SOME ASPECTS OF OPTIMIZATION IN NON LINEAR DYNAMICS

Abstract

On the basis of a very simple example of crash simulation, different issues concerning optimization in non linear dynamics are reviewed, such as robustness, convergence and choice of the best formulation and/or optimization technique within those available through *optiSlang*. Robust optimization (RDO) is also performed on the example problem.

Contents

Edmondo Di Pasquale

(*edmondo@simtech.fr*)

July 2007

Introduction: a crash course in crash

Crash performance of a structure amounts to the ability to dissipate a given amount of energy while:

- Limiting the injuries to weak users
- Sustaining reasonably little damage
- Minimizing the cost (mass for a given material and technology)

In today' automotive industry, crash performance is measured by standardized tests, imposed by government bodies or de facto regulators (NTHS, EURONCAP, insurance companies).

These tests may vary widely, but they have some common features …

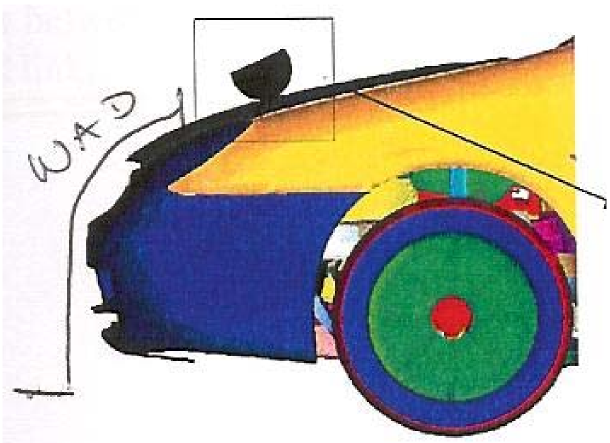
… one of which is that the structure must demonstrate the ability to :

•Dissipate (relatively) small amount of energies with little aggressiveness and/or local damage.
•Dissipate (relatively) large amount of energy avoiding catastrophic effects.

The trade-off between these two conflicting requirements leads naturally to a constrained optimization problem.

For automotive crash, we require:

- Limited (repairable) damage for insurance tests
- Absence of occupant injuries for 65 Km/h test

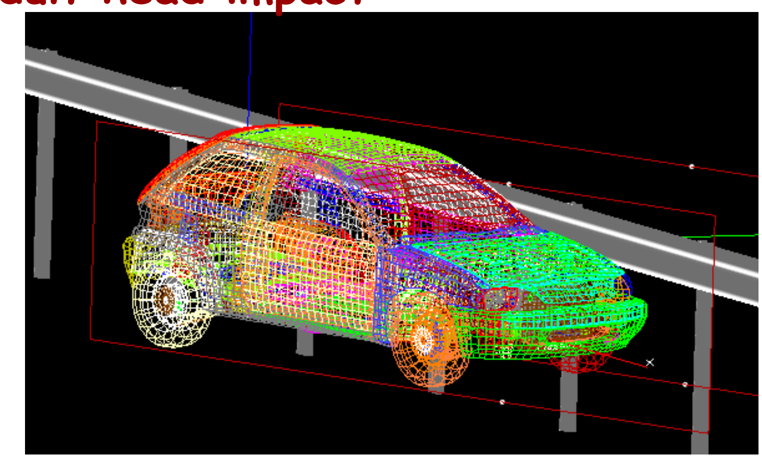


For pedestrian head impact, the same requirement (HIC) amounts to:

- A very flexible structure for child head impact
- A relatively stiff structure for adult head impact

For road barrier crash, we require:

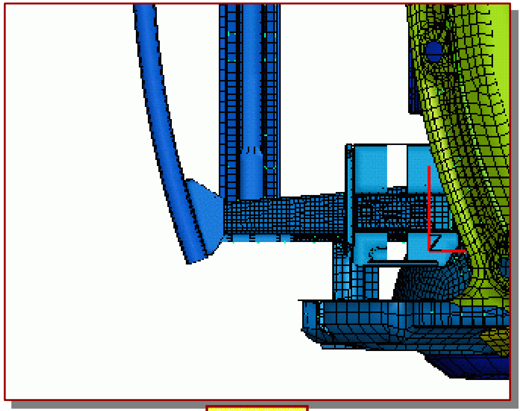
- Low acceleration for light vehicle impact
- High restraining capability for truck/bus impact

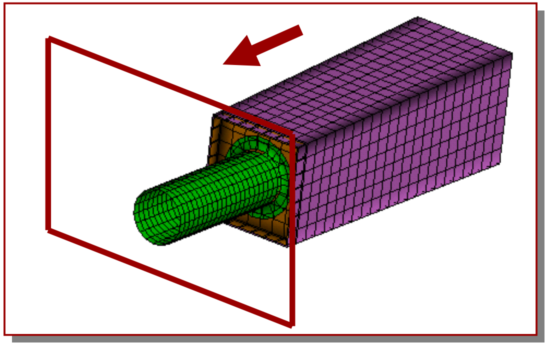


The actual structures designed to sustain these impacts are very complex and diverse.

However, many such structures can be modeled as a sequence of structural elements with increasing stiffness and energy dissipation capabilities.

In the following, we call it a ***multi-stage structure***.



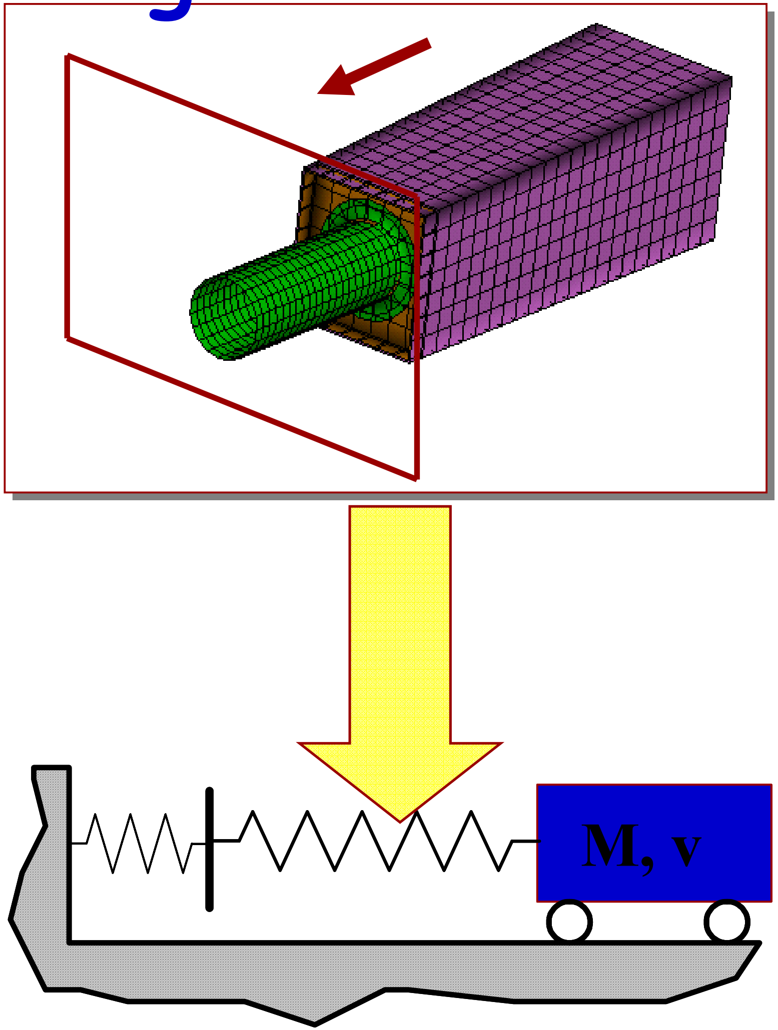


Simplified multi-stage structure

A multi-stage system is modeled with 2 non-linear springs.

Each spring may have different failure modes.

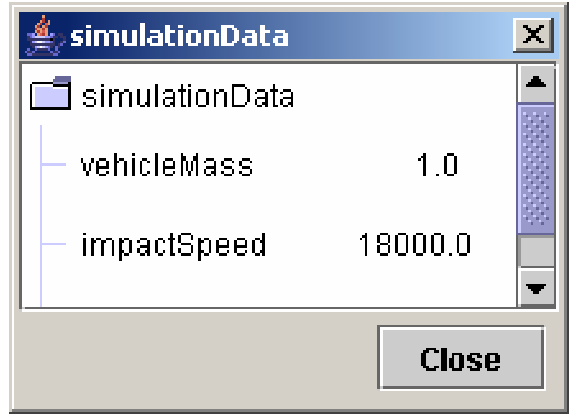
The rest of the vehicle is modeled by a lumped mass at an initial speed.



For the simplified modeling we use a in-house application, developed using ENKIDOU, a SimTech –proprietary library of Java components for the development of vertical applications.

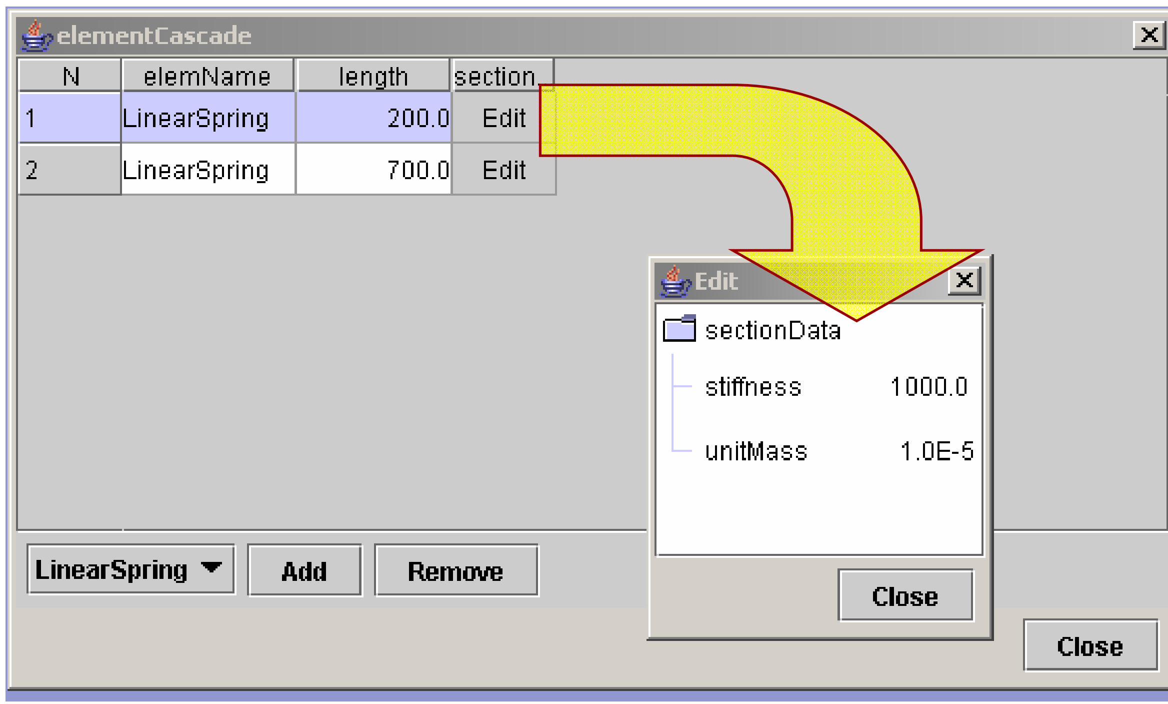
The impact parameters are the mass and initial speed of the impacting vehicle (impactor).

The multi-stage structure can be composed of any number of elements with different characteristics.

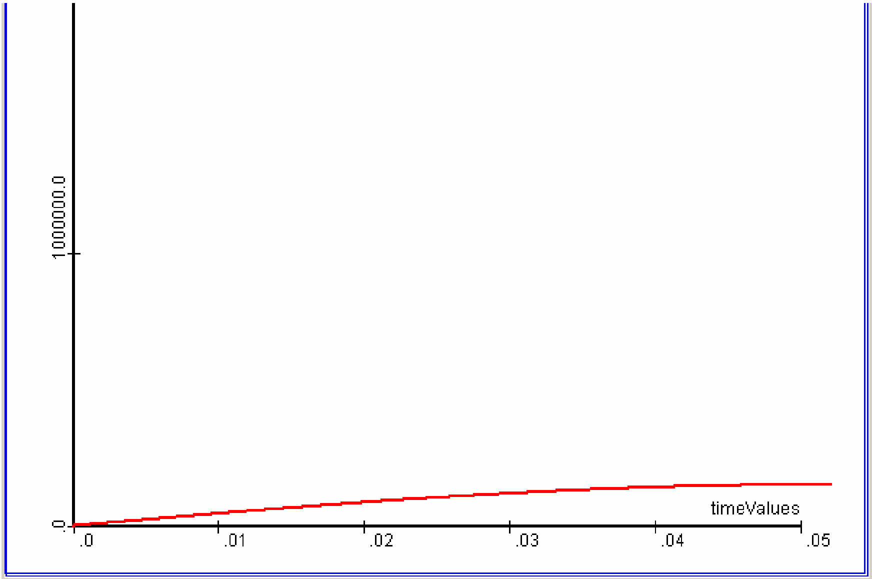


For the present study, we consider a two-stage structure made out of linear spring failing at a given deformation (length).

Non linearity comes from failure.

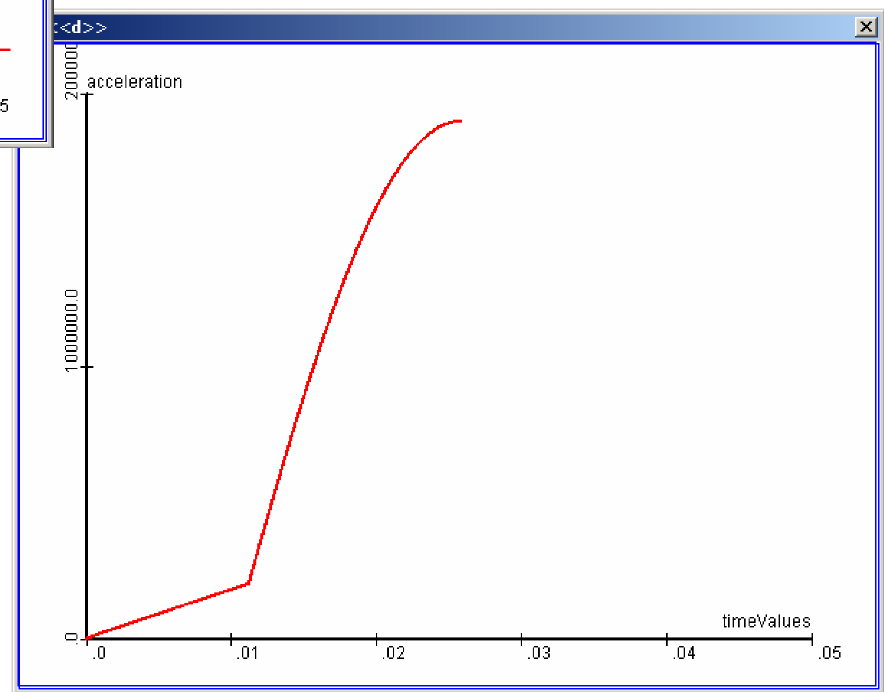


This simple model captures some of the main features of the impact phenomena under investigation.

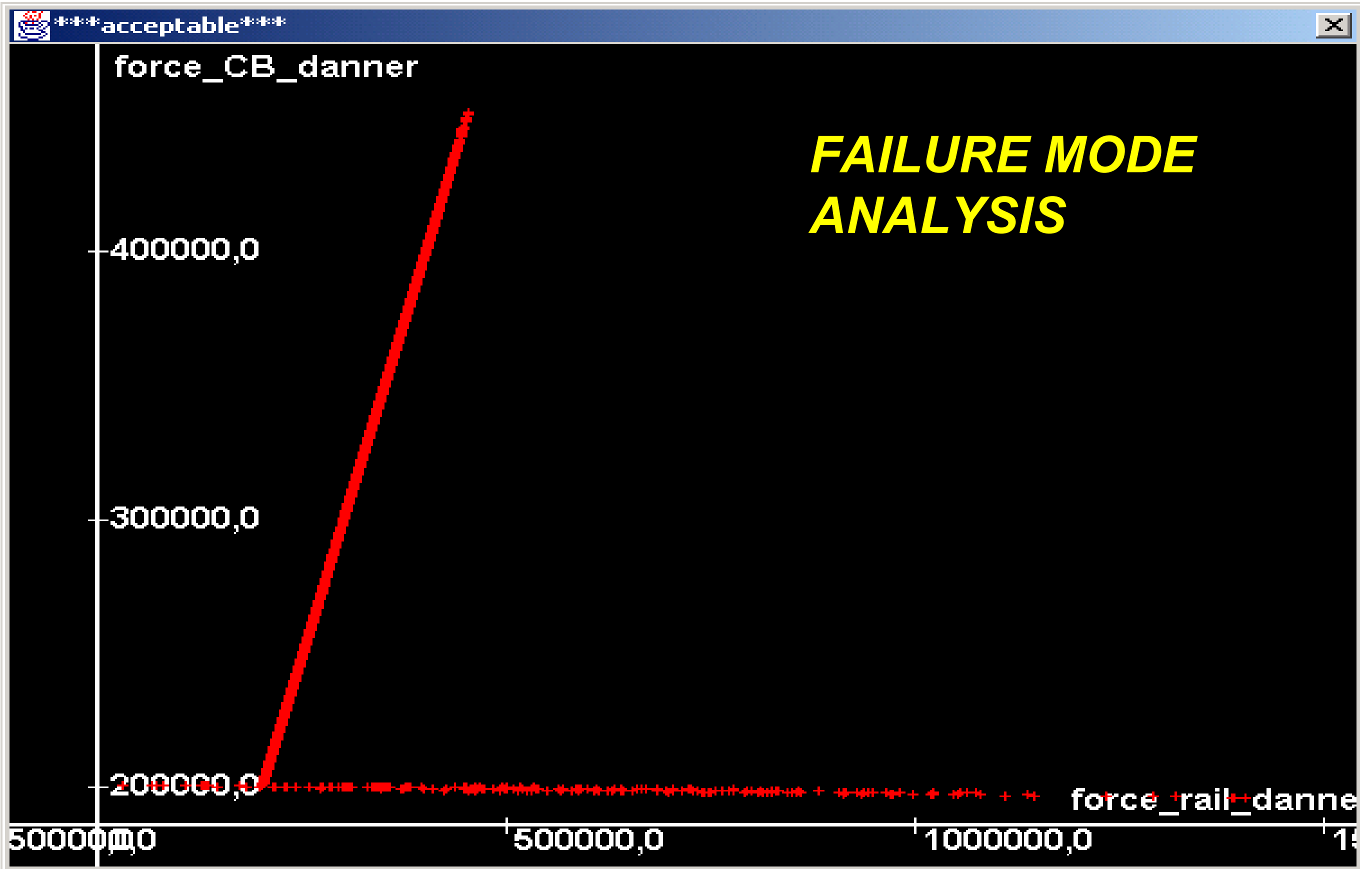


At low speed (5 m/sec) we have a one long event, where the first spring dissipate energy with low acceleration.

At high speed (20 m/sec) we have two events.
First, the soft spring is crushed.
The stiff spring dissipate the energy with high acceleration.

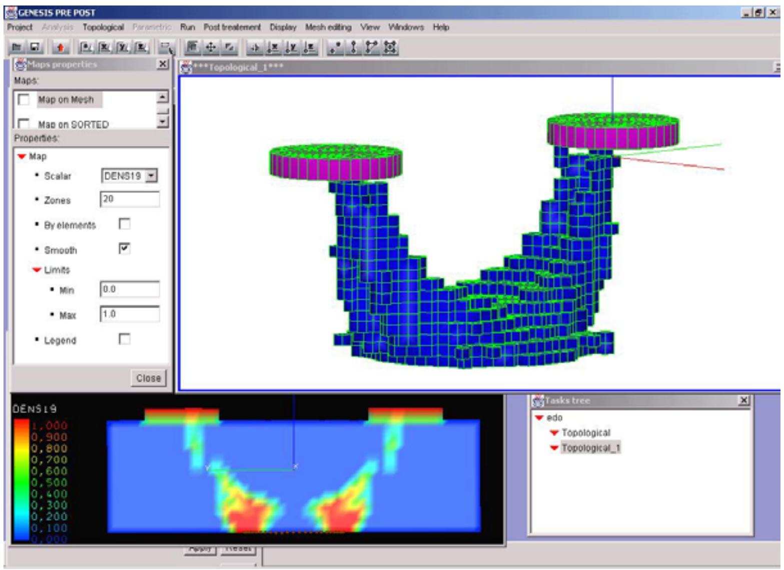


If we put more non linearity, we get more complex behavior. In particular, this simple model may represent bifurcations (different failure modes)

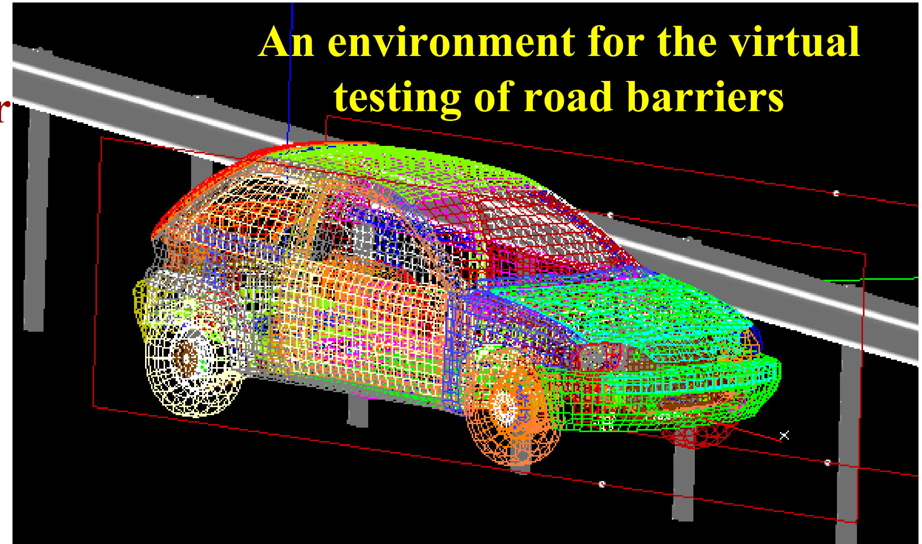


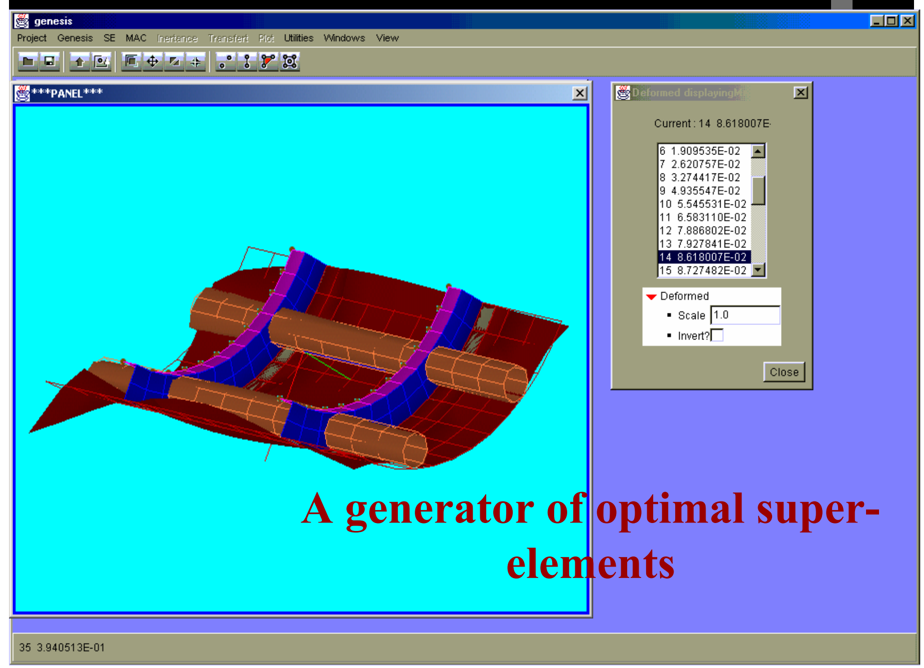
Some other examples of ENKIDOU vertical applications

A specialized (pre-) and post-processor for VR&D GENESIS



An environment for the virtual testing of road barriers





A generator of optimal super-elements

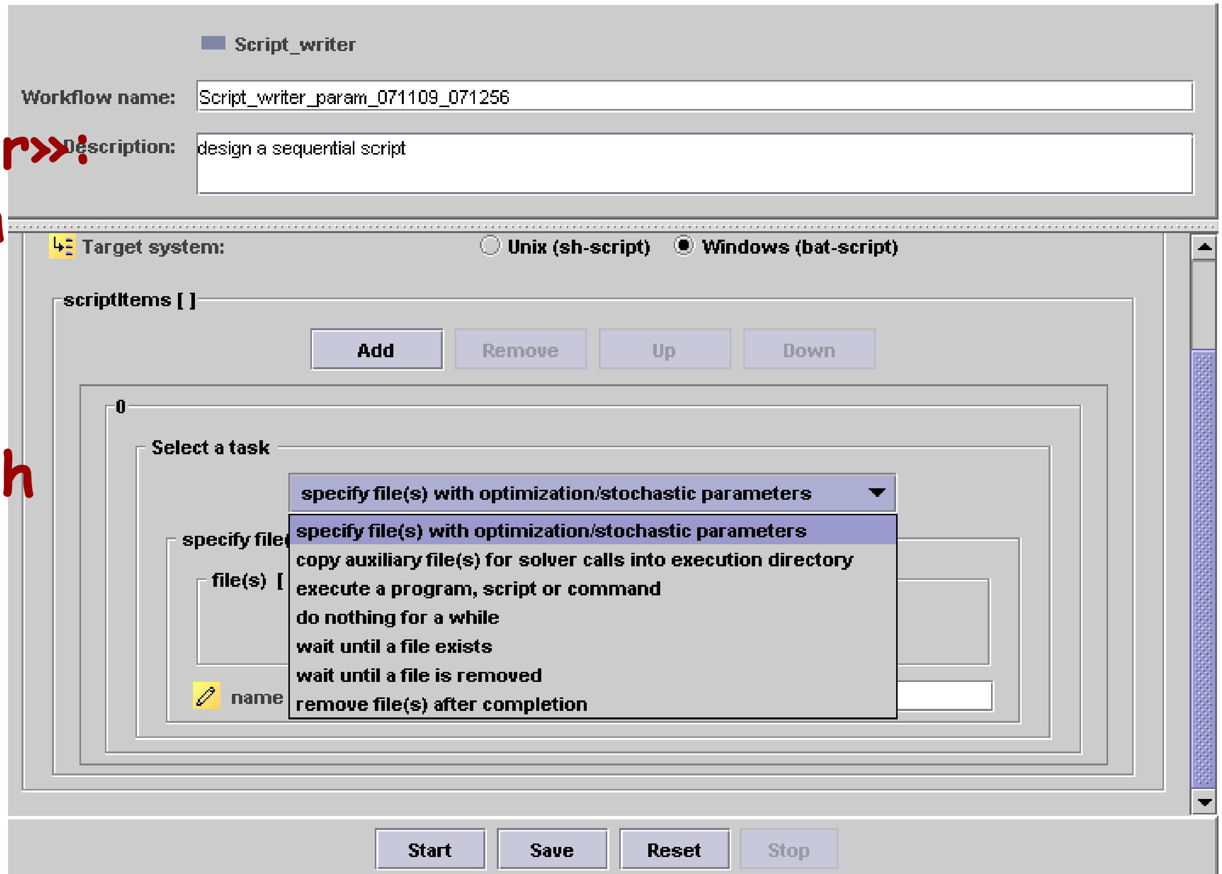
optiSlang SET-UP

Optimization problem set-up

The simulation process for our problem involves two crash simulation:

Element «danner»:
low speed crash
(5 m/sec)

Element «bfd»:
high speed crash
(20 m/sec).



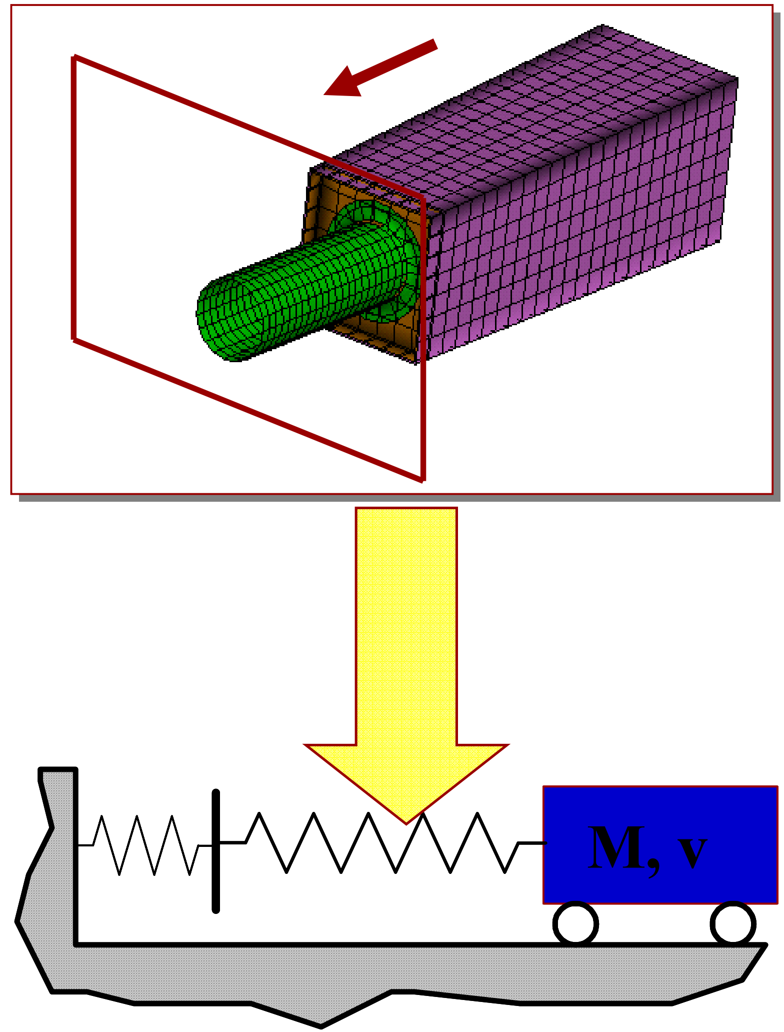
This specific set-up is proper to an automotive crash design problem.

Hence, we call element 1

"crash box"

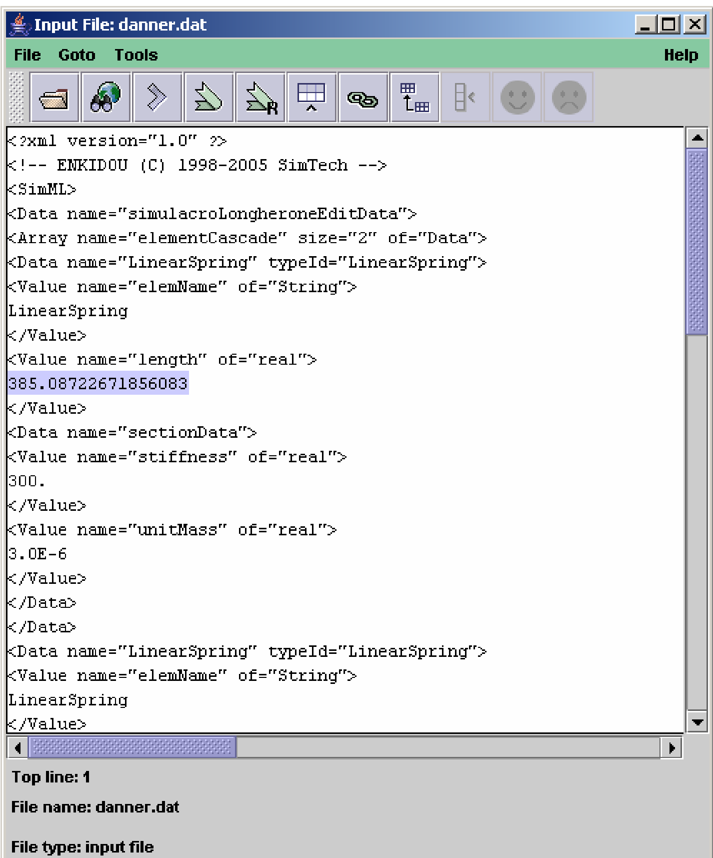
and element 2

"rail"

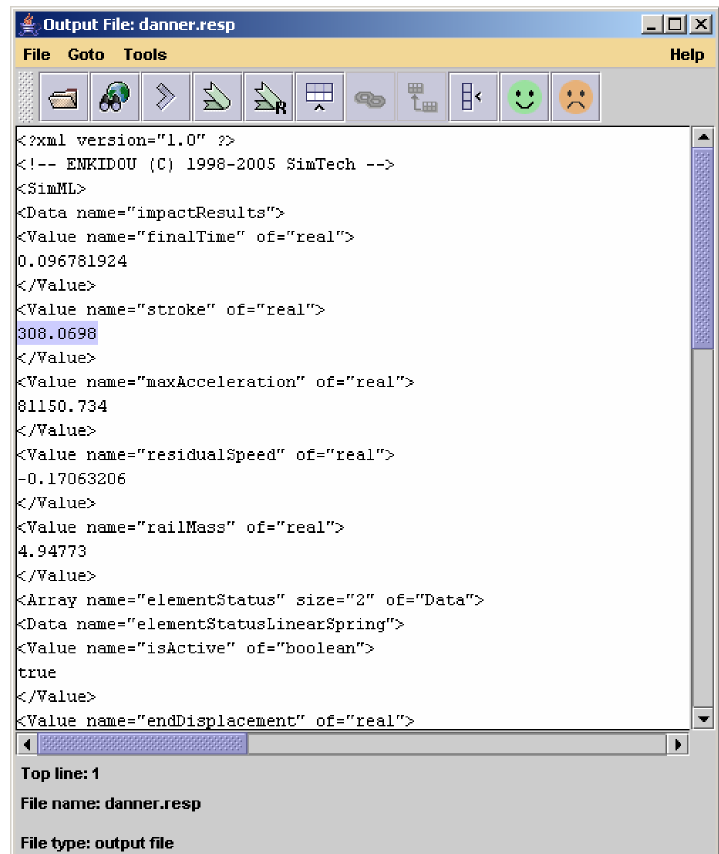


Each element has input and response parameters defined from the xml I/O file of the application …

input parameters for "danner"

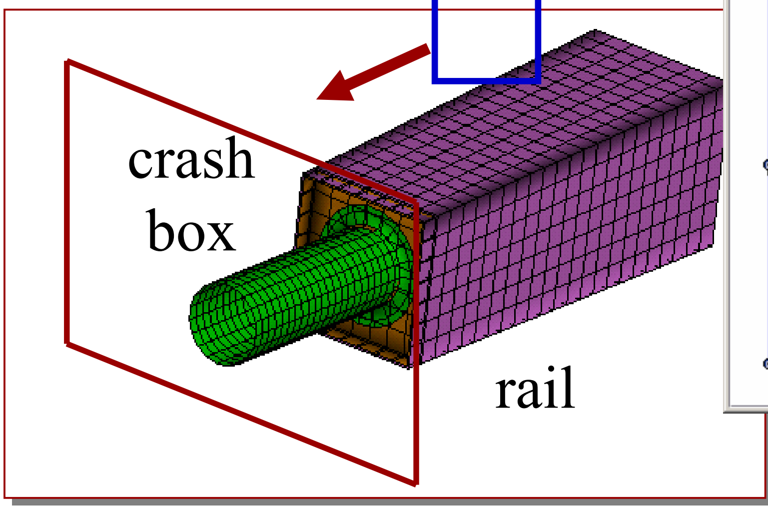


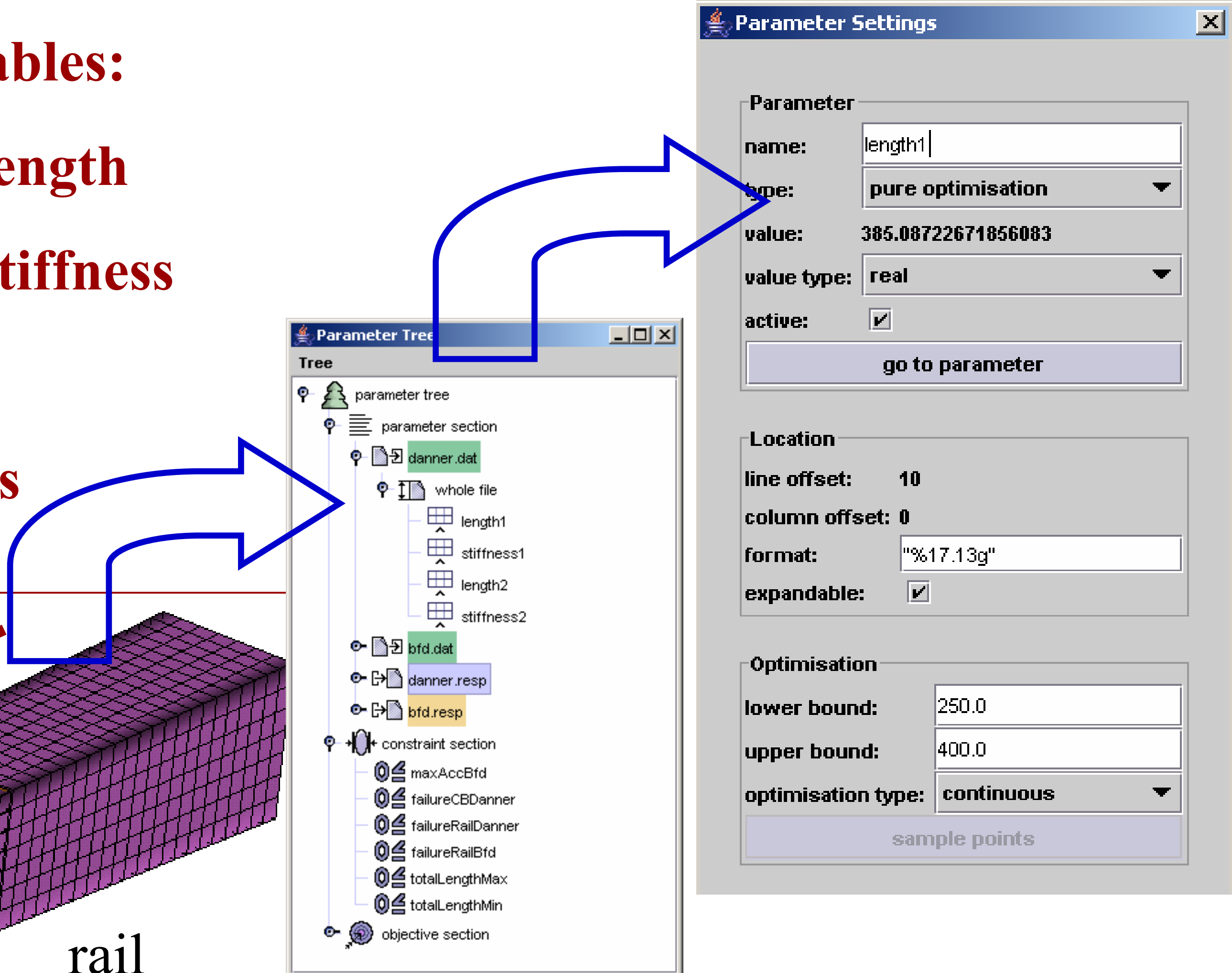
response parameters for "danner"



Design variables:

- crash box length
- crash box stiffness
- rail length
- rail stiffness





Objective function:

- Mass ~ length1*stiffness1 + length2*stiffness2

Constraints:

- Total length (architecture)
- Rail deformation in high speed crash (residual space for engine)
- Rail deformation in low speed crash (no damage)
- Crash box deformation in low speed crash (no crushing)
- Maximum acceleration in high speed crash

Inequality: 0<=

Name	Constraint
maxAccBfd	1.5e6 -maxAccBfd
failureCBDanner	0.8 - dannerStroke/length1
failureRailDanner	0.05 - dannerRailDisp/length2
failureRailBfd	0.55 - bfdRailDisp/length2
totalLengthMax	1100 - length1+length2
totalLengthMin	(length1+length2) - 800

New Remove

OK Cancel

PRELIMINARY CLOUD ANALYSIS

Prior to the actual optimization, we run a (relatively) huge random sampling of the design space. The resulting cloud has been analyzed with our in-house tools.

Total number of shots is 23300.

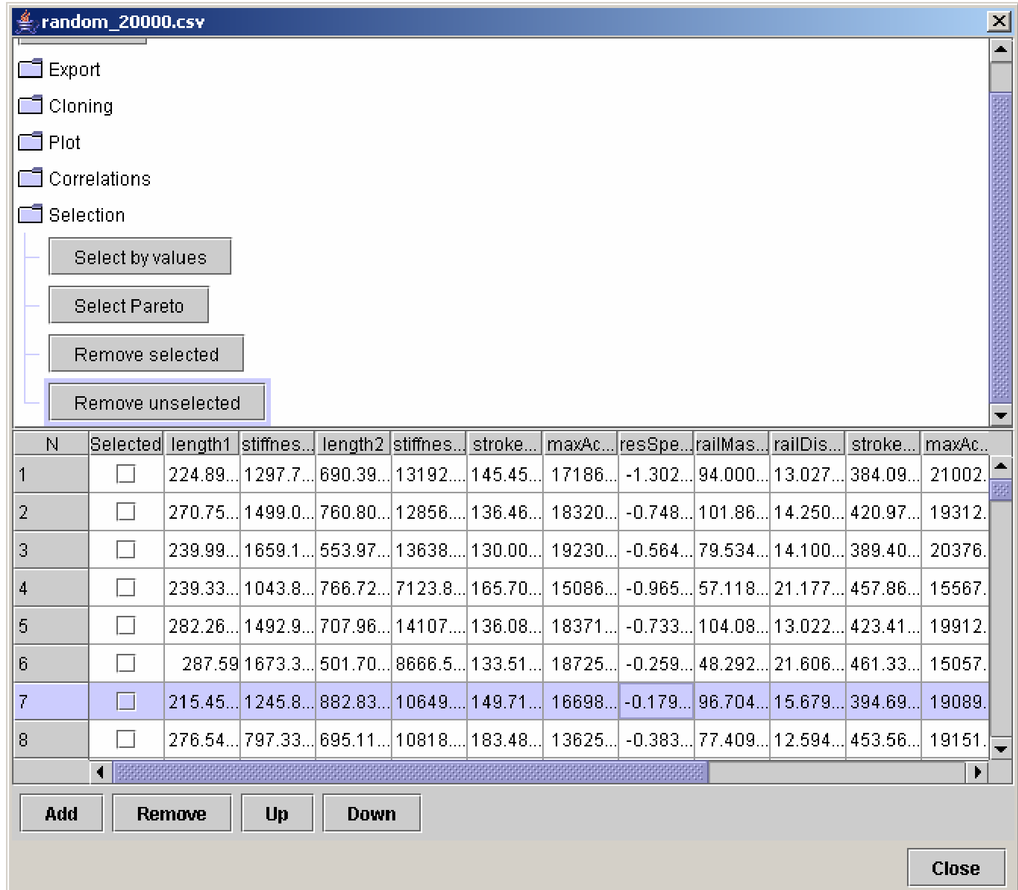
This looks like a lot of points, but in practice it means 1 point every:

12 mm of crash box length

218 N/mm of crash box stiffness

30 mm of rail length

645 N/mm of rail stiffness



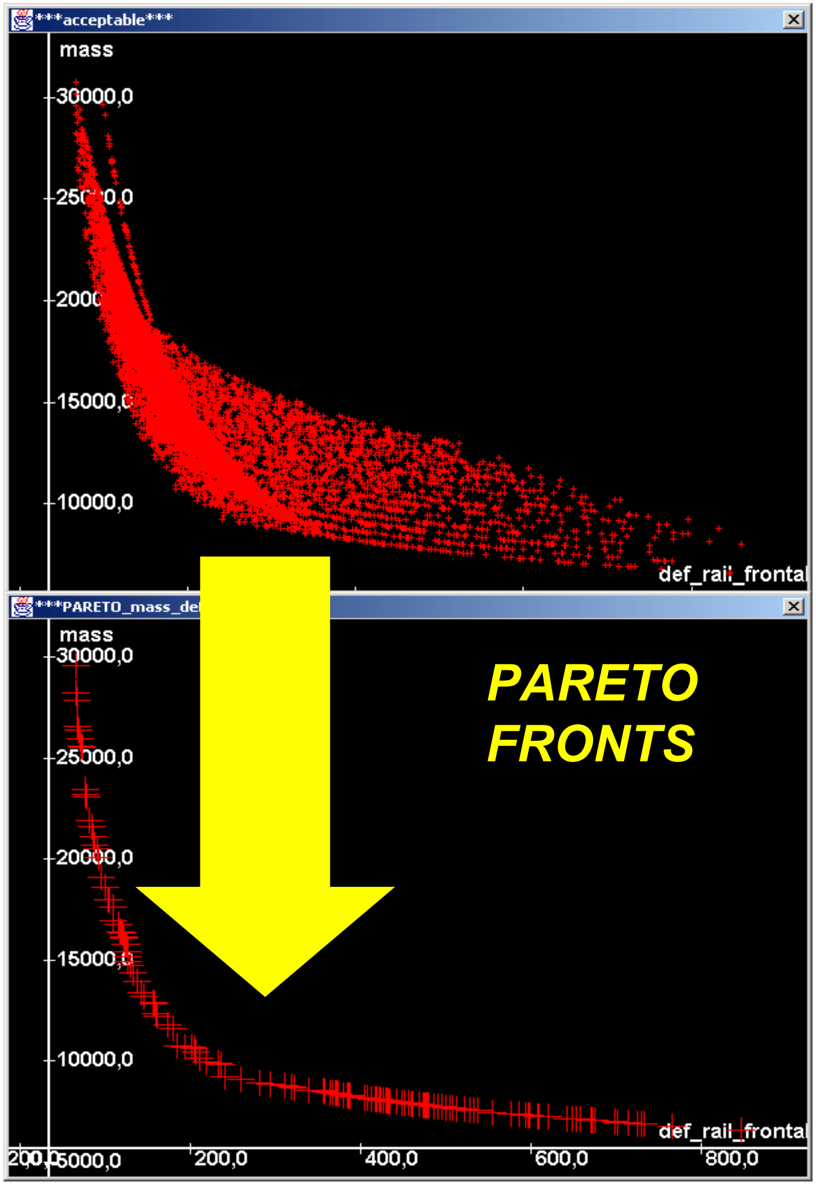
N	Selected	length1	stiffnes...	length2	stiffnes...	stroke...	maxAc...	resSpe...	railMas...	railDis...	stroke...	maxAc..
1	☐	224.89...	1297.7...	690.39...	13192....	145.45...	17186...	-1.302...	94.000...	13.027...	384.09...	21002.
2	☐	270.75...	1499.0...	760.80...	12856....	136.46...	18320...	-0.748...	101.86...	14.250...	420.97...	19312.
3	☐	239.99...	1659.1...	553.97...	13638....	130.00...	19230...	-0.564...	79.534...	14.100...	389.40...	20376.
4	☐	239.33...	1043.8...	766.72...	7123.8...	165.70...	15086...	-0.965...	57.118...	21.177...	457.86...	15567.
5	☐	282.26...	1492.9...	707.96...	14107....	136.08...	18371...	-0.733...	104.08...	13.022...	423.41...	19912.
6	☐	287.59	1673.3...	501.70...	8666.5...	133.51...	18725...	-0.259...	48.292...	21.606...	461.33...	15057.
7	☐	215.45...	1245.8...	882.83...	10649....	149.71...	16698...	-0.179...	96.704...	15.679...	394.69...	19089.
8	☐	276.54...	797.33...	695.11...	10818....	183.48...	13625...	-0.383...	77.409...	12.594...	453.56...	19151.

In order to get a visual information about the admissible domain, we apply the following transformations:

- Selection of admissible shots
- Identification of the Pareto surface of the admissible shots

ADMISSIBLE BOX :
1338 POINTS FROM 23300

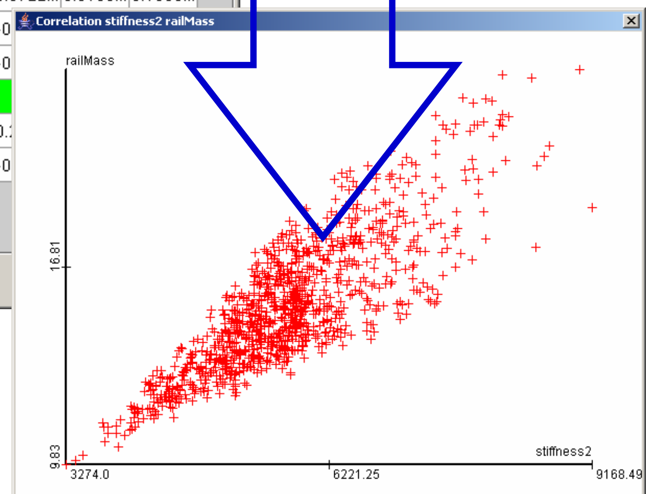
PARETO SURFACE:
1085 POINTS



For the exploration of the Pareto surface, we use the correlation analysis and other basic statistics.

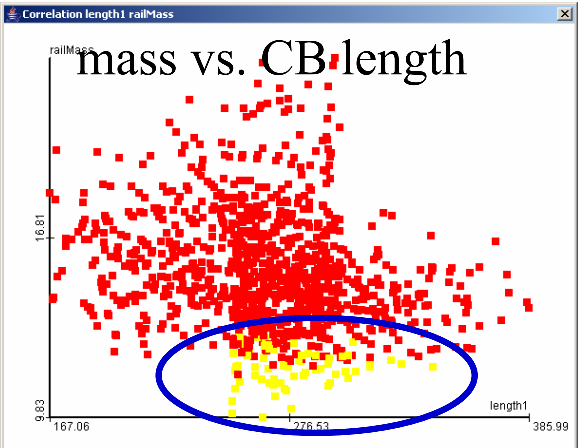
corr

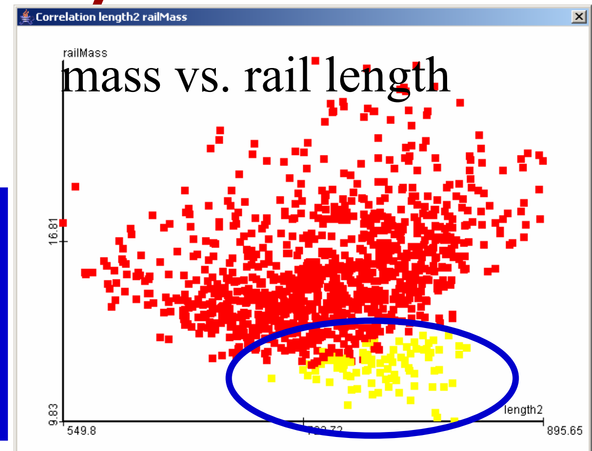
Title	length1	stiffnes...	length2	stiffnes...	stroke...	maxAc...	resSpe...	railMas...	railDis...	stroke...	maxAc...	resSpe...	railMass	failure...	failure...	failure...	totalLe...
length1	1.0	-0.358...	-0.120...	-0.177...	0.3754...	-0.375...	0.1342...	-0.221...	-0.133...	0.8156...	-0.473...	0.0917...	-0.221...	-0.539...	-0.087...	0.5054...	0.4294...
stiffne...	-0.358...	1.0	-0.032...	0.3776...	-0.948...	0.9842...	-0.236...	0.4819...	0.5019...	-0.680...	-0.086...	-0.003...	0.4819...	-0.544...	0.5320...	-0.539...	-0.223...
length2	-0.120...	-0.032...	1.0	-0.342...	0.0813...	-0.082...	0.0289...	0.2068...	0.2762...	0.0803...	-0.215...	0.0594...	0.2068...	0.1790...	-0.277...	-0.678...	0.8446...
stiffne...	-0.177...	0.3776...	-0.342...	1.0	-0.428...	0.4681...	-0.120...	0.8354...	-0.585...	-0.640...	0.7482...	-0.113...	0.8354...	-0.242...	-0.384...	-0.328...	-0.406...
stroke...	0.3754...	-0.948...	0.0813...	-0.428...	1.0	-0.984...	0.2313...	-0.499...	-0.457...	0.7018...	0.0093...	0.0078...	-0.499...	0.5649...	-0.516...	0.5141...	0.2765...
maxAc...	-0.375...	0.9842...	-0.082...	0.4681...	-0.984...	1.0	-0.239...	0.5406...	0.4273...	-0.724...	0.0088...	-0.015...	0.5406...	-0.558...	0.4875...	-0.538...	-0.277...
resSp...	0.1342...	-0.236...	0.0289...	-0.120...	0.2313...	-0.239...	1.0	-0.126...	-0.092...	0.2021...	-0.043...	-0.018...	-0.126...	0.0955...	-0.108...	0.1342...	0.0988...
railMa...	-0.221...	0.4819...	0.2068...	0.8354...	-0.499...	0.5406...	-0.126...	1.0	-0.355...	-0.633...	0.5650...	-0.072...	1.0	-0.267...	-0.456...	-0.752...	0.0686...
railDis...	-0.133...	0.5019...	0.2762...	-0.585...	-0.457...	0.4273...	-0.092...	-0.355...	1.0	0.0201...	-0.768...	0.1063...	-0.355...	-0.284...	0.8434...	-0.141...	0.1790...
stroke...	0.8156...	-0.680...	0.0803...	-0.640...	0.7018...	-0.724...	0.2021...	-0.633...	0.0201...	1.0	-0.583...	0.1017...	-0.633...	-0.083...	-0.047...	0.6272...	0.5129...
maxAc...	-0.473...	-0.086...	-0.215...	0.7482...	0.0093...	0.0088...	-0.043...	0.5650...	-0.768...	-0.583...	1.0	-0.159...	0.5650...	0.4078...	-0.633...	-0.231...	-0.451...
resSp...	0.0917...	-0.003...	0.0594...	-0.113...	0.0078...	-0.015...	-0.018...	-0.072...	0.1063...	0.1017...	-0.159...	1.0	-0.072...	-0.065...	0.0722...	0.0185...	0.1035...
railMa...	-0.221...	0.4819...	0.2068...	0.8354...	-0.499...	0.5406...	-0.126...	1.0	-0.355...	-0.633...	0.5650...	-0.072...	1.0	-0.267...	-0		
failure...	-0.539...	-0.544...	0.1790...	-0.242...	0.5649...	-0.558...	0.0955...	-0.267...	-0.284...	-0.083...	0.4078...	-0.065...	-0.267...	1.0	-0		
failure...	-0.087...	0.5320...	-0.277...	-0.384...	-0.516...	0.4875...	-0.108...	-0.456...	0.8434...	-0.047...	-0.633...	0.0722...	-0.456...	-0.377...			
failure...	0.5054...	-0.539...	-0.678...	-0.328...	0.5141...	-0.538...	0.1342...	-0.752...	-0.141...	0.6272...	-0.231...	0.0185...	-0.752...	0.0184...	0.		
totalLe...	0.4294...	-0.223...	0.8446...	-0.406...	0.2765...	-0.277...	0.0988...	0.0686...	0.1790...	0.5129...	-0.451...	0.1035...	0.0686...	-0.128...	-0		



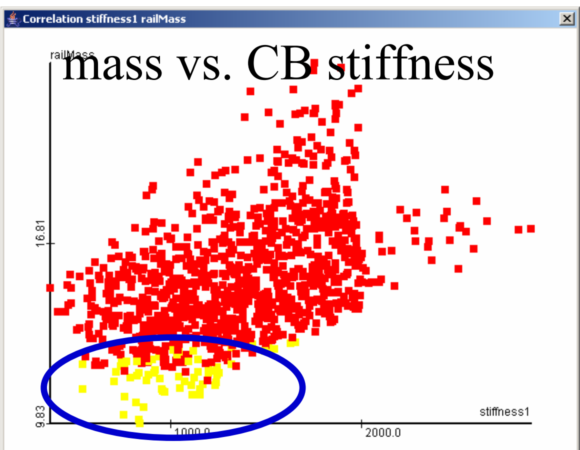
From the analysis of our Pareto surface, we can obtain some prior knowledge about our optimization problem.

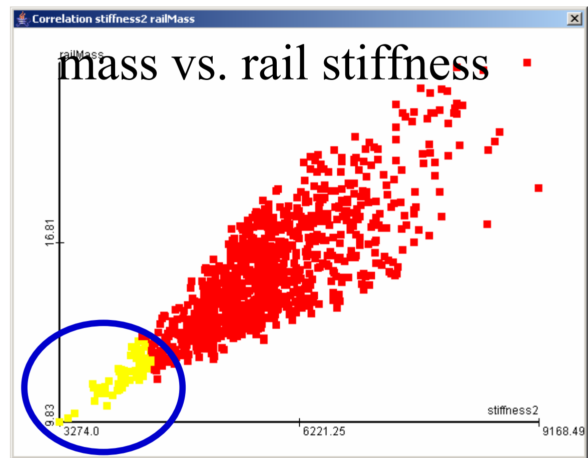
First, we can be reasonably sure that there is a global minimum and where it is approximately.



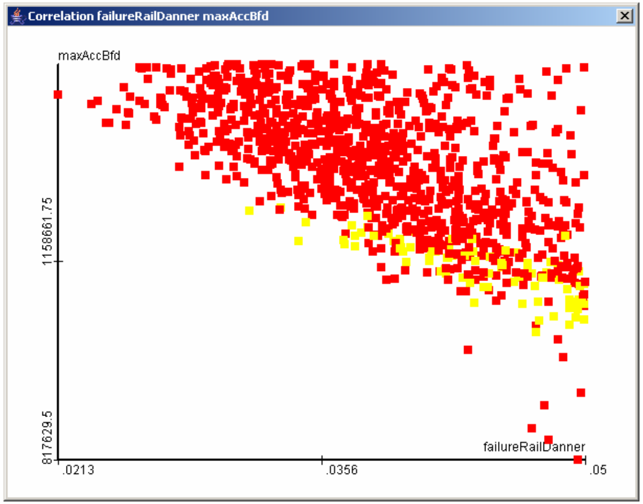


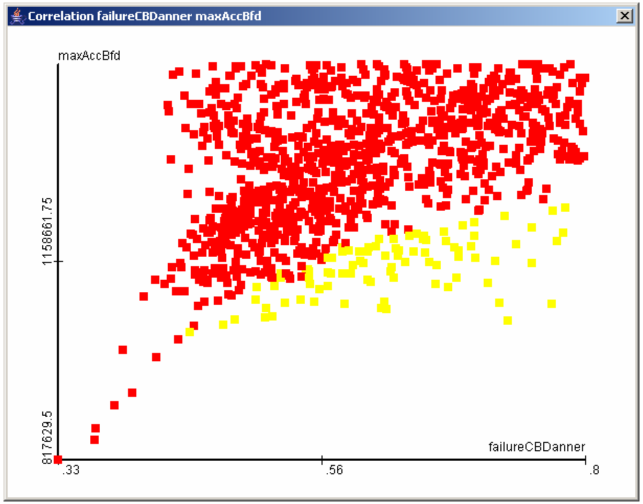
area around global minimum

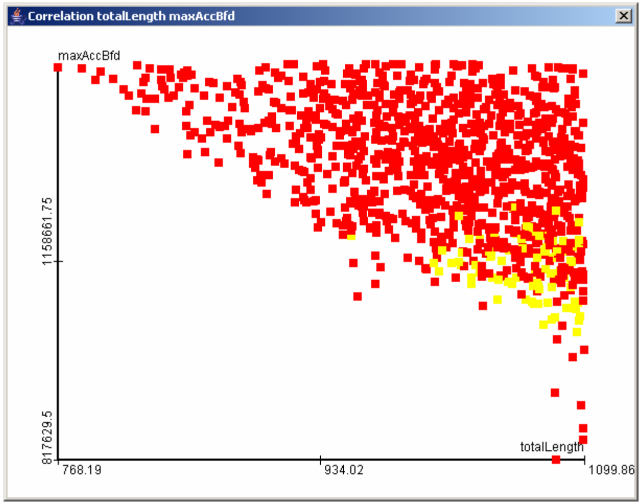


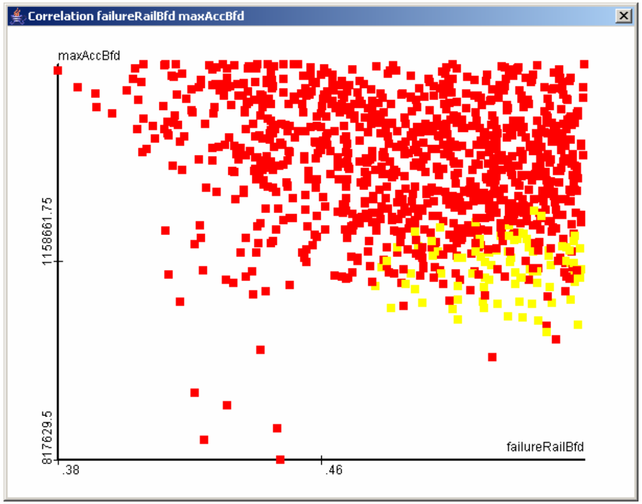


We can also foresee which will be the active constraints. In our case, all but the maximum acceleration in the high speed test.



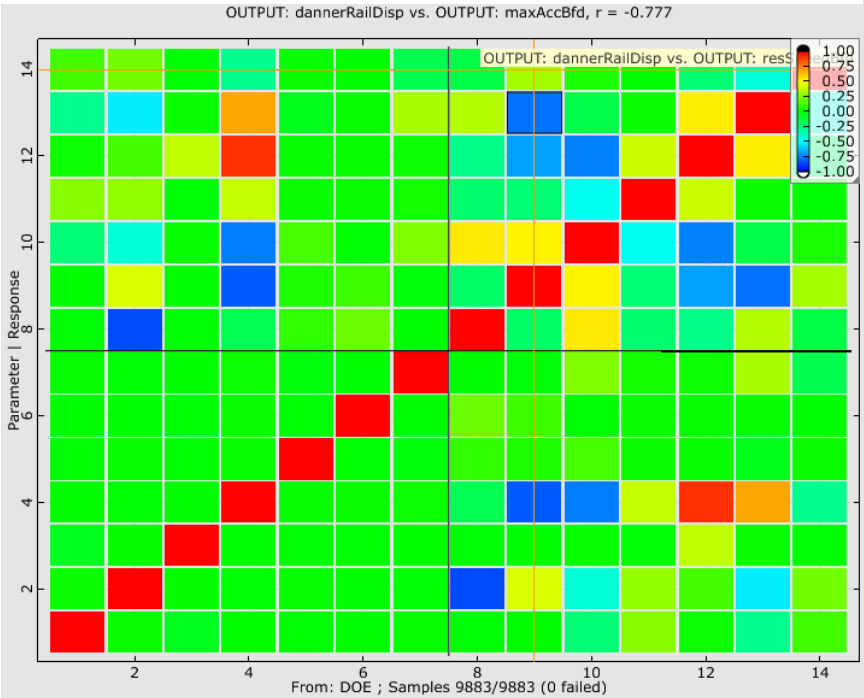


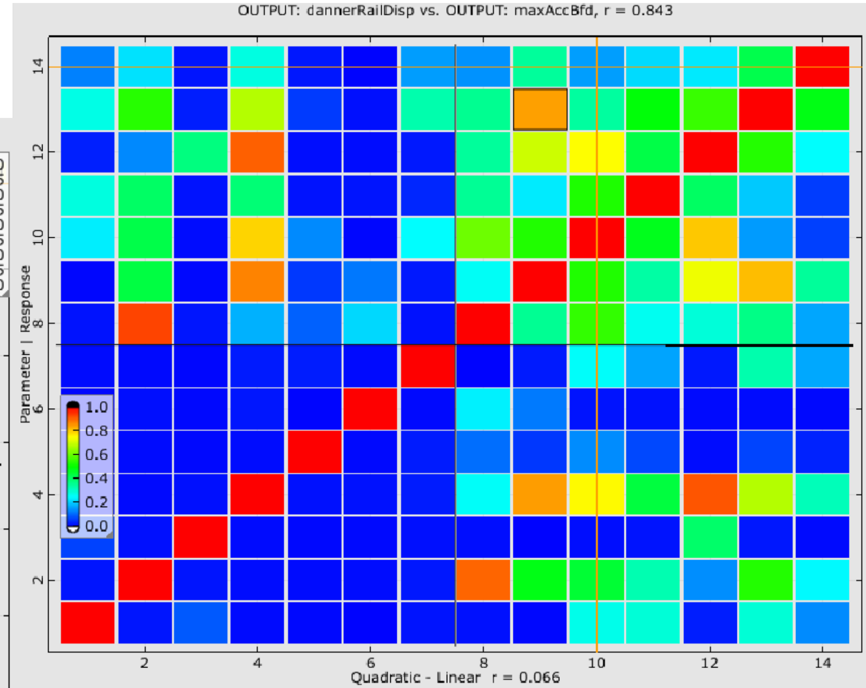




Statistical analysis using *optiSlang*

linear correlations

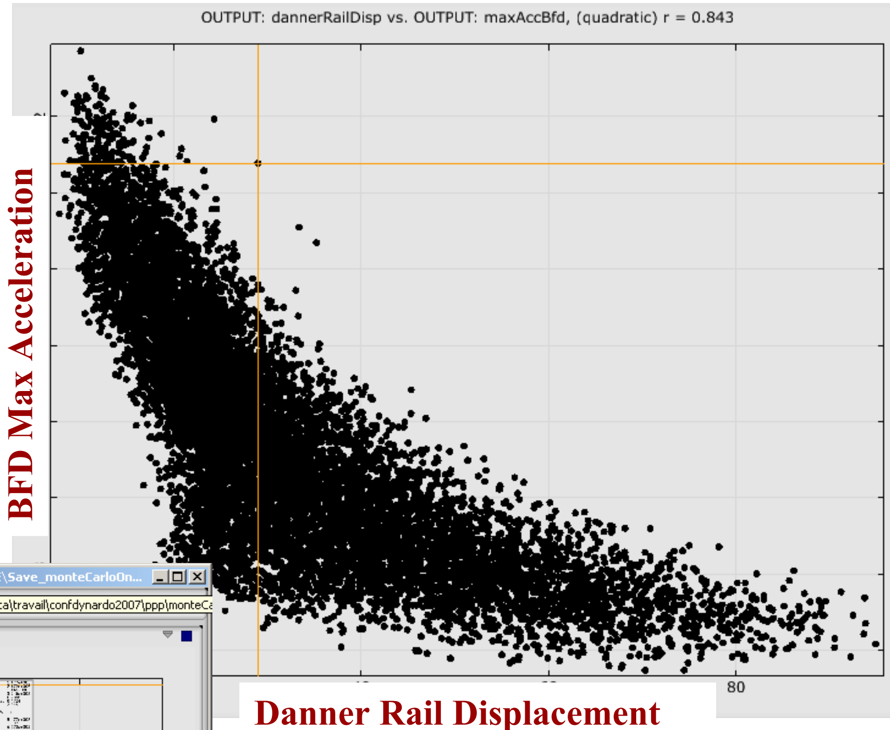


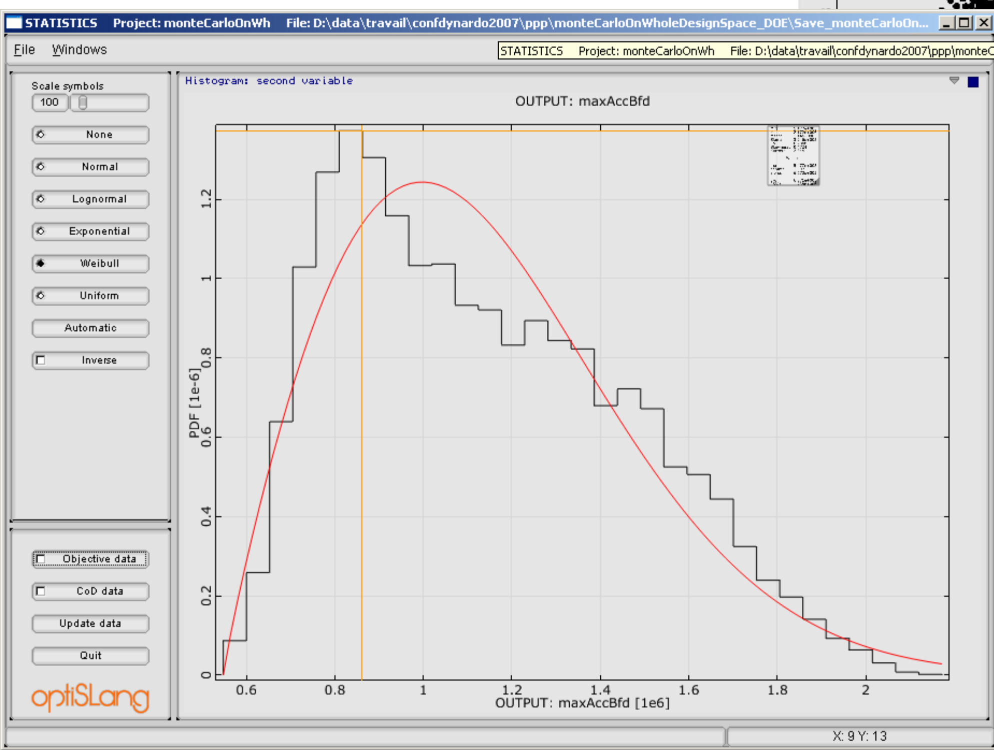


quadratic correlations

ANT-HILL plots

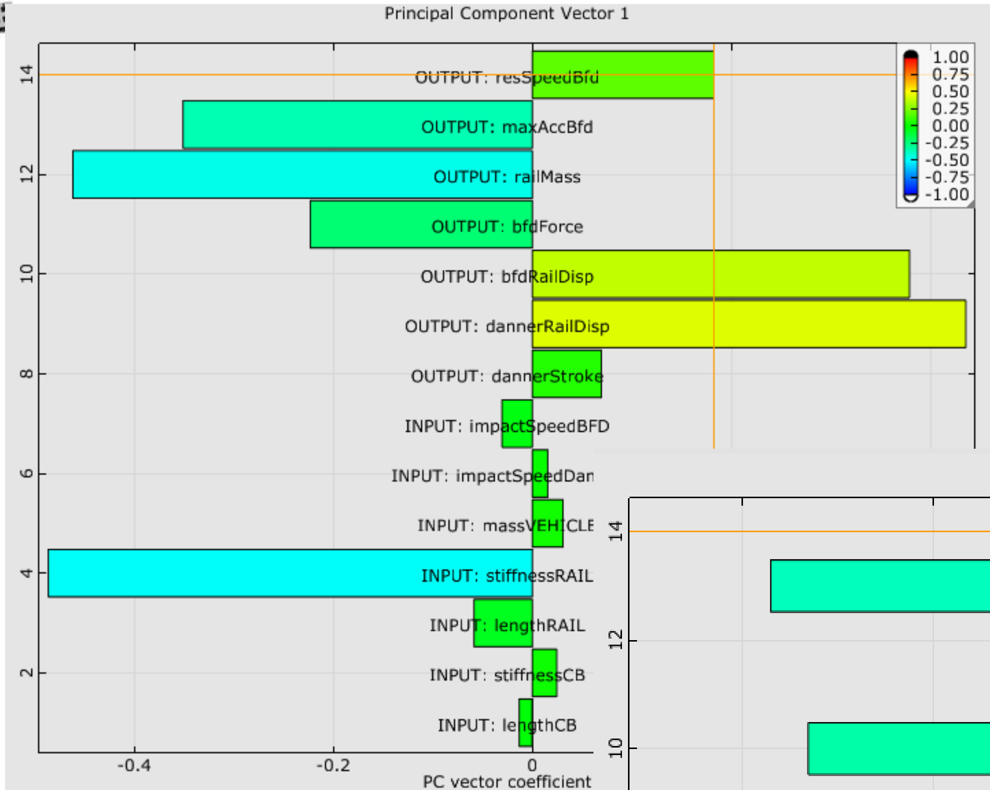
(scatter plots)

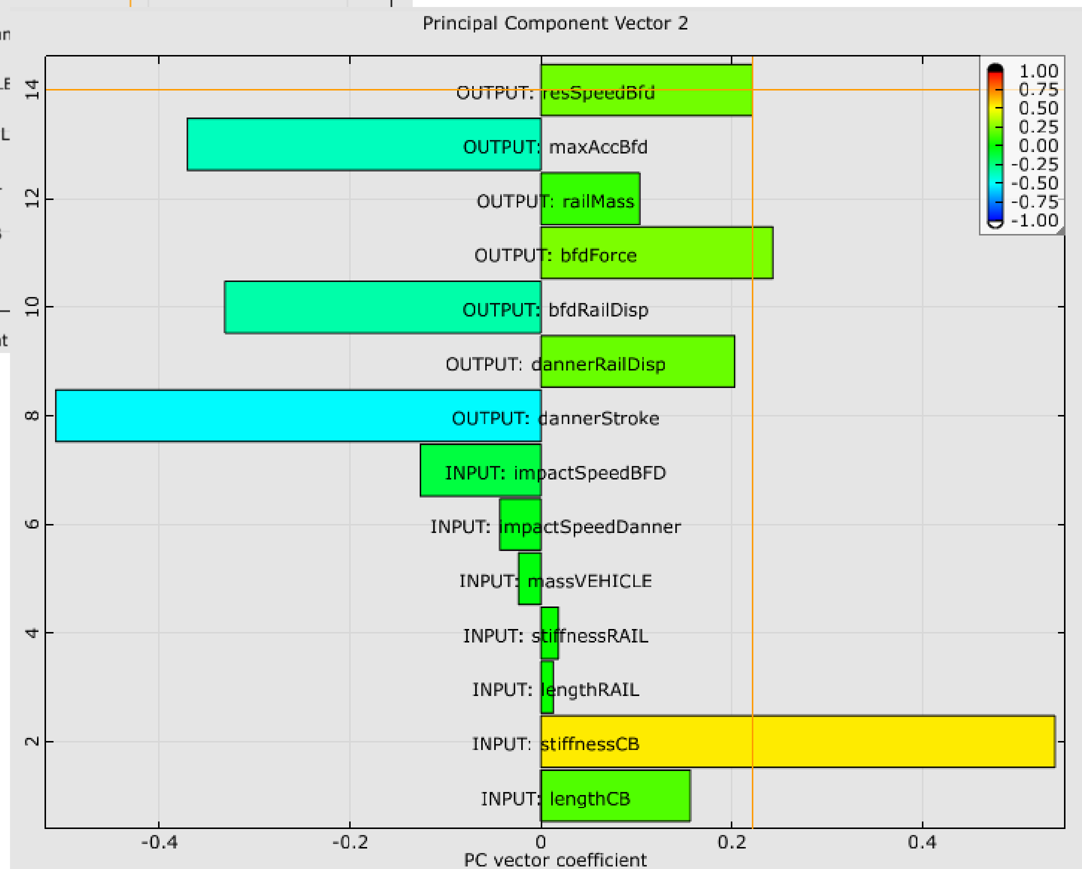


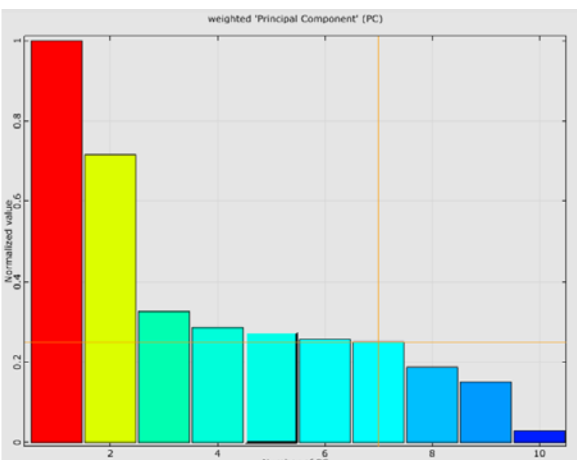


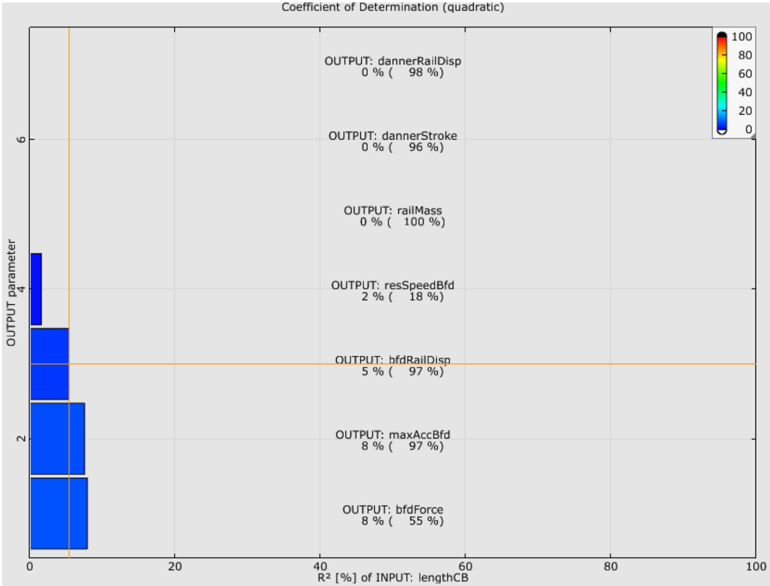
distribution functions

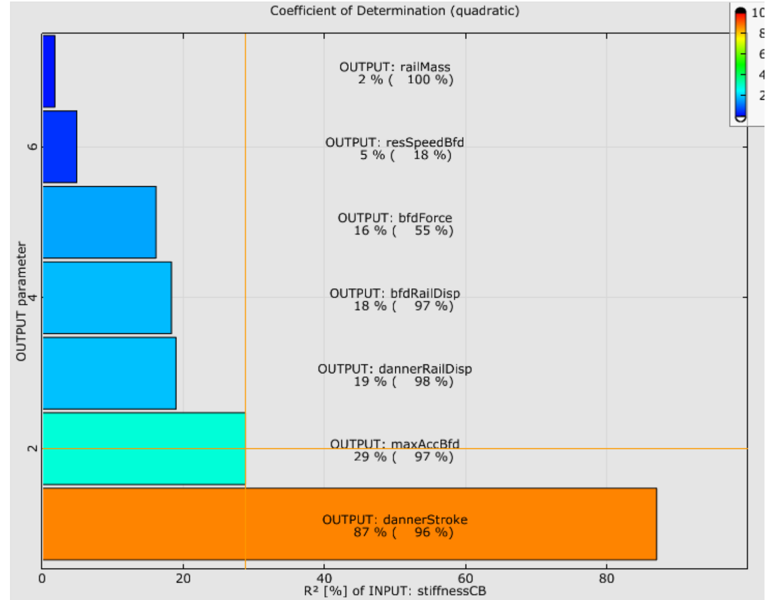
principal component analysis

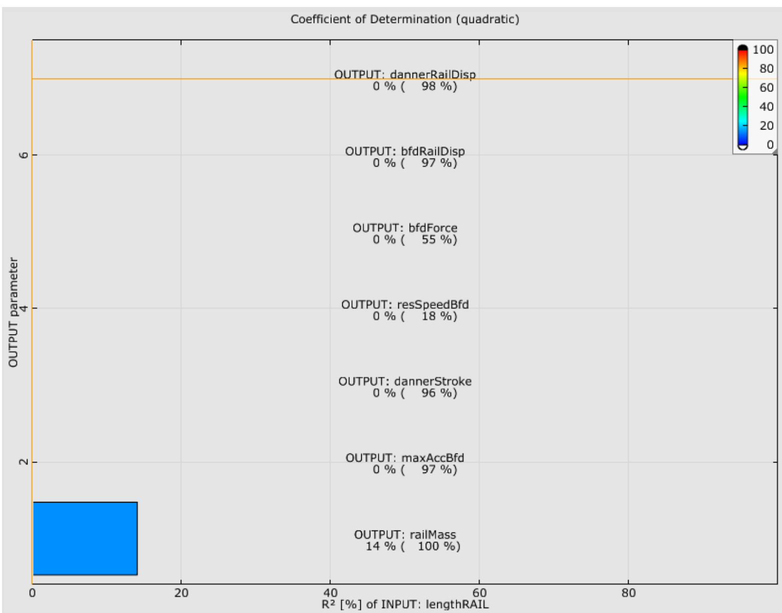


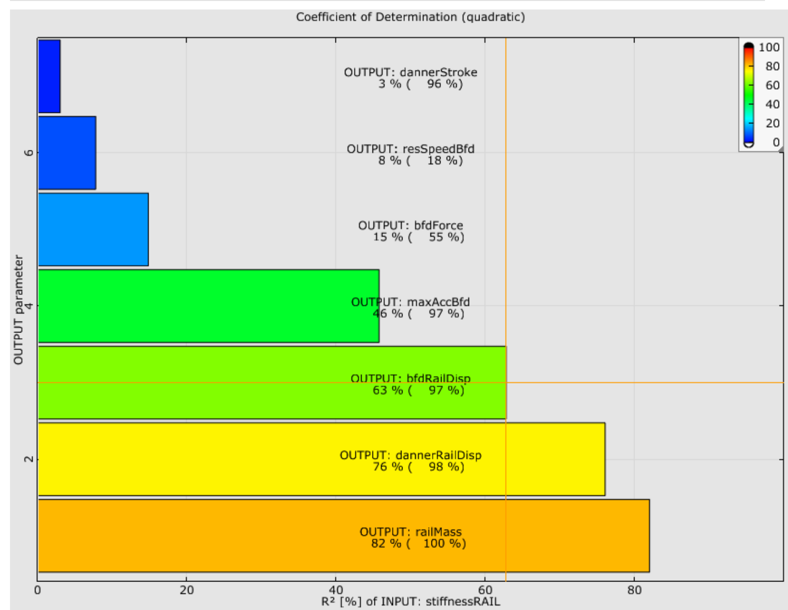





Coefficient of Determination (quadratic)
OUTPUT: dannerRailDisp
0 % (98 %)
OUTPUT: dannerStroke
0 % (96 %)
OUTPUT: railMass
0 % (100 %)
OUTPUT: resSpeedBfd
2 % (18 %)
OUTPUT: bfdRailDisp
5 % (97 %)
OUTPUT: maxAccBfd
8 % (97 %)
OUTPUT: bfdForce
8 % (55 %)
OUTPUT parameter
R² [%] of INPUT: lengthCB


Coefficient of Determination (quadratic)
OUTPUT: railMass
2 % (100 %)
OUTPUT: resSpeedBfd
5 % (18 %)
OUTPUT: bfdForce
16 % (55 %)
OUTPUT: bfdRailDisp
18 % (97 %)
OUTPUT: dannerRailDisp
19 % (98 %)
OUTPUT: maxAccBfd
29 % (97 %)
OUTPUT: dannerStroke
87 % (96 %)
OUTPUT parameter
R² [%] of INPUT: stiffnessCB


Coefficient of Determination (quadratic)
OUTPUT: dannerRailDisp
0 % (98 %)
OUTPUT: bfdRailDisp
0 % (97 %)
OUTPUT: bfdForce
0 % (55 %)
OUTPUT: resSpeedBfd
0 % (18 %)
OUTPUT: dannerStroke
0 % (96 %)
OUTPUT: maxAccBfd
0 % (97 %)
OUTPUT: railMass
14 % (100 %)
OUTPUT parameter
R² [%] of INPUT: lengthRAIL


Coefficient of Determination (quadratic)
OUTPUT: dannerStroke
3 % (96 %)
OUTPUT: resSpeedBfd
8 % (18 %)
OUTPUT: bfdForce
15 % (55 %)
OUTPUT: maxAccBfd
46 % (97 %)
OUTPUT: bfdRailDisp
63 % (97 %)
OUTPUT: dannerRailDisp
76 % (98 %)
OUTPUT: railMass
82 % (100 %)
OUTPUT parameter
R² [%] of INPUT: stiffnessRAIL

OPTIMIZATION USING DIFFERENT ALGOS OF optiSlang

Gradient based

Response surface

Adaptative response surface

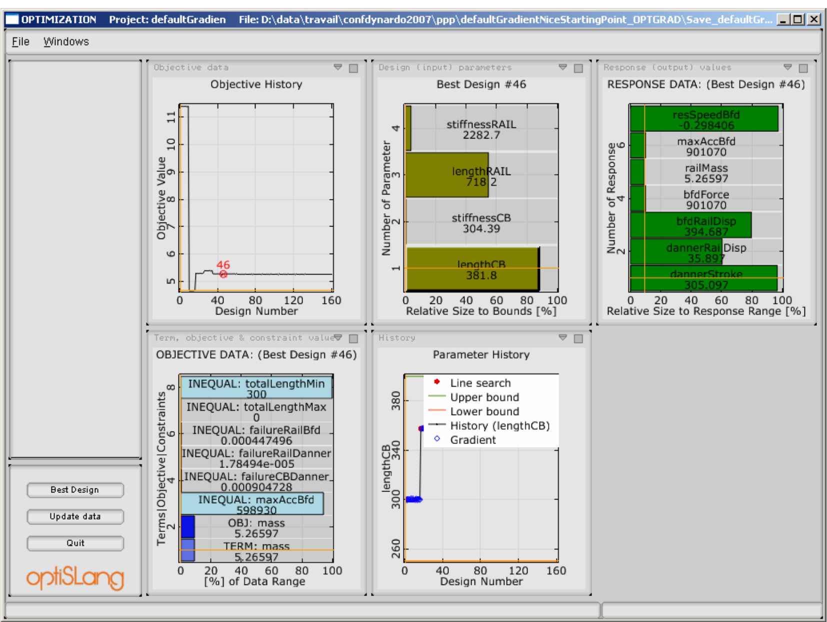
Evolutionary optimization

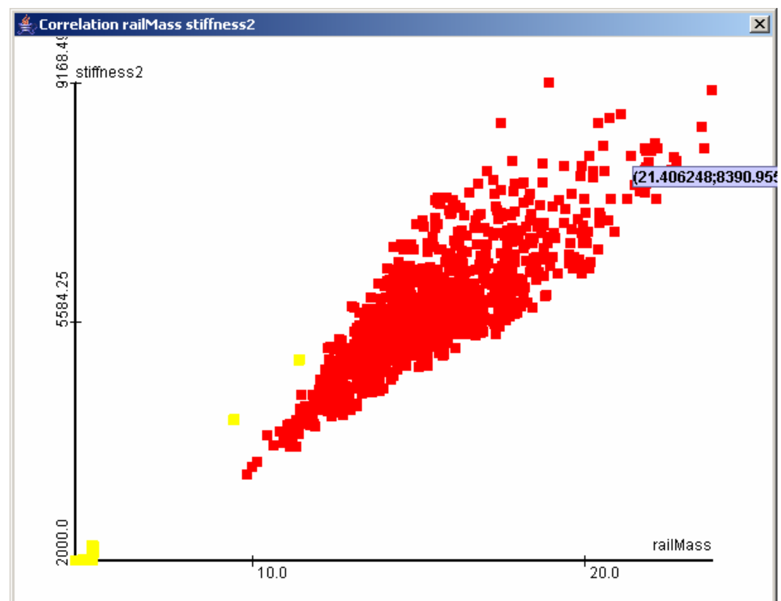
Optimization using gradient method (NLPQLP)

For the first optimization, we start from a “nice” point, where constraint violation is not important.

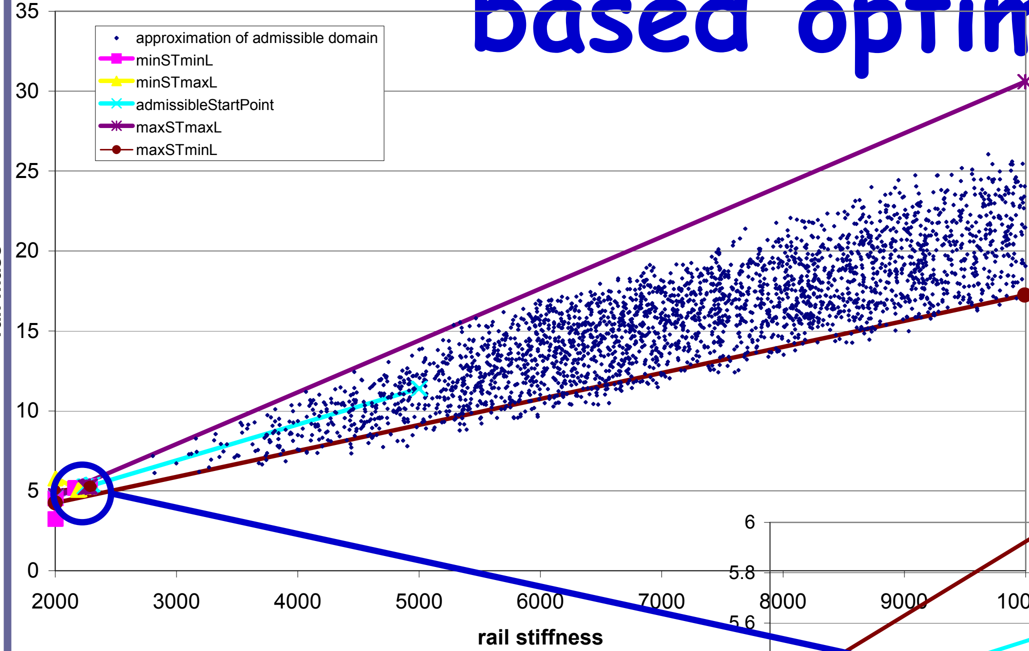
Results are excellent for a total of 160 design points.

	initial value	final value
CB LENGTH	300	381
CB STIFFNESS	1000	304
RAIL LENGTH	700	718
RAIL STIFFNESS	5000	2283
MASS	11.4	5.27



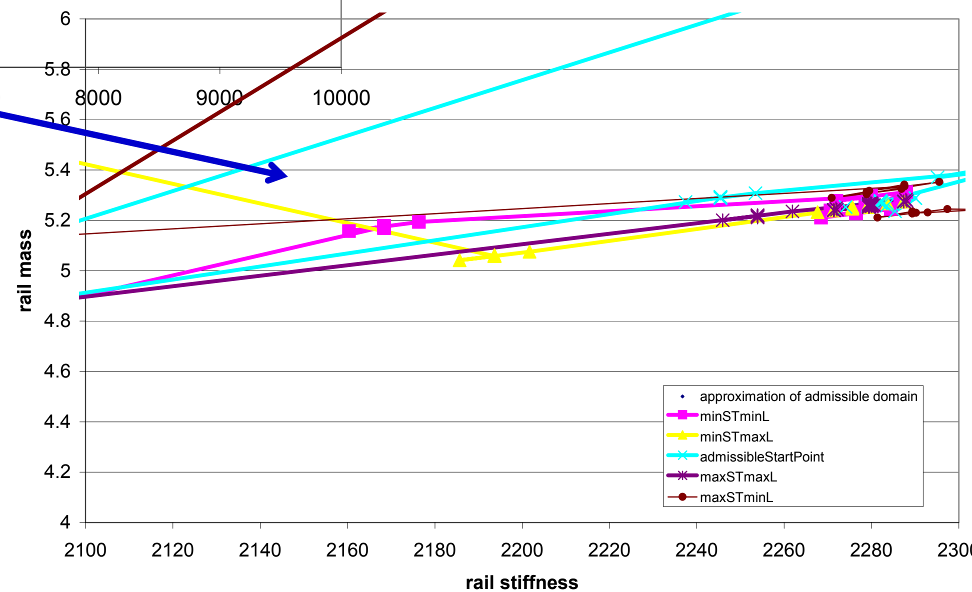


Robustness of the gradient based optimizer



We have run the optimization with four more different starting point, at the corners of the design space

NLPQLP converges very rapidly towards the same optimal value

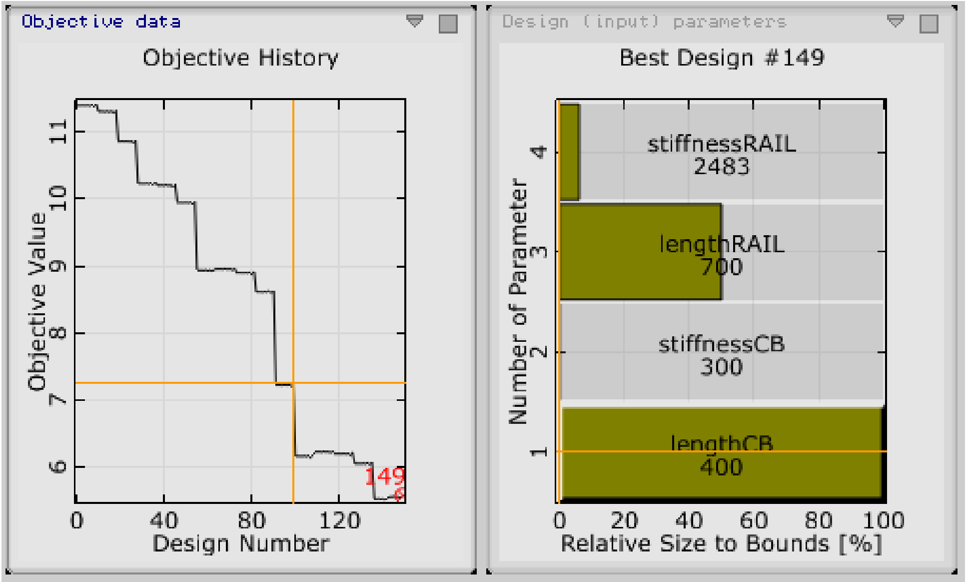


RSM optimization

Starting point: 9-point DOE

Convergence is achieved in 150 points

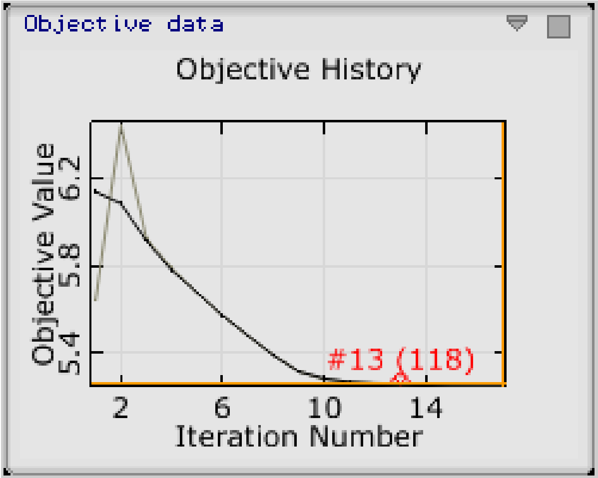
CB LENGTH	400
CB STIFFNESS	300
RAIL LENGTH	700
RAIL STIFFNESS	2483
MASS	5.57

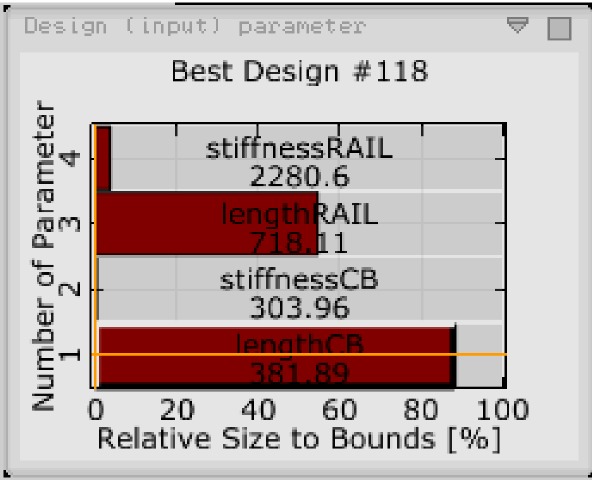


ARSM optimization

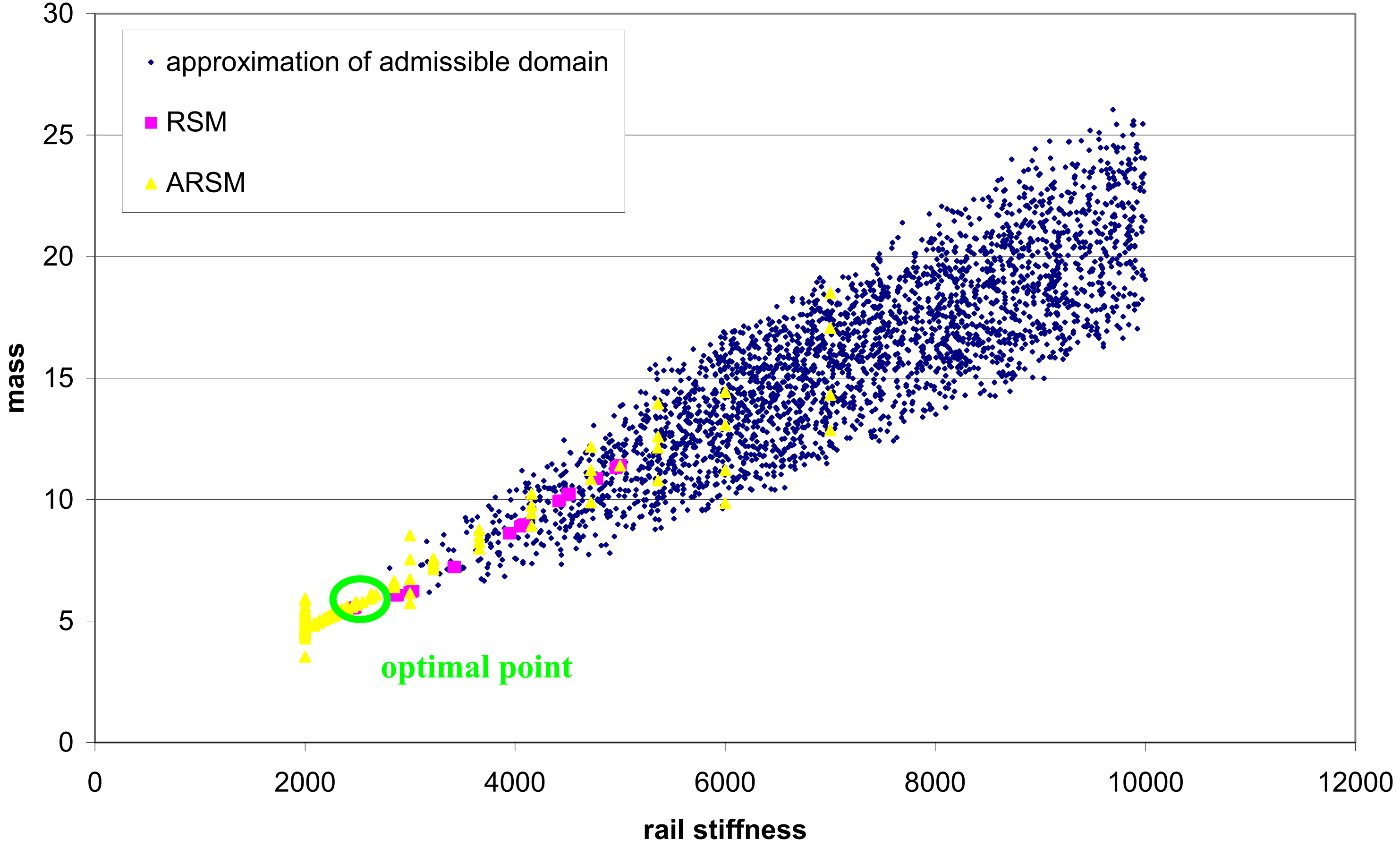
Convergence is achieved in 155 points

CB LENGTH	382
CB STIFFNESS	304
RAIL LENGTH	718
RAIL STIFFNESS	2281
MASS	5.26

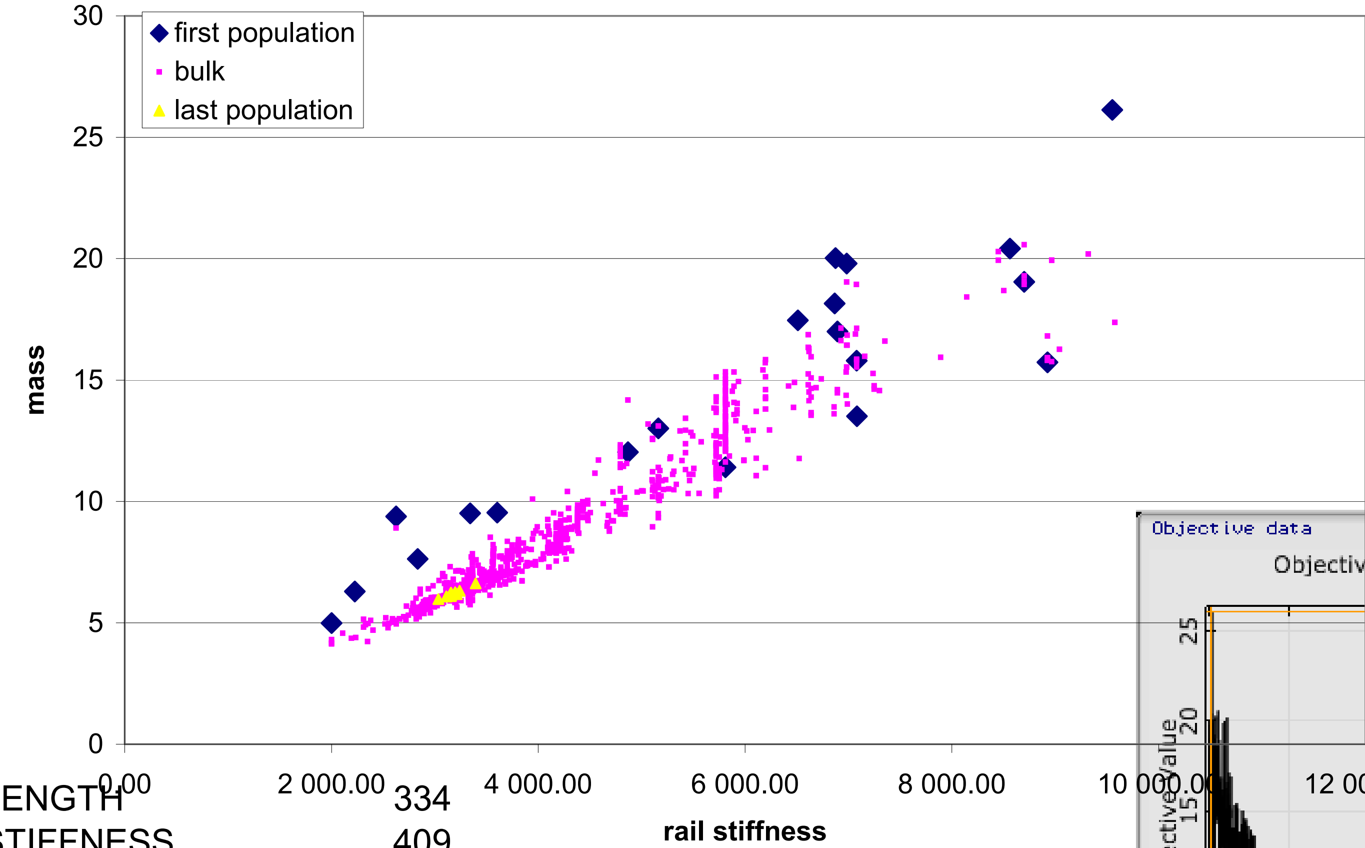


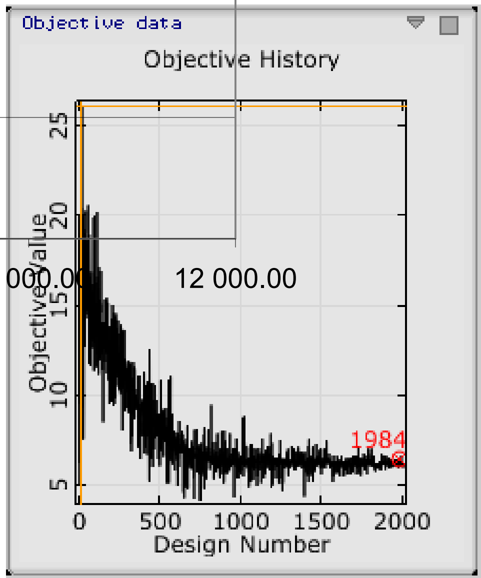


Convergence of RS based methods



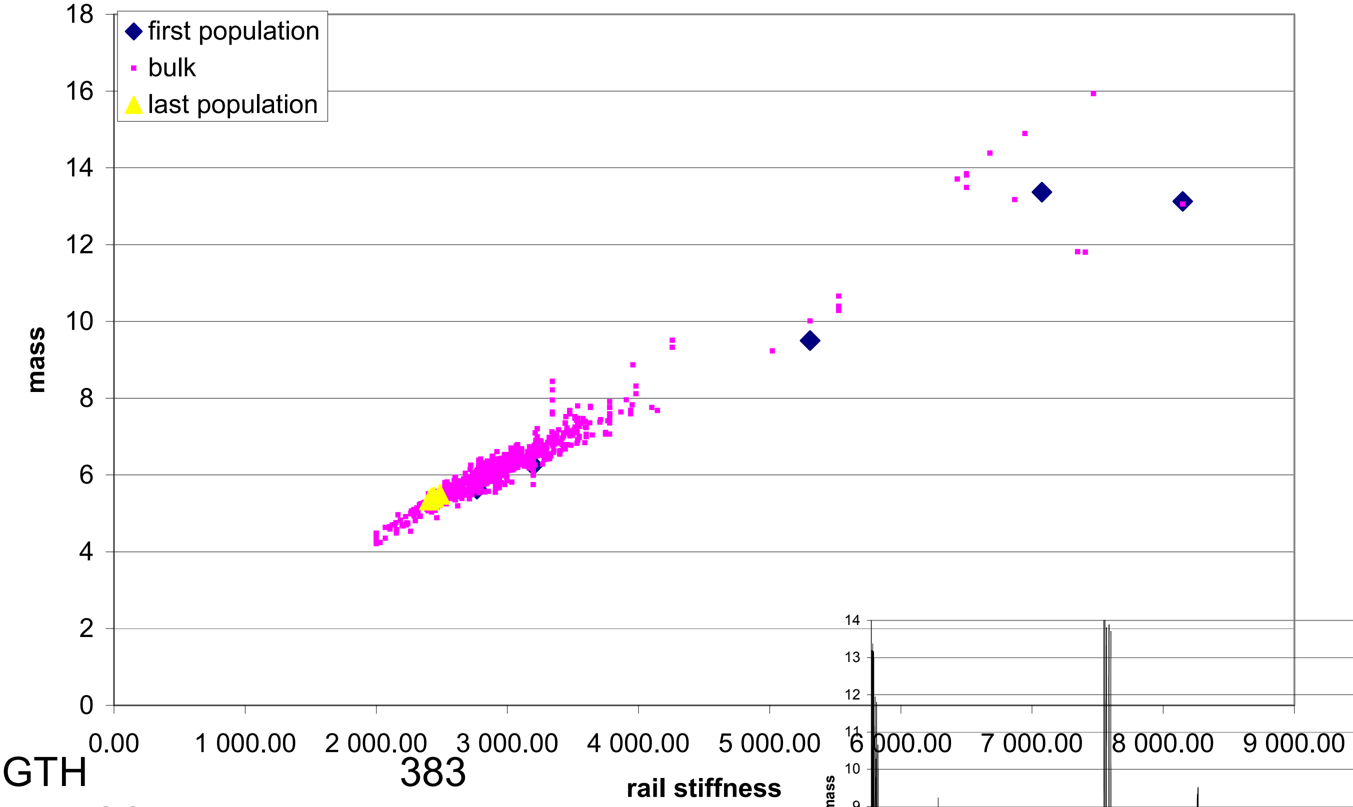
Evolutionary algorithm
1. Global search



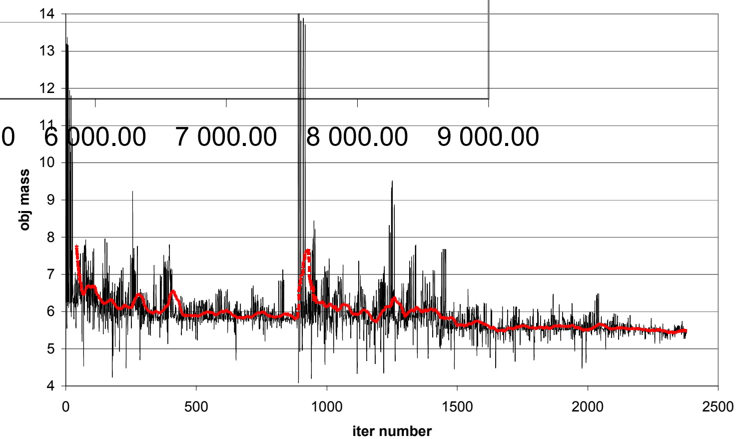


CB LENGTH	334
CB STIFFNESS	409
RAIL LENGTH	609
RAIL STIFFNESS	3308
MASS	6.45
N° ITER	2000

Evolutionary algorithm
2. Design Improvement



CB LENGTH	383
CB STIFFNESS	300
RAIL LENGTH	695
RAIL STIFFNESS	2438
MASS	5.42
N° ITER	2380



ROUBSTNESS ANALYSIS with optiSlang

Plain MonteCarlo

Latin Hypercube Sampling

Robustness problem overview

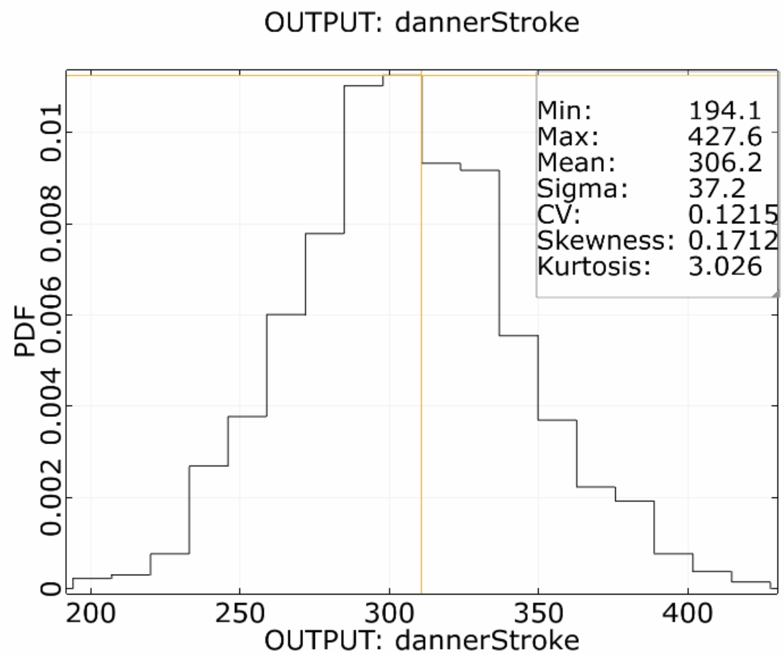
2Spring_robustness1.pro

Robust | Output | Constraints | Objectives

Name	Distribut...	Mean	CoV	Stddev	Lower ...	Upper Cut	Active
lengthCB	normal	381.0	1.0E-4	0.0381	-	-	✔
stiffnes...	normal	304.0	0.1	30.400...	-	-	✔
lengthR...	normal	718.0	1.0E-4	0.0718	-	-	✔
stiffnes...	normal	2283.0	0.1	228.3	-	-	✔
massV...	normal	1.0	0.1	0.1	-	-	✔
impactS...	normal	5000.0	0.1	500.0	-	-	✔
impactS...	normal	20000.0	0.1	2000.0	-	-	✔

Cancel OK

Robustness analysis results

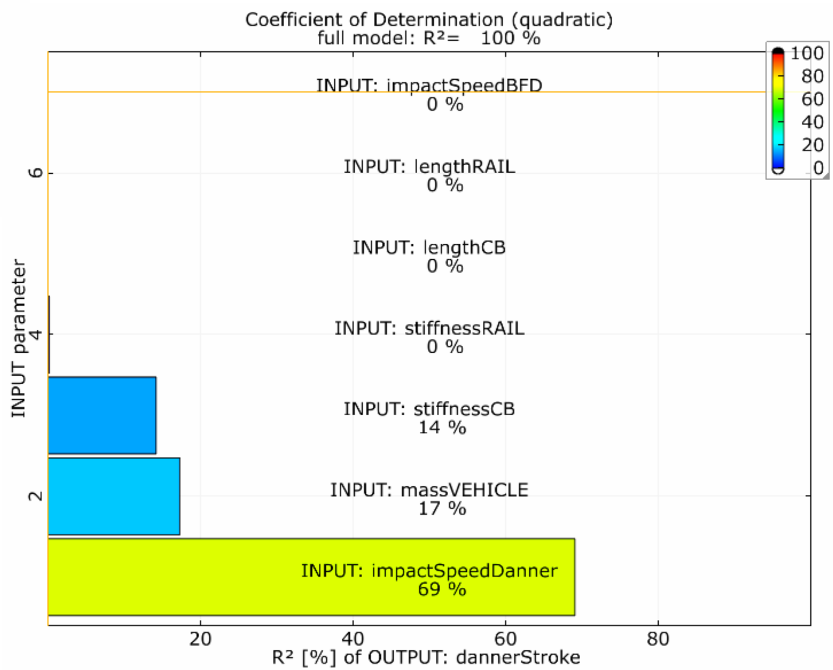


Response variability:

Variance, pdf, etc …

Variance analysis:

Which input factor contributes to the variability of which output ?



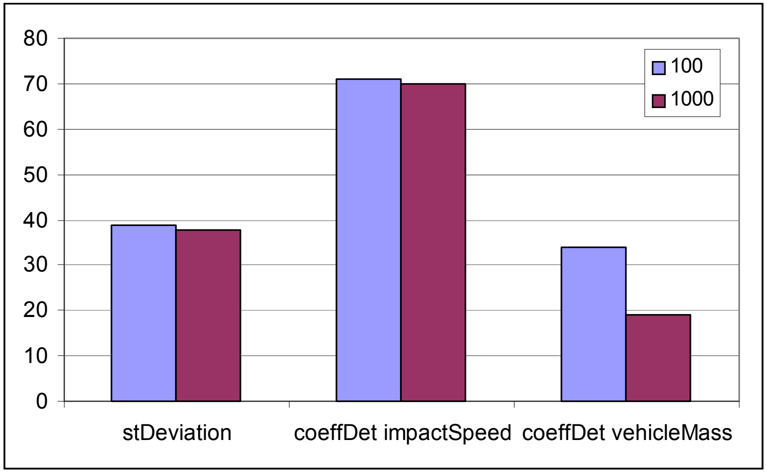
Robustness analysis: PMC vs. LHS

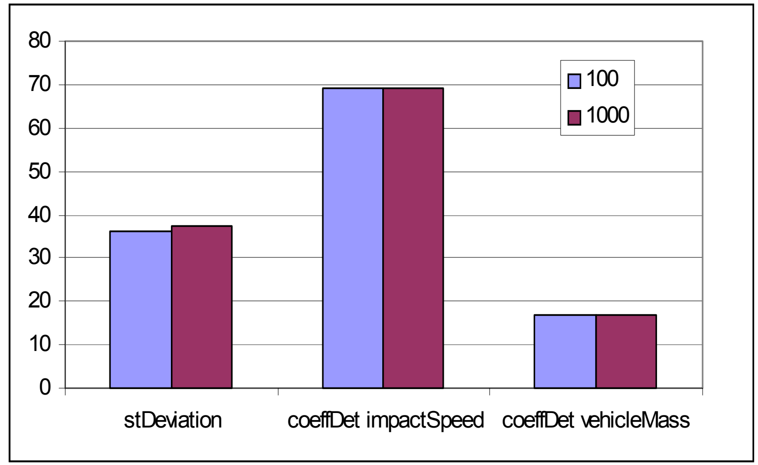
LHS converges faster than PMC …

PMC

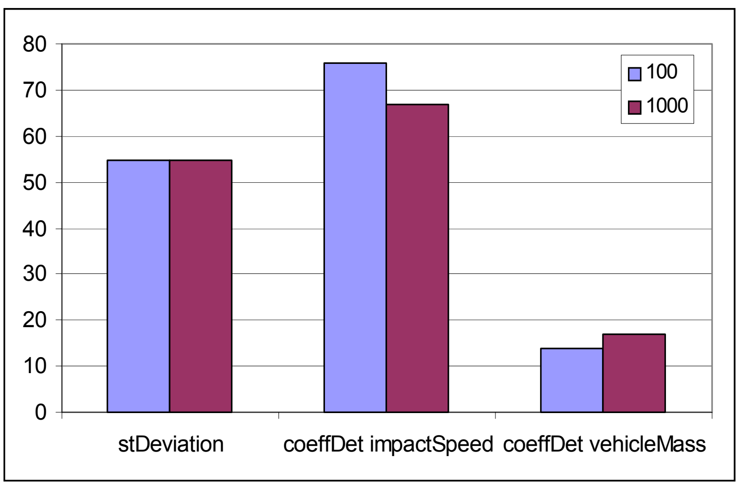
LHS

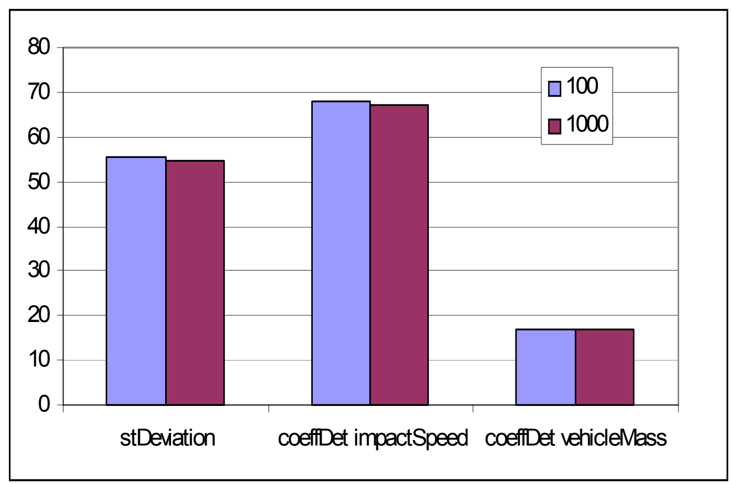
Danner stroke





BFD rail displ.

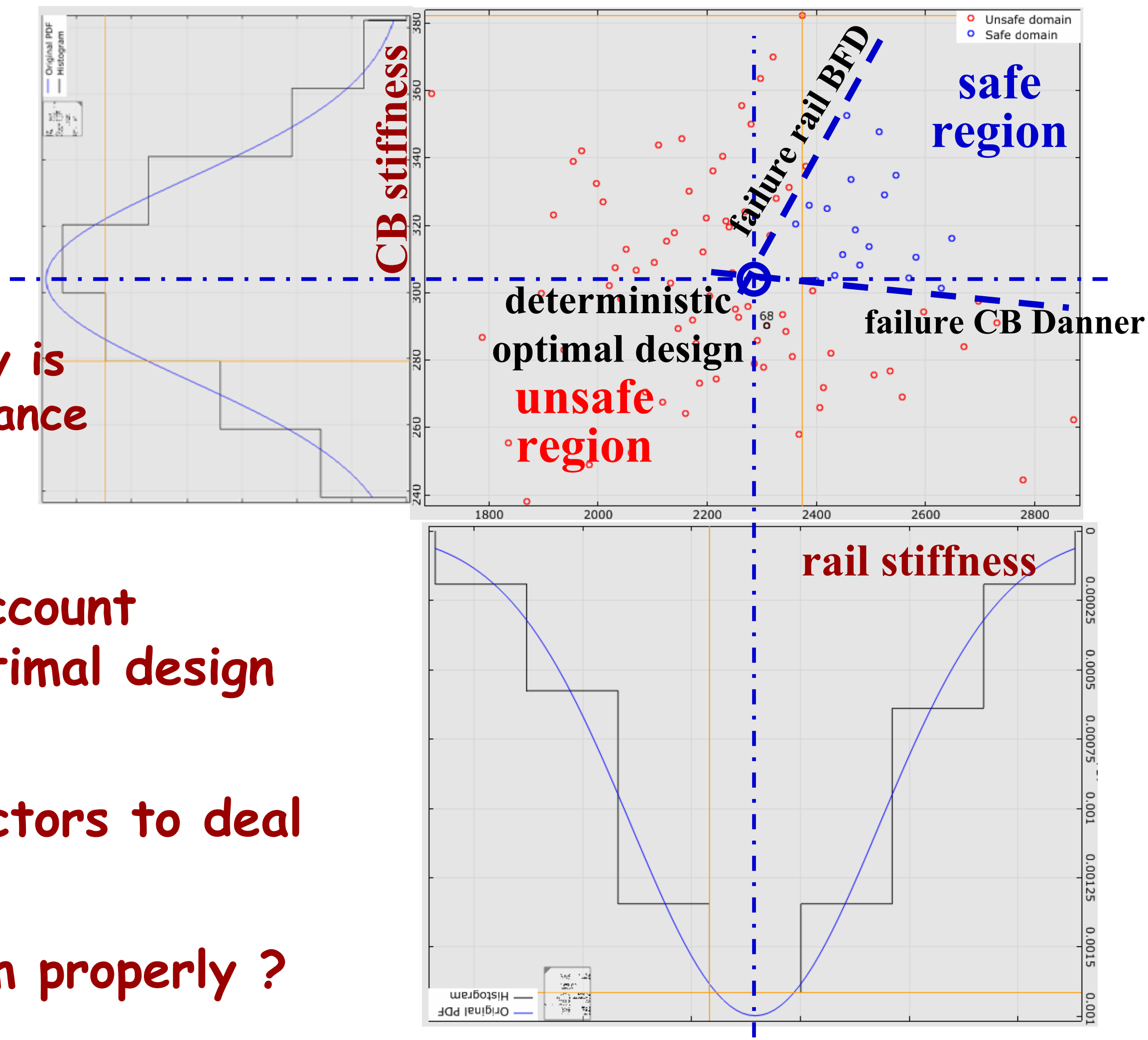




ROBUST OPTIMIZATION motivation

stiffness CoV	failure probability
10%	80%
5%	80%
...	...

N.B.: failure probability is independent of the variance of the design variables



When we take into account uncertainties, the optimal design point is not robust.

We all use safety factors to deal with this problem.

How can we find them properly ?

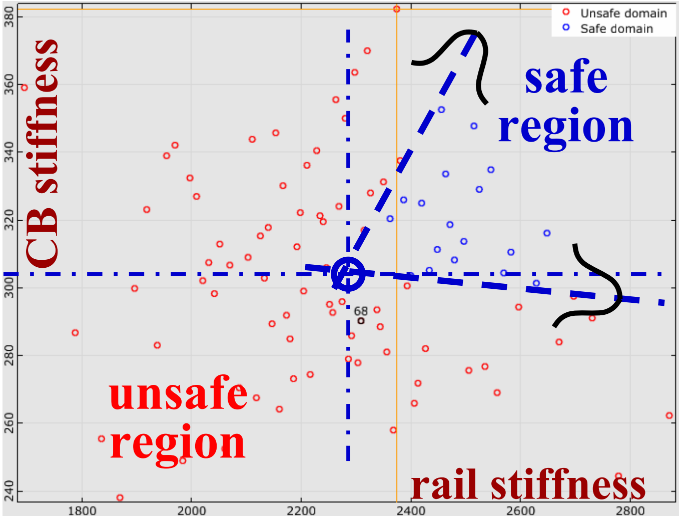
ROBUST OPTIMIZATION
definition of safety factors

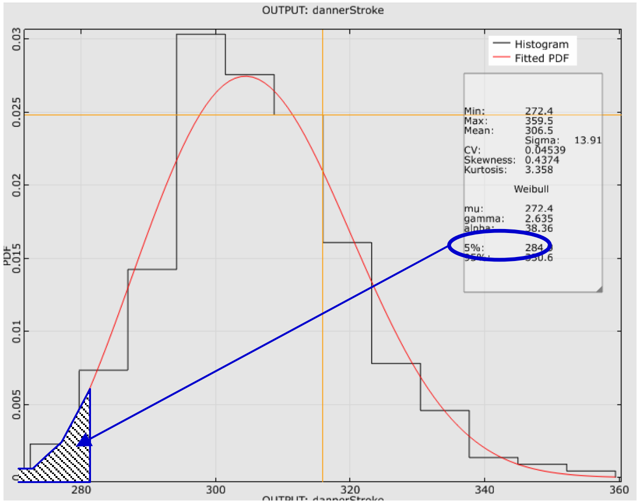
1. Find active constraints
2. From reliability analysis, find the values of the responses such that $PoF_{resp} = PoF_{target}$
3. Change constraint value and run a new (deterministic) optimization
4. Repeat if necessary

… in our case:

Active constraints:

- Danner CB failure
- CFB and Danner rail failure





ROBUST OPTIMIZATION definition of safety factors

Under the hypothesis that active constraints are independent, the safety factors can be found by elementary probability:

No failure == $(q_1 > 0)$ && $(q_2 > 0)$ … && $(q_n > 0)$

or

$P(\text{no failure}) = P(q_1 > 0)*P(q_2 > 0) \ldots *P(q_n > 0)$

One (engineering) solution is thus that the new constraint value is such that

$P(q_i > 0) = [P(\text{no failure})]^{1/n}$

ROBUST OPTIMIZATION
robust solution

Data scatter:

CoV on element stiffness = 10%

Global PoF_{target} = 3%

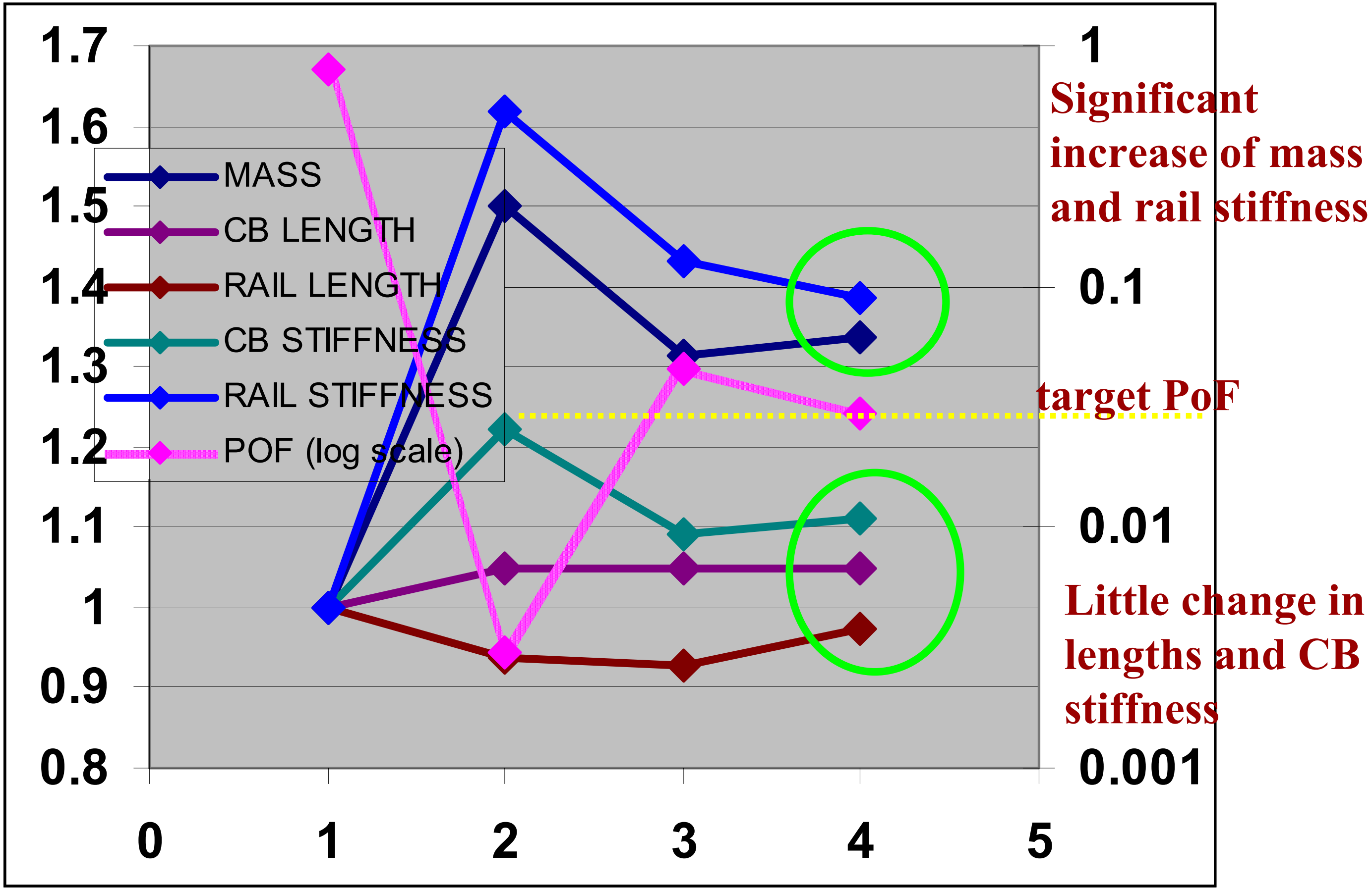
There are 3 active constraints:

- Danner max stroke,
- Danner rail displacement,
- BFD rail displacement

For each of the constraint, PoF_{target} = 1%

ROBUST OPTIMIZATION
robust solution

The procedure converges in 4 iterations



CONCLUSIONS

- The analysis of simplified models is interesting for the formulation of optimization problems
- optiSlang optimization performs well on non-linear, dynamic problems. ARSM is particularly fast and accurate. Evolutionary algorithms are predictably slow in convergence.
- Approximate robust optimization is possible with using the present features of optiSlang (with a little more pdf analysis …)