Volume 43 Numbers 1–3 January–April 2007 ISSN 0376-0421
ELSEVIER
PROGRESS IN
AEROSPACE SCIENCES
AN INTERNATIONAL REVIEW JOURNAL
Editors:
B E RICHARDS
M F PLATZER

Available online at www.sciencedirect.com
ScienceDirect

Progress in Aerospace Sciences 43 (2007) 1–41

PROGRESS IN AEROSPACE SCIENCES

www.elsevier.com/locate/paerosci

High-order methods for the Euler and Navier–Stokes equations on unstructured grids

Z.J. Wang*

Department of Aerospace Engineering, Iowa State University, 2271 Howe Hall, Ames, IA 50011, USA

Available online 19 July 2007

Abstract

This article reviews several unstructured grid-based high-order methods for the compressible Euler and Navier–Stokes equations. We treat the spatial and temporal discretizations separately, hoping that it is easier to spot the similarities and differences of various types of methods. Our main focus is to present the basic design principles of each method, and highlight its pros and cons when appropriate. Sample computational results are shown to illustrate the capability of selected methods. These high-order methods are expected to be more efficient than low-order methods for problems requiring high accuracy, such as wave propagation problems, vortex-dominated flows including high-lift configuration, helicopter blade vortex interaction, as well as large eddy simulation and direct numerical simulation of turbulence. We conclude the paper with several current challenges in the proliferation of high-order methods in the aerospace community.
© 2007 Elsevier Ltd. All rights reserved.

Keywords: High-order; Unstructured grids; CFD; Navier–Stokes; Euler equations

Contents

*Tel.: +1 515 294 1614.
E-mail address: zjw@iastate.edu

0376-0421/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.paerosci.2007.05.001

1. Introduction

Computational fluid dynamics (CFD) has undergone tremendous development as a discipline for three decades, and is used routinely to complement the wind tunnel in the design of aircraft [1,2]. This has been made possible by progresses in many fronts, including numerical algorithms for the Navier–Stokes equations, grid generation and adaptation, turbulence modeling, flow visualization, as well as the dramatic increase in computer CPU and network speeds. Nearly all production flow solvers are based on second-order numerical methods, either finite volume (FV) [3–8], finite difference (FD) [9–11] or finite element (FE) [12–17,163]. They are capable of delivering design-quality Reynolds Averaged Navier–Stokes (RANS) results with several million cells (degrees of freedom or DOFs) on commercial Beowulf clusters within a few hours.

As impressive as these second-order codes are, there are still many flow problems considered out of reach, e.g., vortex-dominated flows including helicopter blade vortex interaction, and flow over high-lift configurations [2]. Unsteady propagating vortices are the main features of these flow problems, and second-order methods are too dissipative to resolve those unsteady vortices. High-order methods (order of accuracy >2) have shown promise in handling such flows. For example, high-order compact methods were demonstrated to produce much better results than low-order ones [18,19]. In order to limit the scope of the present paper, we only discuss unstructured grid-based high-order methods. Interested readers can refer to a comprehensive review article by Ekaterinaris [20] published in this journal for high-order methods on structured grids.

The advantage of high-order methods is well known. The error of a numerical method is said to be order $k+1$ if the solution error e decreases with mesh size h according to

$$e \propto h^{k+1}.$$

For a fourth order scheme, if the grid size halves, the error should be $\frac{1}{16}$ of that on the coarse mesh. The number of solution unknowns or DOFs in a simulation is of course closely related to the mesh size and the order of accuracy of the numerical method. These DOFs can be the solutions at certain grid locations (e.g., for FD and FE methods), or the averaged solutions in control volumes (e.g., for FV method), or the expansion coefficients of the solution with respect to a basis set (e.g., for mode-based spectral and spectral element methods). In FD and FV methods, each grid point or cell has one DOF. Then the total number of solution unknowns is related to the mesh size according to

$$\text{NDOFs} \propto \frac{1}{h^d},$$

where d is the physical dimension (1, 2 or 3). In the discontinuous Galerkin (DG) [21] and spectral volume (SV) [22,23] methods, for example, each simplex element or cell has multiple DOFs, depending on the order of accuracy $k+1$ (with a degree k polynomial reconstruction). The total number of DOFs is then

$$\text{NDOFs} \propto \frac{m_{k,d}}{h^d},$$

where $m_{k,d}$ is the number of unknowns in an element sufficient to reconstruct a degree k polynomial and is given by

$$m_{k,d} = \frac{\prod_{i=1}^{d}(k+i)}{d!}. \tag{1}$$

The subscripts in $m_{k,d}$ will be often omitted if there is no confusion. In this paper, we assume an optimal

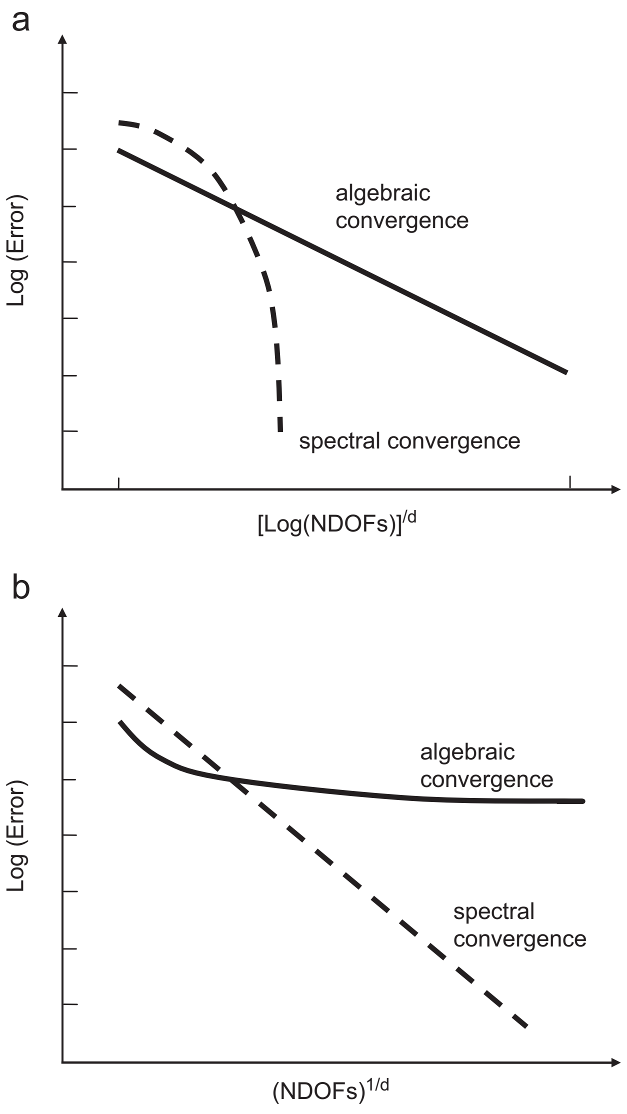


Fig. 1. Illustration of algebraic (or fixed order) convergence with *h*-refinement, and spectral (or exponential) convergence with *p*-refinement, where NDOFs is the total number of degrees of freedom, and d is the space dimension. (a) Log–log scale, (b) linear–log scale.

order of accuracy $k+1$ whenever the reconstruction polynomial is of degree k.

When comparing the cost and accuracy between different methods, it is important to ‘‘level the playing field’. It is obvious that the mesh size is not the only parameter deciding the total cost. For example, on a given mesh, a second-order DG scheme should produce a much more accurate solution than a second-order accurate FV scheme because there are multiple DOFs in a single element in the DG method. On the other hand, the cost of the DG method on the same mesh should be much higher than that of the FV method for the same reason. Therefore, it is much fairer to compare the DG and FV methods in terms of the cost and accuracy for the same number of DOFs.

In order to decide which method should be used for a particular application, a generic plot showing the rate of convergence in terms of NDOFs is helpful. For a fixed order of accuracy $k+1$, *h*-refinement (mesh size refinement) produces a constant rate of convergence as shown in Fig. 1a, which is called ‘‘algebraic convergence’’. The order of accuracy is the ‘‘constant’’ slope of log(e) vs. [log(NDOFs)]$/d$. On a fixed mesh, *p*-refinement (polynomial order refinement) produces a variable rate of convergence, which increases with the NDOFs. This is called ‘‘exponential convergence’’ or ‘‘spectral convergence’’, as shown in the same figure. The convergence rate is obviously much faster than algebraic convergence with increasing NDOFs. If these figures are plotted in a linear–log scale, one obtains Fig. 1b. In this scale, exponential convergence is a straight line, while algebraic convergence displays a deteriorating rate with increasing NDOFs. We want to point out that exponential convergence can be achieved only when the solution is very smooth with continuous high-order derivatives. Otherwise, algebraic convergence is expected even with *p*-refinement.

It is obvious that with the same NDOFs, higher-order methods should produce less solution errors than lower-order ones. Then why are they not used as widely as lower-order methods in the CFD community? There are several reasons. First, higher-order methods are generally not as robust as lower-order ones. Convergence to steady state can stall or in the worst case diverge. In transonic and supersonic flows with discontinuities, high-order methods usually lack the robust and versatile shock-capturing capability expected and found in lower-order ones. Second, the spatial operators in high-order methods are much stiffer than lower-order ones. As a result, it is much more challenging to converge a high-order simulation to steady state. If simple explicit methods are used for high-order operators, the allowable CFL number is much smaller than for low order ones. Given the fact that computational grids are highly clustered near solid boundaries to resolve the viscous boundary layer, explicit high-order methods are usually not competitive against well established implicit lower-order methods in terms of the error and cost ratio. Third, for certain types of design problems, lower-order methods are capable of meeting the accuracy requirement with a reasonable cost in human and computer resources. For example, the lift and drag of commercial airplanes under cruise conditions can often be accurately predicted using RANS simulations on low cost Beowulf clusters. In this case, the need is simply not there to invest in developing the capability and revising the design process to accommodate high-order simulations. Many years of experience in grid generation, validation and the CFD process would need to be re-examined and updated.

Ultimately, the deciding factor in choosing one method over another is the ratio of accuracy over cost. For example, if an error of 5 drag counts is acceptable, then whichever method costs the least CPU time will be the choice. Let us simplify our discussion by considering steady problems only. The cost of converging a flow simulation to steady state is related to the NDOFs in the following expression:

$$\text{Cost} \propto (\text{NDOFs})^{c(k)},$$

where c is larger than 1 and usually dependent on k. When $c = 1$, the method is optimum and is said to be $O(N)$ (costing $O(\text{NDOFs})$ operations). Although it is possible to achieve $O(N)$ cost with advanced solution methods (such as multigrid) for relatively simple problems, general $O(N)$ methods for the compressible Navier–Stokes equations for complex configurations are yet to be developed. Usually higher-order solvers have slower convergence rate. If we combine Fig. 1 with this cost estimate, we obtain a generic error vs. cost plot, shown in Fig. 2.

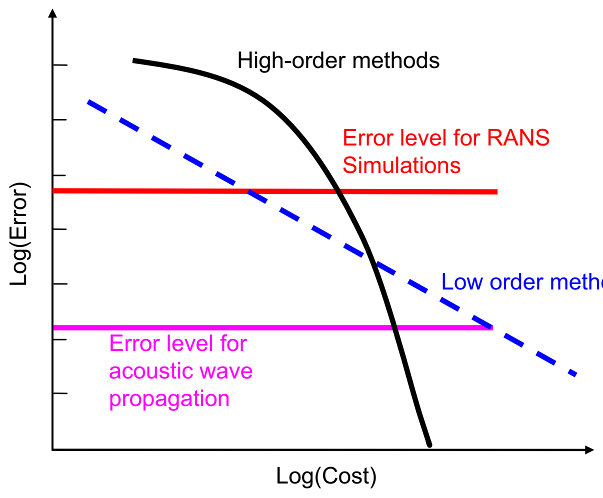


Fig. 2. A generic error vs. cost plot for low- and high-order methods to determine which method should be chosen for certain applications.

Based on the error requirement, one can then choose the most cost effective method. It is very clear from the figure that the higher the accuracy requirement, the more high-order methods are favored. It is no wonder in the computational aeroacoustics (CAA) community, low-order methods are rarely used for wave propagation problems because of the stringent requirement on the dissipation and dispersion characteristics.

Given a fixed number of DOFs, the highest accuracy is achieved with the spectral method [24,25,161] on one single element. The obvious drawback of the spectral method is the difficulty in mapping a complex computational domain into a single element. To alleviate this difficulty, spectral element or multidomain spectral methods have been developed by various researchers [26–28], for various cell types including simplexes [29,30]. These methods strike a good compromise between accuracy and geometric flexibility. In fact, all methods discussed in this paper share this compromise.

The paper is organized as follows. In the next session, we present the governing equations. After that, spatial discretization methods are discussed. Main features of the k-exact FV, essentially non-oscillatory (ENO), weighted ENO (WENO), continuous FE, DG, residual distribution (RD), SV and spectral difference (SD) methods are reviewed, followed by some example computational results. Next, time integration and iterative solution methods are described, starting from explicit Runge–Kutta method, then various implicit solution approaches and finally *hp*-multigrid methods. We conclude the paper by presenting several current challenges in high-order methods.

2. Governing equations

Consider the unsteady compressible Navier–Stokes equations written in conservative form

$$\frac{\partial Q}{\partial t} + \nabla \cdot \vec{F}(Q, \nabla Q) \equiv \frac{\partial Q}{\partial t} + \nabla \cdot [\vec{F}_{\mathrm{i}}(Q) - \vec{F}_{\mathrm{v}}(Q, \nabla Q)] = 0, \tag{2}$$

where Q is the vector of conserved variables $Q = \{\rho, \rho\vec{v}, E_t\}^{\mathrm{T}}$, $\vec{F}$ is the total flux vector including both the inviscid flux vector $\vec{F}_{\mathrm{i}}$ and viscous flux vector $\vec{F}_{\mathrm{v}}$. Let the computational domain be $\Omega \subset R^d$, with the dimension $d = 1, 2, 3$. The boundary of Ω is $\partial\Omega$ and its outward unit normal is $\vec{n}$.

The weighted residual form of the governing equations can be easily derived by multiplying (2) with an arbitrary weighting function W and integrating over the computational domain to obtain

$$\int_\Omega \left[\frac{\partial Q}{\partial t} + \nabla \cdot \vec{F}(Q, \nabla Q)\right] W \, \mathrm{d}V = \int_\Omega \frac{\partial Q}{\partial t} W \, \mathrm{d}V + \int_{\partial\Omega} W\vec{F}(Q, \nabla Q) \cdot \vec{n} \, \mathrm{d}S - \int_\Omega \nabla W \cdot \vec{F}(Q, \nabla Q) \, \mathrm{d}V = 0. \tag{3}$$

In the special case of $W = 1$, the following integral form of the governing equations is obtained:

$$\int_\Omega \frac{\partial Q}{\partial t} \, \mathrm{d}V + \int_{\partial\Omega} \vec{F}(Q, \nabla Q) \cdot \vec{n} \, \mathrm{d}S = 0. \tag{4}$$

This easy to understand, yet elegant form must have contributed to the popularity of the FV method in the CFD community! The convective and diffusive parts of the governing equations have very different physical behavior and are often treated differently [31]. For example, during the development of shock-capturing methods, the most attention was paid to the discretization of inviscid fluxes, as shock waves are characteristics of hyperbolic equations. Therefore it is often fruitful to consider a subset of the Navier–Stokes equations, i.e., the Euler equations, especially in the initial stage of new method development. The Euler equations are obtained if we drop the viscous fluxes. If we use the notation $\vec{F}(Q)$ instead of $\vec{F}(Q, \nabla Q)$, we assume the Euler equations.

In the implementation of many numerical methods, it is often more efficient to map a physical element (or cell) into a standard element. The governing equations should also be transformed from the physical domain to the computational domain. Consider the following transformation:

$$x = x(\xi, \eta, \zeta),$$
$$y = y(\xi, \eta, \zeta),$$
$$z = z(\xi, \eta, \zeta), \tag{5}$$

with a non-singular Jacobian

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

Its inverse transformation exists, and relates to the Jacobian according to

$$\frac{\partial(\xi, \eta, \zeta)}{\partial(x, y, z)} = \begin{bmatrix} \xi_x & \xi_y & \xi_z \\ \eta_x & \eta_y & \eta_z \\ \zeta_x & \zeta_y & \zeta_z \end{bmatrix} = J^{-1}.$$

Hence the metrics can be computed using

$$\xi_x = (y_\eta z_\zeta - y_\zeta z_\eta)/|J|, \quad \xi_y = (x_\zeta z_\eta - x_\eta z_\zeta)/|J|, \quad \xi_z = (x_\eta y_\zeta - x_\zeta y_\eta)/|J|,$$

$$\eta_x = (y_\zeta z_\xi - y_\xi z_\zeta)/|J|, \quad \eta_y = (x_\xi z_\zeta - x_\zeta z_\xi)/|J|, \quad \eta_z = (x_\zeta y_\xi - x_\xi y_\zeta)/|J|,$$

$$\zeta_x = (y_\xi z_\eta - y_\eta z_\xi)/|J|, \quad \zeta_y = (x_\eta z_\xi - x_\xi z_\eta)/|J|, \quad \zeta_z = (x_\xi y_\eta - x_\eta y_\xi)/|J|.$$

The governing equations in the computational domain take the following form:

$$\frac{\partial \tilde{Q}}{\partial t} + \nabla_\xi \cdot \vec{\tilde{F}} = 0, \tag{6}$$

where $\tilde{Q} = |J| \cdot Q$ and $\vec{\tilde{F}} = |J| J^{-1} \cdot \vec{F}$.

In order to present basic ideas in the easiest way to understand fashion, the following scalar wave equation is often considered:

$$\frac{\partial u}{\partial t} + \nabla \cdot (u\vec{a}) \equiv \frac{\partial u}{\partial t} + \vec{a} \cdot \nabla u = 0, \tag{7}$$

where u is a (scalar) state variable, and $\vec{a}$ is the wave velocity vector. The viscous effects can be included by adding a scalar diffusion term

$$\frac{\partial u}{\partial t} + \vec{a} \cdot \nabla u = \mu \nabla^2 u, \tag{8}$$

where μ is the viscosity. In one dimension, (8) degenerates to

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = \mu \frac{\partial^2 u}{\partial x^2}, \quad a > 0, \ \mu \geqslant 0. \tag{9}$$

When $\mu = 0$, (9) becomes

$$\frac{\partial u}{\partial t} + a \frac{\partial u}{\partial x} = 0, \tag{10}$$

which is perhaps the most popular model equation in CFD – the 1D scalar wave equation.

3. Space discretization methods

In this section, the basic formulations of several numerical methods on unstructured grids are described. For simplicity, we consider 2D problems.

The computational domain Ω is discretized into N non-overlapping polygonal elements (cells) V_i, with volume $|V_i|$. In most cases, we consider triangular elements in 2D.

3.1. K-Exact FV and ENO/WENO methods

The high-order k-exact method was developed by Barth and Frederickson [32]. It belongs to a general class of Godunov-type FV methods. This type of FV methods can trace its root to the first-order Godunov method [33], which was later extended to second-order by van Leer [34,35], third order by Colella and Woodward [36], and to arbitrary order of accuracy by Harten et al. [37], all on structured grids. With the explosive adoption of unstructured grids in the CFD community, the Godunov-type method was extended to unstructured grids [3,4,6,38–45]. In fact, most of the unstructured grid-based production CFD codes and commercial codes use the second-order version. Furthermore, ideas from the ENO [37,162] methods were also extended to unstructured grids by many researchers [46–49]. WENO schemes were developed later to improve upon ENO schemes, in Liu et al. [50] and Jiang and Shu [51], and also extended to unstructured grids by Friedrich [52], and Hu and Shu [53]. Advantages of WENO schemes over ENO include the smoothness of numerical fluxes, better steady-state convergence, and better accuracy using the same stencils. For a review of ENO and WENO schemes, see [54]. Since these methods differ only in solution reconstruction, they are described together.

Applying the integral form of the Euler equations to control volume V_i, we obtain the following semi-discrete FV scheme:

$$\frac{d\overline{Q}_i}{dt}|V_i| + \int_{\partial V_i} \vec{F}(Q)\cdot\vec{n}\,dS = \frac{d\overline{Q}_i}{dt}|V_i| + \sum_{f\in\partial V_i}\int_f \vec{F}(Q)\cdot\vec{n}\,dS = 0, \tag{11}$$

where $\overline{Q}_i \equiv (1/|V_i|)\int_{V_i} Q\,dV$ is the volume averaged state variable. Define the face averaged normal flux to be

$$\overline{F}_f = \frac{\int_f \vec{F}(Q)\cdot\vec{n}\,dS}{|f|}, \tag{12}$$

where $|f|$ is the area of face f. Then (11) can be further written as

$$\frac{d\overline{Q}_i}{dt}|V_i| + \sum_{f\in\partial V_i}\overline{F}_f|f| = 0. \tag{13}$$

Mathematically (13) is equivalent to (2) if the solution is sufficiently smooth. In a numerical approximation, however, $\overline{F}_f$ can rarely be computed exactly even if $\overline{Q}_i$ is known. For simplicity, we use polynomials to approximate the solution. In spite of this, the normal flux is usually not a polynomial. Instead, a Gaussian quadrature formula is employed to compute the face integral

$$\overline{F}_f \approx \sum_q w_q \vec{F}(\vec{r}_{f,q})\cdot\vec{n}_f, \tag{14}$$

where $\vec{r}_{f,q}$ is the Gauss quadrature point and w_q the quadrature weight. The total number of quadrature points is determined based on the desired quadrature precision. For example, one point is used for linear reconstruction, but two points are needed for quadratic and cubic reconstructions. Because state variables are approximated by piece-wise polynomials, the solution is discontinuous across cell interfaces. According to Godunov, the interface normal flux is replaced by a Riemann flux

$$\vec{F}(\vec{r}_{f,q})\cdot\vec{n}_f \approx \hat{F}(Q^-(\vec{r}_{f,q}), Q^+(\vec{r}_{f,q}), \vec{n}_f), \tag{15}$$

where Q^- and Q^+ are the reconstructed solutions inside element V_i and outside V_i (or inside the neighboring cell). The simplest Riemann flux is the Rusanov flux [55]

$$\hat{F}(Q^-(\vec{r}), Q^+(\vec{r}), \vec{n}) \equiv \tfrac{1}{2}[\vec{F}(Q^-(\vec{r}))\cdot\vec{n} + \vec{F}(Q^+(\vec{r}))\cdot\vec{n} - \lambda(Q^+(\vec{r}) - Q^-(\vec{r}))], \tag{16}$$

where λ is the maximum absolute wave speed in direction $\vec{n}$, evaluated at an average state of Q^- and Q^+. Other more sophisticated approximate Riemann solvers can also be used [56–58,165]. All FV- type methods (k-exact, ENO and WENO) are identical up to now, and they differ only in how the solution is reconstructed given the cell averaged state variables. The reconstruction problem reads: construct a degree k polynomial $p_i(\vec{r})$ given the cell averaged means at a local stencil $S_i\{\overline{Q}_j\}_{j\in S_i}$ such that $p(\vec{r})$ is a $(k+1)$th order approximation of Q, i.e.,

$$p_i(\vec{r}) = Q(\vec{r}) + O(h^{k+1}). \tag{17}$$

3.1.1. k-Exact reconstruction

The k-exactness property means that if the solution is a degree k or lower polynomial, the solution is reconstructed exactly. Another property which any reconstruction must satisfy is conservation, i.e., the mean of the reconstructed polynomial on V_i must be $\overline{Q}_i$

$$\frac{\int_{V_i} p_i(\vec{r})\,\mathrm{d}V}{|V_i|} = \frac{\int_{V_i} Q(\vec{r})}{|V_i|} \equiv \overline{Q}_i. \tag{18}$$

Let the centroid of V_i be $\vec{r}_i = (x_i, y_i)$. The mean preserving property can be easily satisfied using the following Taylor expansion:

$$\begin{aligned} p_i(\vec{r}) = \overline{Q}_i &+ \frac{\partial Q_i}{\partial x}(x - x_i) + \frac{\partial Q_i}{\partial y}(y - y_i) \\ &+ \frac{\partial^2 Q_i}{\partial x \partial y}[(x - x_i)(y - y_i) - \overline{x_i y_i}] \\ &+ \frac{1}{2}\frac{\partial^2 Q_i}{\partial x^2}[(x - x_i)^2 - \overline{x_i^2}] \\ &+ \frac{1}{2}\frac{\partial^2 Q_i}{\partial y^2}[(y - y_i)^2 - \overline{y_i^2}] + \cdots, \end{aligned} \tag{19}$$

where

$$\begin{aligned} \overline{x_i y_i} &= \frac{1}{|V_i|}\int_{V_i}(x - x_i)(y - y_i)\,\mathrm{d}V, \\ \overline{x_i^2} &= \frac{1}{|V_i|}\int_{V_i}(x - x_i)^2\,\mathrm{d}V, \\ \overline{y_i^2} &= \frac{1}{|V_i|}\int_{V_i}(y - y_i)^2\,\mathrm{d}V. \end{aligned} \tag{20}$$

The above equation can be viewed as an expansion with respect to a set of zero mean basis functions (except the first term) for degree k polynomials $\{\varphi_l(\vec{r})\}_{l=1}^{m-1}$

$$p_i(\vec{r}) - \overline{Q}_i = \sum_{l=1}^{m-1} b_l \varphi_l(\vec{r}), \tag{21}$$

where b_l is the expansion coefficient. We need at least $m-1$ neighboring cells in the local stencil to compute the coefficients. To determine these coefficients, we again require that the polynomial also is mean preserving in the local stencil

$$\frac{\int_{V_j} p_i(\vec{r})\,\mathrm{d}V}{|V_j|} = \frac{\sum_{l=1}^{m-1} b_l \int_{V_j}\varphi_l(\vec{r})\,\mathrm{d}V}{|V_j|} + \overline{Q}_i = \overline{Q}_j, \quad \forall j \in S_i. \tag{22}$$

A compact form of (22) is

$$Ab = \overline{Q}, \tag{23}$$

where matrix A has elements $a_{j,l} = \int_{V_j}\varphi_l(\vec{r})\,\mathrm{d}V/|V_j|$, $b = \{b_l\}_{l=1}^{m-1}$ and $\overline{Q} = \{\overline{Q}_j - \overline{Q}_i\}_{j \in S_i}$. For (23) to have a solution, there must be at least $m-1$ neighboring cells in S_i. Generally speaking, the larger the stencil, the more stable the reconstruction is. In practice, in order to prevent singular stencils, more than $m-1$ neighboring cells are included in the stencil. Therefore (23) is solved in the least-squares sense, which can be expressed using the following pseudo-inverse:

$$b = A^{+1}\overline{Q}. \tag{24}$$

Substituting (24) into (21), we obtain

$$p_i(\vec{r}) - \overline{Q}_i = \Theta(\vec{r})b = \Theta(\vec{r})A^{+}\overline{Q}, \tag{25}$$

where $\Theta(\vec{r}) = \{\varphi_1(\vec{r}), \ldots, \varphi_{m-1}(\vec{r})\}$. Denote $\overline{L}(\vec{r}) = \Theta(\vec{r})A^{+}$. Finally the reconstruction can be written as

$$p_i(\vec{r}) - \overline{Q}_i = \overline{L}(\vec{r})\overline{Q}. \tag{26}$$

In an actual numerical implementation, it is the most efficient to store the values of $\vec{L}(\vec{r})$ at the Gauss quadrature points. $\overline{L}(\vec{r})$ is dependent only on the grid and the local stencil, and can be pre-computed and stored. However, since a set of coefficients must be stored for every quadrature point, and each element has different sets of coefficients, the memory requirement in 3D for a high-order k-exact scheme may be large [59].

For flow problems involving discontinuities, the high-order reconstruction is oscillatory due to the so-called Gibbs phenomenon. Data limiting becomes necessary to avoid non-physical state such as negative pressure or density. A popular approach developed by Barth and Jespersen [5] is to limit the reconstructed polynomial to satisfy a local maximum principle

$$\min\{\overline{Q}_i, \overline{Q}_j|_{j \in S_i}\} \leqslant p_i(\vec{r}) \leqslant \max\{\overline{Q}_i, \overline{Q}_j|_{j \in S_i}\}, \quad \vec{r} \in V_i. \tag{27}$$

If $p_i(\vec{r})$ is linear, it is relatively easy to guarantee (27) because the minimum and maximum occur at the cell vertices. Otherwise, the extremes can occur any where in the cell. If the maximum principle (27) is violated, the solution is assumed locally linear

$$p_i(\vec{r}) = \overline{Q}_i + \nabla Q_i \cdot (\vec{r} - \vec{r}_i), \tag{28}$$

where ∇Q_i can be computed using a linear least squares reconstruction including only the immediate face neighbors. If (28) still violates the maximum principle, the solution gradient is reduced until (27) is satisfied by introducing a scalar limiter function

$$p_i(\vec{r}) = \overline{Q}_i + \Psi_i \nabla Q_i \cdot (\vec{r} - \vec{r}_i), \tag{29}$$

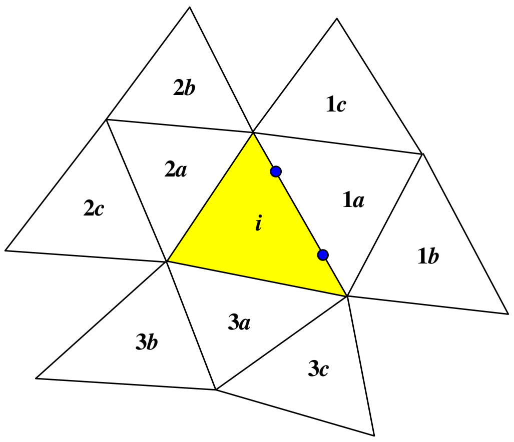


Fig. 3. Candidate reconstruction stencils in an ENO reconstruction.

with $\Psi_i \in [0,1]$. A weaker condition than (27) is that the solutions at all the quadrature points are bounded by the minimum and maximum mean. This condition is very easy to enforce, but does not guarantee (27).

3.1.2. ENO and WENO reconstruction

The basic idea in an ENO reconstruction is to adaptively select the "smoothest" one from a set of candidate reconstructions built from several different local stencils. For a degree k ENO reconstruction, there must exist at least two different stencils supporting a degree k polynomial reconstruction. After that, the smoothest reconstruction is selected based on a smoothness criterion. Finally this polynomial is used to compute the state variables at the Gauss quadrature points, which are employed to compute the Riemann fluxes for a FV update of the DOFs. Refer to Fig. 3. For cell i, the following three candidate stencils can be identified:

$$S_{i,1} = \{V_i, V_{1a}, V_{1b}, V_{1c}\},$$

$$S_{i,2} = \{V_i, V_{2a}, V_{2b}, V_{2c}\},$$

$$S_{i,3} = \{V_i, V_{3a}, V_{3b}, V_{3c}\}.$$

Each stencil can be used to produce a unique linear polynomial resulting in a total of three reconstructions $\{p_{i,j}(\vec{r})\}_{j=1}^{3}$. A possible measure of solution smoothness is the norm of the reconstruction polynomial gradient $|\nabla p_{i,j}(\vec{r})|$. One can select the polynomial with the minimum norm. Since the reconstruction is linear, the resultant ENO scheme is only second-order accurate although a total of nine neighboring cells are used.

In a WENO reconstruction developed by Hu and Shu [53], a key step in building a high-order WENO scheme based on lower-order polynomials is carried out in the following. We want to construct several linear polynomials whose weighted average will give the same result as the quadratic reconstruction $p_i^2(\vec{r})$ at each quadrature point (the weights are different for different quadrature points). More specifically, the following nine linear polynomials are built by agreeing with the cell averages of the solution on the following stencils: $p_{i,1}$ (on triangles: i; $1a$; $2a$), $p_{i,2}$ (on triangles: i; $2a$; $3a$), $p_{i,3}$ (on triangles: i; $3a$; $1a$), $p_{i,4}$ (on triangles: i; $1a$; $1b$), $p_{i,5}$ (on triangles: i; $1a$; $1c$), $p_{i,6}$ (on triangles: i; $2a$; $2b$), $p_{i,7}$ (on triangles: i; $2a$; $2c$), $p_{i,8}$ (on triangles: i; $3a$; $3b$), $p_{i,9}$ (on triangles: i; $3a$; $3c$). For each quadrature point $\vec{r}_q$, linear weights γ_j are found, which are constants depending only on the local geometry of the mesh, such that the linear polynomial obtained from a linear combination of these $p_{i,j}(\vec{r})$:

$$p_i(\vec{r}) = \sum_{j=1}^{9} \gamma_j p_{i,j}(\vec{r}) \tag{30}$$

satisfies

$$p_i(\vec{r}_q) = p_i^2(\vec{r}_q), \tag{31}$$

where $p_i^2(\vec{r})$ is the quadratic k-exact reconstruction using the cell averages on all nine neighboring cells, as shown in Fig. 3. Obviously, for degree 0 and 1 polynomials, any weights satisfying

$$\sum_{j=1}^{9} \gamma_j = 1 \tag{32}$$

are solutions of (31). More details on how these weights are determined are contained in [53].

Another major step in the WENO reconstruction is to find positive non-linear weights, which return the high-order weights in smooth region. For stencils containing a shock wave, the non-linear weights for those stencils should be very small. To ensure stability near shocks, we need non-negative weights, and thus we need non-negative linear weights to start with. A grouping of polynomials is used by Hu and Shu [53] to achieve positivity. For example, the nine linear polynomials are combined into three groups

$$\sum_{j=1}^{9} \gamma_j p_{i,j}(\vec{r}) = \sum_{j=1}^{3} \tilde{\gamma}_j \tilde{p}_{i,j}(\vec{r}). \tag{33}$$

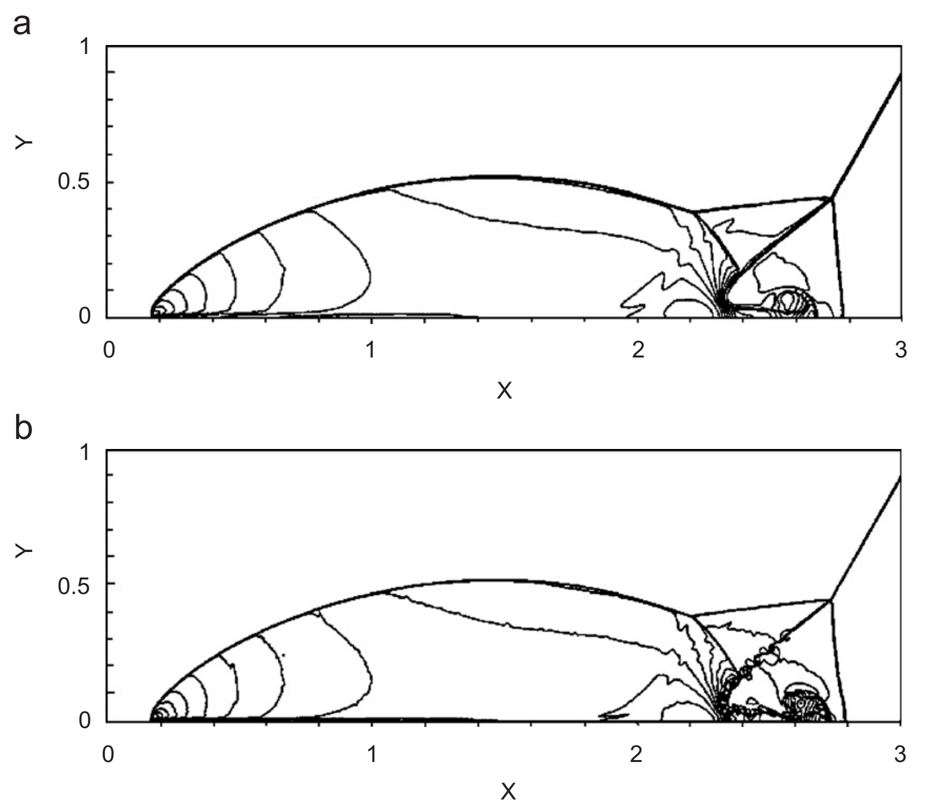


Fig. 4. Computed density contours using WENO schemes for the double Mach reflection problem (courtesy of Hu and Shu [53]). (a) Third-order WENO (1.28 M DOFs), (b) fourth-order WENO (1.28 M DOFs).

Each $\tilde{p}_{i,j}(\vec{r})$ is still a linear polynomial with positive coefficients $\tilde{\gamma}_j$. Finally the smoothness indicator due to Jiang and Shu [51] is used. For a polynomial $p(\vec{r})$ with degree up to k, we define the following measurement for smoothness:

$$\Xi = \sum_{1\leqslant|\phi|\leqslant k} \int_{V_i} |V_i|^{|\phi|-1} [D^{\phi} p(\vec{r})]^2 \, \mathrm{d}V, \tag{34}$$

where ϕ is a multi-index and D is the derivative operator. The non-linear weights are then defined as

$$\varpi_j = \frac{\tilde{\varpi}_j}{\sum_l \tilde{\varpi}_l}, \quad \tilde{\varpi}_l = \frac{\tilde{\gamma}_l}{(\varepsilon + \Xi_l)^2}, \tag{35}$$

where Ξ_l is the smoothness indicator for polynomial $\tilde{p}_{i,l}(\vec{r})$, and ε is a small positive number [53].

To demonstrate the capability of the WENO method, a sample computational example of double Mach reflection by Hu and Shu [53] is shown here. This problem is originally from [60]. The computational domain for this problem is chosen to be $[0, 4] \times [0, 1]$. The reflecting wall lies at the bottom of the computational domain starting from $x = \frac{1}{6}$. Initially a right-moving Mach 10 shock is positioned at $x = \frac{1}{6}, y = 0$ and makes a 60° angle with the x-axis. For the bottom boundary, the exact post-shock condition is imposed for the region from $x = 0$ to $\frac{1}{6}$ and a solid wall boundary condition is used for the rest. For the top boundary of the computational domain, the solution is set to describe the exact motion of the Mach 10 shock. The left boundary is set as the exact post-shock condition, while the right boundary is set as outflow boundary. The computed density contours using the third- and fourth-order WENO scheme on a mesh with $h = \frac{1}{400}$ are displayed in Fig. 4, and enlarged views of the same plots around the double Mach stems are shown in Fig. 5. In both computations, regular triangular grids were used. Since each cell has one DOF, the total number of DOFs is 1.28 million for both simulations. We can clearly see that the fourth-order scheme resolved the complicated flow structure under the triple Mach stem much better than the third-order scheme, and the discontinuities are captured with high resolution.

3.2. Continuous FE methods

In this subsection, high-order FE methods employing continuous polynomials across element

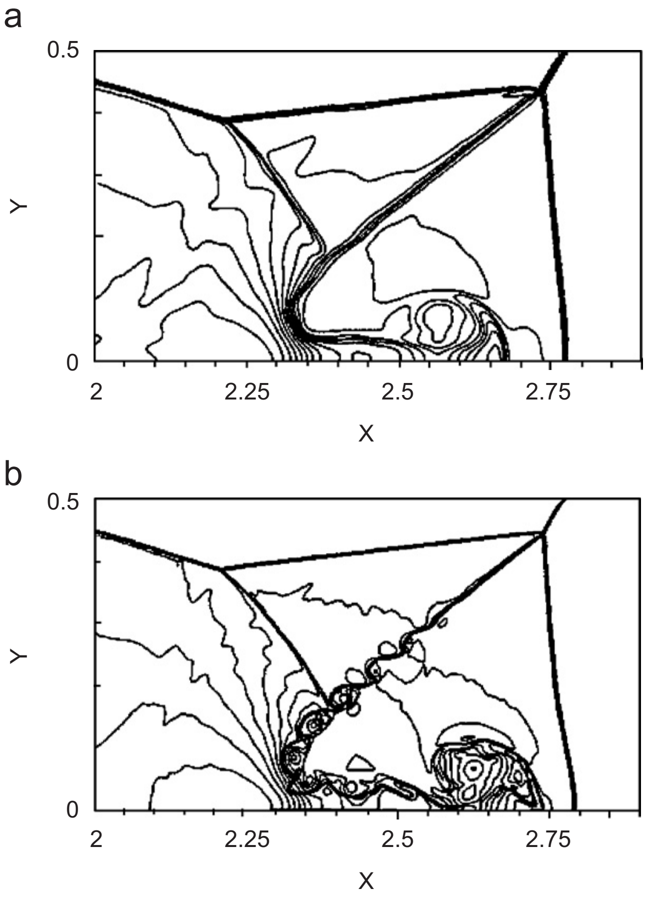


Fig. 5. Close-up view of the density contours for the double Mach reflection problem (courtesy of Hu and Shu [53]). (a) Third-order WENO (1.28 M DOFs), (b) fourth-order WENO (1.28 M DOFs).

interfaces are briefly discussed. There is a vast literature on these methods, and interested readers should consult many well-known books and review articles [14,61–66,160] for more details. It is well known that the continuous Galerkin method is not stable for the wave equation, similar to the fact that the central FD scheme is unstable. To illustrate this point, consider the 1D wave equation on domain $[0, X]$ with uniform mesh size $h = X/N$ and a periodic boundary condition. The DOFs are the solutions at the grid points $\{u_i\}_{i=1}^{N}$. The approximate solution in the computational domain is piece-wise linear with the following hat basis function:

$$\varphi_i = \begin{cases} [x-(i-1)h]/h & (i-1)h \leqslant x < ih, \\ [(i+1)h - x]/h, & ih \leqslant x < (i+1)h, \\ 0 & \text{otherwise}. \end{cases} \tag{36}$$

The solution can be written as

$$u_h = \sum_{i=1}^{N} \varphi_i u_i. \tag{37}$$

The Galerkin weighted residual statement for the wave equation is then

$$\frac{\mathrm{d}}{\mathrm{d}t}\sum_{i=1}^{N} u_i \int_0^X \varphi_i\varphi_j \,\mathrm{d}x + (\varphi_j a u)|_0^X - \int_0^X \frac{\mathrm{d}\varphi_j}{\mathrm{d}x} a \sum_{i=1}^{N} \varphi_i u_i \,\mathrm{d}x = 0, \quad \forall j = 1, \ldots, N. \tag{38}$$

A simplification of (38) gives

$$\frac{1}{6}\frac{\mathrm{d}u_{j-1}}{\mathrm{d}t} + \frac{2}{3}\frac{\mathrm{d}u_j}{\mathrm{d}t} + \frac{1}{6}\frac{\mathrm{d}u_{j+1}}{\mathrm{d}t} + a\frac{u_{j+1}-u_{j-1}}{2h} = 0, \quad \forall j = 1, \ldots, N. \tag{39}$$

A coupled tri-diagonal system with central difference for the space derivative is obtained, which has no numerical dissipation. Many types of stabilizing techniques have been developed in the FE community to remedy the stability problem. Examples include the streamline upwind Petrov–Galerkin (SUPG) [14], Galerkin/least squares [15,65–67], Taylor Galerkin method [17] amongst many others. To understand the basic idea, we present the SUPG here. Consider the FD upwind scheme for the wave equation

$$\frac{\mathrm{d}u_i}{\mathrm{d}t} = -a\frac{u_i - u_{i-1}}{h}. \tag{40}$$

The modified equation for (40) is simply

$$\frac{\partial u_i}{\partial t} + a\frac{\partial u_i}{\partial x} = \frac{ah}{2}\frac{\partial^2 u_i}{\partial x^2} + \cdots. \tag{41}$$

To mimic this upwind idea in the Galerkin FE method, one can apply the Galerkin approach to solve $\partial u/\partial t + a\partial u/\partial x - (ah/2)\partial^2 u/\partial x^2 = 0$ instead of the original wave equation, i.e.,

$$\int_0^X W\left(\frac{\partial u}{\partial t} + a\frac{\partial u}{\partial x} - \frac{ah}{2}\frac{\partial^2 u}{\partial x^2}\right)\mathrm{d}x = \int_0^X W\frac{\partial u}{\partial t} + \int_0^X a\left(W + \frac{h}{2}\frac{\mathrm{d}W}{\mathrm{d}x}\right)\frac{\partial u}{\partial x}\,\mathrm{d}x - \frac{ahW}{2}\frac{\partial u}{\partial x}\bigg|_0^X = 0. \tag{42}$$

If we ignore the unsteady and boundary terms (for the wave equation, $\partial u/\partial x$ is usually not given at boundaries and can be safely assumed 0), (42) can be interpreted as using a new weighting function

$$W' = W + \frac{h}{2}\frac{\mathrm{d}W}{\mathrm{d}x}. \tag{43}$$

for the original wave equation. This is the basic idea of the SUPG [14]. An equivalent interpretation is adding the following stabilization term to the Galerkin formulation:

$$\int_0^X \frac{ah}{2}\frac{\mathrm{d}W}{\mathrm{d}x}\frac{\partial u}{\partial x}\,\mathrm{d}x. \tag{44}$$

One can also adjust the amount of artificial dissipation by introducing a parameter $\alpha \in [0,1]$. For the 2D wave equation, the corresponding weight and stabilization terms including α are

$$W' = W + \frac{\alpha\bar{h}}{2|\vec{a}|}(\vec{a}\cdot\nabla W), \tag{45}$$

$$\int_\Omega \frac{\alpha\bar{h}}{2|\vec{a}|}(\vec{a}\cdot\nabla W)\nabla\cdot(\vec{a}u)\,\mathrm{d}V, \tag{46}$$

where $\bar{h}$ is the approximate local mesh size in the flow direction. Eq. (45) can be further simplified by defining $\tau = \alpha\bar{h}/2|\vec{a}|$, which is a time scale related to the wave speed and local mesh size. For the Euler and Navier–Stokes equations, they can be generalized as

$$W' = W + \tau(\vec{J}\cdot\nabla W), \tag{47}$$

$$\int_\Omega \tau(\vec{J}\cdot\nabla W)\nabla\cdot\vec{F}\,\mathrm{d}V, \tag{48}$$

where $\vec{J} = \partial\vec{F}/\partial Q$, and τ is a time-scale matrix. Many different ways exist in the literature on how to choose τ [14,63,68].

It is obvious that for the 2D wave equation, the SUPG scheme is linear in the DOFs. The scheme may not be stable for discontinuities. In practice, ad hoc shock-capturing terms must be added for stability for strong shock waves. As noted by Venkatakrishnan et al. [69], no shock-capturing terms can guarantee stability in SUPG for general cases.

3.3. DG methods

The DG FE method was originally developed by Reed and Hill [70] to solve the neutron transport equation. The development of the DG method for hyperbolic conservation laws was pioneered by Cockburn, Shu and their collaborators in a series of papers on the Runge–Kutta DG (RKDG) method [21,71–73]. Bassi and Rebay demonstrated the DG method for the compressible Euler and Navier–Stokes equations in obtaining high-order accuracy [74–76]. Many other researchers made significant contributions in the development of the DG method, e.g., [77–86]. Refer to [87] for a comprehensive review on the DG history and literature.

In the past decade, there has been an explosive interest in the DG method because of its attractive features. The DG method combines two unique characteristics of the FV and FE methods, i.e., the physics of wave propagation being accounted for by means of Riemann solvers, and high accuracy obtained using high-order polynomials within elements. Due to the compactness of the discretization, the DOFs associated with any element are coupled only with those of the neighboring elements sharing a face. This feature makes the DG method ideally suited for parallel computers. This compactness also results in highly sparse matrices in a linearized implicit time integration scheme.

To present the basic idea, consider the 2D Navier–Stokes equations (2). Again the approximate solution is a piece-wise polynomial of degree k or less with basis $\{\varphi_j(\vec{r})\}_{j=1}^m$. Therefore the solution on V_i is

$$Q(\vec{r},t) \approx p(\vec{r},t) = \sum_{j=1}^m u_j(t)\varphi_j(\vec{r}). \tag{49}$$

Applying the weighted residual form to element V_i, we obtain

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V_i} WQ\,\mathrm{d}V + \int_{\partial V_i} W\vec{F}(Q,\nabla Q)\cdot\vec{n}\,\mathrm{d}S - \int_{V_i}\nabla W\cdot\vec{F}(Q,\nabla Q)\,\mathrm{d}V = 0. \tag{50}$$

Because the solution is discontinuous across element interfaces, the normal flux across the interface is not uniquely defined. To couple the neighboring elements, the normal flux in (50) is replaced with a common numerical flux (the Riemann flux in the case of inviscid flow) computed based on the solutions and gradients on both element V_i and its neighbor

$$\vec{F}(Q,\nabla Q)\cdot\vec{n} \approx \hat{F}(Q^-,\nabla Q^-,Q^+,\nabla Q^+,\vec{n}), \tag{51}$$

where Q^- and ∇Q^- are the solution and gradients on V_i, Q^+ and ∇Q^+ are from the neighboring element. The inviscid flux is just the Riemann flux used in the FV method, depending on Q^-, Q^+ and the unit normal. The computation of the viscous flux has been addressed by many researchers [75,76,78,88,89], and it was shown a naïve treatment causes degradation in accuracy or loss of consistency. Here we present two popular approaches to compute viscous fluxes in the DG context, the local DG approach by Cockburn and Shu [90] and a compact approach due to Bassi and Rebay [76]. Introduce an auxiliary variable $\vec{G} = \nabla Q$. The Navier–Stokes equations become the following system of two first-order equations:

$$\vec{G} = \nabla Q,$$

$$\frac{\partial Q}{\partial t} + \nabla \cdot [\vec{F}_{\mathrm{i}}(Q) - \vec{F}_{\mathrm{v}}(Q, \vec{G})] = 0. \tag{52}$$

Applying the DG discretization to the these equations, we obtain

$$\int_{V_i} W\vec{G}\,\mathrm{d}V = -\int_{V_i} Q\nabla W\,\mathrm{d}V + \int_{\partial V_i} W\hat{Q}\vec{n}\,\mathrm{d}S, \tag{53}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V_i} WQ\,\mathrm{d}V + \int_{\partial V_i} W\hat{F}_{\mathrm{i}}(Q^-, Q^+, \vec{n})\,\mathrm{d}S + \int_{\partial V_i} W\vec{F}_{\mathrm{v}}(\hat{Q}, \hat{G})\cdot\vec{n}\,\mathrm{d}S - \int_{V_i} \nabla W \cdot \vec{F}(Q, \vec{G})\,\mathrm{d}V = 0. \tag{54}$$

In (53) and (54), since Q and $\vec{G}$ are discontinuous across element interfaces, they have been replaced with "numerical fluxes" $\hat{Q}$ and $\hat{G}$ in the surface integrals. In Bassi and Rebay original approach [75], simple averages were used for the numerical fluxes, i.e.,

$$\hat{Q} = (Q^- + Q^+)/2,$$

$$\hat{G} = (\vec{G}^- + \vec{G}^+)/2. \tag{55}$$

Cockburn and Shu [90] showed optimal convergence with the following choice of numerical fluxes:

$$\hat{Q} = Q^+,$$

$$\hat{G} = \vec{G}^-. \tag{56}$$

Bassi and Rebay later improved their original approach with a more compact and more accurate one outlined next [76]. Rewrite (53) as

$$\int_{V_i} W(\vec{G} - \nabla Q)\,\mathrm{d}V = \int_{\partial V_i} W(\hat{Q} - Q^-)\vec{n}\,\mathrm{d}S. \tag{57}$$

Define the "element" correction as $\vec{R} = \vec{G} - \nabla Q$. Then obviously we have

$$\int_{V_i} W\vec{R}\,\mathrm{d}V = \int_{\partial V_i} W(\hat{Q} - Q^-)\vec{n}\,\mathrm{d}S = \sum_{f\in\partial V_i}\int_f W(\hat{Q} - Q^-)\vec{n}\,\mathrm{d}S. \tag{58}$$

Let $\vec{R}_f$ be the correction due to face f computed according to

$$\int_{V_i} W\vec{R}_f\,\mathrm{d}V = \int_f W(\hat{Q} - Q^-)\vec{n}\,\mathrm{d}S. \tag{59}$$

Then the following relation between $\vec{R}$ and $\vec{R}_f$ holds:

$$\vec{R} = \sum_{f\in\partial V_i}\vec{R}_f. \tag{60}$$

Finally the numerical flux becomes

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_{V_i} WQ\,\mathrm{d}V + \int_{\partial V_i} W\hat{F}_{\mathrm{i}}(Q^-, Q^+, \vec{n})\,\mathrm{d}S + \sum_{f\in\partial V_i}\int_f W\hat{F}_v(Q^-, \nabla Q^- + \vec{R}_f^-, Q^+, \nabla Q^+ + \vec{R}_f^+, \vec{n})\,\mathrm{d}S - \int_{V_i}\nabla W\cdot\vec{F}(Q, \nabla Q + \vec{R})\,\mathrm{d}V = 0. \tag{61}$$

The viscous numerical flux is computed using simple averages

$$\hat{F}_v(Q^-, \nabla Q^- + \vec{R}_f^-, Q^+, \nabla Q^+ + \vec{R}_f^+, \vec{n}) = \tfrac{1}{2}[\vec{F}_{\mathrm{v}}(Q^-, \nabla Q^- + \vec{R}_f^-) + \vec{F}_{\mathrm{v}}(Q^+, \nabla Q^+ + \vec{R}_f^+)]\cdot\vec{n}. \tag{62}$$

Since the fluxes are usually non-linear functions of the state variable, Gauss quadrature formulas are used to compute both the surface and volume integrals. There is a precision requirement for these quadrature rules, i.e., the surface quadrature formula should be exact for degree $2k+1$ polynomials and the volume integral must be exact for degree $2k$ polynomials.

For high-order wall boundaries, the standard approach is to use high-order iso-parametric elements near solid walls [74]. This approach is quite complex to implement, especially in three dimensions. A much simpler approach was recently developed by Krivodonova and Berger [81], with

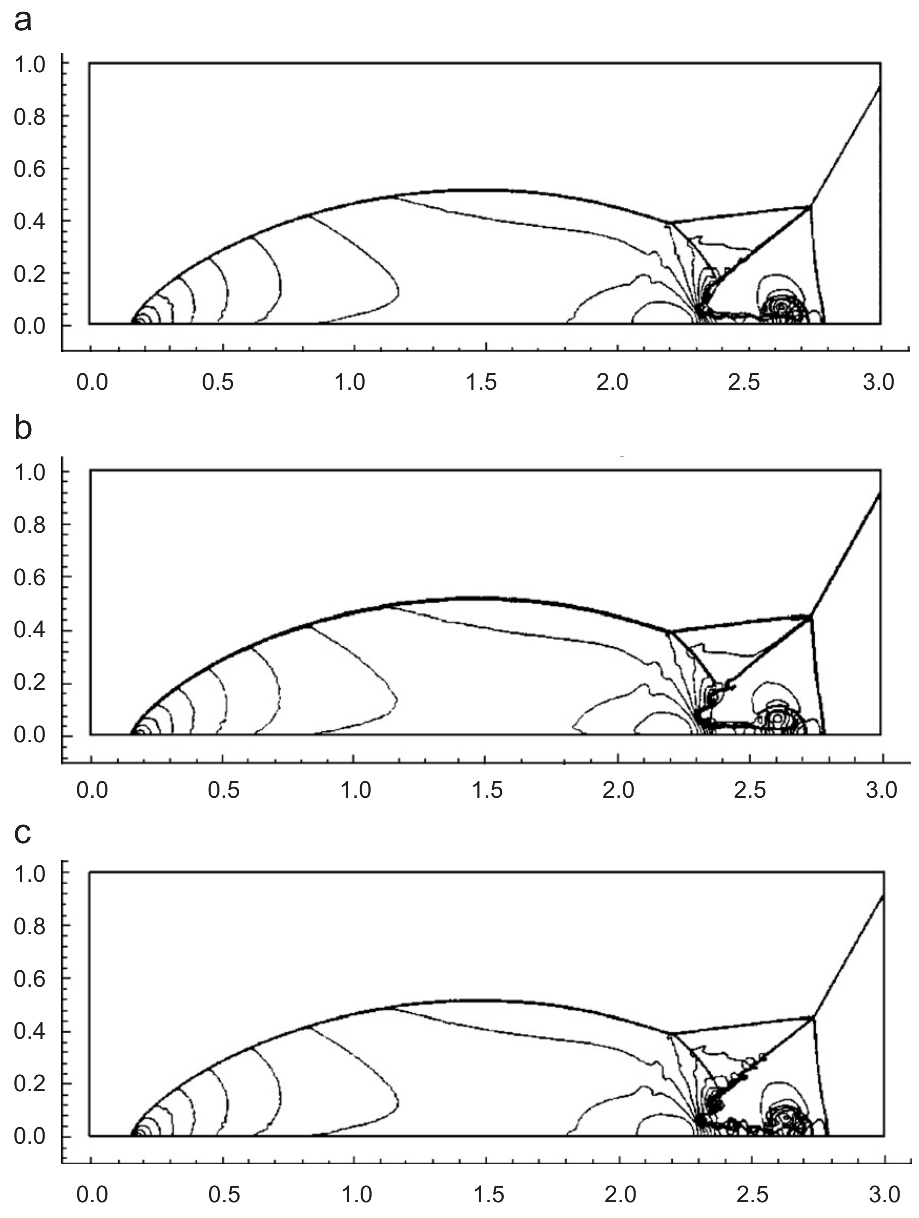


Fig. 6. Computed density contours using DG schemes for the double Mach reflection problem (courtesy of Cockburn and Shu [73]). (a) Second-order DG, $h=\frac{1}{480}$ (2.76 million DOFs), (b) third-order DG, $h=\frac{1}{240}$ (1.38 million DOFs), (c) third-order DG, $h=\frac{1}{480}$ (5.53 million DOFs).

strikingly good numerical results. In fact, this elegant approach was successfully implemented for the SV method [91].

Letting W be each of the basis functions, we obtain a system of equations about the DOFs. Denote $\tilde{u}$ the global DOFs, the equations can be cast in the following form:

$$\frac{\mathrm{d}\tilde{u}}{\mathrm{d}t}=-M^{-1}R(\tilde{u}), \tag{63}$$

where M is the global mass matrix. For compressible flow problems with discontinuities, data limiting becomes necessary. Various total variation diminishing (TVD) and total variation bounding (TVB) limiters [92] were developed by Cockburn and Shu [73] and are not repeated here.

The double Mach reflection problem was also tackled with the second- and third-order DG schemes [73], and selected results are shown in Figs. 6 and 7 for quadrilateral elements. The mesh size for the second-order simulation is $h=\frac{1}{480}$, resulting in a total NDOFs of 2.76 million. For the third-order scheme, results for two mesh sizes, $h=\frac{1}{240}$ and $\frac{1}{480}$, are shown, with 1.38 and 5.53 million DOFs, respectively. We can see that the third-order scheme on the coarse mesh has qualitatively the same

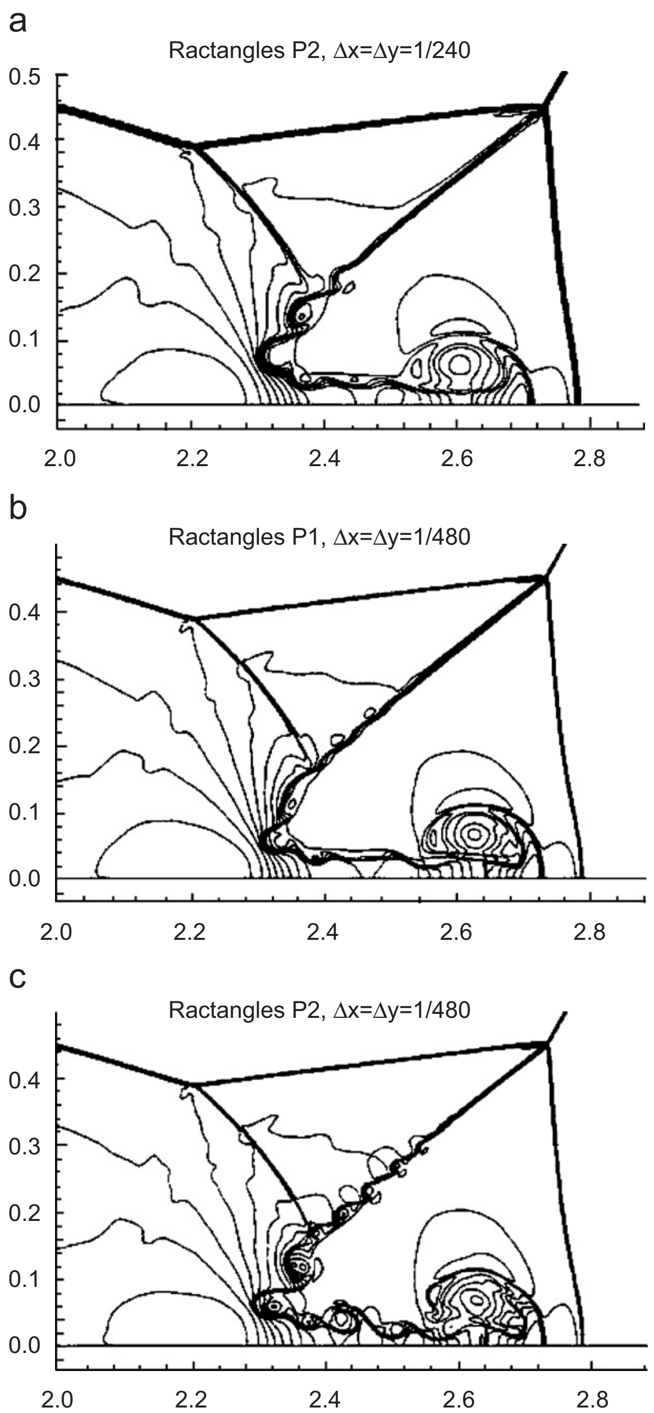


Fig. 7. Close-up view of the density contours computed with DG schemes for the double Mach reflection problem (courtesy of Cockburn and Shu [73]). (a) Third-order DG, $h = \frac{1}{240}$, (1.38 million DOFs), (b) second-order DG, $h = \frac{1}{480}$, (2.76 million DOFs), (c) third-order DG, $h = \frac{1}{480}$, (5.53 million DOFs).

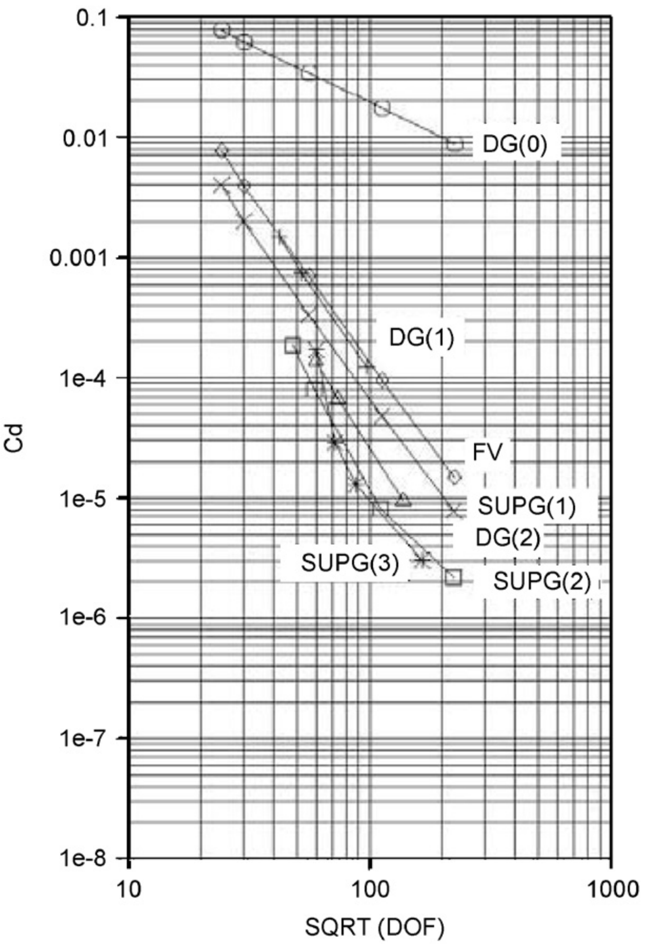


Fig. 8. Inviscid flow over an NACA0012 airfoil (courtesy of Venkatakrishnan et al. [69]).

resolution as the second-order scheme on the fine mesh for the fine details of the complicated near the Mach stem. The third-order scheme on the fine mesh gives a much better resolution for these structures than the second-order scheme with the same number of elements.

An extensive comparison of SUPG and DG was recently carried out by Venkatakrishnan et al. [69] for a variety of problems, and some very interesting accuracy results are presented here. The first case is an inviscid flow over the NACA0012 airfoil using the FV, SUPG and DG methods. The flow conditions are $M_\infty = 0.5$ and $\alpha = 0°$. Fig. 8 shows the computed C_d with the second-order FV, SUPG(1–3) (polynomial degree $k = 1-3$) and DG(0–2) (DG(0) being the Godunov method). Notice that all the methods show no better than second order accuracy as the grids are refined. The reason for this behavior is because of the slope discontinuity at the trailing edge, which limits the accuracy to at most second order. For the same polynomial order and NDOFs, SUPG appears to give lower drag. A similar simulation was also performed for a Joukowski airfoil, which has a cusp trailing edge. At zero angle of attack, there is no slope discontinuity at the trailing edge. The computed C_d is shown in Fig. 9. All methods seem

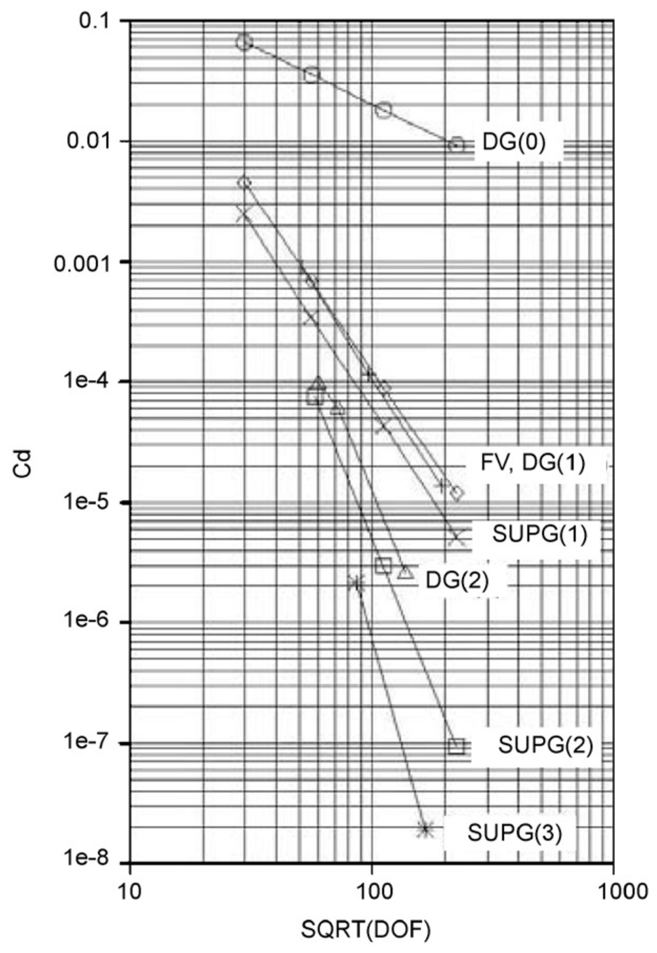


Fig. 9. Inviscid flow over a Joukowski airfoil (courtesy of Venkatakrishnan et al. [69]).

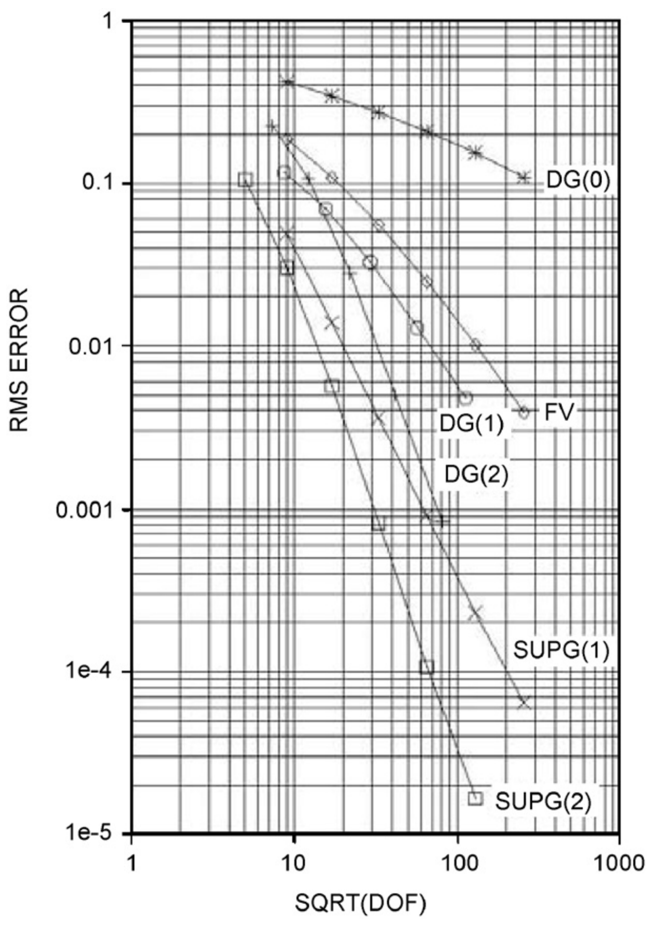


Fig. 10. High Reynolds number suction boundary layer, $Re = 10^6$ (courtesy of Venkatakrishnan et al. [69]).

to achieve $(2k+1)$th order accuracy. SUPG gives lower drag than the FV and DG methods. The third case is an accuracy study with a viscous flow problem—the suction boundary layer, which has an analytical solution [65]. The computed x-velocity error is displayed in Fig. 10 for a high Reynolds number of 10^6. SUPG(1) and SUPG(2) produce the expected $(k+1)$th order. DG(0–2) appears to lose half an order of accuracy. SUPG is again the most accurate.

The high-order DG method has been used by several researchers to compute turbulent flows [93,94]. One such computation is presented next. Bassi et al. [93] computed the unsteady vortex shedding behind a turbine blade using a k–ω turbulence model. The computational grid is displayed in Fig. 11. The grid has 5411 triangular elements and consists of a layer of anisotropic elements around the blade surface. The simulation started with a first-order DG scheme, and then proceeded to second- and third-order schemes. Fig. 12 shows a snapshot of the turbulence intensity fields of the third-order solution. In the figure the alternate vortex shedding structure and the marked widening of the turbulent wake are clearly evident. Fig. 13 presents the evolution of the density field during a vortex shedding cycle. The vortex shed by the pressure side appears to be stronger than that shed by the suction side. This is due to the different boundary layer development on the two sides of the blade.

3.4. RD methods

The RD (or fluctuation splitting [164]) methods refer to a class of numerical methods, which distribute a cell-based residual to the nodes forming the cell in order to update the nodal solutions, i.e., the DOFs. There are two major components in the RD method: the computation of the cell residual and the distribution of the cell residual. In the development of the RD methods, many ideas from FE, FD and FV methods were borrowed. The upwind RD methods were conceived by Roe [95] and then further developed in collaboration with Deconinck and collaborators [96–100]. The RD

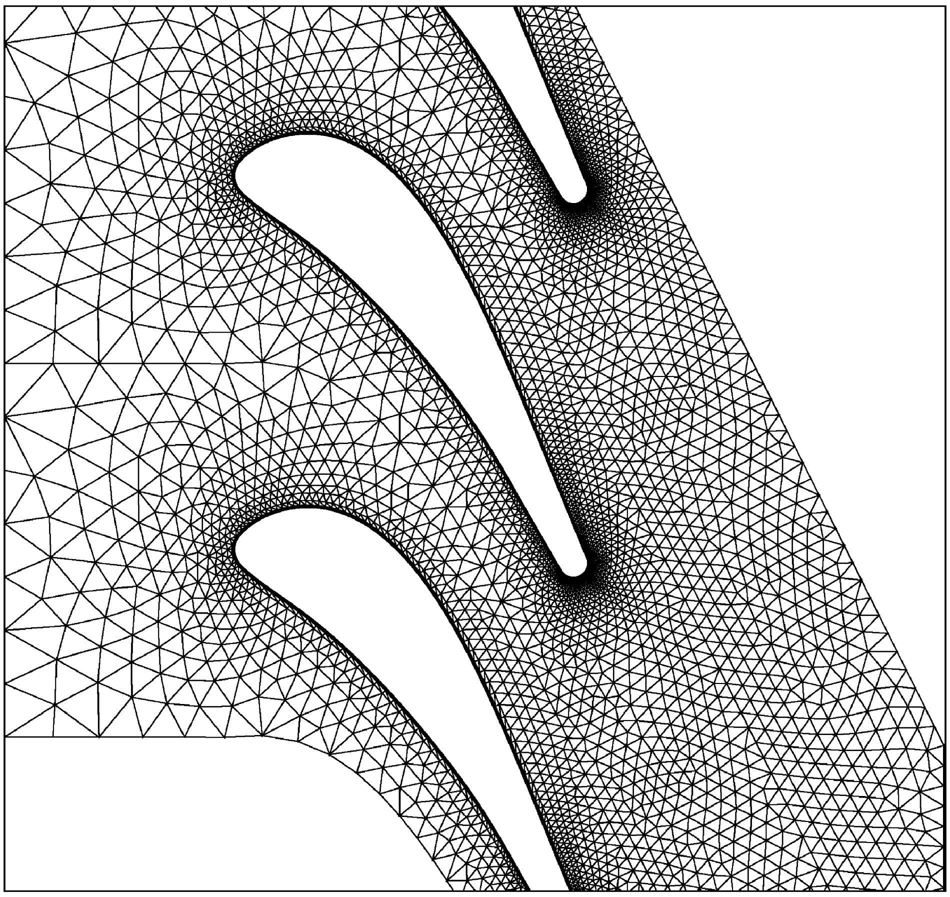


Fig. 11. Global view of the grid around turbine blades (courtesy of Bassi et al. [93]).

methods have been reported in the von Karman Institute Lecture Series [97,98]. Other research groups also made significant contributions [101,102]. More recently, Abgrall and Roe successfully extended the method to higher than second order [103]. The extensions to unsteady and viscous flows have been carried out by many researchers, e.g., in [98,104]. A comprehensive review of the RD methods is given by Abgrall in [105].

To fix the basic idea, consider the steady wave equation in 2D

$$\nabla \cdot (u\vec{a}) = 0. \tag{64}$$

Domain Ω is discretized into triangular elements, and the ith element V_i has vertices i_1, i_2 and i_3. The DOFs are the solutions at the vertices $\{u_j\}$. The cell residual for V_i is defined as

$$\Phi^i = \int_{V_i} \nabla \cdot (u\vec{a})\,\mathrm{d}V = \int_{\partial V_i} u\vec{a} \cdot \vec{n}\,\mathrm{d}S. \tag{65}$$

The cell residual is then distributed to all the nodes forming the cell in a conservative manner. Let the distribution of Φ^i to node i_j be Φ^i_j, with $j = 1, 2, 3$. Then the conservation condition is

$$\sum_{j=1}^{3} \Phi^i_j = \Phi^i. \tag{66}$$

For any given node r, the total update is related to the sum of the distributions from all the cells sharing the node

$$u_r \Leftarrow \sum_{i,r=i_j} \Phi^i_j. \tag{67}$$

When a steady-state solution is reached, all the cell residuals should vanish, and so does the update for each vertex, i.e.,

$$\sum_{i,r=i_j} \Phi^i_j = 0. \tag{68}$$

The system (68) is never solved directly, but via an iterative procedure. One simple example is given by

$$u_r^{n+1} = u_r^n - \frac{\Delta t}{|C_r|} \sum_{i,r=i_j} \Phi^i_j, \tag{69}$$

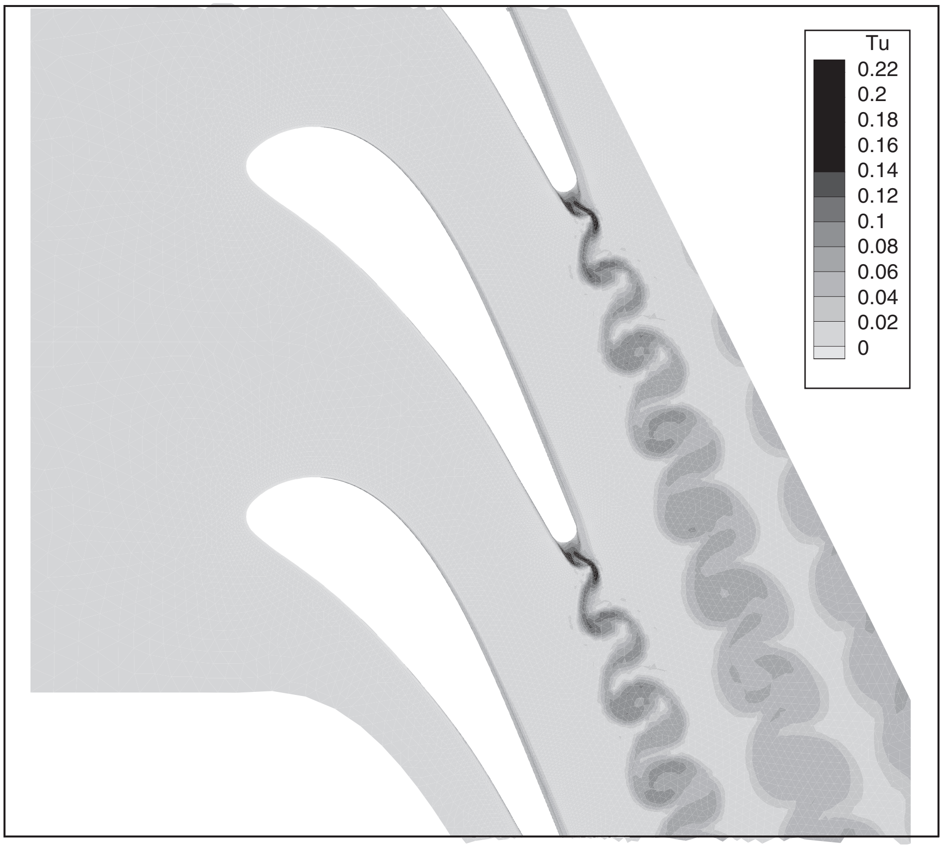


Fig. 12. Snapshot of the turbulent intensity using third-order DG (courtesy of Bassi et al. [93]).

where Δt is the pseudo-time step and $|C_r|$ is the area of the dual control volume surrounding node r. We hope to reach the steady state once enough time steps are marched. One can see the obvious connection of (69) with a vertex centered FV method, from which more ideas can be borrowed, such as local time stepping.

A wide variety of schemes have been developed following the RD philosophy. We briefly review some of the most important ones in the history of the RD methods. Assume the solution is linear within an element, the cell residual can be easily found to be

$$\Phi^i = \sum_{j=1}^{3} k_j u_j, \tag{70}$$

where $k_j = \frac{1}{2}\vec{a} \cdot \vec{n}_j$, and $\vec{n}_j$ is the inward normal vector of the side opposite to vertex j, whose norm is the length of the side. The N (narrow) scheme [96] distributes the cell residual in the following manner:

$$\Phi_j^i = k_j^+(u_j - \tilde{u}), \tag{71}$$

where $k_j^+ = \max(k_j, 0)$, $\tilde{u}$ is a weighted average of the following form:

$$\tilde{u} = \frac{\sum_j k_j^- u_j}{\sum_j k_j^-} \tag{72}$$

and $k_j^- = \min(k_j, 0)$. Eq. (71) can be interpreted in the following way. There are two possible types of triangles, the one target triangles and the two targets triangles, as shown in Fig. 14. In fact, since $\sum_{j=1}^{3} k_j = 0$, one or two of the k_j's must be positive, corresponding to the one or two target triangles. In the one target case, we assume $k_1 > 0$, and $k_2, k_3 < 0$, then $\Phi_1^i = \Phi^i$ and $\Phi_2^i, \Phi_3^i = 0$. In the two target case, we assume $k_1, k_2 > 0$, and $k_3 < 0$. Then $\Phi_3^i = 0$, and $\Phi_1^i = k_1(u_1 - u_3)$, $\Phi_2^i = k_2(u_2 - u_3)$.

The extension of the RD scheme to higher than second-order accuracy was first conducted by Abgrall and Roe [103]. Extra DOFs are added on a triangle to fit a higher than linear polynomial. For example, three edge points are added, in addition to the three vertices, to construct a quadratic polynomial on each triangle. A cell (or element) residual Φ^{T} can be computed based on the quadratic reconstruction. Then this residual must be

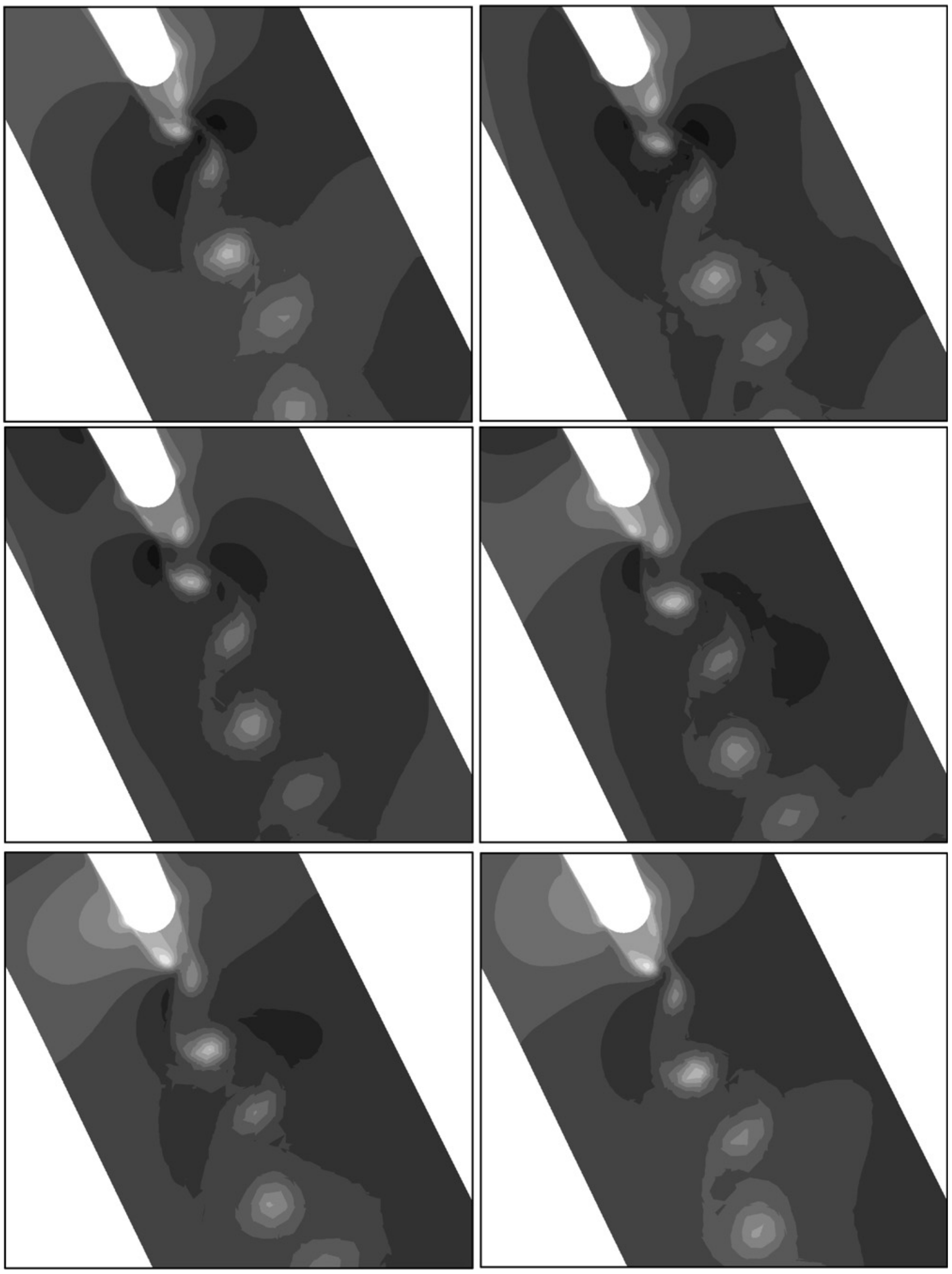


Fig. 13. Turbine blade density fields during a vortex shedding cycle, third-order DG (courtesy of Bassi et al. [93]).

distributed to all the DOFs in the cell in a conservative manner

$$\sum_{j=1}^{6} \Phi_j^T = \Phi^T. \tag{73}$$

In [103], it is shown that the scheme is third-order accurate (in 2D) if the residuals satisfy

$$\Phi_j^T = O(h^{2+2}). \tag{74}$$

In the d-dimensional case, for a $(k+1)$th order accurate scheme, the condition (74) is replaced by

$$\Phi_j^T = O(h^{d+k}). \tag{75}$$

The easiest way of fulfilling conditions (75) for $(k+1)$th order of accuracy is that $\Phi_j^T = \beta_j^T \Phi^T$ where β's are uniformly bounded. In [103], the following scheme was developed. Define 4 sub-triangles

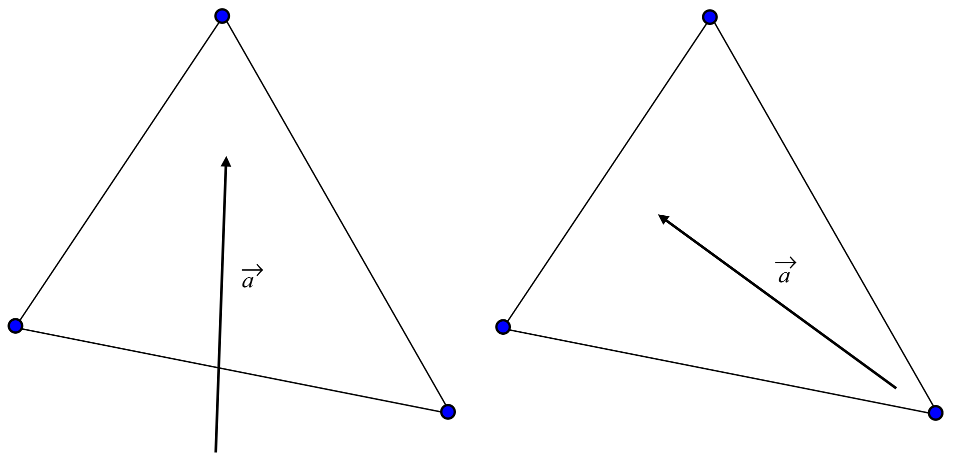


Fig. 14. One target and two target cases for the N scheme.

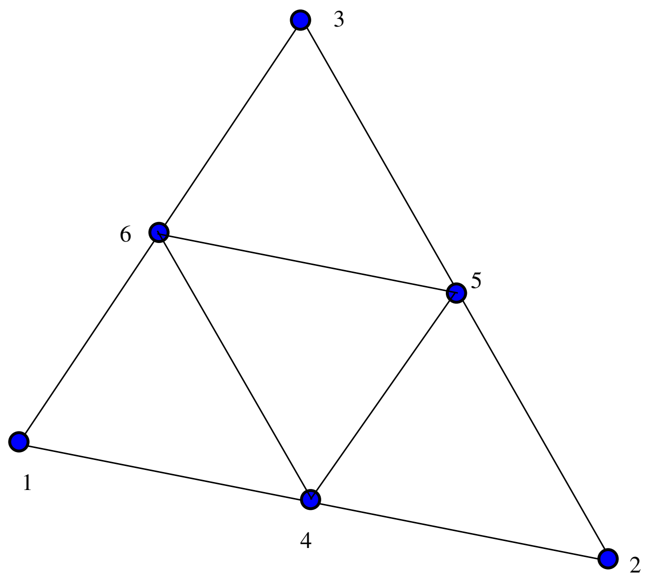


Fig. 15. A quadratic element in the RD method showing six DOFs.

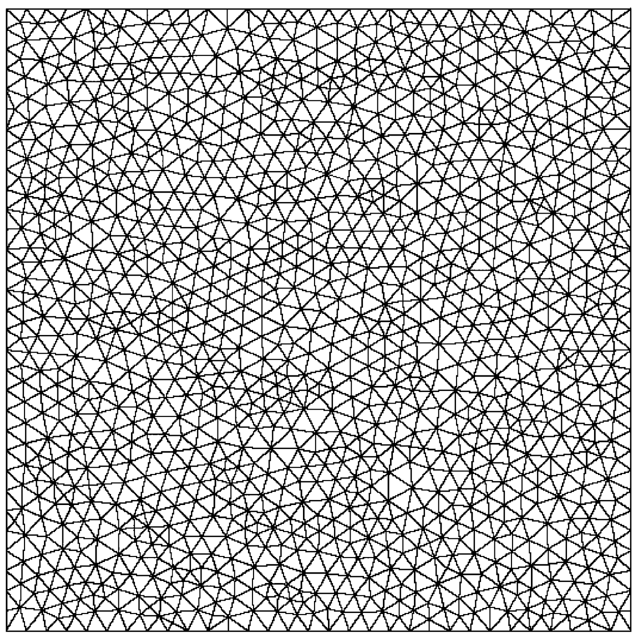


Fig. 16. Computational mesh for the Burger's equation (courtesy of Abgrall and Roe [103]).

$T_\xi = (1,4,6), (4,2,5), (5,3,6), (6,5,4)$, as shown in Fig. 15. In these four sub-triangles, we can consider the flowing high-order residuals based on the quadratic reconstruction

$$\Phi^{T_\xi} = \int_{T_\xi} \vec{a} \cdot \nabla u \, \mathrm{d}V. \tag{76}$$

Then in T_ξ, the N scheme is applied resulting in $\Phi_j^{N,\xi}$, and define

$$\beta_j^\xi = \frac{(\Phi_j^{N,\xi}/\Phi^{T_\xi})^+}{\sum_{l \in T_\xi} (\Phi_l^{N,\xi}/\Phi^{T_\xi})^+}. \tag{77}$$

Then define

$$\Phi_j^{T_\xi} = \beta_j^\xi \Phi^{T_\xi}. \tag{78}$$

Finally the residual sent by T to j is defined by

$$\Phi_j^T = \sum_{T_\xi, j \in T_\xi} \Phi_j^{T_\xi}. \tag{79}$$

As mentioned earlier, the RD method has been successfully extended to hyperbolic systems including the Euler and Navier–Stokes equations [98], and toto unsteady problems [104]. Interested readers can consult these references.

One sample computation with the high-order RD schemes from [103] is shown here. Consider the Burger equation

$$\frac{1}{2}\frac{\partial u^2}{\partial x} + \frac{\partial u}{\partial y} = 0, \quad x \in [0,1] \times [0,1],$$

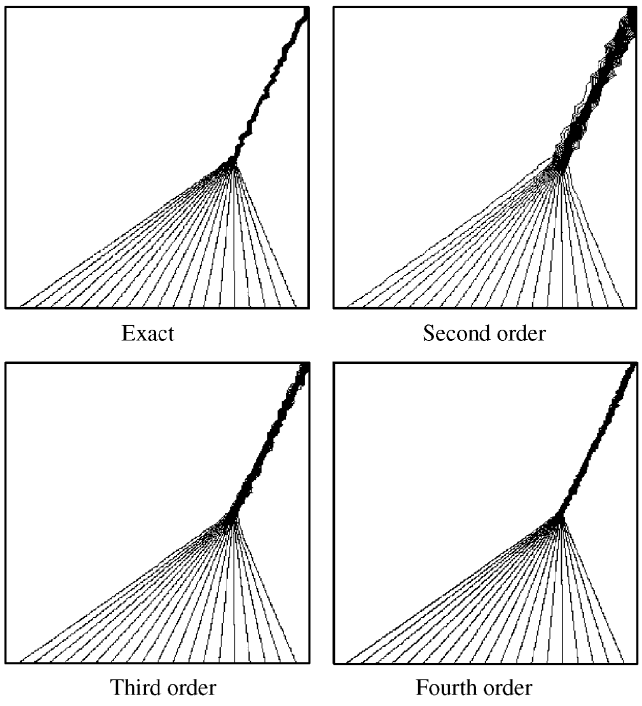


Fig. 17. Isolines of the exact and computed solutions for the Burger's equations using high-order RD schemes (courtesy of Abgrall and Roe [103]).

$u(x,y) = 1.5 - 2x$ on the inflow boundary.

The exact solution is

$$u(x,y) = \begin{cases} -0.5 & \text{if } y \leqslant 0.5 \text{ and } -2(x-3/4)+y-1/2 \leqslant 0, \\ 1.5 & \text{if } y \leqslant 0.5 \text{ and } -2(x-3/4)+y-1/2 \geqslant 0, \\ \max\left(-0.5, \min\left(1.5, \dfrac{x-3/4}{y-1/2}\right)\right) & \text{else.} \end{cases}$$

The computational grid is displayed in Fig. 16 with 1041 vertices and 1960 elements. The solution contours computed with second- to fourth-order RD schemes are compared with the exact solution in Fig. 17. Note that the solution is free of oscillations and the higher-order schemes do have higher resolution.

3.5. *SV and SD methods*

In this subsection, we briefly discuss the SV and SD methods, two recently developed and still evolving methods for compressible flow computation. Both the SV and SD methods employ the same solution space as the DG method, i.e., element-wise discontinuous polynomials. They differ on how the DOFs are updated. The SV method is similar to a FV method, while the SD method is close to a FD method.

The SV method was developed by Wang, Liu and their collaborators [22,23,106–109] for hyperbolic conservation laws. The SV method has been successfully extended to 2D Navier–Stokes equations [109], and 3D Maxwell equations [108]. Chen [110,111] developed many high-order SV partitions for simplexes in 2D and 3D with relatively small Lebesgue constants. Comparisons between the SV and DG methods were made by Sun and Wang [112] and by Zhang and Shu [113]. The SV method was also applied to solve the 3D Euler and Navier–Stokes equations by Haga et al. [114] on Japan's Earth Simulator. More recently, Van den Abeele et al. [115] and Van den Abeele and Lacor [116] performed Fourier analysis for both 1D and 2D SV methods, and identified a weak instability in several SV partitions. New partitions are derived which showed improved stability properties. In addition, Harris et al. [91] developed a more efficient quadrature free implementation for the SV method, which is expected to be significantly faster than the standard quadrature-based SV method, especially in 3D. Numerical tests in 2D have successfully demonstrated the new implementation.

The SD method was developed by Liu et al. [117,118] to further improve the efficiency and ease of implementation of high-order methods for simplexes. The SD method is easier to implement than the DG and SV methods because it does not involve surface or volume integrals, especially for high-order curved boundaries. The SD method was successfully extended to the Euler equations by Wang and Liu [119], and their collaborators [120], and to Navier–Stokes by May and Jameson [121], and Wang et al. [122]. May and Jameson obtained high-order convergence for shock waves in 1D with a special basis function using post-processing [123]. Huang et al. [124] implemented a p-adaptive space and time SD scheme, and demonstrated sharp discontinuity-capturing. The SD method in 1D and for 2D quadrilateral mesh is similar to the staggered-grid multidomain spectral method by Kopriva et al. [27,125].

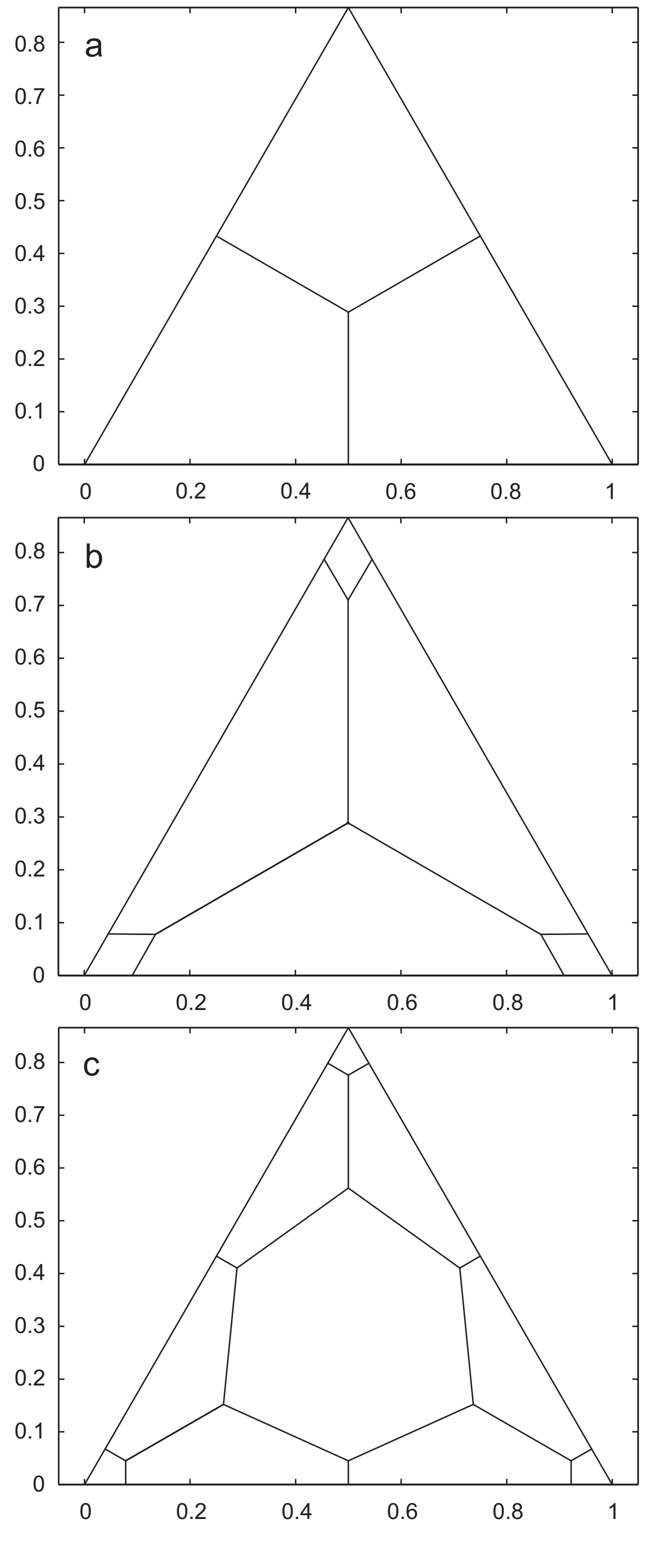


Fig. 18. Partitions of various orders in a triangular spectral volume, the third- and fourth-order partitions are found by Van den Abeele and Lacor [116]. (a) Second-order, (b) third-order, (c) fourth-order.

Therefore the SD method can be viewed the extension of the staggered-grid multidomain spectral method to simplexes.

3.5.1. SV method

In the SV method, each simplex element V_i is partitioned into m sub-cells called control volumes (CVs) $V_{j,i}$, as shown in Fig. 18. Applying the integral form of the governing equation on $V_{j,i}$ gives

$$\frac{\mathrm{d}\overline{Q}_{j,i}}{\mathrm{d}t}|V_{j,i}| + \int_{\partial V_{j,i}} \vec{F}(Q)\cdot\vec{n}\,\mathrm{d}S = 0, \tag{80}$$

where $|V_{j,i}|$ is the volume of $V_{j,i}$, and $\overline{Q}_{j,i}$ are the CV averaged conservative variables defined by

$$\overline{Q}_{j,i} = \frac{1}{|V_{j,i}|}\int_{V_{j,i}} Q\,\mathrm{d}V, \tag{81}$$

which are the DOFs. On a SV V_i, the DOFs are used to construct a degree k polynomial using

$$p_i(\vec{r}) = \sum_{j=1}^{m} \overline{L}_{j,i}(\vec{r})\overline{Q}_{j,i}. \tag{82}$$

The "shape function" like polynomials $\overline{L}_{j,i}(\vec{r})$ associated with $V_{j,i}$ should satisfy

$$\frac{1}{|V_{j,i}|}\int_{V_{j,i}} L_{l,i}\,\mathrm{d}V = \delta_{jl}, \tag{83}$$

where δ_{jl} is the Kronecker delta. Obviously, the reconstruction (82) satisfies the k-exactness property. In addition, the polynomial is an $(k+1)$th order approximation of the solution, if the solution is sufficiently smooth. On the interface between two SVs, the reconstructed solutions are generally not continuous. In this case, the normal flux is replaced with a Riemann flux, given in (16)

$$\frac{\mathrm{d}\overline{Q}_{j,i}}{\mathrm{d}t}|V_{j,i}| + \sum_{f\in\partial V_{j,i}} \int_f \hat{F}(Q^-, Q^+, \vec{n})\,\mathrm{d}S = 0. \tag{84}$$

For interior CV faces, the analytical fluxes are used since the solution is continuous inside the SV. The surface integral is usually computed using $(k+1)$th order Gauss quadrature formula, which is exact for degree k or less polynomials. If the governing equations are linear (such as the Maxwell equation), the surface integral can be computed exactly because the flux vector is also a degree k polynomial.

Although the extension of the SV method to three dimensions is straightforward theoretically, many interior polygonal faces will be generated, which require Gauss quadratures to compute the surface integrals. For example, a pentagonal face existing in the partition of a tetrahedral SV is split into three

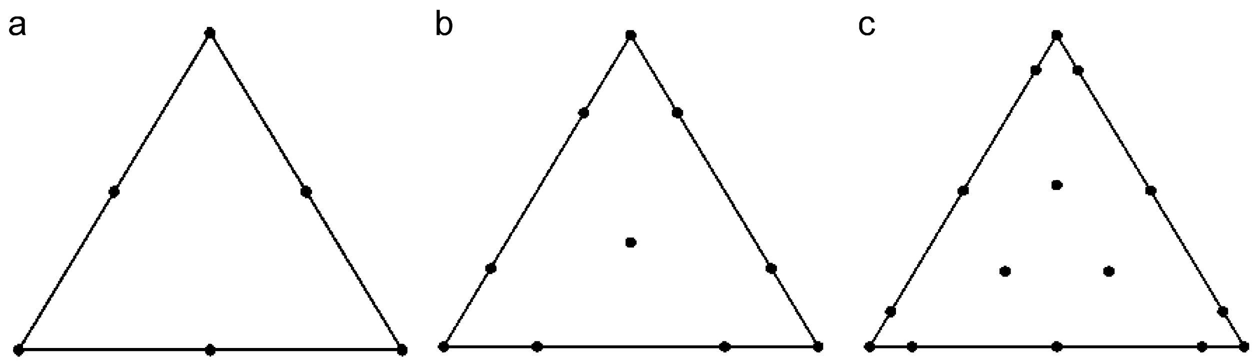


Fig. 19. Nodal sets in a triangular SV supporting quadratic, cubic and quartic data reconstructions for the flux vector, shown in (a), (b) and (c), respectively.

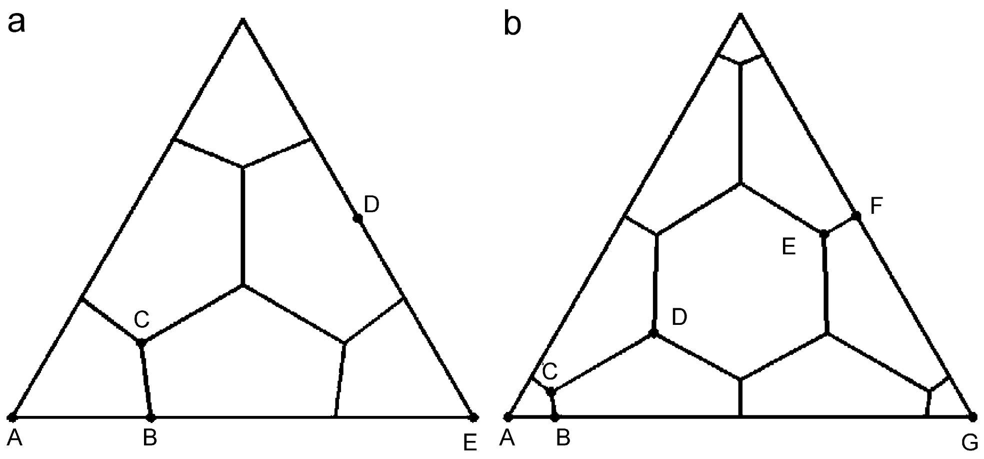


Fig. 20. General third- and fourth-order SV partitions. (a) Third-order, (b) fourth-order.

triangles. To carry out the integration, a Gauss quadrature formula of appropriate precision is then employed for each triangle [126]. The partition of a tetrahedron can be so complicated that hundreds or thousands of Gauss quadrature points per SV may be necessary to compute the face integrals to the desired precision, making the 3D SV method very expensive. To overcome this deficiency, a quadrature free approach has been developed [91]. In the new approach, a near optimal nodal set, such as those shown in Fig. 19, is selected from Hesthaven [127]. This nodal set is then used to reconstruct a degree $k+1$ polynomial approximation for the flux vector, and then the flux integrals are computed analytically, without the need for Gauss quadrature formulas. The flux vector $\vec{F}$ can be computed at any point using

$$\vec{F}(\vec{r}) = \sum_{j=1}^{m_{k+1}} Z_j(\vec{r})\vec{F}_j, \tag{85}$$

where $\vec{F}_j$ is the flux vector evaluated at node j, and $Z_j(r)$ are the Lagrange shape functions defined by the nodal set. This allows for the flux integral on each internal face to be computed as a weighted average of the flux evaluated at the nodal set, i.e.,

$$\int_f \vec{F} \cdot \vec{n}\, \mathrm{d}S = A_f \sum_{j=1}^{m_{k+1}} \overline{Z}_j \vec{F}_j \cdot \vec{n}_f, \tag{86}$$

where $\overline{Z}_j$ are the face-averaged shape functions for face f. In an actual implementation, face-averaged shape functions are computed during preprocessing for a standard element and then the physical face area is multiplied. For the SV-bounding faces, the Riemann flux integral can also be computed without the use of a Gauss quadrature. For example, (16) is integrated over each SV-bounding CV face, and the resulting face integral can be expressed as the integral of a Riemann flux as follows:

$$\int_f \hat{F}(Q^-, Q^+, \vec{n})\, \mathrm{d}S = \frac{A_f}{2}[\overline{F}^- + \overline{F}^+ - \lambda_f(\overline{Q}^+ - \overline{Q}^-)], \tag{87}$$

where

$$\overline{F}^- = \frac{1}{A_f}\int_f \vec{F}(Q^-)\cdot\vec{n}\,\mathrm{d}S,$$

$$\overline{F}^+ = \frac{1}{A_f}\int_f \vec{F}(Q^+)\cdot\vec{n}\,\mathrm{d}S,$$

$$\overline{Q}^- = \frac{1}{A_f}\int_f Q^-\,\mathrm{d}S, \quad \overline{Q}^+ = \frac{1}{A_f}\int_f Q^+\,\mathrm{d}S, \tag{88}$$

and λ_f is taken as the maximum absolute eigenvalue which is evaluated at the face center based on a average state. Analysis in [91] shows that this quadrature free approach preserves the accuracy of the SV method. Test cases for a variety of problems verified this accuracy analysis.

Van den Abeele and Lacor [116] recently identified a weak instability in several SV partitions, and proposed stable ones. For third- and fourth-order SV schemes (SV3 and SV4), the general partitions in 2D are shown in Fig. 20. The SV3 partition has two free parameters $\alpha_3 = |AB|/|AE|$, $\beta_3 = |AC|/|AD|$. The SV4 partition has four free parameters $\alpha_4 = |AB|/|AG|$, $\beta_4 = |AC|/|AF|$, $\chi_4 = |EF|/|AF|$, $\delta_4 = |AD|/|AF|$. The two stable SV3 and SV4 partitions are displayed in Fig. 18, and have the following parameters: $\alpha_3 = 0.091$, $\beta_3 = 0.18$; $\alpha_4 = 0.078$, $\beta_4 = 4\alpha_4/3$, $\chi_4 = \beta_4/2$ and $\delta_4 = 9\alpha_4/2$.

If the flow has discontinuities, data limiting is required in the reconstruction. In the SV method, limiters can be implemented at two levels, either the SV level or the CV (subcell) level. This is easily done because of the availability of subcell averages.

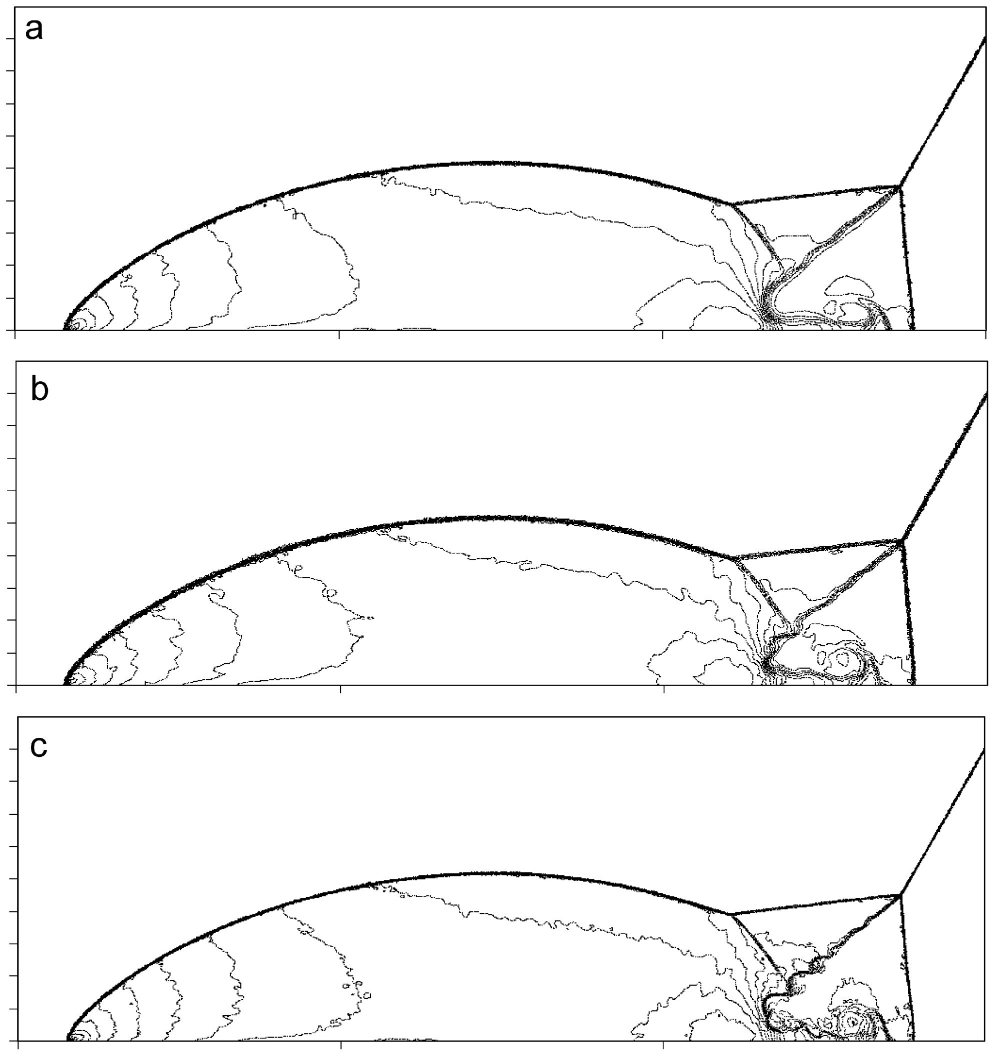


Fig. 21. Density contours computed using SV schemes with the Rusanov flux and TVD limiter, 30 even contours between 1.25 and 21.5 [107]. (a) Second-order SV, $h = \frac{1}{120}$ (392,384 DOFs), (b) third-order SV, $h = \frac{1}{60}$ (197,616 DOFs), (c) third-order SV, $h = \frac{1}{120}$ (784,968 DOFs).

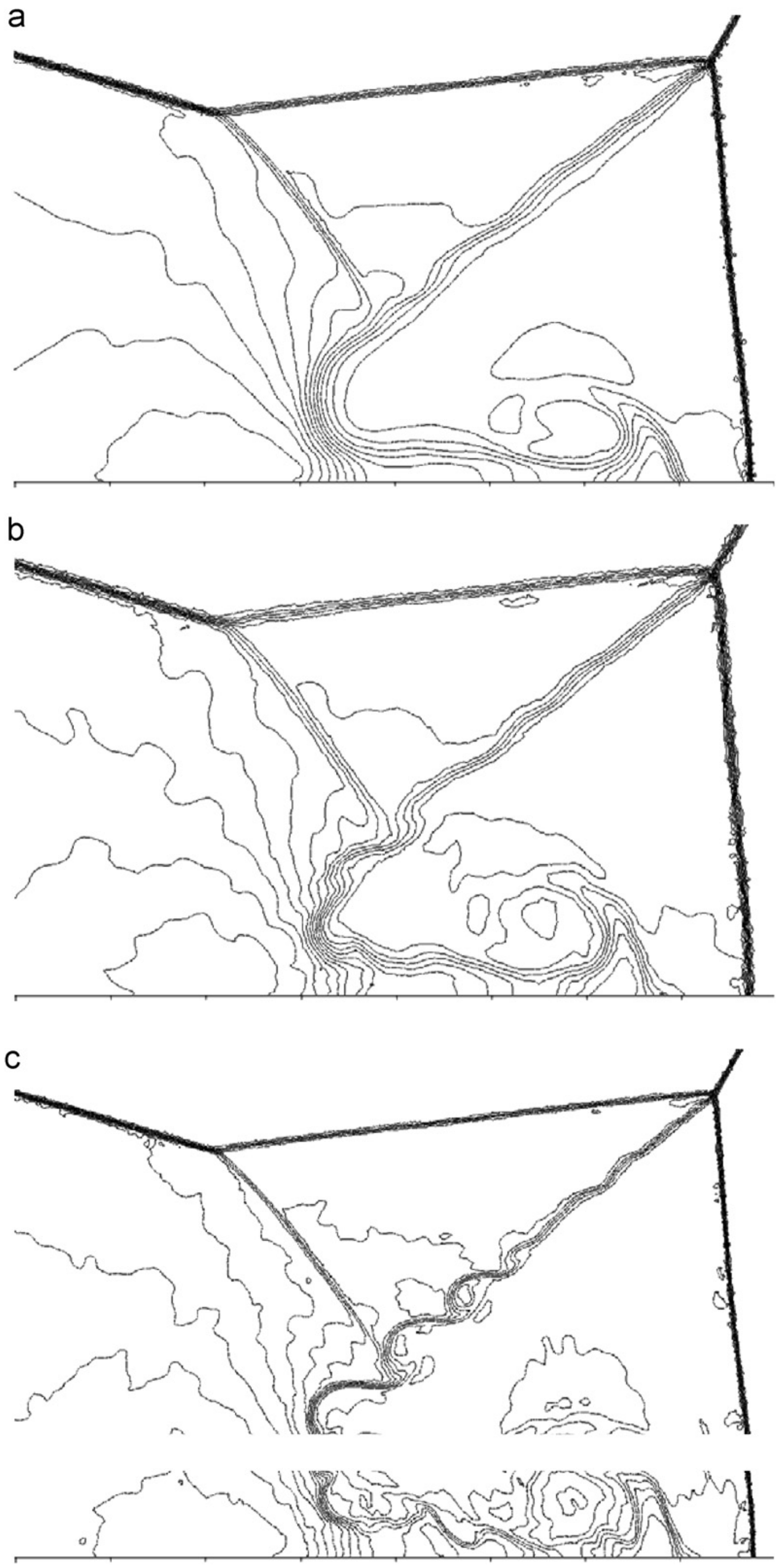


Fig. 22. Close-up view of the density contours near the double Mach stem [107]. (a) Second-order SV, $h = \frac{1}{120}$ (392,384 DOFs), (b) third-order SV, $h = \frac{1}{60}$ (197,616 DOFs), (c) third-order SV, $h = \frac{1}{120}$ (784,968 DOFs).

The subcell-based limiting can enhance the resolution for discontinuities. More details are contained in [107].

The extension of the SV method to the Navier–Stokes equation was discussed in [109]. An approach similar to the local DG scheme was employed to discretize the viscous flux. High-order accuracy was demonstrated for viscous flow.

The double Mach reflection case was also computed using the SV schemes. Two meshes with mesh size of $\frac{1}{60}$ and $\frac{1}{120}$ were used in the simulation. The coarse and fine computational grids have 32,936 and 130,828 cells, respectively, corresponding to 98,808, and 392,484 DOFs for the second-order SV scheme, and 197,616 and 784,968 DOFs for the third-order SV scheme. The density contours are displayed in Fig. 21. The close-up view is shown in Fig. 22. It is obvious that the third-order SV scheme has much higher resolution than the second-order SV scheme for the complex flow structures near the double Mach stem. In fact, the density contours computed with the third-order scheme on the coarse mesh display finer structures than those computed with the second-order scheme on the fine mesh, as shown in Fig. 22.

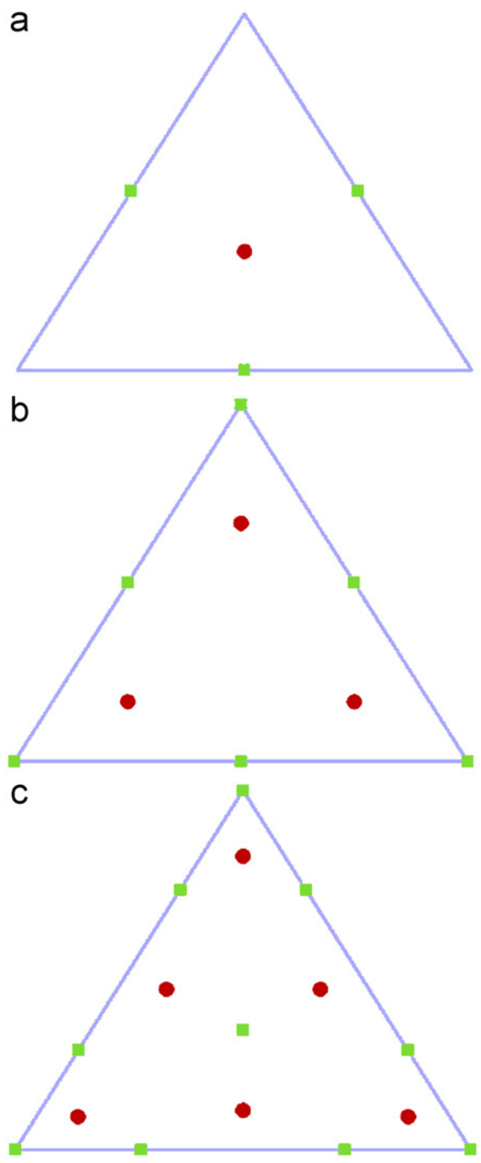


Fig. 23. Solution (solid circles) and flux points (solid squares) for first, second and third-order SD schemes. (a) First order, (b) second order, (c) third-order.

3.5.2. SD method

In a SD method, two sets of grid points, i.e., the *solution points* and *flux points* are defined in each element. The solution points are the locations where the nodal values of the conservative variables Q are specified (usually Gauss quadrature points). Flux points are the locations where the nodal values of fluxes are computed. The DOFs in the SD method are the conservative variables at the solution points. Fig. 23 displays the placements of solution and flux points for the first to third-order SD schemes [117]. Let the position vector of the jth solution point at cell i be denoted by $\vec{r}_{j,i}$, and the kth flux point at cell i be denoted by $\vec{r}_{k,i}$. Denote $Q_{j,i}$ the solution at $\vec{r}_{j,i}$. Given the solutions at $\vec{r}_{j,i}$, an element-wise degree k polynomial can be constructed using Lagrange-type polynomial basis, i.e.,

$$p_i(\vec{r}) = \sum_{j=1}^{m} L_{j,i}(\vec{r})Q_{j,i}, \tag{89}$$

where $L_{j,i}(\vec{r})$ are the cardinal basis functions. With (89), the solutions of Q at the flux points $\vec{r}_{k,i}$ can be computed easily. Since the solutions are discontinuous across element boundaries, the fluxes at the element interfaces are not uniquely defined. We again use approximate Riemann solvers to determine these fluxes. Obviously, in order to ensure conservation, the normal component of the flux vector on each face should be identical for the two cells sharing the face. To ensure conservation, a 1D Riemann solver is employed in the face normal direction to compute the common normal flux. Consider the face flux point shown in Fig. 24, and denote the outgoing normal from cell V_i to cell 1 $\vec{n}_1$. For this interface point, Q^- is computed from V_i and Q^+ is computed from cell 1. Then the common normal component of the flux can be computed with any Riemann solver $\hat{F}(Q^-, Q^+, \vec{n})$.

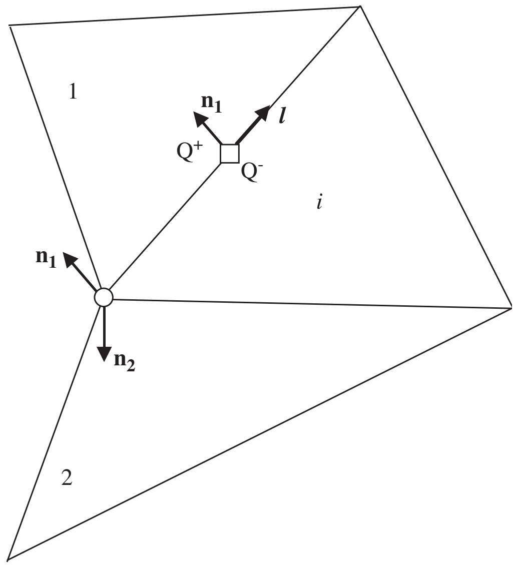


Fig. 24. Illustration of flux computation for face and corner points.

Since the tangential component of the flux does not affect the conservation property, we have the complete freedom in determining it at the face point. Let the unit vector in the tangential direction be $\vec{l}$. Here we offer two possibilities. One is to use a unique tangential component by averaging the two tangential components from both sides of the face, i.e.,

$$F_l = F_l(Q^-, Q^+, \vec{l}) = \tfrac{1}{2}\{[\vec{F}(Q^-) + \vec{F}(Q^+)] \cdot \vec{l}\}. \tag{90}$$

The other possibility is to use its own tangential component, allowing the tangential component to be discontinuous. Therefore, the tangential component of the flux on either side is not modified. For cell V_i, the tangential and normal components are $(\vec{F}(Q^-) \cdot \vec{l}, \hat{F})$, and for V_i's neighboring cell, they become $(\vec{F}(Q^+) \cdot \vec{l}, \hat{F})$.

For a corner flux point in cell V_i, two faces (from cell V_i) share the corner point, as shown in Fig. 24. Let the unit normals of the two faces be $\vec{n}_1$ and $\vec{n}_2$. Once again, the normal components of flux $\hat{F}_1$ and $\hat{F}_2$ in $\vec{n}_1$ and $\vec{n}_2$ directions are computed with a 1D Riemann solver in the normal directions. The full flux vector can then be uniquely determined from the two normal flux components

$$\vec{F} \cdot \vec{n}_1 = \hat{F}_1, \tag{91}$$

$$\vec{F} \cdot \vec{n}_2 = \hat{F}_2. \tag{92}$$

It is important to emphasize here that fluxes at cell corner points do not have unique values for all the cells sharing the corner. In spite of that, local conservation is guaranteed because neighboring cells do share a common normal flux at all the flux points. Once the fluxes at all the flux points are re-computed, they are used to form a degree $k+1$ polynomial, i.e.,

$$\vec{P}_i(\vec{r}) = \sum_{l=1}^{m_{k+1}} Z_{l,i}(\vec{r}) \vec{F}_{l,i}, \tag{93}$$

where $Z_{l,i}(\vec{r})$ are the set of cardinal basis functions defined by $\vec{r}_{l,i}$ and $\vec{F}_{l,i} = \vec{F}(\vec{r}_{l,i})$. Obviously, the divergence of the flux at any point within the cell can be computed using

$$\nabla \cdot \vec{P}_i(\vec{r}) = \sum_{l=1}^{m_{k+1}} \nabla Z_{l,i}(\vec{r}) \cdot \vec{F}_{l,i}. \tag{94}$$

To update the solutions at the solution points $\vec{r}_{j,i}$, we need to evaluate the divergence at these points, which can be easily computed according to

$$\nabla \cdot \vec{P}_i(\vec{r}_{j,i}) = \sum_{l=1}^{m_{k+1}} \nabla Z_{l,i}(\vec{r}_{j,i}) \cdot \vec{F}_{l,i}. \tag{95}$$

Finally the semi-discrete scheme to update the solution unknowns can be written as

$$\frac{\mathrm{d}Q_{j,i}}{\mathrm{d}t} + \sum_{l=1}^{m_{k+1}} \nabla Z_{l,i}(\vec{r}_{j,i}) \cdot \vec{F}_{l,i} = 0. \tag{96}$$

The SD method for quadrilateral or hexahedral grid is similar to the staggered grid multidomain spectral method [27,125]. It is particularly attractive because all the spatial operators are 1D in nature. In 2D, the solution and flux points are usually the Gauss and Gauss–Lobatto points, as shown in Fig. 25. Each physical element (possibly curved) is first transformed into a standard element (square). The governing equations are also transformed from the physical space into the computation space resulting in (6). In the standard element, the solution and flux points are defined by

$$X_s = \frac{1}{2}\left[1 - \cos\left(\frac{2s-1}{2N} \cdot \pi\right)\right], \quad s = 1, 2, \ldots, k+1, \tag{97}$$

$$X_{s+1/2} = \frac{1}{2}\left[1 - \cos\left(\frac{s}{N} \cdot \pi\right)\right], \quad s = 0, 1, \ldots, k+1. \tag{98}$$

Let the corresponding Lagrange interpolation shape functions be $h_i(X) = \prod_{s=1, s\neq i}^{k+1}((X - X_s)/(X_i - X_s))$

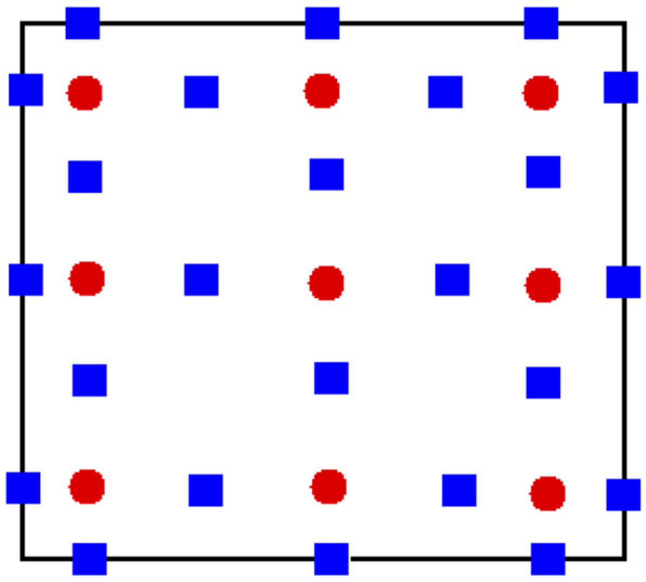


Fig. 25. Distribution of solution points (circles) and flux points (squares) in standard element for a third-order SD scheme.

and $l_{i+1/2}(X)=\prod_{s=0,s\neq i}^{k+1}((X-X_{s+1/2})/(X_{i+1/2}-X_{s+1/2}))$. The reconstructed solution for the conserved variables in the standard element is just the tensor products of the three 1D polynomials, i.e.,

$$Q(\xi,\eta,\varsigma)=\sum_{l=1}^{k+1}\sum_{j=1}^{k+1}\sum_{i=1}^{k+1}\frac{\tilde{Q}_{i,j,l}}{|J_{i,j,l}|}h_i(\xi)\cdot h_j(\eta)\cdot h_l(\varsigma). \tag{99}$$

Similarly, the reconstructed flux polynomials take the following form:

$$\tilde{F}(\xi,\eta,\varsigma)=\sum_{l=1}^{k+1}\sum_{j=1}^{k+1}\sum_{i=0}^{k+1}\tilde{F}_{i+1/2,j,l}l_{i+1/2}(\xi)\cdot h_j(\eta)\cdot h_l(\varsigma), \tag{100}$$

$$\tilde{G}(\xi,\eta,\varsigma)=\sum_{l=1}^{k+1}\sum_{j=0}^{k+1}\sum_{i=i}^{k+1}\tilde{G}_{i,j+1/2,l}h_i(\xi)\cdot l_{j+1/2}(\eta)\cdot h_l(\varsigma), \tag{101}$$

$$\tilde{H}(\xi,\eta,\varsigma)=\sum_{l=0}^{k+1}\sum_{j=1}^{k+1}\sum_{i=1}^{k+1}\tilde{H}_{i,j,l+1/2}h_i(\xi)\cdot h_j(\eta)\cdot l_{l+1/2}(\varsigma), \tag{102}$$

where $\tilde{F},\tilde{G},\tilde{H}$ are the components of $\vec{F}$. The reconstructed fluxes are only element-wise continuous, but discontinuous across cell interfaces. For the inviscid flux, a Riemann solver is employed to compute a common flux at interfaces to ensure conservation and stability. In summary, the algorithm to compute the inviscid flux derivatives consists of the following steps:

1. given the conserved variables at the solution points $\{\tilde{Q}_{i,j,l}\}$, compute the conserved variables at the flux points $\{Q_{i+1/2,j,l}, Q_{i,j+1/2,l}, Q_{i,j,l+1/2}\}$;
2. compute the inviscid fluxes at the interior flux points using the solutions computed at Step 1, and compute the inviscid flux at element interfaces using a Riemann solver;
3. compute the derivatives of the fluxes at all the solution points according to

$$\left(\frac{\partial\tilde{F}}{\partial\xi}\right)_{i,j,l}=\sum_{r=0}^{k+1}\tilde{F}_{r+1/2,j,l}\cdot l'_{r+1/2}(\xi_i), \tag{103}$$

$$\left(\frac{\partial\tilde{G}}{\partial\eta}\right)_{i,j,l}=\sum_{r=0}^{k+1}\tilde{G}_{i,r+1/2,l}\cdot l'_{r+1/2}(\eta_j), \tag{104}$$

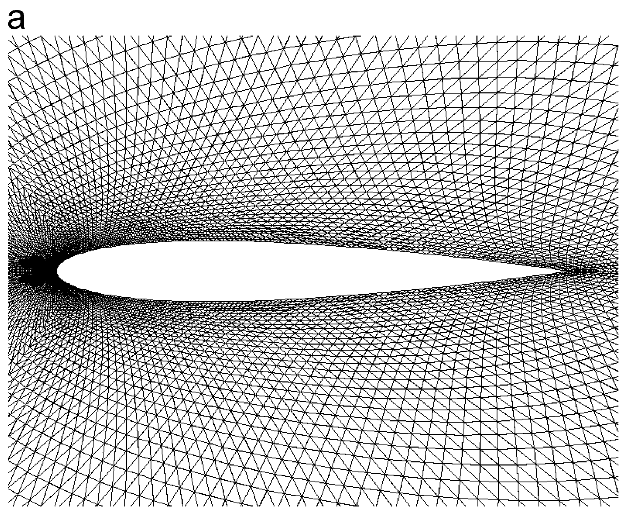
$$\left(\frac{\partial\tilde{H}}{\partial\varsigma}\right)_{i,j,l}=\sum_{r=0}^{k+1}\tilde{H}_{i,j,r+1/2}\cdot l'_{r+1/2}(\varsigma_l), \tag{105}$$

4. use the above derivatives to update the DOFs, i.e.,

$$\frac{\partial\tilde{Q}_{i,j,l}}{\partial t}=-\left(\frac{\partial\tilde{F}}{\partial\xi}+\frac{\partial\tilde{G}}{\partial\eta}+\frac{\partial\tilde{H}}{\partial\varsigma}\right)_{i,j,l}. \tag{106}$$

Two example simulations with the SD method are presented next. The first one is a subsonic flow around a NACA0012 airfoil at Mach = 0.4, and angle of attack of 5°. In this simulation, the computational results using the third-order SD scheme on a coarse mesh with 72 × 24 × 2 triangles are compared with those using a second order FV method on a much finer mesh of 192 × 64 × 2 triangles. Therefore the number of DOFs in the FV simulation is 24,576 while it is 20,736 in the SD simulation. The computational meshes used for both the SD and FV methods are displayed in Fig. 26. The average entropy error with the second-order FV method is 1.04e − 5, while the average entropy error with the third-order SD scheme is 4.86e − 6, which is more than a factor of 2 smaller. The entropy errors along the airfoil surface are plotted in Fig. 27. Note that although the second-order FV scheme used a much finer grid, the solution quality of the third-order SD scheme is superior.

The second example is a viscous flow around a sphere, computed using a hexahedral grid with 6144 cells. The Reynolds number based on the diameter was chosen to be 118. The flow was simulated using the



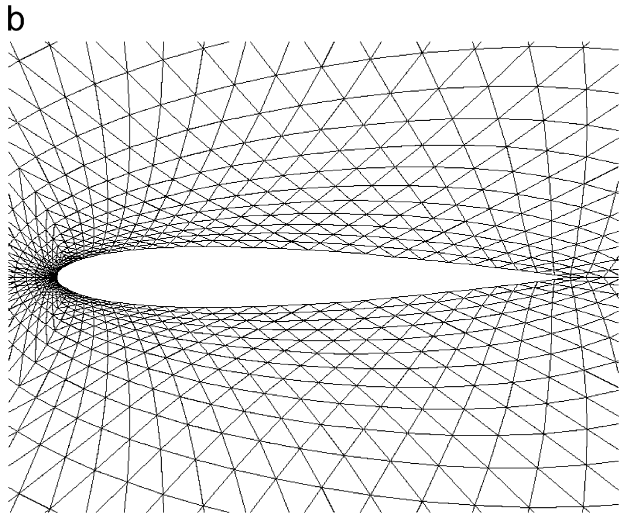


Fig. 26. Computational grids for subsonic flow around a NACA0012 airfoil [120]. (a) Second-order FV grid, (b) third-order SD grid.

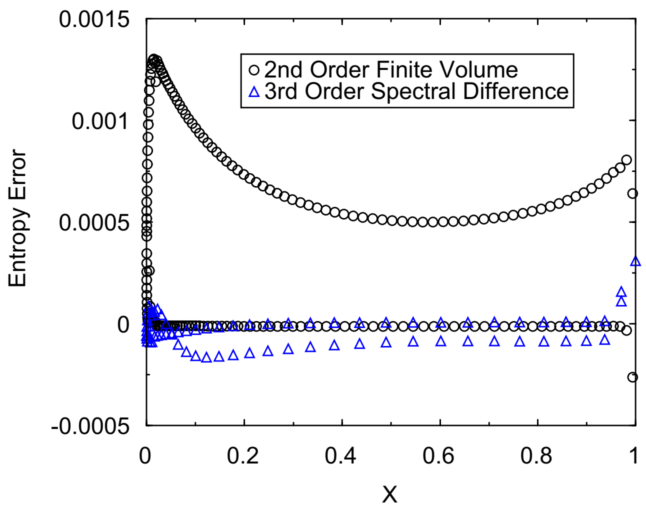


Fig. 27. Comparison of entropy error along the airfoil surface [120].

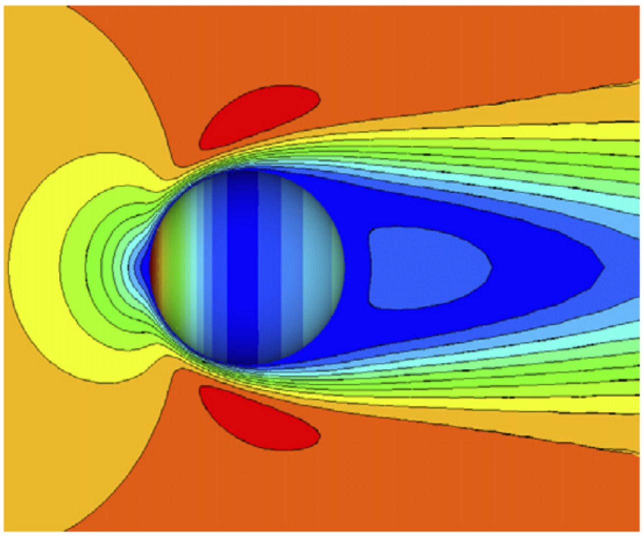


Fig. 28. Computed pressure (on the sphere) and Mach number (on $z = 0$ plane) distributions using the fourth- and sixth-order SD schemes. Mach contours start at Mach $= \frac{1}{40}$ with a $\frac{1}{40}$ interval [128].

fourth- and sixth-order SD schemes to assess the numerical error. The computed Mach number contours using both schemes are shown in Fig. 28. Note that the contours are nearly on top of each other, indicating an excellent agreement. The computational streamlines are compared with the experimental streamlines in Fig. 29. The size of the separation region in the computation agrees very well with that of the experiment, at least in the "eye ball" norm. The skin friction coefficients at the wall computed with both schemes are plotted in Fig. 30, and they are right on top of each other. In fact, the skin friction coefficients differ less than 0.1% between the fourth- and sixth-order results. The predicted separation angle from both schemes is 123.6° (the wind side stagnation point has an angle of 0), and the length of the separation region is 1.04D.

4. Time integration/iterative solution approaches

It is well known that high-order spatial operators are much stiffer than lower-order ones. For time accurate problems, the allowable CFL number decreases with increasing order of accuracy for explicit schemes. For viscous problems with highly clustered mesh to resolve the viscous boundary layer, explicit high-order methods are severely limited by the time step size, and usually not competitive against low-order implicit methods in terms of efficiency. A major pacing item in the CFD community is the

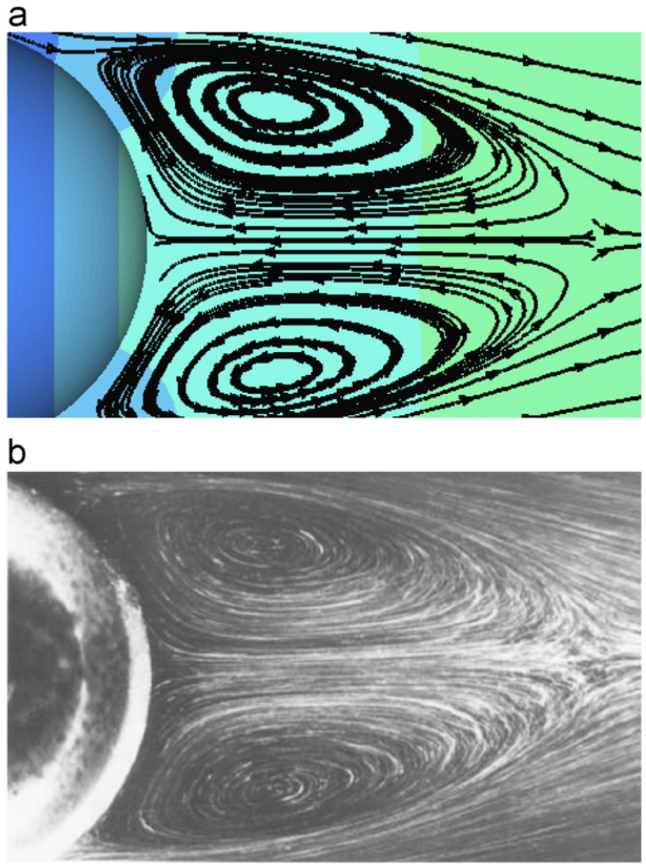


Fig. 29. Comparison of streamlines of the flow field between computation and experiment [128]. (a) Computation, (b) experiment.

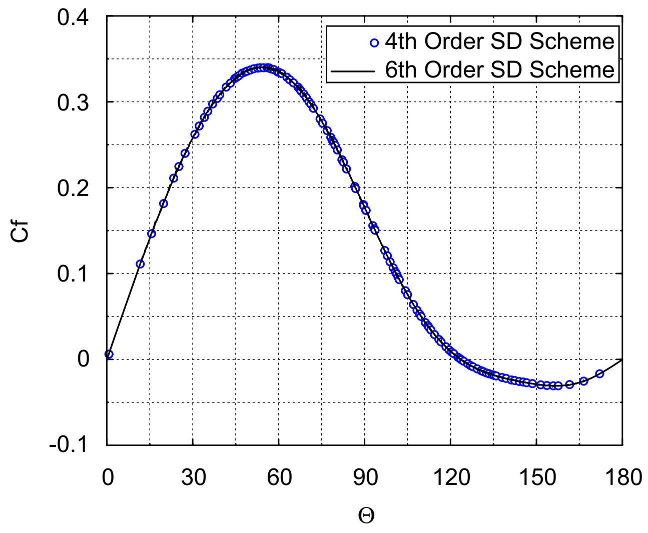


Fig. 30. Predicted skin friction coefficient profiles with the fourth- and sixth-order SD schemes [128].

development of efficient time integration/iterative solution approaches for high-order methods, several of which are reviewed in the next subsections.

Consider the following semi-discretized equation with any of the spatial operators discussed earlier

$$M\frac{\mathrm{d}\tilde{u}}{\mathrm{d}t}+R(\tilde{u})=0, \tag{107}$$

where $\tilde{u}$ represents the global DOFs, M is a "mass matrix", and R is the spatial residual operator. The steady solution is obtained from $R(\tilde{u})=0$. In the FV, SV and SD methods, matrix M is an identity matrix. In the DG method, M is a block-diagonal matrix and can be easily inverted. In continuous FE methods, matrix M is a globally coupled matrix and a large system of equations must be solved even with "explicit" time integration schemes.

4.1. Explicit Runge–Kutta methods

If M is easily invertible, (107) can be re-cast in the following form:

$$\frac{\mathrm{d}\tilde{u}}{\mathrm{d}t}=-M^{-1}R(\tilde{u}). \tag{108}$$

Many well-known explicit Runge–Kutta schemes can be used to march (108) in time. Several widely used ones are presented next.

4.1.1. Fourth-order four-stage Runge–Kutta scheme

This time integration scheme was first made popular by the landmark paper of Jameson et al. [129] because of its large stability limit. Later it was used in many production CFD codes. The scheme can be written as

$$\begin{aligned}
&\tilde{u}^{(0)}=\tilde{u}^{n},\\
&\tilde{u}^{(1)}=\tilde{u}^{(0)}-\frac{\Delta t}{2}M^{-1}R(\tilde{u}^{(0)}),\\
&\tilde{u}^{(2)}=\tilde{u}^{(0)}-\frac{\Delta t}{2}M^{-1}R(\tilde{u}^{(1)}),\\
&\tilde{u}^{(3)}=\tilde{u}^{(0)}-\Delta t M^{-1}R(\tilde{u}^{(2)}),\\
&\tilde{u}^{(4)}=\tilde{u}^{(0)}-\frac{\Delta t}{6}M^{-1}[R(\tilde{u}^{(0)})+2R(\tilde{u}^{(1)})\\
&\qquad +2R(\tilde{u}^{(2)})+R(\tilde{u}^{(3)})],\\
&\tilde{u}^{n+1}=\tilde{u}^{(4)}.
\end{aligned} \tag{109}$$

Jameson [130] also found the following low storage four-stage scheme effective for his multigrid flow solver:

$$\tilde{u}^{(0)}=\tilde{u}^{n},$$

$$\tilde{u}^{(1)} = \tilde{u}^{(0)} - \frac{\Delta t}{3} M^{-1} R(\tilde{u}^{(0)}),$$

$$\tilde{u}^{(2)} = \tilde{u}^{(0)} - \frac{4\Delta t}{15} M^{-1} R(\tilde{u}^{(1)}),$$

$$\tilde{u}^{(3)} = \tilde{u}^{(0)} - \frac{5\Delta t}{9} M^{-1} R(\tilde{u}^{(2)}),$$

$$\tilde{u}^{(4)} = \tilde{u}^{(0)} - \Delta t M^{-1} R(\tilde{u}^{(3)}),$$

$$\tilde{u}^{n+1} = \tilde{u}^{(4)}. \tag{110}$$

Other variations are also possible, and have been used extensively in the CFD community.

4.1.2. *Strong-stability-preserving (SSP) Runge–Kutta schemes*

This class of time integration method was originally developed by Shu [131], and Shu and Osher [132] and named TVD Runge–Kutta schemes. It was further studied by many researchers, e.g., in [133,134]. These schemes preserve the stability properties of forward Euler in any norm or semi-norm. They have been popular for high-order spatial operators because of its TVD or SSP property. The coefficients of these schemes are not unique. Optimal versions with maximum CFL numbers have been derived in [131] for the second- and third-order schemes, and by Spiteri and Ruuth [134] for the fourth-order counterpart. More details can be found in a recent review article by Gottlieb [135]. The optimum second to fourth-order schemes are presented next.

SSPRK (2, 2): Two stage, second-order

$$\tilde{u}^{(1)} = \tilde{u}^{n} - \Delta t M^{-1} R(\tilde{u}^{n}),$$

$$\tilde{u}^{n+1} = \tfrac{1}{2}\tilde{u}^{n} + \tfrac{1}{2}\tilde{u}^{(1)} - \tfrac{1}{2}\Delta t M^{-1} R(\tilde{u}^{(1)}). \tag{111}$$

SSPRK (3, 3): Three stage, third-order

$$\tilde{u}^{(1)} = \tilde{u}^{n} - \Delta t M^{-1} R(\tilde{u}^{n}),$$

$$\tilde{u}^{(2)} = \tfrac{3}{4}\tilde{u}^{n} + \tfrac{1}{4}\tilde{u}^{(1)} - \tfrac{1}{4}\Delta t M^{-1} R(\tilde{u}^{(1)}),$$

$$\tilde{u}^{n+1} = \tfrac{1}{3}\tilde{u}^{n} + \tfrac{2}{3}\tilde{u}^{(2)} - \tfrac{2}{3}\Delta t M^{-1} R(\tilde{u}^{(2)}). \tag{112}$$

SSPRK (5, 4): Five stage, fourth-order

$$\tilde{u}^{(1)} = \tilde{u}^{n} - 0.391752226571890\Delta t M^{-1} R(\tilde{u}^{n}),$$

$$\tilde{u}^{(2)} = 0.444370493651235\tilde{u}^{n} + 0.555629506348765\tilde{u}^{(1)} - 0.368410593050371\Delta t M^{-1} R(\tilde{u}^{(1)}),$$

$$\tilde{u}^{(3)} = 0.620101851488403\tilde{u}^{n} + 0.379898148511597\tilde{u}^{(2)} - 0.251891774271694\Delta t M^{-1} R(\tilde{u}^{(2)}),$$

$$\tilde{u}^{(4)} = 0.178079954393132\tilde{u}^{n} + 0.821920045606868\tilde{u}^{(3)} - 0.544974750228521\Delta t M^{-1} R(\tilde{u}^{(3)}),$$

$$\begin{aligned}\tilde{u}^{n+1} = {} & 0.517231671970585\tilde{u}^{(2)} \\ & + 0.096059710526147\tilde{u}^{(3)} \\ & - 0.063692468666290\Delta t M^{-1} R(\tilde{u}^{(3)}) \\ & + 0.386708617503269\tilde{u}^{(4)} \\ & - 0.226007483236906\Delta t M^{-1} R(\tilde{u}^{(4)}).\end{aligned} \tag{113}$$

4.2. *Implicit methods*

It has long been recognized that implicit algorithms are necessary to overcome the time step limit suffered by explicit algorithms especially for highly clustered viscous meshes [136]. Many types of implicit algorithms have been successfully developed for unstructured grid-based solvers in the last two decades, e.g., the element Jacobi, Gauss–Seidel, precondition GMRES [93,137,166], matrix-free Krylov [138], lower–upper symmetry Gauss–Seidel [62,139], and line-implicit algorithms [140]. In addition, these implicit algorithms can serve as "smoothers" for geometric or *p*-multigrid approaches. Many of these algorithms have been successfully applied to high-order spatial discretizations described earlier. In almost all implicit approaches, the non-linear system of equations is linearized and then solved with an iterative solution algorithm. Using the backward Euler scheme for time integration, we obtain the following non-linear system:

$$M\frac{\tilde{u}^{n+1} - \tilde{u}^{n}}{\Delta t} + R(\tilde{u}^{n+1}) = 0, \tag{114}$$

which can be linearized to give

$$\mathbf{A}\Delta\tilde{u} = -R(\tilde{u}^{n}), \tag{115}$$

where $\Delta\tilde{u} = \tilde{u}^{n+1} - \tilde{u}^{n}$ and $\mathbf{A} = (M/\Delta t + \partial R/\partial\tilde{u})$. In the limit of $\Delta t \to \infty$, (115) becomes the Newton method for the non-linear system $R(\tilde{u}^{n+1}) = 0$. The left-hand side implicit operator is usually a large sparse block matrix, whose direct inversion is often too expensive. This large matrix can be partitioned into the block diagonal, lower and upper matrices

$$\mathbf{A} = \mathbf{D} + \mathbf{L} + \mathbf{U}.$$

Many well established algorithms for linear systems can be used to solve (115). Several of those are described next.

4.2.1. Block Jacobi and Gauss–Seidel

The block Jacobi (BJ) and Gauss–Seidel (GS) methods can be applied to solve (115) in at least two possible manners. The first one is to solve the linearized equation iteratively until a given convergence tolerance or a maximum number of sweeps is achieved. For example, the Jacobi iteration takes the following form:

$$\mathbf{D}\Delta\tilde{u}^{(k+1)} = -R(\tilde{u}^n) - (\mathbf{L}+\mathbf{U})\Delta\tilde{u}^{(k)}, \tag{116}$$

where k is an iteration index and $\Delta\tilde{u}^{(0)} = 0$. This approach was sometimes called the linearized element Jacobi method. Similarly, the GS approach is

$$\begin{aligned}\Delta\tilde{u}^{(k+1)} &= \mathbf{D}^{-1}(-R(\tilde{u}^n) - \mathbf{L}\Delta\tilde{u}^{(k+1)} - \mathbf{U}\Delta\tilde{u}^{(k)}) \\ &= (\mathbf{D}+\mathbf{L})^{-1}(-R(\tilde{u}^n) - \mathbf{U}\Delta\tilde{u}^{(k)}).\end{aligned} \tag{117}$$

Much better performance than the BJ method was demonstrated in [141] for a high-order DG implementation. Generally, faster convergence may be achieved by a symmetric GS (SGS) variation, in which both forward and backward sweeps are applied alternately. After the forward sweep given in (117), the following backward sweep is applied by reversing the order from element N to 1.

$$\begin{aligned}\Delta\tilde{u}^{(k+1)} &= \mathbf{D}^{-1}(-R(\tilde{u}^n) - \mathbf{U}\Delta\tilde{u}^{(k+1)} - \mathbf{L}\Delta\tilde{u}^{(k)}) \\ &= (\mathbf{D}+\mathbf{U})^{-1}(-R(\tilde{u}^n) - \mathbf{L}\Delta\tilde{u}^{(k)}).\end{aligned} \tag{118}$$

The second approach is to apply (116) once and then proceed to the next time step, i.e.,

$$\mathbf{D}\Delta\tilde{u} = -R(\tilde{u}^n) \tag{119}$$

which is usually called the "element implicit" approach, or the non-linear element Jacobi approach. The corresponding low storage GS version is then

$$\Delta\tilde{u} = -\mathbf{D}^{-1}R(\tilde{u}^*), \tag{120}$$

where $\tilde{u}^*$ represents the latest available solution. In other words, after (120) is solved for any element, the DOFs are immediately updated, and used to compute the residuals on all the rest of the elements. In all of the above BJ and GS approaches, an exact LU decomposition procedure is usually used to solve the linear system in each element, and the lower and upper matrices of **D** are pre-computed and stored. Another variation for all the above approaches is to freeze the diagonal block matrix **D** for several time steps in order to further boost computational efficiency. This variant was used and named quasi-non-linear element Jacobi method in [141].

The first variation obviously takes much more storage than the second one because of the need to store both **L** and **U**. However, it can allow a much larger time step because the full linearized system is solved at each time step.

4.2.2. Preconditioned GMRES approach and matrix-free implementation

Bassi and Rebay successfully developed a preconditioned GMRES (PGMRES) approach [137] for the DG method [75] to solver the compressible Navier–Stokes equations. A matrix-free Krylov approach was developed by Rasetarinera and Hussaini [138] with an efficient LU-SGS preconditioner. Very good convergence properties were demonstrated for high-order DG schemes with p up to 3. Any successful GMRES iterative approach needs a good pre-conditioner **P**. An equivalent preconditioned form of (115) is

$$\mathbf{P}^{-1}\mathbf{A}\Delta\tilde{u} = -\mathbf{P}^{-1}R(\tilde{u}^n). \tag{121}$$

To achieve fast convergence, **P** should be "close" to the implicit operator, but much easier to invert. A good compromise between efficiency and storage requirement is to use the main block diagonal (element) matrix **D**. Other preconditioners include the SGS approach, and the multilevel approach.

If the GMRES approach is implemented in the conventional fashion, the full implicit matrix **A** including the lower and upper matrixes needs to be stored. In large 3D computations, the storage requirement may become prohibitive. To remedy this deficiency, various matrix-free implementations of GMRES have been developed [138]. This is possible because all the operations involving matrix **A** in the GMRES algorithm is associated with the computation of matrix-vector products. As these products can be approximated using FDs, the algorithm can be implemented without forming the Jacobian matrix explicitly

$$\frac{\partial R(\tilde{u})}{\partial\tilde{u}}\tilde{u} \approx \frac{R(\tilde{u}+\varepsilon\tilde{u}) - R(\tilde{u})}{\varepsilon}, \tag{122}$$

where ε is a small parameter scaled appropriately with the norm of $\tilde{u}$ and the precision of the computer. In this implementation, it is not necessary to store **A**, thus yielding a considerable saving in storage compared to the standard implementation. However, the pre-conditioning matrix **P** must be formed and stored.

4.2.3. *Non-linear LU-SGS approach*

The original LU-SGS approach was developed by Yoon and Jameson [142] to solve compressible flow on structured grids, and demonstrated high solution efficiency with low storage requirements. Later, it was extended and applied to hybrid structured and unstructured grids [143]. Unstructured-grid-based LU-SGS schemes have demonstrated performance similar to that on structured grids [139]. In the LU-SGS scheme, a special first-order approximation is employed in linearizing the left-hand side resulting in the reduction of the block diagonal matrices to diagonal matrices. As a result, LU-SGS does not require any extra memory compared to explicit methods and is free from any matrix inversion. All of the off-diagonal matrices still contribute to the implicit operator through one forward and one backward sweep of a Gauss–Seidel iteration, thus significantly improving efficiency over an explicit scheme. The special first-order approximation used in deriving LU-SGS does degrade convergence rate, especially after several orders of convergence [62]. To further improve the convergence rate, Chen and Wang [62] and Jameson and Caughey [144] developed a block (preconditioned) non-linear LU-SGS (BLU-SGS) approach. Chen and Wang applied the approach to solve turbulent flows on arbitrary grids and demonstrated much better convergence rate than the original LU-SGS approach for a wide variety of flow problems. Jameson and Caughey used the approach as a multigrid smoother, and achieved convergence to truncation error level in just a few multigrid cycles. The most significant advantage of the BLU-SGS algorithm is the low storage requirement (only the diagonal block matrix is stored), and faster convergence to steady state than the fully linearized version, which requires the storage of the full matrix. The faster convergence is because the non-linear equation (114) is solved at each time step, rather than the linearized version. Therefore for time-accurate computations, the non-linear BLU-SGS approach is much more efficient than the standard approach of using multiple Newton sweeps for the non-linear system to converge the unsteady residual. In the following, we present a very compact form of the non-linear BLU-SGS approach [145]. Let $\tilde{\mathbf{D}} = (\mathbf{D} - M/\Delta t)$. It is then obvious

$$\frac{\partial R}{\partial \tilde{u}} = \tilde{\mathbf{D}} + \mathbf{L} + \mathbf{U}. \tag{123}$$

The forward sweep of the GS iteration for (115) is

$$\begin{aligned}\mathbf{D}\Delta\tilde{u}^{(k+1)} &= -R(\tilde{u}^n) - \mathbf{U}\Delta\tilde{u}^{(k+1)} - \mathbf{L}\Delta\tilde{u}^{(k)} \\ &= -R(\tilde{u}^n) - (\mathbf{U} + \mathbf{L})\Delta\tilde{u}^{(*)},\end{aligned} \tag{124}$$

where $\Delta\tilde{u}^{(*)}$ denotes the latest available solution updates, and $\tilde{u}^{(*)} = \tilde{u}^n + \Delta\tilde{u}^{(*)}$. Linearizing $R(\tilde{u}^{(*)})$, we obtain

$$R(\tilde{u}^{(*)}) \approx R(\tilde{u}^n) + (\tilde{\mathbf{D}} + \mathbf{L} + \mathbf{U})\Delta\tilde{u}^{(*)}. \tag{125}$$

So

$$R(\tilde{u}^n) + (\mathbf{L} + \mathbf{U})\Delta\tilde{u}^{(*)} = R(\tilde{u}^{(*)}) - \tilde{\mathbf{D}}\Delta\tilde{u}^{(*)}. \tag{126}$$

Substituting (126) into (124) gives

$$\left(\frac{M}{\Delta t} + \tilde{\mathbf{D}}\right)\Delta\tilde{u}^{(k+1)} = -R(\tilde{u}^{*}) + \tilde{\mathbf{D}}\Delta\tilde{u}^{(k)}, \tag{127}$$

which can be further simplified to

$$\mathbf{D}(\Delta\tilde{u}^{(k+1)} - \Delta\tilde{u}^{(k)}) = -R(\tilde{u}^{(*)}) + \frac{M}{\Delta t}\Delta\tilde{u}^{(*)}. \tag{128}$$

Note that the right-hand side is nothing but the unsteady residual evaluated with the latest available solutions. Multiple SGS sweeps can be employed to solve (128) iteratively. This form is also very easy to implement because the residual operator can be treated as a black box. The ease of implementation is particularly important for high-order methods because of the complexity of the spatial operators. Only the diagonal block is stored since M is usually an identity matrix or similar for a particular type of elements. When this SGS iteration converges, we are actually solving the non-linear equation (114) rather than its linearized version (115). Because of the removal of any errors due to linearization, the convergence rate of the non-linear version should be faster. Note that the non-linear equation (114) is solved without multiple Newton sweeps. This is very significant for unsteady flow problems and the present approach is expected to be much more efficient.

4.2.4. *Line-implicit approach*

The line-implicit approach was very popular for structured grid-based flow solvers as coordinate lines are readily available. In fact, approximate factorization and alternating direction implicit (ADI) algorithms widely used in structured grid methods are equivalent to applying a “line” implicit scheme in each of the coordinate directions. The line-implicit solver is very attractive since the resultant block tri-diagonal system can be solved efficiently. Much faster convergence rate is obtained than an element implicit scheme since the flow solutions on the line are solved in a coupled manner. The use of the line-implicit approach is not as widespread in unstructured grid-based solvers because there are no obvious coordinate lines in unstructured grids. Special techniques need to be developed to construct lines based on the computational grid and/or flow solution. There are many possibilities on how the lines are constructed. Two approaches are briefly described here.

In [140], Mavriplis developed an algorithm to build lines in an anisotropic mesh using a weighted graph technique. The rationale is to provide strong coupling along directions with the strongest grid clustering used to resolve viscous boundary layers or shear layers. In the weighted graph approach, each edge of the mesh is assigned a weight which represents the degree of coupling in the discretization. For simplicity, the edge weights are taken as the inverse of the edge length. The ratio of maximum to average adjacent edge weight is precomputed for every mesh vertex. The vertices are then sorted according to this ratio. The first vertex in this ordered list is then picked as the starting point for a line. The line is built by adding to the original vertex the neighboring vertex which is most strongly connected to the current vertex, provided this vertex does not already belong to a line, and provided the ratio of maximum to minimum edge weights for the current vertex is greater than a specified threshold. The line terminates when no additional vertex can be found. New lines are constructed from the rest of the vertices using the same approach. Using this line-implicit solver as a smoother for a multigrid solver, Mavriplis was able to demonstrate aspect ratio independent convergence rate for a viscous flow computation [42].

More recently, a line-implicit solver was developed by Fidkowski et al. [146] to serve as the smoother for a p-multigrid approach solving the Navier–Stokes equations. A new algorithm for constructing lines was developed, and it is based on both the computational grid and the underlying physics in some sense. The main idea is to strongly couple lines of elements connected along directions of strong convection or grid anisotropy. The coupling between the elements is computed from a first-order (element-wise constant) discretization of a scalar convection diffusion equation in the following form:

$$Y(s) = \nabla \cdot (\rho\vec{v}s) - \nabla \cdot (\mu \nabla s), \tag{129}$$

where momentum $\rho\vec{v}$ and viscosity μ are taken from the latest available solution. Then the coupling between two elements j and k that share a face is given by the "residual Jacobian"

$$C_{j,k} = \max\left(\left|\frac{\partial Y_j}{\partial s_k}\right|, \left|\frac{\partial Y_k}{\partial s_j}\right|\right). \tag{130}$$

After that, a two stage line creation and connection algorithm was developed to constructed line sets. A block tri-diagonal linear system is built and solved for each line to provide implicitness coupling all the elements along the line. This line solver was

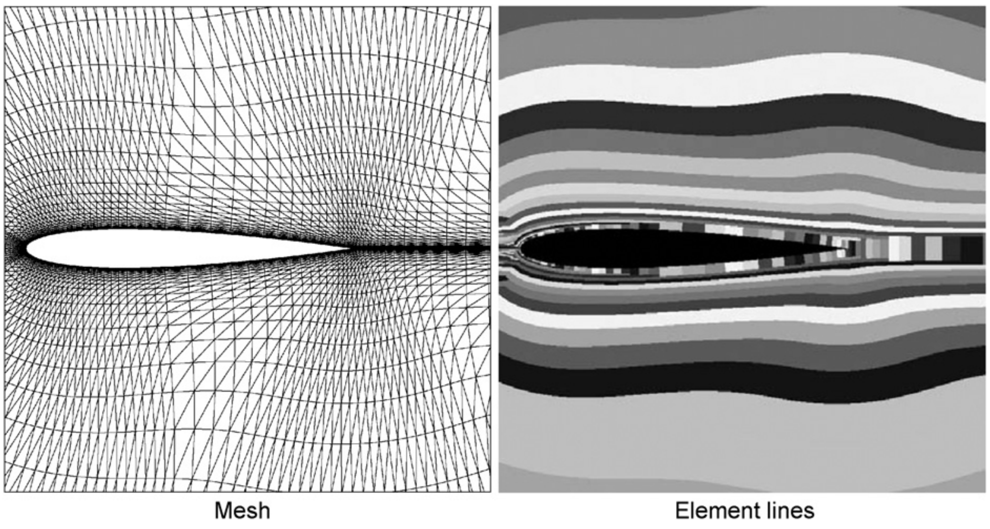


Fig. 31. Lines formed around the NACA0012 airfoil at $M = 0.5$, $Re = 5000$, $\alpha = 0°$ (courtesy of Fidkowski, Darmofal et al. [146]).

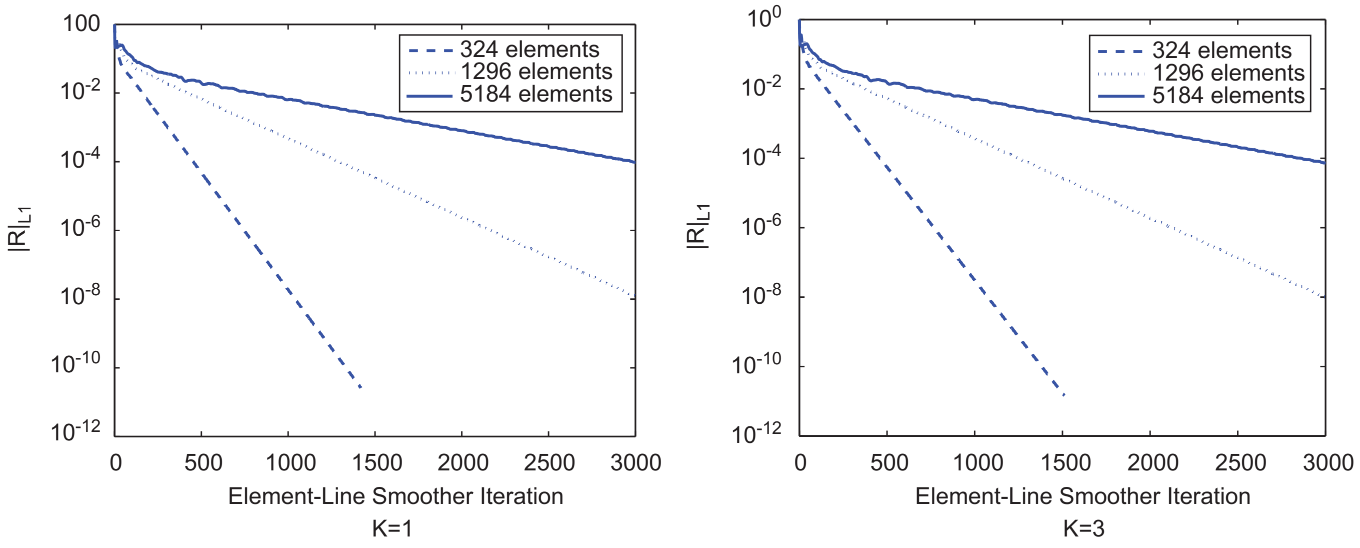


Fig. 32. Element line smoother convergence histories vs. grid size for inviscid duct flow at $M = 0.5$ (courtesy of Fidkowski et al. [146]).

shown to be much more effective as a *p*-multigrid smoother than an element Jacobi approach, and *p*-independent multigrid convergence rates were demonstrated. Shown in Fig. 31 is a viscous computational grid around the NACA0012 airfoil and the element lines generated using this approach [146]. Note that the lines are perpendicular to the wall boundary in the viscous boundary layer, but aligned with streamlines elsewhere. With this line-implicit smoother, Fidkowski et al. were able to achieve *p*-independent convergence rate for both inviscid and viscous flows. One typical convergence history plot for inviscid flow through a duct with various polynomial orders is shown in Fig. 32. The convergence rate is obviously independent of *p*, though still dependent on *h*.

4.3. Geometric and p-multigrid methods

The standard multigrid algorithm (geometric multigrid, or *h*-multigrid) has been used very effectively in CFD to accelerate the rate of convergence to steady state. Jameson popularized multigrid in CFD with his original combination of Runge–Kutta time stepping and multigrid approaches [130], and since then there have been numerous advances in the field, e.g., [38,140,144,147–151]. In an *h*-multigrid implementation, multilevels of coarse grids are generated either from the finer grids or independently, with each coarser grid roughly doubling the mesh size of the next finer mesh in all directions. Several factors contribute to the effectiveness of multigrid approach. First, all smoothers are more efficient in damping high frequency errors than low frequency ones. Errors on a fine mesh are represented on a coarser mesh at higher frequency, which can be damped more effectively. Second, larger time steps can be used on the coarser mesh (in the case of explicit time stepping schemes) and errors can be driven out of the computational domain faster.

The most critical element of a successful multigrid approach is the development of an effective "smoother" to remove various types of "stiffness" in the spatial operator. For example, the stiffness due to low flow speed and grid anisotropy was known to degrade the performance of multigrid solvers. Many novel numerical techniques were developed to address them, including low speed preconditioning [152–154], line-implicit solvers [140] and semi-coarsening [151]. The most impressive multigrid results are due to Jameson and Caughey [144], who demonstrated convergence to truncation error level using just a few multigrid cycles with a preconditioned non-linear LU-SGS smoother.

In addition, *p*-multigrid, or multiorder, solution strategies can be used in an analogous manner to accelerate convergence of high-order methods to steady state. Lower-order spatial operators in a *p*-multigrid approach serve as the "coarse grid" operators in the *h*-multigrid counterpart. Lower-order operators have similar advantages: they have more numerical damping and allow larger time steps than high-order operators. The *p*-multigrid algorithm was introduced by Ronquist and Patera

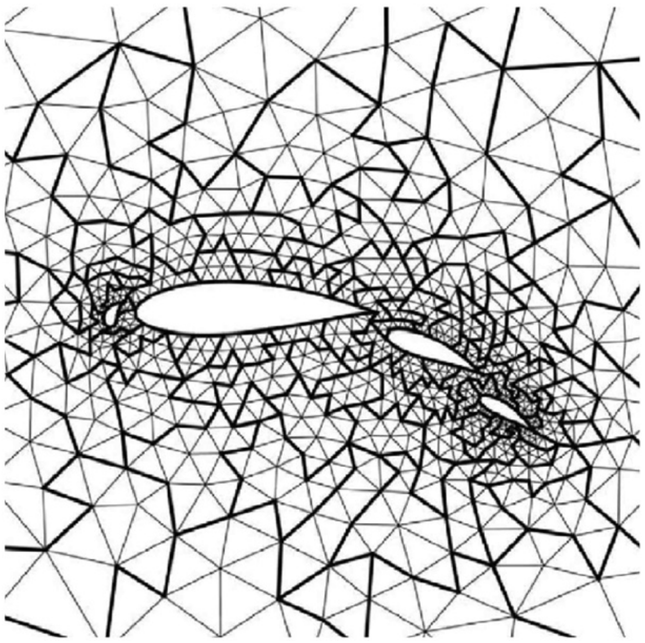


Fig. 33. A typical two level *h*-multigrid mesh configuration (courtesy of Nastase and Mavriplis [141]).

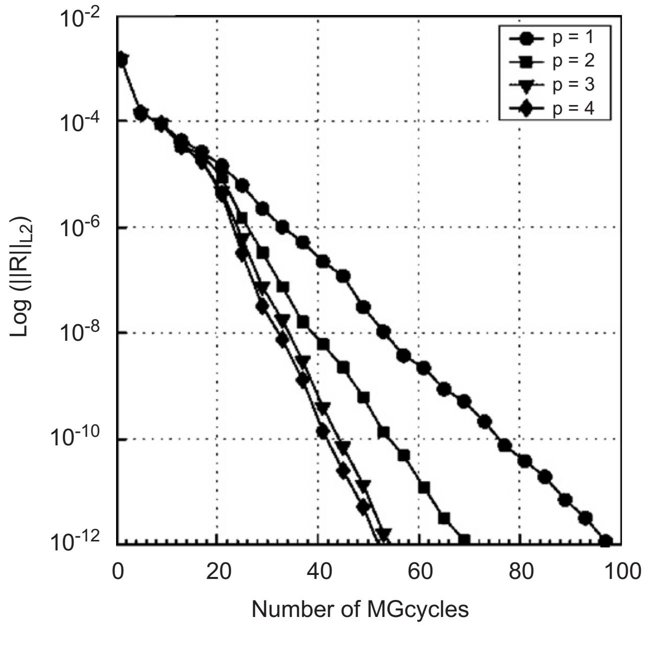


Fig. 34. The full hp-multigrid convergence vs. the number of multigrid (MG) cycles, on a mesh size of $N = 5916$ elements and polynomial degrees (p), for the four-element airfoil problem (courtesy of Nastase and Mavriplis [41]).

[155], and later analyzed by Maday and Munoz [156] for a 1D, Galerkin spectral element discretization of the Laplace equation. More recently, Bassi and Rebay presented a *p*-multigrid solution method for the DG discretization of the Euler equations [157]. Helenbrook et al. [158] examined the perfor-

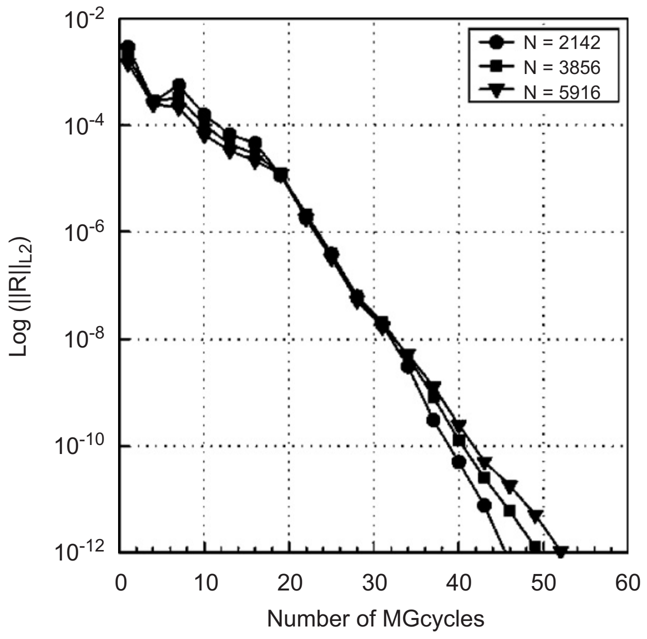


Fig. 35. The full hp-multigrid convergence vs. the number of multigrid (MG) cycles, on various fine grid problem sizes and polynomial degree 4, for the four-element airfoil problem (courtesy of Nastase and Mavriplis [41]).

mance of *p*-multigrid for Laplace equation and the convection equation in two dimensions. Fidkowski et al. [146] developed a *p*-multigrid approach using an effective line-implicit smoother for their high-order DG Navier–Stokes solver, and demonstrated *p*-independent convergence rate for *p* up to 3 with fourth-order accuracy. Luo et al. demonstrated a low storage *p*-multigrid approach for a 3D DG Euler solver [159], in which an explicit Runge–Kutta scheme is used for the highest order discretization, while implicit SGS smoothers are employed for the lower-order operators. They demonstrated an order of magnitude speed up for *p* up to 2. Nastase and Mavriplis [141] developed an *hp*-multigrid approach for their high-order inviscid DG solver, and demonstrated *h*-independent and *p*-independent convergence rates. The coupling of *p*- and *h*-multigrid procedures, through the use of agglomerated coarse levels for unstructured meshes, increases the overall solution efficiency compared to a *p*-alone multigrid procedure, and the benefits of the *hp*-multigrid approach are expected to increase for finer meshes.

An example computation of inviscid flow around a four element airfoil by Nastase and Mavriplis [141] is presented here. One typical computational grid including the agglomerated coarser level is displayed in Fig. 33. A full multigrid strategy with a non-linear FAS *hp*-multigrid scheme was used in the simulation. The smoother is a linear GS approach, in which the diagonal and off-diagonal matrices are stored. In Fig. 34, the convergence rate is shown for a fixed mesh size of $N = 3856$, for various p discretizations. The convergence rate increases slightly with higher-order accurate discretizations. Fig. 35 shows the convergence rate for the $p = 4$ discretization on the various grids for the four-element airfoil configuration. The convergence rate is nearly h and p independent.

5. Conclusions

Several high-order numerical methods for the Euler and Navier–Stokes equations on unstructured grids are reviewed in this article. Both spatial and temporal discretizations are discussed. For problems with high-accuracy requirement such as wave propagation problems and vortex-dominated flows, high-order methods are (asymptotically) more efficient than low-order ones based on a simple generic analysis.

Although both structured and unstructured grid-based high-order methods will co-exist and excel for different type of flow problems, high-order unstructured grid methods offer several distinctive advantages:

- geometric flexibility;
- *hp* grid adaptivity;
- easier to maintain load balance on parallel computers;
- less stringent on grid smoothness requirement.

We can perhaps distinguish the methods reviewed here according to several different criteria to see the differences and similarities.

Number of degrees per element: The k-exact, ENO/WENO FV methods employ one DOF per element or cell, while all the rest employ multiple DOFs in each element. As a result, neighboring data from a local stencil are used to reconstruct a high-order polynomial on the element in the k-exact and ENO/WENO methods.

Continuous or discontinuous: The continuous FE and RD methods employ continuous polynomials, while the rest use discontinuous polynomials. Those methods with discontinuous polynomials employ Riemann fluxes, in one form or the other, to achieve stability.

Discretization approach: The continuous and discontinuous FE methods use a weighted residual, usually a Galerkin or Petrov–Galerkin, approach. The k-exact, ENO/WENO, and SV methods apply a FV approach (also a Petrov–Galerkin approach), while the SD method use a FD approach. The RD method utilizes ideas from FV and FE methods.

Some common challenges for all high-order methods include the following:

- Discontinuity capturing: Any shock-capturing methods will degrade to first-order accuracy locally near a discontinuity because the error in the location of the shock is proportional to the mesh size. Methods which offer natural subcell resolution (such as the SV method) can make the error smaller, but cannot change the order. This argument suggests *h*-refinement near shock waves, coupling with a piece-wise constant reconstruction, which is the robust, first-order Godunov method, with *p*-refinement elsewhere. How a locally first-order scheme affects the solution elsewhere needs to be investigated, especially for unsteady flow problems.
- Efficient time integrator for steady problems: High Reynolds number viscous flow problems have disparate length scales in the computational domain, and the computational grids are highly clustered in the boundary layer. Therefore, explicit methods are too slow for steady problems due to the CFL condition. It appears *hp*-multigrid methods offer the most promising solution. However, the development of effective smoothers for high-order operators may be hindered by computer core memory. The memory requirement for the element Jacobi matrices in 3D for degree higher than 3 polynomials may be prohibitive. The main challenge will be to develop effective and low storage smoothers for high-order operators.
- High-order grid generation and flow visualization: Almost all researchers in high-order methods pointed out the importance of high-order boundary representation. Many demonstrated the inadequacy of piece-wise linear facet representation used in low-order methods. As most grid generation packages were developed for lower-order methods, capabilities should be added to generate coarser grids with high-order (at least quadratic) boundaries. The generation of highly clustered viscous meshes near high-order walls is another challenge, as more elements near the boundary layer must be better than linear to avoid grid lines crossing into each other. The same can be said about visualization.

Almost all graphics engines assume linear data. It may be more efficient to consider high-order data in the visualization layer.

Acknowledgments

I would like to thank my collaborators in the development of high-order methods, Drs. Yen Liu, Yuzhi Sun, Georg May and Antony Jameson. In addition, I gratefully acknowledge support by Michigan State University, Iowa State University, the National Science Foundation, the Air Force Office of Scientific Research, the Department of Energy and the Defense Advanced Research Project Agency. Thanks are also due to R. Abgrall, F. Bassi, B. Cockburn, D. Darmofal, D. Mavriplis, C.-W. Shu, K. Van den Abeele, V. Venkatakrishnan for the permission to use figures from their publications.

References

[1] Vos JB, Rizzi A, Darracq D, Hirschel EH. Navier–Stokes solvers in European aircraft design. Prog Aerospace Sci 2002;38:601–97.

[2] Vassberg JC. Expectations for computational fluid dynamics. Int J Comput Fluid Dyn 2005;19(8):549–58.

[3] Aftosmis M, Gaitonde D, Tavares TS. Behavior of linear reconstruction techniques on unstructured meshes. AIAA J 1995;33(11):2038–49.

[4] Anderson WK. A grid generation and flow solution method for the Euler equations on unstructured grids. J Comput Phys 1994;110:23–38.

[5] Barth TJ, Jespersen DC. The design and application of upwind schemes on unstructured meshes. AIAA Paper No. 89–0366, 1989.

[6] Frink NT. Recent progress toward a three dimensional unstructured Navier–Stokes flow solver. AIAA Paper No. 94–0061, 1994.

[7] LeVeque RJ. Finite volume methods for hyperbolic problems. Cambridge: Cambridge University Press; 2002.

[8] Knight DD. Elements of numerical methods for compressible flow. Cambridge: Cambridge University Press; 2006.

[9] Boris JP, Book DL. Flux corrected transport,1 SHASTA, a fluid transport algorithm that works. J Comput Phys A 1973;11:38–69.

[10] Harten A. High resolution schemes for hyperbolic conservation laws. J Comput Phys 1983;49:357–93.

[11] Steger JL, Warming RF. Flux vector splitting of the inviscid gas dynamics equations with application to finite difference methods. J Comput Phys 1981;40:263.

[12] Biswas R, Devine KD, Flaherty JE. Parallel, adaptive finite element methods for conservation laws. Appl Numer Math 1994;14:255–83.

[13] Hassan O, Morgan K, Peraire J. An implicit finite element method for high speed flows. AIAA Paper No. 90–0402, January 1990.

[14] Hughes TJR. Recent progress in the development and understanding of SUPG methods with special reference to the compressible Euler and Navier–Stokes equations. Int J Numer Methods Fluids 1987;7:1261–75.

[15] Hughes TJR, Franca LP, Hulbert GM. A new finite element formulation for computational fluid dynamics: VIII. The Galerkin least squares method for advective–diffusive equations. Comput Methods Appl Mech Eng 1989;73:173–89.

[16] Hughes TJR, Mallet M. A new finite element formulation for CFD: IV. A discontinuity-capturing operator for multidimensional advective–diffusive systems. Comput Methods Appl Mech Eng 1986;58(3):329–56.

[17] Lohner R, Morgan K, Zienkiewicz OC. An adaptive finite element procedure for compressible high speed flows. Comput Methods Appl Mech Eng 1985;51:441–65.

[18] Visbal MR, Gaitonde DV. On the use of higher-order finite-difference schemes on curvilinear and deforming meshes. J Comput Phys 2002;181(1):155–85.

[19] Fujii K. Progress and future prospects of CFD in aerospace–wind tunnel and beyond. Prog Aerospace Sci 2005;41:455–70.

[20] Ekaterinaris JA. High-order accurate, lownumerical diffusion methods for aerodynamics. Prog in Aerospace Sci 2005;41:192–300.

[21] Cockburn B, Shu C-W. TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws II: general framework. Math Comput 1989; 52:411–35.

[22] Wang ZJ. Spectral (finite) volume method for conservation laws on unstructured grids: basic formulation. J Comput Phys 2002;178:210–51.

[23] Wang ZJ, Liu Y. Spectral (finite) volume method for conservation laws on unstructured grids II: extension to two-dimensional scalar equation. J Comput Phys 2002; 179(2):665–97.

[24] Gottlieb S, Orszag A. Numerical analysis of spectral method: theory and applications. Philadelphia, PA: SIAM; 1977.

[25] Canuto C, Hussaini MY, Quarteroni A, Zang TA. Spectral methods in fluid dynamics. New York: Springer; 1987.

[26] Patera AT. A spectral element method for fluid dynamics: laminar flow in a channel expansion. J Comput Phys 1984; 54:468–88.

[27] Kopriva DA. A staggered-grid multidomain spectral method for the compressible Navier–Stokes equations. J Comput Phys 1998;143:125–48.

[28] Karniadakis GE, Sherwin SJ. Spectral-hp element methods. Oxford: Oxford University Press; 1999.

[29] Carpenter MH, Gottlieb D. Spectral methods on arbitrary grids. J Comput Phys 1996;129(1):74–86.

[30] Hesthaven JS, Teng CH. Stable spectral methods on tetrahedral elements. SIAM J Sci Comput 2000;21:2352–80.

[31] Van Leer B, Nomura S. Discontinuous Galerkin for diffusion. AIAA Paper No. 2005–5108, 2005.

[32] Barth TJ, Frederickson PO. High-order solution of the Euler equations on unstructured grids using quadratic reconstruction. AIAA Paper No. 90–0013, 1990.

[33] Godunov SK. A finite-difference method for the numerical computation of discontinuous solutions of the equations of fluid dynamics. Mat Sb 1959;47:271.

[34] Van Leer B. Towards the ultimate conservative difference scheme II. Monotonicity and conservation combined in a second-order scheme. J Comput Phys 1974;14:361.

[35] Van Leer B. Towards the ultimate conservative difference scheme V. A second order sequel to Godunov's method. J Comput Phys 1979;32:101–36.
[36] Colella P, Woodward P. The piecewise parabolic method for gas-dynamical simulations. J Comput Phys 1984;54.
[37] Harten A, Engquist B, Osher S, Chakravarthy S. Uniformly high-order essentially non-oscillatory schemes III. J Comput Phys 1987;71:231.
[38] Braaten ME, Connell SD. Three dimensional unstructured adaptive multigrid scheme for the Navier–Stokes equations. AIAA J 1996;34(2):281–90.
[39] Jameson A, Baker TJ, Weatherill NP. Calculation of inviscid transonic flow over a complete aircraft. AIAA Paper 86–0103, 1986.
[40] Kallinderis Y, Khawaja A, McMorris H. Hybrid prismatic/tetrahedral grid generation for complex geometries. AIAA J 1996;34:291–8.
[41] Lohner R, Parikh P. Generation of three-dimensional unstructured grids by the advancing front method. Int J Numer Methods Fluids 1988;8:1135–49.
[42] Mavriplis DJ. Unstructured grid techniques. Annu Rev Fluid Mech 1997;29:473–514.
[43] Pirzadeh S. Three-dimensional unstructured viscous grids by the advancing-layers method. AIAA J 1996;34(1):43–9.
[44] Venkatakrishnan V. A perspective on unstructured grid flow solvers. AIAA J 1996;34(3):533–47.
[45] Wang ZJ, Chen RF. Anisotropic cartesian grid method for viscous turbulent flow. AIAA Paper No. 2000–0395, 2000.
[46] Abgrall R. On essentially non-oscillatory schemes on unstructured meshes: analysis and implementation. J Comput Phys 1994;114:45–58.
[47] Durlofsky LJ, Enquist B, Osher S. Triangle based adaptive stencils for the solution of hyperbolic conservation laws. J Comput Phys 1992;98:64.
[48] Ollivier-Gooch CF. Quasi-ENO schemes for unstructured meshes based on unlimited data-dependent least-squares reconstruction. J Comput Phys 1997;133:6–17.
[49] Sonar T. On the construction of essentially non-oscillatory finite volume approximations to hyperbolic conservation laws on general triangulations: polynomial recovery accuracy and stencil selection. Comput Methods Appl Mech Eng 1997;140:157.
[50] Liu XD, Osher S, Chan T. Weighted essentially non-oscillatory schemes. J Comput Phys 1994;115:200–12.
[51] Jiang G, Shu C-W. Efficient implementation of weighted ENO schemes. J Comput Phys 1996;126:202.
[52] Friedrich O. Weighted essentially non-oscillatory schemes for the interpolation of mean values on unstructured grids. J Comput Phys 1998;144:194–212.
[53] Hu C, Shu C-W. Weighted essentially non-oscillatory schemes on triangular meshes. J Comput Phys 1999;150:97–127.
[54] Shu C-W. Essentially non-oscillatory and weighted essentially non-oscillatory schemes for hyperbolic conservation laws. In: Cockburn B, Johnson C, Shu C-W, Tadmor E, Quarteroni A, editors. Advanced Numerical Approximation of Nonlinear Hyperbolic Equations, Lecture notes in mathematics, vol. 1697. Berlin: Springer; 1998. p. 325–432.
[55] Rusanov VV. Calculation of interaction of non-steady shock waves with obstacles. J Comput Math Phys USSR 1961;1:261–79.
[56] Roe PL. Approximate Riemann solvers, parameter vectors, and difference schemes. J Comput Phys 1981;43:357–72.
[57] Liou M-S. Mass flux schemes and connection to shock instability. J Comput Phys 2000;160:623–48.
[58] Jameson A. Analysis and design of numerical schemes for gas dynamics. I. Artificial diffusion, upwind biasing, limiters and their effect on accuracy and multigrid convergence. Int J Comput Fluid Dyn 1994;4:171–218.
[59] Delanaye M, Liu Y. Quadratic reconstruction finite volume schemes on 3D arbitrary unstructured polyhedral grids. AIAA Paper No. 99–3259-CP, 1999.
[60] Woodward P, Colella P. The numerical simulation of two-dimensional fluid flow with strong shocks. J Comput Phys 1984;54:115.
[61] Zienkiewicz OC, Taylor RL. Finite element method. 5th ed. London: Butterworth-Heinemann; 2000.
[62] Chen RF, Wang ZJ. Fast block lower-upper symmetric Gauss Seidel scheme for arbitrary grids. AIAA J 2000; 38(12):2238–45.
[63] Heinrich JC, Papper DW. Intermediate finite element method. London, Philadelphia, PA: Taylor & Francis; 1999.
[64] Solin P, Segeth K, Dolezel I. Higher-order finite element methods, London, Boca Raton: Chapman & Hall, CRC; 2004.
[65] Jiang BN. The least-squares finite element method: theory and applications in computational fluid dynamics and electromagnetics. Berlin: Springer; 2006.
[66] Bochev PB, Gunzburger MD. Least-squares: finite element methods. Berlin: Springer; 2007.
[67] Gerritsma M, De Maerschalck B. The least squares spectral element method CFD–higher order discretization methods. In: Deconinck H, Ricchiuto M, editors. von Karman Institute for Fluid Dynamics; 2006.
[68] Bonhaus DL. A high-order accurate finite element method for viscous compressible flow, AIAA Paper No. 99–0780, 1999.
[69] Venkatakrishnan V, Allmaras SR, Kamenetski DS, Johnson FT. Higher order schemes for the compressible Navier–Stokes equations. AIAA Paper No. 2003–3987, 2003.
[70] Reed WH, Hill TR. Triangular mesh methods for the neutron transport equation. Technical Report LA-UR-73–479, Los Alamos Scientific Laboratory; 1973.
[71] Cockburn B, Lin S-Y, Shu C-W. TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws III: one-dimensional systems. J Comput Phys 1989;84:90–113.
[72] Cockburn B, Hou S, Shu C-W. TVB Runge–Kutta local projection discontinuous Galerkin finite element method for conservation laws IV: the multidimensional case. Math Comput 1990;54:545–81.
[73] Cockburn B, Shu C-W. The Runge–Kutta discontinuous Galerkin finite element method for conservation laws V: multidimensional systems. J Comput Phys 1998;141: 199–224.
[74] Bassi F, Rebay S. High-order accurate discontinuous finite element solution of the 2D Euler equations. J Comput Phys 1997;138:251–85.
[75] Bassi F, Rebay S. A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations. J Comput Phys 1997;131(1):267–79.
[76] Bassi F, Rebay S. Numerical evaluation of two discontinuous Galerkin methods for the compressible Navier–Stokes

equations. Int J Numer Methods Fluids 2002;40(1): 197–207.

[77] Atkin HL, Shu C-W. Quadrature-free implementation of discontinuous Galerkin method for hyperbolic equations. AIAA J 1998;36:775–82.

[78] Baumann CE, Oden TJ. A discontinuous hp finite element method for the Euler and Navier–Stokes equations. Int J Numer Methods Fluids 1999;31(1):79–95.

[79] Doleji V. On the discontinuous Galerkin method for the numerical solution of the Navier–Stokes equations. Int J Numer Methods Fluids 2004;45(10):1083–106.

[80] Huynh HT. An upwind moment scheme for conservation laws. in: Groth, Zingg, editors. Proceedings of the third international conference of CFD, Toronto, Canada, 2004.

[81] Krivodonova L, Berger M. High-order accurate implementation of solid wall boundary conditions in curved geometries. J Comput Phys 2006;211(2):492–512.

[82] Lin SY, Chin YS. Discontinuous Galerkin finite element method for Euler and Navier–Stokes equations. AIAA J 1993;31(11):2016–26.

[83] Lomtev I, Karniadakis GE. A discontinuous Galerkin method for the Navier–Stokes equations. Int J Numer Methods Fluids 1999;29(5):587–603.

[84] Lowrie, RB, Roe PL, van Leer B. A space-time discontinuous Galerkin method for the time accurate numerical solution of hyperbolic conservation laws. AIAA Paper No. 95–1658, 1995.

[85] Remaki M, Habashi WG. A discontinuous Galerkin method/HLLC solver for the Euler equations. Int J Numer Methods Fluids 2003;43(12):1391–405.

[86] Van der Vegt JJW, Van der Ven H. Space time discontinuous Galerkin finite element method with dynamic grid motion for inviscid compressible flows: I. General formulation. J Comput Phys 2002;182(2):546–85.

[87] Cockburn B, Karniadakis GE, Shu CW. The development of discontinuous Galerkin methods. In: Cockburn B, Karniadakis GE, Shu CW, editors. Discontinuous Galerkin methods. Berlin: Springer; 2000.

[88] Arnold, DN, Brezzi, F, Cockburn, B, Marin LD. Unified analysis of discontinuous Galerkin methods for elliptic problems. SIAM J Numer Anal 2001;39:1749–79.

[89] Oden JT, Babuska I, Baumann CE. A discontinuous hp finite element method for diffusion problems. J Comput Phys 1998;146(2):491–519.

[90] Cockburn B, Shu C-W. The local discontinuous Galerkin method for time-dependent convection diffusion system. SIAM J Numer Anal 1998;35:2440–63.

[91] Harris R, Wang ZJ, Liu Y. Efficient implementation of high-order spectral volume method for multidimensional conservation laws on unstructured grids. AIAA Paper No. 2007–912, 2007.

[92] Shu C-W. TVB uniformly high-order schemes for conservation laws. Math Comput 1987;49:105–21.

[93] Bassi F, Crivellini A, Rebay S, Savini M. Discontinuous Galerkin solutions of the Reynolds-averaged Navier–Stokes and K2o turbulence model equations. Comput Fluids 2005;34(4–5):507–40.

[94] Nguyen NC, Persson P-O, Peraire J. RANS solutions using high-order discontinuous Galerkin methods. AIAA Paper No. 2007–0914, 2007.

[95] Roe PL. Optimum upwind advection on a triangular mesh. ICASE Report 90–75, 1990.

[96] Deconinck H, Struijs R, Bourgeois G, Roe PL. Compact advection schemes on unstructured meshes. VKI Lecture Series 1993-04. Comput Fluid Dyn 1993.

[97] Paillere H, Deconinck H, Struijs R, Roe PL, Mesaros LM, Muller JD. Computations of compressible flows using fluctuation splitting on triangular meshes. AIAA Paper No. 93–3301-CP, 1993.

[98] Paillere H, Deconinck H, van der Weide E. Upwind residual distribution methods for compressible flow: an alternative to finite volume and finite element methods. Presented at 28th CFD VKI lecture series; 1997.

[99] Barth TJ, Deconinck H. High-order methods for computational physics. Berlin: Springer; 1999.

[100] Nishikawa H. Higher order discretization of diffusion terms in residual distribution methods. In: Deconinck H, Ricchiuto M, editors. CFD—higher order discretization methods. von Karman Institute for Fluid Dynamics; 2006.

[101] Caraeni D, Fuchs L. Compact third-order multidimensional upwind discretization for steady and unsteady flow simulations. Comput Fluids 2005;34:419–41.

[102] Corre C, Hanss G, Lerat A. A residual-based compact scheme for the unsteady compressible Navier–Stokes equations. Comput Fluids 2005;34(4–5):561–80.

[103] Abgrall R, Roe PL. High-order fluctuation splitting schemes on triangular mesh. J Sci Comput 2003;19:3–36.

[104] Ricchiuto M, Csik A, Deconinck H. Residual distribution for general time-dependent conservation laws. J Comput Phys 2005;209:249–89.

[105] Abgrall R. Residual distribution schemes: current status and future trends. Comput Fluids 2006;35:641–69.

[106] Wang ZJ, Liu Y. Spectral (finite) volume method for conservation laws on unstructured grids III: one-dimensional systems and partition optimization. J Sci Comput 2004;20:137–57.

[107] Wang ZJ, Zhang L, Liu Y. Spectral finite volume method for conservation laws on unstructured grids IV: extension to two-dimensional systems. J Comput Phys 2004;194(2):716–41.

[108] Liu Y, Vinokur M, Wang ZJ. Spectral (finite) volume method for conservation laws on unstructured grids V: extension to three-dimensional systems. J Comput Phys 2006;212:454–72.

[109] Sun Y, Wang ZJ, Liu Y. Spectral (finite) volume method for conservation laws on unstructured grids VI: extension to viscous flow. J Comput Phys 2006;215(1):41–58.

[110] Chen Q-Y. Partitions of a simplex leading to accurate spectral (finite) volume reconstruction. SIAM J Sci Comput 2006;27(4):1458–70.

[111] Chen QY. Partitions for spectral finite volume reconstruction in the tetrahedron. J Sci Comput 2006;29(3): 299–319.

[112] Sun Y, Wang ZJ. Evaluation of discontinuous Galerkin and spectral volume methods for scalar and system conservation laws on unstructured grid. Int J Numer Methods Fluids 2004;45(8):819–38.

[113] Zhang M, Shu CW. An analysis and a comparison between the discontinuous Galerkin method and the spectral finite volume methods. Comput Fluids 2005;34(4–5):581–92.

[114] Haga T, Ohnishi N, Sawada K, Masunaga A. Spectral volume computation of flowfield in aerospace application using earth simulator. AIAA Paper No. 2006–2823, 2006.

[115] Van den Abeele K, Broeckhoven T, Lacor C. Dispersion and dissipation properties of the 1D spectral volume

method and application to a p-multigrid algorithm. J Comput Phys 2007;224:616–36.
[116] Van den Abeele K, Lacor C. An accuracy and stability study of the 2D spectral volume method. J. Comput Phys, in press.
[117] Liu Y, Vinokur M, Wang ZJ. Discontinuous spectral difference method for conservation laws on unstructured grids. In: Proceedings of the 3rd international conference on computational fluid dynamics, Toronto, Canada, July 12–16, 2004.
[118] Liu Y, Vinokur M, Wang ZJ. Discontinuous spectral difference method for conservation laws on unstructured grids. J Comput Phys 2006;216:780–801.
[119] Wang ZJ, Liu Y. The spectral difference method for the 2D Euler equations on unstructured grids. AIAA Paper No. 2005–5112, 2005.
[120] Wang ZJ, Liu Y, May G, Jameson A. Spectral difference method for unstructured grids II: extension to the Euler equations. J Sci Comput 2007;32:45–71.
[121] May G, Jameson A. A spectral difference method for the Euler and Navier–Stokes equations. AIAA Paper No. 2006–304, 2006.
[122] Wang ZJ, Sun Y, Liang C, Liu Y. Extension of the SD method to viscous flow on unstructured grids. In: Proceedings of the 4th international conference on computational fluid dynamics, Gent, Belgium, July 2006.
[123] May G, Jameson A. High-order accurate methods for high-speed flow. AIAA Paper No. 2005–5251, 2005.
[124] Huang PG, Wang ZJ, Liu Y. An implicit space–time spectral difference method for discontinuity capturing using adaptive polynomials. AIAA-2005–5255, 2005.
[125] Kopriva DA, Kolias JH. A conservative staggered-grid Chebyshev multidomain method for compressible flows. J Comput Phys 1996;125:244.
[126] Liu Y, Vinokur M. Exact integration of polynomials and symmetric quadrature formulas over arbitrary polygonal grids. J Comput Phys 1998;140(1):122–1247.
[127] Hesthaven JS. From electrostatics to almost optimal nodal sets for polynomial interpolation in a simplex. SIAM J Numer Anal 1998;35(2):655–76.
[128] Sun Y, Wang ZJ, Liu Y. High-order multidomain spectral difference method for the Navier–Stokes equations on unstructured hexahedral grids. Commun Comput Phys 2007;2:310–33.
[129] Jameson A, Schmidt W, Turkel E. Numerical solution of the Euler equations by finite volume methods using Runge–Kutta time stepping schemes. AIAA Paper No. 81–1259; 1981.
[130] Jameson A. Solution of the Euler equations by a multigrid method. Appl Math Comput 1983;13:327–56.
[131] Shu C-W. Total-variation-diminishing time discretizations. SIAM J Sci Stat Comput 1988;9:1073–84.
[132] Shu C-W, Osher S. Efficient implementation of essentially non-oscillatory shock-capturing schemes. J Comput Phys 1988;77:439–71.
[133] Gottlieb S, Shu CW. Total variation diminishing Runge–Kutta schemes. Math Comput 1998;67:73–85.
[134] Spiteri RJ, Ruuth SJ. A new class of optimal high-order strong-stability preserving time discretization methods. SIAM J Numer Anal 2002;40:469–91.
[135] Gottlieb S. On high-order strong stability preserving Runge–Kutta and multi step time discretizations. J Sci Comput 2005;25(1/2):105–28.
[136] Venkatakrishnan V, Mavriplis DJ. Implicit solvers for unstructured meshes. J Comput Phys 1993;105(1):83–91.
[137] Saad Y, Schultz MH. GMRES: A genearlized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 1986;7:865.
[138] Rasetarinera P, Hussaini MY. An efficient implicit discontinuous Galerkin method. J Comput Phys 2001; 172(2):718–38.
[139] Sharov D, Nakahashi K. Low speed preconditioning and LUSGS scheme for 3D viscous flow computations on unstructured grids. AIAA Paper No. 98–0614, January 1998.
[140] Mavriplis DJ. Multigrid strategies for viscous flow solvers on anisotropic unstructured meshes. J Comput Phys 1998; 145:141–65.
[141] Nastase CR, Mavriplis DJ. High-order discontinuous Galerkin methods using an hp-multigrid approach. J Comput Phys 2006;213:330–57.
[142] Yoon S, Jameson A. Lower-upper symmetric-Gauss–Seidel method for the Euler and Navier–Stokes equations. AIAA Journal 1988;26:1025–6.
[143] Soetrisno M, Imlay ST, Roberts DW. A zonal implicit procedure for hybrid structured-unstructured grids. AIAA Paper No. 94–0617, 1994.
[144] Jameson A, Caughey DA. How many steps are required to solve the Euler equations of steady compressible flow: in search of a fast solution algorithm. In: 15th AIAA computational fluid dynamics conference, Anaheim, CA, June 2001.
[145] Sun Y, Wang ZJ, Liu Y, Chen CL. Efficient implicit LU-SGS algorithm for high-order spectral difference method on unstructured hexahedral grids. AIAA Paper No. 2007–0313, 2007.
[146] Fidkowski KJ, Oliver TA, Lu J, Darmofal DL. p-Multigrid solution of high-order discontinuous Galerkin discretizations of the compressible Navier–Stokes equations. J Comput Phys 2005;207:92–113.
[147] Allmaras S. Analysis of semi-implicit preconditioners for multigrid solution of the 2-D Navier Stokes equations. AIAA Paper 95–1651, AIAA 12th computational fluid dynamics conference, San Diego, CA, June 1995.
[148] Alonso JJ, Martinelli L, Jameson A. Multigrid unsteady Navier Stokes calculations with aeroelastic applications. AIAA Paper 95–0048, AIAA 33rd aerospace sciences meeting, Reno, NV, January 1995.
[149] Hemker PW, Koren B. Defect correction and non-linear multi-grid for the steady Euler equations. Lecture Notes, 1988-05 von Karman Institute for Fluid Dynamics, 1988.
[150] Mavriplis DJ, Jameson A. Multigrid solution of the Navier–Stokes equations on triangular meshes. AIAA J 1990;28:1415–25.
[151] Mulder WA. A high-resolution Euler solver based on multigrid, semi-coarsening, and defect correction. J Comput Phys 1992;100:91–104.
[152] Turkel E. Preconditioned methods for solving the incompressible and low speed compressible equations. J Comput Phys 1987;72:277.
[153] Choi YH, Merkle CL. The application of preconditioning in viscous flows. J Comput Phys 1993;105(2):207–23.
[154] Van Leer B, Lee WT, Roe PL. Characteristic time-stepping or local preconditioning of the Euler equations. AIAA Paper No. 91–1552; 1991.

[155] Ronquist EM, Patera AT. Spectral element multigrid I: formulation and numerical results. SIAM J Sci Comput 1987;2(4):389–406.

[156] Maday Y, Munoz R. Spectral element multigrid part 2: theoretical justification. ICASE, Technical Report 1988; 88–73.

[157] Bassi F, Rebay S. Numerical solution of the Euler equations with a multiorder discontinuous finite element method. In: Proceedings of the second international conference on computational fluid dynamics, Sydney, Australia, 15–19 July 2002.

[158] Helenbrook B, Mavriplis D, Atkins H. Analysis of p-multigrid for continuous and discontinuous finite element discretizations. AIAA Paper 2003–3989, AIAA, 2003.

[159] Luo H, Baum JD, Lohner R. A p-multigrid discontinuous Galerkin method for the Euler equations on unstructured grids. J Comput Phys 2006;211:767–83.

[160] Chung TJ. Computational fluid dynamics. Cambridge: Cambridge University Press; 2002.

[161] Gottlieb D, Hesthaven J. Spectral methods for hyperbolic problems. J Comput Appl Math 2001;128:83–131.

[162] Harten A, Chakravarthy S. Multi-dimensional ENO Schemes for general geometries, ICASE Report No. 91–76; 1991.

[163] Peraire J, Vahdati M, Morgan K, Zienkiewicz OC. Adaptive remeshing for compressible flow computations. J Comput Phys 1987;72:449–66.

[164] Roe PL. Characteristic-based schemes for the Euler equations. Annu Rev Fluid Mech 1986;18:337–65.

[165] Van Leer B. Flux-vector splitting for the Euler equations. Lecture notes in physics, 1982; vol. 170. p. 507.

[166] Vassberg JC. A fast, implicit unstructured-mesh Euler method. AIAA Paper No. 92–2693, 1992.