Efficient and Flexible Solution Strategies for Large-Scale, Strongly Coupled Multi-Physics Analysis and Optimization Problems

by

James Westfall

B.S., University of Dayton, 2002

M.S., Air Force Institute of Technology, 2007

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Aerospace Engineering Sciences

2016

This thesis entitled:
Efficient and Flexible Solution Strategies for Large-Scale, Strongly Coupled Multi-Physics
Analysis and Optimization Problems
written by James Westfall
has been approved for the Department of Aerospace Engineering Sciences

_____

Dr. Kurt Maute

_____

Dr. Carlos Felippa

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the
content and the form meet acceptable presentation standards of scholarly work in the above
mentioned discipline.

Westfall, James (Ph.D., Aerospace Engineering)

Efficient and Flexible Solution Strategies for Large-Scale, Strongly Coupled Multi-Physics Analysis and Optimization Problems

Thesis directed by Dr. Kurt Maute

Aerospace problems are characterized by strong coupling of different disciplines, such as fluid-structure interactions. There has been much research over the years on developing numerical solution methods tailored to each of the different disciplines. The classical approach to solving these strongly coupled systems is to stitch together these individual solvers by solving for one discipline and using the solution as boundary conditions for the successive disciplines. In more recent years, research has focused on numerical methods that handle solving coupled disciplines together. These methods offer the potential of better computational efficiency. These coupled solution methods range from monolithic solution strategies to decoupled partitioned strategies. This research develops a flexible finite element analysis tool which is capable of analyzing a range of aerospace problems including highly coupled incompressible fluid-structure interactions and turbulent compressible flows. The goal of this research is to access the viability of streamline-upwind Petrov-Galerkin (SUPG) finite element analysis for compressible turbulent flows. Additionally, this research uses a selection of nonlinear solution methods, linear solvers, iterative preconditioners, varying degrees of coupling, and coupling strategies to provide insight into the computational efficiency of these methods as they apply to turbulent compressible flows and incompressible fluid-structure interaction problems. The results suggest that SUPG finite element analysis for compressible flows may not be robust enough for optimization problems due to ill-conditioned matrices in the linear approximation. This research also shows that it is the degree of coupling and criticality of the coupling that drives the selection of the most efficient nonlinear and linear solution methods.

Dedication

To my loving wife Kala, thank you for all of your support and encouragement and for taking care of everything else for me so that I could concentrate on this. To my daughter Charlee, thank you for your sacrifice while I was working on this.

Acknowledgements

Contents

Tables

Table

Figures

Figure

Algorithms

Algorithm

Chapter 1

Introduction

The aerospace industry is heavily invested in optimization techniques which aim at assisting engineers in designing novel aerospace systems. They need to be accurate, robust, easy to use, computationally efficient and able to handle problems of realistic engineering interest for these methods to be viable design tools. There has been much research in accurately modeling different disciplines associated with aerospace systems. These disciplines can include compressible and incompressible fluid dynamics, turbulence, finite deformation of structures, electrostatic, magnetic, radiation, and thermal phenomena, among others. This research focuses on using finite element analysis to solve turbulent compressible flow and incompressible fluid-structure interaction (FSI) problems. Robust and efficient methods have been developed to solve the particular systems of equations, developed through numerical analysis, for each of these disciplines. Those methods have been specifically tailored for those numerical systems.

Over the past two decades, there has been a focus on the interactions between different disciplines. The coupling of these disciplines can pose particular challenges to the numerical analysis. Optimization routines can often require many analyses and therefore only increase the need for robust, accurate and efficient numerical methods. New numerical methods have been an area of heavy research in order to solve coupled analysis and include the analysis in an optimization framework.

There are two major classes that these numerical methods typically fall into. That is a monolithic approach and a segregated or partitioned approach. The partitioned approaches partition the problem into different subsystems, often by discipline, and solve each subsystem separately. The

manner in which the different subsystems are coupled is one thing that distinguishes the various partitioned approaches. Monolithic approaches do not partition the analysis into subsystems, but rather solve all equations simultaneously. The monolithic approach is appealing because it aims at achieving quadratic convergence associated with Newton's method when using a consistent linear approximation. Unfortunately, Newton's method requires that the full linearization be built and stored each iteration. The monolithic approach typically has a larger memory requirement than segregated methods because of this. Also, the solution of the linear subproblem tends to take more computational effort because all unknowns are included in each linear solution. This approach has been adopted for fluid-structure interactions (FSI) and fluid-structure-thermal interactions for compressible flow discretized by finite elements by Howard[33]. Heil has applied the monolithic method to FSI problems using incompressible fluid dynamics, also discretized by finite elements[28]. The monolithic approach can use a consistent linearization of the system in which all pieces are included. The more flexible the structure is, the more highly coupled the FSI system is and this can lead to very ill-conditioned systems. In his work, Heil employed a partially coupled, approximate Jacobian with and without a pressure Schur complement approximation as a preconditioner to help facilitate the efficient solution of the ill-conditioned system. This preconditioning scheme was proven to help alleviate some of the computational burden in successive solutions of the linearized system[28]. The monolithic approach can also use an approximate linearization of the system in which parts or all of the blocks which couple the fluid and structure are not considered. This is a quasi-Newton method which can result in more Newton iterations, but less difficulty in solving the linear subproblems. There is also less computational effort in building the linear subproblems for this scheme. Two different modified Newton approaches were employed by Ghattas and Li to incompressible flow[25].

Farhat et al.[23] developed a three-field formulation for FSI problems. One field is defined for the fluid equations, one for the structural equations, and one which governed the fluid mesh motion. This three-field formulation helped facilitate the development of segregated approaches. Just like the monolithic approach, there are both weakly coupled and strongly coupled segregated approaches for solving a set of equations associated with multiple fields. In the strongly coupled

approach, the linearization of the equations includes all terms, including those blocks which couple the different fields. In the weakly coupled approaches these coupling blocks are left out, leaving the coupling in the nonlinear residual only. At the most basic, a segregated approach is one in which each field is individually solved. One distinct advantage of this type of approach is that it can utilize different preconditioners and linear solvers for each field of equations. This allows the tailored and already well developed solution methods to be applied. Another advantage is that only a portion of the full problem is being solved at any one time. So, this means each linear subproblem is smaller in size, requiring less memory and less computational effort than the monolithic approach. Also, for the strongly coupled approaches, the coupling blocks, in general, do not need to be explicitly built and stored. Another advantage of segregated approaches is that each field can be discretized via different methods. Segregated approaches also allow for using different software tools for each field. The segregated methods do have disadvantages as well. Strong coupling in a system can cause a solution to diverge, and weakly coupled schemes can often be less stable and less accurate for transient problems[28]. A classic segregated approach is to employ nonlinear block Gauss-Seidel(NLBGS) methods. NLBGS approaches have been employed by Maute et al.[55], Maute and Allen[54], Maute et al.[56] and others to the Euler equations discretized by finite volume methods. More recently Barcelos and Maute[11] have applied NLBGS to compressible flows to include the modeling of turbulence. This work made use of finite volumes for the fluid analysis and was limited to subsonic flows only.

Ideally one would like to combine the convergence rates of a monolithic approach with flexibility of domain specific solvers of a segregated approach. This was the aim of Kim et al.[43] with their improved multi-level Newton method. In this method, there is a subsystem for each field of equations which is solved for using Newton's method. There is also an outer Newton loop which solves a monolithic system for the solution updates. Another method for achieving Newton-like performance is Schur-Newton-Krylov method developed by Barcelos et al.[10] for Euler flow using finite volume methods. As the name implies, this method uses the Newton method to to solve a Schur complement formulation of the FSI problem and the linear subproblem is solved via a

Krylov method. This method boasts improved robustness and efficiency over the standard NLBGS methods.

The choice of which method to use depends on several factors. The first factor is whether the software is being used for analysis or optimization. The monolithic approach is particularly attractive for gradient based optimization routines because the Jacobian matrix, or first-order partial derivatives, developed during the Newton method can be reused to calculate the sensitivities for the objective function and constraints. Often times only one nonlinear algorithm is implemented, limiting the user to just the one choice. Implementing other nonlinear algorithms can often times require significant changes to the core of the software which discourages or prevents a user from adopting that method. Another factor is what disciplines are being solved. Numerical analysis for each field produces linear or nonlinear systems with specific characteristics. A method that is good for one collection of analysis types may not be good or optimal for another. The number of fields also influences the choice of methods. Maute et al.[56] have noted that going from a two-field FSI problem to a three-field FSI problem can increase the computational complexity by 25%. Another factor is problem size. Some methods are better suited for small problems while others can easily handle larger ones. Ideally one would like to be able to pick and choose different solution methods for the analysis, or parts of the analysis, and for the sensitivity analysis. Having the ability to use each of these algorithms in the same framework, same discretization method and on the same problems of interest enables the direct comparison of the various methods. This facilitates the comparison of accuracy, convergence rates and overall efficiency.

The goal of this work is to extend the numerical methods briefly described above to include turbulent compressible fluid analysis in a finite element discretization and make direct comparisons between different nonlinear and linear solvers. This provides valuable insight into the suitability of particular solution strategy configurations for various classes of problems. The choice of using finite element discretization is driven by the desire to do turbulent compressible fluid-structure interaction optimization problems using gradient based optimization routines. Developing a consistent analytical first-order derivative of the finite element equations may provide a computationally

efficient method for computing optimization sensitivities. The conclusion of this research is that the finite element compressible computational fluid dynamics method developed here is not robust enough to be used in an optimization routine, nor is it extensible to large scale problems of interest. Due to the lack of robustness of the compressible methods, an incompressible FSI problem is also used. The FSI problem is used to make comparisons among nonlinear and linear solvers. It is also used to highlight the contrast with the compressible flow analysis. All work in this research is accomplished using the Finite Element Multi-Disciplinary Optimization Code (FEMDOC) developed at University of Colorado Boulder, Center for Aerospace Structures. It also makes use of Trilinos[29] developed at Sandia National Laboratories. The specific Trilinos packages used are Epetra for linear algebra, Amesos[64, 65] and AztecOO for linear solver interfaces, and IFPACK[63] and ML[24] for linear system preconditioning.

The rest of this document is organized as follows. First, the implementation of the compressible turbulent fluid equations in FEMDOC is thoroughly discussed. This includes the stabilization methods, the shock capturing methods, the development of an auxiliary field for increased accuracy and stability, and the specific implementation of the one-equation Spalart-Allmaras turbulence model. The fluid-structure coupling strategies are discussed next. This includes a comparison of how the different coupling formulations can impact the computational efficiency. Following that, three nonlinear solution methods are discussed. They are compared for efficiency using a compressible turbulent flow problem and an incompressible FSI problem. Next, linear solvers are discussed and compared for efficiency using the same test cases. Finally, conclusions as to the viability of compressible finite element analysis and efficiency of nonlinear and linear solution methods is discussed. The nomenclature associated with the development of equations and algorithms is listed at the end of this document. The reader is referred to this as a reference throughout the document.

Chapter 2

Compressible Turbulent Computational Fluid Dynamics by Finite Element

This research focuses on subsonic and supersonic compressible turbulent external flow analysis. This analysis is made possible by the following steps. The full Navier-Stokes and turbulence equations are discretized and made weak via the Galerkin finite element method. The streamline upwind Petrov-Galerkin(SUPG) stabilization method is applied, choosing one of several stabilization parameters to scale the upwind bias. Next, one of several shock capturing operators are applied. The resulting discretized equations are then integrated over the fluid domain via Gauss integration. The particulars of these steps will be discussed in the following sections.

2.1   Navier Stokes Equations

The Navier-Stokes equations consist of partial differential equations representing the conservation of mass, Equation 2.1, one equation for each spatial dimension representing the conservation of momentum, Equation 2.2, and the conservation of energy, Equation 2.4.

These equations include a correction to the inviscid flux to account for the mesh motion in an Arbitrary Lagrange-Eulerian (ALE) formulation. For aeroelastic analysis, the structural problem can be formulated in a Lagrange reference frame while the fluid is formulated in an ALE reference frame. The mesh associated with the fluid is then allowed to move, accounting for the structural deformation. The fluid mesh velocities, v_i^m, can be accounted for through the inviscid flux correction term seen in each of the conservation equations. For the details of this implementation, the reader

is referred to Howard's PhD thesis[32]. All equations use the Einstein notation for indices.

$$\frac{\partial \rho}{\partial t} + \frac{\partial \rho v_i}{\partial x_i} - v_i^m \frac{\partial \rho}{\partial x_i} = S^c \tag{2.1}$$

$$\frac{\partial(\rho v_j)}{\partial t} + \frac{\partial}{\partial x_i}\left(\rho v_j v_i + p\delta_{ij}\right) - v_i^m \frac{\partial \rho v_j}{\partial x_i} = \frac{1}{Re}\frac{\partial \tau_{ij}}{\partial x_i} + S_i^m \tag{2.2}$$

$$\tau_{ij} = \mu\left(\frac{\partial v_i}{\partial x_j} + \frac{\partial v_j}{\partial x_i}\right) - \frac{2}{3}\mu\delta_{ij}\left(\frac{\partial v_k}{\partial x_k}\right) \tag{2.3}$$

$$\frac{\partial(\rho E)}{\partial t} + \frac{\partial}{\partial x_i}\left(\rho E v_i + p v_i\right) - v_i^m \frac{\partial \rho E}{\partial x_i} = \frac{1}{Re}\left(\frac{\partial(\tau_{ij}v_j)}{\partial x_i} - \frac{\partial q_i}{\partial x_i}\right) + S^e \tag{2.4}$$

$$q_i = -\kappa\frac{\partial T}{\partial x_i} \tag{2.5}$$

The momentum equations use the deviatoric stress tensor defined in Equation 2.3, which makes the assumption of a Newtonian fluid for the bulk viscosity. The energy equation uses the heat flux vector defined in Equation 2.5. The thermal conductivity, κ, is calculated assuming a constant Prandtl number as shown in Equation 2.6.

$$\kappa = \frac{c_p \mu}{Pr} \tag{2.6}$$

To close the Navier-Stokes equations, this work uses an ideal gas assumption, as shown in Equations 2.7, and Sutherland's law for the viscosity, shown in Equation 2.8.

$$p = \rho R T, \qquad c_v = \frac{R}{\gamma - 1}, \qquad c_p = \frac{\gamma R}{\gamma - 1} \tag{2.7}$$

$$\mu_l(T) = \mu_{ref}\frac{T^{\frac{3}{2}}}{T + T_{ref}} \tag{2.8}$$

For simplicity, these equations can be represented in vector form and rearranged to define the strong form of the residual for the fluid equations, as seen in Equation 2.9.

$$\mathbf{R}_f(\mathbf{U}) \equiv \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} - v_i^m \frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial \mathbf{G}_i}{\partial x_i} - \mathbf{S} = \mathbf{0} \tag{2.9}$$

In the vector form of the Navier-Stokes equations, the vector of conservative variables, $\mathbf{U}$, the inviscid flux vector, $\mathbf{F}_i$, and viscous flux vector, $\mathbf{G}_i$, can be represented in the following manner. The Reynolds number has been included for cases where the nondimensional form of the equations are used, otherwise this value is 1.0.

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho v_j \\ \rho E \end{pmatrix}, \quad \mathbf{F}_i(\mathbf{U}) = \begin{pmatrix} \rho v_i \\ \rho v_i v_j + p\delta_{ij} \\ \rho E v_i + p v_i \end{pmatrix}, \quad \mathbf{G}_i(\mathbf{U}) = \frac{1}{Re} \begin{pmatrix} 0 \\ \tau_{ij} \\ \tau_{ij} v_j - q_i \end{pmatrix} \tag{2.10}$$

The source vector, $\mathbf{S}$, in Equation 2.9 is typically a null vector, but could be non-zero if accounting for body forces or in instances such as reacting flows. In this work, it is a null vector except for when the turbulence equations are added as discussed next.

2.2   Spalart-Allmaras One-Equation Turbulence Model

There are several methods for modeling turbulence. These include direct numerical simulation in which the domain is resolved and solved down to the smallest turbulent length scales. This method results in a large number of degrees of freedom (DOFs) and is consequently computationally expensive. Another method is large eddy simulation in which the large eddies are resolved and the small eddies below the prescribed length scale are modeled. Finally, another class of methods is the Reynolds-averaged Navier-Stokes (RANS) models. In this class of turbulence models, the Navier-Stokes equations are solved for the mean flow with a Reynolds stress term added to account for the velocity perturbations. How the Reynolds stress terms are calculated is what distinguishes between the different RANS models. The particular RANS model used in this work is the Spalart-Allmaras model.

The Spalart-Allmaras model is a primarily heuristic one-equation model for turbulent eddy viscosity[68]. This model uses the Boussinesq hypothesis which amounts to adding the turbulent viscosity to the laminar viscosity in the Navier-Stokes equations. This model is a relatively low computational expense to model turbulence. It works well for modeling many problems of engineering interest, specifically external flow aerospace applications. For these reasons, it was chosen as

the model of choice for this work. There are many modifications to the Spalart-Allmaras equation to increase robustness and include different physical characteristics of turbulent flow. For completeness, the original equation and definitions are included here. This equation is solved for the transported variable $\tilde{\nu}$, which is eddy viscosity.

$$\frac{\partial \tilde{\nu}}{\partial t} + v_j \frac{\partial \tilde{\nu}}{\partial x_j} = c_{b1}(1 - f_{t2})\tilde{S}\tilde{\nu} - \left[c_{w1}f_w - \frac{c_{b1}}{\kappa^2}f_{t2}\right]\left(\frac{\tilde{\nu}}{d}\right)^2$$
$$+ \frac{1}{\sigma}\left[\frac{\partial}{\partial x_j}\left((\nu + \tilde{\nu})\frac{\partial \tilde{\nu}}{\partial x_j}\right) + c_{b2}\frac{\partial \tilde{\nu}}{\partial x_i}\frac{\partial \tilde{\nu}}{\partial x_i}\right] \tag{2.11}$$

In Equation 2.11, the variables are defined as shown below, with d being the distance to the nearest wall.

$$\chi = \frac{\tilde{\nu}}{\nu} \qquad f_{v1} = \frac{\chi^3}{\chi^3 + c_{v1}^3} \qquad f_{v2} = 1 - \frac{\chi}{1 + \chi f_{v1}} \qquad f_{t2} = c_{t3}\exp\left(-c_{t4}\chi^2\right)$$

$$W_{ij} = \frac{1}{2}\left(\frac{\partial v_i}{\partial x_j} - \frac{\partial v_j}{\partial x_i}\right) \qquad \Omega = \sqrt{2W_{ij}W_{ij}} \qquad \tilde{S} = \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2}f_{v2}$$

$$r = \min\left[\frac{\tilde{\nu}}{\tilde{S}\kappa^2 d^2}, 10\right] \qquad g = r + c_{w2}(r^6 - r) \qquad f_w = g\left[\frac{1 + c_{w3}^6}{g^6 + c_{w3}^6}\right]^{1/6}$$

The constants in Equation 2.11, are defined as:

$$c_{b1} = 0.1355 \quad \sigma = 2/3 \quad c_{b2} = 0.622 \quad \kappa = 0.41 \qquad c_{w2} = 0.3$$
$$c_{w3} = 2 \quad c_{v1} = 7.1 \quad c_{t3} = 1.2 \quad c_{t4} = 0.5 \quad c_{w1} = \frac{c_{b1}}{\kappa^2} + \frac{1 + c_{b2}}{\sigma}$$

Finally the turbulent eddy viscosity in the fluid equations can be calculated using:

$$\mu_t = \rho\tilde{\nu}f_{v1} \tag{2.12}$$

2.3 Compressible Spalart-Allmaras

Equation 2.11 was developed for incompressible flows. There are different ways to adapt it for compressible flows. Several methods were explored[17, 69], but the greatest success was found

using that of Wervaecke et al.[77, 78], which can be seen in Equation 2.13. Shown here it includes the Reynolds number for non-dimensional runs, otherwise it is set to 1.0.

$$
\begin{aligned}
\frac{\partial \rho \tilde{\nu}}{\partial t} + \frac{\partial \rho \tilde{\nu} v_j}{\partial x_j} \;=\; & c_{b1} \bar{S} \rho \tilde{\nu} - \frac{1}{Re} c_{w1} f_w \rho \left(\frac{\tilde{\nu}}{d} \right)^2 \\
& + \frac{1}{\sigma Re} \frac{\partial}{\partial x_j} \left(\mu_l \frac{\partial \tilde{\nu}}{\partial x_j} + \sqrt{\rho \tilde{\nu}} \frac{\partial \sqrt{\rho \tilde{\nu}}}{\partial x_j} \right) + \frac{c_{b2}}{\sigma Re} \frac{\partial \sqrt{\rho \tilde{\nu}}}{\partial x_i} \frac{\partial \sqrt{\rho \tilde{\nu}}}{\partial x_i}
\end{aligned}
\tag{2.13}
$$

One problem that can occur when using the Spalart-Allmaras equation is that the transported variable can go negative, which can cause slow convergence or even divergence. There are many different ways suggested to keep the turbulent variable strictly positive. The method used here adds a term, f_{v3}. This method allows for the variable to go slightly negative, maintaining positive turbulent eddy viscosity, and then tend toward zero. The method can be found on NASA's Turbulence Modeling Resource web page[1], but it otherwise is not found in any official publications. It is reported to have an "odd effect on transition at low Reynolds numbers[69]," but this has not been observed during this research. This method is chosen to keep the turbulence variable positive because several were initially utilized and it is the most effective method. The implementation of that modification and the use of non-dimensional equations requires the redefinition of the terms below:

$$
\tilde{S} = f_{v3} \Omega + \frac{\tilde{\nu}}{\kappa^2 d^2 Re} f_{v2}
$$

$$
f_{v2} = \frac{1}{(1 + \chi/c_{v2})^3} \qquad f_{v3} = \frac{(1 + \chi f_{v1})(1 - f_{v2})}{\chi} \qquad c_{v2} = 5 \qquad r = \frac{\tilde{\nu}}{\tilde{S} \kappa^2 d^2 Re}
$$

Equation 2.13 does not include the trip term, f_{t2} seen in the original Equation 2.11. In this work, the flow is considered to be fully turbulent and therefore a trip term does not need to be used. The boundary conditions for the Spalart-Allmaras equation are shown in Equation 2.14.

$$
\rho \tilde{\nu}_{wall} = 0 \qquad 3 \mu_{l\infty} \leq \rho \tilde{\nu}_{\infty} \leq 5 \mu_{l\infty}
\tag{2.14}
$$

The vector form of the Navier-Stokes residual, Equation 2.9, can then be extended to include the Spalart-Allmaras equation by redefining the vectors as seen below:

$$\mathbf{U} = \begin{pmatrix} \rho \\ \rho v_j \\ \rho E \\ \rho \tilde{\nu} \end{pmatrix}, \quad \mathbf{F}_i(\mathbf{U}) = \begin{pmatrix} \rho v_i \\ \rho v_i v_j + p \delta_{ij} \\ \rho E v_i + p v_i \\ \rho \tilde{\nu} v_i \end{pmatrix} \tag{2.15}$$

$$\mathbf{G}_i(\mathbf{U}) = \frac{1}{Re} \begin{pmatrix} 0 \\ \tau_{ij} \\ (\tau_{ij} v_j - q_i) \\ \frac{1}{\sigma}\left(\mu_l \frac{\partial \tilde{\nu}}{\partial x_i} + \sqrt{\rho}\tilde{\nu} \frac{\partial \sqrt{\rho}\tilde{\nu}}{\partial x_i} \right) \end{pmatrix} \tag{2.16}$$

$$\mathbf{S}(\mathbf{U}) = \begin{pmatrix} 0 \\ 0 \\ 0 \\ c_{b1}\bar{S}\rho\tilde{\nu} - \frac{1}{Re}c_{w1}f_w\rho\left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{c_{b2}}{\sigma Re}\frac{\partial \sqrt{\rho}\tilde{\nu}}{\partial x_i}\frac{\partial \sqrt{\rho}\tilde{\nu}}{\partial x_i} \end{pmatrix} \tag{2.17}$$

Given that Spalart-Allmaras is a turbulent eddy viscosity model, the dynamic viscosity in Equation 2.3 is updated to include the turbulent eddy viscosity. The new dynamic viscosity is defined by adding Equations 2.8 and 2.12, seen in Equation 2.18.

$$\mu = \mu_l(T) + \mu_t = \mu_{ref}\frac{T^{\frac{3}{2}}}{T + T_{ref}} + \rho\tilde{\nu}f_{v1} \tag{2.18}$$

2.4   SUPG Weak Formulation

Following standard finite element formulation, the strong form of the residual in Equation 2.9 is then made weak by multiplying by a test function and integrated over the domain, resulting in the weak form of the Navier-Stokes and Spalart-Allmaras equations, seen in Equation 2.19.

$$\int_\Omega \hat{\mathbf{W}}^T \left[\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} - v_i^m \frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial \mathbf{G}_i}{\partial x_i} - \mathbf{S} \right] d\Omega = \mathbf{0} \tag{2.19}$$

The standard test functions are then modified to apply an upwind bias, which is needed to stabilize the numerical solution. The SUPG method is chosen because it is touted as being accurate, efficient and one of the most popular stabilization methods[7]. In the SUPG formulation, the test functions are modified with the formulation seen in Equation 2.20.

$$\hat{\mathbf{W}} = \mathbf{W} + \boldsymbol{\tau}_{supg}\mathbf{A}_i^T\frac{\partial \mathbf{W}}{\partial x_i} \tag{2.20}$$

$$\mathbf{A}_i = \frac{\partial \mathbf{F}_i}{\partial \mathbf{U}} \tag{2.21}$$

Substituting Equation 2.20 into Equation 2.19 results in the vector form of the weak SUPG formulation, seen in Equation 2.22.

$$\int_\Omega \left(\mathbf{W} + \boldsymbol{\tau}_{supg}\mathbf{A}_i^T\frac{\partial \mathbf{W}}{\partial x_i}\right)^T \left[\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} - v_i^m\frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial \mathbf{G}_i}{\partial x_i} - \mathbf{S}\right] d\Omega = \mathbf{0} \tag{2.22}$$

The Galerkin terms and the SUPG terms are then separated and integration by parts is done on the Galerkin flux divergence terms. This results in a volume integral and a boundary integral for the Galerkin terms, seen in Equation 2.23.

$$\sum_{e=1}^{n_e} \int_{\Omega^e} \mathbf{W}^T\left(\frac{\partial \mathbf{U}}{\partial t} - v_i^m\frac{\partial \mathbf{U}_i}{\partial x_i} - \mathbf{S}\right) + \frac{\partial \mathbf{W}^T}{\partial x_i}\left(\mathbf{G}_i - \mathbf{F}_i\right) d\Omega^e - \int_\Gamma \mathbf{W}^T\left(\mathbf{G}_i - \mathbf{F}_i\right)\hat{n}_i \, d\Gamma$$

$$+ \sum_{e=1}^{n_e} \int_{\Omega^e} \frac{\partial \mathbf{W}^T}{\partial x_k}\mathbf{A}_k\boldsymbol{\tau}_{supg}\left[\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} - v_i^m\frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial \mathbf{G}_i}{\partial x_i} - \mathbf{S}\right] d\Omega^e = \mathbf{0} \tag{2.23}$$

Finally, for high speed flows, it is necessary to add isotropic artificial numerical diffusion in the region of high gradients. This is done in order to capture shocks in the flow. There are several forms of this shock capturing operator which are discussed later. Adding in one example of a shock capturing term, results in the final weak form of the residual as seen in Equation 2.24.

$$\sum_{e=1}^{n_e} \int_{\Omega^e} \mathbf{W}^T\left(\frac{\partial \mathbf{U}}{\partial t} - v_i^m\frac{\partial \mathbf{U}_i}{\partial x_i} - \mathbf{S}\right) + \frac{\partial \mathbf{W}^T}{\partial x_i}\left(\mathbf{G}_i - \mathbf{F}_i\right) d\Omega^e - \int_\Gamma \mathbf{W}^T\left(\mathbf{G}_i - \mathbf{F}_i\right)\hat{n}_i \, d\Gamma$$

$$+ \sum_{e=1}^{n_e} \int_{\Omega^e} \frac{\partial \mathbf{W}^T}{\partial x_k}\mathbf{A}_k\boldsymbol{\tau}_{supg}\left[\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} - v_i^m\frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial \mathbf{G}_i}{\partial x_i} - \mathbf{S}\right] d\Omega^e$$

$$+ \sum_{e=1}^{n_e} \int_{\Omega^e} \delta\left(\frac{\partial \mathbf{W}^T}{\partial x_i}\frac{\partial \mathbf{H}}{\partial x_i}\right) d\Omega^e = \mathbf{0} \tag{2.24}$$

Here the shock capturing operator works on the total enthalpy rather than the total energy, as shown in Equation 2.25.

$$\mathbf{H} = \begin{pmatrix} \rho \\ \rho v_j \\ \rho H \\ \rho \tilde{\nu} \end{pmatrix} = \begin{pmatrix} \rho \\ \rho v_j \\ \rho E + p \\ \rho \tilde{\nu} \end{pmatrix} \tag{2.25}$$

2.5 Stabilization Matrix

The stabilization matrix, $\boldsymbol{\tau}_{supg}$, is a matrix that scales the amount of upwind bias for each equation. There has been much research into the formulation of this matrix[37, 47, 71, 16]. Hughes et al.[36] provides an excellent historical summary of the development of the SUPG method. The formulations used in this work are among the latest developments in the SUPG scaling. These are particularly attractive because the matrix is a diagonal matrix of the form in Equation 2.26, shown for three dimensional cases.

$$\boldsymbol{\tau}_{supg} = \mathrm{diag}\left(\tau_c, \tau_m, \tau_m, \tau_m, \tau_e, \tau_t\right) \tag{2.26}$$

2.5.1 Stabilization Matrix τ_{BK}

The first stabilization formulation is one that is used by Kirk et al.[45, 44]. This scheme allows for more stabilization as the time step gets larger, the local velocity gets smaller, the local viscosity gets smaller, the local density gets larger, or the local thermal conductivity gets smaller. The variable, $h_{\mathbf{v}}$, is defined in Equation 2.30 and is the characteristic element length scale aligned along the flow direction. The element length scale is used to add more stabilization in the areas of coarser grids.

$$\tau_c = \left[\left(\frac{2}{\Delta t}\right)^2 + \left(\frac{2\left(\|\mathbf{v}\| + c\right)}{h_{\mathbf{v}}}\right)^2\right]^{-1/2} \tag{2.27}$$

$$\tau_m = \left[\left(\frac{2}{\Delta t} \right)^2 + \left(\frac{2 \left(\|\mathbf{v}\| + c \right)}{h_{\mathbf{v}}} \right)^2 + \left(\frac{4\mu}{\rho h_{\mathbf{v}}^2} \right)^2 \right]^{-1/2} \tag{2.28}$$

$$\tau_e = \left[\left(\frac{2}{\Delta t} \right)^2 + \left(\frac{2 \left(\|\mathbf{v}\| + c \right)}{h_{\mathbf{v}}} \right)^2 + \left(\frac{4\kappa}{\rho c_p h_{\mathbf{v}}^2} \right)^2 \right]^{-1/2} \tag{2.29}$$

$$h_{\mathbf{v}} = 2 \left(\sum_{n=1}^{n_n} | \hat{\mathbf{v}} \cdot \nabla N^n | \right)^{-1} \tag{2.30}$$

2.5.2   Stabilization Matrix τ_{SB}

The second stabilization matrix used is one by Bova et al.[13]. This stabilization removes the dependency on the time increment and also reduces the stream-wise stabilization in places where there is isotropic numerical diffusion added by the shock capturing parameter, shown here as δ. This implementation also has a different definition of length scale than Kirk et al.

$$\tau_c = \left[\left(\frac{\|\mathbf{v}\| + c}{h_{\mathbf{v}}} \right)^2 + (\delta)^2 \right]^{-1/2} \tag{2.31}$$

$$\tau_m = \left[\left(\frac{\|\mathbf{v}\| + c}{h_{\mathbf{v}}} \right)^2 + (\delta)^2 + \left(\frac{\mu}{\rho h_{\mathbf{v}}^2} \right)^2 \right]^{-1/2} \tag{2.32}$$

$$\tau_e = \left[\left(\frac{\|\mathbf{v}\| + c}{h_{\mathbf{v}}} \right)^2 + (\delta)^2 + \left(\frac{\kappa}{\rho c_p h_{\mathbf{v}}^2} \right)^2 \right]^{-1/2} \tag{2.33}$$

$$h_{\mathbf{v}} = \mathcal{C} \sqrt{\frac{v_k v_k}{v_i g_{ij} v_j}} \tag{2.34}$$

$$g_{ij} = \frac{\partial \xi_k}{\partial x_i} \frac{\partial \xi_k}{\partial x_j} = [g^{ij}]^{-1} \tag{2.35}$$

Here the length scale makes use of the co-variant metric tensor, g_{ij}, and a constant, $\mathcal{C}$, which is determined by the range of the parametric coordinates used in the element. For $-1 \leq \xi_i \leq 1$, $\mathcal{C} = 2$, or for $0 \leq \xi_i \leq 1$, then $\mathcal{C} = 1$.

2.5.3   Turbulence Stabilization

The turbulence equation requires its own stabilization. The stabilization parameter for the turbulence equation is taken directly from Hauke[27] for an advection-diffusion-reaction equation.

$$\tau_t = \frac{1}{\frac{2\kappa}{m_k h^2} Pe_2 + \mid s \mid Pe_1} \tag{2.36}$$

$$Pe_1 = \max\left(1.0, \frac{2\kappa}{m_k h^2 \mid s \mid}\right) \tag{2.37}$$

$$Pe_2 = \max\left(1.0, \frac{m_k \|\boldsymbol{v}\| h}{\kappa}\right) \tag{2.38}$$

$$h = \sqrt{\frac{1}{n_d} \sum_{i=1}^{n_d} g^{ii}} \tag{2.39}$$

$$\kappa = \frac{\mu_l + \mu_t}{\rho} \tag{2.40}$$

$$s = c_{b1} \bar{S} \rho \tilde{\nu} - \frac{1}{Re} c_{w1} f_w \rho \left(\frac{\tilde{\nu}}{d}\right)^2 + \frac{c_{b2}}{\sigma Re} \frac{\partial \sqrt{\rho \tilde{\nu}}}{\partial x_i} \frac{\partial \sqrt{\rho \tilde{\nu}}}{\partial x_i} \tag{2.41}$$

The source in Equation 2.41 includes the production, destruction and what is occasionally considered part of the diffusion term, c_{b2}. This has been included here as it does not fit into the advection or diffusion flux framework and therefore is more like a production term. For bi-linear elements used in this work the parameter m_k is 1/3.

2.6   Shock Capturing Parameter

There are several methods to define a shock capturing parameter. All of the methods aim at adding minimal local numerical diffusion in the presence of large solution gradients. This is used in order to capture shocks over a few elements while improving numerical stability. Each method tries to add the minimum needed numerical diffusion so as to maintain accurate results. Three

methods used in this research will be discussed in the following sections. Each of the following three methods are modified to include the complete strong form of the residual, so that where the strong form is satisfied the shock capturing should vanish. Also, to evaluate the strong form of the residual, the divergence of the viscous fluxes is calculated using nodally reconstructed viscous fluxes as discussed later.

2.6.1 Shock Capturing Parameter ν

The first shock capturing parameter is originally taken from Shakib et al[66] and modified by Bova et al.[13]. The version used by Bova et al. is the one used in this work. It has been modified to include the ALE correction and the turbulence source term found in the strong form of the residual in the numerator. The term added to the weak form of the fluid residual is as follows:

$$\sum_{e=1}^{n_e} \int_{\Omega^e} \nu \left(\frac{\partial \mathbf{W}^T}{\partial x_i} g^{ij} \frac{\partial \mathbf{H}}{\partial x_j} \right) d\Omega^e \tag{2.42}$$

The shock capturing parameter is then defined as:

$$\nu = \left[\frac{\left\| \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} - v_i^m \frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial \mathbf{G}_i}{\partial x_i} - \mathbf{S} \right\|_{\mathbf{A}_0^{-1}}}{\|\Delta \mathbf{U}\|_{\mathbf{A}_0^{-1}} + g^{ij} \frac{\partial U}{\partial x_i} \mathbf{A}_0^{-1} \frac{\partial U}{\partial x_j}} \right]^{1/2} \tag{2.43}$$

$$\Delta \mathbf{U} = \frac{\partial \mathbf{U}}{\partial t} \Delta t \tag{2.44}$$

Here it is necessary to define the entropy operator seen in Equation 2.45, where $\mathbf{A}_0^{-1}$ is the transformation matrix from entropy variables to conservative variables and e is the internal energy.

$$\| \cdot \|_{\mathbf{A}_0^{-1}} = \{\cdot\}^T \mathbf{A}_0^{-1} \{\cdot\} \tag{2.45}$$

$$
\mathbf{A}_0^{-1} = \frac{1}{\rho}
\begin{bmatrix}
\gamma - \left(\frac{v_x^2 + v_y^2 + v_z^2}{2e} \right)^2 & \frac{(v_x^2 + v_y^2 + v_z^2)v_x}{2e^2} & \frac{(v_x^2 + v_y^2 + v_z^2)v_y}{2e^2} & \frac{(v_x^2 + v_y^2 + v_z^2)v_z}{2e^2} & 1 - \frac{v_x^2 + v_y^2 + v_z^2}{2e^2} \\
& \left(\frac{v_x}{e} \right)^2 + \frac{1}{e} & \frac{v_x v_y}{e^2} & \frac{v_x v_z}{e^2} & \frac{v_x}{e^2} \\
& & \left(\frac{v_y}{e} \right)^2 + \frac{1}{e} & \frac{v_y v_z}{e^2} & \frac{v_y}{e^2} \\
& \text{symmetric} & & \left(\frac{v_z}{e} \right)^2 + \frac{1}{e} & \frac{v_z}{e^2} \\
& & & & \left(\frac{1}{e^2} \right)
\end{bmatrix}
\tag{2.46}
$$

2.6.2 Shock Capturing Parameter δ

This shock capturing parameter was originally defined by Hughes et al.[35], adapted for conservative variables by Aliabadi and Tezduyar[3] and later adapted by Kirk et al.[44, 45]. The version adapted by Kirk et al. is the one used here, again with the modification to the strong form of the residual to include the ALE correction and turbulence source terms for completeness. The term added to the weak form of the fluid residual is as follows:

$$
\sum_{e=1}^{n_e} \int_{\Omega^e} \delta \left(\frac{\partial \mathbf{W}^T}{\partial x_i} \frac{\partial \mathbf{H}}{\partial x_i} \right) d\Omega^e
\tag{2.47}
$$

The shock capturing parameter is defined as:

$$
\delta = \left[\frac{\left\| \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} - v_i^m \frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial \mathbf{G}_i}{\partial x_i} - \mathbf{S} \right\|_{\mathbf{A}_0^{-1}}}{\| \nabla \xi \cdot \nabla \mathbf{U} \|_{\mathbf{A}_0^{-1}} + \| \nabla \eta \cdot \nabla \mathbf{U} \|_{\mathbf{A}_0^{-1}} + \| \nabla \zeta \cdot \nabla \mathbf{U} \|_{\mathbf{A}_0^{-1}}} \right]^{1/2}
\tag{2.48}
$$

2.6.3 Shock Capturing Parameter $\mathbf{YZ}\beta$

The so called $\mathbf{YZ}\beta$ shock capturing parameter by Tezduyar and Senga[72] has several different implementations. This method is of particular interest, because of the simplicity. Not only is it simple to calculate, but the derivatives are also much simpler to calculate analytically than the other two methods. The term added to the weak form of the residual is as follows:

$$
\sum_{e=1}^{n_e} \int_{\Omega^e} \nu \left(\frac{\partial \mathbf{W}^T}{\partial x_i} \frac{\partial \mathbf{H}}{\partial x_i} \right) d\Omega^e
\tag{2.49}
$$

The shock capturing parameter is defined as:

$$\nu = \left\| \mathbf{Y}^{-1}\mathbf{Z} \right\| \left(\sum_{i=1}^{n_{sd}} \left\| \mathbf{Y}^{-1}\frac{\partial \mathbf{U}}{\partial x_i} \right\|^2 \right)^{\frac{\beta}{2}-1} \left(\frac{h_{shock}}{2} \right)^{\beta} \tag{2.50}$$

$\mathbf{Z}$ is defined to be the complete strong form of the fluid residual, again a modification from that presented by Tezduyar and Senga.

$$\mathbf{Z} = \frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_i}{\partial x_i} - v_i^m \frac{\partial \mathbf{U}}{\partial x_i} - \frac{\partial \mathbf{G}_i}{\partial x_i} - \mathbf{S} \tag{2.51}$$

$\mathbf{Y}$ is defined to be a diagonal matrix with reference values for each degree of freedom. The freestream values are used as the reference values in this work.

$$\mathbf{Y} = \text{diag} \left(\ (\mathbf{U}_1)_{ref}, \quad (\mathbf{U}_2)_{ref}, \quad \cdots \quad , \quad (\mathbf{U}_n)_{ref} \ \right) \tag{2.52}$$

This parameter makes use of a length scale different from the one used in the stabilization method. This uses a density gradient aligned element length scale.

$$h_{shock} = 2 \left(\sum_{n=1}^{n_n} | \ \mathbf{j} \cdot \nabla N^n \ | \right)^{-1}, \qquad \mathbf{j} = \frac{\nabla \rho}{\| \nabla \rho \|} \tag{2.53}$$

A value of $\beta = 1$ can be used for smoother shocks while $\beta = 2$ for stronger shocks.

$$\nu = \frac{1}{2} \left((\nu)_{\beta=1} + (\nu)_{\beta=2} \right) \tag{2.54}$$

The work in this research makes use of Equation 2.54 and strictly uses the average of the two, which is hard coded, resulting in Equation 2.55.

$$\nu = \frac{h_{shock}}{4} \left\| \mathbf{Y}^{-1}\mathbf{Z} \right\| \left(\left(\sum_{i=1}^{n_d} \left\| \mathbf{Y}^{-1}\frac{\partial \mathbf{U}}{\partial x_i} \right\|^2 \right)^{-\frac{1}{2}} + \frac{h_{shock}}{2} \right) \tag{2.55}$$

2.7   Nodal Reconstruction

The SUPG stabilized term in the final weak form of the residual, Equation 2.24, contains the divergence of the viscous fluxes. This poses a problem, particularly for the bi-linear elements used in this work. The viscous flux contains a first derivative in the deviatoric stress term. Taking

the divergence of the viscous flux results in a second order derivative, for which all but the mixed terms vanish for bi-linear elements. One common solution is to ignore this term all together. Another solution is to use higher order shape functions which allow for a better second derivative calculation. Jansen et al.[39] suggest to do a nodal reconstruction of the viscous fluxes and then apply the divergence operator to the nodally reconstructed viscous fluxes when integrating over the element. This is the option used in this work.

Nodal reconstruction is used in this work for other parameters as well. Kirk[44] suggests that a discontinuous shock capturing field can allow jump discontinuities in the conservative variables. Enforcing a continuous shock capturing field can lead to better stabilization. Bova et al.[13] make a similar argument for the element length scale, used in the stabilization, yielding better results for a smooth field and stabilization matrices calculated at the nodes. For consistency and application of the same arguments made for the fluid equations, a nodal reconstruction is done for turbulence as well. This is done for the turbulence diffusive flux in each spatial direction. The turbulence stabilization parameter itself is reconstructed at the nodes instead of the length scale. This is done because the source terms in the stabilization are also dependent on gradients, so reconstructing the stabilization parameter requires fewer variables. The nodal reconstruction can be accomplished through a global L_2 projection method as seen in the following equations, where ϕ is any reconstructed quantity.

$$\mathbf{R}_p = \sum_{e=1}^{n_e} \int_{\Omega^e} \mathbf{N}^T (\mathbf{N}\phi - \phi_{gp}) \, d\Omega^e = 0 \tag{2.56}$$

Equation 2.56 states that the value of ϕ calculated at a given Gauss point is equal to the value interpolated from the nodal values. Recognizing the mass matrix shown in Equation 2.57, it can be substituted into Equation 2.56.

$$\mathbf{M} = \mathbf{N}^T \mathbf{N} \tag{2.57}$$

$$\mathbf{R}_p = \sum_{e=1}^{n_e} \int_{\Omega^e} \mathbf{M}\phi - \mathbf{N}^T \phi_{gp} \, d\Omega^e = 0 \tag{2.58}$$

For a true global L_2 projection the full mass matrix, $\mathbf{M}$, must be used, but this can lead to negative values for an otherwise strictly positive field. This is a problem, particularly for the length scale and the shock capturing parameter. So, instead a lumped mass matrix is used to solve this problem.

2.8 Evaluation of Terms

There is a choice of how and where to calculate and build each of the terms in Equation 2.24. Bova et al. have suggested that building the inviscid fluxes, their Jacobians, and the stabilization matrices at the nodes and then interpolating these quantities to the quadrature points leads to better stability[13]. This is what is done in this research as shown in the following equations:

$$\mathbf{F}_i = \sum_{n=1}^{n_n} N_n \mathbf{F}_i(\mathbf{U}_n), \qquad \frac{\partial \mathbf{F}_i}{\partial x_i} = \sum_{n=1}^{n_n} N_{n,i} \mathbf{F}_i(\mathbf{U}_n) \tag{2.59}$$

$$\tau_{supg} = \sum_{n=1}^{n_n} N_n \tau_{supg}(\mathbf{U}_n, \phi_n), \qquad \mathbf{A}_i = \sum_{n=1}^{n_n} N_n \frac{\partial \mathbf{F}_i(\mathbf{U}_n)}{\partial \mathbf{U}_n} \tag{2.60}$$

The viscous flux is calculated and built at the quadrature point from interpolated conservative and primitive variables as shown in Equation 2.61. The primitive variables are those variables that can be derived from the conservative variables and the constitutive model. The primitive variables include velocities, pressure, temperature and internal energy. These primitive variables are necessary to compute the finite element residual and Jacobian. Primitive variables are used in the Navier-Stokes equations, Equations 2.2 thru 2.8, the Spalart-Allmaras turbulence equation, Equation 2.13, the stabilization matrix, Equations 2.27 thru 2.34, and the shock capturing operators, Equations 2.43, 2.48, and 2.50. The gradients used to calculate the deviatoric stress and heat flux in the viscous fluxes are calculated by applying the gradient operator to nodal values of primitive variables.

$$\mathbf{G}_i = \mathbf{G}_i(\mathbf{N}^T \mathbf{U}) \tag{2.61}$$

The gradients of the conservative variables and enthalpy variables are built by applying the gradient operator at the quadrature point to nodal values as shown in Equation 2.62.

$$\frac{\partial \mathbf{U}}{\partial x_i} = \sum_{n=1}^{n_n} N_{n,i} \mathbf{U}(\mathbf{U}_n), \qquad \frac{\partial \mathbf{H}}{\partial x_i} = \sum_{n=1}^{n_n} N_{n,i} \mathbf{H}(\mathbf{U}_n) \qquad (2.62)$$

As discussed earlier, the divergence of the viscous fluxes used in the strong form of the fluid residual is calculated by applying the divergence operator to the nodally reconstructed viscous fluxes. The mesh velocities are interpolated from nodal mesh velocities.

$$\frac{\partial \mathbf{G}_i}{\partial x_i} = \sum_{n=1}^{n_n} N_{n,i}(\phi_i)_n, \qquad v_i^m = \sum_{n=1}^{n_n} N_n(v_i^m)_n \qquad (2.63)$$

All quantities using nodal reconstruction are interpolated from the nodally reconstructed values.

$$\phi_i = \sum_{n=1}^{n_n} N_n(\phi_i)_n \qquad (2.64)$$

Finally, the contravariant metric tensor is calculated from the node coordinates and the derivative of the natural coordinates evaluated at the quadrature point.

$$g^{ij} = g^{ij}\left(\mathbf{x}_n, (\boldsymbol{\xi}_{,k})_{gp}\right) \qquad (2.65)$$

2.9 Adaptive Time Stepping

In computational fluid dynamics, adaptive time stepping is often used, particularly in a quasi-steady-state analysis. Adaptive time stepping is used to increase the size of the time step from a small time step, used to get the flow started, to a larger time step once the flow is developed. There are two time stepping schemes used in this work.

The first time stepping scheme used comes from the work of Gresho et al.[26, 42], originally applied to incompressible fluid dynamics. The time step is chosen based on Equation 2.66 for first-order accurate time integration.

$$\Delta t^{n+1} = \Delta t^n \left(2\frac{\epsilon}{d^{n+1}}\right)^{0.5} \qquad (2.66)$$

$$d^{n+1} = \frac{\left\| \mathbf{U}_p^{n+1} - \mathbf{U}_0^{n+1} \right\|_2}{\sqrt{n_{\text{dof}}} \left\| \mathbf{U} \right\|_\infty} \tag{2.67}$$

Here, n_{dof}, is the number of DOFs. The superscripts are for the times step number, n, and the subscript is for the nonlinear iteration number, p. The only user input for this time stepping scheme is, ϵ, which is a maximum for the "relative norm of the error for the next step[26]."

The second time stepping scheme used is based on the Courant-Friedrichs-Lewy (CFL) condition. This is a necessary but not sufficient condition for numerical stability for explicit time integration schemes. The time integration used in this work is the implict backward Euler scheme. Implicit schemes do not have the same restrictions on time step size for numerical stability as the explicit schemes, but the CFL number can still be used to control the growth of the time step. For this time stepping scheme an initial CFL number is chosen and linearly increased. At each time step, the time step size is chosen based on Equation 2.68.

$$\Delta t^n = \min_{1}^{n_{\text{nodes}}} \left(\frac{h_v}{||\mathbf{v}|| + c} \right) \text{CFL}^n \tag{2.68}$$

The quantity inside the minimum is calculated at each node. Here, h_v is the flow aligned length scale, $\mathbf{v}$ is the velocity vector, and c is the local speed of sound. The time step is then chosen by taking the minimum of the nodally calculated quantity multiplied by the target CFL number for that time step, denoted in Equation 2.68 as CFL^n.

2.10 Model Verification

The implementation of the SUPG compressible fluid equations and Spalart-Allmaras equation discussed above identifies several methods to increase stability and accuracy. Any deviation from previously published work is also identified. Each of these modifications and methods are easily turned off in the flexible environment that FEMDOC provides. It is the experience of the author that those methods identified above result in the most accurate and robust solution method. A few examples are shown below in order to verify that the methods are implemented correctly.

2.10.1 Subsonic Turbulent Flat Plate

The first example is a subsonic turbulent flat plate, seen in Figure 2.1. This example comes from NASA's Turbulence Modeling Resource web page[1]. The only difference is the analysis done here is with slightly different boundary conditions. All quantities are prescribed at the inlet and the top boundary. Nothing is prescribed at the outlet.
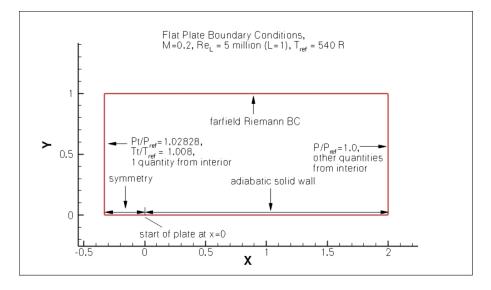
Figure 2.1: NASA turbulent flat plate setup[1]

The website provides five different structured meshes. Starting with the coarsest mesh, each consecutive mesh is created by cutting each element in half in both directions. The data provided is taken from the most refined mesh and is given for NASA's finite volume solver, CFL3D. While the CFL3D results are based on the most refined mesh (544 x 384 elements), the second most coarse mesh (68 x 48 elements) is used for the FEMDOC results. This is done to show that FEMDOC is able to achieve acceptable results from a much more coarse mesh. Table 2.1 lists the analysis parameters used. This example utilizes a partitioned nonlinear solver and as such the monolithic linear system is partitioned into subsystems by degree of freedom type as identified in Table 2.1. The partitioned nonlinear solvers are further discussed in Chapter 4.

Table 2.1: Analysis parameters used for the subsonic turbulent flat plate

Num of Procs	6
Time Solver	Backwards Euler
Time Stepping Scheme	CFL
Starting CFL	0.05
Max CFL	200.0
Stabilization	$\boldsymbol{\tau}_{SB}$
Shock Capturing	ν
Nonlinear Solver	Staggered Newton
Subsystem 1	
DOFs	$\mathbf{G}_i^n$
Relaxation	1.0
Linear Solver	MUMPS
Subsystem 2	
DOFs	$h_\mathbf{v}, \nu, \tau_t$
Relaxation	1.0
Linear Solver	MUMPS
Subsystem 3	
DOFs	$\rho\tilde{\nu}$
Relaxation	0.8
Linear Solver	MUMPS
Subsystem 4	
DOFs	$\rho, \rho v_i, \rho E$
Relaxation	0.8
Linear Solver	MUMPS
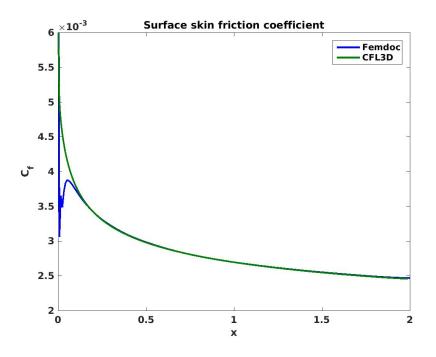
Figure 2.2: Coefficient of friction for flat plate[1]

The leading edge friction peak is not captured quite as well on the coarse mesh, but the rest of the plate yields the same results as seen in Figure 2.2. The leading edge of the plate is better captured using a more refined mesh as would be expected. The velocity profile is plotted at a particular x location of 0.97. These results are nearly identical.
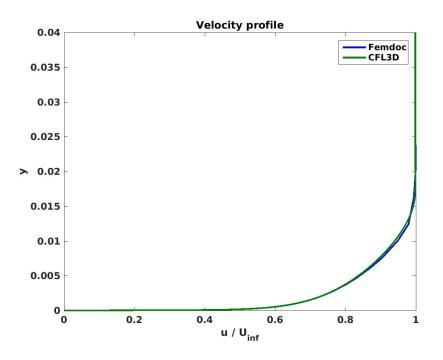
Figure 2.3: Velocity profile $x = 0.97$ for flat plate[1]

The next set of results require the definition of non-dimensional velocity, u^+, and non-dimensional wall distance, y^+.

$$u^* = \sqrt{\frac{c_f 0.5 \rho_\infty v_\infty^2}{\rho}} \tag{2.69}$$

$$u^+ = \frac{\|\mathbf{v}\|}{u^*} \tag{2.70}$$

$$y^+ = \frac{u^* \rho \, d}{\mu_l} \tag{2.71}$$

The coefficient of friction appears in Equation 2.69. The coefficient of friction could be calculated and nodally averaged as a pre-processing step, but this is computationally expensive. As an alternative and keeping with common practice, an estimation of the coefficient of friction is used. There are many choices as to how to estimate it, but the one chosen in this work uses the empirically calibrated 1/7 power law for turbulent boundary layers[22], shown in Equation 2.72.

$$c_f = 0.0594 Re_\infty^{-\frac{1}{5}} \tag{2.72}$$

Shown in Figure 2.4 on a logarithmic scale for y^+, the profiles match exactly. This is dependent on the estimation for the coefficient of friction. For a different model or even a different constant in the same model, different results could be shown. The important point here is that the profile matches that of NASA's CFL3D.
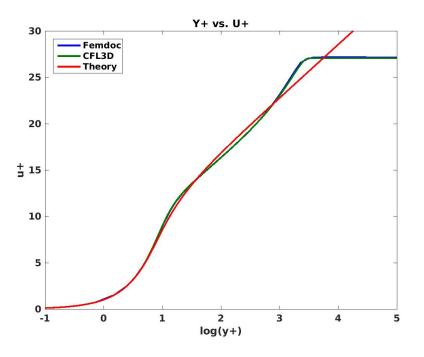
Figure 2.4: u^+ vs. y^+ $x = 0.97$ for NASA flat plate[1]

2.10.2   Subsonic Turbulent Backward Facing Step

The second example is a subsonic turbulent backward facing step depicted in Figure 2.5. This example also comes from NASA's Turbulence Modeling Resource web page[1]. The boundary conditions applied are the same as published on NASA's website and as shown in Figure 2.5. The results here are not only given for CFL3D code, but for experimental data as well. Thus, this example serves as a validation of the turbulence model as well as a verification of correct implementation.
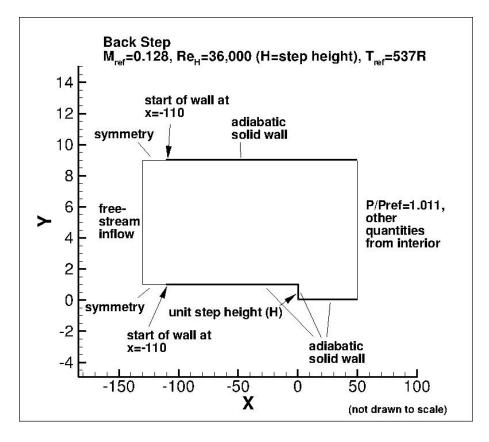
Figure 2.5: Backward facing step setup[1]

As in the last example, the NASA website provides five different meshes for the back step, but in this case the results of the CFL3D code are provided for the second finest mesh. As before, the second coarsest mesh is used to do the analysis with FEMDOC. In other words, the mesh used for FEMDOC has elements with a characteristic length four times that of the CFL3D results. Table 2.2 provides the parameters for this analysis.

Table 2.2: Analysis parameters used for the subsonic turbulent backward facing step

Num of Procs	6
Time Solver	Backwards Euler
Time Stepping Scheme	Gresho
Gresho ϵ	1.0
Max Time Step Increase	10%
Stabilization	$\boldsymbol{\tau}_{SB}$
Shock Capturing	ν
Nonlinear Solver	Monolithic Newton
Relaxation	0.2
Linear Solver	UMFPACK

The coefficient of pressure plots in Figure 2.6 are shifted such that the C_p downstream at $x/H = 40$ is 0.0. This was done for the data reported from CFL3D and is likewise done for FEMDOC data to make an accurate comparison.
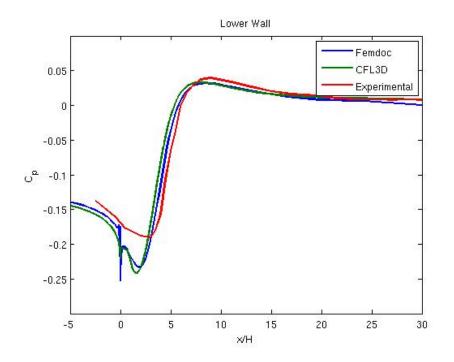
Figure 2.6: Coefficient of pressure for backward facing step[1]

Figure 2.6 shows that again FEMDOC is able to produce nearly the same results as CFL3D with four times as coarse a mesh. This also shows that for a large part of the area of interest around

the step, FEMDOC produces slightly better results when compared to experimental, but for the most part they are very similar. In the case of the coefficient of friction shown in Figure 2.7, the results match well between the two codes, although here CFL3D does capture the friction leading up to the step better. FEMDOC does accurately predict the reattachment point.
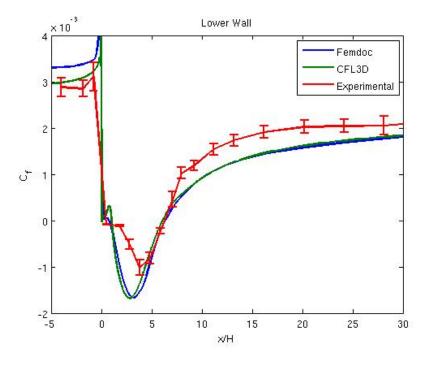
Figure 2.7: Coefficient of friction for backward facing step[1]

For further comparison, the results for Wervaecke et al.[78] are shown because it is another SUPG finite element code, for a more direct comparison. These results also include direct numerical simulation results and results for a mixed finite element and finite volume implementation. The C_p plot here is not shifted as in Figure 2.6. The results here are very similar to FEMDOC and compare quite well with the direct numerical simulation.

(a) Pressure coefficient

(b) Friction coefficient
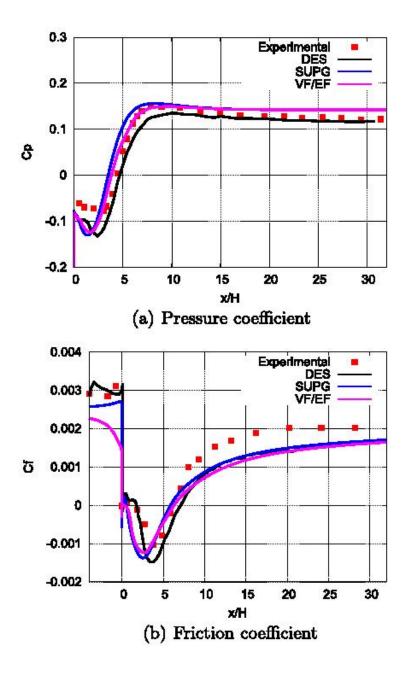
Figure 2.8: Backward facing step results by Wervaecke et al.[78]

2.10.3   Euler Supersonic Compression Corner

The next example is a Mach 2.0 flow into a $10°$ compression corner with no turbulence, published by Tezduyar and Senga[72]. This example is used because it is a supersonic flow. The domain used is a square mesh with 20 x 20 quadrilateral elements. The left and top edges are inlet conditions with all quantities prescribed. The wall on the bottom of the domain has an Euler

slip-condition enforced. The right edge is an outlet with nothing prescribed or enforced.
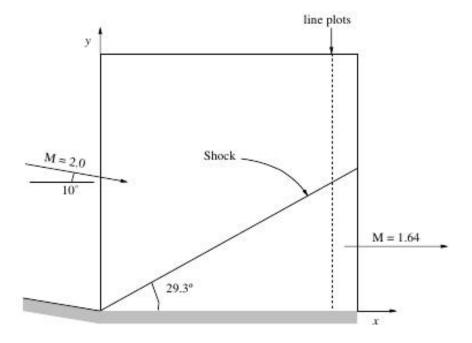
Figure 2.9: Tezduyar supersonic compression corner setup[72]

The parameters chosen for the supersonic compression corner are found in Table 2.3.

Table 2.3: Analysis parameters used for the supersonic compression corner

Num of Procs	6
Time Solver	Backwards Euler
Time Stepping Scheme	None
Stabilization	$\boldsymbol{\tau}_{BK}$
Shock Capturing	$\mathbf{YZ}\boldsymbol{\beta}$ and δ
Nonlinear Solver	Monolithic Newton
Relaxation	1.0
Linear Solver	UMFPACK

The results reported by Tezduyar and Senga[72] can be seen in Figure 2.10.
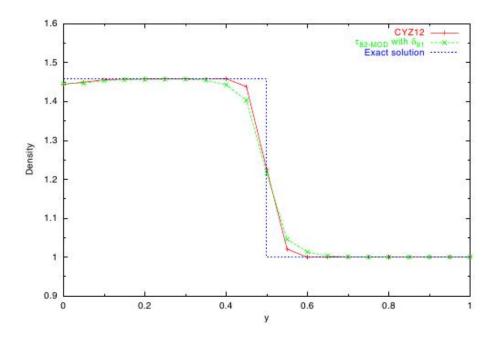
Figure 2.10: Results published by Tezduyar and Senga[72]

The results for FEMDOC, shown in Figure 2.11, correlate well with Tezduyar and Senga. **YZβ** is labeled in Figures 2.10 and 2.11 as CYZ12 to be consistent with Tezduyar's labels. The shock capturing parameter is a nodally reconstructed value. The nodal reconstruction system can be built using a different integration than the fluid system. Here it is solved using both one point and two point by two point integration rules. The one point integration matches exactly with Tezduyar's results when overlaid. The overlay is not shown here because it is too difficult to distinguish between the two sets of results. While it is difficult to see here, the two point by two point integration yields slight more diffusive results, nearly the same as the δ results. This suggests that a one point integration rule is best, which equates to a nodally reconstructed variable being the volume weighted average of an element constant value for the patch of connected elements.
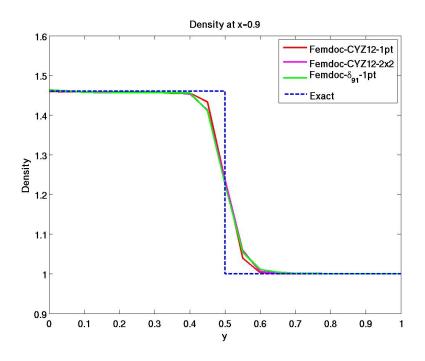
Figure 2.11: FEMDOC results for Tezduyar compression corner

2.10.4   Supersonic Flat Plate With No Turbulence

The next example is a supersonic flat plate with no turbulence. This example is Carter's problem taken from Shakib et al.[67] and was originally compared with FEMDOC by Howard[32]. The results are briefly presented here in order to show that the more recent work did not change the results obtained by Howard previously. The boundary conditions are the same as shown in Figure 2.12. The analysis parameters are listed in Table 2.4
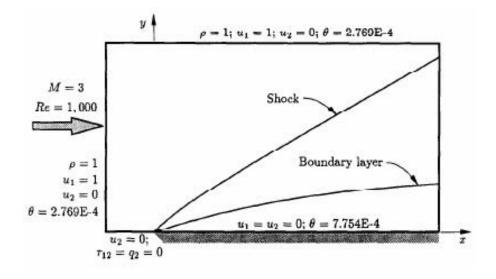
Figure 2.12: Carter's problem setup taken from Shakib et al.[67]

Table 2.4: Analysis parameters used for the supersonic flat plate with no turbulence

Num of Procs	6
Time Solver	Backwards Euler
Time Stepping Scheme	Gresho
Gresho ϵ	0.1
Max Time Step Increase	20%
Stabilization	$\boldsymbol{\tau}_{BK}$
Shock Capturing	δ
Nonlinear Solver	Monolithic Newton
Relaxation	1.0
Linear Solver	GMRES
Preconditioner	ILU

Figure 2.13 shows the FEMDOC results for coefficient of pressure overlaid with the published results from Shakib et al.[67] and they correlate quite well.
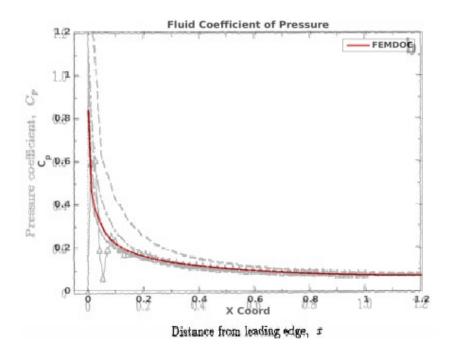
Figure 2.13: Carter's problem pressure comparison with Shakib et al.[67]

2.11 Ill-Conditioning of Linear Approximations for Turbulent Compressible CFD by SUPG Finite Element

Section 2.10 shows four examples where FEMDOC is able to produce accurate results. These examples and others instill confidence that the compressible fluid and turbulence models are implemented correctly. The experience using this method, however, is that it is not very robust. While accurate results are able to be obtained, slight changes in the finite element meshes, chosen nonlinear solver, or linear solution method can cause an analysis solution to diverge. This can be attributed to the nonlinearity of the system and the ill-conditioned linear approximations produced using the SUPG finite element method for compressible flow.

The condition number of a matrix is an estimation of how sensitive the solution of a linear system with that matrix is to changes in the right-hand side vector. Ill-conditioning can be caused by poorly scaled equations, which can lead to large disparities in sensitivities for some DOFs. For finite element problems the condition number of the Jacobian can loosely be interpreted as the sensitivity

of the solution to changes in the nonlinear residual. Therefore, for a Jacobian with a high condition number, known as ill-conditioned, the round-off errors associated with calculating the nonlinear residual can potentially result in large changes to the solution of the linear approximation. Likewise, ill-conditioned systems can indicate the nonlinear residual is relatively insensitive to changes in the solution for particular DOFs. If the condition number of a Jacobian is 10^k, then k digits of accuracy can be lost[79, 52]. IEEE double precision is about fifteen digits of accuracy. Therefore, a condition number greater than 10^{15} may, in general, result in total loss of precision. The condition number of large sparse matrices are typically computationally expensive to calculate. Often, an estimation of the condition number can be used to gain insight. In this research MATLAB R2015b$^{\circledR}$ is used to calculate the condition number estimate using a block algorithm for the matrix 1-norm[30].

The condition numbers reported here are for the first time step and first nonlinear iteration. They are only given here for a relative comparison. As the analysis continues, the condition number of the linear approximations increases due to flow development and larger time steps. The condition number of the Jacobian for a monolithic system using a dimensional analysis of the NASA turbulent flat plate example from Section 2.10.1 is 2.6×10^{23}. The conditioning of the linear approximation can be improved by way of nondimensional analysis. Simply choosing to do a nondimensional analysis reduces the condition number of the linear approximation to 1.1×10^{20}. The condition numbers reported previously are for a fully consistent Jacobian. Often times an approximate linearization can be used to reduce the ill-conditioning of the linear system and simultaneously smooth the nonlinearity in the system. Dropping off-diagonal coupling terms makes the linear approximation more diagonally dominant and is an effective way to improve the condition number. Various approximate Jacobian formulations have been explored during the course of this research which reduce the condition number to as low as 1.4×10^{12}. Less accurate linear approximations may cause the nonlinear method to converge more slowly. Therefore, a careful balance must be achieved in order to optimize the approximation for both the linear solver and the nonlinear solver. Experience suggests that the approximate Jacobian shown in Equation 2.73 is the best balance between the linear solver and nonlinear solver for both computational efficiency and robustness,

with regards to the NASA turbulent flat plate example. For illustration purposes let ,$\mathbf{U}$ be the fluid variables, $\rho\tilde{\nu}$ is the turbulence variable, h is the length scale, δ is the shock capturing parameter, $\mathbf{G}$ are the nodal viscus fluxes for the fluid system, τ_t is the stabilization parameter for the turbulence equation, and $\mathbf{G}_t$ are the diffusive fluxes for the turbulence equation. $\mathbf{R}$ is the residual associated with each equation for the subscript variables.

$$
\begin{array}{c}
\\
\mathbf{R_U} \\
\mathbf{R}_{\rho\tilde{\nu}} \\
\mathbf{R}_h \\
\mathbf{R}_\delta \\
\mathbf{R_G} \\
\mathbf{R}_{\tau_t} \\
\mathbf{R}_{\mathbf{G}_t}
\end{array}
\begin{array}{ccccccc}
\mathbf{U} & \rho\tilde{\nu} & h & \delta & \mathbf{G} & \tau_t & \mathbf{G}_t \\
\left[\begin{array}{ccccccc}
\dfrac{\mathbf{R_U}}{\mathbf{U}} & \cancel{\dfrac{\mathbf{R_U}}{\rho\tilde{\nu}}} & \dfrac{\mathbf{R_U}}{h} & \cancel{\dfrac{\mathbf{R_U}}{\delta}} & \dfrac{\mathbf{R_U}}{\mathbf{G}} & 0 & 0 \\[3mm]
\cancel{\dfrac{\mathbf{R}_{\rho\tilde{\nu}}}{\mathbf{U}}} & \dfrac{\mathbf{R}_{\rho\tilde{\nu}}}{\rho\tilde{\nu}} & 0 & \dfrac{\mathbf{R}_{\rho\tilde{\nu}}}{\delta} & 0 & \dfrac{\mathbf{R}_{\rho\tilde{\nu}}}{\tau_t} & \dfrac{\mathbf{R}_{\rho\tilde{\nu}}}{\mathbf{G}_t} \\[3mm]
\cancel{\dfrac{\mathbf{R}_h}{\mathbf{U}}} & 0 & \dfrac{\mathbf{R}_h}{h} & 0 & 0 & 0 & 0 \\[3mm]
\cancel{\dfrac{\mathbf{R}_\delta}{\mathbf{U}}} & \cancel{\dfrac{\mathbf{R}_\delta}{\rho\tilde{\nu}}} & 0 & \dfrac{\mathbf{R}_\delta}{\delta} & \cancel{\dfrac{\mathbf{R}_\delta}{\mathbf{G}}} & 0 & \cancel{\dfrac{\mathbf{R}_\delta}{\mathbf{G}_t}} \\[3mm]
\cancel{\dfrac{\mathbf{R_G}}{\mathbf{U}}} & 0 & 0 & 0 & \dfrac{\mathbf{R_G}}{\mathbf{G}} & 0 & 0 \\[3mm]
\dfrac{\mathbf{R}_{\tau_t}}{\mathbf{U}} & \dfrac{\mathbf{R}_{\tau_t}}{\rho\tilde{\nu}} & 0 & 0 & 0 & \dfrac{\mathbf{R}_{\tau_t}}{\tau_t} & 0 \\[3mm]
\dfrac{\mathbf{R}_{\mathbf{G}_t}}{\mathbf{U}} & \dfrac{\mathbf{R}_{\mathbf{G}_t}}{\rho\tilde{\nu}} & 0 & 0 & 0 & 0 & \dfrac{\mathbf{R}_{\mathbf{G}_t}}{\mathbf{G}_t}
\end{array}\right]
\end{array}
\tag{2.73}
$$

The terms that are dropped are shown with the strikeout. The condition number for this approximate Jacobian is 4.1×10^{14}. Equation 2.73 shows that the fluid and turbulence equations are decoupled and this has proven to improve the robustness. Also, the length scale and shock capturing parameter need to remain strictly positive, so the off-diagonal contributions are dropped to aid in this. With condition numbers on the order of 10^{14}, the linear approximations are still poorly conditioned. Loss of precision is expected, but the degree to which this is the case is dependent on the linear solver and is discussed in Chapter 5.

2.12   Nonlinearity of Turbulent Compressible CFD by SUPG Finite Element

The formulation of the compressible turbulent flow by SUPG finite element analysis results in ill-conditioned linear approximations that requiring solving. The solutions of those ill-conditioned linear approximations experience a loss of precision. This loss of precision can cause the nonlinear solution to diverge, have poor convergence rates, or completely stall in convergence. The sensitivity to linear solution error is a characteristic of the nonlinear problem formulation. Highly nonlinear

systems are more sensitive to loss of precision in the linear solution. SUPG finite element analysis for turbulent compressible flow is highly nonlinear and sensitive to the loss of precision in the solution of the linear approximation. To show the impact of the linear solution error, the same analysis is run three times and the nonlinear residual norm is plotted at each time step. Ideally, multiple runs should result in the same nonlinear residual after each Newton iteration and for each time step. However, linear solvers can have non-deterministic solutions, as further discussed in Section5.1.1. The non-deterministic solution is amplified by the loss of precision. The analysis used to show this is the same subsonic turbulent flat plate example in Section 2.10.1, using the parameters identified in Table 2.5.

Table 2.5: Analysis parameters used to demonstrate nonlinearity of the subsonic turbulent flat plate

Num of Procs	6
Time Solver	Backwards Euler
Time Stepping Scheme	CFL
Starting CFL	0.05
Max CFL	200.0
Stabilization	$\boldsymbol{\tau}_{SB}$
Shock Capturing	ν
Nonlinear Solver	Monolithic Newton
Linear Solver	MUMPS
Relaxation	0.8
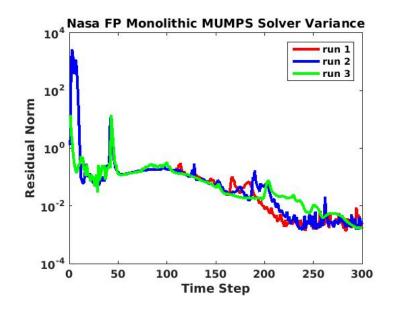
Figure 2.14: Monolithic nonlinear residual norm variance using MUMPS

Figure 2.14 shows that the nonlinear residual norm can vary by three orders of magnitude. The ill-conditioned approximations lead to a loss of precision in the linear solution and because the nonlinear problem formulation is highly nonlinear, this loss of precision in the linear solution propagates through the analysis. This shows that the nonlinear formulation is very sensitive to the loss of precision in the linear solution. The first two runs are compared for each DOF at each node at each time step and the maximum nodal relative difference, defined in Equation 2.74, is plotted against the time step.

$$\text{Relative Difference} = \frac{|\text{val}_1 - \text{val}_2|}{\max(|\text{val}_1|, |\text{val}_2|)} \tag{2.74}$$

Figure 2.15 shows that the difference can be up to 200%, indicating a different sign for the two solutions. The 200% is for the momentum in the y-direction which is comparatively very small in magnitude. Given that this a turbulent flow, the difference in vertical flow perturbations are not of much concern. The turbulence variable and horizontal momentum remain above 10% difference and this is a concern. This also shows that variation can be larger at the beginning, but eventually dampens out to a quasi-steady-state maximum difference, showing that the two runs are converging to slightly different solutions.
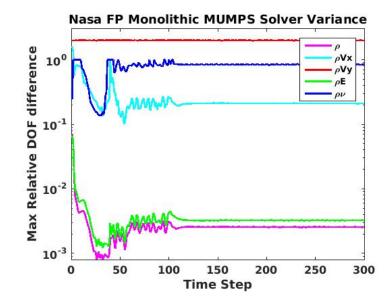
Figure 2.15: Maximum DOF difference between solution 1 and 2 using MUMPS

The adaptive time stepping scheme used above is the CFL scheme previously discussed. This adaptive time stepping scheme is effective and common, but because of its dependence on the solution, it can be a contributing factor to the solution variation discussed above. Therefore, the same analysis as above is shown for an adaptive time stepping scheme which is not dependent on the solution. The time stepping scheme used in the following result uses Equation 2.75, such that the time step size is increased 10% each time step.

$$\Delta t^n = 1.1(\Delta t^{n-1}) \tag{2.75}$$

This calculation of the adaptive time stepping eliminates this as a contributing factor to the nonlinearity. Even with this nonlinearity eliminated, the loss of precision in the linear solution has the same effect, as shown in Figure 2.16.
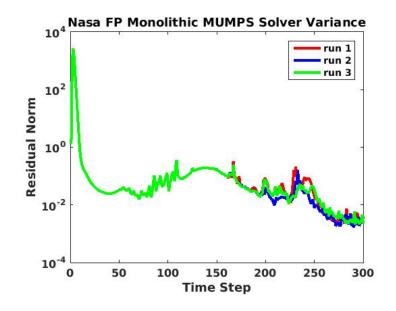
Figure 2.16: Monolithic nonlinear residual norm variance using MUMPS and a predictable time step

The ill-conditioning of the linear approximations and the nonlinearity of the compressible SUPG finite element method result in a lack of robustness. The linear solver chosen can have an influence on how many digits of precision are lost, potentially mitigating the negative results due to ill-conditioning. The nonlinear solution method chosen can likewise potentially mitigate the negative results due to the nonlinearity of the system. This is further discussed in subsequent chapters. As this is explored, an incompressible fluid-structure interaction problem is used for comparison with the turbulent compressible flow problem. The incompressible fluid-structure interaction problem is chosen as it can also be a numerically difficult problem to solve, resulting in ill-conditioned linear approximations, but the formulation of the problem is not as nonlinear as the turbulent compressible example.

Chapter 3

Fluid-Structure Coupling Strategy

Fluid-structure interaction problems require coupling different disciplines at the interface boundary. Traditionally, a partitioned approach is used in an iterative fashion until the solution on both domains is resolved. This consists of obtaining separate solutions on each of the domains and passing Dirichlet-like boundary conditions at the interface, $\mathbf{\Gamma_{fs}}$, to handle the coupling. In more recent years there has been much research dedicated to obtaining solutions on both the fluid domain, $\mathbf{\Omega_f}$, and the structure domain, $\mathbf{\Omega_s}$, via the same discretization method. The mortar method[8] and discontinuous Galerkin[76] methods are two examples of how the interface conditions can be weakly enforced.
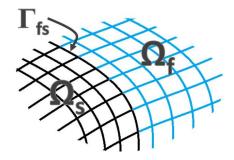
Figure 3.1: Fluid-structure interaction domains

Residual based coupling is another coupling strategy that can be used when both the fluid domain and the structural domain are discretized using finite elements. Residual based coupling is attractive because it does not increase the size of the linear systems to be solved and it is straight

forward to implement. Residual based coupling is used in this work.

Residual based coupling is briefly explained in the following sections, but the reader is referred to Howard[32] and Howard and Bova[34] for more details. The fluid-structure interaction work in this research focuses on incompressible fluid flow. Therefore, the formulation is discussed using incompressible fluid primitive variables; pressure, p, and velocities, $\mathbf{v}$. The incompressible fluid implementation was previously done by Kreissl[46] and Jenkins[40]. This work utilizes the incompressible model to explore several different coupling strategies and their implication on computational efficiency and robustness.

3.1   Residual Based Coupling

3.1.1   Three Field Formulation

The residual based coupling starts with a three field arbitrary Lagrangian-Eularian(ALE) formulation first introduced by Farhat et al.[23]. A fluid-structure interaction problem can be broken up into the structural domain, $\mathbf{\Omega_s}$, the fluid domain, $\mathbf{\Omega_f}$, and the boundary between them, $\mathbf{\Gamma_{fs}}$, as seen in Figure 3.1. The structural equilibrium equations are formulated in a Lagrangian reference frame with structural displacement degrees of freedom $\mathbf{u^s}$. The fluid Navier-Stokes equations are formulated in an ALE reference frame with an ALE correction due to the moving mesh, as previously discussed in Chapter 2. This results in a two field formulation for the fluid domain; the fluid degrees of freedom and the mesh deformation degrees of freedom, $\mathbf{u^f}$. Figure 3.2 shows this in a graphical representation.
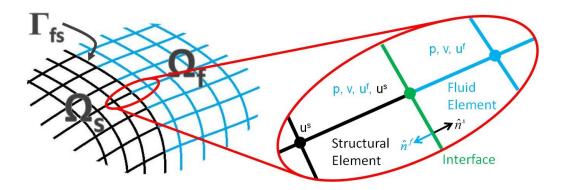
Figure 3.2: Fluid-structure three field formulation with degrees of freedom

3.1.2   Interface Conditions

All three fields exist at the interface between domains. It is at this interface that the different fields are coupled. Regardless of the method chosen, the interface conditions remain the same. The first condition is that the structural displacements must match the fluid mesh displacements as seen in Equation 3.1.

$$u_i^f - u_i^s = 0 \tag{3.1}$$

Secondly, for a stick boundary condition, the velocity of the structure must match the fluid velocity at the interface as seen in Equation 3.2. That is to say the fluid velocity at the interface is zero relative to the structure.

$$v_i^f - \dot{u}_i^s = 0 \tag{3.2}$$

The third interface condition is the traction interface condition. The structural traction must equal the fluid traction. The fluid traction includes both the deviatoric stress and the stress due to pressure. Defining the element normal vector to be pointing outward results in fluid and structural elements having equal and opposite normals as shown in Figure 3.2. Therefore the traction forces for the fluid and structure are also equal and opposite using these normals, as shown in Equation 3.3.

$$\sigma_{ij}^f \hat{n}_j^f + \sigma_{ij}^s \hat{n}_j^s = 0 \tag{3.3}$$

These interface conditions can be enforced through a modification to the finite element residual and Jacobian at each of the interface nodes. This is the basis of residual based coupling.

3.1.3   Residual and Jacobian Modifications

First, a brief review of the residual equations for both the fluid and solid domains. The Navier-Stokes momentum absent body forces, Equation 3.4,

$$\rho \frac{\mathrm{D}v_i}{\mathrm{D}t} = \frac{\partial \sigma_{ij}}{\partial x_j} \tag{3.4}$$

is tested with an appropriate test function, W^n, at a given node, n, and integrated over the domain resulting in the Galerkin weak form of the residual shown in Equation 3.5.

$$\mathbf{R}_f^n \equiv \int_{\Omega_f} W^n \left(\rho \frac{\mathrm{D}v_i}{\mathrm{D}t} - \frac{\partial \sigma_{ij}}{\partial x_j} \right) \, d\Omega_f = 0 \tag{3.5}$$

Choosing then to integrate the stress term by parts and splitting the boundary term between the fluid-structure interface, $\mathbf{\Gamma_{fs}}$, and the rest of the fluid boundary, $\mathbf{\Gamma_f}$, results in Equation 3.6.

$$\mathbf{R}_f^n = \int_{\Omega_f} \left(W^n \rho \frac{\mathrm{D}v_i}{\mathrm{D}t} + \frac{\partial W^n}{\partial x_j} \sigma_{ij} \right) \, d\Omega_f - \int_{\Gamma_f} W^n \sigma_{ij} \hat{n}_j^f \, d\Gamma_f - \int_{\Gamma_{fs}} W^n \sigma_{ij} \hat{n}_j^f \, d\Gamma_{fs} = 0 \tag{3.6}$$

Now for the sake of convenience all terms except the interface boundary stress term are grouped and defined to be the modified residual at a node, $\bar{\mathbf{R}}_f^n$, shown in Equation 3.7.

$$\mathbf{R}_f^n = \bar{\mathbf{R}}_f^n - \int_{\Gamma_{fs}} W^n \sigma_{ij} \hat{n}_j^f \, d\Gamma_{fs} = 0 \tag{3.7}$$

The same process is used on the structural domain. First, start with the elastodynamic equation absent of body forces, Equation 3.8.

$$\rho \frac{\partial^2 u_i}{\partial t^2} + C \frac{\partial u_i}{\partial t} = \frac{\partial \sigma_{ij}}{\partial x_j} \tag{3.8}$$

Following the Galerkin method again, make Equation 3.8 weak by testing it with an appropriate test function at a node, Y^n, and integrate it over the domain, resulting in Equation 3.9.

$$\mathbf{R}_s^n \equiv \int_{\Omega_s} Y^n \left(\rho \frac{\partial^2 u_i}{\partial t^2} + C \frac{\partial u_i}{\partial t} - \frac{\partial \sigma_{ij}}{\partial x_j} \right) \, d\Omega_s = 0 \tag{3.9}$$

Again, the stress term is integrated by parts and the boundary term is split between the fluid-structure interface and the rest of the structural boundary, resulting in Equation 3.10.

$$
\begin{aligned}
\mathbf{R}_s^n = \int_{\Omega_s} \left(Y^n \left(\rho \frac{\partial^2 u_i}{\partial t^2} + C \frac{\partial u_i}{\partial t} \right) + \frac{\partial Y^n}{\partial x_j} \sigma_{ij} \right) \, d\Omega_s \\
- \int_{\Gamma_s} Y^n \sigma_{ij} \hat{n}_j^s \, d\Gamma_s - \int_{\Gamma_{fs}} Y^n \sigma_{ij} \hat{n}_j^s \, d\Gamma_{fs} = 0
\end{aligned}
\tag{3.10}
$$

Finally the modified structural residual, $\bar{\mathbf{R}}_s^n$, is conveniently defined as all terms except the interface stress term, as shown in Equation 3.11.

$$
\mathbf{R}_s^n = \bar{\mathbf{R}}_s^n - \int_{\Gamma_{fs}} Y^n \sigma_{ij} \hat{n}_j^s \, d\Gamma_{fs} = 0
\tag{3.11}
$$

Now, the interface traction condition shown in Equation 3.3 is tested and integrated over the interface resulting in the Galerkin weak form, shown in Equation 3.12.

$$
\int_{\Gamma_{fs}} W^n \sigma_{ij}^f \hat{n}_j^f \, d\Gamma_{fs} + \int_{\Gamma_{fs}} W^n \sigma_{ij}^s \hat{n}_j^s \, d\Gamma_{fs} = 0
\tag{3.12}
$$

If the same test functions are used for the fluid domain and the structural domain, shown in Equation 3.13,

$$
W^n = Y^n \|_{\Gamma_{fs}}
\tag{3.13}
$$

then Equations 3.7, 3.11 and 3.13 can be substituted into Equation 3.12, resulting in the new interface traction condition, as seen in Equation 3.14.

$$
\bar{\mathbf{R}}_f^n + \bar{\mathbf{R}}_s^n = 0
\tag{3.14}
$$

Following this approach, the residuals at a given node are modified to enforce the three interface conditions. First, the fluid momentum residual at a given node, n, on the fluid-structure interface is replaced with Equation 3.14. Second, the fluid mesh deformation residual at the same node is replaced by Equation 3.1. Third, the structural deformation residual at the interface nodes

is replaced by Equation 3.2. This is summarized as follows.

$$\mathbf{R}_f^n \longrightarrow \bar{\mathbf{R}}_f^n + \bar{\mathbf{R}}_s^n$$

$$\mathbf{R}_m^n \longrightarrow u_i^f - u_i^s \tag{3.15}$$

$$\mathbf{R}_s^n \longrightarrow v_i^f - \dot{u}_i^s$$

The nodal residual replacements in Equation 3.15 result in the following modifications to the Jacobian at the same node. The residuals down the left side of the matrix and the degrees of freedom along the top of the matrix are shown to make the organization and structure of the Jacobian clear to the reader.

$$\begin{array}{c} \\ \mathbf{R}_f^n \\ \\ \mathbf{R}_m^n \\ \\ \mathbf{R}_s^n \end{array} \begin{array}{ccc} v^f & u^f & u^s \\ \left[\begin{array}{ccc} \frac{\partial \bar{\mathbf{R}}_f^n}{\partial v^f} & \cancel{\frac{\partial \bar{\mathbf{R}}_f^n}{\partial u^f}} & \frac{\partial \bar{\mathbf{R}}_s^n}{\partial u^s} \\ 0 & 1 & -1 \\ 1 & 0 & \frac{-1}{\Delta t} \end{array} \right] \end{array} \tag{3.16}$$

The bottom right term in the Jacobian matrix, seen in Equation 3.16, is shown as $\frac{-1}{\Delta t}$. This is representative of the fact that this entry is the derivative of the time rate of change of the variable. The actual entry here is dependent on the type of time integration used and so the $\frac{-1}{\Delta t}$ would be replaced by the appropriate derivative associated with the chosen time integration. The derivative of the fluid momentum equations with respect to the fluid mesh displacements is shown with a strike through. This is to indicate that while there is a contribution of the mesh displacements to the fluid momentum equations, it is not considered in this work.

3.1.4   Alternative Residual and Jacobian Modifications

The residual based method described above follows previous work on residual based coupling of Howard[32] and Howard and Bova[34]. It is considered strong form residual based coupling because the velocity and displacement conditions are enforced pointwise. The work done in this research explores the impact of a few possible variants to the method. The first variant is weak form coupling of the velocity and displacement coupling conditions. This means following the Galerkin method of testing the interface conditions and integrating over the interface boundary as shown in

Equation 3.17. This has a direct traceability to continuum mechanics at the interface and is not just a product of the finite element method used.

$$
\begin{aligned}
\mathbf{R}_f^n &\longrightarrow \bar{\mathbf{R}}_f^n + \bar{\mathbf{R}}_s^n \\
\mathbf{R}_m^n &\longrightarrow \int_{\Gamma_{fs}} W^n (u_i^f - u_i^s) \, d\Gamma_{fs} \qquad (3.17) \\
\mathbf{R}_s^n &\longrightarrow \int_{\Gamma_{fs}} W^n (v_i^f - \dot{u}_i^s) \, d\Gamma_{fs}
\end{aligned}
\qquad
\begin{array}{c}
\begin{array}{ccc} v^f & u^f & u^s \end{array} \\
\begin{array}{c} \mathbf{R}_f^n \\[8pt] \mathbf{R}_m^n \\[8pt] \mathbf{R}_s^n \end{array}
\left[\begin{array}{ccc}
\dfrac{\partial \bar{\mathbf{R}}_f^n}{\partial v^f} & \dfrac{\partial \bar{\mathbf{R}}_f^n}{\partial u^f} & \dfrac{\partial \bar{\mathbf{R}}_s^n}{\partial u^s} \\[8pt]
0 & \Delta \Gamma_{fs} & -\Delta \Gamma_{fs} \\[8pt]
\Delta \Gamma_{fs} & 0 & \dfrac{-\Delta \Gamma_{fs}}{\Delta t}
\end{array} \right]
\end{array}
\qquad (3.18)
$$

This is the equivalent of scaling the two equations with the equivalent mass associated with the interface node. Equation 3.18 uses $\Delta \Gamma_{fs}$ to denote the nodal equivalent mass.

The next variant is in the storage of the interface conditions. In previous work[32, 34], the traction interface condition was stored in the fluid momentum equations, resulting in adding part of the structural residual to part of the fluid residual at the interface nodes. Alternatively, one could chose to store the traction interface condition in the structural momentum equations. This is just a matter of rearranging the equations, but more importantly it determines which degrees of freedom for each equation are on the diagonal and which degrees of freedom are in an off-diagonal position. This rearrangement is captured in Equations 3.19 and 3.20. This can be done in a strong sense and enforced pointwise instead of integrating as well.

$$
\begin{aligned}
\mathbf{R}_f^n &\longrightarrow \int_{\Gamma_{fs}} W^n (v_i^f - \dot{u}_i^s) \, d\Gamma_{fs} \\
\mathbf{R}_m^n &\longrightarrow \int_{\Gamma_{fs}} W^n (u_i^f - u_i^s) \, d\Gamma_{fs} \qquad (3.19) \\
\mathbf{R}_s^n &\longrightarrow \bar{\mathbf{R}}_f^n + \bar{\mathbf{R}}_s^n
\end{aligned}
\qquad
\begin{array}{c}
\begin{array}{ccc} v^f & u^f & u^s \end{array} \\
\begin{array}{c} \mathbf{R}_f^n \\[8pt] \mathbf{R}_m^n \\[8pt] \mathbf{R}_s^n \end{array}
\left[\begin{array}{ccc}
\Delta \Gamma_{fs} & 0 & \dfrac{-\Delta \Gamma_{fs}}{\Delta t} \\[8pt]
0 & \Delta \Gamma_{fs} & -\Delta \Gamma_{fs} \\[8pt]
\dfrac{\partial \bar{\mathbf{R}}_f^n}{\partial v^f} & \dfrac{\partial \bar{\mathbf{R}}_f^n}{\partial u^f} & \dfrac{\partial \bar{\mathbf{R}}_s^n}{\partial u^s}
\end{array} \right]
\end{array}
\qquad (3.20)
$$

The last variant builds on the previous one. If the traction condition is stored in the structural momentum equations, then the velocity condition is stored in the fluid momentum equation as seen above. This allows the user to enforce velocity matching between the fluid velocities and the fluid mesh velocity rather than structural velocities. This is shown in Equations 3.21 and 3.22. The choice to couple the fluid velocities and fluid mesh velocities without first rearranging the equations

would result in zeros on the diagonal of the Jacobian and that is not desirable. One of the primary advantages of residual based coupling over the use of multi-point constraints is not having to deal with zeros on the diagonal of the linear systems.

$$\mathbf{R}_f^n \longrightarrow \int_{\Gamma_{fs}} W^n (v_i^f - \dot{u}_i^f) \, d\Gamma_{fs}$$

$$\mathbf{R}_m^n \longrightarrow \int_{\Gamma_{fs}} W^n (u_i^f - u_i^s) \, d\Gamma_{fs} \qquad (3.21)$$

$$\mathbf{R}_s^n \longrightarrow \bar{\mathbf{R}}_f^n + \bar{\mathbf{R}}_s^n$$

$$
\begin{array}{c}
\\
\mathbf{R}_f^n \\
\mathbf{R}_m^n \\
\mathbf{R}_s^n
\end{array}
\begin{array}{ccc}
v^f & u^f & u^s \\
\left[\begin{array}{ccc}
\Delta\Gamma_{fs} & \frac{-\Delta\Gamma_{fs}}{\Delta t} & 0 \\
0 & \Delta\Gamma_{fs} & -\Delta\Gamma_{fs} \\
\frac{\partial\bar{\mathbf{R}}_f^n}{\partial v^f} & \cancel{\frac{\partial\bar{\mathbf{R}}_f^n}{\partial u^f}} & \frac{\partial\bar{\mathbf{R}}_s^n}{\partial u^s}
\end{array}\right] & &
\end{array}
\qquad (3.22)
$$

This also could be done pointwise rather than integrated over the interface. It is worth noting that the integration over the interface results in a full mass matrix in the Jacobian, which is well known to cause problems. So, this work utilizes a lumped mass matrix for the interface conditions.

3.2   Comparison of Coupling Variations

The variations on the residual modification coupling strategy used by Howard[32] and Howard and Bova[34] result in different linear systems to be solved. The linear systems vary in off-diagonal strength, scaling of the equations and even which degrees of freedom are coupled. The work done in this research focuses on comparing the computational benefits of the different variations. To do this comparison, the test case used is a fluid-structure interaction benchmark originally proposed by Turek and Hron [73], subsequently referred to as the Turek problem.

The Turek problem is an incompressible laminar flow over a rigid cylinder and trailing flag in a channel, as seen in Figure 3.3. This problem is used because it is characterized by large deformations, resulting in highly coupled fluid and structure domains. The high degree of coupling makes it a particularly difficult problem to solve.

Fig. 1. Computational domain
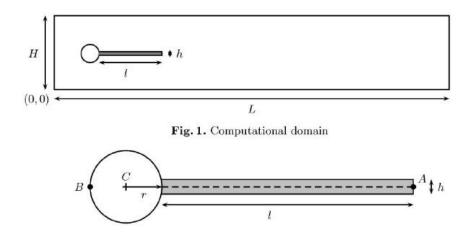
Figure 3.3: Turek and Hron cylinder and flag in laminar incompressible flow[73]

Turek and Hron propose several different benchmarks. The one used here is referred to as "FSI3" in their paper[73], to which the reader is referred to for full details on the problem definition. Notably, this benchmark is a low Reynolds number flow, $Re = 200.0$, with a parabolic inlet velocity which is ramped up at the start. The cylinder and flag are slightly offset in the channel, creating asymmetric flow conditions. The asymmetry induces oscillations on the flag, rather than rely on numerical precision to do so[73]. The large deformation in the flag results in shedding vortices in the channel flow. Figure 3.4 shows the resulting vortices in a snapshot of time.
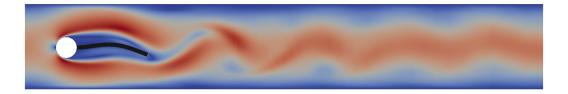
Figure 3.4: Shedding vortices of Turek problem

The work presented here focuses on the efficiency of coupling formulations. Each coupling formulation results in the same time dependent response. The vertical displacement at the tip of the flag is plotted and compared to the results obtained by Turek and Hron in Figure 3.5.
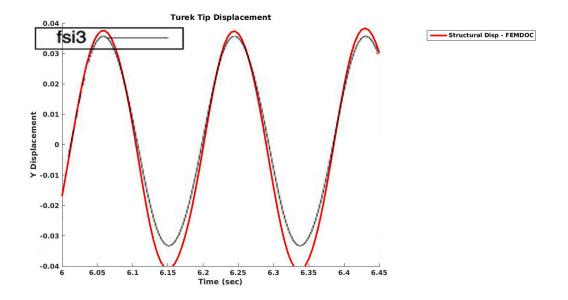
Figure 3.5: Flag tip vertical displacement overlaid with Turek and Hron's results[73]

This shows the frequency of the periodic oscillations to be nearly the same while the amplitude to be off by about 15%. The time integrator used is the Generalized-α[18, 31] method, a type of Newmark[57] solver. More accurate results of the structural response can be obtained through better fine tuning the parameters of the Generalized-α time solver. These results are sufficient since the focus here is on computational efficiency.

The basis for comparing computational efficiency of the coupling formulations is cumulative time spent in the linear solver. The assembly of the linear system is a significant part of the computational time for a solution, but it is the same for all coupling variations. Therefore it is only a function of the number of assemblies and thus the number of linear solves needed. Furthermore, the assembly efficiency is primarily a function of programming efficiency and not based on coupling formulation, so it is not of interest here.

For the following comparisons, all problems are formulated monolithically and Newton's method is used for the nonlinear problem. In each of the plots, the naming convention is as follows. The first part identifies which equation the traction condition is stored in, structural momentum or fluid momentum equations. The second part identifies which degree of freedom the fluid velocity

is coupled to for the velocity interface condition, structural velocity or fluid mesh velocity. The third part identifies whether the velocity and displacement conditions are enforced pointwise or integrated. This is summed up in Table 3.1.

Table 3.1: Fluid-structure coupling naming convention correlation with equations

Struc-MeshVel-Point	Equations 3.21 without integration
Struc-MeshVel-Integ	Equations 3.21
Struc-StrucVel-Point	Equations 3.19 without integration
Struc-StrucVel-Integ	Equations 3.19
Fluid-StrucVel-Point	Equations 3.15
Fluid-StrucVel-Integ	Equations 3.17

3.2.1   Direct Linear Solver

The first comparison of the coupling variations is done using a direct linear solver. The direct linear solver used here is MUltifrontal Massively Parallel sparse direct Solver (MUMPS). The MUMPS solver is used through the AMESOS[64, 65] package as part of Trilinos[29]. All solutions here are run on the same 6 processor cores. The nonlinear residual norm is driven to below 10^{-7} or a drop of more than seven orders of magnitude for each time step. Each analysis is restarted from the same initial conditions and ran for one hundred time steps at 5×10^{-3} seconds each, for a total of a 0.5 second analysis.

Figure 3.6 shows the cumulative time spent in the linear solver. This time includes the symbolic and numeric factorizations as well as the time for the forward and backward substitution to solve the linear system. The symbolic factorization only happens once for the initial solve in these comparisons. This is because the non-zero structure of the linear system is not changing from one linear solve to the next, so the same symbolic factorization is used throughout the analysis, thereby minimizing the computational time.
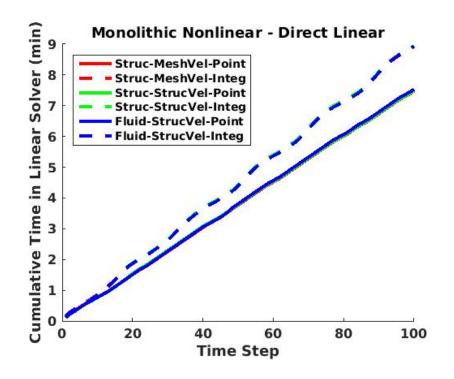
Figure 3.6: MUMPS linear solver time for Turek FSI3 problem

Here it is shown that structure of the linear system does not have an influence. The arrangement of the interface conditions and the choice of coupling for velocities does not have an influence. What does have an influence however is the choice of strong coupling versus weak coupling. The weak coupling requires more computational time in the linear solver. Figure 3.7 shows that it is the number of linear solves required that is driving the additional computational time.
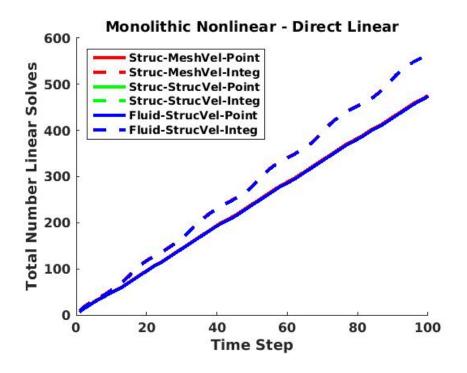
Figure 3.7: Number of linear solves for Turek FSI3 problem

Figures 3.6 and 3.7 show that there is slightly more of an oscillatory nature to the solution time for the weakly enforced coupling. This can be seen even more so when the number of linear solves required for each time step is plotted, as seen in Figure 3.8. This shows that the weakly enforced conditions require anywhere between four and eight linear solves per time step while the strongly enforced coupling only requires four or five linear solves per time step.
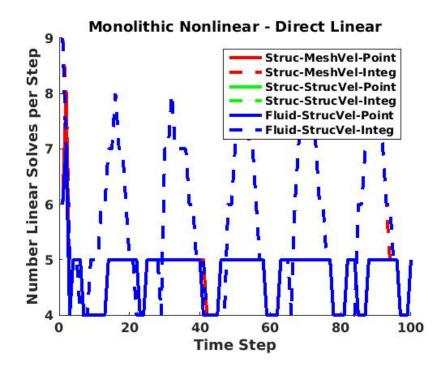
Figure 3.8: Number of linear solves per time step for Turek FSI3 problem

For the case of Turek's FSI3 highly coupled benchmark problem, if using a direct parallel linear solver such as MUMPS, a strong pointwise coupling of the displacements and velocities may result in a more computationally efficient solution. Direct solvers however aren't typically practical for very large problems. For these problems an iterative solver is usually more appropriate.

3.2.2   Iterative Linear Solver

The iterative solution algorithm used for comparison of coupling strategies is the well known Generalized Minimum Residual (GMRES) method[61]. This is part of the AztecOO package within Trilinos[29]. The preconditioner used for this comparison is an incomplete lower/upper (ILU) factorization contained in the Trilinos package IFPACK[63]. The time solver used in this section is the same as in Section 3.2.1. Also as before, each analysis was restarted from the same initial conditions and ran for one hundred time steps at 5×10^{-3} seconds each, for a total of a 0.5 second analysis. The nonlinear residual norm was driven to below 10^{-7} or a drop of more than seven orders of magnitude for each time step.

Similar to the previous section, the cumulative linear solver time reported here includes the time to construct the preconditioner, calculate the preconditioner and iterate within GMRES. The construction of the preconditioner is a one time cost similar to the symbolic factorization of the direct solver. The cumulative linear solver time for each of the six coupling variations can be seen in Figure 3.9. This shows that for the iterative solver, weak coupling versus strong coupling doesn't have much of an influence. What does have an impact is the ordering of the Jacobian matrix. Figure 3.9 shows that when the traction condition is stored in the structural momentum equations and the fluid velocities are coupled with the structural velocities there is an impact on the computational efficiency of the solution. In other words, the coupling method shown in Equations 3.19 and 3.20 is the least computationally efficient using GMRES in this example.
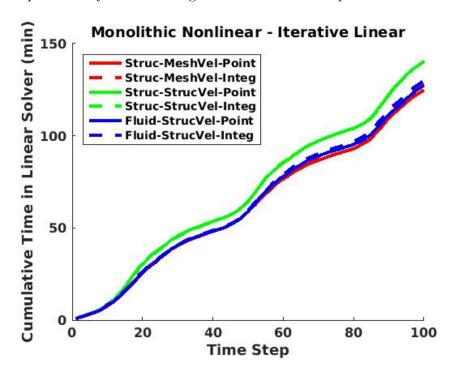
Figure 3.9: Iterative linear solver time for Turek FSI3 problem

Often it is the number of iterations in the linear solver that drive the computational time, but Figure 3.10 shows that there isn't a significant difference in the number of iterations required. In fact, the least computationally efficient method also required the fewest linear iterations, but

not by much.
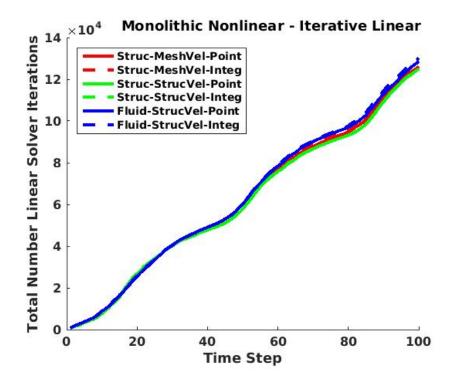
Figure 3.10: Number of linear solver iterations for Turek FSI3 problem

Figure 3.11 shows that the number of linear solves needed to drop the nonlinear residual norm is part of the disparity in efficiency. This leads to a greater number of calculations of the preconditioner.
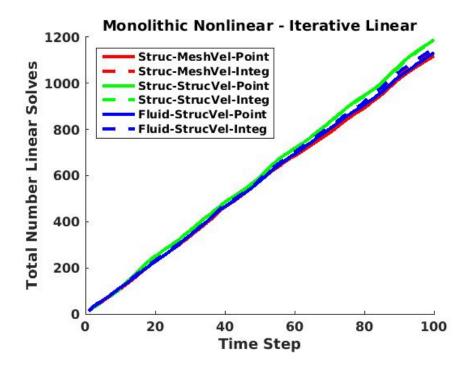
Figure 3.11: Number of linear solves for Turek FSI3 problem

Figure 3.12 shows it is the time to calculate the preconditioner that accounts for the disparity in efficiencies. The formulation shown in Equations 3.19 and 3.20 place the fluid residual derivatives and the structural velocities in the furthest off-diagonal positions. Strong off-diagonal terms cause the preconditioner to be more expensive to compute[12]. The preconditioner is more computationally expensive because a greater fill is required to keep the solution from diverging. The ILU level-of-fill used for the more computationally efficient analyses was four, while it was necessary to increase it to six for the less computationally efficient analyses. The level of fill for ILU preconditioning is discussed in greater detail in Section 5.2.2.1.
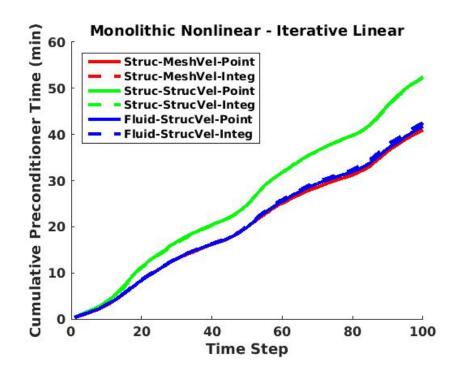
Figure 3.12: Computational time spent in GMRES calculating the preconditioner for Turek FSI3 problem

3.2.3   Selection of Coupling Strategies

The Turek problem is a highly coupled fluid-structure interaction problem. This problem results in ill-conditioned linear approximations regardless of the coupling formulation used. For all the above analyses, the condition number of the linear approximations, prior to any preconditioning or reordering, varies between 1.0×10^{18} and 3.3×10^{20} for each linear solve. In this example, the coupling formulations do not have a significant impact on how ill-conditioned the linear approximation is, but they do impact the efficiency of the solvers to solve the ill-conditioned matrix. For reference, Equation 3.21, both with and without weakly enforcing the velocity and displacement conditions, results in a nonzero sparsity pattern as shown in Figure 3.13 for the above analyses. The number of nonzero entries is noted as nz.
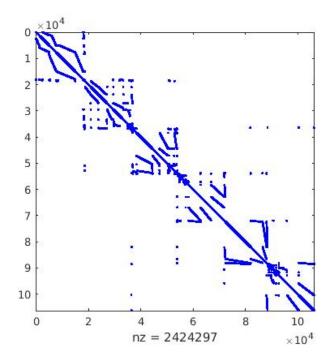
Figure 3.13: Nonzero sparsity structure of linear approximation for Turek FSI3 problem

The four formulations that couple the fluid velocities with the structural velocities, Equations 3.15 thru 3.21, have the same nonzero sparsity structure, which is different than shown above. Figure 3.14 shows the entries that exist in the structure of one matrix, but not the other.
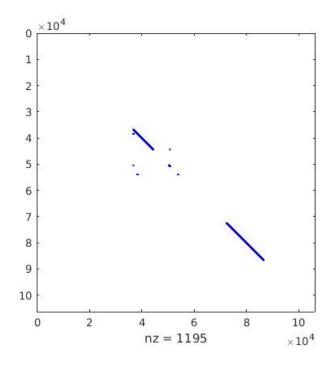
Figure 3.14: Difference in nonzero sparsity structure of linear approximation between coupling strategies for Turek FSI3 problem

This is shown to illustrate that the difference in sparsity pattern is small, less than 0.05% difference in nonzero entries. This is influenced by the fact that the fluid-structure interface is a relatively small portion of the computational domain for this example. In the above examples, the difference in sparsity pattern did not have an influence on computational efficiency. For other problems of interest where the interface is a larger portion of the domain, the change in sparsity pattern may have more an influence on the results.

Section 3.2.1 shows that the efficiency of the linear solver MUMPS is more impacted by the scaling of the equations. Section 3.2.2 shows that the ordering of the equations has more of an impact on the efficiency of GMRES using ILU as a preconditioner. This shows that one will not know a priori what the most computationally efficient coupling formulation will be. The choice is necessarily influenced by the characteristics of the linear solver chosen, re-ordering scheme used, preconditioner used, and strength of coupling between degrees of freedom. One would like to have the ability to choose between the different coupling formulations so as to find and choose the most

computationally efficient formulation for each specific analysis.

3.3   Nonlinearity of Incompressible FSI Example

As noted in the previous chapter, compressible turbulent flow by SUPG finite element analysis is highly nonlinear. It was previously stated that the incompressible fluid-structure interaction problem was chosen in part because it is not as nonlinear as the compressible turbulent analysis. For completeness, verification of this claim is shown here. The first nonlinear residual norm of each time step is used for the plot since this is a dynamic problem. As previously discussed, the linear approximations for the Turek FSI3 problem produces linear approximations which are very ill-conditioned, and as such will result in a loss of precision in the linear solution. Figure 3.15 shows that there is no significant influence of this loss of precision.
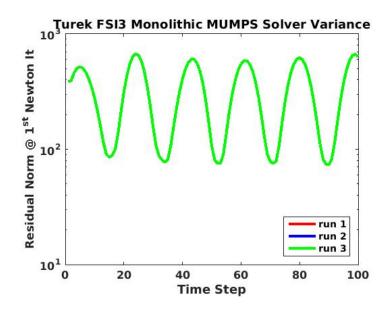
Figure 3.15: Monolithic nonlinear residual norm variance using MUMPS

Figure 3.16 shows there is still some variation in the solution due to loss of precision, but it does not grow throughout the analysis and therefore does not have a significant influence. This is because the nonlinear formulation is less sensitive to the loss of precision as compared to the turbulent compressible flat plate example.
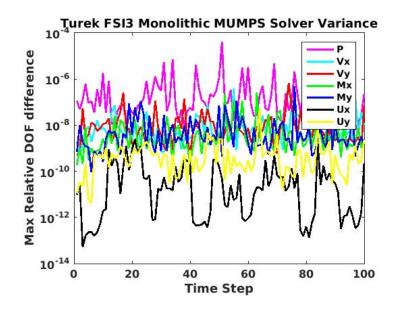
Figure 3.16: Maximum DOF difference between the first and second solution using MUMPS

Chapter 4

Nonlinear Solver

In general, solving the nonlinear equations is done in an iterative fashion by building a linear approximation of the nonlinear system of equations about an initial guess, solving the linear sub-problem, updating the guess based on the linear solution and starting over. There are several ways in which a nonlinear solver can go about building the linear approximation. Multi-physics and multi-field finite element problems have two categories of nonlinear methods. The first category is monolithic nonlinear solvers. Monolithic solvers are distinguished by solving all equations simultaneously in the same large system. The second category is partitioned solvers. Partitioned solvers, partition the large monolithic system into smaller subsystem problems and solve those individually. The nonlinear solvers used in this work are briefly discussed here.

4.1 Monolithic Newton Solver

Newton's method is a powerful root finding method that is attractive because of the potential for quadratic convergence. A monolithic Newton solver is particularly attractive for gradient based optimization analysis, because the Jacobian, or first-order partial derivatives of the residual function, that is built during a monolithic Newton iteration can be reused to calculate the sensitivities for the objective function and constraints. This can make for an efficient gradient based optimization and finite element pairing.

Let $\mathbf{R}$ be a set of generic nonlinear residual functions that is dependent upon the independent

variables, $\mathbf{x}$

$$\mathbf{R}(\mathbf{x}) = 0 \qquad (4.1)$$

Algorithm 1 shows the relaxed Newton's method, here k is the iteration number of Newton's method, such that $\mathbf{x}^k$ is the solution at iteration k. This is called a relaxed Newton's method because of the introduction of the relaxation factor, c. If the relaxation factor is equal to 1.0 then Newton's method is recovered. A selection of $c > 1.0$ is over-relaxed, while $c < 1.0$ is under-relaxed, or damped. When the initial guess, $\mathbf{x}^0$ is not sufficiently close to the solution and the nonlinear function, $\mathbf{R}$, is highly nonlinear, Newton's method can suffer from poor convergence or even divergence. This is often the case with computational fluid dynamics problems where the initial guess is a uniform flow field, which is not sufficiently close to the solution of the developed flow field. In these cases, using an under-relaxation of the Newton step can improve nonlinear convergence rates. For this reason under-relaxation of the Newton step is often utilized in this work.

Algorithm 1 Relaxed Newton's Method

1: $k = 0$
2: **while** not converged **do**
3:      build $\mathbf{R}(\mathbf{x}^k)$, $\frac{\partial \mathbf{R}}{\partial \mathbf{x}}(\mathbf{x}^k)$
4:      $\left[\frac{\partial \mathbf{R}(\mathbf{x}^k)}{\partial \mathbf{x}} \right] \left[\Delta \mathbf{x}^{k+1} \right] = - \left[\mathbf{R}(\mathbf{x}^k) \right]$ ▷ solve for $\Delta \mathbf{x}_i^{k+1}$
5:      $\mathbf{x}^{k+1} = \mathbf{x}^k + c \Delta \mathbf{x}^{k+1}$ ▷ update solution
6:      $k = k + 1$ ▷ increment iteration
7:      check for convergence   ▷ various convergence criteria can be defined
8: **end while**

What makes this a monolithic method is that fact that all fields or blocks are assembled into one linear system and solved simultaneously. For instance, in the turbulent flat plate example, the nodally projected degrees of freedom, $\boldsymbol{\Phi}$, the fluid degrees of freedom, $\mathbf{U}$, and the turbulence degrees of freedom, $\rho \tilde{\boldsymbol{\nu}}$, are assembled as shown in Equation 4.2. The matrix on the left-hand side may have some off-diagonal blocks which are strictly zero, depending on the formulation of the

problem and decision to use an approximate Jacobian.

$$
\begin{bmatrix}
\mathbf{M} & \frac{\partial \mathbf{R}_p}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}_p}{\partial \rho \tilde{\boldsymbol{\nu}}} \\[2mm]
\frac{\partial \mathbf{R}_f}{\partial \boldsymbol{\Phi}} & \frac{\partial \mathbf{R}_f}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}_f}{\partial \rho \tilde{\boldsymbol{\nu}}} \\[2mm]
\frac{\partial \mathbf{R}_t}{\partial \boldsymbol{\Phi}} & \frac{\partial \mathbf{R}_t}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}_t}{\partial \rho \tilde{\boldsymbol{\nu}}}
\end{bmatrix}
\begin{bmatrix}
\Delta \boldsymbol{\Phi} \\[2mm]
\Delta \mathbf{U} \\[2mm]
\Delta \rho \tilde{\boldsymbol{\nu}}
\end{bmatrix}
= -
\begin{bmatrix}
\mathbf{R}_p \\[2mm]
\mathbf{R}_f \\[2mm]
\mathbf{R}_t
\end{bmatrix}
\tag{4.2}
$$

For the case of a fluid-structure interaction problem like Turek's FSI3 example, the fluid, mesh and structural fields are assembled into one linear system as shown in Equation 4.3. Equation 4.3 can be modified for residual based coupling as discussed in Section 3.1.4 previously.

$$
\begin{bmatrix}
\frac{\partial \mathbf{R}_f}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}_f}{\partial \mathbf{u}^m} & \frac{\partial \mathbf{R}_f}{\partial \mathbf{u}^s} \\[2mm]
\frac{\partial \mathbf{R}_m}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}_m}{\partial \mathbf{u}^m} & \frac{\partial \mathbf{R}_m}{\partial \mathbf{u}^s} \\[2mm]
\frac{\partial \mathbf{R}_s}{\partial \mathbf{U}} & \frac{\partial \mathbf{R}_s}{\partial \mathbf{u}^m} & \frac{\partial \mathbf{R}_s}{\partial \mathbf{u}^s}
\end{bmatrix}
\begin{bmatrix}
\Delta \mathbf{U} \\[2mm]
\Delta \mathbf{u}^m \\[2mm]
\Delta \mathbf{u}^s
\end{bmatrix}
= -
\begin{bmatrix}
\mathbf{R}_f \\[2mm]
\mathbf{R}_m \\[2mm]
\mathbf{R}_s
\end{bmatrix}
\tag{4.3}
$$

Newton's method will in general will have quadratic convergence if the starting guess is sufficiently close to the solution, the Jacobian in nonsingular, and a consistent Jacobian is used[14]. In computational fluid dynamics an approximate Jacobian is often used because it is too computationally expensive to calculate the consistent Jacobian, the user wants to reduce the ill-conditioning of the Jacobian by using an approximation, or the user wants to essentially smooth the nonlinearities by using an approximate Jacobian. The use of an approximate Jacobian, under-relaxation, and a starting point "far" from the solution usually prevents quadratic nonlinear convergence rates. The slower convergence rate is a tradeoff to increase robustness. FEMDOC has the ability to use a consistent Jacobian for the compressible turbulent flows, but experience shows that the increased ill-conditioning of the Jacobian causes too great an impact to the robustness of the solution routine.

4.2   Partitioned Solvers

The nonlinear system can be partitioned into an arbitrary number of smaller subsystems. These smaller systems can potentially be more computationally efficient to solve and require less memory as well. The monolithic system can be partitioned in such a way as to produce linear

approximations with particular characteristics. This allows for the use of linear solvers that are tailored and tuned to be efficient and robust for those specific linear systems. The use of optimum linear solvers tailored for each partitioned system is the primary advantage of partitioned solvers. Two partitioned nonlinear solvers are used in this research.

4.2.1 Staggered Newton Solver

The first partitioned nonlinear solver used is referred to as a staggered Newton solver. The nonlinear system is partitioned into n nonlinear subsystems. For the ith nonlinear subsystem, $\mathbf{R}_i$, the only independent variables are those contained in the subsystem, $\mathbf{x}_i$. Both $\mathbf{R}_i$ and $\mathbf{x}_i$ are vectors. All degrees of freedom contained in other subsytems are held constant. This method consists of using Newton's method to solve each partitioned subsystem. It is considered staggered because the subsystems are solved sequentially and the solution of each subsystem is used to build the nonlinear residual and Jacobian for each subsequent subsystem. The staggered Newton solver does not take into account the off-diagonal blocks which couple the subsystems. This method is summarized in Algorithm 2. Here k is the iteration number for the staggered Newton solver and i denotes the subsystem that is being solved.

Algorithm 2 Staggered Newton Solver For Arbitrary Number Of Subsystems

---

1: $k = 0$
2: **while** not converged **do**
3:     **for** $i = 1$ to n **do**
4:         $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k$ ▷ initialize solution
5:         **while** nonlinear subsystem not converged **do**   ▷ subsystem Newton loop
6:             build $\mathbf{R}_i(\mathbf{x}_1^{k+1}...\mathbf{x}_i^{k+1}, \mathbf{x}_{i+1}^k...\mathbf{x}_n^k)$, $\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_i}(\mathbf{x}_1^{k+1}...\mathbf{x}_i^{k+1}, \mathbf{x}_{i+1}^k...\mathbf{x}_n^k)$
7:             $\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_i}\right]\left[\Delta\mathbf{x}_i^{k+1}\right] = -\left[\mathbf{R}_i\right]$ ▷ solve for $\Delta\mathbf{x}_i^{k+1}$
8:             $\mathbf{x}_i^{k+1} = \mathbf{x}_i^{k+1} + c\Delta\mathbf{x}_i^{k+1}$ ▷ update solution
9:             check for convergence of inner nonlinear problem
10:        **end while**
11:    **end for**
12:    $k = k + 1$ ▷ increment iteration
13:    check for convergence   ▷ various convergence criteria can be defined
14: **end while**

4.2.2   Nonlinear Block Gauss-Seidel

Nonlinear block Gauss-Seidel (NLBGS) is a staggered solution method which does take into account the off-diagonal coupling blocks. Block Gauss-Seidel (BGS) is an extension of the point Gauss-Seidel method. BGS is developed by first looking at the linear approximation partitioned into blocks, or subsystems, as shown in Equation 4.4 for n arbitrary subsystems. The subscripts denote the subsystem number.

$$
\begin{bmatrix}
\frac{\partial \mathbf{R}_1}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{R}_1}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{R}_1}{\partial \mathbf{x}_n} \\[4pt]
\frac{\partial \mathbf{R}_2}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{R}_2}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{R}_2}{\partial \mathbf{x}_n} \\[4pt]
\vdots & \vdots & \ddots & \vdots \\[4pt]
\frac{\partial \mathbf{R}_n}{\partial \mathbf{x}_1} & \frac{\partial \mathbf{R}_n}{\partial \mathbf{x}_2} & \cdots & \frac{\partial \mathbf{R}_n}{\partial \mathbf{x}_n}
\end{bmatrix}
\begin{bmatrix}
\Delta \mathbf{x}_1 \\[4pt] \Delta \mathbf{x}_2 \\[4pt] \vdots \\[4pt] \Delta \mathbf{x}_n
\end{bmatrix}
= -
\begin{bmatrix}
\mathbf{R}_1 \\[4pt] \mathbf{R}_2 \\[4pt] \vdots \\[4pt] \mathbf{R}_n
\end{bmatrix}
\tag{4.4}
$$

Isolating the equations for one subsystem and rearranging provides the basis for BGS as shown in Equation 4.5.

$$
\frac{\partial \mathbf{R}_1}{\partial \mathbf{x}_1} \Delta \mathbf{x}_1 = -\mathbf{R}_1 - \frac{\partial \mathbf{R}_1}{\partial \mathbf{x}_2} \Delta \mathbf{x}_2 \ \cdots - \frac{\partial \mathbf{R}_1}{\partial \mathbf{x}_n} \Delta \mathbf{x}_n
\tag{4.5}
$$

This shows that the residual on the right-hand side is modified by the off-diagonal blocks multiplied by the other subsystem solutions. The algorithm used in this work is nonlinear block Gauss-Seidel because the subsystem nonlinear residual and Jacobian block entries are calculated using the updated solution. This is a staggered solution method because the subsystems are solved sequentially and subsequent subsystems are calculated using the current solution. The off-diagonal blocks do not need to be stored as they are only used in a matrix-vector product and subtracted from the residual. This method is summarized in Algorithm 3. The NLBGS iteration number is k and i is the current subsystem being solved.

Algorithm 3 Nonlinear Block Gauss-Seidel For Arbitrary Number Of Subsystems

1: $k = 0$
2: **while** not converged **do**
3:     **for** $i = 1$ to n **do**
4:         build $\mathbf{R}_i(\mathbf{x}_1^{k+1}...\mathbf{x}_{i-1}^{k+1}, \mathbf{x}_i^k...\mathbf{x}_n^k)$, $\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_j}(\mathbf{x}_1^{k+1}...\mathbf{x}_{i-1}^{k+1}, \mathbf{x}_i^k...\mathbf{x}_n^k)$ for $j = 1$ to n
5:         $\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_i}\right]\left[\Delta\mathbf{x}_i^{k+1}\right] = -[\mathbf{R}_i] - \sum_{j=1}^{i-1}\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_j}\right]\left[\Delta\mathbf{x}_j^{k+1}\right] - \sum_{j=i+1}^{n}\left[\frac{\partial \mathbf{R}_i}{\partial \mathbf{x}_j}\right]\left[\Delta\mathbf{x}_j^k\right]$ ▷ solve for $\Delta\mathbf{x}_i^{k+1}$
6:         $\mathbf{x}_i^{k+1} = \mathbf{x}_i^k + \Delta\mathbf{x}_i^{k+1}$ ▷ update solution
7:     **end for**
8:     $k = k + 1$ ▷ increment iteration
9:     check for convergence   ▷ various convergence criteria can be defined
10: **end while**

4.3   Comparison of Nonlinear Methods

The nonlinear solution method chosen will dictate how linear approximations are developed. The assembly time for building linear approximations is a large part of the overall computational time for an analysis. The assembly time is more driven by programming efficiency than anything else. The assembly process in FEMDOC, at the current time, is written such that it loops over every element in the entire computational domain and only those degrees of freedom of interest are assembled into the linear approximation. This is what one would expect for building the monolithic system, but it is suboptimal for the partitioned solution methods. This is done purposefully to maximize flexibility and not computational efficiency. The partitioned solvers require a greater number of assemblies, but of smaller subsystems. The overhead computational expense associated with each assembly can greatly influence which nonlinear method is least computationally expensive. This is code specific and will not be a consideration here. The assembly time is therefore left out when comparing the computational efficiency of the different nonlinear solution methods. The basis of comparison for the nonlinear solvers in this work is the time spent in the linear solver, solving the linear systems developed in the nonlinear methods.

4.3.1   Turbulent Compressible Flow

The first example used to compare the nonlinear solution methods is the turbulent subsonic flat plate found in Section 2.10.1. The results shown here use a rectangular mesh with 136 x 96

elements. The parameters used in this analysis are shown in Table 4.1. For purposes of comparison, MUMPS is used as the linear solver when comparing the three nonlinear solvers. The nonlinear system is broken into four different subsystems, as shown in Table 4.1, for the staggered Newton solver and the nonlinear block Gauss-Seidel solver.

Table 4.1: Analysis parameters used for the subsonic turbulent flat plate nonlinear method comparison

Num of Procs	6
Time Solver	Backwards Euler
Time Stepping Scheme	CFL
Starting CFL	0.05
Max CFL	200.0
Stabilization	$\boldsymbol{\tau}_{SB}$
Shock Capturing	ν
Nonlinear Solver	Multiple
Linear Solver for all	MUMPS
Monolithic Relaxation	0.8
Subsystem 1	
DOFs	$\mathbf{G}_i^n$
Stag Relaxation	1.0
NLBGS Relaxation	1.0
Subsystem 2	
DOFs	$h_{\mathbf{v}}, \nu, \tau_t$
Stag Relaxation	1.0
NLBGS Relaxation	1.0
Subsystem 3	
DOFs	$\rho\tilde{\nu}$
Stag Relaxation	0.8
NLBGS Relaxation	0.25
Subsystem 4	
DOFs	$\rho, \rho v_i, \rho E$
Stag Relaxation	0.8
NLBGS Relaxation	0.25

The turbulent subsonic flat plate problem is a steady-state flow. It is solved in a quasi-steady-state method in which time steps are used to advance the solution. The analysis using all nonlinear solvers are run for three hundred time steps. The real interest from a computational efficiency

standpoint is how quickly the nonlinear residual norm drops as compared to computational effort. The measure of computational effort here is defined by the time spent in the direct linear solver MUMPS. Figure 4.1 shows that in the case of the subsonic turbulent flat plate, the staggered Newton solver is more computationally efficient. The monolithic Newton method requires roughly 4.5 times the amount of computational time as compared to the staggered Newton method for the same number of time steps. Moreover, the monolithic methods stalls in reducing the nonlinear residual. NLBGS reduces the nonlinear residual much less than the other two nonlinear methods for the same number of time steps.
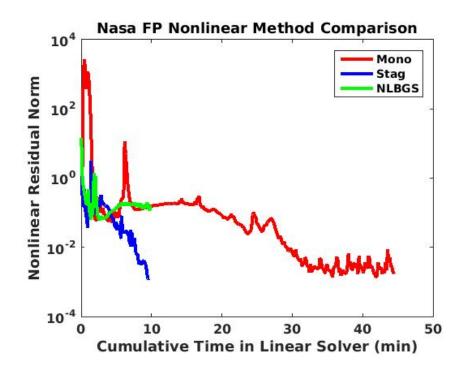
Figure 4.1: Time spent in MUMPS linear solver for the different nonlinear solution methods on the subsonic turbulent flat plate

The time spent in the linear solver, MUMPS, can be broken down into two portions, the factorization time and the time to solve the forward and backward substitution for the solution. Figures 4.2 and 4.3 shows the decomposition of the time spent in the linear solver.
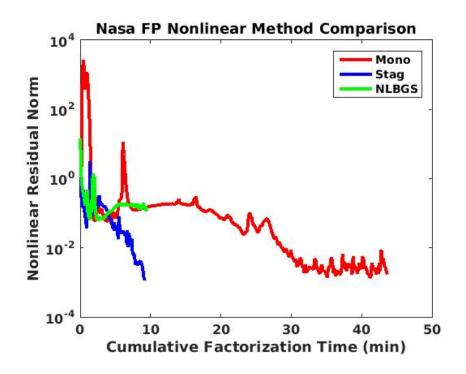
Figure 4.2: Time spent on factorization in MUMPS linear solver for the different nonlinear solution methods on the subsonic turbulent flat plate
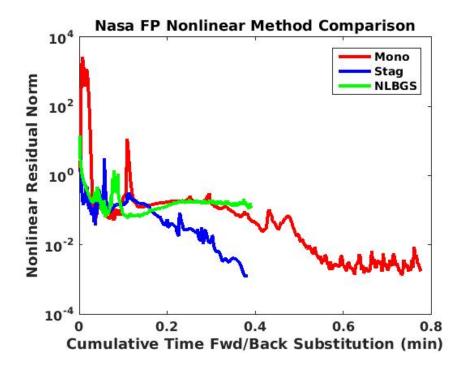
Figure 4.3: Time spent in the forward and backward substitution in MUMPS linear solver for the different nonlinear solution methods on the subsonic turbulent flat plate

Almost all of the time in the direct linear solver, MUMPS, is spent calculating the factorization as one would expect. Table 4.2 shows the size and condition number for the first nonlinear iteration linear approximations. The staggered Newton method and NLBGS both have the same characteristics as they have the same Jacobian for the first iteration and similar condition numbers thereafter. The condition numbers are an approximate estimation of required computational effort to calculate the factorizations and accuracy of the solution.

Table 4.2: Subsonic turbulent flat plate nonlinear method comparison of linear approximation characteristics

Monolithic System	
Number of Dofs	197808
1st Iteration Condition Number	4.1×10^{14}
Subsystem 1	
Number of Dofs	93023
1st Iteration Condition Number	9.9×10^{5}
Subsystem 2	
Number of Dofs	39867
1st Iteration Condition Number	9.9×10^{5}
Subsystem 3	
Number of Dofs	12943
1st Iteration Condition Number	2.6×10^{6}
Subsystem 4	
Number of Dofs	51975
1st Iteration Condition Number	2.1×10^{8}

The staggered Newton method and NLBGS both produce smaller systems will smaller condition numbers as compared to the monolithic Newton method. Therefore the computational effort to solve each linear system is smaller for the two staggered nonlinear methods. This is exactly what one would expect. The disparity between the staggered Newton method and NLBGS is because of the required under-relaxation. Table 4.1 shows that NLBGS requires more under-relaxation. The under-relaxation used with NLBGS is critical to convergence and not know a priori[9]. The off-diagonal weak coupling in NLBGS provides more nonlinearity in the residual and therefore requires more under-relaxation. The time spent to solve each linear system is the same for both NLBGS

and staggered Newton method, but the nonlinear residual norm is reduced less by NLBGS due to more under-relaxation. These results also show that for this specific example the off-diagonal coupling between the subsystems does not help computational efficiency. The staggered Newton method does not take into account any off-diagonal coupling, NLBGS utilizes weak off-diagonal coupling and monolithic Newton uses strong off-diagonal coupling. The monolithic Newton solver stalls in reducing the nonlinear residual norm. The condition number for the linear approximations increase by up to three orders of magnitude as the flow develops and the time step is increased. This is the case for all three nonlinear methods, with the exception of subsystems one and two in the staggered methods. The monolithic Newton method may stall in reducing the nonlinear residual norm because of the increased ill-conditioning of the linear approximation. The monolithic Newton method results in linear approximations with condition numbers greater than 10^{15}, meaning that linear solution accuracy is not reliable. The stalling may also be attributed to the increased nonlinearity of the developed flow. This could potentially be mitigated by using an adaptive under-relaxation. The results show that even if an adaptive under-relaxation was used to mitigate the stalling of the monolithic Newton method, it would still not be more computational efficient than the staggered Newton method.

Figure 4.4 shows all computational time for this example, to include the assembly time. This is not a completely equitable comparison due to the lack of optimized assembly code for each routine, as discussed in Section 4.3.
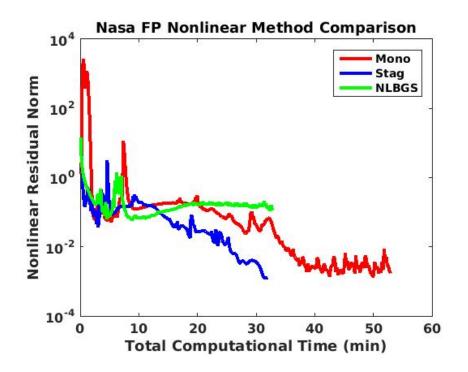
Figure 4.4: Total computational time, including assembly, for the different nonlinear solution methods on the subsonic turbulent flat plate

Figure 4.4 is shown because it offers a reference point. If the assembly routines were optimized for the partitioned solvers, then these methods would have improved efficiencies beyond what is shown here. Therefore, Figure 4.4 and Figure 4.1 can be considered a sort of bounds on the expected computational time, for this example problem, of a code optimized for one nonlinear method. No code will completely be able to eliminate assembly time and achieve only the results in Figure 4.1, but a code optimized for efficiency would see improvements, specifically for the partitioned solvers, above that in Figure 4.4. Even with the less than optimal assembly routine, Figure 4.4 shows that a staggered Newton nonlinear solver offers considerable computational benefit over nonlinear block Gauss-Seidel and a monolithic Newton solver, for this example problem.

4.3.2   Incompressible FSI

The second example used to compare the nonlinear solution methods is Turek's incompressible fluid-structure interaction problem found in Section 3.2. The parameters used in this analysis are

shown in Table 4.3. For the purpose of comparison, the same direct parallel solver, MUMPS, is used for all three analyses. Also, the coupling strategy used is the same for all three analyses. The linear system is broken into four different subsystems, as shown in Table 4.3, for the staggered Newton and the nonlinear block Gauss-Seidel methods. The first subsystem contains all structural deformation DOFs. The second subsystem contains all incompressible fluid DOFs and fluid mesh deformation DOFs that reside on the interface between the structure and fluid domains. The third subsystem contains all fluid mesh deformation DOFs that are not on the interface boundary. The fourth subsystem contains all incompressible fluid DOFs that are not on the interface boundary. All other parameters are the same as identified in Section 3.2.1.

Table 4.3: Analysis parameters used for the Turek FSI3 problem nonlinear method comparison

Num of Procs	6
Time Solver	Newmark Second Order
Coupling Strategy	Struc-MeshVel-Point
Nonlinear Solver	Multiple
Max NL Iter per Time Step	15
Linear Solver for all	MUMPS
Monolithic Relaxation	1.0
Subsystem 1	
DOFs	$\mathbf{u}^s$
Relaxation	1.0
Subsystem 2	
DOFs	$p, \mathbf{v}, \mathbf{u}^f$ on $\mathbf{\Gamma_{fs}}$
Relaxation	1.0
Subsystem 3	
DOFs	$\mathbf{u}^f$ on $\mathbf{\Omega}$
Relaxation	1.0
Subsystem 4	
DOFs	$p, \mathbf{v}$ on $\mathbf{\Omega}$
Relaxation	1.0

The Turek incompressible FSI problem is a time dependent problem, so the presentation of data is different than that for the last example. Each time step is driven down to the same nonlinear residual norm. The computational efficiency here is still defined as the time spent in the linear

solver summed over all nonlinear iterations. This computational time is plotted against the time step in the analysis. The following results show that the partitioned solution methods diverge before completing the analysis. The failure of these methods is dependent on the coupling strategy and the partitioning of the complete set of DOFs into subsystems. The coupling strategy and system partitioning identified in Table 4.3 is the best performing combination found for both partitioned solvers. The partitioned nonlinear solvers are unable to solve even one time step for some other coupling strategies. The failure of the partitioned nonlinear solvers makes the monolithic Newton nonlinear solver the most robust option for this example problem and formulation. Figure 4.5 shows that even without the failures of the partitioned solvers the monolithic method is the most computationally efficient for the Turek FSI problem.
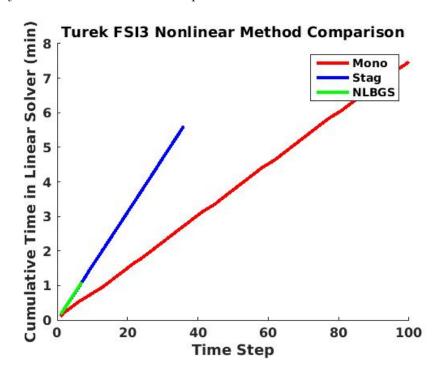
Figure 4.5: Time spent in the MUMPS linear solver for the different nonlinear solution methods on the Turek FSI problem

Figure 4.6 shows that the increased time in the linear solver is due to the number of linear solves required for each nonlinear method. The partitioned nonlinear solvers use the maximum

allowed nonlinear solver iterations for each time step without being able to drop the nonlinear residual norm to the required convergence criteria.
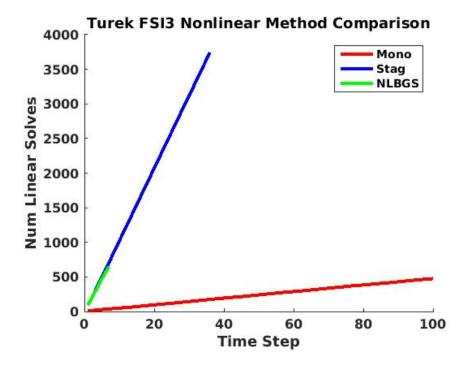
Figure 4.6: Time spent applying the forward and backward substitution in MUMPS linear solver for the different nonlinear solution methods on the Turek FSI problem

The nonlinear residual norm experiences oscillations due to nonlinearity that prevents the partitioned methods from converging each time step. There is no under-relaxation for the results shown here, as identified in Table 4.3. For this example the under-relaxation parameter needs to be reduced to below 0.4 in order to eliminate the nonlinear oscillations. Doing so results in well over a hundred and fifty nonlinear iterations per time step. This exacerbates the computational disparity already identified. For both partitioned solvers, there is no under-relaxation parameter which could be found for this example that reduces the nonlinear iterations to below the maximum allowed for this analysis. Given that the monolithic Newton solver is able to achieve convergence for each time step in an average of six Newton iterations, there is no benefit to explore much higher iteration counts for the partitioned nonlinear solvers. This is supported by previous research which

shows that the partitioned solvers lack robustness and efficiency, for strongly coupled systems, particularly for ill-conditioned systems[9, 10]. Table 4.4 shows that the monolithic Newton produces ill-conditioned linear approximations. Despite the monolithic Newton method having larger more ill-conditioned linear approximations than the partitioned solvers, it is more computationally efficient for this example because it accounts for the strong off-diagonal coupling between subsystems. Accounting for this strong coupling is important to the convergence rates for this example.

Table 4.4: Turek FSI3 nonlinear method comparison of linear approximation characteristics

Monolithic System	
Number of Dofs	106340
1st Iteration Condition Number	1.0×10^{18}
Subsystem 1	
Number of Dofs	2856
1st Iteration Condition Number	1.8×10^5
Subsystem 2	
Number of Dofs	1242
1st Iteration Condition Number	3.4×10^9
Subsystem 3	
Number of Dofs	40888
1st Iteration Condition Number	2.7×10^6
Subsystem 4	
Number of Dofs	61354
1st Iteration Condition Number	6.9×10^{11}

4.3.3   Selection of Nonlinear Solver

Much like the coupling strategy, there is no one most efficient solver. The fluid-structure interaction problem above is characterized by being highly coupled, in part due to the choice of coupling methods. The monolithic Newton method better accounts for this strong off-diagonal coupling. The fluid-structure interaction problem presented here benefits from better accounting for the coupling. The turbulent flat plate on the other hand benefits from solving the subsystem in a staggered approach and does not rely as heavily on the strong off-diagonal coupling blocks for convergence. The choice of nonlinear solver is problem dependent. Efficiency can be gained by

having a library of nonlinear solvers to choose from.

Chapter 5

Linear Solvers

The purpose of the nonlinear solver is to produce a linear approximation about a solution guess. A robust and accurate linear solver is critical to solving the linear approximation. The linear system, shown in Equation 5.1, consists of a matrix of coefficients, $\mathbf{A}$, a right-hand side vector, $\mathbf{b}$, and the solution vector to be found, $\mathbf{x}$.

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{5.1}$$

The linear systems arising from finite element analysis problems tend to be large sparse linear systems. There are many algorithms developed to solver linear systems. These are largely categorized to be either direct linear solvers or iterative linear solvers. This chapter is not intended to be an exhaustive comparison of different linear solvers. Instead, a few examples are used to explore the influence of the linear solver in solving compressible turbulent flow problems via SUPG finite element analysis and incompressible fluid-structure interaction problems.

5.1 Direct Solvers

Direct linear solvers are attractive because they can theoretically find the exact solution for a linear system, up to round-off errors. Direct linear solvers also tend to be more robust than iterative linear solvers[12]. The down side is that they require more floating point operations and more memory than their iterative counterparts. Many direct sparse linear solvers are decomposition methods in which the matrix of coefficients is replaced by an equivalent, or approximate, set of matrices that have a particular structure. They solve the linear system using the same four generic

steps[48]. The first step is to reorder the matrix rows and columns such that it has an optimal structure for the subsequent steps. This can be done through different ordering schemes and is generically shown by the application of a row permutation matrix, as shown in Equation 5.2. When only the rows are reordered then it is known as partial pivoting, while if both the rows and columns are reordered then it is referred to as full pivoting.

$$\mathbf{PAx} = \mathbf{Pb} \tag{5.2}$$

The second step is a symbolic factorization of the matrix which determines the non-zero structure of the factors. The third step is the numerical factorization of the matrix in which the actual values in the factors are determined. All of the direct sparse linear solvers in this work utilize a lower/upper (LU) triangle decomposition. The LU decomposition is comprised of finding a factorization where the reordered coefficient matrix is equal to a lower triangle matrix multiplied by an upper triangle matrix, as shown in Equation 5.3 for a system with n degrees of freedom.

$$\begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix} = \mathbf{PA} \tag{5.3}$$

The fourth and final step is the application of the factored matrices in a forward and back substitution in order to solve the linear system. The forward substitution solves Equation 5.4 for the interim solution, $\mathbf{y}$.

$$\begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \mathbf{y} = \mathbf{Pb} \tag{5.4}$$

The backward substitution then solves Equation 5.5 for the solution, $\mathbf{x}$.

$$\begin{bmatrix} u_{11} & u_{12} & \ldots & u_{1n} \\ 0 & u_{22} & \ldots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & u_{nn} \end{bmatrix} \mathbf{x} = \mathbf{y} \tag{5.5}$$

The forward and backward substitution is relatively computationally cheap to solve as it only requires $\mathcal{O}(2n^2)$ floating point operations for dense matrices where as the LU decomposition requires $\mathcal{O}(\frac{2}{3}n^3)$ floating point operations for dense matrices. This makes these methods particularly efficient when required to solve using multiple right-hand sides, b, because each additional right-hand side vector only requires an additional forward and backward solve. All of the direct linear solvers used in this work use sparse matrices and work work with general non-symmetric matrices. The Amesos[64, 65] package, part of the Trilinos[29] project, is used in this work to access the direct linear solvers.

Serial direct solvers utilize only one processor and the memory associated with that processor. This limits the applicability to small problems of mostly academic relevance for finite element problems. While less relevant to larger problems, a serial direct solver is used in this work for comparison purposes. The serial direct solver used here is UMFPACK[20] which uses a multifrontal approach to achieve an LU decomposition.

Parallel direct sparse linear solvers partition the large sparse linear system and distribute the pieces among different processors. These methods are attractive because they distribute the workload among a number of processors. This comes at the expense of communication between processors. For large problems, the communication expense is outweighed by the benefit of distributed workload. This research compares SuperLU-Dist and MUMPS. SuperLU-Dist is a parallel version of SuperLU, which uses right-looking Gauss elimination with static pivoting to achieve the LU decomposition[49, 50, 51, 21]. MUMPS is a parallel multifrontal approach to achieve an LU decomposition[4, 5]. Both parallel direct sparse linear solvers are designed for distributed memory parallel machines.

5.1.1   Behavior of Direct Solvers for Ill-conditioned Systems

As previously identified, the NASA turbulent compressible flat plate problem and the Turek fluid-structure interaction problem, FSI3, produced ill-conditioned linear approximations using the formulations contained in this research. The NASA flat plate analysis requires the solution of linear systems with a Jacobian that has condition numbers over 10^{17} for the monolithic system. The Turek problem experiences condition numbers over 10^{20}. Therefore, it is necessary to understand how the different linear solvers are able to solve systems which are badly conditioned.

The first set of results shown use a randomly generated 3D Laplacian, discretized using a seven-point stencil and resulting in a 6000 x 6000 matrix, $\mathbf{A}$, which has a condition number of 3.4×10^5. This well-conditioned system is used as a basis of comparison. The matrix is generated and then multiplied by a vector of ones to create a right-hand side, $\mathbf{b}$. This makes a vector of ones the known solution, $\mathbf{x}_{\text{exact}}$. The linear solver is then used to solve the created linear system one hundred times, each time comparing the solution vector to the known solution of ones. The vector norm of the difference is plotted for each run. The error for UMFPACK and the serial version of SuperLU are shown in Figures 5.1 and 5.2. The results show error on the order of 10^{-14}.
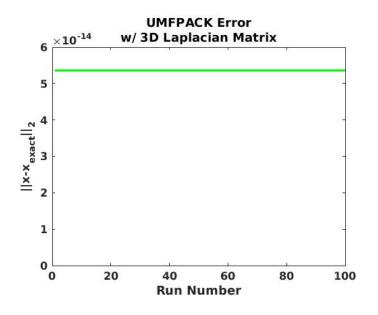
Figure 5.1: Error for UMFPACK with a 3D Laplacian matrix
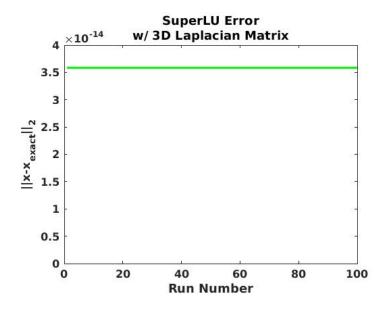
Figure 5.2: Error for SuperLU with a 3D Laplacian matrix

The same process is applied to the distributed version of the SuperLU solver and MUMPS using six cores. Figures 5.3 and 5.4 show that the error distribution is on the order of 10^{-13} for the distributed solvers. Importantly, the distributed solvers experience some variation in the error. The variation in the error is due to the non-deterministic behavior of the Basic Linear Algebra Subroutines (BLAS) library used to compile the distributed direct solvers. The distributed direct solvers are compiled using a multi-threaded version of AMD Core Math Library (ACML). Multi-threaded and parallel BLAS libraries experience non-deterministic behavior for non-associative operations such as multiplication and addition[38, 19, 60, 6]. The variation in solution is further exacerbated by the loss of precision. As discussed in Section 2.11, the logarithm of the condition number for a matrix can roughly be considered the number of digits of precision lost. So, using double precision and the identified matrix with a condition number on the order of 10^5, one can only reliably expect ten to twelve digits of precision. The variation in Figures 5.3 and 5.4 is due to the multi-threaded ACML BLAS and amplified by the loss of precision. The error for the distributed version of SuperLU is slightly higher than MUMPS, although they are both on the same order of magnitude. The distributed version of SuperLU also has one less significant digit as seen by the variation in the error.
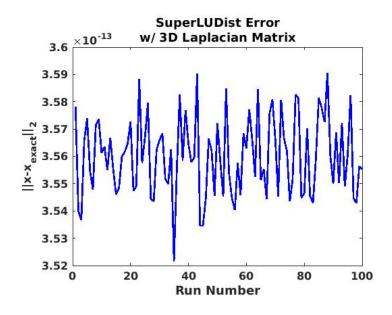
Figure 5.3: Error for SuperLU-Dist with a 3D Laplacian matrix distributed over six cores
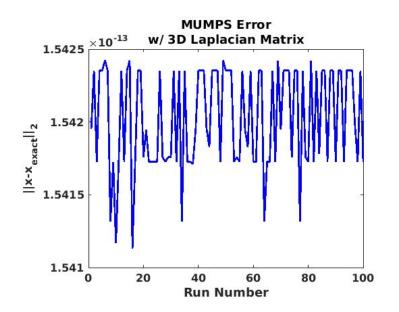
Figure 5.4: Error for MUMPS with a 3D Laplacian matrix distributed over six cores

The parallel direct linear solvers considered have error on the order of 10^{-13} and the serial direct linear solvers considered have error on the order of 10^{-14} for a well conditioned system with a condition number on the order of 10^5. Those results set a basis of comparison to understand how the same linear solvers behave with an ill-conditioned system. For the following set of results, a

matrix taken from one Newton iteration of the NASA turbulent flat plate problem is used. This is a square matrix with 19693 DOFs and a condition number of 3.3×10^{18}. The matrix is multiplied by a vector of ones to create the right-hand side, resulting in a vector of ones being the exact result, as done before.
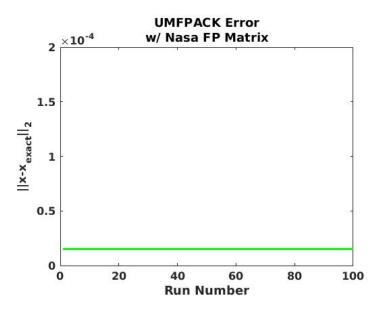
Figure 5.5: Error for UMFPACK with a compressible turbulent flow matrix
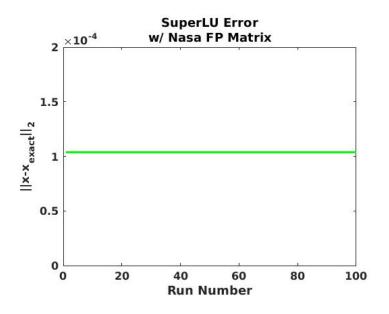
Figure 5.6: Error for SuperLU with a compressible turbulent flow matrix

Figures 5.5 and 5.6 show that UMFPACK has an order of magnitude less error than the serial version of SuperLU. Neither solver experiences any variation in the error, even with the more ill-conditioned matrix.
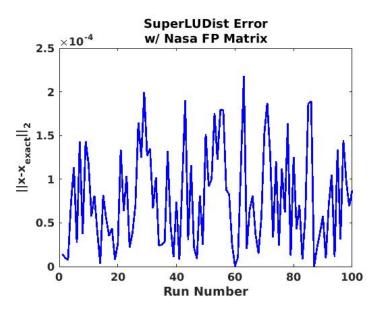
Figure 5.7: Error for SuperLU-Dist with a compressible turbulent flow matrix distributed over six cores
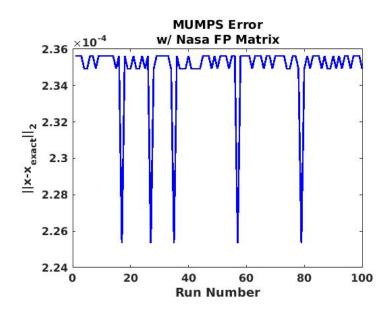
Figure 5.8: Error for MUMPS with a compressible turbulent flow matrix distributed over six cores

Figures 5.7 and 5.8 show that for the ill-conditioned matrix, MUMPS has up to an order of magnitude more error than the distributed version of SuperLU. SuperLU continues to have more variation in the solution and has error that can differ from one run to the next by an order of magnitude. So, for this specific matrix the distributed version of SuperLU is more accurate, but MUMPS is more precise. The amount of error determines the accuracy, the amount of variation in the solution indicates the precision of the solution. The results for the serial version of SuperLU are only provided here to demonstrate that the variation in the solution is an artifact of the ACML BLAS libraries used to compile the distributed direct solvers studied and not the serial direct solvers. Given that UMFPACK is more accurate and neither solver has any variation in the solution, UMFPACK is the only serial direct linear solver used for further comparisons.

5.2   Iterative Solvers

Iterative solvers are attractive because they require less memory and solution time than their direct linear solver counterparts. Iterative solvers scale much better with problem size, especially for 3-D flows, than direct solvers do[12]. Iterative solvers will converge in a number of iterations less than or equal to the number of degrees of freedom in the system. Iterative solvers tend to be less robust than direct linear solvers though. GMRES[61, 67] is the only iterative method used in this research and is accessed through the Trilinos AztecOO package[29]. GMRES is part of a class of iterative solvers called Krylov subspace solvers. The nth Krylov subspace is shown in Equation 5.6.

$$K_n = K_n(A, b) = \text{span}\left\{b, Ab, A^2b, \ldots, A^{n-1}b\right\} \tag{5.6}$$

Krylov subspace solvers consists of making successive solution guesses that are linear combinations of the Krylov subspace. GMRES is a particular Krylov subspace method that uses Arnoldi iteration to build and use orthonormal basis vectors that span the Krylov subspace. It creates a linear least squares problem to minimize the norm of a residual based on the original system and a linear combination of the orthonormal basis vectors. GMRES is generally effective on non-symmetric

problems and is a good general purpose parallel iterative linear solver.

5.2.1   Preconditioning

"It is widely recognized that preconditioning is the most critical ingredient in the development of efficient solvers for challenging problems in scientific computation, and that the importance of preconditioning is destined to increase even further[12]." The convergence of iterative solvers is greatly dependent on the condition number of the matrix used[59]. Using an ill-conditioned matrix without preconditioning can cause iterative solvers to converge very poorly or even fail to converge. It is therefore critical to effectively precondition the system to be able to efficiently and consistently obtain a solution. In general, there is no general purpose preconditioner which is known a priori[12]. Different preconditioning techniques need to be explored for each problem.

Preconditioning a linear system is a method by which the original linear system is modified such that it is easier to solve. The preconditioned system should have a lower condition number than the original system. Left preconditioning is shown in Equation 5.7 and right preconditioning is shown in Equation 5.8.

$$\mathbf{P}^{-1}\mathbf{A}\mathbf{x} = \mathbf{P}^{-1}\mathbf{b} \tag{5.7}$$

$$\mathbf{A}\mathbf{P}^{-1}\mathbf{P}\mathbf{x} = \mathbf{b} \tag{5.8}$$

The preconditioner is a method to compute $\mathbf{P}^{-1}$, or more often a method to compute the product of a vector with $\mathbf{P}^{-1}$. The ideal preconditioner is $\mathbf{P} = \mathbf{A}$, as it would yield an identity matrix on the left hand side of Equation 5.7, but this is as expensive to compute as the inverse of the original matrix, therefore not saving any computational effort. The goal is to find a preconditioner $\mathbf{P}^{-1}$ that best balances the computational effort to compute with how accurately it approximates the actual inverse.

The preconditioners utilized in this work are part of the Trilinos packages IFPACK[63] and ML[24]. The three classes of preconditioners used in this work are incomplete factorizations and

point relaxation found in IFPACK, and algebraic multigrid found in ML. Point relaxation methods are fairly straight forward and not used very often. Point Jacobi relaxation is used in this work only for the subsystems which are diagonal matrices. In this work, these arise when using the partitioned nonlinear methods and neglecting off-diagonal contributions as discussed in Section 2.11. Point Jacobi is not further discussed here, because it is not often used.

5.2.2   Incomplete Factorization Preconditioners

Incomplete factorization methods require little user input and are easy to use preconditioners. These methods consist of computing an incomplete lower and upper (ILU) approximate factorization of the original matrix, as shown in Equation 5.9 for a system with n unknowns. Often times this decomposition uses either a unit-lower or a unit-upper matrix in which the diagonal entries of the respective matrix are one. The exact LU decomposition is computationally expensive and contains a large number of off-diagonal terms, requiring significant memory. The incomplete factorization aims at reducing both the computational time and the memory requirement.

$$
\mathbf{P} = \begin{bmatrix} l_{11} & 0 & \dots & 0 \\ l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & \dots & u_{1n} \\ 0 & u_{22} & \dots & u_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & u_{nn} \end{bmatrix} \approx \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \tag{5.9}
$$

At the most basic an incomplete factorization uses the nonzero structure of the original matrix and only computes the these terms in the factorized matrices, which is also known as zero fill-in. This is commonly referred to as ILU(0), and a basic algorithm for this can be seen in Algorithm 4. This algorithm produces a unit-lower and an upper approximate decomposition for the original matrix. Most often either partial-pivoting or full-pivoting is used to reorder the matrix for efficiency and robustness prior to the decomposition, but this algorithm is shown for the natural ordering of the matrix.

Algorithm 4 An Algorithm for Incomplete LU Decomposition With Zero Fill-in ILU(0)

1: Let $l_{ii} = 1 \quad \forall \ i \leq n$
2: Let $u_{11} = a_{11}$
3: If $u_{11} = 0$ Then STOP ▷ Factorization Impossible
4: **for** $j = 2, ..., n$ **do**
5:      IF $a_{1j} \neq 0$ THEN $u_{1j} = a_{1j}/l_{11}$
6:      IF $a_{j1} \neq 0$ THEN $l_{j1} = a_{j1}/u_{11}$
7: **end for**
8: **for** $i = 2, ..., n-1$ **do**
9:      $u_{ii} = a_{ii} - \sum_{k=1}^{i-1} l_{ik}u_{ki}$
10:      If $u_{ii} = 0$ Then STOP ▷ Factorization Impossible
11:      **for** $j = i+1, ..., n$ **do**
12:          IF $a_{ij} \neq 0$ THEN $u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik}u_{kj}$
13:          IF $a_{ji} \neq 0$ THEN $l_{ji} = \frac{1}{u_{ii}} \left[a_{ji} - \sum_{k=1}^{i-1} l_{jk}u_{ki} \right]$
14:      **end for**
15: **end for**
16: Let $u_{nn} = a_{nn} - \sum_{k=1}^{n-1} l_{nk}u_{kn}$ ▷ If $u_{nn} = 0$ Then **A** is singular

Algorithm 4 is a generic algorithm for a dense matrix factored on a single processor. Modifications are made to compute the factorization of a sparse matrix so that large gains in efficiency can be made, but at the expense of algorithm complexity. This type of ILU preconditioner is the least accurate, least computationally expensive and requires the least amount of memory of the ILU types. To increase the accuracy and robustness, the factorization is allowed to have additional off-diagonal terms not contained in the nonzero structure of the original matrix, known as fill-in. The amount of additional terms allowed is controlled through the level of fill parameter. No additional fill would be ILU(0). As the level of fill, k, is increased, ILU(k) becomes a more accurate approximation of the exact factorization at the expense of memory and computational time. In general terms, for a level of fill $k > 0$, each fill-in element of the decomposition is assigned a level based on the graph, or non-zero structure, of the original matrix. Then, each fill-in element of the decomposition with a level greater than k is dropped.

For indefinite matrices, ILU(k) may not optimally account for the size of elements when dropping fill-in elements[62]. A drop tolerance or threshold can be defined and used in conjunction with the level of fill. It is assumed that terms below this threshold have little influence on the result and therefore can be dropped for the sake of efficiency. ILUT(p, τ) uses both a defined level

of fill and threshold tolerance to compute the LU factorization. Generically, ILUT(p, τ) computes the factorization, drops fill-in terms that are below the threshold, τ, and then keeps only the p largest fill-in elements in a row and column. This of course is made more complex when using sparse matrices. This method results in a changing nonzero structure of the preconditioner as the value of the terms computed are not known a priori.

This research is concerned with parallel computing for efficiency. ILU(k) and ILUT(p, τ) require the processor to have access to the entire original matrix, which means it is computed on a single processor. Additive Schwarz domain decomposition is a method which can be used to create a parallel preconditioner using serial algorithms on the subdomains. Additive Schwarz consists first of parsing the linear system into m subdomains as shown in Equations 5.10 and 5.11[63].

$$\mathbf{P}_{\mathrm{A}}^{-1} = \sum_{i=1}^{m} \mathbf{P}_i \mathbf{A}_i^{-1} \mathbf{R}_i \tag{5.10}$$

$$\mathbf{A}_i = \mathbf{R}_i \mathbf{A} \mathbf{P}_i \tag{5.11}$$

Here $\mathbf{R}_i$ is the operator that restricts the rows of $\mathbf{A}$ to only those included in the local subdomain and $\mathbf{P}_i$ is the operator that prolongates the local subdomain into the global matrix. In using FEMDOC, the computational domain is decomposed and distributed among the processors in a pre-processing step. The local subdomain, $\mathbf{A}_i$ then consists of all rows for the DOFs which live on the local processor. The local linear system can then be solved using the method of choice, incomplete LU factorizations in this work. The local problem with zero overlap is shown in Equation 5.12, where the local residual vector, $\mathbf{r}_i$, and local solution vector, $\mathbf{z}_i$ both reside on the local processor as well.

$$\mathbf{z}_i = \mathbf{A}_i^{-1} \mathbf{r}_i \tag{5.12}$$

For zero overlap, the columns in $\mathbf{A}_i$ which correspond to off-processor DOFs are dropped, leaving a square matrix. This also means there is no required communication between processors. The local problem can be augmented with off processor information by specifying a domain overlap greater

than zero. In this case, the local problem is augmented with off-processor information as shown in Equation 5.13, where e denotes external processor owned DOFs.

$$\mathbf{z}_i = \mathbf{P}_i \begin{bmatrix} \mathbf{A}_i & \mathbf{A}_{ie} \\ \mathbf{A}_{ei} & \mathbf{A}_{ee} \end{bmatrix}^{-1} \mathbf{R}_i \mathbf{r}_i \tag{5.13}$$

For one level overlap, the local matrix is augmented with all off-processor rows corresponding to non-zero entries in the off-processor columns of the local matrix rows. The new rows which are imported then bring new off-processor column entries. So, for subsequent levels of overlap, the same process is applied and the off-processor rows which correspond the new off-processor columns are imported. For an overlap greater than zero, communication between processors is needed to import matrix rows and residual vector entries and export solutions to the owning processor. This communication comes at a computational expense. The reader is referred to the IFPACK documentation[63] and Cai and Sarkis[15] for more details, but suffice it to say that an increased overlap produces a preconditioner which is a better approximation of the inverse of the original matrix, at a computational and communication expense. The amount of overlap is set to one for all of the following fill comparisons.

5.2.2.1   ILU Level of Fill Comparison

The level of fill used in ILU preconditioning influences both the efficiency and accuracy of the iterative solver. The following results show one set of examples of how influential it can be. The Turek FSI3 benchmark problem is used for comparison. Three different formulations of the fluid-structure coupling discussed in Section 3.1.4 are shown for comparison. The plots are titled consistent with the naming convention in Section 3.1.4 as well.

The first level of fill comparison is with the traction condition stored in the structural momentum equations, fluid velocities coupled to the mesh velocities and the coupling is integrated over the interface boundary. Figure 5.9 shows that as the level of fill is increased, the amount of computational time spent in the linear solver increases as well. This is in part due to the fact that the increased fill requires more computational time to compute the preconditioner. Recall that the

ILU level of fill is based on the graph of the original matrix. So, the off-diagonal fill-in entries are only computed if they are less than or equal to the level of fill chosen. More fill-in requires more floating point operations to compute those entries.
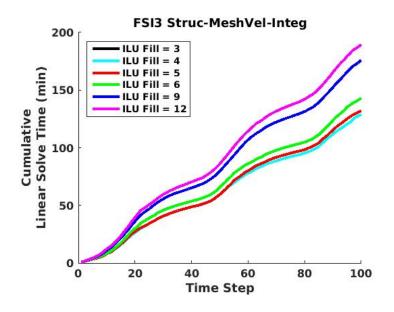
Figure 5.9: ILU level of fill comparison of linear solver time using Turek FSI3 problem

The increased cost for computing the preconditioner is not the only contributing factor to the difference in efficiency. As the level of fill is increased, the nonlinear solution method begins to converge more slowly, particularly for the highest fill levels. This can be seen in Figure 5.10, indicated by the fact the higher fills require more linear solves to converge the nonlinear residual.
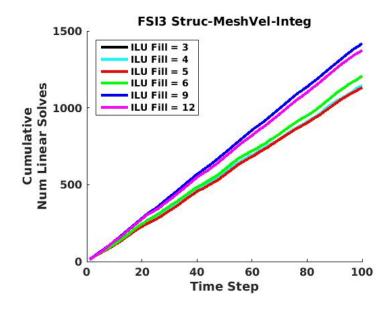
Figure 5.10: ILU level of fill comparison of number of linear solves needed using Turek FSI3 problem

This indicates that the nonlinear problem benefits from the smoothing of an approximate Jacobian. For too little fill, as for a level of fill of three, the approximate decomposition is not accurate enough, and this results in solutions states which are not physically possible. This includes the computation of negative element Jacobian determinants. So, while the linear solver is able to converge, the calculation of the nonlinear residual fails. This can possibly be mitigated by lowering the tolerance for the convergence of the linear solver, but for this example this results in stalled convergence for the nonlinear solver.

The second comparison has the traction condition stored in the fluid momentum equations, the fluid velocities are coupled with the structural velocities and the displacement and velocity conditions are integrated. Figure 5.11 shows these results, which are very similar to Figure 5.9 with one notable exception.
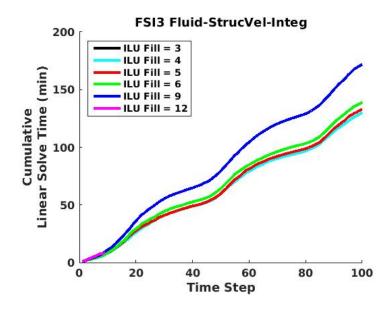
Figure 5.11: ILU level of fill comparison of linear solver time using Turek FSI3 problem

The larger level of fill of twelve causes the nonlinear solution method convergence to slow too much. This causes the nonlinear solution method to reach the maximum number of allowed iterations before converging. This is interesting because with the different coupling conditions, and therefore a slightly different non-zero structure of the Jacobian matrix, the nonlinear convergence is more sensitive to the fill-in of the approximate decomposition. This means that one cannot just err on the side of more a more computationally expensive preconditioner and expect good results, but rather there is more of a window than a floor for the best choice of preconditioner. Too little fill-in or too much fill-in can both cause the nonlinear residual convergence to stall or diverge.

The third comparison is similar to the last, Figure 5.11, except that the traction condition is stored in the fluid momentum equations. Figure 5.12 demonstrates even more clearly that there is a window of level of fill to find. The four failures in this analysis are all due to the divergence of the nonlinear solution method. The three lowest level of fills produce approximations which are too inaccurate, while the largest level of fill produces an approximation which also suffers.
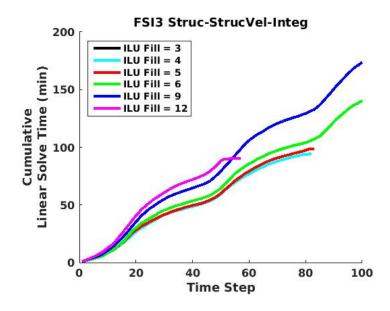
Figure 5.12: ILU level of fill comparison of linear solver time using Turek FSI3 problem

This fluid-structure interaction example can benefit from the use of an approximate linearization. An approximation which is too inaccurate can cause the nonlinear solver to diverge. Likewise, the use of a preconditioner which has too much fill-in can cause the nonlinear solver to diverge as well. The the formulation of the coupling conditions does have some influence on the nonlinear solver convergence sensitivity to the accuracy of the linear approximation. The range of effective fill-ins is not something that can be known a priori, but rather discovered through experience.

5.2.2.2   ILUT Level of Fill Comparison

The same comparison done for ILU level of fill in Section 5.2.2.1, is done for ILUT. The drop tolerance is set to 10^{-12} for all comparisons. This drop tolerance is chosen in order to have an approximation close to that produced by ILU(k) for similar fill-ins. All else being equal, only the level of fill is changed during the comparisons. Again, the Turek FSI3 benchmark is used for the comparisons. The range used for level of fill here is from one to six and the sequence of comparisons is the same as in Section 5.2.2.1.

Figure 5.13 shows that none of the levels of fill examined result in a completed analysis. For a level of fill of one, this example fails to converge the linear residual or the nonlinear residual. Both

the linear solver and the nonlinear solver reach the maximum allowed iterations before convergence. This demonstrates that using $\text{ILUT}(1, 10^{-12})$ results in too inaccurate an approximation. For levels of fill two and above, the nonlinear solution eventually diverges, resulting in an interim solution which is not physically possible. Namely, the mesh displacements cause an inverted element, resulting in the computation of a negative element Jacobian determinant.
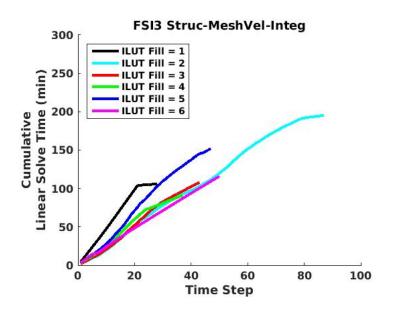
Figure 5.13: ILUT level of fill comparison of linear solver time using Turek FSI3 problem

Increasing the level of fill for ILUT does not necessarily mean increasing the computational time spent in the linear solver. This is because increasing the level of fill does increase the computational cost of the preconditioner, but it also results in fewer iterations of the iterative linear solver. This can bee seen in Figures 5.14 and 5.15. This is the trend one expects to see. The key to finding the optimally efficient selection is balancing these two competing trends.
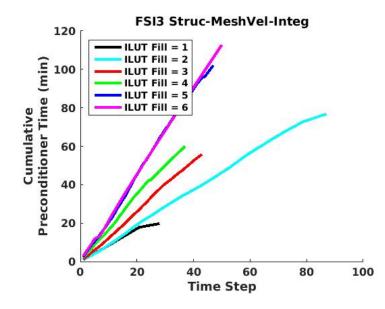
Figure 5.14: ILUT level of fill comparison of time to compute preconditioner using Turek FSI3 problem
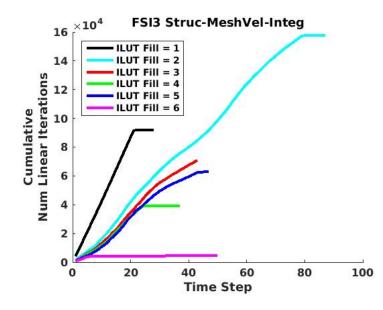
Figure 5.15: ILUT level of fill comparison of number of linear iterations needed using Turek FSI3 problem

Figure 5.16 shows that a level of fill of two allows the analysis to complete the full one hundred time steps. This also confirms that there is not a very clear relationship between the level of fill and the computational time spent in the linear solver and preconditioner. Here again, a level of fill of

one is too inaccurate an approximation, resulting in the failure of the linear and nonlinear solvers to converge within the maximum iterations. A level of fill of three or more causes a divergence of the nonlinear problem.
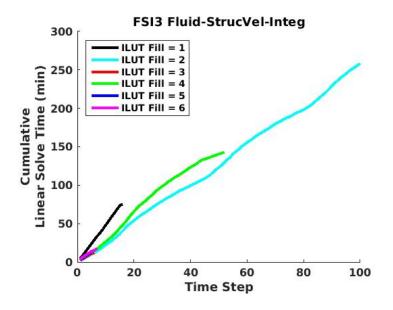
Figure 5.16: ILUT level of fill comparison of linear solver time using Turek FSI3 problem

Figure 5.17 also shows results similar to Figure 5.16. A level of fill of two works, while all others attempted fail for the same reasons as discussed previously.
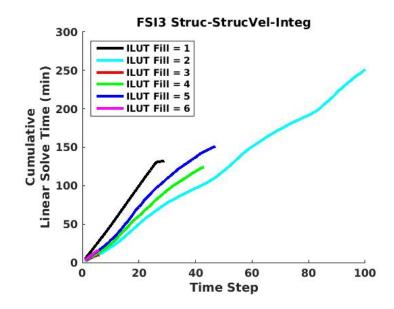
Figure 5.17: ILUT level of fill comparison of linear solver time using Turek FSI3 problem

These results invite the question of how the drop tolerance and diagonal perturbation can improve the performance. These parameters influence the performance of the preconditioner in much the same way. They can not be know a priori and there is a very narrow window for success. To verify this, one example of a drop tolerance sweep is provided in Figure 5.18.
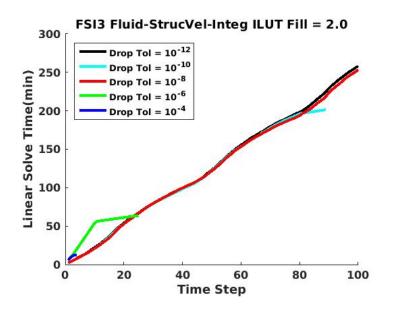
Figure 5.18: ILUT drop tolerance comparison of linear solver time using Turek FSI3 problem

This shows the performance of the ILUT can be even more sensitive to the drop tolerance. Drop tolerances of 10^{-8} and 10^{-12} result in successful completion of the analysis, while 10^{-10} results in a failure. The analysis using the drop tolerance of 10^{-10} is able to converge the nonlinear and linear solvers for eighty-one times steps. Then on the next time step, unexpectedly the nonlinear problem diverges, causing a solution which leads to inverted elements. The other failures are due to a failure of the linear and nonlinear solvers to converge in the maximum allowed iterations. This result highlights that it is not only the amount of fill-in and the size of the retained fill-ins which matter, but also the location of fill-ins is also very important to the accuracy and robustness of the approximate decomposition.

These results, along with those for ILU(k) are specific to this example used. This cannot, in general, be applied to another problem. They are shown here to highlight four things. First, these results show that these preconditioners are sensitive to the parameters chosen. Secondly, it is generally agreed upon that these parameters can not be known a priori[63]. Thirdly, these results corroborate that incomplete factorizations tend to be unstable[63]. Finally, these parameter sweeps demonstrate just how narrow a margin for success there is in setting these parameters for this example problem. This supports the idea that this fluid-structure example is a difficult problem to solve. This leads one to find a more sophisticated preconditioner in order to improve robustness and efficiency.

5.2.3   Smoothed Aggregation

Multigrid uses a series of more and more coarse grids, hence the term multigrid, to accelerate the solution. The multigrid solution is obtained using these grids in what is commonly referred to as a multigrid V cycle. Algorithm 5 shows a high level example of a V cycle[24]. The V cycle starts at the finest grid level. If pre-smoothing is used, then a relatively computationally inexpensive linear solution method, denoted S^1, is used to solve the linear system. The error is then projected onto the next coarser grid and provided to the next level as the right-hand side for the linear system. This continues until reaching the coarsest level, at which time the solution is obtained using the

coarse level solver. The coarse level solution is then projected onto each finer grid and at each level post-smoothing can be applied using a relatively computationally cheap linear solution method, S^2. To overview, the error is transferred from the finest grid down to the coarsest grid and a solution is obtained on the coarsest grid. The solution is then transferred from the coarsest grid back up to the finest grid. At each level, if selected, the error can be smoothed on the way down the V cycle using a pre-smoother, S^1. Likewise the solution can smoothed on the way up the V cycle using a post-smoother, S^2. All of this can be seen in Algorithm 5 where N_{levels} is the number of levels used, k is the level number, and $\mathbf{P}_k$ is the projection operator for each level. Lines five and six in Algorithm 5 show that this particular algorithm uses the transpose of the projection operator as the restriction operator. This is often the case, but in general a different restriction operator can be utilized.

Algorithm 5 Mulitgrid V Cycle[24]

1: **procedure** MULTILEVEL($\mathbf{A}_k, \mathbf{b}, \mathbf{u}, k$) $\triangleright$ Solve $\mathbf{A}_k\mathbf{u} = \mathbf{b}$
2:      $u = S^1_k(\mathbf{A}_k, \mathbf{b}, \mathbf{u})$ $\triangleright$ Apply pre-smoothing
3:      **if** $k \neq N_{\text{levels}} - 1$ **then** $\triangleright$ Not on coarsest level
4:          $\mathbf{P}_k = \text{DetermineInterpolant}(\mathbf{A}_k)$ $\triangleright$ Create grid transfer operator
5:          $\hat{\mathbf{r}} = \mathbf{P}_k^T(\mathbf{b} - \mathbf{A}_k\mathbf{u})$ $\triangleright$ Interpolate error to next level
6:          $\hat{\mathbf{A}}_{k+1} = \mathbf{P}_k^T\mathbf{A}_k\mathbf{P}_k$ $\triangleright$ Calculate matrix for next level
7:          $\mathbf{v} = 0$ $\triangleright$ Initialize solution for next level
8:          MULILEVEL($\hat{\mathbf{A}}_{k+1}, \hat{\mathbf{r}}, \mathbf{v}, k + 1$) $\triangleright$ Process next level
9:          $\mathbf{u} = \mathbf{u} + \mathbf{P}_k\mathbf{v}$ $\triangleright$ Interpolate solution from coarser level
10:         $\mathbf{u} = S^2_k(\mathbf{A}_k, \mathbf{b}, \mathbf{u})$ $\triangleright$ Apply post-smoothing
11:     **end if**
12: **end procedure**

The way in which the hierarchy of grids and associated transfer operators are constructed is one thing that distinguishes between multigrid methods. Geometric multigrid uses information obtained from the actual mesh. Algebraic multigrid[70, 80] uses the properties of the coefficient matrix to create the hierarchy of grid transfer operators. One advantage of algebraic multigrid is that it can much more easily be used on unstructured meshes since no information from the mesh is needed.

The smoothed aggregation contained in ML[24] is a type of algebraic multigrid preconditioner.

What distinguishes smoothed aggregation from other algebraic multigrid methods is that the grid transfer operators, or prolongators, $\mathbf{P}_k$, are themselves smoothed when the hierarchy is created. For detailed information on smoothed aggregation methods the reader is referred to [74, 75, 2, 58].

The ML package defaults to using a damped Jacobi iteration to smooth the prolongators. Another option, which is exclusively used in this work, is to determine the coarse level prolongators by solving an energy minimization problem. The energy based minimization is "suitable for highly convective nonsymmetric fluid flow problems[24]." The goal of the energy based minimization method is to find a set of basis vectors contained in the coarse level prolongators which have a minimal energy. The details on the energy minimization algorithm can be found in [53]. The research contained here exclusively uses METIS[41] to provide the aggregation information. METIS uses the nonzero structure of the matrix and strength of entries to create a graph of the matrix. The work in this research provides the number of partial differential equations, so that the degrees of freedom at each node can be grouped. This allows for a block matrix graph to be created, which can lead to better convergence rates[24]. METIS then uses the block matrix graph to determine an optimal aggregation for the matrix. This information is provided to the smoothed aggregation algorithm to create the tentative prolongators before they are smoothed using the energy minimization method.

An advantage of this type of preconditioner is that a more robust and computationally expensive direct linear solver can be used on a much more coarse grid. This saves memory and computational effort over using the direct linear solver as the main liner solver. This comes at the expense of setting up and computing the multigrid hierarchy. The goal is to leverage the robustness of direct linear solvers and the computational efficiency of iterative linear solvers. There are an infinite number of ways for setting up the smoothed aggregation preconditioner. An in depth study of smoothed aggregation is not done here, but it is utilized in this work for the NASA compressible turbulent flat plate problem.

5.3   Linear Solver Comparisons

The choice of linear solver is highly dependent on the problem. Each class of problem has linear solvers that are tuned to produce the the most efficient results for it. This comparison is not intended to show which linear solver is best, but rather which linear solver is best for each of these examples. This work produces a framework within FEMDOC which is utilized to explore the best solution for complex multi-physics problems. Results are displayed consistent with Section 4.3.

5.3.1   Turbulent Compressible Flow

The formulation of the compressible turbulent flow by SUPG finite element analysis results in ill-conditioned linear approximations that require solving. As discussed in Chapter 4, the choice of nonlinear solution method also impacts how ill-conditioned the linear approximations are. This section investigates how the selection of the linear solver has an impact on the analysis, as each solver performs differently for the same ill-conditioned linear system. Section 5.1.1 shows that the two distributed direct solvers can experience a variation in solution due to the loss of precision associated with ill-conditioning, while UMFPACK does not have any variation in the solution due to the ill-conditioning of the matrix. Section 2.12 shows that SUPG compressible finite element analysis is highly nonlinear and very sensitive to the loss of precision caused by ill-conditioned linear approximations. The first set of results investigates how the linear solver selection can impact the NASA turbulent flat plate example. To show the impact of the linear solver selection, the nonlinear residual norm is plotted at each time step as previously done. The parameters used for this comparison are identified in Table 5.1.

Table 5.1: Analysis parameters used to compare linear solvers for the subsonic turbulent flat plate

Num of Procs	6
Time Solver	Backwards Euler
Time Stepping Scheme	CFL
Starting CFL	0.05
Max CFL	200.0
Stabilization	$\boldsymbol{\tau}_{SB}$
Shock Capturing	ν
Nonlinear Solver	Monolithic Newton
Linear Solver	Various
Relaxation	0.8

Figure 5.19 shows that solving the monolithic system using UMFPACK as the linear solver results in no variation in the three runs. This is consistent with previous results and shows that UMFPACK is able to solve the ill-conditioned linear approximations, obtaining the same solution to within machine precision, despite the ill-conditioned linear approximations. To be clear, this does not mean the solutions are accurate to within machine precision, but rather the solution is the same each time to within machine precision. Furthermore, when the solution files are compared, the results for every DOF at every node for every time step are exactly the same, to within a tolerance of 10^{-16}.
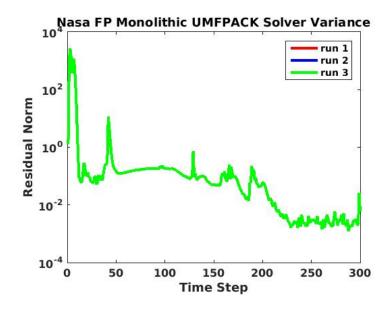
Figure 5.19: Monolithic nonlinear residual norm variation using UMFPACK

Solving the same problem using MUMPS as the linear solver and distributed over six cores does result in variation as shown in Figure 5.20. This shows that the nonlinear residual norm can vary by three orders of magnitude.
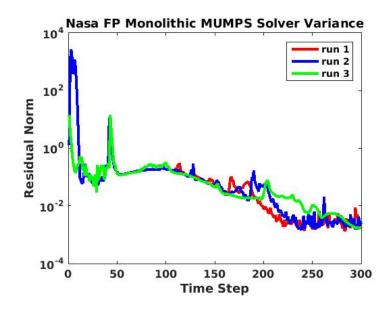
Figure 5.20: Monolithic nonlinear residual norm variation using MUMPS

Figure 5.21 shows the same results using SuperLU-Dist as the linear solver. The result is only

shown for fifty iterations because the analysis fails. The analysis fails because the loss of precision results in a solution which is not physically possible, to include 0.0 density and negative internal energy at various locations.
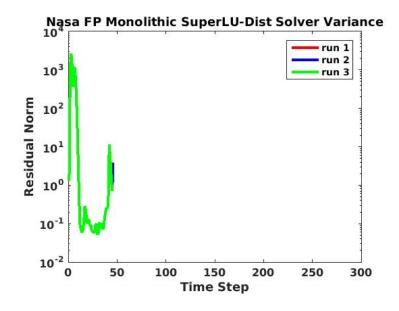

Figure 5.21: Monolithic nonlinear residual norm variation using SuperLU-Dist

The difference in DOFs for the first and second analysis using the distributed version of SuperLU is shown in Figure 5.22. This shows the initial difference in solutions is very small, but after a while the error growth increases dramatically, resulting in a solution which is not physically possible.
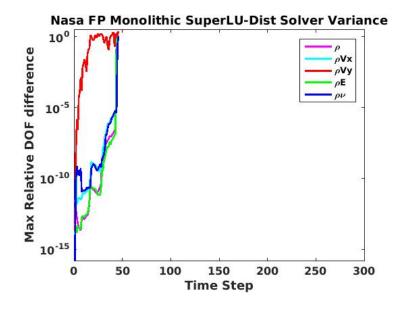
Figure 5.22: Maximum DOF difference between solution 1 and 2 using SuperLU-Dist

These results are shown only to highlight how the performance of each linear solver can suffer from ill-conditioning in different ways. Each of the two parallel direct linear solvers have a loss of precision in the solution due to ill-conditioning. Highly nonlinear problems such as SUPG compressible finite element can be very sensitive to this loss of precision, causing the nonlinear problem to converge to different solutions or potentially even failure of the nonlinear solution. These results give more insight into the nonlinear problem than to the linear solvers themselves, but they do stress the importance of exploring different linear solvers for very difficult problems.

A comparison between, UMFPACK, MUMPS, the distributed version of SuperLU, and GM-RES is made using the parameters in Table 5.2. The comparison done here is consistent with previous comparisons of the turbulent flat plate example. The basis for comparison is the computational time in the linear solver as compared to the drop in the nonlinear residual norm. One could compare the time of each linear solver to solve the same linear system. One could also compare the error of each linear solver when solving the same linear system. While this may be a more direct comparison of the linear solvers it is not of interest here for the following reason. The solution coming out of each linear solver is different and this has an influence on the nonlinear convergence. A more accurate solution may actually cause the nonlinear convergence to slow or

possibly even cause is to diverge. Therefore, choosing a linear solver that is more accurate or more computationally efficient for one linear system is not a good indicator as to how the selection of this linear solver will influence the overall efficiency or robustness of the analysis. So, while comparing the nonlinear residual norm versus computational time takes into account more factors than just the linear solver selection, it is a better comparison of how the linear solver selection impacts the computational efficiency for this example. These results only apply to this example problem and a similar study would need to be done for other problems.

Table 5.2: Analysis parameters used for the subsonic turbulent flat plate linear solver comparison

Num of Procs	6
Time Solver	Backwards Euler
Time Stepping Scheme	CFL
Starting CFL	0.05
Max CFL	200.0
Stabilization	$\boldsymbol{\tau}_{SB}$
Shock Capturing	ν
Nonlinear Solver	Staggered Newton
Linear Solver For All Subsystems	Multiple
Subsystem 1	
DOFs	$\mathbf{G}_i^n$
Relaxation	1.0
Preconditioner	Point Jacobi
Subsystem 2	
DOFs	$h_{\mathbf{v}}, \nu, \tau_t$
Relaxation	1.0
Preconditioner	Point Jacobi
Subsystem 3	
DOFs	$\rho\tilde{\nu}$
Relaxation	0.8
Preconditioner	Additive Schwarz & ILUT
Overlap	0
Fill	5.0
Drop Tolerance	10^{-6}
Subsystem 4	
DOFs	$\rho, \rho v_i, \rho E$
Relaxation	0.8
Preconditioner	Smoothed Aggregation
Smoother Type	Additive Schwarz & ILU
Overlap	2
Fill	6
Maximum Levels	4
Aggregation Type	METIS
Nodes per Aggregate	60
Pre or Post Smoothing	Post
Coarse Solver	UMFPACK or MUMPS
Energy Minimization Enabled	True
Preconditioner	Additive Schwarz & ILU
Level of Fill	3
Preconditioner	Additive Schwarz & ILUT
Fill	2.0
Drop Tolerance	10^{-6}

Table 5.2 shows that point Jacobi is used as the preconditioner for the first two subsystems. This is because these subsystems are used for solving the nodal reconstruction DOFs, discussed in Section 2.7, and are therefore very well conditioned linear systems. Point Jacobi is a computationally inexpensive preconditioner for well conditioned systems. The turbulence equation in subsystem three is also well conditioned enough to be solved using the more simple black box type preconditioner, ILUT. The fluid equations in subsystem four, however, produce more ill-conditioned linear approximations which are more difficult to solve. Therefore three different preconditioners are used and compared for this subsystem. The parameters for the three different preconditioners used on the fluid subsystem are all shown in Table 5.2. The preconditioner parameters shown in Table 5.2 are only used in conjunction with GMRES as the linear solver. The parameters identified in Table 5.2 are those which are found to produce the best results. The parameters shown for the smoothed aggregation result in three levels for the multilevel method. Level zero, the orginal matrix, contains 53156 global rows, level one contains 880 global rows, and level two, the coarsest level, contains 24 global rows. As shown in Section 5.1.1, each direct linear solver will produce a solution with different amounts of error, depending on the conditioning of the system. Therefore, for the analyses using a parallel direct linear solver, the best of three runs is used for comparison in the results below.
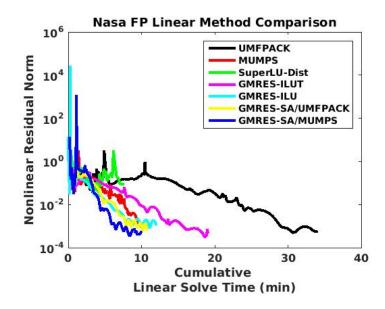
Figure 5.23: Comparison of the total time spend in the linear solver for subsonic turbulent flat plate

For the comparisons shown in Figure 5.23, SuperLU-Dist fails to complete the analysis. Reducing the relaxation did not help SuperLU-Dist. Just as in Figure 5.21, there are no analysis parameters found which allows SuperLU-Dist to complete the analysis. The failure is due to solutions which have large error resulting from ill-conditioned Jacobians. These solutions are not physically possible, such as 0.0 density, and as such result in failed nonlinear residual calculations. Figure 5.23 also shows that the iterative solver, GMRES, paired with smoothed aggregation is able to achieve the best efficiency. The following results show a breakdown of the time in the linear solver in order to understand where the computational benefit comes from.
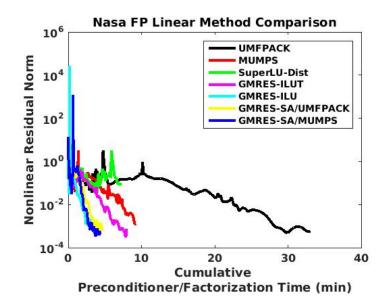
Figure 5.24: Comparison of the time spent calculating the preconditioner or factorization for subsonic turbulent flat plate

Figure 5.24 shows that the preconditioners associated with GMRES take much less time, up to an order of magnitude, to compute the preconditioner than the direct methods require to perform the LU factorization. This is what one would expect. As previously discussed, it is well known that the factorization time is the more computationally expensive part of the direct methods. UMFPACK takes much more because it is a serial algorithm completed on a single processor. If one were needing to solve using multiple right-hand sides, then it is expected that direct solvers may then be more computationally efficient, because the preconditioner would not need to be refactored. The factorization expense is amortized over the number of right-hand sides.
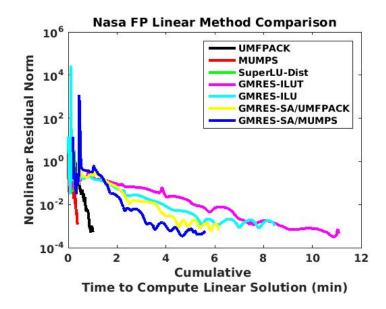
Figure 5.25: Comparison of the time spent in the forward/backward substitution for the direct solvers or iterating for GMRES for subsonic turbulent flat plate

Figure 5.25 shows that direct solvers take much less time to obtain the solution. For the direct solvers, this is the time for the forward and backward substitution. For GMRES, this is the time iterating within GMRES. These results are not surprising either as the forward and backward substitution are fairly computationally cheap. Figure 5.26 shows the cumulative number of linear iterations needed for each of the preconditioners. This explains why smoothed aggregation is the more computationally efficient preconditioner for this example. Figure 5.24 shows that smoothed aggregation is the cheapest preconditioner to compute and Figure 5.26 shows that it does the best job reducing the linear residual in fewer iterations.
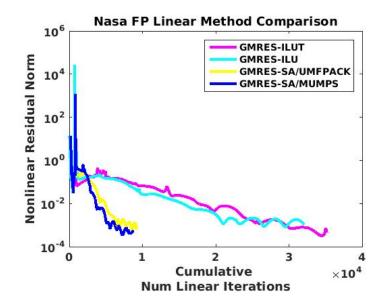
Figure 5.26: Comparison of linear solver iterations for subsonic turbulent flat plate

These results show that smoothed aggregation was able to be configured successfully to achieve the accuracy benefits of using a direct linear solver while capitalizing on the computational efficiency benefits of an iterative solver. It is known that additive Schwarz methods are not scalable and convergence suffers when more processors are used[63]. Smoothed aggregation may be a viable path to achieve scalability and good convergence rates.

5.3.2   Incompressible FSI

The Turek incompressible fluid-structure interaction problem is a highly coupled multiphysics problem. The high degree of coupling produces linear systems with strong off-diagonal contributions, which makes this a problem of interest for linear solver comparison. The most efficient run for each of the direct linear solvers is used, likewise the most efficient fill is used for each of the incomplete factorization preconditioners. The applicable parameters are found in Table 5.3, but not every parameter applies to each analysis. Smoothed aggregation is not utilized here either. Section 4.3.2 showed that the Turek problem suffers when using partitioned nonlinear solvers, therefore a monolithic Newton nonlinear method is used. Smoothed aggregation greatly benefits from a block graph partitioning. The monolithic system does not have the same number

of equations at each node and therefore block graph partitioning cannot be used. A successful set
of parameters for smoothed aggregation using point graph partitioning has yet to be found for this
problem. UMFPACK is not used for this analysis because these results all use a monolithic Newton
nonlinear solver and the monolithic system is too large and requires too much memory, such that
UMFPACK cannot be utilized.

Table 5.3: Analysis parameters used for the Turek FSI3 problem linear method comparison

Num of Procs	6
Time Solver	Newmark Second Order
Coupling Strategy	Fluid-StrucVel-Integ
Nonlinear Solver	Monolithic Newton
Relaxation	1.0
Linear Solver	Multiple
ILU Fill	4
ILUT Fill	2.0
ILUT Drop Tolerance	10^{-12}
Domain Decomposition Overlap	1

Figure 5.27 shows that for this example parallel direct linear solvers are much more efficient
than GMRES with incomplete LU factorization, with MUMPS being the most efficient. The follow-
ing results decompose the time in the linear solver to better understand where the computational
benefit comes from.
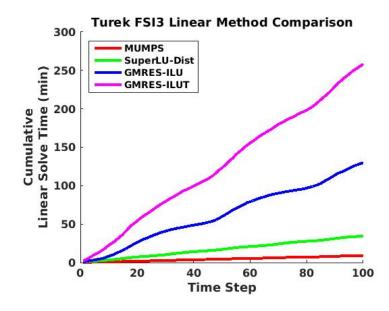
Figure 5.27: Comparison of the total time spend in the linear solver for Turek FSI3 problem

Figures 5.28 shows that iterative methods require much less time to do the forward and backward solve than preconditioned GMRES takes to iterate using these preconditioners. This is consistent with what one would expect to see.
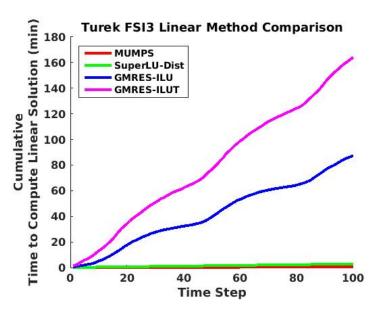
Figure 5.28: Comparison of the time spent calculating forward/backward solve for direct solvers or iterating in GMRES for Turek FSI3 problem

Figure 5.29 reveals that the number of linear solves required from each linear solver is a large contributor to the decreased computational efficiency of the iterative method. Each linear solve of the iterative solvers does not drop the nonlinear residual norm as much as the direct solvers. The incomplete factorizations used for preconditioning are an inaccurate approximation to the original Jacobian and therefore slow the convergence of nonlinear solver. The parameter sweeps in Sections 5.2.2.1 and 5.2.2.1 show that increasing the fill does not provide a computational benefit, as the fills used here are the most computationally efficient. The results here, along with those in Sections 5.2.2.1 and 5.2.2.1, also show that increasing the fill-in does not necessarily translate to a more accurate incomplete factorization. The results shown here reveal that this problem benefits from the accurate and complete factorization used in the direct linear solvers. So, if increasing the fill-in of the incomplete factorizations strictly meant a more accurate preconditioner, one would expect the increased fill-in to result in fewer linear solves, as seen with the direct solvers, but this is not the case for the results shown in Sections 5.2.2.1 and 5.2.2.1.
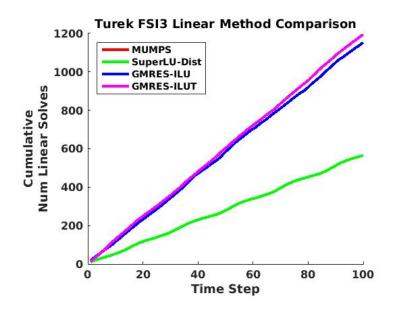

Figure 5.29: Comparison of the number of linear solves needed for Turek FSI3 problem

Figure 5.30 reveals that the two preconditioned GMRES solutions require more time to calculate the preconditioner than the direct linear solvers require to calculate the factorizations.
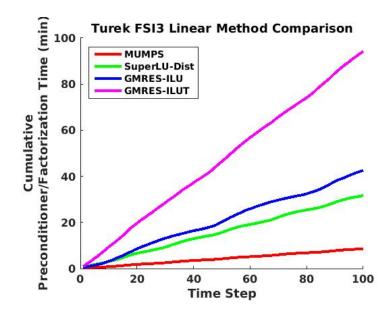
Figure 5.30: Comparison of the time spent calculating the preconditioner or factorization for Turek FSI3 problem

Figure 5.31 shows that the time to compute the factorization in MUMPS is less than the time to compute either of the preconditioners. This is surprising as it is expected that the direct solver factorization would be more computationally expensive. Two possible contributing factors are communication requirements and scalability. The difference in time between the preconditioners and the factorizations are small and may be attributed to how efficiently the communication between processors happens for each of the algorithms. Secondly, it may just be that the additive Schwarz method is suffering from poor scalability and MUMPS is not. The scalability of additive Schwarz is a known problem[63]. More investigation into this needs to happen to get to the root cause of the efficiency disparity.

Figure 5.31: Comparison of the average computational cost of calculating the preconditioner or factorization for Turek FSI3 problem

Chapter 6

Conclusions

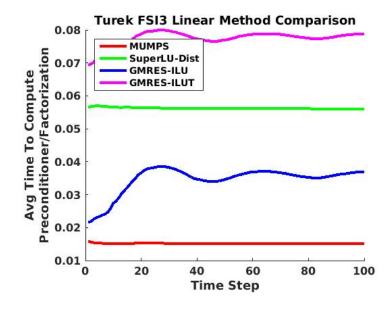The goal of this research was to develop a computationally efficient method for compressible turbulent fluid-structure interaction optimization which would be extensible to large design problems. Using finite elements to discretize the entire domain is an attractive path in order to achieve that goal. The experience using SUPG finite element analysis for compressible flows during this research reveals that it is not currently robust enough to be used for optimization or large problems requiring parallel linear solvers with distributed memory.

Chapter 2 thoroughly explains the implementation of the SUPG finite element compressible flow and the Spalart-Allmaras turbulence equation. It also shows that accurate results are achievable using this method. These accurate results are obtained through trial and error to find the right combination of nonlinear solver, linear solver, domain discretization, domain decomposition, stabilization, and shock capturing. These choices cannot be known a priori. The SUPG finite element method for compressible flow described in this work suffers from ill-conditioned linear approximations which yield linear solutions that lack sufficient precision and accuracy. Furthermore, the equations are very nonlinear, so the insufficient precision and accuracy can cause the nonlinear convergence to slow or even diverge. It is the ill-conditioning and nonlinearity which makes this method very sensitive to the parameters chosen. A fluid-structure interaction optimization problem using an ALE formulation results in a different mesh each design iteration. This does not work for a method which is so sensitive to slight changes. For these reasons, SUPG finite element analysis for compressible flows is not a viable path for optimization or large problems.

An incompressible fluid-structure interaction problem is used to show that although it also has ill-conditioned linear approximations, the nonlinear problem is not as sensitive to the loss of precision in the linear solution. The parallel direct solvers MUMPS and SuperLU-Dist used in this work exhibit a particular sensitivity to the ill-conditioning of the linear approximations, while the serial direct solvers UMFPACK and SuperLU do not suffer the same loss of precision due to ill-conditioning. Serial direct linear solvers, while accurate and robust, are limited in problem size due to memory requirements.

The question of efficiency is largely problem dependent. The choice of nonlinear and linear methods will depend on the characteristic of the flow and the problem formulation. For instance, the choice of formulation for fluid-structure coupling can impact the choice of preconditioners, linear solver, and overall computational efficiency. Anyone doing complex finite element analysis needs to have a library of nonlinear solvers, linear solvers and preconditioners in order to find a combination which is most efficient for a specific problem.

The implementation and utilization of a consistent Jacobian was of interest in this work. There are two reasons for this. The primary interest is to be able to use the consistent Jacobian in calculating exact analytical optimization sensitivities. The hope is that this would lead to efficient gradient based optimization analysis and design. The secondary interest in a consistent Jacobian is in regard to the impact on nonlinear convergence rates of the finite element analysis. A consistent Jacobian leads to a more ill-conditioned matrix further exacerbating the loss of precision and subsequent suffering of the nonlinear convergence. Future work could rigorously investigate which contributions to the Jacobian could be omitted to reduce the ill-conditioning and provide optimal smoothing to the nonlinearity of the problem.

A smoothed aggregation preconditioner proved to be computationally efficient for the NASA turbulent flat plate example. Future work should investigate the many different parameters associated with creating a smoothed aggregation preconditioner. GMRES preconditioned with smoothed aggregation shows promise and more work should be done to determine if it can be tuned even more and how robust of an option it is. Smoothed aggregation should also be applied to a large set of

examples to understand if it can effectively mitigate the problems associated with ill-conditioning and nonlinearity of SUPG compressible finite element. If it is successful on a larger number of examples, then scalability studies should be done to determine the applicability of the smoothed aggregation to large problems formulated with SUPG finite element.

Finally, only one fluid-structure interaction example was used in this work to compare the behavior of SUPG finite element for compressible turbulent flows. The formulation of the coupling between fluid and structure does have some impact on computational efficiency. This study could be applied to a larger set of problems to understand how broadly applicable it is. Fluid-structure interaction problems which have a larger portion of the domain on the interface may be more impacted by the formulation of the coupling conditions. Also, the Turek example used in this work has a high degree of coupling between the fluid and structure. For problems in which the coupling is not as strong, the coupling formulation may not have as large an influence. The relationship between strength of coupling and efficiency of coupling formulations should be further investigated.

Bibliography

[1] NASA turbulence modeling resource. http://turbmodels.larc.nasa.gov/index.html, 2012.

[2] Mark Adams, Marian Brezina, Jonathan Hu, and Ray Tuminaro. Parallel multigrid smoothing: polynomial versus Gauss-Seidel. Journal of Computational Physics, 188(2):593 – 610, 2003.

[3] S. K. Aliabadi and T. E. Tezduyar. Parallel fluid dynamics computations in aerospace applications. International Journal for Numerical Methods in Fluids, 21(10):783–805, 1995.

[4] Patrick R. Amestoy, Iain S. Duff, Jean-Yves L'Excellent, and Jacko Koster. A fully asynchronous multifrontal solver using distributed dynamic scheduling. SIAM Journal on Matrix Analysis and Applications, 23(1):15–41, 2001.

[5] Patrick R. Amestoy, Abdou Guermouche, Jean-Yves L'Excellent, and Stéphane Pralet. Hybrid scheduling for the parallel solution of linear systems. Parallel Computing, 32(2):136 – 156, 2006. Parallel Matrix Algorithms and Applications (PMAA04).

[6] Alyssa Anderson. Achieving numerical reproducibility in the parallelized floating point dot product. Honors Theses, 2014. Paper 30.

[7] Matthias Augustin, Alfonso Caiazzo, André Fiebach, Jürgen Fuhrmann, Volker John, Alexander Linke, and Rudolf Umla. An assessment of discretizations for convection-dominated convection–diffusion equations. Computer Methods in Applied Mechanics and Engineering, 200(47):3395–3409, 2011.

[8] Frank P. T. Baaijens. A fictitious domain/mortar element method for fluid-structure interaction. Int. J. Numer. Meth. Fluids, 35(7):743–761, 2001.

[9] Manuel Barcelos, Henri Bavestrello, and Kurt Maute. Efficient solution strategies for steady-state aeroelastic analysis and design sensitivity analysis. In 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, NY, August 30–September 1, number AIAA-2004-4476, 2004.

[10] Manuel Barcelos, Henri Bavestrello, and Kurt Maute. A Schur-Newton-Krylov solver for steady-state aeroelastic analysis and design sensitivity analysis. Computer Methods in Applied Mechanics and Engineering, 195:2050–2069, 2006.

[11] Manuel Barcelos and Kurt Maute. Aeroelastic design optimization for laminar and turbulent flows. Computer Methods in Applied Mechanics and Engineering, 197(19):1813–1832, 2008.

[12] Michele Benzi. Preconditioning techniques for large linear systems: a survey. Journal of Computational Physics, 182(2):418–477, 2002.

[13] Steven Bova, Ryan Bond, and Benjamin Kirk. Stabilized finite element scheme for high speed flows with chemical non-equilibrium. In Aerospace Sciences Meetings, pages 4–7. American Institute of Aeronautics and Astronautics, 2010.

[14] Richard L. Burden and J. Douglas Faires. Numerical Analysis. Thompson, 2005.

[15] X. Cai and M. Sarkis. A restricted additive Schwarz preconditioner for general sparse linear systems. SIAM Journal on Scientific Computing, 21(2):792–797, 1999.

[16] Lucia Catabriga, Alvaro LGA Coutinho, and Tayfun E Tezduyar. Compressible flow SUPG stabilization parameters computed from degree-of-freedom submatrices. Computational Mechanics, 38(4-5):334–343, 2006.

[17] Stéphane Catris and Bertrand Aupoix. Density corrections for turbulence models. Aerospace Science and Technology, 4(1):1 – 11, 2000.

[18] J. Chung and G. M. Hulbert. A time integration algorithm for structural dynamics with improved numerical dissipation: The generalized-α method. Journal of Applied Mechanics, 60(2):371–375, June 1993.

[19] Sylvain Collange, David Defour, Stef Graillat, and Roman Iakymchuk. Numerical Reproducibility for the Parallel Reduction on Multi- and Many-Core Architectures. working paper or preprint, September 2015.

[20] Timothy A. Davis. Algorithm 832: Umfpack v4.3—an unsymmetric-pattern multifrontal method. ACM Trans. Math. Softw., 30(2):196–199, June 2004.

[21] James W. Demmel, Stanley C. Eisenstat, John R. Gilbert, Xiaoye S. Li, and Joseph W. H. Liu. A supernodal approach to sparse partial pivoting. SIAM J. Matrix Analysis and Applications, 20(3):720–755, 1999.

[22] O.W. Eshbach, B.D. Tapley, and T.R. Poston. Eshbach's Handbook of Engineering Fundamentals. A Wiley-Interscience publication. Wiley, 1990.

[23] Charbel Farhat, Michel Lesoinne, and Nathan Maman. Mixed explicit/implicit time integration of coupled aeroelastic problems: Three-field formulation, geometric conservation and distributed solution. International Journal for Numerical Methods in Fluids, 21(10):807–835, 1995.

[24] M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, and M.G. Sala. Ml 5.0 smoothed aggregation user's guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.

[25] O. Ghattas and X. Li. Domain decomposition methods for sensitivity analysis of a nonlinear aeroelasticity problem. International Journal of Computational Fluid Dynamics, 11(1-2):113–130, 1998.

[26] Philip M. Gresho, Robert L. Lee, Stevens T. Chan, and Robert L. Sani. Solution of the time-dependent incompressible Navier-Stokes and Boussinesq equations using the Galerkin finite element method. In Reimund Rautmann, editor, Approximation Methods for Navier-Stokes

Problems, volume 771 of Lecture Notes in Mathematics, pages 203–222. Springer Berlin Heidelberg, 1980.

[27] G. Hauke. A simple subgrid scale stabilized method for the advection-diffusion-reaction equation. Computer Methods in Applied Mechanics and Engineering, 191(27-28):2925 – 2947, 2002.

[28] Matthias Heil. An efficient solver for the fully coupled solution of large-displacement fluid-structure interaction problems. Computer Methods in Applied Mechanics and Engineering, 193(1-2):1 – 23, 2004.

[29] Michael Heroux, Roscoe Bartlett, Vicki Howle Robert Hoekstra, Jonathan Hu, Tamara Kolda, Richard Lehoucq, Kevin Long, Roger Pawlowski, Eric Phipps, Andrew Salinger, Heidi Thornquist, Ray Tuminaro, James Willenbring, and Alan Williams. An Overview of Trilinos. Technical Report SAND2003-2927, Sandia National Laboratories, 2003.

[30] Nicholas J. Higham and Franoise Tisseur. A block algorithm for matrix 1-norm estimation, with an application to 1-norm pseudospectra. SIAM Journal on Matrix Analysis and Applications, 21(4):1185–1201, 2000.

[31] Hans M Hilber, Thomas JR Hughes, and Robert L Taylor. Improved numerical dissipation for time integration algorithms in structural dynamics. Earthquake Engineering & Structural Dynamics, 5(3):283–292, 1977.

[32] M. Howard. Finite Element Modeling and Optimization of High-Speed Aerothermoelastic Systems. PhD thesis, University of Colorado, 2010.

[33] M. Howard and Kurt Maute. Shape optimization of axisymmetric bodies subject to aerodynamic heating and ablation. 2008.

[34] Micah Howard and Steven Bova. Coupling strategies for high-speed aeroheating problems. In Aerospace Sciences Meetings. American Institute of Aeronautics and Astronautics, 2011.

[35] T. J. R. Hughes and M. Mallet. A new finite element formulation for computational fluid dynamics: Iv. a discontinuity-capturing operator for multidimensional advective-diffusive systems. Computational Methods in Applied Mechanics and Engineering, 58:329–336, 1986.

[36] T. J. R. Hughes, Guglielmo Scovazzi, and Tayfun E. Tezduyar. Stabilized methods for compressible flows. Journal of Scientific Computing, 43:343–368, 2010.

[37] Thomas JR Hughes and TE Tezduyar. Finite element methods for first-order hyperbolic systems with particular emphasis on the compressible Euler equations. Computational Methods in Applied Mechanics and Engineering, 45(1):217–284, 1984.

[38] Roman Iakymchuk, Sylvain Collange, David Defour, and Stef Graillat. ExBLAS: Reproducible and Accurate BLAS Library. working paper or preprint, July 2015.

[39] Kenneth E. Jansen, S. Scott Collis, Christian Whiting, and Farzin Shaki. A better consistency for low-order stabilized finite element methods. Computer Methods in Applied Mechanics and Engineering, 174:153 – 170, 1999.

[40] Nicholas Jenkins and Kurt Maute. Level set topology optimization of stationary fluid-structure interaction problems. Structural and Multidisciplinary Optimization, 52(1):179–195, 2015.

[41] George Karypis and Vipin Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM Journal on Scientific Computing, 20(1):359–392, 1998.

[42] David A Kay, Philip M Gresho, David F Griffiths, and David J Silvester. Adaptive time-stepping for incompressible flow part ii: Navier-Stokes equations. SIAM Journal on Scientific Computing, 32(1):111–128, 2010.

[43] J. Y. Kim, N. R. Aluru, and D. A. Tortorelli. Improved multi-level Newton solvers for fully coupled multi-physics problems. International Journal for Numerical Methods in Engineering, 58(3):463–480, 2003.

[44] Benjamin S. Kirk. Adiabatic shock capturing in perfect gas hypersonic flows. International Journal for Numerical Methods in Fluids, 64(9):1041–1062, 2010.

[45] Benjamin S. Kirk and Graham F. Carey. Development and validation of a SUPG finite element scheme for the compressible Navier-Stokes equations using a modified inviscid flux discretization. International Journal for Numerical Methods in Fluids, 57(3):265–293, 2008.

[46] Sebastian Kreissl. Topology Optimization of Flow Problems Modelled by the Incompressible Navier-Stokes Equations. PhD thesis, University of Colorado, 2011.

[47] Gerald John Le Beau, SE Ray, SK Aliabadi, and TE Tezduyar. SUPG finite element computation of compressible flows with the entropy and conservation variables formulations. Computer Methods in Applied Mechanics and Engineering, 104(3):397–422, 1993.

[48] X. Li. Direct solvers for sparse matrices. http://crd-legacy.lbl.gov/~xiaoye/SuperLU/SparseDirectSurvey.pdf, page 91.

[49] Xiaoye S. Li. An overview of SuperLU: Algorithms, implementation, and user interface. ACM Transactions on Mathematical Software, 31(3):302–325, September 2005.

[50] Xiaoye S. Li and James W. Demmel. SuperLU-DIST: A scalable distributed-memory sparse direct solver for unsymmetric linear systems. ACM Trans. Mathematical Software, 29(2):110–140, June 2003.

[51] X.S. Li, J.W. Demmel, J.R. Gilbert, iL. Grigori, M. Shao, and I. Yamazaki. SuperLU Users' Guide. Technical Report LBNL-44289, Lawrence Berkeley National Laboratory, September 1999. `http://crd.lbl.gov/~xiaoye/SuperLU/`. Last update: August 2011.

[52] Daniel Lichtblau and Eric W. Weisstein. Condition number. From MathWorld—A Wolfram Web Resource, 2016. [Online; accessed 18-January-2016].

[53] J. Mandel, M. Brezina, and P. Vaněk. Energy optimization of algebraic multigrid bases. Computing, 62(3):205–228, 1999.

[54] K. Maute and M. Allen. Conceptual design of aeroelastic structures by topology optimization. Structural and Multidisciplinary Optimization, 27:27–42, 2004.

[55] Kurt Maute, Melike Nikbay, and Charbel Farhat. Coupled analytical sensitivity analysis and optimization of three-dimensional nonlinear aeroelastic systems. AIAA Journal, 39(11):2051–2061, 2001.

[56] Kurt Maute, Melike Nikbay, and Charbel Farhat. Sensitivity analysis and design optimization of three-dimensional nonlinear aeroelastic systems by the adjoint method. International Journal for Numerical Methods in Engineering, 56(6):911–933, 2003.

[57] N. M. Newmark. A method of computation for structural dynamics. Journal of the Engineering Mechanics Division, 85(3):67–94, 1959.

[58] Luke N. Olson and Jacob B. Schroder. Smoothed aggregation multigrid solvers for high-order discontinuous Galerkin methods for elliptic problems. Journal of Computational Physics, 230(18):6959 – 6976, 2011.

[59] A. Pyzara, B. Bylina, and J. Bylina. The influence of a matrix condition number on iterative methods' convergence. In Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on, pages 459–464, Sept 2011.

[60] Todd R. Getting reproducible results with Intel® MKL. 2011.

[61] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems. SIAM Journal on Scientific and Statistical Computing, 7(3):856–869, 1986.

[62] Yousef Saad. ILUT: A dual threshold incomplete LU factorization. Numerical Linear Algebra with Applications, 1(4):387–402, 1994.

[63] M. Sala and M. Heroux. Robust algebraic preconditioners with IFPACK 3.0. Technical Report SAND-0662, Sandia National Laboratories, 2005.

[64] M. Sala, K. Stanley, and M. Heroux. Amesos: A set of general interfaces to sparse direct solver libraries. In Proceedings of PARA'06 Conference, Umea, Sweden, 2006.

[65] Marzio Sala, Kendall S Stanley, and Michael A Heroux. On the design of interfaces to sparse direct solvers. ACM Transactions on Mathematical Software (TOMS), 34(2):9, 2008.

[66] Farzin Shakib, T. J. R. Hughes, and Zdenek Johan. A new finite element formulation for computational fluid dynamics: X. the compressible Euler and Navier-Stokes equations. Computer Methods in Applied Mechanics and Engineering, 89:141–219, 1991.

[67] Farzin Shakib, Thomas J.R. Hughes, and Zdenk Johan. A multi-element group preconditioned gmres algorithm for nonsymmetric systems arising in finite element analysis. Computer Methods in Applied Mechanics and Engineering, 75(1–3):415–456, 1989.

[68] P. R. Spalart and S. R. Allmaras. A one-equation turbulence model for aerodynamic flows. Recherche Aerospatiale, 1(AIAA-Paper 92-0439):5–21, 1994.

[69] Philippe Spalart. Trends in turbulence treatments. In Meeting Paper Archive. American Institute of Aeronautics and Astronautics, 2000.

[70] K. Stüben. A review of algebraic multigrid. Journal of Computational and Applied Mathematics, 128(12):281 – 309, 2001. Numerical Analysis 2000. Vol. VII: Partial Differential Equations.

[71] T. E. Tezduyar and S. Sathe. Stabilization parameters in SUPG and PSPG formulations. Journal of Computational and Applied Mechanics, 4(1):71–88, 2003.

[72] Tayfun E. Tezduyar and Masayoshi Senga. SUPG finite element computation of inviscid supersonic flows with yzβ shock-capturing. Computers & Fluids, 36(1):147 – 159, 2007.

[73] Stefan Turek and Jaroslav Hron. Proposal for numerical benchmarking of fluid-structure interaction between an elastic object and laminar incompressible flow. In Hans-Joachim Bungartz and Michael Schfer, editors, Fluid-Structure Interaction, volume 53 of Lecture Notes in Computational Science and Engineering, pages 371–385. Springer Berlin Heidelberg, 2006.

[74] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. Computing, 56(3):179–196, 1996.

[75] Petr Vanék, Marian Brezina, Jan Mandel, et al. Convergence of algebraic multigrid based on smoothed aggregation. Numerische Mathematik, 88(3):559–579, 2001.

[76] Hongwu Wang and Ted Belytschko. Fluid-structure interaction by the discontinuous-Galerkin method for large deformations. International Journal for Numerical Methods in Engineering, 77(1):30–49, 2009.

[77] C. Wervaecke, H. Beaugendre, and B. Nkonga. A stabilized finite element method for compressible turbulent flows. In V European Conference on Computational Fluid Dynamics ECCOMAS CFD 2010, 2010.

[78] C. Wervaecke, H. Beaugendre, and B. Nkonga. A fully coupled RANS Spalart-Allmaras SUPG formulation for turbulent compressible flows on stretched-unstructured grids. Computer Methods in Applied Mechanics and Engineering, 233-236(0):109 – 122, 2012.

[79] Wikipedia. Condition number, 2016. [Online; accessed 18-January-2016].

[80] Yang and Ulrike Meier. Boomeramg: a parallel algebraic multigrid solver and preconditioner. Applied Numerical Mathematics, 41(1):155–177, 2002.

Nomenclature

$()_\infty$ Freestream values

$()_n$ Nodal value

$()_{gp}$ Gauss point value

$\bar{\mathbf{R}}$ Modified residual

Δt Time step

δ, ν Shock capturing parameter

δ_{ij} Kronecker delta

$\dot{u}^s$ Structural velocity

Γ Fluid domain surface

γ Ratio of specific heats

$\hat{n}_i$ Unit normal vector

$\hat{\mathbf{v}}$ Velocity unit vector

$\hat{\mathbf{W}}$ Modified test functions

κ Fluid thermal conductivity

$\mathbf{A}_0^{-1}$ Transformation matrix from entropy to conservative variables

$\mathbf{x}_n$ Spacial coordinates for node n

$\mathbf{A}_i$ Inviscid flux Jacobians

$\mathbf{F}_i^{ale}$ ALE inviscid flux correction

$\mathbf{F}_i$ Inviscid fluxes

$\mathbf{G}_i$ Viscous fluxes

$\mathbf{S}$ Vector of source terms

$\mathbf{U}$ Vector of conservative variables

$\mathbf{W}$ Test functions

$\mathcal{C}$ Range of computational coordinates

$\mathbf{M}$ Mass Matrix

$\mathbf{N}$ Element shape function

$\mathbf{R}_f$ Strong form of the fluid residual

$\mathbf{R}_p$ Residual for nodally reconstructed(projected) values

$\mathbf{\Gamma_{fs}}$ Fluid-structure interface boundary

$\mathbf{\Gamma_f}$ Fluid surface not including the fluid-structure interface

$\mathbf{\Gamma_s}$ Structural surface not including the fluid-structure interface

$\mathbf{\Omega_f}$ Fluid domain

$\mathbf{\Omega_s}$ Structural domain

$\mathbf{\Phi}$ Vector of all nodally reconstructed values

$\boldsymbol{\tau}_{supg}$ SUPG stabilization term

μ	Fluid viscosity (Turbulent + Laminar)
μ_l	Laminar viscosity
μ_t	Turbulent viscosity
ν	Kinematic viscosity
Ω	Fluid domain volume
Ω^e	Element volume
ϕ	A generic nodally reconstructed value
ρ	Local density
τ_c	Stabilization scaling for the conservation of mass equation
τ_e	Stabilization scaling for the conservation of energy equation
τ_m	Stabilization scaling for the conservation of momentum equations
τ_t	Stabilization scaling for the turbulence equation
τ_{ij}	Fluid deviatoric stress tensor
$\tilde{\nu}$	Spalart-Allmaras transported variable, eddy viscosity
ξ, η, ζ	Element natural coordinates
C	Structural damping
c	Speed of sound
c_f	Coefficient of friction
c_p	Specific heat for constant pressure
c_v	Specific heat for constant volume

d Distance to nearest wall

E Local total energy per unit mass

g^{ij} Contra-variant metric tensor

g_{ij} Co-variant metric tensor

H Total enthalpy per unit mass

$h_{\mathbf{v}}$ Flow aligned element length scale

h_{shock} Density gradient aligned element length scale

K_n Nth Krylov subspace

n_d Number of dimensions

n_e Number of elements

n_n Number of nodes

p Local fluid pressure

Pr Prandtl number

q_i Heat flux vector

R Gas constant

R_m Fluid mesh motion strong form of the residual

R_s Structural strong form of the residual

Re Reynolds number

S^c Source term in continuity equation

S^e Source term in the energy equation

S_i^m Source term in momentum equation

T Local fluid temperature

t Time

u, u^s Structural displacement

u^+ Non-dimensional turbulent velocity output variable

u^f Fluid mesh displacement

v^f Fluid velocity

v_i Fluid velocity in spacial direction

v_i^m Mesh velocity components

x_i Spacial direction

Y Structural test functions

y^+ Non-dimensional turbulent wall distance output variable

CFL   Courant-Friedrichs-Lewy number