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

Kyle B. Thompson

Mechanical and Aerospace Engineering Department
North Carolina State University

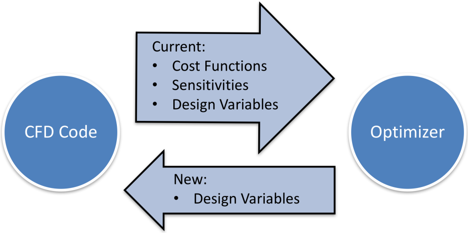
Aerothermodynamics Branch
NASA Langley Research Center

Month Day, 2017

Outline

Introduction - Design

- Gradient-based design optimization is based on the minimization of a target "cost" function by changing a set of design variables
- A CFD code can be coupled with a numerical optimization package to iteratively improve target aerothermodynamic quantities, by change inputs to the CFD code



CFD-Optimizer Relationship

Introduction - Design

- The top-level design process is simple, but CFD sensitivity analysis is expensive

Introduction - Design

- The top-level design process is simple, but CFD sensitivity analysis is expensive
- Need efficient way to compute cost function sensitivities for large number of design variables

Introduction - Design

- The top-level design process is simple, but CFD sensitivity analysis is expensive
- Need efficient way to compute cost function sensitivities for large number of design variables

Direct differentiation approach - Expensive

- Navier-Stokes equations can be directly differentiated to yield sensitivity derivatives necessary for gradient-based optimization
- Finite difference requires a minimum of **one flow solution for each design variable sensitivity**
- Prohibitively expensive for large number of design variables

Introduction - Design

- The top-level design process is simple, but CFD sensitivity analysis is expensive
- Need efficient way to compute cost function sensitivities for large number of design variables

Adjoint approach - More efficient

- Solve adjoint equations in addition to Navier Stokes flow equations to obtain sensitivity derivatives
- **One flow and adjoint solution needed for each cost function**, regardless of number of design variables
- Considerably more efficient than direct differentiation approach for large number of design variables

Introduction - Design

- Adjoint-based design optimization is widely adopted in compressible, perfect gas CFD solvers
- Reacting flow solvers have lagged in adopting adjoint-based approach, due to
  1. Complexity of linearizing the additional equations for multi-species chemical kinetics
  2. Resorting to Automatic Differentiation tools incurs performance overhead that is implementation-specific
  3. Serious memory and computational cost concerns when simulating a large number of species
- Points 1 and 2 can be overcome through stubbornness (or hiring a graduate student. . .)
- Point 3 is a serious concern, if reacting flow solver are to be made attractive for design optimization

Introduction - Improvement to State of the Art

- Current state of the art
  - Attempts made at both continuous[1] and discrete[2] adjoint formulations for a compressible reacting flow solver
  - These attempts suffer from quadratic scaling in memory and computational cost with number of species
  - Recent scheme at Barcelona Supercomputing Center[3] is promising, but only for incompressible reacting flows
- Improvement to the state of the art
  - New decoupled scheme for both hypersonic flow solver and adjoint solver that is robust for high-speed flows in chemical non-equilibrium
  - New schemes significantly improve scaling in computational cost and memory with number of species

[1] **Copeland**.

[2] **Lockwood**.

[3] **Esfahani:2016aa**.

Introduction - Decoupled Approach

- Reacting gas simulations require solving a large number of conservation equations
- Memory concerns
  - Size of Jacobians scales quadratically with number species in gas mixture
  - Solving system of equations in a tightly-coupled fashion can be limited by memory constraints
- Cost concerns
  - Cost of solving the linear system scales quadratically with number of species in gas mixture
- Efficiently solving adjoint problem is a primary motivator
  - Solving adjoint system particularly costly if linear solver is slow
  - Can be necessary to store jacobian twice $\rightarrow$ large memory overhead

Introduction - Decoupled Approach

- Loosely-coupled solvers have become popular in the combustion community[4]
  - Decouple species conservation equations from meanflow equations, and solve two smaller systems

$$\underset{(4+ns)\times(4+ns)}{\begin{pmatrix} \square & \square & \cdots & \square \\ \square & \square & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \square & \cdots & \cdots & \square \end{pmatrix}} \rightarrow \underset{5\times 5}{\begin{pmatrix} \square & \cdots & \square \\ \vdots & \ddots & \vdots \\ \square & \cdots & \square \end{pmatrix}} \text{ and } \underset{ns\times ns}{\begin{pmatrix} \square & \boxslash & \cdots & \boxslash \\ \boxslash & \square & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ \boxslash & \cdots & \cdots & \square \end{pmatrix}}$$

- Candler, et al.[5] originally derived this for Steger-Warming scheme, this work extends to Roe FDS scheme

[4] **Sankaran**.

[5] **candler**.

Outline

Fully-Coupled Point Implicit Flow Solver

- All work presented is for inviscid flows in chemical non-equilibrium, using a one-temperature model, but is extendable to viscous flows.
- Beginning with the semi-discrete form

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

$$\mathbf{U} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{ns} \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \quad \mathbf{F}\cdot\mathbf{S} = \begin{pmatrix} \rho_1 \overline{U} \\ \vdots \\ \rho_{ns}\overline{U} \\ \rho u \overline{U} + p s_x \\ \rho u \overline{U} + p s_y \\ \rho u \overline{U} + p s_z \\ (\rho E + p)\overline{U} \end{pmatrix} S, \quad \mathbf{W} = \begin{pmatrix} \dot{\rho}_1 \\ \vdots \\ \dot{\rho}_{ns} \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Fully-Coupled Point Implicit Flow Solver

- Using the Roe FDS scheme to compute the inviscid flux at the face, $\mathbf{F}^f$, and linearizing the system results in

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

- Which can be thought of more simply as

$$\mathbf{Au} = \mathbf{b}$$

$$\mathbf{A} \rightarrow \begin{array}{c} (4+ns)\times(4+ns) \\ \text{Jacobian Block} \end{array}$$

$$\mathbf{b} \rightarrow \begin{array}{c} (4+ns)\times 1 \\ \text{Residual} \end{array}$$

Fully-Coupled Point Implicit Flow Solver

- Constructing the Jacobian in a fully-coupled fashion results in large, dense block matricies
- Using a stationary iterative method (i.e., Gauss-Seidel, SSOR, etc.), work is dominated by matrix-vector products

$$\text{Cost} \rightarrow O((4+ns)^2)$$

- Leads to onerous quadratic scaling with respect to number of species

Decoupled Point Implicit Flow Solver

- The main idea is to separate the meanflow and species composition equations, adding a new equation for the total mixture density
- Leads to two sets of conserved variables

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

Meanflow Species Composition

Decoupled Point Implicit Flow Solver

- The fluxes are solved in two sequential steps
  - The mixture fluxes are first solved as

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

  - Followed by the species fluxes

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

- Since the mixture density was determined in the first step, step two actually solves for the species mass fractions

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

$$\hat{\mathbf{V}} = (c_1, \ldots, c_{ns})^T, c_s = \rho_s/\rho$$

Decoupled Point Implicit Flow Solver

- The Roe FDS scheme species mass fluxes can be rewritten as

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

$$\frac{\partial \hat{\mathbf{F}}_{\rho_s}}{\partial c_s^L} = w \mathbf{F}_\rho + (1-w)\rho^L \lambda^+ - w\rho^R \lambda^-$$

$$\frac{\partial \hat{\mathbf{F}}_{\rho_s}}{\partial c_s^R} = (1-w)\mathbf{F}_\rho + (w-1)\rho^L \lambda^+ + w\rho^R \lambda^-$$

- Jacobian Approximations

$$\text{Step 1:} \quad \left.\frac{\partial \mathbf{F}}{\partial \mathbf{U}'}\right|_{\hat{\mathbf{V}}} = \underset{c_s=\text{Constant}}{5 \times 5 \, \text{Roe FDS Jacobian}}$$

$$\text{Step 2:} \quad \left.\frac{\partial \mathbf{F}}{\partial \hat{\mathbf{V}}}\right|_{\hat{\mathbf{U}}'} = \begin{pmatrix} \frac{\partial F_{\rho 1}}{\partial c_1} & & 0 \\ & \ddots & \\ 0 & & \frac{\partial F_{\rho ns}}{\partial c_{ns}} \end{pmatrix}$$

Decoupled Point Implicit Flow Solver

- Chemical source term linearized via

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

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

- Full system to be solved in step two

$$\rho^{n+1}\frac{\delta \hat{\mathbf{V}}^{n}}{\Delta t} + \frac{1}{V}\sum_{f}\left(\frac{\partial \hat{\mathbf{F}}^{f}}{\partial \mathbf{V}^{L}}\delta \mathbf{V}^{L} + \frac{\partial \hat{\mathbf{F}}^{f}}{\partial \hat{\mathbf{V}}^{R}}\delta \hat{\mathbf{V}}^{R}\right)^{n,n+1}\mathbf{S}^{f} - \mathbf{C}^{n,n+1}\delta \mathbf{V}^{n}$$

$$= -\frac{1}{V}\sum_{f}(\hat{\mathbf{F}}^{n,n+1}\cdot \mathbf{S})^{f} + \mathbf{W}^{n,n+1} - \hat{\mathbf{V}}^{n}\left(\frac{\delta \rho^{n}}{\Delta t} - R_{\rho}\right)$$

$$R_{\rho} = -\frac{1}{V}\sum_{f}\sum_{s}(\hat{F}^{n,n+1}_{\rho_s}\cdot \mathbf{S})$$

- R_{ρ} is included to preserve $\sum_{s} c_s = 1$, $\sum_{s} \delta c_s = 0$.

Cost and Memory Savings of the Decoupled Flow Solver

- Most significant savings comes from the source term linearization being purely node-based
  - Convective contributions to block Jacobians are diagonal
  - Source term jacobian is dense block Jacobian
  - In the global system (w/chemistry), all off-diagonal block jacobians are diagonal

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

- Matrix-vector products $\rightarrow$ inner products: $O(ns^2) \rightarrow O(ns)$

Cost and Memory Savings of the Decoupled Flow Solver

- Comparing size of Jacobian systems, using Compressed Row Storage

$$\begin{aligned} \mathbf{A}_d &= \text{Decoupled system Jacobians} \\ \mathbf{A} &= \text{Fully-coupled system Jacobians} \end{aligned}$$

$$\begin{aligned} Relative\ Memory\ Cost &= \frac{size(\mathbf{A}_d)}{size(\mathbf{A})} \\ &= \lim_{ns \to \infty} \frac{(ns^2 + 5^2)(N_{nodes}) + (ns + 5^2)(N_{nbrs})}{(ns + 4)^2(N_{nodes} + N_{nbrs})} \\ &= \frac{N_{nodes}}{N_{nodes} + N_{nbrs}} \end{aligned}$$

Outline

Derivation of Discrete Adjoint Formulation

- The derivation of the adjoint approach to compute design sensitivities begins with forming the Lagrangian and differentiating with respect to the design variables

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

$\mathbf{D}$ = design variables | f = cost function
$\mathbf{Q}$ = flow variables | $\mathbf{R}$ = flow residual
$\mathbf{X}$ = computational grid | Λ = costate variables

Derivation of Discrete Adjoint Formulation

- The derivation of the adjoint approach to compute design sensitivities begins with forming the Lagrangian and differentiating with respect to the design variables

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

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

$$+ \left\{ \left[\frac{\partial \mathbf{R}}{\partial \mathbf{D}}\right]^T + \left[\frac{\partial \mathbf{X}}{\partial \mathbf{D}}\right]^T \left[\frac{\partial \mathbf{R}}{\partial \mathbf{X}}\right]^T \right\} \Lambda$$

$\mathbf{D}$ = design variables $\quad f$ = cost function

$\mathbf{Q}$ = flow variables $\quad \mathbf{R}$ = flow residual

$\mathbf{X}$ = computational grid $\quad \Lambda$ = costate variables

Derivation of Discrete Adjoint Formulation

- The derivation of the adjoint approach to compute design sensitivities begins with forming the Lagrangian and differentiating with respect to the design variables

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

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

$$+ \left\{\left[\frac{\partial \mathbf{R}}{\partial \mathbf{D}}\right]^T + \left[\frac{\partial \mathbf{X}}{\partial \mathbf{D}}\right]^T \left[\frac{\partial \mathbf{R}}{\partial \mathbf{X}}\right]^T\right\}\Lambda$$

$\mathbf{D}$ = design variables

$\mathbf{Q}$ = flow variables

$\mathbf{X}$ = computational grid

f = cost function

$\mathbf{R}$ = flow residual

Λ = costate variables

Derivation of Discrete Adjoint Formulation

- Need to eliminate flow variable dependence on design variables, $\frac{\partial \mathbf{Q}}{\partial \mathbf{D}}$
- Adjoint equation

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

- Solve for Λ and compute sensitivity derivatives

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

Fully Coupled Adjoint Solver

- Adjoint problem is a linear system

$$\begin{pmatrix} \frac{\partial \mathbf{R}_{\rho_i}}{\partial \rho_j}^T & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho_j}^T & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho_j}^T \\ \frac{\partial \mathbf{R}_{\rho_i}}{\partial \rho \mathbf{u}}^T & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}}^T & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}}^T \\ \frac{\partial \mathbf{R}_{\rho_i}}{\partial \rho E}^T & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E}^T & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E}^T \end{pmatrix} \begin{pmatrix} \Lambda_{\rho_i} \\ \Lambda_{\rho \mathbf{u}} \\ \Lambda_{\rho E} \end{pmatrix} = - \begin{pmatrix} \frac{\partial f}{\partial \rho_i} \\ \frac{\partial f}{\partial \rho \mathbf{u}} \\ \frac{\partial f}{\partial \rho E} \end{pmatrix}$$

- Can be solved with Krylov method (i.e. GMRES), but time marching similar to flow solver shown to be more robust

$$\left(\frac{V}{\Delta t} \mathbf{I} + \frac{\partial \mathbf{R}_1}{\partial \mathbf{Q}}^T \right) \Delta \Lambda = -\frac{\partial f}{\partial \mathbf{Q}} - \frac{\partial \mathbf{R}_2}{\partial \mathbf{Q}}^T \Lambda^n$$

- Straightforward to formulate, but cost and memory requirements scale quadratically with number of species

Decoupled Adjoint Scheme

- The decoupled flow solver has an analog in the adjoint
- First, recognize that the decoupled flow solver can be rewritten as a fully coupled system, with a change of variables and change of equations

$$\underbrace{\mathbf{U} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{ns} \\ \rho\mathbf{u} \\ \rho E \end{pmatrix} \rightarrow \mathbf{V} = \begin{pmatrix} c_1 \\ \vdots \\ c_{ns} \\ \rho \\ \rho\mathbf{u} \\ \rho E \end{pmatrix}}_{\text{Change of Variables}}, \quad \underbrace{\mathbf{R_U} = \begin{pmatrix} \mathbf{R}_{\rho_1} \\ \vdots \\ \mathbf{R}_{\rho_{N_s}} \\ \mathbf{R}_{\rho\mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix} \rightarrow \mathbf{R_V} = \begin{pmatrix} \mathbf{R}_{\rho_1} - c_1 \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \vdots \\ \mathbf{R}_{\rho_{N_s}} - c_{N_s} \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \sum_{i=1}^{N_s} (\mathbf{R}_{\rho_i}) \\ \mathbf{R}_{\rho\mathbf{u}} \\ \mathbf{R}_{\rho E} \end{pmatrix}}_{\text{Change of Equations}}$$

$$c_s = \frac{\rho_s}{\rho}, \quad \rho = \sum_{i=1}^{N_s} (\rho_i)$$

Decoupled Adjoint Scheme

- This change of variables/equations results in non-square transformation matricies

$$\frac{\partial \mathbf{U}}{\partial \mathbf{V}} = \begin{pmatrix} \rho & \dots & 0 & c_1 & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \rho & c_{ns} & 0 & 0 \\ 0 & \dots & 0 & 0 & 1 & 0 \\ 0 & \dots & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}} = \begin{pmatrix} 1 - c_1 & -c_1 & \dots & -c_1 & 0 & 0 \\ -c_2 & 1 - c_2 & \dots & -c_2 & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -c_{N_s} & -c_{N_s} & \dots & 1 - c_{N_s} & 0 & 0 \\ 1 & 1 & \dots & 1 & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 & 0 \\ 0 & 0 & \dots & 0 & 0 & 1 \end{pmatrix}$$

Decoupled Adjoint Scheme

- Using the transformation matricies, $\frac{\partial \mathbf{U}}{\partial \mathbf{V}}$ and $\frac{\partial \mathbf{R_U}}{\partial \mathbf{R_V}}$, it possible to treat the decoupled approach as a series of matrix operations

$$\frac{\partial \mathbf{R_V}}{\partial \mathbf{V}} = \frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}} \frac{\partial \mathbf{R_U}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}}$$

Decoupled Adjoint Scheme

- Using the transformation matricies, $\frac{\partial \mathbf{U}}{\partial \mathbf{V}}$ and $\frac{\partial \mathbf{R_U}}{\partial \mathbf{R_V}}$, it possible to treat the decoupled approach as a series of matrix operations

$$\frac{\partial \mathbf{R_V}}{\partial \mathbf{V}} = \frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}} \frac{\partial \mathbf{R_U}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}}$$

- These matrix operations can be thought of as left and right preconditioning

$$\frac{\partial \mathbf{R_V}}{\partial \mathbf{V}} = \underbrace{\left(\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}}\right)}_{\text{Left Preconditioner}} \left(\frac{\partial \mathbf{R_U}}{\partial \mathbf{U}}\right) \underbrace{\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)}_{\text{Right Preconditioner}}$$

Decoupled Adjoint Scheme

- Using the transformation matricies, $\frac{\partial \mathbf{U}}{\partial \mathbf{V}}$ and $\frac{\partial \mathbf{R_U}}{\partial \mathbf{R_V}}$, it possible to treat the decoupled approach as a series of matrix operations

$$\frac{\partial \mathbf{R_V}}{\partial \mathbf{V}} = \frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}} \frac{\partial \mathbf{R_U}}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial \mathbf{V}}$$

- These matrix operations can be thought of as left and right preconditioning

$$\frac{\partial \mathbf{R_V}}{\partial \mathbf{V}} = \underbrace{\left(\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}}\right)}_{\text{Left Preconditioner}} \left(\frac{\partial \mathbf{R_U}}{\partial \mathbf{U}}\right) \underbrace{\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)}_{\text{Right Preconditioner}}$$

- This transformation avoids having to explicitly form the jacobian $\frac{\partial \mathbf{R_V}}{\partial \mathbf{V}}$, which are complicated than $\frac{\partial \mathbf{R_U}}{\partial \mathbf{U}}$

Decoupled Adjoint Scheme

- Rewrite the adjoint system from $\mathbf{R_V}(\mathbf{V})$

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

Decoupled Adjoint Scheme

- Rewrite the adjoint system from $\mathbf{R_V}(\mathbf{V}) \rightarrow \mathbf{R_U}(\mathbf{U})$

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

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

Decoupled Adjoint Scheme

- Rewrite the adjoint system from $\mathbf{R_V}(\mathbf{V}) \rightarrow \mathbf{R_U}(\mathbf{U})$

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

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

- Effectively another Left/Right Preconditioning

$$\underbrace{\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)^T}_{\text{Left Preconditioning}} \left(\frac{\partial \mathbf{R_U}}{\partial \mathbf{U}}\right)^T \underbrace{\left(\frac{\partial \mathbf{R_V}}{\partial \mathbf{R_U}}\right)^T \Lambda_\mathbf{V}}_{\text{Right Preconditioning}} = -\underbrace{\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)^T}_{\text{Left Preconditioning}} \left(\frac{\partial f}{\partial \mathbf{U}}\right)$$

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

Decoupled Adjoint Solver

- Time march adjoint solution with approximate jacobians

$$\left(\frac{V}{\Delta t}\mathbf{I} + \frac{\partial \mathbf{R}_{\mathbf{V}_1}}{\partial \mathbf{V}}^T\right)\Lambda_{\mathbf{V}} = -\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)^T \underbrace{\left(\frac{\partial \mathbf{R}_{\mathbf{U}_2}}{\partial \mathbf{U}}\Lambda_{\mathbf{U}} + \frac{\partial f}{\partial \mathbf{U}}\right)}_{\text{Fully Coupled Residual}}$$

- Split $\frac{\partial \mathbf{R}_{\mathbf{V}_1}}{\partial \mathbf{V}}$ in same fashion as flow solver

Decoupled Adjoint Solver

- Time march adjoint solution with approximate jacobians

$$\left(\frac{V}{\Delta t}\mathbf{I} + \frac{\partial \mathbf{R}_{\mathbf{V}_1}}{\partial \mathbf{V}}^T\right)\Lambda_{\mathbf{V}} = -\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)^T \underbrace{\left(\frac{\partial \mathbf{R}_{\mathbf{U}_2}}{\partial \mathbf{U}}\Lambda_{\mathbf{U}} + \frac{\partial f}{\partial \mathbf{U}}\right)}_{\text{Fully Coupled Residual}}$$

- Split $\frac{\partial \mathbf{R}_{\mathbf{V}_1}}{\partial \mathbf{V}}$ in same fashion as flow solver

$$\begin{pmatrix}
\frac{\partial \mathbf{R}'_{\rho 1}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}'_{\rho 1}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}'_{\rho 1}}{\partial \rho} & \frac{\partial \mathbf{R}'_{\rho 1}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}'_{\rho 1}}{\partial \rho E} \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial \mathbf{R}'_{\rho N_s}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}'_{\rho N_s}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}'_{\rho N_s}}{\partial \rho} & \frac{\partial \mathbf{R}'_{\rho N_s}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}'_{\rho N_s}}{\partial \rho E} \\
\frac{\partial \mathbf{R}_{\rho}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho E} \\
\frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} \\
\frac{\partial \mathbf{R}_{\rho E}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho E}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E}
\end{pmatrix}$$

Decoupled Adjoint Solver

- Time march adjoint solution with approximate jacobians

$$\left(\frac{V}{\Delta t}\mathbf{I} + \frac{\partial \mathbf{R}_{\mathbf{V}_1}}{\partial \mathbf{V}}^T\right)\Lambda_{\mathbf{V}} = -\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)^T \underbrace{\left(\frac{\partial \mathbf{R}_{\mathbf{U}_2}}{\partial \mathbf{U}}\Lambda_{\mathbf{U}} + \frac{\partial f}{\partial \mathbf{U}}\right)}_{\text{Fully Coupled Residual}}$$

- Split $\frac{\partial \mathbf{R}_{\mathbf{V}_1}}{\partial \mathbf{V}}$ in same fashion as flow solver

$$\begin{pmatrix}
\frac{\partial \mathbf{R}'_{\rho 1}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}'_{\rho 1}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}'_{\rho 1}}{\partial \rho} & \frac{\partial \mathbf{R}'_{\rho 1}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}'_{\rho 1}}{\partial \rho E} \\
\vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\
\frac{\partial \mathbf{R}'_{\rho N_s}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}'_{\rho N_s}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}'_{\rho N_s}}{\partial \rho} & \frac{\partial \mathbf{R}'_{\rho N_s}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}'_{\rho N_s}}{\partial \rho E} \\
\frac{\partial \mathbf{R}_{\rho}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho}}{\partial \rho E} \\
\frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho \mathbf{u}}}{\partial \rho E} \\
\frac{\partial \mathbf{R}_{\rho E}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho E}}{\partial c_{N_s}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho \mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E}
\end{pmatrix}$$

Decoupled Adjoint Solver

- Time march adjoint solution with approximate jacobians

$$\left(\frac{V}{\Delta t}\mathbf{I} + \frac{\partial \mathbf{R}_{\mathbf{V}_1}}{\partial \mathbf{V}}^T\right)\Lambda_{\mathbf{V}} = -\left(\frac{\partial \mathbf{U}}{\partial \mathbf{V}}\right)^T \underbrace{\left(\frac{\partial \mathbf{R}_{\mathbf{U}_2}}{\partial \mathbf{U}}\Lambda_{\mathbf{U}} + \frac{\partial f}{\partial \mathbf{U}}\right)}_{\text{Fully Coupled Residual}}$$

- Split $\frac{\partial \mathbf{R}_{\mathbf{V}_1}}{\partial \mathbf{V}}$ in same fashion as flow solver

$$\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}
\rightarrow
\begin{pmatrix}
\frac{\partial \mathbf{R}_{\rho 1}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho 1}}{\partial c_{N_s}} \\
\vdots & \ddots & \vdots \\
\frac{\partial \mathbf{R}_{\rho N_s}}{\partial c_1} & \cdots & \frac{\partial \mathbf{R}_{\rho N_s}}{\partial c_{N_s}}
\end{pmatrix},
\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}$$

Species Equations

Mixture Equations

Decoupled Adjoint Solver

- Separate into two systems and solve as block jacobi scheme

$$\boxed{\left(\frac{\rho V}{\Delta t}\mathbf{I} + \frac{\partial \mathbf{R}_{\mathbf{V}_i}}{\partial c_s}\right)\Delta\Lambda_{\mathbf{V}_{c_i}} = -\rho\left(\frac{\partial f}{\partial \rho_s} + \frac{\partial \mathbf{R}_{\mathbf{U}}}{\partial \rho_s}\Lambda_{\mathbf{U}}\right)}$$

$$\left[\frac{V}{\Delta t}\mathbf{I} + \begin{pmatrix} \frac{\partial \mathbf{R}_\rho}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho} \\ \frac{\partial \mathbf{R}_\rho}{\partial \rho\mathbf{u}} & \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial \rho\mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho\mathbf{u}} \\ \frac{\partial \mathbf{R}_\rho}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix}\right] \begin{pmatrix} \Delta\Lambda_{\mathbf{V}_\rho} \\ \Delta\Lambda_{\mathbf{V}_{\rho\mathbf{u}}} \\ \Delta\Lambda_{\mathbf{V}_{\rho E}} \end{pmatrix} =$$
$$- \begin{pmatrix} \frac{\partial f}{\partial \rho} \\ \frac{\partial f}{\partial \rho\mathbf{u}} \\ \frac{\partial f}{\partial \rho E} \end{pmatrix} - \begin{pmatrix} \frac{\partial \mathbf{R}_\rho}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial \rho} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho} \\ \frac{\partial \mathbf{R}_\rho}{\partial \rho\mathbf{u}} & \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial \rho\mathbf{u}} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho\mathbf{u}} \\ \frac{\partial \mathbf{R}_\rho}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial \rho E} & \frac{\partial \mathbf{R}_{\rho E}}{\partial \rho E} \end{pmatrix} \begin{pmatrix} \Lambda_\rho \\ \Lambda_{\rho\mathbf{u}} \\ \Lambda_{\rho E} \end{pmatrix} - \begin{pmatrix} \frac{\partial \mathbf{R}_\rho}{\partial c_s}^T \\ \frac{\partial \mathbf{R}_{\rho\mathbf{u}}}{\partial c_s}^T \\ \frac{\partial \mathbf{R}_{\rho E}}{\partial c_s}^T \end{pmatrix} \Lambda_{c_s}$$