



# Conception optimale pour l'ingénieur (Aerospace)

---

C2 by Prof. J. Morlier  
2025

# AU PROGRAMME

## Python based

lundi 31 mars 2025			
		09h15 - 12h45	MORLIER Joseph
		14h00 - 16h15	MORLIER Joseph
mardi 01 avril 2025			
		09h15 - 12h45	MORLIER Joseph
		14h00 - 16h15	MORLIER Joseph
mercredi 02 avril 2025			
		09h15 - 12h45	MORLIER Joseph MURADÁS ODRIOZOLA Daniel
		14h00 - 16h15	MAS COLOMER JOAN MURADÁS ODRIOZOLA Daniel
jeudi 03 avril 2025			
		09h15 - 12h45	MAS COLOMER JOAN MURADÁS ODRIOZOLA Daniel

Intro: Sustainable Aviation (Materials) With Both Eyes Open
Design optimization 1: constrained optimization, MOO, Sensibility with examples
Project DO 1 2 3

Topology Optimization with examples
Material ecoselection, Ashby Diagram and more

Projet DO 1 2 3
Wrap up and demo from students

Intro to MDAO Static Aeroelastic problem is a MDAO problem
Airbus PROJECT by TEAM of 3 (marked*)

vendredi 04 avril 2025		ORAL MARKED*	
		09h15 - 11h30	MORLIER Joseph MURADÁS ODRIOZOLA Daniel

# KEEP STUDENTS ACTIVE

## First Exercise



<https://mdolab.engin.umich.edu/assets/optimizationGame/>

SM305

49

# HOW?

## Example of Optimizers

### CHALLENGE #1 On Rosenbrock function minimization

<https://colab.research.google.com/drive/1rGHklpVM4Gqy9eq1Y10fiQW9umljeEx>

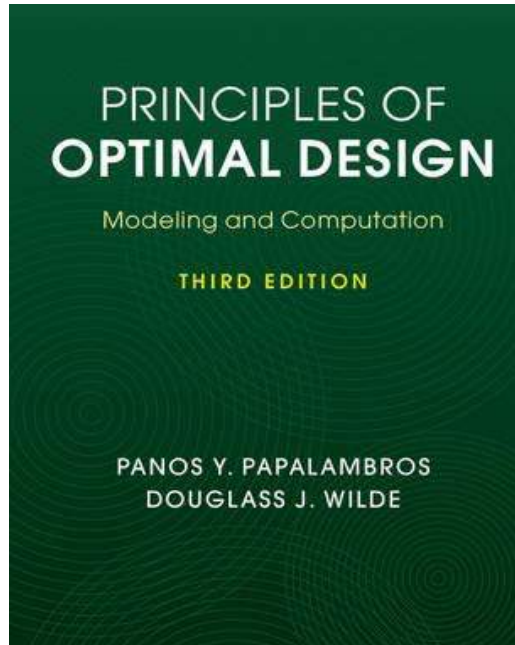
SM305

60

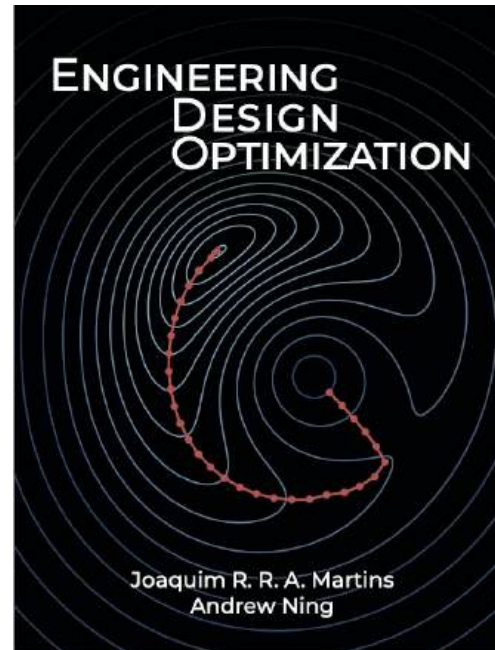
# Part1 : Constrained Optimization

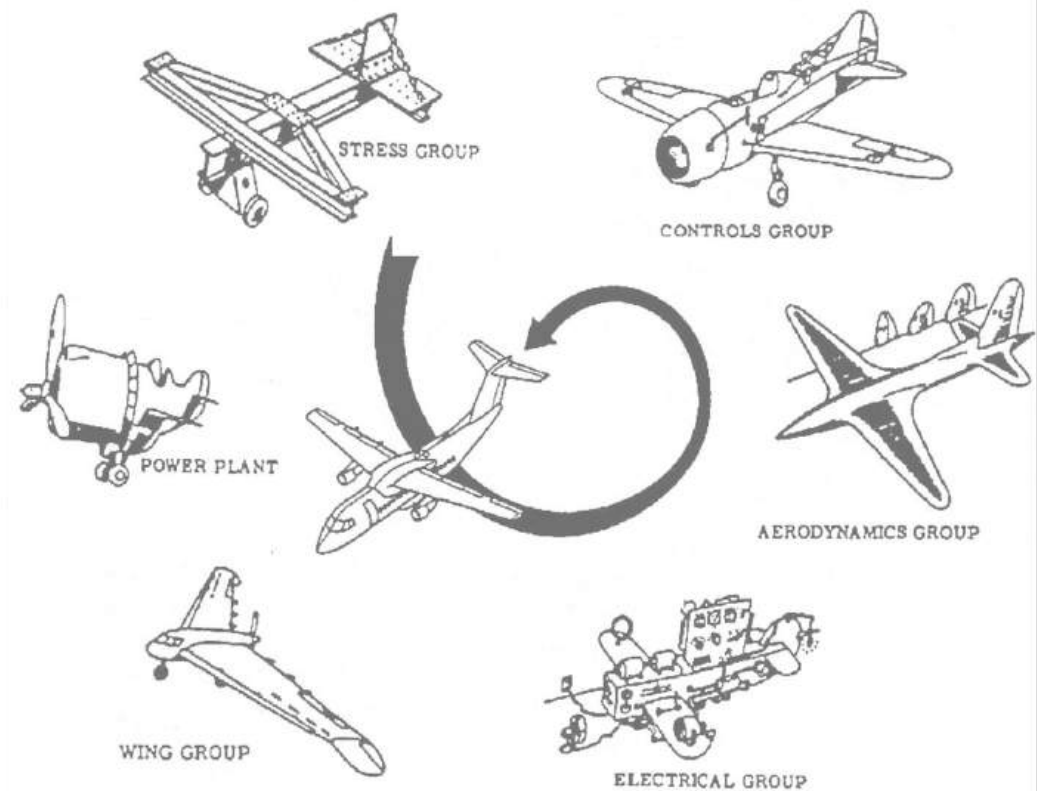
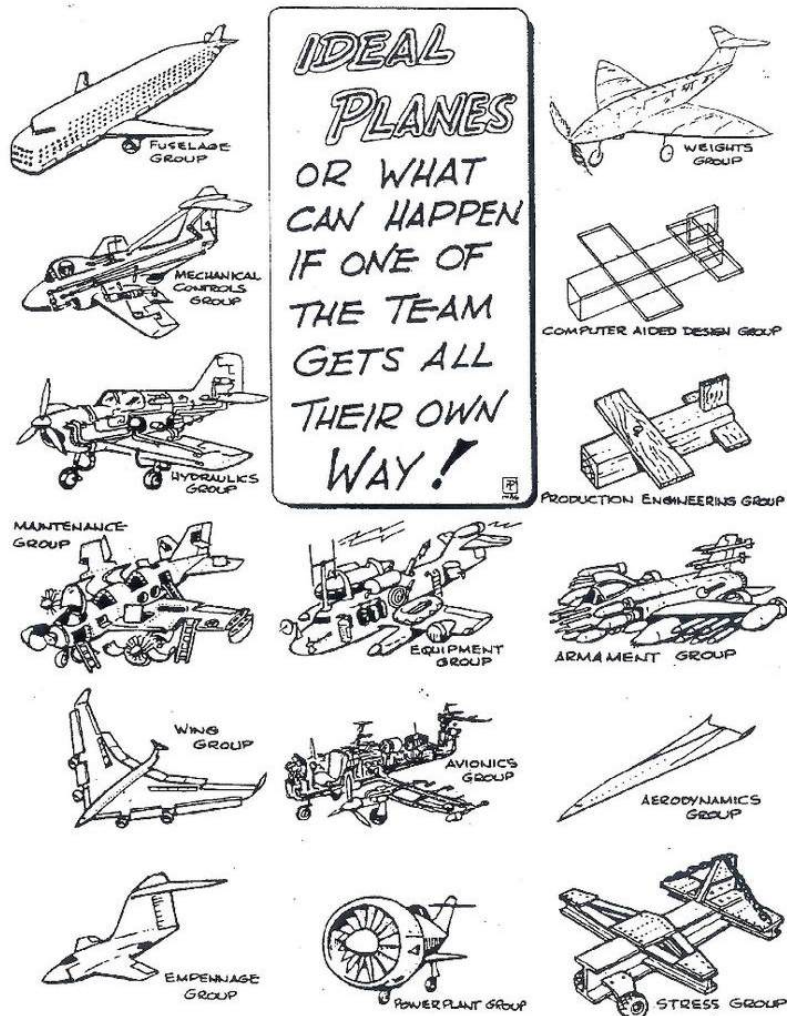
A recap of past courses (and more)?

# Good Starting Point ( $x_0$ )



38€ ebook

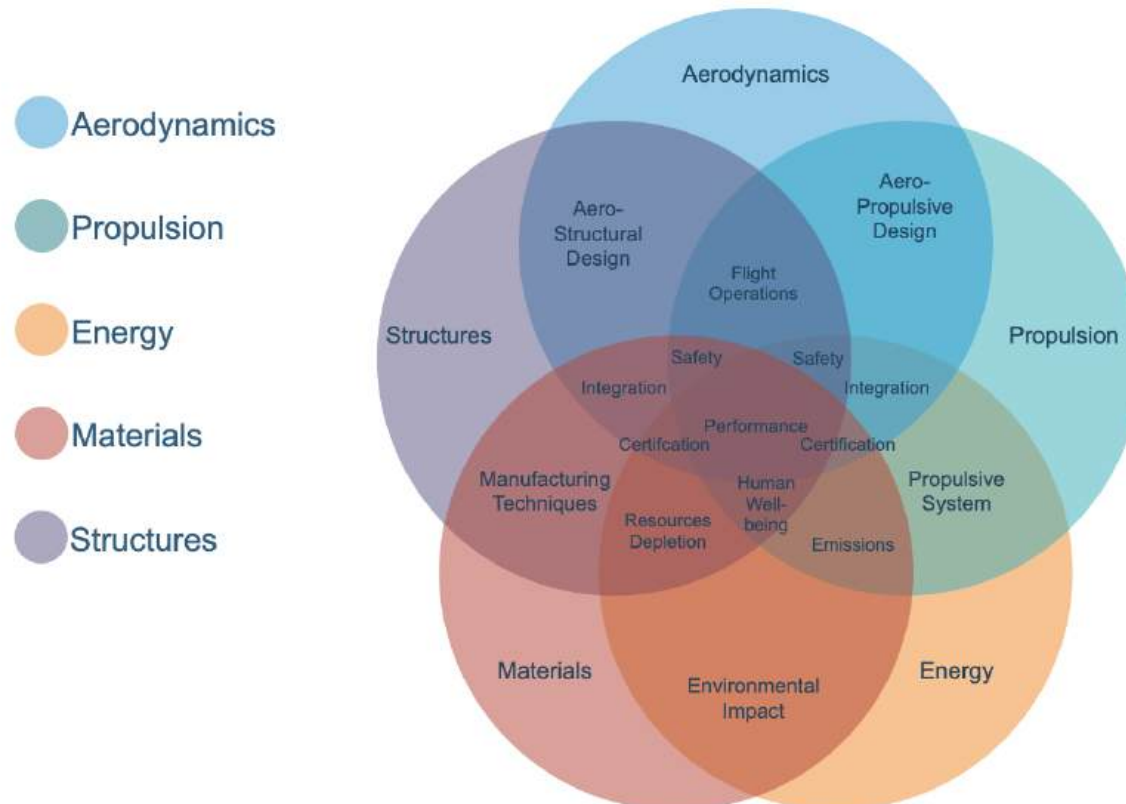




## Multidisciplinary Design Optimization (MDO)

# Venn Diagram

To illustrate the multidisciplinary nature of designing an aircraft and how the fundamentals disciplines are interlinked we can recur to a Venn diagram



F. Afonso, et al., "Strategies towards a more sustainable aviation: a systematic review", *Progress in Aerospace Sciences*, Vol. 137, 100878, 2023, <https://doi.org/10.1016/j.paerosci.2022.100878>

# A Curated list of (mostly) opensource softwares



# A good example

## Software

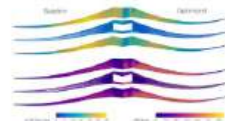
The software packages listed below are all distributed under open source licenses. These are research codes, so they require a strong background in programming and some persistence to get them to work. Unfortunately we are not able to provide support except for collaborators and sponsors. However, we strive to provide as much documentation as we can and continually work towards improving the usability.

**Webfoil:** This is an online tool for airfoil analysis and optimization. It also includes a vast database of airfoils. [\[Webfoil site\]](#) [\[Paper\]](#)



**pyOptSparse:** Interface to various optimization packages. pyOptSparse includes OptView, a visualization tool to explore the optimization history. [\[Code\]](#) [\[Documentation\]](#)

pyOptSparse



**TACS:** A general purpose structural finite-element code with adjoint derivatives that is developed by Prof. Graeme Kennedy. [\[Code\]](#) [\[Paper\]](#)

**SMT:** The surrogate modeling toolbox (SMT) is an open-source Python package consisting of libraries of surrogate modeling methods (e.g., radial basis functions, kriging), sampling methods, and benchmarking problems. SMT is designed to make it easy for developers to implement new surrogate models in a well-tested and well-document platform, and for users to have a library of surrogate modeling methods with which to use and compare methods. [\[Code\]](#) [\[Paper\]](#)

SMT

<https://mdolab.engin.umich.edu/software>

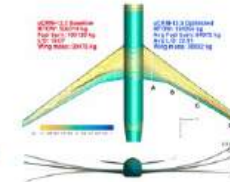
**ADflow:** (pronounced 'A-D-flow') CFD solver that can handle structured multi-block and overset meshes. It includes an adjoint solver for computing derivatives and can be used in the MACH-Aero framework for aerodynamic shape optimization. [\[Code\]](#) [\[Documentation\]](#) [\[Paper\]](#)



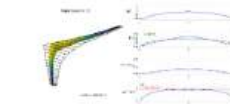
**DAFOAM:** (pronounced "dahfoam") is a suite of adjoint solvers for OpenFOAM that enable the computation of derivatives for aerodynamic shape optimization. [\[Code\]](#) [\[Documentation\]](#) [\[Paper\]](#)



**MACH-Aero:** A framework for aerodynamic design optimization that couples a CFD solver (e.g. ADflow or OpenFOAM), geometry parametrization (e.g. pyGeo), mesh deformation (e.g. IDWarp), and optimizer interface (pyOptSparse). [\[Code\]](#) [\[Documentation and Tutorials\]](#)



**OpenAeroStruct:** A lightweight aerostructural optimization code that can optimize a wing design in minutes on a laptop. [\[Code\]](#) [\[Documentation\]](#)



**OpenMDAO:** A framework for coupling multiple numerical models and performing multidisciplinary analysis and optimization. OpenMDAO is developed by NASA and uses numerical techniques developed in the MDO Lab. [\[OpenMDAO in a nutshell\]](#) [\[OpenMDAO site\]](#) [\[Paper\]](#)

openMDAO

# A Curated list of (mostly) opensource softwares

## Tools for decarbonizing air transportation:

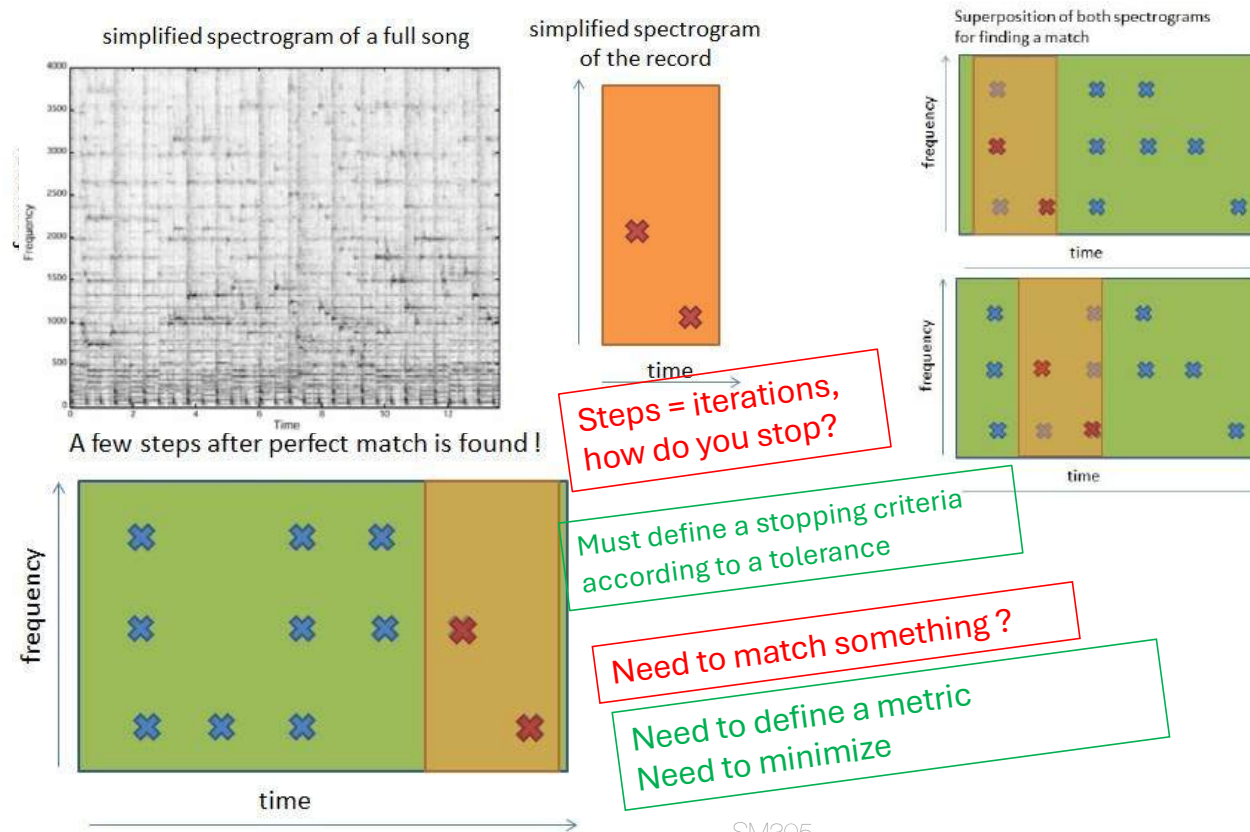
<https://cascade.boeing.com/>  
<https://aeromaps.eu>  
<https://github.com/AeroMAPS/AeroMAPS>  
<https://www.leadsresearchgroup.com/technology-dashboard>  
<https://github.com/contrailcirrus/pycontrails?tab=readme-ov-file>  
<https://github.com/sustainableaviation>  
<https://github.com/Aircraft-Operations-Lab>  
<https://psesh.github.io/aviation.html>  
<https://github.com/leadsgroup>  
<https://www.leadsresearchgroup.com/software>  
<https://github.com/prototypes/open-sustainable-technology>  
<https://www.witness4climate.org/optimizing-investments-in-energy-production-technology>  
<https://junzis.com/open-source>

## Tools for aircraft design

<https://github.com/peterdsharpe/AeroSandbox/tree/master>  
[https://openmdao.github.io/Aviary/examples/OAS\\_subsystem.html](https://openmdao.github.io/Aviary/examples/OAS_subsystem.html)  
<https://github.com/OpenMDAO/Aviary>  
<https://github.com/ideas-um/FAST>  
<https://github.com/MIT-LAE/TASOPT.jl>  
<https://github.com/mdolab/OpenAeroStruct>  
<https://github.com/mdolab/openconcept>  
<https://github.com/fast-aircraft-design/FAST-OAD>  
<https://web.mit.edu/drela/Public/web/>  
<https://lsdo.eng.ucsd.edu/software>  
<https://github.com/mid2SUPAERO/LCA4MDAO>  
<https://github.com/ImperialCollegeLondon/sharpy>  
<https://www.aircraftflightmechanics.com/NotesIntroduction.html>  
<https://github.com/MIT-LAE>  
<https://github.com/OpenVSP/OpenVSP>  
<https://github.com/suavecode>  
<https://github.com/camUrban/PteraSoftware>  
<https://github.com/cfsengineering/CEASIOMpy>  
<https://github.com/DLR-AE/PanelAero>  
<https://github.com/DLR-AE>  
<https://github.com/ImperialCollegeLondon/PinhoLab-WingBox>  
<https://commonresearchmodel.larc.nasa.gov>  
<https://github.com/facebookarchive/FBHALE>  
<https://github.com/mid2SUPAERO/ecoHALE>

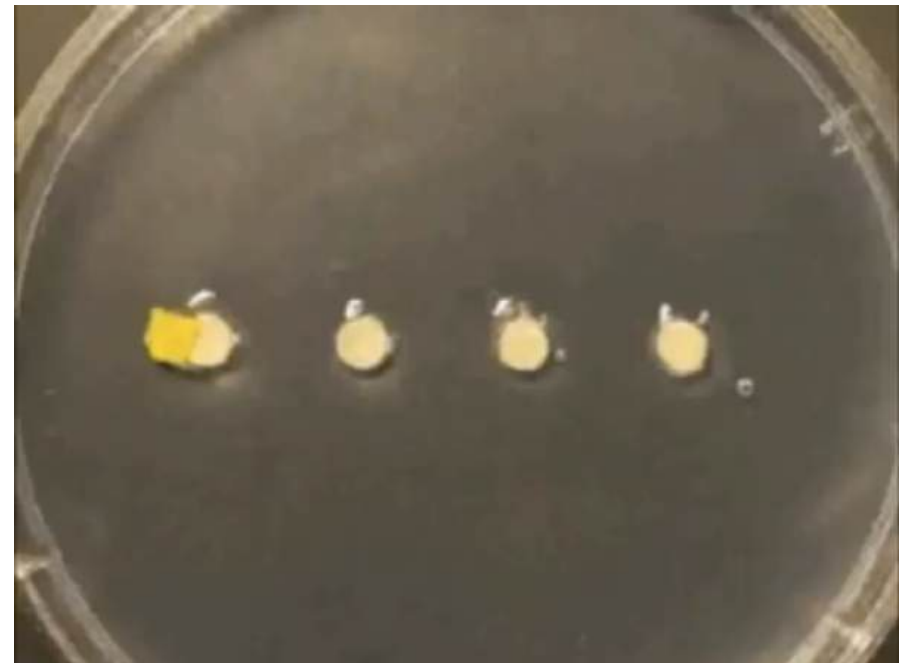
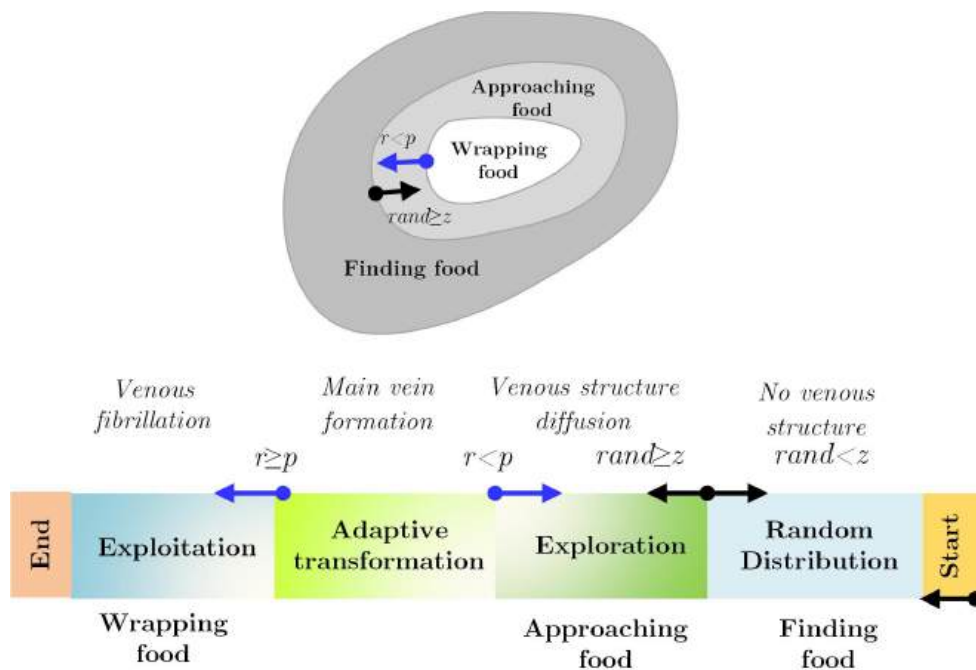
# Optimization is everywhere

<http://coding-geek.com/how-shazam-works/>

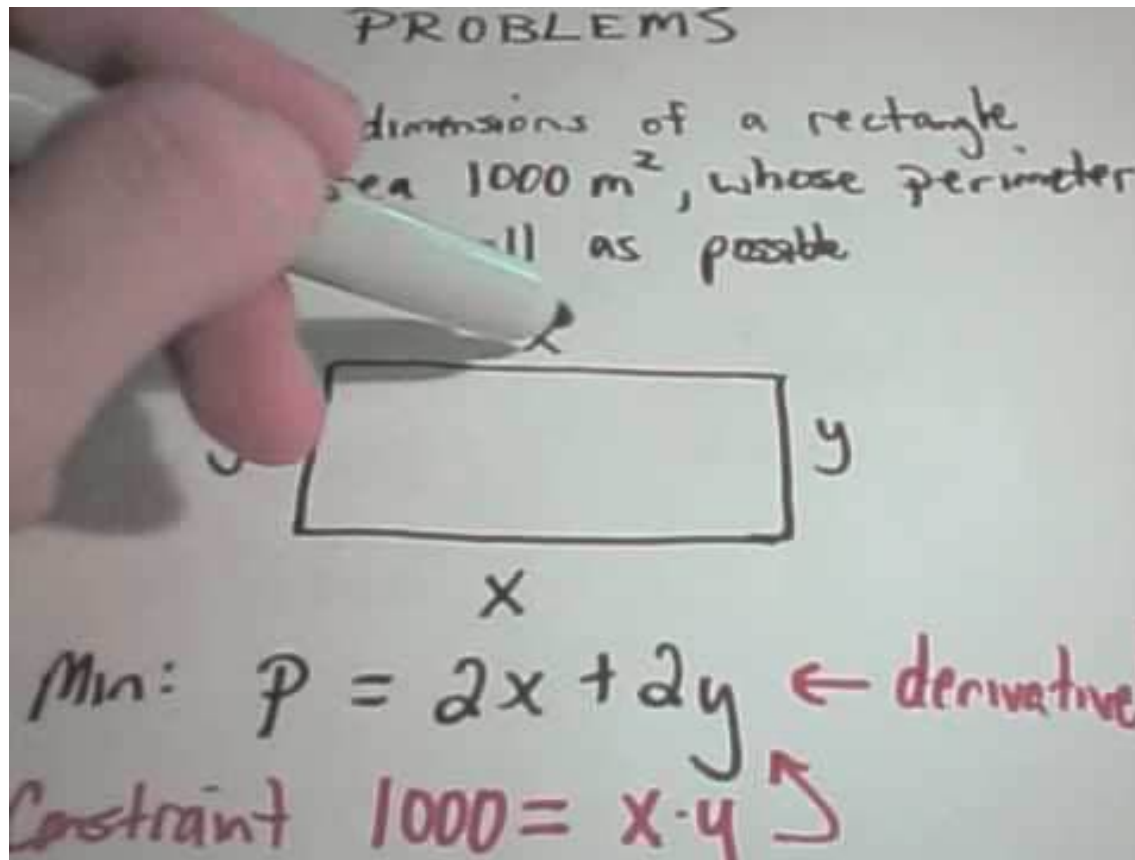


# Optimization is Everywhere

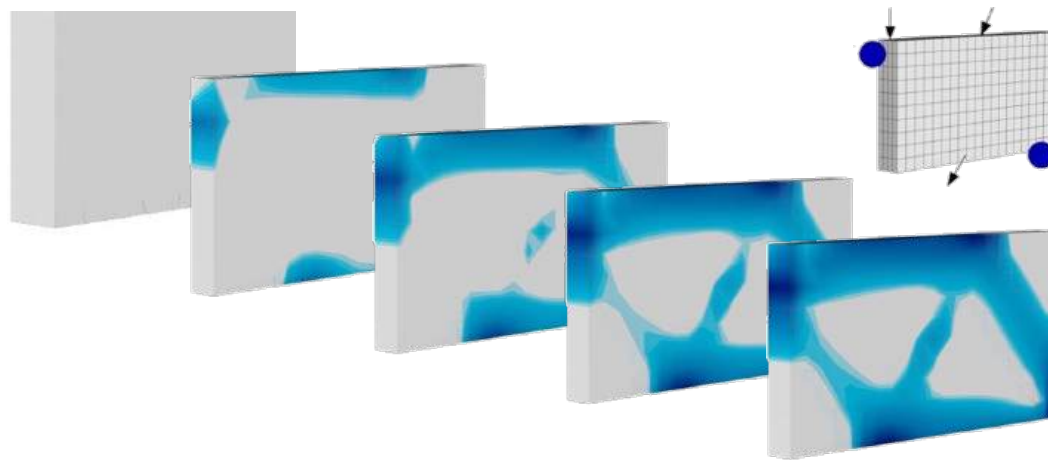
<https://www.aliasgharheidari.com//SMA.html>



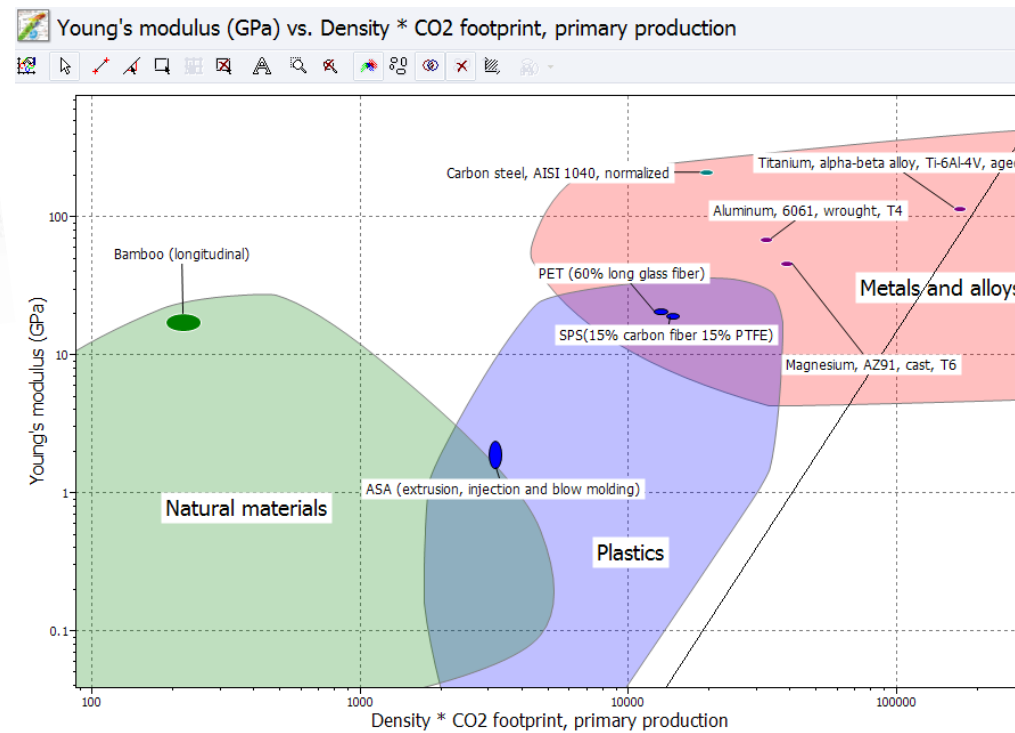
From engineering to maths (\*next course)



# design of bicycle frame by optimization



# design of bicycle frame by EcoDesign

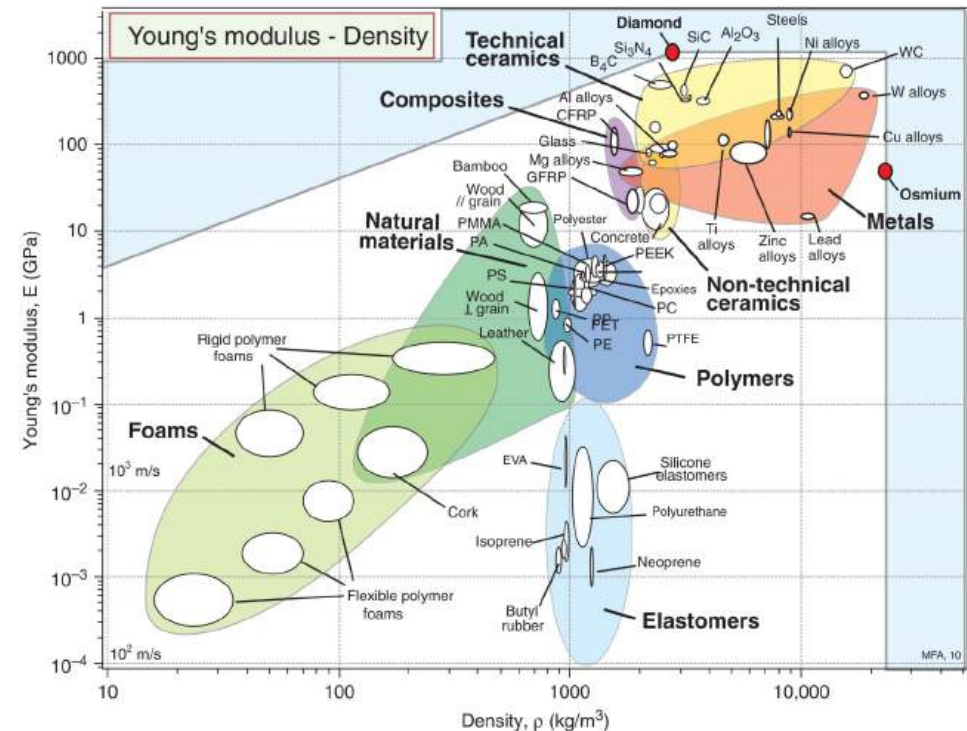




# Optimization is Everywhere

To get started, you need to select the right material. Why use steel if you can get away with aluminum?

There's a weight reduction of approximately two thirds on the table, albeit with compromises: reduced stiffness, higher cost, more difficulty in welding. A great way of visualizing these tradeoffs and rankings of mechanical properties is with Ashby Charts, which essentially represent the menu of materials that an engineer can select.



*Ashby Chart for evaluating stiffness-to-weight ratio of conventional materials. This essentially represents the menu of (conventional) materials that engineers have to choose from.*

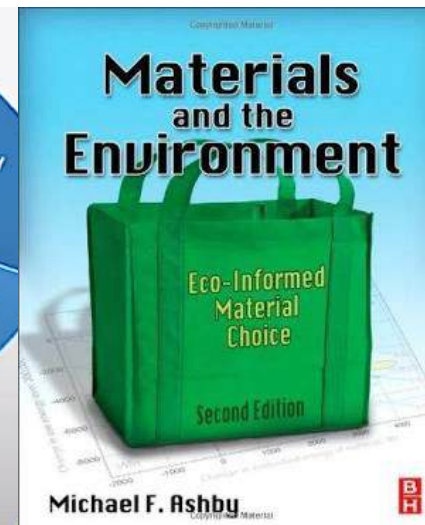
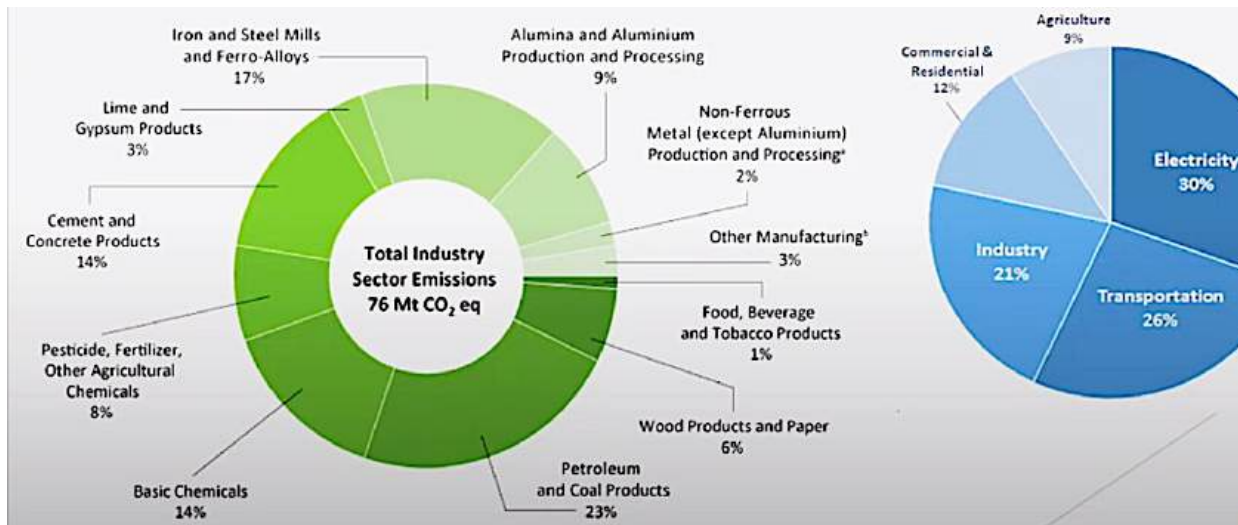


# How to ECOdesign tomorrow's structures?

#Structural materials used in a massive way → huge environmental impact

#The essential technologies for the transition, in particular green energy, will translate into considerable demand for metals that have become strategic.

#In anticipation of 2050, the total tonnage of concrete, steel, aluminum etc... necessary for the development of these energies will be 2 to 8 times the world production of 2010. !!!



**Ecoconception et matériaux**

Yves Bréchet

01 mars 2013 – 10:00 – 11:00 – Cours  
Amphithéâtre Guillaume Budé - Marcelin Berthelot

Diffusé avec le soutien de la  
Fondation Bertinore Schueller

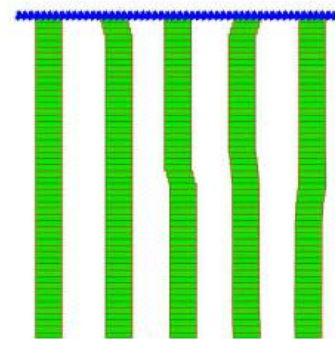
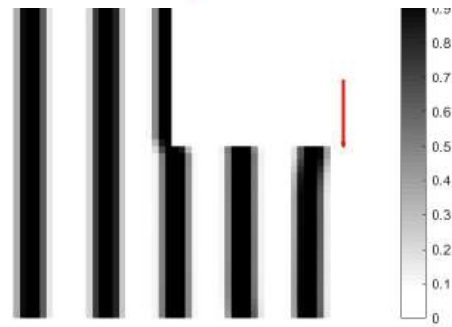
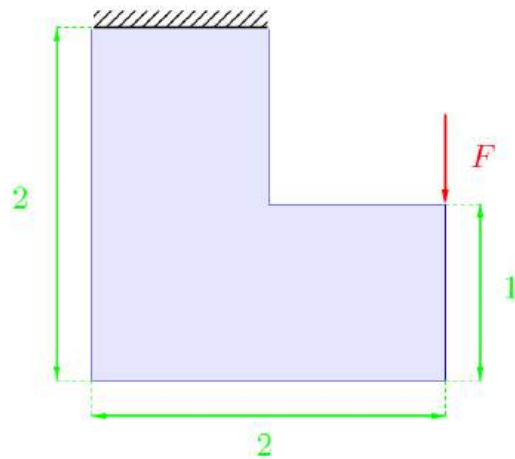
Le développement durable impose la prise en compte des impacts environnementaux dans l'usage des matériaux. Le cours illustrera des développements récents sur cette question en insistant sur la nécessité de considérer les matériaux dans un système, et non pas le matériau de façon isolé. Ce domaine,

# How to ECOdesign tomorrow's structures?

Prof. Joseph Morlier, Vilas Bhat\* (MAE 2019),

# Topology Optimization using Generalized Geometric Projection

# Additive Manufacturing in the CAD/CAE design loop

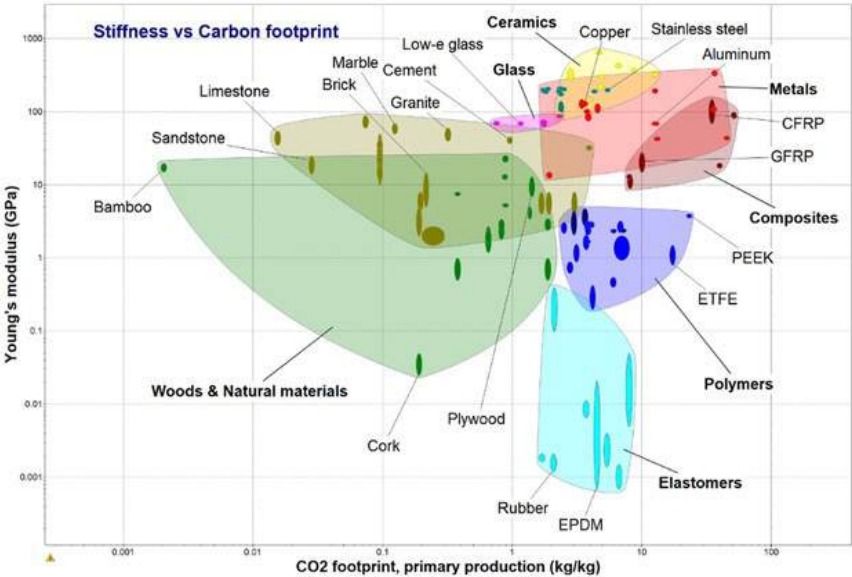


# How to ECOdesign tomorrow's structures?

Prof. Joseph Morlier, Edouard Duriez (X-SUPAERO 2018)

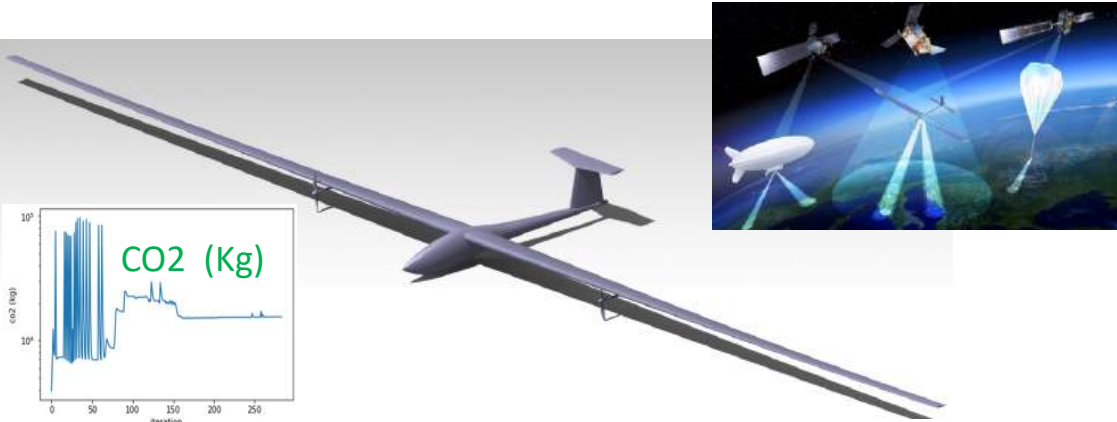
#Ecodesign

#MDO Trade-off between use phase (young's modulus and density) and production phase (CO2 footprint)



OPENAEROSTRUCT	Initial formulation	New formulation
Objective function	Fuel burn	CO2 impact of the drone
Constraints	<ul style="list-style-type: none"> <li>Mechanical failure</li> <li>Range by Breguet equation</li> <li>Flight (L=W)</li> </ul>	<ul style="list-style-type: none"> <li>Enough power</li> <li>Simple buckling constraint</li> <li>Mechanical failure</li> <li>Flight (L=W)</li> </ul>
Design variables	<ul style="list-style-type: none"> <li>Thicknesses</li> <li>Twist</li> <li>Thickness to chord ratio</li> </ul>	<ul style="list-style-type: none"> <li>Thicknesses</li> <li>Twist</li> <li>Thickness to chord ratio</li> <li>Density</li> </ul>

Jasa, J. P., Hwang, J. T., & Martins, J. R. R. A. (2018). Open-source coupled aerostructural optimization using Python. Structural and Multidisciplinary Optimization, 57(4), 1815–1827. doi: 10.1007/s00158-018-1912-8



[https://www.esa.int/Applications/Navigation/Crossing\\_drones\\_with\\_satellites\\_ESA\\_eyes\\_high-altitude\\_aerial\\_platforms](https://www.esa.int/Applications/Navigation/Crossing_drones_with_satellites_ESA_eyes_high-altitude_aerial_platforms)

## Optimization is Everywhere

<https://www.3dprintingmedia.network/tesla-shows-massive-generatively-designed-3d-printed-part-in-model-y-underbody/>



The current underbody part made of 70 different components

20

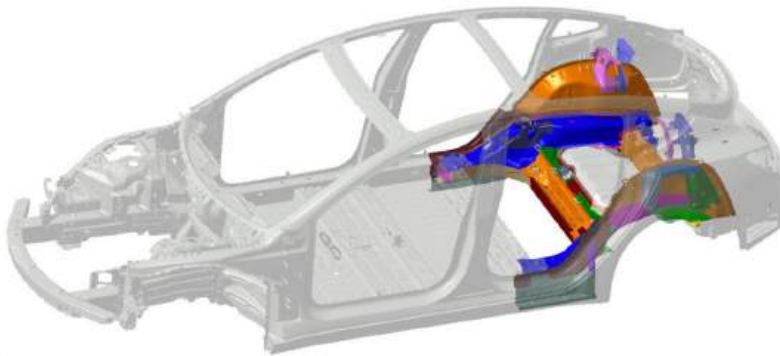


The generatively designed underbody, made of 2 and eventually 1 single piece.

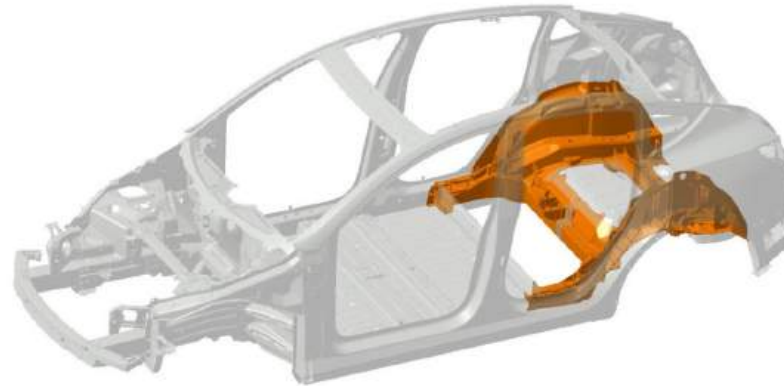
SM305

Optimization is Everywhere

Think different!!!



Model 3 rear underbody  
70 pieces of metal



Model Y rear underbody  
2 pieces of metal (eventually a single piece)

the use of 3D printing for sand casts such as that offered by voxeljet and ExOne for to enable the reduction of subassemblies (form 70 to 1) in a custom cast can bring about a significant transition even before metal AM can be used to produce such large metal parts directly. Producing a complex cast that can reduce the number of parts to this degree needs digital casting technology

Let's take the example of a structure with 10 structural elements (10 design variables).

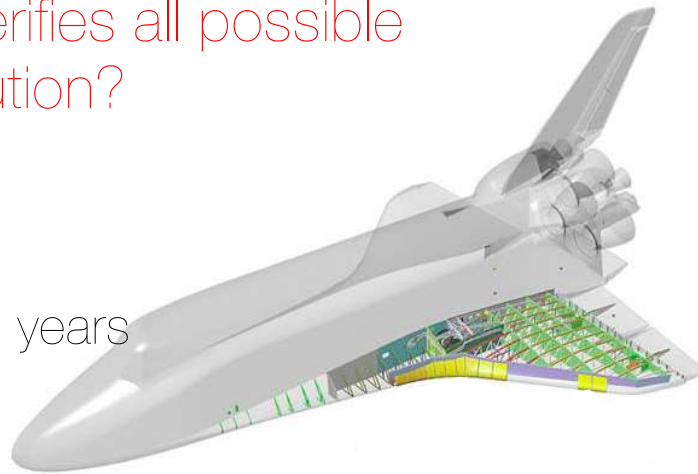
Each variable is associated with a section type (U, Z, I etc ...: 10 sections available per variable).

A structural analysis "costs" only 1 second of calculation.

*Question:*

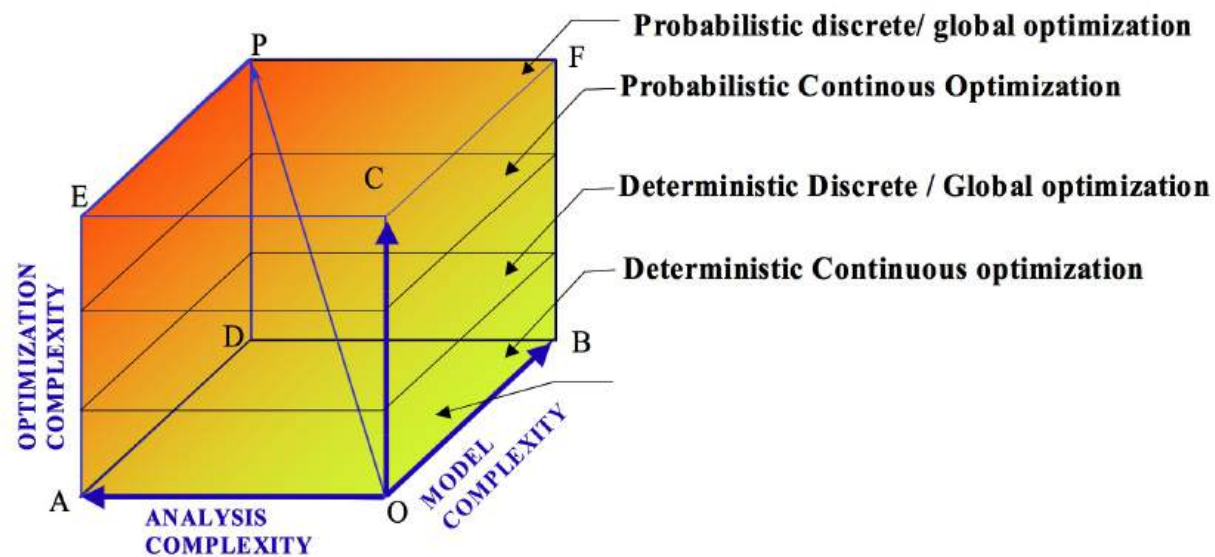
How long will the calculation last that verifies all possible combinations to ensure an optimal solution?

...  $10^{10}$  seconds At least 317 years

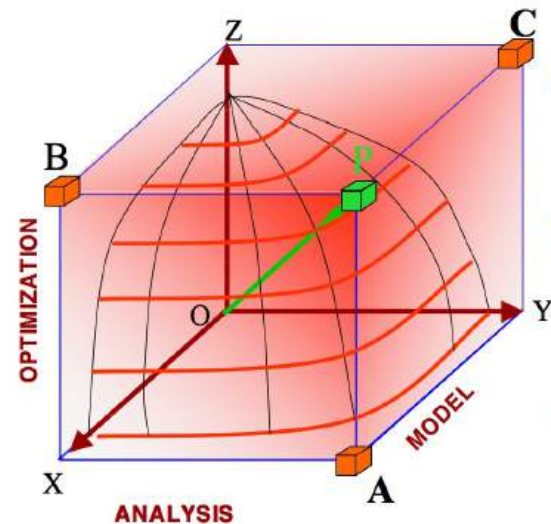




# Haftka's principle



# Haftka's Examples



## (A) Model and Analysis Complexity

- e.g., Complex 3-D structure requiring non-linear analysis

## (B) Model and Optimization Complexity

- e.g., Probabilistic design of a complex 3-D structure

## (C) Analysis and Optimization Complexity

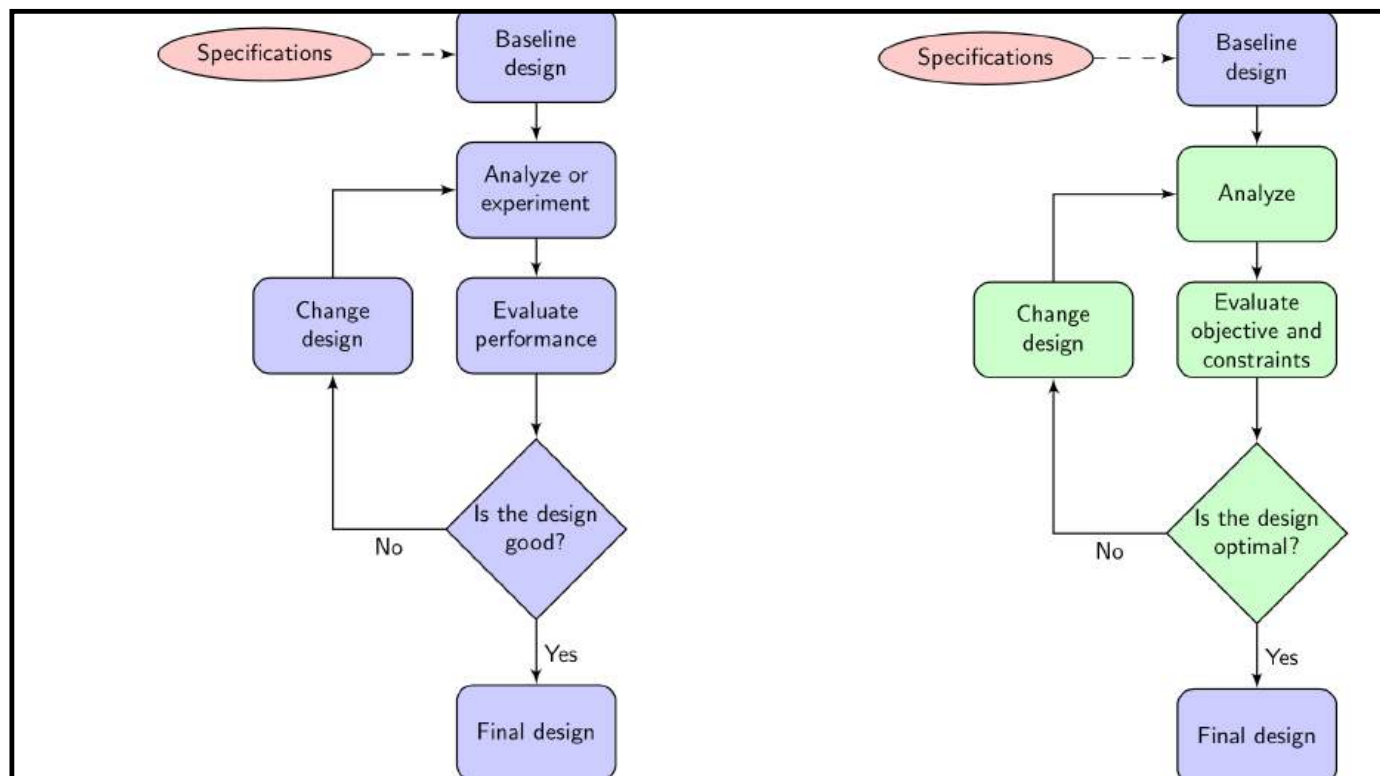
- e.g., Non-deterministic optimization with non-linear analysis (or combined macro and micro mechanical failure models)

## (P) Model, Analysis and Optimization Complexity

- e.g., Non-deterministic optimization of Complex 3-D structure requiring nonlinear analysis



## Conventional vs Optimal Design



# Definition ?

- “Making things better”
- “Generating more profit”
- “Determining the best”
- “Do more with less”

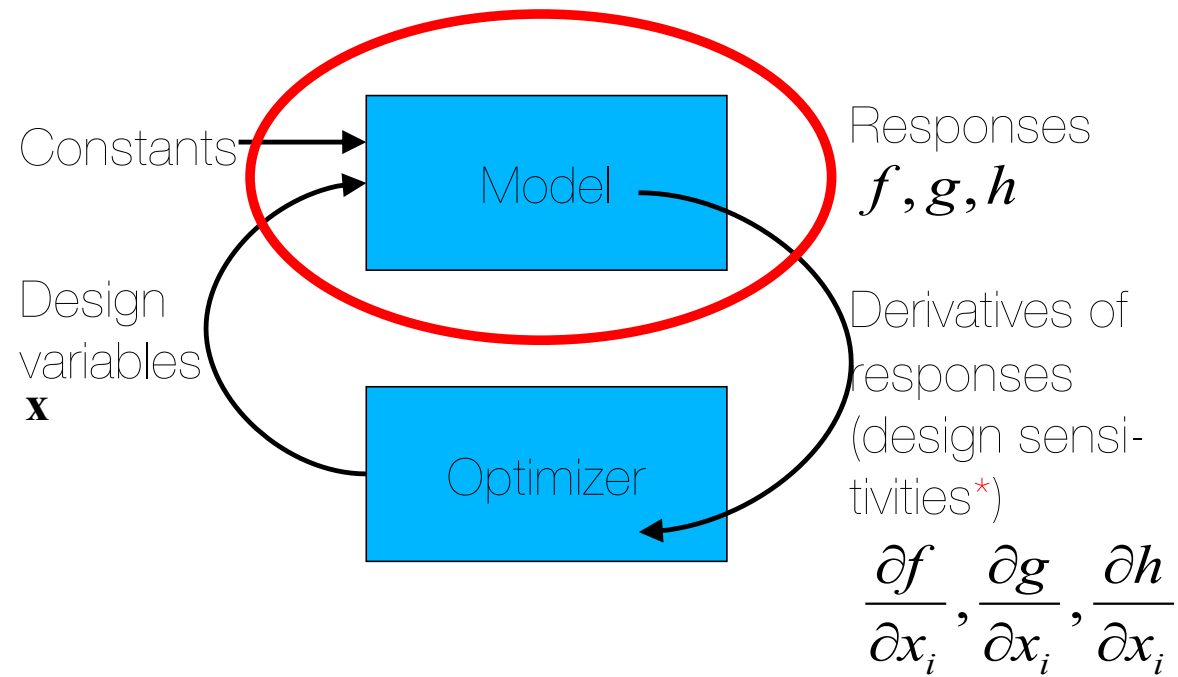


- “The determination of values for design variables which minimize (maximize) the objective, while satisfying all constraints” : Optimal design

# Engineer's goals?

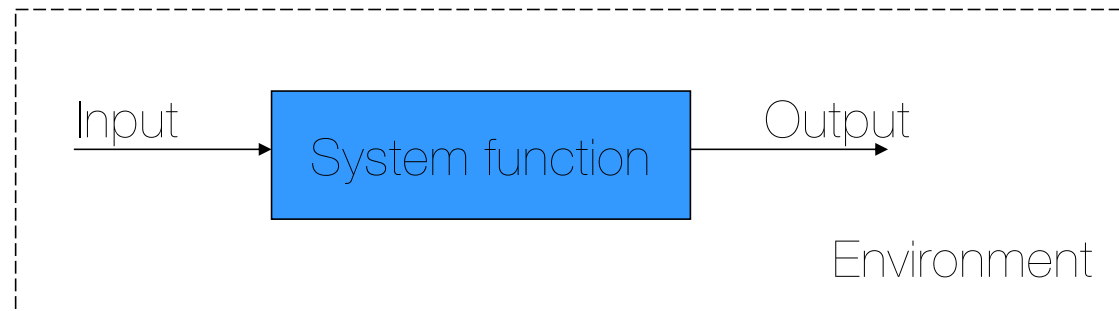
- The engineer must often answer a contradictory problem:
  - Airplane wing, F1 : rigidity vs mass
  - Confort in turbulence vs speed of flutter
  - Pressurized tank: use (form) / buckling loadEverything else is trivial!

# How?



\*see doc on LMS

# System approach



- Wonder:
  - What are the inputs / outputs?
  - What is the system / environment?
  - Hierarchize the system

# Optimization types

$$\min_{x \in \mathbb{N}} J(x)$$

Discrete:  
Operational research,  
combinatory, theory of  
graph

$$\min_{x \in D \subset \mathbb{R}^N} J(x)$$

Finite dimension:  
Optimality criteria,  
differentiability, convex  
analysis,  
constraints/unconstrained

$$\min_{u \in H} J(u)$$

Variational  
method:  
Physics,  
mechanics,  
economy

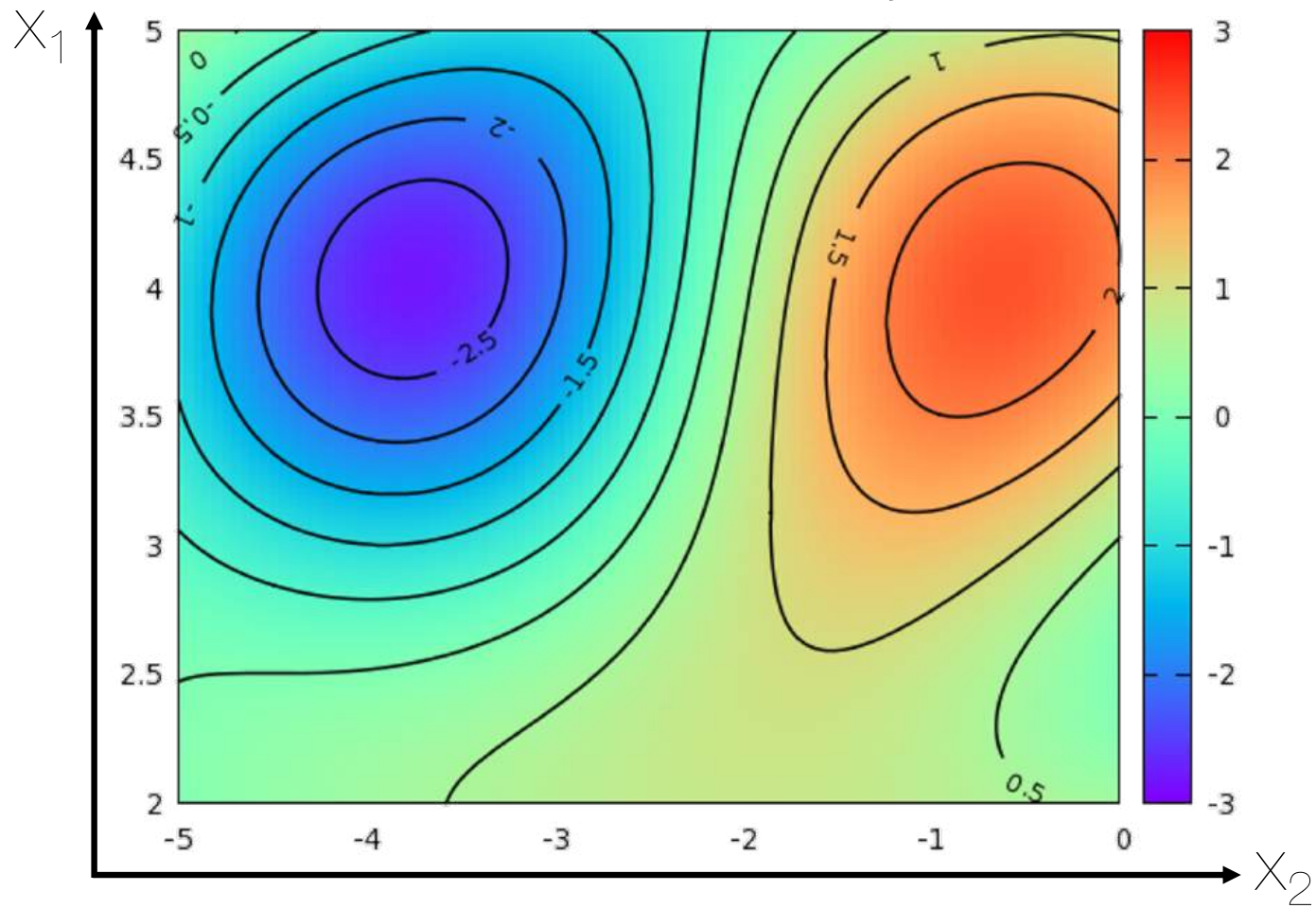
$$\min_{\Omega \subset \mathbb{R}^N} J(\Omega)$$

Shape:  
Properties, optimality  
criteria

<https://neos-guide.org/content/optimization-tree-alphabetical>

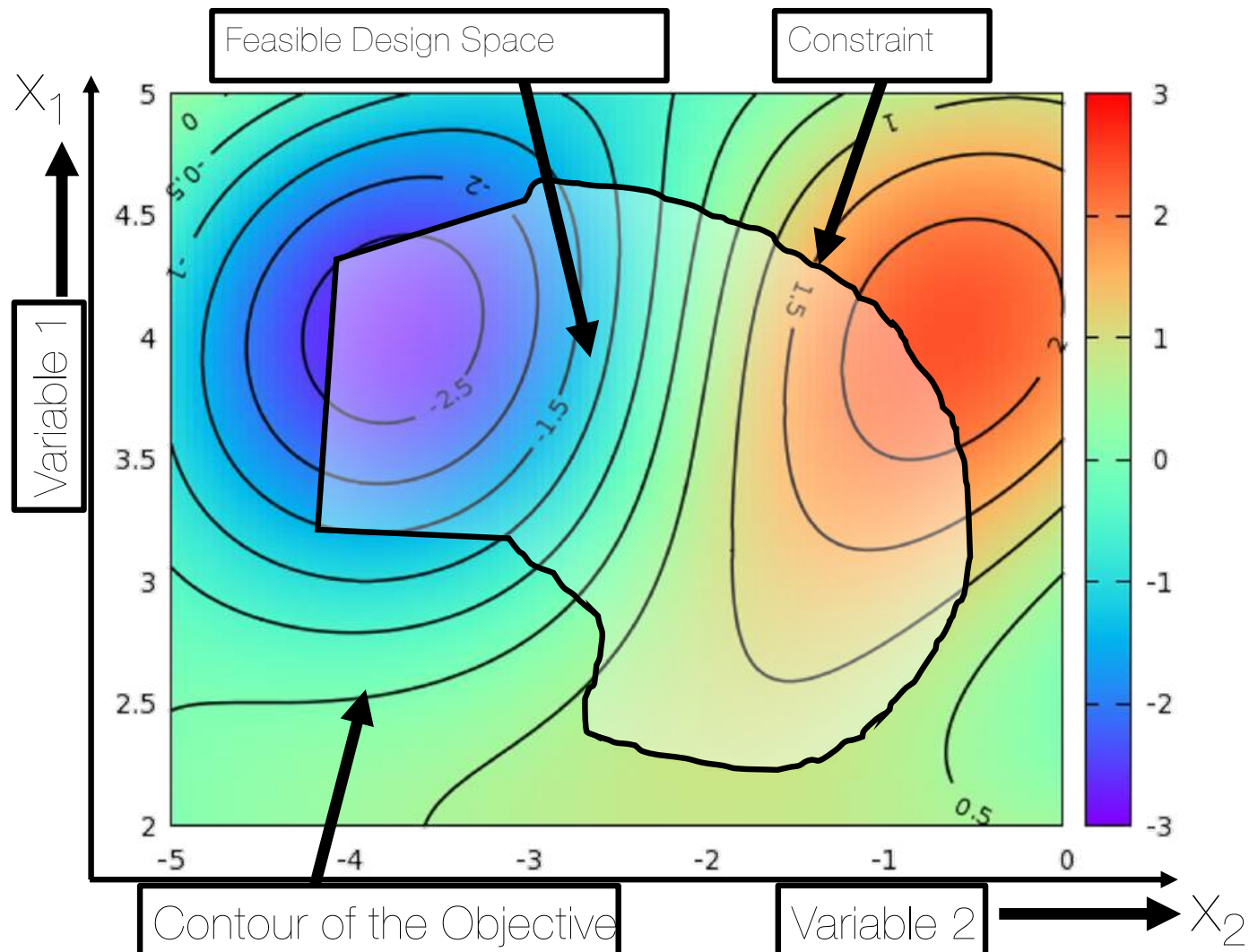
The BIG picture

Variables and the Objective



The BIG picture

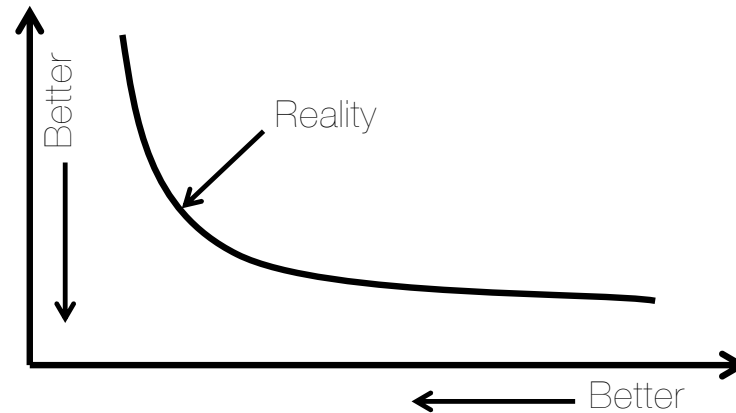
## Variables, Objective & Constraints



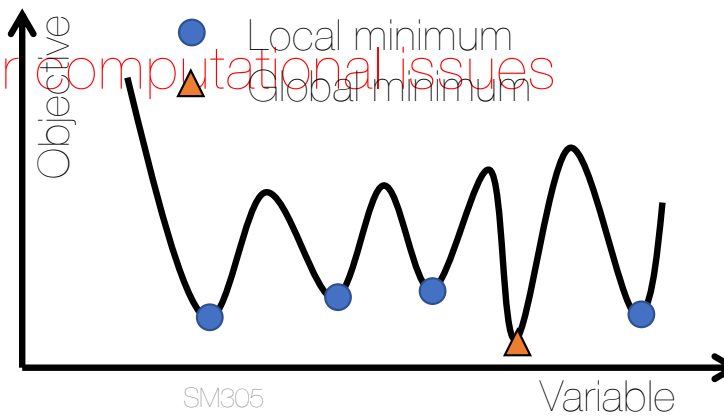


# Common issues in the optimization of engineering systems

Multiple objectives



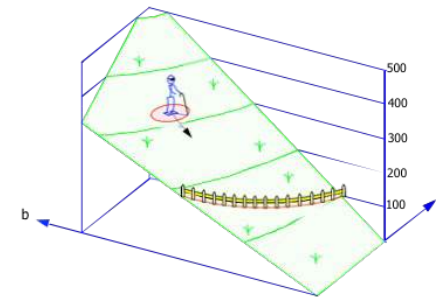
Multiple local solutions and other computational issues



# Gradient based Optimization (Maximize/Minimize)



The optimum (the solution) will depend on the starting point !!!  
And this point of arrival may be a local minimum and not an overall minimum.



**Transformation des programmes de minimisation :** tout programme de minimisation peut aisément être transformé en programme de maximisation en remplaçant la fonction objectif  $f$  par son opposée  $-f$  :

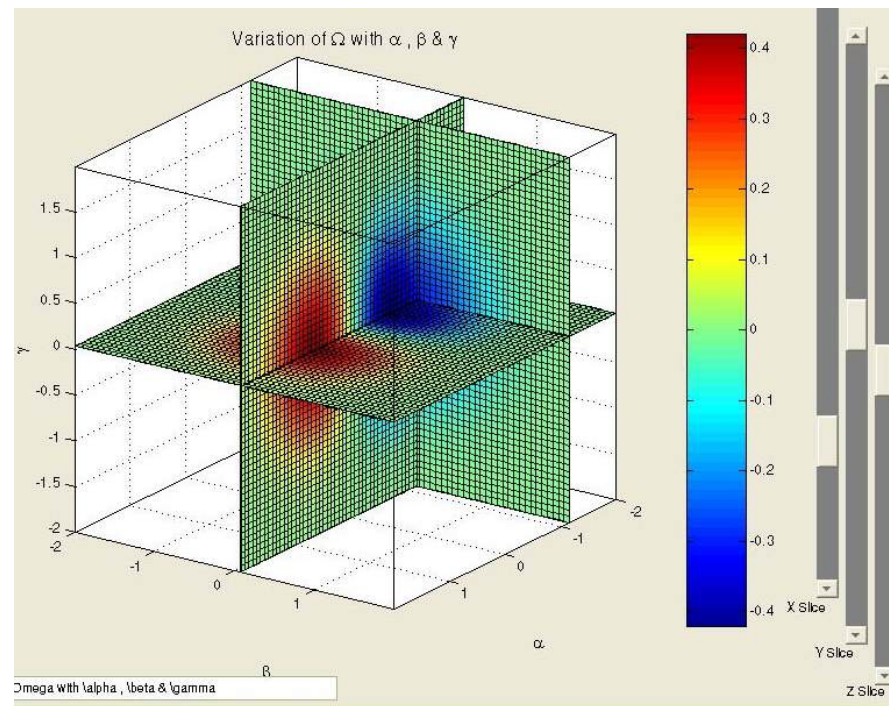
$$\left| \begin{array}{ll} \min_{\tilde{x}} & f(\tilde{x}) \\ \text{s.c.} & \text{contraintes} \end{array} \right| \Leftrightarrow \left| \begin{array}{ll} \max_{\tilde{x}} & -f(\tilde{x}) \\ \text{s.c.} & \text{contraintes} \end{array} \right|$$

# In N dimension?

Of course in N-D,

N Design variables

P constraints, it may be difficult to visualize...



# Derivatives ?

In optimization

$$\min_{x \in X} J(x)$$

Quasi-Newton method  
Gradient descent  
Gauss–Newton algorithm  
Levenberg–Marquardt  
algorithm  
Trust region  
Nelder–Mead method  
MATLAB, SOL200 etc...

Why computing derivative of criteria J ?

- Optimality criteria  $J'(x)=0$
- To calculate an approximated solution:
- Steepest descent  $X_{n+1} = X_n - \text{rho} * J'(X_n)$
- Newton  $X_{n+1} = X_n - [D^2J(X_n)]^{-1} J'(X_n)$

# Negative null form [MATLAB]

$$\begin{array}{ll} \underset{\mathbf{x}}{\text{Minimize}} & f(\mathbf{x}) \\ \text{subject to} & \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \\ & \mathbf{x} \in \mathcal{X} \subseteq \mathbb{R}^n \end{array} \quad \begin{array}{l} \mathbf{x} = \text{(column) vector of design variables} \\ \mathbf{h} = [h_1, h_2, \dots, h_{m_h}]^T \\ \mathbf{g} = [g_1, g_2, \dots, g_{m_g}]^T \end{array}$$

Other formulations:

- positive null form ( $\mathbf{g}(\mathbf{x}) \geq \mathbf{0}$ )
- negative unity form ( $\mathbf{g}(\mathbf{x}) \leq \mathbf{1}$ )
- positive unity form ( $\mathbf{g}(\mathbf{x}) \geq \mathbf{1}$ )

Cf cours 1A SUPAERO

CONDITIONS KKT

# Criteria to optimize (performance)

- Some examples of criteria to minimize (or maximize)
  - Cost
  - Mass
  - Aerodynamic drag
  - Lift
  - Etc.



# Design variables

- The section of a beam in an assembly (lattice)
- The number of ribs in a wingbox
- The skin thickness of a wingbox
- The orientation and sequencing of a composite
- The coordinates defining a NACA profile
- Etc ...

In optimization of structures, we distinguish the variables of shape, geometry, and materials

# Constraints

## Mechanical

- Von Mises (Stresses)
- Buckling load
- Eigenfrequencies

$$g = \frac{\sigma_{\max}(\mathbf{x})}{\sigma_{\text{allowed}}} - 1 \leq 0$$

Scaled (Reserve Factor)

$$g = \sigma_{\max}(\mathbf{x}) - \sigma_{\text{allowed}} \leq 0$$

vs.

Unscaled





# Optimization Steps

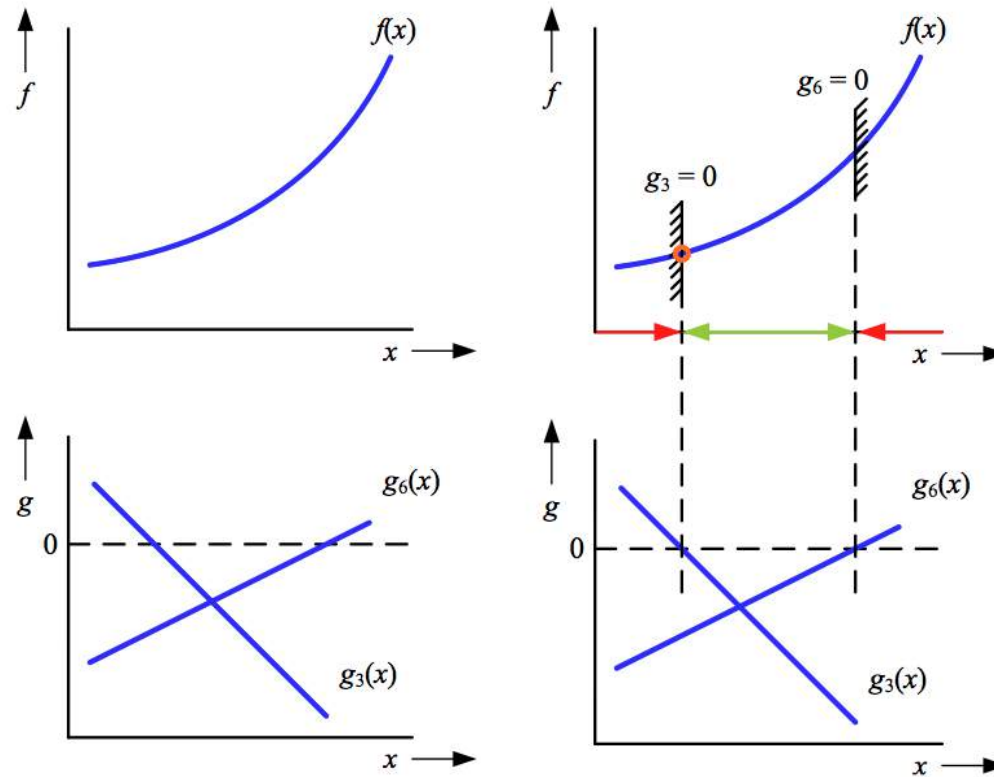
- 1 Select design variables
- 2 Select objective criterion in terms of design variables (to minimize or maximize)
- 3 Determine constraints in terms of design variables, which must be satisfied
- 4 Determine design variable values which minimize (maximize) the objective while satisfying all constraints

[Papalambros & Wilde 2000: Principles of optimal design]

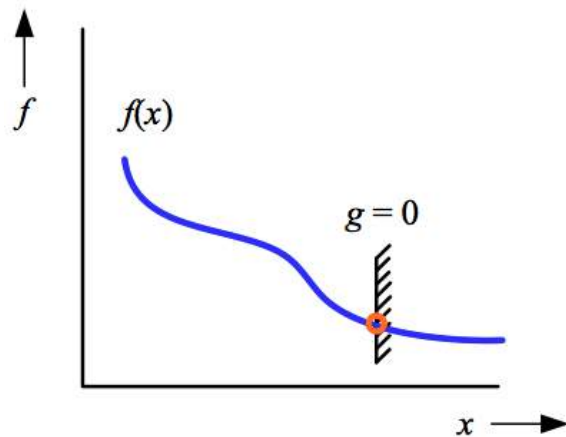
# Classification

- Problems:
  - Constrained vs. unconstrained
  - Single level vs. multilevel
  - Single objective vs. multi-objective
  - Deterministic vs. stochastic
- Responses:
  - Linear vs. nonlinear
  - Convex vs. nonconvex (later!)
  - Smooth vs. nonsmooth
- Variables:
  - Continuous vs. discrete (integer)

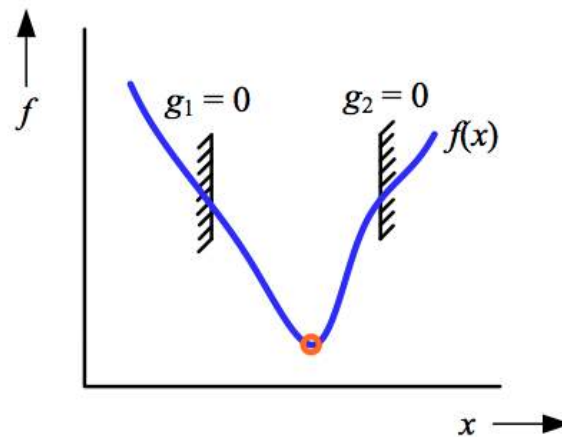
# Visualization: 1D example



# Constrained VS unconstrained optimum

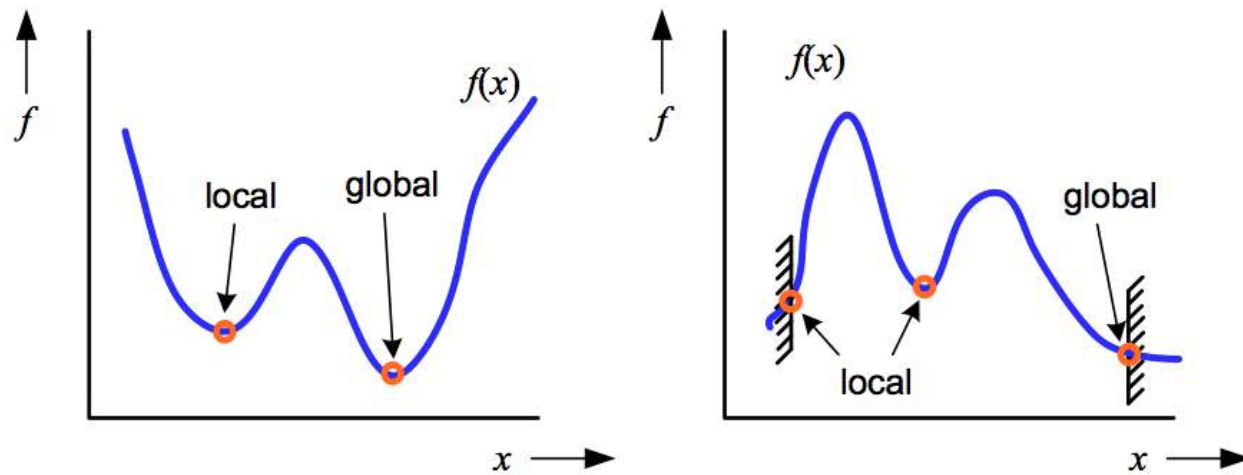


- constrained optimum
- bounded optimum

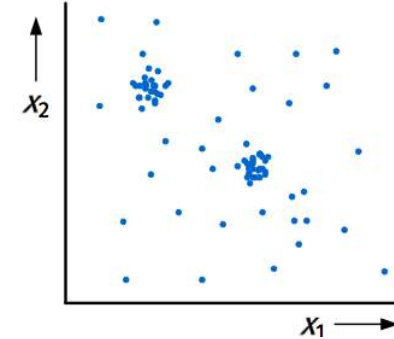


- unconstrained optimum
- interior optimum

# Multimodality (multiple local minima)



# Genetic Algorithms GA



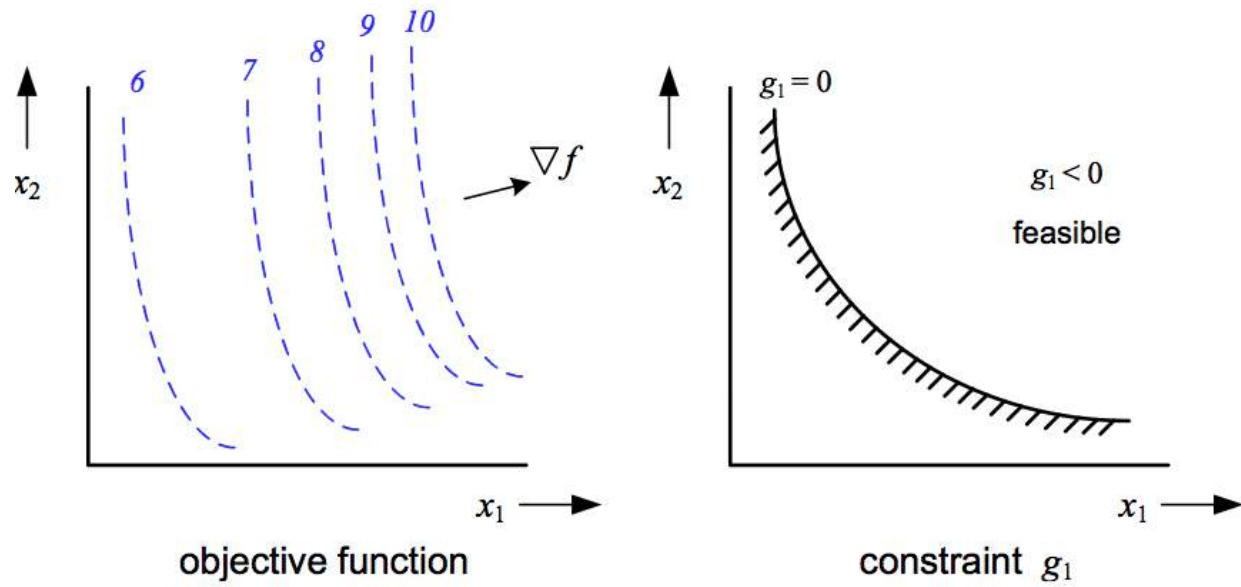
- The first "metaheuristic": 1975, John Holland publishes "Adaptation in natural and artificial systems »
- Widely used, easy to program, easy to use and robust.

*1 / Problems of continuous GLOBAL optimization*

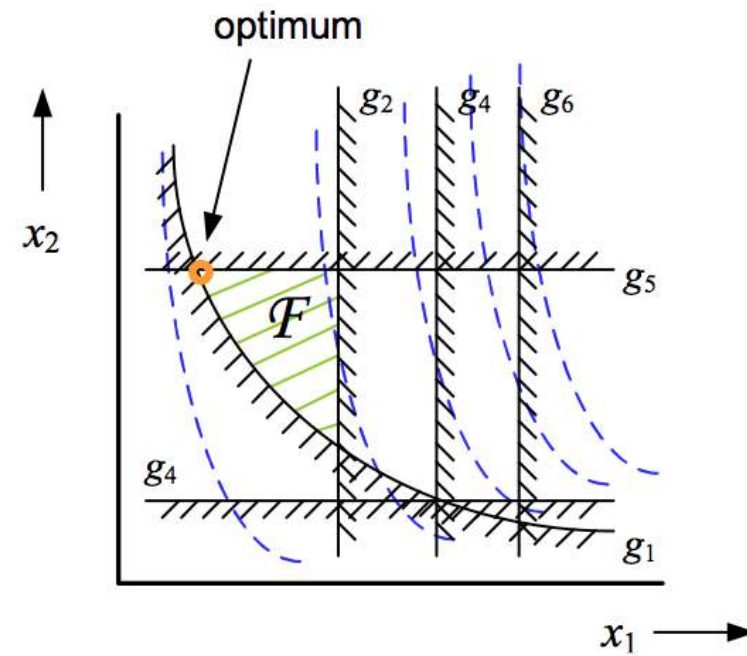
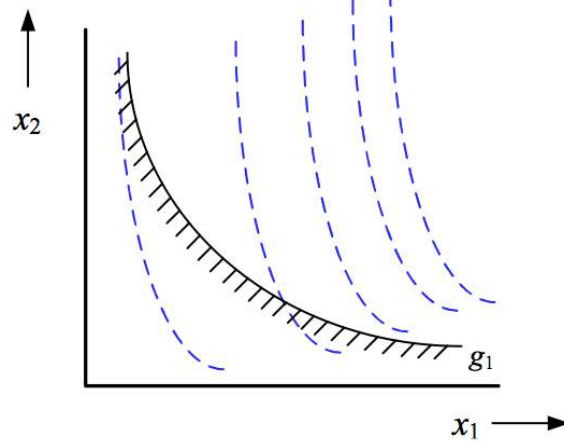
*2 / Problems of combinatorial (discrete) GLOBAL optimization:*

→ For these problems, either discrete values can only be used for the variables, or we must change elements of the problem to "change the values of the variables"

# 2D example



# 2D example





# First Exercise

## Optimization Game

« Back

Do you think you can beat a gradient-based optimizer to the minimum of an unknown function? Try it out [here!](#)

Find the minimum by clicking in the domain to the right.

If you are experiencing display issues, please try Chrome or Firefox.

Current position: (4.7, 1.7)

Function value: 22921.4

Clicks used: 25/25

Best score:

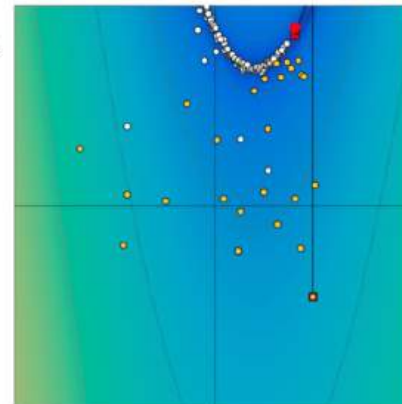
☐ Easy ☐ Intermediate ☒ Hard

User ☐ Gradient-free Gradient-based

Gradient-based path Minimum

Number of function evaluations per method:

User	Gradient-free	Gradient-based
DNF	122	18



<https://mdolab.engin.umich.edu/assets/optimizationGame/>

# Stopping criteria

## Condition on optimality

$$\|\nabla L(\mathbf{x}_{k+1})\| < \varepsilon$$

(Matlab: TolFun)

$$\mathbf{g}_j(\mathbf{x}_{k+1}) < \varepsilon \quad \text{and} \quad |h_j(\mathbf{x}_{k+1})| < \varepsilon$$

(Matlab: TolCon)

## or a condition on change in $\mathbf{x}$

$$\|\mathbf{x}_{k+1} - \mathbf{x}_k\| < \varepsilon$$

(Matlab: TolX)

## or a condition on the number of iterations

$$k \leq k_{\max}$$

(Matlab: MaxIter)

## or a combination of these, with $\varepsilon > 0$

Type	Domaine	Function
Scalar Minimization	$\min_a f(a)$ such that $a_1 < a < a_2$	fminbnd
Unconstrained Minimization	$\min_x f(x)$	fminunc fminsearch
Linear Programming	$\min_x f^T(x)$ such that $Ax \leq b, Aeq.x = beq, lb \leq x \leq ub$	linprog
Quadratic Programming	$\min_x \frac{1}{2}x^T Hx + f^T x$ such that $Ax \leq b, Aeq.x = beq, lb \leq x \leq ub$	quadprog
Constrained Minimization	$\min_x f(x)$ such that $c(x) \leq 0, ceq(x) = 0$ $Ax \leq b, Aeq.x = beq, lb \leq x \leq ub$	fmincon
Goal Attainment	$\min_{x,\gamma} \gamma$ such that $F(x) - \omega\gamma \leq goal$ $c(x) \leq 0, ceq(x) = 0$ $Ax \leq b, Aeq.x = beq, lb \leq x \leq ub$	fgoalattain
MiniMax	$\min_x \max_{\{F_i\}} \{F_i(x)\}$ such that $c(x) \leq 0, ceq(x) = 0$ $Ax \leq b, Aeq.x = beq, lb \leq x \leq ub$	fminimax
Semi-Infinite Minimization	$\min_x f(x)$ such that $K(x, \omega) \leq 0 \forall \omega$ $c(x) \leq 0, ceq(x) = 0$ $Ax \leq b, Aeq.x = beq, lb \leq x \leq ub$	fseminf

# Linear constraints

```

min     $f(x) = (x_1^2 + x_2^2 - 1)^2$ 
       $-1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1,$ 
s.t.    $x_1 + x_2 \geq 1$ 
       $x_1 x_2 \geq \frac{1}{2}, x_2 \geq x_1^2, x_1 \geq x_2^2$ 

A = [-1, -1]; b = -1;
lb = [-1; -1]; ub = [1; 1];

c(x) =  $\begin{bmatrix} \frac{1}{2} - x_1 x_2 \\ x_1^2 - x_2 \\ x_2^2 - x_1 \end{bmatrix}; ceq(x) = [];$ 

```

```

% myobj.m
function f=myobj(x)
f = (x(1)^2+x(2)^2-1)^2;

% mycon.m
function [c, ceq]=mycon(x)
c=[1/2-x(1)*x(2);
  x(1)^2-x(2);
  x(2)^2-x(1)]; % nonlinear inequalities c(x) <= 0
ceq=[]; % nonlinear equalities ceq(x) = 0;

% main file for fmincon
[x,fval] = fmincon(@myobj,x0,A,b,[],[],lb,ub,
                  @mycon,options);

```

$$x_1 + x_2 > 1$$

$$-x_1 - x_2 < -1$$

$$xT = [x_1, x_2]$$

$$A = [-1, -1]$$

$$b = -1$$

$$Ax = b \rightarrow 1 \times 2 \times 2 \times 1 = 1 \times 1$$

## The syntax for fmincon

`[x,fval,exitflag]=fmincon(objfun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options);`

- x: optimal solution; fval: optimal value; exitflag: exit condition
- objfun: objective function (usually written in a separate M file)
- x0: starting point (can be infeasible)
- A: matrix for linear inequalities; b: RHS vector for linear inequalities
- Aeq: matrix for linear equalities; beq: RHS vector for linear equalities
- lb: lower bounds; ub: upper bounds
- Nonlcon: [c,ceq]=constraintfunction(x)

```

min     $f(x) = (x_1^2 + x_2^2 - a)^2$ 
       $-1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1,$ 
s.t.    $x_1 + x_2 \geq 1$ 
       $x_1 x_2 \geq \frac{1}{2}, x_2 \geq x_1^2, x_1 \geq x_2^2$ 

```

```

% myobj.m
function f=myobj(x, a)
f = (x(1)^2+x(2)^2-a)^2;
% main file for fmincon
a = 1;
[x,fval] = fmincon(@myobj(x,a),x0,A,b,[],[],lb,ub,
                  @mycon,options);

```

# Demo Matlab

- Example1 (Local optimisation)
- Example2 (Global optimisation)
- Example3 (With/Without gradients)
- <https://fr.mathworks.com/help/gads/solving-a-mixed-integer-engineering-design-problem-using-the-genetic-algorithm.html>

For example, consider Rosenbrock's function

$$f(x) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2,$$

which is described and plotted in [Solve a Constrained Nonlinear Problem](#). The gradient of  $f(x)$  is

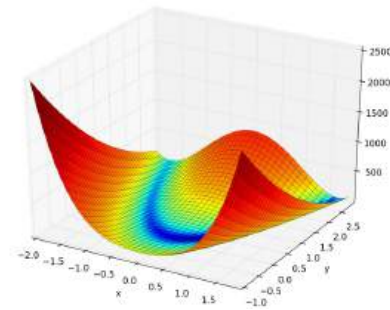
$$\nabla f(x) = \begin{bmatrix} -400(x_2 - x_1^2)x_1 - 2(1 - x_1) \\ 200(x_2 - x_1^2) \end{bmatrix},$$

and the Hessian  $H(x)$  is

$$H(x) = \begin{bmatrix} 1200x_1^2 - 400x_2 + 2 & -400x_1 \\ -400x_1 & 200 \end{bmatrix}.$$

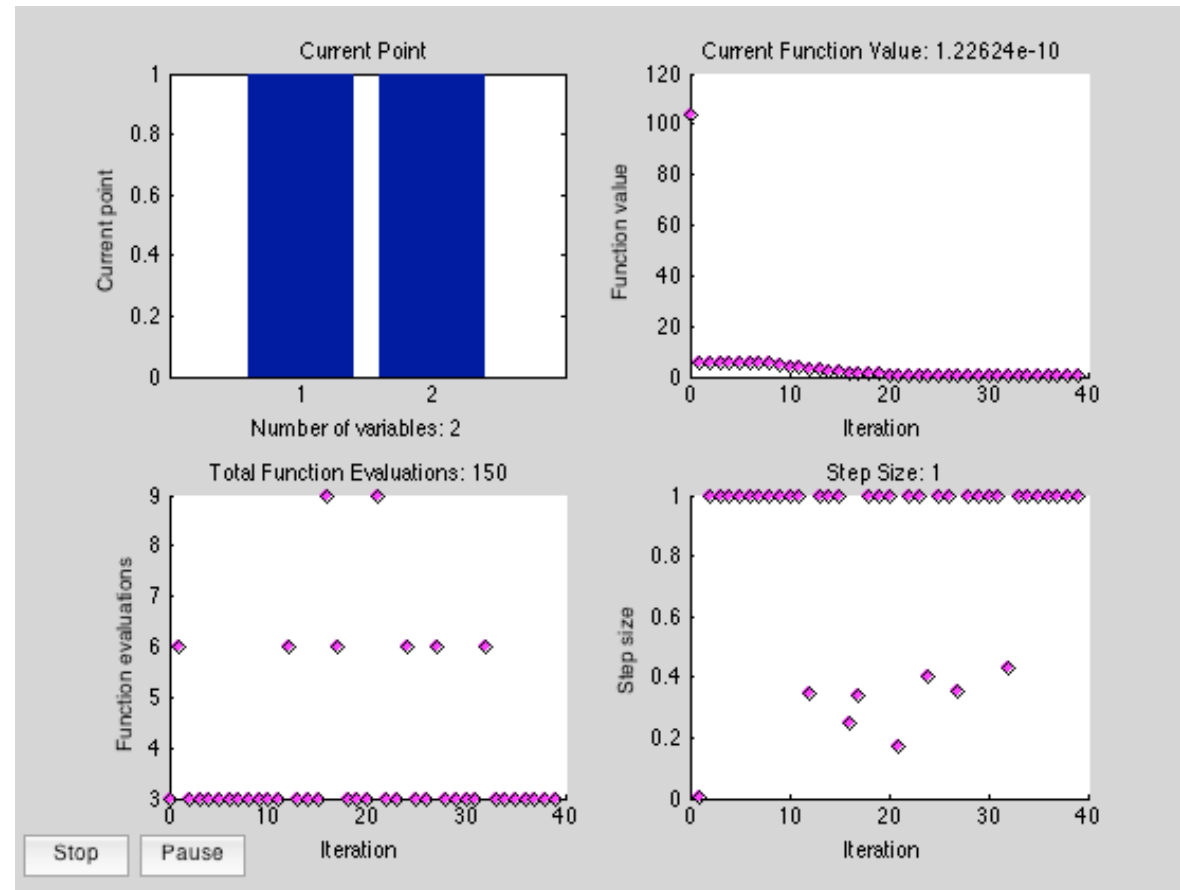
`rosenthree` is an unconditionalized function that returns the Rosenbrock function with its gradient and Hessian:

```
function [f g H] = rosenthree(x)
% Calculate objective f, gradient g, Hessian H
f = 100*(x(2) - x(1)^2)^2 + (1-x(1))^2;
g = [-400*(x(2)-x(1)^2)*x(1)-2*(1-x(1));
     200*(x(2)-x(1)^2)];
H = [1200*x(1)^2-400*x(2)+2, -400*x(1);
     -400*x(1), 200];
```

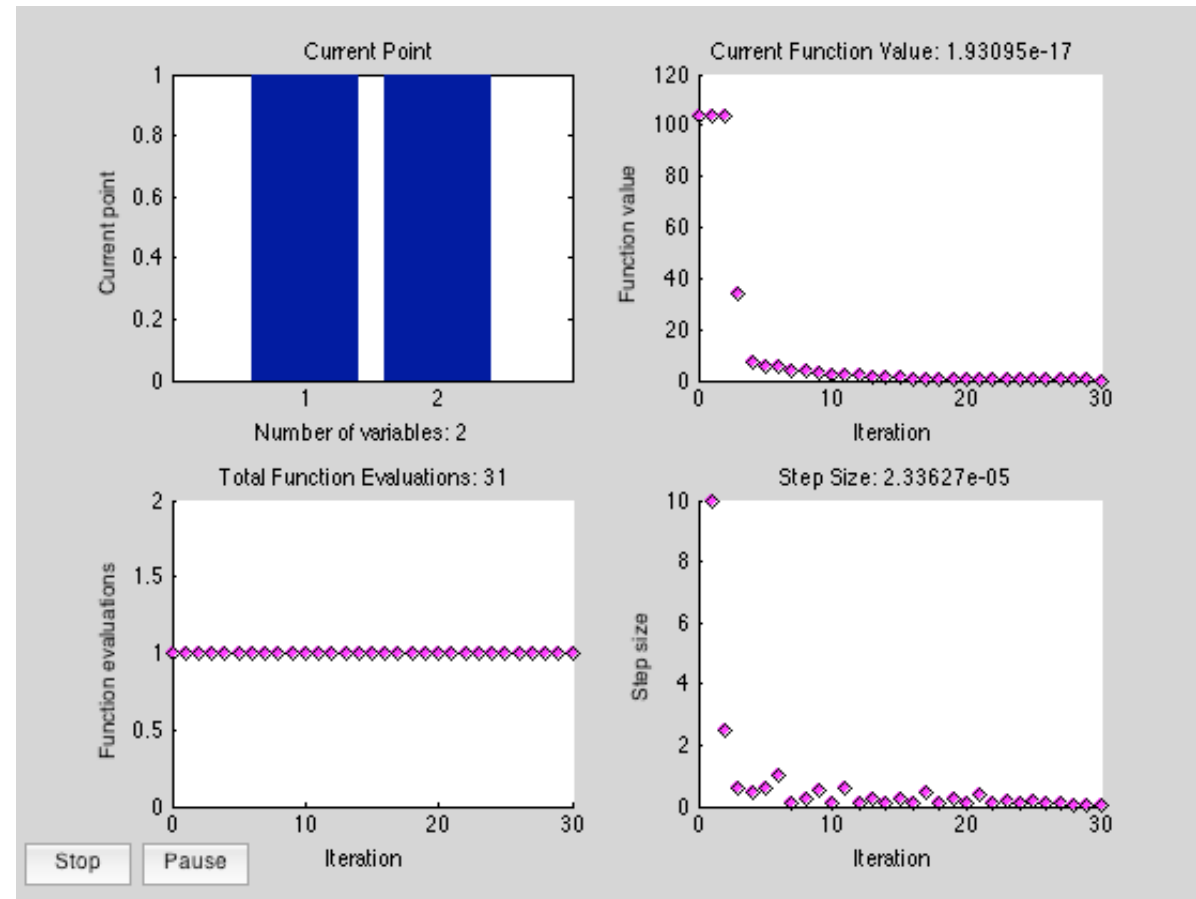


```
syms x y ;
f=100*(y - x^2)^2 + (1-x)^2
fx=diff(f,x)
fy=diff(f,y)
%analytical gradients
f =(x - 1)^2 + 100*(- x^2 + y)^2
fx =2*x - 400*x*(- x^2 + y) - 2
fy =- 200*x^2 + 200*y
g=[fx;fy]
```

# Results (w/o gradient)

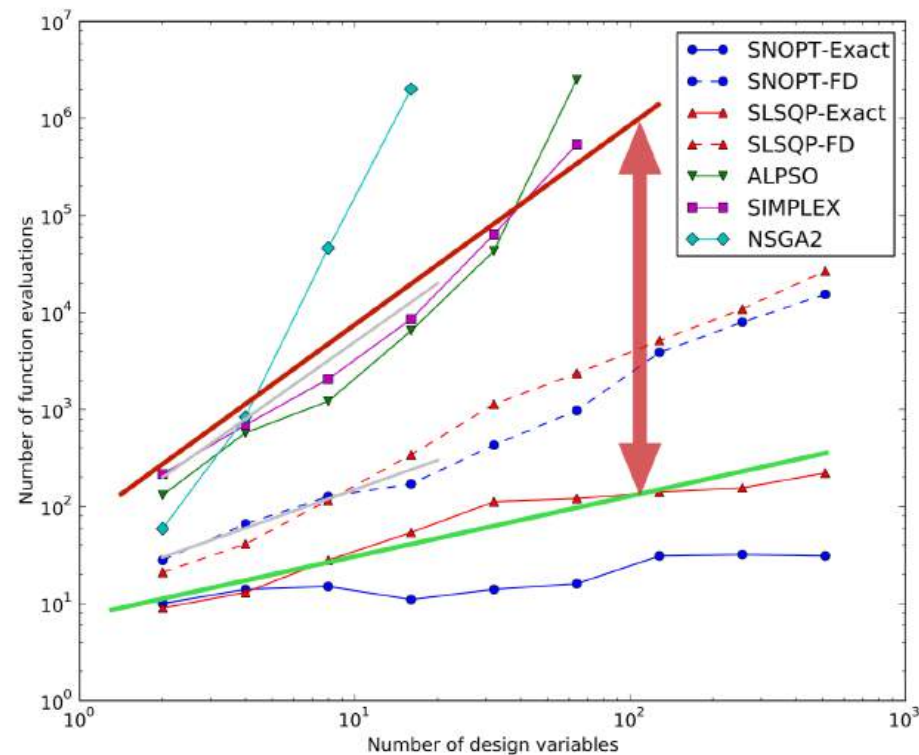


# Results (w/ gradient)

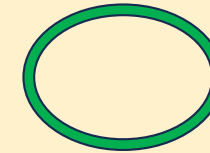




Gradient-based optimization is our only hope  
to explore large-dimensionality design spaces\*



# Second exercice



$f(x)$

$h(x)$

$g(x)$

## 1 Optimization problems in practice

Rewrite the following practical optimization problems using the following canonical form

$$\min_{x \in \Omega} f(x), \quad \Omega = \{x \in \mathbb{R}^n \mid h(x) = 0, g(x) \leq 0\}$$

where  $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$  represents  $m$  equality constraints ( $h(x) = [h_i(x)]_{i=1}^m$ ) and  $g : \mathbb{R}^n \rightarrow \mathbb{R}^p$  represents  $p$  inequality constraints ( $g(x) = [g_j(x)]_{j=1}^p$ ).

1. Find two positive numbers whose sum is 300 and whose product is a maximum.
2. Find two positive numbers whose product is 750 and for which the sum of one and 10 times the other is a minimum.

Warning  
Matlab Negative Null Form  
Python Positive Null Form

# Example of Optimizers

**CHALLENGE #1** On Rosenbrock function minimization

<https://colab.research.google.com/drive/1rGHklpVM4Gqy9eq1Y10flQW9umljeEx>

# Example of using autograd

- [https://colab.research.google.com/drive/1oDwKKU-WUnogMAxTg0Fx\\_NBuk5-8LD1x](https://colab.research.google.com/drive/1oDwKKU-WUnogMAxTg0Fx_NBuk5-8LD1x)

# Reformulation

Basic idea: convert to an unconstrained optimization problem

- Penalty function methods
- Append a penalty for violating constraints (exterior penalty methods)
- Append a penalty as you approach infeasibility (interior point methods)
- Method of Augmented Lagrange multipliers

# Augmented Lagrange Method

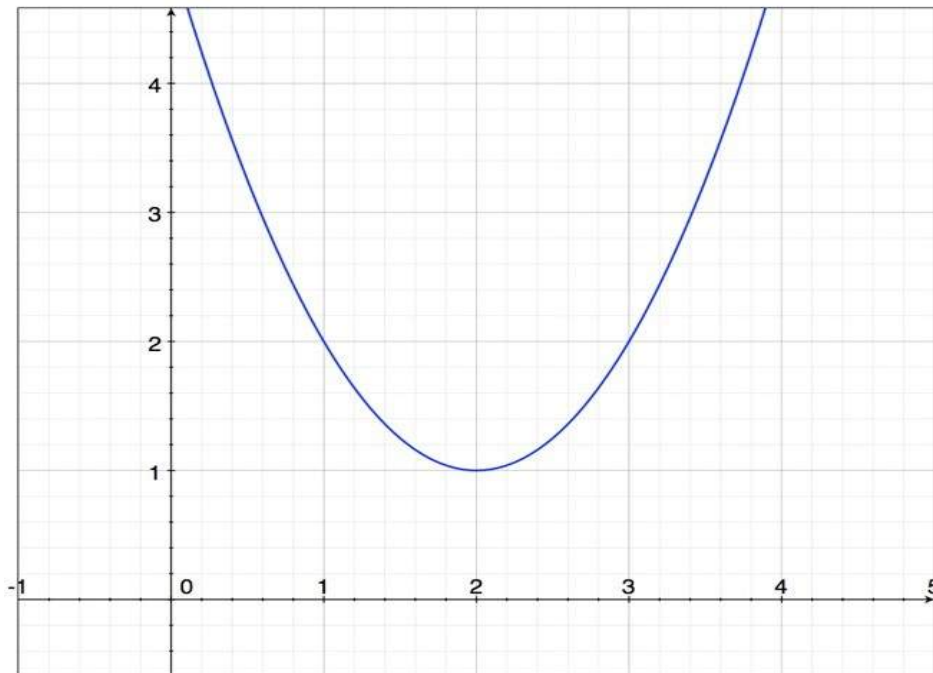
- Adaptation of penalty method for equality constraints

$$p_{\text{Lagrange}}(\mathbf{x}) = \frac{1}{2}\rho \sum_i (h_i(\mathbf{x}))^2 - \sum_i \lambda_i h_i(\mathbf{x})$$

$$\boldsymbol{\lambda}^{(k+1)} = \boldsymbol{\lambda}^{(k)} - \rho \mathbf{h}(\mathbf{x})$$

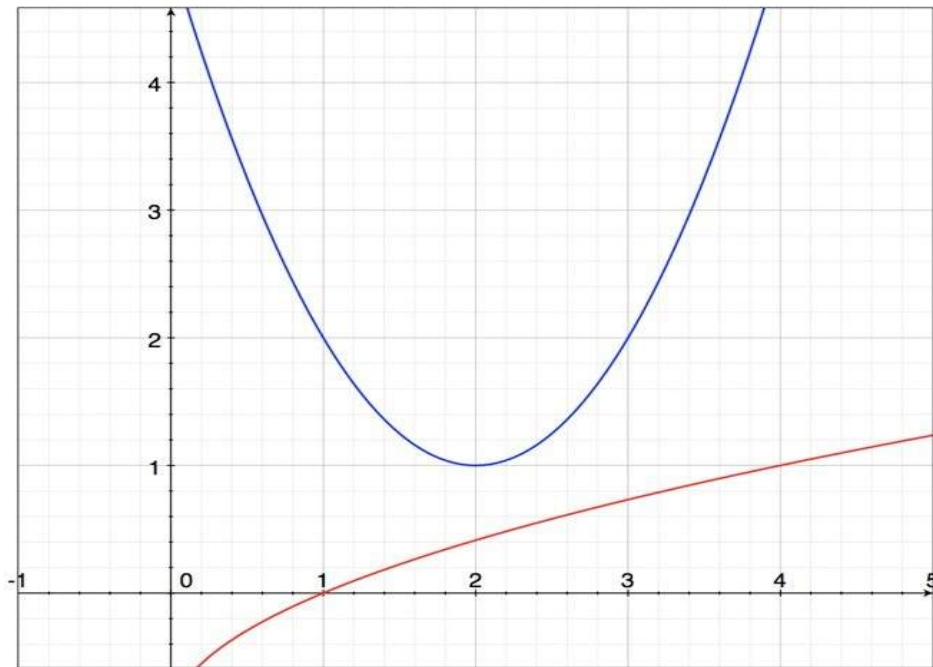
- $\lambda$  converges towards the Lagrange multiplier

# Example



Objective:  $f(x) = (x - 2)^2 + 1$

# Example

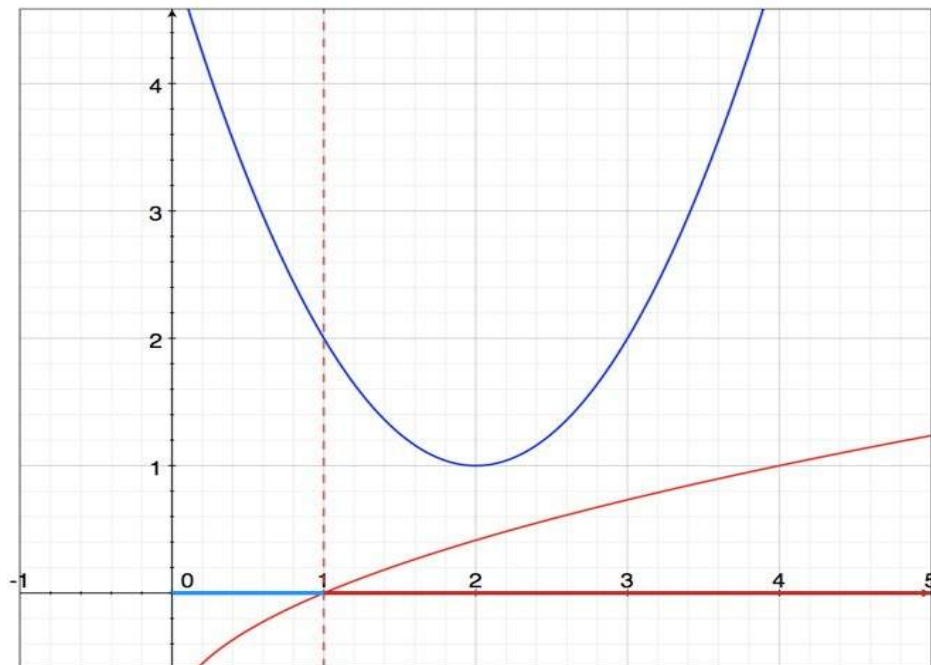


Objective:  $f(x) = (x - 2)^2 + 1$

Constraint:  $c(x) \leq \sqrt{x} - 1$



# Example



Objective:  $f(x) = (x - 2)^2 + 1$

Constraint:  $c(x) \leq \sqrt{x} - 1$

Bound:  $x \geq 0$

Feasible Region:  $0 \leq x \leq 1$

# Penalty function methods

1. Initialize penalty parameter
2. Initialize solution guess
3. Minimize penalized objective starting from guess
4. Update guess with the computed optimum
5. Go to 3., repeat

The diagram shows the formula  $\pi(x, \rho) = f(x) + \rho\phi(x)$  with three annotations: a blue arrow from 'Objective' to  $f(x)$ , a green arrow from 'Penalty function' to  $\phi(x)$ , and a purple arrow from 'Penalty parameter (non-negative)' to  $\rho$ .

$$\pi(x, \rho) = f(x) + \rho\phi(x)$$

Objective

Penalty function

Penalty parameter (non-negative)

# How to find the Penalty function?

With the constraint  $x - 5 \leq 0$ , we need a penalty that is:

- 0 when  $x - 5 \leq 0$  (the constraint is satisfied)
- positive when  $x - 5$  is  $> 0$  (the constraint is violated)

This can be done using the operation

$$P(x) = \max(0, x - 5)$$

which returns the maximum of the two values, either 0 or whatever  $(x - 5)$  is.

We can make the penalty more severe by using

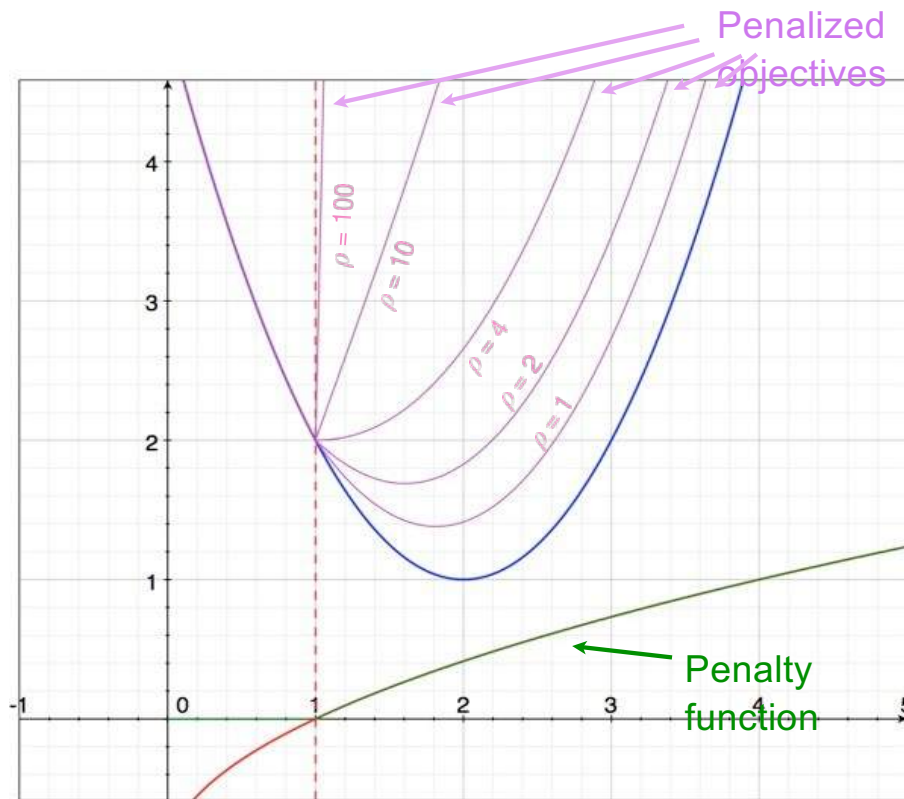
$$P(x) = \max(0, x - 5)^2.$$

This is known as a **quadratic loss function**.

# Linear Exterior Penalty Function

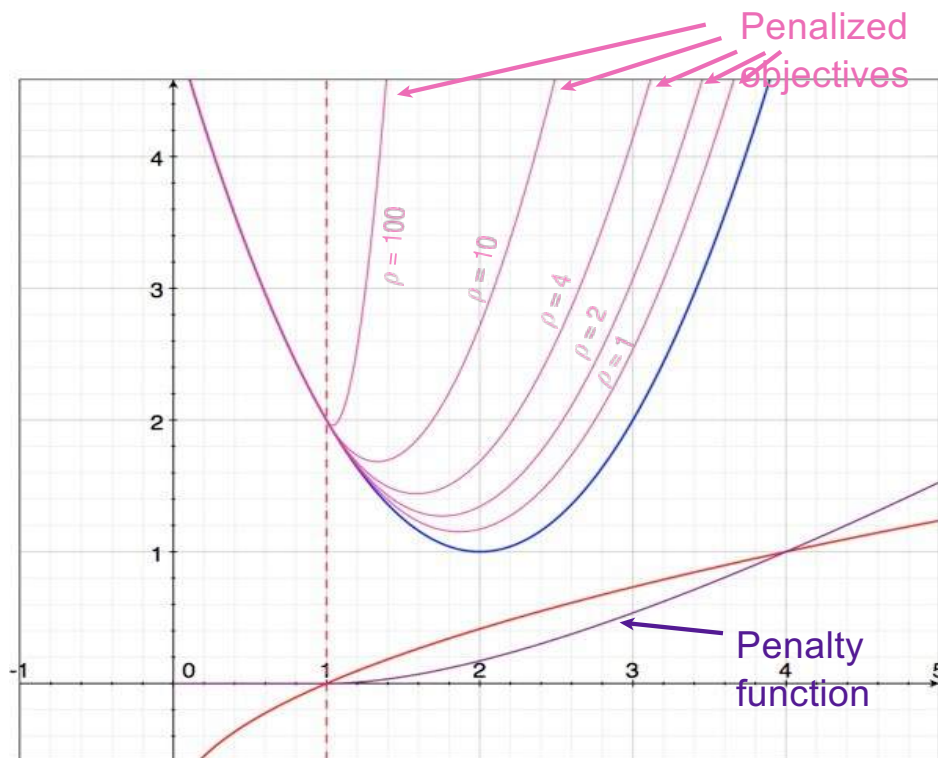
$$\phi_i(x) = \max(0, c_i(x) - u_i)$$

$u_i$ =RHS (negative null form)



# Quadratic Exterior Penalty Function

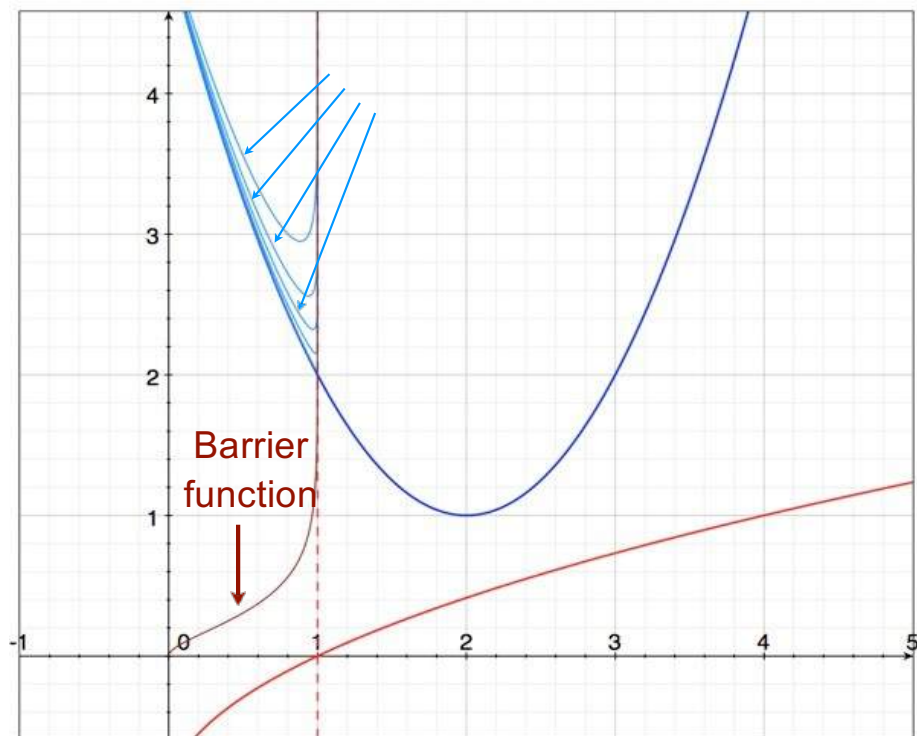
$$\phi_i(x) = \left[ \underbrace{\max(0, c_i(x) - u_i)}_{\text{Constraint violation}} \right]^2$$



# Interior-Point Methods

$$\pi(x, \mu) = f(x) - \underbrace{\mu \log(u_i - c_i(x))}_{\text{Barrier function}}$$

Barrier parameter

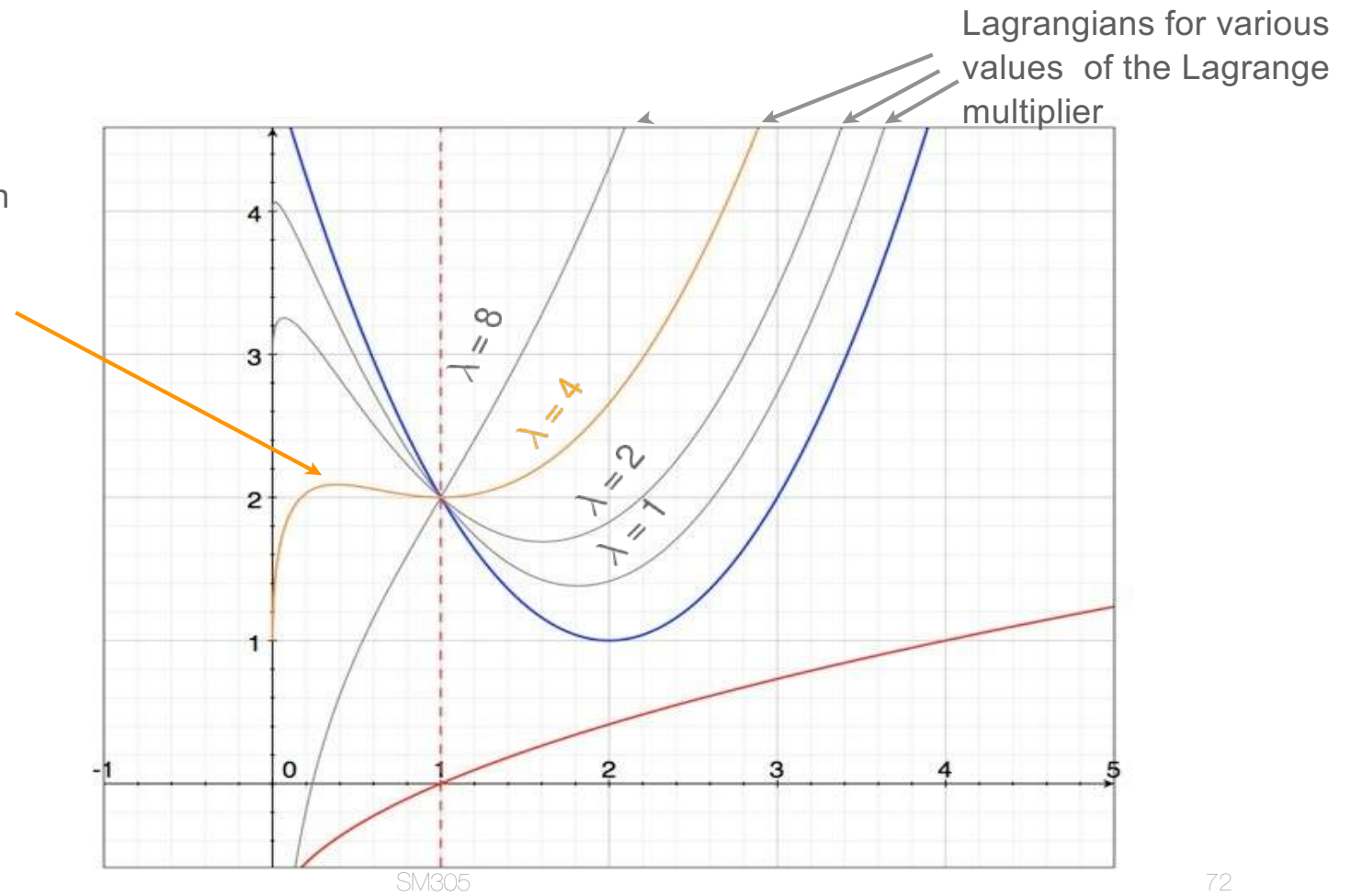


Also called barrier methods, interior point methods ensure that each step is feasible. This allows premature termination to return a nearly optimal, feasible point. Barrier functions are implemented similar to penalties but must meet the following conditions:

1. Continuous
2. Non-negative
3. Approach infinity as  $x$  approaches boundary

# Lagrange Multipliers

There is an optimal  $\lambda$  for which we obtain the constrained solution in  $x$  by minimizing the Lagrangian for that  $\lambda$



# Summary

- Constraints are requirements on the design points that a solution must satisfy
- Some constraints can be transformed or substituted into the problem to result in an unconstrained optimization problem
- Analytical methods using Lagrange multipliers yield the generalized Lagrangian and the necessary conditions for optimality under constraints
- A constrained optimization problem has a dual problem formulation that is easier to solve and whose solution is a lower bound of the solution to the original problem



# Summary

- Penalty methods penalize infeasible solutions and often provide gradient information to the optimizer to guide infeasible points toward feasibility
- Interior point methods maintain feasibility but use barrier functions to avoid leaving the feasible set

# Supplementary materials & lecture notes

## CHALLENGE #3 Reformulate the problem

$$\min_{\mathbf{x}} f(\mathbf{x}) = (x_1 - 3)^2 + (x_2 - 3)^2$$

subject to

$$h(\mathbf{x}) = x_1 + x_2 - 4 = 0$$

to an unconstrained optimization problem using penalization.

$$P(\mathbf{x}, R) = f(\mathbf{x}) + \Omega(R, g(\mathbf{x}), h(\mathbf{x}))$$

$$\Omega = R h_j^2(\mathbf{x})$$

find  $\mathbf{x}1^*, \mathbf{x}2^*$  for  $R = [10, 100, 1000, 10000, 10^5]$

# Supplementary materials

**CHALLENGE #3** Reformulate the problem

<https://colab.research.google.com/drive/1rGHlkIpVM4Gqy9eq1Y10fIQW9umljeEx>

# Let's start minimizing

- Read and finish the notebook called 00-intro.ipynb and 02-constrained-optimization.ipynb

# Part2: MultiObjective Optimization

Trade offs?

# MultiObjective Optimization (MOO)

Definition:

Multi-objective optimization or Pareto optimization (also known as multi-objective programming, vector optimization, multicriteria optimization, or multiattribute optimization) is an optimization problem that involves

more than 1 objective function to be optimized simultaneously.

→ optimal decisions need to be taken in the presence of trade-offs between two or more conflicting objectives

<https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/download/2198/2030/>

# MultiObjective Optimization (MOO)

Main message: Multiobjective optimization is somewhat of a misnomer  
- you actually have to have predefined weightings for each of the objectives you care about, or implement them as constraints

<https://openmdao.github.io/PracticalMDO/Notebooks/Optimization/multiobjective.html>

# Trade off

When doing multiobjective optimization,  
I strongly recommend performing multiple single-  
objective optimizations instead of using a multiobjective  
optimizer.

- Let's view aerostructural wing design for an example. In a simple trade-off, if you extend the wingspan you get more lift and aerodynamic performance, but your structural masses and costs become greater to withstand the larger load.
- This trade-off means that you cannot maximize the aerodynamic performance and minimize the structural weight - there must be some sort of a balancing act. In reality we care about the total performance of the airplane, more than any subdiscipline, so our objective function usually captures effects from multiple disciplines at once.



# Multi-Objective Optimization

$$\begin{aligned} & \min f_i(x), \quad i = 1, \dots, d \\ \text{Subject to } & g(x) \geq, \quad h(x) = 0 \\ & F(x) = [f_1(x), \dots, f_d(x)] \end{aligned}$$

We know how to do this:

$$\min f(x)$$

$$\text{Solution: } f(x) = \sum_i w_i f_i(x)$$

# Multi-Objective Optimization

$$\begin{aligned} & \min f_i(x), \quad i = 1, \dots, d \\ \text{Subject to } & g(x) \geq, \quad h(x) = 0 \\ & F(x) = [f_1(x), \dots, f_d(x)] \end{aligned}$$

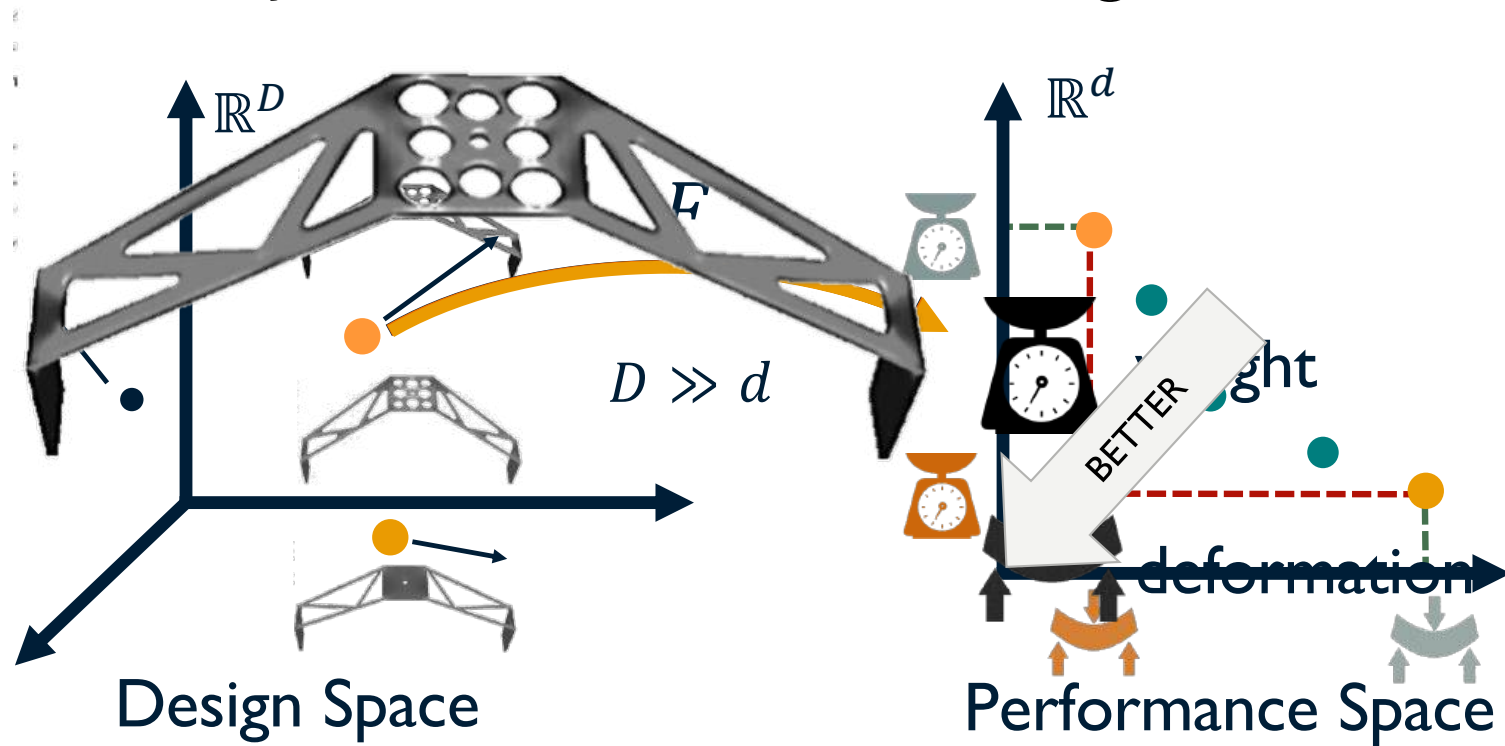
We know how to do this:

$$\min f(x)$$

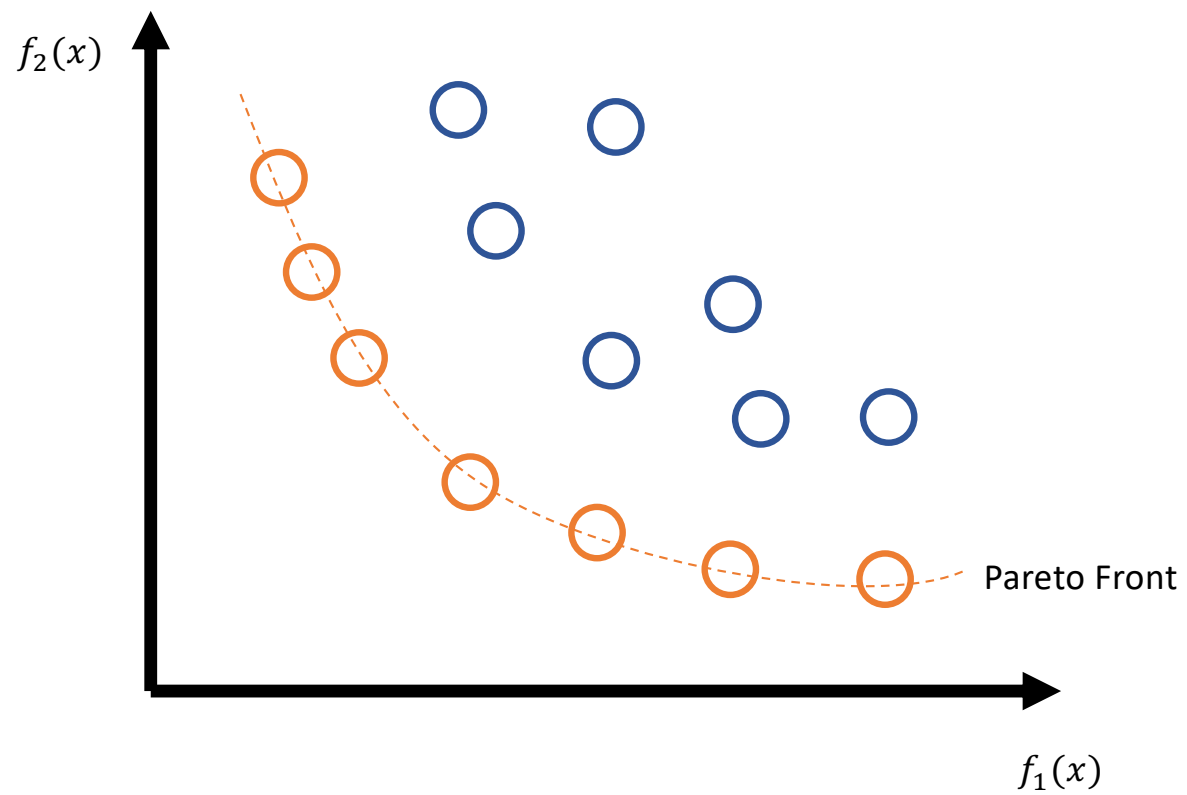
$$\text{Solution: } f(x) = \sum_i w_i f_i(x)$$

How do you pick the weights?

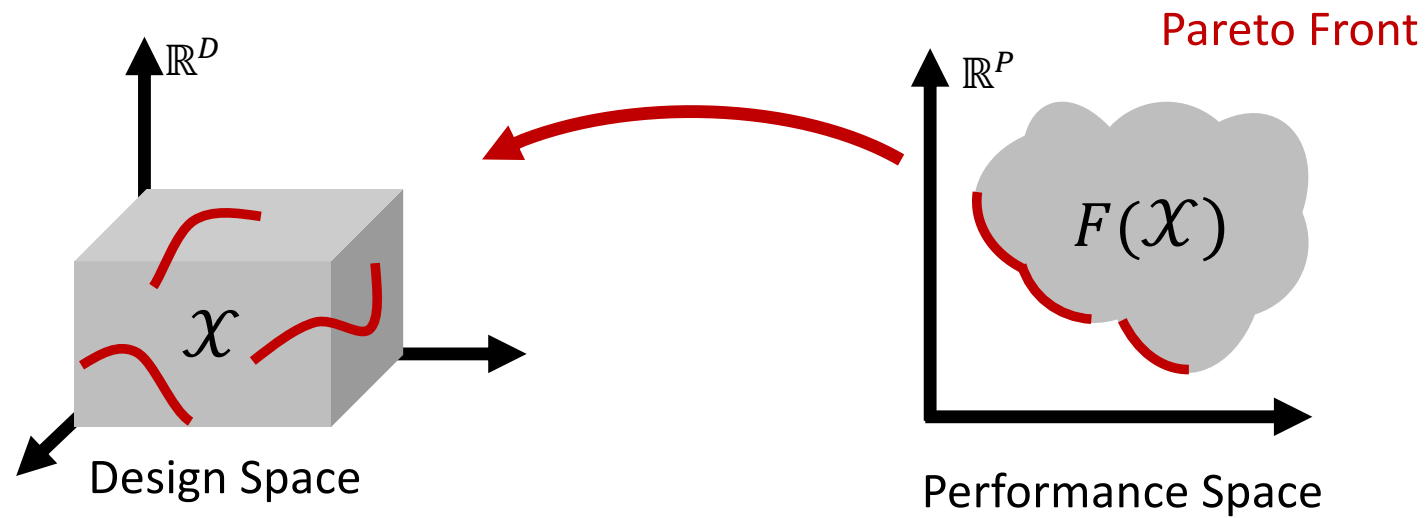
# When Objectives are Conflicting



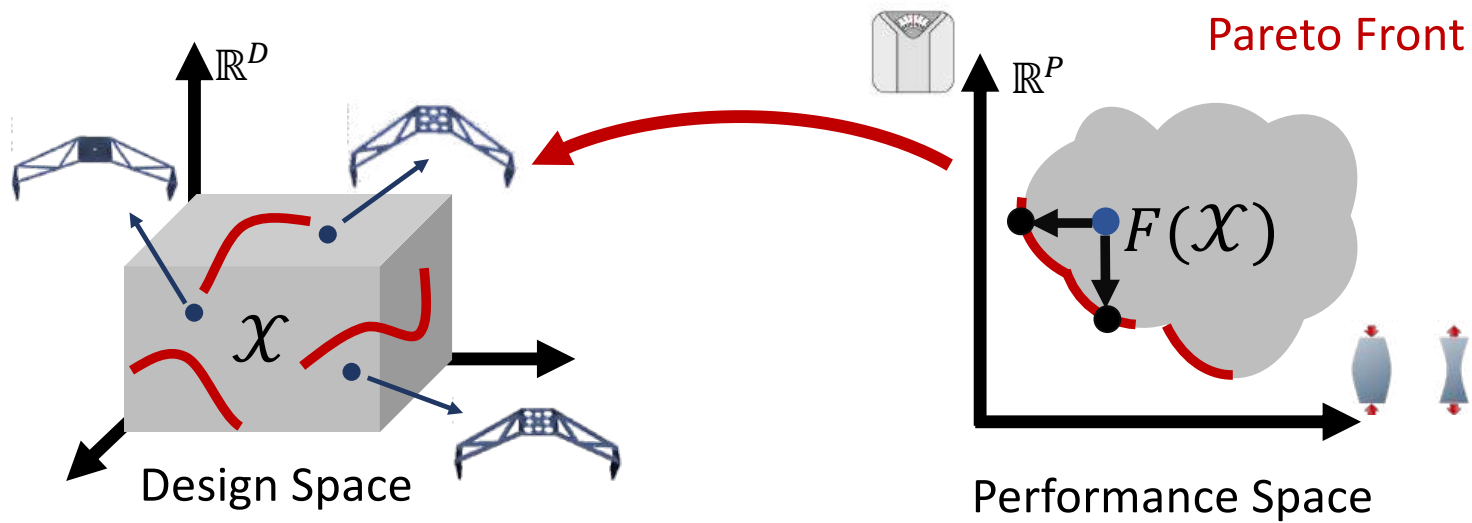
For Minimization



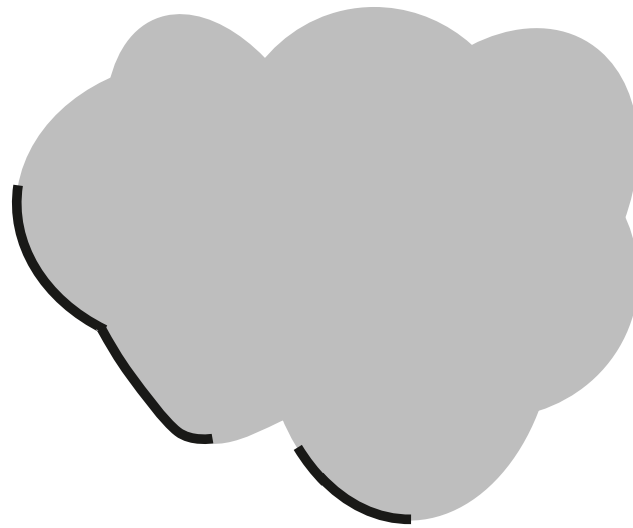
# Space of Optimal Solutions



# Space of Optimal Solutions



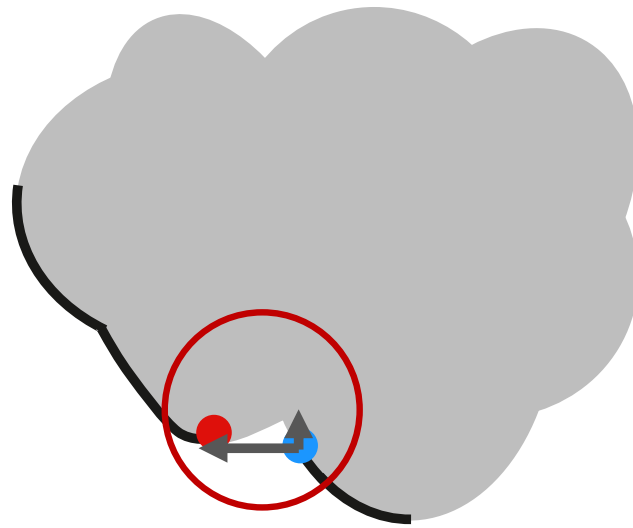
# The Geometry of the Front



Not a straight line!

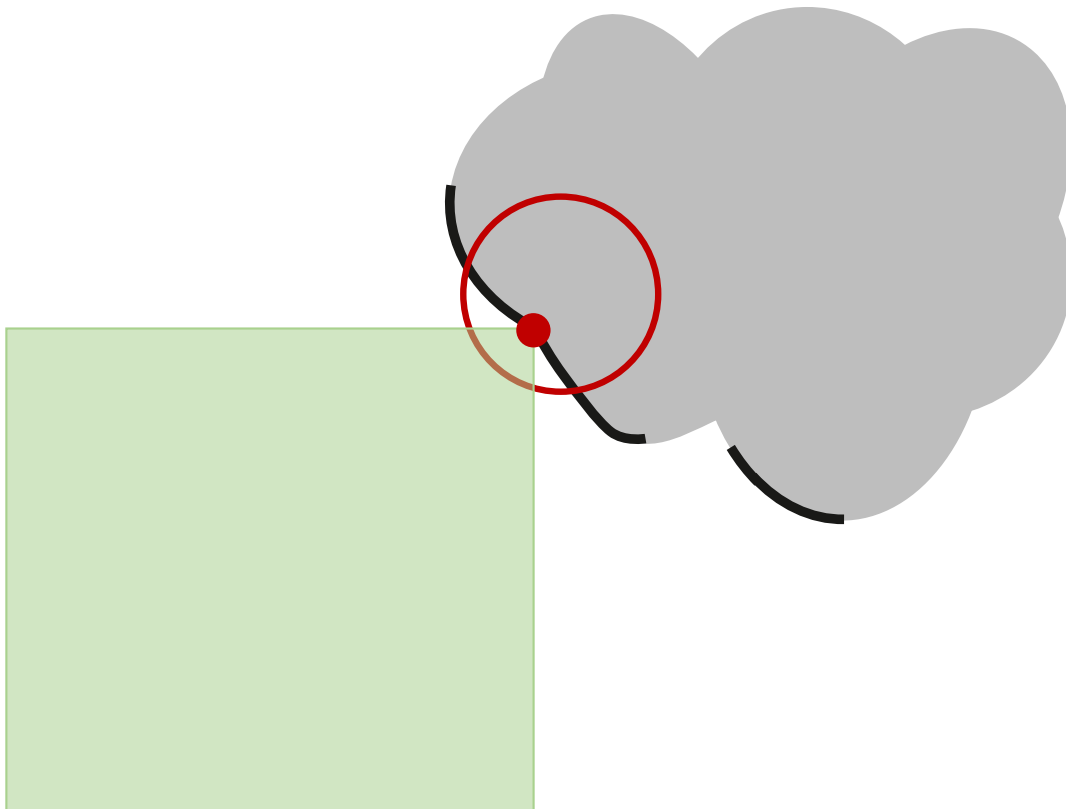
Solution:  $f(x) = \sum_i w_i f_i(x)$  ☹

The Front Can Have Gaps





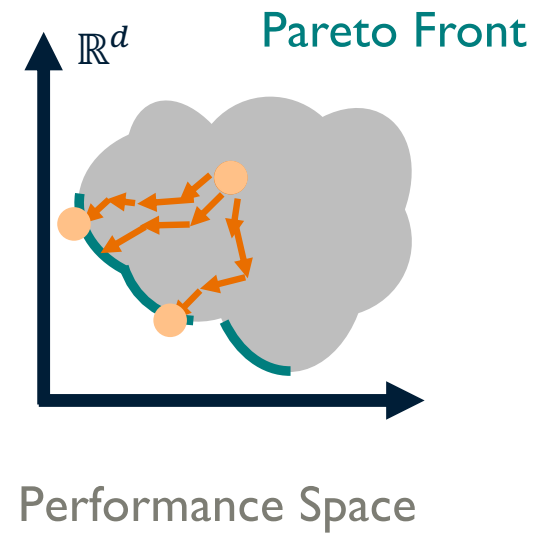
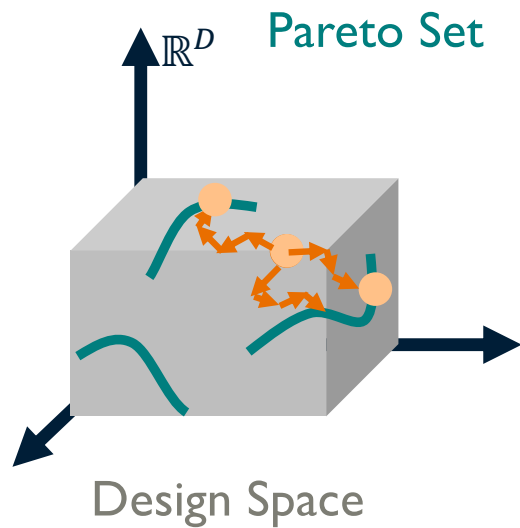
The Front Can Have Non-Convex Regions



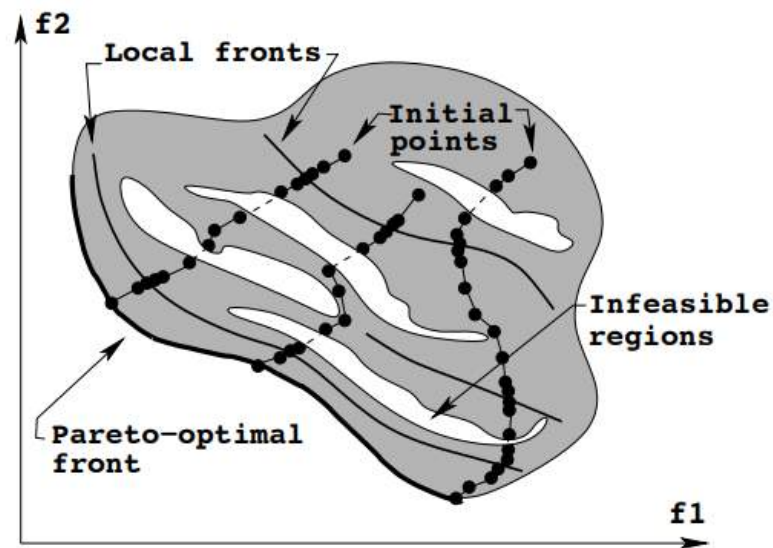
# Pareto Front Discovery

Main Challenge:

- Converge to optimal solutions
- Diverse set that describes the full front

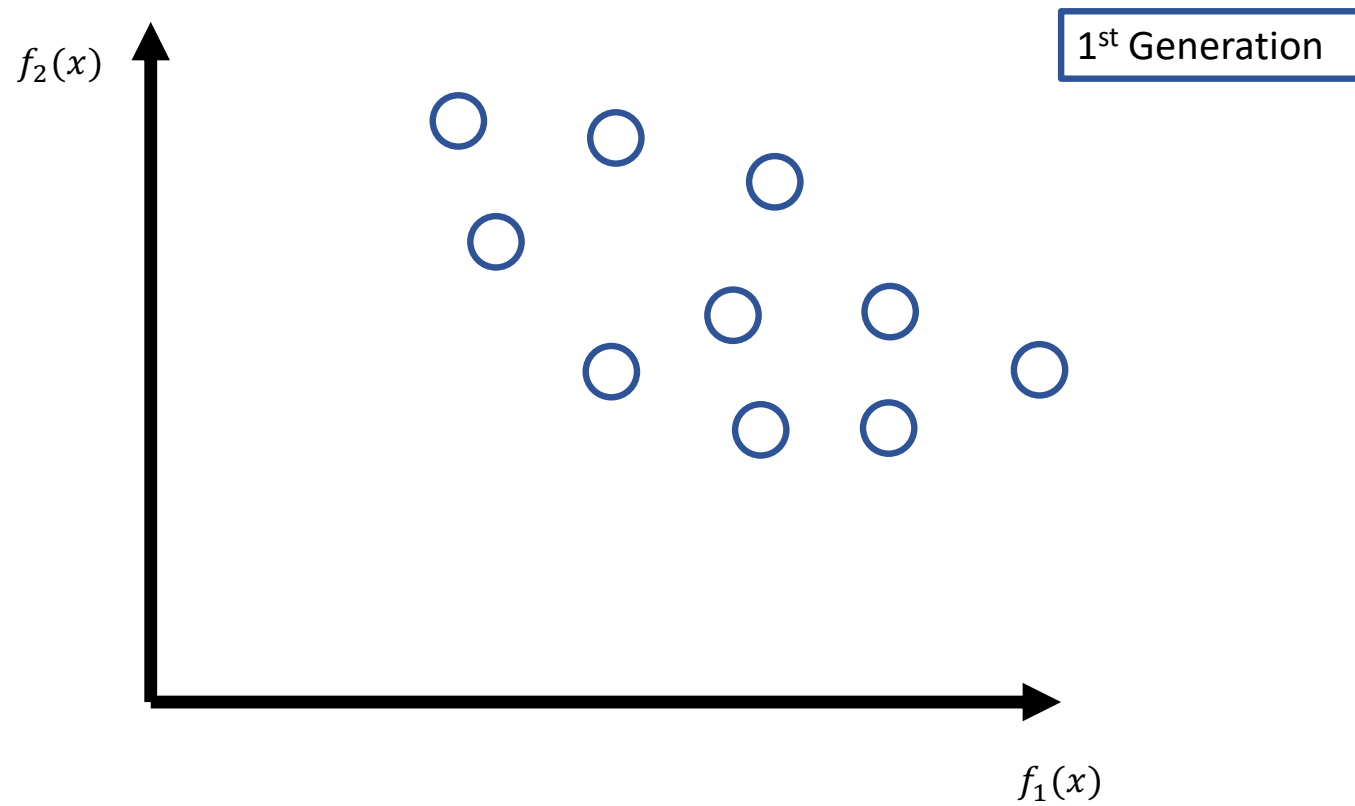


Problem: Each Single Objective Optimization is not SIMPLE

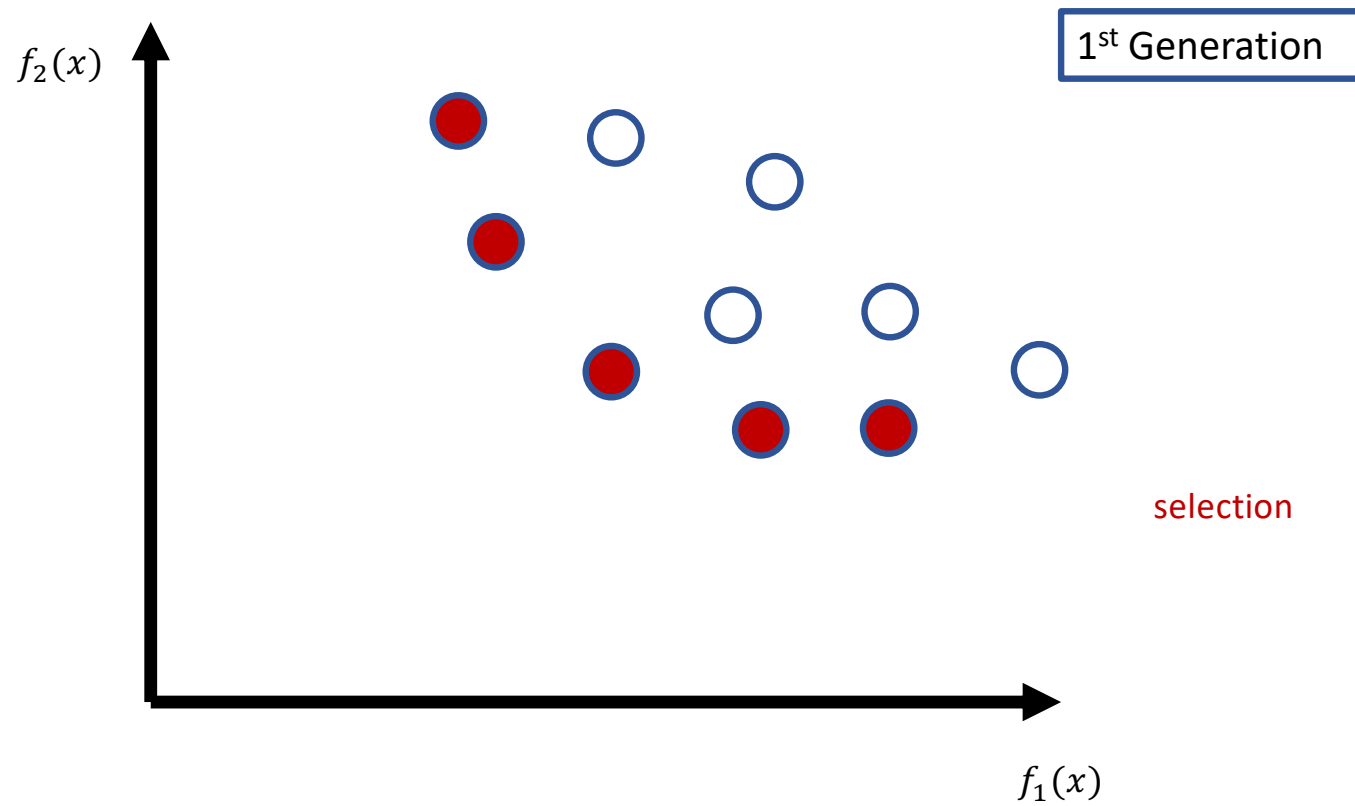


Move many points in parallel towards the front at the same time?

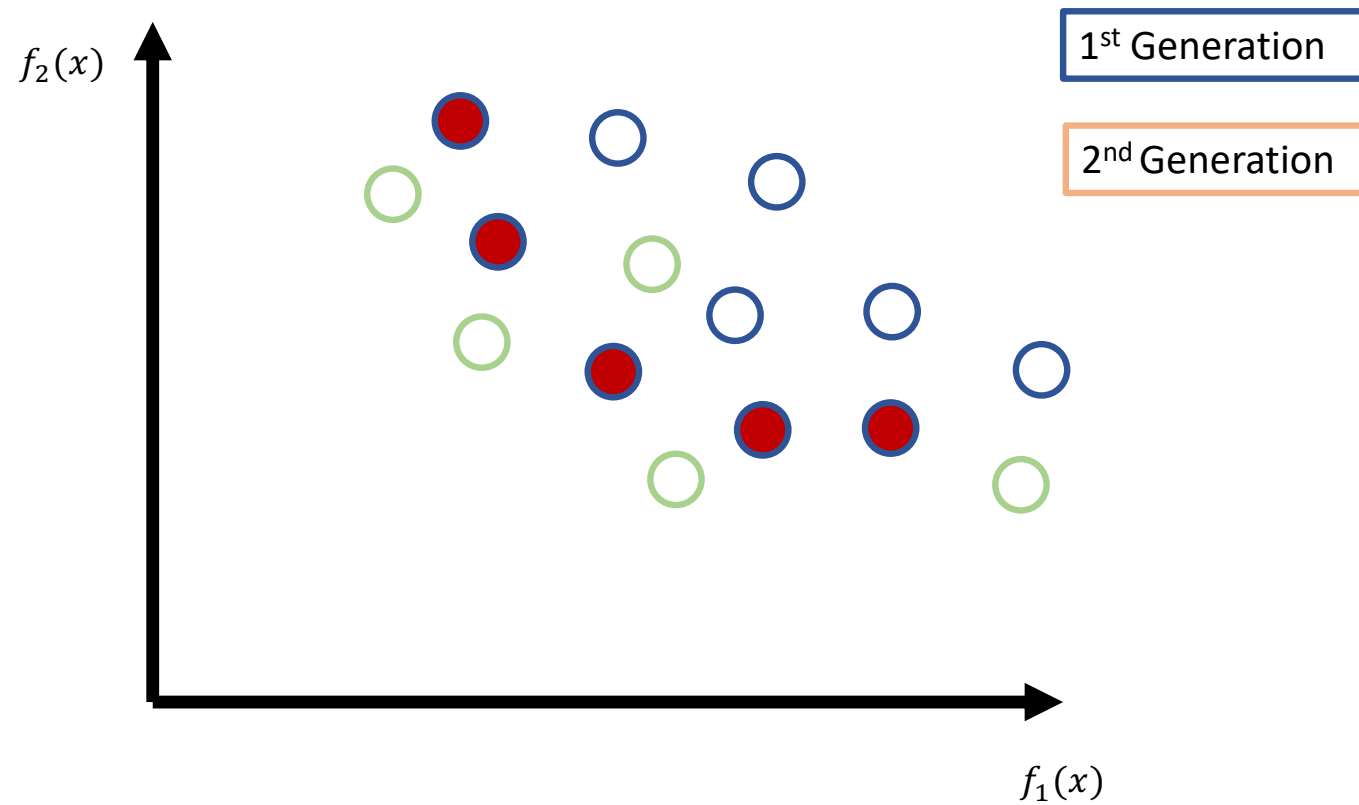
# Evolutionary Algorithms



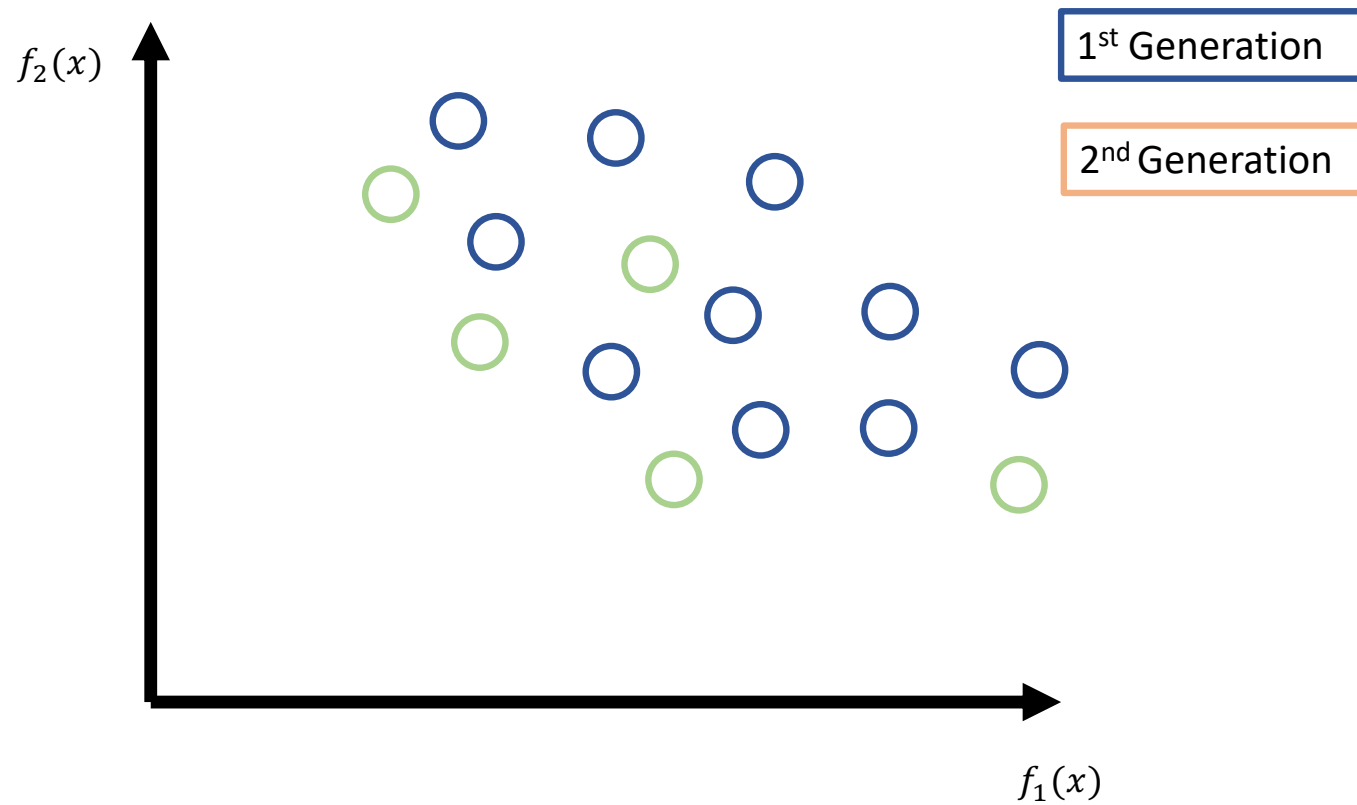
# Evolutionary Algorithms



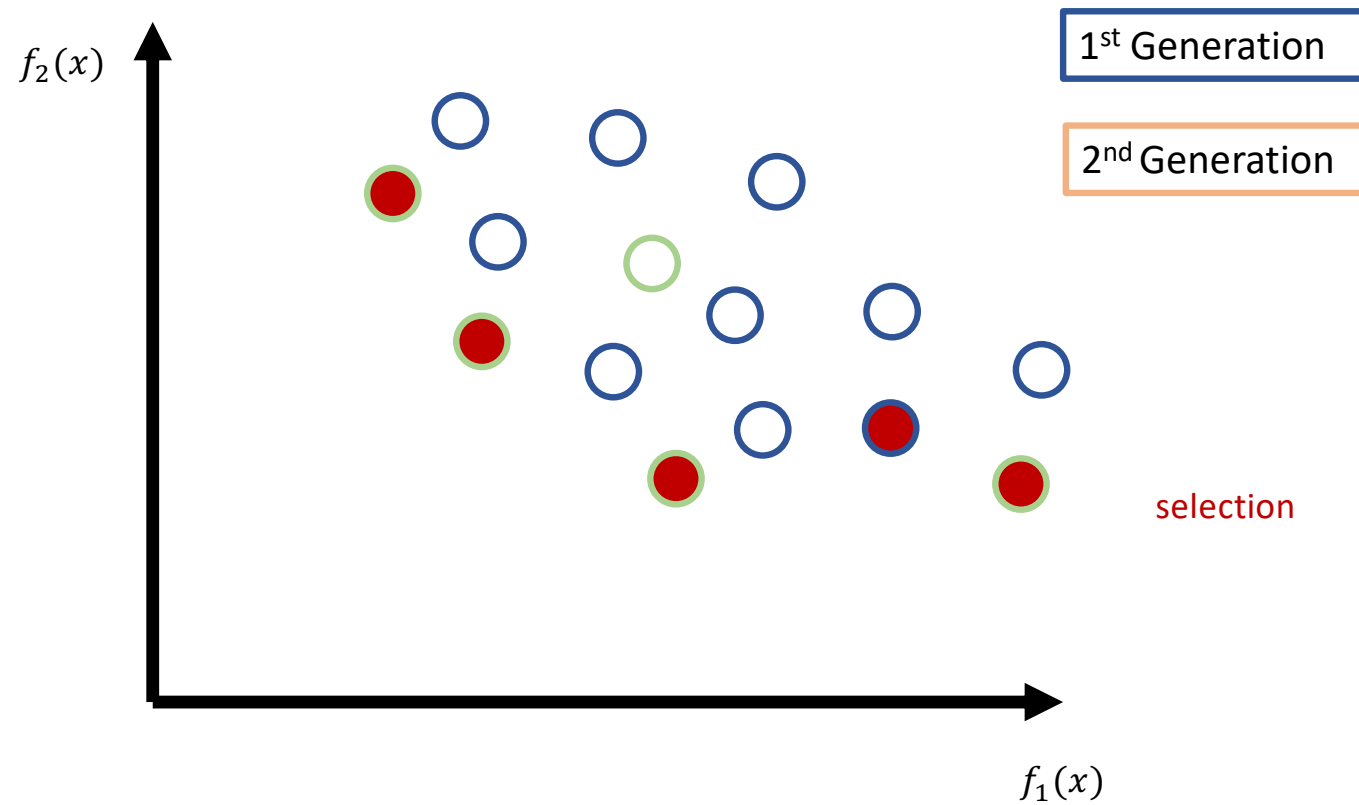
# Evolutionary Algorithms



# Evolutionary Algorithms

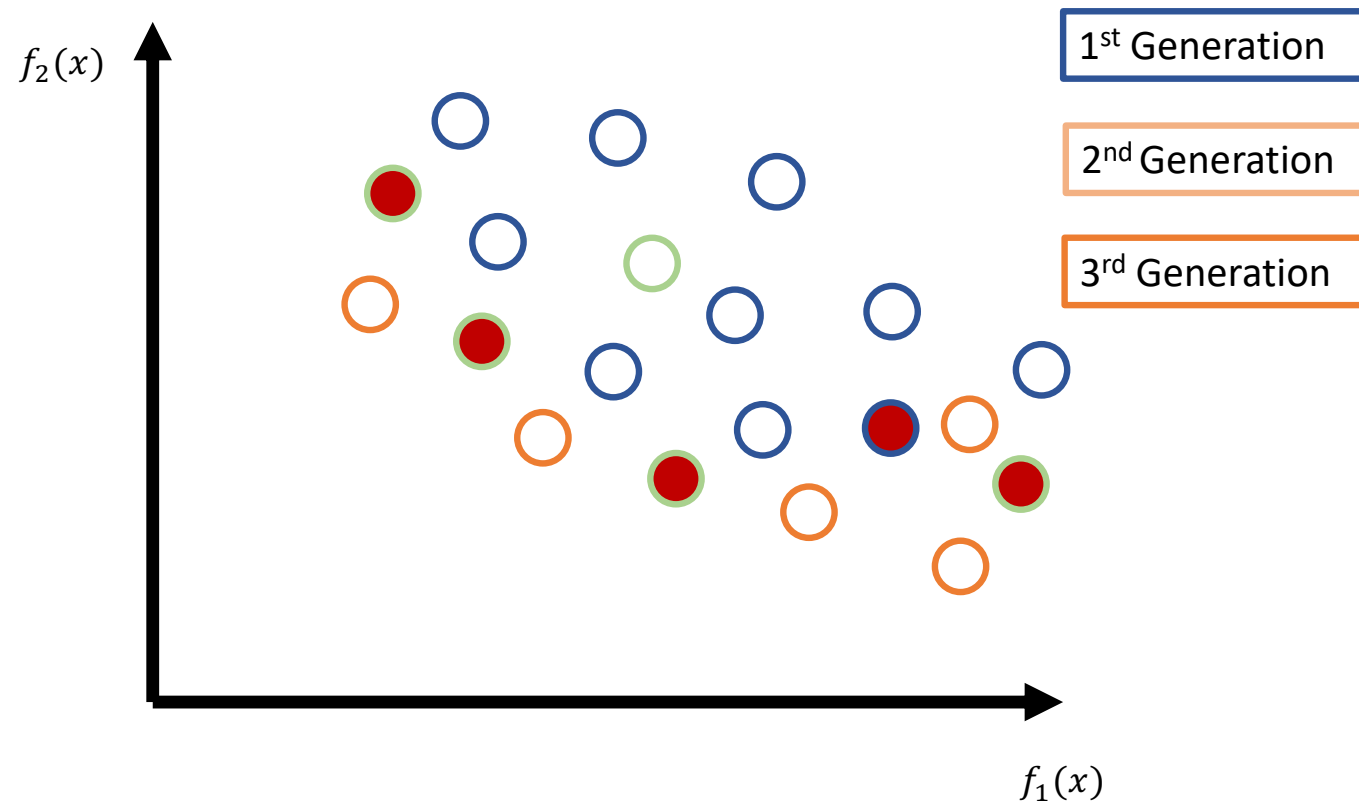


# Evolutionary Algorithms

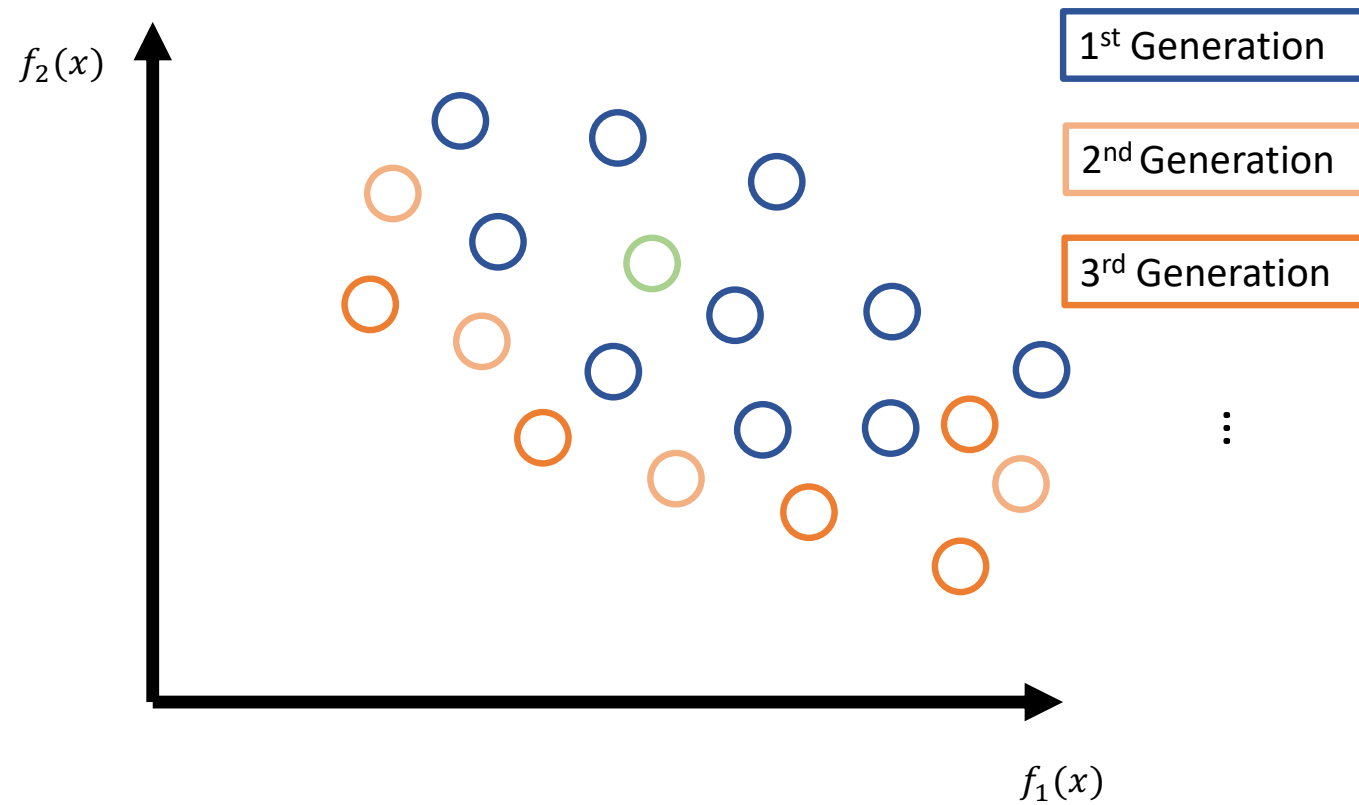




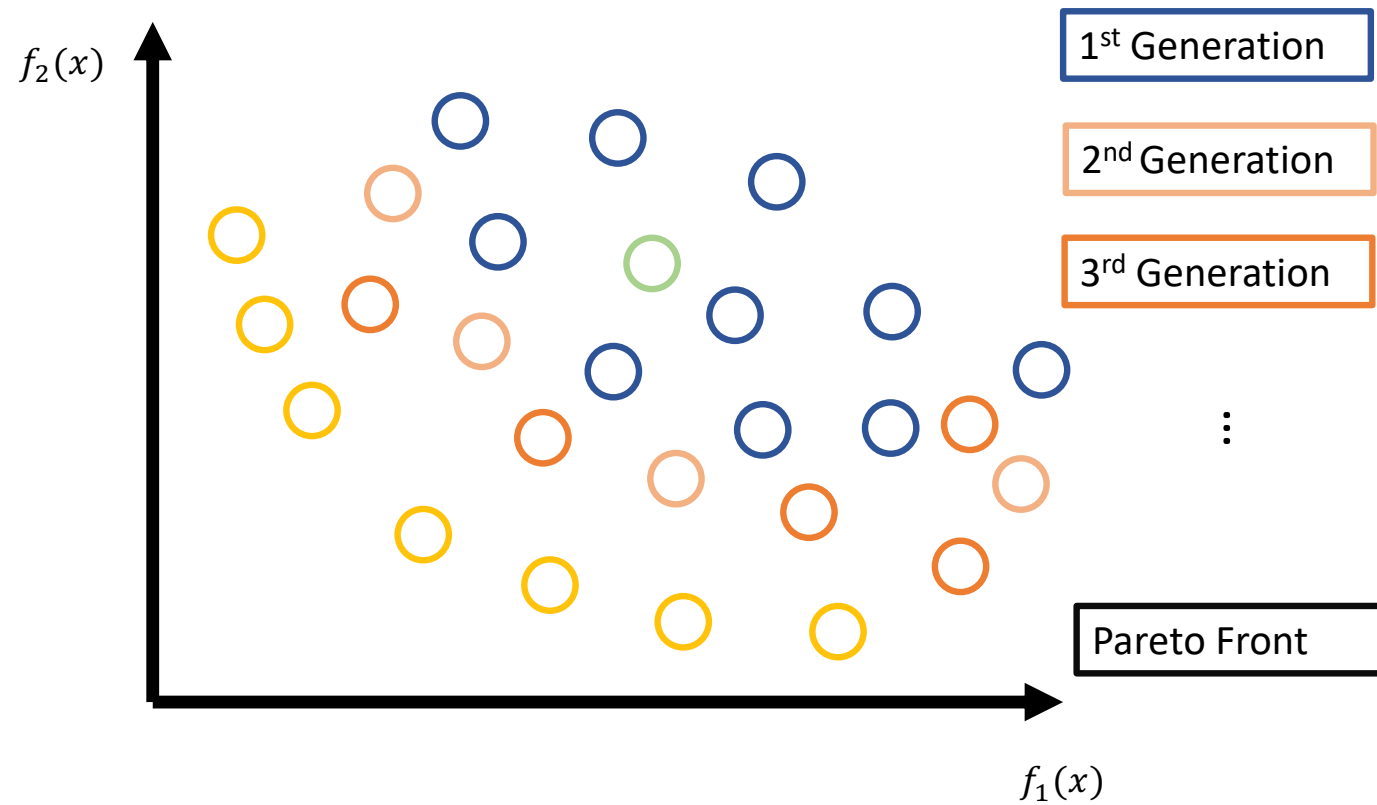
# Evolutionary Algorithms



# Evolutionary Algorithms



# Evolutionary Algorithms



# Elitist Non-dominated Sorting GA or NSGA-II

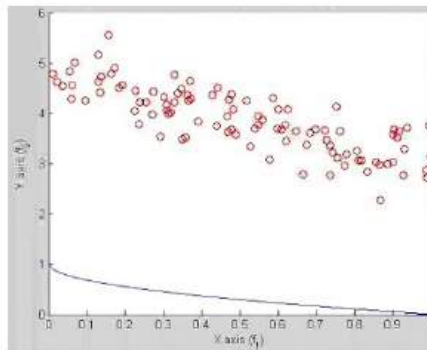


Figure 10: Initial population.

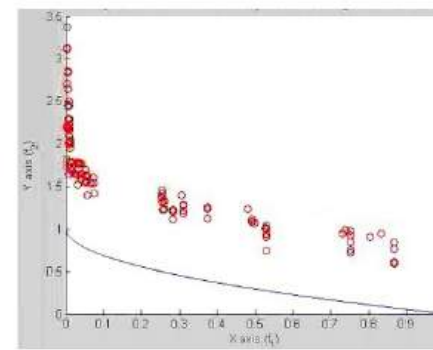


Figure 11: Population at generation 10.

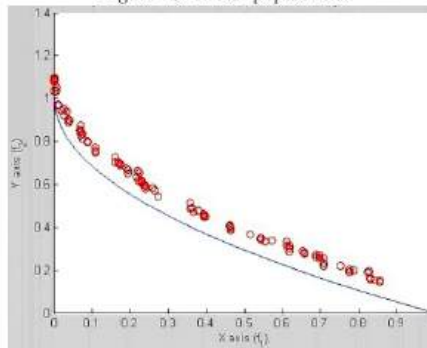


Figure 12: Population at generation 30.

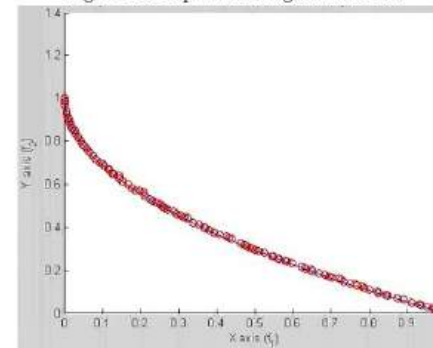
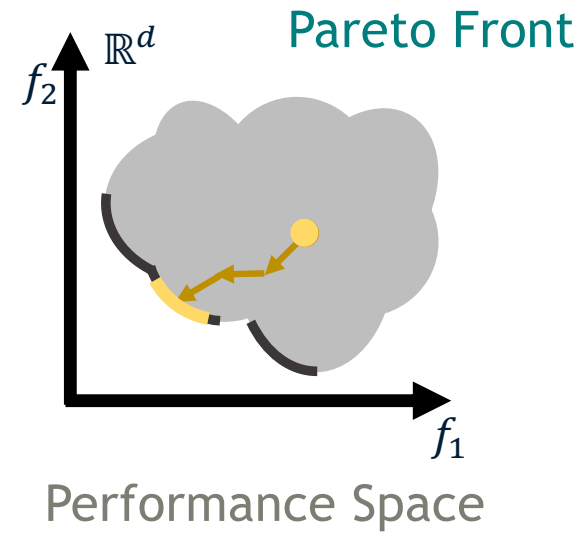
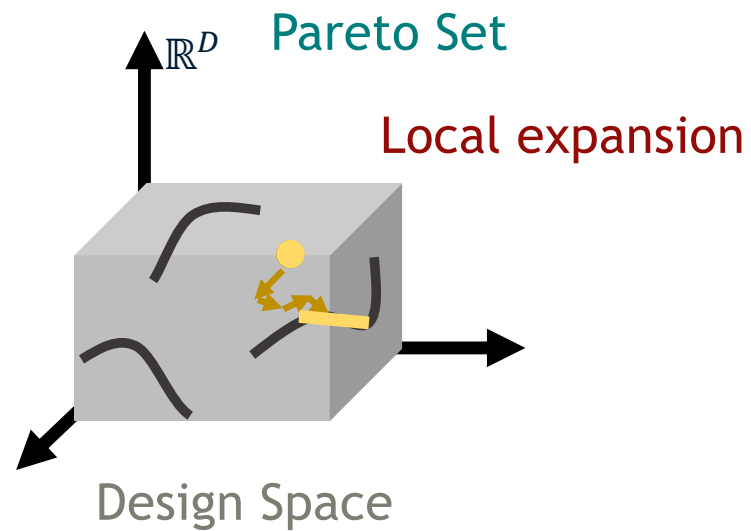


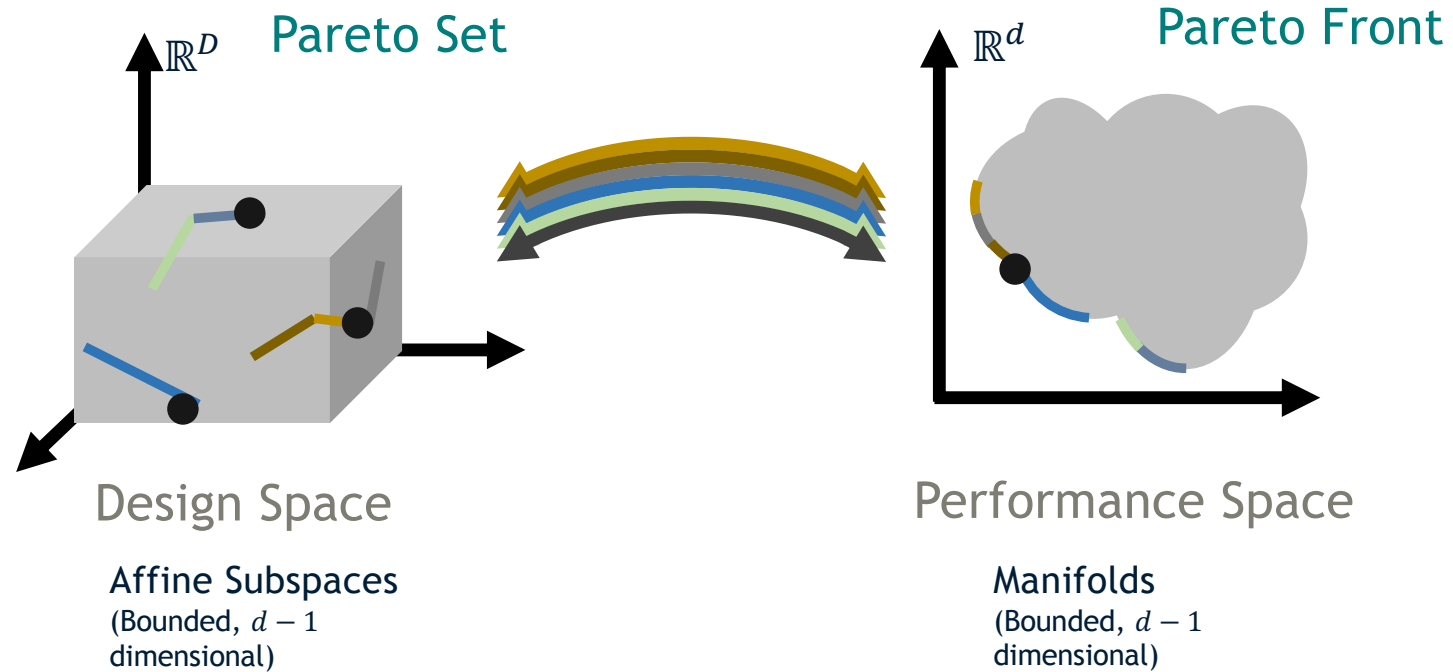
Figure 13: Population at generation 100.

Source: Deb et al 2002

# Finding the Full (Continuous) Front



# Set of Manifolds



To finish Complementary materials online

**CHALLENGE #4** Could you use PYMOO?

[https://pymoo.org/getting\\_started/preface.html](https://pymoo.org/getting_started/preface.html)

<https://colab.research.google.com/drive/1EjHgXfbP21WvgWmGPGE7CRh2w42ejCJz>

# In Practice



# Weighted sum method

The weighted sum method simply combines multiple objective functions by adding them together with some weights on each function.

An example is shown here,

$$f_{\text{obj}}(x) = \alpha g(x) + \beta h(x)$$

where  $g(x)$  and  $h(x)$  are two objectives we're trying to optimize simultaneously

and  $\alpha$  and  $\beta$  are weighting variables for each of the objective functions.

# Weighted sum method

If you cared much more about the value of  $\mathbf{g}(\mathbf{x})$ , you would set  $\alpha$  to be larger than  $\beta$ .

However, it's not that simple as  $\mathbf{g}(\mathbf{x})$  and  $\mathbf{h}(\mathbf{x})$  might have radically different magnitudes.

If  $\mathbf{g}(\mathbf{x})$  is on the order 5,000 while  $\mathbf{h}(\mathbf{x})$  is on the order 0.1 ,

this would cause the optimizer to prioritize changing the design to make  $\mathbf{g}(\mathbf{x})$  smaller.

# Weighted sum method

An example of this is if we're trying to optimize the structural weight (large magnitude) and the coefficient of drag  $C_D$  (small value) for an aircraft:

- $f_{\text{obj}}(\mathbf{x}) = f_{\text{structural weight}}(\mathbf{x})[\text{kg}] + f_{C_D}$

If you care about both objectives approximately equally, you should scale both objective functions by an appropriate amount to make them approximately the same magnitude. The general form looks like this:

- $f_{\text{obj}}(\mathbf{x}) = \alpha \frac{g(\mathbf{x})}{g_0} + (1 - \alpha) \frac{h(\mathbf{x})}{h_0}$

where  $g_0$  and  $h_0$  are those magnitude scalars.

In the previously mentioned example, this might look like:

- $f_{\text{obj}}(\mathbf{x}) = \alpha \frac{f_{\text{structural weight}}(\mathbf{x})[\text{kg}]}{2700[\text{kg}]} + (1 - \alpha) \frac{f_{C_D}}{0.025}$

# Epsilon constraint method

- The epsilon-constrained ( $\epsilon$ -constrained) method is another way to perform multiobjective optimization where one objective function is minimized by the optimizer while the other objective functions are constrained to specific values. This ensures that a given design is optimal is one objective for a given value in the other objective functions.
- The  $\epsilon$ -constrained method is the preferred way to create Pareto fronts because it produces a more robust curve and avoids rare situations where the Pareto front is non-convex that may be troublesome if using the weighted-sum method.

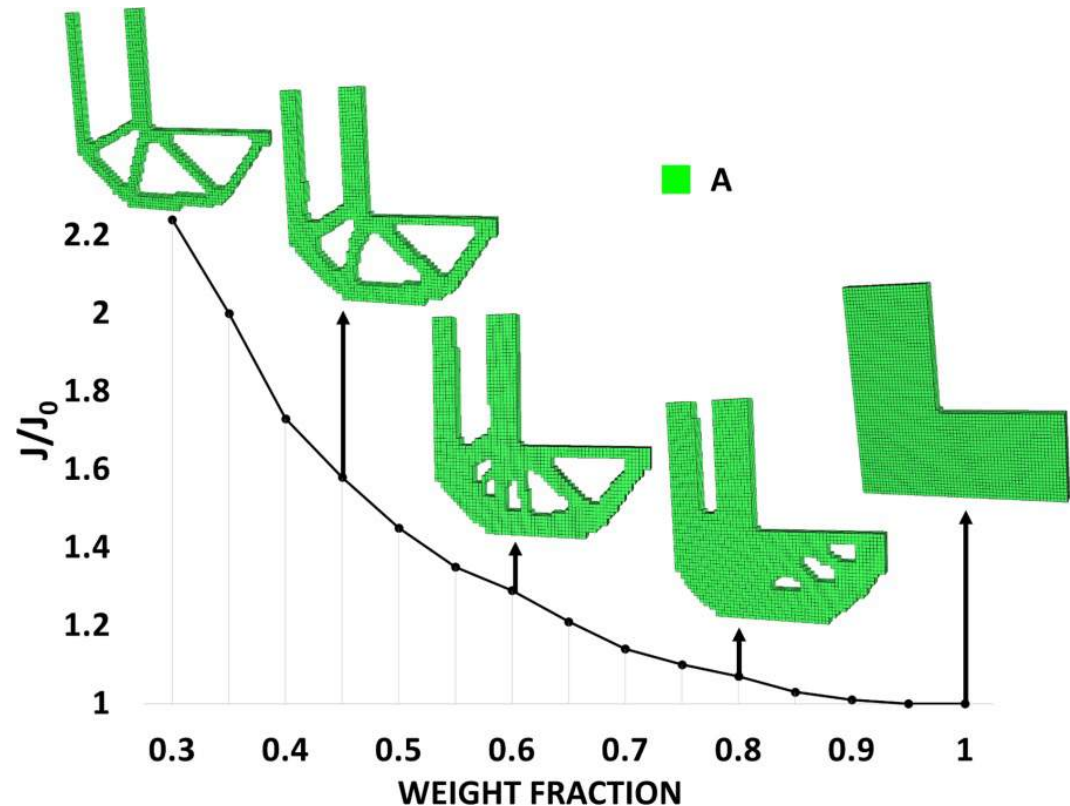
```
prob.model.add_subsystem('paraboloid_1',  
    om.ExecComp('g = (x-3)**2'), promotes=['*'])  
prob.model.add_subsystem('paraboloid_2',  
    om.ExecComp('h = (x+1)**2 + 3*x'),  
    promotes=['*'])  
prob.model.add_subsystem('objective',  
    om.ExecComp('f = alpha * g + beta * h'),  
    promotes=['*'])
```

# Pareto Compliance vs volfrac

A topology optimization problem based on the power-law approach, where the objective is to minimize compliance can be written as

$$\left. \begin{array}{l} \min_{\mathbf{x}}: c(\mathbf{x}) = \mathbf{U}^T \mathbf{K} \mathbf{U} = \sum_{e=1}^N (x_e)^p \mathbf{u}_e^T \mathbf{k}_0 \mathbf{u}_e \\ \text{subject to: } \frac{V(\mathbf{x})}{V_0} = f \\ \quad : \mathbf{K} \mathbf{U} = \mathbf{F} \\ \quad : 0 < \mathbf{x}_{\min} \leq \mathbf{x} \leq 1 \end{array} \right\}, \quad (1)$$

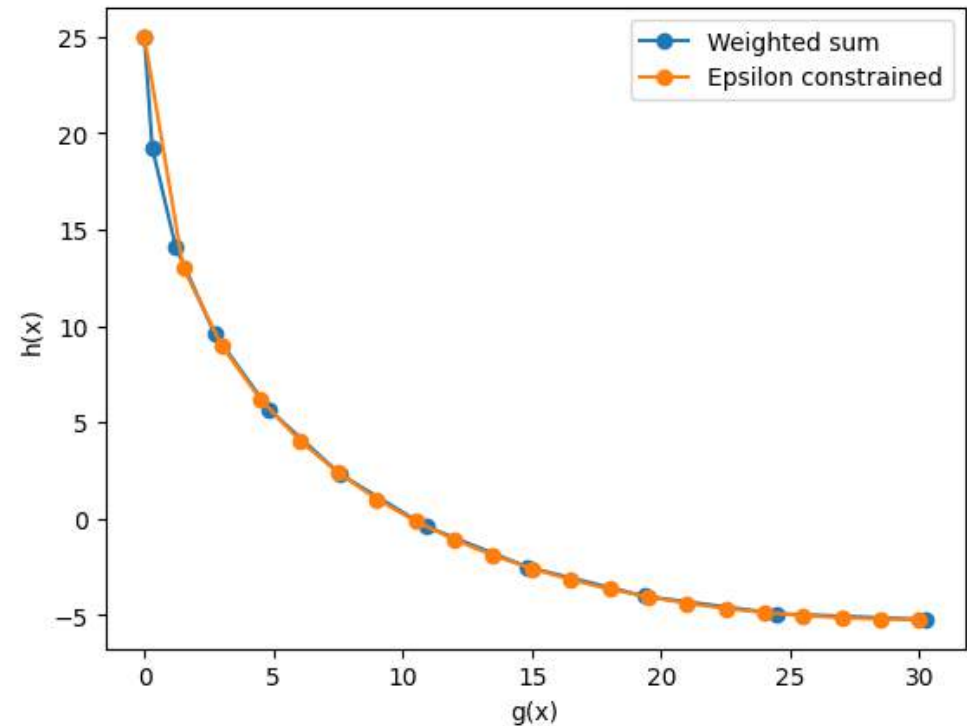
where  $\mathbf{U}$  and  $\mathbf{F}$  are the global displacement and force vectors, respectively,  $\mathbf{K}$  is the global stiffness matrix,  $\mathbf{u}_e$  and  $\mathbf{k}_e$  are the element displacement vector and stiffness matrix, respectively,  $\mathbf{x}$  is the vector of design variables,  $\mathbf{x}_{\min}$  is a vector of minimum relative densities (non-zero to avoid singularity),  $N$  ( $= \text{nelx} \times \text{nely}$ ) is the number of elements used to discretize the design domain,  $p$  is the penalization power (typically  $p = 3$ ),  $V(\mathbf{x})$  and  $V_0$  is the material volume and design domain volume, respectively and  $f$  (**volfrac**) is the prescribed volume fraction.



# Epsilon constraint method

This method also allows you to easily control the spacing for a Pareto front. Given a spread of  $g(x)$  values for constraints, you can directly set the spacing produced by a series of optimizations.

For a bi-objective optimization, a good way to know what bounds to use for the constraint values are to run two optimizations first: one unconstrained using the first objective and the other unconstrained with the second objective only. From those two optimizations, the resulting values of the second and first objectives, respectively, could be your constraint limits.



The page features several decorative blue elements: a large 'X' shape formed by two intersecting lines, a small triangle in the top left corner, and a larger right-angled triangle in the bottom left corner.

# MERCI

Institut Supérieur de l'Aéronautique et de l'Espace

10, avenue Edouard Belin – BP 54032

31055 Toulouse Cedex 4 – France

T +33 5 61 33 80 80

[www.isae-supaero.fr](http://www.isae-supaero.fr)

