CHALMERS

Research report 2017:01

A Computer Code for Sensitivity Analysis and Multiobjective Optimization: SAMO Tutorial

Seyed Milad Mousavi Bideleh and Viktor Berbyuk

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2017

Research report 2017:01

A Computer Code for Sensitivity Analysis and Multiobjective Optimization: SAMO Tutorial

by

Seyed Milad Mousavi Bideleh and Viktor Berbyuk

Department of Mechanics and Maritime Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden, 2017

A Computer Code for Sensitivity Analysis and Multiobjective Optimization: SAMO Tutorial

SEYED MILAD MOUSAVI BIDELEH AND VIKTOR BERBYUK

Research report 2017:01

Department of Mechanics and Maritime Sciences
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone +46 (0)31 772 1000

Table of Contents

ABSTRACT

SAMO stands for **S**ensitivity **A**nalysis and **M**ultiobjective **O**ptimization and is a computer code implemented in MATLAB to carry out a computationally efficient global sensitivity analysis and multiobjective optimization with many design applications. Current report is prepared to support SAMO users. Several case studies are considered including application of SAMO in global sensitivity analysis of bogie dynamics with respect to suspension components which in fact shows how SAMO can be used in a co-simulation environment with commercial multibody softwares like SIMPACK to solve complicated global sensitivity analysis and multiobjective optimization problems. The global sensitivity analysis works based on the multiplicative dimensional reduction method which significantly reduces the computational efforts required to evaluate sensitivity indices in comparison with to the ordinary methods. Furthermore, genetic algorithm is employed to carry out the multiobjective optimization. At the end, the theories behind global sensitivity analysis and multiobjective optimization approaches used to develop SAMO are given.

Keywords: Global sensitivity analysis, multiobjective optimization, multiplicative dimensional reduction method, genetic algorithm.

PERFACE

This work has been accomplished during October 2016 until June 2017 at the Department of Applied Mechanics, (since May 1, 2017 - Department of Mechanics and Maritime Sciences), Chalmers University of Technology, Gothenburg, Sweden. This report is for the work done on extension of SD9 project of the Chalmers railway mechanics center of excellence (CHARMEC).

The project is financially supported by the Ekman family foundation which is gratefully acknowledged.

This report covers SAMO computer code and some application examples. The computer codes for different examples can be downloaded via the following link:

https://chalmersuniversity.box.com/s/00otvu52231rxeih5dgfrdp03pv63er8

Please be aware a password is required to access the codes. Contact the authors to receive the password. Please report any bugs in the codes to the authors.

Seyed Milad Mousavi Bideleh (bideleh@gmail.com),

Viktor Berbyuk (viktor.berbyuk@chalmers.se)

June 2017, Gothenburg.

1. SAMO

SAMO is a computer code developed in MATLAB for global sensitivity analysis (GSA) and multiobjective optimization with different applications. The GSA is carried out using the multiplicative dimensional reduction method (M-DRM). The code is capable to proceed to multiobjective optimization which is done by using the genetic algorithm (GA). Based on the results of global sensitivity analysis, the user can decide the desired design parameters for multiobjective optimization. The optimization results are then presented in terms of Pareto set and Pareto front. The general overview of SAMO is shown in Fig. 1.

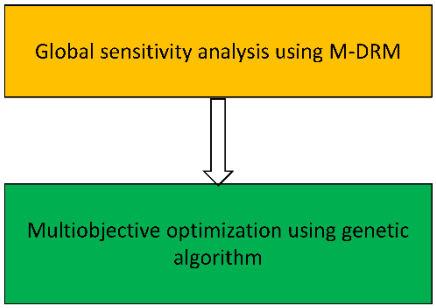


Fig. 1: Structure of SAMO.

1.1 Getting starting with SAMO

In order to start working with SAMO, three computer files are required as shown in Fig. 2. It is important to have these files in the working directory of MATLAB. It should also be noted that the names of the files should not be changed. The excel file “InputParms.xls” includes the input settings for GSA and multiobjective optimization problems. The M-file “MBSD.m” includes the multibody dynamics formulations of the system and finally the protected M-file “SAMO” is the main code that should be executed to run the global sensitivity analysis and multiobjective optimization. The structure of these files are discussed in details in the subsequent sections.

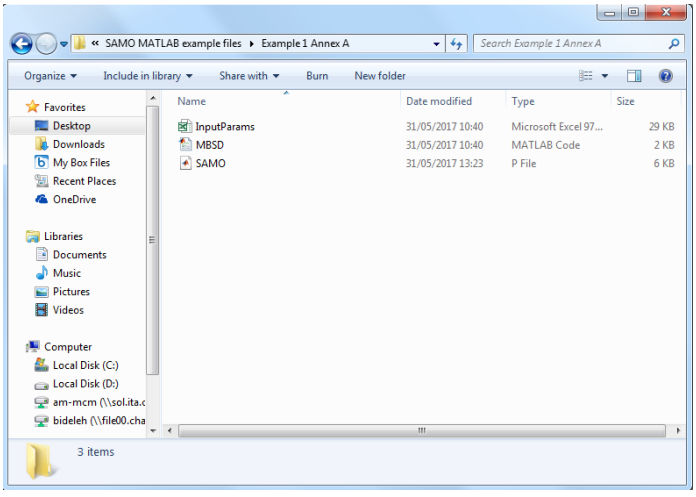
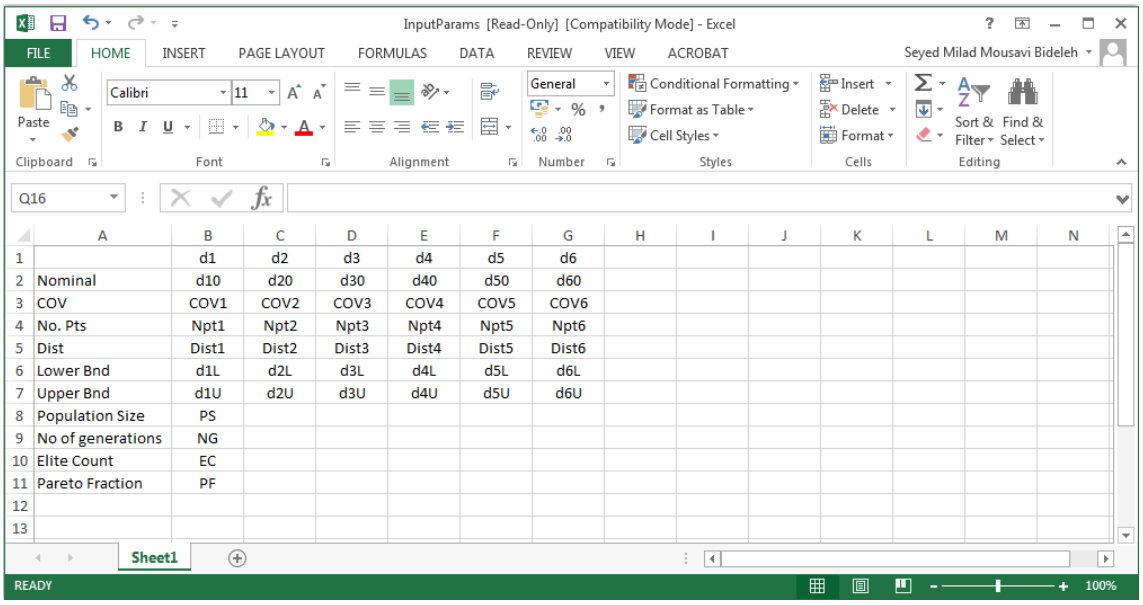


Fig. 2: Required files for running SAMO.

1.1.1 Format of the input data files

The format of the input settings for global sensitivity analysis and multiobjective optimization in "InputParams.xls" is shown in Fig. 3 for a general case. The first row denotes the design parameters index. As an example, there are six design parameters d1-d6 as shown in Fig. 3. It should be noted that users can include as many input design parameters as they want by simply adding or reducing some columns into the InputParams file. In the second row, the mean values of the design parameters (d10-d60) should be entered which is in fact the cut center **c** introduced in the annex B.



	A	B	C	D	E	F	G
1		d1	d2	d3	d4	d5	d6
2	Nominal	d10	d20	d30	d40	d50	d60
3	COV	COV1	COV2	COV3	COV4	COV5	COV6
4	No. Pts	Npt1	Npt2	Npt3	Npt4	Npt5	Npt6
5	Dist	Dist1	Dist2	Dist3	Dist4	Dist5	Dist6
6	Lower Bnd	d1L	d2L	d3L	d4L	d5L	d6L
7	Upper Bnd	d1U	d2U	d3U	d4U	d5U	d6U
8	Population Size	PS					
9	No of generations	NG					
10	Elite Count	EC					
11	Pareto Fraction	PF					
12							
13							

Fig. 3: Format of the input settings for global sensitivity analysis and multiobjective optimization.

The third row includes the coefficient of variation (COV) of each design parameter. Number of integration abscissas and distribution of the input design parameters are given in the fourth and fifth rows, respectively. It should be noted that the number of points for Gaussian quadrature integration must be an integer. Furthermore, the user can use either "n" or "N" to generate a normal distribution or enter "l" or "L" to create a lognormal distribution. If the file includes a non-integer number for number of integration points or some other letters than "n", "N", "l", or "L" for the distribution, the code gives an error to user.

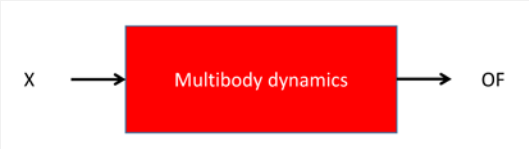
The GA settings including lower and upper bounds for variations of the design parameters, population size, number of generations, elite count, and Pareto fraction are also given in the corresponding rows of the InputParams.xls file. More details on GA settings is found in the MATLAB documentation for `gamultiobj` function [1].

As aforementioned, the number of columns could be variable and it does not affect the generality of the code, but it is important to fill out the respective data in rows 2-7 in the input data file. Moreover, the order and format of the file should not be changed as well. For example, it is not allowed to enter the COV in the second row or mean values in the third row.

1.1.2 Format of the multibody dynamics code

The objective functions should be calculated in the “MBSD.m” file. The general format is shown in Fig. 4. In fact, the user should implement his own multibody dynamics code in a way that the code reads vector of input design parameters **X** and delivers the vector of respective objective functions **OF** as shown by the block diagram in Fig. 4. It should be noted that the vector **OF** which includes the values of objective functions for each set of the design parameters **X**, should be saved in a column wise manner such that each column indicates respective values of each of the objective functions. It is also recommended not to make any change in the rest of the code shown in Fig. 4. Some examples are shown in the annex A to clarify this part more.

```
function OF = MBSD(X)
global m indx indxOF d0 iter optresult
iter

if length(X) == m
%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for GSA
%%%%%%%%%%%%%%%%%%%%%%%%
    d = X;
    iter = iter+1;
else
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    d = d0;
    for i = 1:length(X)
        d(indx(i)) = X(i);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% The MBSD code should be written here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```




```
if length(X) ~= m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
OFOpt = zeros(1,length(indxOF));

   for i = 1:length(indxOF)
        OFOpt(i) = OF((indxOF(i)));
   end

optresult(iter,:) = [X OF];

OF = OFOpt;
iter = iter+1;
xlswrite('optresults.xls',optresult);
end
```

Fig. 4: Format of the “MBSD.m” code.

1.2. Example: Wheel torque/travel time optimization of a vehicle

The aim of this example is to study the sensitivity of wheel torque and travel time with respect to different design parameters. A bi-objective optimization problem is also solved to yield the tradeoff solutions of the wheel torque/travel time optimization problem.

The free body diagram of the vehicle is shown in Fig. 5.

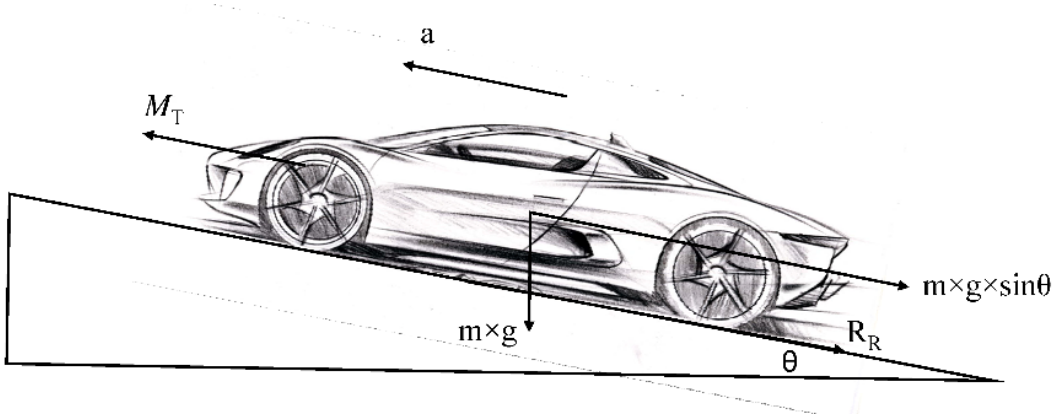


Fig. 5: Free body diagram of a vehicle on a ramp.

The motor traction force can be approximated as:

$$M_T = R_R + G_R + F_A \,. \tag{1}$$

Here, R_R is the rolling resistance given by

$$R_R = m \times g \times C_r \,. \tag{2}$$

The gravitational resistance force G_R is

$$G_R = m \times g \times \sin\theta \,. \tag{3}$$

Acceleration force (F_A) is expressed as

$$F_A = m \times a_{max} \,. \tag{4}$$

The time required to achieve the maximum speed is

$$t_a = \frac{V_{max}}{a_{max}}, \tag{5}$$

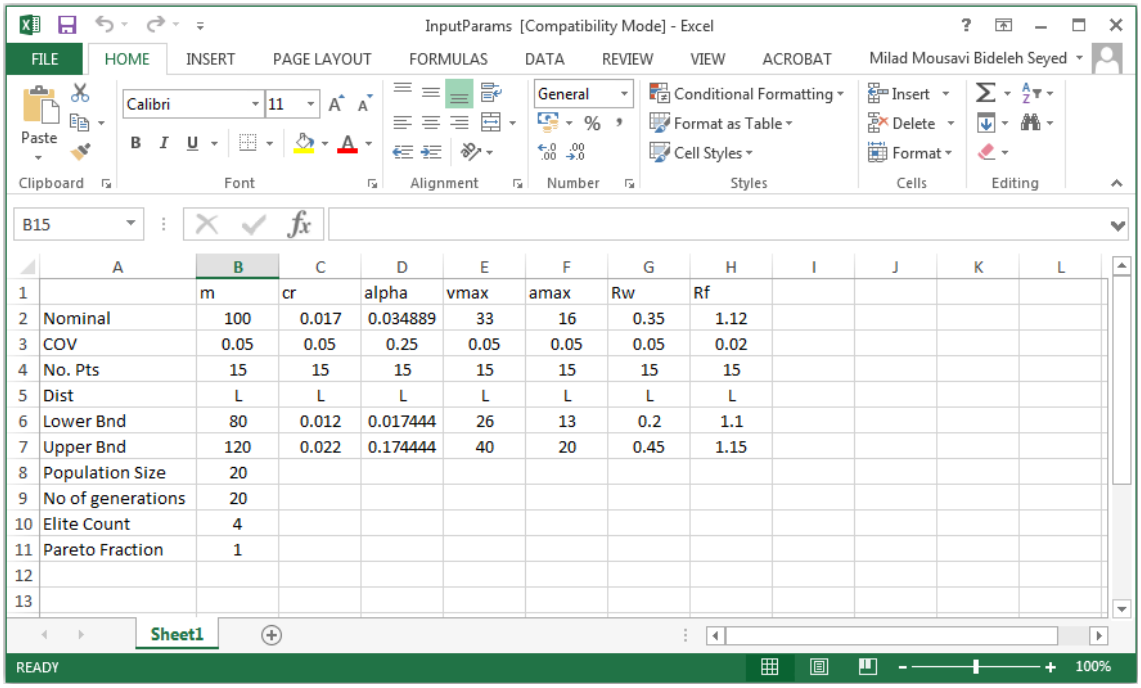
where, V_{max} is the maximum speed and a_{max} is the maximum acceleration.

The wheel torque (T_W) is approximated as

$$T_W = M_T \times R_W \times R_f \,, \tag{6}$$

where, R_W is the wheel radius and R_f is the resistance factor.

Task: It is desired to shorten the journey time and reduce fuel consumption. This can be done by minimizing the wheel torque (T_W) and the time required to achieve maximum speed (t_a). Therefore, T_W and t_a are the two objective functions. A GSA is carried out first to study the effects of different design parameters on these objective functions. The input data file is shown in Fig. 6.



	A	B	C	D	E	F	G	H
1		m	cr	alpha	vmax	amax	Rw	Rf
2	Nominal	100	0.017	0.034889	33	16	0.35	1.12
3	COV	0.05	0.05	0.25	0.05	0.05	0.05	0.02
4	No. Pts	15	15	15	15	15	15	15
5	Dist	L	L	L	L	L	L	L
6	Lower Bnd	80	0.012	0.017444	26	13	0.2	1.1
7	Upper Bnd	120	0.022	0.174444	40	20	0.45	1.15
8	Population Size	20						
9	No of generations	20						
10	Elite Count	4						
11	Pareto Fraction	1						
12								
13								

Fig. 6: Input data file.

It can be seen that there are 7 design parameters. The mean values and variation bounds are chosen from [2]. The MBSD.mat code is shown in Fig. 7. The design parameters are updated using the following lines in the MBSD code:

```
mv = d(1);
cr = d(2);
alpha = d(3);
vm = d(4);
am = d(5);
Rw = d(6);
Rf = d(7);
```

Based on the updated design parameters the objective functions are calculated using Eqs. (5, 6). See, Fig. 7.

```
Tw = mv*9.81*(cr+sin(alpha)+am/9.81)*Rw*Rf;
tm = vm/am;
OF = [tm;Tw]';
```

```
function OF = MBSD(X)
global m indx indxOF d0 iter optresult
iter

if length(X) == m
%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for GSA
%%%%%%%%%%%%%%%%%%%%%%%%%
    d = X;
    iter = iter+1;
else
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    d = d0;
    for i = 1:length(X)
        d(indx(i)) = X(i);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% The MBSD code should be written here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

mv = d(1);
cr = d(2);
alpha = d(3);
vm = d(4);
am = d(5);
Rw = d(6);
Rf = d(7);

Tw = mv*9.81*(cr+sin(alpha)+am/9.81)*Rw*Rf;
tm = vm/am;

OF = [tm;Tw]';

if length(X) ~= m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
OFOpt = zeros(1,length(indxOF));

   for i = 1:length(indxOF)
        OFOpt(i) = OF((indxOF(i)));
   end

optresult(iter,:) = [X OF];

OF = OFOpt;
iter = iter+1;

xlswrite('optresults.xls',optresult);

end
```

Fig. 7: MBSD code for wheel torque/travel time optimization example

To run the GSA, one should type SAMO in the MATLAB workspace. The SAMO function reads the InputData file and performs the GSA based on the multibody dynamics of the system. The GSA results in terms of the total sensitivity indices of different objective functions with respect to design parameters are shown in Fig. 8. Each color bar indicates a number which shows the sensitivity. Higher number in

the color bar reflect more sensitivity. SAMO also automatically plots the GSA results of each objective function separately which are not shown here.

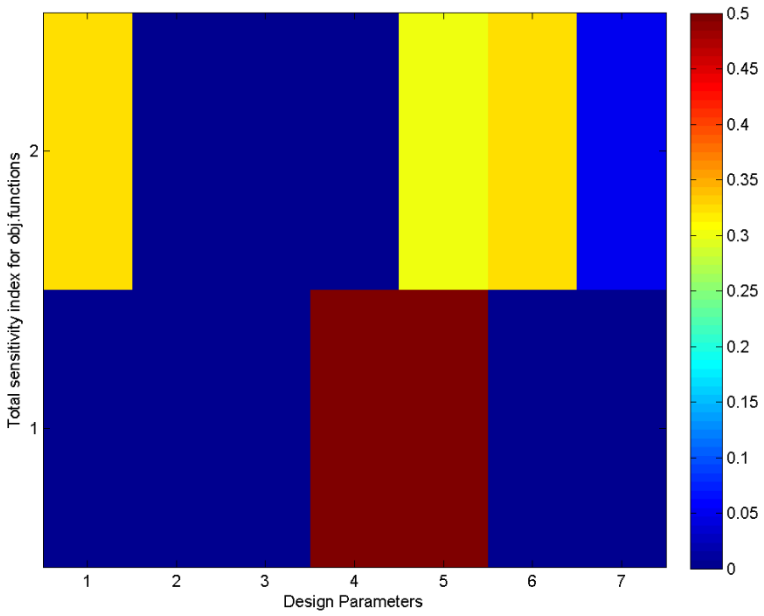


Fig. 8: Sensitivity indices.

After the GSA, SAMO asks user whether or not to proceed with optimization:

```
To proceed to optimization press Y, to return press N?
```

The code ends by pressing “n” or “N”. But if the user press “y” or “Y”, the code asks for the design parameter indices for optimization:

```
Please enter the design parameters index number for optimization in
an ascending order and vector form.

E.g. [1 3 4 7]
```

Assume that mass (m), maximum speed (V_{max}), maximum acceleration (a_{max}), and the wheel radius (R_W) are chosen as the design parameters to minimize the wheel torque (T_W) and the time required to achieve maximum speed (t_a). Therefore, the user should enter [1 4 5 6] in the MATLAB command line.

```
Please enter the objective functions index number for optimization in
an ascending order and vector form.

E.g. [1 3 4 7]
```

Since both objective functions T_w and t_a are desired to be optimized the user should enter [1 2] in the MATLAB command line. The multiobjective optimization is then carried out using GA and the prescribed settings in the InputData file. The Pareto optimized results are shown in terms of Pareto sets

and Pareto fronts. SAMO automatically provides 2D plots for all the Pareto sets and Pareto fronts in a two by two basis.

The Pareto set and Pareto front are also saved on “Pareto Set.xls” and “Pareto Front.xls” files, respectively. These excel files can be used to plot the normalized and absolute Pareto fronts that are shown in Fig. 9. The value associated with the initial guess is shown by the red cross (×).

The optimized values of the design parameters are shown in Fig. 10.

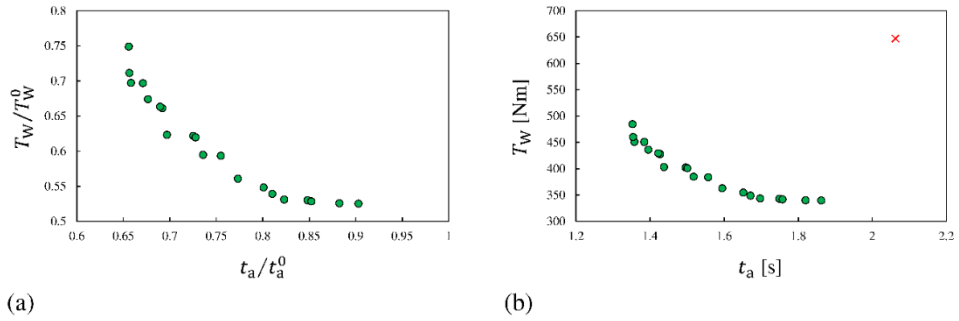


Fig. 9: Pareto front: a) Normalized objective functions; b) Absolute values.

It can be seen that increasing the wheel radius and reducing maximum speed minimizes the accelerating time. In contradict, reducing the wheel radius and increasing maximum speed minimizes the wheel torque.

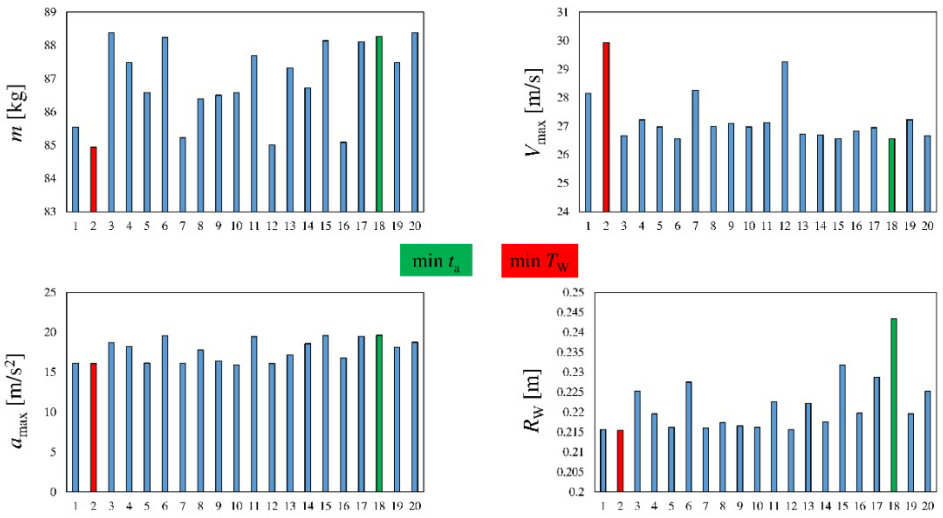


Fig. 10: Pareto set.

Annex A

Examples

Example 1: *Eigen values of a spring-mass system*

This example is taken from section 4.7 of [3]. Consider the 3 degree of freedom (DOF) mass-spring system shown in Fig. A1.

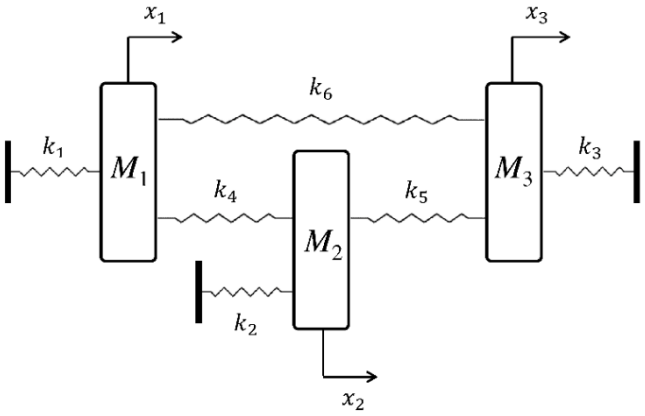


Fig. A1: Linear 3 DOF mass-spring system.

The equations of motion are given by Eqs. (A1):

$$\begin{aligned} &M_1\ddot{x}_1 + \left(k_1 + k_4 + k_6\right)x_1 - k_4 x_2 - k_6 x_3 = 0 \\ &M_2\ddot{x}_2 - k_4 x_1 + \left(k_2 + k_4 + k_5\right)x_2 - k_5 x_3 = 0 \\ &M_3\ddot{x}_3 - k_6 x_1 - k_5 x_2 + \left(k_3 + k_5 + k_6\right)x_3 = 0 \end{aligned}, \tag{A1}$$

or in matrix form:

$$M\ddot{\mathbf{x}} + K\mathbf{x} = 0\,. \tag{A2}$$

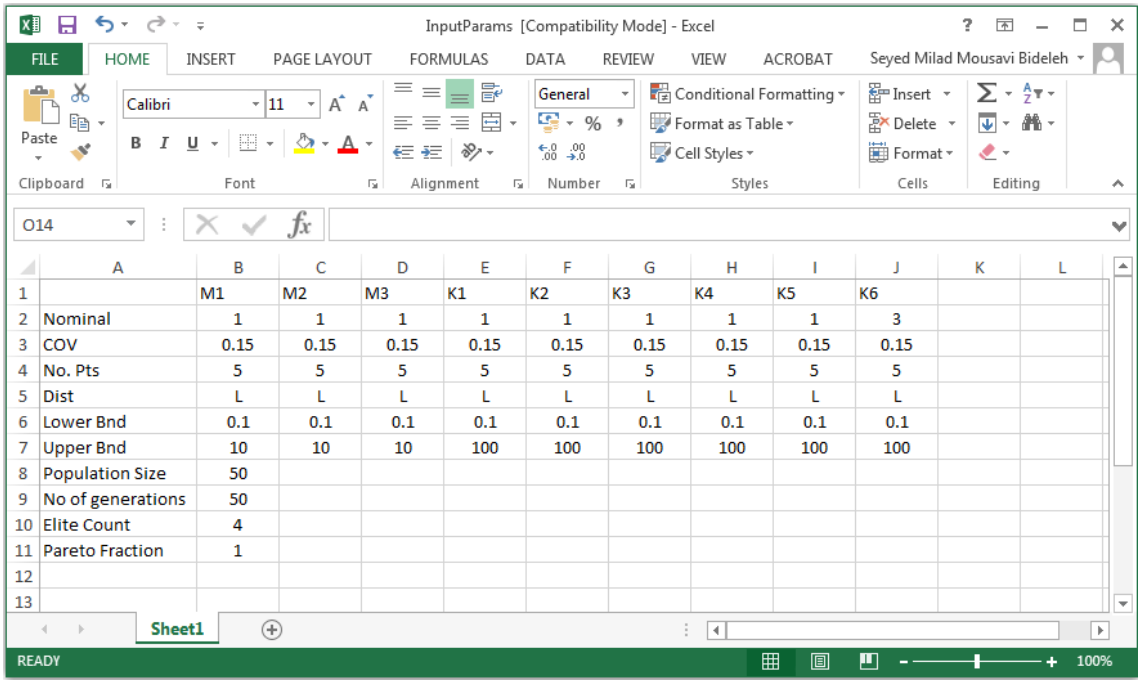
Here, mass (M) and stiffness (K) matrices are defined by Eqs. (A3, A4), respectively:

$$M = \begin{bmatrix} M_1 & 0 & 0 \\ 0 & M_2 & 0 \\ 0 & 0 & M_3 \end{bmatrix}, \tag{A3}$$

$$K = \begin{bmatrix} k_1 + k_4 + k_6 & -k_4 & -k_6 \\ -k_4 & k_2 + k_4 + k_5 & -k_5 \\ -k_6 & -k_5 & k_3 + k_5 + k_6 \end{bmatrix}. \tag{A4}$$

The target is to study the GSA of the system's eigen values with respect to the design parameters.

The input data file ("InputParams.xls") for GSA is shown in Fig. A2. There are 9 design parameters (3 masses and 6 stiffnesses) with mean, COV, and number of integration abscissas given in [3]. A lognormal distribution of the input data is considered.

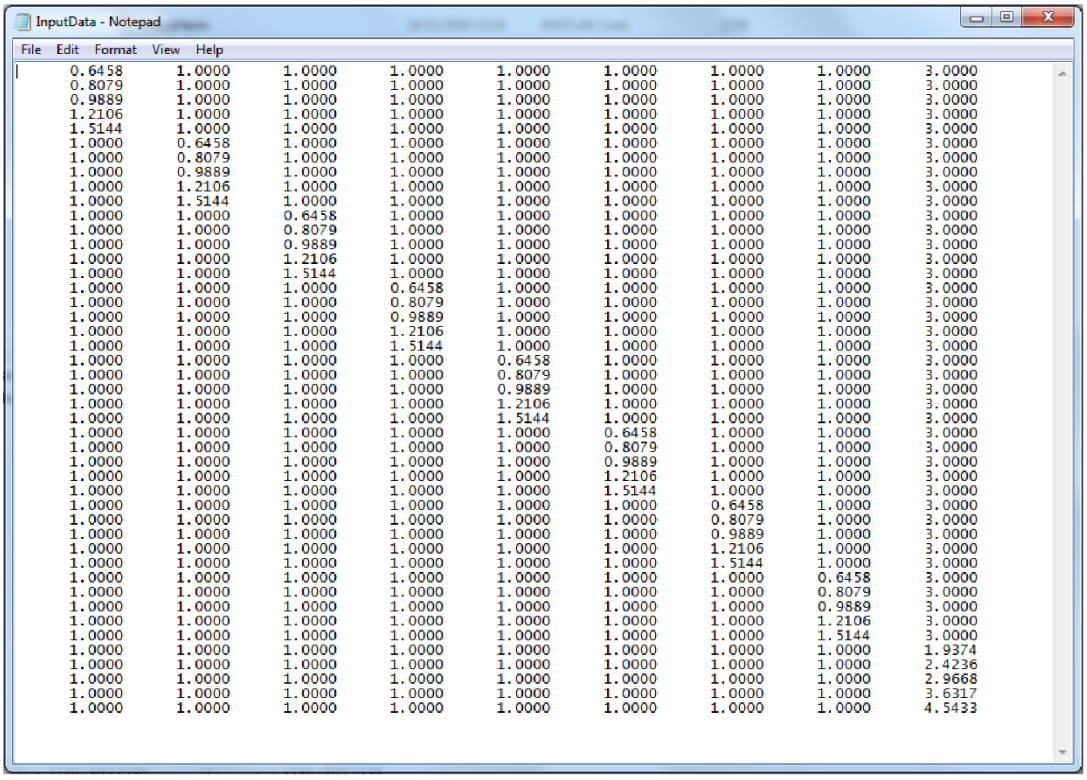


InputParams [Compatibility Mode] - Excel

	A	B	C	D	E	F	G	H	I	J	K	L
1		M1	M2	M3	K1	K2	K3	K4	K5	K6		
2	Nominal	1	1	1	1	1	1	1	1	3		
3	COV	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15	0.15		
4	No. Pts	5	5	5	5	5	5	5	5	5		
5	Dist	L	L	L	L	L	L	L	L	L		
6	Lower Bnd	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1		
7	Upper Bnd	10	10	10	100	100	100	100	100	100		
8	Population Size	50										
9	No of generations	50										
10	Elite Count	4										
11	Pareto Fraction	1										
12												
13												

Fig. A2: Input data file for GSA.

SAMO reads the input data file and based on the provided setting creates the simulation matrix which is saved as a text file (“InputData.txt”) shown in Fig. A3.



InputData - Notepad

File Edit Format View Help

0.6458	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
0.8079	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
0.9889	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.2106	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.5144	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	0.6458	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	0.8079	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	0.9889	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.2106	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.5144	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	0.6458	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	0.8079	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	0.9889	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.2106	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.5144	1.0000	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	0.6458	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	0.8079	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	0.9889	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.2106	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.5144	1.0000	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	0.6458	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	0.8079	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	0.9889	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.2106	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.5144	1.0000	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	0.6458	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	0.8079	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	0.9889	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.2106	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.5144	1.0000	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.6458	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.8079	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9889	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.2106	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.5144	1.0000	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.6458	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.8079	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.9889	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.2106	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.5144	3.0000
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.9374
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	2.4236
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	2.9668
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	3.6317
1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	4.5433

Fig. A3: Simulation matrix for GSA.

Each column in the simulation matrix represents variations of a particular design parameter based on the order given in Fig. A2. The system response and objective functions (eigen values in this example) should be evaluated for design inputs given by each row of the simulation matrix. Therefore, the number of function evaluations is equal to the number of rows of the simulation matrix, i.e. $n \times N$ (9 design parameters × 5 integration abscissas = number of simulations 45).

To calculate the objective functions (3 eigen values), it is necessary to modify the multibody system dynamics (MBSD) code in MATLAB. As explained before, the MBSD code reads the simulation matrix and for each particular row of this matrix updates the design parameters and calculates the objective functions, see Fig. A4. It should be noted that the objective functions should be stored column-wised. As an example, the first, second, and third eigen values of the system are calculated and saved in a column-wised manner as shown in Fig. A5. This file is then stored and will be used later on by the main code to calculate the total GSA indices.

The sensitivity of the first, second, and third eigen values with respect to different design parameters are shown in Figs. A6 (a-c). The mapping between the design parameters and the total global sensitivity indices are also summarized in Fig. A7 in which assigns a color to each sensitivity index. The higher value on the color bar represents higher sensitivity.

The results are in excellent agreement with those reported in [3].

After the GSA, the code asks user whether or not continue with multiobjective optimization:

```
To proceed to optimization press Y, to return press N?
```

The code ends by pressing “n” or “N”. But if the user enter “y” or “Y”, the code asks for the design parameter indices for optimization:

```
Please enter the design parameters index number for optimization in an ascending order and vector form.

E.g. [1 3 4 7]
```

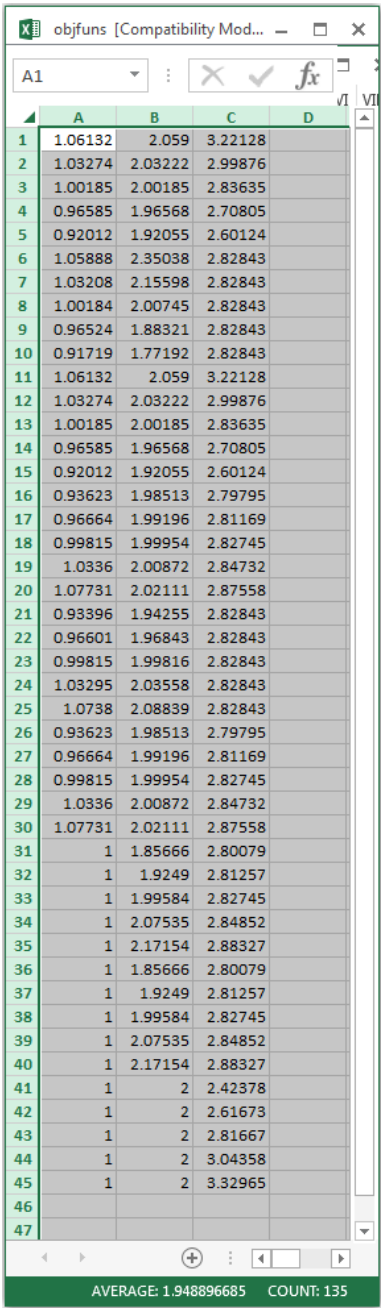
Assume that M_1, M_2, M_3, K_4, and K_6 are chosen for optimization. So, the input vector is [1 2 3 7 9]

```
Please enter the objective functions index number for optimization in an ascending order and vector form.

E.g. [1 3 4 7]
```

All three objective functions (first, second, and third eigen values) can be assumed as the objective functions to be used in multiobjective optimization. Therefore, the user can enter [1 2 3] to take into account all three objective functions.

```
function OF = MBSD(X)
global m indx indxOF d0 iter optresult
iter

if length(X) == m
%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for GSA
%%%%%%%%%%%%%%%%%%%%%%%%%
    d = X;
    iter = iter+1;
else
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    d = d0;
    for i = 1:length(X)
        d(indx(i)) = X(i);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%% The MBSD code should be written here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        M = diag([d(1),d(2),d(3)]);
        K = [d(4)+d(7)+d(9) -d(7) -d(9);-d(7)
d(5)+d(7)+d(8) -d(8);-d(9) -d(8) d(6)+d(8)+d(9)];

        E = eig(K,M);
        e1 = sqrt(E(1));
        e2 = sqrt(E(2));
        e3 = sqrt(E(3));

OF = [e1;e2;e3]';

if length(X) ~= m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
OFOpt = zeros(1,length(indxOF));

   for i = 1:length(indxOF)
        OFOpt(i) = OF((indxOF(i)));
   end

optresult(iter,:) = [X OF];

OF = OFOpt;
iter = iter+1;

xlswrite('optresults.xls',optresult);

end
```

Fig. A4: Function MBSD.mat to calculate the objective functions.



	A	B	C	D
1	1.06132	2.059	3.22128	
2	1.03274	2.03222	2.99876	
3	1.00185	2.00185	2.83635	
4	0.96585	1.96568	2.70805	
5	0.92012	1.92055	2.60124	
6	1.05888	2.35038	2.82843	
7	1.03208	2.15598	2.82843	
8	1.00184	2.00745	2.82843	
9	0.96524	1.88321	2.82843	
10	0.91719	1.77192	2.82843	
11	1.06132	2.059	3.22128	
12	1.03274	2.03222	2.99876	
13	1.00185	2.00185	2.83635	
14	0.96585	1.96568	2.70805	
15	0.92012	1.92055	2.60124	
16	0.93623	1.98513	2.79795	
17	0.96664	1.99196	2.81169	
18	0.99815	1.99954	2.82745	
19	1.0336	2.00872	2.84732	
20	1.07731	2.02111	2.87558	
21	0.93396	1.94255	2.82843	
22	0.96601	1.96843	2.82843	
23	0.99815	1.99816	2.82843	
24	1.03295	2.03558	2.82843	
25	1.0738	2.08839	2.82843	
26	0.93623	1.98513	2.79795	
27	0.96664	1.99196	2.81169	
28	0.99815	1.99954	2.82745	
29	1.0336	2.00872	2.84732	
30	1.07731	2.02111	2.87558	
31	1	1.85666	2.80079	
32	1	1.9249	2.81257	
33	1	1.99584	2.82745	
34	1	2.07535	2.84852	
35	1	2.17154	2.88327	
36	1	1.85666	2.80079	
37	1	1.9249	2.81257	
38	1	1.99584	2.82745	
39	1	2.07535	2.84852	
40	1	2.17154	2.88327	
41	1	2	2.42378	
42	1	2	2.61673	
43	1	2	2.81667	
44	1	2	3.04358	
45	1	2	3.32965	
46				
47				

Fig. A5: First, second, and third eigen values corresponding to each row of the simulation matrix.

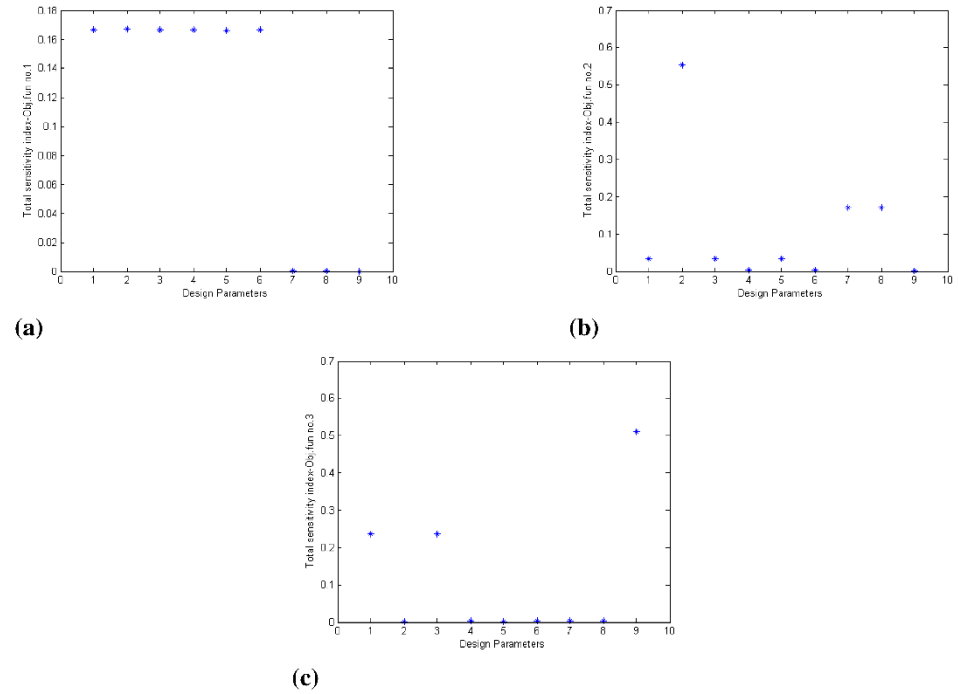


Fig. A6: Sensitivity indices of different objective function with respect to the design parameters: a) First eigen value; b) Second eigen value; c) Third eigen value.

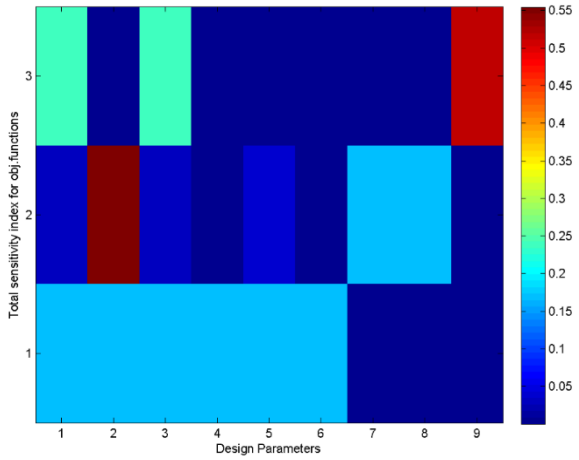


Fig. A7: Sensitivity indices.

The multiobjective optimization starts based on the lower and upper bounds of the parameters, population size, number of generations, elite count, and Pareto fraction settings shown in Fig. A2. The optimization results i.e. Pareto front and Pareto set are automatically saved on excel sheets “Pareto Front.xls” and “Pareto Set.xls”, respectively. Pareto optimized results are also automatically plotted on a 2D basis. Furthermore, the minimum value of each objective function together with the corresponding Pareto sets are being displayed on the screen.

Run the exemplary computer files associated with this example to see more details.

Example 2: *Thermally induced stress intensity factor*

This example is taken from [3, 4]. A cracked plate under thermal loading is shown in Fig. A8.

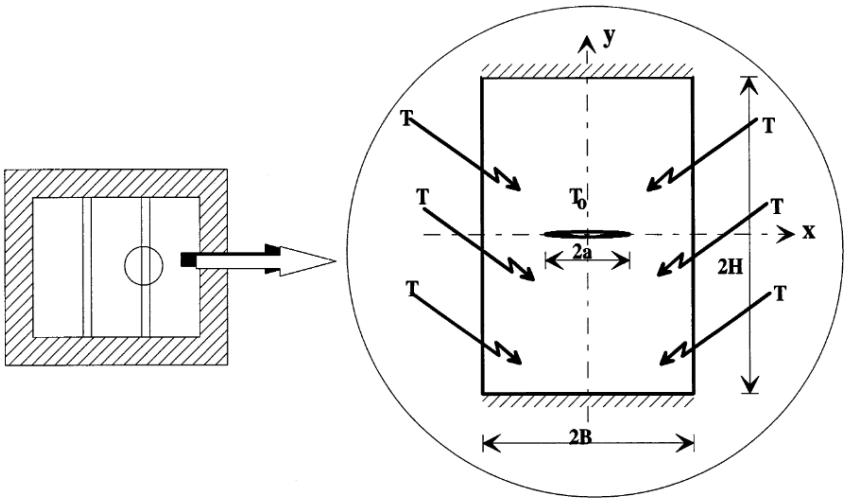


Fig. A8: Cracked plate under thermal loading [4].

The analytical expression for the thermally induced intensity factor of the rectangular plate with a crack size of a is given as follows:

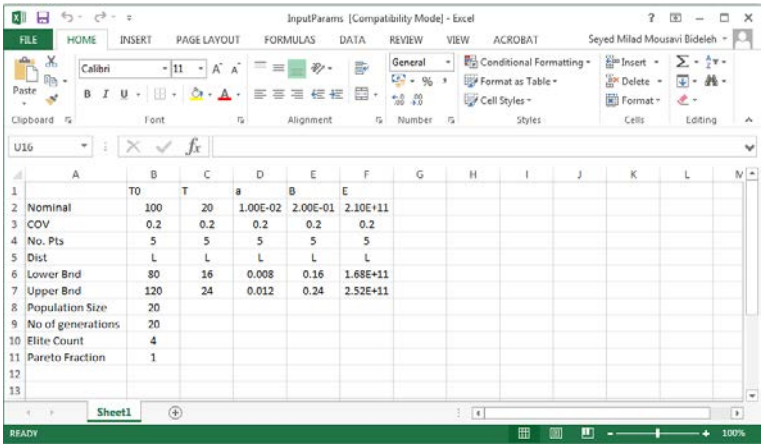
$$K_{\mathrm{IC}}(\mathbf{X}) = -\alpha E(T - T_0)\sqrt{\frac{\pi a}{\cos(\pi a/4B)}}\left[1 - 0.025\left(\frac{a}{2B}\right)^2 + 0.06\left(\frac{a}{2B}\right)^4\right]. \tag{A5}$$

The definition of the parameters and respective values are given in Table A1.

Table A1: Mean and COV of different parameters for GSA.

Variable	Description	Distribution	Mean	COV
T_0	Initial hot temperature	Lognormal	100 ºC	0.2
T	Amphibian cool temperature	Lognormal	20 ºC	0.2
a	Crack size	Lognormal	10 mm	0.2
B	Width of plate	Lognormal	200 mm	0.2
E	Young's module	Lognormal	210 GPa	0.2
α	Thermal expansion coefficient	Deterministic	12.5×10^{-6} ºC^{-1}	-

The target is to study GSA of the thermally induced intensity factor with respect to the parameters listed in Table A1 and use the GSA results to minimize the thermally induced intensity factor. The input parameters file is shown in Fig. A9.



	A	B	C	D	E	F
1		T0	T	a	B	E
2	Nominal	100	20	1.00E-02	2.00E-01	2.10E+11
3	COV	0.2	0.2	0.2	0.2	0.2
4	No. Pts	5	5	5	5	5
5	Dist	L	L	L	L	L
6	Lower Bnd	80	16	0.008	0.16	1.68E+11
7	Upper Bnd	120	24	0.012	0.24	2.52E+11
8	Population Size	20				
9	No of generations	20				
10	Elite Count	4				
11	Pareto Fraction	1				
12						
13						

Fig. A9: Input parameters for GSA and optimization of K_{IC}.

The MBSD code for this example is shown in Fig. A10:

```
function OF = MBSD(X)
global m indx indxOF d0 iter optresult
iter

if length(X) == m
%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for GSA
%%%%%%%%%%%%%%%%%%%%%%%%%
    d = X;
    iter = iter+1;
else
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    d = d0;
    for i = 1:length(X)
        d(indx(i)) = X(i);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% The MBSD code should be written here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
T0 = d(1);
T = d(2);
a = d(3);
B = d(4);
E = d(5);
Alpha = 12.5e-6;
KIC = -Alpha*E*(T-T0)*sqrt(pi*a/cos(pi*a/(4*B)))*(1-
0.025*(a/(2*B))^2+0.06*(a/(2*B))^4);
OF = KIC';

if length(X) ~= m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
OFOpt = zeros(1,length(indxOF));

   for i = 1:length(indxOF)
        OFOpt(i) = OF((indxOF(i)));
   end

optresult(iter,:) = [X OF];
OF = OFOpt;
iter = iter+1;

xlswrite('optresults.xls',optresult);
end
```

Fig. A10: The MBSD code for thermally induced stress intensity factor example.

By running the SAMO, the total sensitivity index shown in Fig. A11 is obtained.

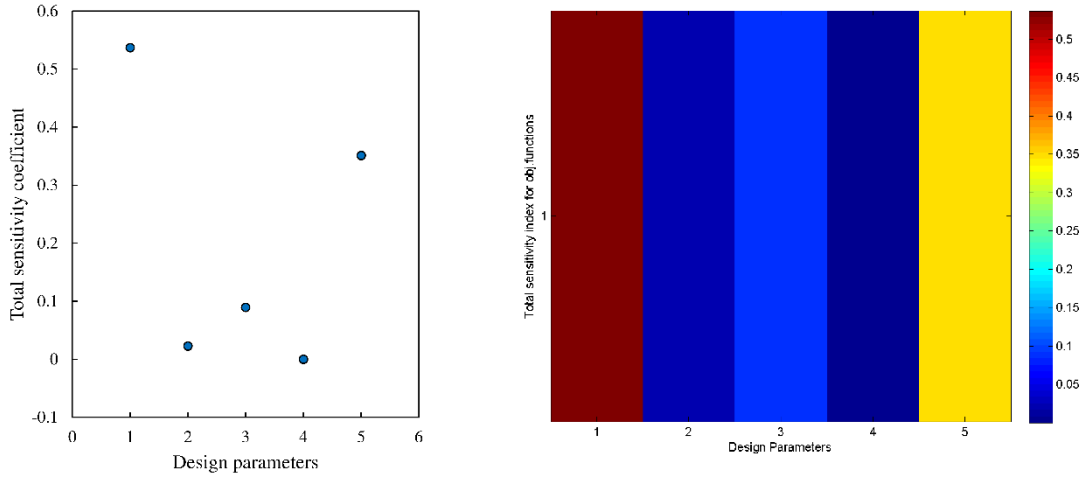


Fig. A11: Total sensitivity index for K_{IC}.

It can be seen that there is an excellent agreement with the values reported in [3].

To minimize the stress intensity factor K_{IC}, one can continue with SAMO to carry out the optimization. This is done by entering "y" or "Y" in the MATLAB command line as follows:

```
To proceed to optimization press Y, to return press N? y
```

```
Please enter the design parameters index number for optimization in an ascending order and vector form.
```

```
E.g. [1 3 4 7]
```

Assume that T_0, a, and E (parameters number 1, 3, and 5) are chosen as the design parameters for optimization of K_{IC}. Therefore, the input vector of design parameters is `[1 3 5]`.

```
Please enter the objective functions index number for optimization in an ascending order and vector form.
```

```
E.g. [1 3 4 7]
```

There is only one objective function here, so the user should enter `[1]`

The optimization results are shown in Fig. A12. It can be seen that the stress intensity factor K_{IC} can be significantly reduced by reducing the initial temperature, crack size, and modulus of elasticity of the plate.

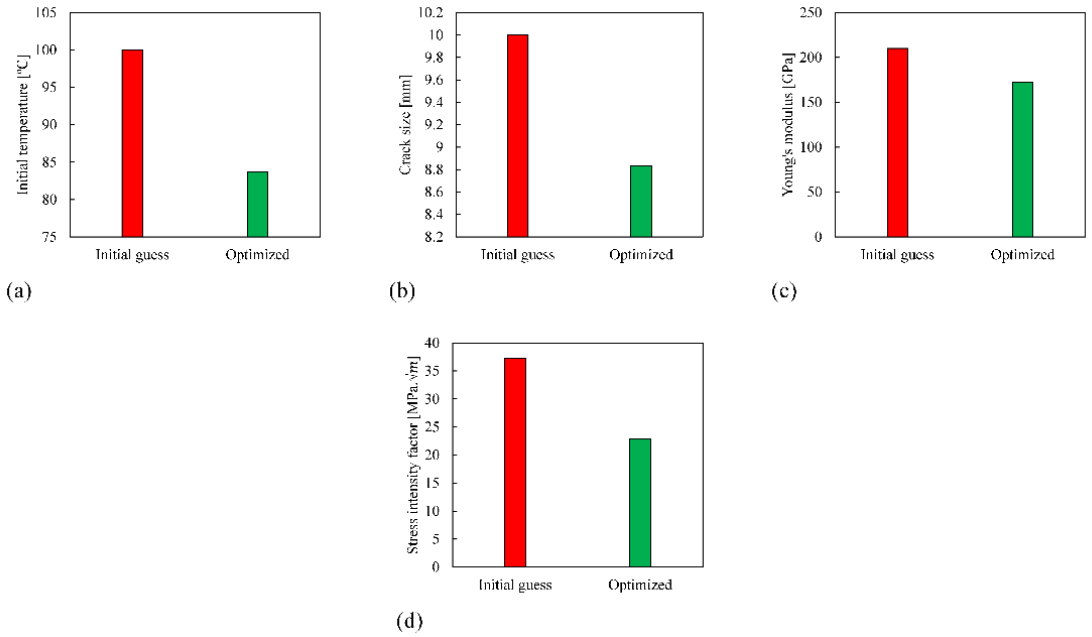


Fig. A12: Optimization results: a) Initial temperature; b) Crack size; c) Modulus of elasticity; d) Stress intensity factor.

Example 3: *Ride comfort and safety of a quarter car vehicle model*

This example is taken from [5]. A quarter car vehicle model is shown in Fig. A13. The equations of motion in matrix form for this vehicle model are given as:

$$M\ddot{\mathbf{x}} + C\dot{\mathbf{x}} + K\mathbf{x} = \mathbf{F}\,, \tag{A6}$$

where, $\mathbf{x}=[x_u,x_s]^T$, and $\dot{x}$, $\ddot{x}$ are the respective time derivatives. The mass matrix (M), damping matrix (C), stiffness matrix (K), and vector of external forces ($\mathbf{F}$) are given as:

$$M = \begin{bmatrix} M_u & 0 \\ 0 & M_s \end{bmatrix}, \tag{A7}$$

$$C = \begin{bmatrix} c_t + c_s & -c_s \\ -c_s & c_s \end{bmatrix}, \tag{A8}$$

$$K = \begin{bmatrix} k_t + k_s & -k_s \\ -k_s & k_s \end{bmatrix}, \tag{A9}$$

$$\mathbf{F} = \begin{bmatrix} k_t x_0 + c_t \dot{x}_0 \\ 0 \end{bmatrix}. \tag{A10}$$

Here, M_s (M_u), k_s (k_t), and c_s (c_t) are the sprung (unsprung) mass, stiffness, and damping, respectively.

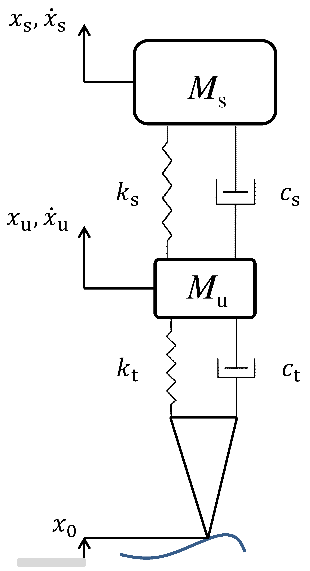


Fig. A13: A quarter car vehicle model.

The road irregularities are modeled as:

$$x_0(t) = a_0 \sin(\omega t)\,,\ \omega = 2\pi v_0 / L\,, \tag{A11}$$

The following objective functions are considered to evaluate the suspension system performance:

$$F_{\ddot{x}_s} = \frac{1}{M_s g}\mathrm{RMS}\left(M_s \ddot{x}_s\right), \tag{A12}$$

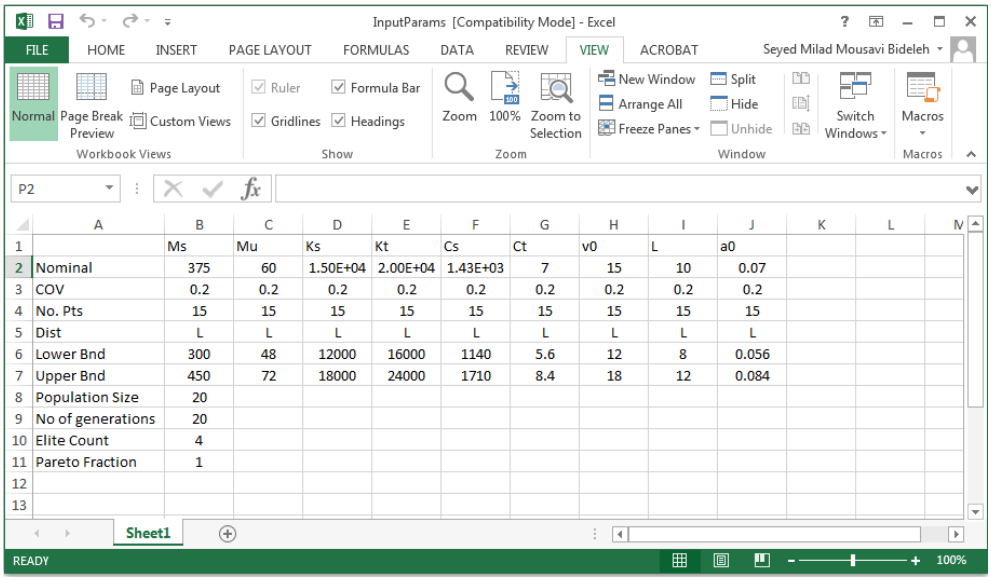
$$F_f = \frac{1}{M_s g}\mathrm{RMS}\left(k_t\left(x_u - x_0\right) + c_t\left(\dot{x}_u - \dot{x}_0\right)\right). \tag{A13}$$

Here, $F_{\ddot{x}_s}$ is the root mean square (RMS) of the sprung mass accelerations and indicates ride comfort. While, F_f is the weighted RMS of the force between the tire and ground and its inverse ($1/F_f$) represents running safety. The structural parameters and problem inputs are given in Table A2.

Table A2: Structural parameters and problem inputs.

M_s [kg]	M_u [kg]	k_s [kN/m]	k_t [kN/m]	c_s [Ns/m]	c_t [Ns/m]	v_0 [m/s]	L [m]	a_0 [m]
375	60	15	20	1425	7	15	10	0.07

The input parameters are shown in Fig. A14. To calculate the objective functions, the MBSD file should be updated as shown in Fig. A15. The equations of motion of the system in the state space form are implemented in the function Quartcar.m. It should be noted that the objective functions are normalized with respect to their initial values. More details can be found in MBSD.m and Quartcar.m files in the appended computer files for this example.



InputParams [Compatibility Mode] - Excel

Seyed Milad Mousavi Bideleh

FILE HOME INSERT PAGE LAYOUT FORMULAS DATA REVIEW VIEW ACROBAT

Normal Page Break Preview Page Layout Custom Views Workbook Views

Ruler Formula Bar Gridlines Headings Show

Zoom 100% Zoom to Selection Zoom

New Window Arrange All Freeze Panes Split Hide Unhide Window

Switch Windows Macros Macros

P2

	A	B	C	D	E	F	G	H	I	J
1		Ms	Mu	Ks	Kt	Cs	Ct	v0	L	a0
2	Nominal	375	60	1.50E+04	2.00E+04	1.43E+03	7	15	10	0.07
3	COV	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2	0.2
4	No. Pts	15	15	15	15	15	15	15	15	15
5	Dist	L	L	L	L	L	L	L	L	L
6	Lower Bnd	300	48	12000	16000	1140	5.6	12	8	0.056
7	Upper Bnd	450	72	18000	24000	1710	8.4	18	12	0.084
8	Population Size	20								
9	No of generations	20								
10	Elite Count	4								
11	Pareto Fraction	1								
12										
13										

Sheet1

READY 100%

Fig. A14: Input parameters for GSA of quarter car vehicle model.

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% The MBSD code should be written here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
global inFun1 inFun2 K M C Kt Ct Mu

Ms = d(1);
Mu = d(2);
Ks = d(3);
Kt = d(4);
Cs = d(5);
Ct = d(6);
v0 = d(7);
L = d(8);
a0 = d(9);
g = 9.81;

M = [Mu 0;0 Ms];
K = [Kt+Ks -Ks;-Ks Ks];
C = [Ct+Cs -Cs;-Cs Cs];

T = linspace(0,10,100);
z0 = zeros(1,4);

inFun1 = @(t)(a0*sin(2*pi*v0/L.*t));
inFun2 = @(t)(a0*2*pi*v0/L*cos(2*pi*v0/L.*t));

[t,z] = ode45(@Quartcar,T,z0);
zdot = zeros(size(z));

for i = 1:length(t)
       zdot(i,:) = Quartcar(t(i),z(i,:)')';
end

Facc = 1/(Ms*g)*sqrt( 1/(t(end) - t(1)) * trapz(t,Ms*zdot(:,4).^2) );

Fsaf = 1/(Ms*g)*sqrt( 1/(t(end) - t(1)) * trapz(t,(Kt*(z(:,1)-
a0*sin(2*pi*v0/L*t))+Ct*(z(:,3)-(a0*2*pi*v0/L*cos(2*pi*v0/L.*t)))).^2) );

OF = [Facc/0.011547434;(1/Fsaf)/4.551818206]';
```

Fig. A15: Implementing the multibody dynamics required to evaluate comfort and safety of the quarter car model.

```
To proceed to optimization press Y, to return press N? y↵
Please enter the design parameters index number for optimization in an ascending order and vector form.
E.g. [1 3 4 7]
```

The total sensitivity indices are shown in Fig. A16. The sprung mass parameters M_s, K_s, and C_s are considered as inputs for optimization.

Therefore, the input to optimization is [1 3 5].

```
Please enter the objective functions index number for optimization in an ascending order and vector form.
E.g. [1 3 4 7]
```

To improve ride comfort and safety simultaneously, both objective functions are chosen for optimization, so the input is [1 2].

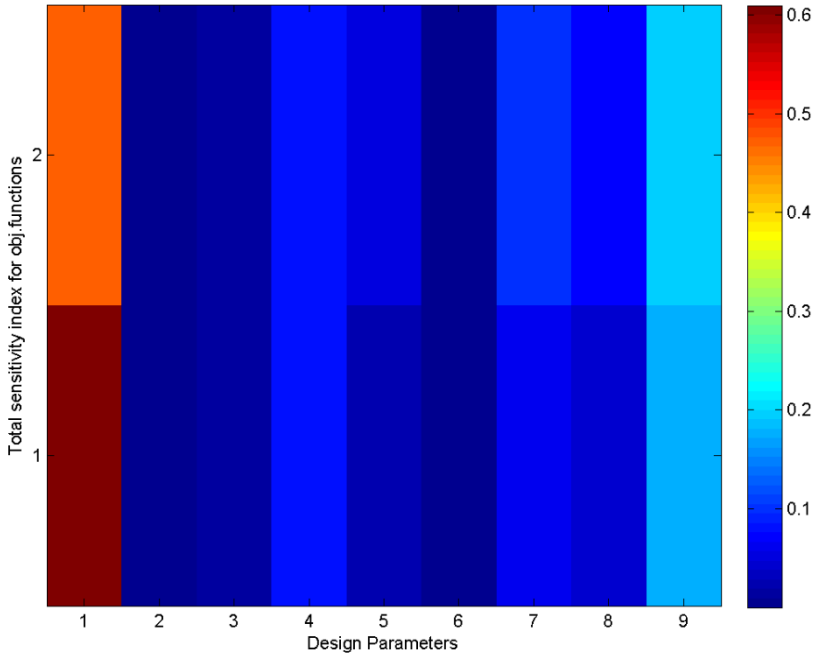


Fig. A16: Total sensitivity indices for the quarter car vehicle model.

The Pareto optimized results are shown in Figs. A17, A18.

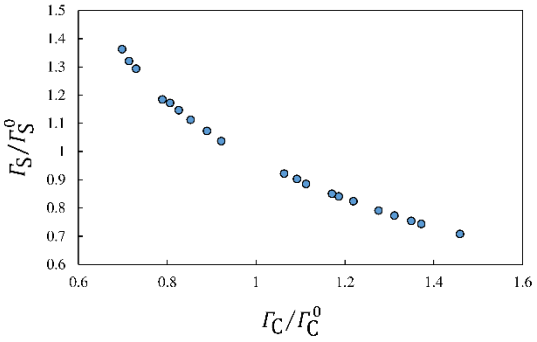


Fig. A17: Pareto front.

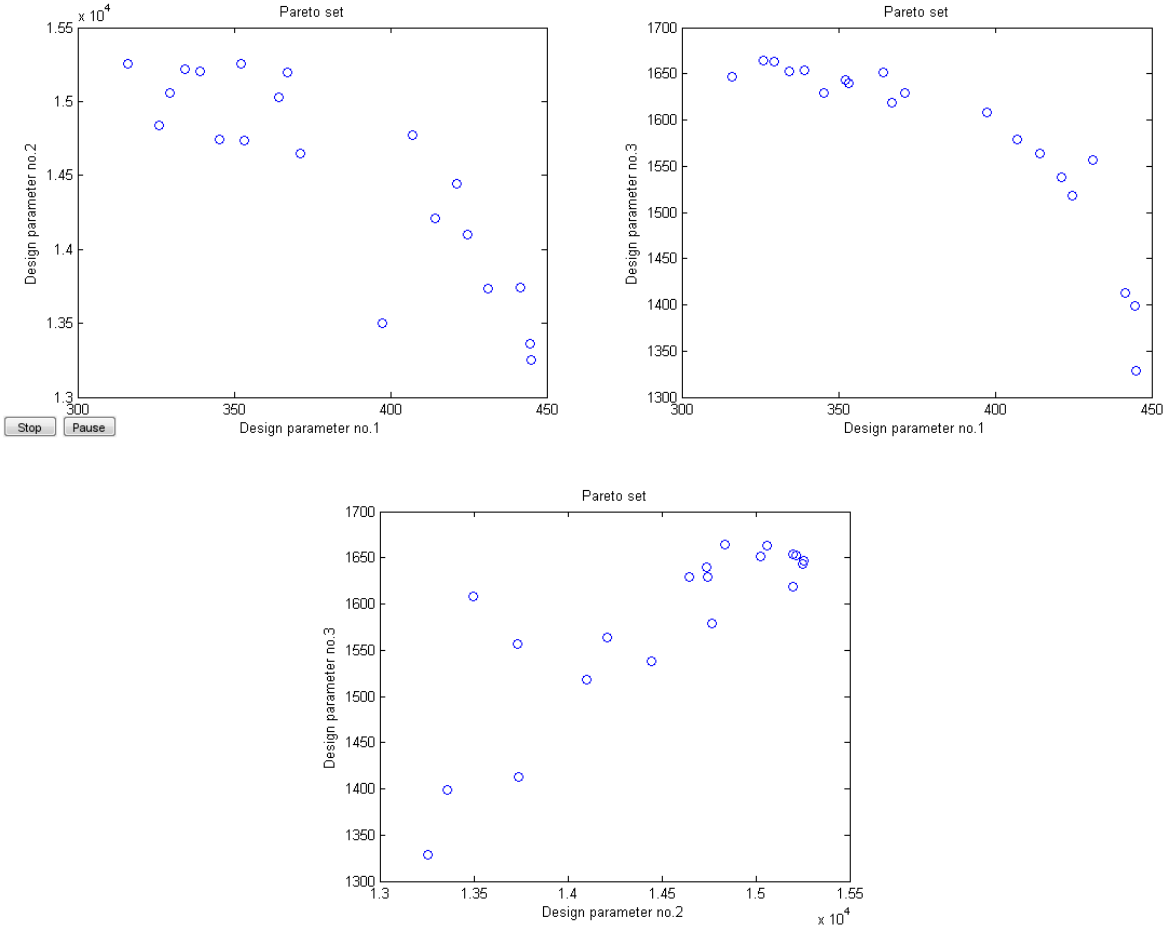


Fig. A18: Pareto set.

Example 4: *GSA of a high speed bogie dynamics w.r.t. suspension components*

This example is chosen to show the usage of SAMO in co-simulation with other software like multibody dynamics software SIMPACK. Similar approach might be used for other commercial software that are able to communicate with MATLAB or MATLAB/SIMULINK in a co-simulation interface.

In this example, effects of suspension system components of a bogie system of a high speed train on vehicle dynamics is investigated using SAMO. The bogie in question is shown in Fig. A19. It constitutes of two wheelsets, four axle boxes, and a bogie frame. All these components are rigid and have six DOFs except for the axle boxes that only allow a single rotation around the wheelset axle. Therefore, the bogie has a total of 22 DOFs.

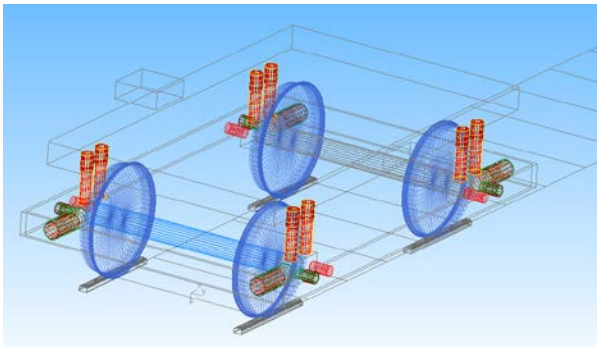


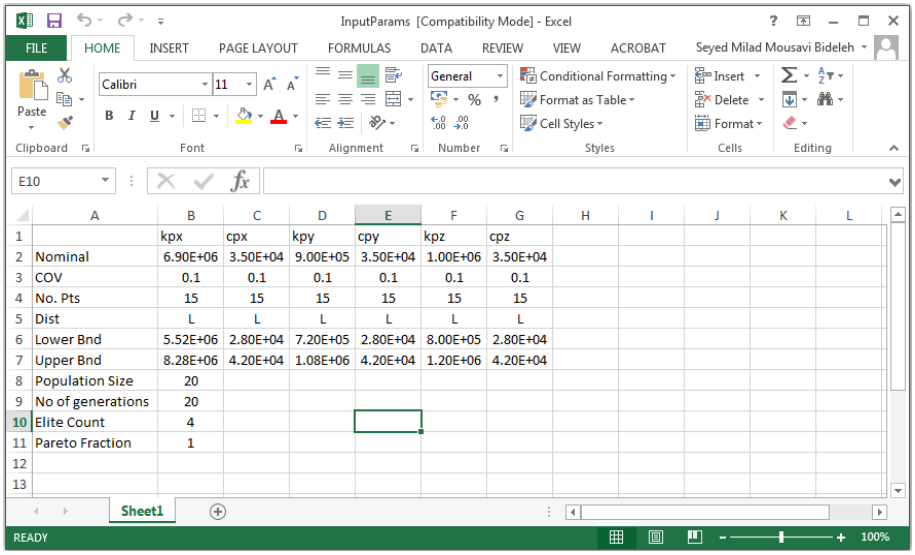
Fig. A19: Bogie model.

In addition to the rigid components, the bogie contains a set of flexible primary suspension elements that are listed in Table A3. All springs and dampers are modeled as point to point linear stiffness and damping, respectively. It should be noted that there are two parallel and equal vertical and lateral suspension spring and damper components. As an example, the stiffness of each of the vertical springs is equal to $K_{pz}/2$.

Table A3: Suspension components.

1	K_{px}	Long prim stiffness
2	C_{px}	Long prim damping
3	K_{py}	Lat prim stiffness
4	C_{py}	Lat prim damping
5	K_{pz}	Vert prim stiffness
6	C_{pz}	Vert prim damping

The input data file is given in Fig. A20. The lower and upper bounds for optimization are considered as -20% and +20% variation around the mean value, respectively.



	A	B	C	D	E	F	G
1		kpx	cpx	kpy	cpy	kpz	cpz
2	Nominal	6.90E+06	3.50E+04	9.00E+05	3.50E+04	1.00E+06	3.50E+04
3	COV	0.1	0.1	0.1	0.1	0.1	0.1
4	No. Pts	15	15	15	15	15	15
5	Dist	L	L	L	L	L	L
6	Lower Bnd	5.52E+06	2.80E+04	7.20E+05	2.80E+04	8.00E+05	2.80E+04
7	Upper Bnd	8.28E+06	4.20E+04	1.08E+06	4.20E+04	1.20E+06	4.20E+04
8	Population Size	20					
9	No of generations	20					
10	Elite Count	4					
11	Pareto Fraction	1					
12							
13							

Fig. A20: Input data file.

The MBSD MATLAB file is given in Fig. A21.

MATLAB function `SubvarUpd` is implemented to update the sub-variables of the bogie model (the six parameters listed in Table A3). The output of the `SubvarUpd` function is shown in Fig. A22. Note that the format of this file is compatible with the multibody dynamics software SIMPACK and changing the standard format might make this file unreadable. It is clear that suitable input file formats should be generated once dealing with other commercial softwares. It should also be noted that this file should be saved in the directory of SIMPACK model. So, the address of the file in the `SubvarUpd` function should be updated based on the working folder. As an example, the file address in the enclosed help file is:

```
C:\Users\bideleh\Desktop\SAMO SIMPACK Example\PrimarySusp.subvar
```

This line in the `SubvarUpd` function should be updated based on the desired working folder. Furthermore, the generated sub-variable file should be linked to the SubVar files in the SIMPACK model as shown in Fig. A23. Additional settings in the SIMPACK model's search path might also be necessary. See, SIMPACK documentation for more details [6].

At this stage, the design parameters are updated and it is time to run the SIMPACK model and get the dynamics response of the system to be able to evaluate the objective functions. This is done in a MATLAB-SIMPACK co-simulation interface with the aid of the `simat` module in MATLAB/SIMULINK.

To use this interface, enter `simat` in MATLAB workspace. In older MATLAB versions, it might be necessary to add `simat` module to MATLAB by clicking on `file/set path/Add Folder…`

In the popped-up window select `simat` from `simpack-2017/partners/mathworks`

See SIMPACK documentation for more details.

```
function OF = MBSD(X)
global m indx indxOF d0 iter optresult

iter

if length(X) == m
%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for GSA
%%%%%%%%%%%%%%%%%%%%%%%%
    d = X;
    iter = iter+1;
else
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    d = d0;
    for i = 1:length(X)
        d(indx(i)) = X(i);
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%% The MBSD code should be written here
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
SubvarUpd(d);              % Update SIMPACK model SubVar file (design parameters)

sim('BogieSimModel.mdl');           % Run Simpack model using simat module in simulink

result = load ('C:\Users\bideleh\Desktop\SMOR SIMPACK Example\output\Bogie_Result.mat');
% Load the results from SIMPACK model. Note: The file address must be updated based on the
current folder
t = result.timeInt.time.values;                  % Integration time

LatAcc = result.timeInt.sensorAccTrans.SS_Bogie.y.values; % Lateral accelerations of Bogie

Q = result.timeInt.RS_result.SRS_RWT_Wheelset.ch_001.values;           % Vertical Contact
force acting on the wheelset (Note: This must be updated for different SIMPACK models)

Y = result.timeInt.RS_result.SRS_RWT_Wheelset.ch_002.values;            % Lateral Contact
force acting on the wheelset (Note: This must be updated for different SIMPACK models)

LatDisp = result.timeInt.RS_result.SRS_RWT_Wheelset.ch_004.values;         % Lateral
displacement of the wheelset (Note: This must be updated for different SIMPACK models)

RMSA = sqrt( 1/(t(end) - t(1)) * trapz(t,LatAcc.^2) );
RMSQ = sqrt( 1/(t(end) - t(1)) * trapz(t,Q.^2) );
RMSY = sqrt( 1/(t(end) - t(1)) * trapz(t,Y.^2) );
RMSd = sqrt( 1/(t(end) - t(1)) * trapz(t,LatDisp.^2) );

OF = [RMSA RMSQ RMSY RMSd];
if length(X) ~= m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% This part is for Multiobj Optimization
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
OFOpt = zeros(1,length(indxOF));

   for i = 1:length(indxOF)
        OFOpt(i) = OF((indxOF(i)));
   end
optresult(iter,:) = [X OF];

OF = OFOpt;
iter = iter+1;

xlswrite('optresults.xls',optresult);
end
```

Fig. A21: The MBSD file.

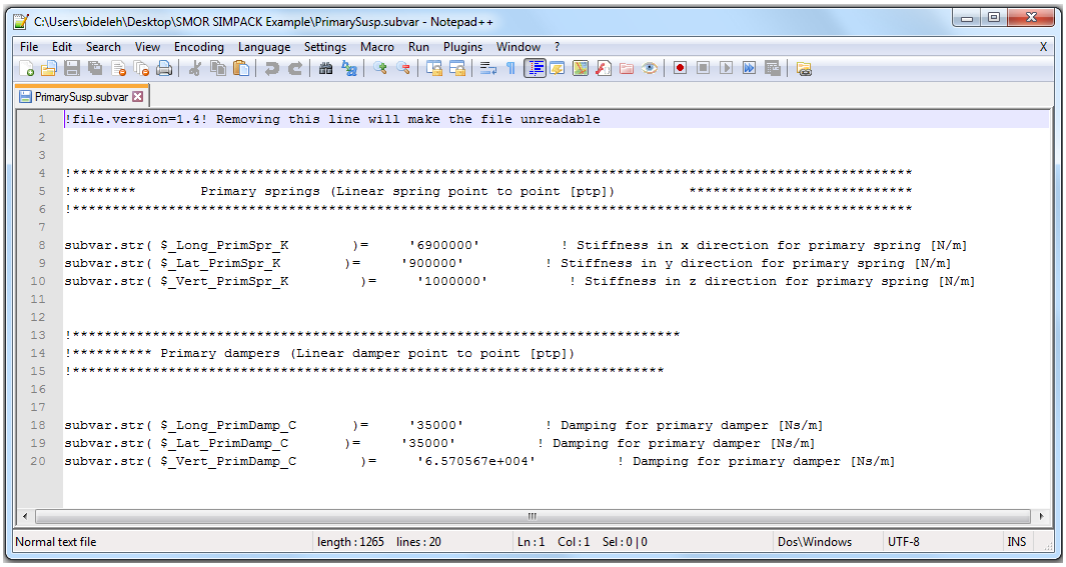


Fig. A22: SIMPACK sub-variable file.

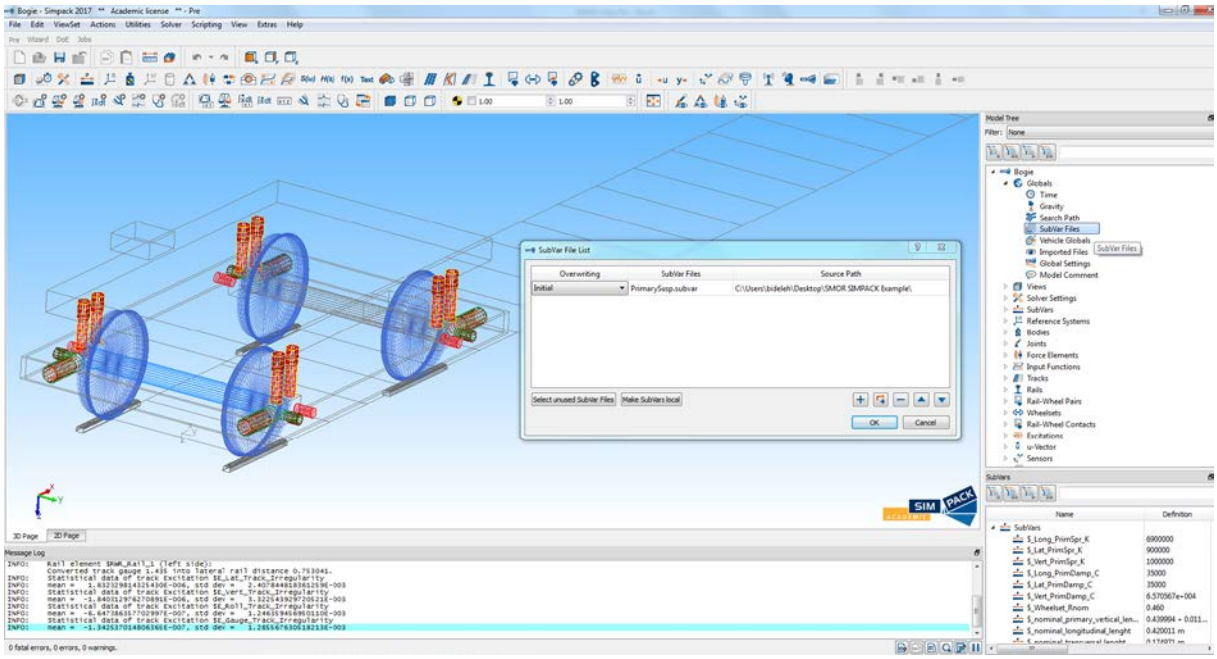


Fig. A23: Link the sub-variable file to the SIMPACK model.

It should be noted that MATLAB function `pathdef.m` will be generated once `simat` is added to MATLAB directory. This file must be removed if you are using MATLAB version 2012 or higher. The `simat` module in SIMULINK is shown in Fig. A24.

To prepare the SIMPACK model for a co-simulation, in SIMPACK GUI select

`Solver/Co-Simulation/Start Command Server`

Getting back to the `simat` module in SIMULINK (Fig. A24), by double clicking on the block the address of the bogie model file in SIMPACK should be linked to the address tab of the `simat` block. See, `BogieSimModel.mdl` and SIMPACK documentation for more details on running co-simulations in MATLAB/SIMULINK-SIMPACK interface.

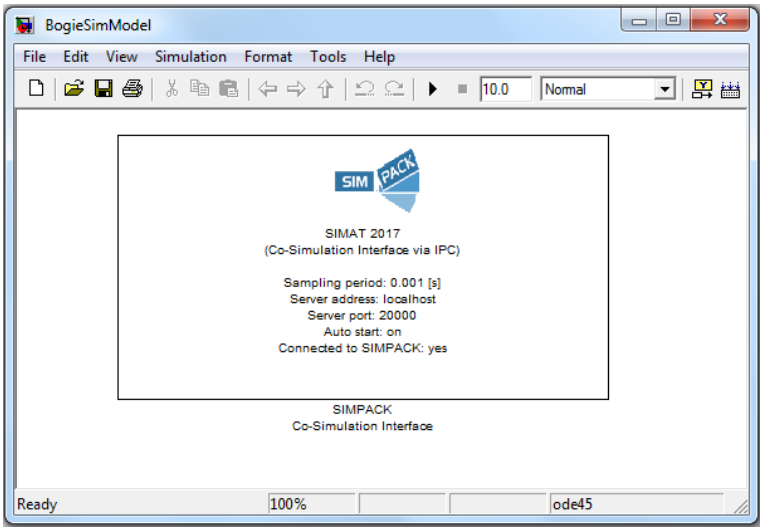


Fig. A24: MATLAB/SIMULINK-SIMPACK co-simulation interface.

MATLAB command `sim('BogieSimModel.mdl')` runs the co-simulation from the MBSD file, see Fig. A21.

Based on the desired objective functions, user should decide which parameters should be measured. This can be done online in SIMULINK by sending SIMPACK outputs to the MATLAB workspace and make post-processing in MATLAB. As an alternative, the results (SIMPACK outputs) can be saved on a mat file which is the case considered here. User can decide where to save the mat file.

Once the current co-simulation is done, it is time to evaluate the objective functions. The mat file `Bogie_Result.mat` is loaded as follows (see also Fig. A21) and of course the file address must be linked to the location of the results file.

```
result = load ('C:\Users\bideleh\Desktop\SAMO SIMPACK Example\...
output\Bogie_Result.mat');
```

Simulation time (t), lateral accelerations of bogie frame (LatAcc), vertical contact force (Q) of the leading axle, lateral contact force (Y) of the leading axle, and lateral displacements of the leading wheelset (LatDisp) are then extracted from the respective channels. SIMPACK sbr file can be used to find the respective channel of each parameter in an easier way.

The RMS value of the lateral accelerations of bogie frame (Γ_a), vertical contact force (Γ_Q), lateral contact force (Γ_Y), and lateral displacements of the wheelset (Γ_d) are chosen as four objective functions.

SAMO evaluates the total global sensitivity indices that are shown in Fig. A25.

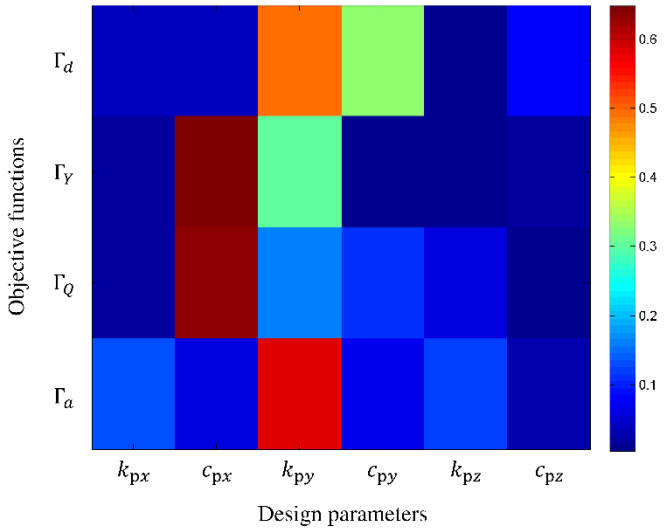


Fig. A25: Sensitivity indices.

After GSA, SAMO asks user to proceed to optimization or not.

```
To proceed to optimization press Y, to return press N? y↵
```

```
Please enter the design parameters index number for optimization in an ascending order and vector form.
```

```
E.g. [1 3 4 7]
```

Here, parameters [2 3 4] i.e. longitudinal damping, lateral stiffness, and lateral damping are used as the design parameters for optimization.

The code also asks for the desired objective functions for optimization

```
Please enter the objective functions index number for optimization in an ascending order and vector form.
```

```
E.g. [1 3 4 7]
```

Here, objective functions [1 3] i.e. the RMS of lateral accelerations of bogie frame (Γ_a) and RMS of the lateral contact forces (Γ_Y) acting on the leading axle are chosen for optimization.

The optimization is carried out based on the settings given in the InputData.xls file. The normalized Pareto front is shown in Fig. A26. It can be seen that with the aid of the Pareto optimized solutions it is possible to improve ride comfort and lateral stability of the vehicle model in questions.

The normalized Pareto set is shown on Fig. A27.

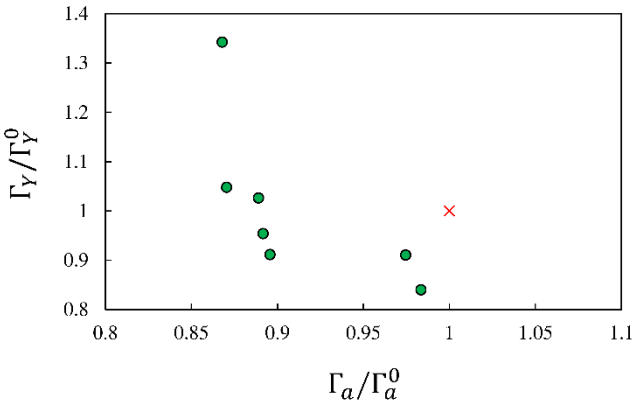


Fig. A26: Normalized Pareto front.

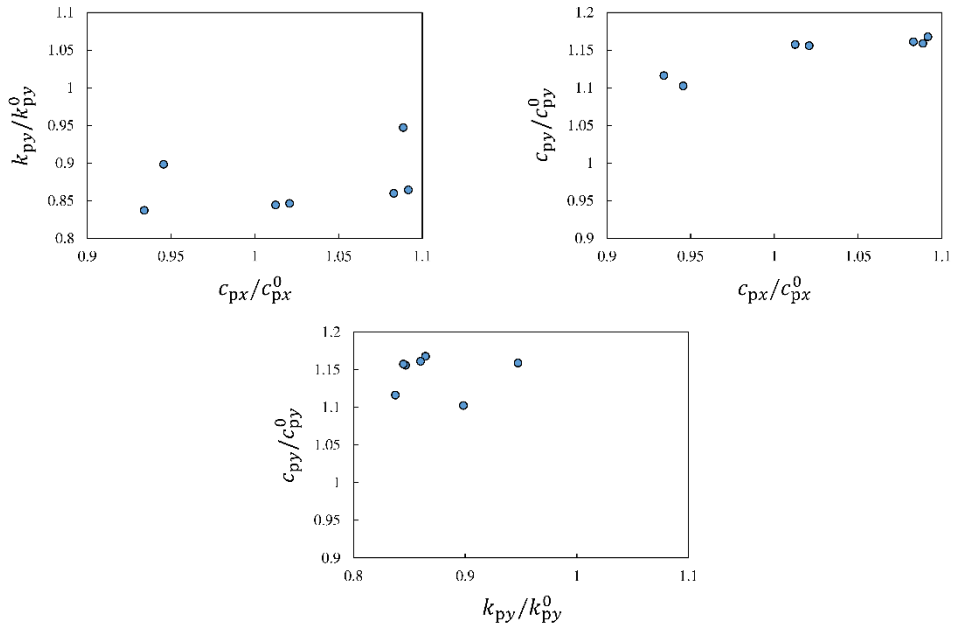


Fig. A27: Normalized Pareto set.

Example 5: *GSA of a one car railway vehicle dynamics w.r.t. suspension components*

The GSA and multiobjective optimization approach considered in SAMO has been also applied to a full scale nonlinear one car railway vehicle model with realistic structural parameters and input data. The GSA results are shown in Fig. A28.

Here, $S_{\Gamma_W}^T$, $S_{\Gamma_C}^T$, $S_{\Gamma_{TS}}^T$, $S_{\Gamma_{St}}^T$, and $S_{\Gamma_{RD}}^T$ indicate the total sensitivity index of wear, ride comfort, track shift force, stability, and risk of derailment, respectively.

More details on GSA and multiobjective optimization of a one car railway vehicle model suspension system using the proposed methods can be found in [7-12].

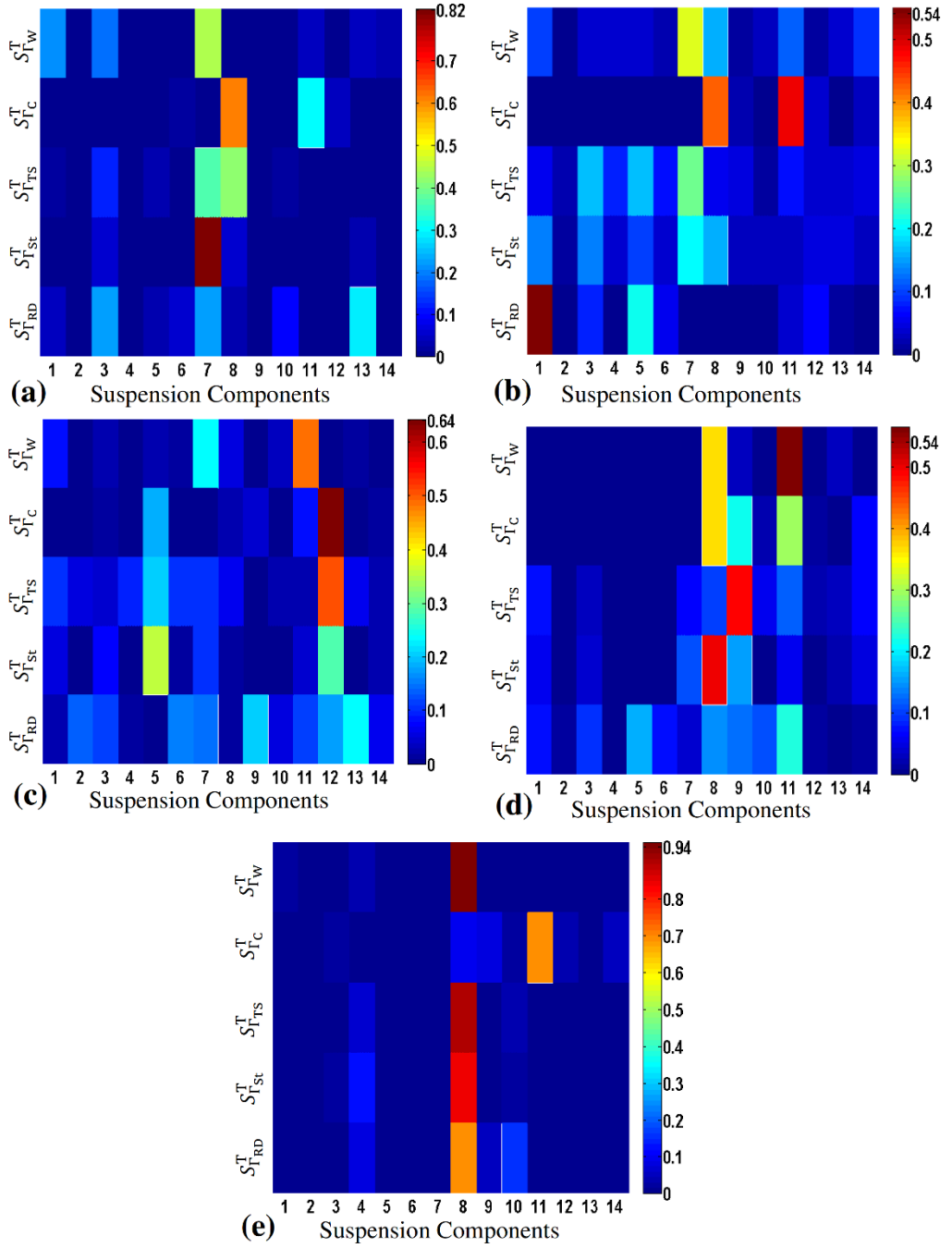


Fig. A28: GSA results for a one car railway vehicle model running with maximum admissible speed on a) R=300 m; b) R=600 m; c) R=1000 m; d) R=3200 m; e) Straight track.

Annex B

Theory

The theories behind GSA and multiobjective optimization approaches used in SAMO are described in details in this section.

B1. Sensitivity analysis

Sensitivity analysis can be carried out either locally or globally. In the following a brief introduction about these two approaches is given.

B1.1 Local sensitivity analysis

In the local methods the effects of design inputs on system response is approximated as partial derivative of an objective function (Γ) with respect to the design parameter (x_i) which is taken around a fixed point x_0. Such methods only take into account the variation of an objective function with respect to a single design parameter at a time. Furthermore, the domain of the input design variables might not be appropriately scanned using the local methods.

B1.2 Global sensitivity analysis

Global sensitivity analysis (GSA) is one of the most prominent steps in design and optimization of multibody systems that can provide informative design insights. In this section, some basic concepts on the GSA formulation are given. In general, different objective functions can be expressed as functions of a set of m independent random variables, i.e. design parameters $\mathbf{d} = [d_1, d_2, ...d_m]^T \in \mathbf{\Omega}$, through the respective deterministic functional relationship $\Gamma = \mathcal{F}(\mathbf{d})$. Where, $\mathbf{\Omega}$ is the domain of input design variables. The mean (μ) and variance (V) of Γ are defined as [3]:

$$\begin{cases} \mu_\Gamma = E_{\mathbf{d}}[\Gamma] = \int_{\mathbf{d}} \mathcal{F}(\mathbf{d})\, f_{\mathbf{d}}(\mathbf{d}) \delta \mathbf{d} \\ V_\Gamma = E_{\mathbf{d}}[(\Gamma - \mu_\Gamma)^2] = E_{\mathbf{d}}\{[\mathcal{F}(\mathbf{d})]^2\} - \mu_\Gamma^2 \end{cases}, \tag{B1}$$

where, $E[.]$ is the expectation operator, and $f_{\mathbf{d}}(\mathbf{d})$ is the joint density of $\mathbf{d}$. Assume that $\mathbf{d}_{-i}$ is a m-1 dimensional sub-vector of $\mathbf{d}$, in which contains all the elements of $\mathbf{d}$ except d_i. Therefore, one can define the following conditional expectation:

$$E_{-i}[\Gamma \mid d_i] = \int_{\mathbf{d}_{-i}} \mathcal{F}(\mathbf{d}_{-i}, d_i)\, f_{\mathbf{d}_{-i}}(\mathbf{d}_{-i}) \delta \mathbf{d}_{-i}, \tag{B2}$$

The primary (S_i) and total (S_{Ti}) sensitivity indices are defined by Eqs. (B3) and (B4), respectively. See e.g. [3, 13, 14] for more details.

$$S_i = \frac{V_i[E_{-i}(\Gamma \mid d_i)]}{V_\Gamma}, \tag{B3}$$

$$S_{Ti} = \frac{E_{-i}[V_i(\Gamma \mid \mathbf{d}_{-i})]}{V_\Gamma}, \tag{B4}$$

It is clear that in order to achieve the global sensitivity indices, multilayer integrals have to be evaluated. This process demands a heavy computational effort. Therefore, it is vital to apply an efficient algorithm to increase the computational proficiency. The M-DRM method can approximate the global sensitivity indices in an efficient and accurate manner.

B1.3 GSA using ANOVA decomposition

The variance-based sensitivity analysis approach applied in SAMO is discussed in this section. After a brief introduction about the method, the simplified sensitivity indices are given.

B1.3.1 Basic concepts

In general, different objective functions (Γ) can be expressed as functions of a set of *n* independent random design variables $\mathbf{X} = [x_1, x_2, ..., x_n]^{\mathbf{T}}$, through the respective functional relationship $\Gamma = \mathcal{F}(\mathbf{X})$.

Based on the ANOVA decomposition concept [15, 16], the function $\mathcal{F}(\mathbf{X})$ can be represented as:

$$\mathcal{F}(\mathbf{X}) = \mathcal{F}^0 + \sum_i \mathcal{F}^i(x_i) + \sum_{i<j} \mathcal{F}^{ij}(x_i, x_j) + ... + \mathcal{F}^{12...n}(x_1, x_2, ..., x_n), \tag{B5}$$

if the function components in (B5) are orthogonal and can be expressed as integrals of $\mathcal{F}(\mathbf{X})$. The following relations can be defined by squaring (B5) and integrating over the domain of input variables [17]:

$$V_{\mathcal{F}} = \int (\mathcal{F})^2 \, \mathrm{d}\mathbf{X} - (\mathcal{F}^0)^2, \tag{B6}$$

and

$$V_{i_1...i_s} = \int (\mathcal{F}^{i_1...i_s})^2 \, \mathrm{d}x_{i_1} ... \mathrm{d}x_{i_s}, \tag{B7}$$

where, $1 \le i_1 < ... < i_s \le n$. The constants $V_{\mathcal{F}}$, and $V_{i_1...i_s}$ are called variances of $\mathcal{F}$, and $\mathcal{F}^{i_1...i_s}$, respectively. The global sensitivity indices are defined as follows:

$$S_{i_1...i_s} = \frac{V_{i_1...i_s}}{V_{\mathcal{F}}}, \tag{B8}$$

The integer *s* is usually referred as the order or dimension of the sensitivity index. It should be noted that:

$$V_{\mathcal{F}} = \sum_{s=1}^{n} \sum_{i_1<...<i_s}^{n} V_{i_1...i_s}, \tag{B9}$$

In a similar manner, the variance and global sensitivity index corresponding to a subset $\mathbf{X}' = [x_{j_1}, x_{j_2}, ..., x_{j_z}]^{\mathrm{T}} \subseteq \mathbf{X}$, $1 \le j_1 < ... < j_z \le n$ are defined by Eqs. (B10) and (B11), respectively.

$$V_{\mathbf{X}'} = \sum_{s=1}^{z} \sum_{(i_1<\ldots<i_s)\in J} V_{i_1\ldots i_s}\,, \tag{B10}$$

$$S_{\mathbf{X}'} = \frac{V_{\mathbf{X}'}}{V_{\mathcal{F}}}\,, \tag{B11}$$

Here, $J = [j_1,\ldots,j_z]$. Assume that $\mathbf{X}''$ is an absolute complement of $\mathbf{X}'$, the total variance and sensitivity index associated with the subset $\mathbf{X}'$, are then defined as follows [17]:

$$V^{\mathrm{T}}_{\mathbf{X}'} = V_{\mathcal{F}} - V_{\mathbf{X}''}\,, \tag{B12}$$

and

$$S^{\mathrm{T}}_{\mathbf{X}'} = \frac{V^{\mathrm{T}}_{\mathbf{X}'}}{V_{\mathcal{F}}}\,, \tag{B13}$$

where, $0 \le S_{\mathbf{X}'} \le S^{\mathrm{T}}_{\mathbf{X}'} \le 1$. The total sensitivity index reflects the total influence of a specific parameter on the system output, including all the possible interactions between that parameter and all the others [18].

B1.3.2 Simplified sensitivity indices

The sensitivity indices expressed based on the HDMR method (ANOVA decomposition) require high-dimensional integrals evaluation. This could be a tough task, especially for complex systems. Therefore, an appropriate approximation is often used to improve the computational efficiency. One of the most effective approaches is cut-HDMR in which the function $\Gamma = \mathcal{F}(\mathbf{X})$ is expressed as a superposition of its values on lines, planes and hyperplanes passing through a fixed reference point (cut center) with coordinates $\mathbf{c} = [c_1,\ldots,c_n]^{\mathrm{T}}$, see e.g. [15]. Based on this concept, Zhang and Pandey, proposed a multiplicative dimensional reduction method (M-DRM) in which a deterministic function $\Gamma = \mathcal{F}(\mathbf{X})$ is approximated as follows [3, 19]:

$$\mathcal{F}(\mathbf{X}) \approx \left[\mathcal{F}(\mathbf{c})\right]^{1-n} \cdot \prod_{i=1}^{n} \mathcal{F}(x_i, \mathbf{c}_{-i})\,, \tag{B14}$$

where, $\mathcal{F}(\mathbf{c})$ is a constant, and $\mathcal{F}(x_i, \mathbf{c}_{-i})$ denotes the function value for the case that all inputs except x_i, are fixed at their respective cut point coordinates. M-DRM (Eq. (B14)) is capable to approximate the function $\Gamma = \mathcal{F}(\mathbf{X})$ with a satisfactory level of accuracy [3, 19] and is particularly useful for approximating the integrals required for evaluation of the sensitivity indices described in the previous section. Using M-DRM and following the procedure described in [3], the primary and higher order sensitivity indices (Eqs. (B11) and (B8), respectively) can be approximated as follows:

$$S_i \approx \frac{\beta_i/\alpha_i^2 - 1}{\left(\prod_{k=1}^{n} \beta_k/\alpha_k^2\right) - 1}\,, \tag{B15}$$

and

$$S_{i_1 \ldots i_s} \approx \frac{\prod_{k=1}^{s}\left(\beta_{i_k}/\alpha_{i_k}^2 - 1\right)}{\left(\prod_{k=1}^{n}\beta_k/\alpha_k^2\right) - 1}, \tag{B16}$$

where, α_k, and β_k are defined as the mean and mean square of the *k*th univariate function, respectively and represented as [3]:

$$\begin{cases} \alpha_k \approx \sum_{l=1}^{N} w_{kl}\mathcal{F}\left(x_{kl}, \mathbf{c}_{-kl}\right) \\ \beta_k \approx \sum_{l=1}^{N} w_{kl}\left[\mathcal{F}\left(x_{kl}, \mathbf{c}_{-kl}\right)\right]^2 \end{cases}, \tag{B17}$$

where, *N* is the total number of integration points, x_{kl}, and w_{kl} are the *l*th Gaussian integration abscissas, and the corresponding weight, respectively.

Finally, the total sensitivity index (given by Eq. (B13)) corresponding to the *i*th parameter (x_i) can be expressed as:

$$S_i^{\mathrm{T}} \approx \frac{1 - \alpha_i^2/\beta_i}{1 - \left(\prod_{k=1}^{n}\alpha_k^2/\beta_k\right)}, \tag{B18}$$

The total sensitivity index given by Eq. (B18) is used in SAMO to reflect the sensitivity.

The accuracy of the sensitivity indices introduced earlier depends on the number of integration points and a convergence study should be accomplished to yield the suitable number of integration points [3, 19]. It should be noted that the total number of function evaluations required for calculating the sensitivity indices using this method is only $n \times N$. Where, *n* is the number of design parameters.

Consequently, in order to accomplish the sensitivity analysis of a system output with respect to an input parameter X_i, a suitable cut point together with a probability distribution have to be chosen. Closed form expressions given by Eqs. (B16), (B17), and (B18) are then utilized to attain the sensitivity indices. The efficiency and applicability of this methodology is already proven through some mathematical and mechanical examples [3].

B1.3.3 Choosing the cut center

An interesting aspect of the cut-HDMR is that in most of the applications with well-defined physical systems, if the cut-HDMR yields a satisfactory level of convergence, the results are independent of the choice of the cut center **c** [15]. However, in practice it is more convenient to consider in service or optimized values of the design parameters as the cut center.

B2. Multiobjective Optimization Using GA

In general, the optimization problem of *m* design parameters $\mathbf{d} = [d_1, d_2, ...d_m]^T \in \mathbf{\Omega}$ (where, $\mathbf{\Omega}$ is the domain of input design variables) with respect to a vector of objective functions $\mathbf{\Gamma}=\mathcal{F}(\mathbf{d})$, that is evaluated from the multibody dynamics response, can be stated as follows:

Determine $\mathbf{d}^*$ and $\mathbf{x}^*(t)$ such that

$$\mathcal{F}(\mathbf{d}^*,\mathbf{x}^*(t))=\min\mathcal{F}(\mathbf{d},\mathbf{x}(t)),\ \mathbf{d}^* \in \mathbf{\Omega}\ , \tag{B19}$$

subject to

$$\Gamma_j(\mathbf{d}) = \mathcal{F}_j(\mathbf{d}) \le \Gamma_j^{\max}, \tag{B20}$$

in which, $\Gamma_j^{\max}$, *j=1, 2,…, n* denote the threshold values, $\mathbf{x}(t)$ is the vector of the design parameters. Genetic algorithm based multiobjective optimization routine in MATLAB is utilized in SAMO to solve the optimization problems. The procedure can be described as follows: in each iteration, MATLAB updates the design parameters file as an input to the multibody dynamics model developed in the MBSD.m file, the dynamic response of the system is then evaluated and the respective objective functions are accordingly attained after a post-processing stage. At this step, the thresholds might be checked to make sure if all the objective functions are within the admissible limits. If at least one of the objective functions violated the thresholds, the vector of the objective functions is penalized to assure that all the Pareto optimized results satisfy the problem constraints. This procedure continues until convergence or the maximum number of generations achieved. In the case of multiobjective optimization problems, the results can be plotted in terms of Pareto set and Pareto front graphs. See e.g. [8, 9, 11] for more details.

B2.1 Genetic algorithm

Genetic algorithm (GA) is an optimization technique which has biological origins and works based on the probabilistic searching. The GA is successfully applied to the multiobjective optimization problem of a variety of complex nonlinear multibody systems such as bogie suspension of high speed trains, see e.g. [20-26].

A GA is generally consists of the following main steps [27]:

- chromosome encoding
- fitness
- selection
- recombination

- evolution

In contradict to the natural GAs which have to follow certain laws observed in nature, the characteristics of the abovementioned steps in a GA with optimization applications are determined by the designer and based on the design requirements.

The ordinary optimization techniques such as Newtown-Raphson and its variants are mostly suitable for convex optimization problems. Such methods utilize local information and as a result might fail to find the global minima. The GA is known to be a method which attains reasonably good global solutions in many applications. However, the initial guess, number of generations, population size, and other settings have to be carefully selected to be able to achieve satisfactory results.

In a multiobjective optimization problem, if no weighting coefficient act on the vector of the objective functions, the GA minimizes every single objective function. Indeed, all the objective functions are treated the same way and there is no need to normalize the objective functions in that case.

When it comes to the complex nonlinear models, it is difficult to impose constraints to the optimization algorithm using MATLAB routine. To overcome such a problem, in the case of a violation of the constraints a penalty factor can be imposed to the objective functions to make sure that the Pareto optimized results remain within the constraints. See, e.g. [8, 9, 11] for more details.

B2.1.1 Chromosome encoding

After selecting the design parameters for optimization, the GA encodes the design parameters. Each encoded design parameter is known as a gene. The complete set of genes (design parameters) that uniquely describe an individual is referred to as a chromosome. Indeed, gene is a particular position or locus in a chromosome.

The particular string representations used for a given problem is known as the GA encoding of the problem. Therefore, the encoded chromosomes are string representations of the solutions to a particular problem. The classical GA uses a bit-string representation to encode the solutions. Bit-string chromosomes composed of a string of genes which contains 0 or 1 characters, see e.g. [27] for more details.

B2.1.2 Fitness

The quality of a chromosome as a solution to a particular problem is determined by the fitness function. Each objective function considered for the optimization problem or a combination of the objective functions can be considered as the fitness function. It is necessary to evaluate the fitness of each particular chromosome. The fitness information are then used to bias the next generations based on the better genes.

B2.1.3 Selection

Based on the fitness, the chromosomes should be selected for recombination and to construct the next generations. In general, the chromosomes which resulted in a better fitness function should have a higher chance to be selected. This might lead to a more highly fit solutions by the upcoming generations. It should be noted that highly fit chromosomes might have a chance to be selected twice or more or even recombined with themselves.

Fitness proportional also known as the roulette wheel is one of the common selection methods. The probability of parenthood (to be selected) in this method is proportional to the fitness. There are many different selection methods such as random stochastic selection, tournament selection, and truncation selection. More details on the selection schemes can be found in [27].

B2.1.4 Recombination

In the recombination, the chromosome of the child is being created using the chromosomes of the parents selected earlier. The main operators of the recombination are known as crossover and mutation. An example of the crossover is shown in Fig. B1. The child properties depend on the crossover point. For instance, in Fig. B1 (a) the child has similar ears to the parent 1 and similar eyes to the parent 2, but in Fig. B1 (b) the child has similar ears to the parent 1 but different eyes from both parents.

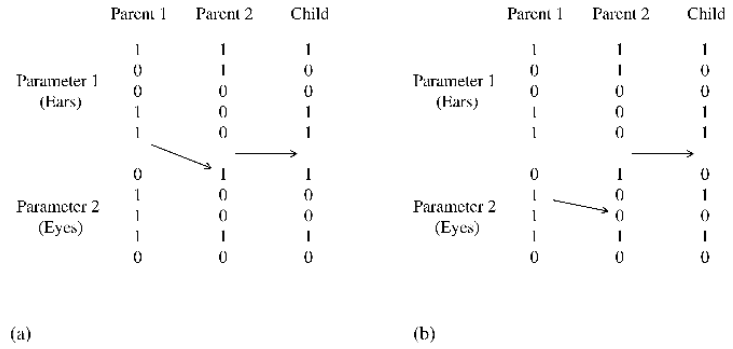


Fig. B1: The crossover example.

Once the child chromosome is generated by the crossover, the GA applies the mutation operator on the resulting chromosome to change one or more properties, see Fig. B2 for example.

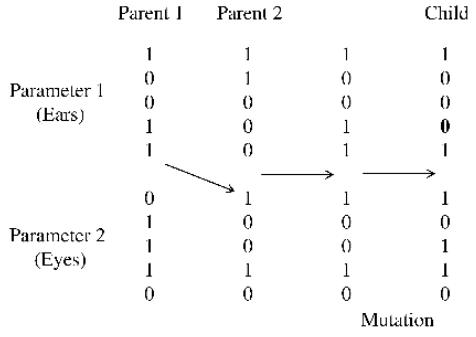


Fig. B2: The mutation example.

How to decide for the crossover and mutation methods to achieve more fitted results depends on the GA settings.

B2.1.5 Evolution

The chromosomes obtained from the previous stages are diverted into the so called successor population. The selection and recombination steps are then repeated until a complete successor population achieved which is going to be considered as the next generation. The GA repeats this process through a number of generations until certain convergence to a best fitness solution or maximum number of iterations achieved.

The evolutionary schemes determine which chromosomes from the source population are eligible to remain unchanged when passing to the successor population. It is vital to employ an appropriate evolutionary scheme. This is usually decided based on the nature of the domain of the input design parameters being searched. One of the most well-known schemes is replacement with elitism. To create the successor population, this scheme preserves the best one or two individuals from the source population and generates the rest through selection and recombination. This method assures that solutions of the highest relative fitness will be appear in the next generation through the selection process, see [27] for more details. The GA flowchart is plotted in Fig. B3.

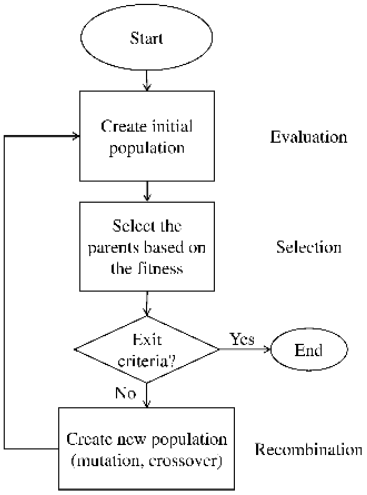


Fig. B3: The GA flowchart.

References

[1] *MATLAB Documentation*: https://se.mathworks.com/help/gads/gamultiobj.html?s_tid=gn_loc_drop.

[2] Chauhan, S., Motor torque calculations for electric vehicle, *International journal of scientific & technology research*, 2015, **4**(8): pp. 126-127.

[3] Zhang, X. and Pandey, M.D., An effective approximation for variance-based global sensitivity analysis, *Reliability Engineering and System Safety*, 2014, **121**: pp. 164-174.

[4] Mohamed, A., Lemaire, M., Mitteau, J.-C., and Meister, E., Finite element and reliability: a method for compound variables — application on a cracked heating system, *Nuclear Engineering and Design* 1998, **185**: pp. 185-202.

[5] Berbyuk, V., *Structural dynamics control*. 2014, Gothenburg, Sweden: Chalmers University of Technology.

[6] *SIMPACK V9.3 documentation*. 2013, INTEC GMBH. Wessling, Germany.

[7] Mousavi Bideleh, S.M., *Multiobjective optimisation and active control of bogie suspension*. 2016, Gothenburg, Sweden: Chalmers University of Technology.

[8] Mousavi Bideleh, S.M. and Berbyuk, V., Global sensitivity analysis of bogie dynamics with respect to suspension components, *Multibody System Dynamics*, 2016, **37**(2): pp. 145-174, DOI: 10.1007/s11044-015-9497-0.

[9] Mousavi Bideleh, S.M., Berbyuk, V., and Persson, R., Wear/comfort Pareto optimisation of bogie suspension, *Vehicle System Dynamics*, 2016, **54**(8): pp. 1053-1076, DOI: 10.1080/00423114.2016.1180405.

[10] Mousavi Bideleh, S.M., Mei, T.X., and Berbyuk, V., Robust control and actuator dynamics compensation for railway vehicles, *Vehicle System Dynamics*, 2016, **54**(12): pp. 1762-1784, DOI: 10.1080/00423114.2016.1234627.

[11] Mousavi-Bideleh, S.M. and Berbyuk, V., Multiobjective optimisation of bogie suspension to boost speed on curves, *Vehicle System Dynamics*, 2016, **54**(1): pp. 58-85, DOI: 10.1080/00423114.2015.1114655.

[12] Mousavi Bideleh, S.M., Robustness analysis of bogie suspension components Pareto optimised values, *Vehicle System Dynamics*, 2017, **55**(8): pp. 1189–1205, DOI: 10.1080/00423114.2017.1305115.

[13] Oakley, J. and O'Hagan, A., Probabilistic sensitivity analysis of complex models:a Bayesian approach, *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 2004, **66**(3): pp. 751-69.

[14] Saltelli, A. and Sobol', I., About the use of rank transformation in sensitivity analysis of model output, *Reliability Engineering and System Safety*, 1995, **50**(3): pp. 225-239.

[15] Rabitz, H. and Alıs, O.F., General foundations of high-dimensional model representations, *Journal of Mathematical Chemistry*, 1999, **25**: pp. 197-233.

[16] Sobol', I.M., Theorems and examples on high dimensional model representation, *Reliability Engineering and System Safety*, 2003, **79**: pp. 187-193.

[17] Sobol', I.M., Global sensitivity indices for nonlinear mathematical models and their Monte Carlo estimates, *Mathematics and Computers in Simulation*, 2001, **55**: pp. 271-280.

[18] Homma, T. and Saltelli, A., Importance measures in global sensitivity analysis of nonlinear models, *Reliability Engineering and System Safety*, 1996, **52**: pp. 1-17.

[19] Zhang, X. and Pandey, M.D., Structural reliability analysis based on the concepts of entropy, fractional moment and dimensional reduction method, *Structural Safety*, 2013, **43**: pp. 28-40.

[20] Mei, T.X., Foo, T.H.E., and Goodall, R.M. Genetic Algorithms for Optimising Active Controls in Railway Vehicles, in *Inst. Elect. Eng. Colloq. Optimization Contr.: Methods Applicat.* 1998. London, U. K., (98/521).

[21] He, Y. and Mcphee, J., Design optimization of rail vehicles with passive and active suspensions: A combined approach using genetic algorithms and multibody dynamics, *Vehicle System Dynamics*, 2002, **37**: pp. 397–408.

[22] Mei, T.X. and Goodall, R.M., Use of multiobjective genetic algorithms to optimize inter-vehicle active suspensions, *Proceedings of the Institution of Mechanical Engineers, Part F: Journal of Rail and Rapid Transit*, 2002, **216**: pp. 53-63.

[23] He, Y. and McPhee, J., A design methodology for mechatronic vehicles: application of multidisciplinary optimization, multibody dynamics and genetic algorithms, *Vehicle System Dynamics*, 2005, **43**(10): pp. 697-733.

[24] Mousavi Bideleh, S.M. and Berbyuk, V. Multiobjective optimization of a railway vehicle dampers using genetic algorithm, in *the ASME 2013 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference IDETC/CIE, Paper DETC2013-12988*. 2013. Portland, Oregon, USA. DOI: 10.1115/DETC2013-12988.

[25] Mousavi Bideleh, S.M. and Berbyuk, V. Optimization of a bogie primary suspension damping to reduce wear in railway operations, in *The ECCOMAS Multibody Dynamics, pp. 1025-1034*. 2013. University of Zagreb, Croatia.

[26] Nejlaoui, M., Houidi, A., Affi, Z., and Romdhane, L., Multiobjective robust design optimization of rail vehicle moving in short radius curved tracks based on the safety and comfort criteria, *Simulation Modelling Practice and Theory*, 2013, **30**: pp. 21-34.

[27] McCall, J., Genetic algorithms for modelling and optimisation, *Journal of Computational and Applied Mathematics*, 2005, **184**(2005): pp. 205-222.