Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 910209, 15 pages
http://dx.doi.org/10.1155/2013/910209

Research Article

Characterization of the Transient Response of Coupled Optimization in Multidisciplinary Design

Erich Devendorf[1] and Kemper Lewis[2]

[1] *Air Force Research Laboratory, Information Directorate, Rome, NY 13044, USA*
[2] *Department of Mechanical and Aerospace Engineering, University at Buffalo—SUNY, Buffalo, NY 14260, USA*

Correspondence should be addressed to Kemper Lewis; kelewis@buffalo.edu

Received 11 February 2013; Accepted 7 May 2013

Academic Editor: Dan Zhang

Time is an asset of critical importance in a multidisciplinary design process and it is desirable to reduce the amount of time spent designing products and systems. Design is an iterative activity and designers consume a significant portion of the product development process negotiating a mutually acceptable solution. The amount of time necessary to complete a design depends on the number and duration of design iterations. This paper focuses on accurately characterizing the number of iterations required for designers to converge to an equilibrium solution in distributed design processes. In distributed design, systems are decomposed into smaller, coupled design problems where individual designers have control over local design decisions and seek to achieve their own individual objectives. These smaller coupled design optimization problems can be modeled using coupled games and the number of iterations required to reach equilibrium solutions varies based on initial conditions and process architecture. In this paper, we leverage concepts from game theory, classical controls, and discrete systems theory to evaluate and approximate process architectures without carrying out any solution iterations. As a result, we develop an analogy between discrete decisions and a continuous time representation that we analyze using control theoretic techniques.

1. Introduction

The design of complex systems presents both technical and logistical challenges to organizations. In some cases, organizations do not have the requisite technical expertise to overcome design challenges or meet design requirements. However, even when an organization possesses sufficient technical expertise, there are many cases where inadequate logistical control can inhibit the application of this design expertise in a meaningful manner. The logistical challenges facing organizations are becoming increasingly significant as the size and sophistication of modern engineered products increase. This growth in sophistication and size leads to the decomposition of design systems as a means to reduce system complexity. Although system decomposition reduces the technical complexity of each subsystem's individual design problems by reducing their size, it introduces significant logistical challenges. Also, it often requires significant approximation of nonlocal behavior, interfaces, and solutions. These decomposed systems are often large and multidisciplinary in nature, with diverse subsystems governed by unique objectives and constraints.

To effectively manage decomposed design problems, it is important to understand how their constituent subsystems interact with one another. One approach to capture and analyze these interactions is to model the collection of subsystems as a distributed design process. In distributed design, understanding subsystem interactions is fundamental to predicting the system equilibrium properties and transient response. To predict these systems' equilibrium properties, which include location and stability, a game theoretic approach was introduced in [1]. This analysis was further supported using an analogy to the cobweb model and proven using mathematical induction in [2]. For large systems, a discrete time linear system approach was used in [3] and was generalized to handle different process configurations in [4, 5]. In this paper, the linear system model developed in [6] is used as the basis for the approximation and analysis of the transient response of distributed design processes.

Transient response refers to the dynamic behavior of a distributed design system, beginning with the first design iteration and ending when the subsystems reach equilibrium. The transient response possesses two properties: (1) the convergence shape and (2) the convergence time [7]. The convergence shape depends on the properties of the distributed design process and could be sinusoidal, exponential, or some combination of responses. Examining the convergence shape is a topic of future work. This paper focuses on convergence time and develops an approach to estimate an upper bound for the number of design iterations required for a distributed design process to converge to an equilibrium solution.

Other work has been done to characterize and approximate the solutions to certain types of two-person games including bimatrix games characterized by nonsymmetrical fuzzy approximations [8–10], multiplayer congestion games [11, 12], and discontinuous games [13]. However, in this work, we do not focus on the approximation of the solution to particular games, but rather the convergence process to arrive at such a solution, if it exists. We focus on two-player continuous games, as they provide a characterization of coupled optimization problems frequently occurring in complex design problems.

The intellectual merit of this work is based on its insight into the dynamics of distributed design processes. The linear system model used in previous work to assess stability is refined to analyze both system stability and transient response. Through the refinement of this model, it is demonstrated that the uniquely discontinuous nature of any decentralized decision network must be considered when they are modeled using systems theory. Furthermore, several control theory principles are shown to be valid when analyzing distributed design processes.

As an initial context for this investigation, we focus on distributed design problems with unconstrained quadratic objective functions. These types of problems are well modeled using linear system theory, but it is recognized that many design problems cannot easily be converted into unconstrained problems and/or do not have quadratic objective functions. While techniques in metamodeling exist to represent higher-order systems using quadratic response surfaces [14], it is desirable to analyze systems in their native mathematical form. We examine quadratic systems in this paper in order to fully understand their fundamental principles before applying the concepts to higher-order systems. As a result, this work represents a critical first step towards leveraging linear system analysis techniques to understand and analyze the behavior of distributed design systems.

This work has broader impacts in any scenario where the decentralization of decisions is present. These scenarios can range from product design to coordinating disaster relief. It provides a deeper understanding of the decision-making process and enables a greater level of process control. This enables decision makers to reach iterative solutions quicker and to set realistic deadlines or timetables. It represents a unique and effective approach to find an upper bound for the time it takes a distributed design process to converge to a stable equilibrium. Furthermore, it provides insight into how to best configure these processes to minimize the maximum number of iterations required to reach equilibrium.

The following sections provide background into concepts foundational to the examination of decentralized decision networks. In Section 2 an overview of stability and convergence concerns in multidisciplinary optimization is presented along with the basic tenets of distributed design processes. Based on the tenets outlined in Section 2, a linear systems approach to examine the transient response of distributed design systems is examined in Section 3. Finally, these results are summarized and areas of future work are identified in Section 4.

2. Materials and Methods

MDO problems have two classifications based on the process used to complete the design [15]. From a structural perspective, the simplest method to solve an MDO problem is to apply an all-at-once approach. In an all-at-once approach designers from different disciplines work as system analyzers to determine objective function and constraint values for a single optimization problem [16, 17]. There are significant advantages to organizing an MDO system to solve a single centralized optimization problem. In a centralized problem, all designers are working towards the exact same objective and information about the entire system is available to designers.

Although these advantages make centralization attractive, it is almost impossible to centralize the design of complex systems. Three major design approaches, Systematic Design, Total Design, and Axiomatic Design, are all structured with the understanding that some level of system decomposition is often desirable to speed development times through parallelization, to reduce system complexity, and to reduce computational time [18–20]. Recognizing this, it is likely that decomposition will remain an important and necessary aspect of product design processes in the near future.

Fortunately for designers a wide range of approaches have been developed to aid in system decomposition. The system decomposition process can be broken into two fundamental steps: (1) identify the necessary subsystems, (2) establish a framework to govern subsystem interactions. The first step in this process is not the topic of this paper, but it is by no means a trivial task. Subsystems can be created based on object decomposition, aspect decomposition, sequential decomposition, and model-based decomposition [21]. A survey of the relative merits of these subsystem creation approaches was performed by Sobieszczanski-Sobieski and Haftka [15].

The second step in system decomposition is the creation of a design framework. MDO frameworks specify the mechanics of how the design problem is solved including the subsystem objective functions, communication protocols, design variable control, and the other coordination procedures required for the subsystems to effectively iterate to a solution. There are a wide range of MDO frameworks which handle these issues differently while making certain guarantees about system convergence and the optimality of the final converged solution. These approaches include analytic

target cascading [22], concurrent subspace optimization [23, 24], bilevel integrated system synthesis [25] and collaborative optimization [26].

Each of these frameworks has its own advantages. For example, analytic target cascading guarantees that the decentralized system converges and that the converged value is globally optimal [27]. It also provides for traceability and facilitates the integration of marketing, business, and design systems [28, 29].

There are several reasons why an MDO framework may not be applied to a complex design problem. Applying a framework requires a significant level of coordination between subsystems and a high level of management expertise [15]. Furthermore, the engineering and design personnel involved must all agree to some extent to the proposed decomposition and framework. There are also some cases that do not naturally lend themselves to formal decomposition or where the parties involved cannot agree on an appropriate framework. In these cases subsystems often act exclusively in their own self interest, attempting to most effectively solve their individual optimization formulations, and communication between subsystems is dictated by the required exchange of design information. When these conditions exist, design problems can be well modeled as distributed design processes. Even when there is communication between design teams, a distributed design process can often be used to model these systems. The assumptions and mechanics governing distributed design problems are discussed in Section 2.1.

2.1. Conditions for Distributed Design Processes. Distributed design processes are iterative and can be cooperative, non-cooperative, or a hybrid of the two. This work examines noncooperative distributed design processes where design subsystems often have conflicting objectives or organizational barriers that prevent them from fully cooperating synchronously. Even when subsystems share the same aggregate design goals, there are cases where they will compete with one another for design resources [30] due to the underlying scarcity of such resources.

In addition to being categorized based on cooperation, distributed design systems can be broken into hierarchical and non-hierarchical realizations. In this work, it is assumed that a design process can be adequately modeled as a non-hierarchical process. This assumption is not overly restrictive in scenarios where distributed design processes are applied, since these scenarios typically lack a strong system level presence. We elaborate the non-cooperative protocol used in this paper in Section 2.2 and discuss criteria for equilibrium stability and transient response in Sections 2.3 and 2.4, respectively. The concept of solution process architecture is introduced in Section 2.5.

2.2. Criteria for Noncooperation in Repeated Games. Efforts to model and analyze subsystem interactions have led to the development of many different models for distributed design processes. One of the first models for distributed design applied mathematical notions of game theory to model design subsystems as players in a non-cooperative game. These players act independently of one another and through successive plays of the game, they eventually reach an equilibrium solution [31]. This model forms the basis for the analysis of distributed design systems in this work and is foundational to the application of other distributed design process models. These assumptions have been defined for distributed design in [2] and are as follows.

(1) Subsystems have knowledge of only their own local objectives;

(2) Subsystems act unilaterally in accordance with their own objectives;

(3) Subsystems have complete control over specific local design variables;

(4) Subsystems communicate by sharing the current value of their local design variables.

The applicability of these assumptions to decentralized design problems is discussed in various contexts in [32–35] using examples that include the design of passenger aircraft, automotive engines, semiconductor chips, and steam turbines. Distributed design problems can also emerge as iterative subproblems in a larger MDO process. For example, in [36, 37] the ordering of decomposed design systems was examined and iterative loops emerged due to subsystem coupling of concurrently executed tasks.

Subsystems in distributed design processes have their own specific objectives they are attempting to achieve. They have complete control over a set of local design variables which appear in their own objective formulation. These variables also act to couple subsystems in a manner that restricts the ability of a subsystem to independently achieve its objectives. The influence of this coupling has been investigated using network theory to model distributed design processes [38]. The examination of such coupling using game theory to define, identify, and classify system equilibrium is the focus of Section 2.3.

2.3. Equilibrium Stability for Noncooperative Processes. Determining if a design system converges to a stable solution is of critical importance to understanding the system. Convergence stability in distributed problems has been a topic of research for some time with the first work being performed by Vincent [31] for two designers, two design variables problems. Vincent introduced the game theory model for distributed design processes, which was investigated further by Lewis [39]. In Vincent's work each player alternates minimizing his or her local objective function value and communicates the associated design variables to the other player. Each step in this alternating process is a play in a sequential game. After repeated playing of the sequential game, the players either converge to a solution or diverge and continue playing indefinitely. When the players converge, they converge to a specific point called the Nash, or non-cooperative equilibrium [40].

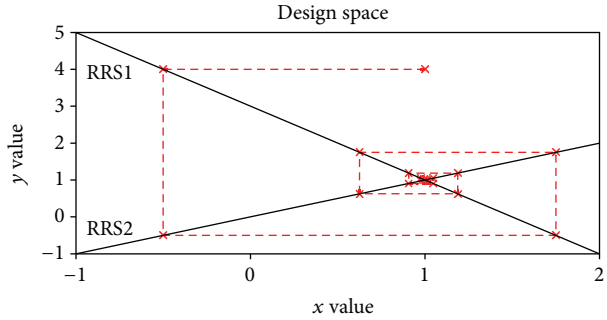

Figure 1: Two subsystems—Nash equilibrium.

Mathematically, in a two-player game a set of solutions described by the vector pair (x_1, x_2) are a Nash solution, (x_{1N}, x_{2N}), if they fulfill the requirements outlined in (1) as

$$F_1(x_{1N}, x_{2N}) = \min_{x_1} F_1(x_1, x_{2N}),$$
$$F_2(x_{1N}, x_{2N}) = \min_{x_2} F_2(x_{1N}, x_2). \quad (1)$$

In (1), F_1 and F_2 are the objective functions for player 1 and player 2 who control design variables x_1 and x_2, respectively. A solution pair (x_1, x_2) that meets the criteria in (1) is a Nash solution because the pair is a minimum for both F_1 and F_2. Although in game theory the participants in a game are called players, in engineering design they are typically called designers or subsystems. In this work, the term subsystem is used more generally, reflecting the linear system basis of the work. The relationship demonstrated in (1) can be understood qualitatively as the point at which no subsystem can unilaterally improve his or her objective function [41]. This expression identifies Nash solutions through an optimization formulation, but they can also be expressed as the intersection of two sets defined by (2) as

$$(x_{1N}, x_{2N}) \in X_{1N}(x_{2N}) \times X_{2N}(x_{1N}),$$
$$X_{1N}(x_2) = \left\{ x_{1N} \mid F_1(x_{1N}, x_2) = \min_{x_1} F_1(x_1, x_2) \right\}, \quad (2)$$
$$X_{2N}(x_1) = \left\{ x_{2N} \mid F_2(x_1, x_{2N}) = \min_{x_2} F_2(x_1, x_2) \right\}.$$

The sets X_{1N} and X_{2N} are the rational reaction sets (RRS) or best response sets [42], which embody all the possible reactions or responses a subsystem may have towards a decision made by another subsystem. While determining the RRSs is not a trivial task, methods have been developed to approximate them for large systems [43]. One of these techniques is to calculate the RRS by taking the gradient of the subsystem's objective function with respect to its local design variables. This calculation is shown in (3) for the two designer problem studied by Vincent [31] as

$$F_1 = x^2 + xy - 3x,$$
$$\frac{\delta F_1}{\delta x} = 2x + y - 3 = 0,$$
$$F_2 = 0.5y^2 - xy, \quad (3)$$
$$\frac{\delta F_2}{\delta y} = y - x = 0.$$

These RRSs are plotted with respect to the design variables, x and y, in Figure 1 along with an illustration of how the solution process converges to the Nash equilibrium at (1, 1). We have included the dotted line in Figure 1 to demonstrate the order of the decisions made by the two subsystems. The subsystems iterate sequentially and begin with the initial conditions set to (1, 4). The x's in Figure 1 shows the actual discrete decision occurring through the repeated decisions, identical to the plays of a sequential game. In Figure 1, subsystem 1 adjusts the value of x and subsystem 2 responds by adjusting y. For the decision making process in Figure 1, the design variable values converge to the Nash equilibrium, defined by the intersection of the players' RRSs. For two subsystem problems, Vincent defined stability criteria based on the subsystems' RRSs [31]. This work was extended by Chanron and Lewis to examine convergence more generally when there are more than two players controlling multiple design variables [3]. Convergence was shown to be a function of the relative slope of the designer's rational reaction sets and linear system theory was applied for large scale problems [3–6]. Work by Smith and Eppinger has demonstrated the same principle using a different set of fundamental principles [44].

In this work, distributed design processes are modeled using linear system theory, using the same approach developed by Chanron and Lewis. Similar to game theoretical models, each subsystem is assumed to solve its optimization problem at distinct and discrete instants in times that are identical for every iteration based on the process configuration. We assume that a state space model has already been developed for the distributed design systems analyzed in this paper. A detailed description of how to create these models can be found in [3]. The general formulation for a subsystem's objective function using nomenclature from linear system theory is shown in (4).

$$F_n = X^T AX + Y^T BY + X^T CY + DX + EY + F. \quad (4)$$

In this representation of the nth subsystem's quadratic objective function, F_n, X is a vector of length i which contains the i local design variables while Y is a vector of length j which contains j nonlocal design variables. The coefficients associated with the second-order elements of F_n for the local design variables are contained in the diagonal $i \times i$ matrix A while the coefficients associated with the non-local design variables are contained in the $j \times j$ matrix B. In this representation the A matrix is formulated as a diagonal to decouple the subsystem's local design variables from one another. This guarantees that each design variable value can be determined independently and a specific RRS can be formulated for each design variable. When these variables are coupled, the design system can still be represented using the form in (4). However, to do so, a change in variables must be performed to decouple the values from one another. The representation in (4) is examined in more depth in [3].

Although the local design variables must be decoupled, it is acceptable for the local and non-local design variables to be coupled together through the coefficients in the $i \times j$ C matrix. The remaining two vectors in (4) capture the linear elements of the system for the local and non-local

design variables and have length i and j, respectively. The term F is simply a scalar and does not play a significant role when analyzing the system stability or transient response. The important elements in (4) emerge when the gradient is taken with respect to the local design variables. Setting this gradient equal to zero results in i decoupled equations that represent the subsystem's RRS. After the RRSs are found for each subsystem, there are m equations, where m is the total number of design variables controlled by subsystems. The RRS is shown in vector form in (5) as

$$\frac{\delta F_n}{\delta X} = 2AX + CY + D = 0. \tag{5}$$

Equation (5) specifies how this subsystems will respond and suggests the system's overall transient response is related to the matrices A, C, and D for each subsystem. Using these matrices, Chanron developed the discrete state-space-based representation to model the design systems using the update relationship and stability criteria in (6)–(9) as

$$X_s^{k+1} = \Phi X_s^k + \Gamma, \tag{6}$$

$$\Phi = -\frac{1}{2}\begin{bmatrix} A_1^{-1} & & & 0 \\ & A_2^{-1} & & \\ & & \ddots & \\ 0 & & & A_m^{-1} \end{bmatrix}\begin{bmatrix} 0 & C_{12}^T & \cdots & C_{1m}^T \\ C_{21}^T & 0 & & \vdots \\ \vdots & & \ddots & \\ C_{m1}^T & \cdots & & 0 \end{bmatrix}, \tag{7}$$

$$\Gamma = -\frac{1}{2}\begin{bmatrix} A_1^{-1}D_1^T \\ \vdots \\ A_m^{-1}D_m^T \end{bmatrix}. \tag{8}$$

In (6), the subscript s denotes that X_s^k is a vector of all the system design variables and the superscript denotes the iteration number which is consistent with linear system theory. Since (6) describes the relationship between the subsystems, X_s^{k+1} is length m containing all the design variables controlled by the subsystems. The design variable values at the $(k+1)$th iteration are a function of the previous design variables at the kth iteration; they are expressed as X_s^k multiplied by a matrix Φ plus a constant Γ. The derivations for Φ and Γ can be found in [3] and are summarized in (7) and (8). Equation (6) was generalized in [5] to be applicable to scenarios where decisions are made asynchronously.

The matrix Φ captures design variable interactions between quadratic elements found in the A and C matrices while the vector Γ captures interactions between quadratic and linear elements found in the A and D matrices, respectively. To populate Φ and Γ, the appropriate A, C, and D matrices must be used and can be determined by examining which subsystem controls the design variable associated with the row being populated. The resulting dimensions for Φ and Γ are $m \times m$ and $m \times 1$, respectively. When examining system stability, only the Φ matrix needs to be considered, and if the condition described by (9) is met then the system is stable

$$r_\sigma(\Phi) < 1. \tag{9}$$

In (9), $r_\sigma(\Phi)$ is the spectral radius, or magnitude of the largest eigenvalue of the matrix Φ [45]. The relationship in (9) specifies that for stable systems, Φ must have a spectral radius less than 1. This is the same stability criteria used for the closed loop state space representations of discrete control systems [46]. In addition to the examination of linear system stability, a case for nonlinear RRSs has also been investigated in [47] using similar criteria. Another extension of this convergence work was performed by Gurnani and Lewis who demonstrated that the introduction of "mistakes" into the design process could cause some inherently unstable problems to converge to a solution [48].

Analyzing system stability is the first step to characterizing transient response. In this paper, we examine configurations that (9) identifies as stable and develop an approach to differentiate between them based on their convergence time. In the next section, examining the transient response of distributed design systems is discussed and the idea of solution process architecture is introduced as a key factor determining the transient response of a distributed design system.

2.4. Transient Response of Distributed Design Systems. The transient response of a distributed design process has two principle aspects. The first is the shape of the transient response. This shape depends on the eigenvalues of the system being analyzed and could be sinusoidal, exponential, or some combination of responses. An example transient response shape is shown in Figure 2, which plots the value of design variables x and y from (3).

Since design decisions occur at a specific instant in time, the design variable plots in Figure 2 are staircase plots representing discrete design variable values. Figure 2(a) tracks the value of x while Figure 2(b) tracks y. In this case, both variables exhibit a decaying sinusoidal response as they approach their equilibrium value at $(1, 1)$ from a starting location of $(1, 2)$. Identifying the shape of the convergence curve for a distributed design process is an important area of future research, but this work focuses on examining the second aspect of transient response, convergence time.

The convergence time in this work is measured by the number of iterations required for the subsystem to reach an equilibrium solution. We use iterations to evaluate convergence time because they are a dimensionless characterization of the system that can be easily translated to the time domain by either mapping estimated task times directly or by leveraging the work transformation matrix approach used in [44]. Although a significant amount of rigor has been brought to the analysis of the stability characteristics of distributed design processes, the convergence time of these processes or of MDO processes in general has focused more on practical implementation than investigating controlling features. Techniques like the critical path method [49] and project evaluation and review technique [50] are the foundational approaches used in network-based project planning. The techniques for network-based project planning provide approaches to organize and execute the design tasks inherent to MDO processes. In the context of distributed design processes, there are tools to specify the ordering or

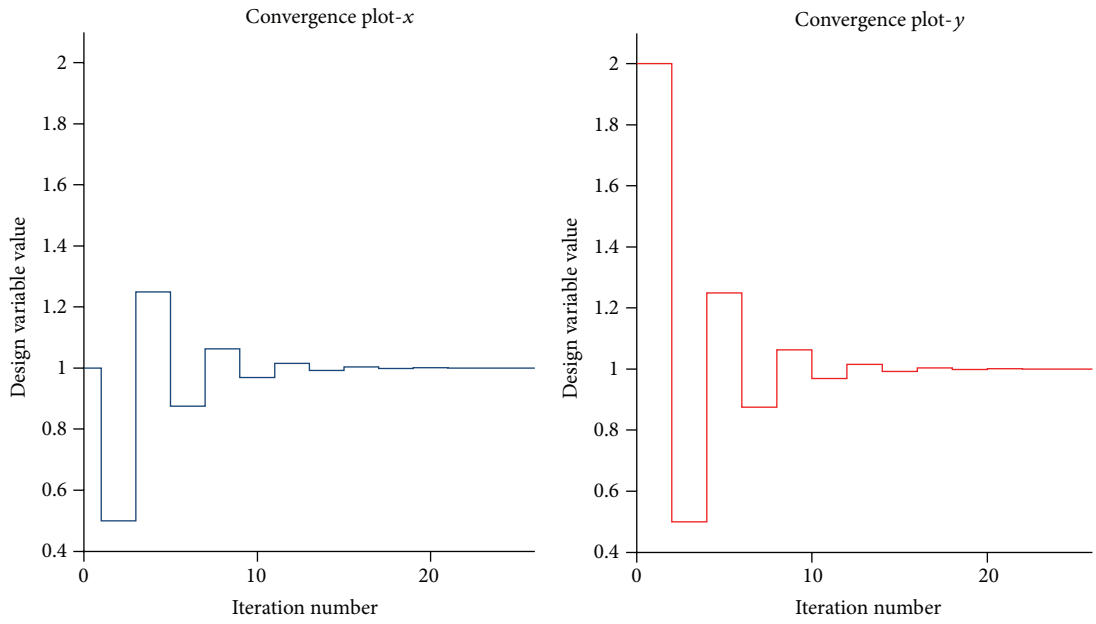

Figure 2: Two design variable convergence plot.

organization of the process which is discussed in Section 2.5. These techniques have been refined to introduce technology like Monte Carlo simulation in [51] to account for random task duration, graph theory in [52] to enable probabilistic organization, and feedback and precedence relationships in [53] to account for required work flow.

More recent formulations have leveraged advances in computing power to prescribe the fastest converging process organization. One of these techniques is an extension of the Design Managers Aid for Intelligent Decomposition (DeMAID) method and is used to reduce the time required for designers to converge to a final solution. In this extension, Rogers utilizes the global sensitivity equations [54] with a weighting scheme to predict an optimal ordering of designers in iterative design loops [55]. The approach taken by Rogers succeeded in reducing the overall design time required for iterative loops in DeMAID when designers are ordered sequentially. This ordering was partially based on an analysis of design structure matrices (DSMs), which were developed in [56] as a means to organize and visualize the coupling between design tasks.

DSMs were also used to minimize the number of feedback loops for sequential processes in [57]. An approach using DSMs to represent the probability of task repetition and durations with Markov chains was presented in [58] and a case was made for estimating the stability and convergence rate of concurrent tasks in [44]. In this work, a DSM-based transformation matrix was used to link design tasks based on the amount of rework tasks generated for one another. An eigenvalue analysis was used to determine the strength of these links and the task coupling. While this approach examines the basic mechanics of the distributed system, it does not account for changes in the ordering of design systems and does not provide sensible bounding conditions for the amount of time required for the system to converge. Although these techniques suggest orderings for the design process, only [44] provides a prediction for the overall convergence time. Simulation-based techniques have predicted the convergence time for concurrent engineering in [59], for overlapping tasks in [60] and for using DSMs in [47] which was further refined in [36].

A general convergence model for use in specifying architectures was presented in [61]. Simulation has also been used to tie process architecture to solution quality and suggest strategies to realize better products in [62]. Another approach attributed design process delays to incomplete sharing of design information and provides a dynamic work transformation model to determine when incomplete sharing occurs in [63]. A case was made for ordering design tasks to reduce the amount of uncertainty inherent to the problem in [64]. An analytical technique derived specifically for distributed design processes examined the relationship between the system transient response and the ordering of the solution process in [65] for two-subsystem systems.

This work differentiates itself from other techniques by providing an approach to determine the upper bound for the number of iterations required for a distributed design process to reach equilibrium. It does not require simulation to evaluate a proposed architecture and provides estimates based on the coupling of the system's component subsystems. This provides a computationally inexpensive initial evaluation of process architectures where the most promising architectures can be evaluated using more expensive, time consuming techniques. It differentiates itself from [65] through its applicability to large design systems and from previous work by requiring no system simulation. Before

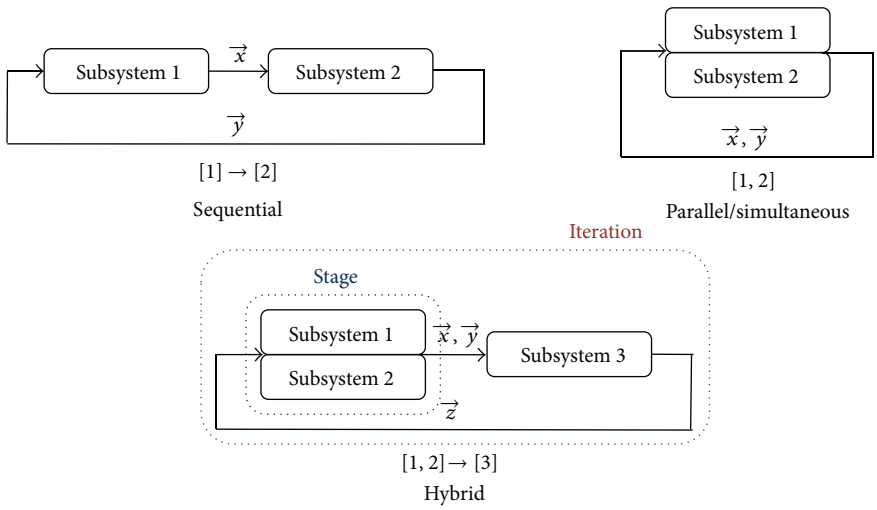

Figure 3: Potential process architectures.

presenting the approach, however, the concept of solution process architecture is introduced and its influence on the systems transient response is discussed.

2.5. Solution Process Architecture. The solution process architecture is the organization or structure used to solve a distributed design process. This structure can include both sequential and simultaneous solution processes. Pure sequential or pure simultaneous process architectures are the two extreme cases for process architectures. In Figure 3, a simple diagram illustrates the iterative process for a purely sequential architecture, a purely simultaneous architecture, and a hybrid approach which utilizes both sequential and parallel elements.

Each of the three process architectures described in Figure 3 represents a single iteration of the solution process. Repeated iterations of the architecture are used to solve a specific design process. These iterations can be further broken down and a single subsystem or a set of subsystems arranged in parallel with one another is called a stage. The difference between iteration, stage, and subsystem is shown for the hybrid architecture in Figure 3. The number of stages in process architectures depends on the number of subsystems and the process architecture chosen. For purely sequential process architectures, the number of process stages is always equal to the total number of subsystems. In contrast simultaneous, or parallel, process architectures always consist of a single stage. The number of process stages for sequential and parallel process architectures provide an upper and lower bound, respectively, for the number of stages in hybrid process architectures. For example, the hybrid process architecture in Figure 3 has two stages.

The stability criteria developed by Chanron and Lewis and shown in (7) is applicable to the parallel process architecture in Figure 3. In a recent extension, this criteria was refined to encompass simultaneous and hybrid systems as well [5]. This extension represents the design system in the same form as (6) but makes some allowances for process architecture changes. It was also demonstrated that process architecture has a significant impact on both the system stability, and the transient response. The location of the equilibrium, however, remains unchanged.

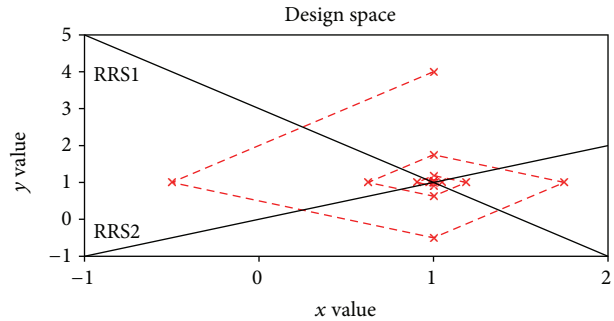

Figure 4: Two subsystem—parallel design architecture.

Some approaches have also utilized process architectures with partial overlapping between subsystems, where a new subsystem begins solving its optimization problem after another has already begun but not finished its optimization [60]. Since this is an initial investigation, we consider the three architectures shown in Figure 3 and assume that all subsystems begin a stage simultaneously with no overlapping.

To illustrate the relationship between process architecture and convergence time, the system described in (3) is simulated using a sequential process architecture in Figure 1. Since the equilibrium design variable values are known a priori, the process is said to have converged if all the design variables values are within 2% of their final values. Given this criterion, convergence takes 26 iterations for a sequential architecture. In contrast, the convergence plot for a parallel architecture for the two subsystem problem is shown in Figure 4.

A comparison of the dotted line representing the convergence path in Figures 1 and 4 demonstrates very different paths from the same starting location at $(1, 4)$ to the same Nash equilibrium at $(1, 1)$. This difference is caused by the way the designers share design variables. The difference in the

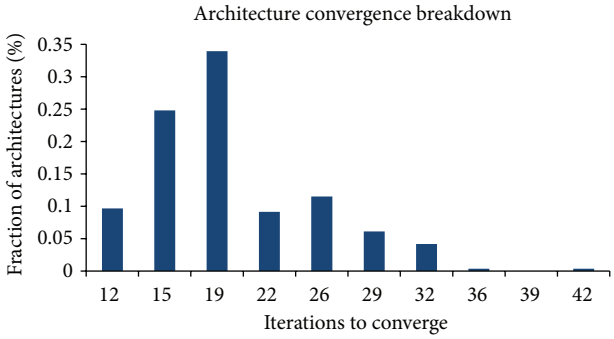

Figure 5: Convergence times for the five designer problem.

path taken by the designers also translates to a difference in solution time, with the parallel system requiring only 22 iterations to converge using the same criteria as used in Figure 1. A more comprehensive examination of the differences between these two architectures has been summarized in a study of convergence time in distributed design systems found in [65].

In a two-designer system, there are only two potential design architectures. However, as the number of designers increases, there are a large number of potential design architectures that fall into the category of hybrid. These hybrid architectures can have a significant impact on the system stability and convergence time. For larger systems, there are a wide range of architecture options with very different associated convergence times. This can be demonstrated by considering the same problem used in [3] to study the stability of large systems. This problem is an unconstrained five-designer problem with sixteen unique design variables. The convergence times for the different randomly generated architectures simulated from a variety of initial starting locations are plotted in Figure 5.

In Figure 5 the architectures are grouped into bins based on the number of iterations required to converge, demonstrating the wide range of convergence times that can be obtained by changing the process architecture. The height of the bars indicates the number of process architectures with a specified range of convergence times, shown on the x-axis. The mean convergence time for the simulated architectures is 25 iterations with the fastest convergence of 14 iterations and the slowest convergence of 42 iterations.

Although all of the process architectures studied in Figure 5 have a stable Nash equilibrium, there are some cases when changing the process architecture can change the system stability. This is because changing the process architecture also changes the system eigenvalues as shown in [5]. The influence of architecture, called topology, on convergence, stability, and equilibrium is also studied for real systems in [66, 67]. Where Braha and Bar-Yarn develop a descriptive approach to characterize decision networks in their work, this paper analyzes normative models to characterize convergence rate. Since system stability can be assessed by analyzing the system eigenvalues, it is proposed in this work that the eigenvalues associated with process architectures can be used to evaluate those systems' transient response. Examining the relationship between the system eigenvalues and the transient response is the focus of Section 3.

Table 1: Spectral radius experiment parameters.

Parameter	Value
Number of designers	4 to 10
Number of design variables	4 to 15
A	−20 to 20
C	−20 to 20

3. Results and Discussion

Since an eigenvalue analysis is used to determine the stability of specific solution process architectures, it is natural to examine eigenvalues to determine the architectures' convergence times. Existing linear systems theory evaluates system eigenvalues to determine settling time, natural frequency, modal response, system damping, and a number of additional properties. Furthermore, empirical evidence in Figure 6 suggests a relationship exists between the spectral radius of a distributed design system and the convergence rate.

The data in Figure 6 was generated by evaluating the spectral radii associated with ten different solution process architectures for five randomly created distributed design systems. The data used to create these systems is shown in Table 1 and the systems themselves are in the form of (7) and (8). In order to reduce the number of possible parameters in the experiment that may bias the result, the values in the D vector were set to zero to guarantee all the design systems had equilibrium solutions at the origin. Also, each subsystem was given local control of one design variable while the remaining variables were randomly allocated to the different subsystems.

The process architectures with spectral radii greater than 1 were not included in Figure 6 because they had unstable equilibrium solutions. To minimize the impact of starting location on the convergence behavior, each data point in Figure 6 is the average of twenty simulations started from a set of different points. Similar to the previous simulation, the process is defined to converge when the design variables are all within 2% of their final values. Although there is some correlation between the spectral radius and the mean number of iterations to converge, the circled architectures demonstrate that this mapping is not monotonic as some previous work has suggested [3]. Systems with the same spectral radius can have very different convergence times. This variation is demonstrated less dramatically across several of the other architectures with smaller spectral radii as well. Even when design systems have the same convergence time, the spectral radii of those systems can vary significantly. The systems generated in this section are used to experimentally support the approach outlined in this paper.

As demonstrated in Figure 6, the spectral radius is insufficient to quantify the convergence time of a distributed design process. For linear control systems, the real and imaginary components of the eigenvalues are used to determine the

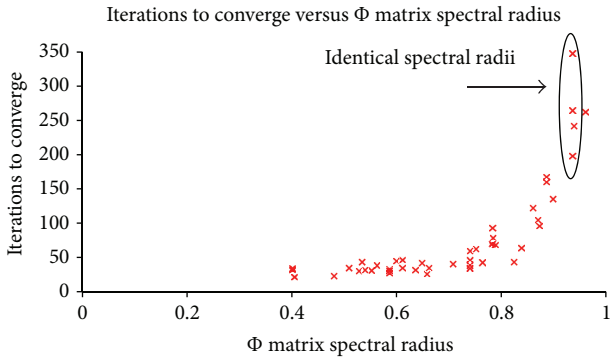

FIGURE 6: Iterations to converge versus phi spectral radius.

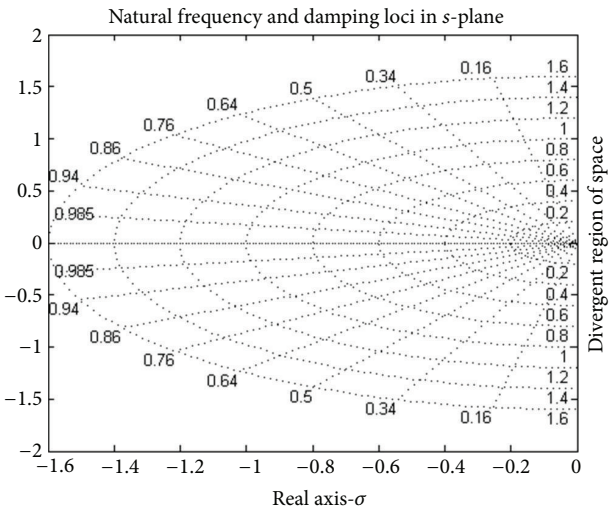

FIGURE 7: *s*-Plane—natural frequency and damping loci.

natural frequency and damping ratio of the system. Using these quantities, an upper bound can be determined for the overall system convergence time. This approach is adapted in this work to quantify convergence time in decentralized design systems. However, determining the natural frequency and damping ratio for distributed design processes is challenging and is the focus of the remainder of this paper.

Since decision networks are inherently discrete systems, they are modeled as discrete time state space systems. To determine the natural frequency and damping ratio of general discrete time systems, they are converted to continuous time approximations. However, distributed design processes are unique because they do not possess any underlying continuity. A continuous time model is, therefore, a true abstraction of the actual behavior.

In this paper, two techniques are examined to convert between the continuous and discrete time domain. These techniques are the zero-order hold and the bilinear, or Tustin, approximation. These two techniques are chosen because they are commonly used in the systems literature and the Tustin approximation is generally acknowledged as the preferred technique for this type of conversion [46]. This paper is not an exhaustive analysis of techniques that can be used to transform discrete distributed design systems into continuous time representations. Instead, it examines two prevalent approximations and identifies the one which results in a continuous time model that can exactly reproduce subsystem decisions at the appropriate discrete point in time.

Before any of these analyses can be conducted, however, a discrete time state space model must first be created to accurately represent the system. Creating these state space models is the topic of [3, 5] and in this paper all state space models are created using these processes. The fundamental difference between discrete time and continuous time state space models is that their eigenvalues are plotted in the z-plane rather than the s-plane. The transforms required to plot these eigenvalues are discussed in the following section.

3.1. Laplace and Z Transforms for Distributed Design Models. The primary challenge in quantifying the convergence time of distributed design processes is successfully transforming them to the continuous time domain. The advantage of a continuous time representation of a system is that it enables the eigenvalues to be plotted in the s-plane to capture the frequency response characteristics of the system. However, to plot a system in the s-plane, it must first be represented as a set of algebraic equations in a single value, typically s, by taking the system's Laplace transform.

The roots, poles, or eigenvalues of these algebraic equations have both real and imaginary components and are plotted in the complex s-plane where the real components are plotted on the x-axis while the imaginary components are plotted on the y-axis. The basic information provided in this plot is a system's natural frequencies, damping ratios, and stability characteristics. Furthermore, these properties can often be determined through inspection. A representation of the s-plane generated using the *sgrid()* command in MATLAB which includes lines of constant natural frequency and damping is shown in Figure 7.

In Figure 7, the lines radiating outward from the origin into the second and third quadrants are lines of constant damping ratio, ζ. The concentric arcs in Figure 7 centered at the origin and extending through the second and third quadrant are curves of constant damped natural frequency, ω_n. Systems with eigenvalues located on the left hand side of the y-axis are stable and settle to an equilibrium value in finite time, while systems with eigenvalues located on the right hand side of the y-axis are unstable and diverge. If the eigenvalues are on the y-axis in the s-plane, they are saddle points and the system oscillates forever, without moving closer or further away from the equilibrium value.

In the same way, an s-plane representation captures the characteristics of a continuous time system, and the z-plane can be used to capture characteristics of discrete time systems. To represent a system in the z-plane, the z transform of the system's time-invariant difference equations is taken to create an analytical expression in terms of a single variable, z. Once again the roots of this expression are plotted, this time in the z-plane. Although the s-plane and z-plane are both complex planes, the z-plane's properties are significantly different. A plot showing contours with constant natural frequency and

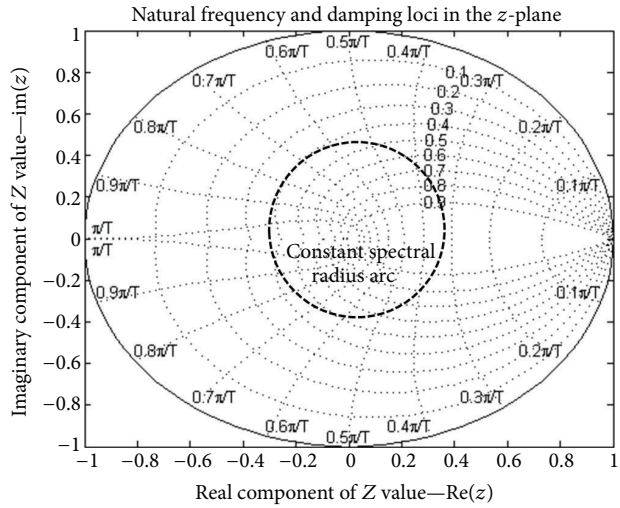

Figure 8: z-Plane—natural frequency and damping loci.

damping in the z-plane was generated using the MATLAB command *zgrid()* and is shown in Figure 8.

The stable region of the z-plane is shown in Figure 8 and corresponds to the region circumscribed by the unit circle, which graphically shows the stability criteria in (9). The region outside the unit circle is unstable and the unit circle itself is the set of saddle points. The region inside the unit circle has two sets of contours. The contours originating from the x-axis at $(1, 0)$ are curves of constant damping ratio. The other set of contours, perpendicular to the unit circle, are arcs of constant natural frequency.

Examination of the z-plane explains why eigenvalues, which correspond to roots of the z transform, with the same magnitude have a wide range of convergence times in Figure 6. Although these eigenvalues have the same magnitude, they map to very different natural frequencies and damping ratios. The damping ratio and natural frequency of a system can be used to determine the system's convergence time. For a second order, linear time invariant system, the convergence time, called the settling time, can be related to the damping ratio and natural frequency using (10) [68] as

$$t_s < 4.6\tau,$$
$$\tau = \frac{1}{\omega_n \zeta}. \tag{10}$$

In (10), t_s is the settling time, which is defined to be the time required for a system to converge to within 2% of its final value as measured from its initial value. The variable τ is the system's time constant which is the inverse of the natural frequency, ω_n, multiplied by the damping ratio, ζ. In this paper, we examine the applicability of (10) to systems that are not inherently continuous and use it to create an upper bound for their convergence time.

To apply (10), the natural frequency and damping ratio associated with the system roots is required. Unlike an s-plane representation, this value cannot be read directly from a plot in the z-plane and it depends on the sampling period for the discrete system. This sampling period is also critical to map system roots between the z-plane and the s-plane, which is a desirable transformation since the analysis of system transient response is often conducted in the frequency domain represented by the s-plane. The relationship in (10) is an example of an approach to approximate the convergence time of a continuous time system. The relationship between the complex variables in the z- and s-planes is mathematically expressed in (11) [46] as

$$z = e^{Ts}. \tag{11}$$

In (11), z is a complex number representing a root in the z-plane while s is the associated root in the s-plane. The value T is the sampling period for the system, usually measured in seconds. Since T can be any positive value, a single point in the z-plane can be mapped to many values in the s-plane depending on the sampling rate. The entire left hand side of the s-plane is represented by the unit circle in the z-plane because both encompass the entirety of the stable regions for a linear system.

The relationship expressed in (11) maps points between the z- and s-planes and can be applied to points in discrete time systems sampled at the appropriate rate. It is leveraged later in this work to transform eigenvalues of distributed design systems. However, to this point it has not been used to map poles of distributed design processes from the z- to s-plane because T has remained indeterminate. To facilitate this mapping, an analogy for the sampling rate of distributed design processes is discussed in the next section and the zero-order hold and bilinear transformation are presented to transform discrete systems into continuous systems and determine T.

3.2. Distributed Design Sampling Rates and Transformations. Distributed design systems are unique because they are truly discrete systems. A distributed design process is composed of a set of point discontinuities that represent specific decisions made at an instant in time. Since the process is governed by discrete iterations, there is no underlying continuity between decision points. This contrasts with most other systems that are fundamentally continuous, but sampled in time to create a discrete representation. The sampling rate for these continuous time systems is of critical importance, because low sample rates can inhibit the accurate reproduction of the continuous time signal. Furthermore, relationships used to analyze and reconstruct the signal for discrete systems generally require a specific sampling rate. To capture the system's transient response for this reconstruction, it must be sampled at a minimum rate called the Nyquist frequency [68].

When a continuous time system is sampled at a rate above the Nyquist frequency, the resulting set of points is an abstraction of the actual continuous time system. Since distributed design processes are by their nature discrete, creating a signal in the continuous time domain is not a reconstruction of the signal but an abstraction of the system's true behavior. This distinction restricts which linear system tools and techniques can be used to analyze and model distributed design processes. In this work, the zero-order

Table 2: Eigenvalues associated with system approximation.

Approximated eigenvalues	
Zero-order hold method	Tustin approximation
−0.367	−0.363
$-0.482 \pm 1.33i$	$-0.740 \pm 1.43i$
−1.62	−1.34

hold and the Tustin, or bilinear, approximation are used to construct a continuous time representation for distributed design processes to determine T. A simulated annealing algorithm was used to determine the sampling rates that led to the most accurate overall representation. The objective function for the simulated annealing algorithm was the distance between the discrete time points and the continuous time curve as measured using an L_2 norm. The design variable in the optimization formulation shown in (12) is T.

$$\text{Minimize: } f(T) = \sqrt{\sum_{j=1}^{k}\sum_{i=1}^{m}\left(x_i^j - g_i^j(T)\right)^2} \qquad (12)$$

$$\text{Subject to: } T > 0.$$

In (12), x_i^k is the design variable value for the ith design variable at the kth iteration. The term $g_i^j(k)$ is the ith design variable value as determined from the continuous time model generated using either the zero-order hold or Tustin approximation at the jth iteration. Finally, T, the sampling rate, is the design variable value for the optimization and is used to develop the continuous time expression $g_i^j(k)$. From this experiment, it is found that a sample rate of 1 sample per second most accurately represented the discrete time system in continuous time for both the Tustin approximation and zero-order hold.

Even though both transformations used the same sampling rate, the zero-order hold and Tustin approximation did not construct identical continuous time representations. As an example, the four largest eigenvalues for one of the simulated systems are summarized in Table 2 for both the zero-order hold representation and the Tustin approximation.

Both approximations result in models with different continuous time eigenvalues. Examination of the eigenvalues in Table 2 shows that both methods identify almost the same first eigenvalue for the design system. However, the second eigenvalue identified is significantly different for each method.

When determining the appropriate sampling rate, the zero-order hold representations are more accurate when considering the distance between the continuous and discrete time points as measured using the objective function in (12). The difference between these two representations is more obvious when plotted. The curves generated for the first design variable of the system summarized in Table 2 are plotted in Figures 9 and 10, respectively.

In Figures 9 and 10, time is plotted on the x-axis and the systems reach their equilibrium value after eleven

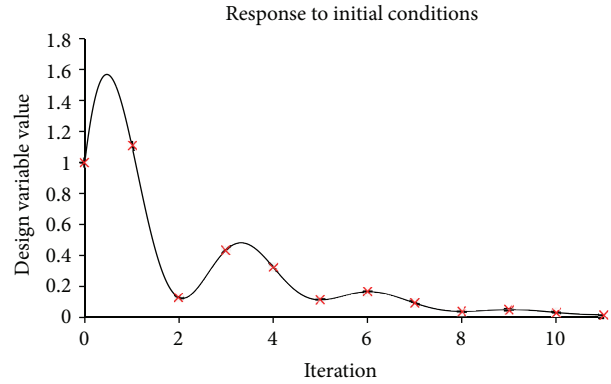

Figure 9: Zero-order hold method.

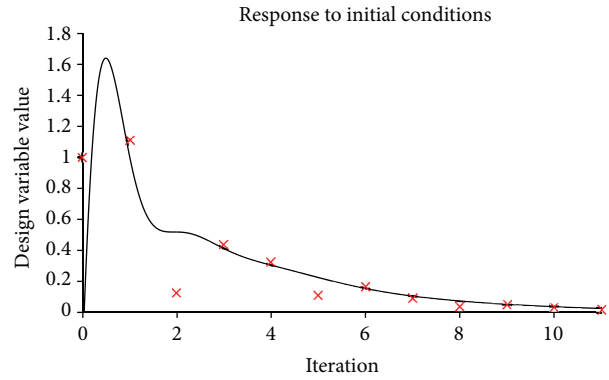

Figure 10: Tustin Approximation.

iterations. The design variable value is plotted on the y-axis, starting from an initial value of one and converging to a value of 0. Both plots show the first design variable for the simulated system and are created using the *initial()* function in MATLAB. The actual discrete design variables, as determined through simulation, are marked by x's in both Figures 9 and 10.

The curve in Figures 9 and 10 is the continuous time approximation for the discrete system. Inspection of the two figures shows that the approximation based on the zero-order hold passes through every discrete design variable value. This representation is more desirable because each discrete design variable represents an actual decision in the design process. The Tustin approximation does not pass through every design point in Figure 10 and does not accurately reproduce the design process.

The zero-order hold produces an accurate continuous time model for the system because its assumptions match extremely well to the fundamental mechanics of distributed design processes. The zero-order hold converts a discrete time signal to continuous time by holding a single sample's values constant over the sampling period. This signal reconstruction technique mirrors distributed design processes, where each subsystem assumes constant non-local design variable values when solving their local optimization problem for a single decision step in the process.

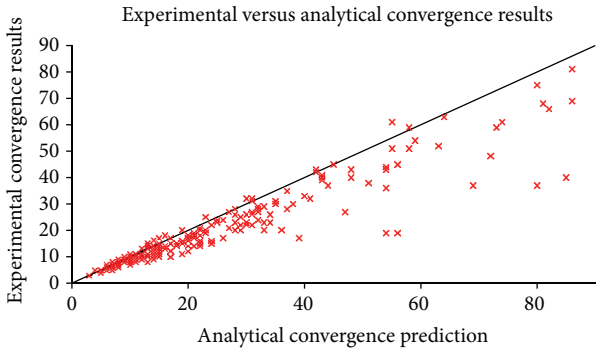

Figure 11: Experimental versus analytical convergence results.

Using the zero-order hold to transform a discrete state space model of a distributed system enables the analysis of the system using classical control techniques. The identification of the appropriate sample rate for these models makes the transformation from the z-plane to the s-plane simpler as well. In the next section, an analytical relationship based on these findings is introduced to transform discrete time eigenvalues to their continuous time equivalents. These continuous time equivalents are then analyzed using control theoretic relationships and the results are validated experimentally.

3.3. Analyzing the Convergence Time of Distributed Design Processes. In this section, the concept of the system time constant is examined for distributed design processes and its applicability to discrete time systems is demonstrated empirically. The mathematical relationship for continuous time systems between the natural frequency, damping ratio, and time constant is summarized in (10). It is demonstrated using the zero-order hold that it is possible to model distributed design processes as continuous time systems, and it is proposed that the relationship in (10) can be used to create an upper bound on the convergence time of distributed design processes as well. Since (10) requires only information about the damping ratio and natural frequency, a complete continuous time model of the system is not required. Instead, a simpler relationship can be used to determine these properties from the discrete time eigenvalues. These relationships are provided in (13) and (14) [46] as follows:

$$\frac{\zeta}{\sqrt{1-\zeta}} = \ln\left(\frac{|z|}{\angle z}\right), \tag{13}$$

$$\omega_n = -\ln\left(\frac{|z|}{T\zeta}\right). \tag{14}$$

In (13) and (14), ζ is the damping ratio, and ω_n is the natural frequency. To determine $|z|$, and $\angle z$ it is assumed that the z-plane form of the eigenvalues is $z = a + bi$, with a being the real part of z and b being the imaginary part. Using a and b the magnitude, $|z|$ is the vector sum of a and b while $\angle z$ is the $\arctan(b/a)$. The parameter T in (14) is the sampling period for the system. Recall that this does not have a direct analogy for distributed design systems but is discussed in Section 3.2 and is 1 sample per iteration.

Equations (10), (13), and (14) are used to evaluate the convergence time for 250 distributed design systems using the parameters outlined in Table 1. The settling time for each system is analytically determined and plotted in Figure 11 against the actual settling time for the system as determined by simulation.

The line in Figure 11 is shown as a reference to compare the convergence time determined by simulation to the analytical convergence rate predictions. It has a slope of one to differentiate between cases where the approach over and under predicted the system convergence time. For almost all the systems, the analytical results, determined using the settling criterion from (10), provide an upper bound for the convergence time of the design systems. Approximately 7.6% of the systems are above the solid line in Figure 11, which means that the approach underestimated the convergence time. The furthest system above the line exceeded the predicted maximum convergence time by 6 iterations, which is 10% of its convergence time. All other points, however, are at most 2 design iterations greater than their predicted value which means that the prediction was very close to the simulated convergence time.

One advantage to this approach is its efficiency, since it only considered the largest pole of the system, required no simulation, and is independent of the starting location. The dynamic behavior of many of the systems may have been understated with this simplification. For example, the system whose convergence was underpredicted by 6 iterations was composed of 6 unique subsystems collectively controlling 32 unique design variables. To capture the dynamics of more sophisticated design problems using a single eigenvalue may be insufficient to appropriately model the system.

In spite of only considering one eigenvalue, the approximation provides an upper bound for many of the systems. One reason for this may be the discrete nature of distributed design processes. A continuous time model for a truly discrete system overestimates the settling time because the gaps between discrete time points enable it to skip past the peaks of the continuous time curve. These peaks may remain outside of the 2% settling time longer than the discrete points unless a discrete time point is located on a peak. Additional areas of future work to increase the effectiveness of this approach for large systems are discussed in the next section.

4. Conclusions

In this paper, the transient response of distributed design processes, as modeled as a two-player continuous game, is characterized. An eigenvalue analysis formed the basis for this examination and concepts native to continuous control theory are used to evaluate and approximate the transient response of distributed design systems. The transient response is broken down into two components: shape and convergence rate. This paper focuses on the convergence rate of distributed design processes and it is shown that the convergence rate cannot be assessed using only the magnitude of the system's largest eigenvalue.

Instead of examining eigenvalue magnitudes, this paper modeled inherently discrete distributed design processes as

continuous time systems. Two commonly used approximations, the zero-order hold and Tustin approximation, are evaluated to determine the parameters required to analyze distributed design processes in continuous time. Both approximations suggest that the best sampling period, a key parameter in the conversion between discrete and continuous time systems, is 1 sample per iteration. It is also shown that the Tustin approximation does not provide an accurate reproduction of distributed design processes in continuous time. On the other hand, the zero-order hold provides a good approximation for these systems because its assumptions match well with the fundamental mechanics of distributed design processes.

By using a continuous time approximation, the convergence rate aspect of a transient response is evaluated. This analysis uses a second-order approximation for the system. It is demonstrated that for many systems a second-order approximation provides a reliable upper bound for the number of iterations required for a design system to converge to an equilibrium solution. In cases where the approach is unable to provide an accurate estimate of the system transient response, the prediction remained in the neighborhood of the time required for the system to converge as determined by simulation.

One of the major contributions of the approach presented is that it enables the evaluation of different design process architectures without the need to simulate the distributed design system. Another contribution of this approach is that it validates an extension of the linear system theory analogy used to model distributed design systems. It identifies the role of sampling in the overall system response and emphasizes the truly discrete nature of distributed design problems. For systems with dominant closed loop poles the approach provides a mathematically provable upper bound for the system convergence time. Even when this criterion is not met, the approach is able to bound the convergence time for most cases without the computational costs associated with simulation. This provides a filter to identify potential process architectures that are worth committing additional resources to investigating. Finally, it provides a basis for the analysis of increasingly complex distributed design systems by examining the basic challenges to predicting system transient response.

Future work will focus on extending the approach to analyze large systems by considering more than one eigenvalue. Performing this analysis will require the identification of the most influential closed loop poles to create a higher-order model of the system. This model will provide greater fidelity and predictive ability while reducing the computational costs to simulate the system. Another area of future work is in the application of model reduction techniques to reduce the size of the models for distributed design systems. Currently distributed design models require a number of states equal to the number of shared design variables. Reducing the model will minimize the computational cost to analyze these systems both with respect to their stability and transient response characteristics. Incorporating our technique with a comprehensive simulation-based approach to evaluate process architecture is another area of future work. Combining the fidelity of simulation-based techniques with the initial response characterization from this work will reduce the computational burden to identify preferred process architectures.

Finally, the expanded applicability of control theory techniques demonstrated in this work provides a foundation to investigate principles used to analyze nonlinear systems. Translating these principles into techniques that provide meaningful evaluations of convergence times will enable the analysis of a broader class of distributed design problems.

Acknowledgments

The authors would like to thank the National Science Foundation, Grant DMII-0322783, and the Moog Corporation for their support of this work. Approved for Public Release; Distribution Unlimited: 88ABW-2013-2106, dated May 1, 2013.

References

[1] V. Chanron and K. Lewis, "A study of convergence in decentralized design," in *Proceedings of the ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC '03)*, 48782, pp. 765–774, Chicago, Ill, USA, September 2003.

[2] V. Chanron and K. Lewis, "A study of convergence in decentralized design processes," *Research in Engineering Design*, vol. 16, no. 3, pp. 133–145, 2005.

[3] V. Chanron and K. Lewis, "Convergence and stability in distributed design of large systems," in *Proceedings of the ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC '04)*, 57344, pp. 593–603, Montreal, Canada, October 2004.

[4] E. Devendorf, P. Cormier, and K. Lewis, "Development of a distributed design toolkit for analyzing process architectures," in *Proceedings of the 13th AIAA ISSMO Multidisciplinary Analysis and Optimization Conference (AIAA '10)*, 9029, Fort Worth, Tex, USA, 2010.

[5] E. Devendorf and K. Lewis, "The impact of process architecture on equilibrium stability in distributed design," *Journal of Mechanical Design*, vol. 133, no. 10, pp. 101001–101013, 2011.

[6] E. Devendorf and K. Lewis, "Quantifying the convergence time of distributed design processes," in *Proceedings of the ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference (DETC '11)*, 48377, Washington, DC, 2011.

[7] Y. C. Kim, L. H. Keel, and S. P. Bhattacharyya, "Transient response control via characteristic ratio assignment," *Institute of Electrical and Electronics Engineers*, vol. 48, no. 12, pp. 2238–2244, 2003.

[8] C.-L. Li, "Characterization of the equilibrium strategy of fuzzy bimatrix games based on *L-R* fuzzy variables," *Journal of Applied Mathematics*, vol. 2012, Article ID 824790, 15 pages, 2012.

[9] V. Vidyottama, S. Chandra, and C. R. Bector, "Bi-matrix games with fuzzy goals and fuzzy pay-offs," *Fuzzy Optimization and Decision Making*, vol. 3, no. 4, pp. 327–344, 2004.

[10] T. Maeda, "On characterization of equilibrium strategy of two-person zero-sum games with fuzzy payoffs," *Fuzzy Sets and Systems*, vol. 139, no. 2, pp. 283–296, 2003.

[11] S. Chien and A. Sinclair, "Convergence to approximate Nash equilibria in congestion games," *Games and Economic Behavior*, vol. 71, no. 2, pp. 315–327, 2011.

[12] I. Caragiannis, A. Fanelli, N. Gravin, and A. Skopalik, "Computing approximate pure Nash equilibria in congestion games," *ACM SIGecom Exchanges*, vol. 11, no. 1, pp. 26–29, 2012.

[13] L. I. de Castro, "Equilibrium existence and approximation of regular discontinuous games," *Economic Theory*, vol. 48, no. 1, pp. 67–85, 2011.

[14] V. M. Perez, J. E. Renaud, and L. T. Watson, "Reduced sampling for construction of quadratic response surface approximations using adaptive experimental design," in *Proceedings of the 43rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference (AIAA '02)*, 1587, Denver, Colo, USA, 2002.

[15] J. Sobieszczanski-Sobieski and R. T. Haftka, "Multidisciplinary aerospace design optimization: survey of recent developments," *Structural and Multidisciplinary Optimization*, vol. 14, no. 1, pp. 1–23, 1997.

[16] A. Khajavirad and J. J. Michalek, "A deterministic lagrangian-based global optimization approach for quasiseparable nonconvex mixed-integer nonlinear programs," *Journal of Mechanical Design*, vol. 131, no. 5, pp. 051009-1–051009-8, 2009.

[17] E. J. Cramer, J. E. Dennis, Jr., 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.

[18] N. P. Suh, *Axiomatic Design: Advances and Applications*, Oxford University Press, New York, NY, USA, 2001.

[19] G. Pahl and W. Beitz, *Engineering Design: A Systematic Approach*, Springer, New York, NY, USA.

[20] S. Pugh, *Creating Innovative Products Using Total Design*, Addison-Wesley, Reading, Mass, 1996.

[21] M. M. Wiecek and J. A. Reneke, "Complex systems decomposition under uncertainty and risk," in *Proceedings of the 6th World Congress of Structural and Multidisciplinary Optimization*, Rio de Janeiro, Brazil, 2005.

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

[23] B. Wujek, J. E. Renaud, and S. Batill, "A concurrent engineering approach for multidisciplinary design in a distributed computing environment," in *Multidisciplinary Design Optimization: State of the Art*, N. Alexandrov and M. Y. Hussaini, Eds., pp. 189–208, NASA, Hampton, Va, USA, 2003.

[24] B. A. Wujek, J. E. Renaud, S. M. Batill, and J. B. Brockman, "Concurrent subspace optimization using design variable sharing in a distributed computing environment," *Concurrent Engineering Research and Applications*, vol. 4, no. 4, pp. 361–376, 1996.

[25] J. Sobieszczanski-Sobieski, J. S. Agte, and R. R. J. Sandusky, "Bi-level integrated system synthesis (BLISS)," NASA Technical Memorandum 208715, NASA, Hampton, Va, USA, 1998.

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

[27] N. Michelena, H. Park, and P. Y. Papalambros, "Convergence properties of analytical target cascading," *AIAA Journal*, vol. 41, no. 5, pp. 897–905, 2003.

[28] Y. Li, Z. Lu, and J. J. Michalek, "Diagonal quadratic approximation for parallelization of analytical target cascading," *Journal of Mechanical Design*, vol. 130, no. 5, pp. 051402-051401–051402-051411, 2003.

[29] S. Tosserams, L. F. P. Etman, P. Y. Papalambros, and J. E. Rooda, "An augmented lagrangian relaxation for analytical target cascading using the alternating directions method of multipliers," in *Proceedings of the 6th World Congress of Structural and Multidisciplinary Optimization*, Rio de Janeiro, Brazil, May 2005.

[30] G. Hardin, "The tragedy of the commons," *Science*, vol. 162, no. 3859, pp. 1243–1248, 1968.

[31] T. L. Vincent, "Game theory as a design tool," *Journal of Mechanisms, Transmissions, and Automation in Design*, vol. 105, no. 2, pp. 165–170, 1983.

[32] J. Eddy, *Solving distributed, non-cooperative design problems using multi agent systems [Ph.D. dissertation]*, University at Buffalo, Buffalo, NY, USA, 2006.

[33] K. Lewis and F. Mistree, "Collaborative, sequential, and isolated decisions in design," *Journal of Mechanical Design*, vol. 120, no. 4, pp. 643–652, 1998.

[34] C. Loch, J. Mihm, and A. Huchzermeier, "Concurrent engineering and design oscillations in complex engineering projects," *Concurrent Engineering Research and Applications*, vol. 11, no. 3, pp. 187–200, 2003.

[35] R. I. Whitfield, A. H. B. Duffy, G. Coates, and W. Hills, "Distributed design coordination," *Research in Engineering Design*, vol. 13, no. 3, pp. 243–252, 2002.

[36] S. Cho and S. D. Eppinger, "A simulation-based process model for managing complex design projects," *IEEE Transactions on Engineering Management*, vol. 52, no. 3, pp. 316–328, 2005.

[37] T. R. Browning and S. D. Eppinger, "Modeling impacts of process architecture on cost and schedule risk in product development," *IEEE Transactions on Engineering Management*, vol. 49, no. 4, pp. 428–442, 2002.

[38] E. Devendorf, M. Devendorf, and K. Lewis, "Using network theory to model distributed design systems," in *Proceedings of the 13th AIAA ISSMO Multidisciplinary Analysis and Optimization Conference (AIAA '10)*, 9027, Fort Worth, Tex, USA, 2010.

[39] K. Lewis and F. Mistree, "Collaborative, sequential and isolated decisions in design," in *Proceedings of the ASME Design Engineering Technical Conferences & Computers and Information in Engineering Conference*, Sacramento, Calif, USA, 1997.

[40] J. Nash, "Non-cooperative games," *Annals of Mathematics*, vol. 54, pp. 286–295, 1951.

[41] G. L. Thompson, "Signaling strategies in N-person games," in *Contributions to the Theory of Games*, H. W. Kuhn and A. W. Tucker, Eds., pp. 267–277, Princeton University Press, Princeton, NJ, USA, 1953.

[42] G. Hernandez, C. C. Seepersad, and F. Mistree, "Designing for maintenance: a game theoretic approach," *Engineering Optimization*, vol. 34, no. 6, pp. 561–577, 2002.

[43] K. Lewis and F. Mistree, "Modeling subsystem interactions: a game theoretic approach," *Journal of Design Manufacturing and Automation*, vol. 1, no. 1, pp. 17–26, 2001.

[44] R. P. Smith and S. D. Eppinger, "Identifying controlling features of engineering design," *Management Science*, vol. 43, no. 3, pp. 176–193, 1997.

[45] D. Poole, *Linear Algebra: A Modern Introduction*, Thomas Learning, 2003.

[46] K. Ogata, *Discrete-Time Control Systems*, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1995.

[47] V. Chanron, T. Singh, and K. Lewis, "Equilibrium stability in decentralized design systems," *International Journal of Systems Science*, vol. 36, no. 10, pp. 651–662, 2005.

[48] A. Gurnani and K. Lewis, "Collaborative, decentralized engineering design at the edge of rationality," *Journal of Mechanical Design*, vol. 130, no. 12, pp. 121101–121109, 2008.

[49] J. Kelly and M. Walker, "Critical-path planning and scheduling," in *Proceedings of the Eastern Joint Computer Conference*, Boston, Mass, USA, 1959.

[50] D. Malcolm, J. Roseboom, C. Clark, and W. Fazar, "Application of a technique for research and development program evaluation," *Journal of Operations Research*, vol. 7, no. 5, pp. 646–669, 1959.

[51] R. V. Slyke, "Monte carlo methods and the PERT problem," *Journal of Operations Research*, vol. 5, no. 11, pp. 839–860, 1963.

[52] A. Pritsker, *Modeling and Analysis Using Q-GERT Networks*, Wiley & Sons, New York, NY, USA, 2nd edition, 1979.

[53] S. Elmaghraby and J. Kamburowski, "The analysis of activity networks under generalized precedence relationships (GPR's)," *Journal of Management Science*, vol. 38, no. 9, pp. 1245–1263, 1992.

[54] J. Sobieszczanski-Sobieski, "On the sensitivity of complex, internally coupled systems," *AIAA Journal*, vol. 28, no. 1, pp. 173–180, 1990.

[55] J. L. Rogers, "DeMAID/GA an enhanced design manager's aid for intelligent decomposition," in *Proceedings of the 6th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization (AIAA '96)*, 4157, Seattle, Wash, USA, 1996.

[56] D. V. Steward, "The design structure system: a method for managing the design of complex systems," *IEEE Transactions on Engineering Management*, vol. 28, no. 3, pp. 71–74, 1981.

[57] S. D. Eppinger, D. E. Whitney, R. P. Smith, and D. A. Gebala, "A model-based method for organizing tasks in product development," *Research in Engineering Design*, vol. 6, no. 1, pp. 1–13, 1994.

[58] R. P. Smith and S. D. Eppinger, "A predictive model of sequential iteration in engineering design," *Management Science*, vol. 43, no. 8, pp. 1104–1120, 1997.

[59] A. Y. Ha and E. L. Porteus, "Optimal timing of reviews in concurrent design for manufacturability," *Management Science*, vol. 41, no. 9, pp. 1431–1447, 1995.

[60] V. Krishnan, S. D. Eppinger, and D. E. Whitney, "A model-based framework to overlap product development activities," *Management Science*, vol. 43, no. 4, pp. 437–451, 1997.

[61] B. A. Huberman and D. M. Wilkinson, "Performance variability and project dynamics," *Computational and Mathematical Organization Theory*, vol. 11, no. 4, pp. 307–332, 2005.

[62] J. Mihm, C. Loch, and A. Huchzermeier, "Problem-solving oscillations in complex engineering projects," *Management Science*, vol. 49, no. 6, pp. 733–750, 2003.

[63] A. Yassine, N. Joglekar, D. Braha, S. Eppinger, and D. Whitney, "Information hiding in product development: the design churn effect," *Research in Engineering Design*, vol. 14, no. 3, pp. 145–161, 2003.

[64] C. Sen, F. Ameri, and J. D. Summers, "An entropic method for sequencing discrete design decisions," *Journal of Mechanical Design*, vol. 132, no. 10, pp. 10100401–10100411, 2010.

[65] E. Devendorf and K. Lewis, "Are we there yet? Investigating the role of design process architecture in convergence time," in *Proceedings of the ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (DETC '09)*, 87517, pp. 997–1008, San Diego, Calif, USA, September 2009.

[66] D. Braha and Y. Bar-Yam, "Topology of large-scale engineering problem-solving networks," *Physical Review E*, vol. 69, no. 1, Article ID 016113, pp. 161131–161137, 2004.

[67] D. Braha and Y. Bar-Yam, "The statistical mechanics of complex product development: empirical and analytical results," *Management Science*, vol. 53, no. 7, pp. 1127–1145, 2007.

[68] K. Ogata, *Modern Control Engineering*, Prentice Hall, New Delhi, India, 4th edition, 2005.