The use of Kernel PCA in Evolutionary Optimization for Computationally Demanding Engineering Applications

Dimitrios Kapsoulis, Konstantinos Tsiakas, Varvara Asouti, Kyriakos Giannakoglou
Parallel CFD & Optimization Unit
School of Mechanical Engineering
National Technical University of Athens
Athens, Greece
Email: {jim.kapsoulis, tsiakost}@gmail.com, vasouti@mail.ntua.gr, kgianna@central.ntua.gr

***Abstract*—Two techniques to further enhance the efficiency of Evolutionary Algorithms (EAs), even those which have already been accelerated by implementing surrogate evaluation models or metamodels to overcome a great amount of costly evaluations, are presented. Both rely upon the use of a Kernel Principal Component Analysis (Kernel PCA or KPCA) of the design space, as this reflects upon the offspring population in each generation. The PCA determines a feature space where the evolution operators should preferably be applied. In addition, in Metamodel–Assisted EA (MAEAs), the PCA can reduce the number of sensory units of metamodels. Due to the latter, the metamodels yield better approximations to the objective function value. This paper extends previous work by the authors which was based on Linear PCA, used for the same purposes. In the present paper, the superiority of using the Kernel (rather than the Linear) PCA, especially in real–world applications, is demonstrated. The proposed methods are assessed in single– and two–objective mathematical optimization problems and, finally, showcased in aerodynamic shape optimization problems with computationally expensive evaluation software.**

I. Introduction

During the last decades, EAs are widely used to solve Single– and Multi–objective optimization problems (SOO or MOO) with or without constraints, by accommodating any ready–to–use evaluation tool. Though EAs are able to find global optima, they require a great number of calls to the problem–specific model (PSM). If the PSM is a computationally demanding software, then the optimization turnaround time may become prohibitively large. Software pertinent to Computational Fluid Dynamics (CFD), Computational Structural Analysis, Computational Electromagnetics etc. are typical examples of PSMs used in real–world applications.

Various methods capable of reducing the computational cost of an EA–based optimization have been devised. In MAEAs [1], [2], [3], [4], [5], [6], [7], surrogate evaluation models (often referred to as "metamodels") can be used to replicate the outcome of the PSM at negligible CPU cost, after being trained on a set of already evaluated individuals. Nowadays, MAEAs have become a standard tool in evolutionary optimization. Response surface models (RSM), artificial neural networks (ANNs) [8] and Gaussian processes [3] are among the most widely used metamodels. In this work, without loss in generality, Radial Basis Function (RBF, [8]) networks, which are three–layer ANNs, are used as metamodels. With the exception of the few starting MAEA generations, all population members are approximately evaluated on local metamodels trained on the fly (on–line trained metamodels; to be explained in Section II) and only the most promising among them undergo re–evaluation on the PSM. The inexact pre–evaluation (IPE) of the population members in each generation is a key feature of our method, see [1], [3], [9]. This MAEA differs noticeably from the majority of other MAEAs which rely upon a single off–line metamodel trained during a pre–processing phase based on Design of Experiment strategies [10].

On the other hand, real–world optimization problems may have a great number of design variables. The so–called "curse of dimensionality" refers to both the performance degradation of EAs or MAEAs in problems with a great number of design variables and the fact that the prediction ability of metamodels becomes questionable with many input units, for a reasonably low number of training patterns.

Even without using metamodels (i.e. in standard EAs), the search process becomes quite faster, if the objective function f is separable. By definition, f is separable if there exist functions $f_1, f_2, \ldots, f_N$, such that: $f(x_1, x_2, \ldots, x_N) = f_1(x_1)f_2(x_2)\ldots f_N(x_N)$. In such a case, it suffices to minimize $f_1(x_1)$ in terms of x_1, $f_2(x_2)$ in terms of x_2 and so on. Since EAs perform better in separable problems, it would be beneficial to transform/map the design space to a new one, usually referred to as the feature space, where the problem becomes "as separable as possible". This mapping can be carried out through the PCA of an appropriate data set.

Moreover, artificially reducing the number of design variables the metamodel "sees" from N to $N-N_{DR}$, using a dimensionality reduction technique, may improve its prediction ability and even reduce the corresponding training cost. This will be the case if the $N–N_{DR}$ design variables the metamodel is trained on are those affecting mostly the objective function. So, in each generation of a MAEA, during the IPE phase, the PCA reduces the number of features the ANN sensory

units see. In other words, the metamodels are trained on patterns having less components than the design variables' array. Keeping only components having large variances, the IPE phase of the MAEA becomes much more effective.

The use of PCA for improving the performance of EAs and/or MAEAs, in large scale problems with a great number of design variables, has been proposed in [11]. The PCA assists the application of evolution operators and the training of metamodels. In the present work, the Kernel PCA is implemented instead of the Linear PCA used in [11]. It will convincingly be demonstrated that this boosts further the convergence of the EAs or MAEAs.

The first part of the results section is dealing with some widely used mathematical optimization problems. Due to their low CPU cost, these can be solved several times, for different random number generator (RNG) seeds and safe conclusions can be drawn. The benefit in these mathematical problems cannot be measured in CPU cost; it will necessarily be measured in terms of the number of evaluations so as the outcome of this study to correspond to the expected gain in expensive real–world problems. In the second part, the PCA variants of EA and MAEA are used to solve aerodynamic shape optimization problems.

II. The (μ, λ) EA & MAEA

Assume that EAs are used to solve the following optimization problem with M_o objective functions and M_c constraints

$$\begin{aligned} &min\vec{F}(\vec{x}) = min\{f_1(\vec{x}), ..., f_{M_o}(\vec{x})\} \\ &subject\ to\ c_j(\vec{x}) \leq 0,\ j=1, \ldots, M_c \end{aligned} \tag{1}$$

where $\vec{x} \in R^N$ is the optimization (or design) variables' array. For all design variables, their lower and upper bounds are known. For $M_o>1$, eq. 1 represents a MOO problem; and the EA computes the (Pareto) front of non–dominated solutions.

A brief description of the background (μ, λ) EA used in this paper follows. It handles three populations, namely the parent (S_μ^g, with μ members), the offspring (S_λ^g, with λ members) and the elite one (S_ϵ^g, with ϵ members at most), where g is the generation counter. S_μ^g results from the application of the parent selection operator to $(S_\lambda^{g-1} \cup S_\epsilon^{g-1})$. S_λ^g results from S_μ^g, via crossover and mutation, including elitism. In MOO problems, the Pareto front is approximated by the S_ϵ of the last generation. Each offspring is given a scalar cost by processing the f_i values of the current population based on dominance and sharing criteria such as those involved in the SPEA and NSGA techniques, [12], [13]. In constrained problems, individuals violating constraints are penalized with an exponential penalty function. In SOO, the scalar cost is nothing more than the objective function's value. All the already evaluated individuals, paired with the corresponding objective function and constraint values, enter a database (DB) which is dynamically updated during the evolution; duplicate entries are not allowed.

In the MAEA, ANNs or any regressor may serve as metamodels, undertaking the IPE of the newly formed offspring in each generation [1], [14]. The IPE of the S_λ^g members starts after a predefined minimum number (T_{DB}^{IPE}) of already evaluated individuals is archived in the DB; up to this point, a standard EA is used. This MAEA relies on on–line trained metamodels; these are trained separately for each new population member on its closest (in terms of Euclidean distances in the normalized design space) already evaluated individuals found in the DB, [9]. The selection of RBF centers and training patterns is important since it affects the prediction ability of the metamodel. At the end of the IPE phase, just a few ($\lambda_e \ll \lambda$) top population members, i.e. the most promising ones based on the metamodel, undergo re–evaluation on the PSM.

III. PCA-driven EAs and MAEAs

Principal Component Analysis (PCA) uses an orthogonal transformation to convert a set ($\mathbf{X}$) of observations $\vec{x} \in R^N$ of possibly correlated variables into a set of uncorrelated variables called principal components. In our case, the data set $\mathbf{X}$ must be formed by collecting M possible solutions to the problem with some common characteristics. The PCA does not need responses to be known (unsupervised learning). In the (μ, λ) EA or MAEA we are dealing with, $\mathbf{X}$ can be formed by either the offspring of the last generation (their common characteristic being that they gradually tend towards the optimal solution(s), at least after the first explorative generations) or the current elite set. The transformation based on the outcome of the PCA correlates each principal component with a variance of the observations; the first component is in the direction of the largest variance, the second is in the direction of the second largest variance and so forth.

Assume that the simplest possible variant of PCA (Linear PCA) is implemented on a set of observations. It is assumed that the data set $\mathbf{X}$ has been post–processed so as to have zero mean ($E[\mathbf{X}]=0$) and unit standard deviation ($E[\mathbf{X}^2]=1$) along all directions. Its covariance matrix is $P_{N\times N} = \frac{1}{M}\mathbf{X}\mathbf{X}^T$. This matrix contains the observations' correlation and is decomposed as $P_{N\times N} = U\Lambda U^T$ where Λ is a diagonal matrix with the eigenvalues of P and U is the matrix with the eigenvectors of P as rows. These eigenvectors are the principal components and the corresponding eigenvalues are their variances. Thus, a feature space based on the principal components becomes available.

Linear PCA performs well in problems with linearly correlated variables [11], [15]. However, in complex nonlinear problems, Linear PCA may not perform well and Kernel PCA (KPCA) [16] is, indeed, a viable alternative.

KPCA transforms the design space to the feature space through a mapping function, $\phi: R^N \rightarrow R^L$. L can be arbitrarily large. Also, as it will be shown below, there is no need to explicitly define the mapping function ϕ. Given a data set $\mathbf{X}$, the covariance matrix $P_{L\times L}$ (or just P hereafter) elements are

$$P_{\tau\sigma} = \frac{1}{M}\sum_{i=1}^{M} \phi_\tau(\vec{x}^i)\phi_\sigma(\vec{x}^i),\ \tau, \sigma = 1, \ldots, L \tag{2}$$

The corresponding eigenproblem is expressed by the following system of L equations

$$P\vec{v}^r = \Lambda^r \vec{v}^r,\ r=1,\ldots,L \tag{3}$$

where Λ is the $(L\times L)$ diagonal matrix with the corresponding eigenvalues. Depending on the value of L, the solution of eqs. 3 may become prohibitively expensive. The so–called kernel trick can help overcome this problem. Each eigenvector can be written in the form

$$\vec{v}^r = \sum_{\tau=1}^{M} a_\tau^r \vec{\phi}(\vec{x}^\tau),\ r=1,\ldots,L \tag{4}$$

or, after plugging eqs. 4 into 3,

$$\sum_{j=1}^{L} P_{ij} \sum_{\tau=1}^{M} a_\tau^r \phi_j(\vec{x}^\tau) = \Lambda^r \sum_{\tau=1}^{M} a_\tau^r \phi_i(\vec{x}^\tau),\quad i=1,\ldots,L \tag{5}$$

The inner product of eqs. 5 and $\vec{\phi}(\vec{x}^q),\ q=1,\ldots,M$, yields a system of M equations,

$$\sum_{i=1}^{L}\sum_{j=1}^{L}\sum_{\tau=1}^{M} a_\tau^r P_{ij}\phi_i(\vec{x}^q)\phi_j(\vec{x}^\tau) =$$
$$\Lambda^r \sum_{i=1}^{L}\sum_{\tau=1}^{M} a_\tau^r \phi_i(\vec{x}^\tau)\phi_i(\vec{x}^q),\ q=1,\ldots,M \tag{6}$$

Using eq. 2, eqs. 6 can be written as

$$\sum_{i=1}^{L}\sum_{j=1}^{L}\sum_{\tau=1}^{M}\sum_{z=1}^{M} a_\tau^r \phi_j(\vec{x}^z)\phi_i(\vec{x}^z)\phi_i(\vec{x}^q)\phi_j(\vec{x}^\tau) =$$
$$M\Lambda^r \sum_{i=1}^{L}\sum_{\tau=1}^{M} a_\tau^r \phi_i(\vec{x}^\tau)\phi_i(\vec{x}^q),\quad q=1,\ldots,M \tag{7}$$

The $(M\times M)$ kernel matrix K is defined as

$$K_{ij} = k(\vec{x}^i,\vec{x}^j) = \vec{\phi}(\vec{x}^i)\vec{\phi}^T(\vec{x}^j) = \sum_{p=1}^{L}\phi_p(\vec{x}^i)\phi_p(\vec{x}^j) \tag{8}$$

Among the most widely used kernels are the polynomial, the radial basis function (RBF) and the sigmoid ones [16], [17]. In this paper, the RBF kernel,

$$k(\vec{x}^i,\vec{x}^j) = exp\left(-\frac{||\vec{x}^i-\vec{x}^j||_2^2}{2\sigma^2}\right) \tag{9}$$

where σ is the width constant, is exclusively used; comparing different nonlinear kernels is beyond the scope of this paper.

Combining eqs. 7 and 8, the final system of equations to be solved (instead of eq. 3) for each eigenvalue is

$$K\vec{a}^q = M\Lambda^q\vec{a}^q,\quad q=1,\ldots,M \tag{10}$$

where $\vec{a}^q$ is the corresponding eigenvector. The use of Kernel PCA within an EA or MAEA includes the following steps:

1) Compute the kernel matrix (eq. 8) using the current offspring population.
2) Compute the eigenvalues (Λ) and eigenvectors (a) via the solution of eqs. 10.
3) Each new individual $\vec{x}$ is projected, into the feature space as follows

$$C_r(\vec{x}) = \vec{v}^r\cdot\vec{\phi}(\vec{x}) = \sum_{i=1}^{M} a_i^r k(\vec{x}^i,\vec{x}),\quad r=1,\ldots,M \tag{11}$$

In the present method, transforming the individuals back to the design space is absolutely necessary. In Linear PCA this is straightforward, which is not the case for the Kernel PCA though. The reason is that a vector $\vec{\phi}(\vec{x}')\in R^L$ might not have a unique representation in R^N. Nevertheless, a vector $\vec{z}\in R^N$ which approximately maps to $\vec{\phi}(\vec{x}')$ can be found. The design vector $\vec{x}'\in R^N$ maps to $\vec{C}(\vec{x}')\in R^M$ as follows

$$C_r(\vec{x}') = \sum_{i=1}^{M} a_i^r k(\vec{x}^i,\vec{x}'),\quad r=1,\ldots,M \tag{12}$$

This representation "compresses" the information of the feature space into M dimensions, while avoiding to compute the mapping function $\vec{\phi}$. A projection of $\vec{C}(\vec{x}')$ to the R^L space is defined as

$$\vec{p}(\vec{x}') = \sum_{r=1}^{M} C_r(\vec{x}')\vec{v}^r \tag{13}$$

The minimization of the Euclidean distances between the projection $\vec{p}(\vec{x}')$ and $\vec{\phi}(\vec{z})$ leads to the computation of

$$\vec{z} = argmin||\vec{p}(\vec{x}') - \vec{\phi}(\vec{z}))|| \tag{14}$$

If the RBF kernel, eq. 9 is used, problem (14) can be solved via the following fixed point iterative algorithm, [17],

$$\vec{z}^{new} = \frac{\sum_{i=1}^{M}\sum_{j=1}^{M} C_i(\vec{x}')a_j^i exp(-\frac{||\vec{x}^j-\vec{z}^{old}||_2^2}{2\sigma^2})\vec{x}^j}{\sum_{i=1}^{M}\sum_{j=1}^{M} C_i(\vec{x}')a_j^i exp(-\frac{||\vec{x}^j-\vec{z}^{old}||_2^2}{2\sigma^2})} \tag{15}$$

Having presented the PCA method, its integration in EA–based optimization follows. The PCA can be used during the evolution, to reduce the turnaround time of the optimization, by transforming the non–separable problem to an "almost separable" one. This transformation can also be used during the IPE phase, by training metamodels with less input units.

A. EA with PCA–driven evolution operators

In an EA assisted by the PCA, the role of the latter is to identify the principal component directions and map the design space to a new feature space, in which the problem becomes "more separable". For the first g_{PCA} generations, the PCA is not applied. In all subsequent generations, the current offspring population is considered to be the set of M training patterns. The offspring population is used due to its diversity (in the course of generations, it becomes populated by members which, or the majority of which at least, are close to the optimal solutions) and because it includes vital

information for the sought optimal solutions. Once the design space is mapped to the feature space via eq. 12, the evolution operators are applied. The principal components are computed using the current offspring population (prior to the application of the evolution operators), stored and used at the beginning and the end of the generation.

Regarding mutation, an increased mutation probability along the directions with small variances should preferably be used in order to re–enforce the exploration in this direction. The mutation probability for each principal component (superscript $i=1,\ldots,M$) is given by, [15],

$$p_{mut}^{i} = \alpha p_{mut} + (1-\alpha) M p_{mut} \frac{y_i}{\sum_{i=1}^{M} y_i} \tag{16}$$

where p_{mut} is a constant, user–defined, mutation probability, $\alpha \in [0,1]$ (preferably kept below 0.5 to lay emphasis on the second term in eq. 16) and y_i is computed as

$$y_i = \frac{V_{max} - V_i}{V_{max} - V_{min}} \tag{17}$$

where V_i are the variances of the current offspring population, after being mapped to the feature space and $V_{max} = \max\{V_1, ..., V_M\}$, $V_{min} = \min\{V_1, ..., V_M\}$.

After the application of the crossover and mutation operators, the population is transformed back to the design space using eq. 15. An EA or MAEA in which the evolution operators are assisted by the Linear PCA will be referred to as an EA(L) or MAEA(L). In contrast, EA(K) or MAEA(K) denote the use of Kernel PCA, for the same purpose.

B. EAs with PCA–Truncated Metamodels

The computational cost of the metamodel training and its prediction accuracy are affected by the problem size. Since λ metamodels should be trained per generation, the training cost should be negligible compared to the cost of a single evaluation on the PSM. Here, dimensionality reduction means to train the metamodels with patterns with less components than the number of design variables. This noticeably improves the prediction accuracy of metamodels.

As mentioned before, the eigenvectors computed by the PCA are associated with the corresponding eigenvalues, which represent the variances. The variances indicate the directions of the feature space along which the members of the current generation are less or more scattered (a high variance indicates that the available data are adequately scattered in this direction). The directions of the feature space with low variances are ignored, since all members are so clustered around the same value in this direction, thus the corresponding components of the training patterns are not expected to affect the prediction ability of the metamodel. The number (N_{DR}) of design variables to be cut–off is user–defined. Thus, the metamodel is trained on data of lower dimension.

This truncation applies only during the IPE phase of MAEAs and after T_{DB}^{PCA} individuals have been archived in the DB. In particular, for each population member, a "personalized" metamodel is built by selecting the necessary training patterns as described in [9]. The training patterns are mapped to the feature space and their principal components associated with the lower eigenvalues (smaller variances) are truncated. A MAEA in which the PCA is employed during the training of the metamodels will be referred to as M(L)AEA or M(K)AEA depending on whether the Linear or Kernel PCA is used. It is evident that M(K)AEA(K) denotes a MAEA with a dual use of Kernel PCA.

IV. Applications To Mathematical Cases

A first demonstration of the proposed methods in four mathematical problems, each of which with different characteristics follows. The optimization platform EASY [18], developed by the group of authors, was used to implement the PCA-based variants of the (μ, λ) EA and MAEA.

Three of them are SOO problems while the last one has two objectives. In all cases, the number of optimization variables $\vec{x} = (x_1, \ldots, x_N)$ is $N = 50$ and $\vec{o} \in R^N$ is the optimal solution (known beforehand, in all of them). Let $\vec{z} = \vec{x} - \vec{o}$. The values of $\vec{o}$ and the $\vec{x}$ bounds are differently specified in each of the following cases.

1) **Ellipsoid Problem** :

$$min\ f(\vec{x}) = \sum_{i=1}^{N} z_i^2 \tag{18}$$

where $-100 \leqslant x_i \leqslant 100$ and $o_i = i/N^2$. This function is unimodal and separable.

2) **Weierstrass Problem** :

$$min\ f(\vec{x}) = \sum_{i=1}^{N} \Big(\sum_{k=0}^{20} [a^k cos(2\pi b^k (z_i + 0.5))] \Big) - N \sum_{k=0}^{20} [a^k cos(\pi b^k)] \tag{19}$$

where $a = 0.5,\ b = 3, -0.5 \leqslant x_i \leqslant 0.5$ and $o_i = i/N$. This function has several local optima with a single global optimum though. It is continuous over the whole search space but differentiable only over a subset of it. Moreover, it is non–separable, which makes it a nice test–bed for assessing the PCA variants of EA.

3) **Rosenbrock Problem** :

$$min\ f(\vec{x}) = \sum_{i=1}^{N-1} (100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) \tag{20}$$

where $-0.5 \leqslant x_i \leqslant 0.5$ and $o_i = i/N$. This function is non–separable, having a very narrow valley (in the search space) separating local and global optima.

4) **Two – Objective Problem** :

$$min\ f_1(\vec{x}) = \sum_{i=1}^{N-1}(100(z_i^2 - z_{i+1})^2 + (z_i - 1)^2) \quad (21)$$

$$min\ f_2(\vec{x}) = \sum_{i=1}^{N}(\sum_{k=0}^{20}[a^k cos(2\pi b^k(w_i + 0.5))]) - N\sum_{k=0}^{20}[a^k cos(2\pi b^k \cdot 0.5)] \quad (22)$$

where $a = 0.5$, $b = 3$, $-0.5 \leqslant x_i \leqslant 0.5$, $\vec{z} = \vec{x} - \vec{o^1}$, $\vec{w} = \vec{x} - \vec{o^2}$ and $o_i^1 = i/N$, $o_i^2 = i/N^2$.

In general, each mathematical case was optimized with all variants of EA, namely EA, EA(L), EA(K), MAEA, MAEA(K), M(K)AEA(K). Each optimization was repeated 20 times with different RNG seeds, so as to reduce their effect upon the optimization procedure. The convergence plots present the averaged performance of the evolution in the course of evaluations on the PSM of the 20 runs. For each run, a stopping criterion of 50000 evaluations was imposed. A (10, 20) EA or MAEA was used in all cases. In the MAEA variants, $T_{DB}^{IPE} = 100$ and $\lambda_e = 3$. In the runs assisted by the PCA, $g_{PCA} = 2$ and, when metamodels are used, $T_{DB}^{PCA} = 100$, $N_{DR} = 40$.

Comparison of the convergence histories of all variants are presented in figs. 1 to 4. For the two–objective problem, the hypervolume indicator measures the quality of the fronts of non–dominated solutions. This practically quantifies the area dominated by the front of non–dominated individuals up to a user–defined nadir point; higher values correspond to better fronts, i.e. those being closer to the Pareto front.

All PCA variants lead to faster convergence compared to the corresponding EA and MAEA (the EA is compared to EA(L), (K) and the MAEA to MAEA(K) and M(K)AEA(K)). The Kernel PCA constantly outperforms the Linear one.

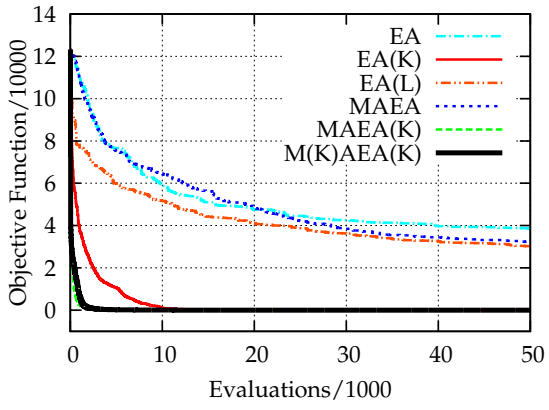

Fig. 1. Ellipsoid Problem: Average (of 20 runs) performance of the EA and MAEA variants.

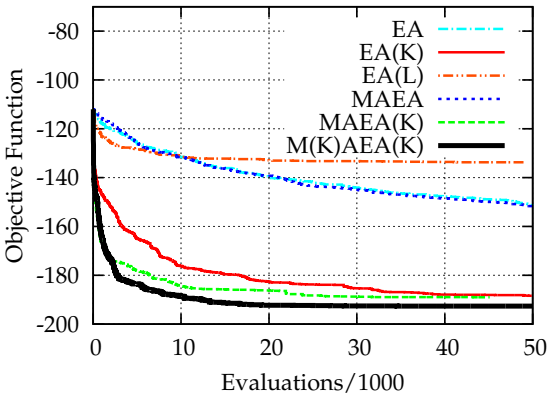

Fig. 2. Weierstrass Problem: Average (of 20 runs) performance of the EA and MAEA variants.

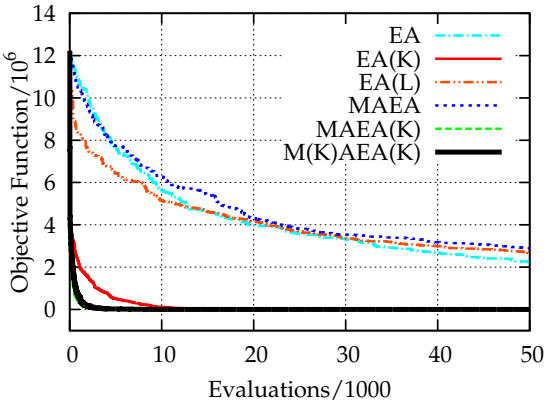

Fig. 3. Rosenbrock Problem: Average (of 20 runs) performance of the EA and MAEA variants.

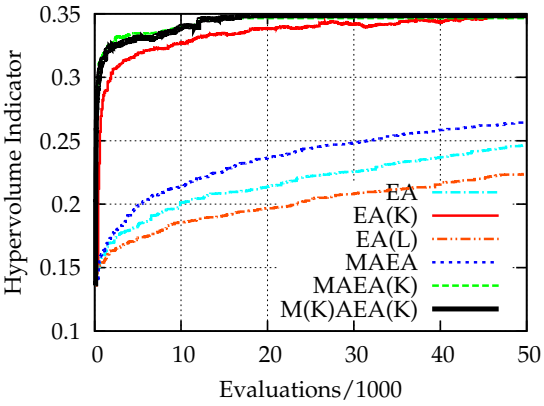

Fig. 4. Two–objective Problem: Average (of 20 runs) performance of the EA and MAEA variants.

V. Aerodynamic Shape Optimization Problems

In this section, the aforementioned variants of EA and MAEA are used to optimize the shapes of an isolated airfoil and a car, based on aerodynamic performance criteria.

A. Isolated Airfoil Shape Optimization

The first real–world optimization case deals with a two–objective problem in which the shape of an airfoil is optimized for minimum drag (C_D) and maximum lift (C_L) coefficients. The flow conditions are: freestream Mach number $M_\infty = 0.19$, flow angle $a_\infty = 5^o$ and Reynolds number based on the chord $Re_c = 1.3 \cdot 10^7$.

The airfoil shape and the mesh deformations are controlled by a morphing technique based on harmonic coordinates. This is a generalization of barycentric coordinates and was initially proposed in [19] for character articulation. A topologically flexible structure called "cage" is used to control deformations. Herein, this concept is extended to CFD applications where, apart from performing the shape morphing, it is important to simultaneously deform/displace the CFD mesh accordingly. To avoid mesh quality degradation due to huge distortions at the boundaries of the cage, a two–cage structure is adopted instead (fig. 5). Each cage is filled with a coarse unstructured mesh, which is of course much coarser than the CFD mesh. The harmonic coordinates of the vertices of the inner cage are computed on the coarse mesh by solving as many Laplace equations as the number of the inner cage boundary vertices, with appropriate boundary conditions. The so–computed harmonic coordinates are, then, interpolated from the cage coarse mesh to the CFD mesh nodes. A harmonic deformation of the CFD mesh can, then, be explicitly defined by the cage boundary vertices displacements. The inner cage controls the shape deformation and mesh morphing while the outer cage limits the effect of morphing; in fact, the outer cage vertices remain still and the Laplace equations don't need to be solved for them.

In this case, 16 vertices were used to define the inner cage (fig. 5). Two of them coincide with the leading and trailing edge of the airfoil and remain still so as to keep the airfoil chord constant. The remaining vertices were allowed to vary along both directions, leading to 28 design variables in total. The coarse cage mesh is formed by triangles generated by the advancing front method. The CFD evaluation is carried out on the GPU–enabled in–house Navier–Stokes equations solver for compressible fluid flows [20], [21]. The CFD mesh is fully unstructured with approximately $64 \cdot 10^3$ nodes. An averaged cost per evaluation of a single individual on a single NVIDIA K20 GPU is about 3min. This wall clock time includes the cost of mesh deformation. The optimization was carried out using a $(20, 40)$ EA, EA(K), MAEA, MAEA(K) and M(K)AEA(K). In the MAEA variants, $T_{DB}^{IPE} = 80$ and $\lambda_e = 2$. In the runs employing the PCA, $g_{PCA} = 2$ and, if metamodels are used, $T_{DB}^{PCA} = 80$, $N_{DR} = 18$. A stopping criterion of 500 evaluations on the CFD model was imposed.

A comparison of the convergence histories of the EA's variants is presented in fig. 6. The integration of KPCA in EAs can compute a better front of non–dominated solutions with the same computational budget. The "optimal" front of non–dominated solutions from all the optimizations (computed by M(K)AEA(K)) is presented in fig. 7 along with three indicative airfoil shapes.

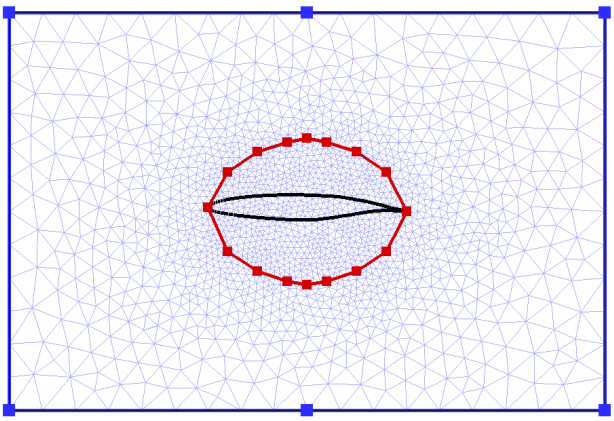

Fig. 5. Shape Optimization of an Isolated Airfoil: The two–cage structure used for shape parameterization and mesh deformation. The coordinates of the vectices of the inner cage (red), excluding those of the leading and trailing edge, stand for the design variables of the problem. The outer cage (blue) is not allowed to change. The (much finer) CFD mesh is not shown.

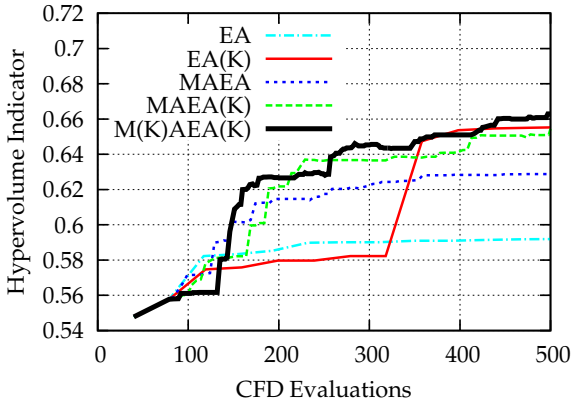

Fig. 6. Shape Optimization of an Isolated Airfoil: Comparison of the convergence history of the optimizations performed in terms of the number of CFD evaluations which is proportional to the CPU cost.

B. Shape Optimization of the DrivAer Car

The last case is concerned with the optimization of the shape of the fastback configuration of the DrivAer car model. Air flows in the axial direction with 11 m/s, while the road and the car's wheels remain static. The aim is to redesign the car for minimum drag coefficientz (C_D). The car shape parameterization was based on the volumetric NURBS method, [21], which additionally undertakes the mesh deformation according to the changing geometry of the rear part of the car. In particular, a $7 \times 7 \times 7$ NURBS control box was used to parameterize the volume around the car's boat tail and rear underbody, fig. 9. The three internal rows of control points in each direction were allowed to vary along all three Cartesian directions.

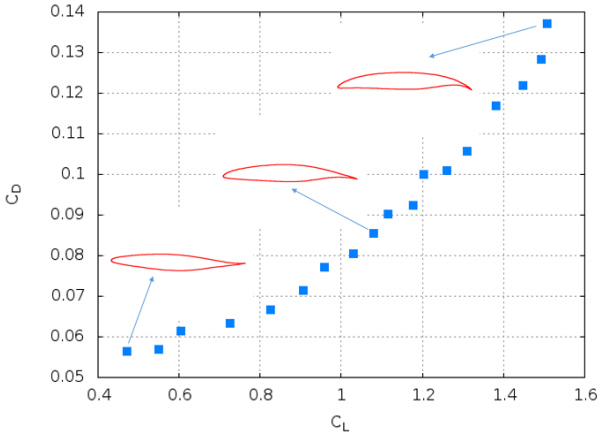

Fig. 7. Shape Optimization of an Isolated Airfoil: The front of non–dominated solutions computed by the M(K)AEA(K).

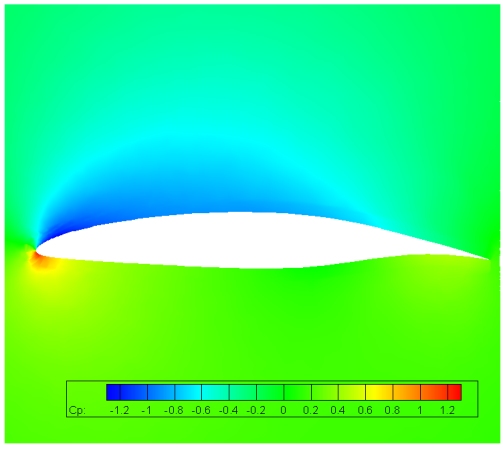

Fig. 8. Shape Optimization of an Isolated Airfoil: Iso–pressure coefficient areas around the airfoil with $C_l = 1.082$, $C_d = 0.0845$ from the front of non–dominated solution of fig. 7.

All other control points were kept fixed, in order to ensure smooth transition between deformable and non–deformable areas, leading to 81 design variables in total. Due to symmetry, only half of the car is modeled. Each candidate solution was evaluated with the incompressible fluid flow variant of the same GPU–enabled Navier–Stokes solver [20]; a steady flow solver was used; depending on the geometry of the rear part of the car, mild flow unsteadiness might appear, in which case the time–averaged drag of the last iterations is used as objective function. The Spalart–Allmaras [22] turbulence model is used. The computational mesh consists of approximately $1.4 \cdot 10^6$ nodes and each evaluation takes about 40min on an NVIDIA K20 GPU, including the morphing process.

The optimization was carried out using a $(10, 20)$ EA, EA(K), MAEA, MAEA(K) and M(K)AEA(K). The stopping criterion was 200 evaluations on the CFD model. In the MAEA variants, $T_{DB}^{IPE} = 20$ and $\lambda_e = 3$. In the runs supported by the PCA, $g_{PCA} = 2$ and, when metamodels are used, $T_{DB}^{PCA} = 20$, $N_{DR} = 71$.

A comparison of the convergence histories of the above

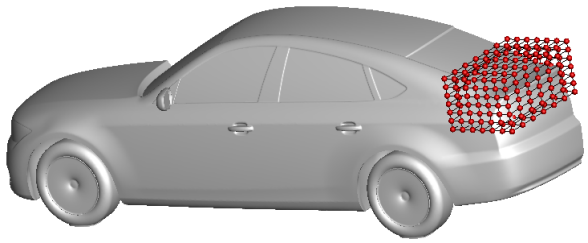

Fig. 9. Shape Optimization of the fastback DrivAer Car: NURBS morphing box. Only the rear part of the car which is encapsulated into the morphing box is allowed to change during the optimization loop.

EA variants can be found in fig. 10. It is clear that, using the Kernel PCA to assist both metamodels and evolution operators, enables the optimization algorithm to perform better, that is to find better solutions with the same computational budget.

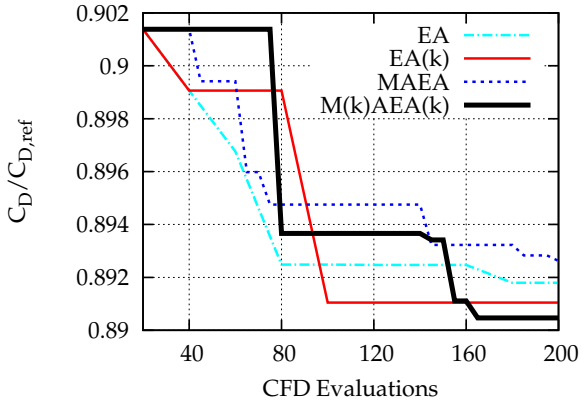

Fig. 10. Shape Optimization of the fastback DrivAer Car: Comparison of the convergence history of the optimizations performed in terms of the number of CFD evaluations which is practically proportional to the CPU cost.

A comparison between the pressure fields on the optimal and baseline shapes of the car is shown in fig. 11.

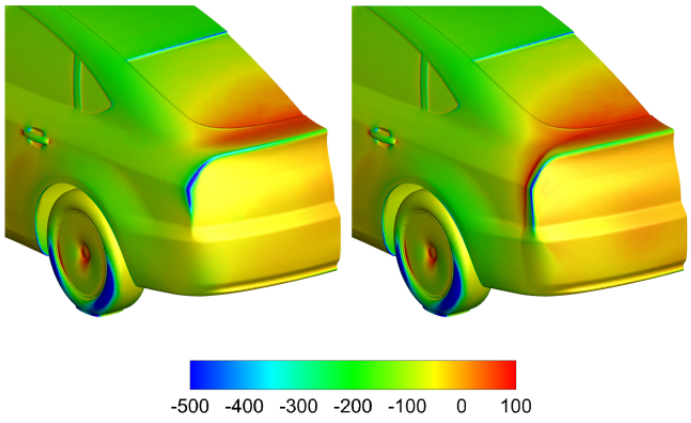

Fig. 11. Shape Optimization of the fastback DrivAer Car: Comparison of the pressure field on the optimal (right) and the baseline (left) shape of the car. The rear part of the car exhibits higher pressure and this reduces the drag.

VI. Conclusion

Two methods capable of improving the efficiency of EAs and MAEAs, when applied to high-dimensional (engineering) optimization problems, were demonstrated. They rely on the KPCA which is performed anew in each generation using the current offspring population. The KPCA can (a) assist the evolution operators by mapping the populations to a feature space, in which the problem appears to be "more separable" and (b) reduce the number of design variables with which the metamodels are trained. The Kernel PCA has the ability to process the design variables more efficiently than the Linear PCA, thus, the evolution operators perform better in the KPCA's feature space. This can lead to a better performing EA–based optimization, as shown in the presented problems. In MAEAs, the number of design variables used to train the metamodels can be significantly reduced, leading to more accurate predictions. The latter can be seen by comparing the metamodel's relative prediction error between MAEA and M(K)AEA; see for instance fig. 12 for the Weierstrass problem of section IV. The EA variants supported by the KPCA outperform not only the conventional EAs but those assisted by the Linear PCA too, as it becomes clear from the results of the mathematical problems. In real–world applications, such as the optimization of aerodynamic shapes, usually with a great number of design variables and costly evaluations, the EA or MAEA which make use of the KPCA can be proved absolutely necessary for these problems to be solved at affordable computational cost. Note that, in this kind of problems which involve expensive CFD software, the computational cost for additionally implementing the KPCA is negligible compared to the cost of a single evaluation.

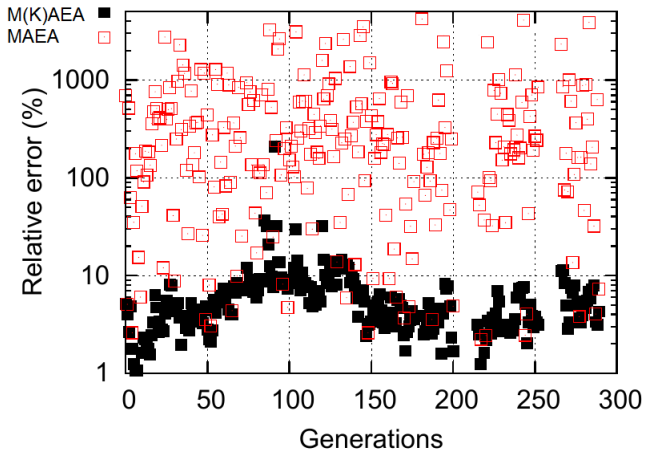

Fig. 12. Comparison of the relative prediction error $(f-\hat{f})/f$ (averaged over the entire offspring population in each generation; $\hat{f}$ is the approximation to the objective function f, based on the metamodel) of the metamodels with and without the use of the KPCA. Computed by off–line processing the MAEA and M(K)AEA results of the Weierstrass problem.

References

[1] M.K. Karakasis, A.P. Giotis and K.C. Giannakoglou. Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. *International Journal for Numerical Methods in Fluids*, 43(10-11):1149–1166, 2003.

[2] Y. Jin, M. Olhofer and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.

[3] M. Emmerich, K.C. Giannakoglou and B. Naujoks. Single- and multi-objective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[4] W. Shyy, N. Papila, R. Vaidyanathan and K. Tucker. Global design optimization for aerodynamics and rocket propulsion component. *Progress in Aerospace Sciences*, 37:59-118, 2001.

[5] S. Jiang, Z. Cai, J. Zhang and Y.S. Ong. Multiobjective Optimization by Decomposition with Pareto-adaptive Weight Vectors. *Seventh International Conference on Natural Computation*, Shanghai, China, July 26-28, 2011.

[6] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 2:61-70, June 2011.

[7] C. K. Goh, D. Lim, L. Ma, Y. S. Ong and P. S. Dutta. A Surrogate-Assisted Memetic Co-evolutionary Algorithm for Expensive Constrained Optimization Problems. *IEEE Congress of Evolutionary Computation*, New Orleans, Los Angeles, 2011.

[8] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, New Jersey, USA, 2nd edition, 1999.

[9] M.K. Karakasis and K.C. Giannakoglou. On the use of metamodel-assisted, multi-objective evolutionary algorithms. *Engineering Optimization*, 38(8):941–957, 2006.

[10] R.H. Myers and D.C. Montgomery. Response Surface Methodology: Process and Product Optimization Using Designed Experiments. 2nd Ed. New York, 2002.

[11] S.A. Kyriacou, S. Weissenberger and K. C. Giannakoglou. Design of a Matrix Hydraulic Turbine using a Metamodel-Assisted EvolutionaryAlgorithm with PCA-DrivenEvolution Operators. *International Journal of Mathematical Modelling and Numerical Optimization* 3 (2): 45-63.

[12] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan. A fast and elitist multiobjective genetic algorithm NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182-197, 2002.

[13] E. Zitzler, M. Laumans and L. Thiele. SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization. *EUROGEN 2001*. Athens, Greece, September 19-21, 2001.

[14] J. Branke and C. Schmidt. Faster convergence by means of fitness estimation. *Soft Computing*, 9(1):13–20, 2005.

[15] S.A. Kyriacou and V.G. Asouti and K.C. Giannakoglou. Efficient PCA-driven EAs and metamodel-assisted EAs, with applications in turbomachinery. *Engineering Optimization* 46 (7): 895-911, 2013.

[16] B. Schölkopf, A. Smola and K. Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Technical Report*, Max-Planck-Institute für biologische Kybernetik, 1996.

[17] Q. Wang. Kernel Principal Component Analysis and its Applications in Face Recognition and Active Shape Models. *CoRR*, 1207(3538), 2012.

[18] The EASY (Evolutionary Algorithms SYstem) software, http://velos0.ltt.mech.ntua.gr/EASY.

[19] P. Joshi, M. Meyer, T. DeRose, B. Green and T. Sanocki. Harmonic Coordinates for Character Articulation. 34^{th} *International Conference and Exhibition on Computer Graphics and Interactive Techniques*, San Diego, California USA, August 7-9, 2007.

[20] I.C. Kampolis, X.S. Trompoukis, V.G. Asouti and K.C. Giannakoglou CFD-based analysis and two-level aerodynamic optimization on graphics processing units. *Computational Methods Applied Mechanical Engineering*, 199(9-12):712-722, 2010.

[21] K.T. Tsiakas, X.S. Trompoukis, V.G. Asouti and K.C. Giannakoglou. Shape Optimization of Wind Turbine Blades using the Continuous Adjoint Method and Volumetric NURBS on a GPU Cluster. *EUROGEN 2015*, Glasgow, UK, September 14-16, 2015.

[22] P. Spalart and S. Allmaras. A one-equation turbulence model for aerodynamic flows. *AIAA Paper* 92-0439, 1992.