

Problem formulation and modeling in simulation-based design

Natalia M. Alexandrov¹ and Robert Michael Lewis²

¹Multidisciplinary Optimization Branch, NASA Langley Research Center

²Department of Mathematics, College of William & Mary

<http://mdob.larc.nasa.gov>

Traditional approach to simulation-based optimization

- Analysis (simulation)
 - Given a vector of design variables x , a simulation or a system of simulations computes responses $u(x)$ of interest by solving a system $A(x, u(x)) = 0$
- Optimization
 - Do until convergence
 1. Build **local models** (usually Taylor series) of the objective and constraints based on information computed **directly by the high-fidelity simulation**
 2. Compute a trial step by solving a local model-based subproblem
 3. Use a globalization technique (e.g., trust regions) to improve convergence
 - End do
- But...

Features of realistic design problems

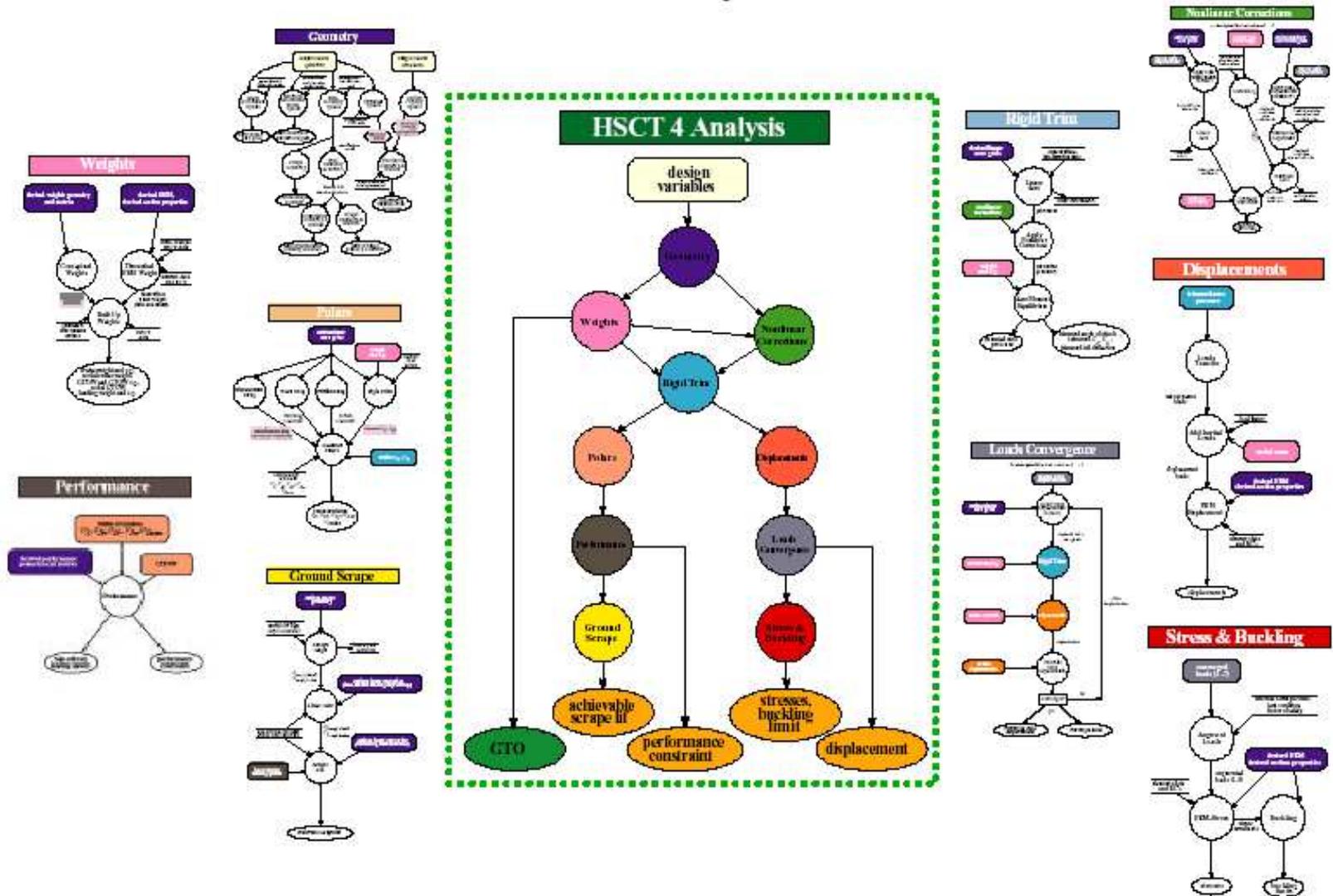
- Examples of difficulties heard in previous talks (e.g., Nielsen)
- Summary of limiting factors
 - Modeling
 - Functions are expensive and not robust
 - Difficult to obtain reliable and affordable derivatives
 - Optimization
 - Algorithms must be fault-tolerant to a high degree
 - Derivative-based optimization is expensive for problems with high-fidelity simulations
 - Derivative-free optimization is prohibitively expensive for large problems, although is becoming more practical (see, e.g., Giunta)

A remedy for expense of simulations

- Instead of local, Taylor series-based models, use models that have **better global approximation** properties:
 - Variable accuracy (converge to user-specified tolerance)
 - Variable resolution (vary degree of mesh refinement)
 - Variable-fidelity physics (e.g., inviscid flow vs. Navier-Stokes)
 - Data fitting (kriging, response surfaces, reduced order, etc.)
- Some examples in these sessions (Campana, Giunta)
- At LaRC
 - Address expense and lack of robustness in function evaluation via Approximation and Model Management Optimization (AMMO, Alexandrov et al.)
 - Demonstrations with variable-fidelity physics and variable resolution models in aerodynamic optimization
 - Convergence to high-fidelity answers with currently 5-fold savings in terms of high-fidelity simulations
- But...

Example: small multidisciplinary analysis (MDA)

Full HSCT 4 Analysis Procedures



Multidisciplinary optimization (MDO)

MDO = systematic approaches to the design of complex, coupled systems

Multidisciplinary: different aspects of the design problem (e.g., controls, aerodynamics, structures, propulsion, etc. for aerospace vehicle)

Design \neq Nonlinear Programming!

Limit discussion to the subset of the total design problem that can be represented as a nonlinear program (NLP)

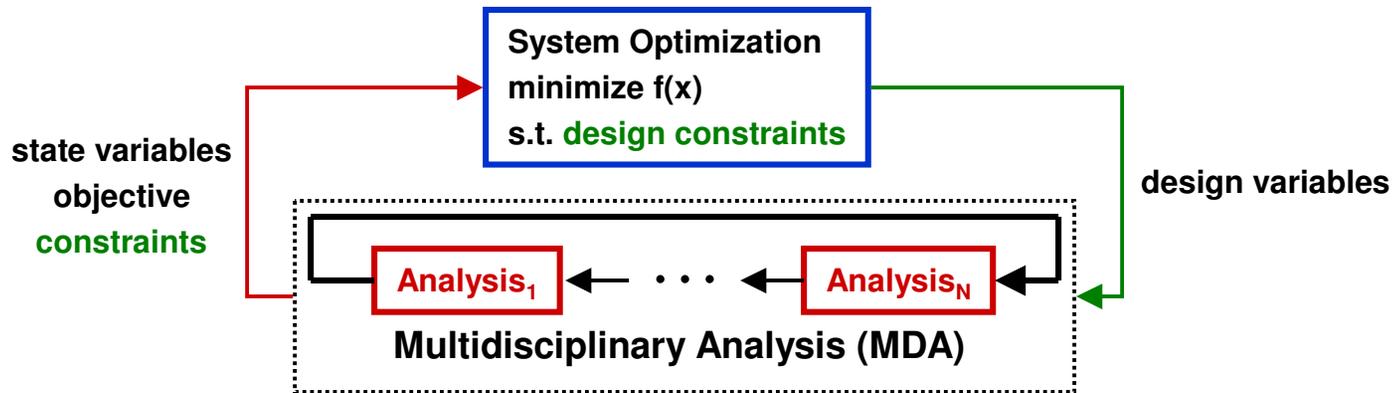
Efficiency considerations in solving MDO problems

- Computational efficiency of disciplinary components
- Problem synthesis (implementation)
 - Disciplinary interfaces
 - Data standards
 - Computational frameworks
- Effect of problem formulation
 - Computational efficiency/tractability
 - Convergence and robustness
 - Disciplinary autonomy
 - Let disciplines design independently
 - Keep local design variables in disciplines

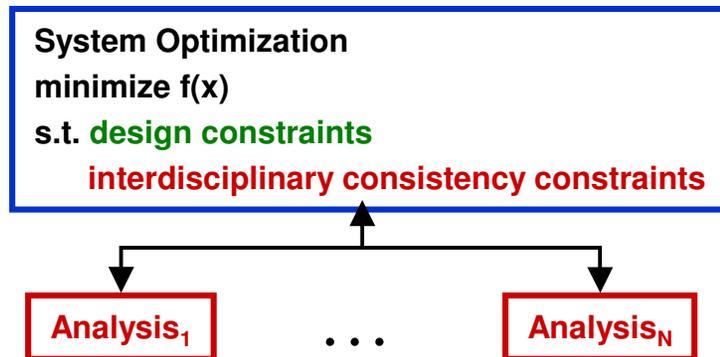
Influence of formulation on performance

Example: HPCCP formulation study, Alexandrov & Kodiyalam, AIAA 1998-4884

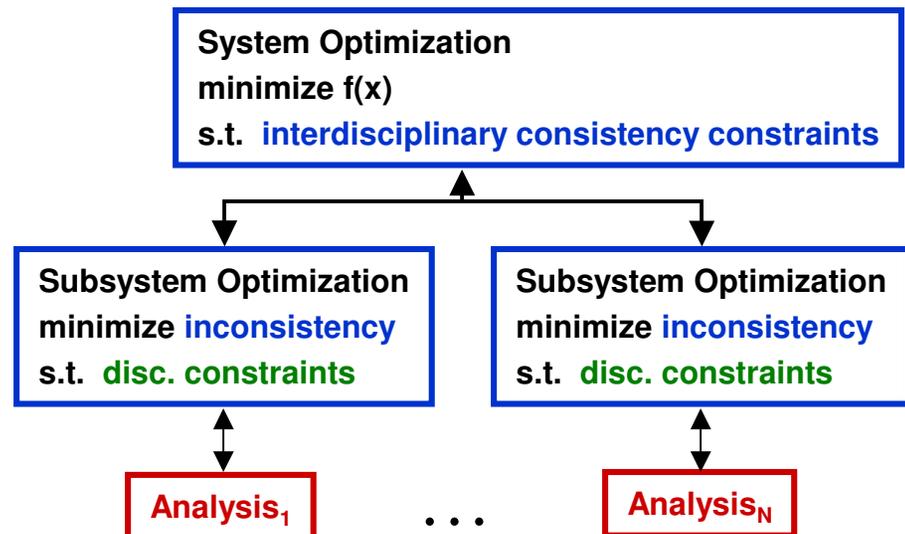
Fully Integrated Optimization (FIO)



Distributed Analysis Optimization (DAO)



Collaborative Optimization (CO)



Influence of formulation on performance

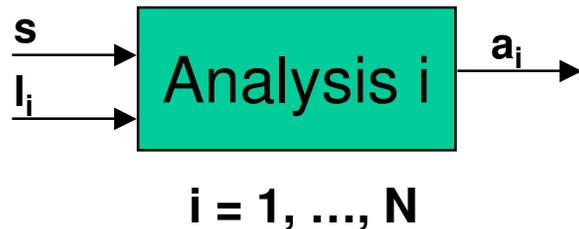
- Analytical features of MDO problem formulations, e.g., the degree of disciplinary autonomy, directly affects the ability of numerical algorithms to solve the problem reliably and efficiently

	1	2	3	4	5	6	7
FIO	610	220	610	81	3234	5024	8730
DAO	9530	8796	382	N/A	544	932	N/A
CO	15626	19872	1785	2102	837	40125	691058

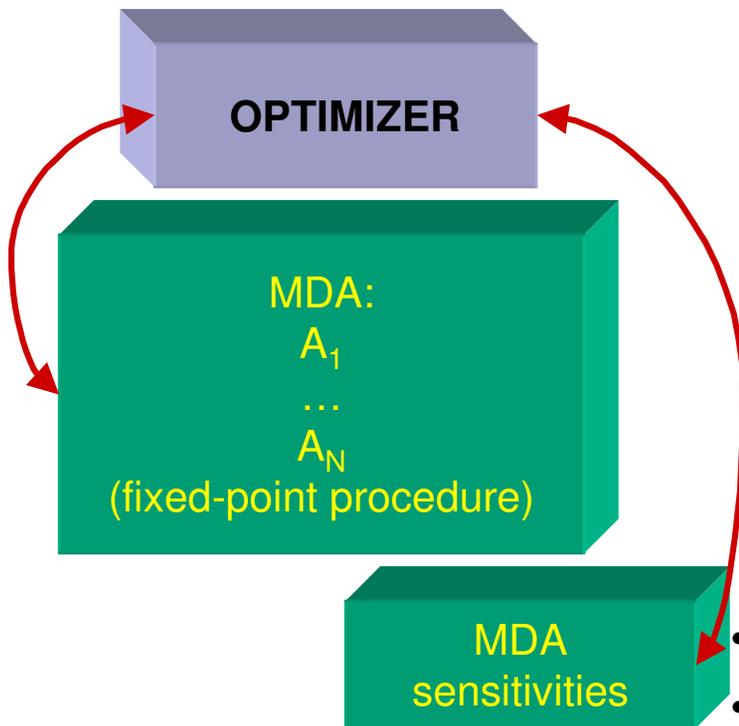
Cost of optimization in terms of analyses for 7 MDO Test Suite problems

MDO Problem Synthesis / Implementation

Problem:
design for objective f with



- Successful MDO-NLP usually in academic environments (simulation codes open to modification) or via *ad hoc* approaches
- Realistic MDO
 - Heroic software integration for MDA
 - MDA = (usually) fixed-point iteration; too rigid
 - May leave no resources for computing derivatives or experimenting with optimization
 - Difficult to get MDA-based objectives and constraints *automatically*
 - To reformulate the problem, need to “unscramble” codes
- ∴ One-of-a-kind, monolithic implementations
- Want flexible and/or hybrid reconfigurable formulations

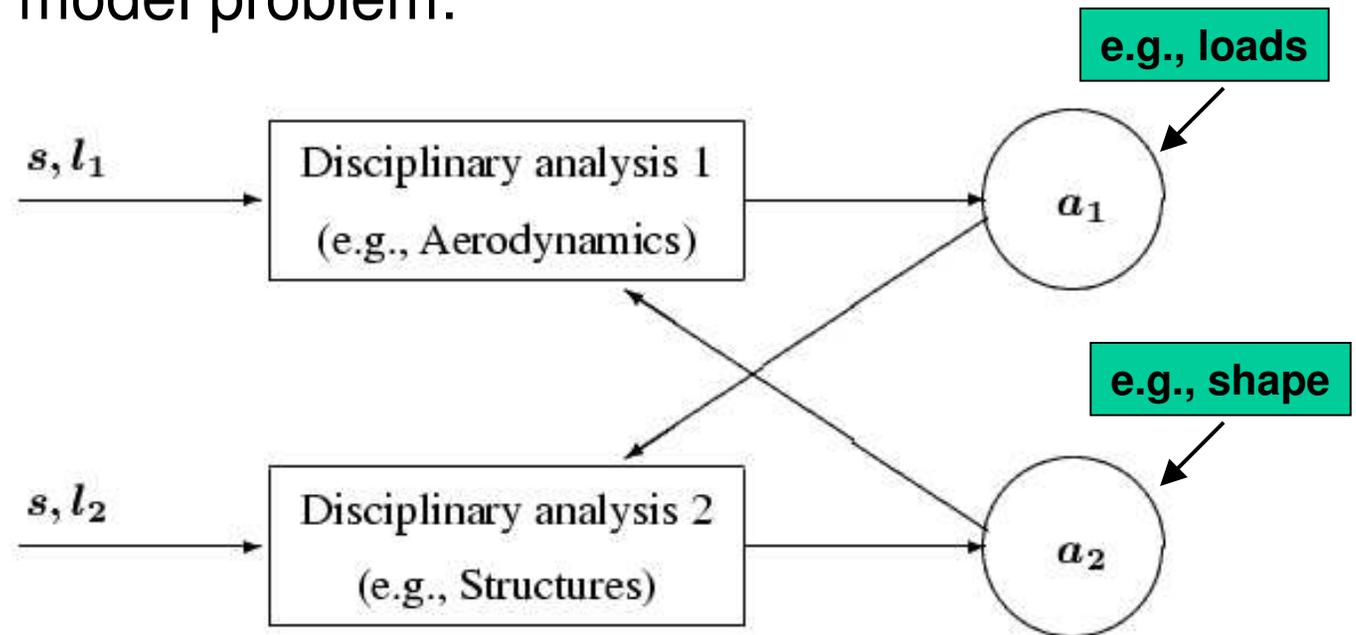


Idea of reconfigurability

- Computational component-based approach to MDO problem synthesis that allows for straightforward transformation among problem formulations within optimization algorithms
 - All MDO formulations are related and share the basic computational components
 - Appropriate implementation enables re-use of components in a straightforward way
- Long-term plan: Tools for formulation analysis and matching with optimization algorithms to be included in computational frameworks

Origins of reconfigurability

- The capacity for reconfigurability stems from the relationship among formulations
- Two-discipline model problem:



- Coupled MDA \sim the physical requirement that a solution satisfy both analyses
- Given $\boldsymbol{x} = (s, l_1, l_2)$, we have

$$a_1 = A_1(s, l_1, a_2)$$

$$a_2 = A_2(s, l_2, a_1)$$

Simultaneous Analysis and Design (SAND)

Write MDA as

$$\begin{aligned} a_1 &= A_1(s, l_1, t_2) \\ a_2 &= A_2(s, l_2, t_1) \\ t_1 &= a_1 \\ t_2 &= a_2 \end{aligned}$$

Relax all couplings;
All variables independent

$$\begin{aligned} &\underset{s, l_1, l_2, a_1, a_2, t_1, t_2}{\text{minimize}} && f(s, t_1, t_2) \\ &\text{subject to} && \begin{cases} c_1(s, l_1, a_1) \geq 0 \\ c_2(s, l_2, a_2) \geq 0 \end{cases} \\ &\text{disciplinary constraints} && \left. \begin{cases} a_1 = A_1(s, l_1, t_2) \\ a_2 = A_2(s, l_2, t_1) \end{cases} \right\} \\ &\text{analysis constraints} && \left. \begin{cases} t_1 = a_1 \\ t_2 = a_2 \end{cases} \right\} \\ &\text{consistency constraints} && \end{aligned}$$

SAND, cont.

- Advantages
 - No need for expensive analyses far from solution
 - Reduced nonlinearity in NLP
- Disadvantages
 - Analyses may not be readily available in residual form
 - Potentially huge number of variables
 - Analysis solution techniques must be integrated with optimization
 - Intermediate designs may not be physically realizable
 - Disciplinary autonomy unclear
- All other formulations may be viewed as derived from the SAND formulation by eliminating a particular set of independent variables from the optimization problem via closing a particular set of constraints or solving optimization problems.

Distributed Analysis Optimization (DAO)

Close disciplinary consistency constraints;
relax the coupling in MDA; maintain disciplinary analyses

A DAO formulation is

$$\begin{array}{ll} \underset{s, l_1, l_2, t_1, t_2}{\text{minimize}} & f(s, t_1, t_2) \\ \text{subject to} & \left. \begin{array}{l} c_1(s, l_1, t_1) \geq 0 \\ c_2(s, l_2, t_2) \geq 0 \end{array} \right\} \text{disciplinary constraints} \\ & \left. \begin{array}{l} t_1 = a_1(s, l_1, t_2) \\ t_2 = a_2(s, l_2, t_1) \end{array} \right\} \text{consistency constraints} \end{array}$$

where the disciplinary responses $a_1(s, l_1, t_2)$ and $a_2(s, l_2, t_1)$ are found by closing the disciplinary analysis constraints

$$\begin{array}{l} a_1 = A_1(s, l_1, t_2) \\ a_2 = A_2(s, l_2, t_1). \end{array}$$

(Versions known as Individual Discipline Feasible, In-Between, etc.)

DAO, cont.

- Advantages

- Some measure disciplinary autonomy
- Fewer design variables than in SAND
- Conventional single-level NLP

- Disadvantages

- Intermediate designs may not be physically realizable (although perhaps less “disciplinary infeasible” than in SAND)
- Disciplinary autonomy limited – optimization problem deals with both local and shared variables

Fully Integrated Optimization (FIO)

The corresponding FIO formulation is

$$\begin{aligned} & \underset{s, l_1, l_2}{\text{minimize}} && f(s, t_1(s, l_1, l_2), t_2(s, l_1, l_2)) \\ & \text{subject to} && c_1(s, l_1, t_1(s, l_1, l_2)) \geq 0 \\ & && c_2(s, l_2, t_2(s, l_1, l_2)) \geq 0 \end{aligned}$$

where we compute $t_1(s, l_1, l_2)$ and $t_2(s, l_1, l_2)$ by solving the MDA

$$\begin{aligned} a_1 &= A_1(s, l_1, t_2) & t_1 &= a_1 \\ a_2 &= A_2(s, l_2, t_1) & t_2 &= a_2. \end{aligned}$$

FIO, cont.

- Advantages

- Smallest set of design variables
- Intermediate designs are realizable – can stop optimization away from optimality for lack of resources

- Disadvantages

- Requires MDA
- Requires derivatives of MDA
- MD processes are difficult to converge
- Disciplinary autonomy limited

Hierarchy of formulations and reconfigurability

- Start with SAND – all variables independent ($s, l_1, l_2, t_1, t_2, a_1, a_2$)
- Eliminate (t_1, t_2, a_1, a_2) via MDA \Rightarrow FIO
- Eliminate (a_1, a_2) via disciplinary analyses \Rightarrow DAO
- Eliminate (a_1, a_2) via disciplinary analyses + eliminate (l_1, l_2) via disciplinary design constraints \Rightarrow generally leads to bilevel optimization problems
 - Significant degree of disciplinary autonomy
 - Bilevel program with a badly behaved system-level problem
 - Causes conventional algorithms to fail or be slow (NMA/RML, AIAA J.)
- **Computational components remain unchanged**
- Standard results on reduced derivatives tell us that the sensitivities in DAO and FIO are related to those in SAND *via variable reduction*
- **Therefore, computational components of one formulation can be reconfigured to yield those of another in the context specific algorithms**

Reduced derivatives

Let

$$\Phi(\mathbf{x}) = \phi(\mathbf{x}, \mathbf{v}(\mathbf{x})).$$

Given \mathbf{x} , $\mathbf{v}(\mathbf{x})$ is computed from

$$\mathbf{S}(\mathbf{x}, \mathbf{v}(\mathbf{x})) = \mathbf{0}.$$

Let \mathbf{W} be the *injection operator* (\mathbf{W}^T is the reduction operator):

$$\mathbf{W} = \mathbf{W}(\mathbf{x}, \mathbf{v}) = \begin{pmatrix} \mathbf{I} \\ -\mathbf{S}_v^{-1}(\mathbf{x}, \mathbf{v}) \mathbf{S}_x(\mathbf{x}, \mathbf{v}) \end{pmatrix}.$$

Define $\boldsymbol{\lambda}$ by

$$\boldsymbol{\lambda} = \boldsymbol{\lambda}(\mathbf{x}, \mathbf{v}) = -(\mathbf{S}_v(\mathbf{x}, \mathbf{v}))^{-T} \nabla_v \phi(\mathbf{x}, \mathbf{v})$$

and the Lagrangian $L(\mathbf{x}, \mathbf{v}; \boldsymbol{\lambda})$ by

$$L(\mathbf{x}, \mathbf{v}; \boldsymbol{\lambda}) = \phi(\mathbf{x}, \mathbf{v}) + \boldsymbol{\lambda}^T \mathbf{S}(\mathbf{x}, \mathbf{v}).$$

Reduced derivatives

The derivatives of ϕ and Φ are related as follows:

$$\nabla_{\mathbf{x}} \Phi(\mathbf{x}) = \underbrace{W^T(\mathbf{x}, \mathbf{v}(\mathbf{x})) \nabla_{(\mathbf{x}, \mathbf{v})} \phi(\mathbf{x}, \mathbf{v}(\mathbf{x}))}_{\text{Reduced gradient}}.$$

Reduced gradient

$$\nabla_{\mathbf{x}\mathbf{x}}^2 \Phi(\mathbf{x}) = \underbrace{W^T \left(\nabla_{(\mathbf{x}, \mathbf{v})}^2 \phi + \nabla_{(\mathbf{x}, \mathbf{v})}^2 S \cdot \lambda \right) W}_{\text{Reduced Hessian of the Lagrangian}},$$

where

Reduced Hessian of the Lagrangian

$$W = W(\mathbf{x}, \mathbf{v}(\mathbf{x}))$$

$$\nabla_{(\mathbf{x}, \mathbf{v})}^2 \phi = \nabla_{(\mathbf{x}, \mathbf{v})}^2 \phi(\mathbf{x}, \mathbf{v}(\mathbf{x}))$$

$$\nabla_{(\mathbf{x}, \mathbf{v})}^2 S \cdot \lambda = \nabla_{(\mathbf{x}, \mathbf{v})}^2 S(\mathbf{x}, \mathbf{v}(\mathbf{x})) \cdot \lambda(\mathbf{x}, \mathbf{v}(\mathbf{x}))$$

$$= \sum_{i=1}^n \lambda_i \nabla_{(\mathbf{x}, \mathbf{v})}^2 S_i.$$

Barrier-SQP approach to SAND

Now illustrate reconfigurability in the context of a specific class of algorithms, barrier-SQP methods

Let

$$F_{\text{SAND}}(s, l_1, l_2, t_1, t_2) = f(s, t_1, t_2) - \mu \left[\sum_i \ln c_1^i(s, l_1, t_1) + \sum_j \ln c_2^j(s, l_2, t_2) \right]$$

Barrier-SQP solves a sequence of subproblems of the form:

$$\begin{array}{ll} \underset{s, l_1, l_2, t_1, t_2, a_1, a_2}{\text{minimize}} & F_{\text{SAND}}(s, l_1, l_2, t_1, t_2) \\ \text{subject to} & a_1 = A_1(s, l_1, t_2) \\ & a_2 = A_2(s, l_2, t_1) \\ & t_1 = a_1 \\ & t_2 = a_2, \end{array}$$

Barrier-SQP approach to DAO

Let

$$F_{\text{DAO}}(s, l_1, l_2, t_1, t_2) = f(s, t_1, t_2) - \mu \left[\sum_i \ln c_1^i(s, l_1, t_1) + \sum_j \ln c_2^j(s, l_2, t_2) \right]$$

Barrier subproblem for DAO is

$$\begin{aligned} & \underset{s, l_1, l_2, t_1, t_2}{\text{minimize}} && F_{\text{DAO}}(s, l_1, l_2, t_1, t_2) \\ & \text{subject to} && t_1 = a_1(s, l_1, t_2) \\ & && t_2 = a_2(s, l_2, t_1), \end{aligned}$$

where the disciplinary responses $a_1(s, l_1, t_2)$ and $a_2(s, l_2, t_1)$ are computed via the disciplinary analyses:

$$\begin{aligned} a_1 &= A_1(s, l_1, t_2) \\ a_2 &= A_2(s, l_2, t_1). \end{aligned}$$

Relationship among SAND, DAO, FIO Sensitivities

Then setting an appropriate (x, v) for each formulation, we have

$$\nabla_{(s,l_1,l_2,t_1,t_2)} F_{\text{DAO}} = W_{\text{DAO}}^T \nabla_{(s,l_1,l_2,t_1,t_2,a_1,a_2)} F_{\text{SAND}}$$

and

$$\nabla_{(s,l_1,l_2,t_1,t_2)}^2 F_{\text{DAO}} = W_{\text{DAO}}^T \nabla_{(s,l_1,l_2,t_1,t_2,a_1,a_2)}^2 F_{\text{SAND}} W_{\text{DAO}}.$$

A similar relationship exists between the sensitivities for solving the barrier-SQP subproblems for SAND and FIO:

$$\nabla_{(s,l_1,l_2)} F_{\text{FIO}} = W_{\text{FIO}}^T \nabla_{(s,l_1,l_2,t_1,t_2,a_1,a_2)} F_{\text{SAND}}$$

and

$$\nabla_{(s,l_1,l_2)}^2 F_{\text{FIO}} = W_{\text{FIO}}^T \nabla_{(s,l_1,l_2,t_1,t_2,a_1,a_2)}^2 F_{\text{SAND}} W_{\text{FIO}},$$

where the expressions for the reduction operators W_{FIO}^T and W_{DAO}^T are given in the paper.

Solving barrier-SQP subproblem

Solving barrier subproblem is an iterative process, in which we approximately solve

$$\begin{aligned} &\text{minimize} && \frac{1}{2} \mathbf{p}^T \mathbf{H} \mathbf{p} + \mathbf{g}^T \mathbf{p} \\ &\text{subject to} && \nabla S^T \mathbf{p} + S = \mathbf{0} \end{aligned}$$

\mathbf{H} - approximation to the Hessian of the Lagrangian

\mathbf{g} - is the gradient of the Lagrangian

\mathbf{p} - step in the iterative process

Reduced-basis approach to barrier-SQP subproblem

- For a **specific choice of algorithm** for solving the barrier-SQP subproblem, can say even more about the relationship among the computational elements needed to solve the three formulations
- The relationship among the sensitivities means that it is possible to implement an optimization algorithm for SAND so that with a **single modification** we obtain an algorithm for DAO or FIO

Reduced-basis barrier-SQP for SAND

Algorithm 1: Reduced-basis algorithm for SAND

Initialization: Choose an initial (x_c, v_c) .

Until convergence, do {

1. Compute the multiplier $\lambda_{SAND} = -S_v^{-1} \nabla_v F_{SAND}$.
2. Test for convergence.
3. Construct a local model of L about (x_c, v_c) .
4. Take a step p^{LF} to improve linear feasibility:

$$p^{LF} = \alpha \begin{pmatrix} 0 \\ -S_v^{-1} S \end{pmatrix}.$$

5. Subject to the improved linear feasibility, improve optimality:

$$\begin{aligned} &\text{minimize} && \frac{1}{2} q^T W^T H W q + (g + H p_{LF})^T W^T q \\ &\text{subject to} && \| p_{LF} + W q \| \leq r. \end{aligned}$$

6. Set $p = (p_x, p_v) = p_{LF} + W q$.
7. Evaluate $(x_+, v_+) = (x_c, v_c) + (p_x, p_v)$ and update $(x_c, v_c), r$. }

Reduced-basis SQP for FIO and DAO

Algorithm 2: Reduced-basis algorithm for SAND + analysis = FIO

Initialization: Choose an initial x_c .

Analysis: Solve $S_{\text{FIO}}(x_c, v_c(x_c)) = 0$ for $v_c(x_c)$.

Until convergence, do {

1–6. These steps remain unchanged.

7. **Analysis: Solve $S_{\text{FIO}}(x_+, v_+) = 0$ for $v_+(x_+)$; evaluate (x_+, v_+) .**

8. This step remains unchanged.

}

Algorithm 3: Reduced-basis algorithm for SAND + analysis = DAO

Initialization: Choose an initial (x_c, v_c) .

Analysis: Solve $S_{\text{DAO}}(x_c, v_c(x_c)) = 0$ for $v_c(x_c)$.

Until convergence, do {

1–6. These steps remain unchanged.

7. **Analysis: Solve $S_{\text{DAO}}(x_+, v_+) = 0$ for $v_+(x_+)$; evaluate (x_+, v_+) .**

8. This step remains unchanged.

}

Other algorithms

- Outlined reconfigurable scheme should work for other methods that handle inequalities via a penalty function (e.g., augmented Lagrangian)
- Active set methods are likely to take more work

Concluding remarks

- MDO problem formulation directly affects the tractability of the problem and efficiency of solution
- **Conjecture:** A method for MDO can possess at most two of the following three attributes:
 - Computational autonomy
 - Computational robustness
 - Computational efficiency
- Regardless of the formulation, there is a clear need for flexible problem synthesis and easy reconfiguration
- Basic computational components combined with transformations within specific algorithms form a promising approach