Proc. of the Second Intl. Conf. on Advances in Mechanical and Automation Engineering - MAE 2015

ISBN: 978-1-63248-045-3 doi: 10.15224/ 978-1-63248-045-3-43

A Framework for simulation-driven engineering design

[Yoel TENNE]

Abstract*— The modern engineering design process often employs computer simulations to evaluate candidate designs. This setup transforms the design process into an optimization problem which involves a computationally-expensive *black-box function*, namely, which lacks an analytic expression and where each function evaluation requires large computational resources. An additional challenge in such settings is that simulation runs may consistently fail for some specific candidate designs, but with the reason for failure being unknown. To effectively handle such challenging problems this paper proposes an engineering optimization framework which incorporates a classifier whose goal is to predict if a candidate design is likely to result in a failed simulation run. This prediction is then used to dynamically divert the optimization search away from such designs, without sending them to the simulation. Numerical experiments based on an airfoil optimization problem show the effectiveness of the proposed approach. *(Abstract)

Keywords*— *engineering design optimization, computer simulations, metamodels, classifiers (key words)

I. Introduction

The modern engineering design process often uses computer simulations to evaluate candidate designs. This is done to enhance the design process, for example, to reduce its duration or cost. Such simulations, which must be properly validated with laboratory experiments, effectively act as an objective function which assigns a merit value to a candidate design. To this end, this transforms the design process into an optimization problem where candidate designs are represented as vectors of design variables, and the goal is to find a design with the best objective value [1]. Therefore in such settings a candidate design and a vector of design variables are equivalent, and are used interchangeably in this paper text. While computer simulations have numerous merits, they also introduce several challenges into the design process: i) each simulation run is *computationally-expensive*, namely, it requires large computational resources and hence only a small number of simulation runs can be made, ii) there is no analytic expression which defines how candidate designs are mapped to their merit value, which effectively renders the simulation as a *black-box function* , and iii) the latter black-box function can have a complicated landscape with multiple local optima, an aspect which further exacerbates the optimization difficulty.

In these settings classical optimization techniques may perform poorly, and this has motivated the development of new and dedicated approaches for such simulation-driven engineering problems [1].

Yoel Tenne*(Author)*
Ariel University
Israel

An established framework is that of using *metamodels*, also termed in the literature as *surrogates* or *response surfaces*, whose goal is to approximate the computationally expensive function (namely, the simulation) while requiring neligible computational resources per evaluation. This way, approximate objective values can be obtained economically, and more designs can be examined during the process.

However, computer simulations introduce another challenge, namely some simulation runs can consistently fail for some specific candidate designs, but with the cause of failure being unknown [3]. In this paper such designs are termed as *simulation-infeasible*, while those for which the simulation completes successfully are termed as *simulation-feasible*. The difficulty arising is that simulation-infeasible designs can consume a significant portion of the available optimization budget but without assisting the search to progress, thereby leading to search stagnation and to a poor final result.

In these settings classical design optimization frameworks can struggle, and therefore this paper proposes a framework which leverages on two specialized features: a) it uses multiple types of metamodels concurrently to benefit from varying approximations, and aggregates their individual predictions into a single one, and b) it incorporates into the optimization process a classifier, adpoted from the domain of computational intelligence, to predict if a candidate design is expected to be simulation-infeasible. If so, the search is diverted towards designs predicted to be simulation-feasible, without sending the design to the simulation. Numerical experiments based on an engineering problem of airfoil shape optimization and benchmarkings against two competing frameworks from the literature show the effectiveness of the proposed framework. The remainder of this paper is organized as follows: Section 2 provides the general background and literature survey, Section 3 describes the proposed framework, and Section 4 presents the numerical experiments and their analysis. Lastly, section 5 concludes this paper.

II. Background

As mentioned in Section 1, two main components of the proposed framework are the metamodels and the classifier. Numerous types of metamodels have been proposed, for example radial basis functions (RBF) and Kriging from geostatistics, artificial neural networks from the domain of computational intelligence, and polynomial approximations from applied mathematics [1]. However the optimal type of metamodel is problem-dependant, and it is often unknown prior to the optimization process which type is the most suitable. One approach to address this is by using multiple types of metamodels concurrently, either by selecting the most suitable metamodel dynamically during the search [2], or by combining their predictions into a single one, a scenario which is often termed in the literature as an *ensemble* [12].

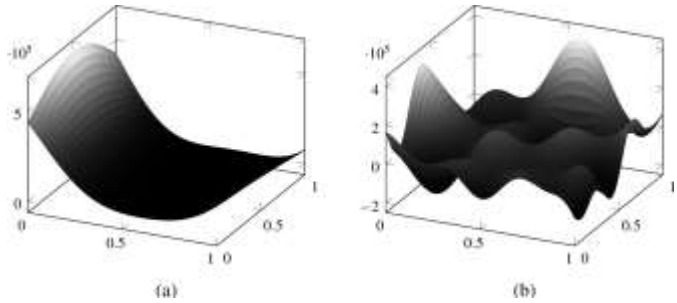
Proc. of the Second Intl. Conf. on Advances in Mechanical and Automation Engineering - MAE 2015

ISBN: 978-1-63248-045-3 doi: 10.15224/ 978-1-63248-045-3-43

Another challenge is that the computer simulation may consistently fail for some candidate designs. Such simulation-infeasible designs have been reported in the literature, for example in references [2,4,5,6,14]. Given their undesirable impact on the search, several techniques have been proposed to handle them. In reference [5], the authors explored the use of a classifier to discard simulation-infeasible vectors, but did not consider the use of metamodels and the impact of their approach on the latter. Reference [4] proposed an approach which severely penalized simulation infeasible vectors and then incorporated them into the metamodel. Alternatively, in reference [6] the authors proposed to completely discard simulation infeasible vectors from the training set of the metamodel. However, within the domain of metamodel-assisted optimization such approaches suffer from several shortcomings: a) assigning simulation infeasible vectors a penalized objective value and then incorporating them using them to train a metamodel can severely deform the landscape of the resultant metamodel, while b) excluding such vectors altogether discards information which may could be used to improve the optimization process. As an example, Figure 1 compares two Kriging metamodels of the Rosenbrock function: (a) shows a metamodel trained by using 30 regular vectors, while (b) shows the resultant metamodel when the latter sample was augmented with 20 vectors which were assigned a penalized objective value, taken as the worst objective value from the baseline sample. The resultant metamodel landscape has been severely deformed and contains many false optima, which further complicates the task of locating an optimum of the true objective function.

Figure 1: The impact of adding penalized vectors to the metamodel

Such issues have motivated the development of alternative approaches to handle simulation-infeasible



vectors. For example, in reference [3] the authors proposed a dual metamodel approach, where one interpolated the objective function and the other interpolated the penalty value between simulation-feasible and -infeasible vectors. Other studies have explored the use of classifiers for constrained non-linear programming, but not focusing on simulation-infeasible vectors [7]. Further exploring the use of classifiers, reference [3] presented preliminary results with a baseline algorithm involving a single metamodel.

III. Proposed framework

To address the shortcoming discussed in Sections 1 and 2, this paper proposes an enhanced framework. The proposed framework begins by generating an initial sample of vectors with the Latin Hypercube sampling method [3]. The latter is used as it provides a space-filling sample which improves the accuracy of the resultant metamodels. The vectors generated are then evaluated with the computer simulation to obtain their corresponding objective value.

In the next step, the framework trains a set of metamodels,

$$m_i(x), i = 1...N_m \quad (1)$$

where the number of metamodels (N_m) and their types are a-priori prescribed by the user. It is emphasized that there is no restriction on the type or number metamodels which can be prescribed. An example of such prescribed settings by the user would be the set Kriging, RBF, and a neural network, which defines $N_m = 3$.

$$w_i = \frac{(e_i)^{-1}}{\sum (e_i)^{-1}} \quad (2)$$

Each metamodel is trained by using all the simulator-feasible vectors which have been evaluated. To benefit from the modelling capabilities of different metamodels types, these individual predictions are aggregated into a single one, termed in the literature as an *ensemble prediction* [12]. This is achieved by assigning each metamodel an *ensemble weight*

where e_i is the root mean square prediction error of the *i*th metamodel. It serves as an indicator for the metamodel accuracy, and is obtained by taking all the feasible vectors evaluated, splitting them into a *training set* and *testing set*, training a metamodel based on the former and testing its prediction on the latter. This way, the above formulation gives a larger influence to metamodels which appear to be more accurate and vice-versa. Once all ensemble weights have been calculated, the ensemble prediction is given by

$$\hat{m}(x) = \sum w_i \cdot m_i(x) \quad (3)$$

Following this step, the proposed framework trains a classifier $c(x)$, whose goal is to predict if a candidate vector is simulator-feasible or not. Mathematically, given a set of vectors which have been assigned to n_s distinct sets $S_j, j = 1...n_s$, a classifier maps a new vector into one of these sets, namely:

$$c(x): R^n \rightarrow S_j \quad (4)$$

In this paper the well-established nearest-neighbour classifier was used [1], which maps the vector x to the set to which its distance is the smallest:

$$c(x): x \in R^n \rightarrow S_k, d(x, S_k) = min\_d(x, S_j) \quad (5)$$

Proc. of the Second Intl. Conf. on Advances in Mechanical and Automation Engineering - MAE 2015

ISBN: 978-1-63248-045-3 doi: 10.15224/ 978-1-63248-045-3-43

After training the classifier the framework performs an optimization search where it seeks the optimum of the ensemble prediction $\hat{m}(x)$. However, the search is performed by using the modified objective function

$$\overline{m}(x) = \begin{cases} \hat{m}(x) if\ \_c(x)\_is\_ \text{classified as SF} \\ p \text{ otherwise} \end{cases} (6)$$

where *p* is a penalized objective value, taken to be the mean of the objective values in the initial sample. This way the optimization algorithm receives the ensemble prediction if the candidate vector *x* is classified as simulation-feasible. Otherwise, the optimizer receives the penalized value, which in turn diverts the search away from the infeasible vector and avoids sending it for evaluation with the simulation. The optimizer used is the real-coded evolutionary algorithm of Chipperfield [8], as it can detect good solutions even for challenging nonconvex functions.

Since the prediction of the individual metamodels, and hence of the ensemble, deviates from the true expensive black-box function it is necessary to safeguard the optimization search to ensure convergence to an optimum of the true objective function, and not to a false optimum artificially created by the metamodels. To achieve this, the optimization search is performed within a *trust-region* approach, as it ensures an asymptotic convergence to a true optimum [13]. Accordingly, the EA searches for an optimum of the ensemble in the trust-region, which is the region centred at the current best solution (x_b) and of radius *r*, where *r* is dynamically updated during the search as explained below.

After the EA search completes it returns a vector $x*$, namely the best solution found in the trust-region, and the latter is then evaluated with true expensive function. Based on the observed function value the following trust-region updates occur (assuming a minimization problem):

if $f(x^*) < f(x_b)$: The optimization step succeeded , which implies that the ensemble prediction is sufficiently accurate. The trust-region radius is doubled, and is centred around the new best solution.

if $f(x^*) \geq f(x_b)$ and the number of vectors in the trust-region is deemed as too small: the optimization step failed, but this may be due to the ensemble prediction being inaccurate. In this case, another vector is sampled in the trust region to improve the prediction accuracy.

if $f(x^*) \geq f(x_b)$ and the number of vectors in the trust-region is deemed as sufficient: the optimization step failed, and the ensemble is considered to be sufficiently accurate. Therefore, the trust-region radius is halved.

After these steps the entire process repeats, until the prescribed limit of simulation runs is reached. To conclude this section, Algorithm 1 gives the workflow of the proposed framework.

It is emphasized that while in this study the proposed framework used three types of metamodels and one type of classifier, any other number or variants of the latter two can be used.

generate an initial sample of vectors and evaluate them with the true objective function;
repeat
    train metamodels with simulation-feasible vectors;
    calculate the ensemble weights;
    train a classifier using all the vectors;
    perform a trust-region step;
    update the trust region based on the observed value;
until limit of simulation runs is reached;

Algorithm 1: The workflow of the proposed framework

IV. Performance analysis

To evaluate its effectiveness, the proposed framework was applied to an engineering problem of airfoil shape optimization. The problem is a suitable test case as it is both representative of real-world expensive black-box optimization problems and it contains simulation-infeasible vectors, as explained below. In this problem the goal is to find an airfoil shape which maximizes the ratio of lift to drag at a specified altitude and speed. Also, to ensure structural integrity, the minimum airfoil thickness (t) between 0.2 to 0.8 of the airfoil chord must be equal to or larger than a critical value $t^* = 0.1$. Accordingly, the objective function used was

$$f = -\frac{c_l}{c_d} + p_t (7),$$

where c_l is the lift coefficient, c_d is the drag coefficient, and p_t is a penalty for violating the minimum thickness constraint, and was defined as

$$p_t = \begin{cases} \frac{t^*}{t} \cdot \left|\frac{c_l}{c_d}\right| \text{if } t < t^* \\ 0\_\text{otherwise} \end{cases} (8)$$

Candidate airfoils were represented with the Hicks-Henne parameterization [9] which uses a baseline profile and adds to it several shape functions

$$b_i(x) = \left[\sin\left(\pi x^{\frac{\log 0.5}{\log(i/(h+1))}} \right) \right], i = 1..h (9)$$

Proc. of the Second Intl. Conf. on Advances in Mechanical and Automation Engineering - MAE 2015

ISBN: 978-1-63248-045-3 doi: 10.15224/ 978-1-63248-045-3-43

where h is user-prescribed. The resultant lower and upper profiles, respectively, are then given by

$$y = y_b + \sum \alpha_i b_i(x) \quad (10)$$

where y_b is the baseline airfoil profile, which was taken in this study to be the NACA0012 symmetric airfoil, and $\alpha_i \in [0,1]$ are weights to be determined. In this study $h = 10$ functions were used for the upper and lower airfoil curves, which resulted in a total of 20 design variables, namely, the weights α_i . The lift and drag coefficients of candidate airfoils were obtained by using XFoil–a computational fluid dynamics simulation for analysis of subsonic airfoils [10]. Each airfoil evaluation required up to 30 seconds on a desktop computer. Figure 2 gives the formulation of the airfoil problem.

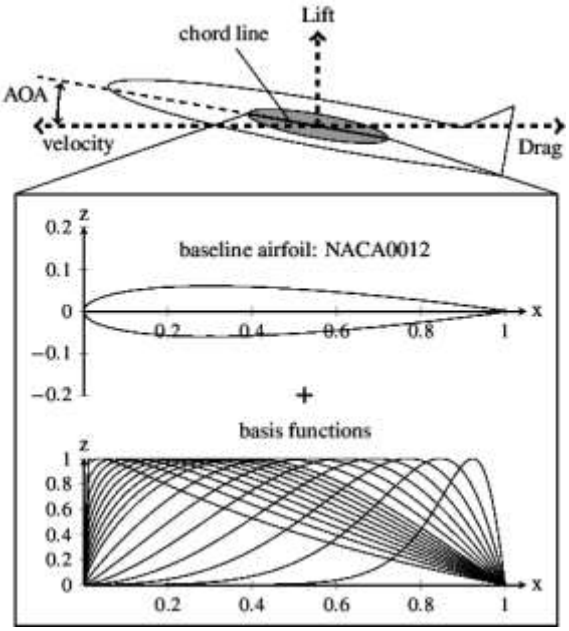


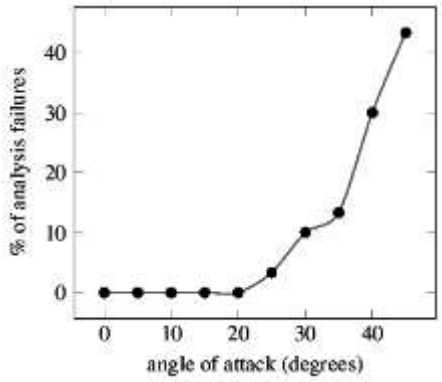
Figure 2: Formulation of the airfoil problem and the Hicks-Henne parameterization

As mentioned above, the airfoil optimization problem is a suitable test case as it contains simulation-infeasible vectors. The prevalence of such vectors mainly depends on the angle of attack (AOA), which is the angle between the airfoil chord line and the velocity, and the flight condition such as speed and altitude. To illustrate the effect of the AOA, 30 different airfoils were evaluated at identical flight conditions, except for the AOA which was increased from 0^o to 40^o . Figure 3 shows the results from which it follows that increasing the angle of attack resulted in more simulation failures. Accordingly, in the numerical tests the settings $AOA = 20^o, 30^o, 40^o$ were used, to ensure many simulation failures would occur during the benchmarking tests. Additionally, the flight condition were fixed at a Mach number of Ma=0.775 (that is, 77.5% of the speed of sound) and an altitude of 32 Kft.

For a thorough evaluation, the proposed framework was compared to two alternative approaches from the literature:

- EA-PS: The evolutionary algorithm with periodic sampling of Ratle [11], which uses a single Kriging metamodel.
- EI-CMAES: A framework which combines the Expected Improvement approach [4] with a covariance matrix adaption evolutionary strategies optimizer.

F



percentage of simulation failures at different angle of attack values.

In all tests the limit was 200 simulation runs, and the initial sample size was fixed at 20 vectors. To support a valid statistical analysis, 30 runs were repeated with each framework in each AOA setting, and Table 1 gives the resultant test statistics of mean and median.

Table 1: Statistics for the objective values

AOA [degrees]	Statistic	Proposed framework	EA-PS	EI-CMAES
20	Mean	**3.688e-01**	4.284e-01	4.977e-01
	Median	**3.687e-01**	4.203e-01	4.220e-01
30	Mean	**7.590e-01**	9.647e-01	7.680e-01
	Median	**7.422e-01**	1.026e+00	7.426e-01
40	Mean	**8.891e-01**	1.107e+00	9.092e-01
	Median	**8.897e-01**	1.105e+00	9.070e-01

The best statistics in each test case are shown in bold.

Results show that the proposed framework consistently outperformed the other two across all test cases, as evident from the mean and median statistics. In particular, the proposed framework maintained its advantage even as the number of failed evaluations increased (higher values of the angle of attack), which shows that it performed well across varied scenarios. From analysis of these results it follows:

- Using an ensemble of metamodels improves the accuracy of the black-box function approximation. This, in turn, improves the effectiveness of the optimization search, when compared to using a single type of metamodel.

ISBN: 978-1-63248-045-3 doi: 10.15224/ 978-1-63248-045-3-43

- In the presence of simulation-infeasible vectors incorporating a classifier into the optimization search can further improve the search by reducing the number of failed simulation runs.

v. Conclusion

Computer simulations are often used in the engineering design process to evaluate candidate designs. In these settings the design process is cast as an optimization problem which involves a computationally expensive black-box function. Furthermore, some simulation runs consistently fail for specific designs, but the reason for failure being unknown. In these challenging settings classical optimization frameworks may perform poorly, and therefore this paper has proposed an optimization framework which: a) uses multiple metamodels concurrently and aggregates their prediction into a single one, and b) incorporates a classifier into the optimization search to predict if a candidate design is likely to crash the simulation. The latter prediction is then used to bias the search away from such designs, without sending them for evaluation with the simulation. Furthermore, to ensure convergence to an optimum of the expensive black-box function, the proposed framework also operates within a trust-region approach. The proposed framework was benchmarked against two competing frameworks from the literature based on an airfoil shape optimization problem. Test results show that the proposed framework consistently outperformed the other frameworks, which thereby highlights the effectiveness of the proposed approach.

References

[1] Y. Tenne and C. K. Goh, eds., Computational Intelligence in Expensive Optimization Problems, vol. 2 of Evolutionary Learning and Optimization. Berlin: Springer, 2010.

[2] Y. Tenne and S. W. Armfield, “A framework for memetic optimization using variable global and local surrogate models,” Journal of Soft Computing, vol. 13, no. 8, pp. 781–793, 2009.

[3] Y. Tenne, K. Izui, and S. Nishiwaki, “A computational intelligence algorithm for expensive engineering optimization problems,” Engineering Applications of Artificial Intelligence, vol. 25, no. 5, pp. 1009–1021, 2012.

[4] D. Büche, N. N. Schraudolph, and P. Koumoutsakos, “Accelerating evolutionary algorithms with Gaussian process fitness function models,” IEEE Transactions on Systems, Man, and Cybernetics–Part C, vol. 35, no. 2, pp. 183–194, 2005.

[5] K. Rasheed, H. Hirsh, and A. Gelsey, “A genetic algorithm for continuous design space search,” Artificial Intelligence in Engineering, vol. 11, pp. 295–305, 1997.

[6] M. T. M. Emmerich, A. Giotis, M. Özedmir, T. Bäck, and K. C. Giannakoglou, “Metamodel-assisted evolution strategies,” in The 7th International Conference on Parallel Problem Solving from Nature–PPSN VII (J. J. Merelo Guervós, ed.), no. 2439 in Lecture Notes in Computer Science, (Berlin), pp. 361–370, Springer, 2002.

[7] S. Handoko, C. K. Kwoh, and Y.-S. Ong, “Using classification for constrained memetic algorithm: A new paradigm,” in Proceedings of the 2008 IEEE International Conference on Systems, Man, and Cybernetics, pp. 547–552, Elsevier, 2008.

[8] A. Chipperfield, P. Fleming, H. Pohlheim, and C. Fonseca, Genetic Algorithm TOOLBOX For Use with MATLAB, Version 1.2. Department of Automatic Control and Systems Engineering, University of Sheffield, Sheffield, 1994.

[9] R. M. Hicks and P. A. Henne, “Wing design by numerical optimization,” Journal of Aircraft, vol. 15, no. 7, pp. 407–412, 1978.

[10] M. Drela and H. Youngren, XFOIL 6.9 User Primer. Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, 2001.

[11] A. Ratle, “Optimal sampling strategies for learning a fitness model,” in The 1999 IEEE Congress on Evolutionary Computation–CEC 1999, (Piscataway, New Jersey), pp. 2078–2085, IEEE, 1999.

[12] R. G. and Regis and C. A. Shoemaker, “ORBIT: Optimization by radial basis function interpolation in trust-regions,” vol. 30, no. 6, pp. 3197–3219, 2008.

[13] A. R. Conn, K. Scheinberg, and L. N. Vicente, Introduction to Derivative-Free Optimization (MPS-Siam Series on Optimization. Philadelphia, Pennsylvania: SIAM, 2009.

[14] J. D. Martin and T. W. Simpson, “Use of Kriging models to approximate deterministic computer models,” AIAA Journal, vol. 43, no. 4, pp. 853–863, 2005 .