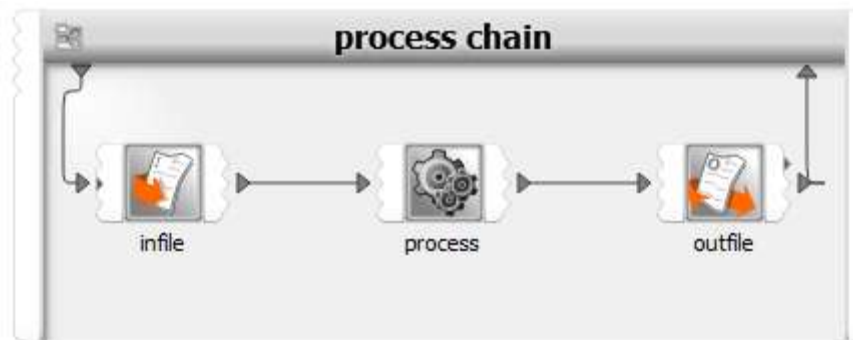
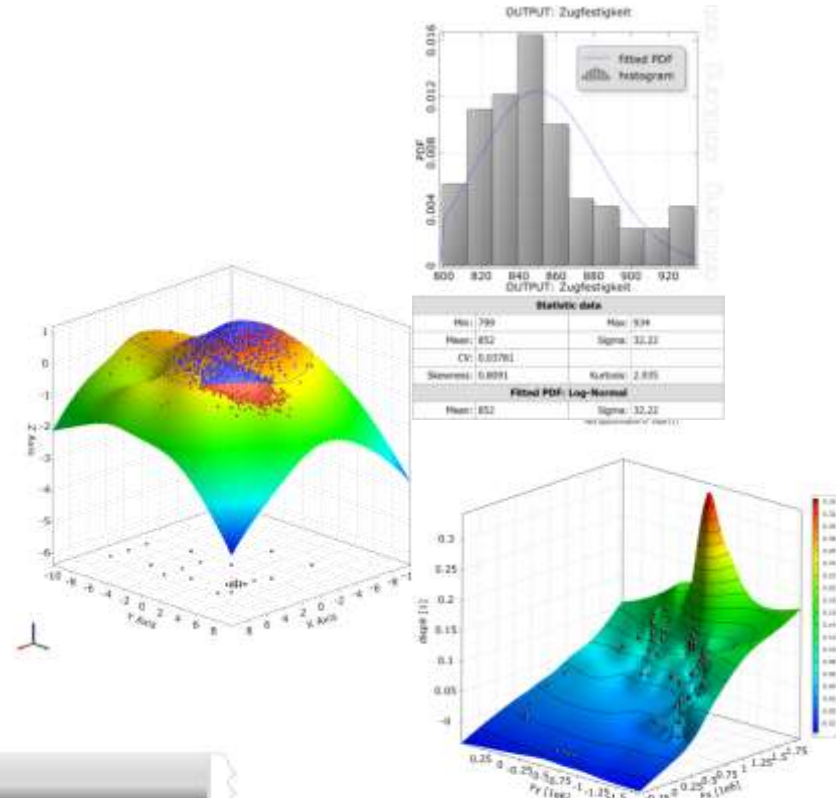


# optiSLang v4



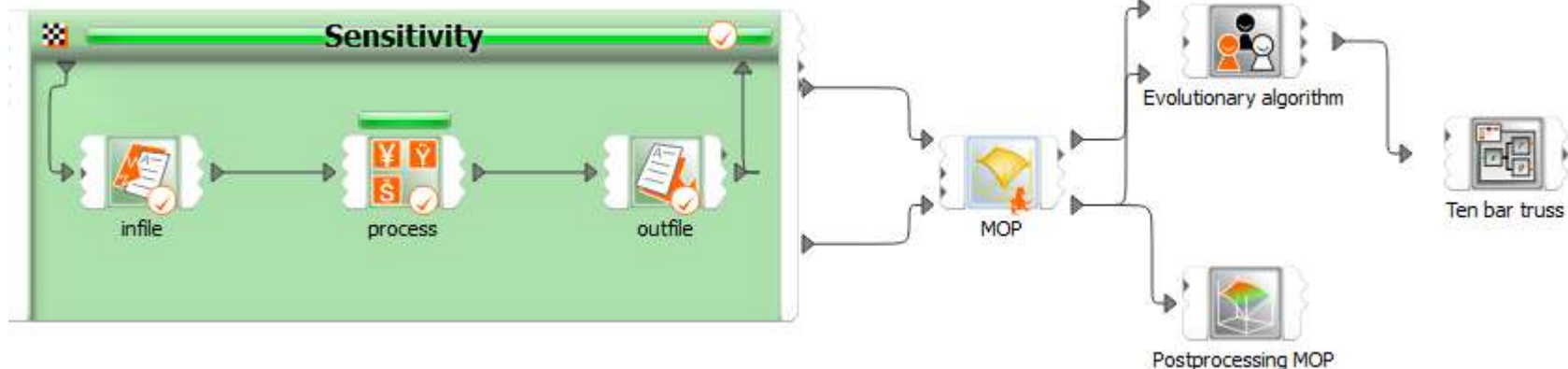
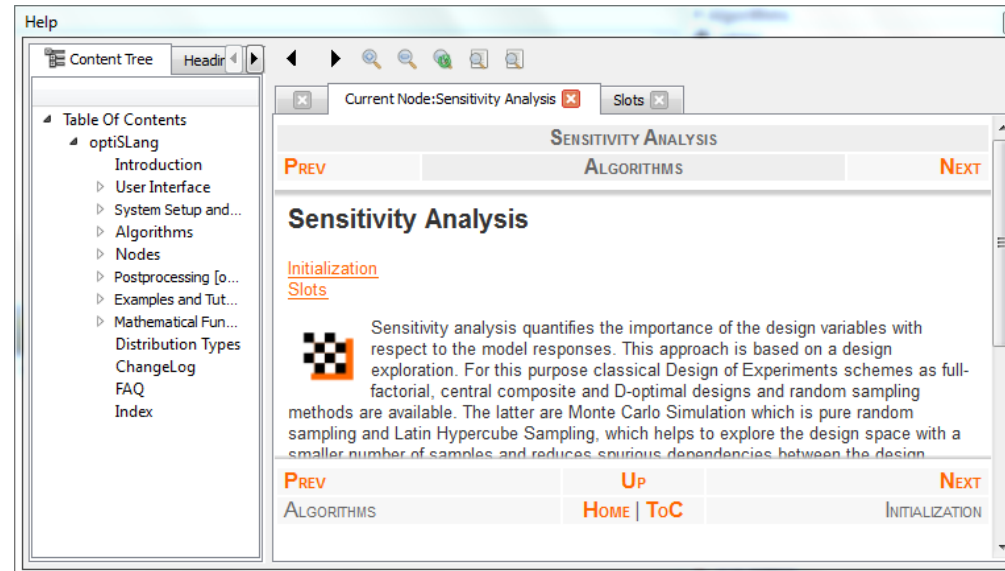
## v3 Compatibility

- Support v3 problem definition
  - Load or
  - Double - click .pro
- Postprocessing
  - v4.0 uses v3 postprocessing
  - Double - click .bin



# Documentation

- **Tutorials**
- **Examples**
- **Supportmail**
- **Method documentation**
- **Context sensitive help**
  - **Tabs**
  - **Search**



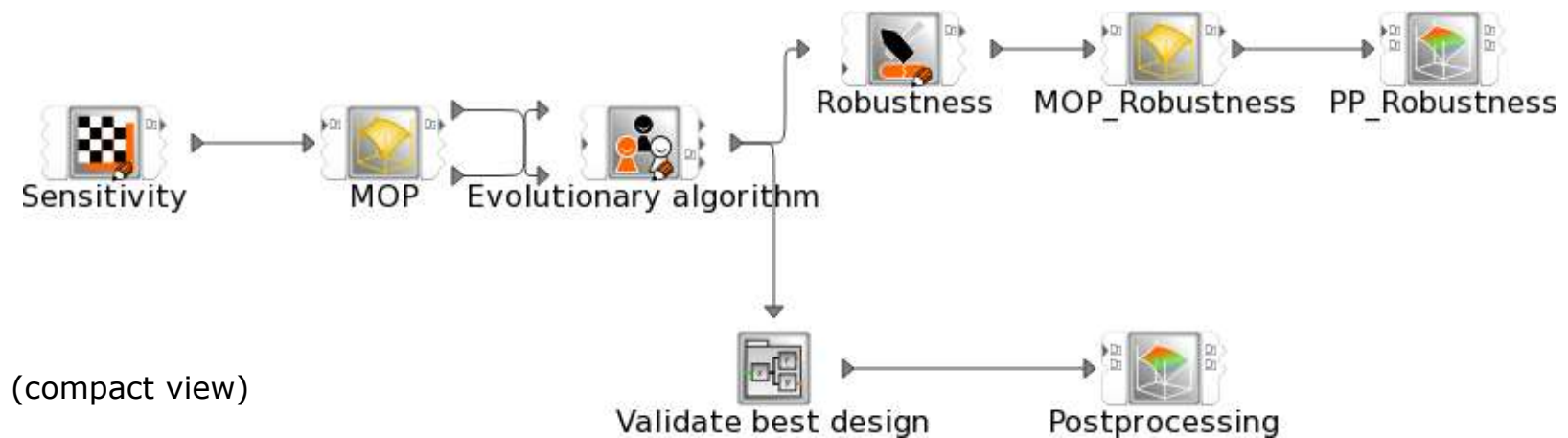
# Wizards

- **Fast and easy creation of:**
  - **Process chain**
  - **Sensitivity analysis**
  - **Optimization task**
  - **Robustness evaluation**
- **Supported by:**
  - **Principle of wizards**
  - **Defaults**
  - **Decision help**



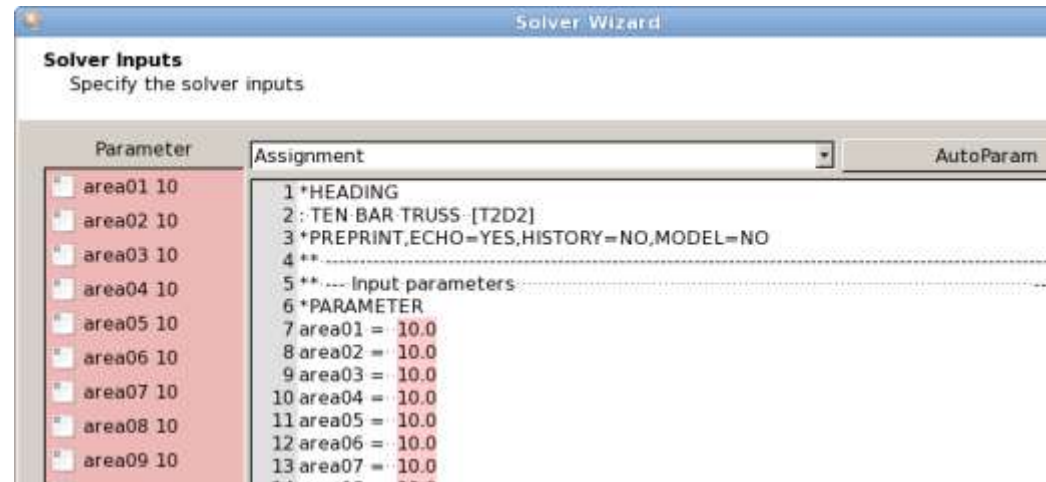
## Wizard based flow creation

- Use templates
- Drag & Drop
- Automatic connect (Parameter, Designs, ...)
- Algorithms are nodes with inputs and outputs



# Input parametrization

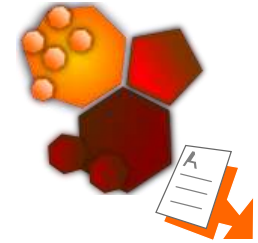
- **Automatic detection**  
*comfortable for a lot of parameters*
- **Parameter table**  
*for fast and easy modification*




	Name	Parameter type	Reference value	Constant	Resolution	Range	Range plot	PDF	Type	Mean	Std. Dev.
1	area01	Det+Stoch	10	<input type="checkbox"/>	Continuous	0.1 20			NORMAL	10	0.288675
2	area02	Det+Stoch	10	<input type="checkbox"/>	Continuous	0.1 20			NORMAL	10	0.288675
3	area03	Det+Stoch	10	<input type="checkbox"/>	Continuous	0.1 20			NORMAL	10	0.288675
4	area04	Det+Stoch	10	<input type="checkbox"/>	Continuous	0.1 20			NORMAL	10	0.288675
5	area05	Det+Stoch	10	<input type="checkbox"/>	Continuous	0.1 20			NORMAL	10	0.288675
6	area06	Det+Stoch	10	<input type="checkbox"/>	Continuous	0.1 20			NORMAL	10	0.288675
7	area07	Det+Stoch	10	<input type="checkbox"/>	Continuous	0.1 20			NORMAL	10	0.288675

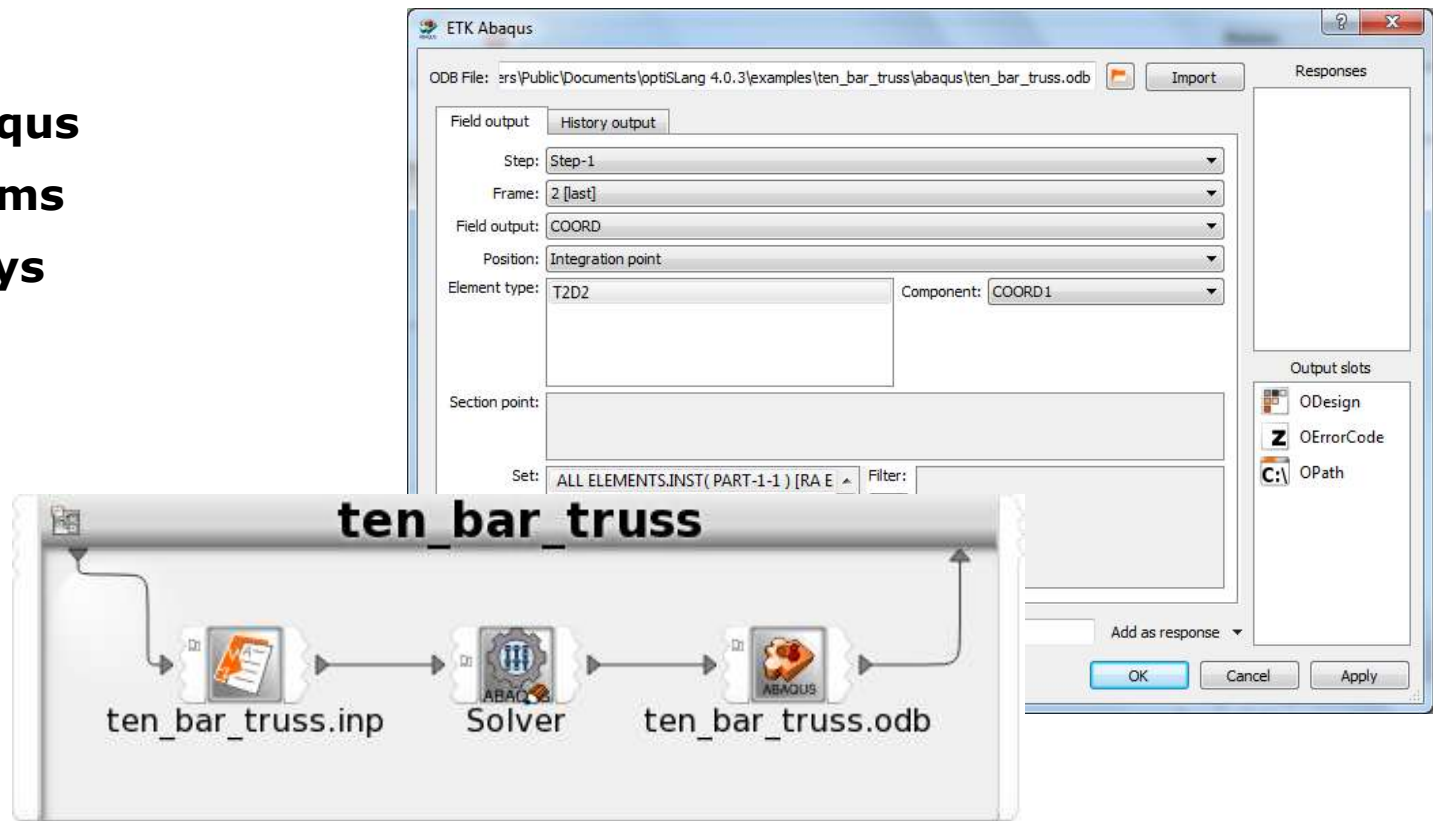
# Output extraction with ETK

**Comfortable extraction of known output file formats  
To be integrated in process chain**



**Nodes for:**

- **Abaqus**
- **Adams**
- **Ansys**



# Simulation X

**SIMULATION X**

Powered by ITI

## Solver specific dialog and execution

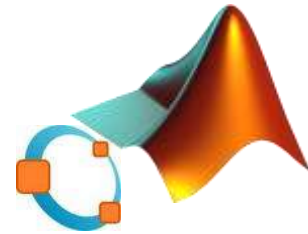
The screenshot shows the SimulationX software interface. The main window is titled "SimulationX" and contains several panels:

- Parameter:** A list of parameters on the left, including "mass.m 1" and "spring.k 20".
- Input slots:** A list of input slots on the bottom left, including "IBaseDir", "IDesign", and "IErrorCode".
- Responses:** A large empty box on the right for simulation results.
- Output slots:** A list of output slots on the bottom right, including "max\_disp.y", "ODesign", "OErrorCode", and "omega.y".
- Table:** A central table with columns "Name", "Value", "Unit", and "Description". It lists simulation parameters and results.

The table content is as follows:

Name	Value	Unit	Description
<b>Parameters</b>			
tStart	0	s	Startzeit
tStop	10	s	Stoppzeit
termCond	0	-	Simulation beenden...
gravity	9.80665	m/s <sup>2</sup>	Gravitationsbeschle...
pAtm	1.01325	bar	Atmosphärendruck
TAtm	20	°C	Atmosphärentempe...
protOn	1	-	Protokollierung ein...
traceOn	1	-	Tracing eingeschaltet
<b>omega</b>			
<b>Results</b>			
y	[1:438]	-	Signalausgang
<b>eventSH2</b>			
<b>Parameters</b>			
direction	0	-	Richtung
a	0.5	-	Grenzwert
y0	0	-	Anfangswert
<b>Results</b>			
y	[1:438]	-	Signalausgang





# Matlab

## Solver specific dialog and execution

Matlab

Parameter

k 20

m 1

Input slots

C:\ IBaseDir

IDesign

Z IErrorCode

N IMaxParallel

otiSLang 4.0.3\examples\oscillator\matlab\oscillator.m

Open

	Name	Type	Value
1	D	SCALAR	0.02
2	Ekin	SCALAR	10
3	envelope	VECTOR	[100:1]
4	indicator	VECTOR	[100:1]
5	k	SCALAR	20
6	m	SCALAR	1
7	num_steps	SCALAR	100
8	omega0	SCALAR	4.47214
9	omega_damped	SCALAR	4.47124
10	sin_values	VECTOR	[100:1]

% Inputs (define reference variables)  
 if ( ~exist('m','var') ) m = 1; end;  
 if ( ~exist('k','var') ) k = 20; end;  
 if ( ~exist('D','var') ) D = 0.02; end;  
 if ( ~exist('Ekin','var') ) Ekin = 10; end;

% Scalar values

Responses

omega\_damped 4.4712

x\_max 0.623..

Output slots

ODesign

Z OErrorCode

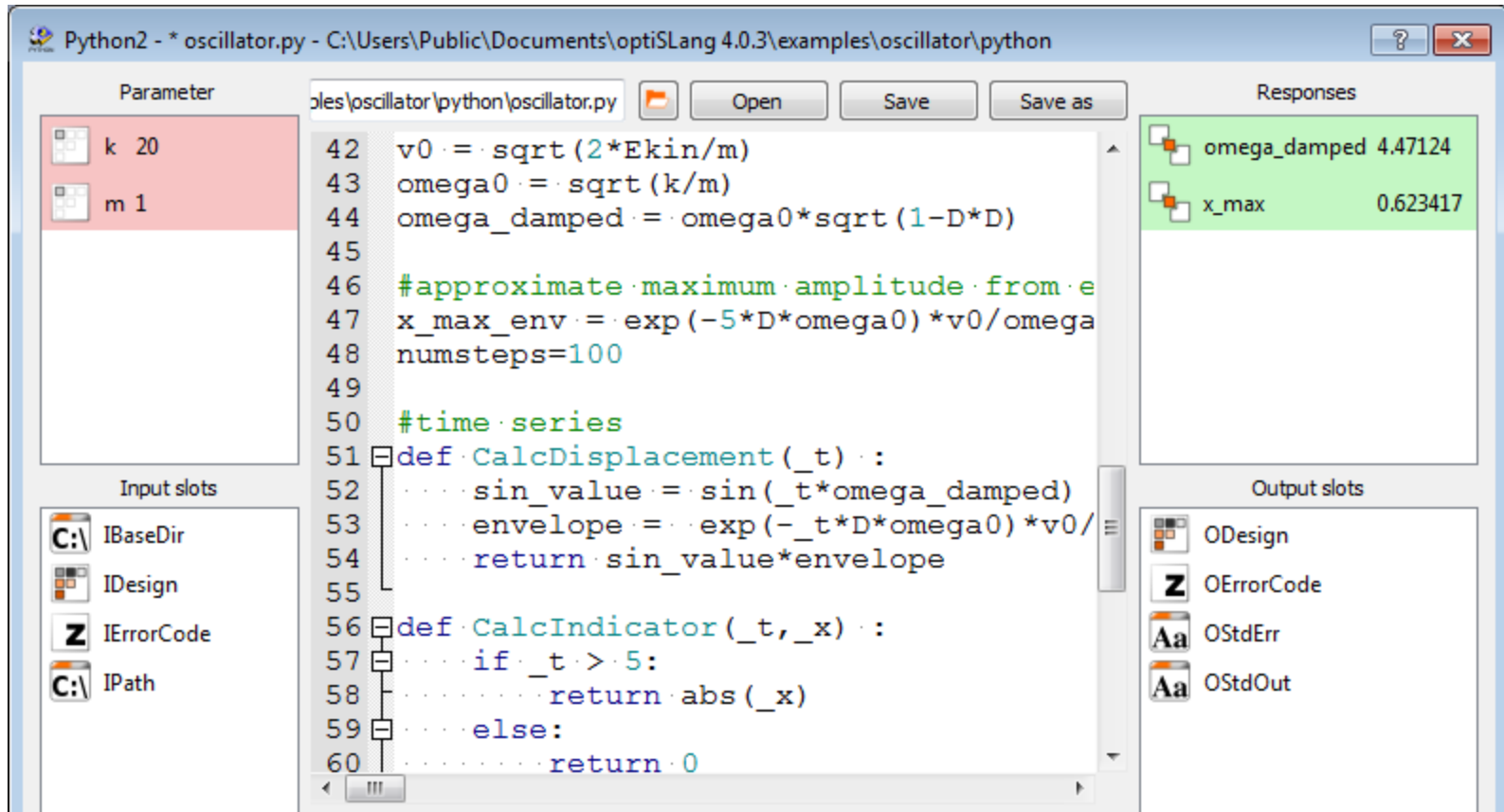
N OMaxParallel

C:\ OPath



# Python

## Solver specific dialog and execution





# Excel

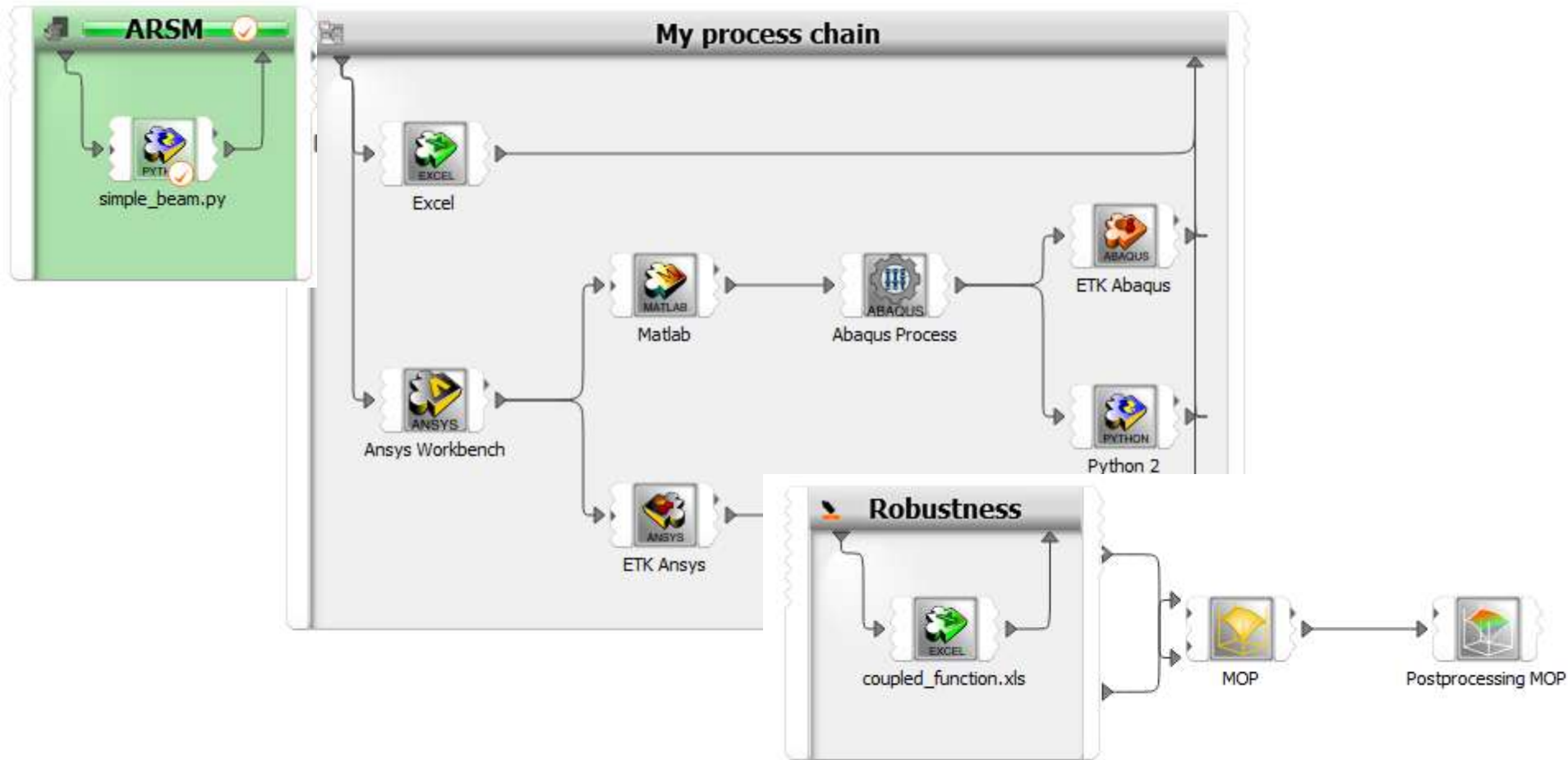
## Solver specific dialog and execution

The screenshot shows the Excel Solver dialog box for the file `\examples\oscillator\excel\oscillator.xls`. The **Parameter** section on the left lists `k 20` and `m 1`. The **Input slots** section lists `C:\ IBaseDir`, `IDesign`, and `IErrCode`. The **Responses** section on the right lists `omega_damped` and `x_max`. The **Output slots** section lists `ODesign`, `OErrorCode`, and `OPath`. The central table displays the following data:

	A	B	C
1	Variable values		
2	k	20	
3	m	1	
4			
5	omega_0	4.47214	
6	omega_damped	4.47124	
7			
8	v0	4.47214	
9	x_max_env	0.639535	
10			
11			
12	x_max	0.623417	

# Integrations

Appear as nodes with inputs and outputs



## Command Line Interface (CLI)

- **Create or modify a project**
  - **In GUI**
  - **With python script**
- **Run the project**
  - **In batch**
  - **No graphical environment needed**

*optislang --batch myproject.opf*

# Python modules and C++ libraries

## Use

- **DOE**
- **Robustness evaluation**
- **Optimization algorithms**
- **MOP / MOP – Solver**

## In

- **External Code (Matlab, Simplorer, ...)**
- **Script / own application**
- **Customized Application**

## Through

- **Import of dynardo Python modules**
- **Use of dynardo C++ libraries (.dll, .so)**

## Release

### optiSLang inside Ansys Workbench

**ACUM**

**26. 10. 2012**

### optiSLang v4

**WOST 9.0**

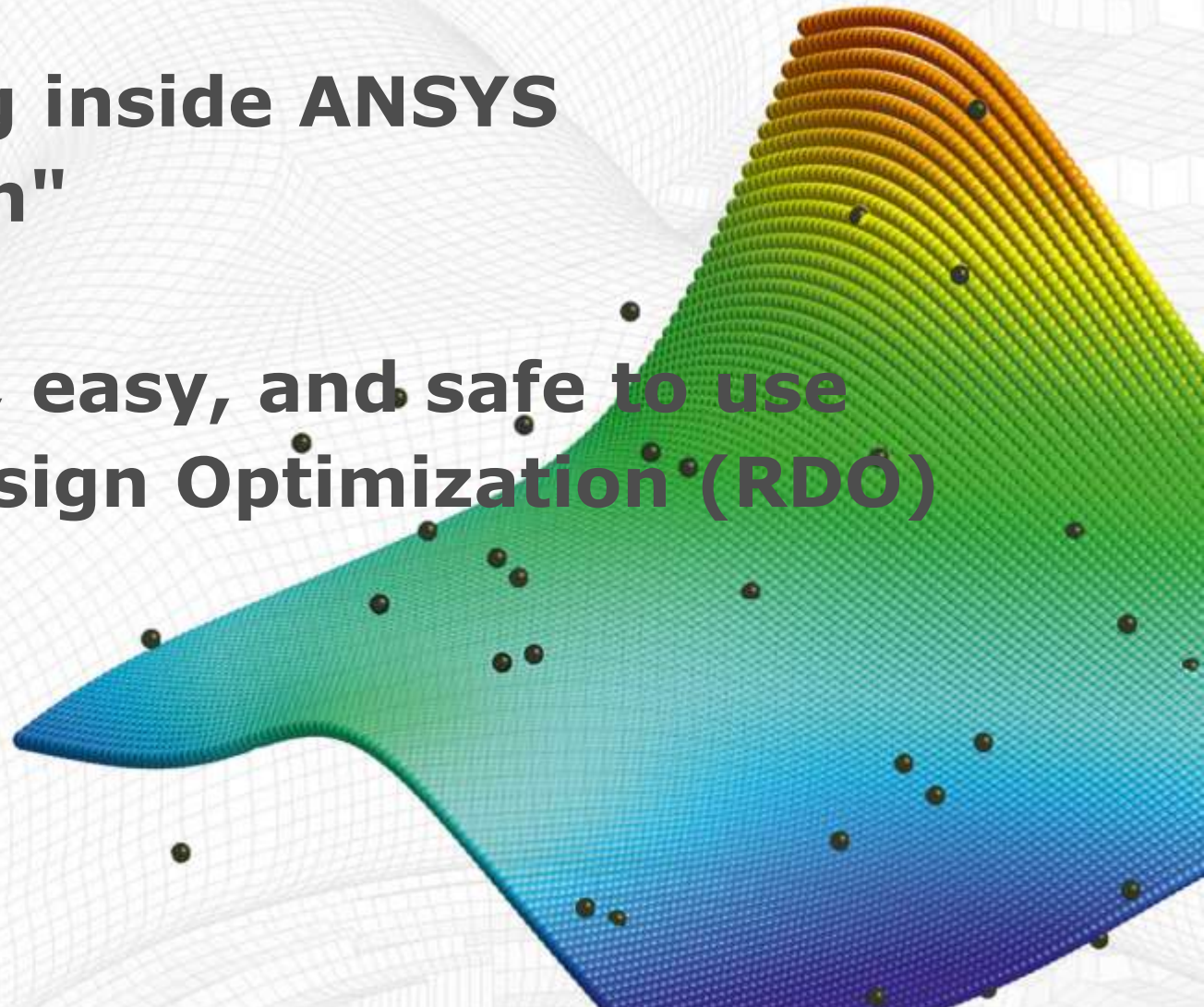
**27. 11. 2012**



# "optiSLang inside ANSYS Workbench"

– efficient, easy, and safe to use  
**Robust Design Optimization (RDO)**

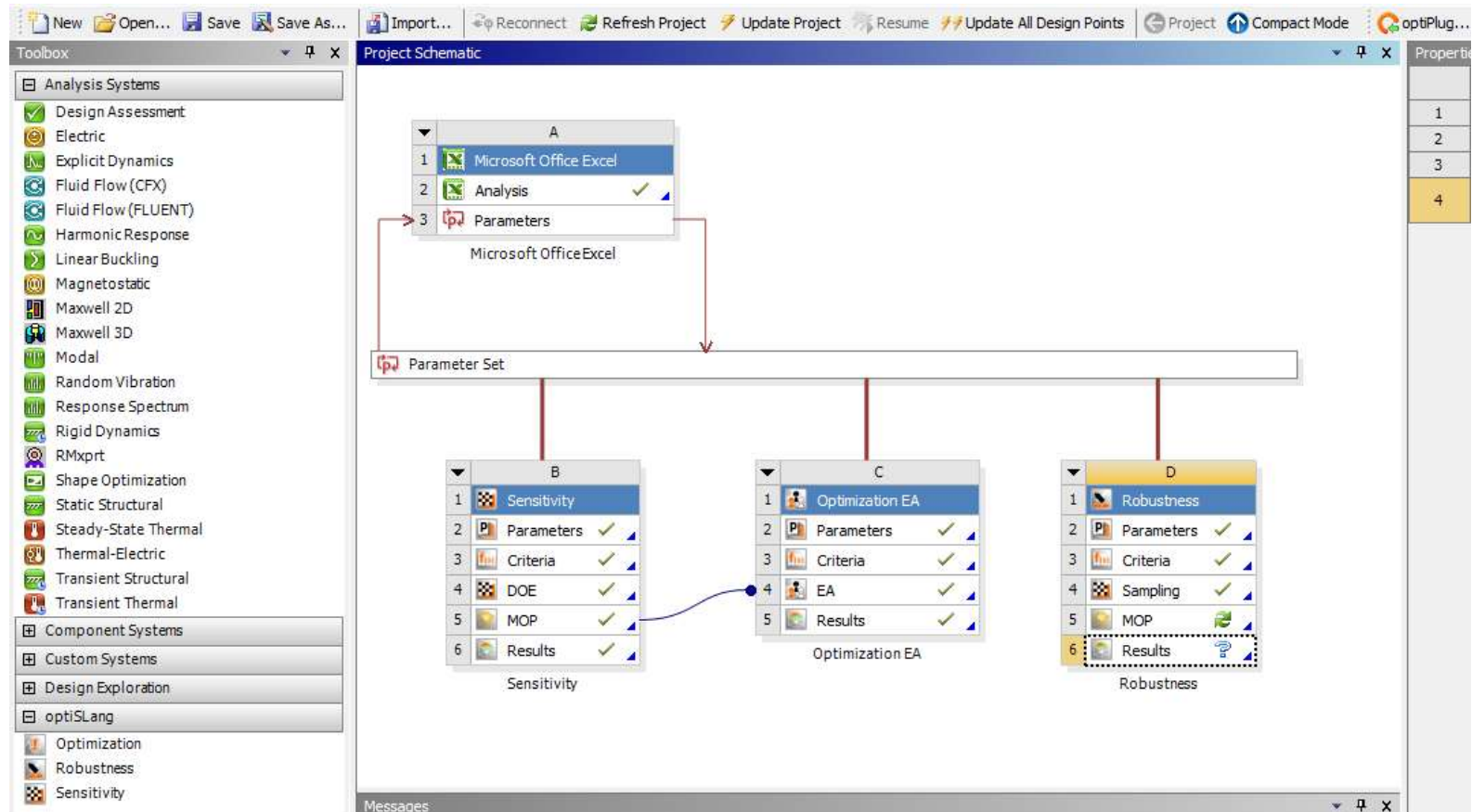
Dynardo GmbH





# optiSLang inside ANSYS Workbench v14

Modules Sensitivity+MOP, Optimization and Robustness+MOP provide „best practise“ optiSLang functionality

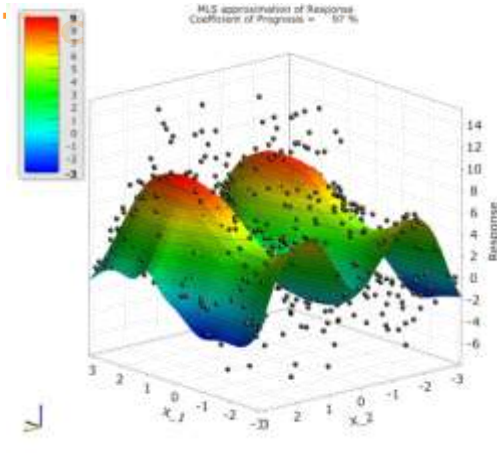


# optiSLang Flows of best Praxis

## Safe to use.

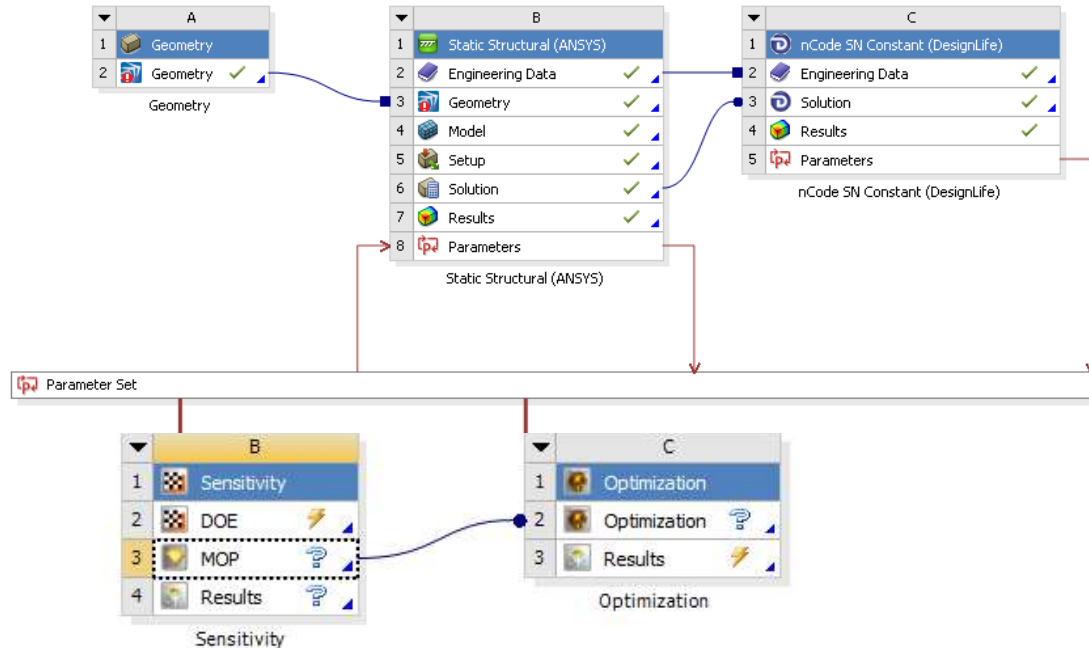
- automate best practice to „black box“ flows
- minimize the risk to miss better designs (optimization)
- minimize the risk to estimate misleading measures for robustness, safety and reliability
- offer easy to use measurements of (response variation) prognosis quality

That task requires sophisticated technology with carefully balance between number of solver calls and safety to reach the RDO goal.



that “non expert” can use it!

# optiSLang inside ANSYS Workbench



ANSYS Workbench  
parametric set up of  
complex simulations

## Easy to use:

- minimize user input
- offer best practise defaults for modules
- offer pre defined post processing modes

**User-friendliness takes care  
that it will be used!**

# Sensitivity Module

## Minimal required user input:

- Definition of parameter variation

The diagram illustrates the minimal required user input for the Sensitivity Module. It shows a project tree on the left with a 'Sensitivity' module containing steps: Sensitivity, Parameters, Criteria, DOE, MOP, and Results. The 'Parameters' step is highlighted with a red circle. An arrow points from this step to a 'Parameter set' dialog box on the right. The dialog box contains a table with 5 rows of parameters (WB\_X1 to WB\_X5) and buttons for OK, Cancel, and Apply.

	Name	Parameter type	Reference value	Resolution	Constant	Range	Range plot
1	WB_X1	Deterministic	0	Continuous	<input type="checkbox"/> non const	-3.14 3.14	
2	WB_X2	Deterministic	0	Continuous	<input type="checkbox"/> non const	-3.14 3.14	
3	WB_X3	Deterministic	0	Continuous	<input type="checkbox"/> non const	-3.14 3.14	
4	WB_X4	Deterministic	0	Continuous	<input type="checkbox"/> non const	-3.14 3.14	
5	WB_X5	Deterministic	0	Continuous	<input type="checkbox"/> non const	-3.14 3.14	

Use design as reference ▾

OK Cancel Apply

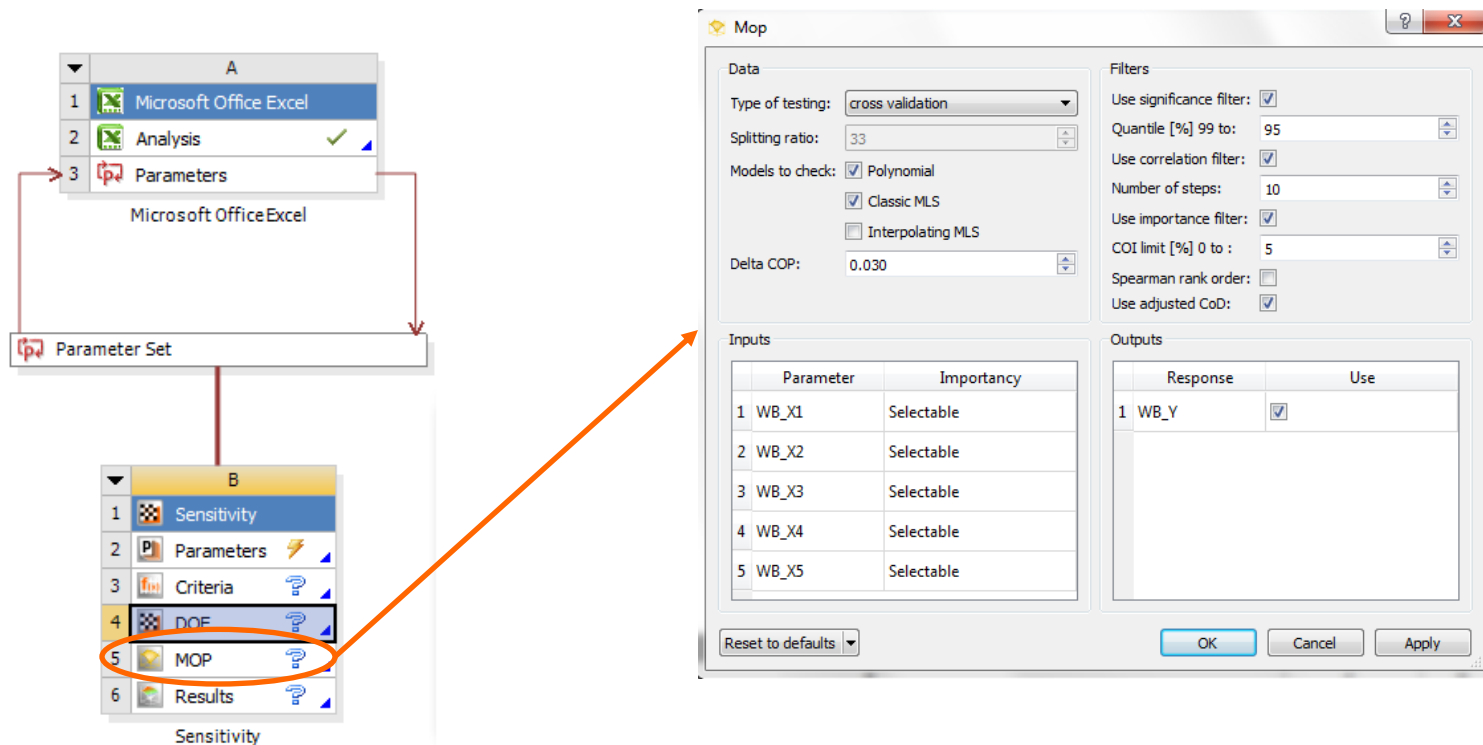
# Sensitivity Module

The Meta Model of optimal Prognosis (MOP) is automatically created out of the DOE-Sampling

**Minimal required user input: non**

Additional features:

- supports removing designs out of DOE Post Processing

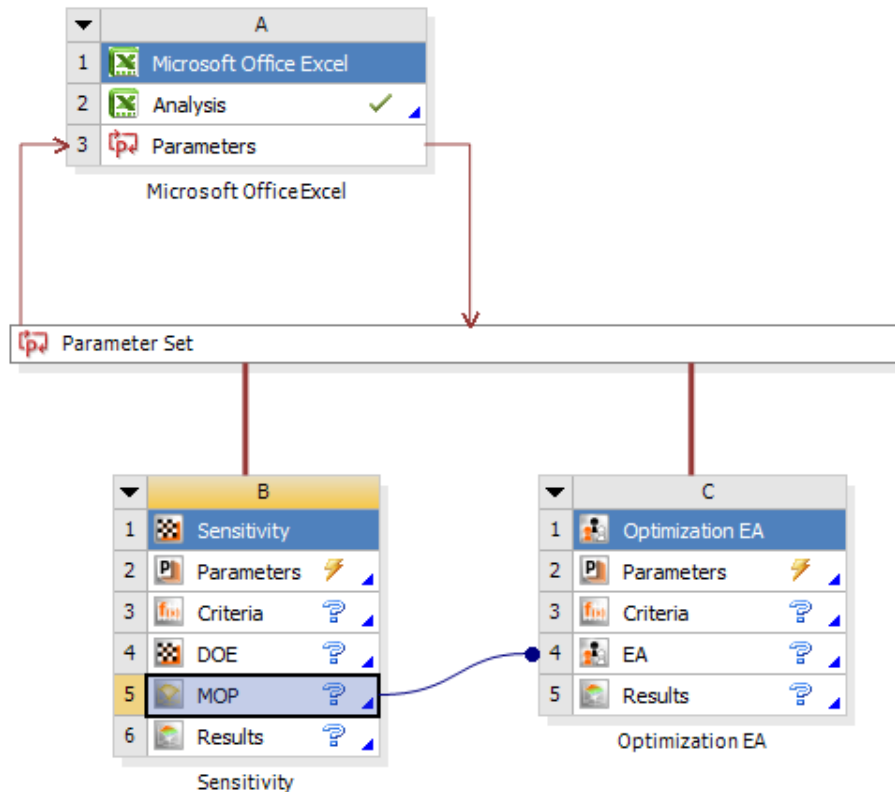


# Optimization using MOP

After sensitivity optimization using of MOP is supported.

## Minimum required user input:

- drop the optimization module onto MOP
- defining objective and constraints



“Optima” which are based on meta models need to be verified!

## Proof optima:

- Automatic verification with real ANSYS call
- Check differences in post processing

## Optimization Wizzard

**optiSLang** helps you to select a suitable optimization algorithm. Support the underlying (automatic) selection process with some additional information about the solver and the problem itself.

Exampel for using MOP and best\_design\_Sensitivity:

1. Set the analysis status as "Preoptimized" (best design from Sensitivity)
2. Set the constraint violations to "Seldom"
3. Set failed designs to "None" (MOP gives always response values)
4. Set solver noise to "None" (MOP gives a smooth surface)

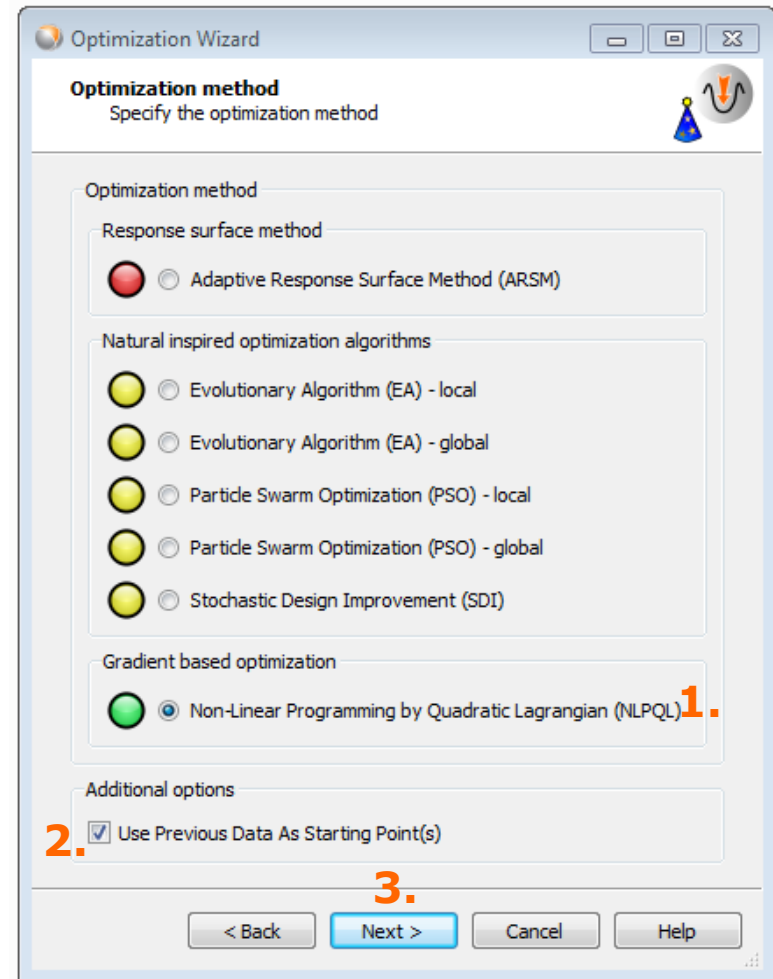
Analysis status:	<input type="text" value="Preoptimized"/>
Constraints violations:	<input type="text" value="Seldom"/>
Failed designs:	<input type="text" value="None"/>
Solver noise:	<input type="text" value="None"/>

# Optimization Wizzard using MOP

Suggested algorithm is NLPQL

Start point is automatically selected

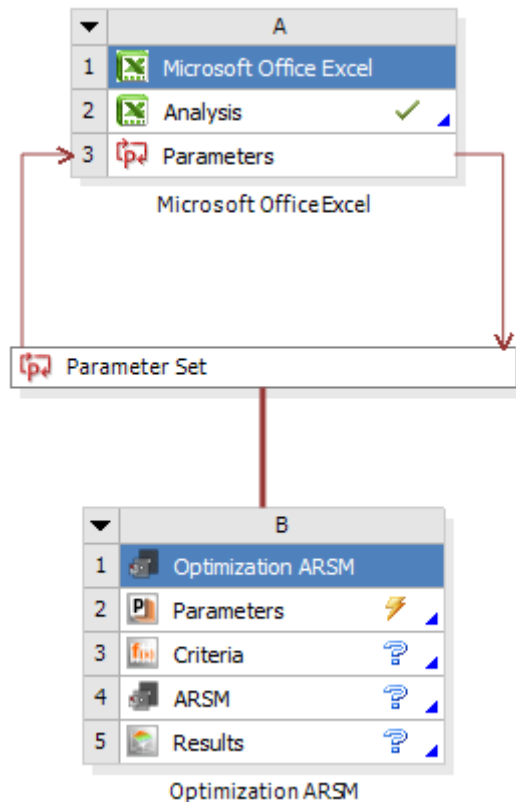
Press "Next"





# Optimization with real design calls

After sensitivity und optimization on MOP the user can continue with gradient-based, NOA-based optimization or ARSM optimization.



## Minimum required user input:

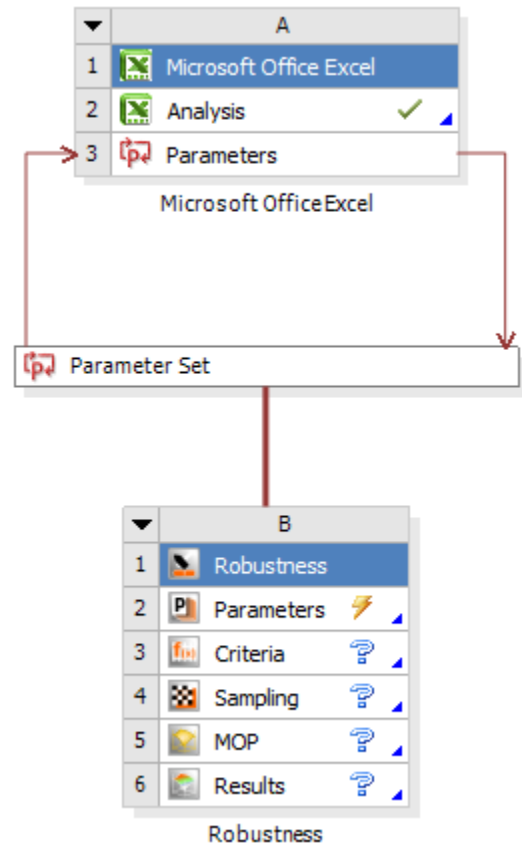
- define objectives and constraints
- choose method (Gradient-based including start design, NOA-based including best designs's out of sensitivity/MOP, ARSM in the domain of the most important optimization parameter)

For all optimizer robust default settings are provided.

*NOA - Nature inspired optimization contains: evolutionary, genetic, particle swarm optimization*

*ARSM – Adaptive Response Surface Method*

# Robustness Evaluation



## Minimum required user input:

- definition of input variation /scatter
- definition of robustness criteria
- number of samples for ALHS

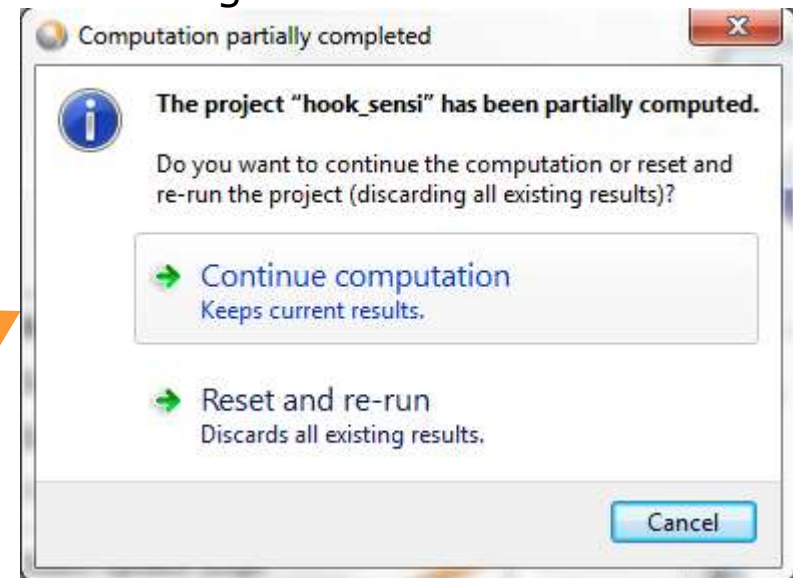
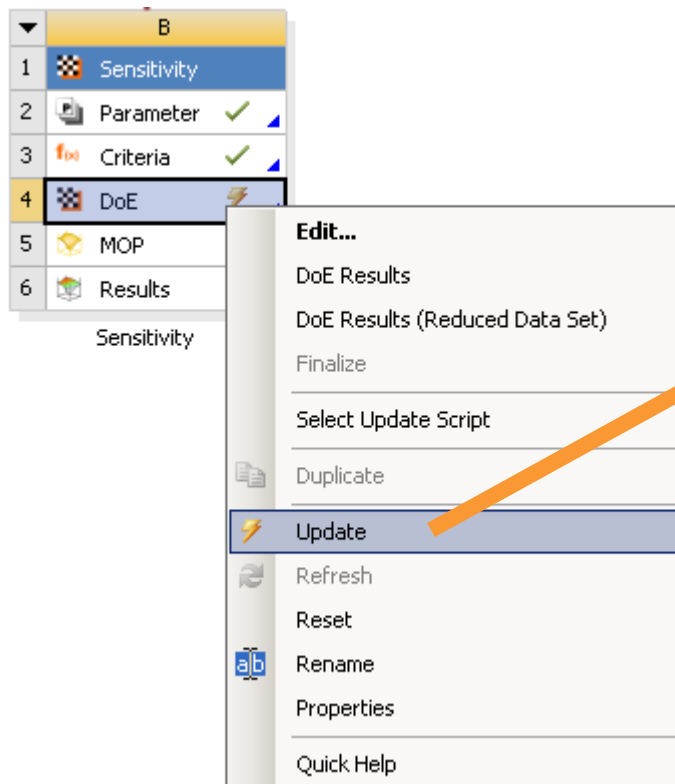
# Update Functionality





# Continue crashed session option inside ANSYS

optiSLang saves every design which was successfully calculated using update at optiSLang container continue or reset can be chosen using continue optiSLang only send unsolved designs





## Recalculate failed designs

- Due to different reasons design evaluations may fail
- With “Recalculate Failed Design Points” you can start them again

**Sensitivity**

Sampling Result designs

	Id	Activation	Violated	Duplicates	Status	AREA01	AREA
1	0.50	<input checked="" type="checkbox"/> active	false		Succeeded	5.871	17.75
2	0.49	<input checked="" type="checkbox"/> active	false		Not succeeded	16.617	1.25
3	0.48	<input checked="" type="checkbox"/> active	false		Succeeded	14.627	11.25
4	0.47	<input checked="" type="checkbox"/> active	false		Not succeeded	9.453	19.75
5	0.46	<input checked="" type="checkbox"/> active	false		Succeeded	2.289	13.75
6	0.45	<input checked="" type="checkbox"/> active	false		Not succeeded	14.229	12.75
7	0.44	<input checked="" type="checkbox"/> active	false		Not succeeded	7.065	11.75
8	0.43	<input checked="" type="checkbox"/> active	false		Not succeeded	18.209	15.75
9	0.42	<input checked="" type="checkbox"/> active	false		Not succeeded	15.025	6.75

OK Cancel Apply

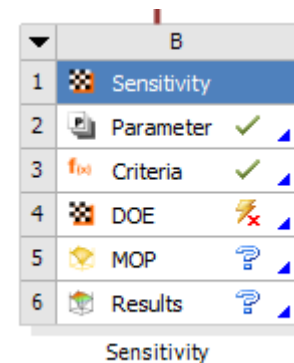
**Edit...**

- DoE Results
- Show Reduced Data Set
- Remove Reduced Data Set
- Finalize
- Recalculate Failed Design Points**
- Select Update Script
- Duplicate
- Update



# Interrupt, save, send & continue

- Stop your analysis
  - At the end of the day
  - If licenses are not available
  - ...
- Save the Workbench project
- Continue later
- saves every successful design run!
- external optimization using MOP possible!





## Update via Python scripting

- ANSYS initialize per default an update mechanism, which updates a complete ANSYS Workbench project
- Mechanism can be overridden via python file
- optiSLang provides this feature for optiSLang design evaluations
- user has full access to his ANSYS model update

Properties of Schematic B4: DoE			▼	🔍	✕
	A	B			
1	Property	Value			
2	General				
3	Component ID	DoE (optiSLang)			
4	Directory Name	Sensitivity			
5	Open Postprocessing during Solver Run	<input type="checkbox"/>			
6	Save Design Point Directories	<input checked="" type="checkbox"/>			
7	Update Options				
8	Run Python Script for Update	<input checked="" type="checkbox"/>			
9	Python Script:	C:\Users\Verwalter\Desktop\Python Scripts\update.py			



# Parallel evaluation using Ansys RSM

- ANSYS RSM is the powerful tool to distribute jobs
- optiSLang can fill the Workbench design table with a predefined number of designs
- ANSYS RSM organizes distribution of jobs
- If ANSYS RSM is installed you only need to:
  - Choose RSM Mode
  - Set max. number of parallel jobs

Properties of Schematic B4: DOE		
	A	B
1	Property	Value
2	General	
3	Component ID	DOE (optiSLang)
4	Directory Name	Sensitivity
5	Open Postprocessing during Solver Run	<input checked="" type="checkbox"/>
6	Open Postprocessing after Calculation	<input type="checkbox"/>
7	Save Design Point Directories	<input type="checkbox"/>
8	Update Options	
9	Use RSM Mode	<input checked="" type="checkbox"/>
10	Preferred Number of Design Points in Parallel	25





# ANSYS HPC Parametric Pack

## optiSLang inside Ansys Workbench v14.5

### optiSLang Algorithm Settings

- Select "Use RSM Mode" to enable parallel design point submission
- Set the "Preferred Number of Design Points in Parallel" to the intended RSM job size

Properties of Schematic B4: DOE		
	A	B
1	Property	Value
2	General	
3	Component ID	DOE (optiSLang)
4	Directory Name	Sensitivity
5	Open Postprocessing during Solver Run	<input type="checkbox"/>
6	Open Postprocessing after Calculation	<input type="checkbox"/>
7	Save Design Point Directories	<input checked="" type="checkbox"/>
8	Notes	
9	Notes	
10	Update Options	
11	Use RSM Mode	<input checked="" type="checkbox"/>
12	Preferred Number of Design Points in Parallel	4

