High-Order Strand Grid Methods for Low Mach and Incompressible Flows

Jonathan Thorne*, Aaron Katz†, Oisin Tong‡, and Yushi Yanagita§
Department of Mechanical and Aerospace Engineering, Utah State University, Logan, UT 84322

A novel high-order finite volume scheme using flux correction methods in conjunction with structured finite difference schemes is extended to low Mach and incompressible flows on strand grids. Flux correction achieves high order by using correction terms in the fluxes in and out of each finite volume cell. The flux correction method is applied in unstructured layers of the strand grid, the layers are then coupled together using a source term containing the derivatives in the strand direction. Strand-direction derivatives are obtained by using summation-by-parts operators for the first derivative and variable coefficients for the second derivatives. A preconditioner is used to increase the applicability of the method to low Mach and incompressible flows. Verification is completed using the method of manufactured solutions. The method is extended with the introduction of a Spalart Allmaras turbulence model. Results found for low-Mach and incompressible flows and is compared to analytical and experimental data.

I. Introduction

Low Mach number flows present many difficulties to compressible computational fluid dynamics (CFD) algorithms in terms of accuracy and convergence rate. Many of these flows have widely varying particle and acoustic speeds, degrading solution convergence [**?**]. Artificial dissipation using second-order CFD methods often reduces the accuracy of the solution, so even when convergence is achieved the solution could be inaccurate. As an added challenge, complex geometry presents difficulties for mesh generation. Meshing can take days or weeks depending on the complexity of the geometry. Progress addressing these challenges can lead to transformational changes for many applications, such as underwater vehicle design.

First, to improve accuracy and convergence for low Mach and incompressible flows preconditioning is often needed. Preconditioning has also shown to be beneficial in flows that have low speed flow regions, such as in boundary layers or at stagnation points. In addition, preconditioning enables the use of arbitrary state equations providing solutions for real gas effects, multiphase flows, and combustion [**?**].

Second, to reduce numerical diffusion a high-order method is needed. Flux correction (FC) was recently introduced by Katz and Sankaran [**?**] that provides high-order spacial accuracy in conjunction with the need for only first-order derivative terms [**?**, **?**]. High-order accuracy is achieved by correction terms in the flux between nodes, creating a method that has low computational overhead with high-order results. This method has been shown to be third-order on arbitrary triangular meshes for inviscid flows [**?**] and approaches fourth-order in highly viscous flows [**?**].

Third, to address complex geometry, strand meshing has shown great potential. Using a strand mesh has been shown to alleviate many difficulties by being able to fully automate volume grid generation. The strands are automatically generated by using a pointing vector developed from a surface tessellation. The strands are then split into segments creating layers of unstructured mesh layers. Over these layers the high-order method is applied. Strand meshing simplifies the meshing process and increases automation, as well as providing a high-order mesh for the use of high-order methods. In addition, with strand meshing techniques self-satisfying domain connectivity (SSDC) is possible, where each processor in a multi-processor computation has access to the entire domain while only needing minimal information.

By using the methods described solutions for low-Mach and incompressible flows will be explored. This work is outlined such that the first section contains the numerical methods needed to create the preconditioner, the addition of the turbulence model, the discretization and transformations needed for a strand meshing method, and a brief

*Masters Student, AIAA Student Member
†Assistant Professor, AIAA Member
‡PHD Candidate, AIAA Student Member
§Masters Student, AIAA Student Member

description of Flux Correction. The next section provides results using the methods described in the numerical methods portion. Finally, conclusions are found and work for the future work is addressed.

II. Numerical Methods

A. Traditional Preconditioning

Numerous matrix preconditioning methods based on matrix dissipation have already been utilized successfully by other researchers to improve performance of compressible CFD methods in low Mach and incompressible regimes [**?**, **?**, **?**]. We will present traditional framework for most preconditioning schemes in three-dimensions. Following the notation of Merkle, et al. [**?**], the Naiver-Stokes equations without source terms are expressed as

$$\frac{\partial Q}{\partial \tau} + \frac{\partial F_j}{\partial x_j} - \frac{\partial F_j^\nu}{\partial x_j} = 0, \tag{1}$$

where Q represents the matrix of conserved variables, τ represents the pseudo-time, and $F_j = (F, G, H)$, is the inviscid fluxes, and $F_j^\nu = (F^\nu, G^\nu, H^\nu)$ represents the viscous fluxes.

$$Q = \begin{Bmatrix} \rho \\ \rho u_i \\ \rho E \end{Bmatrix}, \quad F_j = \begin{Bmatrix} \rho u \\ \rho u_i u_j + p\delta_{ij} \\ \rho h u_j \end{Bmatrix}, \quad F_j^\nu = \begin{Bmatrix} 0 \\ \sigma_{ij} \\ \sigma_{ij} u_i - q_j \end{Bmatrix}. \tag{2}$$

The variables include ρ, $u_j = (u, v, w)$, E, p and h^0, which represent density, velocity in the j^{th} direction, total energy per unit mass, and enthalpy per unit mass, respectively. δ_{ij} is the chronicler delta and σ_{ij} is the deviatoric stress tensor. q_j is the j^{th} component of the heat flux vector. The stress tensor is defined as

$$\sigma_{ij} = 2\mu \left(S_{ij} - \frac{1}{3} \frac{\partial u_k}{\partial x_k} \delta_{ij} \right), \tag{3}$$

where μ is the dynamic viscosity and S_{ij} is the rate of strain tensor, defined as

$$S_{ij} = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right). \tag{4}$$

Preconditioning methods often begin with a conversion from the conserved variables to primitive variables,

$$Q_v = \begin{Bmatrix} \rho \\ u_j \\ T \end{Bmatrix}, \tag{5}$$

where Q_v is the matrix of primitive variables and T is temperature. This is done to simplify the preconditioning process. Conversion between the primitive variables and the conserved variables are completed via the Jacobian matrix,

$$\Gamma \equiv \frac{\partial Q}{\partial Q_v} = \begin{pmatrix} \rho_p & 0 & 0 & 0 & \rho_T \\ u\rho_p & \rho & 0 & 0 & u\rho_T \\ v\rho_p & 0 & \rho & 0 & v\rho_T \\ w\rho_p & 0 & 0 & \rho & w\rho_T \\ h^0\rho_p + \rho h_p - 1 & \rho u & \rho v & \rho w & h^0\rho_T + \rho h_T \end{pmatrix}, \tag{6}$$

where subscripts p and T denote partial differentiation (e.g. $\rho_p = \frac{\partial \rho}{\partial p}$). The goal of the preconditioner is to reduce wave speeds disparities, provide well poisedness, and ensure diagonal dominance. This is done by replacing the matrix Γ in the pseudo-time derivative term with a preconditioning matrix Γ_p.

$$\Gamma_p = \begin{pmatrix} \rho'_p & 0 & 0 & 0 & \rho_T \\ u\rho'_p & \rho & 0 & 0 & u\rho_T \\ v\rho'_p & 0 & \rho & 0 & v\rho_T \\ w\rho'_p & 0 & 0 & \rho & w\rho_T \\ h^0\rho'_p + \rho h_p - 1 & \rho u & \rho v & \rho w & h^0\rho_T + \rho h_T \end{pmatrix}, \tag{7}$$

it is observed that Γ_p is identical to Γ except for the new ρ'_p term. The selection of ρ'_p is accomplished by forcing the acoustic wave speed to be the same order of magnitude as the particle speeds. Sankaran [**?**] and Merkle, et al. [**?**] both take this approach and note the existence of a preconditioned speed of sound,

$$V_p^2 \equiv \frac{\rho h_t}{d'} \tag{8}$$

where d' is defined as

$$d' \equiv \rho h_T \rho'_p + \rho_T \left(1 - \rho h_p\right). \tag{9}$$

The convergence and dissipation scaling properties of the preconditioner are dependent primarily on the preconditioned speed of sound chosen. Traditionally this is chosen via the particle speed,

$$V_p = \sqrt{(u^2 + v^2 + w^2)}. \tag{10}$$

B. Optimally Defined Preconditioning

In many flows, there exists many different wave speeds. Because Γ_p is a local value, it is possible to make a definition of the preconditioner that is optimally defined such that convergence is improved in the entire domain. This optimally defined preconditioning scheme would be able to switch on and off dependent on the nessesity of the flow, based on the particle speed. This is done through the definition a preconditioned Mach number,

$$\begin{aligned} M_p &= \begin{cases} \sqrt{\frac{2M^2}{1-2M^2}} & M < .5 \\ 1 & M \geq .5 \end{cases}, \\ V_p &= M_p c, \end{aligned} \tag{11}$$

where M_p is the preconditioned Mach number, M is the local Mach number, and c is the speed of sound. With this definition the inviscid eigenvalues are defined as

$$\begin{aligned} \lambda_{1,2} &= \frac{1}{2}\left(u_n\left(1 + M_p^2\right) \pm \left(u_n^2\left(M_p^2 - 1\right)^2 + 4V_p^2\right)\right), \\ \lambda_{3,4,5} &= u_n. \end{aligned} \tag{12}$$

When $M_p > .5$ then the standard compressible eigenvalues are returned, however, when the flow is incompressible the eigenvalues introduce an artificial compressibility constant. This is analogous to the compressibility term introduced by Chorin **??**. With this optimally defined system both incompressible and compressible functionality can be achieved. In addition, convergence will be improved in flows that contain stagnation points, or regions of low-speed flow.

C. Addition of Spalart Allmaras Turbulence Model

Oisin will probably write this...

D. Flux Correction High-Order Method

High-order flux correction is a third-order spatially accurate method developed by Katz and Sankaran []. A more complete derivation is contained in other works [], but for completeness, a brief derivation is included. Flux correction involves adding a correction term to the fluxes, such as its name suggests. Looking at the inviscid fluxes and dissipation,

$$\mathcal{F}_{0i}^h = \frac{1}{2}\left(\mathcal{F}_0 + \mathcal{F}_i\right) - \frac{1}{2}\left|\Gamma^{-1}\mathcal{A}\left(Q_R, Q_L\right)\right|\left(Q_R - Q_L\right). \tag{13}$$

Where $\mathcal{F}$ is the inviscid fluxes doted with the unit normal, 0 and i denote the node, $\mathcal{A} = \partial\mathcal{F}/\partial\mathcal{Q}$ is the directed flux Jacobian, and Q_R and Q_L are the conserved variables at the left and right node. Note that the preconditioning matrix Γ is introduced into the artificial dissipation term. the directed flux Jacobian can be broken into the left and right eigenvectors and the eigenvectors,

$$\left|\Gamma^{-1}\mathcal{A}\right| = M\left|\lambda\right|M^{-1}, \tag{14}$$

where the eigenvalues are given in Equation 12. The right and left flux and dissipation terms are given by

$$\begin{aligned} F_L &= \mathcal{F}_0 + \frac{1}{2}\mathbf{\Delta r}^T \nabla^h, \\ F_R &= \mathcal{F}_i - \frac{1}{2}\mathbf{\Delta r}^T \nabla^h, \\ Q_L &= Q_0 + \frac{1}{2}\mathbf{\Delta r}^T \nabla^h Q_0, \\ Q_R &= Q_i - \frac{1}{2}\mathbf{\Delta r}^T \nabla^h Q_i, \end{aligned} \tag{15}$$

where $\mathbf{r}$ is the position vector along the edge. The gradient is discretized as

$$\nabla^h = \nabla + O(h^p), \tag{16}$$

where p is the order of accuracy, and $O(h^p)$ denotes the truncation error of the gradient procedure. For flux correction, the order of accuracy of the flux and solution gradients must be at least second-order to maintain consistency ($p \geq 2$). The advantage of flux correction over other high-order methods is that it bypasses the need for second derivatives and does not require high-order quadrature. This makes the method an efficient way to achieve third-order accuracy without excess computational expense.

E. Strand Grid Methods

Strand grids show great potential in modern CFD. They provide a method for consistent mesh generation and automation. Strands are produced by crating a surface tessellation from which straight lines (strands) are extruded emanating from the surface. This creates the potential for easy scalability, as the strands are developed from the surface structure and emanated out. These strands are then broken into levels generating surfaces in the mesh. This can be seen in Figure 1, where each clipping index defines a new surface mesh.

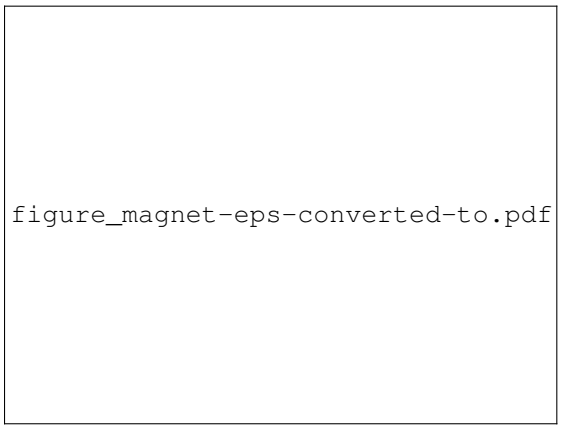

Figure 1. Example of strand mesh extrusion

After initial mesh generation, each level of the mesh is then transformed into the computational domain, seen in Figure 2. In the computational space $\Delta\eta = 1/(N-1)$. Upon transformation to computational space Equation 1 becomes

$$\frac{\partial \hat{Q}}{\partial t} + \frac{\partial \hat{F}}{\partial r} + \frac{\partial \hat{G}}{\partial s} + \frac{\partial \hat{H}}{\partial \eta} - \frac{\partial \hat{F}^\nu}{\partial r} - \frac{\partial \hat{G}^\nu}{\partial s} - \frac{\partial \hat{H}^\nu}{\partial \eta} = \hat{S} \tag{17}$$

where $\hat{Q}$, $\hat{S}$, $\hat{F}$, $\hat{G}$, $\hat{H}$, $\hat{F}^\nu$, $\hat{G}^\nu$, and $\hat{H}^\nu$ and the transformation matrix are defined as,

$$\hat{Q} \equiv JQ, \quad \hat{S} \equiv JS,$$
$$\hat{F} \equiv J\left(r_x F + r_y G + r_z H\right), \quad \hat{F}^\nu \equiv J\left(r_x F^\nu + r_y G^\nu + r_z H^\nu\right),$$
$$\hat{G} \equiv J\left(s_x F + s_y G + s_z H\right), \quad \hat{G}^\nu \equiv J\left(s_x F^\nu + s_y G^\nu + s_z H^\nu\right),$$
$$\hat{H} \equiv J\left(\eta_x F + \eta_y G + \eta_z H\right), \quad \hat{H}^\nu \equiv J\left(\eta_x F^\nu + \eta_y G^\nu + \eta_z H^\nu\right),$$
$$\begin{pmatrix} r_x & s_x & \eta_x \\ r_y & s_y & \eta_y \\ r_z & s_z & \eta_z \end{pmatrix} = \frac{1}{J}\begin{pmatrix} y_s z_\eta - z_s y_\eta & z_r y_\eta - y_r z_\eta & y_r z_s - z_r y_s \\ z_s x_\eta - x_s y_\eta & x_r z_\eta - z_r x_\eta & z_r x_s - x_r z_s \\ x_s y_\eta - y_s x_\eta & y_r x_\eta - x_r y_\eta & x_r y_s - y_r x_s \end{pmatrix},$$
$$J = x_\eta\left(y_r z_s - z_r y_s\right) + y_\eta\left(z_r x_s - x_r z_s\right) + z_\eta(x_r y_s - y_r x_s).$$

Here, J is the Jacobian of the transformation, and partial differentiation is denoted with a subscript (e.g. $\partial x/\partial s = x_s$).

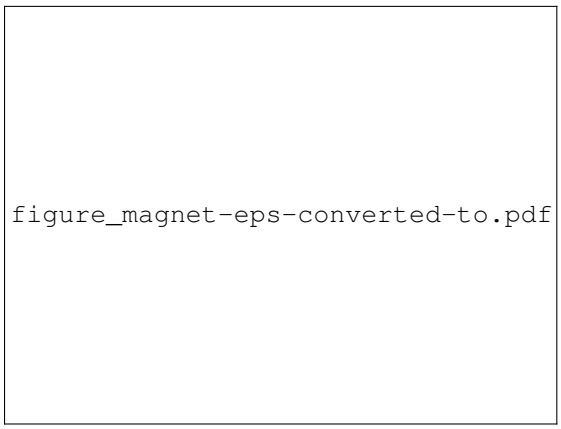

Figure 2. Transformation between physical and computational domains

After transformation, each strand layer is then connected by using high-order finite differences based on summation-by-parts along the η direction for the first derivatives then uses variable coefficients for the second derivatives. Along the r-s plane high-order FC will be implemented. The r-s plane is then combined with the η direction by a source term. This source term can be found by rearranging Equation 17 and moving the η-derivatives to the left side of the equation with the source terms, resulting in

$$\frac{\partial \hat{Q}}{\partial \tau} + \frac{\partial \hat{F}}{\partial r} + \frac{\partial \hat{G}}{\partial s} - \frac{\partial \hat{F}^\nu}{\partial r} - \frac{\partial \hat{G}^\nu}{\partial s} = \tilde{S}, \tag{18}$$

$$\tilde{S} \equiv \hat{S} - \frac{\partial \hat{Q}}{\partial t} - \frac{\partial \hat{H}}{\partial \eta} + \frac{\partial \hat{H}^\nu}{\partial \eta}.$$

Because of this transformation done in Equation 18, the problem is simplified to essentially a two dimensional problem combined with source terms. This becomes convenient when the method is later parallelized when solving the problem, as it creates a situation of SSDC. Because of the simplicity of the mesh and the need to only do computations in the two-dimensional domain each processor can have access to its own mesh. Processor interactions are minimized by only needing the source terms, increasing computational speed with multiple processors.

III. Results

A. Analytical Verification

1. MMS Verification

Me

2. Creeping Flow

Me

B. Experimental Verification

1. Reynolds Sweep with Sphere

Me

2. Turbulent Low-Mach Flow over Body 1

Yushi

3. Turbulent Incompressible Flow over Model 4155

Yushi

4. Turbulent Incompressible Flow over Model 4159

Yushi

IV. Conclusions