ATLANTIS PRESS

Advances in Intelligent Systems Research, volume 143

2nd International Conference on Applied Mathematics, Modelling and Statistics Application (AMMSA 2018)

An Integrated Design Optimization Solving Platform Based on Surrogate-in-Process

Qian Yin, Boxing Wang, Yizhong Wu* and Xin Liu

National CAD Supported Software Engineering Centre, Huazhong University of Science and Technology, Wuhan, PR China

*Corresponding author

***Abstract*—In the paper, a visual and integrated design optimization platform based on Surrogate-in-Process (SIP), called FlowComputer, is developed to support modular process modeling in intuitive and explicit manners. Besides discipline analysis components and generic modular controlling components, several surrogate-related components, including design of experiment (*DOE*) component, surrogate model (*SM*) component, referenced surrogate (*RefS*) component and system analysis (*SA*) component, are designed to support the implementation of various surrogate-based design process by dragging-and-dropping. Demonstration on a test problem indicates that the SIP-based platform provides the ability to define general optimization design process based on surrogate models.**

Keywords—surrogate-in-process (SIP); design optimization solving platform; surrogate reference; system analysis; flowComputer

I. INTRODUCTION

Various multidisciplinary design optimization (MDO) frameworks [1, 2] have been proposed to address the challenge of couplings in engineering systems. And surrogate techniques, also called response surface method, are now playing increasingly important roles in MDO process due to the computationally expensive cost to high fidelity simulations. Viana et al. [3] reviewed various types of surrogates and their applications in engineering. By constructing approximations of discipline simulation codes, surrogate-based design could decrease the counts of computational expensive analyses [4, 5]. Within concurrent subspace optimization with response surfaces (CSSO/RS) framework [6], surrogate models of coupling variables to design variables are employed to evaluate the values of coupling variables without iterative multidiscipline system analysis.

Diverse discipline analysis tools, various optimization algorithms, and different surrogate technologies, which typically involved in engineering design, further complicates engineering problem solving. Efficient problem design environments are necessary for practitioners to facilitate MDO implementation on complex engineering problems.

Various problem solving platforms are developed to provide collaborative and integrated design environments. Web technologies are also employed in the development of MDO platforms to utilize design resources geographically distributed [7-11]. Commercial MDO software tools, i.e., ModelCenter [12], iSIGHT [13], VisualDOC [14], et al., are often equipped with friendly graphical interfaces and capacities for data visualization, provide the ability to integrate commercial tools or legacy codes, integrate diverse design optimization algorithms, and support distributed and parallel computations. These commercial platforms are mainly focused on the integration of discipline tools and the capability for various design exploration methods [15, 16]. Simple MDO frameworks, which can be converted into one or multiple nonlinear programming problems, i.e., individual discipline feasible (IDF) [17], collaborative optimization (CO) [18], can be implemented directly based on optimizer-like components and wrapped analysis components within some commercial software tools [13, 19]. Within these platforms, surrogate models are generally embedded in particular algorithms, or typically serve as fixed approximations of discipline simulations, and is hard to be further improved during during the solving process.

Several open source MDO platforms, i.e., DAKOTA [20], pyMDO [21], OpenMDAO [22], Rave [23], are also developed to provide a variety of optimization, surrogate and other design methods. OpenMDAO and pyMDO could support automatic implementations of different MDO frameworks and their variants from specific problem descriptions [22, 24]. Graphical user interfaces supporting visual and intuitive implementations of different engineering designs are generally less provided in these open accessible platforms.

To provide intuitive and flexible implementation of engineering design processes based on surrogates, a visual and integrated design optimization platform based on Surrogate-in-Process (SIP), called FlowComputer, is developed in the study. Besides various discipline integration components and generic process controlling components, i.e., *Loop* component, *Parallel* component, *Optimizer* component, et al., diverse surrogate-related components, including design of experiment (*DOE*) component, surrogate model (*SM*) component, referenced surrogate (*RefS*) component and system analysis (*SA*) component, are designed to support implementations of design process based on SIP. All these components could be incorporated in the definition of various MDO processes by dragging-and-dropping in the FlowComputer platform.

The rest of the paper is organized as follows. In next section, an integrated MDO platform, called FlowComputer, is introduced, and discipline integration and general flow controlling are described. Section 3 details the surrogate-related components and the Surrogate-in-Process approach. The implementation of an MDO framework based on SIP is also investigated. In Section 4, the demonstration of a surrogate-

based MDO framework on an MDO problem is presented. Conclusions and future works are presented in Section 5.

II. FLOWCOMPUTER: AN INTEGRATED MDO ENVIRONMENT BASED ON PROCESS

FlowComputer is designed to provide a visual and intuitive problem solving environment for process-based implementations of various engineering designs. The design process comprises different kinds of functional components to implement discipline analysis, flow controlling, design exploration, and other design functionalities.

The main user interface of FlowComputer, shown as Figure 1, is composed of flow view, components tree view, components class view and components list view. According to a selected component class in the component class view, available components are displayed in the component list view. The components are able to be dragged and dropped into the flow view, and a particular design flow is organized by selected components following certain criterions. All components are displayed as a tree in the components tree view. Once an engineering design process is well defined, it can be executed automatically and monitored visually, and results can be displayed graphically.

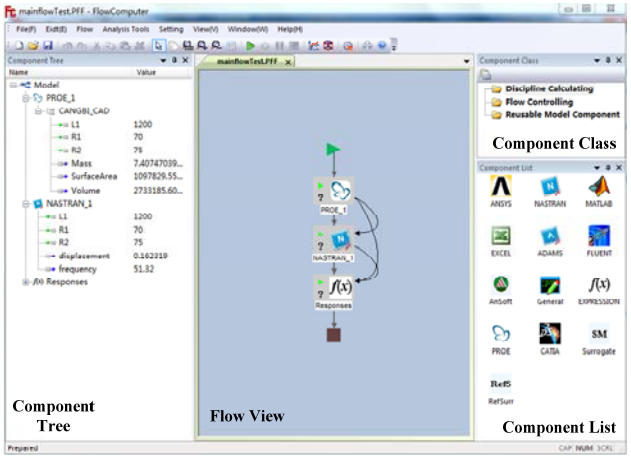

FIGURE I. THE MAIN USER INTERFACE OF FLOWCOMPUTER

A. Discipline Integration Based on COTS Wrapping

By commercial-off-the-shelf (COTS) wrapping, FlowComputer could integrate commercial software tools, or legacy codes, as discipline calculation components in engineering design processes. One of the major COTS wrapping approaches to discipline integration is the In-Process-Out (IPO) method, in which input files (I), process program (P) and output files (O) are used to integrate discipline tools. Another approach, plug-in method, is used to extract input and output variables from the model file of a third-party software tool by its API interface. Currently, most discipline analysis tools, such as Ansys, Nastran, Adams, Abqus, Fluent, Ansoft, MATLAB, Pro/E, CATIA, Excel, and so on, can be wrapped in FlowComputer. And a generic discipline wrapper is provided to integrate various discipline analysis tools, especially the legacy simulation codes. In addition, *Expression* component is also provided to evaluate a set of explicit expressions.

B. Flow Controlling Components

Design flows usually include sequential flow, parallel flow, loop flow, branch flow, subflow, et al. By these flows, various components could be organized as a complex design process. In FlowComputer, sequential flow is the default flow, and the other flows are implemented by generic flow controlling components, including *Parallel* component, *Loop* component, *If/Else* component, *Subflow* component, et al. Flow controlling components could have multiple flow branches, and discipline components can be dragged and dropped onto specific branch of the controlling components. Components located on different branches are executed following the controlling rules of the components.

Furthermore, a serial of special flow controlling components, e.g., optimizer (*OPT*) component, et al., are design to carry out various design exploration methods. The definition is similar to the approach of generic flow controlling components. Design exploration algorithms are integrated to carry out corresponding methods.

An example of *OPT* component is shown as Figure 2. The generic discipline component, entitled TextBook, on the branch is performed iteratively following the selected optimization algorithm. Diverse optimization algorithms, integrated from OPB package self-developed, DAKOTA optimization library [20] and NLopt package [25] by ".dll", are available in current version.

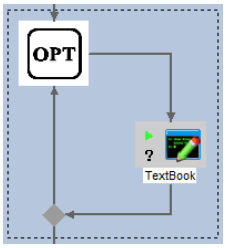

FIGURE II. AN EXAMPLE OF *OPT* COMPONENT

In the flow view, once double clicking the icon of a flow controlling component, a setting dialog will be popped up. Figure 3 shows the dialog for an *OPT* component to define the optimization problem, whose elements include design variables, constraints, objectives, optimization algorithm selected, et al. Variables on the component tree can be dragged and dropped into corresponding locations as design variables, constraints, or objectives.

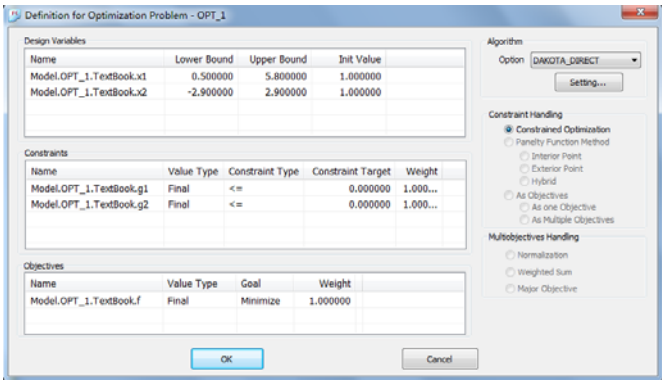

FIGURE III. THE DEFINITION OF AN OPTIMIZATION PROBLEM

III. IMPLEMENTATION OF SURROGATE-IN-PROCESS

A. *Design Optimization Based on Surrogate*

Surrogate techniques could reduce function evaluation cost and enhance engineering design efficiency by modeling approximation to discipline simulation and providing optimization support [26]. The general flowchart of surrogate-based optimization is shown as Figure 4. A sampling method and expensive evaluations on simulation models are performed to generate a set of design variables and corresponding responses, and to further construct particular surrogate models. Model validation and improving surrogate with new points are optional. Thus, optimizations can be implemented directly on surrogate, or be executed repeatedly with sequentially improving accuracy of surrogate.

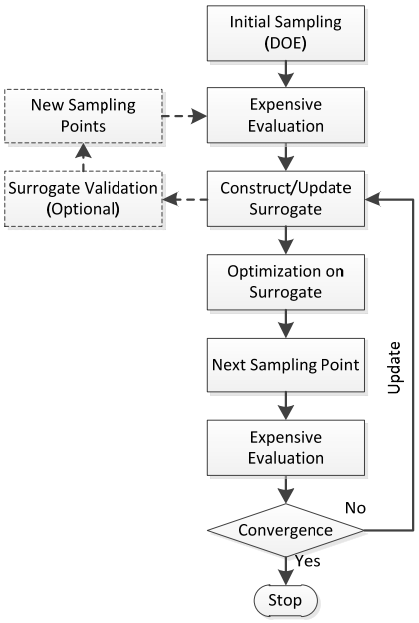

FIGURE IV. THE GENERAL FLOWCHART OF SURROGATE-BASED OPTIMIZATION

This paper is focused on providing a general approach of constructing surrogate-based processes using diverse functional components, entitled Surrogate-In-Process (SIP), to apply various surrogate techniques. The construction and application of surrogate are typically implemented through a serial of stages: sampling, surrogate building, surrogate evaluating, surrogate updating, et al. Hence, several surrogate-related components, including design of experiment (*DOE*) component, surrogate model (*SM*) component, referenced surrogate (*RefS*) component and system analysis (*SA*) component, are designed to implement these operations on surrogates.

B. *Surrogate-related Components*

- Design of experiment (*DOE*) component

DOE component is responsible for generating initial sampling matrix and further obtaining corresponding responses by discipline simulations or multidisciplinary system analyses. An example of a *DOE* component is shown as Figure 5. A component, named *CA*, on the branch is performed for each sample point in the design matrix generated by selected sampling method.

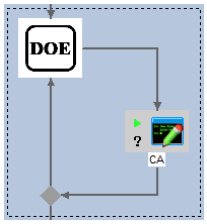

FIGURE V. AN EXAMPLE OF *DOE* COMPONENT

A dialog of setting for *DOE* component is shown as Figure 6. Design variables and responses can be drag-and-dropped from the component tree view, and the bounds of design variables are required. The sampling method can be selected from the dropdown list box, and levels for the method need be specified.

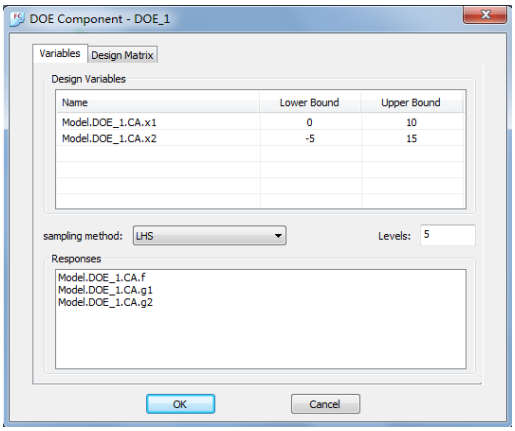

FIGURE VI. THE SETTING FOR *DOE* COMPONENT

- Surrogate model (*SM*) component

SM component is designed to store surrogate model information, such as surrogate model objects, surrogate type and settings, design variables, response variables, current sampling points, et al. According the information, a surrogate model of particular type could be built. The model could be updated using new sampling points during the solving process. Usually, sampling points for surrogate building come from the

outputs of a particular *DOE* component. In current version, Kriging models [27], radial basis functions (RBF) models [28] and polynomial regression models [29], are available.

After an *SM* component is dragged and dropped into some design process, the definition of surrogate model is carried out in the dialogue shown as Figure 7. Surrogate type and corresponding settings, and data source can be specified. In general, the design variables and responses are able to be dragged and dropped from the component tree view.

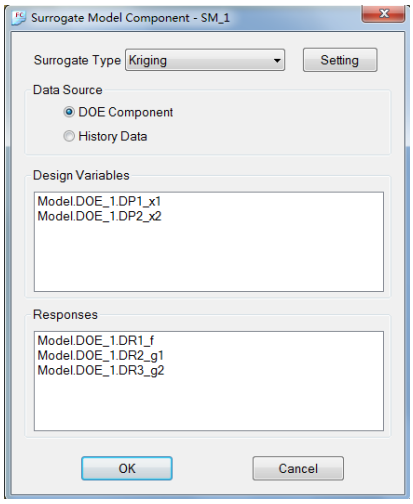

FIGURE VII. THE SETTING FOR *SM* COMPONENT

■ Referenced surrogate (*RefS*) component

RefS component provides various operations on the referenced surrogate model, including evaluating the values of responses to a set of design variables, updating surrogate model with new generated points, and so on. *RefS* component can be dragged and dropped into specified location of some design process to implement particular operation on the referenced surrogate. Figure 8 shows the setting dialog for the *RefS* component, and the referenced *SM* component and corresponding operation on the model can be specified. Once the referenced surrogate is well built, *RefS* component can be employed to evaluate the response values to appointed design variables, or to update/rebuild the referenced *SM* model with new sampling points during design process.

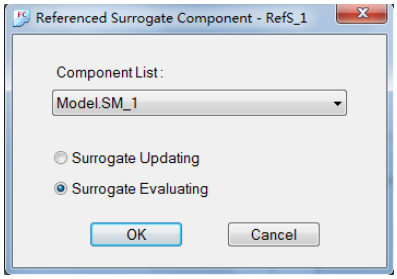

FIGURE VIII. THE SETTING FOR *REFS* COMPONENT

In current version of FlowComputer, evaluating and updating on the referenced model are provided. Other operations on referenced surrogate models, i.e., maximizing the minimal distance of a new sample point to existing points in design space, obtaining the maximal curvature radius of a particular surrogate model, obtaining the maximal value of an expected improvement function for a Kriging model, et al, will be carried out in future.

■ System analysis (*SA*) component

If some disciplines depend on each other, system analysis (*SA*) component is required to archive the multidiscipline feasibility. With the *SA* component, coupled disciplines can be dragged and dropped onto the branch of the component, and dependence relationships can be constructed. When the *SA* component is executed, the minus links, representing feedback couplings, are torn temporary to eliminate the dependence on each other, and an iterative procedure is performed to enforce feedback coupling variables to be converged.

Figure 9 shows an example of *SA* component with two generic discipline components depending on each other. The two components are performed iteratively until the compatibility consistent tolerances to feedback couplings are converged. The functionality of system analysis is also embedded in other special flow controlling components, i.e., *DOE* component, so that system analysis is performed directly within each procedure of the components.

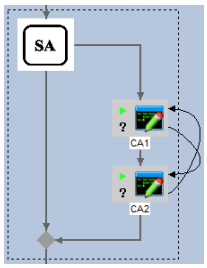

FIGURE IX. AN EXAMPLE OF *SA* COMPONENT

C. SIP-based Implementations of MDO Frameworks

The general components and the surrogate-related components can be incorporated to define various optimization processes based on surrogate techniques. In this section, the SIP-based implementation of MDF/RS framework is investigated.

Multidisciplinary feasible (MDF) [17] treats the original problem as traditional constrained problem, and performs system analysis in each function evaluation, which may result in numerous discipline analyses and extra computational cost. Surrogate models of the objective function and the constraints could be employed to replace the computational expensive system analysis, and enhance the efficiency of MDF.

For an MDO problem, the general optimization problem of MDF/RS is as formulation (1).

$$\begin{aligned} \text{minimize} \quad & \tilde{f}(Z, X) \\ \text{with respect to} \quad & Z, X \\ \text{subject to} \quad & \tilde{g}(Z, X) \le 0 \end{aligned} \tag{1}$$

where Z and X represent the global and local design variables, $\tilde{f}$ and $\tilde{g}$ represent the surrogate models of the objective function and the constraints with respect to design variables.

The implementation of MDF/RS on a MDO problem with two disciplines is shown as Figure 10. Here, the *DOE*, *OPT* and *SA* components are collapsed. The *DOE* component generates a serial of initial sampling points, and *SM* component builds particular surrogate models using the points. Then, the example begins an iterative process using *Loop* component until particular criteria are satisfied. During each *Loop* cycle, the *OPT* component find an optimal solution using *RefS* component for evaluating, named *Surr_Evaluation*, the values of responses, the optimal solution is passed to an *SA* component to obtain responses on the true model, and if the surrogate model is not accurate enough at the optimal solution, *RefS* component for updating, named *Surr_Update*, is performed to update the surrogate model. The *Expression* component, named *Converger*, is used to calculate the error between current optimum and previous optimum, and the error of responses between surrogate model and true simulation model.

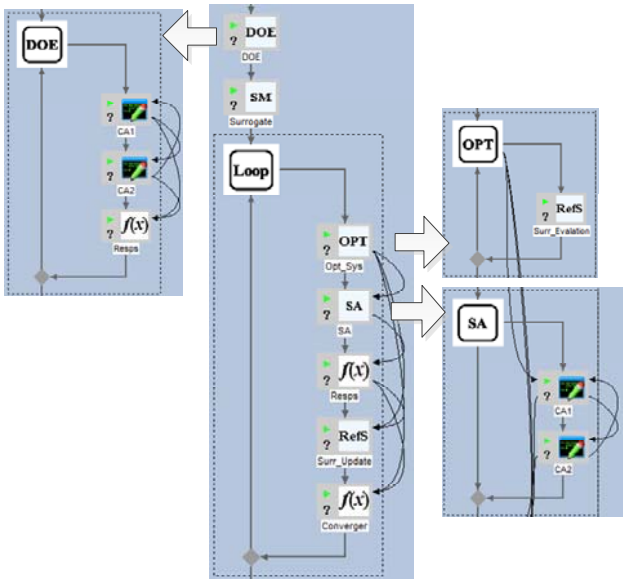

FIGURE X. THE IMPLEMENTATION OF MDF/RS

IV. TESTING OF MDO BASED ON SURROGATE

The surrogate-based MDF framework on a MDO problem, Sellar problem [6], is implemented in the SIP-based platform. Optimization results are evaluated with relative error of objective as formulation (2), and distance to the optimal solution as formulation (3).

$$\varepsilon_f = \left| \frac{f^* - f}{f} \right| \tag{2}$$

where ε_f denotes the relative error of objective, f is the theoretical optimum, and f^* is the optimal value that a particular MDO framework found.

$$l = \left\| X^* - X \right\|_2 \tag{3}$$

where l is the distance to the optimal solution, X is the theoretical optimal design variables, and X^* is the optimal design variables found.

A. The Sellar Problem

The Sellar problem [6], shown as formulation (4), is a commonly used MDO test problem with two disciplines. In the following text, the two disciplines are represented by '*CA1*' and '*CA2*', respectively. The global optimum of the Sellar problem is (z_1*, z_2*, x_1*) = (1.9776, 0, 0), where the objective value is f* = 3.18339.

$$\begin{aligned} \text{minimize} \quad & x_1^2 + z_2 + y_1 + e^{-y_2} \\ \text{with respect to} \quad & z_1, z_2, x_1 \\ \text{subject to} \quad & 1 - y_1 / 3.16 \le 0 \\ & y_2 / 24 - 1 \le 0 \\ & y_1 = z_1^2 + x_1 + z_2 - 0.2 y_2 \\ & y_2 = \sqrt{y_1} + z_1 + z_2 \\ & -10 \le z_1 \le 10 \\ & 0 \le z_2 \le 10 \\ & 0 \le x_1 \le 10 \end{aligned} \tag{4}$$

where y_1 and y_2 are the output variables of the two disciplines.

In FlowComputer, the SIP-based design process is similar to Figure 10. The selected optimization algorithm is SLSQP from NLopt package [25], and the type of surrogate is selected as Kriging [27] in DAKOTA surfpack [20]. Gradient information is obtained by central finite difference method with a step of 10^{-4}.

B. Test Results and Discussion

The problem is successfully solved. Table 1 presents the optimization results starting from the point (z_1, z_2, x_1) = (5.0, 2.0, 1.0). The relative error of objective and the l2-norm distance to the exact optimum are also listed. The data indicate that the MDF/RS framework solved the problem with well accuracy.

TABLE I. THE OPTIMUM SOLUTIONS FOUND AND ERRORS TO THE EXACT OPTIMUM OF THE SELLAR PROBLEM

	Test result
z_1*	1.9776
z_2*	0
x_1*	0
f*	3.18339
ε_f	5.12×10^{-7}
l^2-norm	4.82×10^{-7}

The numbers of system analysis and the number of discipline evaluations are listed in Table2. As surrogate models are used in these frameworks, the numbers indicate the function evaluation counts of true models.

TABLE II. THE NUMBER OF EVALUATIONS FOR THE SELLAR PROBLEM

	Number of Evaluation
System Analysis	16
CA1	103
CA2	103

Note: the numbers denote the evaluation counts of true model, excluding surrogate model.

The convergence history of the MDF/RS implementation is presented in Figure 11. The iterations are reported by the system optimization optimum at each main cycle (*Loop* component). In Figure 11, the former 7 points are the initial sampling points employed to build surrogate models for couplings to design variables. Here, Latin hypercube sampling (LHS) method with 7 levels is used for the *DOE* component. After DOE, MDF/RS presents rapid convergences with well improvements.

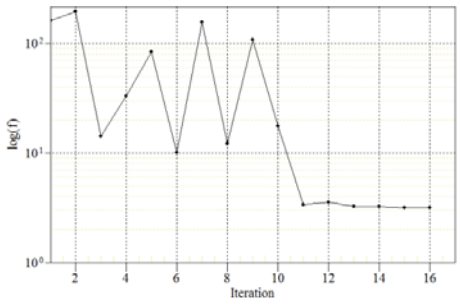

FIGURE XI. THE CONVERGENCE OF MDF/RS ON THE SELLAR PROBLEM

V. CONCLUSIONS

A visual and integrated design optimization platform based on Surrogate-in-Process (SIP), called FlowComputer, is developed in the study. Besides various discipline integration components and generic process controlling components, several surrogate-related components, i.e., design of experiment (*DOE*) component, surrogate model (*SM*) component, referenced surrogate (*RefS*) component and system analysis (*SA*) component, are designed to support the implementation of SIP-based design process.

A surrogate-based MDO framework, MDF/RS, is implemented on an MDO problem, and the framework could solve the problem with equivalent accuracy to the optimal solutions. The implementation demonstrates that the SIP-based platform provides the ability to define design exploration process.

The future work includes two aspects: 1) implementations of other MDO frameworks in FlowComputer and implementations on more test cases should be further investigated; 2) other surrogate referencing operations, i.e., maximizing the minimal distance of a new sample point to existing points in design space, obtaining the maximal value of an expected improvement function for a Kriging model, et al., should be incorporated into the platform.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China [grant number 51575205 and 51775472].

REFERENCES

[1] Sobieszczanski-Sobieski, J. and Haftka, R.T., Multidisciplinary aerospace design optimization: survey of recent developments. Structural optimization, 1997. 14(1): p. 1-23.

[2] Martins, J.R.R.A. and Lambe, A.B., Multidisciplinary Design Optimization: A Survey of Architectures. AIAA Journal, 2013. 51(9): p. 2049-2075.

[3] Viana, F.A.C., Simpson, T.W., Balabanov, V., et al., Special Section on Multidisciplinary Design Optimization: Metamodeling in Multidisciplinary Design Optimization: How Far Have We Really Come? AIAA Journal, 2014. 52(4): p. 670-690.

[4] Sobieski, I.P. and Kroo, I.M., Collaborative Optimization Using Response Surface Estimation. AIAA Journal, 2000. 38(10): p. 1931-1938.

[5] Kodiyalam, S. and Sobieszczanski-Sobieski, J., Bilevel Integrated System Synthesis with Response Surfaces. AIAA Journal, 2000. 38(8): p. 1479-1485.

[6] Sellar, R., Batill, S., and Renaud, J., Response surface based, concurrent subspace optimization for multidisciplinary system design, in 34th Aerospace Sciences Meeting and Exhibit. 1996, American Institute of Aeronautics and Astronautics.

[7] Scott, W., Hongman, K., James, M., et al., A Web Centric Architecture for Deploying Multi-Disciplinary Engineering Design Processes, in 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference. 2004, American Institute of Aeronautics and Astronautics.

[8] Wang, Y.D., Shen, W.M., and Ghenniwa, H., WebBlow: a Web/agent-based multidisciplinary design optimization environment. Computers in Industry, 2003. 52(1): p. 17-28.

[9] Lee, H.-J., Lee, J.-W., and Lee, J.-O., Development of Web services-based Multidisciplinary Design Optimization framework. Advances in Engineering Software, 2009. 40(3): p. 176-183.

[10] Sobolewski, M., Object-Oriented Service Clouds for Transdisciplinary Computing, in Cloud Computing and Services Science, I. Ivanov, M. van Sinderen, and B. Shishkov, Editors. 2012, Springer New York. p. 3-31.

[11] Liu, C.W., Jin, X.X., and Li, L.S., A web services-based multidisciplinary design optimization framework for complex engineering systems with uncertainties. Computers in Industry, 2014. 65(4): p. 585-597.

[12] Kim, H., Malone, B., and Sobieszczanski-Sobieski, J., A distributed, parallel, and collaborative environment for design of complex systems. AIAA Paper, 2004. 1848.

[13] Koch, P.N., Evans, J.P., and Powell, D., Interdigitation for effective design space exploration using iSIGHT. Structural and Multidisciplinary Optimization, 2002. 23(2): p. 111-126.

[14] Vladimir, B., Christophe, C., Dipankar, G., et al., VisualDOC: A Software System for General Purpose Integration and Design

Optimization, in 9th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization. 2002, American Institute of Aeronautics and Astronautics.

[15] Weck, O.d., Agte, J., Sobieszczanski-Sobieski, J., et al., State-of-the-Art and Future Trends in Multidisciplinary Design Optimization, in 48th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference. 2007, American Institute of Aeronautics and Astronautics.

[16] Agte, J., de Weck, O., Sobieszczanski-Sobieski, J., et al., MDO: assessment and direction for advancement—an opinion of one international group. Structural and Multidisciplinary Optimization, 2010. 40(1-6): p. 17-33.

[17] Cramer, E.J., Dennis, J.J.E., Frank, P.D., et al., Problem Formulation for Multidisciplinary Optimization. SIAM Journal on Optimization, 1994. 4(4): p. 754-23.

[18] Braun, R.D., Collaborative optimization: An architecture for large-scale distributed design. 1996, Stanford University: Ann Arbor. p. 207-207 p.

[19] Tiwari, S., Dong, H., Lankalapalli, S., et al., Multidisciplinary Optimization with VisualDOC, in 10th World Congress on Structural and Multidisciplinary Optimization. 2013: Orlando, Florida, USA.

[20] Adams, B.M., Bauman, L.E., Bohnhoff, W.J., et al., Dakota, a multilevel parallel objectoriented framework for design optimization, parameter estimation, uncertainty quantification, and sensitivity analysis: Version 5.3.1 user's manual. 2013: Technical Report SAND2010-2183, Sandia National Laboratories, Albuquerque, NM, Updated Jan. 2013.

[21] Martins, J.R.R.A., Marriage, C., and Tedford, N., pyMDO: An Object-Oriented Framework for Multidisciplinary Design Optimization. ACM Transactions on Mathematical Software, 2009. 36(4): p. 1-25.

[22] Gray, J., Moore, K.T., Hearn, T.A., et al., Standard Platform for Benchmarking Multidisciplinary Design Analysis and Optimization Architectures. AIAA Journal, 2013. 51(10): p. 2380-2394.

[23] Daskilewicz, M.J. and German, B.J., Rave: A Computational Framework to Facilitate Research in Design Decision Support. Journal of Computing and Information Science in Engineering, 2012. 12(2): p. 021005-021005.

[24] Tedford, N.P. and Martins, J.R.R.A., Benchmarking multidisciplinary design optimization algorithms. Optimization and Engineering, 2010. 11(1): p. 159-183.

[25] Johnson, S.G. The NLopt nonlinear-optimization package. Available from: http://ab-initio.mit.edu/nlopt.

[26] Wang, G.G. and Shan, S., Review of Metamodeling Techniques in Support of Engineering Design Optimization. Journal of Mechanical Design, 2006. 129(4): p. 370-380.

[27] Krige, D., A Statistical Approach to Some Basic Mine Valuation Problems on the Witwatersrand. Journal of the Chemical, Metallurgical and Mining Society of South Africa, 1951. 52(6): p. 119-139.

[28] Broomhead, D.S. and Lowe, D., Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks. Complex Systems, 1988. 2: p. 321-355.

[29] Box, G.E.P. and Wilson, K.B., On the Experimental Attainment of Optimum Conditions. Journal of the Royal Statistical Society. Series B (Methodological), 1951. 13(1): p. 1-45.