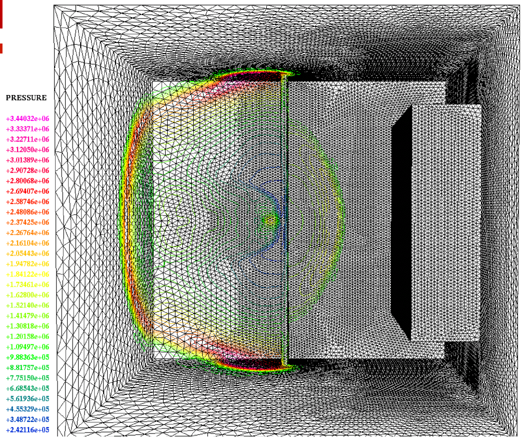
η=0.4077

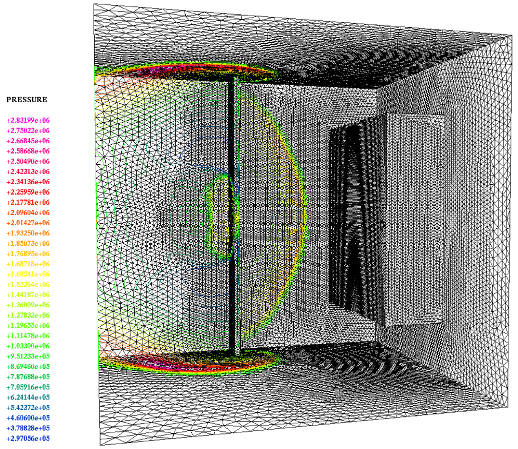


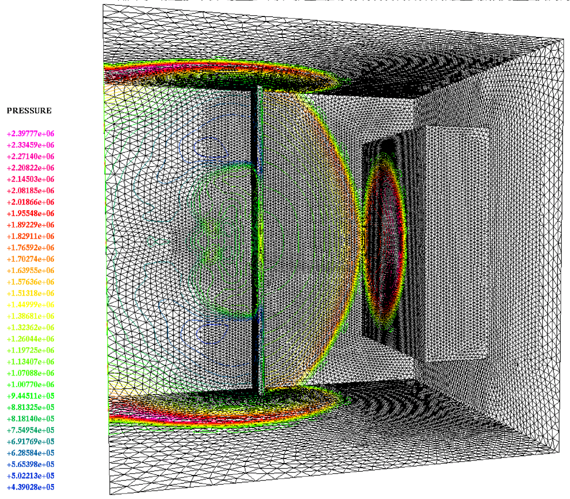
η=0.51

Example 7. An air blast wave past a wall





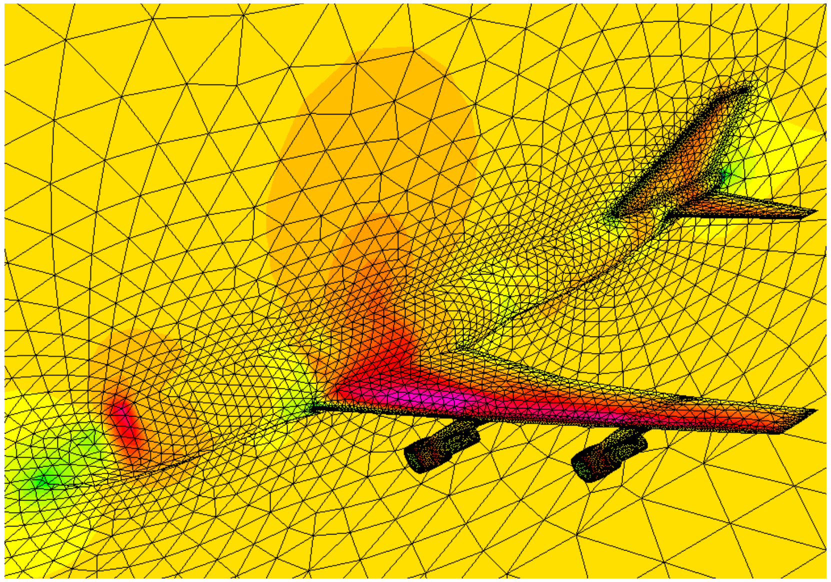




This example is presented to demonstrate that the RDG method can be used for solving problems of practical interests for engineering-type configurations.

Example 8. Transonic flow past a B747 configuration

- Demonstrate that the HWENO(P_1P_2) method can be used for computing complicated flows of practical interest.
- Flow condition: M_∞=0.85, α=2°



(nelem = 253,577, npoin = 48,851, nboun = 11,802)

Computed Mach Number Contours

Concluding Remarks

- A reconstructed discontinuous Galerkin method based on a Hierarchical WENO reconstruction, HWENO(P_1P_2) has been developed for computing shock waves on hybrid grids.

- The HWENO(P_1P_2) method is able to provide sharp resolution of shock waves essentially without over- and under-shoots for discontinuities and achieve the designed third-order of accuracy for smooth flows.

- RDG methods have the potential to provide a superior alternative to the traditional FV methods, and to become a main choice for the next generation of CFD codes.

Thank you !

DG Methods for Elliptical Problems

- The main advantages of DGMs in dealing with hyperbolic equations do not come into play when considering purely elliptic problems.
- However, using greater flexibility of DGMs which is attained by not requiring continuity in inter-element boundaries may prove advantageous.
- The potential of DGMs in the context of the Navier-Stokes equations is still worth exploring.

DG Method for Elliptical Problems

- Interior Penalty (IP) methods
- Local Discontinuous Galerkin (LDG) methods
- Bassi-Rebay Methods
- Recovery Methods

- J. Douglas, Jr. and T. Dupont, Interior penalty procedures for elliptic and parabolic Galerkin Method, Lectures Notes in Physics 58, Springer Verlag, Berlin, 1976.
- D. N. Arnold, An interior penalty finite element method with discontinuous element, SIAM J Numer. Anal. Vol. 19, pp. 742-760, 1982.
- B. Cockburn and C. W. Shu, The local discontinuous Galerkin method for time-dependent convection diffusion problems, SIAM J Numer Anal., Vol. 35, pp. 2440-2463, 1998.
- F. Bassi and S. Rebay, High-order accurate discontinuous finite element method for the numerical solution of the compressible Navier-Stokes equations, J Comp Phys , Vol. 131, pp. 267-279, 1997.
- B. van Leer and M. Lo, A Discontinuous Galerkin Method for Diffusion Based on Recovery, AIAA-2007-4083, 2007.

DG Methods for Elliptical Problems

- Consider 1-D scalar Poisson's equation using DG(P1)

$$-\frac{d^2u}{dx^2} = f$$

- The DG formulation leads

Find $u_h \in V_h^p$ such as

$$\sum_{\Omega_i \in T_h} \left(-\frac{du_h}{dx} v_h \Big|_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} + \int_{\Omega_i} \frac{du_h}{dx} \frac{dv_h}{dx} dx \right) = \sum_{\Omega_i \in T_h} \int_{\Omega_i} f v_h dx \quad \forall v_h \in V_h^p$$

- The value of $\frac{du_h}{dx}$ at the interface is not unique and need to be defined.

DG Methods for Elliptical Problems

- A natural choice is to use central flux, as no upwind mechanism is provided.

$$\left.\frac{du_h}{dx}\right|_{x_{i-\frac{1}{2}}} = \frac{1}{2}\left(\left.\frac{du_h}{dx}\right|_{x_{i-\frac{1}{2}}}^{L} + \left.\frac{du_h}{dx}\right|_{x_{i-\frac{1}{2}}}^{R}\right)$$

- The DG formulation becomes

Find $u_h \in V_h^p$ such as

$$\sum_{\Omega_i \in T_h}\left(-\frac{1}{2}\left(\frac{du_h^L}{dx} + \frac{du_h^R}{dx}\right)v_h\Bigg|_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} + \int_{\Omega_i}\frac{du_h}{dx}\frac{dv_h}{dx}dx\right) = \sum_{\Omega_i \in T_h}\int_{\Omega_i} f v_h dx \quad \forall v_h \in V_h^p$$

- The problem: Scheme is not consistent !!!

Inconsistency

- Examine Laplace's equation with homogeneous Dirichlet BCs.

$$-\frac{d^2u}{dx^2} = 0 \qquad \text{on } [a, b]$$

$$u(a) = u(b) = 0$$

- The exact solution is $u(x) = 0$.

- $\frac{du_h}{dx} = 0$ = 0 everywhere, discrete equations satisfied exactly regardless of magnitude of u_h

First order system approach

- Background: A central flux leads to an unstable scheme.
- Why? It does not take into account the fact that a DG solution is discontinuous at the cell interfaces.
- How to solve this issue ? Transform the second order equation into a first-order system of equations, as DG is naturally suitable for the first-order system of equations
- Consider the following Laplace's equation

$$-\Delta u = f$$

- This equation can be rewritten as a a first-order system of equations by introducing an auxiliary vector variable **q**,

$$-\nabla \bullet \mathbf{q} = f$$

$$\mathbf{q}\text{-}\nabla u = 0$$

First order system approach

- Application of the DG formulation leads

Find $(u_h, \mathbf{q}_h) \in V_h^p \mathrm{x} \mathbf{W}_h^p$ such as

$$\int_{\Omega_i} \mathbf{q}_h \bullet \nabla v_h d\Omega - \int_{\Gamma_i} \mathbf{q}_h \bullet \mathbf{n} v_h d\Gamma = \int_{\Omega_i} f v_h dx \quad \forall v_h \in V_h^p$$

$$\int_{\Omega_i} \mathbf{q}_h \bullet \mathbf{w}_h d\Omega + \int_{\Omega_i} u_h \nabla \bullet \mathbf{w}_h d\Omega - \int_{\Gamma_i} u_h \mathbf{w}_h \bullet \mathbf{n} d\Gamma = 0 \quad \forall \mathbf{w}_h \in \mathbf{W}_h^p$$

$$\int_{\Omega_i} \mathbf{q}_h \bullet \mathbf{w}_h d\Omega + \int_{\Omega_i} u_h \nabla \bullet \mathbf{w}_h d\Omega - \sum_j \int_{\Gamma_{ij}} u_h \mathbf{w}_h \bullet \mathbf{n} d\Gamma = 0 \quad \forall \mathbf{w}_h \in \mathbf{W}_h^p$$

First-order system approach

- Need to choose u_h at the interface
- No upwind mechanism -> Choose central flux,

$$u_h\big|_{\Gamma_{ij}} = \frac{1}{2}\left(u_h\big|_{\partial\Omega_i} + u_h\big|_{\partial\Omega_j}\right)$$

$$= u_h\big|_{\partial\Omega_i} + \frac{1}{2}\left(u_h\big|_{\partial\Omega_j} - u_h\big|_{\partial\Omega_i}\right) = u_h\big|_{\partial\Omega_i} + \frac{1}{2}\overset{\text{Jump operator}}{[\![u_h]\!]_{\Gamma_{ij}}}$$

$$\int_{\Omega_i} \mathbf{q}_h \bullet \mathbf{w}_h d\Omega + \int_{\Omega_i} u_h \nabla \bullet \mathbf{w}_h d\Omega - \sum_j \int_{\Gamma_{ij}} u_h \mathbf{w}_h \bullet \mathbf{n} d\Gamma = 0 \quad \forall \mathbf{w}_h \in \mathbf{W}_h^p$$

$$\Big\downarrow$$

$$\int_{\Omega_i} \mathbf{q}_h \bullet \mathbf{w}_h d\Omega + \int_{\Omega_i} u_h \nabla \bullet \mathbf{w}_h d\Omega$$

$$- \sum_j \int_{\Gamma_{ij}} u_h\big|_{\partial\Omega_i} \mathbf{w}_h \bullet \mathbf{n} d\Gamma - \sum_j \frac{1}{2} \int_{\Gamma_{ij}} [\![u_h]\!]_{\Gamma_{ij}} \mathbf{w}_h \bullet \mathbf{n} d\Gamma = 0 \quad \forall \mathbf{w}_h \in \mathbf{W}_h^p$$

First-order system approach

- Define a global lifting operator δ as

$$\int_{\Omega_i} \boldsymbol{\delta} \bullet \mathbf{w}_h d\Omega + \sum_j \int_{\Gamma_{ij}} \frac{1}{2} [\![u_h]\!]_{\Gamma_{ij}} \mathbf{w}_h \bullet \mathbf{n} d\Gamma = 0 \quad \forall \mathbf{w}_h \in \mathbf{W}_h^p$$

$$\int_{\Omega_i} \mathbf{q}_h \bullet \mathbf{w}_h d\Omega + \int_{\Omega_i} u_h \nabla \bullet \mathbf{w}_h d\Omega$$

$$- \sum_j \int_{\Gamma_{ij}} u_h \big|_{\partial\Omega_i} \mathbf{w}_h \bullet \mathbf{n} d\Gamma - \sum_j \int_{\Gamma_{ij}} \frac{1}{2} [\![u_h]\!]_{\Gamma_{ij}} \mathbf{w}_h \bullet \mathbf{n} d\Gamma = 0 \quad \forall \mathbf{w}_h \in \mathbf{W}_h^p$$

$$\Big\downarrow$$

$$\mathbf{q}_h = \nabla u_h - \boldsymbol{\delta}$$

Bassi-Rebay 1

- Compute the lift-operator:

  Find $\boldsymbol{\delta} \in \mathbf{W}_h^p$ such as

$$\int_{\Omega_i} \boldsymbol{\delta} \bullet \mathbf{w}_h \, d\Omega + \sum_j \int_{\Gamma_{ij}} \frac{1}{2} [\![u_h]\!]_{\Gamma_{ij}} \mathbf{w}_h \bullet \mathbf{n} \, d\Gamma = 0 \quad \forall \mathbf{w}_h \in \mathbf{W}_h^p$$

- The primal form is then given by

$$\int_{\Omega_i} (\nabla u_h - \boldsymbol{\delta}) \bullet \nabla v_h \, d\Omega - \sum_j \int_{\Gamma_{ij}} (\nabla u_h - \boldsymbol{\delta}) \bullet \mathbf{n} v_h \, d\Gamma = \int_{\Omega_i} f v_h \, dx \quad \forall v_h \in V_h^p$$

$$\Big\downarrow$$

$$\int_{\Omega_i} (\nabla u_h - \boldsymbol{\delta}) \bullet \nabla v_h \, d\Omega - \sum_j \int_{\Gamma_{ij}} \langle\!\langle (\nabla u_h - \boldsymbol{\delta}) \bullet \mathbf{n} \rangle\!\rangle v_h \, d\Gamma = \int_{\Omega_i} f v_h \, dx \quad \forall v_h \in V_h^p$$

Average operator

Bassi-Rebay 1

- Unstable for pure elliptic problems
- Stencil no longer compact

Bassi-Rebay 2

- Define a local lift operator $\boldsymbol{\delta}_l$ for each interface or boundary face as

  Find $\boldsymbol{\delta}_l \in \mathbf{W}_h^p$ such as

  $$\int_{\Omega_i} \boldsymbol{\delta}_l \bullet \mathbf{w}_h d\Omega + \int_{\Gamma_{ij}} \frac{1}{2} [\![u_h]\!]_{\Gamma_{ij}} \mathbf{w}_h \bullet \mathbf{n} d\Gamma = 0 \quad \forall \mathbf{w}_h \in \mathbf{W}_h^p$$

- The local lift operator $\boldsymbol{\delta}_l$ is one interface (and boundary) contributions to the global lift operator $\boldsymbol{\delta}$.

- Replace the global lift operator in the interface integral by the local lift operator.

Bassi-Rebay 2

- The DG formulation in primal form becomes

$$\int_{\Omega_i}(\nabla u_h - \boldsymbol{\delta})\bullet\nabla v_h d\Omega - \sum_j \int_{\Gamma_{ij}} \langle\langle(\nabla u_h - \boldsymbol{\delta}_l)\bullet\mathbf{n}\rangle\rangle v_h d\Gamma = \int_{\Omega_i} f v_h dx \quad \forall v_h \in V_h^p$$

- Compact !!!

- Stability can be established (proven) if a stabilization parameter η (>3) is used in the local lift operator.

Find $\boldsymbol{\delta}_l \in \mathbf{W}_h^p$ such as

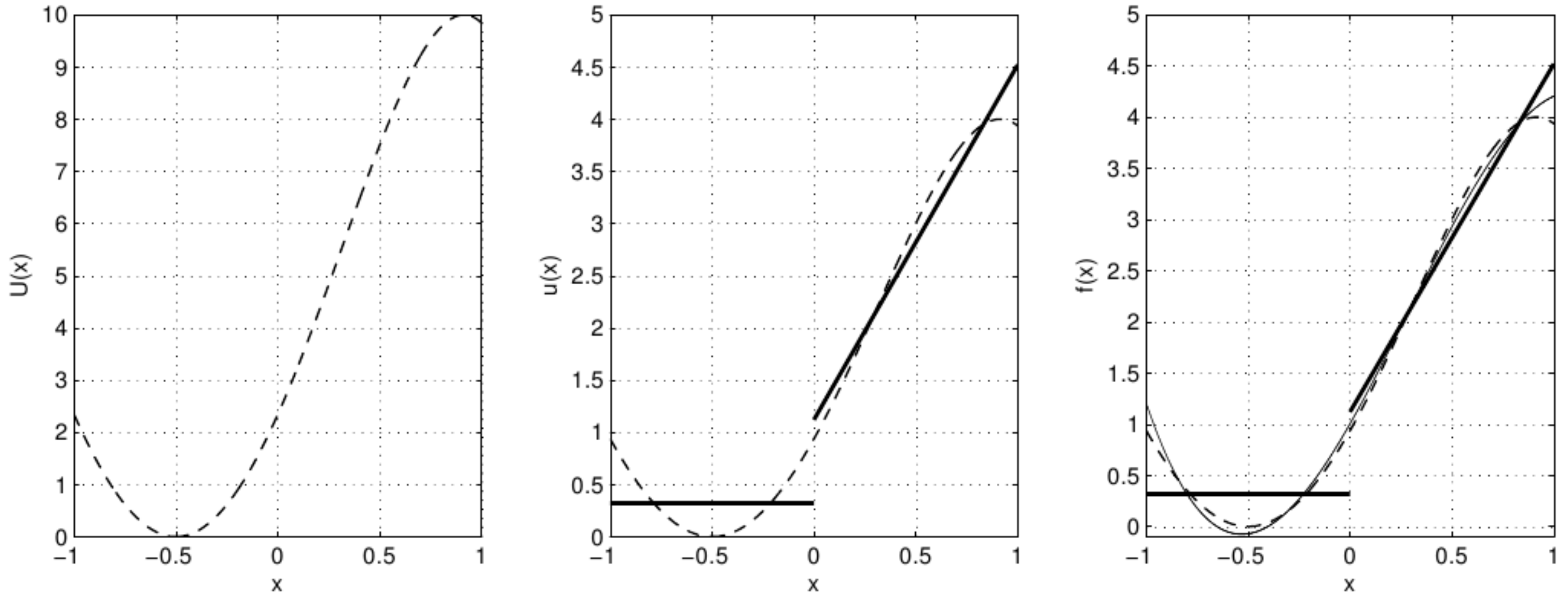
$$\int_{\Omega_i}\boldsymbol{\delta}_l \bullet \mathbf{w}_h d\Omega + \eta\int_{\Gamma_{ij}}\frac{1}{2}[\![u_h]\!]_{\Gamma_{ij}}\ \mathbf{w}_h \bullet \mathbf{n} d\Gamma = 0 \quad \forall \mathbf{w}_h \in \mathbf{W}_h^p$$

Recovery-based DG for Diffusion

- Idea: Obtain a continuous polynomial solution on the union of two cells that shares an interface, where the diffusive fluxes need to be defined.

- How ? The smooth solution is locally recovered, that is indistinguishable from the discontinuous discrete solution in the weak sense. (can be thought as a weak interpolation.)

Recovery-based DG for Diffusion



- Recovery in 1D, P=2. Shown are from left to right, the original quartic initial values U (dashed), the piecewise linear discretization u (bold) together with U, and the cubic recovered distribution f (thin) together with u and U, on the adjacent intervals (-1,0), and (0,1). All three distributions yield the same value when taking their inner product with either test function on either interval, making them indistinguishable in the weak sense.

Recovery-based DG for Diffusion

- Following are the formulas for recovering the smooth solution u_{ij} from two discontinuous solution u_i and u_j on the union of neighboring cells Ω_i and Ω_j :

$$\int_{\Omega_i} u_{ij} B_k^i \, d\Omega = \int_{\Omega_i} u_i B_k^i \, d\Omega, \qquad \mathrm{k} = 1, \ldots, \mathrm{N}$$

$$\int_{\Omega_j} u_{ij} B_k^j \, d\Omega = \int_{\Omega_j} u_j B_k^j \, d\Omega, \qquad \mathrm{k} = 1, \ldots, \mathrm{N}$$

where B^i and B^j are the basis functions for cell I and j respectively, and N is the dimension of DG space.

Recovery-based DG for Diffusion

- How to construct the basis B^{ij} defined on $\Omega_i U \Omega_j$ is trivial in 1D. (although extremely challenging in 2D/3D).

Objective

- **Development of high accurate 3rd order temporal discretization methods for RDG methods for unsteady flow simulations.**
  - to reduce the temporal discretization error,
  - for LES and DNS applications.

- **A class of implicit RK schemes – ESDIRK: Explicit first stage, Single Diagonal coefficient, diagonally Implicit Runge-Kutta**
  - Allow variable time-step size.
  - Can be constructed to be A- and L-stable for arbitrary order in time.
  - Found to be more efficient in terms of computational cost for a given accuracy level as compared to the lower-order implicit schemes.

Time Integration

Semi-discrete system of nonlinear equations:

$$M\frac{dU}{dt} = R(U)$$

M: Mass matrix

U: Global vector of unknown conservative variables

R: Residual vector

Time integration schemes: explicit method

- Explicit three-stage third-order TVD Runge-Kutta Scheme:

$$\mathbf{U}^{(1)} = \mathbf{U}^n + \Delta t\, M^{-1}\, \mathbf{R}(\mathbf{U}^n)$$

$$\mathbf{U}^{(2)} = 3/4\mathbf{U}^n + 1/4[\mathbf{U}^{(1)} + \Delta t\, M^{-1}\, \mathbf{R}(\mathbf{U}^{(1)})]$$

$$\mathbf{U}^{n+1} = 1/3\mathbf{U}^n + 2/3[\,\mathbf{U}^{(2)} + \Delta t M^{-1}\mathbf{R}(\mathbf{U}^{(2)})]$$

This method is linearly stable for a Courant number less than or equal to $1/(2p+1)$.

TVDRK scheme is not efficient, when the maximum allowable time step imposed by an explicit stability requirement is much smaller than that imposed by the acceptable level of time accuracy.

Time integration schemes: implicit method

- **The *m*-stage ESDIRK scheme:**

(i) $\mathbf{U}^{(1)} = \mathbf{U}^n$

(ii) For $s = 2, \ldots, m$

$$\mathbf{U}^{(s)} = \mathbf{U}^n + \Delta t \sum_{j=1}^{s} a_{sj} M^{-1} \mathbf{R}(\mathbf{U}^{(j)})$$

(iii) $\mathbf{U}^{n+1} = \mathbf{U}^{(m)}$

where a_{sj} are the Butcher coefficient of the scheme. The Butcher table for the 3rd-order ESDIRK3 scheme (m=4) is listed below (the values are given in Appendix):

c_1=0	a_{11} (=0)	0	0	0
c_2	a_{21}	$a_{22}=a_{44}$	0	0
c_3	a_{31}	a_{32}	$a_{33}=a_{44}$	0
c_4=1	$a_{41}=b_1$	$a_{42}=b_2$	$a_{43}=b_3$	a_{44}
$\mathbf{U}^{n+1}$	b_1	b_2	b_3	b_4

The *m*-stage ESDIRK scheme

- The first stage is explicit since $a_{11}=0$.
- A system of non-linear equation is solved at each individual stage since the set of a_{sj} has the form of a lower triangular matrix.
- The solution at the last stage is the solution at the next time step, and c_s represents the point in the time interval $[t, t+\Delta t]$, and satisfies

$$c_s = \sum_{j=1}^{s} a_{sj} \qquad s = 1, 2, 3, \ldots, 4$$

- Note that ESDIRK2 (m=2) is nothing but the 2nd-order Crank-Nicholson scheme (CN2).

Clearly, how to devise an efficient method for the solution of the non-linear system of the equations is crucial to the success of the ESDIRK scheme.

References

- Xia, Y., Luo, H., Frisbey, M., and Nourgaliev, R., A Set of Parallel, Implicit Methods for a Reconstructed Discontinuous Galerkin Method for the Compressible Flows on 3D Arbitrary Grids, **Computers & Fluids**., 2014, http://dx.doi.org/10.10.16/j.compfluid.2014.01.023

- Xia, Y., Luo, H., and Nourgaliev, R., An implicit Hermite WENO reconstruction-based discontinuous Galerkin method on tetrahedral grids, **Computers & Fluids**., 2014, http://dx.doi.org/10.1016/j.compfluid.2014.02.027.

- Xia Y, Luo, H., and Nourgaliev, R. An Implicit Reconstructed Discontinuous Galerkin Method Based on Automatic Differentiation for the Navier-Stokes Equations on Tetrahedron Grids, AIAA-2013-0687, 51st AIAA Aerospace Sciences Meeting, Grapevine, Texas, Jan. 7-10, 2013.

- Luo, H., Segawa, H., and Visbal, M.R., An Implicit Discontinuous Galerkin Method for the Unsteady Compressible Navier-Stokes Equations, **Computers & Fluids**, Vol. 53, pp. 133-144, 2012.

Computational Results

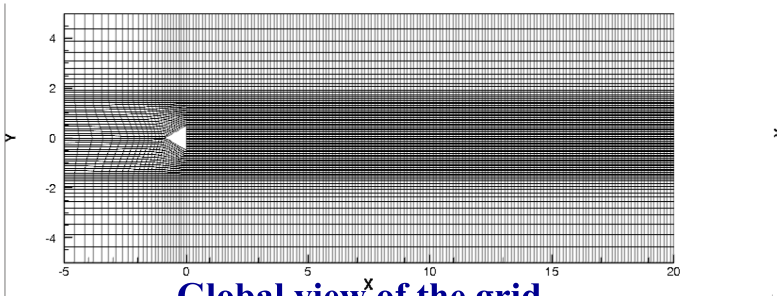
- **Numerical Examples**
  - 1. Inviscid shedding flow past a triangular wedge
  - 2. Kármán vortex street at Re = 200
  - 3. Viscous flow past an SD7003 airfoil
  - 4. Implicit large eddy simulation of a lid driven cavity

- **Default parameters for solving the pseudo-time system**
  - Linear solver: LU-SGS preconditioned GMRES algorithm
  - The pseudo time-step term is off, which is equivalent to solving a quasi-Newton system at each implicit Runge-Kutta stage
  - The relative residual tolerance is 1.0×10^{-4}.
  - The maximum iteration number is 5.

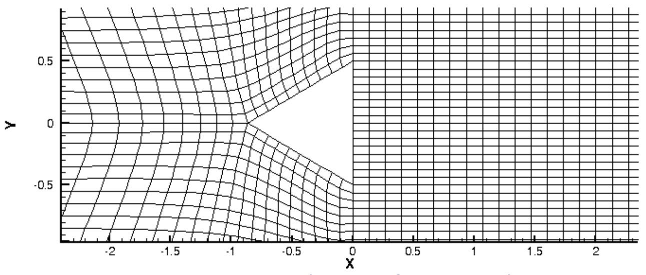
- **Compilation and runtime toolkit**
  - METIS for domain partitioning
  - PGI Fortran compiler + OpenMPI

Example 1. Inviscid shedding flow past a triangular wedg

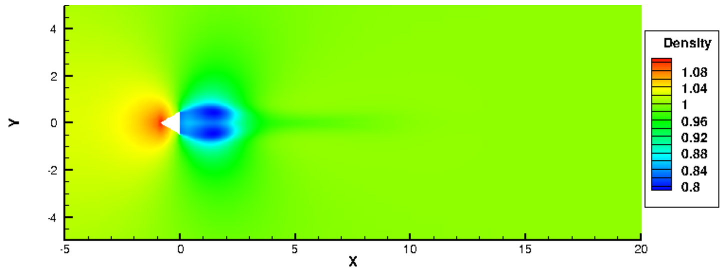
- **Objective**: illustrate the importance of the temporal discretization schemes on the accuracy of the numerical solutions
- **Grid**: 13, 250 hexahedral elements, 27, 026 grid point, and 27, 026 quadrilateral faces
- **Initial condition**: we use intermediate solution ($M_\infty = 0.5$, α=0°) obtained by DG(P0) as IC for the unsteady shedding flow



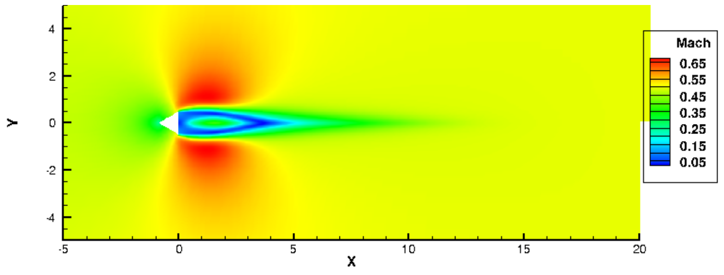
Global view of the grid



Local view of the grid



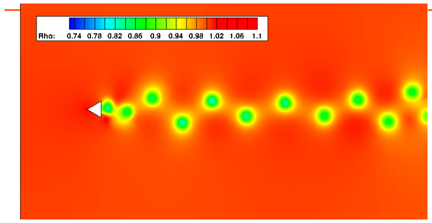
Density contour by P0 solution



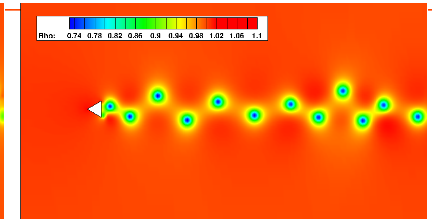
Mach number contour by P0 solution

Example 1. Inviscid shedding flow past a triangular wedg

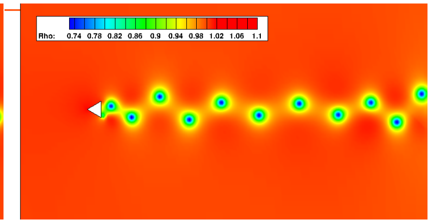
- Comparison of computed density contours at t = 400 (M_∞ = 0.5, α=0°)
  - With a fixed time-step size of dt = 0.05



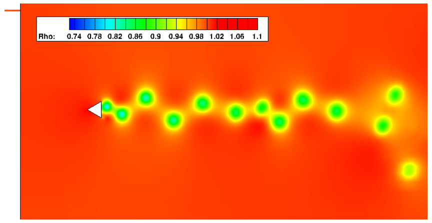
BDF1 + RDG(P1P2)



IRK2 +RDG(P1P2)

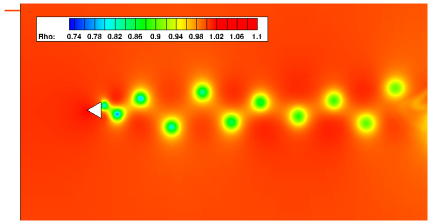


IRK3+RDG(P1P2)

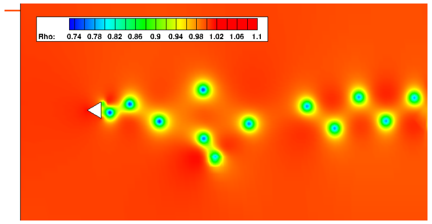


IRK3+DG(P1)

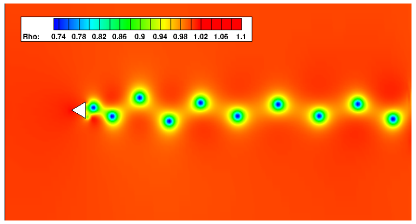
  - With a fixed time-step size of dt = 0.10



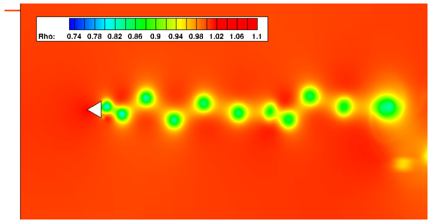
BDF1 + RDG(P1P2)



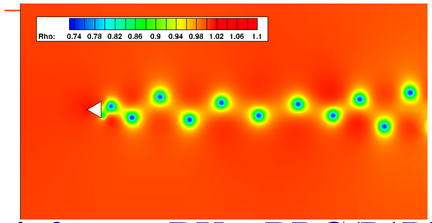
IRK2 + RDG(P1P2)



IRK3 + RDG(P1P2)



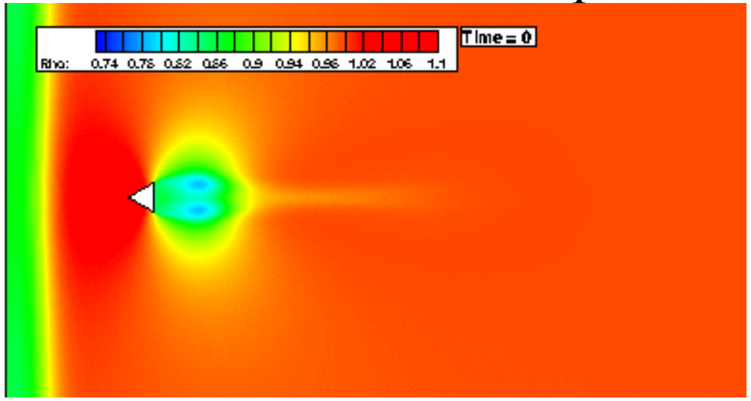
IRK3 + DG(P1)



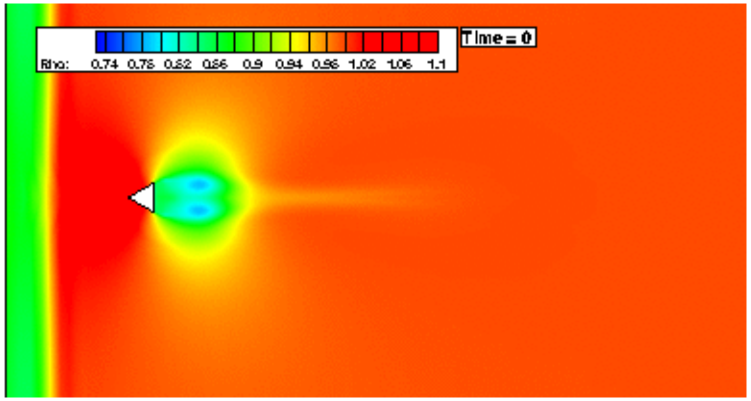
Reference solution: explicit 3-stage RK + RDG(P1P2) with a fixed dt = 0.0004

Example 1. Inviscid shedding flow past a triangular wedg

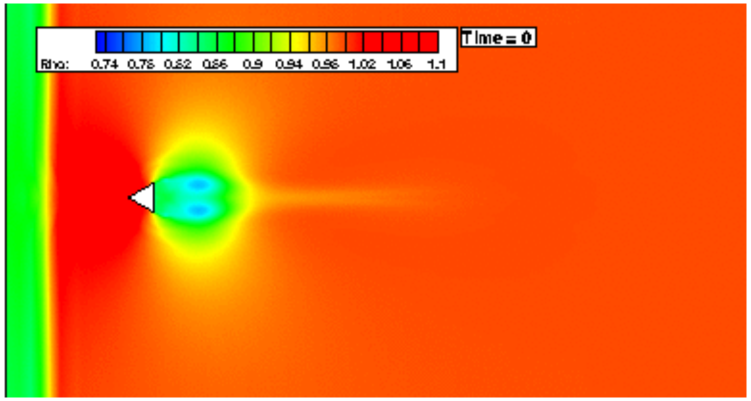
- Animations (up to solution time t = 400)
  - With a fixed time-step size of dt = 0.10



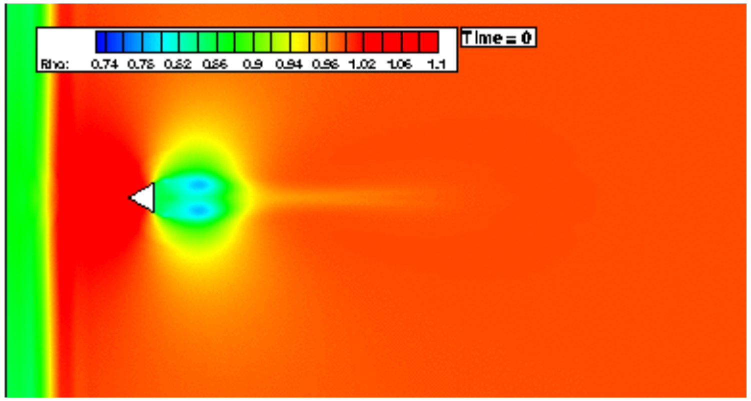
BDF1 + RDG(P1P2)



IRK3+DG(P1)



IRK2 + RDG(P1P2)



IRK3+RDG(P1P2)

Example 1. Inviscid shedding flow past a triangular wedg

- Comparison of the CPU time (evaluated by running on 64 cores) between the explicit and implicit methods.

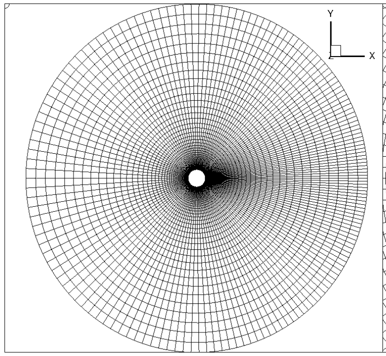
For solution at t = 40	Time-step size	Time steps	CPU time (sec)
IRK2 + RDG(P1P2)	dt = 0.05	800	1,770
IRK3 + RDG(P1P2)	dt = 0.05	800	5,182
IRK2 + RDG(P1P2)	dt = 0.10	400	1,008
IRK3 + RDG(P1P2)	dt = 0.10	400	2,825
Explicit RK3 + RDG(P1P2)	dt = 0.0004	800,000	13,498

- Performance of the LU-SGS preconditioned GMRES solver
  - In average, a drop of **4** orders of magnitude for the unsteady residual can be achieved within **5** inner iterations at each implicit RK stage

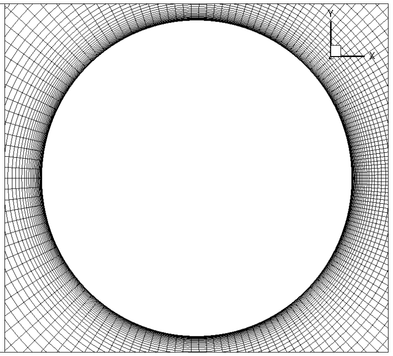
The IRK3+RDG(P1P2) method provides accurate solutions in space and time and requires much less CPU time compared with its explicit counterpart!

Example 2. Kármán vortex street at Re = 200

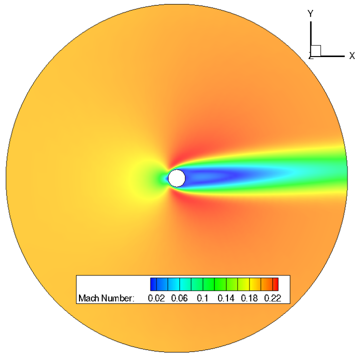
- **Grid**: 10,204 hexahedral elements, 20,800 grid points, and 20,800 boundary faces. The normal grid spacing near the cylinder surface is 0.001 (normalized by the cylinder diameter)
- **Boundary condition**: no-slip, adiabatic condition on cylinder surface, symmetry condition on spanwise wall, characteristic condition at far-field.
- **Initial condition**: we use steady-state solution ($M_\infty = 0.2$, $\alpha=3°$, Re = 50) obtained by DG(P0) as IC for the vortex shedding



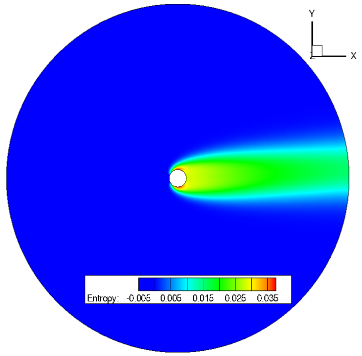
Grid: global view



Grid: local View



Mach Number

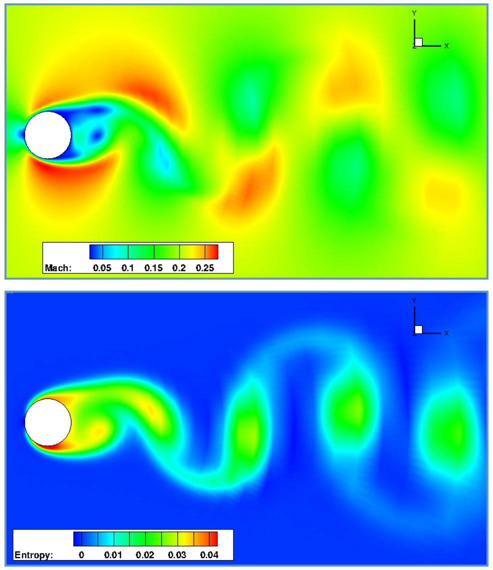


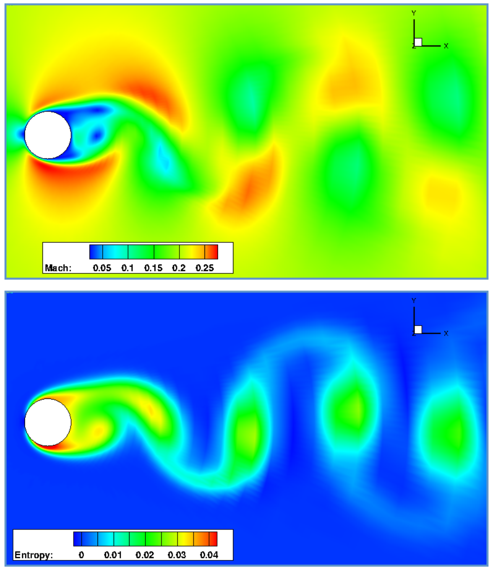
Entropy

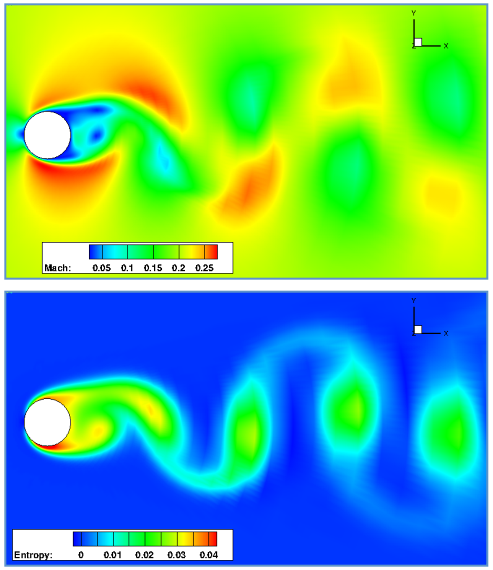
You can find the grid and report at the NASA website
http://www.grc.nasa.gov/WWW/Acoustics/code/adpac/sample/CYLINDER_VORTEX_SHEDDING/

Example 2. Kármán vortex street at Re = 200

- Comparison of the computed instantaneous Mach number and entropy contours ($M_\infty = 0.2$, $\alpha = 0°$, Re = 200)

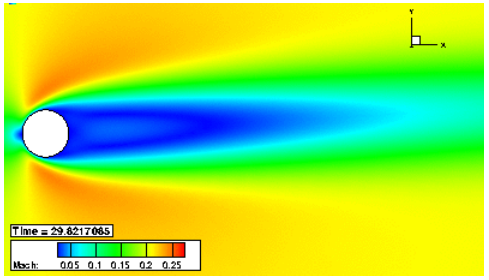


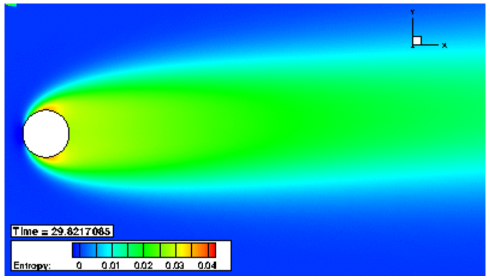




IRK2+RDG(P1P2), dt=0.05 **IRK3+RDG(P1P2), dt=0.05** **IRK3+RDG(P1P2), dt=0.5**

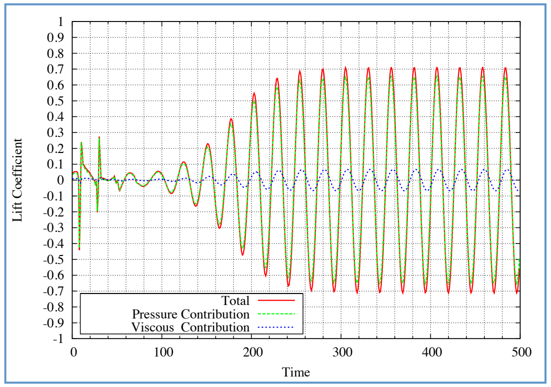
- Animations (up to solution time t = 40)

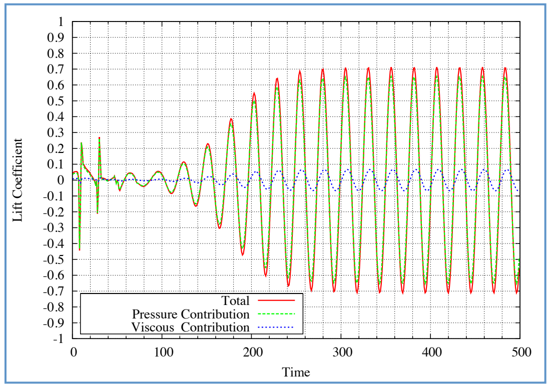


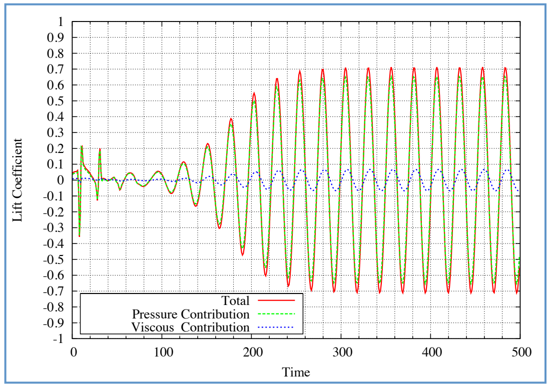


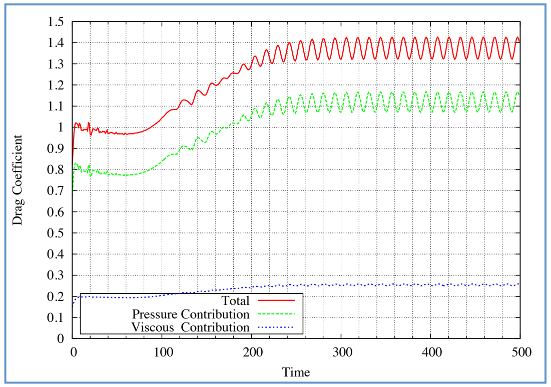
Example 2. Kármán vortex street at Re = 20

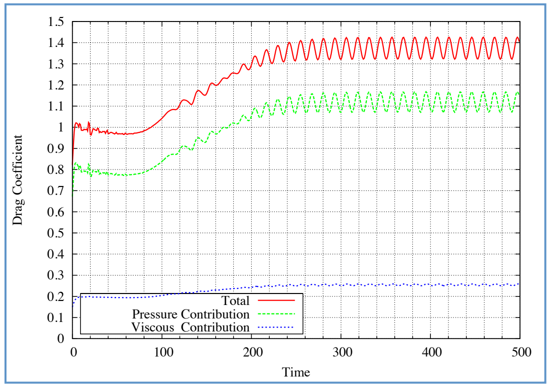
- Time histories of lift and drag coefficients (Strouhal number = 1.923)

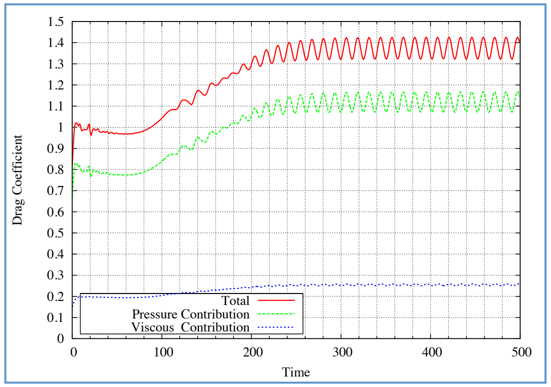












IRK2+RDG(P1P2), dt=0.05 **IRK3+RDG(P1P2), dt=0.05** **IRK3+RDG(P1P2), dt=0.5**

Agree well with the results in the referred literature!

Example 2. Kármán vortex street at Re = 200

- Comparison of the CPU time (evaluated by running on 128 cores) between the explicit and implicit methods.

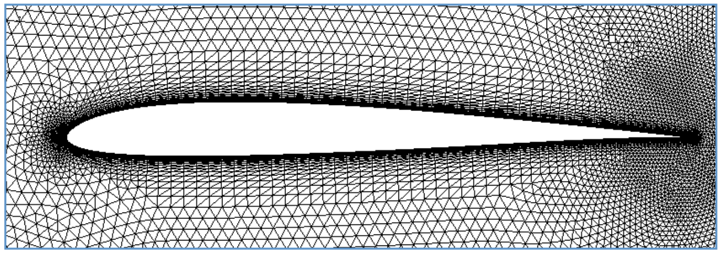
For solution at t = 40	Time-step size	Time steps	CPU time (sec)
IRK2 + RDG(P1P2)	dt = 0.05	10,000	1,603
IRK3 + RDG(P1P2)	dt = 0.05	10,000	5,524
IRK3 + RDG(P1P2)	dt = 0.50	1,000	1,047
Explicit RK3 + RDG(P1P2)	dt = 0.00005	10,000,000	Estimated 77,960

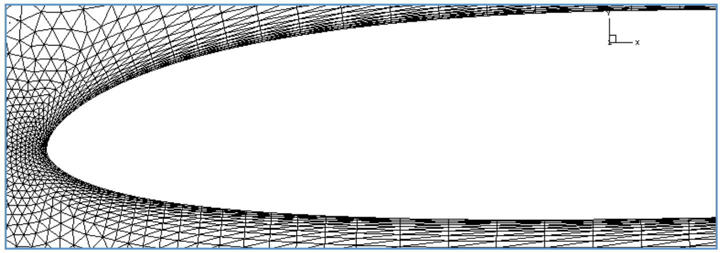
- Performance of the LU-SGS preconditioned GMRES solver
  - In average, a drop of **4** orders of magnitude for the unsteady residual can be achieved within **5** inner iterations at each implicit RK stage

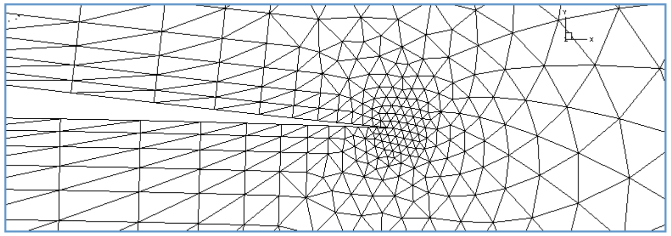
- The IRK's can greatly accelerate the solution over its explicit counterpart, while rendering accurate solution in time and space for viscous flows.
- The IRK3 enables the use of much larger time-step size and thus can improve the overall efficiency.

Example 3. Viscous flow past an SD7003 airfoil

- **Grid**: 50,781 prismatic elements, 52,176 grid points, 101,562 triangular boundary faces, and 279 quadrilateral boundary faces.
- **Boundary condition**: no-slip, adiabatic condition on the airfoil surface, symmetry condition on spanwise wall, characteristic condition on far-field.
- **Initial condition**: uniform flow ($M_\infty = 0.1$, $\alpha=4°$, $Re = 10,000$) in the field.

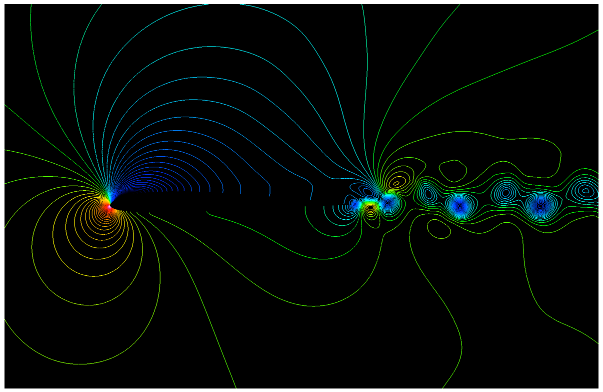

Airfoil: global view

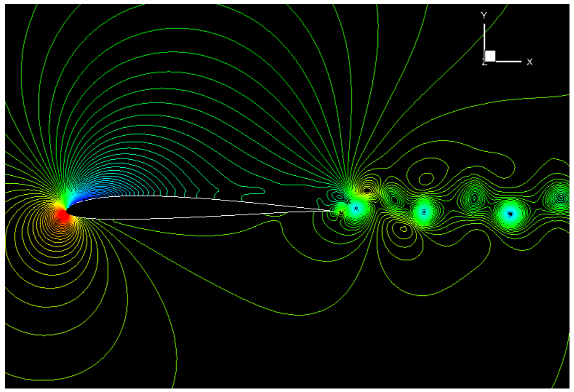

Airfoil: leading edge


Airfoil: trailing edge

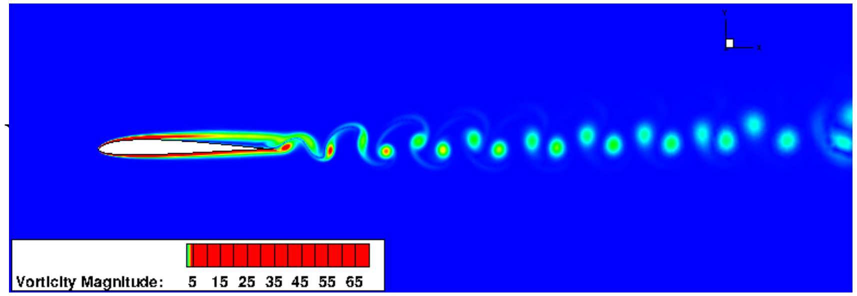
Example 3. Viscous flow past an SD7003 airfoil

- Comparison of the computed instantaneous pressure number contours


By the compact method*


By IRK3+RDG(P1P2), dt = 0.01

- Comparison of the computed



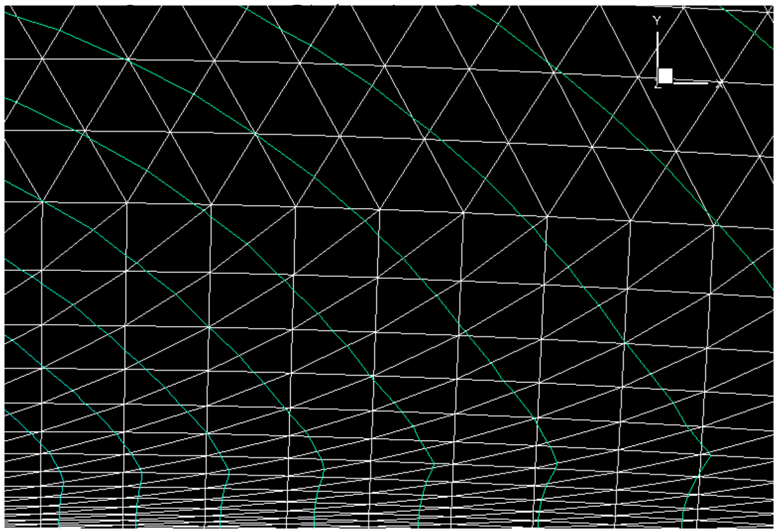
By the compact method*

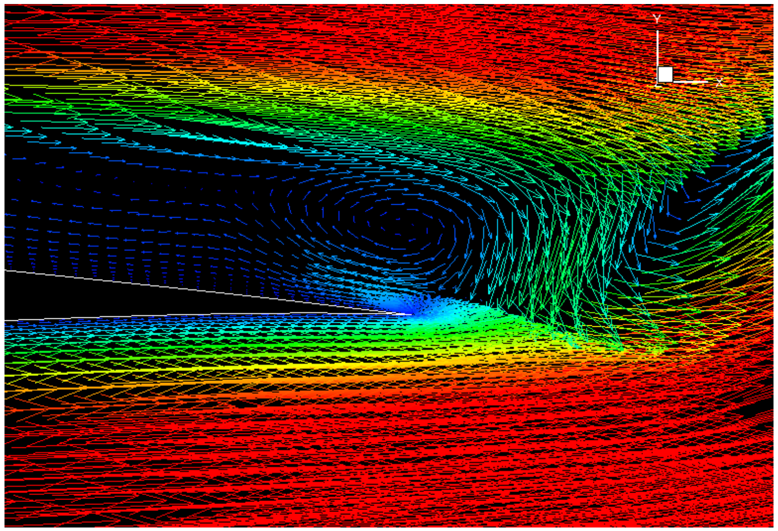
By IRK3+RDG(P1P2), dt = 0.01

* Raymond E Gordnier and Miguel R Visbal. Compact Difference Scheme Applied to Simulation of Low-Sweep Delta Wing Flow. AIAA journal, 43(8):1744–1752, 2005.

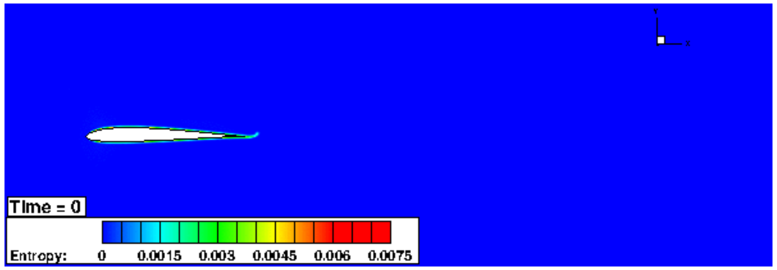
Example 3. Viscous flow past an SD7003 airfoil

- Local details of the computed instantaneous solution by

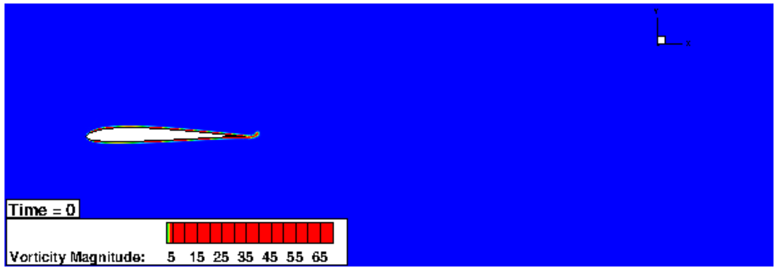

Pressure contours near the upper surface


Velocity vectors near the trailing edge

- Animations (up to solution time t = 100 with dt = 0.01 and 1 sec / frame)



Entropy contours



vorticity Magnitude contiurs

Example 3. Viscous flow past an SD7003 airfoil

- Comparison of the CPU time (evaluated by running on 256 cores) between the explicit and implicit methods.

For solution at t = 100	Time-step size	Time steps	CPU time (sec)
IRK3 + RDG(P1P2)	dt = 0.01	10,000	83,178
Explicit RK3 + RDG(P1P2)	dt = 0.00001	10,000,000	Estimated 1,669,400

A speedup factor of more than 200 by IRK3 over its explicit counterpart !

- Performance of the LU-SGS preconditioned GMRES solver
    - In average, a drop of **4** orders of magnitude for the unsteady residual can be achieved within **5** inner iterations at each implicit RK stage

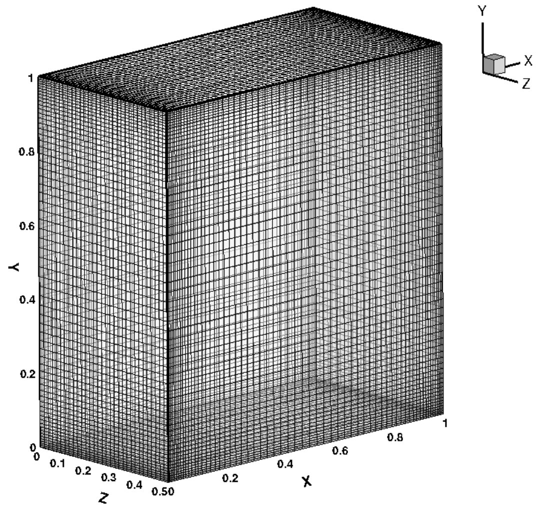
Indeed, the relative tol. = 10^{-4} is a overkill in running these problems. If we use relative tol. = 10^{-2}, even higher speedup may be achieved.

Example 4. Implicit LES of a lid driven cavit

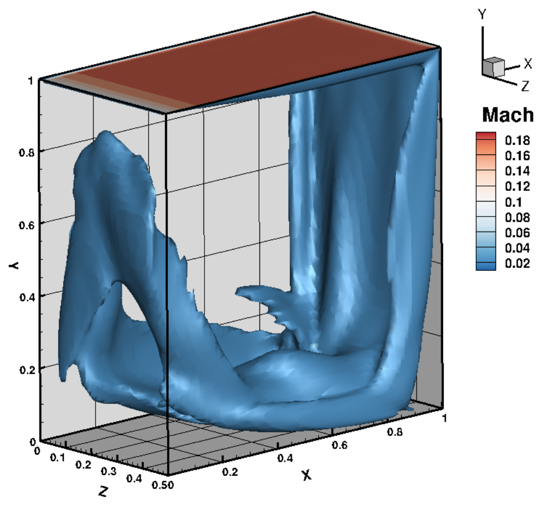
- **Implicit LES**
  - Without the use of an explicit sub-grid scale model.
- **Why DG methods?**
  - The DG methods only dissipate the scales that the model is not able to capture correctly, thus acting like a sub-grid scale model.
- **Why RDG methods?**
  - DG methods like P2, P3, and P4 have shown the ability of helping improve the solution accuracy in a few benchmark DNS and LES problems. Yet they are expensive in terms of computing time and storage requirement.
  - Assess the RDG methods like P1P2 and even P2P3 for computing large-scale.
- **Why 3D lid driven cavity?**
  - The 3D lid driven cavity presents complex physical phenomena, though the geometry is simple. Therefore it is an adequate example to assess the performance of the implicit LES with the developed methods.

Example 4. Implicit LES of a lid driven cavit

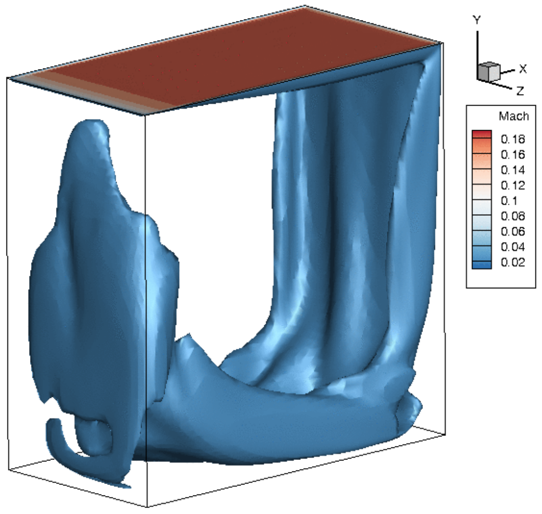
- **Problem description**
  - Domain: x = [0, 1], y = [0, 1], and z = [-0.25, 0.25] (x: y: z = 1: 1: 0.5).
  - Top lid velocity $\mathbf{v}_b$ = (0.2, 0, 0), Re = 10,000.
  - No-slip, adiabatic conditions for the rest of boundary walls.
  - Grid: 64x64x32 grid points; h_{min} = 0.005 in x-y plane (y^+ = 3.535); uniform grid distribution in spanwise z-direction.



The 64x64x32 grid



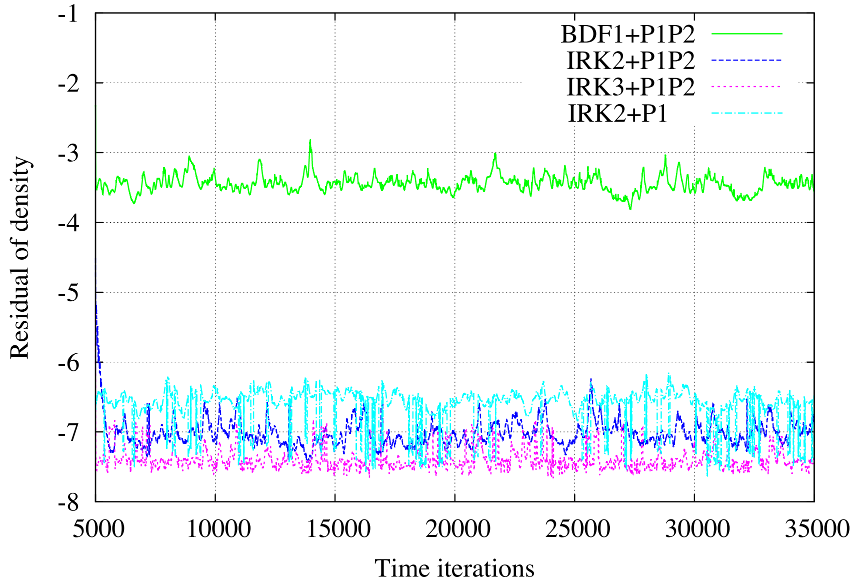
Instantaneous Mach No. iso-surface



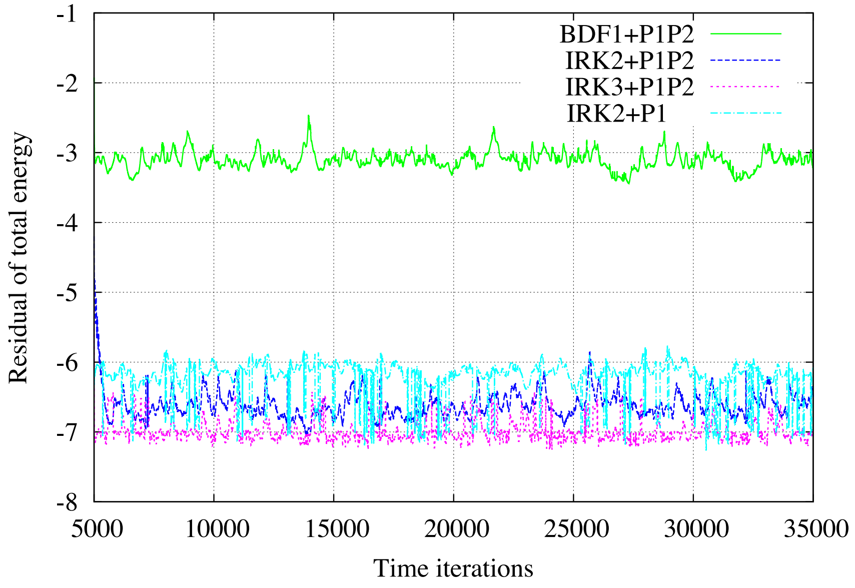
Animated Mach No. iso-surface

Example 4. Implicit LES of a lid driven cavity

- **Problem setup**
  - Step 1. Run 5000 time steps with BDF1+DG(P1) and CFL = 500 from zero-velocity field, so that the flow filed reaches a cyclically oscillating status.
  - Step 2. Restart the computation with a fixed time-step size of dt = 0.1, and use a desired method as shown below. The width of window for time averaging is 30 second per frame (every 300 steps).



Density residual vs. time steps (fixed dt = 0.1)

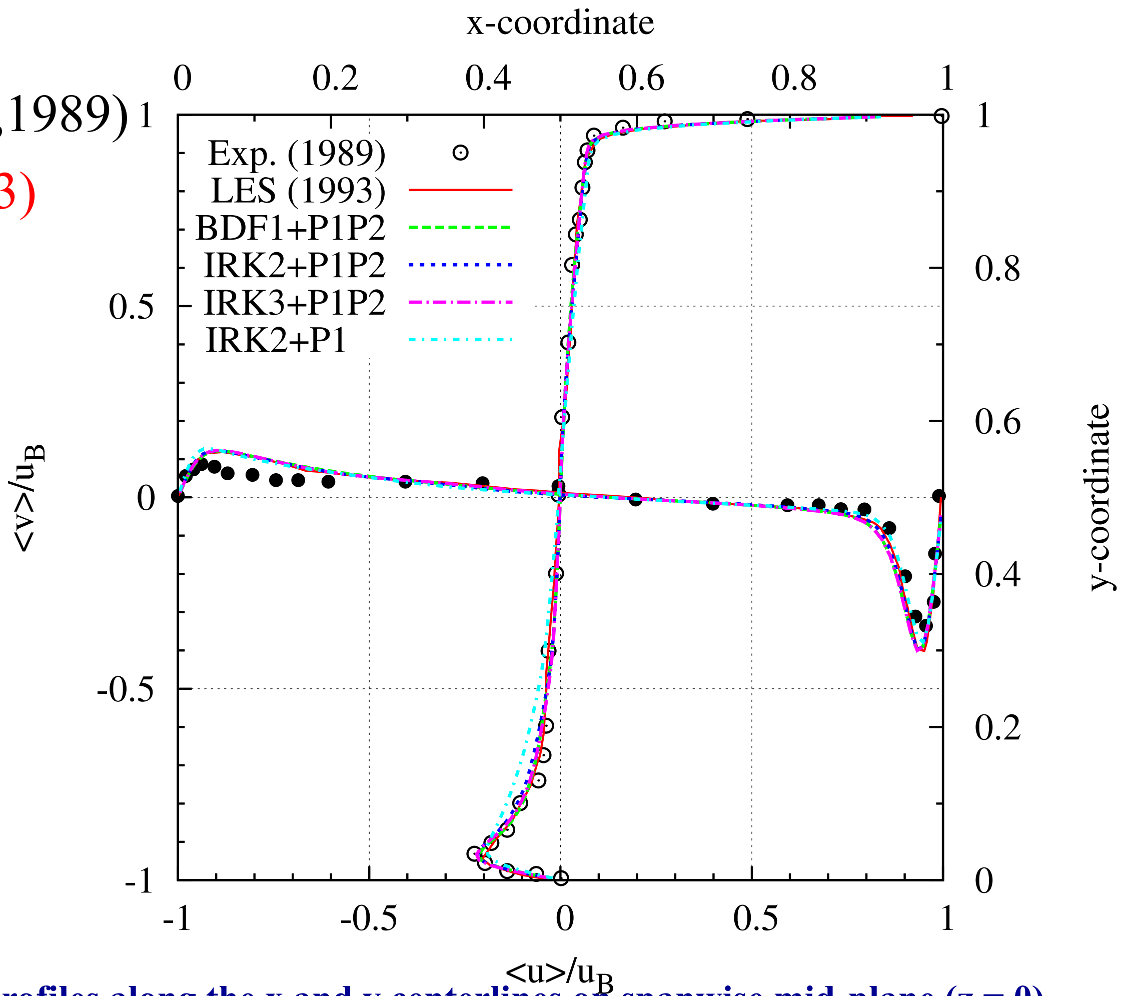


Total energy residual vs. time steps (fixed dt = 0.1)

Example 4. Implicit LES of a lid driven cavit

- **Mean velocities**
  - Exp. (Prasad&Koseff,1989)
  - LES (Zang et al., 1993)
  - BDF1+RDG(P1P2)
  - IRK2+RDG(P1P2)
  - IRK3+RDG(P1P2)
  - IRK2+DG(P1)

- RDG(P1P2) match all well.
- DG(P1) is a little off near bottom region.

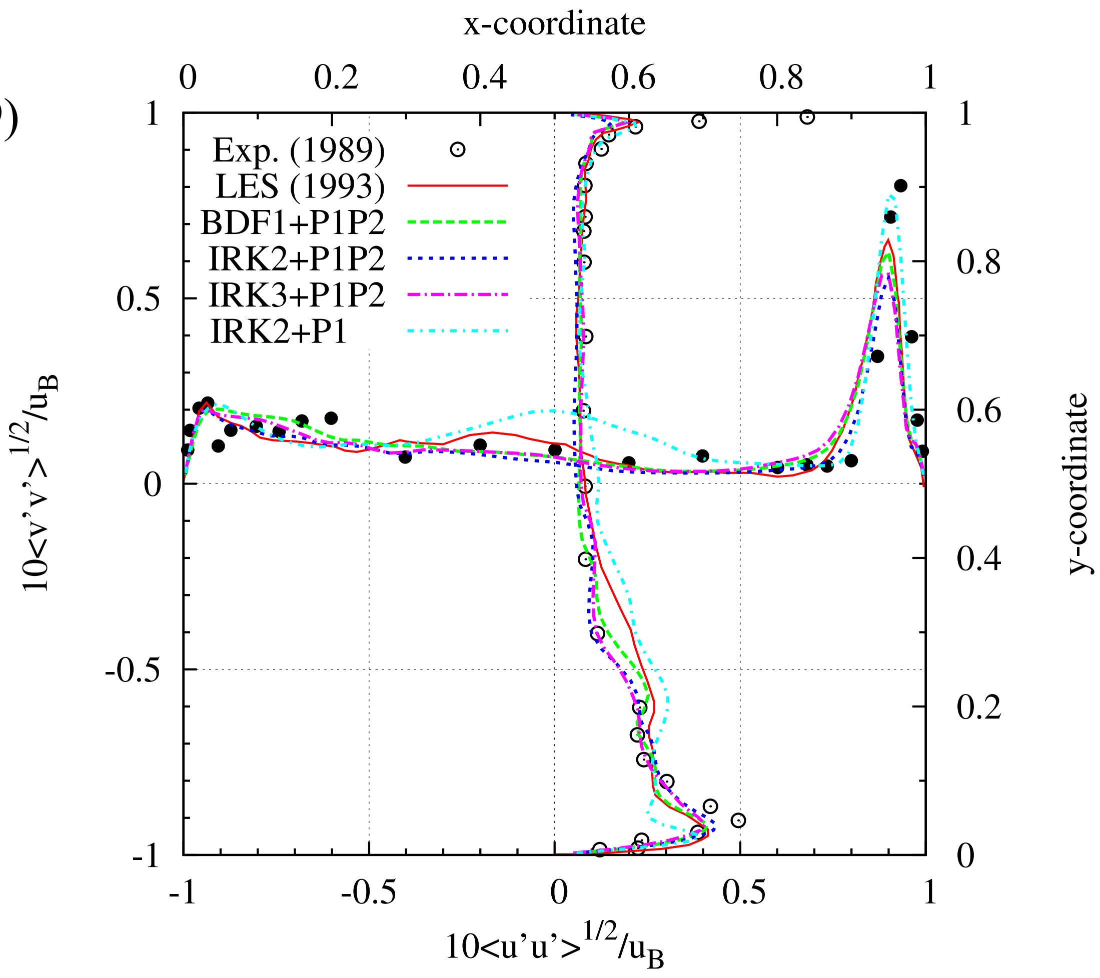


Profiles along the x and y centerlines on spanwise mid-plane (z = 0)

Example 4. Implicit LES of a lid driven cavit

- **RMS velocities**
  - Exp. (Prasad&Koseff,1989)
  - LES (Zang et al., 1993)
  - BDF1+RDG(P1P2)
  - IRK2+RDG(P1P2)
  - IRK3+RDG(P1P2)
  - IRK2+DG(P1)

- DG(P1) is not accurate enough.
- RDG(P1P2) matches exp. data well!
- IRK's are slightly better than BDF1.
- IRK3 is close to IRK2.

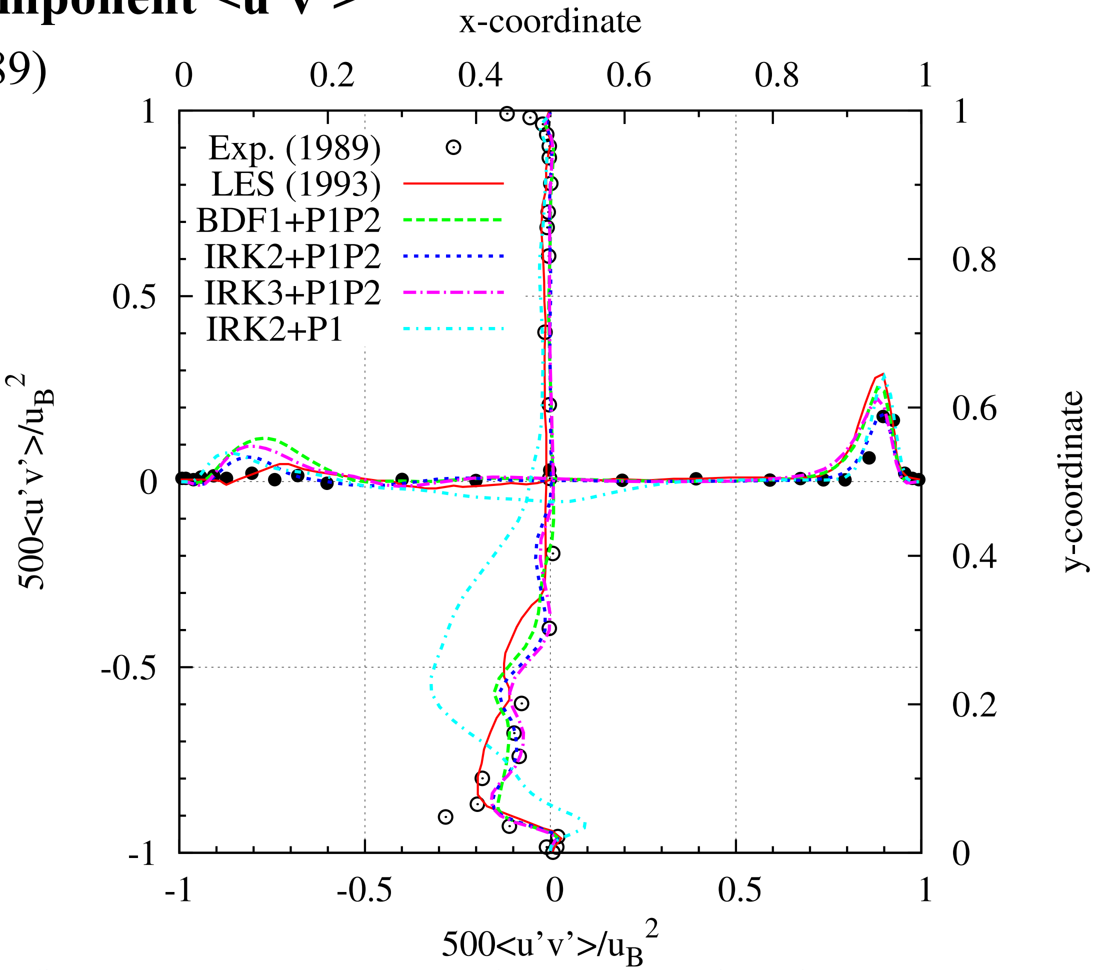


Profiles along the x and y centerlines on spanwise mid-plane (z = 0)

Example 4. Implicit LES of a lid driven cavit

- **Reynolds stress tensor component <u'v'>**
  - Exp. (Prasad&Koseff,1989)
  - LES (Zang et al., 1993)
  - BDF1+RDG(P1P2)
  - IRK2+RDG(P1P2)
  - IRK3+RDG(P1P2)
  - IRK2+DG(P1)

- DG(P1) is far from good in lower region.
- RDG(P1P2) matches exp. data well!
- IRK's are better than BDF1 in some regions.
- IRK3 is close to IRK2.



Profiles along the x and y centerlines on spanwise mid-plane (z = 0)

Example 4. Implicit LES of a lid driven cavity

- Comparison of the CPU time (evaluated by running on 256 cores) between the explicit and implicit methods.

For solution at t = 3000	Time-step size	Time steps	CPU time (sec)
BDF1 + RDG(P1P2)	dt = 0.1	30,000	52,542
IRK2 + RDG(P1P2)	dt = 0.1	30,000	86,066
IRK3 + RDG(P1P2)	dt = 0.1	30,000	263,010
IRK2 + DG(P1)	dt = 0.1	30,000	69,050
Explicit RK3 + RDG(P1P2)	dt = 0.0001	30,000,000	Estimated 7,347,942

- LU-SGS preconditioned GMRES solver
  - In average, a drop of **4** orders of magnitude for the unsteady residual can be achieved within **5** inner iterations at each implicit RK stage.

- A speedup factor of more than 85 by IRK over its explicit counterpart!
- IRK+RDG(P1P2) greatly improve solution accuracy for implicit LES without much extra cost than the underlying IRK+DG(P1)!

Concluding Remarks

- A reconstructed discontinuous Galerkin method based on a Hierarchical WENO reconstruction, HWENO(P_1P_2) has been developed for compressible flows at all speeds on hybrid grids.

- The HWENO(P_1P_2) method is able to provide sharp resolution of shock waves essentially without over- and under-shoots for discontinuities and achieve the designed third-order of accuracy for smooth flows.

- RDG methods have the potential to provide a superior alternative to the traditional FV methods, and to become a main choice for the next generation of CFD codes.

- A higher-order RDG-based CFD code will ultimately deliver a more accurate, efficient, robust, and reliable simulation tool with confidence that will enable us to solve flow problems at resolutions never before possible by the current state-of-the-art CFD technology.

Current Work

- Extension of the RDG method for turbulent flows
- Implementation of *hp*-adaptation on hybrid grids
- Port of the RDGFLO code on hybrid CPU/GPU architectures