

Lecture 01: Multidisciplinary System Analysis and Design Optimization (MSADO)

Introduction

January 21, 2015

Prof. Douglas Allaire

Introductions

Douglas Allaire, Ph.D.

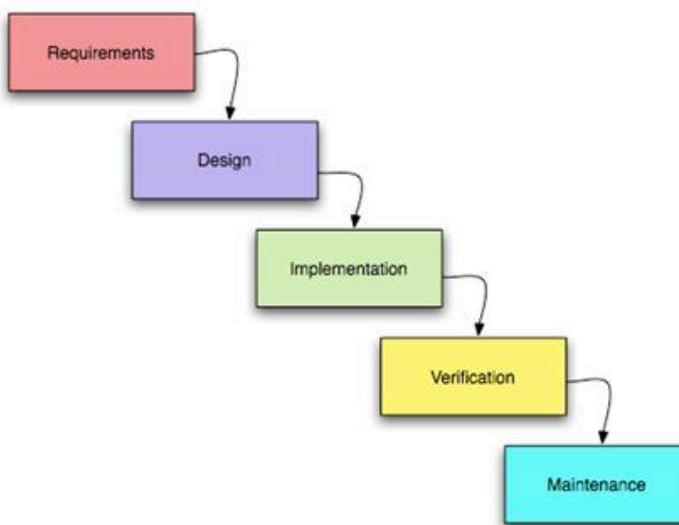
Assistant Professor, dallaire@tamu.edu

Today's Topics

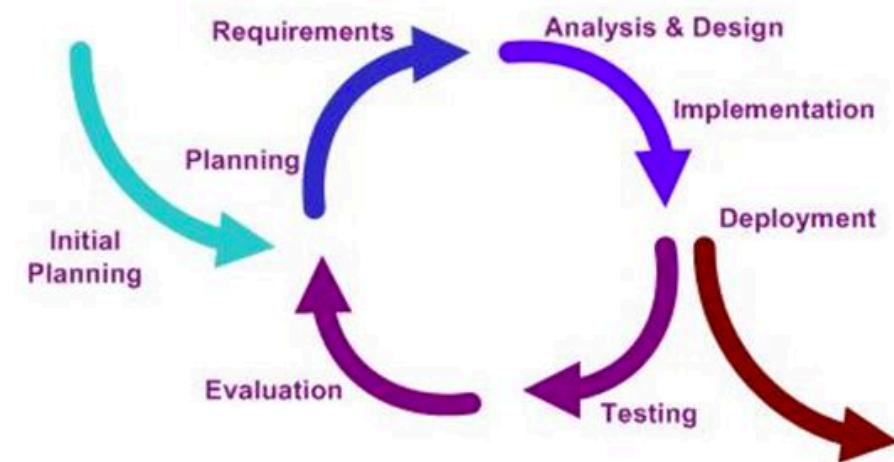
- Course Rationale
- Role of MSADO in Systems Engineering
- Learning Objectives
- A historical perspective on MDO
- MSADO Framework introduction

Course Rationale

Computational Design and Concurrent Engineering (CE) are becoming an increasingly important part of the Product Development Process (PDP) in Industry



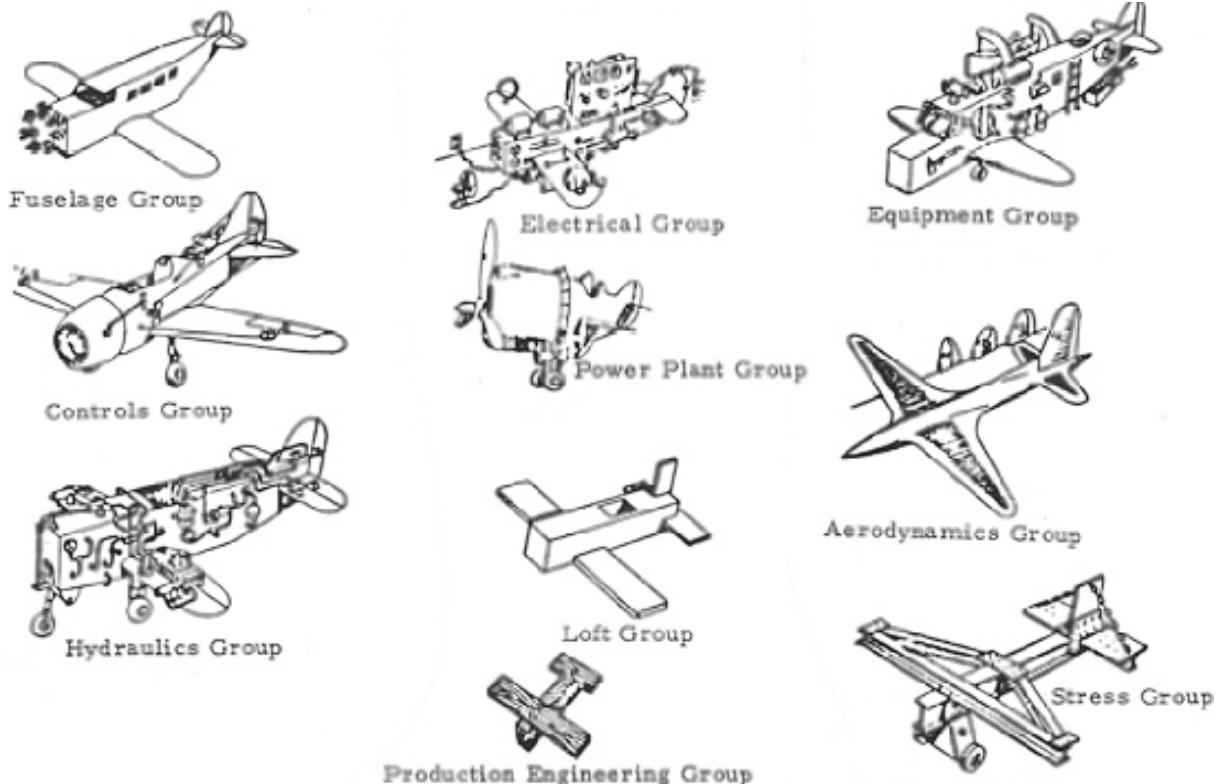
VS.



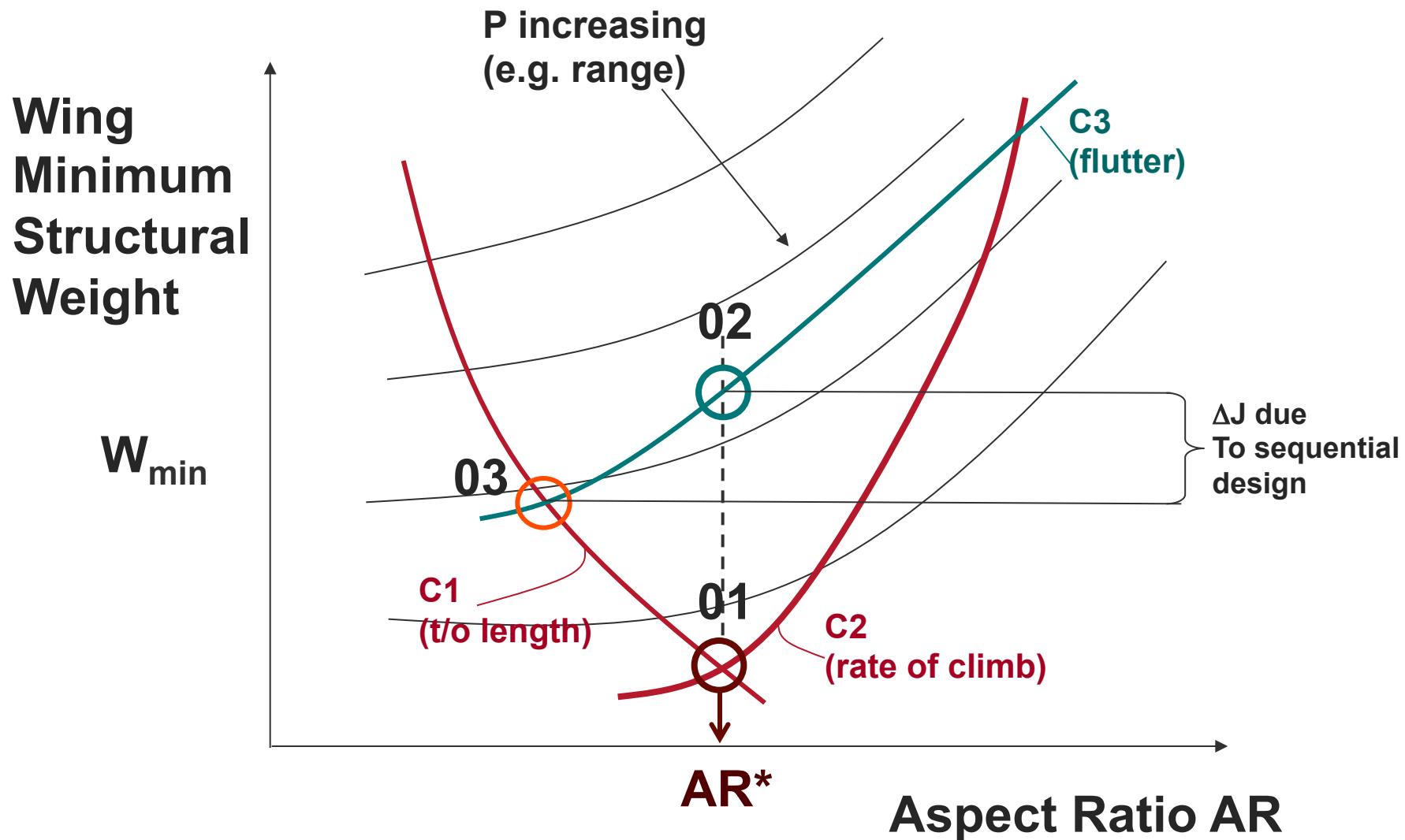
A collection of subsystems?



?
=



Sequential vs. Concurrent Design



Role of MSADO in Engineering Systems

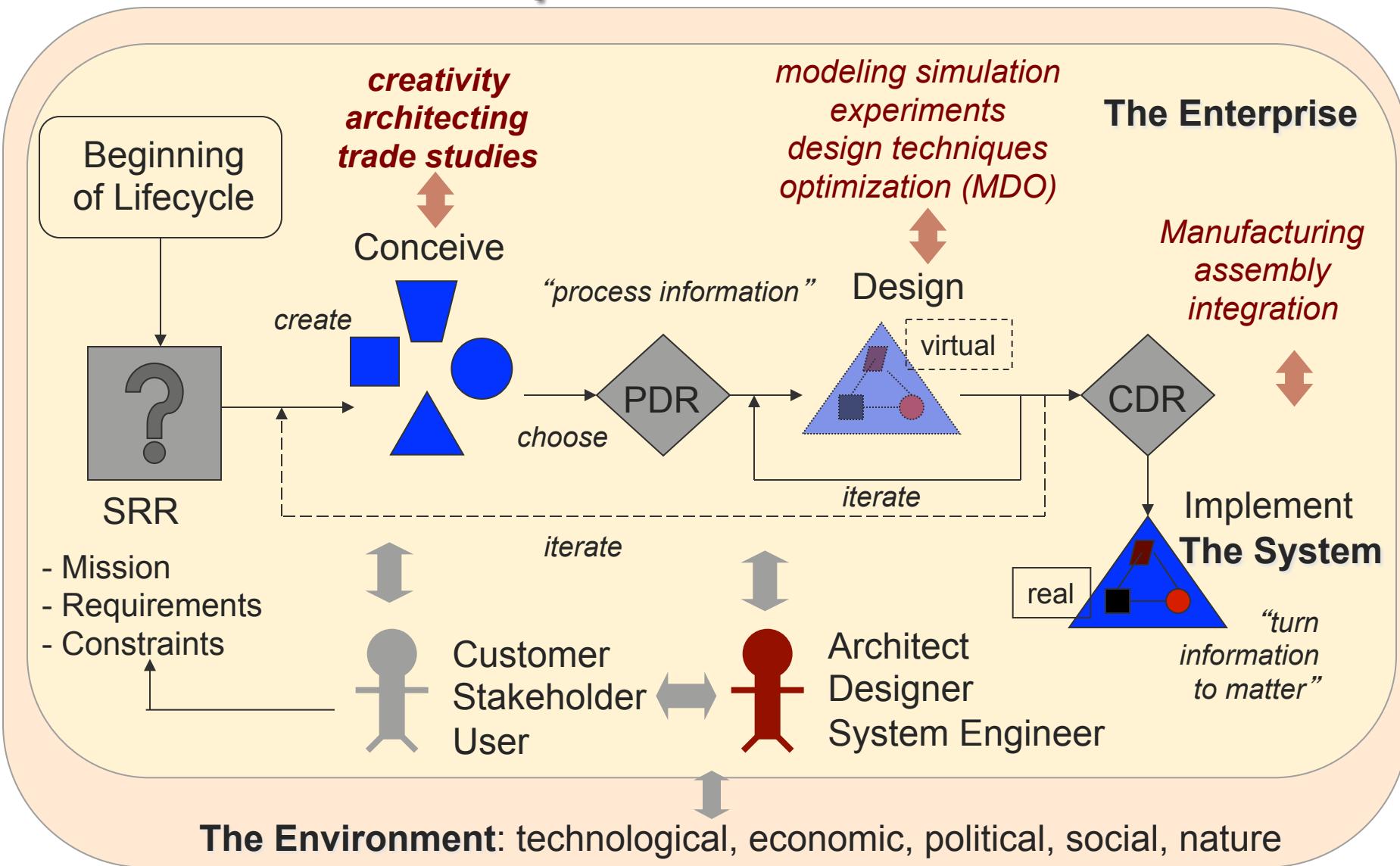
Goal: Create advanced and complex engineering systems that must be competitive not only in terms of performance, but also in terms of life-cycle value.

Need: A rigorous, quantitative multidisciplinary design methodology that can work hand-in-hand with the intuitive non-quantitative and creative side of the design process.



This class presents the current state-of-the-art in concurrent, multidisciplinary design optimization (MDO)

Product Development Process



Example: BWB Aircraft

Boeing Blended Wing Body Concept



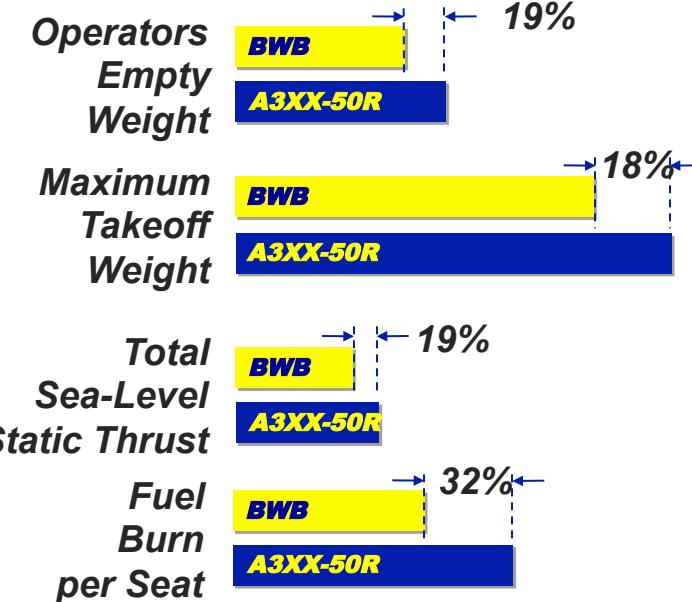
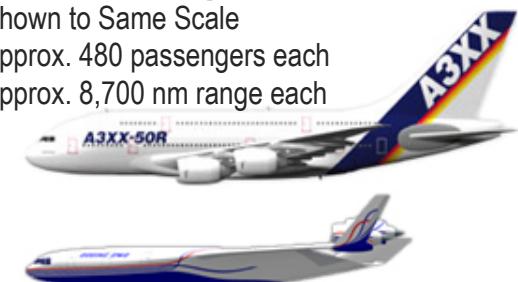
Goal: Find a design for a family of blended wing aircraft that will combine aerodynamics, structures, propulsion and controls such that a competitive system emerges - as measured by a set of metrics that matter to the operator.

Aircraft Comparison

Shown to Same Scale

Approx. 480 passengers each

Approx. 8,700 nm range each

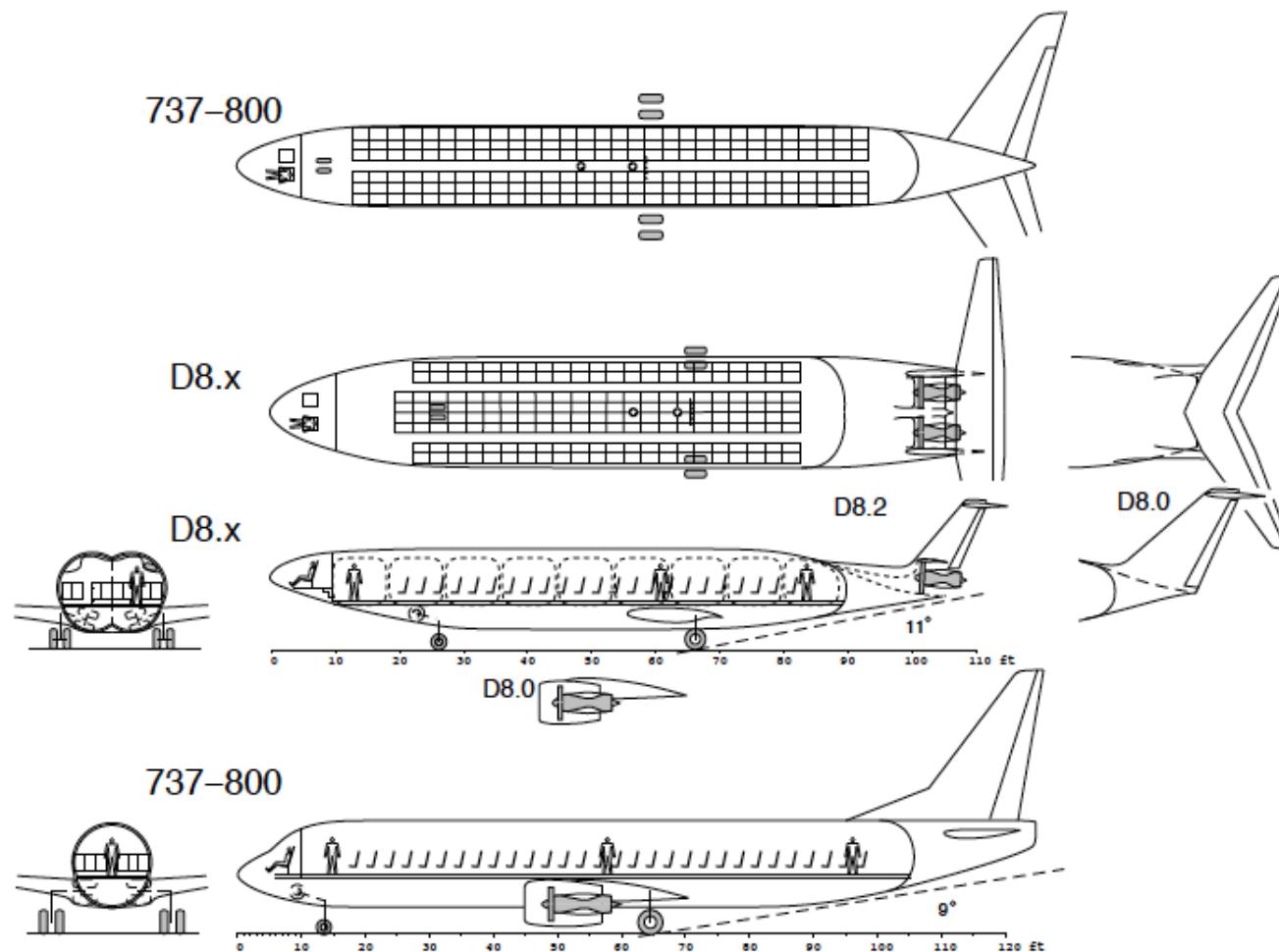


© Boeing

MIT-Aurora D8 ‘Double Bubble’



D8 Concept – Fuselage Comparison



Course Objectives

The course will

- Enhance Texas A&M's offerings in the area of simulation and optimization of multidisciplinary systems during the conceive and design phases
- Develop and codify a normative (prescriptive) approach to multidisciplinary modeling and quantitative assessment of new or existing system/product designs
- Engage both faculty and graduate students in the emerging research field of MDO

Learning Objectives

The students will

- 1) Learn how MSADO can support the product development process of complex, multidisciplinary engineered systems
- 2) Learn how to rationalize and quantify a system architecture or product design problem by selecting appropriate objective functions, design variables, parameters, and constraints
- 3) Subdivide a complex system into smaller disciplinary models, manage their interfaces and reintegrate them into an overall system model

Learning Objectives

- 4) Be able to use various optimization techniques such as sequential quadratic programming, simulated annealing or genetic algorithms and select the ones most suitable to the problem at hand
- 5) Perform a critical evaluation and interpretation of simulation and optimization results, including sensitivity analysis and exploration of performance, cost, and risk tradeoffs
- 6) Be familiar with the basic concepts of multiobjective optimization, including the conditions for optimality and the computation of the pareto front

Learning Objectives

- 7) Sharpen their presentation skills, acquire critical reasoning with respect to the validity and fidelity of their MSADO models and experience the advantages and challenges of teamwork

MSADO Pedagogy

- Assignments A1-A5
- Lectures
- Readings
- Class Project
- eCampus

Assignments

Part (a)

Small, simple problems to be solved individually, many just by hand or with a computer. Goal is to ensure learning of the key ideas regardless of chosen project

Part (b)

Application of theory to a project of your choice from either existing class projects or a project related to your research. Solution individually or in teams of two or three.

- Assignments A1-A5 scheduled bi-weekly.
- Usually 2 weeks given to complete.

Lectures

Schedule will be posted on eCampus
(eventually...)

Module 1: Problem Formulation and Setup

Module 2: Optimization and Search Methods

Module 3: Multiobjective Challenges

Module 4: Implementation Issues and Applications

Class Project

Form teams of 2-3 students

Alternative A – Use a pre-existing project

These are prepared simulation codes that you can use as your class project for solving part (b) of the assignments in lieu of a personal research-related problem:

(C-Code) – general aviation aircraft

(MATLAB) – satellite constellations

(Excel) – space shuttle fuel tank

Alternative B – Formulate your own project (preferred)

- This is an opportunity to push your research forward
- Must be a design problem, must be multidisciplinary
- Write 1 page project proposal for A1 (part b)

Tools and Infrastructure

- **Physical Infrastructure**
 - May have some organized labs to support assignments in lecture (will require laptop computers)
 - MEEN labs
- **Computational Infrastructure**
 - Individual PC/laptop or MEEN computers
- **Software Infrastructure**
 - Matlab (w/Optimization toolbox)
 - Excel (w/solver)
 - Write your own optimizer (C/C++/Python/other)
 - Other packages

Textbook

Panos Y. Papalambros and Douglass J. Wilde, “Principles of Optimal Design – Modeling and Computation”, 2nd edition, ISBN 0 521 62727 3, (paperback), Cambridge University Press, 2000 – Recommended
<http://www.optimaldesign.org>

Others (Recommended):

Garret N. Vanderplaats, “Numerical Optimization Techniques for Engineering Design”, ISBN 0-944956-01-7, Third Edition, Vanderplaats Research & Development Inc., 2001

R. E. Steuer.” Multiple Criteria Optimization: Theory, Computation and Application”. Wiley, New York, 1986

David E. Goldberg, “Genetic Algorithms – in Search, Optimization & Machine Learning”, Addison –Wesley, ISBN 0 201 15767-5, 1989 -

Grading

Assignments A1-A5*	50%
Project Presentation	20%
Final Report (Paper)	20%
Active Participation	10%

No mid-term or final exams

* *Each assignment is 10%*

eCampus

- Lecture notes
- Assignments
- Readings
- Grades

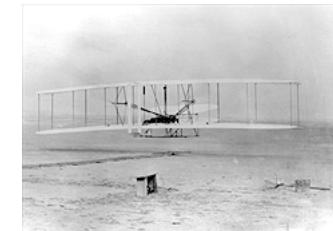
Historical Perspective on MDO

The need for MDO can be better understood by considering the historical context of progress in aerospace vehicle design.

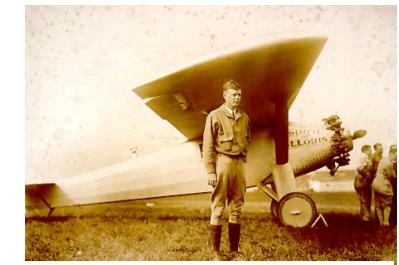
- **1903** Wright Flyer makes the first manned and powered flight.
- **1927** Charles Lindbergh crosses the Atlantic solo and nonstop
- **1935 DC-3** enters service (12,000 to be produced)
- **1958 B707** enters service
- **1970 B747** enters service
- **1974 A300** enters service
- **1976 Concorde** enters service



Concorde



Wright



Spirit of St. Louis



B707



DC3



A300



B747

1970-1990 Cold War and Maturity

- Big slump in world economy (“oil crisis” 1973), airline industry and end of Apollo program leads to a reduction of engineering workforce around 25%
- Two major new developments: Computer aided design (CAD), Procurement policy changes for airlines and the military
- Earlier quest for maximum performance has been superseded by need for a “balance” among performance, life-cycle cost, reliability, maintainability, and other “-ilities”
- Reflected by growth in design requirements (see next slide). Competition in airline industry drives operational efficiency.

Growth in design requirements

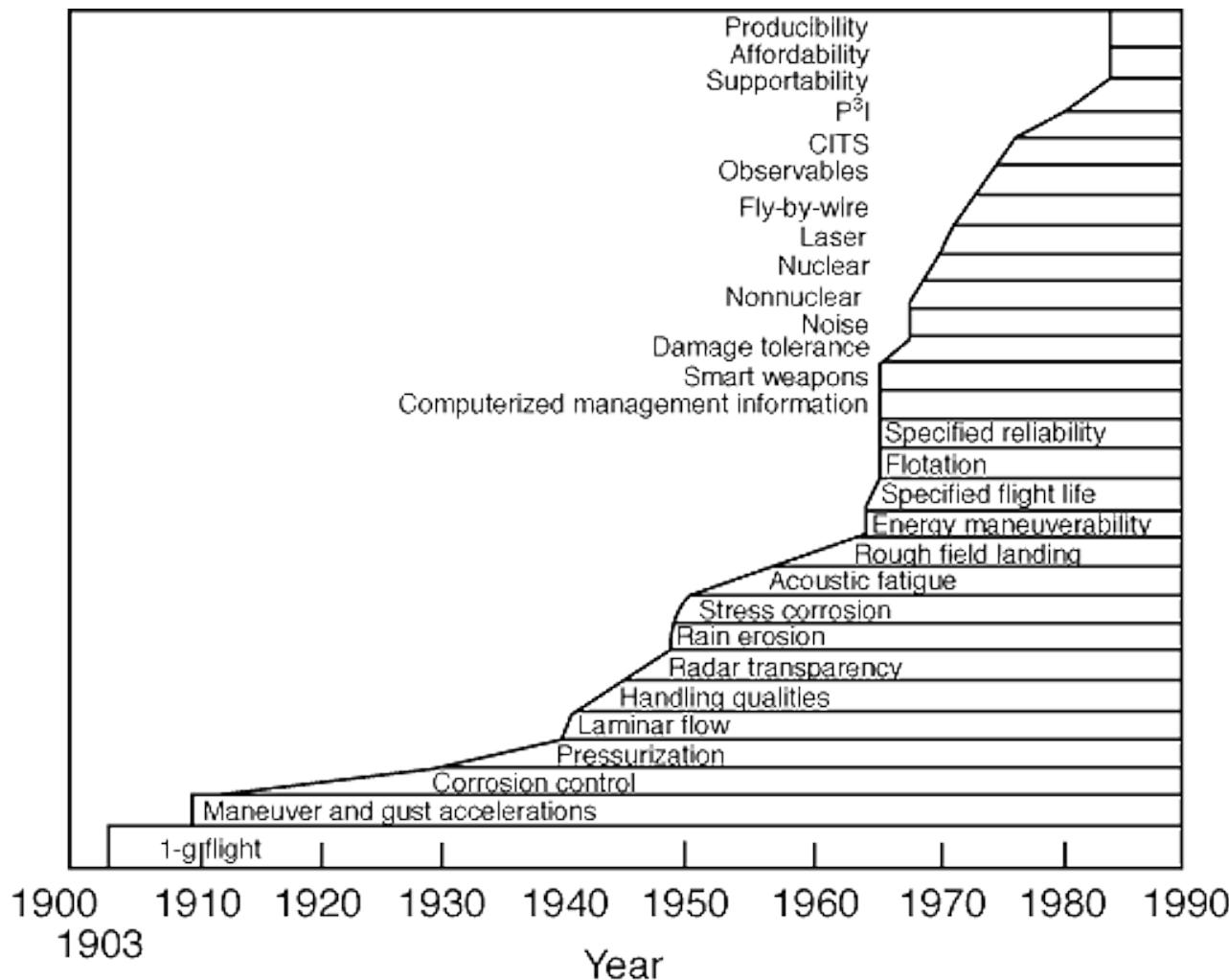
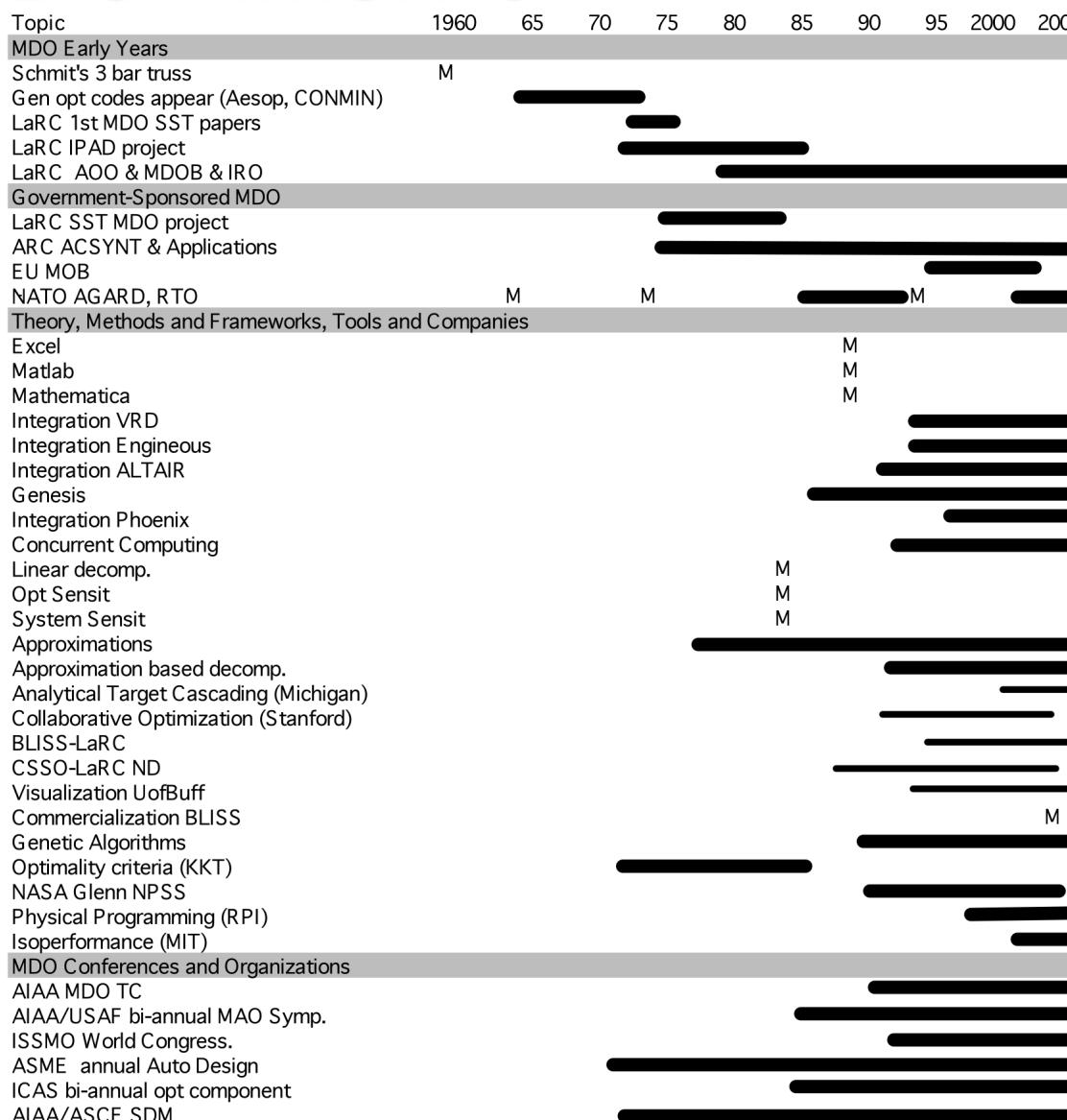


Fig.1 Design requirements growth for aerospace vehicles.

1990-present

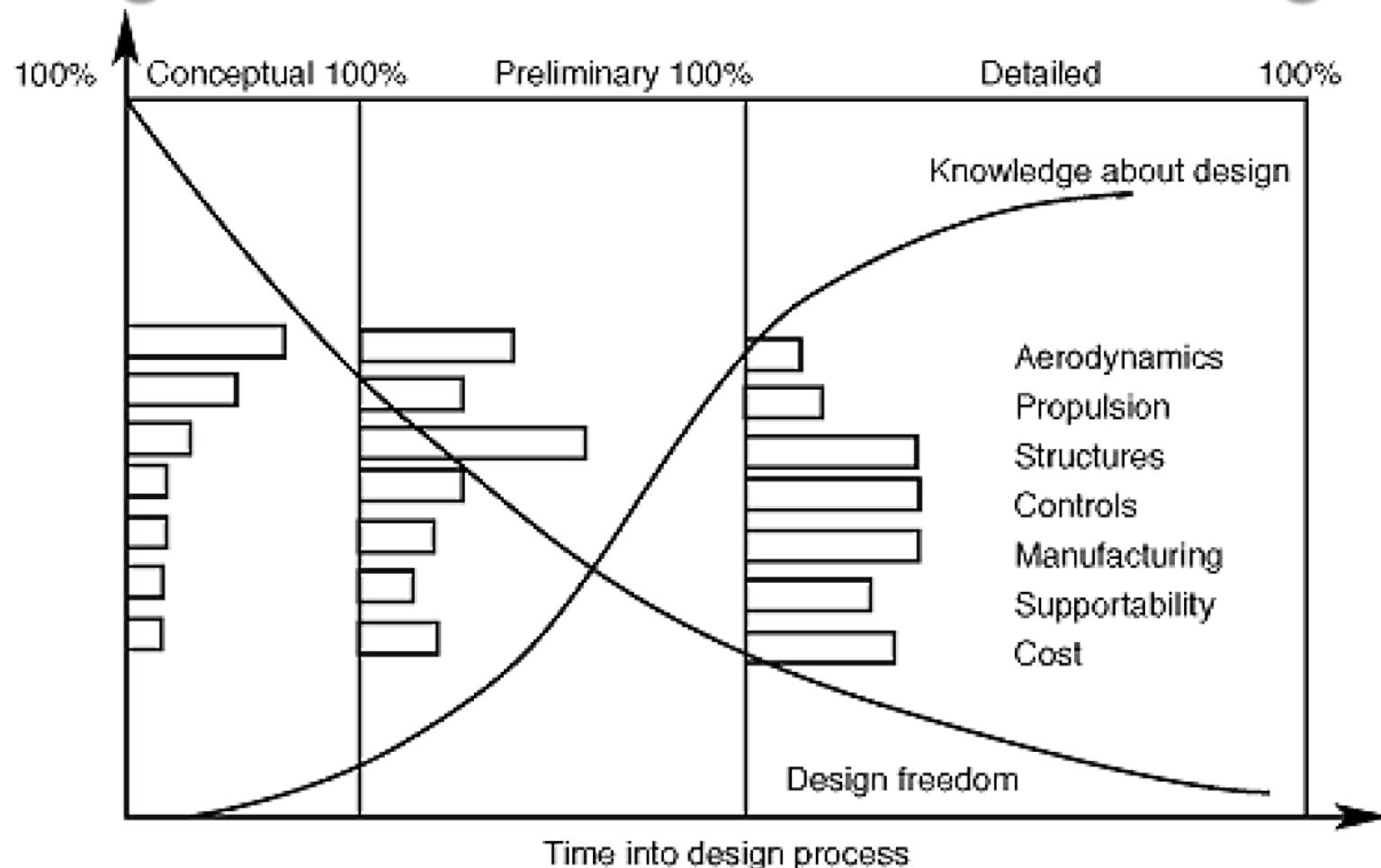
- Multidisciplinary design extended to other industries: spacecraft, automobiles, electronics and computers, transportation, energy, and architecture
- Thrusts in government and industry to **improve productivity and quality** in products and processes
- Design processes: Globalization results in distributed, **decentralized** design teams, high performance PC has replaced centralized super-computers, disciplinary design software (Nastran, CAD/CAM) very mature, Internet and LAN's allow easy information transfer
- Advances in optimization algorithms: e.g., Genetic Algorithms, Simulated Annealing, MDO software, e.g., iSIGHT, Model Center, OpenMDAO,...

MDO Timeline



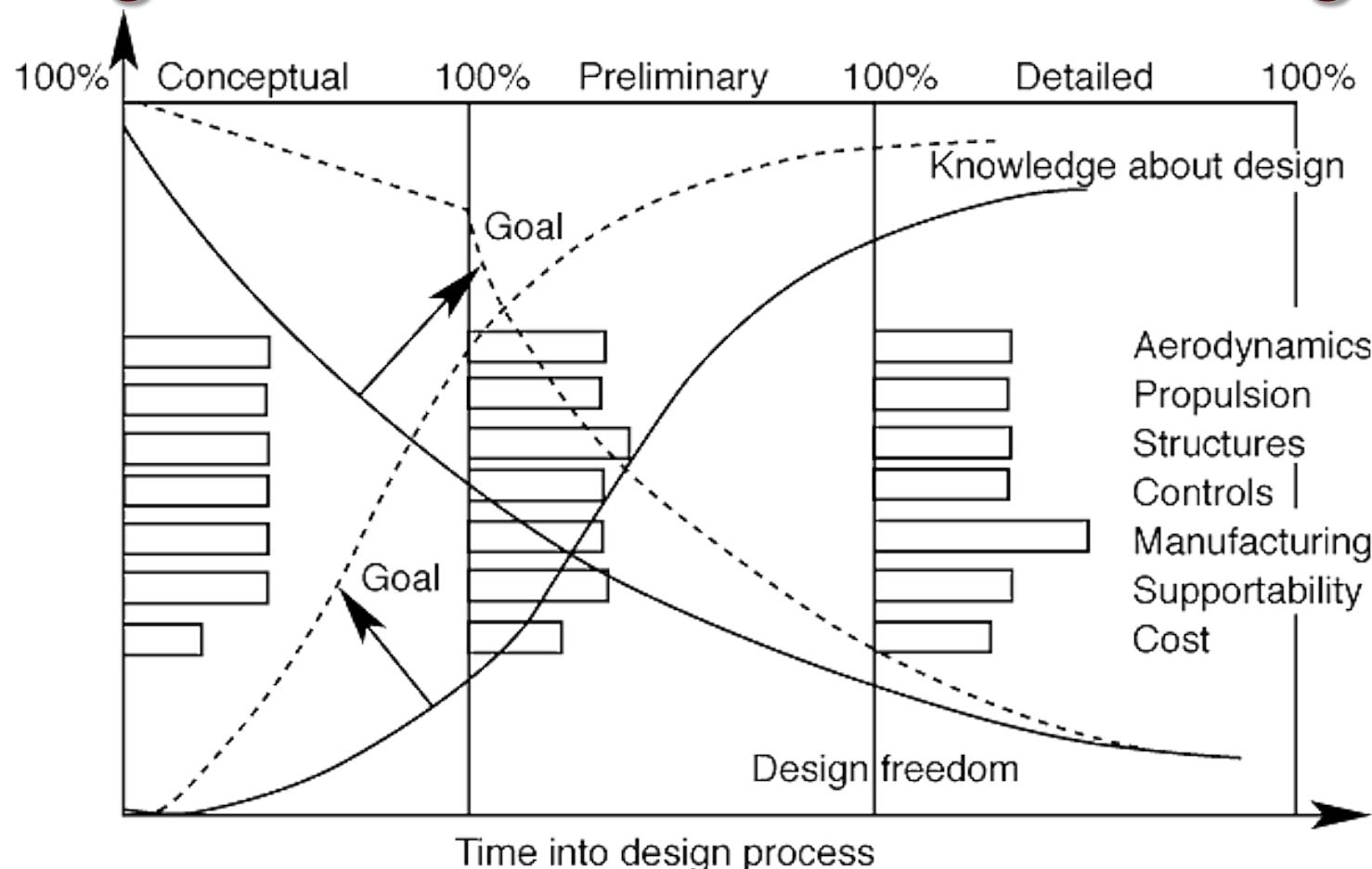
Agte J., de Weck O.,
Sobieszczański-Sobieski
J., Arendsen P., Morris A.,
Speick M., “MDO:
assessment and direction
for advancement - an
opinion of one international
group”, *Structural and
Multidisciplinary
Optimization*, 40 (1) 17-33,
January 2010

Design Freedom versus Knowledge



Goal of MDO: Gain design knowledge earlier and retain design freedom longer into the development process.

Design Freedom versus Knowledge



Goal of MDO: Gain design knowledge earlier and retain design freedom longer into the development process.

Definitions

Multidisciplinary – comprised of more than one traditional disciplinary area described by governing equations from various physical, economic, social fields

System – A system is a physical or virtual object that exhibits some behavior or performs some function as a consequence of interactions between the constituent elements

Design – The process of conceiving and planning an object or process with a specific goal in mind. In the context of this class this refers to the conceiving of a system that will subsequently be implemented and operated for some beneficial purpose.

Optimization – To find a system design that will minimize some objective function. The objective function can be a vector comprising measures of system behavior (“performance”), resource utilization (“time, money, fuel,...”) or risk (“stability margins...”).

Problem Formulation and Setup

(NLP)

$$\min \mathbf{J}(\mathbf{x}, \mathbf{p}) \longrightarrow \text{objective}$$

$$\begin{aligned} \text{s.t. } \mathbf{g}(\mathbf{x}, \mathbf{p}) \leq 0 \\ \mathbf{h}(\mathbf{x}, \mathbf{p}) = 0 \end{aligned} \longrightarrow \text{constraints}$$

$$x_{i,LB} \leq x_i \leq x_{i,UB} \longrightarrow \text{bounds}$$

$$\text{where } \mathbf{J} = \begin{bmatrix} J_1(\mathbf{x}) & \dots & J_z(\mathbf{x}) \end{bmatrix}^T$$

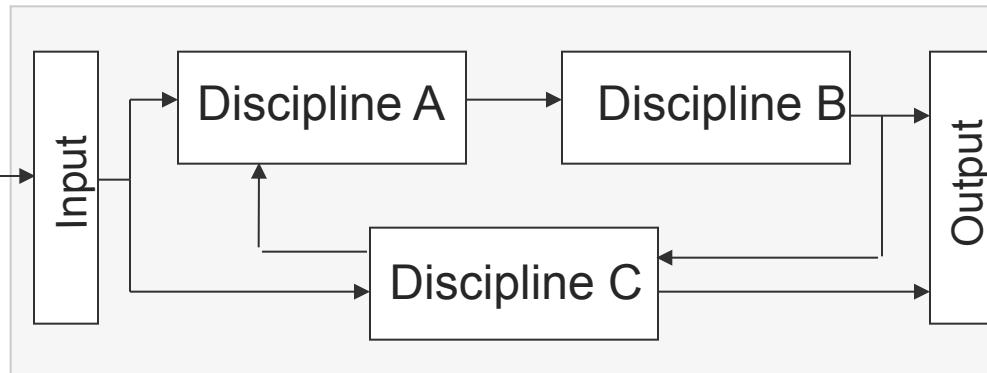
$$\mathbf{x} = \begin{bmatrix} x_1 & \dots & x_i & \dots & x_n \end{bmatrix}^T \longrightarrow \text{design vector}$$

MSADO Framework

Design Vector

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Simulation Model



Objective Vector

$$\begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_z \end{bmatrix}$$

Coupling

Multiobjective Optimization

Optimization Algorithms

Tradespace Exploration (DOE)

Numerical Techniques
(direct and penalty methods)

Heuristic Techniques
(SA, GA)

Approximation Methods

Sensitivity Analysis

Isoperformance

Coupling

Special Techniques

Challenges of MSADO

- Deal with design models of realistic size and fidelity that will not lead to erroneous conclusions
- Reduce the tedium of coupling variables and results from disciplinary models, such that engineers don't spend 50-80% of their time doing data transfer
- Allow for creativity while leveraging rigorous, quantitative tools in the design process. Hand-shaking: qualitative vs. quantitative
- Data visualization in multiple dimensions
- Incorporation of higher-level upstream and downstream system architecture aspects in early design: staged deployment, safety and security, environmental sustainability, platform design, etc...

Summary of what you will accomplish

- **Learning Objectives:**
 - Decompose and integrate multidisciplinary design models
 - Formulate meaningful problems mathematically
 - Explore design space and understand optimization
 - Critically analyze results, including sensitivity analysis
- **Understand current state of the art in MSADO**
 - See depth and breadth of applications in industry and science
 - Get a feel for interaction of quantitative-qualitative design
 - Understand limitations of techniques
 - Good overview of literature in the field
- **Benefit your research directly ... and have fun!**

Assignments

- Read Chapter 1
 - Papalambros, “Principles of Optimal Design”
 - On eCampus
- Read 1991 AIAA MDO TC Whitepaper
 - On eCampus
- Upload a 1 page CV/Resume to eCampus
- A1 will be handed out (on eCampus) on Monday

Lecture 02: Multidisciplinary System Analysis and Design Optimization (MSADO)

Problem Formulation

January 26, 2015

Prof. Douglas Allaire

Today's Topics

- MDO definition
- Optimization problem formulation
- MDO in the design process
- MDO challenges

MDO Definition

What is MDO?

- A methodology for the design of complex engineering systems and subsystems that coherently exploits the synergism of mutually interacting phenomena
- Optimal design of complex engineering systems which requires analysis that accounts for interactions amongst the disciplines (= parts of the system)
- “How to decide what to change, and to what extent to change it, when everything influences everything else.”

Ref: AIAA MDO website <http://www.aiaa.org>

Engineering Design Disciplines

Aircraft:

Aerodynamics

Propulsion

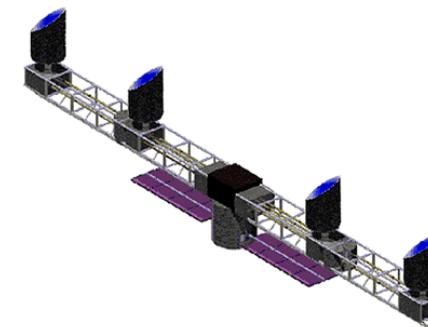
Structures

Controls

Avionics/Software

Manufacturing

others



Automobiles:

Engines

Body/chassis

Aerodynamics

Electronics

Hydraulics

Industrial design

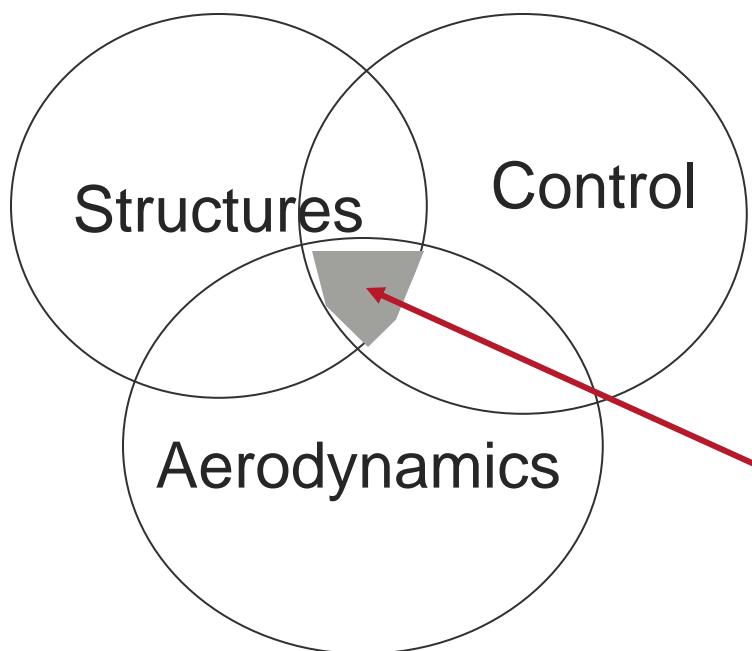
others



Fairly mature, but advances in theory, methodology, computation and application foster substantial payoffs

Multidisciplinary Aspects of Design

Emphasis is on the multidisciplinary nature of the complex engineering systems design process. Aerospace vehicles are a particular class of such systems.



Emphasis in recent years has been on advances that can be achieved due to the interaction of two or more disciplines.

System Level Optimization

Why system level, multidisciplinary optimization?

- Disciplinary specialists tend to strive towards improvement of objectives and satisfaction of constraints in terms of the variables of their own discipline
- In doing so they generate side effects – often unknowingly – that other disciplines have to absorb, usually to the detriment of the overall system performance

Aircraft “Optimization”

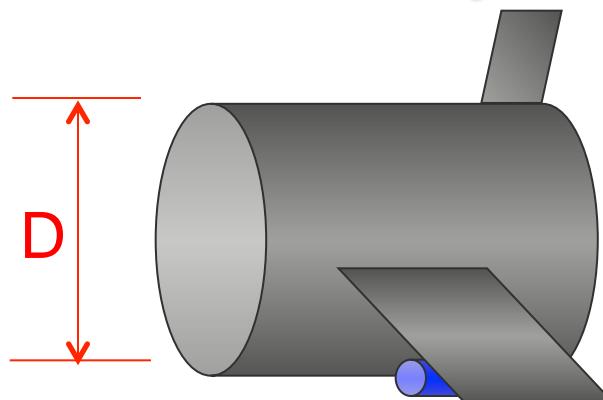
Marketing:

Aero:

Structures:

Propulsion:

Aircraft “Optimization”



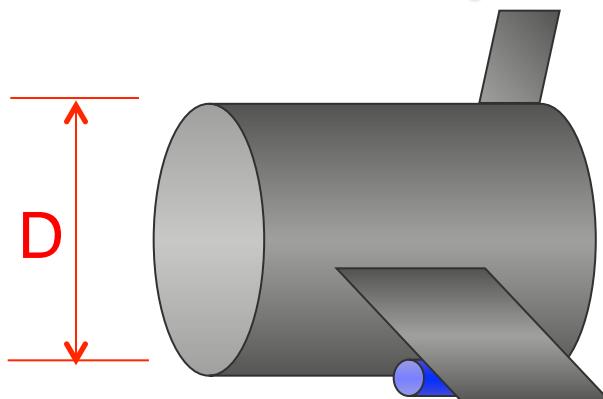
Marketing: maximize passenger volume
→ Cabin diameter

Aero:

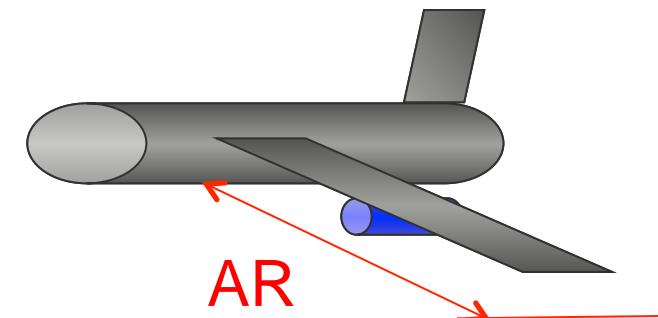
Structures:

Propulsion:

Aircraft “Optimization”



Marketing: maximize passenger volume
→ Cabin diameter

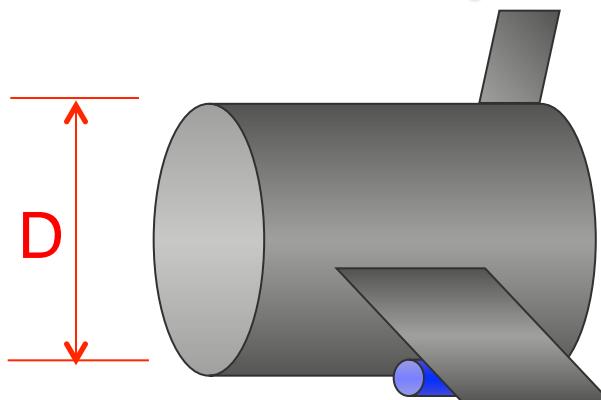


Aero: maximize L/D
→ Aspect Ratio

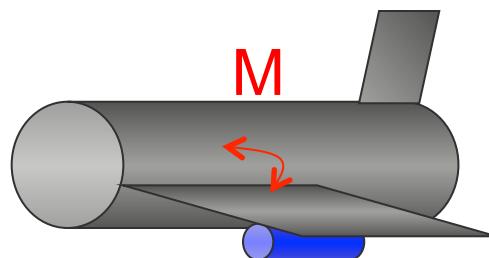
Structures:

Propulsion:

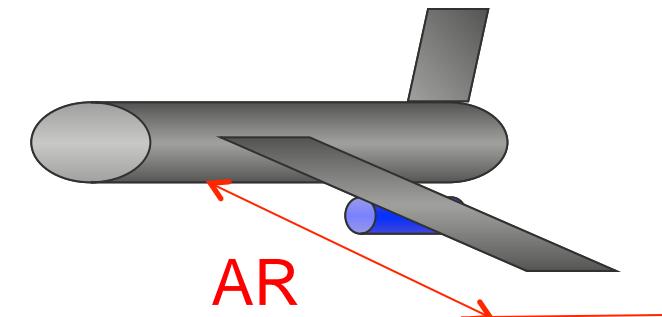
Aircraft “Optimization”



Marketing: maximize passenger volume
→ Cabin diameter



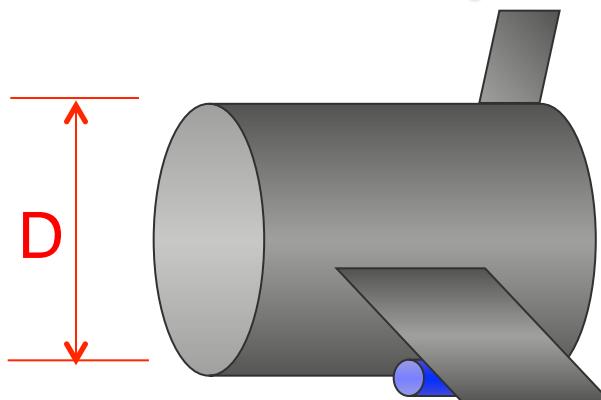
Structures: minimize structural mass
→ Wing-root moment



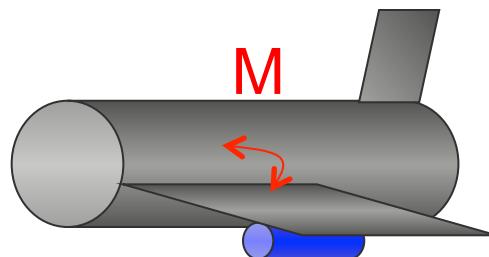
Aero: maximize L/D
→ Aspect Ratio

Propulsion:

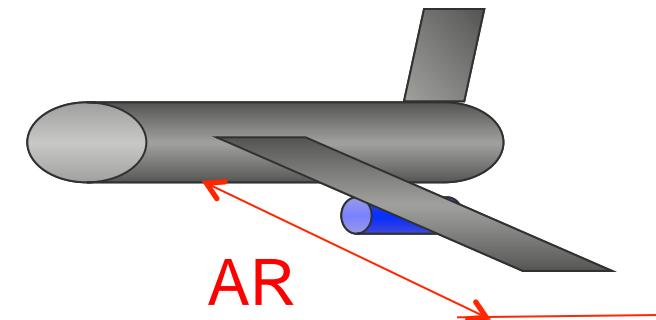
Aircraft “Optimization”



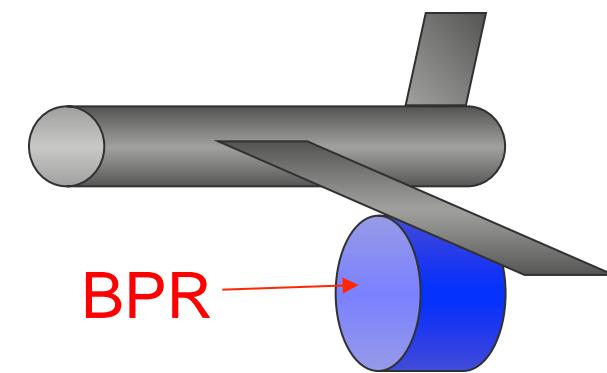
Marketing: maximize passenger volume
→ Cabin diameter



Structures: minimize structural mass
→ Wing-root moment



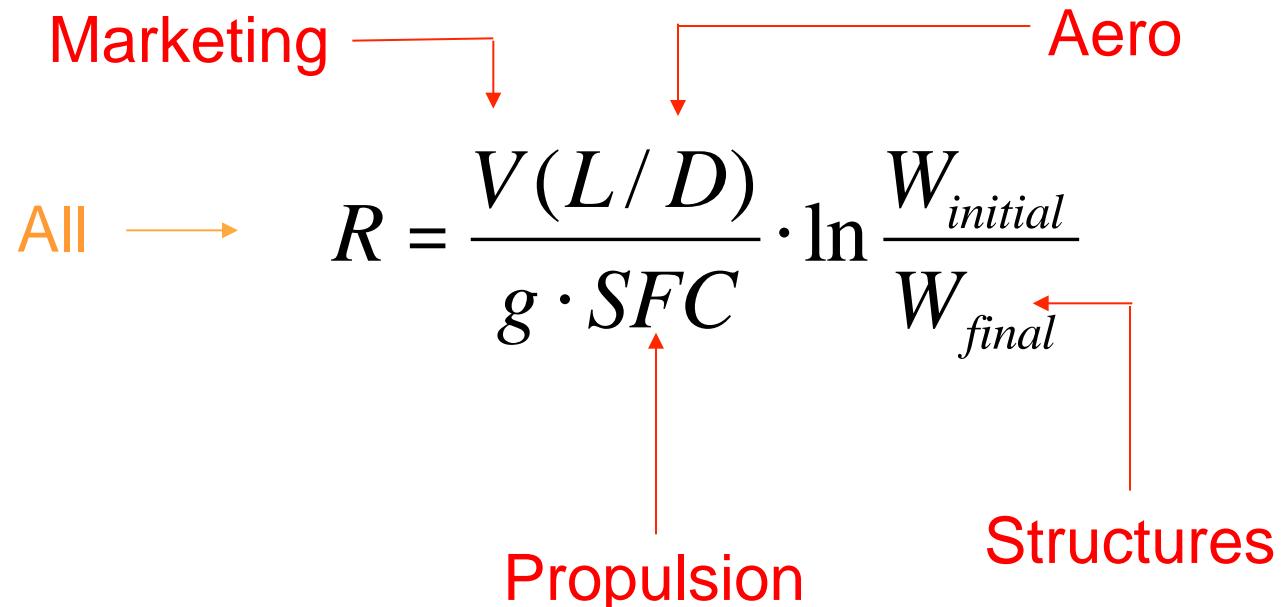
Aero: maximize L/D
→ Aspect Ratio



Propulsion: minimize specific fuel consumption (SFC) → Bypass Ratio

System Level Optimization

Bréguet
Range
Equation



R = Range [m]

V = Flight velocity [m/s]

SFC = Specific Fuel Consumption [kg/s/N]

L/D = Lift-over-Drag ratio [N/N]

g = gravitational acceleration [m/s²]

$W_{initial}$ = Initial (takeoff) weight [N]

W_{final} = Weight at end of flight [N]

$W_{fuel} = W_{initial} - W_{final}$ Fuel quantity [N]

Human Interface Aspects of Design

It is wrong to think of MDO as “automated” or “push-button” design:

- The human strengths (creativity, intuition, decision-making) and computer strengths (memory, speed, objectivity) should complement each other
- The human will always be the meta-designer
- Challenges of defining an effective interface – continuous vs. discrete thinking
- Challenges of visualization in multidimensional space, e.g., search path from initial design to final design



Human element is a key component in any successful system design methodology

Quantitative vs. Qualitative

Conceiving
different
concepts

Evaluation,
selection of
concepts

Human inventiveness, creativity, intuition, experience

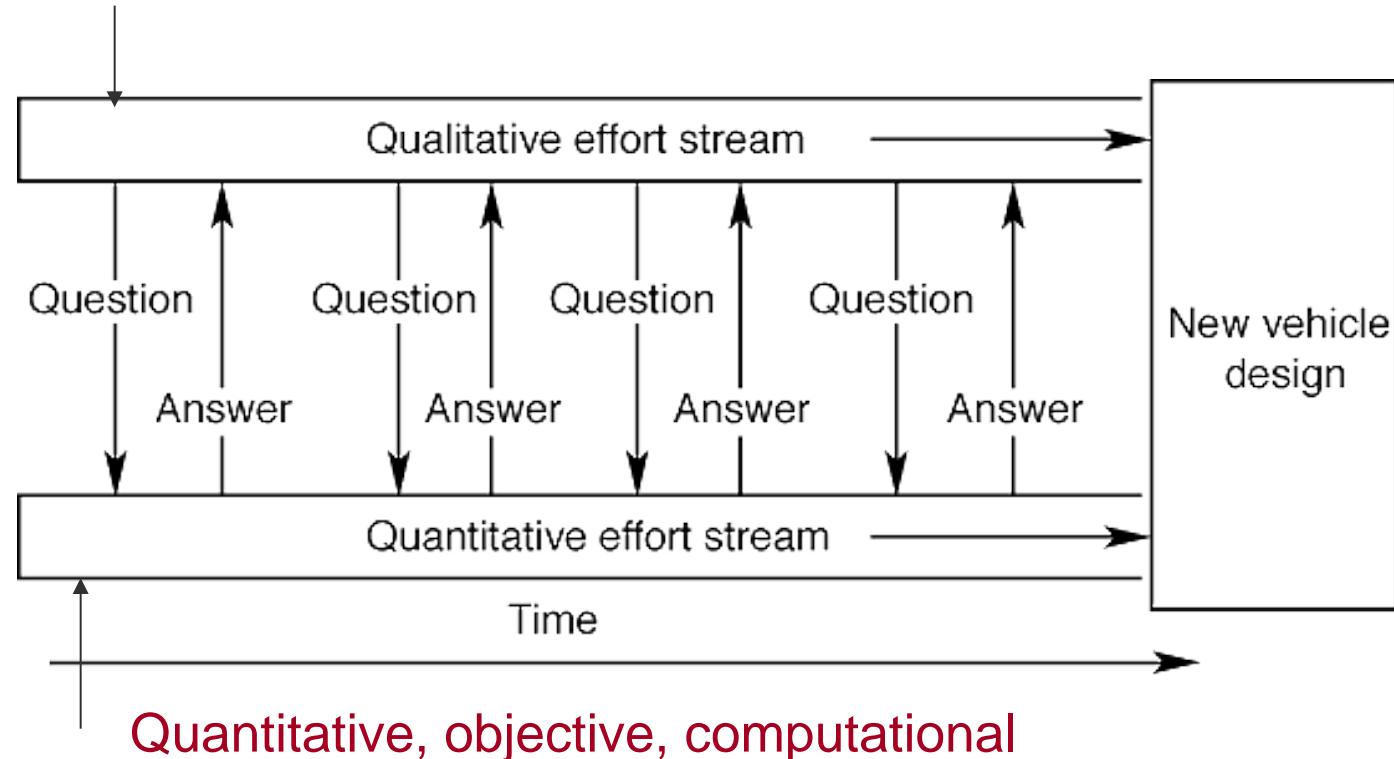
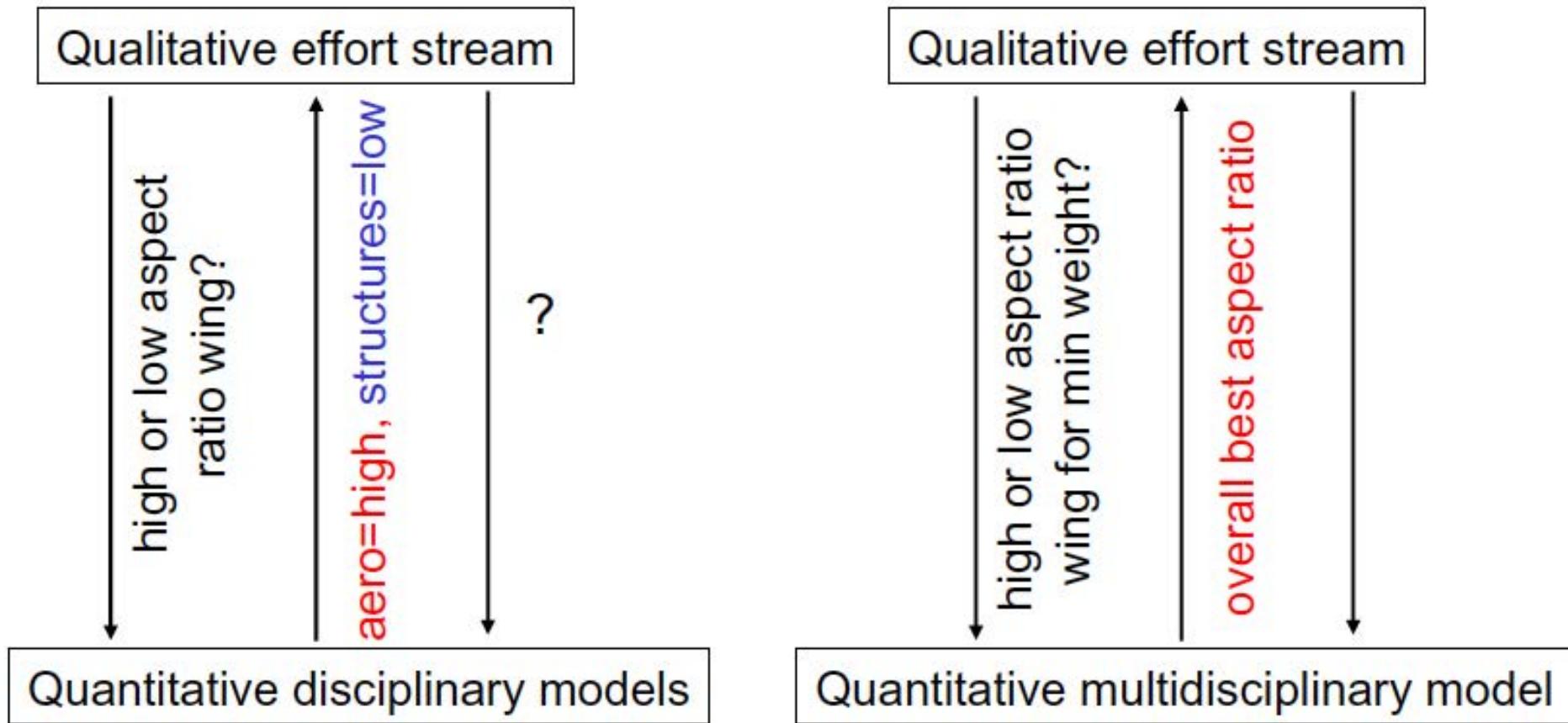


Fig.13 Parallel, qualitative, and quantitative efforts in design.

Human mind is the driving force in the design process. MDO is a way of formalizing the quantitative tool to apply the best trade-offs.

Quantitative vs. Qualitative

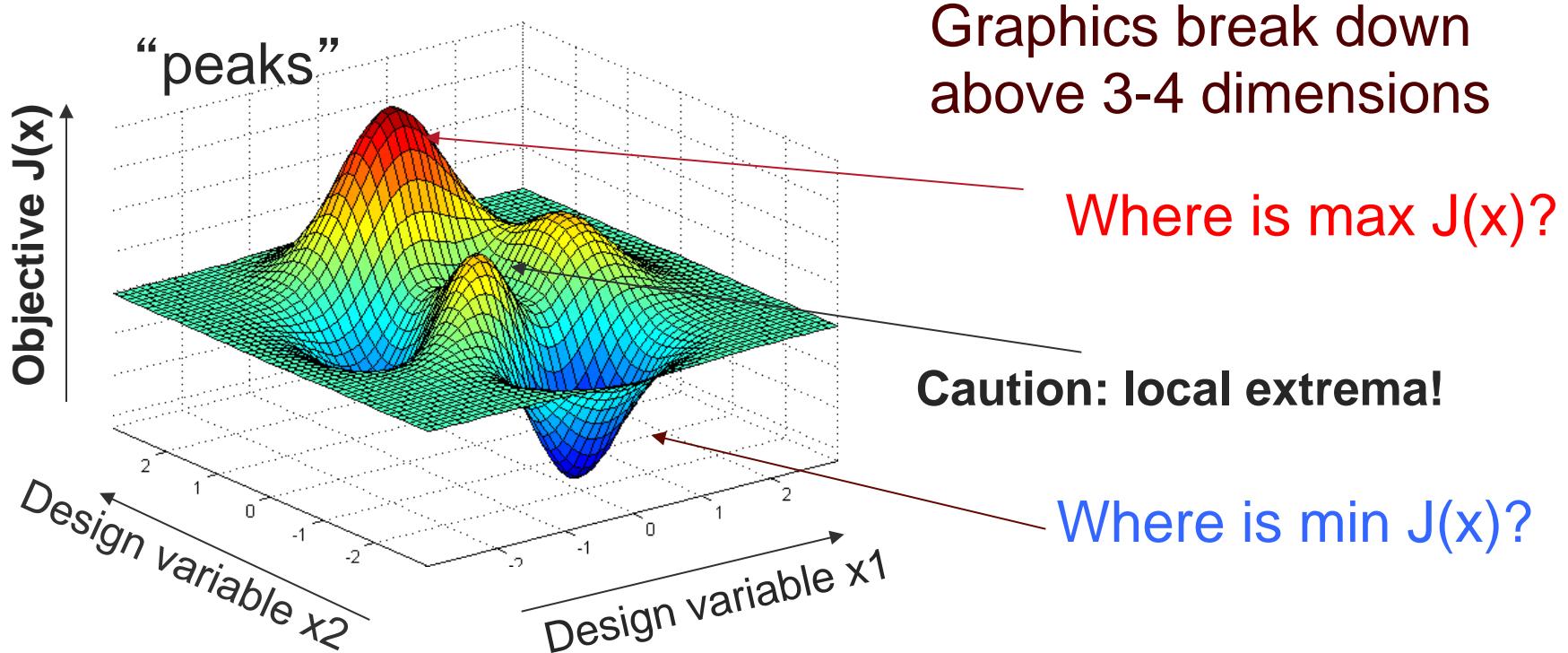


MDO is a way of formalizing the quantitative tool to apply the best trade-offs. The question provides a metric; the answer accounts for both disciplinary and interaction information.

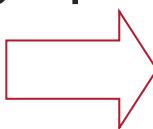
Optimization Problem Formulation

Optimization Aspects of Design

- Optimization methods have been combined with design synthesis and parametric analysis for ca. 40 years
- Traditionally used graphical methods to find maximum or minimum of a multivariate function (“carpet plot”), but...



Combinatorial Explosion

- Any design can be defined by a vector in multidimensional space, where each design variable represents a different dimension
 - For $n > 3$ a combinatorial “explosion” takes place (“curse of dimensionality”) and the design space cannot be computed and plotted (using say a grid) in polynomial time
 - Numerical optimization offers an alternative to the graphical approach and “brute force” evaluation
-  During past three decades much progress has been made in numerical optimization

Formal Notation

Quantitative side of the design problem may be formulated as a problem of Nonlinear Programming (NLP)

$$\min \mathbf{J}(\mathbf{x}, \mathbf{p})$$

$$\text{s.t. } \mathbf{g}(\mathbf{x}, \mathbf{p}) \leq 0$$

$$\mathbf{h}(\mathbf{x}, \mathbf{p}) = 0$$

$$x_{i,LB} \leq x_i \leq x_{i,UB} \quad (i = 1, \dots, n)$$

This is the problem formulation that we will discuss this semester.

$$\text{where } \mathbf{J} = \begin{bmatrix} J_1(\mathbf{x}, \mathbf{p}) & \cdots & J_z(\mathbf{x}, \mathbf{p}) \end{bmatrix}^T$$

$$\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_i & \cdots & x_n \end{bmatrix}^T$$

$$\mathbf{g} = \begin{bmatrix} g_1(\mathbf{x}, \mathbf{p}) \cdots g_{m_1}(\mathbf{x}, \mathbf{p}) \end{bmatrix}^T$$

$$\mathbf{h} = \begin{bmatrix} h_1(\mathbf{x}, \mathbf{p}) \cdots h_{m_2}(\mathbf{x}, \mathbf{p}) \end{bmatrix}^T$$

Objectives

The objective can be a vector \mathbf{J} of z system responses or characteristics we are trying to maximize or minimize

$$\mathbf{J} = \begin{bmatrix} J_1 \\ J_2 \\ J_3 \\ J_4 \\ \vdots \\ J_z \end{bmatrix} = \begin{bmatrix} \text{cost } [\text{\$}] \\ \text{range } [\text{km}] \\ \text{weight } [\text{kg}] \\ \text{data rate } [\text{bps}] \\ \vdots \\ \text{ROI } [\%] \end{bmatrix}$$

Often the objective is a scalar function, but for real systems often we attempt multi-objective optimization:

$$\mathbf{x} \mapsto \mathbf{J}(\mathbf{x})$$

Some objectives can be conflicting.

Design Variables

Design vector \mathbf{x} contains n variables that form the design space

During design space exploration or optimization we change the entries of \mathbf{x} in some rational fashion to achieve a desired effect

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} \text{aspect ratio [-]} \\ \text{transmit power [W]} \\ \# \text{ of apertures [-]} \\ \text{orbital altitude [km]} \\ \vdots \\ \text{control gain [V/V]} \end{bmatrix} \quad x_i \text{ can be ...}$$

Real: $x_i \in \mathcal{R}$

Integer: $x_i \in \mathcal{I}$

Binary: $x_i \in \{0, 1\}$

Boolean: $x_i \in \{\text{true}, \text{false}\}$



Design variables are “controlled” by the designers

Parameters

Parameters p are quantities that affect the objective J , but are considered fixed, i.e., they cannot be changed by the designers.

Sometimes parameters p can be turned into design variables x_i to enlarge the design space.

Sometimes parameters p are former design variables that were fixed at some value because they were found not to affect any of the objectives J , or because their optimal level was predetermined.

Constraints

Constraints act as boundaries of the design space \mathbf{x} and typically occur due to finiteness of resources or technological limitations of some design variables.

Often, but not always, optimal designs lie at the intersection of several active constraints.

Inequality constraints:

$$g_j(\mathbf{x}) \leq 0 \quad j = 1, 2, \dots, m_1$$

Equality constraints:

$$h_k(\mathbf{x}) = 0 \quad k = 1, 2, \dots, m_2$$

Bounds:

$$x_{i,LB} \leq x_i \leq x_{i,UB} \quad i = 1, 2, \dots, n$$



Objectives are what we are trying to achieve
Constraints are what we cannot violate
Design variables are what we can change

Constraints versus Objectives

It can be difficult to choose whether a condition is a constraint or an objective.

For example: should we try to minimize cost, or should we set a constraint stating that cost should not exceed a given level?

The two approaches can lead to different designs.

Sometimes, the initial formulation will need to be revised in order to fully understand the design space.

In some formulations, all constraints are treated as objectives (physical programming)

Example Problem Statement

design variables

objective function

Minimize the **take-off weight of the aircraft** by changing **wing geometric parameters** while satisfying the given **range and payload requirements** at the given **cruise speed**.

constraints

parameter

Group Exercise... (10 mins)

Pair up and think about an engineering system of your choice

1. Consider the preliminary design phase.

Identify:

- Important disciplines
- Potential objective functions
- Potential design variables
- System parameters
- Constraints and bounds

2. Report out

MDO in the Design Process

What MDO really does

MDO mathematically traces a path in the design space from some initial design \mathbf{x}_0 towards improved designs (with respect to the objective J).

It does this by operating on a large number of variables and functions simultaneously – a feat beyond the power of the human mind.

The path is not biased by intuition or experience.

This path instead of being invisible inside a “black box” becomes more visible by various MDO techniques such as sensitivity analysis and visualization



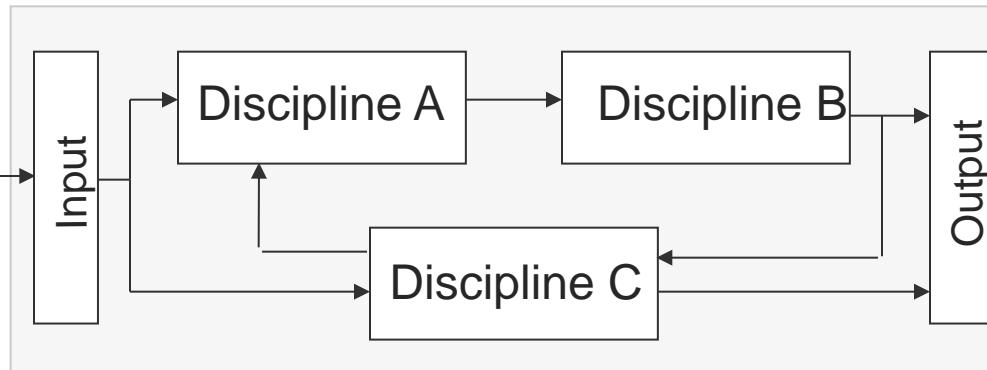
Optimization does not remove the designer from the loop, but it helps conduct trade studies

Recall - MSADO Framework

Design Vector

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$

Simulation Model



Objective Vector

$$\begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_z \end{bmatrix}$$

Coupling

Multiobjective Optimization

Optimization Algorithms

Tradespace Exploration (DOE)

Numerical Techniques
(direct and penalty methods)

Heuristic Techniques
(SA,GA)

Approximation Methods

Sensitivity Analysis

Isoperformance

Coupling

Output Evaluation

Simulation versus Optimization

There are two distinct components of the MSADO process:
The **optimization algorithm** decides how to move through
the design space.

The **simulation model** evaluates (analyses) designs
chosen by the optimizer. Both objective functions and
constraints must be evaluated.

Sometimes, disciplinary simulation models can be used in
an optimization framework, but often they are not
appropriate.

There are several different approaches to couple the
optimizer and the simulation models (Lecture 6).

Typical Process in MDO

- (1) Define overall system requirements
- (2) Define design vector \mathbf{x} , objective \mathbf{J} and constraints
- (3) System decomposition into modules
- (4) Modeling of physics via governing equations at the module level - module execution in isolation
- (5) Model integration into an overall system simulation
- (6) Benchmarking of model with respect to a known system from past experience, if available
- (7) Design space exploration (DoE) to find sensitive and important design variables x_i
- (8) Formal optimization to find $\min \mathbf{J}(\mathbf{x})$
- (9) Post-optimality analysis to explore sensitivity and tradeoffs: sensitivity analysis, approximation methods, isoperformance, include uncertainty

In Practice...

- (i) Step through (1)-(8)
- (ii) The optimizer will use an error in the problem setup to determine a mathematically valid but physically unreasonable solution

OR

The optimizer will be unable to find a feasible solution (satisfies all constraints)

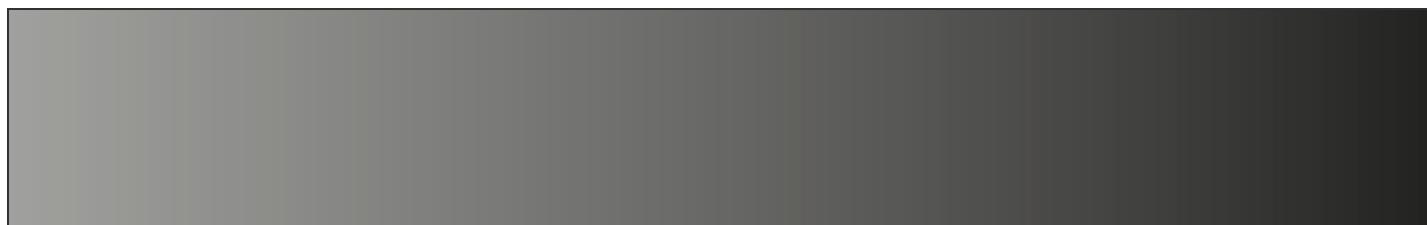
- (iii) Add, remove or modify constraints and/or design variables
- (iv) Iterate until an appropriate model is obtained

Although MDO is an automated formalization of the design process, it is a highly interactive procedure...

MDO Uses

- The ‘MD’ portion of ‘MDO’ is important on its own
- Often MDO is used not to find the truly optimal design, but rather to find an improved design, or even a feasible design...

Range of design objectives



Feasible

Improved

Optimal

Pareto

from Giesing, 1998

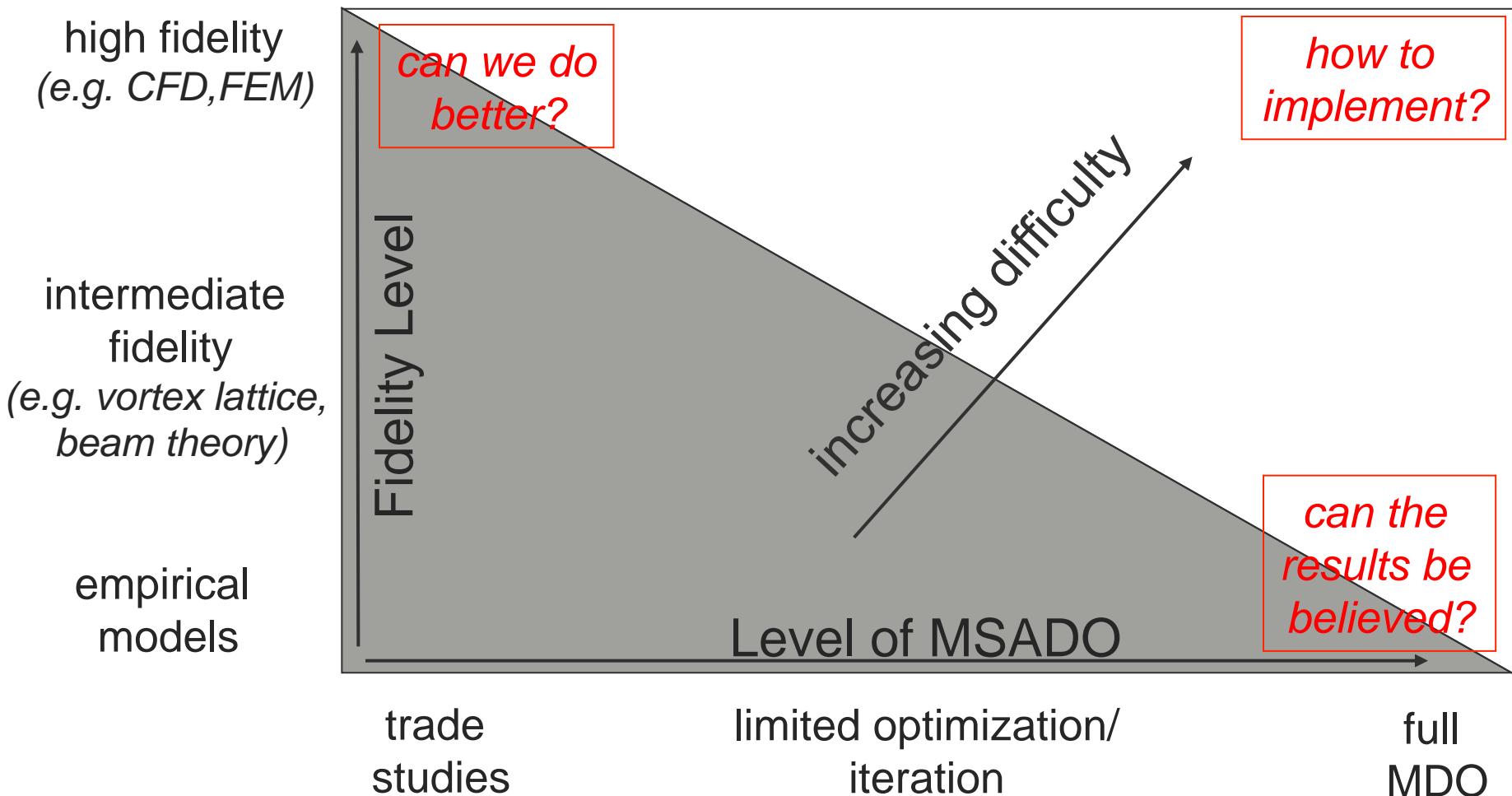
MDO Challenges

MDO Challenges

- Fidelity/expense of disciplinary models: Fidelity is often sacrificed to obtain models with short computation times.
- Complexity: Design variables, constraints, and model interfaces must be managed carefully.
- Communication: The user interface is often very unfriendly and it can be difficult to change problem parameters.
- Flexibility: It is easy for an MDO tool to become very specialized and only valid for one particular problem.

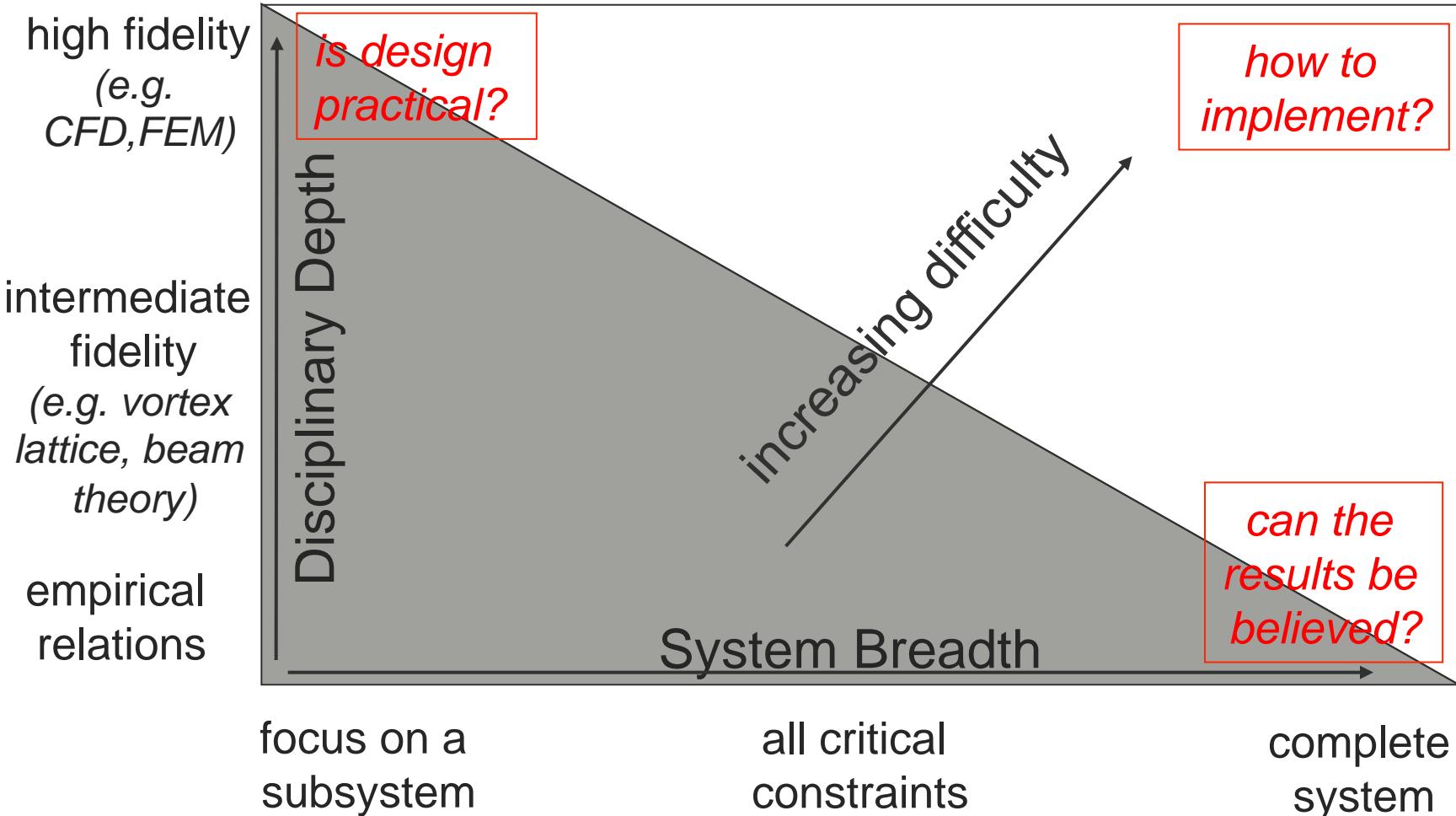
How do we prevent MDO codes from becoming complex, highly specialized tools which are used by a single person (often the developer!) for a single problem?

Fidelity vs. Expense



from Giesing, 1998

Breadth vs. Depth



MDO Pros/Cons

Advantages

- Reduction in design time
- Systematic, logical design procedure
- Handles wide variety of design variables and constraints
- Not biased by intuition or experience

Disadvantages

- Computational time grows rapidly with number of dv's
- Numerical problems increase with number of dv's
- Limited to range of applicability of analysis programs
- Will take advantage of analysis errors to provide mathematical design improvements
- Difficult to deal with discontinuous functions

Lecture Summary

- MDO is not a stand-alone, automated design process
- MDO is a valuable tool that requires substantial human interaction and complements other design tools
- Elements of an MDO framework
- MDO challenges



Next lecture will address Modeling & Simulation

Readings

- Giesing and Barthelemy 1998
- Agte et al. 2010
- Both are posted on eCampus

References

- Kroo, I.: “MDO applications in preliminary design: status and directions,” AIAA Paper 97-1408, 1997.
- Kroo, I. and Manning, V.: “Collaborative optimization: status and directions,” AIAA Paper 2000-4721, 2000.
- Sobieski, J. and Kroo, I.: “Aircraft design using collaborative optimization,” AIAA Paper 96-0715, 1996.
- Balling, R. and Wilkinson, C.: “Execution of multidisciplinary design optimization approaches on common test problems,” AIAA Paper 96-4033, 1996.
- Giesing, J. and Barthelemy, J.: “A summary of industry MDO applications and needs”, AIAA White Paper, 1998.
- AIAA MDO Technical Committee: “Current state-of-the-art in multidisciplinary design optimization”, 1991.
- Agte, J., de Weck, O., Sobieszczanski-Sobieski, J., Arendsen, P., Morris, A., and Speck, M.: “MDO: assessment and direction for advancement—an opinion of one international group, Struct Multidisc Optim (2010) 40:17-33.

Lecture 03: Multidisciplinary System Analysis and Design Optimization (MSADO)

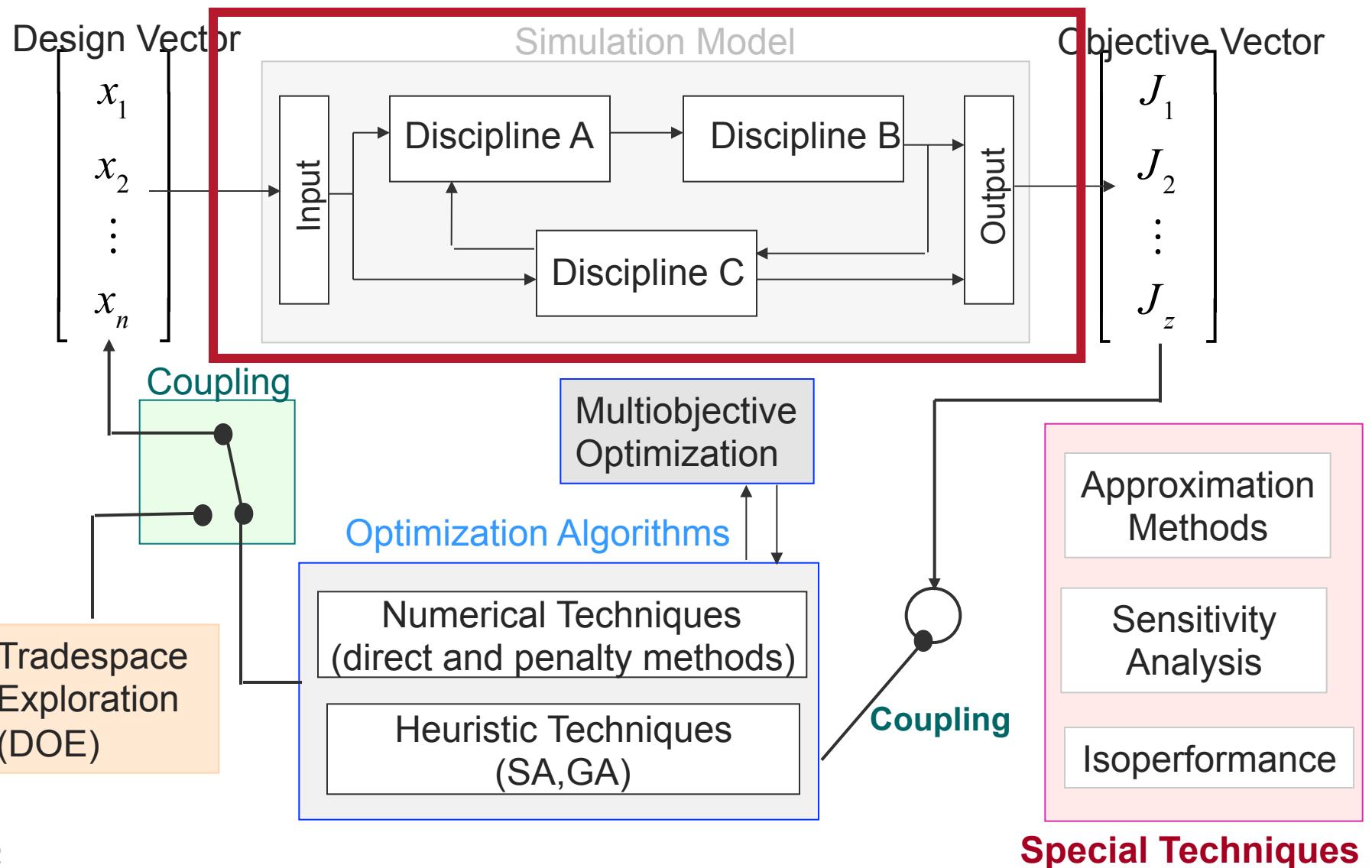
Modeling and Simulation

January 28, 2015

Prof. Douglas Allaire

MSDO Framework

today



Today's Topics

Definitions of Modeling and Simulation

- Physics-based modeling
- Empirical modeling

Model/Simulation Development Process

- Module identification
- Module ordering: DSM's and N² diagrams
- Module coding: fidelity and benchmarking
- Module execution = simulation

Computational Issues

- Coupling disparate CAE/CAD tools

Definitions

Definitions

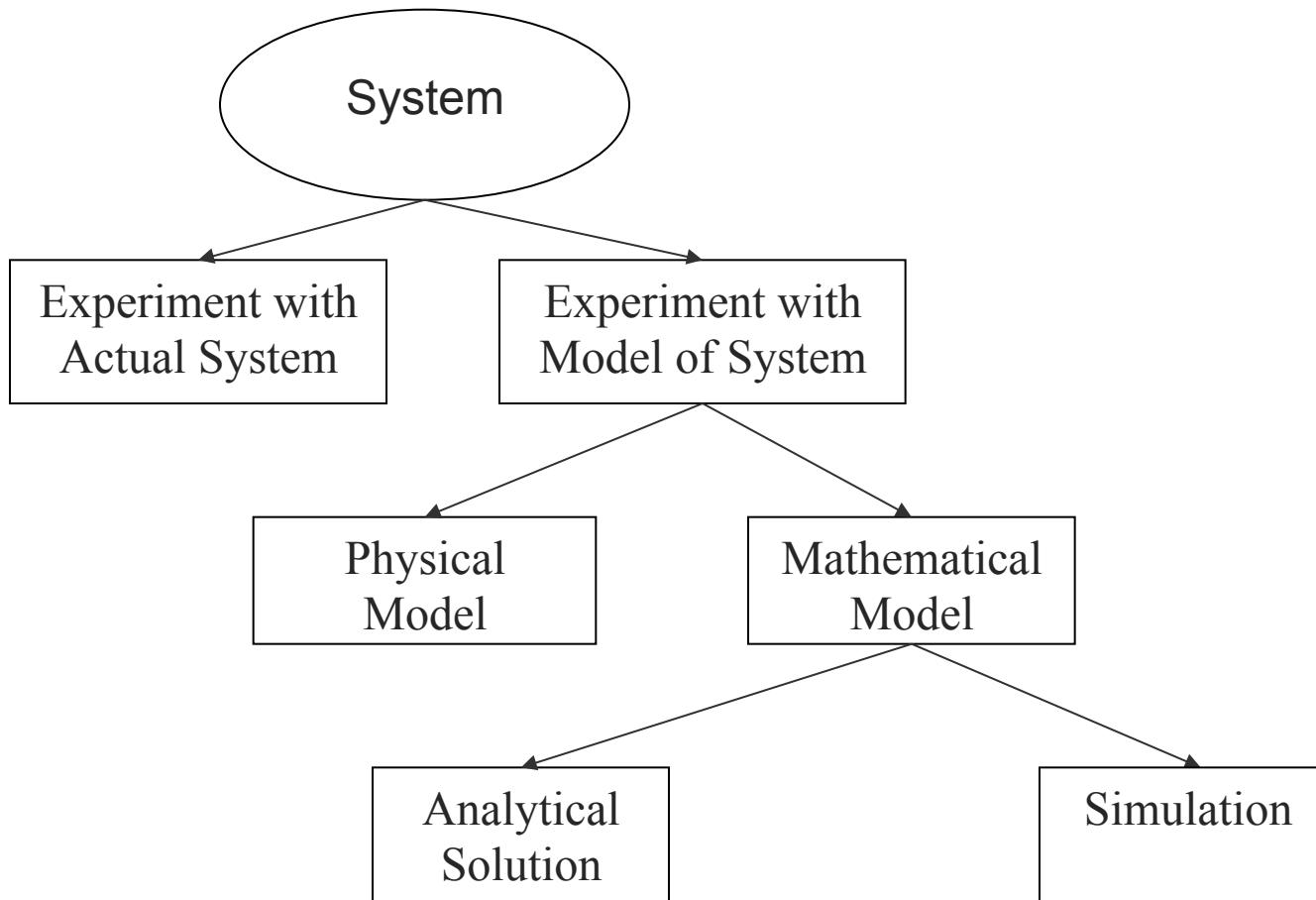
Definition: ***Model*** (*as used in this class*)

A model is a mathematical object that has the ability to predict the behavior of a real system under a set of defined operating conditions and simplifying assumptions.

Definition: ***Simulation*** (*as used in this class*)

Simulation is the process of exercising a model for a particular instantiation of the system and specific set of inputs in order to predict the system response.

From the Reference



Law & Kelton (2000), Simulation Modeling and Analysis 3rd ed., McGraw-Hill, Inc.

Additional Detail – Next Chart

The following chart includes additional detail to emphasize the factors that differentiate a **model** and a **simulation**

Simulation/Model Factors:

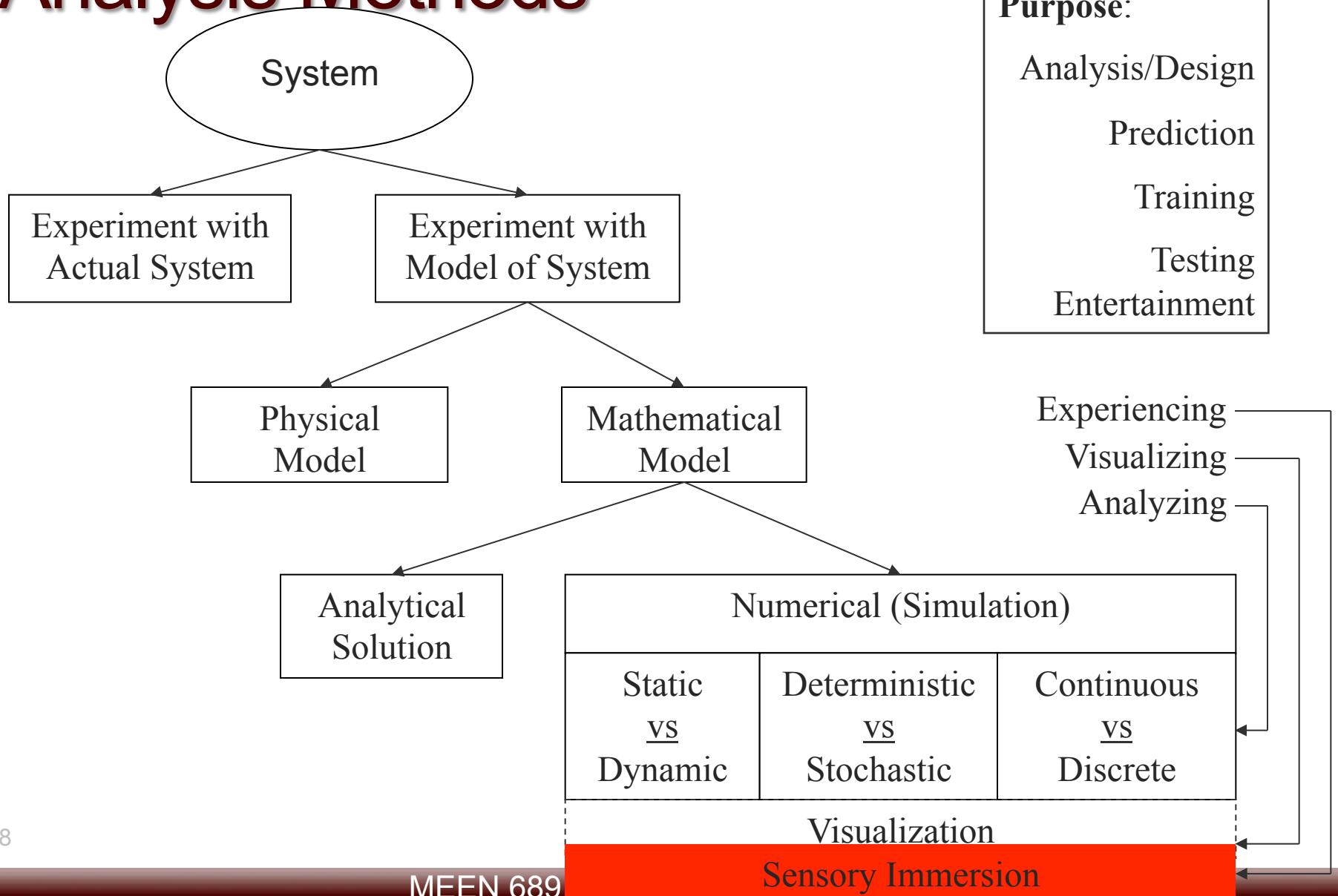
- Real World Variability
- Reaction to Events

These relate back to the **purpose** of the sim/
model

Models should not include all the details for all purposes

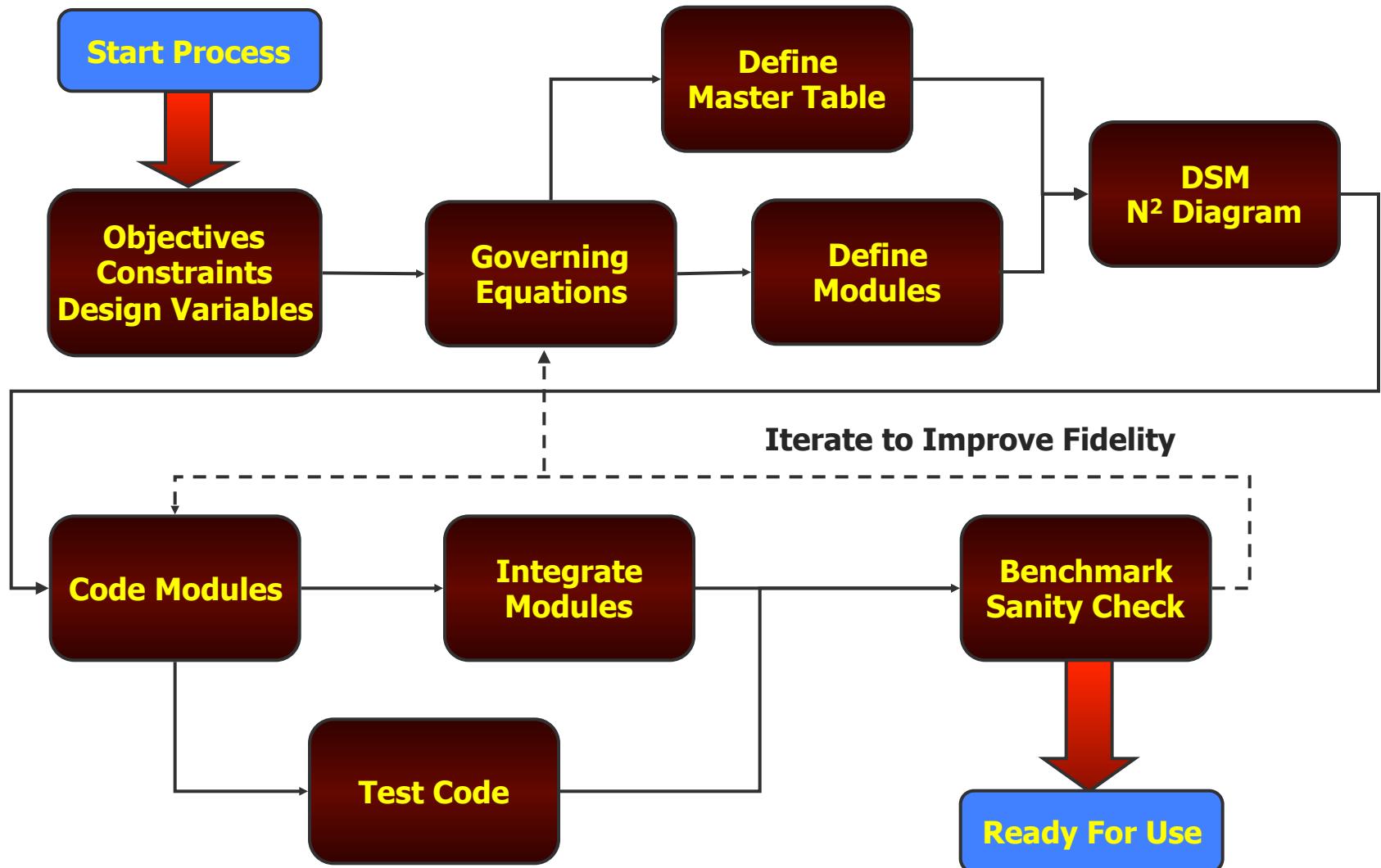
- They quickly become unwieldy & expensive

Analysis Methods



Modeling and Simulation Development Process

Model Development Process



Objectives, Constraints, Design Variables

- Define Objectives J This is how we want system to behave
- Define Design Variables x Things about system we can change
- Define Constraints and Bounds g, h Must satisfy this
- Determine important fixed parameters p Fixed, outside our control yet important

Influence Matrix

	x_1	...	x_n
J_i	+	+	o
g_j	+	o	+

model relationships

+ influence
o no influence

Physics Based Modeling

- Start with governing equations
- Continuum Mechanics for physical systems
- Introduce Boundary Conditions
- Introduce Initial Conditions
- External forcing functions
- Discretize system

Governing Equations

Continuum (Structural) Mechanics

$$S = \begin{bmatrix} \sigma_x & \tau_{xy} & \tau_{xz} \\ \tau_{yx} & \sigma_y & \tau_{yz} \\ \tau_{zx} & \tau_{zy} & \sigma_z \end{bmatrix} \text{ stress tensor}$$

A 3D rectangular element is shown with three forces applied to its top surface: F_1 to the left, F_2 downwards, and F_3 to the right. A point on the right face of the element is labeled with a displacement vector \vec{u} . A line segment from this point to another point on the left face represents the strain vector $\vec{\epsilon}_x$.

$$\epsilon_x = \frac{\partial u}{\partial x} \quad \text{strain}$$

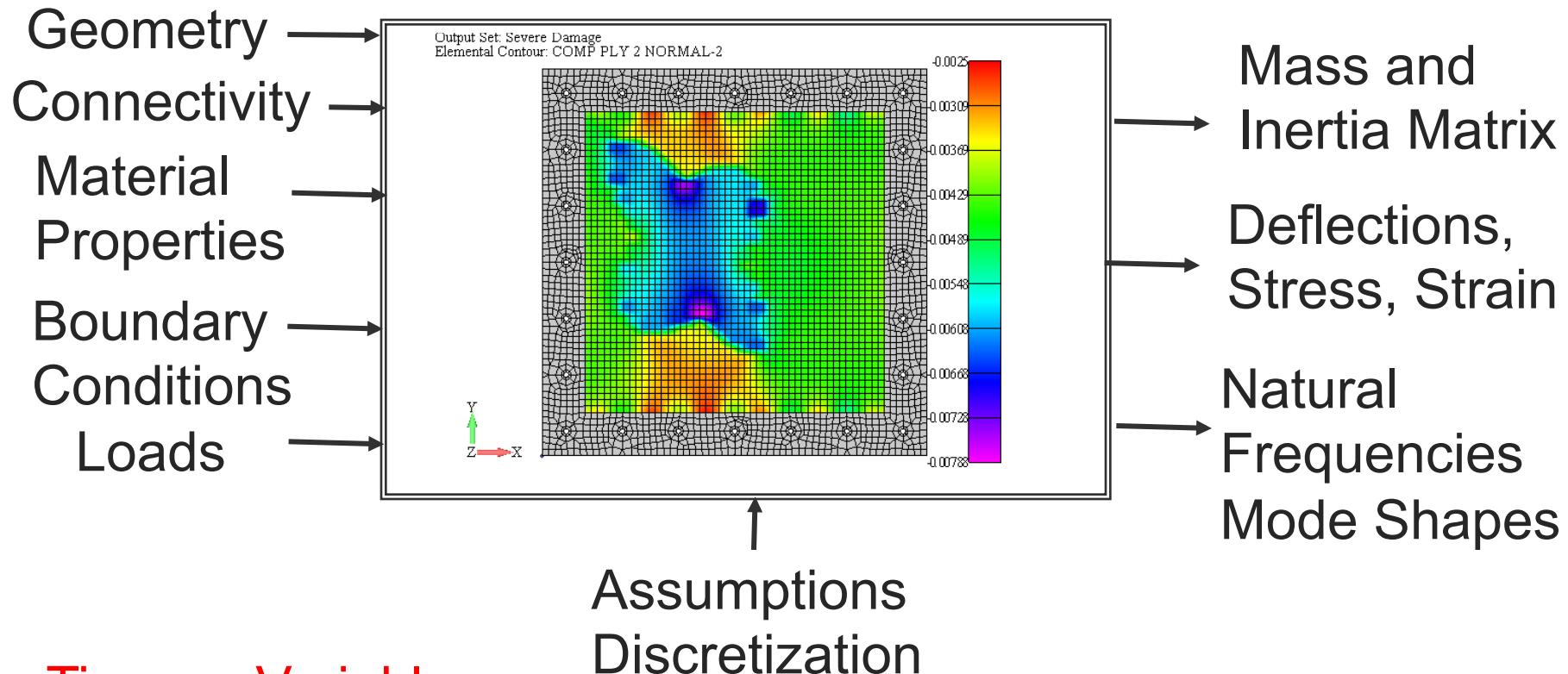
-Equilibrium Equations $\Sigma F_i = 0$

-Constitutive equations $\sigma_x = E \epsilon_x$

-Compatibility equations $\epsilon_x = \frac{dx' - dx}{dx}$

Example: Finite Element Model

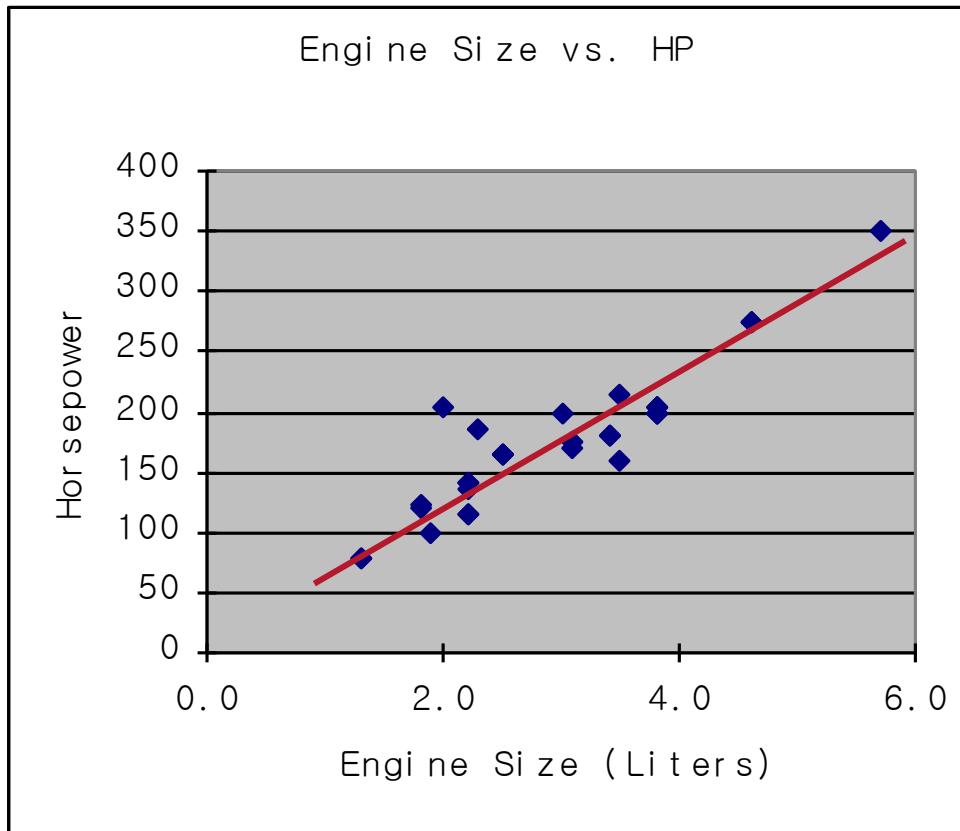
$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{F}$$



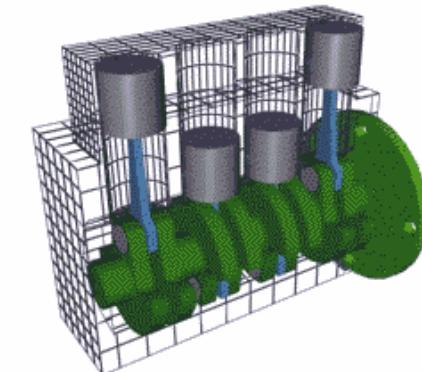
Empirical Modeling

- Derive a model, not from physics and first principles, **but from observation, i.e., data**
- Usually leads to low order models
- Only valid under similar operating conditions
- Many cost models are of this nature

Example: Empirical Modeling



$$\text{HP} = 51.48 \cdot \text{ED} + 23.12$$



... could do physics-Based modeling of this in-line 4 engine, but instead do ...

Linear Regression

$$HP = \alpha \cdot ED + \beta$$

How to decompose a system

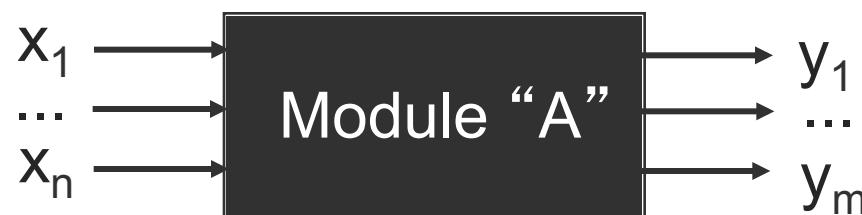
- What to do when a system is new – no experience?
- First define “black boxes” or modules based on:
disciplinary tradition, degree of coupling of governing
equations or availability of analysis software
- Crisply define inputs and outputs of each module

Ref: Rogers, J.L.: “A Knowledge-Based Tool for Multilevel Decomposition of a Complex Design Problem”, NASA TP2903, 1989

Module Definition

What is a module in MSADO?

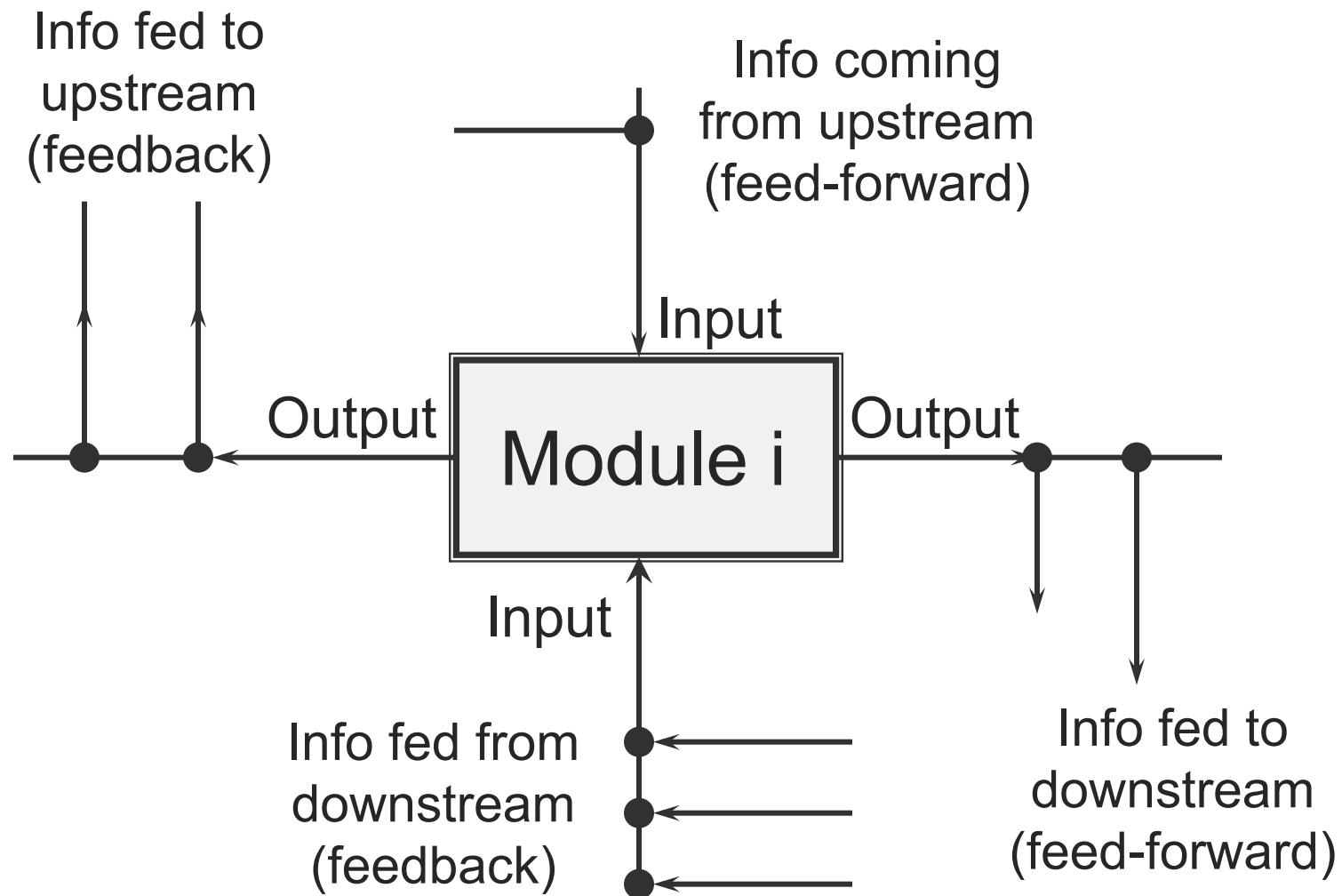
A module in multidisciplinary system analysis and design optimization is a finite group of tightly coupled mathematical relationships who are under the responsibility of a particular individual or organization, and where some variables represent independent inputs while others are dependent outputs. The module frequently appears as a “black box” to other individuals or organizations.



Modules for Simulation

- A **module** within a simulation architecture may be defined as a piece of computer code which:
 - Performs a **compact** set of calculations.
 - Contains a **single** entry point and exit point.
 - May be tested in **isolation**.
- Attributes of a **good modular unit** within a simulation architecture include:
 - **High internal coupling within the module**
 - All sub-functions within the module contribute to form a **single primary function**.
 - **Low coupling between modules**
 - **Minimize** the number of variables that flow **between** modules.
 - **Minimization of feedback loops**
 - Data flow is processed **sequentially** from input to output.

Module Inputs and Outputs

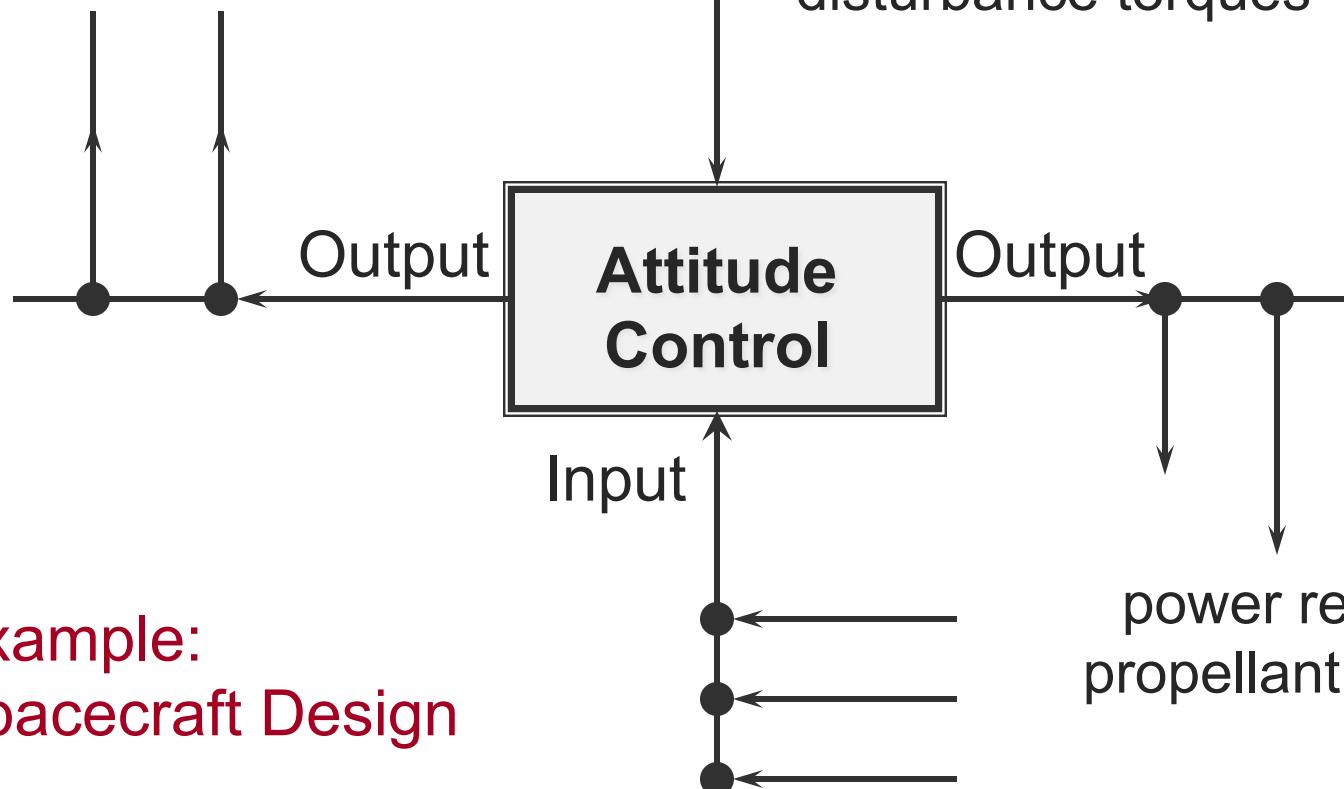


Module Example

sensors

actuators

(e.g. CMGs)

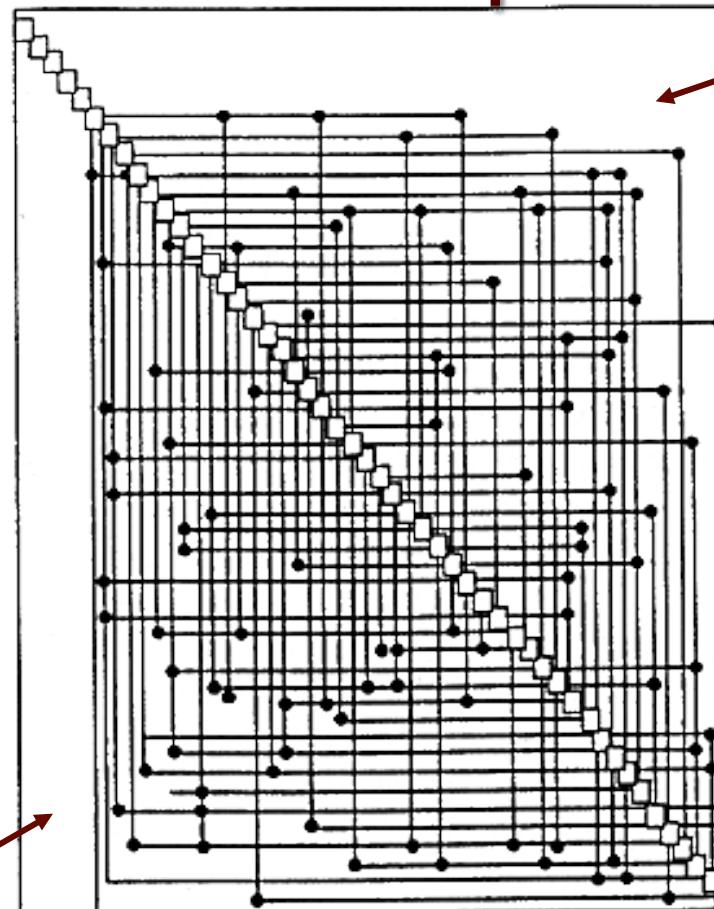


Example:
Spacecraft Design

The N² Diagram

- An NxN matrix used to develop and **organize interface information**.
- Similar to a Design Structure Matrix (DSM)
- Each module within the simulation architecture is placed along the **diagonal**.
- Provides a visual representation of the **flow of information** through the simulation architecture.
- Helps to **identify critical modules** that have many inputs and outputs. The fidelity of critical modules should be thoroughly tested and verified.
- Explicitly defines all **inputs** and **outputs** for macro-modules and modules.
- Allows for “plug and play”
 - Independent testing
 - Alternative modules easily analyzed
 - Can increase overall model fidelity incrementally

Initial Random Sequencing



feedback

feedforward

Execution sequence goes from upper left corner to lower right corner

Problem: Each instance of feedback requires an iteration

Initial, random arrangement of modules on the N-square matrix diagonal.

Ordered Sequence

The need

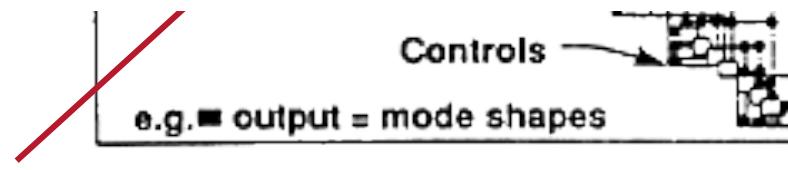
Identify coupling between discipline or subsystems in a complex system.

The solution

Automatic identification of subsystems based on interdependence of design variable and constraints.



Ref: D.V. Steward. “Partitioning and Tearing Systems of Equations”, SIAM Journal of Numerical Analysis. Ser. B, vol.2, no.2, 1965, pp.345-365

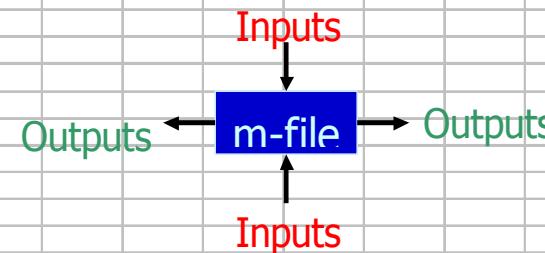


Systematic permutation groups and reduces number of feedback loops

Modules in the N-square matrix resequenced to reduce feedback.

Example: Overview N² Diagram

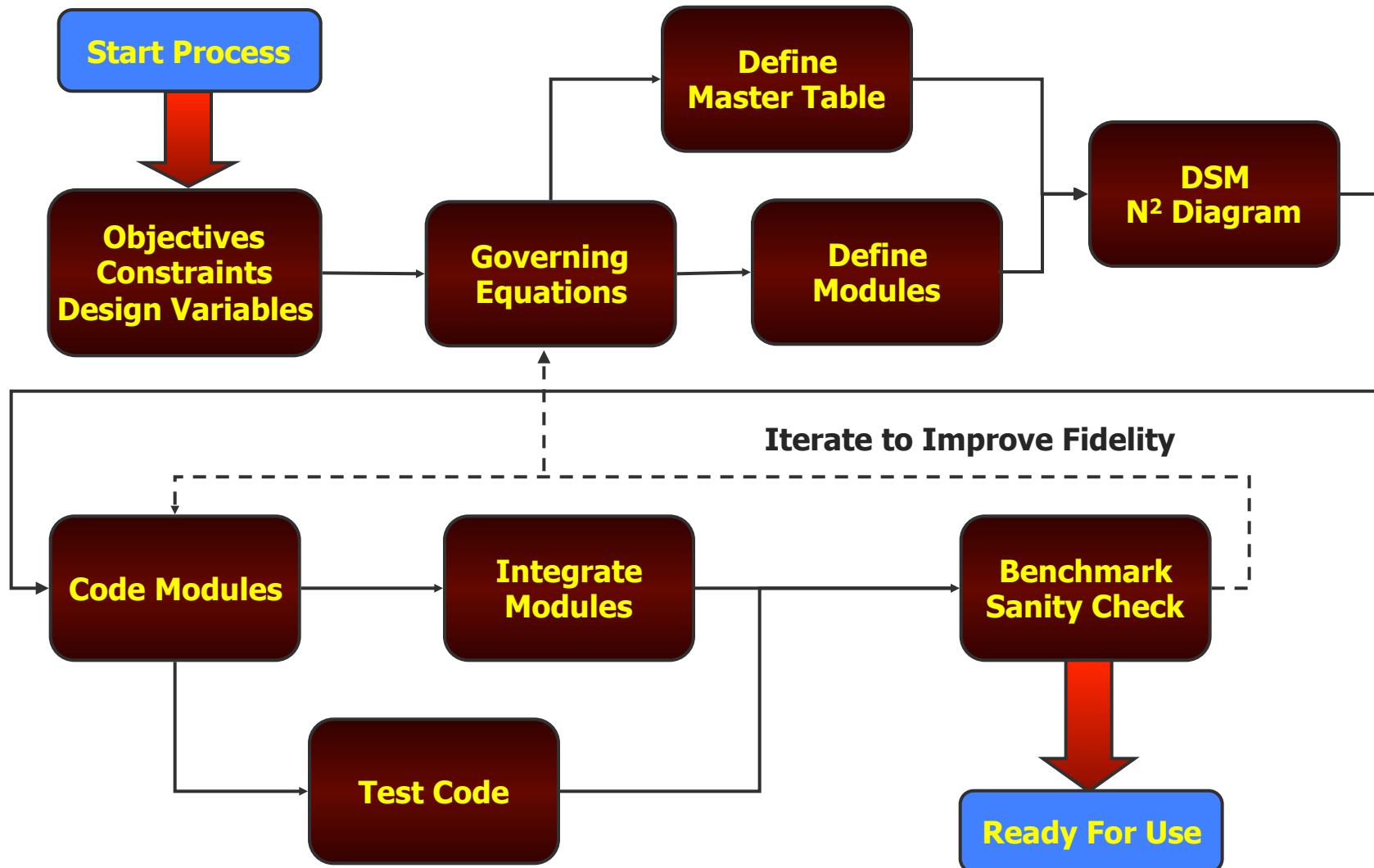
TMAS N-Squared Diagram



Coding - Benchmarking

- Coding of modules can be done in parallel, once the I/O structure has been decided
- Use “dummy” input data to exercise modules in isolation
- Integrate modules step-by-step starting from upper left corner in N²-Diagram
- Do end-to-end simulation test before release
- Benchmark (“validate”) simulation against known cases (experimental data)

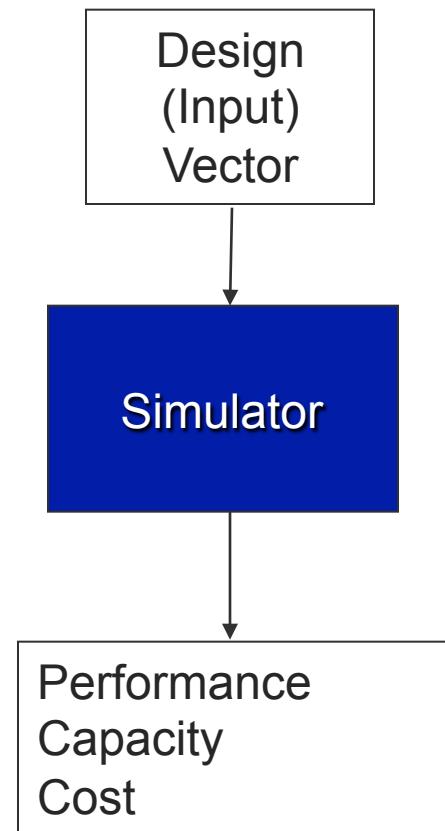
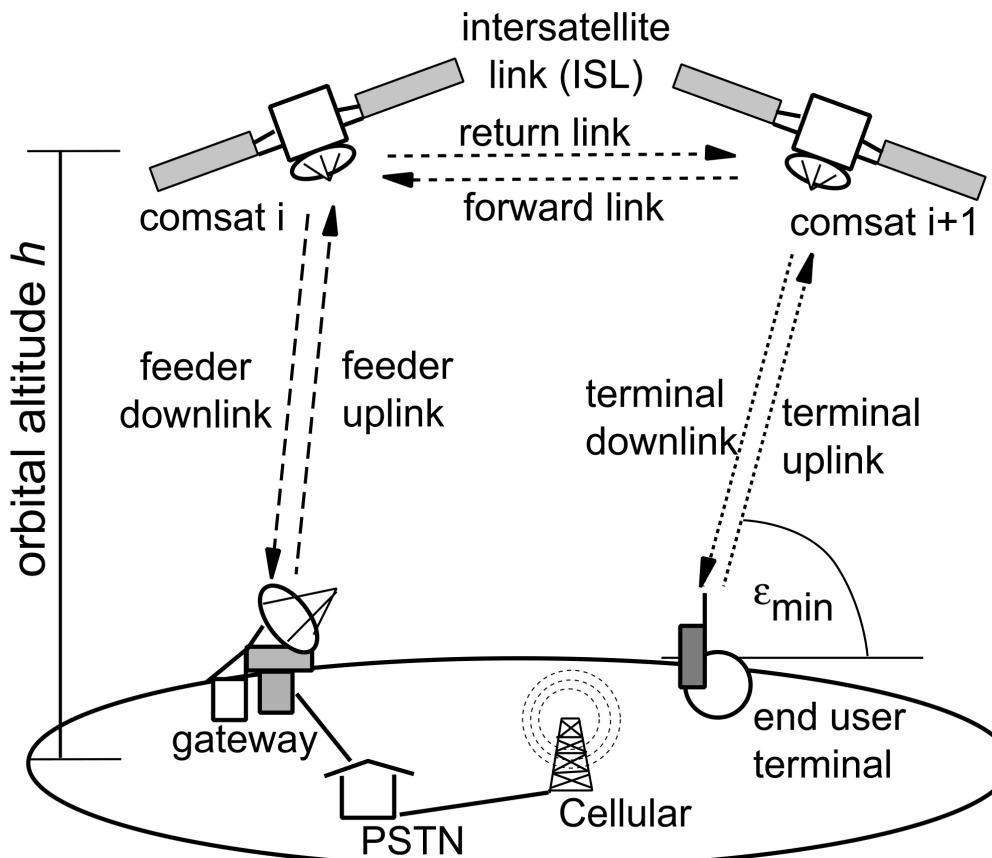
Recap



Example

de Weck, O. L. and Chang D., "Architecture Trade Methodology for LEO Personal Communication Systems ", *20th International Communications Satellite Systems Conference*, Paper No. AIAA-2002-1866, Montréal, Québec, Canada, May 12-15, 2002

Example: Communications Satellites



Can we quantify the conceptual system design problem using modeling and simulation?

Design (Input) Vector X

- The design variables are:
 - Constellation Type: C
 - Orbital Altitude: h
 - Minimum Elevation Angle: ε_{\min}
 - Satellite Transmit Power: P_t
 - Antenna Size: D_a
 - Multiple Access Scheme MA:
 - Network Architecture: ISL

$x_{1440} =$

```
C: 'walker'  
h: 2000  
emin: 12.5000  
Pt: 2400  
DA: 3  
MA: 'MFCD'  
ISL: 0
```

Design Space

Polar, Walker	
500,1000,1500,2000	[km]
2.5,7.5,12.5	[deg]
200,400,800,1600,2400	[W]
1.0,2.0,3.0	[m]
MF-TDMA, MF-CDMA	[-]
yes, no	[-]

This results in a 1440 full factorial, combinatorial conceptual design space

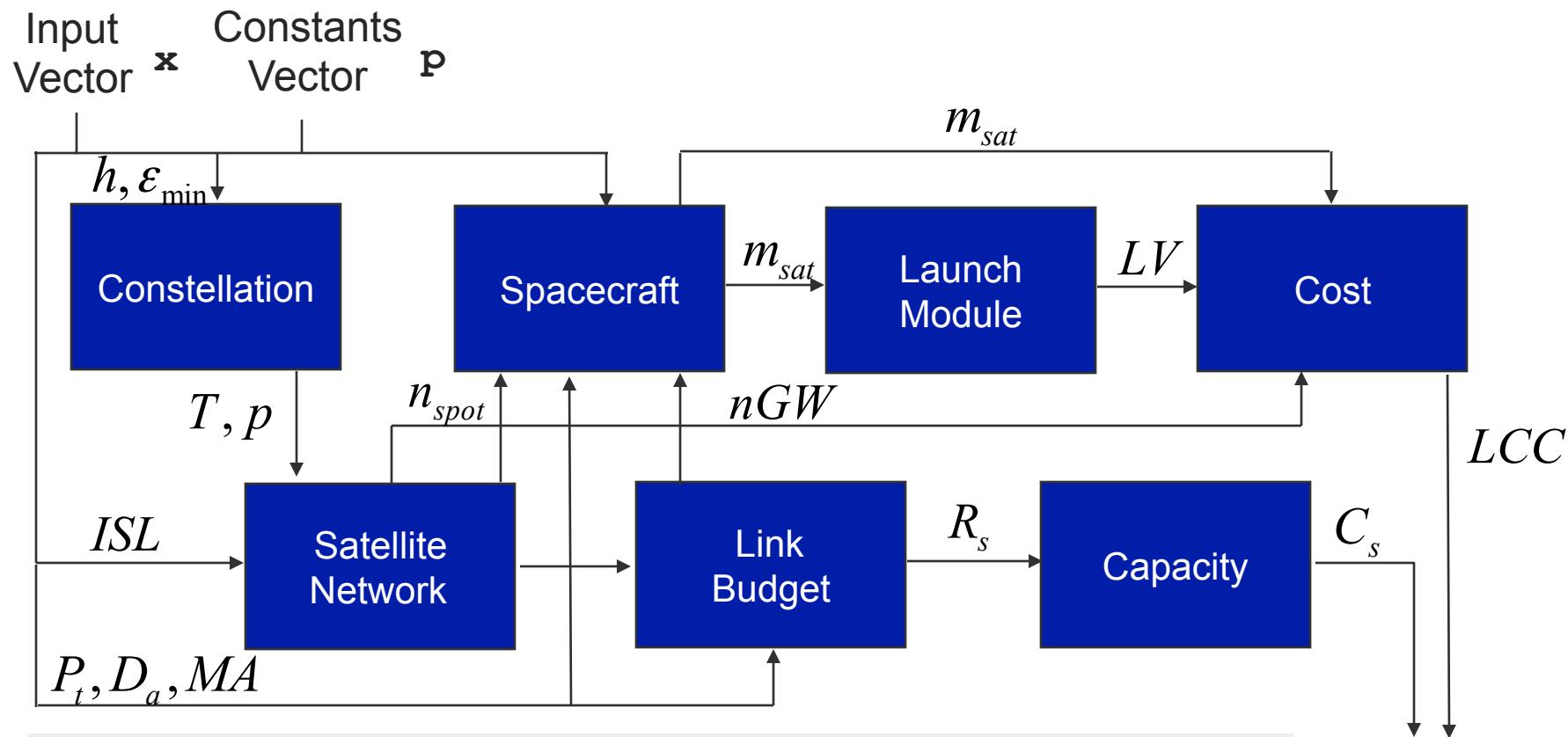
Objective Vector (Output) J

- Performance (fixed)
 - Data Rate per Channel: $R=4.8$ [kbps]
 - Bit-Error Rate: $p_b=10^{-3}$
 - Link Fading Margin: 16 [dB]
- Capacity
 - C_s : Number of simultaneous duplex channels
 - C_{life} : Total throughput over life time [min]
- Cost
 - Lifecycle cost of the system (LCC [\$]), includes:
 - Research, Development, Test and Evaluation (RDT&E)
 - Satellite Construction and Test
 - Launch and Orbital Insertion
 - Operations and Replenishment
 - Cost per Function, CPF [\$/min]

Consider

$$J_{1440} = \begin{cases} C_s : 1.4885e+005 \\ C_{\text{life}} : 1.0170e+011 \\ \text{LCC} : 6.7548e+009 \\ \text{CPF} : 6.6416e-002 \end{cases}$$

Multidisciplinary Simulator Structure



m_{sat} Satellite Mass

T Number of Satellites

p Number of orbital planes

n_{spot} Number of spot beams

nGW Number of gateways

LV Launch vehicle selection

Output Vector J

Note: Only partial input-output relationships shown

Governing Equations

a) Physics-Based Models

Energy per bit over noise ratio:

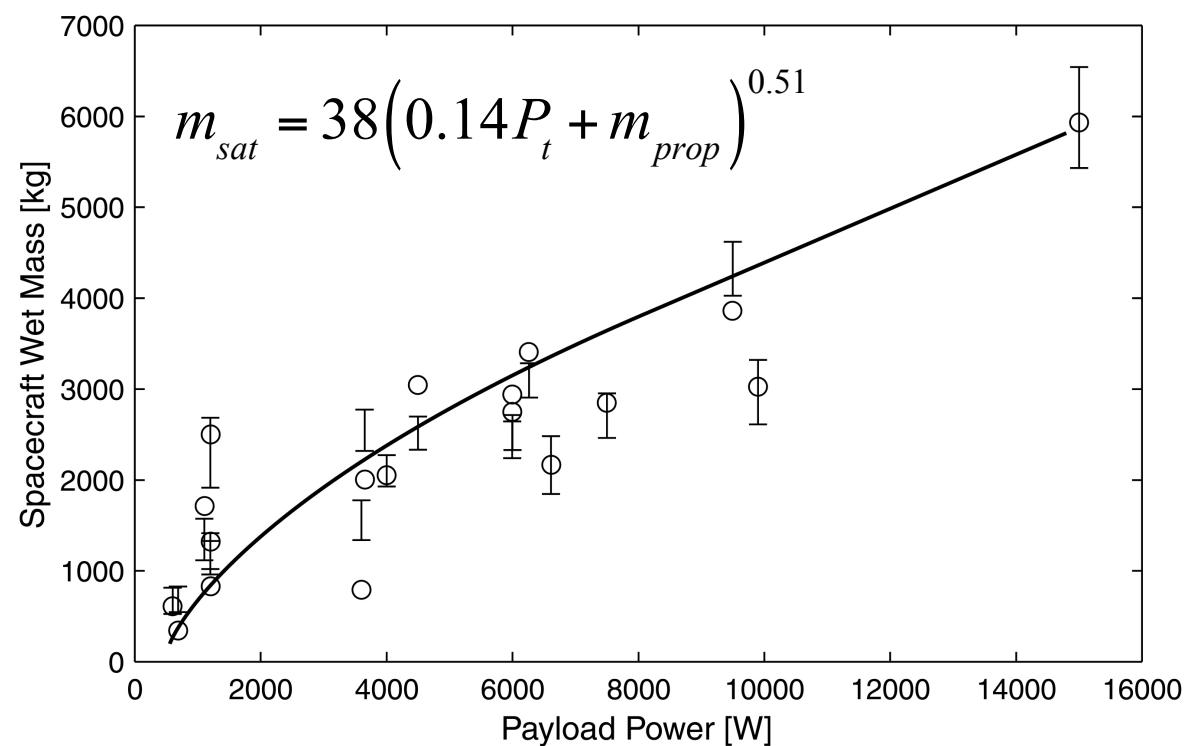
$$\frac{E_b}{N_0} = \frac{P G_r G_t}{k L_{\text{space}} L_{\text{add.}} T_{\text{sys.}} R}$$

(Link Budget)

b) Empirical Models

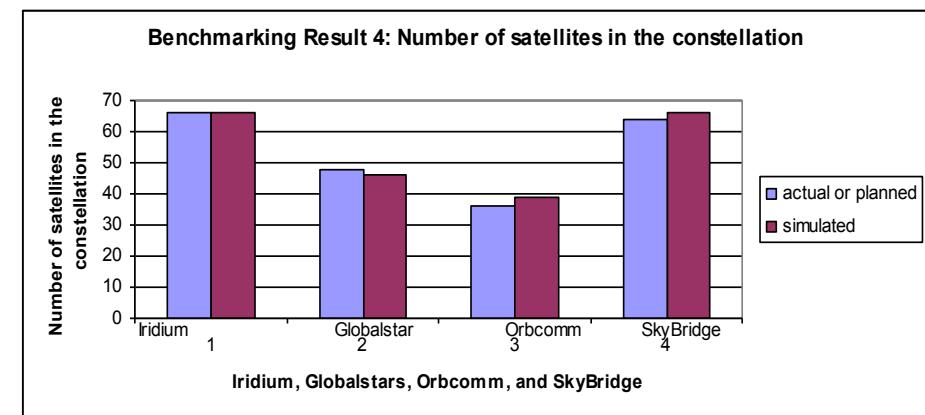
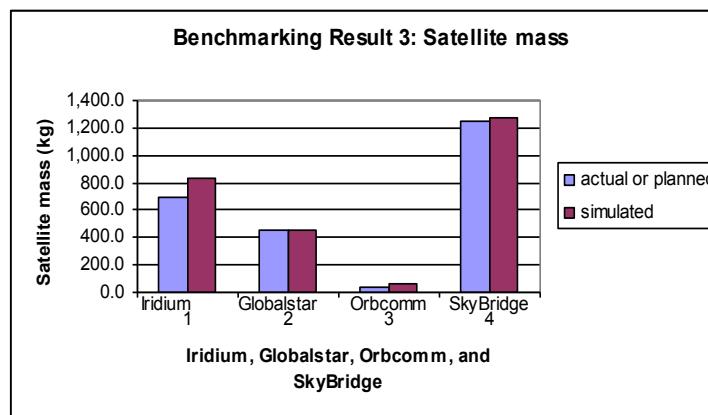
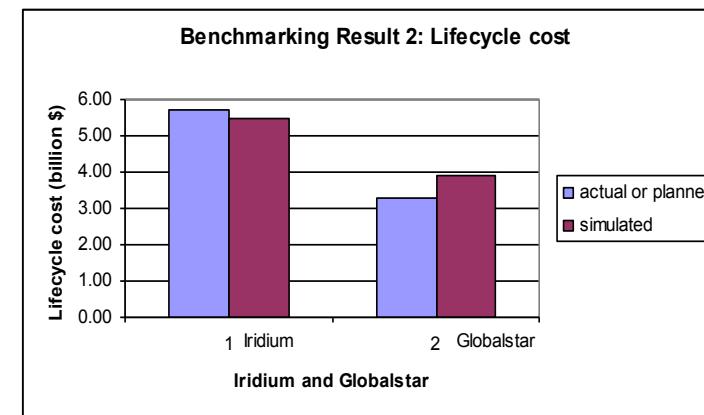
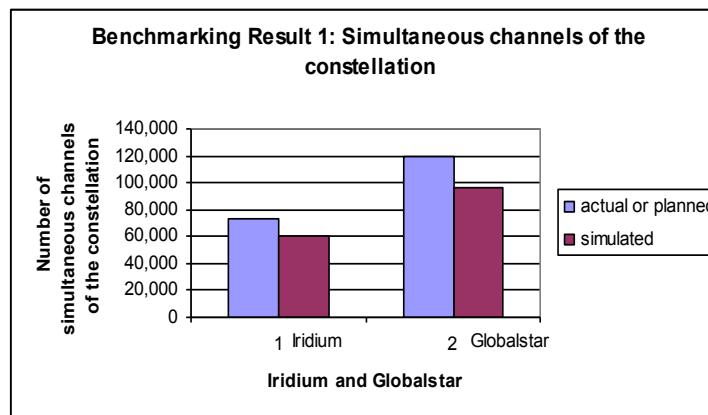
(Spacecraft)

Scaling models derived from FCC database

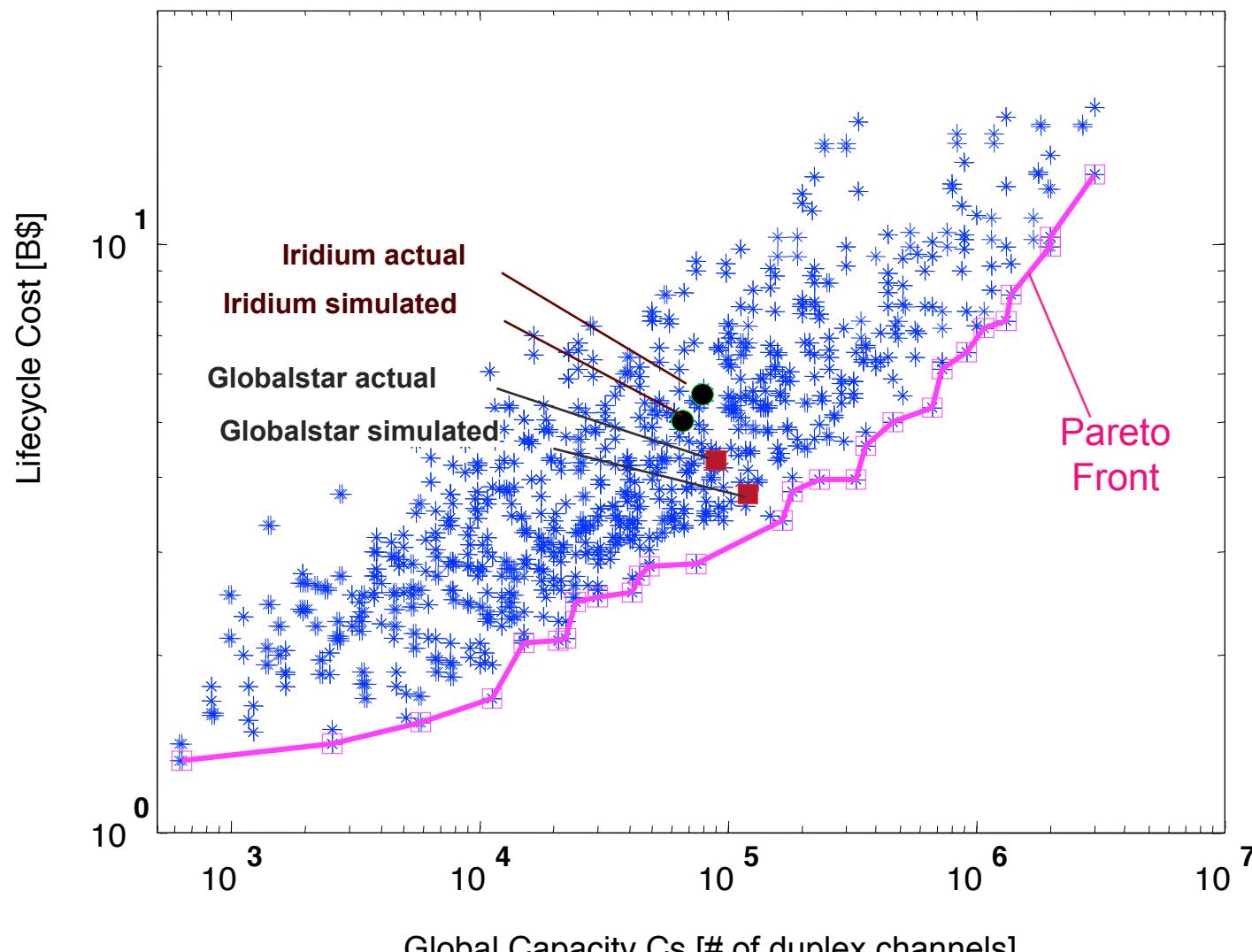


Benchmarking

Benchmarking is the process of validating a model or simulation by comparing the predicted response against reality.



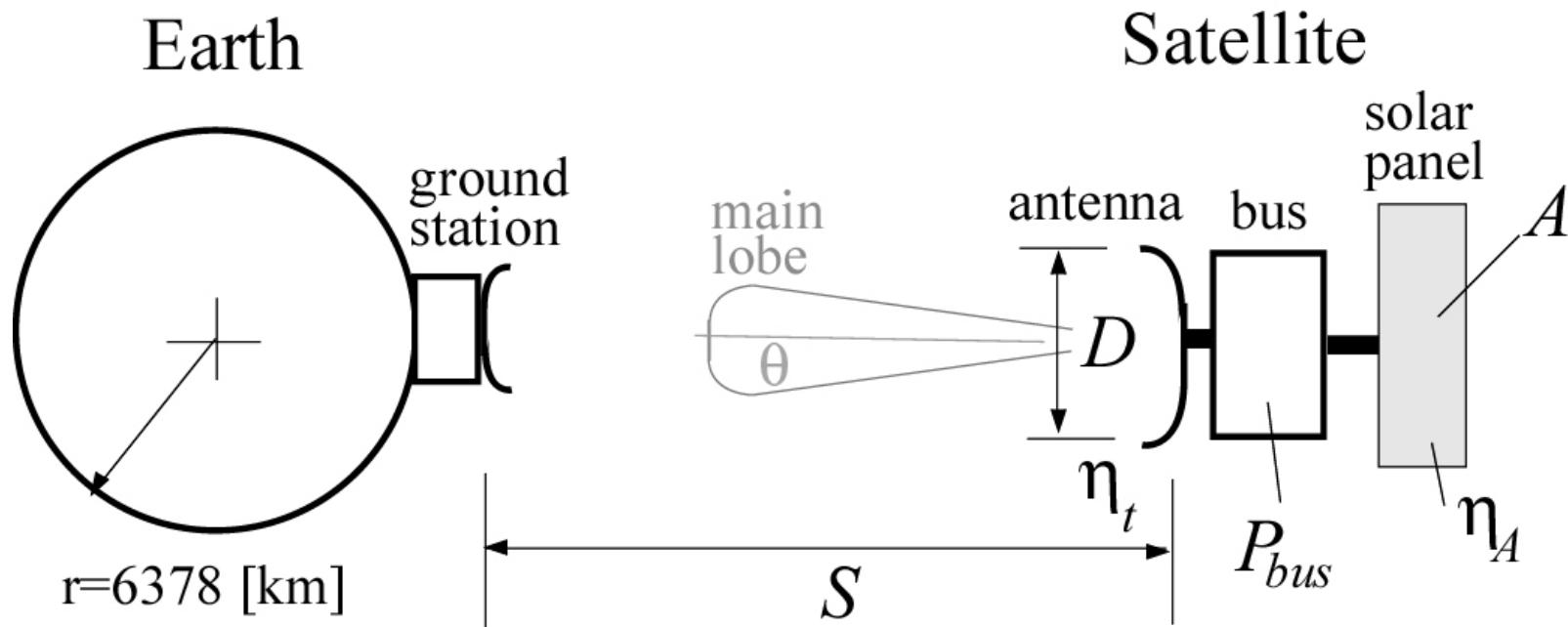
Simulation Results



Simple Example (Prep for Homework A1)

Example: Communications Satellite

Design of a geosynchronous communications satellite



Design problem (Define D.V., objectives, constraints):

How should antenna (D) and solar array (A) be sized for a given orbital period (p) such that a data rate requirement ($R=R_{req}$) is met, while minimizing cost (C) ?

Comsat: Governing Equations

Objective: $\min C$, *Constraint:* $R \geq R_{req}$

Communications: $R = \alpha P_t \frac{D^2 \eta_t}{16 S^2}$ [bps]
(link budget)

Power: $P_t = A \eta_A W_o \cos(\theta_{avg}) - P_{bus}$ [W]
(power budget)

Orbits: $p = 1.66 \cdot 10^{-4} (S + r_E)^{3/2}$ [min]
(orbital period)

Cost: $C = 2500 \cdot D^2 + 12000 \cdot (A + 1) + 100 \cdot P_{bus}$ [\$]
(cost budget)

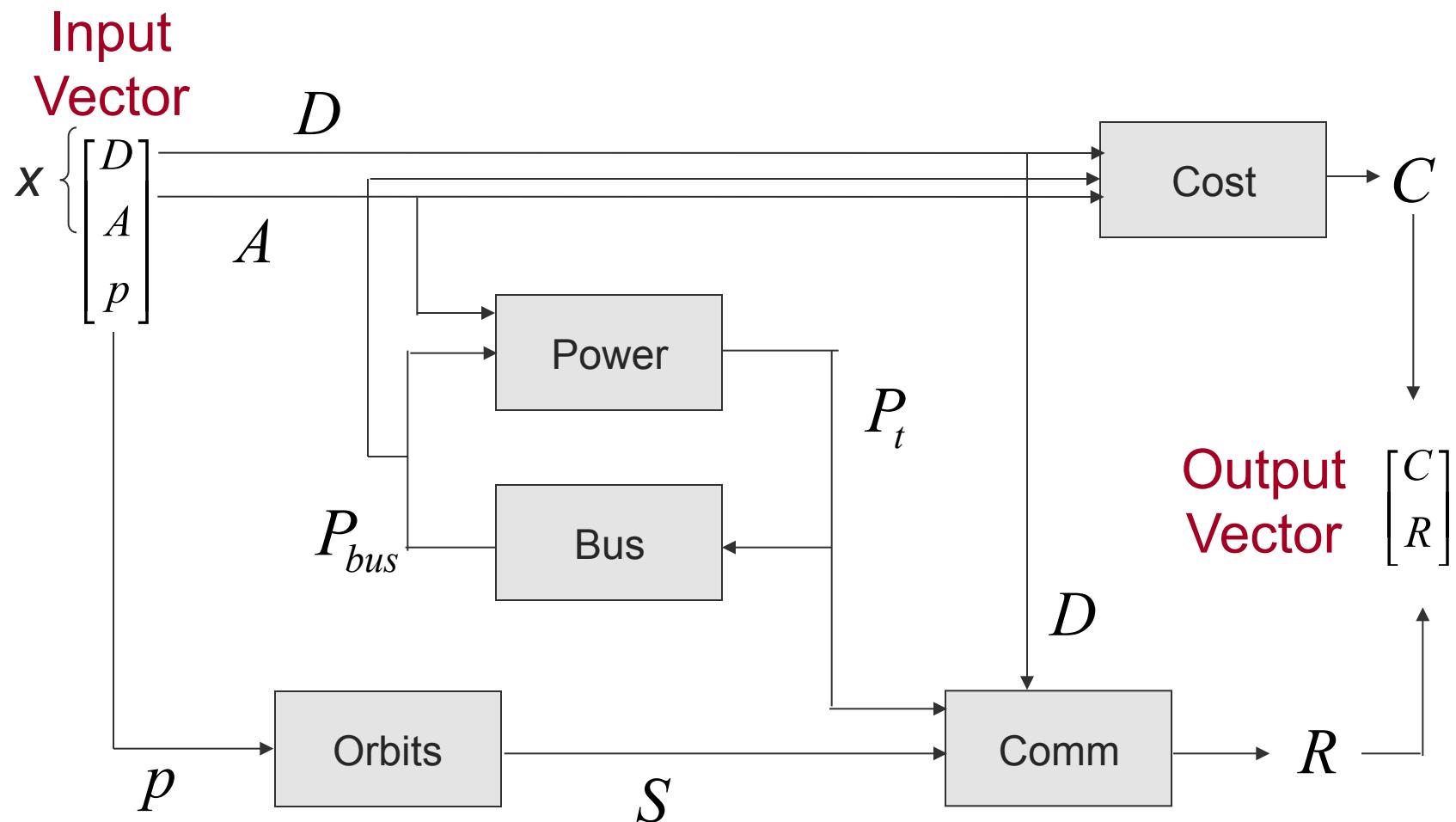
Bus Engineering: $P_{bus} = 10 \cdot \sqrt{P_t}$ [W]

ComSat: Master Table

D	Antenna Diameter	[m]	design var.
A	Solar Panel Area	[m^2]	design var.
p	Orbital Period	[min]	design var.
R	Data Rate	[bps]	constraint
C	Cost	[\\$]	objective
P_t	Transmitter Power	[W]	dependent
P_{bus}	Bus Power	[W]	dependent
θ_a	Sun incidence angle	[deg]	parameter
$\eta_{a,t}$	array/xmit efficiencies	[%]	parameter
S	Orbital altitude	[km]	dependent
α	constant	[$-$]	parameter
W_o	Solar constant	[W/m ²]	parameter

ComSat Block Diagram

BLOCK DIAGRAM



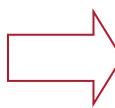
ComSat N² Diagram

In	p	D	A		D,A	
	Orbits	S				
		Comm				R
	Pt	Power	Pt			
		Pbus	Bus	Pbus		
					Cost	C
						Out

iterative block

Computational Implementation

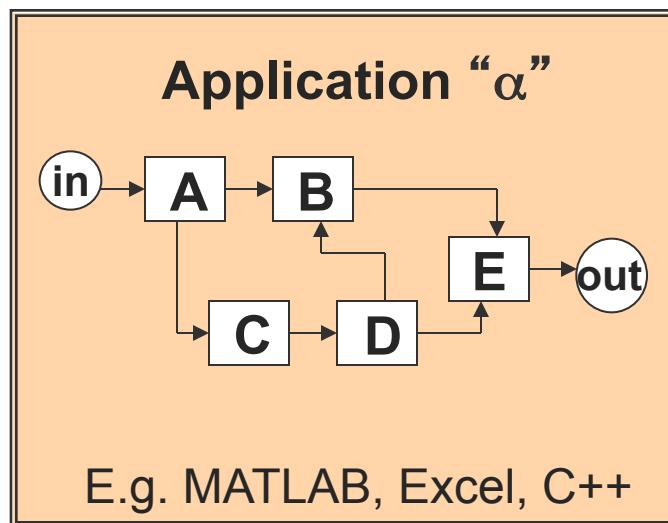
Computational Issues

- Computer technologies have been changing the environment of engineering design - enabling MDO
 - Hardware: Advances in processor speed, memory and storage
 - Software: Powerful disciplinary analysis and simulation programs (e.g. Nastran, Fluent ...)
 - This also creates new difficulties: Most activities involve stand-alone programs and many engineers spend 50-80% of their time organizing data and moving it back-and-forth between applications
-  Data must be shared between disciplines more easily

Coupling Disparate Tools

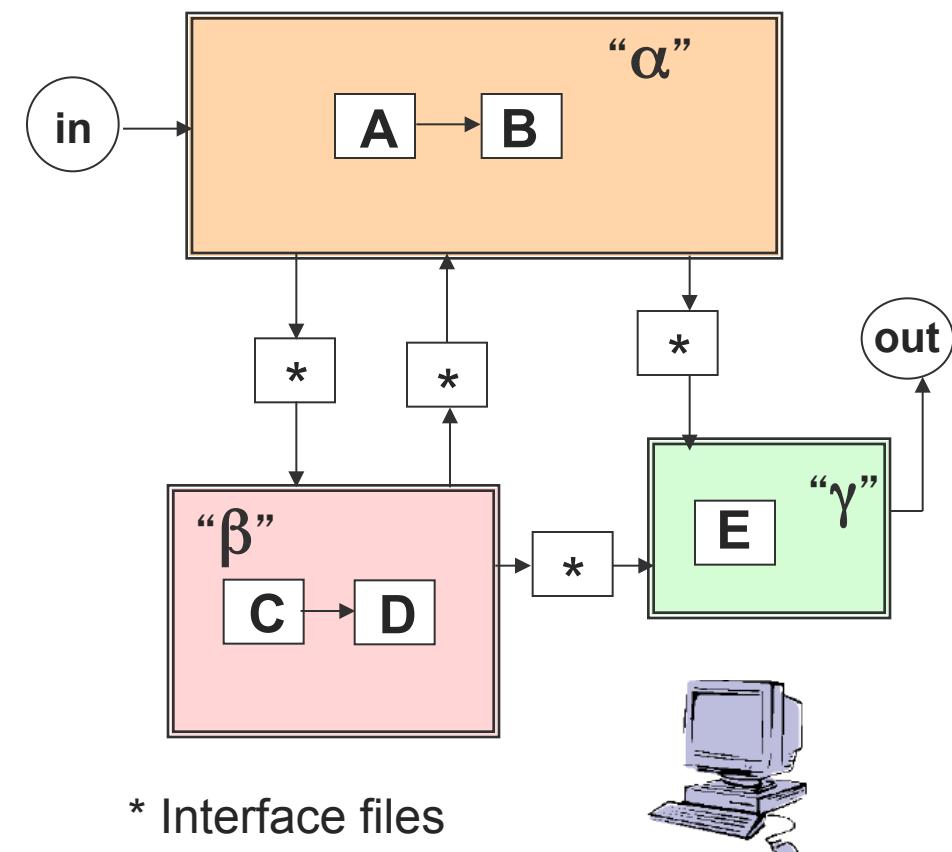
Case 1:

Within one application on the same computer



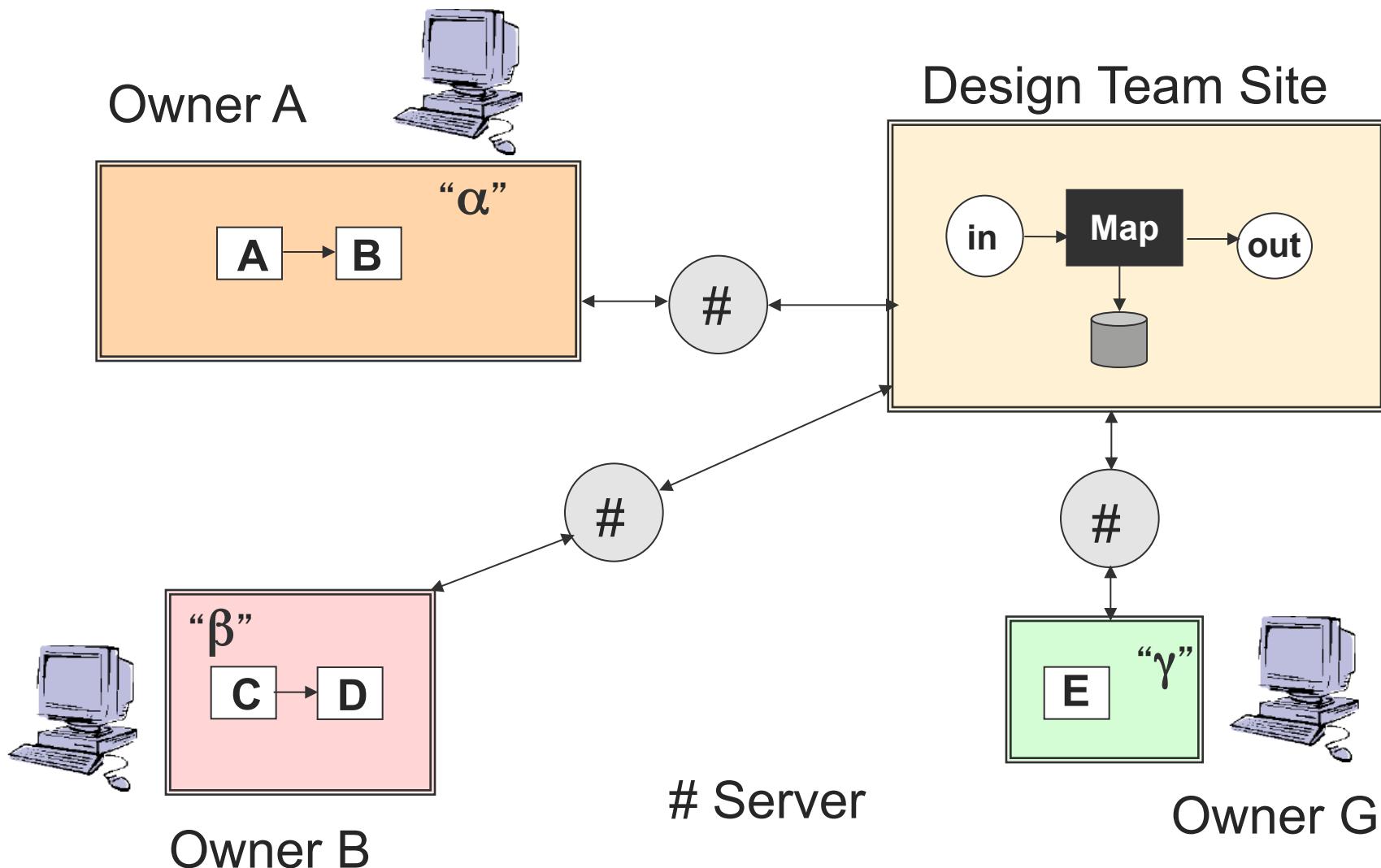
Case 2:

Between different applications on the same computer



Implementation and Ownership

Case 3: In a LAN or WAN environment



Modeling-Simulation Environments

- Integrated Modeling & Simulation
 - Write functions and integrate via Master script
 - Matlab, Mathematica are popular environments
- ICEMaker
 - Developed at Caltech/JPL
 - linked spreadsheets (client server)
- DOME (MIT) - CO (Oculus)
 - DOME based peer-peer system
 - API's into numerous Engineering applications
- FIPER (Simulia – Dassault Systems)
 - Client-server enterprise system
 - Targeted at the corporate environment
- PHX Model Center
 - Phoenix Integration – Flagship Product
 - Desktop Integration Environment

Lecture Summary

- Follow a logical model & simulation development process, don't forget benchmarking
- Decomposition is crucial in order to facilitate code integration and coupling
- N²/DSM Matrix is useful tool to organize data
- Minimize the number of feedback loops

References

- Rogers, James L.: DeMAID/GA User's Guide - Design Manager's Aid for Intelligent Decomposition with a Genetic Algorithm, April 1996, NASA TM – 110241.
- Steward, D.V., 1981, *Systems Analysis and Management: Structure, Strategy, and Design*, New York: Petrocelli.
- D.V. Steward. “*Partitioning and Tearing Systems of Equations*”, *SIAM Journal of Numerical Analysis. Ser.B*, vol.2, no.2, 1965, pp.345-365
- de Weck, O. L. and Chang D., ”Architecture Trade Methodology for LEO Personal Communication Systems “, 20th International Communications Satellite Systems Conference, Paper No. AIAA-2002-1866, Montréal, Québec, Canada, May 12-15, 2002
- Ulrich, K.T., and S.D. Eppinger, 1995, *Product Design and Development* , McGraw-Hill.
- The Design Structure Matrix Website, <http://www.dsmweb.org/>

Lecture 04: Multidisciplinary System Analysis and Design Optimization (MSADO)

Numerical Optimization 1

February 2, 2015

Prof. Douglas Allaire

Today's Topics

- Existence & Uniqueness of an Optimum Solution
- Karush-Kuhn-Tucker Conditions
- Convex Spaces
- Unconstrained Problems

Disclaimer!

- This is not a classic optimization class...
- The aim is not to teach you the details of optimization algorithms, but rather to expose you to different methods.
- We will utilize optimization techniques – the goal is to understand enough to be able to utilize them wisely.

Learning Objectives

After the next two lectures you should:

- Be familiar with what gradient-based (and some gradient-free) optimization techniques are available
- Understand the basics of how each technique works
- Be able to choose which optimization technique is appropriate for your problem
- Understand what to look for when the algorithm terminates
- Understand why the algorithm might fail

How to choose an algorithm?

- Number of design variables
- Type of design variables (real/integer, continuous/discrete)
- Linear/nonlinear
- Continuous/discontinuous objective behavior
- Equality/inequality constraints
- Discontinuous feasible spaces
- Initial solution feasible/infeasible
- Availability of gradient information
- Simulation code (forward problem) runtime

Standard Problem Definition

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

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

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

$$x_i^l \leq x_i \leq x_i^u \quad i = 1, \dots, n$$

- 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**.

Linear vs. Nonlinear

The objective function is a linear function of the design variables if each design variable appears only to the first power with constant coefficients multiplying it.

$$J(\mathbf{x}) = x_1 + 2x_2 - 3.4x_3 \quad \text{is linear in } \mathbf{x} = [x_1 \ x_2 \ x_3]^T$$

$$J(\mathbf{x}) = x_1x_2 + 2x_2 - 3.4x_3 \quad \text{is nonlinear in } \mathbf{x}$$

$$J(\mathbf{x}) = \cos(x_1) + 2x_2 - 3.4x_3 \quad \text{is nonlinear in } \mathbf{x}$$

Linear vs. Nonlinear

A constraint is a linear function of the design variables if each design variable appears only to the first power with constant coefficients multiplying it.

$$6x_1 - x_2 - 2x_3 < 10 \quad \text{is linear in } \mathbf{x}$$

$$6x_1 - x_2 - 2x_3^2 < 10 \quad \text{is nonlinear in } \mathbf{x}$$

$$6x_1 - \sin(x_2) - 2x_3 < 10 \quad \text{is nonlinear in } \mathbf{x}$$

Iterative Optimization Procedures

Many optimization algorithms are iterative:

where

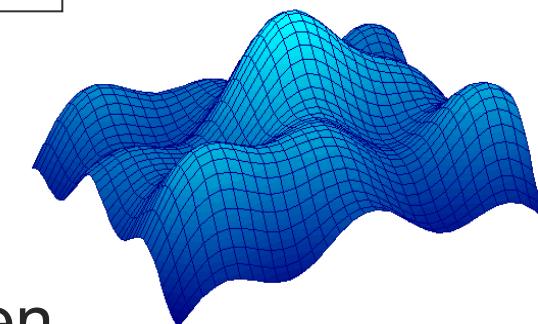
$$\mathbf{x}^q = \mathbf{x}^{q-1} + \alpha^q \mathbf{S}^q$$

q =iteration number

\mathbf{S} =vector search direction

α =scalar distance

and the initial solution \mathbf{x}^0 is given

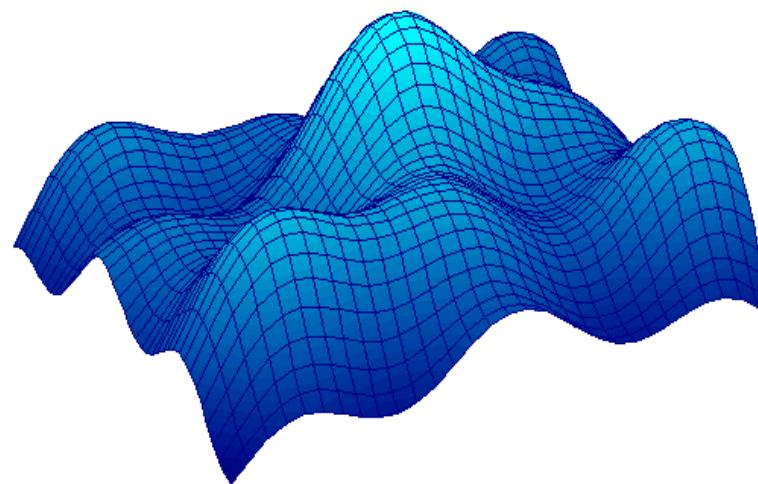


The algorithm determines the search direction \mathbf{S} according to some criteria.

Gradient-based algorithms use gradient information to decide where to move. Gradient-free algorithms use sampling and/or heuristics.

Iterative Optimization Procedures

Matlab demo



Gradient Vector

Consider a function $J(\mathbf{x})$, $\mathbf{x}=[x_1, x_2, \dots, x_n]$

The gradient of $J(\mathbf{x})$ at a point \mathbf{x}^0 is a vector of length n :

$$\nabla J(\mathbf{x}^0) = \begin{bmatrix} \frac{\partial J}{\partial x_1}(\mathbf{x}^0) \\ \frac{\partial J}{\partial x_2}(\mathbf{x}^0) \\ \vdots \\ \frac{\partial J}{\partial x_n}(\mathbf{x}^0) \end{bmatrix}$$

Each element in the vector is evaluated at the point \mathbf{x}^0 .

Hessian Matrix

Consider a function $J(\mathbf{x})$, $\mathbf{x}=[x_1, x_2, \dots, x_n]$

The second derivative of $J(\mathbf{x})$ at a point \mathbf{x}^0 is a matrix of size $n \times n$:

$$\mathbf{H}(\mathbf{x}^0) \equiv \nabla^2 J(\mathbf{x}^0) = \begin{bmatrix} \frac{\partial^2 J}{\partial x_1^2} & \frac{\partial^2 J}{\partial x_1 \partial x_2} & \cdots & \cdots & \frac{\partial^2 J}{\partial x_1 \partial x_n} \\ \frac{\partial^2 J}{\partial x_1 \partial x_2} & \ddots & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ \frac{\partial^2 J}{\partial x_1 \partial x_n} & & & & \frac{\partial^2 J}{\partial x_n^2} \end{bmatrix}$$

Each element in the matrix is evaluated at the point \mathbf{x}^0 .

Gradients & Hessian Example

$$J(\mathbf{x}) = 3x_1 + x_1x_2 + x_3^2 + 6x_2^3x_3$$

Taylor Series

Consider scalar case:

$$f(z) = f(z^0) + \frac{df}{dz} \Big|_{z^0} (z - z^0) + \frac{1}{2} \frac{d^2f}{dz^2} \Big|_{z^0} (z - z^0)^2 + \dots$$

When function depends on a vector:

$$\begin{aligned} J(\mathbf{x}) &= J(\mathbf{x}^0) + \underbrace{\left[\nabla J(\mathbf{x}^0) \right]^T}_{1 \times n} (\mathbf{x} - \mathbf{x}^0) \\ &\quad + \frac{1}{2} \underbrace{(\mathbf{x} - \mathbf{x}^0)^T}_{1 \times n} \underbrace{\mathbf{H}(\mathbf{x}^0)}_{n \times n} \underbrace{(\mathbf{x} - \mathbf{x}^0)}_{n \times 1} + \dots \end{aligned}$$

The gradient vector and Hessian matrix can be approximated using finite differences if they are not available analytically or using adjoints (L8).

Existence & Uniqueness of an Optimal Solution

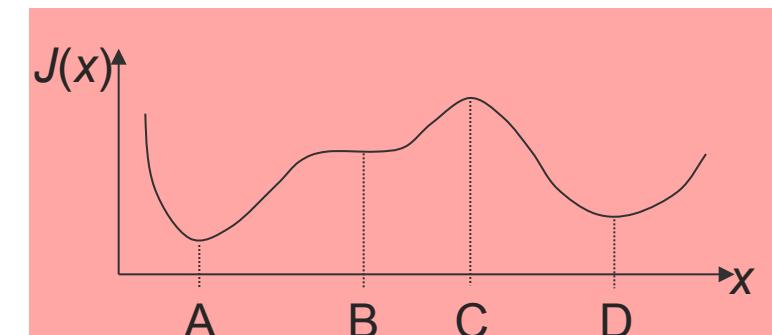
- Usually cannot guarantee that global optimum is found
 - multiple solutions may exist
 - numerical ill-conditioning
 - start from several initial solutions
- Can determine mathematically if have relative minimum
- Under certain conditions can guarantee global optimum (special class of optimization problems or with global optimization methods)
- It is very important to interrogate the “optimum” solution

Existence & Uniqueness: Unconstrained Problems

- Unconstrained problems: at minimum, gradient must vanish

$$\|\nabla J(\mathbf{x}^*)\| = 0$$

- \mathbf{x}^* is a stationary point of J
- necessary but not sufficient
- here A,B,C,D all have $\nabla J=0$



- Calculus: at minimum, second derivative > 0
- Vector case: at minimum, $\mathbf{H}(\mathbf{x}^*) > 0$ (positive definite)

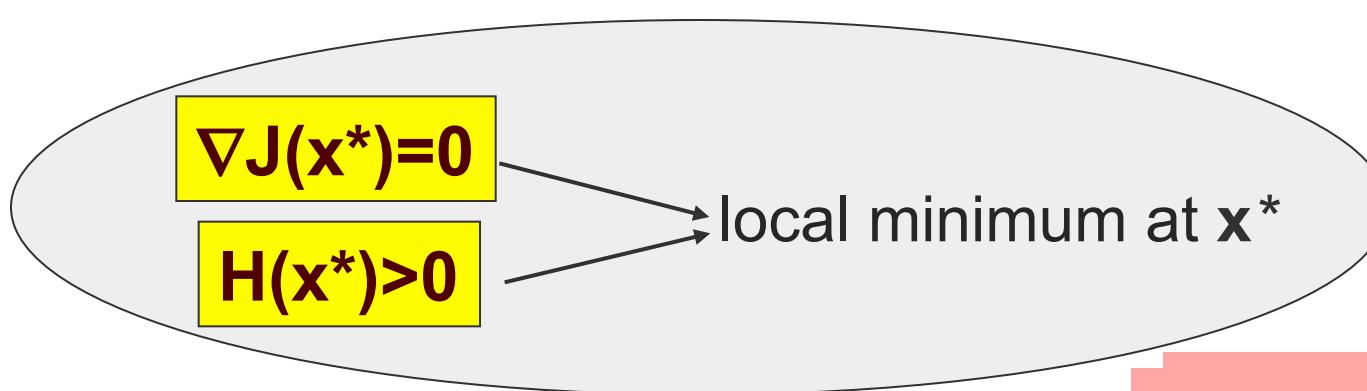
SPD Matrices

- Positive definiteness: $\mathbf{y}^T \mathbf{H} \mathbf{y} > 0$ for all $\mathbf{y} \neq 0$
- Consider the i^{th} eigenmode of \mathbf{H} : $\mathbf{H} \mathbf{v}_i = \lambda_i \mathbf{v}_i$
- If \mathbf{H} is a symmetric matrix, then the eigenvectors of \mathbf{H} form an orthogonal set: $\mathbf{v}_i^T \mathbf{v}_j = \delta_{ij}$
- Any vector \mathbf{y} can be thought of as a linear combination of eigenvectors: $\mathbf{y} = \sum a_i \mathbf{v}_i$
- Then: $\mathbf{y}^T \mathbf{H} \mathbf{y} = \sum_i a_i \mathbf{v}_i^T \mathbf{H} \sum_j a_j \mathbf{v}_j = \sum_{i,j} a_i \mathbf{v}_i^T \lambda_j a_j \mathbf{v}_j = \sum_i a_i^2 \lambda_i$
- Therefore, if the eigenvalues of \mathbf{H} are all positive, \mathbf{H} is SPD.

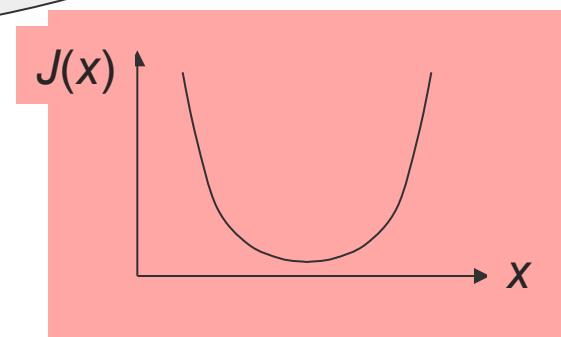
Existence & Uniqueness: Unconstrained Problems

Necessary and sufficient conditions for a minimum
(unconstrained problem):

1. Gradient must vanish
2. Hessian must be positive definite



The minimum is only guaranteed to be a **global** optimum if $H(x) > 0$ for all values of x (e.g. simple parabola).



Existence & Uniqueness: Constrained Problems

At optimum:

- at least one constraint on design is active
- ∇J does not have to be zero

In order to improve design:

- move in direction that decreases objective
- move in direction that does not violate constraints

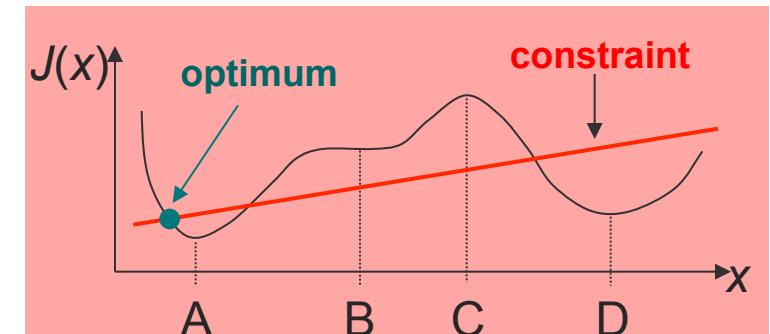
Usable direction = any direction that reduces objective

$$\mathbf{S}^T \nabla J(\mathbf{x}) \leq 0$$

Feasible direction = a direction in which a small move will not violate constraints

$$\mathbf{S}^T \nabla g_i(\mathbf{x}) \leq 0 \quad (\text{for all active constraints } i)$$

Note that these conditions may be relaxed for certain algorithms.



Lagrangian Functions

- A constrained minimization can be written as an unconstrained minimization by defining the Lagrangian function:

$$L(\mathbf{x}, \lambda) = J(\mathbf{x}) + \sum_{j=1}^{m_1} \lambda_j g_j(\mathbf{x}) + \sum_{k=1}^{m_2} \lambda_{m_1+k} h_k(\mathbf{x})$$

- $L(\mathbf{x}, \lambda)$ is called the **Lagrangian function**.
- λ_i is the j^{th} **Lagrange multiplier**
- It can be shown that a stationary point \mathbf{x}^* of $L(\mathbf{x}, \lambda)$ is a stationary point of the original constrained minimization problem.

Lagrangian Example

$$\min J(\mathbf{x}) = x_1^2 + 3x_2^2$$

$$\text{s.t. } x_1 + x_2 = 1$$

Karush-Kuhn-Tucker (KKT) Conditions

If \mathbf{x}^* is optimum, these conditions are satisfied:

1. \mathbf{x}^* is feasible

2. $\lambda_j g_j(\mathbf{x}^*) = 0, \ j=1,\dots,m_1$ 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.

KKT 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*

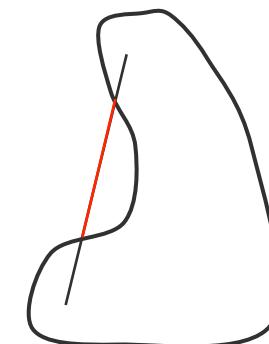
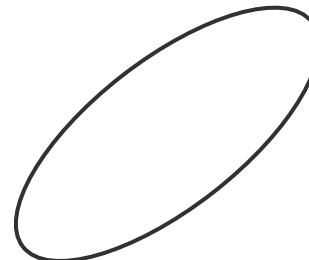
Condition 3: the gradient of the Lagrangian vanishes at the optimum

Convex Sets

Consider a set, and imagine drawing a line connecting any two points in the set.

If every point along that line is inside the set, then the set is **convex**.

If **any** point along that line is outside the set, then the set is **non-convex**.

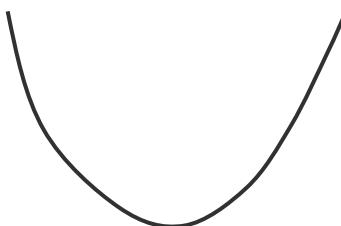


The line connecting points x^1 and x^2 is given by

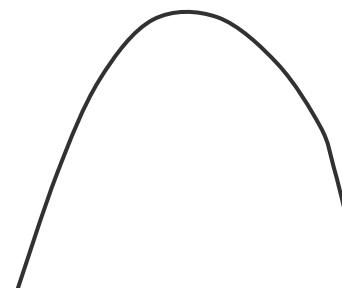
$$w = \theta x^1 + (1-\theta)x^2, \quad 0 \leq \theta \leq 1$$

Convex Functions

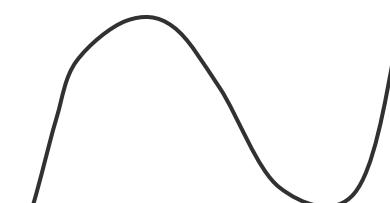
Informal definition: a convex function is one that will hold water, while a concave function will not hold water...



convex



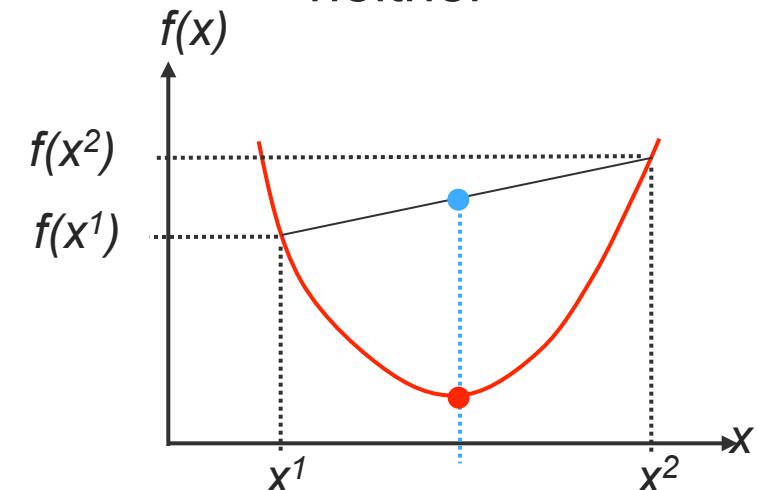
concave



neither

A function $f(x)$ bounding a convex set is convex if:

$$f[\theta \mathbf{x}^1 + (1 - \theta) \mathbf{x}^2] \leq \theta f(\mathbf{x}^1) + (1 - \theta) f(\mathbf{x}^2)$$



Convex Spaces

Pick any two points in the feasible region. If all points on the line connecting these points lie in the feasible region, then the constraint surfaces are convex.

If the objective function is convex, then it has only one optimum (the global one) and the Hessian matrix is positive definite for **all possible** designs.

If the objective function and all constraint surfaces are convex, then the design space is convex, and the KKT conditions are sufficient to guarantee that \mathbf{x}^* is a global optimum.

In general, for engineering problems, the design space is not convex ...

Convergence Criteria

Algorithm has converged when ...

no change in the objective function is obtained

OR the maximum number of iterations is reached

Once the “optimal” solution has been obtained, the KKT conditions should be checked.

Types of Optimization Algorithms

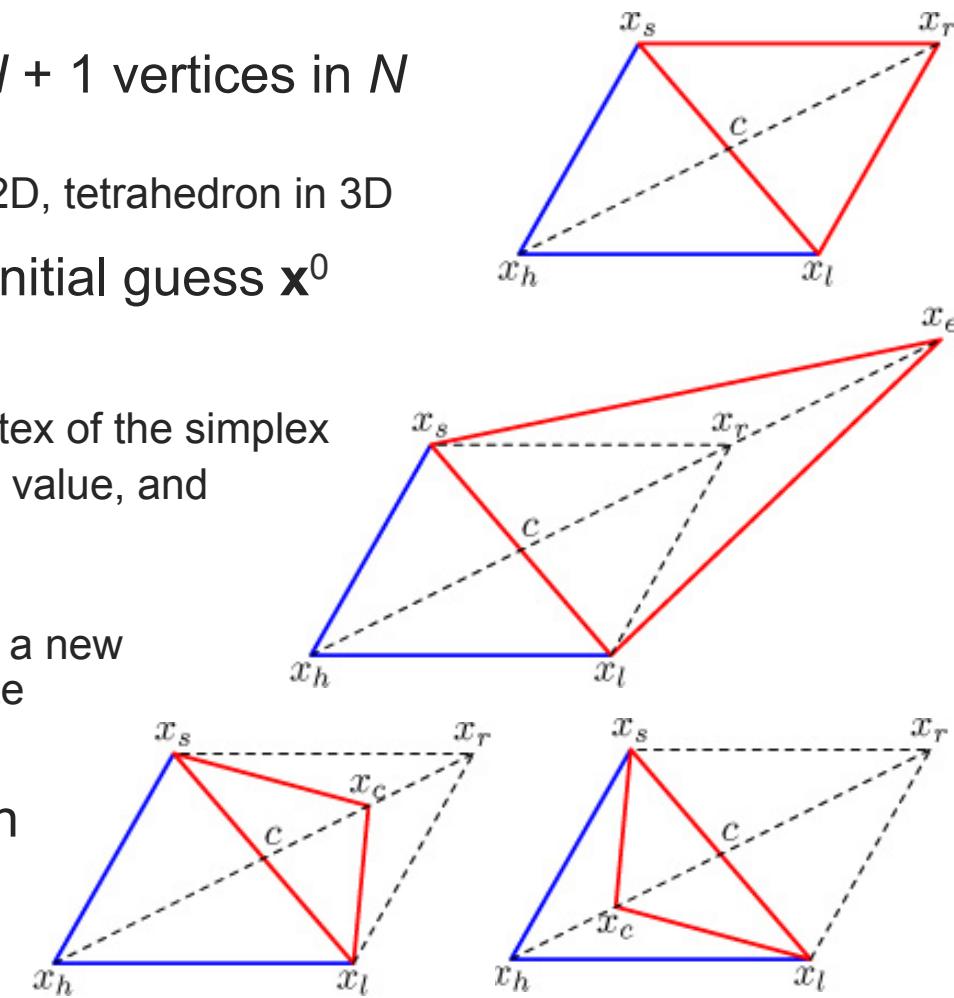
- Global optimization
- Local derivative-free optimization
- Local gradient-based optimization
- Heuristic methods



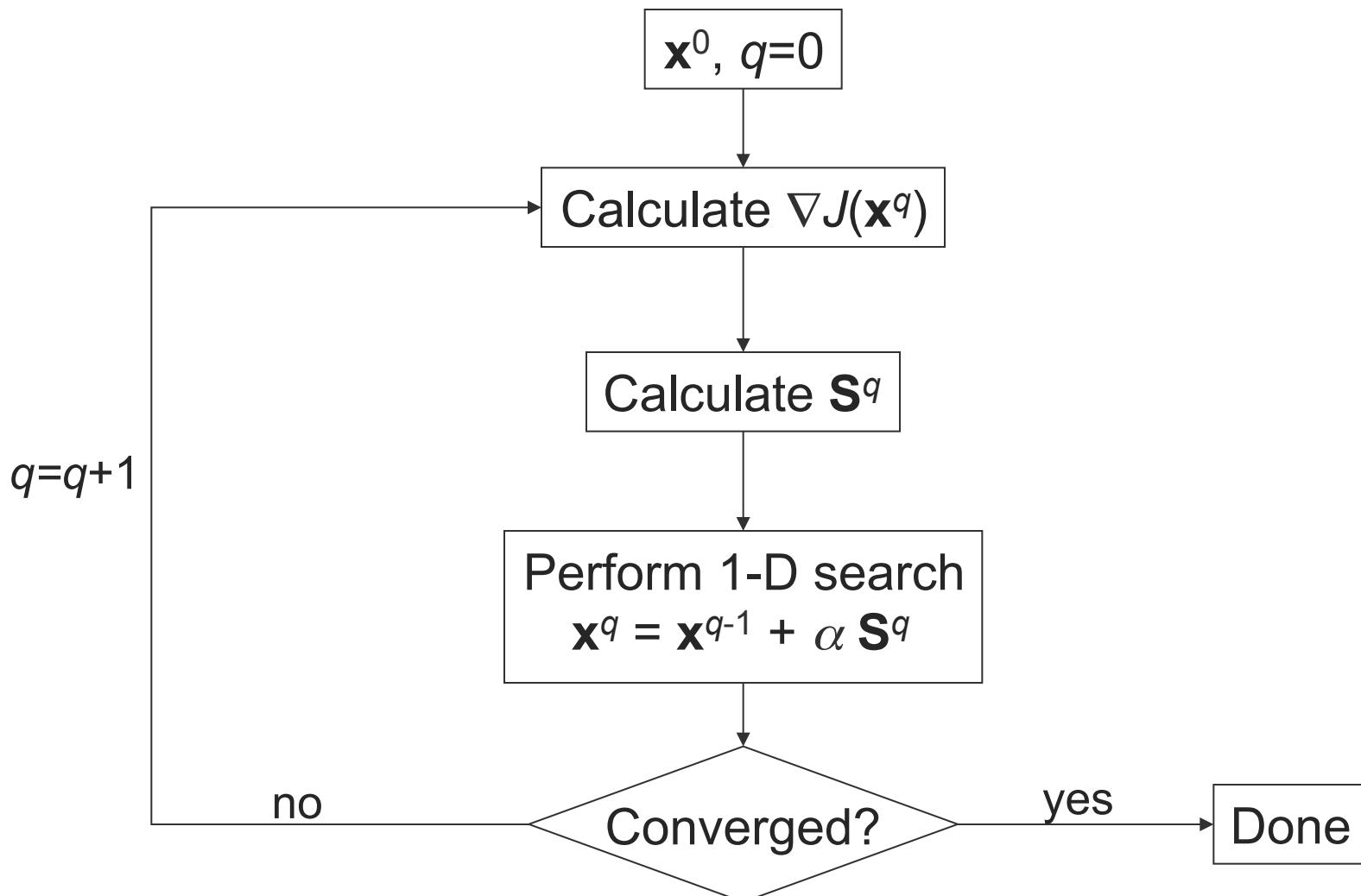
*Most methods
have some
convergence
analysis and/or
proofs.*

Local Derivative Free Optimization: Nelder-Mead Simplex

- A simplex is a special polytope of $N + 1$ vertices in N dimensions
 - e.g., line segment on a line, triangle in 2D, tetrahedron in 3D
- Form an initial simplex around the initial guess \mathbf{x}^0
- Repeat the following general steps:
 - Compute the function value at each vertex of the simplex
 - Order the vertices according to function value, and discard the worst one
 - Generate a new point by “reflection”
 - If the new point is acceptable, generate a new simplex. Expand or contract simplex size according to quality of new point.
- Converges to a local optimum when the objective function varies smoothly and is unimodal, but can converge to a non-stationary point in some cases
- “fminsearch” in Matlab



Gradient-Based Optimization Process



Unconstrained Problems: Gradient-Based Solution Methods

- First-Order Methods
 - use gradient information to calculate **S**
 - steepest descent method
 - conjugate gradient method
 - quasi-Newton methods
- Second-Order Methods
 - use gradients and Hessian to calculate **S**
 - Newton method
- Methods to calculate gradients in Lecture 8.
- Often, a constrained problem can be cast as an unconstrained problems and these techniques used.

Steepest Descent

$$\mathbf{S}^q = -\nabla J(\mathbf{x}^{q-1})$$

- $\nabla J(\mathbf{x})$ is the direction of max decrease of J at \mathbf{x}

Algorithm:

choose \mathbf{x}^0 , set $\mathbf{x}=\mathbf{x}^0$

repeat until converged:

$$\mathbf{S} = -\nabla J(\mathbf{x})$$

choose α to minimize $J(\mathbf{x}+\alpha\mathbf{S})$

$$\mathbf{x} = \mathbf{x} + \alpha\mathbf{S}$$

- Doesn't use any information from previous iterations
- converges slowly
- α is chosen with a 1-D search (interpolation or Golden section)

Conjugate Gradient

$$\mathbf{S}^1 = -\nabla J(\mathbf{x}^0)$$

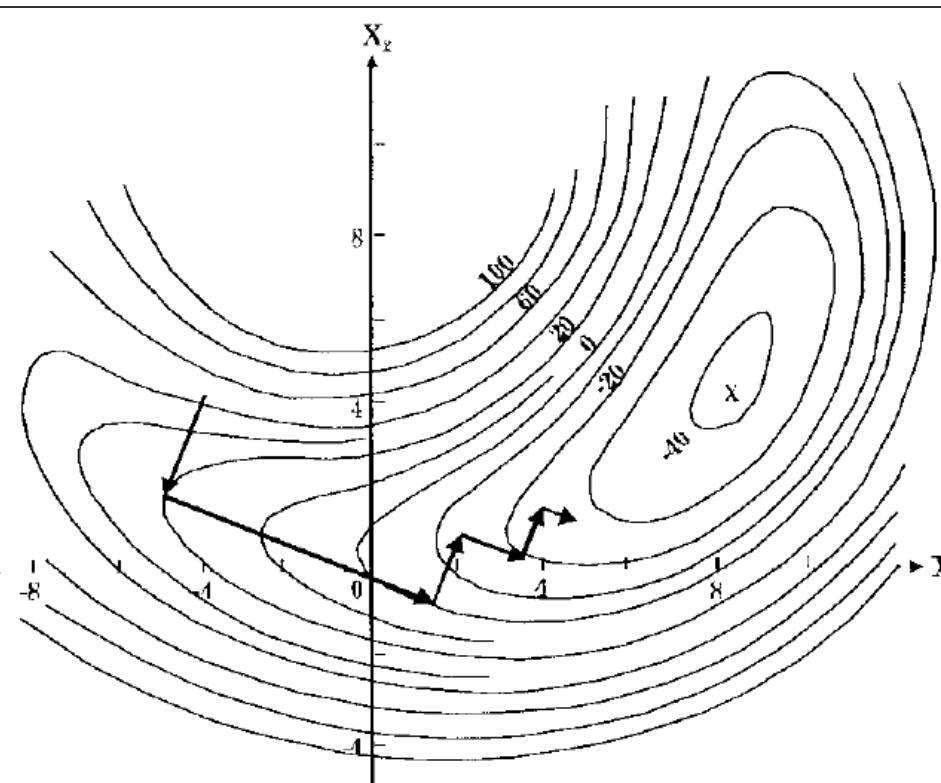
$$\mathbf{S}^q = -\nabla J(\mathbf{x}^{q-1}) + \beta^q \mathbf{S}^{q-1}$$

$$\beta^q = \frac{\left| \nabla J(\mathbf{x}^{q-1}) \right|^2}{\left| \nabla J(\mathbf{x}^{q-2}) \right|^2}$$

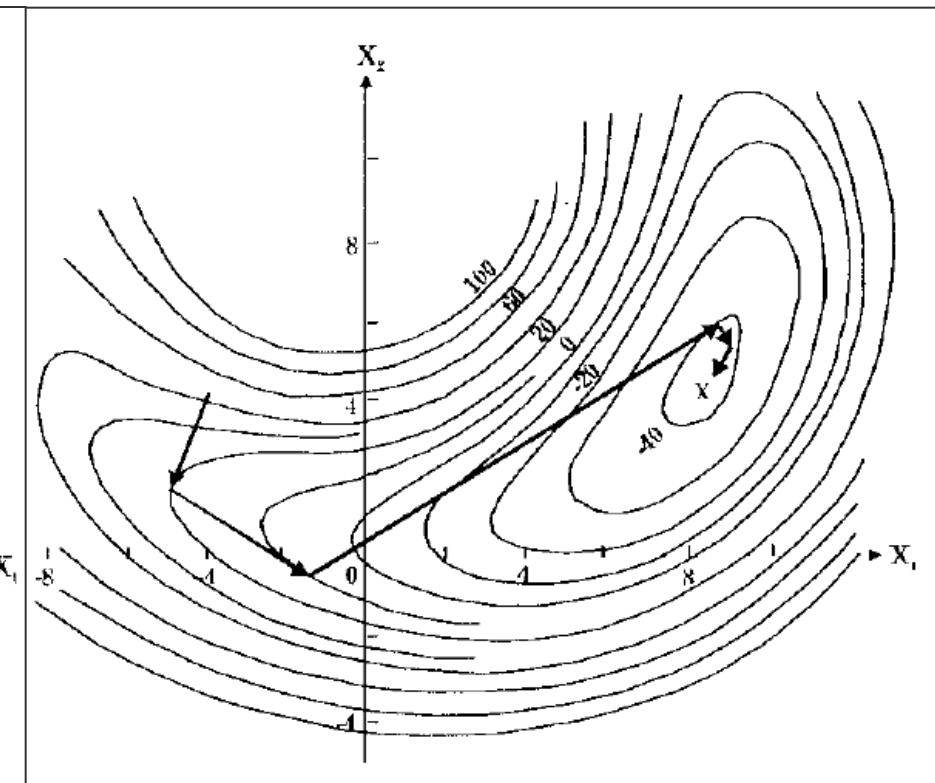
- search directions are now conjugate
- directions \mathbf{S}^j and \mathbf{S}^k are conjugate if $\mathbf{S}^{j\top} \mathbf{H} \mathbf{S}^k = 0$
(also called H-orthogonal)
- makes use of information from previous iterations without having to store a matrix

Geometric Interpretation

Steepest descent



Conjugate gradient



Newton's Method

Taylor series:

$$J(\mathbf{x}) \approx J(\mathbf{x}^0) + \nabla J(\mathbf{x}^0)^T \delta\mathbf{x} + \frac{1}{2} \delta\mathbf{x}^T \mathbf{H}(\mathbf{x}^0) \delta\mathbf{x}$$

where $\delta\mathbf{x} = \mathbf{x} - \mathbf{x}^0$

differentiate: $\nabla J(\mathbf{x}) \approx \nabla J(\mathbf{x}^0) + \mathbf{H}(\mathbf{x}^0) \delta\mathbf{x}$

at optimum $\nabla J(\mathbf{x}^*) = 0$

$$\Rightarrow \nabla J(\mathbf{x}^0) + \mathbf{H}(\mathbf{x}^0) \delta\mathbf{x} = 0$$

$$\delta\mathbf{x} = -[\mathbf{H}(\mathbf{x}^0)]^{-1} \nabla J(\mathbf{x}^0)$$

Newton's Method

$$\mathbf{S} = -[\mathbf{H}(\mathbf{x}^0)]^{-1} \nabla J(\mathbf{x}^0)$$

- if $J(\mathbf{x})$ is quadratic, method gives exact solution in one iteration
- if $J(\mathbf{x})$ not quadratic, perform Taylor series about new point and repeat until converged
- a very efficient technique if started near the solution
- \mathbf{H} is not usually available analytically, and finite difference is too expensive ($n \times n$ matrix)
- \mathbf{H} can be singular if J is linear in a design variable

Quasi Newton

$$\mathbf{S}^q = -\mathbf{A}^q \nabla J(\mathbf{x}^{q-1})$$

- Also known as variable metric methods
- Objective and gradient information is used to create an approximation to the inverse of the Hessian
- \mathbf{A} approaches \mathbf{H}^{-1} during optimization of quadratic functions
- Convergence is similar to second-order methods (strictly 1st order)
- Initially: $\mathbf{A}=\mathbf{I}$, so \mathbf{S}^1 is steepest descent direction
then: $\mathbf{A}^{q+1} = \mathbf{A}^q + \mathbf{D}^q$
where \mathbf{D} is a symmetric update matrix
$$\mathbf{D}^q = \text{fn}(\mathbf{x}^q - \mathbf{x}^{q-1}, \nabla J(\mathbf{x}^q) - \nabla J(\mathbf{x}^{q-1}), \mathbf{A}^q)$$
- Various methods to determine \mathbf{D}
e.g. Davidon-Fletcher-Powell (DFP)
Broydon-Fletcher-Goldfarb-Shanno (BFGS)

One-Dimensional Search (Choosing α)

- Polynomial interpolation
 - pick several values for α
 - fit polynomials to $J(\alpha)$
 - efficient, but need to be careful with implementation
- Golden section search
 - easy to implement, but inefficient
- The one-dimensional search is one of the more challenging aspects of implementing a gradient-based optimization algorithm

Lecture Summary

- Gradient vector and Hessian matrix
- Existence and uniqueness
- Optimality conditions
- Convex spaces
- Unconstrained Methods

The next lecture will focus on gradient-based techniques for nonlinear constrained optimization. We will consider SQP and penalty methods. These are the methods most commonly used for engineering applications.

Lecture 05: Multidisciplinary System Analysis and Design Optimization (MSADO)

Numerical Optimization 2

February 9, 2015

Prof. Douglas Allaire

Today's Topics

- Sequential Linear Programming
- Penalty and Barrier Methods
- Sequential Quadratic Programming

Technique Overview

Steepest Descent
Conjugate Gradient
Quasi-Newton
Newton

UNCONSTRAINED

Simplex – linear
SLP – often not effective
SQP – nonlinear, expensive, common in engineering applications
Exterior Penalty – nonlinear, discontinuous design spaces
Interior Penalty – nonlinear
Generalized Reduced Gradient – nonlinear
Method of Feasible Directions – nonlinear
Mixed Integer Programming

CONSTRAINED

Standard Problem Definition

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

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

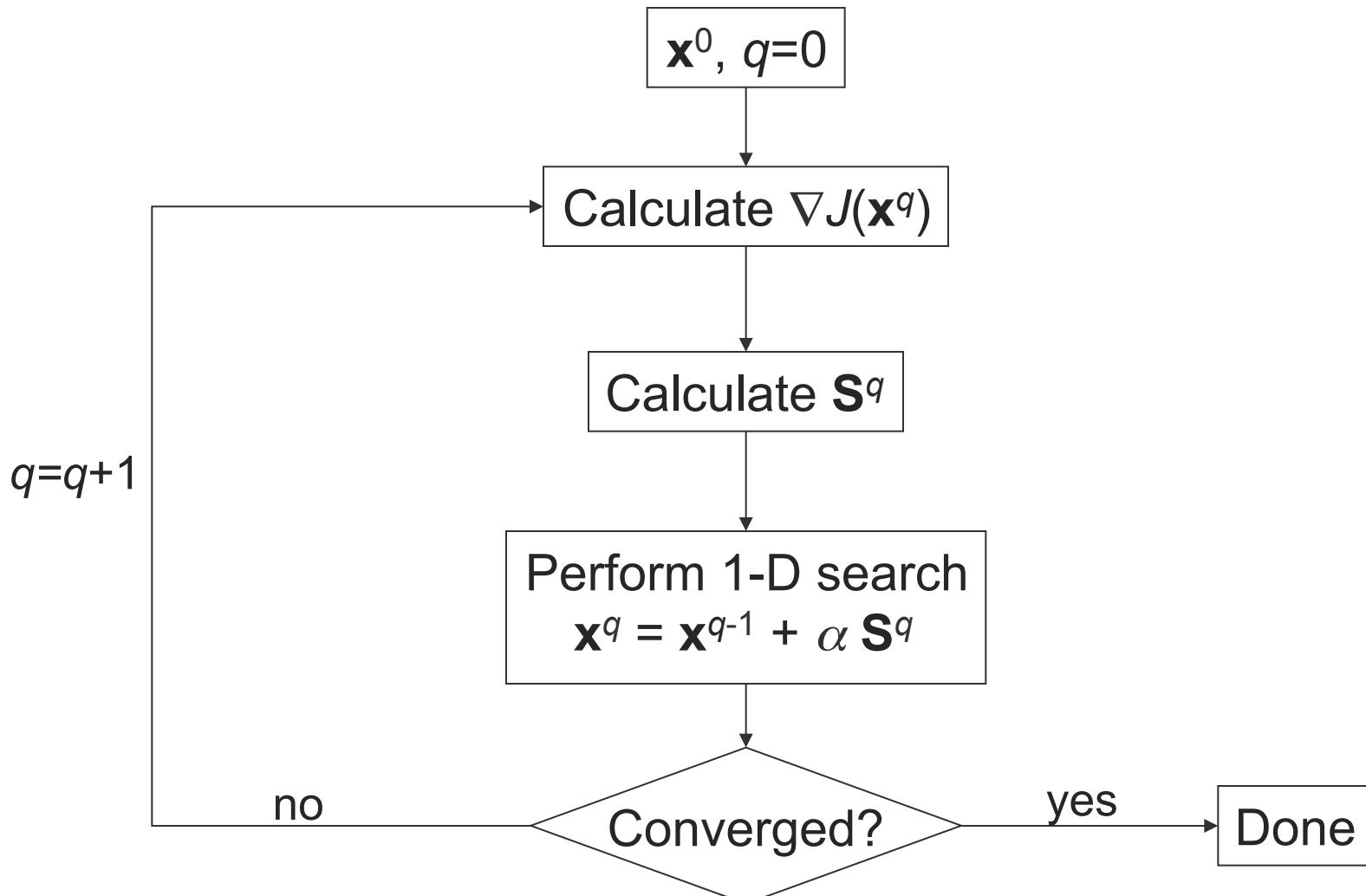
$$h_k(\mathbf{x}) = 0 \quad k = 1, \dots, m_2$$

$$x_i^l \leq x_i \leq x_i^u \quad i = 1, \dots, n$$

For now, we consider a single objective function, $J(\mathbf{x})$. There are n design variables, and a total of m constraints ($m=m_1+m_2$).

For now we assume all x_i are real and continuous.

Optimization Process



Constrained Optimization

Definitions:

- Feasible design: a design that satisfies all constraints
- Infeasible design: a design that violates one or more constraints
- Optimum design: the choice of design variables that minimizes the objective function while satisfying all constraints

In general, constrained optimization algorithms try to cast the problem as an unconstrained optimization and then use one of the techniques we looked at in Lecture 4.

Linear Programming

Most engineering problems of interest are nonlinear

- Can often simplify nonlinear problem by linearization
- LP is often the basis for developing more complicated NLP algorithms

Standard LP problem:

$$\begin{aligned} \min \quad & J(\mathbf{x}) = \sum_{i=1}^n c_i x_i \\ & \sum_{i=1}^n a_{ij} x_i = b_j \quad j = 1, \dots, m \\ & x_i \geq 0 \quad i = 1, \dots, n \end{aligned}$$

$$\begin{aligned} \min \quad & J(\mathbf{x}) = \mathbf{c}^T \mathbf{x} \\ \text{Ax} = \mathbf{b} \\ x_i \geq 0 \quad & i = 1, \dots, n \end{aligned}$$

All LP problems can be converted to this form.

Linear Programming

To convert inequality constraints to equality constraints, use additional design variables:

$$\sum_{i=1}^n a_{ji} x_i \leq b_j \text{ becomes } \sum_{i=1}^n a_{ji} x_i + x_{n+1} = b_j$$

where $x_{n+1} \geq 0$

x_{n+1} is called a **slack variable**

e.g. the constraint

$$x_1 + x_2 \leq 1$$

can be written

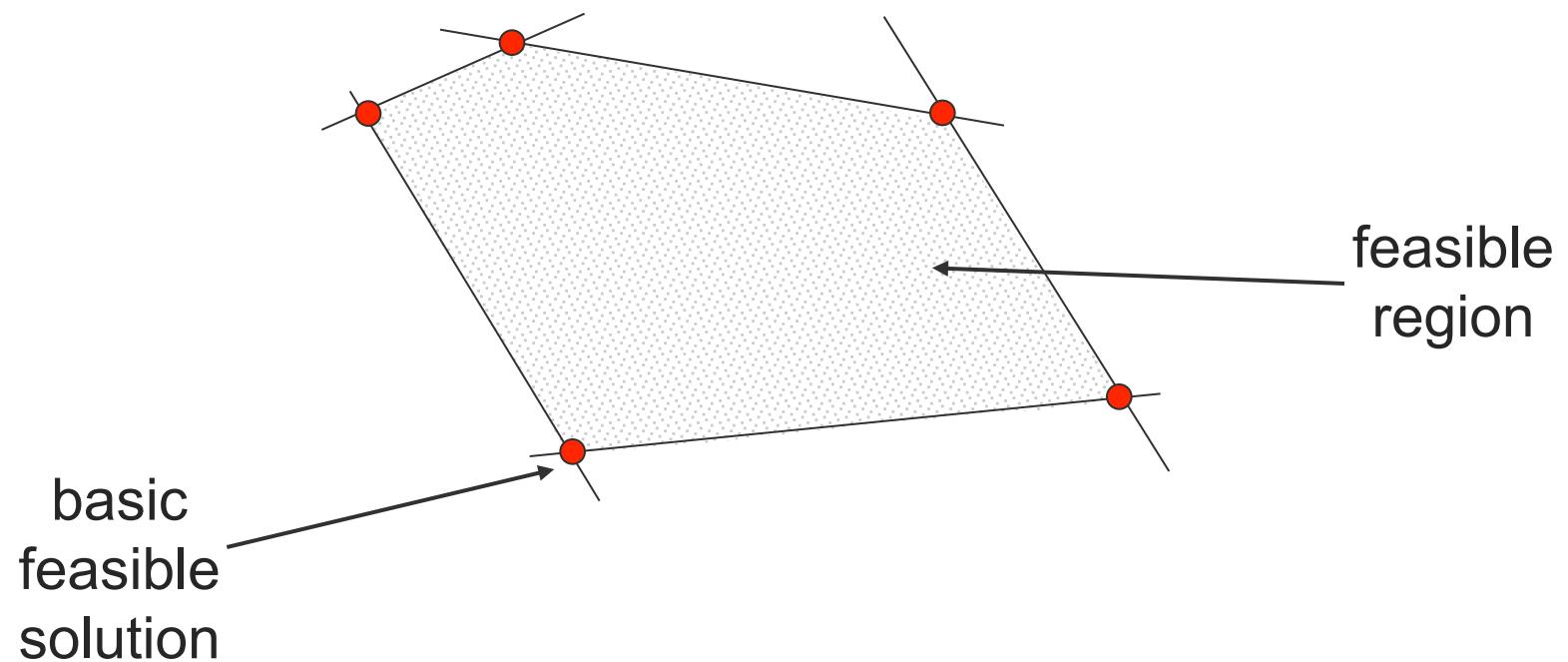
$$\begin{aligned} x_1 + x_2 + x_3 &= 1 \\ x_i &\geq 0, \quad i=1,2,3 \end{aligned}$$

if $x_3=0$, this constraint is active

Simplex Method

Solutions at the “vertices” of the design space are called **basic feasible solutions**.

The Simplex algorithm moves from BFS to BFS so that the objective always improves.



Sequential Linear Programming

Consider a general nonlinear problem linearized via first order Taylor series:

$$\min \quad J(\mathbf{x}) \approx J(\mathbf{x}^0) + \nabla J(\mathbf{x}^0)^T \delta \mathbf{x}$$

$$\text{s.t. } g_j(\mathbf{x}) \approx g_j(\mathbf{x}^0) + \nabla g_j(\mathbf{x}^0)^T \delta \mathbf{x} \leq 0$$

$$h_k(\mathbf{x}) \approx h_k(\mathbf{x}^0) + \nabla h_k(\mathbf{x}^0)^T \delta \mathbf{x} = 0$$

$$x_i^l \leq x_i + \delta x_i \leq x_i^u$$

where $\delta \mathbf{x} = \mathbf{x} - \mathbf{x}^0$

This is an LP problem with the design variables contained in $\delta \mathbf{x}$. The functions and gradients evaluated at \mathbf{x}^0 are constant coefficients.

Sequential Linear Programming

1. Initial guess \mathbf{x}^0
2. Linearize about \mathbf{x}^0 using first-order Taylor series
3. Solve resulting LP to find $\delta\mathbf{x}$
4. Update: $\mathbf{x}^1 = \mathbf{x}^0 + \delta\mathbf{x}$
5. Linearize about \mathbf{x}^1 and repeat:

$$\mathbf{x}^q = \mathbf{x}^{q-1} + \delta\mathbf{x}$$

where $\delta\mathbf{x}$ is the solution of an LP (model linearized about \mathbf{x}^{q-1}).

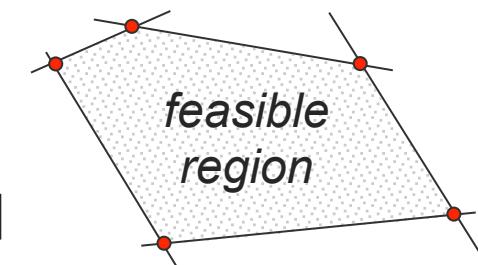
Sequential Linear Programming

- Linearization approximation is only valid close to \mathbf{x}^0
- Need to restrict size of update $\delta\mathbf{x}$
- Not considered to be a good method

Nonlinear vs. Linear Constraints

Linear constraints:

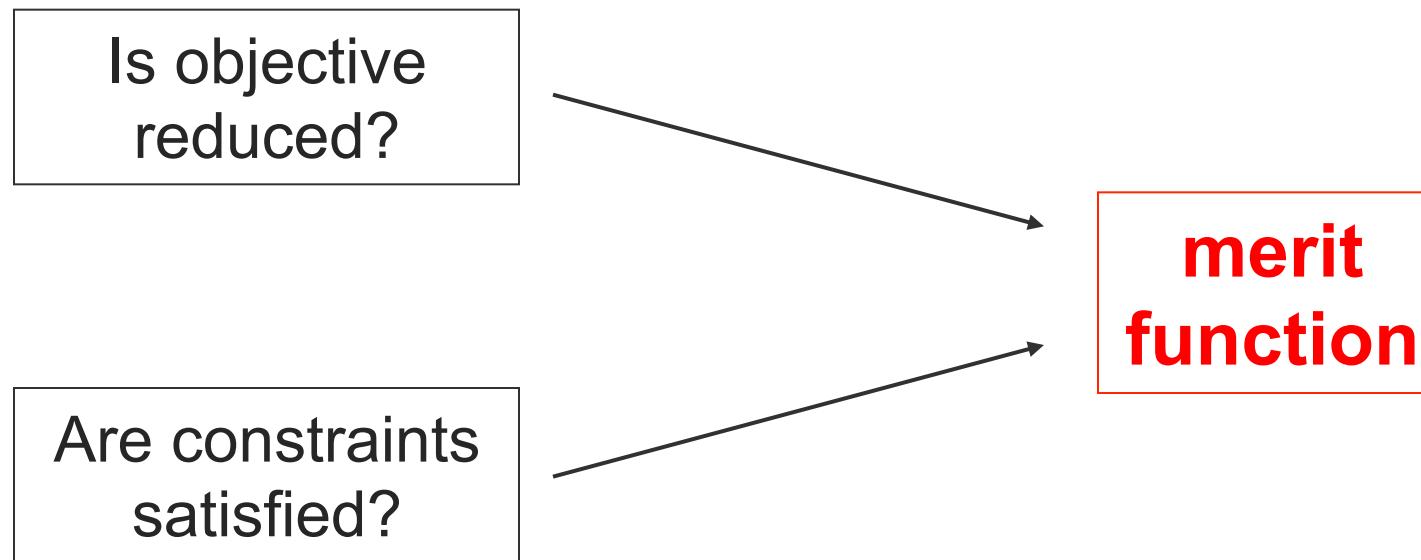
- start from initial feasible point, then all subsequent iterates are feasible
- can construct search direction & step length so that constraints are satisfied
- improvement from \mathbf{x}_k to \mathbf{x}_{k+1} based on $J(\mathbf{x})$



Nonlinear constraints:

- not straightforward to generate a sequence of feasible iterates
- if feasibility is not maintained, then need to decide whether \mathbf{x}_{k+1} is “better” than \mathbf{x}_k

Merit Function



$$\text{merit function} = f(J(\mathbf{x}), \mathbf{g}(\mathbf{x}), \mathbf{h}(\mathbf{x}))$$

examples: penalty function methods, barrier function methods

Subproblems

- Many optimization algorithms get to the optimum by generating and solving a sequence of subproblems that are somehow related to the original problem.
- Typically there are two tasks at each iteration:
 1. Calculate search direction
 2. Calculate the step length
- Sometimes, the initial formulation of a subproblem may be defective
 - i.e. the subproblem has no solution or the solution is unbounded
- A valid subproblem is one for which a solution exists and is well defined
- The option to abandon should be available if the subproblem appears defective
- It is possible that a defective subproblem is an accurate reflection of the original problem having no valid solution

Penalty and Barrier Methods

General Approach:

- minimize objective as unconstrained function
- provide penalty to limit constraint violations
- magnitude of penalty varies throughout optimization
- called sequential unconstrained minimization techniques (SUMT)
- create pseudo-objective:

$$\Phi(\mathbf{x}, r_p) = J(\mathbf{x}) + r_p P(\mathbf{x})$$

$J(\mathbf{x})$ = original objective function

$P(\mathbf{x})$ = imposed penalty function

r_p = scalar multiplier to determine penalty magnitude

p = unconstrained minimization number

Penalty method approaches useful for incorporating constraints into derivative-free and heuristic search algorithms.

Penalty Methods

Four basic methods:

- (i) Exterior Penalty Function Method
- (ii) Interior Penalty Function Method
 - (Barrier Function Method)
- (iii) Log Penalty Function Method
- (iv) Extended Interior Penalty Function Method

Effect of penalty function is to create a local minimum of unconstrained problem “near” \mathbf{x}^* .

Penalty function adds a penalty for infeasibility, barrier function adds a term that prevents iterates from becoming infeasible.

Exterior Penalty Function Method

$$\Phi(\mathbf{x}, \rho_p) = J(\mathbf{x}) + \rho_p P(\mathbf{x})$$

$$P(\mathbf{x}) = \sum_{j=1}^{m_1} \left\{ \max [0, g_j(\mathbf{x})] \right\}^2 + \sum_{k=1}^{m_2} [h_k(\mathbf{x})]^2$$

- if all constraints are satisfied, then $P(\mathbf{x})=0$
- ρ_p = penalty parameter; starts as a small number and increases
- if ρ_p is small, $\Phi(\mathbf{x}, \rho_p)$ is easy to minimize but yields large constraint violations
- if ρ_p is large, constraints are all nearly satisfied but optimization problem is numerically ill-conditioned
- if optimization stops before convergence is reached, the design will be infeasible

Quadratic Penalty Function

$$\Phi_Q(\mathbf{x}, \rho_p) = J(\mathbf{x}) + \rho_p \left(\sum_{j=1}^{\hat{m}_1} [\hat{g}_j(\mathbf{x})]^2 + \sum_{k=1}^{m_2} [h_k(\mathbf{x})]^2 \right)$$

- $\hat{g}_j(\mathbf{x})$ contains those inequality constraints that are violated at \mathbf{x} .
- It can be shown that $\lim_{\rho_p \rightarrow \infty} \mathbf{x}^*(\rho_p) = \mathbf{x}^*$
- $\Phi_Q(\mathbf{x}, \rho_p)$ is well-defined everywhere

Algorithm:

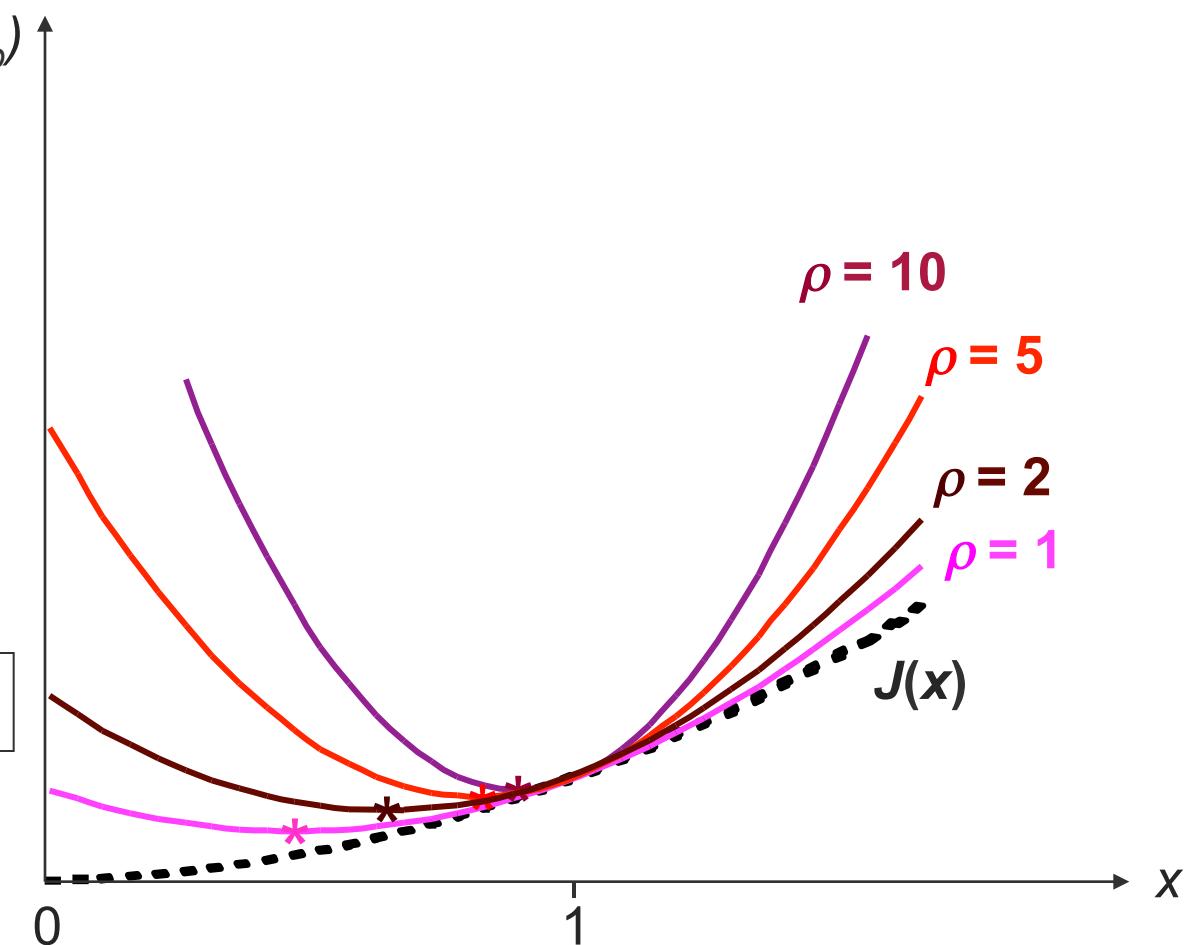
- choose ρ_0 , set $k=0$
- find $\min \Phi_Q(\mathbf{x}, \rho_k) \Rightarrow \mathbf{x}_k^*$
- if not converged, set $\rho_{k+1} > \rho_k$, $k \leftarrow k+1$ and repeat

Note: \mathbf{x}_k^* could be infeasible wrt the original problem

Quadratic Penalty Function Example

$$\begin{aligned} \min J(x) &= x^2 \\ \text{s.t. } x - 1 &= 0 \end{aligned}$$

$$\Phi_Q(x, \rho) = x^2 + \rho(1-x)^2$$



Absolute Value Penalty Function

$$\Phi_A(\mathbf{x}, \rho_p) = J(\mathbf{x}) + \rho_p \left(\sum_{j=1}^{\hat{m}_1} |\hat{g}_j(\mathbf{x})| + \sum_{k=1}^{m_2} |h_k(\mathbf{x})| \right)$$

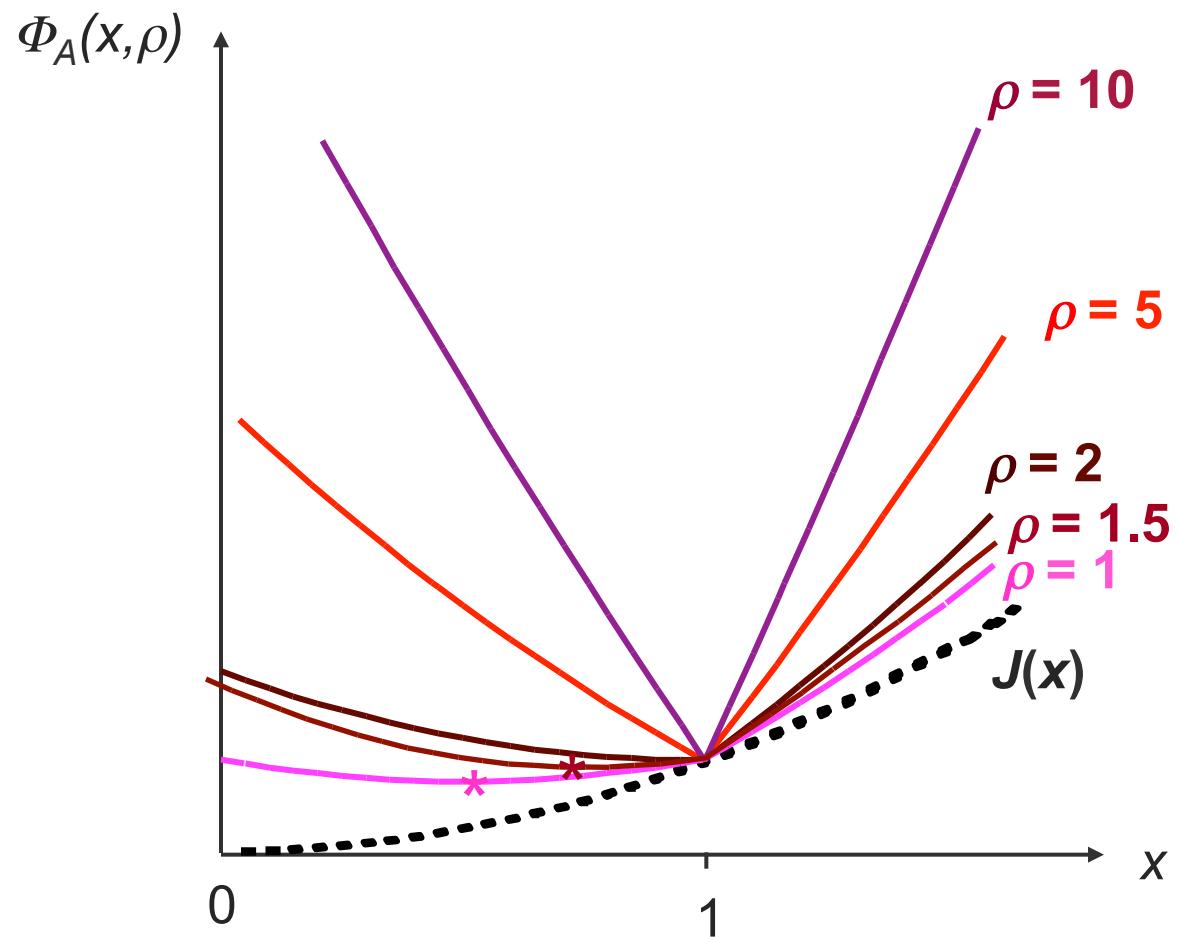
where $\hat{g}_j(\mathbf{x})$ contains those inequality constraints which are violated at \mathbf{x} .

- Φ_A has discontinuous derivatives at points where $\hat{g}_j(\mathbf{x})$ or $h_k(\mathbf{x})$ are zero.
- The crucial difference between Φ_Q and Φ_A is that we do not require $\rho_p \rightarrow \infty$ for \mathbf{x}^* to be a minimum of Φ_A , so we can avoid ill-conditioning
- Instead, for some threshold $\bar{\rho}_p$, \mathbf{x}^* is a minimum of Φ_A for any $\rho_p > \bar{\rho}_p$
- Sometimes called exact penalty functions

Absolute Value Penalty Function Example

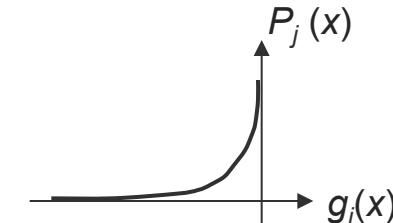
$$\begin{aligned} \min J(x) &= x^2 \\ \text{s.t. } x - 1 &= 0 \end{aligned}$$

$$\Phi_A(x, \rho) = x^2 + \rho |1-x|$$



Interior Penalty Function Method (Barrier Function Method)

$$P(\mathbf{x}) = \sum_{j=1}^{\hat{m}_1} \frac{-1}{\hat{g}_j(\mathbf{x})}$$



$$\Phi(\mathbf{x}, r_p, \rho_p) = J(\mathbf{x}) + r_p \sum_{j=1}^{\hat{m}_1} \frac{-1}{\hat{g}_j(\mathbf{x})} + \rho_p \sum_{k=1}^{m_2} [h_k(\mathbf{x})]^2$$

- r_p = barrier parameter; starts as a large positive number and decreases
- barrier function for inequality constraints only
- sequence of improving feasible designs
- $\Phi(\mathbf{x}, r_p, \rho_p)$ discontinuous at constraint boundaries

Barrier Function Method

$$P(\mathbf{x}) = \sum_{j=1}^{m_1} -\ln[-g_j(\mathbf{x})]$$

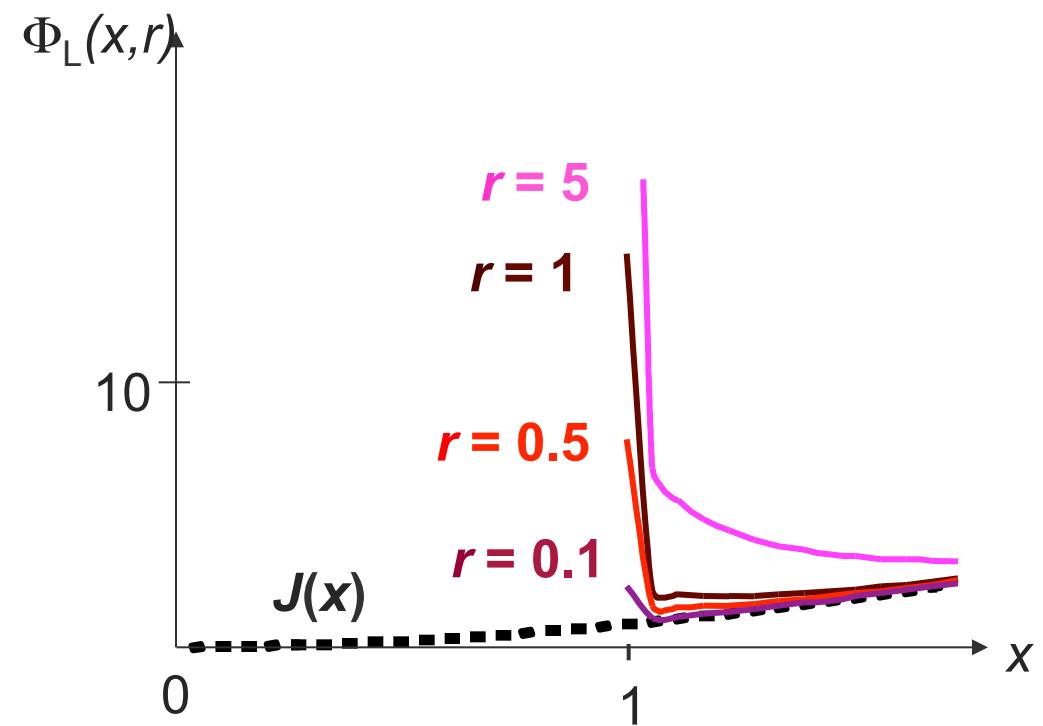
$$\Phi_L(\mathbf{x}, r_p, \rho_p) = J(\mathbf{x}) - r_p \sum_{j=1}^{\hat{m}_1} \ln(-\hat{g}_j(\mathbf{x}))$$

- often better numerically conditioned than $\frac{-1}{g_j(\mathbf{x})}$
- penalty function has a positive singularity at the boundary of the feasible region
- penalty function is undefined for $g_i(\mathbf{x}) > 0$
- $\lim_{r_p \rightarrow 0} \mathbf{x}^*(r_p) = \mathbf{x}^*$

Barrier Function Example

$$\begin{aligned} \min J(x) &= x^2 \\ \text{s.t. } x - 1 &\geq 0 \end{aligned}$$

$$\Phi_L(x, r) = x^2 - r \ln(x-1)$$



Extended Interior Penalty Function Method

Combine features of interior/exterior methods

$$P(\mathbf{x}) = \sum_{j=1}^{m_1} \tilde{g}_j(\mathbf{x})$$

-linear extended
penalty function

where $\tilde{g}_j(\mathbf{x}) = \begin{cases} -\frac{1}{g_j(\mathbf{x})} & \text{if } g_j(\mathbf{x}) \leq \varepsilon \\ -\frac{2\varepsilon - g_j(\mathbf{x})}{\varepsilon^2} & \text{if } g_j(\mathbf{x}) > \varepsilon \end{cases}$

$-\varepsilon$ is a small negative number, and marks transition from interior penalty to extended penalty

$-\varepsilon$ must be chosen so that Φ has positive slope at constraint boundary

-can also use quadratic extended and variable penalty functions

Sequential Quadratic Programming

- Create a quadratic approximation to the Lagrangian
- Create linear approximations to the constraints
- Solve the quadratic problem to find the search direction, \mathbf{S}
- Perform the 1-D search
- Update the approximation to the Lagrangian

Sequential Quadratic Programming

Create a subproblem
with **quadratic objective function:**

$$\min Q(\mathbf{S}^k) = J(\mathbf{x}^k) + \nabla J(\mathbf{x}^k)^T \mathbf{S}^k + \frac{1}{2} \mathbf{S}^T \mathbf{B} \mathbf{S}$$

and **linear constraints**

$$\text{s.t. } \nabla g_j(\mathbf{x}^k)^T \mathbf{S}^k + g_j(\mathbf{x}^k) \leq 0 \quad j = 1, \dots, m_1$$

$$\nabla h_k(\mathbf{x}^k)^T \mathbf{S}^k + h_k(\mathbf{x}^k) = 0 \quad k = 1, \dots, m_2$$

- Design variables are the components of \mathbf{S}
- $\mathbf{B}=\mathbf{I}$ initially, then is updated to approximate \mathbf{H} (as in quasi-Newton)

Incompatible Constraints

- The constraints of the subproblem can be incompatible, even if the original problem has a well-posed solution
- For example, two linearized constraints could be linearly dependent
- This is a common occurrence in practice
- Likelihood of incompatible constraints reduced by allowing flexibility in RHS (e.g. allow scaling factors in front of $g_j(\mathbf{x}^k)$ term)

$$\nabla g_j(\mathbf{x}^k)^\top \mathbf{S}^k + \gamma_j g_j(\mathbf{x}^k) \leq 0 \quad j = 1, \dots, m_1$$

- Typically $\gamma=0.9$ if constraint is violated and $\gamma=1$ otherwise
- Doesn't affect convergence, since specific form of constraint is only crucial when \mathbf{x}^k is close to \mathbf{x}^*

Sequential Quadratic Programming

- Widely used in engineering applications e.g. NLOpt
- Considered to be one of the best gradient-based algorithms
- Fast convergence for many problems
- Strong theoretical basis
- Matlab: fmincon (medium scale)

Lecture Summary

We have seen the following nonlinear techniques:

- Penalty and Barrier Methods
- Sequential Quadratic Programming

It is important to understand

- when it is appropriate to use these methods
- the basics of how the method works
- why the method might fail
- what your results mean (numerically vs. physically)

References

Practical Optimization, P.E. Gill, W. Murray and M.H. Wright, Academic Press, 1986.

Numerical Optimization Techniques for Engineering Design, G.N. Vanderplaats, Vanderplaats R&D, 1999.

Optimal Design in Multidisciplinary Systems, AIAA Professional Development Short Course Notes, September 2002.

NLOPT: Open-source library for nonlinear optimization
http://ab-initio.mit.edu/wiki/index.php/NLopt_Algorithms

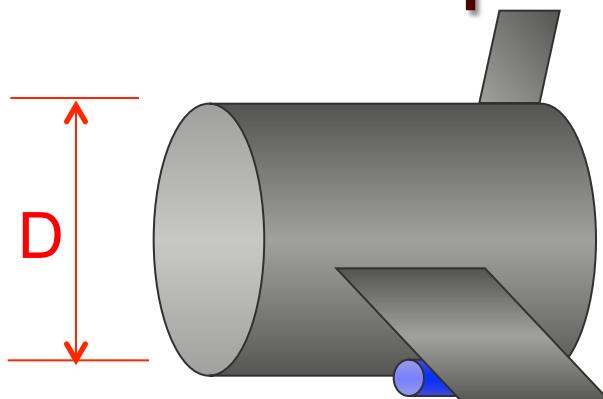
Lecture 06: Multidisciplinary System Analysis and Design Optimization (MSADO)

Decomposition and Coupling

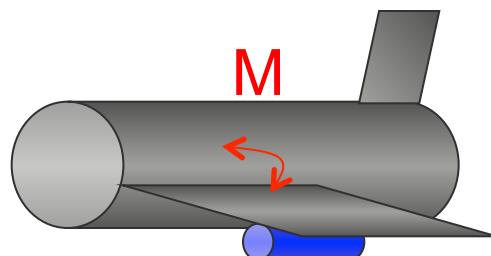
February 11, 2015

Prof. Douglas Allaire

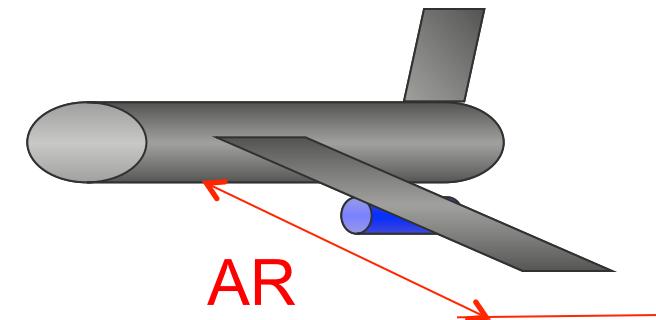
Aircraft “Optimization”



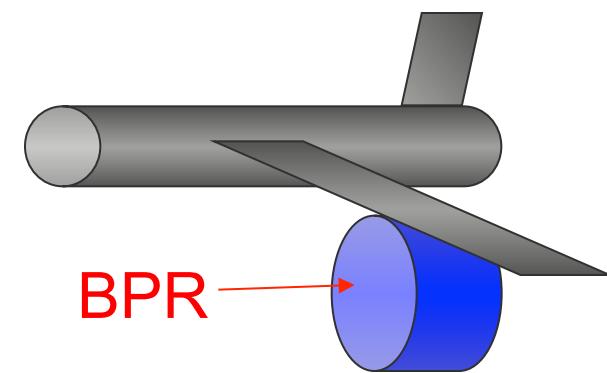
Marketing: maximize passenger volume
→ Cabin diameter



Structures: minimize structural mass
→ Wing-root moment



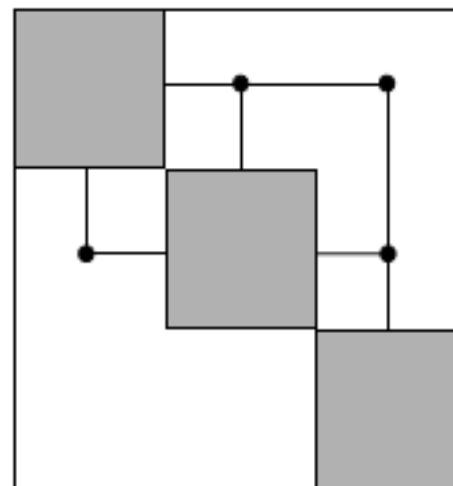
Aero: maximize L/D
→ Aspect Ratio



Propulsion: minimize specific fuel consumption (SFC) → Bypass Ratio

Today's Topics

- In Lecture 3 we discussed the standard approach: Sequential modular analysis



Modules are executed sequentially with or without feedback loops.

- MDO frameworks
 - Distributed analysis
 - Distributed design

Other Approaches:

Fundamentally Different Approaches in MDO

- Distributed Analysis
 - Disciplinary models provide analysis
 - All optimization done at system level
- Distributed Design
 - Provide disciplinary models with design tasks
 - Optimization at subsystem and system levels

Concurrent subspace optimization, Collaborative optimization, Bi-level integrated system synthesis, ...and many more

Standard Optimization Problem

Given

$$x \in \Re^n$$

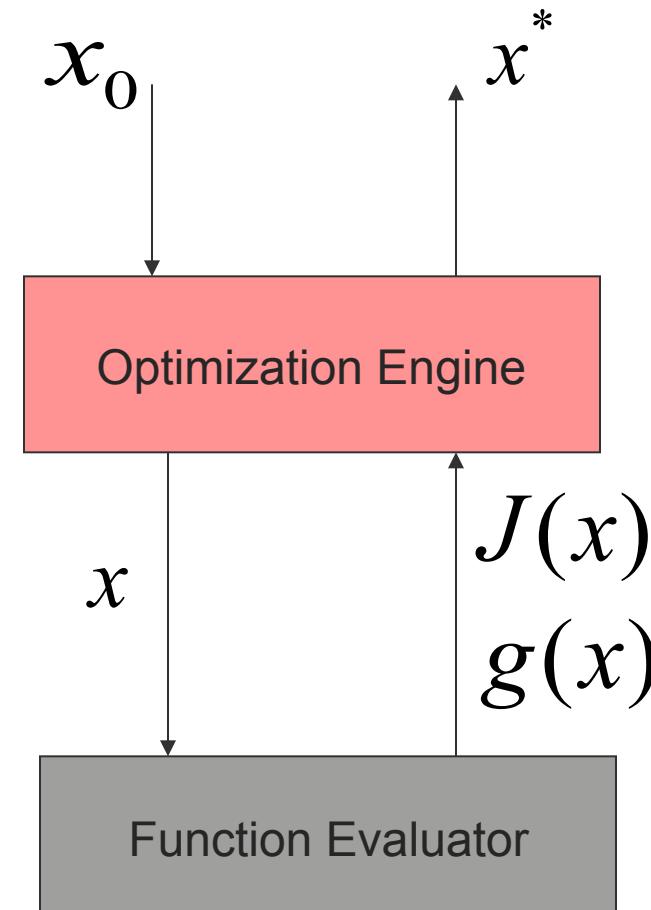
$$J : \Re^n \rightarrow \Re^z$$

$$g : \Re^n \rightarrow \Re^m$$

Solve the problem

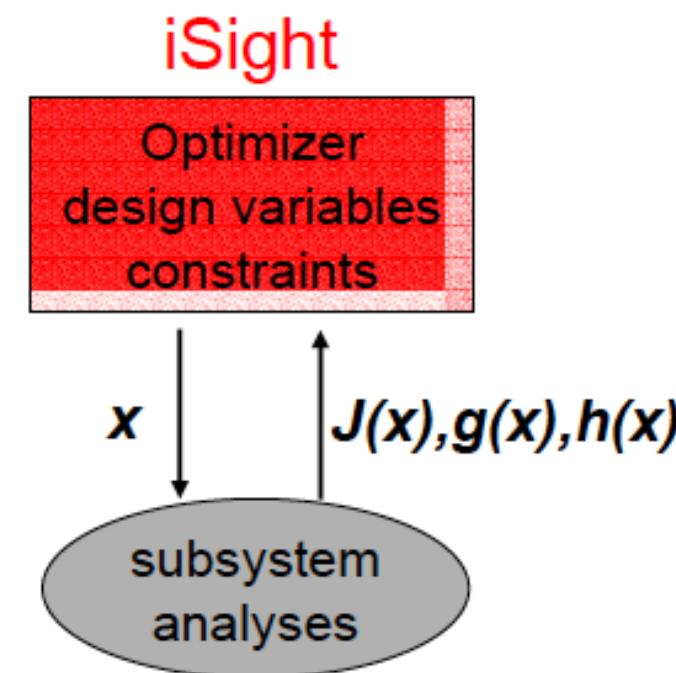
$$\min J(x)$$

$$\text{s.t. } g(x) \geq 0$$

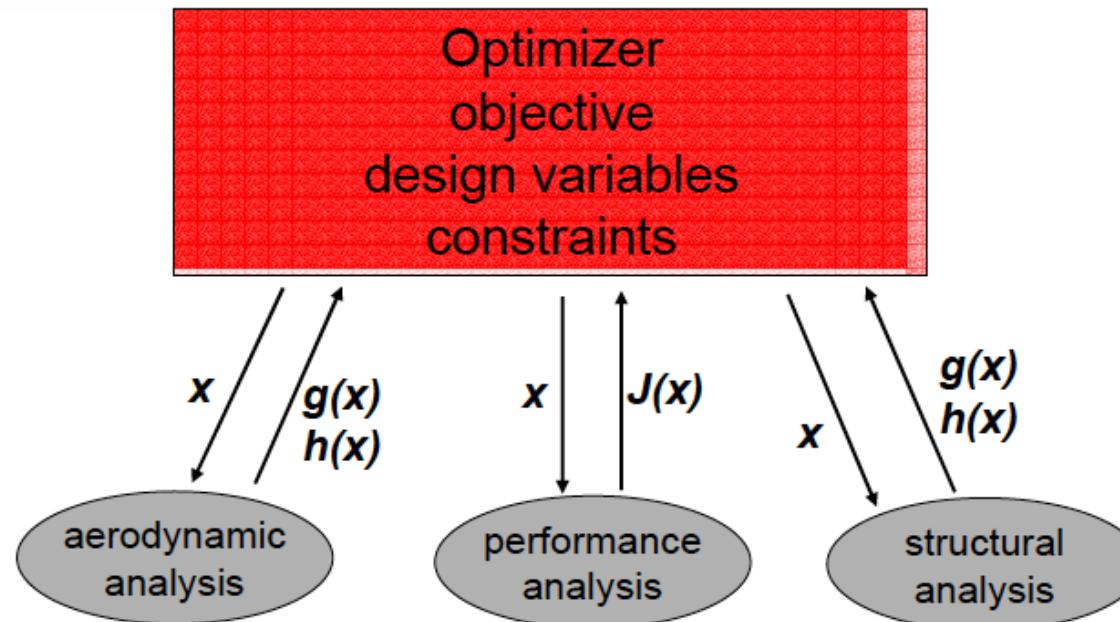


Distributed Analysis

- Disciplinary models provide analysis
- Optimization is controlled by some overseeing code or database
 - E.g., ISight

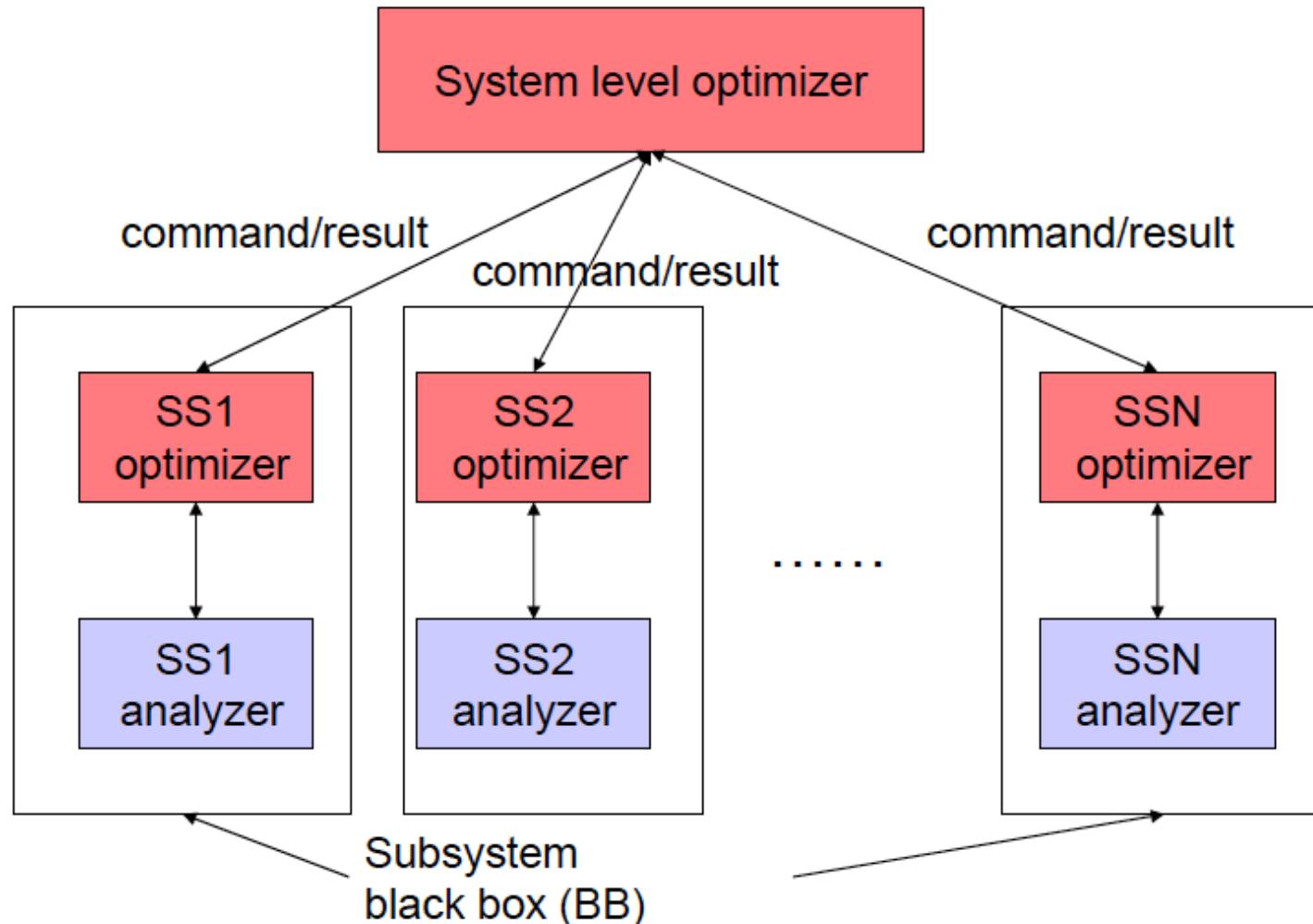


Distributed Analysis



- During the optimization, the overseeing code keeps track of the values of the design variables and objectives
- The values of the design variables are changed according to the optimization algorithm
- Disciplinary models are asked to evaluate constraints/objective

Distributed Design

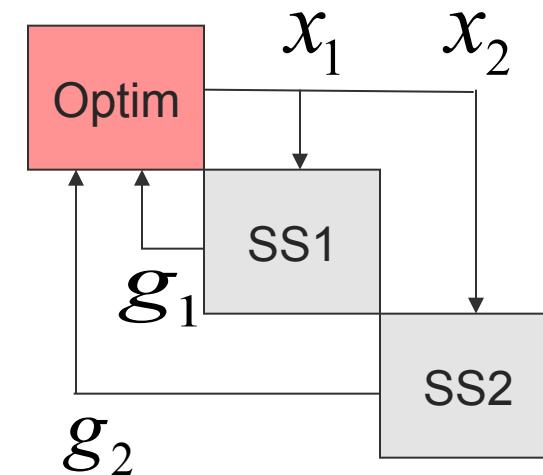
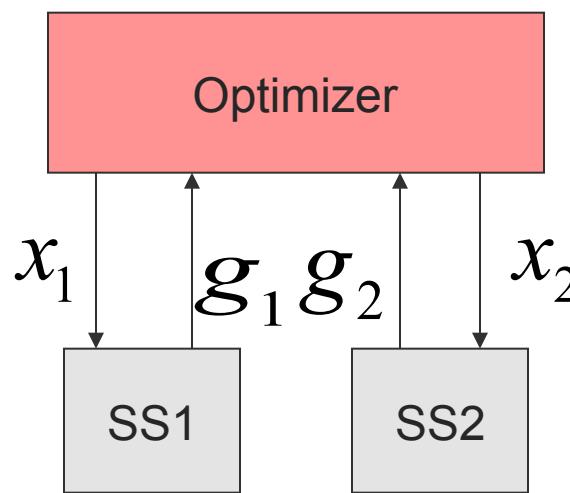


Advantages of Decoupling

Computation of $g(x)$ can be very time consuming, want to divide the work and compute in parallel.

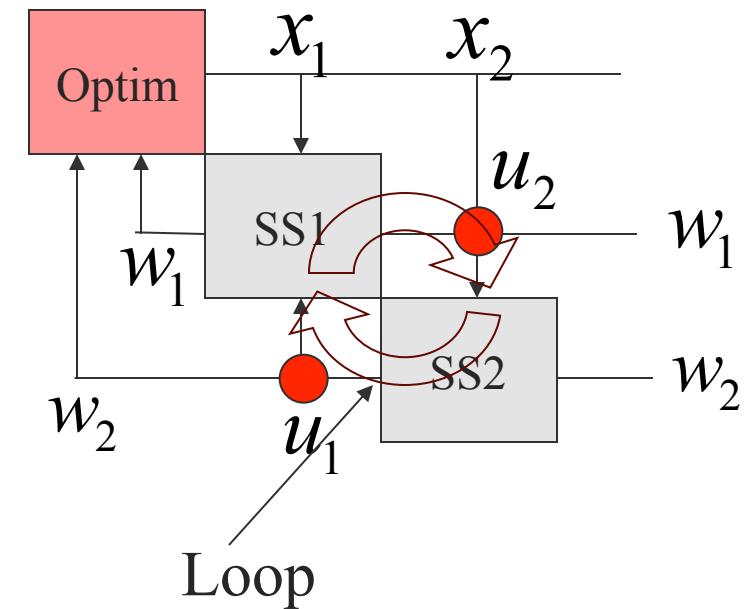
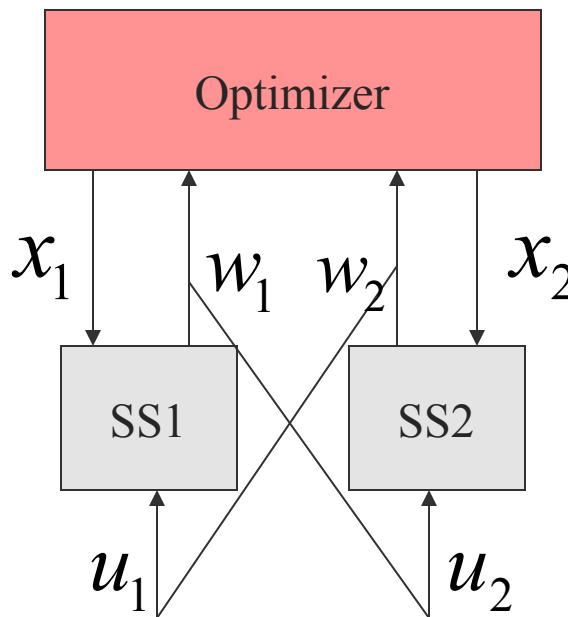
For example, if $x = (x_1, x_2)$, where $x_1 \in \mathbb{R}^{n^1}, x_2 \in \mathbb{R}^{n^2}$
and $g(x) = (g_1(x_1), g_2(x_2))$

Then g_1 and g_2 can be computed in parallel. Graphically,



Coupled Situation

The decoupled constraints assumption is not general. Subsystems can be coupled and loops can arise. For example,



x: decision variables

w: SS outputs (constraint, cost)

u: SS input (dependent)

vline: SS input
hline: SS output

Computation of w_1 and w_2 requires an iterative method.

Coupling

- An example where such a loop happens is as follows:

$$\min J(x_1, x_2)$$

$$\begin{aligned} \text{s.t. } w_1 &= g_1(x_1, g_2(x_2, w_1)) \geq 0 \\ w_2 &= g_2(x_2, g_1(x_1, w_2)) \geq 0 \end{aligned}$$

where $x_1 \in \mathbb{R}^{n^1}, x_2 \in \mathbb{R}^{n^2}, g_i : x_i \times u_i \mapsto w_i, i = 1, 2$

- w_1 and w_2 satisfy coupled relations at each optimization iteration. At each constraint evaluation, nonlinear equations must be solved (e.g. by Newton's method) in order to obtain w_1 and w_2 , which can be time consuming.

Want a way to return to the situation of decoupled constraints.

Surrogate Variables (“Tearing”)

Information loop can be broken by introducing surrogate variables.

$$\min J(x_1, x_2)$$

$$\text{s.t. } \begin{aligned} w_1 &= g_1(x_1, g_2(x_2, w_1)) \geq 0 \\ w_2 &= g_2(x_2, g_1(x_1, w_2)) \geq 0 \end{aligned}$$



$$\min J(x_1, x_2)$$

s.t.

$$g_1(x_1, u_1) \geq 0$$

$$g_2(x_2, u_2) \geq 0$$

$$u_2 - g_1(x_1, u_1) = 0$$

$$u_1 - g_2(x_2, u_2) = 0$$

- u_1 and u_2 are **decision variables** acting as the inputs to $g_1(\text{SS1})$ and g_2 (SS2). Introducing surrogate variables breaks information loop but **increases the number of decision variables**.

Numerical Example

$$\min J_1 + J_2$$

$$\text{s.t. } w_1 \geq 0$$

$$w_2 \geq 0$$

decoupled



$$\text{where } J_1 = x_1^2 + x_2^2$$

$$J_2 = (x_3 - 3)^2 + (x_4 - 4)^2$$

$$w_1 = x_1^3 - x_2^3 + 2w_2$$

$$w_2 = x_3^3 - x_4^3 + 2w_1$$



coupled

$$\min x_1^2 + x_2^2 + (x_3 - 3)^2 + (x_4 - 4)^2$$

$$\text{s.t. } w_1 = g_1(x_1, x_2, x_3, x_4) \geq 0$$

$$w_2 = g_2(x_1, x_2, x_3, x_4) \geq 0$$

$$\min x_1^2 + x_2^2 + (x_3 - 3)^2 + (x_4 - 4)^2$$

$$\text{s.t. } w_1 = x_1^3 - x_2^3 + 2x_5 \geq 0$$

$$w_2 = x_3^3 - x_4^3 + 2x_6 \geq 0$$

$$x_1^3 - x_2^3 + 2x_5 - x_6 = 0$$

$$x_3^3 - x_4^3 + 2x_6 - x_5 = 0$$

Solution:

$$x = (0, 0, 3, 4; 24.67, 12.33)$$

coupled: 0.25s

uncoupled: 0.03s

Distributed Design Methods

- Disciplinary models are provided with design tasks
- Optimization is performed at a subsystem level in addition to the system level

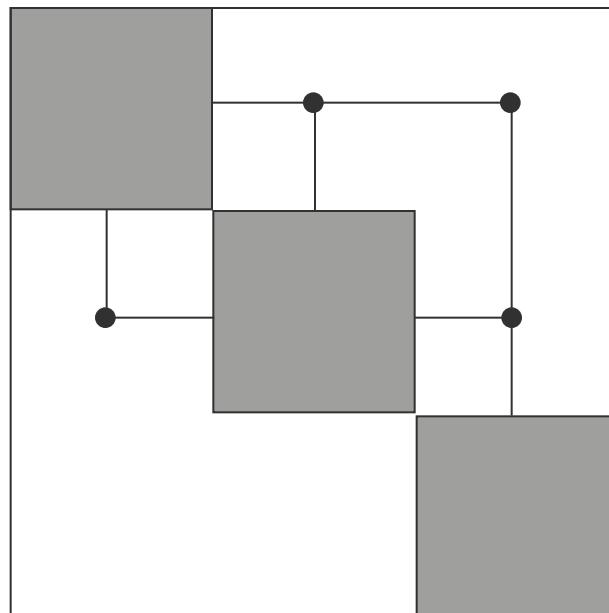
Concurrent Subspace Optimization (CSO)

- Divide the design problem into several discipline-related subspaces
- Each subspace shares responsibility for satisfying constraints while trying to reduce a global objective

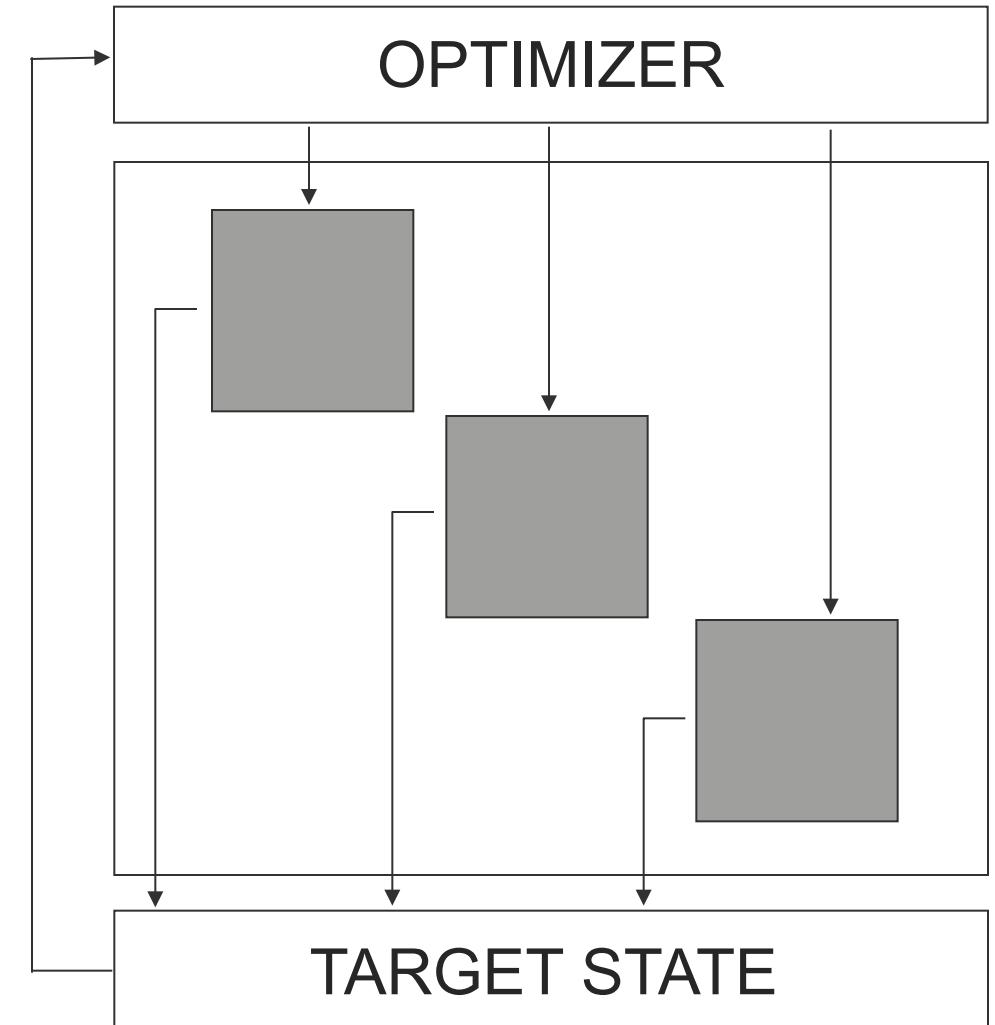
Collaborative Optimization

- Disciplinary teams satisfy local constraints while trying to match target values specified by a system coordinator
- Preserves disciplinary-level design freedom

Collaborative Optimization



Coupled



Uncoupled

Collaborative Optimization

Two levels of optimization:

- A system-level optimizer provides a set of targets.
 - These targets are chosen to optimize the system-level objective function
- A subsystem optimizer finds a design that minimizes the difference between current states and the targets.
 - Subject to local constraints

Collaborative Optimization

$$\min J_{\text{sys}}$$

wrt: $\mathbf{x}_0 = \{\text{target variables}\}$

s.t. $J_k = 0 \quad \forall \text{ subproblems}_k$

$$\begin{array}{c} \{\mathbf{x}_0\} \\ \xrightarrow{J_{\text{sys}}} \end{array}$$

performance analysis

$$\begin{array}{c} \{\mathbf{x}_0\} \\ \nearrow \\ \searrow \\ J_1 \end{array}$$

$$\begin{array}{c} \{\mathbf{x}_0\} \\ \nearrow \\ \searrow \\ J_k \end{array}$$

$$\min J_1 = \left\{ \begin{array}{l} \text{target variables} - \text{local variables} \end{array} \right\}^2$$

$\mathbf{x} = \{\text{local variables}\}$

s.t. $\{\text{local constraints}\}$

$$\begin{array}{c} \{\mathbf{x}\} \\ \downarrow \\ \uparrow \text{computed results} \end{array}$$

analysis for subsystem 1

$$\min J_k = \left\{ \begin{array}{l} \text{target variables} - \text{local variables} \end{array} \right\}^2$$

$\mathbf{x} = \{\text{local variables}\}$

s.t. $\{\text{local constraints}\}$

$$\begin{array}{c} \{\mathbf{x}\} \\ \downarrow \\ \uparrow \text{computed results} \end{array}$$

analysis for subsystem k

CO – Subsystem Level

$$\begin{aligned}\min J_1 &= \left\{ \frac{\text{target variables} - \text{local variables}}{\text{variables}} \right\}^2 \\ \mathbf{x} &= \{\text{local variables}\} \\ \text{s.t.} & \quad \{\text{local constraints}\}\end{aligned}$$

- The subsystem optimizer modifies local variables to achieve the best design for which the set of local variables and computed results most nearly matches the system targets
- The local constraints must also be satisfied

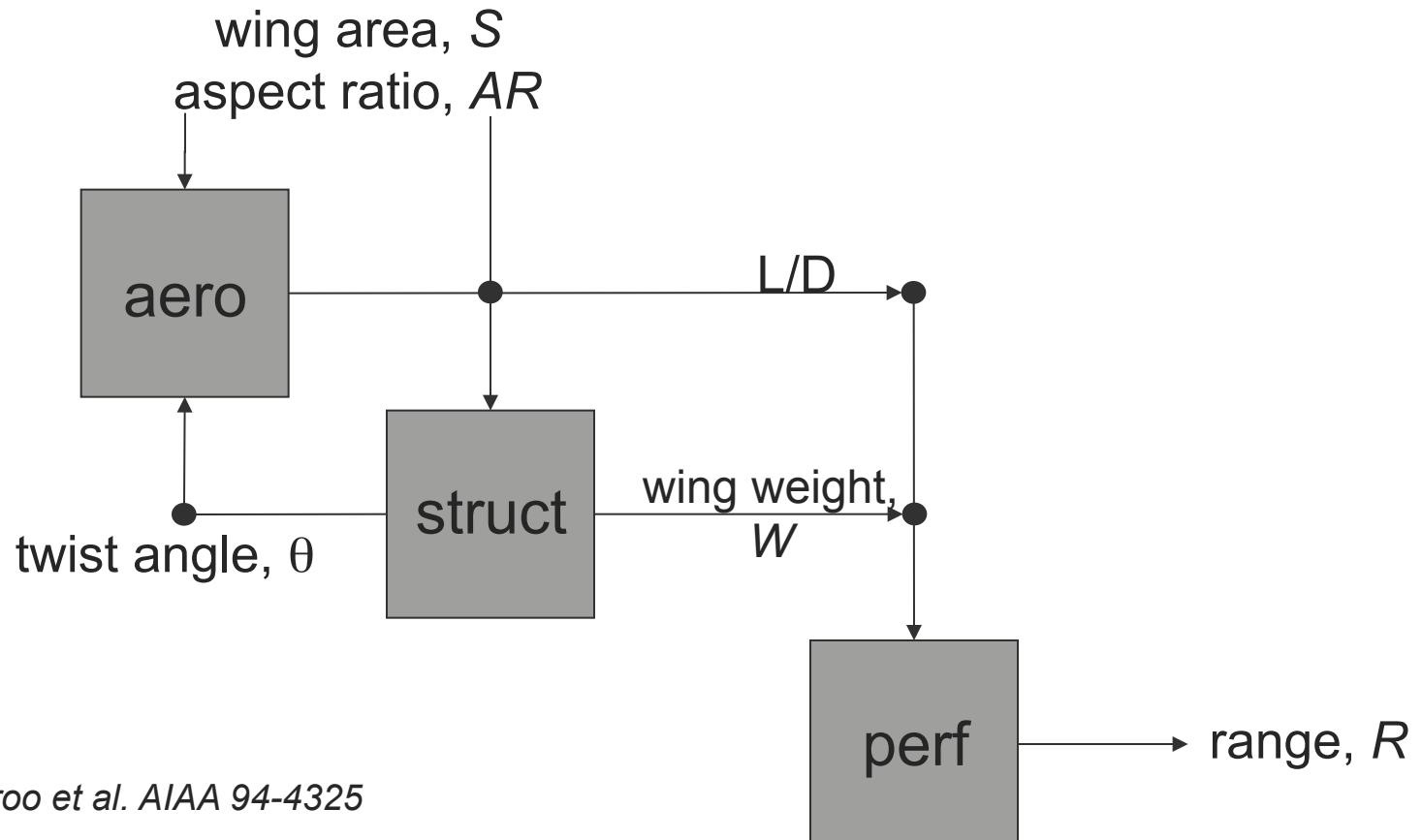
CO – System Level

$$\begin{aligned} & \min J_{\text{sys}} \\ \text{wrt: } & \mathbf{x}_0 = \{\text{target variables}\} \\ \text{s.t. } & J_k = 0 \quad \forall \text{ subproblems}_k \end{aligned}$$

- System-level optimizer changes target variables to improve objective and reduce differences J_k
 - $J_k=0$ are called compatibility constraints
 - compatibility constraints are driven to zero, but may be violated during the optimization

CO Example: Aircraft Design

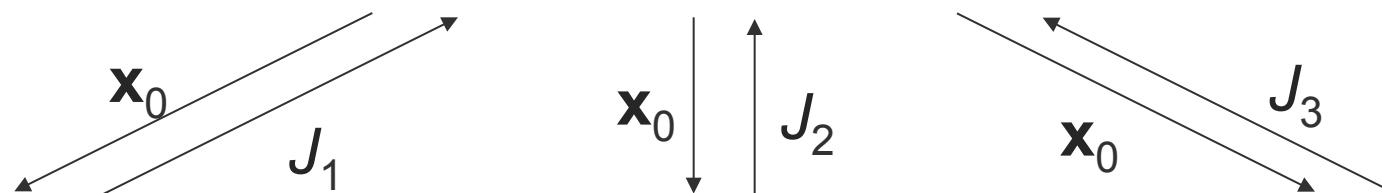
Consider a simple aircraft design problem:
maximize range for a given take-off weight by choosing
wing area, aspect ratio, twist angle, L/D, and wing weight.



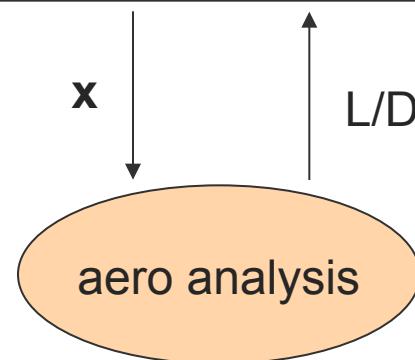
modified from Kroo et al. AIAA 94-4325

CO Example: Aircraft Design

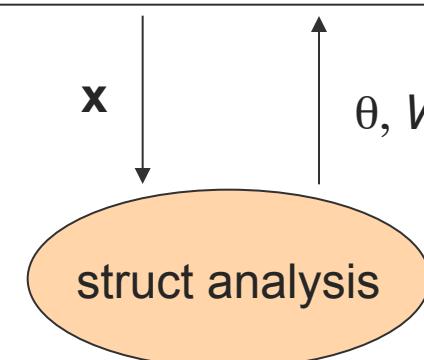
$$\begin{aligned} & \max R_0 \\ & \mathbf{x}_0 = [R_0 \ S_0 \ AR_0 \ \theta_0 \ L/D_0 \ W_0]^T \\ & \text{s.t. } J_1=0, J_2=0, J_3=0 \end{aligned}$$



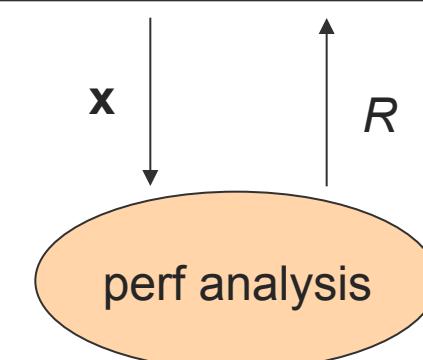
$$\begin{aligned} & \min J_1 \\ & J_1 = (S-S_0)^2 + (AR-AR_0)^2 + \\ & \quad (\theta-\theta_0)^2 + (L/D-L/D_0)^2 \\ & \mathbf{x} = [AR \ \theta]^T \end{aligned}$$



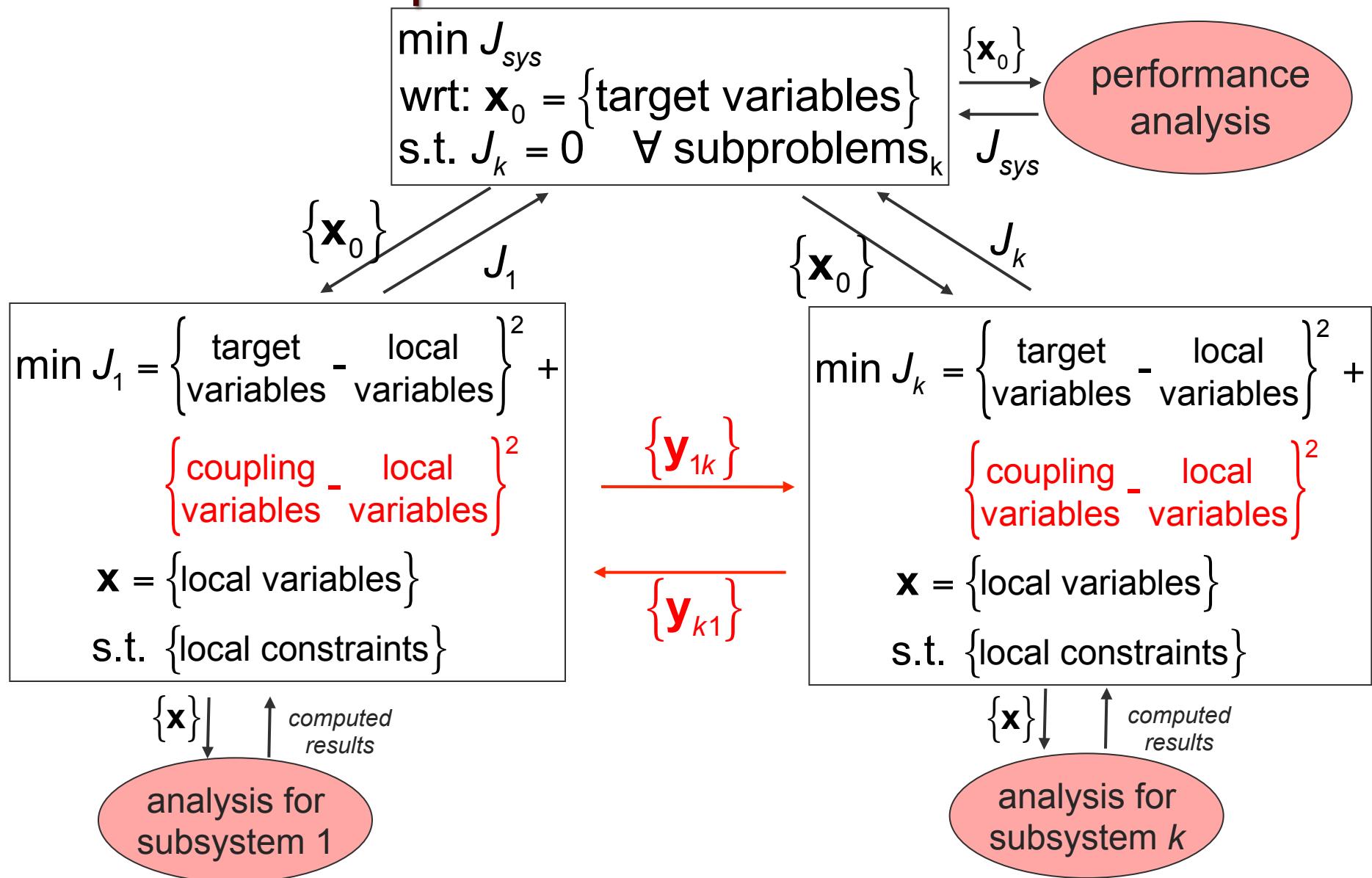
$$\begin{aligned} & \min J_2 \\ & J_2 = (S-S_0)^2 + (AR-AR_0)^2 + \\ & \quad (\theta-\theta_0)^2 + (W-W_0)^2 \\ & \mathbf{x} = [S \ AR]^T \end{aligned}$$



$$\begin{aligned} & \min J_3 \\ & J_3 = (L/D-L/D_0)^2 + (W-W_0)^2 \\ & \quad + (R-R_0)^2 \\ & \mathbf{x} = [L/D \ W]^T \end{aligned}$$



Collaborative Optimization



Collaborative Optimization

\mathbf{x}_0 = system-level target variable values

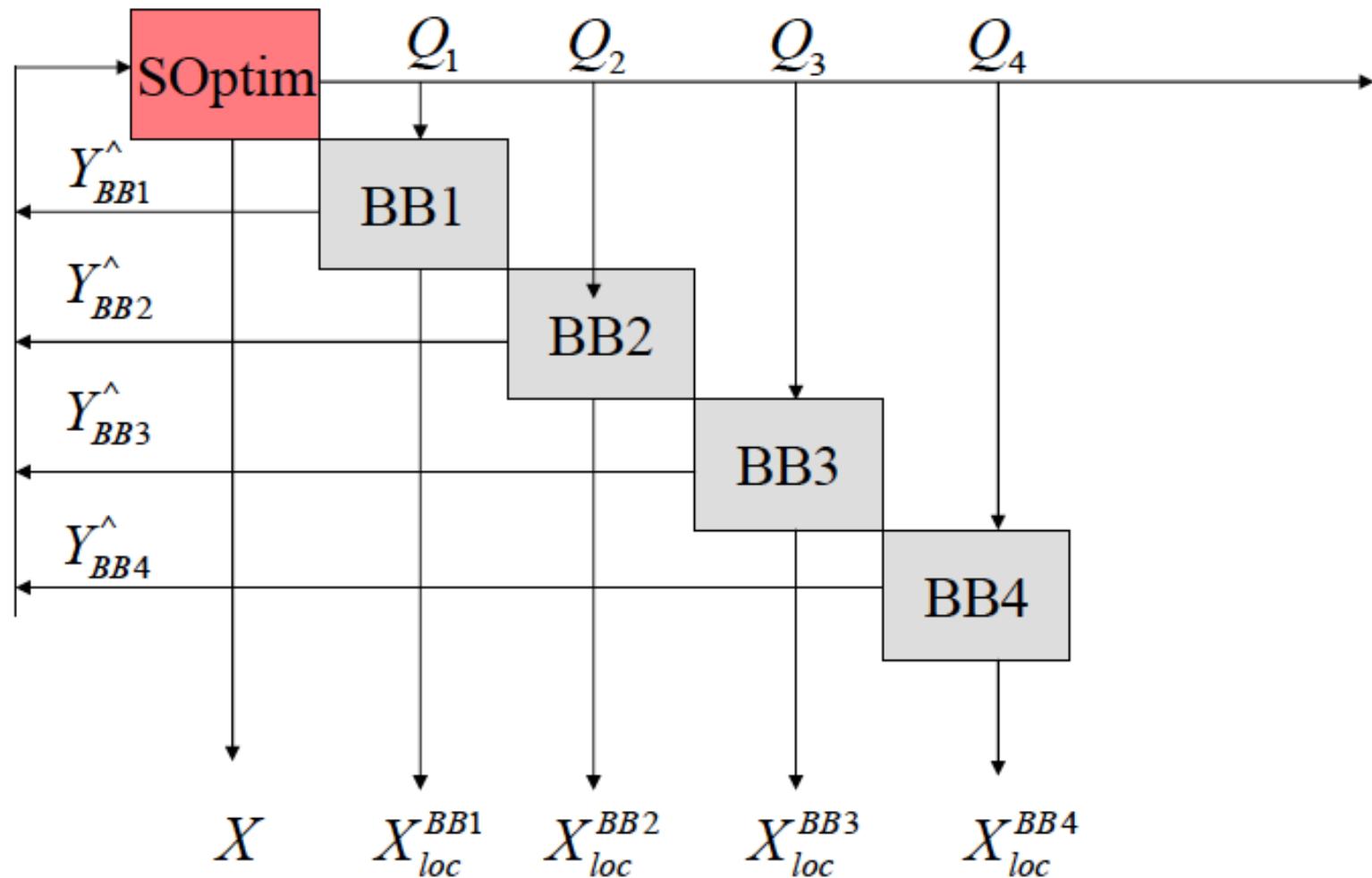
\mathbf{x} = subsystem local variables

\mathbf{y}_{ij} = coupling variables

- \mathbf{y}_{ij} = outputs of subsystem j which are needed as inputs to subsystem i .
- Coupling equations must also be satisfied, so coupling variables are included in subsystem objective.
- Used to reduce the number of system-level parameters.

BLISS 2000 Schematic

$$Q_i = \{X_{sh}, Y_{BBi}^*, w_{BBi}\}$$



Black Box (BB)

A black box has the following properties:

1. BB has its **own local variables** (X_{loc}) and has the exclusive right to determine X_{loc} . X_{loc} is a subset of decision variables that can appear explicitly only in the associated BB.
2. BB must satisfies its **constraints** at each system level iteration.
3. BB operates **independently** of other BB's. Neither its inputs nor its outputs are directly communicated between other BB's. Also, BB assumes no knowledge (e.g. X_{loc}) of other BB's. Instead, BB connection is done implicitly via the system optimizer, by the use of Y^* .
4. Computation methods within a BB are not restricted by BLISS. (It can be simulation or just an intelligent guess.)

BLISS 2000 Formulation

- BLISS is a bi-level optimization algorithm. The subsystem optimization formulation is as follows:

Given: $Q = \{X_{sh}, Y^*, w\}$

X_{sh} : share decision variable

variables: $U = \{X_{loc}, Y^\wedge\}$

Y^* : input to BB from other BB (surrogate var)

min: $f(U) = \sum_i w_i Y_i^\wedge$

w : weight used in BB optimization

s.t. $g(U) \leq 0$, for each BB

X_{loc} : local decision variable

$h(U) = 0$, for each BB

Y^\wedge : output of BB (to system and/or other BB)

$U_l \leq U \leq U_u$

$g(\bullet)$: BB inequality constraints

output: Y^\wedge

$h(\bullet)$: BB equality constraints

keep: X_{loc}

U_{lower} : lower bound on local variables

U_{upper} : upper bound on local variables

Lecture Summary

- Decomposition
 - Distributed analysis
 - Distributed design
- Coupling
- Collaborative Optimization
- Bi-Level Integrated System Synthesis

References

Jaroslaw Sobieski, Altus, Phillips, Sandusky, “Bi-level Integrated System Synthesis for Concurrent and Distributed Processing” AIAA Journal, Vol. 41, No.10, October 2003, pp. 1996-2003

I.P. Sobieski and I.M. Kroo. *Collaborative Optimization Using Response Surface Estimation*. AIAA Journal Vol. 38 No. 10. Oct 2000.

R.D. Braun and I.M. Kroo. *Development and Application of the Collaborative Optimization Architecture in a Multidisciplinary Design Environment*.

ICASE/NASA Langley Workshop on MDO, March 13-16, 1995

Erin J. Cramer et al. *Problem Formulation for Multidisciplinary Optimization*. SIAM Journal of Optimization. Vol. 4, No. 4 pp. 754-776, Nov 1994

Natalia M. Alexandrov (ed). *Multidisciplinary Design Optimization – State of the Art*. SIAM. 1994.

Lecture 07: Multidisciplinary System Analysis and Design Optimization (MSADO)

Design Space Exploration

February 18, 2015

Prof. Douglas Allaire

Today's Topics

- Design of experiments overview
- Full factorial design
- Parameter study
- One at a time
- Latin hypercubes
- Orthogonal arrays
- Effects
- DoE Paper Airplane Experiment

Design of Experiments

- A collection of statistical techniques providing a systematic way to sample the design space
- Useful when tackling a new problem for which you know very little about the design space
- Study the effects of multiple input variables on one or more output parameters
- Often used before setting up a formal optimization problem
 - Identify key drivers among potential design variables
 - Identify appropriate design variable ranges
 - Identify achievable objective function values
- Often, DOE is used in the context of **robust design**. Today we will just talk about it for design space exploration.

Design of Experiments

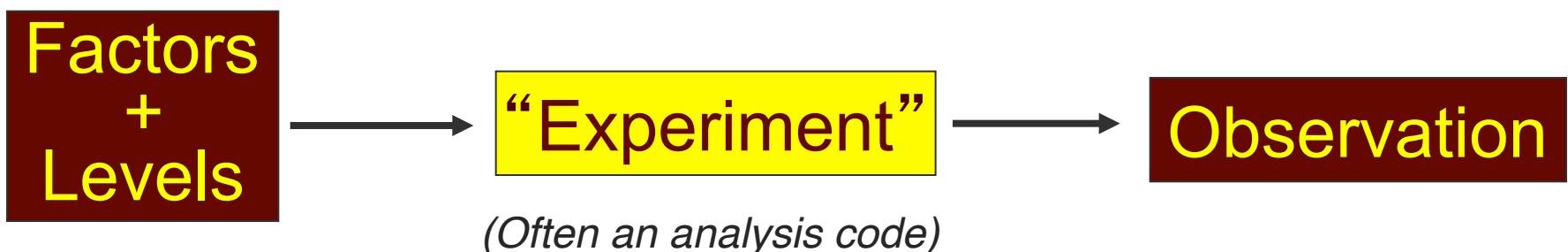
Design variables = **factors**

Values of design variables = **levels**

Noise factors = variables over which we have no control
e.g., manufacturing variation in blade thickness

Control factors = variables we can control e.g., nominal
blade thickness

Outputs = **observations** (= objective functions)



Matrix Experiments

- Each row of the matrix corresponds to one experiment.
- Each column of the matrix corresponds to one factor.
- Each experiment corresponds to a different combination of factor levels and provides one observation.

Expt No.	Factor A	Factor B	Observation
1	A1	B1	η_1
2	A1	B2	η_2
3	A2	B1	η_3
4	A2	B2	η_4

Here, we have two factors, each of which can take two levels.

Full Factorial Experiment

- Specify levels for each factor
- Evaluate outputs at every combination of values

n factors → l^n observations
— complete but expensive!

l levels

2 factors, 3 levels each:

$$l^n = 3^2 = 9 \text{ expts}$$

4 factors, 3 levels each:

$$l^n = 3^4 = 81 \text{ expts}$$



Expt No.	Factor	
	A	B
1	A1	B1
2	A1	B2
3	A1	B3
4	A2	B1
5	A2	B2
6	A2	B3
7	A3	B1
8	A3	B2
9	A3	B3

Fractional Factorial Experiments

- Due to the combinatorial explosion, we cannot usually perform a full factorial experiment
- So instead we consider just *some* of the possible combinations
- Questions:
 - How many experiments do I need?
 - Which combination of levels should I choose?
- Need to balance experimental cost with design space coverage

Fractional Factorial Design

Initially, it may be useful to look at a large number of factors superficially rather than a small number of factors in detail:

f_1	$I_{11}, I_{12}, I_{13}, I_{14}, \dots$
f_2	$I_{21}, I_{22}, I_{23}, I_{24}, \dots$
f_3	$I_{31}, I_{32}, I_{33}, I_{34}, \dots$

many levels

vs.

f_1	I_{11}, I_{12}
f_2	I_{21}, I_{22}
\vdots	\vdots
\vdots	\vdots
f_n	I_{n1}, I_{n2}

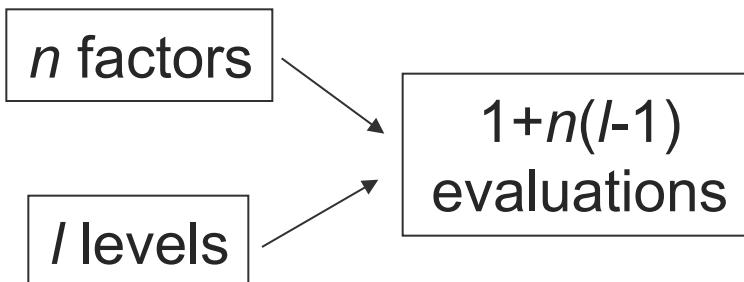
many factors

DoE Techniques Overview

TECHNIQUE	COMMENT	EXPENSE ($l=\#$ levels, $n=\#$ factors)
Full factorial design	Evaluates all possible designs.	l^n - grows exponentially with number of factors
Orthogonal arrays	Don't always seem to work - interactions?	Moderate – depends on which array
One at a time	Order of factors?	$1+n(l-1)$ - cheap
Latin hypercubes	Poor coverage if divisions are large.	l - cheap
Parameter study	Captures no interactions.	$1+n(l-1)$ - cheap

Parameter Study

- Specify levels for each factor
- Change one factor at a time, all others at base level
- Consider each factor at every level



Expt No.	Factor			
	A	B	C	D
1	A1	B1	C1	D1
2	A2	B1	C1	D1
3	A3	B1	C1	D1
4	A1	B2	C1	D1
5	A1	B3	C1	D1
6	A1	B1	C2	D1
7	A1	B1	C3	D1
8	A1	B1	C1	D2
9	A1	B1	C1	D3

4 factors, 3 levels each:

$$1+n(l-1) =$$

$$1+4(3-1) = 9 \text{ expts}$$



Baseline : A1, B1, C1, D1

Parameter Study

- Select the best result for each factor

Expt No.	Factor				Observation
	A	B	C	D	
1	A1	B1	C1	D1	η_1
2	A2	B1	C1	D1	η_2
3	A3	B1	C1	D1	η_3
4	A1	B2	C1	D1	η_4
5	A1	B3	C1	D1	η_5
6	A1	B1	C2	D1	η_6
7	A1	B1	C3	D1	η_7
8	A1	B1	C1	D2	η_8
9	A1	B1	C1	D3	η_9

1. Compare η_1, η_2, η_3
 $\Rightarrow A^*$
2. Compare η_1, η_4, η_5
 $\Rightarrow B^*$
3. Compare η_1, η_6, η_7
 $\Rightarrow C^*$
4. Compare η_1, η_8, η_9
 $\Rightarrow D^*$

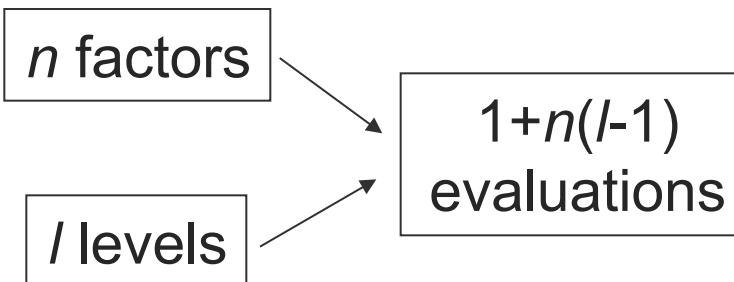


“Best design” is
 A^*, B^*, C^*, D^*

- Does not capture interaction between variables

One At a Time

- Change first factor, all others at base value
- If output is improved, keep new level for that factor
- Move on to next factor and repeat



4 factors, 3 levels each:

$$1+n(l-1) =$$

$$1+4(3-1) = 9 \text{ expts}$$

Expt No.	Factor			
	A	B	C	D
1	A1	B1	C1	D1
2	A2	B1	C1	D1
3	A3	B1	C1	D1
4	A*	B2	C1	D1
5	A*	B3	C1	D1
6	A*	B*	C2	D1
7	A*	B*	C3	D1
8	A*	B*	C*	D2
9	A*	B*	C*	D3

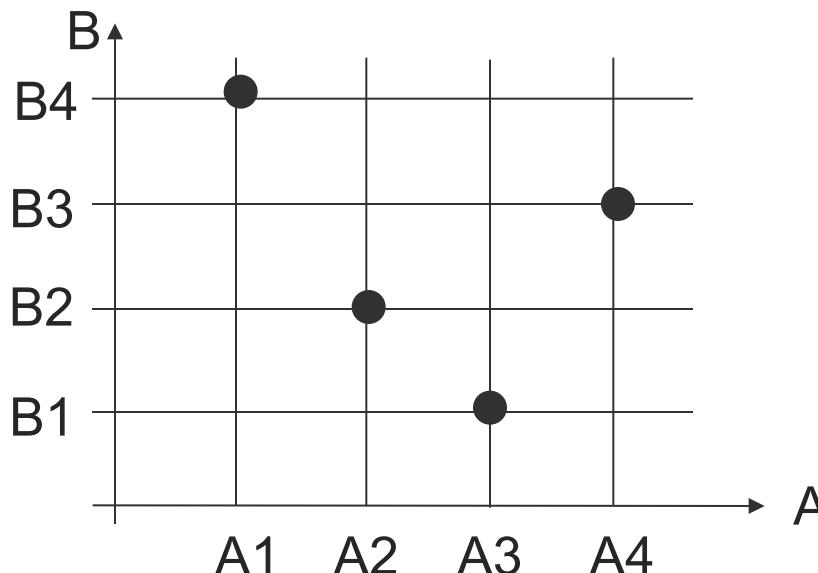
- Result depends on order of factors

Parameter Study vs. One at a Time

- Parameter study:
 - Chances are you will not actually evaluate the “best design” as part of your original experiment
 - “Best design” is chosen by extrapolating each factor’s behavior, but interactions are not considered
- One at a Time:
 - The “best design” is a member of your matrix experiment
 - Some interactions are captured, even though the result depends on the order of the factors

Latin Hypercubes

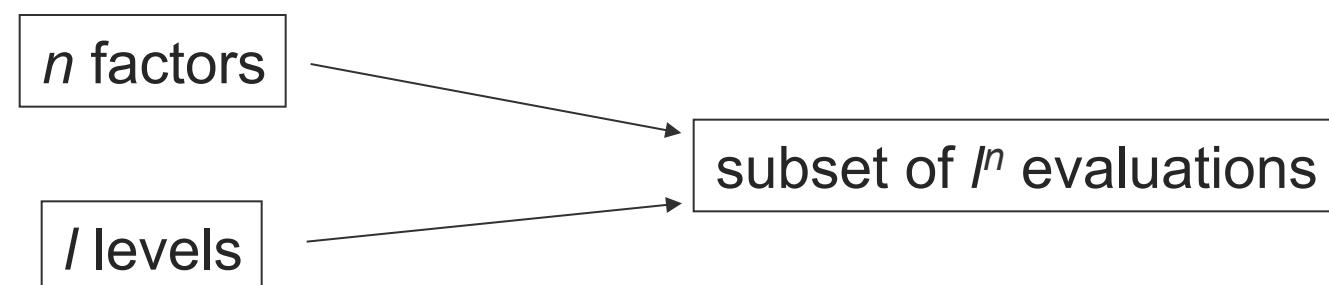
- Divide design space into l divisions for each factor
- Combine levels randomly
 - specify l points
 - use each level of a factor only once
- e.g. two factors, four levels each:



- Can have poor coverage although user has control over number of divisions
- Recent work to achieve space-filling designs

Orthogonal Arrays

- Specify levels for each factor
- Use arrays to choose a subset of the full-factorial experiment
- Subset selected to maintain orthogonality between factors

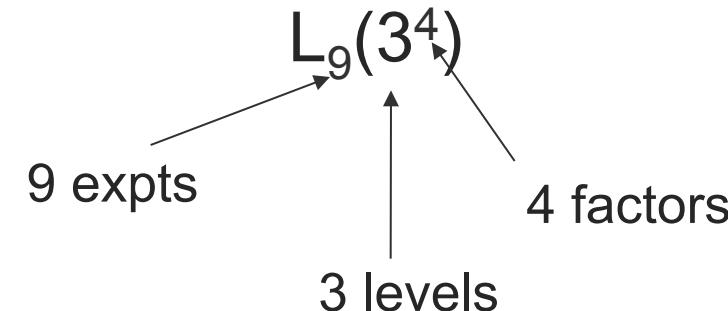
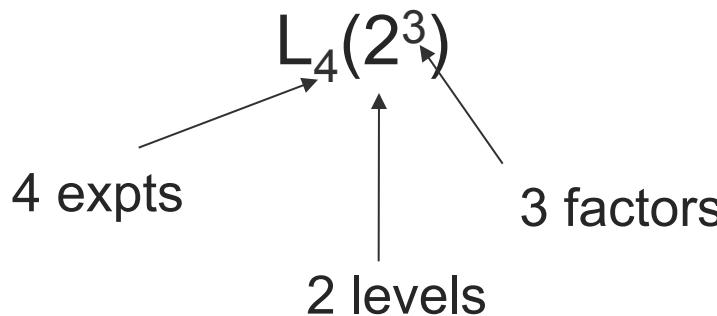


- Does not capture all interactions, but is efficient
- Experiment is balanced

Orthogonal Arrays

Expt No.	Factor		
	A	B	C
1	A1	B1	C1
2	A1	B2	C2
3	A2	B1	C2
4	A2	B2	C1

Expt No.	Factor			
	A	B	C	D
1	A1	B1	C1	D1
2	A1	B2	C2	D2
3	A1	B3	C3	D3
4	A2	B1	C2	D3
5	A2	B2	C3	D1
6	A2	B3	C1	D2
7	A3	B1	C3	D2
8	A3	B2	C1	D3
9	A3	B3	C2	D1



Orthogonality

Notice that for any pair of columns, all combinations of factor levels occur and they occur an equal number of times.

This is the **balancing property**.

In general, the balancing property is sufficient for orthogonality. There is a formal statistical definition of orthogonality, but we will not go into it here.

$L_9(3^4)$

Expt No.	Factor			
	A	B	C	D
1	A1	B1	C1	D1
2	A1	B2	C2	D2
3	A1	B3	C3	D3
4	A2	B1	C2	D3
5	A2	B2	C3	D1
6	A2	B3	C1	D2
7	A3	B1	C3	D2
8	A3	B2	C1	D3
9	A3	B3	C2	D1

*All of the combinations
(1-1, 1-2, 1-3, 2-1, 2-2,
2-3, 3-1, 3-2, 3-3)
occur once for each
pair of columns.*

Effects

Once the experiments have been performed, the results can be used to calculate effects.

The *effect* of a factor is the change in the response as the level of the factor is changed.

- **Main effects:** averaged individual measures of effects of factors
- **Interaction effects:** the effect of a factor depends on the level of another factor

Often, the effect is determined for a change from a minus level (-) to a plus level (+) (2-level experiments).

Effects

Consider the following experiment:

- We are studying the effect of three factors on the price of an aircraft
- The factors are the number of seats, range and aircraft manufacturer
- Each factor can take two levels:

Factor 1: Seats $100 < S_1 < 150$ $150 < S_2 < 200$

Factor 2: Range (nm) $2000 < R_1 < 2800$ $2800 < R_2 < 3500$

Factor 3: Manufacturer $M_1 = \text{Boeing}$ $M_2 = \text{Airbus}$

Main Effects

$L_8(2^3)$
(full factorial
design)

Expt No.	Seats (S)	Range (R)	Mfr (M)	Price (observation)
1	S1	R1	M1	P ₁
2	S1	R1	M2	P ₂
3	S1	R2	M1	P ₃
4	S1	R2	M2	P ₄
5	S2	R1	M1	P ₅
6	S2	R1	M2	P ₆
7	S2	R2	M1	P ₇
8	S2	R2	M2	P ₈

The **main effect** of a factor is the effect of that factor on the output averaged across the levels of other factors.

Main Effects

Question: what is the main effect of manufacturer? i.e. from our experiments, can we predict how the price is affected by whether Boeing or Airbus makes the aircraft?

Expt No.	Seats (S)	Range (R)	Mfr (M)	Price (observation)
1	S1	R1	M1	P ₁
2	S1	R1	M2	P ₂
3	S1	R2	M1	P ₃
4	S1	R2	M2	P ₄
5	S2	R1	M1	P ₅
6	S2	R1	M2	P ₆
7	S2	R2	M1	P ₇
8	S2	R2	M2	P ₈

} expts 1 and 2 differ only in the manufacturer

$$\frac{(P_2 - P_1) + (P_4 - P_3) + (P_6 - P_5) + (P_8 - P_7)}{4} = \text{main effect of manufacturer}$$

Main Effects – Another Interpretation

overall mean response:

$$m = \frac{P_1 + P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8}{8}$$

avg over all expts
when M=M1 :

$$m_{M1} = \frac{P_1 + P_3 + P_5 + P_7}{4}$$

effect of mfr level M1 $= m_{M1} - m$

effect of mfr level M2 $= m_{M2} - m$

Effect of factor level can be defined for multiple levels

main effect of mfr $= m_{M2} - m_{M1}$

Main effect of factor is defined as difference between two levels

NOTE: The main effect should be interpreted individually **only** if the variable does not appear to interact with other variables

Main Effect Example

Expt No.	Aircraft	Seats (S)	Range (R)	Mfr (M)	Price (\$M)
1	717	S1	R1	M1	24.0
2	A318-100	S1	R1	M2	29.3
3	737-700	S1	R2	M1	33.0
4	A319-100	S1	R2	M2	35.0
5	737-900	S2	R1	M1	43.7
6	A321-200	S2	R1	M2	48.0
7	737-800	S2	R2	M1	39.1
8	A320-200	S2	R2	M2	38.0

100 < S1 < 150

150 < S2 < 200

2000 < R1 < 2800

2800 < R2 < 3500

M1 = Boeing

M2 = Airbus

Sources:

Seats/Range data: Boeing Quick Looks

Price data: Aircraft Value News

Airline Monitor, May 2001 issue

Main Effect Example

$$\begin{aligned}\text{overall mean price} &= 1/8 * (24.0 + 29.3 + 33.0 + 35.0 + 43.7 + 48.0 + 39.1 + 38.0) \\ &= 36.26\end{aligned}$$

$$\begin{aligned}\text{mean of experiments with M1} &= 1/4 * (24.0 + 33.0 + 43.7 + 39.1) \\ &= 34.95\end{aligned}$$

$$\begin{aligned}\text{mean of experiments with M2} &= 1/4 * (29.3 + 35.0 + 48.0 + 38.0) \\ &= 37.58\end{aligned}$$

$$\text{Main effect of Boeing (M1)} = 34.95 - 36.26 = -1.3$$

$$\text{Main effect of Airbus (M2)} = 37.58 - 36.26 = 1.3$$

$$\text{Main effect of manufacturer} = 37.58 - 34.95 = 2.6$$

Interpretation?

Interaction Effects

We can also measure interaction effects between factors.

Answers the question: does the effect of a factor depend on the level of another factor?

e.g. Does the effect of manufacturer depend on whether we consider shorter range or longer range aircraft?

The interaction between manufacturer and range is defined as half the difference between the average manufacturer effect with range 2 and the average manufacturer effect with range 1.

$$\text{mfr} \times \text{range interaction} = \frac{\text{avg mfr effect with range 2} - \text{avg mfr effect with range 1}}{2}$$

Interaction Effects

range R1 : expts 1,2,5,6

range R2 : expts 3,4,7,8

Expt No.	Seats (S)	Range (R)	Mfr (M)	Price (\$M)
1	S1	R1	M1	24.0
2	S1	R1	M2	29.3
3	S1	R2	M1	33.0
4	S1	R2	M2	35.0
5	S2	R1	M1	43.7
6	S2	R1	M2	48.0
7	S2	R2	M1	39.1
8	S2	R2	M2	38.0

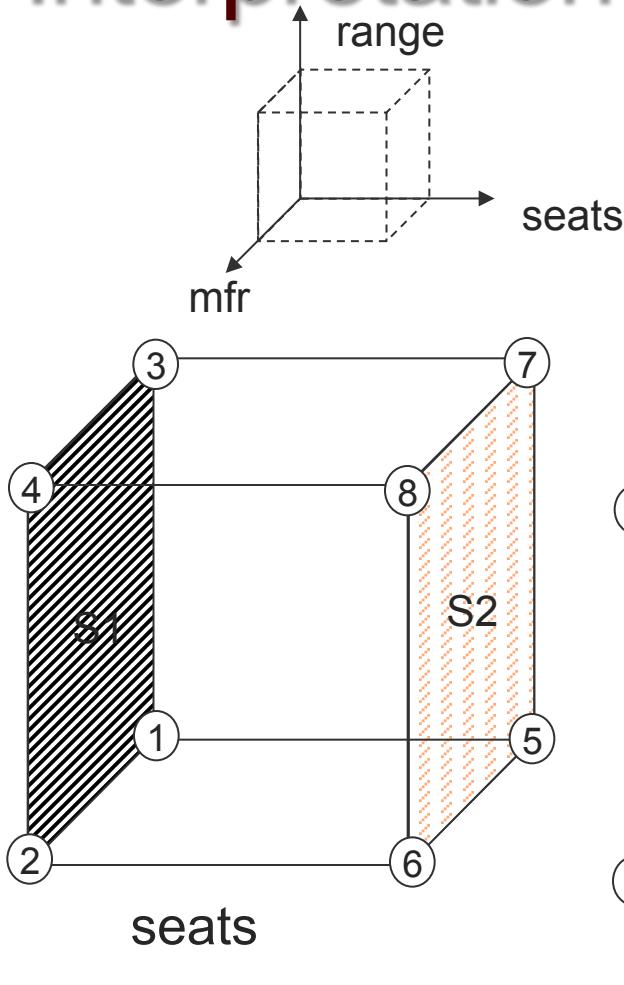
$$\text{avg mfr effect with range 1} = \frac{(P_2 - P_1) + (P_6 - P_5)}{2} = \frac{(29.3 - 24.0) + (48.0 - 43.7)}{2} = 4.8$$

$$\text{avg mfr effect with range 2} = \frac{(P_4 - P_3) + (P_8 - P_7)}{2} = \frac{(35.0 - 33.0) + (38.0 - 39.1)}{2} = 0.45$$

$$\text{mfr} \times \text{range interaction} = \frac{0.45 - 4.8}{2} = -2.2$$

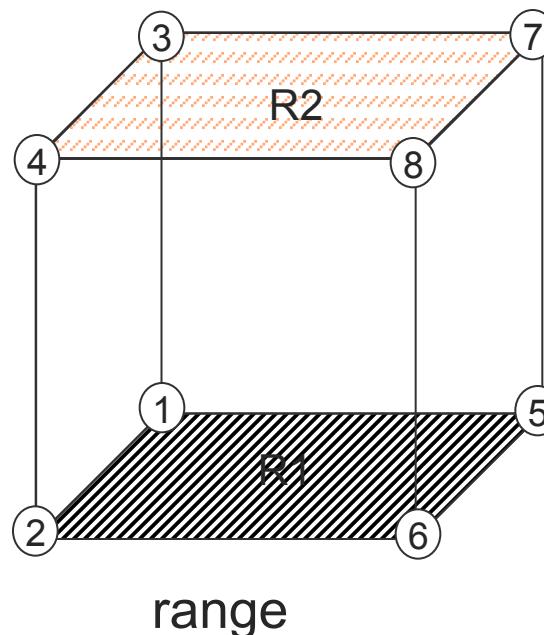
Interpretation?

Interpretation of Effects

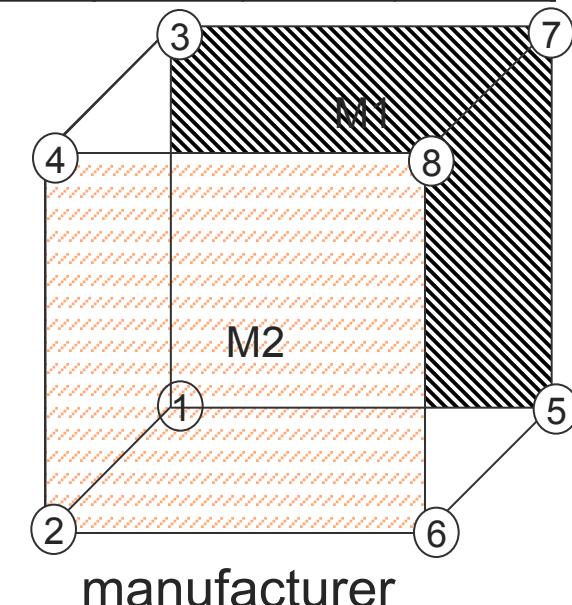


Main effects are the difference between two averages

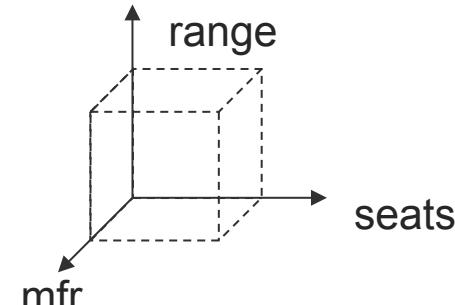
Expt No.	Seats (S)	Range (R)	Mfr (M)
1	S1	R1	M1
2	S1	R1	M2
3	S1	R2	M1
4	S1	R2	M2
5	S2	R1	M1
6	S2	R1	M2
7	S2	R2	M1
8	S2	R2	M2



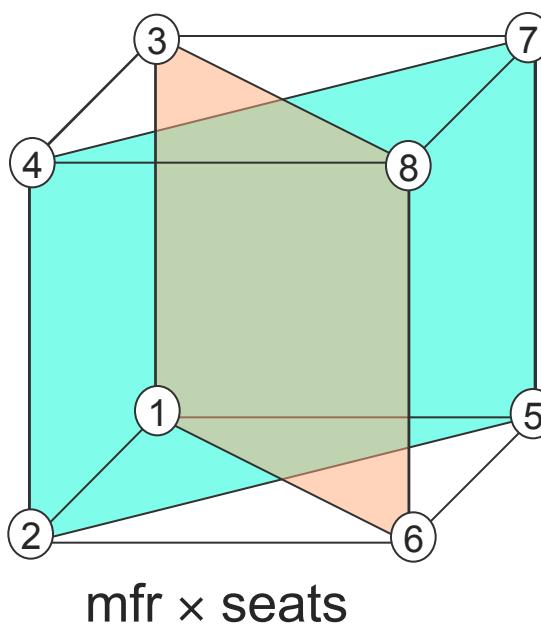
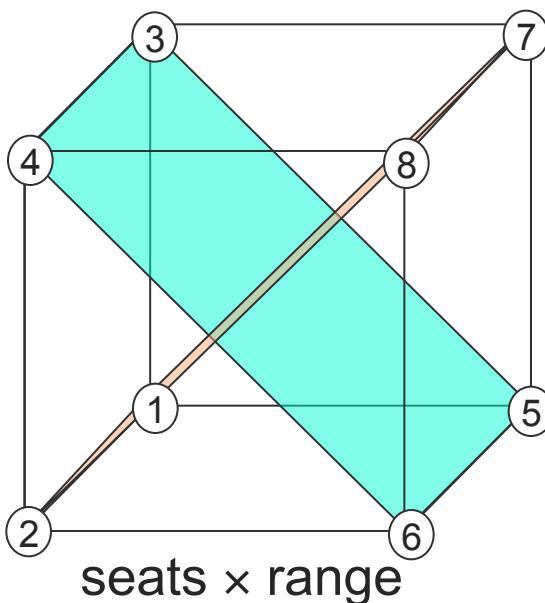
$$\text{main effect of mfr} = \frac{(P_8 + P_6 + P_4 + P_2) - (P_7 + P_5 + P_3 + P_1)}{4}$$



Interpretation of Effects

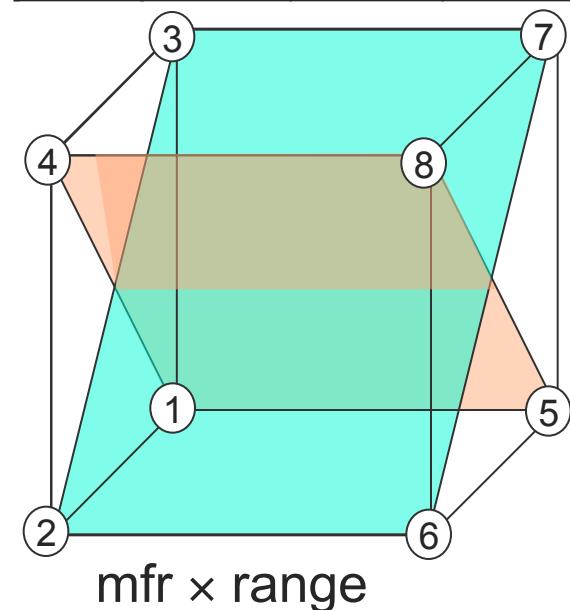


Interaction effects are also the difference between two averages, but the planes are no longer parallel



$$\begin{aligned} \text{mfr} \times \text{range} \\ \text{interaction} &= \frac{(P_8 + P_5 + P_4 + P_1) - (P_7 + P_6 + P_3 + P_2)}{4} \end{aligned}$$

Expt No.	Seats (S)	Range (R)	Mfr (M)
1	S1	R1	M1
2	S1	R1	M2
3	S1	R2	M1
4	S1	R2	M2
5	S2	R1	M1
6	S2	R1	M2
7	S2	R2	M1
8	S2	R2	M2



Design Experiment

Objective: Maximize Airplane Glide Distance

Design Variables:

- Weight Distribution
- Stabilizer Orientation
- Nose Length
- Wing Angle

Three levels for each design variable.

Design Experiment

Full factorial design : $3^4=81$ experiments

We will use an $L_9(3^4)$ orthogonal array:

Expt No.	Weight A	Stabilizer B	Nose C	Wing D
1	A1	B1	C1	D1
2	A1	B2	C2	D2
3	A1	B3	C3	D3
4	A2	B1	C2	D3
5	A2	B2	C3	D1
6	A2	B3	C1	D2
7	A3	B1	C3	D2
8	A3	B2	C1	D3
9	A3	B3	C2	D1

Design Experiment

Things to think about ...

Given just 9 out of a possible 81 experiments,
can we predict the optimal airplane?

Do some design variables seem to have a
larger effect on the objective than others
(sensitivity)?

Are there other factors affecting the results
(noise)?

References

- Phadke, : *Quality Engineering Using Robust Design*, Prentice Hall, 1995
- Box, G.; Hunter, W. and Hunter, J.: *Statistics for Experimenters*, John Wiley & Sons, 1978.

Lecture 08: Multidisciplinary System Analysis and Design Optimization (MSADO)

Gradient Calculation and
Sensitivity Analysis

February 23, 2015

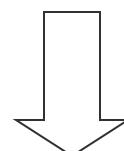
Prof. Douglas Allaire

Today's Topics

- Gradient calculation methods
 - Analytic and Symbolic
 - Finite difference
 - Complex step
 - Adjoint method
 - Automatic differentiation
- Post-Processing Sensitivity Analysis
 - Effect of changing design variables
 - Effect of changing parameters
 - Effect of changing constraints

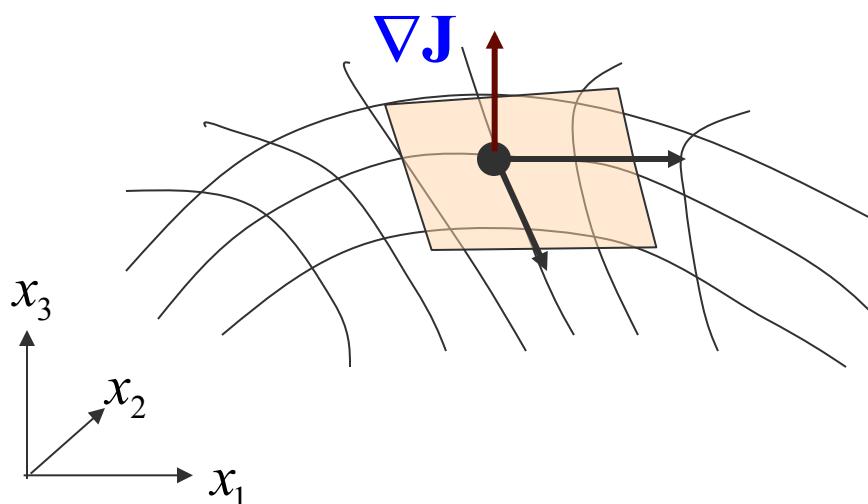
Definition of the Gradient

“How does the function J value change locally as we change elements of the design vector \mathbf{x} ? ”



Compute partial derivatives $\frac{\partial J}{\partial x_i}$ of J with respect to x_i

$$\nabla \mathbf{J} = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \\ \vdots \\ \frac{\partial J}{\partial x_n} \end{bmatrix}$$

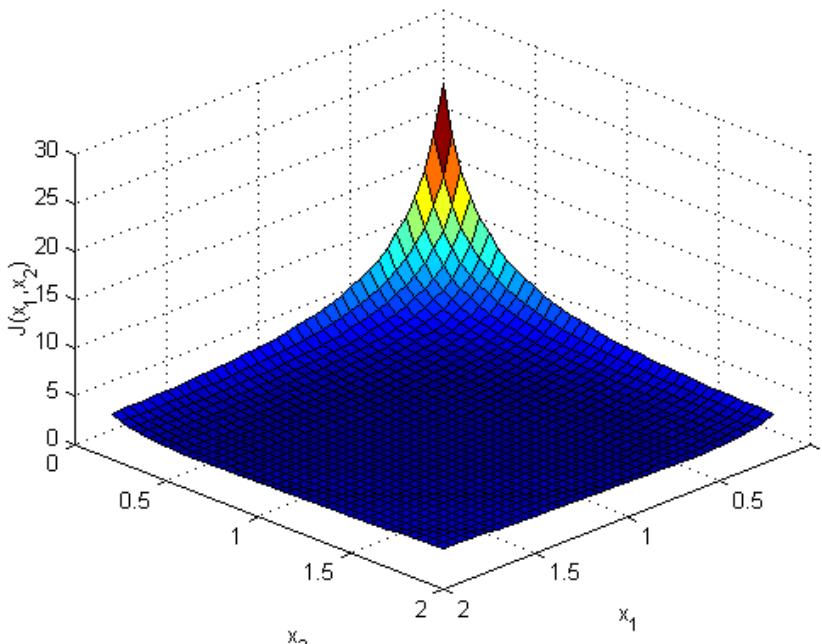


Gradient vector points normal to the tangent hyperplane of $J(\mathbf{x})$. That is, the gradient is orthogonal to the level sets of $J(\mathbf{x})$.

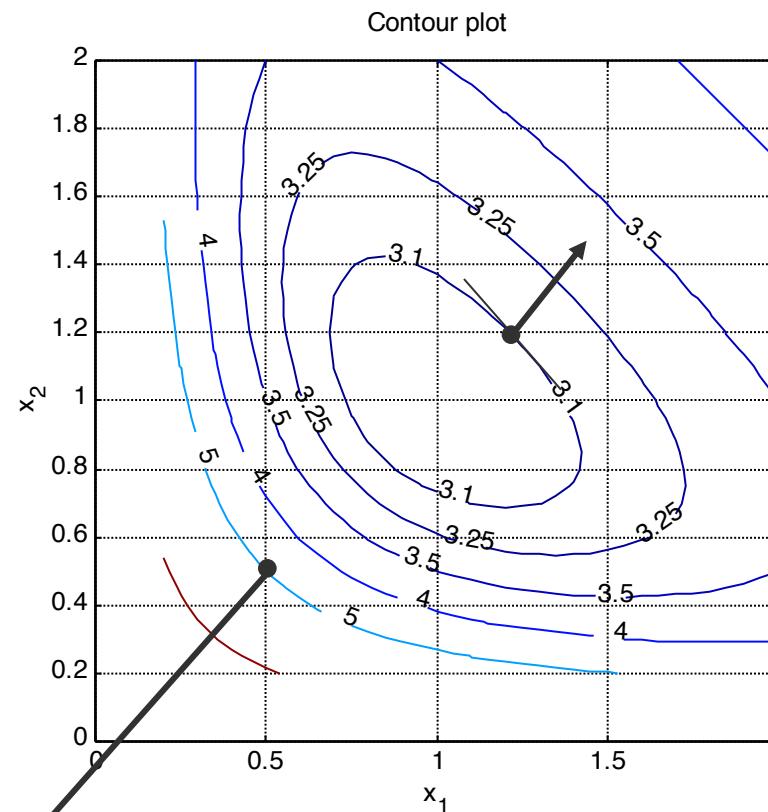
Geometry of Gradient Vector (2D)

Example function:

$$J(x_1, x_2) = x_1 + x_2 + \frac{1}{x_1 \cdot x_2}$$



$$\nabla J = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 - \frac{1}{x_1^2 x_2} \\ 1 - \frac{1}{x_1 x_2^2} \end{bmatrix}$$

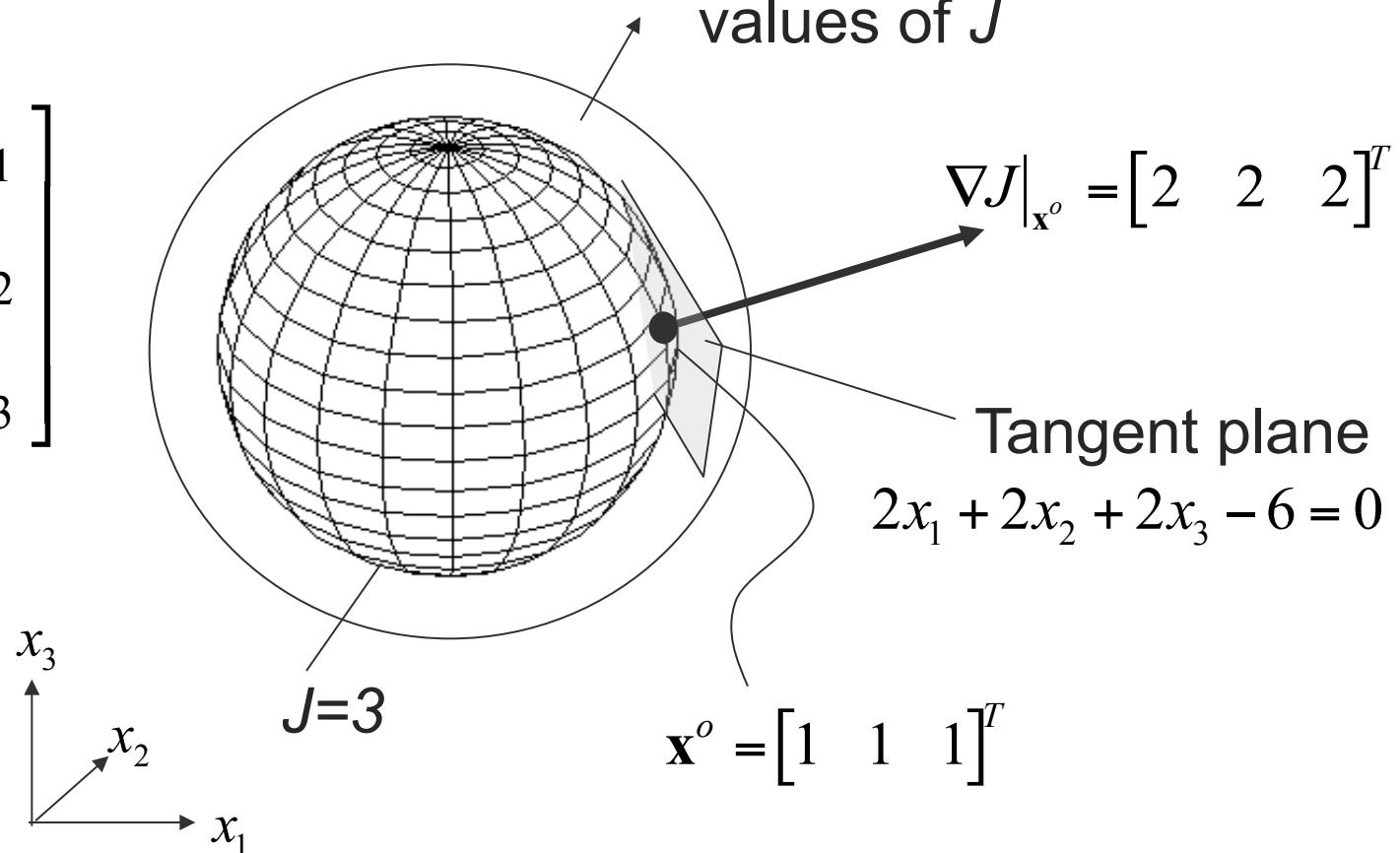


Gradient normal to contours

Geometry of Gradient Vector (3D)

Example $J = x_1^2 + x_2^2 + x_3^2$

$$\nabla J = \begin{bmatrix} 2x_1 \\ 2x_2 \\ 2x_3 \end{bmatrix}$$



Gradient vector points to larger values of J

Other Gradient-Related Quantities

- Jacobian: Matrix of derivatives of multiple functions w.r.t. vector of variables

$$\mathbf{J} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_z \end{bmatrix} \quad \xrightarrow{\hspace{1cm}} \quad \nabla \mathbf{J} = \begin{bmatrix} \frac{\partial J_1}{\partial x_1} & \frac{\partial J_2}{\partial x_1} & \dots & \frac{\partial J_z}{\partial x_1} \\ \frac{\partial J_1}{\partial x_2} & \frac{\partial J_2}{\partial x_2} & \dots & \frac{\partial J_z}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial J_1}{\partial x_n} & \frac{\partial J_2}{\partial x_n} & \dots & \frac{\partial J_z}{\partial x_n} \end{bmatrix}$$

$z \times 1 \qquad \qquad \qquad n \times z$

- Hessian: Matrix of second-order derivatives

$$\mathbf{H} = \nabla^2 J = \begin{bmatrix} \frac{\partial^2 J}{\partial x_1^2} & \frac{\partial^2 J}{\partial x_1 \partial x_2} & \dots & \frac{\partial^2 J}{\partial x_1 \partial x_n} \\ \frac{\partial^2 J}{\partial x_2 \partial x_1} & \frac{\partial^2 J}{\partial x_2^2} & \dots & \frac{\partial^2 J}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 J}{\partial x_n \partial x_1} & \frac{\partial^2 J}{\partial x_n \partial x_2} & \dots & \frac{\partial^2 J}{\partial x_n^2} \end{bmatrix} \quad n \times n$$

Why Calculate Gradients

- Required by gradient-based optimization algorithms
 - Normally need gradient of objective function and each constraint w.r.t. design variables at each iteration
 - Newton methods require Hessians as well
- Isoperformance/goal programming
- Robust design
- Post-processing sensitivity analysis
 - determine if result is optimal
 - sensitivity to parameters, constraint values

Analytical Sensitivities

If the objective function is known in closed form, we can often compute the gradient vector(s) in closed form (analytically):

Example: $J(x_1, x_2) = x_1 + x_2 + \frac{1}{x_1 \cdot x_2}$

Analytical Gradient: $\nabla J = \begin{bmatrix} \frac{\partial J}{\partial x_1} \\ \frac{\partial J}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 1 - \frac{1}{x_1^2 x_2} \\ 1 - \frac{1}{x_1 x_2^2} \end{bmatrix}$

Example

$x_1 = x_2 = 1$

$J(1,1)=3$

$\nabla J(1,1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Minimum

For complex systems analytical gradients are rarely available

Symbolic Differentiation

- Use symbolic mathematics programs
- e.g., Matlab, Maple, Mathematica

```
» syms x1 x2
» J=x1+x2+1/(x1*x2);
» dJdx1=diff(J,x1)
dJdx1 = 1-1/x1^2/x2
» dJdx2=diff(J,x2)
dJdx2 = 1-1/x1/x2^2
```

construct a symbolic object

difference operator

Finite Differences (I)

Function of a single variable $f(x)$

- First-order finite difference approximation of gradient:

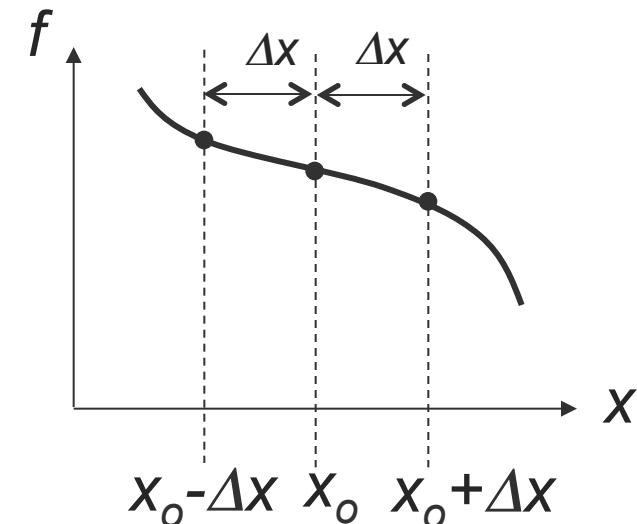
$$f'(x_o) = \frac{f(x_o + \Delta x) - f(x_o)}{\Delta x} + O(\Delta x)$$

Forward difference approximation to the derivative

- Second-order finite difference approximation of gradient:

$$f'(x_o) = \frac{f(x_o + \Delta x) - f(x_o - \Delta x)}{2\Delta x} + O(\Delta x^2)$$

Central difference approximation to the derivative



$x_o - \Delta x \quad x_o \quad x_o + \Delta x$

Finite Differences (II)

Approximations are derived from Taylor Series expansion:

$$f(x_o + \Delta x) = f(x_o) + \Delta x f'(x_o) + \frac{\Delta x^2}{2} f''(x_o) + O(\Delta x^3)$$

Neglect second order and higher order terms; solve for gradient vector:

$$f'(x_o) = \underbrace{\frac{f(x_o + \Delta x) - f(x_o)}{\Delta x}}_{\text{Forward Difference}} + \underbrace{O(\Delta x)}_{\text{Truncation Error}}$$

$O(\Delta x) = \frac{\Delta x}{2} f''(\xi)$

$$x_o \leq \xi \leq x_o + \Delta x$$

Finite Differences (III)

Take Taylor expansion backwards at $x_o - \Delta x$

$$f(x_o + \Delta x) = f(x_o) + \Delta x f'(x_o) + \frac{\Delta x^2}{2} f''(x_o) + O(\Delta x^3) \quad (1)$$

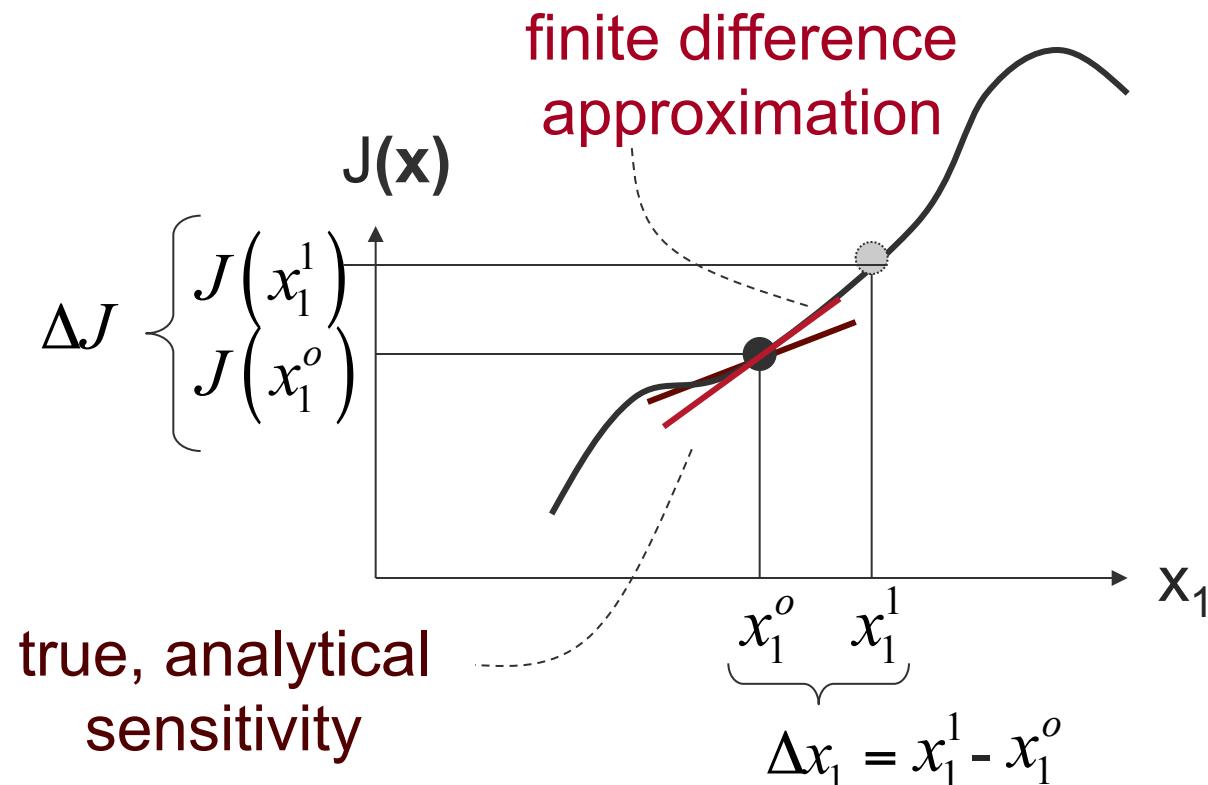
$$f(x_o - \Delta x) = f(x_o) - \Delta x f'(x_o) + \frac{\Delta x^2}{2} f''(x_o) + O(\Delta x^3) \quad (2) \leftarrow$$

(1) - (2) and solve again for derivative

$$f'(x_o) = \underbrace{\frac{f(x_o + \Delta x) - f(x_o - \Delta x)}{2\Delta x}}_{\text{Central Difference}} + \underbrace{O(\Delta x^2)}_{\text{Truncation Error}}$$
$$O(\Delta x^2) = \frac{\Delta x^2}{6} f'''(\xi)$$
$$x_o \leq \xi \leq x_o + \Delta x$$

Finite Differences (IV)

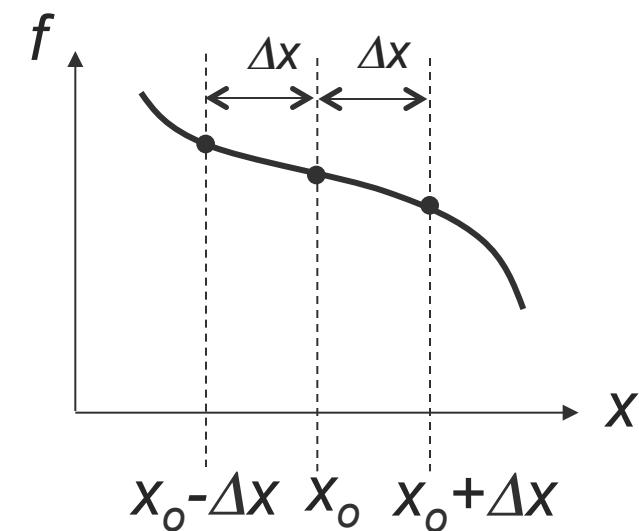
$$\frac{\partial J}{\partial x_1} \approx \frac{J(x_1^1) - J(x_1^o)}{x_1^1 - x_1^o} = \frac{J(x_1^o + \Delta x_1) - J(x_1^o)}{\Delta x_1} = \frac{\Delta J}{\Delta x_1}$$



Finite Differences (V)

- Second-order finite difference approximation of second derivative:

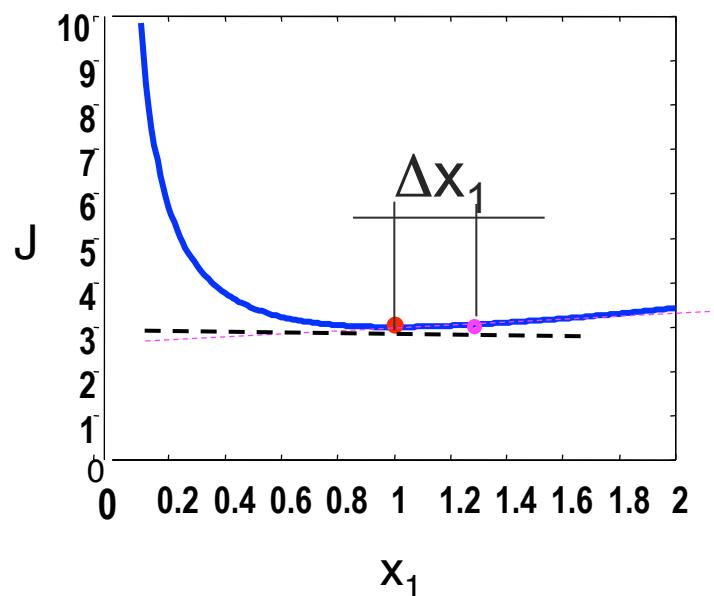
$$f''(x_o) \approx \frac{f(x_o + \Delta x) - 2f(x_o) + f(x_o - \Delta x)}{\Delta x^2}$$



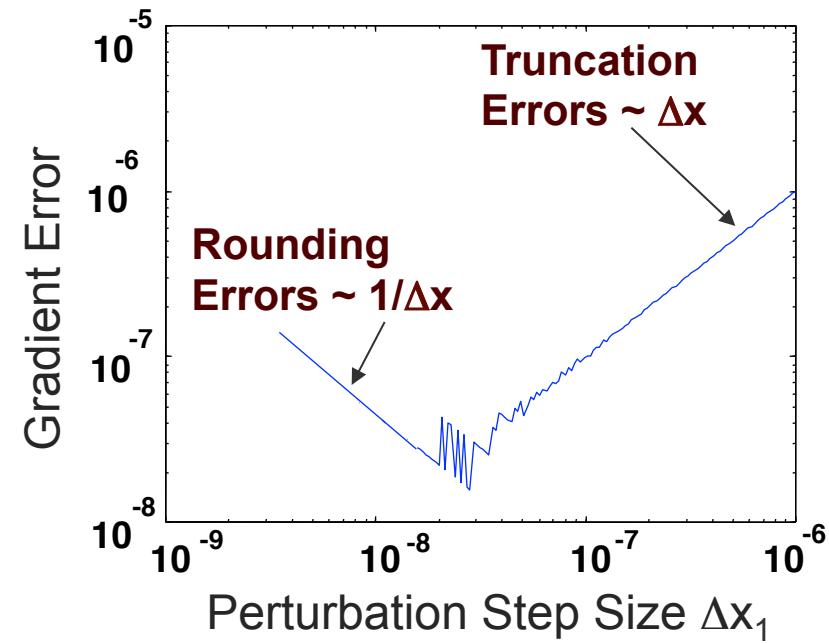
Errors of Finite Differencing

Caution: - Finite differencing generally has errors
- Very dependent on perturbation size

$$J(x_1, x_2) = x_1 + x_2 + \frac{1}{x_1 \cdot x_2}$$



$$\begin{aligned} x_1 &= x_2 = 1 \\ J(1,1) &= 3 \quad \nabla J(1,1) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \end{aligned}$$



→ Choice of Δx is critical

Computational Expense of FD

$$F(J_i)$$

Cost of a single objective function evaluation of J_i

$$n \cdot F(J_i)$$

Cost of gradient vector one-sided finite difference approximation for J_i for a design vector of length n

$$z \cdot n \cdot F(J_i)$$

Cost of Jacobian finite difference approximation with z objective functions

Example: 6 objectives

30 design variables

1 sec per function evaluation

3 min of CPU time
for a single Jacobian estimate - expensive!

Complex Step Derivative

- Similar to finite differences, but uses an imaginary step

$$f'(x_0) \approx \frac{\text{Im}[f(x_0 + i\Delta x)]}{\Delta x}$$

- Second order accurate
- Can use very small step sizes e.g. $\Delta x \approx 10^{-20}$
 - Doesn't have rounding error, since it doesn't perform subtraction
- Limited application areas
 - Code must be able to handle complex step values

J.R.R.A. Martins, I.M. Kroo and J.J. Alonso, An automated method for sensitivity analysis using complex variables, AIAA Paper 2000-0689, Jan 2000

Automatic Differentiation

- Mathematical formulas are built from a finite set of basic functions, e.g., additions, $\sin x$, $\exp x$, etc.
- Using chain rule, differentiate analysis code: add statements that generate derivatives of the basic functions
- Tracks numerical values of derivatives, does not track symbolically as discussed before
- Outputs modified program = original + derivative capability
- e.g., ADIFOR (Fortran), TAPENADE (C, Fortran), TOMLAB (Matlab), many more...
- Resources at <http://www.autodiff.org/>

Adjoint Methods

Consider the following problem:

$$\begin{aligned} \text{Minimize} \quad & J(\mathbf{x}, \mathbf{u}) \\ \text{s.t.} \quad & \mathbf{R}(\mathbf{x}, \mathbf{u}) = \mathbf{0} \end{aligned}$$

where \mathbf{x} are the design variables and \mathbf{u} are the state variables.
The constraints represent the state equation.

e.g., wing design: \mathbf{x} are shape variables, \mathbf{u} are flow variables, $\mathbf{R}(\mathbf{x}, \mathbf{u})=0$ represents the Navier Stokes equations.

We need to compute the gradients of J wrt \mathbf{x} :

$$\frac{dJ}{d\mathbf{x}} = \frac{\partial J}{\partial \mathbf{x}} + \frac{\partial J}{\partial \mathbf{u}} \frac{d\mathbf{u}}{d\mathbf{x}}$$

Typically the dimension of \mathbf{u} is very high (thousands/millions).

Adjoint Methods

$$\frac{dJ}{dx} = \frac{\partial J}{\partial x} + \frac{\partial J}{\partial u} \frac{du}{dx}$$

- To compute du/dx , differentiate the state equation:

$$\frac{dR}{dx} = \frac{\partial R}{\partial x} + \frac{\partial R}{\partial u} \frac{du}{dx} = 0$$

$$\frac{\partial R}{\partial u} \frac{du}{dx} = - \frac{\partial R}{\partial x}$$

$$\frac{du}{dx} = - \left(\frac{\partial R}{\partial u} \right)^{-1} \frac{\partial R}{\partial x}$$

Adjoint Methods

- We have

$$\frac{dJ}{dx} = \frac{\partial J}{\partial x} + \frac{\partial J}{\partial u} \frac{du}{dx} = \frac{\partial J}{\partial x} - \underbrace{\frac{\partial J}{\partial u} \left(\frac{\partial R}{\partial u} \right)^{-1}}_{\lambda^T} \frac{\partial R}{\partial x}$$

- Now define

$$\lambda = \left[\frac{\partial J}{\partial u} \left(\frac{\partial R}{\partial u} \right)^{-1} \right]^T \quad \lambda^T$$

- Then to determine the gradient:

First solve $\left(\frac{\partial R}{\partial u} \right)^T \lambda = \left(\frac{\partial J}{\partial u} \right)^T$ (adjoint equation)

Then compute

$$\frac{dJ}{dx} = \frac{\partial J}{\partial x} - \lambda^T \frac{\partial R}{\partial x}$$

Adjoint Methods

- Solving adjoint equation

$$\left(\frac{\partial \mathbf{R}}{\partial \mathbf{u}} \right)^T \boldsymbol{\lambda} = \left(\frac{\partial J}{\partial \mathbf{u}} \right)^T$$

about same cost as solving forward problem
(function evaluation)

- Adjoints widely used in aerodynamic shape optimization, optimal flow control, geophysics applications, etc.
- Some automatic differentiation tools have ‘reverse mode’ for computing adjoints

Post-Processing Sensitivity Analysis

- A sensitivity analysis is an important component of post-processing
- Key to understanding which design variables, constraints, and parameters are important drivers for the optimum solution
- How sensitive is the “optimal” solution J^* to changes or perturbations of the design variables x^* ?
- How sensitive is the “optimal” solution x^* to changes in the constraints $g(x)$, $h(x)$ and fixed parameters p?

Sensitivity Analysis

Questions for aircraft design:

How does my solution change if I

- change the cruise altitude?
- change the cruise speed?
- change the range?
- change material properties?
- relax the constraint on payload?
- ...



Airbus A380

Sensitivity Analysis: Spacecraft

Questions for spacecraft design:

How does my solution change if I

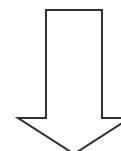
- change the orbital altitude?
- change the transmission frequency?
- change the specific impulse of the propellant?
- change launch vehicle?
- change desired mission lifetime?
- ...



Globalstar

Sensitivity Analysis

“How does the optimal solution change as we change the problem parameters?”



*effect on design variables
effect on objective function
effect on constraints*

Want to answer this question without having to solve the optimization problem again.

Normalization

In order to compare sensitivities from different design variables in terms of their *relative* sensitivity it may be necessary to normalize:

$$\frac{\partial J}{\partial x_i} \Bigg|_{\mathbf{x}^0}$$

“raw” - unnormalized sensitivity =
partial derivative evaluated at point $\mathbf{x}_{i,0}$

$$\frac{\Delta J/J}{\Delta x_i/x_i} = \frac{x_{i,0}}{J(\mathbf{x}^0)} \cdot \frac{\partial J}{\partial x_i} \Bigg|_{\mathbf{x}^0}$$

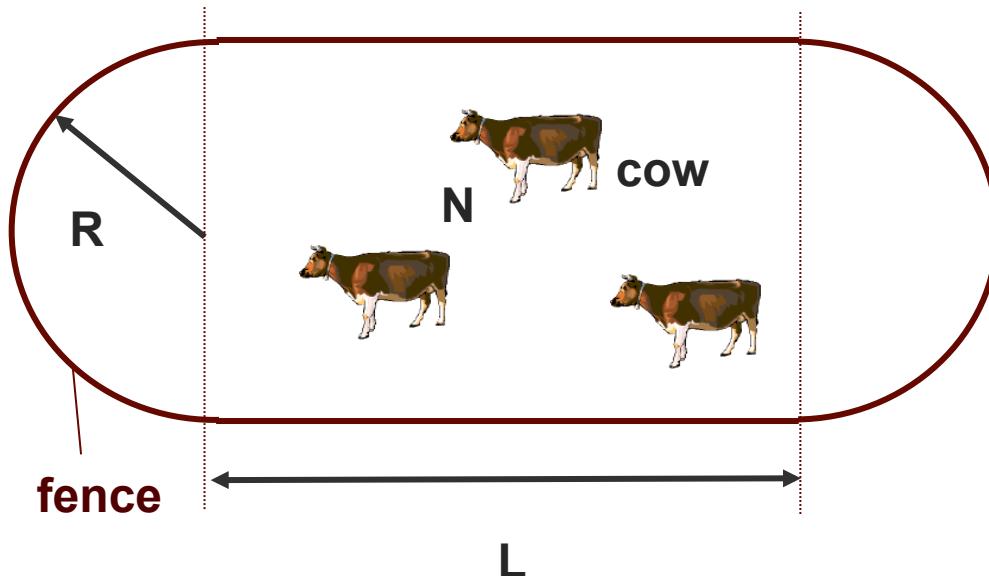
Normalized sensitivity captures
relative sensitivity
 \sim % change in objective per
% change in design variable



Important for comparing effect between design variables

Example: Dairy Farm Problem

“Dairy Farm” sample problem



$$\left. \begin{array}{l} L - \text{Length} = 100 \text{ [m]} \\ N - \# \text{ of cows} = 10 \\ R - \text{Radius} = 50 \text{ [m]} \end{array} \right\} x^o$$

With respect to which design variable is the objective most sensitive?

Parameters:
 $f=100\$/m$
 $n=2000\$/cow$
 $m=2\$/liter$

$$A = 2LR + \pi R^2$$

$$F = 2L + 2\pi R$$

$$M = 100 \cdot \sqrt{A / N}$$

$$C = f \cdot F + n \cdot N$$

$$I = N \cdot M \cdot m$$

$$P = I - C$$

Dairy Farm Sensitivity

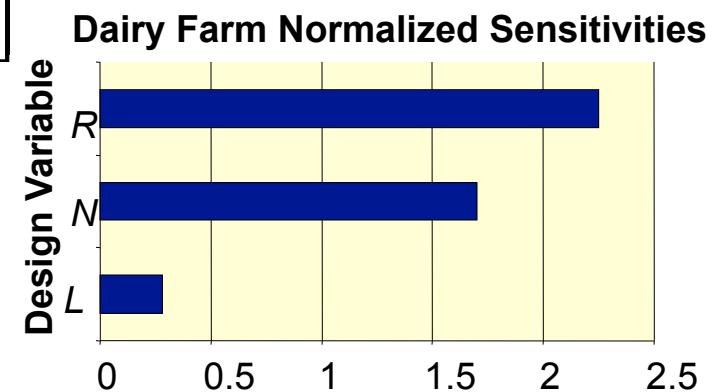
- Compute objective at $\mathbf{x}^o \quad J(\mathbf{x}^o) = 13092$
- Then compute raw sensitivities

$$\nabla J = \begin{bmatrix} \frac{\partial P}{\partial L} \\ \frac{\partial P}{\partial N} \\ \frac{\partial P}{\partial R} \end{bmatrix} = \begin{bmatrix} 36.6 \\ 2225.4 \\ 588.4 \end{bmatrix}$$

- Normalize

$$\nabla \bar{J} = \frac{\mathbf{x}^o}{J(\mathbf{x}^o)} \nabla J = \begin{bmatrix} \frac{100}{13092} \cdot 36.6 \\ \frac{10}{13092} 2225.4 \\ \frac{50}{13092} 588.4 \end{bmatrix} = \begin{bmatrix} 0.28 \\ 1.7 \\ 2.25 \end{bmatrix}$$

- Show graphically with tornado chart



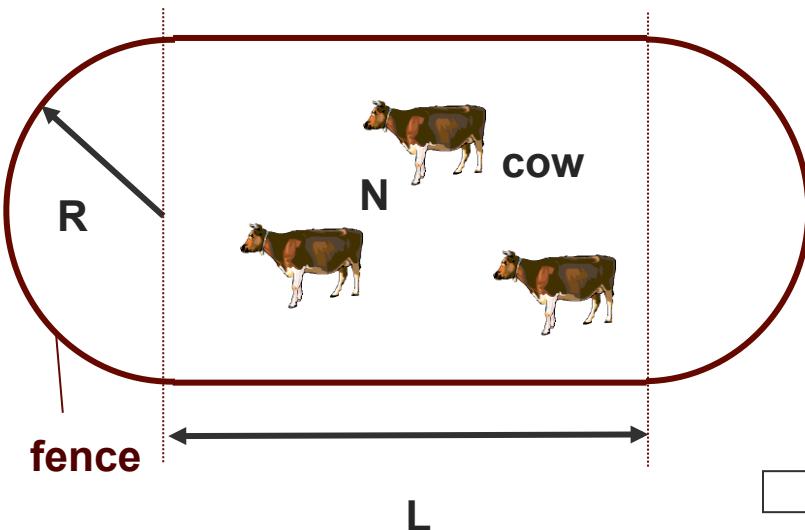
Parameters

Parameters p are the fixed assumptions.

How sensitive is the optimal solution x^* with respect to fixed parameters ?

Example:

“Dairy Farm” sample problem



Maximize Profit

Optimal solution:

$$x^* = [R=106.1\text{m}, L=0\text{m}, N=17 \text{ cows}]^\top$$

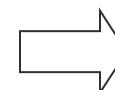
Fixed parameters:

Parameters:

$f=100\$/\text{m}$ - Cost of fence

$n=2000\$/\text{cow}$ - Cost of a single cow

$m=2\$/\text{liter}$ - Market price of milk



How does x^* change as parameters change?

Sensitivity Analysis

$$\text{KKT conditions: } \nabla J(\mathbf{x}^*) + \sum_{j \in M} \lambda_j \nabla \hat{g}_j(\mathbf{x}^*) = 0$$

$$\hat{g}_j(\mathbf{x}^*) = 0, \quad j \in M$$

$$\lambda_j > 0, \quad j \in M$$

Set of active constraints

For a small change in a parameter, p , we require that the KKT conditions remain valid:

$$\frac{d(\text{KKT conditions})}{dp} = 0$$

Rewrite first equation:

$$\frac{\partial J}{\partial x_i}(\mathbf{x}^*) + \sum_{j \in M} \lambda_j \frac{\partial \hat{g}_j}{\partial x_i}(\mathbf{x}^*) = 0, \quad i = 1, \dots, n$$

Sensitivity Analysis

Recall chain rule. If: $Y = Y(p, \mathbf{x}(p))$ then

$$\frac{dY}{dp} = \frac{\partial Y}{\partial p} + \sum_{k=1}^n \frac{\partial Y}{\partial x_k} \frac{\partial x_k}{\partial p}$$

Applying to first equation of KKT conditions:

$$\begin{aligned} & \frac{d}{dp} \left(\frac{\partial J(\mathbf{x}, p)}{\partial x_i} + \sum_{j \in M} \lambda_j(p) \frac{\partial \hat{g}_j(\mathbf{x}, p)}{\partial x_i} \right) \\ &= \frac{\partial^2 J}{\partial x_i \partial p} + \sum_{j \in M} \lambda_j \frac{\partial^2 \hat{g}_j}{\partial x_i \partial p} + \sum_{k=1}^n \left(\frac{\partial^2 J}{\partial x_i \partial x_k} + \sum_{j \in M} \lambda_j \frac{\partial^2 \hat{g}_j}{\partial x_i \partial x_k} \right) \frac{\partial x_k}{\partial p} + \sum_{j \in M} \frac{\partial \lambda_j}{\partial p} \frac{\partial \hat{g}_j}{\partial x_i} = 0 \\ & \sum_{k=1}^n A_{ik} \frac{\partial x_k}{\partial p} + \sum_{j \in M} B_{ij} \frac{\partial \lambda_j}{\partial p} + c_i = 0 \end{aligned}$$

Sensitivity Analysis

Perform same procedure on equation: $\hat{g}_j(x^*, p) = 0$

$$\frac{\partial \hat{g}_j}{\partial p} + \sum_{k=1}^n \frac{\partial \hat{g}_j}{\partial x_k} \frac{\partial x_k}{\partial p} = 0$$

$$\sum_{k=1}^n B_{kj} \frac{\partial x_k}{\partial p} + d_j = 0$$

Sensitivity Analysis

In matrix form we can write:

$$\begin{array}{c} \xrightarrow{n} \quad \xrightarrow{M} \\ \uparrow \quad \uparrow \\ \left[\begin{array}{cc} A & B \\ B^T & 0 \end{array} \right] \left\{ \begin{array}{l} \delta \mathbf{x} \\ \delta \boldsymbol{\lambda} \end{array} \right\} + \left\{ \begin{array}{l} c \\ d \end{array} \right\} = 0 \end{array}$$

$$A_{ik} = \frac{\partial^2 J}{\partial \mathbf{x}_i \partial \mathbf{x}_k} + \sum_{j \in M} \lambda_j \frac{\partial^2 \hat{g}_j}{\partial \mathbf{x}_i \partial \mathbf{x}_k}$$

$$B_{ij} = \frac{\partial \hat{g}_j}{\partial \mathbf{x}_i}$$

$$c_i = \frac{\partial^2 J}{\partial \mathbf{x}_i \partial p} + \sum_{j \in M} \lambda_j \frac{\partial^2 \hat{g}_j}{\partial \mathbf{x}_i \partial p}$$

$$d_j = \frac{\partial \hat{g}_j}{\partial p}$$

$$\delta \mathbf{x} = \left\{ \begin{array}{l} \frac{\partial \mathbf{x}_1}{\partial p} \\ \frac{\partial \mathbf{x}_2}{\partial p} \\ \vdots \\ \frac{\partial \mathbf{x}_n}{\partial p} \end{array} \right\} \quad \delta \boldsymbol{\lambda} = \left\{ \begin{array}{l} \frac{\partial \lambda_1}{\partial p} \\ \frac{\partial \lambda_2}{\partial p} \\ \vdots \\ \frac{\partial \lambda_M}{\partial p} \end{array} \right\}$$

Sensitivity Analysis

We solve the system to find $\delta\mathbf{x}$ and $\delta\lambda$, then the sensitivity of the objective function with respect to p can be found:

$$\frac{dJ}{dp} = \frac{\partial J}{\partial p} + \nabla J^T \delta\mathbf{x}$$

$$\Delta J \approx \frac{dJ}{dp} \Delta p$$

(first-order approximation)

$$\Delta\mathbf{x} \approx \delta\mathbf{x} \Delta p$$

To assess the effect of changing a different parameter, we only need to calculate a new RHS in the matrix system.

Sensitivity Analysis – Constraints

- We also need to assess when an active constraint will become inactive and vice versa
- An active constraint will become inactive when its Lagrange multiplier goes to zero:

$$\Delta\lambda_j = \frac{\partial\lambda_j}{\partial p} \Delta p = \delta\lambda_j \Delta p$$

Find the Δp that makes λ_j zero:

$$\lambda_j + \delta\lambda_j \Delta p = 0$$

$$\Delta p = \frac{-\lambda_j}{\delta\lambda_j} \quad j \in M$$

This is the amount by which we can change p before the j^{th} constraint becomes inactive (to a first order approximation)

Sensitivity Analysis – Constraints

An inactive constraint will become active when $g_j(\mathbf{x})$ goes to zero:

$$g_j(\mathbf{x}) = g_j(\mathbf{x}^*) + \Delta p [\nabla g_j(\mathbf{x}^*)^T \delta \mathbf{x}] = 0$$

Find the Δp that makes g_j zero:

$$\Delta p = \frac{-g_j(\mathbf{x}^*)}{\nabla g_j(\mathbf{x}^*)^T \delta \mathbf{x}}$$

for all j not active at \mathbf{x}^*

- This is the amount by which we can change p before the j^{th} constraint becomes active (to a first order approximation)
- If we want to change p by a larger amount, then the problem must be solved again including the new constraint
- Only valid close to the optimum

Lagrange Multiplier Interpretation

- Consider the problem:

$$\text{minimize } J(\mathbf{x}) \text{ s.t. } \mathbf{h}(\mathbf{x})=0$$

with optimal solution \mathbf{x}^*

- What happens if we change constraint k by a small amount?

$$h_k(\mathbf{x}^*) = \varepsilon \quad h_j(\mathbf{x}^*) = 0, \quad \forall j \neq k$$

- Differentiating w.r.t ε

$$\nabla h_k \frac{d\mathbf{x}^*}{d\varepsilon} = 1 \quad \nabla h_j \frac{d\mathbf{x}^*}{d\varepsilon} = 0, \quad \forall j \neq k$$

Lagrange Multiplier Interpretation

- How does the objective function change?

$$\frac{dJ}{d\varepsilon} = \nabla J \frac{d\mathbf{x}^*}{d\varepsilon}$$

- Using KKT conditions:

$$\frac{dJ}{d\varepsilon} = \left(-\sum_j \lambda_j \nabla h_j \right) \frac{d\mathbf{x}^*}{d\varepsilon} = -\sum_j \lambda_j \nabla h_j \frac{d\mathbf{x}^*}{d\varepsilon} = -\lambda_k$$

- Lagrange multiplier is negative of sensitivity of cost function to constraint value. Also called *shadow price*.

Lecture Summary

- Gradient calculation approaches
 - Analytical and Symbolic
 - Finite difference
 - Automatic Differentiation
 - Adjoint methods
- Sensitivity analysis
 - Yields important information about the design space, both as the optimization is proceeding and once the “optimal” solution has been reached.

Reading
Papalambros – Section 8.2 Computing Derivatives

Lecture 09: Multidisciplinary System Analysis and Design Optimization (MSADO)

An Introduction to Simulated Annealing

February 25, 2015

Prof. Douglas Allaire

Today's Topics

- Heuristics
- Background in Statistical Mechanics
 - Atom Configuration Problem
 - Metropolis Algorithm
- Simulated Annealing Algorithm
- Sample Problems and Applications
- Summary

Learning Objectives

- Review background in **Statistical Mechanics**: configuration, ensemble, entropy, heat capacity
- Understand the basic assumptions and steps in **Simulated Annealing (SA)**
- Be able to transform **design problems** into a combinatorial optimization problem suitable to SA
- Understand **strengths and weaknesses** of SA

What is a Heuristic?

- A Heuristic is simply a **rule of thumb** that hopefully will find a good answer.
- **Why** use a Heuristic?
 - Heuristics are typically used to solve **complex** (large, nonlinear, non-convex (i.e., contain local minima)) **multivariate combinatorial optimization problems** that are difficult to solve to optimality.
- Unlike gradient-based methods in a convex design space, heuristics are **NOT guaranteed** to find the true **global optimal solution** in a single objective problem, but should find many good solutions (the **mathematician's** answer vs. the **engineer's** answer)
- Heuristics are **good at dealing with local optima** without getting stuck in them while searching for the global optimum.

Types of Heuristics

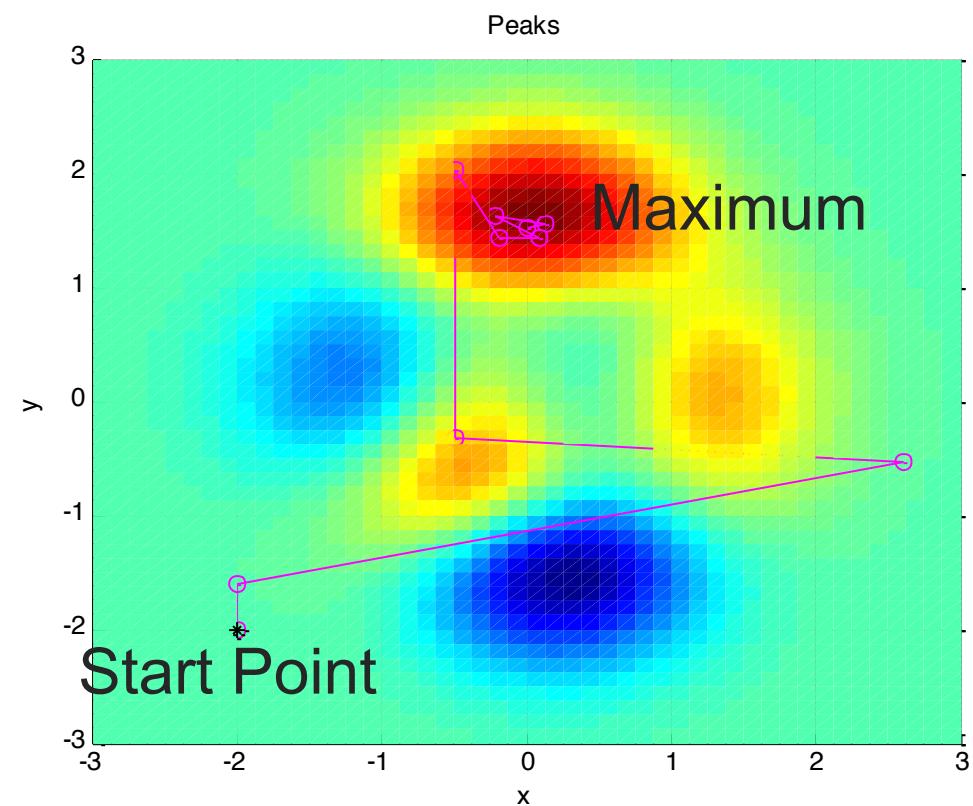
- Heuristics Often Incorporate Randomization
- **3 Most Common Heuristic Techniques**
 - Simulated Annealing
 - Genetic Algorithms
 - Tabu Search
 - New Methods: Particle Swarm Optimization, etc...

Origin of Simulated Annealing (SA)

- **Definition:** A **heuristic** technique that mathematically mirrors the **cooling** of a set of atoms to a state of minimum energy.
- **Origin:** Applying the field of **Statistical Mechanics** to the field of **Combinatorial Optimization** (1983)
- Draws an **analogy** between the cooling of a material (search for minimum energy state) and the solving of an optimization problem.
- **Original Paper** Introducing the Concept
 - Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., “Optimization by Simulated Annealing,” *Science*, Volume 220, Number 4598, 13 May 1983, pp. 671-680.

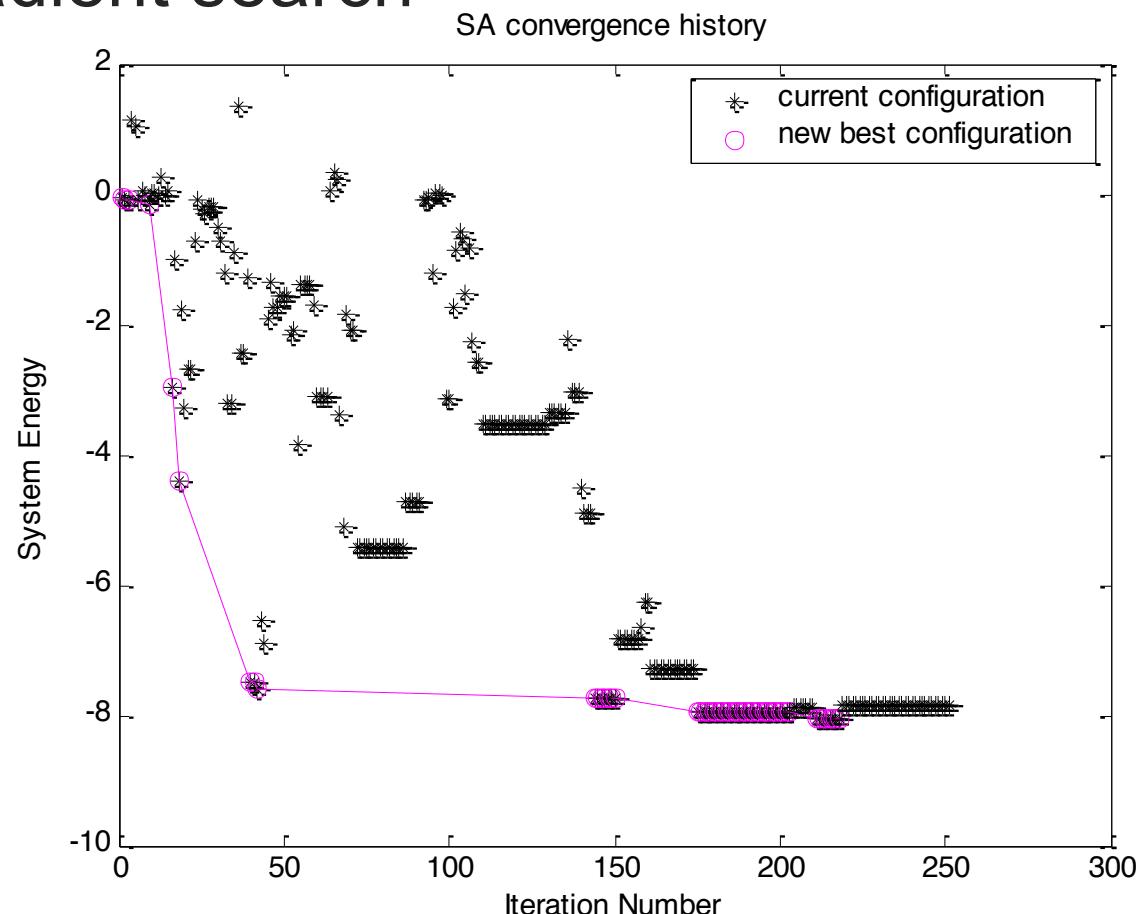
Matlab “peaks” function

- Difficult due to plateau at $z=0$, local maxima
- SAdemo1
- $x_0=[-2,-2]$
- Optimum at
- $x^*=[0.012, 1.524]$
- $z^*=8.0484$



“peaks” convergence

- Initially ~ nearly random search
- Later ~ gradient search



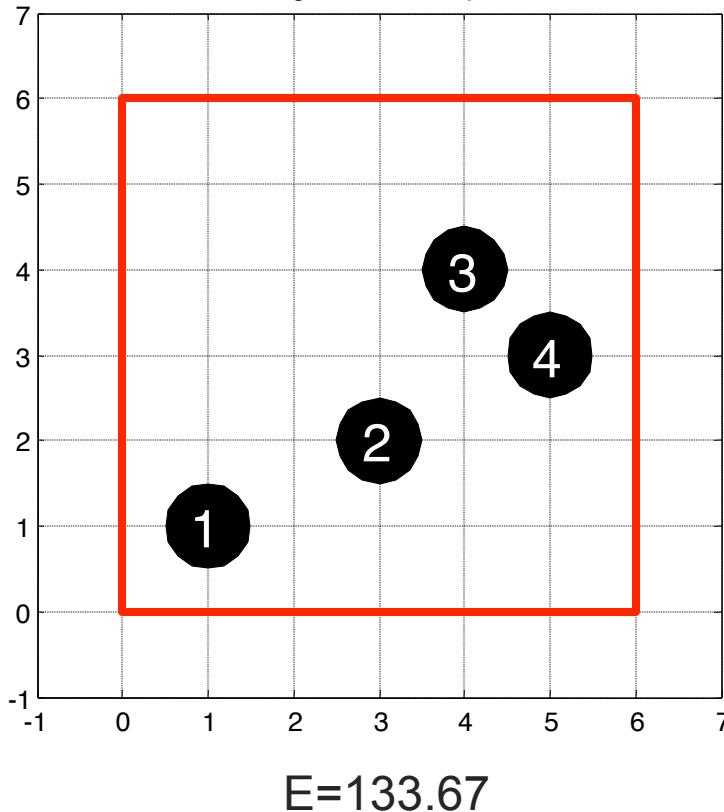
The Analogy

- **Statistical Mechanics:** The behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature.
- **Combinatorial Optimization:** Finding the minimum of a given function depending on many variables.
- **Analogy:** If a liquid material **cools and anneals too quickly**, then the material will solidify into a **sub-optimal** configuration. If the liquid material **cools slowly**, the crystals within the material will solidify **optimally into a state of minimum energy** (i.e. ground state).
 - This ground state corresponds to the minimum of the cost function in an optimization problem.

Sample Atom Configuration

Original Configuration

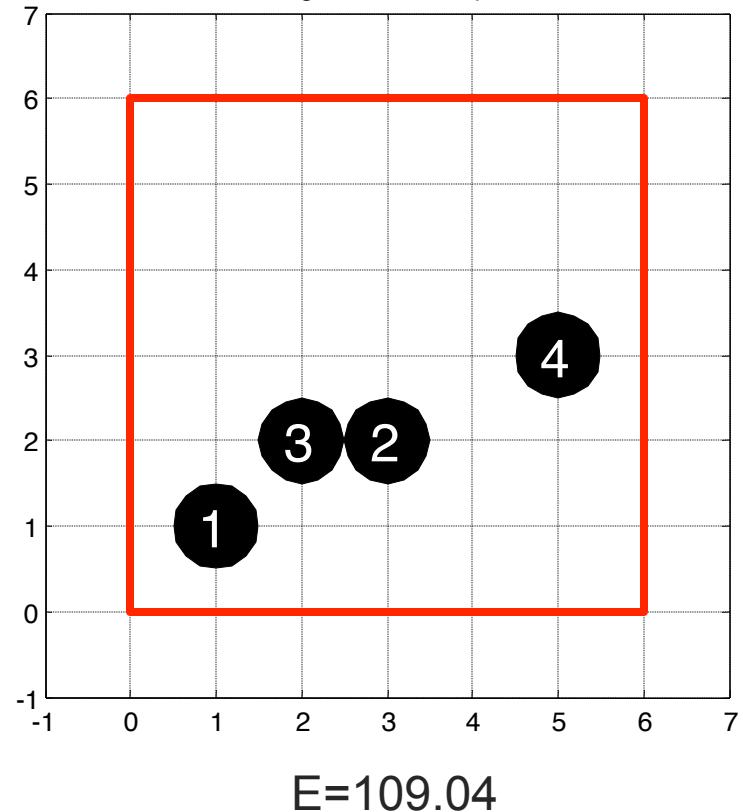
Atom Configuration - Sample Problem



Energy of original (configuration)

Perturbed Configuration

Atom Configuration - Sample Problem



Perturbing = move a random atom to a new random (unoccupied) slot

Configurations

- Mathematically describe a configuration
 - Specify coordinates of each “atom”

$$\{r_i\} \text{ with } i=1,\dots,4 \quad r_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}, r_2 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}, r_3 = \begin{bmatrix} 4 \\ 4 \end{bmatrix}, r_4 = \begin{bmatrix} 5 \\ 3 \end{bmatrix}$$

- Specify a slot for each atom

$$\{r_i\} \text{ with } i=1,\dots,4 \quad R = [1 \quad 12 \quad 19 \quad 23]$$

Energy of a state

- Each state (configuration) has an energy level associated with it

$$H(q, \dot{q}, t) = \sum_i \dot{q}_i p_i - L(q, \dot{q}, t)$$

Hamiltonian

$$H = T + V = E_{tot}$$

↑ ↑

kinetic potential

Energy

Energy of configuration
=

Objective function value of design

Energy sample problem

- Define energy function for “atom” sample problem

$$E_i = \underbrace{mgy_i}_{\text{potential energy}} + \underbrace{\sum_{j \neq i}^N \left((x_i - x_j)^2 + (y_i - y_j)^2 \right)^{\frac{1}{2}}}_{\text{kinetic energy}}$$

$$E(R) = \sum_{i=1}^N E_i(r_i)$$

Absolute and relative position of each atom contributes to Energy

Compute Energy of Configuration

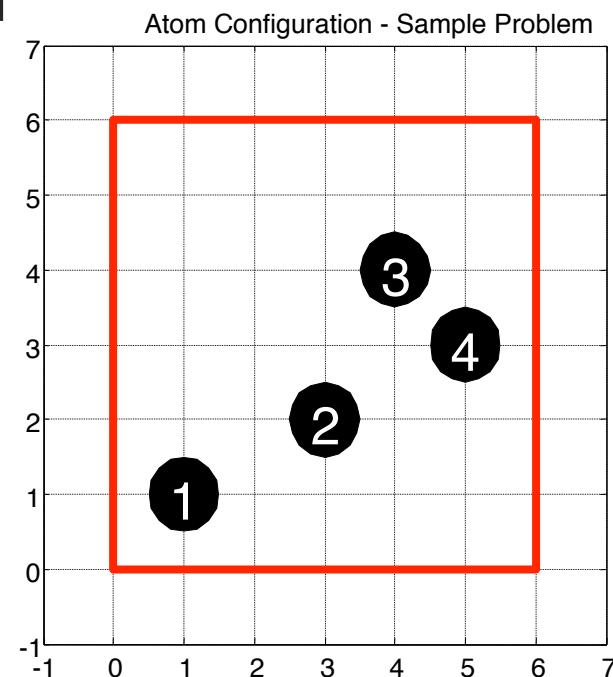
- Energy of initial configuration

$$E_1 = 1 \cdot 10 \cdot 1 + \sqrt{5} + \sqrt{18} + \sqrt{20} = 20.95$$

$$E_2 = 1 \cdot 10 \cdot 2 + \sqrt{5} + \sqrt{5} + \sqrt{5} = 26.71$$

$$E_3 = 1 \cdot 10 \cdot 4 + \sqrt{18} + \sqrt{5} + \sqrt{2} = 47.89$$

$$E_4 = 1 \cdot 10 \cdot 3 + \sqrt{20} + \sqrt{5} + \sqrt{2} = 38.12$$



Total Energy Configuration A: $E(\{r_A\})= 133.67$

Boltzmann Probability

$$N_R = \frac{P!}{(P - N)!}$$

Number of configurations
P=# of slots=25
N=# of atoms =4 } N_R=6,375,600

What is the likelihood that a particular configuration will exist in a large ensemble of configurations?

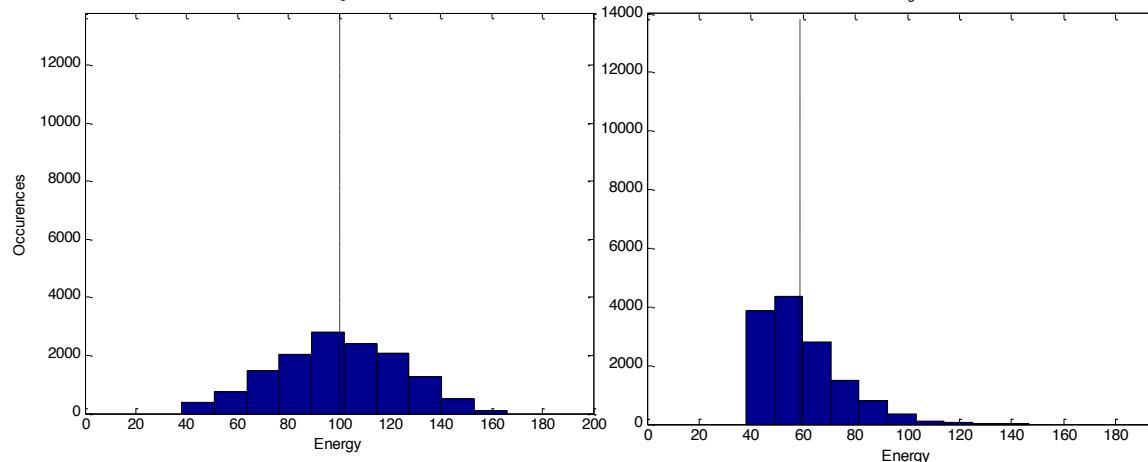
$$P(\{r\}) = \exp\left[\frac{-E(\{r\})}{k_B T}\right]$$

Boltzmann probability depends on energy and temperature

Boltzmann Collapse at Low T

T=100

T=100 - E_{avg}=100.1184

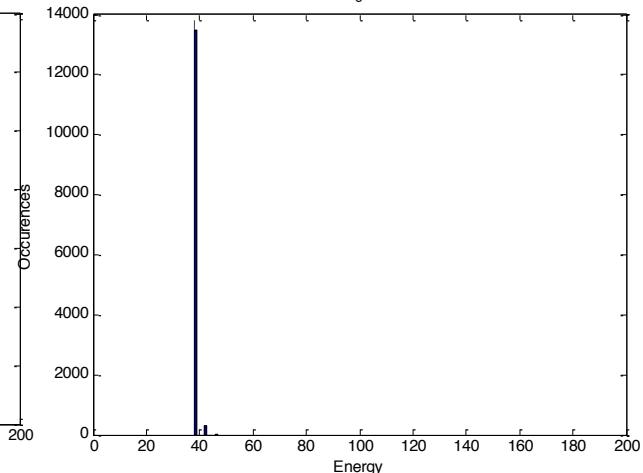


T=10

T=10 - E_{avg}=58.9245

T=1

T=1 - E_{avg}=38.1017



T high

T low

Boltzmann Distribution collapses to the lowest energy state(s) in the limit of low temperature



Basis of search by Simulated Annealing

Partition Function Z

- Ensemble of configurations can be described statistically
- Partition function, Z , generates the ensemble average

$$Z = \sum_{i=1}^{N_R} \exp\left(\frac{-E_i}{k_B T}\right)$$

Initially (at $T \gg 0$) equal to the number of possible configurations

Free Energy

$$E_{avg} = \bar{E}(T) = \frac{1}{N_T} \sum_{i=1}^{N_T} E_i(T) \leftarrow \text{Average Energy of all Configurations in an ensemble}$$

$$F(T) = -k_B T \ln Z = \bar{E}(T) - TS$$

$$E_{avg} = \bar{E}(T) = \frac{\sum_{i=1}^{N_R} \exp\left(-\frac{E_i(T)}{k_B T}\right) E_i(T)}{Z} = \frac{-d \ln Z}{d(1/k_B T)}$$

Relates average Energy at T with Entropy S

Specific Heat and Entropy

- Specific Heat

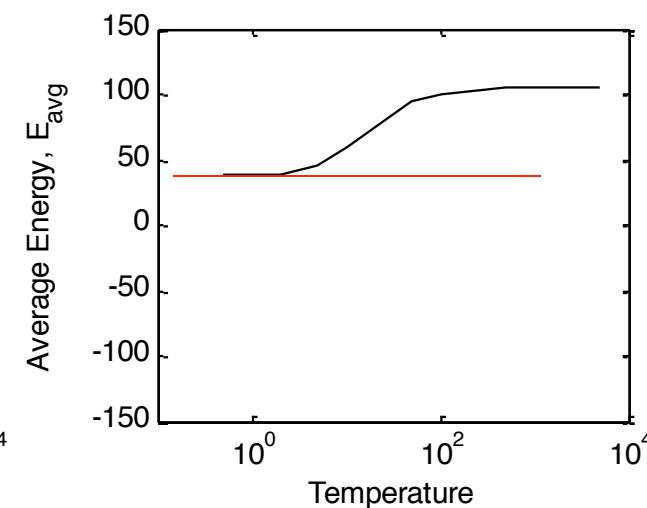
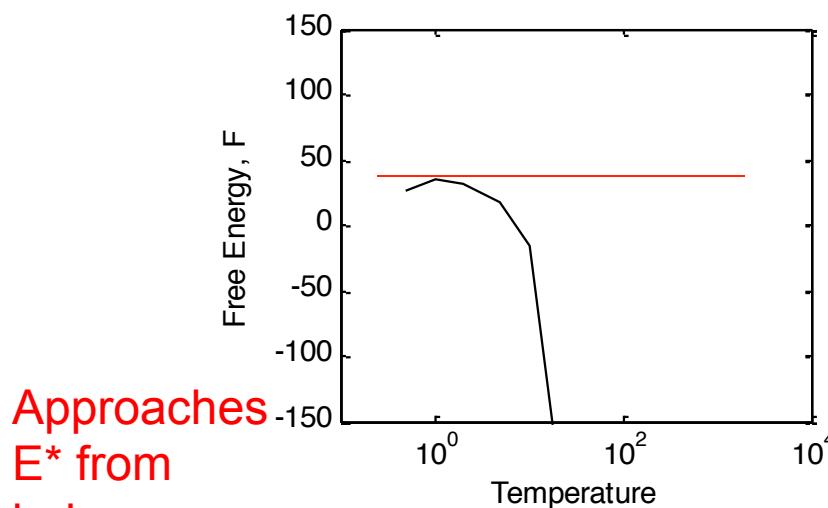
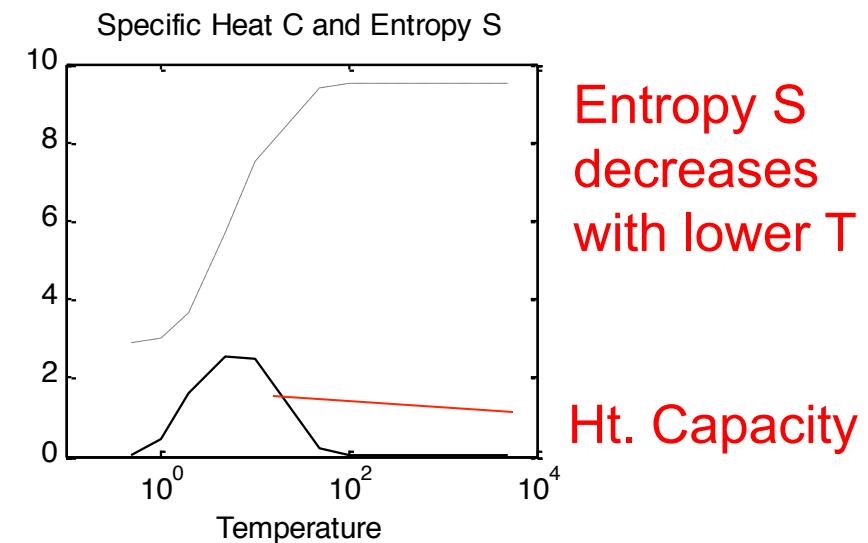
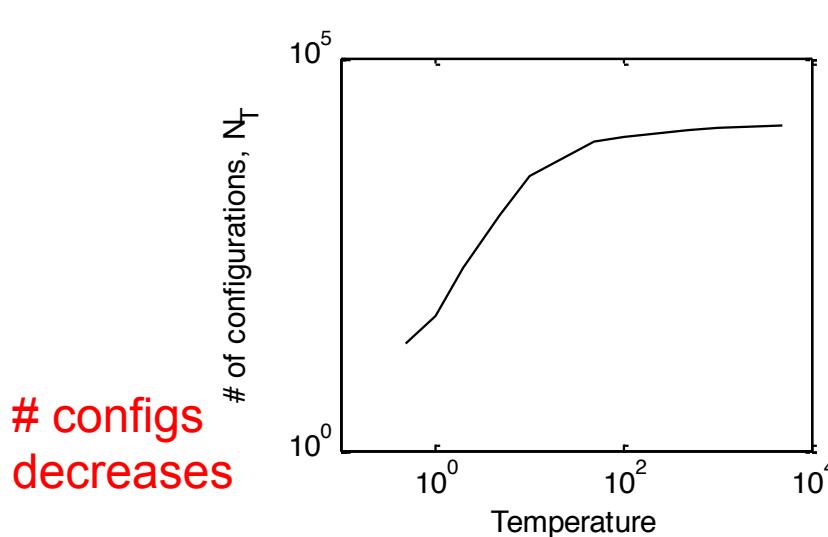
$$C(T) = \frac{d\bar{E}(T)}{dT} = \frac{\left[\frac{1}{N_R} \sum_{i=1}^{N_R} E_i^2(T) - \bar{E}(T)^2 \right]}{k_B T^2} = \frac{\bar{E}^2(T) - \bar{E}(T)^2}{k_B T^2}$$

- Entropy

$$S(T) = S(T_1) - \int_T^{T_1} \frac{C(T)}{T} dT \quad \frac{dS(T)}{dT} = \frac{C(T)}{T}$$

Entropy is ~ equal to $\ln(\# \text{ of unique configurations})$

Low Temperature Statistics



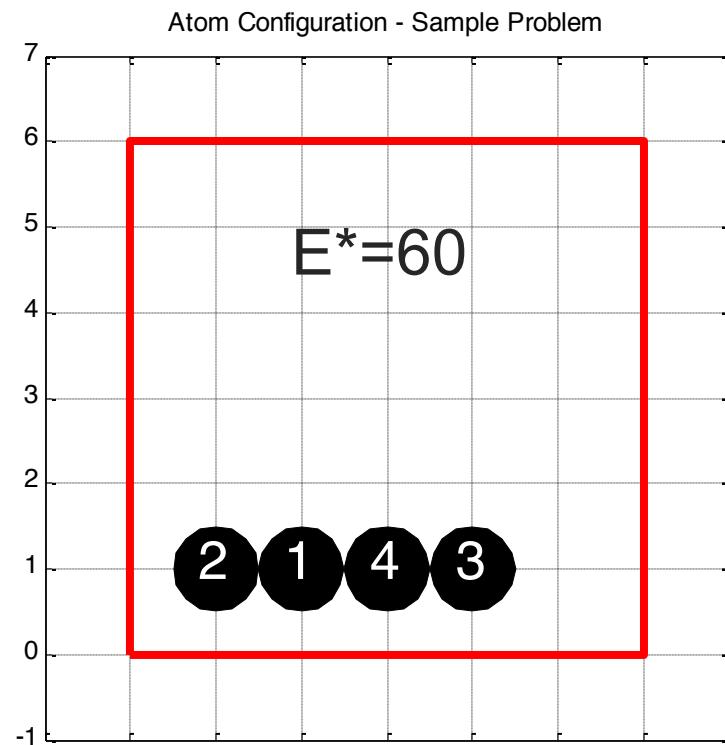
Approaches
E* from
below

Approaches
E* from
above

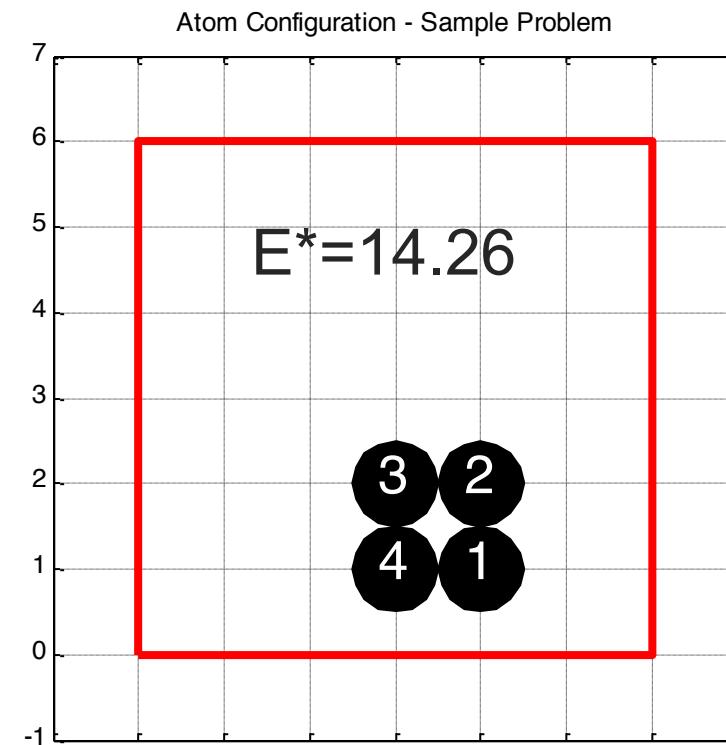
Minimum Energy Configurations

- Sample Atom Placement Problem

Uniqueness?



Strong gravity $g=10$



Low gravity $g=0.1$

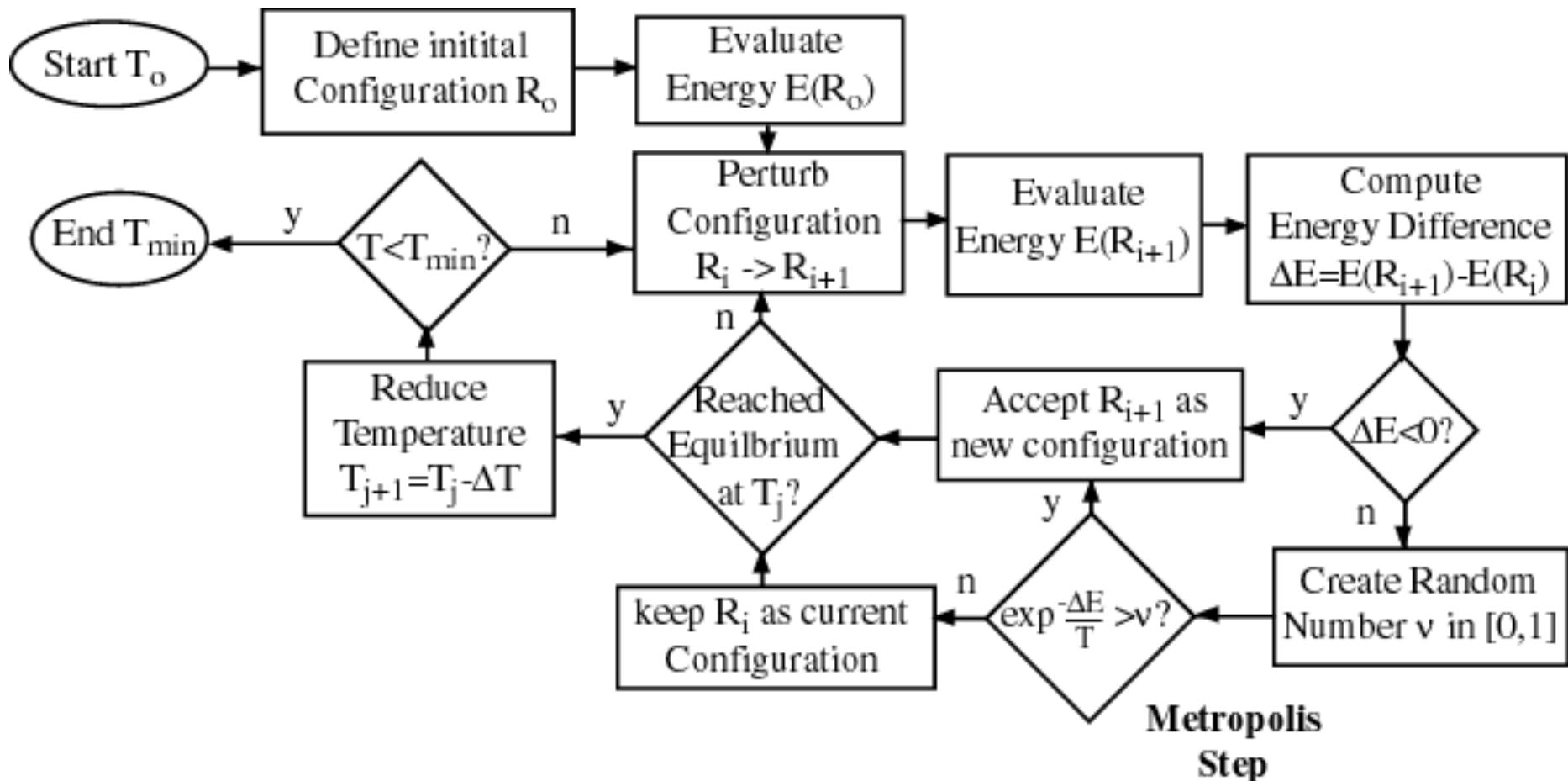
Dilemma

- Cannot compute energy of all configurations !
 - Design space often too large
 - Computation time for a single function evaluation can be large
- Use Metropolis Algorithm, at successively lower temperatures to find low energy states
 - Metropolis: Simulate behavior of a set of atoms in thermal equilibrium (1953)
 - Probability of a configuration existing at T → Boltzmann Probability $P(r,T) = \exp(-E(r)/T)$

The SA Algorithm

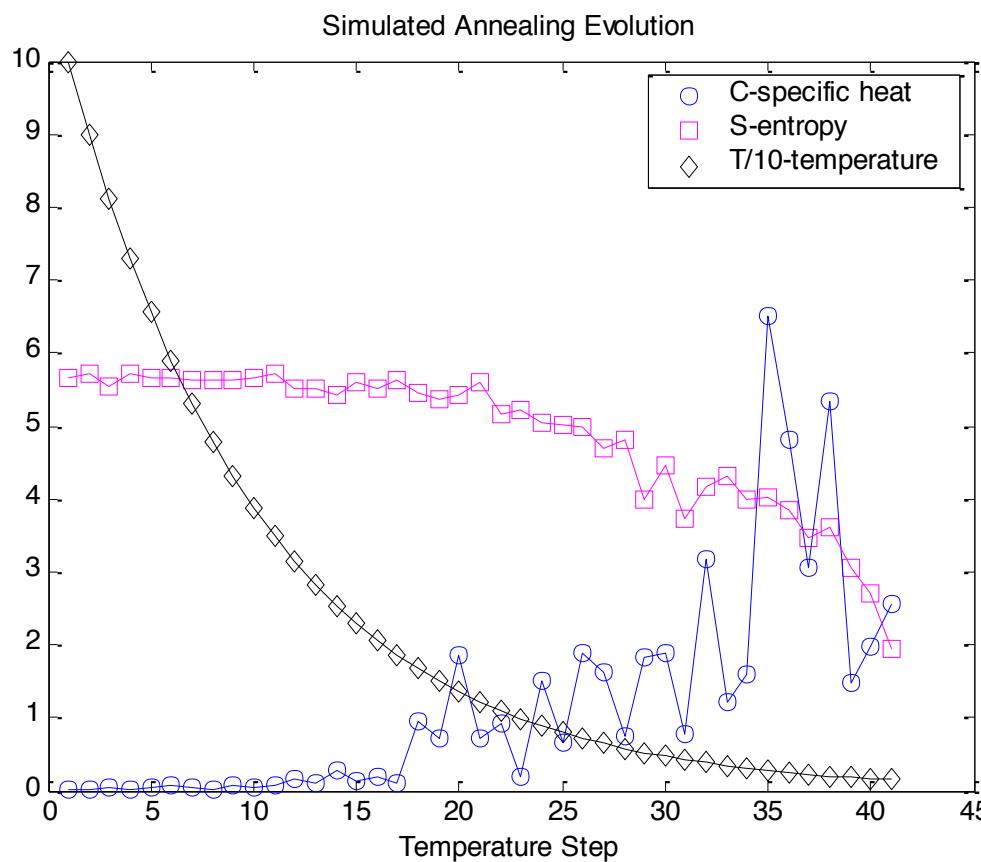
- **Terminology:**
 - X (or R or Γ) = Design Vector (i.e. Design, Architecture, Configuration)
 - E = System Energy (i.e. Objective Function Value)
 - T = System Temperature
 - Δ = Difference in System Energy Between Two Design Vectors
- **The Simulated Annealing Algorithm**
 - 1) Choose a random X_i , select the initial system temperature, and specify the cooling (i.e., annealing) schedule
 - 2) Evaluate $E(X_i)$ using a simulation model
 - 3) Perturb X_i to obtain a neighboring Design Vector (X_{i+1})
 - 4) Evaluate $E(X_{i+1})$ using a simulation model
 - 5) If $E(X_{i+1}) < E(X_i)$, X_{i+1} is the new current solution
 - 6) If $E(X_{i+1}) > E(X_i)$, then accept X_{i+1} as the new current solution with a probability $e^{(-\Delta/T)}$ where $\Delta = E(X_{i+1}) - E(X_i)$.
 - 7) Reduce the system temperature according to the cooling schedule.
 - 8) Terminate the algorithm.

SA Block Diagram



Exponential Cooling

- Typically $(T_1/T_o) \sim 0.7-0.9$



$$T_{k+1} = \left(\frac{T_1}{T_o} \right)^k T_k$$

Can also do linear
or step-wise cooling
...
But exponential cooling
often works best.

Key Ingredients for SA

1. A concise description of a **configuration** (architecture, design, topology) of the system (**Design Vector**).
2. A random generator of **rearrangements** of the elements in a configuration (**Neighborhoods**). This generator encapsulates rules so as to generate only valid configurations. Perturbation function.
3. A quantitative **objective function** containing the trade-offs that have to be made (**Simulation Model and Output Metric(s)**). Surrogate for system energy.
4. An **annealing schedule** of the temperatures and/or the length of times for which the system is to be evolved.

Traveling Salesman Problem

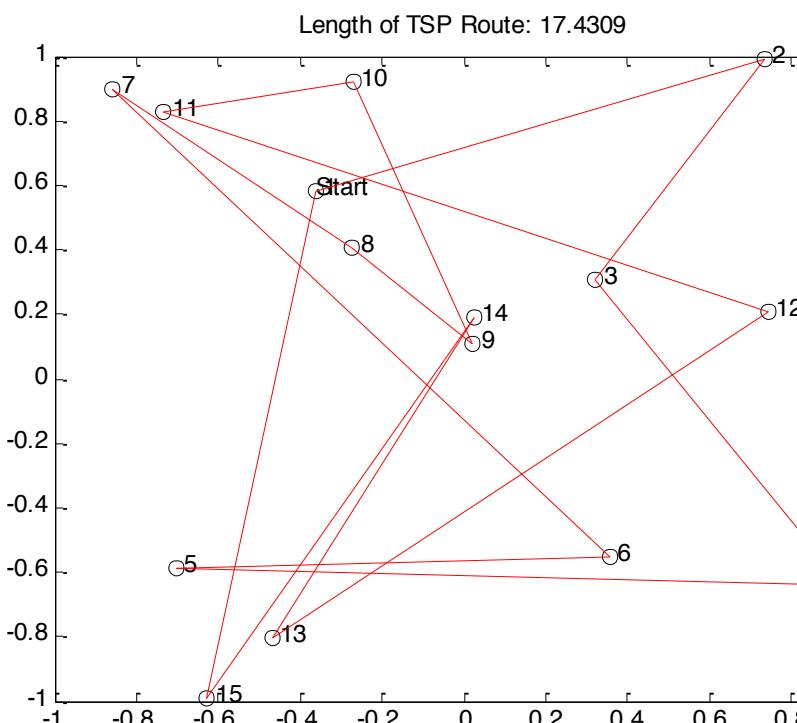
- N cities arranged randomly on $[-1, 1]$
- Choose $N=15$
- SAdemo2.m
- Minimize “cost” of route (length, time,...)
- Visit each city once, return to start city

$$l(R) = \underbrace{\sum_{i=1}^N \sqrt{\sum_{j=1}^2 (x_j(R_{i+1}) - x_j(R_i))^2}}_{\text{visit } N \text{ cities}} + \sqrt{\sum_{j=1}^2 (x_j(R_1) - x_j(R_N))^2}$$

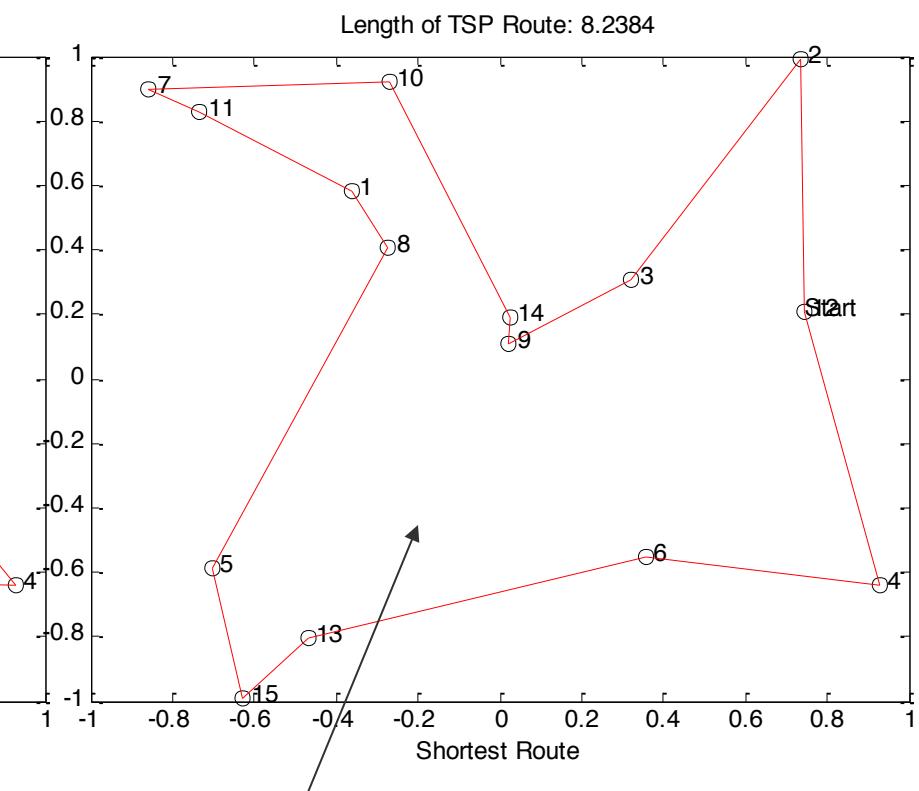
return home to first city

TSP Problem (Cont.)

Initial (Random) Route
Length: 17.43



Final (Optimized) Route
Length: 8.24



Result with SA

Structural Optimization

- Define:
 - Design Domain
 - Boundary Conditions
 - Loads
 - Mass constraint
- Subdivide domain
 - N x M design “cells”
 - Cell density $\rho=1$ or $\rho=0$
- Where to put material to minimize compliance?

$$\text{find } \rho_i \quad i = 1, \dots, N$$

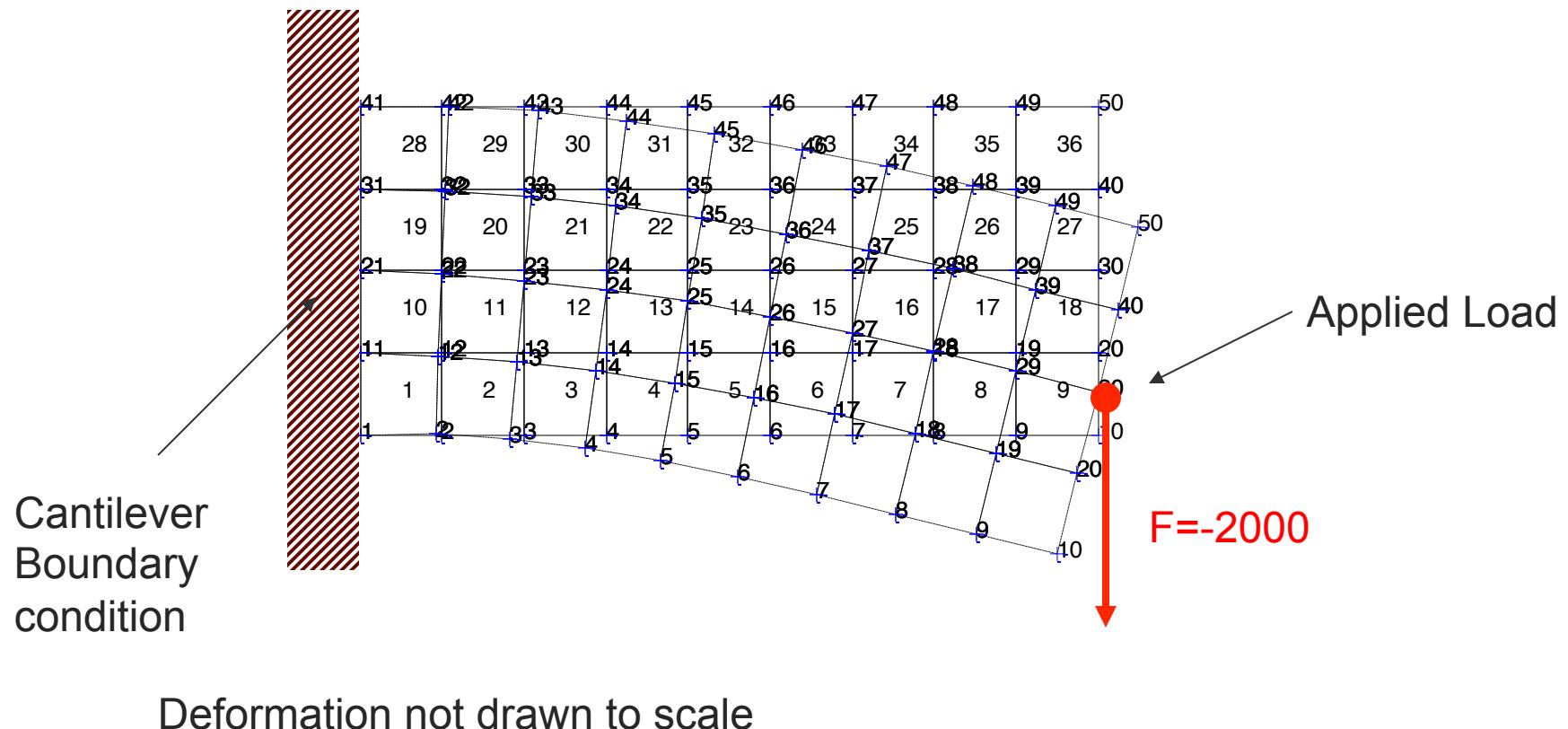
$$\min C = f^T u(\rho_i)$$

$$\text{s.t. } u = K^{-1}f$$

$$\text{s.t. } \sum_{i=1}^N V_i \rho_i \leq m_{\max}$$

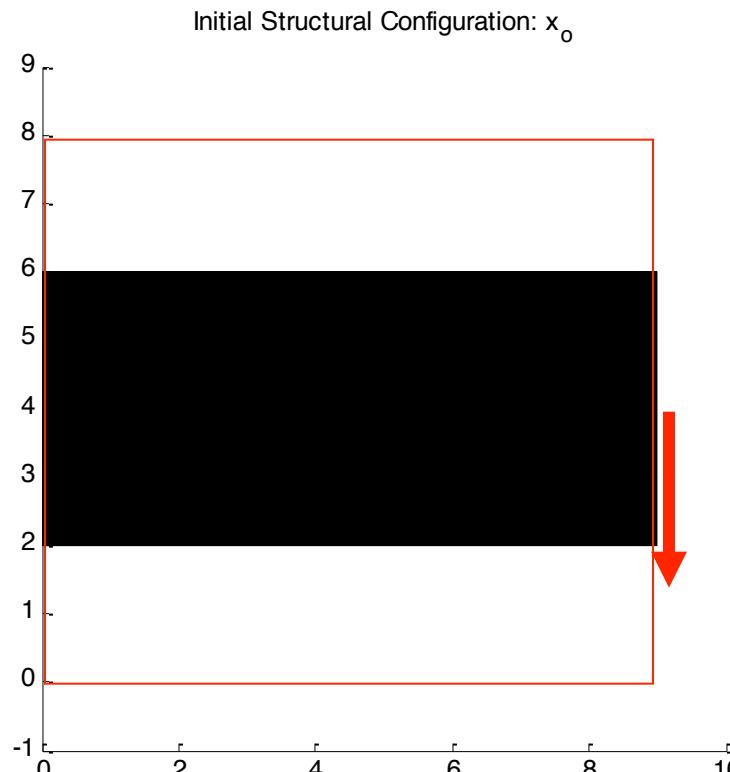
Structural Topology Optimization (II)

“Energy” = strain energy = compliance
Computed via Finite Element Analysis



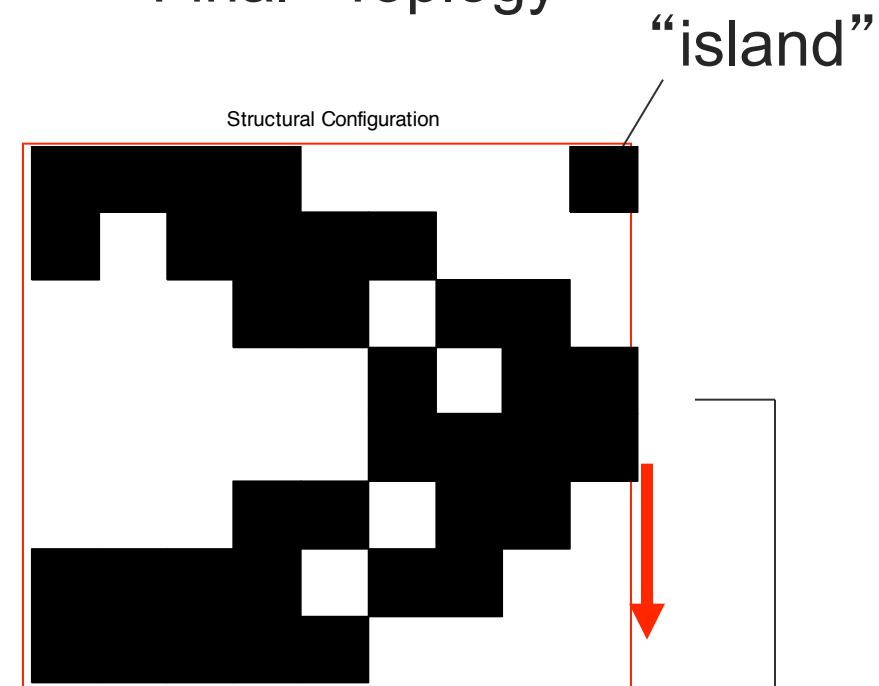
Structural Topology Optimization

Initial Structural Toplogy



Compliance C=593.0

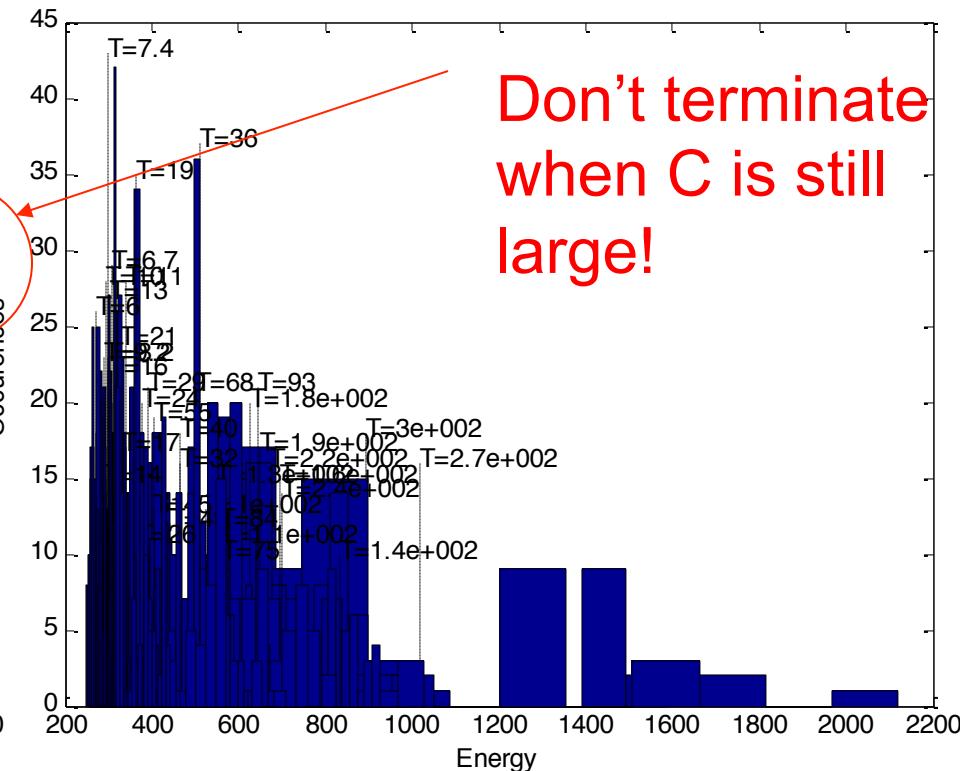
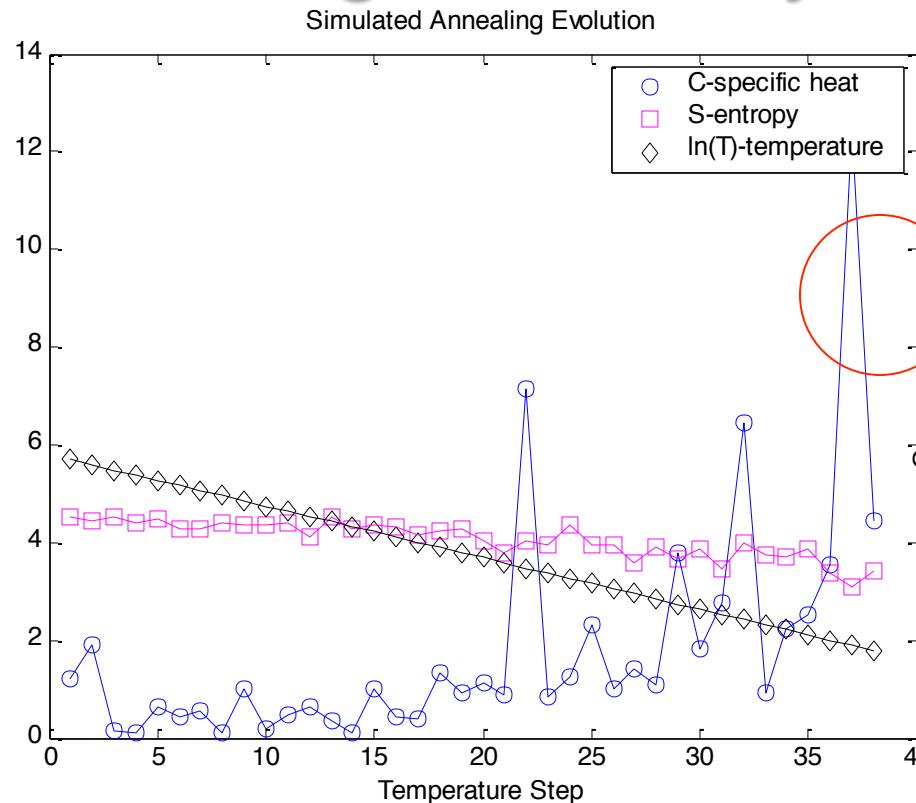
“Final” Toplogy



Compliance C=236.68

Not “optimal” – often
need post-processing

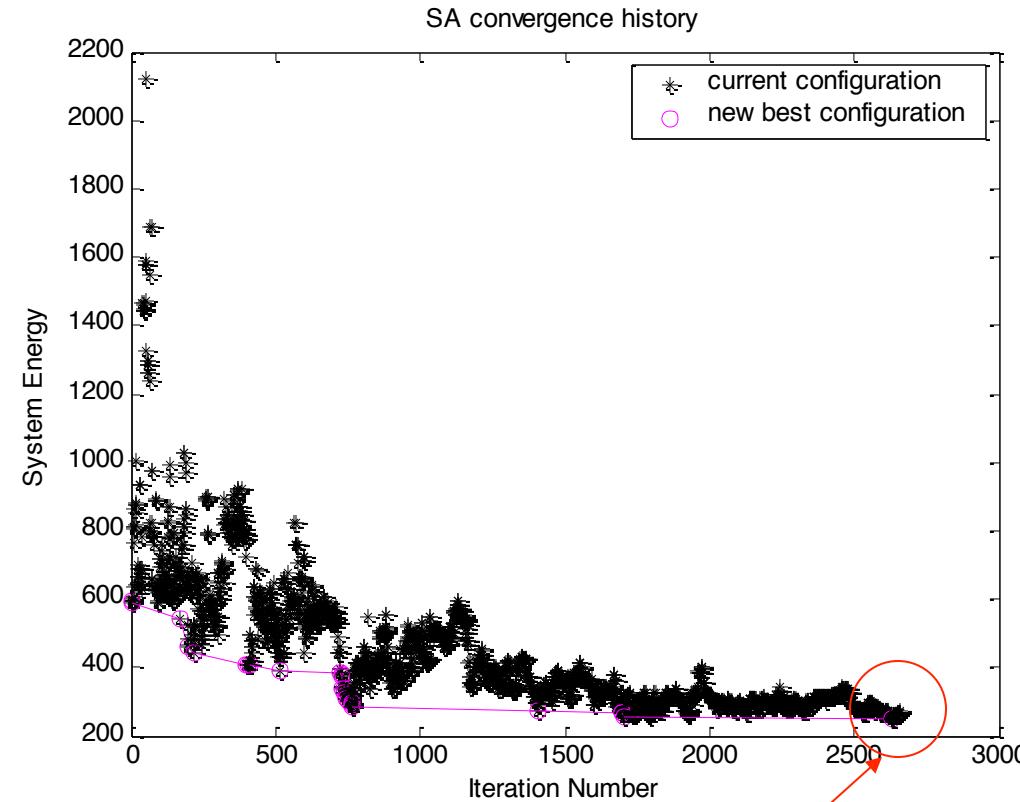
Structural Optimization – Convergence Analysis



Boltzmann Distributions

Evolution of
- Entropy, Temperature, Specific Heat

Premature Termination



Indicator: Best Configuration found only shortly before Simulated Annealing terminated.

Summary: Steps of SA

- **The Simulated Annealing Algorithm**

- 1) Choose a random X_i , select the initial system temperature, and outline the cooling (ie. annealing) schedule
- 2) Evaluate $E(X_i)$ using a simulation model
- 3) Perturb X_i to obtain a neighboring Design Vector (X_{i+1})
- 4) Evaluate $E(X_{i+1})$ using a simulation model
- 5) If $E(X_{i+1}) < E(X_i)$, X_{i+1} is the new current solution
- 6) If $E(X_{i+1}) > E(X_i)$, then accept X_{i+1} as the new current solution with a probability $e^{(-\Delta/T)}$ where $\Delta = E(X_{i+1}) - E(X_i)$.
- 7) Reduce the system temperature according to the cooling schedule.
- 8) Terminate the algorithm.

Research Research in SA

- Alternative Cooling Schedules and Termination criteria
- Adaptive Simulated Annealing (ASA) – determines its own cooling schedule
- Hybridization with other Heuristic Search Methods (GA, Tabu Search ...)
- Multiobjective Optimization with SA
- Parallel Tempering

References

- Cerny, V., "Thermodynamical Approach to the Traveling Salesman Problem: An Efficient Simulation Algorithm", *J. Opt. Theory Appl.*, 45, 1, 41-51, 1985 .
- Cahanim B. E., Hewitt J. N., and de Weck O.L., "The Design of Radio Telescope Array Configurations using Multiobjective Optimization: Imaging Performance versus Cable Length", *The Astrophysical Journal, Supplement Series*, 154, 705-719, October 2004
- Jilla, C.D., and Miller, D.W., "Assessing the Performance of a Heuristic Simulated Annealing Algorithm for the Design of Distributed Satellite Systems," *Acta Astronautica*, Vol. 48, No. 5-12, 2001, pp. 529-543.
- **Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by Simulated Annealing," *Science, Volume 220, Number 4598, 13 May 1983, pp. 671-680.***
- Metropolis,N., A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, "Equation of State Calculations by Fast Computing Machines", *J. Chem. Phys.*,21, 6, 1087-1092, 1953.