De-Aliasing through Over-Integration Applied to the Flux Reconstruction and Discontinuous Galerkin Methods

Seth C. Spiegel,* H. T. Huynh,† and James R. DeBonis†
NASA Glenn Research Center, Cleveland, OH, 44135, USA

High-order methods are quickly becoming popular for turbulent flows as the amount of computer processing power increases. The flux reconstruction (FR) method presents a unifying framework for a wide class of high-order methods including discontinuous Galerkin (DG), Spectral Difference (SD), and Spectral Volume (SV). It offers a simple, efficient, and easy way to implement nodal-based methods that are derived via the differential form of the governing equations. Whereas high-order methods have enjoyed recent success, they have been known to introduce numerical instabilities due to polynomial aliasing when applied to under-resolved nonlinear problems. Aliasing errors have been extensively studied in reference to DG methods; however, their study regarding FR methods has mostly been limited to the selection of the nodal points used within each cell. Here, we extend some of the de-aliasing techniques used for DG methods, primarily over-integration, to the FR framework. Our results show that over-integration does remove aliasing errors but may not remove all instabilities caused by insufficient resolution (for FR as well as DG).

I. Introduction

Accurate and efficient numerical methods for the simulation of complex aerodynamic flows are desired to help reduce the design and development costs of aerodynamic vehicles and aeropropulsion systems. The primary challenge lies in the prediction of the turbulent motion within the flow and its effect on the overall motion of the fluid. Advanced methods such as large-eddy simulation (LES) and direct numerical simulation (DNS) offer great advancements in predicting turbulence, but these methods require significant improvements in accuracy and efficiency in order to become widely used. Focusing on either the accuracy or efficiency aspect of the underlying numerical algorithms, when developing a computational fluid dynamics (CFD) code, usually demands concessions from the other parameter.

Promising results have been found applying LES in the realm of aeropropulsion, but the spatial and temporal resolution needed for LES requires high-order (higher than second) numerical schemes historically only found in structured grid methods with large stencils. These methods have numerous drawbacks: difficulty in grid generation, high sensitivity to grid quality, complex boundary conditions, and poor scalability for parallel computing. These drawbacks severely limit the ability of these codes to compute complex aeropropulsion configurations, e.g., noise suppressing nozzles.[1]

In an attempt to alleviate the difficulties of structured grid generation for complex geometries, researchers have applied LES methods to unstructured grids.[2,3] However, the limited second-order accuracy of current unstructured methods requires an exorbitant number of grid points and offers little incentive over the traditional structured methods.[4,5] The flux reconstruction (FR) method[6–8] provides a simple, efficient, and easy to implement method for solving the Navier-Stokes equations on unstructured grids and is accurate to an arbitrary order. By employing the high-order FR method with LES on unstructured grids, limitations related to structured grid generation, grid resolution, and computational efficiency can be overcome.[9–12]

High-order spectral-/finite-element type methods, including discontinuous Galerkin (DG) and FR methods, use a finite set of basis functions to construct a numerical solution within each grid cell. When applied

*NASA Postdoctoral Fellow, Inlets and Nozzles Branch, 21000 Brookpark Road, and AIAA Member.
†Aerospace Engineer, Inlets and Nozzles Branch, 21000 Brookpark Road, and AIAA Associate Fellow.

to nonlinear equations, all of these methods can generate errors if any of the modes arising from the nonlinear terms are outside the span of the set of basis functions. In these cases, the energy from the unresolved (under-integrated) modes are aliased onto the lower modes producing so-called aliasing errors. The effect aliasing errors can have on a given simulation is highly unpredictable, and many researchers have shown they can produce severe instabilities within a problem.[13–21]

Aliasing errors have been extensively studied in reference to DG methods resulting in numerous stabilization techniques aimed at de-aliasing the numerical solution. One of the most commonly used techniques is a modal filter[16,18,22] which is used to add dissipation to the highest modes, thus reducing the magnitude of the oscillations within the polynomial. However, the common exponential filter contains three adjustable parameters and the amount of filtering required to remove any instabilities is highly dependent on the problem being solved. Therefore, filtering in general is not very user friendly. Other stabilization methods include but are not limited to: the spectral vanishing viscosity method[13,19,23–27] (artificial viscosity), superconsistent collocation,[28,29] and polynomial de-aliasing[13,14,18,19] which is also referred to as over-integration.

Regarding FR methods, the study of aliasing errors has mostly been limited to the selection of the nodal points used within each cell.[17,20,21,30–33] In this work, we look to extend some of the de-aliasing techniques used for DG methods, primarily over-integration, to the FR framework. In section II, we briefly review the FR method for the 1D conservation law. In section III, we derive the interpolation and projection operators and subsequently describe their use for over-integration within the FR method. Section IV describes the governing Euler equations. In section V, we apply over-integration to many different cases of the isentropic Euler vortex problem to see the effects it has on simulations of varying resolutions. Finally, conclusions and discussion are provided in section VI.

II. Numerical Methods

II.A. Spatial Discretization

We briefly review the FR method below. Consider the 1D conservation law

$$u_t + f_x = 0 \tag{1}$$

where the subscripts t and x denote partial differentiation with respect to time and space, respectively. Let the computational domain $\mathbf{\Omega}$ be divided into an arbitrary number of nonoverlapping cells. Each cell is transformed from physical space within the global domain to a common reference space/element, $\mathbf{I}$, which in 1D is the line interval, $\mathbf{I} = [-1, 1]$. With r denoting the local coordinate variable in the reference space, the mapping function $r = \chi(x)$ is used to complete this transformation. Similarly, the inverse function $x = \chi^{-1}(r)$ maps the reference element back to physical space. The details of such mappings, especially for higher dimensions, are beyond the scope of this paper; the interested reader can find additional information in many texts on spectral-/finite-element type methods, including but not limited to Ref. 15,16,19,34. The following review focuses on using the FR method to solve the transformed version of equation (1) within each reference element.

At a given time t, let the exact solution to the transformed conservation law within the reference element be given by $u_{\text{E}}(r,t)$ which may be a nonsmooth and/or discontinuous function. Let $u(r,t)$ be a polynomial of degree P and approximate $u_{\text{E}}(r,t)$ within the reference element. This approximate *solution polynomial*, $u(r,t)$, is local to each element, and is generally discontinuous across cell interfaces. With the dependence on time being understood, we will drop the variable t from $u_{\text{E}}(r,t)$ and $u(r,t)$ to simplify notation.

Let N_P refer to the minimum number of *solution points* needed to define the solution polynomial within the reference element, and let $\boldsymbol{\xi}$ denote the set of solution points, ξ_n, $n = 1, 2, \ldots, N_P$. Using tensor products to extend this to higher dimensions, the $\mathcal{N}$-dimensional conservation law requires $N_P = (P+1)^{\mathcal{N}}$ solution points within each computational cell.

The Lagrange polynomial is defined as the interpolating polynomial through a set of nodal points, for example $\boldsymbol{\xi}$, that takes the value 1 at ξ_n and zero at all other points $\xi_{i\neq n}$:

$$\ell_n(r) = \prod_{\substack{i=1 \\ i\neq n}}^{N_P} \frac{r - \xi_i}{\xi_n - \xi_i} \tag{2}$$

Note that the Lagrange polynomial in equation (2) is a polynomial of degree $P = N_P - 1$ and the set of Lagrange polynomials for all solution points in $\boldsymbol{\xi}$ is a set of N_P polynomials each of degree P.

A *nodal coefficient* of the solution, denoted by $\tilde{u}_n$, is simply the value of the approximate solution polynomial evaluated at the solution point, ξ_n, i.e., $\tilde{u}_n = u(\xi_n)$. Let $\tilde{\boldsymbol{u}}_P$ denote a vector containing the set of N_P nodal coefficients for the solution at each of the solution points. Assuming that all of $\tilde{\boldsymbol{u}}_P$ is known, we can use the Kronecker delta property of equation (2) to construct the approximate solution polynomial as

$$u(r) = \sum_{n=1}^{N_P} \tilde{u}_n \ell_n(r) \tag{3}$$

The above relation is why the set of Lagrange polynomials can also be referred to as a *nodal basis* set. In this section and section III, the tilde accent will be used exclusively to identify the nodal coefficients for a polynomial function.

The derivative of the solution polynomial is a linear combination of the derivatives of the polynomial basis functions

$$\frac{d}{dr}u(r) = u_r(r) = \sum_{n=1}^{N_P} \tilde{u}_n \left[\frac{d}{dr}\ell_n(r)\right] \tag{4}$$

Using the previous equation, we can define the derivative operator, $\boldsymbol{D}_r$, as the $N_P \times N_P$ matrix with elements

$$D_r[i,j] = \left[\frac{d}{dr}\ell_j(r)\right]\Big|_{\xi_i} \tag{5}$$

that transforms the nodal coefficients, $\tilde{\boldsymbol{u}}_P$, into the derivative nodal coefficients, $\tilde{\boldsymbol{u}}_r$. Note that this derivative is exact only when the function being differentiated is a polynomial of degree P or less.

If we use the nodal solution to evaluate the flux function, f, at each solution point, then interpolate these N_P values by a polynomial of degree P, we get what is called the *discontinuous flux function*. It only uses information inside a given cell and does not account for interaction between it and the data in the neighboring cells. Using the superscript 'D' for discontinuous, the discontinuous flux polynomial, $f^{\mathrm{D}}(r)$, is given by

$$f^{\mathrm{D}}(r) = \sum_{n=1}^{N_P} \tilde{f}_n \ell_n(r) \tag{6}$$

where the nodal coefficients of the discontinuous flux, $\tilde{f}_n$, are evaluated at the solution points using the nodal coefficients of the solution, i.e., $\tilde{f}_n = f(\tilde{u}_n)$. This collocation projection results in $f^{\mathrm{D}}(r)$ being a polynomial of degree P, which only matches the true flux function if the conservation law in equation (1) is the linear advection equation. For a nonlinear conservation law such as those found in the Euler equations or Navier-Stokes equations, the true flux function evaluated using the solution polynomial is of much higher degree than P and possibly even irrational, and $f^{\mathrm{D}}(r)$ above may not provide a good approximation. We will explore other ways of approximating the true flux function later.

With $f^{\mathrm{D}}(r)$ given by equation (6), we can compute the derivative of the approximated discontinuous flux using the derivative operator from equation (5). Denoting $\tilde{\boldsymbol{f}}$ as the column vector containing the coefficients $\tilde{f}_n$, $n = 1, 2, \ldots, N_P$, the derivative $\frac{d}{dr}\left[f^{\mathrm{D}}(r)\right]$ can be approximated using the matrix-vector product

$$\tilde{\boldsymbol{f}}_r = \boldsymbol{D}_r \tilde{\boldsymbol{f}} \tag{7}$$

where, $\tilde{\boldsymbol{f}}_r$ is the column vector containing the values of the approximate flux derivative at each solution point. Note that the collocation projection of the discontinuous flux in equation (6) is of degree P. Consequently, its derivative is of degree $P-1$.

The method for computing the divergence of the discontinuous flux given by equation (7) will be referred to as the Lagrange polynomial (LP) method since it utilizes the derivative of the Lagrange polynomials to evaluate f_r^{D}. An alternative method can be formulated by using the chain rule to rewrite equation (7) as

$$\tilde{\boldsymbol{f}}_r = \frac{\partial f}{\partial u}\boldsymbol{D}_r \tilde{\boldsymbol{u}}_P \tag{8}$$

Naturally, this will be referred to as the chain rule (CR) method and requires the evaluation of the flux Jacobian at each solution point, making it more expensive than the LP method. Whereas numerical experiments have shown that the CR method yields more accurate solutions over the LP method for nonlinear

problems,[35] the method itself is not fully conservative. A modification to the CR method was presented in Ref. 36 that ensures conservation.

If we employ either of the above methods for f_r^{D} to evaluate f_x for the conservation law, we obtain erroneous solutions since such a derivative does not include the interaction of the data between adjacent cells. Therefore, we seek to construct a so-called *continuous flux function*, denoted by $F(r)$, which accounts for this interaction between adjacent cells and approximates the discontinuous flux function in some sense, to which we can then calculate its derivative. The continuous flux function will be obtained by adding a correction to f^{D}, the discontinuous one.

To this end, at each (cell) interface, let u_{L} and u_{R} be the values taken on by the solution polynomials to its left and right, respectively. An upwind flux value common for the two adjacent cells, denoted by $f^* = f^*(u_{\mathrm{L}}, u_{\mathrm{R}})$, can then be defined via the wind direction. Here, for the case of the Euler equations, we use Roe's flux[37] with an entropy fix.[38]

Let f^*_{LB} denote the upwind flux at the 'left boundary' of the cell and f^*_{RB} denote the upwind flux at the 'right boundary' of the cell. We can then construct the continuous flux function, $F(r)$, as a polynomial of degree $P+1$ by requiring that it takes on the upwind flux values at the two cell interfaces, and it approximates f^{D} via $P-1$ conditions. Instead of defining F, we define $F - f^{\mathrm{D}}$, which approximates the zero function. At the two interfaces $r = \pm 1$, the function $F(r) - f^{\mathrm{D}}(r)$ takes on the following values, which are called the flux jumps:

$$F(-1) - f^{\mathrm{D}}(-1) = f^*_{\mathrm{LB}} - f^{\mathrm{D}}(-1) \qquad \text{and} \qquad F(1) - f^{\mathrm{D}}(1) = f^*_{\mathrm{RB}} - f^{\mathrm{D}}(1) \tag{9}$$

Therefore, $F(r) - f^{\mathrm{D}}(r)$ has prescribed left and right interface values, is of degree $P+1$, and approximates zero.

We now separate the prescription of the jump at the left interface from that of the right. This separation plays a critical role in the new approach. Again, using the local coordinate, let g_{LB} be the *correction function* for the left interface of the reference element defined by

$$g_{\mathrm{LB}}(-1) = 1, \qquad g_{\mathrm{LB}}(1) = 0 \tag{10}$$

and g_{LB} is a polynomial of degree $P+1$ approximating the zero function in some sense. We always choose g_{RB} by reflection: $g_{\mathrm{RB}}(r) = g_{\mathrm{LB}}(-r)$. As a result,

$$g_{\mathrm{RB}}(-1) = 0, \qquad g_{\mathrm{RB}}(1) = 1 \tag{11}$$

Consider the left interface $r = -1$. The polynomial

$$f^{\mathrm{D}}(r) + [f^*_{\mathrm{LB}} - f^{\mathrm{D}}(-1)]\, g_{\mathrm{LB}}(r) \tag{12}$$

provides a correction at the left interface for $f^{\mathrm{D}}(r)$ by changing the flux value at this interface from $f^{\mathrm{D}}(-1)$ to f^*_{LB}, while leaving the value at the right interface unchanged, namely, $f^{\mathrm{D}}(1)$. Next, the polynomial

$$F(r) = f^{\mathrm{D}}(r) + [f^*_{\mathrm{LB}} - f^{\mathrm{D}}(-1)]\, g_{\mathrm{LB}}(r) + [f^*_{\mathrm{RB}} - f^{\mathrm{D}}(1)]\, g_{\mathrm{RB}}(r) \tag{13}$$

provides corrections to both interfaces; using equations (10) and (11), one can easily verify that $F(-1) = f^*_{\mathrm{LB}}$ and $F(1) = f^*_{\mathrm{RB}}$. Thus, the above $F(r)$ is of degree $P+1$, takes on the upwind flux values at the two interfaces, and approximates $f^{\mathrm{D}}(r)$ in the same sense that g_{LB} and g_{RB} approximate the zero function.

The derivative of $F(r)$ at the solution point ξ_n is evaluated as

$$\left.\frac{\partial\, F(r)}{\partial r}\right|_{\xi_n} = F_r(\xi_n) = \tilde{f}_{r,n} + [f^*_{\mathrm{LB}} - f^{\mathrm{D}}(-1)]\, g'_{\mathrm{LB}}(\xi_n) + [f^*_{\mathrm{RB}} - f^{\mathrm{D}}(1)]\, g'_{\mathrm{RB}}(\xi_n) \tag{14}$$

Finally, F_r is transformed from reference space to physical space using the inverse mapping χ^{-1}, where it can then be employed to approximate f_x at each solution point within the physical grid cell. The solution to equation (1) is then advanced in time via an explicit Runge-Kutta method.

In order to complete this brief review of the FR method, we must define the correction function g_{LB}; the definition of g_{RB} follows by reflection as previously discussed. With L_k denoting the (standard) Legendre polynomial of degree k, the right Radau polynomial of degree k (with $k > 0$) is defined by

$$R_{R,k} = \frac{(-1)^k}{2}\,(L_k - L_{k-1}) \tag{15}$$

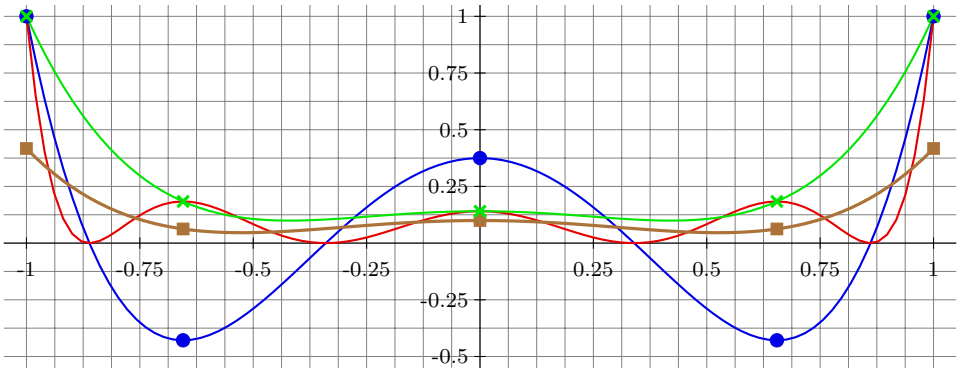


Figure 1: An example of polynomial aliasing when computing a nonlinear flux function. The blue curve is the solution polynomial and the blue dots give the value of this polynomial at the solution points. The red curve is the analytic flux polynomial and the green marks give the flux function evaluated at the solution points. A collocation projection results in the green curve which is the "aliased" flux polynomial through the flux values at the solution points. The brown curve is the L^2 projection of the analytic flux onto the solution polynomial space.

A family of correction functions for the left boundary that results in stable schemes is given by

$$g_{\text{LB}} = \alpha\, R_{R,P+1} + (1-\alpha)\, R_{R,P} \qquad \text{with} \qquad 0 \leq \alpha \leq 1 \tag{16}$$

Note that $\alpha = 1$ results in the DG method, and $\alpha = \frac{P+1}{2P+1}$ results in the modified Spectral Difference (SD) method that is stable.

II.B. Temporal Discretization

The results presented in this paper were integrated in time using the 3-stage strong stability preserving (SSP) Runge-Kutta method,[39] which is 3rd-order accurate in time. All of the results used a constant (global) time step to advance the solution in time. The size of the time step for each simulation was chosen to be sufficiently small meaning further reduction in its size would not produce significant changes in the solution error.

III. De-Aliasing using Over-Integration

High-order spectral-/finite-element type methods, including DG and FR, use a finite set of basis functions to construct a numerical solution within each grid cell. When applied to nonlinear equations, all of these methods can generate errors if any of the modes arising from the nonlinear terms are outside the span of the set of basis functions. In these cases, the energy from the unresolved modes are aliased onto the lower modes producing so-called aliasing errors. The effect these errors can have on a given simulation is highly unpredictable, and many researchers have shown they can produce severe instabilities within a problem.[13–21]

An example of how these aliasing errors can develop is shown in figure 1 where we have defined the flux function to be the square of the solution, i.e., $f(u) = u^2$. The blue curve represents the solution within the grid cell as a polynomial of degree 4 and the five blue dots give the value of the solution at each of the solution points. Analytically applying the flux function to the solution polynomial results in the analytic flux polynomial, which is shown as the red curve and is a polynomial of degree 8. Evaluating the analytic flux polynomial at each of the solution points results in the green 'x' symbols. A collocation projection of the analytic flux is the polynomial of degree 4 that interpolates these values of the flux function evaluated at the solution points. This is represented as the green curve in figure 1 and illustrates how the analytic flux polynomial is "aliased" onto a polynomial space of lower degree. Finally, a Galerkin or L^2 projection of the analytic flux onto the solution polynomial space is the best possible approximation in the L^2 norm that we can achieve of the analytic flux as a polynomial of degree 4. This projected flux polynomial is given by the brown curve and the brown squares mark the value of the polynomial at the solution points in order to show the difference from the collocation flux values.

One way to prevent aliasing errors is through over-integration, which is also referred to as polynomial de-aliasing. Let the *analytic flux*, f_{A}, be the exact analytic expression resulting from the evaluation of the

flux function using the solution polynomial u, i.e., $f_A = f(u)$. The goal of over-integration is to get an optimal approximation of the analytic flux as a polynomial within the solution polynomial space. This is accomplished by using a set of *quadrature points* to increase the sampling of the flux function above what is capable by the solution points. The interpolating polynomial created from this increased sampling is of a higher degree than the solution polynomial space. Finally, applying an L^2 projection to this higher-degree flux polynomial results in the best possible approximation of analytic flux in the L^2 norm as a polynomial within the solution polynomial space.

If the analytic flux happens to be a polynomial, we can recreate it with minimal error if enough sampling points are used. Figure 1 is an example of the analytic flux being a polynomial. For this particular example, if we used the value of the solution polynomial at 9 optimally-located points to evaluate the flux function, we could recreate the red curve exactly. However, care must be taken to limit the amount of over-integration that is used to only what is necessary to exactly recreate the analytic flux. At some limit, further sampling becomes superfluous, wasting computational resources to yield no additional information.

On the other hand, if the analytic flux is not a smooth and/or continuous function, no amount of over-sampling will be able to exactly recreate the analytic flux as a polynomial, and some aliasing error will be introduced because of this approximation. In these situations, we choose an amount of over-integration and hopefully the aliasing error will be negligible compared to the overall simulation error.

We implemented over-integration into our FR solver in two different ways. The first method uses the quadrature points to over-sample the flux function resulting in an approximation of the analytic flux as a polynomial of degree greater than P. An L^2 projection is then used to project this over-sampled flux onto the solution polynomial space. This method will be referred to as flux over-integration, and it involves a complex series of steps that adds two interpolation and two projection operations to the base FR method. The second method uses the quadrature points to over-sample the residual function to get an optimal approximation of the residual within the solution polynomial space. Referred to as residual over-integration, this method is much easier to implement since it adds only one projection operation to the base FR method. However, residual over-integration is generally more expensive since it requires computing the complete residual function using the quadrature points. Further details on both of these methods are provided in sections III.B and III.C.

III.A. Interpolation and Projection Operators

In order to use over-integration, we need to define two matrix operators: the interpolation and projection operators. The projection operator, $\boldsymbol{P}_n$, is used to project a nodal solution onto the space of polynomials of degree n, i.e., $\mathcal{P}_n$. The interpolation operator, $\boldsymbol{\mathcal{I}}_\eta$, is used to evaluate a function at a set of arbitrary nodal points, $\boldsymbol{\eta}$. For the sake of brevity, some details will be omitted in the process of defining these two operators that is to follow. A more thorough explanation of the general procedure being presented here can be found in Ref. 19.

We start by identifying some fundamental parameters and their respective notation. Recalling that P is the degree of the solution polynomial, let Q refer to the degree of the polynomial used for over-integration, where $Q \geq P$. Let N_Q refer to the minimum number of quadrature points needed to define the over-integration polynomial within the reference element $\mathbf{I}$. Using tensor products to extend the 1D reference element to higher dimensions, the total number of quadrature points for the $\mathcal{N}$-dimensional reference element is $N_Q = (Q+1)^{\mathcal{N}}$.

Note that we will use the term *nodal*, as in *nodal set* or *nodal points*, to refer to any arbitrary set of points within the reference element, e.g., both the solution and quadrature points are sets of nodal points. In order to discern between these two nodal sets we will adhere to the following notation. We retain the notation from section II.A when referencing the solution points: $\boldsymbol{\xi}$ will denote the set of N_P solution points, and the subscript P will denote a set relating to the solution points with the index n referring to a component of such a set, e.g., ξ_n. We define similar notation for the quadrature points: $\boldsymbol{\eta}$ will denote the set of N_Q quadrature points, and the subscript Q will denote a set relating to the quadrature points with the index q referring to a component of such a set, e.g., η_q. The term $\boldsymbol{\zeta}$ will be used to denote an arbitrary nodal set containing N nodal points, ζ_i, whenever we need to present a concept that is not unique to either the solution or quadrature points. Finally, the subscripts i and j will represent dummy indices which will be used whenever one of the dedicated indices is already present within an equation.

To help illustrate the various nodal sets used within the reference element, an example of a 2D reference element is provided in figure 2. This example corresponds to an element with $P = 1$ and $Q = 4$ resulting

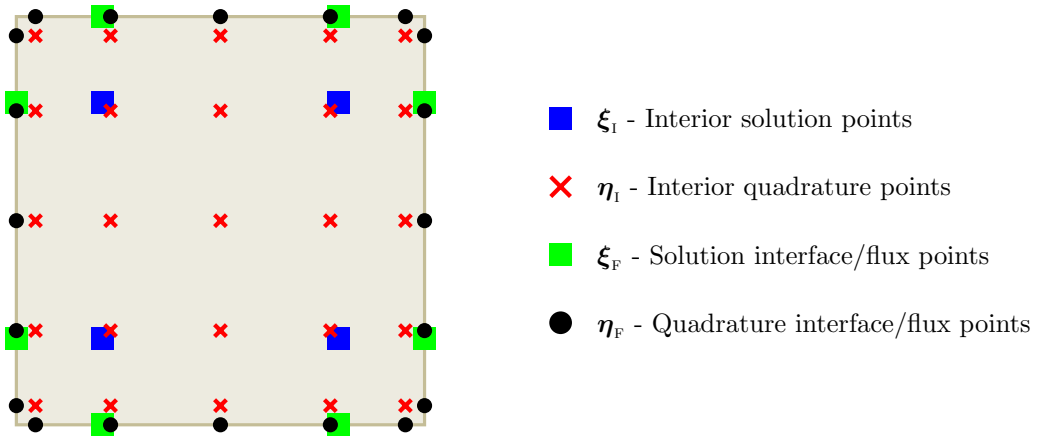


Figure 2: A generic grid cell illustrating the solution and quadrature points within the cell interior and along the cell faces.

in $N_P = (P+1)^2 = 4$ solution points (shown as blue squares) and $N_Q = (Q+1)^2 = 25$ quadrature points (shown as red 'x' symbols). Shown along the edges of the element in figure 2 are the interface/flux points required to account for interaction between adjacent cells. The nodal points along each edge correspond to the respective 1D nodal sets, with the solution flux points shown as green squares and the quadrature flux points shown as black circles.

The choice of nodal sets that are used can have an affect on the stability, accuracy, and efficiency of nodal-based polynomial methods like FR[31] and DG.[40,41] There are two families of nodal points that are used within this work: the Gauss-Legendre and the Lobatto-Gauss-Legendre nodes. In 1D, the Gauss-Legendre nodes, or more simply the Gauss nodes, correspond to the N zeros of the Legendre polynomial of degree N within the interval $[-1, 1]$. The Lobatto-Gauss-Legendre nodes, or more simply the Lobatto nodes, correspond to the $N-2$ zeros of the derivative of the Legendre polynomial of degree $N-1$ within the interval $[-1, 1]$, plus the two end points ± 1.

Each of these nodal sets is associated with a *quadrature rule* that defines a matching set of *quadrature weights* for each nodal point. Note that the terms quadrature rule and quadrature weights imply no direct reference to the quadrature points and simply refers to a set of nodes and matching weights that can be used to numerically approximate an exact integral by finite summation. Let $\boldsymbol{w}$ denote the set of N quadrature weights, w_i, $i = 1, 2, \ldots, N$, associated with each of the nodal points in $\boldsymbol{\zeta}$. With $a(r)$ representing some arbitrary function defined on $\mathbf{I}$, we can use the quadrature rule associated with $\boldsymbol{\zeta}$ to approximate the following integral as

$$\int_{-1}^{1} a(r)\ \mathrm{d}r \approx \sum_{i=1}^{N} a(\zeta_i)\, w_i \tag{17}$$

The benefit of using the set of N Gauss nodes and associated quadrature weights is the approximation in equation (17) is exact when $a(r)$ is a polynomial of degree $2N-1$ or less. When using the set of N Lobatto nodes, the approximation in equation (17) is exact when $a(r)$ is a polynomial of degree $2N-3$ or less.

Next, the inner product between two arbitrary functions, $a(r)$ and $b(r)$, within the reference element $\mathbf{I}$, is defined by the equation

$$(a\,, b) = \int_{\mathbf{I}} a(r)\, b(r)\ \mathrm{d}r \tag{18}$$

The two functions $a(r)$ and $b(r)$ are considered orthogonal to each other if the integral in equation (18) evaluates to 0. Using the quadrature rule associated with $\boldsymbol{\zeta}$, equation (18) can be approximated as

$$(a\,, b) \approx \sum_{i=1}^{N} a(\zeta_i)\, b(\zeta_i)\, w_i \tag{19}$$

Let $\boldsymbol{a}$ and $\boldsymbol{b}$ denote vectors that contain the values of their respective functions, $a(r)$ and $b(r)$, evaluated at each of the nodal points in $\boldsymbol{\zeta}$. Additionally, let $\boldsymbol{w}_\zeta$ denote the set of quadrature weights associated with $\boldsymbol{\zeta}$.

The weight matrix, $\boldsymbol{W}$, is then defined as the square, diagonal matrix with $\boldsymbol{w}_\zeta$ located along its diagonal, i.e., $\boldsymbol{W} = \text{diag}(\boldsymbol{w}_\zeta)$. Using $\boldsymbol{a}$, $\boldsymbol{b}$, and $\boldsymbol{W}$, the summation in equation (19) can be rewritten in matrix form as

$$(a\,,b) = \boldsymbol{a}^\mathsf{T}\boldsymbol{W}\boldsymbol{b} \tag{20}$$

In equation (20), the assumptions have been made that a and b are both polynomials and the quadrature rule associated with the nodal set $\boldsymbol{\zeta}$ is sufficiently accurate to exactly evaluate the integral in equation (19).

We wish to define an interpolation operator, $\boldsymbol{\mathcal{I}}$, which outputs the nodal coefficients at the quadrature points when given the nodal coefficients at the solution points. Let $\tilde{u}_q$ and $\tilde{\boldsymbol{u}}_Q$ relate to the quadrature points and be the respective equivalents to $\tilde{u}_n$ and $\tilde{\boldsymbol{u}}_P$. The nodal coefficients for all the quadrature points, can be found using equation (3)

$$\tilde{u}_q = u(\eta_q) = \sum_{n=1}^{N_P} \tilde{u}_n\, \ell_n(\eta_q) \qquad \text{for} \quad q = 1, 2, \ldots, N_Q \tag{21}$$

It is clear that the Lagrange polynomial term on the right side of equation (21) is simply a $N_Q \times N_P$ dimensioned matrix that is multiplied by the vector $\tilde{\boldsymbol{u}}_P$ resulting in $\tilde{\boldsymbol{u}}_Q$. Therefore, we can rewrite equation (21) in matrix form as

$$\tilde{\boldsymbol{u}}_Q = \boldsymbol{\mathcal{I}}_\eta \tilde{\boldsymbol{u}}_P \tag{22}$$

where the interpolation operator, $\boldsymbol{\mathcal{I}}_\eta$, with elements $\boldsymbol{\mathcal{I}}_{qn} = \ell_n(\eta_q)$, is a matrix that interpolates the solution nodal coefficients to the quadrature points, $\boldsymbol{\eta}$. For example, if $P = 1$ and $Q = 2$, the interpolation operator is the matrix

$$\boldsymbol{\mathcal{I}}_\eta = \begin{bmatrix} \ell_{\xi_1}(\eta_1) & \ell_{\xi_2}(\eta_1) \\ \ell_{\xi_1}(\eta_2) & \ell_{\xi_2}(\eta_2) \\ \ell_{\xi_1}(\eta_3) & \ell_{\xi_2}(\eta_3) \end{bmatrix} \tag{23}$$

where ℓ_{ξ_n} is used to specify the Lagrange polynomial that takes the value 1 at the solution point ξ_n and zero at all other solution points $\xi_{i\neq n}$.

Instead of using the nodal form of the solution polynomial given by equation (3), we can decompose the polynomial into hierarchical modes by using a set of *modal basis* functions and their corresponding *modal coefficients*. For an $\mathcal{N}$-D solution, let N_M refer to the maximum (or final) mode which corresponds to a polynomial basis function of degree $\mathcal{N} \times P$. Whereas N_M denotes the maximum mode, the total number of modes is actually the same as the total number of solution points because we have defined the modal index, m, to begin at 0; this was done to signify that the first mode, i.e., $m = 0$, corresponds to the constant mode where all the polynomial exponents are 0. Thus, the value of N_M is $N_M = (P+1)^\mathcal{N} - 1 = N_P - 1$. Combining all of these notations, let $\hat{\boldsymbol{u}}_P$ denote the set of modal coefficients, $\hat{u}_m$, $m = 0, 1, \ldots, N_M$; we have used the 'hat' accent in order to distinguish it from the tilde accented nodal coefficients.

Let $\boldsymbol{\phi}_P$ be the set of N_P modal basis functions, $\phi_m(r)$, $m = 0, 1, \ldots, N_M$, that directly correspond to the modal coefficients in $\hat{\boldsymbol{u}}_P$. Note that for the 1D case, the basis function $\phi_m(r)$ is a polynomial of degree m. Let $\boldsymbol{B}$ denote the basis matrix where each entry in the matrix is given by $\boldsymbol{B}_{ij} = \phi_j(\zeta_i)$. Thus, each column vector of the matrix represents one of the basis functions evaluated at each point in a nodal set, whereas each row vector of the matrix represents each basis function evaluated at a specific nodal point.

In the formulation of the projection operator, we will deal with two basis matrices. The first basis matrix, which retains the notation $\boldsymbol{B}$, or $\boldsymbol{B}_{Q,P}$, has the dimensions $N_Q \times N_P$ and corresponds to each of the N_P basis functions evaluated at each of the N_Q quadrature points. Using the same example of $P = 1$ and $Q = 2$ from equation (23), the basis matrix expands to

$$\boldsymbol{B}_{2,1} = \begin{bmatrix} \phi_0(\eta_1) & \phi_1(\eta_1) \\ \phi_0(\eta_2) & \phi_1(\eta_2) \\ \phi_0(\eta_3) & \phi_1(\eta_3) \end{bmatrix} \tag{24}$$

In the present work, the normalized Legendre polynomials are used as the modal basis functions. These are ideal because they fit the hierarchical requirement and the Legendre polynomial of degree m is orthogonal to $\mathcal{P}_{m-1}$.

The second basis matrix that we will use is denoted by $\boldsymbol{V}$, and represents the N_P basis functions evaluated at each of the N_P solution points, i.e., $\boldsymbol{V} = \boldsymbol{B}_{P,P}$. The matrix $\boldsymbol{V}$ is denoted differently than $\boldsymbol{B}$ to immediately recognize that it is square and invertible, whereas $\boldsymbol{B}$ is generally not square and thus not invertible.

Using the modal coefficients and corresponding basis functions, the approximate solution polynomial from equation (3) can be reconstructed using

$$u(r) = \sum_{m=0}^{N_M} \hat{u}_m \phi_m(r) \tag{25}$$

If we use equation (25) to evaluate the nodal coefficients for each of the N_Q quadrature points, we end up with

$$\tilde{u}_q = u(\eta_q) = \sum_{m=0}^{N_M} \hat{u}_m \phi_m(\eta_q) \qquad \text{for} \quad q = 1, 2, \ldots, N_Q \tag{26}$$

Similar to how equation (21) was rewritten as equation (22), we can rewrite equation (26) in matrix form as

$$\tilde{\boldsymbol{u}}_Q = \boldsymbol{B}\hat{\boldsymbol{u}}_P \tag{27}$$

If we were instead to use equation (25) to evaluate the nodal coefficients for each of the N_P solution points, we end up with the matrix equation

$$\tilde{\boldsymbol{u}}_P = \boldsymbol{V}\hat{\boldsymbol{u}}_P \tag{28}$$

Because the matrix $\boldsymbol{V}$ is invertible, we can very easily rearrange the above equation to solve for the modal coefficients given the nodal coefficients at the solution points, i.e.,

$$\hat{\boldsymbol{u}}_P = \boldsymbol{V}^{-1}\tilde{\boldsymbol{u}}_P \tag{29}$$

From the previous equations, we can see that the basis matrix is essentially an operator that transforms the modal solution into the nodal solution. This completes the definition of all the necessary components that are required for us to define the projection operator.

Before we continue on to the projection operator, however, we take the opportunity to quickly present an alternative method for evaluating the interpolation operator from the basis matrices. If we plug equation (29) into equation (27), and then set the result equal to equation (22), we end up with

$$\boldsymbol{B}\left[\boldsymbol{V}^{-1}\tilde{\boldsymbol{u}}_P\right] = \tilde{\boldsymbol{u}}_Q = \boldsymbol{\mathcal{I}}_\eta \tilde{\boldsymbol{u}}_P \tag{30}$$

The alternative formulation of the interpolation operator is clearly

$$\boldsymbol{\mathcal{I}}_\eta = \boldsymbol{B}\boldsymbol{V}^{-1} \tag{31}$$

We now continue on with defining the projection operator. Because the exact solution, $u_{\text{E}}(r)$, is not necessarily in the space of polynomials spanned by the basis functions of the approximate solution, $u(r)$, the residual error, $R(u)$, from the approximation can be expressed as

$$R(u) = u_{\text{E}}(r) - u(r) \tag{32}$$

Let φ represent an arbitrary test function. We can apply the inner product of equation (32) with respect to the test function, φ, resulting in

$$\big(\varphi, R(u)\big) = \big(\varphi, u_{\text{E}}\big) - \big(\varphi, u\big) \tag{33}$$

From the method of weighted residuals, we require that the left hand side of the previous equation is 0.

The Galerkin projection, also referred to as the L^2 projection, uses the modal basis functions, $\boldsymbol{\phi}$, as the test functions in equation (33). Let us assume that we have used a set of quadrature points, $\boldsymbol{\eta}$, that is large enough to sufficiently sample the exact solution such that

$$u_{\text{E}}(r) \approx \sum_{q=1}^{N_Q} \tilde{u}_q \, \ell_q(r) \tag{34}$$

After setting the left hand side of equation (33) to zero, if we replace u_{E} with the approximation in equation (34), u with the modal formulation from equation (25), and the test function with the set of modal basis functions, we end up with

$$\left(\phi_i \,, \sum_{m=0}^{N_M} \hat{u}_m \phi_m\right) = \left(\phi_i \,, \sum_{q=1}^{N_Q} \tilde{u}_q \, \ell_q\right) \qquad \text{for} \quad i = 0, 1, \ldots, N_M \tag{35}$$

The summations can be pulled out of the inner products since $\tilde{u}_q$ and $\hat{u}_m$ are just coefficients within their respective summations. Thus equation (35) simplifies to

$$\sum_{m=0}^{N_M} \hat{u}_m \left(\phi_i \, , \phi_m\right) = \sum_{q=1}^{N_Q} \tilde{u}_q \left(\phi_i \, , \ell_q\right) \qquad \text{for} \quad i = 0, 1, \ldots, N_M \tag{36}$$

Using equation (19) and the quadrature rule associated with the quadrature points, the inner products in equation (36) can also be expressed as summations

$$\sum_{m=0}^{N_M} \hat{u}_m \sum_{j=1}^{N_Q} \phi_i(\eta_j) \, \phi_m(\eta_j) \, w_j = \sum_{q=1}^{N_Q} \tilde{u}_q \sum_{j=1}^{N_Q} \phi_i(\eta_j) \, \ell_q(\eta_j) \, w_j \qquad \text{for} \quad i = 0, 1, \ldots, N_M \tag{37}$$

Using equation (20), this can be rewritten in matrix form as

$$\boldsymbol{B}^{\mathsf{T}} \boldsymbol{W} \boldsymbol{B} \, \hat{\boldsymbol{u}}_{P} = \boldsymbol{B}^{\mathsf{T}} \boldsymbol{W} \boldsymbol{I} \, \tilde{\boldsymbol{u}}_{Q} \tag{38}$$

where $\boldsymbol{I}$ is the identity matrix created from the Lagrange polynomials for the quadrature points, i.e.,

$$\boldsymbol{I} = \ell_q(\eta_j) = \delta_{qj} \qquad \text{for} \quad q, j = 1, 2, \ldots, N_Q \tag{39}$$

The leading term on the left hand side of equation (38), $\boldsymbol{B}^{\mathsf{T}} \boldsymbol{W} \boldsymbol{B}$, results in a square matrix that can be inverted and moved to the right side giving

$$\hat{\boldsymbol{u}}_{P} = \left[\left(\boldsymbol{B}^{\mathsf{T}} \boldsymbol{W} \boldsymbol{B} \right)^{-1} \boldsymbol{B}^{\mathsf{T}} \boldsymbol{W} \right] \tilde{\boldsymbol{u}}_{Q} \tag{40}$$

The previous equation takes the set of N_Q nodal values defined at the quadrature points, $\tilde{\boldsymbol{u}}_{Q}$, and projects these values onto the solution polynomial space of degree P, resulting in the set of modal coefficients for the basis functions of the solution polynomial, $\hat{\boldsymbol{u}}_{P}$.

Using equation (28), we can transform the projected modal coefficients from equation (40) into the physical values at the solution points. Multiplying $\boldsymbol{V}$ by the term in square brackets from equation (40) gives the final projection operator,

$$\boldsymbol{P}_{P} = \boldsymbol{V} \left(\boldsymbol{B}^{\mathsf{T}} \boldsymbol{W} \boldsymbol{B} \right)^{-1} \boldsymbol{B}^{\mathsf{T}} \boldsymbol{W} \tag{41}$$

where $\boldsymbol{P}_{P}$ denotes that the projection is onto the polynomial space of degree P. Thus, the projection operator is used to project the Q-degree polynomial defined by the nodal coefficients $\tilde{\boldsymbol{u}}_{Q}$ to a polynomial of degree P, which is then evaluated at the solution points resulting in the nodal coefficients $\tilde{\boldsymbol{u}}_{P}$, i.e.,

$$\tilde{\boldsymbol{u}}_{P} = \boldsymbol{P}_{P} \tilde{\boldsymbol{u}}_{Q} \tag{42}$$

Note that the projection operator above is simply a matrix and its use only requires multiplying it by the vector of nodal coefficients for the quadrature points. Since the matrices defining the operator are based on quadrature rules and basis functions that do not change during a simulation, the operator matrix is only required to be defined once during the preprocessing phase of the simulation.

As a quick exercise, we take a look at what happens to the projection operator if over-integration is not used, i.e., $Q = P$ and $\boldsymbol{\eta} = \boldsymbol{\xi}$. In this specific case, the basis matrices $\boldsymbol{B}$ and $\boldsymbol{V}$ are equivalent. Substituting $\boldsymbol{V}$ for $\boldsymbol{B}$ throughout equation (41) results in

$$\boldsymbol{P}_{P} = \boldsymbol{V} \left(\boldsymbol{V}^{\mathsf{T}} \boldsymbol{W} \boldsymbol{V} \right)^{-1} \boldsymbol{V}^{\mathsf{T}} \boldsymbol{W} \tag{43}$$

Since all of these matrices are now square and invertible, we can use some matrix identities to expand equation (43) as

$$\boldsymbol{P}_{P} = \boldsymbol{V} \boldsymbol{V}^{-1} \boldsymbol{W}^{-1} (\boldsymbol{V}^{\mathsf{T}})^{-1} \boldsymbol{V}^{\mathsf{T}} \boldsymbol{W} \tag{44}$$

After canceling terms, the above projection operator is clearly the identity matrix. This is exactly what we expect because here we are trying to project a polynomial of degree P onto the polynomial space of degree P.

III.B. Flux Over-Integration

The first over-integration method focuses on preventing aliasing errors that might arise from the evaluation of two flux related terms: the discontinuous flux divergence within the cell interior and the common flux at the cell interfaces. Because these two terms are independent of each other, we can split this method into two phases with each phase corresponding to one term. Both phases are similar in that they involve four basic steps: interpolating the nodal solution to the quadrature points, evaluate a flux function at the quadrature points, projecting this higher-order flux polynomial down to the solution polynomial space, and finally interpolating this projected flux polynomial from the quadrature points back to the solution points. As we showed in equations (41) and (42), the final step of interpolating the projected flux from the quadrature points to the solution points can be absorbed into the projection operator for improved efficiency since both operations are matrix multiplications.

Let $\boldsymbol{\xi}$ and $\boldsymbol{\eta}$ respectively refer to sets of solution and quadrature points as in section III.A, where we were exclusively dealing with interior nodal points. However, we are now required to differentiate between interior and interface nodal sets, because the first phase of flux over-integration involves the flux function within the interior of the element, whereas the second phase involves the flux along the interfaces. We therefore introduce the subscripts 'I' and 'F' to denote the respective interior and interface nodal sets. The legend in figure 2 provides an example of this notation as applied to the reference quadrilateral element.

The interpolation and projection operators for the interior points follow directly from section III.A. It is not immediately clear though, how these operators are applied to the interface nodal points where the spatial dimension is one lower than that of the interior nodal points. The interpolation operator is actually straightforward; assuming we know the location of the interface points, the matrix that interpolates a polynomial from $\boldsymbol{\xi}_{\text{I}}$ to either $\boldsymbol{\xi}_{\text{F}}$ or $\boldsymbol{\eta}_{\text{F}}$ is evaluated using a method similar to equation (21). Whereas the projection operator for the interface polynomial is still defined using equation (41), it requires all of the matrices on the right side of the equation to be recreated for the face reference element. In equation (41), the matrices $\boldsymbol{V}$, $\boldsymbol{B}$, and $\boldsymbol{W}$ are all defined for a $\mathcal{N}$-D element and not applicable to the $(\mathcal{N}-1)$-D polynomial we are looking to project at the interface. By using $\boldsymbol{\xi}_{\text{F}}$ and $\boldsymbol{\eta}_{\text{F}}$ to create interface versions of $\boldsymbol{V}$, $\boldsymbol{B}$, and $\boldsymbol{W}$, we can create the interface projection operator with equation (41).

III.B.1. Over-Integrating the Discontinuous Flux Divergence

The first phase of flux de-aliasing is applying over-integration to the evaluation of the discontinuous flux divergence. We start with assuming that the nodal coefficients are known at the interior solution points, $\boldsymbol{\xi}_{\text{I}}$. The steps for evaluating the de-aliased discontinuous flux divergence are as follows:

1. Interpolate the solution from $\boldsymbol{\xi}_{\text{I}}$ to $\boldsymbol{\eta}_{\text{I}}$.
2. Evaluate the flux function at $\boldsymbol{\eta}_{\text{I}}$ using the interpolated solutions.
3. Project the flux polynomial to the solution polynomial space.
4. Interpolate the projected flux polynomial from $\boldsymbol{\eta}_{\text{I}}$ to $\boldsymbol{\xi}_{\text{I}}$.
5. Compute the derivative of the projected flux polynomial at each $\boldsymbol{\xi}_{\text{I}}$.

Following the previous steps results in the discontinuous flux as a polynomial of degree P, thus its divergence is of degree $P-1$. This is same as the base FR method except here f^{D} is a better approximation of the analytic flux in the L^2 norm.

Additionally, we can define an alternative method to over-integrate the discontinuous flux by rearranging the final three steps as:

3. Compute the derivative of the flux polynomial at each $\boldsymbol{\eta}_{\text{I}}$.
4. Project the derivative of the flux polynomial to the solution polynomial space.
5. Interpolate the projected flux derivative from $\boldsymbol{\eta}_{\text{I}}$ to $\boldsymbol{\xi}_{\text{I}}$.

This slight modification results in $f^{\text{D}}_r(r)$ being a polynomial of degree P instead of $P-1$ as it was before. However, this can create some consistency errors when evaluating the flux jumps at the cell interfaces. The values $f^{\text{D}}(\pm 1)$, which are used to compute the flux jumps at the interfaces in equation (9), need to be consistent with the projected derivative. This means we need to create a discontinuous flux polynomial of degree $P+1$ from $f^{\text{D}}_r(r)$ in order to compute the flux jumps in a consistent manner.

Another consequence of projecting the derivative is that the derivative operator needs to be created from the Lagrange polynomials for the quadrature points instead of the solution points. Therefore, computing the

flux derivative at the quadrature points is significantly more expensive because of the increase in operations required by the matrix-vector product. Brainstorming these and other complications that we encountered led to an alternative method, residual over-integration, which is described in section III.C.

III.B.2. Over-Integrating the Upwind Flux

At each interface/flux point, the upwind flux, f^*, is found by solving the Riemann problem that exists from the (generally) discontinuous solution between the two cells on the interface, i.e., $\tilde{u}_L \neq \tilde{u}_R$. Numerous approximate Riemann solvers have been derived for solving this problem; they all operate on a pair of individual states and the underlying algorithms generally contain nonsmooth functions and/or conditional statements. However, for polynomial methods, the Riemann problem at an interface is actually between two discontinuous solution *polynomials*, $u_L(r) \neq u_R(r)$. The solution to this Riemann problem is extremely complicated and all but guaranteed to not be a polynomial. Therefore, the upwind flux could very likely be the source of aliasing errors if we take an insufficient sampling of the total interface flux.

The quadrature flux points, $\boldsymbol{\eta}_{\text{F}}$, are used to get a better sampling of the total interface flux, which can then be projected onto the solution polynomial space at the solution flux points $\boldsymbol{\xi}_{\text{F}}$. For a given interface between two cells, the steps to achieve this are as follows:

1. From the left and right cells on the interface, interpolate the solution from their respective $\boldsymbol{\xi}_{\text{I}}$ to the shared quadrature flux points, $\boldsymbol{\eta}_{\text{F}}$.
2. Evaluate f^* at $\boldsymbol{\eta}_{\text{F}}$ using the left and right interpolated solutions.
3. Project f^* to the solution polynomial space.
4. Interpolate f^* from $\boldsymbol{\eta}_{\text{F}}$ to $\boldsymbol{\xi}_{\text{F}}$.

After following these steps, each solution flux point will contain the nodal value of the optimal polynomial approximation of the upwind flux within the solution polynomial space.

III.C. Residual Over-Integration

The second over-integration method focuses on preventing aliasing errors by over-sampling the residual function. Using the interpolation and projection operators from section III.A, over-integration of the residual proceeds as follows:

1. Interpolate the solution from $\boldsymbol{\xi}_{\text{I}}$ to $\boldsymbol{\eta}_{\text{I}}$.
2. Use the base FR method at the nodes $\boldsymbol{\eta}_{\text{I}}$ and $\boldsymbol{\eta}_{\text{F}}$ to evaluate the residual function at $\boldsymbol{\eta}_{\text{I}}$.
3. Project the higher-order residual polynomial to the solution polynomial space.
4. Interpolate the projected residual polynomial from $\boldsymbol{\eta}_{\text{I}}$ to $\boldsymbol{\xi}_{\text{I}}$.
5. Update the solution at $\boldsymbol{\xi}_{\text{I}}$ using the de-aliased residual polynomial.

In the previous algorithm, there is a redundancy proceeding through steps 4, 5, and restarting back at step 1. In step 4, the projected residual is interpolated from the quadrature points to the solution points for the sole purpose of evaluating the updated solution in step 5. Restarting back at step 1 just interpolates the updated solution right back to the quadrature points. We can instead skip the interpolations all together and perform the solution update at the quadrature points since this is simply a linear combination of the solution and residual polynomials. This removes any need for using the solution points since all operations are performed at the quadrature points.

The key feature of this second over-integration method is that it is very easy to implement within an existing code since it only requires the addition of a single matrix multiplication to each time (sub-)step. The code is basically mislead into thinking it is running a simulation with the solution order set to the quadrature order. This way the code sets up the simulation with the solution points actually being the quadrature points and all supporting data structures are created accordingly. As the simulation is advanced in time, the residual function is evaluated as usual within each time step or Runge-Kutta stage. Just before the solution is updated after a time step or Runge-Kutta stage, the higher-order residual polynomial is projected down to the solution polynomial space. Finally, the solution is updated using this de-aliased residual polynomial before continuing to the next time step.

IV. Governing Equations

The Euler equations describe the motion of an inviscid fluid such as the isentropic vortex that is the topic of this paper. The 2D Euler equations for a calorically perfect, compressible fluid can be written in differential form as

$$\frac{\partial}{\partial t}\begin{bmatrix}\rho\\ \rho v_x\\ \rho v_y\\ \rho E\end{bmatrix}+\frac{\partial}{\partial x}\begin{bmatrix}\rho v_x\\ \rho v_x^2\\ \rho v_x v_y\\ v_x\left(\rho E+p\right)\end{bmatrix}+\frac{\partial}{\partial y}\begin{bmatrix}\rho v_y\\ \rho v_x v_y\\ \rho v_y^2\\ v_y\left(\rho E+p\right)\end{bmatrix}=0 \tag{45}$$

where ρ, v_i, p, and E respectively refer to the density, velocity in coordinate direction i, pressure, and the total energy per unit mass of the system. Thermodynamic closure is found through the total energy equation

$$\rho E=\frac{p}{\gamma-1}+\tfrac{1}{2}\rho\left(v_x^2+v_y^2\right) \tag{46}$$

where γ is the ratio of specific heats for the fluid in question. Additionally, the ideal gas law

$$p=\rho R_{gas}T \tag{47}$$

is utilized to relate the temperature, T, to the pressure and density where R_{gas} is the specific gas constant of the fluid. The speed of sound, a, is computed using the relation

$$a=\sqrt{\frac{\gamma p}{\rho}}=\sqrt{\gamma R_{gas}T} \tag{48}$$

In the present work, the fluid is assumed to be calorically perfect air where $\gamma=1.4$ and the dimensionalized specific gas constant is $R_{gas}=287.15$ J/(kg·K).

V. Numerical Results

In the discussion of the results we will frequently refer to the degree of the solution polynomial used for a simulation as well as the degree of the polynomial used for over-integration. Therefore, we need to define a few notational conventions that will be used in this section to refer to these quantities. Let $\mathcal{P}n$ refer to a solution polynomial of degree n that nominally results in a $(n+1)^{\text{th}}$-order method and let $\mathcal{Q}m$ refer to over-integrating using a polynomial of degree m. Combining these, we will use $\mathcal{P}n\mathcal{Q}m$ to refer to a $\mathcal{P}n$ solution that is over-integrated to $\mathcal{Q}m$.

V.A. Isentropic Euler Vortex Problem

The isentropic Euler vortex problem is commonly used[16,17,20,36,42–47] for testing the order of accuracy of a numerical method because it is easy to implement and the exact solution is known at all times. This is especially the case for high-order methods that are designed to produce minimal numerical dissipation. This problem is ideal for high-order methods because their theoretical accuracy should allow them to propagate the vortex with minimal entropy production for an indefinite amount of time.

In Ref. 48, we made an attempt to create a definition for the isentropic Euler vortex problem that unifies many of the variations that exist in literature. While performing some h-refinement studies with this problem, we encountered a few instabilities of varying strengths. We were able to determine that some of these instabilities were caused by poorly chosen boundary conditions and the size of the computational domain being too small. By zeroing the mean flow velocity to keep the vortex stationary, we were able to circumvent the problem with the boundary conditions and successfully complete an h-refinement study showing the long term stability of the FR method.

Whereas the h-refinement study was successful, some of the lower-resolution simulations for the stationary vortex still showed signs of instabilities. Since this problem has been shown to exhibit aliasing errors that can affect the stability of the numerical simulation,[17,20,21] we identified these cases as good candidates for testing over-integration as a means of reducing aliasing errors and improving stability. For completeness, we will also explore the effects of over-integration on an extremely under-resolved stationary vortex as well as its general effects related to the nodal quadrature through an h/p-refinement study.

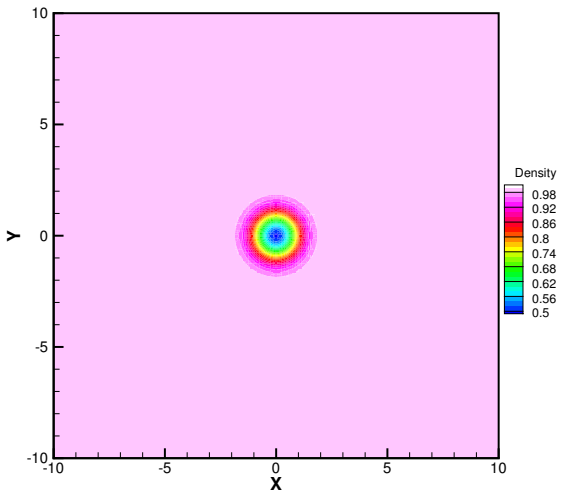


Figure 3: Contours of the initial density profile for the stationary vortex problem.

Additional instabilities can be introduced within the system by periodic boundaries and we need to isolate the effects of over-integration on possible aliasing instabilities. Keeping the vortex stationary is the only way to run the vortex problem for any significant amount of time without using periodic boundary conditions. Therefore, we will use the stationary version of the isentropic Euler vortex problem as defined in Ref. 48.

A condensed definition for the stationary vortex problem follows since we are not concerned about unifying variations of the vortex problem found in literature. The initial conditions for the vortex problem are found by superimposing perturbations in velocity and temperature onto a uniform mean flow. These perturbations are defined such that the entropy, $S = p/\rho^\gamma$, is constant throughout the entire grid domain, i.e., $\delta S = 0$. In order to keep the vortex stationary, the mean flow velocity, $(v_{x,\infty}, v_{y,\infty})$, is required to be zero and the free-stream density, ρ_∞, and pressure, p_∞, are set to one.

The perturbations in velocity and temperature are defined as

$$\begin{aligned} \delta v_x &= -\,y\,\Omega \\ \delta v_y &= +\,x\,\Omega \\ \delta T &= -\frac{(\gamma-1)}{2}\,\Omega^2 \end{aligned} \tag{49}$$

where Ω is a Gaussian function of the form

$$\Omega = \frac{5}{2\pi\sqrt{\gamma}}\,\mathrm{e}^{-\frac{1}{2}\left(1-x^2-y^2\right)} \tag{50}$$

These perturbations, along with the free-stream conditions and the isentropic relations between density, pressure, and temperature, are used to define the initial flow conditions of the primitive variables as

$$\begin{aligned} \tilde{\rho}_0 &= (1+\delta T)^{\frac{1}{\gamma-1}} \\ \tilde{v}_{x,0} &= \delta v_x \\ \tilde{v}_{y,0} &= \delta v_y \\ \tilde{p}_0 &= \tfrac{1}{\gamma}\,(1+\delta T)^{\frac{\gamma}{\gamma-1}} \end{aligned} \tag{51}$$

where the subscript 0 denotes a quantity given at $t = 0$. The tilde accents on the primitive variables indicate that each is a nondimensional quantity where ρ_∞, a_∞, and T_∞ have been used as the characteristic density, velocity, and temperature scales, respectively.

The last remaining item needed to complete the problem is the definition of the computation domain. For all of the simulations in this work, we used a square domain with grid boundaries located at

$(x, y) = (\pm 10, \pm 10)$. A number of different grid resolutions was used with each grid divided into regular, nonoverlapping quadrilateral cells of equal size. Finally, all grid boundaries were set to characteristic boundary conditions to allow for information to flow in/out of the domain as needed. A contour plot showing the initial density profile resulting from this problem definition is shown in figure 3.

The beauty of this problem is that it can be run indefinitely and the exact analytical solution at any point in time is simply the initial conditions. For the propagating vortex, time can additionally be measured in flow-through periods or the number of times the vortex propagates completely through the domain and returns back to its initial location. Even though a flow-through period for the stationary vortex is undefined, we will still use it to measure the elapsed time of our simulations. In this case, one period is equivalent to the amount of time for a propagating vortex with $v_{x,\infty} = 1$ to complete one flow-through period.

V.B. Over-Integration Applied to a Very Under-Resolved Case

The first case used for testing over-integration is an attempt to visualize the effect that it has on the modes of the solution. Aliasing errors are more commonly found with under-resolved problems. In these cases, there is modal energy that should exist in modes higher than the span of the polynomial basis functions. These energies are erroneously resolved by the highest modes of the polynomial basis resulting in an aliased solution. Therefore, we should be able to use an extremely under-resolved case to see how well over-integration is able to remove energy from the highest modes in the areas where these modes should not be activated.

To create a very under-resolved case, we used a grid containing 10^2 cells and a $\mathcal{P}8$ solution with and without over-integration to $\mathcal{P}8\mathcal{Q}16$. The simulations were run for one period and the absolute value of the density modes were plotted for each grid cell in figure 4, with all modes less than 1.0×10^{-7} cut off (set to white).

The two images at the top of figure 4 are given to illustrate how the modes are to be interpreted within each cell. These two images show up-close views of the grid cell bounded by the lines $x = 0$, $x = 2$, $y = 2$, and $y = 4$ from the respective image located directly below. The box in the lower left corner of the cell gives the value of the modal coefficient, c_m, for the polynomial term $c_m x^0 y^0$, i.e., the constant mode. Moving horizontally, the exponent for x increases from left to right. Moving vertically, the exponent for y increases from bottom to top. The box at the top right gives the value of the modal coefficient for the polynomial term $c_m x^8 y^8$.

As discussed previously, we would expect to see the modes or boxes towards the top and right sides of each grid cell to be cut off or darker blue when using over-integration. More importantly, we would expect to see the boxes in the outer regions of the domain to have only the constant coefficient activated since those regions only contain the mean flow. Both of these expectations are clearly met when comparing the over-integrated modal solution at the bottom right of figure 4 to the base solution located at the bottom left of figure 4. This is a clear indication that over-integration is working as intended.

V.C. Over-Integration Applied to Well-Resolved Cases

We know that over-integration is working as intended in the previous under-resolved case, but what happens if we apply it to a moderately or well-resolved case? We tested various levels of over-integration on five different grid levels, 40^2, 80^2, 120^2, 160^2, and 200^2, and using $\mathcal{P}2$–$\mathcal{P}4$ order solutions. For each of the combinations of grid level and polynomial order, we computed the density error through one period using three "base" methods without over-integration: Gauss-LP, Lobatto-LP, and Lobatto-CR. We then compare those results to simulations with varying degrees of over-integration applied to the Gauss-LP and Lobatto-LP methods. We did not use over-integration with the Lobatto-CR method since it already offers an additional order of accuracy over the Lobatto-LP method. Instead we wanted to see how over-integration with the Lobatto-LP method compares to the base Lobatto-LP and Lobatto-CR methods.

There are a couple of consistencies between figure 5 and the following figures 6 and 7 to identify before continuing. First, across all three figures the dashed lines give the base Lobatto-LP (long dashes) and Lobatto-CR (short dashes) results and the solid line with square symbols gives the Gauss-LP results. Second, all the results on a given grid resolution are plotted using the same color in all the three figures. More specifically, the red, green, blue, magenta, and black groups of lines respectively represent the grid levels containing 40^2, 80^2, 120^2, 160^2, and 200^2 cells.

Figure 5 shows the results for all of the $\mathcal{P}2$ order simulations on each of the grid levels. Two sets of simulations involving over-integration are shown alongside the three base methods: Gauss-LP (solid line with

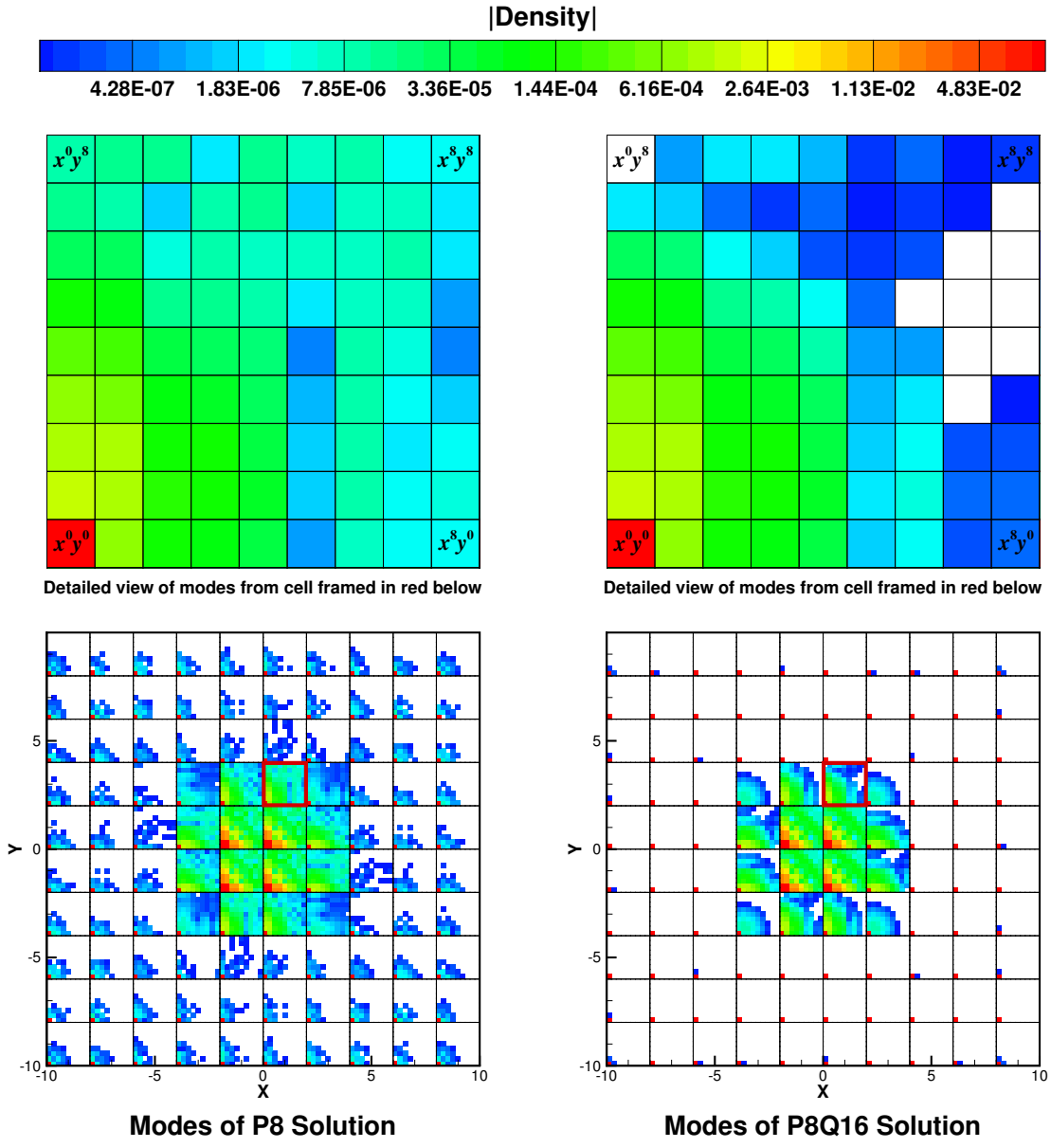


Figure 4: Comparison of the $\mathcal{P}8$ modal solution without over-integration (left) and with over-integration to $\mathcal{P}8\mathcal{Q}16$ (right). The top two images show a detailed view of the modes for the cells highlighted in red in the respective image below. All modes less than 1.0×10^{-7} have been cut off (set to white).

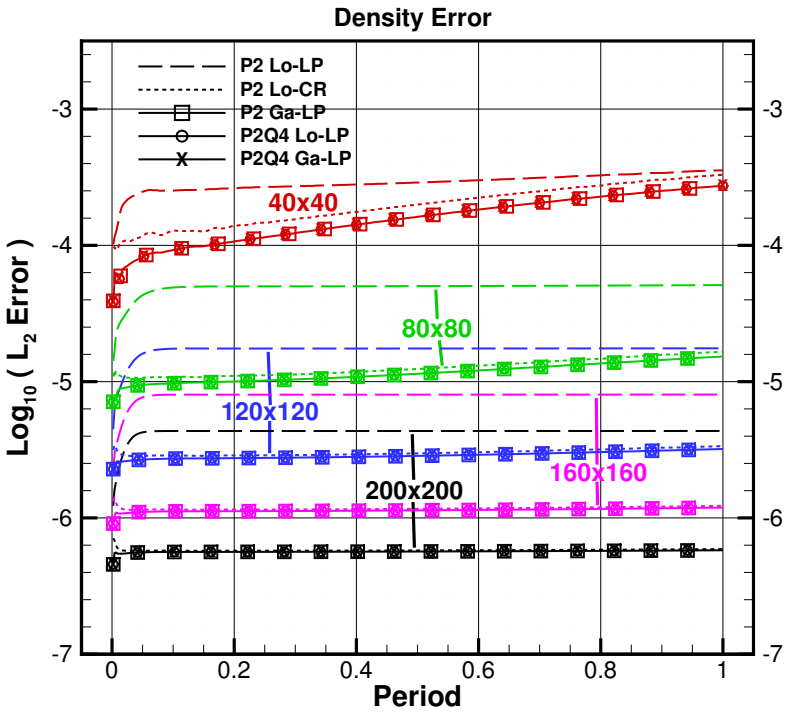


Figure 5: Comparison of $\mathcal{P}2$ solutions with and without over-integration on multiple grid levels.

'x' symbols) and Lobatto-LP (solid line with circle symbols) both over-integrating to $\mathcal{P}2\mathcal{Q}4$. The results for the three base methods follow the same trends as found by other researchers:[33] the error of the Lobatto-LP method is between 0.5–1.0 magnitude higher than the Gauss-LP method and the error of the Lobatto-CR method is just slightly higher than the Gauss-LP method. The very interesting result is that the error from both over-integration methods is identical to the base Gauss-LP method across all grid levels.

Figure 6 shows the results for all of the $\mathcal{P}3$ order simulations on each of the grid levels. As with the $\mathcal{P}2$ results, we have included two sets of simulations involving over-integration alongside the three base methods. Again, we have included over-integration to $\mathcal{P}3\mathcal{Q}6$ with the Gauss-LP method (solid line with circle symbols) but this time we reduced the over-integration with the Lobatto-LP method (solid line with 'x' symbols) by one order to $\mathcal{P}3\mathcal{Q}5$. Even with this change, all of the over-integration results are again identical to the base Gauss-LP method.

Finally, figure 7 shows the results for all of the $\mathcal{P}4$ order simulations on each of the grid levels. For this case, we did not include any over-integration results for the Gauss-LP method and instead focused on finding what degree of over-integration is required for the Lobatto-LP method to reach the same result as the base Gauss-LP method. We have included three sets of simulations with varying degrees of over-integration with the Lobatto-LP method alongside the three base methods: over-integrating to $\mathcal{P}4\mathcal{Q}8$ (solid line with diamond symbols), $\mathcal{P}4\mathcal{Q}6$ (solid line with circle symbols), and $\mathcal{P}4\mathcal{Q}5$ (solid line with 'x' symbols). Again, all of the over-integration results are identical to the base Gauss-LP results.

There are three important conclusions that can be made from applying over-integration to these moderately to well-resolved cases. First, when using Gauss solution points, over-integrating does nothing and gives the same result as if it had not been used at all. Second, when using Lobatto solution points, over-integration improves the answer to that found by using Gauss solution points. Finally, over-integrating by just one degree higher might be enough to remove any aliasing errors inherent to using Lobatto solution points.

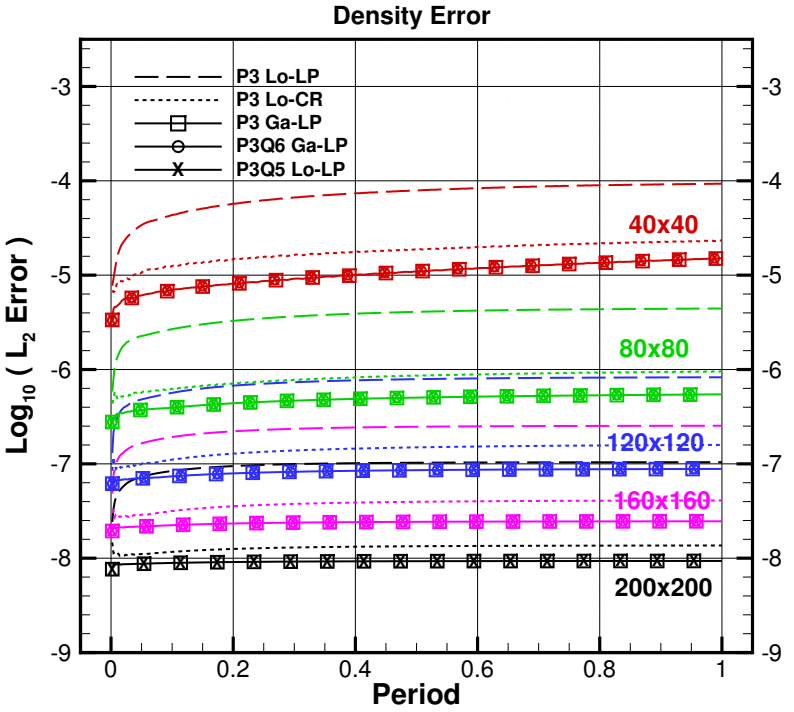


Figure 6: Comparison of $\mathcal{P}3$ solutions with and without over-integration on multiple grid levels.

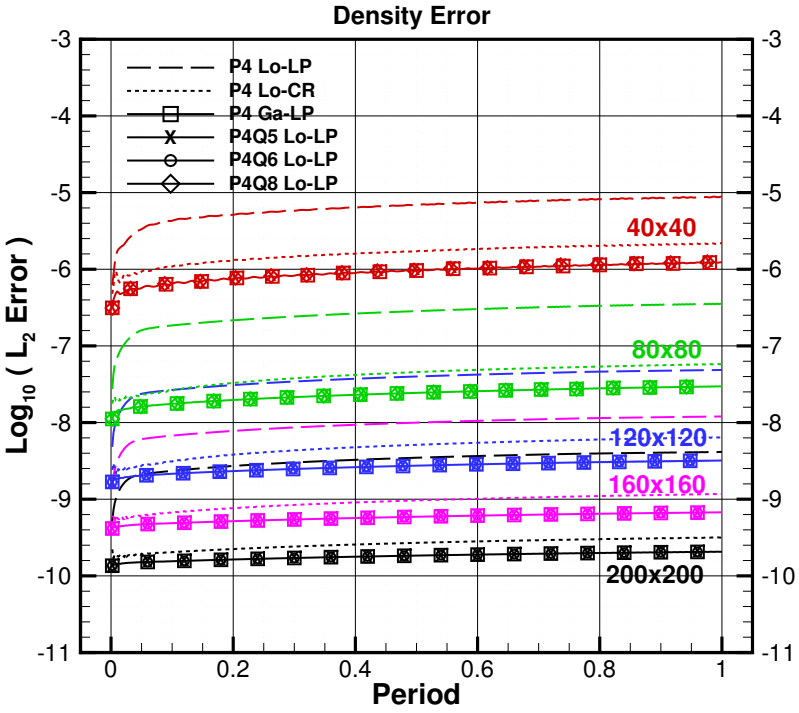


Figure 7: Comparison of $\mathcal{P}4$ solutions with and without over-integration on multiple grid levels.

V.D. Over-Integration Applied to Instabilities

In Ref. 48, some under-resolved cases for the stationary vortex were found that contained instabilities of varying strengths. These instabilities can be split into two categories: "fatal" and "non-fatal" instabilities. A fatal instability is one that caused the simulation to crash due to negative pressure or density. Non-fatal instabilities are those cases that demonstrate instabilities but are still able to finish the simulation without crashing. What we want to see is whether over-integration can suppress these instabilities and provide some additional robustness when applied to an under-resolved simulation. We chose two cases for this test, one exhibiting a fatal instability and the second exhibiting a non-fatal instability.

The case involving a fatal instability uses the Gauss-LP method with a $\mathcal{P}2$ solution on a 40^2 grid and the base result without over-integration is seen in figure 8a as the black line with square symbols. We repeated this test case using the weak form of the DG Spectral Element Method (DGSEM) to ensure that this instability is not something that was only found using the FR method. The DGSEM results are shown as a blue line with diamond symbols and is exactly identical to the FR Gauss-LP results. Also shown in figure 8a are two additional results that use over-integration with the Gauss-LP method: the first over-integrating to $\mathcal{P}2\mathcal{Q}4$ (green line with 'x' symbols) and the second over-integrating to $\mathcal{P}2\mathcal{Q}10$ (red line with circle symbols). In figure 8a, where each of the lines terminates is the point in time at which the simulation blows up. Without over-integration, the simulation blew up after four periods for both of the methods without over-integration. Adding over-integration was not able to improve the stability for this test case. Over-integrating to $\mathcal{P}2\mathcal{Q}4$ slightly reduced the magnitude of the overall error but the simulation still diverged at the same point as the simulations without over-integration. Over-integrating to $\mathcal{P}2\mathcal{Q}10$ reduced the error and allowed to simulation to proceed longer but only for an additional half period.

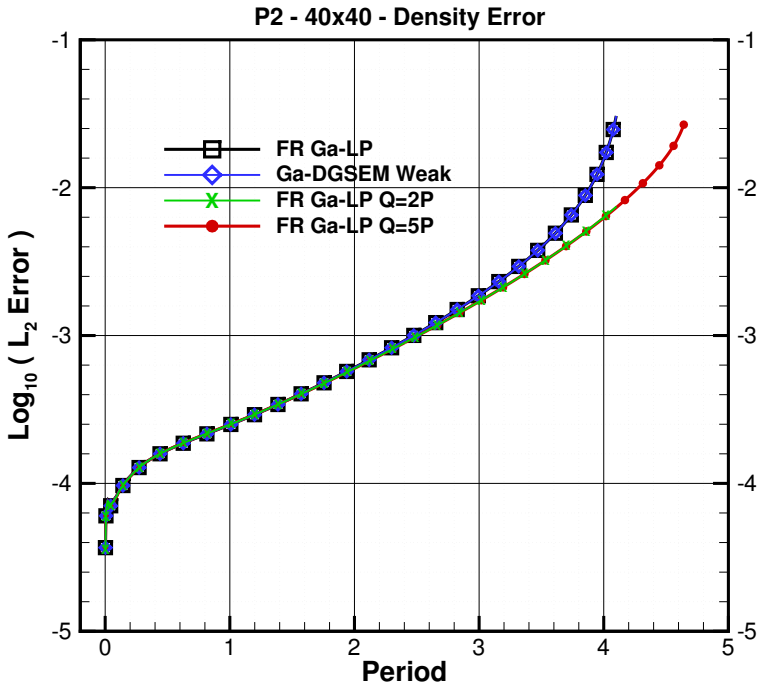
The non-fatal instability case is the same as the previous case except that the grid resolution has been increased to a grid containing 80^2 cells. The base result using the Gauss-LP method is shown in figure 8b as the black line with square symbols. As done with the previous case, we over-integrated to $\mathcal{P}2\mathcal{Q}10$ to ensure we had enough resolution to stabilize the solution. However, over-integration did not improve the stability of the simulation and gave a very similar solution to the simulation without over-integration.

VI. Conclusions

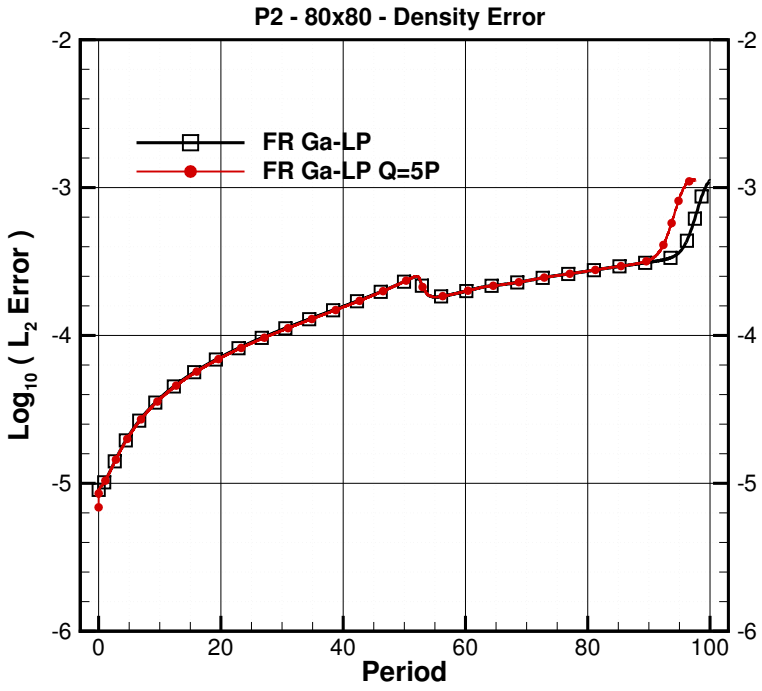
The results presented in this work give conflicting stories on the success of using over-integration as a means of preventing aliasing errors for the FR and DG methods. As shown in sections V.B and V.C, over-integrating indeed improves the quality of the solution, especially for under-resolved simulations and for methods utilizing a reduced order quadrature (e.g., solution points are Lobatto points). However, in section V.D, over-integrating applied to some test cases involving numerical instabilities does not improve the robustness of the FR and DG methods.

Because of the difficulties that we have experienced with the vortex problem,[48] it seems likely that the instabilities we were trying to fix with over-integration in section V.D are not due to aliasing errors. One reason for this conjecture is a small detail from the final jump in error that is found in both simulations shown in figure 8b. These jumps coincide with an exponential fit found in Ref. 48 that appeared to be related to the accumulation of machine roundoff errors caused by simulating a closed system with all periodic boundaries. Conversely, this could also be a coincidence and there is still some sort of aliasing driven instability that has yet to be resolved. Either way, further testing is required to determine the cause of these instabilities that we have encountered with the stationary vortex problem.

The results from section V.C show that over-integration successfully removes any aliasing errors that are inherent to the use of Lobatto quadratures. This is especially promising when looking to use triangular grid cells in 2D and tetrahedra and other unstructured cell types in 3D where Gaussian quadrature rules are not defined. Nodal sets based on a barycentric-type mapping of Lobatto quadrature for triangular elements is popular with many unstructured nodal DG methods. Our results indicate that over-integrating by an order or two might be sufficient in removing most of the aliasing errors found in the Lobatto nodal sets for triangles. In general when the flow is well-resolved, aliasing errors often do not cause numerical instabilities. The focus of future work includes applying over-integration to the Navier-Stokes equations and then to 3D unstructured cells (tetrahedra, pyramids, prisms, and hexahedra geometries).



(a) Over-integration applied to a fatal instability. Where each line terminates is the point that the simulation failed.



(b) Over-integration applied to a non-fatal instability. The $\mathcal{P}2\mathcal{Q}10$ simulation terminates before it reaches 100 periods because it reached the total wall time the job had been allocated and not because it diverged.

Figure 8: Application of different amounts of over-integration to reduce instabilities caused by insufficient resolution.

References

[1]DeBonis, J., "Progress Towards Large-Eddy Simulations for Prediction of Realistic Nozzle Systems," *AIAA Journal of Propulsion and Power*, Vol. 23, No. 5, 2007, pp. 971–980.

[2]Forsythe, J., Squires, K., Wurtzler, K., and Spalart, P., "Detached-Eddy Simulation of Fighter Aircraft at High-Alpha," *Journal of Aircraft*, Vol. 41, No. 2, 2004, pp. 193–200.

[3]Mavripilis, D., Pelaez, J., and Kandil, O., "Large Eddy and Detached Eddy Simulations Using an Unstructured Multigrid Solver," *DNS/LES - Progress and Challenges. Proceedings of the Third AFOSR International Conference on DNS/LES*, Columbus, OH, 2001.

[4]Moreau, S., Christophe, J., and Roger, M., "LES of the Trailing-Edge Flow and Noise of a NACA0012 Airfoil Near Stall," *Proceedings of the Summer Program 2008, Center for Turbulence Research*, Stanford University/NASA Ames, 2008.

[5]Kang, S., Iaccarino, G., Ham, F., and Moin, P., "Prediction of Wall-Pressure Fluctuation in Turbulent Flows with an Immersed Boundary Method," *Journal of Computational Physics*, Vol. 228, No. 18, 2009, pp. 6753–6772.

[6]Huynh, H., "A Flux Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin Methods," AIAA Paper 2007-4079, Jun 2007.

[7]Huynh, H., "A Reconstruction Approach to High-Order Schemes Including Discontinuous Galerkin for Diffusion," AIAA Paper 2009-403, Jan 2009.

[8]Huynh, H., "High-Order Methods Including Discontinuous Galerkin by Reconstructions on Triangular Meshes," AIAA Paper 2011-44, Jan 2011.

[9]Vermeire, B. C., Cagnone, J.-S., and Nadarajah, S., "ILES Using the Correction Procedure via Reconstruction Scheme," AIAA Paper 2013-1001, Jan 2013.

[10]Vermeire, B. C., Nadarajah, S., and Tucker, P. G., "Canonical Test Cases for High-Order Unstructured Implicit Large Eddy Simulation," AIAA Paper 2014-0935, Jan 2014.

[11]Skarolek, V. and Miyaji, K., "Transitional Flow over a SD7003 Wing Using Flux Reconstruction Scheme," AIAA Paper 2014-0250, Jan 2014.

[12]Haga, T., Tsutsumi, S., Kawai, S., and Takaki, R., "Large-Eddy Simulation of a Supersonic Jet Using High-Order Flux Reconstruction Scheme," AIAA Paper 2015-0831, Jan 2015.

[13]Kirby, R. M. and Karniadakis, G. E., "De-Aliasing on Non-Uniform Grids: Algorithms and Applications," *Journal of Computational Physics*, Vol. 191, No. 1, 2003, pp. 249–264.

[14]Kirby, R. M. and Sherwin, S. J., "Aliasing Errors Due to Quadratic Nonlinearities on Triangular Spectral/hp Element Discretisations," *Journal of Engineering Mathematics*, Vol. 56, No. 3, 2006, pp. 273–288.

[15]Kopriva, D. A., "Metric Identities and the Discontinuous Spectral Element Method on Curvilinear Meshes," *Journal of Scientific Computing*, Vol. 26, No. 3, 2006, pp. 301–327.

[16]Hesthaven, J. S. and Warburton, T., *Nodal Discontinuous Galerkin Methods: Algorithms, Analysis, and Applications*, Vol. 54 of *Texts in Applied Mathematics*, Springer New York, 2008.

[17]Castonguay, P., Vincent, P., and Jameson, A., "Application of High-Order Energy Stable Flux Reconstruction Schemes to the Euler Equations," AIAA Paper 2013-686, Jan 2011.

[18]Gassner, G. J. and Beck, A. D., "On the Accuracy of High-Order Discretizations for Underresolved Turbulence Simulations," *Theoretical and Computational Fluid Dynamics*, Vol. 27, No. 3-4, 2013, pp. 221–237.

[19]Karniadakis, G. and Sherwin, S., *Spectral/hp Element Methods for Computational Fluid Dynamics*, Oxford University Press, 2nd ed., 2013.

[20]Williams, D. M. and Jameson, A., "Nodal Points and the Nonlinear Stability of High-Order Methods for Unsteady Flow Problems on Tetrahedral Meshes," AIAA Paper 2013-2830, Jun 2013.

[21]Witherden, F. D. and Vincent, P. E., "An Analysis of Solution Point Coordinates for Flux Reconstruction Schemes on Triangular Elements," *Journal of Scientific Computing*, Vol. TBD, No. TBD, 2014, pp. TBD.

[22]Kanevsky, A., Carpenter, M. H., and Hesthaven, J. S., "Idempotent Filtering in Spectral and Spectral Element Methods," *Journal of Computational Physics*, Vol. 220, No. 1, 2006, pp. 41–58.

[23]Tadmor, E., "Convergence of Spectral Methods for Nonlinear Conservation Laws," *SIAM Journal on Numerical Analysis*, Vol. 26, No. 1, 1989, pp. 30–44.

[24]Maday, Y., Ould Kaber, S. M., and Tadmor, E., "Legendre Pseudospectral Viscosity Method for Nonlinear Conservation Laws," *SIAM Journal on Numerical Analysis*, Vol. 30, No. 2, 1993, pp. 321–342.

[25]Karamanos, G.-S. and Karniadakis, G. E., "A Spectral Vanishing Viscosity Method for Large-Eddy Simulations," *Journal of Computational Physics*, Vol. 163, No. 1, 2000, pp. 22–50.

[26]Guo, B.-y., Ma, H.-p., and Tadmor, E., "Spectral Vanishing Viscosity Method for Nonlinear Conservation Laws," *SIAM Journal on Numerical Analysis*, Vol. 39, No. 4, 2001, pp. 1254–1268.

[27]Kirby, R. M., *Toward Dynamic Spectral/hp Refinement: Algorithms and Applications to Flow-Structure Interactions*, Ph.D. thesis, Brown University, Providence, RI, 2003.

[28]Funaro, D., "Some Remarks about the Collocation Method on a Modified Legendre Grid," *Computers & Mathematics with Applications*, Vol. 33, No. 1, 1997, pp. 95–103.

[29]Funaro, D., *Spectral Elements for Transport-Dominated Equations*, Springer Berlin Heidelberg, 1997.

[30]Vincent, P., Castonguay, P., and Jameson, A., "A New Class of High-Order Energy Stable Flux Reconstruction Schemes," *Journal of Scientific Computing*, Vol. 47, No. 1, 2011, pp. 50–72.

[31]Jameson, A., Castonguay, P., and Vincent, P., "On the Non-Linear Stability of Flux Reconstruction Schemes," *Journal of Scientific Computing*, Vol. 50, No. 2, 2012, pp. 434–445.

[32]De Grazia, D., Mengaldo, G., Moxey, D., Vincent, P. E., and Sherwin, S. J., "Connections Between the Discontinuous Galerkin Method and High-Order Flux Reconstruction Schemes," *International Journal for Numerical Methods in Fluids*, Vol. 75, No. 12, 2014, pp. 860–877.

[33]Yu, M., Wang, Z., and Liu, Y., "On the Accuracy and Efficiency of Discontinuous Galerkin, Spectral Difference and Correction Procedure via Reconstruction Methods," *Journal of Computational Physics*, Vol. 259, 2014, pp. 70–95.

[34]Kopriva, D. A., *Implementing Spectral Methods for Partial Differential Equations: Algorithms for Scientists and Engineers*, Springer Science & Business Media, 2009.

[35]Wang, Z. and Gao, H., "A Unifying Lifting Collocation Penalty Formulation Including the Discontinuous Galerkin, Spectral Volume/Difference Methods for Conservation Laws on Mixed Grids," *Journal of Computational Physics*, Vol. 228, No. 21, 2009, pp. 8161–8186.

[36]Gao, H. and Wang, Z., "A Conservative Correction Procedure via Reconstruction Formulation with the Chain-Rule Divergence Evaluation," *J. Comput. Physics*, Vol. 232, No. 1, 2013, pp. 7–13.

[37]Roe, P. L., "Approximate Riemann Solvers, Parameter Vectors, and Difference Schemes," *Journal of Computational Physics*, Vol. 43, No. 2, 1981, pp. 357–372.

[38]Huynh, H. T., "Accurate Upwind Methods for the Euler Equations," *SIAM Journal on Numerical Analysis*, Vol. 32, No. 5, 1995, pp. 1565–1619.

[39]Gottlieb, S. and Shu, C.-W., "Total Variation Diminishing Runge-Kutta Schemes," *Mathematics of Computation*, Vol. 67, No. 221, 1998, pp. 73–85.

[40]Kopriva, D. A. and Gassner, G., "On the Quadrature and Weak Form Choices in Collocation Type Discontinuous Galerkin Spectral Element Methods," *Journal of Scientific Computing*, Vol. 44, No. 2, 2010, pp. 136–155.

[41]Gassner, G. and Kopriva, D. A., "A Comparison of the Dispersion and Dissipation Errors of Gauss and Gauss-Lobatto Discontinuous Galerkin Spectral Element Methods," *SIAM Journal on Scientific Computing*, Vol. 33, No. 5, 2011, pp. 2560–2579.

[42]Shu, C.-W., "Essentially Non-oscillatory and Weighted Essentially Non-oscillatory Schemes for Hyperbolic Conservation Laws," *Advanced Numerical Approximation of Nonlinear Hyperbolic Equations*, edited by A. Quarteroni, Vol. 1697 of *Lecture Notes in Mathematics*, Springer Berlin Heidelberg, 1998, pp. 325–432.

[43]Wang, Z. J., Liu, Y., May, G., and Jameson, A., "Spectral Difference Method for Unstructured Grids II: Extension to the Euler Equations," *Journal of Scientific Computing*, Vol. 32, No. 1, 2007, pp. 45–71.

[44]Vincent, P., Castonguay, P., and Jameson, A., "Insights from von Neumann Analysis of High-Order Flux Reconstruction Schemes," *Journal of Computational Physics*, Vol. 230, No. 22, 2011, pp. 8134–8154.

[45]Wang, Z., Fidkowski, K., Abgrall, R., Bassi, F., Caraeni, D., Cary, A., Deconinck, H., Hartmann, R., Hillewaert, K., Huynh, H., Kroll, N., May, G., Persson, P.-O., van Leer, B., and Visbal, M., "High-order CFD Methods: Current Status and Perspective," *International Journal for Numerical Methods in Fluids*, Vol. 72, No. 8, 2013, pp. 811–845.

[46]Witherden, F. D., Farrington, A. M., and Vincent, P. E., "PyFR: An Open Source Framework for Solving Advection-Diffusion Type Problems on Streaming Architectures Using the Flux Reconstruction Approach," *Computer Physics Communications*, Vol. 185, No. 11, 2014, pp. 3028–3040.

[47]Yu, M. and Wang, Z. J., "On the Accuracy and Efficiency of Several Discontinuous High-Order Formulations," AIAA Paper 2013-855, Jan 2013.

[48]Spiegel, S., Huynh, H. T., and DeBonis, J. R., "Survey of the Isentropic Euler Vortex Problem Using High-Order Methods," AIAA Paper 2015-TBD, Jun 2015.