Journal of Computational Physics 423 (2020) 109756

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

High-order point-value enhanced finite volume method for two-dimensional hyperbolic equations on unstructured meshes

Li-Jun Xuan [*], Joseph Majdalani [*]

Department of Aerospace Engineering, Auburn University, Auburn, AL, 36849, USA

ARTICLE INFO

Article history:
Received 24 September 2019
Received in revised form 8 June 2020
Accepted 31 July 2020
Available online 10 August 2020

Keywords:
Discontinuous Galerkin
Compact high order
Approximate delta function
Point-value enhanced finite volume

ABSTRACT

Presently, an approximate delta function (ADF) is defined in multi-dimensional space in the form of a finite-order polynomial that holds identical integral properties to the Dirac delta function when used in conjunction with a finite-order polynomial integrand spanning a finite domain. By using this ADF over a two-dimensional unstructured mesh, a compact high-order point-value enhanced finite volume method (PFV) is constructed. The corresponding scheme is capable of storing and updating cell-averaged values inside each element as well as the unknown quantities and their derivatives (when needed) on vertex points. The sharing of vertex information with surrounding elements reduces the number of degrees of freedom relative to other compact and high-order methods of comparable convergence rate. To ensure conservation, cell-averaged values are updated using an identical approach to the one traditionally followed in the finite volume method. To verify the performance of the PFV scheme at successive orders, its accuracy and stability are vetted using benchmark problems associated with the two-dimensional wave and Euler equations encompassing a total of six standard test cases.

1. Introduction

As an extension to the singular Dirac delta distribution, Huynh [1,2] has introduced an elegant polynomial representation of an approximate delta function (ADF) in one-dimensional space. This ADF holds identical integral properties to those of a Dirac delta distribution when applied to a finite-order polynomial integrand over a finite domain. Along similar lines, Xuan and Majdalani [3] have examined and extended the ADF specification by allowing it to accommodate an arbitrary number of auxiliary coefficients while introducing, in the same context, an ADF derivative weight function that can be used to extract any order derivative of a finite-order polynomial at an arbitrary point.

Accordingly, an ADF can be used to represent the values and successive derivatives of finite-order polynomials at certain points through the use of straightforward integration over finite domains. Moreover, an ADF can be used as an effective tool to recover, extend, and construct high-order schemes in several computational fluid dynamics (CFD) applications.

Following this line of inquiry, Xuan and Majdalani [3] have managed to recover both Taylor-based and nodal-based Discontinuous Galerkin (DG) methods [4] as well as the flux reconstruction method (FR) [5,6,1] using ADF as a vehicle. Furthermore, by leveraging ADF properties, Xuan and Majdalani [3] have introduced a compact high-order point-value enhanced

[*] Corresponding authors.
E-mail addresses: LJ.Xuan@auburn.edu (L.-J. Xuan), joe.majdalani@auburn.edu (J. Majdalani).

https://doi.org/10.1016/j.jcp.2020.109756
0021-9991/© 2020 Elsevier Inc. All rights reserved.

finite volume (PFV) method in one-dimensional space that has proven to be both stable and accurate. For this reason, the present study aims at extending the PFV approach to a two-dimensional setting, specifically by redesigning the framework presented in Ref. [3] to the general solution of the conservation equations.

Traditionally, low-order methods are preferred in computational fluid dynamics applications because of their simplicity and suitability to tackle a large array of phenomenological problems. The alternative is to employ high-order methods, which have the potential to produce more accurate solutions, although the increase in precision is usually offset by elevated degrees of complexity and reduced robustness. The effort to improve the accuracy of high-order methods while sustaining their stability and performance characteristics can thus be viewed as a vital and recurring research topic.

For example, the favorable attributes associated with the Discontinuous Galerkin (DG) method have expedited its acceptance as one of the most ubiquitously used high-order methods in the treatment of the Navier–Stokes equations. After being implemented in the formulation of a neutron transport problem by Reed and Hill [4], it has been further explored by LaSaint and Raviart [7] and then generalized in the context of computational fluid mechanics by Cockburn [8,9], Shu [10], Bassi [11,12], and others. At a fundamental level, the DG method provides approximations of the Galerkin type to partial differential equations (PDEs) that are applicable to a finite element; it then transforms the ensuing equations into a set of ordinary differential equations (ODEs) that are amenable to numerical integration.

Fundamentally, several other methods that are based on differential discretization schemes have been developed with the aim of achieving high-order accuracy. Amongst them, it may be worth noting the staggered-grid spectral technique [13], the spectral difference technique [14,15], and the spectral volume technique [16]. One may also cite the method of flux reconstruction (FR) [5,6,1], which has later morphed into the correction procedure via reconstruction (CPR) [17–19].

In related work, a constrained interpolation profile (CIP) with multi-moment finite volume (MFV) method has been judiciously formulated by Xiao et al. [20]. In addition to its ability to store the cell-averaged value of a given element as traditionally performed, MFV may be noted for allowing the incorporation of extra degrees of freedom (DOFs) that can be distributed, rather unconventionally, on the element's edge and nodal points. By specifically sharing this additional information with neighboring cells, the MFV framework may be perceived as an efficient scheme for reducing the number of DOFs relative to other high-order schemes exhibiting similar error margins. Moreover, the ability of MFV to share extra DOFs while securing mass conservation has been shown to enhance the scheme's robustness. In similar context, it may be worth mentioning the Active Flux (AF) method [21,22], which treats the unknown values at edge-based flux points as independent DOFs that can be refreshed while marching in time.

Bearing these efforts in mind, the cell-averaged value of each element in Ref. [3] are stored along with multiple DOFs that comprise both values and derivatives at cell interfaces. In fact, a general DOF setting for storing multiple DOFs on an element and its interfaces is presented quite elegantly by Huynh [23] for the linear, one-dimensional wave equation.

In two-dimensional space, it may be recognized that cell vertices are shared by more elements than edges, and are therefore subject to a higher sharing frequency than any other spatial position or interface. Consequently, it may be posited that placing all additional DOFs on vertices will give rise to an efficient platform to enhance element-to-element communication while increasing the amount of data shared. As demonstrated previously in Ref. [3], it is possible to increase the amount of information being shared all the while leveraging the benefits of both MFV and AF techniques by placing, not only the unknown point values, but also the values of the derivatives, at the vertices themselves. Furthermore, updating the cell-averaged values in a manner that is identical to that used by the finite volume (FV) approach can be adopted to ensure the physical conservation of the scheme. Finally, by realizing that the accuracy of the scheme only increases with the amount of data stored at optimally chosen points, the present technique may be coined, consistently with Ref. [3], a "point-value enhanced finite volume (PFV)" method.

To be more specific, the scheme that we pursue here will be denoted as $\mathrm{P}_n\mathrm{FV}$ ($n \in \mathbb{N}$) when the coefficients of an nth-order polynomial are stored on the vertex. The scheme is further designated as $\mathrm{P}_n\mathrm{FV}_m$ ($m \in \mathbb{N}$) when the unknown function is approximated by an mth-order polynomial in each element, with a practical choice of $m \geq n$. In fact, if more DOFs are assigned to an element, as in the DG case, then the scheme can be further extended using the $\mathrm{P}_n\mathrm{DG}_m^k$ ($k \in \mathbb{N}$) notation, where the coefficients of a kth-order polynomial are stored inside the element, while practically keeping $k \leq m$. This point is further elaborated by Huynh [23] in his treatment of the linear, one-dimensional wave equation. As such, and in the remainder of this study, the characterization of the $\mathrm{P}_n\mathrm{FV}_m$ scheme, which restores the $\mathrm{P}_n\mathrm{DG}_m^0$ scheme as a special case, will constitute our main focus. From a practical standpoint, the implementation of additional DOFs to increase the amount of element-wise information in the context of a $\mathrm{P}_n\mathrm{FV}_m$ formulation has the potential to be more effective than in the $\mathrm{P}_n\mathrm{DG}_m^k$ framework for $k > 0$.

To further elaborate, we remark that the manner by which the updating of additional information is achieved on nodal points and edges constitutes, perhaps, the most essential aspect of the MFV and AF schemes. For example, the use of the characteristic solution of the linear wave equation in Ref. [23] is implemented in accordance with the AF scheme. In the present work, we implement the ADF procedure developed in Ref. [3] and set the integral domain to encompass all of the elements surrounding the vertex in question. Moreover, by increasing the radius of influence that accompanies each update, the stability of the scheme is systematically enhanced. As for the temporal updating of the ODEs associated with each DOF, a conventional third-order total variation diminishing (TVD) Runge-Kutta scheme will be used [24].

It may be instructive to note, at the outset, that although the PFV and Huynh's $\mathrm{P}\mu\mathrm{I}\nu$ ($\mu, \nu \in \mathbb{N}$) methods [23] share the same DOF setting, they are different in two particular aspects:

1. The underlining approximate solution in the $\mathrm{P}\mu\mathrm{I}\nu$ method remains continuous across the interface of each element. In

the PFV method, the underlining approximation is always discontinuous at all interfaces. Consequently, it is possible to use the Riemann flux, which is known to enhance the stability of the scheme both locally and globally, for either linear or nonlinear equations.
2. The PμIν method relies on the characteristic solution of the linear wave equation to update the interfacial DOFs. In the PFV method, the ADF procedure is adopted, thus granting it broader versatility and selectivity in manipulating the conservation equations, especially for nonlinear equations in multi-dimensional space.

This essential sequel is organized as follows. The definition of a multi-dimensional approximate delta function is first introduced in Section 2. This is followed by Section 3, which details the formulation of the ADF polynomials in two-dimensional space. By leveraging the two-dimensional ADF concept, the construction of the PFV method is presented in Section 4 and then illustrated in Section 5, where its performance and effectiveness in resolving several benchmark problems, such as the two-dimensional linear wave and Euler equations, with specific applications to an assortment of classical examples, are systematically described. The ensuing simulations help to verify the accuracy and stability associated with the PFV technique based on its maximum CFL values and error convergence rates with successive mesh refinements. The ability of the PFV scheme to rely on a small number of DOFs relative to the DG method of the same order is discussed in Section 6. Lastly, several concluding remarks are offered in Section 7.

2. Approximate delta function (ADF) in multi-dimensional space

As described in Ref. [3], the Nth-order ADF polynomial in multiple dimensions may be defined as

$$\int_{\Omega} P_N(\boldsymbol{x})\tilde{\delta}_N(\boldsymbol{x},\boldsymbol{z})\,\mathrm{d}\boldsymbol{x} = P_N(\boldsymbol{z}), \tag{1}$$

where $\boldsymbol{x},\boldsymbol{z}\in\mathbb{R}^d$, "$d$" being the number of spatial dimensions, and $P_N(\boldsymbol{x})$ stands for an arbitrary Nth-order polynomial. At the outset, $\tilde{\delta}_{N|M}(\boldsymbol{x},\boldsymbol{z})$ can be used to designate an $(N+M)$th-order polynomial that satisfies

$$\int_{\Omega} P_N(\boldsymbol{x})\tilde{\delta}_{N|M}(\boldsymbol{x},\boldsymbol{z})\,\mathrm{d}\boldsymbol{x} = P_N(\boldsymbol{z}), \tag{2}$$

where, as shown in Ref. [3], the total number of arbitrary coefficients, N_C, that can be accommodated by equation (2) is

$$N_C = \frac{(N+d+M)!}{(N+M)!d!} - \frac{(N+d)!}{N!d!}.$$

Note that for $M=0$, one recovers $\tilde{\delta}_{N|0}(\boldsymbol{x},\boldsymbol{z}) = \tilde{\delta}_N(\boldsymbol{x},\boldsymbol{z})$.

With the above definition in hand, the $(N+M)$th-order ADF polynomial derivative weight function that is needed to calculate the nth-order derivative of $P_N(\boldsymbol{z})$ with respect to $\{z_1, z_2, \cdots\}$ may be written as

$$\int_{\Omega} P_N(\boldsymbol{x})\frac{\partial^n\tilde{\delta}_{N|M}(\boldsymbol{x},\boldsymbol{z})}{\partial z_1^{n_1}\partial z_2^{n_2}\cdots}\,\mathrm{d}\boldsymbol{x} = \frac{\partial^n P_N(\boldsymbol{z})}{\partial z_1^{n_1}\partial z_2^{n_2}\cdots}, \tag{3}$$

where $n = n_1 + n_2 + \cdots$.

3. ADF representation in two spatial dimensions

In a given domain Ω, one may define the operator

$$\langle p, q\rangle = \int_{\Omega} pq\,\mathrm{d}x\,\mathrm{d}y, \tag{4}$$

which appears in textbooks on quantum mechanics (e.g., Sakurai [25]), and is commonly used in the description of the DG method [10]. Denoting a vector $\boldsymbol{x}=(x,y)$ in two-dimensional space, in order to evaluate $\tilde{\delta}_0(\boldsymbol{x},\boldsymbol{x}_p)$, $\tilde{\delta}_1(\boldsymbol{x},\boldsymbol{x}_p)$, and $\tilde{\delta}_2(\boldsymbol{x},\boldsymbol{x}_p)$, one may specify the point $\boldsymbol{x}_p=(x_p,y_p)$ at the origin of the coordinate system. In what follows, the zeroth, first, and second-order ADF representations are described along with the ADF's piecewise polynomial representation.

3.1. Zeroth-order ADF formulation

For the zeroth-order polynomial $\tilde{\delta}_0(\boldsymbol{x},\boldsymbol{0})$, we have

$$\langle\tilde{\delta}_0(\boldsymbol{x},\boldsymbol{0}),1\rangle = 1 \text{ or } \tilde{\delta}_0(\boldsymbol{x},\boldsymbol{0}) = \frac{1}{A_\Omega}, \tag{5}$$

where A_Ω represents the surface area of the domain under consideration, Ω.

3.2. First-order ADF formulation

For $P_1(\boldsymbol{x}) = r_{0,0} + r_{1,0}x + r_{0,1}y$, we can take $\tilde{\delta}_1(\boldsymbol{x}, \mathbf{0}) = a_{0,0} + a_{1,0}x + a_{0,1}y$. Subsequently, the definition of $\tilde{\delta}_1$, namely,

$$\langle \tilde{\delta}_1(\boldsymbol{x}, \mathbf{0}), P_1(\boldsymbol{x}) \rangle = P_1(0) = r_{0,0} \tag{6}$$

leads to

$$\langle \tilde{\delta}_1(\boldsymbol{x}, \mathbf{0}), 1 \rangle = 1,\ \langle \tilde{\delta}_1(\boldsymbol{x}, \mathbf{0}), x \rangle = 0,\ \langle \tilde{\delta}_1(\boldsymbol{x}, \mathbf{0}), y \rangle = 0 \tag{7}$$

or

$$\boldsymbol{Aa} = \boldsymbol{l}_1, \tag{8}$$

where

$$\boldsymbol{A} = \begin{pmatrix} \langle 1,1 \rangle & \langle x,1 \rangle & \langle y,1 \rangle \\ \langle 1,x \rangle & \langle x,x \rangle & \langle y,x \rangle \\ \langle 1,y \rangle & \langle x,y \rangle & \langle y,y \rangle \end{pmatrix}, \tag{9}$$

$$\boldsymbol{a} = \left(a_{0,0}, a_{1,0}, a_{0,1}\right)^T \quad \text{and} \quad \boldsymbol{l}_1 = (1,0,0)^T. \tag{10}$$

The coefficients of the ADF polynomial are thus at hand. Similarly, by taking the ADF derivative weight function as a two-variable polynomial of the form $\tilde{\delta}'_{1,x}(\boldsymbol{x}, \mathbf{0}) = b_{0,0} + b_{1,0}x + b_{0,1}y$, we may specify it as

$$\langle \tilde{\delta}'_{1,x}(\boldsymbol{x}, \mathbf{0}), P_1(\boldsymbol{x}) \rangle = \left.\frac{\partial P_1}{\partial x}\right|_p = r_{1,0}. \tag{11}$$

By substituting the polynomial expression into the above expression, $\tilde{\delta}'_{1,x}(\boldsymbol{x}, \mathbf{0})$ can be determined from following linear system

$$\boldsymbol{Ab} = \boldsymbol{l}_2, \tag{12}$$

where

$$\boldsymbol{b} = \left(b_{0,0}, b_{1,0}, b_{0,1}\right)^T \quad \text{and} \quad \boldsymbol{l}_2 = (0,1,0)^T. \tag{13}$$

For the y-derivative, we may equivalently assume an ADF derivative weight function of the form $\tilde{\delta}'_{1,y}(\boldsymbol{x}, \mathbf{0}) = c_{0,0} + c_{1,0}x + c_{0,1}y$. As before, we get

$$\boldsymbol{Ac} = \boldsymbol{l}_3, \tag{14}$$

where

$$\boldsymbol{c} = \left(c_{0,0}, c_{1,0}, c_{0,1}\right)^T \quad \text{and} \quad \boldsymbol{l}_3 = (0,0,1)^T. \tag{15}$$

3.3. Second-order ADF formulation

At this stage, our two-variable polynomial representation can be written as

$$\begin{aligned} &\tilde{\delta}_2(\boldsymbol{x}, \mathbf{0}) = a_{0,0} + a_{1,0}x + a_{0,1}y + a_{2,0}x^2 + a_{1,1}xy + a_{0,2}y^2, \\ &\tilde{\delta}'_{2,x}(\boldsymbol{x}, \mathbf{0}) = b_{0,0} + b_{1,0}x + b_{0,1}y + b_{2,0}x^2 + b_{1,1}xy + b_{0,2}y^2, \\ &\tilde{\delta}'_{2,y}(\boldsymbol{x}, \mathbf{0}) = c_{0,0} + c_{1,0}x + c_{0,1}y + c_{2,0}x^2 + c_{1,1}xy + c_{0,2}y^2. \end{aligned} \tag{16}$$

The same approach described above yields the multi-dimensional matrix equalities,

$$\boldsymbol{Aa} = \boldsymbol{l}_1,\ \boldsymbol{Ab} = \boldsymbol{l}_2, \boldsymbol{Ac} = \boldsymbol{l}_3, \tag{17}$$

where $\boldsymbol{l}_1 = (1,0,0,0,0,0)^T$, $\boldsymbol{l}_2 = (0,1,0,0,0,0)^T$, $\boldsymbol{l}_3 = (0,0,1,0,0,0)^T$, and

$$\boldsymbol{A} = \begin{pmatrix} \langle 1,1 \rangle & \langle x,1 \rangle & \langle y,1 \rangle & \langle x^2,1 \rangle & \langle xy,1 \rangle & \langle y^2,1 \rangle \\ \langle 1,x \rangle & \langle x,x \rangle & \langle y,x \rangle & \langle x^2,x \rangle & \langle xy,x \rangle & \langle y^2,x \rangle \\ \langle 1,y \rangle & \langle x,y \rangle & \langle y,y \rangle & \langle x^2,y \rangle & \langle xy,y \rangle & \langle y^2,y \rangle \\ \langle 1,x^2 \rangle & \langle x,x^2 \rangle & \langle y,x^2 \rangle & \langle x^2,x^2 \rangle & \langle xy,x^2 \rangle & \langle y^2,x^2 \rangle \\ \langle 1,xy \rangle & \langle x,xy \rangle & \langle y,xy \rangle & \langle x^2,xy \rangle & \langle xy,xy \rangle & \langle y^2,xy \rangle \\ \langle 1,y^2 \rangle & \langle x,y^2 \rangle & \langle y,y^2 \rangle & \langle x^2,y^2 \rangle & \langle xy,y^2 \rangle & \langle y^2,y^2 \rangle \end{pmatrix}. \tag{18}$$

This assortment of three equations enables us to solve for the coefficients of $\tilde{\delta}_2$ and the two first-order derivative weight functions ($\tilde{\delta}'_{2,x}$, $\tilde{\delta}'_{2,y}$), namely,

$$\begin{aligned}
\boldsymbol{a} &= \left(a_{0,0}\; a_{1,0}\; a_{0,1}\; a_{2,0}\; a_{1,1}\; a_{0,2}\right)^T, \\
\boldsymbol{b} &= \left(b_{0,0}\; b_{1,0}\; b_{0,1}\; b_{2,0}\; b_{1,1}\; b_{0,2}\right)^T, \\
\boldsymbol{c} &= \left(c_{0,0}\; c_{1,0}\; c_{0,1}\; c_{2,0}\; c_{1,1}\; c_{0,2}\right)^T.
\end{aligned} \tag{19}$$

Although it is equally straightforward to evaluate the ADF weight functions for the second-order derivatives $\tilde{\delta}''_{2,xx}$, $\tilde{\delta}''_{2,xy}$, and $\tilde{\delta}''_{2,yy}$, they are omitted here because they will not affect the numerical scheme at the order to be presently explored.

3.4. Derivation of ADF polynomials for special cases

In the interest of clarity, we now illustrate step-by-step the manner by which ADF polynomials can be generated for several particular cases. We start with a square domain, which is demarcated by four vertices $(-1,-1)$, $(1,-1)$, $(1,1)$, and $(-1,1)$; in this case, one finds $A_\Omega = 4$ and then use (5) to evaluate δ_0. For higher-order ADFs, one computes the following volume integrals on the square domain:

$$\langle x^m, y^n\rangle = \int_\Omega x^m y^n \,\mathrm{d}V = \frac{\left[1-(-1)^{m+1}\right]\left[1-(-1)^{n+1}\right]}{(m+1)(n+1)}, \quad n, m = 0, 1, 2, \cdots \tag{20}$$

This enables us to determine the components of the coefficient matrix $\boldsymbol{A}$, which is given by (9) for the first-order ADF and (18) for the second-order ADF. The linear equations (17) corresponding to each ADF polynomial can then be solved, thus leading to,

$$\begin{aligned}
&\tilde{\delta}_0(\boldsymbol{x},\boldsymbol{0}) = \tilde{\delta}_1(\boldsymbol{x},\boldsymbol{0}) = \tfrac{1}{4},\; \tilde{\delta}_2(\boldsymbol{x},\boldsymbol{0}) = \tilde{\delta}_3(\boldsymbol{x},\boldsymbol{0}) = \tfrac{7}{8} - \tfrac{15}{16}\left(x^2+y^2\right), \\
&\tilde{\delta}_4(\boldsymbol{x},\boldsymbol{0}) = \tfrac{1}{256}\left[486 - 1350\left(x^2+y^2\right) + 945\left(x^4+y^4\right) + 900x^2y^2\right];
\end{aligned} \tag{21}$$

and

$$\tilde{\delta}'_{1,x}(\boldsymbol{x},\boldsymbol{0}) = \tilde{\delta}'_{2,x}(\boldsymbol{x},\boldsymbol{0}) = \tfrac{3}{4}x,\; \tilde{\delta}'_{3,x}(\boldsymbol{x},\boldsymbol{0}) = \tilde{\delta}'_{4,x}(\boldsymbol{x},\boldsymbol{0}) = \tfrac{15}{16}x\left(6 - 7x^2 - 3y^2\right). \tag{22}$$

Then taking advantage of the problem's inherent symmetry, we can deduce $\tilde{\delta}'_{\cdot,y}$ by swapping x and y in $\tilde{\delta}'_{\cdot,x}$.

Our next example is a triangular domain that is prescribed by three vertices $(-1/3,-1/3)$, $(2/3,-1/3)$, and $(-1/3,2/3)$. Following the same approach, we proceed by calculating the coefficients of matrix $\boldsymbol{A}$ and then solve the linear system (17) to determine the ADFs for point $\boldsymbol{0} = (0,0)$. We get

$$\begin{aligned}
\tilde{\delta}_0(\boldsymbol{x},\boldsymbol{0}) &= \tilde{\delta}_1(\boldsymbol{x},\boldsymbol{0}) = 2, \quad \tilde{\delta}_2(\boldsymbol{x},\boldsymbol{0}) = \tilde{\delta}_3(\boldsymbol{x},\boldsymbol{0}) = \tfrac{16}{3} - 40\left(x^2+xy+y^2\right), \\
\tilde{\delta}_4(\boldsymbol{x},\boldsymbol{0}) &= \tfrac{890}{81} - \tfrac{560}{3}\left(x^2+xy+y^2+x^2y+xy^2\right) \\
&\quad + \tfrac{1400}{3}\left[x^4 + 2xy(x^2+y^2) + 3x^2y^2 + y^4\right].
\end{aligned} \tag{23}$$

$$\begin{aligned}
&\tilde{\delta}'_{1,x}(\boldsymbol{x},\boldsymbol{0}) = 48x + 24y, \quad \tilde{\delta}'_{2,x}(\boldsymbol{x},\boldsymbol{0}) = 240xy + 120y^2 + 80x + 40y, \\
&\tilde{\delta}'_{3,x}(\boldsymbol{x},\boldsymbol{0}) = \tfrac{1000}{9}(2x+y) - \tfrac{280}{3}\left\{8\left[x^3 + 3xy(x+y) + y^3\right] + 2xy + y^2\right\}, \\
&\tilde{\delta}'_{4,x}(\boldsymbol{x},\boldsymbol{0}) = \tfrac{9800}{27}(2x+y) - 2800\left[2xy(6x^2 + 9xy + 4y^2) + y^4\right] \\
&\qquad + \tfrac{1120}{3}\left\{2xy + y^2 - 12\left[2x^3 + 3xy(x+y) + y^3\right]\right\}.
\end{aligned} \tag{24}$$

Again a swapping of x and y in $\tilde{\delta}'_{\cdot,x}$ enables us to specify $\tilde{\delta}'_{\cdot,y}$.

3.5. Piecewise polynomial ADF formulation

So far, the ADF formulation has consisted of finite-order polynomials. In practice, it is also possible to represent the ADF as a piecewise polynomial. For example, we have the ability to define at a given vertex p,

$$\tilde{\delta}_N^{(p)}(\boldsymbol{x},\boldsymbol{x}_p) = \sum_k w_k \tilde{\delta}_N^{(p_k)}(\boldsymbol{x},\boldsymbol{x}_p), \quad w_k > 0, \quad \sum_k w_k = 1, \tag{25}$$

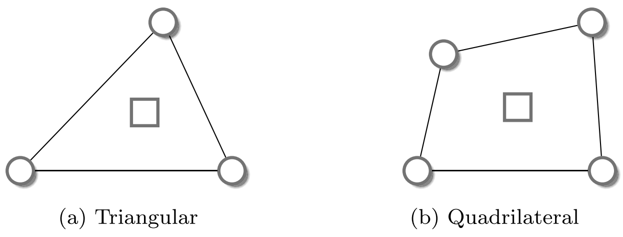


Fig. 1. Setting of DOFs for triangular and quadrilateral elements. Here the symbol □ denotes a cell-averaged value while ○ refers to a solution point DOF.

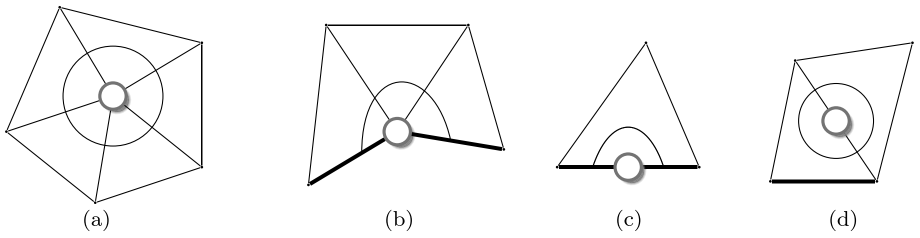


Fig. 2. Integral domain of an ADF on a triangular mesh where the SP is assigned to different key locations: (a) central region SP, (b) boundary vertex SP, (c) external boundary edge SP, and (d) internal boundary edge SP. Here the thick black line is used to demarcate the edge of the boundary.

where $\tilde{\delta}_N^{(p_k)}(\boldsymbol{x}, \boldsymbol{x}_p)$ denotes a finite-order ADF polynomial defined on each of the elements Ω_k sharing the same vertex p, and which can then vanish on the rest of the elements according to

$$\tilde{\delta}_N^{(p_k)}(\boldsymbol{x}, \boldsymbol{x}_p) = \begin{cases} \tilde{\delta}_N(\boldsymbol{x}, \boldsymbol{x}_p) & \boldsymbol{x} \in \Omega_k \\ 0 & \boldsymbol{x} \in \Omega \backslash \Omega_k \end{cases}, \tag{26}$$

where the $\Omega = \bigcup_k \Omega_k$ domain encompasses all of the surrounding elements that share the same vertex p.

To make further headway, and for simplicity's sake, only equal weights such as $w_1 = w_2 = \cdots$ will be selected from this point forward. Subsequently, owing in large part to a wide range of preliminary numerical simulations, we are able to confirm that the discontinuous piecewise polynomial formulation leads to greater accuracy than the single continuous polynomial representation. For this reason, only the discontinuous piecewise polynomial formulation will be considered in the remainder of this study. The discussion of continuous polynomial representations will be relegated to a separate study.

4. Point-value enhanced finite volume (PFV) method

As a direct extension of the PFV method in one-dimensional space [3], we distribute the DOFs on the mesh according to Fig. 1, where circles depict values recorded on the vertex, and squares refer to cell-averaged values at the center of the element. In what follows, each circular point is coined a solution point (SP). As indicated previously, we use $\mathrm{P_0FV}$ to denote the method when only a value is saved on the SP. Similarly, we call it $\mathrm{P_1FV}$ when a value and two derivatives are recorded. In like manner, the designation $\mathrm{P}_n\mathrm{FV}$ is used when the coefficients of an nth-order polynomial are stored. Presently, the emphasis will be placed on the behavior of $\mathrm{P_0FV}$ and $\mathrm{P_1FV}$ to better understand their characteristics before venturing into even higher-order schemes.

To derive the updating ODEs for the SP values and derivatives (whenever inclusion of such derivatives is necessary), we use the ADF procedure and allow the integral domain to encompass all of the elements surrounding the SP, as shown in Fig. 2(a) for a central region element. In this case, the integral domain incorporates all elements swept by the curved line. For a boundary element, the integral domain illustrated in Fig. 2(b) annexes the adjacent elements that are delimited by the thick line at the domain's boundary. In fact, our numerical simulations show that, for an element with a boundary edge, an extra SP on the face of the element is sometimes required to stabilize the scheme at the third order. As for the integral domain associated with an edge located SP, it is shown in Figs. 2(c) and (d) for both external and internal boundary edges. It can thus be seen that the application of ADF tools enables us to position the SP at flexible locations, particularly, at sites that can be used to enhance the scheme's stability as well as its accuracy in a manner that is compatible with the requirements associated with a given problem.

To illustrate the construction of the PFV method at different orders, we now proceed to consider the scalar conservation equation,

$$\frac{\partial u}{\partial t} + \nabla \cdot \boldsymbol{F}(u) = 0. \tag{27}$$

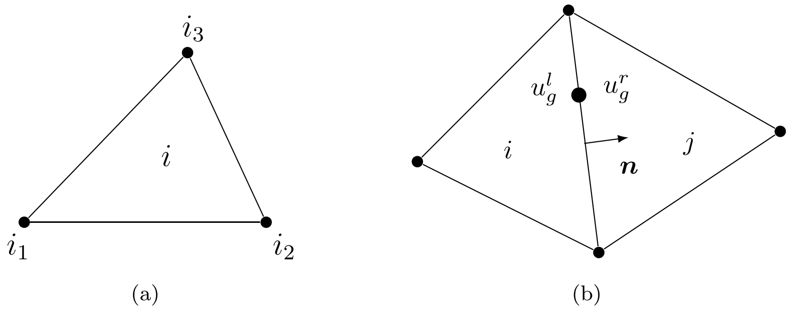


Fig. 3. Illustration of (a) the reconstruction stencil and (b) interfacial flux computation terminology.

This form can be used to reproduce several classical equations in mechanics and physics. For example, by taking $\boldsymbol{F}(u) = \frac{1}{2}(u^2, u^2)$, this equation returns Burgers' equation in two-dimensional space; similarly, by setting $\boldsymbol{F}(u) = (au, bu)$, with constant a and b, we recover the linear wave equation. In what follows, each step needed to construct the PFV framework is sequentially described.

4.1. P_0FV_1 development

4.1.1. Reconstruction steps

In the P_0FV formulation, it is only necessary to record the value of the unknown function, u_p, on the SP. The solution in each element can be reconstructed as a first-order polynomial using the least-squares method based on the SP values assigned to the vertex of each element. The approximate solution in the element may be written as

$$\tilde{u}_i = \bar{u}_i + a\xi + b\eta, \; \xi = \frac{x - x_{c_i}}{\Delta x_i} \quad \text{and} \quad \eta = \frac{y - y_{c_i}}{\Delta y_i} \tag{28}$$

where $\bar{u}_i$ denotes the cell-averaged value, the (x_{c_i}, y_{c_i}) pair specifies the coordinates at the center of the element, and $(\Delta x_i, \Delta y_i)$ represent the geometrical length scales of element i in the x and y directions; these may be specified as

$$\Delta x_i = \max_{j,k} |x_{i_j} - x_{i_k}|, \quad \Delta y_i = \max_{j,k} |y_{i_j} - y_{i_k}|, \tag{29}$$

where i_j and i_k stand for the indices of any two vertex points in Fig. 3(a). We further introduce $\xi_{i_k} = (x_{i_k} - x_{c_i})/\Delta x_i$ and $\eta_{i_k} = (y_{i_k} - y_{c_i})/\Delta y_i$. Consequently, the information on the SP of a triangular element may be expressed as

$$\begin{pmatrix} \xi_{i_1} & \eta_{i_1} \\ \xi_{i_2} & \eta_{i_2} \\ \xi_{i_3} & \eta_{i_3} \end{pmatrix} \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} u_{i_1} - \bar{u}_i \\ u_{i_2} - \bar{u}_i \\ u_{i_3} - \bar{u}_i \end{pmatrix}. \tag{30}$$

Equation (30) leads to an overdetermined system that can be solved straightforwardly using the normal equation method, i.e., by minimizing the sum of the square differences between the left and right-hand sides.

4.1.2. Updating ODEs

The half-discretized ODEs for the updating of $\bar{u}_i$ can be taken to mirror the finite volume procedure by setting

$$\frac{\mathrm{d}\bar{u}_i}{\mathrm{d}t} = -\int\limits_{\partial\Omega_i} \boldsymbol{n} \cdot \boldsymbol{F}(u)\, \mathrm{d}S \approx -\int\limits_{\partial\Omega_i} \hat{F}(\boldsymbol{n}, u^l, u^r)\, \mathrm{d}S, \tag{31}$$

where the face integral is calculated by Gaussian quadrature of the facial Riemann flux, $\hat{F}(\boldsymbol{n}, u^l, u^r)$; the latter is evaluated at the Gaussian point based on the values interpolated from the left and right elements, as shown in Fig. 3(b). These are given by

$$\begin{aligned} u_g^l &= \bar{u}_i + a_i \frac{x_g - x_{c_i}}{\Delta x_i} + b_i \frac{y_g - y_{c_i}}{\Delta y_i}, \\ u_g^r &= \bar{u}_j + a_j \frac{x_g - x_{c_j}}{\Delta x_i} + b_j \frac{y_g - y_{c_j}}{\Delta y_i}. \end{aligned} \tag{32}$$

Here the coordinates (x_g, y_g) locate each Gaussian point on the face of the element. A Riemann flux can be subsequently used to compute the numerical flux at each Gaussian point simply from

$$\hat{F}_{\boldsymbol{n},g} = F(\boldsymbol{n}, u_g^l, u_g^r). \tag{33}$$

In this vein, the updating ODEs for SP values may be retrieved from

$$\begin{aligned}\frac{\mathrm{d}u_p}{\mathrm{d}t} &= \int\limits_\Omega \frac{\partial u}{\partial t}\tilde{\delta}_1(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}V = -\int\limits_\Omega \nabla\cdot\boldsymbol{F}(u)\tilde{\delta}_1(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}V\\ &= \int\limits_\Omega \boldsymbol{F}(u)\cdot\nabla\tilde{\delta}_1(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}V - \int\limits_{\partial\Omega}\boldsymbol{n}\cdot\boldsymbol{F}\tilde{\delta}_1(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}S\\ &\approx \int\limits_\Omega \boldsymbol{F}(u)\cdot\nabla\tilde{\delta}_1(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}V - \int\limits_{\partial\Omega}\hat{F}(\boldsymbol{n},u^l,u^r)\tilde{\delta}_1(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}S,\end{aligned} \tag{34}$$

where the $\Omega=\bigcup_m\Omega_m$ domain comprises all of the elements surrounding the SP, and $\partial\Omega$ encompasses all of the edges on Ω_m; in the above, $\tilde{\delta}_1(\boldsymbol{x},\boldsymbol{x}_p)$ represents the first-order ADF for the SP defined on Ω.

Then given the element-wise reconstructed $\tilde{u}_i(x,y)$, the domain integral can be carried out using a Gaussian quadrature in each element pursuant to

$$\int\limits_{\Omega_i}\boldsymbol{F}(u)\cdot\nabla\tilde{\delta}_1(\boldsymbol{x})\,\mathrm{d}V = \sum_g w_g\boldsymbol{F}(u_g)\cdot\nabla\tilde{\delta}_1(\boldsymbol{x}_g), \tag{35}$$

where $u_g=\bar{u}_i+a_i(x_g-x_{c_i})/\Delta x_i+b_i(y_g-y_{c_i})/\Delta y_i$, and w_g is the Gaussian weight. One can then write

$$\int\limits_{\Omega}\boldsymbol{F}(u)\cdot\nabla\tilde{\delta}_1(\boldsymbol{x})\,\mathrm{d}V = \sum_m\int\limits_{\Omega_m}\boldsymbol{F}(u)\cdot\nabla\tilde{\delta}_1(\boldsymbol{x})\,\mathrm{d}V, \tag{36}$$

where the all-inclusive $\bigcup_m\Omega_m$ domain incorporates all of the elements surrounding the SP. Note that the domain integrals remain unchanged for all of the forthcoming PFV construction effort except for the expression needed to calculate the value on the Gaussian point, u_g; the latter can be obtained using the reconstructed polynomial approximation, $\tilde{u}_i(x,y)$, which is specific to each PFV order. For this reason, the detailed description of the domain integrals will not be repeated in the remainder of this work.

4.2. P_1FV_1 development

4.2.1. Reconstruction steps

In the $\mathrm{P_1FV}$ design, the information saved on the SP can be written as

$$(u_p,u_{1,p},u_{2,p})\equiv\left(u_p,\Delta x_p\frac{\partial u_p}{\partial x},\Delta y_p\frac{\partial u_p}{\partial y}\right), \tag{37}$$

where $(\Delta x_p,\Delta y_p)$ denote the length scales on the SP, which can be calculated from the average of the lengths $(\Delta x_m,\Delta y_m)$ of the surrounding elements. Here a weighted least-squares reconstruction is used to build a first-order polynomial on each element. This is accomplished by defining

$$\boldsymbol{B}=\begin{pmatrix}\xi_{i_1} & \frac{\Delta x_{i_1}}{\Delta x_i} & 0 & \xi_{i_2} & \frac{\Delta x_{i_2}}{\Delta x_i} & 0 & \xi_{i_3} & \frac{\Delta x_{i_3}}{\Delta x_i} & 0\\ \eta_{i_1} & 0 & \frac{\Delta y_{i_1}}{\Delta y_i} & \eta_{i_2} & 0 & \frac{\Delta y_{i_2}}{\Delta y_i} & \eta_{i_3} & 0 & \frac{\Delta y_{i_3}}{\Delta y_i}\end{pmatrix}, \tag{38}$$

$$\boldsymbol{W}=\mathrm{diag}\{1,w,w,1,w,w,1,w,w\}(w\ge 0),\qquad \boldsymbol{a}=(a,b)^T,$$

$$\text{and}\qquad \check{\boldsymbol{u}}=\left(u_{i_1}-\bar{u}_i,u_{1,i_1},u_{2,i_1},u_{i_2}-\bar{u}_i,u_{1,i_2},u_{2,i_2},u_{i_3}-\bar{u}_i,u_{1,i_3},u_{2,i_3}\right)^T.$$

In this context, the linear system's vector $\boldsymbol{a}$ that determines the solution coefficients may be deduced from

$$\boldsymbol{BWB}^T\boldsymbol{a}=\boldsymbol{BW}\check{\boldsymbol{u}}. \tag{39}$$

It may be readily confirmed that when $w=0$, the reconstruction degenerates to the one associated with $\mathrm{P_0FV_1}$.

4.2.2. Updating ODEs

The updating ODEs for the cell-averaged and nodal values remain identical to those developed for $\mathrm{P_0FV_1}$. Here two additional equations are required to capture the evolution of nodal derivatives. These are

$$\frac{\mathrm{d}u_{1,p}}{\mathrm{d}t}\approx\int\limits_\Omega\boldsymbol{F}(u)\cdot\nabla\tilde{\tilde{\delta}}'_{1,x}(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}V-\int\limits_{\partial\Omega}\hat{F}(\boldsymbol{n},u^l,u^r)\tilde{\tilde{\delta}}'_{1,x}(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}S, \tag{40}$$

and

$$\frac{\mathrm{d}u_{2,p}}{\mathrm{d}t} \approx \int\limits_{\Omega} \boldsymbol{F}(u)\cdot\nabla\tilde{\tilde{\delta}}'_{1,y}(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}V - \int\limits_{\partial\Omega} \hat{F}(\boldsymbol{n},u^l,u^r)\tilde{\tilde{\delta}}'_{1,y}(\boldsymbol{x},\boldsymbol{x}_p)\,\mathrm{d}S, \tag{41}$$

where $\tilde{\tilde{\delta}}'_{1,x}(\boldsymbol{x},\boldsymbol{x}_p)=\Delta x_p\tilde{\delta}'_{1,x}(\boldsymbol{x},\boldsymbol{x}_p)$ and $\tilde{\tilde{\delta}}'_{1,y}(\boldsymbol{x},\boldsymbol{x}_p)=\Delta y_p\tilde{\delta}'_{1,y}(\boldsymbol{x},\boldsymbol{x}_p)$.

4.3. $\mathrm{P}_1\mathrm{FV}_2$ development

4.3.1. Reconstruction steps

As the order is increased, the weighted least-squares may be used again to reconstruct a second-order polynomial from the information available at the center of the element and its nodes. We now set

$$\tilde{u}=\bar{u}_i+a\xi+b\eta+c(\xi^2-r_1)+d(\xi\eta-r_2)+e(\eta^2-r_3),\ r_1=\overline{\xi^2},\ r_2=\overline{\xi\eta},\ r_3=\overline{\eta^2},$$

where $\overline{x}$ refers to the cell average of x on element i. We also define $p_{i_k}=\Delta x_{i_k}/\Delta x_i$, $q_{i_k}=\Delta y_{i_k}/\Delta y_i$,

$$\boldsymbol{B}=\begin{pmatrix} \xi_{i_1} & p_{i_1} & 0 & \xi_{i_2} & p_{i_2} & 0 & \xi_{i_3} & p_{i_3} & 0\\ \eta_{i_1} & 0 & q_{i_1} & \eta_{i_2} & 0 & q_{i_2} & \eta_{i_3} & 0 & q_{i_3}\\ \xi_{i_1}^2-r_1 & 2p_{i_1}\xi_{i_1} & 0 & \xi_{i_2}^2-r_1 & 2p_{i_2}\xi_{i_2} & 0 & \xi_{i_3}^2-r_1 & 2p_{i_3}\xi_{i_3} & 0\\ \xi_{i_1}\eta_{i_1}-r_2 & p_{i_1}\eta_{i_1} & q_{i_1}\xi_{i_1} & \xi_{i_2}\eta_{i_2}-r_2 & p_{i_2}\eta_{i_2} & q_{i_2}\xi_{i_2} & \xi_{i_3}\eta_{i_3}-r_2 & p_{i_3}\eta_{i_3} & q_{i_3}\xi_{i_3}\\ \eta_{i_1}^2-r_3 & 0 & 2q_{i_1}\eta_{i_1} & \eta_{i_2}^2-r_3 & 0 & 2q_{i_2}\eta_{i_2} & \eta_{i_3}^2-r_3 & 0 & 2q_{i_3}\eta_{i_3} \end{pmatrix},$$

and $\boldsymbol{a}=(a,b,c,d,e)^T$. We thus arrive at the same linear system expressed in equation (39), which can be used to extract the solution for $\boldsymbol{a}$.

4.3.2. Updating ODEs

At this order, the left and right approximations of the Gaussian points may be calculated from

$$u_g^l=\bar{u}_i+a_i\xi_{g,i}+b_i\eta_{g,i}+c_i(\xi_{g,i}^2-r_{1,i})+d_i(\xi_{g,i}\eta_{g,i}-r_{2,i})+e_i(\eta_{g,i}-r_{3,i}),$$
$$u_g^r=\bar{u}_j+a_j\xi_{g,j}+b_j\eta_{g,j}+c_j(\xi_{g,j}^2-r_{1,j})+d_j(\xi_{g,j}\eta_{g,j}-r_{2,j})+e_j(\eta_{g,j}-r_{3,j}),$$

where $\xi_{g,i}=(x_g-x_{c_i})/\Delta x_i$, $\eta_{g,i}=(y_g-y_{c_i})/\Delta y_i$, $\xi_{g,j}=(x_g-x_{c_j})/\Delta x_j$, and $\eta_{g,j}=(y_g-y_{c_j})/\Delta y_j$.

While the updating ODEs remain precisely the same as in $\mathrm{P}_1\mathrm{FV}_1$, we now gain the alternative choice of using $\tilde{\delta}_2(\boldsymbol{x},\boldsymbol{x}_p)$, $\tilde{\delta}'_{2,x}(\boldsymbol{x},\boldsymbol{x}_p)$, and $\tilde{\delta}'_{2,y}(\boldsymbol{x},\boldsymbol{x}_p)$ to replace $\tilde{\delta}_1(\boldsymbol{x},\boldsymbol{x}_p)$, $\tilde{\delta}'_{1,x}(\boldsymbol{x},\boldsymbol{x}_p)$, and $\tilde{\delta}'_{1,y}(\boldsymbol{x},\boldsymbol{x}_p)$ for the nodal evolution. Nonetheless, our numerical experiments, which are detailed in forthcoming sections, show that when the second-order ADF is used, the scheme can become unstable. To avoid this situation, it is possible to replace the second-order ADF by the first-order ADF in $\mathrm{P}_1\mathrm{FV}_2$ in a manner to preserve the scheme's targeted third-order accuracy. Although an appropriate choice of four auxiliary coefficients in ADF's $\tilde{\delta}_{2|3}(\boldsymbol{x},\boldsymbol{x}_p)$ is likely to overcome the stability issue, work in this direction is beyond the scope of this study.

At this juncture, it may be instructive to note that the extension of the foregoing analysis of the scalar conservation equation to Euler's equation can be achieved straightforwardly by replacing the scalar flux computation with the Riemann flux vector for Euler's equation in both interfacial and domain integrals.

4.4. Curved element treatment

For problems involving curved boundaries, it is sometimes necessary to capture the curved edges as precisely as possible to achieve the desired accuracy while maintaining the stability of the high-order scheme. In such a situation, a uniformly second-order mesh may be used, where all of the elements in the mesh are transformed to standard linear elements using a second-order polynomial transformation, as shown in Fig. 4. The (ζ,ψ) variables featured here represent the transformed coordinates.

4.5. Temporal evolution scheme

In all of the upcoming numerical experiments, the treatment of time evolution is implemented as follows: for steady problems, we find it sufficient to use a second-order TVD Runge-Kutta scheme. However, for unsteady problems, we find it necessary to employ a third-order TVD Runge-Kutta time marching scheme.

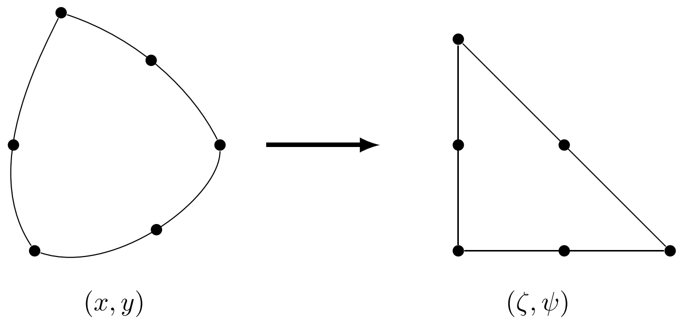


Fig. 4. Using a P2 polynomial to transform a curved triangular element to a standard linear element.

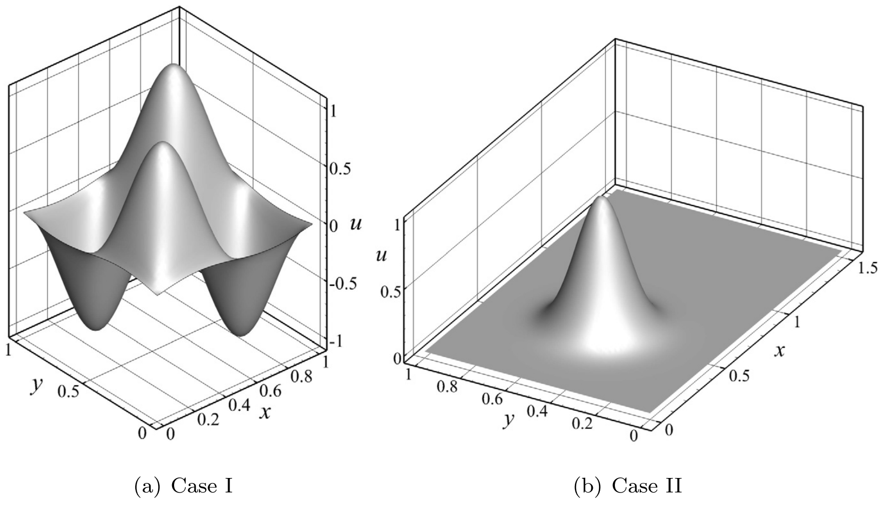


Fig. 5. Solution contours for $u(x, y, 0)$ corresponding to the initial spatial distributions for cases I (sinusoid) and II (Gaussian bump).

5. Numerical verification

In order to test the accuracy and stability of the PFV scheme, two benchmark cases are considered: (1) the linear wave equation and (2) Euler's nonlinear equation. These fundamental equations are often considered and discussed in textbooks on computational physics (e.g., Landau et al. [26]). To simplify the underlying notation, we use $\mathrm{P}_m\mathrm{FV}_n$-$\delta_k$ to notate the scheme where mth-order polynomial coefficients are recorded on the SP, an nth-order polynomial approximation is used to reconstruct the solution in each element, and the updating ODEs for SP DOFs are derived using kth-order ADF integration.

5.1. Verification based on the two-dimensional wave equation

The accuracy and stability of the PFV scheme at different orders is first tested in conjunction with the two-dimensional linear wave equation

$$\frac{\partial u}{\partial t} + a_x\frac{\partial u}{\partial x} + a_y\frac{\partial u}{\partial y} = 0; \quad a_x = 1,\ a_y = 0 \text{ and } u(x, y, t = 0) = u_0(x, y). \tag{42}$$

A classical solution to equation (42) is given by $u(x, y, t) = u_0(x - a_x t, y - a_y t)$. To make further headway, we explore two cases of initial and boundary conditions that are illustrated in Fig. 5. These correspond to:
(I) $u_0 = \sin(2\pi x)\sin(2\pi y)$ with $(x, y) \in [0, 1] \times [0, 1]$ and periodic boundary conditions in both x- and y-directions. The ensuing computations are carried out over one period of time up to $T = 1$.
(II) $u_0 = e^{-50\left[(x-0.5)^2+(y-0.5)^2\right]}$ and $(x, y) \in [0, 1.5] \times [0, 1]$. Horizontally, the values on left and right boundaries of this Gaussian bump, $x = 0$ and $x = 1.5$, are set equal to 0. Vertically, at $y = 0$ and $y = 1$, we set $\frac{\partial u}{\partial y} = 0$. Here the computations are performed up to $T = 0.5$ in order to keep the main distribution confined within the computational domain.

Note that the geometric configuration and spatial distributions of the initial conditions chosen for case II minimize the effects of edge treatment on the accuracy of the ADF formulation. Because the values are zero along the boundaries, the errors that can accumulate in conjunction with the use of a boundary vertex SP, as depicted in Fig. 2(b), become inconsequential compared to the errors that accrue in the central portion of the domain.

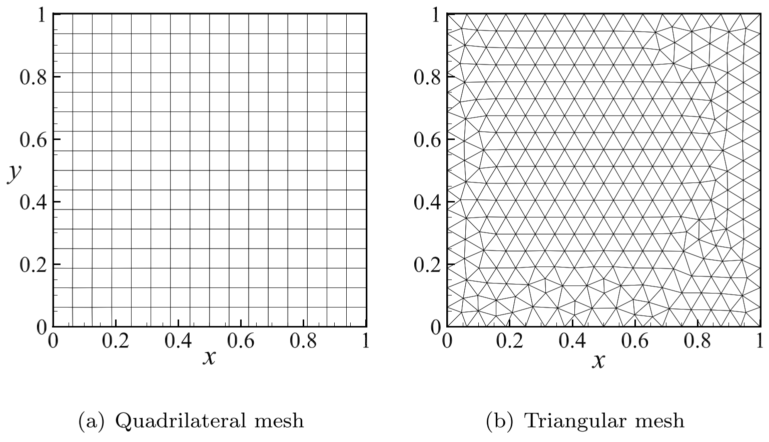


Fig. 6. Quadrilateral and triangular meshes used in the square computational domain of size $[0, 1] \times [0, 1]$.

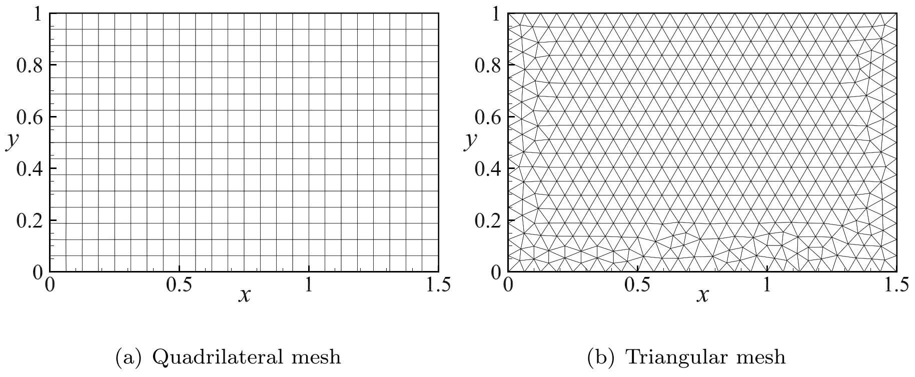


Fig. 7. Quadrilateral and triangular meshes used in the rectangular computational domain of size $[0, 1.5] \times [0, 1]$.

To demonstrate the convergence of the error, five successively refined meshes are used for both quadrilateral and triangular elements using cell sizes of $h = \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}$, and $\frac{1}{64}$. Meanwhile, Figs. 6 and 7 illustrate the meshes used for cases I and II, respectively.

In our effort to characterize the error convergence rate, the accuracy of the numerical simulation is measured using the standard L_2 functional error,

$$L_2 = \sqrt{\int_\Omega [u_{\text{num}}(x, y) - u_{\text{exact}}(x, y)]^2 \, \mathrm{d}V},$$

where u_{num} refers to the reconstructed numerical approximation and u_{exact} represents the theoretical solution given by

$$\begin{cases} u(x, y, t) = \sin[2\pi(x - t)] \sin(2\pi y) & \text{(case I)}, \\ u(x, y, t) = e^{-50\left[(x-t-0.5)^2 + (y-0.5)^2\right]} & \text{(case II)}. \end{cases}$$

The error convergence behavior of different order PFVs as well as their slopes for cases I and II is captured in Figs. 8 and 9, respectively. The convergence rates, which are specified on the legends by the graphical slopes, are calculated from the errors on grid sizes of $h = 1/16$ and $1/64$.

In exploring the behavior of the P_0FV_1 approximation, two different orders of ADF polynomials are tested, i.e., $\tilde{\delta}_0$ and $\tilde{\delta}_1$ in the updating of the DOFs on the SPs. The simulations are thus intended to investigate the influence of the different order ADFs on the accuracy of the scheme for a fixed cell reconstruction and DOF arrangement. Note that the P_1FV_2-δ_2 scheme is not examined here. It is found to be unstable and thus requiring a particularly dedicated treatment that must be deferred to a separate study.

For case I, the errors on the quadrilateral mesh show that, on the one hand, the P_0FV_1-δ_0 achieves second-order precision (with a slope of 2.12), and yet remains less accurate than P_0FV_1-δ_1 because of its larger L_2 error magnitude. On the other hand, P_0FV_1-δ_1 exhibits a lower convergence rate (with a slope of 1.83), and yet is accompanied by a lower L_2 error.

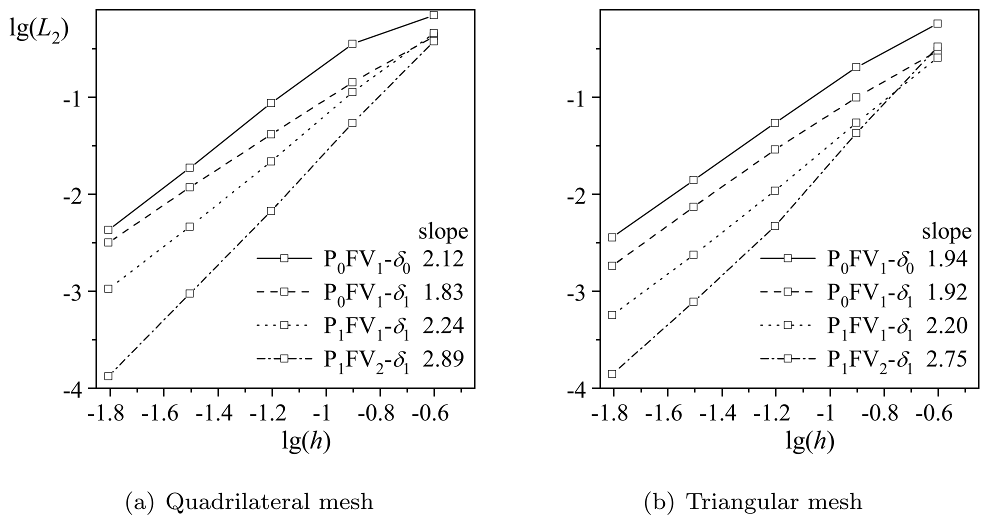


Fig. 8. Error convergence behavior of different order PFVs for the linear wave equation with case I simulated on a $[0, 1] \times [0, 1]$ spatial domain and a runtime interval of $T = 1$.

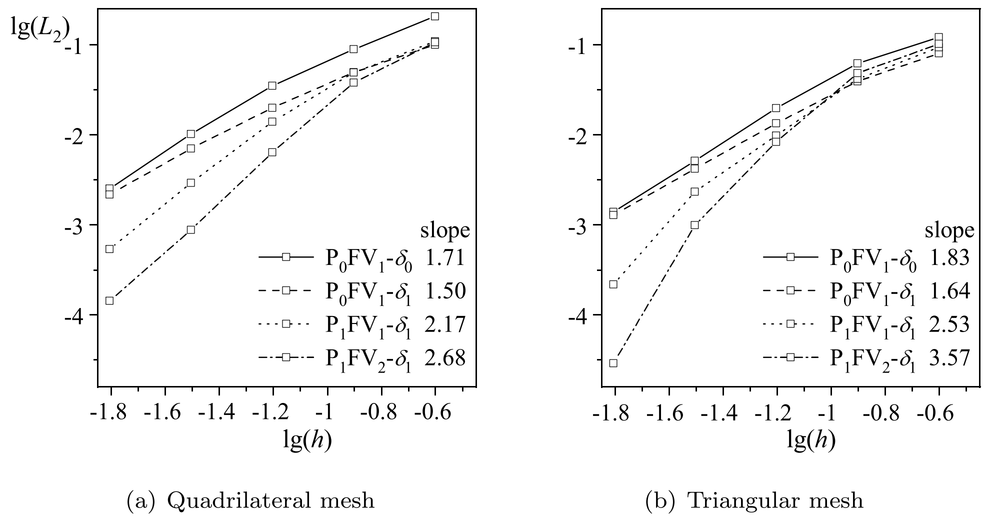


Fig. 9. Error convergence behavior of different order PFVs for the linear wave equation with case II simulated on a $[0, 1.5] \times [0, 1]$ spatial domain and a runtime interval of $T = 0.5$.

However, when we consider the behavior of these two schemes on a triangular mesh, both P_0FV_1-δ_0 and P_0FV_1-δ_1 produce similar convergence rates of 1.94 and 1.92 with P_0FV_1-δ_1 showing slightly better error performance.

As for the second-order P_1FV_1-δ_1 approximation, the scheme leads to slightly better than expected design order, namely, of 2.24 and 2.2 for the quadrilateral and triangular meshes, respectively. Conversely, the third-order P_1FV_2-δ_1 scheme exhibits slightly lower convergence rate than its design order on both quadrilateral and triangular meshes. It may be speculated that the slight decline in convergence rate in this case may be caused by the use of δ_1, instead of δ_2, for updating the DOFs on the SPs.

For the Gaussian bump distribution function of case II, similar convergence characteristics are observed. In this case, the error on the boundary appears to be trivial and negligible compared to the errors evolving in the central region of the solution domain. The most notable differences in the results include significant improvements in the P_1FV_1-δ_1 and P_1FV_2-δ_1 convergence rates using the triangular mesh, namely, 2.53 and 3.57, which exceed the corresponding design orders of 2 and 3, respectively.

Finally, in order to evaluate the robustness of the scheme in resolving the wave equation, we conservatively use case I on the quadrilateral mesh to test the stability of the different PFVs. Using the square mesh with a grid size of $h = 1/16$, our simulations are carried out to a very long time of $T = 200$. The evolution of the maximum value of u in the entire computational domain is subsequently used to evaluate the stability of the ongoing computations. In this process, the maximum stable CFLs are calculated and reported in Table 1.

Table 1
Maximum stable CFL computed at different PFV orders using the two ADF polynomials $\tilde{\delta}_0$ and $\tilde{\delta}_1$.

	P$_0$FV$_1$-δ_0	P$_0$FV$_1$-δ_1	P$_1$FV$_1$-δ_1	P$_1$FV$_2$-δ_1
CFL	1.20	0.67	0.70	0.71

Forthwith, it may be important to note the surprising stability characteristics displayed by the PFV scheme at increasing orders, where even a third-order scheme is capable of achieving a maximum CFL of 0.71. More specifically, the maximum stable CFL decreases only slightly (or does not decrease at all) with successive increases in the order. Such behavior makes this approach manifestly promising for later extensions to higher orders. It also stands in sharp contrast to the CFL performance associated with most compact high-order schemes, such as the DG method, where the maximum stable CFL decreases precipitously when the order is increased.

5.2. Verification based on the two-dimensional Euler equation

Euler's equation for unsteady compressible inviscid motion can be expressed in conservative form as

$$\frac{\partial \boldsymbol{U}}{\partial t}+\frac{\partial \boldsymbol{F}_j(\boldsymbol{U})}{\partial x_j}=0, \tag{43}$$

which is defined on $\boldsymbol{R}^d$, with d denoting the number of spatial dimensions, and where the conservative state vector $\boldsymbol{U}$ and flux vector $\boldsymbol{F}$ are specified as

$$\boldsymbol{U}=(\rho,\rho u_i,\rho e)^T,\quad \boldsymbol{F}_j=\left(\rho u_j,\rho u_i u_j+p\delta_{ij},u_j(\rho e+p)\right)^T;\quad i,j=1,2,\cdots,d. \tag{44}$$

In the above, the summation convention is used and ρ, p, and e denote the density, pressure, and specific total energy of the fluid, respectively; moreover, u_i refers to the velocity of the flow in the coordinate direction x_i. In conjunction with (43), the state relation for a perfect gas may be written as

$$p=(\gamma-1)\rho\left(e-\tfrac{1}{2}u_j u_j\right), \tag{45}$$

where γ is the ratio of the specific heats, which we take to be 1.4 unless specified differently.

In this work, the cell edge Riemann flux function is estimated using the Harten-Lax-van Leer Contact (HLLC) approximate Riemann flux [27], which has been successfully used to resolve compressible flows on unstructured grids [28] in both laminar and turbulent regimes [29–31].

5.2.1. Study based on the isentropic vortex propagation problem

In this section, the convection of the two-dimensional inviscid isentropic vortex, which is illustrated in Fig. 10, is used to test the accuracy of the PFV scheme. The analytical solution to this problem, at any time t, may be deduced from the advection of the initial motion, which consists of a linear superposition of a uniform flow, $\boldsymbol{U}_\infty=(\rho_\infty,U_\infty,V_\infty,p_\infty)=(1,1,0,1)$, and a perturbation of the velocity components, U and V, entropy S, and temperature T. These are given by

$$\begin{pmatrix}\tilde{U}\\ \tilde{V}\end{pmatrix}=\frac{\epsilon}{2\pi}e^{0.5(1-r^2)}\begin{pmatrix}y_0-y\\ x-x_0\end{pmatrix},\quad \tilde{S}=0,\quad \tilde{T}=\frac{(1-\gamma)\epsilon^2}{8\gamma\pi^2}e^{1-r^2}, \tag{46}$$

where $r^2=(x-x_0)^2+(y-y_0)^2$ and the (x_0,y_0) coordinates specify the vortex center. As for ϵ, it stands for the vortex strength. The initial values for this problem can be readily expressed as

$$\begin{gathered}\rho=(T_\infty+\tilde{T})^{\frac{1}{\gamma-1}},\quad \rho U=\rho(U_\infty+\tilde{U}),\quad \rho V=\rho(V_\infty+\tilde{V}),\\ e=\tfrac{p}{(\gamma-1)\rho}+\tfrac{1}{2}(U^2+V^2)\quad\text{and}\quad p=\rho^\gamma.\end{gathered} \tag{47}$$

For the sake of illustration, we set the vortex strength at $\epsilon=5$ and position the center of the vortex at $(5,5)$. A suitable computational domain Ω is chosen to extend over $(x,y)\in[0,15]\times[0,10]$. In this manner, a Dirichlet boundary condition of $\boldsymbol{U}=\boldsymbol{U}_\infty$ may be imposed at all four boundaries where the freestream velocity is recovered. Our computations are carried out up to time $T=5$ while the errors accrued in the results are measured by the L_2 error in the density calculation, namely,

$$L_2=||\rho_{\text{num}}(x,y)-\rho_{\text{exact}}(x,y)||_{L_2(\Omega)}=\sqrt{\int\limits_\Omega[\rho_{\text{num}}(x,y)-\rho_{\text{exact}}(x,y)]^2\,\mathrm{d}V}, \tag{48}$$

where $\rho_{\text{num}}(x,y)$ corresponds to the reconstructed numerical approximation and $\rho_{\text{exact}}(x,y)$ refers to the exact solution. Five successively refined meshes with grid sizes of $h=\frac{10}{4},\frac{10}{8},\frac{10}{16},\frac{10}{32}$, and $\frac{10}{64}$ are sequentially used to verify the error convergence order.

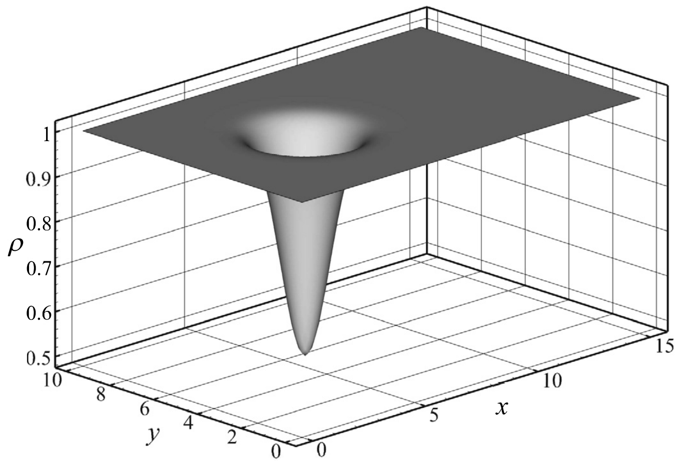


Fig. 10. Density contours of the initial distribution in the isentropic vortex propagation problem involving an ideal gas.

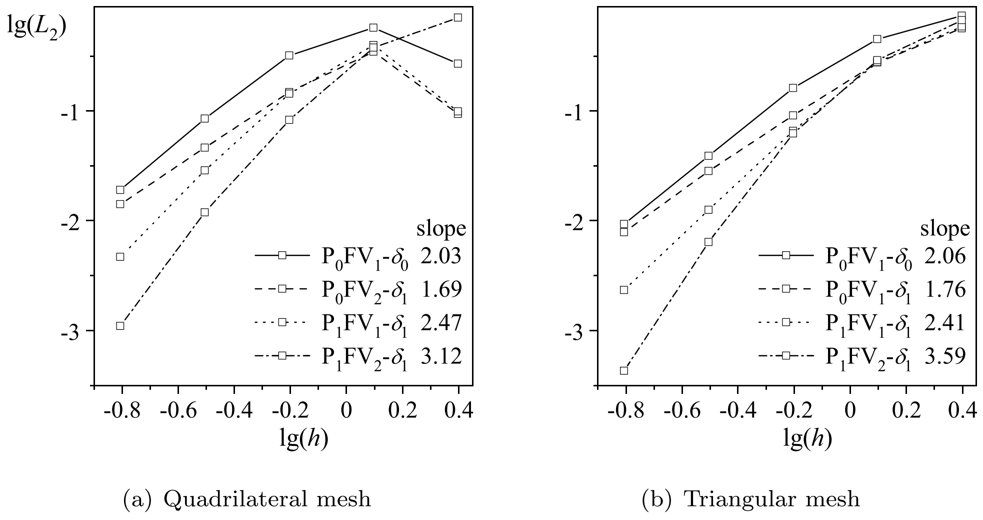


Fig. 11. Error convergence of different order PFVs for the isentropic vortex propagation problem simulated on a spatial domain of size $[0, 15] \times [0, 10]$ and a runtime interval of $T = 5$.

Fig. 11 displays the error convergence behavior of the PFV method at different orders on both quadrilateral and triangular meshes. The slopes are computed from the errors accrued using the last three meshes of $h = \frac{10}{16}$, $\frac{10}{32}$, and $\frac{10}{64}$.

A cursory examination of the error slopes helps to confirm that all of their design orders are well achieved with the exception of the P_0FV_1-δ_1 scheme. Having error slopes of 1.69 and 1.76 on the quadrilateral and triangular meshes, this scheme falls slightly short of its designated second-order convergence rate despite its lower error magnitude than P_0FV_1-δ_0.

Conversely, with slopes of 2.47 and 2.41 on quadrilateral and triangular meshes, we find P_1FV_1-δ_1 to be more accurate in both error magnitude and convergence rate than its P_0FV_1-δ_0 counterpart. It also exceeds its intended second order. The same may be said of the third-order P_1FV_2-δ_1, which converges more rapidly than its design order by achieving rates of 3.12 and 3.59 on the quadrilateral and triangular meshes, respectively.

5.2.2. Study based on the flow through a channel with a Gaussian bump

Let us now consider the internal flow problem in a channel with a height of 0.8 and a length of 3, thus requiring a computational domain that extends over $(x, y) \in [-1.5, 1.5] \times [0, 0.8]$. As shown in Fig. 12, the bump on the lower wall is prescribed by $y = 0.0625e^{-25x^2}$. On the left side of the domain, the inflow Mach number is taken to be 0.5 at a zero angle of attack. Moreover, the total pressure and temperature are imposed as inflow boundary conditions at the left boundary, while a constant static pressure is specified at the outflow boundary. Finally, the upper and lower walls are designated as slip walls.

In order to characterize the error response to grid refinement, two sets of unstructured second-order quadrilateral (9-point) and triangular (6-point) meshes, shown in Fig. 12, are selected and successively refined to coarse, medium, and fine

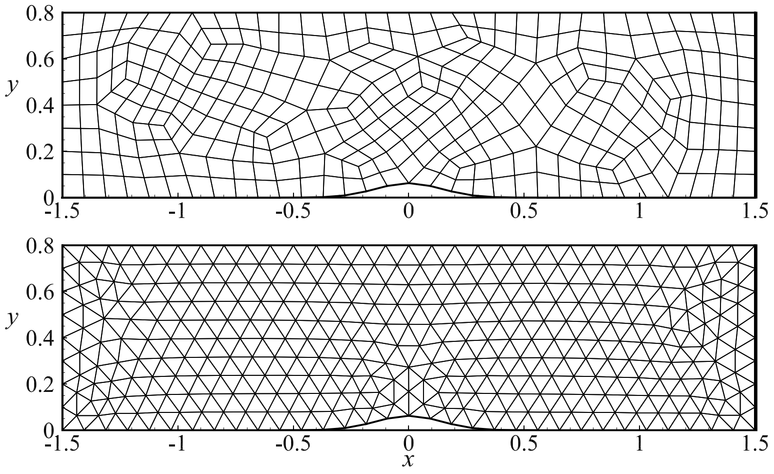
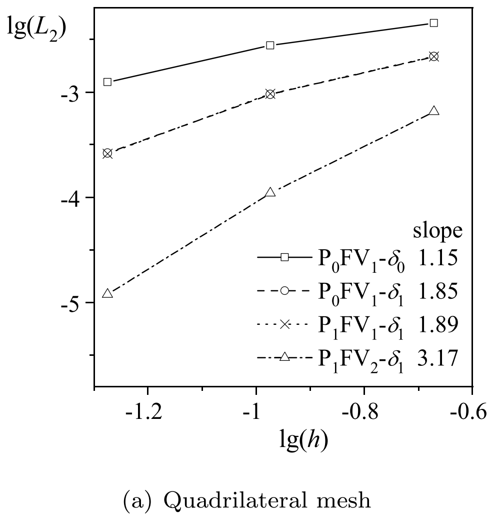


Fig. 12. Medium-size quadrilateral and triangular meshes used in the simulation of the Gaussian bump channel flow problem.



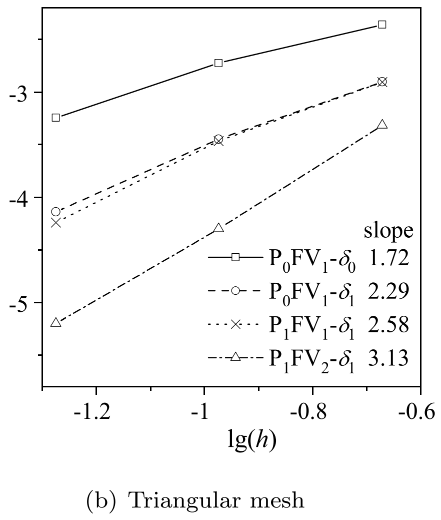


Fig. 13. Error convergence of different order PFVs for the Gaussian bump channel flow problem simulated on a spatial domain of size $[-1.5, 1.5] \times [0, 0.8]$, $M_\infty = 0.5$, and a suitably long runtime interval.

meshes, thus helping to validate the error convergence rate at different PFV orders. In this process, the error is measured using the L_2-norm of relative entropy increase, specifically

$$L_2 = \sqrt{\frac{1}{V}\int\limits_{\Omega}\left[\frac{p}{p_\infty}\left(\frac{\rho_\infty}{\rho}\right)^{\gamma} - 1\right]^2 \mathrm{d}V}. \tag{49}$$

Our simulations are run continuously until the L_2 error has converged to a constant value. Error results are then extracted and displayed in Fig. 13, where the convergence characteristics of different order PFVs are captured on quadrilateral and triangular meshes. The error slopes reported on the legends are calculated based on the errors corresponding to the medium and fine meshes.

Graphically, it may be seen that the error entailed in the P_0FV_1-δ_0 simulation continues to converge consistently with a second-order scheme, albeit slightly reduced to a slope of 1.15 on a quadrilateral mesh, and slightly accelerated to a slope of 1.72 on a triangular mesh. As for the errors in P_0FV_1-δ_1 and P_1FV_1-δ_1, they are found to be nearly indiscernible on the quadrilateral mesh, with convergence rates of 1.85 and 1.89, respectively. On the triangular mesh, P_1FV_1-δ_1 converges at a rate of 2.58, which is slightly faster than the 2.29 rate accrued by P_0FV_1-δ_1. As such, it may be safely stated that both schemes achieve their intended second order. As for the third-order P_1FV_2-δ_1 formulation, it appears to be substantially more accurate and faster converging on both quadrilateral and triangular meshes where its error decreases at a whopping 3.17 and 3.13 rates.

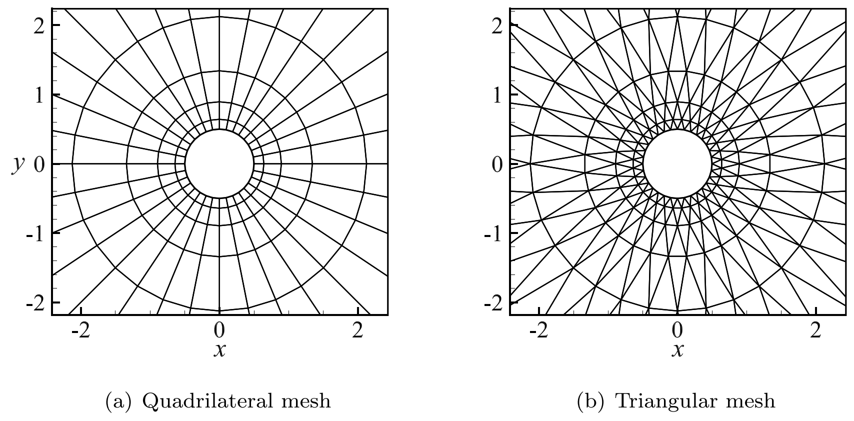


Fig. 14. Medium-size quadrilateral and triangular meshes used in the simulation of the flow past a cylinder problem.

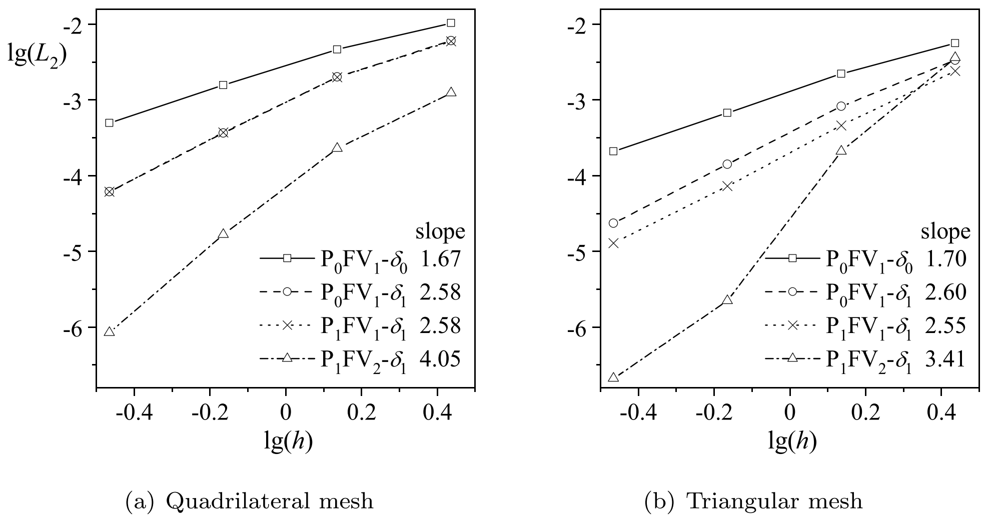


Fig. 15. Error convergence of different order PFVs for the flow past a cylinder problem simulated using $M_\infty = 0.38$.

5.2.3. Study based on the flow past a cylinder

This simulation considers the well-known test case of a subsonic flow past a circular cylinder at a Mach number of $M_\infty = 0.38$. A typical computational domain consists of a circular region having a radius of $R = 12$ that wraps around a smooth cylinder with a radius of $a = 0.5$. As usual, a Dirichlet boundary condition may be imposed along the outer circle representing the farfield boundary edge while a slip condition may be permitted along the circumferential wall of the cylinder.

In our effort to capture the curved wall accurately throughout the computational domain, we use quadratic quadrilateral (9-point) and triangular (6-point) meshes, which are successively refined to coarse, medium, fine, and very fine sizes. For the reader's convenience, the medium quadrilateral and triangular meshes are shown in Fig. 14.

As in the problem corresponding to an internal channel flow with a Gaussian bump, we measure the error using the L_2-norm of relative entropy increase given by (49). Subsequently, the simulations are carried out until such time when the residual has dropped to 1.0×10^{-8} and the L_2 error has converged to a constant value.

At this juncture, the convergence characteristics of different order PFVs may be extracted and illustrated in Fig. 15 using either quadrilateral or triangular meshes. We start with the performance of P_0FV_1-δ_0, which leads to convergence rates of 1.5 and 1.7 on the quadrilateral and triangular meshes, respectively. In both cases, the error falls slightly below its quadratic design order.

On the quadrilateral mesh, it is interesting that the errors attributed to P_0FV_1-δ_1 resemble those of the P_1FV_1-δ_1 so closely that their convergence rates of 2.58 cannot be graphically distinguished in Fig. 15(a). More importantly, we note that, irrespective of the mesh used, both P_1FV_1-δ_1 and P_1FV_2-δ_1 converge faster than their design orders of 2 and 3 by achieving rates of (2.58, 2.55) and (4.05, 3.41) using quads and triangles. In fact, P_1FV_2-δ_1 may be seen to super-converge to the fourth order on a quadrilateral mesh.

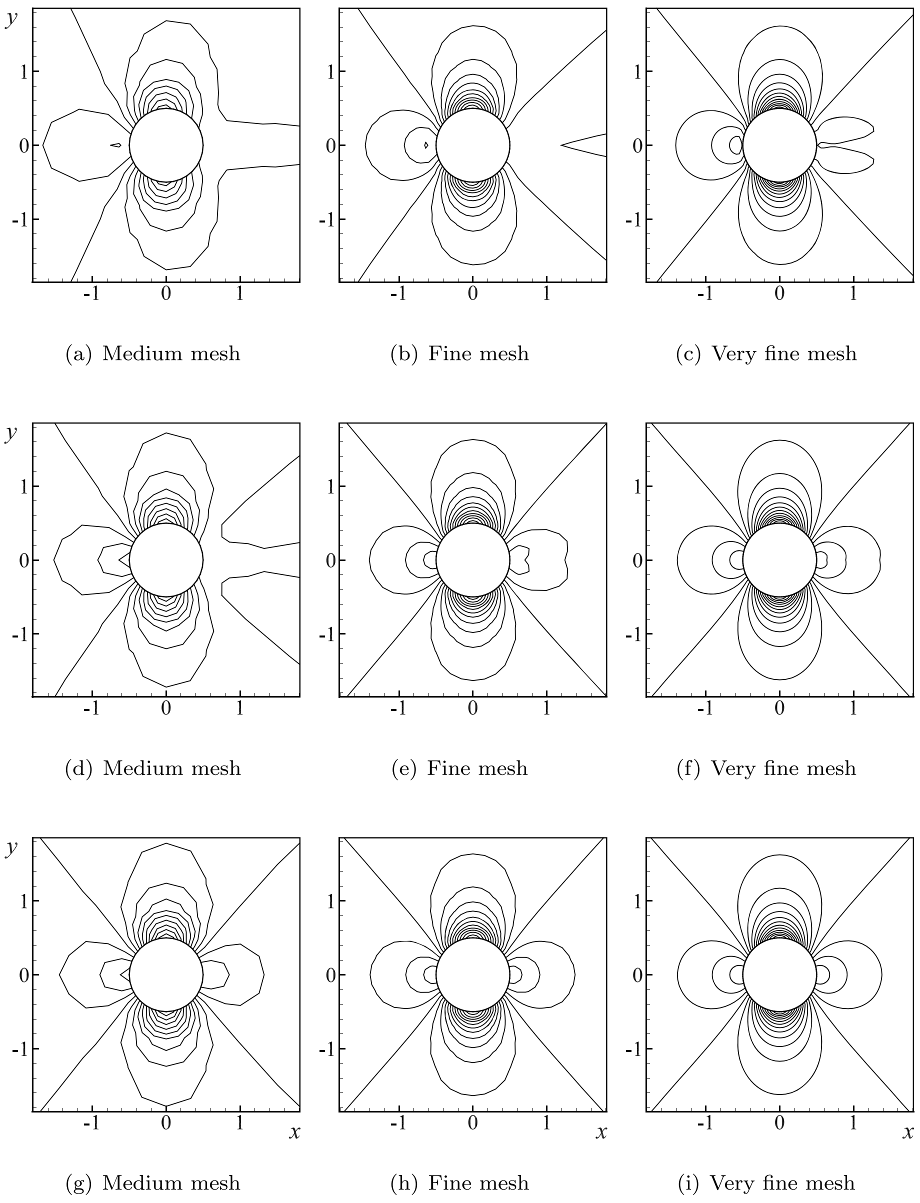


Fig. 16. Simulated density contours containing 16 levels between 0.76 and 1.06 on quadrilateral medium, fine, and very fine meshes using P_0FV_1-δ_0 (top), P_1FV_1-δ_1 (middle), and P_1FV_2-δ_1 (bottom) for the flow past a cylinder with $M_\infty = 0.38$.

Moving on to the triangular mesh results depicted in Fig. 15(b), we note that P_1FV_1-δ_1 remains more accurate than P_0FV_1-δ_1 despite its convergence rate of 2.55 being slightly lower than the 2.60 rate achieved by P_0FV_1-δ_1. Nonetheless, both schemes converge faster than their quadratic design order.

To better illustrate how the error reduction rate affects the quality of the simulation results, we use Figs. 16 and 17 to display the density contours predicted by P_0FV_1-δ_0, P_1FV_1-δ_1, and P_1FV_2-δ_1 on quadrilateral and triangular meshes, respectively. On quadrilateral meshes, both the error and density contours show that P_1FV_2-δ_1, which is characterized by a convergence rate of 4.05, is substantially more accurate than P_1FV_1-δ_1 on a finer mesh. This can be seen by contrasting the contours of Fig. 16(f) to those in (h), where the symmetry of the contours is already well established. Upon closer scrutiny, it may be graphically verified that using the high-order P_1FV_2-δ_1 method on a fine mesh in (h) leads to better symmetry and therefore more accuracy than using P_1FV_1 on a very fine mesh in (f). This observation is, in fact, corroborated by the error magnitudes provided in Fig. 15, i.e., where the second triangular symbol from the far left, which denotes P_1FV_2-δ_1 on a fine mesh, is noticeably lower than the cross symbol that appears at the far left, thus representing P_1FV_1 on a very fine mesh. The benefits of using a high-order method are thereby confirmed.

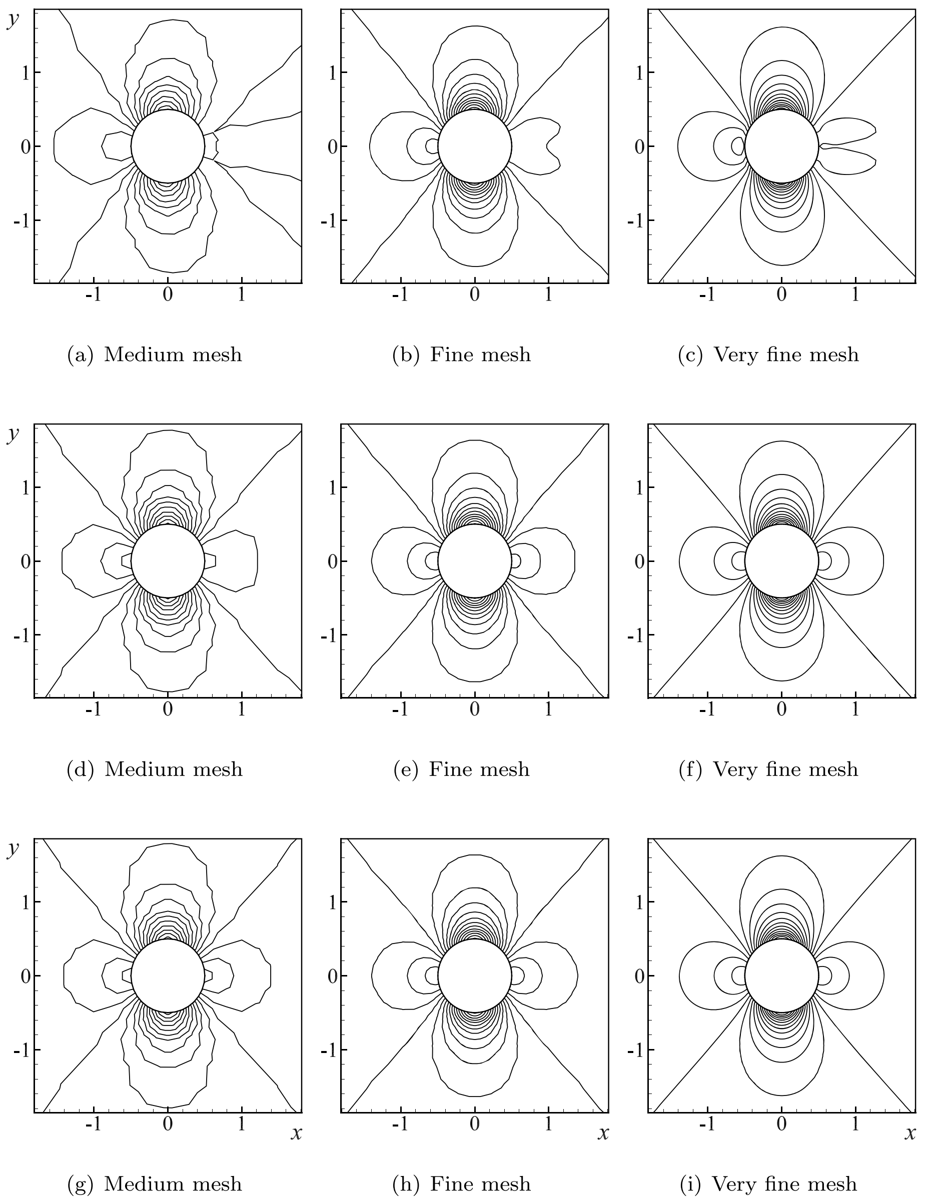
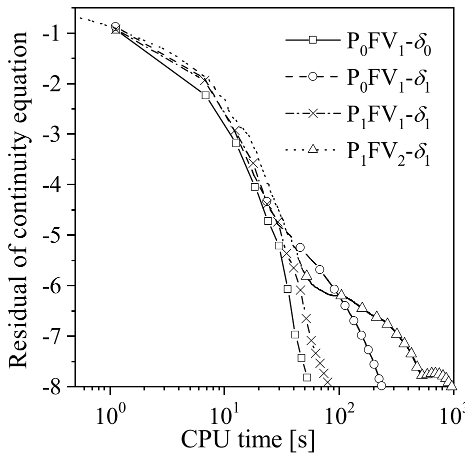


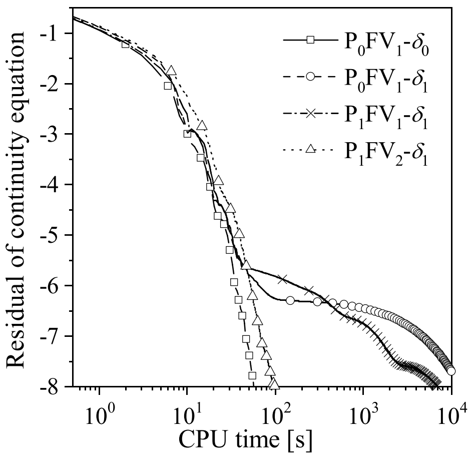
Fig. 17. Simulated density contours containing 16 levels between 0.76 and 1.06 on triangular medium, fine, and very fine meshes using P_0FV_1-δ_0 (top), P_1FV_1-δ_1 (middle), and P_1FV_2-δ_1 (bottom) for the flow past a cylinder with $M_\infty = 0.38$.

As we turn our attention to the simulation results on a triangular mesh in Fig. 17, similar trends may be observed. In this case, however, the error magnitude in the P_1FV_1 scheme remains substantially lower than its counterpart on a quadrilateral mesh, as per Fig. 15(b), despite its nearly identical convergence rate of 2.55. For this reason, the quality of the density contours in Fig. 17(e) on a fine mesh are visually equivalent to those in (h) using the higher-order P_1FV_2-δ_1 with a 3.41 convergence rate. A similar argument can be used to explain why the contours obtained using P_1FV_2-δ_1 in (g) on a medium mesh already exhibit a sufficient degree of symmetry despite their underlying coarseness. Although the 3.41 convergence rate of P_1FV_2-δ_1 on a triangular grid appears to be lower than its 4.05 counterpart on a quadrilateral grid, the actual error magnitudes entailed on a triangular grid remain much smaller in Fig. 15(b) for fine and very fine meshes. Here too, some of the advantages of using a high-order method on a triangular mesh are confirmed.

In order to compare the CPU cost of different orders of PFV schemes in solving the problem at hand, we use Fig. 18 to examine the convergence history of the relative residual of the continuity equation on a fine mesh; the latter is defined relative to the initially computed time-rate of change of the density, namely,



(a) Quadrilateral mesh



(b) Triangular mesh

Fig. 18. The residual convergence history of different orders of PFV schemes for flow past a cylinder computed on a fine mesh using both (a) quadrilateral and (b) triangular elements.

Table 2
CPU cost to perform 1000 steps by each of the PFV schemes while solving the flow past a cylinder on a fine mesh using both quadrilateral and triangular elements. N_e represents the total number of elements used.

	N_e	P_0FV_1-δ_0	P_0FV_1-δ_1	P_1FV_1-δ_1	P_1FV_2-δ_1
Quadrilateral mesh	1024	3.64 s	4.65 s	4.80 s	7.95 s
Triangular mesh	2048	3.99 s	5.68 s	6.54 s	11.16 s

$$Res_\rho = \frac{\sqrt{\int_\Omega R^2(\boldsymbol{x},t)\,\mathrm{d}V}}{\sqrt{\int_\Omega R^2(\boldsymbol{x},0)\,\mathrm{d}V}}, \tag{50}$$

where $R(\boldsymbol{x},t) = \partial\rho/\partial t$. Using both quadrilateral and triangular elements, it may be seen that the residuals associated with different PFV orders converge at comparable speeds until the residual reaches approximately $10^{-5.5}$, where the flow pattern becomes essentially well established. This behavior is quite unique and different from the typical convergence character of other techniques, such as the DG method, where the convergence rate has a tendency to decrease, often significantly, with successive increases in the design order. Using a quadrilateral mesh in part (a) of the graph, it must be noted that, after the residual error drops below the $10^{-5.5}$ value, the higher-order schemes begin to converge more slowly than their lower-order counterparts, except for P_0FV_1-δ_1. The latter tends to converge more slowly than P_1FV_1-δ_1, albeit less accurate.

For the triangular mesh, the convergence rate of the higher-order schemes follows a similar trend as the relative residual falls below the $10^{-5.5}$ mark in part (b) of the graph. This is true of all cases considered except for the third-order P_1FV_2-δ_1, which converges towards its final steady state more rapidly than both the second-order P_0FV_1-δ_1 and P_1FV_1-δ_1.

For added clarity, we use Table 2 to display the CPU times to perform 1000 steps by each of the foregoing PFV schemes. As one would expect, the CPU time increases as the order of the scheme or its complexity is increased. Naturally, for the same scheme, the CPU cost is lower on a quadrilateral mesh versus a triangular mesh, because the latter requires nearly twice the number of elements to cover the same computational domain. To further support these observations, we note that P_0FV_1-δ_0 has no domain integral because $\nabla\tilde{\delta}_0 = 0$, which makes it slightly faster-converging than P_0FV_1-δ_1.

Moving on to the second-order schemes, both P_0FV_1-δ_1 and P_1FV_1-δ_1 rely on the same number of Gaussian points. However, P_1FV_1-δ_1 requires two additional DOFs on the solution points, thus necessitating extra time for temporal matching. For this reason, the run times for P_1FV_1-δ_1 are slightly longer than those of P_0FV_1-δ_1 on both meshes. In the same vein, the third-order P_1FV_2-δ_1 is found to be about 66–70 percent more time consuming while being one order more precise than the second-order P_1FV_1-δ_1.

5.2.4. Study based on the simulated flow past a NACA0012 airfoil

The last test case involves an inviscid subsonic flow past a NACA0012 airfoil with a chord length of unity at a Mach number of $M_\infty = 0.63$ and an angle of attack of $\alpha = 2°$. The computational domain can be confined to a circular region of radius $R = 20$ and then discretized using a linear triangular mesh as shown in Fig. 19.

Different orders of PFVs are readily validated using this physical configuration, with all of the residuals being effectively suppressed below 10^{-10}. Along the surface of the airfoil, the pressure coefficient and relative entropy error are calculated from

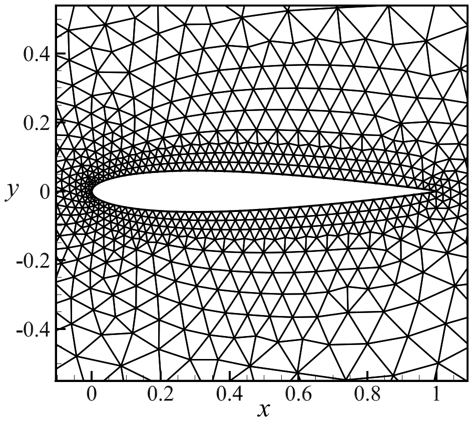


Fig. 19. Linear triangular mesh used for the simulation of the flow past a NACA0012 airfoil.

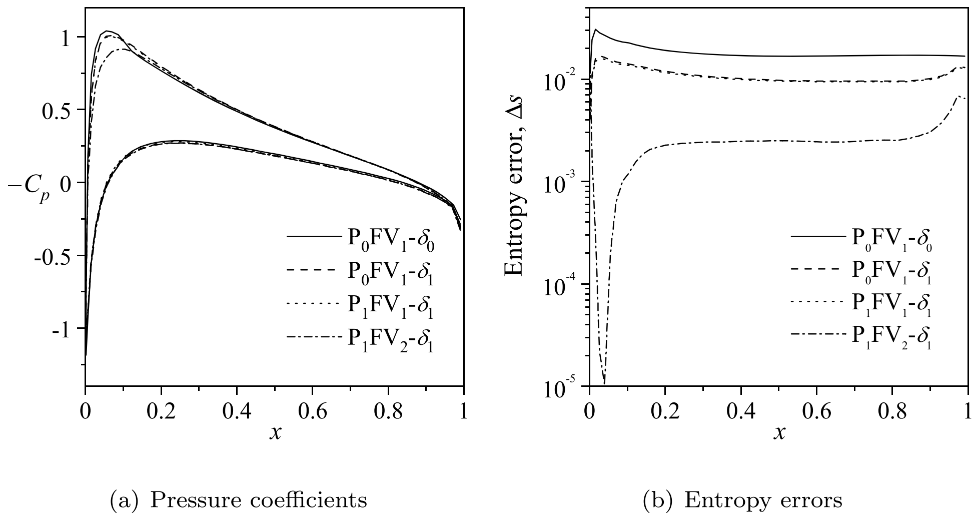


Fig. 20. Comparison of (a) pressure coefficients over the entire airfoil as well as (b) entropy errors over the top airfoil using different order PFVs for the subsonic flow past a NACA0012 airfoil at $M_\infty = 0.63$ and $\alpha = 2°$ angle of attack.

$$C_p = \frac{2(p - p_\infty)}{\rho_\infty U_\infty^2} \quad \text{and} \quad \Delta s = \frac{s - s_\infty}{s_\infty},$$

where $s = p\rho^{-\gamma}$ and $s_\infty = p_\infty \rho_\infty^{-\gamma}$. By way of comparison, the pressure coefficient computed over the entire surface of the airfoil as well as the entropy error, which is evaluated over the top surface of the airfoil, are shown in Fig. 20 at different PFV orders. Consistently with the previous benchmark cases, both P_0FV_1-δ_1 and P_1FV_1-δ_1 are seen to display similar degrees of precision, which are slightly higher than the predictive accuracy of P_0FV_1-δ_0. Unsurprisingly, P_1FV_2-δ_1 proves to be the most accurate scheme of those considered here.

Before leaving this section, we also compare the pressure and Mach number contours produced by P_1FV_1-δ_1 and P_1FV_2-δ_1 in Fig. 21. Here too, the Mach number contours corresponding to P_1FV_2-δ_1 are found to be the most precise.

6. On the reduced number of DOF requirements for the PFV method

One of the most distinctive features of the PFV method stands in its DOF setting, where the unknown value and possible derivatives are recorded at the solution point (SP) which, in turn, coincides with the vertex of an interior element. This DOF setting can naturally reduce the total number of DOFs and thus enhance the stability of the simulation by enabling the user to impose additional constraints on the continuity, for example, which has been well vetted in previous numerical simulation efforts.

To be more specific, the total number of degrees of freedom (NDOF) in the PFV method can be calculated from

$$\text{NDOF} = N_\text{e} + (N_\text{p} + N_\text{bf}) m_\text{p}, \tag{51}$$

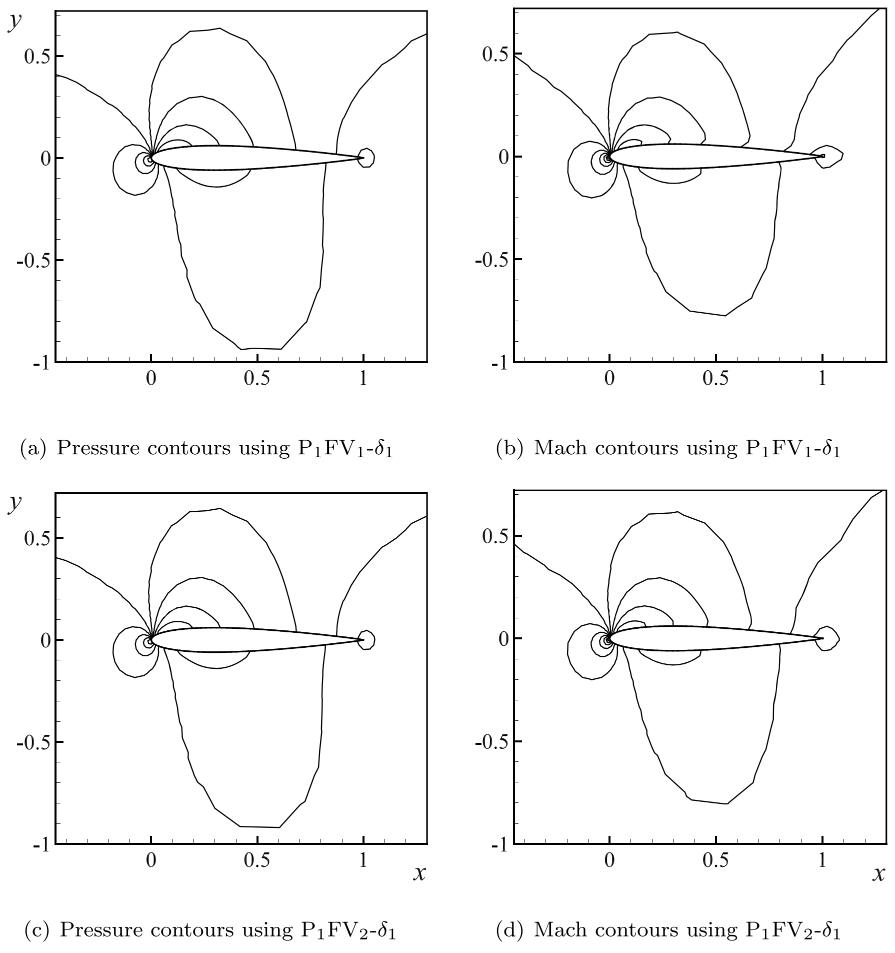


Fig. 21. Comparison of (a, c) pressure and (b, d) Mach number contours using (a, b) P_1FV_1-δ_1 (top) and (c, d) P_1FV_2-δ_1 (bottom) for the subsonic flow past a NACA0012 airfoil at $M_\infty = 0.63$ and $\alpha = 2°$ angle of attack. Here the pressure contours contain 10 levels between 1.4 and 2.3 while the Mach contours contain 13 levels between 0.1 and 0.9.

where N_e represents the number of elements, N_p denotes the number of vertices, N_{bf} stands for the number of boundary faces, and m_p refers to the number of DOFs stored on each SP. As for the effective NDOF contribution, which takes into account the order of the local polynomial reconstruction in each element (and thus determines the order of scheme), it can be determined from

$$n_{eff} = 1 + n_{vert} m_p, \tag{52}$$

where n_{vert} corresponds to the number of vertices in each element. For a boundary element, we have a slightly modified count of

$$n_{eff} = 1 + (n_{vert} + n_{bf}) m_p, \tag{53}$$

where n_{bf} specifies the number of boundary faces in a boundary element.

Let us now compare the NDOF associated with the PFV and that of the Discontinuous Galerkin (DG) method in a two-dimensional setting. For a triangular mesh with a large number of elements, we recall the well-known reduction in the number of vertices, namely, $N_p \approx N_e/2$. Using second and third-order PFV and DG schemes, the NDOF and N_e may be evaluated on either triangular or quadrilateral meshes. In the interest of clarity, the results are cataloged in Table 3.

Using a triangular mesh, the comparison is compelling: The PFV method saves half or more of the total NDOF required by the DG method of the same order. In fact, the effective NDOF on each element in P_1FV_2 can be readily calculated to be $1 + 3 \times 3 = 10$, which is sufficient to construct a third-order polynomial capable of achieving a fourth-order scheme. However, it also leads to a less sparse Jacobian matrix, which will be further discussed below.

For a quadrilateral mesh, we have $N_p \approx N_e$. The results of Table 3 show that the PFV method saves 1/3 of the NDOF required by the DG method of the same order. Moreover, the effective NDOF on each element in P_1FV_2 can be determined to be $1 + 4 \times 3 = 13$, which is sufficient to construct a third-order polynomial capable of achieving a fourth-order scheme.

Table 3
Comparison of the NDOF and effective NDOF for each element n_{eff} of the DG and PFV methods for two-dimensional triangular and quadrilateral meshes.

	Triangular				Quadrilateral			
	2nd-order		3rd-order		2nd-order		3rd-order	
	DGp$_1$	P$_0$FV$_1$	DGp$_2$	P$_1$FV$_2$	DGp$_1$	P$_0$FV$_1$	DGp$_2$	P$_1$FV$_2$
NDOF	$3N_e$	$1.5N_e$	$6N_e$	$2.5N_e$	$3N_e$	$2N_e$	$6N_e$	$4N_e$
n_{eff}	3	4	6	10	3	5	6	13

Table 4
Comparison of the NDOF and effective NDOF for each element n_{eff} of the DG and PFV methods for two-dimensional triangular and quadrilateral meshes at higher orders.

	Triangular				Quadrilateral			
	4th-order		5th-order		4th-order		5th-order	
	DGp$_3$	P$_1$FV$_3$	DGp$_4$	P$_2$FV$_4$	DGp$_3$	P$_1$FV$_3$	DGp$_4$	P$_2$FV$_4$
NDOF	$10N_e$	$2.5N_e$	$15N_e$	$4N_e$	$10N_e$	$4N_e$	$15N_e$	$7N_e$
n_{eff}	10	10	15	19	10	13	15	25

Speculatively speaking, the effectiveness of the PFV framework and the computational advantages that it offers relative to other compact methods are expected to only improve at higher orders, as projected in Table 4. The extension of the present framework to orders higher than 3 will therefore require additional work.

To achieve a fourth-order DG scheme, for example, the total NDOF that is required jumps to $10N_e$. In the PFV framework, only $2.5N_e$ NDOFs are needed to achieve the same order. Should one further explore the NDOF requirements to achieve a fifth order, the ability of the PFV design to outperform its DG counterpart becomes even more visible (Table 4). While the DG requirement increases to $15N_e$, the PFV is capable of securing the same fifth order with only $4N_e$, namely, with 73 percent fewer NDOFs.

7. Conclusion

This work extends the concept of a polynomial-based approximate delta function (ADF) to multiple spatial dimensions while providing the detailed formulation of the corresponding two-dimensional ADF polynomials at the first and second orders. Then the multi-dimensional ADF concept is judiciously adapted in the construction and design of a point-value enhanced finite volume (PFV) method, which stores and updates the cell-averaged values along with the values and derivatives of the unknown quantities at ideally located solution points (SPs). Away from the boundaries, the SPs are taken at the vertices of each interior element. For boundary elements, the central points of the boundary faces are specified as the SPs. Most importantly, the updating of the additional DOFs at the SPs takes advantage of the ADF properties, which enable us to leverage the information provided on all of the elements surrounding the SP as part of the integration domain. As for the updating of the cell-averaged values, we simply mimic the manner by which it is performed in the finite volume method and thus ensure the conservation of the scheme.

In each element of the PFV framework, we recall that the unknown quantities are reconstructed using the entire body of information available on the element and its vertices. Because nodal information can be effectively communicated with all of the surrounding elements, the PFV scheme is shown to be efficient in saving the number of DOFs compared to other compact methods that are developed to the same order. On a triangular mesh, for example, at least half of the DOFs are saved compared to the DG method of the same order and, through preliminary studies, the savings are projected to further improve at increasing orders. Moreover, in addition to the preservation of continuity, which is enhanced by the sharing of nodal information, the updating of nodal quantities on multiple elements leads to a markedly more stable algorithm. In the benchmark cases presented here, such as the linear wave equation, the CFL values are found to be at least three times larger than those entailed in the DG method of the same order. Not only are the CFL values relatively large, they only decrease slightly or do not decrease at all with successive increases in the design order. As such, it is speculated that the PFV technique can be extended to high-order numerical schemes while improving their stability and accuracy.

In this work, detailed descriptions and formulations of P$_0$FV$_1$, P$_1$FV$_1$, and P$_1$FV$_2$ are systematically carried out chiefly on triangular and quadrilateral grids with several mesh sizes that range from medium to very fine. Throughout this process, the error convergence rates associated with these methods are rigorously verified.

To be more specific, the PFV schemes and their convergence rates are verified using a suite of benchmark problems ranging from the linear wave equation to the nonlinear Euler equation while revisiting six sub-problems. Test cases considered include harmonically varying functions, internal and external channel flows with Gaussian bumps, isentropic vortex propagation, and the subsonic motion past either a cylinder and a NACA0012 airfoil. It is gratifying that the ensuing numerical experiments readily confirm the stability and accuracy of the PFV schemes up to an effective fourth order, which is realized on a quadrilateral mesh in the case of flow past a cylinder.

Although the PFV method shows favorable stability and DOF saving characteristics in this two-dimensional exploratory study, several tasks lie ahead, as the need to address some unresolved questions lingers. Among those open questions are (a) the choices between piecewise and continuous ADF polynomials, (b) the most effective boundary solution points needed to stabilize higher-order schemes, and (c) the optimal strategy to extend the PFV framework to viscous problems, where derivatives of the unknown functions must be considered in the evaluation of fluxes. This can be especially important in the implementation of implicit time matching, where the Jacobian matrix in the PFV scheme becomes less sparse than its DG counterpart. Until such studies are performed, however, the question of whether the DOF savings in the PFV scheme will outweigh the incurred costs relative to other compact methods will remain speculative at best. To overcome this deficiency, a Jacobian-free approach will need to be implemented in conjunction with an implicit iterative formulation of the PFV method to insure a favorable cost-benefit outcome. Efforts in this direction are presently underway.

In addition to the efforts to address these open questions, we hope that the extension of the two-dimensional PFV method to the solution of the viscous Navier–Stokes equations and to three-dimensional settings will form the basis of further inquiry.

CRediT authorship contribution statement

Li-Jun Xuan: Conceptualization, Methodology, Software, Data curation, Typesetting - Original draft preparation, Visualization, Investigation, Validation. **Joseph Majdalani**: Writing, Visualization, Reviewing and Editing, Interpretation, Validation, Supervision, Funding.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported partly by the National Science Foundation, through grant no. 1761675, and partly by the Hugh and Loeda Francis Chair of Excellence, Department of Aerospace Engineering, Auburn University. The authors thank Dr. Xiaodong Liu at the Los Alamos National Laboratory for his insightful comments and suggestions.

References

[1] H.T. Huynh, High-order methods including discontinuous Galerkin by reconstructions on triangular meshes, AIAA Paper 2011-44, in: 49th AIAA Aerospace Sciences Meeting, Orlando, FL, 2011.
[2] H.T. Huynh, On Formulations of Discontinuous Galerkin and Related Methods for Conservation Laws, Tech. Rep. NASA/TM—2014-218135, NASA Glenn Research Center, Cleveland, OH, 2014.
[3] L.-J. Xuan, J. Majdalani, A point-value enhanced finite volume method based on approximate delta function, J. Comput. Phys. 355 (15) (2018) 37–58, https://doi.org/10.1016/j.jcp.2017.10.059.
[4] W.H. Reed, T.R. Hill, Triangular mesh methods for the neutron transport equation, Tech. Rep. LA-UR-73, in: Los Alamos Scientific Laboratory Report, Los Alamos, NM, 1973.
[5] H.T. Huynh, A flux reconstruction approach to high-order schemes including discontinuous Galerkin methods, AIAA Paper 2007-4079, in: 18th AIAA Computational Fluid Dynamics Conference, Miami, FL, 2007.
[6] H.T. Huynh, A reconstruction approach to high-order schemes including discontinuous Galerkin for diffusion, AIAA Paper 2009-403, in: 47th AIAA Aerospace Sciences Meeting, Orlando, FL, 2009, https://doi.org/10.2514/6.2009-403.
[7] P. LaSaint, P.A. Raviart, On a Finite Element Method for Solving the Neutron Transport Equation. Mathematical Aspects of Finite Elements in Partial Differential Equations, Academic Press, 1974, pp. 89–145.
[8] B. Cockburn, G. Karniadakis, C.W. Shu (Eds.), Discontinuous Galerkin Methods: Theory, Computation, and Application, Springer, 2000.
[9] B. Cockburn, C.-W. Shu, The local discontinuous Galerkin methods for time dependent convection diffusion systems, SIAM J. Numer. Anal. 35 (6) (1998) 2440–2463, https://doi.org/10.1137/S0036142997316712.
[10] C.-W. Shu, Discontinuous Galerkin Method for Time Dependent Problems: Survey and Recent Developments, Springer, 2012.
[11] F. Bassi, S. Rebay, A high-order accurate discontinuous finite element method for the numerical solution of the compressible Navier–Stokes equations, J. Comput. Phys. 131 (2) (1997) 267–279, https://doi.org/10.1006/jcph.1996.5572.
[12] F. Bassi, S. Rebay, High-order accurate discontinuous finite element solution for the 2D Euler equations, J. Comput. Phys. 138 (2) (1997) 251–285, https://doi.org/10.1006/jcph.1997.5454.
[13] D.A. Kopriva, J.H. Kolias, A conservative staggered-grid Chebyshev multidomain method for compressible flows, J. Comput. Phys. 125 (1) (1996) 244–261, https://doi.org/10.1006/jcph.1996.0091.
[14] Y. Liu, M. Vinokur, Z.J. Wang, Discontinuous spectral difference method for conservation laws on unstructured grids, in: C. Groth, D.W. Zingg (Eds.), Computational Fluid Dynamics 2004, Springer, Berlin, Heidelberg, 2006, pp. 449–454.
[15] Z.J. Wang, Y. Liu, G. May, Spectral difference method for unstructured grids II: extension to the Euler equations, J. Sci. Comput. 32 (1) (2007) 45–71, https://doi.org/10.1007/s10915-006-9113-9.
[16] Z.J. Wang, L. Zhang, Y. Liu, Spectral (finite) volume method for conservation laws on unstructured grids IV: extension to two-dimensional Euler equations, J. Comput. Phys. 194 (2) (2004) 716–741, https://doi.org/10.1016/j.jcp.2003.09.012.
[17] Z.J. Wang, H. Gao, A unifying lifting collocation penalty formulation including the discontinuous Galerkin, spectral volume/difference methods for conservation laws on mixed grids, J. Comput. Phys. 228 (21) (2009) 8161–8186, https://doi.org/10.1016/j.jcp.2009.07.036.
[18] Z.J. Wang, H. Gao, A high-order lifting collocation penalty formulation for the Navier–Stokes equations on 2-D mixed grids, AIAA Paper 2009-3784, in: 19th AIAA Computational Fluid Dynamics Conference, San Antonio, TX, 2009.

[19] H. Gao, Z.J. Wang, H.T. Huynh, Differential formulation of discontinuous Galerkin and related methods for the Navier–Stokes equations, Commun. Comput. Phys. 13 (4) (2013) 1013–1044, https://doi.org/10.4208/cicp.020611.090312a.

[20] B. Xie, S. Li, A. Ikebata, F. Xiao, A multi-moment finite volume method for incompressible Navier–Stokes equations on unstructured grids: volume-average/point-value formulation, J. Comput. Phys. 277 (2014) 138–162, https://doi.org/10.1016/j.jcp.2014.08.011.

[21] T.A. Eymann, Active Flux Schemes, Ph.D. dissertation, Aerospace Engineering and Scientific Computing, University of Michigan, 2013.

[22] T.A. Eymann, P.L. Roe, Multidimensional active flux schemes, AIAA Paper 2013-2940, in: 21st AIAA Computational Fluid Dynamics Conference, San Diego, CA, 2013.

[23] H.T. Huynh, On high-order upwind methods for advection, AIAA Paper 2017-3093, in: 23rd AIAA Computational Fluid Dynamics Conference, Denver, CO, 2017.

[24] C.-W. Shu, S. Osher, Efficient implementation of essentially non-oscillatory shock-capturing schemes, J. Comput. Phys. 77 (2) (1988) 439–471, https://doi.org/10.1016/0021-9991(88)90177-5.

[25] J.J. Sakurai, J. Napolitano, Modern Quantum Mechanics, Cambridge University Press, 2017.

[26] R.H. Landau, M.J. Páez, C.C. Bordeianu, A Survey of Computational Physics: Introductory Computational Science, Princeton University Press, 2008.

[27] E. Toro, M. Spruce, W. Speares, Restoration of the contact surface in the HLL-Riemann solver, Shock Waves 4 (1) (1994) 25–34, https://doi.org/10.1007/BF01414629.

[28] X. Liu, L. Xuan, Y. Xia, H. Luo, A reconstructed discontinuous Galerkin method for the compressible Navier–Stokes equations on three-dimensional hybrid grids, Comput. Fluids 152 (2017) 217–230, https://doi.org/10.1016/j.compfluid.2017.04.027.

[29] X. Liu, Y. Xia, H. Luo, L. Xuan, A comparative study of Rosenbrock-type and implicit Runge-Kutta time integration for discontinuous Galerkin method for unsteady 3D compressible Navier-Stokes equations, Commun. Comput. Phys. 20 (4) (2016) 1016–1044, https://doi.org/10.4208/cicp.300715.140316a.

[30] X. Liu, Y. Xia, J. Cheng, H. Luo, Development and assessment of a reconstructed discontinuous Galerkin method for the compressible turbulent flows on hybrid grids, AIAA Paper 2016-1359, in: 54th AIAA Aerospace Sciences Meeting, San Diego, CA, 2016.

[31] C. Wang, J. Cheng, M. Berndt, N. Carlson, H. Luo, Application of nonlinear Krylov acceleration to a reconstructed discontinuous Galerkin method for compressible flows, AIAA Paper 2016-3964, in: 46h AIAA Fluid Dynamics Conference, Washington, DC, 2016.