ABSTRACT

THOMPSON, KYLE B. Aerothermodynamic Design Sensitivities for a Reacting Gas Flow Solver on an Unstructured Mesh Using a Discrete Adjoint Formulation. (Under the direction of Hassan Hassan and Peter Gnoffo.)

An algorithm is described to efficiently compute aerothermodynamic design sensitivities using a decoupled variable set. In a conventional approach to computing design sensitivities for reacting flows, the species continuity equations are fully coupled to the conservation laws for momentum and energy. In this algorithm, the species continuity equations are solved separately from the mixture continuity, momentum, and total energy equations. This decoupling simplifies the implicit system, so that the flow solver can be made significantly more efficient, with very little penalty on overall scheme robustness. Most importantly, the computational cost of the point implicit relaxation is shown to scale linearly with the number of species for the decoupled system, whereas the fully coupled approach scales quadratically. Also, the decoupled method significantly reduces the cost in wall time and memory in comparison to the fully coupled approach.

This decoupled approach for computing design sensitivities with the adjoint system is demonstrated for inviscid flow in chemical non-equilibrium around a re-entry vehicle with a retro-firing annular nozzle. The sensitivities of the surface temperature and mass flow rate through the nozzle plenum are computed with respect plenum conditions and verified against complex-variable finite-difference. The decoupled scheme significantly improved the computational time and memory required to complete the optimization, making this an attractive method for high-fidelity design of hypersonic vehicles.

Aerothermodynamic Design Sensitivities for a Reacting Gas Flow Solver on an Unstructured Mesh Using a Discrete Adjoint Formulation

by
Kyle B. Thompson

A dissertation submitted to the Graduate Faculty of
North Carolina State University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Aerospace Engineering

Raleigh, North Carolina

2017

APPROVED BY:

Hassan Hassan Co-chair of Advisory Committee	Peter Gnoffo Co-chair of Advisory Committee
Jack Edwards	Hong Luo

John Griggs

DEDICATION

To my parents and friends.

BIOGRAPHY

The author was born on December 11th, 1990, in Willow Springs, North Carolina. He recieved a BS in Aerospace Engineering from North Carolina State University in 2012, and subsequently recieved his MS in Aerospace Engineering from North Carolina State University in 2014. After recieving he MS, he began work in the Aerothermodynamics branch of NASA Langley Research Center, via the Pathways program. He completed this disseration on adjoint-based optimization while working at NASA Langley Research Center.

ACKNOWLEDGEMENTS

First, I would like to thank Dr. Peter Gnoffo, who has been an unending source of support during my time NASA Langley Research Center. I would like thank my parents for all of their encouragement over the years, and their commitment to seeing I be given the best opportunity to make the most of myself. I would like to thank my advisor, Dr. Hassan, for instilling in me a diligence to improve myself and for mentoring me through many challenges. I would like to thank the Entry Systems Modeling Project within NASA's Game Changing Development Program for their funding and support of this research. Finally, I would like to recognize the FUN3D team at NASA Langley Research Center, for their support and availability to discuss many challenging problems I have encountered during the course of my time at NASA.

TABLE OF CONTENTS

List of Tables . . . . . **vii**

List of Figures . . . . . **viii**

Chapter 1 Introduction **1**

Chapter 2 Governing Equations **6**
2.1 Reacting Flow Conservation Equations 6
2.2 Thermodynamic Relationships 7
2.3 Chemical Kinetics Model 9

Chapter 3 Numerical Solution of Flow Equations **11**
3.1 Fully-Coupled Point Implicit Method 11
3.2 Decoupled Point Implicit Method 13
3.3 Predicted Cost and Memory Savings of the Decoupled Implicit Problem . 17
3.4 Higher Order Reconstruction 20

Chapter 4 Numerical Solution of Adjoint Equations **22**
4.1 Discrete Adjoint Derivation 23
4.2 Block Jacobi Adjoint Decoupling 24
4.3 Higher-order Reconstruction Linearizations 32
4.4 Memory and Computational Cost of Exact Second Order Linearizations . 34

Chapter 5 Demonstration Problem: Hypersonic Retro-firing Annular Jet 35
5.1 Annular Jet Configuration and Test Conditions 35
5.2 Annular Jet Flow Features and Steadiness Dependence on Cone Angle . . 37
5.3 Mesh Refinement Study 40
5.4 Sensitivity to Frozen Flux Limiter 42

Chapter 6 Design Optimization **47**
6.1 Integrated Quantities of Interest 48
6.2 Composite Cost Function Definition and Components 50
6.3 Design Variables 50
6.4 Obtaining Sensitivity Gradients for Design Variables 51
6.5 First-Order Inverse Design Optimization 52
6.6 Second-Order Direct Design Optimization 54

Chapter 7 Verification of Adjoint Sensitivity Gradients **58**
7.1 Forward-mode Sensitivities Using Complex-Variables 58

7.2 Verification of 2nd-order Adjoint Linearizations 60

Chapter 8 Relative Efficiency of Decoupled Flow Solver 63
8.1 5 km/s Flow over Cylinder 63
8.1.1 Cylinder - Verification of Implementation 64
8.1.2 Cylinder - Memory Cost 66
8.1.3 Cylinder - Computational Cost 66
8.2 15 km/s Flow over Spherically-Capped Cone 68
8.2.1 Sphere-Cone - Verification of Implementation 69
8.2.2 Sphere-Cone - Convergence Quality 69
8.2.3 Sphere Cone - Convergence Improvement with Exact First-Order Linearizations 70
8.3 Re-entry Vehicle with Retro-Firing Annular Jet 72
8.3.1 Annular Jet: Verification of Implementation 73
8.3.2 Annular Jet: Convergence Quality 74
8.3.3 Annular Jet: Relative Speedup 75

Chapter 9 Relative Efficiency of Decoupled Adjoint Solver 78
9.1 Verification of Consistency 78
9.2 Relative Memory Savings and Speedup 80

Chapter 10Concluding Remarks 84

References 87

Appendix 92
Appendix A Derivations 93
A.1 Decoupled Flux Derivation 93
A.2 Quadratic Interpolation Between Thermodynamic Curve Fits 96
A.3 Change of Variable Sets 97
A.4 Change of Equation Sets 104

LIST OF TABLES

Table 5.1 Annular nozzle geometry inputs. 37
Table 5.2 Flow conditions. 37
Table 5.3 Plenum conditions. 40

Table 6.1 Direct design optimization improvement. 56

Table 7.1 Sensitivity derivative comparison - perfect gas. 61
Table 7.2 Sensitivity derivative comparison - H_2-N_2 frozen. 61
Table 7.3 Sensitivity derivative comparison - H_2-N_2 reacting. 62

Table 8.1 Difference with frozen chemistry. 73
Table 8.2 Difference with finite-rate chemistry. 73
Table 8.3 Relative speedup for frozen chemistry. 76
Table 8.4 Relative speedup for finite-rate chemistry. 76

Table 9.1 Angle of attack sensitivity relative difference. 79
Table 9.2 Annular jet plenum sensitivity relative difference. 80
Table 9.3 Relative speedup. 81

LIST OF FIGURES

Figure 3.1 Edge reconstruction. 20

Figure 4.1 Example stencil for least-squares gradient evaluation. 32

Figure 5.1 Annular jet geometry. 36
Figure 5.2 Annular jet temperature contours, blowing pure H_2. 38
Figure 5.3 Annular jet H_2O density contours, blowing pure H_2. 38
Figure 5.4 Annular jet sonic line comparison, blowing pure H_2. 39
Figure 5.5 Uniformly refined meshes. 41
Figure 5.6 Grid convergence. 41
Figure 5.7 Typical residual convergence behavior. 43
Figure 5.8 Convergence of surface temperature with different flux limiters. . 44
Figure 5.9 Van Leer and Van Albada flux limiter impact on surface temperature. 45

Figure 6.1 Cost function component integrated areas. 49
Figure 6.2 Cost function and component history. 52
Figure 6.3 Inverse design history. 53
Figure 6.4 Direct design cost function. 55
Figure 6.5 Direct design history. 56

Figure 8.1 50×50 cylinder grid. 65
Figure 8.2 Cylinder predicted quantities. 65
Figure 8.3 Memory required convergence study. 67
Figure 8.4 Relative speedup for the decoupled scheme vs. fully coupled scheme. 67
Figure 8.5 Relative cost of flow solver components for 11-species. 68
Figure 8.6 Sphere-cone predicted quantities. 70
Figure 8.7 Sphere-cone convergence details. 71
Figure 8.8 Relative speedup using exact linearizations. 72
Figure 8.9 Second-order residual convergence history. 75

Figure 9.1 Simulation time comparison. 81
Figure 9.2 Memory required to store full linearizations. 82

NOMENCLATURE

Roman Symbols

$A,\ A_d,\ A_m$	Jacobian Matrices
A_i	Face area m^2
a	Speed of sound, m/s
$\mathbf{b}$, $\mathbf{R}$	Residual vector
B_i^r	Equilibrium constant curve fit coefficients
c_s	Species s mass fraction
C	Decoupled scheme chemical source term Jacobian
C_j	Cost function component value
$C_{f,r}$	Preexponential factor of reaction r
$C_{p,s}$	Specific heat at constant pressure of species s
$C_{v,s}$	Specific heat at constant volume of species s
D	Decomposed diagonal Jacobian matrix
$dv_1,\ dv_2,\ dv_3$	Eigenvector components
e_s	Internal energy of species s, J/kg
E	Total energy per unit mass, J/kg
$E_{f,r}$	Activation energy of reaction r, J
F'_ρ	Decoupled scheme mixture mass flux, $kg\ m^2/s$
F'_{ρ_s}	Decoupled scheme mass flux of species s, $kg\ m^2/s$
$\mathbf{F},\ \mathbf{F'},\ \hat{\mathbf{F}}$	Flux vectors
$k_{f,r}$	Forward reaction rate coefficient of reaction r
$k_{b,r}$	Backward reaction rate coefficient of reaction r
$K_{c,r}$	Equilibrium constant of reaction r

L	Adjoint equations Lagrangian
M_s	Molecular weight of species s
$\dot{m}_p$	Plenum mass flow rate kg/s
N_s	Number of species
$\mathbf{n}$	Face unit normal vector
$n_{f,r}$	Exponent of temperature preexponential term for forward reaction r
n_x, n_y, n_z	Face unit normal vector components
$\mathbf{N}$	Face normal vector, m^2
N_{nodes}	Number of nodes
N_{nz}	Number of non-zero off-diagonal entries in Jacobian
N_{nb}	Number of neighbors around local node
O	Decomposed off-diagonal Jacobian matrix
p	Pressure, N/m^2
p_j	Cost function component power
q	Primitive variables
R_ρ	Decoupled scheme constraint
$R_{f,r}$	Forward reaction rate of reaction r
$R_{b,r}$	Backward reaction rate of reaction r
R_u	Universal Gas Constant, J/kg^3
$\mathbf{U}, \mathbf{V}, \mathbf{U}', \hat{\mathbf{U}}$	Conservative variable vectors
$\overline{U}$	Normal velocity, m/s^2
$\mathbf{u}$	Velocity vector, m/s
$u,\ v,\ w$	Components of velocity, m/s
V	Cell volume, m^3

$\tilde{w}$	Roe scheme weighting factor
w_j	Composite cost function component weight
$\hat{\mathbf{V}}$	Decoupled mass fractions vector
$\mathbf{W}$	Chemical source term vector
$\mathbf{x}$, $\mathbf{y}$, $\mathbf{z}$	Spatial coordinates

Greek Symbols

$\lambda_1,\ \lambda_2,\ \lambda_3$	Acoustic and convective eigenvalues
$\lambda^-,\ \lambda^+$	Species flux effective eigenvalues
Λ	Diagonal eigenvalue matrix
ϕ	flux limiter result
ϕ_p	plenum fuel-air ratio
ρ	Mixture density, kg/m^3
ρ_s	Species s density, kg/m^3
$\nu''_{s,r}$	Stoichiometric coefficient of product species s in reaction r
$\nu'_{s,r}$	Stoichiometric coefficient of reactant species s in reaction r
ω_r	Chemical source term scaling factor
ω_s	Chemical source term of species s

Subscripts

ns	Number of species
o	Reference value
∞	Freestream condition

Superscripts

*	Cost function component target
$n, n+1$	Time level
$R,\ L$	Right and left state quantities
f	Face

Chapter 1

Introduction

In the last decades, computational fluid dynamics (CFD) codes have matured to the point that it is possible to obtain high-fidelity design sensitivities, which can be coupled with optimization packages to enable design optimization for a variety of inputs[4, 2]. Typically, CFD codes are coupled with a numerical optimization package[16, 25, 14, 17], where the CFD analysis evaluates an aerodynamic "cost function" of interest, based on a set of design variables, and passes this cost function to the optimizer. The optimizer then returns a new set of inputs to the CFD code to improve the previous design. These optimization procedures also require the sensitivity derivatives of the cost function with respect to the design variables.

The most straightforward approach to computing design sensitivities is a finite-difference approach. The finite-difference approach effectively turns a CFD code into "black box", where each design variables is perturbed and used as an input to a CFD code. This requires very little effort to implement, and can be attractive if the number of cost functions is large in comparison to the number of design variables, since the sensitivity of all cost functions to a single design variable can be computed from a

single finite-difference evaluation. The finite-difference approach, however, requires at least one analysis from the CFD code per design variable, making it ill-suited for design optimization problems with a large number of design inputs.

In recent years, an adjoint-based approach to computing sensitivities has been implemented in many compressible CFD codes[29, 32, 35, 42]. This adjoint-based approach requires a single flow and "adjoint" solution to compute the sensitivity of all design variables to a given cost function. In essence, the adjoint-based approach requires an additional solution per cost function, whereas the finite-difference approach requires an additional solution per design variable. Because the number of design variables is, usually, much larger than the number of cost functions in CFD applications, the adjoint-based approach is much more efficient than the finite-difference approach for computing design sensitivities.

Reacting gas CFD codes have lagged in adopting this adjoint-based approach, but interest has increased in recent years[11, 12, 26]. The relatively small number of reacting gas CFD codes that employ adjoint-based sensitivity analysis is likely due to the significant jump in complexity of the linearizations required, especially in regard to the chemical source term and the dissipation term in the Roe flux difference splitting (FDS) scheme[44]. There is also a split in the reacting gas solver community over the methodology used to compute adjoint-based sensitivity derivatives, specifically whether to implement a continuous or discrete version of the adjoint equations. In a continuous formulation of the adjoint equations, the flow solver equations are linearized and then discretized, whereas in discrete formulation the flow solver equations are discretized and then linearized. Copeland et al.[11] have adopted a continuous formulation of the adjoint equations for flows in chemical and thermal non-equilibrium, and have implemented this formulation in the CFD code SU2[38] from Stanford University. The reasoning cited

for choosing a continuous formulation of the adjoint equations, was that, among other things, the adjoint system was not well suited to being solved by the same methods used for solving the flow solver equations and the resulting adjoint solution could lead to non-physical oscillations that fail to capture the continuous adjoint solution. Lockwood et al.[27, 26], on the other hand, have published favorable results for a discrete adjoint formulation for both parameter sensitivity and uncertainty quantification.

For the work presented here, a discrete adjoint formulation is chosen over a continuous one. The reason for this decision is that the merits a of continuous adjoint formulation do not outweigh the testing and practical benefits of a discrete adjoint formulation. Because the discretized flow solver equations are linearized in a discrete adjoint solver, the adjoint solution can be verified using finite-difference derivatives. This is not possible with a continuous formulation, as the solution to the continuous problem will not exactly match the discretization; therefore, from a practical standpoint, the verification of a continuous adjoint solution is much more ambiguous than for a discrete adjoint solution. The claim that non-equilibrium flow solvers are not guaranteed to solve the adjoint system is disputed here, as Nielsen et al.[36] proved that a discrete adjoint scheme can be made dual exact, where the discrete adjoint approach matches the direct differentiation approach exactly for each time step.

An additional obstacle that may prevent adjoint-based sensitivity analysis in reacting gas solvers is the extreme problem size associated with high energy physics. The additional equations required in reacting gas simulations lead to large Jacobians that scale quadratically in size to the number of governing equations. This scaling leads to a significant increase in the memory required to store the flux linearizations and the computational cost of the point solver. As reacting gas CFD solvers are used to solve increasingly more complex problems, this onerous quadratic scaling of computational

cost and Jacobian size will ultimately surpass the current limits of hardware and time constraints on achieving a flow solution[13].

To mitigate this scaling issue, Candler et al.[10] proposed a scheme to for a modified form of the Steger-Warming flux vector splitting scheme[28, 48]. In that work, it was shown that quadratic scaling between the cost of solving the implicit system and additional species mass equations can be reduced from quadratic to linear scaling by decoupling the species mass equations from the mixture mass, momentum, and energy equations and solving the two systems sequentially. This work extends the aforementioned work from the modified form of the Steger-Warming flux vector splitting method to the Roe flux difference splitting (FDS) scheme, and, more importantly, extends it for the first time to the solution of the adjoint formulation. Candler et al.[9] later analyzed stability issues associated with the decoupled formulation of the flow solver in [10]. A new chemical source term scaling procedure is introduced in the present work that mitigates these stability concerns.

As its primary focus, this work demonstrates that a variation of the decoupled point-implicit scheme can be applied to both the flow solver and adjoint solver in the reacting gas path of the FUN3D CFD code[5], and significantly improve the efficiency in both computational cost and memory required. Additionally, exactly linearizing the Roe FDS scheme is shown to significantly improve performance and robustness in the flow solver over previous approximate linearizations used[23].

An axi-symmetric hypersonic re-entry vehicle with an annular jet is chosen as the main demonstration problem in this work. This design was previously examined by Gnoffo et al.[21] for a perfect gas as a means of increasing the time-averaged vehicle drag via unsteady interactions between the annular jet plume and the bow shock. A steady solution was found for that geometry and it was postulated that the annular jet plume

could be used as an active cooling mechanism to augment or replace the passive thermal protection system (TPS) on the vehicle. This work extends the previous work to inviscid flow in chemical non-equilibrium, and applies sensitivities computed by the adjoint solver to minimize the vehicle outer surface temperature and mass flow rate through the annular jet plenum by changing the conditions at the plenum.

Chapter 2

Governing Equations

In this section, the conservative equations governing fluid flow for inviscid, chemically reacting flow are presented. This research is extendable to multi-temperature models to account for higher excitation modes than the translational mode; however, the focus of this research uses a 1-temperature model, so only a single, total energy equation is used. The thermodynamic relations and chemical kinetics models used are also presented in detail here.

2.1 Reacting Flow Conservation Equations

Conservation equations for a fluid mixture that is in chemical non-equilibrium and thermal equilibrium can be written as

$$\textbf{Species Conservation}: \quad \frac{\partial \rho_s}{\partial t} + \frac{\partial \rho_s u}{\partial x} + \frac{\partial \rho_s v}{\partial y} + \frac{\partial \rho_s w}{\partial z} = \omega_s \tag{2.1}$$

Mixture Momentum Conservation :

$$\begin{aligned}
\frac{\partial \rho u}{\partial t} + \frac{\partial \rho u^2}{\partial x} + \frac{\partial \rho uv}{\partial y} + \frac{\partial \rho uw}{\partial z} &= -\frac{\partial p}{\partial x} \\
\frac{\partial \rho v}{\partial t} + \frac{\partial \rho vu}{\partial x} + \frac{\partial \rho v^2}{\partial y} + \frac{\partial \rho vw}{\partial z} &= -\frac{\partial p}{\partial y} \\
\frac{\partial \rho w}{\partial t} + \frac{\partial \rho wu}{\partial x} + \frac{\partial \rho wv}{\partial y} + \frac{\partial \rho w^2}{\partial z} &= -\frac{\partial p}{\partial z}
\end{aligned} \tag{2.2}$$

Total Energy Conservation :

$$\frac{\partial \rho E}{\partial t} + \frac{\partial \left(\rho E + p\right) u}{\partial x} + \frac{\partial \left(\rho E + p\right) v}{\partial y} + \frac{\partial \left(\rho E + p\right) w}{\partial z} = 0 \tag{2.3}$$

2.2 Thermodynamic Relationships

The pressure, p, is defined as the sum of the partial pressures of the species

$$p = \sum_{s=1}^{N_s} p_s \tag{2.4}$$

and the partial pressure of species s, p_s, is defined as

$$p_s = \frac{\rho_s R_u T}{M_s} \tag{2.5}$$

where R_u is the universal gas constant and M_s is the molecular weight of species s. The total energy per unit mass E, is defined as

$$E = \sum_{s=1}^{N_s} c_s e_s + \frac{u^2 + v^2 + w^2}{2} \tag{2.6}$$

where c_s is the mass fraction of species s, defined as

$$c_s = \frac{\rho_s}{\rho} \tag{2.7}$$

and e_s is the specific internal energy of species s, defined as

$$e_s = \int_{T_{ref}}^{T} C_{v,s}\, dT + e_{s,o} \tag{2.8}$$

where T_{ref} is a reference temperature and $e_{s,o}$ is the specific energy of formation for species s. In practice, the specific heat at constant volume for species s, $C_{v,s}$, is not used directly. Instead, the specific heat at constant pressure for species s, $C_{p,s}$, is determined via thermodynamic curve fits and related to $C_{v,s}$ via

$$C_{v,s} = C_{p,s} - \frac{R_u}{M_s} \tag{2.9}$$

The thermodynamic properties curve fit tables developed by McBride, Gordon, and Reno[6] were used to compute $C_{p,s}$, and quadratic blending function of the form

$$C_{p,s} = aT^2 + bT + c \tag{2.10}$$

was used to ensure that $C_{p,s}$ and enthalpy were continuous across temperature ranges. The details of coefficients a, b, and c, in Eq. 2.10 and their derivation are given in Appendix A.2

2.3 Chemical Kinetics Model

The production and destruction of species is governed by the source terms, w_s, defined as

$$w_s = M_s \sum_{r=1}^{N_r} \left(\nu_{s,r}^{''} - \nu_{s,r}^{'} \right) \left(R_{f,r} - R_{b,r} \right) \tag{2.11}$$

where N_r is the number of reactions, $\nu_{s,r}^{'}$ and $\nu_{s,r}^{''}$ are the stoichiometric coefficients for the reactants and products, respectively, and $R_{f,r}$ and $R_{b,r}$ are the forward and backward rates for reaction r, respectively. The forward and backward reaction rates are defined as

$$R_{f,r} = 1000 \left[k_{f,r} \prod_{s=1}^{N_s} \left(0.001 \rho_s / M_s \right) \nu_{s,r}^{'} \right] \tag{2.12}$$

$$R_{b,r} = 1000 \left[k_{b,r} \prod_{s=1}^{N_s} \left(0.001 \rho_s / M_s \right) \nu_{s,r}^{''} \right] \tag{2.13}$$

where $k_{f,r}$ and $k_{b,r}$ are the forward and backward rate coefficients, respectively. It should be noted that all terms on the RHS for Eq.s (2.12-2.13) are in cgs units. The factors 1000 and 0.001 are required to convert from cgs to mks, so that $R_{f,r}$ and $R_{b,r}$ are in mks units. The rate coefficients are expressed in the manner defined by Park[40], but for a one-temperature model; thus, the forward and back rate coefficients are defined as

$$k_{f,r} = C_{f,r} T^{n_{f,r}} exp\left(-E_{f,r} / kT \right) \tag{2.14}$$

$$k_{b,r} = \frac{k_{f,r}}{K_{c,r}} \tag{2.15}$$

where $K_{c,r}$ is the equilibrium constant for reaction r, and k is the Boltzmann constant. The preexponential constants $C_{f,r}$ and $n_{f,r}$, as well as the activation energy, $E_{f,r}$, are

documented by Gnoffo[23]. The equilibrium coefficient is computed according to the curve fit defined by Park[39]

$$K_{c,r} = exp(B_1^r + B_2^r \ln Z + B_3^r Z + B_4^r Z^2 + B_5^r Z^3) \tag{2.16}$$

$$Z = 10000/T \tag{2.17}$$

where the curve fit constants, B_i^r, are also documented by Gnoffo[23]

Chapter 3

Numerical Solution of Flow Equations

In this section, the discretization and method of solution of the governing equations from Chapter 2 is derived. Traditionally, the reacting gas path of FUN3D[5] has employed an implicit, fully coupled scheme in the flow solver. In addition to providing details of this fully coupled scheme, a decoupled scheme based on the Roe FDS scheme[44] is derived to improve the computational efficiency of the flow solver and decrease the relative memory required. Details on the reconstruction scheme used in FUN3D to extend the baseline finite-volume scheme to higher-order accuracy are also provided in this section.

3.1 Fully-Coupled Point Implicit Method

The governing equations presented in Eq.s (2.1-2.3) can be recast in vector form as

$$\frac{\partial \mathbf{U}}{\partial t} + \nabla \cdot \mathbf{F} = \mathbf{W} \tag{3.1}$$

or, in semi-discrete form,

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{1}{V}\sum_{f}(\mathbf{F}\cdot\mathbf{N})^{f} = \mathbf{W} \tag{3.2}$$

summing over all faces, f, in the domain, where V is the cell volume, $\mathbf{W}$ is the chemical source term vector, and $\mathbf{N}$ is the face outward normal vector. The vectors of conserved variables and fluxes are:

$$\mathbf{U} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{ns} \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \quad \mathbf{F} = \begin{pmatrix} \rho_1 \overline{U} \\ \vdots \\ \rho_{ns} \overline{U} \\ \rho u \overline{U} + p n_x \\ \rho v \overline{U} + p n_y \\ \rho w \overline{U} + p n_z \\ (\rho E + p)\overline{U} \end{pmatrix} \tag{3.3}$$

where $\overline{U}$ is the outward pointing normal velocity, E is the total energy of the mixture per unit mass as defined in Eq. 2.6. The flux and species chemical source term at the next time level can be approximated as

$$\begin{aligned} \mathbf{F}^{n+1} &\approx \mathbf{F}^{n} + \tfrac{\partial \mathbf{F}}{\partial \mathbf{U}}\delta\mathbf{U}^{n} \\ \mathbf{W}^{n+1} &\approx \mathbf{W}^{n} + \tfrac{\partial \mathbf{W}}{\partial \mathbf{U}}\delta\mathbf{U}^{n} \end{aligned} \tag{3.4}$$

where $\delta\mathbf{U}^n = \mathbf{U}^{n+1} - \mathbf{U}^n$. Using an implicit time integration, the implicit scheme becomes:

$$\frac{\delta\mathbf{U}^n}{\Delta t} + \frac{1}{V}\sum_f (\frac{\partial\mathbf{F}^f}{\partial\mathbf{U^L}}\delta\mathbf{U}^L + \frac{\partial\mathbf{F}^f}{\partial\mathbf{U^R}}\delta\mathbf{U}^R)^n\mathbf{N}^f - \frac{\partial\mathbf{W}}{\partial\mathbf{U}}\delta\mathbf{U}^n = -\frac{1}{V}\sum_f(\mathbf{F}^f\cdot\mathbf{N}^f)^n + \mathbf{W}^n \quad (3.5)$$

or, put more simply:

$$A\delta\mathbf{U}^n = \mathbf{b} \quad (3.6)$$

where A is the Jacobian matrix of the fully coupled system, and $\mathbf{b}$ is the residual vector. For a point implicit relaxation scheme, the Jacobian matrix can be split into its diagonal and off-diagonal elements, with the latter moved to the RHS:

$$A = O + D \quad (3.7)$$

Each matrix element is a square $(ns+4)\times(ns+4)$ matrix. This system can be solved iteratively by using a multi-color matrix ordering to perform a series of Gauss-Seidel sweeps. The computational work for the Gauss-Seidel scheme is dominated by matrix-vector multiplications of elements of O with $\delta\mathbf{U}$, which are $O((N_s+4)^2)$ operations. In the next section, it is shown that decoupling the system reduces these matrix-vector multiplications to $O(5^2+N_s)$ operations.

3.2 Decoupled Point Implicit Method

If the species mass equations are replaced by a single mixture mass equation, the mixture equations can be separated from the species mass equations and the conserved variables

become

$$\mathbf{U}' = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} \quad \hat{\mathbf{U}} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{ns} \end{pmatrix} \tag{3.8}$$

with corresponding flux vectors

$$\mathbf{F}' = \begin{pmatrix} \rho\overline{U} \\ \rho u\overline{U} + pn_x \\ \rho v\overline{U} + pn_y \\ \rho w\overline{U} + pn_z \\ (\rho E + p)\overline{U} \end{pmatrix} \quad \hat{\mathbf{F}} = \begin{pmatrix} \rho_1\overline{U} \\ \vdots \\ \rho_{ns}\overline{U} \end{pmatrix} \tag{3.9}$$

Solving the flux vectors is performed in two sequential steps. The mixture fluxes, $\mathbf{F}'$, are first solved as

$$\frac{\partial \mathbf{U}'}{\partial t} + \frac{1}{V}\sum_f (\mathbf{F}' \cdot \mathbf{N})^f = 0 \tag{3.10}$$

followed by the species fluxes, $\hat{\mathbf{F}}$, as

$$\frac{\partial \hat{\mathbf{U}}}{\partial t} + \frac{1}{V}\sum_f (\hat{\mathbf{F}} \cdot \mathbf{N})^f = \hat{\mathbf{W}} \tag{3.11}$$

The same point-implicit relaxation that uses multi-color Gauss-Seidel sweeps is used to update the conserved variables in $\mathbf{U}'$, and all associated auxiliary variables, such as temperature, pressure, speed of sound, etc. are updated to be consistent with the new state of $\mathbf{U}'$. This update is done holding the mass fraction state constant, and will always

result in the relaxation of a five-equation system. Decoupling the variable sets in Eq. 3.8 does trade an implicit relationship between the mixture and species equations for an explicit one; thus, this decoupling can have an impact on the stability of the scheme, especially due to the non-linearity of the chemical source term[40].

The solution of the species mass equations takes a different form. Based on the work of Candler et al.[10], the decoupled variables can be rewritten in terms of mass fraction, as follows:

$$\delta\hat{\mathbf{U}}^n = \rho^{n+1}\hat{\mathbf{V}}^{n+1} - \rho^n\hat{\mathbf{V}}^n = \rho^{n+1}\delta\hat{\mathbf{V}}^n + \hat{\mathbf{V}}^n\delta\rho^n \tag{3.12}$$

where $\hat{\mathbf{V}} = (c_1, \ldots, c_{ns})^T$, and $c_s = \rho_s/\rho$ is the mass fraction of species s. While the derivation of the species mass equations is different for the Roe FDS scheme from that of Steger-Warming proposed by Candler et al.[10], the final result takes a similar form:

$$\hat{F}_{\rho_s} = c_s F'_\rho + (c_s^L - \tilde{c}_s)\rho^L\lambda^+ + (c_s^R - \tilde{c}_s)\rho^R\lambda^- \tag{3.13}$$

where F'_ρ is the total mass flux computed previously using all $\mathbf{U}'$ variables, $\tilde{}$ denotes a Roe-averaged quantity, and $\lambda^{+/-}$ are functions of the Roe averaged eigenvalues. Likewise, linearizing the species mass fluxes with respect to the $\hat{\mathbf{V}}$ variables yields

$$\hat{\mathbf{F}}^{n+1} = \hat{\mathbf{F}}^n + \frac{\partial\hat{\mathbf{F}}}{\partial\hat{\mathbf{V}}^L}\delta\hat{\mathbf{V}}^L + \frac{\partial\hat{\mathbf{F}}}{\partial\hat{\mathbf{V}}^R}\delta\hat{\mathbf{V}}^R \tag{3.14}$$

$$\frac{\partial\hat{\mathbf{F}}}{\partial\hat{\mathbf{V}}^L} = \tilde{w}F_\rho + (1-\tilde{w})\rho^L\lambda^+ - \tilde{w}\rho^R\lambda^- \tag{3.15}$$

$$\frac{\partial\hat{\mathbf{F}}}{\partial\hat{\mathbf{V}}^R} = (1-\tilde{w})F_\rho + (\tilde{w}-1)\rho^L\lambda^+ + \tilde{w}\rho^R\lambda^- \tag{3.16}$$

A full derivation of Eq.s (3.13-3.16), along with the definition of $\tilde{w}$, is included in Appendix A.1. The chemical source term is linearized in the same manner as the fully

coupled scheme; however, the updated $\mathbf{U}'$ variables are used to evaluate the Jacobian, and the chain rule is applied to linearize $\hat{\mathbf{W}}$ with respect to the species mass fractions:

$$\hat{\mathbf{W}}^{n+1} = \hat{\mathbf{W}}^{n} + \left.\frac{\partial \hat{\mathbf{W}}}{\partial \mathbf{U}}\right|_{\mathbf{U}'} \frac{\partial \mathbf{U}}{\partial \hat{\mathbf{V}}} \tag{3.17}$$

For simplicity of notation, we define

$$C = \left.\frac{\partial \hat{\mathbf{W}}}{\partial \mathbf{U}}\right|_{\mathbf{U}'} \frac{\partial \mathbf{U}}{\partial \hat{\mathbf{V}}} \tag{3.18}$$

The decoupled system to be solved becomes:

$$\begin{aligned} \rho^{n+1}\frac{\delta \hat{\mathbf{V}}^{n}}{\Delta t} + \frac{1}{V}\sum_{f}\left(\frac{\partial \hat{\mathbf{F}}^{f}}{\partial \hat{\mathbf{V}}^{L}}\cdot \mathbf{N}^{f}\delta \hat{\mathbf{V}}^{L} + \frac{\partial \hat{\mathbf{F}}^{f}}{\partial \hat{\mathbf{V}}^{R}}\cdot \mathbf{N}^{f}\delta \hat{\mathbf{V}}^{R}\right)^{n,n+1} - C^{n,n+1}\delta \hat{\mathbf{V}}^{n} \\ = -\frac{1}{V}\sum_{f}\left(\hat{\mathbf{F}}^{n,n+1}\cdot \mathbf{N}\right)^{f} + \mathbf{W}^{n,n+1} - \frac{1}{V}R_{\rho}\hat{\mathbf{V}} \end{aligned} \tag{3.19}$$

where

$$R_{\rho} = \sum_{f}\sum_{s=1}^{N_s}(\hat{F}_{\rho_s}^{n,n+1}\cdot \mathbf{N}) \tag{3.20}$$

is included to preserve the constraint that the mass fractions sum to unity, i.e., $\sum_{s} c_s = 1$, $\sum_{s} \delta c_s = 0$

Candler et al.[9] later found that the decoupled scheme, using a modified Steger-Warming flux splitting, was less stable than the fully coupled scheme when large reaction rates were present, particularly for exothermic reaction. This same instability was observed in numerical experiments for Eq. 3.20. To mitigate this issue, Candler et al. proposed an alternative decoupling of variable sets that was more stable, particularly if endothermic reaction dominated the flow. Instead of implementing this alternative

decoupled scheme, Eq. 3.20 is modified with a scalar scaling factor, ω_r, on the chemical source term $\mathbf{W}^{n,n+1}$

$$\begin{aligned}\rho^{n+1}\frac{\delta\hat{\mathbf{V}}^n}{\Delta t}+\frac{1}{V}\sum_f\left(\frac{\partial\hat{\mathbf{F}}^f}{\partial\hat{\mathbf{V}}^L}\cdot\mathbf{N}^f\delta\hat{\mathbf{V}}^L+\frac{\partial\hat{\mathbf{F}}^f}{\partial\hat{\mathbf{V}}^R}\cdot\mathbf{N}^f\delta\hat{\mathbf{V}}^R\right)^{n,n+1}-C^{n,n+1}\delta\hat{\mathbf{V}}^n \\ =-\frac{1}{V}\sum_f\left(\hat{\mathbf{F}}^{n,n+1}\cdot\mathbf{N}\right)^f+(\omega_r)(\mathbf{W}^{n,n+1})-R_\rho\end{aligned} \tag{3.21}$$

Ramping this scaling factor from zero to one over the course of timestepping the solution to steady-state preserves robustness of the decoupled scheme, and numerical experiments show that it has minimal impact on the convergence rate. Typically, the large reaction rates are only present during the transients early on in the simulation, and damping the source term during this phase does not affect the final result, provided the ramping is completed before the end of the simulation.

3.3 Predicted Cost and Memory Savings of the Decoupled Implicit Problem

In decoupling the species equations, the most significant savings comes from the source term linearization being purely node-based[23]. Solving the mean flow equations is conducted in the same manner as the fully coupled system. All entries in the Jacobian A_m are linearizations of the mixture equation fluxes, which results in 5×5 matrices. All entries in the Jacobian A_d are linearizations of the species mass fluxes, which results in $ns \times ns$ matrices. Because there is no interdependence of species, except through the chemical source term, all contributions due to linearizing the convective flux are purely diagonal $N_s \times N_s$ matrices. Via Eq. 3.7, we decompose A_d into its diagonal and

off-diagonal elements, resulting in the following linear system:

$$\begin{pmatrix} \square & & & \\ & \ddots & & \\ & & \square & \\ & & & \ddots \\ & & & & \square \end{pmatrix} \begin{pmatrix} \delta\hat{\mathbf{V}}_1 \\ \vdots \\ \delta\hat{\mathbf{V}}_i \\ \vdots \\ \delta\hat{\mathbf{V}}_{nodes} \end{pmatrix} = \begin{pmatrix} \hat{b}_1 \\ \vdots \\ \hat{b}_i \\ \vdots \\ \hat{b}_{nodes} \end{pmatrix} - \begin{pmatrix} (\sum_{j=1}^{N_{nb}} [\diagdown]\delta\hat{\mathbf{V}}_j)_1 \\ \vdots \\ (\sum_{j=1}^{N_{nb}} [\diagdown]\delta\hat{\mathbf{V}}_j)_i \\ \vdots \\ (\sum_{j=1}^{N_{nb}} [\diagdown]\delta\hat{\mathbf{V}}_j)_{nodes} \end{pmatrix} \tag{3.22}$$

where $\square$ represents a dense $N_s \times N_s$ matrix, $[\diagdown]$ represents a diagonal matrix, and $\delta\hat{\mathbf{V}}_j$ is the decoupled variable update on the node j that neighbors node i, where N_{nb} is the number of nodes neighboring node i. Thus, the non-zero entries in the off-diagonal matrix can be reduced from diagonal matrices to vectors. This results in significant savings in both computational cost and memory, as the only expensive operation left in solving the implicit system is dealing with the diagonal entries in the Jacobian. An LU decomposition of an $N \times N$ matrix requires $\sim N^3/3$ operations, whereas the the matrix vector products of equivalent dimension requires N^2 operations and the vector inner products cost. Including the LU decomposition operations, the relative cost of the linear solves in the decoupled scheme compared to the fully coupled scheme is approximately

$$\text{Relative Computational Cost} = \frac{\frac{(N_s+4)^3}{3} N_{nodes} + (N_s+4)^2 (S_{GS})(N_{nz})}{\frac{(N_s)^3+(5)^3}{3} N_{nodes} + (N_s+5^2)(S_{GS})(N_{nz})} \tag{3.23}$$

where S_{GS} is the number of multi-color Gauss-Seidel sweeps, N_{nodes} is the number of nodes, and N_{nz} is the number of non-zero off-diagonal entries stored using compressed row storage[15]. Because $(S_{GS)(N_nz} >> N_{nodes}$, the LU decomposition cost is negligible except when the number of species is extremely large. $N_{nb}(N_s)$ operations. If the LU decomposition costs are dropped from Eq. 3.23, the relative cost of the linear solve

becomes

$$\text{Relative Computational Cost} = \frac{(N_s + 4)^2}{(N_s + 5^2)} \tag{3.24}$$

and can therefore expect nearly linear speedup in the linear solver cost with the number of species when using the decoupled system over the fully coupled system. It should be noted that cost predicted by operation count of an LU decomposition does not usually manifest equivalently in computational cost. Significant compiler optimizations are possible if a storage scheme is adopted where memory collocation of the diagonal block jacobians is maintained. These optimizations decrease the cost of the LU decomposition relative to the Gauss-Seidel sweeps further, and thus the LU decomposition has never been seen as a dominant cost for hypersonic simulations.

Using compressed row storage, the relative memory savings in the limit of a large number of species for the Jacobian is given by

$$\begin{aligned} \textit{Relative Memory Cost} &= \frac{size(A_d)}{size(A)} \\ &= \lim_{N_s \to \infty} \frac{(N_s^2 + 5^2)(N_{nodes}) + (N_s + 5^2)(N_{nz})}{(N_s + 4)^2(N_{nodes} + N_{nz})} \\ &= \frac{N_{nodes}}{N_{nodes} + N_{nz}} \end{aligned} \tag{3.25}$$

For a three-dimensional structured grid, each node has six neighbors, i.e., $N_{nz} = 6N_{nodes}$; therefore, we can expect the Jacobian memory required to decrease by a factor of seven using this decoupled scheme. Interestingly, for a grid that is not purely hexahedra, $N_{nz} > 6N_{nodes}$; thus, this decoupled scheme provides higher relative memory savings on unstructured grids than structured grids when using compressed row storage.

3.4 Higher Order Reconstruction

To achieve higher-order accuracy, the FUN3D solver uses an extension of the unstructured MUSCL (U-MUSCL) reconstruction scheme developed by Burg et al[7, 8], which is itself an extension of the Monotonic Upstream-centered Scheme for Conservation Laws (MUSCL) scheme developed by Van Leer[50]. The implementation is a combination of central differencing and the original U-MUSCL scheme

$$\begin{aligned} q_L &= q_1 + (1-\kappa)\left[\phi\left(\frac{\partial q_1}{\partial x}dx + \frac{\partial q_1}{\partial y}dy + \frac{\partial q_1}{\partial z}dz\right)\right] + \frac{\kappa}{2}(q_2 - q_1) \\ q_R &= q_2 + (1-\kappa)\left[\phi\left(\frac{\partial q_2}{\partial x}dx + \frac{\partial q_2}{\partial y}dy + \frac{\partial q_2}{\partial z}dz\right)\right] + \frac{\kappa}{2}(q_1 - q_2) \end{aligned} \tag{3.26}$$

where $\kappa = 0.5$ is usually used on unstructured grids with mixed elements. Figure 3.1 shows that $q_{L,R}$ are the primitive variables at the left and right sides of the edge midpoint, where the flux is evaluated, $q_{1,2}$ are the primitive variables at nodes 1 and 2. The

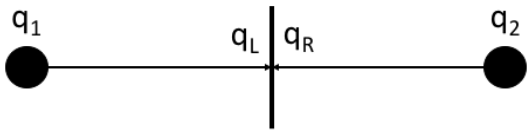


Figure 3.1: Edge reconstruction.

variable ϕ is the result of the scalar flux limiter function, that is required to preserve monotonicity in the second order reconstruction near discontinuities. FUN3D supports a larger variety of flux limiters that fall into two main categories: edge-based limiters, and stencil-based limiters. The edge based limiters are evaluated for two nodes at each

edge. Different values of ϕ can exist at each node for edge-based limiter, and the results are not "freezable" at each node in FUN3D. Stencil-based limiters are evaluated at each node, based on the gradients that are computed there. For these limiters, the value of ϕ is unique to each node and can be stored with the flow solution as an additional variable. The frozen state of the limiter is only re-evaluated if the reconstruction forces a non-physical state at the dual volume interface.

For this study, smooth Van Albada[49], Van Leer[51], and Minmod[45] flux limiter functions are used. Each of these limiters is augmented with a heuristic pressure limiter by Park[41]. The choice of these limiters impacts solution convergence and accuracy. It is important to note that the smooth Van Albada averaging function is given as

$$\phi\left(a,b\right)=\frac{(b^2+\varepsilon^2)a+(a^2+\varepsilon^2)b}{a^2+b^2+2\varepsilon^2} \tag{3.27}$$

where a and b are the left and right node gradients, and ε is the "smoothing coefficient". This smoothing coefficient is used to tune the flux limiter to a variety of the problems, and it is advised to be the reciprocal of the mean aerodynamic chord (MAC) in grid units for FUN3D[5]. The choice of ε is critical to some hypersonic applications involving chemistry, and will be discussed later.

Chapter 4

Numerical Solution of Adjoint Equations

This section details the derivation of the adjoint equations to be solved in conjunction with the primal flow equations. The primary goals of this research are to:

1. Compute sensitivities of aerodynamic and aerothermodynamic quantities to design variables

2. Utilize a decoupled approach in the solution of the adjoint

To achieve these goals, the sensitivity to the primal flow equation formulation must first be solved. For a discrete adjoint formulation, this requires a solution of costate variables that relate a change in the flow equation residuals to a change in the function of interest. The adjoint solver also suffers from the quadratic scaling in computational cost and memory required, similar to the primal flow solver. To mitigate this scaling, a decoupled scheme is derived that is consistent with the decoupled flow solver.

4.1 Discrete Adjoint Derivation

The derivation for the discrete adjoint begins with forming the Lagrangian as

$$L(\mathbf{D}, \mathbf{Q}, \mathbf{X}, \mathbf{\Lambda}) = f(\mathbf{D}, \mathbf{Q}, \mathbf{X}) + \mathbf{\Lambda}^T \mathbf{R}(\mathbf{D}, \mathbf{Q}, \mathbf{X}) \tag{4.1}$$

where $\mathbf{R}$ is the residual of the flow equations, $\mathbf{D}$ is the vector of design variables, $\mathbf{Q}$ is the vector of conserved variables, and $\mathbf{X}$ is the mesh coordinates. Differentiating with respect to the design variables, $\mathbf{D}$, yields

$$\frac{\partial L}{\partial \mathbf{D}} = \left\{ \frac{\partial f}{\partial \mathbf{D}} + \left[\frac{\partial \mathbf{X}}{\partial \mathbf{D}}\right]^T \frac{\partial f}{\partial \mathbf{X}} \right\} + \left[\frac{\partial \mathbf{Q}}{\partial \mathbf{D}}\right]^T \left\{ \frac{\partial f}{\partial \mathbf{Q}} + \left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^T \mathbf{\Lambda} \right\} + \left\{ \left[\frac{\partial \mathbf{R}}{\partial \mathbf{D}}\right]^T + \left[\frac{\partial \mathbf{X}}{\partial \mathbf{D}}\right]^T \left[\frac{\partial \mathbf{R}}{\partial \mathbf{X}}\right]^T \right\} \mathbf{\Lambda} \tag{4.2}$$

To eliminate the dependence of conserved variables, $\mathbf{Q}$, on the design variables, we solve the adjoint equation

$$\left[\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}\right]^T \mathbf{\Lambda} = -\frac{\partial f}{\partial \mathbf{Q}} \tag{4.3}$$

Where the Lagrange multipliers (hereafter referred to as costate variables), Λ, are the cost function dependence on the residual

$$\mathbf{\Lambda} = -\frac{\partial f}{\partial \mathbf{R}} \tag{4.4}$$

Eq. 4.2 can ultimately be used in error estimation and sensitivity analysis for design optimization. With the second term in Eq. 4.2 eliminated, the derivative of the Lagrangian becomes

$$\frac{\partial L}{\partial \mathbf{D}} = \left\{ \frac{\partial f}{\partial \mathbf{D}} + \left[\frac{\partial \mathbf{X}}{\partial \mathbf{D}}\right]^T \frac{\partial f}{\partial \mathbf{X}} \right\} + \left\{ \left[\frac{\partial \mathbf{R}}{\partial \mathbf{D}}\right]^T + \left[\frac{\partial \mathbf{X}}{\partial \mathbf{D}}\right]^T \left[\frac{\partial \mathbf{R}}{\partial \mathbf{X}}\right]^T \right\} \mathbf{\Lambda} \tag{4.5}$$

By solving the adjoint equation (Eq. 4.3) to obtain the costate variable vector, $\mathbf{\Lambda}$, we can now use a non-linear optimizer to determine the optimum set of design variables, $\mathbf{D}^*$. This optimization can be done using **SNOPT[18]**, **KSOPT[25]**, or **NPSOL[17]** in FUN3D, as well as a host of other non-linear optimizers.

4.2 Block Jacobi Adjoint Decoupling

It is possible to decouple the adjoint equations in a fashion similar to that done to the primal flow equations. In this decoupled adjoint formulation, the conserved variables are split identically to the flow equations, with the fully-coupled vector of conserved variables

$$\mathbf{U} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{ns} \\ \rho\mathbf{u} \\ \rho E \end{pmatrix} \tag{4.6}$$

split into

$$\mathbf{U}' = \begin{pmatrix} \rho \\ \rho\mathbf{u} \\ \rho E \end{pmatrix}, \qquad \hat{\mathbf{V}} = \begin{pmatrix} c_1 \\ \vdots \\ c_{ns} \end{pmatrix} \tag{4.7}$$

With this splitting, the mixture equations for single point in the global system of the decoupled flow solve can be written as

$$\left[\frac{V}{\Delta t}\mathbf{I}+\begin{pmatrix}\frac{\partial \mathbf{R}_{\rho}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho E}\\ \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E}\\ \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E}\end{pmatrix}\right]\begin{pmatrix}\Delta \rho\\ \Delta \rho \mathbf{u}\\ \Delta \rho E\end{pmatrix}=\begin{pmatrix}\mathbf{R}_{\rho}\\ \mathbf{R}_{\rho \mathbf{u}}\\ \mathbf{R}_{\rho E}\end{pmatrix} \tag{4.8}$$

Likewise, the species mass equations for a single point can be written as

$$\left[\frac{\rho V}{\Delta t}\mathbf{I}+\begin{pmatrix}\frac{\partial \mathbf{R}_{\rho_1}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_1}}{\partial c_{ns}}\\ \vdots & \ddots & \vdots\\ \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial c_{ns}}\end{pmatrix}\right]\begin{pmatrix}\Delta c_1\\ \vdots\\ \Delta c_{ns}\end{pmatrix}=\begin{pmatrix}\mathbf{R}'_{\rho_1}\\ \vdots\\ \mathbf{R}'_{\rho_{ns}}\end{pmatrix} \tag{4.9}$$

$$\mathbf{R}'_{\rho_s}=\mathbf{R}_{\rho_s}-c_s\mathbf{R}_{\rho} \tag{4.10}$$

$$\mathbf{R}_{\rho}=\sum_{i=1}^{N_s}\left(\mathbf{R}_{\rho_i}\right) \tag{4.11}$$

Eq.s (4.8-4.9) omitted dependencies between equation sets, and it was been found[10] that omitting these dependencies does not hinder convergence the primal flow solver. Because the adjoint requires an exact linearization of the converged steady-state solution, however, these cross-system dependencies must be accounted for in the decoupled adjoint formulation.

The next step is to reconcile the split conserved variables, $\mathbf{U}'$ and $\hat{\mathbf{V}}$, with the conserved variable vector $\mathbf{Q}$ in the discrete adjoint formulation given in Eq. 4.3. This begins by recognizing that the decoupled scheme can actually also be solved as a fully-coupled system of equations that involves the change of variables

$$\mathbf{U} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{ns} \\ \rho\mathbf{u} \\ \rho E \end{pmatrix} \rightarrow \mathbf{V} = \begin{pmatrix} c_1 \\ \vdots \\ c_{ns} \\ \rho \\ \rho\mathbf{u} \\ \rho E \end{pmatrix} \tag{4.12}$$

as well as the change of equations

$$\mathbf{R}_{\mathbf{U}} = \begin{pmatrix} \mathbf{R}_{\rho_1} \\ \vdots \\ \mathbf{R}_{\rho_{N_s}} \\ \mathbf{R}_{\rho\mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \rightarrow \mathbf{R}_{\mathbf{V}} = \begin{pmatrix} \mathbf{R}_{\rho_1} - c_1 \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \vdots \\ \mathbf{R}_{\rho_{N_s}} - c_{N_s} \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \mathbf{R}_{\rho\mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \tag{4.13}$$

The derivation of the transformation matrices need to convert between these two systems are included in Section A.3. From this perspective, the decoupled scheme linear system

for the flow solver can be viewed, with no approximations, as

$$\left[\frac{V}{\Delta t}\mathbf{I}+\begin{pmatrix} \frac{\partial \mathbf{R}'_{\rho_1}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}'_{\rho_1}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}'_{\rho_1}}{\partial \rho} & \frac{\partial \mathbf{R}'_{\rho_1}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}'_{\rho_1}}{\partial \rho E} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}'_{\rho_{N_s}}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}'_{\rho_{N_s}}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}'_{\rho_{N_s}}}{\partial \rho} & \frac{\partial \mathbf{R}'_{\rho_{N_s}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}'_{\rho_{N_s}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho E}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho E}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix}\right]\begin{pmatrix} \Delta c_1 \\ \vdots \\ \Delta c_{N_s} \\ \Delta \rho \\ \Delta \rho \mathbf{u} \\ \Delta \rho E \end{pmatrix} = \begin{pmatrix} \mathbf{R}'_{\rho_1} \\ \vdots \\ \mathbf{R}'_{\rho_{N_s}} \\ \mathbf{R}_{\rho} \\ \mathbf{R}_{\rho \mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \tag{4.14}$$

In addition to Eq.s (3.15-3.16), the iterative mechanism used in the flow solver makes the following approximations to the Jacobians

$$\begin{aligned} \frac{\partial \mathbf{R}_{\rho}}{\partial c_s} = \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial c_s} = \frac{\partial \mathbf{R}_{\rho E}}{\partial c_s} &= 0 \\ \frac{\partial \mathbf{R}'_{\rho_s}}{\partial \rho} = \frac{\partial \mathbf{R}'_{\rho_s}}{\partial \rho \mathbf{u}} = \frac{\partial \mathbf{R}'_{\rho_s}}{\partial \rho E} &= 0 \end{aligned} \tag{4.15}$$

which results in a significantly sparser matrix

$$\left[\frac{V}{\Delta t}\mathbf{I}+\left(\begin{array}{ccc|ccc} \frac{\partial \mathbf{R}_{\rho_1}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_1}}{\partial c_{N_s}} & & & \\ \vdots & \ddots & \vdots & & 0 & \\ \frac{\partial \mathbf{R}_{\rho_{N_s}}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_{N_s}}}{\partial c_{N_s}} & & & \\ \hline & & & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho E} \\ & 0 & & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} \\ & & & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{array}\right)\right]\begin{pmatrix} \Delta c_1 \\ \vdots \\ \Delta c_{N_s} \\ \Delta \rho \\ \Delta \rho \mathbf{u} \\ \Delta \rho E \end{pmatrix} = \begin{pmatrix} \mathbf{R}'_{\rho_1} \\ \vdots \\ \mathbf{R}'_{\rho_{N_s}} \\ \mathbf{R}_{\rho} \\ \mathbf{R}_{\rho \mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \tag{4.16}$$

that results in the approximate factorization used by the flow solver. This is a more complete picture of the decoupled scheme, but it indicates that Eq. 4.14 must be used to formulate the adjoint system of equations, since Eq. 4.16 omits a significant part of the Jacobian.

The simplest approach to solving Eq. 4.3 with system of equations $\mathbf{R_V}$ and dependent variables $\mathbf{V}$ is to solve

$$\frac{\partial \mathbf{R_V}}{\partial \mathbf{V}}^T \Lambda_{\mathbf{V}} = -\frac{\partial f}{\partial \mathbf{V}} \tag{4.17}$$

directly, via a linear solver such as GMRES [46]. Nielsen [34] found that time-marching the FUN3D discrete adjoint linear system with the same point-implicit scheme as the FUN3D flow solver was more robust, and solved systems where GMRES tended to stall. The time marching algorithm also has the benefit of reusing the approximate Jacobians

from the flow solver, transforming Eq. 4.17 into

$$\left(\frac{V}{\Delta t}\mathbf{I}+\frac{\partial\tilde{\mathbf{R}}_{\mathbf{V}}}{\partial\mathbf{V}}\right)^{T}\Delta\Lambda_{\mathbf{V}}=-\left(\frac{\partial\mathbf{R}_{\mathbf{V}}}{\partial\mathbf{V}}^{T}\Lambda_{\mathbf{V}}+\frac{\partial f}{\partial\mathbf{V}}\right) \tag{4.18}$$

where $\frac{\partial\tilde{\mathbf{R}}_{\mathbf{V}}}{\partial\mathbf{V}}$ can include all of the Jacobian approximations used by the flow solver Jacobians, including those in Eq. 4.15. Eq. 4.18 can be solved using the same iterative mechanism as the flow solver; however, the exact linearizations of $\frac{\partial\mathbf{R}_{\mathbf{V}}}{\partial\mathbf{V}}$ are different and more complex than the linearizations required by the traditional fully-coupled system, $\frac{\partial\mathbf{R}_{\mathbf{U}}}{\partial\mathbf{U}}$. This makes the decoupled scheme somewhat less attractive, since the re-implementing the exact linearizations for the change of variable and change of equations in the decoupled scheme requires a significant amount of coding and verification. An alternative to Eq. 4.18 is to recast the decoupled scheme again as a series of matrix transformations on the fully-coupled system of equations. For the flow solver this involves

$$\begin{aligned}\left(\frac{V}{\Delta t}\mathbf{I}+\frac{\partial\mathbf{R}_{\mathbf{U}}}{\partial\mathbf{U}}\right)\Delta\mathbf{U}&=\mathbf{R}_{\mathbf{U}}\\ \frac{\partial\mathbf{R}_{\mathbf{V}}}{\partial\mathbf{R}_{\mathbf{U}}}\left(\frac{V}{\Delta t}\mathbf{I}+\frac{\partial\mathbf{R}_{\mathbf{U}}}{\partial\mathbf{U}}\right)\left(\frac{\partial\mathbf{U}}{\partial\mathbf{V}}\right)\Delta\mathbf{V}&=\frac{\partial\mathbf{R}_{\mathbf{V}}}{\partial\mathbf{R}_{\mathbf{U}}}\mathbf{R}_{\mathbf{U}}\end{aligned} \tag{4.19}$$

From the perspective of Eq. 4.19, the decoupled flow solver scheme is actually the result of left and right preconditioning on the fully-coupled scheme that can be generically written as

$$\begin{aligned}\mathbf{M}\left(\frac{V}{\Delta t}\mathbf{I}+\mathbf{A}_{\mathbf{U}}\right)(\mathbf{B}\Delta\mathbf{V})&=\mathbf{M}\mathbf{R}_{\mathbf{U}}\\ \Delta\mathbf{U}&=\mathbf{B}\Delta\mathbf{V}\end{aligned} \tag{4.20}$$

where $\mathbf{M}$ is the left preconditioner, $\frac{\partial\mathbf{R}_{\mathbf{V}}}{\partial\mathbf{R}_{\mathbf{U}}}$, $\mathbf{B}$ is the right preconditioner, $\frac{\partial\mathbf{U}}{\partial\mathbf{V}}$, and $\mathbf{A}_{\mathbf{U}}$ is the Jacobian matrix for the fully coupled system, $\frac{\partial\mathbf{R}_{\mathbf{U}}}{\partial\mathbf{U}}$. A full derivation of the trans-

formation of variables sets, $\frac{\partial \mathbf{U}}{\partial \mathbf{V}}$ is included in Appendix A.3. Eq. 4.20 is crucial towards the understanding of the adjoint system of equations, since the transpose operation will reverse the order of operations of these matrix products. Based on Eq. 4.20, the Jacobian for the system based on $\mathbf{R_V}$ and $\mathbf{V}$, denoted as $\mathbf{A_V}$, can be written as

$$\mathbf{A_V} = \mathbf{M}\mathbf{A_U}\mathbf{B} \tag{4.21}$$

along with its transpose

$$\mathbf{A_V}^T = (\mathbf{M}\mathbf{A_U}\mathbf{B})^T = \mathbf{B}^T\mathbf{A_U}^T\mathbf{M}^T \tag{4.22}$$

Thus, in the adjoint linear system of equations, $\mathbf{B}$ becomes the left preconditioner, and $\mathbf{M}$ becomes the right preconditioner. Applying the matrix operations in Eq. 4.22, Eq. 4.17 can be re-written as

$$\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)^T \left(\frac{\partial \mathbf{R_U}}{\partial \mathbf{U}}\right)^T \left(\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}}\right)^T \Lambda_\mathbf{V} = -\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)^T \left(\frac{\partial f}{\partial \mathbf{U}}\right) \tag{4.23}$$

and applying the same time-marching strategy as Eq. 4.18 leads to

$$\left(\frac{V}{\Delta t}\mathbf{I} + \frac{\partial \tilde{\mathbf{R}}_\mathbf{V}}{\partial \mathbf{V}}\right) \Delta\Lambda_\mathbf{V} = -\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)^T \left[\left(\frac{\partial \mathbf{R_U}}{\partial \mathbf{U}}\right)^T \left(\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}}\right)^T \Lambda_\mathbf{V} + \left(\frac{\partial f}{\partial \mathbf{U}}\right)\right] \tag{4.24}$$

Where the adjoint costate variables associated with the fully coupled equations, $\mathbf{R_U}$ can be recovered from the right preconditioning

$$\Delta\Lambda_\mathbf{U} = \left(\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}}\right)^T \Delta\Lambda_\mathbf{V} \tag{4.25}$$

Based on Eq.s (4.24-4.25), it is possible to reuse the exact Jacobian of the fully coupled scheme, $\mathbf{A_U}$, instead of computing the exact Jacobian of the decoupled system, $\mathbf{A_V}$. This reused is very attractive, since the implementation of the fully coupled scheme does not need to be changed at the low-level linearizations. Instead, the residual of the adjoint can be formed in the exact same fashion as the fully coupled scheme, and a series of matrix operations can then be performed to transform the equations and dependent variables into those used by the decoupled scheme. A full derivation of the change of equation sets, $\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}}$, is included in Appendix A.4

The iterative mechanism required by the adjoint solver has been found to be much more flexible than the flow solver. In the flow solver, the mixture variables were updated before the species mass fractions, with the former used to compute the later at the next time level. This same process can be carried out on the costate variable vector, $\Lambda_\mathbf{V}$, in the adjoint solver, which would correspond to a block Gauss-Seidel type scheme; however, through testing, their is no significant advantage found applying costate variables at the next time level to compute costate variables at the previous time level. A much more cost effective approach is to apply a block jacobi scheme, where all costate variables are updated at the same time level, regardless of order. Because information is not passed between the costate variables for the mixture equations and costate variables for the species equations, only a single residual vector is required to be computed per timestep in the block jacobi adjoint solver, rather than two required by a block gauss-seidel type adjoint solver. This algorithm is more efficient than the flow solver iterative mechanism, which requires two residual vectors to be computed per timestep.

4.3 Higher-order Reconstruction Linearizations

The reconstruction scheme detailed in Section 3.4 must be exactly linearized in the adjoint solver to achieve correct sensitivity derivatives. The adjoint uses a frozen flux limiter, for purposes that will be discussed in a later section, and is therefore held as constant in the linearization computations. All gradient information in Eq. 3.26, $\frac{\partial q_{1,2}}{\partial x}$, $\frac{\partial q_{1,2}}{\partial y}$, and $\frac{\partial q_{1,2}}{\partial z}$, are computed using least-squares. For the example stencil shown in Figure 4.1 this is effectively computed as

$$\begin{aligned} \frac{\partial q}{\partial x} &= \sum_{i=1}^{5} W_{x,i} \left(q_i - q_0 \right) \\ \frac{\partial q}{\partial y} &= \sum_{i=1}^{5} W_{y,i} \left(q_i - q_0 \right) \\ \frac{\partial q}{\partial z} &= \sum_{i=1}^{5} W_{z,i} \left(q_i - q_0 \right) \end{aligned} \tag{4.26}$$

where the z direction would come from geometry out of the page. In practice, the higher

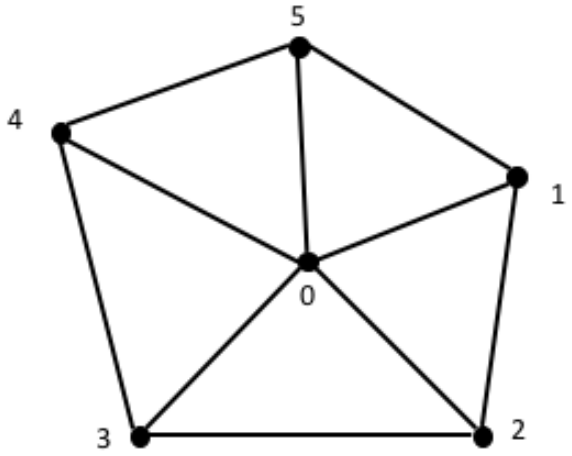


Figure 4.1: Example stencil for least-squares gradient evaluation.

order linearizations can be managed easily by constructing a list of neighboring nodes for each node. By the chain rule the linearization of the residual, $\mathbf{R}$, is then evaluated in two parts

$$\frac{\partial \mathbf{R}\left(\mathbf{Q}^{*}\left(\mathbf{Q}\right)\right)}{\partial \mathbf{Q}} = \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}^{*}} \frac{\partial \mathbf{Q}^{*}}{\partial \mathbf{Q}} \tag{4.27}$$

where $\mathbf{Q}$ are the conserved variables at each node, and $\mathbf{Q}^{*}$ are the higher order terms computed in the U-MUSCL reconstruction; therefore, after computing the exact first-order Jacobian of the Roe FDS scheme, $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}$, the higher order linearizations are computed by making a circuit around each node to pick up the contributions from the least squares gradient computation. For the example stencil in Figure 4.1, the contribution to the residual from looping around node 0 would be

$$\frac{\partial \mathbf{R}_0}{\partial \mathbf{Q}^{*}} \frac{\partial \mathbf{Q}^{*}}{\partial \mathbf{Q}_0} = \frac{\partial \mathbf{R}_0}{\partial \mathbf{Q}^{*}} \left[\sum_{i=1}^{5} \left(1-\kappa\right) \left(-W_{x,i}dx - W_{y,i}dy - W_{z,i}dz\right) \frac{\partial q_0}{\partial \mathbf{Q}_0} \right] \tag{4.28}$$

An important point that this illustrates is that the form of the higher order linearizations does not change significantly based on the primitive variable vector q, only affecting the primitive to conserved variable Jacobian, $\frac{\partial q_0}{\partial \mathbf{Q}_0}$, which is easily computed. This is a powerful telescoping property of the reconstruction scheme from Section 3.4, and it significantly decreases the complexity of the code required to implement the second order term linearizations. The downside of looping around each node is the high number of cache misses, as the data structures in FUN3D are not conducive to co-locating memory based on stencil. This results in the evaluation of the second order linearizations being the dominant computational cost in the adjoint solver.

4.4 Memory and Computational Cost of Exact Second Order Linearizations

As shown in Section 4.3, the second order reconstruction results in an extended stencil. This significantly decreases the sparsity of the Jacobian, since the linearizations at node must now include the nearest neighbors. The cost of computing these linearizations can quickly dominate other costs of the adjoint, but can be significantly mitigated if the linearizations of second order Jacobian are stored. This essentially trades all the memory in the simulation for computational speed, and may not be possible for large mesh sizes. Since all of the linearizations in the second order Jacobian must be exact, the memory saving approximations of the decoupled scheme can only be applied to the LHS Jacobians in Eq. 4.24. This last point makes the quadratic scaling of memory required with the number of species a significant concern of the adjoint. The savings provided by the decoupled scheme become a very significant benefit in cases involving many species, since the memory freed by the sparsity on the LHS of Eq. 4.24 can be used by the RHS linearizations to increase in computational efficiency.

Chapter 5

Demonstration Problem: Hypersonic Retro-firing Annular Jet

To demonstrate the advantages and the robustness of the decoupled flow and adjoint solvers relative to the fully coupled flow and adjoint solvers, a demonstration problem is chosen with sufficiently challenging physics. The problem must include a sufficient sensitivity to the chemistry model to highlight the stability concerns of the decoupled flow solver presented in Section 8.2. In this case, the sensitivity manifests through the formation of a large, recirculating buffer gas zone that covers most of the vehicle outer surface. This chapter details the geometry and test conditions of the demonstration problem, as well as the characteristics of the flow solution.

5.1 Annular Jet Configuration and Test Conditions

The geometry chosen is a hypersonic re-entry vehicle with a retro-firing annular nozzle, as shown in Figure 5.1. This geometry was originally investigated by Gnoffo et al[21]

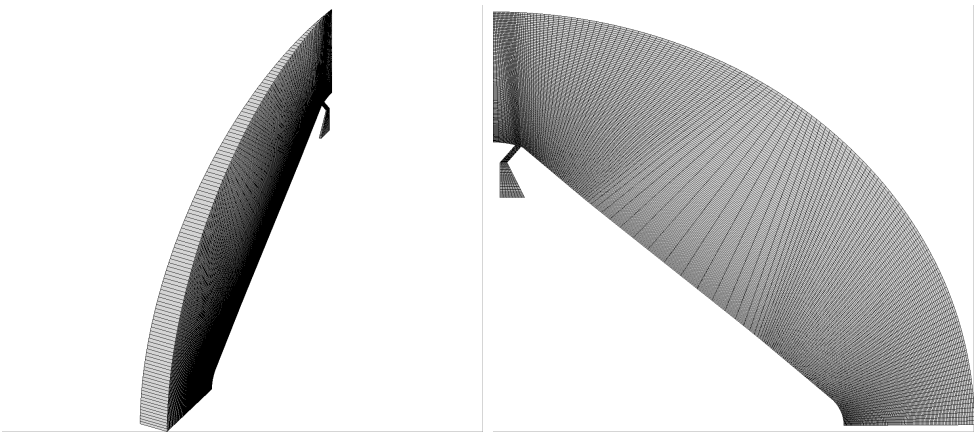


Figure 5.1: Annular jet geometry.

to obtain increased drag from "pulsing" the annular jet to obtain a beneficial effect from the unsteady shock interaction with the plume of the jet. It was also determined from the aforementioned work that a steady solution to the Euler equations exists for this annular jet configuration. This problem is attractive for optimization, because the annular jet plenum conditions significantly affect aerodynamic and aerothermodynamic quantities of interest on the vehicle surface. These effects are very non-linear, due to the shock and jet plume interaction, and it is therefore very difficult to intuitively understand the relationship between the plenum conditions and vehicle surface quantities. Because the adjoint solver is rigorously derived from the discretized governing equations, the relationship between the plenum and surface can be directly determined, making this an excellent showcase problem for adjoint-based sensitivity information.

The geometry was generated with the parameters shown in Table 5.1, with the mesh originally created as structured grid and then converted to an unstructured grid comprised of hexahedra and prismatic elements. The flow conditions for a Mach 20 condition are shown in Table 5.2, with all simulations conducted using a 9-species Hydrogen-air mixture comprised of H_2, N_2, O_2, H, N, O, NO, OH, and H_2O.

Parameter	Description	Value
r_{throat}	nozzle throat radius, m	0.02
$r_{plenum,inner}$	inside nozzle radius at plenum face, m	0.02
$r_{plenum,outer}$	outside nozzle radius at plenum face, m	0.07
$r_{exit,inner}$	inside nozzle radius at exit, m	0.064
$r_{exit,outer}$	outside nozzle radius at exit, m	0.08
l_{conv}	distance from plenum to throat, m	0.05
θ_c	cone half angle, deg	50.0

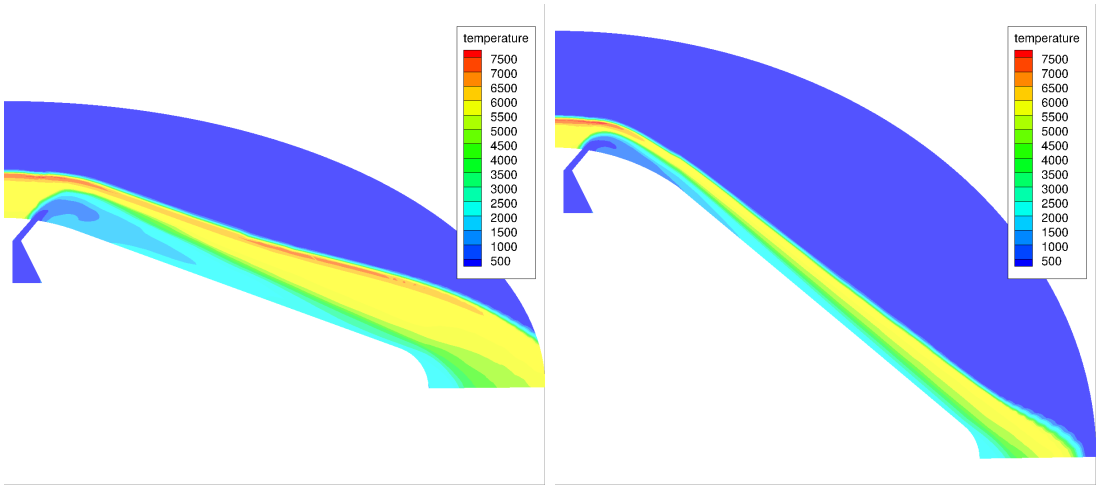
Table 5.1: Annular nozzle geometry inputs.

Flow Condition	Description	Value
V_∞	freestream velocity, m/s	5686.24
ρ_∞	freestream density, kg/m^3	0.001
T_∞	freestream temperature, K	200.0
M_∞	freestream Mach number (derived)	20.0

Table 5.2: Flow conditions.

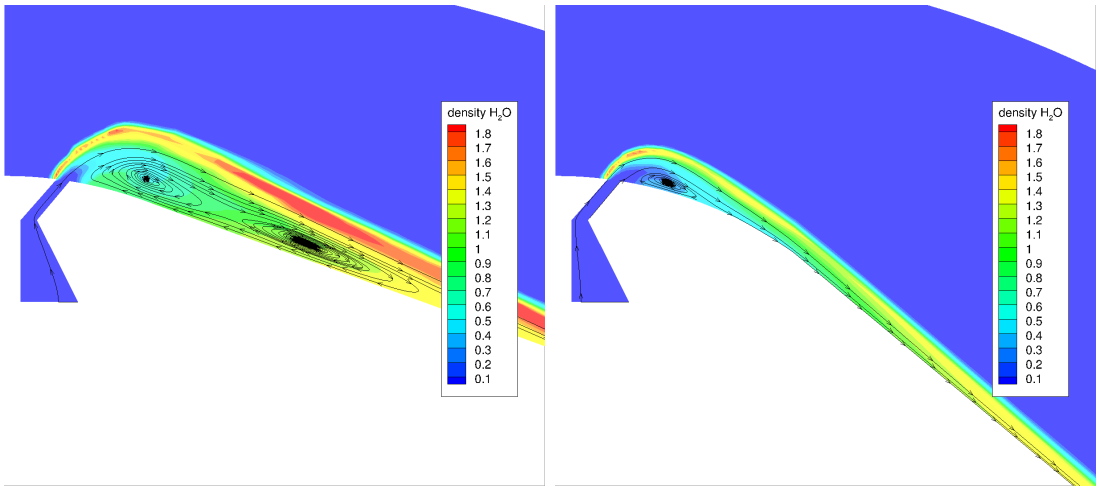
5.2 Annular Jet Flow Features and Steadiness Dependence on Cone Angle

Figures (5.2-5.3) show contours for annular plenum blowing pure hydrogen at 200,000 Pa and $500K$, and illustrate the complexity of the physics associated with this annular jet configuration. The temperature contours in Figure 5.2 show that the annular jet configuration creates a buffer gas region behind the bow shock. This buffer gas provides a beneficial cooling mechanism for the outer surface of the vehicle. The production of water can be seen in Figure 5.3, with the overlayed streamlines indicating that a separation bubble forms just downstream of the annular nozzle exit. The degree of separation in this flow is sensitive to the cone angle of the vehicle, and it has been observed that lowering the cone angle reduces the size of this separation bubble.



(a) 70^o Cone angle (b) 50^o Cone angle

Figure 5.2: Annular jet temperature contours, blowing pure H_2.

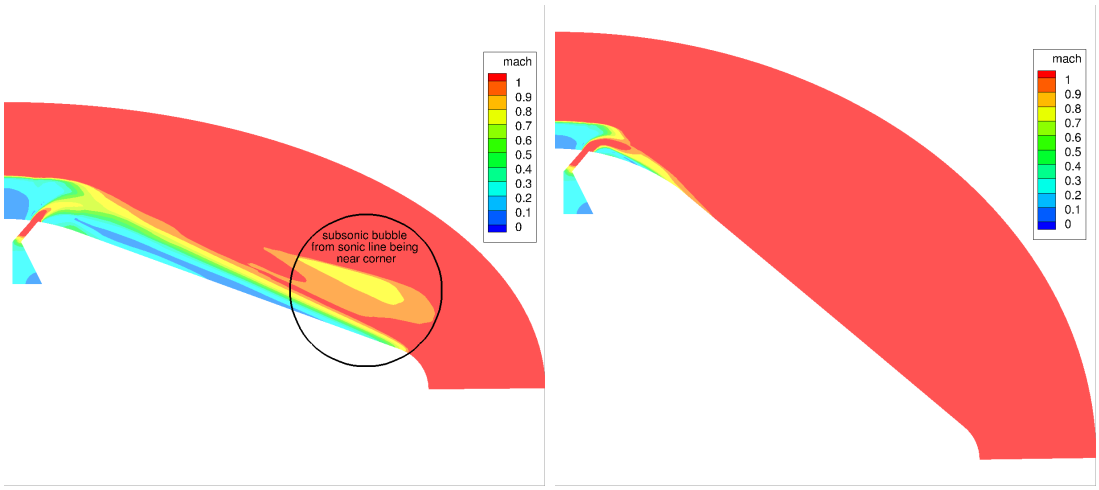


(a) 70^o Cone angle (b) 50^o Cone angle

Figure 5.3: Annular jet H_2O density contours, blowing pure H_2.

The size of the separation bubble shown in Figure 5.3 has implications on the steadiness of the flow. While the reacting gas path of the FUN3D flow solver has the capability to simulate unsteady flows, the newly developed FUN3D reacting gas adjoint solver is currently limited in scope to steady flows. A full design optimization covers a wide va-

riety of flow solutions, and it is therefore important to verify that the geometry chosen results in steady flow for all design perturbations. A 70^o cone angle was initially chosen, instead of 50^o, because of a rich history of investigation associated with the Mars Pathfinder probe[22]. It was heuristically determined that blowing a light gas with a high cone angle leads to a sonic corner body. Numerical experiments indicate that the annular jet is susceptible to very low frequency and low amplitude pulsing of the entrained separation bubble on sonic corner bodies, where the separation bubble near the nozzle exit is relatively large. Figure 5.4 shows a comparison of mach number for the same flow



(a) 70^o Cone angle (b) 50^o Cone angle

Figure 5.4: Annular jet sonic line comparison, blowing pure H_2.

conditions as those in Figures (5.2-5.3). Figure 5.4a highlights the subsonic bubble that is indicative of a sonic corner body, whereas Figure 5.4b shows that decreasing the cone angle moves the sonic line upstream and eliminates the subsonic bubble. The strong dependence between the position of the sonic line on the vehicle and the cone angle is the primary reason that the cone angle was chosen to be decreased to 50^o, rather that

using the 70^o chosen by Gnoffo et al.[21]. The comparison between the 50^o and 70^o cone angle geometries was done for a wide range of plenum conditions spanning the design space to be discussed in the next chapter, and no subsonic bubble was ever found for the 50^o cone angle.

5.3 Mesh Refinement Study

To determine the flow solution mesh sensitivity, a grid convergence study was attempted by uniformly refining the original mesh, using the cut-cell method developed by Park[41] that uniformly subdivides each element in the mesh. The solution on each grid level

Plenum Condition	Value
Plenum Pressure, $P_{p,o}$, Pa	200,000
Plenum Temperature, $T_{p,o}$, K	500
Plenum Fuel-Air Ratio ϕ_p	0.7

Table 5.3: Plenum conditions.

was computed using the freestream conditions in Table 5.2 and the plenum conditions in Table 5.3, and Figure 5.5 shows the progression of refined meshes generated by this method, and Figure 5.6 shows the grid convergence of surface temperature and plenum mass flow rate for a nominal case. On the fine mesh ($431,000$ Nodes), a frozen limiter was required to maintain stead flow. The flow solution for the very fine mesh ($3,100,000$ Nodes) was very unsteady, and freezing the flux limiter did not recover a steady flow solution. The values for surface temperature and mass flow rate on the very fine mesh are time averaged quantities presented for qualitative purposes only. This grid convergence study shows that the demonstration problem enables unsteadiness on the finest mesh. For

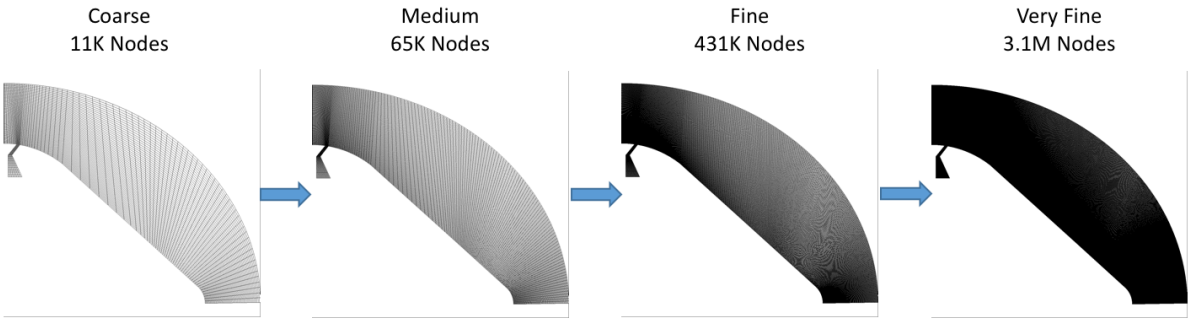


Figure 5.5: Uniformly refined meshes.

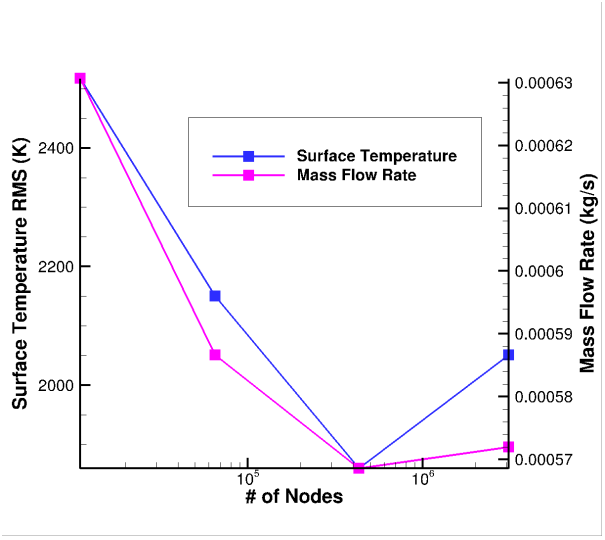


Figure 5.6: Grid convergence.

the purposes of demonstrating the decoupled discrete adjoint, the discretized flow solution must remain steady; therefore, the coarse mesh (11,000 nodes) is used as a baseline for the remainder of this study. The adjoint formulation only requires steady flow on the given grid; therefore, this test retains all of the challenging interaction associated with a large, recirculating buffer zone over the cone with a strong dependence on the reacting gas physics.

5.4 Sensitivity to Frozen Flux Limiter

The use of a flux limiter is mandatory to maintain monotonicity in hypersonic applications, where strong shocks are present; however, the solver is very sensitive to changes in the reconstruction, and a "ringing" of the residual is often observed[19] when using a flux limiter in FUN3D. While this sub-convergence of the flow equation residuals is not detrimental to the flow solver results, as most aerothermodynamic quantities are usually sufficiently converged by this point, the adjoint-formulation is predicated on the residual being machine zero. Because of this last point, the stalled convergence can cause the adjoint solver to give incorrect results or cause the adjoint solution to diverge. To prevent this divergence, the adjoint is required to be run with a "frozen" limiter, where a limiter value is only updated and re-frozen where a reconstruction is unrealisable using the current limiter value. Freezing the limiter in this way usually results in the residuals converging to machine precision.

A particular difficulty of this approach comes from simulations on shock-misaligned meshes involving chemical reactions. Because the annular jet plume and bow shock cannot remain aligned with the mesh during a design optimization, there is a larger degree of ringing as the scheme captures the discontinuities of these flow phenomena. The ringing is mitigated when the limiter is frozen; however, the solution will experience high-frequency errors as the shock and plume re-position due to the reconstruction changing somewhat asymmetrically. Species will quickly deplete and form as the shock and plume move, potentially requiring further reevaluation of the flux limiter at those nodes. This process will eventually settle out, and the flow solution will converge to machine zero, with the definition of machine zero being dependent on the conditioning of the problem being solved. Figure 5.7 shows a typical convergence history for the annular jet case with

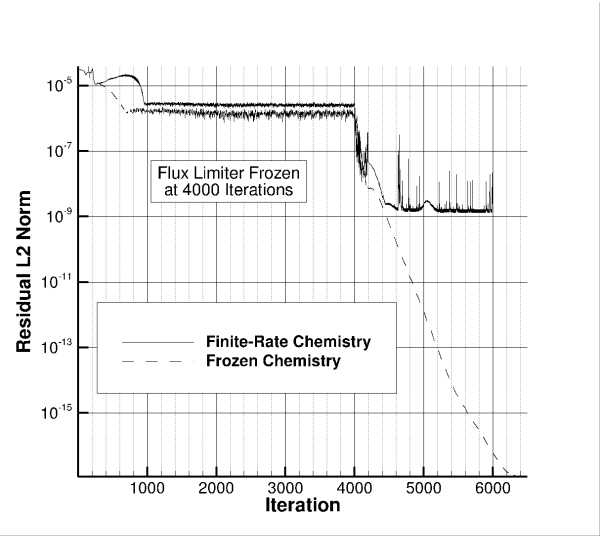


Figure 5.7: Typical residual convergence behavior.

a flux limiter engaged on the coarse mesh. If no chemical reactions are allows to take place, i.e. "frozen" flow, the residual of all equations can be decreased to to less than 10^{-16}. If finite rate chemistry is included, convergence ends at $\sim 10^{-8}$. The difference is believed to be associated with the high degree of stiffness in the chemical source terms. The large reaction rates significantly increase the condition number of the Jacobian, and further updates to the solution do not improve convergence. This is not a problem for the adjoint solution, as both levels of convergence are sufficient to obtain an adjoint solution; however, the convergence of the flow equations with chemistry is marred by the high degree of sensitivity to the flux limiter used. It can be seen in Figure 5.7 that convergence to steady state is less smooth when chemistry is involved, and this is directly tied to the flux limiter field. In FUN3D, the frozen flux limiter is re-evaluated when a reconstruction is unrealisable. This reevaluation causes high-frequency errors, which manifest as "spikes" in the residual history. Fortunately, the reevaluation is only needed at a small number of nodes (usually < 10 nodes), so integrated aerothermodynamic quantities are unaffected

by these reevaluations when convergence to steady-state is reached. It should be noted that while the results presented here are only on the coarsest (baseline) mesh level, the refined meshes also exhibit similar behavior.

Because the solution is not grid converged, there is a significant difference between the first-order and second-order solutions; therefore, the choice flux limiters presented in Section 3.4 has a large impact on the solution. Figure 5.8 shows that the RMS of

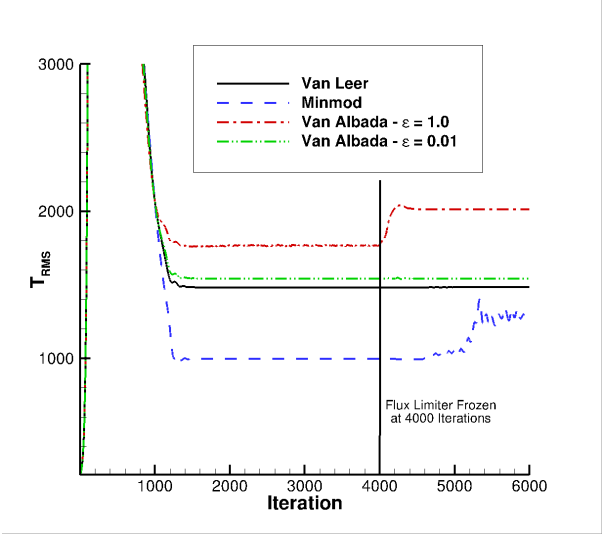


Figure 5.8: Convergence of surface temperature with different flux limiters.

surface temperature on the outer surface of the vehicle converges to significantly different results before the flux limiter is frozen at iteration 4000. The Minmod flux limiter exhibits very large oscillations after freezing, due to the large number of non-smooth reevaluations of the flux limiter were reconstructions became unrealisable. These oscillations in the solution make the Minmod flux limiter unusable for design optimization of this case. The Van Albada flux limiter converges to a much more steady solution after being frozen; however, care must be taken to "tune" this particular limiter based

the grid. The tunable parameter, ε, was evaluated at the recommended value of 1.0, and was found to increase the surface temperature by $> 20\%$ after freezing. This is unacceptable for design optimization, as it indicates a high dependence on iteration that the flux limiter is frozen. The Van Leer flux limiter behaves much better in this regard;

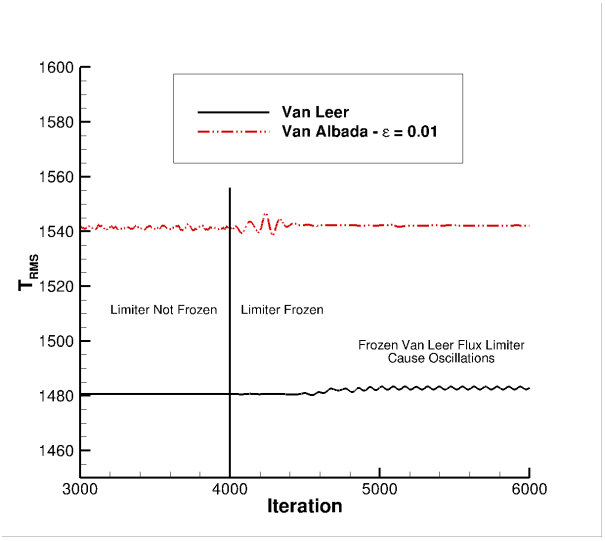


Figure 5.9: Van Leer and Van Albada flux limiter impact on surface temperature.

however, Figure 5.9 shows that the Van Leer limiter is also prone to oscillations in the surface temperature RMS. The oscillations are of small enough amplitude that it might be reasonable to use the Van Leer flux limiter in a design optimization of this problem, but a better alternative is to revisit the Van Albada flux limiter. Figure 5.9 also shows that if ε is tuned to a value of 0.01 the surface temperature RMS very closely matches the average of the ringing value before the limiter was frozen. For this reason, the Van Albada flux was used with $\varepsilon = 0.01$, which tends towards the highly limited solution, as it seems to produce the most steady result with the least dependence on the iteration that the flux limiter is frozen.

None of the aforementioned flux limiter choices are ideal. Because of the sensitivity exhibited to freezing the limiter, integrated aerothermodynamic quantities results can vary between solution with the same inputs, but different starting states. Tuning the Van Albada flux limiter mitigates this issue, but does not solve it entirely, with some cases resulting in a surface temperature RMS variance of $+/-5K$ with identical solver inputs but different flow starting conditions. This is a recognized problem, but one that is outside of the scope of this study. A number of novel discretizations at at NASA Langley Research Center[20, 30, 31] show promise in mitigating or eliminating the need for a flux limiter, while achieving higher-order spatial accuracy, and future work may incorporate them.

Chapter 6

Design Optimization

Design optimization is a wide field that encompasses methods generally falling into two categories: local gradient-based optimization, and heuristic global optimization. Local gradient-based optimization techniques focus on the determining an optimality condition by evaluating a function and its gradients. Provided certain conditions are met, it can be proven that the optimization procedure will find a local minimum or maximum on a bounded domain. Examples of local gradient-based optimization methods include steepest-descent[14], sequential quadratic programming (SQP)[16], as well as an interesting method that converts a constrained optimization problem into an unconstrained on by employing the Kreisselmeier-Steinhauser function[52]. A heuristic global optimization seeks to find the global extrema of a function. Although these methods are powerful, because of their heuristic nature they are not guaranteed to find the absolute optimum condition and are not the focus this research.

In the field of optimization, the function of interest is referred to as the "cost function" or "objective function". Optimization methods seek to minimize this function; therefore, if the intent is to find the maximum value of the function, it should be formulated as

the negative of the original. This section focuses on the optimization of the annular jet demonstration problem that was discussed in Chapter 5. Details on the implementation of the cost function components and design variables used in the optimization are presented, as well a sample first-order inverse design optimization and second-order direct design optimization. The purpose of an inverse design optimization is to match a target design by perturbing a set of design inputs, whereas the purpose of a direct design optimization is to improve an initial design by perturbing those same design inputs.

6.1 Integrated Quantities of Interest

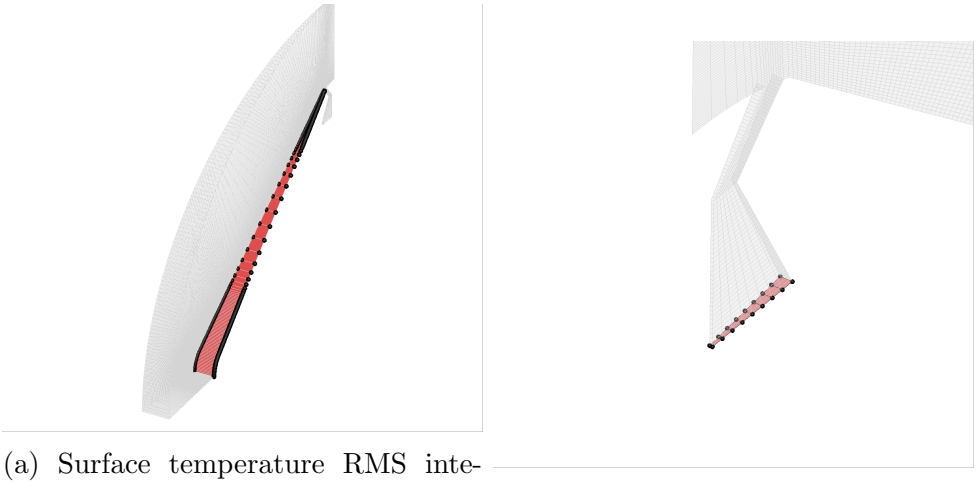
The primary objective of this optimization is to explore the effects of the plume from the annular jet interacting with the bow shock. Rather than focus on the net force contribution of this annular jet geometry to drag, as was previously investigated by Gnoffo et. al [21], the primary objective of this study is to efficiently cool the surface of the vehicle with minimal mass added to the vehicle. Since only inviscid flow was done in this study, and the root-mean-square (RMS) of surface temperature is taken as an analog to the surface heating rate. Likewise, the mass flow rate through nozzle plenum boundary is taken as an analog to the addition of mass to the vehicle.

The RMS of surface temperature is defined as

$$T_{RMS} = \sqrt{\frac{\sum_i^{N_{faces}} \left(T_{RMS} A_i\right)^2}{\sum_i^{N_{faces}} \left(A_i\right)^2}} \tag{6.1}$$

The area-weighted RMS of surface temperature was chosen over a simple area-weighted average of surface temperature, because the temperatures on the vehicle forebody are likely non-uniform, and there are regions of very high temperature near the stagnation

region of a vehicle forebody in hypersonic flows. Squaring of temperature in the RMS will give greater weight to the these high-temperature regions in the design, as these are the in the most danger of burn-through.



(a) Surface temperature RMS integrated area

(b) Mass flow rate integrated area

Figure 6.1: Cost function component integrated areas.

The mass flow rate, $\dot{m}_p$, through the area outlined in Figure 6.1b is computed as

$$\dot{m}_p = \sum_{i}^{N_{faces}} \left(\rho_i \overline{U} A\right) \tag{6.2}$$

and is used a metric for the amount of propellent that is required to be carried by the vehicle for blowing. Again, for the purposes of this demonstration problem, a lower mass flow rate at the plenum is equated to less total vehicle mass.

6.2 Composite Cost Function Definition and Components

The cost function (or objective function) as formulated in FUN3D is a composite, weighted function

$$f = \sum_{j=1}^{N_{func}} w_j \left(C_j - C_{j^*}\right)^{p_j} \tag{6.3}$$

Where w_j, C_{j^*}, and p_j are the weight, target, and power of cost function component j. C_j is the component value, which is evaluated at each flow solution. For example, an optimization problem that seeks to minimize the surface temperature RMS without decreasing the drag, the cost function is defined as

$$f = w_1 \left(T_{RMS}\right)^2 + w_2 \left(C_D - C_D^*\right)^2 \tag{6.4}$$

For this case the component weights must be determined heuristically, to normalize the changes in drag coefficient, C_D, and surface temperature Root-Mean-Square (RMS) T_{RMS}. The terms in Eq. 6.4 are squared to provide a convex design space.

6.3 Design Variables

The design variables for the optimization problem are the plenum total pressure, $P_{p,o}$, plenum total temperature, $T_{p,o}$, and plenum "fuel-air ratio", ϕ_p. These are provided explicitly in the optimization problem, and are used to directly set the flow conditions on plenum face boundary condition in the nozzle, shown in Figure 6.1b. For a reacting gas mixture, the "fuel-air ratio" specifies the mass fractions for two species leaving the

plenum. For example, if an $H_2 - N_2$ mixture is ejected from the annular nozzle, the mass fractions of H_2 and N_2 are given by

$$\begin{aligned} c_{H_2} &= \phi_p \\ c_{N_2} &= 1 - \phi_p \end{aligned} \tag{6.5}$$

Thus, the ratio ϕ_p dictates the mass fractions for two species injected into the domain via the plenum boundary.

6.4 Obtaining Sensitivity Gradients for Design Variables

The sensitivity gradients for the plenum design variables are easily obtained by manipulating Eq. 4.5 to obtain

$$\frac{\partial L}{\partial \mathbf{D}} = \frac{\partial f}{\partial \mathbf{D}} + \frac{\partial \mathbf{R}}{\partial \mathbf{D}}^T \mathbf{\Lambda} \tag{6.6}$$

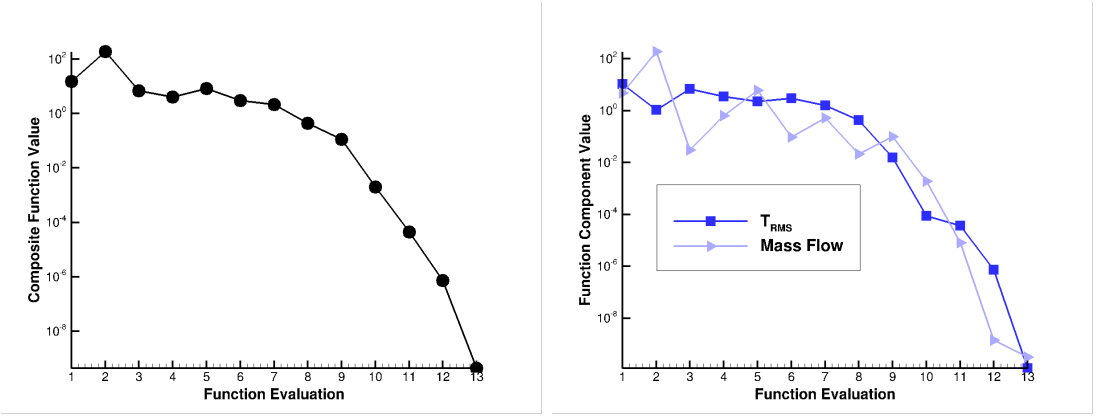
For the cost function components listed in Section 6.2, there is no direct dependence on the plenum design variables; thus, Eq. 6.6 can be reduced to

$$\frac{\partial L}{\partial \mathbf{D}} = \frac{\partial \mathbf{R}}{\partial \mathbf{D}}^T \mathbf{\Lambda} \tag{6.7}$$

Once the adjoint co-state variables $\mathbf{\Lambda}$ have been computed by solving the adjoint equations (Eq. 4.3) the sensitivity derivatives of the cost function with respect to the plenum design variables are obtained by evaluating relatively inexpensive matrix-vector products.

6.5 First-Order Inverse Design Optimization

The optimization procedure is done using the *opt_driver* utility in FUN3D, which is a wrapper utility that executes the FUN3D flow solver, adjoint solver, and optimization algorithm sequentially. These steps are repeated until a termination criterion is reached. In practice, the termination of the optimization occurs when the cost function reaches a tolerance of less that 10^{-8}, or when continuing towards the optimal condition would exceed the prescribed upper or lower bounds of the design variables. Due to the difficulties encountered with the flux limiter that were described in Section 5.4, a first-order inverse design optimization is shown in this section. Results of the second-order inverse design optimization failed due to the high sensitivity to the interaction at which the limiter is frozen.



(a) Composite value (b) Component values

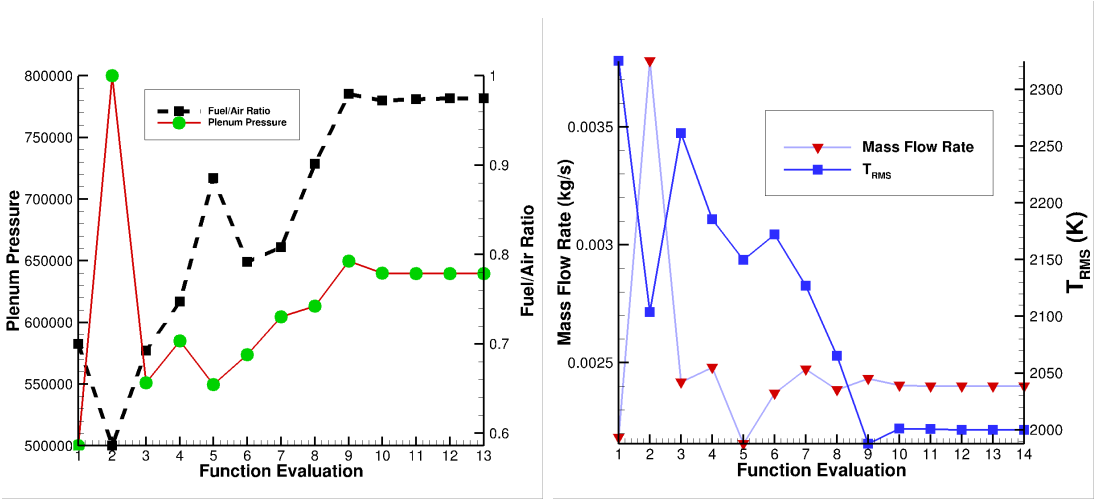
Figure 6.2: Cost function and component history.

To demonstrate the inverse design capability of an adjoint-based design optimization, targets of 2000 K for the surface temperature RMS and 0.0024 kg/s for the annular

nozzle mass flow rate were specified. These targets were chosen semi-arbitrarily, and were heuristically determined to be feasible based the design variable bounds. The design variables specified for this optimization were the plenum total pressure, $P_{p,o}$ and the plenum "fuel-air ratio", ϕ_p. The plenum total temperature, $T_{p,o}$, was fixed at 500 K. A species mixture consisting of H_2 and N_2 was blown from the plenum, with the mass fractions dictated by ϕ_p as described in Eq. 6.5. For the cost function, the weights were chosen heuristically, such that

$$\frac{w_1}{w_2} = \frac{(T_{RMS} - {T_{RMS}}^*)^2}{(\dot{m} - \dot{m}^*)^2} \tag{6.8}$$

This results in a roughly equivalent weighting between the T_{RMS} and $\dot{m}$, which is desired as both targets should be met at optimality. Using the SNOPT optimizer, Figure 6.2a



(a) Design variable history (b) Design variable history

Figure 6.3: Inverse design history.

shows that the target design was met within 13 function evaluations. SNOPT explores

the entire design space, as is show in the second function evaluation, where a spike in the cost function occurred. Figure 6.3a indicates that the optimizer tried the upper bound for the plenum pressure design variable, and found that sensitivity derivatives indicated that the lower pressure was required. This is common, and is an effective way to insure that there is a local minimum in the prescribed bound. The optimization terminated when the cost function value was less than the tolerance of 10^{-8}, and Figure 6.2b shows that both components of the cost function were within one order of magnitude of each other during the optimization. This last point is important, since non-normalized components can skew the optimization results, where competing components can cause oscillations in the function evaluations and stall the optimization procedure. Figure 6.3b shows the history of the surface temperature RMS and annular nozzle mass flow rate, with the design targets. The optimization clearly made significant progress early, with smaller gains as the solution approached the target.

6.6 Second-Order Direct Design Optimization

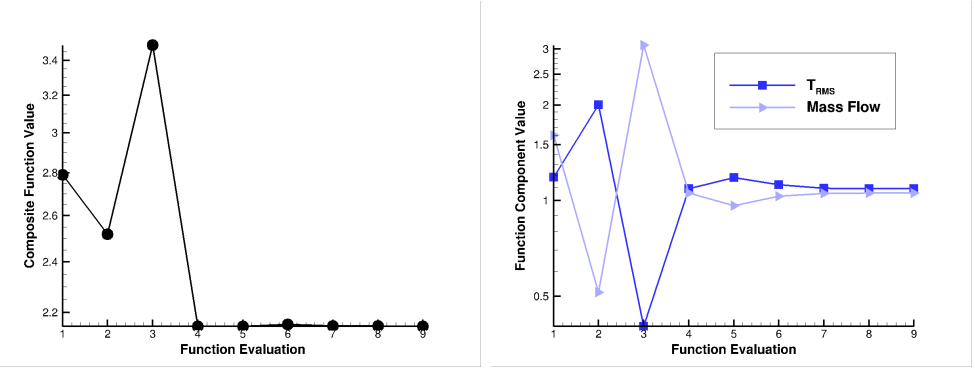
A direct design problem is possible with the second-order reconstruction scheme, by defining the termination of the optimization as the point where changes in the design variables or cost function are below a supplied threshold. Using the same cost function components and design variables as in Section 6.5, a direct design problem was completed to minimize both mass flow rate and surface temperature RMS. The cost function was formulated as

$$f = w_1 \left(\dot{m}_p\right)^2 + w_2 \left(T_{RMS}\right)^2 \tag{6.9}$$

With the weights chosen in the same manner as the inverse design problem to insure that the components are normalized to the same order of magnitude

$$\frac{w_1}{w_2} = \frac{(\dot{m}_p)^2}{(T_{RMS})^2} \tag{6.10}$$

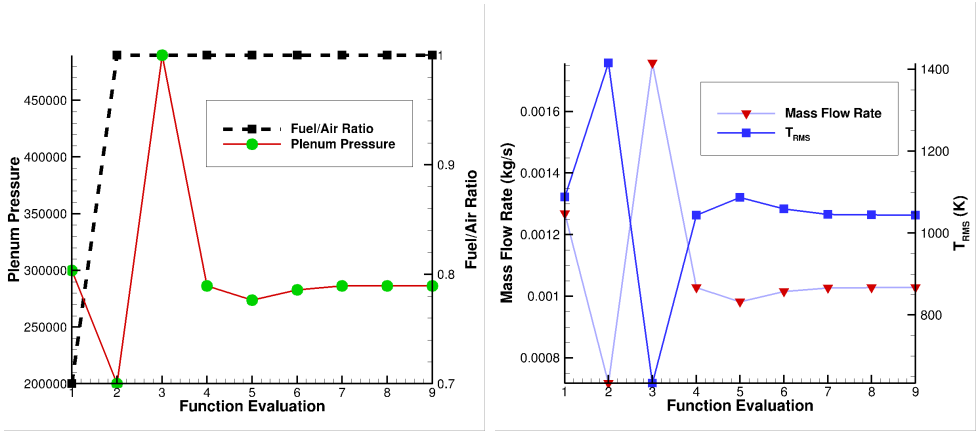
The SNOPT optimizer was able to very quickly determine that blowing pure H_2, i.e. $\phi_p = 1.0$, would yield the lowest surface temperature RMS and mass flow rate. Figure 6.4a shows that most of the improvement in the design was made within the first three function evaluations, and the subsequent steps were significantly less. The optimization was terminated by the ninth function evaluation, as no meaningful improvement to the design was made after that point. Figure 6.4b verifies that weights determined by



(a) Composite cost function value (b) Cost function components

Figure 6.4: Direct design cost function.

Eq. 6.10 were indeed sufficient to normalize $\dot{m}_p$ and T_{RMS} contributions to the composite cost function. Figure 6.5a shows that the optimization was largely dependent on the plenum pressure, and Table 6.1 shows that larger pressure at the plenum resulted in a 4.044% lower surface temperature, and 18.93% lower mass flow rate. Thus, the optimizer



(a) Design variable history. (b) $\dot{m}_p$ and T_{RMS} design history

Figure 6.5: Direct design history.

Component	Initial	Final	Improvement
$\dot{m}$, kg/s	1.268e-3	1.028e-3	18.93%
T_{RMS}, K	1088	1044	4.044%

Table 6.1: Direct design optimization improvement.

was able to improve both components of the cost function. This is an excellent problem for a high-fidelity gradient-based optimization, as the non-linear effects of plenum-shock interaction makes the optimum plenum condition difficult to determine intuitively.

There is a much stronger dependence on the choice of cost function weights for this problem than for the inverse design problem in Section 6.5. For the inverse design problem, the target mass flow rate and surface temperature RMS were known a priori; therefore, the weighting was chosen as a purely normalizing measure to accelerate convergence to the target condition. For this direct design problem the target mass flow rate and surface temperature RMS are not know a priori, and the weights chosen have a direct impact on the optimum condition. A "skewed" weighting may be advisable from an engineering perspective when attempting a direct design approach. For example, if

the surface thermal protection (TPS) is rated to withstand much higher surface heating than what is nominally predicted, a higher weight might be given to the mass flow rate in order to decreased the required vehicle mass. The heuristic nature of this approach can be avoided by setting a component target, or converting a composite cost function component to an explicit constraint. The latter option is more robust, but comes at the cost of an additional adjoint solution.

Chapter 7

Verification of Adjoint Sensitivity Gradients

In this section the sensitivity gradients computed by the adjoint-formulation are verified against finite-difference derivatives with a complex step. This details the methods by which the gradient information is computed in the forward-mode and in the reverse-mode.

7.1 Forward-mode Sensitivities Using Complex-Variables

The sensitivities can be computed in the forward-mode by using finite-difference. While this approach is unfavorable to use in practice, because a minimum number of flow solves equivalent to the number of design variables are required, it is a straight-forward way to verify that sensitivities computed by the adjoint solver are correct. To avoid cancellation errors associated with real-variable finite difference, an approach that uses complex variables was originally suggested by Squire and Trapp[47] and evaluated by Newman et

al[33]. This complex-variable approach can be used to determine the derivative of a real valued function, f, by considering the Taylor series expansion of f using a complex step ih

$$f(x+ih) = \sum_{k=0} \left(\frac{(ih)^k}{!k} \frac{\partial^k f}{\partial x^k} \right) = f(x) + ih\frac{\partial f}{\partial x} - \frac{h^2}{2}\frac{\partial^2 f}{\partial x^2} - \frac{h^3}{6}\frac{\partial^3 f}{\partial x^3} + \cdots \tag{7.1}$$

taking the imaginary parts of both sides of Eq. 7.1 and solving for the first derivative yields

$$\begin{aligned} \frac{\partial f}{\partial x} &= \frac{Im\left[f(x+ih)\right]}{h} - \frac{h^2}{6}\frac{\partial^3 f}{\partial x^3} \\ &= \frac{Im\left[f(x+ih)\right]}{h} - O(h^2) \end{aligned} \tag{7.2}$$

Thus, this complex-variable approach provides a means of computing the cost function derivatives without subtractive cancellation error, giving truly second order accuracy. The power of using this approach is that the derivative of any function with respect to a single can be computed without any additional work, other than the function be made to use complex arithmetic and a complex-valued perturbation be applied.

In practice, this complex-variable approach is implemented by transforming the source code such that all real variables used in a function evaluation are made to be complex variables. A specified step size is then added to the complex part of the design variable that the cost function is to be linearized with respect to in the function evaluation. Provided the "complexified" solver converges in the same manner as the real-variable solver, the complex part of the function of interest is the sensitivity derivative.

As mentioned before, the complex-variable and real-variable solvers must converge to the same solution for the comparison between the adjoint-computed sensitivity derivatives and those computed by the complex-variable approach to be valid. Section 5.4 described the need for a frozen limiter to satisfy the Lagrangian in Eq. 4.1, from which the discrete adjoint equations are derived. A significant obstacle with using a frozen limiter is that the

real-variable and complex-variable solvers do not freeze the limiter identically. Although the solvers are nearly identical, their floating-point operations will be different since complex arithmetic is involved in only the complex-variable solver. The flux limiter formulation is sensitive to these differences and will result in a different converged state, regardless of the limiter being frozen at the same iteration between the solvers. To overcome this, the complex-variable solver must not be allowed to change the value of the flux limiter. Enforcing consistency between the real-value and complex-value flow solver is done by converging the real-value flow solver to machine precision, and then starting the complex-variable solver with the flow field of the converged real-variable solution. The complex step is then added to the design variable; however, the flux limiter is frozen. Because limiter value is constant, the real solution from the complex-variable and real-variable solutions will match identically, and the complex part of the solution will be converged without ever needing to update the flux limiter. This will ensure that the sensitivity derivatives from the complex and adjoint solvers match to high precision for hypersonic cases with a frozen flux limiter.

7.2 Verification of 2nd-order Adjoint Linearizations

To verify the hand-coded linearizations implemented in the adjoint solver, the derivatives of the drag, surface temperature, and mass flow rate cost functions components for the annular nozzle geometry were computed using both the adjoint method and the complex-variable approach. To facilitate checking the linearizations of all of these components efficiently, a composite cost function was formed with three components

$$f = w_1 \left(\dot{m}_p - \dot{m}_p^*\right)^2 + w_2 \left(T_{RMS} - T_{RMS}^*\right)^2 + w_3 \left(C_D - C_D^*\right)^2 \tag{7.3}$$

By combining all components into a single composite function, only one real-valued flow and adjoint solution is needed to obtain the sensitivity derivatives for all design variables, computed by Eq. 6.7. One complex-valued flow solution is needed for each design variable. Table 7.1 shows the relative difference between the sensitivity derivatives computed by Eq. 6.6 and Eq. 7.2, for a perfect gas annular jet simulation. The adjoint and complex

Design Variable	Adjoint	Complex	Relative Difference
$P_{p,o}$	0.12067860106210E-04	0.12067860106758E-04	4.54e-11
$T_{p,o}$	0.36654635117980E-03	0.36654635118188E-03	5.67e-12

Table 7.1: Sensitivity derivative comparison - perfect gas.

solvers match within machine zero, which is $\sim 10^{-15}$ for this case. Table 7.2 shows the same comparison as that in Table 7.1, but with a H_2-N_2 mixture ejected in a 5-species freestream air mixture with frozen flow There is very strong agreement between

Design Variable	Adjoint	Complex	Relative Difference
$P_{p,o}$	-0.18451007644622E-06	-0.184510076442032E-06	2.27e-11
$T_{p,o}$	0.62086963151678E-03	0.620869631517086E-03	4.93e-13
ϕ_p	-0.34045335117520E-01	-0.340453351177196E-01	5.86e-12

Table 7.2: Sensitivity derivative comparison - H_2-N_2 frozen.

all the design variable sensitivities computed by the adjoint and complex solvers, with all matching discretely to within 11 digits. Table 7.3 shows the same comparison as those in Table 7.2, with reactions allowed to take place. It is clear that these do not match as well as those in Tables (7.1-7.2). The difference is explained by the relative convergence of the flow solver residuals shown in Figure 5.7. Because convergence of

Design Variable	Adjoint	Complex	Relative Difference
$P_{p,o}$	-0.11081315601976E-06	-0.110529774659536E-06	2.56e-03
$T_{p,o}$	0.19089941237390E-03	0.190892847933810E-03	3.44e-05
ϕ_p	-0.28035409045530E-01	-0.280251731728184E-01	3.65e-04

Table 7.3: Sensitivity derivative comparison - H_2-N_2 reacting.

the flow solver residuals involving chemistry typically stalls at eight orders of magnitude higher than without chemistry, it should be likewise be expected that the sensitivity derivatives also degrade in comparison. The large condition number increase from the chemical source term linearizations being added to the Jacobian is believed to be the cause of the stalled convergence. Despite this poor relative agreement, Section 6.6 demonstrates that the accuracy of the derivatives in Table 7.3 is more than sufficient to complete design optimizations.

Chapter 8

Relative Efficiency of Decoupled Flow Solver

At this point, the reacting gas adjoint in FUN3D has been utilized to obtain sensitivity information needed to drive design optimization, and the sensitivities of the fully coupled scheme have been validated against finite-difference derivatives with a complex step for accuracy. A key point of this study is to demonstrate the increase computational efficiency and relative memory saving of using a decoupled variable set for the flow and adjoint solvers, instead of a fully coupled variable set. This chapter details the benefits of applying the decoupled flow solver over the fully coupled solver for a variety of test cases, including the annular jet demonstration problem.

8.1 5 km/s Flow over Cylinder

Demonstrating the improved efficiency in cost and memory required to utilize the decoupled scheme, and that both the fully coupled and decoupled approaches converge to

the same result, a grid convergence study was conducted on a simple cylinder geometry (radius 0.5 m). Due to the presence of strong shocks in blunt body flows, it was advantageous to generate structured-type grids to preserve grid alignment with the bow shock. A 50×50, 100×100, and 200×200 family of grids were adapted using the adaptation capability in FUN3D[3] to produce shock-aligned grids. These grids serve as a surrogate for conducting a grid convergence study, in that differences observed between the decoupled and fully coupled schemes decrease as the average mesh spacing decreases. These grids are unstructured, consisting totally of hexahedra elements with a single cell in the spanwise direction, and the 50×50 grid is shown in Figure 8.1. The cell elements of these grids were also subdivided into tetrahedral elements, and it was verified that there are no issues with a true unstructured grid topology. The free stream conditions used were $V_\infty = 5000\ m/s$, $\rho_\infty = 0.001\ kg/m^3$, and $T_\infty = 200\ K$. Several chemical kinetics models were used, including a 5-species model with 5 reactions, an 11-species model with 22 reactions, and an 18-species model with 29 reactions. All cases were run in thermal equilibrium, with a one-temperature model.

8.1.1 Cylinder - Verification of Implementation

In order to be valid, the decoupled scheme must yield converged solutions that are nearly identical to those of the fully coupled system. To quantitatively assess this, we compare the predicted surface pressure, surface temperature, and the species composition on the stagnation line for both schemes. Figure 8.2 shows the predicted quantities on the 100×100 grid, for species mixture of N, N_2, O, O_2, and NO with five reactions. All results are indeed nearly identical, with temperature and pressure matching discretely to eight digits and the species mass fractions on the stagnation line matching to four digits.

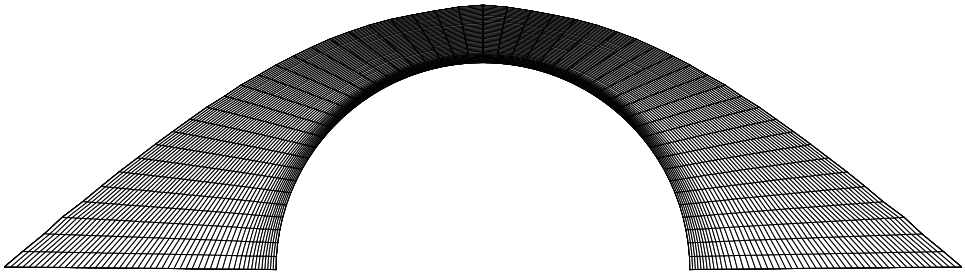
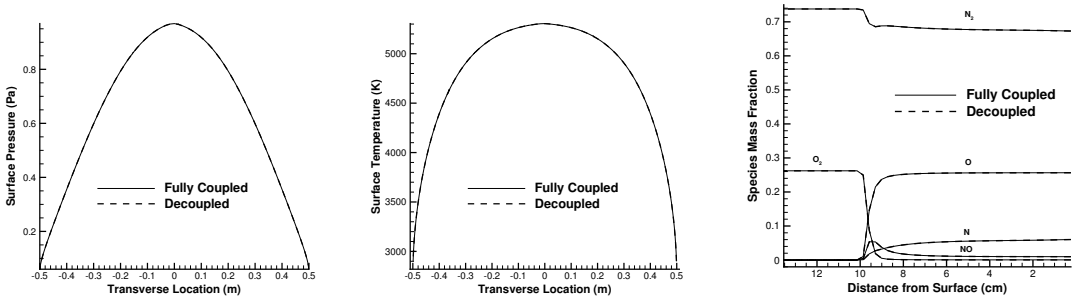


Figure 8.1: 50×50 cylinder grid.

This difference was further reduced on the finest grid level of 200×200, suggesting that both schemes converge to the same solution with grid refinement.



(a) Surface pressure (b) Surface temperature (c) Mass fractions on stagnation line

Figure 8.2: Cylinder predicted quantities.

8.1.2 Cylinder - Memory Cost

In order to determine the required memory of the decoupled scheme compared to the fully coupled scheme, a convergence study was conducted using Valgrind[1] to determine the memory actually allocated by FUN3D for an increasing number of species. Figure 8.3 shows that the relative memory cost converges asymptotically to $\sim 1/4$, which is nearly twice the predicted value of 1/7. For the implementation of FUN3D, this is correct because the off-diagonal entries are reduced from double to single precision. Each structured grid node has six neighboring nodes, with the exception of those at the boundary. Because each of these six neighboring nodes yields single precision, off-diagonal Jacobian elements,

$$N_{nz} = \frac{6N_{nodes}}{2} = 3N_{nodes} \tag{8.1}$$

Substituting Eq. 8.1 into Eq. 3.25, the relative memory cost is:

$$Relative\ Memory\ Cost = \frac{N_{nodes}}{N_{nodes} + N_{nz}} = \frac{N_{nodes}}{N_{nodes} + (3N_{nodes})} = \frac{1}{4} \tag{8.2}$$

thus, the relative memory saved by using the decoupled scheme correctly approaches a factor of 1/4.

8.1.3 Cylinder - Computational Cost

As stated before, the cost of solving the decoupled implicit system should scale approximately linearly with the number of species, whereas the fully coupled problem should scale quadratically; thus, the speedup of the implicit solve should be approximately linear when comparing the decoupled and fully coupled approaches. Figure 8.4 shows this to be true for the cylinder test case, but that the total speedup of the problem is less than

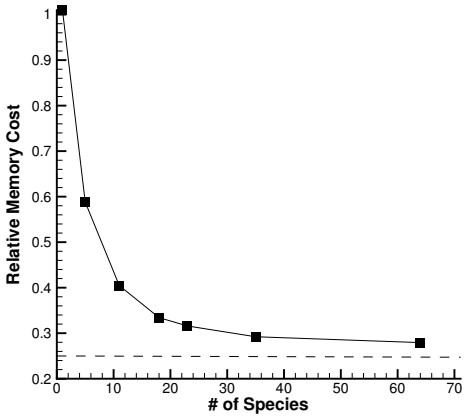


Figure 8.3: Memory required convergence study.

that of just the linear solve. The fact that the overall gains are not as large as those for

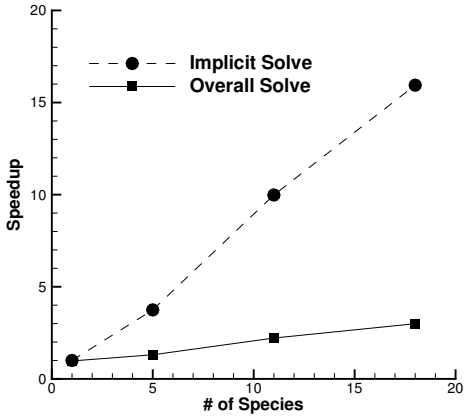


Figure 8.4: Relative speedup for the decoupled scheme vs. fully coupled scheme.

the implicit solve is due to Amdahl's law: the overall speedup is always limited by the slowest component. There are many other factors that scale with the number of species, especially calculating the species source term and its linearization. Figure 8.5 shows that

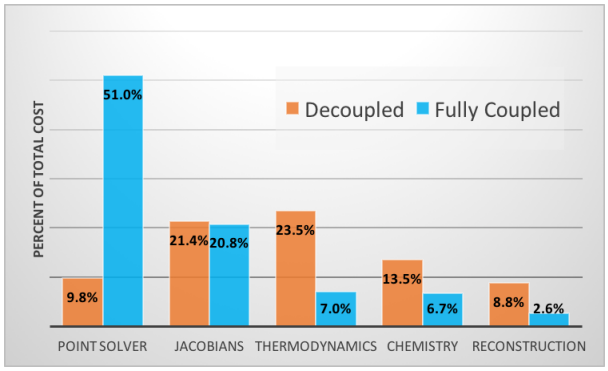


Figure 8.5: Relative cost of flow solver components for 11-species.

Jacobian evaluations and computing the thermo-chemical state have superseded the linear solver as the dominant cost of the simulation. The overall speedup of the decoupled scheme over the fully coupled scheme is constrained by these, and also highlights the sensitivity to the Jacobian evaluation frequency. When the non-linear update is rapidly converging the residuals, the Jacobian is usually updated only one every 10 iterations. Likewise, when the non-linear update is not resulting a sustained residual decrease the Jacobian is updated every iteration. This has a large effect on relative computational efficiency, and poorly converging solutions can be expected to show less relative speedup between the fully coupled and decoupled schemes.

8.2 15 km/s Flow over Spherically-Capped Cone

To ensure that the decoupled scheme is robust and accurate at higher velocities, both the fully coupled and decoupled approaches were run on a sphere-cone geometry identical to that presented by Candler et al. [10] (10 cm nose radius, 1.1 m length, 8^o cone angle). For this case, a simple 64×64 hexahedra grid was constructed, and freestream conditions were set as $V_\infty = 15000\ m/s$, $\rho_\infty = 0.001\ kg/m^3$, $T_\infty = 200\ K$. Due to the large reaction

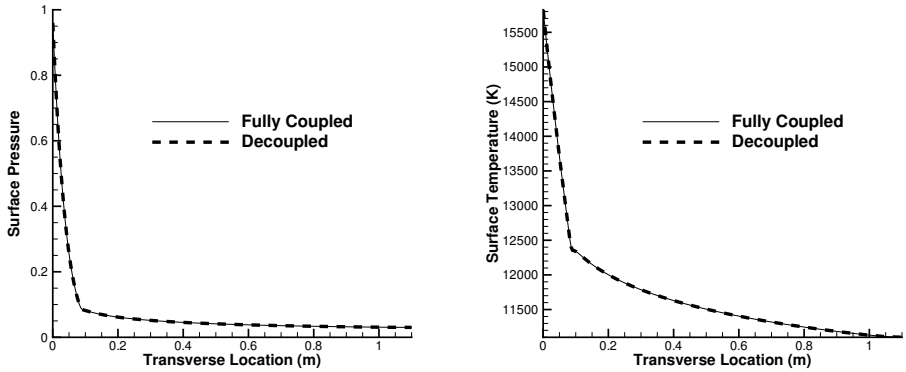
rates during the transients at startup, when the stagnation temperatures are very high, it was necessary to employ the scaling factor, ω_r in order to ramp the magnitude of chemical source term. Ramping was only required in the first 500 iterations, and no scaling was performed on the source term when the solution was well with the radius of convergence.

8.2.1 Sphere-Cone - Verification of Implementation

As with the cylinder test case, the surface pressure and temperature were used as metrics to determine that both the decoupled and fully coupled approaches give the same answer when converged to steady-state. The species composition consisted of N, N_2, O, O_2, NO, N^+, N_2^+, O^+, O_2^+, NO^+, and electrons, with 22 possible reactions. Figure 8.6 shows that both methods again yield similar results, and the high stagnation temperature indicates that this is an inviscid, one-temperature simulation. This demonstrates that the decoupled approach is able to converge to the same solution as the fully coupled solution, in spite of the chemical reactions proceeding very rapidly due to a high stagnation temperature.

8.2.2 Sphere-Cone - Convergence Quality

The limits on the stability of the decoupled scheme derives from introducing explicitness in creating and destroying species. By scaling the magnitude of the chemical source term during the transient phase of the solve, this instability can be mitigated, and the convergence of decoupled scheme approaches that of the fully coupled scheme. Scaling of chemical source term was done identically between the decoupled and fully coupled scheme by ramping the factor ω from 0.001 to 1.0 over the first 500 timesteps. Figure 8.7 shows that the convergence of both schemes progresses nearly identically, with the



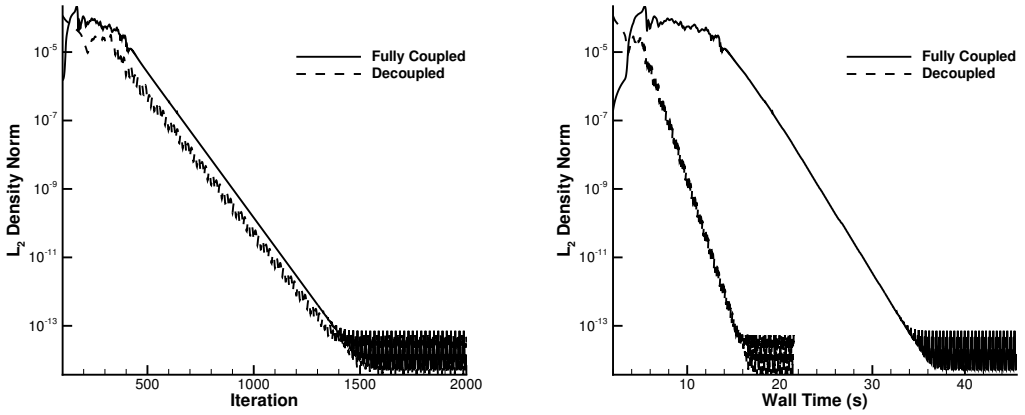
(a) Surface pressure (b) Surface temperature

Figure 8.6: Sphere-cone predicted quantities.

decoupled scheme converging in significantly less computational time and, interestingly, fewer timesteps. This demonstrates that the decoupled scheme has significant potential to improve the efficiency of high-velocity simulations, and that the stiffness of the source term can be overcome in the presence of large chemical reaction rates.

8.2.3 Sphere Cone - Convergence Improvement with Exact First-Order Linearizations

An important benefit of implementing an adjoint solver in FUN3D derives from the requirement of exact linearizations, because the linearizations needed to form the residual in the adjoint solver can be reused by the flow solver. Due to the high storage requirements associated with exactly linearizing the gradient terms in the reconstruction, an approximate Jacobian is used in the flow solver. This approximate Jacobian consists of linearizations of the first-order scheme, as is used to solve the governing equations via defect correction. Previously, the reacting gas path employed a Jacobian also, that



(a) Iterations to convergence (b) Computational time to convergence

Figure 8.7: Sphere-cone convergence details.

approximated the linearizations of the Roe FDS scheme[23] as

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}} = \frac{1}{2}\left(\frac{\partial \mathbf{R}_c}{\partial \mathbf{U}} + \frac{\partial \mathbf{R}_{\tilde{c}}}{\partial \mathbf{U}}\right) \tag{8.3}$$

where $\frac{\partial \mathbf{R}_c}{\partial \mathbf{U}}$ is the linearization of the convective portion of the Roe FDS scheme, and $\frac{\partial \mathbf{R}_{\tilde{c}}}{\partial \mathbf{U}}$ is the approximation to the dissipation term in the Roe RDS scheme, formed by evaluating $\frac{\partial \mathbf{R}_c}{\partial \mathbf{U}}$ with Roe averaged quantities. This approximation has been is used by others [43] to mitigate the development time of implementing an exact linearization of the dissipation term in the Roe FDS scheme, as well the large computational cost in computing those linearizations at each Jacobian update.

To demonstrate any benefit of the exact linearization of the Roe FDS scheme on iterative convergence of the flow solver, the sphere cone case was converged using both the Jacobian employing exact linearizations of the Roe FDS scheme flux, and the Jacobian employing the approximation in Eq. 8.3. Figure 8.8 shows that up to a five-species air

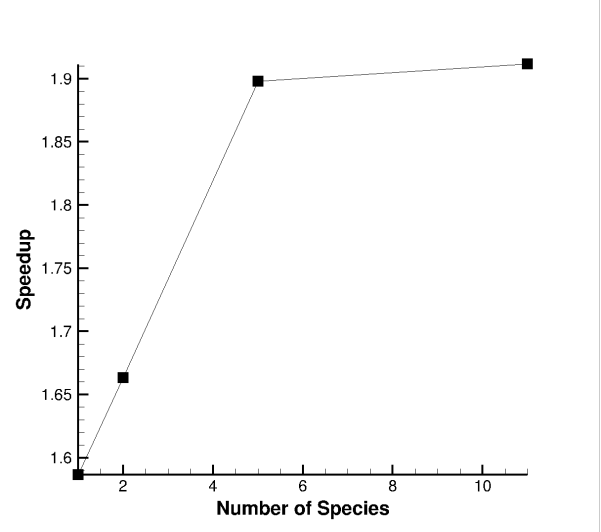


Figure 8.8: Relative speedup using exact linearizations.

mixture, the convergence to steady state is completed in 58% to 90% less simulation time. For the 11 species air mixture, involving ionization, the speed up promptly plateaus, as the additional costs incurred by computing the exact linearizations supersede the improved iterative convergence. It can be inferred from Figure 8.8 that the improvements compounded by the decoupled scheme, which affords iterative converge similar to the exact fully coupled scheme, will be significantly faster than the baseline scheme that was originally implemented in the reacting gas path of FUN3D.

8.3 Re-entry Vehicle with Retro-Firing Annular Jet

The demonstration problem presented in Chapters (5-6) presents the most challenging problem for the decoupled scheme, due to the Hydrogen-air combustion. To verify that the solution is not effected by employing the decoupled scheme over the fully coupled scheme, the comparison is made between solutions of the decoupled and fully coupled

Quantity	Decoupled	Fully Coupled	Relative Difference
$\dot{m}_p$	0.8450858226893225E-03	0.8450858226893034E-03	2.26E-14
T_{RMS}	0.1508600871984388E+04	0.1508600871984388E+04	0

Table 8.1: Difference with frozen chemistry.

Quantity	Decoupled	Fully Coupled	Relative Difference
$\dot{m}_p$	0.8448720721551258E-03	0.8448720721551088E-03	2.01E-14
T_{RMS}	0.1545729169703027E+04	0.1545720949811712E+04	5.32E-06

Table 8.2: Difference with finite-rate chemistry.

flow solvers on the cost function component quantities in Section 6.2. The freestream conditions in Table 5.2 were used for this comparison, along with the plenum conditions in Table 5.3.

8.3.1 Annular Jet: Verification of Implementation

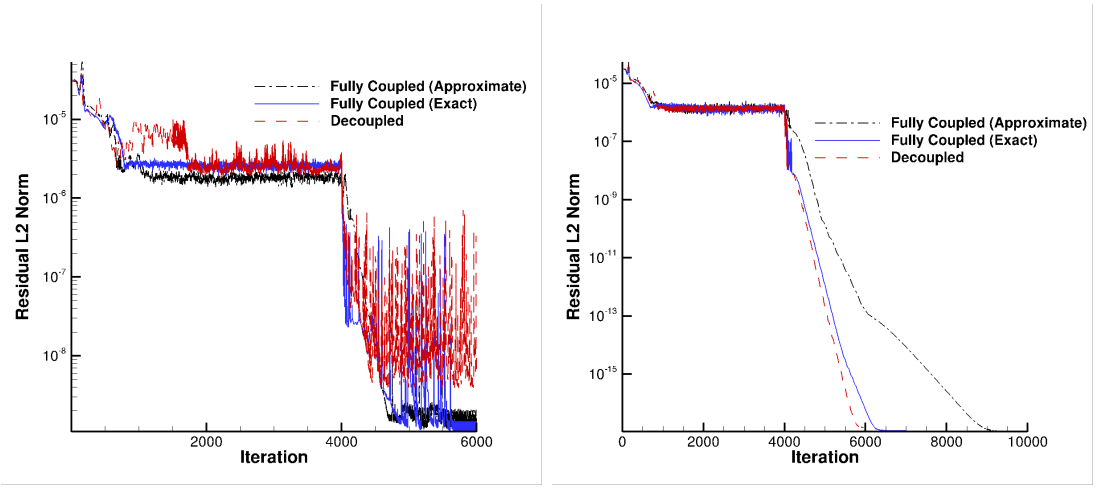
To determine consistency between the fully coupled and decoupled schemes, the surface temperature RMS, T_{RMS}, and mass flow rate, $\dot{m}_p$, were computed and compared. Because of the flux limiter sensitivity and poor relative convergence with finite-rate chemistry discussed in Section 5.4, the comparison was made for both frozen flow and flow in chemical non-equilibrium. Table 8.1 shows that, for frozen chemistry, the decoupled and fully coupled schemes match discretely for surface temperature, and to within approximately machine precision for the mass flow rate through the plenum. The residual was less than 10^{-16} for all equations in this comparison, and only the last digit of both quantities in Table 8.1 were changing by the end of the solution. Table 8.2 shows, for reacting chemistry, similar agreement with regard to the plenum mass flow rate, but poorer agreement of the surface temperature. Given the varying degrees of convergence shown in Figure 5.7, this difference in surface temperature is not surprising. At the point

the convergence stall, with the residual of all equations hanging at $\sim 10^{-8}$, only the last nine digits were still changing. This indicates that improving the conditioning of the problem, in order to enable further converge of the flow equation residuals to machine zero, would likely decrease the relative difference between the fully coupled and decoupled flow solver schemes. For engineering applications this difference is trivial and it has only minimal impact, if any, on the design optimization procedure. The effect of the flux limiter sensitivity is much greater on the design optimization procedure, and it was found that the using the decoupled scheme or the fully-coupled scheme in the optimization from Chapter 5 converged to the same result if the limiter fields were made identical between the schemes.

8.3.2 Annular Jet: Convergence Quality

Due to startup transients, where reaction rates grew very large as nearly all H_2 and O_2, and most of N_2, dissociated near the stagnation region, the source term scaling factor, ω_r, was employed for this problem to maintain stability. It was empirically determined that the source term scaling could be removed within the first third of the way to convergence for this problem.

Figure 8.9 shows a comparison of the residual convergence for the fully coupled schemes, with exact first-order linearizations and the approximations described in Section 8.2.3, and the decoupled scheme. The comparing Figures (8.9a-8.9b), there is a stark difference both the minimum residual convergence and the "smoothness" of the convergence profile. When the no chemical source terms are present (frozen flow), the flow equation residuals are able to be converged to $< 10^{-16}$, with very few high-frequency errors encountered after the limiter is frozen at 4000 iterations. When chemistry is en-



(a) Finite-rate chemistry (b) Frozen chemistry

Figure 8.9: Second-order residual convergence history.

gaged, the convergence of the flow equation residuals stalls at approximately 10^{-8}, and there are a large number of high-frequency errors in the solution updates after the limiter is frozen, due to the frozen limiter needing to be re-evaluated. Figure 8.9b shows that the decoupled scheme is more sensitive to the frozen limiter than either of the fully coupled schemes. This is not overly surprising, since the re-evaluation of the flux limiter is sensitive to subtle changes in the species densities, and the decoupled scheme Jacobians have a large number of approximations with regard to species density linearizations. As show in Section 8.3.1, the impact of this poor convergence quality on the quantities of interest is minimal, and this degree of convergence is more than sufficient to obtain the sensitivity gradients required in design optimization.

8.3.3 Annular Jet: Relative Speedup

The computational cost of decoupled scheme and fully coupled scheme with exact first-order linearizations is compared to the previously implemented fully coupled scheme

with approximate first-order linearizations. For flow with frozen chemistry, show in Figure 8.9a, the solution is considered converged when the flow residual L_2 norm is less than 10^{-16}, and the relative speedup is shown in Table 8.3 The decoupled scheme is

Scheme	Time (s)	Speedup
Fully Coupled (Approximate)	348.6	1.0 (baseline)
Fully Coupled (Exact)	265.6	1.31
Decoupled	138.7	2.51

Table 8.3: Relative speedup for frozen chemistry.

considerably faster than either of the two fully coupled schemes, and the improved iterative convergence afforded by the exact linearizations prove valuable to the fully coupled scheme for this case, where the residuals can be converged an additional 10 orders of magnitude after the limiter is frozen. Determining the relative speedup is difficult for the schemes in Figure 8.9b, where finite-rate chemistry is engaged, due to noise in the convergence history. Because of this, total solution time is used in lieu of a residual convergence level to compute the relative speedup in Table 8.4 The gains associated with chemistry

Scheme	Time (s)	Speedup
Fully Coupled (Exact)	280.6	1.0 (baseline)
Fully Coupled (Approximate)	270.4	1.04
Decoupled	168.6	1.66

Table 8.4: Relative speedup for finite-rate chemistry.

engaged are significantly less than those with frozen chemistry for this annular jet case. Due to stalled convergence, there is no clear advantage of using exact linearizations over approximate linearizations in the fully coupled scheme, although the relative savings of

use approximate linearizations are almost negligible (4%). The decoupled scheme is 66% faster than the exact, fully coupled scheme, and the difference between this result and the frozen flow result is the poorer iterative convergence quality. As discussed in Section 8.1.3, the frequency of the Jacobian evaluation is tied to the decrease in the residual from each non-linear solution update. Due to the poorer convergence seen in Figure 8.9b, the Jacobian is reevaluated at each iteration. This significantly diminishes the relative speedup of the decoupled scheme. This causes the Jacobian update to become a bottleneck in computational time required, and the additional cost of computing chemical source term and its Jacobian at each timestep reduces the relative speedup further in comparison to frozen flow.

Chapter 9

Relative Efficiency of Decoupled Adjoint Solver

At this point, the reacting gas adjoint in FUN3D has been utilized to obtain sensitivity information needed to drive design optimization, and the sensitivities of the fully coupled scheme have been validated against complex Frechet derivatives for accuracy. A key point of this study is to demonstrate the increase computational efficiency and relative memory saving of using a decoupled variable set for the flow and adjoint solvers, instead of a fully coupled variable set. This chapter details the benefits of applying the decoupled adjoint solver over the fully coupled solver for the annular jet demonstration problem.

9.1 Verification of Consistency

Section 4.2 showed in detail that, in the adjoint, changing from the fully coupled variable and equation sets to the decoupled variable and equation sets could be accomplished by a series of matrix transformations on the Jacobian. This transformation is equivalent

to a left and right preconditioning of the fully coupled scheme, enabling the reuse of the approximate Jacobians used by the flow solver. Provided the adjoint system is well conditioned and convergence is sufficient, preconditioning should not affect the final adjoint solution; thus, we expect to recover the exact same solution from both the fully coupled and decoupled adjoint schemes.

The 5 km/s spanwise cylinder case from Section 8.1 and the 15 km/s sphere cone case from Section 8.2 are re-examined here to determine the consistency of the adjoint solution computed by the fully coupled schemes. In this case, the sensitivity of the drag coefficient, C_D to the angle of attack, α, is computed by both schemes and compared. Table 9.1 shows that the solutions match discretely to at least 11 digits, and is indicative

Case	Decoupled Sensitivity	Fully Coupled Sensitivity	Rel. Diff
Cylinder	0.764409218044021E-07	0.764409218054372E-07	1×10^{-11}
Sphere-Cone	-0.877773486565165E-02	-0.877773486565157E-02	2×10^{-14}

Table 9.1: Angle of attack sensitivity relative difference.

that the decoupled iterative mechanism can be used to compute the adjoint solution without changing the result. It should be recognized here that, in the continuous sense, the drag on a cylinder is independent of angle of attack; however, in the discrete sense of this grid, which is only a half-body cylinder, there is a minimal dependence between angle of attack and drag.

For the annular jet demonstration problem, the sensitivities computed in Table 7.3 were checked to verify that decoupled and fully coupled adjoint solutions matched to give the same sensitivity. Table 9.2 also gives excellent agreement between the decoupled and fully coupled adjoint schemes. Examining the differences of sensitivities in Tables (9.1-

Design Variable	Decoupled Sensitivity	Fully Coupled Sensitivity	Rel. Diff
$P_{p,o}$	-0.11081315601976E-06	-0.11081315649260E-06	4×10^{-9}
$T_{p,o}$	0.19089941243495E-03	0.19089941237390E-03	3×10^{-10}
ϕ_p	-0.28035409093945E-01	-0.28035409045530E-01	1×10^{-9}

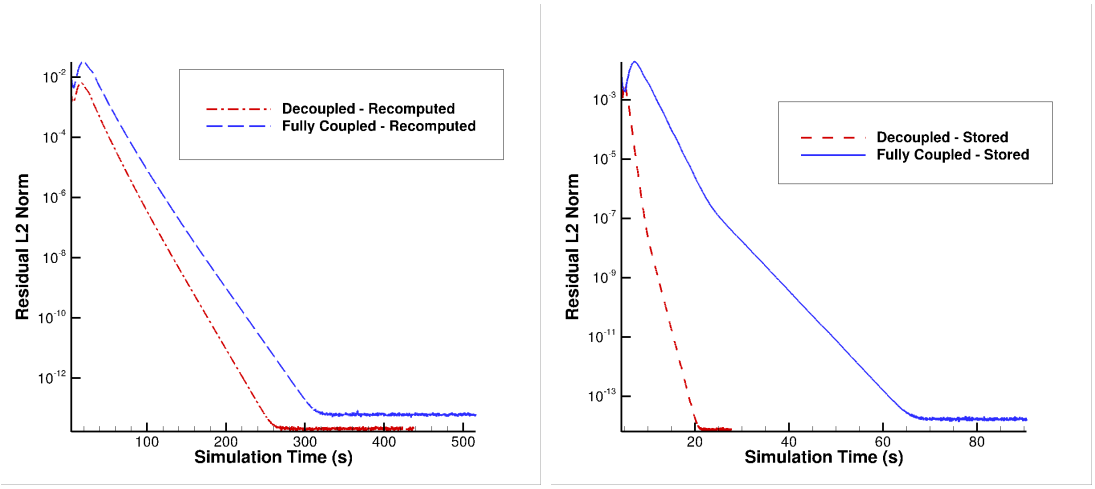
Table 9.2: Annular jet plenum sensitivity relative difference.

9.2), it is clear that the absolute difference is almost always on the order of 10^{-15} each case. Since double precision was used in all floating point operations for these cases, this is consistent with the level of precision used.

As a note on robustness, none of the stability issues found in the decoupled flow solver have been observed for the decoupled adjoint scheme. The non-linear nature of the chemical source term was the primary stability concern in the decoupled flow solver, since the explicitness introduced by the decoupled mass fraction update exacerbates CFL restrictions. These problems disappear in the adjoint equations, since it is a linear system being solved instead of a highly non-linear one. The full stability and convergence properties of the fully coupled adjoint solver were recovered by the decoupled adjoint solver, and it is therefore concluded that there is no advantage of solving the adjoint system in a fully coupled manner.

9.2 Relative Memory Savings and Speedup

The computational cost of the solution process in the adjoint can be greatly improved by storing the exact linearizations of the discrete flow solver equations, as discussed in Section 4.4. For a second order scheme Figure 9.1 shows the comparison of the fully coupled adjoint and decoupled adjoint time to solution with the two mechanisms for computing the exact linearizations. Table 9.3 shows that the speedup of decoupled scheme



(a) Linearizations always recomputed (b) Linearizations stored

Figure 9.1: Simulation time comparison.

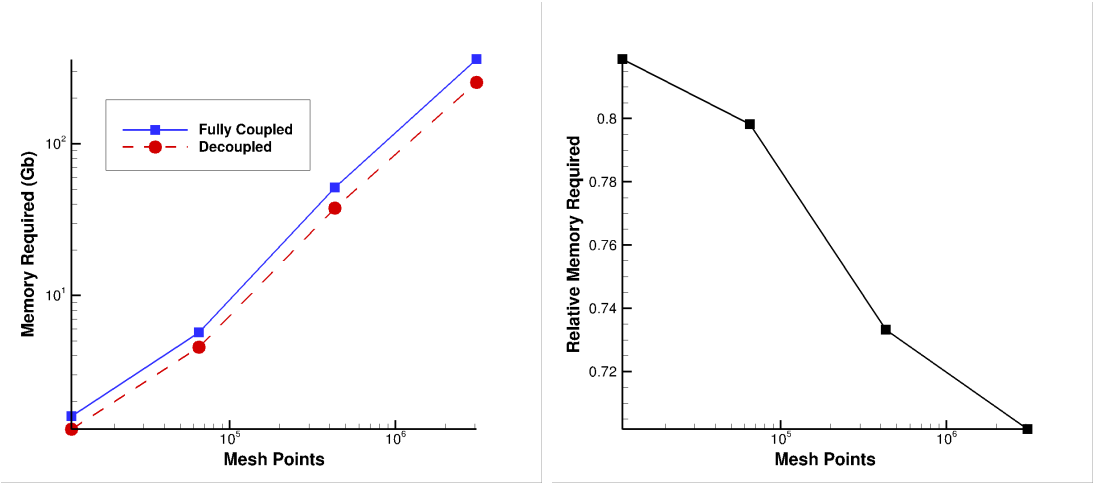
is only truly realized when the exact linearizations are store, rather than recomputed at each timestep. This is because computing the exact linearizations in the adjoint is the

Scheme	Linearizations	Time to Convergence (s)	Speedup
Fully Coupled	Recomputed	309.4	1.0 (baseline)
Decoupled	Recomputed	244.2	1.27
Fully Coupled	Stored	61.22	5.05
Decoupled	Stored	18.69	**16.6**

Table 9.3: Relative speedup.

dominant cost. The comparison between storing the linearizations and recomputing them is somewhat unfair, as the ability to store the full Jacobian of the second order reconstruction may not be possible, due to memory constraints. That said, a significant advantage of the decoupled scheme over the fully coupled scheme is the memory savings offered by the LHS Jacobian sparsity. It was show in Section 3.3 that the predicted

memory savings for a infinite number of species is 1/7 for a hexahedra grid where the average number of neighbors is 6. Unfortunately, the storage requirements for the exact linearizations will not change between the decoupled and fully coupled schemes; however, as the average number of neighbors in the grid increases, the relative memory savings will also increase. To effectively demonstrate this, the amount of memory required by the adjoint solver with the exact linearizations stored was determined for each mesh of the refinement study in Section 5.3. Figure 9.2 is somewhat encouraging a face value,



(a) Absolute memory required (b) Relative memory required

Figure 9.2: Memory required to store full linearizations.

since it was possible to store the exact linearizations for the 3.1 million node mesh, simulating a 9-species hydrogen air mixture. Figure 9.2a show, as expected, that the memory requirements for both the fully coupled and decoupled schemes scale directly with the number of mesh points. The relative savings for this case are nowhere near the predicted 1/7 value, because the exact linearizations require significantly more memory than anything else store in the adjoint solver. That said, the $\sim 20\%$ decrease in memory

required by the decoupled adjoint solver is significant, since the ability to store the exact linearizations is a binary problem: either there is enough memory or there is not. The increased savings are as important as the speedup, if not more, since the decoupled scheme can facilitate larger problems to be solved with greater than an order of magnitude increase in computational efficiency, if the exact linearizations can be stored.

Chapter 10

Concluding Remarks

An implicit, decoupled scheme has been implemented in the reacting gas path of the FUN3D flow solver, and an adjoint solver employing a fully coupled and decoupled scheme has also been implemented. The decoupled adjoint scheme demonstrated here is a significant improvement to the state of the art in the field of sensitivity analysis for reacting flows. The decreased memory and computational costs afforded by solving a decoupled dependent variable set in the adjoint solver enabled a speedup greater than an order of magnitude for some cases.

The consistency and robustness of decoupling the variable sets in the flow solver was examined. It was found that the decoupled scheme only exhibited stability issues when the chemical source term was very large, due to large reaction rates. This stability issue was mitigated by multiplying the chemical source term uniformly by scaling term, which could be ramped from zero to one over the course of the simulation. Provided that the ramping was completed, the true steady-state solution, done without ramping, was recovered. The consistency of the solutions computed by the fully coupled and decoupled methods was found to match discretely to eight digits for surface temperature

and pressure, and four digits for mass fractions on the stagnation line of a spanwise cylinder case in a 5 km/s flow. It was also determined that this difference in mass fraction was reduced with mesh spacing.

Sample inverse and direct design optimizations were completed on a hypersonic re-entry vehicle geometry that employed a retro-firing annular jet. Using the sensitivity derivatives computed by the adjoint, the annular jet plenum conditions were modified to improve the surface temperature RMS and mass flow rate through the annular jet plenum. The direct design case demonstrated the ability to tune a design based on favorable engineering attributes, and serves as a proof-of-concept for future work to develop re-entry vehicles with active thermal protection systems.

To determine the accuracy of the sensitivity derivatives provided by the adjoint solver, a complex-variable approach was employed. By transforming the source code of FUN3D, complex-variable Frechet derivatives were computed and compared to those computed by the adjoint solver. Both of these were found to match to a high degree of accuracy when both the flow solver and adjoint solver were sufficiently converged. Cases involving strong chemical reactions were typically found to produce the greater differences between the flow and adjoint solvers; however, the accuracy of the sensitivity derivatives was still sufficient to complete direct and inverse design optimizations.

An analog to the decoupled iterative mechanism in the flow solver was derived for the adjoint solver, and its consistency and efficiency compared to the fully coupled scheme was examined. The adjoint solution was found to discretely match the fully coupled solution, and numerical tests showed that the decoupled adjoint scheme was over three times more efficient than the fully coupled adjoint scheme for the annular jet demonstration problem, if the exact linearizations were stored rather than recomputed at each time step. The decoupled adjoint scheme also afforded 20% less memory required than the

fully coupled adjoint scheme, and the decoupled scheme was over 16 times faster than the fully coupled scheme when the decoupled scheme utilized storing the exact linearizations and the fully coupled scheme did not. Future work will include methods to speedup the adjoint exact linearization computation, as well as schemes that employ mixed storage and recomputing of these linearizations, in order to further improve the efficiency of the decoupled adjoint scheme.

The ability to perform high-fidelity sensitivity analysis and design optimization on reacting flows is significantly helped by the increased computational efficiency that the decoupled flow and adjoint schemes provide. The improvement to iterative convergence provided by exact linearizations in the Jacobians, also compounds the beneficial aspects of the decoupled schemes. It is hoped that these improvements will encourage interest in the field of adjoint-based sensitivity analysis, by enabling solutions to problems that were intractable due to the high computational and memory cost requirements. Future work will extend these schemes to viscous flows, incorporate multi-temperature models, and apply adjoint-based sensitivities to mesh adaptation and mesh movement strategies.

REFERENCES

[1] ACM SIGPLAN 2007 Conference on Programing Language Design and Implementation. *How to Shadow Every Byte of Memory Used by a Program*, San Diego, CA, June 2007.

[2] Dinesh Balagangadhar and Subrata Roy. Design sensitivity analysis and optimization of steady fluid-thermal systems. *Computer Methods in Applied Mechanics and Engineering*, 190(42):5465–5479, 2001.

[3] Robert Bartels, Veer Vatsa, Jan-Renee Carlson, and Raymond Mineck. Fun3d grid refinement and adaptation studies for the ares launch vehicle. In *28th AIAA Applied Aerodynamics Conference*. AIAA, 2015/10/06 2010.

[4] Oktay Baysal and Mohamed E Eleshaky. Aerodynamic design optimization using sensitivity analysis and computational fluid dynamics. *AIAA journal*, 30(3):718–725, 1992.

[5] Robert T Biedron, Jan-Reneé Carlson, Joseph M Derlaga, Peter A Gnoffo, Dana P Hammond, William T Jones, Bil Kleb, Elizabeth M Lee-Rausch, Eric J Nielsen, Michael A Park, et al. Fun3d manual: 12.9. 2016.

[6] Martin A. Reno Bonnie J. McBride, Sanford Gordon. Coefficients for calculating thermodynamic and transport properties of individual species. Technical report, NASA, 1993.

[7] Clarence OE Burg. Higher order variable extrapolation for unstructured finite volume rans flow solvers. *AIAA Paper*, 4999:2005, 2005.

[8] Clarence OE Burg, Chunhua Sheng, James C Newman III, Wesley Brewer, Eric Blades, and David L Marcum. Verification and validation of forces generated by an unstructured flow solver. *AIAA paper*, 3983:2003, 2003.

[9] Graham Candler, Pramod Subbareddy, and Ioannis Nompelis. Analysis of a decoupled implicit method for aerothermodynamics and reacting flows. In *51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 1006, 2013.

[10] Graham V. Candler, Pramod K. Subbareddy, and Ioannis Nompelis. Decoupled implicit method for aerothermodynamics and reacting flows. *AIAA Journal*, 51(5):1245–1254, 2015/04/23 2013.

[11] Sean R. Copeland, Francisco Palacios, and Juan J. Alonso. Adjoint-based aerothermodynamic shape design of hypersonic vehicles in non-equilibrium flows. In *52nd Aerospace Sciences Meeting*. American Institute of Aeronautics and Astronautics, 2014.

[12] Mohammad K. Esfahani and Guillaume Houzeaux. Implementation of discrete adjoint method for parameter sensitivity analysis in chemically reacting flows. In *57th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*. American Institute of Aeronautics and Astronautics, 2016/09/20 2016.

[13] Paul F. Fischer. *Scaling Limits for PDE-Based Simulation (Invited)*. American Institute of Aeronautics and Astronautics, 2015/10/21 2015.

[14] Roger Fletcher and Michael JD Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6(2):163–168, 1963.

[15] Alan George and Joseph W. Liu. *Computer Solution of Large Sparse Positive Definite*. Prentice Hall Professional Technical Reference, 1981.

[16] Philip E. Gill, Walter Murray, and Michael A. Saunders. Snopt: An sqp algorithm for large-scale constrained optimization. *SIAM Review*, 47(1):99–131, 2005.

[17] Philip E. Gill, Elizabeth Wong, Walter Murray, and Michael Saunders. *User's Guide for NPSOL 5.0: A FORTRAN Package for Nonlinear Programming*, July 1998.

[18] Philip E. Gill, Elizabeth Wong, Walter Murray, and Michael Saunders. *User's Guide for SNOPT Version 7.4: Software Large-Scale Nonlinear Programming*, January 2015.

[19] Peter A Gnoffo. Simulation of stagnation region heating in hypersonic flow on tetrahedral grids. *AIAA Paper*, 3960(7):52, 2007.

[20] Peter A Gnoffo. Global series solutions of nonlinear differential equations with shocks using walsh functions. *Journal of Computational Physics*, 258:650–688, 2014.

[21] Peter A Gnoffo, Kyle Thompson, and Ashley Korzun. Tapping the brake for entry, descent, and landing. In *46th AIAA Fluid Dynamics Conference*, page 4277, 2016.

[22] Peter A Gnoffo, K James Weilmuenster, Robert D Braun, and Christopher I Cruz. Influence of sonic-line location on mars pathfinder probe aerothermodynamics. *Journal of Spacecraft and Rockets*, 33(2):169–177, 1996.

[23] R. N.; Gnoffo, P. A.; Gupta and J. L. Shinn. Conservation equations and physical models for hypersonic air flows in thermal and chemical nonequilibrium. Technical Paper 2867, NASA, 1989.

[24] Ami Harten. High resolution schemes for hyperbolic conservation laws. *Journal of Computational Physics*, 49(3):357–393, 1983.

[25] P.W. Jansen and R.E. Perez. Constrained structural design optimization via a parallel augmented lagrangian particle swarm optimization approach. *Computers & Structures*, 89(13–14):1352 – 1366, 2011.

[26] Brian Lockwood and Dimitri Mavriplis. Parameter sensitivity analysis for hypersonic viscous flow using a discrete adjoint approach. In *48th AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 447, 2010.

[27] Brian Lockwood, Markus Rumpfkeil, Wataru Yamazaki, and Dimitri Mavriplis. Uncertainty quantification in viscous hypersonic flows using gradient information and surrogate modeling. In *49th AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, page 885, 2010.

[28] Robert W. MacCormack and Graham V. Candler. The solution of the navier-stokes equations using gauss-seidel line relaxation. *Computers and Fluids*, 17(1):135–150, 1989.

[29] Dimitri J. Mavriplis. Multigrid solution of the discrete adjoint for optimization problems on unstructured meshes. *AIAA Journal*, 44(1):42–50, January 2006.

[30] Alireza Mazaheri and Hiroaki Nishikawa. Very efficient high-order hyperbolic schemes for time-dependent advection–diffusion problems: Third-, fourth-, and sixth-order. *Computers & Fluids*, 102:131–147, 2014.

[31] Alireza Mazaheri and Hiroaki Nishikawa. High-order shock-capturing hyperbolic residual-distribution schemes on irregular triangular grids. *Computers & Fluids*, 131:29–44, 2016.

[32] Marian Nemec, Michael J. Aftosmis, Scott M. Murman, and Thomas H. Pulliam. Adjoint formulation for an embedded-boundary cartesian method. AIAA Paper 2005–877, 2005.

[33] James C Newman, W Kyle Anderson, and David L Whitfield. Multidisciplinary sensitivity derivatives using complex variables. *Mississippi State University Publication, MSSU-EIRS-ERC-98-08*, 1998.

[34] Eric J Nielsen. *Aerodynamic design sensitivities on an unstructured mesh using the Navier-Stokes equations and a discrete adjoint formulation.* PhD thesis, Citeseer, 1998.

[35] Eric J Nielsen and W Kyle Anderson. Recent improvements in aerodynamic design optimization on unstructured meshes. *AIAA journal*, 40(6):1155–1163, 2002.

[36] Eric J Nielsen, James Lu, Michael A Park, and David L Darmofal. An implicit, exact dual adjoint solution method for turbulent flows on unstructured grids. *Computers & Fluids*, 33(9):1131–1155, 2004.

[37] Tanner Nielsen and Jack R Edwards. Numerical simulation of supersonic premixed turbulent combustion. In *55th AIAA Aerospace Sciences Meeting*, page 0601, 2017.

[38] Francisco Palacios, Juan Alonso, Karthikeyan Duraisamy, Michael Colonno, Jason Hicken, Aniket Aranake, Alejandro Campos, Sean Copeland, Thomas Economon, Amrita Lonkar, et al. Stanford university unstructured (su 2): an open-source integrated computational environment for multi-physics simulation and design. In *51st AIAA Aerospace Sciences Meeting Including the New Horizons Forum and Aerospace Exposition*, page 287, 2013.

[39] Chul Park. On convergence of computation of chemically reacting flows. In *AIAA, Aerospace Sciences Meeting*, volume 1, 1985.

[40] Chul Park. Assessment of two-temperature kinetic model for ionizing air. *Journal of Thermophysics and Heat Transfer*, 3(3):233–244, 1989.

[41] Michael Andrew Park. *Anisotropic output-based adaptation with tetrahedral cut cells for compressible flows*. PhD thesis, Massachusetts Institute of Technology, 2008.

[42] James J Reuther, Antony Jameson, Juan J Alonso, Mark J Rimlinger, and David Saunders. Constrained multipoint aerodynamic shape optimization using an adjoint formulation and parallel computers, part 1. *Journal of aircraft*, 36(1):51–60, 1999.

[43] Enrico Rinaldi, Rene Pecnik, and Piero Colonna. Exact jacobians for implicit navier–stokes simulations of equilibrium real gas flows. *Journal of Computational Physics*, 270:459–477, 2014.

[44] P.L Roe. Approximate riemann solvers, parameter vectors, and difference schemes. *Journal of Computational Physics*, 43(2):357 – 372, 1981.

[45] PL Roe. Characteristic-based schemes for the euler equations. *Annual review of fluid mechanics*, 18(1):337–365, 1986.

[46] Youcef Saad and Martin H Schultz. Gmres: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986.

[47] William Squire and George Trapp. Using complex variables to estimate derivatives of real functions. *Siam Review*, 40(1):110–112, 1998.

[48] Joseph L Steger and R.F Warming. Flux vector splitting of the inviscid gasdynamic equations with application to finite-difference methods. *Journal of Computational Physics*, 40(2):263 – 293, 1981.

[49] GD Van Albada, Bram Van Leer, and WW Roberts Jr. A comparative study of computational methods in cosmic gas dynamics. In *Upwind and High-Resolution Schemes*, pages 95–103. Springer, 1997.

[50] Bram Van Leer. Towards the ultimate conservative difference scheme. v. a second-order sequel to godunov's method. *Journal of computational Physics*, 32(1):101–136, 1979.

[51] VN Vatsa and JA White. Calibration of a unified flux limiter for ares-class launch vehicles from subsonic to supersonic speeds. *Joint ARMY-NAVYNASA-Air Force, Las Vegas, NV, USA, April. JANNAF Paper*, (2009), 2009.

[52] Gregory A Wrenn. An indirect method for numerical optimization using the kreisselmeir-steinhauser function. 1989.

APPENDIX

Appendix A

Derivations

A.1 Decoupled Flux Derivation

For the Roe flux difference splitting scheme, the species mass fluxes are given by

$$F_{\rho_s} = \frac{\rho_s^L \overline{U}^L + \rho_s^R \overline{U}^R}{2} - \frac{\tilde{c}_s(\lambda_1 dv_1 + \lambda_2 dv_2) + \lambda_3 dv_{3_s}}{2} \tag{A.1}$$

$$dv_1 = \frac{p^R - p^L + \tilde{\rho}\tilde{a}(\overline{U}^R - \overline{U}^L)}{\tilde{a}^2} \tag{A.2}$$

$$dv_2 = \frac{p^R - p^L - \tilde{\rho}\tilde{a}(\overline{U}^R - \overline{U}^L)}{\tilde{a}^2} \tag{A.3}$$

$$dv_{3_s} = \frac{\tilde{a}^2(\rho_s^R - \rho_s^L) - \tilde{c}_s(p^R - p^L)}{\tilde{a}^2} \tag{A.4}$$

$$\lambda_1 = \mid \overline{\mathbf{U}} + \tilde{a} \mid, \quad \lambda_2 = \mid \overline{\mathbf{U}} - \tilde{a} \mid, \quad \lambda_3 = \mid \overline{\mathbf{U}} \mid \tag{A.5}$$

where the ˜ notation signifies a Roe-averaged quantity, given by:

$$\tilde{\mathbf{U}} = \tilde{w}\tilde{\mathbf{U}}^L + (1-\tilde{w})\tilde{\mathbf{U}}^R \tag{A.6}$$

$$\tilde{w} = \frac{\tilde{\rho}}{\tilde{\rho}+\rho^R} \tag{A.7}$$

$$\tilde{\rho} = \sqrt{\rho^R \rho^L} \tag{A.8}$$

The species mass fluxes must sum to the total mass flux; thus, the total mixture mass flux is given as

$$F_\rho = \sum_s F_{\rho_s} = \frac{\rho^L \overline{U}^L + \rho^R \overline{U}^R}{2} - \frac{\lambda_1 dv_1 + \lambda_2 dv_2 + \lambda_3 dv_3}{2} \tag{A.9}$$

$$dv_3 = \frac{\tilde{a}^2(\rho^R - \rho^L) - (p^R - p^L)}{\tilde{a}^2} \tag{A.10}$$

Multiplying Eq. A.9 by the Roe-averaged mass fraction and substituting it into Eq. A.1 results in:

$$F_{\rho_s} = \tilde{c}_s F_\rho + \frac{(c_s^L - \tilde{c}_s)\rho^L(\overline{U}^L + \mid \tilde{U} \mid)}{2} + \frac{(c_s^R - \tilde{c}_s)\rho^R(\overline{U}^R - \mid \tilde{U} \mid)}{2} \tag{A.11}$$

It should be noted here that the Roe-averaged normal velocity, $\tilde{U}$, requires an entropy correction in the presence of strong shocks[24]. This correction has a dependence on the roe-averaged speed of sound, and therefore has a dependence on the species mass fractions; however, through numerical experiments it has been determined that omitting this dependence does not adversely affect convergence. The notation can be further

simplified by defining the normal velocities as follows:

$$\lambda^+ = \frac{\overline{U}^L + \mid \tilde{U} \mid}{2}, \quad \lambda^- = \frac{\overline{U}^R - \mid \tilde{U} \mid}{2} \tag{A.12}$$

Finally, substituting Eq. A.12 into Eq. A.11 yields the final result for calculating the species flux in the decoupled system:

$$F_{\rho_s} = \tilde{c}_s F_\rho + (c_s^L - \tilde{c}_s)\rho^L \lambda^+ + (c_s^R - \tilde{c}_s)\rho^R \lambda^- \tag{A.13}$$

Forming the convective contributions to the Jacobians is straightforward. Because the $\mathbf{U}'$ level variables are constant, only the left, right, and Roe-averaged state mass fractions vary. Differentiating Eq. A.13 with respect to the mass fraction, c_s, the left and right state contributions are

$$\frac{\partial F_{\rho_s}}{\partial c_s^L} = \tilde{w} F_\rho + (1 - \tilde{w})\rho^L \lambda^+ - \tilde{w}\rho^R \lambda^- \tag{A.14}$$

$$\frac{\partial F_{\rho_s}}{\partial c_s^R} = (1 - \tilde{w}) F_\rho + (\tilde{w} - 1)\rho^L \lambda^+ + \tilde{w}\rho^R \lambda^- \tag{A.15}$$

Because there is no dependence between species in decoupled convective formulation, the Jacobian block elements are purely diagonal for the convective contributions, of the form

$$\begin{pmatrix} \frac{\partial F_{\rho_1}}{\partial c_1} & & 0 \\ & \ddots & \\ 0 & & \frac{\partial F_{\rho ns}}{\partial c_{ns}} \end{pmatrix} \tag{A.16}$$

A.2 Quadratic Interpolation Between Thermodynamic Curve Fits

We seek to blend the two thermodynamic curve fits in such a way that we maintain c_0 continuity in both specific heat (C_p) and enthalpy (h). To accomplish this, a quadratic function must be used, of the form

$$aT^2 + bT + c = C_p \tag{A.17}$$

The coefficients a, b, and c are determined by solving the system that results from the boundary value problem

$$\begin{cases} a{T_1}^2 + bT_1 + c = C_{p_1} \\ a{T_2}^2 + bT_2 + c = C_{p_2} \\ a\frac{\left({T_2}^3 - {T_1}^3\right)}{3} + b\frac{\left({T_2}^2 - {T_1}^2\right)}{2} + c\left(T_2 - T_1\right) = h_2 - h_1 \end{cases} \tag{A.18}$$

Where the x_1 and x_2 subscripts describe the left and right states, respectively. Solving the linear system, the coefficients are

$$\begin{cases} a = \frac{3\left(C_{p_2} + C_{p_1}\right)}{(T_2 - T_1)^2} - \frac{6(h_2 - h_1)}{(T_2 - T_1)^3} \\ b = -\frac{2\left[(C_{p_2} + 2C_{p_1})T_2 + (2C_{p_2} + C_{p_1})T_1\right]}{(T_2 - T_1)^2} + \frac{6(T_2 + T_1)(h_2 - h_1)}{(T_2 - T_1)^3} \\ c = \frac{C_{p_1}T_2(T_2 + 2T_1) + C_{p_2}T_1(T_1 + 2T_2)}{(T_2 - T_1)^2} - \frac{6T_1T_2(h_2 - h_1)}{(T_2 - T_1)^3} \end{cases} \tag{A.19}$$

This can be simplified to

$$\begin{cases} a = 3B - A \\ b = \frac{-2(C_{p_1}T_2 + C_{p_2}T_2)}{(T_2 - T_1)^2} + (T_2 + T_1)(A - 2B) \\ c = \frac{C_{p_1}{T_2}^2 + C_{p_2}{T_1}^2}{(T_2 - T_1)^2} + T_1 T_2 (2B - A) \end{cases} \tag{A.20}$$

$$A = \frac{6(h_2 - h_1)}{(T_2 - T_1)^3} \tag{A.21}$$

$$B = \frac{C_{p_2} + C_{p_1}}{(T_2 - T_1)^2} \tag{A.22}$$

Note that this does not ensure that entropy will be continuous across curve fits.

A.3 Change of Variable Sets

The decoupled scheme developed by Candler et. al[10] is based upon the change of variables

$$\mathbf{U} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{ns} \\ \rho\mathbf{u} \\ \rho E \end{pmatrix} \rightarrow \mathbf{V} = \begin{pmatrix} c_1 \\ \vdots \\ c_{ns} \\ \rho \\ \rho\mathbf{u} \\ \rho E \end{pmatrix} \tag{A.23}$$

To avoid confusion between variable sets, we re-write the variable vectors, $\mathbf{U}$ and $\mathbf{V}$, in a more generic sense

$$\mathbf{U} = \begin{pmatrix} u_1 \\ \vdots \\ u_{ns+2} \end{pmatrix} \rightarrow \mathbf{V} = \begin{pmatrix} v_1 \\ \vdots \\ v_{ns+3} \end{pmatrix} \tag{A.24}$$

For simplicity, consider a system with two species, ρ_1 and ρ_2. Using the relationship $\rho_s = c_s\rho$, then the original variable vector, $\mathbf{U}$ can be rewritten in terms of the new variables, $\mathbf{V}$ as

$$\mathbf{U} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} v_1 v_3 \\ v_2 v_3 \\ v_4 \\ v_5 \end{pmatrix} \tag{A.25}$$

This allows the derivation of the Jacobian

$$\frac{\partial \mathbf{U}}{\partial \mathbf{V}} = \begin{pmatrix} \frac{\partial u_1}{\partial v_1} & \frac{\partial u_1}{\partial v_2} & \frac{\partial u_1}{\partial v_3} & \frac{\partial u_1}{\partial v_4} & \frac{\partial u_1}{\partial v_5} \\ \frac{\partial u_2}{\partial v_1} & \frac{\partial u_2}{\partial v_2} & \frac{\partial u_2}{\partial v_3} & \frac{\partial u_2}{\partial v_4} & \frac{\partial u_2}{\partial v_5} \\ \frac{\partial u_3}{\partial v_1} & \frac{\partial u_3}{\partial v_2} & \frac{\partial u_3}{\partial v_3} & \frac{\partial u_3}{\partial v_4} & \frac{\partial u_3}{\partial v_5} \\ \frac{\partial u_4}{\partial v_1} & \frac{\partial u_4}{\partial v_2} & \frac{\partial u_4}{\partial v_3} & \frac{\partial u_4}{\partial v_4} & \frac{\partial u_4}{\partial v_5} \end{pmatrix} = \begin{pmatrix} v_3 & 0 & v_1 & 0 & 0 \\ 0 & v_3 & v_2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.26}$$

At this point, it is important to note that the Jacobian in Eq. A.26 has two psuedo-inverse matricies, that correspond to the right and left inverse. The right inverse, $\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\big|_R$,

can be constructed based on the previously defined steps

$$\left.\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\right|_R = \begin{pmatrix} \frac{\partial v_1}{\partial u_1} & \frac{\partial v_1}{\partial u_2} & \frac{\partial v_1}{\partial u_3} & \frac{\partial v_1}{\partial u_4} \\ \frac{\partial v_2}{\partial u_1} & \frac{\partial v_2}{\partial u_2} & \frac{\partial v_2}{\partial u_3} & \frac{\partial v_2}{\partial u_4} \\ \frac{\partial v_3}{\partial u_1} & \frac{\partial v_3}{\partial u_2} & \frac{\partial v_3}{\partial u_3} & \frac{\partial v_3}{\partial u_4} \\ \frac{\partial v_4}{\partial u_1} & \frac{\partial v_4}{\partial u_2} & \frac{\partial v_4}{\partial u_3} & \frac{\partial v_4}{\partial u_4} \\ \frac{\partial v_5}{\partial u_1} & \frac{\partial v_5}{\partial u_2} & \frac{\partial v_5}{\partial u_3} & \frac{\partial v_5}{\partial u_4} \end{pmatrix} = \begin{pmatrix} \frac{1-v_1}{v_3} & \frac{-v_1}{v_3} & 0 & 0 \\ \frac{-v_2}{v_3} & \frac{1-v_2}{v_3} & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.27}$$

It is easily verified that the matrix product of Eq.s (A.26-A.27) produces identity

$$\left.\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\,\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\right|_R = \begin{pmatrix} v_3 & 0 & v_1 & 0 & 0 \\ 0 & v_3 & v_2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1-v_1}{v_3} & \frac{-v_1}{v_3} & 0 & 0 \\ \frac{-v_2}{v_3} & \frac{1-v_2}{v_3} & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.28}$$

however, Eq. A.28 is not associative

$$\left.\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\right|_R \frac{\partial \mathbf{U}}{\partial \mathbf{V}} = \begin{pmatrix} \frac{1-v_1}{v_3} & \frac{-v_1}{v_3} & 0 & 0 \\ \frac{-v_2}{v_3} & \frac{1-v_2}{v_3} & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_3 & 0 & v_1 & 0 & 0 \\ 0 & v_3 & v_2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.29}$$

$$= \begin{pmatrix} 1-v_1 & -v_1 & \frac{-(v_1)^2-v_1v_2+v_1}{v_3} & 0 & 0 \\ -v_2 & 1-v_2 & \frac{-(v_2)^2-v_1v_2+v_2}{v_3} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

to correctly compute identity, the property of matrix transpose multiplication is used

$$\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\left.\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\right|_R\right)^T = \left.\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\right|_R^T \frac{\partial \mathbf{U}}{\partial \mathbf{V}}^T$$

$$= \begin{pmatrix} \frac{1-v_1}{v_3} & \frac{-v_2}{v_3} & 1 & 0 & 0 \\ \frac{-v_1}{v_3} & \frac{1-v_2}{v_3} & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_3 & 0 & 0 & 0 \\ 0 & v_3 & 0 & 0 \\ v_1 & v_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.30}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

This is critical in understanding the relationships needed to switch transform variables sets. For linearizations of the residual, $\mathbf{R}$, the correct transformation from the variable set $\mathbf{U}$ to the variable set $\mathbf{V}$ is

$$\frac{\partial \mathbf{R}}{\partial \mathbf{V}} = \frac{\partial \mathbf{R}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \tag{A.31}$$

which is intuitively understood; however, the transformation from the variable set $\mathbf{V}$ to the variable set $\mathbf{U}$ must follow Eq. A.30

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}} = \left(\frac{\partial \mathbf{R}}{\partial \mathbf{V}}^T \frac{\partial \mathbf{V}}{\partial \mathbf{U}}^T \right)^T \tag{A.32}$$

The transposition in Eq. A.31 is critical, as the linearizations will be incorrect if the multiplication is done without it. Fortunately, Eq. A.31 is rarely seen in practice, as most linearizations are done for the fully-coupled system that requires Eq. A.32 to transform the linearizations

$$\frac{\partial \mathbf{R}}{\partial \mathbf{V}} = \frac{\partial \mathbf{R}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}} = \begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_1} & \dots & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho E} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_1} & \dots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_1} & \dots & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_1} & \dots & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix} \begin{pmatrix} \rho & \dots & 0 & c_1 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \rho & c_{ns} & 0 & 0 \\ 0 & \dots & 0 & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.33}$$

likewise, in the adjoint the transformation is applied to the tranpose of the Jacobian

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T = \frac{\partial \mathbf{U}}{\partial \mathbf{V}}^T \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T = \begin{pmatrix} \rho & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & \rho & 0 & 0 \\ c_1 & \dots & c_{ns} & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_1} & \dots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_1} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_1} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_1} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_{ns}} & \dots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_{ns}} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho \mathbf{u}} & \dots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho E} & \dots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix} \tag{A.34}$$

Since the tranformation is left-multiplied, the matrix vector products of the exact Jacobian with costate variables, Λ, in the adjoint linear system can be done first, and the

transformation can then be applied to the system

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \Lambda = \begin{pmatrix} \rho & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & \rho & 0 & 0 \\ c_1 & \dots & c_{ns} & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_1}\Lambda_{\rho_1} & \dots & + \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_1}\Lambda_{\rho_{ns}} & + \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_1}\Lambda_{\rho \mathbf{u}} & + \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_1}\Lambda_{\rho E} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_{ns}}\Lambda_{\rho_1} & \dots & + \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_{ns}}\Lambda_{\rho_{ns}} & + \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_{ns}}\Lambda_{\rho \mathbf{u}} & + \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_{ns}}\Lambda_{\rho E} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho \mathbf{u}}\Lambda_{\rho_1} & \dots & + \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho \mathbf{u}}\Lambda_{\rho_{ns}} & + \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}}\Lambda_{\rho \mathbf{u}} & + \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}}\Lambda_{\rho E} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho E}\Lambda_{\rho_1} & \dots & + \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho E}\Lambda_{\rho_{ns}} & + \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E}\Lambda_{\rho \mathbf{u}} & + \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E}\Lambda_{\rho E} \end{pmatrix} \tag{A.35}$$

This indicates the important point that the transformation of the adjoint residual is not dependent on the number of equations solved, but only the number of dependent variables the equations are linearized with respect to, namely $\mathbf{U}$.

A.4 Change of Equation Sets

In addition to a change of variables, the decoupled scheme also changes the equations being solved, effectively solving is one more equation than the fully-coupled scheme. The residual vector for the decoupled scheme, $\mathbf{R_V}$, and the residual vector for the fully

coupled scheme, $\mathbf{R_U}$, can be written as

$$\mathbf{R_U} = \begin{pmatrix} \mathbf{R}_{\rho_1} \\ \vdots \\ \mathbf{R}_{\rho_{N_s}} \\ \mathbf{R}_{\rho\mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \rightarrow \mathbf{R_V} = \begin{pmatrix} \mathbf{R}_{\rho_1} - c_1 \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \vdots \\ \mathbf{R}_{\rho_{N_s}} - c_{N_s} \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \mathbf{R}_{\rho\mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \tag{A.36}$$

Eq. A.36 shows that $\mathbf{R_V}$ is completely comprised of components from $\mathbf{R_U}$; therefore, a relationship between these equation sets can be derived in a similar fashion to Appendix A.3. Rewriting $\mathbf{R_V}$ in terms of $\mathbf{R_U}$

$$\mathbf{R_V} = \begin{pmatrix} \mathbf{R}_{\mathbf{U}_s} - c_s \left(\sum_{i=1}^{N_s} \mathbf{R}_{\mathbf{U}_i} \right) \\ \sum_{i=1}^{N_s} \mathbf{R}_{\mathbf{U}_i} \\ \mathbf{R}_{\mathbf{U}_{N_s+1}} \\ \mathbf{R}_{\mathbf{U}_{N_s+2}} \end{pmatrix} \tag{A.37}$$

it is possible to form the transformation

$$\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}} = \begin{pmatrix} 1-c_1 & -c_1 & \dots & -c_1 & 0 & 0 \\ -c_2 & 1-c_2 & \dots & -c_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -c_{N_s} & -c_{N_s} & \dots & 1-c_{N_s} & 0 & 0 \\ 1 & 1 & \dots & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 \end{pmatrix} \tag{A.38}$$

The adjoint costate variable vector Λ is effectively defined as

$$\Lambda = -\frac{\partial f}{\partial \mathbf{R}} \tag{A.39}$$

Thus, there are two different costate variable vectors for the equation sets defined in Eq. A.37

$$\Lambda_{\mathbf{U}} = -\frac{\partial f}{\partial \mathbf{R_U}} \tag{A.40}$$

$$\Lambda_{\mathbf{V}} = -\frac{\partial f}{\partial \mathbf{R_V}} \tag{A.41}$$

based on Eq.s (A.40-A.41), it clear that the transformation defined in Eq. A.38 is the mapping between these costate variable sets

$$\Lambda_{\mathbf{U}} = \left(\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}}\right)^T \Lambda_{\mathbf{V}} \tag{A.42}$$

Eq. A.42 enables a right preconditioning of the adjoint system of equations, where the costate variables associated with the decoupled system of equations can be transformed into the costate variables associated with fully coupled system of equations, as post-processing step. This also decreases the length of the solution vector storage required, since the fully coupled system has one less equation than the decoupled system.