MIT esd

16.888
ESD.77

Multidisciplinary System Design Optimization (MSDO)

Post-Optimality Analysis

Lecture 16

Olivier de Weck

Karen Willcox

Today's Topics

- Optimality Conditions & Termination
  - Gradient-based techniques
  - Heuristic techniques
- Objective Function Behavior
- Scaling

Standard Problem Definition

$$\min \; J(\mathbf{x})$$

$$\text{s.t.} \; g_j(\mathbf{x}) \le 0 \quad j = 1,..,m_1$$

$$h_k(\mathbf{x}) = 0 \quad k = 1,..,m_2$$

$$x_i^{\ell} \le x_i \le x_i^{u} \quad i = 1,..,n$$

For now, we consider a single objective function, $J(\mathbf{x})$.

There are n design variables, and a total of m constraints ($m=m_1+m_2$).

The bounds are known as side constraints.

Karush-Kuhn-Tucker Conditions

If **x*** is optimum, these conditions are satisfied:

1. **x*** is feasible
2. $\lambda_j \, g_j(\mathbf{x}^*) = 0, \;\; j=1,..,m_1$ and $\lambda_j \geq 0$
3. $$\nabla J(x^*) + \sum_{j=1}^{m_1} \lambda_j \nabla g_j(x^*) + \sum_{k=1}^{m_2} \lambda_{m_1+k} \nabla h_k(x^*) = 0$$

   $$\lambda_j \geq 0$$

   λ_{m_1+k} unrestricted in sign

The KKT conditions are necessary and sufficient if the design space is convex.

Karush-Kuhn-Tucker Conditions: Interpretation

Condition 1: the optimal design satisfies the constraints

Condition 2: if a constraint is not precisely satisfied, then the corresponding Lagrange multiplier is zero

- *the j^{th} Lagrange multiplier represents the sensitivity of the objective function to the j^{th} constraint*
- *can be thought of as representing the "tightness" of the constraint*
- *if λ_j is large, then constraint j is important for this solution*

Condition 3: the gradient of the Lagrangian vanishes at the optimum

Optimization for Engineering Problems

- Most engineering problems have a complicated design space, usually with several local optima
- Gradient-based methods can have trouble converging to the global optimum, and sometimes fail to find even a local optimum
- Heuristic techniques offer no guarantee of optimality, neither global nor local
- Your post-optimality analysis should address the question:
  - How confident are you that you have found the global optimum?
  - Do you actually care?

Optimization for Engineering Problems

- Usually cannot guarantee that absolute optimum is found
  - local optima
  - numerical ill-conditioning
    - → gradient-based techniques should be started from several initial solutions
    - → best solution from a heuristic technique should be checked with KKT conditions or used as an initial condition for a gradient-based algorithm
- Can determine mathematically if have relative minimum but KKT conditions are only sufficient if the problem is convex
- It is very important to interrogate the “optimum” solution

Termination Criteria: Gradient-Based

Gradient-based algorithm is terminated when ...

an acceptable solution is found

OR

algorithm terminates unsuccessfully

Need to decide:

- when an acceptable solution is found
- when to stop the algorithm with no acceptable solution
  - when progress is unreasonably slow
  - when a specified amount of resources have been used (time, number of iterations, etc.)
  - when an acceptable solution does not exist
  - when the iterative process is cycling

Termination Criteria: Gradient-Based

- Is $\mathbf{x}^k$ an acceptable solution?
  - does $\mathbf{x}^k$ *almost* satisfy the conditions for optimality?
  - has the sequence $\{ \mathbf{x}^k \}$ converged?

- Often the first question is difficult to test
- The tests are often approximate
- Often rely on the answer to the second question
- But convergence to a non-optimal solution or extended lack of progress can look the same as convergence to the correct solution!
- No one set of termination criteria is suitable for all optimization problems and all methods

Termination Criteria: Gradient-Based

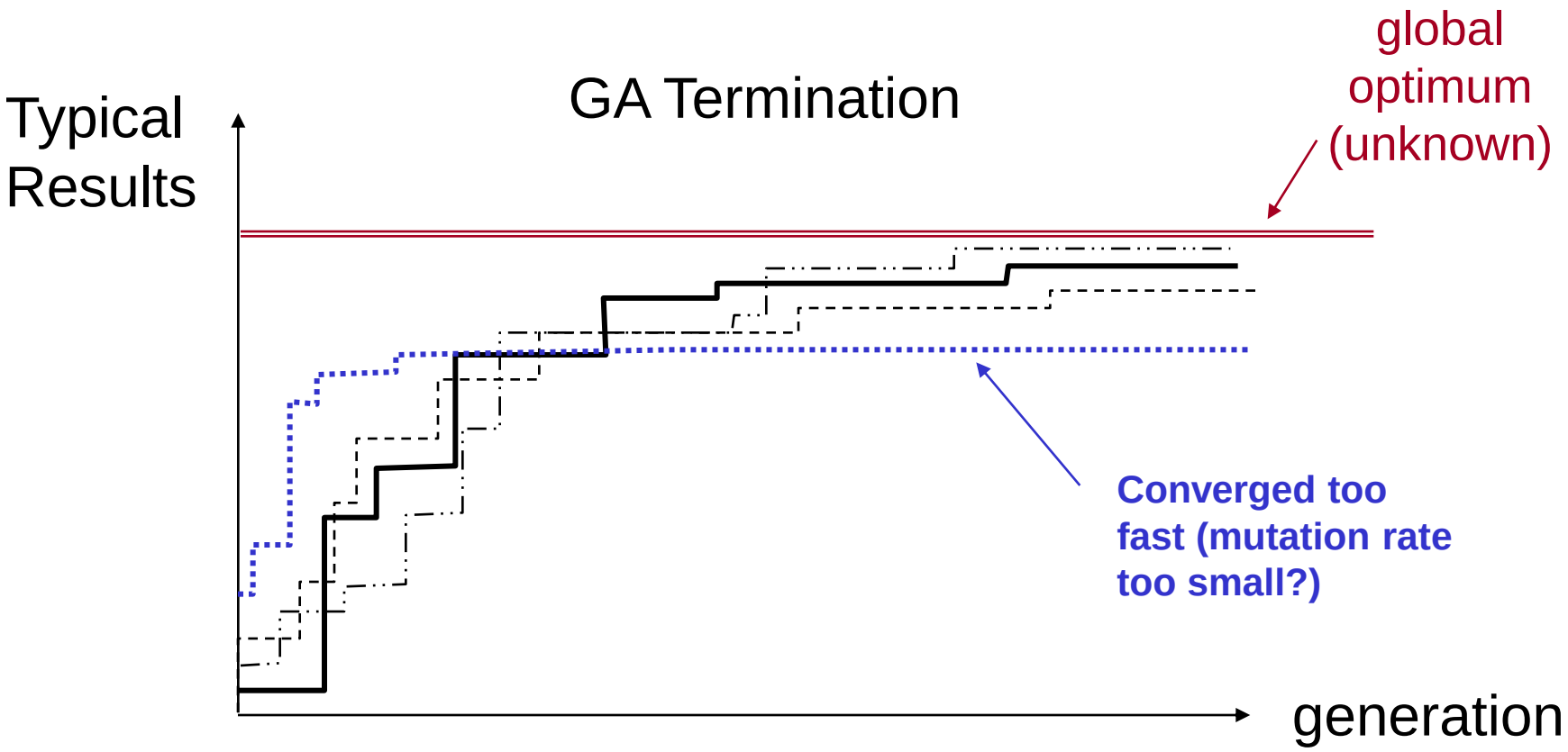
Has the sequence $\{ \mathbf{x}^k \}$ converged?

Ideally: $\left| J^k - J^* \right| \leq \varepsilon$ or $\left\| \mathbf{x}^k - \mathbf{x}^* \right\| \leq \varepsilon$

In practice: $\left| J^k - J^{k+1} \right| \leq \varepsilon$ or $\left\| \mathbf{x}^k - \mathbf{x}^{k+1} \right\| \leq \varepsilon$

Also should check constraint satisfaction: $\left\| \mathbf{g}^k \right\| \leq \varepsilon$

Termination Criteria: Heuristics

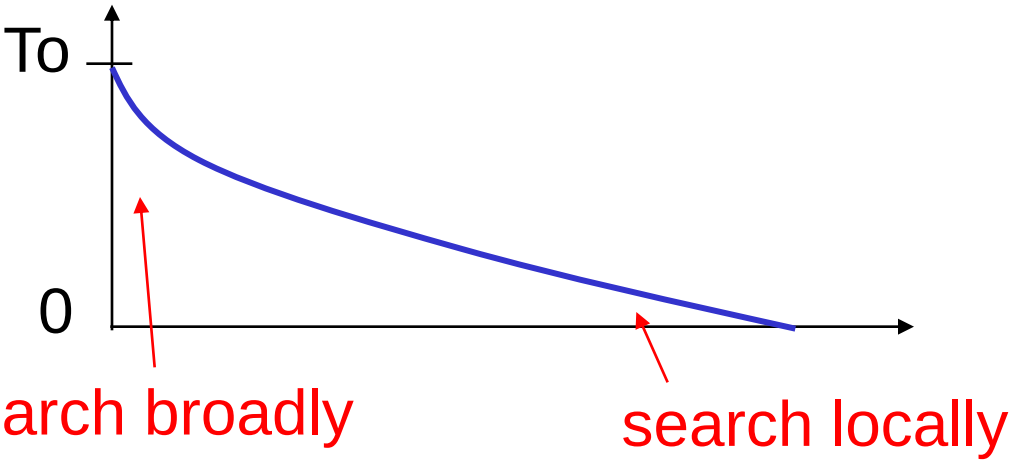


- *GEN = max(GEN):* maximum # of generations reached
- Stagnation in Fitness, no progress made on objective
- Dominant Schema have emerged

Termination Criteria: Heuristics

- Simulated Annealing - cooling schedule: T(k)=f(k, To)

Search stops when $T(k)<\varepsilon$, where $\varepsilon>0$, but small



- **Tabu** search termination
  - Usually after a predefined number of iterations
  - Best solution found is reported
  - No guarantee of optimality

Post-Optimality Analysis

- Already talked about **sensitivity** analysis (Lecture 9)
  - How does optimal solution change as a parameter is varied?
  - How does optimal solution change as a design variable value is varied?
  - How does optimal solution change as constraints are varied?
- Also would like to understand key drivers in optimal design
- We also saw in Lecture 9 that the values of the Lagrange multipliers at the optimal solution give information on how modifying the constraints affects the solution.

Objective Function Behavior

Consider the quadratic function:

$$\Phi(\mathbf{x}) = \mathbf{c}^T\mathbf{x} + \frac{1}{2}\mathbf{x}^T\mathbf{H}\mathbf{x}$$

The behavior of $\Phi(\mathbf{x})$ in the neighborhood of a local minimum is determined by the eigenvalues of $\mathbf{H}$.

$$\Phi(\hat{\mathbf{x}} + \alpha\mathbf{p}) = \mathbf{c}^T(\hat{\mathbf{x}} + \alpha\mathbf{p}) + \frac{1}{2}(\hat{\mathbf{x}} + \alpha\mathbf{p})^T\mathbf{H}(\hat{\mathbf{x}} + \alpha\mathbf{p})$$

$$= \mathbf{c}^T\hat{\mathbf{x}} + \frac{1}{2}\hat{\mathbf{x}}^T\mathbf{H}\hat{\mathbf{x}} + \alpha\mathbf{c}^T\mathbf{p} + \frac{1}{2}\alpha^2\mathbf{p}^T\mathbf{H}\mathbf{p} + \frac{\alpha}{2}\hat{\mathbf{x}}^T\mathbf{H}\mathbf{p} + \frac{\alpha}{2}\mathbf{p}^T\mathbf{H}\hat{\mathbf{x}}$$

$$\Phi(\hat{\mathbf{x}} + \alpha\mathbf{p}) = \Phi(\hat{\mathbf{x}}) + \alpha\mathbf{p}^T(\mathbf{H}\hat{\mathbf{x}} + \mathbf{c}) + \frac{1}{2}\alpha^2\mathbf{p}^T\mathbf{H}\mathbf{p}$$

Objective Function Behavior

$$\Phi(\hat{\mathbf{x}} + \alpha\mathbf{p}) = \Phi(\hat{\mathbf{x}}) + \alpha\mathbf{p}^T(\mathbf{H}\hat{\mathbf{x}} + \mathbf{c}) + \frac{1}{2}\alpha^2\mathbf{p}^T\mathbf{H}\mathbf{p}$$

Consider the neighborhood of the optimal solution:

$$\hat{\mathbf{x}} = \mathbf{x}^*$$

$$\nabla\Phi(\mathbf{x}^*) = \mathbf{H}\mathbf{x}^* + \mathbf{c} = 0 \qquad \text{or} \qquad \mathbf{H}\mathbf{x}^* = -\mathbf{c}$$

$$\Phi(\mathbf{x}^* + \alpha\mathbf{p}) = \Phi(\mathbf{x}^*) + \frac{1}{2}\alpha^2\mathbf{p}^T\mathbf{H}\mathbf{p}$$

The behavior of Φ in the neighborhood of $\mathbf{x}^*$ is determined by $\mathbf{H}$.

Objective Function Behavior

Let $\mathbf{v}_j$, λ_j be the j^{th} eigenvector and eigenvalue of **H**:

$$H\mathbf{v}_j = \lambda_j \mathbf{v}_j$$

and $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$ since **H** is symmetric.

Consider the case when $\mathbf{p}=\mathbf{v}_j$:

$$\Phi(\mathbf{x}^* + \alpha \mathbf{v}_j) = \Phi(\mathbf{x}^*) + \frac{1}{2}\alpha^2 \mathbf{v}_j^T \mathbf{H} \mathbf{v}_j$$

$$= \Phi(\mathbf{x}^*) + \frac{1}{2}\alpha^2 \lambda_j$$

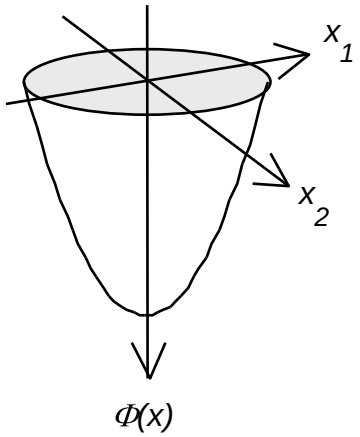
As we move away from $\mathbf{x}^*$ along the direction $\mathbf{v}_j$, the change in the objective depends on the sign of λ_j.

Objective Function Behavior

$$\Phi(\mathbf{x}^* + \alpha \mathbf{v}_j) = \Phi(\mathbf{x}^*) + \frac{1}{2}\alpha^2 \lambda_j$$

- If λ_j>0, Φ increases
- If λ_j<0, Φ decreases
- If λ_j=0, Φ remains constant
- When all λ_j>0, x* is a minimum of Φ
- The contours of Φ are ellipsoids
  - principal axes in directions of eigenvectors
  - lengths of principal axes inversely proportional to square roots of eigenvalues

Objective Function Behavior



direction of first eigenvector

$\mathbf{v}_1$

$\mathbf{v}_2$

direction of second eigenvector

contours of constant Φ

- If $\lambda_2=\lambda_1$, the contours are circular
- As λ_2/λ_1 gets very small, the ellipsoids get more and more stretched
- If any eigenvalue is very close to zero, Φ will change very little when moving along that eigenvector

Hessian Condition Number

- The condition number of the Hessian is given by

$$\kappa(\mathbf{H}) = \frac{\lambda_1}{\lambda_n}$$

- When $\kappa(\mathbf{H})=1$, the objective function contours are circular
- As $\kappa(\mathbf{H})$ increases, the contours become elongated
- If $\kappa(\mathbf{H})>>1$, the change in the objective function due to a small change in **x** will vary radically depending on the direction of perturbation
- $\kappa(\mathbf{H})$ can be computed via a Cholesky factorization ($\mathbf{H}=\mathbf{L}\mathbf{D}\mathbf{L}^T$)

Scaling

- In theory we should be able to choose any scaling of the design variables, constraints and objective functions without affecting the solution
- In practice, the scaling can have a large effect on the solution

$\Rightarrow$ numerical accuracy, numerical conditioning

- From Papalambros, p. 352: "*scaling is the single most important, but simplest, reason that can make the difference between success and failure of a design optimization algorithm*"

Scaling

$$\begin{aligned}\min\ & x_1^2 + 3x_2 + x_3 \\ \text{s.t.}\ & 5x_1 + 2x_2^3 \geq 1 \\ & 6x_1^2 - 3x_3 \geq 2 \\ & x_i \geq 0\end{aligned}$$

scale objective

$$\begin{aligned}\min\ & 10x_1^2 + 30x_2 + 10x_3 + 28 \\ \text{s.t.}\ & 5x_1 + 2x_2^3 \geq 1 \\ & 6x_1^2 - 3x_3 \geq 2 \\ & x_i \geq 0\end{aligned}$$

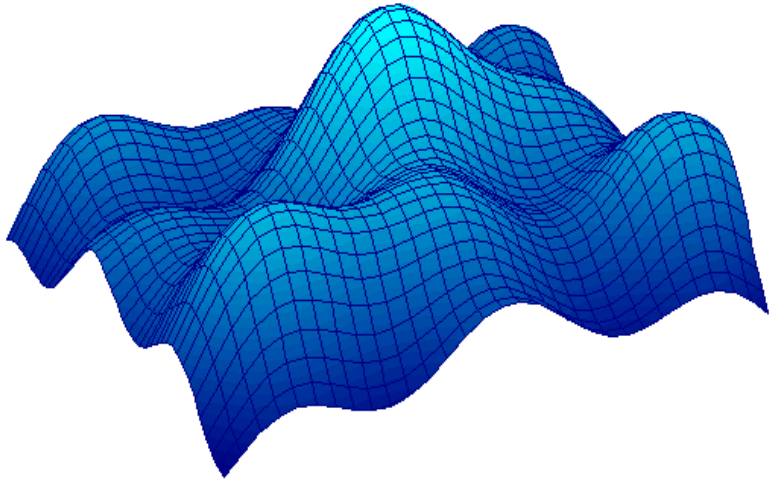
scale constraint

scale design variable

$$\begin{aligned}\min\ & x_1^2 + 3x_2 + x_3 \\ \text{s.t.}\ & 50x_1 + 20x_2^3 \geq 10 \\ & 6x_1^2 - 3x_3 \geq 2 \\ & x_i \geq 0\end{aligned}$$

$$\begin{aligned}\min\ & x_1^2 + 3x_2 + 10\tilde{x}_3 \\ \text{s.t.}\ & 5x_1 + 2x_2^3 \geq 1 \\ & 6x_1^2 - 30\tilde{x}_3 \geq 2 \\ & x_i \geq 0\end{aligned}$$

MATLAB Demo: Scaling



Design Variable Scaling

- In aircraft design, we are combining variables of very different magnitudes
- e.g. aircraft range ~ 10^6 m

  wing span ~ 10^1 m

  skin thickness ~ 10^{-3} m
- Need to non-dimensionalize and scale variables to be of similar magnitude in the region of interest
- Want each variable to be of similar weight during the optimization

Design Variable Scaling

- Consider the transformation **x** = **Ly**, where **L** is an arbitrary nonsingular transformation matrix
- If at any iteration in the algorithm, $\mathbf{x}^k = \mathbf{L}\mathbf{y}^k$ (using exact arithmetic), then the algorithm is said to be **scale invariant**
- In practice, this property will not hold
- The conditioning of the Hessian matrix at **x*** gives us information about the scaling of the design variables
- When **H**(**x**) is ill-conditioned, $J(\mathbf{x})$ varies much more rapidly along some directions than along others
- The ill-conditioning of the Hessian is a form of bad scaling, since similar changes in ||**x**|| do not cause similar changes in J

Design Variable Scaling

- We saw that if $\mathbf{H}(\mathbf{x}^*)$ is ill-conditioned ($\kappa(\mathbf{H})>>1$), then the change in the objective function due to a small change in $\mathbf{x}$ will vary radically depending on the direction of perturbation
- J($\mathbf{x}$) may vary so slowly along an eigenvector associated with a near-zero eigenvalue that changes that should be significant are lost in rounding error
- We would like to scale our design variables so that $\kappa(\mathbf{H})\sim1$
- In practice this may be unachievable (often we don't know $\mathbf{H}$)
- Often, a diagonal scaling is used where we consider only the diagonal elements of $\mathbf{H}(\mathbf{x}^0)$ and try to make them close to unity

Objective Function Scaling

- In theory, we can multiply $J(\mathbf{x})$ by any constant or add a constant term, and not affect the solution
- In practice, it is generally desirable to have $J \sim O(1)$ in the region of interest
- Algorithms can have difficulties if $J(\mathbf{x})$ is very small everywhere, since convergence is usually tested using some small quantity
- Inclusion of a constant term can also cause difficulties, since the error associated with the sum may reflect the size of the constant rather than the size of $J(\mathbf{x})$

  e.g. $\min x_1^2+x_2^2$ vs. $\min x_1^2+x_2^2+1000$

Constraint Scaling

A well scaled set of constraints has two properties:

- each constraint is well conditioned with respect to perturbations in the design variables
- the constraints are balanced with respect to each other, *i.e.* all constraints have an equal weighting in the optimization

The scaling of constraints can have a major effect on the path chosen by the optimizer. For example, many algorithms maintain a set of active constraints and from one iteration to the next they interchange one active and one inactive constraint. Constraint scaling impacts the selection of which constraint to add or delete.

Scaling

Two reasons to scale:

1. At the beginning of optimization, using $\mathbf{x}^0$, to improve algorithm performance (e.g. decrease number of iterations).

2. At the end of optimization, using $\mathbf{x}^*$, to make sure that the "optimal" solution is indeed the best we can achieve.

   $\Rightarrow$ BWB example

Scaling Example

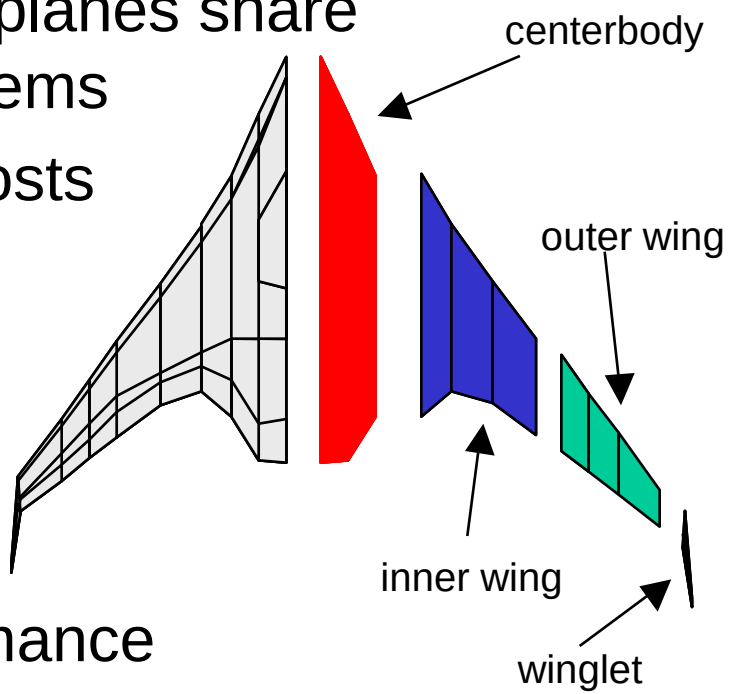
- Consider optimization of the BWB
- Rather than optimizing just a single aircraft, we want to design a family of aircraft
- This family has commonality – the planes share common parts, planforms and systems
- Commonality can help to reduce costs

  e.g. manufacturing costs

  design costs

  spare parts

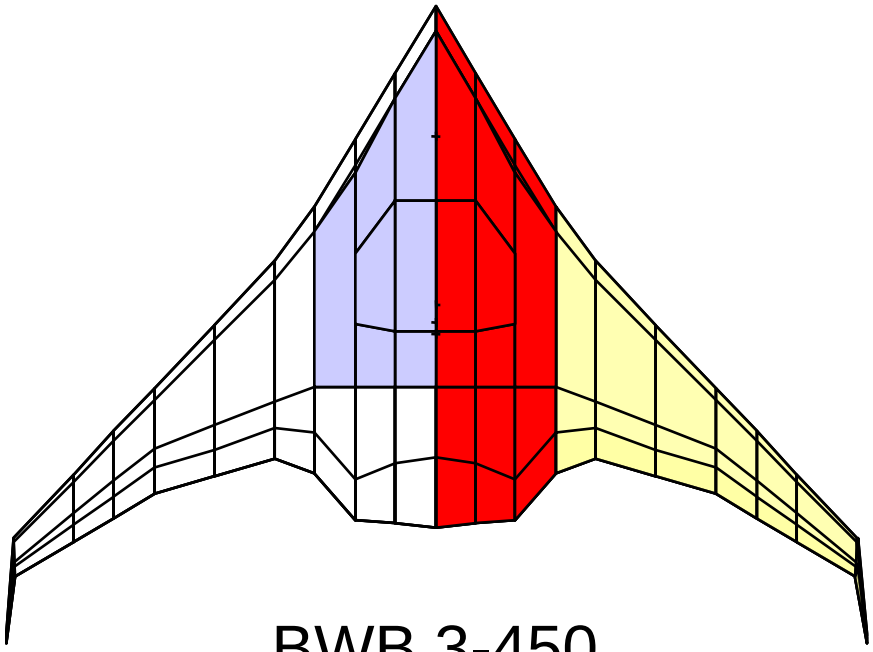
  crew training



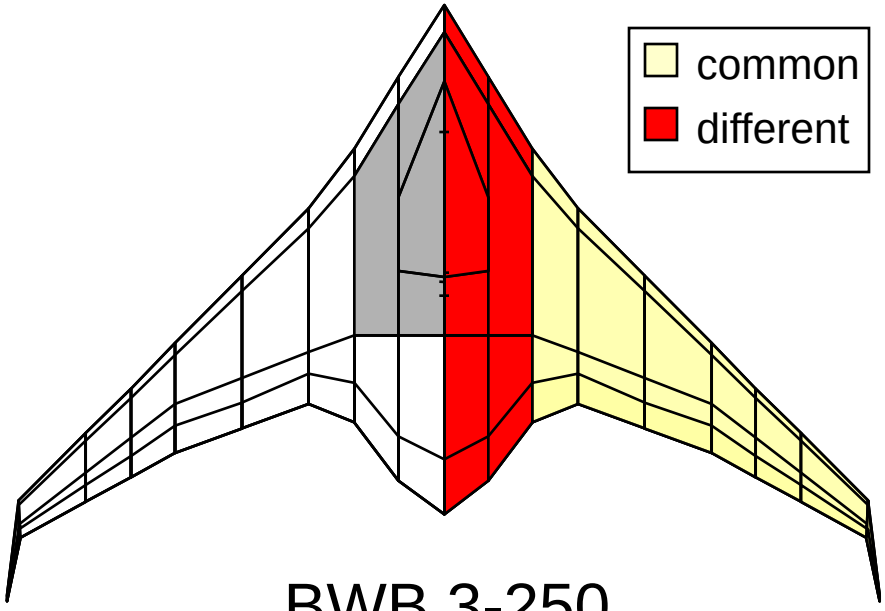
- But will require a trade with performance
- It is easier to achieve commonality with the BWB than with conventional tube & wing aircraft

Scaling Example

- Consider a two-aircraft family with common wings:



BWB 3-450
475 passengers, 8550 nm

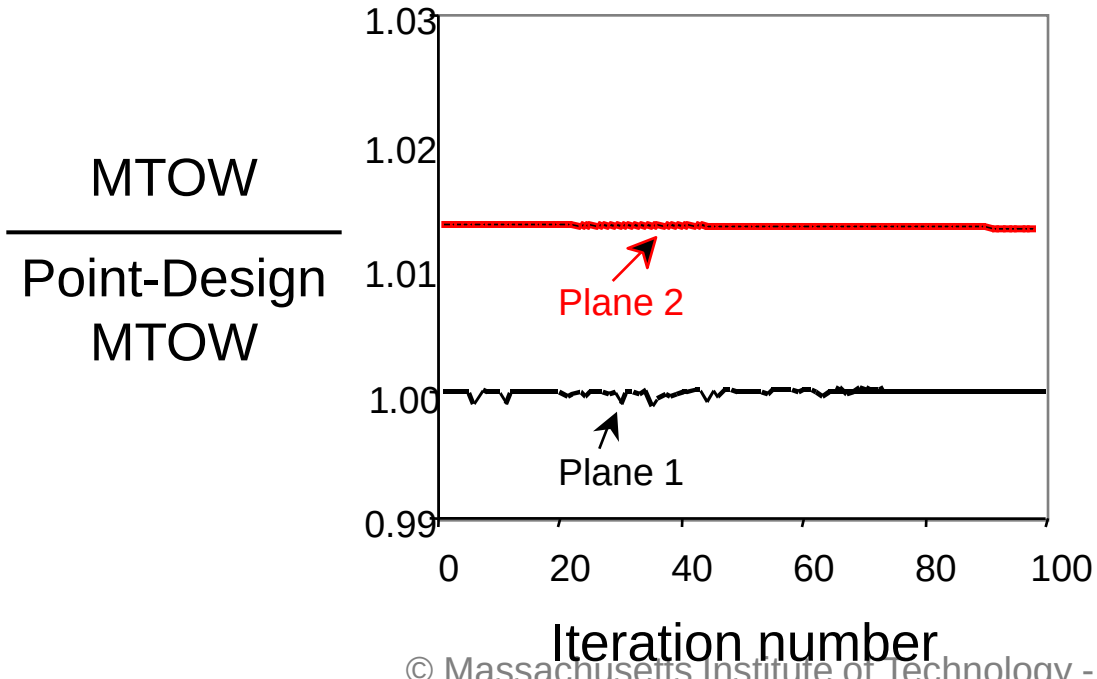


BWB 3-250
272 passengers, 8550 nm

- We set up an MDO framework for each aircraft
- We link the variables that are common between the two aircraft

Scaling Example

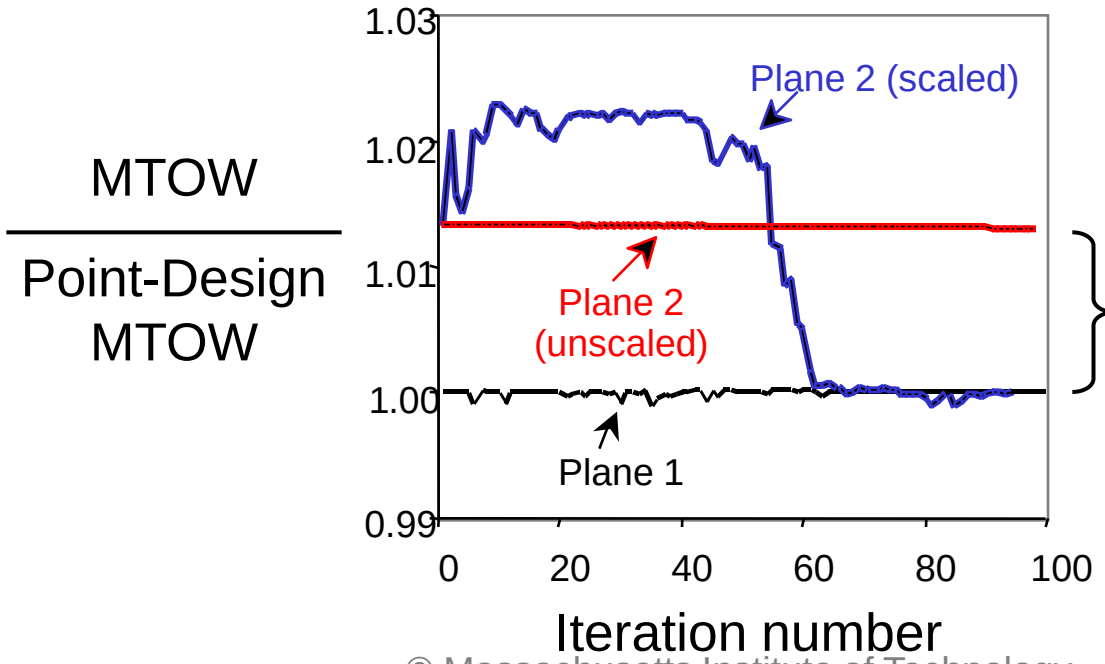
- In order to test the framework, we first try to optimize a family with no commonality
- We should get the point-design solutions for each plane
- However, the algorithm (SQP) does not converge to the correct solution for the smaller plane!



MTOW results for last 100 iterations

Scaling Example

- When we look at the Hessian matrix at the solution, we see that the diagonal entries corresponding to the design variables of Plane 2 are badly scaled
- We rescale these variables, and now the algorithm converges



MTOW results for last 100 iterations

Lecture Summary

- Optimality Conditions
- Objective Function Behavior
- Scaling
- In practice, optimization can be very difficult to implement: algorithms can behave badly, and it can be difficult (impossible) to verify that a solution is truly optimal
- Numerical accuracy is a real issue and can drastically affect results, especially if the problem is not well scaled
- It is very important to interrogate the “optimum” solution carefully
- The mathematical tools you learn are very useful in practice, but they must be applied carefully!

References

- Gill, P.E, Murray, W. and Wright, M. H., *Practical Optimization*, Academic Press, London, 1981
- Vanderplaats, G.N., *Numerical Optimization Techniques for Engineering Design*, Vanderplaats R&D, 1999.
- Willcox, K. and Wakayama, S., “Simultaneous Optimization of a Multiple-Aircraft Family,” Journal of Aircraft, Vol. 41, No.4, July 2003, pp. 616-622.

ESD.77 / 16.888 Multidisciplinary System Design Optimization

Spring 2010