0, 7→1: Optimization

1, 5→2: MDA

2: $x_0, x_1$
3: $x_0, x_2$
4: $x_0, x_3$
6: $x$

2: $y^t_2, y^t_3$
3: $y^t_3$

5: $y_1$
6: $y_1$

5: $y_2$
6: $y_2$

5: $y_3$
6: $y_3$

7: $f, c$

6: Functions

A Very Short Course on Multidisciplinary Design Optimization

Joaquim R. R.A. Martins • John T. Hwang

Multidisciplinary Design Optimization Laboratory
http://mdolab.engin.umich.edu
Vitae

Bio

• 1991–1995: M.Eng. in Aeronautics, Imperial College, London, UK
• 1996–2002: M.Sc. and Ph.D. in Aeronautics and Astronautics, Stanford University
• 2002–2008: Assistant Professor, University of Toronto, Institute for Aerospace Studies
• 2008–2009: Associate Professor, University of Toronto, Institute for Aerospace Studies
• 2009–2015: Associate Professor, University of Michigan, Dept. of Aerospace Engineering
• 2015–: Professor, University of Michigan, Dept. of Aerospace Engineering
• 2015–2016: Professeur Invité, ISAE–SUPAERO

Highlights

• Canada Research Chair in Multidisciplinary Optimization (2002–2009)
• Keynote speaker at the International Forum on Aeroelasticity and Structural Dynamics (Stockholm, 2007)
• Keynote speaker at the Aircraft Structural Design Conference (London, 2010)
• Associate editor for Optimization and Engineering, Structural and Multidisciplinary Optimization
• Marie Curie Fellow (2015–2016)
Design Optimization
What is Multidisciplinary Design Optimization — MDO?

Baseline design usually requires some engineering intuition and represents an initial idea. In the conventional design process this baseline design is analyzed in some way to determine its performance. This could involve numerical modeling or actual building and testing. The design is then evaluated based on the results and the designer then decides whether the design is good enough or not. If the answer is no — which is likely to be the case for at least the first few iterations — the designer will change the design based on its intuition, experience or trade studies. When the design is satisfactory, the designer will arrive at the final design.

For more complex engineering systems, there are multiple levels and thus cycles in the design process. In aircraft design, these would correspond to the preliminary, conceptual and detailed design stages.

The design optimization process can be pictured using the same flow chart, with modifications to some of the blocks. Instead of having the option to build a prototype, the analysis step must be completely numerical and must not involve any input from the designer. The evaluation of the design is strictly based on numerical values for the objective to be minimized and the constraints that need to be satisfied. When a rigorous optimization algorithm is used, the decision to finalize the design is made only when the current design satisfies the necessary optimality conditions that ensure that no other design “close by” is better. The changes in the design are made automatically by the optimization algorithm and do not require the intervention of the designer. On the other hand, the designer must decide in advance which parameters can be changed. In the design optimization process, it is crucial that the designer formulate the optimization problem well. We will now discuss the components of this formulation in more detail: the objective function, the constraints, and the design variables.
Design Variables, Objective Function, and Constraints

Additional Design Variables

Most practical design problems have vastly more than the two design variables \( \{\text{AR}, S\} \) assumed in the examples above. A basic rule is that any adjustable quantity which is likely to have a strong effect on the constrained objective function should be considered as a design variable. One such candidate is the wing taper ratio \( \frac{c_t}{c_r} = \lambda \), which clearly has a powerful effect on the tip deflection in relation (12). If \( \lambda \) is chosen as a new design variable, the design space is now three dimensional as shown in Figure 5.

\[
\begin{align*}
\{\text{AR}, S, \lambda\} & \quad (14) \\
\end{align*}
\]

Figure 5: Three-variable design space. As before, each point represents a unique design.

[Figure 4: Wing deflection/span contours (dashed) superimposed on objective function contours (solid). The contour \( \delta/b = 0.05 \) is the constraint boundary. Black dot shows the constrained optimum-design minimum power point.]

[Figure 5: Three-variable design space. As before, each point represents a unique design.]

[Drela, 2006]
1. Multidisciplinary Design Optimization
1.1 Introduction
1.2 Multidisciplinary Analysis
1.3 Extended Design Structure Matrix
1.4 Monolithic Architectures
   Multidisciplinary Feasible (MDF)
   Individual Discipline Feasible (IDF)
   Simultaneous Analysis and Design (SAND)
   The All-at-Once (AAO) Problem Statement
1.5 Distributed Architectures
   Classification
1.6 Computing Coupled Derivatives
Multidisciplinary Design Optimization

1. Multidisciplinary Design Optimization
   1.1 Introduction
   1.2 Multidisciplinary Analysis
   1.3 Extended Design Structure Matrix
   1.4 Monolithic Architectures
   1.5 Distributed Architectures
   1.6 Computing Coupled Derivatives
Introduction 1

- In the last few decades, numerical models that predict the performance of engineering systems have been developed, and many of these models are now mature areas of research. For example . . .

- Once engineers can predict the effect that changes in the design have on the performance of a system, the next logical question is what changes in the design produced optimal performance. The application of the numerical optimization techniques described in the preceding chapters address this question.

- Single-discipline optimization is in some cases quite mature, but the design and optimization of systems that involve more than one discipline is still in its infancy.

- When systems are composed of multiple systems, additional issues arise in both the analysis and design optimization.

- MDO researchers think industry will not adopt MDO more widely because they do not realize their utility.
Introduction 2

- Industry think that researchers are not presenting anything new, since industry has already been doing multidisciplinary design.
- There is some truth to each of these perspectives . . .
- Real-world aerospace design problem may involve thousands of variables and hundreds of analyses and engineers, and it is often difficult to apply the numerical optimization techniques and solve the mathematically correct optimization problems.
- The kinds of problems in industry are often of much larger scale, involve much uncertainty, and include human decisions in the loop, making them difficult to solve with traditional numerical optimization techniques.
- On the other hand, a better understanding of MDO by engineers in industry is now contributing a more widespread use in practical design.

Why MDO?

- Parametric trade studies are subject to the “curse of dimensionality”.
- Iterated procedures for which convergence is not guaranteed.
- Sequential optimization that does not lead to the true optimum of the system
Introduction 3

Objectives of MDO:
- Avoid difficulties associated with sequential design or partial optimization.
- Provide more efficient and robust convergence than by simple iteration.
- Aid in the management of the design process.

Difficulties of MDO:
- Communication and translation
- Time
- Scheduling and planning
- Implementation
Typical Aircraft Company Organization

Personnel hierarchy

Design process
MDO Architectures

- MDO focuses on the development of strategies that use numerical analyses and optimization techniques to enable the automation of the design process of a multidisciplinary system.
- The big challenge: make such a strategy scalable and practical.
- An MDO architecture is a particular strategy for organizing the analysis software, optimization software, and optimization subproblem statements to achieve an optimal design.
- Other terms are used: “method”, “methodology”, “problem formulation”, “strategy”, “procedure” and “algorithm”.

J.R.R.A. Martins
Multidisciplinary Design Optimization
August 2012 8 / 75
## Nomenclature and Mathematical Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>Vector of design variables</td>
</tr>
<tr>
<td>$y^t$</td>
<td>Vector of coupling variable targets (inputs to a discipline analysis)</td>
</tr>
<tr>
<td>$y$</td>
<td>Vector of coupling variable responses (outputs from a discipline analysis)</td>
</tr>
<tr>
<td>$\bar{y}$</td>
<td>Vector of state variables (variables used inside only one discipline analysis)</td>
</tr>
<tr>
<td>$f$</td>
<td>Objective function</td>
</tr>
<tr>
<td>$c$</td>
<td>Vector of design constraints</td>
</tr>
<tr>
<td>$c^c$</td>
<td>Vector of consistency constraints</td>
</tr>
<tr>
<td>$\mathcal{R}$</td>
<td>Governing equations of a discipline analysis in residual form</td>
</tr>
<tr>
<td>$N$</td>
<td>Number of disciplines</td>
</tr>
<tr>
<td>$n()$</td>
<td>Length of given variable vector</td>
</tr>
<tr>
<td>$m()$</td>
<td>Length of given constraint vector</td>
</tr>
<tr>
<td>$()_0$</td>
<td>Functions or variables that are shared by more than one discipline</td>
</tr>
<tr>
<td>$()_i$</td>
<td>Functions or variables that apply only to discipline $i$</td>
</tr>
<tr>
<td>$()^*$</td>
<td>Functions or variables at their optimal value</td>
</tr>
<tr>
<td>$()$</td>
<td>Approximation of a given function or vector of functions</td>
</tr>
<tr>
<td>$\hat{()}$</td>
<td>Duplicates of certain variable sets distributed to other disciplines</td>
</tr>
</tbody>
</table>
Nomenclature and Mathematical Notation 2

- In MDO, we make the distinction between:
  - **Local** design variables \( x_i \) — directly affect only one discipline
  - **Shared** design variables \( x_0 \) — directly affect more than one discipline.

- Full vector of design variables \( x = [x_0^T, x_1^T, \ldots, x_N^T]^T \)

- A discipline analysis solves a system of equations that computes the state variables. Examples?

- In many formulations, *independent copies* of the coupling variables must be made to allow discipline analyses to run independently and in parallel.

- These copies are also known as *target variables*, which we denote by a superscript \( t \).

- To preserve consistency between the coupling variable inputs and outputs at the optimal solution, we define **consistency constraints**

\[
  c_i^c = y_{i}^t - y_i
\]

which we add to the optimization problem formulation.
Example: Aerostructural Problem Definition 1

- Common example used throughout this chapter to illustrate the notation and MDO architectures.
- Suppose we want to design the wing of a business jet using low-fidelity analysis tools.
- Model the aerodynamics using a panel method
- Model the structure as a single beam using finite elements
Example: Aerostructural Problem Definition 2

Aerodynamic inputs: angle-of-attack ($\alpha$), wing twist distribution ($\gamma_i$)

Aerodynamic outputs: lift ($L$) and the induced drag ($D$).
Example: Aerostructural Problem Definition 3

- Structural inputs: thicknesses of the beam \( t_i \)
- Structural output: beam weight, which is added to a fixed weight to obtain the total weight \( W \), and the maximum stresses in each finite-element \( \sigma_i \).
- In this example, we want to maximize the range of the aircraft, as given by the Breguet range equation,

\[
f = \text{Range} = \frac{V}{c} \frac{L}{D} \ln \left( \frac{W_i}{W_f} \right).
\]

- The multidisciplinary analysis consists in the simultaneous solution of the following equations:

\[
\begin{align*}
R_1 &= 0 \Rightarrow \quad A\Gamma - v(u, \alpha) = 0 \\
R_2 &= 0 \Rightarrow \quad Ku - F(\Gamma) = 0 \\
R_3 &= 0 \Rightarrow \quad L(\Gamma) - W = 0
\end{align*}
\]
Example: Aerostructural Problem Definition 4

- The complete state vector is

\[
\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} \Gamma \\ u \\ \alpha \end{bmatrix}.
\]

- The angle of attack is considered a state variable here, and helps satisfy \( L = W \).

- The design variables are the wing sweep (\( \Lambda \)), structural thicknesses (\( t \)) and twist distribution (\( \gamma \)).

\[
x_0 = \Lambda
\]

\[
x = \begin{bmatrix} t \\ \gamma \end{bmatrix},
\]

- Sweep is a shared variable because changing the sweep has a direct effect on both the aerodynamic influence matrix and the stiffness matrix.
Example: Aerostructural Problem Definition 5

- The other two sets of design variables are local to the structures and aerodynamics, respectively.
- In later examples, we will see the options we have to optimize the wing in this example.
Multidisciplinary Analysis 1

- To find the coupled state of a multidisciplinary system we need to perform a multidisciplinary analysis — MDA.
- This is often done by repeating each disciplinary analysis until $y_i^t = y_i^r$ for all $i$.

---

**Input:** Design variables $x$

**Output:** Coupling variables, $y$

0: Initiate MDA iteration loop

repeat

1: Evaluate Analysis 1 and update $y_1(y_2, y_3)$
2: Evaluate Analysis 2 and update $y_2(y_1, y_3)$
3: Evaluate Analysis 3 and update $y_3(y_1, y_2)$

until $4 \rightarrow 1$: MDA has converged
The design structure matrix (DSM) was originally developed to visualize the interconnections between the various components of a system.

- **Original ordering**
- **Improved ordering**

> Fixed-point iteration, such as the Gauss–Seidel algorithm above converge slowly and sometimes do not converge at all.

> One way to improve the disciplines, is to reorder the sequence and possibly do some inner loops for more coupled clusters.
Extended Design Structure Matrix
A Unified Description of MDO Architectures
Motivation

• No comprehensive description of MDO architectures in a unified notation

• Often not enough detail when a given MDO architecture is described

• Flow diagrams are not standardized

• Flow diagrams do not provide as much information as they could — lack of information density

[Hayighat, Liu and Martins, 2009]

1. Multidisciplinary analysis
2. Response surface update
3. Concurrent subspace optimization
4. Concurrent multidisciplinary analyses
5. Response surface update
6. System level optimization

[ Martins and Lambe, Consortium Workshop, 2009]
minimize $f_0(x, y)$
with respect to $x, y^t, \bar{y}, y$
subject to $c_0(x, y) \geq 0$
$c_i(x_0, x_i, y_i) \geq 0$
$c^c_i = y^t_i - y_i = 0$
$\mathcal{R}_i(x_0, x_i, y^t_{j \neq i}, \bar{y}_i, y_i) = 0$

Convention: $x = [x_0^T, x_1^T, \ldots, x_N^T]^T$ and $y = [y_1^T, \ldots, y_N^T]^T$

Convention: $c_i, c^c_i$, and $\mathcal{R}_i$ exist for $i = 1, \ldots, N$

All architectures solve an equivalent reformulation of this problem
The $N^2$ Diagram and Design Structure Matrix

- Components on main diagonal, coupling data on off-diagonal nodes
- Component inputs in same column, component outputs in same row
- External inputs and outputs may also be included
Extending the DSM syntax: A Gauss–Seidel Multidisciplinary Analysis Example
A Jacobi Multidisciplinary Analysis Example

(no data) → 0, 2 → 1: MDA

1: \( y^t, y^t_2, y^t_3 \)

1: \( x_0, x_1 \)

1: \( x_0, x_2 \)

1: \( x_0, x_3 \)

1: Analysis 1

2: \( y_1 \)

2: \( y_1 \)

1: Analysis 2

2: \( y_2 \)

2: \( y_2 \)

1: Analysis 3

2: \( y_3 \)

2: \( y_3 \)
An Optimization Problem

- Follow sequence of numbers and thin black lines
- When number or index is repeated, procedures can be parallelized
- Close the loops
The XDSM (Extended Design Structure Matrix) is a tool used to visualize MDO processes. It is an extension of the classical Design Structure Matrix commonly used in systems engineering to describe the interfaces among components of a complex system. In a computational MDO context, the complex system in the MDO architecture, the components of the system are pieces of software (disciplinary analyses, optimization algorithms, surrogate models, etc.) used by the architecture, and the interfaces between components are the data exchanged by this software. Because the architecture also contains an algorithm defining the order in which the software is run, a numbering system and lines depicting the process are introduced in the diagram. In this way, we are able to capture all of the data and process flow of an architecture in a single diagram.

The full details of how to construct and interpret XDSMs are the subject of the paper cited in the footnote.1

For those interested in constructing XDSMs for their own work, see the attached files. We drew our diagrams using the TikZ package in LaTeX. The files contain the specific block and line styles, TikZ library imports, and formats that are common to all of our diagrams. We have also included some example diagrams and a how-to guide for the LaTeX files. Comments and suggestions are welcome.

A Python script for automatically generating XDSM tex sources has been added. This script contains a class to which components and dependencies can be added, and this class automatically writes a tex file that draws the diagonal and off-diagonal blocks, as well as data flow lines. Details can be found in the Python script.

Sequential Optimization vs. MDO
Example: Aerostructural Optimization — Sequential Design vs. MDO 1

- One commonly used approach to design is to perform a sequential “optimization” approach, which consists in optimizing each discipline in sequence:

1. For example, we could start by optimizing the aerodynamics,

   \[
   \begin{align*}
   \text{minimize} & \quad D(\alpha, \gamma_i) \\
   \text{w.r.t.} & \quad \alpha, \gamma_i \\
   \text{s.t.} & \quad L(\alpha, \gamma_i) = W
   \end{align*}
   \]

2. Once the aerodynamic optimization has converged, the twist distribution and the forces are fixed

3. Then we optimize the structure by minimizing weight subject to stress constraints at the maneuver condition, i.e.,

   \[
   \begin{align*}
   \text{minimize} & \quad W(t_i) \\
   \text{w.r.t.} & \quad t_i \\
   \text{s.t.} & \quad \sigma_j(t_i) \leq \sigma_{\text{yield}}
   \end{align*}
   \]
Example: Aerostructural Optimization — Sequential Design vs. MDO 2

4. Repeat until this sequence has converged.
Example: Aerostructural Optimization — Sequential Design vs. MDO

The MDO procedure differs from the sequential approach in that it considers all variables simultaneously.

\[
\begin{aligned}
\text{minimize} & \quad \text{Range} (\alpha, \gamma_i, t_i) \\
\text{w.r.t.} & \quad \alpha, \gamma_i, t_i \\
\text{s.t.} & \quad \sigma_{\text{yield}} - \sigma_j (t_i) \geq 0 \\
& \quad L (\alpha, \gamma_i) - W = 0
\end{aligned}
\]
Example: Aerostructural Optimization — Sequential Design vs. MDO
Example: Aerostructural Optimization — Sequential Design vs. MDO 5

[Chittick and Martins, SMO, 2008]
Monolithic MDO Architectures
Monolithic Architectures

- Monolithic architectures solve the MDO problem by casting it as single optimization problem.
- Distributed architectures, on the other hand, decompose the overall problem into smaller ones.
- Monolithic architectures include:
  - Multidisciplinary Feasible — MDF
  - Individual Discipline Feasible — IDF
  - Simultaneous Analysis and Design — SAND
  - All-At-Once — AAO
Multidisciplinary Feasible (MDF) 1

- The MDF architecture is the most intuitive for engineers.
- The optimization problem formulation is identical to the single discipline case, except the disciplinary analysis is replaced by an MDA

\[
\begin{align*}
\text{minimize} & \quad f_0 (x, y(x, y)) \\
\text{with respect to} & \quad x \\
\text{subject to} & \quad c_0 (x, y(x, y)) \geq 0 \\
& \quad c_i (x_0, x_i, y_i (x_0, x_i, y_j \neq i)) \geq 0 \quad \text{for} \quad i = 1, \ldots, N.
\end{align*}
\]
Multidisciplinary Feasible (MDF) 2

0, 7→1: Optimization

1, 5→2: MDA

2: Analysis 1

3: Analysis 2

4: Analysis 3

6: Functions

x*(0)

y^t,(0)

x, 7→1:

2 : x_0, x_1

3 : x_0, x_2

4 : x_0, x_3

6 : x

1, 5→2:

2 : y^t_2, y^t_3

3 : y^t_3

2 : Analysis 1

3 : y_1

4 : y_1

6 : y_1

3 : Analysis 2

4 : y_2

6 : y_2

4 : Analysis 3

6 : y_3

7 : f, c

5 : y_1

5 : y_2

5 : y_3

5 : Analysis 1

3 : Analysis 2

4 : Analysis 3

6: Functions

J.R.R.A. Martins
Multidisciplinary Feasible (MDF) 3

Advantages:
- Optimization problem is as small as it can be for a monolithic architecture
- Always returns a system design that satisfies the consistency constraints, even if the optimization process is terminated early — good from the practical engineering point of view

Disadvantages:
- Intermediate results do not necessarily satisfy the optimization constraints
- Developing the MDA procedure might be time consuming, if not already in place
- Gradients of the coupled system more challenging to compute (more in later section)
Example: Aerostructural Optimization with MDF

\[
\begin{align*}
\text{minimize} & \quad - R \\
\text{w.r.t.} & \quad \Lambda, \gamma, t \\
\text{s.t.} & \quad \sigma_{\text{yield}} - \sigma_i(u) \geq 0
\end{align*}
\]

where the aerostructural analysis is as before:

\[
\begin{align*}
A\Gamma - v(u, \alpha) &= 0 \\
K(t, \Lambda)u - F(\Gamma) &= 0 \\
L(\Gamma) - W(t) &= 0
\end{align*}
\]
Individual Discipline Feasible (IDF) 1

The IDF architecture decouples the MDA, adding consistency constraints, and giving the optimizer control of the coupling variables.

\[
\begin{align*}
\text{minimize} & \quad f_0 (x, y (x, y^t)) \\
\text{with respect to} & \quad x, y^t \\
\text{subject to} & \quad c_0 (x, y (x, y^t)) \geq 0 \\
& \quad c_i (x_0, x_i, y_i (x_0, x_i, y_{j \neq i}^t)) \geq 0 \quad \text{for } i = 1, \ldots, N \\
& \quad c_i^c = y_i^t - y_i (x_0, x_i, y_{j \neq i}^t) = 0 \quad \text{for } i = 1, \ldots, N.
\end{align*}
\]

- **Advantages:**
  - Optimizer typically converges the multidisciplinary feasibility better than fixed-point MDA iterations

- **Disadvantages:**
  - Problem is potentially much larger than MDF, depending on the number of coupling variables
  - Gradient computation can be costly
The large problem size can be mitigated to some extent by careful selection of the disciplinary variable partitions or aggregation of the coupling variables to reduce information transfer between disciplines.

\[ x^{(0)}, y^t(0) \]

0,3→1: Optimization

\[ x^*, y^*_i \]

1: \( x_0, x_i, y^t_j \neq i \)

2: \( x, y^t \)

1: Analysis \( i \)

2: Functions

3: \( f, c, c^c \)
Example: Aerostructural Optimization Using IDF

\[
\text{minimize} \quad -R \\
\text{w.r.t.} \quad \Lambda, \gamma, t, \Gamma^t, \alpha^t, u^t \\
\text{s.t.} \quad \sigma_{\text{yield}} - \sigma_i \geq 0 \\
\quad \Gamma^t - \Gamma = 0 \\
\quad \alpha^t - \alpha = 0 \\
\quad u^t - u = 0
\]
Simultaneous Analysis and Design (SAND) 1

- SAND makes no distinction between disciplines, and can also be applied to single discipline problems.

- The governing equations are constraints at the optimizer level.

\[
\begin{align*}
\text{minimize} \quad & f_0 (x, y) \\
\text{with respect to} \quad & x, y, \bar{y} \\
\text{subject to} \quad & c_0 (x, y) \geq 0 \\
& c_i (x_0, x_i, y_i) \geq 0 \quad \text{for} \quad i = 1, \ldots, N \\
& R_i (x_0, x_i, y, \bar{y}_i) = 0 \quad \text{for} \quad i = 1, \ldots, N.
\end{align*}
\]

- Advantages:
  - If implemented well, can be the most efficient architecture

- Disadvantages:
  - Intermediate results do not even satisfy the governing equations
  - Difficult or impossible to implement for “black-box” components
Simultaneous Analysis and Design (SAND) 2

$x^{(0)}, y^{(0)}, \bar{y}^{(0)}$

$x^*, y^*$

0, 2 $\rightarrow$ 1: Optimization

1: $x, y$

1: $x_0, x_i, y, \bar{y}_i$

2: $f, c$

2: $R_i$

1: Functions

1: Residual $i$
Aerostructural Optimization Using SAND 1

\[
\text{minimize} \quad -R \\
\text{w.r.t.} \quad \Lambda, \gamma, t, \Gamma, \alpha, u \\
\text{s.t.} \quad \sigma_{\text{yield}} - \sigma_i(u) \geq 0 \\
A\Gamma = v(u, \alpha) \\
K(t)u = f(\Gamma) \\
L(\Gamma) - W(t) = 0
\]
The All-at-Once (AAO) Problem Statement 1

- AAO is not strictly an architecture, as it is not practical to solve a problem of this form: the consistency constraints are linear and can be eliminated, leading to SAND.
- Some inconsistency in the name, in the literature
- We present AAO for completeness, and to relate this to the other monolithic architectures.

\[
\begin{align*}
\text{minimize} & \quad f_0(x, y) + \sum_{i=1}^{N} f_i(x_0, x_i, y_i) \\
\text{with respect to} & \quad x, y^t, y, \bar{y} \\
\text{subject to} & \quad c_0(x, y) \geq 0 \\
& \quad c_i(x_0, x_i, y_i) \geq 0 \quad \text{for } i = 1, \ldots, N \\
& \quad c_i^c = y_i^t - y_i = 0 \quad \text{for } i = 1, \ldots, N \\
& \quad R_i(x_0, x_i, y_j \neq i, \bar{y}_i, y_i) = 0 \quad \text{for } i = 1, \ldots, N.
\end{align*}
\]
The All-at-Once (AAO) Problem Statement 2

- As we can see, it includes all the constraints that other monolithic architectures eliminated.
The All-at-Once (AAO) Problem Statement

minimize \( f_0(x, y) + \sum_{i=1}^{N} f_i(x_0, x_i, y_i) \)
with respect to \( x, y^t, y, \bar{y} \)
subject to
\begin{align*}
  c_0(x, y) &\geq 0 \\
  c_i(x_0, x_i, y_i) &\geq 0 \quad \text{for } i = 1, \ldots, N \\
  c^c_i = y_i^t - y_i &= 0 \quad \text{for } i = 1, \ldots, N \\
  R_i(x_0, x_i, y_{j \neq i}^t, \bar{y}_i, y_i) &= 0 \quad \text{for } i = 1, \ldots, N
\end{align*}

Distributed MDO Architectures
Distributed Architectures

- Monolithic MDO architectures solve a single optimization problem
- Distributed MDO architectures decompose the original problem into multiple optimization problems
- Some problems have a special structure and can be efficiently decomposed, but that is usually not the case
- In reality, the primary motivation for decomposing the MDO problem comes from the structure of the engineering design environment
- Typical industrial practice involves breaking up the design of a large system and distributing aspects of that design to specific engineering groups.
- These groups may be geographically distributed and may only communicate infrequently.
- In addition, these groups typically like to retain control of their own design procedures and make use of in-house expertise
### Classification of MDO Architectures

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CO</td>
<td>Copies of the shared variables are created for each discipline, together with corresponding consistency constraints. Discipline subproblems minimize difference between the copies of shared and local variables subject to local constraints. A surrogate model of the local optimum with respect to the shared variables is maintained. Then, system subproblem minimizes objective subject to shared constraints subject to consistency constraints.</td>
</tr>
<tr>
<td>BLISS-2000</td>
<td>Discipline subproblems minimize the objective with respect to local variables subject to local constraints. A surrogate model of the local optimum with respect to the shared variables is maintained. Then, system subproblem minimizes objective subject to shared design and consistency constraints, considering the disciplinary preferences.</td>
</tr>
<tr>
<td>QSD</td>
<td>Each discipline is assigned a “budget” for a local objective and the discipline subproblems maximize the margin in their local constraints and the budgeted objective. System subproblem minimizes a shared objective and the budgets of each discipline subject to shared design constraints and positivity of the margin in each discipline.</td>
</tr>
<tr>
<td>IPD/EPD</td>
<td>Applicable to MDO problems with no shared objectives or constraints. Like ATC, copies of shared variables are used for every discipline subproblem and the consistency constraints are relaxed with a penalty function. Unlike ATC, the simple structure of the disciplinary subproblems is exploited to compute post-optimality sensitivities to guide the system subproblem.</td>
</tr>
<tr>
<td>ECO</td>
<td>As in CO, copies of the shared design variables are used. Disciplinary subproblems minimize quadratic approximations of the objective subject to local constraints and linear models of nonlocal constraints. Shared variables are determined by the system subproblem, which minimizes the total violation of all consistency constraints.</td>
</tr>
<tr>
<td>ATC</td>
<td>Copies of the shared variables are used in discipline subproblems together with the corresponding consistency constraints. These consistency constraints are relaxed using a penalty function. System and discipline subproblems solve their respective relaxed problem independently. Penalty weights are increased until the desired consistency is achieved.</td>
</tr>
<tr>
<td>BLISS</td>
<td>Coupled derivatives of the multidisciplinary analysis are used to construct linear subproblems for each discipline with respect to local design variables. Post-optimality derivatives from the solutions of these subproblems are computed to form the system linear subproblem, which is solved with respect to shared design variables.</td>
</tr>
<tr>
<td>CSSO</td>
<td>In system subproblem, disciplinary analyses are replaced by surrogate models. Discipline subproblems are solved using surrogates for the other disciplines, and the solutions from these discipline subproblems are used to update the surrogate models.</td>
</tr>
<tr>
<td>MDOIS</td>
<td>Applicable to MDO problems with no shared objectives, constraints, or design variables. Discipline subproblems are solved independently assuming fixed coupling variables, and then a multidisciplinary analysis is performed to update the coupling.</td>
</tr>
<tr>
<td>ASO</td>
<td>System subproblem is like that of MDF, but some disciplines solve a discipline optimization subproblem within the multidisciplinary analysis with respect to local variables subject to local constraints. Coupled post-optimality derivatives from the discipline subproblems are computed to guide the system subproblem.</td>
</tr>
</tbody>
</table>

The figure illustrates the classification of MDO architectures, highlighting the differences between monolithic and distributed architectures. The Monolithic architectures (Monolithic IDF and Monolithic MDF) centralize all optimization processes, while Distributed architectures (Distributed IDF and Distributed MDF) distribute the optimization tasks among disciplines. The Penalty architecture integrates shared objectives or constraints, while the CO and BLISS architectures manage shared variables and constraints through different methods. The distributed architectures (Distributed IDF and Distributed MDF) further elaborate on how shared variables are managed and updated across disciplines.
Concurrent Subspace Optimization (CSSO) 1

The CSSO system subproblem is given by

\[
\begin{align*}
\text{minimize} & \quad f_0 (x, \tilde{y} (x, \tilde{y})) \\
\text{with respect to} & \quad x \\
\text{subject to} & \quad c_0 (x, \tilde{y} (x, \tilde{y})) \geq 0 \\
& \quad c_i (x_0, x_i, \tilde{y}_i (x_0, x_i, \tilde{y}_{j \neq i})) \geq 0 \text{ for } i = 1, \ldots, N
\end{align*}
\]

and the discipline \( i \) subproblem is given by

\[
\begin{align*}
\text{minimize} & \quad f_0 (x, y_i (x_i, \tilde{y}_{j \neq i}), \tilde{y}_{j \neq i}) \\
\text{with respect to} & \quad x_0, x_i \\
\text{subject to} & \quad c_0 (x, \tilde{y} (x, \tilde{y})) \geq 0 \\
& \quad c_i (x_0, x_i, y_i (x_0, x_i, \tilde{y}_{j \neq i})) \geq 0 \\
& \quad c_j (x_0, \tilde{y}_j (x_0, \tilde{y})) \geq 0 \quad \text{for } j = 1, \ldots, N \ j \neq i.
\end{align*}
\]
Concurrent Subspace Optimization (CSSO) 2
CSSO Algorithm

**Input:** Initial design variables $x$

**Output:** Optimal variables $x^*$, objective function $f^*$, and constraint values $c^*$

0: Initiate main CSSO iteration

```
repeat
  1: Initiate a design of experiments (DOE) to generate design points
  for Each DOE point do
    2: Initiate an MDA that uses exact disciplinary information
    repeat
      3: Evaluate discipline analyses
      4: Update coupling variables $y$
      until 4 $\rightarrow$ 3: MDA has converged
      5: Update the disciplinary surrogate models with the latest design
    end for
    6 $\rightarrow$ 2
  7: Initiate independent disciplinary optimizations (in parallel)
  for Each discipline $i$ do
    repeat
      8: Initiate an MDA with exact coupling variables for discipline $i$ and approximate coupling variables for the other disciplines
      repeat
        9: Evaluate discipline $i$ outputs $y_i$, and surrogate models for the other disciplines, $\tilde{y}_{j\neq i}$
        until 10 $\rightarrow$ 9: MDA has converged
        11: Compute objective $f_0$ and constraint functions $c$ using current data
      until 12 $\rightarrow$ 8: Disciplinary optimization $i$ has converged
    end for
  13: Initiate a DOE that uses the subproblem solutions as sample points
  for Each subproblem solution $s$ do
    14: Initiate an MDA that uses exact disciplinary information
    repeat
      15: Evaluate discipline analyses.
      until 16 $\rightarrow$ 15 MDA has converged
      17: Update the disciplinary surrogate models with the newest design
    end for
    18 $\rightarrow$ 14
  19: Initiate system-level optimization
  repeat
    20: Initiate an MDA that uses only surrogate model information
    repeat
      21: Evaluate disciplinary surrogate models
      until 22 $\rightarrow$ 21: MDA has converged
      23: Compute objective $f_0$, and constraint function values $c$
      until 24 $\rightarrow$ 20: System level problem has converged
    until 25 $\rightarrow$ 1: CSSO has converged
```

J.R.R.A. Martins
Collaborative Optimization (CO) 1

The CO2 system subproblem is given by:

\[
\begin{align*}
\text{minimize} & \quad f_0 \left( x_0, \hat{x}_1, \ldots, \hat{x}_N, y^t \right) \\
\text{with respect to} & \quad x_0, \hat{x}_1, \ldots, \hat{x}_N, y^t \\
\text{subject to} & \quad c_0 \left( x_0, \hat{x}_1, \ldots, \hat{x}_N, y^t \right) \geq 0 \\
& \quad J^*_i = ||\hat{x}_{0i} - x_0||^2_2 + ||\hat{x}_i - x_i||^2_2 + ||y^t_i - y_i \left( \hat{x}_{0i}, x_i, y^{t\neq i}_j \right)||^2_2 = 0 \quad \text{for} \quad i = 1, \ldots, N
\end{align*}
\]

where \( \hat{x}_{0i} \) are duplicates of the global design variables passed to (and manipulated by) discipline \( i \) and \( \hat{x}_i \) are duplicates of the local design variables passed to the system subproblem.

The discipline \( i \) subproblem in both CO1 and CO2 is

\[
\begin{align*}
\text{minimize} & \quad J_i \left( \hat{x}_{0i}, x_i, y_i \left( \hat{x}_{0i}, x_i, y^{t\neq i}_j \right) \right) \\
\text{with respect to} & \quad \hat{x}_{0i}, x_i \\
\text{subject to} & \quad c_i \left( \hat{x}_{0i}, x_i, y_i \left( \hat{x}_{0i}, x_i, y^{t\neq i}_j \right) \right) \geq 0.
\end{align*}
\]
Collaborative Optimization (CO) 2

0, 2→1: System Optimization

1: $x_0, \hat{x}_1...N, y^t$

1: System Functions

1.0, 1.3→1.1: Optimization $i$

1.1: $\hat{x}_{0i}, x_i$

1.2: $\hat{x}_{0i}, x_i$

1.1: Analysis $i$

1.2: $y_i$

1.2: $y_{j\neq i}^t$

2: $f_0, c_0$

2: $f_i, c_i, J_i$

2: $J_i^*$

2: $x_{0i}, x_i^*$
CO Algorithm 1

**Input:** Initial design variables $x$

**Output:** Optimal variables $x^*$, objective function $f^*$, and constraint values $c^*$

0: Initiate system optimization iteration

repeat

1: Compute system subproblem objectives and constraints

for Each discipline $i$ (in parallel) do

1.0: Initiate disciplinary subproblem optimization

repeat

1.1: Evaluate disciplinary analysis

1.2: Compute disciplinary subproblem objective and constraints

1.3: Compute new disciplinary subproblem design point and $J_i$

until 1.3 $\rightarrow$ 1.1: Optimization $i$ has converged

end for

2: Compute a new system subproblem design point

until 2 $\rightarrow$ 1: System optimization has converged
Aerostructural Optimization Using CO 1

System-level problem:

\[
\begin{align*}
\text{minimize} & \quad - R \\
\text{w.r.t.} & \quad \Lambda^t, \Gamma^t, \alpha^t, u^t, W^t \\
\text{s.t.} & \quad J_1^* \leq 10^{-6} \\
& \quad J_2^* \leq 10^{-6}
\end{align*}
\]

Aerodynamics subproblem:

\[
\begin{align*}
\text{minimize} & \quad J_1 = \left(1 - \frac{\Lambda}{\Lambda^t}\right)^2 + \sum \left(1 - \frac{\Gamma_i}{\Gamma_i^t}\right)^2 + \left(1 - \frac{\alpha}{\alpha^t}\right)^2 + \left(1 - \frac{W}{W^t}\right)^2 \\
\text{w.r.t.} & \quad \Lambda, \gamma, \alpha \\
\text{s.t.} & \quad L - W = 0
\end{align*}
\]
Aerostructural Optimization Using CO 2

Structures subproblem:

\[
\text{minimize } J_2 = \left(1 - \frac{\Lambda}{\Lambda^t}\right)^2 + \sum \left(1 - \frac{u_i}{u_i^t}\right)^2 \\
\text{w.r.t. } \Lambda, t \\
\text{s.t. } \sigma_{\text{yield}} - \sigma_i \geq 0
\]
Bilevel Integrated System Synthesis (BLISS) 1

The system level subproblem is formulated as

\[
\text{minimize} \quad (f_0^*)_0 + \left( \frac{df_0^*}{dx_0} \right) \Delta x_0 \\
\text{with respect to} \quad \Delta x_0 \\
\text{subject to} \quad (c_0^*)_0 + \left( \frac{dc_0^*}{dx_0} \right) \Delta x_0 \geq 0 \\
(c_i^*)_0 + \left( \frac{dc_i^*}{dx_0} \right) \Delta x_0 \geq 0 \quad \text{for} \quad i = 1, \ldots, N \\
\Delta x_{0L} \leq \Delta x_0 \leq \Delta x_{0U}.
\]
Bilevel Integrated System Synthesis (BLISS) 2

The discipline $i$ subproblem is given by

\[
\begin{align*}
\text{minimize} \quad & (f_0)_0 + \left( \frac{df_0}{dx_i} \right) \Delta x_i \\
\text{with respect to} \quad & \Delta x_i \\
\text{subject to} \quad & (c_0)_0 + \left( \frac{dc_0}{dx_i} \right) \Delta x_i \geq 0 \\
& (c_i)_0 + \left( \frac{dc_i}{dx_i} \right) \Delta x_i \geq 0 \\
& \Delta x_{iL} \leq \Delta x_i \leq \Delta x_{iU}.
\end{align*}
\]

Note the extra set of constraints in both system and discipline subproblems denoting the design variables bounds.
Bilevel Integrated System Synthesis (BLISS) 3
BLISS Algorithm

**Input:** Initial design variables $x$

**Output:** Optimal variables $x^*$, objective function $f^*$, and constraint values $c^*$

0: Initiate system optimization

repeat

1: Initiate MDA

repeat

2: Evaluate discipline analyses
3: Update coupling variables

until 3 $\rightarrow$ 2: MDA has converged
4: Initiate parallel discipline optimizations

for Each discipline $i$ do

5: Evaluate discipline analysis
6: Compute objective and constraint function values and derivatives with respect to local design variables
7: Compute the optimal solutions for the disciplinary subproblem

end for

8: Initiate system optimization
9: Compute objective and constraint function values and derivatives with respect to shared design variables using post-optimality analysis
10: Compute optimal solution to system subproblem

until 11 $\rightarrow$ 1: System optimization has converged
Analytical Target Cascading (ATC) 1

The ATC system subproblem is given by

\[
\text{minimize } \quad f_0 (x, y^t) + \sum_{i=1}^{N} \Phi_i (\hat{x}_{0i} - x_0, y_i^t - y_i (x_0, x_i, y^t)) + \\
\Phi_0 (c_0 (x, y^t))
\]

with respect to \( x_0, y^t \),

where \( \Phi_0 \) is a penalty relaxation of the global design constraints and \( \Phi_i \) is a penalty relaxation of the discipline \( i \) consistency constraints. The \( i^{th} \) discipline subproblem is:

\[
\text{minimize } \quad f_0 (\hat{x}_{0i}, x_i, y_i (\hat{x}_{0i}, x_i, y_{j \neq i}^t), y_{j \neq i}^t) + f_i (\hat{x}_{0i}, x_i, y_i (\hat{x}_{0i}, x_i, y_{j \neq i}^t)) + \\
\Phi_i (y_i^t - y_i (\hat{x}_{0i}, x_i, y_{j \neq i}^t), \hat{x}_{0i} - x_0) + \\
\Phi_0 (c_0 (\hat{x}_{0i}, x_i, y_i (\hat{x}_{0i}, x_i, y_{j \neq i}^t), y_{j \neq i}^t))
\]

with respect to \( \hat{x}_{0i}, x_i \)

subject to \( c_i (\hat{x}_{0i}, x_i, y_i (\hat{x}_{0i}, x_i, y_{j \neq i}^t)) \geq 0. \)
Analytical Target Cascading (ATC) 2

0,8→1: w update

5,7→6: System Optimization

5: System and Penalty Functions

7: $f_0, \Phi_0...N$

6: $x_0^i, x_i$

6: $w$

3: $w_i$

2: $y^i_j \neq i$

3: $x_0^i, x_i$

3: $x_0, y^t$

2: $x_0, y^t$

2: $x_0^i, x_i$

3: $x_0, y^t$

6: $x_0, y^t$

6: $x_0^i, x_i$

6: $x_0^i, x_i$

2: $y^i_i$

3: $y_i$

3: $y_i$

4: $f_i, c_i, \Phi_0, \Phi_i$
ATC Algorithm

**Input:** Initial design variables $x$

**Output:** Optimal variables $x^*$, objective function $f^*$, and constraint values $c^*$

0: Initiate main ATC iteration

repeat
  for Each discipline $i$ do
    1: Initiate discipline optimizer
    repeat
      2: Evaluate disciplinary analysis
      3: Compute discipline objective and constraint functions and penalty function values
      4: Update discipline design variables
    until $4 \rightarrow 2$: Discipline optimization has converged
  end for
  5: Initiate system optimizer
  repeat
    6: Compute system objective, constraints, and all penalty functions
    7: Update system design variables and coupling targets.
  until $7 \rightarrow 6$: System optimization has converged
  8: Update penalty weights
until $8 \rightarrow 1$: Penalty weights are large enough
Asymmetric Subspace Optimization (ASO) 1

The system subproblem in ASO is

$$\text{minimize} \quad f_0 (x, y (x, y)) + \sum_k f_k (x_0, x_k, y_k (x_0, x_k, y_{j \neq k}))$$

with respect to \( x_0, x_k \)

subject to \( c_0 (x, y (x, y)) \geq 0 \)

\( c_k (x_0, x_k, y_k (x_0, x_k, y_{j \neq k})) \geq 0 \)

for all \( k \),

where subscript \( k \) denotes disciplinary information that remains outside of the MDA. The disciplinary problem for discipline \( i \), which is resolved inside the MDA, is

$$\text{minimize} \quad f_0 (x, y (x, y)) + f_i (x_0, x_i, y_i (x_0, x_i, y_{j \neq i}))$$

with respect to \( x_i \)

subject to \( c_i (x_0, x_i, y_i (x_0, x_i, y_{j \neq i})) \geq 0 \).
Asymmetric Subspace Optimization (ASO) 2
ASO Algorithm

**Input:** Initial design variables \( x \)

**Output:** Optimal variables \( x^* \), objective function \( f^* \), and constraint values \( c^* \)

0: Initiate system optimization

repeat

1: Initiate MDA

repeat

2: Evaluate Analysis 1
3: Evaluate Analysis 2
4: Initiate optimization of Discipline 3

repeat

5: Evaluate Analysis 3
6: Compute discipline 3 objectives and constraints
7: Update local design variables

until 7 \( \rightarrow \) 5: Discipline 3 optimization has converged
8: Update coupling variables

until 8 \( \rightarrow \) 2 MDA has converged

9: Compute objective and constraint function values for all disciplines 1 and 2

10: Update design variables

until 10 \( \rightarrow \) 1: System optimization has converged
Classification of MDO Architectures

**Monolithic**
- **AAO:** Remove $c_i^e, y^i$
- **SAND:** Remove $R, y, \bar{y}$

**Distributed IDF**
- **CO:** Copies of the shared variables are created for each discipline, together with corresponding consistency constraints. Discipline subproblems minimize difference between the copies of shared and local variables subject to local constraints. A surrogate model of the local optimum with respect to the shared variables is maintained. Then, system subproblem minimizes objective subject to shared constraints subject to consistency constraints.

**BLISS-2000:** Discipline subproblems minimize the objective with respect to local variables subject to local constraints. A surrogate model of the local optimum with respect to the shared variables is maintained. Then, system subproblem minimizes objective subject to shared constraints subject to consistency constraints.

**QSD:** Each discipline is assigned a “budget” for a local objective and the discipline problems maximize the margin in their local constraints and the budgeted objective. System subproblem minimizes a shared objective and the budgets of each discipline subject to shared design constraints and positivity of the margin in each discipline.

**Penalty**
- **ATC:** Copies of the shared variables are used in discipline subproblems together with the corresponding consistency constraints. These consistency constraints are relaxed using a penalty function. System and discipline subproblems solve their respective relaxed problem independently. Penalty weights are increased until the desired consistency is achieved.

**IPD/EPD:** Applicable to MDO problems with no shared objectives or constraints. Like ATC, copies of shared variables are used for every discipline subproblem and the consistency constraints are relaxed with a penalty function. Unlike ATC, the simple structure of the disciplinary subproblems is exploited to compute post-optimality sensitivities to guide the system subproblem.

**ECO:** As in CO, copies of the shared design variables are used. Disciplinary subproblems minimize quadratic approximations of the objective subject to local constraints and linear models of nonlocal constraints. Shared variables are determined by the system subproblem, which minimizes the total violation of all consistency constraints.

**Distributed MDF**
- **CSSO:** In system subproblem, disciplinary analyses are replaced by surrogate models. Discipline subproblems are solved using surrogates for the other disciplines, and the solutions from these discipline subproblems are used to update the surrogate models.

**BLISS:** Coupled derivatives of the multidisciplinary analysis are used to construct linear subproblems for each discipline with respect to local design variables. Post-optimality derivatives from the solutions of these subproblems are computed to form the system linear subproblem, which is solved with respect to shared design variables.

**MDOIS:** Applicable to MDO problems with no shared objectives, constraints, or design variables. Discipline subproblems are solved independently assuming fixed coupling variables, and then a multidisciplinary analysis is performed to update the coupling.

**ASO:** System subproblem is like that of MDF, but some disciplines solve a discipline optimization subproblem within the multidisciplinary analysis with respect to local variables subject to local constraints. Coupled post-optimality derivatives from the discipline subproblems are computed to guide the system subproblem.

Example: A Framework for Automatic Implementation of MDO 2

[ Martins et al., ACM TOMS, 2009 ]
[ Tedford and Martins, Optimization and Engineering, 2010 ]
Computing derivatives: review and unification

[Martins and Hwang, AIAAJ, 2013]
What’s in a name?

- **Sensitivity analysis**: Includes much more than derivatives of functions and numerical models

- **Sensitivity derivative**: Somewhat redundant?

- **Design sensitivities**: Acceptable term

- **Derivative**: Matches the scope of this talk most closely

- **Gradient/Jacobian**: This vector/matrix of derivatives is what we need for optimization

## Scope

- First order derivatives

- Deterministic numerical models
Applications of Derivatives

• Numerical optimization
  ‣ For gradient-based optimization, need the gradient of the objective and constraints to iterate and satisfy the KKT optimality conditions
  ‣ Only viable option for problems with large numbers of design variables

• Construction of linear approximations
• Gradient-enhanced surrogate models
• Newton-type methods
• Functional analysis
• Parameter estimation
• Aircraft stability derivatives
Computational Cost vs. Number of Variables

![Graph showing computational cost vs. number of variables.](image)

The figure illustrates the cost of gradient evaluation for first-order finite differencing and the coupled adjoint method versus the number of design variables. The normalized time is shown on the y-axis, and the number of design variables on the x-axis.

Figure 9: Gradient evaluation cost for first-order finite differencing and the coupled adjoint method versus number of design variables; one unit of normalized time corresponds to one aerostructural solution for each geometric design variable. Nevertheless, each additional design variable requires only 0.005% of the aerostructural solution time.

It is worth comparing the current results with the previous work of Martins et al. \[31\]. In that work, the coupled adjoint cost was found to scale with the number of design variables according to \[3^{0.4} + 0.2^{0.1} N_x\]. Since the constant term in the equation includes the aerostructural solution, the coupled adjoint solution had a baseline cost of 2.4. The present method scales according to \[1.67 + 0.23N_x\], as indicated in Figure 9. This corresponds to a baseline cost for the coupled adjoint of 0.15, i.e., a 72% reduction relative to the previous implementation. This is primarily due to the elimination of the finite differencing that was used to compute the off-diagonal coupled adjoint terms. This improvement is even more significant in absolute terms because the aerostructural solution of the new implementation is also much more efficient. Additionally, the slope in the dependency on the number of design variables has been reduced by over two orders of magnitude. This is achieved by eliminating the use of finite-difference derivatives in the total-derivative equation (15).

We have shown that the new implementation of the coupled adjoint method exhibits extremely good design-variable scaling. The coupled computational cost can be considered practically independent of the number of design variables, and it is now feasible to compute gradients with respect to 35 of 42 American Institute of Aeronautics and Astronautics

Objectives

- Review all methods for computing derivatives of multidisciplinary systems
- Unify the theory behind these methods
- Create opportunities for new insights

\[ \frac{\partial C}{\partial v} \frac{dv}{dc} = I = \frac{\partial C^T}{\partial v} \frac{dv^T}{dc}. \]
Methods for Computing Derivatives

- Finite differences
- Complex step
- Symbolic differentiation
- Automatic differentiation: forward and reverse
- Analytic methods: direct and adjoint
- Coupled derivatives of multidisciplinary systems

Figure 1: Classification of methods for derivative computations: the differentiation methods are the building blocks for other methods, each of which considers a different level of decomposition.

Finite-difference formulas are derived from combining Taylor series expansions. Using the right combinations of these expansions, it is possible to obtain finite-difference formulas that estimate an arbitrary order derivative with any required order truncation error. The simplest finite-difference formula can be directly derived from one Taylor series expansion, yielding

\[
\frac{df}{dx} \bigg|_j = \frac{f(x + e_j h) - f(x)}{h} + O(h)
\]

which is directly related to the definition of derivative. Note that in general there are multiple functions of interest, and thus \(f\) can be a vector that includes all the outputs of a given component. The application of this formula requires the evaluation of a component at the reference point \(x\), and one perturbed point \(x + e_j h\), and yields one column of the Jacobian (1). Each additional column requires an additional evaluation of the component. Hence, the cost of computing the complete Jacobian is proportional to the number of input variables of interest, \(n_x\).

Finite-difference methods are widely used to compute derivatives due to their simplicity and the fact that they can be implemented even when a given component is a black box. Most gradient-based optimization algorithms perform finite-differences by default when the user does not provide the required gradients.

When it comes to accuracy, we can see from the forward-difference formula (2) that the truncation error is proportional to the magnitude of the perturbation, \(h\). Thus it is desirable to decrease \(h\) as much as possible. The problem with decreasing \(h\) is that the perturbed value of the functions of interest will approach the reference values. When using finite-precision arithmetic, this leads to subtractive cancellation: a loss of significant digits in the subtraction operation. In the extreme case, when \(h\) is small enough, all digits of the perturbed functions will match the reference values, yielding zero for the derivatives. Given the opposite trends exhibited by the subtractive cancellation error and truncation error, for each \(x\) there is a best \(h\) that minimizes the overall error.

Due to their flexibility, finite-difference formulas can always be used to compute derivatives, at any level of nesting. They can be used to compute derivatives of a single function, composite functions, iterative functions or any system with multiply nested components.

B. Complex Step

The complex-step derivative approximation, strangely enough, computes derivatives of real functions using complex variables. This method originated with the work of Lyness and Moler [20] and Lyness [21]. They developed several methods that made use of complex variables, including a reliable method for calculating the \(n\)th derivative of an analytic function. However, only later was this theory rediscovered by Squire and Trapp [22], who derived a simple formula for estimating the first derivative.

The complex-step derivative approximation, like finite-difference formulas, can also be derived using a Taylor series expansion. Rather than using a real step \(h\), we now use a pure imaginary step, \(ih\). If \(f\) is a real function in real
When a component is just a series of explicit functions, we can consider the component itself to be an explicit composite function. In cases where the computation of the outputs requires iteration, it is helpful to denote the computation as a vector of residual equations,

\[ r = R(v) = 0 \tag{8} \]

where the algorithm changes certain components of \( v \) until all the residuals converge to zero (or in practice, to within a small specified tolerance). The subset of \( v \) that is iterated to achieve the solution of these equations are called the state variables.

To relate these concepts to the usual conventions in sensitivity analysis, we now separate the subsets in \( v \) into independent variables \( x \), state variables \( y \) and quantities of interest, \( f \). Note that these do not necessarily correspond exactly to the component inputs, intermediate variables and outputs, respectively. Using this notation, we can write the residual equations as,

\[ r = R(x, y(x)) = 0 \tag{9} \]

where \( y(x) \) denotes the fact that \( y \) depends implicitly on \( x \) through the solution of the residual equations (9). It is the solution of these equations that completely determines \( y \) for a given \( x \). The functions of interest (usually included in the set of component outputs) also have the same type of variable dependence in the general case, i.e.,

\[ f = F(x, y(x)) \tag{10} \]

When we compute the values \( f \), we assume that the state variables \( y \) have already been determined by the solution of the residual equations (9). The dependencies involved in the computation of the functions of interest are represented in Figure 3. For the purposes of this paper, we are ultimately interested in the total derivatives of quantities \( f \) with respect to \( x \).
Finite Differences

\[
f(x + e_j h) = f(x) + h \frac{df}{dx_j} + \frac{h^2}{2} \frac{d^2 f}{dx_j^2} + \ldots \quad \Rightarrow
\]

\[
\frac{df}{dx_j} = \frac{f(x + e_j h) - f(x)}{h} + O(h)
\]

- Want to decrease truncation error by decreasing the step, but...
- Subtractive cancellation becomes worse as step decreases
- Require \( n_x \) evaluations of \( f \)

\[
\begin{bmatrix}
\frac{df_1}{dx_1} & \ldots & \frac{df_1}{dx_{nx}} \\
\vdots & \ddots & \vdots \\
\frac{df_{nf}}{dx_1} & \ldots & \frac{df_{nf}}{dx_{nx}}
\end{bmatrix}
\]

\( n_f \times n_x \)
Finite Differences

\[ f(x + e_jh) = f(x) + h \frac{df}{dx_j} + \frac{h^2}{2} \frac{d^2f}{dx^2_j} + \ldots \]

\[ \frac{df}{dx_j} = \left( \frac{f(x + e_jh) - f(x)}{h} \right) + \mathcal{O}(h) \]

\[
\begin{align*}
  f(x + h) & = 1.234567890123431 \\
  f(x) & = 1.234567890123456 \\
  \Delta f & = -0.0000000000000025
\end{align*}
\]
Complex Step

\[
f(x + ih e_j) = f(x) + ih \frac{df}{dx_j} - \frac{h^2}{2} \frac{d^2 f}{dx_j^2} - \frac{ih^3}{6} \frac{d^3 f}{dx_j^3} + \ldots
\]

\[
\frac{df}{dx_j} = \frac{\text{Im}[f(x + ih e_j)]}{h} + O(h^2)
\]

- No subtractive cancellation
- Precision of derivative matches that of \( f \)
- Flexible implementation

[ Martins, et al., ACM TOMS, 2003]
Consider a function, $f = u + iv$, of the complex variable, $z = x + iy$. If $f$ is analytic the Cauchy–Riemann equations apply, i.e.,

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y},$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x}.$$

We can use the definition of a derivative in the right hand side of the first Cauchy–Riemann to get

$$\frac{\partial u}{\partial x} = \lim_{h \to 0} \frac{v(x + i(y + h)) - v(x + iy)}{h}$$

where $h$ is a small real number.

[ Martins, et al., ACM TOMS, 2003 ]
Complex Step: Another Derivation

- Since the functions are real functions of a real variable, \( y = 0, u(x) = f(x) \) and \( v(x) = 0 \) and we can write,

\[
\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{\text{Im} [f (x + ih)]}{h}.
\]

- For a small discrete \( h \), this can be approximated by,

\[
\frac{\partial f}{\partial x} \approx \frac{\text{Im} [f (x + ih)]}{h}.
\]

[ Martins, et al., ACM TOMS, 2003 ]
Complex Step: Another Derivation

\[
\frac{\partial F}{\partial x} \approx \frac{F(x + h) - F(x)}{h}
\]

\[
\frac{\partial F}{\partial x} \approx \frac{\text{Im}[F(x + ih)] - \text{Im}[F(x)]}{\text{Im}[ih]}
\]

\[
\Rightarrow \frac{\partial F}{\partial x} \approx \frac{\text{Im}[F(x + ih)]}{h}
\]

[Martins, et al., ACM TOMS, 2003]
Example: The Complex-Step Method Applied to a Simple Function 2

Relative error of the derivative vs. decreasing step size

[ Martins, et al., ACM TOMS, 2003 ]
Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Box

Thinking Inside the Bo
Algorithm, Variables, and Functions

Consider the sequence of all the variables and functions in an algorithm

\[
v_i = V_i(v_1, v_2, \ldots, v_{i-1}), \quad i = 1, \ldots, n
\]

Assume a given variable depends only on previous one: all loops must be unrolled

The partial derivative of any of these functions wrt to any other variable is

\[
\frac{\partial V_i}{\partial v_j} = \frac{V_i(v_1, \ldots, v_{j-1}, v_j + h, v_{j+1}, \ldots, v_{i-1}) - V_i(\cdot)}{h}
\]
Residual Functions and State Variables

Computational model can be represented as a set of residuals of the governing equations:

\[ r = R(x, Y(x)) = 0 \]

The function of interest is:

\[ f = F(x, Y(x)). \]
One Chain to Rule Them All

Consider a set of variables \( \mathbf{v} = [v_1, \ldots, v_n]^T \)

and a set of functions \( \mathbf{C} = [C_1(\mathbf{v}), \ldots, C_n(\mathbf{v})]^T \).

where the variables are uniquely defined by constraints \( C_i(\mathbf{v}) = 0 \),

Linearizing these functions:

\[
\Delta \bar{\mathbf{c}} = \frac{\partial \mathbf{C}}{\partial \mathbf{v}} \Delta \mathbf{v},
\]

After some manipulations, this yields the chain rule

\[
\frac{\partial \mathbf{C}}{\partial \mathbf{v}} \frac{d \mathbf{v}}{d \mathbf{c}} = I = \frac{\partial \mathbf{C}^T}{\partial \mathbf{v}} \frac{d \mathbf{v}^T}{d \mathbf{c}}.
\]
Chain Rule in Matrix Form

**Variables and Constraints**

![Diagram showing variables and constraints]

\[
\begin{bmatrix}
\frac{\partial C_1}{\partial v_1} & \cdots & \frac{\partial C_1}{\partial v_n} \\
\frac{\partial C_2}{\partial v_1} & \cdots & \frac{\partial C_2}{\partial v_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial C_n}{\partial v_1} & \cdots & \frac{\partial C_n}{\partial v_n}
\end{bmatrix}
\begin{bmatrix}
dv_1 \\
dv_2 \\
\vdots \\
dv_n
\end{bmatrix} =
\begin{bmatrix}
\frac{\partial C_1}{\partial v_1} & \cdots & \frac{\partial C_1}{\partial v_n} \\
\frac{\partial C_2}{\partial v_1} & \cdots & \frac{\partial C_2}{\partial v_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial C_n}{\partial v_1} & \cdots & \frac{\partial C_n}{\partial v_n}
\end{bmatrix}
\begin{bmatrix}
dc_1 \\
dc_2 \\
\vdots \\
dc_n
\end{bmatrix}
\]

\[
C(v) =
\begin{bmatrix}
C_1(v_1, \ldots, v_n) \\
C_2(v_1, \ldots, v_n) \\
\vdots \\
C_n(v_1, \ldots, v_n)
\end{bmatrix}
\]

**Derivation**

\[
\begin{bmatrix}
\frac{\partial C}{\partial v} \\
\frac{\partial C}{\partial c}
\end{bmatrix}
\begin{bmatrix}
dv \\
dc
\end{bmatrix} =
\begin{bmatrix}
\frac{\partial C}{\partial v}^T \\
\frac{\partial C}{\partial c}
\end{bmatrix}^T
\begin{bmatrix}
dv \\
dc
\end{bmatrix}
\]

**Forward form**

\[
\sum_{k=1}^{n} \frac{\partial C_i}{\partial v_k} \frac{dv_k}{dc_j} = \delta_{ij}
\]

**Reverse form**

\[
\sum_{k=1}^{n} \frac{dv_i}{dc_k} \frac{\partial C_k}{\partial v_j} = \delta_{ij}
\]
Chain Rule in Matrix Form

**Variables and Constraints**

Variables and constraints are represented in a network diagram, where each box represents a constraint function, and the arrows indicate the flow of dependencies. The off-diagonal entries represent the data, and the diagonal entries are the functions in the process.

**Derivation**

The forward form of the chain rule is shown as follows:

\[
\frac{\partial C}{\partial v} \frac{dv}{dc} = I = \left( \frac{\partial C}{\partial v} \right)^T \left( \frac{dv}{dc} \right)^T
\]

Where \( C \) is a function of \( v \), \( v \) is a vector of variables, and \( C(v) \) is the vector of constraint functions.

**Forward form**

\[
\sum_{k=1}^{n} \frac{\partial C_i}{\partial v_k} \frac{dv_k}{dc_j} = \delta_{ij}
\]

**Reverse form**

\[
\sum_{k=1}^{n} \frac{dv_i}{dc_k} \frac{\partial C_k}{\partial v_j} = \delta_{ij}
\]
Monolithic Differentiation

**Variables and Constraints**

\[
\begin{align*}
\mathbf{v} &= \begin{bmatrix} \mathbf{x} \\ \mathbf{f} \end{bmatrix} \\
C(\mathbf{v}) &= \begin{bmatrix} \mathbf{x} - \mathbf{x}^0 \\ \mathbf{f} - \mathbf{F}(\mathbf{x}) \end{bmatrix}
\end{align*}
\]

**Derivation**

\[
\begin{align*}
\frac{\partial \mathbf{C}}{\partial \mathbf{v}} \begin{bmatrix} d\mathbf{v} \\ d\mathbf{c} \end{bmatrix} &= I = \left( \frac{\partial \mathbf{C}}{\partial \mathbf{v}} \right)^T \begin{bmatrix} d\mathbf{v}^T \\ d\mathbf{c}^T \end{bmatrix} \\
\begin{bmatrix} \frac{\partial (\mathbf{x} - \mathbf{x}^0)}{\partial \mathbf{x}} & \frac{\partial (\mathbf{x} - \mathbf{x}^0)}{\partial \mathbf{f}} \\ \frac{\partial (\mathbf{f} - \mathbf{F})}{\partial \mathbf{x}} & \frac{\partial (\mathbf{f} - \mathbf{F})}{\partial \mathbf{f}} \end{bmatrix} \begin{bmatrix} d\mathbf{x} d\mathbf{x} \\ d\mathbf{f} d\mathbf{f} \end{bmatrix} &= I = \begin{bmatrix} \frac{\partial (\mathbf{x} - \mathbf{x}^0)^T}{\partial \mathbf{x}} & \frac{\partial (\mathbf{f} - \mathbf{F})^T}{\partial \mathbf{f}} \\ \frac{\partial (\mathbf{x} - \mathbf{x}^0)^T}{\partial \mathbf{f}} & \frac{\partial (\mathbf{f} - \mathbf{F})^T}{\partial \mathbf{f}} \end{bmatrix} \begin{bmatrix} d\mathbf{x}^T d\mathbf{f}^T \\ d\mathbf{f}^T \end{bmatrix}
\end{align*}
\]

Monolithic differentiation (from forward form)

\[
\frac{d\mathbf{f}}{d\mathbf{x}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}
\]

Monolithic differentiation (from reverse form)

\[
\frac{d\mathbf{f}}{d\mathbf{x}} = \frac{\partial \mathbf{F}}{\partial \mathbf{x}}
\]
Monolithic Differentiation

Variables and Constraints

\[ v = \begin{bmatrix} x \\ f \end{bmatrix} \]

\[ C(v) = \begin{bmatrix} x - x^0 \\ f - F(x) \end{bmatrix} \]

Derivation

\[
\begin{bmatrix} \frac{\partial C}{\partial v} \\ \frac{dv}{dc} \end{bmatrix} = I = \begin{bmatrix} \frac{\partial C}{\partial v}^T \\ \frac{dv}{dc}^T \end{bmatrix}
\]

\[
\begin{bmatrix} \frac{\partial(x - x^0)}{\partial x} & \frac{\partial(x - x^0)}{\partial f} \\ \frac{\partial(f - F)}{\partial x} & \frac{\partial(f - F)}{\partial f} \end{bmatrix} \begin{bmatrix} dx \\ dx \\ df \\ df \end{bmatrix} = I = \begin{bmatrix} \frac{\partial(x - x^0)^T}{\partial f} & \frac{\partial(f - F)^T}{\partial f} \\ \frac{\partial(x - x^0)^T}{\partial x} & \frac{\partial(x - x^0)^T}{\partial f} \end{bmatrix} \begin{bmatrix} dx^T \\ dx^T \\ df^T \\ df^T \end{bmatrix}
\]

\[
\begin{bmatrix} I & 0 \\ \frac{\partial F}{\partial x} & I \end{bmatrix} \begin{bmatrix} I & 0 \\ df & I \end{bmatrix} = I = \begin{bmatrix} I & \frac{\partial F^T}{\partial x} \\ 0 & I \end{bmatrix} \begin{bmatrix} I & \frac{df^T}{dx} \\ \frac{df}{dx} & I \end{bmatrix}
\]

Monolithic differentiation (from forward form)

\[
\frac{df}{dx} = \frac{\partial F}{\partial x}
\]

Monolithic differentiation (from reverse form)

\[
\frac{df}{dx} = \frac{\partial F}{\partial x}
\]
Algorithmic Differentiation

Variables and Constraints

\[
\begin{align*}
&t_1 - T_1 \\
&t_2 - T_2 \\
&t_3 - T_3 \\
&\cdots \\
&t_n - T_n
\end{align*}
\]

\[
\begin{align*}
&\frac{\partial T_2}{\partial t_1} \begin{array}{c}
1 \\
0 \\
\vdots \\
-1 \\
0
\end{array} \\
&\frac{\partial T_n}{\partial t_1} \begin{array}{c}
0 \\
\vdots \\
0 \\
1 \\
-1 \\
0
\end{array}
\end{align*}
\]

\[
C(v) = \begin{bmatrix}
t_1 - T_1(t) \\
t_2 - T_2(t_1) \\
\vdots \\
t_n - T_n(t_1, \ldots, t_{n-1})
\end{bmatrix}
\]

Derivation

\[
\begin{bmatrix}
\frac{\partial C}{\partial v} \\
\frac{d v}{d c}
\end{bmatrix} = I = \begin{bmatrix}
\frac{\partial C}{\partial v}^T \\
\frac{d v}{d c}^T
\end{bmatrix}
\]

Forward mode AD

\[
\frac{dt_i}{dt_j} = \delta_{ij} + \sum_{k=j}^{i-1} \frac{\partial T_i}{\partial t_k} \frac{dt_k}{dt_j}
\]

Reverse mode AD

\[
\frac{dt_i}{dt_j} = \delta_{ij} + \sum_{k=j+1}^{i} \frac{dt_i}{dt_k} \frac{\partial T_k}{\partial t_j}
\]
AD Example

\[ x_1 \]
\[ x_2 \]
\[ y_1 \]
\[ y_2 \]
\[ R_1(x_1, x_2, y_1, y_2) = x_1 y_1 + 2y_2 - \sin x_1 \]
\[ R_2(x_1, x_2, y_1, y_2) = -y_1 + x_2^2 y_2 \]
\[ F_1(x_1, x_2, y_1, y_2) = y_1 \]
\[ F_2(x_1, x_2, y_1, y_2) = y_2 \sin x_1 \]

\textbf{FUNCTION} \ F(x) \ \\
\textbf{REAL} :: x(2), det, y(2), f(2) \ \\
det = 2 + x(1) \times x(2)^{**2} \ \\
y(1) = x(2)^{**2} \times \text{SIN}(x(1))/\text{det} \ \\
y(2) = \text{SIN}(x(1))/\text{det} \ \\
f(1) = y(1) \ \\
f(2) = y(2) \times \text{SIN}(x(1)) \ \\
\text{RETURN} \ \\
\text{END FUNCTION} \ F
FUNCTION F_D(xd, f)
    REAL :: x(2), xd(2)
    REAL :: det, detd
    REAL :: y(2), yd(2)
    REAL :: f(2), f_d(2)

    detd = xd(1)*x(2)**2 + x(1)*2*x(2)*xd(2)
    det = 2 + x(1)*x(2)**2
    yd = 0.0
    yd(1) = ((2*x(2)*xd(2)*SIN(x(1))+x(2)**2*xd(1)*COS(x(1)))*det-
        x(2)**2*&
        SIN(x(1))*detd)/det**2
    y(1) = x(2)**2*SIN(x(1))/det
    yd(2) = (xd(1)*COS(x(1))*det-SIN(x(1))*detd)/det**2
    y(2) = SIN(x(1))/det

    f_d = 0.0
    f_d(1) = yd(1)
    f(1) = y(1)
    f_d(2) = yd(2)*SIN(x(1)) + y(2)*xd(1)*COS(x(1))
    f(2) = y(2)*SIN(x(1))

RETURN
END FUNCTION F_D

FUNCTION F(x)
    REAL :: x(2), det, y(2), f(2)
    det = 2 + x(1)*x(2)**2
    y(1) = x(2)**2*SIN(x(1))/det
    y(2) = SIN(x(1))/det
    f(1) = y(1)
    f(2) = y(2)*SIN(x(1))

RETURN
END FUNCTION F
FUNCTION F(x)
REAL :: x(2), det, y(2), f(2)
det = 2 + x(1)*x(2)**2
y(1) = x(2)**2*SIN(x(1))/det
y(2) = SIN(x(1))/det
f(1) = y(1)
f(2) = y(2)*SIN(x(1))
RETURN
END FUNCTION F

SUBROUTINE F_B(x, xb, fb)
REAL :: x(2), yb(2)
REAL :: f(2), fb(2)
REAL :: det, detb, tempb, temp

det = 2 + x(1)*x(2)**2
y(1) = x(2)**2*SIN(x(1))/det
y(2) = SIN(x(1))/det
f(1) = y(1)
f(2) = y(2)*SIN(x(1))
RETURN
END SUBROUTINE F_B
AD Example: Forward and Reverse

\[
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
-1 & -2 & 1 & 0 & 0 & 0 & 0 \\
-0.18 & -0.561 & 0.093 & 1 & 0 & 0 & 0 \\
-0.18 & 0 & 0.093 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 \\
-0.152 & 0 & 0 & 0 & -0.841 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 \\
0 \\
0 \\
1 \\
2 \\
0 \\
1 \\
0
\end{bmatrix}
= \begin{bmatrix}
1 & 0 \\
0 & 1 \\
1 & 2 \\
0.087 & 0.374 \\
0.087 & -0.187 \\
0.087 & 0.374 \\
0.224 & -0.157 \\
0 & 0
\end{bmatrix}
\begin{bmatrix}
1 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & -1 & -0.18 & -0.18 & 0 & -0.152 \\
0 & 1 & -2 & -0.561 & 0 & 0 & 0 \\
0 & 0 & 1 & 0.093 & 0.093 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & -1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & -0.841 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
0.087 \\
0.374 \\
-0.093 \\
1 \\
0 \\
0 \\
0
\end{bmatrix}
= \begin{bmatrix}
0.087 & 0.224 \\
0.374 & -0.157 \\
-0.093 & -0.079 \\
1 & 0 \\
0 & 0.841 \\
1 & 0 \\
0 & 1
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
0 \\
0 \\
0 \\
0
\end{bmatrix}
\]

100
Analytic Methods

Variables and Constraints

![Diagram showing the relationship between variables and constraints.]

\[ x - x^0 \]
\[ r - R \]
\[ f - F \]

\[ v = \begin{bmatrix} x \\ y \\ f \end{bmatrix} \]

\[ C(v) = \begin{bmatrix} x - x^0 \\ r - R(x, y) \\ f - F(x, y) \end{bmatrix} \]

Derivation

\[
\begin{bmatrix}
\frac{\partial x}{\partial v} & \frac{\partial y}{\partial v} & \frac{\partial f}{\partial v}
\end{bmatrix}
\begin{bmatrix}
\frac{dv}{dc}
\end{bmatrix}
= I
\]

\[
\begin{bmatrix}
\frac{\partial(x - x^0)}{\partial x} & \frac{\partial(x - x^0)}{\partial y} & \frac{\partial(x - x^0)}{\partial f}
\end{bmatrix}
\begin{bmatrix}
\frac{dx}{dc} & \frac{dy}{dc} & \frac{df}{dc}
\end{bmatrix}
= I
\]

\[
\begin{bmatrix}
I & 0 & 0
0 & \frac{\partial R}{\partial y} & \frac{\partial R}{\partial f}
0 & \frac{\partial F}{\partial y} & \frac{\partial F}{\partial f}
\end{bmatrix}
\begin{bmatrix}
\frac{dy}{dc} & \frac{dy}{dc} & \frac{df}{dc}
\end{bmatrix}
= I
\]

Direct method

\[
\frac{\partial R}{\partial y} \frac{dy}{dx} = -\frac{\partial R}{\partial x}
\]

\[
\frac{df}{dx} = \frac{\partial F}{\partial x} + \frac{\partial F}{\partial y} \frac{dy}{dx}
\]

Adjoint method

\[
\frac{\partial R^T}{\partial y} \frac{df^T}{dr} = -\frac{\partial F^T}{\partial y}
\]

\[
\frac{df}{dr} = \frac{\partial F}{\partial y} + \frac{df}{dr} \frac{\partial R}{\partial x}
\]
## Analytic Methods: Direct vs. Adjoint

### From forward chain rule

\[
\begin{bmatrix}
  I \\
  \frac{\partial R}{\partial x} \\
  \frac{\partial F}{\partial x}
\end{bmatrix}
\begin{bmatrix}
  \begin{bmatrix} I \\
  0 \\
  0 
\end{bmatrix} \\
  \begin{bmatrix} \frac{\partial R}{\partial y} \\
  \frac{\partial F}{\partial y}
\end{bmatrix} \\
  \frac{\partial f}{\partial x}
\end{bmatrix}
= 
\begin{bmatrix} I \\
  0 \\
  0 
\end{bmatrix}
\]

\[
\frac{df}{dx} = \frac{\partial F}{\partial x} + \frac{\partial F}{\partial y} \frac{dy}{dx} - \frac{\partial R}{\partial x} \frac{dy}{dx}
\]

### Solution

\[
\frac{df}{dx} = \frac{\partial F}{\partial x} \frac{dy}{dy} \frac{\partial R}{\partial y}^{-1} \frac{\partial R}{\partial x}
\]

### From reverse chain rule

\[
\begin{bmatrix}
  I \\
  -\frac{\partial R}{\partial x} \\
  0
\end{bmatrix}
\begin{bmatrix}
  \begin{bmatrix} dF^T \\
  df^T
\end{bmatrix} \\
  \frac{\partial R^T}{\partial x} \\
  \frac{\partial F^T}{\partial y}
\end{bmatrix}
= 
\begin{bmatrix} 0 \\
  0 \\
  I 
\end{bmatrix}
\]

### Direct method

\[
\frac{df}{dx} = \frac{\partial F}{\partial x} + \frac{\partial F}{\partial y} \frac{dy}{dx}
\]

### Adjoint method

\[
\frac{df}{dx} = \frac{\partial F}{\partial x} \frac{df}{dr} \frac{\partial R}{\partial x} - \frac{\partial R^T}{\partial y} \frac{df^T}{dr}
\]

### Figure 10: Block matrix diagrams illustrating the structure of the direct and adjoint equations, assuming that \( n_y > n_f \). The blue matrices contain partial derivatives, which are relatively cheap to compute, and the red matrices contain the total derivatives computed by solving the linear systems.

---

2. Direct Method

The direct method involves solving the linear system with \( \frac{\partial R}{\partial x} \) as the right-hand side vector, which results in the linear system \( (32) \). This linear system needs to be solved for \( n_x \) right-hand sides to get the full Jacobian matrix \( \frac{dy}{dx} \). Then, we can use \( \frac{dy}{dx} \) in Eq. \( (30) \) to obtain the derivatives of interest, \( \frac{df}{dx} \).

As in the case of finite differences, the cost of computing derivatives with the direct method is proportional to the number of design variables, \( n_x \). In a case where the computational model is a nonlinear system, the direct method can be advantageous. Both methods require the solution of a system of the same size \( n_x \) times, but the direct method just solves the linear system \( (32) \), while the finite-difference method solves the original nonlinear system \( (4) \). Even though the various solutions required for the finite-difference method can be warm-started from a previous solution, a nonlinear solution will typically require multiple iterations to converge. The direct method is even more advantageous when a factorization of \( \frac{\partial R}{\partial x} \) is available, since each solution of the linear system consists in an inexpensive back substitution. There are some cases where a hybrid approach that combines the direct and adjoint methods is advantageous.

---

[Martins and Hwang, AIAAJ, 2013]
Coupled Analytic Methods: Residual Form

Variables and Constraints

\[
\begin{align*}
\begin{array}{c}
\frac{\partial x}{\partial x} & \frac{\partial x}{\partial y_1} & \cdots & \frac{\partial x}{\partial y_N} \\
r_1 - R_1 & & \cdots & \\
\vdots & & \cdots & \\
r_N - R_N & & \cdots & \frac{\partial x}{\partial y_N}
\end{array}
\end{align*}
\]

\[
v = \begin{bmatrix} x \\ y_1 \\ \vdots \\ y_N \\ f \end{bmatrix}
\]

\[
C(v) = \begin{bmatrix} r_1 - R_1(x, y_1, \ldots, y_N) \\ \vdots \\ r_N - R_N(x, y_1, \ldots, y_N) \\ f - F(x, y_1, \ldots, y_N) \end{bmatrix}
\]

Derivation

\[
\frac{\partial C}{\partial v} \begin{bmatrix} \frac{dv}{dc} \end{bmatrix} = I = \left( \frac{\partial C}{\partial v} \right)^T \begin{bmatrix} \frac{dv}{dc} \end{bmatrix}^T
\]

\[
\begin{bmatrix}
I & 0 & \cdots & 0 & 0 \\
\frac{\partial R_1}{\partial x} & \frac{\partial R_1}{\partial y_1} & \cdots & \frac{\partial R_1}{\partial y_N} & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
\frac{\partial R_N}{\partial x} & \frac{\partial R_N}{\partial y_1} & \cdots & \frac{\partial R_N}{\partial y_N} & 0 \\
\frac{\partial F}{\partial x} & \frac{\partial F}{\partial y_1} & \cdots & \frac{\partial F}{\partial y_N} & I
\end{bmatrix}
\begin{bmatrix}
dy_1 \\
dy_1 \\
\vdots \\
dy_N \\
dy_1 \\
dy_N \\
\vdots \\
\ddr_1 \\
\vdots \\
\ddr_N
\end{bmatrix}
= I =

\begin{bmatrix}
I - \frac{\partial R_1^T}{\partial x} & \cdots & \frac{\partial R_N^T}{\partial x} & - \frac{\partial F^T}{\partial x} \\
0 & \frac{\partial R_1^T}{\partial y_1} & \cdots & \frac{\partial R_N^T}{\partial y_1} & - \frac{\partial F^T}{\partial y_1} \\
0 & \frac{\partial R_1^T}{\partial y_N} & \cdots & \frac{\partial R_N^T}{\partial y_N} & - \frac{\partial F^T}{\partial y_N} \\
0 & 0 & \cdots & 0 & I
\end{bmatrix}
\begin{bmatrix}
dy_1^T \\
dy_1^T \\
\vdots \\
dy_N^T \\
dy_1^T \\
dy_N^T \\
\vdots \\
\ddr_1^T \\
\vdots \\
\ddr_N^T
\end{bmatrix}
\]

Coupled direct: residual form

\[
df = \frac{\partial F}{\partial x} + \left[ \frac{\partial F}{\partial y_1} \cdots \frac{\partial F}{\partial y_N} \right] \begin{bmatrix}
dy_1 \\
dy_1 \\
\vdots \\
dy_N \\
dy_1 \\
dy_N \\
\vdots \\
\ddr_1 \\
\vdots \\
\ddr_N
\end{bmatrix}
\]

Coupled adjoint: residual form

\[
df = \frac{\partial F}{\partial x} + \left[ \frac{\partial f}{\partial y_1} \cdots \frac{\partial f}{\partial y_N} \right] \begin{bmatrix}
dr_1 \\
dr_1 \\
\vdots \\
r_N \\
r_1 \\
r_N \\
\vdots \\
\ddr_1 \\
\vdots \\
\ddr_N
\end{bmatrix}
\]

\[
\frac{\partial R_1}{\partial x} & \cdots & \frac{\partial R_N}{\partial x} \\
\frac{\partial R_1}{\partial y_1} & \cdots & \frac{\partial R_N}{\partial y_1} \\
\vdots & \ddots & \vdots \\
\frac{\partial R_1}{\partial y_N} & \cdots & \frac{\partial R_N}{\partial y_N}
\end{bmatrix}
\begin{bmatrix}
dy_1 \\
dy_1 \\
\vdots \\
dy_N \\
dy_1 \\
dy_N \\
\vdots \\
\ddr_1 \\
\vdots \\
\ddr_N
\end{bmatrix}
\]
Coupled Analytic Methods: Functional Form

Variables and Constraints

\[
\begin{align*}
&x - x^0 & x & x \cdots x \\
y_1 - Y_1 & y_1 & y_1 \cdots y_1 \\
&y_N - Y_N \cdots y_N & y_N \\
f - F
\end{align*}
\]

\[
\begin{bmatrix}
v \end{bmatrix} =
\begin{bmatrix}
x \\
y_1 \\
\vdots \\
y_N \\
f
\end{bmatrix},
\]

\[
C(v) =
\begin{bmatrix}
x - x^0 \\
y_1 - Y_1(x, y_2, \ldots, y_N) \\
\vdots \\
y_N - Y_N(x, y_1, \ldots, y_{N-1}) \\
f - F(x, y_1, \ldots, y_N)
\end{bmatrix}
\]

Derivation

\[
\begin{bmatrix}
\frac{\partial C}{\partial v} & \frac{dv}{dc}
\end{bmatrix}
= I =
\begin{bmatrix}
\frac{\partial C}{\partial v}^T & \frac{dv}{dc}^T
\end{bmatrix}
\]

Coupled direct: functional form

\[
\begin{bmatrix}
I & \cdots & \frac{\partial Y_1}{\partial y_N} \\
\vdots & \ddots & \vdots \\
-\frac{\partial Y_N}{\partial y_1} & \cdots & I
\end{bmatrix}
\begin{bmatrix}
\frac{dy_1}{dx} \\
\vdots \\
\frac{dy_N}{dx}
\end{bmatrix}
= \begin{bmatrix}
\frac{\partial Y_1}{\partial x} \\
\vdots \\
\frac{\partial Y_N}{\partial x}
\end{bmatrix}
\]

\[
\frac{df}{dx} = \frac{\partial F}{\partial x} + \begin{bmatrix}
\frac{\partial F}{\partial y_1} & \cdots & \frac{\partial F}{\partial y_N}
\end{bmatrix}
\begin{bmatrix}
\frac{dy_1}{dx} \\
\vdots \\
\frac{dy_N}{dx}
\end{bmatrix}
\]

Coupled adjoint: functional form

\[
\begin{bmatrix}
I & \cdots & \frac{\partial Y_N^T}{\partial y_1} \\
\vdots & \ddots & \vdots \\
-\frac{\partial Y_1^T}{\partial y_N} & \cdots & I
\end{bmatrix}
\begin{bmatrix}
\frac{df}{dy_1} \\
\vdots \\
\frac{df}{dy_N}
\end{bmatrix}
= \begin{bmatrix}
\frac{\partial F^T}{\partial y_1} \\
\vdots \\
\frac{\partial F^T}{\partial y_N}
\end{bmatrix}
\]

\[
\frac{df}{dx} = \frac{\partial F}{\partial x} + \begin{bmatrix}
\frac{df}{dy_1} & \cdots & \frac{df}{dy_N}
\end{bmatrix}
\begin{bmatrix}
\frac{dy_1}{dx} \\
\vdots \\
\frac{dy_N}{dx}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{\partial Y_1}{\partial y_1} \\
\vdots \\
\frac{\partial Y_N}{\partial y_N}
\end{bmatrix}
\]

\[
\begin{bmatrix}
\frac{\partial Y_1}{\partial x} \\
\vdots \\
\frac{\partial Y_N}{\partial x}
\end{bmatrix}
\]
Application of Coupled Adjoint Derivatives

Application to MDO of small satellite

Computational framework for gradient-based MDAO

\[
\frac{\partial C_i}{\partial \mathbf{v}} \quad M_i^{-1} \quad \mathbf{y}_i \quad C_i \quad \mathbf{v}_i^* \quad \mathbf{v}_i
\]

\begin{align*}
(4) & & (3) & & (2) \quad (1) \\
\text{Component} & | \text{Jacobian} & | \text{Preconditioner} & | \text{solve} \\
\text{Triangular} & \text{Exact} & \text{Back subst.} & \text{Exact solve} \\
\text{Preconditioned} & \text{Exact} & \text{Preconditioner} & \text{No action} \\
\text{Factorized} & \text{Exact} & \text{Exact inverse} & \text{No action} \\
\text{Jacobian-free} & \text{Directional derivative} & \text{No action} & \text{No action}
\end{align*}

Used to solve for millions of states and tens of thousands of design variables

[Hwang et al., AIAA SDM, 2013]
Further Reading

http://mdolab.engin.umich.edu/publications


Other Relevant publications


