Lecture 10: Multidisciplinary System Analysis and Design Optimization (MSADO)

An Introduction to Genetic Algorithms

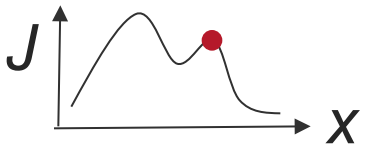
March 2, 2015

Prof. Douglas Allaire

Heuristic Search Techniques

Main Motivation for Heuristic Techniques:

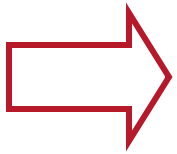
(1) To deal with local optima and not get trapped in them



(2) To allow optimization for systems, where the design variables are not only continuous, but discrete (categorical), integer or even Boolean

$x_i \notin \Re$

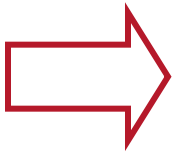
x_i ={1,2,3,4,5}, x_i ={ ‘A’ ,’ B’ ,’ C’ } x_i ={true, false}



These techniques do not guarantee that global optimum can be found. Generally Karush-Kuhn-Tucker conditions do not apply.

Principal Heuristic Algorithms

- Genetic Algorithms (Holland – 1975)
  - Inspired by genetics and natural selection – max fitness
- Simulated Annealing (Kirkpatrick – 1983)
  - Inspired by statistical mechanics– min energy
- Particle Swarm Optimization (Eberhart Kennedy - 1995)
  - Inspired by the social behavior of swarms of insects or flocks of birds – max "food"



These techniques all use a combination of randomness and heuristic "rules" to guide the search for global maxima or minima

Today: Genetic Algorithms

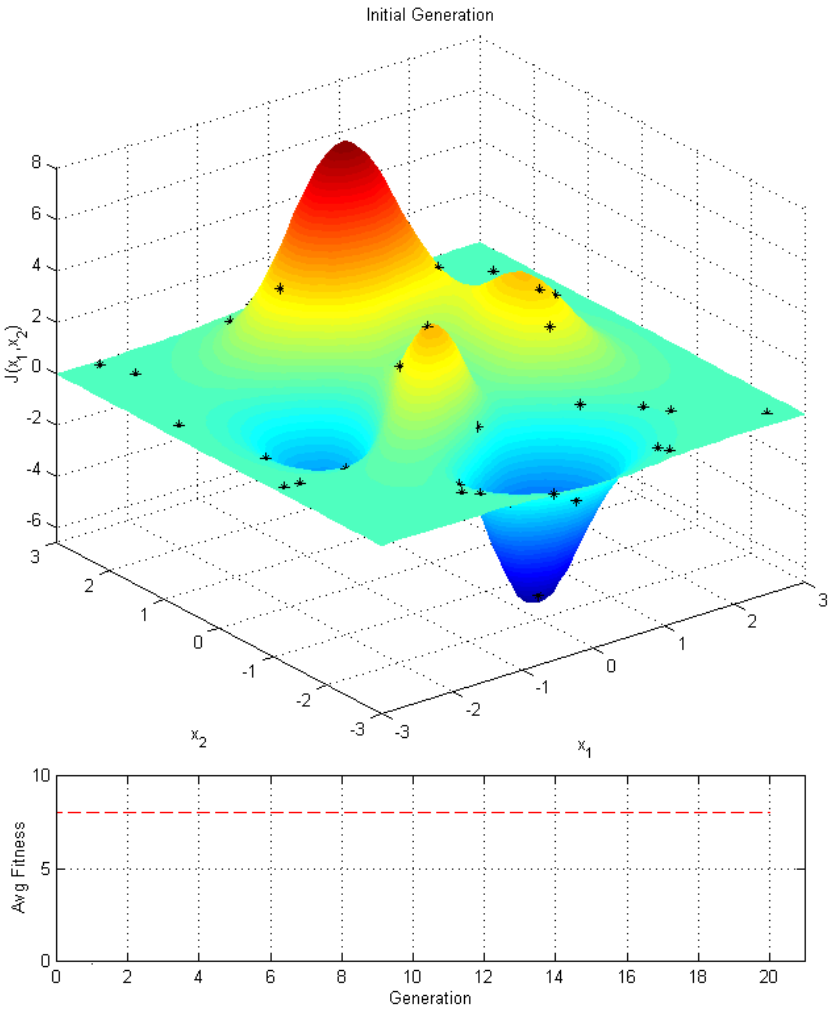
- Genetics and Natural Selection
- A simple genetic algorithm (SGA)
- "The Genetic Algorithm Game"
- Encoding - Decoding (Representation)
- Fitness Function - Selection
- Crossover – Insertion - Termination

Premise of GAs

- Natural Selection is a very successful organizing principle for optimizing individuals and populations of individuals
- If we can mimic natural selection, then we will be able to optimize more successfully
- A possible design of a system – as represented by its design vector **x** - can be considered as an individual who is fighting for survival within a larger population.
- Only the fittest survive – Fitness is assessed via objective function *J*.

Matlab GA demo ("peaks")

- Maximize "peaks" function
- Population size: 40
- Generations: 20
- Mutation Rate: 0.002



Natural Selection

Charles Darwin (1809-1882)

Controversial and very influential book (1859)
On the origin of species by means of natural selection, or the preservation of favored races in the struggle for life

Observations:

- Species are continually developing
- Homo sapiens sapiens and apes have common ancestors
- Variations between species are enormous
- Huge potential for production of offspring, but only a small/moderate percentage survives to adulthood

⇨ Evolution = natural selection of inheritable variations

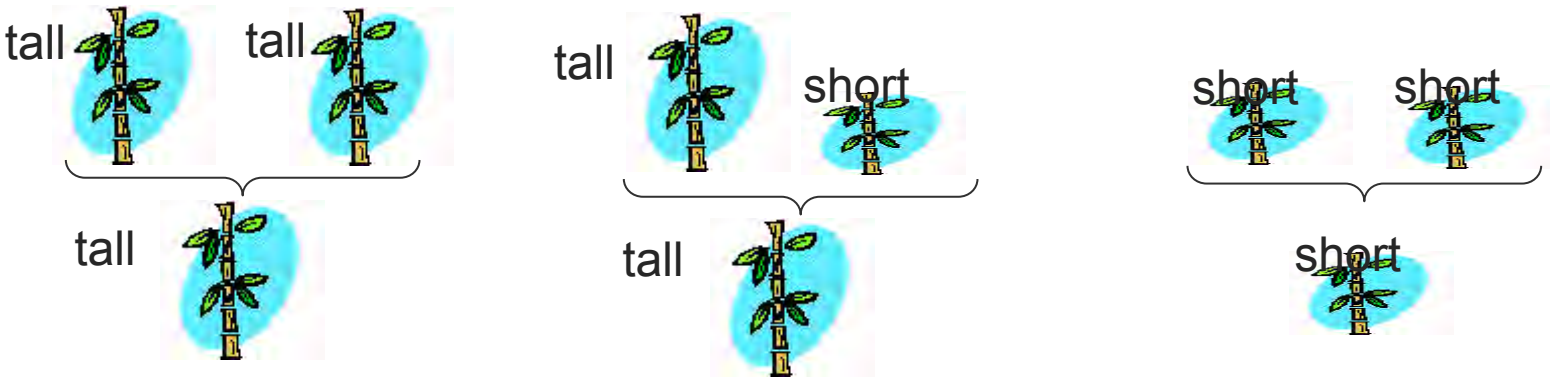
Inheritance of Characteristics

Gregor Mendel (1822-1884)

Investigated the inheritance of characteristics (“traits”)

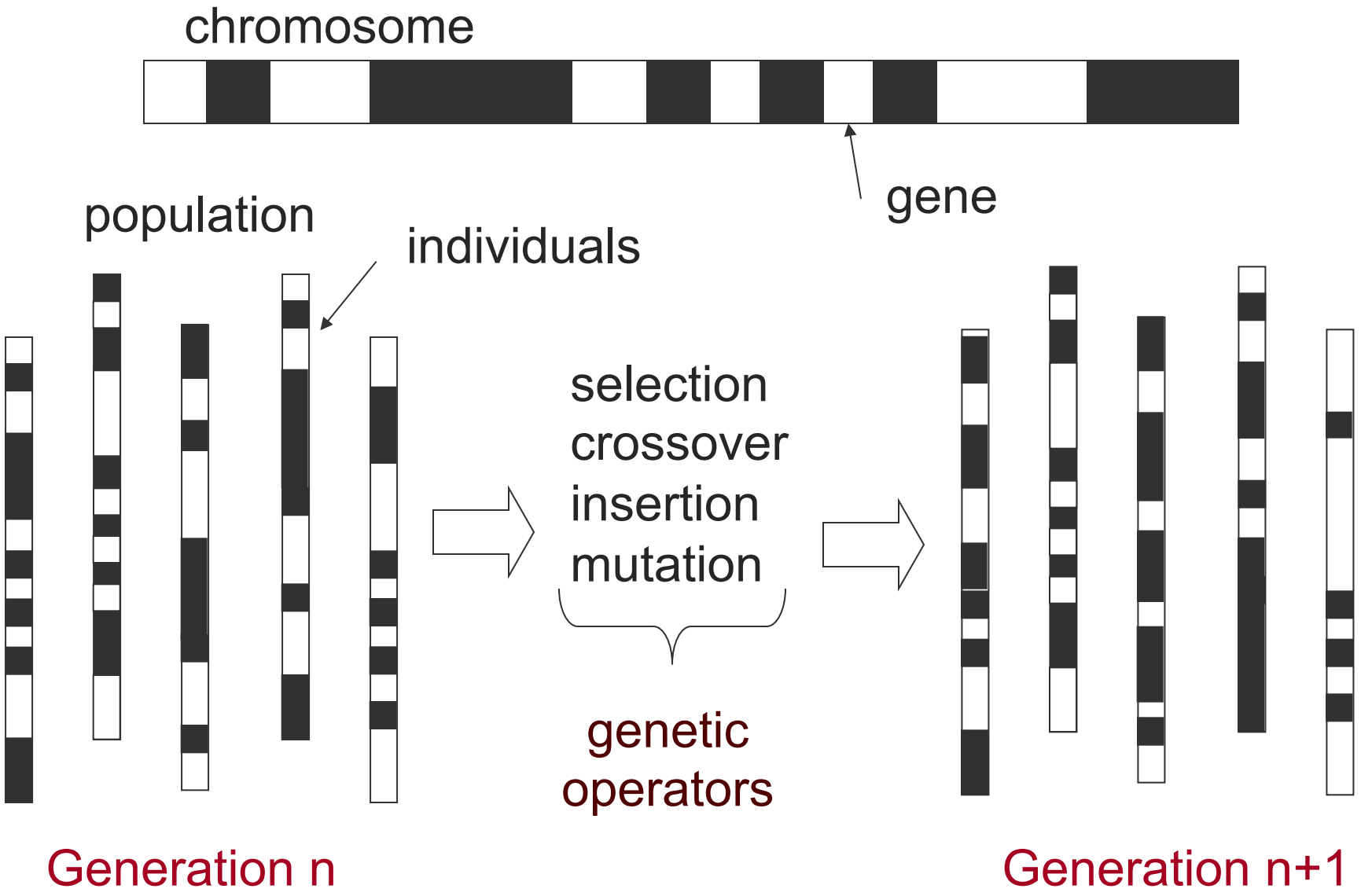
Conducted extensive experiments with pea plants

Examined hybrids from different strains of plant

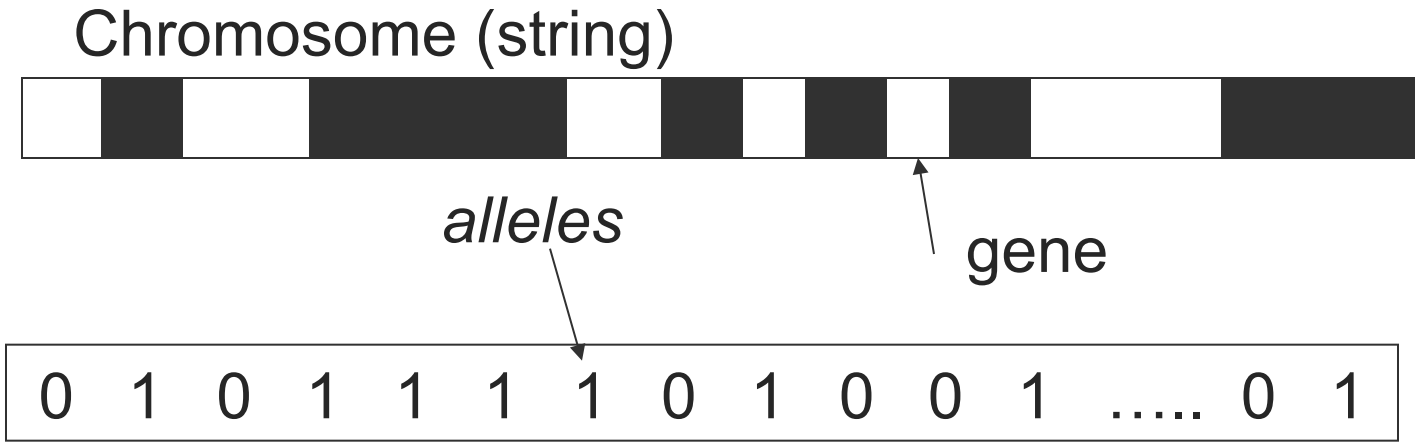


Character (gene) for tallness is dominant
Character (gene) for shortness is recessive

GA Terminology



Chromosomes

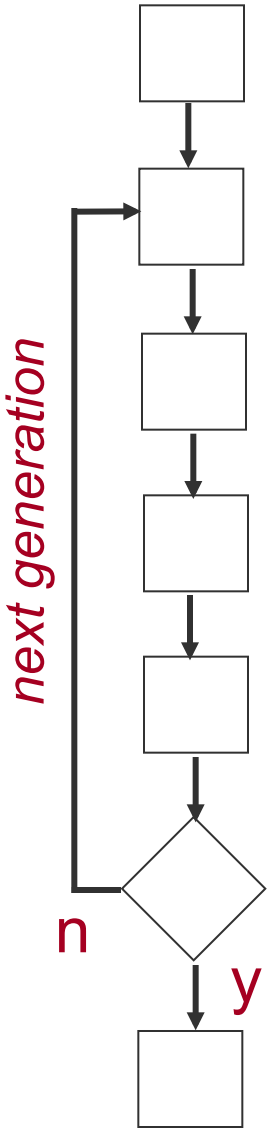


Each chromosome represents a solution, often using strings of 0's and 1's. Each bit typically corresponds to a gene. This is called binary encoding.

The values for a given gene are the alleles.

A chromosome in isolation is meaningless - need decoding of the chromosome into phenotypic values

GA over several generations



Initialize Population (initialization)

Select individual for mating (selection)

Mate individuals and produce children (crossover)

Mutate children (mutation)

Insert children into population (insertion)

Are stopping criteria satisfied ?

Finish

Ref: Goldberg (1989)

The GA Game

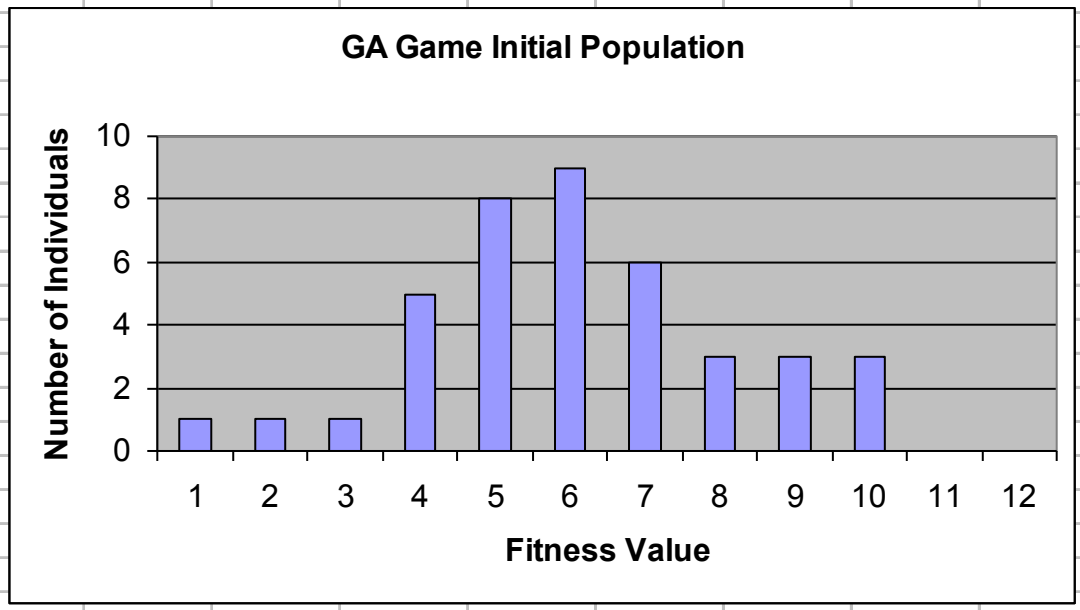
Ca. 15 minutes

Population size: N=40
Mean Fitness: F=6.075

Generation 1:

(Fitness F = total number of 1's in chromosome)

1	1	1
2	1	2
3	1	3
4	5	20
5	8	40
6	9	54
7	6	42
8	3	24
9	3	27
10	3	30
11	0	0
12	0	0
	40	6.075



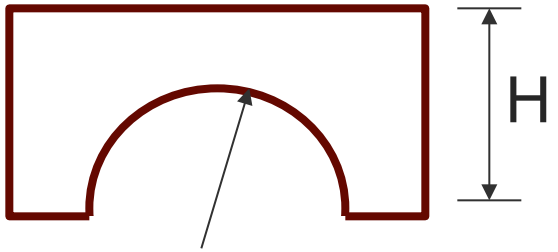
0 <= F <= 12

Goal: Maximize Number of "1"s

Creating a GA on a computer

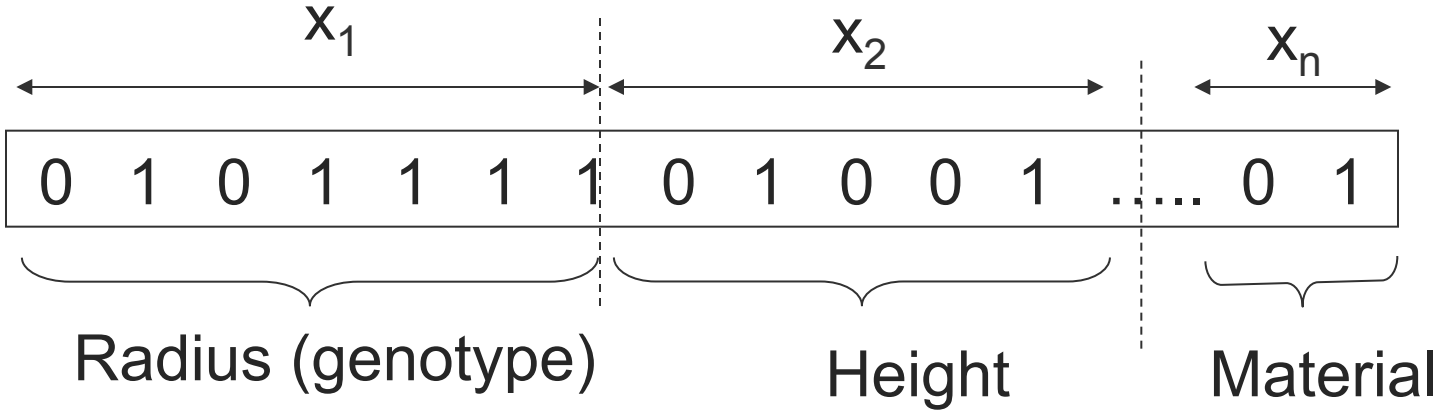
(1) define the representation (encoding-decoding)
(2) define "fitness" function *F*
    - incorporate feasibility (constraints) and objectives
(3) define the genetic operators
    - initialization, selection, crossover, mutation, insertion
(4) execute initial algorithm run
    - monitor average population fitness
    - identify best individual
(5) tune algorithm
    - adjust selection, insertion strategy, mutation rate

Encoding - Decoding

	genotype		**phenotype**
	coded domain		*decision domain*
Biology	UGCAACCGU **("DNA" blocks)**	*expression* → ← *sequencing*	 "blue eye"
Design	10010011110 **(chromosome)**	*decoding* → ← *encoding*	 Radius R=2.57 [m]

Genetic Code: (U,C,G,A are the four bases of the nucleotide building blocks of messenger-RNA): Uracil-Cytosin-Adenin-Guanin - A triplet leads to a particular aminoacid (for protein synthesis) e.g. UGG-tryptophane

Decoding



E.g., binary encoding of integers:

10100011

$(1*2^7+0*2^6+1*2^5+0*2^4+0*2^3+0*2^2+1*2^1+1*2^0)$

128 + 0 + 32 + 0 + 0 + 0 + 2 + 1 = 163

Coding and decoding MATLAB functions available:
decode.m, encode.m

Binary Encoding Issues

Number of bits dedicated to a particular design variable is very important.

Resolution depends on:
- upper and lower bounds x_{LB}, x_{UB}
- number of bits

Number of bits needed:

$$nbits = \left\lceil \frac{\ln\left(\frac{x_{UB} - x_{LB}}{\Delta x}\right)}{\ln 2} \right\rceil$$

$x \in \mathbb{R}$

Example

```
[G]=encode(137.56,50,150,8)
G = 1     1     0     1     1     1     1     1
[X]=decode(G,50,150,8);
X = 137.4510
```

So $\Delta x = (150-50)/2^8 = 0.39$

Loss in precision !!!

Other Encoding Schemes

Not all GA chromosomes are binary strings
Can use a different ALPHABET for GA coding

Most common is binary alphabet {0,1}

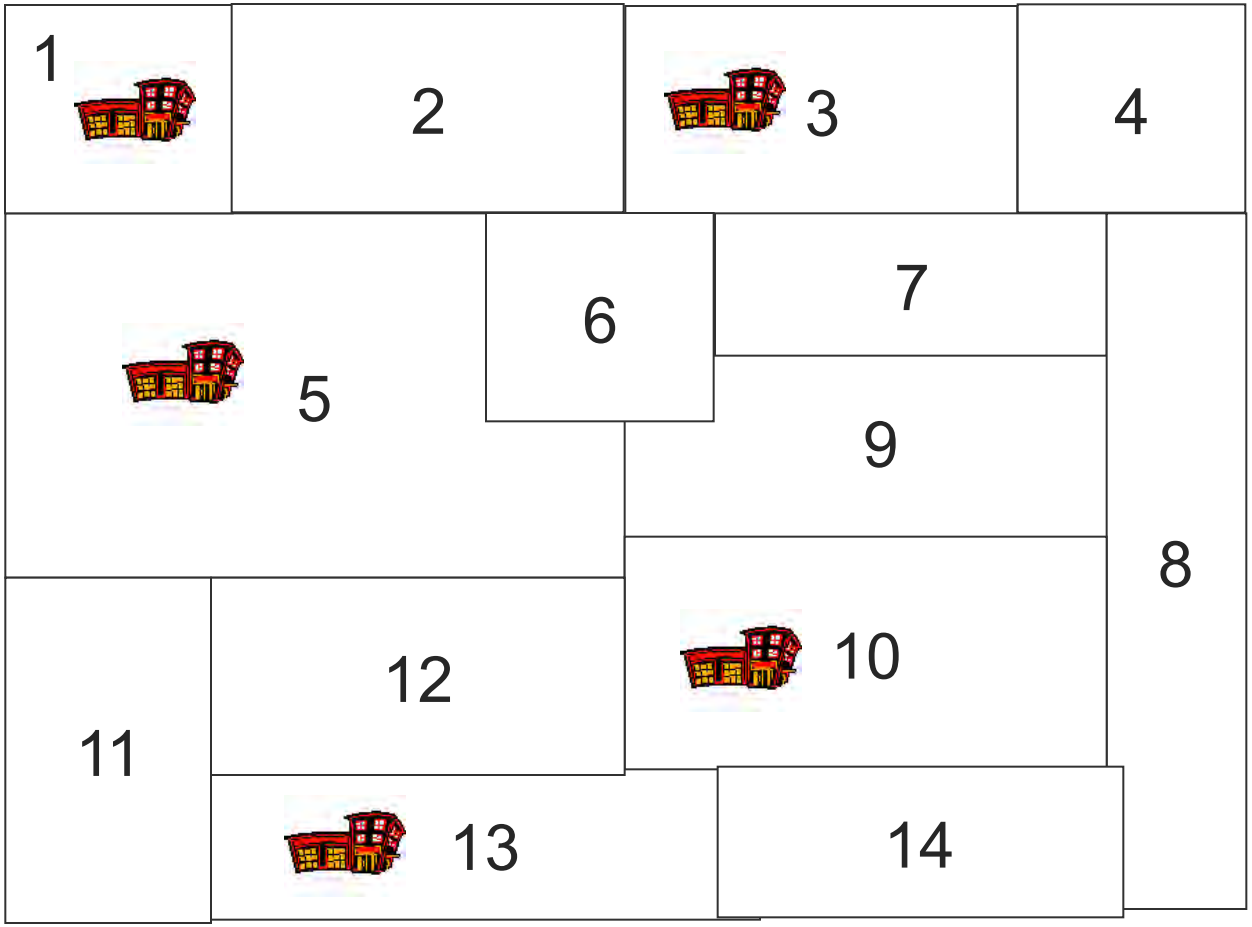
can also have

The set of symbols is the "alphabet"

- ternary: {0,1,2} {A,B,C}
- quaternary: {0,1,2,3} {T,G,C,A} => biology
- integer: {1,2,….13,….}
-real valued: {3.456 7.889 9.112}
-Hexadecimal {1,2,..,A,B,C,D,E,F}

Used for Traveling Salesman Problem

A Representation for a fire station location problem

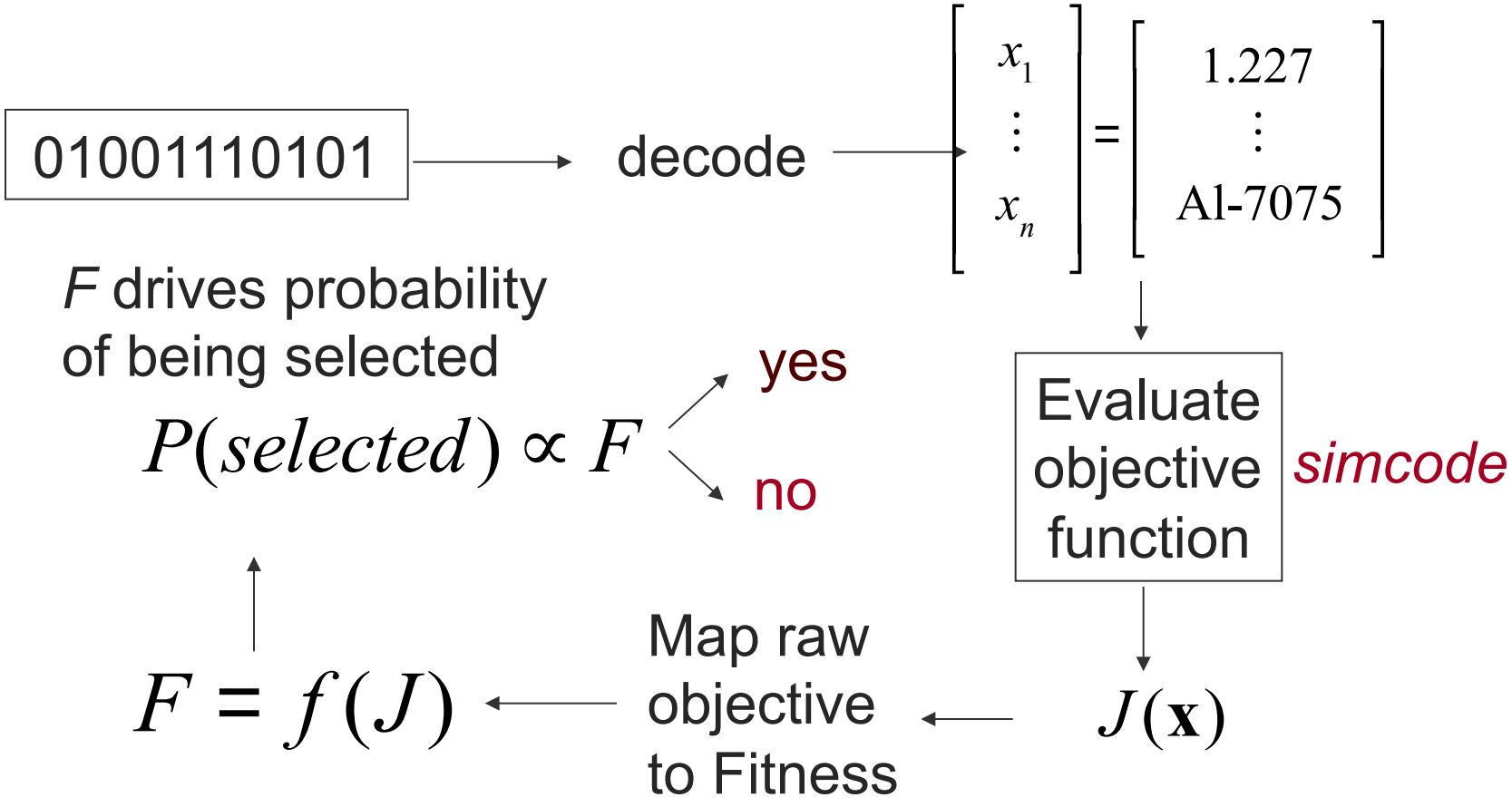


1 0 1 0 1 0 0 0 0 1 0 0 1 0

"1" represents a fire station

Fitness and Selection Probability

Typically, selection is the most important and most computationally expensive step of a GA.



Fitness Function

- Choosing the right fitness function is very important, but also quite difficult
- GAs do not have explicit "constraints"
- Constraints can be handled in different ways:
  - via the fitness function – penalty for violation
  - via the selection operator ("reject constraint violators")
  - implicitly via representation/coding e.g. only allow representations of the TSP that correspond to a valid tour
  - Implement a repair capability for infeasible individuals

⇨ Choosing the right fitness function: an important genetic algorithm design Issue

Selection by Ranking

- Goal is to select parents for crossover
- Should create a bias towards more fitness
- Must preserve diversity in the population

(1) Selection according to RANKING

Example: Let $D = \sum_{j \in P} (1/j)$

select the k^{th} most fit member of a population

to be a parent with probability $P_k = \left(\frac{1}{k}\right) D^{-1}$

⇨ Better ranking has a higher probability of being chosen, e.g. 1st $\propto 1$, 2nd $\propto 1/2$, 3rd $\propto 1/3$ ...

Selection by Fitness

(2) Proportional to FITNESS Value Scheme

Example: Let $\bar{F} = \sum_{j \in P} Fitness(j)$

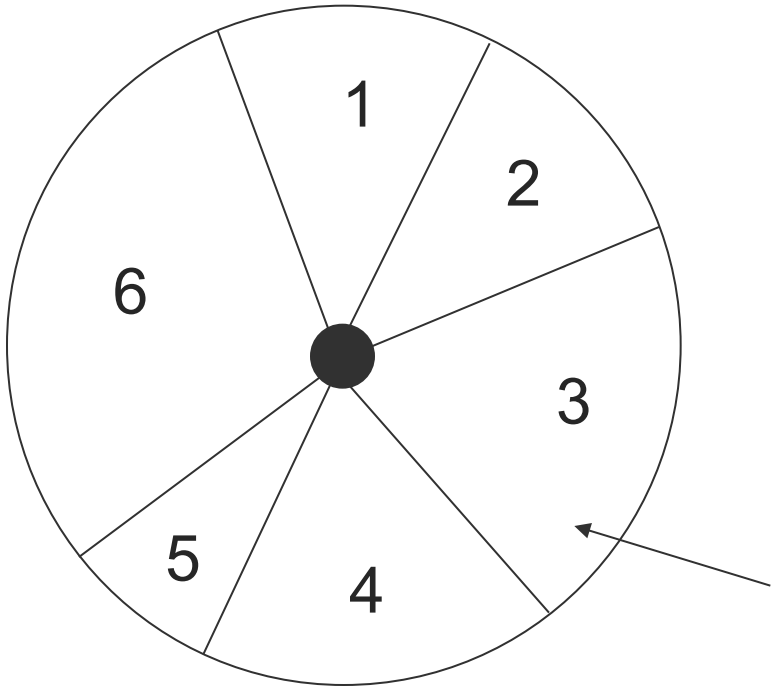
select the k^{th} most fit member of a population

to be a parent with probability $P_k = Fitness(k) \cdot \bar{F}^{-1}$

⇨ Probability of being selected for crossover is directly proportional to raw fitness score.

Roulette Wheel Selection

Roulette Wheel Selection



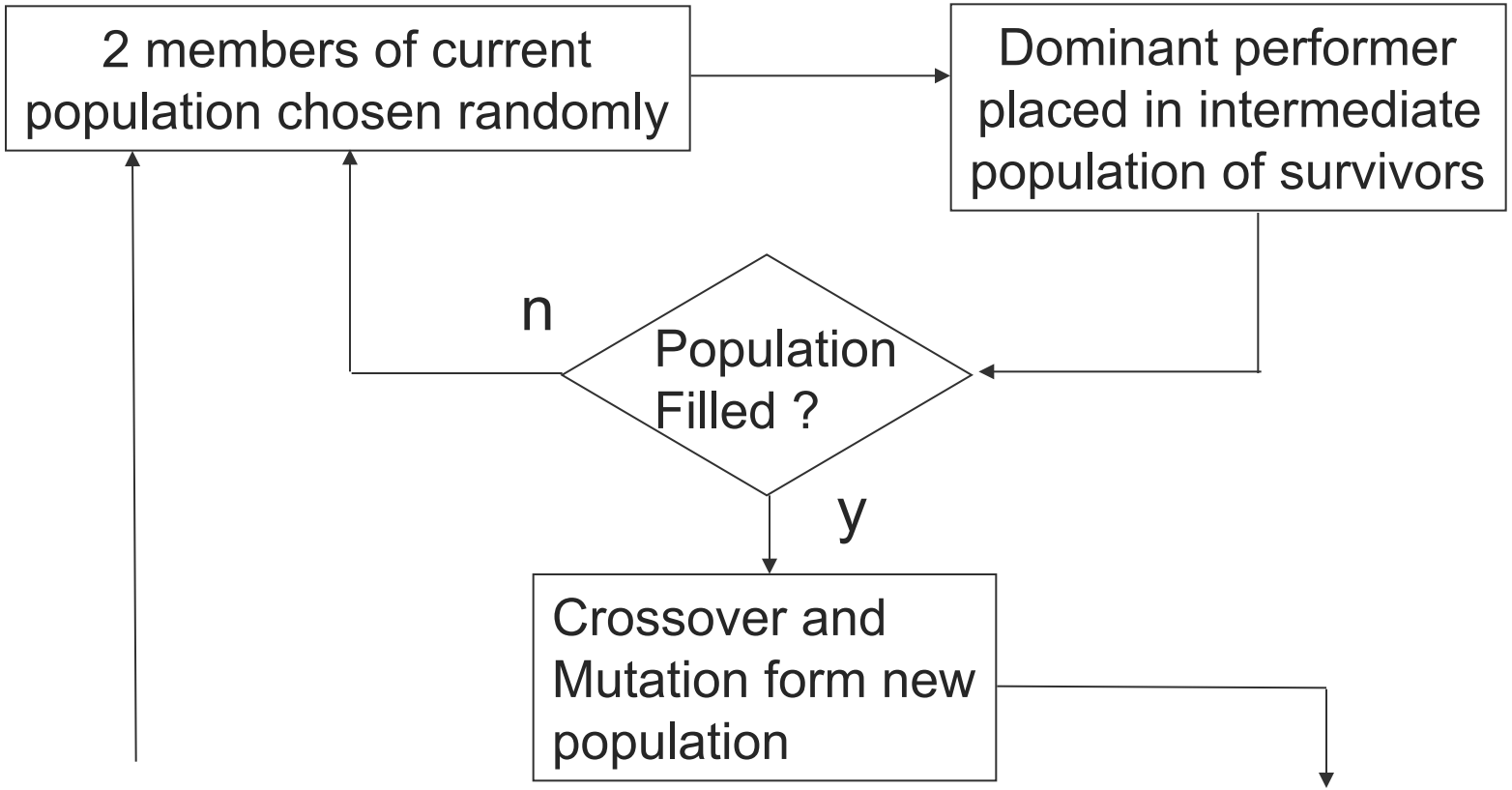
Probabilistically select individuals based on some measure of their performance.

Sum Sum of individual's selection probabilities

3rd individual in current population mapped to interval [0,*Sum*]

Selection: generate random number in [0,*Sum*]
Repeat process until desired # of individuals selected
Basically: stochastic sampling with replacement (SSR)

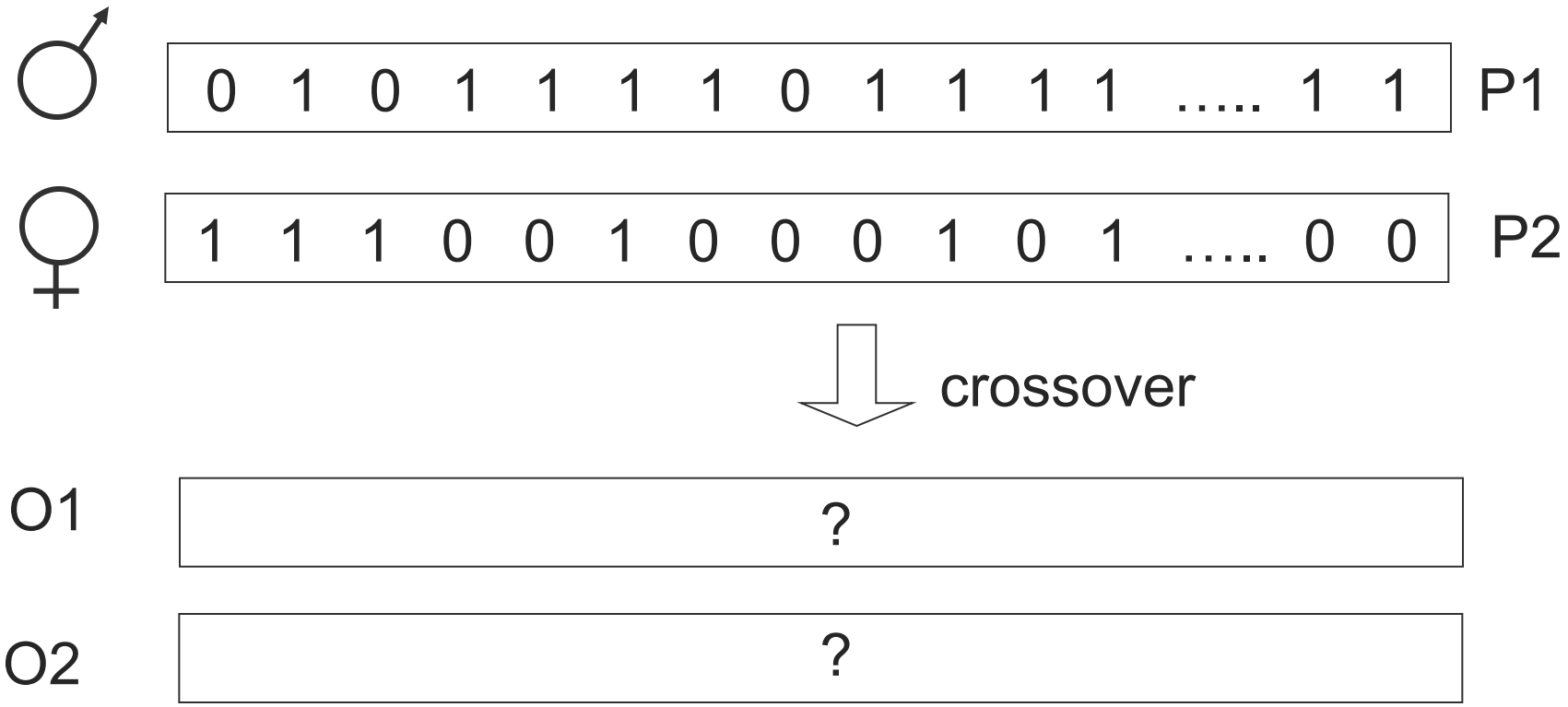
Tournament Selection



Old Population	Fitness
101010110111	8
100100001100	4
001000111110	6

Survivors	Fitness
101010110111	8
001000111110	6
101010110111	8

Crossover

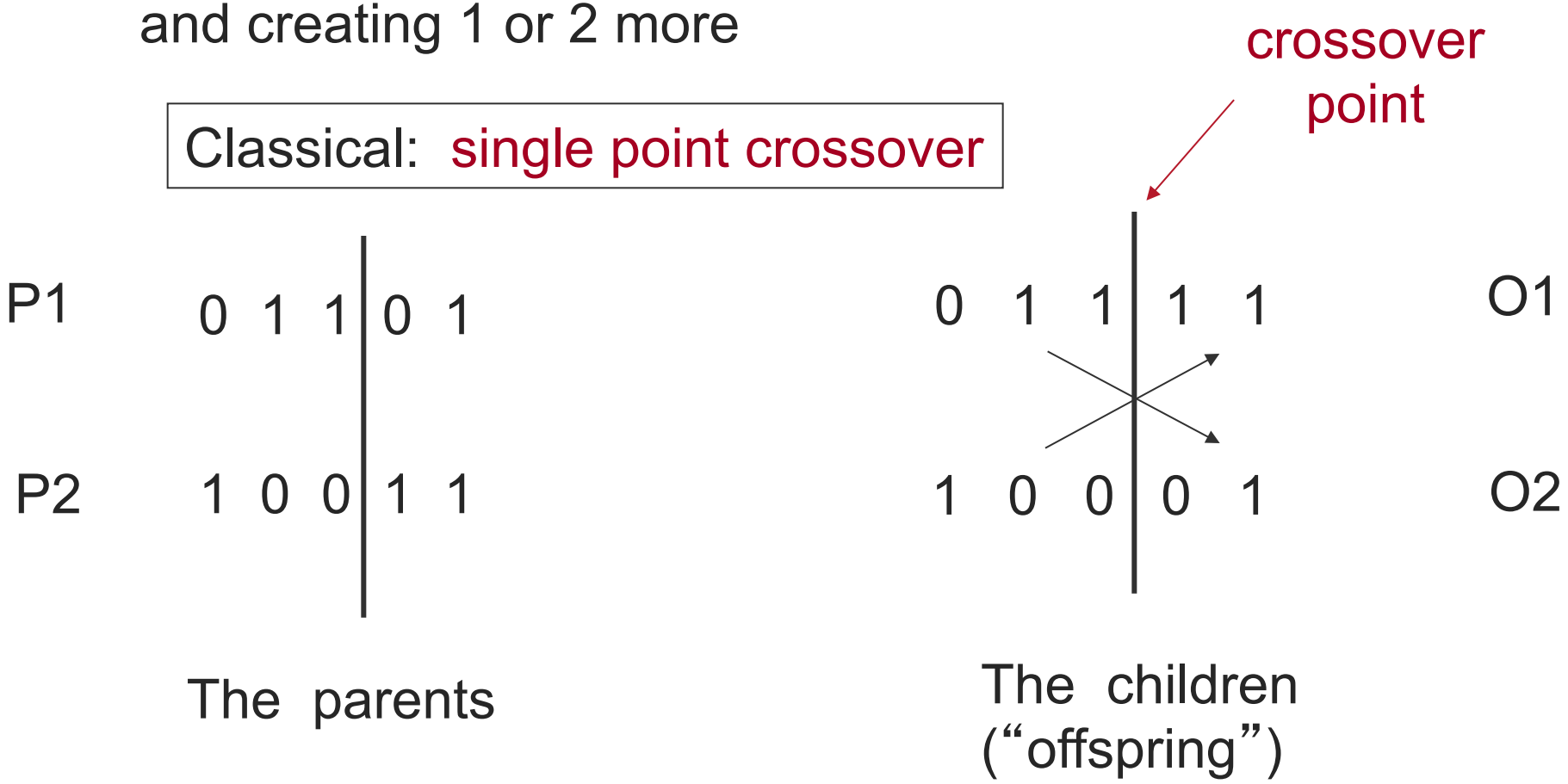


Question: How can we operate on parents P1 and P2 to create offspring O1 and O2 (same length, only 1's and 0's)?

Crossover Operator (I)

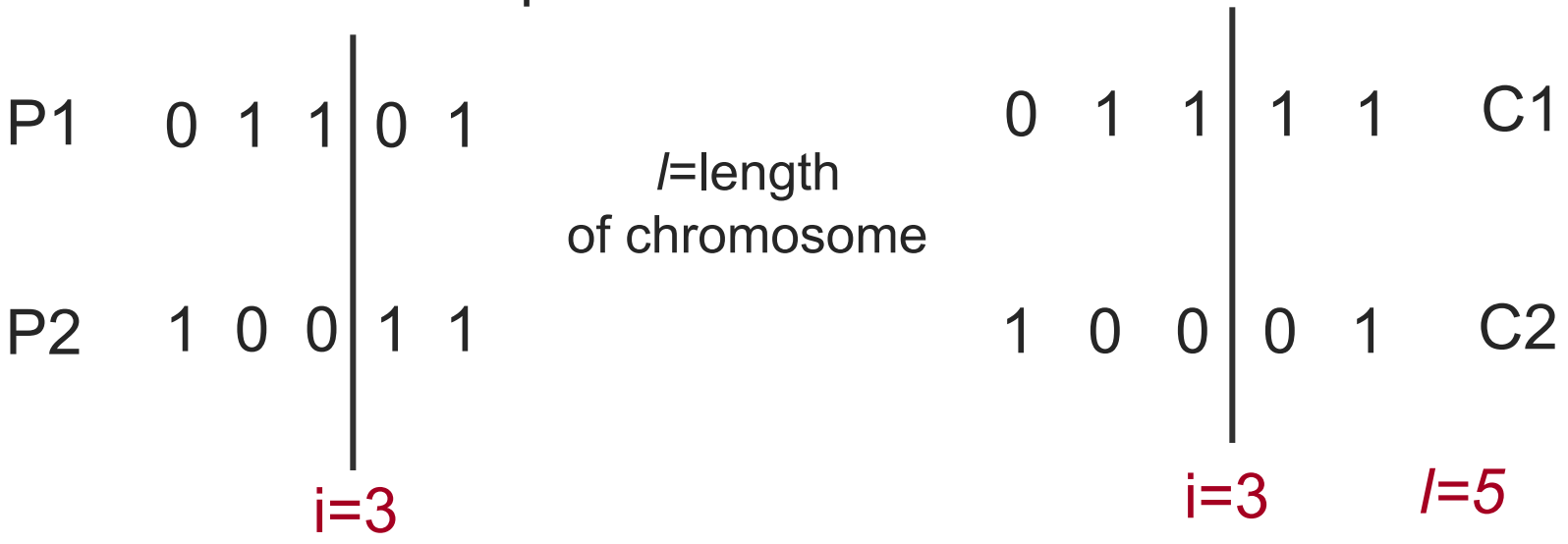
Crossover (mating) is taking 2 solutions, and creating 1 or 2 more

Classical: single point crossover



Crossover Operator (II)

More on 1-point crossover ….

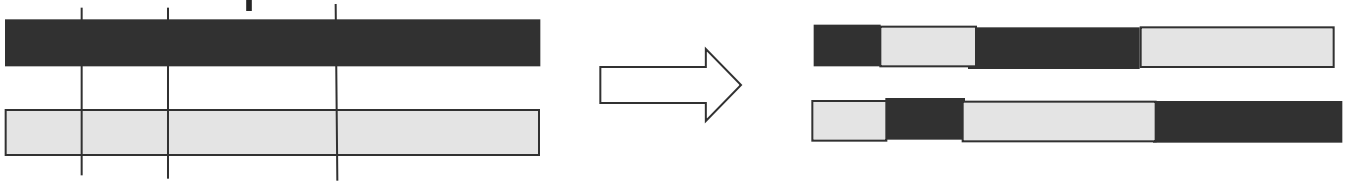


A crossover bit "i" is chosen (deliberately or randomly), splitting the chromosomes in half.

Child C1 is the 1st half of P1 and the 2nd half of P2
Child C2 is the 1st half of P2 and the 2nd half of P1

Crossover Operator (III)

- One can also do a 2-point crossover or a multi-point crossover



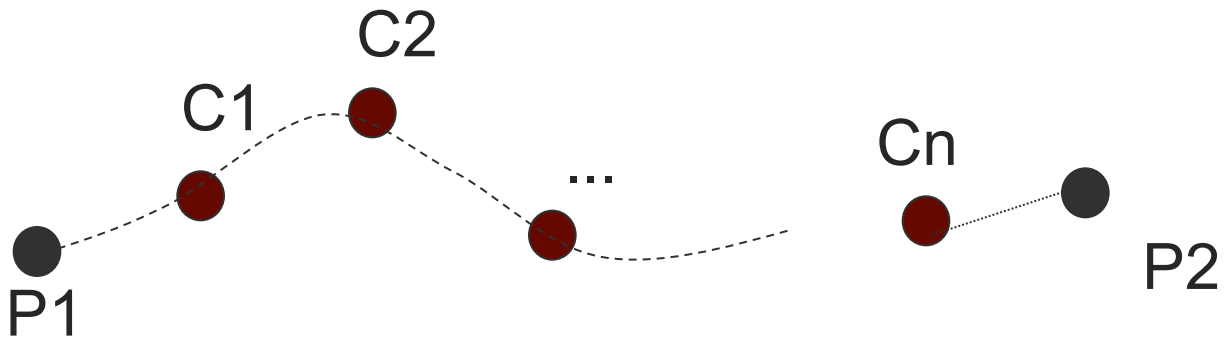
- The essential aspect is to create at least one child (solution/design) from two (or more) parent (solutions/designs)
  - there are many ways to do this

Some crossover operations:

- single point versus multiple point crossover
- path re-linking

Path Relinking

- Given Parents P1 and P2
- Create a sequence of children
  - The first child is a neighbor of P1
  - Each child is a neighbor of the previous child
  - The last child is a neighbor of P2



Example: Path Relinking

Parents

P1 1 0 0 1 0 0 1 and 0 0 1 0 1 0 0 P2

Children

1 0 0 1 0 0 0

1 0 0 1 1 0 0

1 0 0 0 1 0 0

1 0 1 0 1 0 0

Create a path of children, then select the best one.

Okay approach, but solutions tend to be interpolations of initial population.

Some Insertion Strategies

- Can replace an entire population at a time (go from generation k to k+1 with no survivors)
  - select N/2 pairs of parents
  - create N children, replace all parents
  - polygamy is generally allowed

N = # of members in population if steady state

- Can select two parents at a time
  - create one child
  - eliminate one member of population (weakest?)

- "Elitist" strategy
  - small number of fittest individuals survive unchanged

- "Hall-of-fame"
  - remember best past individuals, but don't use them for progeny

Initialization

Somehow we need to create an initial population of solutions to start the GA. How can this be done?

- Random initial population, one of many options
- Use random number generator to create initial population (caution with seeds!)
- Typically use uniform probability density functions (pdf's)
- Typical goal: Select an initial population that has both quality and diversity

Example:

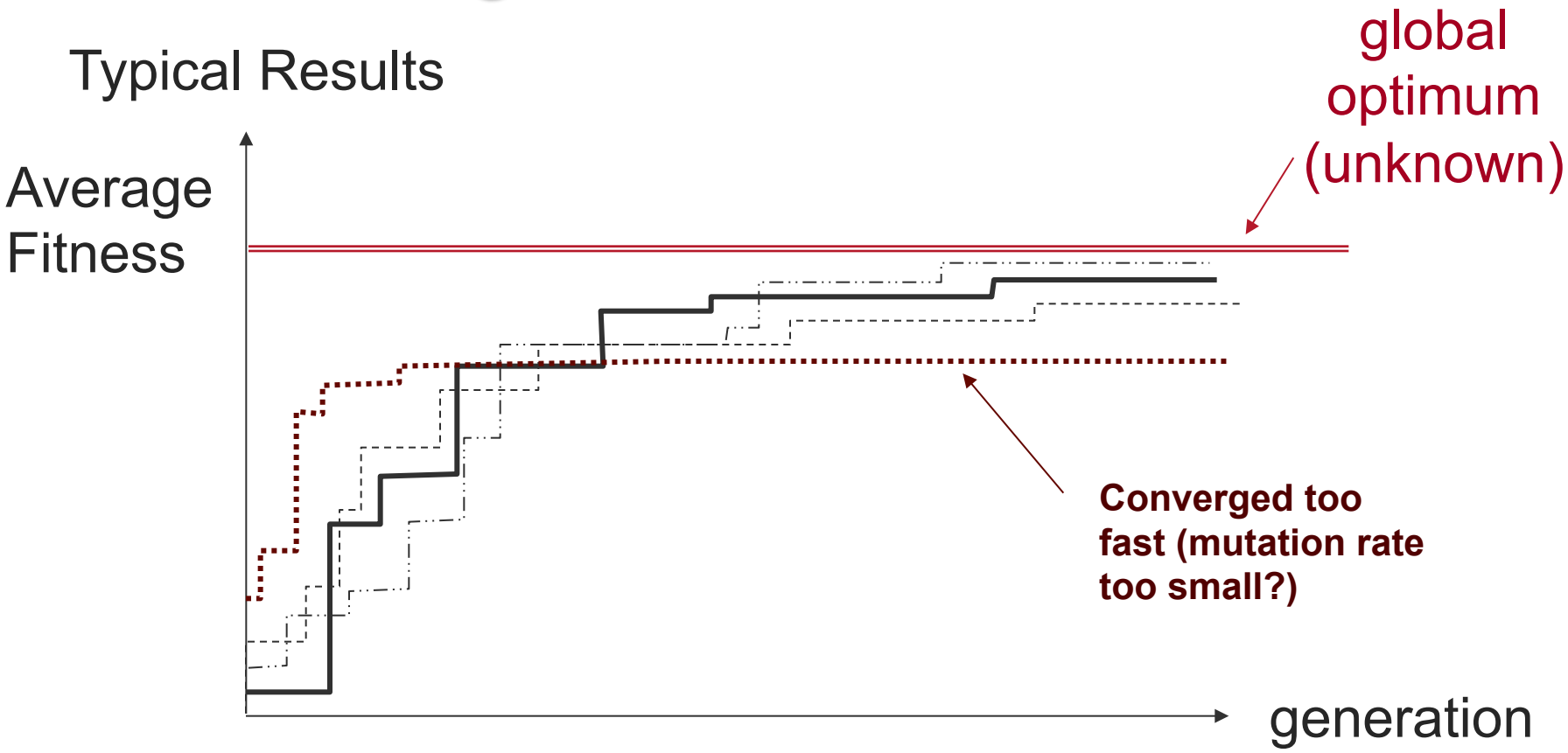
N_{ind} - size of binary population
L_{ind} - Individual chromosome length

```
round(rand(1,6))    >>  1 1 1 1 0 0
```

Need to generate N_{ind} x L_{ind} random numbers from {0,1}

Rule of thumb: Population Size at Least $N_{ind} \sim 4\ L_{ind}$

GA Convergence



Average performance of individuals in a population is expected to increase, as good individuals are preserved and bred and less fit individuals die out.

GA Stopping Criteria

Some options:

- X number of generations completed - typically O(100)
- Mean deviation in performance of individuals in the population falls below a threshold σ_J<x (genetic diversity has become small)
- Stagnation - no or marginal improvement from one generation to the next: $(J_{n+1}-J_n)<X$

GAs versus other methods

Differ from traditional search/optimization methods:

- GAs search a population of points in parallel, not only a single point
- GAs use probabilistic transition rules, not deterministic ones
- GAs work on an encoding of the design variable set rather than on the variables themselves
- GAs do not require derivative information or other auxiliary knowledge - only the objective function and corresponding fitness levels influence search

Next Lecture

- Speciality GA's
- Particle Swarm Optimization (PSO)
- Tabu Search (TS)
- Selection of Optimization Algorithms
  - Which algorithm is most suited to my problem?
- Design Optimization Applications

Book References

Holland J., “Adaptation in Natural and Artificial Systems”, University of Michigan Press, 1975

Goldberg, D.E.,” Genetic Algorithms in Search, Optimization and Machine Learning”, Addison Wesley, 1989

Lecture 11: Multidisciplinary System Analysis and Design Optimization (MSADO)

Genetic Algorithms (cont.), Particle Swarm Optimization, Tabu Search, Optimization Algorithm Selection

March 9, 2015

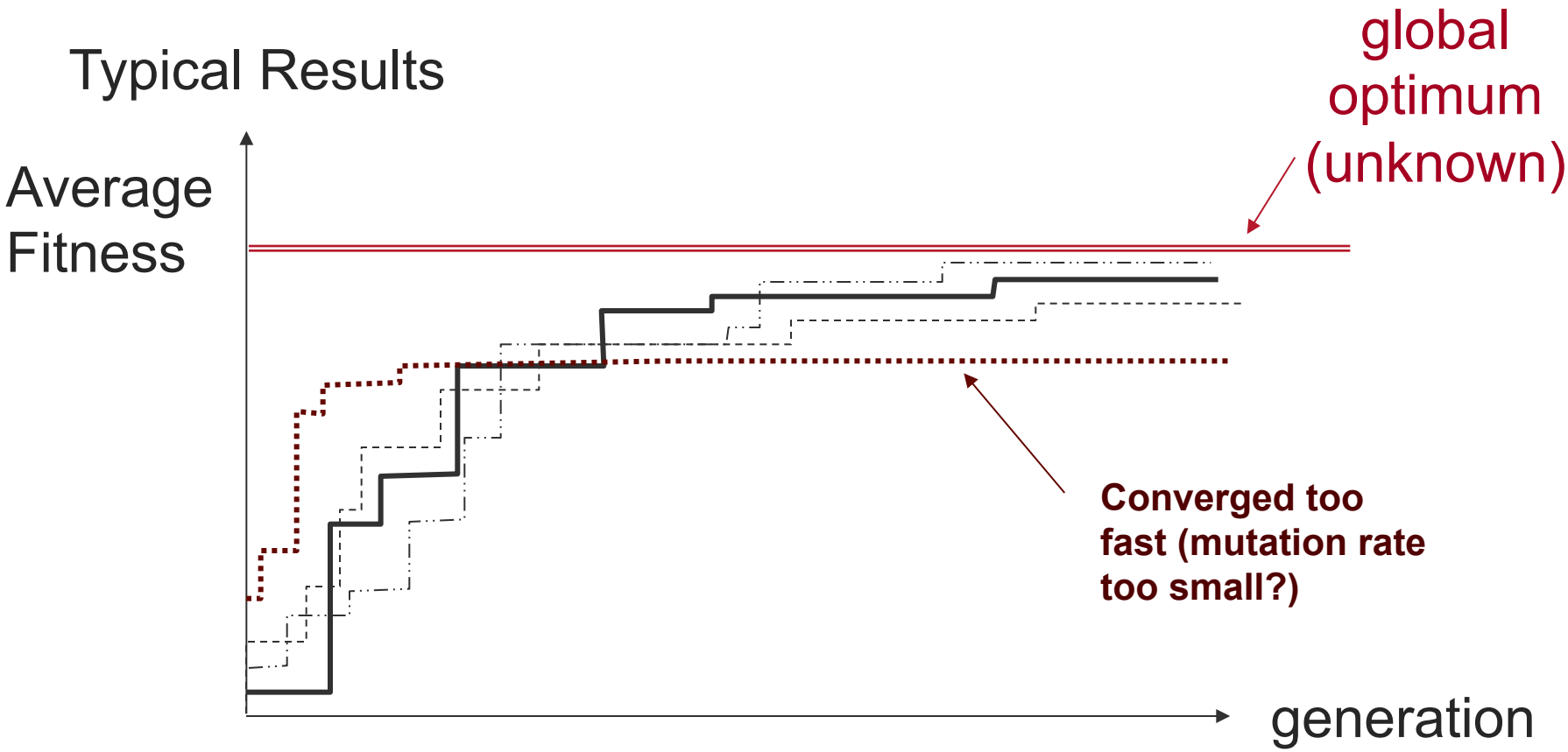
Prof. Douglas Allaire

Today's Topics

- Genetic Algorithms (part 2)
- Particle Swarm Optimization
- Tabu Search
- Selection of Optimization Algorithms

GA Convergence

Typical Results



Average performance of individuals in a population is expected to increase, as good individuals are preserved and bred and less fit individuals die out.

GA vs. Traditional Methods

Differ from traditional search/optimization methods:

- GAs search a population of points in parallel, not only a single point
- GAs use probabilistic transition rules, not deterministic ones
- GAs work on an encoding of the design variable set rather than the variable set itself
- GAs do not require derivative information or other auxiliary knowledge - only the objective function and corresponding fitness levels influence search

Parallel GA's

GA's are very amenable to parallelization.

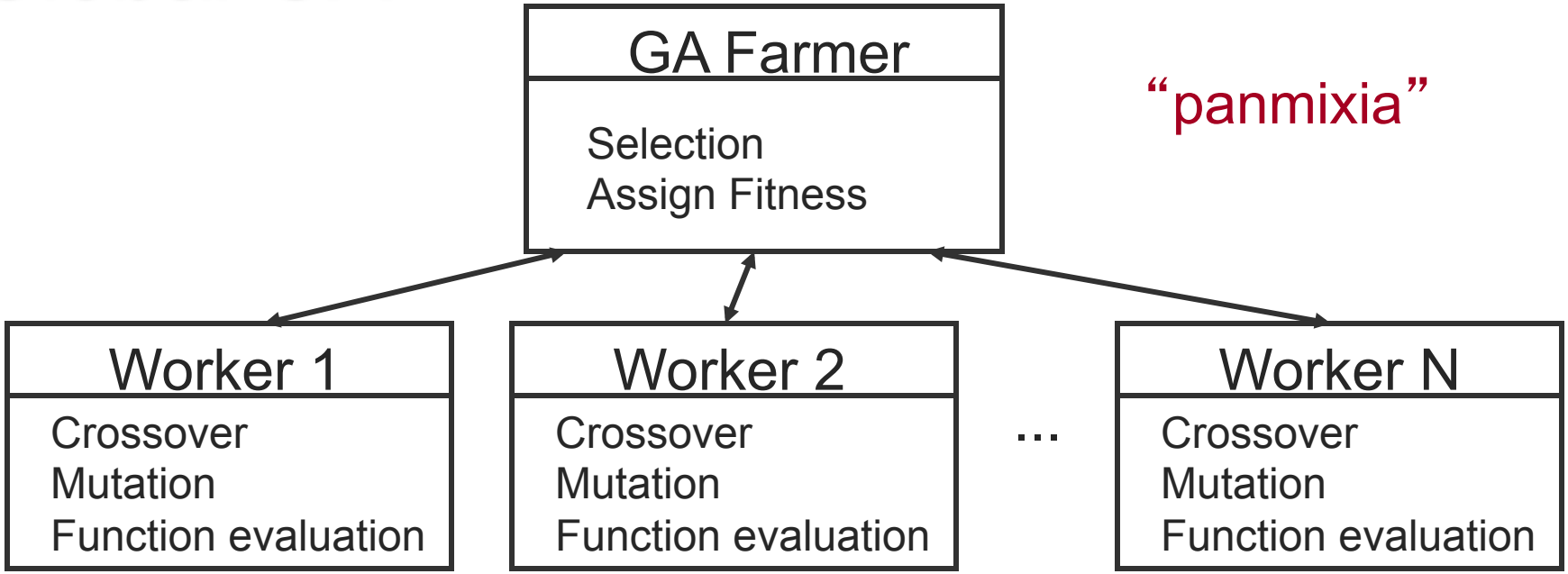
Motivations: - faster computation (parallel CPU's)
- attack larger problems
- introduce structure and geographic location

There are three classes of parallel GA's:

- Global GA's
- Migration GA's
- Diffusion GA's

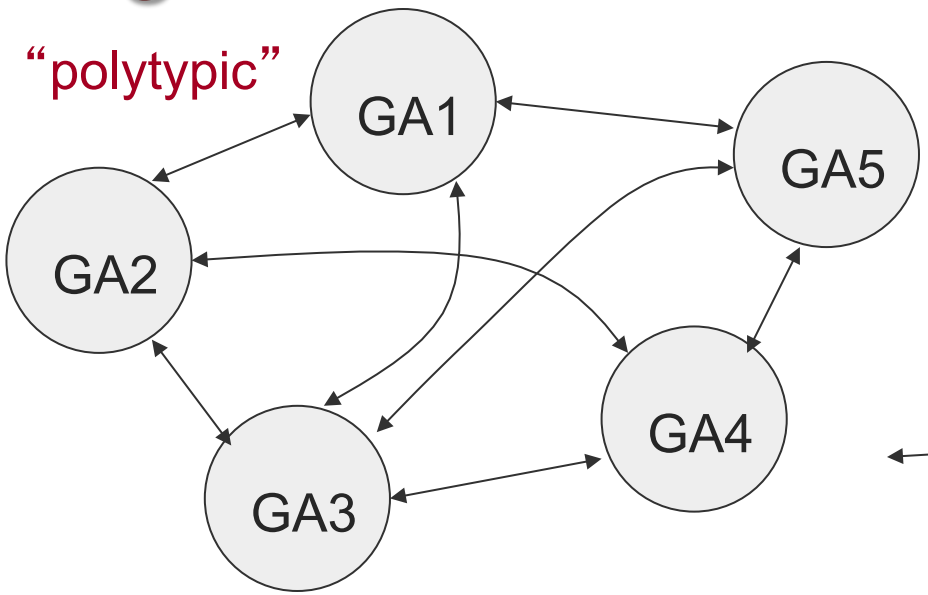
Main differences lie in :
- population structure
- method of selecting individuals for reproduction

Global GA



- GA Farmer node initializes and holds entire population
- Interesting when objective function evaluation expensive
- Typically implemented as a master-slave algorithm
- Balance serial-parallel tasks to minimize bottlenecks
- Issue of synchronous/asynchronous operation

Migration GA



Does NOT operate globally on a single population

Each node represents a subgroup relatively isolated from each other

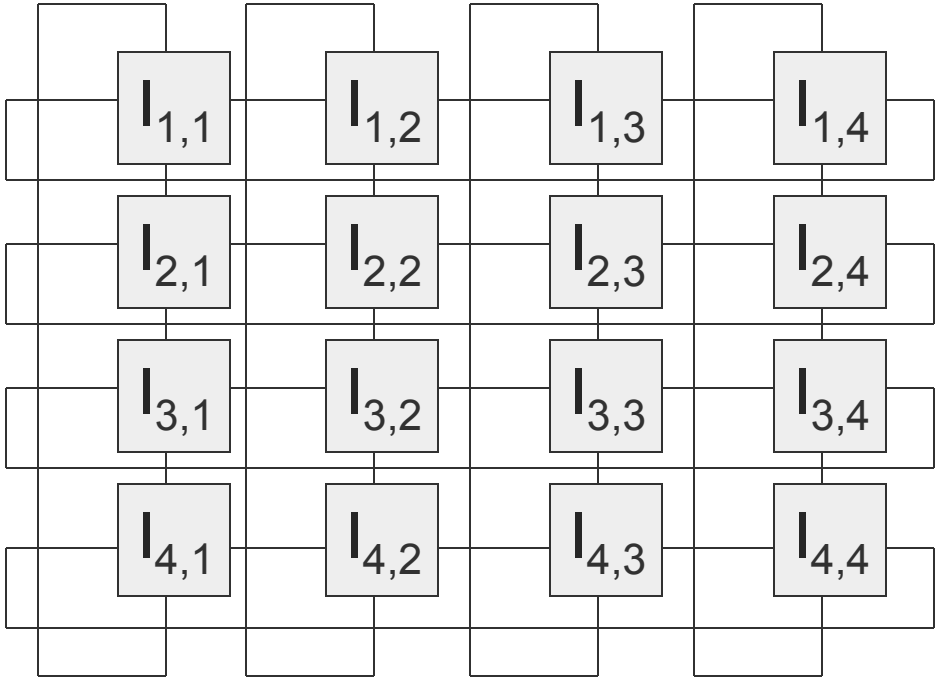
"breeding groups"= demes

```
-- Each node (GAi)
WHILE not finished
  SEQ
      … Selection
      … Reproduction
      … Evaluation
      PAR
          … send emigrants
          … receive immigrants
```

More closely mimics biological metaphor

First introduced by Grosso in 1985

Diffusion GA's



Toroidal-Mesh parallel processing network

```
-- Each Node (Ii,j)
WHILE not finished
SEQ
… Evaluate
PAR
   … send self to neighbors
   … receive neighbors
… select mate
… reproduce
```

Neighborhood, cellular or fine-grained GA

- Population is a single continuous structure, but
- Each individual is assigned a geographic location
- Breeding only allowed within a small local neighborhood
- Example: I(2,2) only breeds with I(1,2), I(2,1),I(2,3),I(3,2)

Good News about GA's

- GA's work well on mixed discrete/ continuous problems
- GA's require little information about problem
- No gradients required
- Simple to understand and set up and implement
- Can operate on various representations
- GA's are very robust
- GA's are stochastic, that is, they exploit randomness
- GA's can be easily parallelized

Bad News about GA's

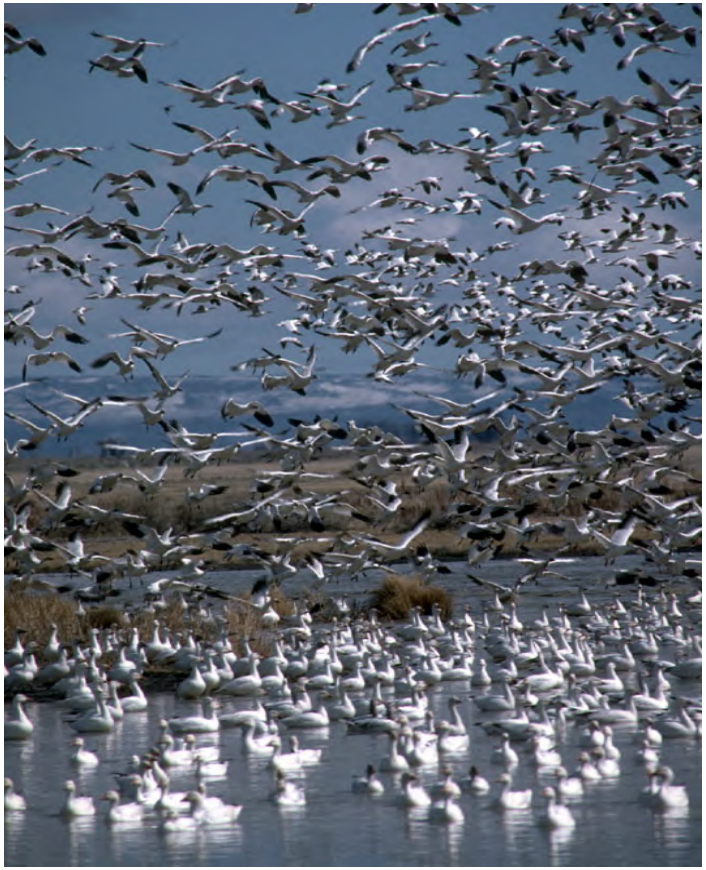
- GA implementation is still an art and requires some experience
- Convergence behavior very dependent on some tuning parameters: mutation rate, crossover, population size
- Designing fitness function can be tricky
- Cumbersome to take into account constraints
- GA's can be computationally expensive
- No clear termination criteria
- No knowledge of true global optimum

Particle Swarm Optimization

Introduced in 1995: Kennedy, J. and Eberhart, R., "Particle Swarm Optimization," Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia 1995, pp. 1942-1945.

Particle Swarm Optimization

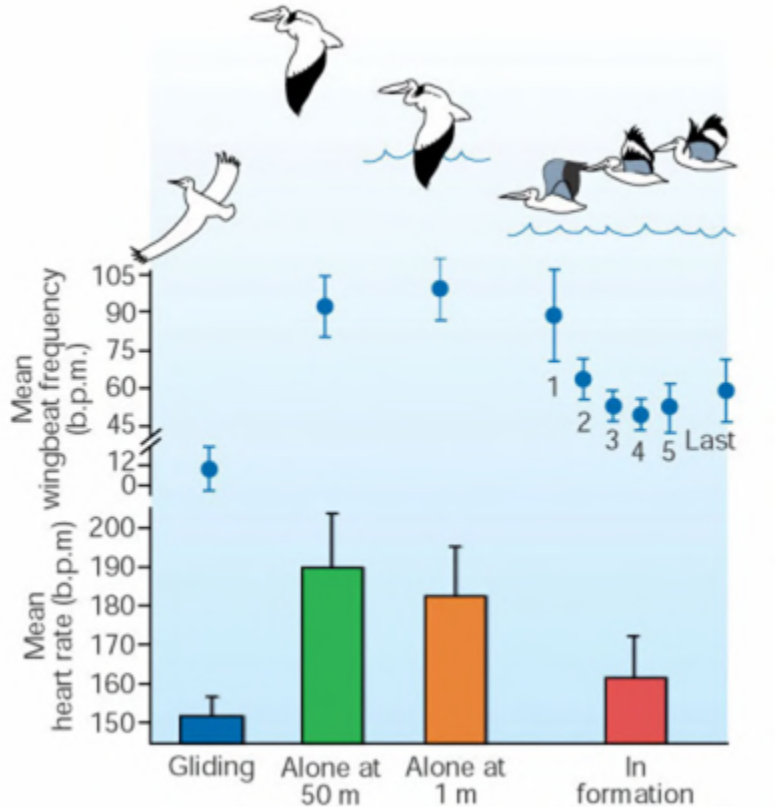
A pseudo-optimization method (heuristic) inspired by the collective intelligence of swarms of biological populations.



Flocks of Birds

Colonies of Insects

Swarming in System Design



A study of great white pelicans has found that birds flying in formation use up to a fifth less energy than those flying solo (Weimerskirch *et al*.).

PSO Conceptual Development

- How do large numbers of birds produce seamless, graceful flocking choreography, while often, but suddenly changing direction, scattering and regrouping?
  - "Decentralized" local processes.
  - Manipulation of inter-individual distances (keep pace and avoid collision).
- Are there any advantages to the swarming behavior for an individual in a swarm?
  - Can profit from the discoveries and previous experience of other swarm members in search for food, avoiding predators, adjusting to the environment, i.e., information sharing yields evolutionary advantage.
- Do humans exhibit social interaction similar to the swarming behavior in other species?
  - Absolutely, humans learn to imitate physical motion early on; as they grow older, they imitate their peers on a more abstract level by adjusting their beliefs and attitudes to conform with societal standards.

Basic PSO Algorithm

- The swarming behavior of the birds could be for the reason of finding optimal food resources.
- A swarming model could be used (with minor modifications) to find optimal solutions for *N*-dimensional, non-convex, multi-modal, nonlinear functions.

Algorithm Description

- Particle Description: each particle has three features
  - Position $\mathbf{x}_k^i$ (this is the i^{th} particle at time k, notice vector notation)
  - Velocity $\mathbf{v}_k^i$ (similar to search direction, used to update the position)
  - Fitness or objective $f\left(\mathbf{x}_k^i\right)$ (determines which particle has the best value in the swarm and also determines the best position of each particle over time)

Initial Swarm

- Initial Swarm
  - No well established guidelines for swarm size, normally 10 to 60.
  - particles are randomly distributed across the design space.

$$\mathbf{x}_0^i = \mathbf{x}_{\min} + rand\left(\mathbf{x}_{\max} - \mathbf{x}_{\min}\right)$$

where $\mathbf{x}_{\min}$ and $\mathbf{x}_{\max}$ are vectors of lower and upper limit values respectively.

  - Evaluate the fitness of each particle and store:
    - particle best ever position (particle memory $\mathbf{p}^i$ here is same as $\mathbf{x}_0^i$)
    - Best position in current swarm (influence of swarm $\mathbf{p}_0^g$)
  - Initial velocity is randomly generated.

$$\mathbf{v}_0^i = \frac{\mathbf{x}_{\min} + rand\left(\mathbf{x}_{\max} - \mathbf{x}_{\min}\right)}{\Delta t} = \frac{\text{position}}{\text{time}}$$

Basic PSO Algorithm

- Velocity Update
  - provides search directions
  - Includes deterministic and probabilistic parameters.
  - Combines effect of current motion, particle own memory, and swarm influence.

New velocity

$$\mathbf{v}_{k+1}^{i} = w\,\mathbf{v}_{k}^{i} + c_1\,rand\,\frac{\left(\mathbf{p}^{i} - \mathbf{x}_{k}^{i}\right)}{\Delta t} + c_2\,rand\,\frac{\left(\mathbf{p}_{k}^{g} - \mathbf{x}_{k}^{i}\right)}{\Delta t}$$

current motion

particle memory influence

swarm influence

inertia factor

0.4 to 1.4

self confidence

1.5 to 2

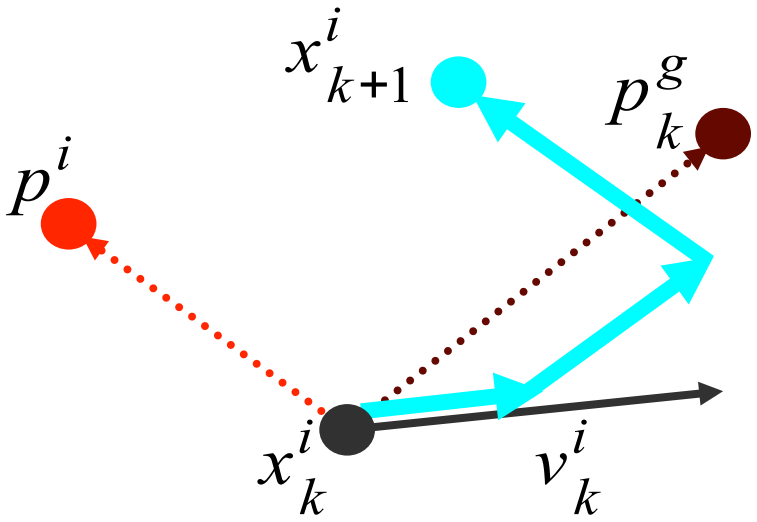
swarm confidence

2 to 2.5

Basic PSO Algorithm

- Position Update
  - Position is updated by velocity vector.

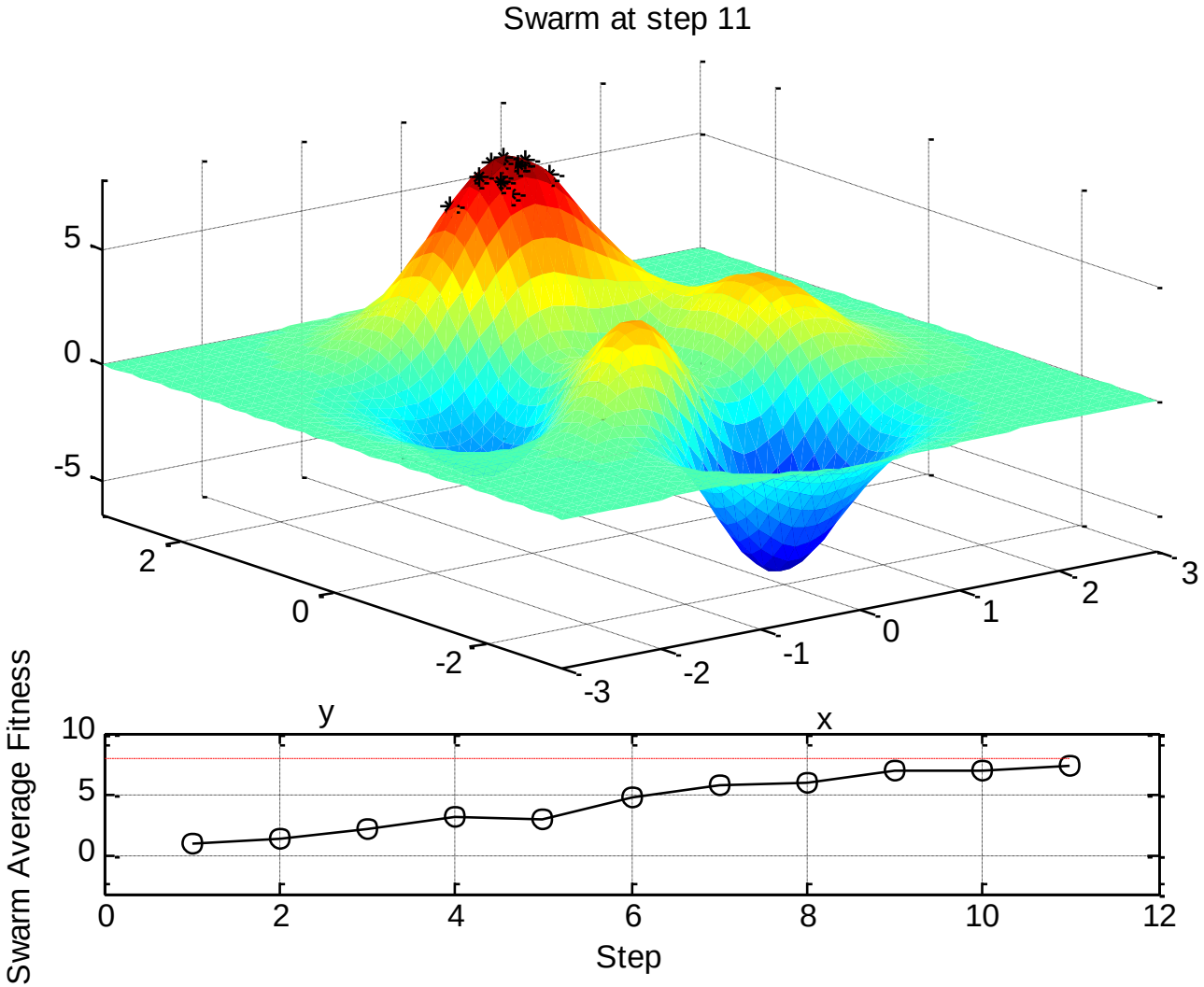
$$\mathbf{x}_{k+1}^{i} = \mathbf{x}_{k}^{i} + \mathbf{v}_{k+1}^{i}\Delta t$$



- Stopping Criteria
  - Maximum change in best fitness smaller than specified tolerance for a specified number of moves (*S*).

$$\left| f\left(\mathbf{p}_{k}^{g}\right) - f\left(\mathbf{p}_{k-q}^{g}\right) \right| \le \varepsilon \qquad q = 1,2,..S$$

PSO Peaks Demo



Constraint Handling

- Side Constraints
  - Velocity vectors can drive particles to "explosion".
  - Upper and lower variable limits can be treated as regular constraints.
  - Particles with violated side constraints could be reset to the nearest limit.

- Functional Constraints
  - Exterior penalty methods (e.g., linear or quadratic).

fitness function

$$f(\mathbf{x}) = \underbrace{\phi(\mathbf{x})}_{\text{objective function}} + \underbrace{\sum_{i=1}^{N_{con}} r_i \left(\max[0, g_i]\right)^2}_{\text{penalty function}}$$

penalty multipliers

  - If a particle is infeasible, last search direction (velocity) was not feasible. Set current velocity to zero.

$$\mathbf{v}_{k+1}^{i} = c_1 \; rand \; \frac{\left(\mathbf{p}^{i} - \mathbf{x}_k^{i}\right)}{\Delta t} + c_2 \; rand \; \frac{\left(\mathbf{p}_k^{g} - \mathbf{x}_k^{i}\right)}{\Delta t}$$

Discretization

- System problems typically include continuous, integer, and discrete design variables.
- Basic PSO works with continuous variables.
- There are several methods that allows PSO to handle discrete variables.
- The literature reports that the simple method of rounding particle position coordinates to the nearest integers provide the best computational performance.

Final Comments on PSO

- This is a method "in the making" - many versions are likely to appear.
- Poor repeatability in terms of:
  - finding optimal solution
  - computational cost
- More robust constraint (side and functional) handling approaches are needed.
- Guidelines for selection of swarm size, inertia and confidence parameters are needed.
- PSO appears to be more computationally efficient than GA

PSO References

- Kennedy, J. and Eberhart, R., "Particle Swarm Optimization," Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia 1995, pp. 1942-1948.

- Venter, G. and Sobieski, J., "Particle Swarm Optimization," AIAA 2002-1235, 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, Denver, CO., April 2002.

- Kennedy, J. and Eberhart, R., *Swarm Intelligence*, Academic Press, 1st ed., San Diego, CA, 2001.

- Hassan R., Cohanim B., de Weck O.L., Venter G., "A Comparison of Particle Swarm Optimization and the Genetic Algorithm", AIAA-2005-1897, 1st AIAA Multidisciplinary Design Optimization Specialist Conference, Austin, Texas, April 18-21, 2005

Tabu Search

Tabu Search (TS)

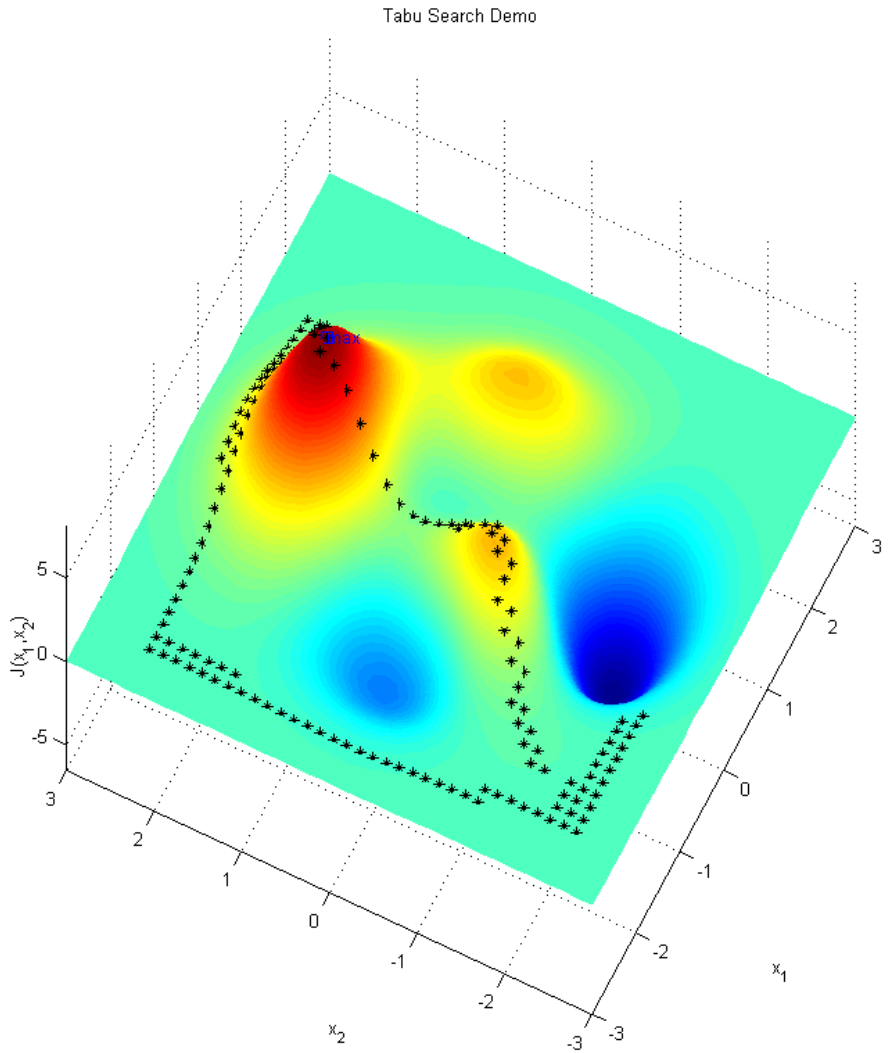
- Attributed to Glover (1990)
- Search by avoiding points in the design space that were previously visited ("tabu") – keep memory !
- Accept a new "poorer" solution if it avoids a solution that was already investigated – maximize new information
- Intent: Avoid local minima
- Record all previous moves in a "running list" = memory
- Record recent, now forbidden, moves in a "tabu" list
- Applied to combinatorial optimization problems

- Glover, F. and M. Laguna. *Tabu Search*. Kluwer, Norwell, MA Glover, F. and M. Laguna. (1997).

Tabu Search (pseudo code)

```
   Given a feasible solution x* with objective
   function value J*, let x := x* with J(x) = J*.
 Iteration:
while stopping criterion is not fulfilled do
   begin
(1)select best admissible move that transforms x
   into x' with objective function value J(x')
   and add its attributes to the running list
(2) perform tabu list management: compute moves
   (or attributes) to be set tabu, i.e., update
   the tabu list
(3) perform exchanges: x := x', J(x) = J(x'); if
   J(x) < J* then J* := J(x), x* := x
endif
endwhile
Result: x* is the best of all determined
   solutions, with objective function value J*.
```

Tabu Search Demo



Algorithm Selection

Selection of Algorithms

First characterize the design optimization problem:

1. Linearity and smoothness of objective function $J(\mathbf{x})$ and constraints $\mathbf{g}(x)$, $\mathbf{h}(x)$
2. Type of design variables **x** (real, integer,…)
3. Number of design variables n
4. Expense of evaluating $J(\mathbf{x})$, $\mathbf{g}(x)$, $\mathbf{h}(x)$
    1. [CPU time, Flops]
5. Expense of evaluating gradient of $J(\mathbf{x})$
6. Number of objectives, z

Nonlinearity

Crumpled Paper Analogy to Show Nonlinearity:

- **Use a sheet of paper to represent the response surface of**

$$J = f(x_1, x_2)$$

- **If the paper is completely "flat", with or without slope, then y is a *Linear* Function which can be represented as**

$$y = c_0 + c_1 x_1 + c_2 x_2$$

- **If the paper is twisted slightly with some curvature, then it becomes a nonlinear function. Low nonlinearity like this may be approximated by a *Quadratic* function like**

$$y = c_0 + c_1 x_1 + c_2 x_2 + c_3 x_1^2 + c_4 x_2^2 + c_5 x_1 x_2$$

- **Crumple the paper and slightly flatten it, then it becomes a "*very nonlinear*" function.** Observe the irregular terrain and determine whether it is possible to approximate the irregular terrain by a simple quadratic function.

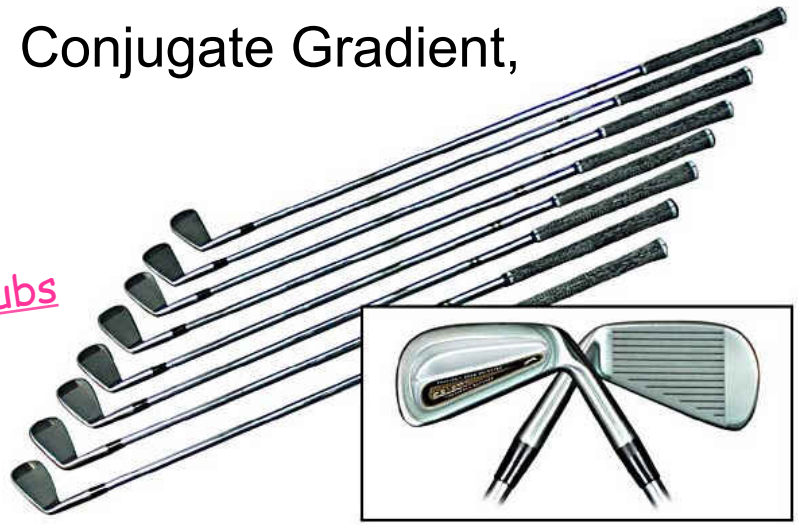
(Rough) Algorithm Selection Matrix

	Linear *J* and *g* and *h*	Nonlinear J or g or h
Continuous, real **x** (all)	Simplex Barrier Methods	SQP (constrained) Newton (unconstrained)
Discrete **x** (at least one)	MILP (e.g. Branch-and-Bound)	GA SA, Tabu Search PSO

Golf Club Analogy

Gradient-Based:

SLP, SQP, MMFD, Conjugate Gradient, Exterior Penalty,...



Stochastic-Based:

Simulated Annealing, Genetic Algorithms.

Credit: Howard Lee, GE

Hybrid Optimization Algorithms:

Use a Combination of "Clubs" to Search Optimum to Leverage the Strength of Individual Club.

Summary

- Gradient Search Techniques
  - Efficient, repeatable, use gradient information
  - Can test solution via KKT (Optimality) conditions
  - Well suited for nonlinear problems with continuous variables
  - Can easily get trapped at local optima

- Heuristic Techniques
  - Used for combinatorial and discrete variable problems
  - Use both a rule set and randomness
  - Don't use gradient information, search broadly
  - Avoid local optima, but are expensive

- Hybrid Approaches
  - Use effective combinations of search algorithms
  - Two sub-approaches
    - Use the classical "pure" algorithms in sequence
    - Hybridize algorithms to include elements of memory, swarm behavior, mixing etc ….. Ongoing research

Lecture 12: Multidisciplinary System Analysis and Design Optimization (MSADO)

Goal Programming and Isoperformance

March 11, 2015

Prof. Douglas Allaire

Optimization method selection

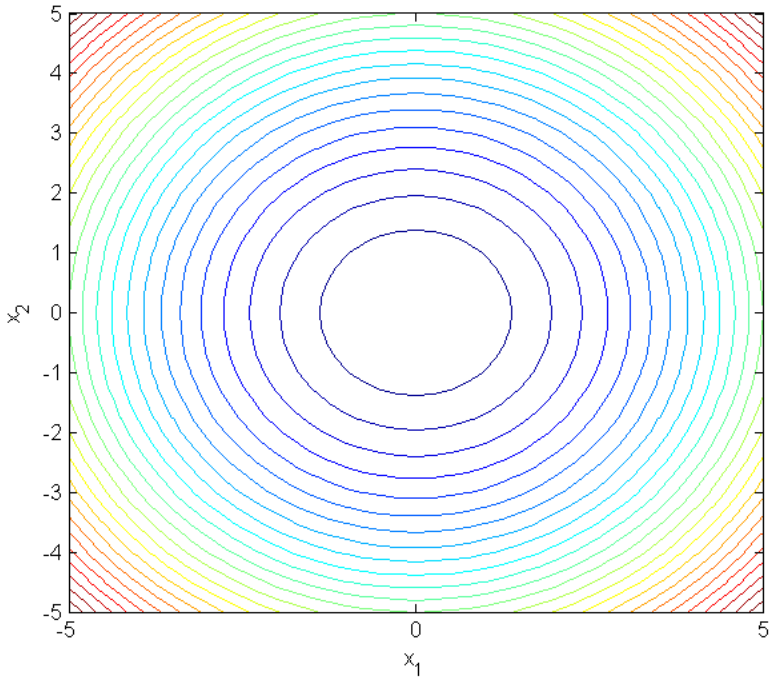
- Gradient Based:
  - Steepest descent
  - Conjugate Gradient
  - Newton's Method
  - Quasi-Newton
- Direct Search:
  - Nelder-Mead Simplex
  - Others (e.g., compass search)
- Note: The gradient methods have a constrained equivalent.
  - Steepest Descent/CG: Use projection
  - Newton/Quasi-Newton: SQP
  - Direct search typically uses barrier or penalty methods

Heuristic Methods

- Simulated Annealing
- Genetic Algorithms
- Particle Swarm Optimization
- Tabu Search

What is a good algorithm?

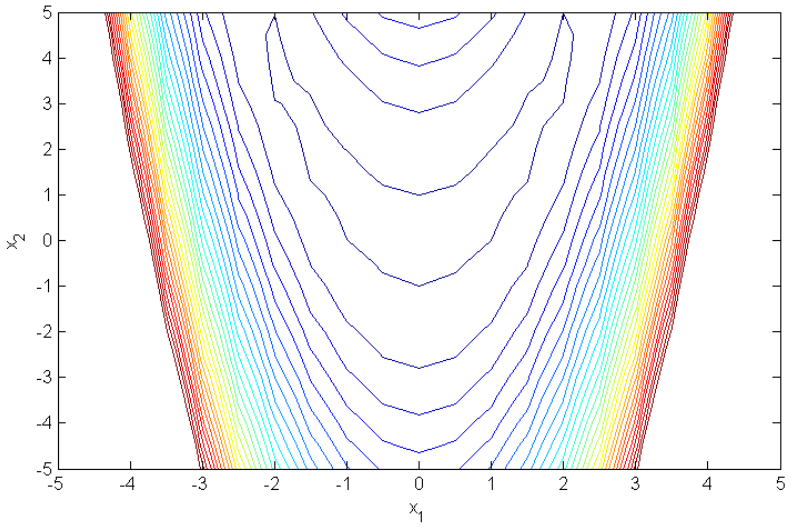
Objective Contours:



- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

What is a good algorithm?

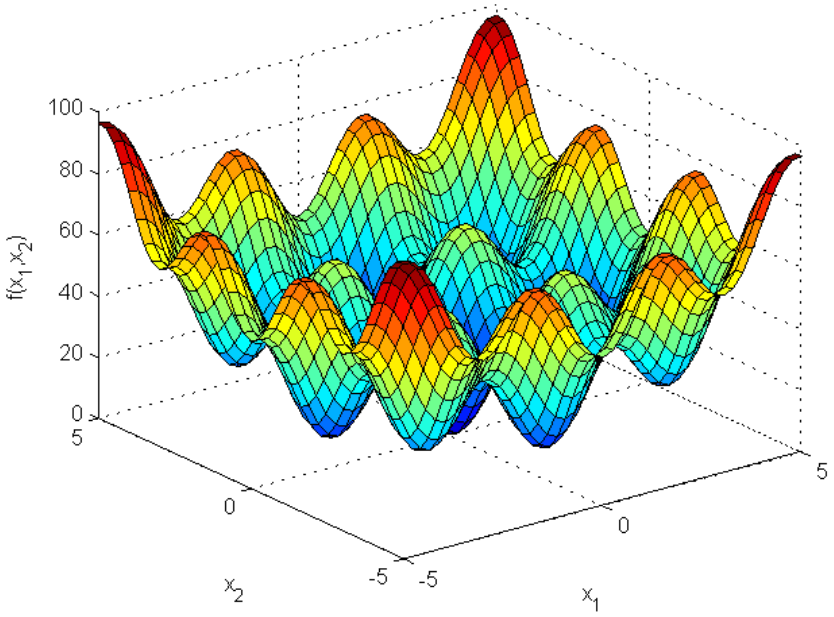
$$f(\mathbf{x}) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$$



- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

What is a good algorithm?

Objective Function:



a) Find quick improvement?

b) Find global optima?

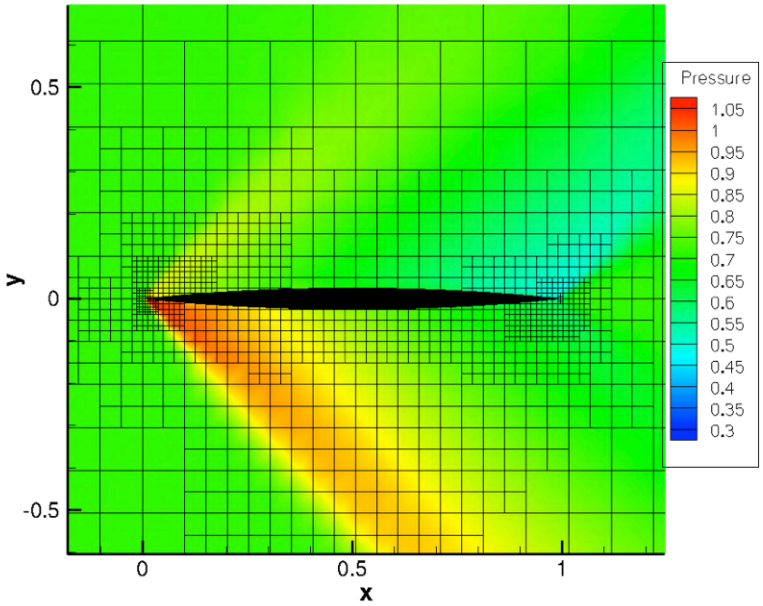
- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

What is a good algorithm?

- $x_1=\{1,2,3,4\}$
- $x_2 \in \Re$
- min $f(x_1,x_2)$

- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

What is a good algorithm?



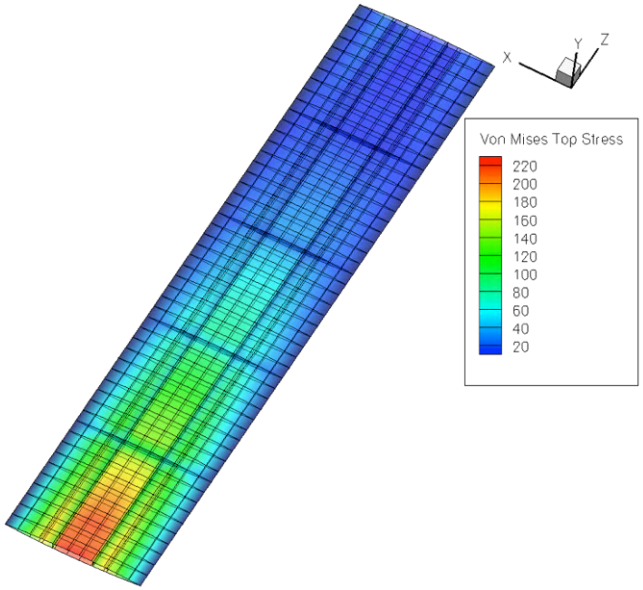
- Airfoil design with CFD
  - Run-time~3 hours

a) Without an adjoint solution?

b) With an adjoint solution?

- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

What is a good algorithm?

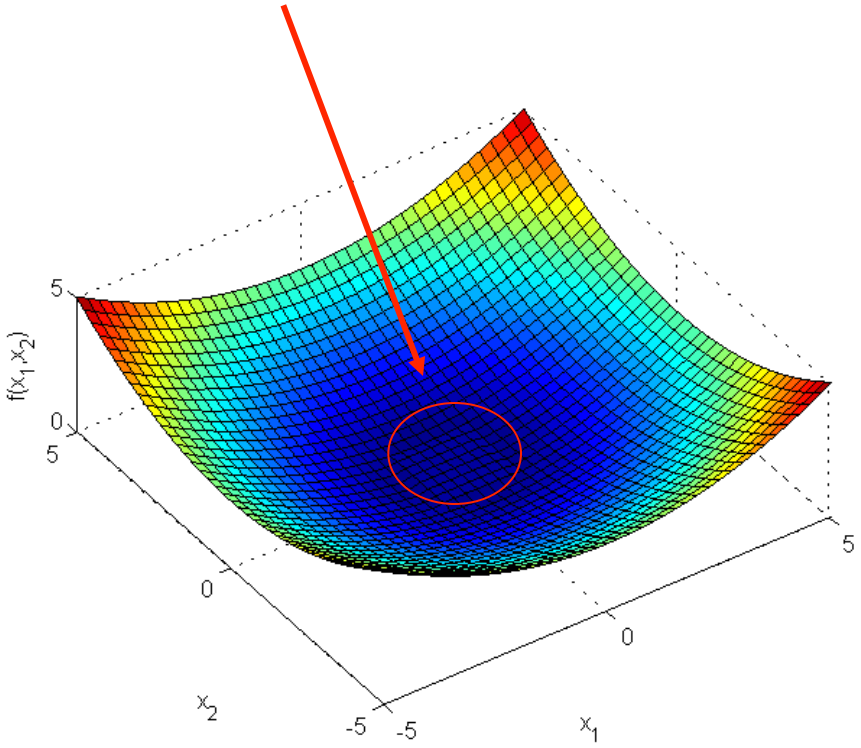


- Minimize weight
  - s.t. stress<σ_{max}
- Nastran output (e.g.,)
  - Stress=3.500×10^4
  - (finite precision)

- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

What is a good algorithm?

- Flat section:



- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

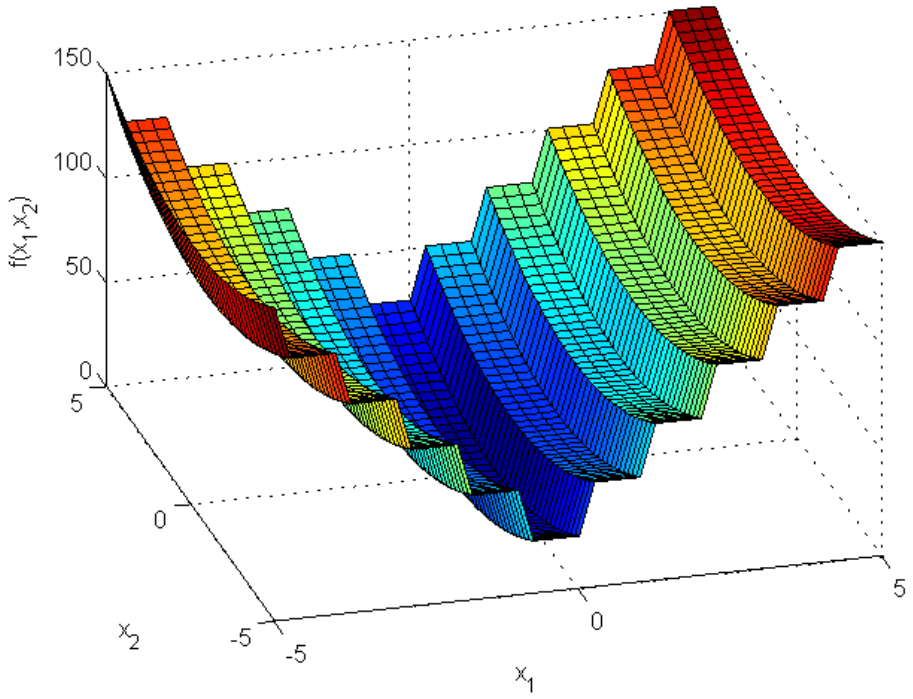
What is a good algorithm?

$$\min c^T \mathbf{x}$$
$$\text{s.t. } A\mathbf{x}=b$$

- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

What is a good algorithm?

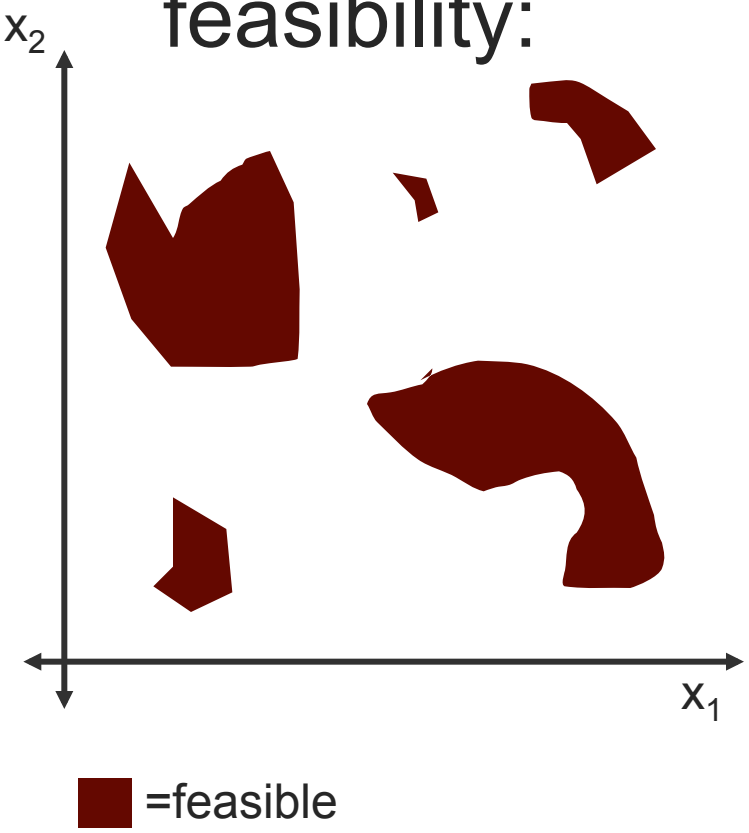
Nonsmooth objective:



- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

What is a good algorithm?

Islands of feasibility:



- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

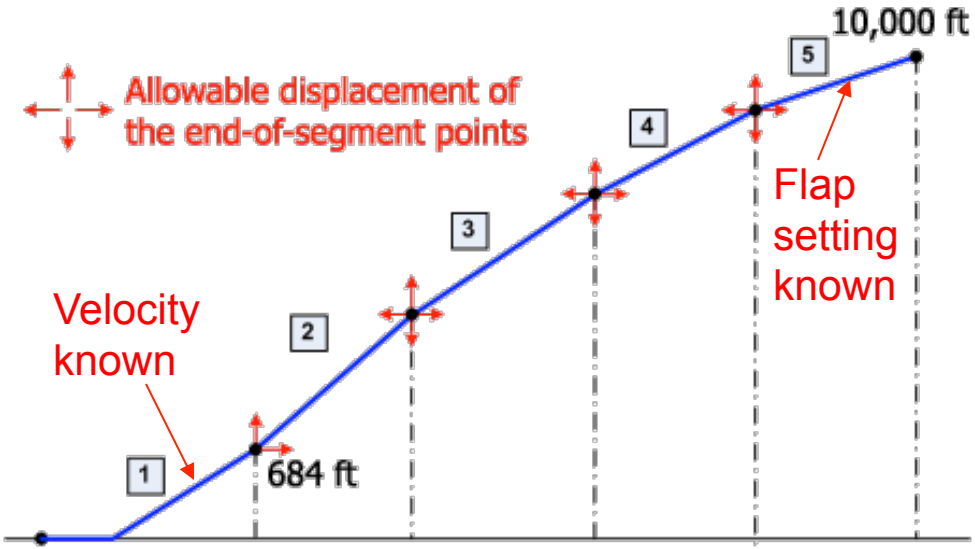
What is a good algorithm?

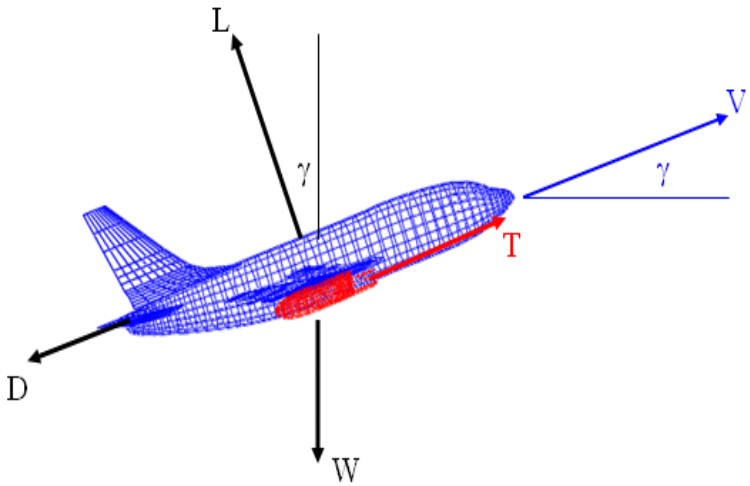
- Problem aspects:
  - Islands of feasibility
  - Many local minima
  - Mixed discrete/ continuous variables
  - Many design variable scales ($10^{-1} \rightarrow 10^{4}$)
  - Long function evaluation time (~2 minutes)

- Steepest descent
- Conjugate gradient
- Newton's Method
- Quasi-Newton
- SQP
- Compass-Search
- Nelder-Mead Simplex
- SA
- GA
- Tabu
- PSO

Example:

- Objectives
  - Time to Climb, Fuel Burn, Noise, Operating Cost
- Design variables
  - Flap setting
  - Throttle setting
  - Velocity
  - Transition Altitude
  - Climb gradient*
  - **18 Total**
- Constraints:
  - Regulations
    - No pilot input below 684 ft
    - Initial climb at V_2+15kts
  - Flap settings
  - Velocity
    - Min: stall
    - Max: max q
  - Throttle
    - Min: engine idle or positive rate of climb
    - Max: full power





Example:

- Exploration Challenges
  - Islands of feasibility
  - Many local minima
  - Mixed discrete/continuous variables
  - Many design variable scales ($10^{-1} \rightarrow 10^{4}$)
  - Long function evaluation time (~2 minutes with noise)

- Sequential Quadratic Programming [Climb time: 312 s]
  - Stuck at local minima
  - Can't handle discrete integers
- Direct Search (Nelder-Mead) [Climb time: 319 s]
  - Similar problems as SQP, but worse results
- Particle Swarming Optimization [Climb time: 319 s]
  - Slow running (8-12 hours), optimum not as good as Genetic Algorithm
- Genetic Algorithm [Climb time: 308 s]
  - No issues with any of the challenges of this problem.
  - No convergence guarantee and SLOW! Run-time ~24 hours.
  - But, best result.

Summary

- You have a large algorithm toolbox.
- You can often tell by inspection what algorithm might work well.
- Try to take advantage of aspects of your problem that will speed convergence.

Isoperformance and Goal programming

Why not performance-optimal?

"The experience of the 1960's has shown that for military aircraft the cost of the final increment of performance usually is excessive in terms of other characteristics and that the overall system must be optimized, not just performance"

Ref: Current State of the Art of Multidisciplinary Design Optimization (MDO TC) - AIAA White Paper, Jan 15, 1991

Industry designs not for optimal performance, but according to targets specified by a requirements document or contract - thus, optimize design for a set of GOALS.

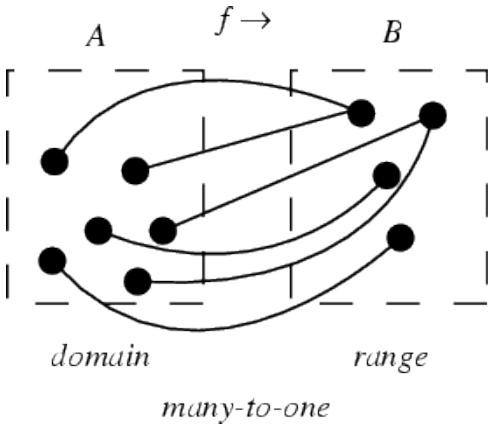
Lecture Outline

- Motivation - why goal programming ?
- Example: Goal Seeking in Excel
- Case 1: Target vector **T** in Range
  = Isoperformance
- Case 2: Target vector **T** out of Range
  = Goal Programming
- Application to Spacecraft Design
- Stochastic Example: Baseball

Target Vector

T

x **J**



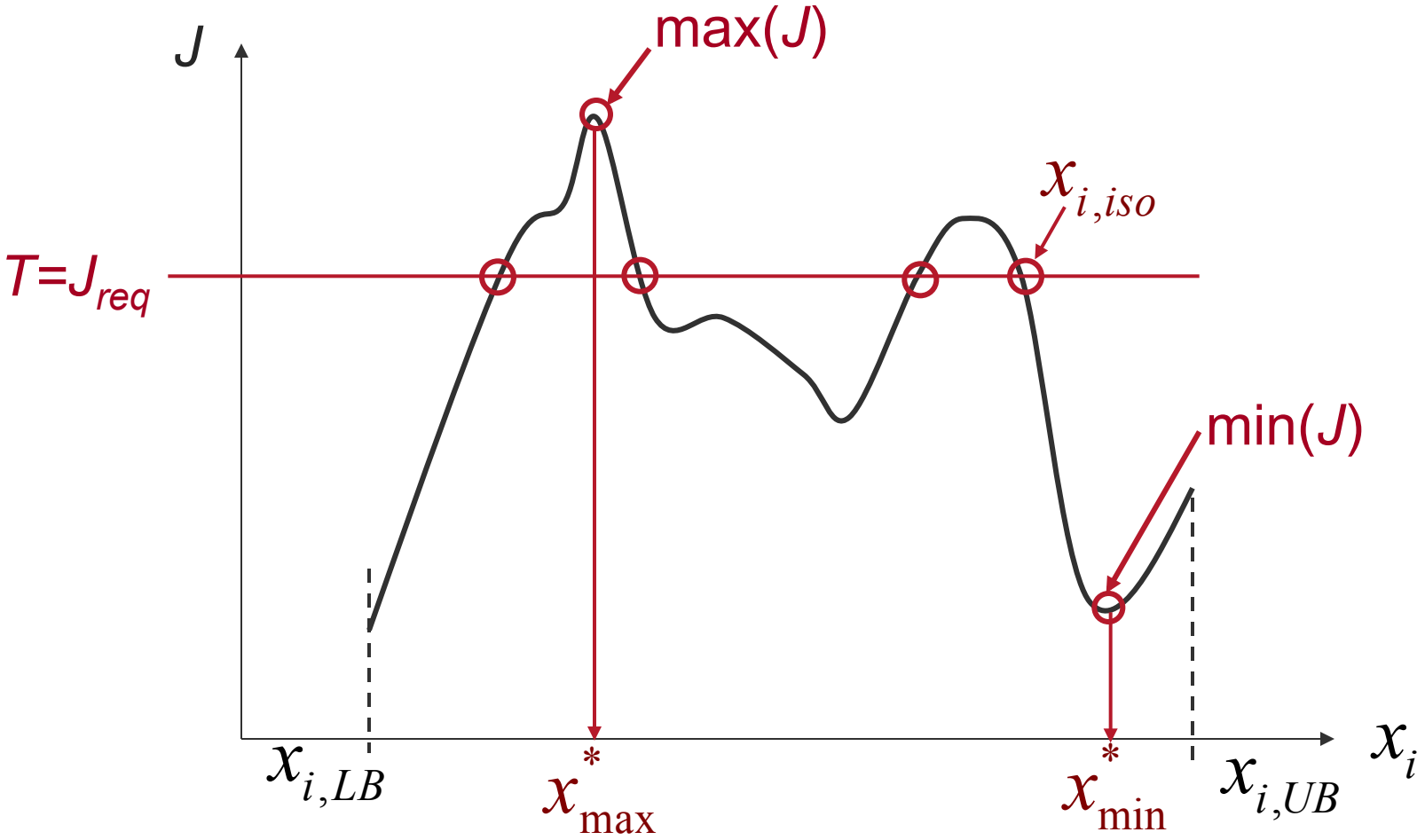
Forward Perspective

Choose **x** ⟶ What is **J** ?

Reverse Perspective

Choose **J** ⟶ What **x** satisfy this?

Goal Seeking



Excel: Tools – Goal Seek

About Goal Seek

Goal Seek is part of a suite of commands sometimes called what-if analysis tools. When you know the desired result of a single formula but not the input value the formula needs to determine the result, you can use the Goal Seek feature available by clicking **Goal Seek** on the **Tools** menu. When goal seeking, Microsoft Excel varies the value in one specific cell until a formula that's dependent on that cell returns the result you want.

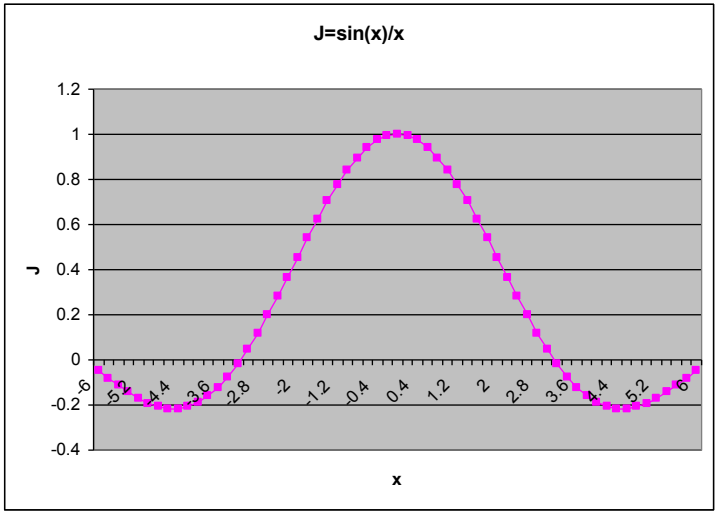
The value in cell B4 is the result of the formula =PMT(B3/12,B2,B1).

	A	B
1	Loan Amount	$ 100,000
2	Term in Months	180
3	Interest Rate	7.02%
4	Payment	($900.00)

Goal seek to determine the interest rate in cell B3 based on the payment in cell B4.

For example, use Goal Seek to change the interest rate in cell B3 incrementally until the payment value in B4 equals $900.00.

Excel - example



sin(x)/x - example

- single variable x
- no solution if *T* is out of range

Goal Seeking and Equality Constraints

- Goal Seeking – is essentially the same as finding the set of points **x** that will satisfy the following "soft" equality constraint on the objective:

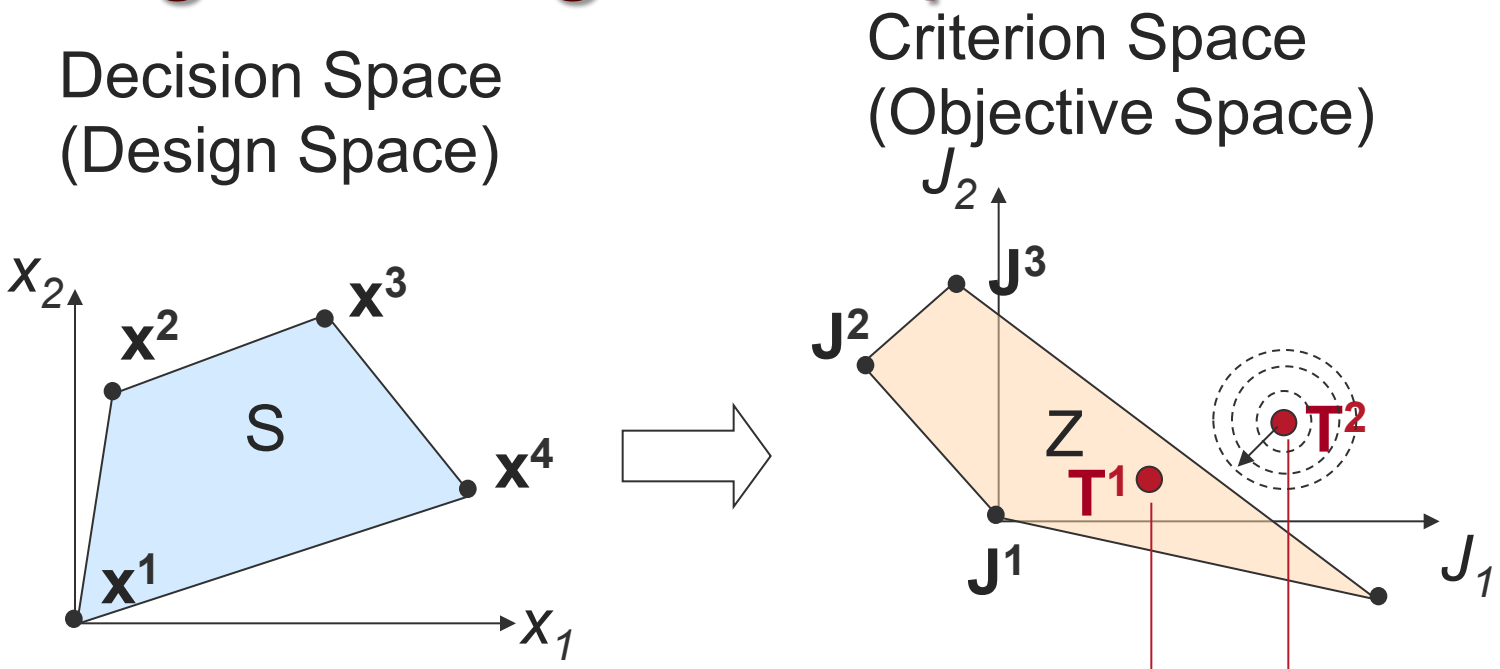
$$\text{Find all } \mathbf{x} \text{ such that } \left| \frac{J(\mathbf{x}) - J_{req}}{J_{req}} \right| \leq \varepsilon$$

Example Target Vector:

$$J_{req}(x) = \begin{bmatrix} m_{sat} \\ R_{data} \\ C_{sc} \end{bmatrix} \equiv \begin{bmatrix} 1000kg \\ 1.5Mbps \\ 15M\$ \end{bmatrix}$$

← Target mass
← Target data rate
← Target Cost

Goal Programming vs. Isoperformance



Case 1: The target (goal) vector is in Z - usually get non-unique solutions = Isoperformance

Case 2: The target (goal) vector is not in Z - don't get a solution - find closest = Goal Programming

Isoperformance Analogy

Non-Uniqueness of Design if n > z

Performance: Buckling Load

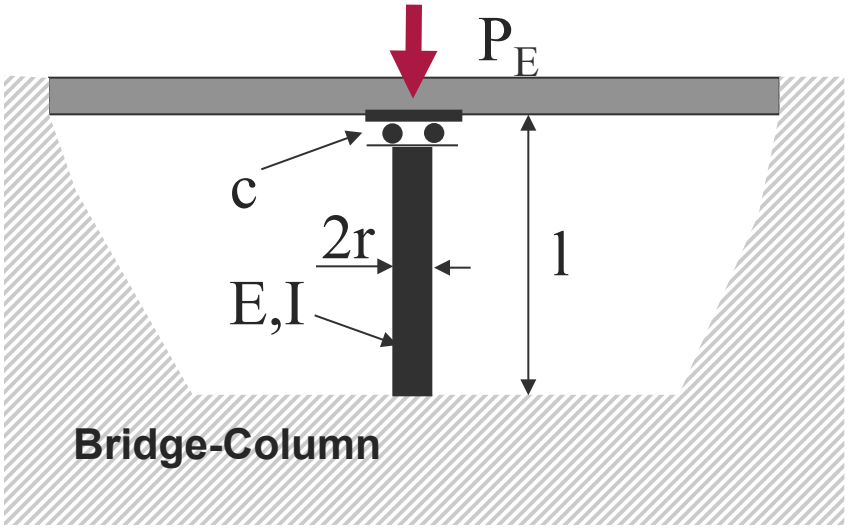
Constants: l=15 [m], c=2.05

Variable Parameters: E, $I(r)$

$$P_E = \frac{c\pi^2 EI}{l^2}$$

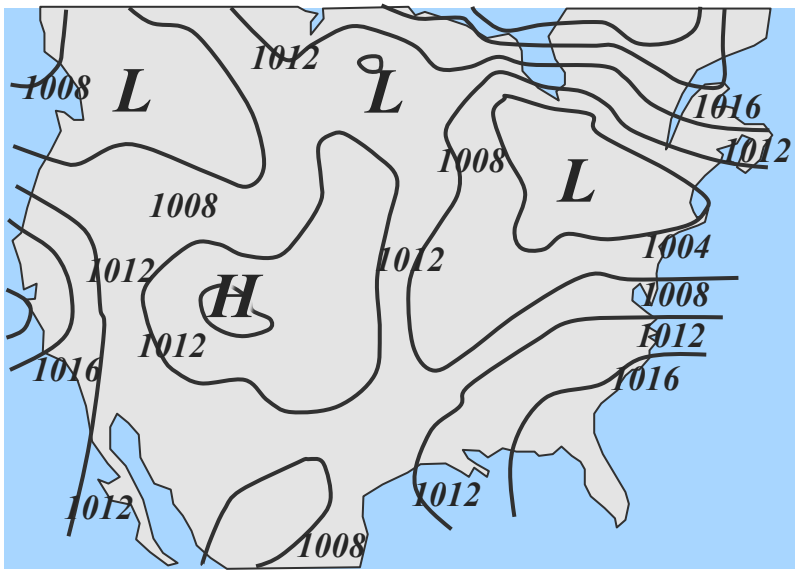
Requirement: $P_{E,REQ} = 1000$ metric tons

Solution 1: V2A steel, r=10 cm , E=19.1e+10
Solution 2: Al(99.9%), r=12.8 cm, E=7.1e+10



Analogy: Sea Level Pressure [mbar]
Chart: 1600 Z, Tue 9 May 2000

Isobars = Contours of Equal Pressure
Parameters = Longitude and Latitude

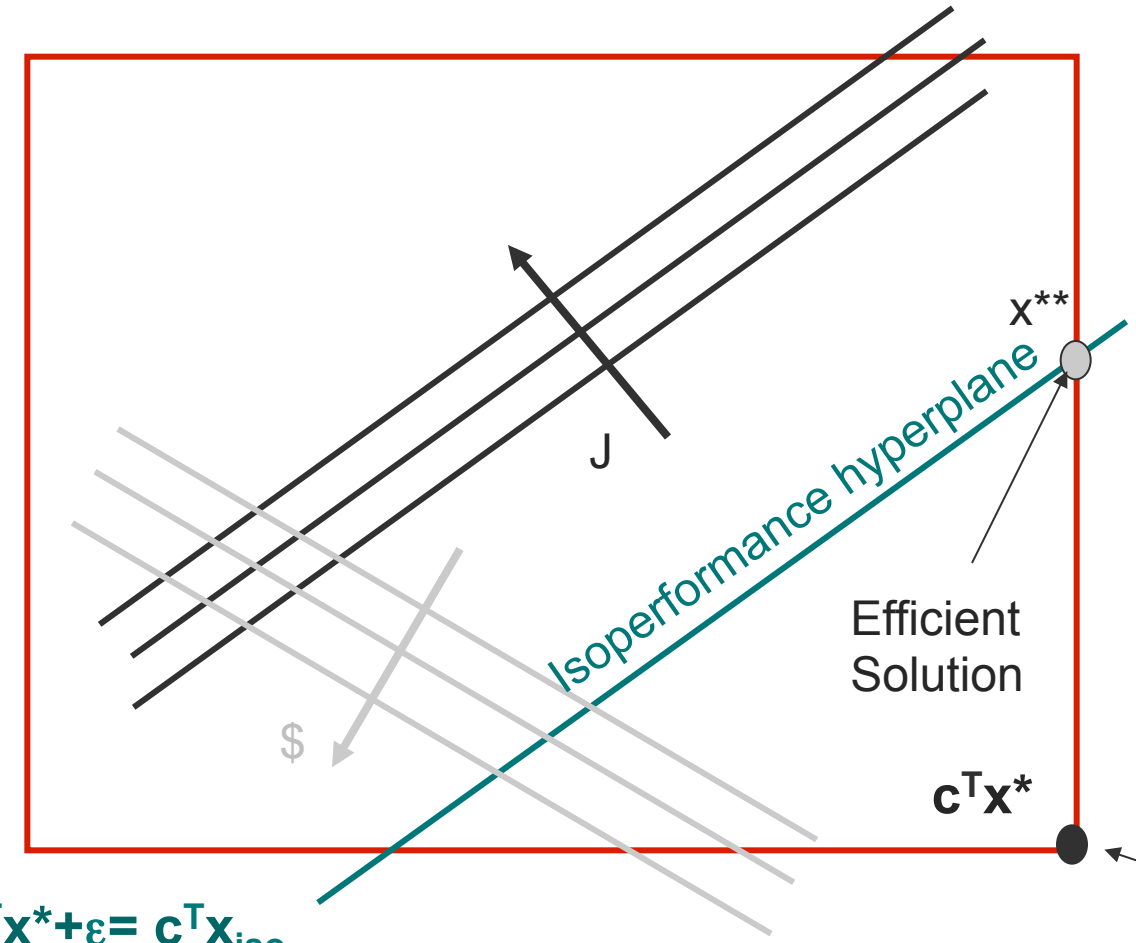


Isoperformance Contours = Locus of constant system performance

Isoperformance and LP

- In LP the isoperformance surfaces are hyperplanes
- Let $c^T x$ be performance objective and $k^T x$ a cost objective

$$\min \quad c^T x$$
$$s.t. \quad x_{LB} \le x \le x_{UB}$$



1. Optimize for performance $\mathbf{c^T x^*}$
2. Decide on acceptable performance penalty ε
3. Search for solution on isoperformance hyperplane that minimizes cost $\mathbf{k^T x^*}$

Performance Optimal Solution

Example

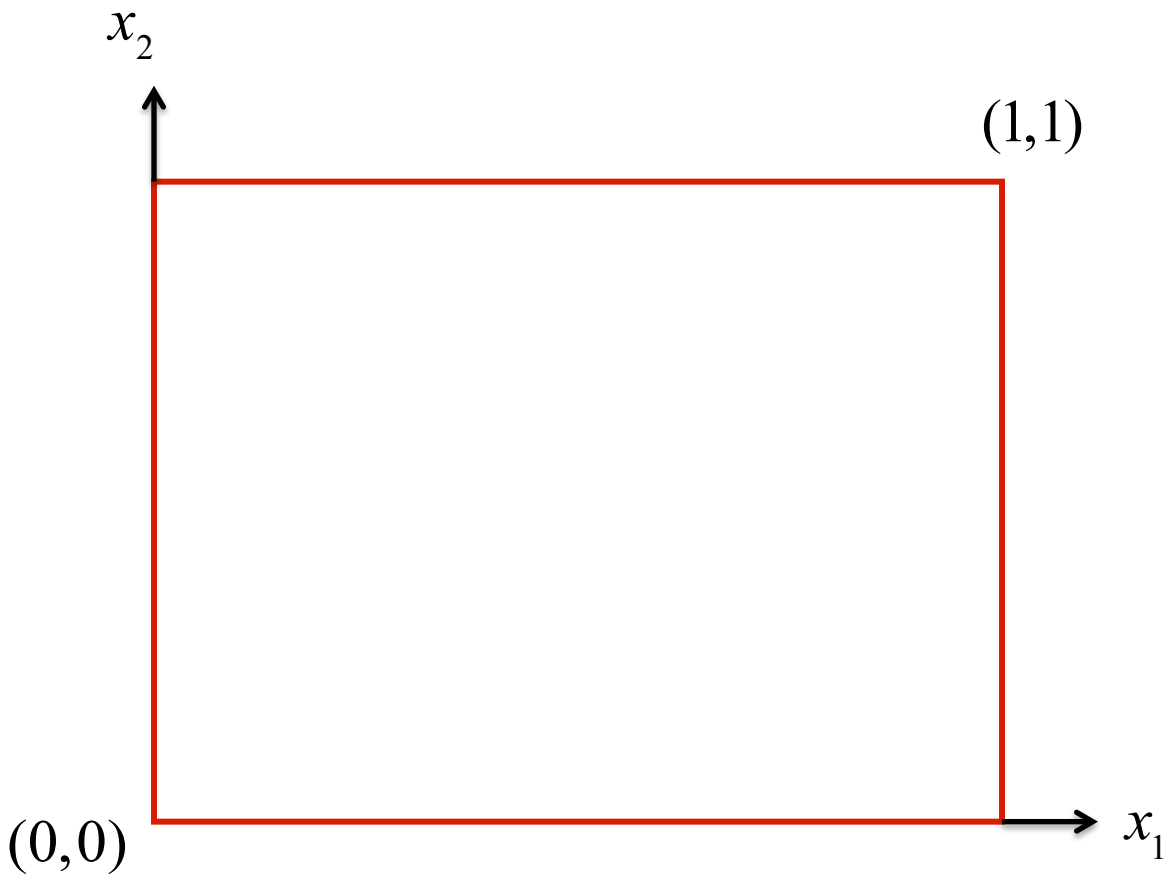
- Draw 3 contours of J
- Solve problem for J^* and $\mathbf{x}^*$

Let $\mathbf{c}^T = [-1, 1]$

$$\min \quad J = \mathbf{c}^T \mathbf{x}$$

$$s.t. \quad 0 \le x_1 \le 1$$

$$0 \le x_2 \le 1$$

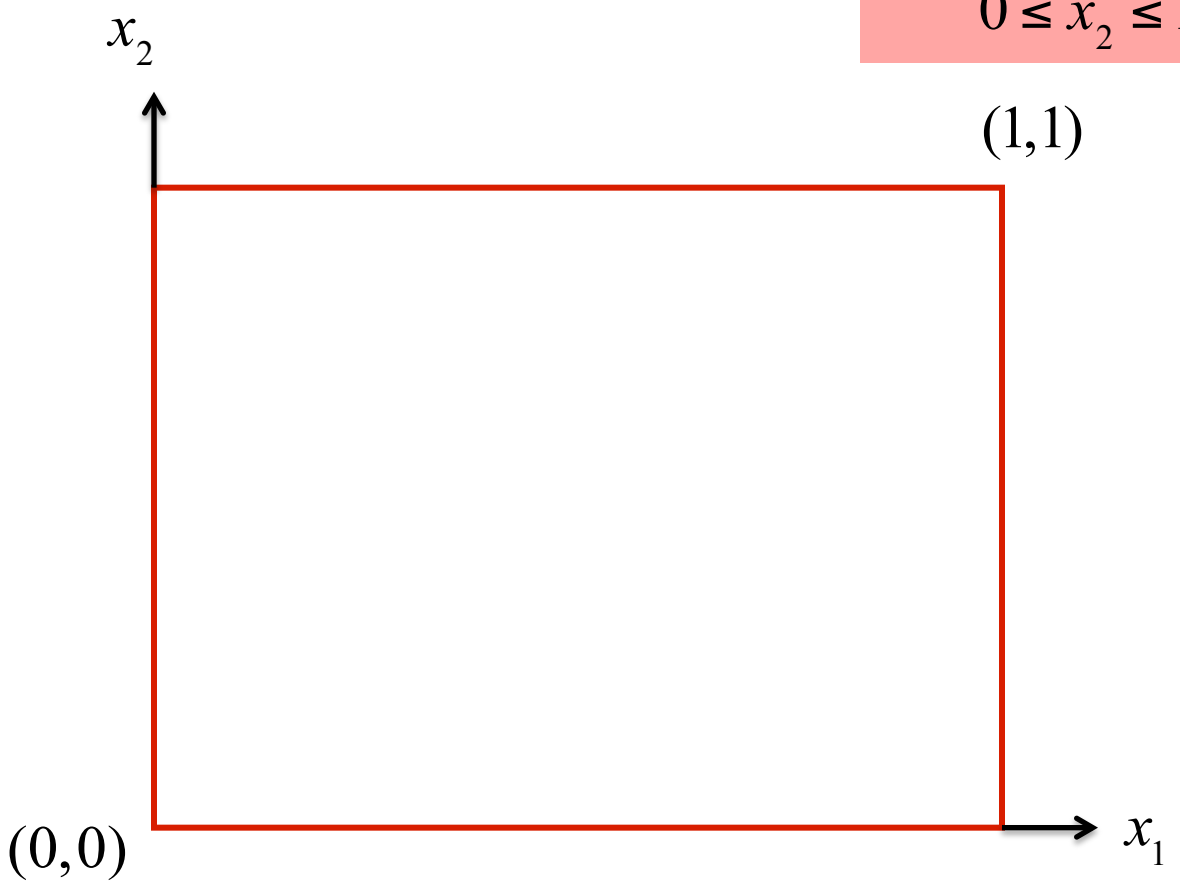


Example

- Draw 3 contours of J_c
- Draw the isoperformance plane
- Solve problem for J_c^* and $\mathbf{x}_{iso}^*$

$$\text{Let} \quad \mathbf{c}^T = [-1,1]$$
$$\min \quad J = \mathbf{c}^T \mathbf{x}$$
$$s.t. \quad 0 \le x_1 \le 1$$
$$0 \le x_2 \le 1$$

$$\text{Let} \quad \mathbf{k}^T = [1,1]$$
$$\min \quad J_c = \mathbf{k}^T \mathbf{x}$$
$$s.t. \quad \mathbf{c}^T \mathbf{x}^* + \varepsilon = \mathbf{c}^T \mathbf{x}_{iso}$$
$$\text{where} \quad \varepsilon = 0.25$$



Problem Statement

Find the performance invariant set

Find all $\mathbf{x}_{iso}^{k} \subseteq \mathbf{X}_{iso}$

Such that

$$\left| \frac{J(\mathbf{x}_{iso}^{k}) - J_{req}}{J_{req}} \right| \leq \tau$$

$$\mathbf{g}(\mathbf{x}_{iso}^{k}) \leq 0$$

$$\mathbf{h}(\mathbf{x}_{iso}^{k}) = 0$$

$$x_{i,LB} \leq x_{iso,i}^{k} \leq x_{i,UB} \quad \forall\ i = 1,2,...,n$$

Tolerance band

Find efficient subset

Find all $\mathbf{x}_{iso}^{l*} \in \mathbf{X}_{iso}^{*} \subseteq \mathbf{X}_{iso}$

Such that there exists

no other feasible $\mathbf{x} \in \mathbf{X}_{iso}$

s.t. $J_c(\mathbf{x}) \leq J_c(\mathbf{x}_{iso}^{*})$

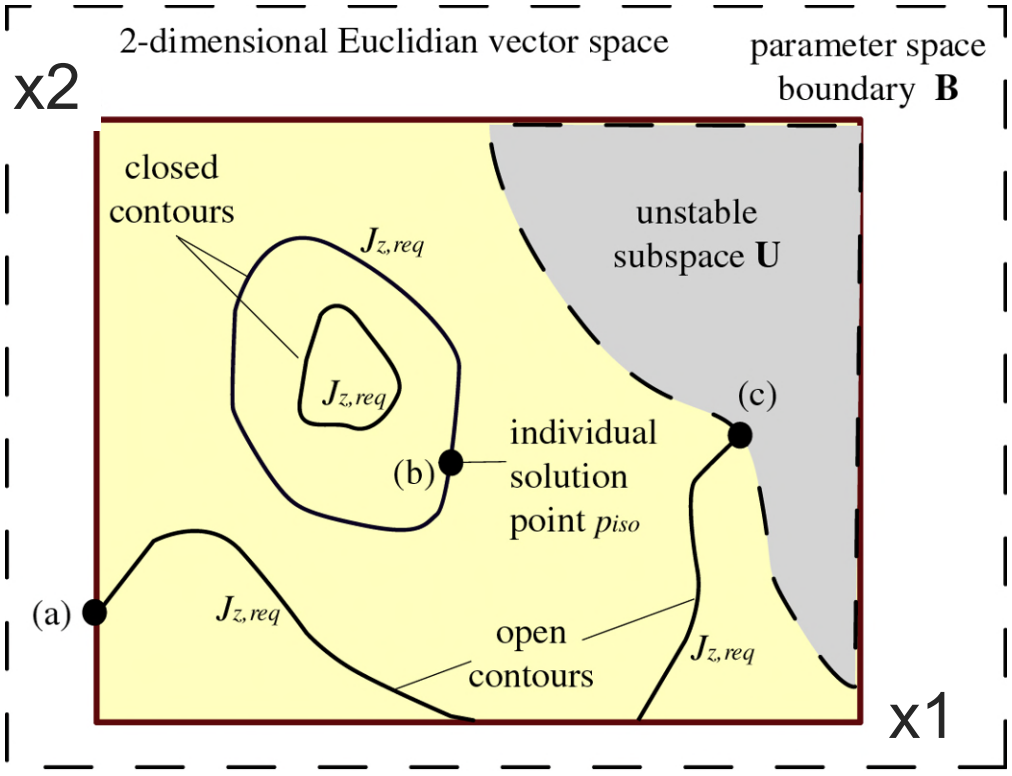
Cost or risk based secondary objective

Bivariate Exhaustive Search (2D)

"Simple" Start: Bivariate Isoperformance Problem

Performance $J_z(x_1, x_2): \; z = 1$

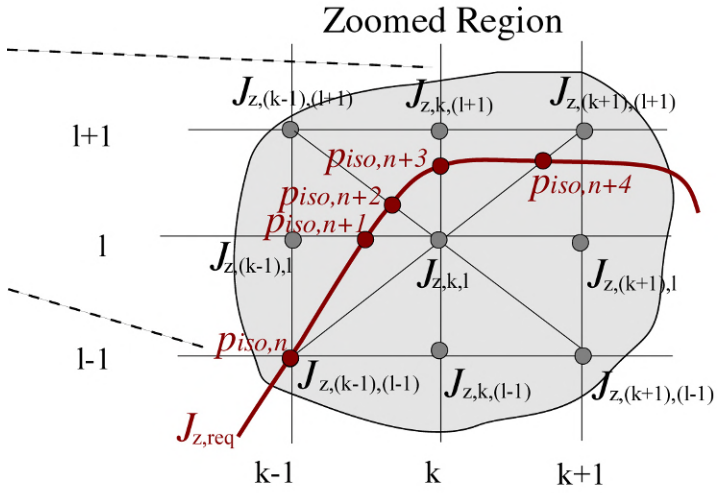
Variables $x_j, j = 1,2: \quad n = 2$



First Algorithm: **Exhaustive Search** coupled with bilinear interpolation

Number of points along j-th axis:

$$n_j = \left\lceil \frac{x_{j,UB} - x_{j,LB}}{\Delta x} \right\rceil$$



Can also use standard contouring code like MATLAB `contourc.m`

Isoperformance with Stochastic Data

Example: Baseball season is starting soon!

What determines success of a team?

Pitching 

ERA

"Earned Run Average"

Batting 

RBI

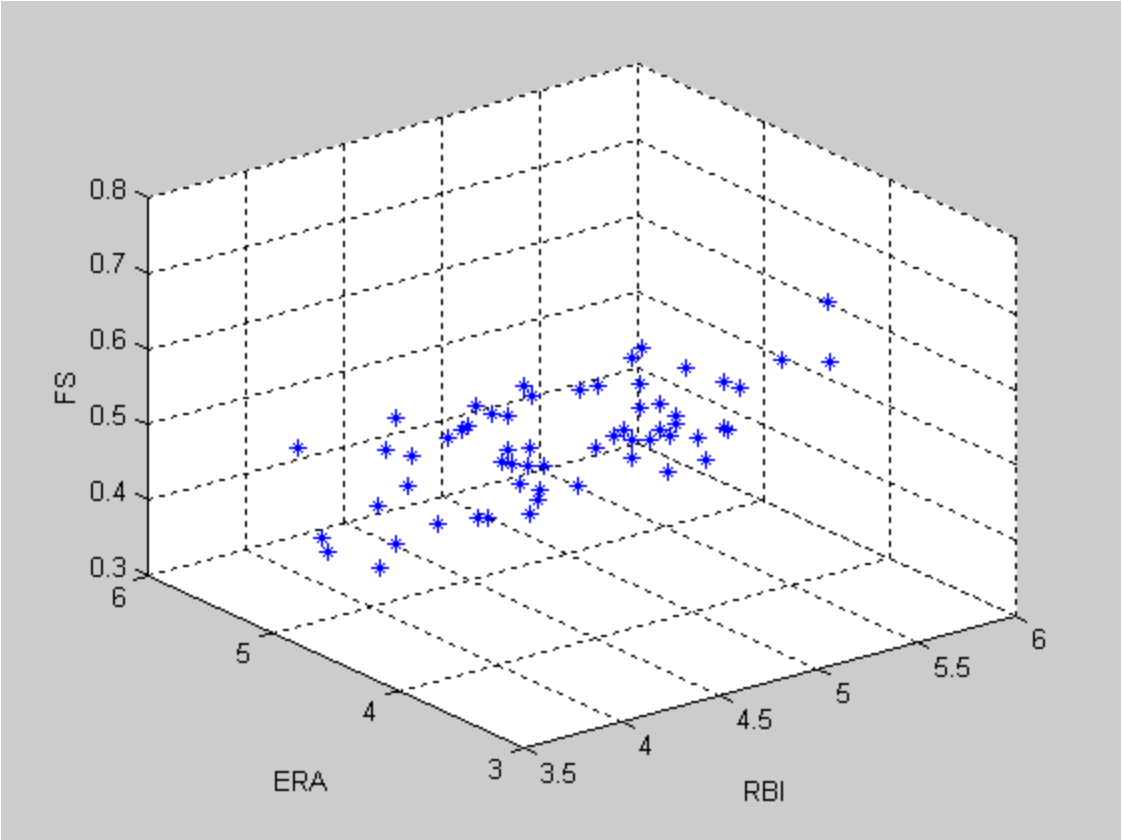
"Runs Batted In"

How is success of team measured ?

FS= Wins/Decisions

Raw Data

Team results for 2000, 2001 seasons: RBI,ERA,FS



Stochastic Isoperformance (I)

Step-by-step process for obtaining (bivariate) isoperformance curves given statistical data:

Starting point, need:

- Model - derived from empirical data set

- (Performance) Criterion

- Desired Confidence Level

Model

Step 1: Obtain an expression from model for expected performance of a "system" for individual design i as a function of design variables $x_{1,I}$ and $x_{2,i}$

1.1 assumed model

$$E\left[J_i\right] = a_0 + a_1\left(x_{1,i}\right) + a_2\left(x_{2,i}\right) + a_{12}\left(x_{1,i} - \overline{x_1}\right)\left(x_{2,i} - \overline{x_2}\right) \qquad (1)$$

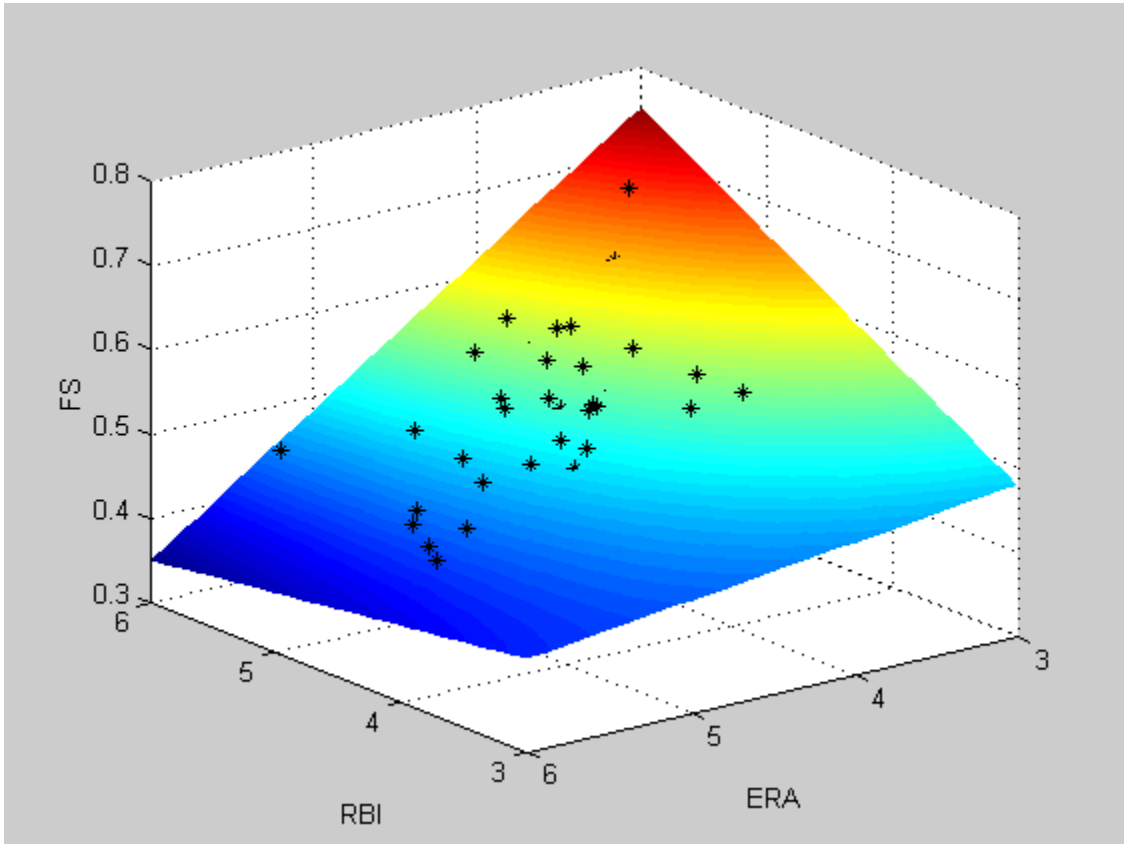
1.2 model fitting

E.g. use Matlab fminunc.m for optimal surface fit or least squares regression

Baseball: Obtain an expression for expected final standings (FS_i) of individual Team *i* as a function of RBI_i and ERA_i

$$E\left[FS_i\right] = a_0 + a_1\left(RBI_i\right) + a_2\left(ERA_i\right) + a_{12}\left(RBI_i - \overline{RBI}\right)\left(ERA_i - \overline{ERA}\right)$$

Fitted Model

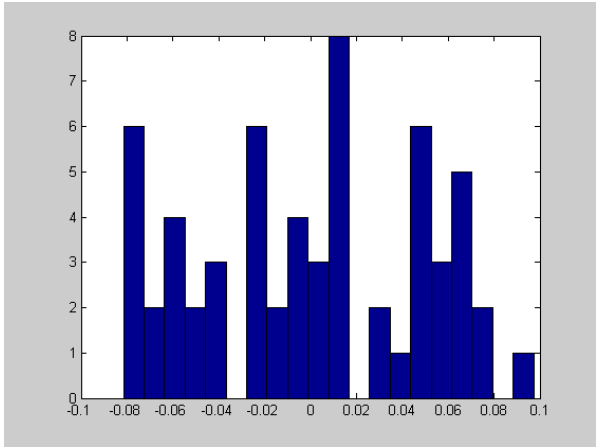


Coefficients:

ao=0.7450
a1=0.0321
a2=-0.0869
a12= -0.0369

RMSE:
Error
σ_e= 0.0493

Error Distribution



Expected Performance

Step 2: Determine expected level of performance for design i such that the probability of adequate performance is equal to specified confidence level

$$E\left[J_i\right] = J_{req} + z\sigma_{\varepsilon} \qquad (2)$$

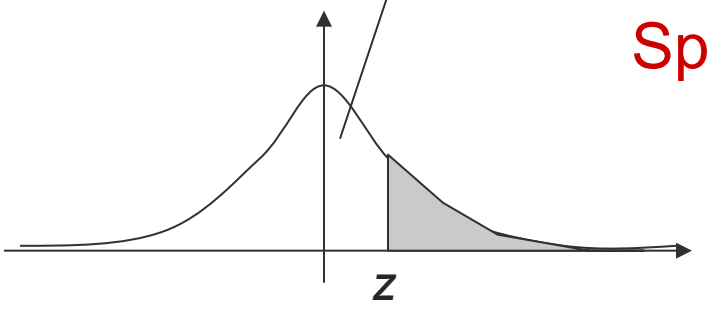
Required performance level

Error Term (total variance)

Confidence level normal variable z (Lookup Table)

Specify

$\Phi(z)$



$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-\frac{z^2}{2}} dz$$

Expected Performance

Baseball:

Performance criterion

- User specifies a final desired standing of FS_i=0.550

Confidence Level

- User specifies a .80 confidence level that this is achieved

Spec is met if for Team *i*:

From normal table lookup

Error term from data

$$E\left[FS_i\right] = .550 + z\sigma_r = .550 + 0.84\left(0.0493\right) = 0.5914$$

If the final standing of team *i* is to equal or exceed .550 with a probability of .80, then the expected final standing for Team i must equal 0.5914

Get Isoperformance Curve

Step 3: Put equations (1) and (2) together

$$J_{req} + z\sigma_r = E[J_i] = a_0 + a_1(x_{1,i}) + a_2(x_{2,i}) + a_{12}(x_{1,i} - \overline{x_1})(x_{2,i} - \overline{x_2}) \quad (3)$$

- Four constant parameters: a_o, a_1, a_2, a_{12}
- Two sample statistics: $\overline{x}_1, \overline{x}_2$
- Two design variables: $x_{1,i}$ and $x_{2,i}$

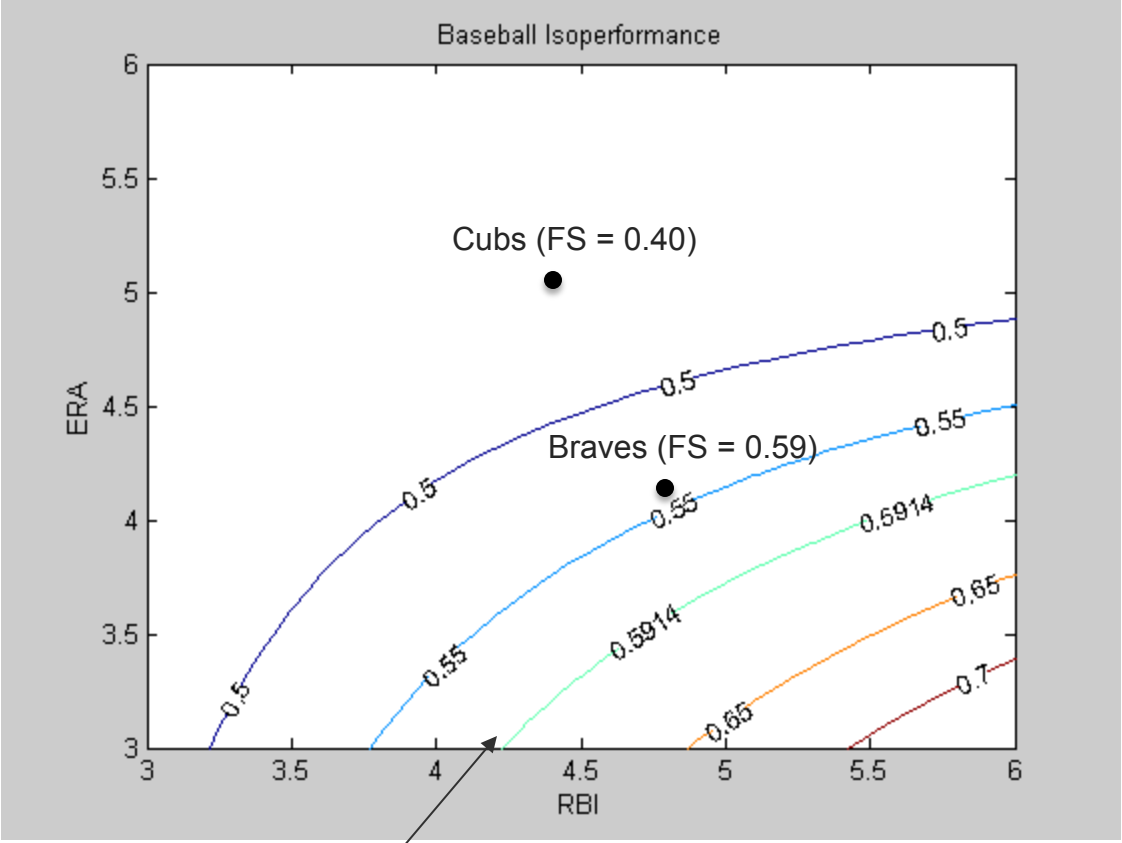
Then rearrange: $x_{2,i} = f(x_{1,i})$

Baseball:

$$RBI_i = \frac{.5914 - a_0 - a_1 ERA_i + a_{12}\overline{RBI}(ERA_i - \overline{ERA})}{a_1 + a_{12}(ERA_i - \overline{ERA})}$$

Equation for isoperformance curve

Stochastic Isoperformance



This is our desired tradeoff curve

Lecture Summary

- Traditional process goes from design space **x** → objective space **J** (forward process)
- Many systems are designed to meet "targets"
  - Performance, Cost, Stability Margins, Mass …
- Methodological Options
  - Formulate optimization problem with equality constraints given by targets
  - Goal Programming minimizes the "distance" between a desired "target" state and the achievable design
  - Isoperformance finds a set of (non-unique) performance invariant solutions → multiple solutions
- Isoperformance works backwards from objective space **J** → design space **x** (reverse process)
  - Deterministically
  - Stochastically

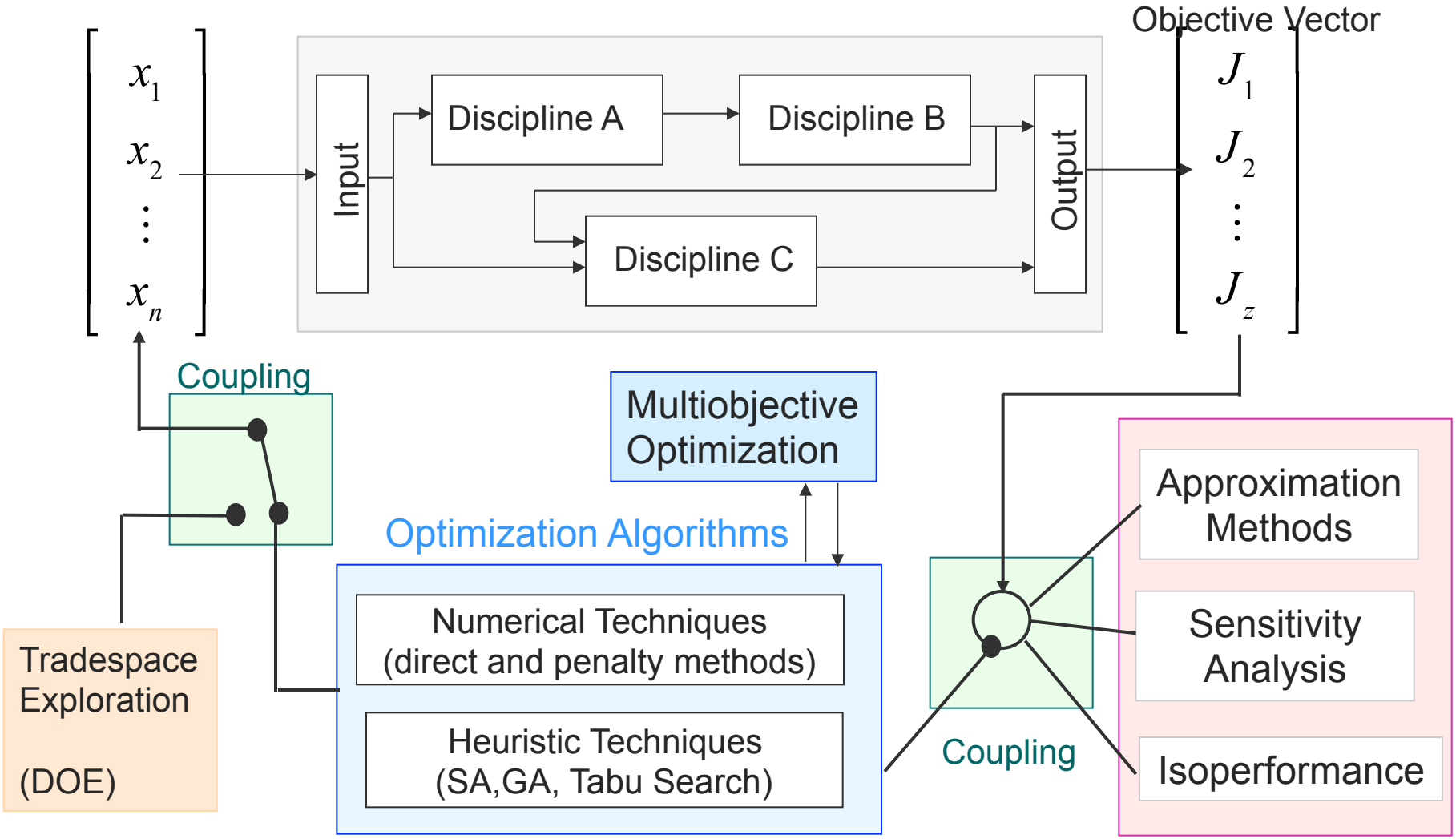
Lecture 13: Multidisciplinary System Analysis and Design Optimization (MSADO)

Multiobjective Optimization (1)

March 23, 2015

Prof. Douglas Allaire

Where in the Framework?



Lecture Outline

- Why multiobjective optimization?
- Example – twin peaks optimization
- History of multiobjective optimization
- Weighted Sum Approach
- Pareto-Optimality
- Dominance and Pareto Filtering

Multiobjective Optimization Problem Formal Definition

Design problem may be formulated as a problem of Nonlinear Programming (NLP). When Multiple objectives (criteria) are present we have a MONLP

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

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

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

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

where

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

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

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

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

Multiple Objectives

The objective can be a vector **J** of *z* system responses or characteristics we are trying to maximize or minimize

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

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

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

Objectives are usually conflicting.

Why multiobjective optimization?

While multidisciplinary design can be associated with the traditional disciplines such as aerodynamics, propulsion, structures, and controls there are also the lifecycle areas of manufacturability, supportability, and cost which require consideration.

After all, it is the balanced design with equal or weighted treatment of performance, cost, manufacturability and supportability which has to be the ultimate goal of multidisciplinary system design optimization.

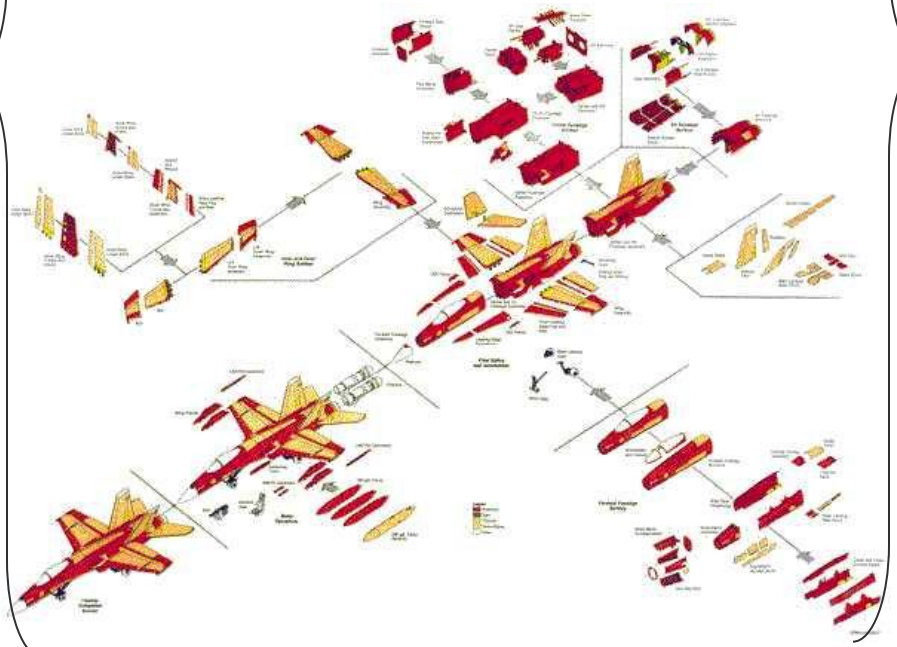
Design attempts to satisfy multiple, possibly conflicting objectives at once.

Example F/A-18 Aircraft

Design Decisions

Aspect Ratio
Dihedral Angle
Vertical Tail Area
Engine Thrust
Skin Thickness

# of Engines
Fuselage Splices
Suspension Points
Location of Mission Computer
Access Door Locations



Objectives

Speed
Range
Payload Capability
Radar Cross Section
Stall Speed
Stowed Volume

Acquisition cost
Cost/Flight hour
MTBF
Engine swap time
Assembly hours

Avionics growth Potential

Multiobjective Examples

Design Optimization

Aircraft Design

max {range}
max {passenger volume}
max {payload mass}
min {specific fuel consumption}
max {cruise speed}
min {lifecycle cost}

$$\mathbf{J} = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_z \end{bmatrix}$$

Operations Research

Production Planning

max {total net revenue}
max {min net revenue in any time period}
min {backorders}
min {overtime}
min {finished goods inventory}

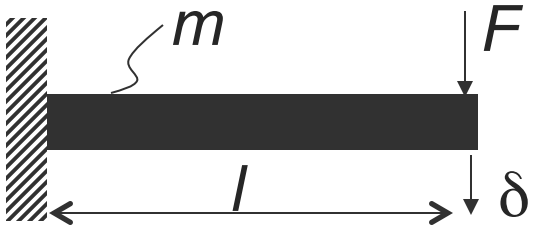
Multiobjective vs. Multidisciplinary

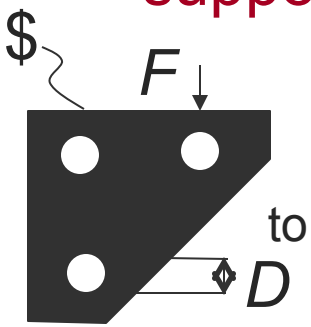
- Multiobjective Optimization
  - Optimizing conflicting objectives
  - e.g., Cost, Mass, Deformation
  - Issues: Form Objective Function that represents designer preference! Methods used to date are largely primitive.

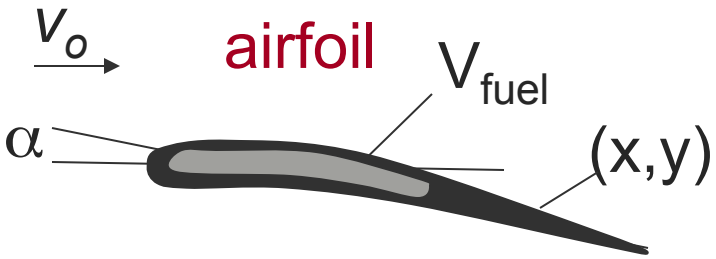
- Multidisciplinary Design Optimization
  - Optimization involves several disciplines
  - e.g. Structures, Control, Aero, Manufacturing
  - Issues: Human and computational infrastructure, cultural, administrative, communication, software, computing time, methods

- All optimization is (or should be) multiobjective
  - Minimizing mass alone, as is often done, is problematic

Multidisciplinary vs. Multiobjective

<table>
<tr><td></td><td>single discipline</td><td>multiple disciplines</td></tr>
<tr><td>single objective</td><td>cantilever beam
 

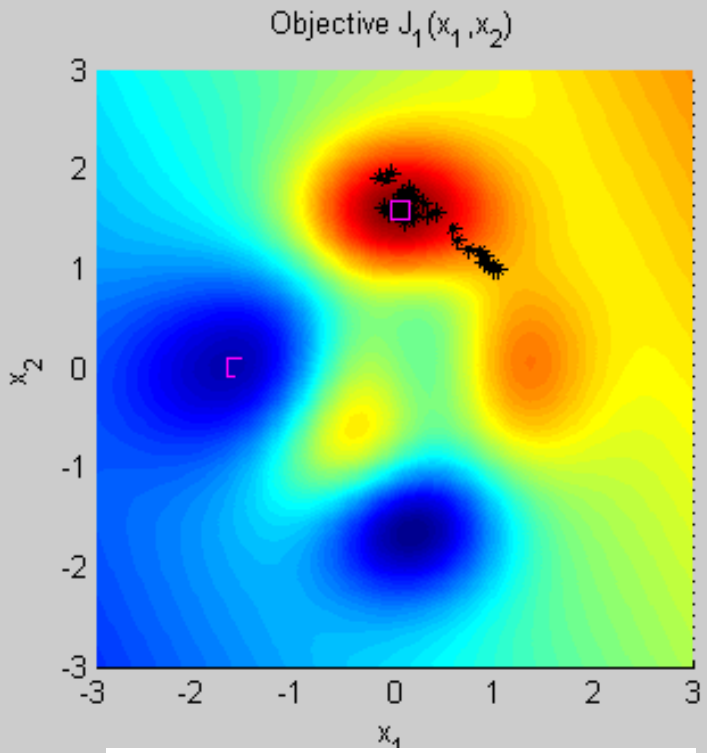
Minimize displacement
s.t. mass and loading constraint</td><td>support bracket
 

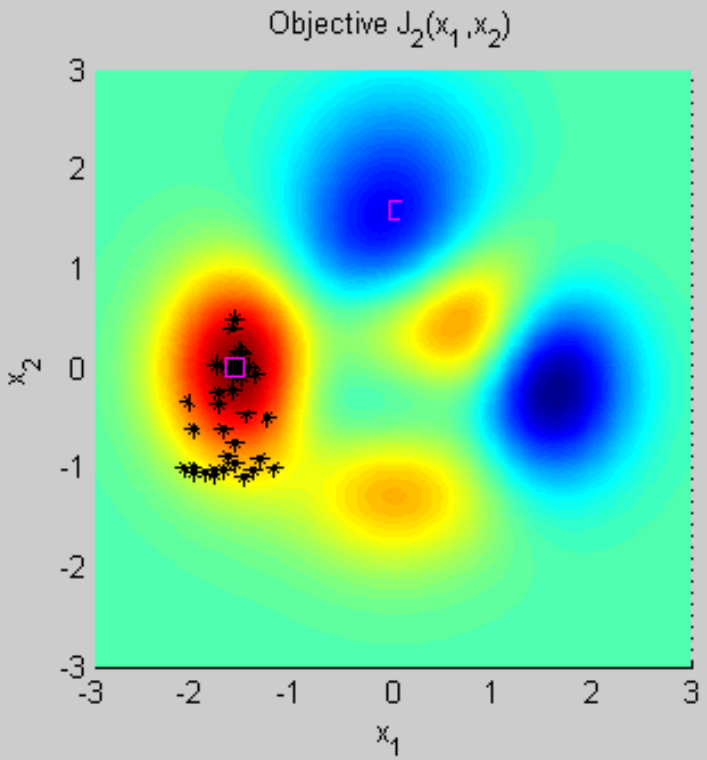
Minimize stamping costs (mfg) subject to loading and geometry constraint</td></tr>
<tr><td>multiple obj.</td><td>airfoil
 

Maximize C_L/C_D and maximize wing fuel volume for specified α, v_o</td><td>commercial aircraft
 
Minimize SFC and maximize cruise speed s.t. fixed range and payload</td></tr>
</table>

Example:

Objective: max **J**= [J_1 J_2]^T (demo)





$$J_1 = 3(1-x_1)^2 e^{-x_1^2-(x_2+1)^2}$$

$$-10\left(\frac{x_1}{5} - x_1^3 - x_2^5\right)e^{-x_1^2-x_2^2}$$

$$-3e^{-(x_1+2)^2-x_2^2} + 0.5(2x_1 + x_2)$$

$$J_2 = 3(1+x_2)^2 e^{x_2^2-(x_1+1)^2}$$

$$-10\left(-\frac{x_2}{5} + x_2^3 - x_1^5\right)e^{x_2^2-x_1^2} - 3e^{-(2-x_2)^2-x_1^2}$$

Double peaks optimization

Optimum for J_1 alone:

$$\mathbf{x}^{1*} = \begin{Bmatrix} 0.0532 \\ 1.5973 \end{Bmatrix}$$

$J_1^* = 8.9280$

$J_2(\mathbf{x}^{1*}) = -4.8202$

Optimum for J_2 alone:

$$\mathbf{x}^{2*} = \begin{Bmatrix} -1.5808 \\ 0.0095 \end{Bmatrix}$$

$J_1(\mathbf{x}^{2*}) = -6.4858$

$J_2^* = 8.1118$

Each point **x1*** and **x2*** optimizes objectives J_1 and J_2 individually. Unfortunately, at these points the other objective exhibits a low objective function value. There is no single point that simultaneously optimizes both objectives J_1 and J_2 !

Tradeoff between J_1 and J_2

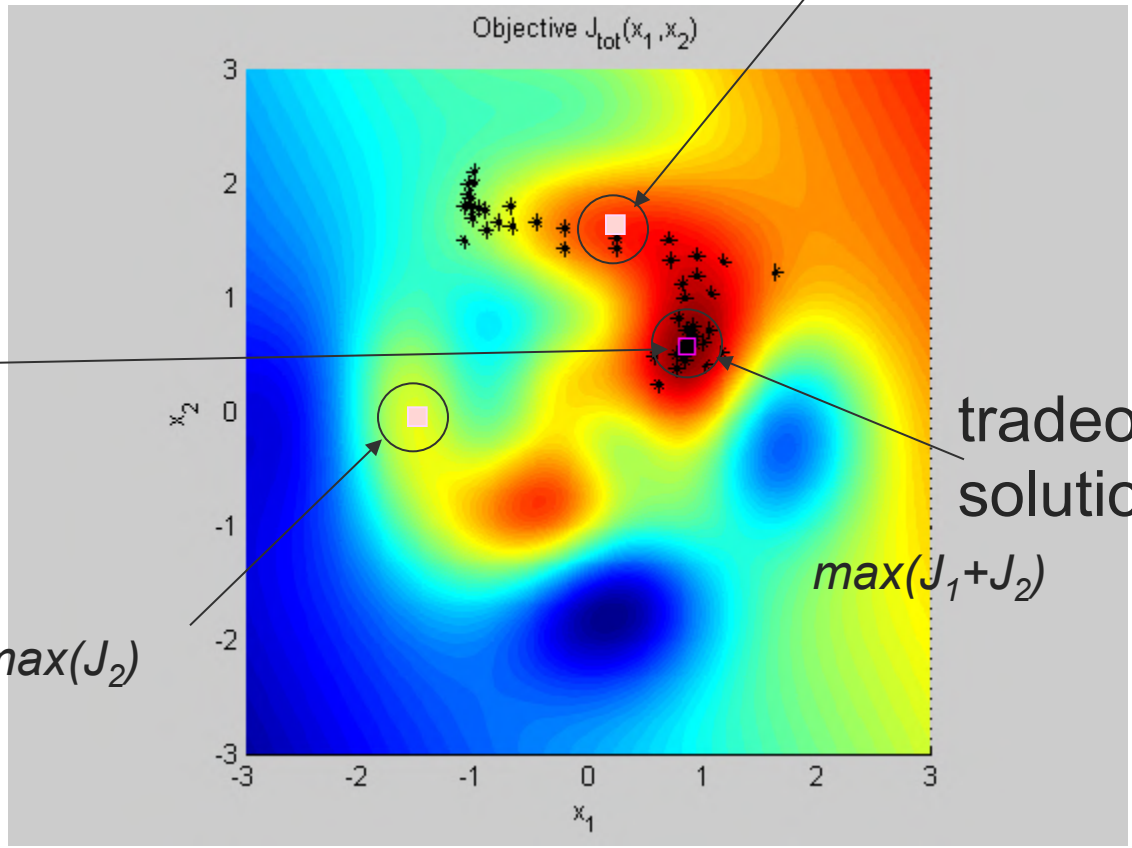
- Want to do well with respect to <u>both</u> J_1 and J_2
- Define new objective function: $J_{tot}=J_1 + J_2$
- Optimize J_{tot}

Result:

$$\mathbf{X}^{tot*} = \begin{Bmatrix} 0.8731 \\ 0.5664 \end{Bmatrix}$$

$$J_{tot}^* = 6.1439$$

$$\mathbf{J}(\mathbf{x}^{tot*}) = \begin{Bmatrix} 3.0173 \\ 3.1267 \end{Bmatrix} = \begin{Bmatrix} J1 \\ J2 \end{Bmatrix}$$



History

- Life is about making decisions. Most people attempt to make the "best" decision within a specified set of possible decisions.
- In 1881, King's College (London) and later Oxford Economics Professor **F.Y. Edgeworth** is the first to define an optimum for multicriteria economic decision making. He does so for the multiutility problem within the context of two consumers, P and π:
  - *"It is required to find a point (x,y,) such that in whatever direction we take an infinitely small step, P and π do not increase together but that, while one increases, the other decreases."*
  - **Reference:** Edgeworth, F.Y., *Mathematical Psychics,* P. Keagan, London, England, 1881.

History

- **Born in Paris in 1848** to a French Mother and Genovese Father
- Graduates from the **University of Turin** in 1870 with a degree in **Civil Engineering**
  - Thesis Title: "The Fundamental Principles of **Equilibrium** in Solid Bodies"
- While working in Florence as a Civil Engineer from 1870-1893, Pareto takes up the study of philosophy and politics and is one of the **first to analyze economic problems with mathematical tools**.
- In 1893, Pareto becomes the Chair of Political Economy at the **University of Lausanne in Switzerland**, where he creates his two most famous theories:
  - Circulation of the Elites
  - **The Pareto Optimum**
    - ***"The optimum allocation of the resources of a society is not attained so long as it is possible to make at least one individual better off in his own estimation while keeping others as well off as before in their own estimation."***
    - **Reference:** Pareto, V., *Manuale di Economia Politica*, Societa Editrice Libraria, Milano, Italy, 1906. Translated into English by A.S. Schwier as *Manual of Political Economy*, Macmillan, New York, 1971.

History

- After the translation of Pareto's ***Manual of Political Economy*** into English, **Prof. Wolfram Stadler** of San Francisco State University begins to apply the notion of Pareto Optimality to the fields of engineering and science in the **middle 1970's**.
- The applications of multi-objective optimization in engineering design grew over the following decades.
- **References:**
  - Stadler, W., "A Survey of Multicriteria Optimization, or the Vector Maximum Problem," Journal of Optimization Theory and Applications, Vol. 29, pp. 1-52, 1979.
  - Stadler, W. "Applications of Multicriteria Optimization in Engineering and the Sciences (A Survey)," *Multiple Criteria* Decision Making – Past Decade and Future Trends, ed. M. Zeleny, JAI Press, Greenwich, Connecticut, 1984.
  - Ralph E. Steuer, "Multicriteria Optimization - Theory, Computation and Application", 1985

Notation and Classification

Traditionally - single objective constrained optimization:

$$\max \mathbf{J} = f(\mathbf{x})$$
$$s.t. \quad \mathbf{x} \in S$$

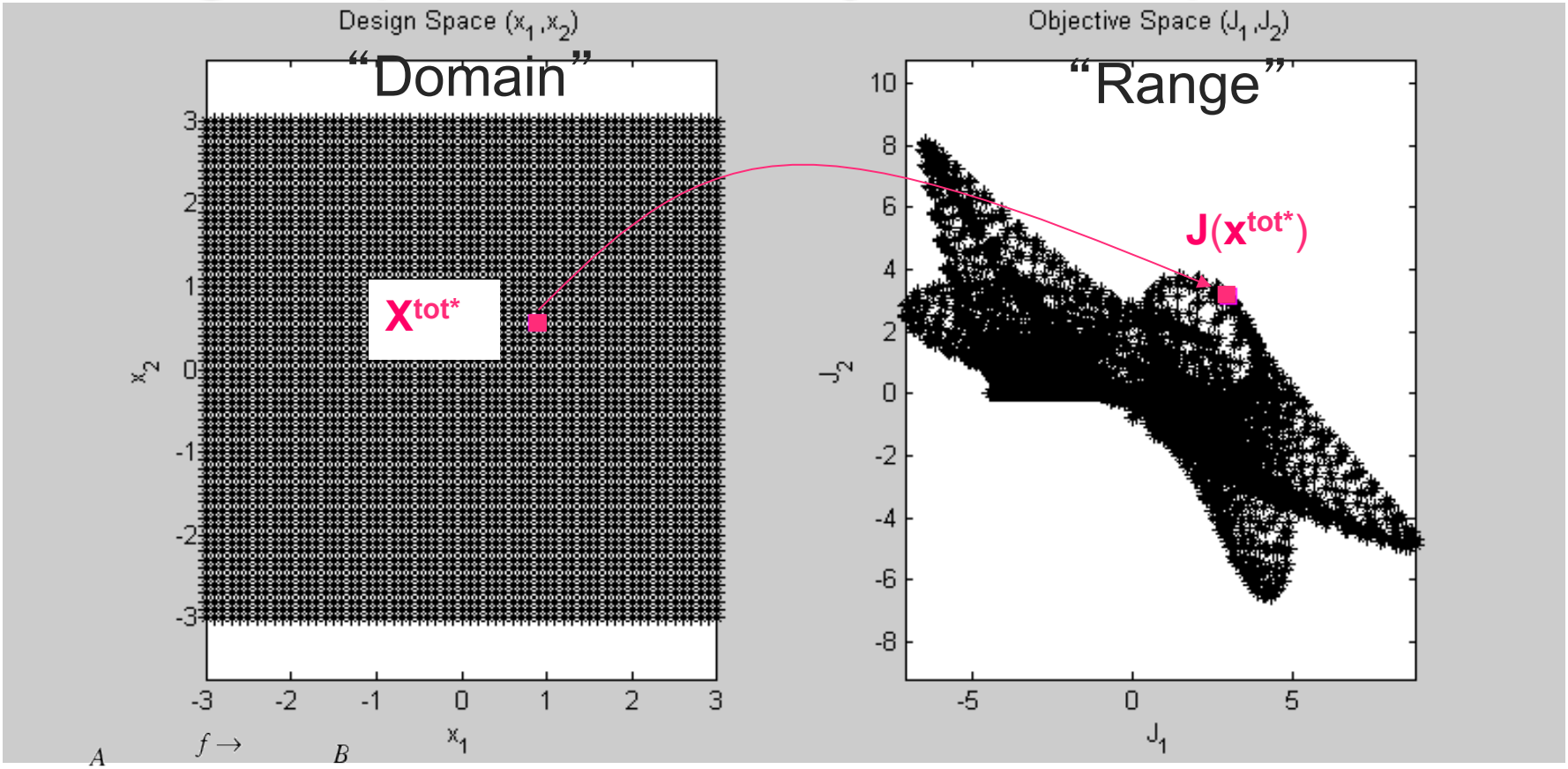
$f(\mathbf{x}) \mapsto J$ objective function

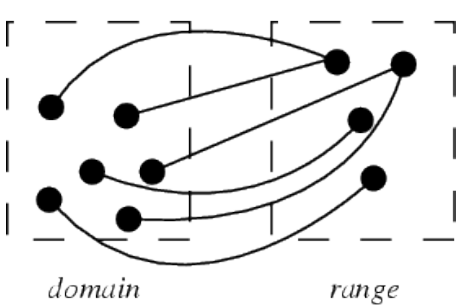
$S \mapsto$ feasible region

If *f*(**x**) linear & constraints linear & single objective = LP
If *f*(**x**) linear & constraints linear & multiple obj. = MOLP
If *f*(**x**) and/or constraints nonlinear & single obj.= NLP
If *f*(**x**) and/or constraints nonlinear & multiple obj.= MONLP

Ref: Ralph E. Steuer, "Multicriteria Optimization - Theory, Computation and Application", 1985

Design Space vs. Objective Space





A function *f* which may (but does not necessarily) associate a given member of the range of *f* with more than one member of the domain of *f*.

Formal Solution of a MOO Problem

Trivial Case:

There is a point $x^* \in S$ that simultaneously optimizes all objectives J_i, where $1 \le i \le z$

Such a point almost never exists - i.e. there is no point that will simultaneously optimizes all objectives at once

Two fundamental approaches:

Scalarization Approaches

$$\max\left\{U\left(J_1, J_2, \ldots, J_z\right)\right\}$$

$$s.t. \quad J_i = f_i(\mathbf{x}) \quad 1 \le i \le z$$

$$\mathbf{x} \in S$$

Preferences included upfront

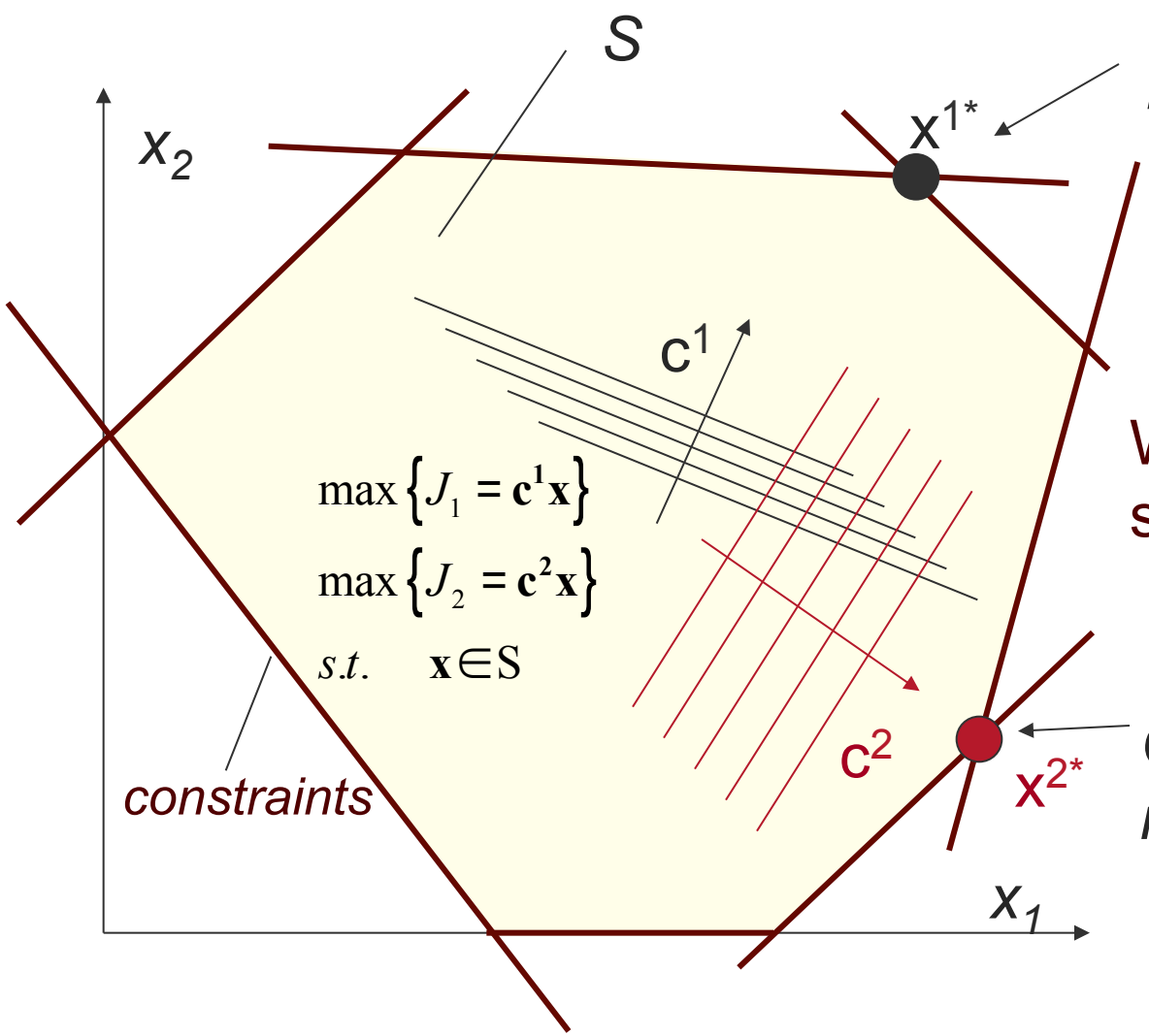
Pareto Approaches

$$J_i^1 \ge J_i^2 \quad \forall\ i$$

and $J_i^1 > J_i^2$ for at least one i

Preferences included a posteriori

SOLP versus MOLP



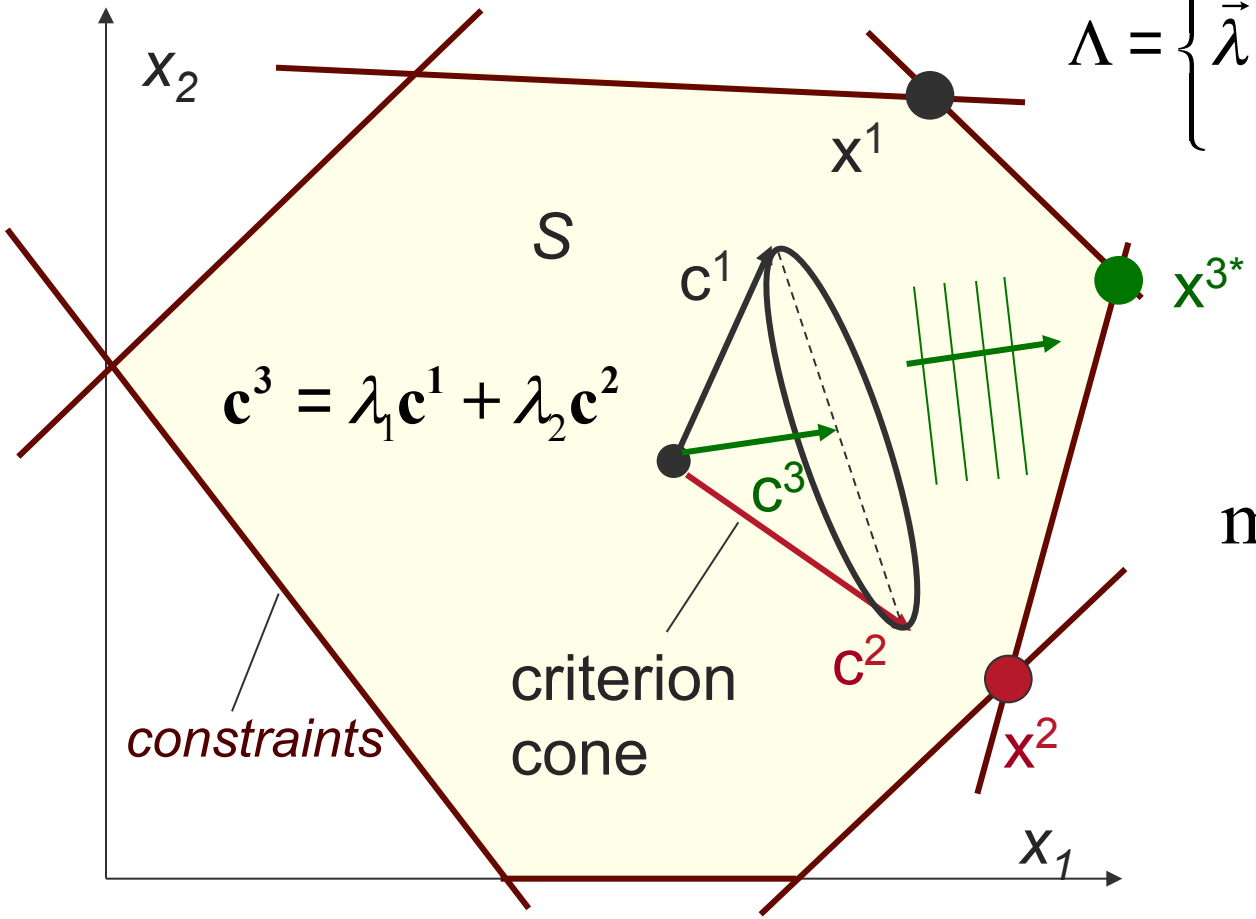
Optimal solution if only J_1 considered

What is the optimal solution of a MOLP?

Optimal solution if only J_2 considered

Weighted-Sum Approach

Each objective *i* is multiplied by a strictly positive scalar λ_i



$$\Lambda = \left\{ \vec{\lambda} \in \mathbb{R}^z \middle| \lambda_i > 0, \sum_{i=1}^{k} \lambda_i = 1 \right\}$$

Solve the composite or WSLP:

$$\max \left\{ \boldsymbol{\lambda}^T \mathbf{C} \mathbf{x} \middle| \mathbf{x} \in S \right\}$$

Strictly convex combination of objectives

Group Exercise: Weights

We are trying to build the "optimal" automobile

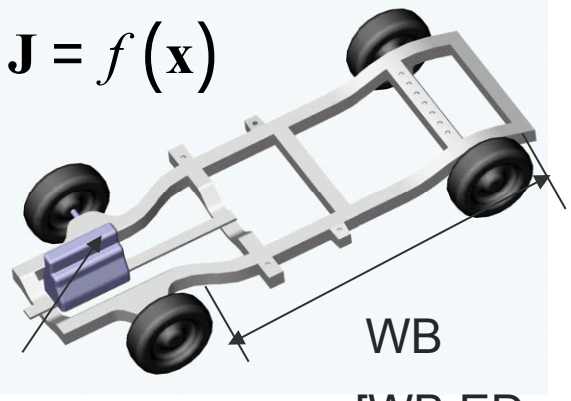
There are six consumer groups:

-G1: "25 year old single" (Cannes, France)
-G2: "family w/3 kids" (St. Louis, MO)
-G3: "electrician/entrepreneur" (Boston, MA)
-G4: "traveling salesman" (Montana, MT)
-G5: "old lady" (Rome, Italy)
-G6: "taxi driver" (Hong Kong, China)

Objective Vector:

J1: Turning Radius [m]
J2: Acceleration [0-60mph]
J3: Cargo Space [m^3]
J4: Fuel Efficiency [mpg]
J5: Styling [Rating 0-10]
J6: Range [km]
J7: Crash Rating [poor-excellent]
J8: Passenger Space [m^3]
J9: Mean Time to Failure [km]

$\mathbf{J} = f(\mathbf{x})$



$\mathbf{x}=[\text{WB ED}\ldots]^T$

Assignment: Determine λ_i , i =1…9

$$\sum_{i=1}^{9} \lambda_i = 1000$$

What are the scale factors sf_i?

- Scaling is critical in multiobjective optimization
- Scale each objective by sf_i: $\bar{J}_i = J_i / sf_i$
- Common practice is to scale by $sf_i = J_i^*$
- Alternatively, scale to initial guess $\mathbf{J}(\bar{\mathbf{x}}_o)=[1..1]^T$
- Multiobjective optimization then takes place in a non-dimensional, unit-less space
- Recover original objective function values by reverse scaling

Example:

J_1=range [sm]
J_2=fuel efficiency [mpg]

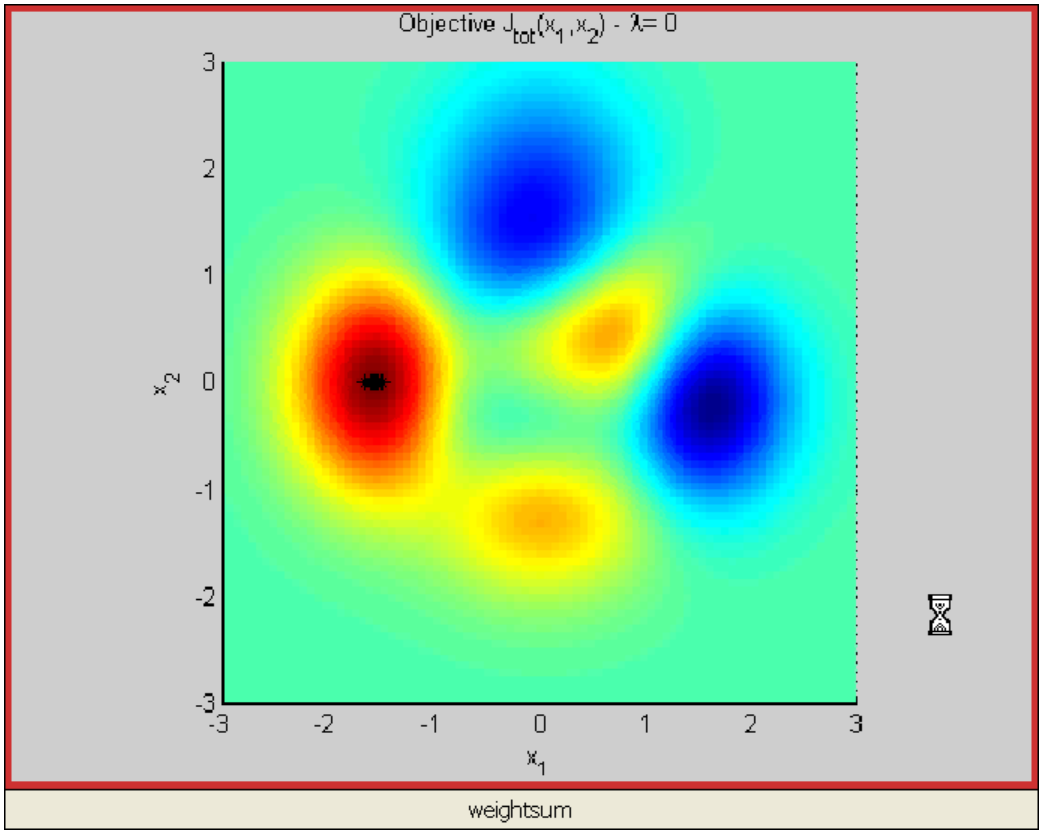
Saab 9-5

sf_1=573.5 [sm]
sf_2=36 [mpg]

Suzuki "Swift"

Weighted Sum: Double Peaks

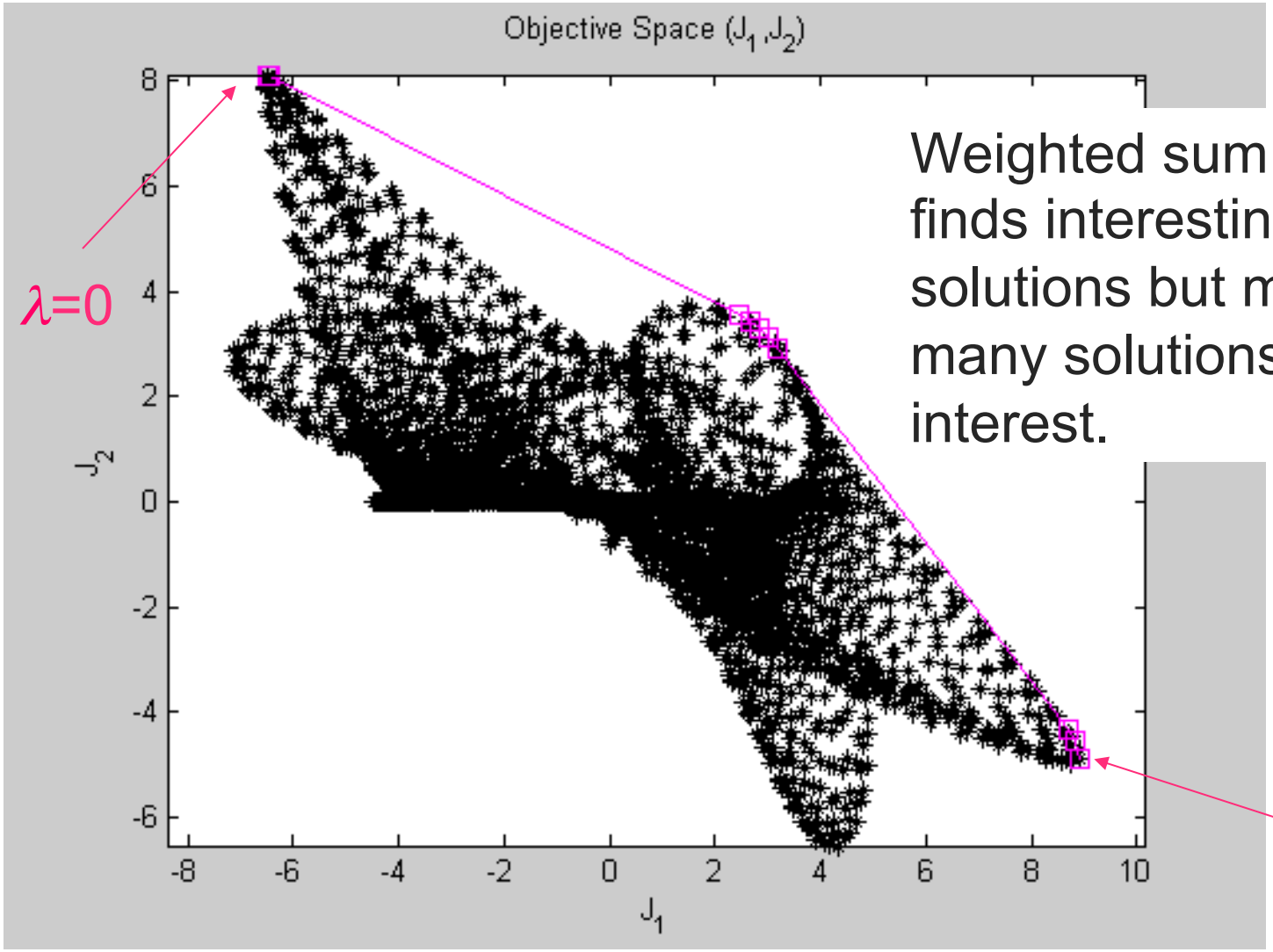
$$J_{tot} = \lambda J_1 + (1-\lambda) J_2 \quad \text{where} \quad \lambda \in [0,1]$$



$\Delta\lambda$=0.05

Demo: At each setting of λ we solve a new single objective optimization problem – the underlying function changes at each increment of λ

Weighted Sum Approach (II)



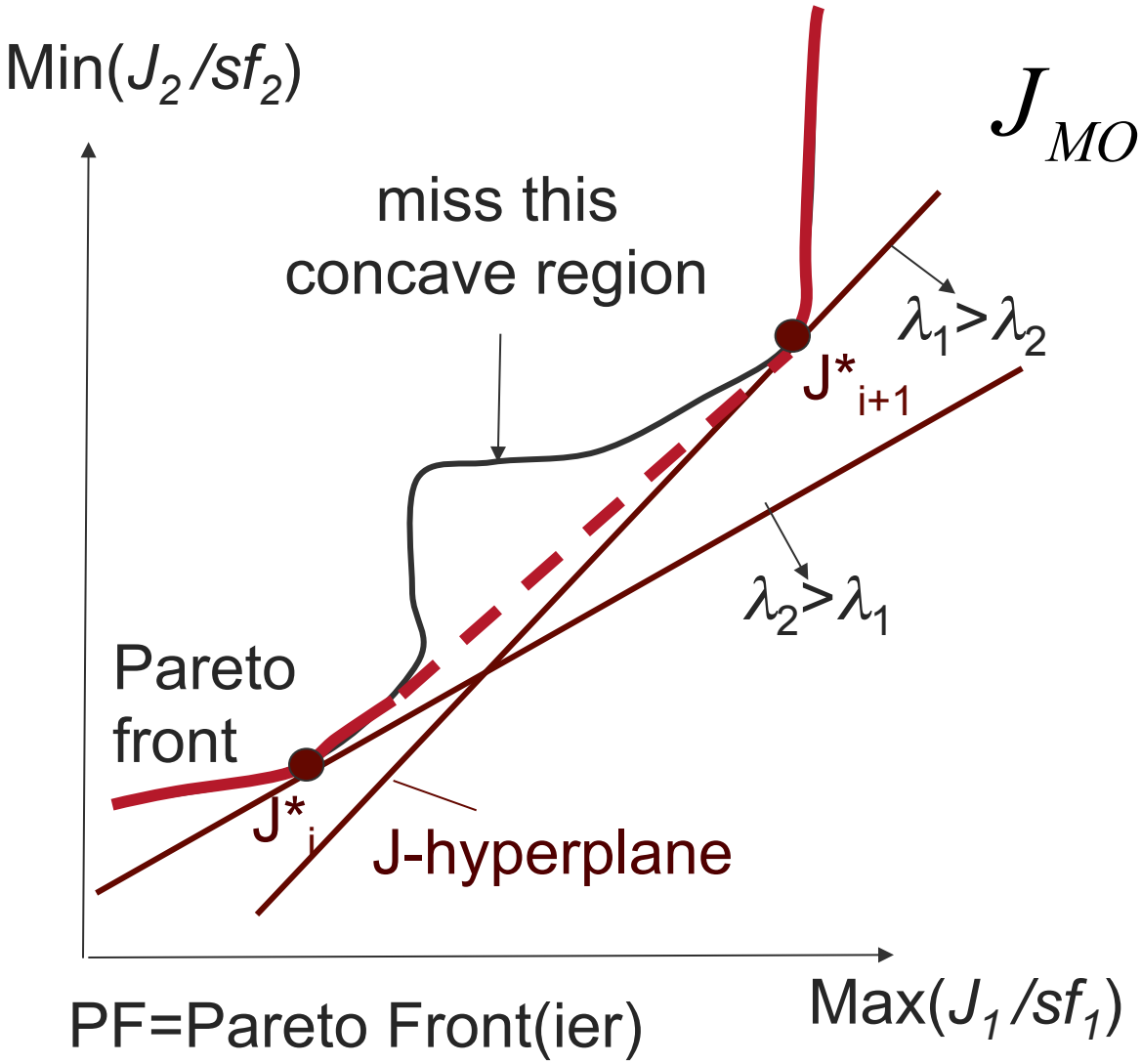
Weighted sum finds interesting, solutions but misses many solutions of interest.

Weighted Sum Approach (III)

$$J_{MO} = \sum_{i=1}^{z} \frac{\lambda_i}{sf_i} J_i$$

weight

scale factor



Properties of optimal solution

$\mathbf{x}^*$ optimal if $\mathbf{J}(\mathbf{x}^*) \geq \mathbf{J}(\mathbf{x})$ (maximization)

for $\mathbf{x}^* \in S$ and for $\mathbf{x} \neq \mathbf{x}^*$

⇒ This is why multiobjective optimization is also sometimes referred to as vector optimization

x* must be an efficient solution

$\mathbf{x} \in S$ is efficient if and only if (iff) its objective vector (criteria) J(**x**) is non-dominated

A point $\mathbf{x} \in S$ is efficient if it is not possible to move feasibly from it to increase an objective without decreasing at least one of the others

Dominance (assuming maximization)

Let $\mathbf{J^1}, \mathbf{J^2} \in \mathbb{R}^z$ be two objective (criterion) vectors.

$$J^i = \begin{bmatrix} J_1 \\ J_2 \\ \vdots \\ J_z \end{bmatrix}$$

Then **J¹** dominates **J²** (weakly) iff

$$\mathbf{J^1} \geq \mathbf{J^2} \quad \text{and} \quad \mathbf{J^1} \neq \mathbf{J^2}$$

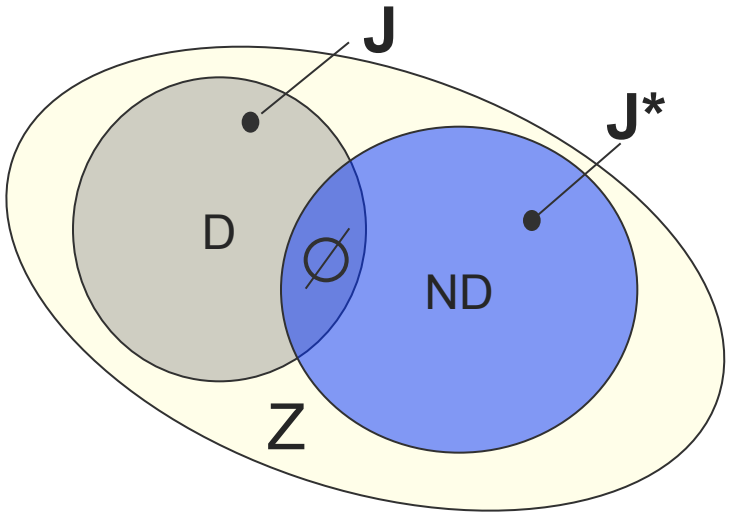
More precisely: $J_i^1 \geq J_i^2 \quad \forall\, i \quad$ and $J_i^1 > J_i^2$ for at least one i

Also **J¹** strongly dominates **J²** iff

More precisely: $\mathbf{J^1} > \mathbf{J^2}$ $\quad J_i^1 > J_i^2 \quad \forall\, i$

Set Theory

Set Theory:



$\mathbf{x} \in S$

$\mathbf{J}, \mathbf{J}^* \in Z$

A solution must be feasible

$D \cap ND = \varnothing$

A solution is either dominated or non-dominated but cannot be both at the same time

$D \subset Z, ND \subset Z$

All dominated and non-dominated solutions must be feasible

$D \cup ND = Z$

All feasible solutions are either non-dominated or dominated

$\mathbf{J}^*(\mathbf{x}^*) \in ND$

Pareto-optimal solutions are non-dominated

Dominance versus Efficiency

- Whereas the idea of dominance refers to vectors in criterion space *J*, the idea of efficiency refers to points in decision space *x*.

- Can use this criterion as a Pareto Filter if the design space has been explored (e.g. DoE).

Dominance Exercise

max{range}	[km]
min{cost}	[$/km]
max{passengers}	[-]
max{speed}	[km/h]

Multiobjective Aircraft Design

#1	#2	#3	#4	#5	#6	#7	#8
7587	6695	3788	8108	5652	6777	5812	7432
321	211	308	278	223	355	401	208
112	345	450	88	212	90	185	208
950	820	750	999	812	901	788	790

Which designs are non-dominated? (5 min)

Dominance Exercise

Algorithm for extracting non-dominated solutions:

Pairwise comparison

#1		#2
7587	>	6695
321	>	211
112	<	345
950	>	820

Score #1	Score #2
1	0
0	1
0	1
1	0
2	vs 2

Neither #1 nor #2 dominate each other

#1		#6
7587	>	6777
321	<	355
112	>	90
950	>	901

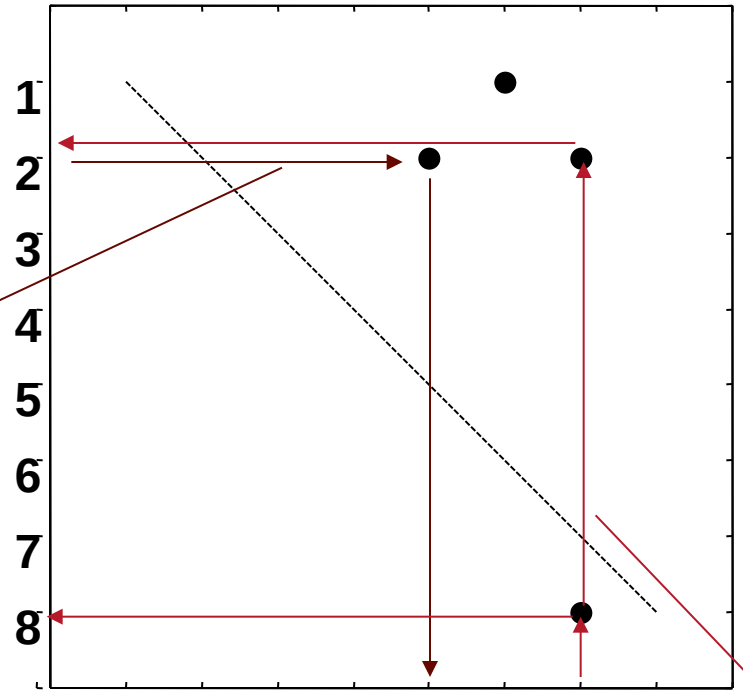
Score #1	Score #6
1	0
1	0
1	0
1	0
4	vs 0

Solution #1 dominates solution #6

In order to be dominated a solution must have a "score" of 0 in pairwise comparison

Domination Matrix

Shows which solution dominates which other solution (horizontal rows) and (vertical columns)



Row Σ

1
2
0
0
0
0
0
1

j-th row indicates how many solutions j-th solution dominates

Solution 2 dominates Solution 5

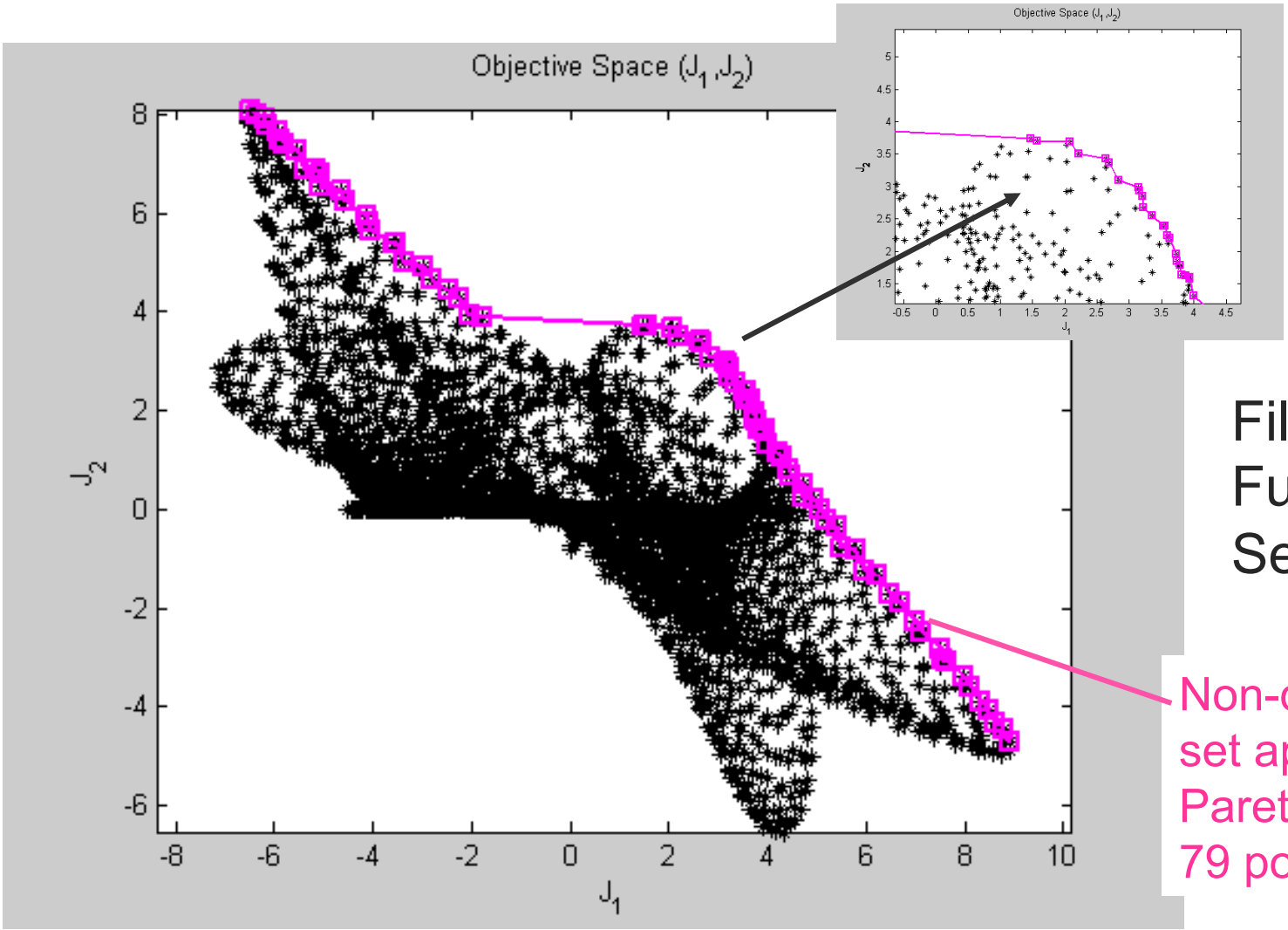
k-th column indicates by how many solutions the k-th solution is dominated

Column Σ → | 0 | 0 | 0 | 0 | 1 | 1 | 2 | 0 |

Solution 7 is dominated by Solutions 2 and 8

Non-dominated solutions have a zero in the column Σ !

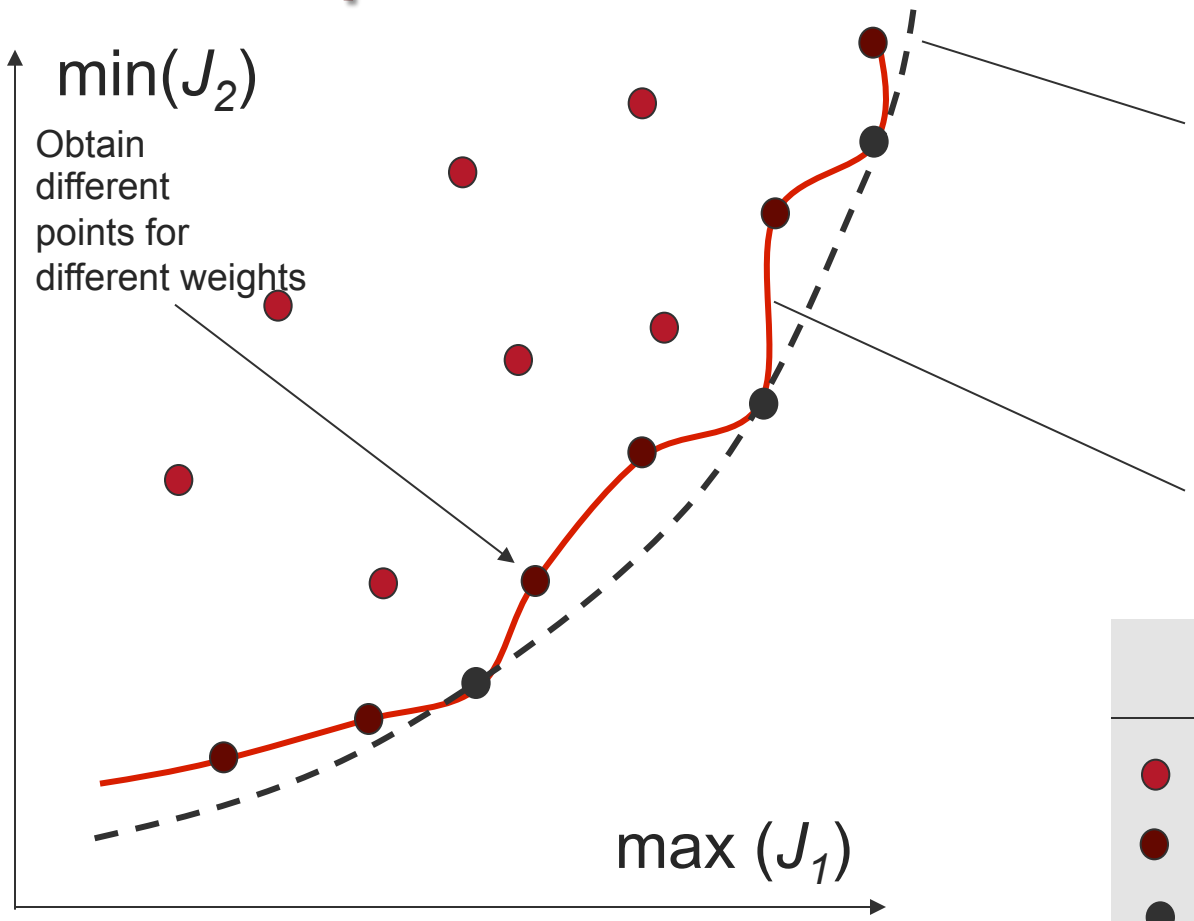
Double Peaks: Non-dominated Set



Filtered the Full Factorial Set: 3721

Non-dominated set approximates Pareto frontier: 79 points (2.1%)

Pareto-Optimal vs ND



True Pareto Front

Approximated Pareto Front

	D	ND	PO
●	√		
●		√	
●		√	√

All pareto optimal points are non-dominated
Not all non-dominated points are pareto-optimal

It's easier to show dominatedness than non-dominatedness!!!

Lecture Summary

- A multiobjective problem has more than one optimal solution
- All points on Pareto Front are non-dominated
- Methods:
  - Weighted Sum Approach (Caution: Scaling!)
  - Pareto-Filter Approach
  - Methods for direct Pareto Frontier calculation next time:
    - AWS (Adaptive Weighted Sum)
    - NBI (Normal Boundary Intersection)

The key difference between multiobjective optimization methods can be found in how and when designer preferences are brought into the process.

…. More in next lecture

Remember…

Pareto Optimal means …..

"Take from Peter to pay Paul"

Lecture 15: Multidisciplinary System Analysis and Design Optimization (MSADO)

Post-Optimality Analysis

April 8, 2015

Prof. Douglas Allaire

Lecture Outline

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

Standard Problem Definition

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

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

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

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

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

Karush-Kuhn-Tucker Conditions

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

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

$$\lambda_j \geq 0$$

λ_{m_1+k} unrestricted in sign

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

Karush-Kuhn-Tucker Conditions: Interpretation

Condition 1: the optimal design satisfies the constraints

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

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

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

Optimization for Engineering Problems

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

Optimization for Engineering Problems

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

Termination Criteria: Gradient-Based

Gradient-based algorithm is terminated when ...

an acceptable solution is found

OR

algorithm terminates unsuccessfully

Need to decide:

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

Termination Criteria: Gradient-Based

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

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

Termination Criteria: Gradient-Based

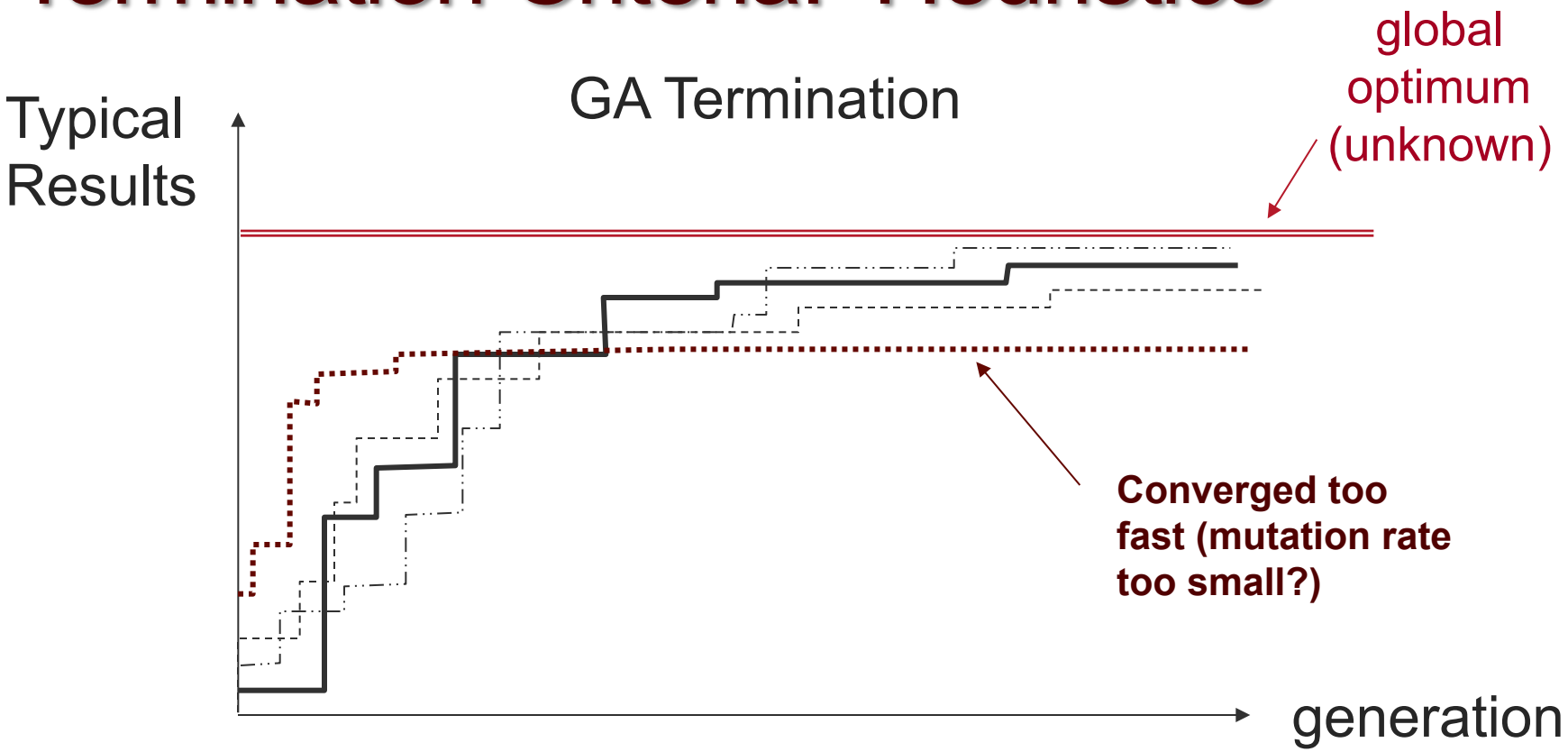
Has the sequence $\{ \mathbf{x}^k \}$ converged?

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

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

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

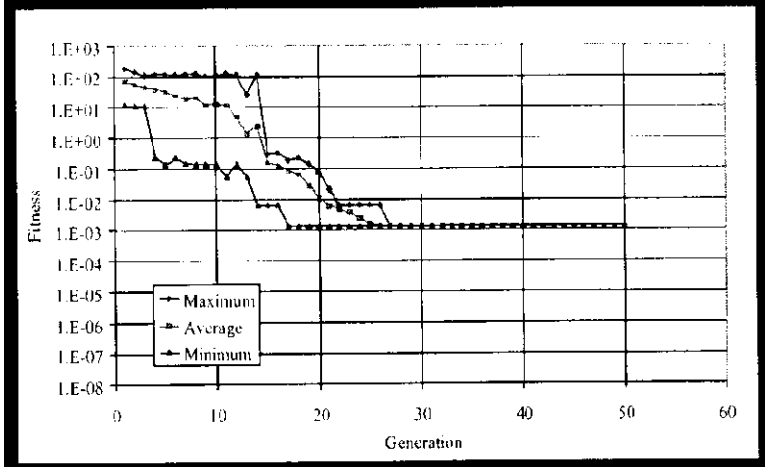
Termination Criteria: Heuristics



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

Termination Criteria: Heuristics

What Does a Solution Look Like in Time?



$$F(x, y) = x^2 + y^2 + 25\left(\sin^2 x + \sin^2 y\right)$$

INITIAL POPULATION	FINAL POPULATION
11001110101001001111 0	01111111111011111111 1
10010100111111101011 01	01111111111011111111 1
01010100111110011110 11	01111111111011111111 1
10111110110010010000 0	01111111111011111111 1
11010111001111101011 01	01111111111011111111 1
01111111110001011011 00	01111111111011111111 1
01001011000011110010 10	01111111111011111111 1
11010101000001000010 01	01111111111011111111 1
01000000010110010110 10	01111111111011111111 1
11011010011000111010 00	01111111111011111111 1
00011101100001001010 11	01111111111011111111 1
11001001110101110011 11	01111111111011111111 1
00110101101101110000 10	01111111111011111111 1
00011001000011111001 10	01111111111011111111 1
11110100101101110010 01	01111111111011111111 1
10000000011010110111 00	01111111111011111111 1
01000011000001111011 11	01111111111011111111 1
10111011011011000100 10	01111111111011111111 1
00001101111010111100 01	01111111111011111111 1
10001011100110001000 10	01111111111011111111 1
00011111110011000101 10	01111111111011111111 1
10010010011011110011 11	01111111111011111111 1
01100011100010100100 00	01111111111011111111 1
11010011001101010111 00	01111111111011111111 1
10011000111011000001 10	01111111111011111111 1
00111110110001010110 00	01111111111011111111 1
11010011100110100110 00	01111111111011111111 1
00010001011010001011 10	01111111111011111111 1
11000010110011011000 10	01111111111011111111 1
01000111111010110000 00	01111111111011111111 1

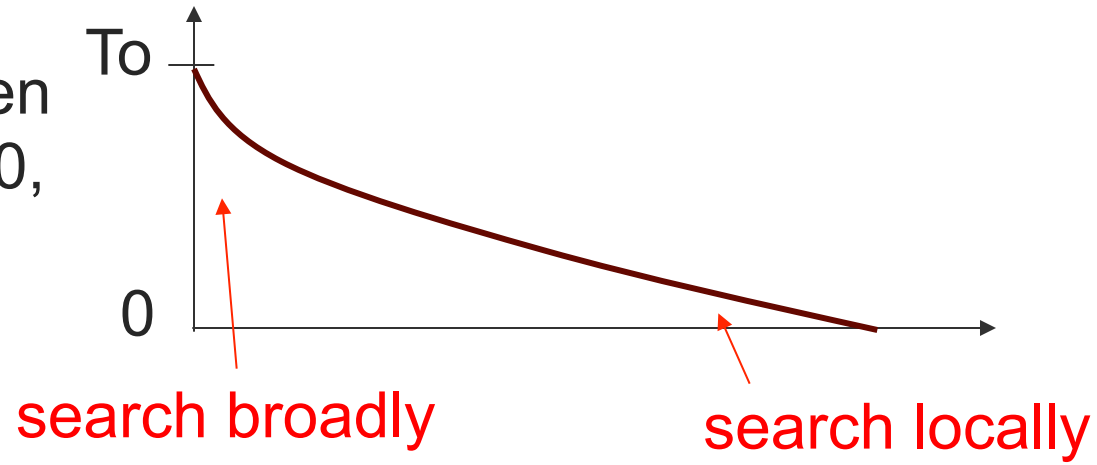
JACOBS Sverdrup

Dominant schema

Termination Criteria: Heuristics

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

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



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

Post-Optimality Analysis

- Already talked about **sensitivity** analysis (Lecture 8)
  - How does optimal solution change as a parameter is varied?
  - How does optimal solution change as a design variable value is varied?
  - How does optimal solution change as constraints are varied?
- Also would like to understand key drivers in optimal design
- We also saw in Lecture 8 that the values of the Lagrange multipliers at the optimal solution give information on how modifying the constraints affects the solution.

Objective Function Behavior

Consider the quadratic function:

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

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

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

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

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

Objective Function Behavior

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

Consider the neighborhood of the optimal solution:

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

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

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

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

Objective Function Behavior

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

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

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

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

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

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

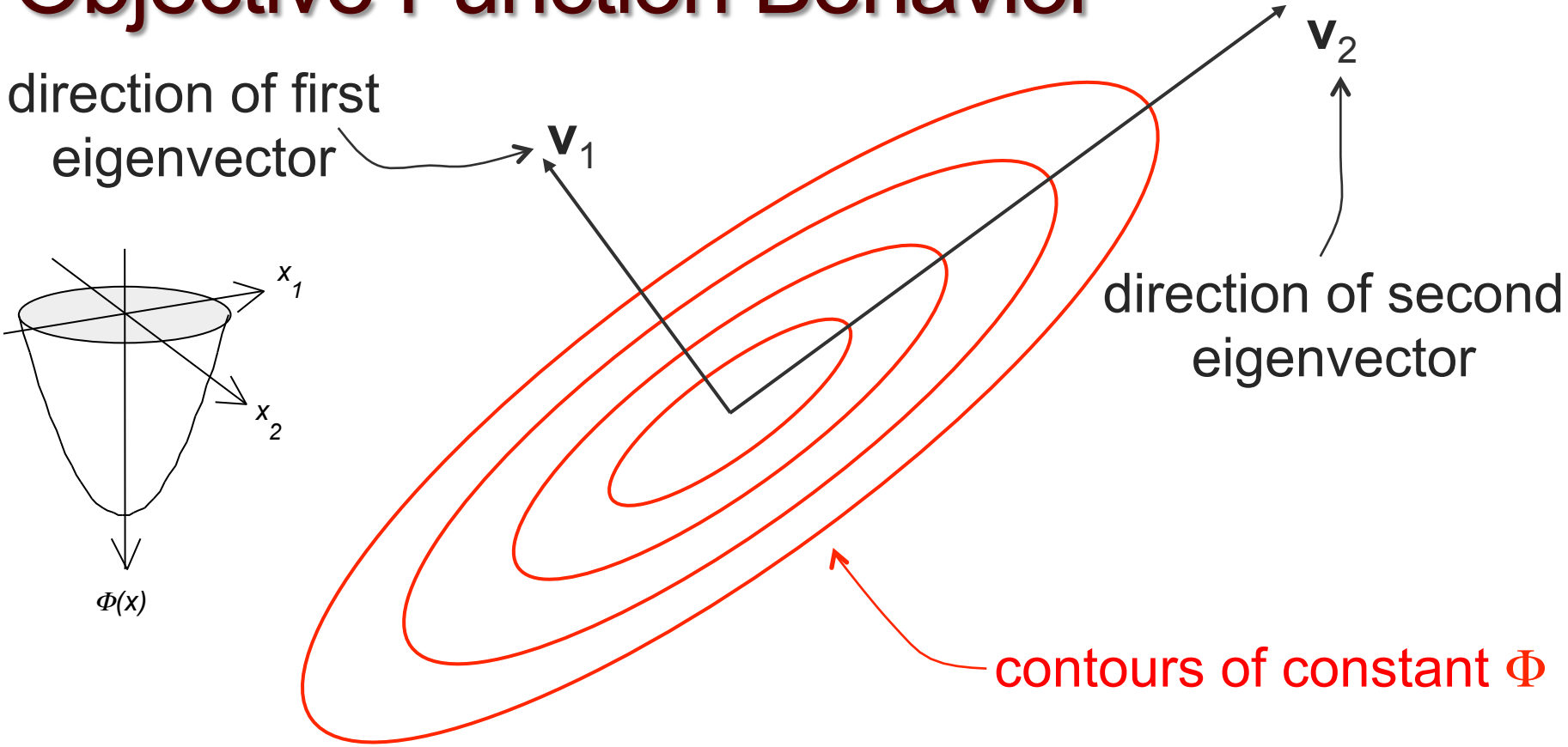
As we move away from $\mathbf{x}^*$ along the direction $\mathbf{v}_j$, the change in the objective depends on the sign of λ_j.

Objective Function Behavior

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

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

Objective Function Behavior



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

Hessian Condition Number

- The condition number of the Hessian is given by

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

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

Scaling

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

$\Rightarrow$ numerical accuracy, numerical conditioning

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

Scaling

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

scale objective

$$\equiv$$

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

scale constraint

$$\equiv$$

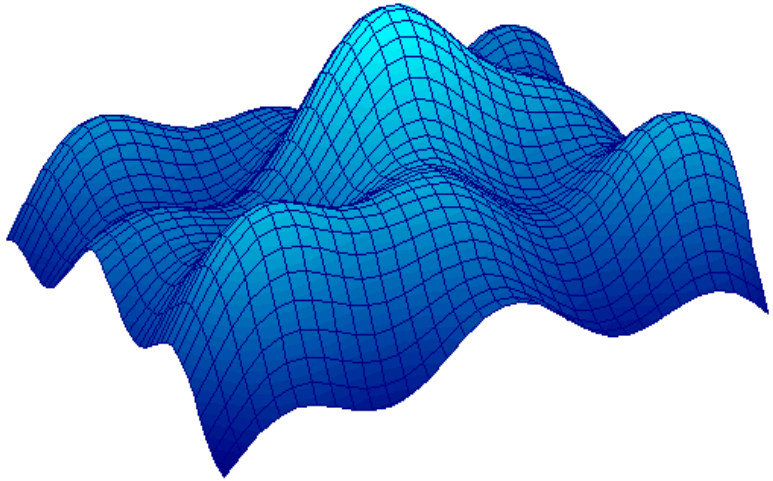
$$\begin{aligned}\min\ & x_1^2 + 3x_2 + x_3 \\ \text{s.t.}\ & 50x_1 + 20x_2^3 \geq 10 \\ & 6x_1^2 - 3x_3 \geq 2 \\ & x_i \geq 0\end{aligned}$$

scale design variable

$$\equiv$$

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

Matlab Demo: Scaling



Design Variable Scaling

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

  wing span ~ 10^1 m

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

Design Variable Scaling

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

Design Variable Scaling

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

Example

$$\min \; 3x_1^2 + 100x_2^2$$

- Propose a scaling $\mathbf{x} = \mathbf{L}\mathbf{y}$
- What do you hope your scaling achieves?

Objective Function Scaling

- In theory, we can multiply $J(\mathbf{x})$ by any constant or add a constant term, and not affect the solution
- In practice, it is generally desirable to have $J \sim O(1)$ in the region of interest
- Algorithms can have difficulties if $J(\mathbf{x})$ is very small everywhere, since convergence is usually tested using some small quantity
- Inclusion of a constant term can also cause difficulties, since the error associated with the sum may reflect the size of the constant rather than the size of $J(\mathbf{x})$
  e.g. $\min x_1^2+x_2^2$ vs. $\min x_1^2+x_2^2+1000$

Constraint Scaling

A well scaled set of constraints has two properties:

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

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

Scaling

Two reasons to scale:

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

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

Lecture Summary

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

Lecture 16: Multidisciplinary System Analysis and Design Optimization (MSADO)

Approximation Methods

April 13, 2015

Prof. Douglas Allaire

Lecture Outline

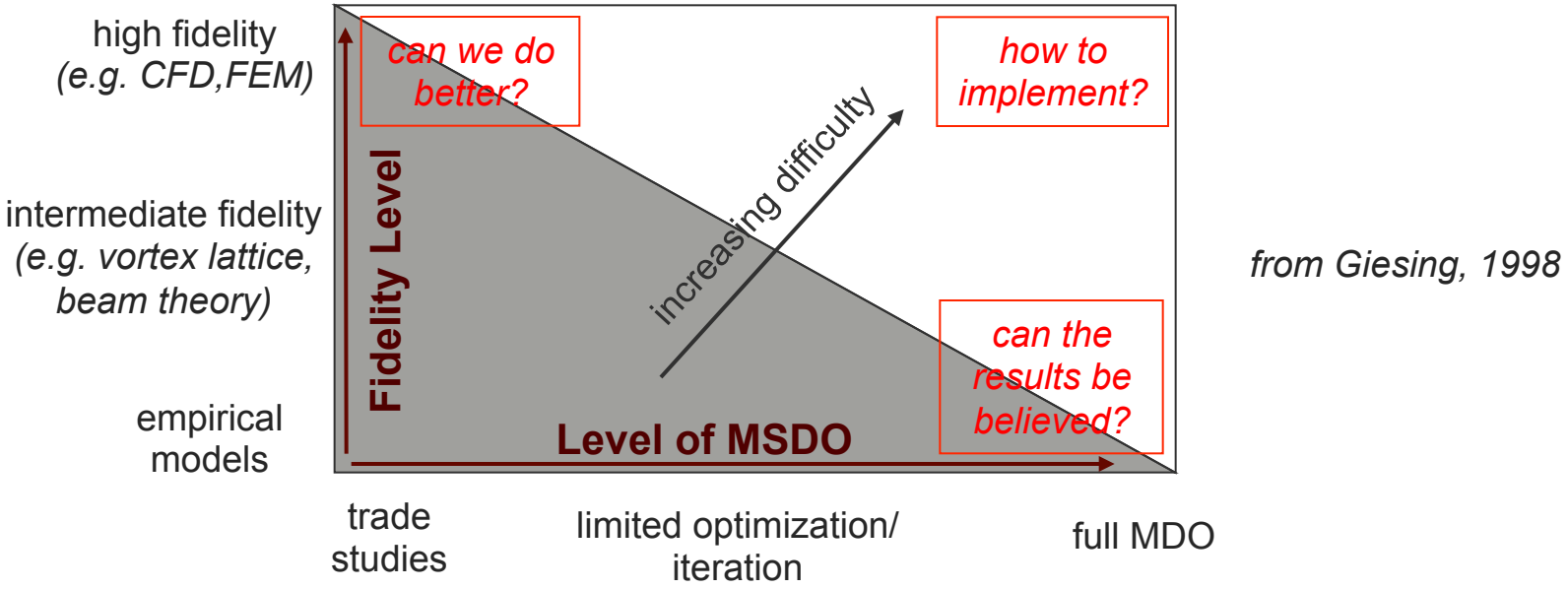
- Surrogate modeling overview
- Data-fit surrogates
  - Response surface modeling
  - Kriging
- Hierarchical Models
- Projection-based reduced-order models

Approximation Methods

- Replace expensive analysis models with an approximation or "surrogate"
- Surrogate is much less computationally expensive to evaluate than the high-fidelity analysis models
- Surrogate model classifications: data-fit, reduced-order models, hierarchical models *(Eldred & Dunlavy, 2006)*
- Surrogates widely used beyond just optimization:
  - Uncertainty quantification (e.g., Monte Carlo simulation methods)
  - Real-time control applications
  - Visualization

Why Approximation Methods

We have seen throughout the course the trade-off between **computational cost** and **fidelity.**



from Giesing, 1998

Approximation methods provide a way to get high-fidelity model information throughout the optimization without the computational expense.

Data Fit Methods

- Sample the simulation at some number of design points
  - Use DOE methods, e.g. Latin hypercube, to select the points
- Fit a surrogate model using the sampled information (typically regression or interpolation)
- Surrogate may be global (e.g., quadratic response surface) or local (e.g., Kriging interpolation)
- Surrogate may be updated adaptively by adding sample points based on surrogate performance (e.g., EGO)

Polynomial Response Surface Method

- Surrogate model is a local or global polynomial model
- Can be of any order
  - Most often quadratic; higher order requires many samples
- Advantages: Simple to implement, visualize, and understand, easy to find the optimum of the response surface
- Disadvantages: May be too simple, doesn't capture multimodal functions well

Global Polynomial Response Surface

- Fit objective function with a polynomial
- e.g., quadratic approximation to a function of n design variables x_1, x_2, …, x_n

$$J(\mathbf{x}) = c_0 + c_1 x_1 + c_2 x_2 + \ldots + c_n x_n + c_{n+1} x_1^2 + c_{n+2} x_1 x_2 + \ldots + c_{p-1} x_n^2$$

- Coefficients determined using a least squares fit to available data
- Update model by including a new function evaluation then doing least squares fit to compute the new coefficients

Global Polynomial Response Surface

Estimation problem: $\mathbf{J} \approx \mathbf{X}\,\mathbf{c}$

$$\mathbf{J} = \begin{bmatrix} J^1 & J^2 & \cdots & J^M \end{bmatrix}^T$$

J=vector containing *M* sampled responses

$$\mathbf{c} = \begin{bmatrix} c_0 \\ c_1 \\ \\ c_{p-1} \end{bmatrix}$$

c=vector containing *p* coefficients

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^1 & x_2^1 & \cdots & x_1^1 x_1^1 & \cdots \\ 1 & x_1^2 & & & & \\ \vdots & \vdots & & & & \\ & & & & & \\ 1 & x_1^M & x_2^M & & x_1^M x_1^M & \end{bmatrix}$$

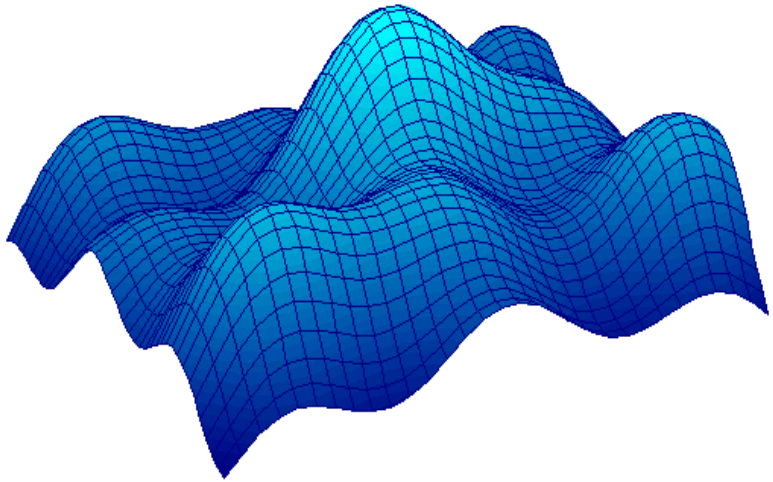
x_1^2 is the value of x_1 for the 2nd sample

X is a $M \times p$ matrix
Each row corresponds to one data sample; each column corresponds to an unknown coefficient

Least squares solution: $\mathbf{c} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{J}$

Polynomial Response Surface Method

Matlab demo: Peaks function



Kriging / Gaussian Process Models

- Adopted from the geostatistics literature
- Based on Gaussian process models
- Assumes that the output function values are correlated in design space, i.e., closer points are more highly correlated
- Can have multiple extrema
- Interpolating method
  - Exact at sample points
- Gives estimate of mean squared error
  - Can use to give error estimates
  - Can use to choose new sample points

Kriging / Gaussian Process Models

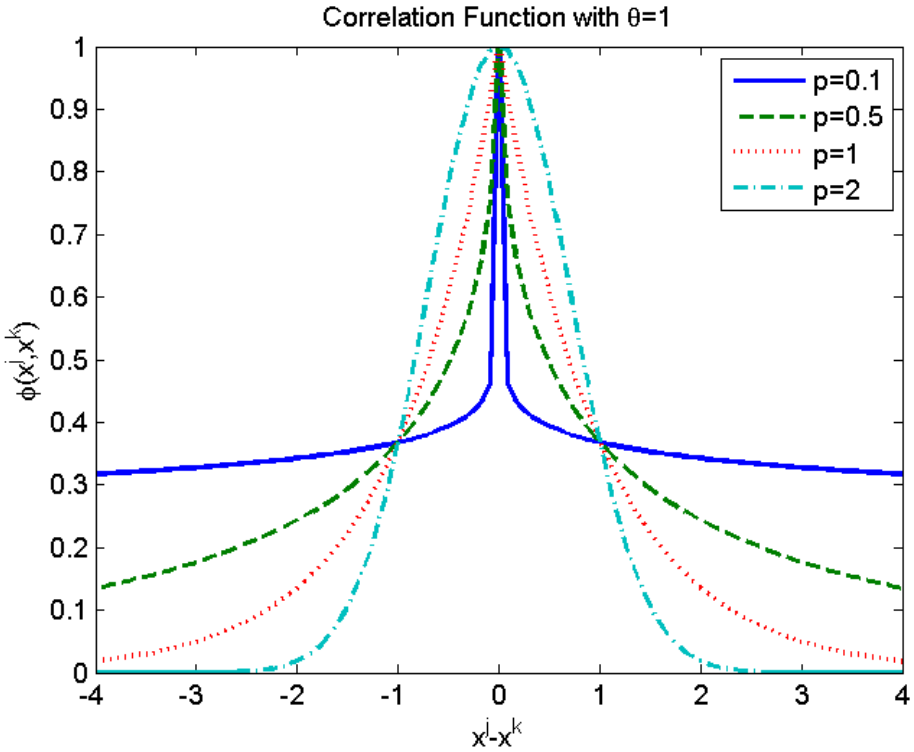
- Given a set of samples of our high-fidelity function, *f*, we want to make a prediction at a new design point $\mathbf{x}$
- Consider two design points $\mathbf{x}^j$ and $\mathbf{x}^k$
- Expect values to be close if the distance between them is small (! smooth function)
- Define the Gaussian correlation function:

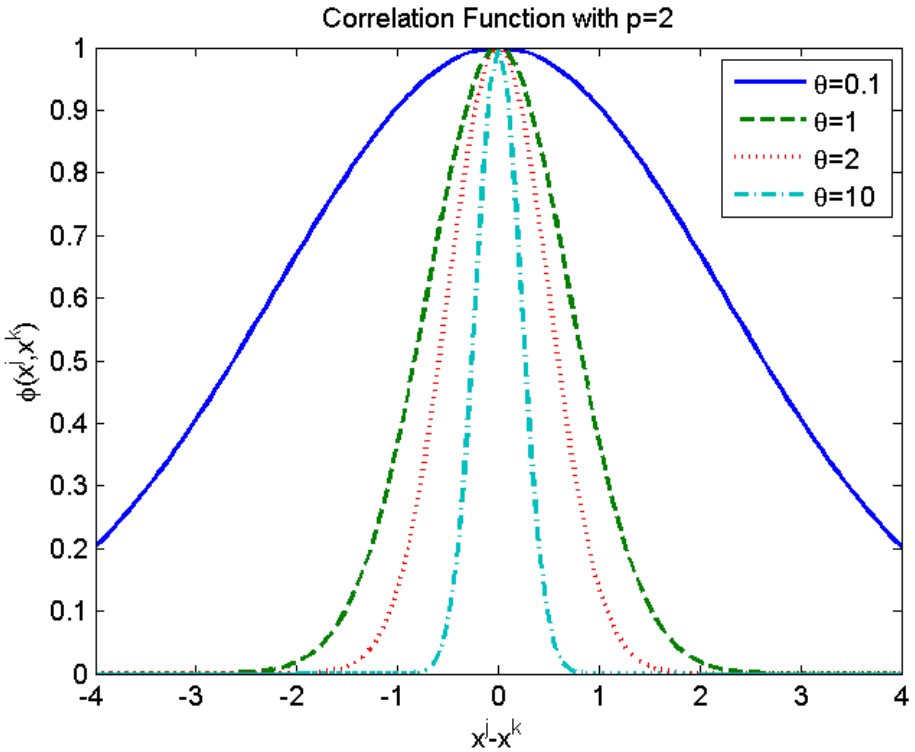
$$\phi(\mathbf{x}^j, \mathbf{x}^k) = \exp\left(-\sum_{i=1}^{n} \theta_i \left| x_i^j - x_i^k \right|^{p_i}\right)$$

ith component of $\mathbf{x}^j$

Gaussian Correlation Functions

$$\phi(\mathbf{x}^j, \mathbf{x}^k) = \exp\left(-\sum_{i=1}^{n} \theta_i \left| x_i^j - x_i^k \right|^{p_i}\right)$$





Each p_i and θ_i is chosen to best fit the data (or possibly user-specified values)

Kriging Model

- General form of a Kriging model

$$\hat{f}(\mathbf{x}) = \mu(\mathbf{x}) + Z(\mathbf{x})$$

- μ(**x**) is a global surface, typically constructed using a polynomial regression model
- Common choices are μ=constant (e.g., equal to the mean of the function samples over the design space) or a linear regression surface
- Z(**x**) is an interpolatory term that describes the local behavior of the model, constructed using the correlation function and a set of interpolation points

Kriging Model Construction

- For a set of M interpolation points $\mathbf{X}^1$, $\mathbf{X}^2$, …, $\mathbf{X}^M$, we have $\bar{\mathbf{f}} = [f^1 \; f^2 \; \ldots \; f^M]^T$

$$\bar{\mu} = [\mu(\mathbf{x}^1) \; \mu(\mathbf{x}^2) \; \ldots \; \mu(\mathbf{x}^M)]^T$$

- Define

$$\mathbf{r}(\mathbf{x}) = \left[\phi(\mathbf{x}, \mathbf{x}^1) \; \phi(\mathbf{x}, \mathbf{x}^2) \; \cdots \phi(\mathbf{x}, \mathbf{x}^M)\right]^T$$

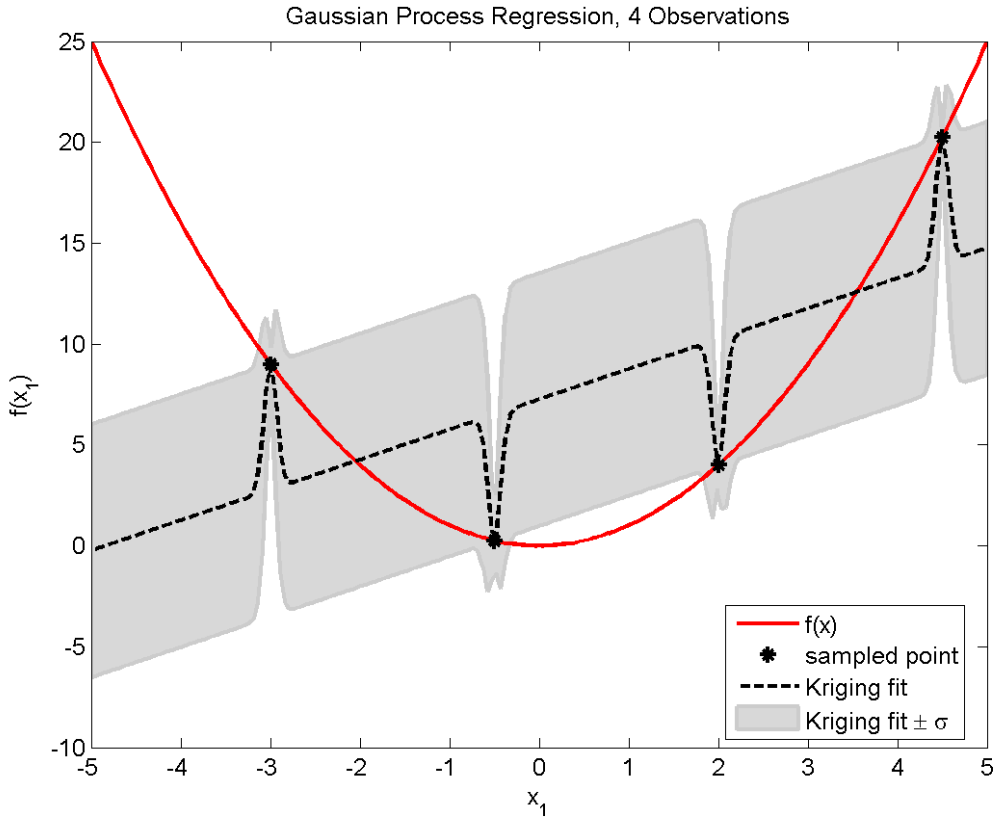
- Then Z($\mathbf{X}$) is given by

$$Z(\mathbf{x}) = \mathbf{r}^T(\mathbf{x}) R^{-1} (\bar{\mathbf{f}} - \bar{\mu})$$

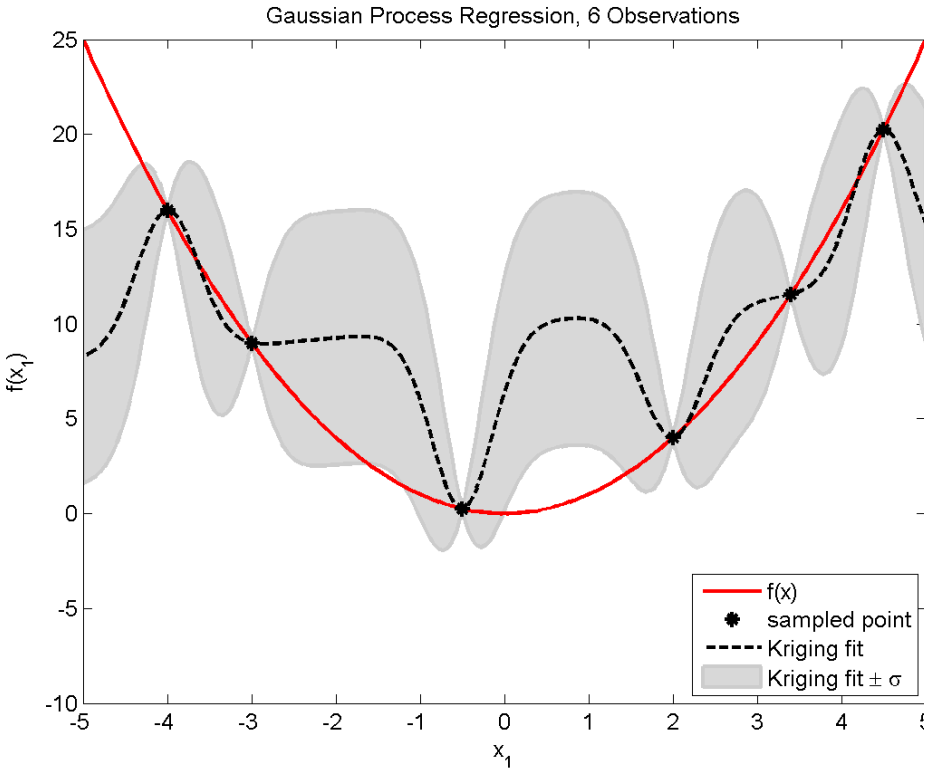
where R is the correlation matrix whose ijth entry is given by $\phi(\mathbf{x}^i, \mathbf{x}^j)$

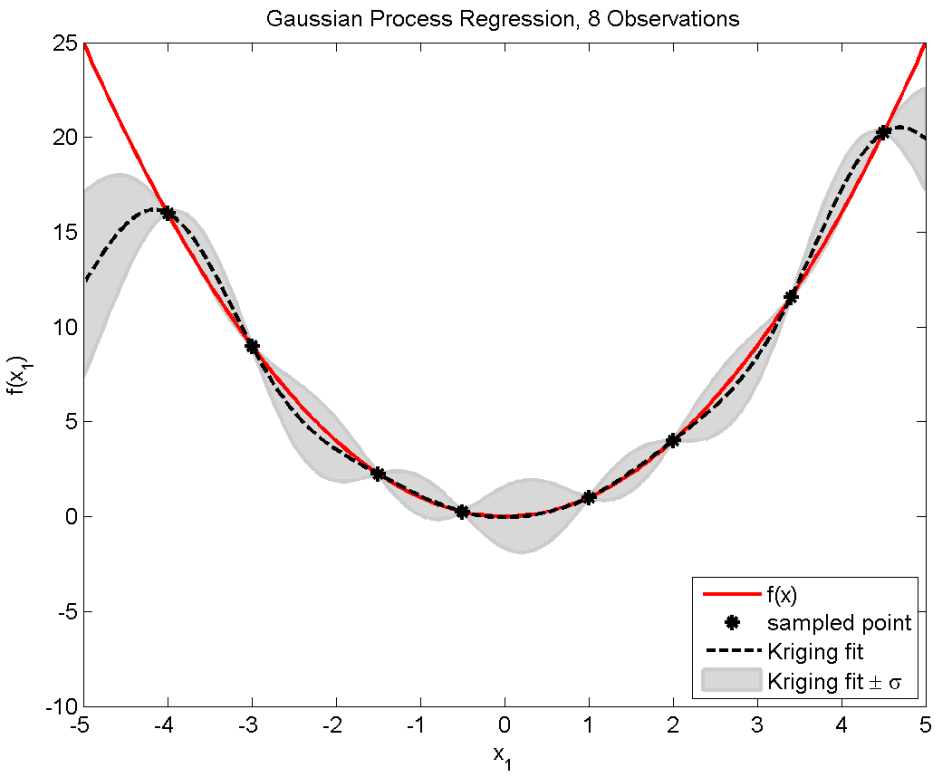
Kriging Example

- Kriging model of $f(x) = x^2$
- $\mu(x)$ is a linear regression model: $\mu(x) = \beta_0 + \beta_1 x$
- Gaussian correlation function with $p=2$, determine θ using a maximum likelihood estimate



Kriging Example





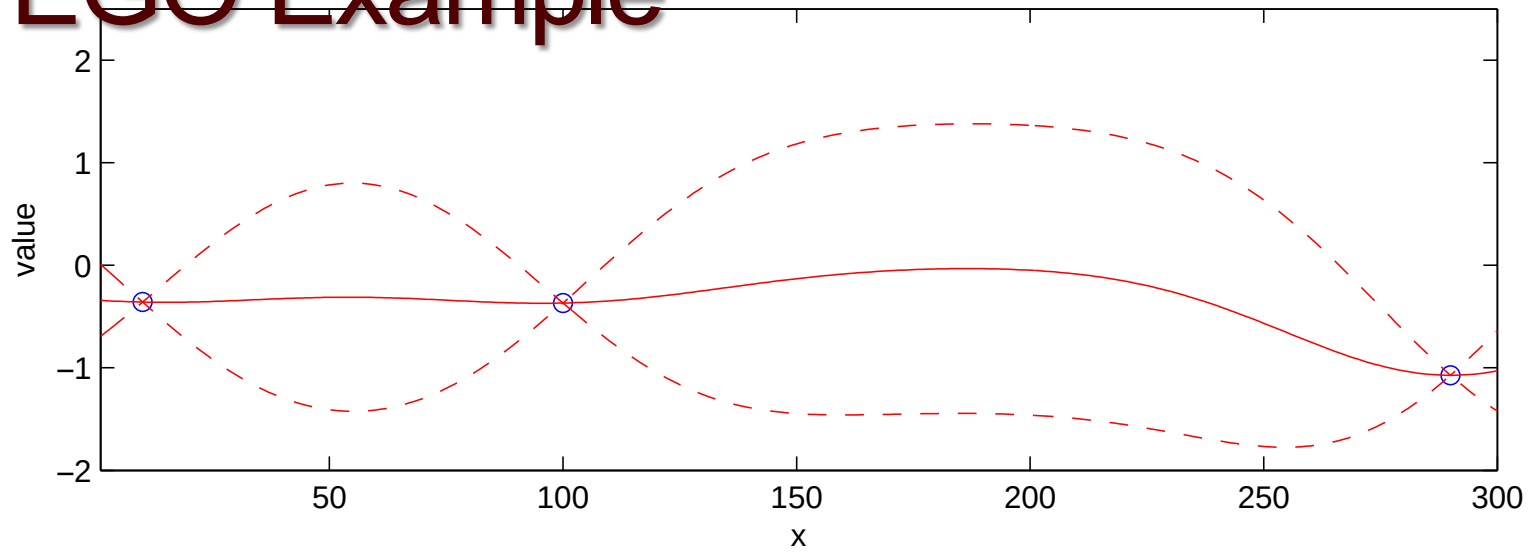
Kriging Extensions and Resources

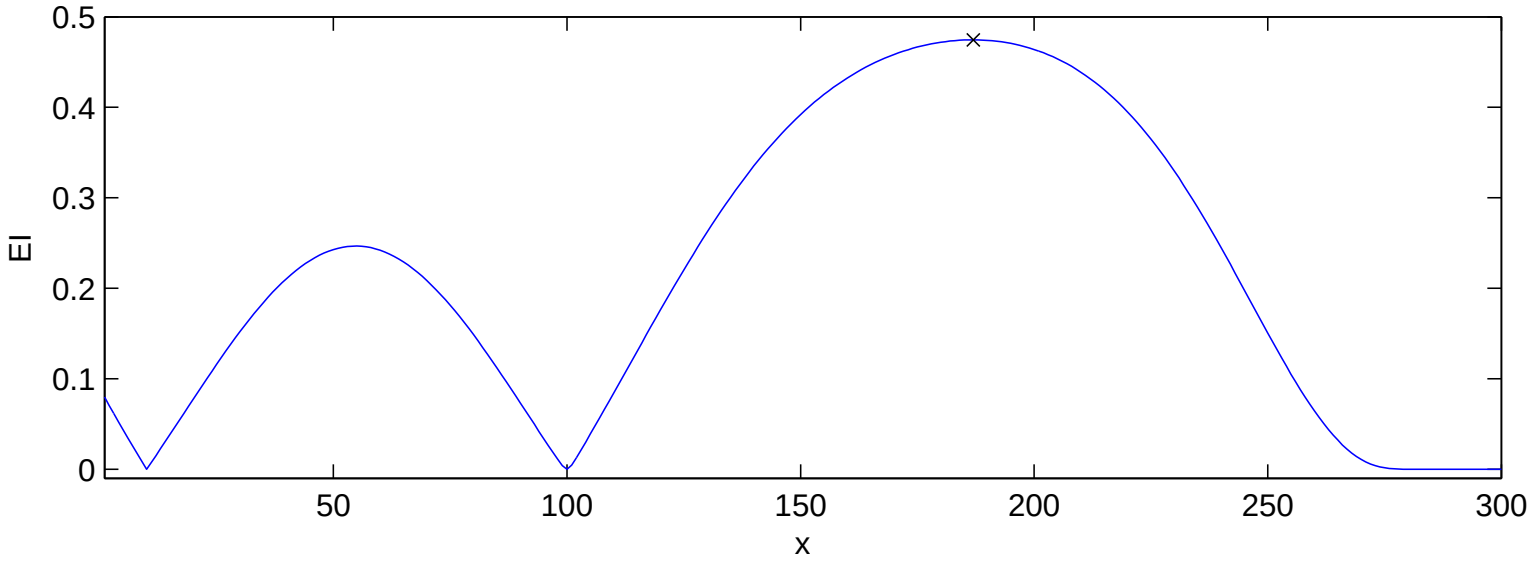
- Soft Kriging allows upper and lower bounds, prior CDFs
- Efficient Global Optimization (EGO)
  - Uses Kriging to find “expected improvement”
  - Samples the point with the largest expected improvement and adds it to the sample set
- Comprehensive description of Kriging-based approximation models in Sacks et al. (1989) and Giunta and Watson (1998)
- Another useful resource is the Matlab Kriging toolbox DACE (Lophaven et al., 2002)

EGO Example

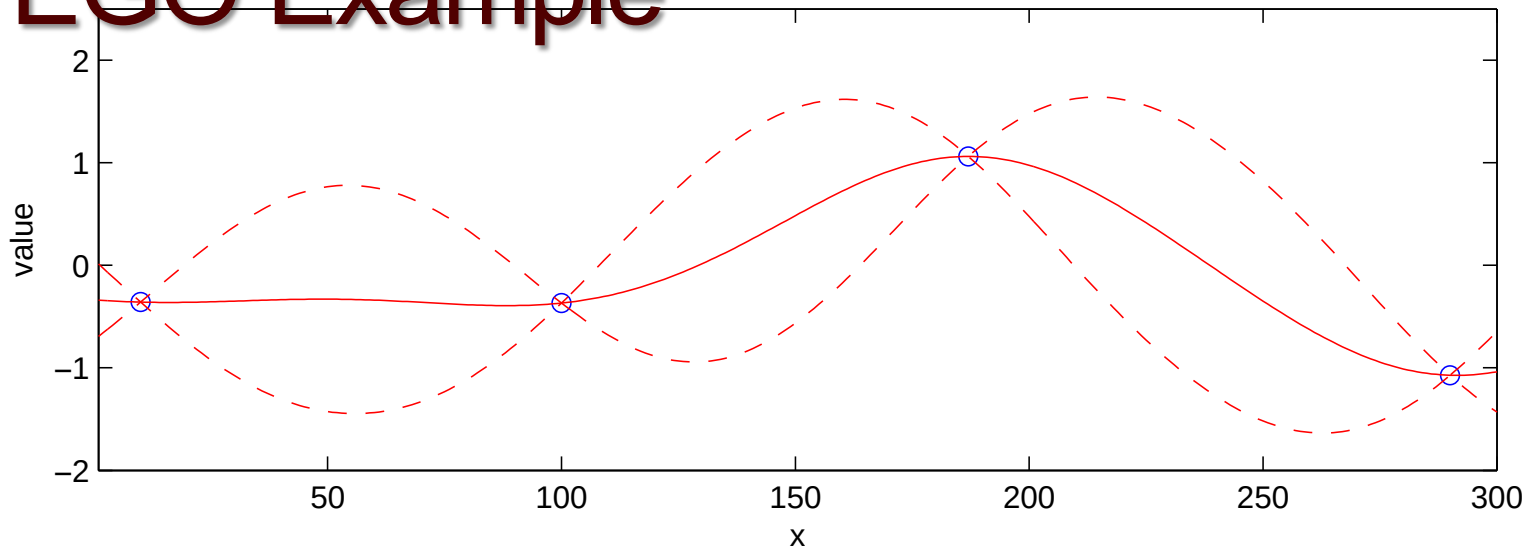
- Expected Improvement is a Bayesian optimization method from the literature [Jones, Schonlau, Welch 1998] for optimizing expensive functions.
  - Input is a design point x, a low-dimensional vector.
  - Output is the performance of that design point.
- Its goal is to find a design with maximal performance.
- It uses Gaussian process regression [Krige 1951] to **predict**.
- It uses value of information [Howard 1966] to **decide**.

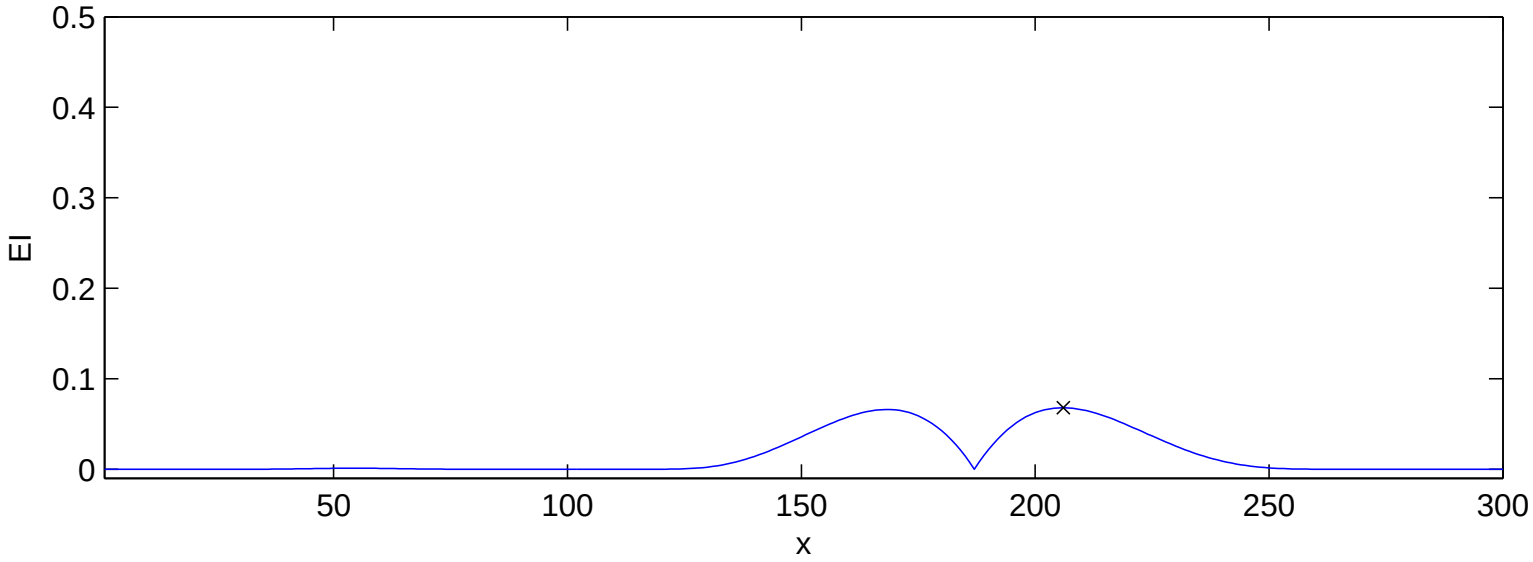
EGO Example


value
x
2
1
0
−1
−2
50
100
150
200
250
300

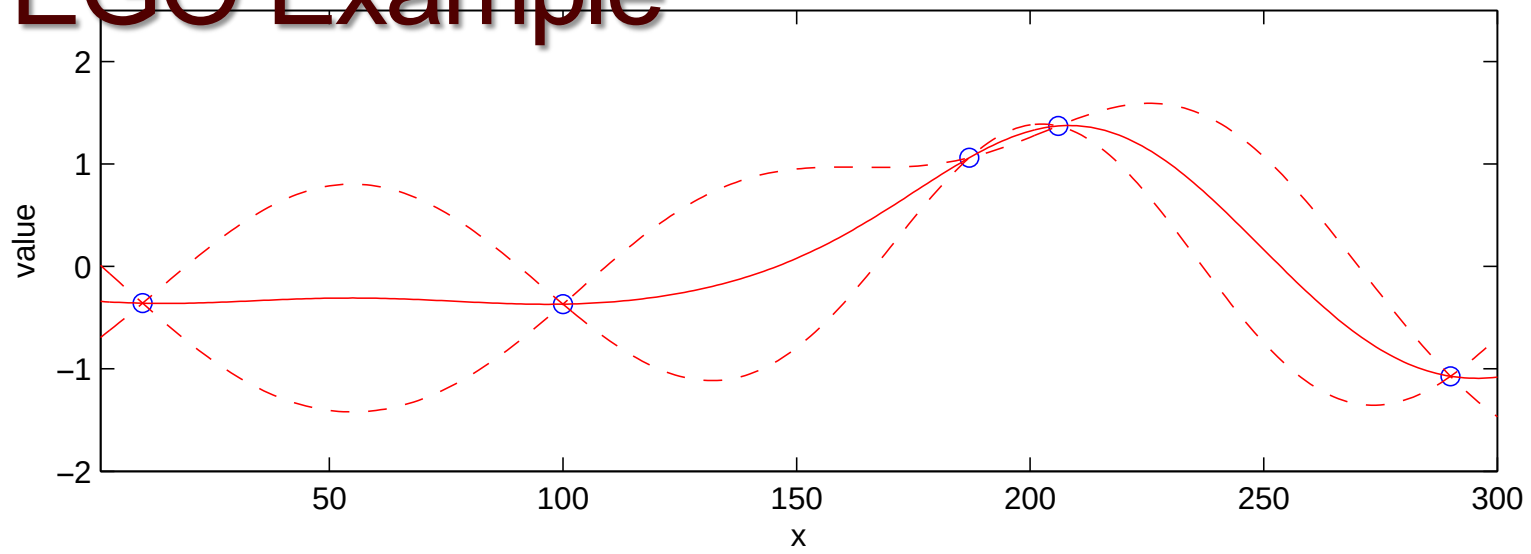

EI
x
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300

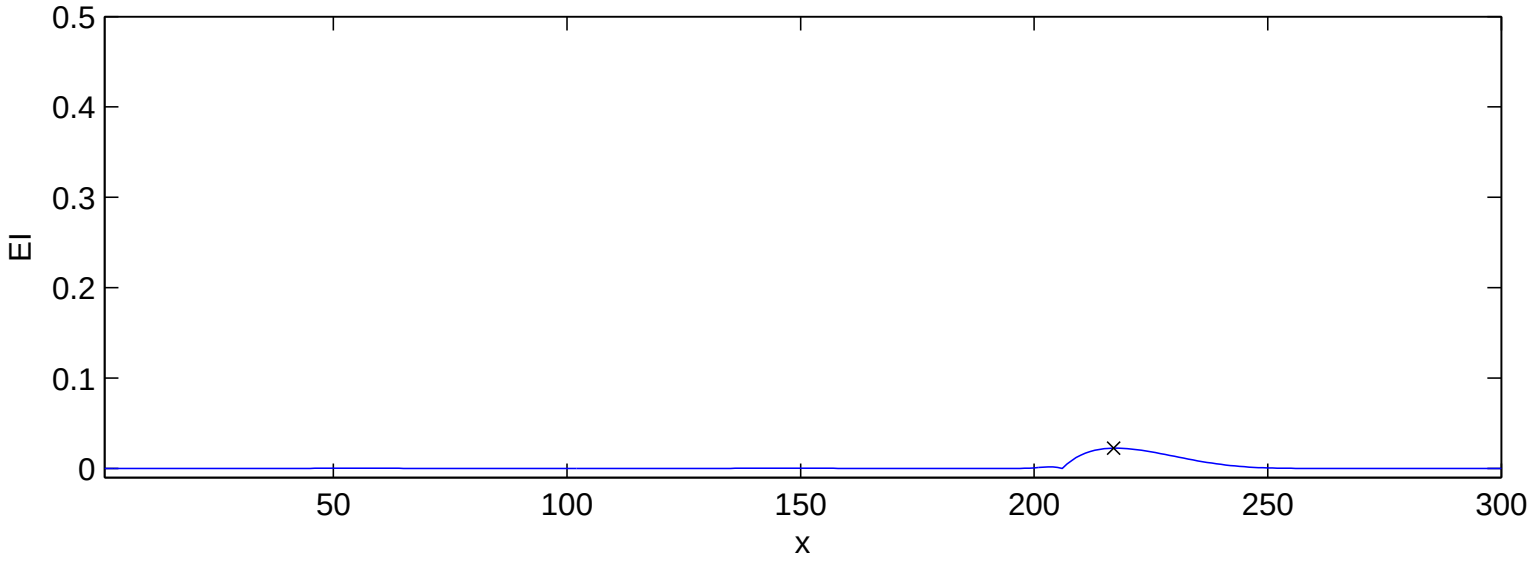
EGO Example


value
x
2
1
0
−1
−2
50
100
150
200
250
300

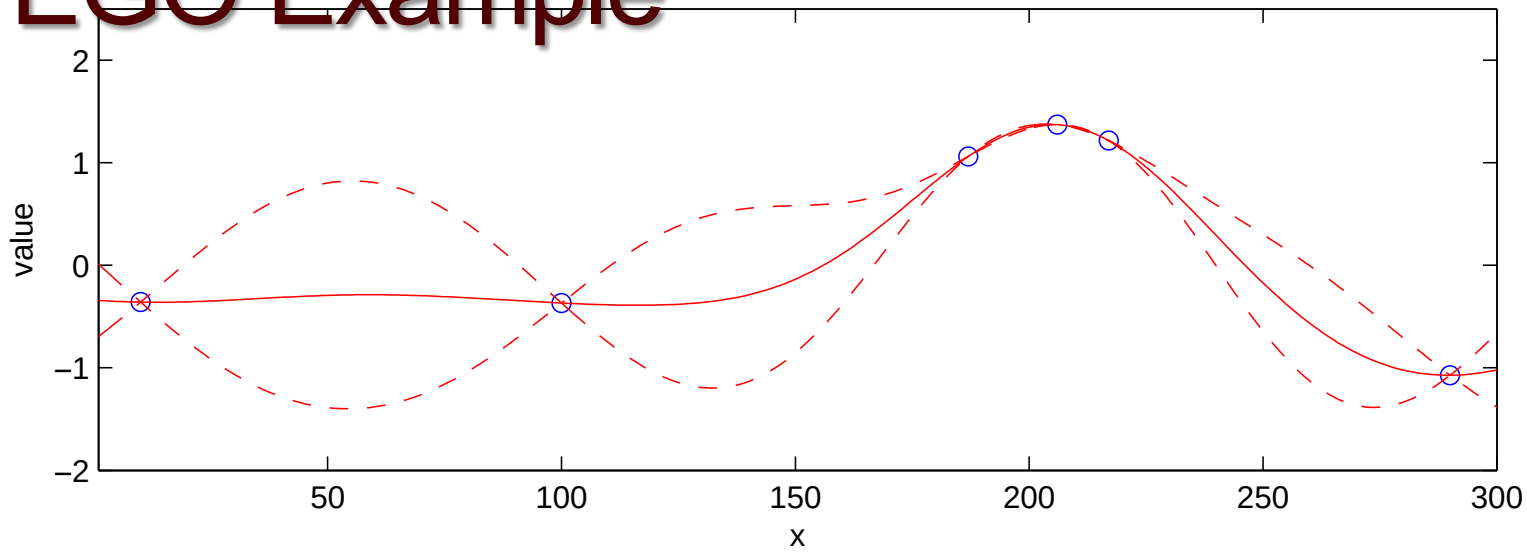

EI
x
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300

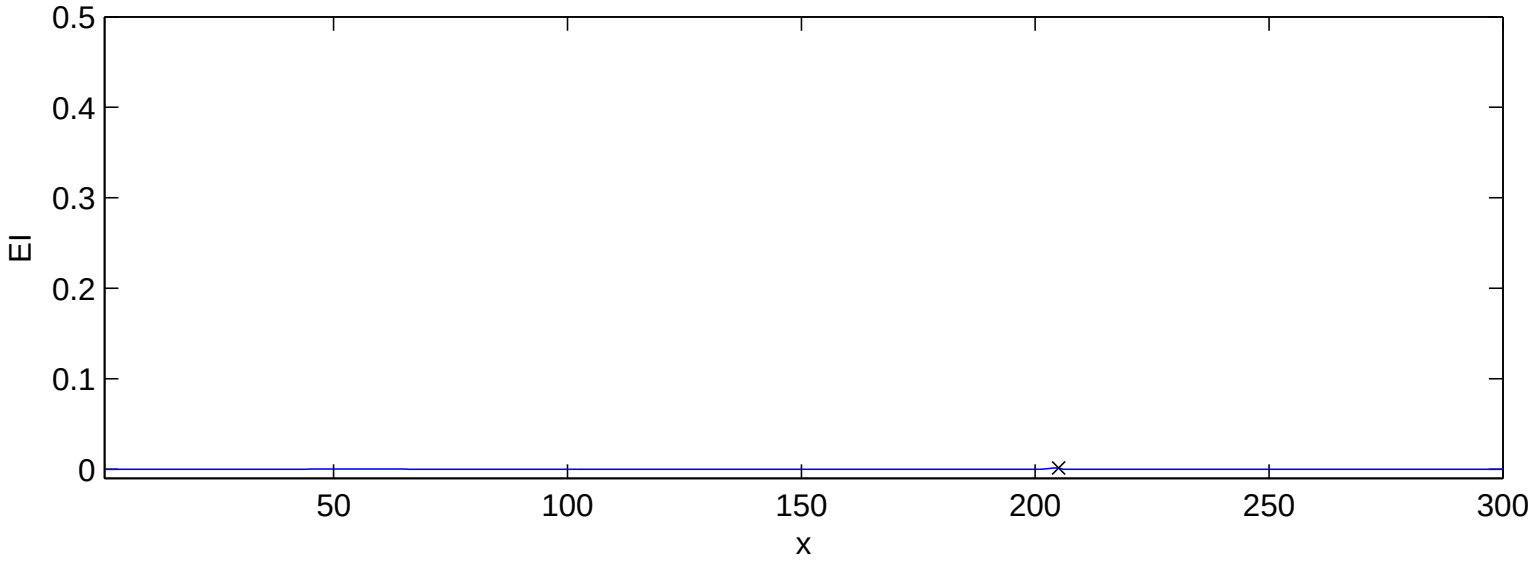
EGO Example


value
x
2
1
0
−1
−2
50
100
150
200
250
300

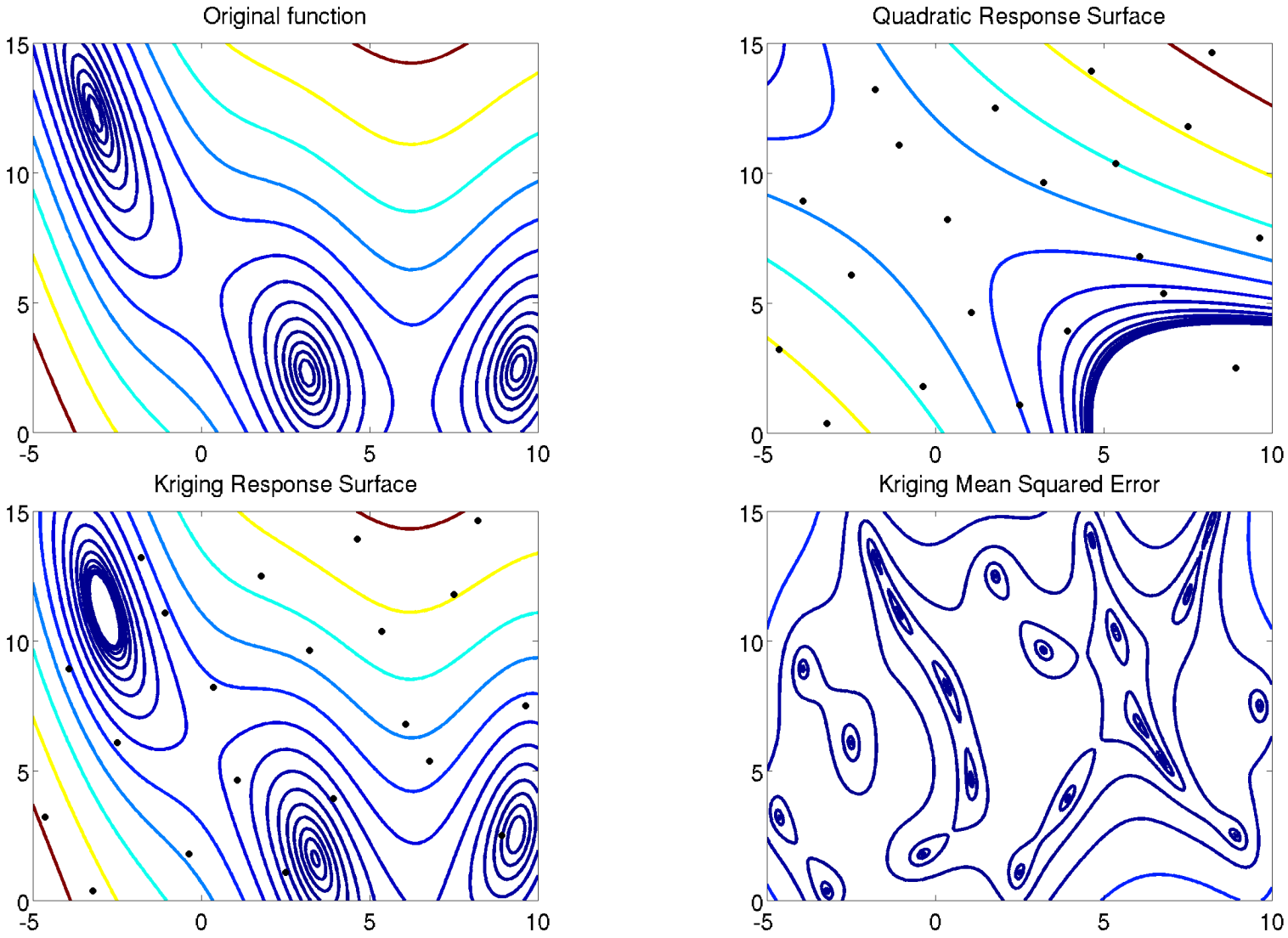

EI
x
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300

EGO Example


value
x
2
1
0
−1
−2
50
100
150
200
250
300

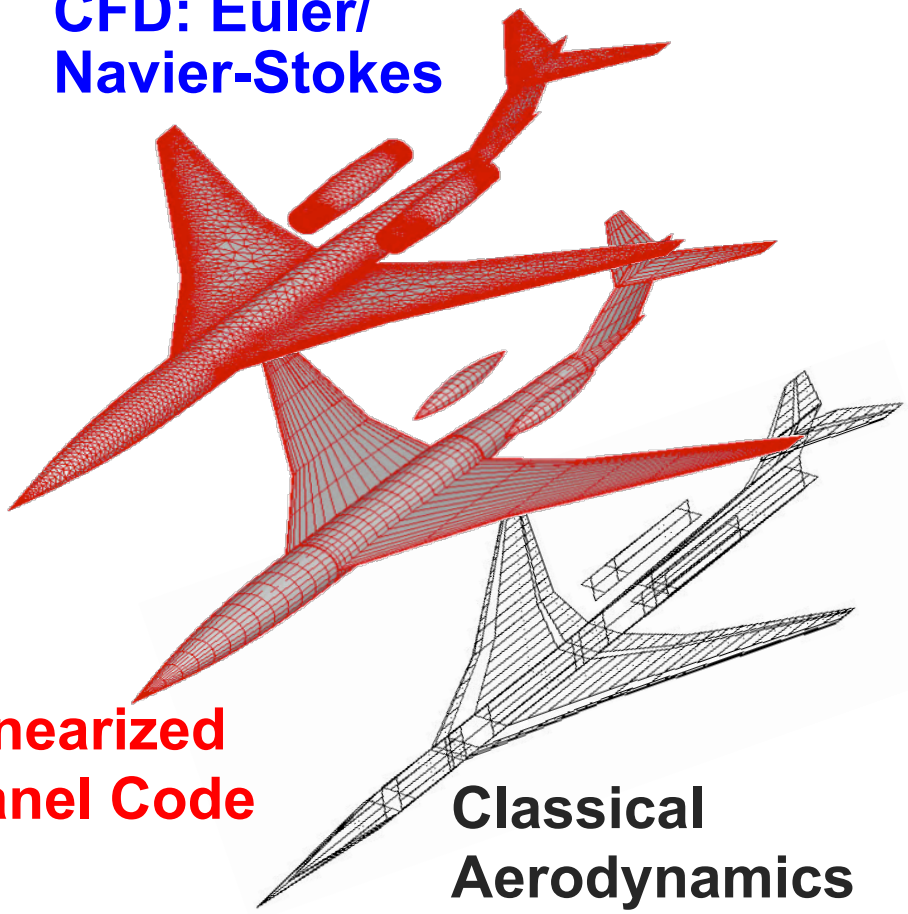

EI
x
0.5
0.4
0.3
0.2
0.1
0
50
100
150
200
250
300

Comparison of Data Fit Methods



Hierarchical Surrogate Models

CFD: Euler/ Navier-Stokes

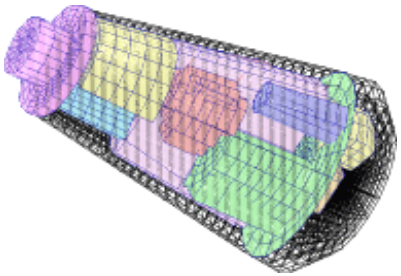


Linearized Panel Code

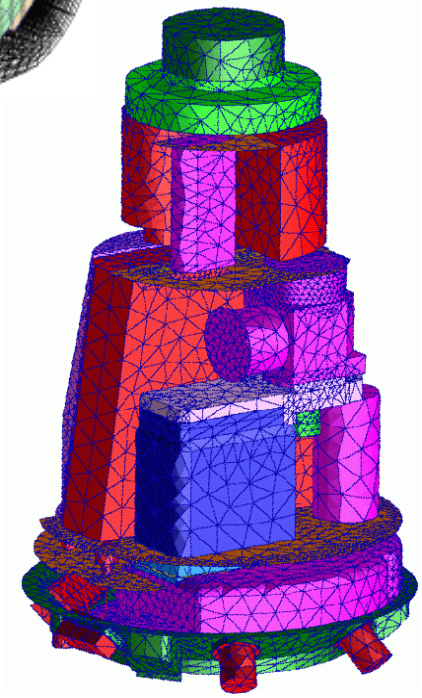
Classical Aerodynamics

J.J. Alonso, I. Kroo, Stanford University

Low-fidelity EM model: 30,000 DOF

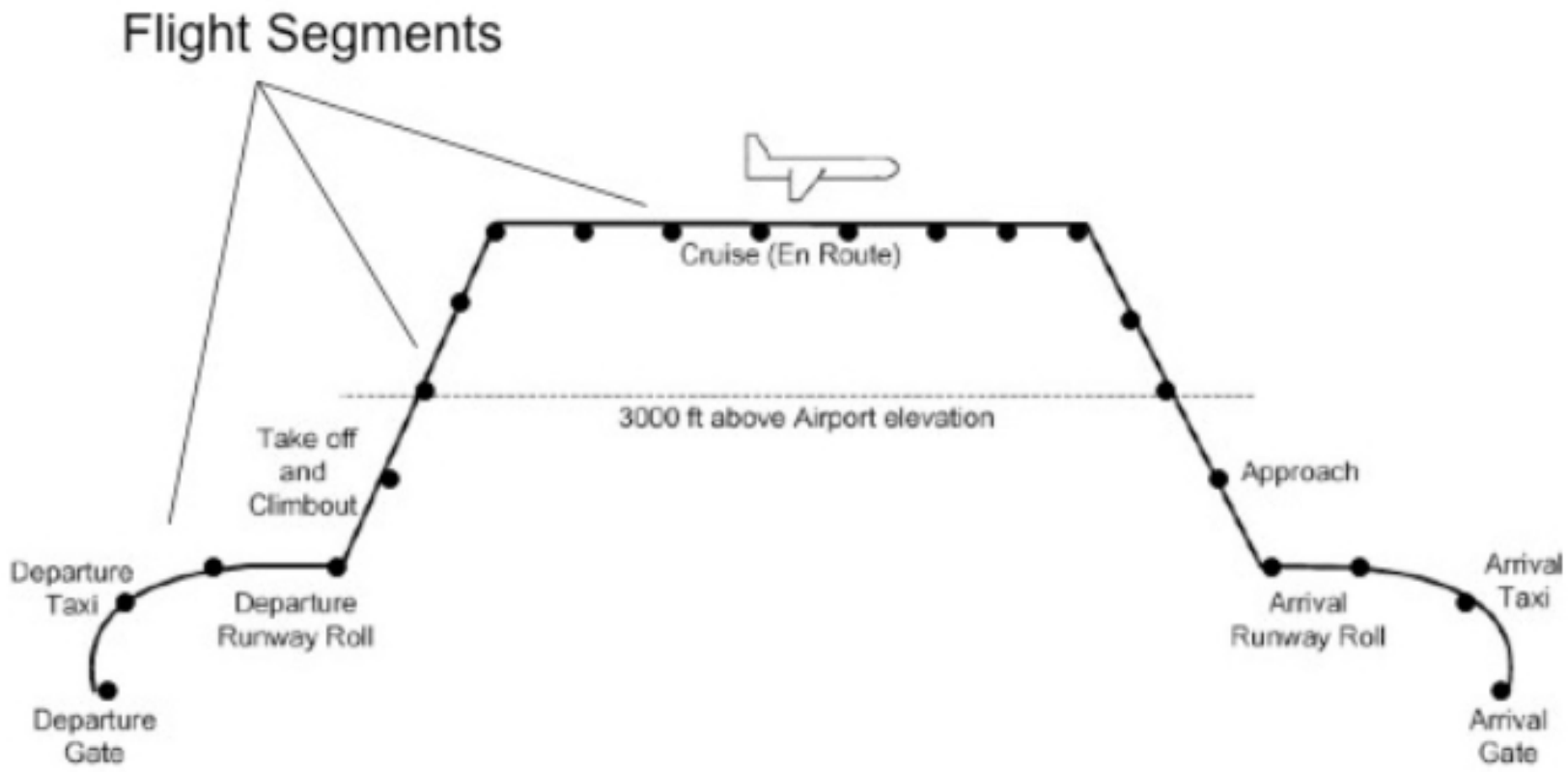


High-fidelity EM model: 800,000 DOF



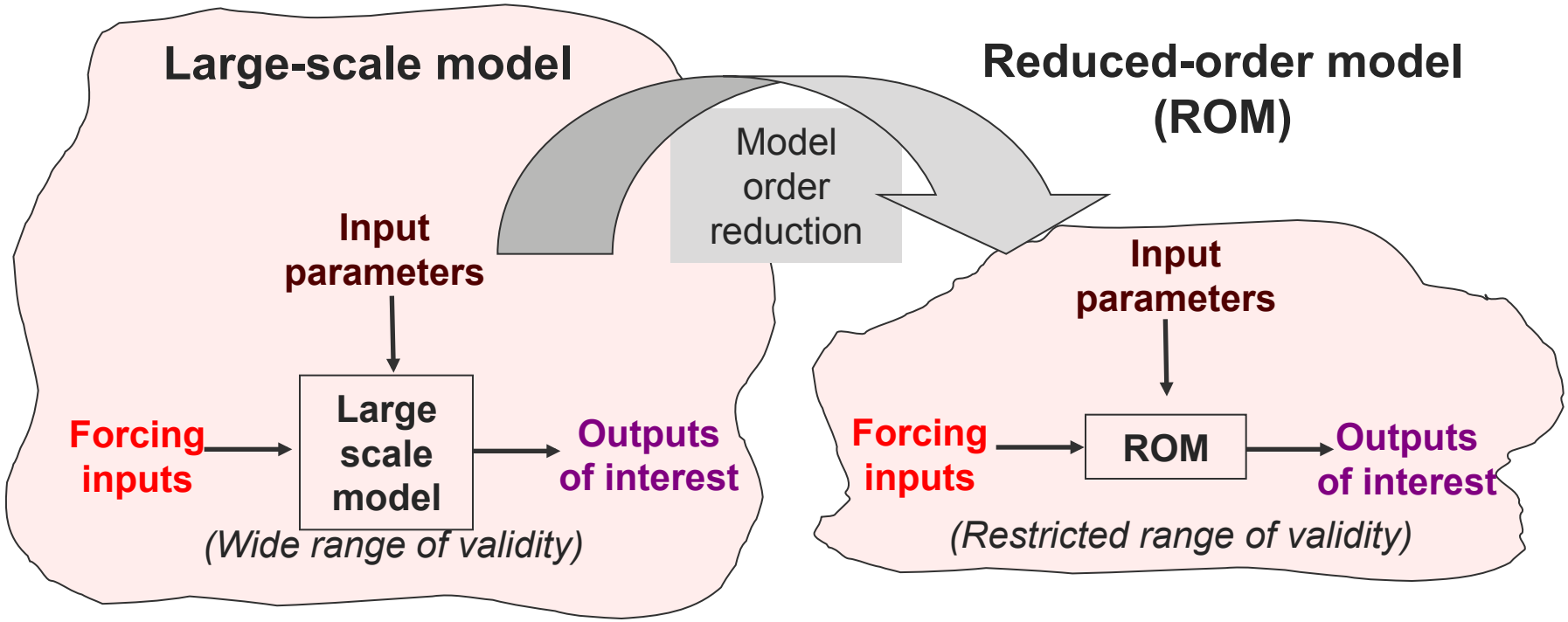
J. Castro, A. Giunta Sandia National Labs

Hierarchical Surrogates



Model Order Reduction

Systematic model order reduction to replicate large-scale model outputs of interest, over a restricted range of forcing inputs and input parameters.



Why Model Reduction?

When is the high-fidelity dynamical system model too expensive for online computations?

- Multidisciplinary applications
- Real-time applications
- Design and optimization
- Probabilistic applications

Parameterized Large-Scale Dynamical Systems

Arising, for example, from systems of ODEs or spatial discretization of PDEs describing the system of interest.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}(\mathbf{p})\mathbf{x} + \mathbf{B}(\mathbf{p})\mathbf{u} \\ \mathbf{y} &= \mathbf{C}(\mathbf{p})\mathbf{x} \end{aligned}$$

$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{p}, \mathbf{u}) \\ \mathbf{y} &= g(\mathbf{x}, \mathbf{p}, \mathbf{u}) \end{aligned}$$

$\mathbf{x} \in \mathbf{R}^N$: state vector

$\mathbf{u} \in \mathbf{R}^{N_i}$: input vector

$\mathbf{p} \in \mathbf{R}^{N_p}$: parameter vector

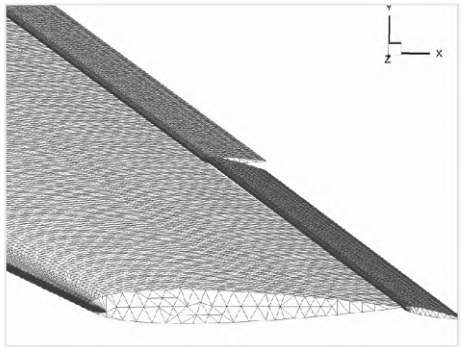
$\mathbf{y} \in \mathbf{R}^{N_o}$: output vector

Systems of this kind often appear as constraints in a design optimization problem (e.g., for optimization of systems governed by PDEs).

Example: CFD Systems

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}(\mathbf{p})\mathbf{x} + \mathbf{B}(\mathbf{p})\mathbf{u} \\ \mathbf{y} &= \mathbf{C}(\mathbf{p})\mathbf{x} \end{aligned} \qquad \begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{p}, \mathbf{u}) \\ \mathbf{y} &= g(\mathbf{x}, \mathbf{p}, \mathbf{u}) \end{aligned}$$

- $\mathbf{x}(t)$: vector of N flow unknowns
  e.g., 2D incompressible Navier Stokes
  P grid points, $N = 3P$
  $\mathbf{x} = [u_1 \; v_1 \; p_1 \; u_2 \; v_2 \; p_2 \cdots\cdots u_P \; v_P \; p_P]^T$
- $\mathbf{p}$: input parameters
  e.g., shape parameters, PDE coefficients
- $\mathbf{u}(t)$: forcing inputs
  e.g., flow disturbances, wing motion
- $\mathbf{y}(t)$: outputs
  e.g., flow characteristic, lift force

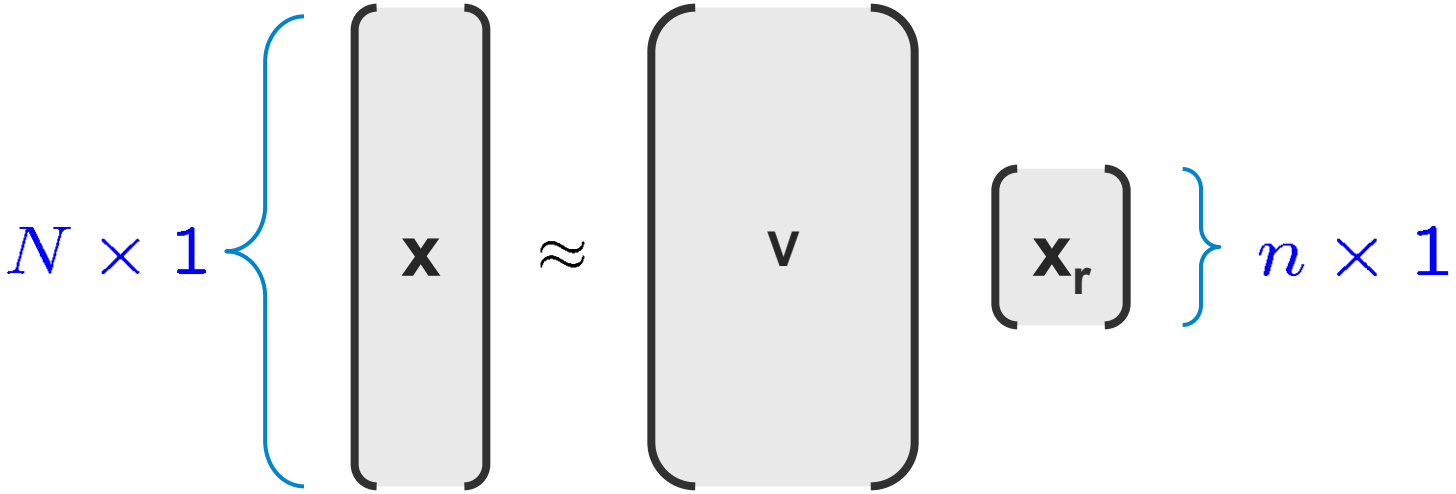


Projection-Based Reduced Models

- Approximate state by a linear combination of basis vectors

$$\mathbf{x} \approx \sum_{i=1}^{n} \mathbf{V}_i x_{r_i}$$

  - Define right basis, **V**



- Project equations onto reduced-order subspace $\mathbf{W}^T\mathbf{V} = \mathbf{I}$
  - Define left basis, **W** (Often use $\mathbf{W} = \mathbf{V}$)

Projection-Based Reduced Models

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{A}(\mathbf{p})\mathbf{x} + \mathbf{B}(\mathbf{p})\mathbf{u} \\ \mathbf{y} &= \mathbf{C}(\mathbf{p})\mathbf{x}\end{aligned}$$

$$\mathbf{x} \approx \mathbf{V}\mathbf{x}_r$$

$$\begin{aligned}\mathbf{r} &= \mathbf{V}\dot{\mathbf{x}}_r - \mathbf{A}\mathbf{V}\mathbf{x}_r - \mathbf{B}\mathbf{u} \\ \mathbf{y}_r &= \mathbf{C}\mathbf{V}\mathbf{x}_r\end{aligned}$$

$$\mathbf{W}^T\mathbf{r} = 0$$

$$\begin{aligned}\mathbf{A}_r(\mathbf{p}) &= \mathbf{W}^T\mathbf{A}(\mathbf{p})\mathbf{V} \\ \mathbf{B}_r(\mathbf{p}) &= \mathbf{W}^T\mathbf{B}(\mathbf{p}) \\ \mathbf{C}_r(\mathbf{p}) &= \mathbf{C}(\mathbf{p})\mathbf{V}\end{aligned}$$

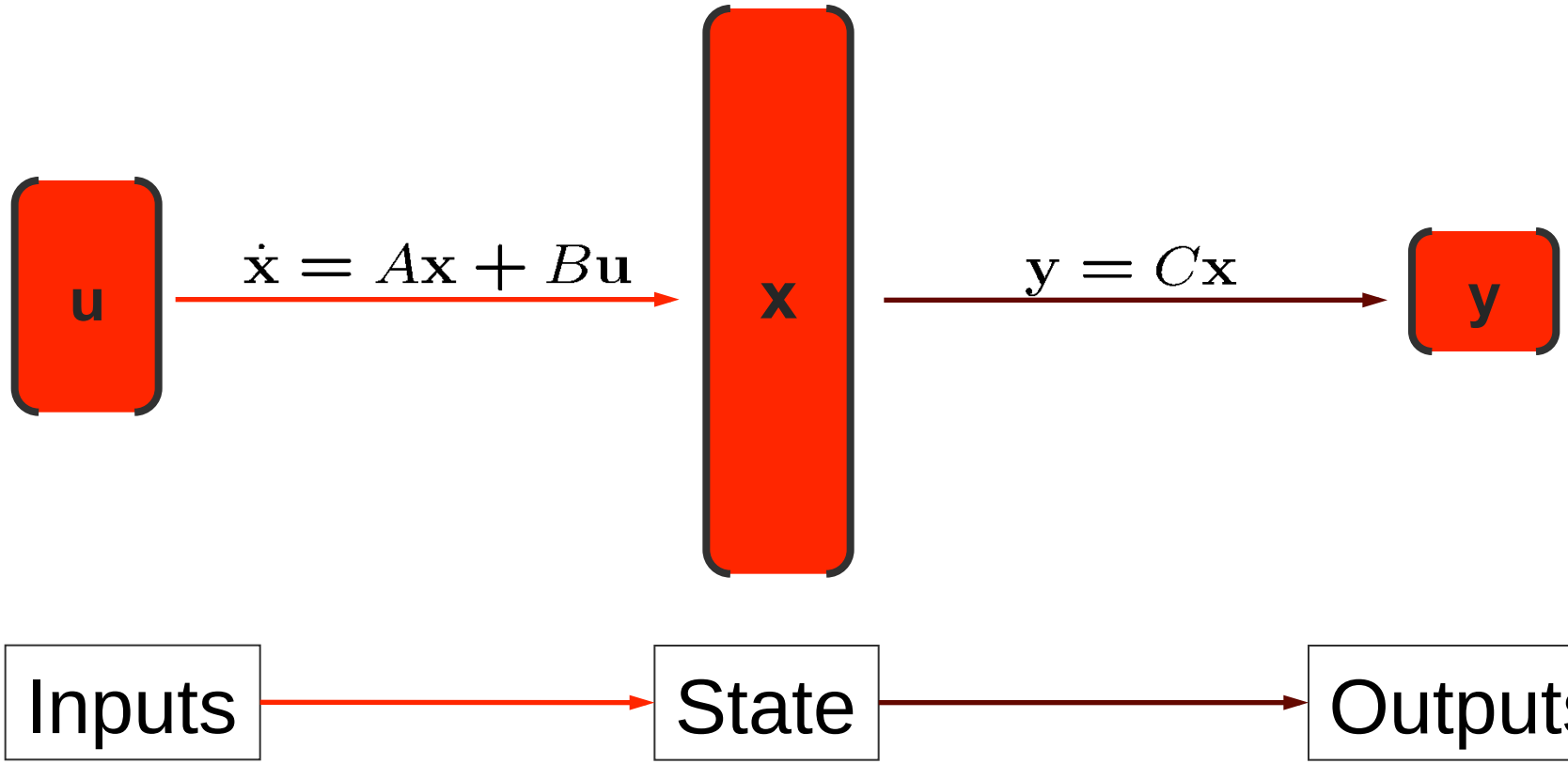
$$\begin{aligned}\dot{\mathbf{x}}_r &= \mathbf{A}_r(\mathbf{p})\mathbf{x}_r + \mathbf{B}_r(\mathbf{p})\mathbf{u} \\ \mathbf{y}_r &= \mathbf{C}_r(\mathbf{p})\mathbf{x}_r\end{aligned}$$

$\mathbf{x} \in \mathbf{R}^N$: state vector
$\mathbf{p} \in \mathbf{R}^{N_p}$: parameter vector
$\mathbf{u} \in \mathbf{R}^{N_i}$: input vector
$\mathbf{y} \in \mathbf{R}^{N_o}$: output vector

$\mathbf{x}_r \in \mathbf{R}^n$: reduced state vector
$\mathbf{V} \in \mathbf{R}^{N \times n}$: reduced basis

Why does model reduction work?

- Input → Output map is often much simpler than the full simulation model suggests



Some Large-Scale Reduction Methods

- Proper orthogonal decomposition *(Lumley, 1967; Sirovich, 1981; Berkooz, 1991; Deane et al. 1991; Holmes et al. 1996)*
  - Use data to generate empirical eigenfunctions
  - Time and frequency domain methods
- Krylov-subspace methods *(Gallivan, Grimme, & van Dooren, 1994; Feldmann & Freund, 1995; Grimme, 1997)*
  - Arnoldi, Lanczos methods
- Balanced truncation *(Moore, 1981; Sorensen & Antoulas, 2002; Li & White, 2002)*
  - Guaranteed stability and error bound for LTI systems
  - Close connection between POD and balanced truncation
- Reduced basis methods *(Noor & Peters, 1980)*
- Fourier model reduction *(Willcox & Megretski, 2005)*
  - Guaranteed stability and error bound for LTI systems

Example: Proper Orthogonal Decomposition

aka Karhunen-Loève expansions, Principal Components Analysis, Empirical Orthogonal Eigenfunctions, …)

Consider *M* snapshots $\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_M} \in \mathcal{R}^n$
(state solutions at different times or parameter values)

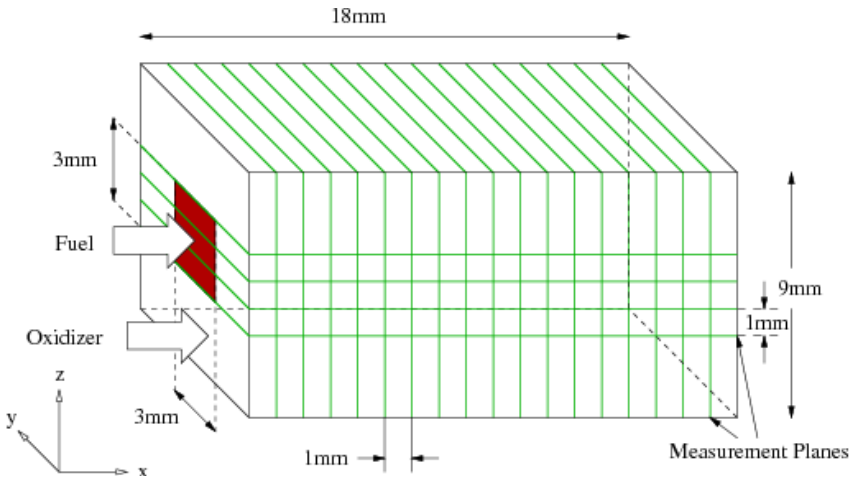
Form the snapshot matrix $X = [\mathbf{x}_1 \ \mathbf{x}_2 \ \ldots \ \mathbf{x}_M]$

Choose the *m* basis vectors $\mathbf{V} = [\mathbf{V}_1 \ \mathbf{V}_2 \cdots \mathbf{V}_m]$
to be left singular vectors of the snapshot matrix, with singular values $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq \sigma_{m+1} \geq \cdots \geq \sigma_M$

This is the optimal projection in a least squares sense:

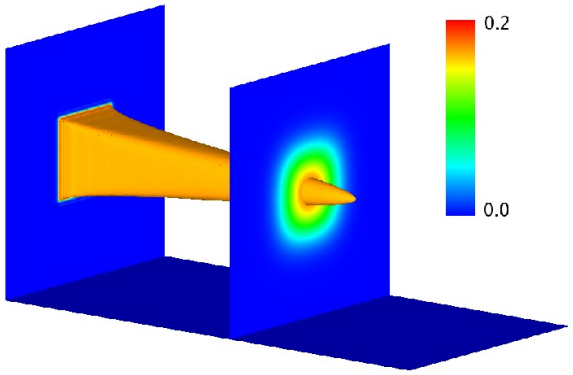
$$\min_V \sum_{i=1}^{M} ||\mathbf{x}_i - VV^T\mathbf{x}_i||_2^2 = \sum_{i=m+1}^{M} \sigma_i^2$$

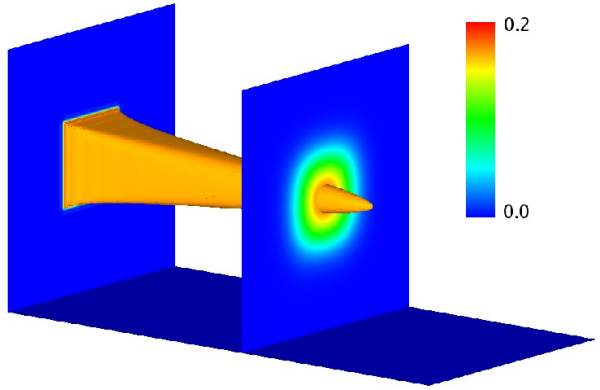
Combustion Chamber: POD



$$\begin{aligned} \dot{\mathbf{x}} &= f(\mathbf{x}, \mathbf{p}, \mathbf{u}) \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned}$$

Parameter: reaction parameters (**p**)
State: fuel concentration (**x**)
Output: fuel concentration (**y**)
Input: source (**u**)





3D FEM solution: *N*=8.5M, *M*=20M.
One forward solve: 13h CPU time.

POD-EIM: *n*=40, *m*=50.
One forward solve: < 0.1s CPU time.

Reduced-Basis Methods

Consider r feasible design vectors: $\mathbf{x}^1$, $\mathbf{x}^2$, ..., $\mathbf{x}^r$

We could consider the desired design to be a linear combination of these basis vectors:

$$\mathbf{x}^* = \sum_{i=1}^{r} \alpha_i \mathbf{x}^i + \mathbf{x}^C$$

scalar coefficient

basis vector

added for generality

Reduced-Basis Methods

We can now optimize $J(\mathbf{x})$ by finding the optimal values for the coefficients α_i.

dimension n → dimension r

- Do one full-order evaluation of resulting answer
- Approach is efficient if $r << n$
- Will give the true optimum only if $\mathbf{x}^*$ lies in the span of $\{\mathbf{x}^i\}$
- Basis vectors could be
  - previous designs
  - solutions over a particular range (DoE)
  - derived using a model reduction methods (e.g., proper orthogonal decomposition)

Reduced-Basis Example

Example using a reduced-basis approach (van der Plaats Fig 7-2): airfoil design for a unique application.

- Many airfoil shapes with known performance are available
- Design variables are (x,y) coordinates at chordwise locations (n~100)
- Use four basis airfoil shapes (low-speed airfoils) which contain the n geometry points
- Plus two basis shapes which allow trailing edge thickness to vary
- r=6 (r<<n)
- Optimize for high speed, maximum lift with a constraint on drag

Reduced-Basis Example

From Vanderplaats Figs. 7-2 and 7-3, pg. 260

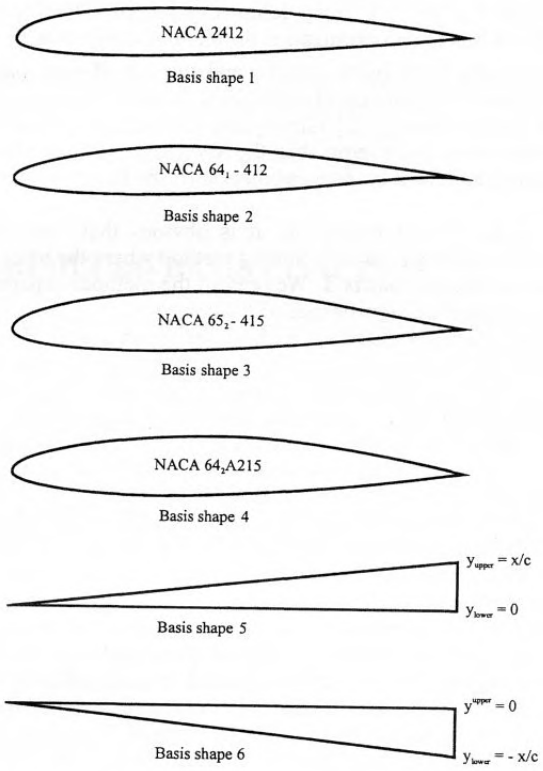


Figure 7-2 Airfoil basic shapes.

basis functions

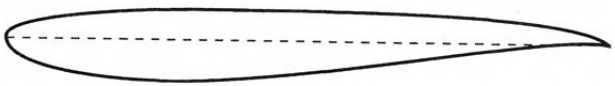


Figure 7-3 Minimum-drag airfoil.

result of optimization using reduced basis

Lecture Summary

- A number of ways to create approximations, or surrogate models
- Each has its own area of application, advantages, and disadvantages
- Data fit surrogates
  - Polynomial response surfaces
  - Kriging
- Model order reduction
  - Reduced basis
  - Proper orthogonal decomposition
- Hierarchical models

References

Alexandrov, N., Dennis, J.E., Lewis, R.M. and Torczon, V., "A trust region framework for managing the use of approximation models in optimization", NASA CR-201745, ICASE Report No. 97-50, October 1997.

Barthelemy, J-F. M. and Haftka, R.T., "Approximation concepts for optimum structural design – a review", *Structural Optimization*, 5:129-144, 1993.

Bashir, O., Willcox, K., Ghattas, O., van Bloemen Waanders, B., and Hill, J., "Hessian-Based Model Reduction for Large-Scale Systems with Initial Condition Inputs," *International Journal for Numerical Methods in Engineering*, Vol. 73, Issue 6, pp. 844-868, 2008.

Bui-Thanh, T., Willcox, K., and Ghattas, O. (2008a). Model reduction for large-scale systems with high-dimensional parametric input space. SIAM Journal on Scientific Computing, 30(6): 3270{3288.

Bui-Thanh, T., Willcox, K., and Ghattas, O. (2008b). Parametric reduced-order models for probabilistic analysis of unsteady aerodynamic applications. AIAA Journal, 46(10):2520-2529.

Conn, A.R., Scheinberg, K. and Vicente, L., "Global Convergence of General Derivative-Free Trust-Region Algorithms to First- and Second-Order Critical Points," *SIAM Journal of Optimization*, Vol. 20, No.1, pp. 387-415, 2009.

Deane, A., Kevrekidis, I., Karniadakis, G., and Orszag, S. (1991). Low-dimensional models for complex geometry flows: Application to grooved channels and circular cylinders. Phys. Fluids, 3(10):2337-2354.

Dowell, E. and Hall, K. (2001). Modeling of fluid-structure interaction. Annual Review of Fluid Mechanics, 33:445-90.

References

Eldred, M. S., and D. M. Dunlavy. "Formulations for surrogate-based optimization with data fit, multifidelity, and reduced-order models." Proceedings of the 11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA-2006-7117, 2006.

Galbally, D., Fidkowski, K., Willcox, K. and Ghattas, O., "Nonlinear Model Reduction for Uncertainty Quantification in Large-Scale Inverse Problems," *International Journal for Numerical Methods in Engineering*, Volume 81, Issue 12, March 2010, pp. 1581-1608.

Gallivan, K., Grimme, E., and Van Dooren, P. (1994). Pade approximation of large-scale dynamic systems with Lanczos methods. Proceedings of the 33rd IEEE Conference on Decision and Control.

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

Giunta, A.A. and Watson, L.T.,"A comparison of approximation modeling techniques: polynomial versus interpolating models", AIAA Paper 98-4758, 1998.

Grimme, E. (1997). Krylov Projection Methods for Model Reduction. PhD thesis, Coordinated-Science Laboratory, University of Illinois at Urbana-Champaign.

Gugercin, S. and Antoulas, A. (2004). A survey of model reduction by balanced truncation and some new results. International Journal of Control, 77:748-766.

Jones, D.R., "A taxonomy of global optimization methods based on response surfaces," Journal of Global Optimization, 21, 345-383, 2001.

Kennedy, M. and O'Hagan, A. (2001). Bayesian calibration of computer models. Journal of the Royal Statistical Society, 63(2):425-464.

LeGresley, P.A. and Alonso, J.J., "Airfoil design optimization using reduced order models based on proper orthogonal decomposition", AIAA Paper 2000-2545, 2000.

References

Lophaven, S., Nielsen, H., and Sondergaard, J. (2002). Aspects of the Matlab toolbox DACE. Technical Report IMM-REP-2002-13, Technical University of Denmark.

March, A. and Willcox, K., "Convergent multifidelity optimization using Bayesian model calibration." *Structural and Multidisciplinary Optimization*, Vol. 46, No.1, 2012, pp. 93-109.

March, A. and Willcox, K., "A Provably Convergent Multifidelity Optimization Algorithm not Requiring High-Fidelity Derivatives," *AIAA Journal*, Vol. 50, No. 5, 2012, pp.1079-1089.

Noor, A. and Peters, J. (1980). Reduced basis technique for nonlinear analysis of structures. AIAA Journal, 18(4):455-462.

Penzl, T. (2006). Algorithms for model reduction of large dynamical systems. Linear Algebra and its Applications, 415(2-3):322-343.

Robinson, T., Willcox, K., Eldred, M., and Haimes, R. "Multifidelity Optimization for Variable-Complexity Design," *AIAA Journal*, Vol.46, No.11, pp. 2814-2822, 2008.

Simpson, T., Peplinski, J., Koch, P., and Allen, J. (2001). Metamodels for computer based engineering design: Survey and recommendations. Engineering with Computers, 17:129-150.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. Part 1: Coherent structures. Quarterly of Applied Mathematics, 45(3):561-571.

Sorensen, D. and Antoulas, A. (2002). The Sylvester equation and approximate balanced reduction. Linear Algebra and its Applications, 351-352:671-700.

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

Lecture 17: Multidisciplinary System Analysis and Design Optimization (MSADO)

Design for Value

April 15, 2015

Prof. Douglas Allaire

Lecture Outline

- An MDO value framework
- Lifecycle cost models
- Value metrics & valuation techniques
- Value-based MDO
- Aircraft example

Optimal Design

- Traditionally, design has focused on **performance**

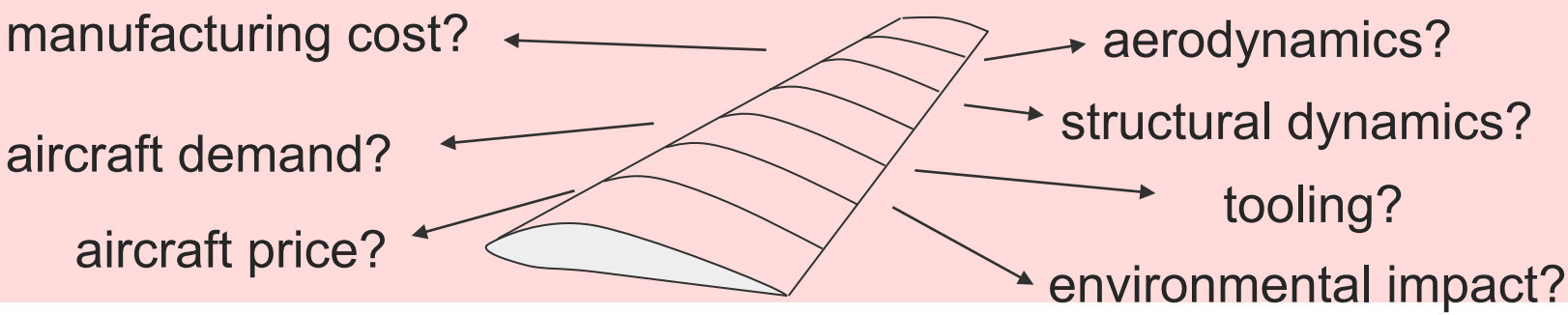
  e.g. for aircraft design

  optimal = minimum weight

- Increasingly, **cost** becomes important
- 85% of total lifecycle cost is locked in by the end of preliminary design.
- But *minimum weight ≠ minimum cost ≠ maximum value*
- What is an appropriate value metric?

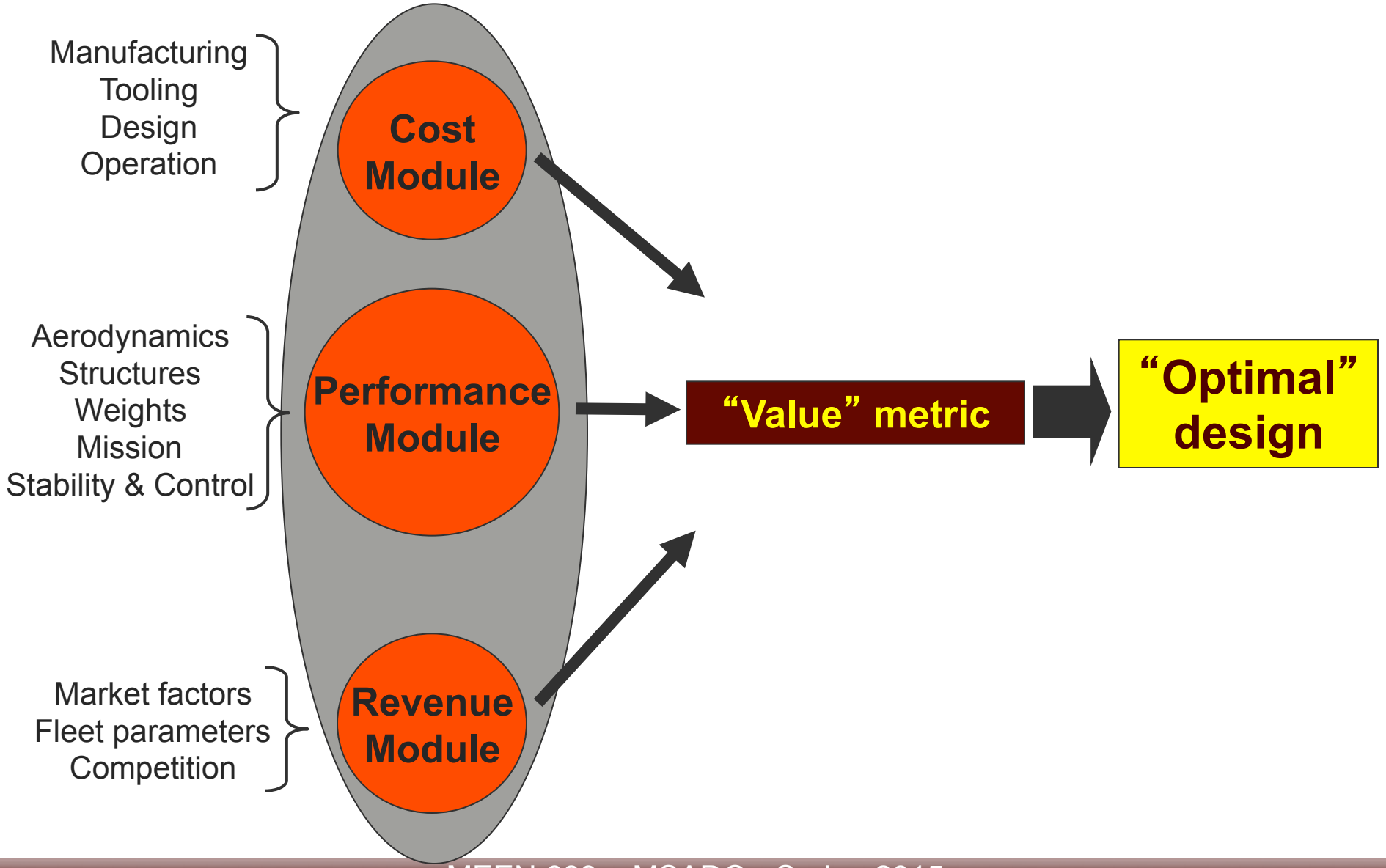
Design Example

- We need to design a particular portion of the wing
- Traditional approach: balance the aero & structural requirements, minimize weight
- We should consider cost: what about an option that is very cheap to manufacture but performance is worse?



- How do we trade performance and cost?
- How much performance are we willing to give up for $100 saved?
- What is the impact of the low-cost design on price and demand of this aircraft?
- What is the impact of this design decision on the other aircraft I build?
- What about market uncertainty?

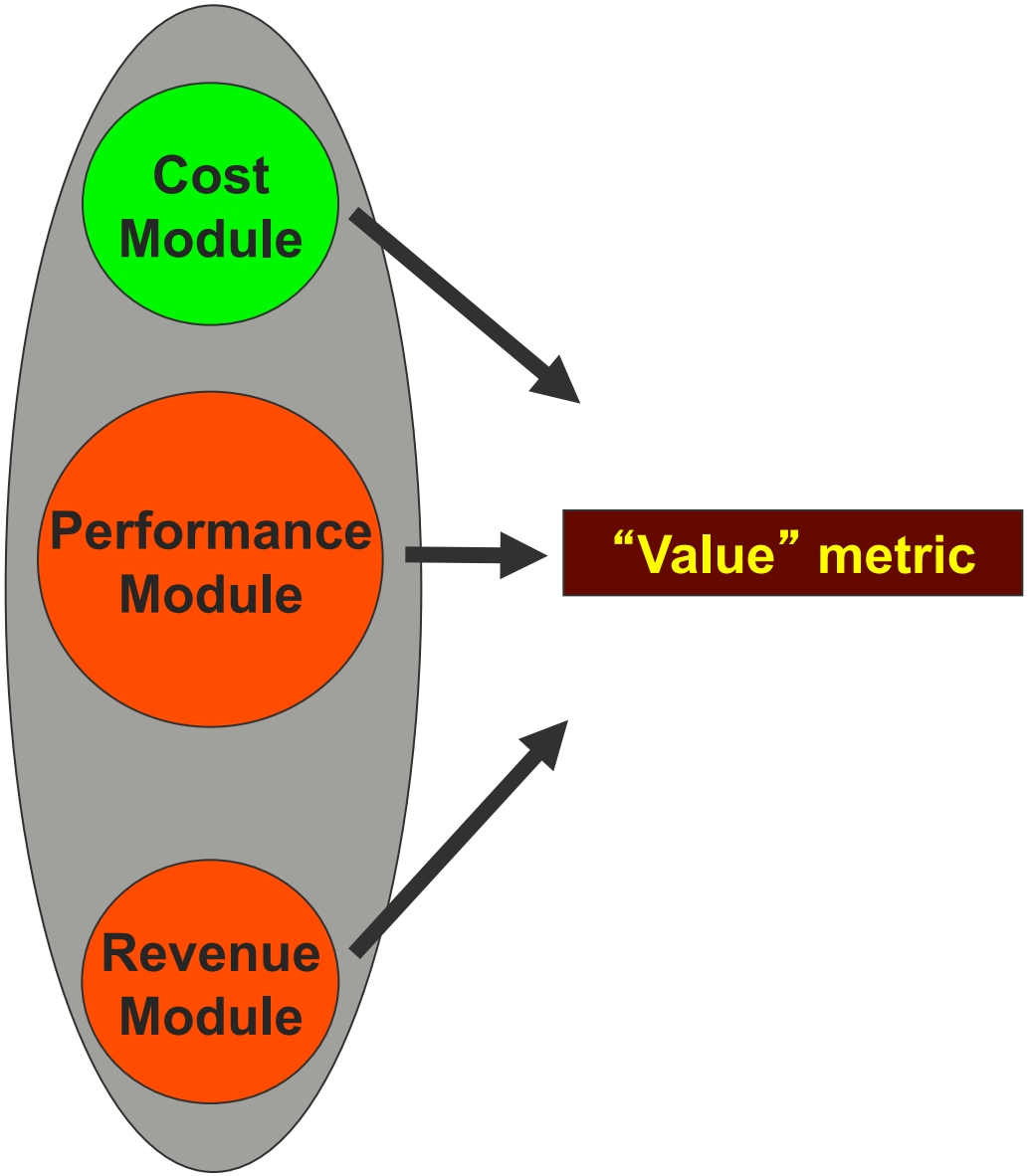
Value Optimization Framework



Challenges

- Cost and revenue are difficult to model
  - often models are based on empirical data
  - how to predict for new designs
- Uncertainty of market
- Long program length
- Time value of money
- Valuing flexibility
- Performance/financial groups even more uncoupled than engineering disciplines

Cost Model



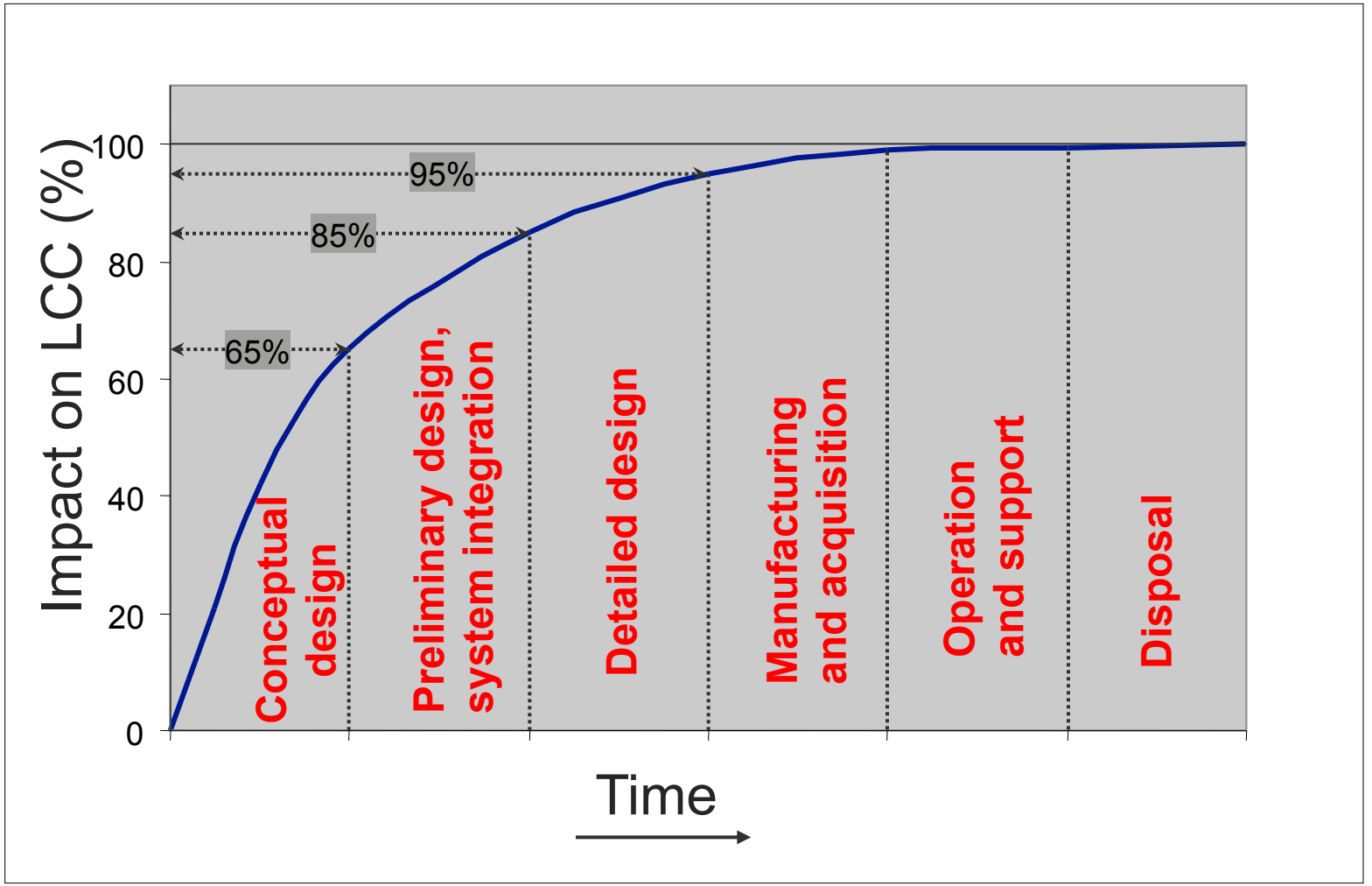
Need to model the lifecycle cost of the system.

Life cycle :
Design - Manufacture - Operation - Disposal

Lifecycle cost :
Total cost of program over life cycle

85% of Total LCC is locked in by the end of preliminary design.

Lifecycle Cost



(From Roskam, Figure 2.3)

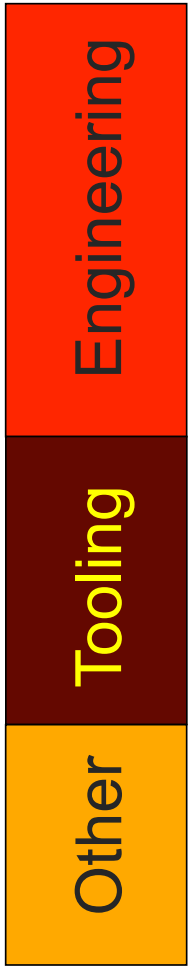
Non-Recurring Cost

Cost incurred one time only:

Engineering
- airframe design/analysis
- configuration control
- systems engineering

Tooling
- design of tools and fixtures
- fabrication of tools and fixtures

Other
- development support
- flight testing



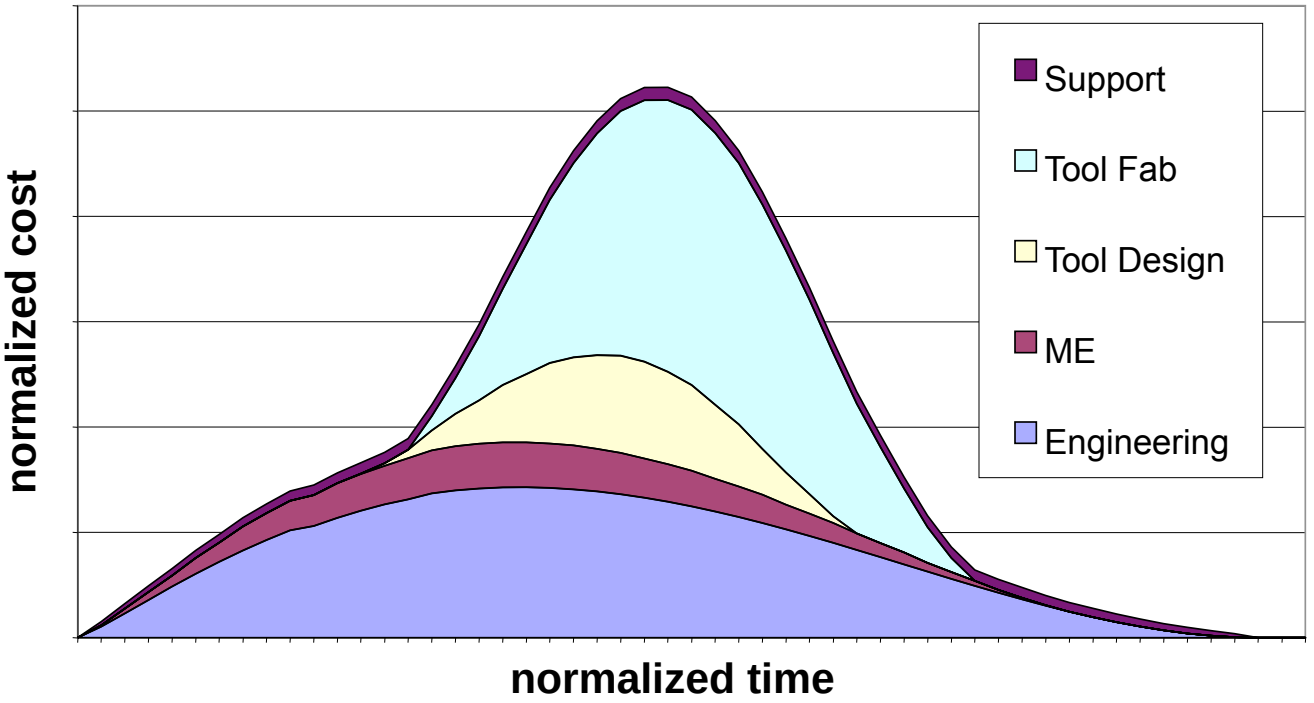
Development Cost Model

- Cashflow profiles based on beta curve:

$$c(t) = Kt^{\alpha-1}(1-t)^{\beta-1}$$

- Typical development time ~6 years
- Learning effects captured – span, cost

(from Markish)



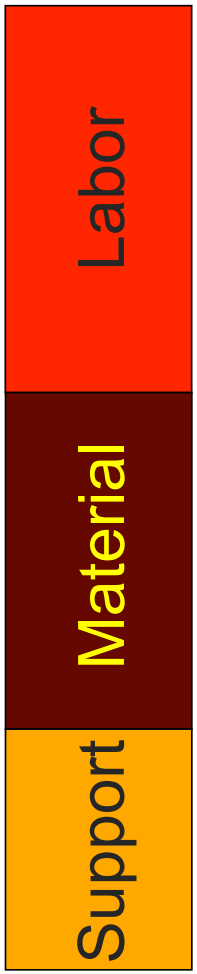
Recurring Cost

Cost incurred per unit:

Labor
- fabrication
- assembly
- integration

Material to manufacture
- raw material
- purchased outside production
- purchased equipment

Production support
- QA
- production tooling support
- engineering support



Learning Curve

As more units are made, the recurring cost per unit decreases.

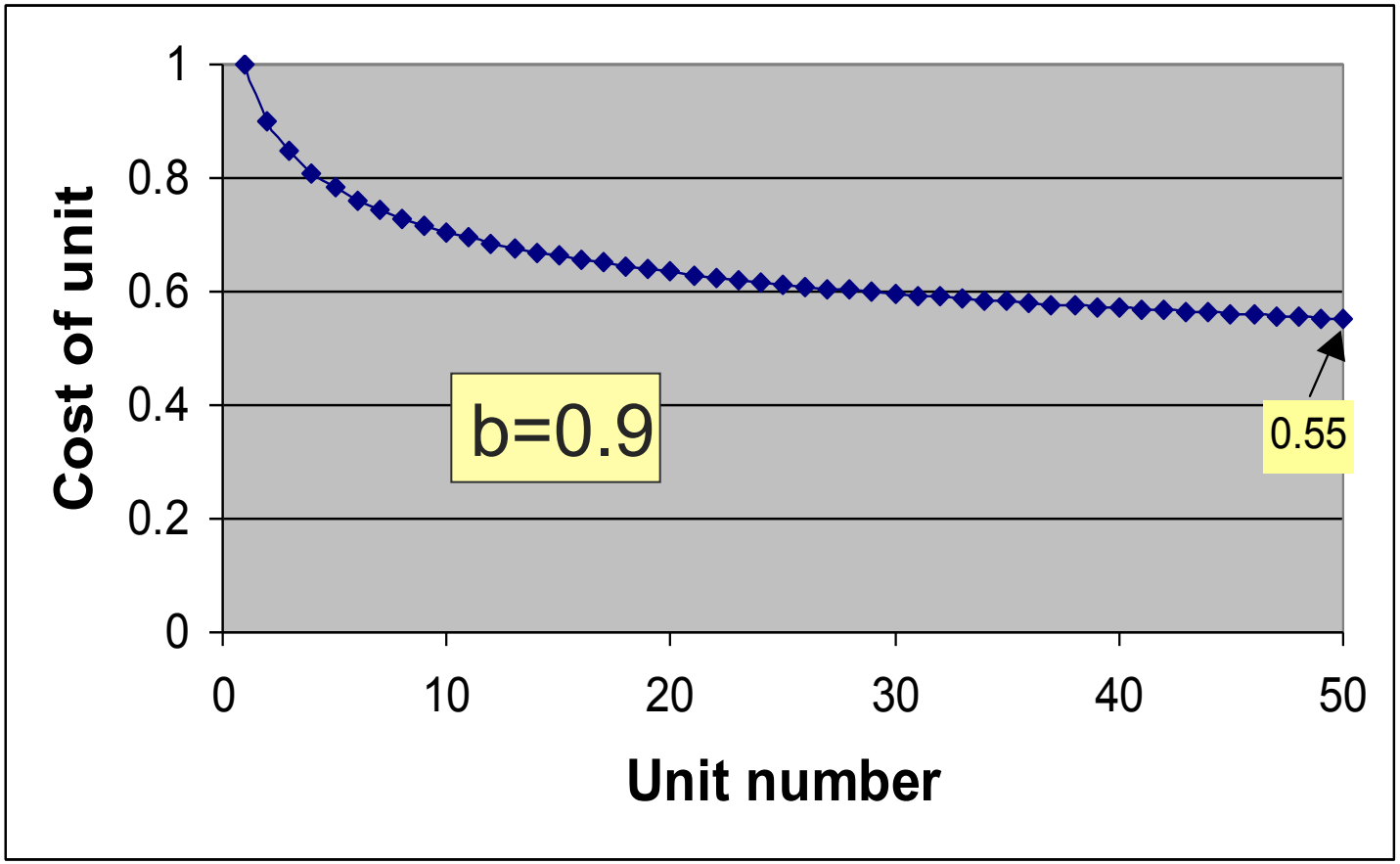
This is the learning curve effect.

e.g. Fabrication is done more quickly, less material is wasted.

$$Y_x = Y_0 x^n$$

Y_x = number of hours to produce unit x
n = log b/log 2
b = learning curve factor (~80-100%)

Learning Curve

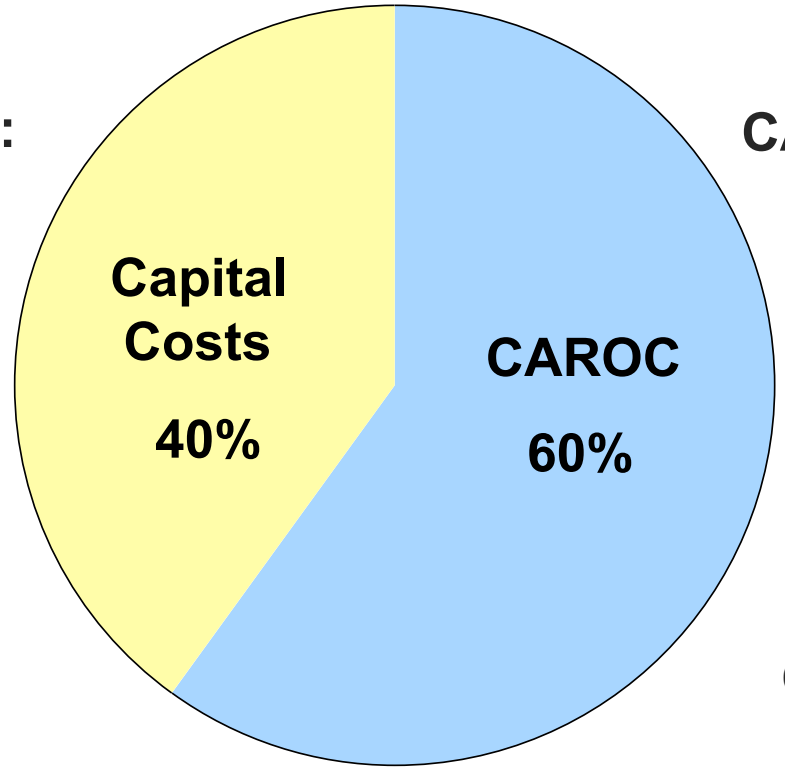


Every time production doubles, cost is reduced by a factor of 0.9

Typical LC slopes: Fab 90%, Assembly 75%, Material 98%

Airplane Related Operating Costs

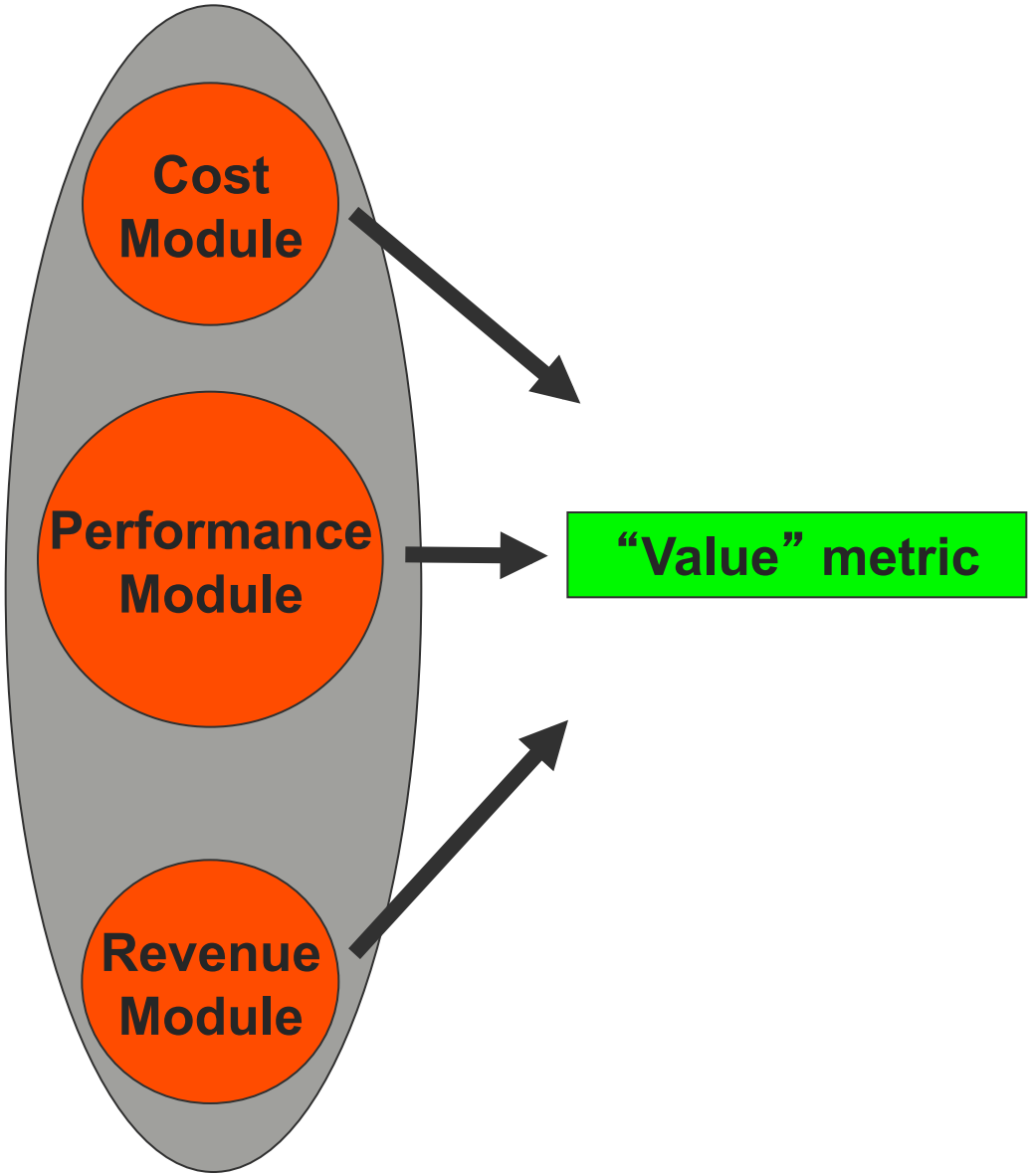
CAPITAL COSTS:
Financing
Insurance
Depreciation



CASH AIRPLANE RELATED OPERATING COSTS:
Crew
Fuel
Maintenance
Landing
Ground Handling
GPE Depreciation
GPE Maintenance
Control & Communications

CAROC is only 60% - ownership costs are significant!

Value Metric

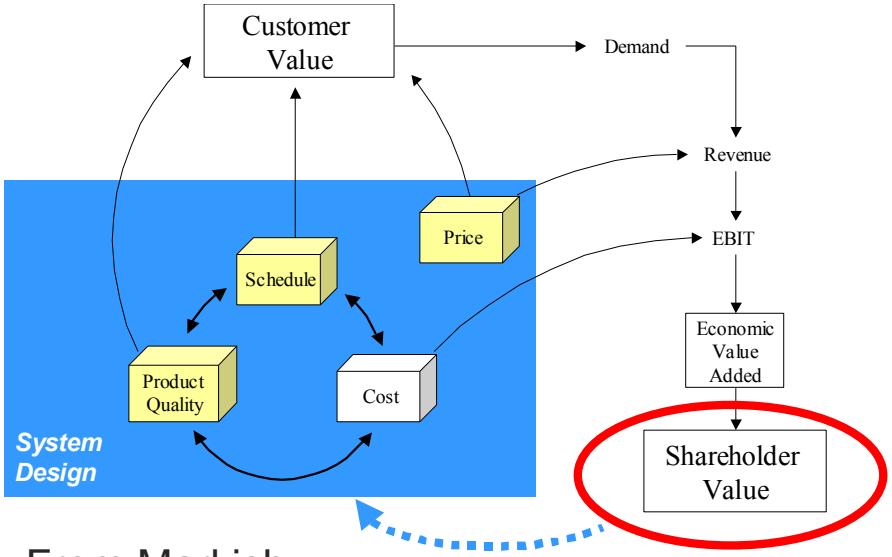


Need to provide a quantitative metric that incorporates cost, performance and revenue information.

In optimization, need to be especially careful about what metric we choose...

What is Value?

- Objective function could be different for each stakeholder *e.g.* manufacturer vs. airline vs. flying public
- Program related parameters vs. technical parameters
  cost, price, production quantity, timing
- Traditionally program-related design uncoupled from technical design



From Markish, Fig. 1, pg 20

Customer value *derived from quality, timeliness, price.* ***Shareholder value*** *derived from cost and revenue, which is directly related to customer satisfaction.*

Value Metrics

Traditional Metrics

performance
weight
speed

Augmented Metrics

cost
revenue
profit
quietness
emissions
commonality
...

The definition of value will vary depending on your system and your role as a stakeholder, but we must define a quantifiable metric.

Valuation Techniques

Investor questions:

- How much will I need to invest?
- How much will I get back?
- When will I get my money back?
- How much is this going to cost me?
- How are you handling risk & uncertainty?

Investment Criteria

- Net present value
- Payback
- Discounted payback
- Internal rate of return
- Return on investment

Net Present Value (NPV)

- Measure of present value of various cash flows in different periods in the future
- Cash flow in any given period discounted by the value of a dollar today at that point in the future
  - "Time is money"
  - A dollar tomorrow is worth less today since if properly invested, a dollar today would be worth more tomorrow
- Rate at which future cash flows are discounted is determined by the "discount rate"
  - Discount rate is equal to the amount of interest the investor could earn in a single time period (usually a year) if s/he were to invest in a "safer" investment

Discounted Cash Flow (DCF)

- Forecast the cash flows, C_0, C_1, ..., C_T of the project over its economic life
  - Treat investments as negative cash flow
- Determine the appropriate opportunity cost of capital (i.e. determine the discount rate r)
- Use opportunity cost of capital to discount the future cash flow of the project
- Sum the discounted cash flows to get the net present value (NPV)

$$NPV = C_0 + \frac{C_1}{1+r} + \frac{C_2}{(1+r)^2} + \ldots + \frac{C_T}{(1+r)^T}$$

DCF Example

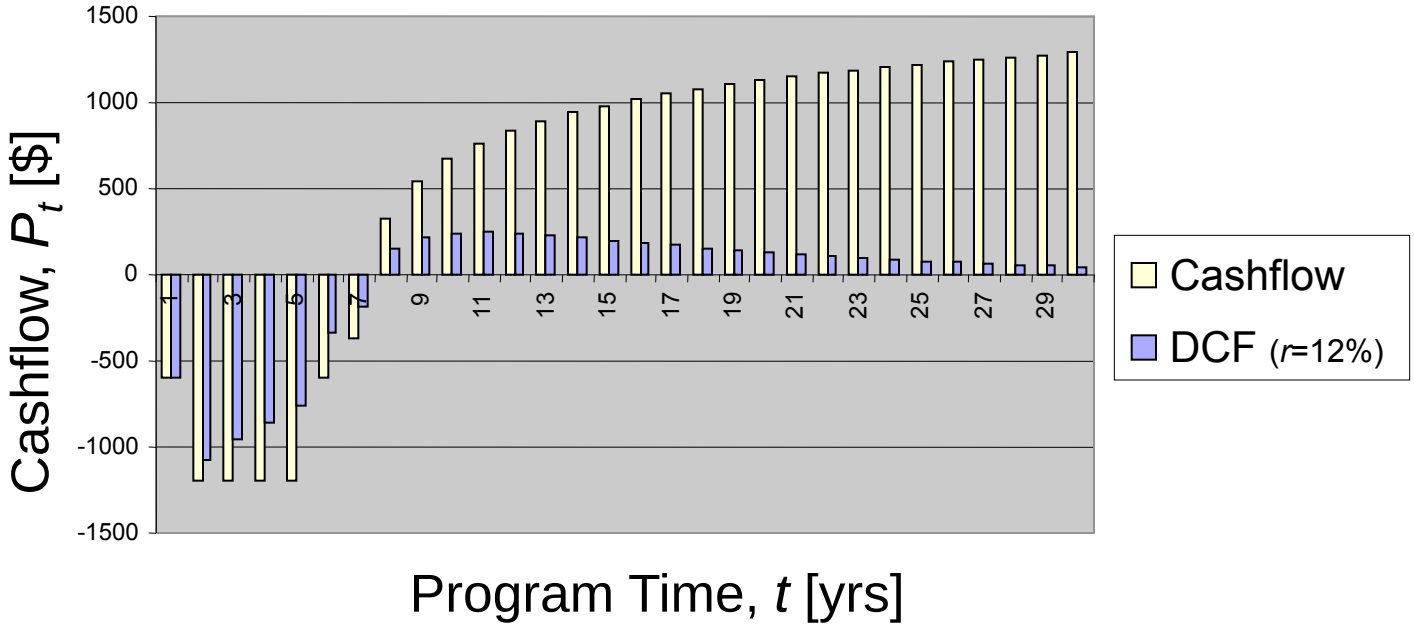
Period	Discount Factor	Cash Flow	Present Value
0	1	-150,000	-150,000
1	0.935	-100,000	-93,500
2	0.873	+300000	+261,000
Discount rate = 7%		NPV =	$18,400

Risk-Adjusted Discount Rate

- DCF analysis assumes a fixed schedule of cash flows
- What about uncertainty?
- Common approach: use a risk-adjusted discount rate
- The discount rate is often used to reflect the risk associated with a project: the riskier the project, use a higher discount rate
- Typical discount rates for commercial aircraft programs: 12-20%
- Issues with this approach?

Net Present Value (NPV)

$$NPV = \sum_{t=0}^{T} \frac{C_t}{(1+r)^t}$$



Payback Period

- How long it takes before entire initial investment is recovered through revenue
- Insensitive to time value of money, i.e. no discounting
- Gives equal weight to cash flows before investment is recovered & no weight to cash flows after cut-off date
- Cannot distinguish between projects with different NPV (insensitive to order of cash flows)

Discounted Payback

- Payback criterion modified to account for the time value of money
  - Cash flows before cut-off date are discounted
- Overcomes objection that equal weight is given to all flows before investment is recovered
- Cash flows after investment is recovered still not given any weight

Internal Rate of Return (IRR)

- Investment criterion is "rate of return must be greater than the opportunity cost of capital"
- Internal rate of return is equal to the discount rate for which the NPV is equal to zero

$$NPV = C_0 + \frac{C_1}{1+IRR} + \frac{C_2}{(1+IRR)^2} + \ldots + \frac{C_T}{(1+IRR)^T} = 0$$

- IRR solution is not unique
  - Multiple rates of return for same project
- IRR doesn't always correlate with NPV
  - NPV does not always decrease as discount rate increases

Return on Investment (ROI)

- Return of an action divided by the cost of that action

$$ROI = \frac{\text{revenue} - \text{cost}}{\text{cost}}$$

- Need to decide whether to use actual or discounted cashflows

Real Options Valuation Approach

- In reality:
  - Cashflows are uncertain
  - Ability to make decisions as future unfolds
- View an aircraft program as a series of investment decisions
- Spending money on development today gives the option to build and sell aircraft at a later date
- Better valuation metric: ***expected NPV*** from dynamic programming algorithm (Markish, 2002)

Principle of Optimality

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Policy: a rule that determines what the controls (decisions) should be, given any value of the state.

Dynamic Programming: Problem Formulation

- The firm:
  - Portfolio of designs
  - Sequential development phases
  - Decision making
- The market:
  - Sale price has a steady growth rate
  - Quantity demanded is unpredictable
  - Units built = units demanded
- Problem objective:
  - Which aircraft to design?
  - Which aircraft to produce?
  - When?

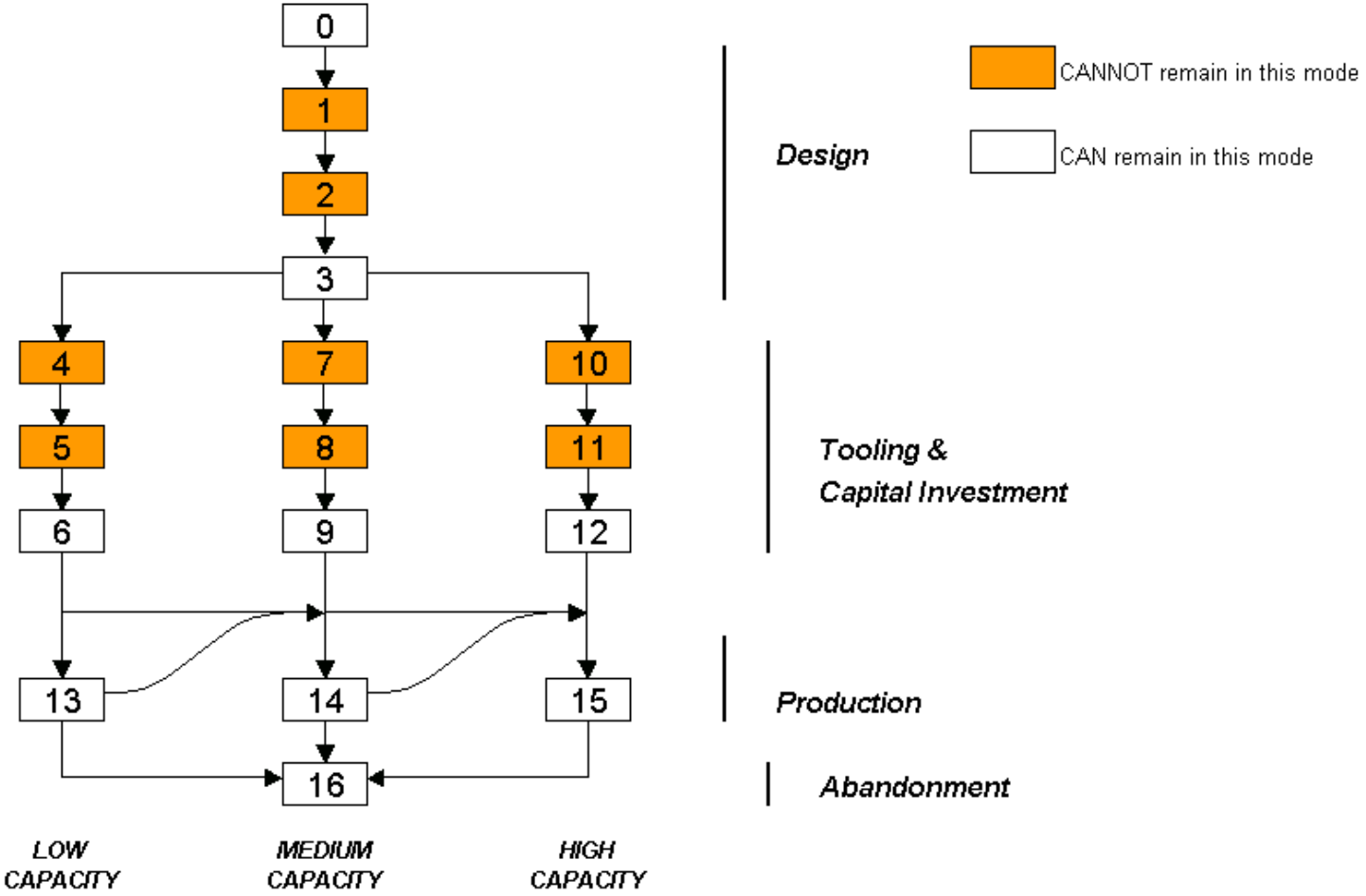
Dynamic Programming: Problem Elements

1. State variables s_t
2. Control variables u_t
3. Randomness
4. Profit function
5. Dynamics

- Solution: $$F_t(s_t) = \max_{u_t}\left\{\pi_t(s_t,u_t) + \frac{1}{1+r}E_t\left[F_{t+1}(s_{t+1})\right]\right\}$$
- Solve recursively

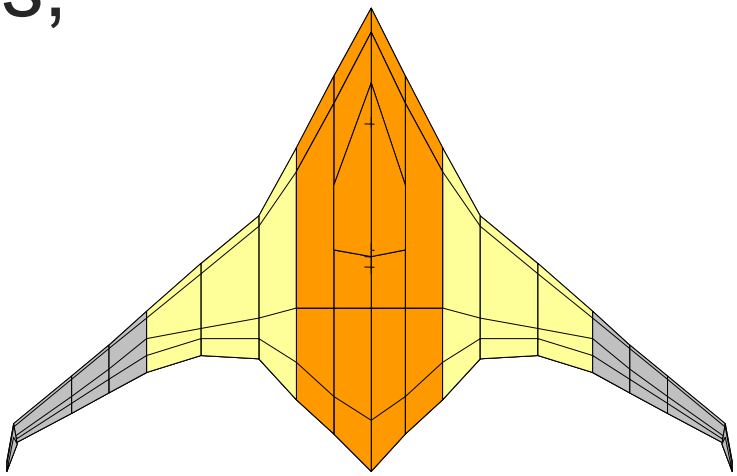
Dynamic Programming: Operating Modes

How to model decision making?

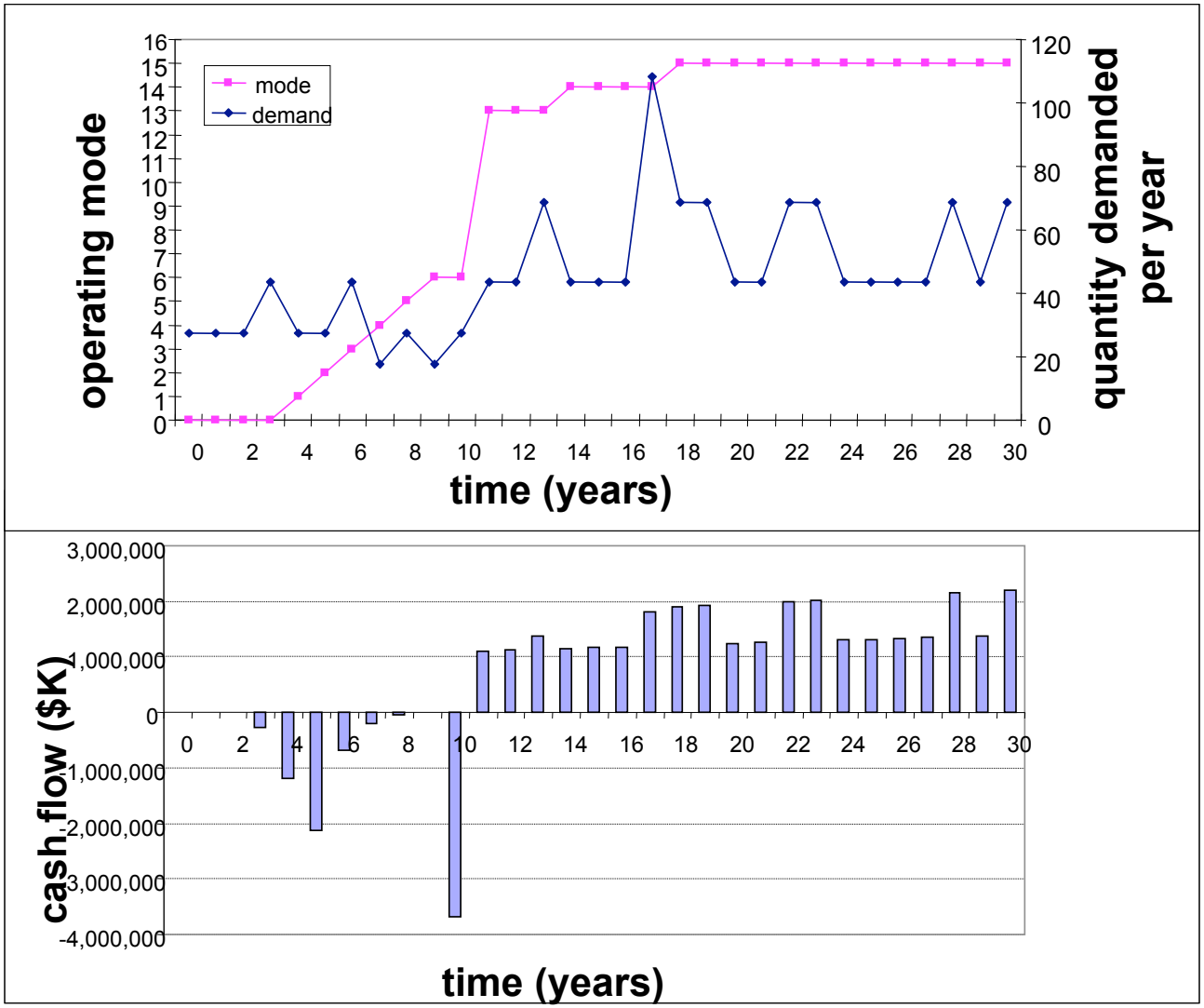


Example: BWB

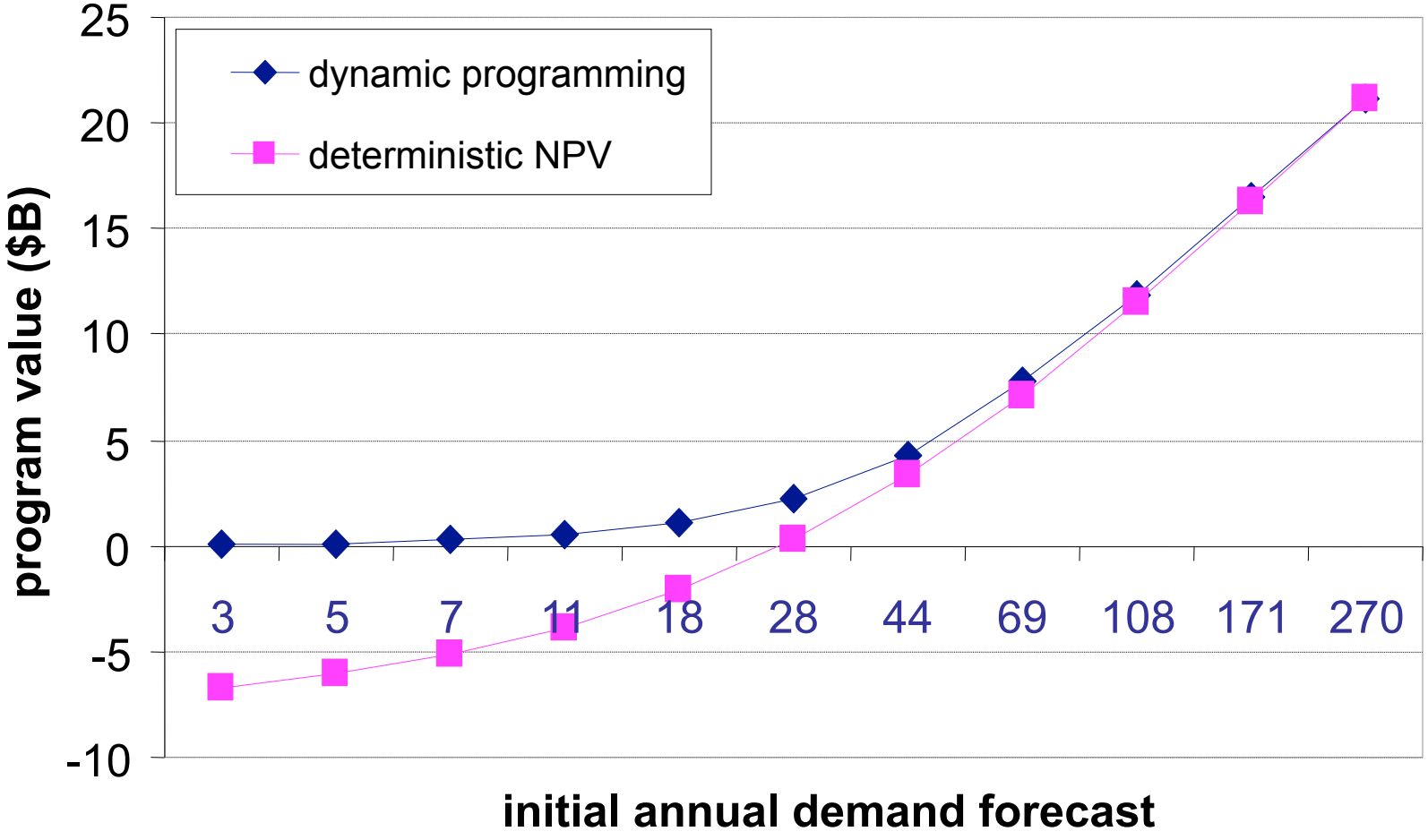
- Blended-Wing-Body (BWB):
  - Proposed new jet transport concept
- 250-seat, long range
- Part of a larger family sharing common centerbody bays, wings, ...



BWB Example: Simulation Run



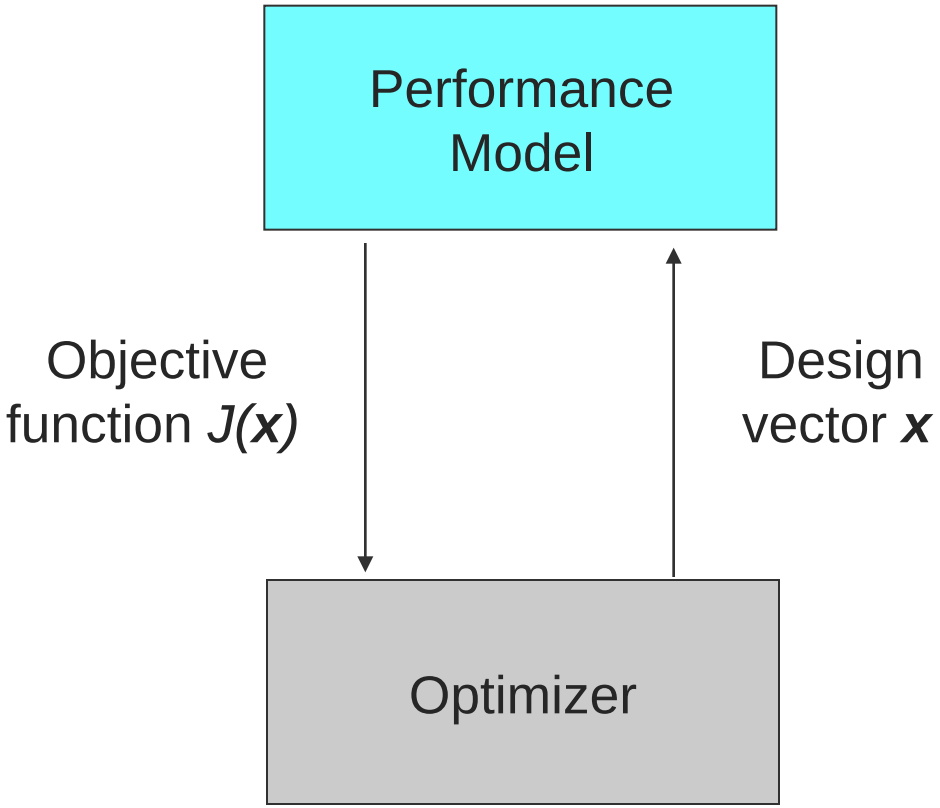
BWB Example: Importance of Flexibility



At baseline of 28 aircraft, DP value is $2.26B versus deterministic value of $325M

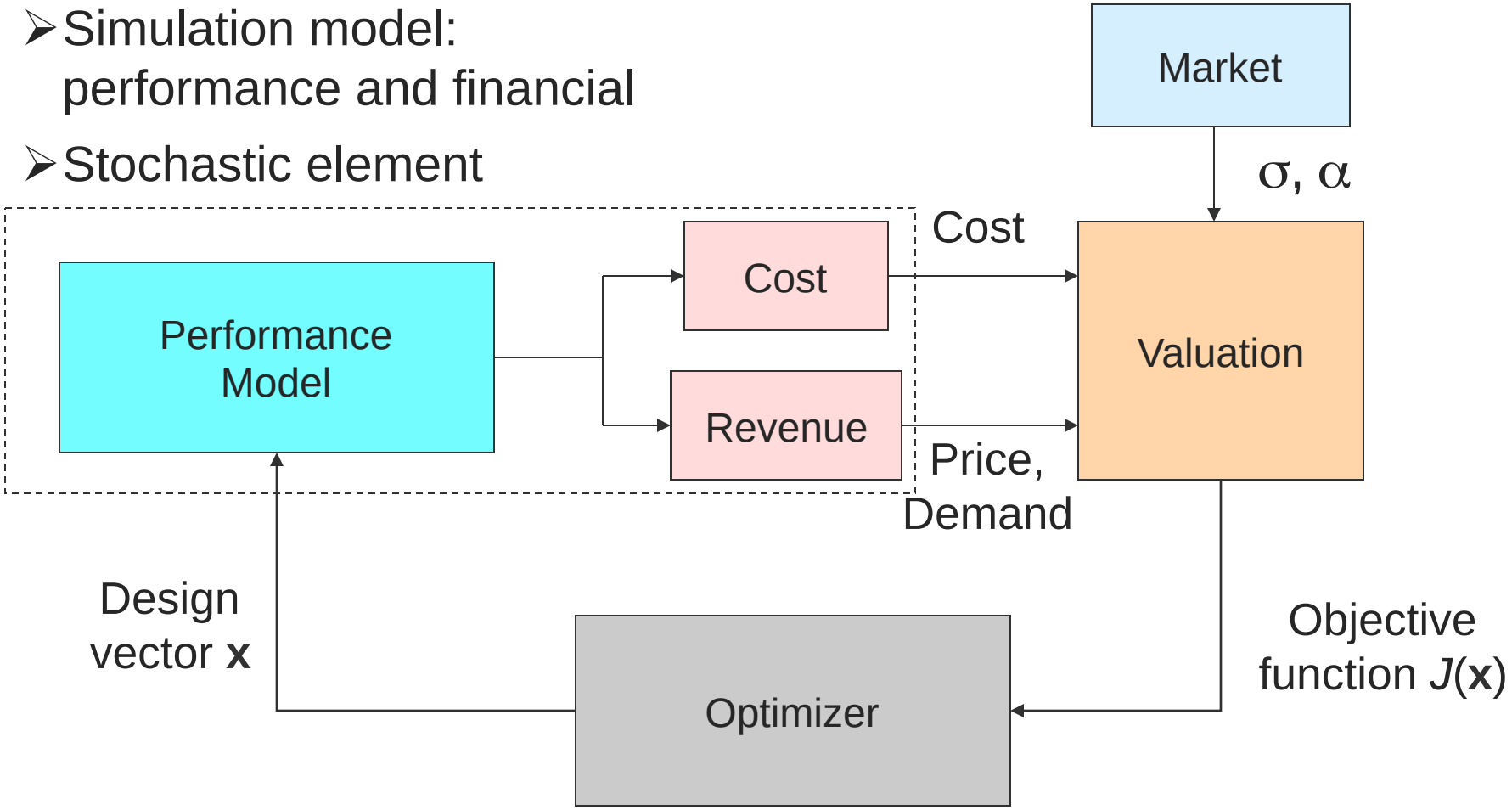
Traditional Design Optimization

- Objective function: usually minimum weight
- Design vector: attributes of design, e.g., planform geometry
- Performance model: contains several engineering disciplines



Coupled MDO Framework

- Objective function: value metric, e.g. NPV
- Simulation model: performance and financial
- Stochastic element



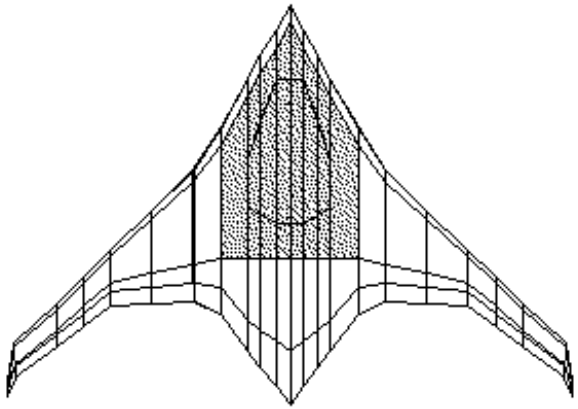
Value-Based Optimization Results

- Boeing BWB case study
  - 475 passengers, 7800 nmi range
  - Baseline: optimized for minimum GTOW
- Outcomes
  - Comparison of min GTOW and max E[NPV] designs
  - Traditional NPV vs. stochastic E[NPV]

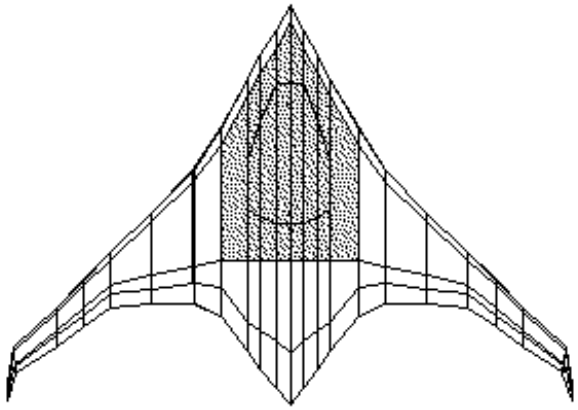
Different Objectives, Different Designs

- New objective results in tradeoff:
  - Lower structural weight, lower cost
  - Higher fuel burn, lower price
- Net result

  2.3% improvement in value
- Overall design very similar
  - Constrained to satisfy design requirements
  - Unable to move dramatically in design space



Minimum-GTOW planform



Maximum-value planform

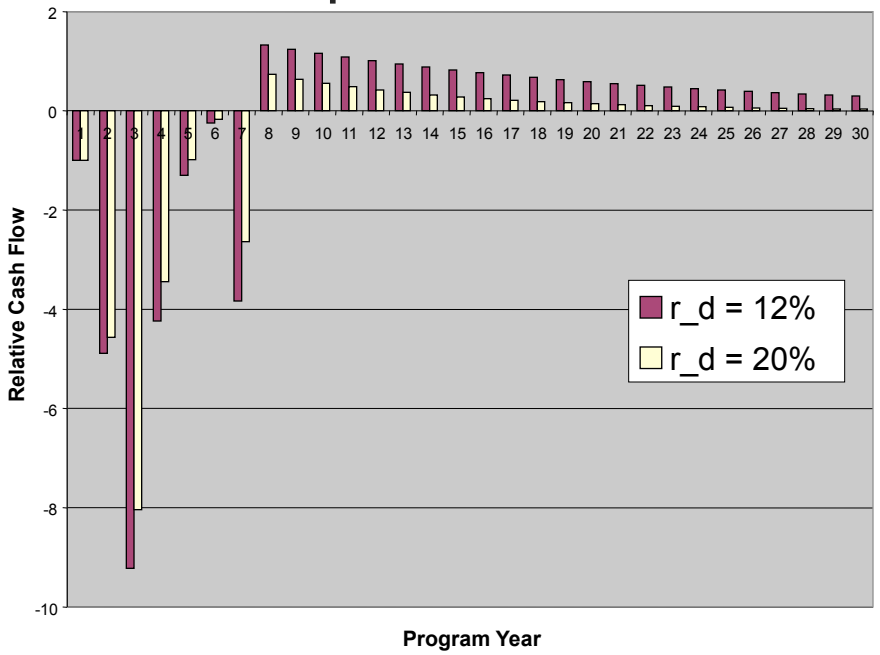
Deterministic vs. Stochastic Valuation

- Discount rates: 12% and 20%
  - Computational expense reduced, but NPV results are negative
  - E[NPV] for 12% design = 0.58% decrease
  - E[NPV] for 20% design = 3.7% decrease

  } relative to max-E[NPV] design

- High-r_d drives design to reduce development costs

- *Traditional NPV not appropriate*
  - *As valuation metric*
  - *As optimization objective*



Lecture Summary

- Designing for value is crucial: for a program to be successful, we cannot focus exclusively on performance
- The definition of value is flexible, and will vary depending on the application and on your interest as a stakeholder
- Financial metrics can be used to quantifying value, but use caution in your choice of value objective function
- Cost and revenue are difficult to model – often use empirical data
- It is important that uncertainty and risk are handled appropriately
- There is much work to be done on this issue

References

Brealey, R. & Myers, S., *Principles of Corporate Finance*, 7th Edition, McGraw-Hill, NY 2003

Markish, J., "Valuation Techniques for Commercial Aircraft Program Design", Masters Thesis, MIT Dept. of Aeronautics & Astronautics, June 2002.

Markish, J. and Willcox, K., "Value-Based Multidisciplinary Techniques for Commercial Aircraft System Design", *AIAA Journal*, Vol. 41, No. 10, October 2003, pp. 2004-12.

Peoples, R. and Willcox, K., "Value-Based Multidisciplinary Optimization for Commercial Aircraft Design and Business Risk Assessment," *Journal of Aircraft*, Vol. 43, No. 4, July-August, 2006, pp. 913-921.

Roskam, J., *Airplane Design Part VIII*, 1990.

Raymer, D., *Aircraft Design: A Conceptual Approach,* 3rd edition, 1999.

Schaufele, R., *The Elements of Aircraft Preliminary Design*, 2000.