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

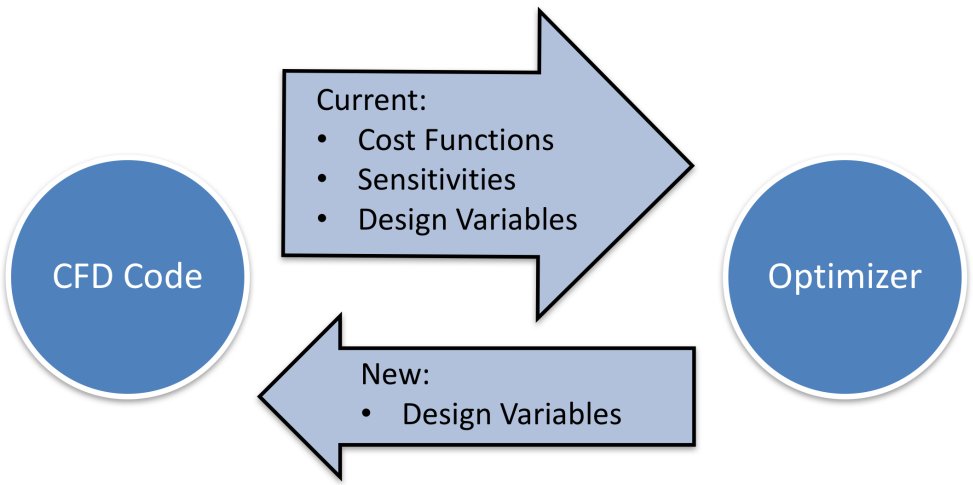
Aerothermodynamics Branch
NASA Langley Research Center

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

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

- 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 Method

- 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 Method

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

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

- 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 Method

- 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 Method

- 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 Method

- 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 Method

- 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

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

$$\text{Step 2:} \quad \left.\frac{\partial \mathbf{F}}{\partial \hat{\mathbf{V}}}\right|_{\hat{\mathbf{U}}'} = \begin{pmatrix} \frac{\partial F_{\rho_1}}{\partial c_1} & & 0 \\ & \ddots & \\ 0 & & \frac{\partial F_{\rho_{ns}}}{\partial c_{ns}} \end{pmatrix}$$

Decoupled Point Implicit Method

- 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

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

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

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

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

Outline

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

$$\mathbf{A}_d = \text{Decoupled system Jacobians}$$
$$\mathbf{A} = \text{Fully-coupled system Jacobians}$$

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

Numerical Results: 2D Cylinder

- Fully-coupled and decoupled methods both implemented in the Generic Gas Path of FUN3D
- Tested on 2D cylinder case
  - $V_\infty = 5000\ m/s$, $\rho_\infty = 0.001\ kg/m^3$, and $T_\infty = 200\ K$
- Inviscid flow, with 1-Temperature model

Numerical Results: 2D Cylinder

- Verification of implementation
  - 5-species air model: N, N_2, O, O_2, and NO with five reactions
  - Surface pressure, surface temperature, and mass fractions on stagnation line agree between decoupled and fully coupled implementations

Numerical Results: 2D Cylinder

- On structured grids $N_{nbrs} \approx 6N_{nodes}$
  - Half precision off-diagonal $N_{nbrs} = \frac{6N_{nodes}}{2}$

$$Memory\ \ Cost \approx \frac{N_{nodes}}{N_{nodes} + N_{nbrs}} = \frac{N_{nodes}}{N_{nodes} + 6N_{nodes}/2} = \frac{1}{4}$$

  - Linear speedup in solver: $\frac{O(ns^2)}{O(ns)} = O(ns)$

Numerical Results: Axisymmetric Spherically Capped Cone

- Verify that the decoupled scheme is robust at high velocities
  - $V_\infty = 15000\ m/s$, $\rho_\infty = 0.001\ kg/m^3$, $T_\infty = 200\ K$.
  - 11-species air model N, N_2, O, O_2, NO, N^+, N_2^+, O^+, O_2^+, NO^+, and electrons, with 22 possible reactions.
  - Inviscid flow, with 1-Temperature model

Numerical Results: Axisymmetric Spherically Capped Cone

- Surface pressure and surface temperature agree between decoupled and fully coupled implementations

Numerical Results: Axisymmetric Spherically Capped Cone

- Necessary to scale source term magnitude by $0.001 \leq w \leq 1$ for the first 500 iterations, due to extreme reaction rates
- Both schemes converge in a similar number of iterations
- Decoupled scheme $\approx$ 2x faster

Outline

Linearization Schemes Review

- Decoupled scheme based on work by Candler, et. al[6]
- Main idea is to "decouple" the conserved variable vector, $\mathbf{U}$, into mixture, $\mathbf{U}'$, and species, $\hat{\mathbf{V}}$, variable vectors

$$\mathbf{U} = \begin{pmatrix} \rho_1 \\ \vdots \\ \rho_{N_s} \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix} \rightarrow \mathbf{U}' = \begin{pmatrix} \rho \\ \rho u \\ \rho v \\ \rho w \\ \rho E \end{pmatrix}, \quad \hat{\mathbf{V}} = \begin{pmatrix} c_1 \\ \vdots \\ c_{N_s} \end{pmatrix} \tag{1}$$

$$c_s = \frac{\rho_s}{\rho}, \quad N_s = \text{Number of species}$$

[1] Graham V. Candler, Pramod K. Subbareddy, and Ioannis Nompelis. "Decoupled Implicit Method for Aerothermodynamics and Reacting Flows", AIAA Journal, Vol. 51, No. 5 (2013), pp. 1245-1254.

Linearization Schemes Review

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

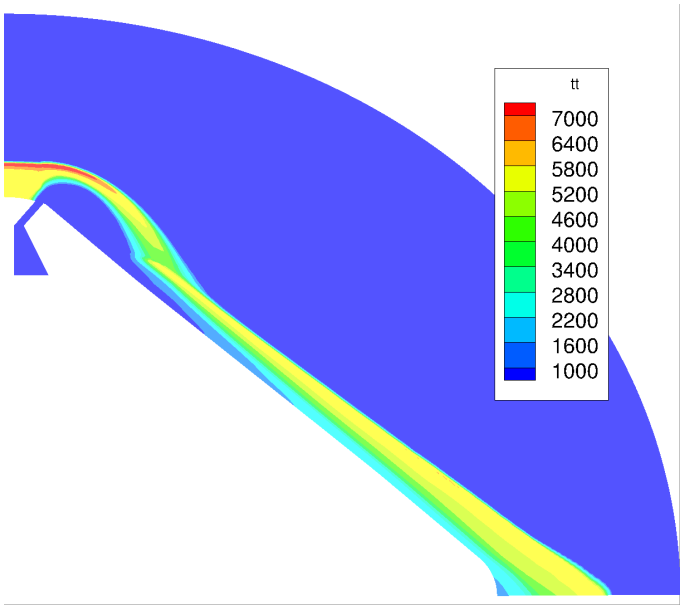
- Cost and memory saving for decoupled scheme stem from decoupled mass fraction block Jacobians, $\frac{\partial R}{\partial \hat{\mathbf{V}}}$, being diagonal for convective flux.
  - Linear solve reduced from $O(N_s^2) \rightarrow O(N_s)$
  - Relative memory savings between decoupled and fully-coupled linearization schemes (for structured-type grid)[7]

$$\lim_{N_s \rightarrow \infty} \frac{\text{decoupled memory req.}}{\text{fully coupled memory req.}} = \frac{1}{7} \tag{2}$$

[2] Relative memory savings theoretically greater for tetrahedral grids

Outline

Annular Nozzle - Geometry



Annular Jet Temperature Contours

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

Flow Conditions

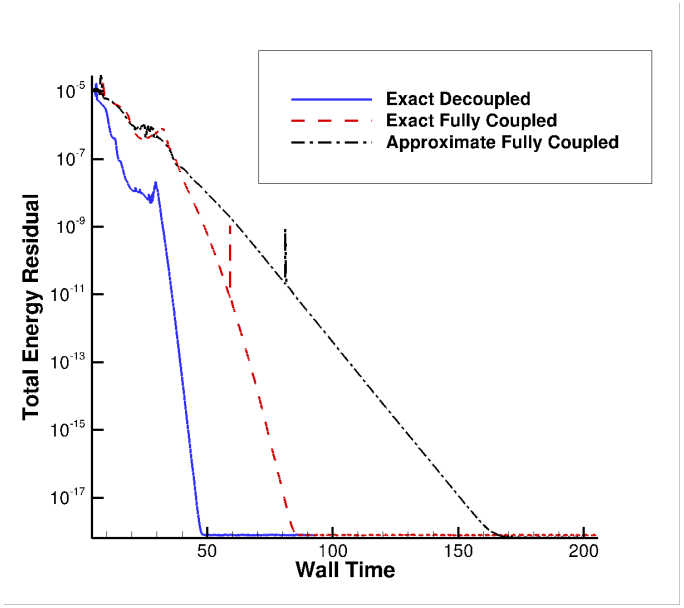
Parameter	Description	Value
r_{throat}	nozzle throat radius, m	0.02
r_{plenum}	nozzle radius at plenum face, m	0.05
$r_{exit,inner}$	inside nozzle radius at exit, m	0.05
$r_{exit,outer}$	outside nozzle radius at exit, m	0.07
l_{conv}	distance from plenum to throat, m	0.05
θ_c	cone half angle, deg	50.0

Annular Nozzle Geometry Inputs

- Note: 50^o cone angle chosen to prevent sonic corner body

Annular Nozzle - Flow Solver Cost Savings

- Best case scenario: decoupled scheme 1st-order w/frozen flow



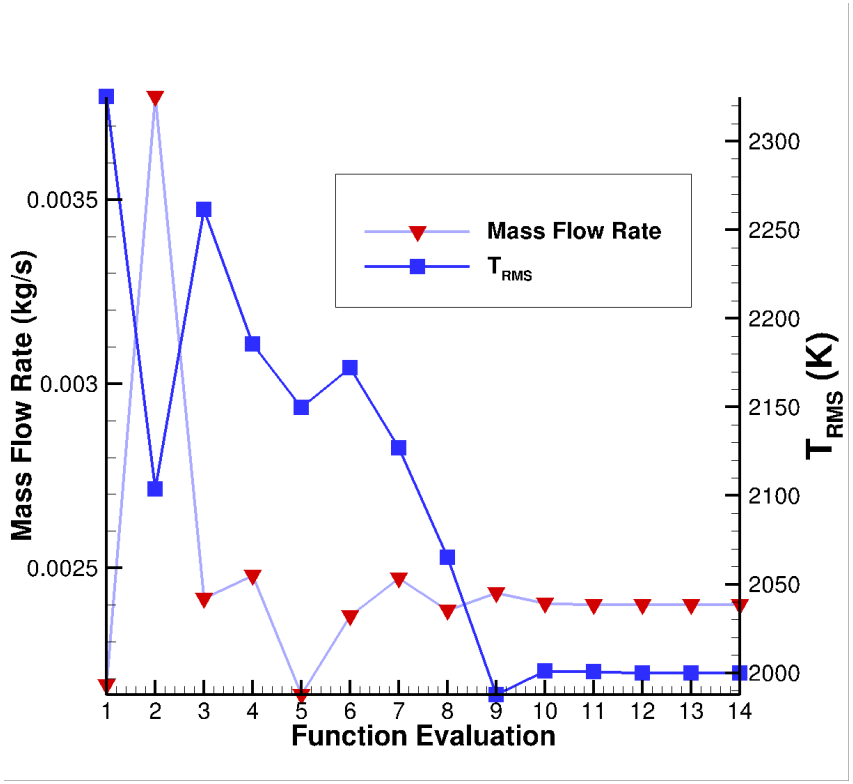
1st-Order Frozen Flow Convergence History

Scheme	Time to Convergence (s)	Speedup
Approx. FC	150.7	1.00 (baseline)
Exact FC	80.71	1.87
Exact DC	45.60	3.30

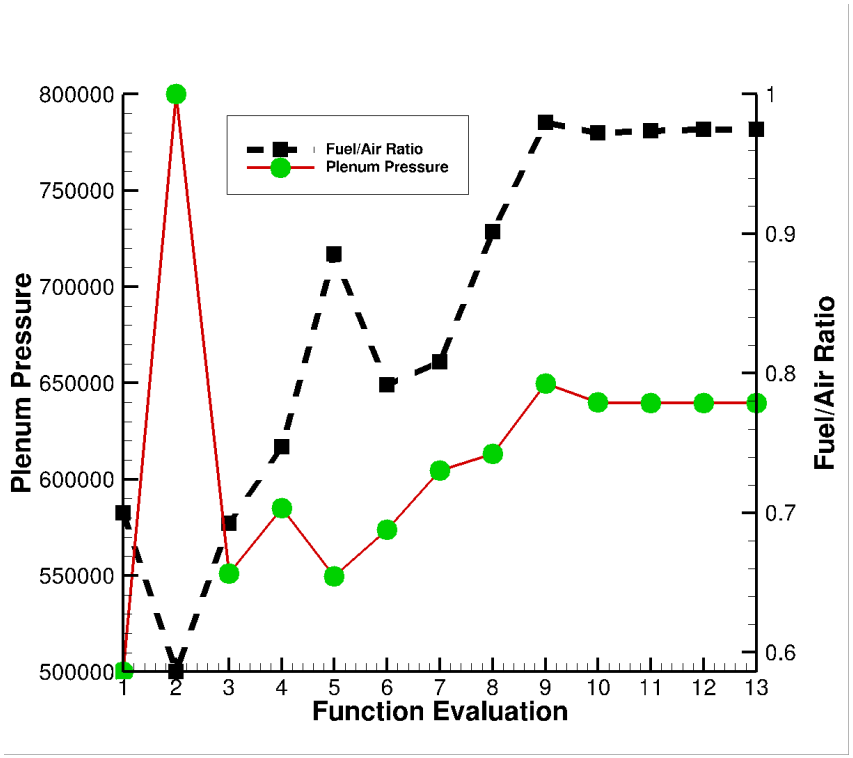
Speedup relative to the approximate, fully-coupled Jacobians

- CFL ramped from 0.1 $\rightarrow$ 30.0 for all schemes
- Exact linearizations significantly improve the rate of convergence after non-linear transients of startup

Annular Nozzle - Inverse Design Optimization



Temperature and mass flow rate design history



Design variable history

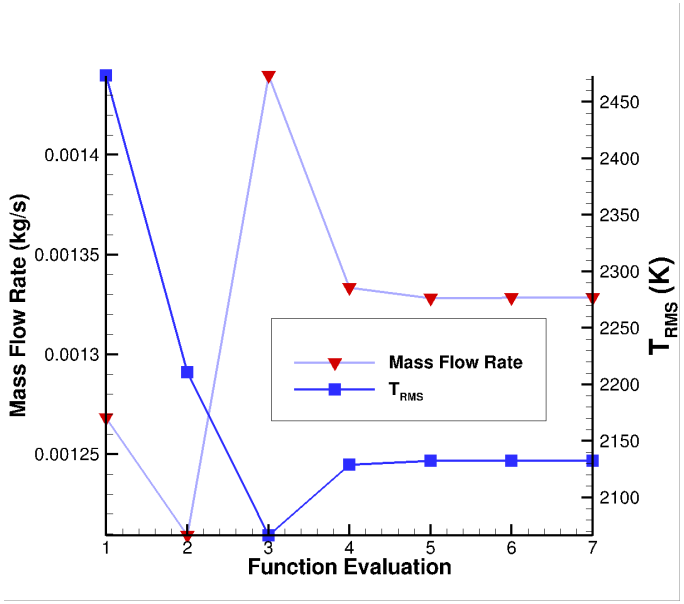
- Plenum pressure and H_2/N_2 ratio chosen as design variables
- Design to within 10^{-4} of target by 11 Flow/Adjoint solves

Outline

Concluding Remarks

- 1st-order inverse and direct design optimization very robust for annular jet
- Decoupled linearizations very robust, and typically can be run with same options as fully-coupled schemes.
  - Source term scaling is still needed for intense chemical reactions (i.e. combustion, full N_2 dissociation, etc.)
  - 2nd-order has more complicated history, but 2x speed is generally recovered by the decoupled scheme over approximate and exact fully coupled schemes
- Fully coupled adjoint robust, and costs $\sim \frac{1}{2}$ flow solution cost
  - Decoupled adjoint yields both super-convergence and divergence. Still under investigation.

Backup - Annular Nozzle Direct Design



Temperature and mass flow rate design history

- Cost function definition

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

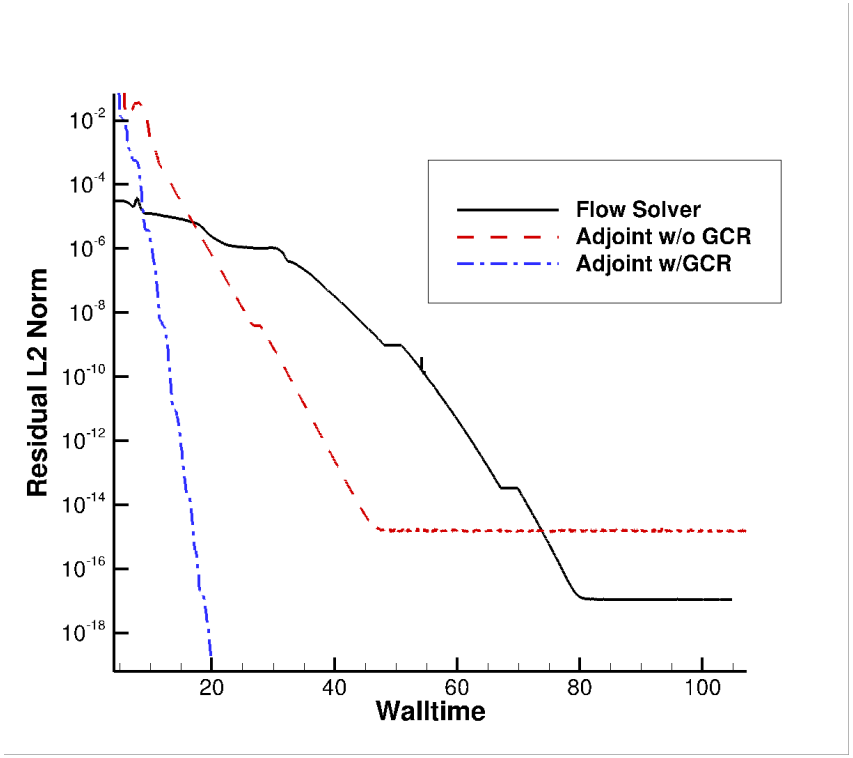
$$\frac{w_1}{w_2} = \frac{\left(T_{RMS}\right)^2}{\left(\dot{m}\right)^2}$$

- Net design improved by 9.3%

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

Design Improvement

Backup - Adjoint Convergence



1st-order Adjoint Convergence

- First order adjoint convergence for exact, fully-coupled linearizations converges in $\sim \frac{1}{2}$ the time required by the flow solver
- If krylov scheme (GCR) is used, the time required is $\sim \frac{1}{4}$ the time required by the flow solver