European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS 2000
Barcelona, 11-14 September 2000
© ECCOMAS

MULTIDISCIPLINARY OPTIMIZATION IN TURBOMACHINERY DESIGN

Rolf Dornberger[*], Dirk Büche[*] and Peter Stoll[*]

[*] ABB ALSTOM POWER Technology Ltd
CH-5405 Baden-Dättwil
Switzerland
e-mail: rolf.dornberger@ch.abb.com

Key words: turbomachinery optimization, multi-objective, multidisciplinary, optimization environment, Pareto-optimization, response surfaces, neural networks, polynomial approximations, preliminary design, 3D blade design.

Abstract. *This paper investigates particular aspects of performing multidisciplinary optimizations in different turbomachinery design steps. The differences in the optimization approaches and used methods in preliminary and final design steps are shown. An optimization environment is developed, which supports multidisciplinary turbomachinery design. The general concept and components are explained. Some of the implemented methods and tools, which are used particularly in multidisciplinary optimization, are presented. Examples show the potential of the proposed methods. Pareto-optimization enables multi-objective optimizations, very important in preliminary design steps. Various objectives are optimized concurrently. An entire set of Pareto-optimal solutions, design variants, can be computed. Response surfaces promise to accelerate the entire design process. They are able to approximate computational expensive solvers within a fraction of their original computing time. Two different schemes, polynomial approximations and neural networks, are presented. Using different examples, both methods are compared. In final turbomachinery design steps, 3D blade design optimizations need quickly converging optimization algorithms treating one single aerodynamic objective. The other involved disciplines lead to further constraints. Extensions for using the optimization environment in such 3D blade optimizations are presented.*

1 INTRODUCTION

In turbomachinery design, the future trend in multidisciplinary design optimization (MDO) is to cover not only aerodynamic and thermodynamic performance of turbines and compressors, but also geometrical requirements, mechanical integrity and manufacturing costs[1]. Life cycle costs, product cycle time, weight, emissions and heat transfer[2] are additional, possible criteria during the design optimization process.

Starting a new design, basic preliminary design steps calculate the mean dimensions of the machine. In this conception phase, the largest influence on the final product is made. The mean diameters and the number of turbomachinery stages are fixed for instance. The expected performance, i.e. efficiency and cost, are approximately calculated and the mechanical integrity is estimated. Deciding on particular variants and continuing the design process, the turbomachinery is designed more precisely using increasingly more complex design tools. For example, the flow path, blade sections and channel contouring are evolved in this phase. While details of the turbomachinery design are settling, the influence on fundamental changes of the entire turbomachinery concept decreases. At the end, a fine-tuning of the 3D design though allows an additional increase in the performance of the turbomachinery at a very high-end level, but the concept cannot be changed anymore.

Hence, as the tasks of the designers are differing from one design step to the next, strategies and methods for optimizing the design must change as well. In this paper, different optimization algorithms and strategies are shown and explained and are used depending on the turbomachinery design step. Emphasis has been put on the optimization methods used in first design steps. In contradiction, an approach applied in the final design phase is presented in order to demonstrate the difference in the optimization approaches and used methods.

2 CONCEPT OF OPTIMIZATION ENVIRONMENT

In order to combine design tools with optimization methods and to run them in an automated way, an optimization environment has been developed[3], as shown in Figure 1. Thus, this optimization environment simplifies the use of various optimization and evaluation methods, of which the most suitable one can be taken for specific design tasks.

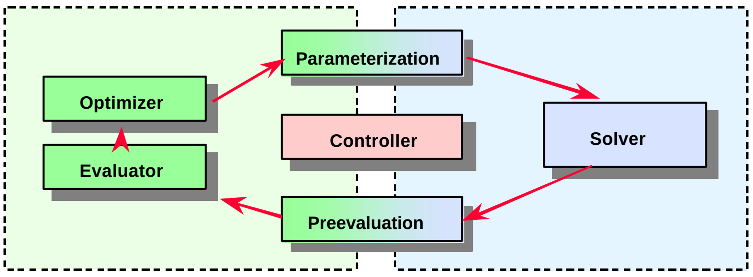

Figure 1: Optimization environment

Besides the choice of suitable optimization and evaluation methods, a sophisticated parameterization supports significantly the optimization process. If basic models are used, the input of the corresponding solver has not to pass a parameterization in advance. This changes, when the solvers with the underlying models become more complex. For example, 3D optimization today provides the possibility of improved designs, where 1D, 2D and Q3D methods are already drained[4]. Consequently, it is very promising to couple optimization tools with 3D CFD codes for performing entire 3D design optimizations. In case of 3D design optimization[5], the parameterization must execute several other tools successively, e.g. tools for generating and modifying geometry models, and meshing tools for creating computational grids. The parameterization helps to reduce the number of design variables generated by the optimizer, although the total number of solver input parameters describing the entire design geometry is very high. Additionally, design expertise can be stored in the parameterization, allowing only reasonable and desired modifications of the geometry. Particularly in turbomachinery blade design, an additional benefit of the parameterization lies in the possibility of modifying complex three-dimensional blades with only a few key parameters within desired ranges.

After the parameterization, different solvers are executed, which compute all objectives and constraints that are relevant to the multidisciplinary design. Particular solvers can run mostly in parallel, but sometimes only sequentially. The output of one solver is partly needed as input for another solver, like in the computation of structural behavior under pressure load. Another aspect of parallelization is given when optimization algorithms are able to produce concurrently more than one solution. In that case, the entire optimization is accelerated significantly, if the implementation supports parallel execution of a solver on different processors.

In order to approximate quickly the results of computational expensive solvers, response surfaces are used promising a substantial reduction in computation time. Different schemes based on neural networks and polynomial approximations have been developed and tested. The response surfaces are trained during the optimization. If they are able to approximate the solution space sufficiently, the optimization is continued using the approximated results. After the optimum has been found in the response surface model, the original solvers are used again, in order to find the real optimum and not the approximated one.

A preevaluation tool processes the output of the solvers. It reduces the data to pure objectives and constraints, and adds the respective ranges to all constraints. If the objectives and constraints are not explicitly contained in the solver output, the preevaluation tool allows calculating them using additional post-processing tools. For example, CFD predicts the 3D flow field through turbomachinery stages. Then the preevaluation tool might calculate the deviation from the flow uniformity as desired design objective. Expertise with respect to assessment criteria of designing turbomachinery can be stored in the preevaluation tool, i.e. properties or design features and their related effects, necessary for designing best turbomachinery.

A controller supervises all components of the optimization loop in order to start the entire optimization process, to keep it running, and to terminate it, when appropriate stopping criteria are fulfilled. Thus, the controller sequentially executes succeeding tools, as soon as the results from the preceding ones are available. Due to distributed computations, it also has to re-compute and/or skip any failing solver computation in order to save the entire optimization process.

However, a design optimization environment can never become a black box. Only if very basic models are applied, the hope of producing excellent stand-alone results without any interaction may become true. In most cases, the optimization process needs direct interaction with the designers. Their design knowledge has to guide the optimizer in the intended direction. This starts with setting up the problem and defining objectives and constraints. Performing sensitivity analyses, combining different disciplines, and interpreting results are the most important tasks, which are necessary for designing the best turbomachinery. Particularly in the more complex design phases, the designers learn the important parameters in turbomachinery design, as well as how to modify particular design features evaluating many different design variants automatically. Thus, the large analyzing capabilities of an optimization environment finally lead to a significant reduction in design time and increase in design quality.

3 PRELIMINARY TURBOMACHINERY DESIGN OPTIMIZATION

3.1 Preliminary Turbomachinery Design

In general, preliminary turbomachinery design starts with a basic configuration program for estimating thermodynamic efficiency. The turbomachinery, i.e. a turbine or compressor, is idealized by a certain number of fictional stages. These stages are extrapolated from a mean stage, in order to fulfill the entire thermodynamic transport equations throughout the turbomachinery. As an example, a sketch of a turbine is shown in Figure 2. At roughly the middle of the turbine the mean stage, consisting of stator and rotor row, can be found. The middle can be defined as the location of the average pressure or the location of half entropy drop over the turbine. The other turbine stages are then extrapolated from this mean stage with respect to varying geometrical and thermodynamic conditions at inlet and outlet. Depending on the formulation, turbine design variables might be mean diameter of the flow path, flow coefficient, loading coefficient, degree of reaction, and chord length of stator and rotor.

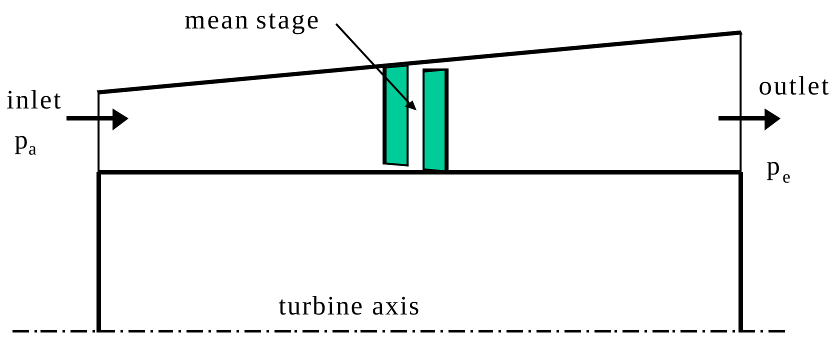

Figure 2: Idealized turbine with the mean stage

In general, such a preliminary design model consists of different parts. In one part, the aerodynamic flow is estimated. In another part, the losses are calculated. As flow and losses depend on each other, the iteration over both leads to an equal balance between them and to the total losses of the turbomachinery. Different effects contribute to the total losses of a turbomachinery. For example, in a steam turbine, profile loss, secondary loss, and leakage loss are the main factors[4]. Profile losses arise due to the boundary layer of the flow along the blade surface. Secondary losses are mainly caused by the vortices produced in the turbulent flow. Leakage losses are caused by the mass flow through the labyrinth sealing at the tip of the blades, and the mixing of the leakage flow with the main flow. When the total losses are computed, the efficiency of the turbomachinery is given, measured in percentage of the ideal isentropic turbomachinery process.

In this preliminary stage, the costs of the turbomachinery can be estimated using more or less sophisticated cost models. For example, a basic approach is counting the estimated number of blades in all turbomachinery rows, assuming different cost factors for different rows, and adding up all the cost contributions including additional costs for hub and casing.

Even in the preliminary design phase, the thermodynamic efficiency and cost of the turbomachinery are not the only concern of the designers. Further models are desired for the manufacturing aspects, mechanical integrity and lifetime prediction.

At the end, different models deliver different objectives and constraints. Particular strategies are necessary to solve such multidisciplinary and multi-objective problems.

3.2 Multi-objective Pareto-Optimization

Each real engineering design consists of many design variables and constraints, while the number of objectives is normally very limited. Particularly, multidisciplinary optimization often leads to complex multi-objective optimization problems. Multi-objective optimization is defined as optimizing several objectives from different engineering disciplines, which are simultaneously addressed and included within one single optimization process.

Nevertheless, the definition of the objectives and constraints is always the most crucial point in the design and demands a high level of design experience. In order to perform such a multi-objective optimization, special assessment schemes are necessary. Pareto-optimization has been found superior, as it delivers sets of Pareto-optimal designs. No other design is concurrently better in all objectives. However, as the computing time can be a limiting factor particularly in following design phases, such extensive optimizations are only reasonable in conjunction with fast solver computations and/or response surfaces (see chapter 3.3).

A special Pareto-optimization technique is developed applying a genetic algorithm with the subdivision method[6]. It performs well and is capable of finding a large number of good multi-objective solutions with remarkable convergence speed. The idea behind the subdivision methods is that different design objectives pose different demands to the turbomachinery properties. Often, these demands compete against each other. Single objective optimization may lead to solutions that are not acceptable for other design objectives. However, a large number of good, but distinguished solutions should always be found, allowing the designers to choose the solution that best fits their demands and is the most efficient for them.

The multi-objective optimization is often performed with a weighted sum method, combining all objectives in one single resulting objective function. This optimization process leads just to special kind of solution with a strong dependence on the weighting coefficients. However, the subdivision method is a particular extension of Pareto-optimization and creates a large number of different solutions with the aim of covering the whole solution space.

For example, the aim of a turbine optimization could be the simultaneous optimization of three main design objectives, such as efficiency, blade stress and manufacturing costs of the turbine. The process of such a preliminary turbine optimization is shown in Figure 3. It consists of five main components. The *Optimizer* proposes sets of design variables. These variables are input data for the *Thermodynamic Turbine Model*, which computes the efficiency of the turbine. It also computes values about the shape of the turbine, e.g. its mean length and diameter, and values about the aerodynamic flow, e.g. the pressure drop over a stage, the Mach number and the flow angles. These results are necessary inputs for the *Mechanical Integrity* model, which computes the stresses and fundamental frequencies of the turbine blades. Parallel to this model, a *First Cost* model estimates the manufacturing costs of the turbine using the computed key parameters of the turbine shape, number of stages and blades per row. All results of the above models are then evaluated by the *Evaluator*, which is the last component in the loop. The *Evaluator* additionally checks if constraints are violated, e.g. if the Mach number in the turbine is too high. The *Optimizer* finally gets from the *Evaluator* a feedback about the quality of the proposed design. After that, it suggests a new solution with slightly different parameters, until a termination criterion is reached.

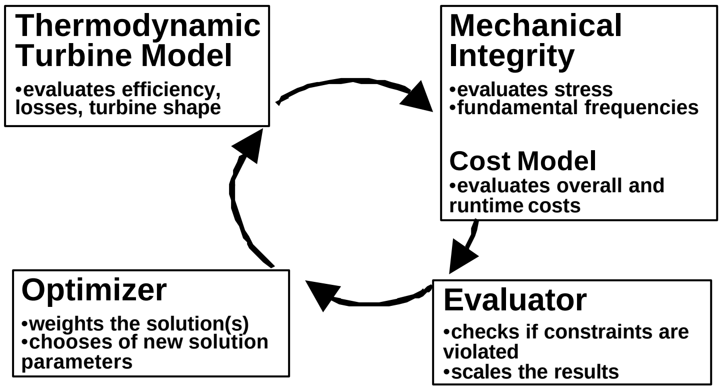
Thermodynamic Turbine Model
•evaluates efficiency, losses, turbine shape
Mechanical Integrity
•evaluates stress
•fundamental frequencies
Cost Model
•evaluates overall and runtime costs
Optimizer
•weights the solution(s)
•chooses of new solution parameters
Evaluator
•checks if constraints are violated
•scales the results

Figure 3: Process of a preliminary turbine optimization.

3.3 Example of Preliminary Design Optimization

As an example in preliminary design optimization, multidisciplinary optimization of a turbine is shown. The aim is to design a high-pressure steam turbine with high thermodynamic efficiency, reasonable structural loading and low initial costs. Thus, a designer may describe his needs as a relationship between these three competing objectives.

In preliminary design optimization, the design tools are mostly computer programs using minimal computing time. In this case, it is possible to generate large numbers of different designs, which are a prerequisite for a complete Pareto-optimization. The optimization loop was already presented in Figure 3. For the optimization algorithm, we apply a genetic algorithm with the fundamental properties of generating Pareto-solutions by crossover and mutation, where the subdivision method enables parallel convergence towards the Pareto-front of ideal solutions using a certain number of individuals. While gradient methods are limited to a single optimal convergence, our genetic algorithm is able to converge against the whole Pareto-front concurrently.

The result of one single optimization run is given in Figure 4. The genetic algorithm finds a three dimensional surface of ideal solutions defined by the 3 objectives: thermodynamic efficiency factor, cost factor and mechanical (von Mises) stress factor. Of course, the higher the efficiency the higher the costs as well as the mechanical stresses. However, due to the quantification of the underlying relations, the designers are able to select this turbine design, which fits best to their needs with respect to efficiency, costs and structural aspects.

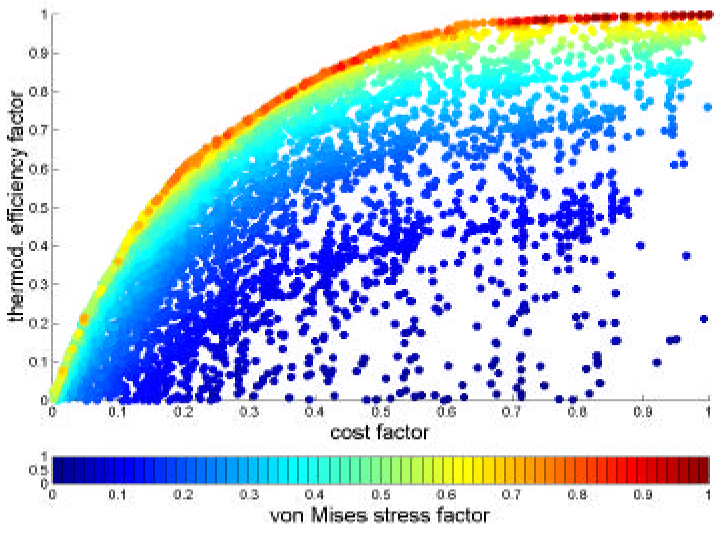

Figure 4: Pareto-front of ideal solutions, generated in preliminary multidisciplinary turbine optimization

4 RESPONSE SURFACES

When the computing time of the applied solvers is mostly small in preliminary design steps (about milliseconds to seconds), an entire Pareto-optimization might be a good choice. However, when the design gets more detailed, and thus the applied solvers become more complex, their computing time rises significantly. In this case, the number of possible optimization iterations is the limiting factor in the entire design. Hence, after converging to one single optimum, the entire optimization process must often be stopped, even if the global optimum or a good Pareto-set is not found.

In order to overcome this problem of spending too much design time with computational expensive solvers, the use of response surfaces promises to accelerate the entire design process. Based on sets of computed solutions, response surfaces are trained using approximations schemes, which are able to approximate results of solvers within a fraction of the original computing time. Response surfaces are thus well suited for guiding optimization processes to the extremes of the solution space. However, as the extremes of the response surfaces are only approximations of the solver results, the original solvers are applied again afterwards, using now the approximated extremes as good initial guesses.

Different methods exist for implementing response surfaces. Mainly, polynomial and neural network schemes are used. Polynomial schemes are effective if polynomial function approximations have to be performed. As the accuracy of such schemes decreases rapidly, when the complexity of the solver models increases, a reasonable balance has to be found between the order of the polynomials, the desired accuracy and the training time. For more complex solver models, neural networks are often more universal and more accurate, as well as their training rates are higher. Nevertheless, the choice of the best-suited neural network schemes and configurations for each new solver is not always evident.

In the next chapter, multidimensional polynomials and neural networks are presented and applied as response surfaces. Their general behavior is shown afterwards using a simple mathematical example. Their practical use is demonstrated approximating the results of the preliminary design tool of chapter 3.

4.1 Polynomial and Neural Network Approximations

As polynomial approximation scheme, rational higher order polynomials of the type

$$y(x_1, x_2, \cdots, x_p) = c_0 + \sum_{1 \le i_1 \le p} c_{i_1} x_{i_1} + \sum_{1 \le i_1 \le i_2 \le p} c_{i_1 i_2} x_{i_1} x_{i_2} + \cdots + \sum_{1 \le i_1 \le i_2 \le \cdots \le i_n \le p} c_{i_1 \ldots i_n} \prod_{j=1}^{n} x_{i_j} \qquad (1)$$

are applied. These polynomials are trained with the input data sets $\mathbf{x}$ and the corresponding result data sets $\mathbf{y}$. The maximal order of the polynomials is limited to p. Using least-squares minimization, the coefficients c_{ij} are adjusted within each possible combination of the polynomials in order to find the polynomial combination, which delivers the smallest errors between the original results $\mathbf{y}$ and the approximated results $\tilde{\mathbf{y}}$.

Using this approach for approximating pure polynomial equations $f = y(x_1, x_2, \cdots, x_q)$ of the order q, the exact equations f and their coefficients c_{ij} can be retrieved, as long as $p \ge q$. However, large approximation errors occur, if the maximal order of the polynomials is chosen too small ($p < q$). This could occur if an exponential equation ($q \to \infty$) is approximated. In this case, the results rather tend to oscillate, particularly at the boundaries of the solution space, than to produce smooth solutions.

Neural networks are another possibility for generating response surfaces, particularly if very complex models (e.g. containing exponential functions) have to be approximated. Mostly feed-forward neural networks with multi-layer perceptrons are proposed for this aim in literature. However, experiences and opinions vary significantly about the right structure of the neural networks[7].

In general, the neural network consists of layers of many neurons, as shown in Figure 5.

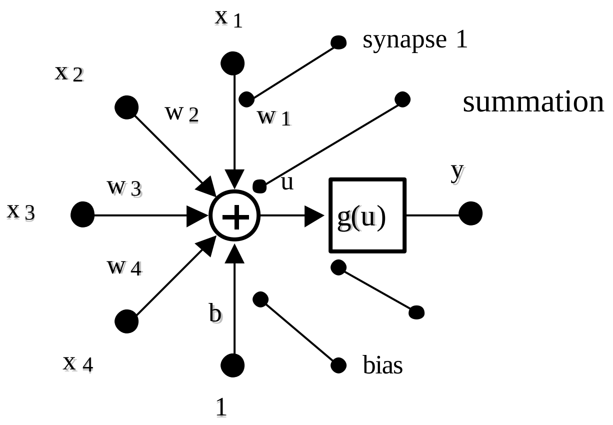

Figure 5: Principle of a neuron

The neurons are coupled with synapses, which amplify or damp the transmitted signals. The factors, which change the input signals $(x_1, x_{2,} \ldots, x_p)$ of the synapses, are given by the weights $(w_{1,} w_2, \ldots, w_p)$. All weighted signals are summed up to an entire signal. Depending on the type of the neuron, an additional bias allows adjusting the entire signal u. This signal passes a particular activation function $g(u)$. The output signal is finally defined as

$$y = g\left(\sum_{1}^{p} w_i x_i + b \right) \tag{2}$$

Connecting several neurons parallel and in sequence into different layers, the entire neural network structure is created. In order to train the neural networks, all weights and biases have to be adjusted using adaptive algorithms, until the relation $\tilde{\mathbf{y}} = f(\mathbf{x})$ fits best the solver output data $\mathbf{y}$.

Although two-layer neural networks are already sufficient to approximate functions, the use of further layers reduces the number of necessary neurons per layer for reaching the same accuracy. Furthermore, the number of training iterations can be reduced significantly in this case. In order to find the best neural network structure with respect to approximation and training properties, different neural network combinations and training algorithms have to be tested in advance.

4.2 Examples of applying response surfaces in turbomachinery design

The general characteristics of the polynomial approximation and neural network schemes shall be demonstrated on the following example equation:

$$f(x_1,x_2) = 3\cdot(1-x_1)^2\cdot\exp\left(-x_1^2-(x_2+1)^2\right) \\ -10\cdot\left(1/5\cdot x_1 - x_1^3 - x_2^5\right)\cdot\exp(-x_1^2-x_2^2) - 1/3\cdot\exp\left(-(x_1+1)^2-x_2^2\right) \tag{3}$$

This function spans a solutions space with 3 maximums and 3 minimums within the area $x_1 = [-2,2]$ and $x_2 = [-2.5,2.5]$, as shown in Figure 6.

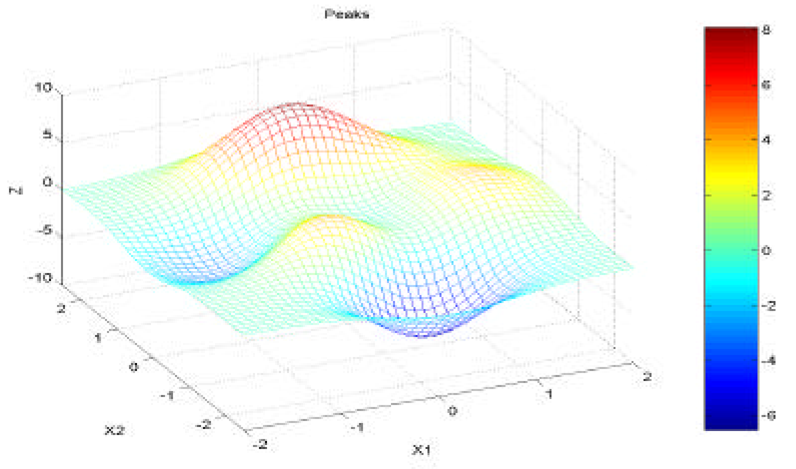

Figure 6: Solution space of the exponential test function

Using a data set of 200 randomly chosen sample points, the polynomial approximation of order $p = 10$ is shown in Figure 7. The red points are the sample points.

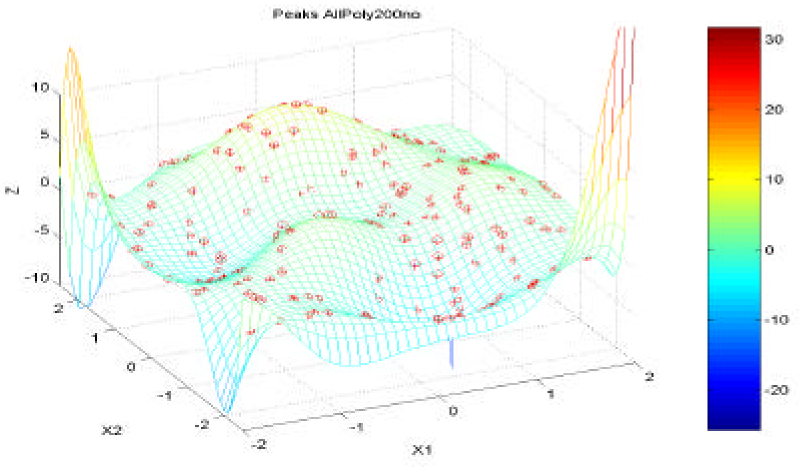

Figure 7: Trained response surface using polynomial approximations

While all three (inner) maximums and minima are represented quite well, the boundaries of the solution space are oscillating so widely, that new extremes are generated with much larger values than the original inner extremes. Increasing the order of the polynomials decreases this effect, but the boundaries never get completely smooth.

In contrast, the same data set of 200 sample points is used for training a response surface consisting of a two-layer feed-forward neural network. This response surface is able to reconstruct the original solution space very well, as shown in Figure 8. All extremes lie near their original positions with a maximal error of less than 1%. Furthermore, the boundaries do not oscillate to the degree of the polynomial approximations.

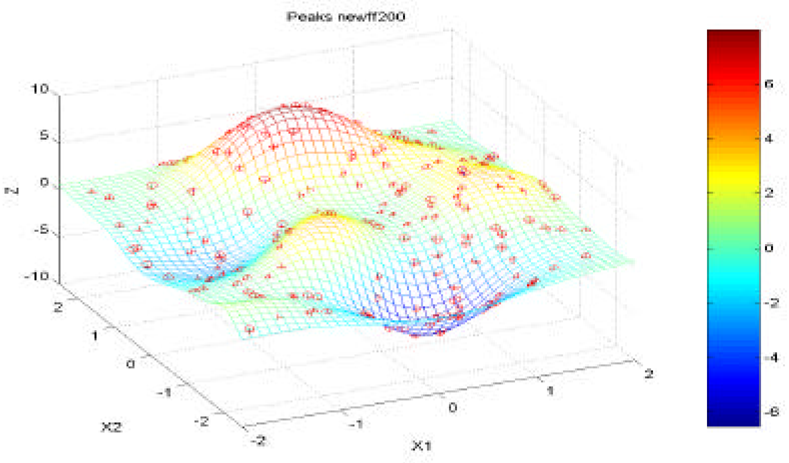

Figure 8: Trained response surface using neural networks

However, the training time of the neural network is about 10 to 20 times larger than that one of the polynomial approximations and the simulation time of the neural network is still about 5 times higher.

Figure 9 shows another example of applying response surfaces. The simulation error of both schemes, polynomial approximations and neural networks, are compared. Results of the preliminary design tool, mentioned in chapter 3, are used to train the response surfaces. The input consists of 6 design variables, the output of 20 results values, describing aerodynamic and geometrical design properties as well as first costs of the turbine. Each output value is trained with an independent response surface. The order of the polynomial approximation scheme is $p = 10$. The neural network contains of a two-layer feed-forward structure with 14 neurons in the first layer and 7 neurons in the seconds layer.

After training 500 randomly chosen sample points, the relative mean error of the response surface based on neural networks ranges at about 0.1 to 0.3%, with 2 exceptions of about 1%. The response surface based on polynomial approximations is significant worse. The maximal error lies at

25.4%, where the average error ranges at about 4% with a very high standard deviation. Only 4 of the 20 output values have reached the same accuracy as the output values of the neural network.

Increasing the number of training sets improves the polynomial approximations. However, the training time has been the same for both schemes in this example. Only in the simulation times are the polynomial approximations superior. Their simulation time is about 10 times smaller than that one of the neural networks.

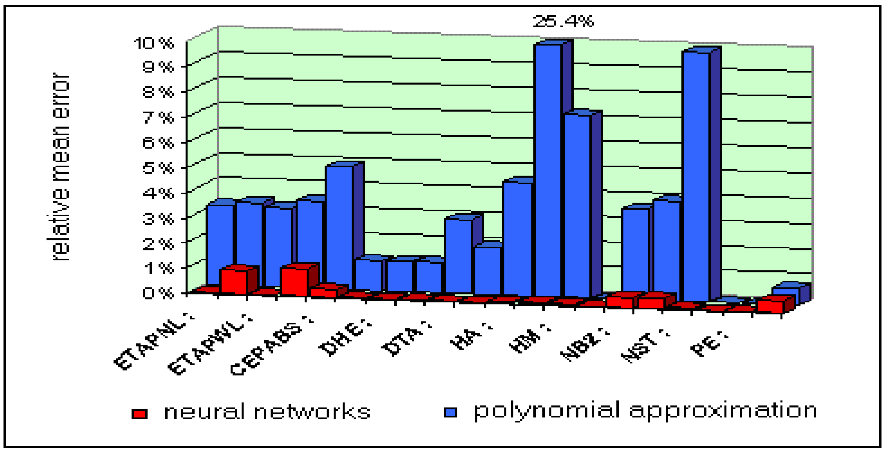

Figure 9: Response surface simulating the preliminary design tool

5 3D BLADE DESIGN OPTIMIZATION

Skipping the flow path and blade design in this paper, we focus on the last steps of turbomachinery design, the final 3D blade design optimization. An advanced blade parameterization scheme[5] is necessary in order to perform particular blade modifications with a limited number of optimization parameters, avoiding the generation of invalid designs a priori. As an extension to the optimization environment presented in chapter 2, the parameterization includes several components, such as profile, blade and grid generation tools, as shown in Figure 10. After the solver computations, the pre-evaluation tool processes the solver output data. First, it applies the postprocessing tools and then selects the objectives and constraints. The objectives and constraints are finally assessed in the evaluator indicating the design quality.

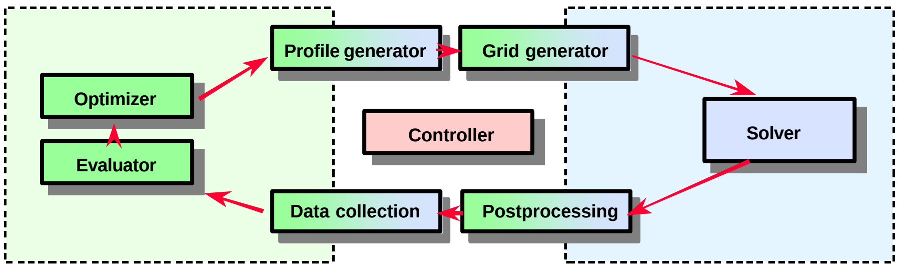

Figure 10: Optimization environment used for 3D blade design optimization

Performing 3D blade design optimizations, the objectives are different compared to the preliminary design. In 3D turbomachinery design, the main solver is a CFD turbomachinery code. With respect to multidisciplinary design optimization, additional solvers are applied to other disciplines (e.g. mechanical integrity, cost analysis…), either parallel to the CFD computations, and/or afterwards using calculated flow quantities. From the CFD calculation, an aerodynamic value is chosen as objective, with the constraints on the design given by the other disciplines.

For example, when optimizing 3D geometry details of a steam turbine stage[5], shown in Figure 11, the overall goal is to raise the efficiency of the entire turbine. This goal leads to a particular aerodynamic objective for the single stage. Such an objective might be minimizing the stage losses, or reaching a prescribed flow distribution after the stage. In general, sound investigations have to be made for deriving reasonable 3D flow assessment criteria for such an objective as well as for the corresponding constraints.

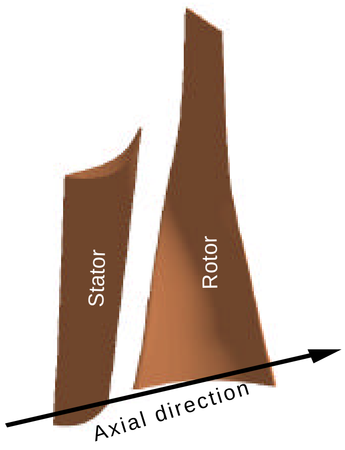

Figure 11: Rotor and stator of a low pressure steam turbine

Figure 12 shows the convergence history of the 3D blade design optimization of this turbine stage. Due to a sophisticated parameterization, only 8 optimization parameters could be used as design variables. They are modifying first and second order lean and sweep of stator blade (row 5) and rotor blade (row 6), bending the blades in circumferential and axial direction. The fitness value is given by the aerodynamic objective with a penalization put on the objective if the constraints are violated.

In our design case, the objective has been to minimize stage losses. Over the number of computed solutions, it can be seen that the fitness value is decreasing on average. The peaks appearing in the fitness history depend partly on the violated constraints, partly on the stochastic optimization algorithm (evolution strategy), which tries randomly to find other good solutions inside the entire valid design space. After about 210 computed solutions the algorithm has converged.

This example shows, how the design optimization process has changed compared to the preliminary design steps. As the applied 3D aerodynamic turbomachinery solvers need extensive computing time (several hours) for the calculation of all flow details, it is more important to use fast convergent optimization algorithms and strategies than to perform entire Pareto-optimizations. Such Pareto-optimizations lose their importance anyway, if only one objective must be minimized.

In order to save computing time, the use of response surfaces seems to be reasonable. However, as long as not enough training data is available, response surfaces cannot be sufficiently accurate. Compared to the 210 computed solutions, necessary to find the optimum of the 3D blade design with an evolution strategy, response surfaces would still be too rough to guide the optimization process in the right direction.

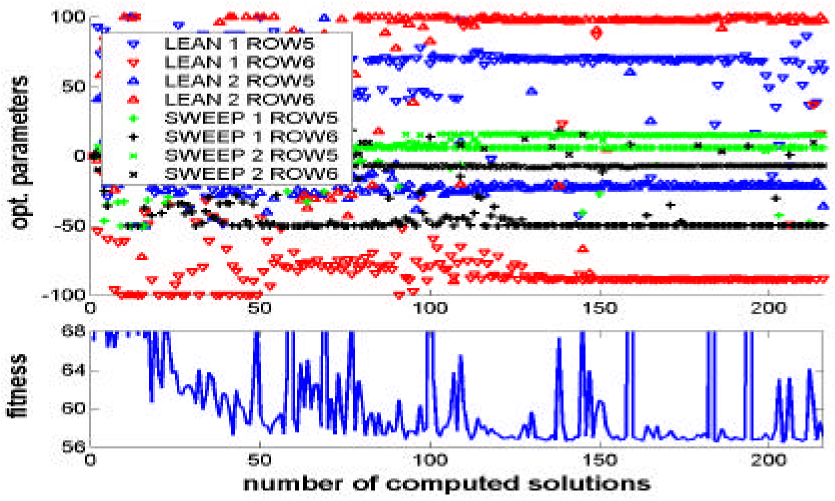

Figure 12: Convergence history of a 3D blade design optimization

12 CONCLUSIONS

Investigating the first and last tasks of multidisciplinary turbomachinery design, the optimization process changes totally.

In preliminary design, various objectives of different disciplines must be treated mutually. In this case, it could be shown that Pareto-optimization is able to deliver clear results showing the state of design technology. The designers can select the most promising design variants for the next design steps.

However, in the last turbomachinery design steps (3D blade design optimization) only one aerodynamic objective is governing the optimization process. Further disciplines, such as mechanical integrity and cost analysis, deliver further constraints for the design. For this particular case, as the aerodynamic computations are very time consuming, it is more important to use fast convergent optimization algorithms than to perform Pareto-optimizations.

Response surfaces promise to accelerate the entire design optimization process, because the are able to quickly approximate the results of computational expensive solvers. Two different schemes, polynomial approximations and neural networks, have been compared.

Exponential functions often cannot be represented sufficiently with response surfaces based on polynomial approximation schemes. Solvers computing such exponential functions (i.e. in the solution of partial differential equations) should better be approximated with response surfaces based on neural networks. However, if the models are pure polynomial functions and/or the solution space is quite smooth, polynomial approximations allow retrieving quickly the underlying mathematical equations, or at least a good approximation of them.

REFERENCES

[1] S. Havakechian and R. Greim, Aerodynamic design of 50 per cent reaction steam turbines, Proc Instn Mech Engrs Vol. 213 Part C, IMech, 1999.

[2] S. Müller and P. Koumoutsakos, Film Cooling Optimization Using Evolution Strategies, submitted to AIAA J. 2000.

[3] R. Dornberger, Optimization Environment Using Soft-Bounded Constraints, 3rd World Congress on Structural and Multidisciplinary Optimization, Buffalo, 1999.

[4] S. Havakechian and R. Greim, Recent advances in aerodynamic design of steam turbine components, VGB conference, 1999.

[5] R. Dornberger, Peter Stoll and Dirk Büche, Multidisciplinary Turbomachinery Blade Design Optimization, 38th AIAA Aerospace Science Meeting, Reno, 2000.

[6] R. Dornberger, D. Büche, Multidisciplinary and multiobjective optimization applied to turbine design, Technical Report AAT-1999-5, ABB ALSTOM POWER Technology Ltd, 1999.

[7] R. Dornberger, B. Hoffmann, Entwicklung von Response Surfaces zur Simulation komplexer Modelle für allgemeine Optimierungsanwendungen, Technical Report AAT-2000-7, ABB ALSTOM POWER Technology Ltd, 2000.