MagOpt - Optimization Tool for Mechatronic Components

Siegfried Silber[a], Werner Koppelstätter[a], Günther Weidenholzer[a], Gerd Bramerdorfer[b]

[a] Linz Center of Mechatronics GmbH, Altenbergerstrasse 69, A-4040 Linz, Austria, siegfried.silber@lcm.at
[b] Johannes Kepler University Linz, Altenbergerstrasse 69, A-4040 Linz, Austria

***Abstract*—MagOpt is a modular software tool for the simulation and optimization of mechatronic components. It features a flexible structure for the storage of complex data and an open and modular interface to existing third party programs like CAD systems, finite element programs and other simulation software. Parametric design optimization can be carried out with various different optimization strategies like gradient-based methods or multi-objective evolutionary or genetic algorithms.**

I. Introduction

In the past the optimization of mechatronic components was done manually by varying key design parameters. With such an approach the number of different designs to be evaluated is limited from just a few up to several hundreds. To speed up the design process a more automated approach is demanded. Normally, calculating a mechatronic component is associated with analyzing a mathematical model that represents the characteristics of this component (in a specific domain). Characteristic quantities of an electric motor are, e.g., electrical properties, like voltages, currents, efficiency and many more properties of the geometry and the material parameters of the motor. Thus, parameter variation incorporates modifying any parameter that has an impact on the performance of the mechatronic component. This, of course, comprises geometric and material parameters. Simulation in different physical domains often necessitates solving nonlinear PDEs by means of finite element analysis (FEA). Usually it takes from several minutes up to hours to solve FEA problems. For a fully automated simulation of mechatronic components these FEA problems are the limiting factor concerning the throughput because of the high requirements in computation time. However, that bottleneck can be opened up by calculating the FEAs in parallel on a computer cluster.

For a systematic optimization approach the simulation and optimization tool MagOpt (short for Magnetic Optimization) has been developed. Although MagOpt was originally designed for magnetic problems it turned out that, due to the very general structure it can also be applied for analyses performed in different physical domains and thus for different mechatronic components. For instance electromagnetic actuators, motors, magnetic bearing systems, hydraulic actuators and many more can be evaluated using MagOpt. Various simulations and optimizations of electrical motors, magnetic bearings and bearingless motors have been carried out recently (see section IV).

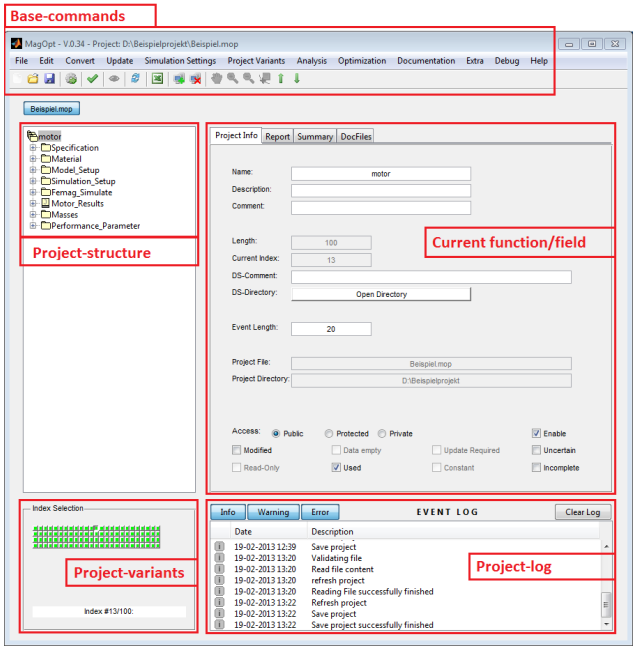

Figure 1. Screenshot of the MagOpt user interface.

II. Functionality of MagOpt

The simulation and optimization tool MagOpt has been developed at the Linz Center of Mechatronics and the Institute of Electrical Drives and Power Electronics of the Johannes Kepler University Linz.

Figure 1 shows a screenshot of the MagOpt main window. This graphical user interface (GUI) is partitioned into several panels where the key element “Project-structure” is used to define the project setup which is represented as a tree structure. Details of the selected item are summarized in the panel “Current function/field”. A project may contain multiple variants differing by a set of altered parameter values based on the same project structure. Messages, warnings and errors which occur during the execution or simulation are displayed in the “Project-log” view.

A MagOpt project is organized by using a tree structure so that related parameters and functions can be arranged in separate branches. The following types of tree elements are currently supported:

- *Project*: Base element that specifies a MagOpt project. This is typically the root element in the parameter tree.

- *Container*: A node element for a branch and can be used for structuring of parameters. Any MagOpt element except a *Project* element can be inserted. The scope and visibility of the parameters inside such a *Container* can be specified (this means how the parameters are to be referenced inside and outside this *Container*).
- *Field*: Element for data storage. Various data types for numerical values, matrices, periodic signals, structs or character strings are supported.
- *Formula*: Simple algebraic equations. Any field element in the tree may be used as argument and the result is stored within the *Formula* element itself.
- *Function*: Interface to complex modules. This element is described in more detail in section *II-B*.

A. Scope and visibility of parameters

Scope and visibility refers to the availability of *Field* parameters for use in *Formulas* and *Functions*. There are three different levels to set the visibility of parameters within a *Container*:

- Public: Parameters in public containers are visible within the complete project tree unless a parameter with the same name exists at a higher level in the tree. These parameters can be referenced by the parameter name (e.g. `paramname`). If the scope of a parameter is hidden by a public parameter with the same name at a higher level in the tree (closer to the root) reference to that parameter can be achieved by applying a uniquely defined parameter identifier e.g., `container.paramname`.
- Protected: These parameters can be accessed inside a branch by using their parameter names. Outside this *Container* at least the name of the protected container has to be added with a dot-separator like `container.paramname`.
- Private: These parameters are only visible inside the container and sub-containers and cannot be referenced from outside.

B. Functions

The MagOpt element *Function* is intended to implement complex calculation modules and thus can be used as interface to external third party simulation tools. These *Functions* have to be coded either in Java or in script languages. Currently the script languages Matlab, Python, JavaScript and Lua are available. A *Function* element has the same properties as a *Container* so that associated parameters can be grouped. The interface of MagOpt parameters to internal Java or script variables is realized by import and export lists as shown in Fig. 2.

With these lists the internal Java or script variables are assigned to MagOpt parameters. If import parameters are not assigned default values are used instead, unless the particular variable is specified as required. In such a case an error message is generated. The interface also supports the specification of physical units. This means that the physical unit of the assigned parameter has to fit the expected unit in the *Function*. If feasible the values are automatically converted to the specified unit in the *Function*, otherwise an error message is thrown.

IMPORTS

Variable	Parameter	OK
N	Ns	☑
p	pz	☑
m	m	☑
SingleLayer	SingleLayer	☑
dyn	dyn	☑
PhaseConnection	PhaseConn	☑
GroupConnection	GroupConn	☑

EXPORTS

Variable	Parameter	OK
WdgData	WdgData	☑
Xi1	xi1	☑
Nu	Nu	☑
yn	yn	☑
Xil		☐
Xim		☐
Xi		☐

Figure 2. Interface to MagOpt Functions. With the import and export list the internal Java or script variables are assigned to MagOpt parameters.

C. Calculation and Simulation

Calculation or simulation means that the *Formulas* are evaluated and the *Functions* are executed. The order in which *Formulas* and *Functions* are processed is resolved by the import and export dependencies. The first *Function* that is evaluated must not have any input parameter that depends on export values of any *Function* or *Formula*. Subsequently, only *Functions* where all the input parameters are updated can be evaluated. This entails that circular dependencies cannot be resolved.

If parameters are modified only *Functions* with direct or indirect (via results of other *Functions*) dependencies of these parameters need to be evaluated. Thus, it is not necessary to evaluate all *Functions* of the project if just a few parameters are modified.

With *Functions* also external third party simulation tools can be executed. The processing time for these programs might become considerably long, especially when finite element programs are executed. To increase the throughput, MagOpt automatically distributes time consuming simulations to a computer cluster and collects the results upon completion of the calculation. As soon as the results are available the corresponding result parameters in the project are set.

D. Components

If MagOpt is employed for simulation and optimization of similar mechatronic components, the projects will also have a quite similar structure. For example, in the case of electric motors this means that any motor has a stator and a rotor and the simulation is done by using a magnetic FEA independent of the size of the motor. But the stators and rotors might be different in size and shape. To avoid that any project has to be set up manually from scratch *Components* are available in MagOpt. Such a *Component* is a collection of MagOpt parameters and can be inserted in the project as a sub-tree. *Components* could of course include *Functions*, *Formulas* and any other type of parameter. To ensure that a *Component* seamlessly fits a project, acceptance filters can be set. For example, if a motor with an internal rotor is simulated only corresponding stators for interior motors are accepted and stators for exterior rotors cannot be selected. For the simulation of various different mechatronic components template projects exist where just a few *Components* need to be added to completely setup the project.

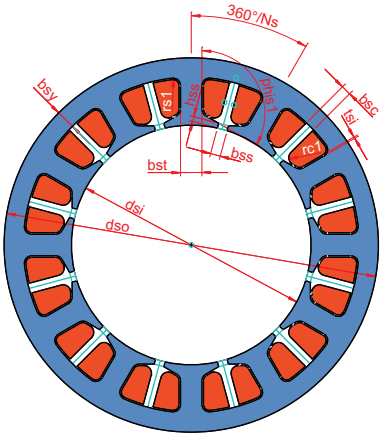

Figure 3. Parametric model of a stator of an electric motor available as a MagOpt *Component*.

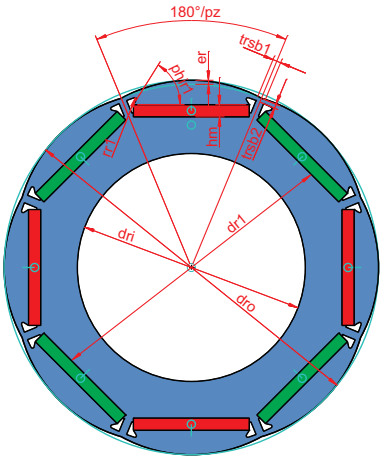

Figure 4. Parametric model of a rotor of an electric motor available as a MagOpt *Component*.

Components for various simulation tasks are currently available. For instance rotor and stator models for synchronous machines are available for different 3D CAD systems (PTC: Creo, Pro/Engineer; Dassault Systems: SolidWorks, Siemens: NX 9...). In Fig. 3 and Fig. 4 a parametric model of a stator and a rotor are shown respectively. All the geometric parameters are directly accessible via corresponding MagOpt parameters. If a MagOpt parameter is modified the CAD model and all other assigned mass properties and drawings are regenerated. The outputs of the CAD system are usually assigned to MagOpt parameters.

For magnetically levitated high speed drives an interface to the multi body simulation software HOTINT is available [1]. With this tool the eigenfrequencies of the shaft are calculated. Moreover, also the mechanical stress due to centrifugal forces in the material can be evaluated. An example of a high-speed rotor simulated with HOTINT is shown in Fig. 5.

III. Optimization algorithms

Optimization of a mechatronic component means to fulfilling the requirements in such a way that certain target parameters (objectives) reach their best or optimal values.

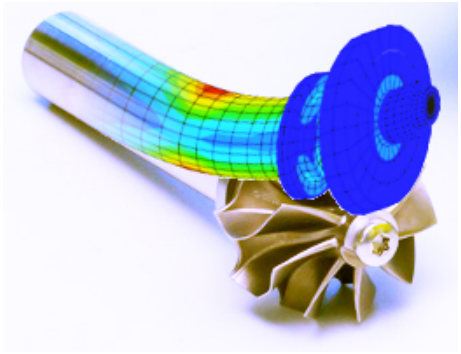

Figure 5. Example of a high-speed rotor where the first bending mode is simulated with HOTINT.

It is obvious that the requirements can be fulfilled with different embodiments. Nevertheless, those embodiments usually feature different characteristics for further performance parameters.

By means of optimization the objective parameters should reach an optimal (a minimal or maximal) value. In many cases there is not just one single objective to be optimized, but there are several simultaneously. This leads to multi-objective optimization problems. For a nontrivial multi-objective optimization problem, there exists no single individual that shows best behavior for each objective. In that case, the objective functions are said to be conflicting, and Pareto-optimal solutions exist. An individual is said to be Pareto-optimal if there are no other designs that better satisfy all the considered objectives. In other words, any improvement in one objective necessitates the worsening of at least one other objective [2].

In MagOpt well known and widely applied genetic algorithms for multi-objective optimization are implemented. Additionally we also implemented improved algorithms in terms of simulation speed. Especially for FEA problems an advanced algorithm was developed that improves convergence of the Pareto-front significantly [3]. Currently the following optimization algorithms are implemented in MagOpt:

- Grid calculation: Calculates any possible parameter combination and thus requires very high computational effort.
- Generational NSGA-II (Non-dominated Sorting Genetic Algorithm II) [4]
- Steady State Asynchronous NSGA-II
- Generational SPEA2 (Strength Pareto Evolutionary Algorithm 2) [5]
- Steady State Asynchronous SPEA2
- DECMO (Differential Evolution-based, Coevolutionary Multi-objective Optimization algorithm)

Evolutionary algorithms such as the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) and Strength Pareto Evolutionary Algorithm 2 (SPEA2) are generation based approaches. This means that a new generation is created as soon as all individuals of the current generation have been evaluated. This, however, is a major disadvantage for optimization problems where FEAs are distributed on a computer cluster, especially when the simulation time varies much between the individuals. This is due to the fact that all the simulations have to be

finished before new individuals can be submitted to the cluster. This leads to load imbalance on the computer cluster.

To overcome this problem steady-state asynchronous algorithms based on NSGA-II and SPEA2 have been developed and implemented in MagOpt [6]. In contrast to generation-based algorithms, using the steady-state algorithm it is not necessary to wait until each individual of a generation is evaluated until a new generation is created. This algorithm allows more generations to be evaluated in parallel even though it seems that this may lead to worse convergence. But for practical examples the convergence of the Pareto-front could even be improved as shown in [6].

A. DECMO

Modern multi-objective evolutionary algorithms use methods based on differential evolution (DE), because benchmark tests showed that with DE methods the design space can be explored far more efficiently. However, as regards some problems, state of the art algorithms like NSGA-II and SPEA2 still significantly outperform DE-based algorithms. These conventional algorithms also seem to be more robust with regard to their parameterization on a wide range of test problems with the disadvantage that the design space is not explored that comprehensively.

To improve the optimization algorithm we combined both algorithms as described above to gain advantage from the robustness of classic multi-objective algorithms and also from the very good performance exhibited by DE methods. This algorithm is called Differential Evolution-based, Coevolutionary Multi-objective Optimization algorithm (**DECMO**) and is described in detail in [3].

IV. Examples

In the last few years various optimization procedures have been performed on electric motors by using MagOpt. For instance, a wheel hub motor for an electric scooter was optimized [7]. Also the efficiency of industrial motors was increased whereas simultaneously the amount of rare earth magnets could be significantly reduced [8], [9], [10], [11]. MagOpt was also used for hardware in the loop optimization for a fast switching hydraulic valve. In that case the control parameters and the current waveform were optimized by controlling the system [12], [13]. Simulation of a passive permanent magnetic bearing and optimization of its stiffness further demonstrate the versatility of MagOpt [14].

Comparison of different embodiments of mechatronic components by their Pareto-fronts is also supported by MagOpt and was done in [15] for fast switching valves.

Acknowledgment

This work has been carried out at LCM GmbH as part of a K2 project. K2 projects are financed using funding from the Austrian COMET-K2 program. The COMET K2 projects at LCM are supported by the Austrian federal government, the federal state of Upper Austria, the Johannes Kepler University and all of the scientific partners which form part of the K2-COMET Consortium. This work is also based on the research project "Nachhaltig ressourcenschonende elektrische Antriebe durch höchste Energie- und Material-Effizienz" within the EU program "Regionale Wettbewerbsfähigkeit OÖ 2007-2013 (Regio 13)" funded by the ERDF and the Province of Upper Austria.

References

[1] D. Reischl, A. Dorninger, A. Fohler, J. Gerstmayr, W. Koppelstätter, S. Silber, and S. Weitzhofer, "Coupled mechanical and electromagnetic optimization of high speed rotors," in *Proc. 14th International Symposium on Magnetic Bearings*, 2014.

[2] R. G. Ajith Abraham, Lakhmi Jain, *Evolutionary Multiobjective Optimization*. Springer-Verlag London Limited, 2005.

[3] A.-C. Zăvoianu, E. Lughofer, W. Amrhein, and E. P. Klement, "Efficient multi-objective optimization using 2-population cooperative coevolution," 2013, pp. 251–258.

[4] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[5] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, K. Giannakoglou *et al.*, Eds. International Center for Numerical Methods in Engineering (CIMNE), 2002, pp. 95–100.

[6] A.-C. Zăvoianu, E. Lughofer, W. Koppelstätter, G. Weidenholzer, W. Amrhein, and E. P. Klement, "On the performance of master-slave parallelization methods for multi-objective evolutionary algorithms," in *Artificial Intelligence and Soft Computing*, ser. Lecture Notes in Artificial Intelligence, L. R. et al., Ed. Springer Berlin Heidelberg, 2013, vol. 7895, pp. 122–134.

[7] W. Gruber, W. Back, and W. Amrhein, "Design and implementation of a wheel hub motor for an electric scooter," in *Vehicle Power and Propulsion Conference (VPPC), 2011 IEEE*, Sept 2011, pp. 1–6.

[8] G. Bramerdorfer, W. Amrhein, S. Silber, and R. Hagen, "Spectral-field design with respect to minimum cogging torque and maximum output power," in *IECON 2010 - 36th Annual Conference on IEEE Industrial Electronics Society*, Nov 2010, pp. 2264–2269.

[9] G. Bramerdorfer, S. Silber, G. Weidenholzer, and W. Amrhein, "Comprehensive cost optimization study of high-efficiency brushless synchronous machines," in *Electric Machines Drives Conference (IEMDC), 2013 IEEE International*, May 2013, pp. 1126–1131.

[10] R. Lohninger, H. Grabner, G. Weidenholzer, S. Silber, and W. Amrhein, "Modelling, simulation and design of a novel permanent magnetic reluctance machine," in *Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM), 2012 International Symposium on*, June 2012, pp. 620–624.

[11] G. Weidenholzer, S. Silber, G. Jungmayr, G. Bramerdorfer, H. Grabner, and W. Amrhein, "A flux-based PMSM motor model using RBF interpolation for time-stepping simulations," in *Electric Machines Drives Conference (IEMDC), 2013 IEEE International*, May 2013, pp. 1418–1423.

[12] P. Foschum, A. Plöckinger, R. Scheidl, G. Weidenholzer, and B. Winkler, "Multi objective genetic optimization of fast switching valves," 2011, pp. 116–128.

[13] P. Foschum, A. Plöckinger, and R. Scheidl, "Hardware in the loop multi objective genetic optimization for efficient valve control," 2012, pp. 123–132.

[14] E. Marth, G. Jungmayr, J. Kobleder, M. Panholzer, and W. Amrhein, "Realization of a multi-pole permanent magnetic bearing with rotating magnetization," in *Proc. 14th International Symposium on Magnetic Bearings*, 2014.

[15] P. Foschum, A. Plöckinger, G. Weidenholzer, and R. Scheidl, "Triz, design optimization and suh's 1st axion - a comparison on the example of fast switching valves," 2012, pp. 498–505.