# 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.)

Approach is described to efficiently compute aerothermodynamic design sensitivities using a decoupled approach. In this approach, the species continuity equations are decoupled from the mixture continuity, momentum, and total energy equations for the Roe flux difference splitting scheme in both the flow and adjoint solvers. 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.

A design optimization of a re-entry vehicle with a annular nozzle on the forebody is completed based on this approach. The sensitivities of the drag coefficient and surface temperature with respect to a plenum inboard the vehicle were computed and verified against complex-variable finite-difference.

# 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<br>Co-chair of Advisory Committee | Peter Gnoffo<br>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

# LIST OF FIGURES

# 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 $s$, $J/kg^3$ |
| $E$ | Total energy per unit mass, $J/kg^3$ |
| $E_{f,r}$ | Activation energy of reaction $r$ |
| $F'_\rho$ | Decoupled scheme mixture mass flux |
| $F'_{\rho_s}$ | Decoupled scheme mass flux of species $s$ |
| $\mathbf{F},\ \mathbf{F}',\ \hat{\mathbf{F}}$ | 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 $kg/s$ |
| $ns$ | Number of species |
| $\mathbf{n}$ | Face unit normal vector |
| $n_x$, $n_y$, $n_z$ | Face unit normal vector components |
| $N$ | 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 |
| $R_\rho$ | 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, $J/kg^3$ |
| $\mathbf{S}$ | Face normal vector, $m^2$ |
| $\mathbf{U}, \mathbf{V}, \mathbf{U}', \hat{\mathbf{U}}$ | Conservative variable vectors |
| $\overline{U}$ | Normal velocity, $m/s^2$ |
| $u,\ v,\ w$ | 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 |
| $\Lambda$ | Diagonal eigenvalue matrix |
| $\phi$ | flux limiter result |
| $\phi_p$ | plenum fuel-air ratio |
| $\rho$ | Mixture density, $kg/m^3$ |
| $\rho_s$ | 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 $r$ |
| $\omega$ | Chemical source term scaling factor |
| $\omega_s$ | Chemical source term of species $s$ |

### Subscripts

| | |
|---|---|
| $o$ | Reference value |
| $\infty$ | Freestream condition |

### Superscripts

| | |
|---|---|
| $^*$ | Cost function component target |
| $n, n+1$ | Time level |
| $n_{f,r}$ | Preexponential factor of reaction $r$ |
| $R,\ L$ | 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 that can be coupled with optimization packages to enable design optimization for a variety of design inputs[4, 2]. In recent years, the benefits of using an adjoint-based formulation to compute sensitivities have been realized and implemented in many compressible CFD codes[24, 25, 28], because of the ability compute all sensitivities at the cost of a single extra adjoint solution, instead of an additional flow solution for each design variable. Reacting gas CFD codes have lagged significantly in adopting this adjoint-based approach, with only a small number of codes having published results[9, 10]. This 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[32].

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 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[11].

To mitigate this scaling issue, Candler et al.[8] proposed a scheme to for a modified form of the Steger-Warming flux vector splitting scheme[23, 36]. In that work, it was shown that quadratic scaling between the cost of solving the implicit system and adding 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.

The work presented demonstrates that this decoupling can be applied to both the flow solver and adjoint solver in a reacting gas CFD code, and significantly improve the efficiency in both computational cost and memory required. Additionally the implementation of exact linearizations is shown to significantly improve performance and robustness in the flow solver over common approximations used when linearizing the Roe FDS scheme. The formulation and linearization of the fluxes for the Roe FDS scheme are presented here, and design optimization for an inviscid reacting flow is conducted for inviscid, reacting flow around an axi-symmetric hypersonic re-entry vehicle with an annular jet.

# 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 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** :

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

## 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

$$E = \sum c_s e_s + \frac{u^2 + v^2 + w^2}{2} \tag{2.6}$$

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[5] were used to compute $C_{p,s}$, and quadratic blending function was used to ensure that $C_{p,s}$ and enthalpy were continuous across temperature ranges. The details of this blending are given in Appendix A.

## 2.3 Chemical Kinetics Model

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

$$w_s = M_s \sum_{r=1}^{N_r} \left( \nu''_{s,r} - \nu'_{s,r} \right) \left( R_{f,r} - R_{b,r} \right) \tag{2.10}$$

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

$$R_{f,r} = 1000 \left[ k_{f,r} \prod_{s=1}^{N_s} \left(0.001 \rho_s / M_s\right) \nu^{'}_{s,r} \right] \tag{2.11}$$

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

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 (2.11-2.12) 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[30], but for a one-temperature model; thus, the forward and back rate coefficients are defined as

$$k_{f,r} = C_{f,r} T^{n_{f,r}} exp\left(-E_{f,r}/kT\right) \tag{2.13}$$

$$k_{b,r} = \frac{k_{f,r}}{K_{c,r}} \tag{2.14}$$

where $K_{c,r}$ is the equilibrium constant for reaction $r$, and $k$ is the Boltzmann constant. The preexponential factors $C_{f,r}$ and $n_{f,r}$, as well as the activation energy, $E_{f,r}$, are documented by Gnoffo[20]. The equilibrium coefficient is computed according to the curve fit defined by Park[29]

$$K_{c,r} = exp(B^r_1 + B^r_2 \ln Z + B^r_3 Z + B^r_4 Z^2 + B^r_5 Z^3) \tag{2.15}$$

$$Z = 10000/T \tag{2.16}$$

where the curve fit constants, $B_i^r$, are also documented by Gnoffo[20]

# Chapter 3

# Numerical Solution of Flow Equations

## 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:

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

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, and $e_s$ is the internal energy of species $s$ as defined in Eq. 2.8. By using the Roe FDS scheme,

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

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

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

or, put more simply:

$$A \delta \mathbf{U}^n = \mathbf{b} \tag{3.6}$$

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:

$$A = O + D \tag{3.7}$$

Each matrix element is a square $(ns + 4) \times (ns + 4)$ matrix. One method of solving this system is a Red-Black Gauss-Seidel scheme[33], where matrix coefficients with even indices are updated first and, subsequently, the coefficients with odd indices are updated. This red-black ordering enables better vectorization in solving the linear system. The computational work for the Gauss-Seidel scheme is dominated by matrix-vector multiplications of elements of $O$ with $\delta\mathbf{U}$, which are $O(N^2)$ operations, where $N = ns + 4$. In the next section, it is shown that decoupling the system reduces these matrix-vector multiplications to $O(M^2 + N)$ operations, where $M = 5$ and $N = ns$.

## 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

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

Solving the flux vector is performed in two sequential steps. The mixture fluxes are first solved as

$$\frac{\partial \mathbf{U}'}{\partial t} + \frac{1}{V}\sum_f (\mathbf{F}' \cdot \mathbf{N})^f = 0 \tag{3.9}$$

followed by the species fluxes as

$$\frac{\partial \hat{\mathbf{U}}}{\partial t} + \frac{1}{V}\sum_f (\hat{\mathbf{F}} \cdot \mathbf{N})^f = \hat{\mathbf{W}} \tag{3.10}$$

Point relaxation uses Red-Black Gauss-Seidel to update the conserved variables in $\mathbf{U}'$ and all associated auxiliary variables, such as temperature, pressure, speed of sound, etc. This is done by holding the thermo-chemical state constant, and will always result in the relaxation of a five-equation system. This 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[30].

The solution of the species mass equations takes a different form. Based on the work of Candler et al.[8], the decoupled variables can be rewritten in terms of mass fraction, as follows:

$$\delta\hat{\mathbf{U}}^n = \rho^{n+1}\hat{\mathbf{V}}^{n+1} - \rho^n\hat{\mathbf{V}}^n = \rho^{n+1}\delta\hat{\mathbf{V}}^n + \hat{\mathbf{V}}^n\delta\rho^n \tag{3.11}$$

where $\hat{\mathbf{V}} = (c_1, \ldots, c_{ns})^T$, and $c_s = \rho_s/\rho$ 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.[8], the final result takes a similar form:

$$\hat{F}_{\rho_s} = c_s F'_\rho + (c_s^L - \tilde{c}_s)\rho^L\lambda^+ + (c_s^R - \tilde{c}_s)\rho^R\lambda^- \tag{3.12}$$

where $F'_\rho$ is the total mass flux computed previously using all $\mathbf{U}'$ variables, and $\tilde{}$ denotes a Roe-averaged quantity. Likewise, linearizing the species mass fluxes with respect to the $\hat{\mathbf{V}}$ variables yields

$$\hat{\mathbf{F}}^{n+1} = \hat{\mathbf{F}}^n + \frac{\partial \hat{\mathbf{F}}}{\partial \hat{\mathbf{V}}^L}\delta\hat{\mathbf{V}}^L + \frac{\partial \hat{\mathbf{F}}}{\partial \hat{\mathbf{V}}^R}\delta\hat{\mathbf{V}}^R \tag{3.13}$$

$$\frac{\partial \hat{\mathbf{F}}}{\partial \hat{\mathbf{V}}^L} = \tilde{w}F_\rho + (1-w)\rho^L\lambda^+ - \tilde{w}\rho^R\lambda^- \tag{3.14}$$

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

A full derivation of Eq.s (3.12-3.15), along with the definition of $\tilde{w}$, is included in Appendix A. 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:

$$\hat{\mathbf{W}}^{n+1} = \hat{\mathbf{W}}^n + \left.\frac{\partial \hat{\mathbf{W}}}{\partial \mathbf{U}}\right|_{\mathbf{U}'} \frac{\partial \mathbf{U}}{\partial \hat{\mathbf{V}}} \tag{3.16}$$

For simplicity of notation, we define

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

The decoupled system to be solved becomes:

$$\begin{aligned}\rho^{n+1}\frac{\delta\hat{\mathbf{V}}^n}{\Delta t}+\frac{1}{V}\sum_f(\frac{\partial\hat{\mathbf{F}}^f}{\partial\hat{\mathbf{V}}^L}\delta\hat{\mathbf{V}}^L+\frac{\partial\hat{\mathbf{F}}^f}{\partial\hat{\mathbf{V}}^R}\delta\hat{\mathbf{V}}^R)^{n,n+1}\mathbf{N}^f-C^{n,n+1}\delta\mathbf{V}^n \\ =-\frac{1}{V}\sum_f(\hat{\mathbf{F}}^{n,n+1}\cdot\mathbf{N})^f+\mathbf{W}^{n,n+1}-\hat{\mathbf{V}}^n\frac{\delta\rho^n}{\Delta t}-R_\rho\end{aligned} \tag{3.18}$$

$$R_\rho=-\frac{1}{V}\sum_f\sum_s(\hat{F}^{n,n+1}_{\rho_s}\cdot\mathbf{N}) \tag{3.19}$$

where $R_\rho$ is included to preserve the constraint that the mass fractions sum to unity, i.e., $\sum_s c_s = 1$, $\sum_s \delta c_s = 0$.

## 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[20]. 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 $5 \times 5$ 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 $ns \times ns$ matrices. Via Eq. 3.7, we decompose $A_d$ into its diagonal and

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

$$\begin{pmatrix} \square & & & \\ & \ddots & & \\ & & \square & \\ & & & \ddots \\ & & & & \square \end{pmatrix} \begin{pmatrix} \delta\hat{\mathbf{V}}_1 \\ \vdots \\ \delta\hat{\mathbf{V}}_i \\ \vdots \\ \delta\hat{\mathbf{V}}_{nodes} \end{pmatrix} = \begin{pmatrix} \hat{b}_1 \\ \vdots \\ \hat{b}_i \\ \vdots \\ \hat{b}_{nodes} \end{pmatrix} - \begin{pmatrix} (\sum_{j=1}^{N_{nb}} [\diagdown]\delta\hat{\mathbf{V}}_j)_1 \\ \vdots \\ (\sum_{j=1}^{N_{nb}} [\diagdown]\delta\hat{\mathbf{V}}_j)_i \\ \vdots \\ (\sum_{j=1}^{N_{nb}} [\diagdown]\delta\hat{\mathbf{V}}_j)_{nodes} \end{pmatrix} \tag{3.20}$$

where $\square$ represents a dense $ns \times ns$ 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 quadratic operation left in solving the implicit system is dealing with the diagonal entries in the Jacobian. Because the off-diagonal entries significantly outnumber the diagonal entries, we can expect nearly linear scaling in cost with the number of species. If compressed row storage[13] is used to only store non-zero off-diagonal entries, the relative memory savings in the limit of a large number of species for the Jacobian is given by

$$\begin{aligned} \textit{Relative Memory Cost} &= \frac{size(A_d)}{size(A)} \\ &= \lim_{ns\to\infty} \frac{(ns^2+5^2)(N_{nodes}) + (ns+5^2)(N_{nz})}{(ns+4)^2(N_{nodes}+N_{nz})} \\ &= \frac{N_{nodes}}{N_{nodes}+N_{nz}} \end{aligned} \tag{3.21}$$

where $N_{nodes}$ is the number of nodes, and $N_{nz}$ is the number of non-zero off-diagonal entries stored using compressed row storage. For a structured grid, each node has six

neighbors in 3D, 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.

# 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 goal of this research is to compute sensitivities of aerodynamic and aerothermodynamic quantities to design variables. To achieve this, the sensitivity to the primal flow equation formulation must first be solved. For a discrete adjoint formulation, this requires a solution of costate variables, relating a change in the flow equation residual to a change in the function of interest. Because of the large number of equations required in a reacting gas solver, the adjoint solver will suffer from the quadratic scaling in computational cost and memory required similarly to the primal flow solver. To mitigate this, 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. 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 (also known as costate variables), $\mathbf{\Lambda}$ are the cost function dependence on the residual

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

This 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 in 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 can be done using **SNOPT[16]**, **KSOPT[22]**, or **NPSOL[15]** in FUN3D, as well as a host of other non-linear optimizers.

## 4.2 Block Jacobi Adjoint Decoupling

It is possible to decoupled 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 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

$$\mathbf{U}' = \begin{pmatrix} \rho \\ \rho\mathbf{u} \\ \rho E \end{pmatrix}, \qquad \hat{\mathbf{U}} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{ns} \end{pmatrix} \tag{4.7}$$

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

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

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

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

Examining Eq.s (4.8-4.9) shows that there are clearly some physical dependencies being omitted, namely $\frac{\partial \mathbf{R}_{\rho_i}}{\partial \rho}$, $\frac{\partial \mathbf{R}_{\rho}}{\partial c_j}$, $\frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial c_j}$, and $\frac{\partial \mathbf{R}_{\rho E}}{\partial c_j}$. It has been found[8] that omitting these dependencies does not hinder convergence the primal flow solver; however, because the adjoint requires an exact linearization of the converged steady-state solution, these must be accounted for in the decoupled adjoint formulation.

The next step is to reconcile the split conserved variables, $\mathbf{U}'$ and $\hat{\mathbf{U}}$, 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

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

details and the derivation of the transformation matricies need to convert between these two systems are included in section A.4. From this perspective, the decoupled scheme

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

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

The iterative mechanism used in the solver makes the following approximations to the jacobians

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

which results in a significantly more sparse 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 signifcant 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 [34]. Nielsen [27] 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

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

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

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

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 $\mathbf{A}_{\mathbf{U}}$ it the jacobian matrix for the fully coupled system, $\frac{\partial\mathbf{R}_{\mathbf{U}}}{\partial\mathbf{U}}$. A full derivation of the transformation

of equations is also included in section A.4. 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 tranpose

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

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

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

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

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

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

$$\Delta\Lambda_\mathbf{U} = \frac{\partial \mathbf{R_U}}{\partial \mathbf{R_V}} \Delta\Lambda_\mathbf{V} \tag{4.25}$$

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 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.

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. Thus, the 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 is also more efficient than the flow solver iterative mechanism, which requires two residual vectors to be computed per timestep.

## 4.3 Higher-order Reconstruction Linearizations

To achieve higher-order accuracy, the FUN3D solver uses an extension of the unstructured MUSCL (U-MUSCL) reconstruction scheme developed by Burg et al[6, 7], which is itself and extension of the Monotonic Upstream-centered Scheme for Conservation Laws (MUSCL) scheme developed by Van Leer[38]. The implementation is a combination of

central differencing and the original U-MUSCL scheme

$$\begin{aligned} q_L &= q_1 + (1-\kappa)\left[\phi\left(\frac{\partial q_1}{\partial x}dx + \frac{\partial q_1}{\partial y}dy + \frac{\partial q_1}{\partial z}dz\right)\right] + \frac{\kappa}{2}(q_2 - q_1) \\ q_R &= q_2 + (1-\kappa)\left[\phi\left(\frac{\partial q_2}{\partial x}dx + \frac{\partial q_2}{\partial y}dy + \frac{\partial q_2}{\partial z}dz\right)\right] + \frac{\kappa}{2}(q_1 - q_2) \end{aligned} \tag{4.26}$$

Figure 4.1 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 variable $\phi$ is the result of the modified Van Albada flux limiter function[37], and

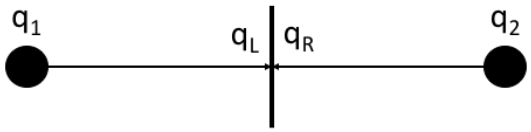



Figure 4.1: Edge Reconstruction

varies between 0 and 1 (effectively blending first-order and second order contributions to the flux evaluation). 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. Finally the gradient information $\frac{\partial q_{1,2}}{\partial x}$, $\frac{\partial q_{1,2}}{\partial y}$, and $\frac{\partial q_{1,2}}{\partial z}$ are computed using least-squares. For the example 2-D stencil shown in Figure 4.2 this is effectively computed as

$$\begin{aligned} \frac{\partial q}{\partial x} &= \sum_{i=1}^{5} W_{x,i}(q_i - q_0) \\ \frac{\partial q}{\partial y} &= \sum_{i=1}^{5} W_{y,i}(q_i - q_0) \\ \frac{\partial q}{\partial z} &= \sum_{i=1}^{5} W_{z,i}(q_i - q_0) \end{aligned} \tag{4.27}$$

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

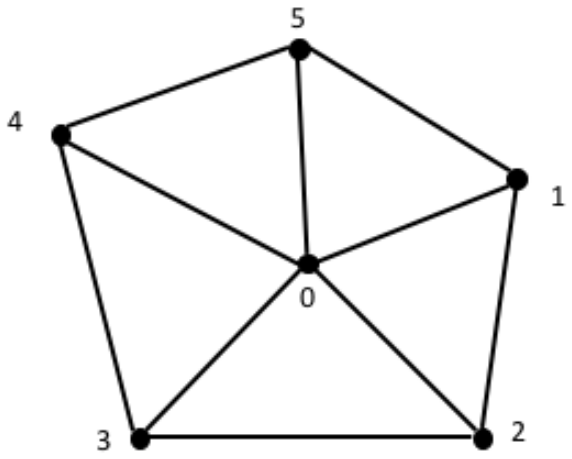



Figure 4.2: Example Stencil for Least-Squares Gradient Evaluation

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, $R$, is then evaluated in two parts

$$\frac{\partial \mathbf{R}\left(\mathbf{Q}^{*}\left(\mathbf{Q}\right)\right)}{\partial \mathbf{Q}} = \frac{\partial \mathbf{R}}{\partial \mathbf{Q}} + \frac{\partial \mathbf{R}}{\partial \mathbf{Q}^{*}} \frac{\partial \mathbf{Q}^{*}}{\partial \mathbf{Q}} \tag{4.28}$$

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 Jacobian of the Roe FDS scheme, $\frac{\partial \mathbf{R}}{\partial \mathbf{Q}}$, the higher order linearization is 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 4.2, the contribution to the residual from

looping around node 0 would be

$$\frac{\partial \mathbf{R}_0}{\partial \mathbf{Q}^*}\frac{\partial \mathbf{Q}^*}{\partial \mathbf{Q}_0} = \frac{\partial \mathbf{R}_0}{\partial \mathbf{Q}^*}\left[\sum_{i=1}^{5}(1-\kappa)\left(-W_{x,i}dx - W_{y,i}dy - W_{z,i}dz\right)\frac{\partial q_0}{\partial \mathbf{Q}_0}\right] \tag{4.29}$$

An important point that this illustrates is that the linearization of the higher order terms is not dependent upon the choice of the primitive variable vector $q$, provided that all of the linearizations are exact. This is powerful, because it allows the decoupled scheme to be extended to higher work, without needing to reformulate the linearizations from the fully-coupled scheme.

## 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 on 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 important boon in these cases, since the memory freed by the sparsity on the LHS of Eq. 4.24 can be used by the RHS linearizations to aid 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 difficult physics. The problem must include a sufficient degree of chemistry to highlight the stability concerns of the decoupled flow solver presented in section 8.2. This chapter details the geometry and test conditions of 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[18] 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

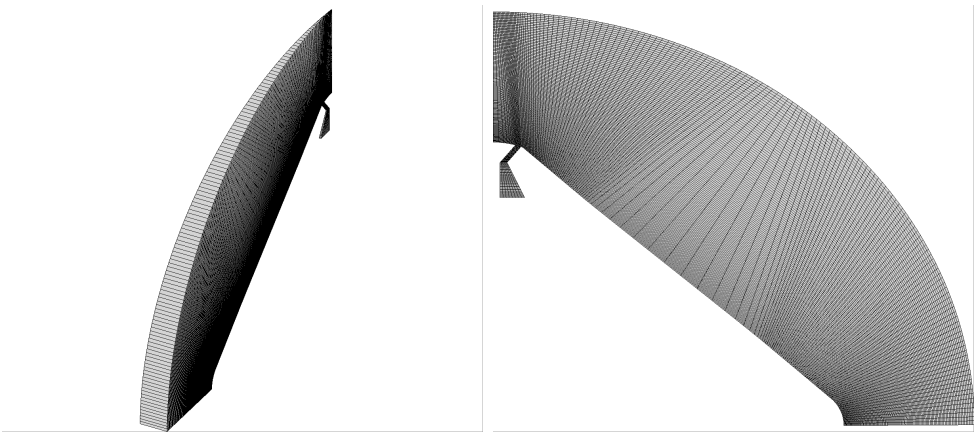


Figure 5.1: Annular Jet Geometry

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 of hexahedra elements. The flow conditions are shown in Table 5.2

| 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 |
| $\theta_c$ | cone half angle, deg | 50.0 |

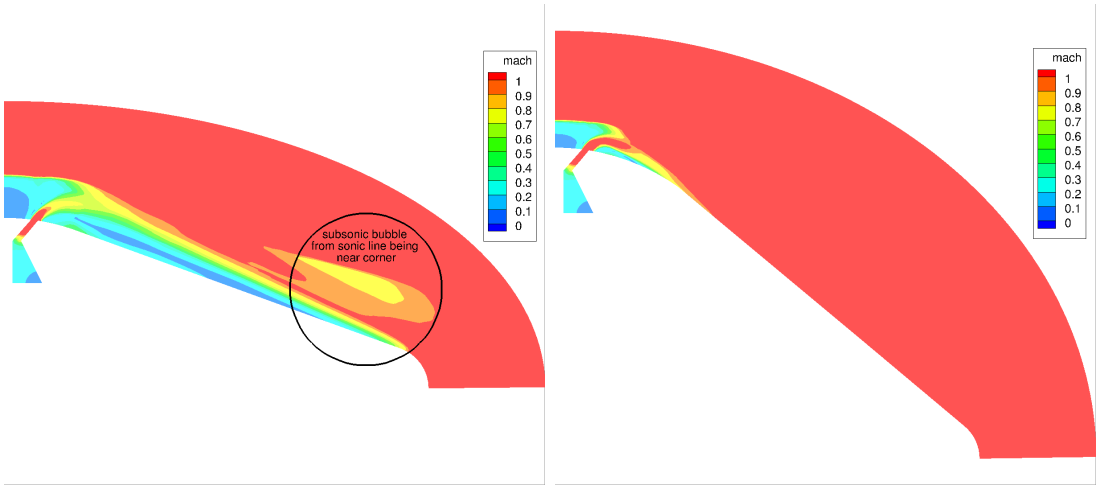
Table 5.1: Annular Nozzle Geometry Inputs

| Flow Condition | Description | Value |
|---|---|---|
| $V_\infty$ | freestream velocity, $m/s$ | 5686.24 |
| $\rho_\infty$ | freestream density, $kg/m^3$ | 0.001 |
| $T_\infty$ | freestream temperature, $K$ | 200.0 |
| $M_\infty$ | freestream Mach number (derived) | 20.0 |

Table 5.2: Flow Conditions

## 5.2 Steadiness Dependence on Cone Angle

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 limited in scope to steady flows only at this time. A full design optimization covers a wide variety of flow solutions, and it is therefore important to verify that the geometry chosen is truly steady. It was heuristically determined that blowing a light gas with a high cone angle leads to a sonic corner body. A similar phenomenon was examined by Gnoffo, Weilmuenster, Braun, and Cruz[19] for the Mars Pathfinder probe on descent into the Martian atmosphere. Figure 5.2 shows a comparison of blowing pure $H_2$ from the plenum on a vehicle with a $50^o$ cone angle and $70^o$ cone angle. Figure 5.2a highlights the subsonic bubble that is indicative of a sonic corner body, whereas Figure 5.2b shows that decreasing the cone angle moves the sonic line upstream and eliminates the subsonic bubble. The




(a) $70^o$ Cone angle blowing pure $H_2$ (b) $50^o$ Cone angle blowing pure $H_2$

Figure 5.2: Cone angle comparison

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[18]. The comparision 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[**?**] that uniformly divides each element in the mesh. Each the solution on each grid level was computed using the freestream conditions in Table 5.2 and the plenum conditions in Table 5.3, and Figure 5.3 shows the progression of refined meshes generated by this

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

Table 5.3: Plenum Conditions

method Figure 5.4 shows the grid convergence of surface temperature and plenum mass

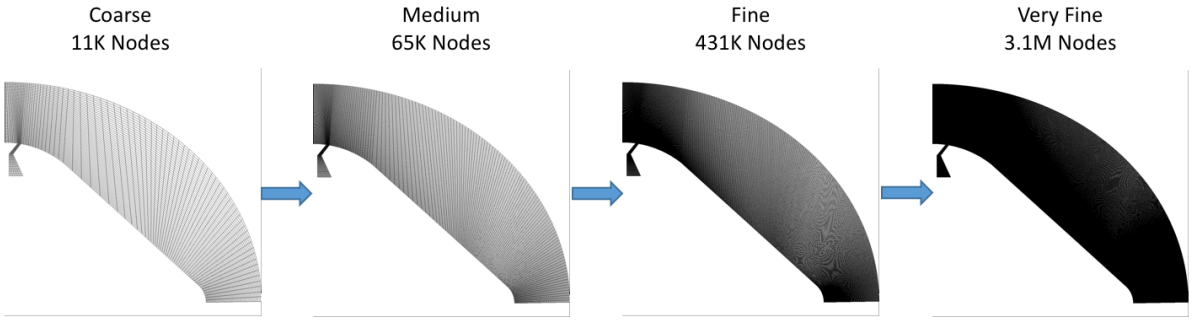



Figure 5.3: Uniformly Refined Meshes

flow rate for a nominal case. the fine mesh (431,000 Nodes) was found to be slightly unsteady, but coerced to be steady when the limiter was frozen. The very fine mesh was very unsteady, and freezing the flux limiter did not coerce the flow back to steady. 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 resolves to unsteadiness on a fine enough mesh. For the purposes of demonstrating the decoupled discrete adjoint, the discretized flow solution must remain steady; therefore, the coarse mesh (11,000 nodes) is used for the remainder of this study, with the caveat that the grid converged flow solution is truly an unsteady problem.

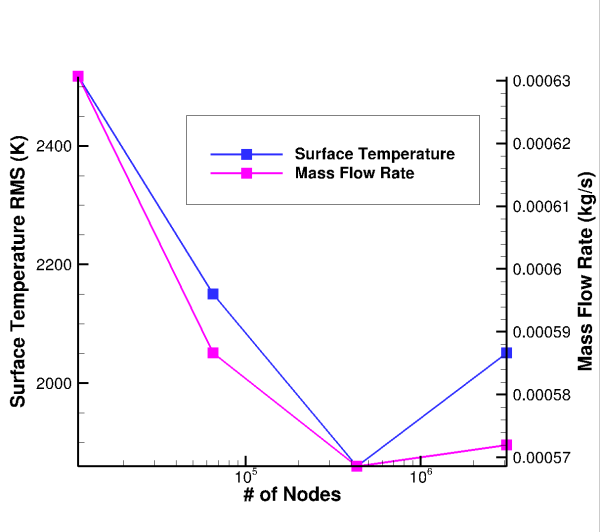



Figure 5.4: Grid convergence

## 5.4 Sensitivity to Frozen Flux Limiter

The use of a flux limiter is mandatory for 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[17] 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, the adjoint is required to be run with a "frozen" limiter, that is only where a reconstruction is unrealisable is the value of the limiter function changed. Doing this usually results in the residuals converging to machine precision.

A particular difficulty of this approach comes from mis-aligned meshes involving chem-

ical reactions. Because the annular jet plume and bow shock cannot remain aligned with the mesh during an 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. Consequently, species in will quickly deplete and form as the shock and plume move, causing the flux limiter as more nodes to require re-evalution. This process will eventually settle out to converge the flow solution to machine zero; however, the solution is highly dependent on the limiter field. Because of this sensitivity, integrated aerothermodynamic quantities results can vary between solution with the same inputs, but different starting states. This is a recognized problem, but one that is outside of the scope of this study. There are a number of recent approaches in the hypersonic community CITE???? that are attempting to mitigate this particular issue with grid alignment, 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[12], sequential quadratic programming (SQP)[14], as well as an interesting method that converts a constrained optimization problem into an unconstrained on by employing the Kreisselmeier-Steinhauser function[39]. 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 geometry and test conditions for the optimization, the implementation of the cost function components and design variables used in the optimization, as well as the interface to the optimizer that is used.

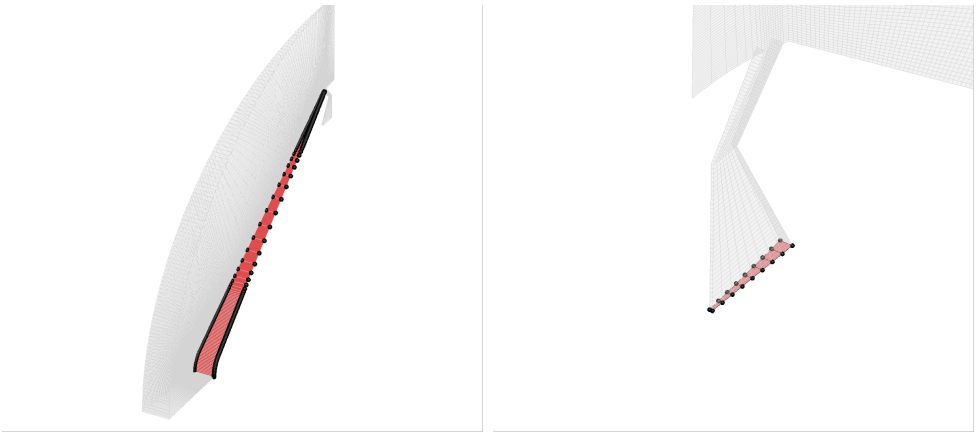
## 6.1 Integrated Quantities of Interest

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 [18], 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 massflow 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) $T_{RMS}$ Integrated Area (b) Mass Flow Rate Integrated Area

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", $\phi_p$. 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 $\phi_p$ 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

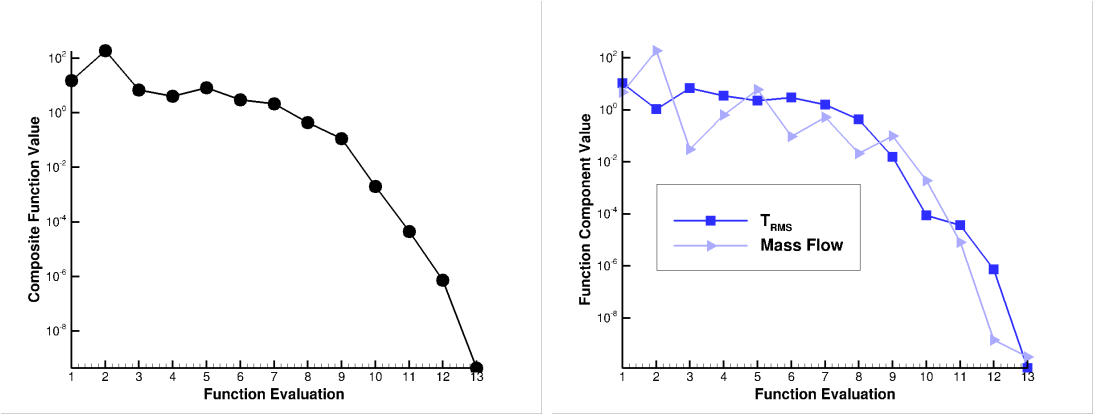
$$\frac{\partial L}{\partial \mathbf{D}} = \frac{\partial \mathbf{R}}{\partial \mathbf{D}}^T \mathbf{\Lambda} \tag{6.7}$$

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 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.

To demonstrate the inverse design capability of an adjoint-based design optimization,



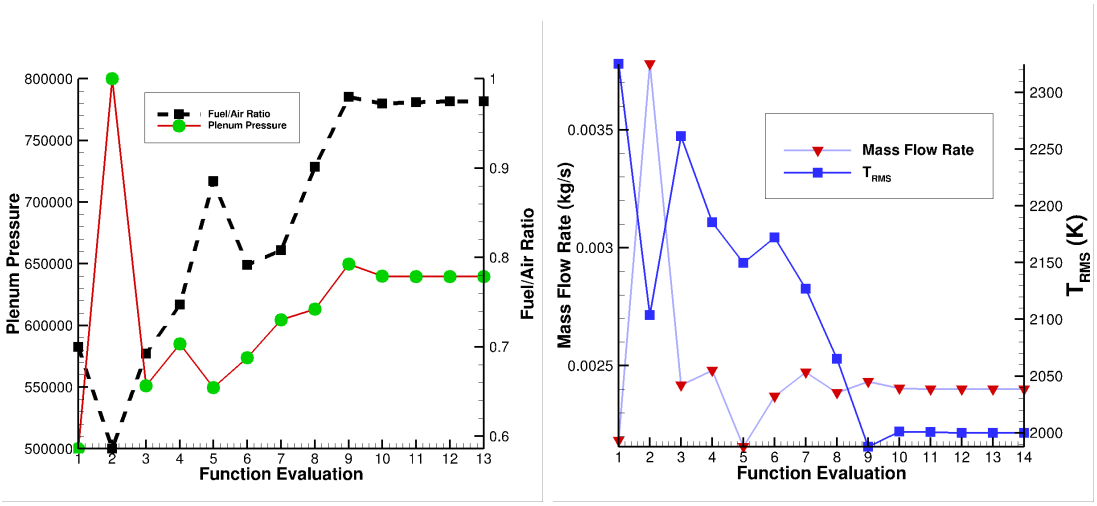

(a) Composite Value (b) Component Values

Figure 6.2: Cost Function and Component History

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-arbitarily, and were heuristically determined to be feasible based the 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", $\phi_p$. A species mixture consisting of $H_2$ and $N_2$ was blown from the plenum, with the mass fractions dictated by $\phi_p$ 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 beween the $T_{RMS}$ and $\dot{m}$, which is desired as both targets should be met at optimality. Using the SNOPT optimizer, Figure 6.2a 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




(a) Design Variable History (b) Design Variable History

Figure 6.3: Inverse Design History

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 Direct Design Optimization

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

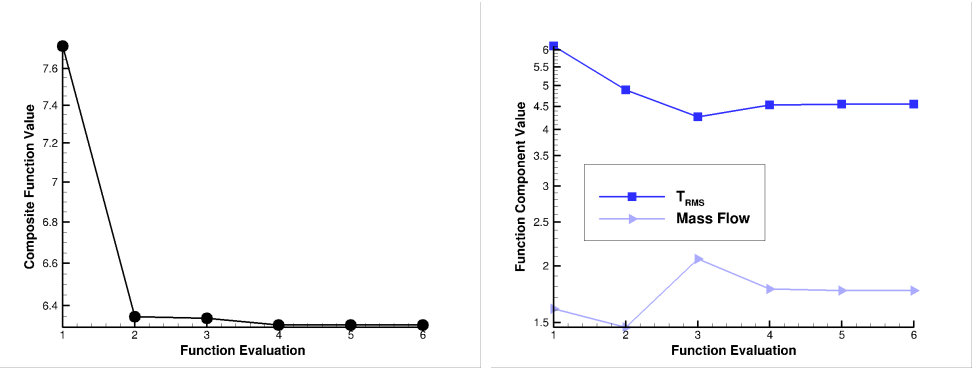
$$f = w_1 \left(\dot{m}_p\right)^2 + w_2 \left(T_{RMS}\right)^2 \tag{6.9}$$

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

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

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 two function evaluations, and the subsequent steps were significantly less. Figure 6.4b verifies that weights determined by 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 13.79% lower surface temperature, at the expense of a 4.6% higher mass flow rate. This is an excellent problem for a high-fidelity gradient-based optimization, as the non-linear effects of plenum-shock interation makes the optimum plenum condition difficult to determine intuitively.

There is a much stronger dependence on the choice of cost function weights for this



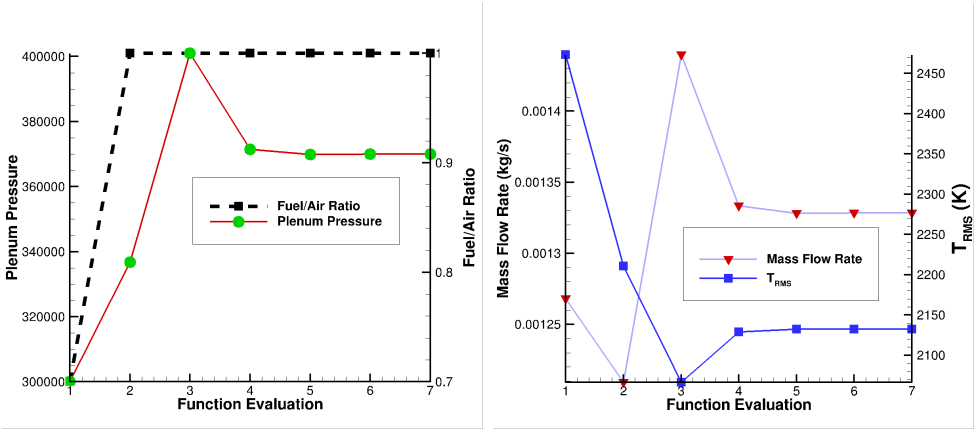

(a) Composite Cost Function Value (b) Cost Function Components

Figure 6.4: Direct Design Cost Function

| Component | Initial | Final | Improvement |
|---|---|---|---|
| $\dot{m}$, $kg/s$ | 1.268e-3 | 1.327e-3 | -4.6% |
| $T_{RMS}$, $K$ | 2473 | 2132 | 13.79% |

Table 6.1: Direct Design Optimization Improvement

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




(a) Design Variable History (b) $\dot{m}_p$ and $T_{RMS}$ Design History

Figure 6.5: Direct Design History

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[35] and evaluated by Newman et

al[26]. 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 identically, 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 a 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 |
| $\phi_p$ | -0.34045335117520E-01 | -0.340453351177196E-01 | 5.86e-12 |

Table 7.2: Sensitivity Derivative Comparison - $H_2$-$N_2$ Frozen

all the design variable senstivities computed by the adjoint and complex solvers, with all matching discretely to within 11 digits. Table 7.3 shows the same comparison as that 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 temperature dependence of the chemical source term, and the conditioning of the temperature jacobian matrices. The

| 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 |
| $\phi_p$ | -0.28035409045530E-01 | -0.280251731728184E-01 | 3.65e-04 |

Table 7.3: Sensitivity Derivative Comparison - $H_2$-$N_2$ Reacting

dissociation reactions near the stagnation region and the combustion reactions that occur when $H_2$ auto-ignites in the shock layer are sensitive to the temperature, and account for a significant numerical stiffness seen in the convergence of the conservation equation residuals. This manifests in the residuals stalling at a much higher value than the previous cases that were run without a chemical source term. Due to the non-dimensionalization in the generic gas path of FUN3D, the condition number of temperature jacobian scales quadratically with the freestream velocity. This is discussed thoroughly in Appendix A.3. Since this is a hypersonic problem, it can be expected that the conditioning of the temperature jacobian has impacted the adjoint and complex solve sensitivities.

# 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 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 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 thermodynamic 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.

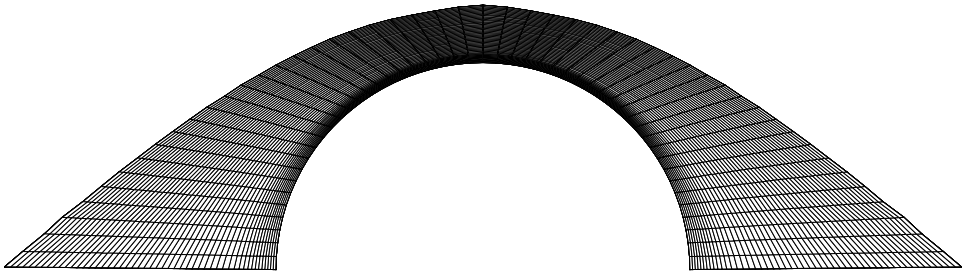
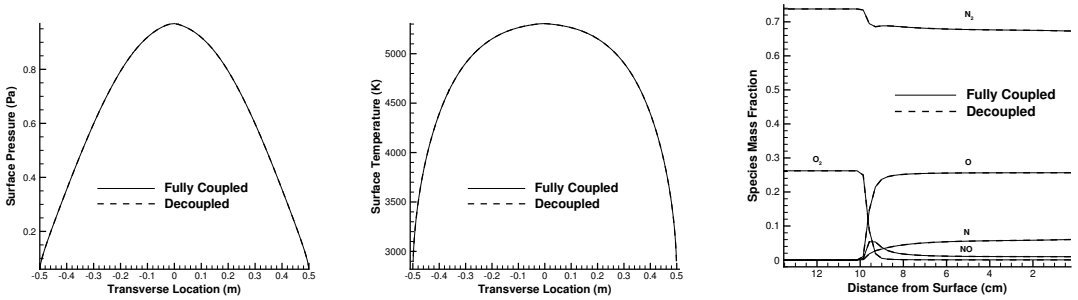


Figure 8.1: 50×50 cylinder grid.

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. 3.21, 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, and that the total speedup of the problem is less than

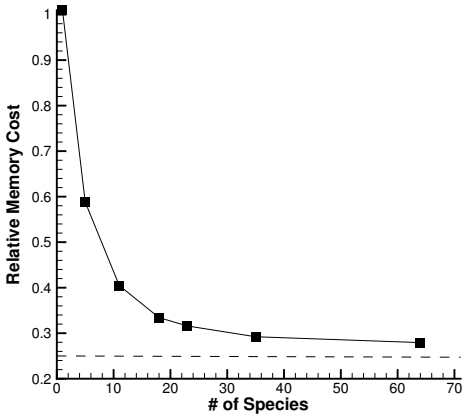



Figure 8.3: Memory required convergence study

that of just the linear solve. It is to be expected that the overall gains are not as large as those for the implicit solve, since there are many other factors that scale with the number of species, especially calculating the species source term and its linearization.

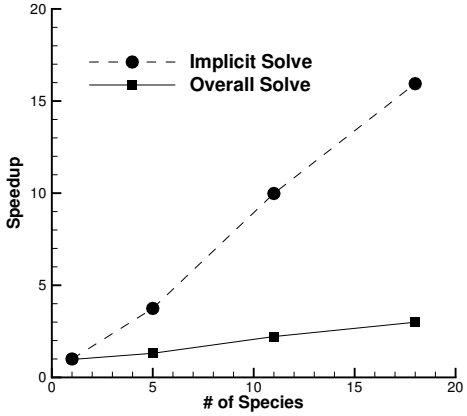


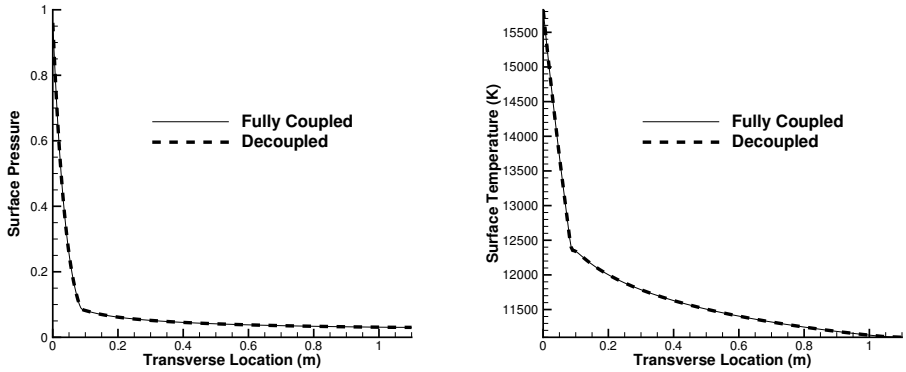

Figure 8.4: Relative speedup for the decoupled scheme vs. fully coupled scheme.

## 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. [8] (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 $V_\infty = 15000\ m/s$, $\rho_\infty = 0.001\ kg/m^3$, $T_\infty = 200\ K$. It was discovered that CFL limitations for the decoupled scheme were prohibitive, because of the stiffness of the chemical source term. In order to converge the scheme in a manner competitive with the fully coupled approach, it was necessary to scale the magnitude of the source term contribution to the flux balance by a value $\omega$, such that $0 \leq \omega \leq 1$. To ensure that the decoupled and fully coupled approaches yielded the same result, a ramping scheme was implemented such that no scaling was performed on the source term when the solution was in a converged state.

### 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 8.5 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



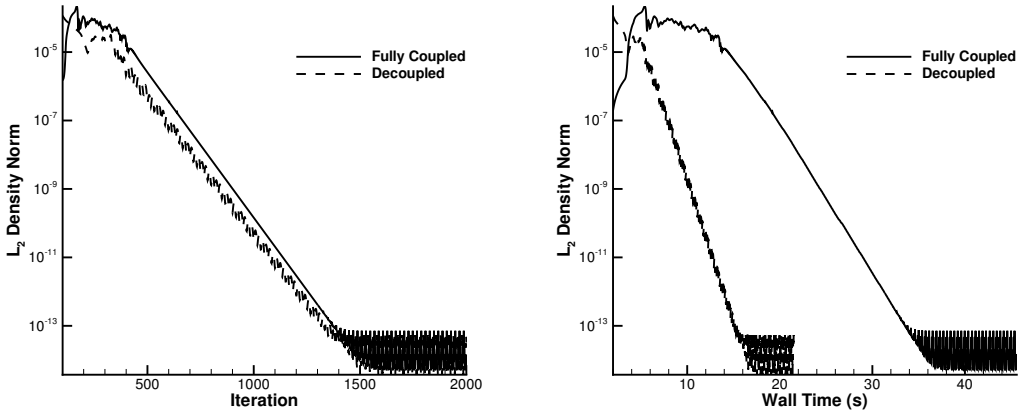

(a) Surface pressure (b) Surface temperature

Figure 8.5: Sphere-cone predicted quantities.

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 $\omega$ from 0.001 to 1.0 over the first 500 timesteps. Figure 8.6 shows that the convergence of both schemes progresses nearly identically, with the 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.




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

Figure 8.6: Sphere-cone convergence details.

### 8.2.3 Sphere Cone - Convergence Improvement with Exact Linearizations

An important benefit of implementing an adjoint solver in FUN3D comes the requirement of exact linearizations, because the linearizations needed to form the residual in the adjoint solver can be reused by the flow solver. Previously, the reacting gas path employed an jacobian that approximated the linearizations of the Roe FDS scheme[20] 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 [31] 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 these exact linearizations 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 8.7 shows that up to a five-species air mixture, the convergence

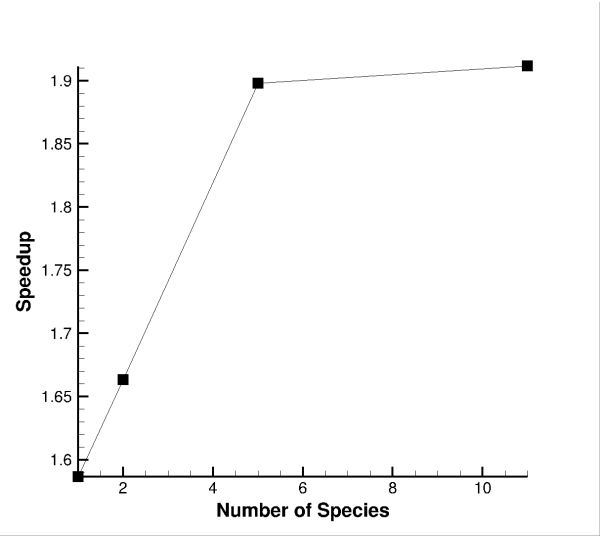



Figure 8.7: Relative Speedup Using Exact Linearizations

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 supercede the improved iterative convergence. It can be inferred from Figure 8.7 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 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

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, $\omega$, 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.

# 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, $\alpha$, is computed by both schemes and compared. Table 9.1

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

Table 9.1: Angle of Attack Sensitivity Relative Difference

shows that the solutions match to discretely to at least 11 digits, and is indicative 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-

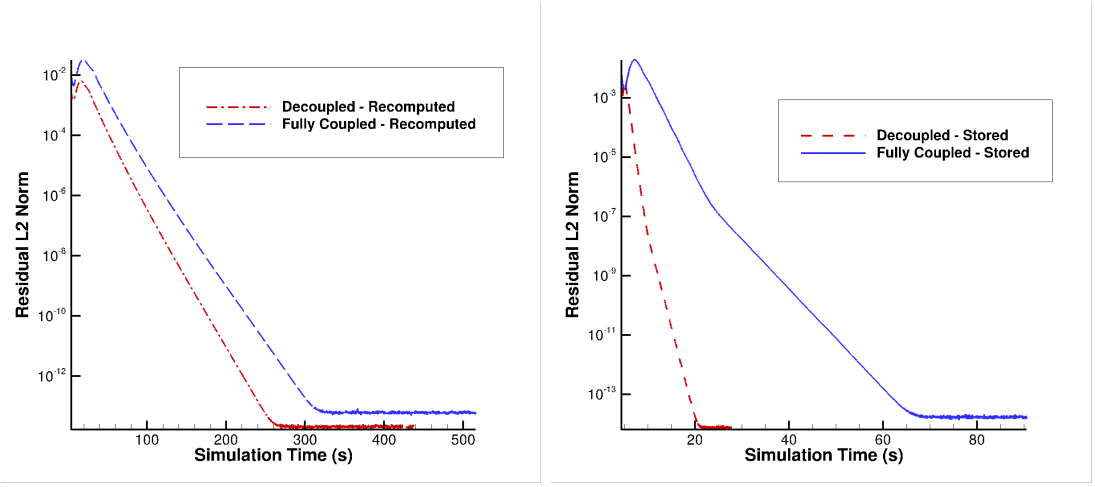
| Design Variable | Decoupled Sensitivity | Fully Coupled Sensitivity | Rel. Diff |
|---|---|---|---|
| $P_{p,o}$ | -0.11081315601976E-06 | -0.11081315601976E-06 | $1 \times 10^{-11}$ |
| $T_{p,o}$ | 0.19089941243495E-03 | 0.19089941237390E-03 | $2 \times 10^{-14}$ |
| $\phi_p$ | -0.28035409093945E-01 | -0.28035409045530E-01 | $2 \times 10^{-14}$ |

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.

## 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




(a) Linearizations Always Recomputed (b) Linearizations Stored

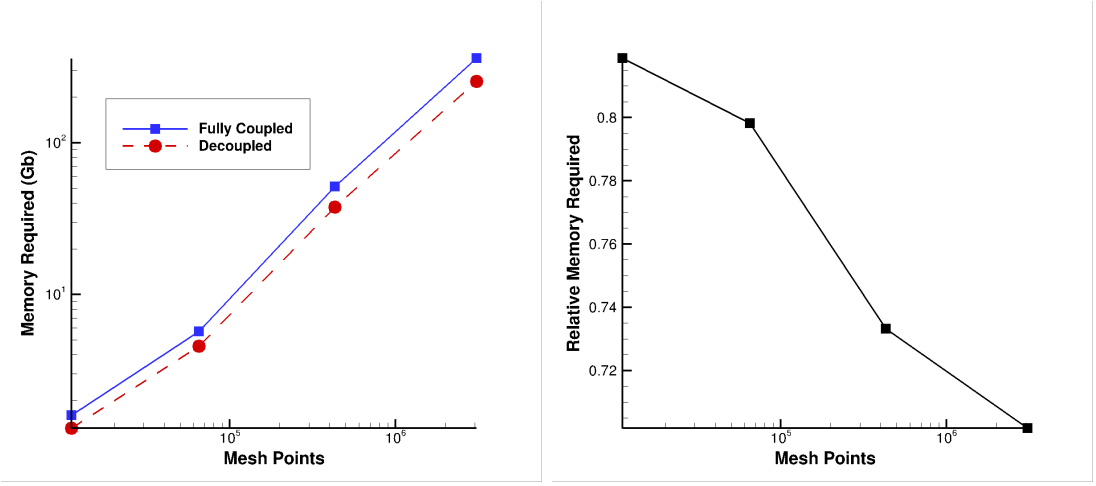
Figure 9.1: Simulation Time Comparison

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 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 hexahedral 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, 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




(a) Absolute Memory Required (b) Relative Memory Required

Figure 9.2: Memory Required to Store Full Linearizations

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 throught 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 accuract of the sensitivity dervatives 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 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 the 3 times more efficient than the fully coupled adjoint scheme for the annular jet demonstration problem, if the exact linearizations were store 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 utiltized 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] Martin A. Reno Bonnie J. McBride, Sanford Gordon. Coefficients for calculating thermodynamic and transport properties of individual species. Technical report, NASA, 1993.

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

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

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

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

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

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

[22] 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.

[23] 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.

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

[25] 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.

[26] 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.

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

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

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

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

[31] 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.

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

[33] Y. Saad. *Iterative Methods for Sparse Linear Systems*, pages 391–392. Society for Industrial and Applied Mathematics, 2003.

[34] 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.

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

[36] 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.

[37] 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.

[38] 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.

[39] 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:

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

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

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

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[21]. 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

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

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

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

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

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 Temperature Linearizations and Non-Dimensionalization

Following the thermodynamic state relations detailed by Gnoffo et. al. [20], the derivatives of temperature with respect to the conserved variable vector

$$\frac{\partial T}{\partial Q} = \begin{pmatrix} \frac{\partial T}{\partial \rho_1} \\ \vdots \\ \frac{\partial T}{\partial \rho_{ns}} \\ \frac{\partial T}{\partial \rho u} \\ \frac{\partial T}{\partial \rho v} \\ \frac{\partial T}{\partial \rho w} \\ \frac{\partial T}{\partial \rho E} \end{pmatrix} \tag{A.23}$$

The derivation of Eq. A.23 begins with the definition of the mixture internal energy, $e$

$$e = \sum_{s=1}^{N_{ns}} (c_s e_s) = \sum_{s=1}^{N_{ns}} \left[ c_s \left( \int_{T_{ref}}^{T} C_{v,s} dT + e_{s,o} \right) \right] \tag{A.24}$$

$$de = \sum_{s=1}^{N_{ns}} (dc_s e_s) + \sum_{s=1}^{N_{ns}} (c_s C_{v,s})\, dT \tag{A.25}$$

solving Eq. A.25 for $dT$

$$dT = \frac{de - \sum_{s=1}^{N_{ns}} (dc_s e_s)}{C_v} \tag{A.26}$$

where $C_v$ is the specific heat at constant volume of the mixture, defined as

$$C_v = \sum_{s=1}^{N_{ns}} (c_s C_{v,s}) \tag{A.27}$$

to transform Eq. A.26 into a relation with conserved variables, the definition of mixture total energy is re-arranged as

$$\rho E = \rho e + \frac{1}{2} \left( \rho u^2 + \rho v^2 + \rho w^2 \right) \tag{A.28}$$

$$e = \frac{\rho E - \frac{1}{2} \left( \rho u^2 + \rho v^2 + \rho w^2 \right)}{\rho} \tag{A.29}$$

$$de = \frac{-(e)\, d\rho - (u)\, d\rho u - (v)\, d\rho v - (w)\, d\rho w + d\rho E}{\rho} \tag{A.30}$$

substituting Eq. A.30 into Eq. A.26, Eq. A.23 can be rewritten as

$$\frac{\partial T}{\partial Q} = \frac{1}{\rho C_v} \begin{pmatrix} \frac{\left(u^2+v^2+w^2\right)}{2} - e_1 \\ \vdots \\ \frac{\left(u^2+v^2+w^2\right)}{2} - e_{ns} \\ -u \\ -v \\ -w \\ 1 \end{pmatrix} \tag{A.31}$$

The non-dimensionalization of variables in the generic gas path of FUN3D are based on freestream dimensional quantities

$$\begin{aligned} \rho &= \rho^{'}/\rho^{'}_{\infty} \\ u &= u^{'}/V^{'}_{\infty} \\ v &= v^{'}/V^{'}_{\infty} \\ w &= w^{'}/V^{'}_{\infty} \\ e &= e^{'}/\left(V^{'}_{\infty}\right)^2 \\ T &= T^{'} \end{aligned} \tag{A.32}$$

where $^{'}$ denotes a dimensional quantity. Note that temperature, $T$, is not nondimensionalized in FUN3D, due to the large number of table look-ups that have temperature in dimensional units of Kelvin. This is an important point when non-dimensionalizing the mixture specific heat at constant volume, $C_v$. In the MKS system, $C_v$ has units $J/kg \cdot K$;

therefore the proper nondimensionalization using the system in Eq. A.32 is

$$C_v = C_v{}' / \left(V_\infty'\right)^2 \tag{A.33}$$

Using Eq.s (A.32-A.33), Eq. A.31 can be rewritten in terms of dimensional quantities.

$$\frac{\partial T}{\partial Q} = \frac{\rho_\infty'}{\rho' C_v{}'} \begin{pmatrix} \frac{\left((u')^2+(v')^2+(w')^2\right)}{2} - e_1' \\ \vdots \\ \frac{\left((u')^2+(v')^2+(w')^2\right)}{2} - e_{ns}' \\ -u'(V_\infty') \\ -v'(V_\infty') \\ -w'(V_\infty') \\ (V_\infty')^2 \end{pmatrix} \tag{A.34}$$

As the reference velocity becomes large, as is the case in hypersonic problems, the quadratic scaling of the final linearization in Eq. A.34, $\frac{\partial T}{\partial \rho E}$ can lead to poor conditioning of the jacobian matrix. This is particular true for the chemical source term, since small changes in temperature can lead to very large changes in species densities.

## A.4 Change of Variable Sets

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

$$\mathbf{U} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{ns} \\ \rho\mathbf{u} \\ \rho E \end{pmatrix} \rightarrow \mathbf{V} = \begin{pmatrix} c_1 \\ \vdots \\ c_{ns} \\ \rho \\ \rho\mathbf{u} \\ \rho E \end{pmatrix} \tag{A.35}$$

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

$$\mathbf{U} = \begin{pmatrix} u_1 \\ \vdots \\ u_{ns+2} \end{pmatrix} \rightarrow \mathbf{V} = \begin{pmatrix} v_1 \\ \vdots \\ v_{ns+3} \end{pmatrix} \tag{A.36}$$

For simplicity, consider a system with two species, $\rho_1$ and $\rho_2$. 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

$$\mathbf{U} = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} v_1 v_3 \\ v_2 v_3 \\ v_4 \\ v_5 \end{pmatrix} \tag{A.37}$$

This allows the derivation of the jacobian

$$\frac{\partial \mathbf{U}}{\partial \mathbf{V}} = \begin{pmatrix} \frac{\partial u_1}{\partial v_1} & \frac{\partial u_1}{\partial v_2} & \frac{\partial u_1}{\partial v_3} & \frac{\partial u_1}{\partial v_4} & \frac{\partial u_1}{\partial v_5} \\ \frac{\partial u_2}{\partial v_1} & \frac{\partial u_2}{\partial v_2} & \frac{\partial u_2}{\partial v_3} & \frac{\partial u_2}{\partial v_4} & \frac{\partial u_2}{\partial v_5} \\ \frac{\partial u_3}{\partial v_1} & \frac{\partial u_3}{\partial v_2} & \frac{\partial u_3}{\partial v_3} & \frac{\partial u_3}{\partial v_4} & \frac{\partial u_3}{\partial v_5} \\ \frac{\partial u_4}{\partial v_1} & \frac{\partial u_4}{\partial v_2} & \frac{\partial u_4}{\partial v_3} & \frac{\partial u_4}{\partial v_4} & \frac{\partial u_4}{\partial v_5} \end{pmatrix} = \begin{pmatrix} v_3 & 0 & v_1 & 0 & 0 \\ 0 & v_3 & v_2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.38}$$

At this point, it is important to note that the jacobian in Eq. A.38 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

$$\left.\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\right|_R = \begin{pmatrix} \frac{\partial v_1}{\partial u_1} & \frac{\partial v_1}{\partial u_2} & \frac{\partial v_1}{\partial u_3} & \frac{\partial v_1}{\partial u_4} \\ \frac{\partial v_2}{\partial u_1} & \frac{\partial v_2}{\partial u_2} & \frac{\partial v_2}{\partial u_3} & \frac{\partial v_2}{\partial u_4} \\ \frac{\partial v_3}{\partial u_1} & \frac{\partial v_3}{\partial u_2} & \frac{\partial v_3}{\partial u_3} & \frac{\partial v_3}{\partial u_4} \\ \frac{\partial v_4}{\partial u_1} & \frac{\partial v_4}{\partial u_2} & \frac{\partial v_4}{\partial u_3} & \frac{\partial v_4}{\partial u_4} \\ \frac{\partial v_5}{\partial u_1} & \frac{\partial v_5}{\partial u_2} & \frac{\partial v_5}{\partial u_3} & \frac{\partial v_5}{\partial u_4} \end{pmatrix} = \begin{pmatrix} \frac{1-v_1}{v_3} & \frac{-v_1}{v_3} & 0 & 0 \\ \frac{-v_2}{v_3} & \frac{1-v_2}{v_3} & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.39}$$

It is easily verified that the matrix product of Eq.s (A.38-A.39) produces identity

$$\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\left.\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\right|_R = \begin{pmatrix} v_3 & 0 & v_1 & 0 & 0 \\ 0 & v_3 & v_2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{1-v_1}{v_3} & \frac{-v_1}{v_3} & 0 & 0 \\ \frac{-v_2}{v_3} & \frac{1-v_2}{v_3} & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.40}$$

however, Eq. A.40 is not associative

$$\left.\frac{\partial \mathbf{V}}{\partial \mathbf{U}}\right|_R \frac{\partial \mathbf{U}}{\partial \mathbf{V}} = \begin{pmatrix} \frac{1-v_1}{v_3} & \frac{-v_1}{v_3} & 0 & 0 \\ \frac{-v_2}{v_3} & \frac{1-v_2}{v_3} & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_3 & 0 & v_1 & 0 & 0 \\ 0 & v_3 & v_2 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.41}$$

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

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

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

$$= \begin{pmatrix} \frac{1-v_1}{v_3} & \frac{-v_2}{v_3} & 1 & 0 & 0 \\ \frac{-v_1}{v_3} & \frac{1-v_2}{v_3} & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} v_3 & 0 & 0 & 0 \\ 0 & v_3 & 0 & 0 \\ v_1 & v_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.42}$$

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

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

$$\frac{\partial \mathbf{R}}{\partial \mathbf{V}} = \frac{\partial \mathbf{R}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}} \tag{A.43}$$

which is intuitively understood; however, the transformation from the variable set $\mathbf{V}$ to the variable set $\mathbf{U}$ must follow Eq. A.42

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}} = \left( \frac{\partial \mathbf{R}}{\partial \mathbf{V}}^T \frac{\partial \mathbf{V}}{\partial \mathbf{U}}^T \right)^T \tag{A.44}$$

The transposition in Eq. A.43 is critical, as the linearizations will be incorrect if the multiplication is done without it. Fortunately, Eq. A.43 is rarely seen in practice, as most linearizations are done for the fully-coupled system that requires Eq. A.44 to transform the linearizations

$$\frac{\partial \mathbf{R}}{\partial \mathbf{V}} = \frac{\partial \mathbf{R}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}} = \begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho E} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix} \begin{pmatrix} \rho & \cdots & 0 & c_1 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \rho & c_{ns} & 0 & 0 \\ 0 & \cdots & 0 & 0 & 1 & 0 \\ 0 & \cdots & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.45}$$

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

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T = \frac{\partial \mathbf{U}}{\partial \mathbf{V}}^T \frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T = \begin{pmatrix} \rho & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & \rho & 0 & 0 \\ c_1 & \dots & c_{ns} & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_1} & \dots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_1} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_1} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_1} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_{ns}} & \dots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_{ns}} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho \mathbf{u}} & \dots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho E} & \dots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix} \tag{A.46}$$

Since the tranformation is right-multiplied, the matrix vector products of the exact jacobian with costate variables, $\Lambda$, in the adjoint linear system can be done first, and the

transformation can then be applied to the system

$$\frac{\partial \mathbf{R}}{\partial \mathbf{U}}^T \Lambda = \begin{pmatrix} \rho & \dots & 0 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & \dots & \rho & 0 & 0 \\ c_1 & \dots & c_{ns} & 0 & 0 \\ 0 & \dots & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_1}\Lambda_{\rho_1} & \dots & + & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_1}\Lambda_{\rho_{ns}} & + & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_1}\Lambda_{\rho \mathbf{u}} & + & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_1}\Lambda_{\rho E} \\ \vdots & \ddots & & \vdots & & \vdots & & \vdots \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_{ns}}\Lambda_{\rho_1} & \dots & + & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_{ns}}\Lambda_{\rho_{ns}} & + & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_{ns}}\Lambda_{\rho \mathbf{u}} & + & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_{ns}}\Lambda_{\rho E} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho \mathbf{u}}\Lambda_{\rho_1} & \dots & + & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho \mathbf{u}}\Lambda_{\rho_{ns}} & + & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}}\Lambda_{\rho \mathbf{u}} & + & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}}\Lambda_{\rho E} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho E}\Lambda_{\rho_1} & \dots & + & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho E}\Lambda_{\rho_{ns}} & + & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E}\Lambda_{\rho \mathbf{u}} & + & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E}\Lambda_{\rho E} \end{pmatrix} \tag{A.47}$$

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 $\mathbf{U}$.

In the decoupled scheme the number of equations effectively solved is one more than the fully-coupled scheme. The residual vector, $R$ for the decoupled scheme can be written

as

$$\mathbf{R} = \begin{pmatrix} \mathbf{R}_{\rho_1} - c_1 \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \vdots \\ \mathbf{R}_{\rho_{N_s}} - c_{N_s} \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \mathbf{R}_{\rho \mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \tag{A.48}$$

The residual vector in Eq. A.48 is composed entirely of components from the fully coupled system; there for the linearizations for the fully-coupled system can be re-used to construct the decoupled adjoint residual

## A.5 Relationship to Adjoint Equation

The flow solver equations can be constructed by the integration of the governing equations. In semi-discrete form, this is

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{1}{V} \sum_{i}^{N_{nodes}} (\mathbf{F}_i \cdot \mathbf{n}_i) = \mathbf{W} \tag{A.49}$$

where

$$\mathbf{F} = \begin{pmatrix} \rho_1 \mathbf{u} \\ \vdots \\ \rho_{N_s} \\ \rho \mathbf{u}^2 + p\mathbf{n} \\ (E+p)\mathbf{u} \end{pmatrix} \tag{A.50}$$

the next time level $n+1$ can be determined from the current time level $n$ if the equations are linearized by the approximations

$$\begin{aligned} \mathbf{F}^{n+1} &\approx \mathbf{F}^n + \frac{\partial \mathbf{F}^n}{\partial \mathbf{U}} d\mathbf{U}^n \\ \mathbf{W}^{n+1} &\approx \mathbf{W}^n + \frac{\partial \mathbf{W}^n}{\partial \mathbf{U}} d\mathbf{U}^n \end{aligned} \tag{A.51}$$

This creates the linear system of equations which may be solved by a quasi-Newton method

$$\left[ \frac{V}{\Delta t}\mathbf{I} + \begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho E} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix} \right] \begin{pmatrix} d\rho_1 \\ \vdots \\ d\rho_{N_s} \\ d\rho \mathbf{u} \\ d\rho E \end{pmatrix} = \begin{pmatrix} \mathbf{R}_{\rho_1} \\ \vdots \\ \mathbf{R}_{\rho_{N_s}} \\ \mathbf{R}_{\rho \mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \tag{A.52}$$

assuming all linearizations are exact, the adjoint system of equations that results from the fully coupled formulation in Eq. A.52 is

$$\begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_1} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_1} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_1} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_{ns}} & \cdots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_{ns}} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho \mathbf{u}} & \cdots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} \\ \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho E} & \cdots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix} \begin{pmatrix} \Lambda_{\rho_1} \\ \vdots \\ \Lambda_{\rho_{N_s}} \\ \Lambda_{\rho \mathbf{u}} \\ \Lambda_{\rho E} \end{pmatrix} = - \begin{pmatrix} \frac{\partial f}{\partial \rho_1} \\ \vdots \\ \frac{\partial f}{\partial \rho_{N_s}} \\ \frac{\partial f}{\partial \rho \mathbf{u}} \\ \frac{\partial f}{\partial \rho E} \end{pmatrix} \tag{A.53}$$

In the new variable set, $\mathbf{V}$, Eq. A.52 is re-written as

$$\left[ \frac{V}{\Delta t}\mathbf{I} + \begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho_1}}{\partial \rho E} \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho_{ns}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} \\ \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_1} & \cdots & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_{ns}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix} \right] \begin{pmatrix} dc_1 \\ \vdots \\ dc_{N_s} \\ d\rho \\ d\rho \mathbf{u} \\ d\rho E \end{pmatrix} = \begin{pmatrix} \mathbf{R}'_{\rho_1} \\ \vdots \\ \mathbf{R}'_{\rho_{N_s}} \\ \mathbf{R}_{\rho} \\ \mathbf{R}_{\rho \mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \tag{A.54}$$

Where the new species density equations $\mathbf{R}'_{\rho_s}$ and mixture density equation $\mathbf{R}_\rho$ are defined as

$$\mathbf{R}'_{\rho_s} = \mathbf{R}_{\rho_s} - c_s \mathbf{R}_\rho \tag{A.55}$$

$$\mathbf{R}_\rho = \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \tag{A.56}$$

Eq.s (A.55-A.56) are critical to the adjoint formulation of Eq. A.54, as primal flow equations have been altered to enforce the constraint that

$$\sum_{i=1}^{N_s} (c_i) = 1, \quad \sum_{i=1}^{N_s} (dc_i) = 0, \tag{A.57}$$

Just as relationships were derived for the variables set $\mathbf{U}$ and $\mathbf{V}$ in section A.4, there are also relationships between the equations, which we denote as $\mathbf{R_U}$ and $\mathbf{R_V}$ for the variable sets $\mathbf{U}$ and $\mathbf{V}$, respectively. the equation set $\mathbf{R_U}$ can be rewritten in terms of

the equation set $\mathbf{R_V}$, to form the jacobian

$$\mathbf{R_U} = \begin{pmatrix} \mathbf{R_V}_i + c_i \left( \mathbf{R_V}_{N_s+1} \right) \\ \mathbf{R_V}_{N_s+2} \\ \mathbf{R_V}_{N_s+3} \end{pmatrix} \tag{A.58}$$

$$\frac{\partial \mathbf{R_U}}{\partial \mathbf{R_V}} = \begin{pmatrix} 1 & 0 & c_i & 0 & 0 \\ 0 & 1 & c_i & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.59}$$

Likewise, the tranformation can be made from $\mathbf{R_V}$ to $\mathbf{R_U}$

$$\mathbf{R_V} = \begin{pmatrix} \mathbf{R}_{\mathbf{U}_i} + c_i \sum_{k=1}^{N_s} (\mathbf{R}_{\mathbf{U}_k}) \\ \sum_{k=1}^{N_s} (\mathbf{R}_{\mathbf{U}_k}) \\ \mathbf{R}_{\mathbf{U}_{N_s+1}} \\ \mathbf{R}_{\mathbf{U}_{N_s+2}} \end{pmatrix} \tag{A.60}$$

$$\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}} = \begin{pmatrix} 1-c_i & -c_i & 0 & 0 \\ -c_i & 1-c_i & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{A.61}$$

The change of equations sets in Eq.s (A.58-A.59) and the change of dependent variable in Eq. A.37 can be used to rewrite the fully coupled system in Eq. A.52 into the decoupled system from Eq. A.54

Based on Eq. 4.23, 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

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.

The exact same preconditioners used by the flow solver can be transposed and used to solve the linear system of equations in the adjoint. This is done in two steps and in reverse order of the iterative mechanism used in the flow solver. First, the adjoint costate variables associated with the species mass equations, $\Lambda_{\rho_s}$, are solved for

$$\left(\frac{V}{\Delta t}\mathbf{I} + \mathbf{A}_d\right) d\Lambda_{\rho_s} = -\left(\sum_{i=1}^{N_s} \frac{\partial \mathbf{R}'_{\rho_i}}{\partial c_s}\Lambda_{\rho_i} + \frac{\partial \mathbf{R}_{\rho}}{\partial c_s}\Lambda_{\rho} + \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial c_s}\Lambda_{\rho\mathbf{u}} + \frac{\partial \mathbf{R}_{\rho E}}{\partial c_s}\Lambda_{\rho E} + \frac{\partial f}{\partial c_s}\right) \tag{A.62}$$

Followed by the adjoint costate variables associated with the mixture equations, $\Lambda_{\rho}$, $\Lambda_{\rho\mathbf{u}}$, and $\Lambda_{\rho E}$

$$\left(\frac{V}{\Delta t}\mathbf{I} + \mathbf{A}_m\right) \begin{pmatrix} d\Lambda_{\rho} \\ d\Lambda_{\rho\mathbf{u}} \\ d\Lambda_{\rho E} \end{pmatrix} = -\begin{pmatrix} \sum_{i=1}^{N_s} \frac{\partial \mathbf{R}'_{\rho_i}}{\partial \rho}\Lambda_{\rho_i} + \frac{\partial \mathbf{R}_{\rho}}{\partial \rho}\Lambda_{\rho} + \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial \rho}\Lambda_{\rho\mathbf{u}} + \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho}\Lambda_{\rho E} + \frac{\partial f}{\partial \rho} \\ \sum_{i=1}^{N_s} \frac{\partial \mathbf{R}'_{\rho_i}}{\partial \rho\mathbf{u}}\Lambda_{\rho_i} + \frac{\partial \mathbf{R}_{\rho}}{\partial \rho\mathbf{u}}\Lambda_{\rho} + \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial \rho\mathbf{u}}\Lambda_{\rho\mathbf{u}} + \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho\mathbf{u}}\Lambda_{\rho E} + \frac{\partial f}{\partial \rho\mathbf{u}} \\ \sum_{i=1}^{N_s} \frac{\partial \mathbf{R}'_{\rho_i}}{\partial \rho E}\Lambda_{\rho_i} + \frac{\partial \mathbf{R}_{\rho}}{\partial \rho E}\Lambda_{\rho} + \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial \rho E}\Lambda_{\rho\mathbf{u}} + \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E}\Lambda_{\rho E} + \frac{\partial f}{\partial \rho E} \end{pmatrix} \tag{A.63}$$