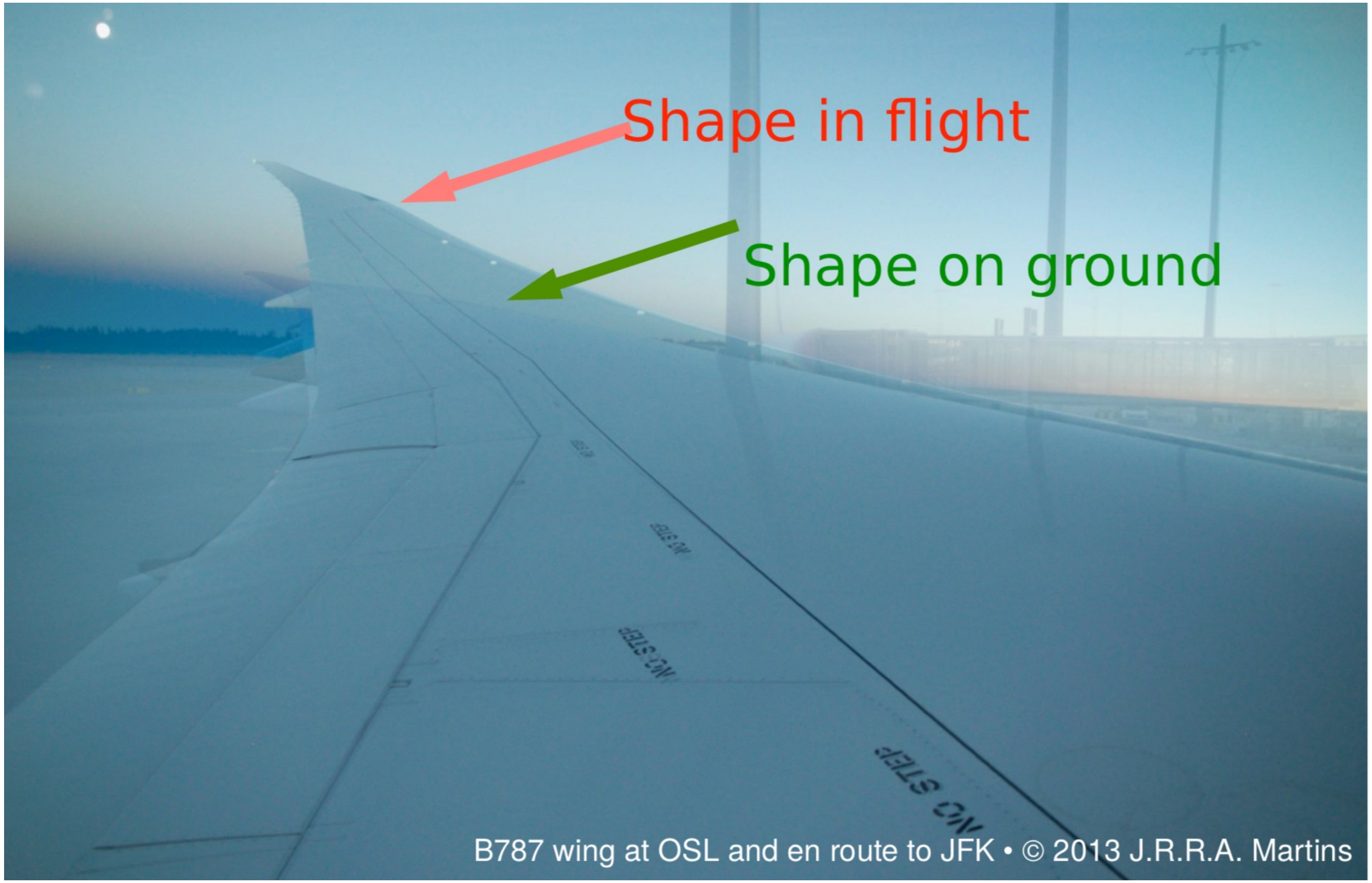
Multidisciplinary Design Optimization

Coupled problem

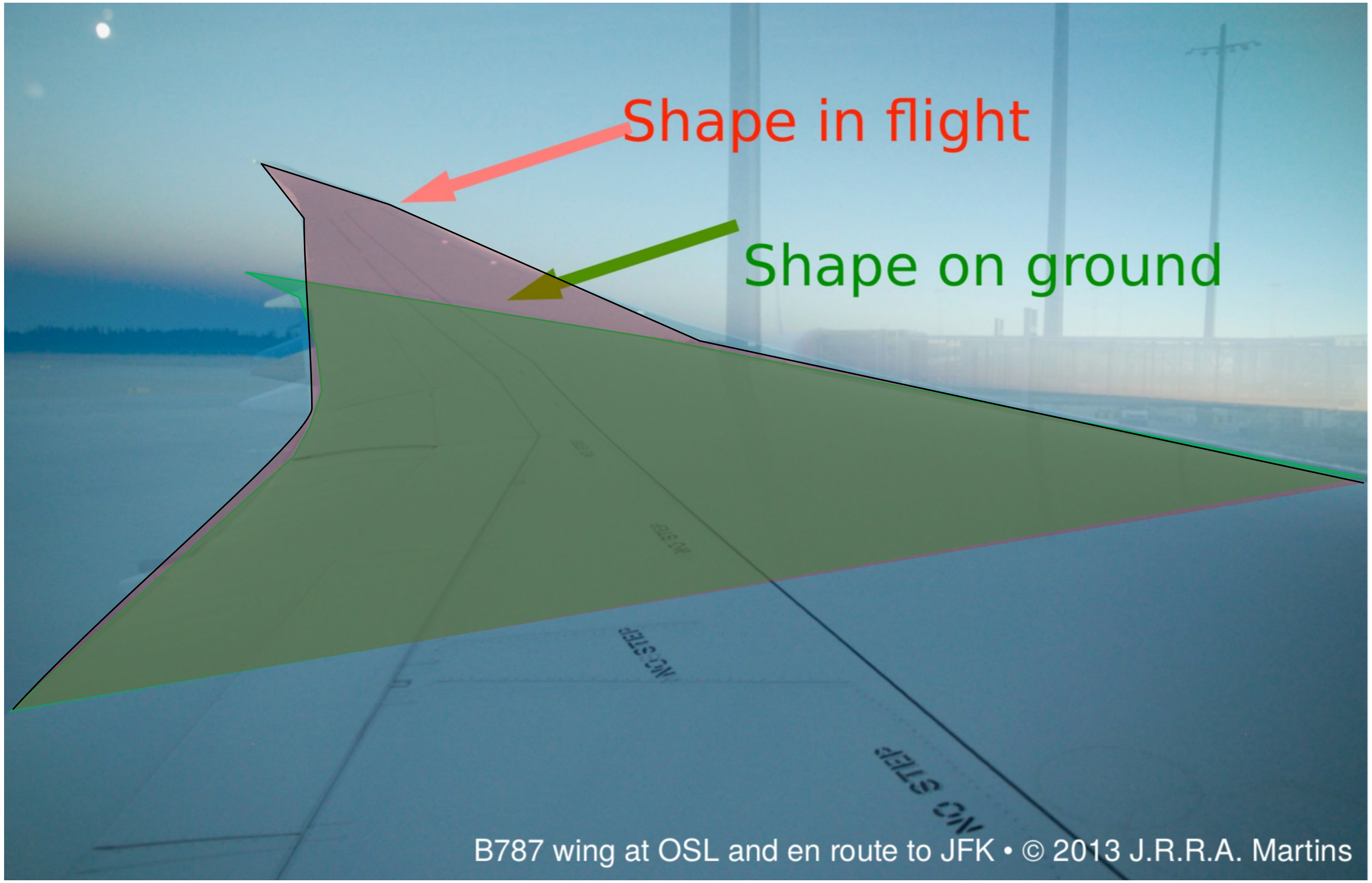
Multidisciplinary Design Optimization

- Multidisciplinary **Design** Optimization (MDO) focuses on solving optimization problems spanning across multiple interacting disciplines

Coupled problem

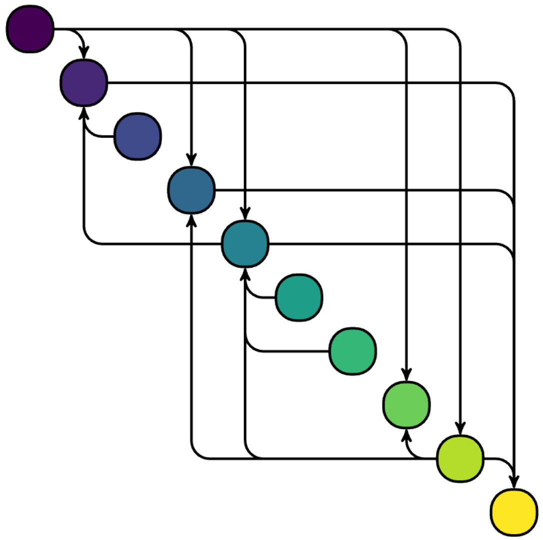


Coupled problem



Interdisciplinary Compatibility

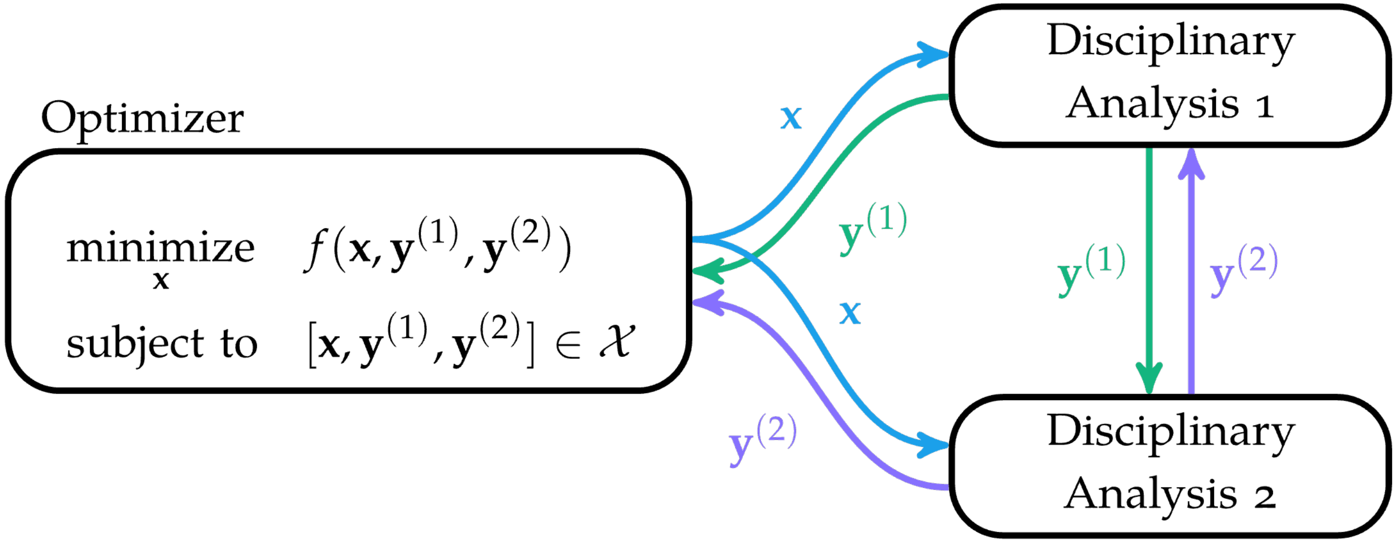
- If there are dependency cycles, then no topological ordering exists
- A general approach is iterative techniques such as the Gauss-Seidel method
- Depending on the nature of the problem, iterative methods can converge slowly



Interdisciplinary Compatibility

No MDO without MDA

- If there are multiple disciplines, then dependencies have to be considered

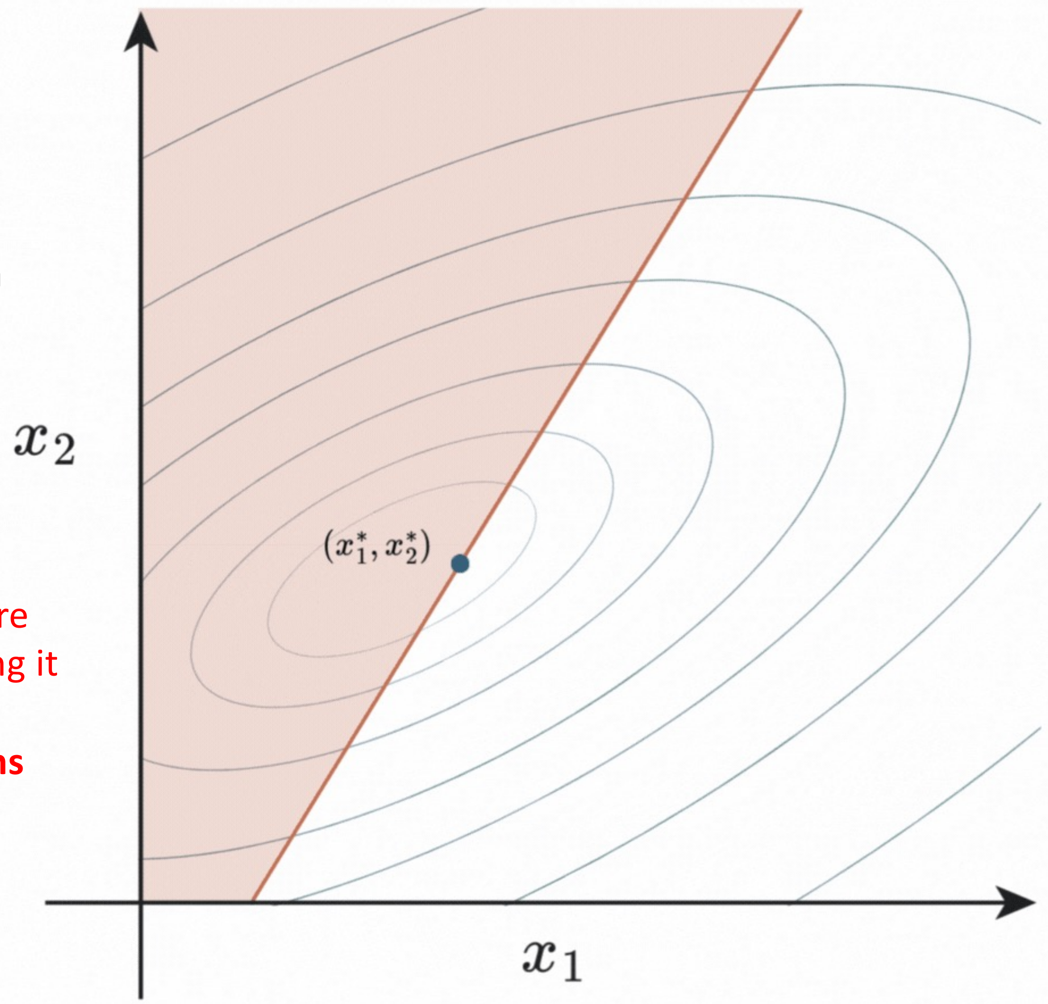


Illustrative scenario

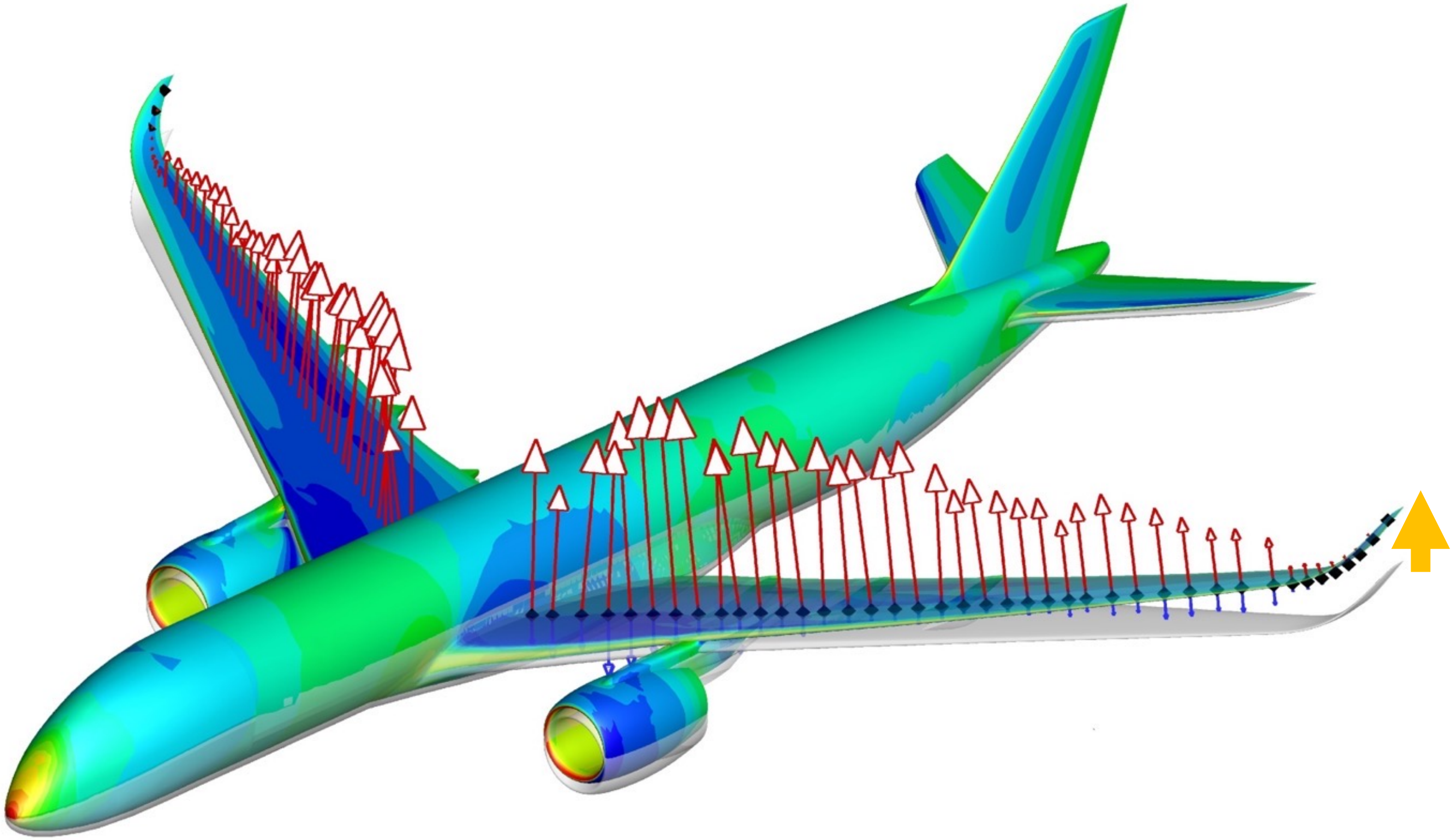
•Optimal values (x_1*, x_2*) lie near the center of an objective function's contours, constrained by a red region.
•In **sequential optimization**, only one variable is adjusted at a time, failing to follow a feasible descent path along the constraint.
•In **simultaneous optimization**, both variables are adjusted together, enabling a feasible descent path toward the true optimum.

Industry practice often follows **sequential optimization**, where each discipline optimizes its part independently before passing it to the next.
•**Sequential optimization** typically leads to **suboptimal designs** due to lack of coordination across disciplines.
Two variables/disciplines: x_1 and x_2.

https://medium.com/optimise-consultancy/what-is-multidisciplinary-design-optimisation-88e6e6b933b3



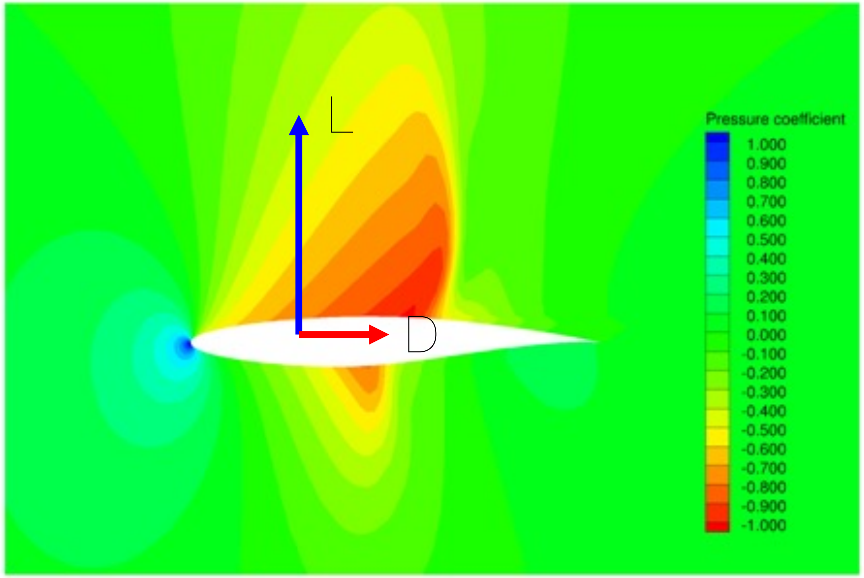
What is an MDA ? Static Aeroelasticity for example?



Source: DLR

But first, what is Disciplinary Optimization?

Example: Aerodynamics (L/D max)

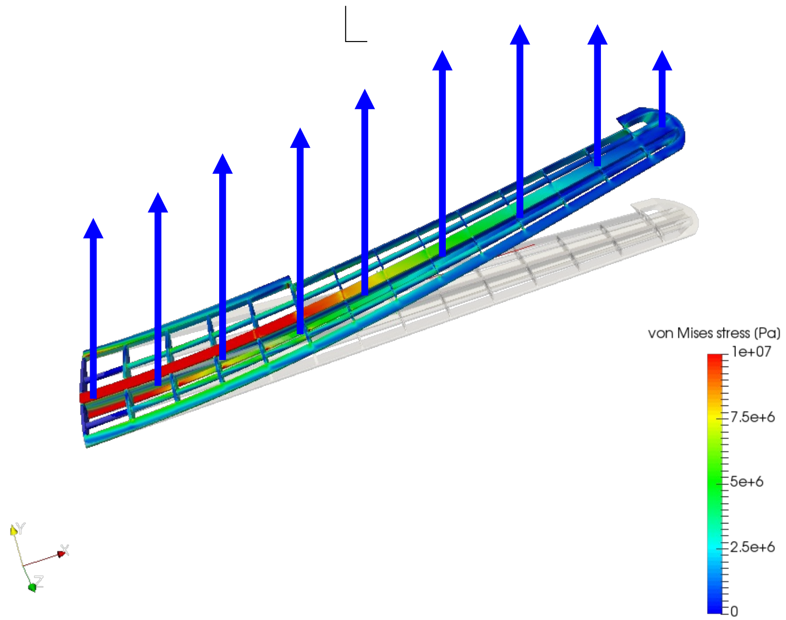


Source: NLR

Minimize D
w.r.t. shape, α
Subject to L = W

What is Disciplinary Optimization (2)?

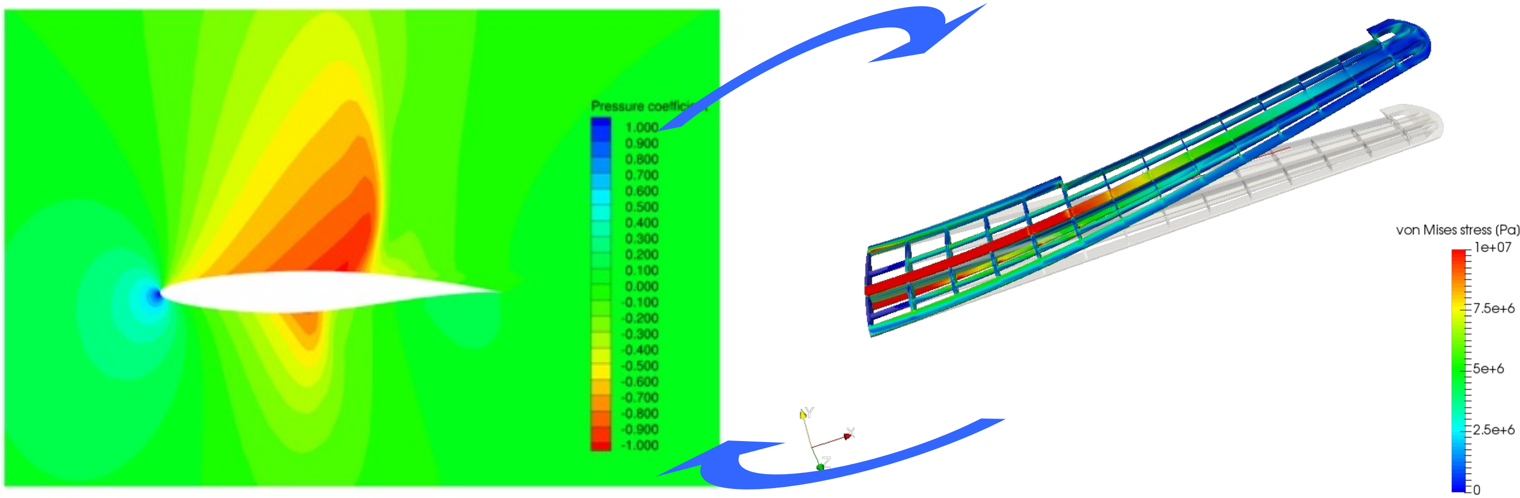
Another example: Structures



Source: simscale.com

Minimize Mass
w.r.t. thicknesses
Subject to $\sigma \leq \sigma y$

However… Disciplines are not isolated:



Structural deformation of wing → changes in the shape exposed to airflow

Changes in the shape exposed to airflow → changes in the aerodynamic loads

Then, how do we solve the complete system?

Nodal forces

Structure Analysis

Nodal displacements

The solution of the complete system is the set of displacements and forces that “satisfy” this loop

Load Transfer

Rendall, T. C. S., & Allen, C. B. (2008). Unified fluid–structure interpolation and mesh motion using radial basis functions. International Journal for Numerical Methods in Engineering, 74(10), 1519-1559.

Radial Basis Function displacement interpolation (same for y and z)

$$u_x = \sum_{i=1}^{N_s} \alpha_i^x \phi(\|\mathbf{x} - \mathbf{x_i}\|) + \gamma_0^x + \gamma_x^x x + \gamma_y^x y + \gamma_z^x z$$

Displacement interpolation matrix

$$u_a = H u_s$$

Loads using principle of virtual work

$$f_s = H^T f_a$$

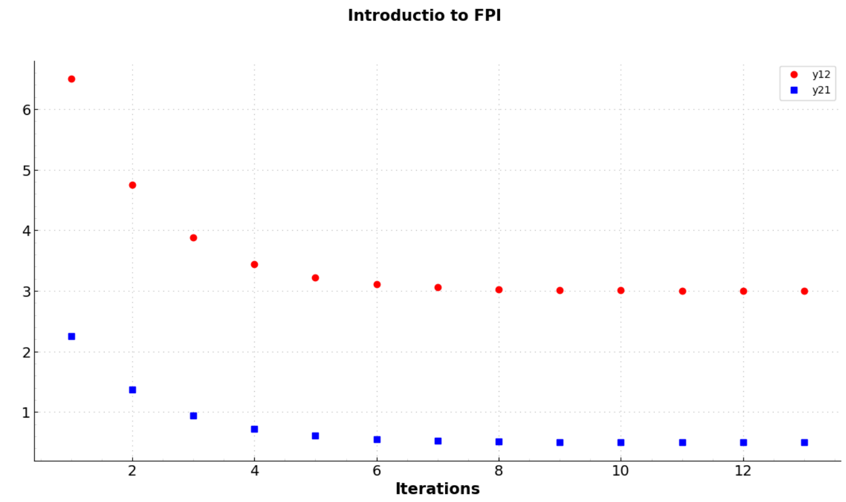
Forces on aero grid points

Aerodynamic Analysis

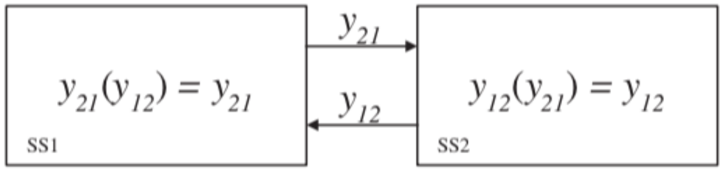
Aero grid displacements

Multi-Disciplinary Analysis

- Computation of the state variables at equilibrium for given x and z
  - Generally computed using a fixed-point algorithm (Jacobi or Gauss-Seidel)
  - Or a root-finding method (Newton-Raphson)



$y21(y12)=0.25\cdot y12-0.25$
$y12(y21)=2+2\cdot y21$



(Step 0) choose initial guess y_{12}^0, set $i = 0$

(Step 1) $i = i + 1$

(Step 2) $y_{21}^i = y_{21}(y_{12}^{i-1})$

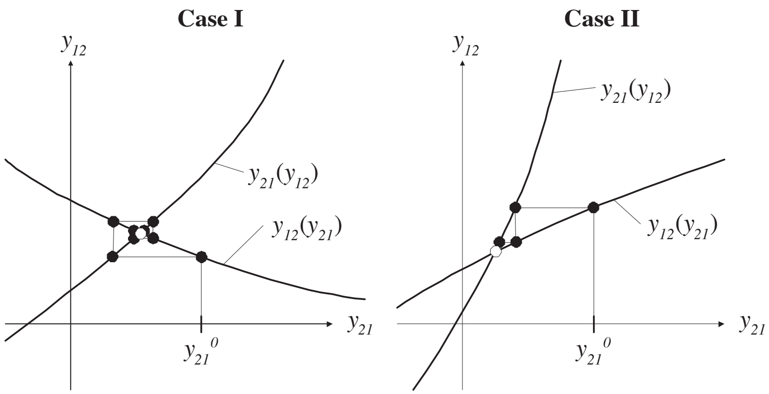
(Step 3) $y_{12}^i = y_{12}(y_{21}^i)$

(Step 4) if $|y_{12}^i - y_{12}^{i-1}| < \varepsilon$ stop, otherwise go to **(Step 1)**

$\boxed{\text{Check the default tolerence}}$

Fixed Point Iteration Convergence

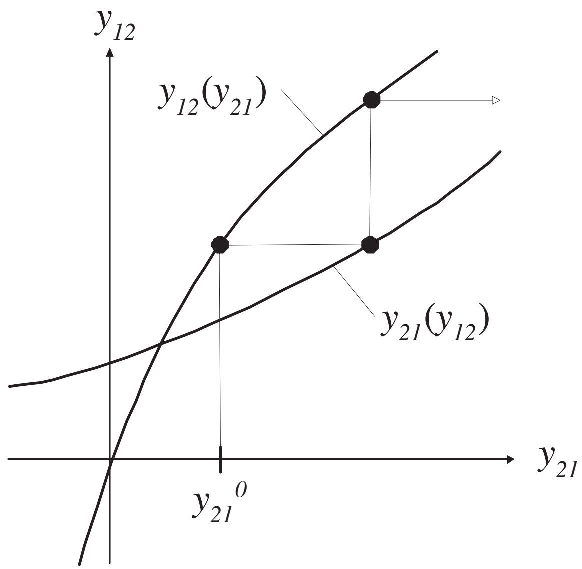
Developed proof of new convergence condition form



$$\left|\frac{\partial y_{21}(y_{21})}{\partial y_{21}}\right| > \left|\frac{\partial y_{12}(y_{21})}{\partial y_{21}}\right| \Leftrightarrow \left|\frac{\partial y_{12}(y_{12})}{\partial y_{12}}\right| > \left|\frac{\partial y_{21}(y_{12})}{\partial y_{12}}\right|$$

$$\mathbf{y_p} = \mathbf{y}(\mathbf{y_p})$$

Fixed Point Iteration Divergence

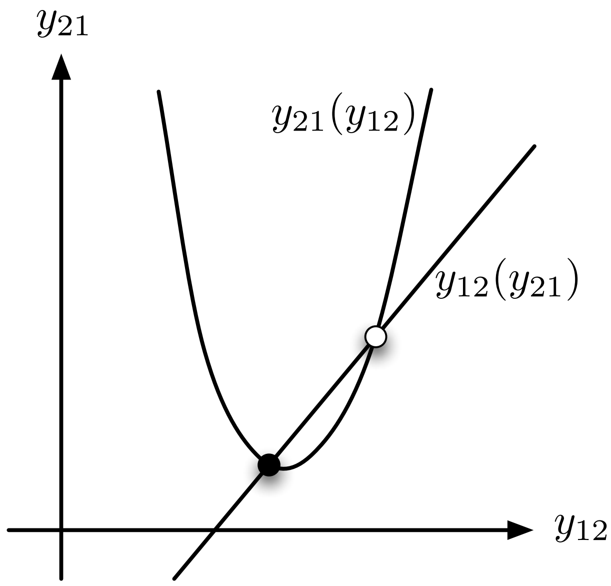


Multiple Fixed Points

FPI:

Unknown if a repelling fixed point would have led to a better solution.

- ● Attractive fixed point
- ○ Repelling fixed point



IF POSSIBLE

- Use inversion
- or Gauss Seidel

$y21 = 0.25 \cdot y12 - 0.25$
$y12 = 2 + 2 \cdot y21$

$-0.25 \cdot y12 + y21 = -0.25$
$y12 - 2 \cdot y21 = 2$

$A*y=b$

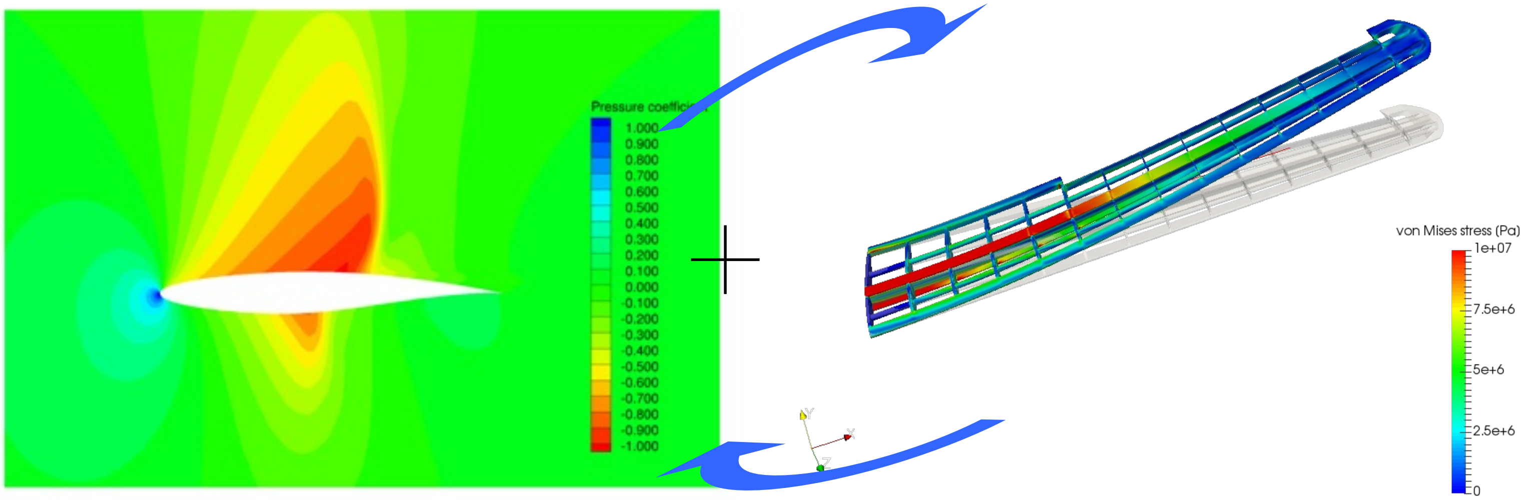
```
>> A=[-0.25 1; 1 -2]

A =

   -0.2500    1.0000
    1.0000   -2.0000

>> b=[-0.25;2]

b =

   -0.2500
    2.0000

>> y=A\b

y =

    3.0000
    0.5000
```

```
# Define sympy symbols
y12, y21 = sp.symbols('y12 y21')

# Define the system of equations
eq1 = y21 - (0.25 * y12 - 0.25)
eq2 = y12 - (2 + 2 * y21)

# Find the exact solution for comparison
solution = sp.solve((eq1, eq2), (y12, y21))
print("Exact Solution:")
print(f"y12 = {solution[y12]}")
print(f"y21 = {solution[y21]}")
```

So start…

https://colab.research.google.com/drive/1Spc5Sh1u0A91Ryk05ep1Xfhy7x7ieKbg#scrollTo=7ab_CXxI3Gmp

So, we need to analyze BOTH disciplines at the SAME TIME



Minimize D, or Mass, {or a combination of D and Mass}
w.r.t. shape, α, thicknesses
Subject to:

$$L = W$$

$$\sigma \leq \sigma_y$$

In practice, how do we solve that problem?

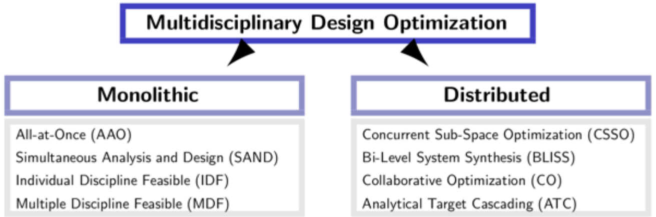
One possible approach: MultiDisciplinary Feasible (MDF, probably the most intuitive one…)

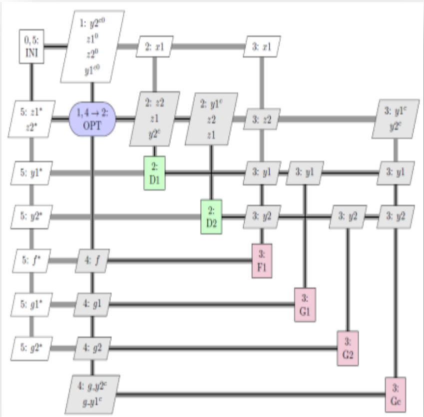
Steps:

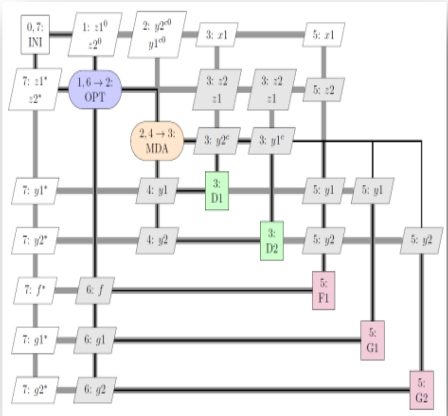
1. Start from a set of particular design variables: shape, α, thicknesses
2. Solve the complete system (with all the interactions) for these values
3. Evaluate objective function and constraints
4. From these values, the optimizer proposes a new set of design variables.

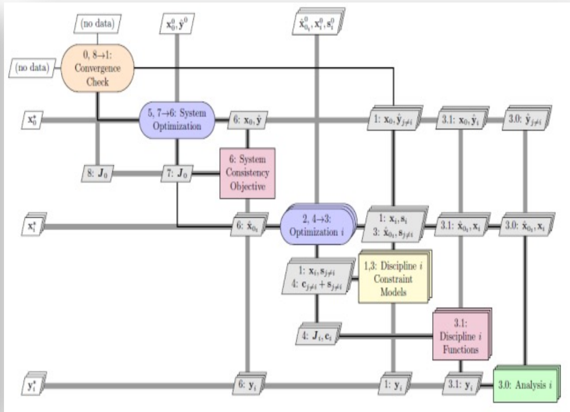
These steps are repeated until the optimum is reached.

Assembling MDO systems

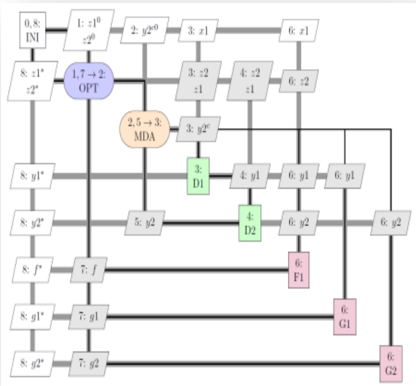


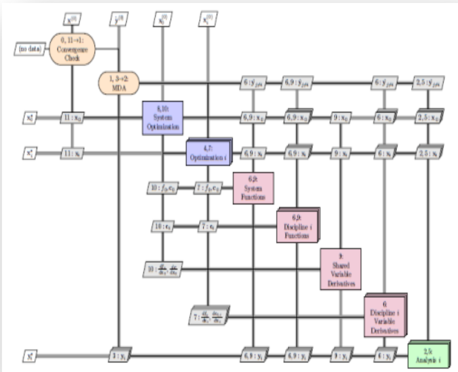


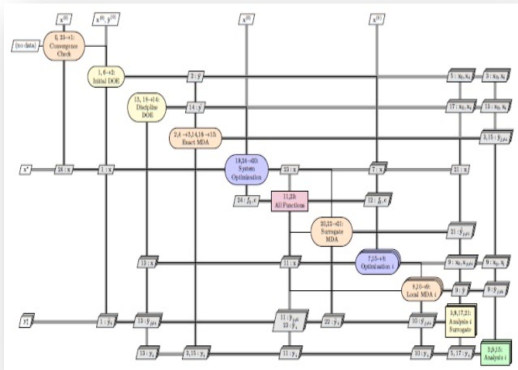




MDF Multidisciplinary Feasible approach— a complete analysis is performed at every optimization iteration. Also known as the All-in-One approach.

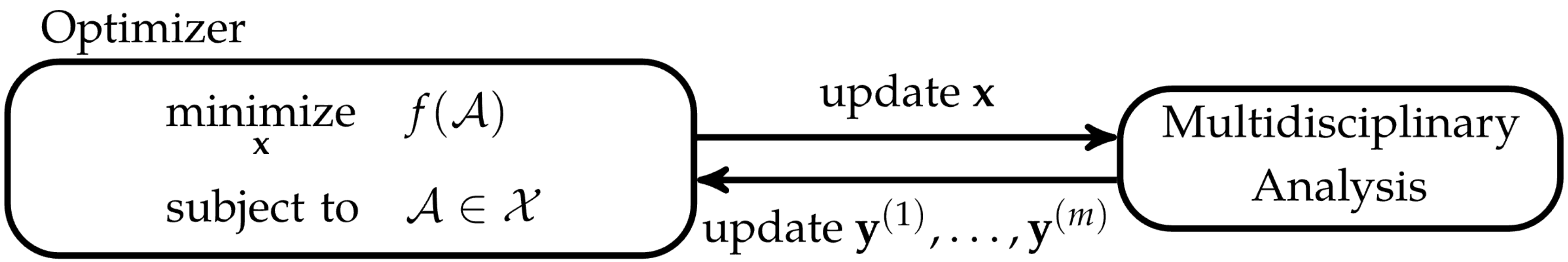






Multidisciplinary Design Feasible

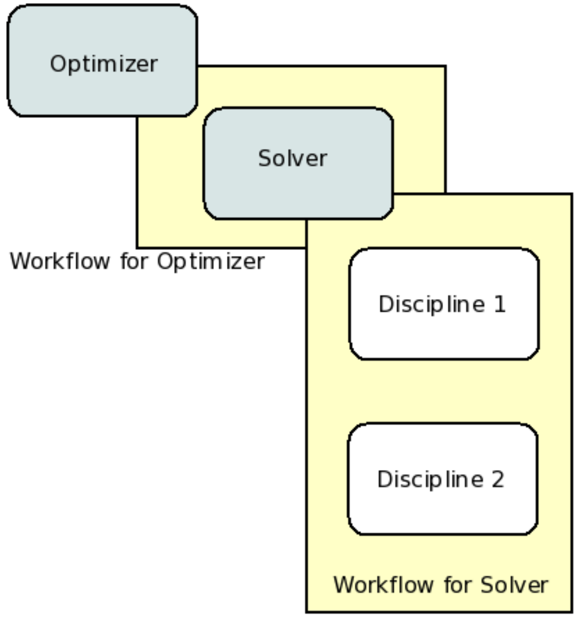
- The multidisciplinary design feasible architecture structures the MDO problem such that standard optimization algorithms can be directly applied to optimize the design variables



$$\begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} && f\left(\mathbf{x}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(m)}\right) \\ &\text{subject to} && \left[\mathbf{x}, \mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(m)}\right] \in \mathcal{X} \end{aligned} \longrightarrow \begin{aligned} &\underset{\mathbf{x}}{\text{minimize}} && f(\text{MDA}(\mathbf{x})) \\ &\text{subject to} && \text{MDA}(\mathbf{x}) \in \mathcal{X} \end{aligned}$$

Multidisciplinary Feasible (MDF)

- The MDF architecture is the most intuitive for engineers
- The optimization problem formulation is identical to the single discipline case, except the disciplinary analysis is replaced by an MDA



Illustrative example: the Sellar problem

2 disciplines involved

Variables: x_1, y_1, y_2, z_1, z_2

We'll see later what are the differences between these variables …

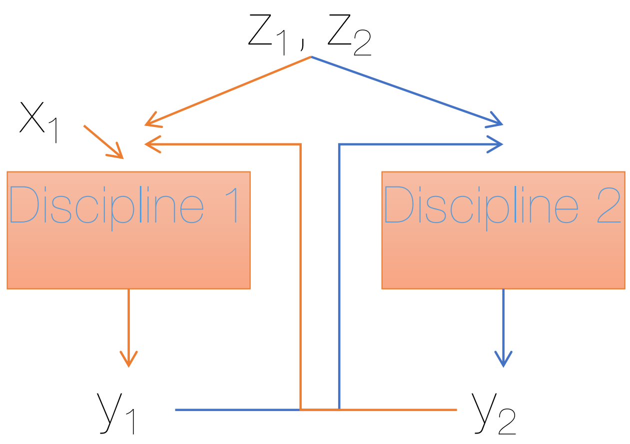
$$\begin{gathered}
\text{minimize } x_1^2 + z_2 + y_1 + \exp(-y_2) \\
\text{with respect to } z, x \text{ or } (z_1,\ z_2,\ x_1) \\
\text{subject to:} \\
3.16 - y_1 \leqslant 0 \\
y_2 - 24 \leqslant 0 \\
-10 \leqslant z_1 \leqslant 10 \\
0 \leqslant z_2 \leqslant 10 \\
0 \leqslant x_1 \leqslant 10
\end{gathered}$$

$$\begin{gathered}
\text{Discipline 1}: y_1(z_1, z_2, x_1, y_2) = z_1^2 + x_1 + z_2 - 0.2y_2 \\
\text{Discipline 2}: y_2(z_1, z_2, y_2) = \sqrt{y_1} + z_1 + z_2
\end{gathered}$$

Sellar, R. S., Batill, S. M., and Renaud, J. E., "Response Surface Based, Concurrent Subspace Optimization for Multidisciplinary System Design", 34th Aerospace Sciences Meeting and Exhibit, Aerospace Sciences Meetings, 1996.

Illustrative example: the Sellar problem

- Design variables: z_1, z_2, x_1 to minimize the objective
- Shared (or global) variables: z_1, z_2
- Local variable: x_1
- Coupling variables: y_1, y_2



$$\boxed{\begin{array}{c}\text{minimize } x_1^2 + z_2 + y1 + e^{-y_2}\\ \text{with respect to } z_1, z_2, x_1\end{array}}$$

subject to:

$$\frac{y_1}{3.16} - 1 \geq 0$$
$$1 - \frac{y_2}{24} \geq 0$$
$$-10 \leq z_1 \leq 10$$
$$0 \leq z_2 \leq 10$$
$$0 \leq x_1 \leq 10$$

$$\boxed{\begin{array}{c}\text{Discipline 1: } y_1(z_1, z_2, x_1, y_2) = z_1^2 + x_1 + z_2 - 0.2y_2\\ \text{Discipline 2: } y_2(z_1, z_2, y_1) = \sqrt{y_1} + z_1 + z_2\end{array}}$$

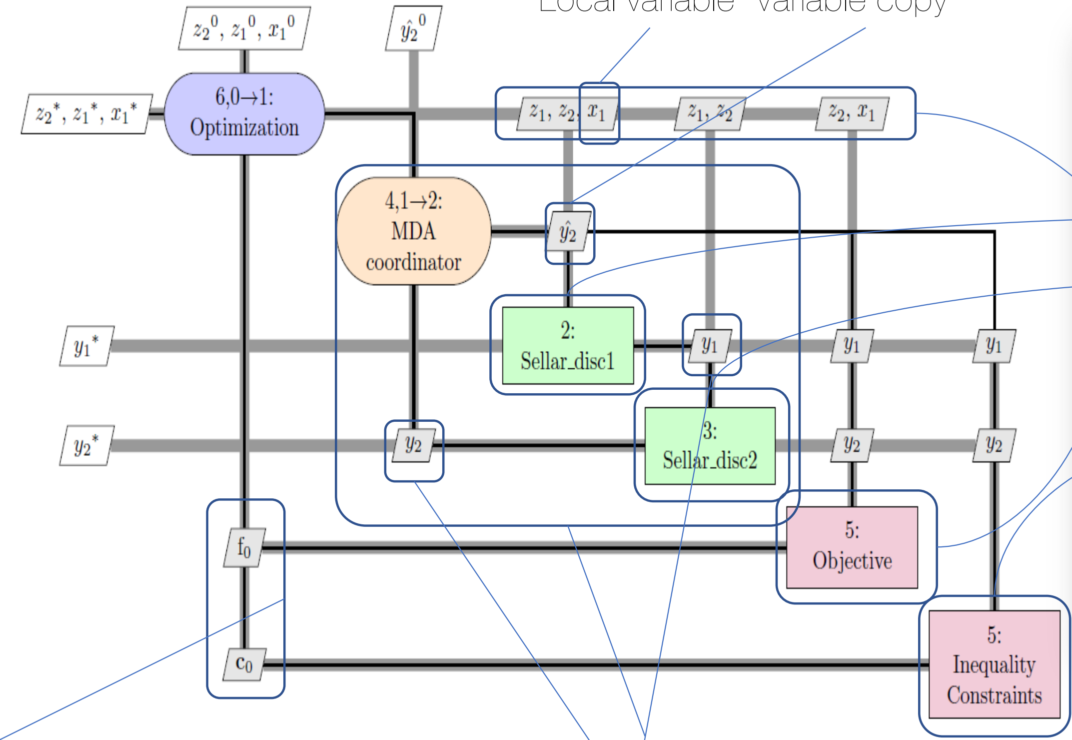
Multidisciplinary analysis (MDA) consists in solution of the following equations

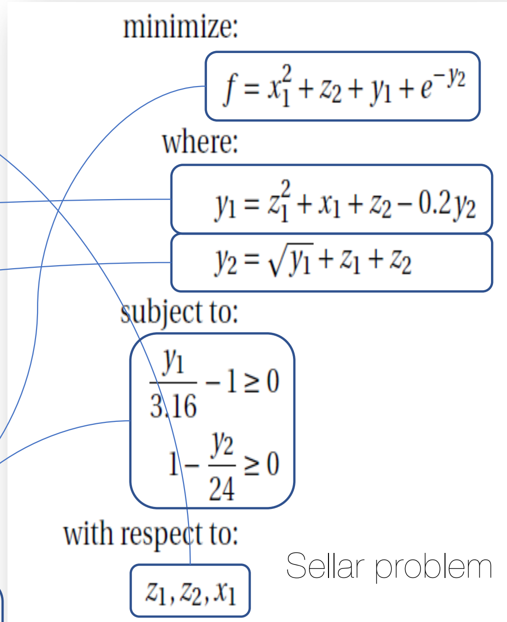
$R_1 = 0$ ➔ y_1 solutions

$R_2 = 0$ ➔ y_2 solutions

MDF illustration on the Sellar problem:
MDF – Gauss-Seidel variant

Local variable Variable copy





Sellar problem

Coordinated MDA system with sequential discipline execution

Coupling variables

Objective and constraints are sent to optimizer only after MDA coordinator has converged (y_2=^y_2)

Sellar Problem using CasADi

https://colab.research.google.com/drive/1-qdgy0EKoe5qhFXzjSiDxLh39oWfbpyb#scrollTo=2LQITUPCibyA

Execute the scenario

Then, we execute the MDO scenario with the inputs of the MDO scenario as a dictionary. In this example, the gradient-based *SLSQP* optimizer is selected, with 10 iterations at maximum:

```
scenario.execute(input_data={"max_iter": 10, "algo": "SLSQP"})
```

Out:

```
    INFO - 13:51:17:       Number of calls to the objective function by the optimizer: 7
    INFO - 13:51:17:    Solution:
    INFO - 13:51:17:       The solution is feasible.
    INFO - 13:51:17:       Objective: 3.1833939516400456
    INFO - 13:51:17:       Standardized constraints:
    INFO - 13:51:17:          c_1 = 5.764277943853813e-13
    INFO - 13:51:17:          c_2 = -20.244722233074114
    INFO - 13:51:17:          y_1 = [2.06834549e-15]
    INFO - 13:51:17:          y_2 = [1.98063788e-15]
    INFO - 13:51:17:       Design space:
    INFO - 13:51:17:       +------+-------------+-------------------+-------------+-------+
    INFO - 13:51:17:       | name | lower_bound |       value       | upper_bound | type  |
    INFO - 13:51:17:       +------+-------------+-------------------+-------------+-------+
    INFO - 13:51:17:       | x    |      0      |         0         |      10     | float |
    INFO - 13:51:17:       | z[0] |     -10     | 1.977638883463326 |      10     | float |
    INFO - 13:51:17:       | z[1] |      0      |         0         |      10     | float |
    INFO - 13:51:17:       | y_1  |     -100    | 1.777638883462956 |     100     | float |
    INFO - 13:51:17:       | y_2  |     -100    | 3.755277766925886 |     100     | float |
    INFO - 13:51:17:       +------+-------------+-------------------+-------------+-------+
    INFO - 13:51:17: *** End MDOScenario execution (time: 0:00:00.128566) ***

{'max_iter': 10, 'algo': 'SLSQP'}
```

https://gemseo.readthedocs.io/en/5.1.0/examples/mdo/plot_sellar.html

GO DEEPER?

- https://openmdao.org/newdocs/versions/latest/basic_user_guide/multidisciplinary_optimization/sellar.html
- https://gitlab.com/pablonorczyk/notes-on-mdao
- **https://colab.research.google.com/github/OpenMDAO/OpenMDAO/blob/master/openmdao/docs/openmdao_book/basic_user_guide/single_disciplinary_optimization/first_optimization.ipynb**

Multidisciplinary Feasible (MDF)

- Advantages:
  - Intuitive procedure/no specialized knowledge required → Easy to incorporate existing models
  - Always return a system design that satisfies the consistency constraints, even if the optimization process is terminated early – good from a pratical engineering point of view

- Disadvantages:
  - Intermediate results do not necessarely satisfy the optimization constraints
  - Cannot be parallelized
  - Developing the MDA procedure with CSM/CFD might be time consuming*, if not already available

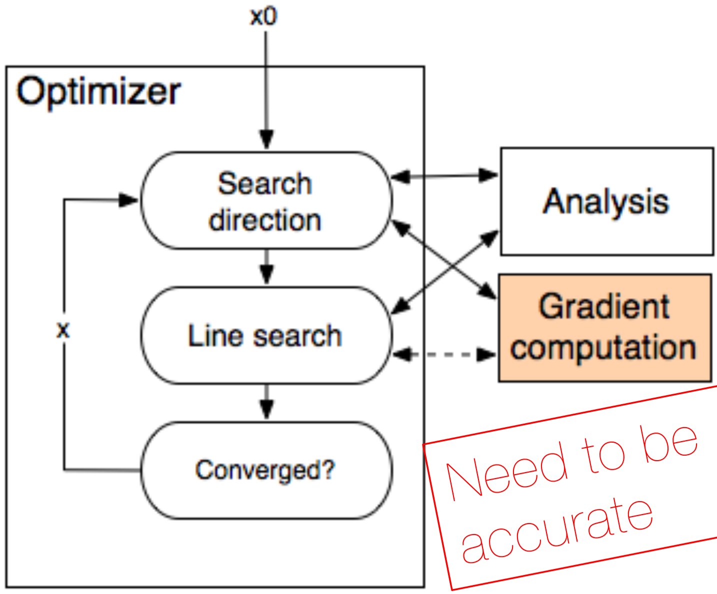
* Automatic mapping, postprocessing etc…

Gradients of the coupled system more challenging to compute

Optimizer solver

- Requirements
- Problem to solve

$$\begin{cases} \min f(x) \\ \text{wrt } x \in R^d \\ \text{st } g_i(x) \leq 0 \text{ for } i = 1, \ldots m \end{cases}$$



Need to be accurate

- Derivative Fee Optimizer (DFO)
  - Evolutionnary Strategies (ES)
  - Surrogate based Optimizer (SBO) or Bayesian Optimization (BO)
  - …
- Gradient based Optimizer

→ Computation of the derivatives of $f(x)$ and $g_i(x)$ to iterate and satisfy the KKT optimality conditions

→ OpenMDAO focus on computation of sensitivities (adjoint vs direct)

$$\frac{\partial f}{\partial x_i}, \frac{\partial g}{\partial x_i}, \frac{\partial h}{\partial x_i}$$

When to use gradient-free optimizers???

1. Very cheap models
2. When you can't compute derivatives

When to use gradient-free optimizers???

Noisy and discontinuous design space

(YOU DON 'T KNOW *a priori*)

When you have a "noisy" design space, it means that the outputs change rapidly for a small change in the inputs. This noise might be caused by computational or physical reasons.

If your model is either C0 or C1 discontinuous, gradient-free methods might make sense for you. C0 discontinuities mean that there are jumps in the design space. These might be caused by if-then conditions or discrete variables in your model or something else. C1 discontinuities mean that the derivative space is not smooth and continuous.

For example, what are the derivatives for a wind turbine having two or three blades? 2.2 or 2.9 blades are not an option, so that's inherently introduces a discontinuity. The derivative doesn't exist for discrete variables.

When to use gradient-free optimizers???

Multimodal problems

(YOU DON 'T KNOW *a priori*)

Gradient-free algorithms don't automatically solve multimodal problems better than gradient-based ones.

DIRECT, ISRES, particle swarm, and evolutionary methods.

Nelder-Mead and COBYLA are local algorithms in that they are not made to explore the global design space. I suggest COBYLA as default optimizer https://github.com/relf/cobyla

class COBYLA(maxiter=1000, disp=False, rhobeg=1.0, tol=None)

Constrained Optimization By Linear Approximation optimizer.
COBYLA is a numerical optimization method for constrained problems where the derivative of the objective function is not known.
Uses scipy.optimize.minimize COBYLA. For further detail, please refer to https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html

Parameters
- maxiter (int) – Maximum number of function evaluations.
- disp (bool) – Set to True to print convergence messages.
- rhobeg (float) – Reasonable initial changes to the variables.
- tol (Optional[float]) – Final accuracy in the optimization (not precisely guaranteed). This is a lower bound on the size of the trust region.