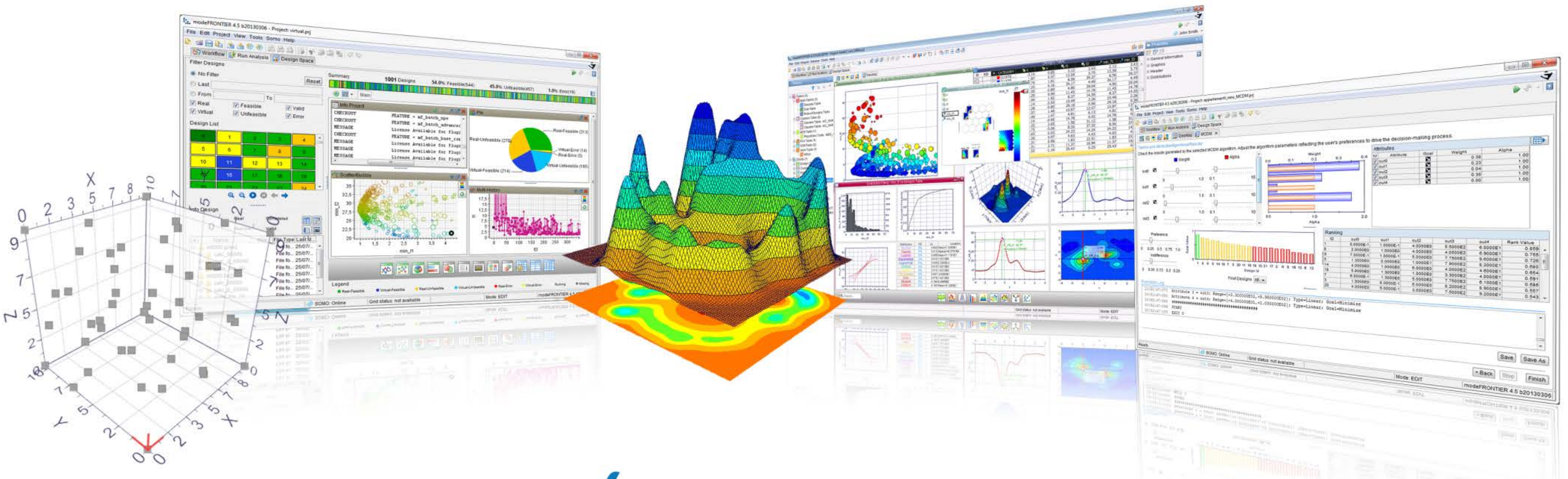
modeFRONTIER

modeFRONTIER Basic Training

Agenda

1. Introduction to modeFRONTIER
2. Process Automation with the Workflow Editor
3. Introduction to DOE and optimization strategies and algorithms
4. Running the optimization
5. Exploring basic post-processing tools
6. Introduction to Response Surfaces

What is modeFRONTIER?



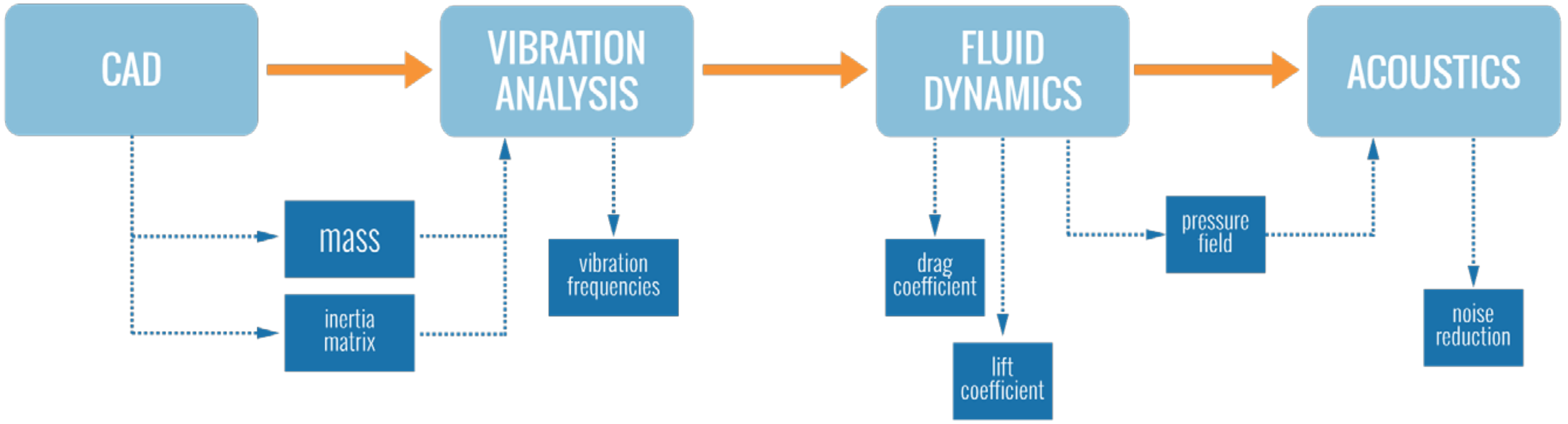
mode FRONTIER

is an integration platform for **multi-objective and multi-disciplinary optimization**. It provides a seamless coupling with third party engineering tools, enables the **automation** of the design simulation process, and facilitates **analytic decision making**.

Multi-disciplinary optimization with modeFRONTIER

modeFRONTIER integrates with **any parametric software** (CAD, CAE, FEM, generic, etc.) **automating** the entire optimization process in which data is transferred from one simulation to the next and the relevant values of outputs and objectives are extracted.

This **multi-disciplinary approach** allows the user to exploit the **interaction** between the disciplines and determine the **global optimum solution**, instead of optimizing each discipline sequentially.



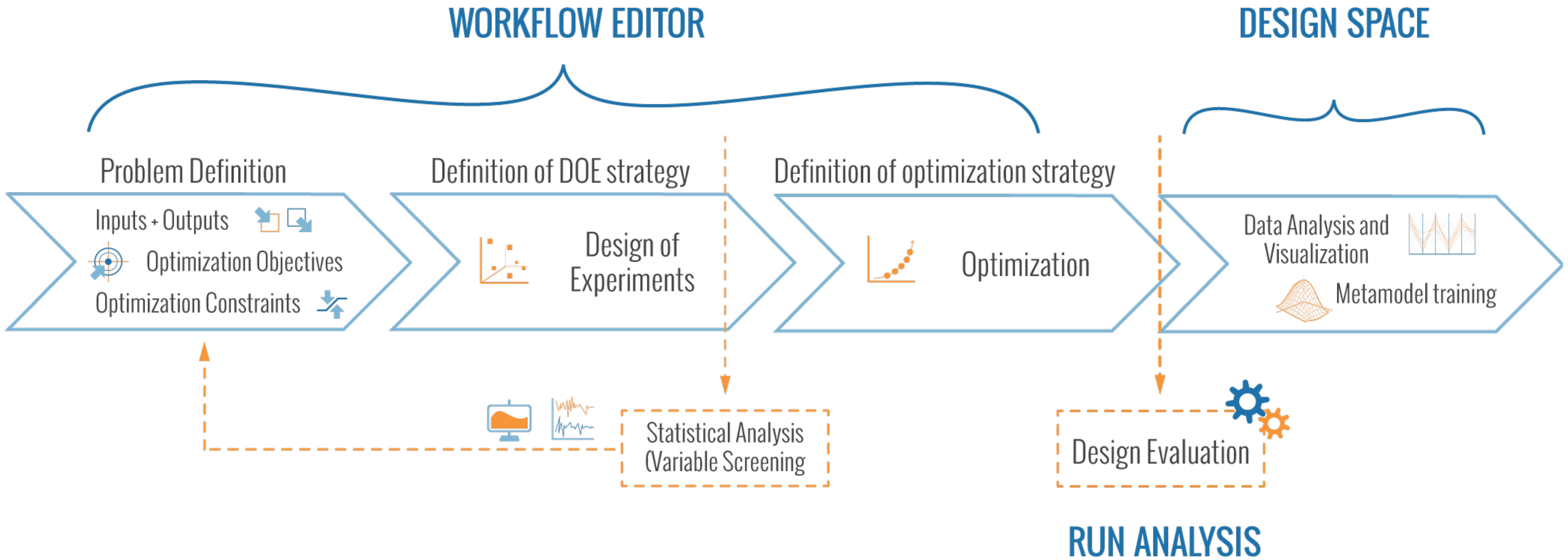
Multi-objective problems are solved using **sophisticated optimization algorithms**, which identify a set of **Pareto designs** whose objective functions are non-dominated by any other design among those tested.

How does modeFRONTIER work?

modeFRONTIER is divided in 3 environments,

- ✓ Workflow
- ✓ Run Analysis
- ✓ Design Space

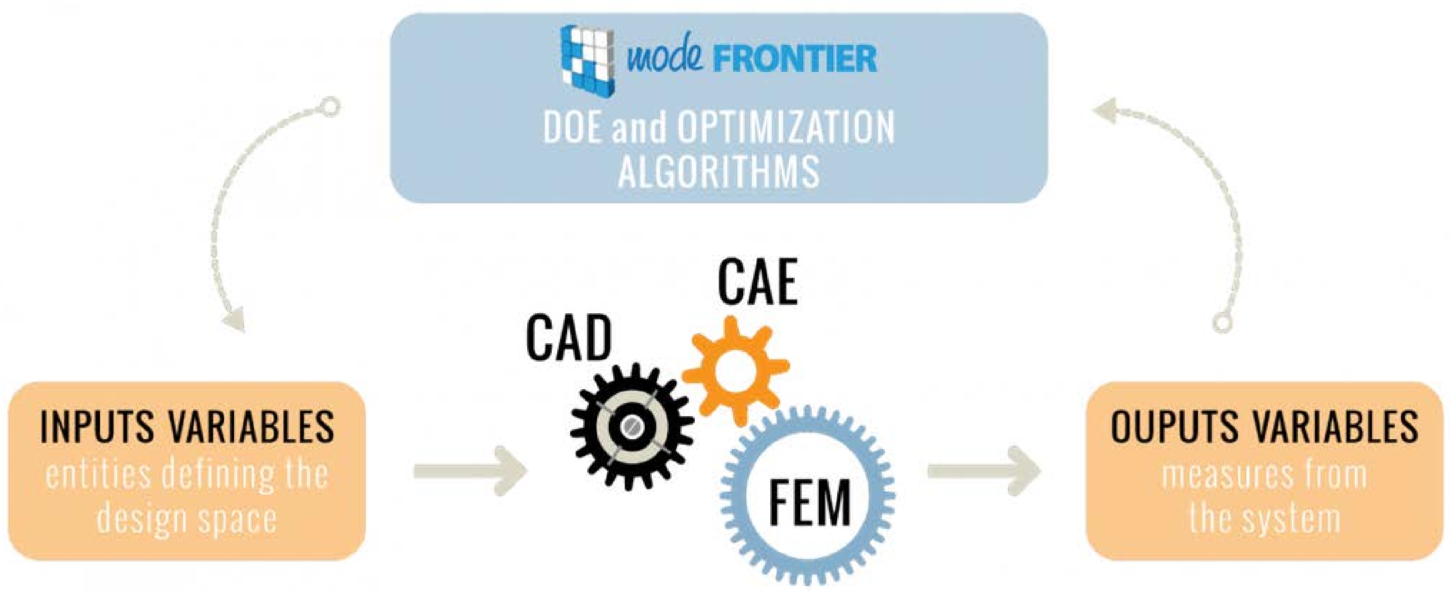
each of which is used to accomplish a specific set of tasks tailored to achieve the ultimate goal – **DESIGN OPTIMIZATION**.



Workflow composition

modeFRONTIER Workflow Editor enables to:

- compose and manage all logical steps of an engineering design problem involving software platforms used in **different disciplines**;
- **automate** the **simulation** process and drive the integrated software platforms, extracting and updating the generated data, and transferring them from one software to another.
- define the I/O variables and **objectives of the optimization**, and the **DOE and optimization strategies**

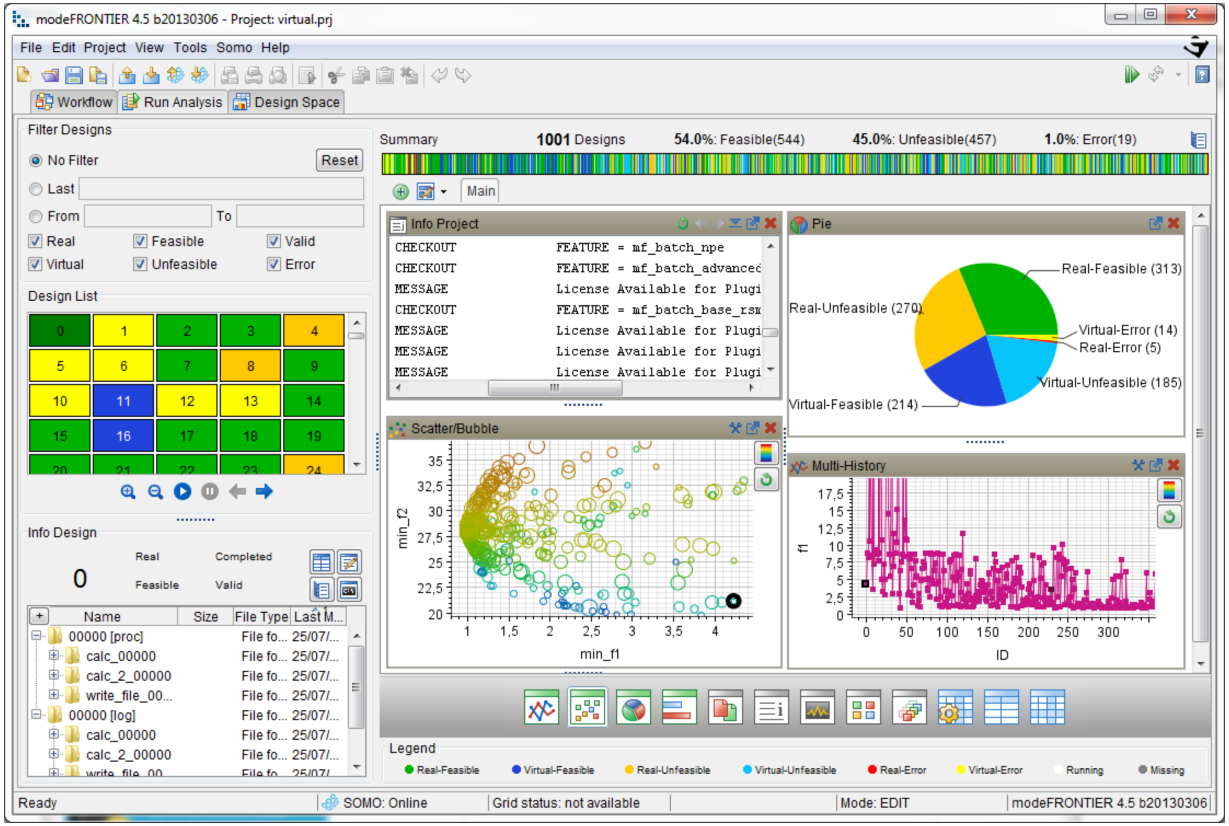


Optimization monitoring

The optimization "takes place" in the Run Analysis environment, enabling real-time monitoring of its progress by means of interactive charts and graphs, and direct access to log and process files.

The Run Analysis is:

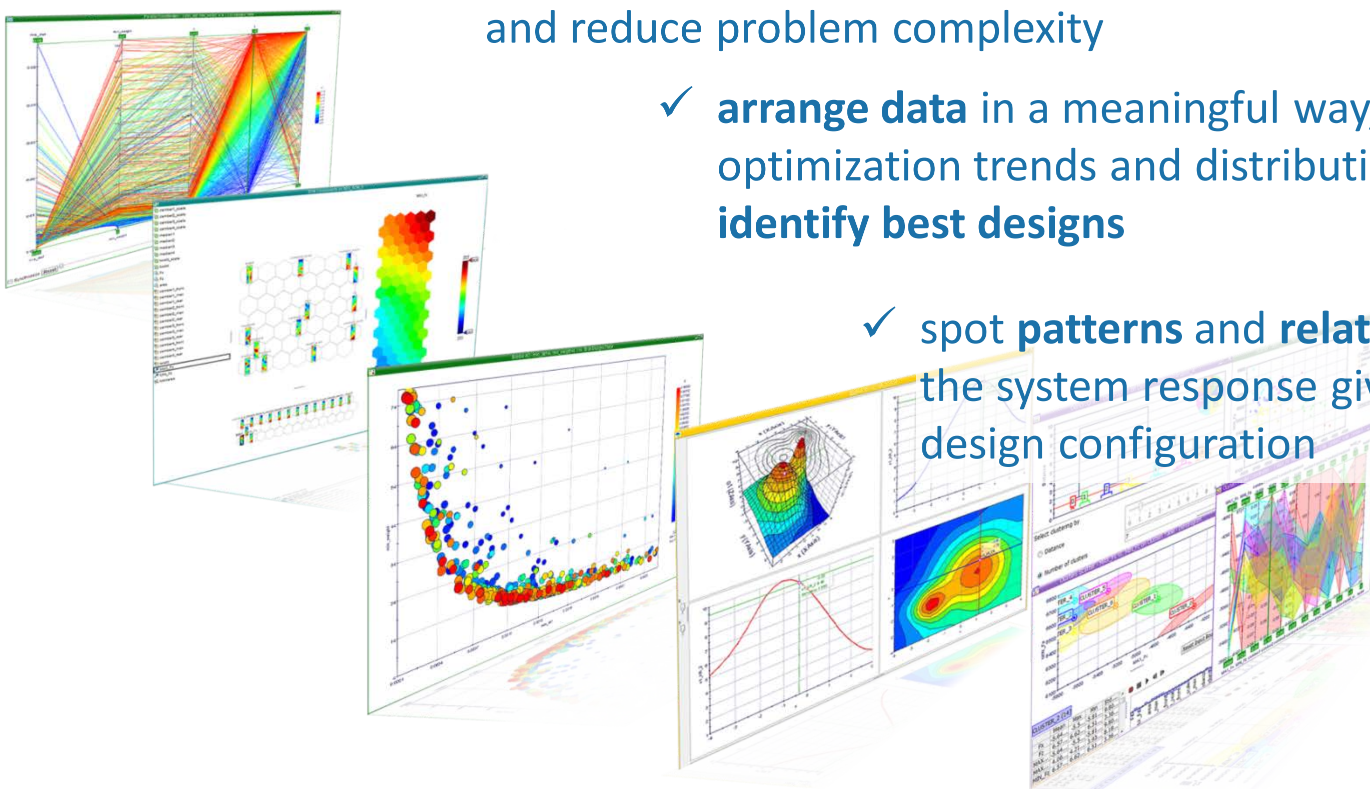
- entirely **modular** (the dashboard can be customized according to user's preferences)
- **dynamic** (as the optimization advances, all components are automatically updated)



Analyzing results

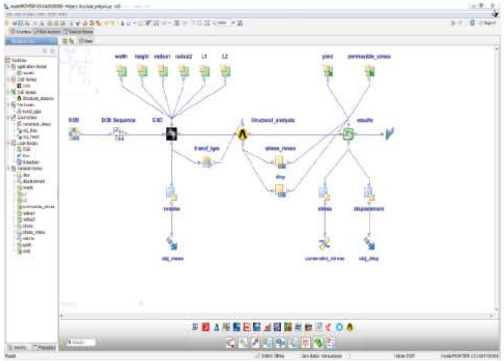
Information underlying the hundreds of complex numerical data generated from parametric studies or optimization campaigns can be **re-arranged and projected in the sophisticated charts and graphs** of the Design Space environment of modeFRONTIER.

- ✓ perform statistical analyses to identify the **most important variables** *a priori* and reduce problem complexity
- ✓ **arrange data** in a meaningful way, visualize optimization trends and distributions and **identify best designs**
- ✓ spot **patterns** and **relationships** governing the system response given a particular design configuration

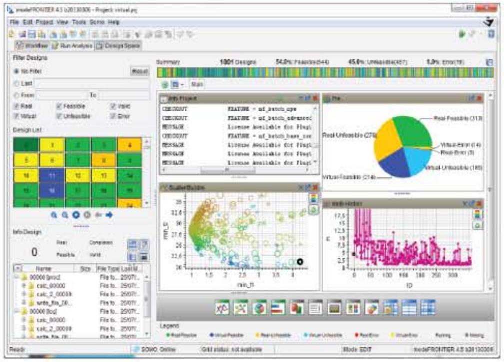


What can you do with modeFRONTIER?

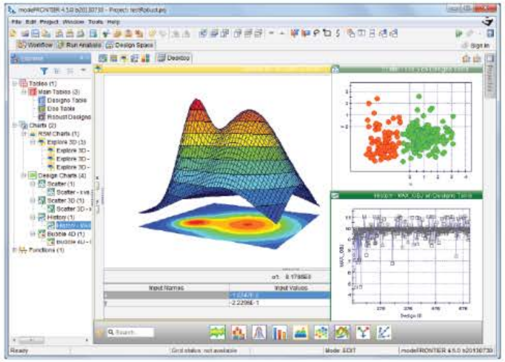
INTEGRATION AND PROCESS AUTOMATION



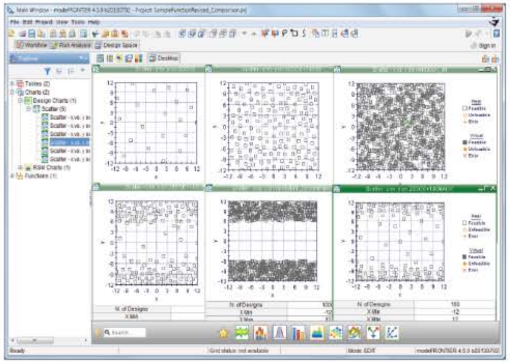
OPTIMIZATION



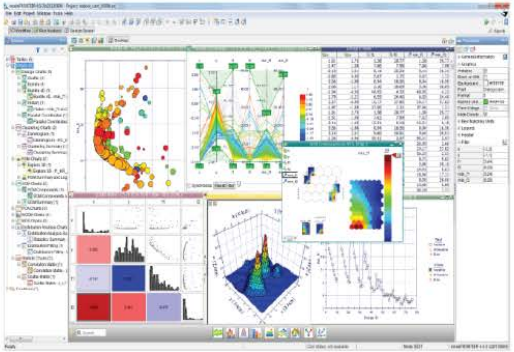
ROBUST DESIGN AND RELIABILITY



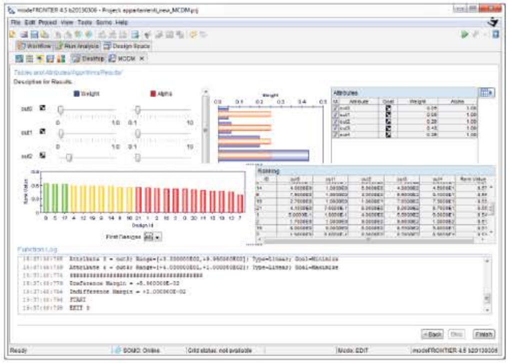
DESIGN SPACE EXPLORATION



ANALYTICS AND VISUALIZATION



DECISION MAKING



modeFRONTIER

Process Automation with the Workflow Editor

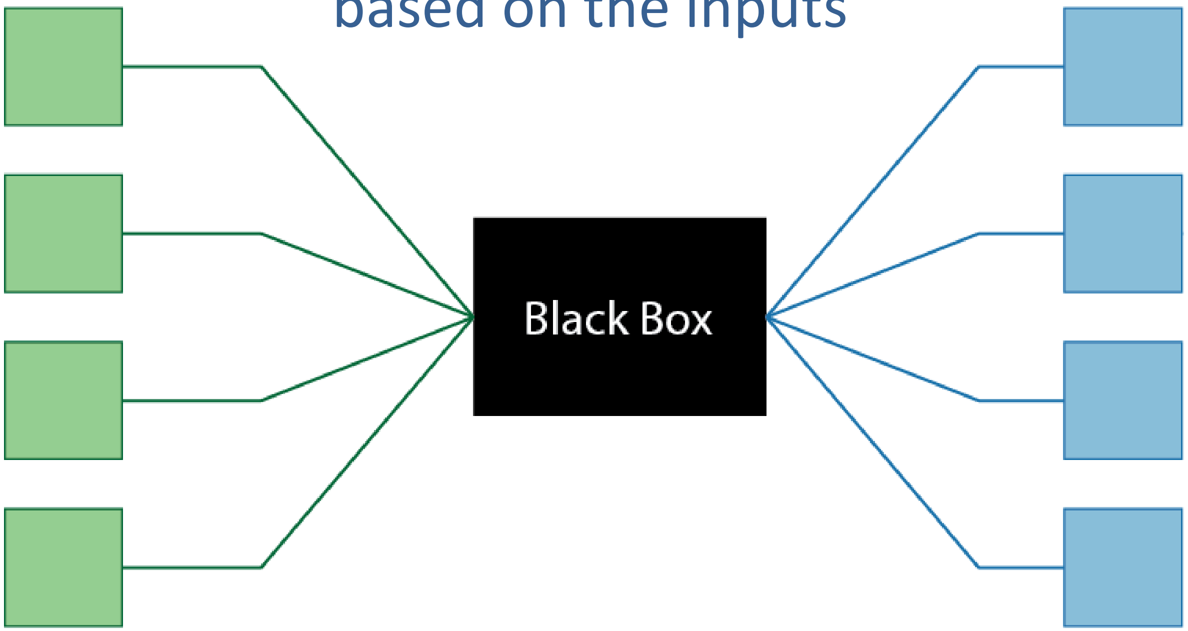
The concept behind modeFRONTIER

modeFRONTIER

Input Variables
Entities defining the Design Space

Black Box
Computes outputs based on the inputs

Output Variables
Measures or responses of the system



Input Variables

Input Variables are the free parameters of a model:

- ✓ the quantities that the engineer can change and control

and/or

- ✓ choices he can make concerning his design

Variables can be either:

- ✓ Discrete (those with a well defined finite set of possible values) , e.g. components from a catalog or number of components
- ✓ Continuous (those that can take on a value between any other two values), e.g. point coordinates, process variables

Output Variables

Output Variables are **measures or responses of the System**. The engineer has no direct control over these variables, but only through the modification of inputs

In optimization there are two more essential types of variables:

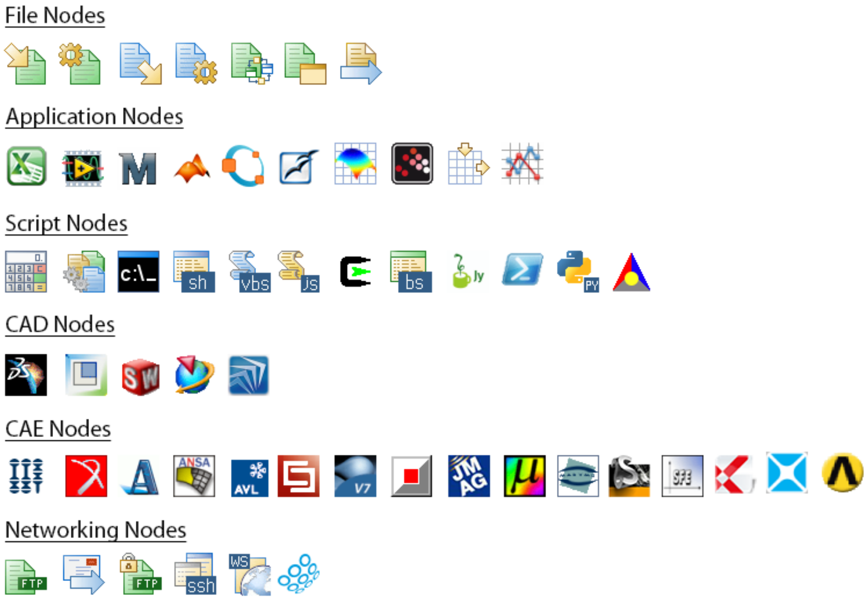
- ✓ Objectives – the most important features of a design, expressed in term of functions to be maximized or minimized, and describing a relationship of the input variables or the results of an operation using input variables
- ✓ Constraints – restrictions imposed on the design due to physical limitations, legislation, etc., and computed as functions of input/output variables. The constraints determine whether a design (combination of inputs and outputs) is feasible or unfeasible.

Black Box

The black box can be:

- ✓ A set of solvers that numerically model and solve the design problem (e.g. CAD/CAE tools)
- ✓ A set of experiments that generates data (e.g. tests in a laboratory)

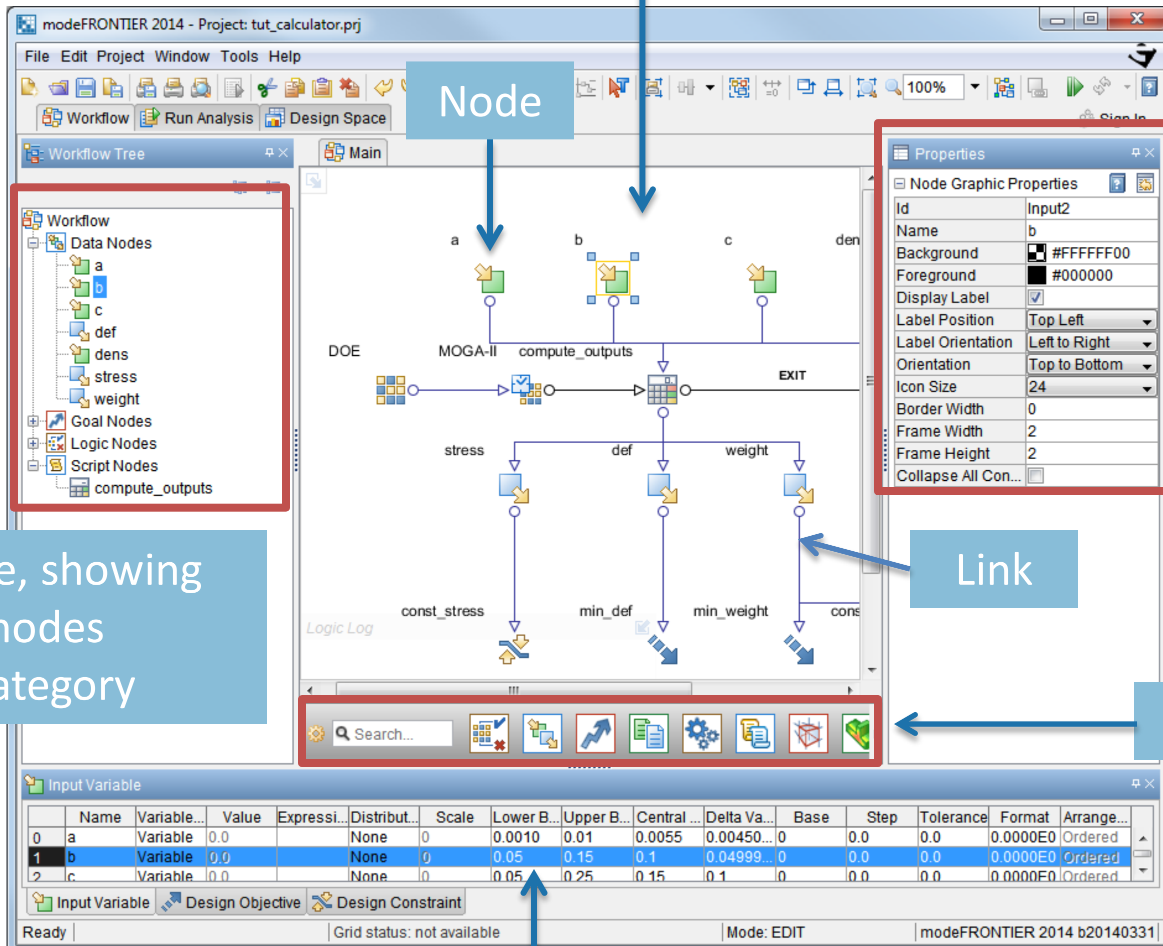
The modeFRONTIER workflow guarantees formalization and management of all logical steps of an engineering process. Its powerful integration capabilities allow product engineers and designers to integrate and drive multiple Computed Aided Engineering (CAE) tools.



Workflow Editor

Workflow canvas

Node



Graphical Properties of the selected workflow component (node or link)

Link

Workflow tree, showing all workflow nodes divided per category

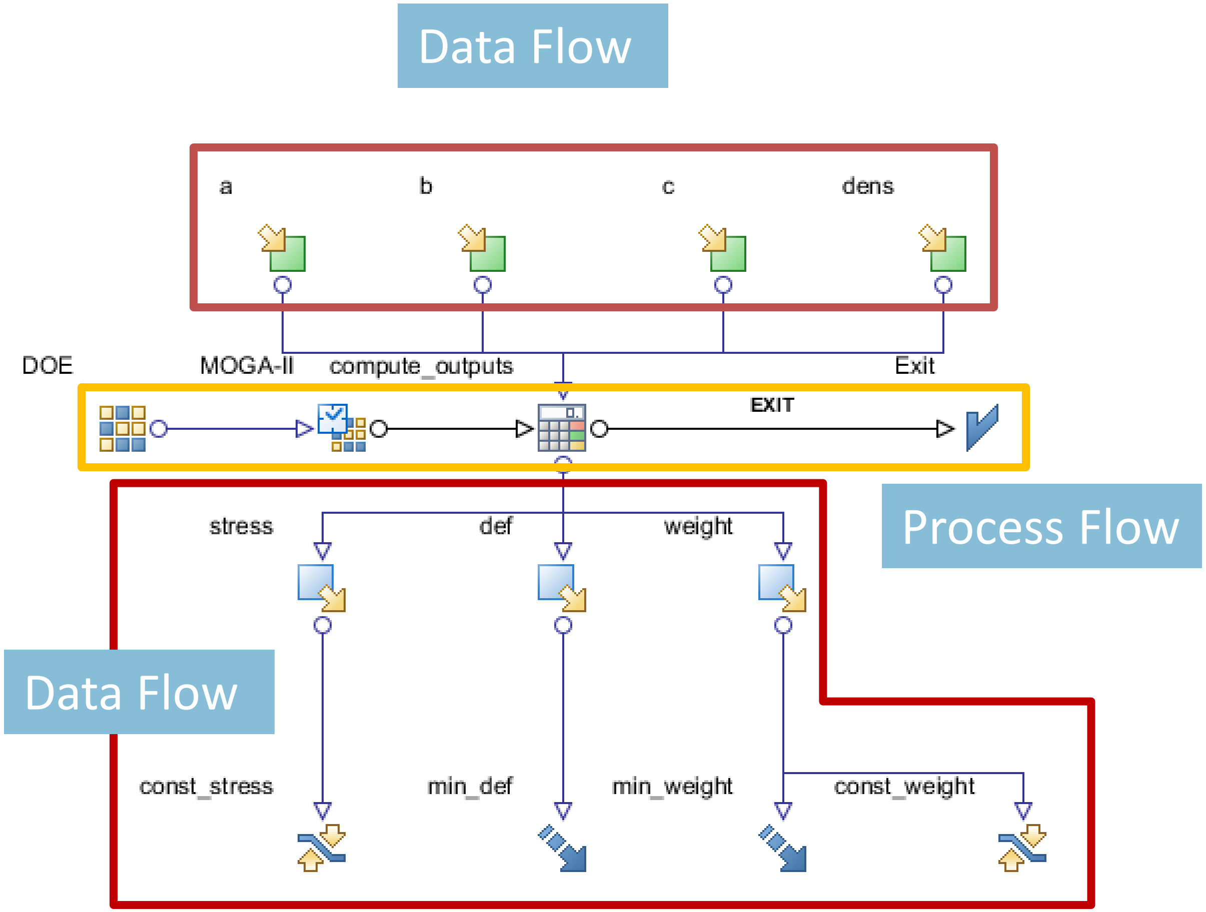
Node Palette

Summary panels for editing variable properties (e.g. range, step, value, function, etc.)

Workflow

The Workflow is the combination of the **Process Flow and the Data Flow.**

Typically, the Process Flow is represented by a set of interconnected application and logic nodes, whereas the Data Flow is represented by a set of variable, buffer and file nodes that link one application to another.



Bricks of an optimization problem – workflow nodes

- ✓ **Input variable nodes**: define the model to be optimized. Their values can fluctuate within a defined **range** according to their **base** (**continuous, discrete or categorical values**)
- ✓ **Application nodes**: software or scripts (solvers) which evaluate the behavior of the model to be optimized
- ✓ **Output variable nodes**: responses on the model behavior upon evaluation, generated by the solver
- ✓ **Objectives**: functions of the output parameters, dictating values to be obtained
- ✓ **Constraints**: values to be respected by a single or a combination of inputs/outputs
- ✓ **Logic Nodes**: nodes indicating the beginning and the end of the evaluation process. Also used for defining the **Design of Experiments** and the **Optimization strategy**.

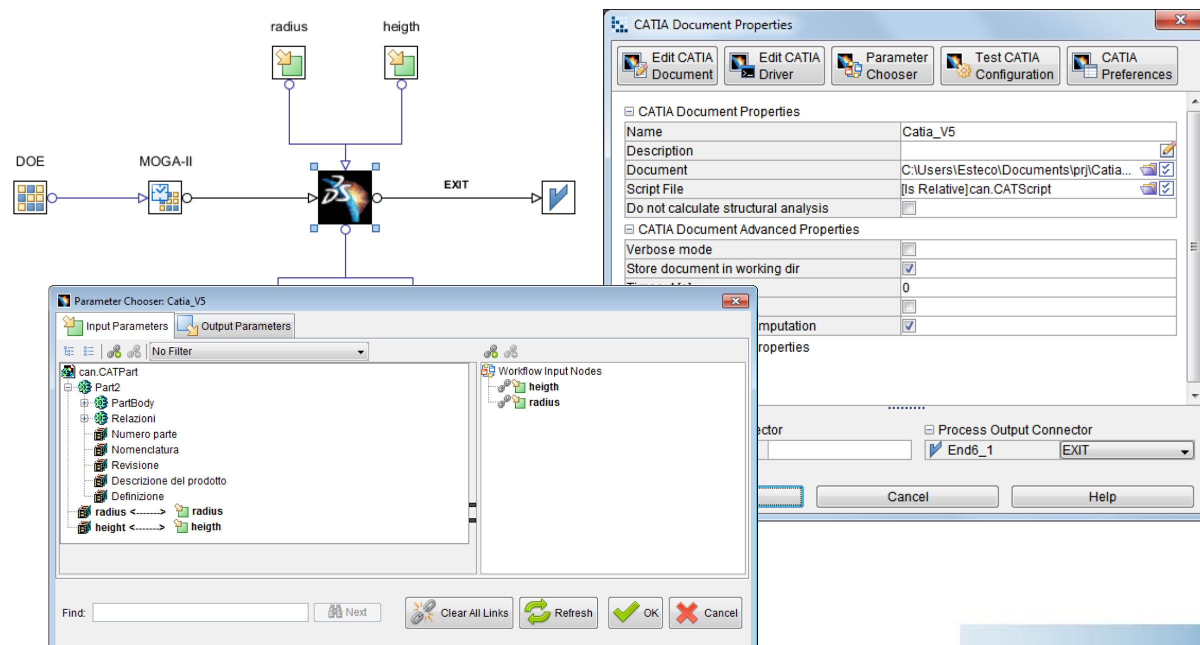
CAD/CAE Nodes: Parameter Chooser

The **Parameter Chooser** enables the automatic workflow building by detecting the input and output parameters specified within the loaded model , and by:

- ✓ creating the corresponding input and output variables in the workflow and establishing links with them, and/or
- ✓ adding links to already existing input and output variables in the workflow

The parameter search process is called **introspection**.

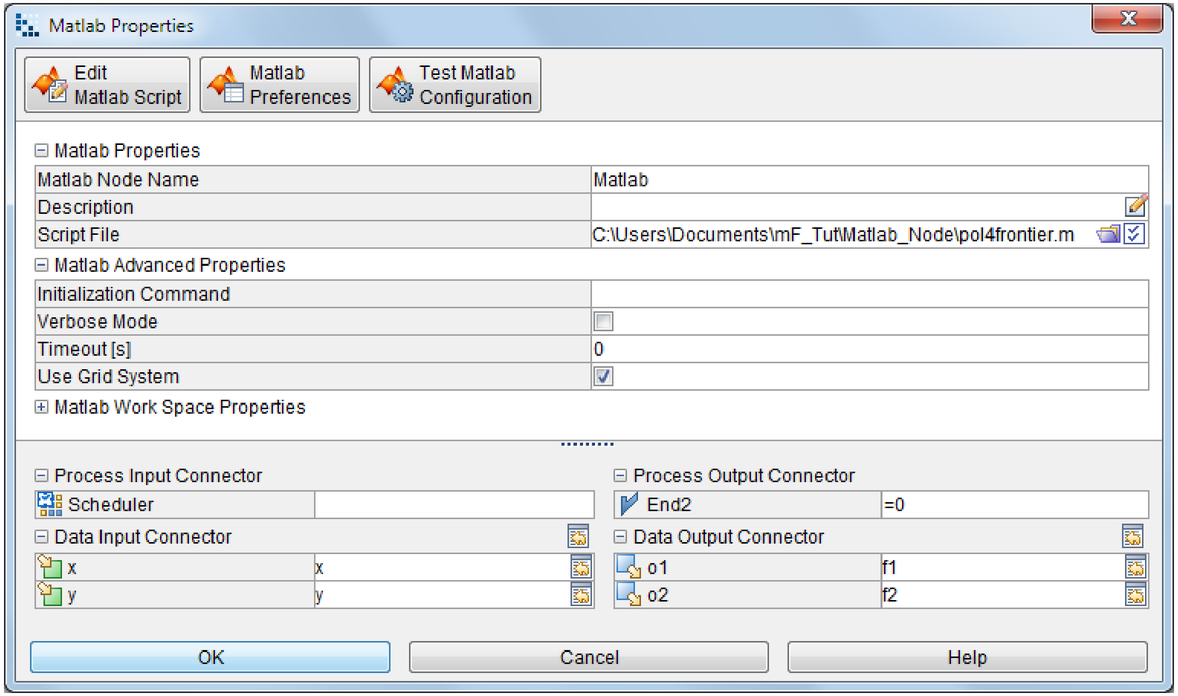
catia_pm.mp4



Linking parameters in other nodes - Matlab

Input and output variable nodes must be inserted in the workflow and connected to the Matlab node before linking them to Matlab parameters. Only then will they appear in the **Data Input Connector/Data Output Connector** panels in the bottom part of the Matlab Properties dialog.

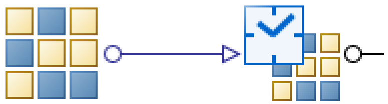
To link Matlab script parameters with mF variable nodes it is sufficient to **write the correct variable name (as in the Matlab script)** in the field next to the corresponding modeFRONTIER variable.

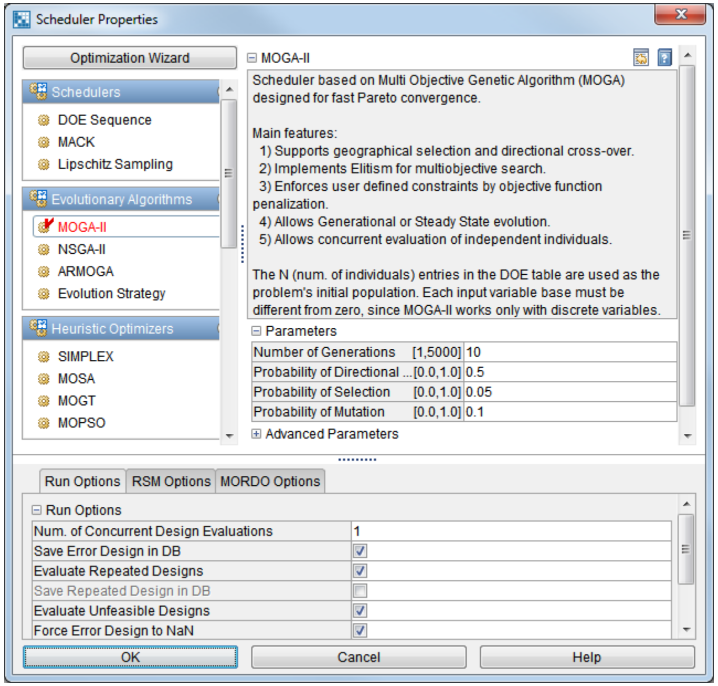


Configure DOE and Optimization algorithms

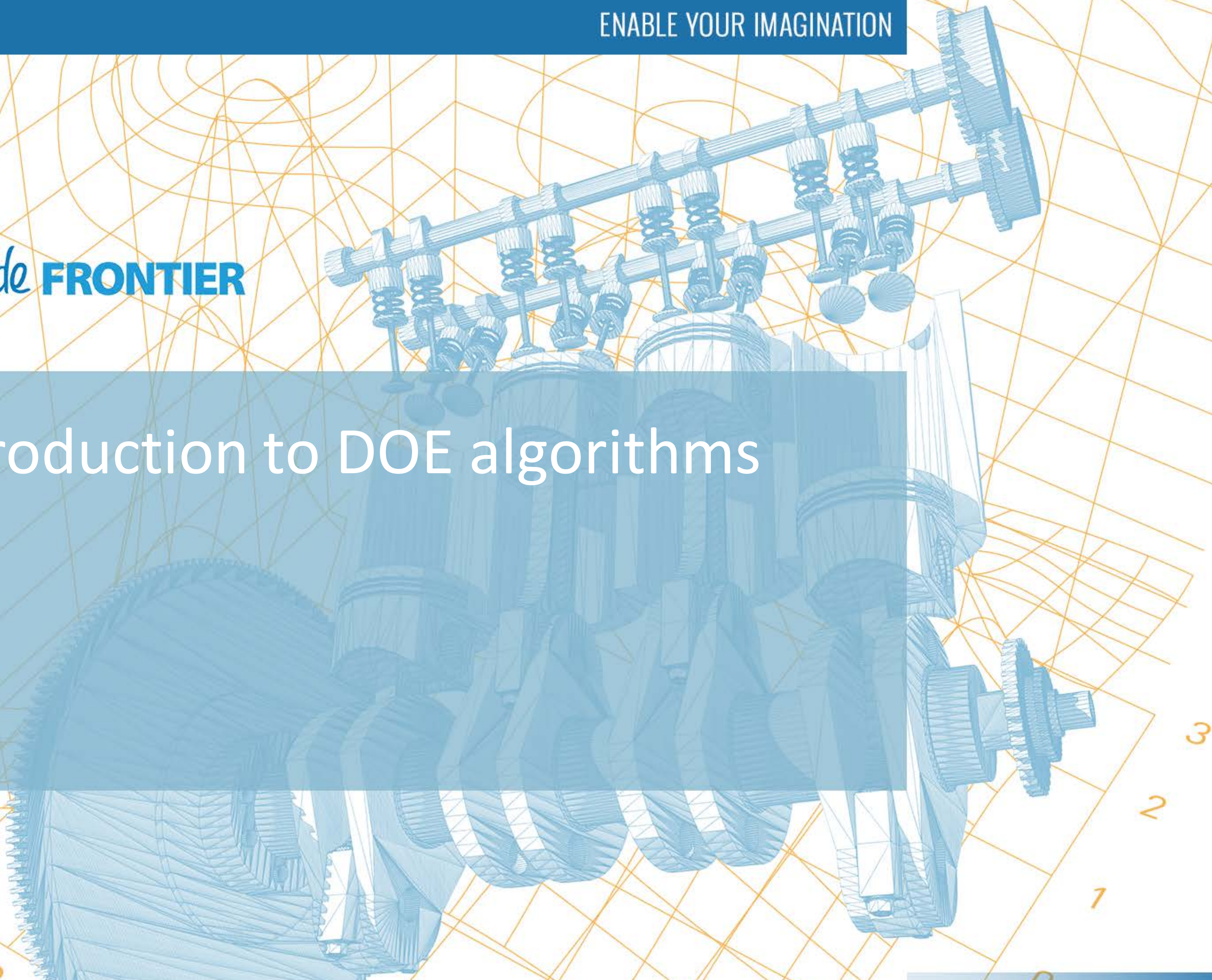
The Scheduler Node is made of two distinct nodes (automatically connected each other), which represent **two distinct phases** of the optimization process, namely:

- ✓ **Design Of Experiments (DOE)**, which supplies a sequence of different design configurations created according to a specific algorithm (to perform a preliminary exploration of the design space), and
- ✓ **Scheduler**, which enables the selection of the appropriate optimization strategy to be applied by using the specified DOE Sequence.





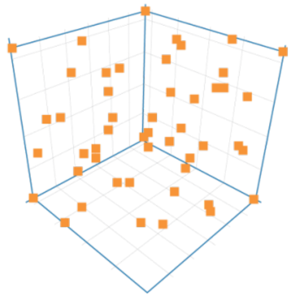
modeFRONTIER

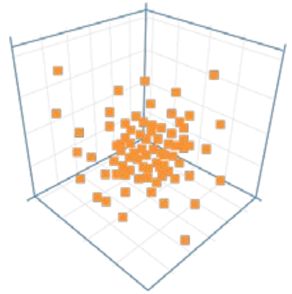


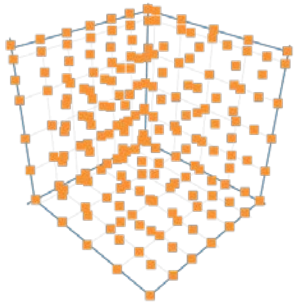
Introduction to DOE algorithms

Definition of Design of Experiments (DOE)

An important preliminary step of an optimisation process is the **initial sampling of the design space** (region of interest in which the values of the input variables are scattered).







DOE can help answer the following questions

- ✓ which are the most important design variables?
- ✓ can we reduce the design space (number of variables and their range)?
- ✓ are some variables related and how?
- ✓ what is a reasonable number of objectives and constraints in my optimization project?

How to choose the suitable DOE method and size?

modeFRONTIER

DOE for Statistical Analysis: correlations between variables should be as low as possible. Recommended algorithms: Factorial DOEs, ULH, Latin Square

DOE for RSM training: uniformity of DOE should be as high as possible for first training . The higher the number of samples, the better the RSM reliability. Recommended algorithms: ULH, Latin Square

DOE for optimization algorithms: each optimization algorithm requires a different DOE size. Robust optimizers are little influenced by DOE quality. Recommended algorithms: ULH, Random, Sobol (not for Simplex optimizer)

General empirical formula:
2 * num_inputs * num_objs

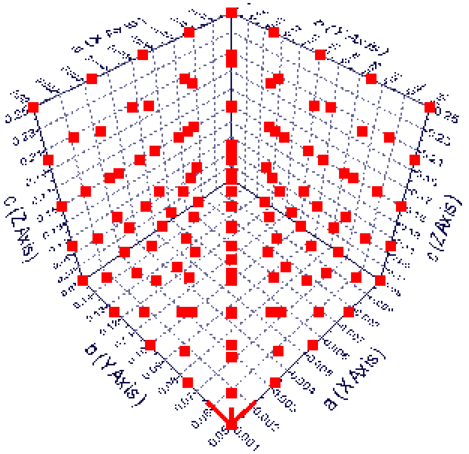
but:

✓ you must be aware of the computational resources you have available and those required by your solver (e.g. CPU)

Factorial DOE

Full Factorial DOE measures the response of every possible combination of factors and factor levels to analyze main and interaction effects.

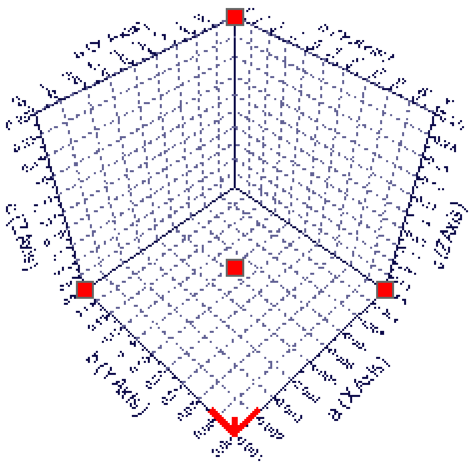
- ✓ Suitable for statistical analysis
- ✓ Computationally intensive with a large number of variables



Example: full factorial of 3 levels (L) with 3 variables (n) generates 27 (L^n number of) designs

Reduced Factorial DOE is composed by a representative subset of the Full Factorial DOE. The largest number of evaluations results in a 2-level Full Factorial.

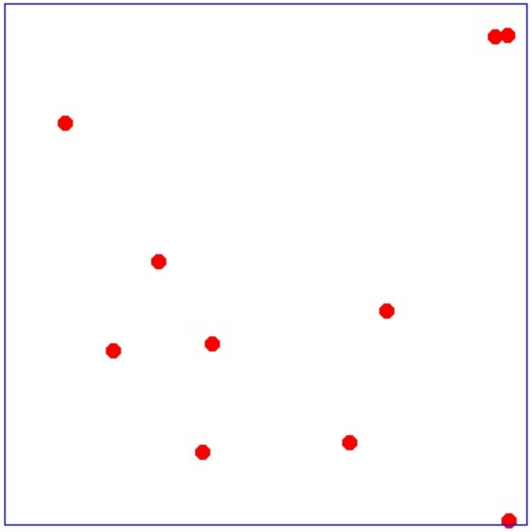
- ✓ Suitable for statistical analysis (tries to preserve main and interaction effects)
- ✓ Not suitable for optimization or RSM training



Example: reduced factorial with n variables generates 2^m number of designs, where m<n

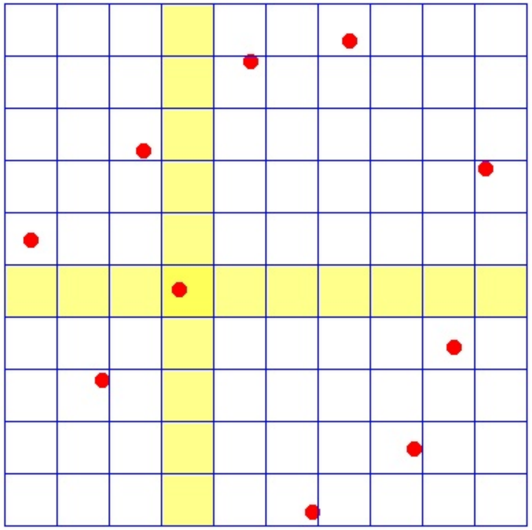
Uniform Latin Hypercube

- ✓ **stochastic space-filler** DOE algorithm (advanced Monte Carlo sampling)
- ✓ generates random numbers conforming to the uniform distribution
- ✓ achieves high **uniformity** levels for each variable
- ✓ tries to **minimize correlations** between input variables and **maximize the distance** between generated designs
- ✓ suitable for RSM training and GA optimization



Random (Monte Carlo)

n = 10

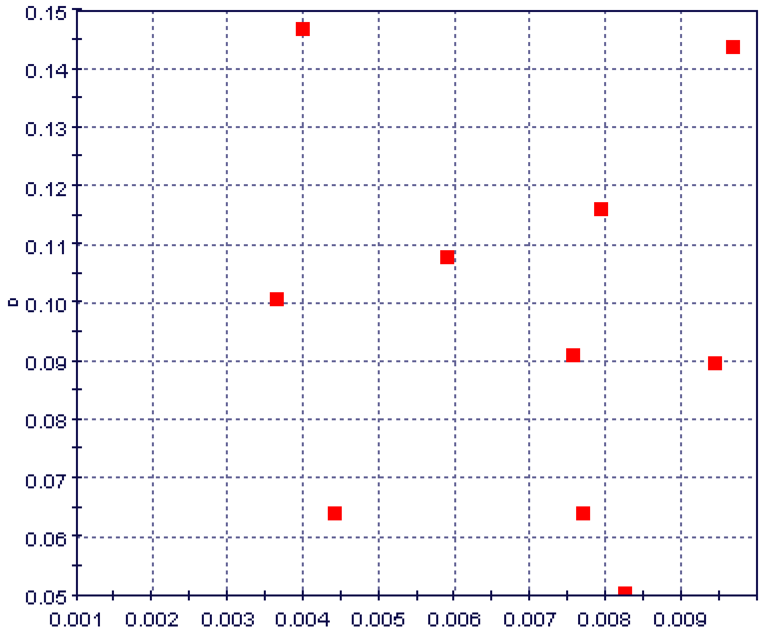


Uniform Latin Hypercube

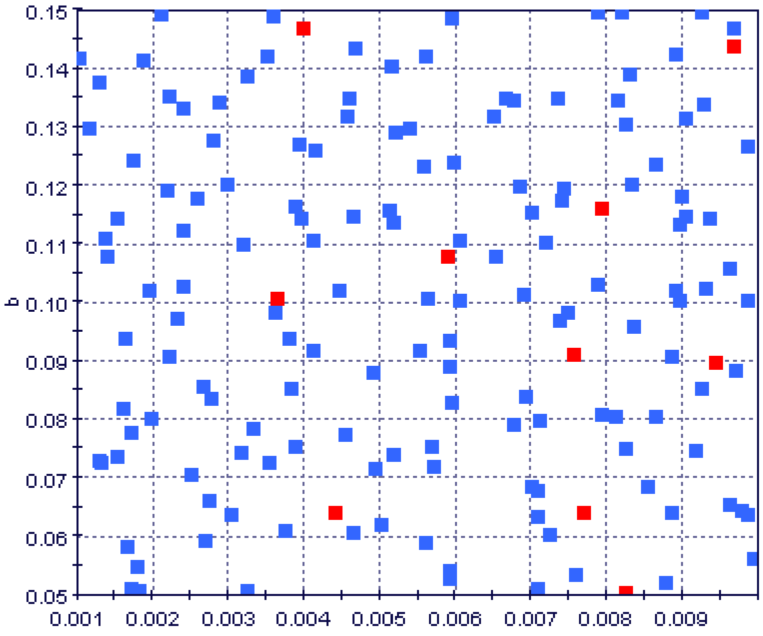
Incremental Space Filler

Augmenting algorithm considering the existing points and adding new points sequentially by maximizing the minimum distance from the existing points

- ✓ suitable for RSM training amd GA optimization
- ✓ uniform space filling
- ✓ rejects unfeasible designs



Initial DOE



Points added using ISF

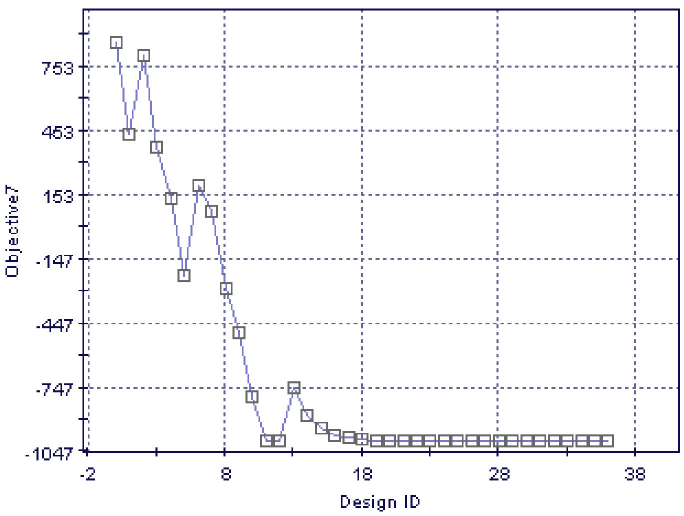
modeFRONTIER

Introduction to optimization strategies

Classification of optimizers

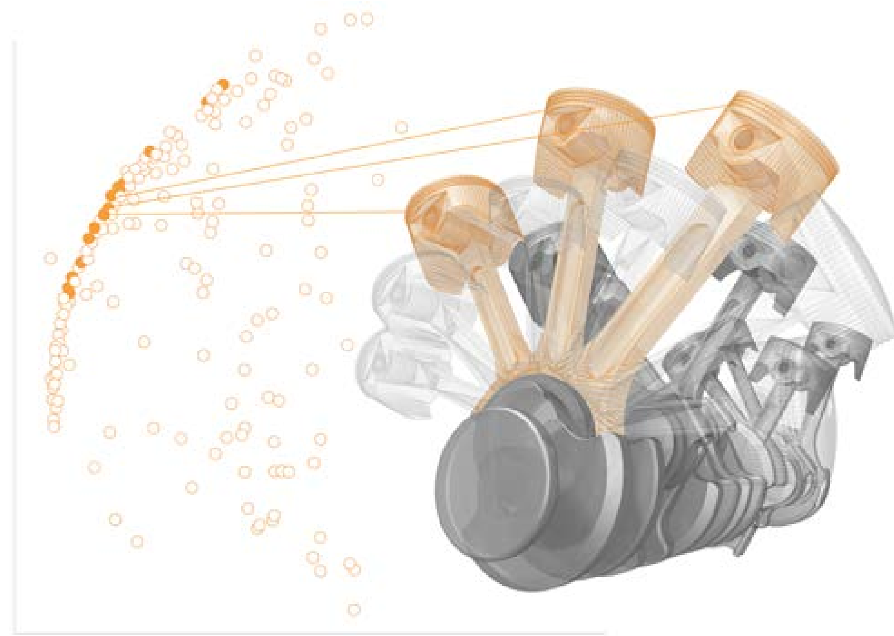
Local Search		Global Search		
Single objective	Multi objective	Single objective	Multi objective	
			Rapid	Robust
B-BFGS	NBI-NLPQLP	SIMPLEX	MOGT	MOGA-II
NLPQLP	NBI-AFSQP		FAST	NSGA-II
MIPSQP				ARMOGA
AFilterSQP				MOPSO
Levenberg-Marquardt				ES
Powell				MOSA
				HYBRID
				SAnGeA
				pilOPT

Single vs. Multi Objective



Single-objective optimization: the algorithm reaches convergence when the value of the given objective cannot be further improved (minimized or maximized)

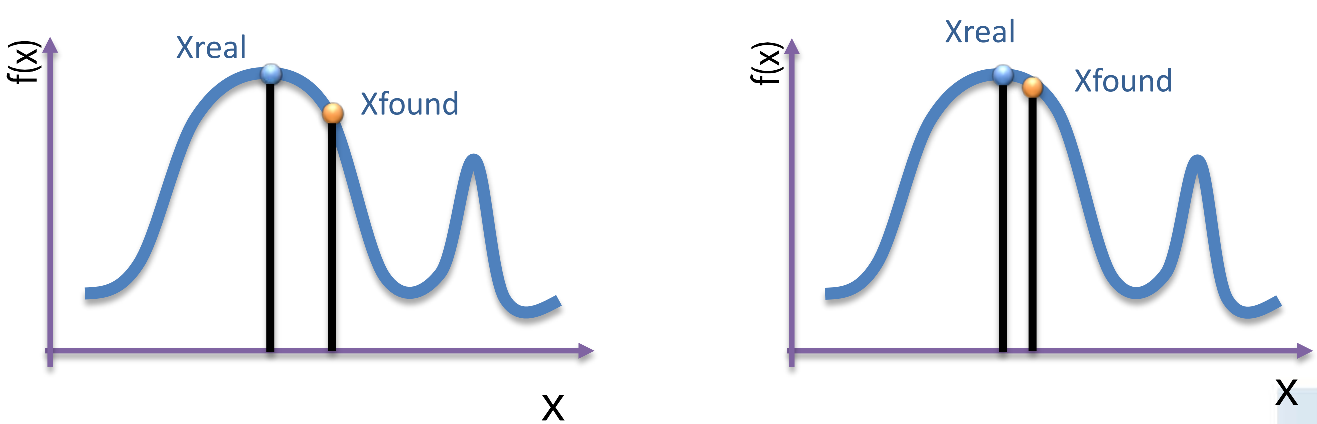
Multi-objective optimization: applied to problems involving multiple objective functions, none of which can be improved without deteriorating the performance of the other. Such solutions are said to be **non-dominated** and form the so called **Pareto Frontier**.



Robustness vs. Accuracy

The **robustness** of an optimization algorithm is the ability to reach the absolute extreme of the objective function.

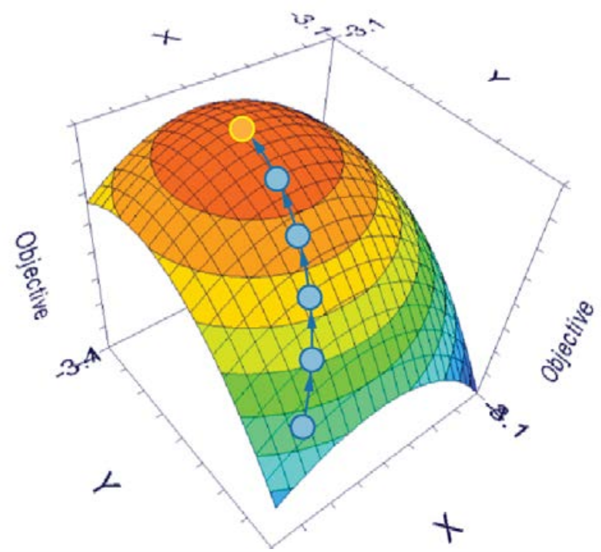
The **accuracy** measures the capability of the optimization algorithm to find the extreme of the objective function.

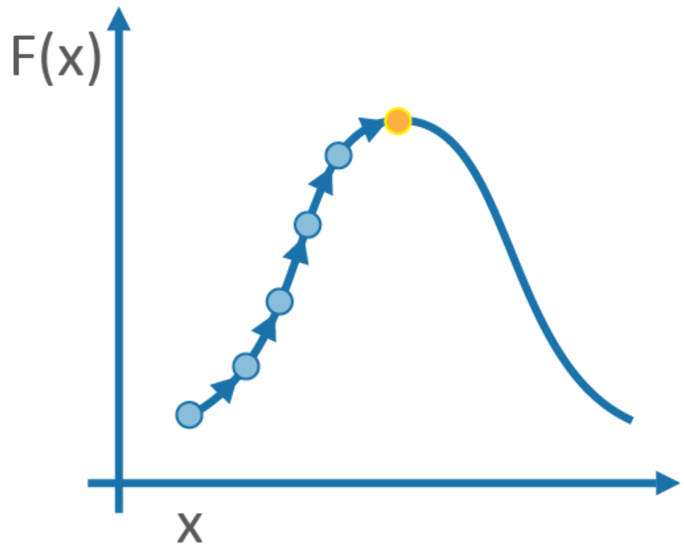


Gradient-based vs. Global Search

Gradient-based algorithms are iterative deterministic methods, which compute new points in the direction of the higher function increase (gradient). Derivatives are approximated by finite differences.

- ✓ Suitable for **local refinement** and any problem dimension, but they may require significant computational effort due to gradient computation
- ✓ **Low robustness** compensated by **high accuracy and convergence speed**
- ✓ May get stuck at **local optima**
- ✓ Intolerant to **noisy objective** functions
- ✓ Not suitable for discrete variables with a low base

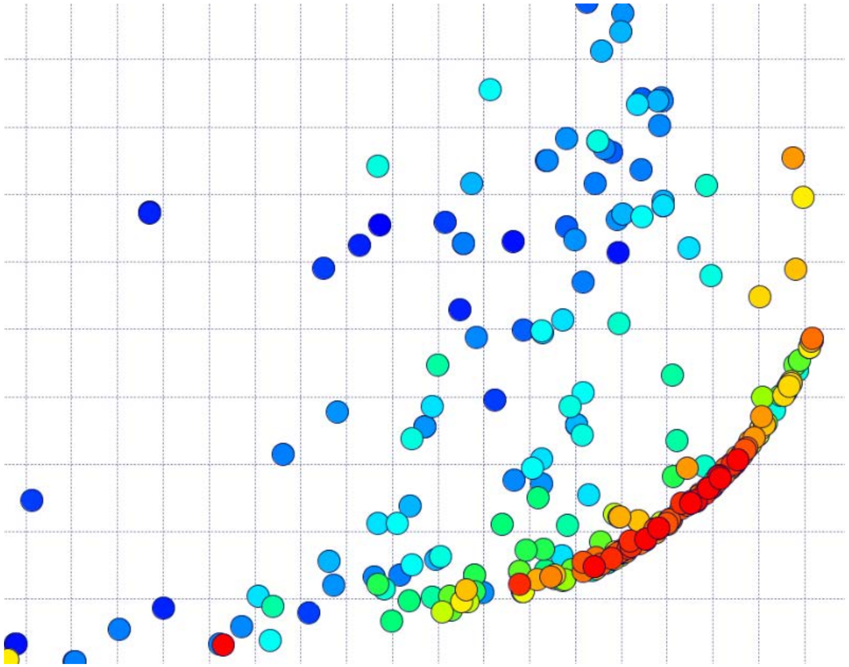




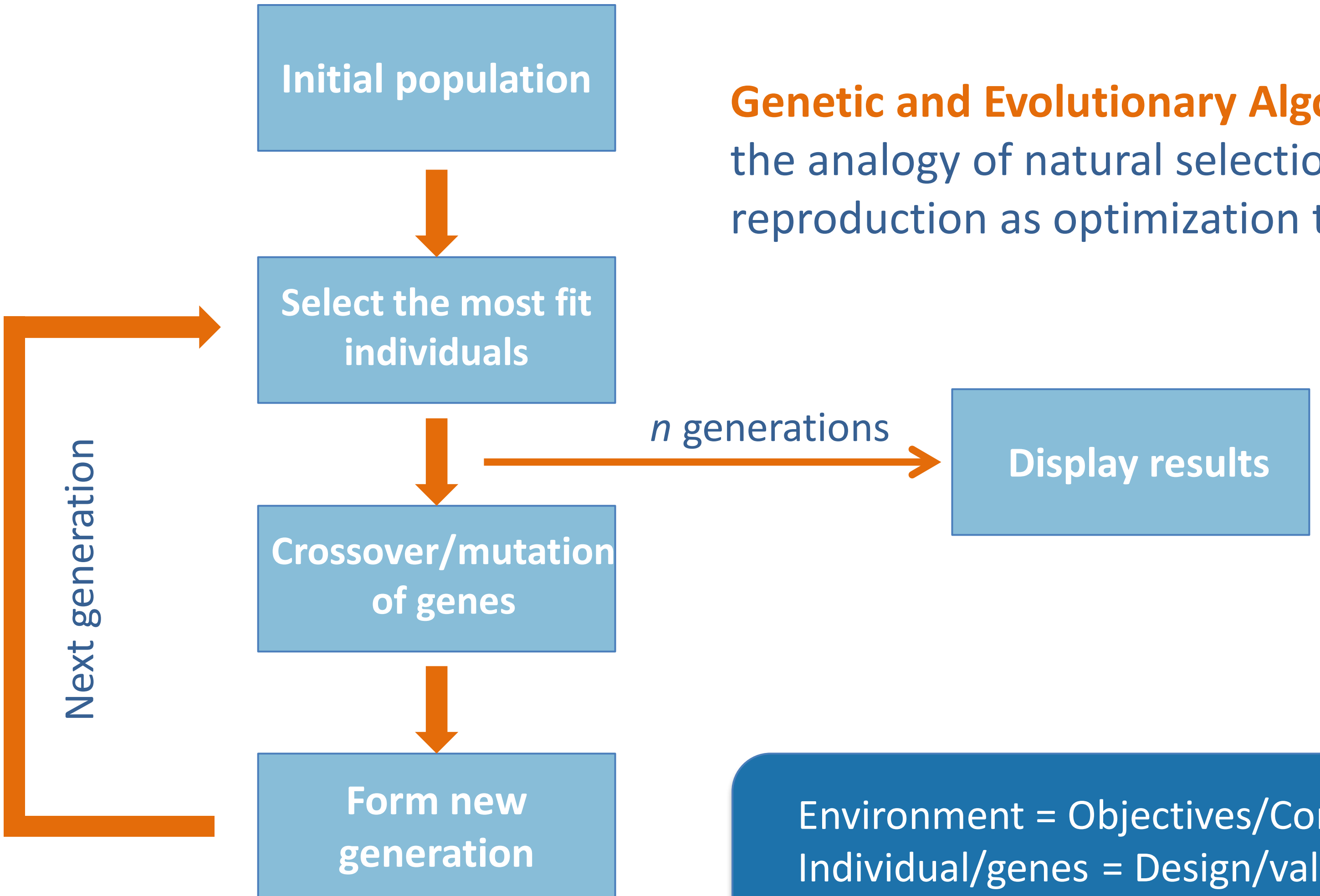
Gradient-based vs. Global Search

Global search algorithms are stochastic methods including heuristic, genetic and evolutionary optimization algorithms that introduce a certain degree of randomness into the search-process to accelerate progress, make the search less sensitive to modeling errors and escape a local optimum to eventually approach a global optimum.

- ✓ Good **global exploration** of the design space
- ✓ Suitable for **any problem dimension**
- ✓ **High robustness**, but **low convergence** rate if high accuracy is required (this problem can be overcome by using Response Surfaces, or RSMs)
- ✓ **Tolerant** to noisy objective functions
- ✓ Handle discrete variables



Genetic algorithms



Genetic and Evolutionary Algorithms use the analogy of natural selection and reproduction as optimization target.

Environment = Objectives/Constraints
Individual/genes = Design/values
Dominance = Solution Fitness

MOGA-II and NSGA-II

modeFRONTIER

MOGA-II and NSGA-II are efficient **single and multi-objective genetic algorithms**. Being very robust, they are also the most commonly used optimization algorithms.

General empirical rule for configuration of MOGA-II and NSGA-II

Maximum DOE size = 2 * n * o
n is the number of variables
o is the number of objectives

Minimum DOE size = 2 * n

Number of generations:
at least 10 generations needed to obtain an efficient search in many engineering problems, usually not more than 20-30 generations are required

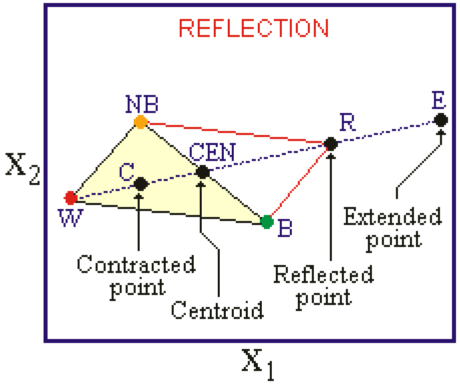
MOGA-II	NSGA-II
✓ Handles constraints by means of the Penalty Policy	✓ Handles constraints by means of the Pareto ranking (constraint domination principle)
✓ High performance with discrete variables	✓ High performance with continuous variables

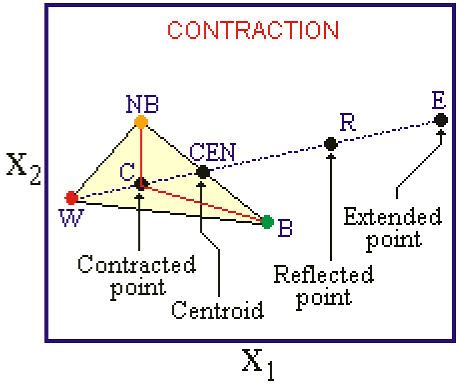
Simplex

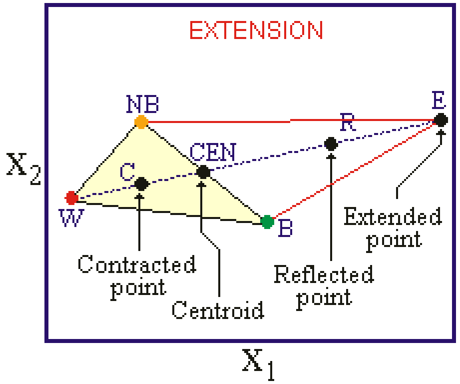
SIMPLEX is a **deterministic single-objective** algorithm indicated for non-linear optimization problems.
It does not require evaluations of derivatives, so it is more robust than the algorithm based on local gradients. It represents a **good compromise between genetic and gradient-based algorithms**.

A simplex is a **polyhedron** containing N+1 points in a N dimensional space (thus in two dimensions it is a triangle, in three dimensions tetrahedron and so forth).

The optimizer moves the initial points along with their function values, closer to the optimal point of the objective until it either exceeds the maximum number of iterations or converges. The movement of the simplex is given by four operations: “Reflection”, “Expansion” and “Contraction” and "Multiple Contraction".





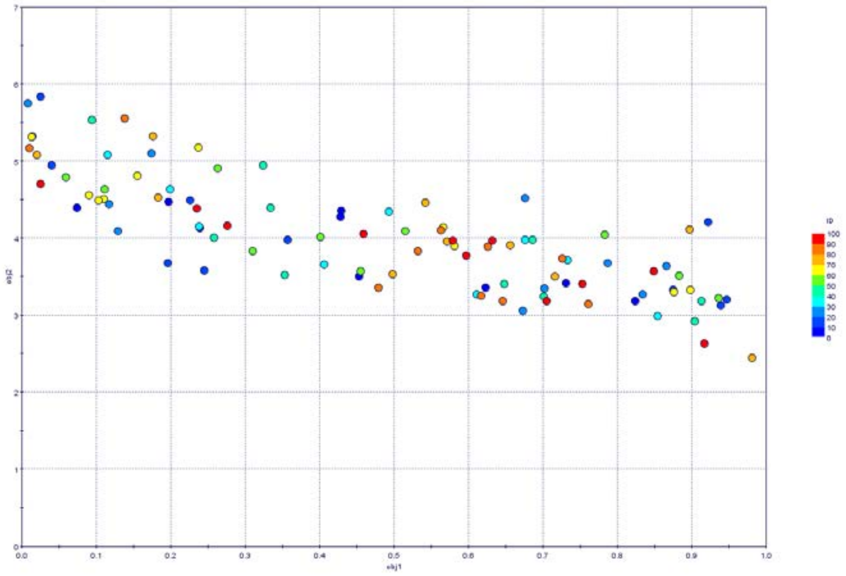


W=worst point
B=best point
NB=next best point

pilOPT

A multi-strategy, self-adapting hybrid algorithm that combines the advantages of local and global search algorithms while offsetting real and RSM-based optimization to search for the Pareto front.

- ✓ available from mF 2014
- ✓ initiated with a **single parameter** (number of design evaluations)
- ✓ efficient even with **complex output functions**
- ✓ dynamically **adjusts the ratio of real and virtual design evaluations** to optimize the use of computational resources



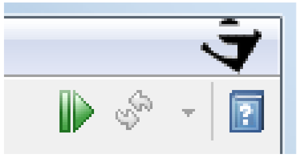
modeFRONTIER

Running the Optimization

Configure DOE and Optimizer; Start the Run

The Project Execution (Run) can be started only when the **Logic Log** presents **no errors**.

Start the Run ( ): you will be asked to save a project under a different name (not compulsory). mF creates a **Project Run Directory** (same name as *.prj file + sequential number)



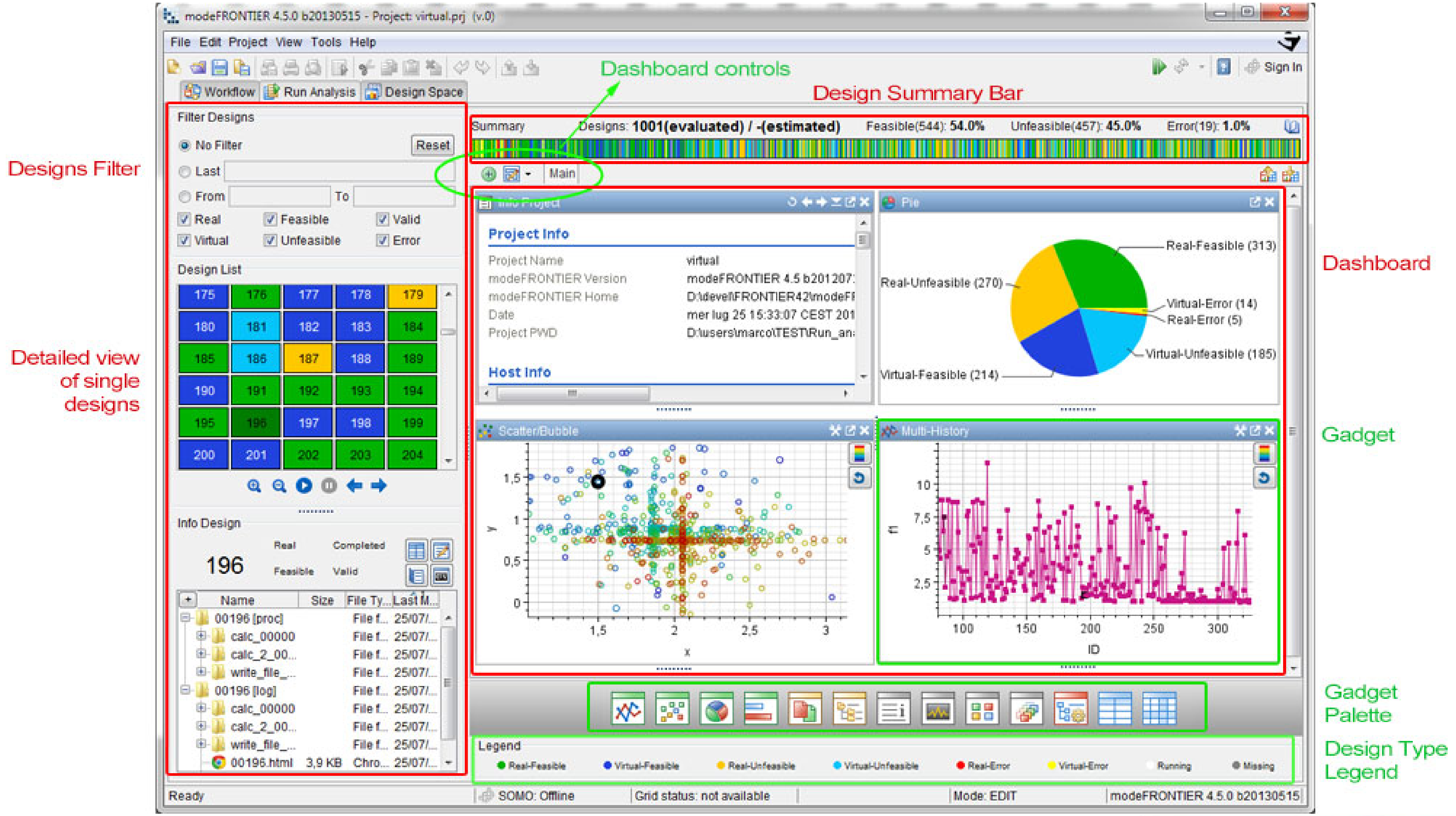
The interface switches automatically to the **Run Analysis** environment.

Define Refresh rate or refresh RA manually

Terminate the run (any design whose evaluation is interrupted is marked as Error design)

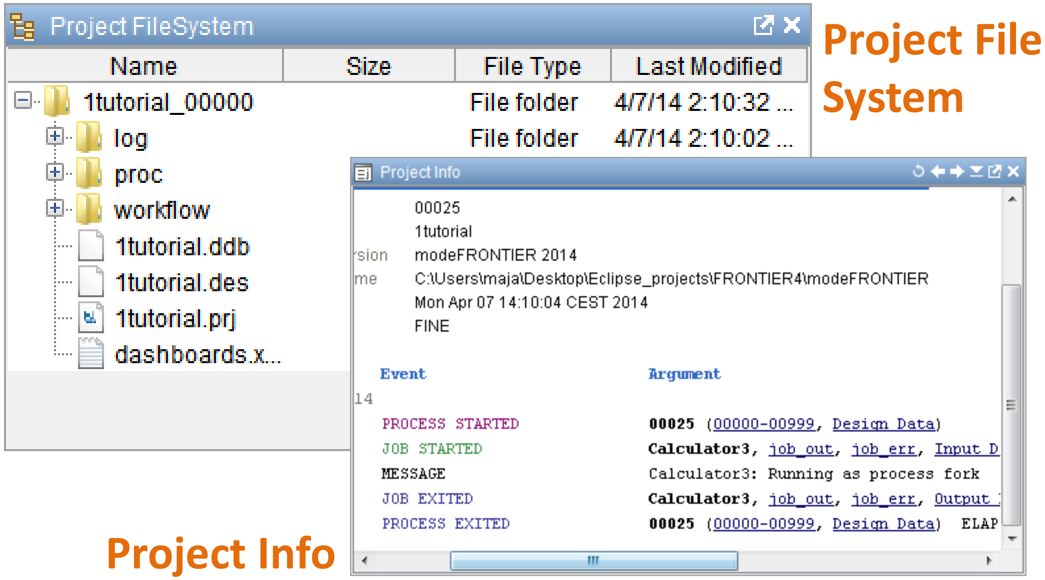
The Run Analysis

Real-time monitoring of the design evaluation progress, with direct links to log files



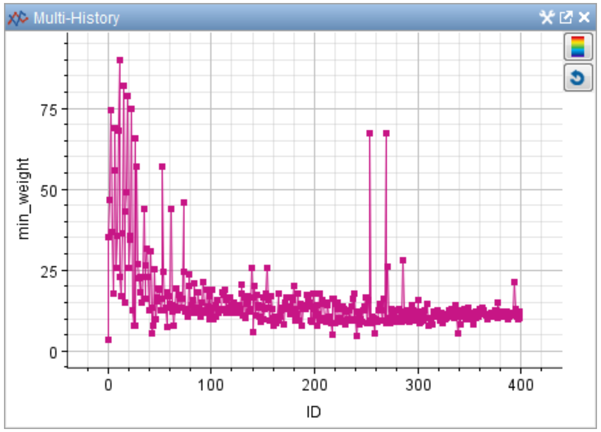
Gadgets

Provide both general information about the entire evaluation session and design-specific information.

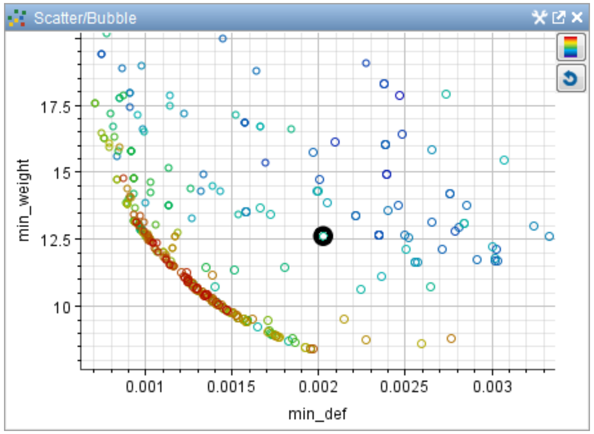


Project File System

Project Info

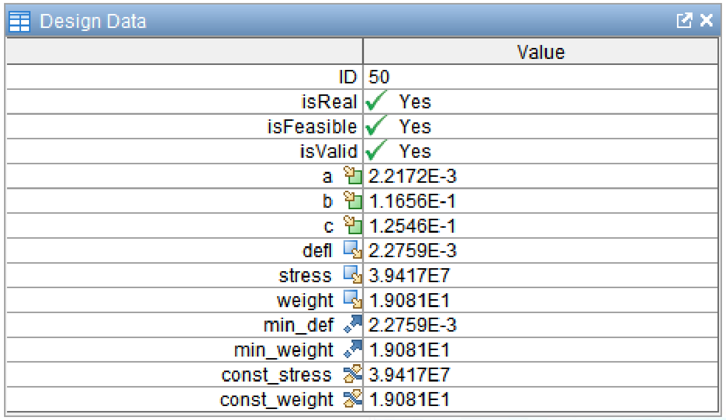


Multi-History Chart



Scatter/Bubble Chart

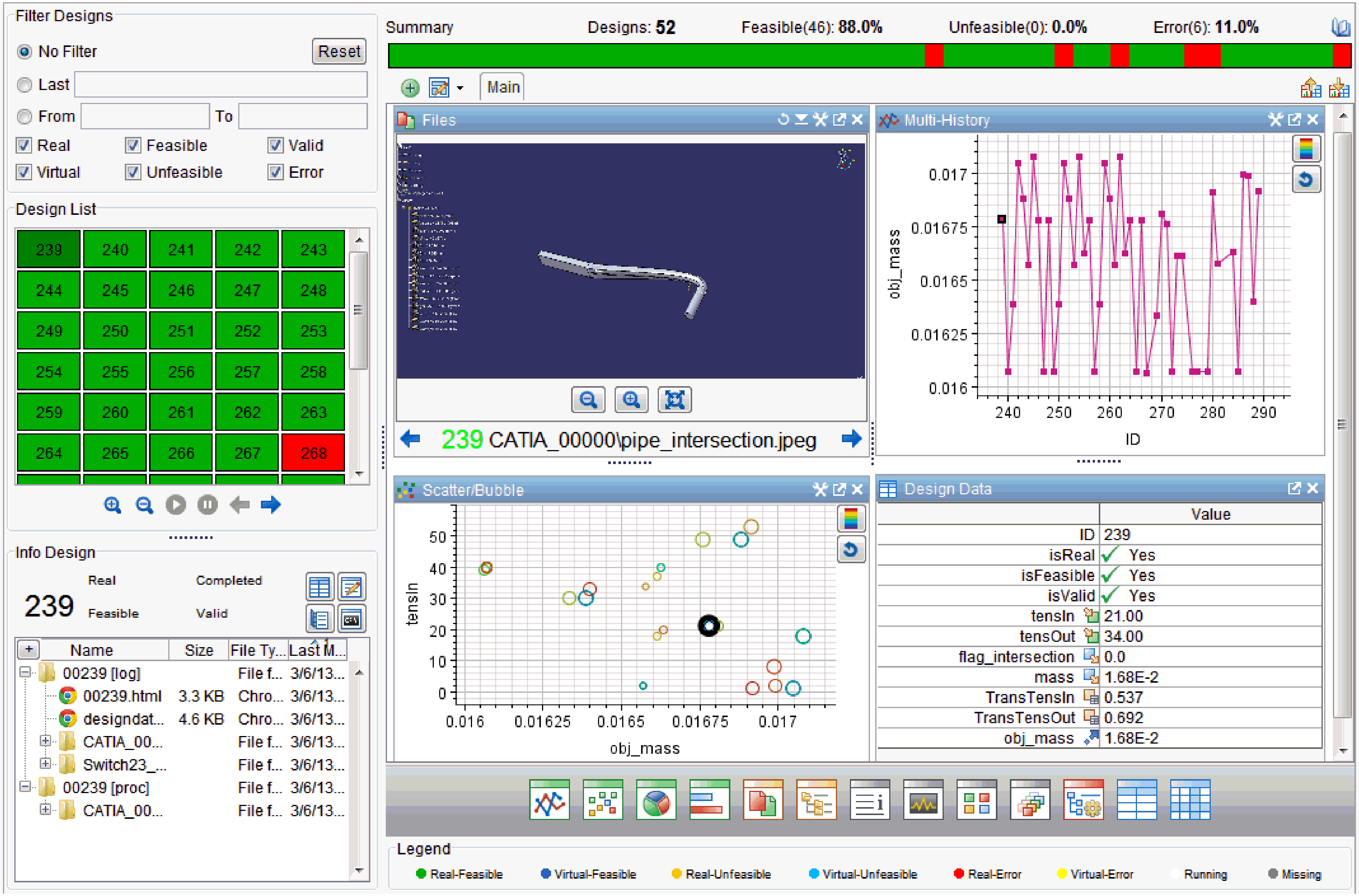
Design Data



Design Data

	Value
ID	50
isReal	Yes
isFeasible	Yes
isValid	Yes
a	2.2172E-3
b	1.1656E-1
c	1.2546E-1
defl	2.2759E-3
stress	3.9417E7
weight	1.9081E1
min_def	2.2759E-3
min_weight	1.9081E1
const_stress	3.9417E7
const_weight	1.9081E1

The Play Function

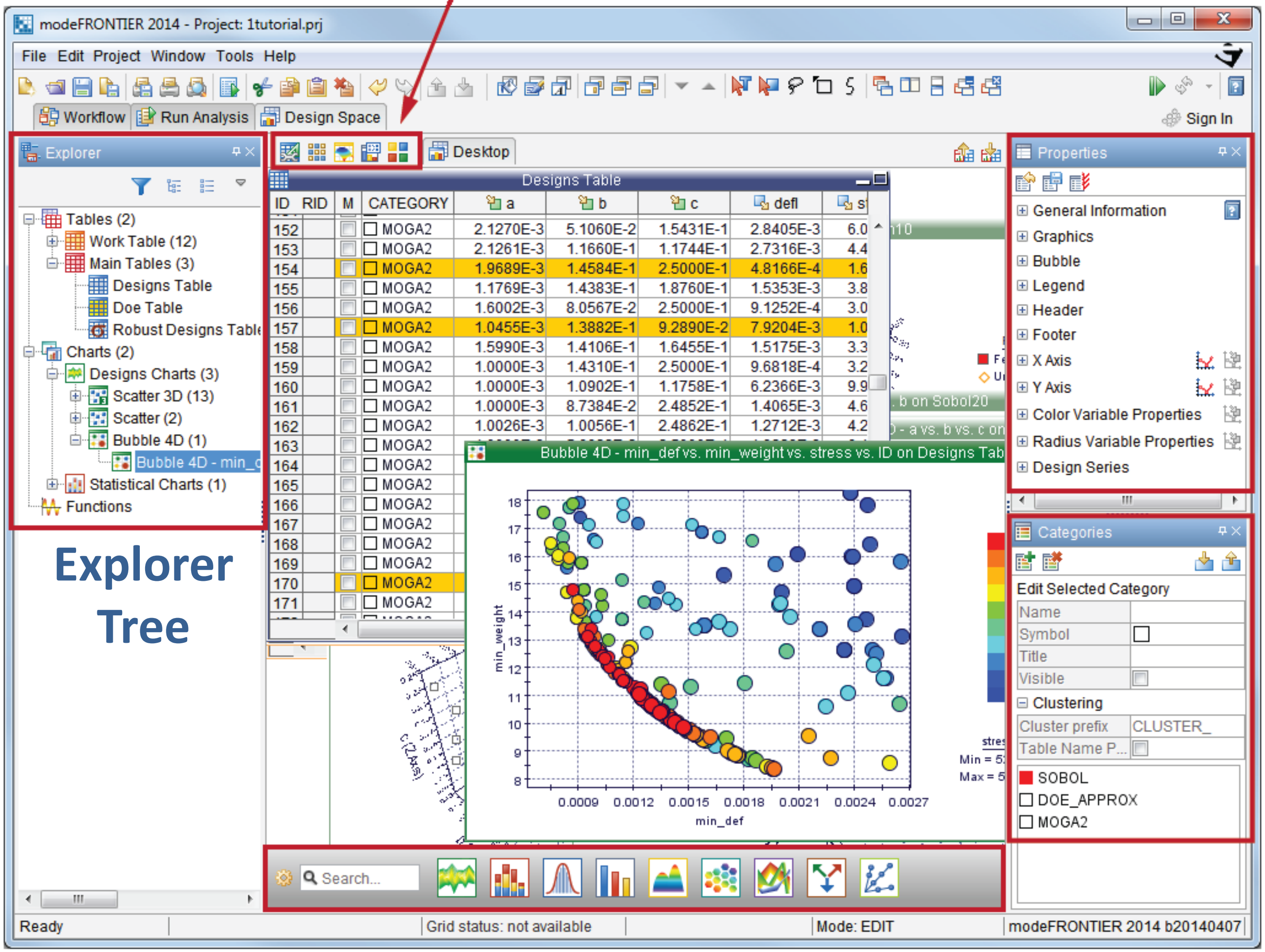


modeFRONTIER

Exploring basic post-processing tools

Design Space Components

Actions Toolbar



Explorer Tree

Properties Panel

Categories Panel

Gadget Palette

Tables

The Design Space is **table-oriented**, meaning that the charts are based on a particular type of table(s). If a table is removed, the chart will be automatically removed as well.

If a chart or a table is **minimized**, it will disappear from the Desktop, but remains available in the Explorer Tree and it can be re-opened from it. If a chart or table is **closed**, it will be deleted with the resulting loss of data.

Designs Table

ID	RID	M	CATEGORY	x	y	f1	f2	min_f1	min_f2
0			ULH	1.01	2.70	1.58	29.77	1.58	29.77
1			ULH	-1.47	1.38	7.86	7.98	7.86	7.98
2			ULH	-0.18	2.51	8.14	20.24	8.14	20.24
3			ULH	-2.83	0.30	5.67	1.72	5.67	1.72
4			ULH	0.56	-2.98	8.94	16.56	8.94	16.56
5			ULH	2.66	1.17	3.36	36.69	3.36	36.69
6			ULH	-1.04	-0.30	48.95	4.32	48.95	4.32
7			ULH	1.79	-2.23	6.55	24.40	6.55	24.40
8			ULH	2.27	-0.99	10.17	27.82	10.17	27.82
9			ULH	-1.97	-1.38	37.90	1.21	37.90	1.21

The Designs Table is the main table containing all designs and variables generated by the optimization run.

Doe Table

ID	M	CATEGORY	x	y
0		RNDDOE	1.45	-0.56
1		RNDDOE	-1.84	-1.05
2		RNDDOE	2.94	-3.10
3		RNDDOE	2.91	2.76
4		RNDDOE	2.81	2.74
5		RNDDOE	-0.65	-0.96
6		RNDDOE	-1.29	0.04
7		RNDDOE	-2.41	1.70
8		RNDDOE	1.00	-2.16
9		RNDDOE	-0.76	-2.26

The DOE Table contains only input values constituting the configuration of designs, which are generated from the *DOE Properties* panel in the workflow by applying one or more DOE techniques.

Table Properties

Designs Table

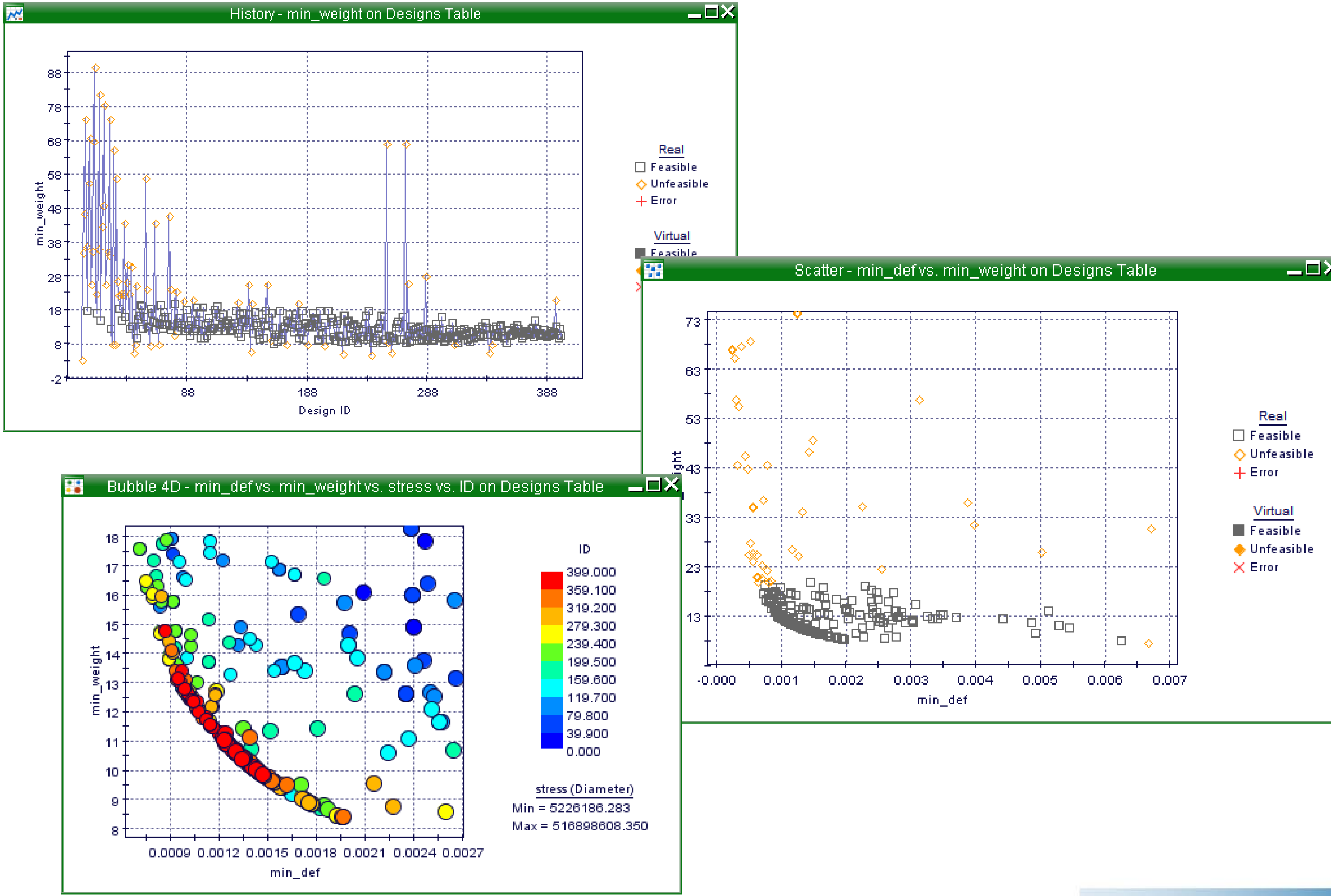
ID	RID	M	CATEGORY	h	r	t	s	v	s_min	v_target	Constrai...
0		☑	☐ SOBOL	1.7500E1	5.5000E0	2.6000E-1	4.9219E2	1.6622E3	4.9219E2	1.3322E3	1.6622E3
1		☐	☐ SOBOL	1.1250E1	3.2500E0	1.4000E-1	1.8114E2	3.7312E2	1.8114E2	4.3120E1	3.7312E2
2		☐	☐ SOBOL	2.3750E1	7.7500E0	3.8000E-1	9.5515E2	4.4792E3	9.5515E2	4.1492E3	4.4792E3
3		☐	☐ SOBOL	1.4375E1	6.6250E0	2.0000E-1	5.7467E2	1.9811E3	5.7467E2	1.6511E3	1.9811E3
4		☐	☐ SOBOL	2.6875E1	2.1250E0	4.4000E-1	2.0768E2	3.8106E2	2.0768E2	5.1062E1	3.8106E2
5		☐	☐ SOBOL	8.1250E0	8.8750E0	8.0000E-2	7.2107E2	☐	7.2107E2	☐	☐

- ✓ **category**: usually DOE or Scheduler that generated the design (categories are customizable)
- ✓ **unfeasible design** (yellow rows): violating one or more constraints (the violated constraint cell is colored orange)
- ✓ **feasible design** (white rows)
- ✓ **error design** (red rows): one or more output values are missing because an error occurred during the run
- ✓ **virtual design** (blue font): computed using an RSM function
- ✓ **real design** (black font): computed using the real solver
- ✓ **selected design**: selected by clicking on its ID
- ✓ **marked design**: selected checkbox in M (third) column

Operations on tables can be performed from their context menu

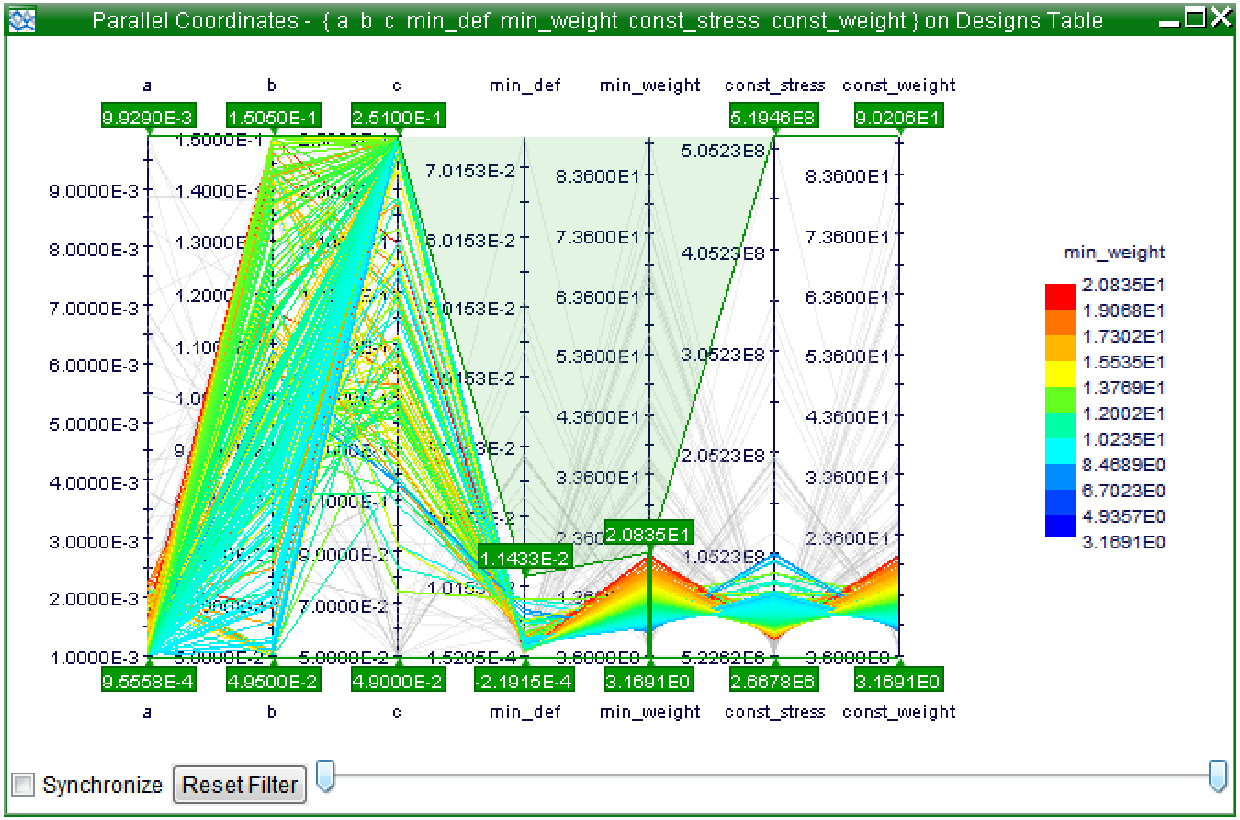
History, Scatter and Bubble Charts

modeFRONTIER



Parallel Coordinates Chart

- ✓ Simultaneous visualization of all variables
- ✓ Filtering of designs according to values (high/low)
- ✓ useful for spotting patterns in variable behavior and correlations
- ✓ the filter can be dynamically applied to other charts

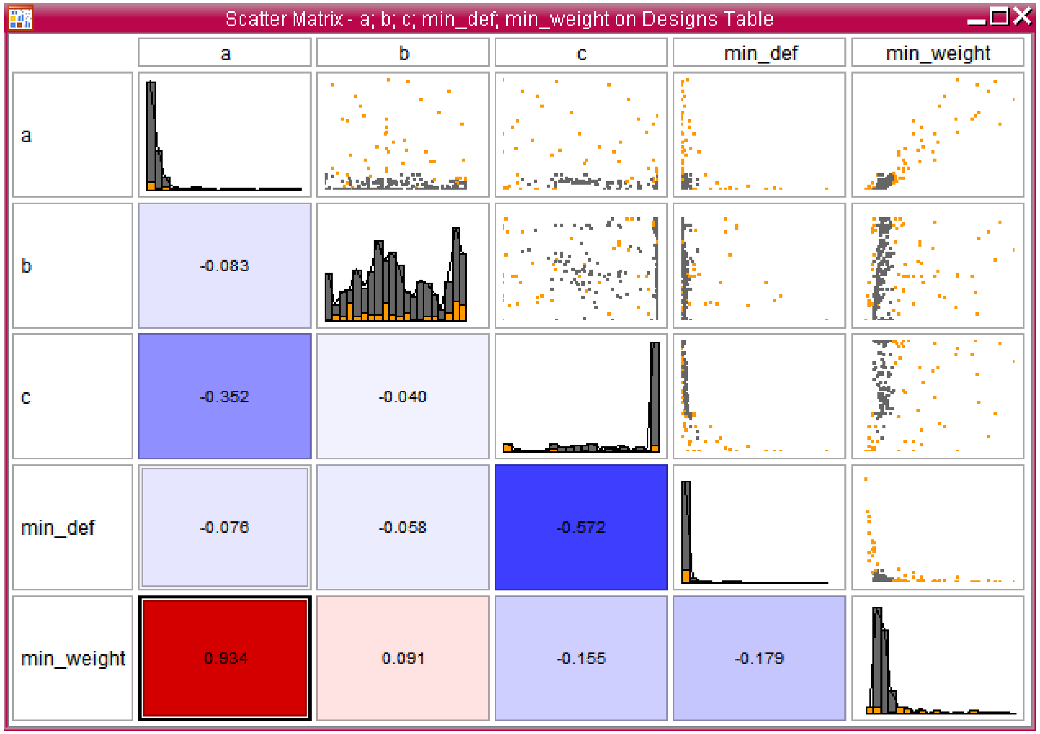


Scatter Matrix Chart

Scatter Matrix Chart is a combination of three charts:

- ✓ Correlation Matrix Chart
- ✓ Scatter Chart
- ✓ Probability Density Function Chart

It shows the measure of association between pairs of selected variables computed using the Pearson correlation and the distribution of each single variable.

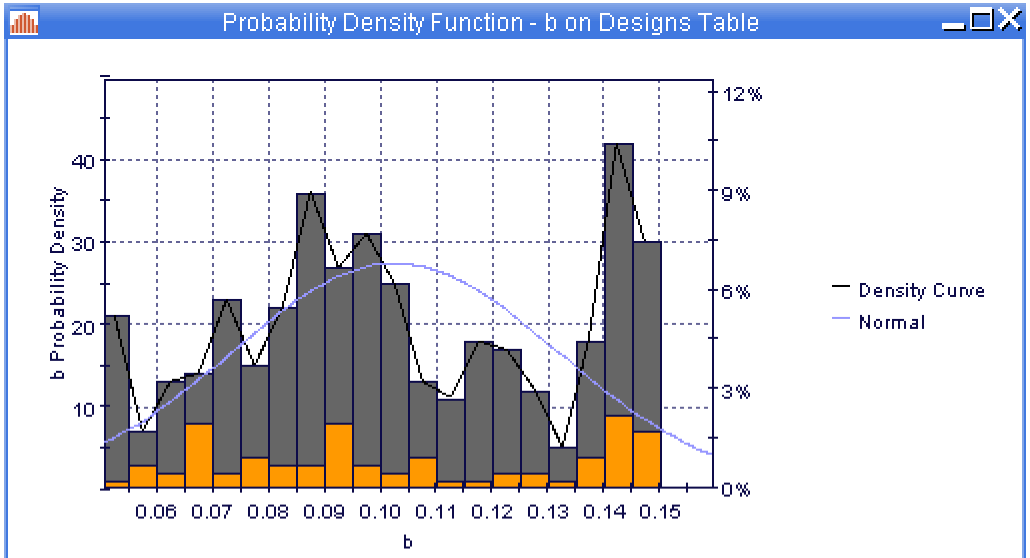


Probability Density Function Chart

The **Probability Density Function Chart** graphically summarizes the distribution of the given variable by splitting its range of values into equally sized classes and counting the number of designs falling into each class.

This chart can help users answer the following questions:

- ✓ How is the dataset distributed over a given range?
- ✓ Where are data located and?
- ✓ Is the data symmetric or skewed?



N. of Designs	400	Std. Deviation	0.029
N. of Classes	20	Skewness	0.084
Min	0.050	Kurtosis	1.951
Max	0.150	Confidence Int. 95%	(0.100,0.106)
Arithmetic Mean	0.103	Confidence Int. 99%	(0.099,0.106)
Variance	0.001	Variation Coeff.	0.283

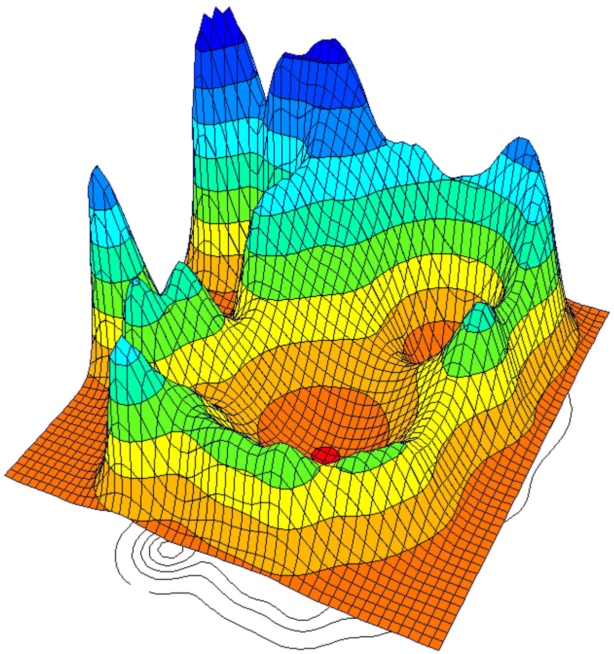
modeFRONTIER

Introduction to Response Surfaces

Response Surface Models

Response Surface Models (RSMs) or metamodels are statistical and numerical models that approximate the input/output behavior of the system under investigation. Starting from a dataset of real designs, an RSM algorithm guesses the value of the unknown function.

RSM-based, or virtual optimization is a valid strategy which serves as a surrogate for heavy simulation processes, allowing engineers to fast-run the classi optimization process



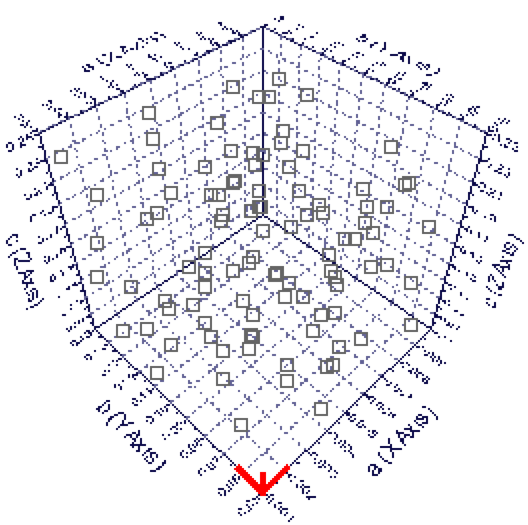
Main Advantages

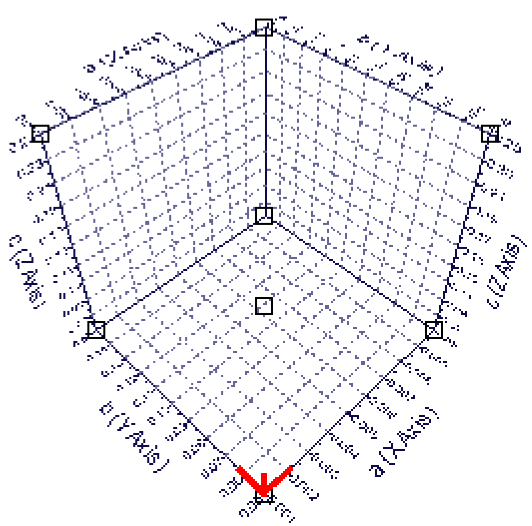
- ✓ perform thousands of design evaluations in a short time
- ✓ accelerate the optimization step
- ✓ use small amounts of data efficiently
- ✓ smart exploitation of available computational resources

How to generate a good DOE for RSM training

- ✓ In general, the higher the number of starting designs, and more accurate the RSM training (the exceptions are the outliers and overfitting):

- ✓ uniformity of DOE should be as high as possible (e.g. Uniform Latin Hypercube; factorial DOEs with few levels should be avoided)



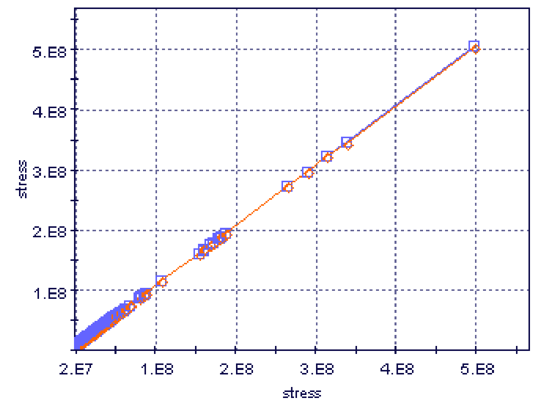


Types of RSM algorithms

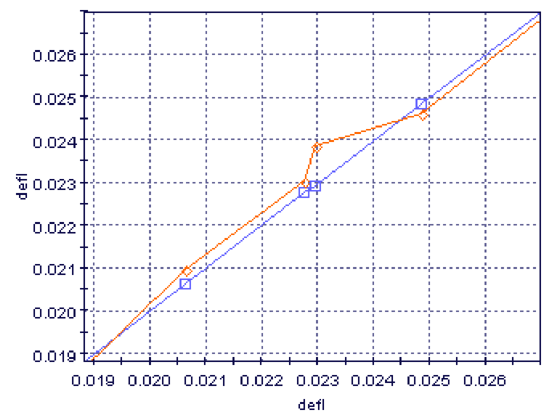
modeFRONTIER

Types of RSMs:

- ✓ **Interpolating**: pass exactly through the training points; unknown coefficients are identified exactly from the system of equations made by this condition (e.g. Shepard-K-nearest, Kriging, Radial Basis Functions)



- ✓ **Approximating**: do not pass exactly through the training points; unknown coefficients are identified by minimizing the extrapolation error on training points (e.g. Polynomial SVD, Gaussian Processes, Neural Networks, Evolutionary Design)



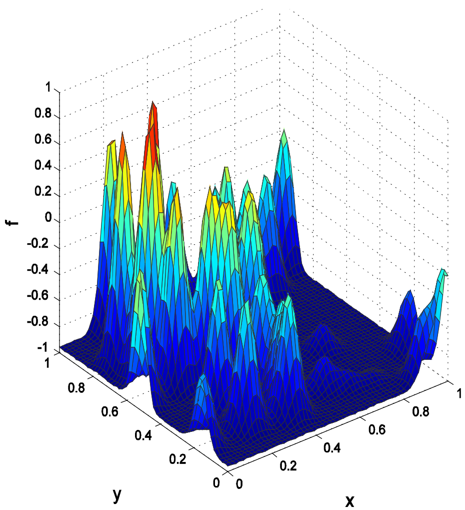
How to select the right RSM algorithm?

- ✓ If the problem is linear and with little noise (few peaks in the function), try the simplest RSM, like Polynomial SVD or Gaussian Processes
- ✓ General recommendation: use simple meta-models first (e.g. low order polynomials), and Kriging and NN for more complex responses
- ✓ In most cases NN, RBG and Kriging seem to be quite efficient

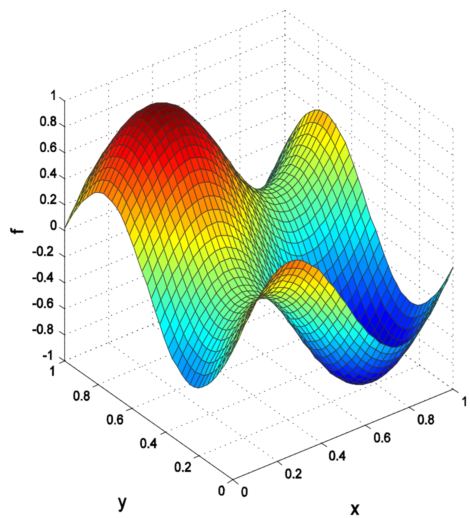
Overfitting and underfitting

The effect of the prediction error increasing due to a too simple model is called **underfitting** whereas the effect of the increased prediction error due to a too complex model is called **overfitting** or overtraining.

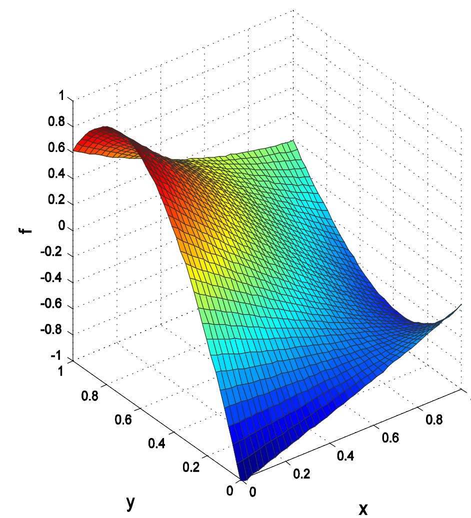
For data sets which are noisy and limited in size a simple calibration model is preferred to prevent the overfitting. For big data sets, which contain only little noise, the best model is more complex resulting in an overall smaller prediction error.



Overfitting



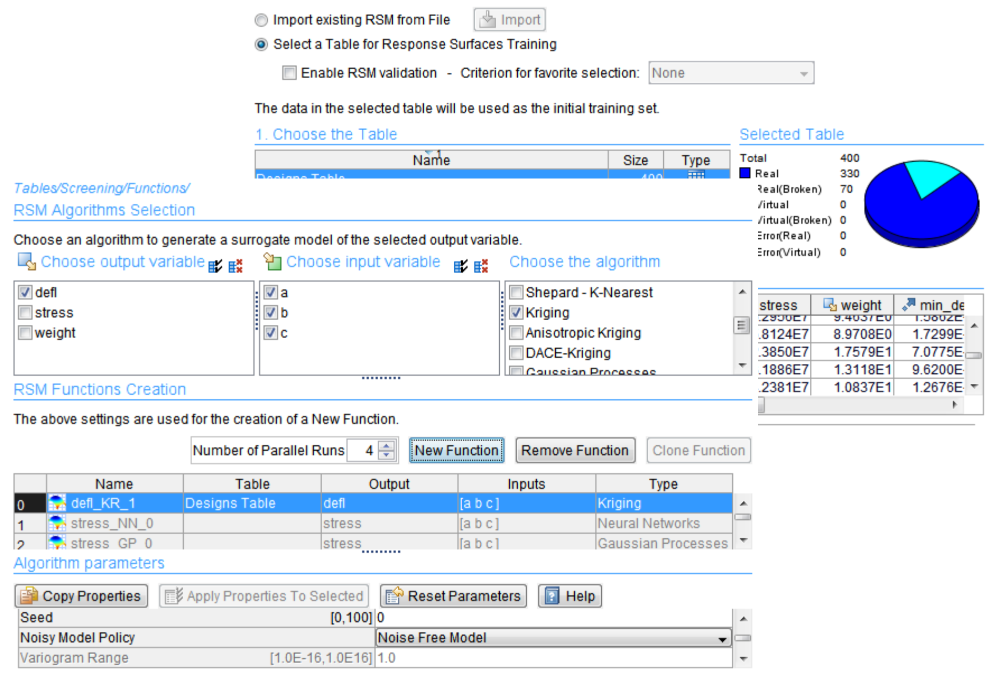
Correct fit



Underfitting

RSM Training in modeFRONTIER done by means of a step-by-step wizard, where the user must choose the training dataset, inputs and outputs, and RSM algorithm(s), and configure the latter.

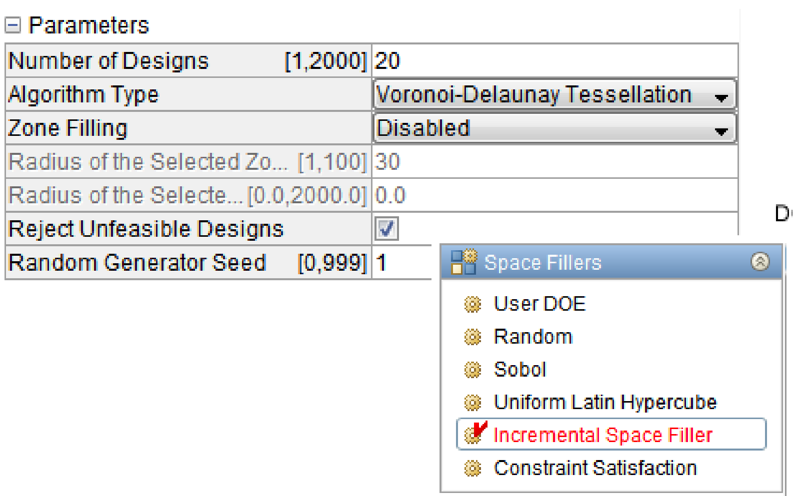
Data scaling is highly recommended!

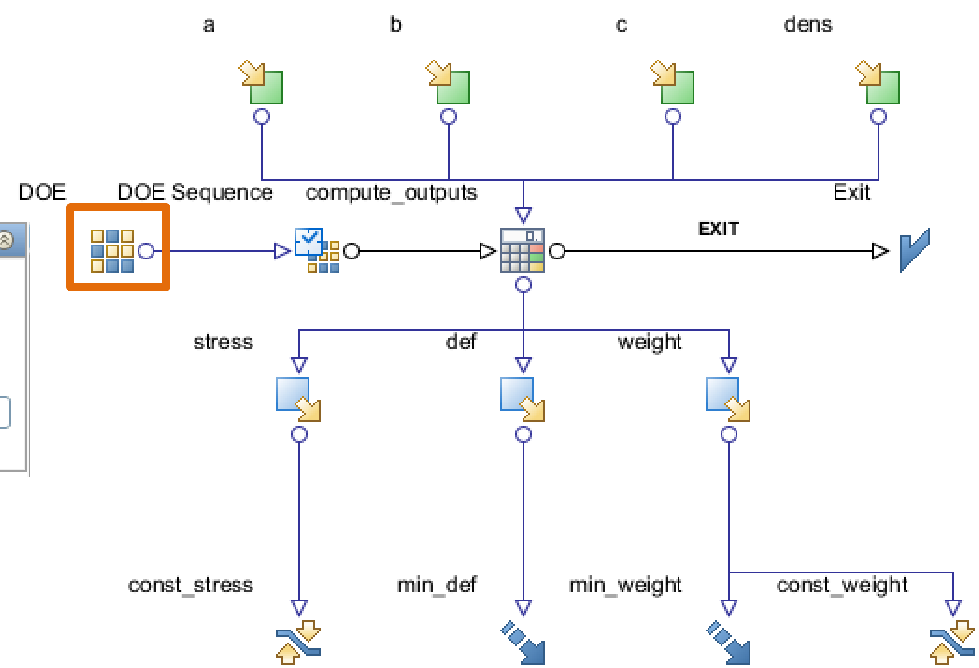


1. **RSMs** are **trained** from an available database of real designs and validated one against another
2. The best model is used to **compute** the outputs of the system; this process is called **virtual optimization**
3. The best designs obtained through virtual optimization are then **evaluated by the real solver**

Let us presume that we are not able to run many design evaluations and that the optimization must be completed by running not more than **40 designs**.

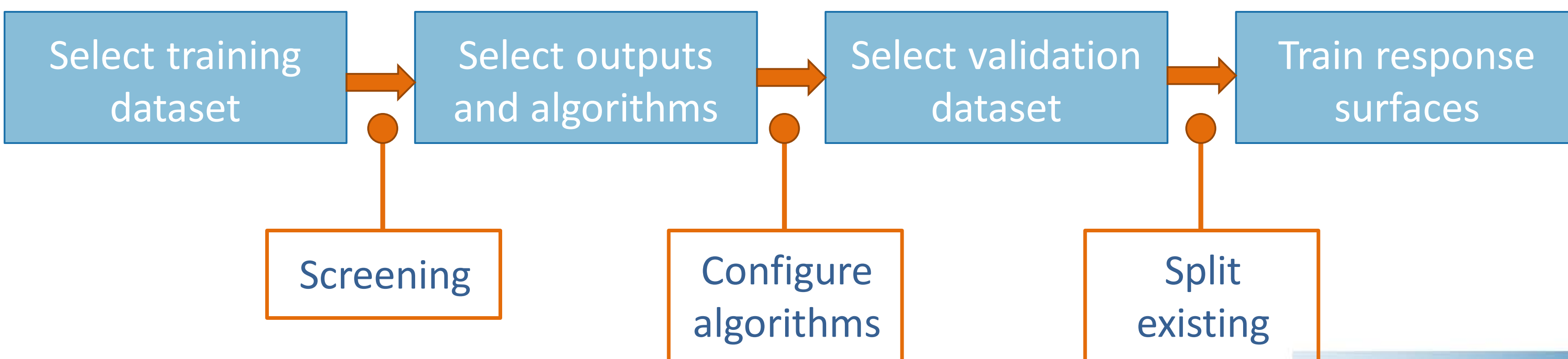
1. Add 20 DOE with **ULH** + 20 with **ISF** (use Voronoi-Delaunay Tesselation as algorithm) in the DOE Node Properties
2. Select **DOE Sequence** in the Scheduler Node – no optimization is performed, only the outputs are calculated from the given input combinations





Hands on RSM Training

1. Select training dataset and enable RSM Validation
2. Perform screening analysis (optional step)
3. Select outputs on which RSM training should be performed, inputs and algorithms.
   Add the New Functions. In the lower part of the panel you can configure algorithm settings for each function.
4. Split training table – a portion of the selected dataset is used for validation
   No point using exactly the same data for training and validation of interpolating RSMs
5. Perform RSM training
6. Check validation table



Choosing the best RSM

Comparison of RSMs trained on the same output by means of performance coefficients:

- ✓ Mean Absolute Error
- ✓ Mean Relative Error
- ✓ Mean Normalized Error
- ✓ R-Squared (R^2 of 1 indicates a perfect fit)
- ✓ AIC (Akaike Information Criterion) - only when comparing models trained with the same algorithm (the best model has the lowest AIC)

RSM Validation - stress

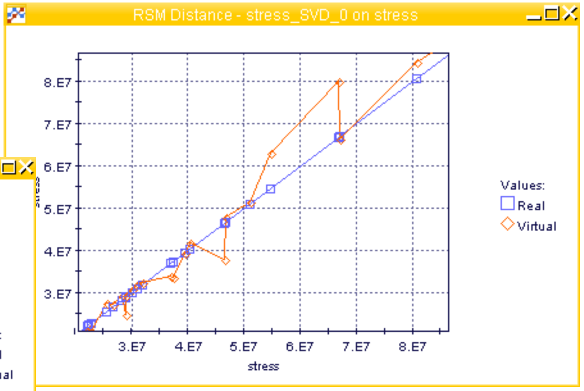
Name	Mean Absol...	Mean Relati...	Mean Norm...	R-squared	AIC
stress_KR_0	6.38E6	4.65E-1	8.40E-2	9.44E-1	2.02E2
stress_NN_0	4.66E5	2.03E-2	6.13E-3	1.00E0	1.49E2
stress_SVD_0	5.50E6	4.73E-1	7.24E-2	9.62E-1	1.65E2

Dark green cells indicate best values, whereas light green to white cells indicate worst values.

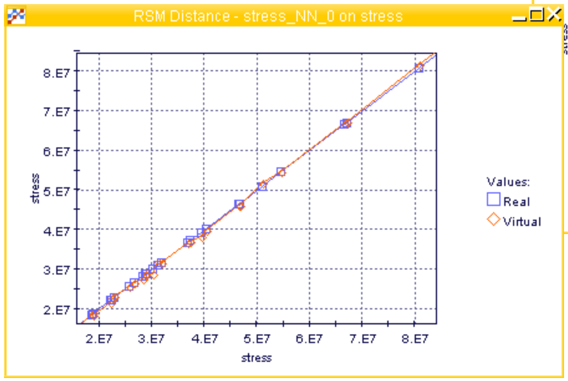
The comparison of RSMs should also be done using charts!

Choosing the best RSM - Charts

RSM Distance Chart – shows the distance between real designs and responses predicted by the RSM algorithm

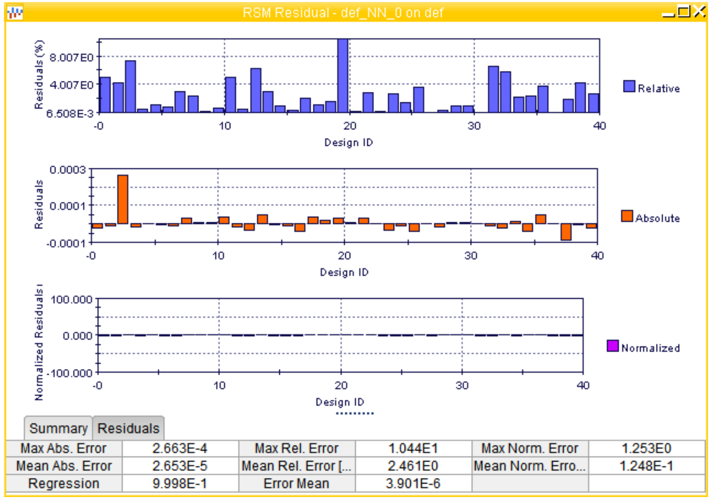


Note:
do not plot an interpolating RSM using the training table only because the error will be 0



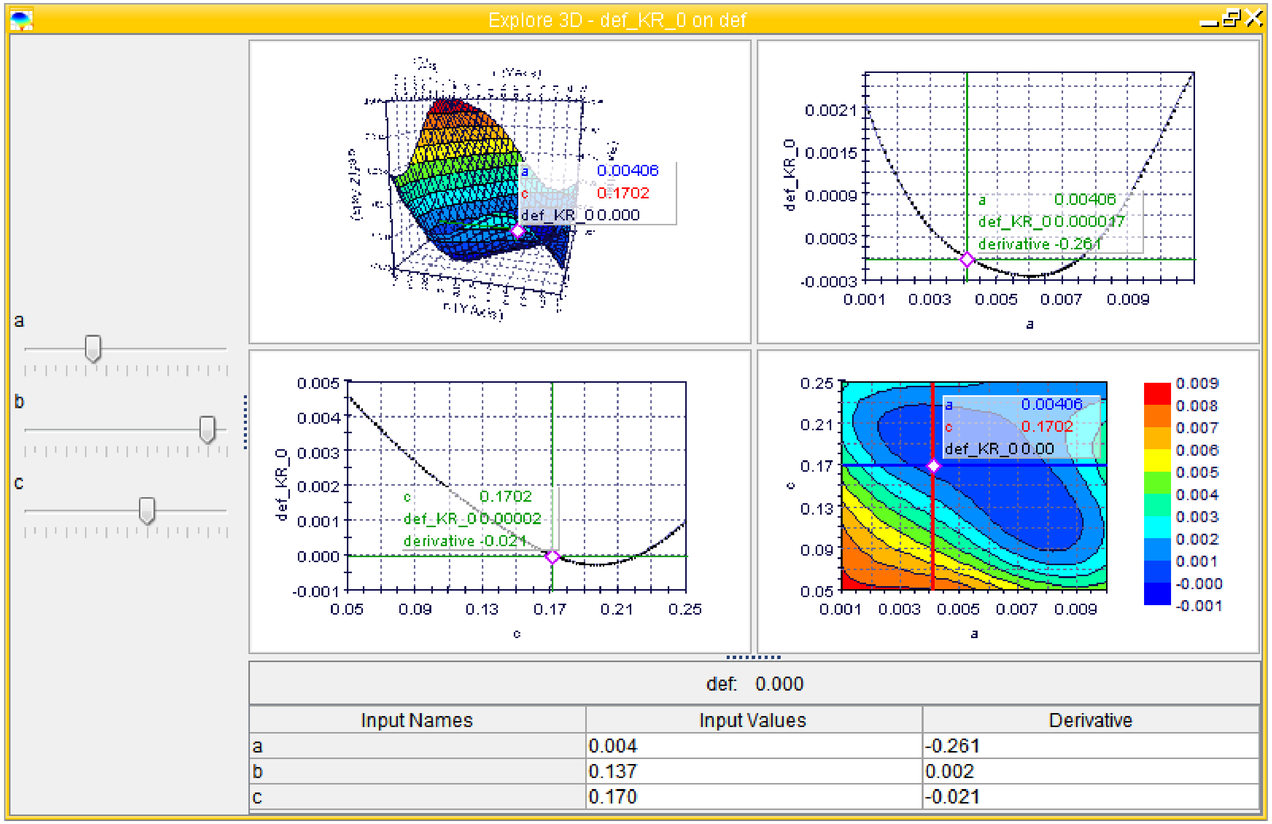
RSM Residual Chart – shows the differences (**residuals**) between real designs and responses predicted by the RSM algorithm in form of a histogram

- ✓ **absolute residual**: actual difference
- ✓ **relative residual**: ratio between absolute residual and real design value (%)
- ✓ **normalized residual**: % of absolute residual in output range



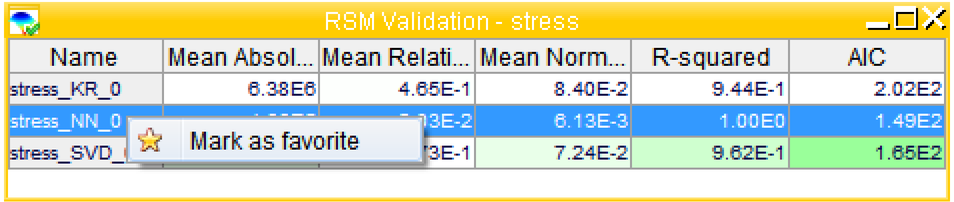
RSM Explore 3D Chart

2x2 chart formed by an **RSM 3D chart**, two **RSM Function** plots and an **RSM Contour** chart.



Values of virtual designs are shown by means of a **Probe**. The coordinates of each designs on the chart are changed using the input **sliders**.

Performing virtual optimization



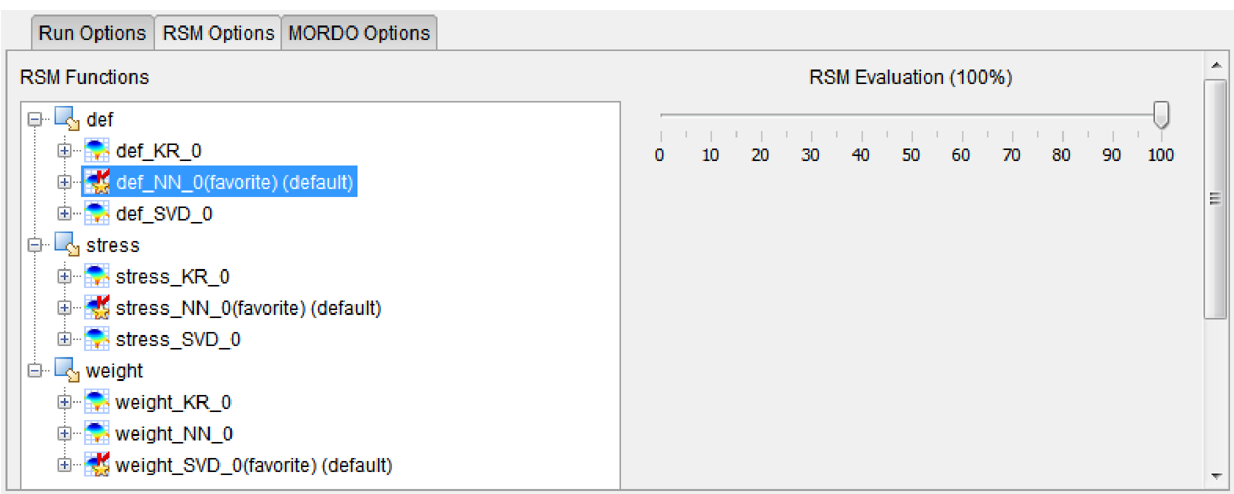
RSM Validation - stress

Name	Mean Absol...	Mean Relati...	Mean Norm...	R-squared	AIC
stress_KR_0	6.38E6	4.65E-1	8.40E-2	9.44E-1	2.02E2
stress_NN_0	[illegible]	…3E-2	6.13E-3	1.00E0	1.49E2
stress_SVD_…	[illegible]	…3E-1	7.24E-2	9.62E-1	1.65E2

The best RSM for each output can be marked as **Favorite** – this RSM will be used for virtual optimization

Virtual or RSM-based optimization is a run performed using RSM functions instead of the real solver to evaluate the their fitness. It is very **fast** so a large number of designs can be computed.

Return to the Workflow and open **Scheduler Properties**: in the **RSM Options** tab move the slider to 100%.

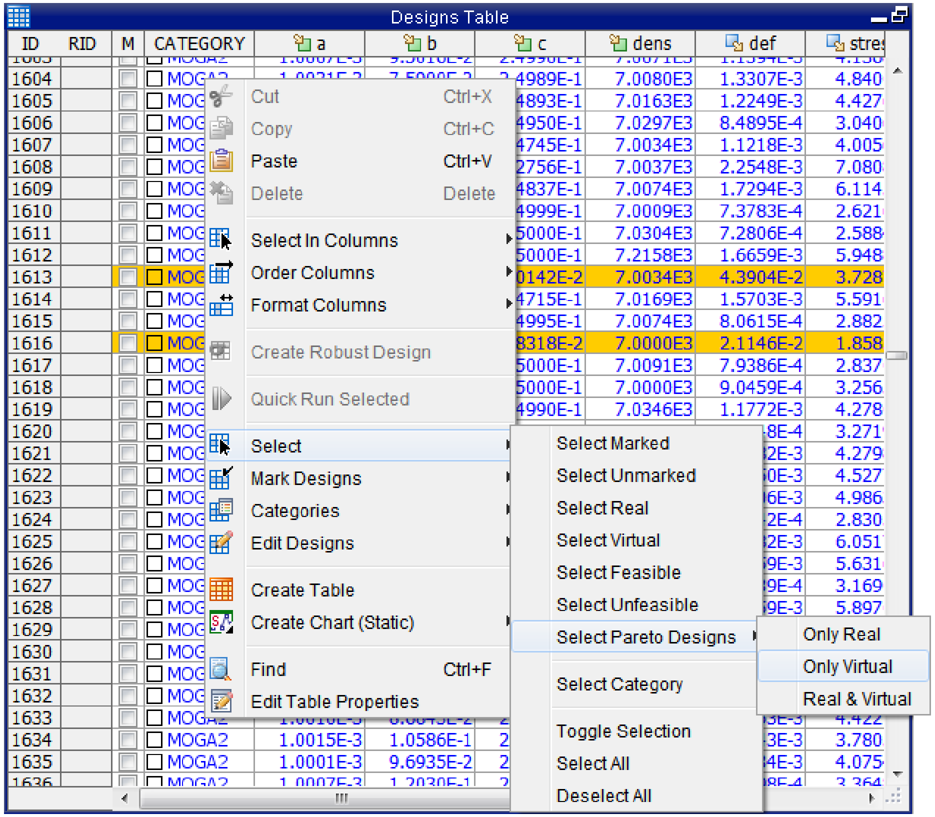


Choose the MOGA-II algorithm and set the Number of Generations to 10.

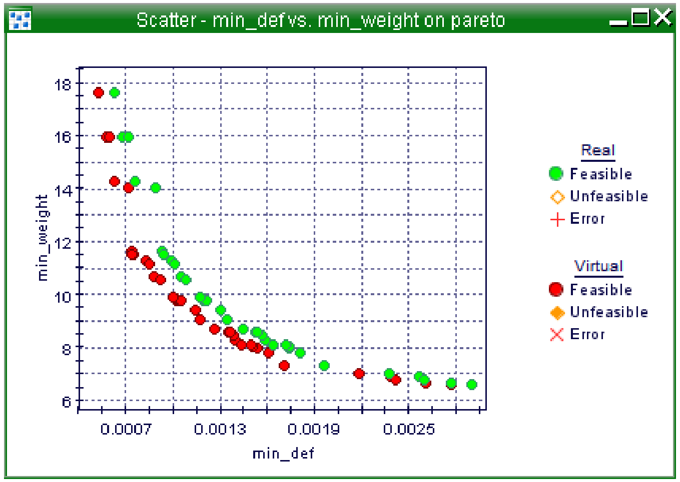
Validating virtual data

The **virtual designs** are added to the **Designs Table** (colored in blue).

Validation of virtual data refers to the computation of output values by using the real solver. The real and virtual output values should then be compared.



Select Pareto virtual designs and perform the **Quick Run** to calculate the real output values.
They can be compared, for example, by means of a Scatter Chart.



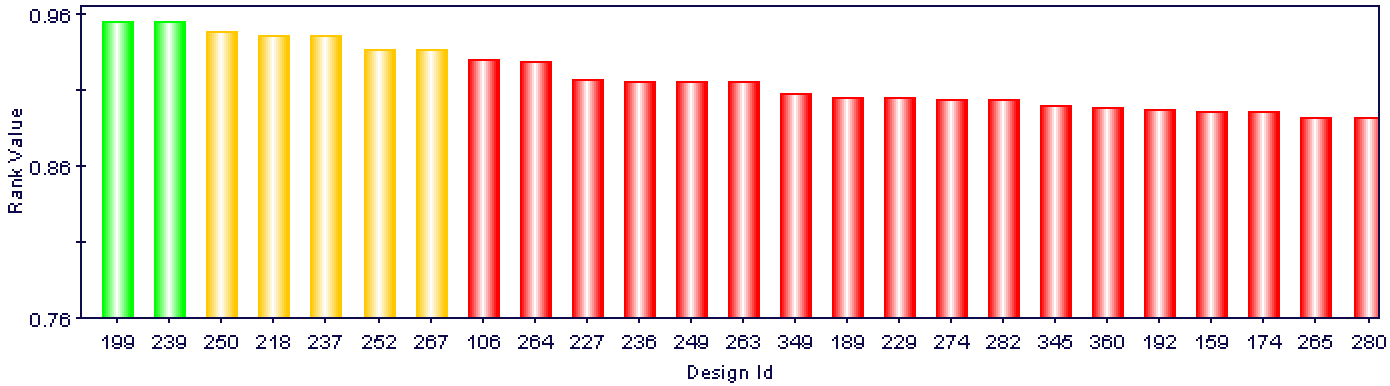
modeFRONTIER

Multi-Criteria Decision Making

Why MCDM?

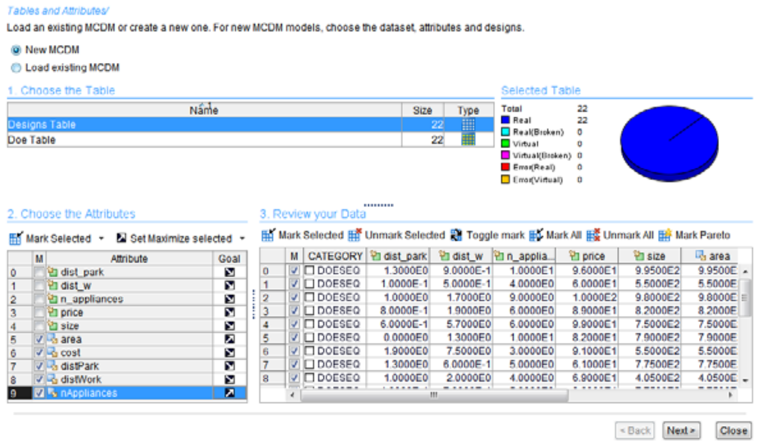
Multi Criteria Decision Making (MCDM) refers to the solving of decision and planning problems involving multiple and generally conflicting requirements. MCDM enables the decision maker to perform a structured analysis and choose one reasonable alternative from among a set of available ones. Each alternative is evaluated in relation to the others in terms of a pre-specified decision rule or set of rules used to rank the available alternatives.

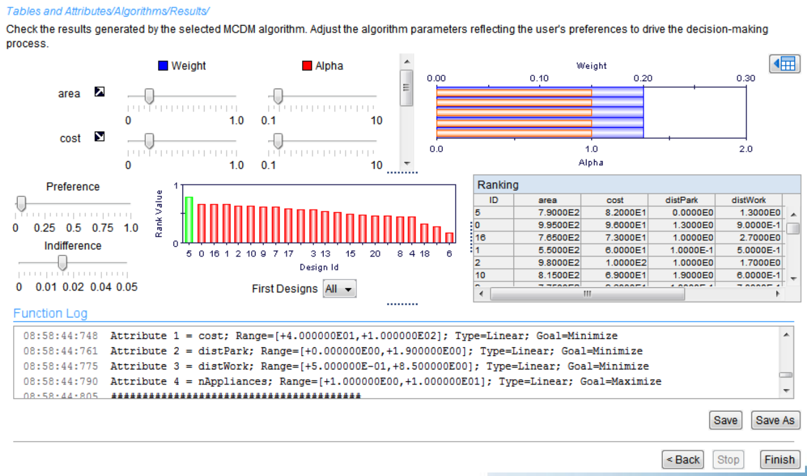
The decision maker is required to indicate their own preferences and the results produced by modeFRONTIER enhance their understanding of the data.



MCDM Wizard

1. Select the **attributes** you wish to consider for taking your decision and indicate whether their values should be maximized or minimized
2. Select data (designs) to be parsed and ranked by the MCDM algorithm
3. Choose an algorithm
4. Adjust weights according to your preferences, and the Preference and Indifference Margins using sliders





mode FRONTIER

Thank you for your attention!