Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 2759762, 9 pages
https://doi.org/10.1155/2017/2759762

Research Article

Performance Evaluation of MDO Architectures within a Variable Complexity Problem

Daiyu Zhang, Baowei Song, Peng Wang, and Yanru He

School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, China

Correspondence should be addressed to Daiyu Zhang; daiyu@mail.nwpu.edu.cn

Received 27 October 2016; Accepted 7 February 2017; Published 23 February 2017

Academic Editor: Quang Phuc Ha

Copyright © 2017 Daiyu Zhang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Though quite a number of multidisciplinary design optimization (MDO) architectures have been proposed for the optimal design of large-scale multidisciplinary systems, how their performance changes with the complexity of MDO problem varied is not well studied. In order to solve this problem, this paper presents a variable complexity problem which allows people to obtain a MDO problem with arbitrary complexity by specifying its changeable parameters, such as the number of disciplines and the numbers of design variables. Then four investigations are performed to evaluate how the performance of different MDO architectures changes with the number of disciplines, global variables, local variables, and coupling variables varied, respectively. Finally, the results supply guidance for the selection of MDO architectures in solving practical engineering problems with different complexity.

1. Introduction

Multidisciplinary design optimization (MDO) is a growing field of research in the design of large-scale engineering systems that consist of a number of interacting subsystems. The main motivation for using MDO is that the performance of a multidisciplinary system is driven not only by the individual disciplines but also by their interactions. By coordinating each discipline and decoupling their interaction reasonably in the design cycle, designers can improve the design and reduce the time and cost simultaneously.

Until now, many MDO architectures have been proposed for the optimal design of large-scale engineering systems. Generally, the architectures can be divided into two classes: monolithic architectures and distributed architectures. Monolithic architectures solve the MDO problem by casting it as a single optimization problem, which is easy to be implemented for small MDO problems. But for complicated engineering systems where people in charge of each discipline work independently of one another, these architectures may encounter big difficulty in integrating all the disciplines together. The monolithic architectures commonly include multidisciplinary feasible (MDF) [1–3], Individual Discipline Feasible (IDF) [1, 2, 4], and Simultaneous Analysis and Design (SAND) [1, 5]. However, the distributed architectures solve the MDO problems by decomposing it into smaller and more manageable discipline optimization problems that have the same solution when reassembled. One big advantage of these architectures is that they promote discipline autonomy and make people in different groups and teams just be in charge of their own fields and use their own discipline legacy. The distributed architectures usually include Concurrent Subspace Optimization (CSSO) [6, 7], Collaborative Optimization (CO) [8–10], Bilevel Integrated Systems Synthesis (BLISS) [11], Bilevel Integrated Systems Synthesis 2000 (BLISS-2000) [12, 13], Analytical Target Cascading (ATC) [14, 15], and MDO based on independent subspaces (MDOIS) [16].

Though many MDO architectures are available, the big challenge is to determine which architecture is the most efficient for a given MDO problem. To solve the problem, a benchmarking study of different MDO architectures needs to be carried out. Though many researches [17–23] have done excellent work in comparing the performances of different MDO architectures and Martins et al. [24] have made an initial exploration in assessing the computational complexity of MDO architectures, there are still some limitations of them. Firstly, many of the test MDO problems are of low dimensionality with few disciplines and variables. They lack the ability to test the performance of different MDO architectures for the practical engineering problems which are often composed of hundreds of design variables, constraints, and coupling variables. Secondly, the MDO architectures may perform differently when the test problem has different number of disciplines, design variables, constraints, or coupling variables. How the performance of MDO architectures changes with the complexity of MDO problem varied is not well studied.

In order to deal with the problems above, two works will be done in this paper. The first one is to present a variable complexity problem which is a nonseparable nonlinear problem that allows people to specify its complexity, such as the number of disciplines, design variables, and coupling variables. This makes it feasible to test the performance of different MDO architectures not only for high complexity problem, but also for problem with arbitrary complexity. The second one is to implement different MDO architectures to solve the variable complexity problem in four cases to evaluate how the performance of MDO architectures changes with the complexity of MDO problem varied.

The remainder of this paper is organized as follows. In Section 2, we state the terminology and mathematical notation that will be used throughout this paper. In Section 3, we present a variable complexity problem whose complexity can be varied by the changeable parameters. In Section 4, we give the specific formulations of four MDO architectures associated with the variable complexity problem. In Section 5, we evaluate how the performance of MDO architectures changes with the complexity of MDO problem varied. Finally, some conclusions are presented in the last section.

2. Terminology and Mathematical Notation

Before introducing the variable complexity problem, we need to describe the terminology and mathematical notation that will be used throughout this paper.

Firstly, we define and clarify several terms that are specific to the field of MDO. Discipline analysis is usually a simulation that models the performance of one discipline in a multidisciplinary system. Global variables are the design variables that are shared by multiple disciplines. Local design variables are the design variables that only apply to one discipline. Coupling variables are the outputs that one discipline passes to other disciplines. Local constraints are the constraints that are only decided by one discipline. Global constraints are the constraints that are determined by multiple disciplines.

Then the common mathematical notation that will be used in the following sections is listed as follows. Note that all vectors in this paper are assumed to be column vectors unless indicated otherwise.

Mathematical Notation for MDO Problem

$\mathbf{x}_0$: the vector of global variables shared by more than one discipline

x_{0i}: the ith element of $\mathbf{x}_0$

$\mathbf{x}_i$: the vector of local design variables included in discipline i

x_{ij}: the jth element of $\mathbf{x}_i$

$\mathbf{y}_i$: the vector of coupling variables included in discipline i

y_{ij}: the jth element of $\mathbf{y}_i$

$\mathbf{c}_i$: the vector of local constraints included in discipline i

c_{ij}: the jth element of $\mathbf{c}_i$

$\mathbf{c}_0$: the vector of global constraints depending on the variables of multiple disciplines

c_{0i}: the ith element of $\mathbf{c}_0$

3. Variable Complexity Problem

In this section, a variable complexity problem is proposed to allow the investigation that how the performance of MDO architectures changes with its complexity varied. It is a nonseparable nonlinear MDO problem which provides the ability to specify the changeable parameters in the following. Those parameters can be used to change the problem complexity as the user desires.

Changeable Parameters in Variable Complexity Problem

N: the number of disciplines

n_{x_0}: the number of global variables

n_{x_i}: the number of local variables included in discipline i

n_{y_i}: the number of coupling variables included in discipline i

n_{c_i}: the number of local constraints included in discipline i

n_{c_0}: the number of global constraints

The remaining part of this section is laid out as follows to introduce the variable complexity problem. Firstly, the discipline analysis, constraint functions, and objective function of the variable complexity problem are described in detail, respectively. Then the monolithic mathematical model of the variable complexity problem is built and a simple example consisting of three disciplines is given for a more vivid description.

3.1. Discipline Analysis. A discipline analysis is a simulation that models the behavior of one discipline. Generally, performing a discipline analysis is to solve an equation group, such as the Navier-Stokes equation in fluid mechanics, the static equilibrium equation in structural mechanics, or the equations of motion in control simulations. So a set of equations are created to represent the analysis of discipline i in the variable complexity problem as follows.

$$\mathbf{B}_{ii}\mathbf{y}_i = -\mathbf{C}_i\mathbf{x}_0 - \mathbf{D}_i\mathbf{x}_i - \sum_{j=1,j\neq i}^{N} \mathbf{B}_{ij}\mathbf{y}_j, \qquad (1)$$

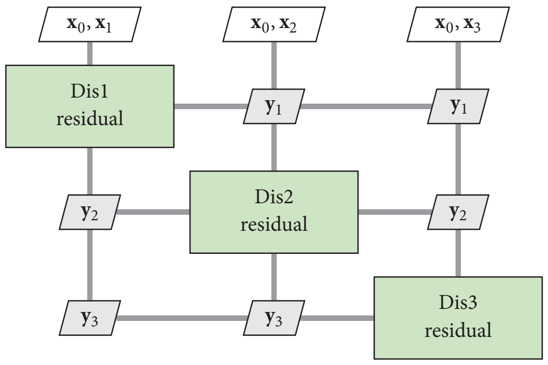


Figure 1: The coupling relationship of the three disciplines.

where $\mathbf{y}_j$ is the vector of coupling variables of discipline j, $\mathbf{B}_{ij}$ is a coefficient matrix of size $n_{y_i} \times n_{y_j}$, $\mathbf{C}_i$ is a coefficient matrix of size $n_{y_i} \times n_{x_0}$, and $\mathbf{D}_i$ is a coefficient matrix of size $n_{y_i} \times n_{x_i}$. In addition, all elements of $\mathbf{B}_{ij}$, $\mathbf{C}_i$, and $\mathbf{D}_i$ are real values.

If we denote (1) by $\mathbf{R}_i = 0$ in residual form as follows, the values of $\mathbf{y}_i$ can be obtained by solving the equation $\mathbf{R}_i = 0$ and they are the solution of the discipline analysis i

$$\mathbf{R}_i = \mathbf{B}_{ii}\mathbf{y}_i + \mathbf{C}_i\mathbf{x}_0 + \mathbf{D}_i\mathbf{x}_i + \sum_{j=1, j\neq i}^{N} \mathbf{B}_{ij}\mathbf{y}_j = 0. \qquad (2)$$

For (2), it is obvious that different disciplines are coupled with each other through the global variables and the coupling variables. In order to make it more clearly, a simple example consisting of three disciplines is described in Figure 1 by the extended design structure matrix (XDSM) method [25] which makes the description of MDO problems and architectures very convenient and legible.

Another important thing is to determine the coefficient matrices in (2). As for $\mathbf{C}_i$ and $\mathbf{D}_i$, they both can be generated randomly or customized by user prior to the start of the optimization. As for $\mathbf{B}_{ii}$, because the discipline analysis needs to be individual disciplinary feasible, which means (2) needs a feasible solution, $\mathbf{B}_{ii}$ needs to be nonsingular. As for $\mathbf{B}_{ij}$, because the whole multidisciplinary analysis which is composed of multiple discipline analyses needs to be multidisciplinary feasible, which means the combined equations composed of multiple equations (2) need a feasible solution, the combined matrix $\mathbf{B}$ shown in (3) needs to be nonsingular

$$\mathbf{B} = \begin{bmatrix} \mathbf{B}_{11} & \mathbf{B}_{12} & \cdots & \mathbf{B}_{1N} \\ \mathbf{B}_{21} & \mathbf{B}_{22} & \cdots & \mathbf{B}_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{B}_{N1} & \mathbf{B}_{N2} & \cdots & \mathbf{B}_{NN} \end{bmatrix}. \qquad (3)$$

3.2. Constraint Functions. As stated in Section 2, there are two kinds of constraints in the MDO problem. One is the local constraints which depend on only one discipline variables. The other one is the global constraints which depend on multiple discipline variables.

For the variable complexity problem, its local constraints included in discipline i are built below

$$\begin{aligned} c_{i1} &= \mathbf{x}_0^{\mathbf{T}}\mathbf{x}_0 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_i + \mathbf{E}_{i1}^{\mathbf{T}}\mathbf{x}_0 + \mathbf{F}_{i1}^{\mathbf{T}}\mathbf{x}_i + \mathbf{G}_{i1}^{\mathbf{T}}\mathbf{y}_i - r_{i1} \leq 0, \\ c_{i2} &= \mathbf{x}_0^{\mathbf{T}}\mathbf{x}_0 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_i + \mathbf{E}_{i2}^{\mathbf{T}}\mathbf{x}_0 + \mathbf{F}_{i2}^{\mathbf{T}}\mathbf{x}_i + \mathbf{G}_{i2}^{\mathbf{T}}\mathbf{y}_i - r_{i2} \leq 0, \\ &\vdots \\ c_{in_{c_i}} &= \mathbf{x}_0^{\mathbf{T}}\mathbf{x}_0 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_i + \mathbf{E}_{in_{c_i}}^{\mathbf{T}}\mathbf{x}_0 + \mathbf{F}_{in_{c_i}}^{\mathbf{T}}\mathbf{x}_i + \mathbf{G}_{in_{c_i}}^{\mathbf{T}}\mathbf{y}_i - r_{in_{c_i}} \\ &\leq 0, \end{aligned} \qquad (4)$$

where $\mathbf{E}_{ij}$ $(j = 1, 2, \ldots, n_{c_i})$ are the coefficient vectors of $\mathbf{x}_0$, $\mathbf{F}_{ij}$ $(j = 1, 2, \ldots, n_{c_i})$ are the coefficient vectors of $\mathbf{x}_i$, $\mathbf{G}_{ij}$ $(j = 1, 2, \ldots, n_{c_i})$ are the coefficient vectors of $\mathbf{y}_i$, and r_{ij} $(j = 1, 2, \ldots, n_{c_i})$ are positive scalars.

Its global constraints are built as follows:

$$\begin{aligned} c_{01} &= \mathbf{x}_0^{\mathbf{T}}\mathbf{x}_0 + \sum_{i=1}^{N}\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_i + \mathbf{H}_1^{\mathbf{T}}\mathbf{x}_0 + \sum_{i=1}^{N}\mathbf{I}_{1i}^{\mathbf{T}}\mathbf{x}_i + \sum_{i=1}^{N}\mathbf{J}_{1i}^{\mathbf{T}}\mathbf{y}_i - s_1 \\ &\leq 0, \\ c_{02} &= \mathbf{x}_0^{\mathbf{T}}\mathbf{x}_0 + \sum_{i=1}^{N}\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_i + \mathbf{H}_2^{\mathbf{T}}\mathbf{x}_0 + \sum_{i=1}^{N}\mathbf{I}_{2i}^{\mathbf{T}}\mathbf{x}_i + \sum_{i=1}^{N}\mathbf{J}_{2i}^{\mathbf{T}}\mathbf{y}_i - s_2 \\ &\leq 0 \\ &\vdots \\ c_{0n_{c_0}} &= \mathbf{x}_0^{\mathbf{T}}\mathbf{x}_0 + \sum_{i=1}^{N}\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_i + \mathbf{H}_{n_{c_0}}^{\mathbf{T}}\mathbf{x}_0 + \sum_{i=1}^{N}\mathbf{I}_{n_{c_0}i}^{\mathbf{T}}\mathbf{x}_i + \sum_{i=1}^{N}\mathbf{J}_{n_{c_0}i}^{\mathbf{T}}\mathbf{y}_i \\ &- s_{n_{c_0}} \leq 0, \end{aligned} \qquad (5)$$

where $\mathbf{H}_i$ $(i = 1, 2, \ldots, n_{c_0})$ are the coefficient vectors of $\mathbf{x}_0$, $\mathbf{I}_{ij}$ $(i = 1, 2, \ldots, n_{c_0};\ j = 1, 2, \ldots, N)$ are the coefficient vectors of $\mathbf{x}_i$, $\mathbf{J}_{ij}$ $(i = 1, 2, \ldots, n_{c_0};\ j = 1, 2, \ldots, N)$ are the coefficient vectors of $\mathbf{y}_i$, and s_i $(i = 1, 2, \ldots, n_{c_0})$ are positive scalars.

Note that all of the coefficient vectors and positive scalars in (4) and (5) can be generated randomly or customized by user prior to the start of the optimization.

3.3. Objective Function. For the objective function of MDO problem, it usually can be divided into two types. One is called the separable objective whose expression is a sum of the local objectives, each of which is formulated just by the variables included in one discipline. The other is called the nonseparable objective whose expression cannot be divided into a sum of local objectives. Because the separable objective unintentionally is preferred to some distributed architectures [16, 26, 27] and the objectives of most engineering problems are nonseparable, the objective of the variable complexity problem is built by a nonseparable function below to show the objectivity and generality

$$\mathbf{f} = \left(\sum_{i=1}^{n_{x_0}} \mathbf{x}_{0i} + \sum_{i=1}^{N} \sum_{j=1}^{n_{x_i}} \mathbf{x}_{ij} + \sum_{i=1}^{N} \sum_{j=1}^{n_{y_i}} \mathbf{y}_{ij} \right)^3 . \quad (6)$$

3.4. Monolithic Model. Based on (2), (4), (5), and (6), the monolithic mathematical model of the variable complexity problem can be obtained as follows by assuming there are N disciplines:

$$\begin{aligned} \text{minimize:}\quad & f\left(\mathbf{x}_0, \mathbf{x}, \mathbf{y}\right) \\ \text{design variable:}\quad & \mathbf{x}_0, \mathbf{x} \\ \text{constraints:}\quad & \mathbf{c}_0\left(\mathbf{x}_0, \mathbf{x}, \mathbf{y}\right) \le 0 \\ & \mathbf{c}_i\left(\mathbf{x}_0, \mathbf{x}_i, \mathbf{y}_i\right) \le 0 \quad i = 1, 2, \ldots, N \\ \text{residuals:}\quad & \mathbf{R}_i\left(\mathbf{x}_0, \mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_{j\neq i}\right) = 0 \\ & \qquad i = 1, 2, \ldots, N, \end{aligned} \quad (7)$$

where $\mathbf{x} = [\mathbf{x}_1^{\mathbf{T}}, \ldots, \mathbf{x}_N^{\mathbf{T}}]^{\mathbf{T}}$, $\mathbf{y} = [\mathbf{y}_1^{\mathbf{T}}, \ldots, \mathbf{y}_N^{\mathbf{T}}]^{\mathbf{T}}$, $\mathbf{c}_0 = [c_{01}, c_{02}, \ldots, c_{0n_{c_0}}]^{\mathbf{T}}$, and $\mathbf{c}_i = [c_{i1}, c_{i2}, \ldots, c_{in_{c_i}}]^{\mathbf{T}}$.

The presented variable complexity problem is composed of the design variables, local constraints, global constraints, and nonseparable objective function. So it is a credible MDO problem to benchmark different MDO architectures without loss of generality. In order to make this problem be easily understood, one simple example consisting of three disciplines is represented in Figure 2 by the XDSM method.

4. MDO Architectures Associated with the Variable Complexity Problem

There are a wide range of MDO architectures, each of which formulates a given problem in a different manner. In this section, four common MDO architectures are introduced to formulate the variable complexity problem. Though only four MDO architectures are included, they serve as templates for future researchers to work from for other MDO architectures. For each architecture above, we firstly make a brief overview of it and then give its specific formulation associated with the variable complexity problem in the remaining section.

4.1. Multidisciplinary Feasible. The multidisciplinary feasible architecture is a traditional MDO approach. Its main idea is to simply place an optimizer over a multidisciplinary analysis (MDA) module that takes in the design variables of the optimizer and iterates between the discipline analyses until all the coupling variables have converged. Because the solution of MDA module is required at each iteration of the optimizer, MDF architecture grantees that the MDO problem is multidisciplinary feasible at each optimization iterations.

An obvious advantage of MDF architecture is that it is easy to be implemented since it is a monolithic architecture where the design variables, objective function, and design constraints are directly under the control of optimizer. Another advantage is that MDF architecture always returns a feasible design even if the optimization is terminated early. This is particularly meaningful in some engineering problems if the goal is just to obtain an improved design that is unnecessary to be strictly mathematically optimal because of the computation difficulty.

The main disadvantage of MDF architecture is that the architecture requires a full MDA to be performed at each optimization iteration, which can be very time-consuming sometimes. Particularly when the gradient-based optimization method is adopted in the optimizer, the gradient calculations are more difficult for MDF architecture because the gradient information in MDF needs to be feasible with all disciplines.

With regard to the variable complexity problem, the specific formulation of MDF architecture can be stated as follows:

$$\begin{aligned} \text{minimize:}\quad & f\left(\mathbf{x}_0, \mathbf{x}, \mathbf{y}\right) \\ \text{design variable:}\quad & \mathbf{x}_0, \mathbf{x} \\ \text{constraints:}\quad & \mathbf{c}_0\left(\mathbf{x}_0, \mathbf{x}, \mathbf{y}\right) \le 0 \\ & \mathbf{c}_i\left(\mathbf{x}_0, \mathbf{x}_i, \mathbf{y}_i\right) \le 0 \quad i = 1, 2, \ldots, N \\ \text{MDA:}\quad & \mathbf{R}_i\left(\mathbf{x}_0, \mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_{j\neq i}\right) = 0 \\ & \qquad i = 1, 2, \ldots, N. \end{aligned} \quad (8)$$

4.2. Individual Discipline Feasible. The Individual Discipline Feasible architecture removes the need for MDA procedure in MDF architecture that guarantees the multidisciplinary feasibility. This architecture enables the discipline analyses to be carried out in parallel by introducing the copies of coupling variables in each discipline and the corresponding consistency constraints. The copies of coupling variables are applied to decouple the discipline analyses so that they no longer depend on the coupling variables generated by other disciplines. The consistency constraints are introduced to make sure a multidisciplinary feasible solution is obtained at the optimal points by constraining the copies of coupling variables to match the actual coupling variables gradually.

An obvious advantage of IDF architecture is that the discipline analyses are decoupled so that they can be performed in parallel, which will greatly increase the optimization efficiency. In addition, IDF architecture is also more robust than MDF architecture in the success rate of optimization because MDF involves a MDA procedure and solving the MDA module may sometimes be prone to error [24].

However, there are also some disadvantages about the IDF architecture. The first one is that the multidisciplinary feasibility is ensured at the last iteration of the optimization, which means the optimization cannot be terminated early for obtaining the optimal design. Another is that the introduction of consistency constraints increases the scale of the optimization problem. In particular, when the number of coupling variables is large, the optimization problem might be too large to solve efficiently.

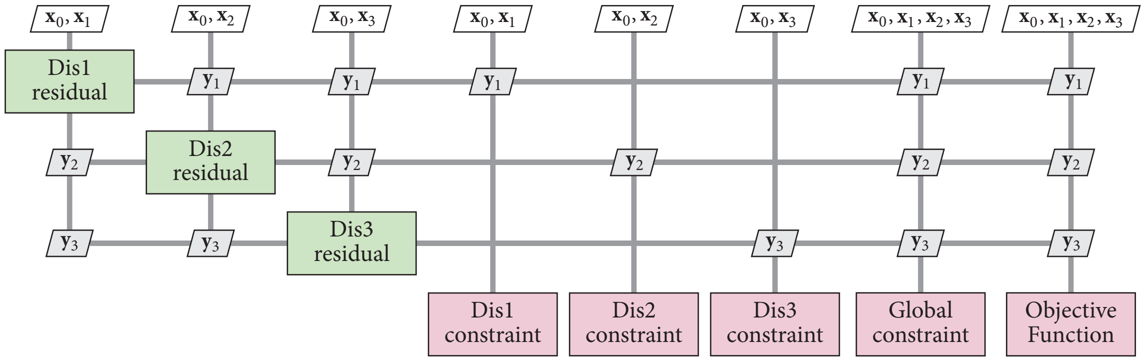


FIGURE 2: The variable complexity problem consisting of three disciplines.

With regard to the variable complexity problem, the specific formulation of IDF architecture can be stated as follows:

$$\begin{aligned}
\text{minimize:}\quad & f(\mathbf{x}_0, \mathbf{x}, \mathbf{y}) \\
\text{design variable:}\quad & \mathbf{x}_0, \mathbf{x}, \hat{\mathbf{y}} \\
\text{constraints:}\quad & \mathbf{c}_0(\mathbf{x}_0, \mathbf{x}, \mathbf{y}) \le 0 \\
& \mathbf{c}_i(\mathbf{x}_0, \mathbf{x}_i, \mathbf{y}_i) \le 0 \quad i = 1, 2, \ldots, N \\
& \hat{\mathbf{y}}_i - \mathbf{y}_i = 0 \quad i = 1, 2, \ldots, N \\
\text{residuals:}\quad & \mathbf{R}_i\left(\mathbf{x}_0, \mathbf{x}_i, \mathbf{y}_i, \hat{\mathbf{y}}_{j\neq i}\right) = 0 \\
& \qquad i = 1, 2, \ldots, N,
\end{aligned} \tag{9}$$

where $\hat{\mathbf{y}}_i$ is the copy of $\mathbf{y}_i$ and $\hat{\mathbf{y}} = [\hat{\mathbf{y}}_1^{\mathbf{T}}, \ldots, \hat{\mathbf{y}}_N^{\mathbf{T}}]^{\mathbf{T}}$.

4.3. Simultaneous Analysis and Design

4.3. Simultaneous Analysis and Design. The Simultaneous Analysis and Design architecture treats the entire multidisciplinary design problem as a single large optimization problem, which is accomplished by converting the discipline analysis equations into equality constraints and adding the coupling variables and state variables into optimization variables. In this case, the task of the optimizer is to solve the discipline analysis and the optimization problem simultaneously. This approach is also referred to as All-at-Once (AAO) in some literature.

The primary advantage of SAND architecture is that the costly solving of discipline analysis is eliminated at each optimization iteration, which can make the optimal solution of MDO problem found at a lower expense by letting the optimizer explore regions that are infeasible with respect to the discipline analysis equations. In addition, the parallel computing can be performed not only between different disciplines, but also within each disciplines. This character makes it possible to further improve the optimization efficiency by making full use of the parallelism of MDO problem.

In spite of having the benefits above, the SAND architecture has some major weaknesses in practice. The first one is that the discipline feasibility is not guaranteed until the last iteration of the optimization process. Therefore, if the optimization is interrupted, the resulting design may not be physically feasible. The second one is that SAND architecture does require all coupling variables, state variables, and discipline analysis equations. But in practical engineering problem, the discipline analysis is mostly computed by specialized software which often operates in a black-box form, so the SAND architecture cannot be implemented in this case. The last one is that the SAND architecture can greatly increase the dimensionality of the optimization problem by converting the discipline analysis equations into equality constraints and adding the coupling variables and state variables into optimization variables. Some gradient-free optimization methods might be prohibitive because the iterations of them show exponential growth with the optimization variables increasing.

With regard to the variable complexity problem, the specific formulation of SAND architecture can be stated as follows:

$$\begin{aligned}
\text{minimize:}\quad & f(\mathbf{x}_0, \mathbf{x}, \mathbf{y}) \\
\text{design variable:}\quad & \mathbf{x}_0, \mathbf{x}, \mathbf{y} \\
\text{constraints:}\quad & \mathbf{c}_0(\mathbf{x}_0, \mathbf{x}, \mathbf{y}) \le 0 \\
& \mathbf{c}_i(\mathbf{x}_0, \mathbf{x}_i, \mathbf{y}_i) \le 0 \quad i = 1, 2, \ldots, N \\
& \mathbf{R}_i\left(\mathbf{x}_0, \mathbf{x}_i, \mathbf{y}_i, \mathbf{y}_{j\neq i}\right) = 0 \\
& \qquad i = 1, 2, \ldots, N.
\end{aligned} \tag{10}$$

4.4. Collaborative Optimization. Collaborative Optimization is a bilevel distributed architecture and was first proposed by Braun and Kroo (1997). CO architecture is designed to provide the discipline autonomy by decomposing the optimization problem into a master problem and a number of independent subproblems. The goal of each subproblem is to minimize the corresponding compatibility function which is defined by the discrepancy between the system level variables and the values obtained by optimizing the subproblem while satisfying the local constraints. The variables in each subproblem consist of local variables and copies of global variables. Any nonlocal coupling variables needed by the discipline analysis are passed from the master problem and kept constant throughout the entire subproblem optimization. The aim of the master problem is to minimize the design objective function while making the compatibility function of each subproblem zero. The variables in master problem are composed of global variables, copies of local variables, and copies of coupling variables that explicitly affect the design objective function.

As a bilevel architecture, there are many advantages about CO. The main one is that it accomplishes the discipline autonomy. This allows the discipline design groups just to be control of and responsible for their own discipline. The people are given the freedom to solve each subproblem optimization using any efficient methods based on the characters of each discipline. In addition, the autonomy also allows each subproblem optimization to be carried out in parallel, which can greatly improve the optimization efficiency.

But CO architecture has major drawback in its mathematical formulation which results in its poor performance in practice. DeMiguel and Murray (2000) [28] and Alexandrov and Lewis (2002) [2] show that there is a breakdown in the constraint qualification of the KKT optimality conditions in the master problem of CO architecture, which slows down the convergence rate for most gradient-based optimization methods and may prevent the CO architecture from converging in the worst case. A linear and a quadratic programming problem given by Alexandrow and lewis (2000) demonstrates that the CO architecture failed to converge to optimal solution even from starting points close to the optimal solution.

With regard to the variable complexity problem, the specific formulation of CO architecture can be stated as follows.

The master problem is as follows:

$$\begin{aligned}
\text{minimize:}\quad & f\left(\mathbf{x}_0, \widehat{\mathbf{x}}, \widehat{\mathbf{y}}\right) \\
\text{design variable:}\quad & \mathbf{x}_0, \widehat{\mathbf{x}}, \widehat{\mathbf{y}} \\
\text{constraints:}\quad & \mathbf{c}_0\left(\mathbf{x}_0, \widehat{\mathbf{x}}, \widehat{\mathbf{y}}\right) \le 0 \\
& J_i^* \\
& = \left\|\mathbf{x}_0 - \widehat{\mathbf{p}}_i\right\|_2^2 + \left\|\widehat{\mathbf{x}}_i - \mathbf{x}_i\right\|_2^2 \\
& \quad + \left\|\widehat{\mathbf{y}}_i - \mathbf{y}_i\right\|_2^2 = 0 \quad i = 1, \ldots, N,
\end{aligned} \tag{11}$$

where $\widehat{\mathbf{x}} = [\widehat{\mathbf{x}}_1^{\mathbf{T}}, \ldots, \widehat{\mathbf{x}}_N^{\mathbf{T}}]^{\mathbf{T}}$ and $\widehat{\mathbf{x}}_i$ is the copy of $\mathbf{x}_i$, $\widehat{\mathbf{y}} = [\widehat{\mathbf{y}}_1^{\mathbf{T}}, \ldots, \widehat{\mathbf{y}}_N^{\mathbf{T}}]^{\mathbf{T}}$ and $\widehat{\mathbf{y}}_i$ is the copy of $\mathbf{y}_i$, and J_i^* is obtained by solving the following optimization subproblem of discipline i

$$\begin{aligned}
\text{minimize:}\quad & J_i \\
& = \left\|\widehat{\mathbf{p}}_i - \mathbf{x}_0\right\|_2^2 + \left\|\mathbf{x}_i - \widehat{\mathbf{x}}_i\right\|_2^2 \\
& \quad + \left\|\mathbf{y}_i - \widehat{\mathbf{y}}_i\right\|_2^2 \\
\text{design variable:}\quad & \widehat{\mathbf{p}}_i, \mathbf{x}_i \\
\text{constraints:}\quad & \mathbf{c}_i\left(\widehat{\mathbf{p}}_i, \mathbf{x}_i, \mathbf{y}_i\right) \le 0 \\
\text{residual:}\quad & \mathbf{R}_i\left(\widehat{\mathbf{p}}_i, \mathbf{x}_i, \mathbf{y}_i, \widehat{\mathbf{y}}_{j\neq i}\right) = 0,
\end{aligned} \tag{12}$$

where $\widehat{\mathbf{p}}_i$ is the copy of $\mathbf{x}_0$ in discipline i and $\widehat{\mathbf{y}}_{j\neq i}$ are the nonlocal coupling variables which are passed from the master problem.

5. Performance Evaluation

The main work of this section is to evaluate how the performance of MDO architectures in Section 4 changes when the changeable parameters in Section 3 are varied. In order to achieve this purpose, some implementation tools, which are used to solve problems ((8)–(12)), are firstly introduced to describe the test environment and then a comprehensive performance analysis is carried out to give an indication of each architecture's performance.

5.1. Implementation Tools. Python [29–31] is selected as the programming language to solve problems ((8)–(12)) because it has the following advantages in the scientific computing area. Firstly, Python is an interpreted language and can be run in interactive mode, which makes it easy for learning and debugging. Secondly, Python supports object-oriented programming and the raising and catching of exceptions defined by user, which make it convenient for scripting and error handling. Thirdly, Python has many useful packages, such as Numpy, Scipy, and pyMPI, which greatly reduce the difficulty in realizing optimization, parallel computing, and so on.

Python Optimization Framework (pyOpt) [32] is a Python-based package for formulating and solving nonlinear constrained optimization problems, which was inspired by a Python wrapper script for the Sparse Nonlinear Optimizer (SNOPT) created by Professor Joaquim Martins in 2000. Different types of open-source and licensed optimizers that solve the general nonlinear optimization problem have been integrated into the package, such as SNOPT, SLSQP, and ALPSO. Because pyOpt can solve the nonlinear constrained optimization problems in an efficient, reusable, and portable manner, it is introduced to provide all the required optimizers for implementing different MDO architectures.

SciPy [33–35] is a collection of mathematical algorithms and convenience functions built on the Numpy extension of Python. It adds significant power to the interactive Python session by providing the user with high-level commands and classes for manipulating and visualizing data. Specifically, the scipy.optimize package provides several commonly used root finding algorithms, such as Broyden, Exicitingmixing, and Krylov. In this paper, these root finding algorithms are used to solve the MDA module in MDF architectures.

5.2. Performance Analysis. In order to evaluate how the performance of MDO architectures changes with the complexity of MDO problem varied, four investigations are performed for the variable complexity problem. In the first one, the number of disciplines is varied while keeping all the other changeable parameters of the variable complexity problem unchanged. In the second one, the number of global variables is varied and similarly keeping the other changeable parameters constant. In the third investigation, only the number of local variables included in each discipline is varied. In the

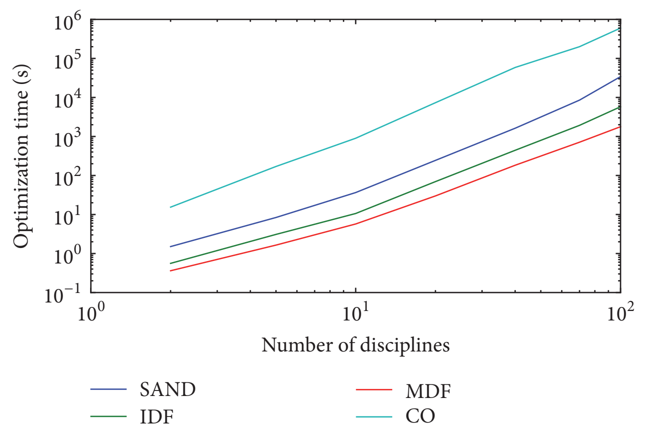


FIGURE 3: The first investigation.

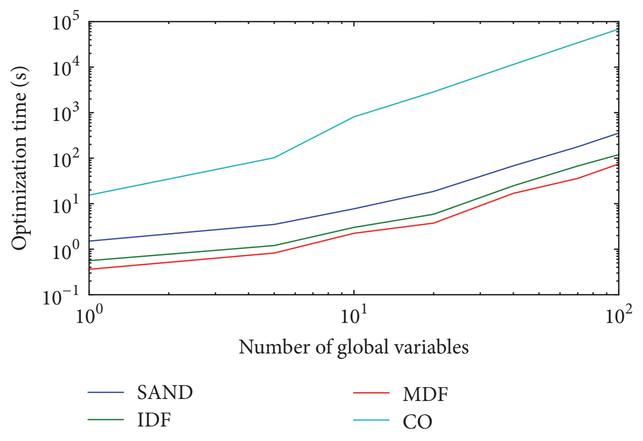


FIGURE 4: The second investigation.

fourth one, the number of coupling variables included in each discipline is varied. In order to specify the four investigations above, an initial variable complexity problem is given by setting $(N, n_{x_0}, n_{x_i}, n_{y_i}, n_{c_i}, n_{c_0})$ at $(2, 1, 2, 2, 2, 1)$ and all the investigations are carried out on it. What is more, for every specific variable complexity problem, the coefficient matrices in Section 3.1, $\mathbf{B}_{ii}$, $\mathbf{B}_{ij}$, $\mathbf{C}_i$, and $\mathbf{D}_i$, are randomly generated between −5 and 5 while keeping $\mathbf{B}_{ii}$ and the combined matrix $\mathbf{B}$ nonsingular. The coefficient vectors in Section 3.2, $\mathbf{E}_{ij}$, $\mathbf{F}_{ij}$, $\mathbf{G}_{ij}$, $\mathbf{H}_i$, $\mathbf{I}_{ij}$, and $\mathbf{J}_{ij}$, are randomly generated between −5 and 5. The positive scalars in Section 3.2, r_{ij} and s_i, are randomly generated between 1 and 5.

When implementing the four MDO architectures in Section 4 for the four investigations above, the SLSQP in pyOpt is selected as the optimizer wherever there is a need for solving optimization problem and all of its parameters are left at the default values resulting in the convergence tolerance on the objective and constraints having to be satisfied with 10^{-6}. The Broyden method in scipy.optimize package is chosen to solve the MDA module in MDF architecture and all its parameters are similarly left at the default values. In addition, the processor of the computer used to perform the optimization is Intel(R)Core(TM) 2 Duo CPU E7500 @ 2.93 GHz × 2 and the memory is 1.9 GiB.

Due to the different structures of each MDO architecture in Section 4, the total optimization time is the most meaningful and effective quantity to compare their performances. Figures 3–6 show the results of the four MDO architectures associated with the four investigations, respectively. In the first investigation, the number of disciplines is specifically varied from 1 to 100. In the second investigation, the number of global variables is varied from 1 to 100. In the third investigation, the number of local variables included in each discipline is varied from 1 to 100. In the fourth investigation, the number of coupling variables outputted from each discipline is similarly varied from 1 to 100.

From these figures, we can see that CO is always the most expensive architecture and it spends significantly more time than the other three architectures. SAND and IDF are the second and third expensive architectures, respectively. MDF is typically the most computationally efficient architecture.

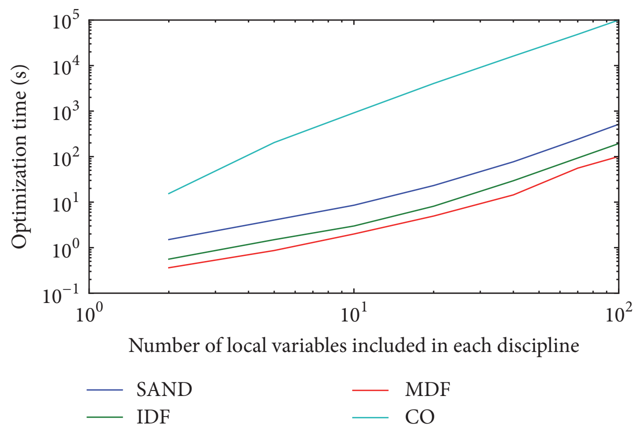


FIGURE 5: The third investigation.

In Figures 4 and 5, with the number of global variables and the number of local variables included in each discipline increasing, the gaps between SAND, IDF, and MDF architectures remain about the same, which means the number of global variables and local variables almost has the same effect on the SAND, IDF, and MDF architectures.

In Figures 3 and 6, with the number of disciplines and the number of coupling variables included in each discipline increasing, the gaps between SAND, IDF, and MDF architectures are becoming gradually larger. It means SAND architecture performs gradually worse than the IDF and MDF architectures and IDF architecture performs gradually worse than MDF architecture with the number of disciplines and coupling variables increasing, respectively. In addition, the number of disciplines has less effect than the number of coupling variables.

For CO architecture, the number of disciplines and coupling variables has the same effect on it and SAND architectures based on Figures 3 and 6. However, its performance becomes gradually worse than SAND architecture with the

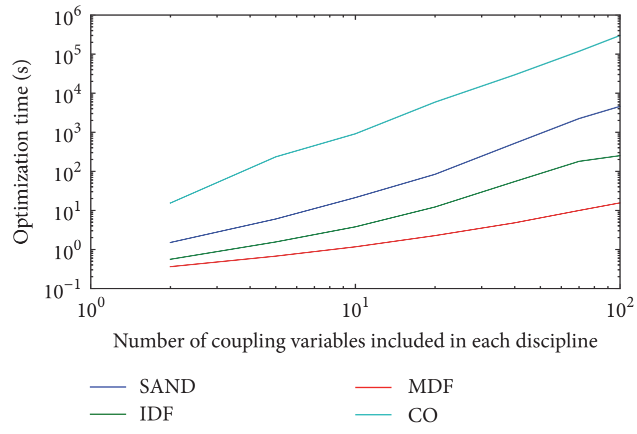


Figure 6: The fourth investigation.

number of global variables and local variables increasing based on Figures 4 and 5.

6. Conclusions

In this paper, two works have mainly been done to evaluate how the performance of MDO architectures changes with the complexity of MDO varied. Firstly, we present a variable complexity problem which is a nonseparable nonlinear problem that allows people to specify its complexity as needed. Secondly, we evaluate the performance of different MDO architectures through implementing four MDO architectures to solve the variable complexity problem under four investigations.

For the first work, the presented variable complexity problem is composed of the design variables, local constraints, global constraints, and nonseparable objective functions and allows people to obtain the MDO problem with arbitrary complexity by varying its complexity. It is more general to represent the practical engineering problems and make it possible to examine the effects of problem complexity on different MDO architecture.

For the second work, the architectures of MDF, IDF, SAND, and CO are implemented to solve the variable complexity problem through programming in Python language in the same computer environment and four investigations are performed for the variable complexity problem by varying its variable parameters which are, respectively, the number of disciplines, the number of global variables, the number of local variables, and the number of coupling variables. The results supply a reference for the selection of MDO architectures in solving practical engineering problems. Though only four common MDO architectures and four investigations are included, they can be regarded as templates for future researchers to benchmark other MDO architectures under other investigations.

Competing Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This research was supported by the National Natural Science Foundation of China under Grant no. 51375389. The authors would like to thank Joaquim R. R. A. Martins for providing valuable suggestions to this research.

References

[1] E. J. Cramer, J. E. Jr. Dennis, P. D. Frank, R. M. Lewis, and G. R. Shubin, "Problem formulation for multidisciplinary optimization," *SIAM Journal on Optimization*, vol. 4, no. 4, pp. 754–776, 1994.

[2] N. M. Alexandrov and R. M. Lewis, "Analytical and computational aspects of collaborative optimization for multidisciplinary design," *AIAA Journal*, vol. 40, no. 2, pp. 301–309, 2002.

[3] R. J. Balling and J. Sobieszczanski-Sobieski, "Optimization of coupled systems: a critical overview of approaches," *AIAA Journal*, vol. 34, no. 1, pp. 6–17, 1996.

[4] I. M. Kroo, "MDO for large-scale design," in *Multidisciplinary Design Optimization: State-of-the-Art*, N. M. Alexandrov and M. Y. Hussaini, Eds., pp. 22–44, SIAM, 1997.

[5] R. T. Haftka, "Simultaneous analysis and design," *AIAA Journal*, vol. 23, no. 7, pp. 1099–1103, 1985.

[6] C. L. Bloebaum, P. Hajela, and J. Sobieszczanski-Sobieski, "Non-hierarchic system decomposition in structural optimization," *Engineering Optimization*, vol. 19, no. 3, pp. 171–186, 1992.

[7] J. E. Renaud and G. A. Gabriele, "Approximation in nonhierarchic system optimization," *AIAA Journal*, vol. 32, no. 1, pp. 198–205, 1994.

[8] R. D. Braun, *Collaborative optimization: an architecture for large-scale distributed design [Ph.D. thesis]*, Stanford University, Stanford, Calif, USA, 1996.

[9] R. Braun, P. Gage, I. Kroo, and I. Sobieski, "Implementation and performance issues in collaborative optimization," in *Proceedings of the 6th AIAA/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, pp. 295–305, AIAA, Bellevue, Wash, USA, September 1996.

[10] X. Li, W. Li, and C. Liu, "Geometric analysis of collaborative optimization," *Structural and Multidisciplinary Optimization*, vol. 35, no. 4, pp. 301–313, 2008.

[11] J. Sobieszczanski-Sobieski, J. S. Agte, and R. R. Sandusky Jr., "Bilevel integrated system synthesis," *AIAA Journal*, vol. 38, no. 1, pp. 164–172, 2000.

[12] J. Sobieszczanski-Sobieski, T. D. Altus, M. Phillips, and R. Sandusky, "Bilevel integrated system synthesis for concurrent and distributed processing," *AIAA Journal*, vol. 41, no. 10, pp. 1996–2003, 2003.

[13] J. Sobieszczanski-Sobieski, "Integrated system-of-systems synthesis," *AIAA Journal*, vol. 46, no. 5, pp. 1072–1080, 2008.

[14] H. M. Kim, *Target cascading in optimal system design [Ph.D. thesis]*, The University of Michigan, Ann Arbor, Mich, USA, 2001.

[15] H. M. Kim, N. F. Michelena, P. Y. Papalambros, and T. Jiang, "Target cascading in optimal system design," *Journal of Mechanical Design, Transactions of the ASME*, vol. 125, no. 3, pp. 474–480, 2003.

[16] M.-K. Shin and G.-J. Park, "Multidisciplinary design optimization based on independent sub-spaces," *International Journal for Numerical Methods in Engineering*, vol. 64, no. 5, pp. 599–617, 2005.

[17] S. Kodiyalam, "Evaluation of Methods for Multidisciplinary Design Optimization (MDO), phase 1," NASA Report CR-1998-208716, 1998.

[18] S. Kodiyalam and C. Yuan, "Evaluation of methods for Multidisciplinary Design Optimization (MDO), part 2," NASA Report CR-2000-210313, 2000.

[19] K. F. Hulme and C. L. Bloebaum, "Simulation-based comparison of Multidisciplinary Design Optimization solution strategies using CASCADE," *Structural and Multidisciplinary Optimization*, vol. 19, no. 1, pp. 17–35, 2000.

[20] N. P. Tedford and J. R. R. A. Martins, "Benchmarking multidisciplinary design optimization algorithms," *Optimization and Engineering*, vol. 11, no. 1, pp. 159–183, 2010.

[21] S. I. Yi, J. K. Shin, and G. J. Park, "Comparison of MDO methods with mathematical examples," *Structural and Multidisciplinary Optimization*, vol. 35, no. 5, pp. 391–402, 2008.

[22] C. J. Marriage and J. R. R. A. Martins, "Reconfigurable semi-analytic sensitivity methods and MDO architectures within the πMDO framework," in *Proceedings of the 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference (MAO 08)*, AIAA, Victoria, Canada, September 2008.

[23] J. R. R. A. Martins and A. B. Lambe, "Multidisciplinary design optimization: a survey of architectures," *AIAA Journal*, vol. 51, no. 9, pp. 2049–2075, 2013.

[24] J. R. R. A. Martins, C. Marriage, and N. Tedford, "PyMDO: an object-oriented framework for multidisciplinary design optimization," *ACM Transactions on Mathematical Software*, vol. 36, no. 4, article no. 20, 2009.

[25] A. B. Lambe and J. R. R. A. Martins, "Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes," *Structural and Multidisciplinary Optimization*, vol. 46, no. 2, pp. 273–284, 2012.

[26] V. DeMiguel and W. Murray, "A local convergence analysis of bilevel decomposition algorithms," *Optimization and Engineering*, vol. 7, no. 2, pp. 99–133, 2006.

[27] R. T. Haftka and L. T. Watson, "Multidisciplinary design optimization with quasiseparable subsystems," *Optimization and Engineering*, vol. 6, no. 1, pp. 9–20, 2005.

[28] A.-V. DeMiguel and W. Murray, "An analysis of collaborative optimization methods," in *Proceedings of the 8th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, AIAA 2000-4720, Long Beach, Calif, USA, September 2000.

[29] M. Lutz, *Programming Python*, O'Reilly & Associates, Cambridge, Mass, USA, 1996.

[30] M. Lutz and D. Ascher, *Learning Python*, O'Reilly & Associates, Cambridge, Mass, USA, 1999.

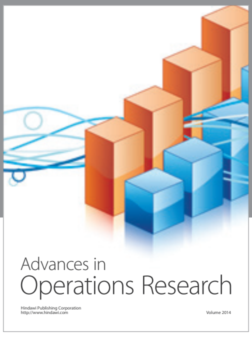
[31] D. M. Beazley, *Python Essential Reference*, Sams, 2006.

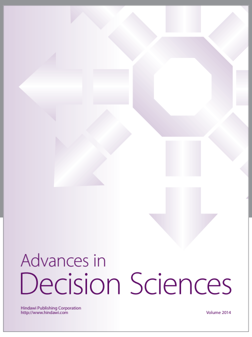
[32] R. E. Perez, P. W. Jansen, and J. R. R. A. Martins, "pyOpt: a Python-based object-oriented framework for nonlinear constrained optimization," *Structural and Multidisciplinary Optimization*, vol. 45, no. 1, pp. 101–118, 2012.

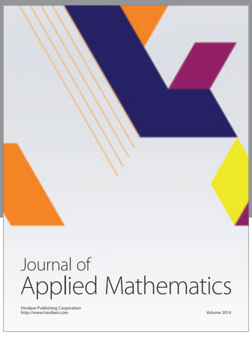
[33] E. Jones, T. Oliphant, P. Peterson et al., *SciPy: Open Source Scientific Tools for Python*, 2015.

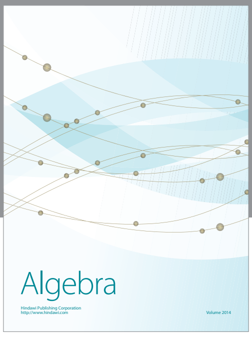
[34] H. P. Langtangen, *Python Scripting for Computational Science*, Springer, 2007.

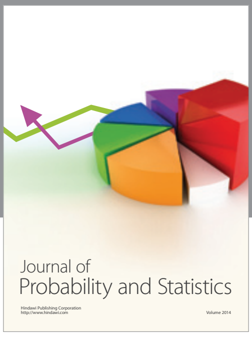
[35] H. P. Langtangen, *A Primer on Scientific Computing with Python*, Springer, 2009.

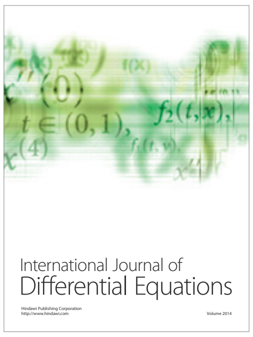












Hindawi

Submit your manuscripts at
https://www.hindawi.com