

presented at the 17th Weimar Optimization and Stochastic Days 2020 | Source: www.dynardo.de/en/library

Recent developments and applications of Field Meta Modelling

Sebastian Wolff, WOST 17 (June 2020)









Field meta models



Customer example: ROM for Predictive Maintenance Digital Twin

- Use flight data to increase maintenance intervals of airplane turbines and reduce cost
- Big data: Not enough data to predict failure events reliable
- Alternative: Combine simulation based Digital Twin with online data

[1] M.Eng. Holger Schulze Spüntrup (ITB Ingenieurgesellschaft für technische Berechnungen mbH), **Real-time processing with 3D meta models for predictive maintenance of aircraft engines**, *CASCON 2018*



Customer example: ROM for Predictive Maintenance Digital Twin

• Steps:

- Simulation model with ANSYS CFX and Mechanical
- optiSLang workflow for variation analysis of virtual sensors (Design of Experiments)
- Goal:
 - Search for a nonlinear 3D ROM
 - Objective: Approximate mechanical stress gradients based on transient flight data

[1] M.Eng. Holger Schulze Spüntrup (ITB Ingenieurgesellschaft für technische Berechnungen mbH), **Real-time processing with 3D meta models for predictive maintenance of aircraft engines**, *CASCON 2018*

Implementing the simulation models in an optiSlang workflow

Customer example: ROM for Predictive Maintenance Digital Twin

- Create Field-MOP for 3D stress field and temperature
- Predict prognosis quality by F-CoP
- Validate F-CoP by additional designs
- Consume Field MOP using DLL in Matlab

[1] M.Eng. Holger Schulze Spüntrup (ITB Ingenieurgesellschaft für technische Berechnungen mbH), **Real-time processing with 3D meta models for predictive maintenance of aircraft engines**, *CASCON 2018*

What is happening here?

- Use virtual sensors as parameters to a nonlinear multi-physical simulation model
- Design of Experiments varying the parameter values systematically
- Process chain in optiSLang producing the result data
- Export data from ANSYS Mechanical for Field MOP generation
- Model understanding and validation by
 - Statistical analysis,
 - Sensitivity analysis,
 - Variation pattern analysis and
 - Estimation of prognosis quality

Prognosis quality: Field Coefficient of Prognosis (F-CoP)

- Check prognosis quality in Field CoP matrix:
 - Single value ("easy to use")
 - Indicates high or low model accuracy at a glance
 - Is an average value of the CoP in space
- Check prognosis quality in Field-CoP 3D plot:
 - Plots prognosis quality for each position
 - Compare with standard deviation as an indicator of the magnitude of variation
- Accept / Repair model / Add designs ?

UX

TEMP

LOGSEOV

UY

UZ

Sensitivity analysis: See all in a single plot

- Which parameter has the largest influence at what location ?
- Where has the ROM a too low accuracy ?
- Further post processing:
 - Use statistical measures to understand the statistics of variations at each position (e.g. mean value, standard deviation, quantiles....)
 - Plot variation patterns to identify correlations in space

Innovations in optiSLang SoS 8 (2020R2): Detailed postprocessing

Open nonlinear interpolation functions used in Field MOP directly in optiSLang post processing

- Plot interpolation functions
- Easily identify statistical outliers in CV plot (highlighted in red)

Compute and visualize absolute differences between original and approximated field designs

Field Meta Models

• 1D (Signal MOP / curves)

 2D (matrices / performance maps)

• 3D (FEM meshes, CFD grids)

<section-header>

Ansys

How to deal with measurements ?

How to deal with measurements in FEM, e.g. geometries ?

Validation

Apply measurement

- Create CAD0 geometry and mesh
- Determine geometric deviations to measurement (STL)
- Morph mesh and compute structural performance

Uncertainty quantification *Robustness and Reliability Analysis*

- Create CAD0 geometry and mesh
- Determine geometric deviations to several measurements (STL)
- Create a statistical shape model
- Generate artifical geometries for DOE and use morphed FEM meshes in CAE

The key: Random fields depending on available data

- No/single measurement: assumptions (synthetic random field model)
- 2. 3-5 measurements: empirical mean+stddev assumed correlation (synthetic random field model)
- **3. Many measurements**: Empirical random field model Anisotropic, inhomogeneous, Non-Gaussian

Customer story: Few geometric measurements How do manufacturing tolerances affect low-cycle fatigue ?

- Casting process (here: gas turbine housings)
- Question: How do geometric imperfections in production influence stress / fatigue behavior ?

Customer story: Many measurements available Which production process is more reliable ?

- GOM Measurements produced compressor wheels of different production processes
- Quality tolerances are all met, but different behavior in fatigue failure

How to approach ?

- Import data and analyse statistical properties of produced geometries
- Setup ANSYS Mechanical model
- Analyse influence of geometry imperfections onto lifetime using an automated robustness analysis

Mean of geometry

New in optiSLang SoS 8 (2020R2): Filtering noisy training data

- Filter noise from measurements based on statistical filtering
- Restore original data by comparing typical correlations among all other measurements
- To be applied to:
 - Geometric measurements of production tolerances (laser scans)
 - Noisy signal data

New in SoS 8 (2020R2): Improved analysis of measurements: Easier identification of potentially "false" measurements

Check if the spatial pattern is broken :

- Compute and compare error norms for each measurement when using a random field, identify potential outliers
- Plot the error when representing a measurement by a random field

	err_max[pstrain]		е	rr_min[pstrain]	е	err_rms[pstrain]		
35	\checkmark	0.0232362	\checkmark	-0.0290202	\checkmark	0.0033847		
75	\checkmark	0.0314974	\checkmark	-0.0343026	\checkmark	0.00259706		
26	\checkmark	0.0273092	\checkmark	-0.0312857	\checkmark	0.00247323		
61	\checkmark	0.0254184	\checkmark	-0.0195148	\checkmark	0.00244799		
8	\checkmark	0.0233878	\checkmark	-0.0251521	\checkmark	0.00243145		

Easily identify histogram tails and measurement outliers by :

- Translating field data into scalars
- Measure the distance from mean
- Easily find measurements being too far from mean

Import data to Field MOP

Import data from text files

- Field MOP training needs:
 - 1. Reference mesh for connectivity and visualization
 - 2. Training data (node or element data saved on the mesh)
- Import data
 - From mesh ASCII files: ANSYS CDB, LS-DYNA Dynain, Abaqus INP, Nastran BDF, STL, VTK...
 - Element/Node data: CSV, LS-PrePost, ...
 - Converter scripts for unsupported file formats
- Mesh mapping
 - Determine coordinate deviations between CAD0 and final geometries
 - Map data between incompatible meshes

- Exports result data and mesh from ANSYS Mechanical to SoS
- Apply geometric imperfections (Uncertainty Quantification and Reverse Engineering) inside Mechanical
- optiSLang integration node for full-automated analysis

Insys

Export data and process automation

1. "Brute force": Direct export and consumption of data

- Export any data from Field MOP database as CSV to Excel, optiSLang etc.
- Connect Field MOP consumpton with 3rd party software through shared libraries:
 - Solve Field MOP and retrieve complete data vectors (3D fields, signals, etc.)
 - Access mesh connectivity
 - Use embedded scripting for full SoS capability including Field Mop creation and I/O
 - ANSI C API and examples for Matlab, C++, Python ...

305								
326	DINARDO_IMOP_API imop_error t iMOP_getModelidents							
327	(const imop db handle t datapase, imop dataopject types data_ty							
349	DYNARDO_FMOP_API_imop_error_t_FMOP_getModelParamidents							
350	(const imop_handle_t imop, char *** const param_idents, size_t							
370	DYNARDO_FMOP_API fmop_error_t FMOP_getParamLowerBounds (const fmop_							
390	DYNARDO FMOP API imop_error_t FMOP_getParamUpperBounds (const imop_							
407	DYNARDO_FMOP_API_fmop_error_t FMOP_getModelTotalAvgFCoP (const fmop							
426	DYNARDO_FMOP_API fmop_error_t FMOP_getModelAvgFCoP (const fmop_hand							
443	DYNARDO_FMOP_API fmop_error_t FMOP_getModelDim (const fmop_handle_t							
469	DYNARDO_FMOP_API fmop_error_t FMOP_getDataPointIndizes (fmop_handle							
499	DYNARDO_FMOP_API fmop_error_t FMOP_getDataPointCoors (fmop_handle_t							
500								
502								
503	/*****/							
509								
535	DYNARDO_FMOP_API_fmop_error_t_FMOP_approxField							
536	(const fmop_handle_t fmop, const double * param_values, double							
537								
	ANSI C/C++							
1	<pre>param_idents = ctypes.POINTER(ctypes.c_char_p)()</pre>							
2	<pre>num_idents = ctypes.c_ulonglong(0)</pre>							
3	sos.FMOP_getModelParamIdents (fmop, ctypes.byref(param_idents), ctypes.							
4								
5	num mesn items - ctypes.c ulonglong(u)							
0	sos.rmor_getmodelDim (Imop, ctypes.byrei (num_mesn_items))							

/ garam_values = (ctypes.c_double * num_idents) (1., 2., 3., 4., 5., 6. gapprox_field = (ctypes.c_double * num_mesh_items.value) () 10 sos.FMOP_approxField (fmop, param_values, ctypes.byref(approx_field))

Python example

2. Innovation: Export FMU 2.0 (Functional Mockup Unit)

- User can write his own analysis macros
- Combine macros into a single automated analysis
- Export workflows to FMU 2.0 (model exchange)
- Consume FMU in optiSLang or TB
- Visualize all 3D fields afterwards in SoS post prociessing

Custom solver definition Define a FMU solver by combining pre	-defined or custom macros		1	fr	n					
Function Quantity iden ✓ Evaluate Field-MOP output TEMP param TEMP input TEMP_env input TEMP_CPU input TEMP_MCU ✓ Maximum value input TEMP output max[TEMP]	t Data type node quantity string scalar quantity scalar quantity scalar quantity node quantity scalar quantity			FUNCTIONAL MOCK-UP UNIT						
	licircuit_fmop.fmu - C:\l	Jsers\ghorvath\Documer	ts - FMU_SoS			– 🗆 🗙				
	Parameter	- FMU,	SoS	Variables	=	Responses				
	TEMP_CPU 19.8	Absolute path	rs\ghorvath\Docur	ments\circuit_fmop.fmu 🔁 👻	Open	max_TEMP_ 0				
	TEMP_MCU 7.425	Inputs		Outputs						
	TEMP_env 19.8	Search for		Search for						
h h a c a c a c		Name	Value	Name	Value					
pelect a function		1 TEMP_CPU	19.8	1 max[TEMP] 0						
		2 TEMP_env	19.8							
	Input slots	3 TEMP_MCU	7.425			Output slots				
	> Standard slots					> Standard slots				
		<	>	<	>					
		Hide advanced op	otions on ☑ Use as solver	Use as output integration	ant visualization					
	< >	Designs per execution:	00000		•	< >				
	Show additional options				ОК	Cancel Apply				

2. Innovation: User macros

Macros may include

- Simple analysis macros (e.g. extract maximum along an edge)
- Post processors (e.g. Log, Exp, von Mises stress from tensor, vector norms)
- Statistical analysis over all designs (for Robustness, Reliability or Fatigue)
- Complex analysis (e.g. identification o tightness of contact areas in high pressure valves; see presentation of Tamasi et al)

Manage macros								?	Х	
Macros saved in this pr	roject:									
ldent	Short	name			FMU	SDB	2x?	@#	^	
createDummyInput	s Creat	e dummy inputs for	test							
elementApproxField	MOP Evalu	ate Field-MOP			\checkmark					
elementEvalRF	Evalu	ate random field			\checkmark					
elementGetMax	Maxir	num value			\leq					
elementGetMin	Minin	num value								
elementSetValue	Set va	lue								
elementSum	Sum	ence								
elementVec3Norm	vec3N	lorm								
elementVonMises3	von N	lises								
identifyContactTigh	tness Identi	fy tightness of a co	ntact area		\checkmark					
nodeApproxF										2
Add	۲									i
	Custom solv	ver definition	fined or custom macros	3		,				
	Function Evaluate Fi Minimum input output Set value 	Quantity ident eld-MOP value pstrain <i>min[pstrain]</i>	Data type element quantity scalar quantity			Sets part chos with	all or a p is to be s en. The r a new id	JE Part of a f set then the modified ent.	field vecti the ident field vect	or to a defined value. If the value of only a of the respective named selection can be tor is stored in rthe database and associated
ſ	input output input param	pstrain oldent[pstrain] new_value namedSelection	element quantity element quantity scalar quantity string			Inp	uts • iIde The i • new The r	nt [<i>input</i> dent of t _value new value	t <i>element</i> he origina [<i>input sca</i> e of the o	<i>ident</i>] al field quantity. <i>alar ident</i>] output field vector at the respective location:
of						Out Fur	puts •oIde The i ther ar	ent [<i>outp</i> dent of t rgumer	o <i>ut eleme</i> he modifie nts	<i>nt ident</i> j ed field quantity.
	Select a functio	1	~	Add	Remove		• nam The i	dent of t	ction [<i>sti</i> he named	<i>ring</i>] d selection, if empty then all active data
										Next Can

Cancel

×

2. Innovation: Consumption of FMUs in optiSLang workflows

- Use Field MOP FMU for simulation in optiSLang
- FMU solver node (Beta option)
 - Autoregister inputs and responses
 - Runs in optimized mode
 - Visualize all 3D fields afterwards in SoS post prociessing
 - Entirely implemented using optiSLang's powerful customization features (Python 3)

🛞 sos_demo.fmu - C:/User	s/Public/Documents/Dyn	ardo/Statistics on Struct	es - FMU_SoS	– 🗆 X					
Parameter 🛞 FM		MU_SoS	🛒 Variables 🗮	Responses					
E_Modul	Absolute path 🔹	lic/Documents/Dynardo/	atistics on Structures/sos_demo.fmu 📂 👻 Open	max_pstrain_ 0					
blank_thickness_1	Inputs		Outputs						
plastic_failure *	Search for		Search for						
poisson_ratio *	Name	Value Lo	Name Value						
real_mat_abs_1 *	1 blank_thickness	0.09 0.080	1 max[pstrain] 0						
real_mat_ord_1 *	2 E_Modul	70000 66642							
real_mat_ord_2	3 plastic_failure	0.438935 0.411							
< >	4 poisson_ratio	0.324433 0.304		< >>					
Input slots	5 real_mat_abs_1	0.04 0.036		Output slots					
	6 real_mat_ord_1	115 101.6							
	7 rho	2.757e-09 2.510							
	0 viold stross	60 40 01 >							
			Instant visualization						
Hide advanced options									
Use as input integration Use as solver Use as output integration									
✓ >> Designs per execution: 100000000									
Show additional options OK Cancel Apply									

3. Process automation for field data in optiSLang Innovation: Improved process nodes (**output** fields)

How does it work ?

- User prepares CAE solver
- User prepares SoS model for export (to CSV ? To mesh file ? To ANSYS Mechanical, LS-DYNA, Abaqus, Nastran?)

For each design:

- 1 optiSLang calls SoS to modify the CAE input deck based on scalar parameters
- 2 optiSLang calls CAE to run with modified mesh

3. Process automation for field data in optiSLang Innovation: Improved process nodes (**input** fields)

How does it work ?

- User prepares CAE solver or measurement that produces field output (e.g. a modified FEM mesh, a STL 3D measurement, a signal)
- User prepares SoS model that imports the file and projects the field data into scalar "parameters"

For each design:

- 1 optiSLang calls the CAE solver
- 2 optiSLang calls SoS to read CAE result and gets the scalar parameters
- optiSLang uses the scalars, e.g. in (Field)MOP, as inputs to CAE solvers or in optimization goals

Summary: optiSLang SoS 8 (2020R2)

Powerful analysis tools for model understanding and approximation

> Improves and simplifies data analysis for beginners and experts

> > Improves workflow automation by field-in and field-out nodes and FMU export

© 2020 ANSYS, Inc. / Confidential

