Surrogate Modeling of Stability Constraints for Optimization of Composite Structures

S. Grihon, E. Burnaev, M. Belyaev, and P. Prikhodko

Abstract Problem of aircraft structural components (wing, fuselage, tail) optimization is considered. Solution of this problem is very computationally intensive, since it requires at each iteration a two-level process. First from previous iteration an update step at full component level must be performed in order to take into account internal loads and their sensitivities in the whole structure involved by changes in local geometry. Second numerous local analyzes are run on isolated elements (for example, super stiffeners) of structural components in order to calculate mechanical strength criteria and their sensitivities depending on current internal loads. An optimization step is then performed from combined global-local sensitivities. This bi-level global-local optimization process is then repeated until convergence of load distribution in the whole structure. Numerous calculations of mechanical strength criteria are necessary for local analyzes and results in great increase of the time between two iterations. In this work an effective method for speeding up the opti-

S. Grihon
ESAZO—Optimisation Centre, Rapid Sizing—Optimisation M&T, Airbus Operations SAS, 316, Route de Bayonne, 31060 Toulouse Cedex, France
e-mail: stephane.grihon@airbus.com

E. Burnaev · M. Belyaev · P. Prikhodko
Intelligent Data Analysis Group, DATADVANCE, Pokrovsky blvd. 3 building 1B, 109028, Moscow, Russia

M. Belyaev
e-mail: mikhail.belyaev@datadvance.net

P. Prikhodko
e-mail: pavel.prikhodko@datadvance.net

E. Burnaev (✉) · M. Belyaev · P. Prikhodko
Data Analysis and Modeling Lab, Institute for Information Transmission Problems, Bolshoy Karetny per. 19, Moscow, 127994, Russia
e-mail: burnaev@iitp.ru

E. Burnaev · M. Belyaev · P. Prikhodko
PreMoLab, Moscow Institute of Physics and Technlogy, 141700, 9, Institutskii per., Dolgoprudny, Moscow Region, Russia

mization process was elaborated. The method uses surrogate models of optimization constraints (mechanical strength criteria) and provides reduction of the structure optimization computational time from several days to a few hours.

Keywords Buckling analysis · Approximation · Mixture of experts · HDA · Composite structure · Surrogate modeling · Optimization

1 Introduction

Aeronautical structures are mainly made of stiffened panels, i.e., thin shells (also called skin) enforced with stiffeners (called frames and stringers) in both the orbital and longitudinal directions. The whole structure is studied by dividing it into elementary parts called super stiffeners, consisting of the theoretical union of a stringer and two half-panels. These basic structures are subject to highly nonlinear phenomena such as buckling, collapse, and damage tolerance.

In order to determine the optimal size of these super stiffeners, static mechanical criteria must be computed using dedicated software based on nonlinear calculations. Thus, the analysis and the dimension estimation of the whole structure is currently computed by running a two-level study: at a global level a finite element (FE) analysis run on the whole FE model provides internal loads applied to each super stiffener; at a local level these loads are used to compute static mechanical criteria. Most of these criteria are formulated using reserve factors (RF): a structure is validated provided all its RFs are greater than one.

Therefore, a detailed design of an aircraft fuselage requires a two-level loop. First, changes from the local geometry, defined at the previous iteration, involve a new internal load distribution in the whole structure; an update step must then be performed to take these changes into account and to compute sensitivities. Second, numerous local analyses are run on isolated super stiffeners to compute mechanical criteria and their sensitivities depending on current internal loads. This bi-level global-local optimization process is then repeated until convergence of the load distribution in the whole structure is achieved.

Local mechanical criteria are computed by local methods, which are used because of the huge dimensionality of the problem ($O(10^4)$ variables and $O(10^5)$ constraints). Local methods require gradients of the constraint functions, defined by static mechanical criteria. These gradients can only be obtained by finite differences. Values of the mechanical strength constraints are computed using dedicated software. A call to this software takes up to a second; as a consequence, the need for finite difference calculations in each of numerous local optimizations greatly increases the time between two update steps.

Therefore, the dimension estimation step in an aircraft development program is a repetitive and time-consuming process. Much time could be saved by using surrogate modeling instead of performing straightforward computing [14, 23]. Thus, the main motivation of this work is a surrogate modeling of buckling analysis in support of composite structure optimization. We want to achieve two goals of great importance for engineers working in the Airbus structural analysis framework: saving time in the pre-sizing processes and having the advantage of response smoothing, since surrogate models (SMs) provide a continuous and differentiable approximation of RFs that sometimes are not themselves continuous (as is often the case for semi-empirical approaches).

For surrogate modeling of static instability phenomena (the buckling and the collapse of a super stiffener) we used the MACROS software toolkit for surrogate modeling and optimization, developed by DATADVANCE [12].

Finally, the constructed MACROS Surrogate Model (MSM) was embedded into the pre-sizing optimization process of A350XWB composite boxes, realized in a pre-sizing tool COMBOX, for checking the validity of the approximation and its use instead of the corresponding constraint functions in the optimization process. It turned out that MSM allows one to obtain smoother convergence to a reasonable solution in fewer iterations with a smoother distribution of thickness/stringer dimensions and reduces the structure optimization computational time from several days to a few hours.

In the following sections we describe the pre-sizing tool COMBOX (Sect. 2), the surrogate modeling and optimization software toolkit MACROS (Sect. 3), the construction of the MSM for Airbus skill tool (Sects. 3 and 4), and the analysis of the optimization results based on the skill tool and constructed SM (Sect. 5). We end this article with some concluding remarks (Sect. 6).

2 COMBOX: A Pre-Sizing Tool Developed for A350XWB

The COMBOX tool (COMposite BOX pre-sizing) was developed in 2005 to support the pre-sizing of the A350XWB composite wing box (see Fig. 1). It has since been continuously improved and is being applied to all A350XWB boxes: wing, horizontal tail plane, and vertical tail plane.

2.1 COMBOX Sizing Process

The COMBOX sizing process encapsulates the full stress process for a wing box (see Fig. 2):

- Mapping of sizing properties,
- Update of a global finite element model (FEM),
- Calculation of internal loads through a static linear analysis based on the global FEM,
- Calculation of strength responses as reserve factors (RFs) through Airbus skill tools.

These are the usual steps of an airframe structural analysis for pre-sizing.

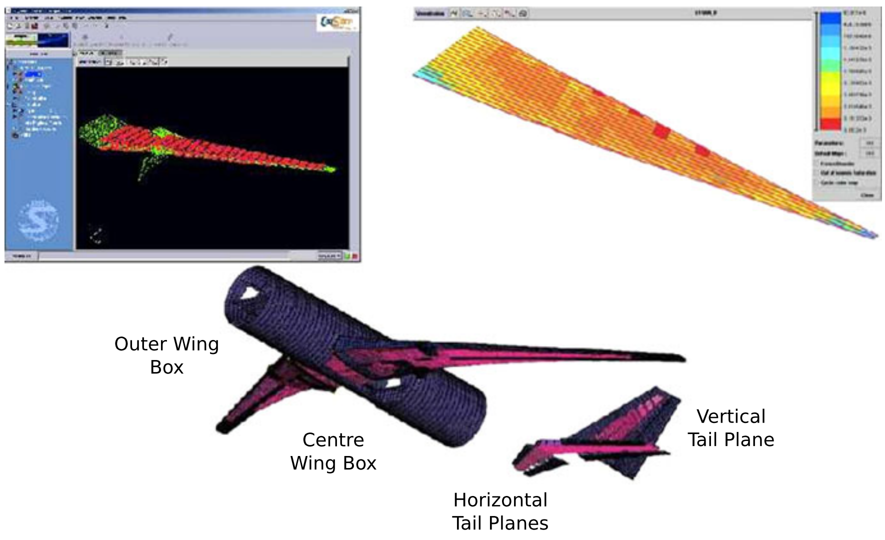


Fig. 1 COMBOX pre-sizing optimization tool is now applied to all A350XWB boxes

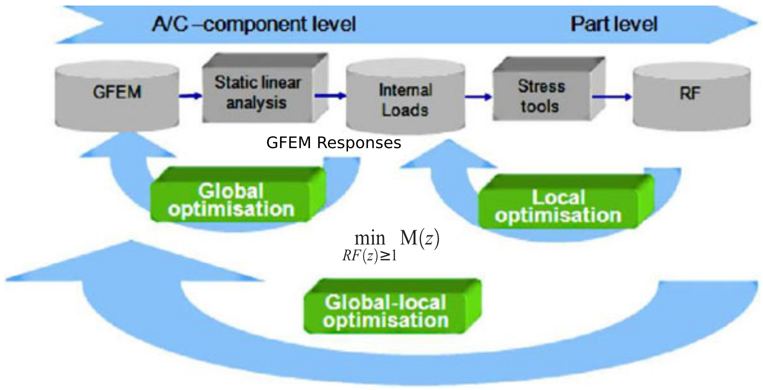


Fig. 2 COMBOX pre-sizing optimization tool is a global-local optimization capability encapsulating the overall stress analysis process

Remarks

- An RF indicates whether the structure is feasible (i.e., has enough strength) with respect to a given mechanical criterion or failure mode. If the RF is greater than one, the structure is feasible. If the RF is less than one, it is not feasible. Therefore, when modeling the dependency of some RF on a vector of design variables $\mathbf{x}$, the highest possible accuracy should be provided for what is called the accuracy domain $\tilde{\mathbf{X}} = \{\mathbf{x} : RF(\mathbf{x}) \in (1-\varepsilon, 1+\varepsilon)\}$, $\varepsilon = 0.2$.
- The simplest example of an RF is a ratio between an allowable stress (for example, material strength) and the applied stress.

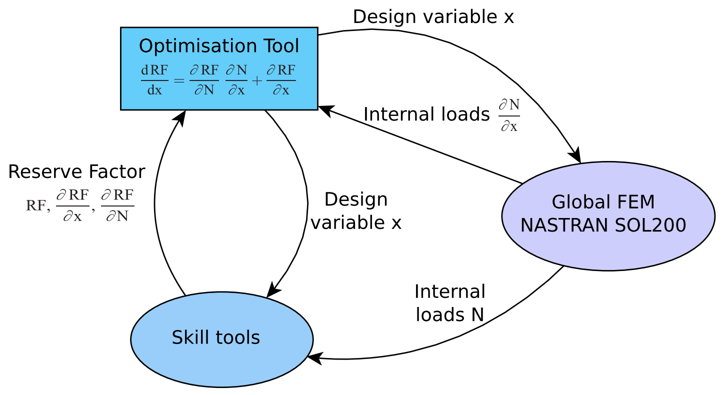

Optimisation Tool
$\frac{dRF}{dx} = \frac{\partial RF}{\partial N}\frac{\partial N}{\partial x} + \frac{\partial RF}{\partial x}$
Design variable x
Internal loads $\frac{\partial N}{\partial x}$
Reserve Factor
RF, $\frac{\partial RF}{\partial x}$, $\frac{\partial RF}{\partial N}$
Design variable x
Global FEM NASTRAN SOL200
Internal loads N
Skill tools

Fig. 3 COMBOX optimization process

- Skill tools are usually analytical semi-empirical tools, which are rather quick and are used for pre-sizing.

2.2 COMBOX Components

COMBOX is based on commercial off-the-shelf software and incorporates four components:

- CAESAM: Software framework from SAMTECH [20] (provides GUI and stress model),
- NASTRAN: Finite element software from MSC [19],
- Skill tools developed by Airbus,
- BOSS Quattro: Optimization software from SAMTECH [21] (provides process manager and optimiser).

2.3 COMBOX Optimization Process

COMBOX is a pre-sizing tool based on numerical optimization (mathematical programming). Therefore, besides sizing calculations (see Sect. 2.1) during optimization process it is necessary to compute the sensitivities of internal loads and RFs and combine them by chain ruling (see Fig. 3). Internal load sensitivities are semi-analytically calculated via the NASTRAN SOL200 module (NASTRAN optimization and sensitivity analysis module). The responses and sensitivities are then sent to the optimization algorithm in BOSS Quattro. CAESAM is mainly used to manage all data, and BOSS Quattro manages the work flow and the optimization process including the sensitivity chain ruling.

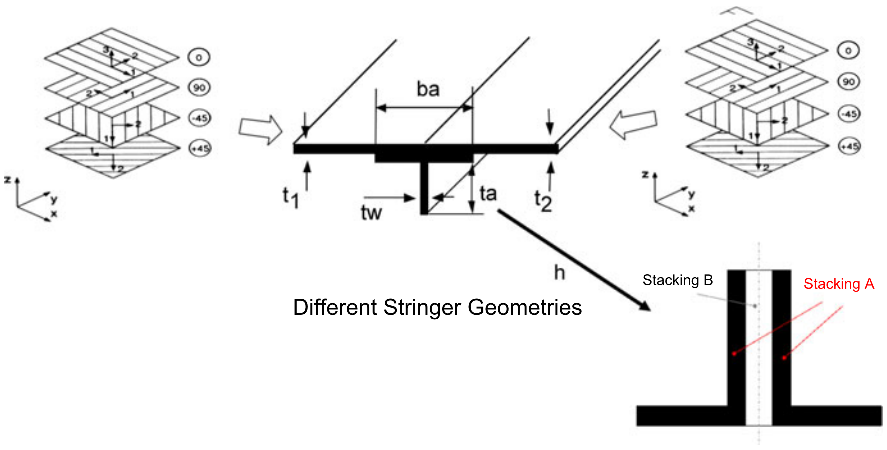


Fig. 4 Illustration of COMBOX design variables

2.4 COMBOX Optimization Problem Formulation

COMBOX is able to address all sizing variables of a composite cover with T-stringers (other stringer sections are possible but not presented here; see also Fig. 4):

- Skin thickness,
- Percentages of standard draping angles: 0 %, 45 %, 90 %,
- T-stringer core and web percentages: 0 %, 45 %, 90 %,
- T-stringer web thickness, core thickness, height, and width.

Bounds are given to these variables to satisfy design rules. Some additional design rules are included like bounds on the As/bt ratio, which represents the ratio of the stringer area to the skin area.

All usual criteria for a composite wing cover sizing are considered (see Fig. 5):

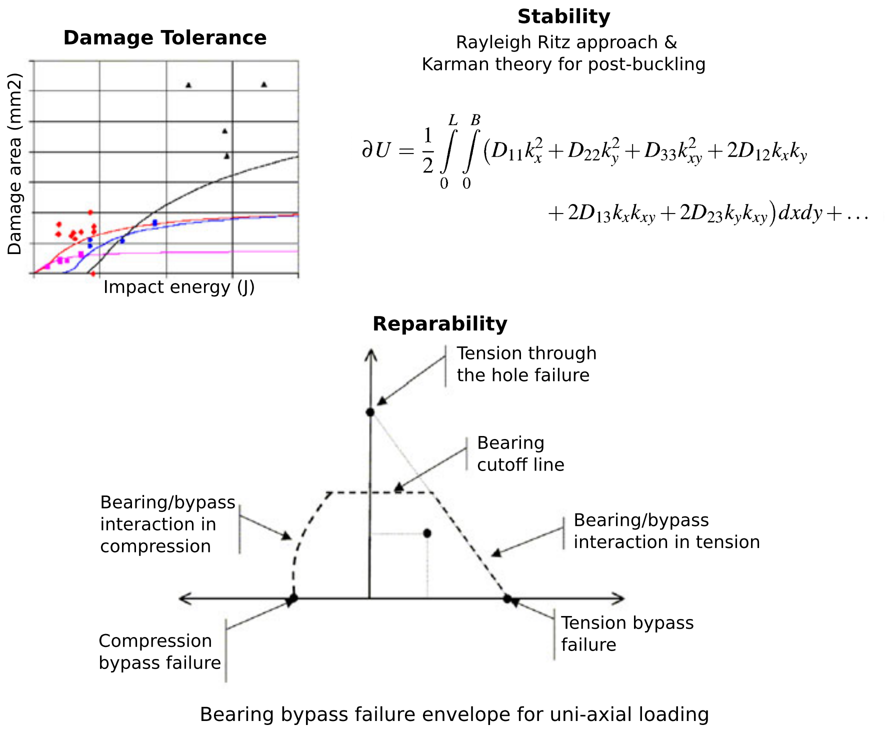
- Local skin buckling and general skin buckling,
- Post-buckling and post-buckling cut-off,
- Skin damage tolerance and stringer damage tolerance,
- Skin reparability and stringer reparability.

RFs are associated to each of these failure modes.

Damage tolerance criteria are there to ensure that the structure can resist small damages. Reparability criteria anticipate some future repairs in the skin (filled hole criteria).

Therefore, the optimization problem can be formulated as

$$M(z) \to \min_{z \in \mathbf{R}^n} \text{ s.t. } \begin{cases} z_{\text{low}} \le z \le z_{\text{up}}, \\ RF_{i,j,k}(N(z), z) \ge 1, \\ \quad i = 1, \dots, N_e,\ j = 1, \dots, N_l,\ k = 1, \dots, N_{\text{fm}}, \\ d_l(z) \ge 1, \end{cases}$$


Damage Tolerance
Damage area (mm2)
Impact energy (J)
Stability
Rayleigh Ritz approach &
Karman theory for post-buckling
Reparability
Tension through the hole failure
Bearing cutoff line
Bearing/bypass interaction in compression
Bearing/bypass interaction in tension
Compression bypass failure
Tension bypass failure
Bearing bypass failure envelope for uni-axial loading

$$\partial U = \frac{1}{2}\int_0^L\int_0^B \big(D_{11}k_x^2 + D_{22}k_y^2 + D_{33}k_{xy}^2 + 2D_{12}k_xk_y + 2D_{13}k_xk_{xy} + 2D_{23}k_yk_{xy}\big)dxdy + \ldots$$

Fig. 5 Illustration of COMBOX strength criteria

where

- The objective function is the mass $M(z)$ of the FEM, independent of percentages, that is to be minimized,
- z is the vector of n optimization variables (skin, stringer thicknesses, dimensions, and percentages),
- $N(z)$ is the vector of internal loads.

The constraints are

- Variable bounds: $z_{\text{low}} \leq z \leq z_{\text{up}}$,
- Strength constraints: $RF_{i,j,k}(N(z), z) \geq 1$, $i = 1, \ldots, N_e$, $j = 1, \ldots, N_l$, $k = 1, \ldots, N_{\text{fm}}$,
- Design constraints: $d_l(z) \geq 1$.

The indexes i, j, k for the strength constraints remind us that there are as many strength constraints as structural elements N_e, external loads N_l, and failure modes N_{fm}. The computational time of the process is mainly contained in the strength analysis due to the high value of $N_e \cdot N_l \cdot N_{\text{fm}}$. On top of that, RF sensitivities are obtained via finite differences; so the number of strength analyses is multiplied by the number of local variables and internal load components (approximately a factor 10).

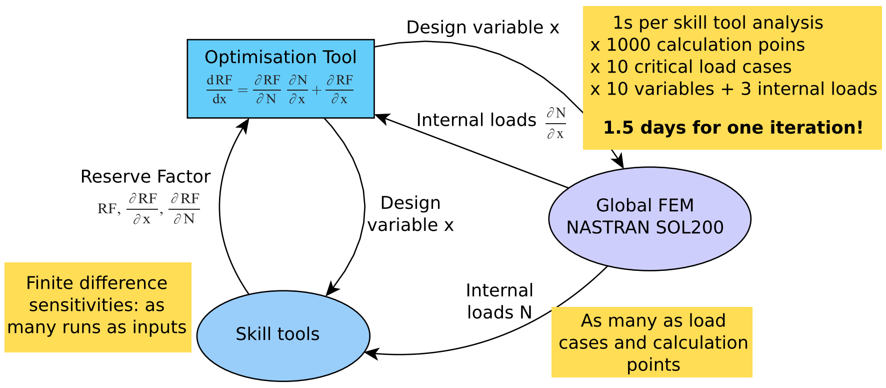

Design variable x
1s per skill tool analysis
x 1000 calculation poins
x 10 critical load cases
x 10 variables + 3 internal loads
1.5 days for one iteration!
Optimisation Tool
$\frac{dRF}{dx} = \frac{\partial RF}{\partial N}\frac{\partial N}{\partial x} + \frac{\partial RF}{\partial x}$
Internal loads $\frac{\partial N}{\partial x}$
Reserve Factor
RF, $\frac{\partial RF}{\partial x}$, $\frac{\partial RF}{\partial N}$
Design variable x
Global FEM NASTRAN SOL200
Finite difference sensitivities: as many runs as inputs
Skill tools
Internal loads N
As many as load cases and calculation points

Fig. 6 Computational times in COMBOX

Therefore, even if strength analysis tools for pre-sizing are rather quick (1*s* per element), the total number of calculations is huge and leads from one to five days per iteration with an optimization process usually converging in 20 iterations (see Fig. 6 for details). To save time in the pre-sizing processes and particularly in the COMBOX tool, it is thus necessary to build numerical approximations of the strength tools using surrogate modeling, which is the main goal of this paper. Besides the time reduction, there is also the advantage of response smoothing. Indeed, SMs give a continuous and differentiable approximation of RFs that sometimes are not themselves continuous (as is often the case for semi-empirical approaches). This is also demonstrated in the current study.

3 MACROS: A Surrogate Modeling and Optimization Software Toolkit

MACROS is a software toolkit for

- Intellectual data analysis, and
- Multidisciplinary optimization,

developed by DATADVANCE [12]. It provides proprietary and state-of-the-art data analysis and optimization techniques.

The MACROS toolkit consists of Generic Tools (GTs) for Dimension Reduction, Important Variable Extraction, Design of Experiments, Approximation, Data Fusion, and Optimization.

GT for Dimension Reduction includes unsupervised and supervised (feature extraction) techniques for automatic reparameterization of an object's description with a smaller number of parameters.

GT for Important Variable Extraction includes techniques for sensitivity analysis necessary for ranking the available parameters with respect to their influence on the given response function and selecting the most important ones.

GT for Design of Experiments enables systematic and efficient analysis of the design space by using classical and advanced methods (full factorial, optimal Latin hypercube, Halton and Sobol sequences, etc.) as well as specially designed adaptive techniques.

GT for Approximation allows automatic construction of fast-running data-based SMs using best-in-class predictive modeling techniques. The tool includes built-in robustness and accuracy assessment, control of SM smoothness, etc., and is efficient for small and huge data samples in low and high dimensions.

GT for Data Fusion allows approximating data of variable fidelities. The tool operates like GT for Approximation, but assumes that the response function is represented by two types of data: scarce high-fidelity data and abundant low-fidelity data. The tool then constructs an enhanced approximation of the high-fidelity model taking into account the abundant low-fidelity data.

GT for Optimization includes efficient state-of-the-art optimization methods to solve various problems (large-scale, linear/nonlinear, unconstrained/constrained, single/multi-objective, and stochastic).

Adaptive and automatic selection of the best method for a given problem on the basis of specially designed decision trees opens up elaborated methods for use by people who interact with problems on the engineering rather than the mathematical level.

4 Construction of MACROS Surrogate Model

Let us describe the following in this section:

- Proposed methodology for surrogate model (SM) construction, based on mixture of experts framework and used for construction of MACROS Surrogate Model (MSM),
- New High Dimensional Approximation (HDA) algorithm, implemented in GT for Approximation (GT Approx) and used for construction of experts (approximations in local regions),
- Differences and similarities between the proposed and already existing approaches for SM construction including results of computational experiments.

Construction of the SM is necessary for obtaining a more computationally efficient approximation of the original dependency. Therefore, let us formulate engineering statement of the approximation problem and then formulate requirements, which we impose on the SM.

4.1 Approximation Problem Statement

Let us denote by

$$S_{\text{learn}} = \left\{(\mathbf{x}_i, y_i), i = 1, \ldots, N_{\text{learn}}\right\} \tag{1}$$

points generated independently randomly such that there is some unknown functional dependency $y_i = f(\mathbf{x}_i)$ between the output value (output) $y_i \in \mathbf{Y} \subset \mathbf{R}^1$ and the input vector (input) $\mathbf{x}_i \in \mathbf{X} \subset \mathbf{R}^p$.

The SM construction problem statement is to construct an approximation (approximator, approximating model) $\hat{f}(\mathbf{x}) = \hat{f}(\mathbf{x}|S_{\text{learn}})$ for the given dependency $f(\mathbf{x})$ using learning sample S_{learn} such that for all $\mathbf{x} \in \mathbf{X}$ (not only for $\mathbf{x} \in S_{\text{learn}}$) the following approximate equality holds:

$$\hat{f}(\mathbf{x}) \approx f(\mathbf{x}), \tag{2}$$

i.e., the approximator $\hat{f}(\mathbf{x})$ has good generalization ability and recovers the given dependency with good accuracy.

Equation (2) is considered to be fulfilled if on independent test set $S_{\text{test}} = \{(\mathbf{x}_j, y_j), j = 1, \dots, N_{\text{test}}\}$ the value of the error

$$\hat{er}_{S_{\text{test}}}(f, \hat{f}) = \frac{1}{N_{\text{test}}} \sum_{j=1}^{N_{\text{test}}} \big(y_j - \hat{f}(\mathbf{x}_j)\big)^2 \tag{3}$$

is small (accuracy is high).

In order for the criterion (3) of approximation quality to make sense, the input vectors from the samples S_{learn} and S_{test} should be generated by the same distribution and distributed in $\mathbf{X}$ sufficiently densely.

In practice, when constructing an SM $\hat{f}(\mathbf{x})$, additional requirements and data generation source properties often should be taken into account.

4.1.1 Specific Requirements on Accuracy

There can exist different requirements on the accuracy of the SM in different domains of the design space $\mathbf{X}$. For example, when constructing an MSM for the considered stability constraints approximation problem, high accuracy of prediction should be provided in the domain $\tilde{\mathbf{X}} \subset \mathbf{X}$, where $\tilde{\mathbf{X}} = \{\mathbf{x} : f(\mathbf{x}) \in (1 - \varepsilon, 1 + \varepsilon)\}$, $\varepsilon = 0.2$. Due to this requirement, it is necessary to construct approximation only in the domain $\tilde{\mathbf{X}}$ using the subsample $\tilde{S} = \{(\mathbf{x}, y) \in S_{\text{learn}} : \mathbf{x} \in \tilde{\mathbf{X}}\}$ and then *glue* it with approximation, constructed in the domain $\mathbf{X} \setminus \tilde{\mathbf{X}}$ using the subsample $S_{\text{learn}} \setminus \tilde{S}$. Since the variation of the approximable function $f(\mathbf{x})$ is smaller in the domain $\tilde{\mathbf{X}}$, than in the whole design space $\mathbf{X}$, then this approach will allow to construct a more accurate approximation for $\mathbf{x} \in \tilde{\mathbf{X}}$.

4.1.2 Spatial Inhomogeneity of the Sample

When decomposing the design space and selecting domains (see Sect. 4.1.1) corresponding to different requirements on the accuracy of the SM, it can happen that the majority of these domains can be represented as the unions of some disconnected sets. Subsamples, corresponding to the selected domains, will also be some unions of clusters of points.

It is obvious that the global SM will have poor accuracy if it is to be constructed using a sample that is some union of several clusters of points. Thus it is reasonable to perform a preliminary decomposition of such a sample into several homogeneous subsamples, each of which is located in the connected subdomain of the design space. Then, using each subsample, a local approximation (expert) is constructed in the corresponding subdomain.

4.1.3 Redundancy in Data

It can also happen that the set of input parameters is redundant in one of the two (or even in both) senses:

- Input parameters can be dependent. In the simplest case it means that several input parameters are correlated.
- It may be that a function does not depend on all input parameters. In this situation, two main scenarios are worth considering:
  - the function weakly depends on several inputs,
  - the function depends not on initial inputs, but on their projection onto some linear subspace of smaller dimension.

Detection and removal of such redundancies in general allows us to significantly improve the quality of the constructed SM.

4.2 Methodology for Surrogate Model Construction

Let some sample S_{learn} (1) be given. Also, let us use GT Approx for construction of an approximation model $y = g(\mathbf{x}, \theta)$ based on the given sample S_{learn}. By *construction of the approximation model* we mean selecting some element $g(\cdot, \theta)$ from the predefined parametric family G by tuning parameters θ such that approximation is optimal with respect to criterion (3).

An elaborated approach for construction of the SM can be described as follows:

1. Methods to remove redundancy. In order to remove redundancy from the input parameters, methods for important variables extraction, dimension reduction, and feature extraction are used. Application of these methods for preprocessing the data will not be considered here further, since these methods are not used for constructing an SM in the considered applied problem. The problem statement and a detailed description of the method for effective dimension reduction are given in [8].

2. Decomposition of the design space into domains $\mathbf{X} = \bigcup_{j=0}^{N_y-1} \mathbf{X}_j$, corresponding to different ranges of the output. This decomposition is useful, since for different ranges of the output we need to provide different approximation accuracies. A detailed description of the procedure is given in Sect. 4.2.1.
3. Decomposition of the domains $\mathbf{X}_j$ into connected subdomains $\mathbf{X}_{j,k}$, corresponding to more regular behavior of the approximable dependency. A detailed description of the process is given in Sect. 4.2.2.
4. Construction of the approximators using the subsamples $S_{j,k} = \{(\mathbf{x}, y) \in S_{\text{learn}} : \mathbf{x} \in \mathbf{X}_{j,k}\}$.
5. Construction of the classifier that *estimates the proximity* from the given point $\mathbf{x}$ to subdomains $\mathbf{X}_j$. A detailed description of the process is given in Sect. 4.2.3.
6. Construction of the final SM by *gluing* obtained approximators. A detailed description of the process is given in Sect. 4.2.3.

4.2.1 Decomposition of the Design Space into Domains Based on Output Values

Decomposition of the design space into domains based on output values can be described as follows:

1. Let $y_{\min}$ and $y_{\max}$ be an upper and a lower bounds on the output value $y = f(\mathbf{x})$. The interval of output variation $y \in [y_{\min}, y_{\max}]$ is partitioned into N_y subintervals, i.e., $[y_{\min}, y_{\max}] \in \bigcup_{j=0}^{N_y-1} [y^{2j}, y^{2j+1}]$, where $y^0 = y_{\min}$, $y^{2N_y-1} = y_{\max}$, and $y^{2j} < y^{2j+1}$, $y^{2j+2} < y^{2j+1}$, $j = 0, 1, \ldots, N_y - 1$. Conditions on the ends of the subintervals provide nonempty intersections of these subintervals. This allows us to provide smooth *gluing* of the corresponding approximators (see Sect. 4.2.3). Selection of the decomposition is done, for example, according to accuracy requirements on the SM, imposed by the subject domain. The decomposition $\mathbf{X} = \bigcup_{j=0}^{N_y-1} \mathbf{X}_j$ of the design space corresponds to such a partition, where $\mathbf{X}_j = \{\mathbf{x} : f(\mathbf{x}) \in [y^{2j}, y^{2j+1}]\}$.
2. The sample S_{learn} is partitioned into N_y subsamples $S_{\text{learn}} = \bigcup_{j=0}^{N_y-1} S_j$ such that $S_j = \{(\mathbf{x}, y) \in S_{\text{learn}} : y \in [y^{2j}, y^{2j+1}]\}$. An approximator $f^j_{\text{approx}}(\mathbf{x})$ is constructed using each subsample S_j. This approximator can be some model from G (e.g., it can be constructed using GT Approx), or it can have a more complex structure; see Sect. 4.2.2.

We should note that the described decomposition of the design space into domains based on output values is not only useful if there are different requirements on the accuracy of the SM in different regions of the design space $\mathbf{X}$. In fact, if the function $f(\mathbf{x})$ is significantly spatially inhomogeneous, then the variability of the function $f(\mathbf{x})$ in the domain $\mathbf{X}_j \subset \mathbf{X}$ is significantly lower than in all the design space $\mathbf{X}$. Thus, if approximations are constructed for domains $\mathbf{X}_j \subset \mathbf{X}$ and are *glued*, then a more accurate SM can be obtained compared to the global SM, constructed at once for all the design space $\mathbf{X}$.

4.2.2 Decomposition of the Design Space into Domains Based on Input Values

In this subsection it is described how to additionally decompose the input design space into subdomains with more regular behavior of the approximable function.

An approximator $f^j_{\text{approx}}(\mathbf{x})$ for $\mathbf{x} \in \mathbf{X}_j$ (see Sect. 4.2.1) can be constructed using the sample S_j as an approximation model from G (by applying GT Approx to the sample S_j). However, if the sample S_j is significantly spatially inhomogeneous or even is represented by several separated clusters of points, then the approximator, constructed on the basis of this inhomogeneous sample, will in general have lower accuracy compared to an approximator constructed using a more uniform sample. In order to increase the accuracy, it is proposed to

- Additionally decompose the domains $\mathbf{X}_j$ into connected subdomains $\mathbf{X}_j = \bigcup_{k=1}^{N_x} \mathbf{X}_{j,k}$, such that
  - subsamples of the sample S_j, belonging to these connected subdomains, are more homogeneous,
  - within the subdomains $\mathbf{X}_{j,k}$ approximable dependency $f(\mathbf{x})$ has actually more regular behavior.
- Construct a separate local approximator $g_{j,k} \in G$ for each subdomain $\mathbf{X}_{j,k}$; then the final approximating model $f^j_{\text{approx}}(\mathbf{x})$ is represented as a continuous mixture of these local models.

For constructing the decomposition $\mathbf{X}_j = \bigcup_{k=1}^{N_x} \mathbf{X}_{j,k}$ on the basis of the sample S_j, $j = 0, \ldots, N_y - 1$, it is proposed to use a Gaussian mixture model (GMM, see [15]). It is assumed that the points $(\mathbf{x}, y) \in S_j$ are generated according to the model

$$\text{Law}(\mathbf{x}, y) = \sum_{k=1}^{N_x} \alpha^j_k \mathcal{N}\big(\mu^j_k, \Theta^j_k\big), \quad \sum_{k=1}^{N_x} \alpha^j_k = 1,\ \alpha^j_k > 0, \tag{4}$$

where μ^j_k and Θ^j_k are the mean vector and the covariance matrix for the k-th normal distribution of the GMM, generating j-th sample S_j. Also, the unconditional distribution of the input vector $\mathbf{x} \in \mathbf{X}_j$ has the form $\text{Law}(\mathbf{x}) = \sum_{k=1}^{N_x} \alpha^j_k \mathcal{N}(\mu^j_{k,x}, \Theta^j_{k,x})$, where $\mu^j_{k,x}$ and $\Theta^j_{k,x}$ are the subvector of the mean vector μ^j_k and the submatrix of the covariance matrix Θ^j_k, respectively.

Parameters of the GMM are estimated using the sample S_j by the standard EM algorithm [15], and in the framework of the GMM model additional clustering of the sample S_j is done. Note that since for clustering we use not only input values but also output values, then we take into account possible spatial inhomogeneity of the function in the domain $\mathbf{X}_j$. Then estimated parameters of the GMM in fact will provide the decomposition $\mathbf{X}_j = \bigcup_{j=1}^{N_x} \mathbf{X}_{j,k}$. Indeed, let us define the Mahalanobis distance from the center of the k-th cluster to the point $\mathbf{x}$ according to the formula $d_{j,k}(\mathbf{x}) = (\mathbf{x} - \mu^j_{k,x})^T (\Theta^j_{k,x})^{-1} (\mathbf{x} - \mu^j_{k,x})$; then the set $\mathbf{X}_{j,k} = \{\mathbf{x} \in \mathbf{X}_j : d_{j,k}(\mathbf{x}) \leq \chi^2_{97\%}(p)\}$, where $\chi^2_{97\ \%}(p)$ is a 97 % quantile of the distribution χ^2 with p degrees of freedom. Local approximators $g_{j,k} \in G$ for each subdomain $\mathbf{X}_{j,k}$ are constructed using subsamples $S_{j,k} = \{(\mathbf{x}, y) \in S_{\text{learn}} : \mathbf{x} \in \mathbf{X}_{j,k}\}$, $k = 1, \dots, N_x$ and GT Approx.

Let us estimate the weight characterizing to what extent the point $\mathbf{x}$ *belongs* to the k-th cluster according to the formula

$$w(k|\mathbf{x}, j) = \frac{1}{2}\left(\tanh\left(1 - 2\frac{d_{j,k}(\mathbf{x}) - \chi^2_{97\%}(p)}{\chi^2_{99\%}(p) - \chi^2_{97\%}(p)}\right) + 1\right), \tag{5}$$

where $\chi^2_{99\%}(p)$ and $\chi^2_{97\%}(p)$ are 99 % and 97 % quantiles of the distribution χ^2 with p degrees of freedom, respectively. In the framework of the model GMM (4), the classifier, which estimates the extent to which the point $\mathbf{x}$ belongs to the k-th cluster, can be constructed according to the formula

$$\hat{w}(k|\mathbf{x}, j) = \frac{w(k|\mathbf{x}, j)}{\sum_r w(r|\mathbf{x}, j)}. \tag{6}$$

The final prediction is obtained using smooth *gluing* of the local models $g_{j,k}$ for all clusters to which the point belongs and is calculated according to the formula

$$f^j_{\text{approx}}(\mathbf{x}) = \sum_{k=1}^{N_x} \hat{w}(k|\mathbf{x}, j) g_{j,k}(\mathbf{x}), \tag{7}$$

where $g_{j,k}(\mathbf{x})$ is a local approximator in the k-th cluster, constructed using the subsample $S_{j,k}$.

In fact, when constructing MSM, we initially tried to use weights equal to the posterior probabilities $\hat{w}(k|\mathbf{x}, j) = \mathrm{P}(k|\mathbf{x}, j)$ of the point $\mathbf{x}$ to belong to the corresponding clusters. Of course, for any point $\mathbf{x}$ there exist such clusters that are located far from it, but nonetheless the corresponding posterior probabilities $\hat{w}(k|\mathbf{x}, j) = \mathrm{P}(k|\mathbf{x}, j)$ are not zero. This means that predictions from local approximators, constructed for these clusters, are taken into account in the mixture (7), thus introducing the error into prediction.

Experiments showed that an additional significant increase in accuracy of MSM can be obtained by cutting the weights according to the distance to the cluster; i.e., we define the weight according to the formula

$$\tilde{w}(k|\mathbf{x}, j) = \begin{cases} \hat{w}(k|\mathbf{x}, j) & \text{if } d_{j,k}(\mathbf{x}) \le \chi^2_{99\%}(p), \\ 0 & \text{else.} \end{cases} \tag{8}$$

However, it is obvious that such an approach introduces discontinuities into MSM, due to which usage of MSM in the optimization process is impossible.

Thus, in order to smooth discontinuities and at the same time additionally penalize predictions (by decreasing the corresponding weights) which correspond to clusters lying far away from the considered point $\mathbf{x}$, we have introduced the approach based on the formula (5). Experiments showed that such an approach is more robust/accurate compared to other possible approaches.

4.2.3 Construction of the Classifier and Calculation of the Output Value of the Surrogate Model

In order to select approximating models $f^j_{\text{approx}}(\mathbf{x})$, $j = 0, \ldots, N_y - 1$ for calculation of the prediction $\hat{f}(\mathbf{x})$, it is proposed to use the classifier $f_{\text{class}}(\mathbf{x}) \in G$ that is a usual approximator, constructed on the basis of the whole sample S_{learn} by GT Approx. In this case the prediction $\hat{f}(\mathbf{x})$ for the given input vector $\mathbf{x}$ is calculated as follows:

- Calculate the value $f_{\text{class}}(\mathbf{x})$. Let us denote by $\#(A)$ the capacity of the set A, $J(\mathbf{x}) = \{j \in (0, \ldots, N_y - 1) : f_{\text{class}}(\mathbf{x}) \in [y^{2j}, y^{2j+1}]\}$. According to the conditions imposed on the decomposition $[y_{\min}, y_{\max}] \in \bigcup_{j=0}^{N_y-1}[y^{2j}, y^{2j+1}]$ of the output range (see Sect. 4.2.1), it holds that $\#(J(\mathbf{x})) \in \{1, 2\}$.
- If $\#(J(\mathbf{x})) = 1$, then calculate the prediction according to the formula $\hat{f}(\mathbf{x}) = f^{J(\mathbf{x})}_{\text{approx}}(\mathbf{x})$.
- If $\#(J(\mathbf{x})) = 2$, then, defining by $j \in J(\mathbf{x})$ the smallest of the two index values from the set $J(\mathbf{x})$, calculate the prediction according to the formula

$$\hat{f}(\mathbf{x}) = f^j_{\text{approx}}(\mathbf{x})\left(1 - v\left(\frac{f^{j+1}_{\text{approx}}(\mathbf{x}) - y^{2j+2}}{y^{2j+1} - y^{2j+2}}\right)\right) + f^{j+1}_{\text{approx}}(\mathbf{x})v\left(\frac{f^{j+1}_{\text{approx}}(\mathbf{x}) - y^{2j+2}}{y^{2j+1} - y^{2j+2}}\right),$$

  where $v(x) = 3x^2 - 2x^3$. This definition of the weight function $v(x)$ ensures smooth *gluing* of the models $f^j_{\text{approx}}(\mathbf{x})$ and $f^{j+1}_{\text{approx}}(\mathbf{x})$.

4.2.4 Positioning of the Proposed Methodology for Surrogate Model Construction Among Other Similar Methodologies

As can be seen from reviews [14, 26], the construction of an SM is usually considered to be just a solution of an approximation problem using conventional approximation methods such as (see [5, 11, 15]):

- Kriging (Gaussian process regression),
- Artificial neural networks (ANNs),
- Radial basis functions (RBFs),
- Support vector regression (SVR),
- Multivariate nonparametric regression,
- Polynomial regression, etc.

However these standard methods cannot provide sufficient accuracy, especially when approximating spatially inhomogeneous functions.

Therefore, in [4] a mixture of experts based methodology called IMAGE (Improved Metamodeling Approximation through Gaussian mixture of Experts) for SM construction was described. However, we have the following significant differences between the approach of [4] and the elaborated methodology.

1. Before decomposing the input design space using the GMM and EM algorithm (as in [4]), it is proposed to:
   - perform decomposition of the design space into domains based on output values. Such an approach is allowed to ensure more accurate approximation in the region $\tilde{\mathbf{X}} = \{\mathbf{x} : f(\mathbf{x}) \in (1-\varepsilon, 1+\varepsilon)\}$, $\varepsilon = 0.2$.
   - perform effective dimension reduction, for which an efficient algorithm based on Gaussian processes was developed; see [8].
2. Instead of using posterior probabilities as weights in mixture (7) (as in [4]), we use weights directly based on the Mahalanobis distance from the considered point $\mathbf{x}$ to clusters combined with a sigmoid activation function. This definition allows the additional penalization of terms in (7) that correspond to clusters located far away from the considered point $\mathbf{x}$.
3. Instead of using standard approximation techniques (RBF, SVR, etc.), as was done in [4], the powerful High Dimensional Approximation (HDA) technique, implemented in GT Approx, is applied. A short description of the HDA algorithm and its comparison with conventional approximation methods are given in Sect. 4.3.

The IMAGE surrogate modeling approach, based on a mixture of experts (standard approximation techniques are used as experts, such as RBF, ANN, SVR, etc.) and described in [4], was extensively compared, using data from the considered problem (see Sects. 1 and 2), with the global approximation algorithm HDA implemented in GT Approx (see Sect. 4.3). It turned out that:

- HDA provides better accuracy of approximation than conventional methods (see Sect. 4.3.4 for details),
- HDA global approximation provides better accuracy of approximation than the IMAGE SM, composed of local experts which were trained using conventional methods (see Sect. 4.3.5 for details).

Therefore it is reasonable to combine the mixture of experts approach for working with spatially inhomogeneous functions with the good approximation properties of HDA. This is done in the proposed framework for SM construction and allows us to obtain an efficient and accurate solution for the considered problem. See Sect. 5 for details.

4.3 *High-Dimensional Approximation*

As was mentioned in Sect. 4.2 (see also Sect. 4.2.2), MACROS GT Approx is used to construct approximation model $g(\mathbf{x})$ using the given sample S_{learn} and best-in-class predictive modeling techniques. For the considered problem, described in

Sects. 1 and 2, the decision tree built in GT Approx automatically selects the High Dimensional Approximation (HDA) method, which is efficient in the case of huge data samples and high input dimensions. The HDA method is briefly described as follows.

HDA approximator $g(\mathbf{x})$ (see also [9] for details) consists of several basic approximators $g_i(\mathbf{x}), i = 1, 2, \dots$, which are iteratively constructed and integrated into $g(\mathbf{x})$ using a specially elaborated boosting algorithm, until the accuracy of approximator $g(\mathbf{x})$ stops to increase. In fact,

$$g(\mathbf{x}) = \frac{1}{B}\sum_{k=1}^{B} g_k(\mathbf{x}), \tag{9}$$

where the number B of basic approximators $g_k(\mathbf{x}), k = 1, 2 \dots, B$ is also estimated by the boosting algorithm. Each $g_k(\mathbf{x}), k = 1, 2, \dots$ is trained on the modified sample $S_k = \{(\mathbf{x}_i, \widetilde{y}_{i,k}), i = 1, \dots, N_{\text{learn}}\}$, where $\widetilde{y}_{i,k}$ is some function of $(\mathbf{x}_i, y_i) \in S_{\text{learn}}$ and $\{\hat{y}_{i,j} = g_j(\mathbf{x}_i), j = 1, 2, \dots, k-1\}$. See [7] for details.

In turn, basic approximators $g_k(\mathbf{x}), k = 1, 2, \dots, B$ are represented as some averages

$$g_k(\mathbf{x}) = \frac{1}{M_k}\sum_{l=1}^{M_k} h_{k,l}(\mathbf{x}), \quad k = 1, 2, \dots, B$$

of elementary approximators $h_{k,l}(\mathbf{x}), l = 1, \dots, M_k, k = 1, 2, \dots, B$, obtained using multistart on their parameters. The value of M_k is estimated by the HDA training algorithm. An elementary approximator model is described in the next subsection.

4.3.1 Elementary Approximator Model

A linear expansion of parametric functions from the dictionary is used as an elementary approximator model in HDA; i.e., the elementary approximator has the form

$$h(\mathbf{x}) = \sum_{j=1}^{q} \alpha_j \psi_j(\mathbf{x}), \tag{10}$$

where $\psi_j(\mathbf{x}), j = 1, \dots, q$ are some parametric functions. Three main types of parametric functions are used (the justification for using these basis functions is given in [15]), namely:

1. Sigmoid basis function $\psi_j(\mathbf{x}) = \sigma(\sum_{i=1}^{p} \beta_{j,i} x_i)$, where $\mathbf{x} = (x_1, \dots, x_p)$, $\sigma(z) = \frac{e^z - 1}{e^z + 1}$. In order to model sharp features a different parameterization can be used, namely,

$$\psi_j(\mathbf{x}) = \sigma\left(\left|\sum_{i=1}^{p} \beta_{j,i} x_i\right|^{\sigma(\alpha_j)+1} \operatorname{sign}\left(\sum_{i=1}^{p} \beta_{j,i} x_i\right)\right),$$

where parameter α_j is adjusted independently of parameters $\beta_j = (\beta_{j,1}, \dots, \beta_{j,p})$. The essence is that for big negative values of α_j the function $\psi_j(\mathbf{x})$ behaves like a step function.
2. Gaussian functions $\psi_j(\mathbf{x}) = \exp(-\|\mathbf{x} - d_j\|_2^2/\sigma_j^2)$.
3. Linear functions $\psi_j(\mathbf{x}) = x_j$, $j = 1, 2, \dots, p$, $\mathbf{x} = (x_1, \dots, x_p)$.

Thus the index set $J = \{1, \dots, q\}$ can be decomposed into three parts $J = J_{\text{lin}} \cup J_{\text{sigmoid}} \cup J_{\text{GF}}$, where $J_{\text{lin}} \subseteq \{1, \dots, p\}$ corresponds to the linear part (linear functions), and J_{sigmoid} and J_{GF} correspond to sigmoid and Gaussian functions, respectively. Therefore, in order to fit the model (10) to the data, we should choose the number and type of functions q, and estimate their parameters by minimizing the mean-square error (performance function) on the learning set.

4.3.2 Training of Elementary Approximator Model

Training of the elementary approximator model (10) consists of the following steps:

1. The parameters of the functions from the dictionary are initialized (see description of the algorithm in [3]). An initial number of functions is selected with redundancy.
2. The model selection is made, i.e., values of q, $\#(J_{\text{sigmoid}})$, and $\#(J_{\text{GF}})$ are estimated, and redundant functions are deleted (see the descriptions of algorithms in [2, 7, 9]).
3. The parameters of the approximator are tuned using a hybrid algorithm based on regression analysis and gradient optimization methods. At each step of this algorithm:

   - parameters of linear decomposition (10) are estimated using ridge regression with adaptive selection of regularization parameter, while parameters of the functions from the decomposition are kept fixed,
   - parameters of the functions from the decomposition are tuned using a trust region based gradient method, while parameters of linear decomposition (10) are kept fixed.

   Additional details of the algorithm can be found in [1, 9].

4.3.3 Positioning of HDA Among Other Approximation Methods

The approximation problem appears in many applications, each of which imposes its specific requirements on the accuracy and properties of the approximator $\hat{f}(\mathbf{x})$. Also, depending on the nature of the data source, the corresponding learning sample can have different characteristics (for example, the size of the learning sample N_{learn} and input dimension p). It is natural that the selection of the approximation method should depend on both the requirements of the considered subject domain and the specific properties of the data. For example, a distinctive feature of many problems in biology is a very high input dimensionality $N_{\text{learn}} \ll p$, so application of nonlinear approximation methods for such problems is senseless. In the current work we consider an approximation problem in the framework of surrogate modeling, which also imposes its own requirements on the approximation method. The main peculiarity of data samples from this applied domain is a large sample size (N_{learn} can be up to several hundred thousand points), with input dimension p usually of size from 3 up to 50. Typically, the unknown function f has a complex nonlinear structure, and the approximation quality should be rather high. Along with that, approximation $\hat{f}$ should be smooth enough and have continuous derivatives, since very often an obtained approximation is then used as a cost function (or as an objective) in some optimization process.

One of the most widespread approximation methods is regression on the basis of Gaussian processes [25] (in engineering applications the name kriging is usually used, see [14]). In the framework of this approach it is assumed that approximable function f can be represented as a linear combination of some known functions (for example, linear functions) and a realization of a Gaussian process with zero mean and covariance function from some parametric family. Approximation construction then reduces to estimation of covariance function parameters using a maximum likelihood method, and the coefficients of linear combination can be explicitly obtained using a least-squares method. Regression based on Gaussian processes is a flexible model with a rather small number of tunable parameters, and it provides accurate recovery of nonlinear dependences even in the case of small sample size. Moreover, due to the probabilistic nature of the model, Gaussian process-based regression provides not only approximation $\hat{f}$ (defined by the posterior mean), but also an accuracy evaluation capability (defined by the posterior variance of the process).

However, this approach has shortcomings. When tuning parameters of the covariance function using the maximum likelihood principle, it is necessary to invert the covariance matrix of size $N_{\text{learn}} \times N_{\text{learn}}$ during each iteration of the tuning process, and the complexity of each inversion is $O(N_{\text{learn}}^3)$. Thus, this operation requires a lot of computational resources (CPU time and memory) for big sample sizes. For $N_{\text{learn}} \gg 1{,}000$ such operations cannot be done within a reasonable time on a modern PC, which restricts the application of kriging if the sample size is big. There is a solution for this problem, based on an approximation of the covariance matrix [10, 25] with computational complexity $O(N_{\text{learn}} m^2)$, where m is the size of some subsample. A sufficiently big part of the initial sample is taken as the subsample, which allows us to widen the range of applicability of Gaussian process regression up to sample sizes equal to 10^4. Also, the use of a covariance matrix approximation instead of its true value decreases the approximation accuracy.

The method based on K nearest neighbors suffers from the *curse of dimensionality*: in the high-dimensional case the notion of vicinity degrades, and the distance to the nearest points becomes comparable to the distance to the faraway points, which decreases the approximation quality.

Another very popular approximation method is based on artificial neural networks (ANNs) [15, 16], including their special subclass: radial basis functions (RBFs). The use of ANNs is largely explained on the basis of theoretical results (see, for example, [24]), which state that the ANN can provide approximation of a wide function class with any predefined accuracy. However, in practice this accuracy is not usually attained, since the theoretical results do not provide a practical method on how to construct an ANN (select the structure of the ANN, *tune* its parameters) for the given data sample S_{learn}. Nevertheless, an ANN is a flexible model which can be easily *extended* (by increasing the number of layers and/or their sizes) if the sample size grows. This flexibility also has drawbacks: ANN-based models have a lot of parameters (especially when there are many hidden layers), which prevents us from constructing robust/accurate approximations when only samples with small sizes are available. Moreover, model selection for an ANN is a very heuristic process.

The standard approach for *tuning* ANN parameters is an error backpropagation algorithm. Other methods exist for the tuning of parameters, which have faster convergence, for example, second-order methods, methods with adaptive learning rate, etc. However, these methods do not take into account the specific structure of the ANN, and in fact can be considered as applications of standard numerical optimization methods for tuning parameters in nonlinear regression. Also, the constructed approximation significantly depends on the (random) initialization of ANN parameters and (random) split of the learning sample into training and validation subsamples. So, ANN-based methods are rather flexible, but not very reliable for approximation construction.

For applications it is enough to consider only a two-layer perceptron as an ANN. Usually the hidden layer is composed either from sigmoid-type or RBF-type functions. However, in fact functions of different types can be mixed in one hidden layer, including, e.g., sigmoids, RBFs, and wavelets. A generalization of the two-layer perceptron can then be obtained with representation like that of Eq. (10). We call this kind of generalization *approximation by linear decomposition in nonlinear functions from a parametric dictionary* and we use the corresponding model as a base model in the HDA algorithm.

A typical algorithm for ANN learning can be decomposed into the following steps:

- Selection of the ANN structure,
- Random initialization of parameters,
- Random splitting of the learning sample into training and validation subsamples,
- Tuning of parameters using some gradient method for minimizing the mean-square error on the training subsample until the error on the validation subsample begins to increase.

However, such algorithms are mainly tailored for ANNs with general structure and do not take into account any specific structure of the model like (10). Therefore, in the HDA framework for all of these steps we have developed specific algorithms, which provide better results compared to the typical algorithms realized in commercial software. This allows us to obtain a good approximation accuracy; see Sect. 4.3.4 for more details.

4.3.4 Results of HDA Comparison with Conventional Approximation Methods

There are two possible ways to estimate the quality of the proposed approximation method. The first way is to check compliance with the requirements on accuracy, which are necessary for successful SM application. These requirements can be obtained only from the engineer and vary greatly for different problems. The second, more universal, method is to compare the quality of the proposed approach with conventional methods for approximation. We will consider realizations of such conventional methods in the software toolkits MatLab [17], modeFrontier [18], which are widely used in many industrial companies. In each of these software toolkits there are several methods for approximation of multidimensional dependency, including ANNs, regression based on Gaussian processes, etc.

A priori two main shortcomings of these toolkits can be pointed out. First, in order to select a particular approximation method for solving a given problem, an engineer must have either some knowledge of machine learning or must perform numerical experiments using all available approximation methods. Otherwise, one might select an approximation method which does not provide the best approximation quality. Moreover, many implemented approximation methods have strong limitations on the input dimension p and sample size N_{learn}, which in general decreases the approximation quality.

Let us consider the results on some indicative problems, covering a wide range of sample sizes and input dimensions (these results were obtained in 2010 during work on the PhD thesis):

- Airplane fuselage composite structure design (data from the Airbus in-house stability tools approximation problem considered here; see also [6] for details),
- Radiator characteristics modeling using dipoles,
- Fuel consumption of an airplane engine.

The accuracy of approximation is estimated using the square root of an error (3) on the test set, divided by the range of the corresponding output.

Remark 1 For the data from the airplane fuselage composite structure design problem, all computational experiments were made on the so-called accuracy domain $\tilde{\mathbf{X}} = \{\mathbf{x} : RF(\mathbf{x}) \in (1-\varepsilon, 1+\varepsilon)\}$, $\varepsilon = 0.2$. This means that the training (1) and test samples had the forms $\tilde{S}_{\text{learn}} = \{(\mathbf{x}, y) \in S_{\text{learn}} : \mathbf{x} \in \tilde{\mathbf{X}}\}$ and $\tilde{S}_{\text{test}} = \{(\mathbf{x}, y) \in S_{\text{test}} : \mathbf{x} \in \tilde{\mathbf{X}}\}$, respectively.

The results of the approximation accuracy comparison are given in Table 1.

Further, let us consider a significantly bigger set of problems (including three problems considered above), based on data from physical experiments or from simulation experiments with some physical model. The characteristics of the corresponding learning samples are rather diverse:

- Input dimension p varies from 2 to 435; the output dimension varies from 1 to 100.

Table 1 Relative approximation errors

Problem name		Composite structure	Dipole	Fuel cons.
Dimension p		16	7	3
Sample size N_{learn}		50,000	5,000	351
MACROS	HDA	**0.0120**	**0.0019**	**0.0044**
MatLab	Linear reg.	0.0806	0.1137	0.2075
	Quadratic reg.	0.0525	0.0971	0.1180
	RBF	**0.0454**	0.0246	0.0172
modeFrontier	K-nearest	0.0753	0.0811	0.0777
	Anisotropic kriging	0.0804	**0.0071**	**0.0126**
	Kriging	**0.0496**	0.0359	0.1845
	RBF	**0.0496**	0.0177	0.0375
	ANN	0.0503	**0.0037**	**0.0147**
	Evolutionary design	0.2242	0.1570	0.0266
	Gaussian processes	0.0898	0.0612	0.0211

- The size of the learning sample is up to 65,000 points.
- The number of considered problems is 30.

For each of these problems we constructed approximations using all available methods. For convenient representation of the comparison results on the full set of problems we used the visualization method described in [13]. This approach provides a graphical comparison of approximations accuracies for several considered methods and a large number of test problems. Let us briefly describe this visualization method using the following notation:

- $M_k, k = 1, \dots, K$ are considered approximation methods,
- $P_l, l = 1, \dots, L$ are considered test problems,
- $e(M_k, P_l)$ is a root mean square approximation error for the approximator, constructed using method M_k on the problem P_l,
- $\widetilde{e}(P_l) = \min_k e(M_k, P_l)$ is a minimal (among all available approximation methods) error for the problem P_l.

For any method M_k and scaling factor $a \geq 1$ let us define the following quantity:

$$P_k(a) = \frac{\#\{l : e(M_k, P_l) \leq a \cdot \widetilde{e}(P_l)\}}{L}.$$

In fact, the quantity $P_k(a)$ shows on which part of problems approximation errors of the method M_k are not a times bigger than minimal (among methods) approximation errors for the corresponding problems. In particular, $P_k(1)$ is equal to the fraction of problems for which the method M_k provides the smallest approximation error.

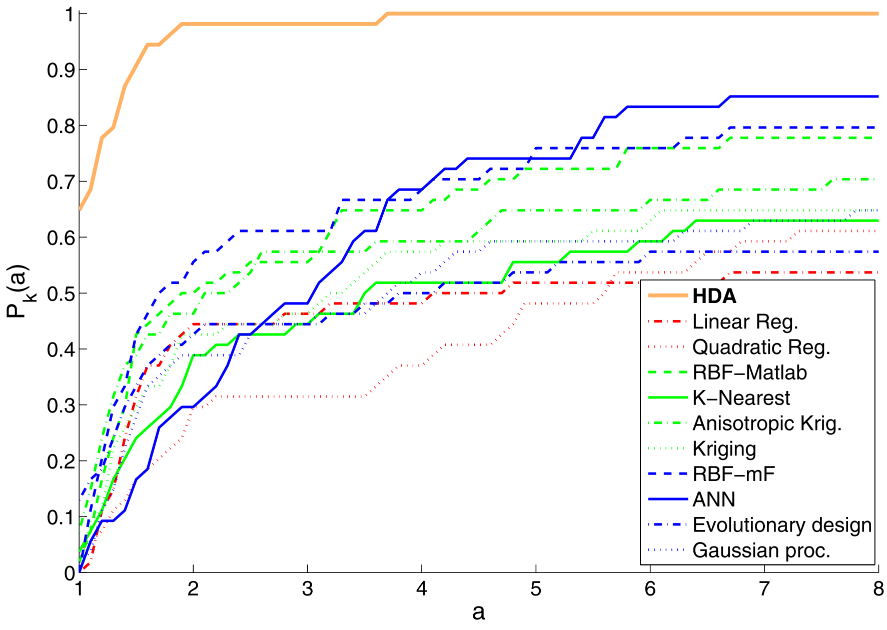


Fig. 7 Results of comparison

According to the obtained results (see Fig. 7), on 65 % of problems the HDA algorithm has the smallest error of approximation. Moreover, profile $P(a)$ of the HDA algorithm converges to 1 rather fast, which also shows its robustness. Therefore, this algorithm provides significantly better accuracy compared to conventional methods, realized in commercial products. Similar results were obtained during the internal comparison of GT Approx in Airbus with in-house software realizing different approximation methods.

4.3.5 Comparison of HDA with IMAGE

A comparison of IMAGE and HDA with conventional methods on the problem of Airbus in-house stability tools approximation showed that these methods had a significantly bigger accuracy than that of the conventional ones (for IMAGE see [4] and for HDA see [6], Sect. 4.3.4). Therefore, the main goal of this section is to compare the accuracies of IMAGE and HDA.

4.3.6 Setup of Experiments

The setup of the experiments can be described as follows:

- Input dimension $p = 20$, input vector defines geometries, material properties, etc.

- Typical learning sample size, used for construction of an approximation, is $N_{\text{learn}} \sim 200\,000$.
- Output characteristics consist of various reserve factors (RF for stringer local buckling, RF for skin local buckling, etc.) and strain values at different locations, also considering bending. The number of output characteristics to approximate is equal to 25.

An RF indicates whether the structure is feasible with respect to a given failure mode. If the RF is greater than one, the structure is feasible. If the RF is less than one it is not feasible. Therefore, when modeling the dependency of an RF on a vector of design variables $\mathbf{x}$, the highest possible accuracy should be provided for values of RF close to one. Thus for each output characteristic two domains were identified, $\mathbf{X}_1 = \{\mathbf{x} \in \mathbf{X} : f(\mathbf{x}) \in [0.5, 1.5]\}$ and $\mathbf{X}_2 = \{\mathbf{x} \in \mathbf{X} : f(\mathbf{x}) \in [0, 3]\}$, and for each of these domains an approximator was constructed using the corresponding train sample. Let us denote these approximators as $\hat{f}_1$ and $\hat{f}_2$, respectively.

As accuracy indicators we used $Emean$ (mean absolute error), $Eq95$ (95 % quantile absolute error), and $Eq99$ (99 % quantile absolute error). Accuracy indicators were estimated using separate test samples for sets $\mathbf{X}_{1,1} = \{\mathbf{x} \in \mathbf{X} : f(\mathbf{x}) \in [0.8, 1.2]\}$ and $\mathbf{X}_{1,2} = \{\mathbf{x} \in \mathbf{X} : f(\mathbf{x}) \in [1.2, 1.5]\}$ in the case of approximator $\hat{f}_1$ and for sets $\mathbf{X}_{2,1} = \{\mathbf{x} \in \mathbf{X} : f(\mathbf{x}) \in [0, 1.5]\}$ and $\mathbf{X}_{2,2} = \{\mathbf{x} \in \mathbf{X} : f(\mathbf{x}) \in [1.5, 3]\}$ in the case of approximator $\hat{f}_2$.

4.3.7 Obtained Results

Due to lack of space, detailed results are given only for two RFs: RF for stringer local buckling (RF STR, see Table 2) and RF for first skin buckling mode (RF PND GEN, see Table 3). We can see that HDA provides significantly higher accuracy.

Let us consider the fraction of cases (out of 50) for which the considered approximation method has the smallest error (of considered type); see the results in Table 4. We can see that for most cases HDA achieves the smallest approximation errors.

Let us quantify to what extent on average the accuracy of HDA is higher than the accuracy of IMAGE. We denote by $E(M)$ an approximation error of some particular type, obtained by a method M on some test problem, $\Delta(M_1, M_2) = (E(M_1) - E(M_2))/E(M_1) \cdot 100\ \%$ for some two methods M_1 and M_2. Then the average gain (*AvgGain*) of using M_2 instead of M_1 is equal to the average of positive values of $\Delta(M_1, M_2)$ over all considered problems. Analogously, the average loss (*AvgLoss*) of using M_2 instead of M_1 is equal to the average of negative values of $\Delta(M_1, M_2)$ over all considered problems. The obtained results for IMAGE (M_1) and HDA (M_2) are given in Table 5. We can see that HDA is significantly more accurate.

Table 2 Accuracy Indicators for RF STR

	Error	IMAGE				HDA
		30 experts	40 experts	45 experts	50 experts	
$\hat{f}_1$ for RF STR						
$\mathbf{x} \in \mathbf{X}_{1,1}$	*Emean*	0.0579	0.0172	0.0279	0.0295	0.0046
	*Eq*95	0.2403	0.1115	0.1528	0.1665	0.0145
	*Eq*99	0.3435	0.1873	0.2453	0.2609	0.0362
$\mathbf{x} \in \mathbf{X}_{1,2}$	*Emean*	0.0982	0.0323	0.0462	0.0492	0.0078
	*Eq*95	0.3639	0.1427	0.1952	0.2122	0.0219
	*Eq*99	0.5378	0.3024	0.3588	0.3782	0.0893
$\hat{f}_2$ for RF STR						
$\mathbf{x} \in \mathbf{X}_{2,1}$	*Emean*	0.1105	0.1039	0.0990	0.0961	0.0056
	*Eq*95	0.3829	0.3636	0.3530	0.3451	0.0191
	*Eq*99	0.5785	0.5526	0.5350	0.5479	0.0599
$\mathbf{x} \in \mathbf{X}_{2,2}$	*Emean*	0.2395	0.2294	0.2254	0.2208	0.0186
	*Eq*95	0.7099	0.7079	0.6857	0.6734	0.0525
	*Eq*99	1.2116	1.2142	1.2116	1.1529	0.1807

Table 3 Accuracy indicators for RF PND GEN

	Error	IMAGE				HDA
		30 experts	40 experts	45 experts	50 experts	
$\hat{f}_1$ for RF PND GEN						
$\mathbf{x} \in \mathbf{X}_{1,1}$	*Emean*	0.0526	0.0575	0.0577	0.0596	0.0399
	*Eq*95	0.2092	0.2212	0.2134	0.2196	0.1136
	*Eq*99	0.3060	0.3153	0.2931	0.2991	0.1784
$\mathbf{x} \in \mathbf{X}_{1,2}$	*Emean*	0.0829	0.0849	0.0841	0.0854	0.0579
	*Eq*95	0.2656	0.2652	0.2528	0.2650	0.1753
	*Eq*99	0.4085	0.3960	0.3900	0.3981	0.2926
$\hat{f}_2$ for RF PND GEN						
$\mathbf{x} \in \mathbf{X}_{2,1}$	*Emean*	0.2679	0.2555	0.2500	0.2498	0.0530
	*Eq*95	0.6374	0.5808	0.5983	0.6011	0.1623
	*Eq*99	0.8298	0.7671	0.7793	0.7967	0.2655
$\mathbf{x} \in \mathbf{X}_{2,2}$	*Emean*	0.2731	0.2597	0.2538	0.2470	0.1039
	*Eq*95	0.7476	0.7033	0.6893	0.6745	0.3238
	*Eq*99	1.0702	1.0016	0.9907	0.9784	0.5705

Table 4 Fraction of cases

Error	HDA	IMAGE
$Emean$	0.853	0.147
$Eq95$	0.941	0.059
$Eq99$	1	0.000

Table 5 Average gain and loss

Error		HDA vs IMAGE
$Emean$	$AvgGain$ (%)	56.20
	$AvgLoss$ (%)	−8.02
$Eq95$	$AvgGain$ (%)	54.27
	$AvgLoss$ (%)	−1.65
$Emean$	$AvgGain$ (%)	48.56
	$AvgLoss$ (%)	0.00

5 Results of Optimization Based on Skill Tool and MSM

In this section some results of using SMs in the pre-sizing optimization tool COMBOX for composite structures are presented. The classical skill tool called PS3 (Plane Super-Stiffener Sizing) was replaced by the MACROS Surrogate Model (MSM). The objective of this study was to check the impact of this replacement on the accuracy, on the convergence of the optimization process, and on the run time.

5.1 Optimization Runs

In this study the optimization was performed on the wing lower and upper covers, as described in Fig. 8. Two test cases for the optimization study were considered corresponding to two starting points:

- The first one is close to an optimal design, obtained by using only the PS3 skill tool.
- The second one is a heavy one, where all design variables are set to their upper bound. This last run illustrates the behavior using SMs for a complete optimization run.

For each test case, the first run was done with MSM (until convergence), and then update and restart with PS3 was performed. Due to limited space only some representative results are shown.

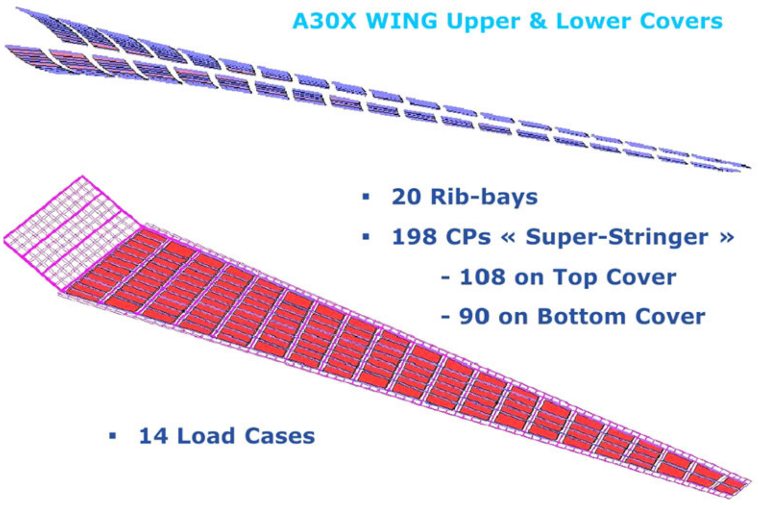


Fig. 8 A30X wing stress model

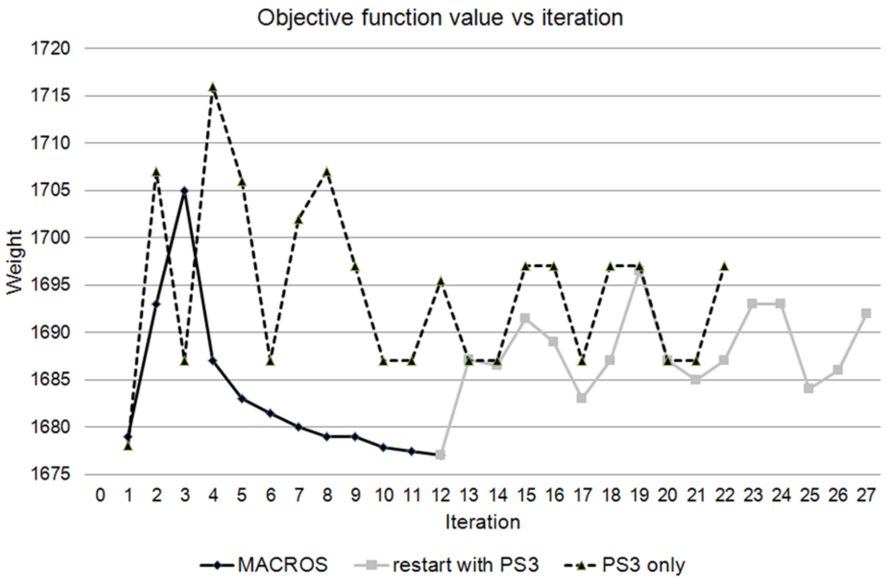


Fig. 9 Evolution of the objective function for initial starting point

5.1.1 Optimization Runs: Initial Starting Point

The comparison with the *pure* PS3 run is presented in Fig. 9 (solid black-gray curve corresponds to MSM and subsequent update with PS3; dashed curve corresponds to the run with only PS3).

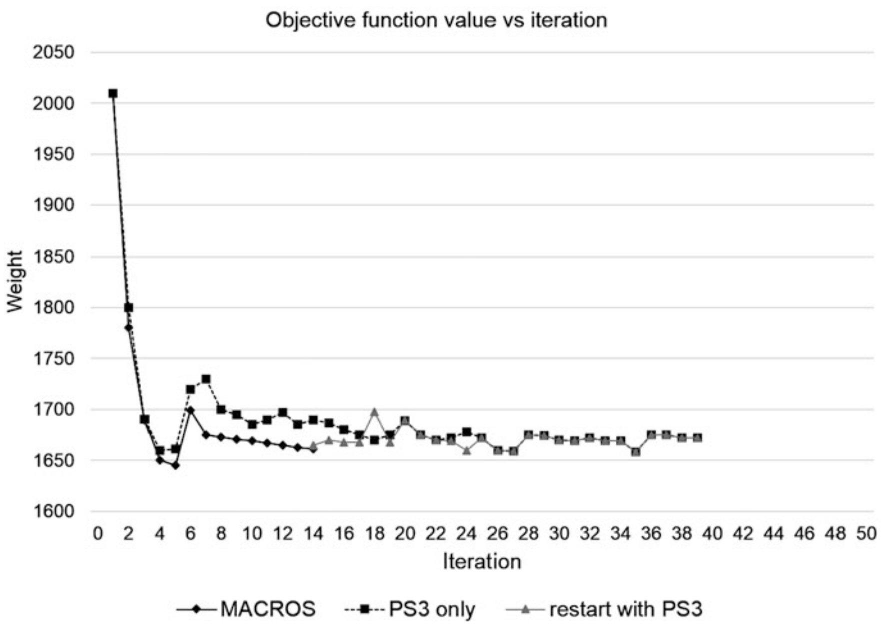


Fig. 10 Evolution of the objective function for heavy starting point

We can observe a more chaotic evolution with PS3 which is not only due to proximity to the optimal solution, but is also linked to some discontinuities in the RFs calculated with PS3.

5.1.2 Optimization Runs: Heavy Starting Point

The evolution of the objective value is given in Fig. 10 (black solid curve corresponds to MSM, gray solid curve corresponds to restart from MSM with PS3, dashed curve corresponds to PS3 only). We observe a smooth evolution with both MSM (SM is a continuous function) and PS3 (the behavior of PS3 is smoother than in the previous run, since the starting point is not as close to the optimal design), but the evolution is nevertheless smoother with MSM.

The evolution of the numbers of violated constraints is given in Fig. 11 and saturated constraints is given in Fig. 12. The peak at the beginning of the left plot appears due to the strategy of active constraints. We can observe an increase of the number of violated constraints when updating the model (due to the switch of skill tool) and after a quick decrease of it. A decrease of the number of saturated constraints at iteration six occurs due to the reactualization of the active constraints and the identification of new violated constraints; a decrease of the number of saturated constraints from MSM to restart with PS3 occurs due to the switch of the skill tool. The restart with PS3 presents less saturated constraints than MSM. This proves a

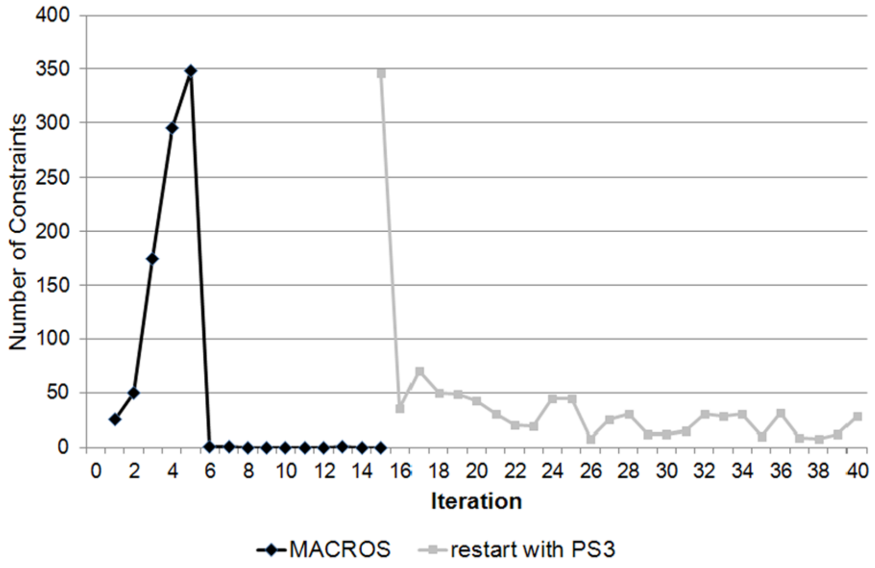


Fig. 11 Evolution of the number of violated constraints

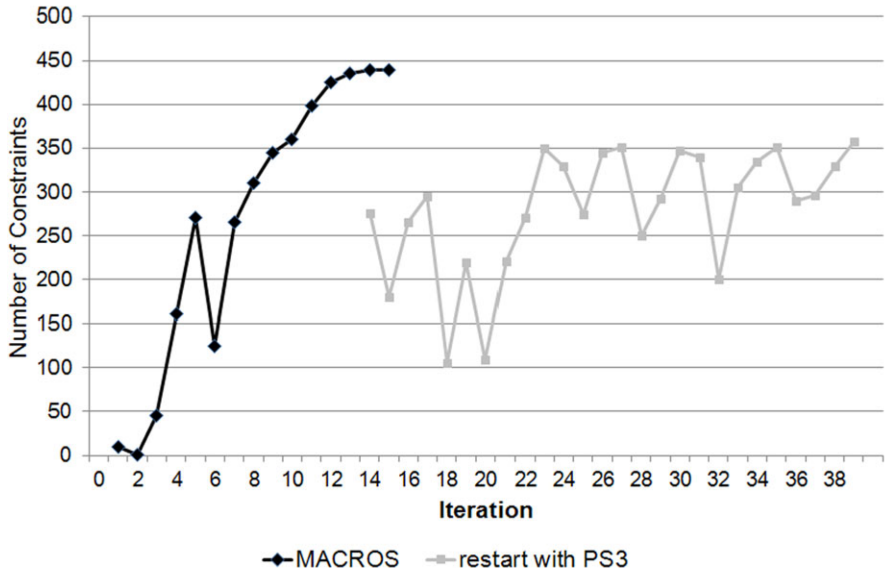
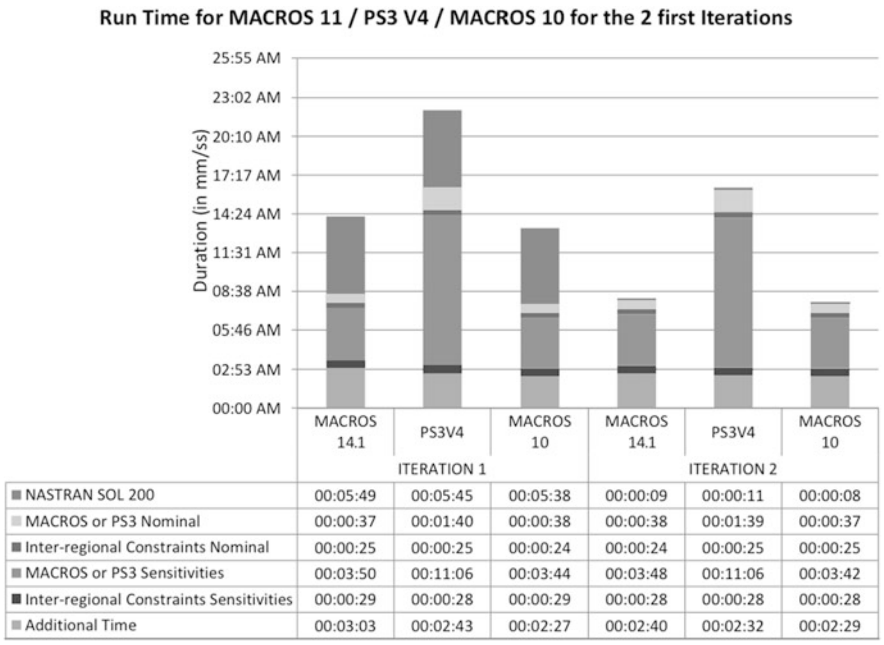


Fig. 12 Evolution of the number of saturated constraints

better quality of the optimum found with MSM, which is linked to the smoothness and mathematical differentiability of MSM, in contrast to PS3.



	MACROS 14.1	PS3V4	MACROS 10	MACROS 14.1	PS3V4	MACROS 10
	ITERATION 1			ITERATION 2		
▪ NASTRAN SOL 200	00:05:49	00:05:45	00:05:38	00:00:09	00:00:11	00:00:08
▫ MACROS or PS3 Nominal	00:00:37	00:01:40	00:00:38	00:00:38	00:01:39	00:00:37
▪ Inter-regional Constraints Nominal	00:00:25	00:00:25	00:00:24	00:00:24	00:00:25	00:00:25
▪ MACROS or PS3 Sensitivities	00:03:50	00:11:06	00:03:44	00:03:48	00:11:06	00:03:42
■ Inter-regional Constraints Sensitivities	00:00:29	00:00:28	00:00:29	00:00:28	00:00:28	00:00:28
▫ Additional Time	00:03:03	00:02:43	00:02:27	00:02:40	00:02:32	00:02:29

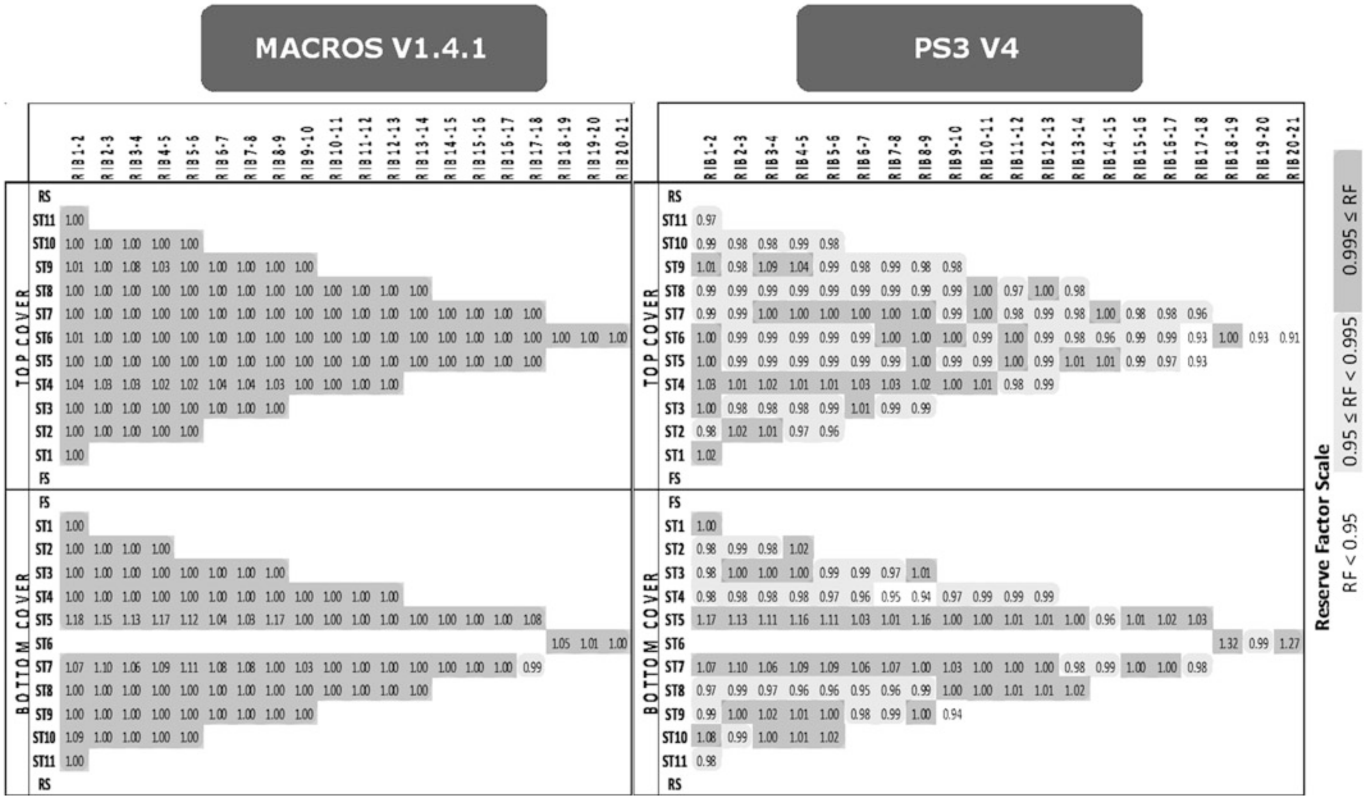
Fig. 13 Run Time (hh/mm/ss) for MACROS 14.1/PS3 V4/MACROS 10 on iterations 1 and 2

5.2 *Execution Time*

The objective of this study was to compare the run time of the MSM and PS3 skill tools for one iteration of the optimization process. The study was done on test case 1 (initial point), during the first two iterations. The sequence of computations for one iteration is the following:

- NASTRAN SOL200: for internal load update and sensitivity analysis,
- Computation of the RFs corresponding to the current design with the PS3 stress process or the MSM,
- Computation of the interregional constraints corresponding to the current design,
- Computation (by using PS3 or MSM) of the sensitivities for all the RFs,
- Computation of the sensitivities for the interregional constraints.

We observed an overall gain factor of at least 2.5 by using MSM instead of PS3 (because of the acceleration of the steps *MACROS or PS3 Nominal* and *MACROS or PS3 Sensitivities*, see Fig. 13). Since MSM provides very fast analytical computation of sensitivities, further significant speedup can be obtained by using analytical sensitivity computations instead of numerical ones. The speedup factor is less than expected, because much system time is spent in the management of jobs via IBM Platform LSF (Load Sharing Facility, a workload management platform for demanding, distributed HPC environments, see [22]).



MACROS V1.4.1

		RIB 1-2	RIB 2-3	RIB 3-4	RIB 4-5	RIB 5-6	RIB 6-7	RIB 7-8	RIB 8-9	RIB 9-10	RIB 10-11	RIB 11-12	RIB 12-13	RIB 13-14	RIB 14-15	RIB 15-16	RIB 16-17	RIB 17-18	RIB 18-19	RIB 19-20	RIB 20-21
TOP COVER	RS																				
	ST11	1.00																			
	ST10	1.00	1.00	1.00	1.00	1.00															
	ST9	1.01	1.00	1.08	1.03	1.00	1.00	1.00	1.00	1.00											
	ST8	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00							
	ST7	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00			
	ST6	1.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	ST5	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00			
	ST4	1.04	1.03	1.03	1.02	1.02	1.04	1.04	1.03	1.00	1.00	1.00	1.00								
	ST3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00												
	ST2	1.00	1.00	1.00	1.00	1.00															
	ST1	1.00																			
	FS																				
BOTTOM COVER	FS																				
	ST1	1.00																			
	ST2	1.00	1.00	1.00	1.00																
	ST3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00												
	ST4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00								
	ST5	1.18	1.15	1.13	1.17	1.12	1.04	1.03	1.17	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.08			
	ST6																		1.05	1.01	1.00
	ST7	1.07	1.10	1.06	1.09	1.11	1.08	1.08	1.00	1.03	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99			
	ST8	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00							
	ST9	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00											
	ST10	1.09	1.00	1.00	1.00	1.00															
	ST11	1.00																			
	RS																				

PS3 V4

		RIB 1-2	RIB 2-3	RIB 3-4	RIB 4-5	RIB 5-6	RIB 6-7	RIB 7-8	RIB 8-9	RIB 9-10	RIB 10-11	RIB 11-12	RIB 12-13	RIB 13-14	RIB 14-15	RIB 15-16	RIB 16-17	RIB 17-18	RIB 18-19	RIB 19-20	RIB 20-21
TOP COVER	RS																				
	ST11	0.97																			
	ST10	0.99	0.98	0.98	0.99	0.98															
	ST9	1.01	0.98	1.09	1.04	0.99	0.98	0.99	0.98	0.98											
	ST8	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	0.99	1.00	0.97	1.00	0.98							
	ST7	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	0.98	0.99	0.98	1.00	0.98	0.98	0.96			
	ST6	1.00	0.99	0.99	0.99	0.99	0.99	1.00	1.00	1.00	0.99	1.00	0.99	0.98	0.96	0.99	0.99	0.93	1.00	0.93	0.91
	ST5	1.00	0.99	0.99	0.99	0.99	0.99	0.99	1.00	0.99	0.99	1.00	0.99	1.01	1.01	0.99	0.97	0.93			
	ST4	1.03	1.01	1.02	1.01	1.01	1.03	1.03	1.02	1.00	1.01	0.98	0.99								
	ST3	1.00	0.98	0.98	0.98	0.99	1.01	0.99	0.99												
	ST2	0.98	1.02	1.01	0.97	0.96															
	ST1	1.02																			
	FS																				
BOTTOM COVER	FS																				
	ST1	1.00																			
	ST2	0.98	0.99	0.98	1.02																
	ST3	0.98	1.00	1.00	1.00	0.99	0.99	0.97	1.01												
	ST4	0.98	0.98	0.98	0.98	0.97	0.96	0.95	0.94	0.97	0.99	0.99	0.99								
	ST5	1.17	1.13	1.11	1.16	1.11	1.03	1.01	1.16	1.00	1.00	1.01	1.01	1.00	0.96	1.01	1.02	1.03			
	ST6																		1.32	0.99	1.27
	ST7	1.07	1.10	1.06	1.09	1.09	1.06	1.07	1.00	1.03	1.00	1.00	1.00	0.98	0.99	1.00	1.00	0.98			
	ST8	0.97	0.99	0.97	0.96	0.96	0.95	0.96	0.99	1.00	1.00	1.01	1.01	1.02							
	ST9	0.99	1.00	1.02	1.01	1.00	0.98	0.99	1.00	0.94											
	ST10	1.08	0.99	1.00	1.01	1.02															
	ST11	0.98																			
	RS																				

Reserve Factor Scale

RF < 0.95	0.95 ≤ RF < 0.995	0.995 ≤ RF

Fig. 14 Check with PS3 of optimum results found with MACROS SM

5.3 *Check of Reserve Factors*

A check of RFs was performed with the optimum based on MSM. The results are presented in Fig. 14. These results show a satisfactory accuracy for a pre-sizing result, according to Airbus experts, considering that a pre-sizing is always to be re-engineered including, e.g., manufacturing constraints. Work is ongoing to further improve the accuracy.

6 Conclusion

A constructed MACROS Surrogate Model (MSM) was embedded into the pre-sizing optimization process of A350XWB boxes realized in the pre-sizing tool COMBOX for checking the validity of the approximation and its usage as a constraint in an optimization process. An analysis/comparison of optimization results based on a skill tool and optimization results based on the constructed MSM was performed and showed that MSM:

- gives a high accuracy of approximation (see also [6]),
- allows one to obtain smoother convergence in fewer iterations with a smoother distribution of thickness/stringer dimensions and a small violation of constraints, which could be easily repaired at the detailed design phase,
- provides a *reduction of structure optimization computational time* from several days to a few hours.

Acknowledgements E. Burnaev, M. Belyaev, and P. Prikhodko were partially supported by the Laboratory for Structural Methods of Data Analysis in Predictive Modeling, MIPT, RF government grant, ag. 11.G34.31.0073.

References

1. Belyaev, M., Lyubin, A.: Some features of optimization problem arising in construction of multidimensional approximation. In: Proceedings of the Conference for Young Scientists and Engineers “Information Technology and Systems—2011”, Gelendzhik, Russia, pp. 415–422 (2011)
2. Belyaev, M.G., Burnaev, E.V., Lyubin, A.D.: Parametric dictionary selection for approximation of multidimensional dependency. In: Proceedings of All-Russian Conference “Mathematical Methods of Pattern Recognition (MMPR-15)”, Petrozavodsk, 11–17 September 2011, pp. 146–150 (2011)
3. Belyaev, M., Burnaev, E., Yerofeyev, P., Prikhodko, P.: Comparative performance of nonlinear regression initialization methods. In: Proceedings of the Conference for Young Scientists and Engineers “Information Technology and Systems—2011”, Gelendzhik, Russia, pp. 315–320 (2011)
4. Bettebghor, D., Bartoli, N., Grihon, S., Morlier, J., Samuelides, M.: Surrogate modelling approximation using a mixture of experts based on EM joint estimation. Struct. Multidiscip. Optim. **43**(2), 243–259 (2011)

5. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Berlin (2006)
6. Burnaev, E.V., Grihon, S.: Construction of the metamodels in support of stiffened panel optimization. In: Proceedings of the International Conference on Mathematical Methods in Reliability, Moscow, Russia, pp. 124–128 (2009)
7. Burnaev, E., Prikhodko, P.: Theoretical properties of procedure for construction of regression ensemble based on bagging and boosting. In: Proceedings of the Conference for Young Scientists and Engineers "Information Technology and Systems—2011", Gelendzhik, Russia, pp. 438–443 (2011)
8. Burnaev, E., Prikhodko, P.: Approaches to dimensionality estimation in the effective dimensionality reduction based on Gaussian processes. In: Information Technologies and Systems 2012, Petrozavodsk, August 19–25. Springer, Berlin (2012)
9. Burnaev, E.V., Belyaev, M.G., Prihodko, P.V.: About hybrid algorithm for tuning of parameters in approximation based on linear expansions in parametric functions. In: Proceedings of the 8th International Conference "Intellectualization of Data Processing" (IDP-2010), Cyprus, 17–24 October 2011, pp. 200–203 (2011)
10. Burnaev, E.V., Zaytsev, A.A., Yanovich, Yu.A.: Consolidation of data with different fidelity on basis of Gaussian processes. In: Proceedings of the Conference "Information Technologies and Systems", pp. 60–65 (2012)
11. Chiles, J.-P., Delfiner, P.: Geostatistics. Modeling Spatial Uncertainty. Wiley, New York (1999)
12. DATADVANCE: www.datadvance.net
13. Dolan, E., More, J.: Benchmarking optimization software with performance profiles. Math. Program., Ser. A **91**, 201–213 (2002)
14. Forrester, A., Sobester, A., Keane, A.: Engineering Design via Surrogate Modeling. A Practical Guide. Wiley, New York (2008)
15. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, Berlin (2008)
16. Haykin, S.S.: Neural Networks: A Comprehensive Foundation. Williams, Baltimore (2006)
17. http://en.wikipedia.org/wiki/Matlab
18. http://en.wikipedia.org/wiki/ModeFRONTIER
19. http://en.wikipedia.org/wiki/Nastran
20. http://www.lmsintl.com/simulation/caesam
21. http://www.lmsintl.com/samtech-boss-quatro
22. http://www-03.ibm.com/systems/technicalcomputing/platformcomputing/products/lsf/index.html
23. Kuleshov, A.P., Bernstein, A.V., Burnaev, E.V.: Adaptive models of complex systems based on data handling. In: Proceedings of the Third International Conference on Inductive Modelling, Kyiv, Ukraine, pp. 64–71 (2010)
24. Pinkus, A.: Approximation theory of the MLP model in neural networks. Acta Numer. **8**, 143–195 (1999)
25. Rasmussen, C.E., Williams, C.K.I.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
26. Wang, G., Shan, S.: Review of metamodeling techniques in support of engineering design optimization. J. Mech. Des. **129**(3), 370–381 (2007)