IEEE Transactions on Magnetics, Vol. 34, No. 5, Septbember 1998, pp. 2936-2939.

A Parametric Finite Element Environment tuned for Numerical Optimization

Uwe Pahner, Ronny Mertens, Herbert De Gersem, Ronnie Belmans and Kay Hameyer
KATHOLIEKE UNIVERSITEIT LEUVEN, DEPT. E.E. (ESAT), DIV. ELEN
Kardinaal Mercierlaan 94, B-3001 Leuven, BELGIUM

***Abstract*—Nowadays, numerical optimization in combination with finite element (FE) analysis plays an important role in the design of electromagnetic devices. To apply any kind of optimization algorithm, a parametric description of the FE problem is required and the optimization task must be formulated. Most optimization tasks described in the literature, feature either special developed algorithms for a specific optimization task, or extensions to standard finite element packages. Here a 2D parametric FE environment is presented, which is designed to be best suited for numerical optimization while maintaining its general applicability. Attention is paid to the symbolic description of the model, minimized computation time and the user friendly definition of the optimization task.**

***Index terms*—optimization methods, finite element methods, software design/development**

I. INTRODUCTION

The finite element analysis is widely accepted for its general application range regarding geometry and problem type (electromagnetic, thermal, motion, coupled problems). This makes it a desirable tool for the optimization of electromagnetic devices, regardless of its computational expense [1]. An optimization problem is formulated by defining an objective (quality) function with a number of design parameters as the variables. The purpose of an optimization is to find the best possible solution to a given problem by the simultaneous variation of the design variables. This requires many objective function evaluations. Two main streams in the combination of FE analysis with different optimization algorithms can be pointed out:

- the development of computer codes that are designed to solve a specific optimization problem as efficient as possible (accepting the loss of the general applicability range of the final code) [2],
- the development of add-on tools to standard finite element packages that are originally not designed to perform repetitive analyses [3].

This paper focuses on the development of the 2D FE code *OLYMPOS*, which is user-friendly, optimized for repetitive analysis while still offering the general application range. *OLYMPOS* solves electrostatic, magnetotstatic, thermal, time-harmonic and motion problems in 2D.

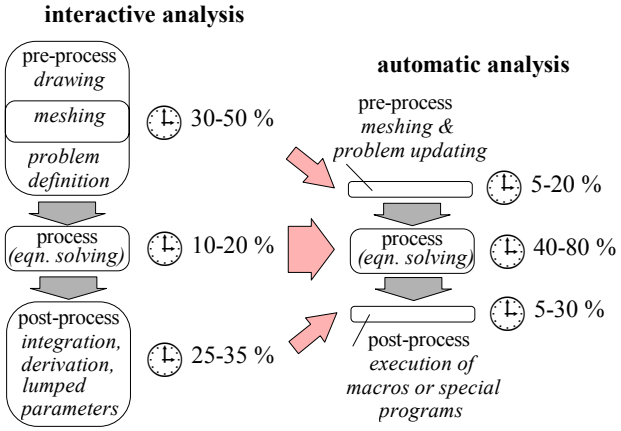

Fig. 1. Schematic of relative times for interactive and automatic FEM analysis runs.

II. SYMBOLIC DESCRIPTION OF FINITE ELEMENT MODELS

To allow a repetitive analysis of a FE problem, a thorough symbolic description is a pre-requisite. Geometry, excitations as well as material data are defined symbolically during the first and only interactive pre-processing (fig. 1). This interactive session is fundamentally different from the completely automatic pre-processing invoked during an optimization or parameter variation. In the interactive session, the user defines the model symbolically, while this definition is only updated and translated during the automatic pre-processing. This update (changing of parameter values), could be caused by either a parameter variation loop, an optimization algorithm or the user. No additional preparation is required to perform either of the three tasks. All parametrization in *OLYMPOS* is based upon a ordered list of parameters, defined interactively by the user in the pre-processor (fig. 2). A subset may later be chosen as the design variables in an optimization task. This list may contain dependent or independent parameters. The dependent parameters are defined by a symbolic expression, that may contain previously listed parameters.

The syntax of these definitions is *Matlab*-like, entered by the user or via macro-commands. Thorough parametrization in terms of FE analysis includes the possible symbolic

```
-name---|----formulation--------
      angle 5
      phase 30.0
    current 11
      curr1 current*sqrt(2)*cos(rad(phase+2*angle))
```

Fig. 2. Ordered list of parameters. An unlimited number of dependent or independent parameters are defined symbolically.

Manuscript received March 31, 1997
Uwe Pahner, +32 (0)16 321020, fax +32 (0)16 321985, uwe@esat.kuleuven.ac.be, http://www.esat.kuleuven.ac.be/elen/elen.html
The authors are grateful to the Belgian "Fonds voor Wetenschappelijk Onderzoek Vlaanderen" for its financial support of this work and the Belgian Ministry of Scientific Research for granting the IUAP No. P4/20 on Coupled Problems in Electromagnetic Systems. The research Council of the K.U.Leuven supports the basic numerical research.

description of boundary conditions, external electric circuits, initial discretization on geometrical boundaries, parameter consistency checks, solver settings and post-processor computations. The parametrization of the model is only the first step during the set-up of an optimization task. Model or optimization constraints have to be specified as well. Depending on the type of constraint, this might be implemented using the penalty method (fig. 3), or if a violation of constraint cannot be permitted, via rejection of the model (fig. 4). The value of the penalty term is added to the value of the objective function in case of an optimization run. The execution of a regular analysis or parameter variation analysis is not influenced. A violation of the model constraints however, leads to the rejection of the model or the stop of the analysis. The definition of constraints and penalties is also based upon the ordered list of parameters.

The variety of optimization tasks is not only characterized by their problem types, geometries, excitations and constraints, but also by the different analysis steps that are required to investigate the specific problem. In *OLYMPOS*, a special feature is the definition of *named analysis procedures* (fig. 5), that define the sequence of steps towards the desired result in case of a non-standard analysis. The concept is similar to a system command batch. But here, the set of commands can include a mixture of pre-processor actions and bi-directional communication between the pre-processor (and optimization controller) and any other program. Weak coupled problems, such as time-harmonic/thermal can be defined. The symbolic problem and analysis description is stored in ASCII format. This allows access to all information if it is intended to link other routines or codes into the analysis process (e.g. specialized post-processor tools). The problem file does not contain information about the FE nodes, elements etc. This ensures low storage requirements. The approach inherits another advantage: the management and organization of different projects is simplified, as all project data are concentrated in one ASCII-file, and different analysis tasks are defined once, to be available and repeatable at any time. Application specific shells (as available for standard FE-packages) can be replaced by application specific analysis procedures.

III. Acceleration of The Finite Element Analysis

```
BEGIN PENALTY
 ((p1<=0.1)||(p1>3.5)) exp(abs((p1-1.5)/1.5))
 p2>2e-3 3.0
END PENALTY
```

Fig. 3. Example of two penalty expression, consisting of the logic expression followed by the function value to add if this expression is TRUE.

```
BEGIN MODEL CONSTRAINTS CHECK
 lb1>=1.0
 lb1<=5.0
 ((lb3<1.0)||(lb3>0.0)&&(sin(rad(angle))<0.5))
 ((ra-lb2)^2)>((sm/2)^2+(ra-lv)^2)
END MODEL CONSTRAINTS CHECK
```

Fig. 4. Example of the syntax for the definition of model constraints, as they are stored in the parametrized model and sketch file (PSK-file).

```
PROCEDURE TORQ
 6
 1 save
 2 meshprob -nd PROJ1.PSK
 3 arachne -nd PROJ1.
 4 cp PROJ1.MX MAXWELL.DAT
 5 maxwell -nd SOLU.TXT PROJ1.PSK
 6 extract data
```

Fig. 5. Example of named analysis procedure (TORQ) as it is stored in the PSK-file. There is no implemented limit in the number of procedures, or steps in a procedure. In this case, the updated model is saved (PROJ1.PSK), the mesh generated, the problem translated, solved and a force computation is performed. Finally, the results are read into the pre-processor.

Mesh generation and solution process are automated. The accuracy and rate of convergence of the FE analysis are strongly depending on the discretization of the model and the appropriate solution algorithm of the system of equations.

Mesh generation and problem translation

Fast and reliable mesh generation is a key requirement when linking FE analysis and numerical optimization. During the development of *OLYMPOS*, the choice was made to implement a fast but minimum Delaunay triangulation (fig. 6). This choice requires a fast and reliable adaptive refinement algorithm to be implemented in the solver modules. In an automatic execution of a named analysis procedure, the obvious first step after changing the value of a parameter is saving the updated symbolic description of the model. In the second step, a triangulation of the model geometry is constructed. These steps could be called the *translation* of the symbolic description of the geometry into a numerically fixed discretization for this particular design. The most convenient and time-saving approach is to link this step with the *translation* of the symbolic problem description into the data structures based upon the newly constructed discretization. All data in the parametrized problem & sketch file are treated initially as a symbolic expression. A parser evaluates the expressions during the input stage. Most data are immediately stored in their numeric form. Parsing is rather time consuming, which leads to the fact that the parsing operations usually requires most of the execution time of a single pre-processing run (see table I and II).

Fig. 6. Initial Discretization of a permanent magnet machine due to two different sets of design parameters. Model a) includes 1737 nodes, 3334 elements, total automatic pre-processing time 1.07 s on a HP C160. Model b).with 5 parameters changed (magnet width, height and burry depth, iron bridge length, rotor angle), requires 1.04 s for 1665 nodes and 3214 elements.

a) b)

Table I*.
Initial problem translation time for a number of FE-models

nb	n_{para}	n_{exp}	n_{char}	t_{trans1} [s]	t_{prep} [s]
1	6	21	156	0.03	<0.01
2	15	27	257	0.03	<0.01
3	50	55	882	0.42	0.05
4	13	397	17404	0.64	0.06
5	0	0	0	0.10	0.47
6	60	242	11619	1.35	0.03
7	50	326	17442	1.74	0.04

- all timings taken on HP C160
-

Table II.
Triangulation, refinement and saving times

nb	t_{triag} [s]	n_{node} initial	n_{elem} initial	t_{ref} [s]	n_{node} final	n_{elem} final	t_{save} [s]	t_t [s]
1	<0.01	148	214	0.04	361	640	0.03	0.11
2	<0.01	123	210	0.03	334	632	0.03	0.11
3	0.08	635	1130	0.16	1737	3334	0.37	1.06
4	0.07	641	1165	0.16	1779	3441	0.35	1.23
5	0.19	1464	2890	0.40	4273	8508	0.40	1.46
6	0.05	461	813	0.11	1248	2387	0.20	1.73
7	0.08	688	1242	0.18	1913	3692	0.27	2.20

n_{para} - the number of parameters in the ordered list,
n_{exp} - the number of symbolic expressions,
n_{char} - the total number of characters of the symbolic expressions,
n_{node} - the number of nodes,
n_{elem} - the number of elements,
t_{trans1} - the processor time for the data input and initial translation,
t_{prep} - the processor time for the preparation of the PSLG, including the removal of the duplicate points,
t_{triag} - the processor time for the initial triangulation (incl. insertion of outline segments and spreading of region labels)
t_{ref} - the processor time for the first mesh refinement,
t_{save} - the processor time for the final translation of the problem and the saving of all data,
t_{elaps} - the elapsed processor time for the automatic pre-processing.

The input for the triangulation algorithm [4] is a *planar straight line graph* (PSLG), which is a non-ordered list of vertices and segments. This PSLG is built from the symbolic description of the geometry, including the parametrized information for the subdivisions along the lines. Each primitive is converted into a list of segments, depending on the number of subdivisions applied. The list of segments is in fact the list of element edges along geometrical primitives. The triangulation, constructed using the divide-and-conquer method in the implementation described by [4], has a runtime of *O(n log n)*. This is a pre-requisite for the desired speed of the full analysis. This algorithm is very robust, which is supported by the organization of the data sets and the use of exact arithmetic. However, not all the PSLG-segments will be represented in this first triangulation. The remaining segments have to be inserted in a following step, leading to the *constrained Delaunay* triangulation which is the initial discretization taken for *OLYMPOS*.

Region labels are assigned using a kind of "virus spreading"-algorithm, which does not require any prior knowledge about the construction of closed regions. The initial triangulation can be followed by a first (geometrical) refinement step. Finally, a second step of the *translation* of the symbolic problem description is required, as some information is only available after the mesh generation :

- the connectivity of the paired constraints,
- the exact cross section of the regions,
- most excitations are transformed from their integral value into densities,
- special types of boundary conditions must be prepared (e.g. cuts).

This final step of *translation* if followed by the saving of the finite element data set. The FE-problem file contains an discretized representation of the symbolic description defined by the present value of the set of parameters. It is often impossible to determine a good discretization of the model before starting the optimization, as the optimized shape of the device can be totally different. A fast and reliable mesh refinement, in combination with various *a posteriori* error estimators is implemented in *OLYMPOS* to control this problem.

Mesh Refinement

A sequence of solution and refinement steps is performed until the desired accuracy is reached. The error estimators based on interpolation theory can be applied for different regions in the model. The intersection of the elements is followed by a quality enhancement of the discretization. Nodes are moved towards the center of gravity of the surrounding nodes and a local Delaunay edge swapping algorithm is applied. This assures an average aspect ratio of the elements close to unity (typical < 1.2) in the final discretization. Particular attention is paid to fast refinement algorithms. Error estimation and refinement requires less than 10% of the overall computation time (fig. 7).

Solution of the System of Equations

The choice of the appropriate solution algorithm depends on the properties of the coefficient matrix, which in term depends on the problem type. Several pre-conditioners can be combined with either CG (Conjugate Gradient) or BiCG (Bi-directional Conjugate Gradient): SOR (Successive Over-Relaxation), SSOR (Symmetric Successive Over-Relaxation),

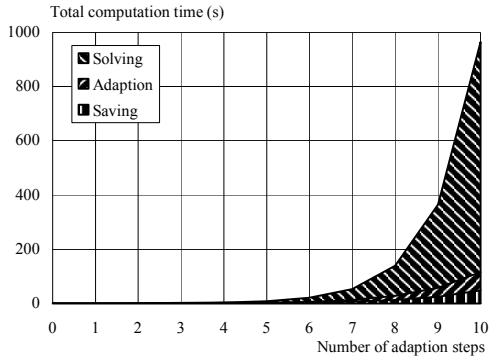

Fig. 7. Relative execution times for different steps of the solution stage. It must be noted that the time for saving the data is of the same order as the time for mesh refinement.

IC (Incomplete Cholesky Decomposition) and AMG (Algebraic Multigrid). The Algebraic Multigrid pre-conditioner is especially well suited for very large systems of equations, as it reduces the computation time significantly (up to 6 times) compared to classical pre-conditioned CG methods [5]. After the mesh refinement, all solvers start with an interpolated solution at the newly imposed nodes. In a non-linear analysis, the number of Newton steps can be reduced down to 25% of the steps with the initial mesh.

IV. Definition and Execution of Optimization Tasks

The pre-processor of the FE package provides all tools for the set-up of an optimization task. Most optimization problems are constrained, often feature multiple objectives and are expected to find the global optimum. A variety of direct search algorithms (Hooke&Jeeves, evolution strategy, genetic algorithms, Simulated Annealing) are chosen, due to their simplicity in the preparation of this type of problem [6]. New developments in evolutionary algorithms, which improve the global convergence speed of the optimization such as Differential Evolution [7] support this choice. The optimization algorithms are implemented in an external optimizer (fig. 8). The three reasons for this separation are:

- Non-FE optimizations are performable.
- The parallel set-up of the optimization task is simplified.
- Easy addition of optimization algorithms.

Based on the symbolic description of the model, the named analysis procedures, penalty and model constraints checks, the definition of the optimization task is unified. The design parameters are selected by their name from the ordered list of parameters. While the user defines design variables in their physical units, the optimizer requires normalized variables. The pre-processor allows to define a normalization and de-normalization function for each design variable. During the optimization, the design objectives are calculated for each intermediate design (local field values, forces, inductances, volumes etc.). They have to be combined to return a scalar valued quality, describing the acceptance of the design in terms of the objectives. As every optimization task might involve non-standard calculations, especially during the post-processing, a general data transfer algorithm is provided. It allows the extraction of data from external ASCII files (e.g. LOG-files) into user defined variables inside the FE-package. It is now possible to formulate a single valued quality function (e.g. weighted sum) in the ordered parameter list, using the same syntax as in the definition of the model. The execution of the optimization is performed either from within a graphical interface, or more efficient in background, involving no graphical output. A stand-alone program allows to monitor the progress of the optimization at run-time. The optimization can safely be stopped and restarted at any time. This feature is required, if settings of the optimization strategy have to be adjusted in the starting phase of an optimization or hardware (network) maintenance requires the interruption of the process.

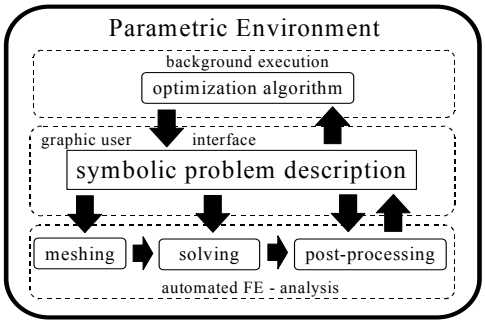

Fig. 8: All analysis and optimization steps are defined by the symbolic description of the problem.

V. Conclusion

A powerful 2D FE-package is presented which is tuned for numerical optimization. Emphasis is put on the combination of various measures to accelerate the FE-analysis, while remaining its applicability to different types of field and optimization problems. A thorough symbolic description of the problem serves as the basis for the optimization of the device. The concept of incorporating the definition of the analysis procedure into the symbolic description of the model inherits the potential of the simple development of application specific tools without the need for newly coded shells.

References

[1] K. Hameyer,"Numerical Optimization of Finite Element Models in Electromagnetics with Global Evolution Strategy", Adaptive Computing in Engineering Control, ed. by I.C.Parmee, Plymouth, 1994, pp.61-66.

[2] I.H. Park, H.B. Lee, I.G. Kwak, S.Y. Hahn, "Design Sensitivity Analysis for Steady State Eddy Current Problems by Continuum Approach" *IEEE Transactions on Magnetics*, vol. 30, no. 5, September 1994, pp. 3411-3414.

[3] B. Trowbridge, "The Role of Optimisation Techniques used for Magnet Design", Proceedings of the 3rd International Workshop on Electric and Magnetic Fields, Liege, Belgium, May 6-9 1996, pp. 15-23.

[4] J.R. Shewchuk, "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator", 1st Workshop on Applied Computational Geometry, Philadelphia, USA, May 1996, pp. 124-133.

[5] J.Ruge and K. Stueben, *Multigrid Methods*, SIAM Philadelphia, 1987.

[6] O.A. Mohammed, "Stochastic Design in Applied Electromagnetics…The Genetic Algorithms Approach and System Optimization Strategies.", Newsletter of the International Compumag Society, vol. 4, no. 2., 1997, pp.5-10.

[7] R. Storn and K. Price,"Minimizing the real functions of the ICEC'96 contest by Differential Evolution", IEEE Conference on Evolutionary Computation, Nagoya, Japan, 1996, pp. 842-844.