Software Supporting Parameter Optimization of Finite Element Models

Burkhard Hensel and Klaus Kabitzsch
Chair of Technical Information Systems
Technische Universität Dresden
Dresden, Germany
e-mail: burkhard.hensel@tu-dresden.de

Abstract — **When very detailed simulation of geometrically complex objects is needed, finite element models are often the best choice due to their description of spatial and temporal behavior of the modelled object. The drawback of finite element models is their large number of parameters. This makes it difficult to match a model to a real measured object by process identification methods. In this paper this problem is addressed by new software that is compatible to ANSYS as the probably most known finite element software.**

Keywords — finite element models; parameter optimization; process identification

I. Introduction

Finite element models (FE models, FEM) describe the behavior of fields both spatially and temporally in arbitrary resolution. Due to that advantage they are used in many engineering domains. The benefit of using high-quality FE models is high accuracy for spatially distributed problems.

In many cases FE models are designed to represent real existing objects. In that case, measurements of the real machine can be used to optimize the parameters of the model in order to match the model as closely as possible to the modelled object. This task is known in systems theory as process identification or system identification. However, the widely-known methods of system identification [1-3] are only suited for models with condensed parameters, e.g. differential equations and transfer functions. There are just a few publications about system identification of FE models (see Section II.C).

This problem is addressed in this paper. New software is presented that provides a user-friendly graphical interface for optimization of FE models using measurements as reference. In order to make reusing a lot of existing models possible, the software is compatible with ANSYS, the probably most known software for FE model creation and simulation. The new software does not provide an own FE solver, it just adds a "control interface for parameter optimization" on top of ANSYS. The software is designed to be extendable to other FE solvers, too.

Section II gives an overview about the problem, its complexity and the state of the art. Section III explains the identification procedure that is supported by the software. An application example is given in Section IV. Finally, conclusions are drawn (Section V).

II. Problem description

A. Kinds of parameters

FE models consist of a set of geometrically connected "elements" (compare Figure 5). Each node of the model is described by a partial differential equation. Additionally, for the elements that are not fully surrounded by other elements boundary conditions are defined.

FE models contain different types of parameters. There are the parameters of the partial difference equation of each node, parameters of the boundary conditions, and the initial values. All types of parameters can be unknown and thus subject of a parameter identification task.

B. Identification problem and complexity reduction

The quality of models is typically evaluated using the difference between the measured output variables $x_{meas,i}$ of the real object and the appropriate simulation output $x_{sim,i}$. The typical performance measure is the root-mean-square error for each measurement point *i*

$$RMSE_i = \sqrt{\frac{\sum_{j=1}^{N}\left(x_{meas,i}(j)-x_{sim,i}(j)\right)^2}{N}}, \qquad (1)$$

where *N* is the number of available measurement samples. If *M* measurement positions shall be taken into account, usually a weighted sum with weights w_i is used:

$$RMSE = \frac{1}{M}\sum_{i=1}^{M} w_i \cdot RMSE_i \qquad (2)$$

A problem of FE models regarding parameter optimization is their large number of parameters. In general, each element or node of an FE model can have an own set of parameters, resulting in an overall number of parameters that will often exceed 1,000 or 10,000. It is well known that for identification of *N* model parameters at least *N* linearly independent tuples of measurement samples must be known, because otherwise an infinite number of solutions may exist. Additionally to this, the more measurements are taken into account, the smaller is the influence of (temporary or statistical) measurement errors on the quality of the estimated parameter values. However, in most cases much less measurement samples will be available. Even if enough measurement samples would be available, the optimization of this amount of parameters would be computationally expensive and very error-prone for numerical reasons.

This problem can only be solved by reducing the number of parameters *before* parameter identification. This can be done in a combination of the following strategies:

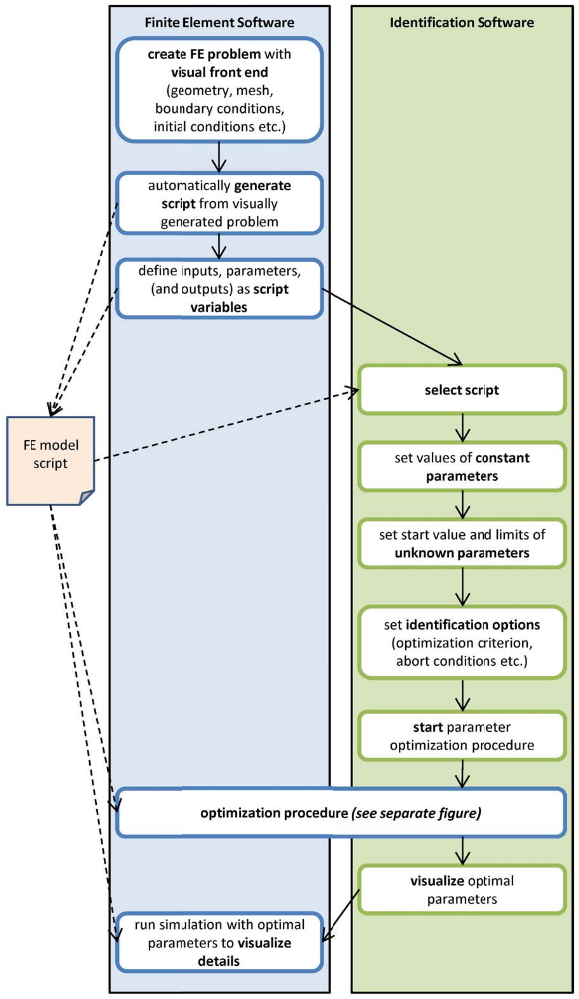
1. In most FE models, many parameters can be assumed to be equal. For example, if many elements of the modelled physical object consist of the same material, all nodes use the same parameter values. Thus, only one value has to be identified and can be used for all nodes together, perhaps with a scaling factor for taking the different sizes of the elements into account.
2. The parameters do not have the same relevance for system identification. For example, elements that are geometrically far away from the measurement points have often only little influence on the model quality and can be parameterized using estimated or empirical values, e.g. from material databases.
3. The parameters are usually not equally uncertain. For example, material properties of known steel types may be known quite accurately while properties that are the result of manual manufacturing may be very uncertain. It is usually a good idea to focus on the most uncertain parameters.

Using these strategies, the number of parameters can often be reduced to a manageable set. It should be considered that system identification does not result in the physically most plausible value for each parameter but in that parameter set that minimizes the performance measure *RMSE*. Due to this difference, modeling errors caused by the parameter reduction will be compensated by “physically wrong” values of the automatically optimized parameters so that the model may fit to the real data better than a physically plausible manual parameter setting.

C. *State of the art*

As stated above, identification of parameters is always needed when an (FE) model represents a real object. However, usual text books about finite element models do not address that topic [4-9], and text books about system identification do not mention FE models [1-3]. To the authors’ knowledge, there are only a few publications that deal with strategies for the parameter optimization of FE models [10-11]. ANSYS Workbench (the probably most known software for FEM design and simulation) provides “Parameter sets” to deal with different parameterizations of a model, but automatic parameter optimization based on measurement data is rather difficult. Khedkar et al. already developed software for updating parameters of ANSYS APDL scripts [12] but did not provide automatic parameter optimization that also requires the automatic evaluation of simulation results.

Because of this lack of software support, in practice rather simple and/or elaborate methods are used reaching from manual trial and error to writing case-specific optimization programs for each model. This experience coincides with the practice of the different project partners of the authors that deal with FE models day by day.


Finite Element Software
Identification Software
create FE problem with visual front end (geometry, mesh, boundary conditions, initial conditions etc.)
automatically generate script from visually generated problem
define inputs, parameters, (and outputs) as script variables
FE model script
select script
set values of constant parameters
set start value and limits of unknown parameters
set identification options (optimization criterion, abort conditions etc.)
start parameter optimization procedure
optimization procedure (see separate figure)
visualize optimal parameters
run simulation with optimal parameters to visualize details
Figure 1. Overall work flow.

III. Software Concept and Identification Procedure

In this section the new software and the work flow that is supported by the software are described.

A. *Basic Workflow*

Figure 1 gives an overview about the work flow that is supported by the new software.

There is a division of work between the FE software and the identification software. The new identification software is not intended or suited for creating new FE models. For that purpose “real” FE software has to be used. Also the boundary conditions as well as the partial differential equation are edited in the FE software. This avoids reimplementation of existing software and allows using all features of existing FE software.

Also the sensor placement, design of experiments and data acquisition (measurement) is assumed to be already

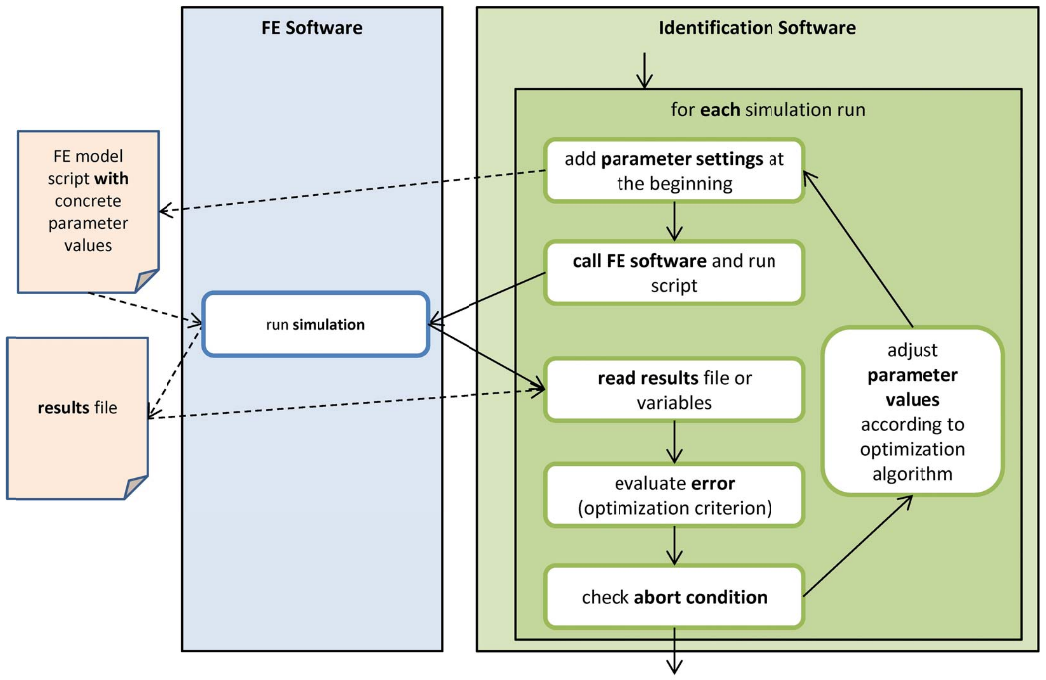
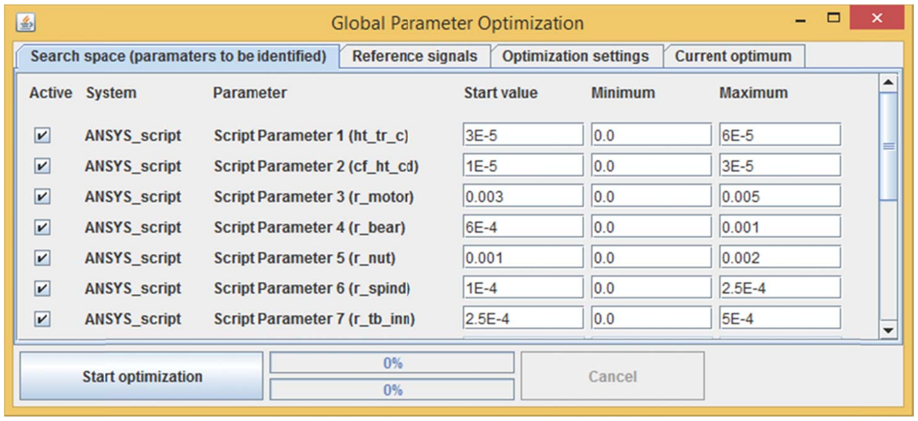

FE Software
Identification Software
for each simulation run
add parameter settings at the beginning
call FE software and run script
read results file or variables
evaluate error (optimization criterion)
check abort condition
adjust parameter values according to optimization algorithm
run simulation
FE model script with concrete parameter values
results file

Figure 2: Optimization procedure.

done and not content of this paper. The data can be imported into the identification software in different formats including usual CSV files with arbitrary column separator fields, header lines, language-specific dot symbols etc.

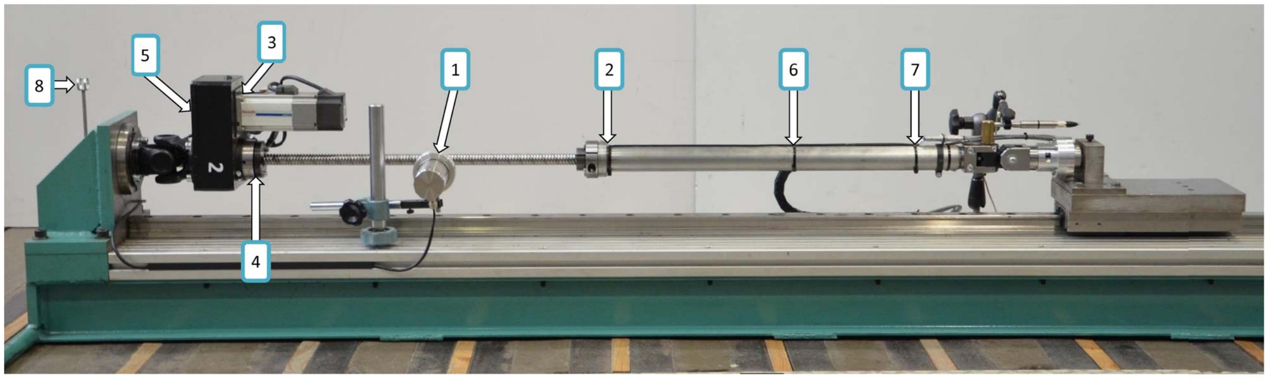
Usual FE software can be automated using script files (scripts). For ANSYS there are two types of scripts, the traditional APDL files and the newer “Journals” for ANSYS Workbench. The scripts contain commands defining geometry, boundary conditions, equation type etc. Since it is—at least for simple problems—much easier to define an FE problem using the GUI (graphical user interface) of the FE software instead of writing a script, most FE software supports the automatic generation (“recording”) of a script for the actions the user does in the GUI. In that way, the “recorded” script can be used afterwards for repeating all these steps. Further, the script can be adapted to solve other FE problems with a similar structure. To simplify that task, the script languages allow defining parameters/variables, i.e. using symbolic names those values can be set comfortably before a simulation run.

These variables are used for communication between the identification software and the FE solver. The identification software sets all unknown variables in the optimization loop, runs a simulation, reads the simulation output signals and evaluates them according to the *RMSE* performance measure. The optimization procedure is given in Figure 2. Which variables are optimized inside which bounds and with which optimization algorithm can be adjusted by the user. This is shown in the screenshot of Figure 3. Also the stop criterion (e.g. a maximum time span



Active	System	Parameter	Start value	Minimum	Maximum
✓	ANSYS_script	Script Parameter 1 (ht_tr_c)	3E-5	0.0	6E-5
✓	ANSYS_script	Script Parameter 2 (cf_ht_cd)	1E-5	0.0	3E-5
✓	ANSYS_script	Script Parameter 3 (r_motor)	0.003	0.0	0.005
✓	ANSYS_script	Script Parameter 4 (r_bear)	6E-4	0.0	0.001
✓	ANSYS_script	Script Parameter 5 (r_nut)	0.001	0.0	0.002
✓	ANSYS_script	Script Parameter 6 (r_spind)	1E-4	0.0	2.5E-4
✓	ANSYS_script	Script Parameter 7 (r_tb_inn)	2.5E-4	0.0	5E-4

Figure 3. Optimization dialog (initial value and limits of parameters)



Temperature sensors:			
1 spindle (pyrometer)	3 drive	5 belt housing	7 tube end (hand)
2 nut	4 bearing	6 tube middle	8 air

Figure 4. Actuator strut used for case study and installed temperature sensors.

for optimization or a maximum number of iterations) can be defined. Three types of optimization are currently provided by the optimization software: Monte Carlo optimization, Simulated Annealing, and Evolutionary Optimization.

After the optimization run has finished, the identified values are shown to the user. All model parameters are set to the best values and a simulation run can show the simulation outputs of the found optimum. The user can either go back to the FE software and analyze this solution using all features of the FE software, or they can change the optimization settings (e.g. wider bounds, other start parameters or another optimization algorithm) to make a further optimization run.

B. *Implementation notes*

The concept has been implemented as a prototype using ANSYS APDL scripts for model description. Since the proposed work flow works for all kinds of script-based simulation software, it is planned to couple also other FE solvers. The basis of the new software is a “general purpose” identification software described in [13].

Since ANSYS provides no API and the results file format is closed, it is necessary to export the relevant time series as CSV files via appropriate APDL code. For that purpose a wizard has been included in the new identification software so that this code can be generated automatically for all user-specified sensor positions. This automatic support is one of the major differences to general purpose system identification software regarding finite element models.

IV. CASE STUDY: ACTUATOR STRUT

The concept of the software is explained using a simplified practical application example.

A. *Description of the modelled object*

The modelled object is an actuator strut, shown in Figure 4. Actuator struts are used in machine tools for exact positioning of the tool and/or work piece. The actuator strut consists of a spindle that is (partially) inside a steel tube. The spindle and the tube are connected via a ball screw nut. A motor rotates the spindle, moving the ball screw nut with the tube to the left or to the right, depending on the rotation direction. The left end of the spindle is mounted in a bearing.

The goal of modelling the actuator strut is to analyze (and later predict) the time-varying temperature field of the actuator strut, because the temperature slightly influences the size due to thermal expansion and therefore the production accuracy of the machine tool of which the actuator strut is a part of [14]. There are three relevant heat sources: The friction of the ball screw nut, the friction of the bearing, and the power loss of the motor. However, the values of these sources, i.e. the heat flows, are unknown as they depend on the handcrafted manufacturing of the concrete strut. Therefore, these values have to be identified using measurements. Since it is also unknown, which amount of the heat generated in the ball screw nut flows into the tube and which amount into the spindle, two parameters are used that have to be identified. Additionally, there is unknown heat transfer from the spindle to the tube via the air inside the tube.

The temperature is measured at 7 positions at the strut and the air, see Figure 4.

The goal of this demonstration example is not to reach a maximum of accuracy but to show the principal work flow. The actuator strut has only been modelled in two dimensions (2D model), because only the change of the length is practically important. Also the movement of the actuator strut is not modelled, because the heat flow can be averaged since one cycle of movement is short (a few seconds) compared to the change of the temperatures (minutes to hours).

The measurement data contains 74 samples. Therefore, theoretically, up to 74 independent parameters could be estimated. However, only the five parameters mentioned above are estimated automatically.

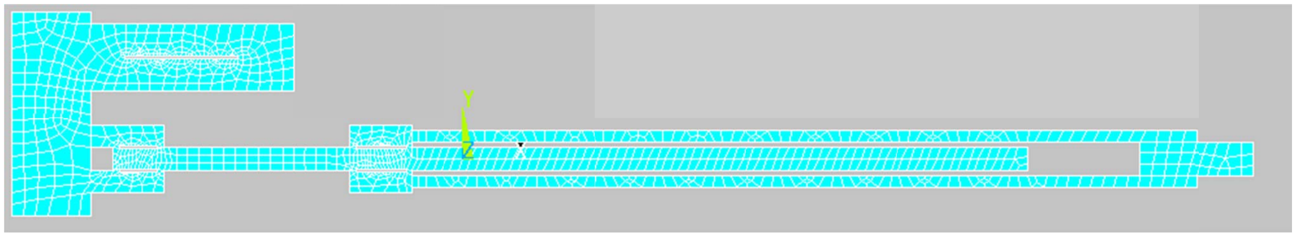


Figure 5. Mesh (elements) of the ANSYS model of the actuator strut.

The parabolic PDE representing the heat flow inside the strut is

$$\frac{\partial T}{\partial t} - \mathrm{div}(k_2 \cdot \mathrm{grad}\, T) = 0 \qquad (3)$$

with the thermal diffusivity (coefficient of temperature conduction)

$$k_2 = \frac{k}{\rho \cdot c}, \qquad (4)$$

where ρ is the density, *c* the specific heat capacity, *T* the temperature difference to the initial temperature, and *k* the coefficient of heat conduction (thermal conductivity). grad *T* is the gradient of the temperature *T*.

The boundaries with contact to the air are modelled as free heat convection, i.e. as "generalized" Neumann boundary conditions

$$\underline{n} \cdot k \cdot \mathrm{grad}\, T = -q \cdot T, \qquad (5)$$

where q is the heat transfer coefficient and $\underline{n}$ the normal vector on the surface.

Heat entries into the modelled object due to friction and the motor are modelled as Neumann boundary conditions

$$\underline{n} \cdot k \cdot \mathrm{grad}\, T = g, \qquad (6)$$

where g is the reference heat flux (i.e. heat flow per area). For that purpose, at the heat sources small regions of "empty space" have been integrated so that the heat can be represented as boundary condition at the boundary of that space. The alternative would be to use a source term Q in the partial difference equation (3), but this solution would be technically more complicated without a high probability for getting significantly better simulation results at the sensor positions.

B. *Model implementation*

ANSYS 18.1 has been used to model the actuator strut. The model has 4197 nodes and 1129 elements that have been generated with the ANSYS "MeshTool" using "Smart size 3" and element type "PLANE77", what is an abbreviation for curved 8-node elements. The mesh is shown in Figure 5. The simulated time span is roughly 3 hours. One simulation took about 19.5s of computational time.

C. *Optimization*

Among the currently supported optimization algorithms, evolutionary optimization produced the best results in the same time for the given example application. However, the quality of results depends strongly on the user-specified bounds of the parameters to be optimized. Therefore, several optimization runs have been done where the optimum of the preceding run has been taken as initial parametrization of the subsequent optimization run. After each run the parameter bounds have been set to roughly the double value of the optimal value before. This has been repeated until the optimized values did not change any more significantly. Each optimization run needed between 30 and 120 minutes containing up to 370 simulations.

The temperature field at the end of the simulation with optimized parameters is shown in Figure 6. The values of the parameters after the full optimization process are presented in Table 1. These values are partially different from the theoretically expected ranges. There are several reasons for that phenomenon: the two-dimensional modelling, inaccuracy of manually estimated parameters, and deviations between the geometrical properties of the model and the real object including exact sensor placement. Exact modelling was not a central goal of this paper (but only the demonstration of the principle work flow) and can be subject of future work. By the way, the two-dimensional modelling is also the reason why the units of the parameters do not match to the three-dimensional ones.

The *RMSE$_i$* for each measurement point (except the air)

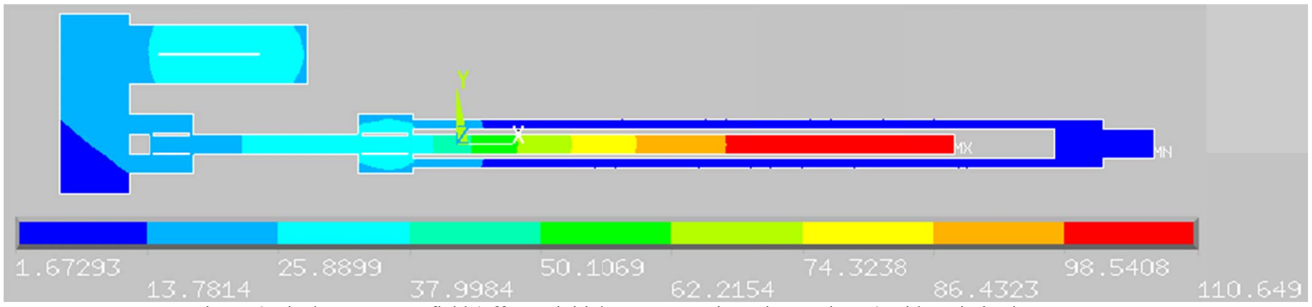


Figure 6: Final temperature field (offset to initial temperature in K, lower picture) with optimized parameters.

Table 1. Values of Parameters after Optimization

Parameter	Value	Unit
Heat transfer coefficient q_2	$3.1 \cdot 10^{-5}$	m^2/s (2D)
Coefficient of temperature conduction k_2	$1.2 \cdot 10^{-5}$	m^2/s
Motor heat flux $g_{2,m}$	$2.5 \cdot 10^{-3}$	$(m^2K)/s$ (2D)
Bearing heat flux $g_{2,b}$	$6.3 \cdot 10^{-4}$	$(m^2K)/s$ (2D)
Nut to tube heat flux $g_{2,nt}$	$1.0 \cdot 10^{-3}$	$(m^2K)/s$ (2D)
Nut to spindle heat flux $g_{2,ns}$	$1.3 \cdot 10^{-4}$	$(m^2K)/s$ (2D)
Radiation heat flux inside the tube $g_{2,it}$	$2.5 \cdot 10^{-4}$	$(m^2K)/s$ (2D)

Table 2. Values of parameters after optimization

Sensor position	$RMSE_i$ in K
Spindle	1.26
Nut	1.54
Drive	2.31
Bearing	2.09
Belt housing	2.96
Tube middle	2.10
Tube end (hand)	0.97
Mean of all	**1.89**

and their mean value are given in Table 2.

If only one measurement point would be optimized, the $RMSE_i$ for that location could become much smaller, because all 5 optimization parameters would be used to minimize this $RMSE_i$. However, in general this results in a larger $RMSE_i$ for the other measurement points.

V. Conclusion, discussion and outlook

This paper presented new software for optimizing parameters of finite element models, i.e. process identification. This is the basis for tasks like model-based deformation compensation due to load and thermal expansion [14].

The advantages of the new software compared to general purpose system identification software are the link to ANSYS as the most widely used finite element software, a wizard for script code generation focused on FE models, a set of optimization methods that are suitable for FE models, and the generic methodology that is in principle compatible to many state-of the art FE solvers. The compatibility to ANSYS has been demonstrated. Therefore, it is not necessary to reimplement existing models for optimizing their parameter values. The same strategy would also work for other script-based FE solvers, e.g. OpenFOAM or ANSYS Workbench. Because of the generic concept, the adaptation effort to support these FE solvers will be relatively small. Additionally, the current state builds the basis for exploring enhanced parameter identification techniques for finite element models. This is simplified due to the modularised architecture of the system identification software that can be simply extended by plug-ins.

Acknowledgment

This work has been funded by the German Research Foundation in the collaborative research centre/Transregio 96.

References

[1] R. Isermann and M. Münchhof, Identification of Dynamic Systems - An Introduction with Applications. Heidelberg: Springer, 2011. DOI: 10.1007/978-3-540-78879-9

[2] L. Ljung, System Identification - Theory for the User. 2nd. Upper Saddle River, NJ, USA: Prentice Hall, 1999.

[3] T. Söderström and P. Stoica, System Identification. Hemel Hempstead: Prentice Hall, 1989.

[4] I. Babuška and T. Strouboulis, The Finite Element Method and its Reliability. New York: Oxford University Press, 2001.

[5] Z. Chen, Finite Element Methods and Their Applications. Berlin: Springer, 2005. https://doi.org/10.1007/3-540-28078-2

[6] G. Dhondt, The Finite Element Method for Three-dimensional Thermomechanical Applications. Chichester: John Wiley & Sons, 2004.

[7] M. R. Eslami, Finite Elements Methods in Mechanics. Cham: Springer, 2014. DOI: 10.1007/978-3-319-08037-6

[8] P. E. Lewis and J. P. Ward. The Finite element method: principles and applications. Wokingham: Addison-Wesley, 1991.

[9] D. W. Pepper, and J. C. Heinrich, The Finite Element Method: Basic Concepts and Applications. Taylor & Francis, Hemisphere Publishing, 1992.

[10] M. Sanayei and P. Rohela, "Automated finite element model updating of full-scale structures with PARameter Identification System (PARIS)." Advances in Engineering Software, 2014: 99-110. https://doi.org/10.1016/j.advengsoft.2013.09.002

[11] H. Wernsing and C. Büskens, "Parameter identification for finite element based models in dry machining applications." Procedia CIRP, 2015: 328-333. https://doi.org/10.1016/j.procir.2015.03.037

[12] S. S. Khedkar, S. S. Chaudhari, and P. D. Kamble, "Analysis of 3-D Model in ANSYS 9.0 by Java Program and Macros Using Interlinking Concept Verification Though the CFD Analysis." Proceedings of the International Conference on Advances in Mechanical Engineering (AME 2010). Kerala, India, 2010. 93-96. DOI: 02.AME.2010.01.532

[13] B. Hensel, S. Schroeder, and K. Kabitzsch. "New coordination software for parameter identification applied to thermal models of an actuator strut." Journal of Computational Engineering, September 2017: 12 pages. https://doi.org/10.1155/2017/3169785.

[14] K. Großmann, ed. Thermo-energetic Design of Machine Tools: A Systemic Approach to Solve the Conflict Between Power Efficiency, Accuracy and Productivity Demonstrated at the Example of Machining Production. Cham: Springer, 2005. DOI: 10.1007/978-3-319-12625-8