Parallel Multipoint Approximation Method for Large-Scale Optimization Problems*

Victor P. Gergel[1], Konstantin A. Barkalov[1], Evgeny A. Kozinov[1], and Vassili V. Toropov[1,2]

[1]Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
{victor.gergel, konstantin.barkalov, evgeny.kozinov}@itmm.unn.ru
[2]Queen Mary University of London, London, UK
v.v.toropov@qmul.ac.uk

Abstract. The paper presents a new development in the Multipoint Approximation Method (MAM) that makes it capable of handling large-scale problems. The approach relies on approximations built in the space of design variables within the iterative trust-region-based framework of MAM. With the purpose of solving high dimensionality problems in a reasonable time, a parallel variant of the Multipoint Approximation Method (PMAM) has been developed. It is supposed that the values of the objective function and those of the constraints are computed using distributed memory (on several cluster nodes), whereas the optimization module runs on a single node using shared memory. Numerical experiments have been carried out on a benchmark example of structural optimization.

Keywords: design optimization, multidisciplinary optimization, multipoint approximation method, parallel computing.

1 Introduction

In the present paper, the multipoint approximation method (MAM) [1–3] and its application to large-scale optimization problems are considered. In problems with a large (in the order of hundreds) number of design variables, MAM has proved to be efficient, e.g., in turbomachinery applications [4–6]. This method is an iterative optimization technique based on mid-range approximations built in trust regions. A trust region is a subdomain of the design space in which a set of design points, produced according to a small-scale design of experiments (DoE), is evaluated. These and a subset of previously evaluated design points are used to build metamodels of the objective and constraint functions that are considered to be valid within a current trust region. The trust region will then translate and change size as optimization progresses. The trust region strategy has gone through several stages of development to account for the presence of numerical

* This study was supported by the Russian Science Foundation, project No. 16-11-10150.

noise in the response function values [7, 8] and occasional simulation failures [9]. The mid-range approximations used in the trust regions, as originally suggested in [1] for structural optimization problems, are intrinsically linear functions (i.e. nonlinear functions that can be reduced to a linear form by a simple transformation) for individual substructures, and an assembly of them for the whole structure. This was enhanced by the use of gradient-assisted metamodels [3], the use of simplified numerical models which is also termed the multi-fidelity approach [10], and the use of analytical models derived by genetic programming [11]. One of the recent developments [12] involves the use of approximation assemblies, i.e. a two stage approximation building process that is conceptually similar to the original one used in [1] but is free from the limitation that lower level approximations are linked to individual substructures.

The Moving Least-Squares Method (MLSM) was proposed in [13] for smoothing and interpolation of scattered data and was later used in the mesh-free form of the finite element method (FEM) [14]. As suggested in [15], it can be used as a technique for metamodeling and in multidisciplinary optimization (MDO) frameworks. The MLSM is a weighted least-squares method where the weights depend on the Euclidean distance from a sample point to where the surrogate model is to be evaluated. The weight value for a certain sample point decays as the distance increases. Describing the weight decay with a Gaussian function tends to be the most useful option, even though many others have been evaluated in [16]. As demonstrated in [17], the cross-validated MLSM can be used both for design variable screening and for surrogate modeling. In order to create an efficient MDO framework for problems with disparate discipline attributes, the optimization approach of MAM was extended in [18] to the use of local DOEs and MLS approximations built in different subspaces of the total design variable space corresponding to the individual disciplines. The subspaces are finally combined into the total design variable space in which the resulting MDO problem is solved.

This paper presents a Parallel Multipoint Approximation Method that makes it capable of handling problems with numbers of design variables in the order of thousands. The parallel variant of the algorithm (with the use of shared memory) has been developed with the purpose of minimizing the work time of the part of the algorithm related to constructing the approximation and solving the approximated problem, but not related to computing the values of the objective function and constraints. The processes of computing the values of the objective function and constraints (with the use of distributed memory) are supposed to be already parallelized.

2 The Multipoint Approximation Method

It would be useful to start with a brief description of MAM. A typical formulation of a constrained optimization problem that MAM works with is as follows:

$$\begin{gathered} \min_{a_i \le x_i \le b_i} F_0(x) \\ \text{s.t. } F_j(x) \le 1,\ j = 1, \dots, M, \end{gathered} \tag{1}$$

where x is a vector of design variables, a and b are the lower and upper bounds for the design variables, respectively, $F_0(x)$ is the objective function, and $F_j(x)$ are the constraints. The numbers of design variables and constraints are n and M, respectively. MAM attempts to solve this problem by using approximations of the objective function and constraints in a series of trust regions. The trust region strategy seeks to zoom in on the region where the constrained minimum is achieved. It aims at finding a trust region that is sufficiently small for the approximations to be of sufficiently good quality to improve the design and contains the point of the constrained minimum as an interior point. The main loop of the MAM is organized as follows.

Algorithm (MAM).

1. Initialization: choose a starting point x^0 and initial trust region $[a^0, b^0]$ such that $x^0 \in [a^0, b^0]$.
2. On the kth iteration, the current approximation to the constrained minimum is x^k, and the current trust region is $[a^k, b^k] \subset [a^0, b^0]$.
   (a) Design of Experiments (DoE). A set of points $x_k^i \in [a^k, b^k]$ is chosen to be used for building approximations. Responses are evaluated at the DoE points and approximations are built using the obtained values. Currently, the pool of approximation methods available in MAM consists of metamodel assemblies [12] and the moving least-squares metamodels [13–16]. Other metamodel types could be used as well.
   Denote the approximate objective function and constraints by $\widetilde{F}_0^k(x)$ and $\widetilde{F}_j^k(x)$, respectively.
   (b) The original optimization problem (1) is replaced by the following:
$$\begin{gathered} \min_{a_i^k \le x_i^k \le b_i} \widetilde{F}_0^k(x) \\ \text{s.t. } \widetilde{F}_j^k(x) \le 1,\ j = 1, \ldots, M. \end{gathered} \tag{2}$$
   The approximate problem (2) is solved using Sequential Quadratic Programming (SQP). The solution of this problem determines the center of the next trust region.
   (c) The size of the next trust region is determined depending on the quality of approximations on the previous iteration, on the history of points x^k, and on the size of the current trust region [7].
   (d) The termination criterion is checked (it is a part of the trust region strategy and depends on the position of the point x^{k+1} in the current trust region, the size of the current trust region and the quality of approximations). If the termination criterion is satisfied, the algorithm proceeds to step 3. Otherwise, it returns to step 2.
3. Optimization terminates. The obtained approximation to the solution of problem (1) is x^{k+1}.

The approximations $\widetilde{F}_j^k(x)$, $j = 0, \ldots, M$, are selected in such a way that their evaluation is inexpensive as compared to the evaluation of the original response functions $F_j(x)$. For example, intrinsically linear functions were successfully used for a variety of design optimization problems in [3, 19]. The approximations are determined by means of the weighted least squares:

$$\min \sum_{p=1}^{P} w_{pj} \left[F_j(x_p) - \widetilde{F}_j^k(x_p, a_j) \right]^2 . \tag{3}$$

In (3), minimization is carried out with respect to the tuning parameters a_j; w_{pj} are the weight coefficients, and P is the number of sampling points in Design of Experiments (DoE), which must not be less than the number of parameters in the vector a_j.

The weight coefficients w_{pj} strongly influence the difference in the quality of the approximations in different regions of the design variable space. Since in realistic constrained optimization problems the optimum point usually belongs to the boundary of the feasible region, the approximation functions should be more accurate in such domain. Thus, the information at the points located near the boundary of the feasible region is to be treated with greater weights. In a similar manner, a larger weight can be allocated to a design with a better objective function (see [3, 19]).

As optimization steps are carried out, a database with response function values becomes available. In order to achieve good quality approximations in the current trust region, an appropriate selection of DoE points must be made. In this work, DoE points in each trust region are generated randomly. Generally, points located far from the current trust region would not contribute to the improvement of the quality of the resulting approximations in the trust region. For this reason, only points located in a neighborhood of the current trust region are taken into account, as depicted in Fig. 1. A box in the space of design variables,

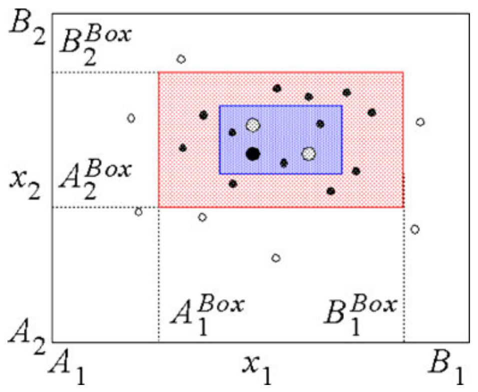

Fig. 1. Current trust region (smaller box) and its extension (larger box): points outside the larger box are not used for building the approximate functions

which is approximately 1.5 to 1.8 times larger than the box representing the current trust region, was found by numerical experimentation to be a reasonable choice for the size of the neighborhood.

In this work, an approach is used that is based on the assembly of different approximate models $\{\varphi_l\}$ into one metamodel using the following form (note that the indices j and k are suppressed to simplify notation):

$$\widetilde{F}(x) = \sum_{l=1}^{NF} b_l \varphi_l(x), \tag{4}$$

where NF is the number of regressors in the model pool $\{\varphi_l\}$, and b_l are the corresponding regression coefficients. The procedure used consists of two subsequent steps. In the first step, the parameters a_l of individual functions (regressors) φ_l in (4) are determined by solving a weighted least-squares problem using a specified DoE of P points:

$$\min \sum_{p=1}^{P} w_p \left[F(x_p) - \varphi_l(x_p, a_l)\right]^2,$$

where minimization is carried out with respect to the tuning parameters a_l.

In the second step, based on the same DoE and keeping the obtained parameters a_l fixed, a vector b in (4) is estimated using the following formulation:

$$\min \sum_{p=1}^{P} w_p \left[F(x_p) - \widetilde{F}(x_p, b)\right]^2,$$

which leads to solving a linear system of NF equations with NF unknowns b_l, where NF is the number of regressors in the model pool $\{\varphi_l\}$.

The selection of the regressors $\{\varphi_l\}$ is based on the number of sampling points currently located in the trust region. In the mid-range approximation framework, inexpensive approximate models for objective and constraint functions are built using the minimum required number of sampling points. The simplest case is that of a linear function of the tuning parameters a:

$$\varphi(x) = a_0 + \sum_{i=1}^{N} a_i x_i.$$

This structure can be extended to an *intrinsically linear* function. Such functions are nonlinear but they can be reduced to linear ones by simple transformations. The most useful function among them is the multiplicative function

$$\varphi(x) = a_0 \prod_{i=1}^{N} {x_i}^{a_i}.$$

Intrinsically linear functions have been successfully used for a variety of design optimization problems. The advantage of these approximation functions is that a relatively small number $N + 1$ (N is the number of design variables) of tuning parameters a_i is to be determined, and the corresponding least-squares problem is solved easily. This is the most important feature of such approximations as it allows applying them to large-scale optimization problems.

Other intrinsically linear functions may be considered in the model pool, e.g.,

$$\varphi(x) = a_0 + \sum_{i=1}^{N} a_i/x_i,$$

$$\varphi(x) = a_0 + \sum_{i=1}^{N} a_i x_i^2,$$

$$\varphi(x) = a_0 + \sum_{i=1}^{N} a_i/x_i^2,$$

$$\varphi(x) = a_0 + \sum_{i=1}^{N} a_i x_i^3,$$

$$\varphi(x) = a_0 + \sum_{i=1}^{N} a_i/x_i^3.$$

As more points are added to the database, the approximations may be switched to higher quality models, e.g., a rational model

$$\varphi(x) = \frac{a_1 + a_2 x_1 + a_3 x_2 + ... + a_{n+1} x_n}{1 + a_{n+2} x_1 + a_{n+3} x_2 + ... + a_{2n+1} x_n}. \tag{5}$$

The coefficients in (5) are determined using a least-squares approach which reduces to a nonlinear optimization problem with a constraint on the sign of the denominator (positive or negative). The latter is necessary in order to prevent the denominator from crossing the zero axis within a specified trust region. One may note that this formulation may yield an objective function with many local minima. Currently, this problem is resolved using optimization restarts from a specified number of initial guesses randomly generated in a trust region.

Tests results demonstrated that, although the above functions may describe the global behavior rather poorly, such approximations prove to be efficient in the mid-range approximation framework of MAM.

3 Parallel Multipoint Approximation Method

Let us consider possible methods of parallelization that could be applied to the problems considered.

First, one can parallelize the computation of the functions describing the optimized object. This way is an obvious as well as necessary one since, in industrial design optimization problems, the computation of even a single function value may take several hours. However, this method is a specific one for each particular problem. Here the computation issues are addressed at the level of the application software in which the industrial modeling is performed (e.g., Ansys, OpenFOAM, etc.).

Second, one can correct the algorithm with the purpose of parallelly computing several values of the objective function and constraints at different points of the search domain. According to the MAM rules, in design of experiments, P sampling points are formed in the current trust region at each iteration. The function values at these points can be computed on different processors (nodes) in parallel. The above corresponds to the parallelization of Step 2a of the algorithm using distributed memory. The number of sampling points generated within each iteration may be set equal to $P = k \cdot NP$, where NP is the number of available processors (or nodes), and $k \geq 1$. In terms of time, the latter will be equivalent to NP function evaluations per step. This method has been implemented successfully in [20] and has demonstrated a good efficiency since, for numbers of design variables of the order of 100, time is mainly consumed by the function evaluations, whereas the work of the MAM itself (in the sequential regime) introduces a minor overhead.

However, when the number of variables becomes of the order of 1000, the work of the sequential part of MAM begins to affect the total problem solving time essentially (assuming that the time of computing the objective function values remains constant). Thus, for the problem considered in Section 4, the time of execution of a single iteration of the method increased by a factor of more than 4000 (from 1.5 up to 6000 seconds) when increasing the number of variables from 100 up to 1000. Another approach to the parallelization of the algorithm has therefore been applied within the framework of the present study: namely, the computational rules of MAM providing for the construction of the approximation, the solution of the approximated problem, and the choice of the next trust region (Steps 2b, 2c, and 2d of the algorithm) were parallelized.

In order to find the most time-consuming parts of the sequential program developed earlier, we applied the Intel VTune Amplifier XE. The analysis performed has shown that the most time-consuming operations in the execution of MAM are matrix multiplication, the solution of the SLAE when constructing the approximations, and the solution of the approximating problem by SQP. Matrix multiplication and solution of SLAE are standard operations implemented in many high-performance libraries. Here we used the corresponding parallel methods from the Intel MKL library. We parallelized the SQP method ourselves using OpenMP.

4 Numerical Example

The example considered in this study is a classical engineering optimization problem known as the scalable cantilevered beam [21]. The engineering object to be optimized is shown in Fig. 2 (taken from [21]).

The design variables are the widths b_i and heights h_i of the segments. The number of segments N can be chosen arbitrarily. The total length of the beam is

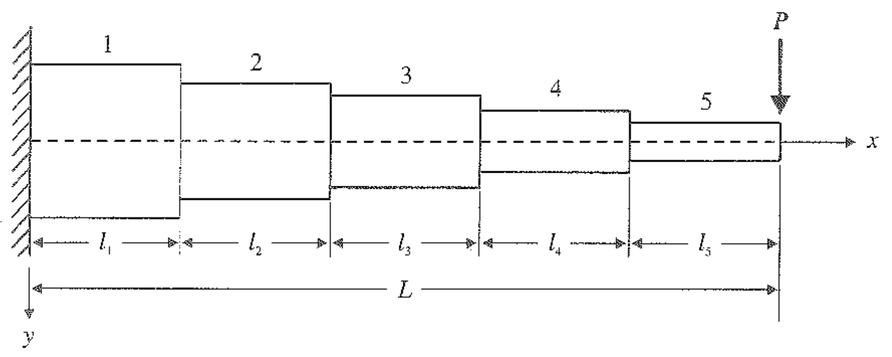

Fig. 2. The cantilevered beam

500 cm, the lengths of the segments are $l_i = 500/N$ cm. There are N geometric constraints (the aspect ratios of the blocks, i.e. heights divided by widths, should not exceed 20) and N constraints on the stress, calculated at the left end of each segment (stresses should not exceed $\bar{\sigma} = 14\,000$ N/cm^2). There is also a constraint on the displacement at the tip, which should not exceed 2.5 cm. The load is $P = 50\,000$ N; the Young's modulus is $E = 2 \cdot 10^7$ N/cm^2.

The deflection y_i at the right end of the ith segment is given by the following recursive formulas:

$$y_0 = y'_0 = 0,$$
$$y'_i = \tfrac{P \cdot l_i}{E \cdot I_i}\left[L + \tfrac{l_i}{2} - \sum_{j=1}^{i} l_j\right] + y'_{i-1},$$
$$y_i = \tfrac{P \cdot l_i^2}{2E \cdot I_i}\left[L - \sum_{j=1}^{i} l_j + \tfrac{2l_i}{3}\right] + y'_{i-1} l_i + y_{i-1}.$$

The moment of inertia of the ith segment is $I_i = b_i h_i^3/12$, and the bending moment at its left end is $M_i = P[L + l_i - \sum_{j=1}^{i} l_j]$. The maximum bending stress in the ith segment is then given by the following formula:

$$\sigma_i = \frac{M_i h_i}{2I_i}$$

We should look for a design of smallest volume $V = \sum_{i=1}^{N} b_i h_i l_i$. The widths b_i vary from 1.0 to 10.0 cm and the heights h_i from 5.0 to 100.0 cm. The optimization problem is formulated as follows:

$$\begin{aligned} &\min_{b,h} V(b,h)\\ \text{s.t. } 1.0 \le\ &b_i \le 10.0,\\ 5.0 \le\ &h_i \le 100.0,\\ &y_N \le 2.5,\\ &\sigma_i \le \bar{\sigma} = 14000,\\ &\tfrac{h_i}{b_i} \le 20. \end{aligned}$$

With $N = 50$ segments (corresponding to 100 design variables), the SQP solution of the problem is $V = 63704.598$ cm^3. The optimal values of the design variables are given below (the first 50 entries are the widths, and the last 50 entries are the heights of the segments):

$$\begin{aligned} b = [&3.246,\ 3.224, 3.202, 3.179, 3.156, 3.133, 3.109, 3.085, 3.061,\ 3.036,\\ &3.011,\ 2.985, 2.959, 2.933, 2.905, 2.878, 2.850, 2.821, 2.792,\ 2.762,\\ &2.731,\ 2.700, 2.668, 2.635, 2.602, 2.567, 2.532, 2.495, 2.458,\ 2.419,\\ &2.379,\ 2.338, 2.295, 2.250, 2.204, 2.156, 2.105, 2.052, 1.996,\ 1.936,\\ &1.873,\ 1.805, 1.732, 1.651, 1.562, 1.462, 1.345, 1.196, 1.023,\ 1.000]\,, \end{aligned}$$

$$\begin{aligned} h = [&64.919,\ 64.480, 64.033, 63.581, 63.122, 62.656, 62.184, 61.703, 61.216,\ 60.720\\ &60.216,\ 59.703, 59.182, 58.651, 58.110, 57.558, 56.996, 56.423, 55.837,\ 55.239\\ &54.628,\ 54.003, 53.363, 52.707, 52.035, 51.344, 50.635, 49.906, 49.155,\ 48.381\\ &47.582,\ 46.754, 45.897, 45.008, 44.082, 43.116, 42.105, 41.041, 39.919,\ 38.729\\ &37.462,\ 36.102, 34.630, 33.020, 31.240, 29.241, 26.910, 23.919, 20.465,\ 14.639]\,. \end{aligned}$$

By MAM, we obtained the solution $V = 63935.360$ cm^3, using 2201 function evaluations (as compared to almost 10 000 evaluations used by SQP). The number of points in the trust region used to build the approximations was 200. The optimal values of the design variables obtained by MAM are given below:

$$\begin{aligned} b = [&3.238,\ 3.204, 3.202, 3.170, 3.176, 3.128, 3.108, 3.089, 3.057,\ 3.003\\ &3.027,\ 2.986, 2.959, 2.952, 2.885, 2.855, 2.864, 2.847, 2.803,\ 2.762\\ &2.737,\ 2.722, 2.630, 2.645, 2.590, 2.558, 2.547, 2.480, 2.693,\ 2.391\\ &2.368,\ 2.310, 2.307, 2.227, 2.176, 2.149, 2.106, 2.016, 2.007,\ 1.925\\ &1.864,\ 1.843, 1.758, 1.635, 1.582, 1.934, 1.332, 1.173, 1.026,\ 2.419]\,, \end{aligned}$$

$$\begin{aligned} h = [&64.764,\ 64.083, 64.036, 63.392, 63.518, 62.566, 62.153, 61.772, 61.149,\ 60.054\\ &60.534,\ 59.728, 59.182, 59.033, 57.695, 57.105, 57.274, 56.943, 56.057,\ 55.233\\ &54.747,\ 54.433, 52.590, 52.902, 51.802, 51.150, 50.943, 49.597, 51.567,\ 47.811\\ &47.373,\ 46.190, 46.140, 44.543, 43.525, 42.991, 42.119, 40.308, 40.141,\ 38.506\\ &37.268,\ 36.839, 35.153, 32.702, 31.615, 28.304, 26.642, 23.447, 20.536,\ 9.417]\,. \end{aligned}$$

The solution obtained by MAM is very close to the reference solution obtained by SQP, except for the last design variable (the height of the last segment), which indicates that the problem is insensitive to this variable near the optimum, making it hard for metamodels to capture this dependence. Both SQP and MAM solutions are, however, feasible and differ only slightly in the value of the objective function.

Next, let us compare the work time of the sequential and parallel algorithms when solving large-scale problems. The dimensionality N of the problem being solved was varied from 100 up to 1000 design variables, which corresponds to a variation of the number of segments of the cantilevered beam from 50 to 500. The number of points in the trust region used to build the approximations was $2N$. Both algorithms were run on a single node of the cluster (the specifications of the node are listed below), the parallel algorithm employed all 16 processors cores available. Since the time of computing the objective function and the constraints in the test problem was negligible, these were computed on the same node.

Table 1. Time and speedup

N	T_{MAM}	T_{PMAM}	Speedup
100	15.4	11.9	1.3
200	167	89	1.9
400	2476	837	3.0
600	9678	2777	3.5
800	27826	6948	4.0
1000	67674	13771	4.9

Table 2. Function evaluations

N	I_{MAM}	F_{MAM}	F_{SQP}
100	10	2201	9901
200	10	4402	28143
400	11	9599	84211
600	10	13200	146046
800	10	17600	221079
1000	10	22000	305308

Table 1 reflects the work time (in seconds) of the sequential algorithm and that of the parallel one subject to the number of variables. The number of MAM iterations I_{MAM} as well as the number of function evaluations F_{MAM} required for solving the problem are presented in Table 2. For comparison purposes, the number of function value computations F_{SQP} that would be required to solve the initial (non-approximated) problem by the SQP method is also given. In all conducted experiments, the objective function values in the sequential and parallel versions of the algorithm were the same (up to computational errors) and negligibly differed from the solution obtained by SQP. The reduction of the number of the function evaluations required to solve the problem using MAM as compared to the use of SQP was demonstrated visibly.

The computational experiments were carried out on a high-performance cluster at Lobachevsky State University of Nizhny Novgorod. A cluster node includes two Intel Sandy Bridge E5-2660 2.2 GHz CPUs and 64 Gb RAM. Each CPU has 8 cores, i.e. a total of 16 physical cores were available at the node. MS Visual Studio 15 and Intel Fortran Compiler were used to implement the algorithm.

5 Conclusions

Recent developments in the Multipoint Approximation Method (MAM) made it capable of solving large-scale industrial optimization problems. The fact that MAM solves the initial problem by using approximations of the objective function and constraints is the primary distinctive feature of the method. Within the framework of the present study, we developed a parallel version of MAM oriented to the reduction of the work time of the optimization algorithm (assuming that the computation of the values of the objective function and constraints has already been parallelized). The experiments performed have demonstrated an acceptable speedup when solving large-scale problems employing 16 cores on a single cluster node. The performance was demonstrated on a benchmark example of structural optimization known as the scalable cantilevered beam.

References

1. Toropov, V.: Simulation approach to structural optimization. Structural Optimization **1**, 37–46 (1989).
2. Toropov, V.: Multipoint approximation method in optimization problems with expensive function values. In: Sydow, A. (ed.) Proceedings of the 4th international symposium on Systems analysis and simulation, pp. 207–212, Elsevier, (1992).
3. Toropov, V., Filatov, A., Polynkin, A.: Multiparameter structural optimization using FEM and multipoint explicit approximations. Structural Optimization **6**, 7–14 (1993).
4. Shahpar, S., Polynkin, A., Toropov, V.: Large scale optimization of transonic axial compressor rotor blades. In: 49th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA, art. no. 2008-2056 (2008).
5. Polynkin, A., Toropov, V., Shahpar, S.: Design optimization of aircraft engine components. In: Proc. of 7th ASMO UK / ISSMO Conf. on Engineering Design Optimization. Process and Product Improvement (2008).
6. Polynkin, A., Toropov, V., Shahpar, S.: Multidisciplinary optimization of turbomachinery based on metamodel built by genetic programming. In: 13th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (2010).
7. van Keulen, F., Toropov, V., Markine, V.: Recent refinements in the multi-point approximation method in conjunction with adaptive mesh refinement. In: McCarthy, J.M. (ed.) Proc. of ASME Design Engineering Technical Conferences and Computers in Engineering Conf., 1–12, ASME, Irvine CA (1996).

8. Toropov, V., van Keulen, F., Markine, V., de Boer, H.: Refinements in the multi-point approximation method to reduce the effects of noisy responses. In: 6th AIAA/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization, pp. 941–951, AIAA (1996).
9. Toropov, V., Markine, V., Holden, C.: Use of mid-range approximations for optimization problems with functions of domain-dependent calculability. In: 3rd ISSMO/UBCAD/UB/AIAA World Congress of Structural and Multidisciplinary Optimization (1999).
10. Toropov, V., Markine, V.: The use of simplified numerical models as mid-range approximations. In: 6th AIAA/NASA/ISSMO Symp. on Multidisciplinary Analysis and Optimization, pp. 952–958 (1996).
11. Toropov, V., Alvarez, L.: Creation of multipoint approximations using genetic programming. In: Parmee, I.C. (ed.) Adaptive Computing in Design and Manufacture, 3rd Int. Conf., pp. 21–24, PEDC, Dartington (1998).
12. Polynkin, A. Toropov, V.: Mid-range metamodel assembly building based on linear regression for large scale optimization problems. Structural and Multidisciplinary Optimization **45**(4), 515–527 (2012).
13. Lancaster, P. Salkauskas, K.: Surfaces generated by moving least squares methods. Mathematics of Computation **87**, 141–158 (1981).
14. Liszka, T.: An interpolation method for an irregular net of nodes. Int. J. Num. Meth. **20**, 1599–1612 (1984).
15. Choi, K., Youn, B., Yang, R.-J.: Moving least squares method for reliability-based design optimization. In: 4th World Congress of Structural and Multidisciplinary Optimization, Dalian, China (2001).
16. Toropov, V., Schramm, U., Sahai, A., Jones, R., Zeguer, T.: Design optimization and stochastic analysis based on the moving least squares method. In: Herskovits, J., Mazorche, S., Canelas, A. (eds.) 6th World Congress of Structural and Multidisciplinary Optimization, art. no. 9412 (2005).
17. Polynkin, A. Toropov, V.: Recognition of design variable inter-dependencies using cross-validated moving least-squares method. In: Proceedings of the 51st AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, AIAA, art. no. 2010-2985 (2010).
18. Ollar, J., Toropov, V., Jones. R.: Mid-range approximations in sub-spaces for MDO problems with disparate discipline attributes. In: 15th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA, art. no. 2014-2437 (2014).
19. van Keulen, F. Toropov, V.: New developments in structural optimization using adaptive mesh refinement and multi-point approximations. Engineering Optimization **29**, 217–234 (1997).
20. Polynkin, A., Toropov, V., Shahpar, S.: Adaptive and parallel capabilities in the multipoint approximation method. In: 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, AIAA, art. no. 2008-5803 (2008).
21. Vanderplaats, G.: Multidiscipline Design Optimization. Vanderplaats Research & Development, Inc., Colorado Springs, (2001).