Politecnico di Milano
Gaia Biasolo, Matteo Giuseppe Cigada, Federico D'Agostini

Lab 02-03: Passive suspension optimization

03/10/2025

Abstract

This report analyzes the optimization of a passive suspension system using a quarter car model to minimize discomfort and road holding. The Pareto optimal set is identified by comparing a grid sorting algorithm against weighted sum and ϵ -constraint numerical methods. Finally, an analytical solution is derived using Fritz John optimality conditions to validate the consistency of the results obtained across all approaches.

Contents

1. Introduction

The goal of this lab is to study the basic issues related to the design of a passive suspension system; this has been carried out by using the quarter car model.

1.1 Description of the problem

In the quarter car model in Figure 1, the suspension and wheel coupling is schematized by means of:

- two lumped masses ($\boldsymbol{m_1}$ representing the unsprung mass and $\boldsymbol{m_2}$ representing the sprung mass);
- an equivalent damping element $\boldsymbol{r_2}$ and an equivalent stiffness element $\boldsymbol{k_2}$ connecting the two lumped masses and representing the suspension characteristics;
- an equivalent stiffness element $\boldsymbol{k_1}$ connecting the unsprung mass to the ground and representing the tire characteristics;
- an imposed displacement $\boldsymbol{\xi_1}$ representing the road irregularity profile.

The tire has been modelled as a purely elastic component (damping is neglected), this may be a valid assumption in case of small deformation speed.

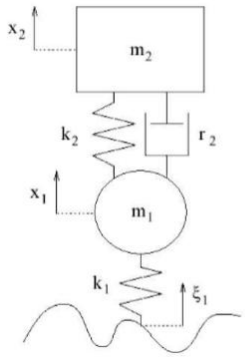

Figure 1

The quarter car model used has some limitations:

- pitch and roll motion of the car body are neglected
- suspension kinematics and related non-linearities are neglected
- suspension spring and damper and related non-linearities are neglected
- the contact patch of the tire is assumed to be a single point

Despite these limitations, some relevant observations can still be made. For instance, the suspension equivalent damping and stiffness values need to be tuned to improve vehicle comfort. The level of comfort is expressed by means of three indexes that need to be minimized.

2. Lab 02

2.1 Objective function space

The following objective functions are the ones to be minimized:

- **Discomfort**: standard deviation of the vertical acceleration

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

- **Road holding**: standard deviation of the vertical forces acting on the tyre

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

- **Working space**: standard deviation of the relative displacement of sprung and unsprung masses

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

A is a constant that depends on the road roughness and on the vehicle speed.

The objective functions space is computed considering all the possible combinations of the two variables in their range.

Discomfort

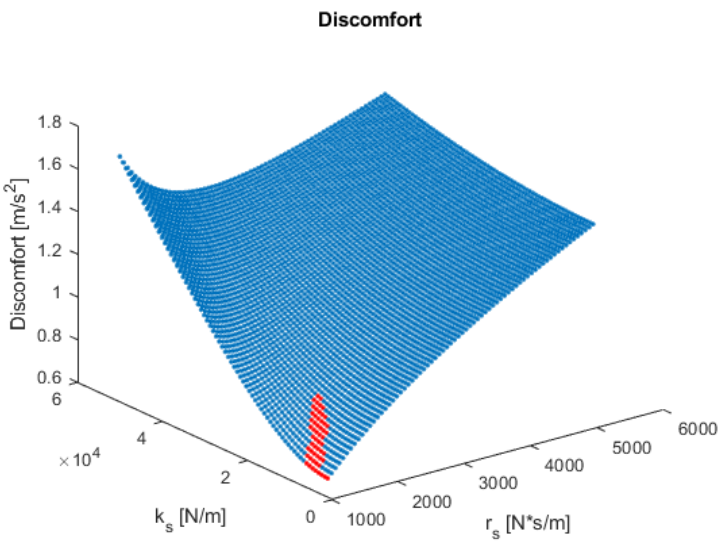

Figure 2

- Increasing suspension stiffness, the discomfort index increases.
- Increasing suspension damping, the discomfort index increases.
- The value highlighted in red are the ones of the pareto optimal set, explained in the following chapter.

Road holding

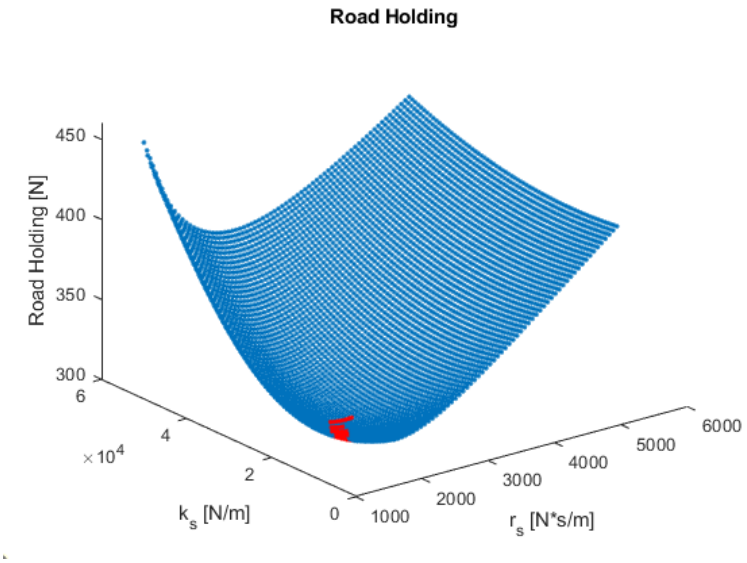

Figure 3

- Increasing suspension stiffness, the road holding index increases
- Increasing suspension damping, the road holding index increases
- The value highlighted in red are the ones of the pareto optimal set, explained in the following chapter

Working space

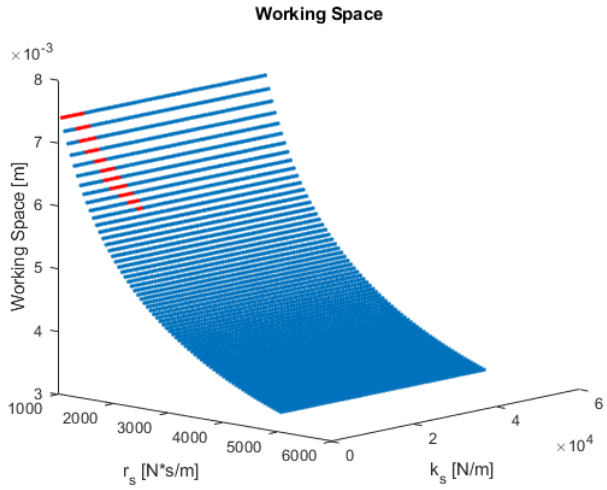

Figure 4

- The working space index is not influenced by suspension stiffness.
- Increasing the suspension damping the working space index decreases.
- The value highlighted in red are the ones of the pareto optimal set, explained in the following chapter.

2.2 Pareto optimal set for two objective functions

Definition of pareto optimal set

The pareto optimal set is a portion of the solutions obtained. A point is considered part of the pareto optimal set, if the value of one objective function can't be improved without worsening the other objective function value.

The code is implementing the condition for a point not to be Pareto optimal.

- Pareto Optimal point (Keep → Flag = 0): "To get better at X, Y must get worse"
- Dominated point (Delete → Flag = 1): "X can get better, and Y get better/stay the same"

The definition is practically applied thanks the following algorithm:

1. Creation of a matrix with all the possible combinations of the two design variables and objective functions, with an additional column of zeros (flag). Setting the whole "flag" column equal to 0 is equal to assuming that at first that every point is Pareto optimal
2. Sorting of the matrix in ascending order of the first objective function
3. Following the block diagram

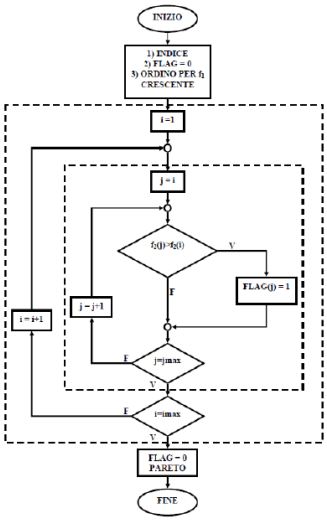

Figure 5

4. The pareto optimal are the ones with the flag = 0

Matlab implementation

```
DV_m(:,3) = discomfort;
DV_m(:,4) = roadhold;
DV_m(:,5) = workspace;
DV_m(:,6) = zeros(length(discomfort),1);

% Algorithm point 3: sorting as function of discomfort
DV_m=sortrows(DV_m,3);

for i = 1:size(DV_m,1)
    for j = i:size(DV_m,1)
        if DV_m(j,4)>DV_m(i,4)
            DV_m(j,6) = 1;
        end
    end
end

ind_v = find(DV_m(:,6)==0)
```

Note that implementing the strong pareto optimality condition (`DV_m(j,4) >= DV_m(i,4)`) or the weak pareto optimality condition (`DV_m(j,4) > DV_m(i,4)`) doesn't make any difference because for numerical reasons the objective functions values are never going to be equal up to the last digit.

Pareto optimal set for discomfort and road holding indexes

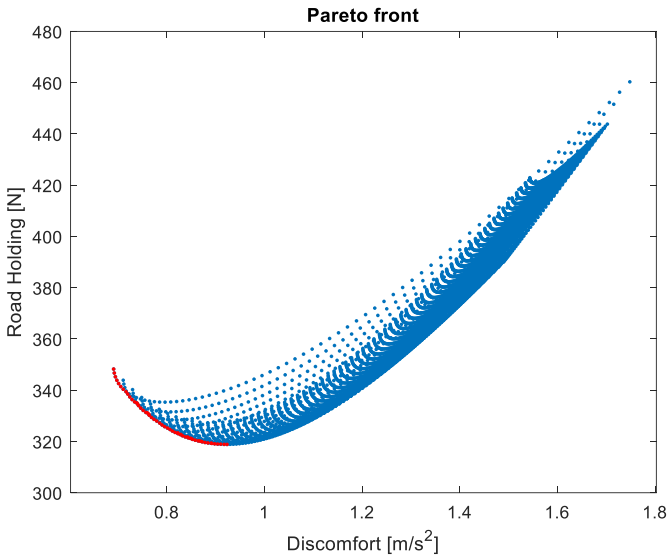

Figure 6

- The Pareto optimal set points are the points highlighted in red; from each of these points it is not possible to improve one index without worsening the other one.

2.3 Pareto optimal set for three objective functions

The concept of pareto optimal set is defined for multiple objective functions, so it is possible to extend the previous implementation to the case of 3 functions.

Matlab implementation

```
DV_m(:,3) = discomfort;
DV_m(:,4) = roadhold;
DV_m(:,5) = workspace;
DV_m(:,6) = zeros(length(discomfort),1);

DV_m=sortrows(DV_m,3);

for i = 1:size(DV_m,1)
    for j = i:size(DV_m,1)
        if DV_m(j,4)>DV_m(i,4) && DV_m(j,5)>DV_m(i,5)
            DV_m(j,6) = 1;
        end
    end
end
```

- To extend the implementation to this case an extra condition must be added to the to cycle, so the final condition applies the definition of pareto optimality to all 3 conditions. If both objective functions in position **j** are found to be better than the one in position **i**, then that solution is not optimal.

Pareto optimal set for discomfort road holding and working space indexes

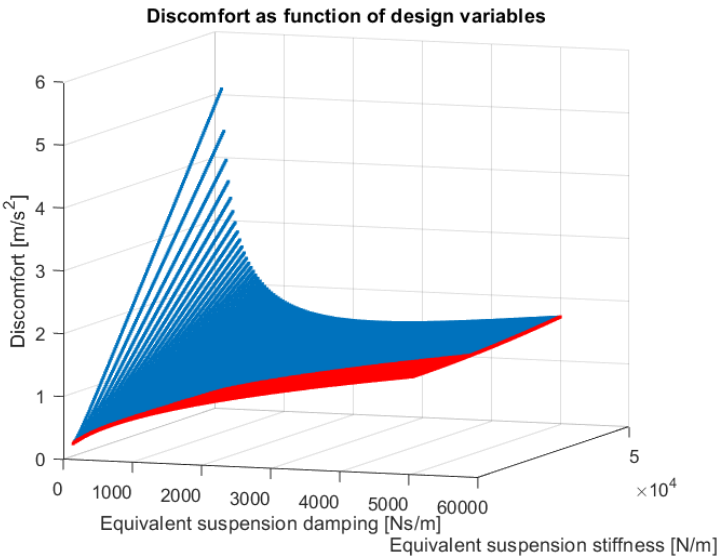

Figure 7

- The minimum of the discomfort function is in the origin

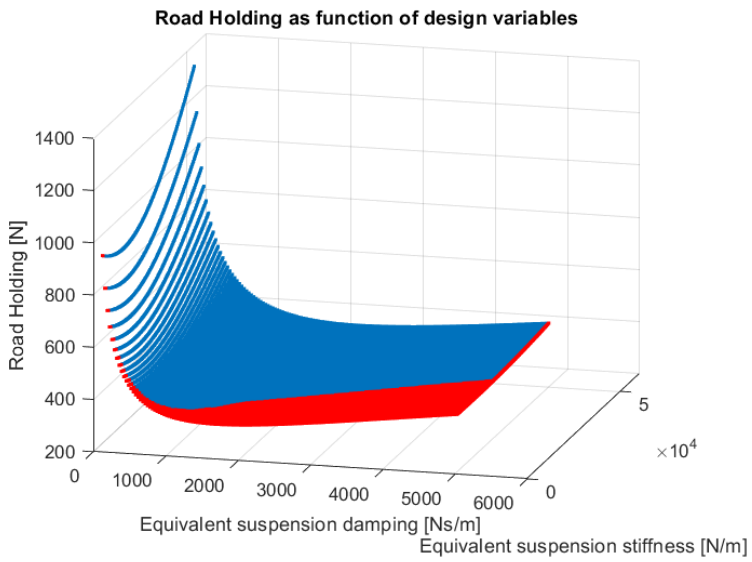

Figure 8

- The minimum of the road holding function is in a point in the middle of the domain, the same as for the case of two objective functions (see Figure 3).

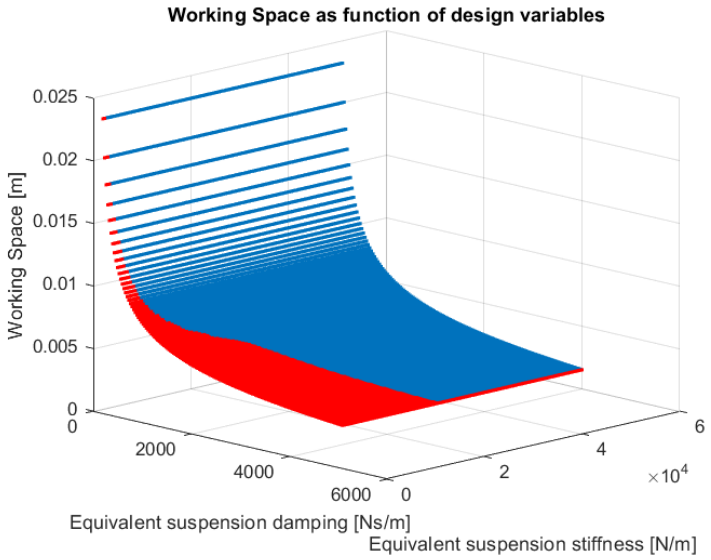

Figure 9

- The minimum of the working space function is for increasing values of the suspension damping, so all the values at the limit of the domain are chosen by the sorting algorithm.

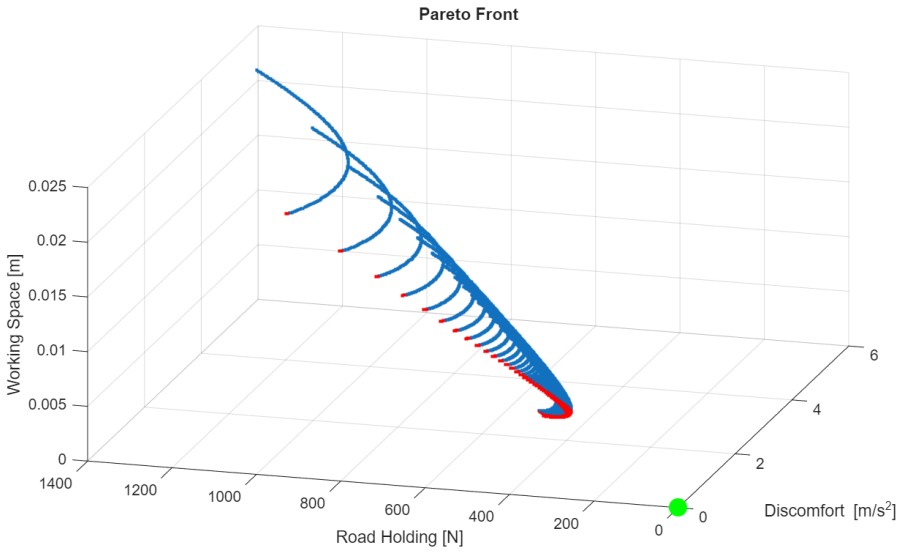

Figure 10

- The Pareto optimal set points are the points highlighted in red, from each of these points it is not possible to improve one index without worsening the other one.

3. Lab 03

3.1 Weighted sum method

In this third assignment, the problem statement was the same as the previous case. The objective functions to be minimized were the Road Holding and the Discomfort.

The main difference is how these two functions are combined. Specifically, in the weighted sum method, we need to define an aggregate objective function:

$$\phi(x) = \lambda_1 \frac{f_{ob1}}{\max(f_{ob1})} + \lambda_2 \frac{f_{ob2}}{\max(f_{ob2})}$$

Where λ_1 and λ_2 are coefficients varying between 0 and 1, with their sum equal to 1. The functions in the weighted sum need to be normalized to avoid problems with their possible different order of magnitude; this has been done considering the maximum value evaluated with the uniform grid.

The workflow of this method is the following:

- Definition of N couples of values λ_1, λ_2.
- For each couple, the function $\phi(x)$ is minimized as single objective and unconstrained problem (*fminsearch* can be used). $\overline{x}$ is computed.
- For each couple of values of λ, a corresponding point of the Pareto front is obtained, this last step is performed by evaluating the objective functions in $\overline{x}$.

This procedure has been implemented in Matlab as follows:

Matlab implementation

```
for i = 1:length(lambda1)
% weigthed sum method function definition
       phi = @(x) lambda1(i) * DS(abs(x))/max_DS + lambda2(i) * RH(abs(x))/max_RH;
       [x, fval, exitflag] = fminsearch(phi, x0);
       x_sol1_m = [x_sol1_m, x];
       x_sol1_m = abs(x_sol1_m);
End
```

Pareto optimal set for discomfort and road holding indexes

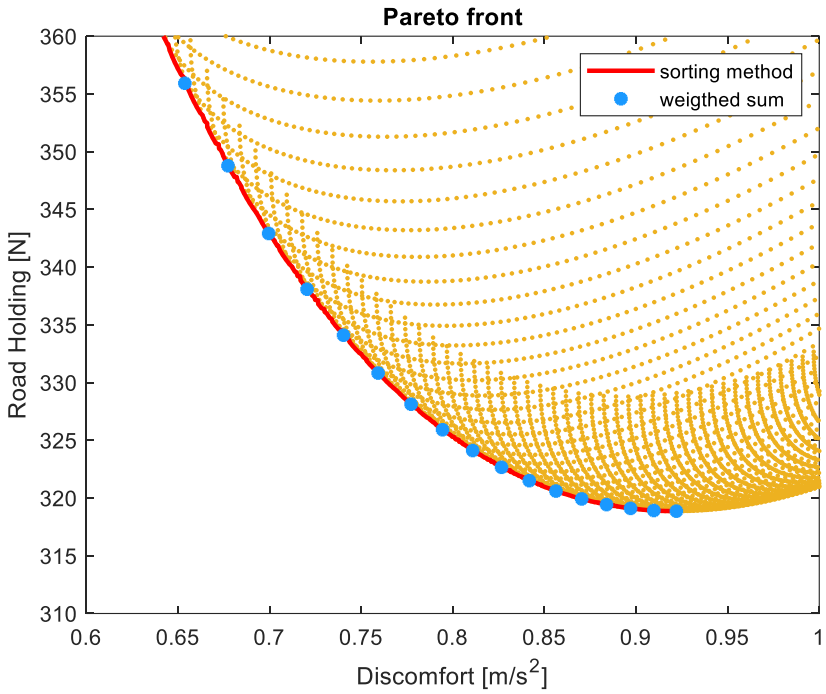

Figure 11

Plotting the solution evaluated with the uniform grid and the solution of the weighted sum and confronting them, the overlap between them is noticeable. It is important to notice that, in this case, the objective function domain is convex; for this reason, the weighted sum method can provide a useful result without missing any Pareto optimal point.

3.2 ε – Constraints method

In this method, one objective function (discomfort) is minimized, while the other one (road holding) is used as constraint; the minimization problem $\min\big(f_1(x), f_2(x)\big)$ is reformulated as:

$$\min f_1(x) \text{ subject to}$$
$$f_2(x) \leq \varepsilon_2$$

- ε_2 is the error, it assumes equally spaced values between the maximum and the minimum of f_2.

The workflow of this method is the following:

1. Definition of a vector of equally spaced values for ε_2.
2. For each value of ε_2, an inequality constraint function is written. In this case, no equality constraint is present on the objective function.

3. For each value of ε_2, the function f_1 is minimized as single objective and constrained problem. The Matlab function *fmincon* is used: it takes as input the objective function to be minimized, the lower and the upper bounds of the design variables (given by the text), the nonlinear constraint function previously defined as well as the initial guess and some options. $\overline{x}$ is computed.
4. For each value of ε_2, a corresponding point of the Pareto front is obtained, this last step is performed by evaluating the objective functions in $\overline{x}$

The Matlab function fmincon has been used to implement two different algorithms to perform the constrained minimization of f_1, that are the SQP algorithm and the interior point algorithm.

3.2.1 SQP algorithm

The base of the algorithm is the introduction of a *Lagrangian function* that allows to convert a constrained problem into an equivalent unconstrained problem. The Lagrangian function is defined as $L(x,\lambda) = f(x) - \lambda^T h(x)$, where $f(x)$ is the objective function given by the problem, λ is a vector of *Lagrangian multipliers* (treated as design variables) and $h(x)$ is a vector of constraints. $L(x,\lambda)$ is for instance minimized as unconstrained.

The main idea of the SQP algorithm is to compute at each iteration a quadratic approximation of the Lagrangian function.

The ε – Constraints method with SQP algorithm has been implemented in Matlab as follows:

Matlab Implementation

```
epsilon = linspace(max_RH, min_RH, 100);

options = optimset('Algorithm','sqp','TolX', 1e-9);

x_sol2_m = [];
for i = 1:length(epsilon)
    x = fmincon(@(x) DS(x),x0,[],[],[],[],[0, 0], [5000, 50000], @(x)
constraint(x,epsilon(i)) ,options);
    x_sol2_m = [x_sol2_m, x];
end
```

The usage of the SQP algorithm requires some tuning of the tolerances to make sure the obtained result is accurate, for this reason, the tolerance set on the difference between the result of two consecutive iterations has been lowered to 1e-9.

3.2.2 Interior-point algorithm

The ε – Constraints method with interior-point algorithm has been implemented in Matlab as follows:

Matlab implementation

```
epsilon = linspace(max_RH, min_RH, 100);

options = optimset('Algorithm','interior-point');

x_sol2_m = [];
for i = 1:length(epsilon)
    x = fmincon(@(x) DS(x),x0,[],[],[],[],[0, 0], [5000, 50000], @(x)
constraint(x,epsilon(i)) ,options);

    x_sol2_m = [x_sol2_m, x];
end
```

Pareto optimal set for discomfort and road holding indexes with alternative procedures

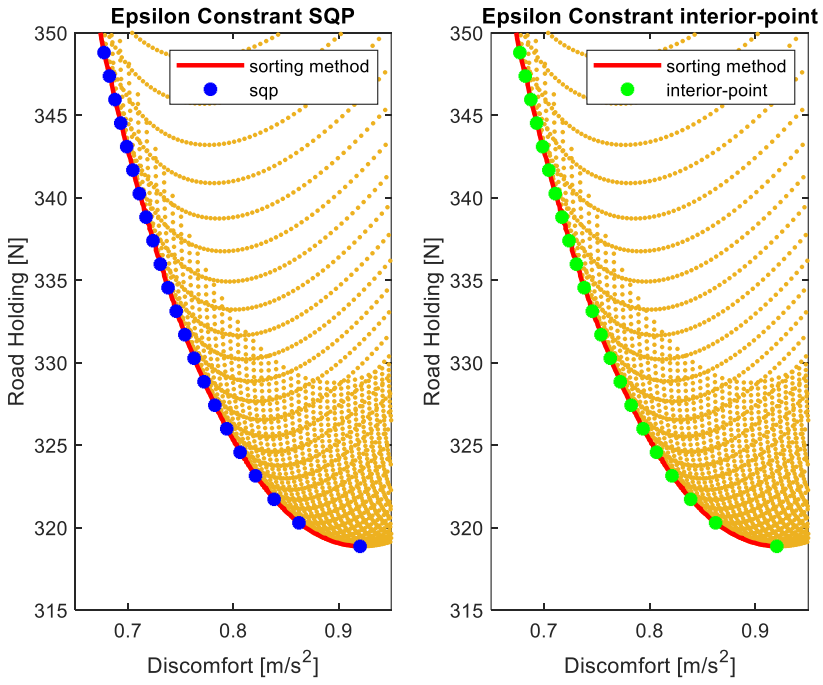

Figure 12

In Figure 12, a comparison between the obtained results is displayed; the results of the two different methods can be considered equal.

3.3 Analytical expression of the pareto optimal set

The starting point to compute the analytical expression of the pareto optimal is the Fritz John optimality condition, that represents the **necessary** condition for x^* to be Pareto optimal set.

$$\begin{cases} -\sum_{k=1}^{n_{of}} \lambda_k \nabla f_k(x^*) = \sum_{i=1}^{n_c} \eta_i \left(\nabla g_i(x^*)\right) \text{ with } \lambda_i, \eta_i \geq 0 \; and \; (\lambda_i, \eta_i) \neq (0,0) \\ g_i(x*) = 0 \; \forall i \end{cases}$$

- f_k is the k-objective function to be minimized
- g_i is the i-constraint of the problem

The analytical expression of the Pareto optimal set can be found for simple cases such as an unconstrained problem with two objective functions and two design variables. Under these assumptions the Fritz John optimality condition becomes

$$\sum_{i=1}^{2} \lambda_i \nabla f_i = 0$$

This leads to the resolution of a 2x2 system whose solution is non-trivial if the following condition is respected:

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

The convention used for this lab is the following:

- $f_1 \rightarrow$ discomfort (DS)
- $f_2 \rightarrow$ road holding (RH)
- $x_1 \rightarrow r_2$ (suspension damping)
- $x_2 \rightarrow k_2$ (suspension stiffness)

The equation written above represents an analytical relationship between the two design variables (it represents $k_2 = k_2(r_2)$). Note that the Fritz John optimality condition is only a necessary condition, therefore, the analytical expression derived includes also points that do not belong to the Pereto optimal set. The actual Pareto optimal set is limited by two points that are the minimum of discomfort and road holding (considered separately); the identification of these points is performed by imposing the optimality condition of single objective problems:

$$\begin{cases} \nabla f_1 = 0 \rightarrow P_1 = (0; 0) \\ \nabla f_2 = 0 \rightarrow P_2 = (1.5786e + 03; 1.2249e + 04) \end{cases}$$

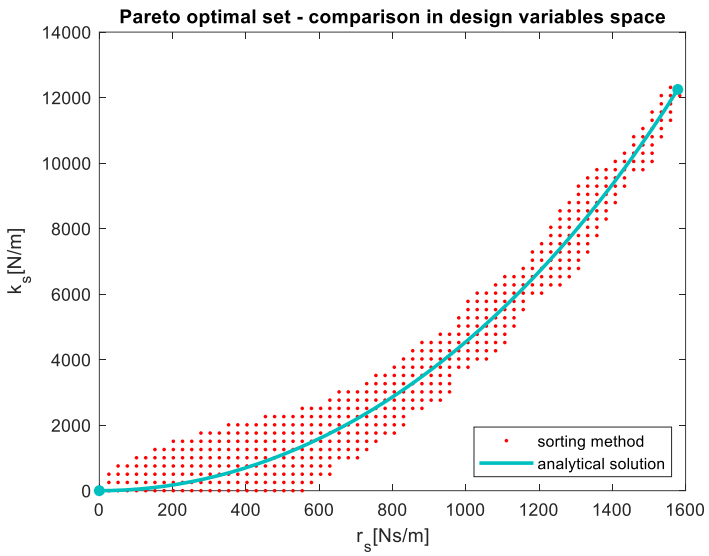

Figure 13

Figure 13 represents the design variables space. Plotting the suspension stiffness k_s against the damping coefficient r_s is essential as it represents the locus of Pareto optimal designs for the quarter-car model. The overlap between the discrete points identified by the sorting method and the continuous line derived from the analytical solution, visually confirms the consistency achieved by the different approaches used. This smooth curve defines the required non-linear relationship between the design variables that must be satisfied to achieve any optimal compromise between the objective of minimizing discomfort and maximizing road holding. This figure serves as the design map, allowing to select a specific combination of stiffness and damping that best suits the desired performance trade-off for the final passive suspension system.

To get the analytical expression of the Pareto optimal set in the objective functions space the following steps need to be implemented:

- Substitute $k_2 = k_2(r_2)$ into the objective functions → $DS(r_2)$; $RH(r_2)$ are obtained
- Compute the inverse of one of the two functions → $r_2 = r_2(DS)$ is obtained
- Substitute it into the expression of the other objective function → $RH = RH(DS)$ is obtained. **This is the analytical expression of the Pareto optimal set in the objective functions domain**.

4. Conclusion

The objective of these labs was to find the Pareto optimal set, minimizing both the discomfort and the road holding of the quarter car model, modifying the damping and the stiffness of the suspension.

The first task required the use of a uniform grid and a sorting algorithm, secondly some numerical methods were adopted (weighted sum method, constraint method) and finally the analytical derivation was carried out.

All these methods allowed us to reach the same Pareto optimal set, as it can be observed in Figure 14. The difference between them mainly lies in the computational cost required and in the tuning of the algorithms themselves. Indeed, the sorting method and the analytical solution can be adopted just for relatively simple multi-objective optimization problems: the first one would otherwise have an extremely large computational cost, while the second is limited by the need to compute partial derivatives.

The most versatile approaches were the two numerical methods. The Weighted Sum Method was effective for this specific problem only because the objective function space proved to be convex. Note that from the theoretical point of view the pareto optimal set stretches up to infinity both along the discomfort and along the road holding axis; the provided diagrams focus on the most interesting region where the best compromise between the optimization of the objective functions is obtained.

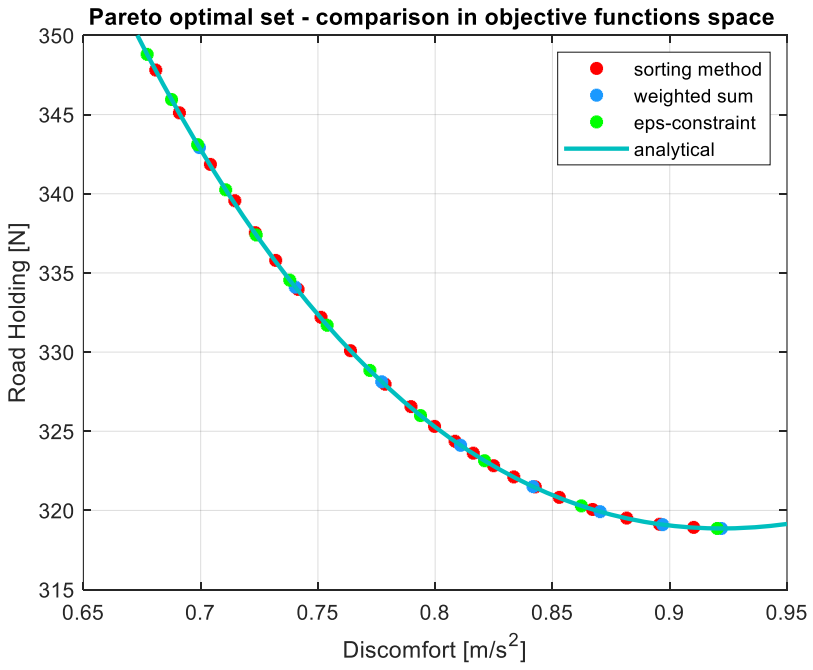

Figure 14