WDS'07 Proceedings of Contributed Papers, Part I, 233–240, 2007. ISBN 978-80-7378-023-4 © MATFYZPRESS

Parameterization of Airfoils and Its Application in Aerodynamic Optimization

J. Hájek

Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic.

Institute of Thermomechanics, Czech Academy of Sciences, Prague, Czech Republic.

Aeronautical Research and Test Institute (VZLÚ), Prague, Czech Republic.

Abstract. This article presents the area of aerodynamic design optimization, and emphasizes the topics of the most intensive current research within this area. Various approaches for shape parameterization, optimization algorithms and metamodelling are briefly explained. A real-world application is presented.

Introduction

Design optimization is a very popular discipline in engineering today. Almost anything that is being designed may be optimized in some way. With today's efficiency, computers may often outperform engineers in optimizing complex designs subject to complex objectives. Therefore, engineers are leaving hand-tuning in favor of automatic optimization. The research in the area of design optimization algorithms and techniques is thus very important for engineering industry.

The no-free-lunch principle

"No-free-lunch" is a general principle that can be observed in many areas of applied mathematics. In the context of design optimization, it can be roughly interpreted as that for every design optimization scheme (or every of its components) there is a compromise between its performance on a specific problem at hand and its general performance. In other words, it can be said that it is not possible to find an optimization algorithm that excels on all sorts of problems. By tuning an algorithm to a specific type of problems, we need to sacrifice the efficiency or applicability for other classes of problems.

Airfoil shape parameterization

In the context of design shape optimization, parameterization means encoding the design by numbers. As such, the parameterization plays a key role in transforming the engineering problem into a mathematical one. By changing the parameterization, we change the mathematical optimization problem; it is therefore not surprising that the choice of parameterization has a strong influence on the whole optimization process. This has been often confirmed in the literature [*Wu et al.*,2003], [*Samareh*,2001]. In the following, we focus on parameterization of airfoils, which is very important for aerodynamics. It can be used to optimize stand-alone airfoils subject to various objectives and constraints, or even included to complex schemes for parameterization of whole wings and airplanes.

Joukowski transformation

This is a classical airfoil parameterization scheme. It consists of transforming a circle in the complex plane via the transformation

$$z = \xi + \frac{1}{\xi}$$

The circle should pass through the point $\xi = 1$. The airfoil shape can be controlled by varying the position of the center of the circle. The big advantage of this method in the past was that a

plane potential flow around the resulting airfoil was solvable analytically by using properties of conformal mappings. The resulting airfoils are called Joukowski airfoils. Although Joukowski airfoils are no longer practically studied or used, they still retain theoretical importance, as they provide the basic understanding of the effects of airfoil thickness and camber on the flow-field around airfoil. Several variations of this method have been developed, such as the extended Joukowski transform [*Douglas Co.*,1990].

Splines

Splines use piecewise polynomial approximations of curves, that in addition ensure some smoothness of the approximating curve. B-splines utilize the same idea, but are built on more elegant and powerful approach as linear combinations of base functions. A further generalization of B-splines are NURBS – non-uniform rational B-splines. These schemes can be used to parameterize and approximate arbitrary curves, and can therefore be utilized in a vast number of applications. There is extensive literature on this subject.

Hicks-Henne shape functions

Hicks-Henne shape functions recently became popular for modelling small or moderate perturbations of "baseline" airfoil shapes for solving various optimization problems such as inverse design. The perturbation is expressed as a linear superposition of "bump" functions of the form

$$y = \sin^3(x^\beta)$$

where β is used to control maximum of the bump function, which is located at

$$x = \pi/2^{\frac{1}{\beta}}.$$

PARSEC

PARSEC is a parameterization scheme targeted at representing subsonic and transonic airfoils, originally developed in [*Sobieczky*,1998] and later used in numerous applications. Its key idea is expressing the airfoil shape as an unknown linear combination of suitable base function, and selecting 12 important geometric characteristics of the airfoil as the control variables, in such a way that the airfoil shape can be determined from these control variables by solving a linear system. The PARSEC parameters are described by Figure 1: Given these parameters,

unknown coefficients

$$a_{1U}, \ldots, a_{6U}, \quad a_{1L}, \ldots, a_{6L}$$

are determined so that the upper (U) and lower (L) surfaces of the airfoil are expressed by the equations:

$$y_U = \sum_{i=1}^{6} a_{iU} x^{i-0.5},$$

$$y_L = \sum_{i=1}^{6} a_{iL} x^{i-0.5}.$$

The advantages of the PARSEC parameterization are: No baseline shape is needed, wide range of airfoil shapes can be generated, typical geometric constraints on the airfoil shape (e.g., thickness) can be expressed or approximated by simple bound or linear constraints. Moreover, the impact of individual PARSEC design parameters on the aerodynamic properties of the airfoil can be predicted more easily.

Comparing PARSEC and splines can be viewed as a typical instance of no-free-lunch principle: PARSEC is designed specifically for airfoils, so that it could perform better on the task of airfoil optimization, but is not applicable to such wide class of problems as spline curves.

gPARSEC

gPARSEC is a generalization of PARSEC, developed at VZLÚ Prague, that allows employing arbitrary base functions in PARSEC and augmenting the scheme with additional degrees of freedom in a flexible manner to facilitate fine tuning of the airfoil shape, while preserving the ability to directly prescribe the PARSEC geometric characteristics of the airfoil. The details are described in [*Hájek*,2007].

Grid parameterization

A typical representation for airfoil shape, used by most computational packages, is a dense sequence of two-dimensional (x-z) coordinates of sample points along the airfoil surface. In principle, it is possible to use the coordinates of these points directly as design variables for airfoil optimization. In this manner, a very flexible parameterization is obtained; however, great care must be taken to preserve smoothness and reasonable shape of the airfoil. Moreover, the number of design variables in this scheme is very high, which makes the optimization problem quite difficult.

Optimization

Local model-based optimization

These algorithms repeatedly build a local approximate model of the objective function. The model is then used to select promising design vectors, the vectors are evaluated, the model is updated and the cycle is repeated until a convergence criterion is met. Almost all classical algorithms fall into this category. The steepest descent algorithm uses a local linear polynomial model. Newton, Quasi-Newton, Conjugate Gradient and Powell method all exploit a quadratic polynomial model. Even higher order, specially structured polynomial models can be exploited in so-called tensor methods. The models may be either regarded as global and be optimized conservatively (the line search approach) or only trusted in a limited area and be optimized in this area (the trust-region approach).

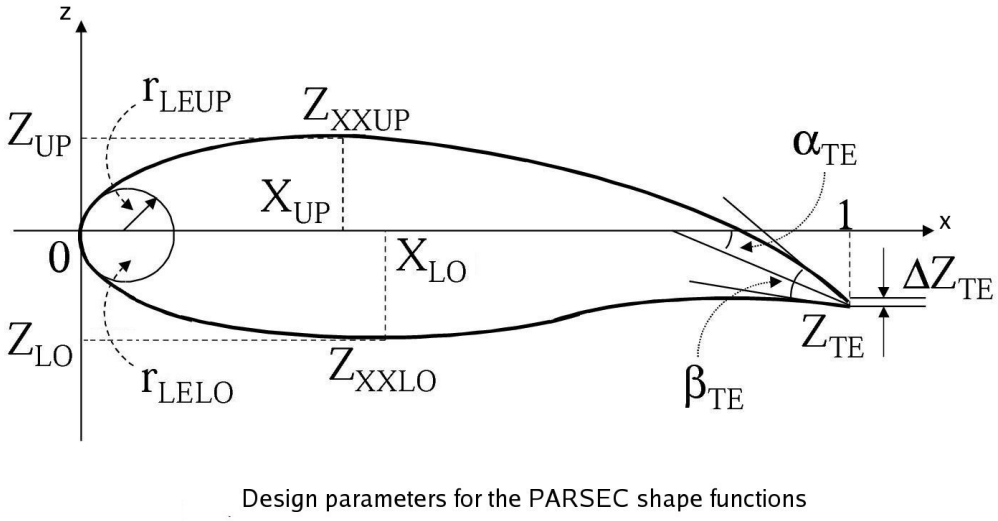

Figure 1. Design variables for PARSEC

Genetic algorithms (GA)

Genetic algorithms were originally developed using an analogy with natural evolution. They maintain a set of designs, called a *population*, which evolve so as to achieve better objective values (called *fitness* in terms of GA). The evolution is performed through the so-called *evolution operators*: mutation, crossover, selection and possibly more, which represent various schemes to generate new individuals or to select which individuals persist into next generation. Originally, GAs were used to tackle combinatorial optimization problems, such as the famous Travelling Salesman problem. For the application in continuous optimization problems, however, several aspects needed to be changed. An instance is the crossover operator. A typical representative of crossover operators for combinatorial problems is *single-point crossover*, defined as follows: For two parents with design vectors $p_1, p_2, \ldots, p_n$ and $q_1, q_2, \ldots, q_n$, a random index m is selected and two children generated are $p_1, \ldots, p_{m-1}, q_m, \ldots, q_n$ and $q_1, \ldots, q_{m-1}, p_m, \ldots, p_n$. For combinatorial problems, this operator is typically efficient, due to the fact that it is able to identify and combine "good" patterns in design vectors. Alternatively, one can use a two-point crossover where the design vectors are split at two points, or even a uniform crossover, where each child design variable is drawn randomly from one of parents. These crossover operations can be readily applied to continuous optimization as well; however, their foundation is lost, because "good patterns" usually do not exist in continuous optimization problems. Instead, for continuous optimization problems, the geometrical distance of design vectors plays a fundamental role. Several crossover schemes have been proposed to target the different nature of continuous optimization; amongst others, we could name the SBX (simulated binary crossover) scheme.

Differential Evolution (DE)

DE is a simple population-based stochastic optimization algorithm. The principal idea of DE is generating new trial vectors from differences between existing trial vectors. In its simplest form, new vectors are generated as

$$\mathbf{x}_{\text{new}} = \mathbf{x}_{r_1} + F(\mathbf{x}_{r_2} - \mathbf{x}_{r_3}).$$

where r_1, r_2, r_3 are random indices, and F is a crossover constant. Afterwards, the newly generated vector is further combined with a randomly chosen individual y from the current population by one of the crossover schemes from previous section. The resulting vector is evaluated, and if it has better fitness than y, y is replaced.

Particle Swarm Optimization (PSO)

The PSO is an optimization algorithm, developed from simulations of the behaviour of bird flocks. It is structured to challenge especially continuous optimization problems. It maintains a set of particles that are moving through the design space and exchanging information. Each particle remembers its position $\mathbf{x}$, velocity $\mathbf{v}$, best position so far $\mathbf{x}_b$ and best position of the whole swarm $\mathbf{x}_g$. The velocities and positions are then repeatedly updated as follows

$$\mathbf{v}_{\text{new}} = \omega\mathbf{v} + 2\mathbf{r}_1 \circ (\mathbf{x}_b - \mathbf{x}) + 2\mathbf{r}_2 \circ (\mathbf{x}_g - \mathbf{x}),$$

$$\mathbf{x}_{\text{new}} = \mathbf{x} + \mathbf{v}.$$

$\circ$ denotes the Hadamard product (component-wise multiplication), $\mathbf{r}_1$ and $\mathbf{r}_2$ are vectors of unit random numbers. The vector $\mathbf{x}_b$ represents long-time memory of the particle, whereas the velocity is a short-time memory. The factor ω is an inertia factor. The smaller it is, the more is the process forced into convergence.

Multi-objective optimization

In real-world problems, we often have more than one objective quantity. For instance, we may want to minimize the drag of an airfoil and maximize the lift. The optimum of a multi-objective optimization problem is no longer a single design, but rather a whole variety of designs, each of which is optimal in a certain sense. The key concept is Pareto dominance: one design dominates another if the former is at least equally good in all objectives as the latter and strictly better in at least one of them. The result of multi-objective optimization is then the set of all non-dominated feasible designs, called a *Pareto front*. Without assigning preference to the individual objectives, every design in the Pareto front can be considered optimal.

Whenever we face a multi-objective optimization problem, we can use two approaches. Either we can reduce the multiple objectives into a single one, usually assigning them some weighting factors, and then use a single-objective optimization algorithm, or we can use a truly multi-objective algorithm that tries to approximate the whole Pareto front, typically by a set of design vectors that try to cover the Pareto front to some extent. Regardless of which approach we use, in the end, we need a single design to proceed with. The principal difference is that when using reduction, we formulate our preferences *a priori* using the objective reduction formula, whereas using a genuine multi-objective optimization algorithm, we select the preferred design vector *a posteriori*, when we already know the shape of the Pareto front.

Metamodels

The performance success of local model-based optimization algorithms leads naturally to the following strategy: Maintain and update a more sophisticated approximate model of the objective function, and repeatedly optimize this model by the robust optimizers presented in previous section. Several well-known methods for approximate modeling of unknown functions are presented in these sections. In optimization context they are usually called metamodels, as they provide a "model of the model". By coupling metamodels with the powerful evolutionary or swarm algorithms, very efficient optimizers can be obtained. This is currently a very research-intensive area.

Neural networks

Neural networks are a technique originally developed for pattern recognition, that takes inspiration from human brain. The structure of a neural network consists of neurons, connected by synapses. Mathematically speaking, neurons are usually simulated by a sigmoid function like $\sigma(t) = 1/(1+e^{-t})$, where t is the neuron input and $\sigma(t)$ is the neuron output. Synapses are realized by linear combinations of neuron outputs. The coefficients of these linear combinations are called *synaptic weights*, and they have to be determined by learning, when a set of sample inputs and outputs is presented to the network and its synaptic weights are adapted so that the prediction best fits the training data. The most famous learning algorithm for neural networks is the *error back-propagation* algorithm, which is essentially a stochastic gradient descent applied to sum of squares of residual errors. Many variants of error back-propagation, as well as other learning algorithms, have been developed and successfully applied to train neural networks. Amongst others, the evolutionary algorithms have also been utilized. There is extensive literature on neural networks.

RBF networks

RBF (radial basis function) networks can be viewed as a special class of neural networks. They typically have a single hidden layer and instead of the sigmoid function that is best fit for 'decisions', it uses radial basis functions of the form

$$\phi(\|\mathbf{x} - \mathbf{c}_i\|, \sigma_i),$$

where $\mathbf{x}$ is the input vector, $\mathbf{c}_i$ are called *centers* and σ_i are *widths*. Commonly, the Gaussian RBF is used:

$$\phi(t, \sigma) = \exp(-\sigma t^2).$$

The prediction then reads

$$y = \sum_{i=1}^{N} w_i \exp(-\sigma_i \|\mathbf{x} - \mathbf{c}_i\|^2).$$

If σ_i and $\mathbf{c}_i$ are given, then w_i can be learnt efficiently by solving a linear least squares problem. Thus, training a RBF networks is often done in two stages: first, σ_i and $\mathbf{c}_i$ are selected based on some heuristic algorithm, then w_i are trained using a suitable linear least squares solver.

RBF networks typically show advantage over classical sigmoid-based neural networks in continuous approximation (regression). Their simplicity and efficiency in regression has made the very popular metamodelling technique.

Support Vector Machines

Originally, support vector machines were also developed for classification problems, and later successfully applied to regression. The key idea is a virtual mapping that maps the input vectors into a virtual finite dimensional linear space, thus creating an unknown function in that space, which is then approximated by a hyperplane (i.e., a linear function). If this is done carefully, then the image of the hyperplane in the original input space can be entirely expressed by using dot products in the virtual space. One then proceeds by selecting a suitable *kernel function* $k(\mathbf{x}, \mathbf{y})$ such as the Gaussian

$$\exp(-\sigma \|\mathbf{x} - \mathbf{y}\|^2)$$

This kernel is used instead of the dot product in the virtual space, which then needs not be known. Once the parameters of the kernel are known, the training reduces to a quadratic programming problem, for which efficient algorithms exist. The predictions are sparse in the sense that only a portion of training data is employed in the prediction.

Gaussian processes

This technique was originally developed in the field of geostatistics, where it is commonly known as Kriging. It models the observed function in the form

$$f(\mathbf{x}) \approx m(\mathbf{x}) + Z(\mathbf{x})$$

where $m(\mathbf{x})$ is a simple function depending on few unknown parameters (often a constant) and $Z(\mathbf{x})$ is a zero-mean stationary Gaussian process with covariance modelled by another function $C(\mathbf{x}, \mathbf{y})$ again depending on several unknown parameters. The unknown parameters in $m(\mathbf{x})$ and $C(\mathbf{x}, \mathbf{y})$ are estimated by some method (usually the maximum likelihood estimator) and then a new predictions can be made as such linear combinations of training data that minimize the prediction variance. The advantages of the method include the availability of prediction variance and automatic learning of unknown parameters. This, however, comes at a high computational cost for large datasets (thousands), because large dense matrices need to be factored in the process. Making Gaussian process predictions more effective is still a subject of intensive current research. At the same time, however, computers are becoming faster, thus making the problem less painful "from below".

Making it work

Solving complex design optimization problems in engineering through stochastic algorithms supported by metamodelling is currently a very popular approach, especially because the great variability of available algorithms. To make things work in practice, several important things need not to be overlooked.

1. *Efficient constraint handling.* Real world optimization problems are usually subject to some constraints. Optimum is only seeked amongst those designs that satisfy the constraints. Some constraints can be difficult to satisfy at all, so it's necessary for an algorithm to treat them efficiently.

2. *Hierarchical constraints.* In real problems, a very typical situation is that some constraints or objectives can't be evaluated unless others are satisfied. And even if they can, it is often unnecessary to do so. An ideal optimization scheme should be able to exploit such a hierarchy.

3. *Parallelism.* The computational burden of design optimization problems in engineering typically lies in some simulation (CFD or FEM) that evaluates the objectives. The nature of stochastic algorithms usually permits parallel evaluation of multiple designs. If this is the case, then it should be ensured that the work is split efficiently amongst the processors.

4. *Failure robustness.* Evaluation of a design using complex simulations may often fail to complete for various reasons. An ideal optimization scheme should be able to cope with this problem.

5. *Reuse of information.* Quite often, an already running optimization process needs some changes, like resetting parameters, changing of metamodels, modification of objectives. Because the design evaluations are very costly, an ideal optimization scheme should in such case be able to reuse as much information as possible from the previous run.

A real-world example

The example mentioned here is an optimization of a subsonic airfoil that has been carried out at VZLÚ as part of the CESAR (Cost Effective Small AiRcraft) European project. These objective quantities were considered: The drag at travel flight regime, the drag and momentum at maneuvering regime, and the maximum lift of the airfoil at landing or taking off flight regime. These four objectives were to be evaluated in both the "laminar" (free transition of boundary layer) and "turbulent" (forced early transition of boundary layer) states, so that leaves eight objectives in total. Two essential geometric constraints were imposed on the airfoil shape to ensure acceptable material properties: the maximum thickness of the airfoil at least 17% of chord length, and the trailing edge gap at least 0.25% of chord length.

The generalized PARSEC parameterization with 20 design variables was used to generate airfoil shapes. Utilizing the well-known and industry-proven XFOIL program, a sophisticated evaluator was created allowing parallel evaluation of many airfoils in an efficient dynamically-scheduled manner on multiprocessor machines (the original XFOIL is an interactive software, intended for manual human-driven use). Because of the relative efficiency of the evaluating scheme (more than 50 airfoils in a minute), and considerably complex behaviour of the objectives, it was decided to optimize without a metamodel.

Two optimization algorithms developed currently at the institute were used: a truly multi-objective genetic algorithm, and a single-objective (i.e. reduction-based) modification of particle swarm algorithm. The genetic algorithm (developed already for much longer period) uses a very small population and an archive to save good designs this technique is called a micro-genetic algorithm. The single-objective PSO algorithm used an automatic adaptation of the inertia

parameter ω and some other small modification to achieve better convergence. Both algorithms performed comparably well.

Because of the nature of multi-objective optimization, there is no single optimal design. We compared our results to an industrial NASA airfoil MS0317, often used for wings expected to operate at both laminar and turbulent conditions. The aim in our case was to concentrate on improving the laminar travelling drag and both laminar and turbulent momentum, while keeping the other objectives, especially the turbulent maximum lift, from deteriorating.

The percentile improvements of three selected airfoils compared to the NASA ms0317 airfoil are shown below:

cd1	cm1	cd2	cd1t	cm1t	cd2t	clm	clmt
9.63%	13.74%	1.44%	1.89%	13.86%	1.17%	-0.32%	1.06%
6.90%	14.53%	2.35%	2.33%	13.72%	1.56%	0.54%	1.98%
6.83%	14.13%	1.31%	2.22%	13.32%	1.37%	1.23%	2.65%

An important observation about the optimization algorithms used is that while they are completely general, they required tuning of some parameters before the most satisfactory results are achieved.

Conclusion – no-free-lunch revisited

It should be clear from this article that the area of design optimization is very rich and active. This can be seen as inevitable from the viewpoint of the no-free-lunch principle, stating that tuning an algorithm to a problem produces better results. But will the principle prevail in the future? The current most active research shows the most interest in connecting very general and powerful algorithms such as the stochastic algorithms with the efficient metamodelling techniques to obtain an universal algorithm. This may not be entirely impossible - for an analogy, one can look at the area of data compression. Even if one exploits a special structure of his data, it is very hard to beat the top general compression algorithms (e.g., bzip2, LZW, mp3) significantly. The no-free-lunch principle vanishes. Whether or not this can happen in the area of design optimization, we shall yet have to see.

References

Sobieczky, H.: Parametric Airfoils and Wings, Notes on Numerical Fluid Mechanics, edited by K. Fujii and G.S. Dulikravich, Vol. 68, Vieweg Verlag, 1998, pp. 71-88

Wu, H.Y., Yang, S.C., Liu, F. and Tsai, H.M., Comparison of Three Geometric Representations of Airfoils for Aerodynamic Optimization, AIAA 2003-4095, 16th AIAA CFD Conference, June 23-26, Orlando, FL, 2003

Samareh, J.A., Survey of shape parametrization techniques for high-fidelity multidisciplinary shape optimization, AIAA Journal, Vol. 39, No. 5, 2001, pp. 877-884

Hájek, J., The generalized PARSEC parameterization for airfoil optimization, submitted Computer Aided Design Journal, Elsevier.

Douglas Aircraft Company: Study of High-Speed Civil Transports, NASA-CR4236, 1990.