# Vibrational analysis of vehicle suspension

Modeling and multi-objective optimization under realistic road inputs using Mathematica and MATLAB

Jonathan Eriksson

Oct 2025

## Contents

# 1 Introduction

**Motivation and limitations**
Vibrational analysis of vehicle suspension systems is important to achieve optimal ride comfort, handling, and vehicle durability. This area of mechanical vibrations has been extensively explored and much knowledge is already available in textbooks, academic papers, and within companies. The project offers a way to showcase analytical and mathematical abilities, proficiency in Mathematica and Matlab, as well as the authors' personal interest in mechanical vibrations. Focus is placed on showcasing understanding pertaining to optimization and more extensive analysis might be avoided due to time constraints.

**Objective**
Model a quarter-car suspension system and analyze its dynamic behavior under road disturbances.

- Make a 2-DOF mechanical model of a quarter car using first principles.
- Generate a realistic road profile.
- Evaluate performance using performance metrics such as suspension deflection, body acceleration, and tire force.
- Perform multi-objective optimization of suspension parameters.
- Analyze the results using Pareto front analysis to find optimal trade-off point.

**Deliverables**
The main deliverables of the project include the following:

- A concise technical report presenting the method and the results.
- Well-structured Mathematica notebook for calculating natural frequencies.
- MATLAB functions and script for optimization.
- Optimal stiffness and damping parameters for selected road class.
- Engineering discussion and final recommendations.

## 2 System description and modeling

### Problem definition

A high-performance vehicle is being designed for rough roads at high speeds. The design team has established that the car should be optimized for ISO road class D. The car uses a passive suspension with stiffness and damping coefficients $k_b$ and $c_b$ for each wheel. Comfort, road holding, and durability are all prioritized. Find the optimal values of the suspension parameters for the best passenger comfort, road holding, and durability.

### 2 degrees-of-freedom quarter-car model

The passenger car is idealized as a 2 degrees-of-freedom (DOF) quarter-car model. The advantages of this is a simple model with both fewer equations of motion and spring/damper parameters compared to a half-car or full-car model. Disadvantages include a lower resolution in the dynamics. For instance, a half-car model could give both front and rear suspension parameters, while the quarter-car assumes these to be identical. The chosen model is therefore most appropriate for early stage design evaluation, where fast analysis and conceptual understanding are prioritized.

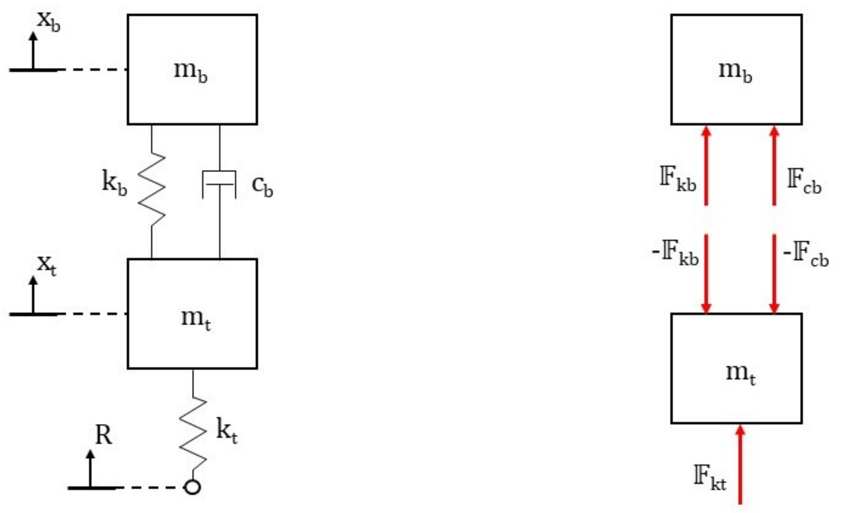

Figure 1: Idealization (left) and free body diagram (right) of the system

The idealization of Figure 1 shows the main body mass $m_b$ with position $x_b$ connected to the wheel suspension mass $m_t$ by the spring $k_b$ and the damper $c_b$. The mass $m_t$ with position $x_t$ is connected to the spring $k_t$ whose free end is subject to road displacement $R$. The free body diagram in Figure 1 does not include gravity as this force is canceled when reference points are chosen to be zero at statically displaced positions of the masses. The mass $m_b$ represents the total mass of the car minus the mass of the wheels and suspension divided by four. The mass $m_t$ is the mass of wheels and suspension divided by four. In reality, some damping is present in tire, but this is considered negligible.

Using Newton's second law

$$\sum \mathbb{F} = \frac{d}{dt}\mathbb{G}$$

on mass $m_b$ and $m_t$ respectively gives:

$$k_b x_t + c_b \frac{dx_t}{dt} - k_b x_b - c_b \frac{dx_b}{dt} = m_b \frac{d^2 x_b}{dt^2} \tag{1}$$

$$k_t R + k_b x_b + c_b \frac{dx_b}{dt} - (k_b + k_t)x_t - c_b \frac{dx_t}{dt} = m_t \frac{d^2 x_t}{dt^2} \tag{2}$$

Where $x_b, x_t$ and $R$ are functions of time.

## Model Parameters

Parameters $m_b, m_t$ and $k_t$ are given in Table 1. Note that $m_b$ is a quarter of the total mass of the vehicle body and $m_t$ is the mass of one wheel with the corresponding suspension. In a real-world setting, the spring stiffness $k_t$ could experimentally be found by measuring static displacement of the tire under different forces. In the case of a design change, the parameters of Table 1 can be changed in MATLAB and Mathematica scripts so that calculations and simulations can be re-evaluated.

| **Parameter** | **Value** | **Unit** |
|---|---|---|
| $m_b$ | 425 | kg |
| $m_t$ | 40 | kg |
| $k_t$ | 170 000 | N/m |

Table 1: Known parameters

## Road profile modeling

ISO 8608 defines road roughness and divides it into classes A-H depending on roughness, where A is a very smooth road and H is an extremely rough road. Each class corresponds to a level of power spectral density (PSD). PSD is a way of quantifying how different spatial frequencies contribute to the overall roughness of a surface and is typically measured in $m^2/cycle/m$.

There are several ways of using PSD to model road displacement mathematically. The sinusoidal approximation is a simple and straightforward method that will be used for this project.

### Power Spectral Density (PSD)

The PSD of the road profile is given by:

$$G(n) = G_0 \left(\frac{n}{n_0}\right)^{-\omega}$$

where:

- $G(n)$ is the PSD at spatial frequency $n$ [cycles/m],
- $G_0$ in $\mathrm{m}^2/\mathrm{cycle/m}$ is the reference PSD at $n_0 = 0.1\,\mathrm{cycle/m}$,
- $\omega$ is the slope parameter
- $n_0 = 0.1\,\mathrm{cycle/m}$ is the reference spatial frequency.

The spatial frequencies $n$ range from $n_0$ to $n_{\max}$ (cycle/m) and are discretized into $N$ intervals:

$$\Delta n = \frac{n_{\max} - n_0}{N}$$

$n_{\max}$ is the maximum number of cycles that will occur per unit length. A high value of $n_{\max}$ will result in smaller closely spaced bumps, while a low value will result in larger, more spread apart bumps. This value is chosen depending on how much "resolution" is needed for the simulation. It is often easier to think in terms of temporal frequency, hence, we can calculate:

$$n_{\max} = \frac{f_{max}}{v}$$

Where $f_{max}$ is the maximum temporal frequency and $v$ is the vehicle speed. $f_{max}$ is chosen high enough so that the natural frequencies of the model are included. This assures meaningful frequencies are included while inconsequential frequencies are excluded.

$N$ represents the number of sinewaves that will make up the signal. The final function will often be a better approximation with a higher value of $N$, although the computation time increases as $N$ increases.

### Frequency conversion

To convert spatial frequencies $n$ to temporal excitation frequencies $f_i$, we use:

$$f_i = v \cdot n_i$$

where $n_i$ are the discretized spatial frequencies, and $v$ is the vehicle speed.

### Amplitudes

Each harmonic component of the road profile has an amplitude $A_i$ defined as:

$$A_i = \sqrt{2\,G(n_i)\,\Delta n}$$

### Random Phase Angles

To simulate a realistic random road, each harmonic component is assigned a random phase angle $\phi_i$ in the range $[0, 2\pi]$

### Road profile as a sum of sinusoids

The vertical displacement of the road profile as a function of time can now be represented as a sum of sinusoidal components:

$$R(t) = \sum_{i=1}^{N} A_i \sin\left(2\pi f_i t + \phi_i\right)$$

### Variance check

We can now check if the variance of the generated road profile matches the expected variance for that road class. Firstly, the theoretical variance is calculated:

$$\sigma_{theory}^2 = \int_{n_0}^{n_{max}} G(n)\, dn$$

This is approximated in Matlab as shown in Appendix 1 "Road profile generation". We then calculated the variance of our generated road profile:

$$\sigma_{sim}^2 = \frac{1}{T} \int_0^T R(t)^2\, dt$$

The relative difference is calculated as:

$$\frac{|\sigma_{sim}^2 - \sigma_{theory}^2|}{\sigma_{theory}^2}$$

The relative difference can now be checked against a tolerance. If the road does not pass the variance check, a new road can be generated and $\sigma_{sim}^2$ recalculated, this can be done until the relative difference is smaller than the tolerance. This is necessary because the generated road profile is an approximation with a limited number of frequencies. The variance check ensures that the generated profile actually matches the spatial frequencies of the chosen road class.

### Parameters

The parameters in table 2 are for a class D road. The class-independent parameters $N$ and $L$ are chosen based on typical values used for accurate simulations.

| Parameter | Symbol | Value |
|---|---|---|
| Reference PSD | $G_0$ | $1024 \times 10^{-6}\,\mathrm{m}^2/\mathrm{cycle}/\mathrm{m}$ |
| Reference frequency | $n_0$ | 0.1 cycle/m |
| Maximum frequency | $f_{max}$ | 30 Hz |
| Slope of PSD | $\omega$ | 2 |
| Number of components | $N$ | 40 |
| Vehicle speed | $v$ | 22.22 m/s |
| Road segment length | $L$ | 100 m |

Table 2: Parameters used in ISO 8608-based road profile synthesis

## 3 Optimization approach

### Performance metrics

#### Body vibration acceleration

Body vibration acceleration (BVA) is the vertical acceleration of the mass $m_b$ in Figure 1. BVA is associated with ride comfort. To asses acceleration over time, a root-mean-square (RMS) value of BVA is used, with a lower value indicating better ride comfort. RMS value of BVA is defined as:

$$\mathrm{BVA} = \sqrt{\frac{1}{T}\int_0^T \left(\frac{d^2x_b}{dt^2}\right)^2 dt}$$

where $\frac{d^2x_b}{dt^2}$ is the vertical acceleration of the mass $m_b$, and $T = \frac{L}{v}$ is the total simulation time, with $L$ being the length of the road profile and $v$ the vehicle speed.

#### Suspension dynamic deflection

The suspension dynamic deflection (SDD) is the deflection $x_t - x_b$ between $m_b$ and $m_t$ in Figure 1. In other words, it is how much the suspension compresses/extends during motion. SDD is important because there are always physical limits on real suspensions. If the deflection is too large, the suspension will bottom out. Large deflections are also associated with increased mechanical stress and wear on the suspension. Furthermore, a root-mean-square (RMS) value is usually calculated for the SDD, to assess its effects over time, with a lower value indicating better durability. RMS value of the SDD is defined as:

$$SDD = \sqrt{\frac{1}{T}\int_0^T \left(x_b(t) - x_t(t)\right)^2 dt}$$

#### Tire dynamic load

Tire dynamic load (TDL) is the force between the tire and the road relative to the static load $k_t(R - x_t) - m_t g$. During optimization, we are not concerned with an absolute value and, therefore, the static part can be neglected giving $k_t(R - x_t)$. The RMS value of TDL is associated with road holding, with a lower value indicating better road holding. This is because a low RMS value of TDL means that the TDL is stable and does not fluctuate over time, and therefore the tire is subject to a more constant force from the road, leading to good traction and stability. RMS value of TDL is defined as:

$$TDL = \sqrt{\frac{1}{T}\int_0^T \left(k_t\left(R - x_t\right)\right)^2 \, dt}$$

## Optimization

### Pareto front

A Pareto front is a set of optimal solutions. A solution lies on the Pareto front if no other solution exists that improves one objective without worsening another. Finding a Pareto front restricts the number of solutions to a smaller and optimal set. This makes analysis, comparison, and trade-off decisions easier while guaranteeing that all choices are efficient.

### Normalization

Since the performance indices (BVA, SDD, and TDL) are measured in different units and have different order of magnitudes, they cannot be compared directly.

Therefore, it will become convenient to do a max-min normalization of the final Pareto points to easily compare them. This makes 0 and 1 in the range $[0, 1]$ represent the minimum and maximum values respectively:

$$\mathrm{X}_{\text{norm}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

### Genetic Algorithms

A genetic algorithm (GA) works like evolution. We have a fitness-function, in our case this is made up of BVA, SDD and TDL. An initial population is generated which consists of potential solutions to the problem. The fitness-function is then used to assess the fitness of the different solutions where only the fittest solutions are kept. Some aspects are then interchanged between these "elite" solutions (parents), this is known as "crossover". There are many methods for achieving crossover. One way is by swapping values so that the parents $(k_{b1}, c_{b1})$ and $(k_{b2}, c_{b2})$ become the children $(k_{b1}, c_{b2})$ and $(k_{b2}, c_{b1})$. After crossover, mutation is applied by making small random changes to the children. The children are then evaluated using the fitness function and the process is repeated until many generations has passed, ideally resulting in optimal solutions.

Initial population size and maximum number of generations are parameters that significantly influence the result of a GA. Larger initial population size and number of generations generally results in more optimal solutions and a better resolution of the Pareto front. Furthermore, enough generations must have passed so that the solution converges. Good population size and maximum number of generations can often be found through trial and error or experience of similar problems.

In MATLAB the function `gamultiobj` from the *Global Optimization Tool-box* is used to easily perform multi-objective optimization based on GA. Although GA:s can be used to find Pareto fronts, they are often not as effective as other methods. GA:s are typically preferred when the problem involves many variables and non-convex, discontinuous, or noisy objective function.

### Pattern search methods

Like GA, a pattern search method (PSM) starts with a random set of candidate solutions. Trial moves are then made in many directions around the candidates and evaluated using the objective function. The moves are accepted if they improve the objective function (or keep diversity on the Pareto front). Repeating this process many times usally gives a good approximation of the Pareto front.

In MATLAB the function `paretosearch` from the *Global Optimization Tool-box* is used to easily perform multi-objective optimization based on PSM. `paretosearch` will often be faster and give a smoother result compared to GA if the optimization problem has few variables and a smooth objective function.

For this project, PSM is used for the reasons stated above.

### Robustness

Dealing with random inputs can be challenging, as the behavior of the model and results might become inconsistent and hard to reproduce. One way of increasing the robustness is to give different inputs and average the objectives. Optimizing based on this average objective will give a solution that is best fit across all the inputs. Three independent road generations of the same road class are used, and an average objective is calculated for each of the three objectives.

An additional way of increasing the robustness of the results is by doing several optimization runs and taking the average optimal values of $k_b$ and $c_b$.

If some robustness is not achieved, the optimization results will not be useful as different runs will give results that are not reproducible. It is therefore imperative that robustness be evaluated in some way. In this project robustness is evaluated using the statistical measures *coefficient of variation* (CV) and *relative root-mean-square deviation* (R-RMS) of the optimized suspension parameters. These statistical measures quantify how much the optimal suspension parameters vary between runs, with low values of CV and R-RMS indicating a high robustness of the optimization.

### Constraints and selection

For an optimization algorithm to find a solution, the decision variables $k_b$ and $c_b$ must be constrained to intervals where the appropriate solutions will be. These ranges are not obvious and can usually be found through experience, calculations, engineering judgment and trial & error. Giving a smaller interval will result in less computation time. We chose the intervals based on common values for cars [1] and calculations. Appendix 2 shows that the range for $k_b$ gives natural frequencies typical for passenger vehicles. Furthermore, the intervals capture the appropriate part of the Pareto front in relation to the utopia point as shown in Figure 2 in Ch. 4 "Results".

$$10\,000 \leq k_b \leq 60\,000 \text{ [N/m]}$$

$$500 \leq c_b \leq 5\,000 \text{ [Ns/m]}$$

Normalized performance metrics BVA, SDD, and TDL for each optimal pair of $k_b$ and $c_b$ give 3D points from which a Pareto front can be plotted.

Choosing solutions is not straightforward and depends on which performance metrics are prioritized. For this problem we want a point that optimally balances the trade-offs between comfort, road holding and durability. "Distance-to-utopia" selection can therefore be used where we define an ideal point (utopia) and then choose the point on the Pareto front that is closest to the ideal point. To balance trade-offs we choose our utopia to be at $(\text{BVA}, \ \text{SDD}, \ \text{TDL}) = (0, 0, 0)$.

For more details about functions and code, see Appendix 1.

## 4 Results

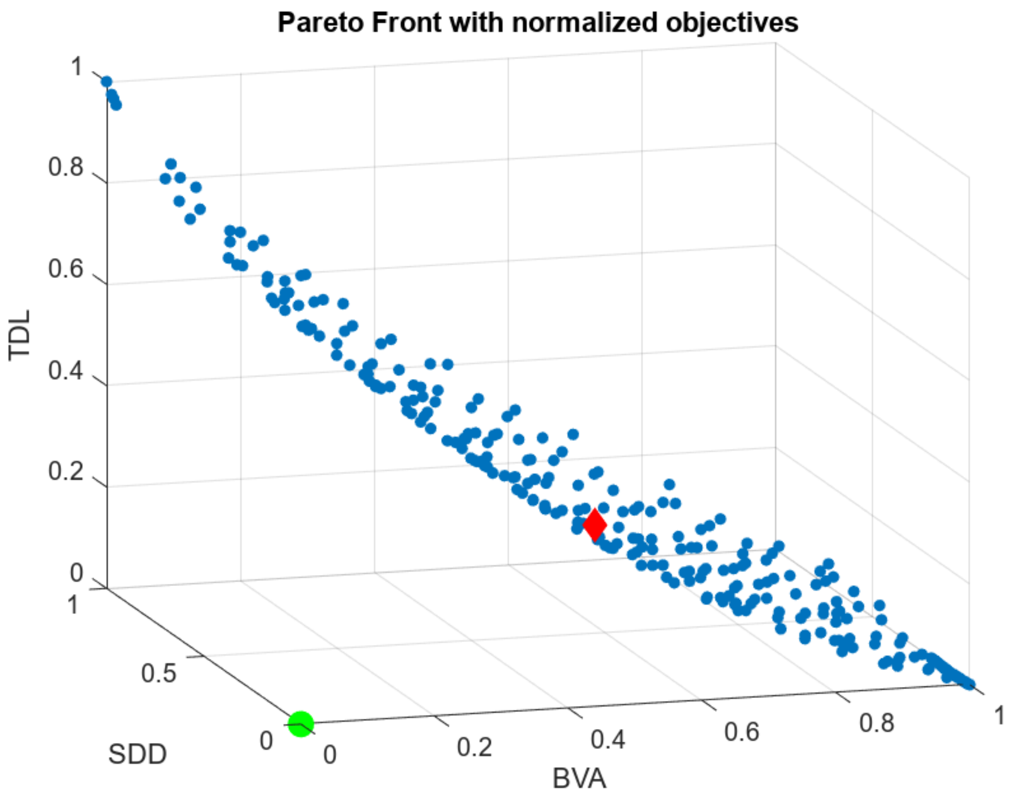

Figure 2: Pareto front from one optimization run. Pareto points (blue), distance-to-utopia point (red) and utopia point (green). BVA, SDD and TDL are min-max normalized.

### Optimal suspension parameters

A somewhat scattered Pareto front, but with a clear shape, was obtained for each run, as exemplified by Figure 2. Plotting the chosen solution and the utopia point gives visual confirmation of the solution being closest to the utopia point.

| Run | $k_b$ [N/m] | $c_b$ [Ns/m] |
|---|---|---|
| 1 | 25 600 | 1 580 |
| 2 | 27 000 | 1 540 |
| 3 | 26 600 | 1 550 |
| 4 | 25 600 | 1 580 |
| 5 | 25 600 | 1 580 |
| Avg | 26 100 | 1 570 |

Table 3: Optimal solutions for each run from distance-to-utopia selection and the average values across all runs.

The optimal points for five independent simulation runs are presented in Table 3. CV and R-RMS values are presented in Table 4.

| Parameter | Mean | Std. Dev. | CV (%) | Relative RMS |
|---|---|---|---|---|
| $k_b$ | 26 100 [N/m] | 664 | 2.25 | 0.023 |
| $c_b$ | 1 570 [Ns/m] | 18.8 | 1.20 | |

Table 4: Statistical summary of optimal suspension parameters across five optimization runs.

## 5 Discussion and conclusion

### Interpretation of optimization results

With BVA, SDD and TDL as objectives to minimize, optimal suspension parameters $k_b$ and $c_b$ were found using the pattern search method *paretosearch* in MATLAB. The optimization was performed five independent times and statistical measures showed that it consistently converged to similar values of $k_b$ and $c_b$.

The optimal values of $k_b$ and $c_b$ are in the same order of magnitude but significantly higher than typical values for a lighter passenger vehicle [1]. This seems reasonable as a stiffer suspension is generally needed when weight is increased significantly (otherwise, deformations would be so large as to cause damage to the suspension).

The two natural frequencies of the 2 DOF model are calculated in Appendix 2. Using the optimal spring stiffness value $k_b = 26\ 100$ N/m gives natural frequencies of 1.16 Hz and 11.2 Hz. These are typical for a passenger vehicle [2][3], indicating that the optimal $k_b$ value is reasonable (natural frequencies only depend on spring stiffness and not damping coefficient).

### Robustness of results

Five independent runs gave a coefficient of variation (CV) of 2.25% for $k_b$ and 1.20% for $c_b$. The two-dimensional relative RMS of the parameter set was 2.3%. This establishes that both $k_b$ and $k_b$ has low variance indicating high robustness of results. Initially, the CV and the relative root mean square were notably higher. Further investigation found that the performance metrics BVA and SDD produced non-smooth and jagged surfaces when plotted. The jaggedness caused the optimization algorithm to get stuck in local (false) minima. The main root cause was non-uniform sampling, consequently, consistent interpolation of all functions fixed this issue, see "Objective function" in Appendix 1.

### Limitations of the model

The 2-DOF model of the car does not include pitch, roll and yaw dynamics. These are typically more significant during braking and cornering.

The tires do exhibit damping in reality which is neglected as it is typically relatively low. Furthermore, all stiffness and damping parameters are not strictly linear. These do affect the performance metrics but is usually only significant during large displacement inputs or impacts.

The road profile is an approximation of a stochastic road input but checking the theoretical and actual variance assures that the sinusoidal approximation is sufficient enough in its representation of the frequencies. Furthermore, only frequencies near natural frequencies

($< 30$ Hz) are used because the effect is low for higher frequencies and would also increase the computation time.

The performance metrics BVA, SDD and TDL are RMS values and therefore might not capture transient behavior or max/min values. These are important considerations when designing a car. As an example, the maximum BVA (non-RMS) value might exceed what is appropriate. Therefore, further analysis is warranted.

The 2-DOF quarter-car model is a good first-order approximation that gives results that are suggestive but not definite. These results can be used in early prototyping and testing but are not appropriate for high-fidelity prototypes or production decisions.

## Possible extensions

A half-car model typically has 4-DOF and include angular motion (pitch), which might be useful when considering braking. Such a model could also give different parameter values for front and rear suspensions which is more realistic. A full-car model would have between 7 and 18-DOF and include angular motion in all directions (roll, pitch yaw) which can be useful when considering both cornering and breaking. A half-car model incorporates force transfer between front and rear, while a full-car model accounts for force transfer between all four wheels. This is more realistic and should give more accurate results, especially when considering breaking and cornering.

The work here can easily be extended to include other road conditions. For example, a bump or a pothole could be modeled and simulated to give insight into different realistic scenarios. Furthermore, this enables the study of the suspension under extreme inputs.

In reality, there might be some uncertainty in the known parameters. For example, the mass of the main car body $m_b$ is influenced by how much fuel is present and to what extent the car is loaded with passengers and cargo. A Monte Carlo method could be used to perform a sensitivity analysis, giving insight into how the uncertainty might affect the objectives and suspension parameters.

Including an active or semi-active suspension would mean that the parameters $k_b$ and $c_b$ are no longer fixed but are controlled by a controller such as a PID-controller. This would mean that the simulation would have to be "real-time", meaning that the simulation time steps correspond to actual time steps in the real world. These sorts of suspensions can interact dynamically, for example, softening the suspension when going over a bump and stiffening the suspension when on a smooth road. Implementing this into the current model would require the addition of a control loop that changes $k_b$ and $c_b$ according to vehicle response.

Lastly, all analytical work should at some point be accompanied by some form of experimental validation. This could be done using accelerometer data from vehicle tests.

## Conclusion

The optimization gave realistic and robust suspension parameters $k_b = 26\,100$ [N/m] and $c_b = 1\,570$ [Ns/m] for a 2-DOF quarter-car model subject to stochastic road input. The natural frequencies calculated based on $k_b$ aligns well with typical natural frequencies for passenger vehicles. Furthermore, both parameter values are similar to typical values for passenger vehicles. Statistical measures, coefficient of variation and relative root mean square, showed that the optimization converged to similar result between five independent runs despite the stochastic road input.

The Pareto front showed how there are trade-offs between comfort, durability, and road holding. Although the Pareto front was somewhat scattered in the mid-range, the shape of the front was clear and was considered sufficient enough for engineering analysis.

The quarter-car model is a good first-order approximation and provides a good basis for design. This work shows that multi-objective optimization combined with realistic road modeling can be very useful tools for suspension tuning.

## 6 References

1. Kong, Yat Sheng *et al.*, “Determining optimal suspension system parameters for spring fatigue life using design of experiment.” *Mechanics & Industry*, Vol. 20, No. 6 (2019). https://doi.org/10.1051/meca/2019062

2. Routley, Daniel, ”Spring Rates and Suspension Frequencies.” *DR Tuned Racing*. https://www.drtuned.com/... (Accessed Sept. 29, 2025).

3. Krishnamoorthy, S., Ramamoorthi, R., and Muthusamy, S., “Modal and frequency response characteristics of vehicle suspension system using full car model.” *IOP Conference Series: Materials Science and Engineering*, Vol. 810, No. 1 (2020). https://doi.org/10.1088/1757-899X/810/1/012056

# Appendix 1 - Matlab code

### Differential equation function

This function is a state-space representation of the differential equations governing the behavior of the quarter-car. This function is called inside an ODE-solver in order to solve the set of differential equations.

```matlab
function dYdt = quarterCarModel(t, Y, kb, cb, params, tRoad, R)

    Rint = interp1(tRoad, R, t, 'pchip');

    mb = params.mb;     % kg
    mt = params.mt;     % kg
    kt = params.kt;     % N/m

    xb = Y(1);
    vb = Y(2);
    xt = Y(3);
    vt = Y(4);

    dxbdt = vb;
    dxtdt = vt;

    dvbdt = (1/mb)*(kb*xt + cb*vt - kb*xb - cb*vb);
    dvtdt = (1/mt)*(kt*Rint + kb*xb + cb*vb - (kb+kt)*xt - cb*vt);

    dYdt = [dxbdt;dvbdt;dxtdt;dvtdt];
end
```

### Road profile generation

This function takes road class, vehicle speed, number of sinusoidal terms, and a tolerance as input and generates a displacement vector R and a corresponding time vector t. The tolerance is used to assure that the variance of the road profile correspond to the theoretical variance within the tolerance.

```matlab
function [t,R] = genRoadProfile(roadClass, speed, numTerms, tol)

    if nargin < 4
        tol = 0.01;
    end

%   Inputs:

%   RoadClass : 'A','B','C','D','E','F','G','H' (ISO 8608 classes)
%   speed      : vehicle speed [km/h]
%   NumTerms   : number of sinusoidal components

    % Convert speed from km/h to m/s
    v = speed / 3.6;


    % Reference parameters
    n0 = 0.1;         % reference spatial frequency [cycles/m]
    fmax = 30;        % Maximum frequency, includes both natural
       frequencies 1-2 Hz and 8-15 Hz  (exact frequencies are
       verified through calculation)
    nmax = fmax/v;         % max spatial frequency [cycles/m]
    omega = 2;        % waviness exponent (standard choice)
    L = 100;          % road length to simulate [m]

    % ISO 8608 roughness coefficients G0 at n0 (m^3/cycles)
    classTable = struct( ...
        'A', 16e-6, 'B', 64e-6, 'C', 256e-6, 'D', 1024e-6, ...
        'E', 4096e-6, 'F', 16384e-6, 'G', 65536e-6, 'H', 262144e-6);

    if isfield(classTable, roadClass)
        G0 = classTable.(roadClass);
    else
        error('Unknown road class. Use A-H.');
    end


    % Discretize spatial frequencies
    n = logspace(log10(n0), log10(nmax), numTerms);

    %n = linspace(n0, nmax, numTerms);
    %dn = n(2)-n(1);
    f = v * n;

    % PSD at each frequency
    G = G0 * (n/n0).^(-omega);

    % Amplitude from PSD
    dn = diff([n, n(end)]);   % spacing for each band
    A = sqrt(2 * G .* dn);

    n = logspace(log10(n0), log10(nmax), numTerms);
    %A = sqrt(2 * G * dn);



    % Time vector
    T = L / v;       % total duration

    fs = 8 * fmax;  % sample rate well above Nyquist rate
    dt = 1/fs;
    t = 0:dt:T;


    success = false;

    while ~success % Generates new road profile until variance check
        is OK
        % Random phases
        phi = 2*pi*rand(1, length(n));

        % Road profile = sum of sinusoids
        R = zeros(size(t));
        for i = 1:length(n)
            R = R + A(i)*sin(2*pi*f(i)*t + phi(i));
        end

        % ---------------------------
        % Variance consistency check
        % ---------------------------
        theoryVar = trapz(n, G);    % theoretical variance from PSD
        simVar = rms(R)^2;        % simulated variance from profile

        relDiff = abs(simVar - theoryVar) / theoryVar;

        if relDiff <= tol
            success = true;
            fprintf('Variance check OK: Theory RMS = %.4f m, Sim RMS
                = %.4f m (Diff = %.2f%%)\n', ...
                sqrt(theoryVar), sqrt(simVar), relDiff*100);
        end
    end

end
```

## Objective function

This function takes a $(k_b, c_b)$ pair, known parameters, time vector, and road displacement as input and outputs the three objectives BVA, SDD and TDL (RMS). The values have all been scaled so as to be in the same order of magnitude. All displacements and velocities are interpolated over a uniform time vector to ensure that the functions become smooth.

```matlab
function J = objectiveFun(a, params, tRoad, R)

    kb = a(1);
    cb = a(2);
    kt = params.kt;


    tRange = [tRoad(1) tRoad(end)];
    Y0 = [0;0;0;0];

    % quarterCarModel contains EOMs and takes kb and cb as input
    [tSol,YSol] = ode45(@(t,Y) quarterCarModel(t, Y, kb, cb, params,
        tRoad, R), tRange, Y0);

    xb = YSol(:,1);
    vb = YSol(:,2);
    xt = YSol(:,3);

    % Objectives

    BVAfactor = 3.5;  % Approx. max values from first run
    SDDfactor = 18e-3;  % Gets the objective into the same order of
       magnitude
    TDLfactor = 5500;  % This prevents numerical instability

    tUniform = linspace(tSol(1), tSol(end), 2000);
    vb_u = interp1(tSol, vb, tUniform, 'pchip');
    xb_u = interp1(tSol, xb, tUniform, 'pchip');
    xt_u = interp1(tSol, xt, tUniform, 'pchip');
    R_u  = interp1(tRoad, R, tUniform, 'pchip');

    ab = gradient(vb_u, tUniform);

    BVA = rms(ab);
    SDD = rms(xb_u - xt_u);
    TDL = rms(kt * (R_u - xt_u));

    J = [BVA/BVAfactor, SDD/SDDfactor, TDL/TDLfactor];

end
```

### Averaging function

This function takes three different road displacements as input. Objectives for each road are then calculated using *objectiveFun*. The average is then calculated for each of the three different objectives and returned by the function.

```matlab
function Javg = objectiveAvg(x, params, tAll, RAll)

    t_1 = tAll{1};
    t_2 = tAll{2};
    t_3 = tAll{3};
    R_1 = RAll{1};
    R_2 = RAll{2};
    R_3 = RAll{3};

    J_1 = objectiveFun(x, params, t_1, R_1);
    J_2 = objectiveFun(x, params, t_2, R_2);
    J_3 = objectiveFun(x, params, t_3, R_3);


    Javg = mean([J_1; J_2; J_3], 1);


end
```

### Optimization code

This is the main code used to run the optimization.

```matlab
clear
close all
params.mb = 1700/4; % Mass of quarter-car body [kg]
params.mt = 40; % Mass of one wheel suspension [kg]
params.kt = 170000; % Tire spring constant [N/m]

%lb = [10000, 500];
%ub = [60000, 5000];
lb = [12000, 1200]; % Lower boundaries ([N/m], [Ns/m])
ub = [32000, 2000]; % Upper boundaries ([N/m], [Ns/m])
NUM_TERMS = 40; % Number of sinusoidal terms in road profile

roadClasses = {'D','D','D'};
speed = 80;                          % fixed vehicle speed
NumRuns = numel(roadClasses);


% Initialize results structure
```

```
results = struct();


% Generate road profiles for all classes
tAll = cell(1,numel(roadClasses));
RAll = cell(1,numel(roadClasses));
for i = 1:numel(roadClasses)
    [tAll{i}, RAll{i}] = genRoadProfile(roadClasses{i}, speed,
       NUM_TERMS);
end

% Options for paretosearch
options = optimoptions('paretosearch', ...
    'UseParallel', true,...
    'ParetoSetSize', 500, ...
    'PlotFcn',{'psplotparetof' 'psplotparetox' 'psplotspread'});

% Running paretosearch
[xPareto,fPareto] = paretosearch(@(x) objectiveAvg(x, params, tAll,
   RAll),2,[],[],[],[],lb,ub,[],options);
```

### Calculation of robustness

This code snippet calculates the coefficient of variance and relative RMS for a set of several $(k_b, c_b)$ pairs. Optimization results for five different runs, that is, optimal points found through distance-to-utopia, are included at the beginning.

```
optParData = [2.559375000000000e+04, 1.581250000000000e+03;
    2.696093750000000e+04, 1.542187500000000e+03
    2.660937500000000e+04, 1.553125000000000e+03;
    2.559375000000000e+04, 1.581250000000000e+03;
    2.559375000000000e+04, 1.581250000000000e+03;]

X = optParData;


% --- Basic stats ---
N = size(X,1);
kb = X(:,1);
cb = X(:,2);

mean_kb = mean(kb);
mean_cb = mean(cb);
std_kb  = std(kb,0);
std_cb  = std(cb,0);
cv_kb = std_kb / mean_kb;
```

```
cv_cb = std_cb / mean_cb;
range_kb = [min(kb), max(kb)];
range_cb = [min(cb), max(cb)];
iqr_kb = iqr(kb);
iqr_cb = iqr(cb);

% --- Distances in parameter space ---
centroid = mean(X,1);                    % 1x2
dists = sqrt(sum((X - centroid).^2,2));  % Euclidean distances (Nx1)
rms_dist = sqrt(mean(dists.^2));
max_dist = max(dists);

% Normalize distance relative to centroid magnitude:
relative_rms = rms_dist / norm(centroid);

fprintf('kb: mean = %.1f, std = %.1f, CV = %.2f%%\n', ...
        mean_kb, std_kb, cv_kb*100);
fprintf('cb: mean = %.1f, std = %.1f, CV = %.2f%%\n', ...
        mean_cb, std_cb, cv_cb*100);
fprintf(' relative RMS = %.3f\n', relative_rms);
```

## Appendix 2 - Calculation of natural frequencies

## Natural frequencies of 2-DOF quarter car model

### Data

```
In[163]:=
    Remove["Global`*"]
    data = {mb → 1700/4 (*kg*),
       mt → 40(*kg*),
       kt → 170 000(*N/m*)};
    optRes = {kb → 26 100(*N/m*)};
```

### Base vectors

```
In[166]:=
    {𝕚, 𝕛, 𝕜} = IdentityMatrix[3];
```

### Dummies

```
In[167]:=
    xb = 𝓍b[t];
    xt = 𝓍t[t];
    R = ℛ[t];
```

### Positions

```
In[170]:=
    𝕣b = xb 𝕚;
    𝕣t = xt 𝕚;
```

### Velocities

```
In[172]:=
    𝕧b = ∂t 𝕣b;
    𝕧t = ∂t 𝕣t;
```

### Momentum

```
In[174]:=
    𝔾I = mb 𝕧b;
    𝔾II = mt 𝕧t;
```

### Forces

```
In[176]:=
    𝔽kb = kb (xt - xb) 𝕚;
    𝔽cb = cb (∂t xt - ∂t xb) 𝕚;
    𝔽kt = kt (R - xt) 𝕚;
```

### Sum

In[179]:=
```
ΣFI = Fkb + Fcb;
ΣFII = -Fkb - Fcb + Fkt;
```

## Newton II

### Model for finding natural frequencies

In[181]:=
```
NII = ΣFI == ∂t GI && ΣFII == ∂t GII /. {R → 0, cb → 0} // FullSimplify
```

Out[181]=

kb xb[t] + mb xb''[t] == kb xt[t] && kb xb[t] == (kb + kt) xt[t] + mt xt''[t]

## Natural frequencies

### Assuming harmonic motion

In[182]:=
```
xb[t_] = X1 Sin[ω t];
xt[t_] = X2 Sin[ω t];
```

In[184]:=
```
FullSimplify[NII, Assumptions → (Sin[ω t] > 0)]
```

Out[184]=

$kb\,X1 == kb\,X2 + mb\,X1\,\omega^2 \;\&\&\; kb\,X1 + mt\,X2\,\omega^2 == (kb + kt)\,X2$

### Natural frequency equation

In[185]:=
```
{t, ℱ} = CoefficientArrays[
    {-kb X1 + kb X2 + mb X1 ω^2 == 0, kb X1 + mt X2 ω^2 - (kb + kt) X2 == 0}, {X1, X2}] // Normal;
```

In[186]:=
```
solω´ = Solve[Det[ℱ] == 0 /. ω → √ω´, ω´];
```

### Natural frequencies

In[187]:=
```
ωn1 = √ω´ /. solω´[[1]] // FullSimplify
ωn2 = √ω´ /. solω´[[2]] // FullSimplify
```

Out[187]=

$$\frac{\sqrt{\frac{kt\,mb + kb\,(mb+mt) - \sqrt{-4\,kb\,kt\,mb\,mt + (kt\,mb + kb\,(mb+mt))^2}}{mb\,mt}}}{\sqrt{2}}$$

Out[188]=

$$\frac{\sqrt{\frac{kt\,mb + kb\,(mb+mt) + \sqrt{-4\,kb\,kt\,mb\,mt + (kt\,mb + kb\,(mb+mt))^2}}{mb\,mt}}}{\sqrt{2}}$$

### Natural frequencies [Hz] using optimal stiffness values

In[189]:=

```
Print["{fn1, fn2}=", {ωn1/(2 π), ωn2/(2 π)} /. data /. optRes // N, " [Hz]"]
```

```
{fn1, fn2}={1.16029, 11.1531} [Hz]
```

### Natural frequencies [Hz] for bounds of $k_b$

For $k_b = 10\,000$ $[N/m]$

In[190]:=

```
Print["{fn1, fn2}=", {ωn1/(2 π), ωn2/(2 π)} /. data /. kb → 10000 // N, " [Hz]"]
```

```
{fn1, fn2}={0.750154, 10.678} [Hz]
```

For $k_b = 60\,000$ $[N/m]$

In[191]:=

```
Print["{fn1, fn2}=", {ωn1/(2 π), ωn2/(2 π)} /. data /. kb → 60000 // N, " [Hz]"]
```

```
{fn1, fn2}={1.6205, 12.1078} [Hz]
```

### Conclusion

The natural frequencies for the car are 1.16 and 11.2 Hz which is typical for a passenger vehicles. Having the stiffness $k_b$ in the range $10\,000 \le k_b \le 60\,000$ gives natural frequencies in the ranges $0.75 \le f_{n1} \le 1.62$ and $10.7 \le f_{n1} \le 12.1$ [Hz]. For passenger cars the mode 1 natural frequency is typically in the range $1 \le f_{n1} \le 1.5$ [Hz] [1], which is captured by our range with some margin.

## Sources

1. S. Krishnamoorthy, R. Ramamoorthi, and S. Muthusamy, “Modal and frequency response characteristics of vehicle suspension system using full car model,” IOP Conference Series: Materials Science and Engineering, vol. 810, no. 1, p. 012056, 2020. DOI: 10.1088/1757-899X/810/1/012056.