



OPTIMAL DESIGN

TRAINING LAB. 2-3

MULTI-OBJECTIVE OPTIMISATION

In order to illustrate all the basic issues related to the design of vehicles (passive) suspension systems, let us consider the simplified 2-degrees of freedom model (“quarter car vehicle model”) able to describe the vehicle dynamic response to an external excitation due to the road irregularity ξ_1 .

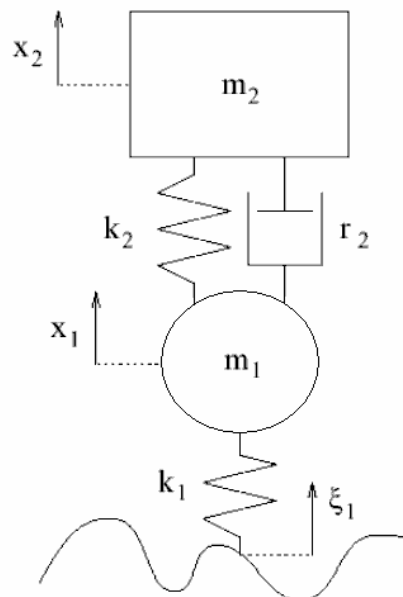


Fig. 1 quarter car vehicle model

The parameters of the model are:

- m_1 : unsprung mass
- m_2 : sprung mass
- k_1 : tyre radial stiffness
- k_2 : suspension equivalent stiffness (supposed linear)
- r_2 : suspension equivalent damping (supposed linear)
- ξ_1 : road irregularity

For the prescribed vehicle the suspension equivalent stiffness and damping (k_2, r_2) have to be determined for minimizing both the two objective functions, namely *discomfort* and *road holding*. Discomfort is defined as the standard deviation of the vertical acceleration in the cabin ($\sigma_{\ddot{x}_2}$), while road holding is defined as the standard deviation of the vertical force acting on the tyre (σ_{F_z}).

The expressions of the objective functions (with also the Working space) are:

$$\sigma_{\ddot{x}_2} = A \cdot \sqrt{\frac{(m_1 + m_2)}{m_2^2 r_2} k_2^2 + \frac{k_1 r_2}{m_2^2}}$$

$$\sigma_{F_z} = A \cdot \sqrt{\frac{(m_1 + m_2)^3}{m_2^2 r_2} k_2^2 - 2 \frac{m_1 k_1 (m_1 + m_2)}{m_2 r_2} k_2 + \frac{k_1 r_2 (m_1 + m_2)^2}{m_2^2} + \frac{k_1^2 m_1}{r_2}}$$

$$\sigma_{x_2 - x_1} = A \cdot \sqrt{\frac{m_1 + m_2}{r_2}}$$

with

$$A = \sqrt{\frac{A_b v}{2}}$$

Reference values for the parameters are reported in Tab. 1.

Tab. 1 parameters of the problem

Parameters	Value
A_b [m]	1.4e-5
v [m/s]	30
m_2 [kg]	230 (adapt to the chosen vehicle)
m_1 [kg]	30 (adapt to the chosen vehicle)
k_1 [N/m]	120000 (adapt to the chosen vehicle)

The field of variation of the design variables are reported in Tab.2

Tab. 2 field of variation of the design variables

Design variable	Field of variation
k_2 [N/m]	0 ÷ 50000
r_2 [Ns/m]	0 ÷ 5000

Requests (Lab. 2):

- Starting from a uniform grid in the design variables (k_2 , r_2) space, evaluate the correspondent objective functions space.
- Compute the Pareto-optimal set considering just two objective functions (Road Holding – Discomfort) in the space of the design variables and objective functions.
- **(OPTIONAL)** Compute the Pareto-optimal set considering all the three objective functions (Road Holding – Discomfort – Working Space) in the space of the design variables and objective functions.

Requests (Lab. 3):

- Compute the Pareto-optimal set in the design variables and objective functions space by applying the weighted sum method and the ϵ -constraints method.
- Determine the analytical expressions of the Pareto-optimal set and compare them with the ones obtained at the previous points.

Definition 2.7 (Pareto-optimal solution). *Given a Multi-Objective Programming problem with n_{dv} design variables and n_{of} objective functions, the Pareto-optimal (i.e. Non-dominated, Efficient or Non-inferior) (vector) solution \mathbf{x}_i satisfies the following conditions*

$\nexists \mathbf{x}_j :$

$$\begin{cases} f_k(\mathbf{x}_j) \leq f_k(\mathbf{x}_i) & k = 1, 2, 3, \dots, n_{of} \\ \exists l : f_l(\mathbf{x}_j) < f_l(\mathbf{x}_i) \end{cases}$$

If the above conditions hold, Pareto-optimal solutions will be denoted by $\mathbf{x}_i = \mathbf{x}_i^*$. Notice that the number of Pareto-optimal solutions approaches infinity. The whole set of Pareto-optimal solutions constitutes the *Pareto-optimal set*. Given a solution \mathbf{x}_i^* , if one tries to change it to improve one objective function, at least another objective function is worsened. For all non-Pareto-optimal solutions the value of at least one objective function f_l can be reduced without increasing the values of the other components. A typical characteristic of a MOP is the absence of a unique point that would optimize all objective functions simultaneously.

Let us assume that a MOP problem with two objective functions and n_{dv} design variables has been formulated and solved, i.e. the Pareto-optimal set has been computed. Fig. (2.5) refers to the two objective function space. The dashed area is the mapping of the design variable space \mathcal{X} into the objective function space (see Sect. 2.8), the Pareto-optimal solutions lie on the curve $b - c$. The solution a is not obviously Pareto-optimal, all the solutions inside the area abc can reduce both the objective functions f_1 and f_2 .

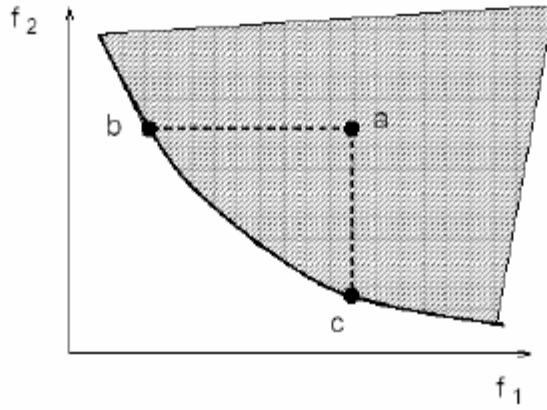


Fig. 2.5. Pareto-optimal set into the criterion space f_1, f_2 .

LAB. 2 TUTORIAL

Step 1: Define the uniform grid in the design variables space

Firstly we have to create two vectors of n equally spaced values for k_2 and r_2 :

$r_2 = [r_{2min} \dots r_{2max}]$; vector of n elements

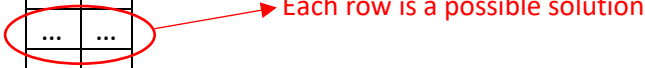
$k_2 = [k_{2min} \dots k_{2max}]$; vector of n elements

Then we have to define all the possible combinations among the design variables:

$DV = \text{combvec}(r_2, k_2)$; this function creates all the possible n^2 combinations of design variables and stores them in matrix DV .

Matrix DV will be a $n^2 \times 2$ matrix with the following structure:

k_2	r_2
...	...
...	...
...	...
...	...
...	...



Each row is a possible solution

Step 2: Calculate the objective functions

Once the design variables space has been defined the objective functions have to be calculated for each possible couple of values of the design variables.

We will end up with another $n^2 \times 2$ matrix that contains the values of the objective functions.

Step 3: Determine the Pareto-optimal set

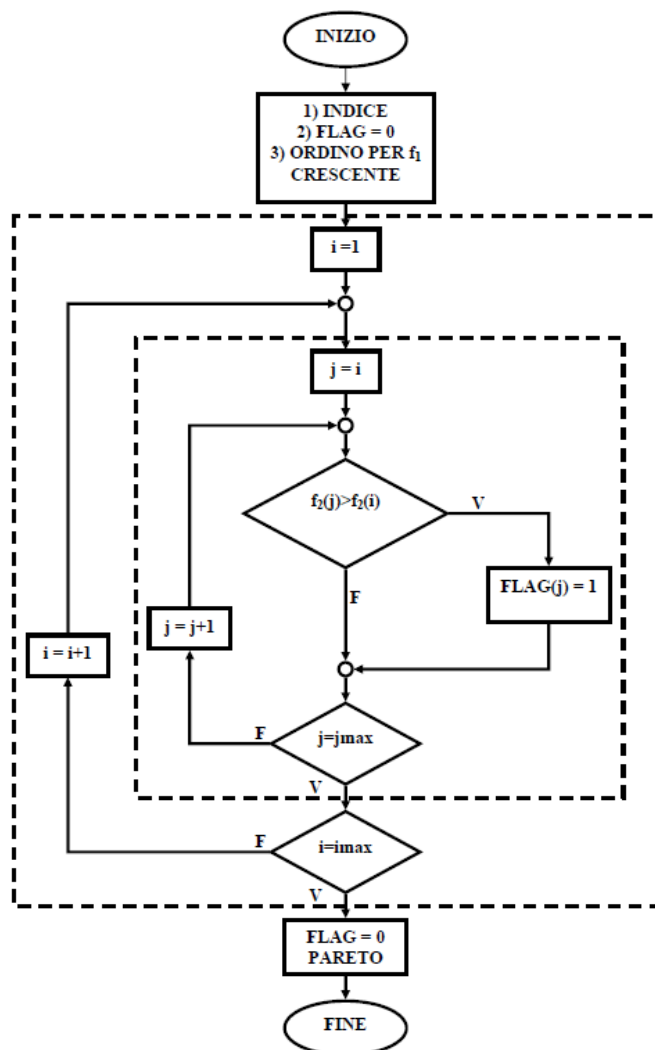
Now the Pareto-optimal solutions have to be extracted from the n^2 solutions obtained.

The sorting method has to be adopted:

- Create a single matrix by combining the matrices of the design variables and objective functions + an additional column of zeros denoted as flag:

k_2	r_2	fob1	fob2	Flag
...	0
...	0
...	0
...	0

- Sort the **entire matrix** with fob1 in ascending order (use matlab command "sortrows")
- Follow the block diagram



- At the end of the process lines with flag=0 are the Pareto-optimal solutions.
- Plot the Pareto-optimal solutions in the design variables and objective functions space.

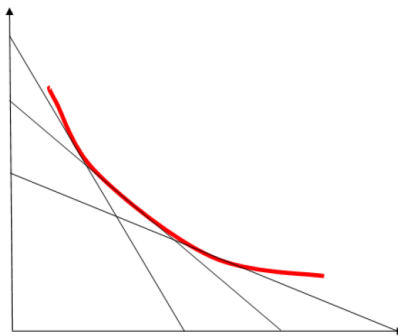
LAB. 3 TUTORIAL

WEIGHTED SUM METHOD

Define an aggregate objective function given by a weighted sum of the objective functions we want to minimize.

$$\phi(x) = \lambda_1 \cdot \frac{fob_1}{\max(fob_1)} + \lambda_2 \cdot \frac{fob_2}{\max(fob_2)}$$

The main idea is to vary the weights between 0 and 1 in order to find different solutions on the Pareto-optimal set:



NOTES:

- It is recommended to normalize the objective functions
- Choose λ_1 and λ_2 such that $\lambda_1 + \lambda_2 = 1$

In Matlab

`[x,fval,exitflag] = fminsearch(fun,x0)`

Simplex method

The matlab function “fun” will calculate the aggregate objective function $\phi(x)$.

NOTE: The function can be called in two ways:

- 1 Using global variables


```

global par
...
par=...
[x,fval,exitflag] =fminsearch(fun,x0,options)
...

function f=fun(x)
global par
...
f=...
end

```

2 Using @

```

...
par=...
[x,fval,exitflag] =fminsearch(@(x) fun(x,par),x0,options)
...

function f=fun(x,par)
...
f=...
end

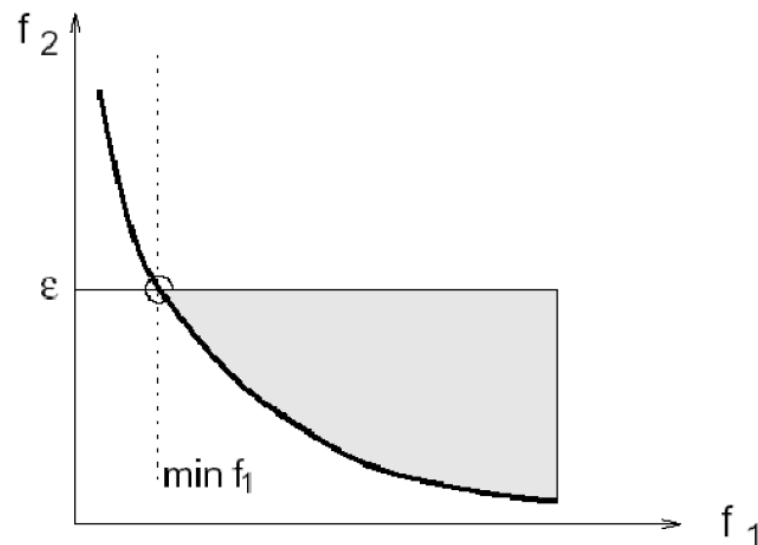
```

“fminsearch” has to be called inside a for loop where at each iteration a different value of λ_1 and λ_2 is considered.

CONSTRAINTS METHOD

$$\min_{\mathbf{x} \in \mathcal{F}} f_1(\mathbf{x})$$

$$f_2(\mathbf{x}) \leq \epsilon_2 \quad f_3(\mathbf{x}) \leq \epsilon_3 \quad \dots \quad f_{n_{of}}(\mathbf{x}) \leq \epsilon_{n_{of}}$$



One of the objective functions is minimized and the rest are treated as design constraints.

By varying ϵ different solutions on the Pareto-optimal set are computed.

In Matlab

`x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)`

Finds the minimum of a problem specified by

$$\min_x f(x) \text{ such that } \begin{cases} c(x) \leq 0 \\ ceq(x) = 0 \\ A \cdot x \leq b \\ Aeq \cdot x = beq \\ lb \leq x \leq ub, \end{cases}$$

Options = optimset('Algorithm','sqp') SQP algorithm

Options = optimset('Algorithm','interior-point') Interior-point algorithm

In our case the function call will be like the following:

```
x = fmincon(@(x) fun(x,par),x0,[],[],[],[],lb_k2,lb_r2,[ub_k2,ub_r2], @(x) constr(x,par),options)
```

with

```
function f = fun(x,par)
```

```
...
```

```
f = objective function to minimize;
```

```
end
```

```
function [c,ceq] = constr(x,par)
```

```
...
```

```
c = nonlinear constraint;
```

```
ceq = [];
```

```
end
```

“fmincon” has to be called inside a for loop where at each iteration a different value of ϵ is considered.

ANALYTICAL DERIVATION OF PARETO-OPTIMAL SET

$$\begin{aligned} \min \quad & \mathbf{F}(\mathbf{x}) = \mathbf{F}(f_1(\mathbf{x}), f_2(\mathbf{x}), \dots, f_k(\mathbf{x})) \\ \text{s.t.} \quad & \mathbf{G}(\mathbf{x}) = (g_1(\mathbf{x}), g_2(\mathbf{x}), \dots, g_w(\mathbf{x})) \leq \mathbf{0}, \quad \mathbf{x} \in \mathbb{R}^n \end{aligned}$$

A necessary condition for \mathbf{x}^* to be a Pareto-optimal solution is given by the *Fritz John necessary conditions*:

$$\begin{aligned} \sum_{i=1}^k \lambda_i \nabla f_i(\mathbf{x}^*) + \sum_{j=1}^w \mu_j \nabla g_j(\mathbf{x}^*) &= \mathbf{0} \\ \mu_j g_j(\mathbf{x}^*) &= 0 \end{aligned}$$

In our case the conditions write:

$$\begin{aligned} \sum_{i=1}^2 \lambda_i \nabla f_i &= \mathbf{0} \Rightarrow \begin{cases} \lambda_1 \frac{\partial f_1}{\partial x_1} + \lambda_2 \frac{\partial f_2}{\partial x_1} = 0 \\ \lambda_1 \frac{\partial f_1}{\partial x_2} + \lambda_2 \frac{\partial f_2}{\partial x_2} = 0 \end{cases} \Rightarrow \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \cdot \begin{Bmatrix} \lambda_1 \\ \lambda_2 \end{Bmatrix} = \begin{Bmatrix} 0 \\ 0 \end{Bmatrix} \\ \det \left(\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_2}{\partial x_1} \\ \frac{\partial f_1}{\partial x_2} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \right) &= 0 \Rightarrow \frac{\partial f_1}{\partial x_1} \cdot \frac{\partial f_2}{\partial x_2} = \frac{\partial f_1}{\partial x_2} \cdot \frac{\partial f_2}{\partial x_1} \end{aligned}$$

Matlab Symbolic Toolbox

How to define symbolic variables:

<code>syms var1 var2...</code>	Creates var1, var2, as complex symbolic variables
<code>syms var1 var2... real</code>	Creates var1, var2, as real symbolic variables
<code>syms var1 var2... positive</code>	Creates var1, var2, as positive real symbolic variables

Useful commands:

<code>diff(fun,x)</code>	Computes the derivative of the symbolic expression fun w.r.t. x
<code>simplify(fun)</code>	Simplifies the symbolic expression fun
<code>pretty(fun)</code>	Visualize the expression more clearly
<code>eval(fun)</code>	Evaluate the symbolic function numerically

<code>solve(fun,x)</code>	Solves the equation $\text{fun}=0$ with respect to x
<code>ezplot(fun,[xmin xmax])</code>	Plots the symbolic function $\text{fun}(x)$ in the specified interval
<code>subs(equation,old_var,new_var)</code>	Substitutes variable old_var with variable new_var in equation
<code>finverse(fun,var)</code>	Computes the inverse function of fun with respect to the symbolic variable var

Back to our problem

For this simple case the Fritz John conditions give the following equation:

$$\frac{\partial f_1}{\partial x_1} \cdot \frac{\partial f_2}{\partial x_2} = \frac{\partial f_1}{\partial x_2} \cdot \frac{\partial f_2}{\partial x_1}$$

The solution to this equation provides an analytical relation $x_1 = f(x_2)$ that contains the Pareto optimal set (in the design variables domain).

The obtained curve has now to be limited by the minima of the objective functions taken separately:

