RATIONAL DESIGN THEORY: A DECISION-BASED FOUNDATION FOR STUDYING DESIGN METHODS

A Dissertation
Presented to
The Academic Faculty

by

Stephanie C. Thompson

In Partial Fulfillment
Of the Requirements for the Degree of
Doctor of Philosophy in Mechanical Engineering

Georgia Institute of Technology

May 2011

RATIONAL DESIGN THEORY: A DECISION-BASED FOUNDATION FOR STUDYING DESIGN METHODS

Approved by:

Dr. Chris Paredis, Advisor
G. W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Bert Bras
G. W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. David Rosen
G. W. Woodruff School of Mechanical Engineering
Georgia Institute of Technology

Dr. Leon McGinnis
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Baabak Ashuri
College of Architecture
Georgia Institute of Technology

Date Approved: January 12, 2011

ACKNOWLEDGEMENTS

This work would not have been possible without the support of many people. First and foremost, I appreciate the guidance and encouragement of my advisor. Thank you, Chris, for your patience and support while I searched for a topic that excited us both. Although this topic is somewhat of a departure from other research in the lab, I think we settled upon something that we can both be proud of. I hope you feel the same way.

Thanks are also due to my committee members: Drs. Bert Bras, Dave Rosen, Leon McGinnis, and Baabak Ashuri. Many thanks for your insightful comments and helpful feedback.

I am grateful for the support of the National Science Foundation. Part of this work was supported by a Graduate Research Fellowship, and part was completed under grant CMMI-0522116. Thank you, taxpayers.

I also sincerely appreciate the support of my fellow lab members. Thank you Roxanne, Alek, Ben, Aditya, Kevin, and Rich! Through multiple office moves and an advisor on sabbatical in Europe, you all kept me going with your thought-provoking discussions and sanity-saving diversions.

Thanks also to the Woodruff School Graduate Women. I am lucky to have made friends with many fantastic women through this group, and the seminars and guest speaker events that were organized really helped me throughout my graduate career.

Last but certainly not least, I am grateful for the support and love from my family and friends. My parents have always encouraged me with their high expectations and unwavering belief in my ability to achieve my goals. My husband, especially, has provided the emotional support, constructive feedback, stimulating conversation, and unending love that I needed to complete this journey.

Thank you all!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

While design theories provide a foundation for representing and reasoning about design methods, existing design theories do not explicitly include uncertainty considerations or recognize tradeoffs between the design artifact and the design process. These limitations prevent the existing theories from adequately describing and explaining observed or proposed design methods.

In this thesis, Rational Design Theory is introduced as a normative theoretical framework for evaluating prescriptive design methods. This new theory is based on a two-level perspective of design decisions in which the interactions between the artifact and the design process decisions are considered. Rational Design Theory consists of normative decision theory applied to design *process* decisions, and is complemented by a decision-theory-inspired conceptual model of design.

The application of decision analysis to design process decisions provides a structured framework for the qualitative and quantitative evaluation of design methods. The qualitative evaluation capabilities are demonstrated in a review of the systematic design method of Pahl and Beitz. The quantitative evaluation capabilities are demonstrated in two example problems. In these two quantitative examples, Value of Information analysis is investigated as a strategy for deciding when to perform an analysis to gather additional information in support of a choice between two design concepts. Both quantitative examples demonstrate that Value of Information achieves very good results when compared to a more comprehensive decision analysis that allows for a sequence of analyses to be performed.

CHAPTER 1: MOTIVATION FOR A RATIONAL DESIGN THEORY

Design theories provide a theoretical foundation for design and can lead to insights that improve the practice of design. Many theories have been proposed, both from a descriptive (How do people currently design?) and a normative perspective (How should people design?) (Finger, et al. 1989a, b). In this dissertation, a theoretical framework is introduced that adopts a *prescriptive* perspective (Bell, et al. 1988). It is recognized that normative theories can only be applied in practice by making approximations and simplifying assumptions. From a prescriptive perspective the question then is: "Which approximations and simplifying assumptions lead to good design methods that are practically implementable?"

The theoretical framework introduced in this thesis attempts to answer this question. The framework, called Rational Design Theory (RDT), is inspired by decision theory. The criterion for "goodness" as referred to in the question above, is *rationality* — a rational design method leads to outcomes that are consistent with the knowledge and preferences of the designer.

In this chapter the motivation and requirements for this framework are introduced. In Section 1.1, the motivation for a new design theory is discussed. Thereafter, requirements for the new theory are identified in Section 1.2. The research questions and hypotheses addressed in this thesis are then introduced in Section 1.3. This chapter concludes with a look forward to the rest of the thesis in Section 1.4.

1.1 THE NEED FOR A NEW, RATIONAL DESIGN THEORY

Several researchers have attempted to formalize design in a mathematical framework to advance the field of design. As Braha and Reich note (Braha, et al. 2003), a mathematical formulation for design helps design researchers to understand the limits of automating design tasks, develop practical guidelines for design, and create design methods and tools. Most theories of design that have been proposed in the past have focused on the classification and representation of the types of knowledge and information that are used in design. However, little or no guidance is provided for the manner in which this knowledge should be applied. Although some authors suggest possible methods or prescribe certain methods based on observations or computational practices, little justification is provided to support the choice of one particular method over another.

A notable exception is the normative perspective advocated by researchers who think about design in terms of decision making. The notion of rationality in design can be attributed to Myron Tribus (Tribus 1969) who proposed to include Bayesian Decision theory in design. Hazelrigg (Hazelrigg 1996) further developed the idea, and from 1996 through 2004 a series of workshops were organized on the topic of Decision-Based Design culminating in an edited book on the topic (Lewis, et al. 2006a).

This earlier work on decision making in design takes a normative perspective, specifying practical design methods inspired by decision theory. However, any practical implementation requires the introduction of assumptions and/or simplifications. These assumptions and simplifications will result in a design artifact that is different from the artifact that would have been obtained under the ideal, exact application of the normative theory. To compare *practical* methods of design, a framework is needed that allows us to reason about the impact of approximations on both the quality of the resulting design artifact and on the cost of the design process used to arrive at that artifact.

There are two different levels of decision-making in a design problem: *process*-related decisions and the *artifact*[1] decision. It is the artifact decision that has been studied in detail in previous work. It concerns the choice of an artifact that provides the best tradeoff (under uncertainty) between performance, cost, reliability, etc. On the contrary, *process* decisions are concerned with the choice of the design actions that ultimately lead to a desirable artifact specification.

An explicit differentiation between artifact and process decisions is crucial for a new theory of design and can be the source for new insights. By modeling the interactions between artifact and process decisions, one can express tradeoffs between the artifact performance and the process performance, tradeoffs that previously could not be taken into account because only the artifact perspective was considered. For example, in Robust Design (Taguchi 1986, Phadke 1989, Bras, et al. 1991, Bras, et al. 1993, Chen, et al. 1996a, Chen, et al. 1996b), one recognizes that there are uncertain factors that influence the performance of the artifact. Rather than maximizing the nominal performance, it is then better to take the uncertainty into account and select an artifact that provides a good balance between nominal performance and variation around the nominal. In this fashion, the possible negative consequences of uncertainty are mitigated by modifying the artifact. In contrast, the designer could have adopted a *process* perspective to deal with the uncertainty. For instance, by investing in additional

[1] The term *artifact* is defined in Section 3.1: The Purpose of Design.

simulations or physical experiments, some of the uncertainty might have been eliminated, reducing the need to make the artifact robust. Rather than investing in the artifact — robustness usually comes at a price — it might be more advantageous to invest in the process; or most likely, it would be most advantageous to find a good balance between investing in the process and investing in the artifact. These tradeoffs between investing in the process or the artifact require a framework in which the interactions between artifact and process decisions are modeled explicitly.

To clarify the discussion in this thesis, a design method must be clearly distinguished from a design theory. The following terminology is adopted throughout this work:

DEFINITION 1 A *design action* is an (information-processing) action undertaken by a designer.

DEFINITION 2 A *design process* is a sequence of design actions.

DEFINITION 3 A *design method* is a specification of a design process — a method for selecting the next design action at each step in a design process.

DEFINITION 4 A *design theory* is a model of the act of designing that allows for interpretation, comparison and prediction of characteristics of design methods.

Having established the motivation for a new design theory, requirements for the theory are identified in the next section.

1.2 REQUIREMENTS FOR A NEW DESIGN THEORY

A design theory must meet two general requirements: adequately describe design processes and provide a theoretical foundation for assessing the quality of design methods. Most existing theories have focused on adequately describing design without providing a foundation for assessing the quality of design methods. To adequately describe design processes, a design theory must meet five requirements, as shown in Table 1.1. In his critique of General Design Theory, Reich identifies two requirements for an adequate description of design processes (Reich 1995). He asserts that a descriptive model must include the generation of a specification or the ability to revise the means for evaluating concepts. In addition, a model of design processes must include the creative process of generating new concepts. Three additional requirements are identified to improve upon existing theories. First, the description of design processes must explicitly include uncertainty in the outcomes of concepts and design tasks. Also, a descriptive model of design processes must not limit the expression of beliefs of preferences. This requirement prevents the new theory from imposing arbitrary beliefs or preferences on a designer, which would lead to irrationality. Finally, the theory must incorporate the allocation of design phase resources as outcomes of design actions. Many of the existing theories of design do not address the resources consumption during the design process. Without considering this reality of design processes, a new theory cannot adequately capture design processes.

Requirements 6 and 7 in Table 1.1 establish a theoretical foundation for the assessment of design method quality. The first requirement is that the theory must recognize a tradeoff between the design artifact and the design process. This is related to requirement 5 that the theory must recognize the cost of design actions. Not only must the costs be modeled, but the theory must recognize that these costs are accrued in the act of improving artifact quality. Thus, there is a tradeoff between reducing design costs and improving artifact quality. The last requirement for the new theory is that it provide a means for evaluating design methods relative to one another. This is another requirement that is not met by existing design theories; however, without a means for evaluation of design methods, a new theory cannot provide guidance for the development of new methods and the improvement of existing design methods. Therefore, a new theory must provide this evaluation capability.

TABLE 1.1. REQUIREMENTS FOR A NEW DESIGN THEORY

	Requirements for a descriptive conceptual model of design
1	Includes the generation and revision of specifications
2	Includes the creative process of generating new concepts
3	Explicitly includes uncertainty
4	No limit on the expression of preferences and beliefs
5	Incorporates the allocation of design phase resources as outcomes of design actions
	Requirements for a normative framework for evaluating design methods
6	Recognizes a tradeoff between artifact and process
7	Provides a means of evaluating design methods relative to one another

These requirements taken together establish the characteristics of a design theory that is capable of evaluating design methods. In the next section the research questions addressed in this thesis are presented.

1.3 RESEARCH QUESTIONS AND HYPOTHESES

The primary research question comes from the primary motivation of this thesis: to provide a theoretical framework for assessing design methods.

Primary Research Question: *What theoretical foundation for design has explanatory power and enables quantitative evaluation of design methods?*

Hypothesis: *Rational Design Theory, the application of normative decision theory to design process decision making, enables the prediction and comparison of design method quality.*

The primary hypothesis is that a new theory of design can provide this explanatory power and enable quantitative evaluation through the application of normative decision theory to design *process* decision making. Because the ideal is rationality, the new theory is called Rational Design Theory. To provide explanatory power the theory must enable qualitative evaluation of design methods. Qualitative evaluation provides the ability to generate hypotheses about good design methods. These hypotheses can then be tested through quantitative evaluation. These two aspects are addressed separately in the secondary research questions.

Secondary Research Question 1: *Can Rational Design Theory be used to compare design methods in a qualitative sense?*

In Section 3.2.3 it is explained that any design method can be broken down into the three elements of a decision problem: alternatives, outcomes, and preferences. When design methods are characterized in such a manner, they can be compared on the basis of those elements. For example, methods for design under uncertainty specify a representation for the uncertain outcomes as well as preferences concerning the outcomes. By characterizing these methods in terms of alternatives, outcomes, and preferences, the methods can be compared qualitatively. This type of comparison has been done in the past (Lee, et al. 2010b). The hypothesis to answer this question seeks to provide organization to this type of comparison.

Secondary Hypothesis 1: *Yes, Rational Design Theory can be used to qualitatively compare design methods because the characterization of a design method in terms of alternatives, outcomes, and preferences enables the systematic identification of differences in the expression of alternatives, outcomes, and preferences as compared to the normative rational ideal.*

As mentioned above, the qualitative comparison capabilities of RDT make it possible to generate hypotheses about what makes a good design method. To test these hypotheses, a theory is needed for determining quantitatively the quality of a design method.

Secondary Research Question 2: *Does Rational Design Theory enable quantitative prediction of design method quality?*

Secondary Hypothesis 2: *Yes, Rational Design Theory enables quantitative prediction of design method quality through the systematic analysis of design process decisions.*

The analysis of design process decisions with respect to rationality enables both qualitative and quantitative evaluation of design methods. In this thesis, these hypotheses are validated in the presentation of Rational Design Theory and a corresponding conceptual model of design in Chapter 3. Qualitative evaluation is demonstrated with a review of the systematic design method of Pahl and Beitz in Chapter 4. The quantitative evaluation ability of RDT is demonstrated in two computational examples in Chapter 5.

1.4 THESIS OVERVIEW AND ROADMAP

In this chapter, the motivation for a new theory of design was presented. Requirements for a new design theory were identified, and research questions and hypotheses were presented. A new theory of design is needed because existing theories do not explicitly include uncertainty considerations, and they do not recognize tradeoffs between the design artifact and the design process. These limitations prevent the existing theories from adequately describing design processes and evaluating design methods.

The thesis continues in the next chapter with a critical review of the existing theories and related work in Section 2.1. In addition, decision theory is reviewed in Section 2.1.8 because it provides a theoretical foundation for Rational Design Theory. It is shown that although the existing theories do not meet the requirements, the application of normative decision theory to design process decisions is an approach capable of simultaneously meeting all the requirements for the new design theory.

In Chapter 3, Rational Design Theory is introduced as an embodiment of the primary research hypothesis. This new theory is based on a two-level perspective of design decisions in which the interactions between the artifact and the design process decisions are considered. In RDT, it is shown that the ideal design method maximizes expected Net Design Utility (NDU), which is a combination of utilities of the artifact and process decisions. To define NDU for particular design problems, RDT is augmented by a descriptive conceptual model of design. The conceptual model captures the information state of artifact concepts in three aspects: concepts, concept predictions, and concept decision criteria. In addition, design actions are classified as synthesis, analysis, or evaluation. After presenting RDT, characteristics of good methods as identified by RDT are presented. This discussion is followed by a return to the related work in design theory. Finally, Chapter 3 is concluded with a critical review with respect to the requirements for the new theory and the research questions and hypotheses.

RDT provides a theoretical framework for the qualitative and quantitative evaluation of design methods. In Chapter 4, qualitative evaluation is demonstrated with a review of the systematic design method of Pahl and Beitz in the context of RDT. The quantitative evaluation capabilities of RDT are demonstrated in two example problems in Chapter 5. These examples serve to validate the second secondary research hypothesis that RDT enables quantitative evaluation of design methods. In these two examples, Value of Information analysis is investigated as a strategy for deciding when to perform an analysis to gather additional information in support of a choice between two design concepts. Both quantitative examples demonstrate that Value of Information achieves very good results when compared to a more comprehensive decision analysis that allows for a sequence of analyses to be performed.

The thesis is concluded in Chapter 6 with a summary and critical review with respect to the research questions and hypotheses. Based on this critical review, opportunities for future work are identified.

CHAPTER 2: LITERATURE REVIEW

Having introduced the problem in the previous chapter, related literature is reviewed in this chapter. In Section 2.1 related works on existing design theories are reviewed to establish the need for a new theory. In Section 2.1.8, an overview of decision theory is provided to support the development of Rational Design Theory in the next chapter.

2.1 EXISTING DESIGN THEORIES AND RELATED WORK

In this section several theories of design and related works are reviewed to establish the state of the art and the research gap. The reviewed works include Simon's Sciences of the Artificial (Simon 1996), General Design Theory (Yoshikawa 1981, Tomiyama, et al. 1987), Function-Behavior-State diagrams (Umeda, et al. 1990), Gero's Design Prototypes (Gero 1990), Decision-Based Design (Hazelrigg 1996, Marston, et al. 1997, Hazelrigg 1998, Krishnamurty 2006, Lewis, et al. 2006b), Coupled Design Process (Braha, et al. 2003), and Concept-Knowledge theory (Hatchuel, et al. 2009). Other significant bodies of work in design literature such as Axiomatic Design[2] (Suh 2001), the systematic design method of Pahl and Beitz (Pahl, et al. 1996), and the TRIZ methodology (Altshuller 2004) are omitted from this review because they propose design *methods* rather than design *theories*.

In this review, it is shown that although existing design theories provide definitions of design, they do not provide a means for comparing the quality of one design method to

[2] Although the term *axiom* implies a theory, the independence and information axioms of Axiomatic Design are not fundamental truths, but rather heuristics for design decision-making. As such, we consider Axiomatic Design to be a design method rather than a design theory.

that of another. Further, existing design theories do not rigorously account for uncertainty, which is a driving factor in the act of design. A critical review of each theory is presented, highlighting the proposed definition of design and any prescriptions for good design methods.

2.1.1 Sciences of the Artificial

Although not a theory of design per se, Herbert Simon's book "The Sciences of the Artificial" (Simon 1996) laid the foundation for establishing theories of design by highlighting the need for a scientific approach to design. He notes that designing as an activity is not restricted to engineers. Rather, anyone who "devises courses of action aimed at changing existing situations into preferred ones" is engaged in the activity of design. Simon discusses a curriculum for design, which is instructive for the creation of a scientific theory of design. He mentions several existing fields that should be included in the curriculum, including utility theory, statistical decision theory, and optimization techniques. Also mentioned are several topics for the curriculum of design, including imperative and declarative logics of design, heuristics for search, allocation of resources for search, theory and structure of design organization (particularly hierarchical systems), and the representation of design problems. Dasgupta revisits Simon's Science of Design two decades after the original publication and evaluates Simon's claims in light of recent work (Dasgupta 1992). He finds that many of the important points of Simon's work, including satisficing and bounded rationality, have not been refuted. Rather, researchers have elaborated on Simon's ideas and developed design tools that are based on the artificial intelligence paradigm introduced by Simon.

Perhaps the two most significant ideas from Simon's work are the notions of bounded rationality and satisficing solutions. Bounded rationality is the idea that the limited cognitive ability of human decision-makers may render the process of finding the optimal design intractable or economically infeasible. Thus, while human decision-makers can act rationally for small problems, the limited processing power of the human brain (or computer) limits rational action for large or ill-defined problems. This limited processing power increases the time needed to search for and identify optimal solutions, and the investment in time to solve the problem often outweighs the benefits obtained by finding the optimal solution. Thus, bounded rationality leads to the need for satisficing solutions.

Satisficing solutions are solutions which are not optimal but are good enough. Satisficing solutions or techniques for finding satisficing solutions are needed when the global optimum is difficult to find. The traveling salesman problem is cited as an example of this type of problem. The difficulty in finding satisficing solutions depends on how high the standards of acceptability are set, not on the size of the search space; thus, one trades the global optimum for a decrease in time to find a solution. In design, this may be manifested by accepting the best alternative among the few that have been identified, without searching for additional alternatives that may yield better performance.

Satisficing is a tradeoff between design process performance and design product performance. This tradeoff must be made in any design process, but is often made arbitrarily. While the existence of bounded rationality and the need for satisficing solutions are not disputed, I believe that the tradeoff between design process performance and the performance of the designed artifact can be made more explicit.

2.1.2 General Design Theory

Tomiyama and Yoshikawa define the design process as an evolution of metamodels in General Design Theory (GDT) (Yoshikawa 1981) and Extended General Design Theory (EDGT) (Tomiyama, et al. 1987). A primary motivation of Yoshikawa in creating GDT was to support the development of future CAD systems. Thus, it was important for the theory to be mathematically rigorous in addition to being sufficiently expressive. GDT builds on a foundation of Axiomatic Set Theory to describe the logical aspects of the design process. This initial theory relies on the assumption of ideal knowledge which is summarized in the three axioms of recognition, correspondence, and operation. The axiom of recognition states that all entities can be recognized by their attributes. The axiom of correspondence states that there is a one-to-one correspondence between the entity set, S', and the set of entity concepts, S. This requires that there be infinite or perfect knowledge of entities. Finally, the axiom of operation states that "the set of abstract concepts is a topology of the set of entity concepts". This axiom enables the use of set operations on concept sets. These three axioms together describe the ideal knowledge of design. In this ideal knowledge, designing is "a mapping of a point in the function space onto a point in the attribute space".

In Extended General Design Theory, Tomiyama and Yoshikawa seek to consider the physical aspects of design in addition to the logical aspects addressed in the original theory. They identify the finiteness and imperfections of real knowledge as compared to the ideal knowledge. Real knowledge is finite because of limited storage capacity and finite operational speed (of the mind or computer), similar to Simon's bounded rationality. In addition, there are many sources of imperfection in the real knowledge, such as nonexistent entities for some abstract concepts, imperfect categorization, poor convergence or errors due to set-theoretic operations, mapping imperfections, and modeling/analysis error. Thus, in the real knowledge, designing is still a mapping from the function space to the attribute space, but because of finiteness and imperfections in the real knowledge, this mapping is accomplished through an intermediate space called the metamodel space. Design in the real knowledge is a stepwise process of refining these metamodels to meet the specifications. Because the refinement converges in the metamodel space, $\boldsymbol{M}$, rather than the attribute space, $\boldsymbol{T}$, the solution, s, is an approximation of the design solution.

Reich provides a thorough review and critique of GDT and finds the assumptions of the theory to be too limiting to adequately describe design (Reich 1995). He asserts that no realistic domains have the topological structure assumed by GDT. Reich also finds that GDT completely ignores the creative process of adding new entity concepts to the domain when the specification initially reduces to the empty set. Furthermore, Reich asserts that the process of design includes the creation and refinement of specifications, which is assumed to be given and fixed in GDT. As a result, revisions of specifications to avoid contradictions, to make the design feasible, or to adapt to changing requirements are not discussed in GDT. These limitations render the conclusions of GDT inapplicable for practical design problems.

One limitation not mentioned by Reich is the lack of an explicit characterization of uncertainty in GDT. Tomiyama and Yoshikawa allude to sources of uncertainty in the imperfections of real knowledge, and characterizations of imperfect knowledge of the design are captured in the intermediate metamodel space between the function and attributes spaces. From this, they recognize that a refined metamodel is merely an approximation of the design solution. It is possible to improve on this representation through explicit characterization of the sources and impacts of uncertainty. Because uncertainty is a driving force in design, it is imperative to characterize uncertainty in a design process as completely and accurately as possible.

2.1.3 Function-Behavior-State

The Function-Behavior-State (FBS) diagram is proposed to establish a clear distinction between function and structure and to establish a mechanism for constructing and reasoning with hierarchical structures of entities (Umeda, et al. 1990). It is a model of entities, not a model of a design process; however, the authors propose to use this diagram to discuss advantages and disadvantages of various design methods and modeling techniques. A function is "a description of behavior abstracted by [a] human through recognition of the behavior in order to utilize it". The *function* of an entity is dependent on the *view* of the person and is established through the subjective F-B relationship. Views can denote levels of abstraction or domains such as mechanics or electromagnetism. Within a view, *behaviors* of an entity can be objectively determined from the *state* of the entity using physical laws in the B-S relationship. *State* is "an instantaneous description of an entity", which includes as an important part the *structure* of the entity.

A significant point is that the authors note that function is a subjective, human notion, whereas state and behavior are objective. Thus, computers can be used to interpret behavior and state, but not function. The FBS diagram is helpful because it separates the subjective function from the objective behavior and state. Using the FBS diagram, one should proceed with design by first identifying functions and then connecting these functions to behaviors and states. Although this seems like a good, common-sense process for design, little justification is provided for the method. In addition, no mention is made of uncertainties in function, behavior, or state, or how these uncertainties might be handled in a design process.

2.1.4 Design Prototypes

Gero describes design as a process of transforming function (F) into design description (D) through a series of intermediate representations, including expected behavior (Be), actual behavior of the structure (Bs), and structure (S) (Gero 1990). Several types of transformations are identified. Direct transformation from F to D is rare, but can sometimes be achieved through catalog design. Accordingly, the F→D transformation only occurs in well-known domains. Translation from F to Be is a process of formulation or specification, in which designers detail their interpretations of the functional description. Analysis is the translation from S to Bs; analysis is an objective process, but the behaviors to be analyzed must be identified in advance. Once the F→Be and S→Bs transformations have been made, the expected and actual behaviors can be compared in a process of evaluation. This process determines if the synthesized structure is capable of meeting the expected behavior. If the structure meets the specifications, the transformation S→D produces the design description. These transformations and combinations of these transformations make up the actions of design: formulation, synthesis, analysis, evaluation, reformulation, and production of the design description.

Gero builds on this model of design processes and introduces the concept of design prototypes. Such prototypes constitute summaries of a design situation and can be reused by experienced designers to save work in the formulation of new problems. These prototypes are not necessarily formally captured by designers, and several prototypes may be referenced in a single design problem. Gero also discusses routine, innovative, and creative design in the context of design prototypes. Routine design corresponds to the selection of instances of design prototypes without modification. Innovative design makes use of existing design prototypes, but the instances of the prototypes are adapted. Creative design requires the creation of new design prototypes.

Gero's work is important for the identification of the actions in design and the types of transformations that occur in design processes. However, little guidance is provided for the preferred sequence of these actions. Gero does allow for reformulation of the problem, which is important. Uncertainty, particularly in the behavior of synthesized structures, is not discussed.

2.1.5 Decision-Based Design

Decision-based design (DBD) is a body of work that recognizes the important role of decision-making in design (Hazelrigg 1996, Marston, et al. 1997, Hazelrigg 1998, Krishnamurty 2006, Lewis, et al. 2006b). The umbrella of DBD represents a collection of similar design strategies rather than a unified design theory. An advantage of this perspective is that it brings the strength of decision theory to design research; however, not all DBD researchers apply decision theory in their work. In particular, there is a large body of work based on Decision Support Problems and the Decision Support Problem Technique (Mistree, et al. 1990, Marston, et al. 1997, Marston, et al. 2000). These works prescribe design methods rather than design theories. In this review, I are more concerned with the use of decision theory as a foundation for a theory of design. An overview of decision theory is presented in Section 2.1.8.

The application of decision theory to design is not a straightforward problem, and different researchers have attacked the problem differently. For example, some authors advocate an enterprise-driven approach: the selection of design solutions via the maximization of single-attribute utility based on profit (Hazelrigg 1998, Kumar, et al. 2006). Other works take a multi-attribute approach, such as work by Thurston and coauthors (Thurston, et al. 1994), Scott and Antonnson (Scott, et al. 1999), and Lewis and coauthors (Lewis, et al. 2006a). Despite differences in the form of the objective function, most decision-theoretic DBD approaches have in common several key characteristics: the characterization of design concepts or candidate solutions as decision alternatives, the explicit characterization of uncertainty (usually modeled with probabilities), and a customizable statement of preferences concerning artifact performance outcomes.

Although DBD has the strong foundation of decision theory, little guidance is provided for assessing the goodness of one decision-based design method over another. Utility theory indicates that decisions should be made to maximize the expected utility of the decision-maker, but it is not clear how to apply this maxim to design. Decisions are made throughout a design process, and previous work has framed those decisions in terms of the designed artifact. While the application of decision theory to the artifact decision has highlighted some ways to improve the evaluation of concepts and the modeling of designer or decision-maker preferences using utility functions, expanding the scope of design decisions to include process resources will enable new insights regarding design processes.

2.1.6 Coupled Design Process

Braha and Reich identify several properties of design processes that should be included in any mathematical model of design (Braha, et al. 2003):

- "Design starts from some abstract specifications and terminates with a description of a product.
- In design, the product specifications are gradually refined. Better understanding of the specifications is a by-product of design.
- Design is an iterative, exploratory, and sometimes chaotic process.
- Intermediate states of the design process might include conflicting specifications and product description.
- Design progresses by context-dependent activities: thinking, alternatives, and decisions emerge from the situation as it unfolds by bringing diverse knowledge to bear on the present context.
- Designers make use of diverse knowledge whether they work alone or collaboratively.
- Design is about finding solutions, not (globally) optimal solutions. Designers generally do not receive the knowledge or resources needed to achieve optimality."

In addition, Braha and Reich note that a mathematical framework of design must have sufficient detail to support theoretical statements about design and to enable the generation of improvements to practical processes.

To fulfill these requirements, Braha and Reich present a mathematical framework of design based on closure spaces. They show that General Design Theory is a special case of their framework. At a basic level, design is a process of translating requirements in the function space into a design description in the structure space. This process is iterative in nature. Refinements of the requirements in the function space and, later, specifications in the structure space are achieved in a stepwise manner via proximal refinement. Synthesis occurs when requirements are matched to structure and development moves from the function space to the structure space. At each step, proximal refinements are selected from within a closure space by a generating element; a sequence of these generating elements is the design process.

Extending the framework from this basic level, design descriptions exist in the Cartesian product of the function and structures spaces, $F \times D$, and at every step in design the description consists of a tuple of function and structure $\langle f, d \rangle$. A refinement step can update one or both aspects of the description. Furthermore, refinement steps are context dependent, requiring the input of both function and structure. Because both are refined concurrently, the process is called Coupled Design. Structural descriptions consist of values for several measureable or observable quantities such as physical or geometrical quantities. Functional descriptions contain functional properties, which are behaviors that an artifact exhibits under certain conditions. Functional descriptions indicate the desired behaviors of the artifact.

Although the authors identify many key characteristics of design, they fail to identify uncertainty and the reduction of uncertainty as a driver of design processes. They assert that imperfect, incomplete, or incorrect knowledge can be accounted for with an approximate closure, and that these approximate closures may help to explain design failures. This, however, is a model of the impacts of uncertainty, not a means for explicitly modeling uncertainty itself. While analysis is discussed using the neighborhood concept, uncertainties in analysis are not addressed. In addition, while the authors show that several distinct design procedures can be modeled in their framework, they do not make any attempt to discuss the quality of a particular method as compared to another.

2.1.7 Concept-Knowledge Theory

Hatchuel and Weil present C-K theory as a general theory of design which includes new models of thought for creativity and innovation (Hatchuel, et al. 2009). They discuss design as a process of expansion of sets in the Concept and Knowledge spaces towards a final design which is a “decidable proposition” in the Knowledge space. They note that a comprehensive definition for design must incorporate two processes: mapping between requirements and solutions and generation of new concepts. These are achieved using the four C-K operators: C-C, C-K, K-C, and K-K. The knowledge-to-knowledge (K-K) operator represents the standard model of thought, whereas the other three operators are new models which are unique to design.

Hatchuel and Weil discuss Braha and Reich's Coupled Design Process in the context of C-K theory and point out that the topological structure involving closure spaces of the Coupled Design Process is limiting because the refinement of predefined closure spaces can only find existing solutions through traditional thought processes rather than creative or innovative solutions.

While C-K theory does incorporate the creative aspect of design, it does not provide guidance concerning the goodness of one design method over another, nor does it allow for the explicit representation of uncertainty.

TABLE 2.1. REQUIREMENTS ANALYSIS OF EXISTING WORK

Requirement	(E)GDT	FBS	Gero	DBD (artifact)	Coupled Design	C-K
Includes the generation and revision of specifications			x		x	
Includes the creative process of generating new concepts						x
Explicitly includes uncertainty				x		
No limit on the expression of preferences and beliefs				x		
Incorporates the allocation of design phase resources as outcomes of design actions						
Recognizes a tradeoff between product and process						
Provides a means of evaluating design methods relative to one another						

2.1.8 Summary of Related Work

As presented in Section 1.2, a theory of design should meet several qualifications. The works reviewed in this section are evaluated with respect to these requirements in this section. While some of the reviewed works meet some of these requirements, there is little existing work in which uncertainty is explicitly considered and tradeoffs between the artifact and process are modeled. Furthermore, most of the existing work limits the expression of preferences and beliefs due to a reliance on set theory. Therefore, a new theory is needed to simultaneously meet all of these requirements. In the next chapter, Rational Design Theory is introduced as a new theory that meets these requirements. In the next section, an overview of decision theory is presented as a foundation for Rational Design Theory.

2.2 DECISION THEORY AS A FOUNDATION FOR A NEW THEORY OF DESIGN

Decision theory is a normative approach to decision-making that builds upon axiomatic utility theory as a foundation (Howard 1966a). The three main components of a decision problem are decision alternatives, outcomes, and preferences. Decision alternatives are the different options from which the decision maker can choose. Outcomes are tied to a particular decision alternative and represent the predicted consequences that will occur if the decision maker chooses that particular alternative. Because of uncertainty, there are many possible outcomes, and the outcomes are usually represented as random variables. In addition, there may be multiple dimensions of outcomes such as cost and various performance metrics. Decision makers have preferences about the outcomes associated with each decision alternative, and they use these preferences to determine which alternative is the most preferred. According to decision theory, a rational decision maker should choose the decision alternative with the highest *expected utility*, where the expectation is taken across all possible outcomes of a decision alternative.

Decision analysis is a prescriptive method for implementing utility theory for complicated decisions (Howard 1968). The analysis is a systematic means for decomposing a decision problem into alternatives, outcomes, and preferences. This decomposition enables the systematic determination of the most preferred decision alternative. The decision-maker's beliefs about the outcomes that may occur are captured as (subjective) probability distributions, and his preferences concerning those uncertain outcomes are modeled with a utility function.

TABLE 2.2. REQUIREMENTS ANALYSIS OF EXISTING WORK AND DECISION ANALYSIS OF DESIGN PROCESS DECISIONS

Requirement	(E)GDT	FBS	Gero	DBD (artifact)	Coupled Design	C-K	Decision Analysis Applied to Process
Includes the generation and revision of specifications			x		x		x
Includes the creative process of generating new concepts						x	x
Explicitly includes uncertainty				x			x
No limit on the expression of preferences and beliefs				x			x
Incorporates the allocation of design phase resources as outcomes of design actions							x
Recognizes a tradeoff between product and process							x
Provides a means of evaluating design methods relative to one another							x

As mentioned previously, it is not immediately obvious how to apply decision theory in design. Howard's definition of a decision can help to identify the appropriate context for a design decision problem. Howard defines a decision as "an irrevocable allocation of resources" (Howard 1966a). Because resources are allocated when a decision is made and an alternative is chosen, the outcomes associated with each decision alternative should reflect this allocation of resources. As has been noted in previous work(Thompson, et al. in press), the resources that are allocated during the design phase are design *process* resources such as manpower, computing time, and money. Traditional formulations of decision-based design problems do not reflect the allocation of these process resources in the outcomes associated with decision alternatives. Rather, the outcomes that are modeled in the traditional perspective are outcomes related to the artifact only, such as performance metrics and manufacturing costs. Taking this artifact-centric perspective prevents the consideration of process-artifact tradeoffs. This kind of tradeoff is essential for understanding the end game of design processes.

Decision theory is an appropriate foundation for rational economic decision-making in the face of uncertainty. Engineering design as a process is a sequence of decisions about how to allocate resources to improve economic outcomes by reducing or adapting to uncertainty. Thus, decision theory is an appropriate foundation for a new rational theory of design. Decision analysis applied to a design process (rather than the artifact) is evaluated with respect to the requirements for a design theory in Table 2.2. As shown in the table, this new approach is capable of meeting the requirements, and this approach provides the foundation for RDT.

2.3 THESIS ROADMAP

In this chapter related literature has been reviewed. First, existing design theories were reviewed and evaluated with respect to the requirements for a new design theory. This review demonstrated that no existing design theories meet the requirements put forth in Chapter 1. In addition, an overview of decision theory and decision analysis was presented. This review showed that decision analysis of design process decisions is an appropriate foundation for a new design theory that incorporates economic tradeoffs between the designed artifact and the design process.

CHAPTER 3: RATIONAL DESIGN THEORY

In this chapter, Rational Design Theory is presented as a new normative framework for evaluating design methods. RDT meets the requirements identified in Chapter 1 and is an improvement over existing design theories reviewed in Section 2.1. RDT builds upon a foundation of decision theory, an overview of which is presented in Section 2.1.8.

Rational Design Theory is introduced in the next section. Thereafter, the RDT conceptual model is presented in Section 3.2 as a complement to RDT. In Section 3.3, characteristics of good design methods are discussed. Once the new ideas have been clearly defined and explained, they are related to the literature in Section 3.4.

In Section 3.5 RDT is critically reviewed with respect to the requirements and research questions. The chapter concludes in Section 3.6 with a look back at the thesis roadmap.

3.1 RATIONAL DESIGN THEORY

3.1.1 The Purpose of Design

I believe that a design process is sparked by the realization of the designer that his well-being could be improved by creating a new object, process, or piece of information. The improvement in well-being could be direct, if the designer designs and artifact for personal use, or indirect, if the designer hands the artifact specification over to another person in return for remuneration. The designer's realization for potential improvement of well-being is tempered by the fact that a design process takes effort; thus, to begin a design process, a designer must have a notion that his well-being could be improved and that this improvement will justify the effort of the design process. For the remainder of this work I refer to the products of a design process as *artifacts*. Some examples of artifacts include physical objects, works of art and literature, and processes.

DEFINITION 5 An *artifact* is anything produced through "human intelligence and effort" (Simon 1996, Baldwin, et al. 2000).

Given the realization of an opportunity, a designer's purpose in undertaking a design process is to improve his well-being. A new artifact is the means for obtaining the improvement. Since artifacts are produced through human effort and the purpose of design is to specify new artifacts, design is necessarily a human activity. The sequence of actions that comprise this activity of design is called a design process (ref. Definition 2).

There are many possible processes to specify a new artifact. For example, instructing someone simply to hack away at a chunk of wood will produce a new artifact; however, this new artifact is unlikely to improve the designer's well-being. To increase the likelihood of improving his well-being with the new artifact, it is desirable for the designer to take a more purposeful approach to its design. The knowledge required to select a particular design process over another while solving a certain design problem is embodied in a design method (ref. Definition 3). The goal of RDT is to shed light on which design methods are better than others and why.

3.1.2 Rationality

DEFINITION 6 The *outcome* of a design process is any modification of the state of the world, including the state of knowledge about potential artifacts, that is a consequence of executing a design process.

Different outcomes result in different levels of well-being. And therefore, the designer who aims to improve his well-being has different preferences for different outcomes. According to decision theory, the designer should choose the design method that leads to the outcomes that are most preferred. By definition, a *rational* decision maker will make this choice such that it is consistent with his beliefs (knowledge) and preferences.

DEFINITION 7 *Rational Behavior* is decision-making behavior that is internally consistent, that is, consistent with the beliefs and preferences of the decision maker.

Making rational decisions is difficult because it is impossible to know in advance what the future outcome of a given design process will be. The future outcome is uncertain. Choosing among actions under uncertainty is a problem that has been studied by von

1. Complete Ordering
For any (u, v) either $u \succ v$ OR $u \prec v$ OR $u \sim v$
2. Transitivity
For any (u, v, w) if $u \succ v$ AND $v \succ w$ THEN $u \succ w$
3. Continuity
For any (u, v, w) such that $u \succ w \succ v$, then for some α, $(0 < \alpha < 1$, $w \sim \alpha u + 1 - \alpha v$
4. Convexity
For any (u, v) such that $u \succ v$, then for any α, $(0 < \alpha < 1)$, $u \succ \alpha u + (1 - \alpha) v$
5. Combining
For any (u, v) , $(0 < \alpha\beta < 1)$ and $\gamma = \alpha\beta$, $\alpha(\beta u + (1 - \beta) v) + (1 - \alpha) v \sim \gamma u + (1 - \gamma) v$

FIGURE 3.1. AXIOMS OF VON NEUMANN-MORGENSTERN UTILITY THEORY (VON NEUMANN, ET AL.).[3]

[3] Note that other researchers have developed slightly differing sets of axioms that lead to similar conclusions (Marschak , Herstein, et al. , Luce, et al. , Berger). Also note that (u, v, w) *are outcomes.* (α, β) *are probabilities.* $u \succ v$ *indicates that outcome* u *is preferred over outcome* v. $u \sim v$ *indicates that outcomes* u *and* v *are equally preferred; the lottery "*$\alpha u + (1 - \alpha) v$*"* should be read as: "a combination of outcomes u, v with the alternative probabilities α, $1 - \alpha$."

Neumann and Morgenstern (von Neumann, et al. 1953). In von Neumann-Morgenstern (vN-M) Utility Theory axioms of “rationality” express characteristics of preferences that must be satisfied. In Figure 3.1, the first vN-M axiom states that the decision maker has preferences over all possible outcomes, and that the decision maker is capable of expressing these preferences. The second vN-M axiom states that preferences should be consistent and transitive. The remaining vN-M axioms concern the consideration of vN-M lotteries. The third vN-M axiom states that preferences should be continuous over a region: any lottery with two outcomes as possibilities can be reduced to an equivalent certain outcome. The fourth vN-M axiom states that preferences should be convex: if something is preferable, then a larger chance of receiving it should always be preferred to a smaller chance. The fifth vN-M axiom states that compound lotteries, or lotteries with a lottery as an outcome, can be reduced to a single lottery.

In RDT, it is assumed that rationality is desirable and that a designer should therefore strive to express preferences and act in a fashion that is consistent with the rationality axioms of vN-M Utility Theory. Von Neumann and Morgenstern proved that any decision maker whose behavior is consistent with the axioms of rationality has a real-valued utility function that is such that the behavior of the decision maker can be explained as maximizing the expected value of his or her utility function. Such utility functions thus provide a mathematical formalism for expressing rational behavior. A designer should therefore strive to maximize his or her expected utility.

3.1.3 Net Design Utility

During a design process, a designer gathers and organizes information about the artifact to increase the likelihood that when the artifact is produced it improves his well-being.

Whether or not the artifact improves well-being is determined to a large extent by the properties[4] of the artifact. The realized properties of the specified artifact therefore contribute significantly to the outcome of a design process. But well-being is also influenced by the resources consumed during a design process. These resources are valuable to designers, and could be used elsewhere if not dedicated to the design process. Thus, all other things being equal, designers prefer to use fewer of these resources during a design process. In previous work in utility-based design (Thurston 1991, Thurston, et al. 1994, Hazelrigg 1998, Thurston 2001), the use of design-process resources has not been included in the utility function explicitly. To distinguish RDT clearly from a utility that only accounts for the artifact perspective, this combined utility is hereafter referred to as Net Design Utility.

DEFINITION 8 *Net Design Utility* is a von Neumann-Morgenstern utility function that encompasses the preferences of the designer with respect to both artifact properties and the use of resources in the design process.

3.1.4 Normative versus Prescriptive

The explicit consideration of the cost of the design process is important because the designer must make decisions not only about the design artifact but also about the design process. When implementing normative decision theory in practice, the designer is forced to make some approximations. For instance, to take preferences into account mathematically, the designer must express his preferences in the form of a utility

[4] A formal definition of *property* is provided in Section 3.2.

function. Determining a mathematical expression for this utility function that is perfectly consistent with the designer's preferences across all possible outcomes would require answering an infinite number of preference elicitation questions. This is not possible in practice. The normative design theory must therefore be approximated by a *prescriptive design method* in which detailed guidance is provide on how and to what level of detail preferences should be elicited. Similar approximations must be made, for instance, for the elicitation and mathematical expression of beliefs and for the computation of the expected utility (e.g., How to compute the expected value? And if by using Monte-Carlo simulation, then how many samples? Etc.).

The question then is: Which prescriptive design method is best? In the previous section, it was argued that one should choose a rational method. However, as soon as any approximations or simplifications are introduced, rationality can no longer be guaranteed. Although one would hope that a good approximation will result in a design artifact that matches the preferences and beliefs of the designer closely, it is unlikely that it will be fully consistent.

This is related to the notion of Bounded Rationality introduced by Herbert Simon (Simon 1997). Simon uses the term 'Bounded Rationality' "to designate rational choice that takes into account the cognitive limitations of the decision maker—limitations of both knowledge and computational capacity" (Simon 1997).   The ultimate goal with RDT is not to describe how economic actors behave (as was Simon's intent) but to *prescribe* how designers can make good decisions in practice. Still, the recognition that there are bounds to rationality when considering the practical implementation of decision theory is important for prescriptive design methods also.

The purpose of Rational Design Theory is to provide a framework in which the question can be answered of "Which prescriptive design method is best?" In RDT, the cost of implementing rational decision making is taken into account explicitly in the NDU, so that one can determine which methods provide a good tradeoff between the cost of the design process and quality of the design artifact. An example of such an analysis can be found in (Thompson, et al. in press). However, RDT itself suffers from the same problem as normative decision theory: the maximization of the expected NDU requires significant resources and is not practically implementable, that is, not implementable as a practical design method.

Instead, one should think of RDT as a framework in which practical methods can be evaluated and compared. The methods being compared are prescriptive; the framework for comparison — RDT itself — is normative. Comparing prescriptive methods is an activity that can be performed by academics at a cost much higher than is justifiable for solving a single design problem. To support such comparison and gain more insight into the specific actions in design processes, a conceptual model of design is presented next.

3.2 A DECISION-THEORY-INSPIRED CONCEPTUAL MODEL FOR DESIGN

To compare different design methods based on their expected Net Design Utility (NDU), it is necessary to adopt a conceptual model of design in which NDU can be meaningfully defined. RDT therefore introduces also a conceptual model that allows us to identify characteristics of good design methods, characteristics a method must exhibit in order to approach the normative ideal as defined in RDT. Although conceptual models of design have been proposed by others (Yoshikawa 1981, Tomiyama, et al. 1987, Gero 1990, Umeda, et al. 1990, Hazelrigg 1996, Marston, et al. 1997, Hazelrigg 1998, Braha, et al.

2003, Krishnamurty 2006, Lewis, et al. 2006b, Hatchuel, et al. 2009), the model presented in this paper is different in the sense that it is consistent with utility theory and the notion of expected NDU. For example, some previous models do not incorporate uncertainty in design, and the notion of *expected* NDU is not meaningful unless utility is subject to uncertainty.

3.2.1 A Two-Level Perspective of Design Decisions

To decide ultimately on the selection of an artifact — an *artifact decision* — one must gather information about which artifact alternatives to consider and what the predicted outcomes and corresponding preferences are for these alternatives. This information gathering process is the result of applying a particular *design method*. At each step in a design process, the design method prescribes which design action to perform next. Selecting a design action to perform is itself a decision problem. The design method prescribes how to formulate and solve this decision problem.

A very simplistic design method could prescribe to enumerate all artifact alternatives, predict the outcomes for all alternatives, and then choose the alternative that is most preferred according to some decision criterion. For all but trivial design problems, the cost of enumerating and evaluating *all* artifact alternatives would be unacceptably large. Most design methods therefore consider a sequence of design actions, which together make up a design process. Allowing for a sequence of design actions is advantageous because it enables the decision maker to consider the information obtained from previous actions when determining the next action of a design process. By considering more (relevant) information when deciding on each step of a design process, promising artifact alternatives can be identified more efficiently. This approach is common in practice.

Consider for example the design of a car. Typically, designers will first use high-level abstract models of automotive powertrain concepts; based on the results obtained from these models, they will then refine the alternatives and focus on a smaller subset of alternatives for more detailed analysis. This is much more efficient than considering all the alternatives at a detailed level. However, when dividing a design process into a sequence of design actions, the designer needs to determine which actions to take and in which order to take them.

A *design method* defines how to choose the next action at each step of a design process. It is important to recognize that choosing the sequence of actions in support of an artifact decision involves additional decisions: *process decisions*. In decision theory, such a sequence of decisions is often modeled as a decision tree in which decision nodes (representing the choice of which action to take in a design process) are interleaved with chance nodes (representing, for instance, the a priori unknown outcomes of design analyses). In an ideal world (in which decision making is free and fully rational), one would explore the entire tree of possible sequences of design actions. In such a decision tree, each decision node inherits the maximum expected utility of its branches, each branch corresponding to a design action. Thus, the utility of a particular decision alternative depends on the outcomes of future decisions. Practical design methods differ from this idealized method in the sense that only a limited number of design actions are considered and that the decision criterion used to choose among them is a heuristic that at best aims to approximate the maximization of expected utility. Rarely is it justified to take the results of future decisions explicitly into account because of the large costs involved. However, the methods do share the characteristic that the choice of design action is based on the current state of information about the design artifact.

This discussion inspires a two-level perspective of design decisions, depicted in Figure 3.2. In this perspective, design process decisions are made throughout a design process to change the state of information of the artifact decision. In turn, the state of the artifact decision must be referenced when making process decisions to explore the potential benefits of design actions.

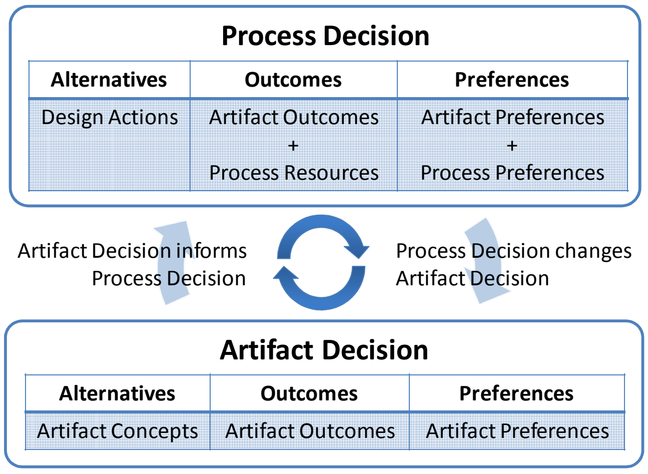

Process Decision
Alternatives
Outcomes
Preferences
Design Actions
Artifact Outcomes + Process Resources
Artifact Preferences + Process Preferences
Artifact Decision informs Process Decision
Process Decision changes Artifact Decision
Artifact Decision
Alternatives
Outcomes
Preferences
Artifact Concepts
Artifact Outcomes
Artifact Preferences

FIGURE 3.2. TWO-LEVEL CONCEPTUAL MODEL OF DESIGN

It is in the context of this two-tiered decision perspective that Rational Design Theory is framed. The RDT conceptual model frames the artifact decision in the context of *concepts*, *concept predictions*, and *concept decision criteria*. These three aspects correspond to the decision alternatives, outcomes, and preferences of the artifact decision. Although utility theory can be used in the artifact decision, it is not necessary to do so. The descriptive model of the information regarding the artifact decision is intended to be sufficiently general to model any decision-making process. In the higher-level process decision, the decision alternatives are design actions such as *synthesis*, *analysis*, and *evaluation*. The outcomes of these action alternatives are updated information states of the artifact decision. Obtaining these outcomes comes at a cost in terms of time, money, and other process resources. The designer has preferences about the outcomes of the artifact decision as well as preferences about expending resources to change the alternatives, outcomes, and preferences of the artifact decision.

When practical design methods are defined within this conceptual model, RDT can be used to compare the methods based on expected NDU. An example of such a comparison can be found in (Thompson, et al. in press). In the next sections, the two-level conceptual model in Figure 3.2 is further refined focusing first on the bottom layer, namely, the artifact decision.

3.2.2 Describing the Artifact Decision: The Artifact Information State

In the RDT conceptual model, the artifact decision is modeled in terms of the Artifact Information State (AIS). Although a structure is imposed for this information model, it is intentionally left as general as possible. It is intended to be sufficiently general so that most, if not all, design methods are supported.

Because the goal of a design action is to gather relevant information about the design artifact, each design action modifies the AIS. Conversely, the AIS model also allows us to identify the possible actions available to a designer in the process decision. These actions—*synthesize*, *analyze*, and *evaluate*—each are manifested by a change in the AIS. The artifact information state consists of three aspects: the specification of *concepts*, *predictions* of the performance of these concepts, and an evaluation *criterion* to rank the concepts with respect to each other. Each of these three aspects is described in terms of *properties* defined on the set of *artifacts*.

3.2.2.1 Artifacts and Properties

As noted previously, an artifact is considered to be anything produced through human intelligence and effort, including all human-produced physical objects and processes.

DEFINITION 9 The artifact set, S' , is the set of all artifacts that currently exist, have existed in the past, or will exist in the future[5].

DEFINITION 10 A *property* is any descriptor of an artifact. Mathematically, a property is a function defined over the artifact set: $p: A \to Y$, with $A \subset S'$ the domain of the property and with Y a topological space that is its range.

Examples of artifact properties are an artifact's function, structure, or behavior, but also any aspect of the manufacturing process intended to produce the artifact, or any other uniquely defined characteristic of the artifact. Some specific examples of properties are shown in Table 3.1. In general, properties have the following characteristics:

- A well-defined property associates a unique value with each artifact in its domain. Practically, it is impossible to enumerate for each artifact in the artifact set the corresponding property value. Instead, one typically describes the meaning of the property either in words or as the result of a measurement process. For instance,

[5] What is considered to be an artifact in RDT is identical to the term "entity" in General Design Theory (GDT) (Yoshikawa 1981); the artifact set corresponds to the entity set in GDT.

"the mass of an artifact in kg" is a property defined for any physical object. One could either assume that the semantics of the expression "the mass of an artifact in kg" are sufficiently precise to define the property, or one could define a measurement process for measuring the mass.

- It is common that properties are defined poorly or incompletely, resulting in ambiguity. For instance, a designer may refer to the fuel economy of a car. But unless he or she precisely defines how the fuel economy is measured, it could be interpreted in a variety of ways, resulting each time in a different value in the range, Y. Although a property associates a specific value with each artifact in its domain, in practice, one may not be able to determine exactly what that value is due to limitations in our ability to observe and measure properties.
- The domain, A, of a property may not extend over the entire artifact set. For instance, the property "mass in kg" is only defined for physical artifacts.
- Because a property can be *any* descriptor of an artifact, there are an infinite number of properties. Strictly speaking, one could argue that only a finite number of these properties are independent, for instance, the properties corresponding to the states of every elementary particle in a physical artifact, but the number of such properties is so large that it is still convenient to characterize it as infinite.

In RDT, properties are used in three different ways. First, properties are used to characterize specific artifacts. Second, constraints on properties are used to specify concepts, and third, properties are used to characterize a designer's beliefs and preferences about possible outcomes of concepts. These uses of properties are discussed in more detail in the following sections.

TABLE 3.1. EXAMPLES OF PROPERTIES

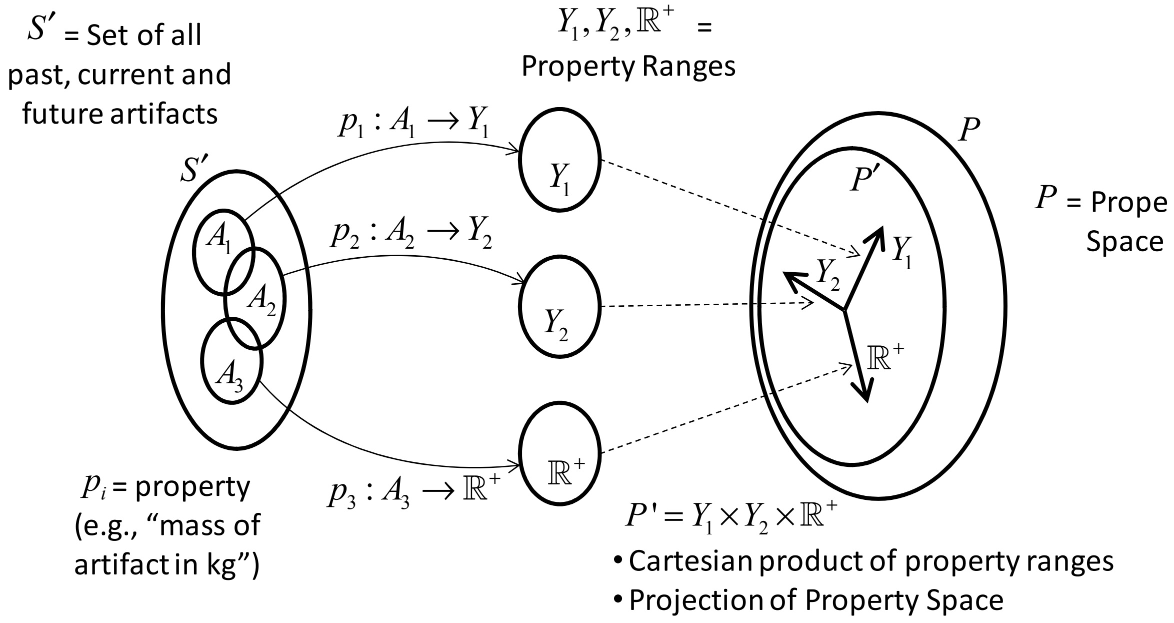
Description	Range	Sample Values
Mass in kg	$\mathbb{R}^+$	0 kg, 50.4 kg
Position of point A relative to axes B	$\mathbb{R}^3$	(0, 1.2, 7); (-3.2, 4, 10)
Number of engine cylinders	$\mathbb{Z}^+$	0, 1, 2, …
Probability of engine failure	$[0,1]$	0, 0.75, 1
Plane angle between cylinders in a V engine	$\mathbb{R}$	0, 0.8, $\pi/2$
Satisfies the function "convert rotational motion to linear motion"	Boolean $\{0,1\}$	0 (false), 1 (true)
Type of engine	Discrete space of enumeration literals	Internal combustion, electric-mechanical hybrid, hydraulic-mechanical hybrid

3.2.2.2 Property Spaces

DEFINITION 11 The *property space*, P, is the collection of the topological spaces of all properties. Mathematically, this space corresponds to the Cartesian product of all the property range spaces, Y.

The property space is depicted graphically in Figure 3.3. When describing artifacts, designers may use a large number of properties. Thus, artifacts map into a multi-dimensional property space. However, not all properties are relevant in a particular context, and it is therefore useful to limit oneself to a finite number of properties, a modeling process that is typically called *abstraction* (Barker 2005). Mathematically, this abstraction corresponds to a *property space projection,* P', the Cartesian product of the ranges of all the properties that are being considered. For example, as shown in the following equation, P', could be the Cartesian product of mb Boolean properties, mr real-valued properties, and mz integer-valued properties.

$$P' = \{0,1\}^{mb} \times \mathbb{R}^{mr} \times \mathbb{Z}^{mz} \tag{3.1}$$



S' = Set of all past, current and future artifacts
S'
A_1
A_2
A_3
p_i = property (e.g., "mass of artifact in kg")
$Y_1, Y_2, \mathbb{R}^+$ = Property Ranges
$p_1 : A_1 \to Y_1$
$p_2 : A_2 \to Y_2$
$p_3 : A_3 \to \mathbb{R}^+$
Y_1
Y_2
$\mathbb{R}^+$
P
P'
P = Prope
Space
Y_1
Y_2
$\mathbb{R}^+$
$P' = Y_1 \times Y_2 \times \mathbb{R}^+$
• Cartesian product of property ranges
• Projection of Property Space

FIGURE 3.3. PROPERTIES AND THE PROPERTY SPACE

Each artifact is represented by a single point in a projection of the property space that includes only those properties relevant to that artifact. Conversely, a point in the property space projection is an abstraction of possibly an infinite number of artifacts. Since only a finite subset of all possible properties is included in the property space projection, some artifacts (that differ from each other only in properties that have been abstracted away) map onto the same point in the property space projection. A complete description of an artifact would require an infinite number of properties and is therefore not practically feasible.

3.2.2.3 Concepts

DEFINITION 12 A *concept* is a partial specification for a hypothetical artifact. Mathematically, a concept, C, is defined as a subset of a property space projection: $C \subseteq P'$

First and foremost, a concept is a *specification*. It specifies an alternative being considered in an artifact decision. At the beginning of a design process, the concepts being considered are often very abstract and vague. Such concepts specify very broad classes of products. For instance, a concept defined as "a vehicle with four wheels and an internal combustion engine" would encompass most of the cars and trucks on the market. As the design process progresses, additional properties of the artifact are specified, leading to concepts that specify ever smaller subsets of the property space. When the design process ends, the final concept selected by the decision maker tends to be defined in quite a bit of detail. When the designer relinquishes control over the decision-making process, a concept specification can be transferred to a manufacturer or subcontractor to complete development and manufacture of the artifact. At this point, the concept may be phrased as requirements in a contractual agreement. Note, however, that requirements in this context should not be interpreted as an expression of preferences. That is, requirements in the RDT conceptual model should be stated as follows: "To be a part of the concept, the artifact shall <requirement>". Thus, the requirement only defines that which is included in the concept. This is opposed to the traditional statement of a requirement as an expression of preference, as in the following phrase: "To be acceptable, the artifact shall <requirement>".

A concept is a *partial* specification. The qualifier "partial" has been included in the definition for emphasis, but is strictly speaking redundant. Regardless of how much detail is included in a concept definition, it is impossible for every property to be specified. Any property that is not explicitly constrained is considered to be left completely unconstrained. Consequently, although a concept may be fully specified in a property space projection, P', it always corresponds to a non-singleton subset of P, the full property space. As a result, there is always some ambiguity in any concept specification.

A concept is a specification for a *hypothetical* artifact. The artifact is hypothetical in the sense that it may never be realized. First, the artifact is only one of many concepts under consideration in the artifact decision. It may not be the concept that is ultimately selected by the designer and may therefore never be realized. Second, the artifact is hypothetical because it may not be realizable. It is important to keep in mind that a concept is defined as a subset of the property space, not the artifact set. It is only an implicit reference to a subset of the artifact set through the inverse mapping of the property functions. For some concepts, the subset of the artifact set that maps onto the concept may be empty, meaning that there are no artifacts that simultaneously satisfy all property specifications. For example, while there are artifacts that are very light, and artifacts that are very strong, there are no artifacts are both very light and very strong. Therefore, the concept of an object that is both very light and very strong maps back through the inverse mapping to the empty set in the artifact space. Even if artifacts exist that map into the subset of the property space encompassed by a given concept, then it may still not be possible to realize any of them. Limitations of current technology may exclude large portions of the artifact set, S' , including the portion containing the desired artifacts. Even if a desired artifact is technologically feasible, it may be difficult to identify it. The challenge of design is that the inverse mapping from property space to artifact set is implicit and cannot be easily resolved. Even worse, it may not be verifiable whether a given artifact maps onto a concept given that the actual behavior of an artifact cannot be determined with certainty. This issue will be further discussed in Section 3.2.2.5 on *concept predictions*.

3.2.2.4 Concept Refinement

Throughout a design process, new concepts are generated as *refinements* of previously defined concepts. As a starting point, the most general concept is defined by the *empty specification*, a complete lack of restrictions on any properties. All artifacts satisfy this specification, thus, the inverse property mapping applied to the empty specification maps it to the entire artifact set, S'. Throughout a design process, the empty specification is gradually refined by restricting the values that properties are allowed to take on. This can happen either by further restricting properties that have already been constrained, or by placing constraints on additional properties, expanding the dimensionality of the property space projection, P', as necessary. Throughout a design process, each new concept being added tends to be more and more constraints, but the total number of concepts potentially being considered in the artifact decision expands.

One could illustrate this by a tree in which each node represents a concept, as is shown in Figure 3.4. Moving from the root of the tree to the leaves corresponds to *concept refinement*: a parent-concept is subdivided into child-concepts by adding additional constraints in the property space projection, which results in subsets of the parent. The arrows represent specialization relations. For example, the root concept, C_1, in Figure 3.4 is subdivided into two additional concepts, C_2 and C_3, by specifying additional constraints on the type of energy conversion. As the design process progresses, the number of concepts (i.e., the number of nodes in the tree) grows, while each new concept tends to be more and more constrained. Note, however, that adding concept refinements does not eliminate the parent concepts from consideration. At any point in a design process, it is possible to backtrack through the tree and start refining concepts that had been previously left unexplored.

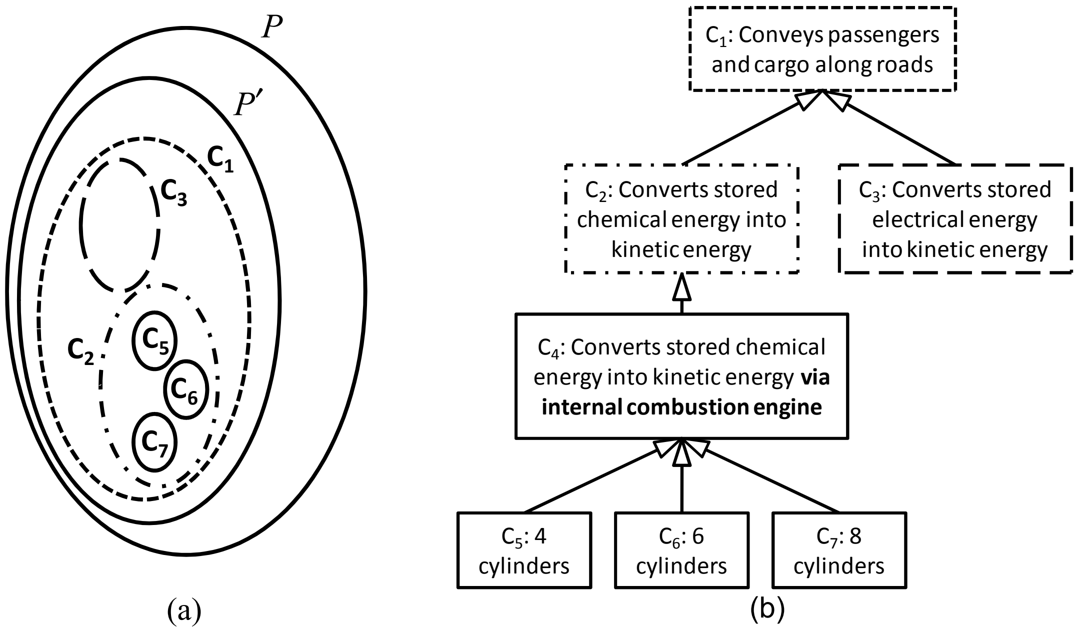

P
P′
C1
C3
C2
C5
C6
C7
C1: Conveys passengers and cargo along roads
C2: Converts stored chemical energy into kinetic energy
C3: Converts stored electrical energy into kinetic energy
C4: Converts stored chemical energy into kinetic energy via internal combustion engine
C5: 4 cylinders
C6: 6 cylinders
C7: 8 cylinders
(a)
(b)

FIGURE 3.4. CONCEPT REFINEMENT DEPICTED (A) AS SUBSETS OF A P' AND (B) IN A TREE STRUCTURE

Although concept refinement often begins with functional specification, proceeds to behavior specification, and ends with structural specification, it is not required for refinement to happen in this order. Sometimes structural details may need to be specified early in a design process, for instance, to meet regulations or other external requirements. Another example is the case of variant design of an existing product. In this case it may make more sense to begin with the existing structural specification and specify new concepts by relaxing some of the constraints on structural properties. For many cases, specifying these structural details early in the design process reduces uncertainty in the artifact outcomes, which is likely to lead to an increase in the expected NDU.

3.2.2.5 Concept Predictions

To determine whether a concept is worthwhile considering, it is necessary to predict the outcomes resulting from selecting the concept. When selecting the concept, the intention is to realize an artifact based on the concept's specifications. For a physical artifact, one could hand the concept to a manufacturer and ask him to build the corresponding artifact. The concept predictions should reflect the designer's beliefs about the artifact the manufacturer will produce.

DEFINITION 13 A *concept prediction*, X, for a concept, C, is a mathematical characterization of the designer's beliefs about the properties of the artifact that will be realized when the concept C is used as a specification.

The concept prediction is shown graphically in Figure 3.5. Mathematically, a concept prediction is represented as probability triple, $(P', \mathcal{F}, X)$, in which the property space projection, P', is the sample space; the Borel σ-algebra, $\mathcal{F}$, is the set of events defined over the sample space; and $X: \mathcal{F} \rightarrow [0,1]$ is the probability measure which maps every event in $\mathcal{F}$ onto a probability expressing the designer's belief about whether the event will occur. Specifically, for a given concept, C, the probability associated with an event defined by a subset D of the property space projection, expresses the designer's belief that the artifact produced based on the concept specification, C, will have property values that lie within D. Practically, X can be represented by a mixed discrete-continuous, multivariate cumulative distribution function.

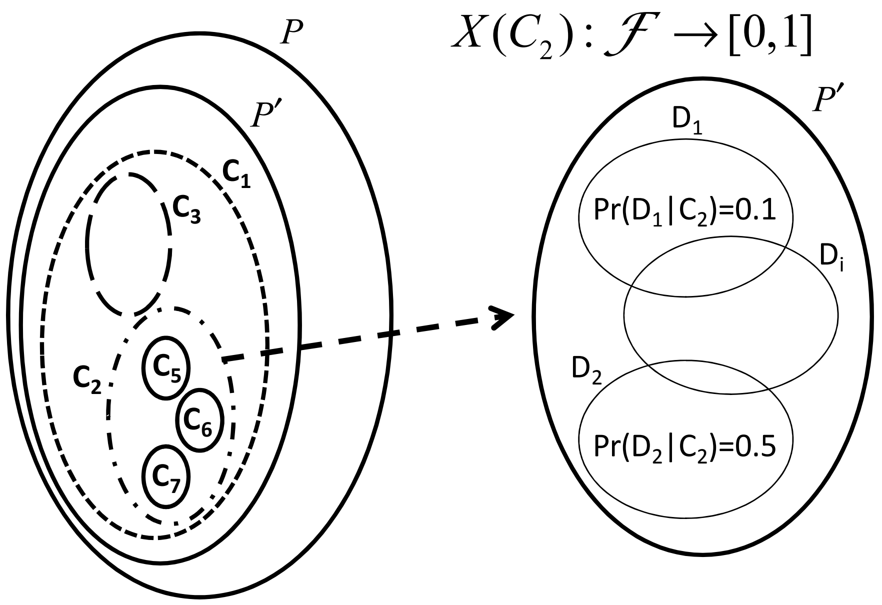


FIGURE 3.5. THE CONCEPT PREDICTION

This definition of a concept prediction is consistent with a subjective view of probabilities, characterizing a decision maker's willingness to act (de Finetti 1964, 1974). Since the concept prediction will be used by the decision maker to guide the selection of his or her actions, an operational interpretation of probability is appropriate. Although the predictions are subjective, a wise decision maker will always adopt beliefs consistent with objective scientific knowledge. In addition, the use of probability theory to model predictions requires that, to avoid irrational behavior, predictions be coherent in accordance with Kolmogorov's axioms (Kolmogorov 1956). Note that since deterministic values are special cases of probabilistic values, they are also covered in the modeling formalism for concept predictions.

Because the predictions ultimately guide the artifact decision, designers must make predictions about any property that contributes to a *concept decision criterion*[6], be it a

[6] The term *concept decision criterion* is defined in the next section.

comprehensive measure of effectiveness, an overall performance measure, or a utility. Although there are an infinite number of properties over which to characterize a concept, not all are used to predict utility of the artifact. In most product development processes the utility of the artifact is measured by profit. It is unnecessary and a waste of resources to predict the possible outcomes of properties which do not contribute to the prediction of artifact utility.

Both concepts and concept predictions are defined over the property space projection, P'. However, rather than *specifying* values for these properties, concept predictions capture the *possible* values of the properties the realized artifact may exhibit. Both the concepts and their associated predictions are often expressed in terms of the same properties, but concepts are expressed as subsets of property space projections while concept predictions are expressed as probability distributions over property space projections. The predictions are always uncertain and/or ambiguous and differ from the concept specifications for a variety of reasons.

First, there is uncertainty related to the interpretation of the concept specification by the manufacturer. Because a concept represents a possibly large set of artifacts, it is not known which one of these artifacts will be targeted by the manufacturer. Especially for concepts considered in the early phases of design, the ambiguity may be large.

On the contrary, rather than being ambiguous, a concept definition could also have been overly restrictive, so that the manufacturer is unable to meet all the constraints. Again, the actual values of the properties obtained if one were to manufacture an artifact based on such a concept specification are quite uncertain.

In addition, we lack perfect knowledge about physics, economics, and other disciplines, so that future events cannot be predicted with certainty. Similarly, manufacturing processes cannot be perfectly controlled, resulting in some inherent uncertainty in the artifacts produced. Thus, even if the concept were to refer to a unique artifact, the properties of that artifact could not be predicted with certainty. It is therefore crucial to make a clear distinction between the specified property values of the concept and the predicted property values of the resulting artifact.

It is important to note that the structure of the artifact must be also be included in the concept prediction. The structure of the artifact is uncertain because of the ambiguity in the specification and variability in manufacturing processes. In other design theories, (e.g.,(Gero 1990)) the uncertainty in artifact structure is not recognized even when actual behavior of the artifact is distinguished from expected behavior. If one were to use a single representation for the artifact structure, it would be impossible to distinguish between the specification, the prediction and the ultimate realization of the artifact structure.

Predictions may be based on both models and direct elicitations of expert opinion. Early in a design process when there is significant ambiguity in the concepts, it may be difficult to formulate models of artifact behavior that are representative for the entire concept, so that predictions may best be directly elicited from experts. As concepts are refined and more properties are specified, it becomes possible to develop better and better predictions of the outcomes. Better predictions can be developed because the ambiguity of the large set of artifacts is reduced, making it possible to develop more detailed models of the underlying physics, economics, etc.

3.2.2.6 Concept Decision Criterion

To make a selection from several specified concepts, the concepts must be ranked with respect to each other. Recall that concepts are the alternatives in the artifact decision. These alternatives must be ranked on the basis of their predicted outcomes. In the RDT conceptual model, the concept decision criterion is a means for comparing concepts with each other on the basis of predicted outcomes.

DEFINITION 14 A *concept decision criterion* is a measure of the desirability of a concept. Mathematically, a concept decision criterion is defined by an evaluation function that maps a probability triple, representing a concept prediction, onto the real axis: $E{:}\,(P', \mathcal{F}, X) \to \mathbb{R}$.

Early in design, the decision criterion may be an informal, direct judgment of the designer, such as a ranking of concepts from best to worst. As the design process progresses, more formal evaluation models are usually formulated that assign a score for each concept based on mathematical predictions of outcomes. Examples of more formal evaluation models are constrained optimization (Mistree, et al. 1990, Bras, et al. 1993), expected utility (Thurston, et al. 1994, Hazelrigg 1998), Reliability-Constrained Optimization (Rao 1992), and Cost-Constrained Optimization (Rao 1992). Depending on the formulation of the evaluation function, the decision rule may be to take the concept with the maximum decision criterion value (e.g., for utility or reliability maximization) or the minimum decision criterion value (e.g., for cost minimization). This mapping from a probability triple onto the real axis is depicted in Figure 3.6.

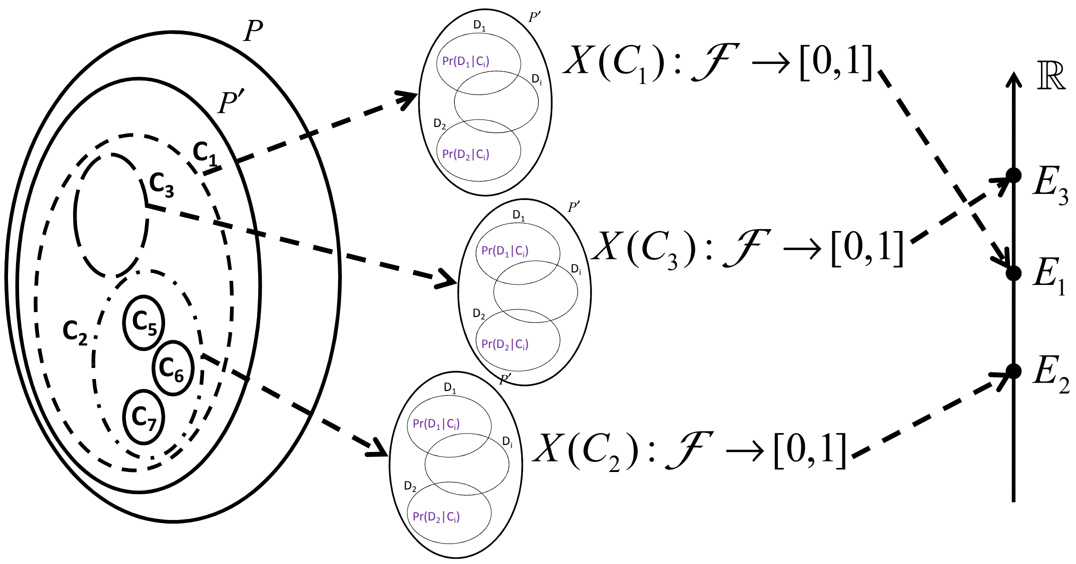

P
P′
C1
C3
C2
C5
C6
C7
D1
D2
Di
Pr(D1|Ci)
Pr(D2|Ci)
$X(C_1): \mathcal{F} \to [0,1]$
$X(C_3): \mathcal{F} \to [0,1]$
$X(C_2): \mathcal{F} \to [0,1]$
$\mathbb{R}$
E_3
E_1
E_2

FIGURE 3.6. THE CONCEPT DECISION CRITERION DEFINED BY EVALUATION FUNCTION, *E*

The concept decision criterion is real-valued, representing, for example, expected utility, cost, or reliability. In the case of a strictly constraint-based evaluation, the criterion value is a simple true or false, meaning that the concept either does or does not meet the constraints. For simplicity, it is assumed that this is expressed mathematically as a mapping onto either the real value 1 or the real value 0. The application of the evaluation function is the design action known as *evaluation*.

Like concept predictions, the concept decision criteria can be updated whenever the evaluation function is refined and reformulated throughout a design process. As mentioned above, decision criteria early in a design process are often informal but direct judgments by the designer. As the design process progresses, the evaluation model is often refined as a parametric function, an objective function, or a logic statement. Evaluation models are often specified or assumed by design methods; for instance, robust design methods assume an evaluation model based on a weighted sum of mean-on-target and minimized variation. Although evaluation models can be reformulated during a design process, it is important to note that the decision criteria are linked to the evaluation model used to produce them. Thus, when an evaluation model is reformulated, previously evaluated concepts will likely need to be re-evaluated using the new evaluation function.

3.2.2.7 Summary of the Artifact Information State

The descriptive model of the AIS consists of three aspects: concepts, concept predictions, and concept decision criteria. Each of these aspects is described in terms of properties. Mathematically, the AIS consists of the set of all concepts that have been specified so far, the set of all property predictions (one multivariate random variable per concept) that have been generated, and at most one decision criterion per concept. This AIS is continually changing during a design process as concepts are added, predictions are generated, and those predictions are evaluated to generate decision criteria. These changes in the AIS are the result of design actions, which are discussed in the next section.

3.2.3 Describing Design Methods as Process Decisions

Returning briefly to the two-level perspective of design decisions, it has been argued that one must consider not only the artifact decision as described in the Artifact Information State, but also *process decisions* as prescribed in design methods.

To allow for design methods to be analyzed it is necessary to frame prescriptive design methods as design process decision problems. That is, the selection of a design action is discussed in terms of alternatives, outcomes, and preferences. In this section, the decision analysis of design process decisions is presented as a means to qualitatively and quantitatively evaluate prescriptive design methods. Decision analysis is a prescriptive method for implementing utility theory for complicated decisions (Howard 1968). The analysis is a systematic means for decomposing a decision problem into alternatives, outcomes, and preferences. This decomposition enables the systematic determination of the most preferred decision alternative. The decision maker's beliefs about the outcomes that may occur are captured as (perhaps subjective) probability distributions, and his preferences concerning those uncertain outcomes are modeled with a utility function.

Decision analysis is one method for applying decision theory to a complex decision problem. Although the RDT conceptual model does not require the use of decision analysis to apply decision theory, decision analysis is used here to provide order to the discussion.

3.2.3.1 Alternatives: Design Actions

The first step in modeling a decision is to identify alternatives from which to select. In a design process decision problem, decision alternatives are the design actions that change the AIS. Alternatives also include the various ways of ending design: selecting a concept for manufacture or project cancellation. All other design actions can be characterized as one of the following three classes: synthesis, analysis, and evaluation. Synthesis is the generation of new concepts. Analysis is the application of knowledge and beliefs to predict the outcomes of concepts. Evaluation is the generation of a decision criterion to rank concepts. Each action impacts the AIS in its own way.

DEFINITION 15 A *synthesis action* is a design action that results in the generation of one or more new concepts.

Because concepts are defined as constraints on the allowed property values, to synthesize is to generate constraints. Refining a concept is the addition of constraints to additional properties or the reduction of the extent of existing constraints. As mentioned previously, refinement often proceeds hierarchically, creating a tree of concepts, as depicted in Figure 3.4. In this figure, a vehicle concept with an internal combustion engine can be refined by specifying the number of cylinders in the engine, which is the addition of a constraint on an additional property. Or, if the number of cylinders had previously been specified as a range, the concept could be refined by specifying a single value for the number of cylinders.

Synthesis actions change the AIS by adding concepts to the AIS. When a concept is refined by adding a constraint, the new concept including the additional constraint is added to the AIS; the old concept is not modified or removed. Maintaining these previous concepts and any associated predictions and criteria supports backtracking if the newly synthesized concept turns out to be worse than expected. At any point in a design process, any of the concepts can be further refined with a synthesis action.

DEFINITION 16 An *analysis action* is a design action that results in a new or updated concept prediction.

Some examples of analysis actions are the expression of beliefs about uncertain events, such as the properties of a new material, and the composition of physics-based models to predict artifact performance, such as a model of the mechanical behavior of that new material to predict the stresses in an artifact component. Analysis actions result in additional prediction information that can then be incorporated in the AIS. This is done by updating the random variable defined over property values corresponding to the current concept. From a practical perspective, this transformation usually takes place through a sequence of models.

DEFINITION 17 An *evaluation action* is a design action that results in the generation or updating of a concept decision criterion.

Evaluation has two parts: establishing the preference model (defining an evaluation function) and computing a concept decision criterion for a specific concept by executing that model. Evaluation may be informal, especially early in design, but is usually achieved by applying a model such as an objective function or other rule. Evaluation also includes the act of revising an evaluation function, a process that could also be called reformulation. This act amounts to stating (the same) preferences in terms of other attributes, as it is unlikely for a designer's preferences themselves to change. Restating preferences in this manner is done to make the approximation of the designer's true preferences more precise as more detail is incorporated in concept specifications and predictions.

Evaluation actions change the AIS by associating a scalar, deterministic criterion value with a concept based on the random variable prediction. In utility theory, for example, this is achieved with the expectation operator. An evaluation model for a constrained optimization could be achieved by augmenting the objective function with a penalty function.

3.2.3.2 Outcomes of Design Actions

Having identified the decision alternatives in a design process decision problem, the second step of decision analysis is to predict the outcomes of these alternatives. In the previous section, the changes in the AIS associated with each action (synthesis, analysis, and evaluation) were discussed. These changes in the AIS are only one aspect of the outcomes of design actions. The other important aspect concerns the consumption of design phase resources such as time, money, manpower, and computing cost. In anticipation of performing a design action, designers will not know the outcome of a particular analysis or synthesis, and they may not know how long an action will take or how much money will be spent. As a result, the outcome of a design action is uncertain, but it is not entirely unknown. Given the type of the design action (synthesis, analysis, or evaluation), designers will know the manner in which the AIS will change. For example, a synthesis action will result in the addition of a new concept or concepts. As for the use of resources, while the cost may not be known with certainty, an estimate can usually be obtained. When models are used for analysis and evaluation actions, designers may be able to predict more accurately the outcomes of these actions based on knowledge of the models themselves. For example, in previous work, the outcomes of simultaneous analysis and evaluation actions were modeled and computed using Bayesian updating (Thompson, et al. in press).

Even in the earliest, most informal stages of design, actions are undertaken with at least some beliefs about what the outcomes may be. These beliefs are what lead a designer to choose one design action over another, even if the beliefs are not formally captured and the decision process is ad hoc. Often, the motivation for a particular design action may simply be to enable another design action, such as synthesizing a new concept in order to be able to analyze and evaluate it. While it is true that designers have beliefs about these outcomes, more research is needed to determine how best to elicit and structure these beliefs mathematically.

3.2.3.3 Preferences about Outcomes of Design Actions

The third step of decision analysis is to apply preferences to the predicted outcomes. As mentioned in the previous section, the outcomes of design actions come from two sources: consumption of design phase resources and changing the AIS. Designers have preferences about both of these aspects. Obviously, designers generally prefer lower consumption of resources, but there is often a tradeoff between the various types of resources, such as time and manpower. Designer preferences about changing the AIS are less intuitive. As mentioned earlier, designers generally do not have direct preferences about a particular state of information. Rather, designers care about the outcome of the artifact decision. Specifically, they prefer to maximize the benefit to be gained by producing an artifact. Therefore, preferences about the state of information of the artifact decision come directly from the artifact decision itself, assuming one were to stop designing and choose the best concept immediately. The utility of the state of information is thus the maximum of the utilities of the synthesized concepts. Although it may seem that few concepts have been mapped through predictions to the criterion in the early stages of design, designers often have informally analyzed and evaluated a concept even at the earliest stages. This informal assessment forms the basis for decisions about design actions early in a design process.

Because the preferences with respect to the product decision come from the evaluation of artifact concepts, designers do not realize the benefits of newly synthesized or analyzed concepts until they have been evaluated. This may make synthesis and analysis actions look bad initially because they seem to have only a cost and no benefit. The benefit is apparent when evaluating the new concepts, which often has a very small cost itself. Sometimes designers may cease analysis of a concept after one bad result without formally evaluating the concept. This is again an instance of an *informal* evaluation on the basis of a single attribute.

These two dimensions (resource consumption and artifact utility) comprise the Net Design Utility. Because the artifact utility is usually profit-based, it is convenient to combine resource consumption and artifact utility on the basis of their effect on the profit. This is not required, however. In accordance with decision theory, they should be combined based on the fundamental objective of the project decision-maker.

3.2.3.4 Evaluating and Comparing Design Methods

Design methods can be viewed as algorithms for making design process decisions; they prescribe decision rules for determining design actions or sequences of actions. Decision analysis can be used to evaluate these decision rules by calculation of the expected NDU. Methods for computing expected utility include forward simulation and backward induction. Both methods amount to averaging the utilities of all the possible outcomes for a particular alternative. This is usually achieved through a numerical simulation of outcomes and provides a quantitative measure of the quality of one design action over another. Although analytical solutions can sometimes be derived for problems with mathematically well-defined outcomes and preferences, analytically solving for expected

NDU is not possible in a general sense. Though quantitative comparison of decision alternatives is only possible for very specific problems, decision analysis does enable qualitative evaluation of design methods in a more general sense through the decomposition of alternatives, outcomes, and preferences.

3.3 CHARACTERISTICS OF GOOD DESIGN METHODS

In the previous sections, RDT was presented as the application of normative decision theory to decisions about design processes with the explicit inclusion of design process costs. In presenting this normative theory, it is acknowledged that approximations and assumptions are necessary in order to arrive at practical design methods, but these practical methods can be evaluated by comparison to the normative ideal of rationality. Combining the normative perspective of RDT with the decision-theory-inspired RDT conceptual model, it is now possible to arrive at some characteristics that good design methods should exhibit. A "good design method," is one that approaches RDT's normative ideal of maximizing expected NDU.

The method is a close approximation of rational decision making applied to design process decisions. Since RDT proposes to evaluate prescriptive methods based on expected NDU, any method that does not approximate normative decision theory will not maximize expected NDU. This does not mean that normative decision theory must be explicitly applied in the design method. In fact, given the significant resources necessary to implement normative decision theory for even small decision problems, it is unlikely that an application of normative decision theory would be preferred by a designer. Instead, heuristics or simple decision rules can be used, which will be much more practical than implementing normative decision theory for each design action. The *outcome* of the heuristics or decision rules, however, should be as close as possible to the outcome if normative decision theory had been applied with appropriate beliefs and preferences.

Heuristics to guide a design process towards maximum expected NDU. Achieving the ideal of maximizing expected NDU requires negotiating a difficult trade-off. It has been recognized here that making decisions about the artifact requires resources that should be taken into account in the assessment of the NDU. Similarly, making decisions about a design process (i.e., applying a design method) also requires resources. These resources again, should be accounted for in the NDU. In (Thompson, et al. in press), the authors explored the selection of analysis actions in a design process in which two artifact alternatives and two engineering analyses were considered – a simple example to illustrate the use of Value of Information as a heuristic for guiding the selection of an appropriate analysis model for design. From this simple example, it is clear that performing a complete analysis to determine which of all the possible design actions maximizes expected NDU at each step in a design process will require far more resources than is justifiable. Therefore, a good design method will need to be based on *heuristics* to guide the selection of appropriate design actions.

Design methods such as the systematic design method of Pahl and Beitz (Pahl, et al. 1996) suggest that design proceeds through phases of conceptual, embodiment, and detailed design, each with their own set of design actions that should be considered. Although such heuristic guidance is useful, it is qualitative and rather vague. The phases are not precisely delineated and do not translate into specific design actions. Such design methods provide heuristics in terms of the order in which high-level design actions should be considered (e.g., which types of constraints to add in order to refine concepts, or which types of analyses to perform), but no quantitative guidance for how many alternative concepts to consider or how detailed an analysis to perform. Nor is it made clear in quantitative terms how much effort should be allocated to each phase.

A key challenge that will need to be addressed in future work is to define more precise and quantitative heuristics for particular classes of design problems to help identify which design actions to use at each step in a design process. The role of RDT is then to provide a validation framework in which the quality of the heuristics can be assessed.

A stopping criterion based on NDU. In previous work on design theory, it was suggested that design should end when a fully specified artifact has been reached (Yoshikawa 1981, Tomiyama, et al. 1987, Gero 1990). From the discussion on properties and concepts in Section 3.2.2, it is clear that it is impossible to fully specify an artifact because an artifact can be characterized by an infinite number of properties. It is therefore meaningless to suggest that design should end when a fully specified artifact has been reached. Instead, RDT suggests that design should end when the maximum expected NDU has been reached, or in other words, when there are no further design actions that increase the expected NDU. This point can be reached for several different reasons.

Commonly, as a design process progresses, one reaches a point where the artifact has been specified to such a level of detail and where its performance has been predicted with such a level of confidence that any further concept refinement or analysis adds little value — less value than the cost of refinement of analysis. At that point, design should end.

Another common scenario is one in which the utility of the artifact decreases as time progresses due to loss of market share or competitive pressures on price (Lee, et al. 2010a). Again, one reaches a point where a delay in the product launch would reduce the artifact's utility more than the increase resulting from further design actions. Even though the artifact may be far from perfect, it would be wise to discontinue further design actions and launch the product. Design should end.

Design should also end when the expected NDU of all currently considered concepts is less than the expected NDU of cancelling the project. Such a point could be reached when analyses or tests reveal that the promising concepts that were considered previously do not quite live up to the expectations or are too risky.

Finally, design could also end when a designer at an original equipment manufacturer (OEM) has specified all the subsystems in sufficient detail that the he or she is confident that suppliers will deliver implementations of the specified subsystems that, when integrated, produce an acceptable artifact. If further refinement of the subsystem specifications costs additional resources or unnecessarily increases the cost of the subsystem itself beyond the benefit obtained from the artifacts performance improvement, then design should end.

It is important to recognize that, in all these scenarios, design ends when a balance between process and artifact utility is reached. Each time, the scenario can be explained in terms of maximization of expected NDU: Design should end when the benefit of each possible remaining design activity in terms of improving the artifact no longer outweighs the negative (cost) consequences of the activity.

An explicit consideration of uncertainty. Uncertainty in the predictions of the performance of design concepts is one of the factors that make design challenging. Comparing different design concepts and choosing appropriate design actions is difficult when the decision maker does not know in advance what the consequences of a particular choice will be. In the RDT conceptual model, uncertainty is accounted for in the concept predictions in terms of random variables. Maximizing expected NDU is then an expression of the decision maker's preferences that takes into account both uncertainty and risk attitude.

However, in practical design methods, using random variables and assessing utility functions requires resources. This use of resources needs to be considered in the NDU. For instance, when predicting the overall utility of a particular artifact, one may use uncertainty quantification techniques such as Monte-Carlo simulation (Fishman 1996) or polynomial chaos expansion (Ghanem, et al. 1990). The accuracy of such uncertainty quantification techniques depends on the number of samples or the order of the approximation polynomials. As the number of samples or the polynomial order increases, the accuracy improves but the required computational resources increase also. Depending on the context of the particular design problem, an appropriate uncertainty representation should be chosen, a representation that balances the cost of computation with the cost of under- or over-estimating the uncertainty. The maximization of expected NDU can be used to guide the selection of an appropriate representation. In future work, it would be interesting to create computational experiments in which design methods with different uncertainty representations are compared, taking into account not only the benefits of more explicit uncertainty representation but also the costs of uncertainty quantification. Uncertainty representations to consider could range from deterministic models with safety factors, to models based on first and second distribution moments, to general Monte-Carlo simulation, or even extensions of probability theory such as Dempster-Shafer theory (Shafer 1976) or imprecise probabilities (Walley 1991). RDT provides a framework for comparing the quality of design methods that use these uncertainty representations. According to RDT, the criterion for comparison should be the maximization of expected NDU.

An explicit consideration of risk. The discussion on uncertainty is also relevant to the consideration of risk. How best to represent uncertainty is determined in part by the consequences of the uncertain events and hence the risk involved. In the design literature, risk has been considered primarily from the perspective of the artifact. In (Lee, et al. 2010b), an overview is provided of how different design methods account for uncertainty and risk. Robust Design (Taguchi 1986, Taguchi, et al. 1990), Risk-Informed Design (Tumer, et al. 2005), and Reliability-Based Design (Rao 1992) are reviewed and interpreted in the context of Utility Theory. The authors demonstrate how each of these methods has an equivalent representation in terms of maximizing expected utility.

However, this consideration of risk from the artifact's perspective is too limiting. One must also consider the mitigation of risk through the use of design actions, or, most generally, by making a trade-off between artifact and process considerations. For example, when designing a car, it may be difficult to anticipate all the vibration pathways from the engine to the other parts of the car. Although resonances could be avoided once the masses and stiffnesses are known, this information is typically unavailable at the early stage of design. To mitigate the risk of having to modify the car design when resonances are discovered in final testing, one could either invest in additional analysis process steps to predict the vibration parameters earlier in the design process (i.e., reduce the uncertainty), or one could add vibration isolation to the artifact (i.e., reduce the consequences). To make a good risk mitigation decision, one should consider both options: reducing the uncertainty by adding process steps for additional analysis or testing, and reducing the impact of the uncertainty by making the artifact more robust to uncertainty. Making trade-offs between these two options requires the consideration of NDU in which both process and artifact consequences are considered.

Distinction between specifications and predictions. Design information about the artifact is formulated in terms of properties. In the RDT conceptual model, this is reflected in the Artifact Information State in which properties appear in both the concept specifications and in the concept predictions. However, in design practice (and in other design theories), such a distinction is not always made. For instance, in Gero's work on design prototypes, a distinction is made between expected behavior (Be) and actual behavior of the structure (Bs)(Gero 1990); in RDT, expected behavior would correspond to specified behavior properties; while actual behavior of the structure is not explicitly considered in RDT because it cannot be known — it can only be predicted. The uncertainty that exists in this prediction plays a crucial role in design and should not be neglected. In addition, Gero considers only one representation of structure (S). In RDT, structure is represented in terms of properties and is therefore not substantially different from behavior. Structure also should be considered in terms of both concept specification and concept prediction.

If a design method is to take uncertainty and risk into account, then the predicted property values need to be considered rather than only the specified values. Concepts are alternatives considered while exploring the design space. When specifying a concept, there is no guarantee that the specified properties are actually realizable. It may well be that one is exploring a dead-end in the concept tree and must backtrack. From a risk management perspective, the uncertainty in the concept predictions must be taken into account when deciding which branches in the concept tree to explore next.

Interpretation of requirements as concept specifications only. In engineering design and especially in systems engineering, requirements are often used in a variety of contexts, and their meaning is often vague. It is therefore useful to clarify how requirements are interpreted in RDT. In the process, it is argued that requirements should be interpreted only as concept specifications.

Requirements as contractual agreements — consistent with RDT. When an OEM specifies subsystems to be produced by a supplier, it provides the supplier with a requirements document in which the constraints on all relevant properties of the subsystem are specified. This is consistent with the notion of a concept specification in the RDT conceptual model. Ideally, the OEM specifies the key characteristics of the subsystem that affect the performance and utility of the overall system (the "whats"), without specifying the implementation details for how these key characteristics are achieved (the "hows"). Assuming the OEM has confidence in its supplier, the predicted values of the subsystem properties should be within the specified property constraints.

Derived requirements as expressions of previous design choices — consistent with RDT. Throughout a design process, requirements are added. Often, implementation choices are expressed in terms of derived requirements. This is consistent with concept specifications in terms of property constraints in the RDT conceptual model. The derived requirements limit the types of system alternatives that are being considered as solutions — they refine or subset the parent concept. However, it is important to recognize that derived requirements only reflect the concept that is currently being explored. If it turns out that the current concept is infeasible or undesirable, then the "requirements" may need to be modified — they are thus not "required" in a strict sense.

Requirements as expressions of preferences — not consistent with RDT. Requirements are often interpreted as preferences. However, in RDT, preferences are expressed as relationships between concept predictions and ultimately as a concept decision criterion. Preferences are thus expressed as a preference or utility function, not in terms of constraints. The discrepancy is due in part to the confusion as to whose preferences are being represented. One could argue that requirements represent the preferences of the customer, while in RDT, only the preferences of the designer are considered. These two are not necessarily aligned. Clearly, if the customer provides a requirements document as part of a contractual agreement and will not pay for the artifact unless all the requirements are met, then it is in the best interest of the designer to produce an artifact that meets the requirements; if not, the designer will not get paid, resulting in a utility that is clearly less than the utility of not taking on the design project in the first place. In this case, the designer should start from a concept specification (to be further refined in the rest of the design process) that includes all the customer's requirements; any artifact that does not map onto this concept will not result in payment by the customer. However, within the set of solutions that do meet all the requirements, the designer may have preferences that are different from the customer's. Similarly, even if a solution that meets all the requirements exists, then a designer may refuse to take on the design project because it does not fit his or her preferences — the project has an expected utility that is less than the expected utility of not taking on the design project.

Requirements as problem definitions — not consistent with RDT. Continuing the discussion of the previous paragraph, one could interpret the requirements specified by the customer as the design problem definition. But this is not consistent with RDT. The problem definition in RDT is always the same: "How to maximize expected NDU?" It just happens that based on the customer's requirements, the designer is compelled to look for NDU maximization opportunities within the scope of the concepts that reflect the customer's requirements. In the case of an in-house design effort, one may also start from an initial set of requirements, for instance, that result from a market analysis in which new product opportunities are identified. However, it is clear then that these requirements are not really a problem definition, but really a first concept to be considered in the search for a solution that maximizes expected NDU. They are not set in stone. If it turns out that further exploration and analysis reveals that a better opportunity for maximizing expected NDU can be found by deviating from the initial requirements, then one should not hesitate to do so. For instance, an automotive company may create a specification for the new car, but in the process of exploring different more detailed concepts for the car, they discover that there is a better opportunity for mobility platform other than a car (e.g., a motorcycle or a personal transporter like a Segway), then the company should not hesitate to pursue this option.

To summarize, in a good design method, requirements should be clearly distinguished from preferences. They should be interpreted as initial steps towards a solution, i.e., the specification of concepts that are being explored in order to maximize expected NDU.

3.4 REVISITING THE RELATED WORK

Having presented a new theory of design, it is useful to look back on the existing theories. Here, each theory is revisited and reinterpreted in the context of RDT.

3.4.1 Simon's Sciences of the Artificial

In his Sciences of the Artificial, Simon claims that designers must seek satisficing solutions because of bounded rationality (Simon 1996). Bounded rationality refers to the limited processing power of the human mind or computer surrogate. Because of this limited processing power, the time required to search a large space for the optimal solution is quite large. Satisficing solutions, then, are solutions which are not optimal, but good enough, because optimal solutions would take too long to find. In RDT, the time and other associated costs with searching for an optimal solution are included in the formulation of the design process decision. These are the design phase resources that are consumed with each design action. In addition, the goal in RDT is not the optimal *artifact* utility but the optimal *net overall* utility, which combines the utility of the artifact and the utility of the process. This makes explicit the tradeoff between the cost of search and the utility of the chosen artifact.

Besides the ideas of bounded rationality and satisficing solutions, Simon also suggests that the topics of utility theory, statistical decision theory, and optimization techniques be included in the curriculum of design. These fields are foundational to RDT.

3.4.2 General Design Theory

In General Design Theory, designing is defined as a mapping from the function space to the attribute space through the metamodel space (Yoshikawa 1981, Tomiyama, et al. 1987). Because of uncertainty referred to in GDT as the "finiteness and imperfections of the real knowledge", the design solution is found in the metamodel space rather than the attribute space, so the solution is actually an approximation of a solution. In Rational Design Theory, the expression of uncertainty in this mapping is made explicit with the inclusion of the concept prediction in the AIS. In addition, concepts can be specified in terms of behavior and structure in addition to specification in terms of function alone as in GDT. As mentioned previously, this is a departure from the traditional notions of a problem definition defined by functional requirements and solution described by structure. This change in perspective reflects the fact that the problem definition in all design problems is to maximize expected NDU.

Reich argued that no realistic domains have the topological structure assumed by GDT, which, along with other limitations, makes the conclusions from GDT inapplicable to practical design problems (Reich 1995). A primary limitation of the topological structure assumed by GDT is the reliance on Axiomatic Set Theory to describe concepts in both the function and attribute spaces. This is problematic because attributes are inherently uncertain, so that set operations do not apply in the attribute space. However, set operations can be used to describe concepts effectively. In Rational Design Theory, concepts are specified as a set of constraints, which designates a subset of the property space projection, P'. Predicted attributes of those concepts are captured as random variables. This enables the uncertainty in predicted attributes to be formalized mathematically without a reliance on axiomatic set theory.

3.4.3 Function-Behavior-State

The Function-Behavior-State (FBS) diagram is a model of entities that separates subjective function from objective behavior and state (Umeda, et al. 1990). Unlike the AIS in RDT, the FBS diagram does not explicitly characterize the uncertainty in the predicted outcomes of concepts. Umeda, Tomiyama, and Yoshikawa argue that design using the FBS diagram should proceed by first specifying functions and then connecting these functions to behaviors and states, but no justification is provided for this method. Their hypothesis that design should proceed in such a manner can be evaluated using RDT.

3.4.4 Gero's Design Prototypes

Gero argues that design is a process of transforming function into the design description through a series of intermediate representations, including the expected behavior, the actual behavior of the structure, and the structure itself (Gero 1990). The transformations between these representations are the design actions of formulation, synthesis, analysis, evaluation, reformulation, and production of the design description. Gero's expected behavior is derived from the function in the action of formulation. The behavior of the structure is derived from the structure in the action of analysis. The expected behavior and the behavior of the structure are compared in the action of evaluation to determine if the given structure meets the expected behavior. New structures are synthesized based on the comparison of the expected and actual behavior. Conversely, new expected behaviors can be generated in the action of reformulation also by comparing expected and actual behaviors.

The fundamental difference between Gero's work and RDT is the problem definition. In RDT, the objective is to maximize expected NDU, and concepts are specified in terms of function, structure and behavior as suggested solutions to meet this objective. In Gero's work, the function and expected behavior are interpreted as a statement of the problem, while the structure is a statement of the solution to the problem. A difference between the expected behavior and the behavior of the structure is recognized, but as RDT demonstrates, there should also be a distinction between the expected structure and the actual structure.

Each of Gero's representations and design actions has a counterpart in the RDT conceptual model, although the representations and actions are organized differently. Analysis is the same action in the RDT conceptual model as in Gero's work, the transformation from structure to actual behavior of the structure; however, in the RDT conceptual model analysis transforms specified function, behavior, and structure into predicted function, behavior, and structure, not only specified structure into predicted behavior. Thus, Gero's structure is contained in the concept specification in the RDT conceptual model, and Gero's actual behavior is represented in the prediction in the RDT conceptual model. Gero's function and expected behavior are also represented in the concept specification in the RDT conceptual model. As mentioned above, Gero's work lacks the distinction between specified and actual function and structure, which are captured in the concept prediction in the RDT conceptual model. Gero's formulation and reformulation actions are the generation of expected behavior either directly from the function (formulation) or through analysis and evaluation (reformulation). In the RDT conceptual model this is equivalent to the formulation of the evaluation function which generates a concept decision criterion from a concept prediction. It is essentially a statement of what constitutes a preferred artifact concept. The comparison of the expected behavior to the actual behavior is evaluation. This is the same action in the RDT conceptual model, but is not necessarily achieved by comparison of desired to actual behavior alone. In the RDT conceptual model, evaluation can be performed on the basis of function, structure, and behavior.

One action mentioned by Gero not explicitly discussed in the RDT conceptual model is the generation of the design description. The design is characterized in the concept specification, and the specification is expressed as a set of constraints on property values. While this characterizes the concept, it may not necessarily be expressed in the form of engineering drawings or CAD files. Drawings and CAD files are tools which reduce the ambiguity in the understanding of the concept specification between the designer and the manufacturer. The generation of such documents is a synthesis action which may or may not be undertaken by the designer during a design process. The decision to undertake this action must depend on the degree of reduction in uncertainty afforded by the generation of the documents and the cost of that generation.

Gero builds on his model of design processes to propose design prototypes which are essentially reusable models of elements of the design representation. Similarly, in the context of RDT, aspects of the AIS and the models used to generate portions of that AIS can be reused from one project to another to avoid rework.

3.4.5 Decision-Based Design

As mentioned in Section 2.1.5, Decision-Based Design is a collection of methods inspired by the notion that design is a series of decisions. This realization enables the use of decision theory as a foundational theory for design. RDT builds on previous work (e.g. (Hazelrigg 1996, Marston, et al. 1997, Hazelrigg 1998, Krishnamurty 2006, Lewis, et al. 2006b)) by restating the design problem as a series of decisions about the design *process*. By changing the context in this manner, tradeoffs can be made between the cost and benefit of the various design actions through the maximization of expected NDU.

Decision theory also inspires the structure of the AIS in the RDT conceptual model as consisting of three aspects: concepts, concept predictions, and concept decision criteria. These aspects correspond to the decision elements of alternatives, outcomes, and preferences, but with respect to the artifact decision rather than the design process decision. This structure provides a means to organize the information generated in design, which in turn enables comparison of design methods.

3.4.6 Coupled Design Process

Braha and Reich describe design as a process of translating product requirements in the function space into a design description in the structure space (Braha, et al. 2003). Their work assumes a method that starts by stating desired outcomes in the context of functional properties. In RDT concepts can be specified in terms of function, but it is not required to do so. Desired artifact outcomes are expressed in the evaluation action that translates a concept's prediction into its criterion. In addition, Braha and Reich do not explicitly model uncertainty in their framework. They discuss approximate closures as a means for explaining design failures due to uncertainty, whereas in RDT uncertainty is explicitly accounted for in the prediction associated with each concept.

3.4.7 Concept-Knowledge theory

Concept-Knowledge theory is proposed as a model of design which accounts for creativity and innovation (Hatchuel, et al. 2009). The goal is to translate a concept, which is initially an undecidable proposition, into knowledge, a decidable proposition. This involves both the mapping between requirements and solutions and the generation of new concepts. The RDT conceptual model captures the generation of new concepts in the synthesis design action, which adds new concepts to the AIS. However, in RDT, the desired outcomes of the artifact are expressed in an evaluation model, which transforms a concept prediction into a concept decision criterion. This transformation can be stated in terms of requirements formulated as property constraints, but it is not necessary to express desired outcomes in this manner. Other methods for expressing desired artifact outcomes include utility functions and objective functions. Because requirements are not necessary, the notion of a design solution which meets requirements is not part of RDT. Design does not end when a solution is found; rather, design should end when the cost of all possible design actions outweighs the potential benefits of those actions.

3.5 CRITICAL REVIEW OF RDT

To validate the proposed theory, RDT is evaluated in the following section with respect to the requirements and research questions set forth in Chapter 1. Thereafter, the strengths and weakness of RDT are discussed.

3.5.1 Meeting the Requirements

In Chapter 1, seven requirements were identified for an explanatory theory of design. In Section 2.1 existing theories were reviewed and it was found that none of the existing theories of design simultaneously meet all seven requirements. Decision theory was reviewed in Section 2.1.8 and the decision analysis of design process decision was identified as a means for simultaneously satisfying all requirements. These findings and the requirements are repeated in Table 3.2. Having presented RDT in this chapter, the requirements are revisited here to establish whether or not they have been met by this new theory.

TABLE 3.2. REQUIREMENTS ANALYSIS OF EXISTING WORK AND DECISION ANALYSIS OF DESIGN PROCESS DECISIONS

Requirement	(E)GDT	FBS	Gero	DBD (artifact)	Coupled Design	C-K	Decision Analysis Applied to Process
Includes the generation and revision of specifications			x		x		x
Includes the creative process of generating new concepts						x	x
Explicitly includes uncertainty				x			x
No limit on the expression of preferences and beliefs				x			x
Incorporates the allocation of design phase resources as outcomes of design actions							x
Recognizes a tradeoff between product and process							x
Provides a means of evaluating design methods relative to one another							x

The first and second requirements are that the theory should include the generation and revision of specifications and the creative process of generating new concepts. In RDT the creative process of generating new concepts is captured in the design action *synthesis*. This action results in the addition of new concepts to the AIS and comes at the cost of design phase resources. The action *synthesis* is a design process decision alternative that can be applied at any point in the design process. Whether or not *synthesis* is a good action to take at any point in time can be evaluated using decision analysis.

The generation and revision of specifications is interpreted to refer to the generation and revision of evaluation functions. This is also enabled by RDT. As discussed in Section 3.2.3.1, the action *evaluation* includes both the formulation and application of an evaluation function to generate a *concept decision criterion* from a *concept prediction*. Like *synthesis*, this action can be applied at any point in a design process, as long as there is a *concept prediction* to be evaluated.

The third requirement is that the theory should explicitly include uncertainty. Decision-based design was the only body of work in the review that met this requirement. Since RDT is also based on decision theory, RDT includes the explicit characterization of uncertainty, but unlike traditional DBD, RDT includes uncertainty in both the prediction of the concept and the prediction of outcomes associated with design actions. Thus, RDT includes the explicit characterization of uncertainty in the artifact and the design process. This characterization of uncertainty enables risk to be considered throughout the design process.

In a similar vein, DBD was the only body of work in the review that met the requirement of placing no limits on the expression of preferences and beliefs. Again RDT inherits this benefit of decision theory. Unlike methods which prescribe forms for uncertain inputs and objective functions, the structure of the AIS places no restriction on the expression of beliefs and preferences in the artifact decision. Likewise, no restrictions are placed on the expression of beliefs and preferences with respect to the process decision.

The fifth requirement is that the theory should include the allocation of design phase resources as outcomes of design actions. This requirement was inspired by Howard's definition of a decision. As discussed in Section 3.2.3.2, there are two aspects to the outcomes of design actions: changes in the AIS and the consumption of design phase resources. In RDT, these outcomes are included in the decision analysis of design process decisions.

Building on the fifth requirement, the sixth requirement is that the theory should recognize a tradeoff between the artifact and the design process. Meeting the fifth requirement is one aspect of this requirement as well, because the design phase resource consumption must be included in the decision formulation in order to consider the tradeoff. To fully recognize the tradeoff, the theory must also include the benefits to be gained from producing the design artifact. In RDT, these two aspects comprise the Net Design Utility. By including both of these aspects in the formulation of the NDU, RDT recognizes this tradeoff.

The last requirement is that the theory should provide a means for evaluating design methods relative to one another. This requirement is met in RDT by the analysis of design process decisions. Since a design method prescribes a way to make design process decisions, analyzing those decisions enables both the comparison of design methods and the evaluation of those methods with respect to the rational ideal. The qualitative comparison is enabled by the decomposition of the decision itself, which allows individual elements of each decision problem to be compared to one another. The quantitative comparison is achieved by solving for the alternative with the highest expected NDU.

Having demonstrated that RDT meets all the requirements identified in Chapter 1, the next step is to revisit the research questions. This is done in the next section.

3.5.2 Answering the Research Questions

The primary research question addressed in this thesis is “what theoretical foundation of design has explanatory power and enables quantitative evaluation of design processes”. The primary hypothesis is that RDT meets these qualifications. This primary hypothesis is decomposed into two secondary research questions addressing the qualitative and quantitative comparison abilities of RDT.

The qualitative comparison abilities of RDT are enabled by the the systematic decomposition of the design process decision into alternatives, outcomes, and preferences. In the RDT conceptual model, the artifact decision is decomposed into concepts, concept predictions, and concept decision criteria. These correspond to the alternatives, outcomes, and preferences of the artifact decision, and are captured in the Artifact Information State. The alternatives in the process decision are the design actions of synthesis, analysis, and evaluation. The outcomes of these alternatives are updated information about the artifact decision and the cost associated with performing the design action. The preferences with respect to these outcomes constitute the Net Design Utility. Providing this structured means for representing a design process decision allows for the comparison of design processes and design methods on the basis of their elements. For example, by placing methods such as robust design and reliability-based design in the context of RDT, one sees that although both are methods for design under uncertainty, the two strategies differ in the structure of the evaluation function for design concepts and may differ in the representation of predictions of design outcomes. A similar analysis, although conducted in the context of utility theory rather than in the context of RDT, was conducted by Lee and coauthors (Lee, et al. 2010b). Specifically, robust design assumes an evaluation function of a weighted sum of the mean and variance of key performance metrics. Reliability-based design, on the other hand can take one of two forms: cost-constrained reliability maximization or reliability-constrained cost minimization. Both of these forms of the evaluation function are distinctly different from a weighted sum of the mean and variance of performance metrics.

The quantitative comparison abilities RDT are enabled by the quantitative analysis of design process decisions. Given two or more very specific sequences of design actions, one can compute the sequence with the maximum expected NDU. This capability is demonstrated in more detail in Chapter 5. Returning to the robust design versus reliability-based design example above, these two methods can be quantitatively evaluated for a specific designer's beliefs and preferences concerning a particular design scenario. Such a quantitative analysis would indicate whether robust design or reliability-based design more closely approximate rational behavior for that designer.

3.5.3 Strengths and Weaknesses of RDT

Although RDT has been shown to meet the requirements and satisfactorily answer the research questions, there are definitely strengths and weaknesses of the theory. The strengths of the theory stem from meeting the requirements. Specifically, RDT is an improvement over existing design theories because it enables the comparison of design methods while recognizing tradeoffs between the process and the artifact and while explicitly accounting for uncertainty. The foundation of decision theory enables RDT to be capable of representing any sequence of design actions and any AIS without limitation of the expression of preferences or beliefs. This means that RDT can be used to analyze any design process, regardless of whether the process is successful in practice.

However, these strengths come at a price. While RDT is sufficiently general to apply to any design method, it may not be immediately obvious how to allocate a particular activity to an action in RDT or how to categorize pieces of information in the AIS. Representing a design process in the context of RDT requires both knowledge of the design process or method and a working knowledge of RDT and decision theory.

Furthermore, the application of decision analysis to design process decisions can be a cumbersome process. Quantitative comparison especially requires often significant computational resources to achieve results for even a small problem. I expect that this can be improved in the future with more practice applying decision analysis to design process decisions and with improvements in computational tools for computing expected utilities of design actions.

3.6 THESIS ROADMAP

In this chapter Rational Design Theory was presented and evaluated with respect to the requirements and research questions. It was found that RDT meets the requirements and also provides the answer to the primary research question. In the next chapter, a review of the systematic design method of Pahl and Beitz is presented to demonstrate the qualitative evaluation capabilities of RDT. Thereafter, in Chapter 5, two example problems are presented to demonstrate the quantitative comparison of design methods enabled by RDT.

CHAPTER 4: A REVIEW OF THE SYSTEMATIC DESIGN METHOD OF PAHL AND BEITZ IN THE CONTEXT OF RDT

In Engineering Design: A Systematic Approach, Gerhard Pahl and Wolfgang Beitz present a four phase approach to design garnered from extensive study of existing design practice (Pahl, et al. 1996). Because of its basis in design practice, and the extensive use of the work as a text for design education, the approach, which will from now on be referred to simply as Pahl and Beitz (P&B) has become a standard method in design research. As such, the interpretation and description of P&B in the context of RDT is useful both to build confidence in the comprehensiveness of RDT as a design theory and to further explain RDT to readers who are already familiar with P&B. Thus, in this chapter, the phases and activities of P&B will be discussed in the context of RDT. First, an overview of P&B is provided including presentation of keywords in the P&B methodology and explanation of a diagram of the P&B method in Section 4.1. Next, the tasks of P&B are discussed in the context of the RDT conceptual model in Section 4.2. In this section, each task of P&B is represented as a change or changes to the Artifact Information State. Finally, a discussion is presented in Section 4.3 in which the characteristics of good design methods identified in Section 3.3 are discussed relative to P&B.

4.1 OVERVIEW OF THE SYSTEMATIC DESIGN METHOD PROPOSED BY PAHL AND BEITZ

In P&B, design activities are divided into four phases: **Planning and Clarifying the Task**, **Conceptual Design**, **Embodiment Design**, and **Detail Design**. The input to the approach is the **Task**, which is determined by the market, company, and economy.

Proceeding through the four phases of design leads to the identification of the **design solution**. In this section, an overview of the method is presented. Keywords in the P&B method are introduced in Section 4.1.1. Thereafter, a diagram of the method is presented and each phase of the method is discussed in detail in Section 4.1.2. A brief commentary on the method concludes this section in Section 4.1.3.

4.1.1 Keywords in P&B

P&B makes use of specialized terminology for key design activities and milestones. Before proceeding through a description of the whole method, these keywords are introduced for clarification. Throughout this chapter, keywords from P&B are denoted by boldface type, while keywords in RDT are denoted by italics.

- **Principal Solution**: the outcome of the **conceptual design** phase and the input to the **embodiment design** phase; a preliminary solution which may take the form of a **function structure**, circuit diagram, flow chart, line sketch, or scale drawing.

- **Concept Variants**: proposed design solutions in the **conceptual design** phase; **concept variants** consist of several **working principles** and **working structures** intended to achieve desired **functions**.

- **Function**: "the general input/output relationship of a system whose purpose is to perform a task"; usually defined with a verb and a noun such as "increase pressure"; **functions** are derived from conversions of matter, energy, and information (aka signals).

- **Function Structure**: a "meaningful and compatible combination of sub-functions into an overall function". **Function structures** are usually depicted as block diagrams.

- **Working Principle**: the interrelationship between the physical effect and the geometric and material characteristics of a solution; identifying the **working principle** is the first step towards implementation of a solution. **Working principles** must be found for each sub-function in the overall **function**.

- **Working Structure**: the combination of several **working principles** to fulfill the overall task. **Working structures** can be depicted using circuit diagrams, flow charts, line drawings and sketches.

- **Construction Structure**: a representation of the **principal solution** that is more concrete than the **working structure**; the **construction structure** incorporates information about production, assembly, transport, etc..

- **Preliminary Layout(s)**: the general arrangement, component shapes and materials of the proposed design solution, determined in a provisional sense; during **embodiment design** several **preliminary layouts** are proposed and evaluated; a primary **preliminary layout** is an intermediate milestone in the **embodiment design** phase on the way to the **definitive layout**.

- **Definitive Layout**: a more definitive arrangement of components, shapes, and materials of the proposed design solution that facilitates a "clear check of function, durability, production, assembly, operation and costs"; The **definitive layout** is the output of the **embodiment design** phase.

- **Requirements List**: The **requirements list** is the output of the **Planning and Clarifying the Task** phase. It is also sometimes called the design specification. The re**quirements list** defines what is acceptable to the customer in the form of demands and wishes, where demands must be met and wishes are preferred but not required. In the evaluation stage, demands are used to screen out variants that do not meet demands in a selection process, whereas wishes are considered during evaluation of variants that meet the demands.

With the exception of the **Requirements List**, each of these keywords describes the design solution, or an element of the design solution, at a point along the transformation from **Task** to **Solution**. The **Requirements List** describes elements of the design solution, but also characterizes the design problem.

4.1.2 Discussion of P&B by Phase

A diagram of the full P&B method is given in Figure 4.1 below, reproduced from Figure 3.3 in the text. Each phase of the method is then described in more detail.

4.1.2.1 Planning and Clarifying the Task

The first phase of P&B is **Planning and Clarifying the Task**. The **Task** is the input to this phase, and the output of this first phase is the **Requirements List**. Activities in this phase include analysis of the market and company situation, search and selection of product ideas, formulation of the product proposal, clarification of the task, and the elaboration of the **Requirements List**. The clarification of task activity is the specification of requirements to be fulfilled by the product and constraints on the design in as much quantitative and qualitative detail as possible.

4.1.2.2 Conceptual Design

Having elaborated the **Requirements List**, the next phase of P&B is **Conceptual Design**. The output of this phase is the **Principal Solution**. Activities include identification of the essential problems, establishment of **Function Structures**, the search for **Working Principles** and **Working Structures**, the combination of these **Working Principles** and **Structures** into **Concept Variants**, and the evaluation of these **Variants** against technical and economic criteria. **Variants** which do not meet the demand requirements are eliminated, and the remaining **Variants** are evaluated by application of wish requirements. Although the **Requirements List** encompasses both technical and economic criteria, evaluation in the **conceptual design** phase is focused more on the technical rather than economic factors.

One or more **Principal Solutions** may be identified during this phase. These preliminary solutions are represented using **Function Structures**, block or circuit diagrams, flow charts, or preliminary drawings. Often, preliminary form design and material selection must be performed to assess technical and economic criteria to compare **Concept Variants**, although this level of detail is not strictly required in the **conceptual design** phase.

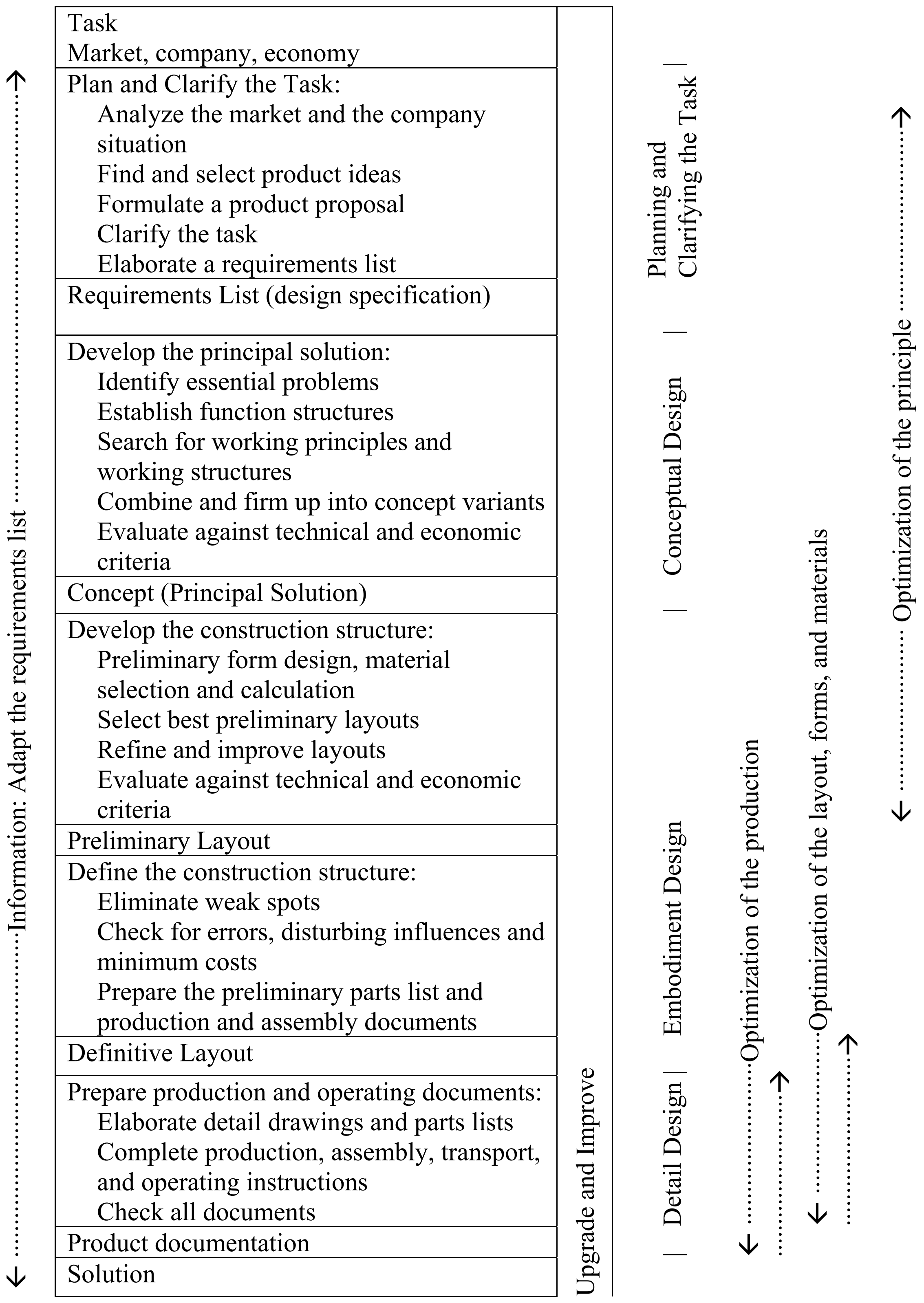

Task
Market, company, economy
Plan and Clarify the Task:
Analyze the market and the company situation
Find and select product ideas
Formulate a product proposal
Clarify the task
Elaborate a requirements list
Requirements List (design specification)
Develop the principal solution:
Identify essential problems
Establish function structures
Search for working principles and working structures
Combine and firm up into concept variants
Evaluate against technical and economic criteria
Concept (Principal Solution)
Develop the construction structure:
Preliminary form design, material selection and calculation
Select best preliminary layouts
Refine and improve layouts
Evaluate against technical and economic criteria
Preliminary Layout
Define the construction structure:
Eliminate weak spots
Check for errors, disturbing influences and minimum costs
Prepare the preliminary parts list and production and assembly documents
Definitive Layout
Prepare production and operating documents:
Elaborate detail drawings and parts lists
Complete production, assembly, transport, and operating instructions
Check all documents
Product documentation
Solution
Information: Adapt the requirements list
Upgrade and Improve
Planning and Clarifying the Task
Conceptual Design
Embodiment Design
Detail Design
Optimization of the production
Optimization of the layout, forms, and materials
Optimization of the principle

FIGURE 4.1. OVERVIEW OF P&B

4.1.2.3 Embodiment Design

After identifying the **Principal Solution(s)**, one proceeds at a more concrete level with **Embodiment Design**. The output of **Embodiment Design** is the **Definitive Layout**. An intermediate milestone is the identification of one or more **preliminary layouts**. Another term for layout is the **Construction Structure**. Leading up to the **Preliminary Layout**, the goal is to develop the **Construction Structure**. This is achieved through preliminary form design, material selection, and analysis; selection of the best **Preliminary Layouts**; refinement and improvement of these layouts; and evaluation of the selected **Preliminary Layouts** with respect to technical and economic criteria. These steps result in the selection of one or more **Preliminary Layouts**. Transitioning from the **Preliminary Layouts** to the **Definitive Layout** is a process of refining the **Construction Structure**. This is accomplished through the elimination of weak spots, checking for errors, disturbing influences and minimum costs, and preparation of a preliminary parts list and production and assembly documents. These documents constitute the description of the **Definitive Layout**. A critical component of the **embodiment design** phase is the evaluation of the economic viability of the project. Only projects which are financially feasible should be continued into the **Detail Design** phase.

4.1.2.4 Detail Design

The final phase of P&B is **Detail Design**. In this phase the **Definitive Layout** is made more concrete through the development of the production documentation. This is achieved through the elaboration of detailed drawings and parts lists; the generation of complete production, assembly, transport, and operating instructions; and thorough checking of all documents.

4.1.2.5 Iteration and Rework

In addition to the activities within specific phases, Pahl and Beitz identify three crucial activities that span multiple phases. These activities may necessitate a return to activities in a previously completed phase, possibly leading to iteration and rework. The activities are shown at the right side of the overview figure and include the optimization of the principal; optimization of the layout, forms, and materials; and optimization of the production. Optimization of the principal spans **Planning and Clarifying the Task**, **Conceptual Design**, and **Embodiment Design** up to the **preliminary layout**. Optimization of the layout, forms, and materials spans **Conceptual Design**, **Embodiment Design**, and **Detail Design**. Optimization of the production encompasses **Conceptual design**, **Embodiment Design** and **Detail Design**. In addition to these optimization activities, information is continuously generated and captured in an adapted **Requirements List**. This is shown at the left side of the overview figure.

4.1.3 Commentary on P&B

The P&B method includes several iterations of a selection procedure. This procedure includes two steps: elimination and preference. First, concepts are eliminated which are not internally consistent, do not satisfy demand requirements, or are unlikely to be technically or economically feasible. The remaining concepts are then ranked in order of preference as formalized by wishes in the **requirements list**. Only the most preferred concepts are taken forward to the next steps of the method. Suggested procedures for evaluating concepts include arbitrary value functions, weighted sums of evaluation criteria, and normalization and ranking procedures. In these procedures there is little if any explicit consideration of uncertainty.

The P&B approach is intended to be comprehensive; however, some common design activities are not specifically identified in the overview of the method because the timing of these activities may vary from project to project. Two examples of these common design activities are prototyping and sub-contracting. Designers are expected to use their expertise to adapt the step-by-step approach when appropriate. Although the full approach is beneficial in original design to prevent overlooked details, some steps of the approach can often be skipped or shortened in the case of adaptive design.

4.2 P&B IN THE CONTEXT OF THE RDT CONCEPTUAL MODEL

All design processes consist of information processing activities. As discussed in Chapter 3, in the RDT conceptual model these activities are characterized as *synthesis*, *analysis*, or *evaluation* by the resulting changes in the *Artifact Information State*. Recall that the *AIS* is a conceptual model of design information in the context of *concepts*, *concept predictions*, and *concept decision criteria*. *Concepts* are incomplete descriptions of hypothetical artifacts, expressed as constraints on the property space. *Concept predictions* characterize the designer's beliefs about the outcomes of the proposed concept, expressed as probability distributions over a property space projection. *Concepts* are ranked on the basis of their predictions using *concept decision criteria*, which are generated by application of an *evaluation function*. *Synthesis* is the generation of new concepts, *analysis* is the generation of concept predictions, and *evaluation* is the generation of *concept decision criteria*.

Each activity identified in P&B generates information that can be captured in the *AIS*. These activities are discussed by phase in Sections 4.2.1 through 4.2.5. A short summary is presented in Section 4.2.6. As discussed above, each of the keywords of P&B

characterize the design solution, with the exception of the **Requirements List** which characterizes elements of the design solution and elements of the design problem. Because these keywords characterize the design solution, they are all represented as *concepts* in the *AIS*. The elements of the **Requirements List** which characterize the design solution are also represented as *concepts* in the *AIS*, while the elements of the **Requirements List** that characterize the design problem are captured in the *evaluation function* which generates the *concept decision criteria* in the *AIS*.

P&B as a process is a transformation of information from one representation of the design solution to another, such as the transformation from the **Preliminary Layout** to the **Definitive Layout** in **Embodiment Design**. Accordingly, in the RDT conceptual model, most activities in P&B are represented as *synthesis* actions which generate *concepts*. In fact, the analysis of P&B in the context of RDT reveals that P&B offers little guidance on *analysis* and *evaluation* in the design process, other than acknowledging that these are critical activities that must be performed. While this may appear to be a deficiency of P&B, it is understandable given the general nature of the method. *Analysis* and *evaluation* activities are highly problem-dependent; thus, it is nearly impossible to provide guidance on how to conduct these activities in a general sense.

In each of the next sections, a table is presented showing the representation of P&B tasks in the RDT conceptual model. The following abbreviations are adopted in these tables:

Abbreviations for changes to *AIS*:

- C: concept
- CP: concept prediction

- CDC: concept decision criterion
- EF: evaluation function

Abbreviations for Type of design action:

- S: synthesis
- A: analysis
- E: evaluation

TABLE 4.1. REPRESENTING “PRODUCT PLANNING AND CLARIFYING THE TASK” IN THE RDT CONCEPTUAL MODEL

	P&B activity	**RDT description**	**Change to AIS**	**Type of Design Action**
1	Analyze the market and company situation	Elicit beliefs about desired properties, company capabilities, market	CP, EF	A, E
2	Find and select product ideas	Generate *concepts* for satisfying EF; Elicit beliefs concerning *concept predictions*; Apply EF to *concepts* to generate *concept decision criteria.*	C, CP, CDC	S, A, E
3	Formulate a product proposal	Refine selected *concepts* in terms of desired artifact properties	C	S
4	Clarify the task	Refine *concept* for demands; Refine *evaluation function* for wishes	C, EF	S, E
5	Elaborate a **Requirements List**	Refine *concept* in solution-neutral terms	C	S

4.2.1 Product Planning and Clarifying the Task

An appropriate name for the first phase of P&B in the context of the RDT conceptual model might be “Defining the Evaluation Function”. This phase is about gathering and formalizing information about desired artifact properties and the relationships between artifact properties, company capabilities, customer preferences, revenue, and costs. A profit model can be constructed using this information to allow for the determination of preferred artifact properties. In P&B, a set of desired artifact properties is selected as a **product idea**, and these properties are formulated in a solution-neutral manner in the **requirements list**. Each of the steps of this phase of P&B are described in the context of RDT in Table 4.1.

As discussed in Chapter 3, some requirements are a statement of preference (wishes) while others define concepts (demands). Thus, wish requirements which express preference are captured in the RDT conceptual model as a formulation of an *evaluation function*, whereas demand requirements which define product concepts are captured as *concepts* in the RDT conceptual model. The mapping of the **requirements list** from P&B to the RDT conceptual model is not clear cut because while concepts and preferences are distinguished in RDT, they are confounded in the P&B **requirements list**.

4.2.2 Conceptual Design

The **conceptual design** phase is the first of several *synthesis-analysis-evaluation* loops in P&B. In this phase the goal is to transform the **Requirements List** into the **Principal Solution**. Initially, *concepts* are refined in terms of **function** and then **working principles** and **working structures**. *Concepts* are first refined and optimized on an element-wise basis to find **working principles** to fulfill each **function**. Then, these elements are combined into full **concept variants** that fulfill all of the desired **functions**. These combined *concepts* are then analyzed and evaluated in order to select the most promising ones for further development in the next phase of design. Each of these steps are presented in Table 4.2.

The advice of P&B for **conceptual design** is to break the design process into sub-problems by **function**. The design process is facilitated by solving a series of smaller, simpler design problems and combining the solutions into a solution to the larger design problem. This is likely to be a preferred process when an innovative solution is needed for a complex design scenario.

TABLE 4.2. REPRESENTING "CONCEPTUAL DESIGN" IN THE RDT CONCEPTUAL MODEL

	P&B activity	**RDT description**	**Change to AIS**	**Type of Design Action**
1	Identify essential problems	Refine *concepts* in terms of "essential problem"	C	S
2	Establish function structures	Refine *concepts* in terms of function	C	S
3	Search for working principles and working structures	Iteratively: • refine *concepts* in terms of **working principles/structures**, • elicit *concept predictions* for the refined *concepts*, • evaluate the refined *concepts*.	C, CP, CDC	S, A, E
4	Combine and firm up into concept variants	Merge *concepts* and predict their outcomes	C, CP	S, A
5	Evaluate against economic and technical criteria	Evaluate merged *concepts* per *evaluation function* to generate CDC	CDC	E

Conceptual design is about identifying the mechanisms for the design artifact. In this phase designers determine how each of the **functions** will be fulfilled by the artifact. The form and materials of the artifact are determined in the next phase, **Embodiment Design**.

4.2.3 Embodiment Design

The **Embodiment Design** phase incorporates additional *synthesis-analysis-evaluation* loops to determine the form, layout, and materials of the design solution. In the first loop, the **preliminary layouts** are determined by generating, analyzing, and selecting among *concepts* that specify a general form and layout and particular materials. These **preliminary layouts** are then refined and analyzed in the next two iterations, after which the best is chosen as the **definitive layout**. In the transformation from **preliminary layout** to **definitive layout**, two loops are performed. First, the *concepts* are analyzed and evaluated with respect to technical and economic criteria, which in the RDT conceptual model is captured in the evaluation function. After identifying the most promising of these *concepts*, new refined *concepts* are generated. These refined concepts are expected to improve the performance by eliminating weak spots. The refined *concepts* are then analyzed and evaluated to select the most promising as the **definitive layout**. In the final step of this phase, the chosen *concept* is further specified by preparing initial production and assembly documents. Each of these steps are interpreted in the context of RDT in Table 4.3.

4.2.4 Detail Design

In the final phase of P&B, the definitive layout is transformed into the design solution by specifying and documenting the *concept* in detailed production and assembly documents. These steps are intended to reduce ambiguity about the *concept* between the designer and manufacturer or other supplier. As shown in Table 4.4, in the context of the RDT conceptual model, this phase consists mostly of *synthesis*, as the chosen **definitive layout** is specified in greater and greater detail as refined *concepts*. The final step of this phase, checking all documents, requires some analysis and evaluation to determine if the refined *concept* indeed optimizes the *evaluation function*.

TABLE 4.3. REPRESENTING "EMBODIMENT DESIGN" IN THE RDT CONCEPTUAL MODEL

	P&B activity	**RDT Description**	**Change to AIS**	**Type of Design Action**
1	Preliminary form design, material selection and calculation	Generate form and material *concepts*, predict outcomes, generate/update *concept decision criteria*	C, CP, CDC	S, A, E
2	Select best preliminary layouts	Generate/update *concept decision criteria*	CDC	E
3	Refine and improve layouts	Generate refined *concepts* from selected *concepts*	C	S
4	Evaluate against technical and economic criteria	Generate/update *concept decision criteria*	CP, CDC	A, E
5	Eliminate weak spots	Generate refined *concepts* to eliminate weak spots	C	S
6	Check for errors, disturbing influences and minimum costs	*Analyze* and *Evaluate* refined *concepts*	CP, CDC	A, E
7	Prepare preliminary parts lists and production and assembly documents	Refine *concepts* to include parts lists, drawings, assembly documents, etc.	C	S

TABLE 4.4. REPRESENTING "DETAIL DESIGN" IN THE RDT CONCEPTUAL MODEL

	P&B activity	**RDT Description**	**Change to AIS**	**Type of Design Action**
1	Elaborate detail drawings and parts lists	Generate refined *concepts* represented by drawings and parts lists	C	S
2	Complete production, assembly, transport, and operating instructions	Generate refined *concepts* represented by assembly, transport, and operating instructions	C	S
3	Check all documents	Predict outcomes and evaluate to verify that the refined *concepts* represented in the documents correspond to the **Design Solution** as specified in the **Requirements List**	CP, CDC	A, E

4.2.5 Other Design Activities:

The four phase-spanning activities of P&B encompass all aspects of the RDT conceptual model, as shown in Table 4.5. Each of the optimization activities consist of *synthesis-analysis-evaluation* loops focusing on a particular aspect of the artifact. In the optimization of the principle, the focus is on identifying and improving the mechanisms for achieving the desired functions. The shapes and substance of the artifact are addressed in the optimization of the layout, forms, and materials. Finally, the process for realizing the artifact is the focus in the optimization of the production. Throughout the P&B method, the **requirements list** is updated to incorporate selected *concepts* and to reflect changes to the *evaluation function*. The incorporation of selected *concepts* is a process of *synthesis*, while changes to the *evaluation function* represent *reformulation*.

TABLE 4.5. REPRESENTING P&B PHASE-SPANNING ACTIVITIES IN THE RDT CONCEPTUAL MODEL

P&B Phase-Spanning Activity	**RDT Description**	**Change to AIS**	**Type of Design Action**
Optimization of the principle	Generate, analyze and evaluate *concepts* in terms of **Working Principle**	C, CP, CDC	S, A, E
Optimization of the layout, forms and materials	Generate, analyze, and evaluate *concepts* in terms of the layout, forms, and materials	C, CP, CDC	S, A, E
Optimization of the production	Generate, analyze, and evaluate *concepts* in terms of production and assembly plans	C, CP, CDC	S, A, E
Adapt the **requirements list**	Update the **Requirements List** to reflect selected *concepts*; Adapt the *Evaluation Function* to reflect changes to the value proposition	C, EF	S, A, E

4.2.6 Summary of P&B in the Context of RDT

By viewing each activity of P&B in the context of the RDT conceptual model, several conclusions come to light. First, it is evident that most of the P&B activities are *synthesis* actions representing the generation of new *concepts.* This is not surprising, as designers must propose many ideas in order to find a *concept* that is likely to add value. One also sees that each phase of P&B involves one or more *synthesis-analysis-evaluation* loops to "select" a *concept* for further refinement. These loops focus on different aspects of the **design solution**, namely the **principle**, the **layout**, and the **production**. Thus, the P&B approach is essentially a heuristic for incrementally refining concepts from an initial abstract task through increasingly detailed representations.

Although the method does direct designers to analyze and evaluate concepts, P&B does not give much guidance about how to do so. This is expected because the method is intended to be general. Designers who use the method are expected to apply their expertise to identify and execute appropriate analyses to facilitate evaluation of *concepts*. Thus, P&B is not helpful for determining which analyses to undertake, how many analyses to perform, how many concepts to evaluate, or when to stop concept refinement. The answers to these questions are problem- and designer-dependent, and will vary greatly from one problem to the next. What guidance is given regarding evaluation and selection of concepts prescribes a two-pronged selection approach of elimination of unsuitable concepts followed by evaluation with respect to wish requirements. This procedure imposes a structure on the preferences of the designer that may be incompatible with his actual preferences. In addition, the evaluation procedures suggested by P&B such as arbitrary value functions, weighted sums of evaluation criteria, and normalization and ranking, have been shown to be subject to logical inconsistencies which can lead to irrational choice (Hazelrigg 2003).

Perhaps the most confusing outcome of this analysis is the interpretation of the **requirements list** in the context of the RDT conceptual model. The **requirements list** incorporates two types of information: *concepts* that are proposed as potential artifacts, and preferences that characterize the desired properties of the artifact. Characterizing a preference as a *concept* is possible and perhaps even beneficial at times, but to do so constitutes a tradeoff between the design process and the designed artifact which should be made consciously by designers.

4.3 THE CHARACTERISTICS OF GOOD DESIGN METHODS EMBODIED BY P&B

In Chapter 3, seven characteristics of good design methods were identified by combining the normative perspective of RDT with the decision-theory-inspired RDT conceptual model. In this context, a "good" design method is one that approaches the RDT ideal of maximizing expected Net Design Utility. Having reviewed the systematic method of P&B in the preceding sections, the P&B method as a whole is evaluated in this section with respect to these six characteristics. Throughout this section, the question under consideration is: "How does the P&B method meet the characteristics of a good design method as identified by Rational Design Theory?".

Characteristic 1: The method is a close approximation of rational decision making applied to design process decisions.

While a particular implementation of P&B could be evaluated for consistency with rational decision-making, it is impossible to evaluate the method as a whole. This is because P&B does not specify how to analyze and evaluate concepts, which as mentioned earlier, is not feasible in a general sense. In addition, rational decision-making is defined within the context of a particular designer's beliefs and preferences. What can be evaluated in a general sense is whether or not following P&B necessarily leads to inconsistency and irrationality.

As noted previously, some of the selection and evaluation procedures suggested by Pahl and Beitz have been shown to lead to irrational choice. However, the authors include the caveat that designers should use their expertise to adapt the method for their particular context. If evaluation and selection procedures are used that appropriately reflect the designer's beliefs and preferences, the method will not necessarily lead to irrational choice. Thus, when applied correctly, the P&B approach can be a framework for rational design process decision-making. To do so, any aspect of P&B that is not necessary or beneficial for a particular design problem should be omitted by the designer. For example, P&B for variant design may require extensive conceptual design to identify a novel solution using an existing platform, but the embodiment and detail design phases may be shortened because of the reuse of the platform. With appropriate adaptation for particular design problems, P&B may be an adequate approximation of rational decision-making applied to design process decisions.

Characteristic 2: Heuristics to guide a design process towards maximum expected NDU.

Characteristic 3: A stopping criterion based on NDU.

Characteristic 4: An explicit consideration of uncertainty.

Characteristic 5: An explicit consideration of risk.

P&B provides guidance concerning the order of high-level decision-making in a design process, but, as mentioned in Chapter 3, this guidance is of a qualitative nature. Characteristics 2-5 require more specific quantitative guidance than is provided in such a general method. NDU in particular, is closely tied to a given design problem and the beliefs and preferences of the designer. Therefore, a general method such as P&B is not likely to be able to provide such tailored guidance.

Evaluation with respect to Characteristics 4 and 5 also requires more detail than is provided in a general method such as P&B. Since P&B does not indicate how to analyze and evaluate alternatives, the method does not prohibit explicit consideration of uncertainty and risk. But the method does not require their explicit consideration either. The method does direct designers to check for “disturbing influences” during the embodiment design phase, but this refers only to the artifact and not to uncertainties and risks in the design process itself.

Consideration of risk: risk consideration is implicit in the heuristic of gradually refining concept. You don’t want to invest in details before establishing a promising architecture.

Characteristic 6: Distinction between specifications and predictions.

A distinction between specifications and predictions is not explicitly required in P&B, but one is not prevented from distinguishing between them either. Since there is not much guidance given on how to perform analysis and evaluation of concepts, P&B does allow for such a distinction when selecting concepts for further refinement. In fact, the authors seem to implicitly acknowledge the potential for discrepancy between specified and actual performance in the steps of embodiment design because the designer is directed to "check for errors, disturbing influences, and minimum costs". Acknowledging a distinction between specifications and predictions is one identified characteristic that could be incorporated into the P&B method without loss of generality.

Characteristic 7: Interpretation of requirements as concept specifications only.

The P&B **Requirements List** is not consistent with RDT because it contains preference information, often referred to as the problem definition, in addition to concept specifications. As mentioned in Chapter 3, the problem definition in RDT is always the same: maximize expected NDU. The P&B **Requirements List** is only consistent with this problem definition if all requirements are considered as concept specifications.

Recall that in the RDT conceptual model, all synthesis actions result in the addition of new concepts to the AIS. The Requirements List in P&B is a representation of chosen concepts only. To be consistent with the RDT notion of concept specification, a **Requirements List** would be needed for each concept considered during the design process.

Even though the P&B method does not meet the identified characteristics of good design methods in a general sense, it has nevertheless been widely and successfully used in practice. In fact, the method itself was created from a survey of best practices in industry. Part of the problem in evaluating P&B with respect to these characteristics is the general nature of the method; i.e., there is too much loss of detail in abstracting away the specifics to evaluate the design process decision-making. The success and popularity of P&B indicates that the guidance, though qualitative in nature, does provide substantial help to designers. One probable advantage of following such a method is that it reduces the number of possible design actions to consider when making decisions about the design process. In the time that it would take to consider a possibly infinite number of design actions, a designer can implement many steps of P&B and be much closer to achieving some utility from generating a new artifact.

4.4 THESIS ROADMAP

In this chapter, the P&B design method has been reviewed and evaluated in the context of RDT and its accompanying conceptual model. The analysis with respect to the RDT conceptual model reveals that most of the activities in P&B are synthesis actions specifying more and more detail about the selected concepts. In each phase a different type of information is considered to transform a nebulous task into a detailed design solution. Several synthesis-analysis-evaluation loops are employed to successively refine first the principle, then the layout, and finally the production of the design solution. Although analysis and evaluation are indicated several times throughout the method, little specific guidance is provided on how to perform those activities. This lack of specific guidance is attributed to the general nature of the method, and the authors indicate that designers should use their expertise and domain knowledge to appropriately particularize P&B for individual design problems.

The evaluation of P&B with respect to the identified characteristics of good design methods is inconclusive. This is because many of the characteristics call for explicit and quantitative guidance and P&B is too general of a method to provide such detail. However, there are no structural limitations within P&B that prevent a particular implementation of P&B from meeting the identified characteristics. When appropriately adapted for particular designers and design problems, P&B is likely to meet many of the identified characteristics, but this cannot be shown in a general sense. To demonstrate the evaluation of specific design methods, two example problems are presented in the next chapter. These examples demonstrate the quantitative evaluation capabilities of RDT.

CHAPTER 5: QUANTITATIVE APPLICATIONS OF RATIONAL DESIGN THEORY: TWO EXAMPLES

Having presented Rational Design Theory in the previous chapter, the quantitative evaluation and comparison abilities of RDT are demonstrated in this chapter. In two separate examples, decision analysis is applied to the design process decision of whether or not to gather additional information about an uncertain parameter. In each example, the available alternatives are compared using decision analysis of a decision tree that incorporates all sequences of the available design actions. In addition, the design method of using Value of Information (VoI) analysis to determine the next step is evaluated by comparing the results of the VoI method with the results of the more comprehensive decision analysis. The examples and ensuing discussion in this chapter serve to validate the second secondary hypothesis that RDT enables the quantitative comparison of the quality of design methods (H2).

In the next section, some background information relevant to both examples is presented. The first example involving a manager's decision between two product proposals is presented in Sections 5.2 and 5.3. The second example of material selection in a pressure vessel design is presented in Section 5.4. For each example, the available alternatives are first compared to one another using decision analysis of a full decision tree with all possible sequences of design actions. Thereafter, the results of that analysis are compared to the predictions given by Value of Information analysis. In Section 5.5 a short discussion is presented to compare and contrast the results from both examples. The chapter concludes in Section 5.6 with a return to the thesis roadmap.

5.1 BACKGROUND INFORMATION FOR BOTH EXAMPLES

Both examples presented in this chapter involve a decision about gathering more information in design. In each example two concepts have been specified, their outcomes have been predicted, and those outcomes have been evaluated. The design actions available to the designers are (1) analysis coupled with evaluation and (2) ending design with the selection of one or the other concept. In considering the analysis/evaluation alternative, the designer must make a tradeoff between reducing the uncertainty in the concept predictions and expending resources to do so. In previous work, design researchers have studied this tradeoff between the artifact and the design process using Value of Information Theory. Thus, in these examples the decision rule of pursuing the analysis with the largest value of information is evaluated. An overview of Value of Information and related work in design is presented next in Section 5.1.1. Thereafter, some details of the mathematical formulation common to both examples are presented in Section 5.1.2.

5.1.1 Value of Information Analysis

The basic premise of the value of information is that the improvement in decision-making ability afforded by additional information can be valued by comparing the decision outcomes with and without the additional information. Howard was the first to introduce value of information and used the concept to determine the optimal number of samples to refine the parameters of a known probability distribution (Howard 1965, 1966b). Building on this foundation, Matheson used value of information to determine which sources of information to consult for a particular decision problem (Matheson 1968).

The value of a particular information source is inherently linked to the decision situation, specifically, the decision alternatives and the information sources under consideration. For a particular decision, the value of information is zero if the information does not result in any change in the optimal decision alternative; it is positive otherwise. Given that the value of information is only useful *before* the decision is made while the true value of the information can only be determined *after* all uncertainties are resolved, one can only calculate the *expected* value of information. The expected value of perfect information (EVPI) can be used if the information source resolves all uncertainty, but usually one must settle instead for the expected value of *imperfect* information (EVII) (Clemen 1996). This is computed by taking the expectation over the remaining uncertainty. The EVII is then simply the difference between the expected utility with the added information and the expected utility without the added information, as summarized in the following equation,

$$\mathrm{E}_y \mathrm{E}_{x|y} \upsilon(x, y) = \mathrm{E}_y \mathrm{E}_{x|y} \left[\pi(x, a_y) - \pi(x, a_0) \right] \tag{5.1}$$

where y is the message received from the information source, x is the state of the world, a_y is the action taken with the new information, a_0 is the action taken without the new information, and π is the payoff function (Lawrence 1999).

Within the design community, Radhakrishnan and McAdams use value of information for model selection (Radhakrishnan, et al. 2005). They proposed to incorporate the costs of design as a function of the levels of abstraction in the models; however, they do not provide a method for computing the value of information associated with a particular model. Panchal and coauthors use the value of information to guide decision-making on model refinement in design (Panchal, et al. 2008). They propose a value-of-information-based approach for stepwise refinement of simulation models by considering both variability and imprecision in the model, since imprecision can be improved with refined models whereas variability is inherent. Demonstrating their approach on a pressure vessel example and an example of the design of a multifunctional material, the authors note that the marginal improvement of each step of simulation refinement decreases as refinement progresses; however, the authors do not explicitly include the cost of the development of the new simulation models in their approach. It seems reasonable to assume that increasingly refined simulation models will cost more to develop and execute, quickly overshadowing the marginal improvement in decision-making ability.

Value of information theory is also used in design with respect to the collection of information in support of an artifact decision; however, the process-related information collection decision is viewed as a separate sub-decision from the artifact decision. Ling and coauthors argue for this separated approach in an example on the design of a pressure vessel with unknown material strength (Ling, et al. 2005, Ling, et al. 2006). They use imprecise probabilities to separately model the imprecision and variability in material strength. Assuming that the true material strength is normally distributed but with unknown parameters, they conduct a computational experiment referencing an omniscient supervisor to bound the value of information of additional material strength samples. A decision policy is needed to resolve this bound on the value of information in order to make the decision concerning whether or not to gather the information. Schlosser and Paredis take a similar approach using imprecise probabilities and probability bounds analysis in the design of an electric vehicle (Schlosser, et al. 2007). Using a model with multiple sources of uncertainty, they use optimization to find the amount of additional information that should be acquired for each uncertain parameter. Although they improve upon Ling and coauthors' work by considering multiple sources of information at once, they assume that the additional information reduces intervals for each uncertain parameter around stationary mean values. Due to this assumption and the symmetrical payoff function used, the large value of information predicted by their optimization is not realized because the decision regarding the number of batteries does not change.

Bradley and Agogino propose the Intelligent Real Time Design methodology (Bradley, et al. 1994). Their method incorporates expected utility decision-making and the value of information to guide information collection in the component selection process in which components are selected from a catalog using an uncertain evaluation function. IRTD both enables the selection of the best component and provides bounds on the value of reducing each of several sources of uncertainty. Wood and Agogino build upon this work to develop Decision-Based Conceptual Design (DBCD), a normative methodology for navigating the design space in conceptual design while considering the value of information (Wood, et al. 2005). In their method, existing design data is used to populate a design space while also incorporating uncertainty in the artifact alternatives represented therein. Demonstrating with an example of motor selection, they use expected value to decide when to narrow the range of values for design parameters and the expected value of perfect information to determine when to refine the objective function (i.e., the evaluation model). Like other work based on value of information theory, DBCD does not compare the cost of new information to the benefits of the new information to determine in an absolute sense if the new information is worth the cost. Rather, DBCD guides designers towards areas of the design space that maximize the expected value of perfect information. This approach is appropriate only if one assumes that "do nothing" is not a decision alternative available to the designer.

In many of the previous studies, the decision to gather more information in support of a design decision is seen as a separate sub-decision. As such, the information acquisition decisions are formulated separately from artifact parameter decisions (Wood, et al. 2005, Ling, et al. 2006). This perspective can be further improved by formulating the decision problem in terms of the design process. If gathering additional information is indeed an alternative available to the designer, then that alternative should be included along with artifact parameter alternatives, not formulated as a separate sub-decision. Thus, the information gathering and design decisions are considered concurrently as one decision problem. This can be accomplished by formulating the decision in terms of the design process rather than the designed artifact and including information-gathering tasks as decision alternatives. This approach provides a richer understanding of the decision problem because it enables the consideration of multiple information-gathering tasks at once as well as sequences of information-gathering tasks. In addition, the direct comparison of artifact parameter alternatives and information gathering alternatives in a single decision problem provides an absolute measure of the value of the information gathering tasks. To achieve this direct comparison, it is necessary to associate a cost with the information gathering alternatives.

5.1.2 Modeling Beliefs about Outcomes of Design Analysis Activities

In both examples in this chapter, analysis/evaluation alternatives are considered, hereafter referred to as Analyze alternatives. To predict the outcomes of the Analyze alternatives, the designer must express her beliefs about the outcome of the material test as well as how the outcome of the test will impact her beliefs about the true strength of the material. Both analyses have two important characteristics: cost and quality. Cost is defined in dollars and impacts the realized profit. The quality is given as a margin of error. It is assumed that the result of the analysis is the true value of the parameter plus a random error term, where the random error term is also distributed normally and has a mean of zero. The margin of error is interpreted as 3σ. This relationship is shown in the following equation,

$$p_\alpha = p + \varepsilon,\ \varepsilon \sim N(0, \sigma_\alpha) \tag{5.2}$$

where p_α is the analysis result in parameter units, p is the true parameter value, ε is the random error term, and σ_α is the standard deviation of that error term.

In both examples, the designer's prior beliefs are modeled as normal distributions. Normal distributions are assumed in order to simplify the computation of posterior distributions conditional on the outcome of the analyses. To incorporate the designer's prior beliefs with the result from the analysis, Bayes' Theorem is used to compute posterior probabilities. The analysis enables the designer to update her prior knowledge about the parameter. In Bayesian terms the prior knowledge is called the prior distribution and the updated knowledge is called the posterior distribution. Due to the properties of the normal distribution, the posterior probability is also normally distributed, however the mean varies as a function of the value from the analysis, p_α. The derived mean and standard deviation of the posterior probability are shown in the following equation,

$$f_{a|A=\alpha}\left(p_\alpha\right) \sim N\left(\frac{\sigma_a^2 p_\alpha + \sigma_\alpha^2 \mu_a}{\sigma_a^2+\sigma_\alpha^2}, \frac{\sigma_a^2\sigma_\alpha^2}{\sigma_a^2+\sigma_\alpha^2}\right) \tag{5.3}$$

where σ_a^2 is the variance of the prior distribution of artifact concept A and μ_a is the mean of the prior distribution of artifact concept A.

The purpose of these examples is to illustrate how decision analysis of design process decisions can provide insight into rational design practice by analyzing the tradeoff between design phase costs and overall profit. Although these are simplified examples for clarity of presentation, there are many similar scenarios of design process decision-making that can be analyzed in a similar manner. Other scenarios include selecting a number of samples to create a surrogate model of artifact performance and deciding whether to perform physical experiments or build physical prototypes. These examples are not intended to be a general approach for all types of design process decisions; rather, they are intended to showcase the possibilities of decision analysis for learning about good design process decision-making. The defining characteristics of the examples are a choice between two concepts with uncertain outcomes, an analysis that can be performed to reduce the uncertainty of either concept with known cost and quality, and a preference model based on profit.

Some may argue that the assumption of an analysis with known cost and quality is unrealistic. In the context of this work, an *analysis* is a source of *information* that changes the beliefs of the decision maker.[7] Although the cost of an analysis may be uncertain, a

[7] This definition establishes an analysis as an *information source* in agreement with the definition of *information* adopted by Ling, Aughenbaugh, and Paredis in (Ling, et al. 2006). Their definition of information differs from that of Lawrence (Lawrence 1999).

deterministic cost of analysis is used here as a first step. As for the known quality of the analysis, I believe that designers undertake steps in a design process with some expectation about the outcomes. As such, designers can formulate those beliefs about the outcomes of analyses as subjective probabilities.

5.2 MANAGER'S DECISION EXAMPLE: ONE ANALYSIS

A manager has been presented with two preliminary product proposals and must decide which project to fund. The expected earnings of the first product are within \$7M of a mean of \$8M. The second product is expected to earn \$15M plus or minus \$10M. Thus the manager has a choice between modest earnings with lower uncertainty and higher earnings with more uncertainty.

The marketing team is available to analyze the product proposals to provide improved estimates on the expected earnings. From experience, the manager has found that the earnings projection from the marketing analysis is usually within \$1.25M of the value that is actually achieved, but the marketing analysis has significant costs. Assigning the marketing team to analyze either one of the product proposals will cost the manager \$33,000.

Given this situation, the manager must decide whether to simply select one of the products using the information at hand or spend additional money to gather more information. It is assumed that the manager must choose one and only one product proposal to fund.

This business scenario is similar to many engineering design decisions. For example, in the design of a new car model the design team will want to estimate the gas mileage of the new vehicle. One could assume that the profit on the car model is correlated linearly with the gas mileage. The team is considering physical testing or computer simulations to more accurately determine the gas mileage of the car concepts. As another example, consider a team designing a pressure vessel. Two new materials are available to use, but the estimated strengths of these materials vary considerably. Given that the company will make a greater profit from a stronger pressure vessel, the team wants to select the material with the greater strength. Physical testing of the new materials can be performed, but the costs in terms of both time and money are significant.

5.2.1 Decision Analysis of the Manager's Decision

The first step in decision analysis is to identify the decision alternatives. The manager has four alternatives: *Select* product A, *select* product B, *analyze* product A, and *analyze* product B. Selecting either product represents ending design an proceeding with that product's proposal. Analyzing either product means that the designer orders the marketing to provide an improved estimate of the earnings and he can delay his decision between the two products until after the test results are in. Thus, each analysis alternative is followed by another decision after receiving the analysis result. The decision alternatives of these subsequent decisions are the same as the first decision with the exception that it is assumed the designer will not analyze a product twice.

The second step of decision analysis is to predict the outcomes of each decision alternative. These outcomes reflect the beliefs of the designer about the actual earnings of each product. The manager has beliefs about the product earnings based on his experience and the values reported from the product proposals. Because the true earnings are uncertain, a range of possible outcomes are possible. The manager formulates his beliefs about the earnings as normal probability distributions. The resulting overlapping probability distributions are shown graphically in Figure 5.1. As shown, the expected value of product A is somewhat smaller than the expected value of product B. However, the variation in product A is less than the variation in product B. The overlap in the two distributions indicates that there are possible outcomes in which product A may result in greater earnings than product B, even though the expected value of product B is higher. From this initial look, it appears that selecting product B is the preferred alternative; however, the option to perform the marketing analysis to reduce uncertainty in the earnings must also be considered.

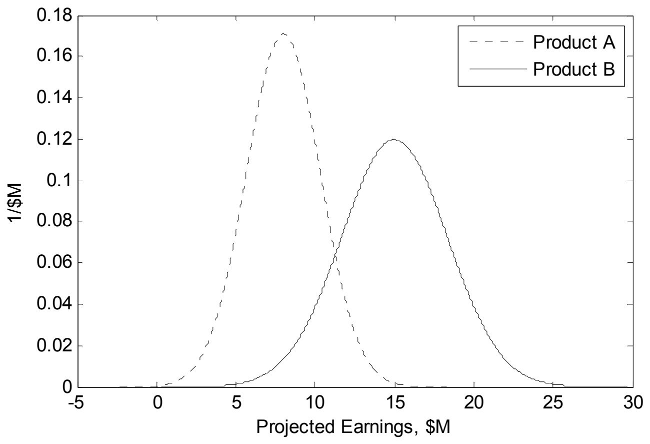


FIGURE 5.1. PROBABILITY DENSITY FUNCTIONS FOR PRODUCTS A AND B.

The outcomes of the Analyze alternatives are predicted as discussed in Section 5.1.2. Cost is defined in dollars and affects the project by reducing the earnings by that amount. As stated previously, the manager knows from experience that the marketing analysis will give a result with a margin of error of $1.25M. The margin of error is interpreted as 3σ, and thus the standard deviation of the random error term is approximately $420,000. This relationship is shown in the following equation,

$$v_\alpha = v + \varepsilon,\ \varepsilon \sim N\left(0, \sigma_\varepsilon\right) \tag{5.4}$$

where v_α is the predicted earnings in dollars from the marketing analysis, v is the true earnings, ε is the random error term, and σ_ε is the standard deviation of that error term. Bayes' Theorem is used to compute the posterior probabilities of the earnings of products A and B given the results of the marketing analysis. Due to the properties of the normal distribution, the posterior probability is also normally distributed. The derived mean and standard deviation of the posterior probability are shown in equation (5.5),

$$f_{a|A=\alpha}\left(v_\alpha\right) \sim N\left(\frac{\sigma_a^2 v_\alpha + \sigma_\varepsilon^2 \mu_a}{\sigma_a^2 + \sigma_\varepsilon^2}, \frac{\sigma_a^2 \sigma_\varepsilon^2}{\sigma_a^2 + \sigma_\varepsilon^2}\right) \tag{5.5}$$

where σ_a^2 is the variance of the prior distribution of the earnings of product A and μ_a is the mean of the prior distribution of the earnings of product A.

The third step of the decision analysis is to elicit the preferences of the manager with respect to risk in a utility function. Preferences are elicited with respect to outcomes in terms of the earnings. The risk attitude of the manager is formalized in an exponential utility function with risk aversion parameter, *R* in equation (5.6). Increasing *R* above zero represents increasing degree of risk aversion. *R* equal to zero represents risk neutrality. For normal distributions and exponential utility, there is an analytical solution to the integral for the expected utility, as shown in equation (5.7) where μ_v and σ_v^2 are the mean and variance of the value in dollars.

$$u(v) = \frac{1 - e^{-Rv}}{R} \tag{5.6}$$

$$E\left[u\left(v\right)\right]=\frac{1-e^{-R\mu_v+\frac{1}{2}\sigma_v^2R^2}}{R} \tag{5.7}$$

The fourth and final step of the decision analysis is to compute the expected utilities of each decision alternative. This can be done by "rolling back" the branches of the designer's decision tree in a process known as backward induction. The manager's decision is represented graphically in a decision tree in Figure 5.2. In a decision tree, boxes represent decisions and circles represent chance events. Arcs emanating from decision boxes represent decision alternatives, whereas arcs emanating from chance events represent outcomes. Performing an analysis on one of the products enables the DM to update his prior knowledge about that product based on the outcome of the analysis. In this decision tree, a and b represent the earnings of products A and B. The estimated values from the marketing analysis are represented by α and β. The functions under the chance events represent probability density functions, where $f(a)$ is the prior distribution of the earnings of product A, $f(b)$ is the prior distribution of the earnings of product B, $f_A(\alpha)$ and $f_B(\beta)$ are the marginal distributions of the outcomes of the marketing analysis on product A and B, and $f_{a|A=\alpha}(a)$ and $f_{b|B=\beta}(b)$ are the posterior distributions of the earnings of products A and B given the values from the marketing analysis. Additionally, $u(\cdot)$ represents the utility of the argument and $c(\cdot)$ represents the cost of running the marketing analysis.

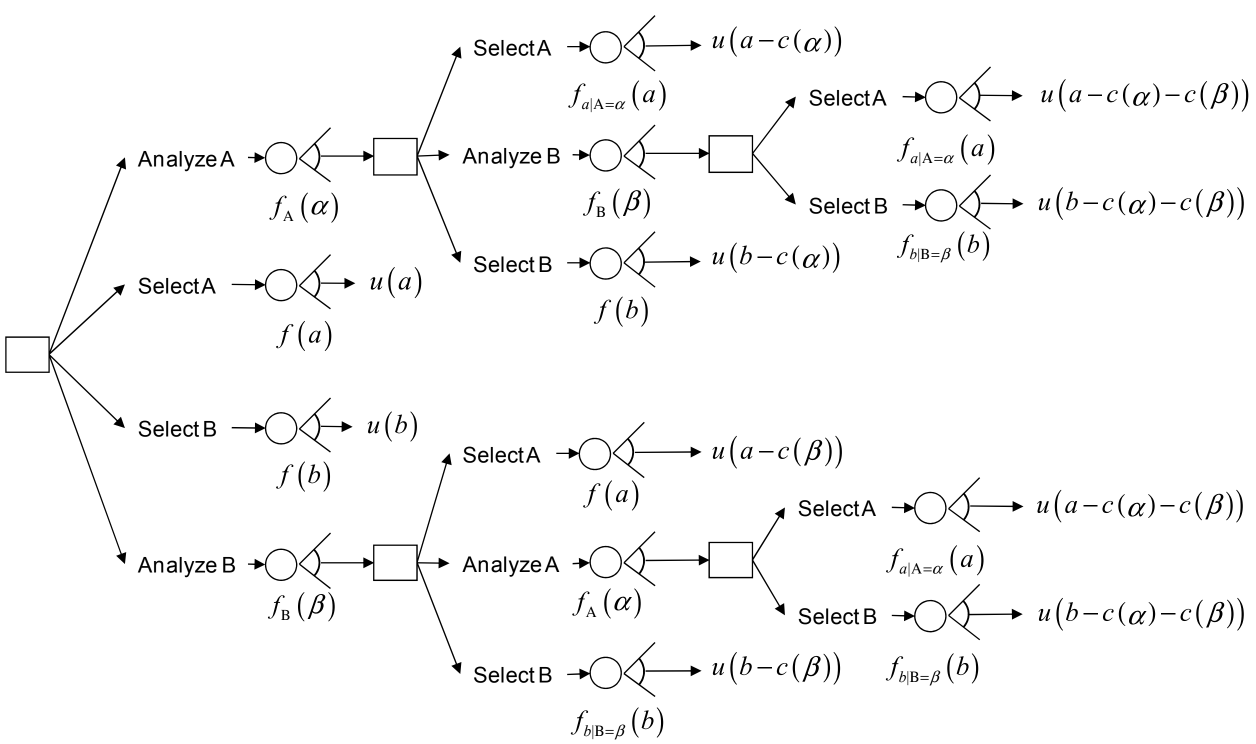


FIGURE 5.2. DECISION TREE FOR THE MANAGER'S DECISION PROBLEM

According to utility theory, the manager should choose the decision alternative which has the highest expected utility. The expected utility of each alternative can be computed by rolling back the branches of the decision tree. Chance events are rolled back by computing the expectation over the outcomes, and decision boxes are rolled back by taking the maximum value over the decision alternatives. For the manager's decision problem, equations (5.8) and (5.9) are used to compute the expected utility of the *Select* alternatives while equations (5.21) and (5.22) are used to compute the expected utility of the *Analyze* alternatives. Equations (5.10) through (5.20) define intermediate quantities which are referred to in equations (5.21) and (5.22). In these equations the subscripts on the utility refer to either the decision alternative of a particular decision node or to a decision node itself. Additionally, the analyses that have been performed are indicated in the subscripts on the utility. For example, $u_{D|\alpha\beta}$ represents the utility of the decision node at which the analysis has been performed on both products and $u_{\mathrm{SB}|\alpha}$ represents the utility of selecting product B after the analysis has been performed on product A.

$$\mathrm{E}\left[u_{\mathrm{SA}}\right] = \int_{-\infty}^{\infty} f(a) u(a)\, da \tag{5.8}$$

$$\mathrm{E}\left[u_{\mathrm{SB}}\right] = \int_{-\infty}^{\infty} f(b) u(b)\, db \tag{5.9}$$

$$\mathrm{E}\left[u_{\mathrm{SA}|\alpha\beta}(\alpha)\right] = \int_{-\infty}^{\infty} f_{\mathrm{a}|\mathrm{A}=\alpha}(a) u\left(a - c(\alpha) - c(\beta)\right) da \tag{5.10}$$

$$\mathrm{E}\left[u_{\mathrm{SB}|\alpha\beta}(\beta)\right] = \int_{-\infty}^{\infty} f_{\mathrm{b}|\mathrm{B}=\beta}(b) u\left(b - c(\alpha) - c(\beta)\right) db \tag{5.11}$$

$$\mathrm{E}\left[u_{\mathrm{D}|\alpha\beta}(\alpha,\beta)\right] = \max\left(\mathrm{E}\left[u_{\mathrm{SA}|\alpha\beta}(\alpha)\right], \mathrm{E}\left[u_{\mathrm{SB}|\alpha\beta}(\beta)\right]\right) \tag{5.12}$$

$$\mathrm{E}\left[u_{\mathrm{SA}|\alpha}(\alpha)\right] = \int_{-\infty}^{\infty} f_{\mathrm{a}|\mathrm{A}=\alpha}(a) u\left(a - c(\alpha)\right) da \tag{5.13}$$

$$\mathrm{E}\left[u_{\mathrm{SB}|\alpha}(\alpha)\right] = \int_{-\infty}^{\infty} f(b) u\left(b - c(\alpha)\right) db \tag{5.14}$$

$$\mathrm{E}\left[u_{\mathrm{AB}|\alpha}(\alpha)\right] = \int_{-\infty}^{\infty} f_{\mathrm{B}}(\beta)\, \mathrm{E}\left[u_{D|\alpha\beta}(\alpha,\beta)\right] d\beta \tag{5.15}$$

$$\mathrm{E}\left[u_{\mathrm{D}|\alpha}(\alpha)\right] = \max\begin{pmatrix} \mathrm{E}\left[u_{\mathrm{SA}|\alpha}(\alpha)\right], \\ \mathrm{E}\left[u_{\mathrm{SB}|\alpha}(\alpha)\right], \\ \mathrm{E}\left[u_{\mathrm{AB}|\alpha}(\alpha)\right] \end{pmatrix} \tag{5.16}$$

$$\mathrm{E}\left[u_{\mathrm{SA}|\beta}(\beta)\right] = \int_{-\infty}^{\infty} f(a) u\left(a - c(\beta)\right) da \tag{5.17}$$

$$\mathrm{E}\left[u_{\mathrm{SB}|\beta}(\beta)\right] = \int_{-\infty}^{\infty} f_{\mathrm{b}|\mathrm{B}=\beta}(b) u\left(b - c(\alpha)\right) db \tag{5.18}$$

$$\mathrm{E}\left[u_{\mathrm{AA}|\beta}(\beta)\right] = \int_{-\infty}^{\infty} f_{\mathrm{A}}(\alpha)\, \mathrm{E}\left[u_{D|\alpha\beta}(\alpha,\beta)\right] d\alpha \tag{5.19}$$

$$\mathrm{E}\left[u_{\mathrm{D}|\beta}(\beta)\right] = \max\begin{pmatrix} \mathrm{E}\left[u_{\mathrm{SA}|\beta}(\beta)\right], \\ \mathrm{E}\left[u_{\mathrm{SB}|\beta}(\beta)\right], \\ \mathrm{E}\left[u_{\mathrm{AB}|\beta}(\beta)\right] \end{pmatrix} \tag{5.20}$$

$$\mathrm{E}[u_{\mathrm{AA}}]=\int_{-\infty}^{\infty} f_{\mathrm{A}}(\alpha)\mathrm{E}\left[u_{\mathrm{D}|\alpha}(\alpha)\right]d\alpha \tag{5.21}$$

$$\mathrm{E}[u_{\mathrm{AB}}]=\int_{-\infty}^{\infty} f_{\mathrm{B}}(\beta)\mathrm{E}\left[u_{\mathrm{D}|\beta}(\beta)\right]d\beta \tag{5.22}$$

The values of all the decision parameters are summarized in Table 5.1. For parameter exploration in the next section, the problem parameters are scaled such that the distribution with the larger standard deviation is scaled to a standard normal distribution with mean of zero and standard deviation of one. These scaled values are also shown in the table.

TABLE 5.1. SUMMARY OF SCENARIO PARAMETERS

	Product Prior Distributions		**Analysis Parameters**	
	A	B	Quality	Cost
Mean ($M)	8.00	15.00		0.03
Margin ($M)	7.00	10.00	1.25	
Standard Deviation ($M)	2.33	3.33	0.42	
Normalize by σ_B				
Mean	2.40	4.50		
Standard Deviation	0.70	1.00	0.13	0.01
Shift by μ_B				
Mean	-2.10	0.00		
Standard Deviation	0.70	1.00		

The parameters of the decision problem thus defined, the expected utilities of each *Analyze* decision alternative are computed using equations (5.21) and (5.22). Numerical integration via adaptive Simpson's quadrature is used to compute the integrals (Gander, et al. 2000). The expected utilities of the *Select* decision alternatives are computed via the analytical solution in equation (5.7). For comparison, the expected utilities of each decision alternative are computed for both a risk neutral and a risk averse manager. The results are presented in Table 5.2. All of the computations are performed in Matlab, and the codes to perform the computations are included in Appendix A. To summarize the Manager's Decision, the artifact and process decisions are shown graphically in Figure 5.3 and Figure 5.4.

Manager's Artifact Decision

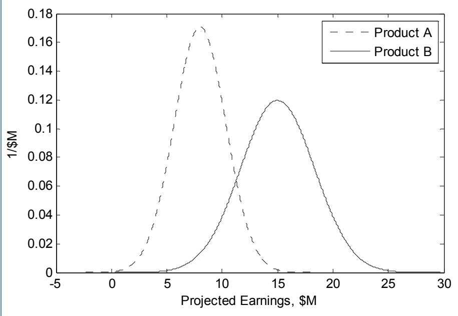
Concepts	Predictions	Decision Criteria
Product A, Product B		Expected Payoff E[π(A)]= $8M E[π(B)]= $15M

FIGURE 5.3. MANAGER'S ARTIFACT DECISION

Manager's Process Decision

Alternatives	Outcomes		Preferences
Design Actions	*Artifact Information State*	*Process Resources*	*Expected Net Payoff*
Select A	No change	$0	E[π(A)]-0
Select B	No change	$0	E[π(B)]-0
Analyze A	Updated prediction and decision criteria for product A	$33,000	E[π(A')]-$33,000
Analyze B	Updated prediction and decision criterion for product B	$33,000	E[π(B')]-$33,000

FIGURE 5.4. MANAGER'S PROCESS DECISION

5.2.2 Comparison with Value of Information Analysis

Value of information is a one-at-a-time analysis; only one message is considered at a time. The decision tree given in Figure 5.2 can be adapted to be equivalent with the value of information approach by truncating the tree after only one analysis. Then, after the analysis has been performed on either product A or product B, the manager must choose between the two products without further analysis. The equations for the value of information for this example are given in equations (5.23) and (5.24). Two equations are given because two information sources are considered. The adapted equations for the expected utility of the *Analyze* alternatives are given in equations (5.25) and (5.27).

TABLE 5.2. EXPECTED UTILITIES OF THE DECISION ALTERNATIVES IN $M, COMPARING THE FULL AND TRUNCATED DECISION TREES

	Risk Neutral $R = 0$		Risk Averse $R = 1$	
	Full	Truncated	Full	Truncated
Select A	8.00	8.00	-16.44	-16.44
Select B	15.00	**15.00**	12.84	12.84
Analyze A	15.01	14.97	13.14	12.79
Analyze B	**15.02**	14.99	**13.16**	**12.94**

$$\mathrm{E}_{\alpha}\mathrm{E}_{x|\alpha}\upsilon(x,\alpha) = \mathrm{E}[u_{\mathrm{AA}}] - \max\left(\mathrm{E}[u_{\mathrm{SA}}], \mathrm{E}[u_{\mathrm{SB}}]\right) \tag{5.23}$$

$$\mathrm{E}_{\beta}\mathrm{E}_{x|\beta}\upsilon(x,\beta) = \mathrm{E}[u_{\mathrm{AB}}] - \max\left(\mathrm{E}[u_{\mathrm{SA}}], \mathrm{E}[u_{\mathrm{SB}}]\right) \tag{5.24}$$

$$\mathrm{E}[u_{\mathrm{AA}}] = \int_{-\infty}^{\infty} f_{\mathrm{A}}(\alpha)\mathrm{E}\left[u_{\mathrm{D}|\alpha}(\alpha)\right]d\alpha \tag{5.25}$$

$$\mathrm{E}\left[u_{\mathrm{D|\alpha}}(\alpha)\right] = \max\left(\mathrm{E}\left[u_{\mathrm{SA|\alpha}}(\alpha)\right], \mathrm{E}\left[u_{\mathrm{SB|\alpha}}(\alpha)\right]\right) \tag{5.26}$$

$$\mathrm{E}[u_{\mathrm{AB}}] = \int_{-\infty}^{\infty} f_{\mathrm{B}}(\beta)\mathrm{E}\left[u_{\mathrm{D}|\beta}(\beta)\right]d\beta \tag{5.27}$$

$$\mathrm{E}\left[u_{\mathrm{D}|\beta}(\beta)\right] = \max\left(\mathrm{E}\left[u_{\mathrm{SA}|\beta}(\beta)\right], \mathrm{E}\left[u_{\mathrm{SB}|\beta}(\beta)\right]\right) \tag{5.28}$$

The expected utilities for each of the decision alternatives are shown in Table 5.2. Values are shown for the full decision tree analysis and the truncated decision tree analysis for both a risk neutral manager and a risk averse manager. The highest expected utility for each scenario is shown in boldface font.

Performing the full decision tree analysis for the risk neutral manager reveals that analyzing product B is the alternative with the highest expected utility, whereas the truncated decision tree analysis for the same manager indicates that selecting product B has the highest expected utility. Thus, for the risk neutral manager, the truncated tree analysis indicates that it is not valuable to perform the marketing analysis on product B if there is no opportunity to then perform the marketing analysis on product A. But it can be seen from the full decision tree analysis that there is a benefit to performing the marketing analysis on product B if there is a subsequent opportunity to perform the analysis on product A.

For the risk averse manager, however, the full and truncated decision tree analyses both yield the same answer: analyze product B. Although the same conclusion is reached, the full decision tree analysis gives a higher expected utility for analyzing product B than the truncated tree analysis. In fact, for both the risk neutral and the risk averse managers, the full decision tree analysis returns a higher expected utility for the *Analyze* alternatives as opposed to the truncated tree analysis, meaning that the ability to perform the subsequent marketing analysis adds value. Although the utility are higher for the full decision tree, the full and truncated tree analyses predict similar expected utilities. Therefore, VoI may be an acceptable decision rule if one is willing to accept a small amount of error in the predicted expected utility.

5.2.3 Exploration of the Effects of the Prior Distribution Parameters

Additional insight can be gained by examining the changes in the results of the decision analysis as the parameters of the manager's problem are varied. Specifically, the effects are explored of changes in prior distributions, the cost and quality of the analysis, and the risk attitude of the manager.

For particular values of analysis cost, analysis quality, and risk aversion, the boundaries between preferred decision alternatives are shown as a function of the mean and standard deviation of the prior distribution representing the earnings of product A in Figure 5.5. These boundaries are found by repeating the analysis from the previous section within a root-finding algorithm. In this plot the mean of the prior distribution representing the earnings of product B is held constant at zero while the mean of the prior distribution representing the earnings of product A varies. The standard deviation of the prior distribution representing the earnings of product B is held constant at one; thus, the prior distribution for product B is a standard normal distribution. Recall the parameter scaling mentioned earlier: all parameters are scaled such that the prior distribution with the largest standard deviation (product B in this case) is scaled to a standard normal distribution. This normalization keeps all the parameters at the same scale and enables this boundary plot to apply to many other problems in addition to the manager's decision problem. Thus, this particular plot maintains the analysis cost and quality from the manager's decision problem, but the values shown are normalized by the standard deviation of product B.

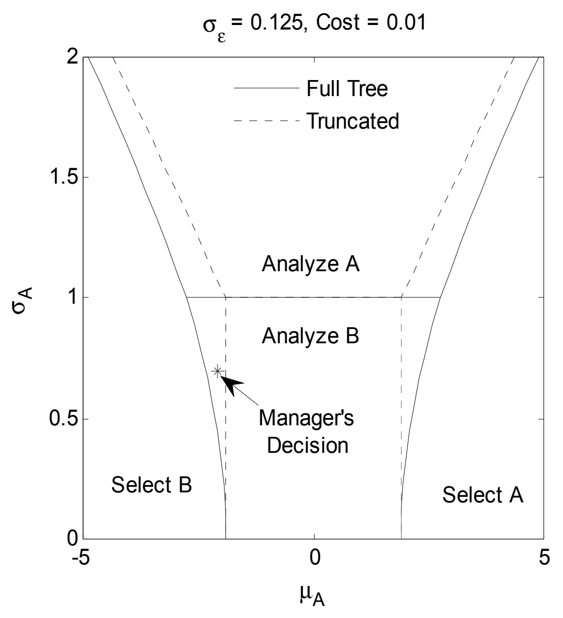


FIGURE 5.5. BOUNDARY PLOT COMPARING FULL AND TRUNCATED DECISION TREES, RISK NEUTRAL, ALL PARAMETERS IN $M

The boundaries from the full decision tree analysis are shown as solid lines, while the boundaries from the truncated decision tree analysis are shown as dotted lines. Also, the point on the plot that reflects the particular prior distributions of the manager's decision problem is indicated with a star. In the figure one sees that it is preferred to perform the analysis on product B when the standard deviation of product A is less than the standard deviation of product B. Likewise, it is preferred to perform the analysis on product A when A has the larger standard deviation. Furthermore, one sees that it is only preferred to perform an analysis when the means of the two concepts are sufficiently close to one another. Intuitively speaking, this result is expected. One should perform an analysis only when it is difficult to tell which concept is likely to be preferred because their expected utilities are close to one another. In other words, it is worth performing the analysis only when there is significant overlap in the utility distributions of the two concepts. When this is the case, the analysis should be performed on the concept with the larger amount of variation to get the most out of the analysis.

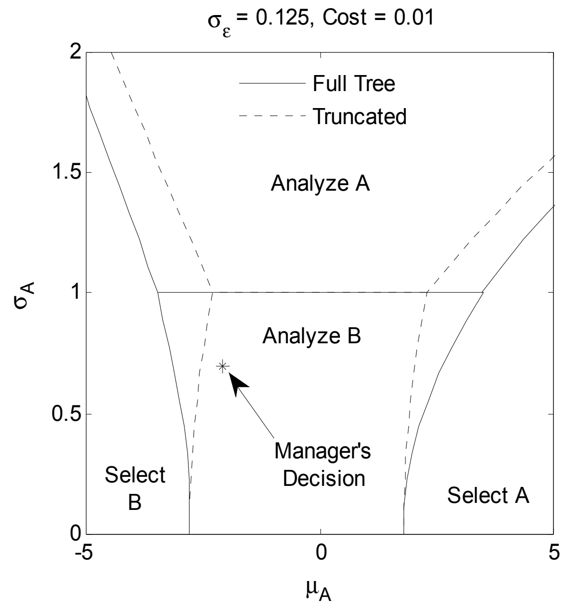


FIGURE 5.6. BOUNDARY PLOT COMPARING FULL AND TRUNCATED DECISION TREES, RISK AVERSE, ALL PARAMETERS IN $M

This boundary plot clearly shows that the full decision tree analysis results in a larger region in which it is preferred to perform an analysis. By truncating the decision tree after only one analysis, traditional value of information calculations can lead DMs to the wrong conclusion for cases in which the two products have similar standard deviations. At the bottom of the boundary plot the standard deviation of product A is equal to zero, meaning that there is no uncertainty in the outcome of product A. It makes sense that the boundaries for the full and truncated decision tree analyses would approach each other at this point, because there is no value in a subsequent marketing analysis if the predicted earnings from one product are already known precisely. When risk aversion is included, one sees a slightly different shape in the boundary plot, as shown in Figure 5.6. In the figure, one sees that risk aversion introduces a curve into the region in which analysis is the preferred alternative. The reason for the curved shape can be seen by examining the exponential utility function, specifically the formula for the expectation of the exponential utility function. In the risk averse case, the expected utility is a function of both the mean and the standard deviation, whereas the expected utility in the risk neutral case is a function of the mean only. Expected utility decreases with increasing standard deviation and increases with increasing mean; thus, to compensate for the decreasing expected utility as the standard deviation increases, the mean value at which the expected utilities between product A and product B are equal must increase. The curved shape is due to the exponential and the squared term in the expected utility. In this boundary plot one again sees that the truncated decision tree analysis gives a smaller area in which performing the marketing analysis is the preferred alternative. One can also see, however, that the manager's decision falls within the region in which the analysis is preferred according to both the full and truncated decision tree analyses.

5.2.4 Exploration of the Effects of the Analysis Cost and Quality

To explore the effect of analysis cost and quality on the boundary plots, additional boundary plots are generated for increasing cost and for decreasing quality. It is expected that the marketing analysis will no longer be preferred for any combination of prior distributions if the analysis is either too costly or has a low enough quality. The progression of increasing cost boundary plots is displayed in Figure 5.7. The progression of decreasing quality boundary plots is displayed in Figure 5.8. Note that quality of the analysis decreases as the standard deviation of epsilon increases. The boundary plot progression for a risk averse manager and increasing analysis cost is shown in Figure 5.9.

In Figure 5.7, one sees that the area of the central region in which performing the analysis is preferred decreases as cost increases. Furthermore, at some point, the region in which analysis is preferred disappears entirely from the boundary plot. Thus, at that point the cost of the analysis is too high and the manager is better off simply choosing the product with the higher expected utility. On the other hand, when there is no cost associated with the analysis, it is always preferred to run the analysis on the more uncertain product. A similar affect is seen in the progression of decreasing quality boundary plots in Figure 5.8. As the quality decreases, the area in which analysis is preferred decreases. At some point the quality of the analysis is too low, and the manager is better off choosing the product with the higher expected utility. When the quality of the analysis is as good as it can possibly be (standard deviation equal to zero) there is still a finite space in which the analysis is preferred. This is because the value of the analysis must be averaged over all the possible outcomes. Thus, even though the analysis will entirely eliminate the uncertainty in that product, one is still left with the uncertainty of what the outcome of the analysis will be given the initial uncertainty in the product. In this case, the region in which the analysis is preferred represents the cases in which the EVPI is greater than or equal to the cost of the analysis.

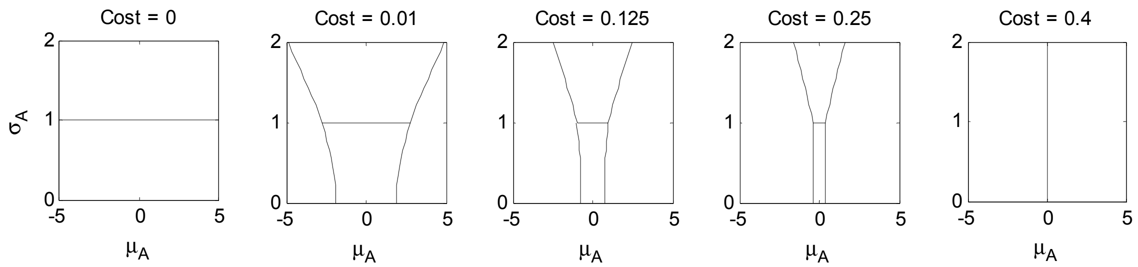


FIGURE 5.7. BOUNDARY PLOTS OF INCREASING COST, RISK NEUTRAL, PARAMETERS IN $M.

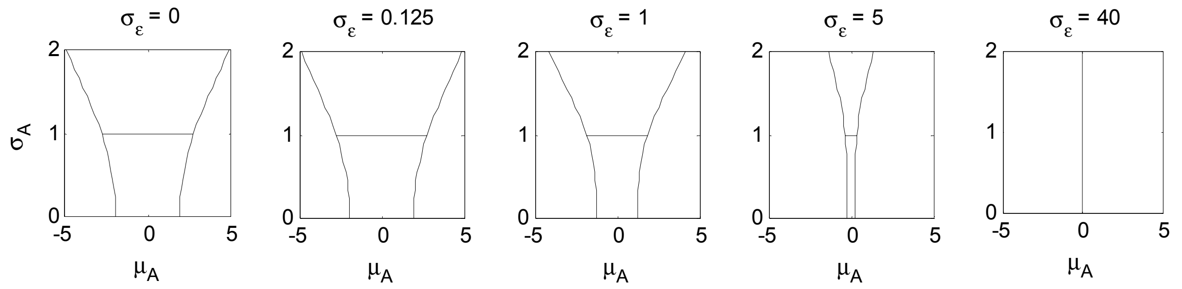


FIGURE 5.8. BOUNDARY PLOTS OF DECREASING QUALITY, RISK NEUTRAL, PARAMETERS IN $M.

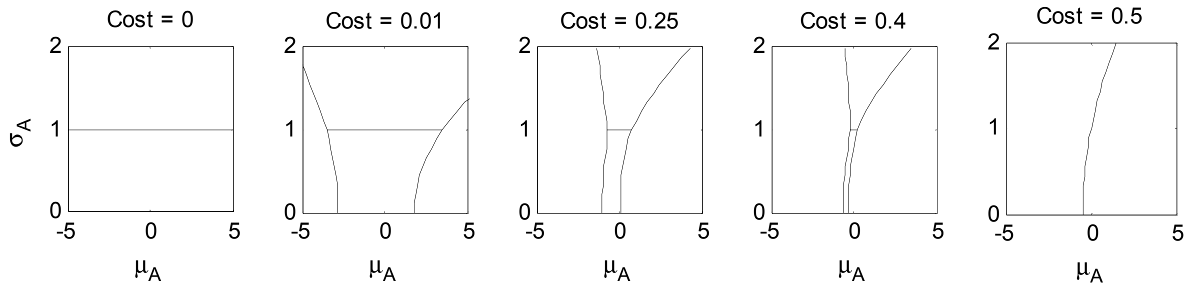


FIGURE 5.9. BOUNDARY PLOTS OF INCREASING COST, RISK AVERSE, PARAMETERS IN $M.

An interesting observation from these boundary plots is that the curvature of the boundaries diminishes as the overall area of the region in which analysis is preferred decreases with increasing cost or decreasing quality. This is significant because the curvature of the boundaries accounts for the difference between the full and truncated decision tree analyses. The conclusion drawn from this observation is that the added value of the ability to perform a subsequent analysis diminishes as the cost and/or quality of the analysis causes the desirability of the analysis to degrade.

To further characterize the impact of the analysis quality and cost on the boundary plots, root-finding is again used to find the boundary between the four-region plots and the two-region plots as a function of analysis cost and the standard deviation of epsilon. This boundary is plotted for *R* values of zero, one, and two in Figure 5.10. The upper right region represents analyses that are not worth the costs; thus, the manager can simply choose the product with the higher expected utility. This means that for the combinations of analysis cost and quality in the upper right half of the plot, the decision maker can determine whether or not to perform the analysis based solely upon the parameters of the analysis itself, independent of the prior distributions of the two concepts under consideration. The lower left region, on the other hand, represents analyses that may be worth performing depending on the values of the product prior distributions. In this case the manager should view the boundary plot to determine the preferred alternative for his particular decision scenario. Although it is useful to screen out some "no brainer" decisions on the basis of the analysis cost and quality, it may be more useful in practice to have a screening test on the basis of the prior distributions of the products, since the cost and quality of potential analyses may be unknown.

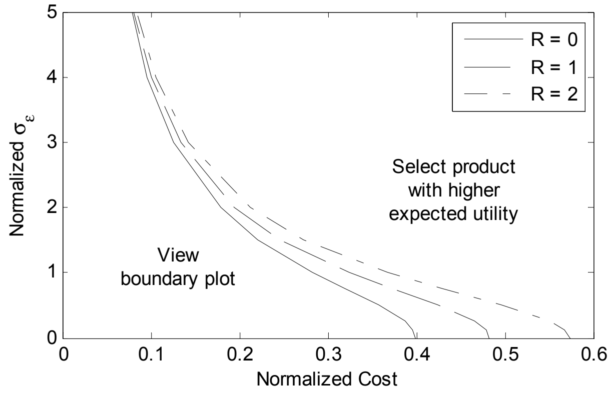


FIGURE 5.10. SCREENING TEST ON ANALYSIS COST AND QUALITY

From this plot, one sees that increasing risk aversion increases the cost at which performing the analysis is no longer worth the expense for a particular value of σ_ε. This makes sense because a risk averse manager would be willing to pay more to perform an analysis to reduce the uncertainty in the potential outcomes. Increasing the risk aversion of the decision maker effectively increases the sensitivity of the decision problem to the quality of the analysis, which is expected because a more risk averse decision maker is more sensitive to uncertainty and the analysis quality directly impacts the uncertainty in the outcomes. Because the added value of a subsequent analysis approaches zero as one approaches this boundary, as noted previously, the boundary shown in this plot is the same regardless of whether the full or truncated decision tree analysis is used to find the boundary.

5.3 MANAGER'S DECISION: TWO ANALYSES

To explore this scenario further, the manager's decision problem was augmented by considering an additional analysis that could be applied to either product. This additional analysis has the same impact on the knowledge about the product earnings as the first marketing analysis, meaning that the predicted earnings are assumed to be the true earnings plus a random error term, and shown in equation (5.2). With this additional analysis it is possible to investigate not only when analysis is preferred, but which analysis is preferred. This augmentation significantly increases the size of the full decision tree, which also increases the computational effort required to compute the expected utilities of alternatives in the full tree. Because of the large size of the decision tree, the full tree is not depicted here. The truncated tree that is used for the Value of Information computation is shown in Figure 5.11. All other aspects of the problem formulation remain the same as reported in the previous section and are not repeated here.

As in the previous section, the results from the full decision tree analysis are compared to the truncated Value of Information analysis. In Figure 5.12 and Figure 5.13, points where the two analyses disagree on the preferred alternative are depicted with an X. Similar to the one analysis case, the full and truncated tree analyses disagree at the outer edges of the central region in which analysis is preferred. In these cases, the full decision tree indicates that an analysis should be performed while the truncated tree analysis indicates that the preferred alternative is to select one of the product proposals. However, unlike in the one analysis case, there is also a central region in which the full and truncated tree analyses disagree. In this region both decision tree analysis indicate that analysis is preferred, but they disagree on which analysis is preferred. This pattern is seen in both the risk neutral and the risk averse cases. As was the case with the single analysis scenario, the expected utilities of the preferred alternatives are very close to one another regardless of which decision tree is used for the computation. Thus, although the truncated Value of Information tree sometimes predicts the wrong decision alternative, the loss in expected utility from choosing that wrong alternative is minimal.

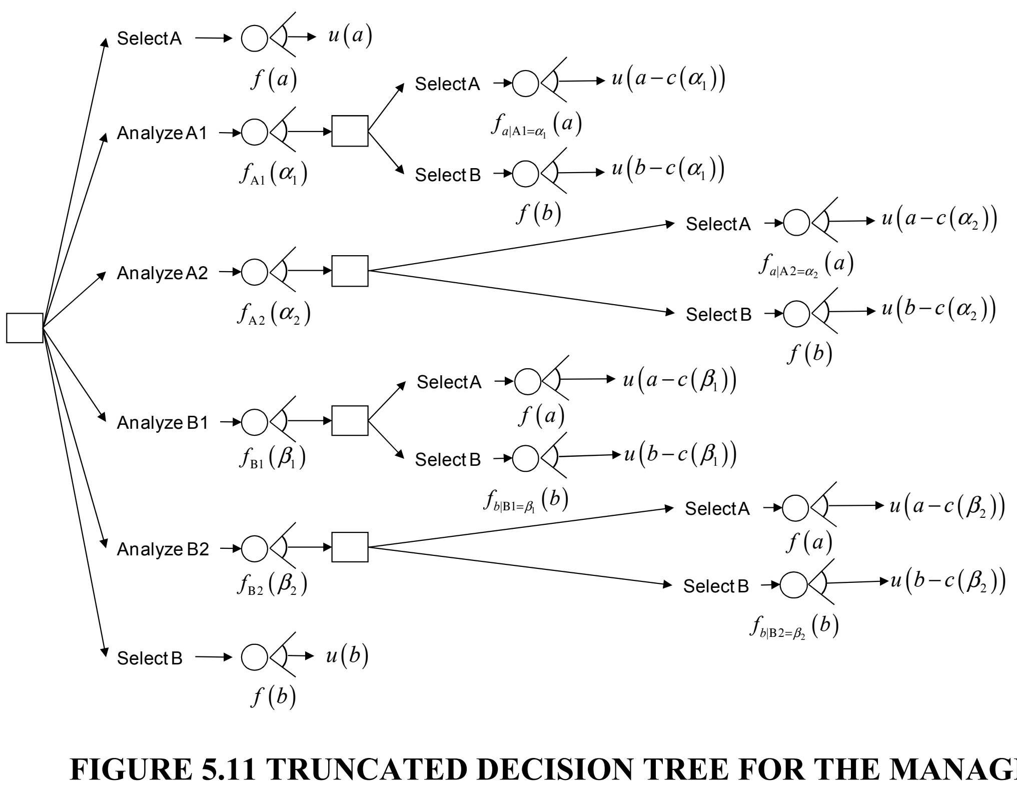


FIGURE 5.11 TRUNCATED DECISION TREE FOR THE MANAGER'S DECISION, TWO ANALYSIS CASE

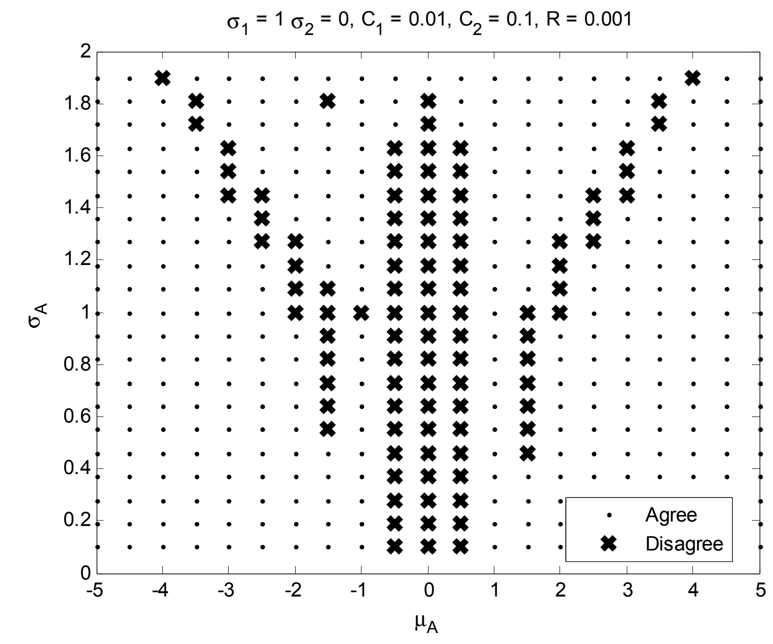


FIGURE 5.12 COMPARISON OF FULL AND TRUNCATED DECISION TREES FOR THE TWO ANALYSIS CASE OF THE MANAGER'S DECISION, RISK NEUTRAL

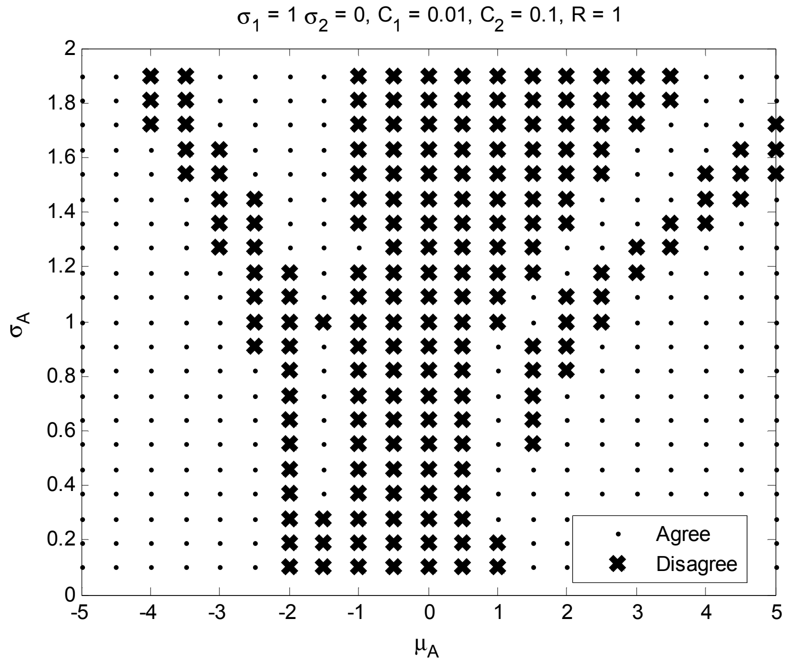


FIGURE 5.13 COMPARISON OF FULL AND TRUNCATED DECISION TREES FOR THE TWO ANALYSIS CASE OF THE MANAGER'S DECISION, RISK AVERSE

5.4 PRESSURE VESSEL DESIGN EXAMPLE

A designer of pressure vessels is choosing between materials for her company's next generation helium tank. The material supplier has suggested two materials: the material used in their other pressure vessels, which has a strength of 1000 $\pm$ 180 MPa, and a new alloy that has a lower expected strength and a larger uncertainty, 950 $\pm$ 270 MPa. The new alloy has a larger uncertainty because it has not yet been tested and characterized carefully, but it has the potential to have a much larger strength than the existing material. Both materials are available at a cost of \$8500/m^3. To reduce the uncertainty in the material strength the designer can request an additional material test. The material supplier offers to perform material strength testing and charges \$660 per test. The test is expected to give the true strength within 100 MPa.

The pressure vessel manufacturing company has an established payoff function which includes the sale price of the tank, material costs assessed on a volumetric basis, a cost for replacing the tank when failure occurs, and the costs of material strength testing, shown in the following equation[8]. The company amortizes the cost of strength testing over 100 vessels for a per-vessel analysis cost of \$6.60.

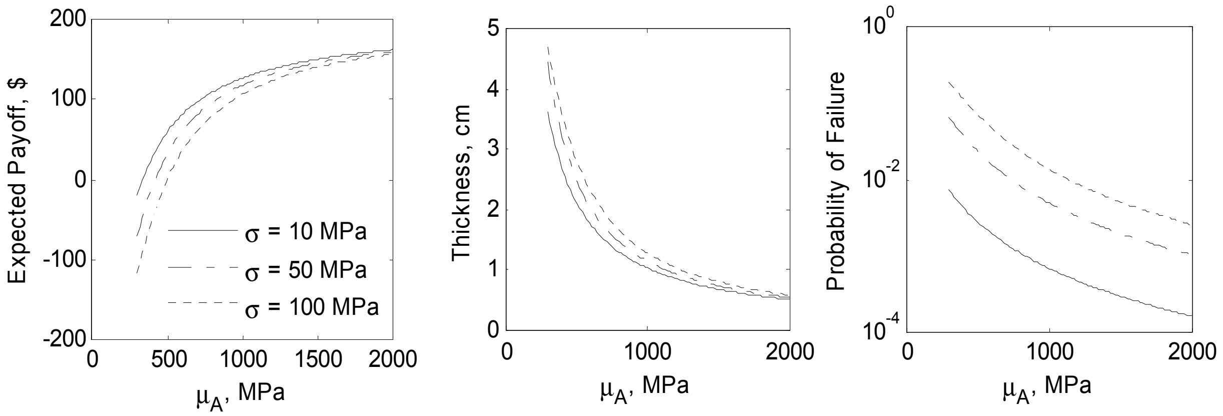
[8] This is the same payoff function used by Ling and coauthors (Ling, et al. 2006), although the failure cost is changed from \$1M to \$300 to represent the cost of replacing the vessel when it fails by leaking rather than bursting.

$$
\begin{aligned}
&\pi(a,x) = P_{selling} - C_{material} vol(a) - C_{failure}\delta(a,x) - C_{analysis} n \\
&where \quad P_{selling} \equiv \text{selling price } = \$200 \\
&\qquad C_{material} \equiv \text{material cost per volume } = \$8500/m^3 \\
&\qquad x \equiv \text{true yield strength of pressure vessel material} \\
&\qquad a \equiv \text{design variable: wall thickness} \\
&\qquad L \equiv \text{ cylinder length} = 1.2\ m \\
&\qquad r \equiv \text{ outer cylinder radius} = 10\ cm \\
&\qquad vol(a) \equiv \text{ volume of material} = \pi\left(\frac{4}{3}a^3 - (L+4r)a^2 + \left(2Lr+4r^2\right)a\right) \\
&\qquad C_{failure} \equiv \text{cost incurred if vessel fails } = \$300 \\
&\qquad \delta(a,x) \equiv \text{failure indicator} = \begin{cases} 0 & \text{if x} \geq \sigma_{\max}(\text{a}) \\ 1 & \text{otherwise} \end{cases} \\
&\qquad C_{analyis} \equiv \text{cost incurred by strength analysis per vessel} = \$6.60 \\
&\qquad n \equiv \text{ number of strength analyses performed}
\end{aligned}
\tag{5.29}
$$

In this example, it is assumed that the dimensions of the new pressure vessel are dictated by standards, with the exception of the wall thickness which must be varied according to the material strength. The wall thickness is varied to optimize expected payoff given the uncertainty in the true material strength. The equation for the expected payoff as a function of the wall thickness is given below, where p is the design pressure for the vessel, 100 MPa. The uncertain material strength only impacts the failure indicator. Thus, to compute the expectation of that portion of the payoff function the cost of failure is multiplied by the probability of failure. Since failure occurs when the maximum stress in the vessel exceeds the material strength, the probability of failure is the value of the cumulative distribution function of the material strength, F_x, evaluated at the hoop stress, σ_H, which is the maximum stress in the pressure vessel.

$$E[\pi(a,x)] = P_{selling} - C_{material}\pi\left(\frac{4}{3}a^3 - (L+4r)a^2 + (2Lr+4r^2)a\right) - C_{failure}F_x(\sigma_H) - C_{analysis}n$$
$$where \quad F_x = P(x \le \sigma_H) \tag{5.30}$$
$$\sigma_H \equiv \text{hoop stress} = \frac{pr}{a}$$

To gain a better understanding of the pressure vessel problem, the impacts of the mean and standard deviation of normally distributed material strength on the expected payoff, vessel wall thickness, and probability of failure are explored in Figure 5.14. An increase in the mean or a decrease in the standard deviation of the material strength reduces the probability of failure for a constant wall thickness. As shown in the figure, however, by maximizing the expected payoff, some of the reduction in failure cost is traded off for a reduction in wall thickness and hence material cost.



(a) (b) (c)

FIGURE 5.14. TRENDS OF EXPECTED PAYOFF (A), VESSEL WALL THICKNESS (B), AND PROBABILITY OF FAILURE (C) VERSUS MEAN AND STANDARD DEVIATION OF MATERIAL STRENGTH AT THE POINT OF MAXIMUM EXPECTED UTILITY (EQUATION 2).

5.4.1 Decision Analysis of the Pressure Vessel Decision

The first step in decision analysis is to identify the decision alternatives. This designer has four alternatives: *Select* material A, *Select* material B, *Analyze* material A, and

Analyze material B. Selecting either material represents ordering that material to begin production of the new pressure vessels. Analyzing either material means that the designer orders a test of that material and can delay her decision between the two materials until after the test results are in. Thus, each analysis alternative is followed by another decision after receiving the test result. The decision alternatives of these subsequent decisions are the same as the first decision with the exception that it is assumed the designer will not retest a material.

The second step of decision analysis is to predict the outcomes of each decision alternative. These outcomes reflect the beliefs of the designer about the actual material strength of each material and the payoff that will be realized from the sale of the resulting pressure vessels. The designer has beliefs about the material strength based on her experience and the values reported from the material supplier. Because the true value of the material strength is uncertain, a range of possible payoff outcomes are possible. The designer formulates her beliefs about the material strength as probability distributions. Normal distributions are assumed in order to simplify the computation of posterior distributions conditional on the outcome of the analyses. Based on those beliefs about the material strength, she can compute the payoff of the *Select* alternatives using equation (5.29). When propagated through the payoff function, the resulting overlapping payoff distributions are shown graphically in Figure 5.15. The top half of the plot shows the possible outcomes of the material strength, while the bottom half of the plot shows the potential payoff outcomes. As shown, the strength of material A is much more uncertain, but also has a lower mean strength than material B. Because of the large variation in the strength of material A, a greater thickness is required for vessels made from material A as compared to material B. Since less material must be used, the potential payoff using material B is higher than potential payoff using material A. From this initial look, it appears that selecting material B is the preferred alternative; however, one must also consider the option to perform the material testing to reduce uncertainty in the material strength.

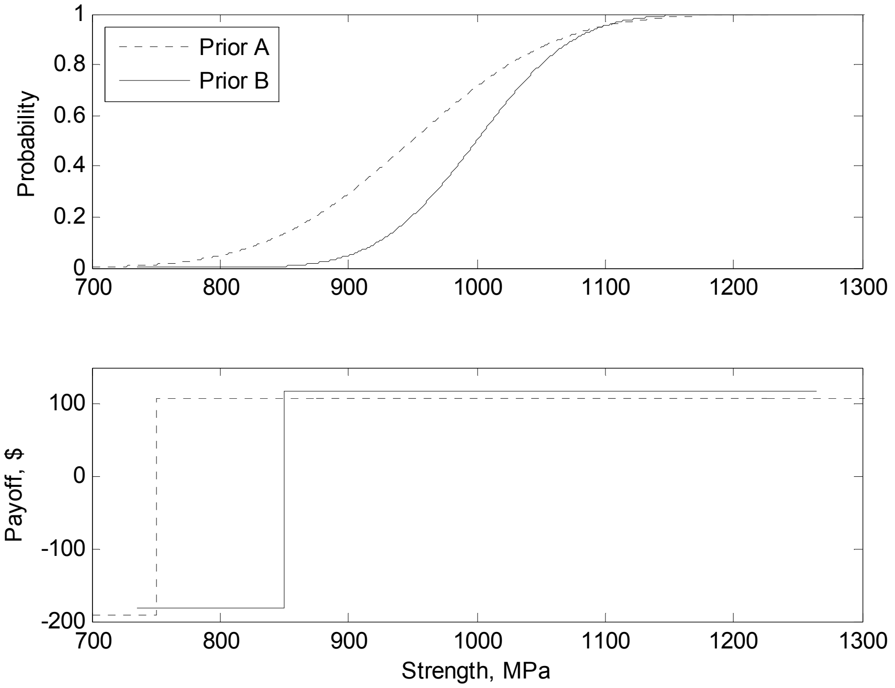


FIGURE 5.15. CUMULATIVE DISTRIBUTION FUNCTIONS FOR MATERIALS A AND B.

To predict the outcome of the *Analyze* alternatives, the designer must express her beliefs about the outcome of the material test as well as how the outcome of the test will impact her beliefs about the true strength of the material. The strength analysis has two important characteristics: cost and quality. Cost is defined in dollars and affects the project by increasing the volumetric price of the material. As for the quality of the test, the material supplier indicates that the test has an error margin of 100 MPa. The designer interprets this information mathematically by assuming that the result of the strength test is the true strength plus a random error term, where the random error term is also distributed normally and has a mean of zero. The margin of error reported by the material supplier is interpreted as 3σ, and thus the standard deviation of the random error term is 33 MPa. This relationship is shown in the following equation,

$$s_\alpha = s + \varepsilon,\ \varepsilon \sim N\left(0, \sigma_\varepsilon\right) \tag{5.31}$$

where s_α is the strength test result in psi, s is the true material strength, ε is the random error term, and σ_ε is the standard deviation of that error term.

To incorporate the designer's prior beliefs of the material strength with the result from the material strength test, Bayes' Theorem is used to compute posterior probabilities. The material strength test enables the designer to update her prior knowledge about the material. With updated beliefs about the material strength, she can also update her beliefs about the expected payoff of the pressure vessel design. Due to the properties of the normal distribution, the posterior probability is also normally distributed, however the mean varies as a function of the value from the analysis, s_α. The derived mean and standard deviation of the posterior probability are shown in the following equation,

$$f_{a|A=\alpha}\left(s_\alpha\right) \sim N\left(\frac{\sigma_a^2 s_\alpha + \sigma_\varepsilon^2 \mu_a}{\sigma_a^2 + \sigma_\varepsilon^2}, \frac{\sigma_a^2 \sigma_\varepsilon^2}{\sigma_a^2 + \sigma_\varepsilon^2}\right) \tag{5.32}$$

where σ_a^2 is the variance of the prior distribution of the strength of material A and μ_a is the mean of the prior distribution of the strength of material A. Once the posterior probability is computed using the above equation, the expected payoff using each material can be computed using equation (5.30).

The third step of the decision analysis is to elicit the preferences of the decision maker with respect to risk in a utility function. Preferences are elicited with respect to outcomes in terms of the payoff. For this example it is assumed that the designer is risk neutral, and, as a result, her utility function and payoff function are one in the same.

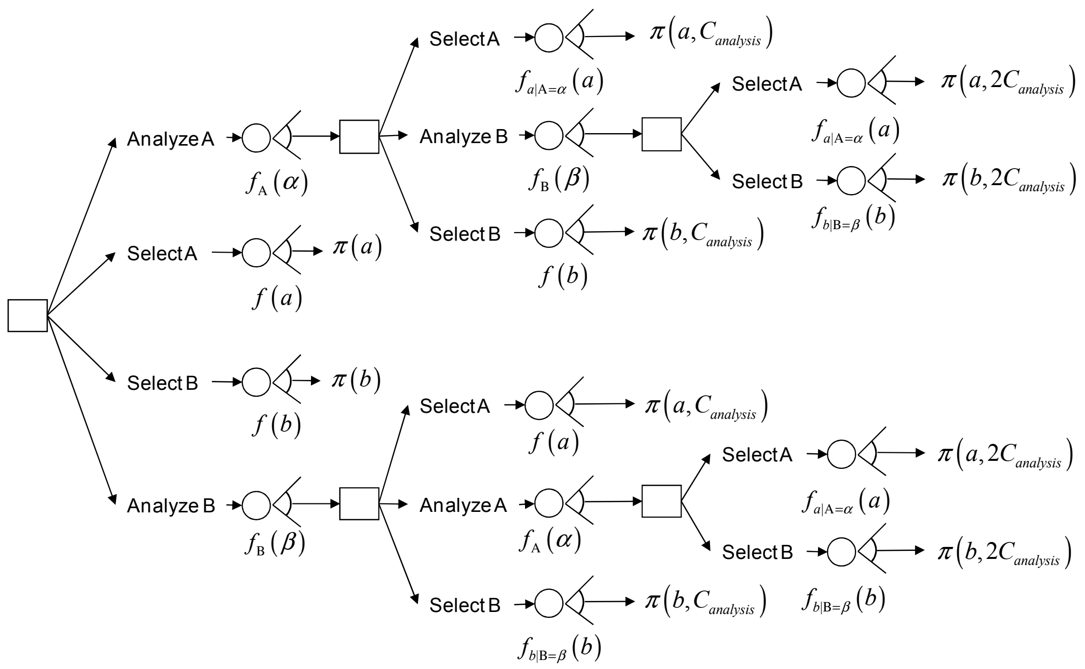


FIGURE 5.16. DECISION TREE FOR THE PRESSURE VESSEL DECISION PROBLEM

The fourth and final step of the decision analysis is to compute the expected utilities of each decision alternative. This can be done by "rolling back" the branches of the designer's decision tree in a process known as backward induction. The designer's decision is represented graphically in a decision tree in Figure 5.16. In this decision tree, *a* and *b* represent the possible strength outcomes of materials A and B. The results from the strength test are represented by α and β. The functions under the chance events represent probability density functions, where $f(a)$ is the prior distribution of the strength of material A, $f(b)$ is the prior distribution of the strength of material B, $f_A(\alpha)$ and $f_B(\beta)$ are the marginal distributions of the outcomes of the strength tests on materials A and B, and $f_{a|A=\alpha}(a)$ and $f_{b|B=\beta}(b)$ are the posterior distributions of the strength of materials A and B given the values from the strength tests. Additionally, $\pi(\cdot)$ represents the payoff as a function of the material strength.

To compute the expected utility of each decision alternative, chance events are rolled back by computing the expectation over the outcomes, and decision boxes are rolled back by taking the maximum value over the decision alternatives. For the pressure vessel decision problem, equations (5.33) and (5.34) are used to compute the expected utility of the *Select* alternatives while equations (5.46) and (5.47) are used to compute the expected utility of the *Analyze* alternatives. Equations (5.35) through (5.45) define intermediate quantities which are referred to in equations (5.46) and (5.47). In these equations the subscripts on the utility refer to either the decision alternative of a particular decision node or to a decision node itself. Additionally, the analyses that have been performed are indicated in the subscripts on the utility. For example, $u_{D|\alpha\beta}$ represents the utility of the decision node at which the strength test has been performed on both materials and $u_{SB|\alpha}$ represents the utility of selecting material B after the strength test has been performed on material A.

$$\mathrm{E}[u_{SA}] = \int_{-\infty}^{\infty} f(a)\pi(a)\,da \tag{5.33}$$

$$\mathrm{E}[u_{SB}] = \int_{-\infty}^{\infty} f(b)\pi(b)\,db \tag{5.34}$$

$$\mathrm{E}\left[u_{\mathrm{SA}|\alpha\beta}(\alpha)\right]=\int_{-\infty}^{\infty}f_{\mathrm{a|A}=\alpha}(a)\pi\left(a,2C_{analysis}\right)da \tag{5.35}$$

$$\mathrm{E}\left[u_{\mathrm{SB}|\alpha\beta}(\beta)\right]=\int_{-\infty}^{\infty}f_{\mathrm{b|B}=\beta}(b)\pi\left(b,2C_{analysis}\right)db \tag{5.36}$$

$$\mathrm{E}\left[u_{\mathrm{D|\alpha\beta}}(\alpha,\beta)\right]=\max\left(\mathrm{E}\left[u_{\mathrm{SA|\alpha\beta}}(\alpha)\right],\mathrm{E}\left[u_{\mathrm{SB|\alpha\beta}}(\beta)\right]\right) \tag{5.37}$$

$$\mathrm{E}\left[u_{\mathrm{SA}|\alpha}(\alpha)\right]=\int_{-\infty}^{\infty}f_{\mathrm{a|A}=\alpha}(a)\pi\left(a,C_{analysis}\right)da \tag{5.38}$$

$$\mathrm{E}\left[u_{\mathrm{SB}|\alpha}(\alpha)\right]=\int_{-\infty}^{\infty}f(b)\pi\left(b,C_{analysis}\right)db \tag{5.39}$$

$$\mathrm{E}\left[u_{\mathrm{AB}|\alpha}(\alpha)\right]=\int_{-\infty}^{\infty}f_{\mathrm{B}}(\beta)\mathrm{E}\left[u_{D|\alpha\beta}(\alpha,\beta)\right]d\beta \tag{5.40}$$

$$\mathrm{E}\left[u_{\mathrm{D|\alpha}}(\alpha)\right]=\max\begin{pmatrix}\mathrm{E}\left[u_{\mathrm{SA|\alpha}}(\alpha)\right],\\ \mathrm{E}\left[u_{\mathrm{SB|\alpha}}(\alpha)\right],\\ \mathrm{E}\left[u_{\mathrm{AB|\alpha}}(\alpha)\right]\end{pmatrix} \tag{5.41}$$

$$\mathrm{E}\left[u_{\mathrm{SA}|\beta}(\beta)\right]=\int_{-\infty}^{\infty}f(a)\pi\left(a,C_{analysis}\right)da \tag{5.42}$$

$$\mathrm{E}\left[u_{\mathrm{SB}|\beta}(\beta)\right]=\int_{-\infty}^{\infty}f_{\mathrm{b|B}=\beta}(b)\pi\left(b,C_{analysis}\right)db \tag{5.43}$$

$$\mathrm{E}\left[u_{\mathrm{AA}|\beta}(\beta)\right]=\int_{-\infty}^{\infty}f_{\mathrm{A}}(\alpha)\mathrm{E}\left[u_{D|\alpha\beta}(\alpha,\beta)\right]d\alpha \tag{5.44}$$

$$\mathrm{E}\left[u_{\mathrm{D}|\beta}(\beta)\right]=\max\begin{pmatrix}\mathrm{E}\left[u_{\mathrm{SA}|\beta}(\beta)\right],\\ \mathrm{E}\left[u_{\mathrm{SB}|\beta}(\beta)\right],\\ \mathrm{E}\left[u_{\mathrm{AB}|\beta}(\beta)\right]\end{pmatrix} \tag{5.45}$$

$$\mathrm{E}\left[u_{\mathrm{AA}}\right]=\int_{-\infty}^{\infty}f_{\mathrm{A}}(\alpha)\mathrm{E}\left[u_{\mathrm{D}|\alpha}(\alpha)\right]d\alpha \tag{5.46}$$

$$\mathrm{E}\left[u_{\mathrm{AB}}\right]=\int_{-\infty}^{\infty}f_{\mathrm{B}}(\beta)\mathrm{E}\left[u_{\mathrm{D}|\beta}(\beta)\right]d\beta \tag{5.47}$$

The parameters of the decision problem thus defined, the expected utilities of each *Analyze* decision alternative are computed using equations (5.46) and (5.47). Numerical integration via adaptive Simpson's quadrature is used to compute the integrals (Gander, et al. 2000). The expected utilities of the *Select* decision alternatives are computed via equations (5.33) and (5.34). The computations are performed in Matlab, and the codes used to perform the computations are included in Appendix B. To summarize the analysis, the artifact and process decisions are depicted graphically in Figure 5.17 and Figure 5.18.

Pressure Vessel Artifact Decision

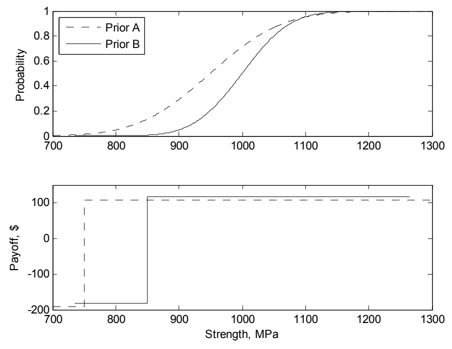
Concepts	Predictions	Decision Criteria
Material A, Material B		Expected Payoff E[π(A)]= $103.88 E[π(B)]= $116.03

FIGURE 5.17. PRESSURE VESSEL ARTIFACT DECISION

Pressure Vessel Process Decision

Alternatives	Outcomes		Preferences
Design Actions	*Artifact Information State*	*Process Resources*	*Expected Net Payoff*
Select A	No change	$0	E[π(A)]-0
Select B	No change	$0	E[π(B)]-0
Analyze A	Updated prediction and decision criteria for material A	$660	E[π(A')]-$660
Analyze B	Updated prediction and decision criterion for material B	$660	E[π(B')]-$660

FIGURE 5.18. PRESSURE VESSEL PROCESS DECISION

5.4.2 Comparison with Value of Information Analysis

Value of information is traditionally computed by examining the impacts of each message independently. Sequences of messages are generally not considered. The decision tree given in Figure 5.16 can be adapted to be equivalent with the value of information approach by truncating the *Analyze* branches after only one analysis. Then, after the analysis has been performed on either material A or B, the designer must choose between the two materials without further analysis. The equations for the value of information for this example are given in equations (5.48) and (5.49). Two equations are given because two information sources are considered. The adapted equations for the expected utility of the *Analyze* alternatives are given in equations (5.50) and (5.52).

TABLE 5.3. EXPECTED UTILITIES OF THE DECISION ALTERNATIVES IN $, COMPARING THE FULL AND TRUNCATED DECISION TREES

	Full	**Truncated**
Select A	103.88	103.88
Select B	116.03	**116.03**
Analyze A	113.30	113.29
Analyze B	**116.04**	116.02

$$\mathrm{E}_{\alpha}\mathrm{E}_{x|\alpha}\upsilon(x,\alpha)=\mathrm{E}[u_{\mathrm{AA}}]-\max\left(\mathrm{E}[u_{\mathrm{SA}}],\mathrm{E}[u_{\mathrm{SB}}]\right) \tag{5.48}$$

$$\mathrm{E}_{\beta}\mathrm{E}_{x|\beta}\upsilon(x,\beta)=\mathrm{E}[u_{\mathrm{AB}}]-\max\left(\mathrm{E}[u_{\mathrm{SA}}],\mathrm{E}[u_{\mathrm{SB}}]\right) \tag{5.49}$$

$$\mathrm{E}[u_{\mathrm{AA}}]=\int_{-\infty}^{\infty} f_{\mathrm{A}}(\alpha)\mathrm{E}\left[u_{\mathrm{D}|\alpha}(\alpha)\right]d\alpha \tag{5.50}$$

$$\mathrm{E}\left[u_{\mathrm{D}|\alpha}(\alpha)\right]=\max\left(\mathrm{E}\left[u_{\mathrm{SA}|\alpha}(\alpha)\right],\mathrm{E}\left[u_{\mathrm{SB}|\alpha}(\alpha)\right]\right) \tag{5.51}$$

$$\mathrm{E}[u_{\mathrm{AB}}]=\int_{-\infty}^{\infty} f_{\mathrm{B}}(\beta)\mathrm{E}\left[u_{\mathrm{D}|\beta}(\beta)\right]d\beta \tag{5.52}$$

$$\mathrm{E}\left[u_{\mathrm{D}|\beta}(\beta)\right]=\max\left(\mathrm{E}\left[u_{\mathrm{SA}|\beta}(\beta)\right],\mathrm{E}\left[u_{\mathrm{SB}|\beta}(\beta)\right]\right) \tag{5.53}$$

The expected utilities for each of the decision alternatives are shown in Table 5.3. Values are shown for the full decision tree analysis and the truncated decision tree analysis. The highest expected utility for each scenario is shown in boldface font.

Performing the full decision tree analysis reveals that analyzing material B is the alternative with the highest expected utility, whereas the truncated decision tree analysis for indicates that selecting material B has the highest expected utility. Note that the full decision tree analysis returns a higher expected utility for both *Analyze* alternatives as compared to the truncated tree analysis, meaning that the ability to perform the subsequent material strength analysis adds value. The conflicting results from the full and truncated decision tree analyses demonstrate that traditional value of information analysis can lead to the wrong conclusion when a sequence of information sources is available; however, there is very little difference in the utilities of the preferred alternatives.

5.4.3 Exploration of the Effects of the Prior Distribution Parameters

The pressure vessel decision problem has been analyzed for the given parameters, but additional insight can be gained by examining the changes in the results of the decision analysis as the parameters of the decision problem are varied. In this section the effects of changes in prior distributions and the cost and quality of the analysis are explored. For particular values of analysis cost, and analysis quality, the boundaries between preferred decision alternatives are shown as a function of the mean and standard deviation of the prior distribution of the strength of material A in Figure 5.5. These boundaries are found by repeating the analysis from the previous section within a root-finding algorithm. In this plot the mean of the prior distribution of the strength of material B is held constant at 1000 MPa while the mean of the prior distribution of the strength of material A varies.

The standard deviation of the prior distribution of the strength of material B is held constant at 33 MPa. This plot maintains the analysis cost and quality from the pressure vessel decision problem. For reference, the point on the plot that reflects the particular prior distributions of the pressure vessel decision problem is indicated with a star.

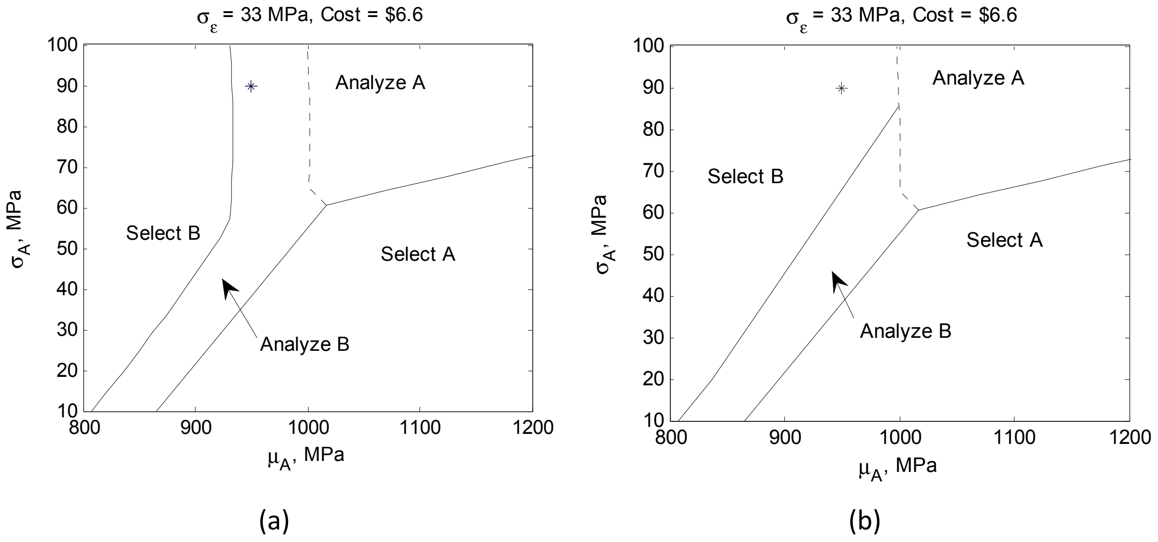


FIGURE 5.19. BOUNDARY PLOT COMPARING FULL (A) AND TRUNCATED (B) DECISION TREES, STRENGTH OF MATERIAL B ~N(1000, 60) MPA

The figure shows that it is preferred to perform the analysis on material A when the standard deviation of material A is greater than the standard deviation of material B (60 MPa) and the mean of material A is nearly equal to or greater than the mean of material B (1000 MPa). Also, the figure indicates that it is preferred to select material A when the mean of material A is larger than the mean of material B and the standard deviation is less than that of material B. It is also preferred to select material A even for some cases in which the mean of material A is less than the mean of material B given that the standard deviation of material A is less than that of material B. For all values of the standard deviation of material A, there is a band of mean values for which it is preferred to analyze material B. The pressure vessel design problem falls within this band. It is preferred to select material B for most cases where the mean of material A is less than that of material B.

Comparing the truncated decision tree plot to the full tree plot, it can be seen that the band in which analyzing material B is the preferred alternative does not extend over all values of the standard deviation of material A as is the case in the full decision tree plot. Thus, it can be concluded that analyzing material B is preferred when the standard deviation of material A is larger than that of material B only if the analysis of material B is followed by the analysis of material A. Therefore, if it is possible to analyze both materials, relying upon traditional value of information analysis can lead to the wrong conclusion about whether the analysis is worth the costs. Other than the absence of the analyze B band for standard deviations greater than B, the truncated decision tree plot is largely the same as the full decision tree plot. This indicates that much of the benefit from analysis comes from the first analysis performed, which explains why the expected utilities computed via the truncated decision tree are very close to the expected utilities computed via the full decision tree. Because of this, VoI may be an acceptable decision rule if some error can be tolerated.

To understand the meaning of these plots, recall that performing the analysis both reduces the uncertainty in the material strength and reduces the expected payoff by the cost of the test. Thus, for analysis to be preferred, the increase in expected payoff from the reduction in uncertainty must outweigh the cost of the test. Starting in the Select A region and moving vertically, performing the analysis on A becomes more attractive as the uncertainty in the strength of material A increases. This is expected because the magnitude of the reduction in uncertainty increases as the standard deviation of A increases, and a greater reduction in uncertainty is more valuable. Next, starting in the Select A region and instead moving left by decreasing the mean strength, analyzing material B becomes preferred and is later outweighed by selecting material B. It makes sense that selecting material B is preferred for low mean strengths of material A, because the likelihood of getting a better value for the strength of material A after analysis is slim. With moderately low mean strengths of material A, it makes sense to do some analysis to determine which material has the higher strength. Because material B has the more uncertain strength in this region (below 60 MPa), analyzing material B will result in a greater reduction of uncertainty than analyzing material A.

When the standard deviation of material A is larger than the standard deviation of material B, it is not immediately obvious why analyzing material B would be preferred over analyzing material A, as is the case for the pressure vessel decision problem indicated by the star in Figure 5.5. Analyzing A only adds value if it results in a change in the choice of material. When the prior belief for the mean strength of material A is lower than that of B, most of the time, the analysis will simply confirm that A is the worst material, so that there is little to be gained from analyzing A. (Notice also from Figure 5.14a that when the strength of the material A does turn out to be larger than B's, the increase in expected utility is relatively small.) Analyzing material B, however, does make sense if it can be followed by an analysis of A, as can be seen in Figure 5.19b. If the analysis of B reveals a worse than expected strength for material B, then it becomes worthwhile to analyze A to verify whether it now has gained the upper hand. Because of the fairly large standard deviation of material A, it is thus possible that analyzing both B and A results in a situation where it is beneficial to switch choices from material B (with larger mean strength and smaller uncertainty) to material A (with smaller mean strength and larger uncertainty). Whether or not the expected payoff of selecting B after analysis on B is larger than the initial expected payoff of selecting B depends on the quality and cost of the analysis. These relationships are explored in the next section.

5.4.4 Exploration of the Effects of the Analysis Cost and Quality

To explore the effect of analysis cost and quality on the boundary plots, additional boundary plots are generated for increasing cost and for decreasing quality. It is expected that the analysis will no longer be preferred for any combination of prior distributions if the analysis is either too costly or has a low enough quality. The progression of increasing cost boundary plots is displayed in Figure 5.20; the analysis standard deviation is held constant at 33 MPa through the progression. The progression of decreasing quality boundary plots is displayed in Figure 5.21. In this progression, the cost is held constant at \$6.60. Note that quality of the analysis decreases as the standard deviation of epsilon increases. In both of the figures, a dotted line indicates the intersection of the two *Analyze* alternatives.

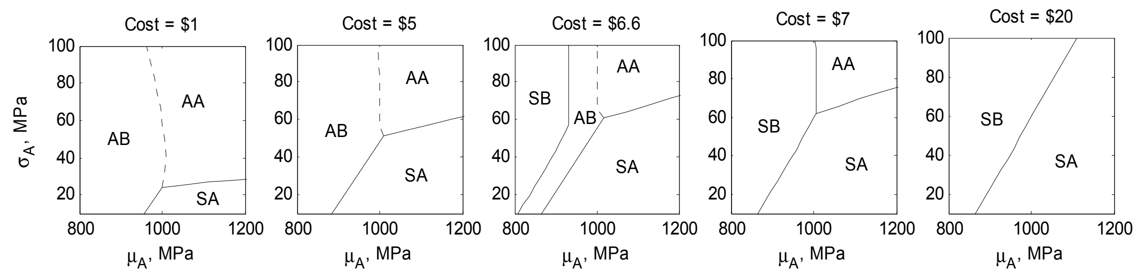


FIGURE 5.20. BOUNDARY PLOTS FOR INCREASING ANALYSIS COST (SA=SELECT A; SB=SELECT B; AA=ANALYZE A; AB=ANALYZE B).

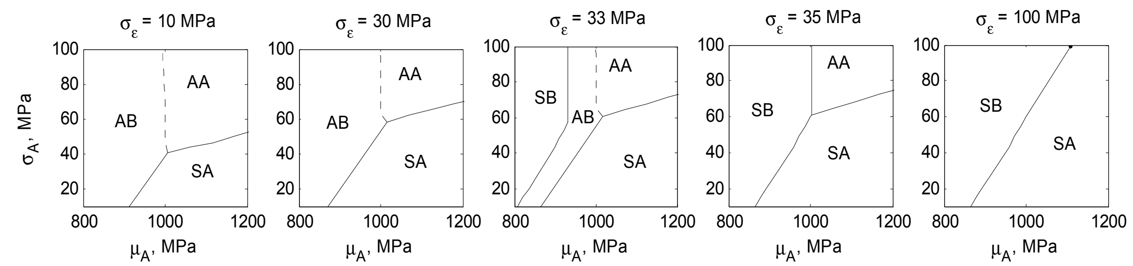


FIGURE 5.21. BOUNDARY PLOTS FOR DECREASING QUALITY OF THE ANALYSIS (SA=SELECT A; SB=SELECT B; AA=ANALYZE A; AB=ANALYZE B).

In both figures, a similar progression is seen. At the left, for low cost and high quality, the boundary plot shows only three regions for Analyze B (AB), Analyze A (AA), and Select A (SA). So for low cost or high quality, analysis of one of the materials is almost always preferred. The exception is when material A has a higher mean and lower standard deviation than material B; in this case it is preferred to select material A. As the cost increases or the quality decreases, the analysis loses value and the boundary plots show four regions: Select B (SB), Analyze B, Analyze A, and Select A. The size of the region in which analyze A is preferred is not substantially impacted, but Select B begins to overtake Analyze B for lower mean values of the strength of material A. At some point Select B overtakes Analyze B entirely, as shown in the fourth plot in each figure. Finally, as the cost continues to increase or the quality continues to decrease, the region in which Analyze A is preferred disappears as well and the only two preferred decision alternatives are Select A and B. These results confirm that both the quality and cost of an analysis impact the desirability of the analysis.

The quality of the analysis impacts the payoff of the pressure vessel via the posterior distribution of material strength. From equation (5.32), the standard deviation of the posterior distribution is always less than the minimum among the prior distribution and analysis standard deviations. Thus, when the analysis standard deviation is smaller than the prior standard deviation, the analysis can provide a significant reduction in uncertainty. However, when the standard deviation of the analysis is larger than that of the prior distribution, the reduction in uncertainty is very small. Also, recall that the uncertainty impacts the expected payoff via the failure cost, but the vessel thickness is selected to optimize the expected payoff. Thus, reducing the uncertainty does not necessarily reduce the probability of vessel failure. Instead, reducing the uncertainty may simply allow for a thinner pressure vessel that reduces material costs.

The cost of the analysis is a simple reduction in the expected payoff of the *Analyze* alternatives. The *Analyze* alternatives must have larger expected utilities than the *Select* alternatives to be preferred, despite this cost penalty. Higher expected utilities are possible with the *Analyze* alternatives because poor outcomes for one material can be avoided in favor of better outcomes in the other material in the subsequent decision between materials A and B.

5.5 DISCUSSION

In this chapter the decision analysis of design process decision problems has been presented to demonstrate the quantitative comparison abilities enabled by Rational Design Theory. Further exploration of the parameters of this scenario revealed that analyses are only desirable if the cost is low and the quality is high. In these cases, it is usually preferred to analyze the parameter that is more uncertain. However, there are some cases in which it is preferred to analyze the parameter that is less uncertain when the less uncertain parameter has a higher mean value. In these cases, the analysis of the less uncertain parameter is usually followed by the analysis of the more uncertain parameter.

The Value of Information decision rule has been evaluated by comparing a full decision tree that allows sequences of decisions to a truncated decision tree which is equivalent to traditional value of information calculations. This comparison shows that the full decision tree analysis is more comprehensive in that it reveals scenarios in which the ability to perform a subsequent analysis adds value; however, the expected utilities of the most preferred alternative are usually very similar regardless of whether the expected utility is computed with the full decision tree or the truncated decision tree. Thus, while the full decision tree analysis provides a richer understanding of the problem when a sequence of information sources is available, Value of Information as a decision rule gets very similar results with less computational expense. This difference in computational expense must be considered when comparing the Value of Information strategy and the full decision tree as prescriptive design methods. Since the Value of Information strategy gets very good results at a much lower computational cost, it is likely to be preferred over the full decision tree analysis.

The analysis presented in these examples is a simple scenario: two discrete concepts and one to two analyses. As a result, the conclusions that can be made in a general sense are limited, but this work forms the foundation for future work. The decision tree analysis in this chapter is limited to the consideration of only one to two analyses for simplicity of the decision tree but also to make the decision tree finite in depth. Because the decision tree analysis is essentially a process of "rolling back" the tree, one must begin at the ends and work back to the beginning. Therefore, the decision tree must be finite so that there is an end at which to begin the rolling back process. But in this work it has been shown that the consideration of subsequent analyses does impact the expected utility of the *Analyze* alternatives. The investigation of additional possible actions will require larger decision trees, which in turn dramatically increases the computational complexity of the decision analysis.

In addition to limiting the depth of the decision tree, the computational complexity of the decision analysis was made manageable in this example by the additional simplifying assumptions of normally distributed uncertain parameters, which simplify the computation of posterior probabilities. The conclusions from these examples are limited by to the assumed relationship between the prior and posterior distributions. The design process decision analysis relies upon the availability of joint probability information through which Bayesian updating can be performed. To strengthen these conclusions, additional joint probability relationships should be studied in future work.

5.6 THESIS ROADMAP

In this chapter two examples were presented to demonstrate how Rational Design Theory enables the quantitative comparison of design methods and the evaluation of decision rules through the decision analysis of design process decision. This chapter serves to validate the secondary hypothesis H2. In the next chapter, the thesis is concluded with a critical review and summary.

CHAPTER 6: CONCLUSION

The thesis is concluded in this chapter with a summary and critical review. The purpose of this work is to establish Rational Design Theory as a new theoretical framework for the evaluation of design methods. A comprehensive review of the thesis up to this point is presented in Section 6.1. In Section 6.2 the research questions and hypotheses are revisited and critically reviewed with emphasis on the validity of the research hypothesis beyond the example problems presented in Chapters 4 and 5. Based on this review, the contributions of this work are discussed in Section 6.3 followed by suggested future work in Section 6.4. The chapter concludes in Section 6.5 with some closing remarks.

6.1 A SUMMARY OF THIS THESIS

Design theories are powerful tools by which design researchers can reason, hypothesize, and learn about good design practice. To provide a sound basis for such reasoning, design theories must embrace the drivers of the design process: uncertainty and economic tradeoffs between the design artifact and the design process. Existing theories of design are reviewed in Section 2.1, and it is demonstrated that they fall short. These theories focus on classification and representation of design knowledge, but they do not provide any justification for the manner in which this knowledge should be applied to achieve good results. Furthermore, most of the existing theories compromise the ability to explicitly express uncertainty in favor of a rigorous mathematical foundation that can be used to infer properties of design. Given the fact that uncertainty is significant in design and that the reduction of uncertainty is a primary design activity, such a compromise is unacceptable.

Rational Design Theory is proposed in Chapter 3 as an improvement on existing theories because it enables the explicit characterization of uncertainty throughout the design process as well as the consideration of tradeoffs between the design process and the designed artifact. In addition, RDT provides a theoretical framework for the comparison and evaluation of design methods as measured by the ideal of maximizing expected Net Design Utility. These capabilities are provided in RDT by applying the normative approach of decision analysis to design process decisions.

RDT is complemented by a descriptive conceptual model of design processes in which the artifact decision and the process decisions are viewed as two levels of design decision making. The artifact decision is described in the Artifact Information State in three aspects: *concepts, concept predictions,* and *concept decision criteria*. These three aspects correspond to the alternatives, outcomes, and preferences of the artifact decision, respectively. In the process decision, each design action changes the concept information state while consuming design phase resources. Design actions include *synthesis*, *analysis*, and *evaluation*. Designers have preferences with respect to the consumption of design phase resources, and they also have preferences concerning the artifact decision. By considering all of these aspects of the process decision in a structured decision analysis, the quality of a particular design action, design process, or design method can be evaluated in a qualitative and quantitative fashion.

The qualitative evaluation capabilities of RDT are demonstrated in Chapter 4 in a review of the systematic design method of Pahl and Beitz (Pahl, et al. 1996). In this review, each of the activities in the Pahl and Beitz method are described in the context of the RDT conceptual model. The method is then evaluated with respect to the characteristics of good design methods as identified in Chapter 3. Because of the general nature of the Pahl and Beitz method, the prescriptions of the method are vague and therefore difficult to conclusively evaluate. Nevertheless, the review demonstrates that the Pahl and Beitz method contains no structural limitations that necessarily lead to irrational behavior. A more conclusive evaluation can only be conducted for a particular implementation of the method for given beliefs and preferences of a particular designer.

The quantitative comparison and evaluation capabilities of RDT are demonstrated in Chapter 5 with two example problems. In both examples, the design strategy of applying the analysis that maximizes the Value of Information is evaluated. The evaluation is achieved by comparing the Value of Information calculation to a more comprehensive decision tree analysis that allows for a sequence of information sources to be considered. In both examples it is found that the more comprehensive decision tree analysis finds cases in which it is beneficial to conduct an analysis while the Value of Information calculation indicates that it is not beneficial to conduct an analysis. Although the Value of Information strategy predicts the wrong preferred design action in some cases, the expected utilities of the preferred decision alternatives as calculated by both methods are very similar. Thus, the Value of Information strategy is found to be an appropriate approximation as long as a small amount of error in expected utility is tolerated. Furthermore, the cost of the decision tree analysis was assumed to be negligible in both the Value of Information strategy and the more comprehensive decision analysis. If the computational expense of these decision tree calculations were included, Value of Information would surely be preferred to the more comprehensive analysis because it takes less time to compute the answer with the Value of Information strategy.

The examples in Chapter 5 demonstrate that RDT provides a structured framework for comparing and evaluating design methods in a quantitative fashion; however, the examples themselves are simplified problems. Even to evaluate such a simplified problem, significant computational resources are required to solve moderately sized decision trees. As a result, it is difficult to quantitatively support *generalized* conclusions with such an analysis. Although improvements can be made in the numerical calculations to speed computation, quantitative analysis will always be limited to very specific problems because beliefs and preferences are difficult to meaningfully generalize.

6.2 REVISITING THE RESEARCH QUESTIONS AND HYPOTHESES

As introduced in Chapter 1, the primary research question addressed in this thesis is as follows:

What theoretical foundation for design has explanatory power and enables quantitative evaluation of design methods?

The hypothesis is that Rational Design Theory provides explanatory power by enabling qualitative and quantitative evaluation of design methods. This is achieved through the application of normative decision theory to design *process* decisions. In addition, the application of normative decision theory is supported by the conceptual model of the design process. This hypothesis is defended by addressing the qualitative and quantitative evaluation of design methods separately in the secondary research questions. Recall that the goal in proposing RDT is to establish a normative theoretical framework for evaluating prescriptive design methods. RDT is not intended as decision-making tool for use during design, but rather as a normative ideal to which prescriptive design methods can be compared.

The first secondary research question refers to the qualitative evaluation of design methods enabled by Rational Design Theory: "*Can Rational Design Theory be used to compare design methods in a qualitative sense?*", This question is answered in the affirmative in Section 3.5.2. The central argument is that Rational Design Theory enables qualitative evaluation of design methods through the decomposition of the design process decision into alternatives, outcomes, and preferences. These elements of the *process* decision reference the Artifact Information State, which is a representation of the *artifact* decision problem. The decomposition of this two-level decision problem enables an element-by-element comparison of design methods, since a design method prescribes a sequence of decisions.

The second secondary research question pertains to the quantitative comparison of design methods: *Does Rational Design Theory enable quantitative prediction of design method quality?*". This question is also answered in the affirmative in Section 3.5.2, and this ability of RDT is demonstrated in two example problems in Chapter 5. The analysis of design process decisions provides a mathematical framework for computing the expected utility of design actions. However, to do so, one must formalize the outcomes and preferences of design actions mathematically. This is only possible for a particular design's beliefs and preferences concerning a very specific design scenario. As such, this requirement can limit the generality of conclusions that can be drawn from such a quantitative analysis. Nevertheless, analysis of design process decisions is a powerful tool that can be used to provide quantitative explanation of the reasons for the success of popular design methods.

6.3 CONTRIBUTIONS

The primary contribution of this thesis is Rational Design Theory: a new theoretical framework that incorporates a conceptual model of design processes and the normative approach of decision theory applied to design process decisions. This theory improves on previous work by explicitly including uncertainty considerations and the cost of the design process without placing limits on beliefs or preferences of the designer. Furthermore, RDT enables quantitative comparison of design methods with respect to a particular set of beliefs and preferences. Such comparisons can lead to insights that support the development of improved design methods. Previous design theories did not provide this type of guidance about the quality of design methods.

In the development of RDT, several insights about design were gained. The primary insight is that design is a decision-making process about design actions; i.e., design is about process decision rather than artifact decisions. Viewing design in this context enables the consideration of tradeoffs between the design process and the quality of the designed artifact. I believe that these tradeoffs are crucial for understanding the reasons behind popular and successful design methods.

Highlighting the tradeoff between the design process and the quality of the design artifact, RDT shows that design is about maximizing the expected NDU; i.e., the utility of the design process, taking into account both the utility of the designed artifact and the cost of the design process. This maximum expected NDU may or may not be realized by producing an artifact, since the production of an artifact requires resources to be spent which may not be recovered through sales of the artifact. Additionally, it may be required to radically reformulate the artifact decision in order to maximize NDU.

Building on this point, another important insight relates to the end of the design phase. RDT demonstrates that design should end when all remaining design actions would reduce rather than increase the expected NDU. Thus, design can end with the production of an artifact, but it is also possible that design should end with the cancellation of the project in order to cut losses.

In Chapter 5, RDT was applied in two example problems to demonstrate the quantitative comparison of design methods under a consistent set of designer beliefs and preferences. In these examples, it was shown that one design method can be quantitatively determined to be preferred over another for particular beliefs and preferences. This type of comparison is useful for studying design processes after completion or in a hypothetical sense, but the analysis is too computationally demanding to be used to support real-time design-process decision making. Although the Value of Information heuristic sometimes predicts a different preferred design action than the more comprehensive decision analysis, the difference in expected NDU between the preferred action as predicted by the decision analysis and the preferred action as predicted by the Value of Information heuristic is very small. Thus, comparing the Value of Information approach to the more comprehensive decision analysis showed that the Value of Information approach is an appropriate approximation of normative decision theory as long as a small amount of error can be tolerated. This comparison demonstrates the application of RDT as a normative framework for quantitative evaluating prescriptive design methods.

6.4 OPPORTUNITIES FOR FUTURE WORK

RDT provides a theoretical foundation for the evaluation of design methods; therefore, there are many opportunities for future work enabled by RDT. These opportunities are summarized in Table 6.1 and are discussed in more detail thereafter. In addition, there are opportunities to enhance RDT with extensions and supporting studies. These opportunities are identified in Table 6.2. There are also some opportunities for future work inspired by the quantitative examples in Chapter 5. These opportunities for future work are given in Table 6.3.

TABLE 6.1. OPPORTUNITIES FOR FUTURE WORK ENABLED BY RDT

1		**Explore the benefits in a design process of explicitly modeling the uncertainty in artifact outcomes.**
	RQ	When is it a good idea to model uncertainty explicitly?
	Tasks	• Discuss the sources and impacts of uncertainty in artifact outcomes. • Review methods for design under uncertainty • Identify a quantitative example for comparing a design method in which uncertainty is explicitly modeled with a design method in which uncertainty is ignored.
2		**Explore the tradeoffs between various design phase resources: time, cost, manpower, etc.**
	RQ	Why is it sometimes a good idea to analyze multiple concepts in parallel?
	Tasks	• Identify the resources allocated in design process decisions. • Identify conflicts between these resources. • Develop and solve a quantitative example exploring one or more conflicts between design phase resources.

A first suggested study enabled by RDT is to explore the benefits of explicitly modeling uncertainty during a design process. A primary motivation for the development of RDT was that existing theories of design do not allow for the explicit characterization of uncertainty. This is important because uncertainty is a primary driver in the design process, and the decisions that designers make about design actions depend on their attitudes toward uncertainty. However, the characterization of uncertainty can be a tedious process. While beliefs about uncertain outcomes *can* be elicited and formalized as subjective probabilities, the process of doing so is time-consuming and requires that the analyst be knowledgeable about belief elicitation. Because of this large cost associated with characterizing uncertainty, it may sometimes be preferred to ignore the uncertainty or crudely approximate the uncertainty in outcomes. In the early stages of design when uncertainty is large, it may make more sense to make decisions on the basis of the best guess about concept outcomes rather than a more comprehensive expression of all possible outcomes. These hypotheses can be quantitatively tested by comparing design methods in the context of RDT. This should be done by comparing the outcomes of an appropriate sequence of design actions with and without an explicit characterization of uncertainty.

A second study enabled by RDT is the investigation of tradeoffs between various design phase resources such as cost, time, manpower, and computing resources. This study is prompted by the observation that designers often conduct analyses of multiple concepts in parallel. Such parallel investigations obviously take less time to complete, but require more resources in other areas such as cost, manpower, and computing resources. One may then ask why it is preferred to run parallel analyses when it is obvious that it will cost more to do so. When deadlines are looming, it is apparent that time may be more important than other resources, but it is possible that parallel investigation is valuable even in the absence of strict deadlines because of missed market opportunities. To investigate these interactions, the design phase resources and the conflicts between them must be identified. Then, a quantitative example can be developed based on RDT to test hypotheses regarding deadlines, market timing, or other potential causes.

In addition to future work enabled by RDT, there are also opportunities for future work to extend and enhance RDT. These are summarized in Table 6.2. The first opportunity for future work would serve to strengthen RDT by providing more examples of design methods explained in the context of RDT. This study leverages the structured framework of RDT to compare and contrast some common design strategies. This type of qualitative comparison is one of the primary capabilities of RDT, as mentioned previously. By explaining well-known design strategies in the context of RDT, this work would help to explore and strengthen the core ideas of RDT. In addition, this study could identify important similarities and differences between existing design methods. In current work, existing methods for design under uncertainty have been compared on the basis of risk attitudes (Lee, et al. 2010b). Although this study is not carried out in the context of RDT, similar types of insights would likely be gained with RDT as a central framework.

Another opportunity for future work to support RDT is a descriptive study of practical design problems. In such a study, researchers would observe and record the beliefs, preferences, and design rationale of designers throughout a practical design project. Then, by interpreting the data in the context of RDT, conclusions can be drawn about how real design processes proceed. Such a study would help to verify that the RDT conceptual model is sufficiently comprehensive such that it is capable of characterizing any design method, regardless of whether it is considered to be a good method.

One area of RDT that could use improvement is the modeling of outcomes of synthesis actions. All synthesis methods differ in the amount of effort required and the quality of concepts produced. It is our view that the various synthesis methods can be quantitatively evaluated in the context of Rational Design Theory. To do this, some means for quantifying the resources spent during synthesis is needed, as well as a way to define the goodness of concepts produced. By gathering data on synthesis actions in practical design processes, researchers can make better estimates of the resources and outcomes of synthesis actions in future investigations.

The next suggested opportunity for future work could support such data gathering in practical design problems. In this third suggested study, the Systems Modeling Language (OMG SysML^{TM})(SysML 2006) is leveraged to provide a formal information model for the artifact information state and elements of the design process decision. The RDT conceptual model can be used to develop an information model for tracking knowledge about concepts and available actions during design. Formalizing such a model in a language such as SysML would facilitate the documentation of design processes and ease information sharing among distributed teams and multiple stakeholders. In addition, this work would provide a theoretical foundation for the use of SysML in systems design processes. Although there are constructs capable of representing uncertainty in SysML, they are not consistently or widely used in current work. Since uncertainty is a driving factor in the development of RDT, adopting a variant of the information model of RDT in SysML would provide some structure to the application of SysML in systems engineering for information capture and process documentation. Furthermore, such a model would support the adoption of RDT as a framework for the documentation of design processes, which would further support the collection of practical design data for future RDT-inspired work.

TABLE 6.2. OPPORTUNITIES FOR FUTURE WORK TO EXTEND AND ENHANCE RDT

1	**Comparison of common design strategies using RDT.**	
	RQ	What are the similarities and differences in common design strategies?
	Tasks	• Identify design strategies for investigation (e.g., robust design, reliability-based design, set-based design, product platform design, etc.) • Describe each strategy in the context of RDT. • Compare the strategies on the basis of the elements of the design process decision and the concept information state. • Quantitatively compare the strategies for a specific design scenario.
2	**Descriptive study of practical design processes.**	
	RQ	How do designers make decisions about the design process in practical design problems?
	Tasks	• Identify a design project in which designers are willing to participate in a design study. • Determine the scope of data gathering: design phase, type of data to gather, how to gather, how to capture, etc. • Gather data. • Interpret data in the context of RDT • Draw conclusions about the design process as recorded.
3	**Formalize concept information state and elements of process decision in SysML**	
	RQ	How can design information be captured in SysML constructs in the context of RDT?
	Tasks	• Enumerate and describe the types of design information in RDT. • Identify SysML constructs that can be used to represent these types of design information. • Demonstrate the documentation and manipulation of design information using these SysML constructs.
4	**Explore impacts of design actions that extend beyond a particular design project.**	
	RQ	How can one account for expected but uncertain benefits of design actions that extend beyond a particular design project, such as the benefits of formulating reusable performance models?
	Tasks	• Identify cases in which the outcomes of a design action may extend beyond a particular design project. • Explore these outcomes with respect to fundamental outcomes of decision-makers. • Propose a means for including these extended benefits in the context of RDT.

Another idea for future work to strengthen RDT is to investigate the impacts of design actions which may have benefits beyond the current design process. The generation of reusable performance models, for example, may have a moderate benefit during the current design project. Such a model may have an enormous benefit in later design projects since the hard work of creating the model has already been completed. It is unclear how to model preferences for such a model within RDT. This is complicated by the fact that the future benefit of the reusable model is uncertain. Although one may anticipate that the model will be reused, it cannot be known with certainty whether or not the model will actually be used or how many times it will be reused. Future work in this area would strengthen the ability of RDT to explain current practice.

In addition to opportunities for future work that are enabled by or enhance RDT itself, there is an opportunity for future work based on the conclusions of the examples from Chapter 5. This opportunity is summarized in Table 6.3.

In the example problems it was found that the Value of Information strategy is a good strategy for determining when to perform an analysis to gather more information in support of a decision between two design concepts. In future work, this conclusion can be used to develop a strategy for the efficient optimization of a large design space using a surrogate model when additional samples can be taken. Finding the global optimum in a large design space is a compromise between broad coverage and detailed exploration. Detailed exploration is enabled by complex simulation models; however, these models are often slow to evaluate. Generating a surrogate model using an interpolation technique can enable broad coverage of this design space with a small amount of error. The creation of the interpolating model requires evaluation of the underlying simulation model.

Evaluating more points using the simulation model reduces the error of the surrogate model, but it may take a long time to evaluate each point. Thus, to decide how many points to sample, one must make a compromise between evaluation time and accuracy. This compromise is similar to the tradeoff investigated in the example problems in this thesis; thus, the conclusions from these example problems can be leveraged to develop heuristics for efficient search of a large design space with surrogate modeling.

TABLE 6.3. OPPORTUNITIES FOR FUTURE WORK INSPIRED BY THE EXAMPLES

	Leverage VoI conclusions to develop a technique for efficient global optimization using a kriging model
RQ	Given that Value of Information is an appropriate strategy for determining when to gather additional information, how can this strategy be used to support efficient global optimization using a surrogate model?
Tasks	• Identify a practical optimization example problem for demonstration and validation. • Develop a decision rule based on the Value of Information of an additional sample for the surrogate model. • Implement this decision rule within an optimization algorithm. • Demonstrate and critically evaluate this technique with respect to other strategies for efficient global optimization.

6.5 CLOSING REMARKS

RDT provides a theoretical framework for the comparison and evaluation of design processes and prescriptive design methods. This comparison is enabled by applying normative decision analysis to design *process* decisions, which is a departure from previous work focusing on design artifact decisions. This process context is a more appropriate context for the application of decision theory to design because it allows for the considerations of tradeoffs between the design process and the performance of the design artifact. I believe that the investigation of these tradeoffs will provide explanation of why particular design methods are successful in particular design and economic situations.

I believe that RDT has the potential to foster a new direction in design research. Presenting new methods in the context of RDT makes it easier to highlight the advantages of new methods over old methods and the differences in the classes of problems for which new methods are intended. In addition, knowledge of the types of information and the transformations of information in design will help designers to hypothesize and test new design methods and improvements to design practice.

APPENDIX A: MANAGER'S DECISION MATLAB CODE

In this appendix, the Matlab code for solving the Manager's Decision problem is presented. A list of the files with associated page numbers is presented in Table A.1.

TABLE A.1. INVENTORY OF M-FILES FOR THE MANAGER'S DECISION PROBLEM

Filename	Page
One Analysis Case	
AnalysisCost.m	172
areaCrossing.m	172
normedTwoOnewRiskAversionFunction.m	217
normedTwoOnewRiskAversionFunction_OneAnalysis.m	217
plotBoundaries_new.m	218
plotBoundaries_OneAnalysis_new.m	220
scanYLdown_new.m	224
scanYLdown_OA_new.m	224
scanYLup_new.m	224
scanYLup_OA_new.m	224
scanYMid_new.m	224
scanYMid_OA_new.m	225
scanYRdown_new.m	225
scanYRdown_OA_new.m	225
scanYRup_new.m	225
scanYRup_OA_new.m	225
TwoOnewRiskAversionFunction.m	227
TwoOnewRiskAversionFunction_OneAnalysis.m	229
utilFunction.m	233
Two Analysis Case	
decisionProblem2_2Function.m	173
E_U_DNminus2.m	174
E_U_Test_X.m	178
evaluateDecision.m	180
findIntegrationLimitsNR.m	184
RunDoEs_TwoTwo.m	221
Scenario1_TwoAnalyses_fromDoEData.m	225
TwoTwowRiskAversionFunction_OneAnalysis.m	229

AnalysisCost.m

```
function [cost] = AnalysisCost(infostate, costsOfAnalyses)

% Inputs:  - infostate:  a 1 x 4 vector indicating which analyses have been
%            performed, i.e., [0 1 0 0]
%          - costsOfAnalyses: a 1 x 4 vector indicating the costs of
%            analyses; [LQAcost, LQBcost, HQAcost, HQBcost]

% Outputs: - cost: the sum of the costs of the analyses that have been
%            performed

cost = infostate*costsOfAnalyses';
```

areaCrossing.m

```
% testing using fzero to find points where number of areas changes

Rvect = [0 1 2];
options = optimset('TolX',1e-4);
stdev_eps_vect = [0 0.125, 0.25, 0.5, 0.75, 1, 1.5, 2, 3, 4, 5];

for jj = 1:length(Rvect)

    C_cross = zeros(length(stdev_eps_vect));

    R = Rvect(jj);

    % stdev_eps_vect = [0.125, 0.25];
    tic
    C_cross(1) = fzero(@(c) plotBoundaries(R,stdev_eps_vect(1),c),
0.4+0.1*(jj-1) , options);
    toc

    for ii = 2:(length(stdev_eps_vect))
        tic
        C_cross(ii) = fzero(@(c) plotBoundaries(R,stdev_eps_vect(ii),c),
C_cross(ii-1), options);
        toc
    end

    plot(C_cross,stdev_eps_vect)
    xlabel('Cost')
    ylabel('\sigma_\epsilon')
    title({'Analysis Quality vs Cost Screening Test';...
            ['R = ',num2str(R)]})
    hgsave(['Screen-R',num2str(R),'-',datestr(now,'HH.MM.SS'),'.fig'])
    close
    csvwrite(['C_cross-',num2str(R),'.csv'],C_cross)

end

%%
% Plot all three on one
data = csvread('ScreeningTestPoints.csv',1,0);
R0 = data(:,1);
```

```
R1 = data(:,2);
R2 = data(:,3);
stdev_eps_vect = [0 0.125, 0.25, 0.5, 0.75, 1, 1.5, 2, 3, 4, 5];

plot(R0,stdev_eps_vect,'k',R1,stdev_eps_vect,'k--',R2,stdev_eps_vect,'k-.')
legend('R = 0','R = 1','R = 2')
xlabel('Cost')
ylabel('Quality')
axis([0 0.6 0 5])
```

decisionProblem2_2Function.m

```
function [SA,SB,A1A,A1B,A2A,A2B] =
decisionProblem2_2Function(mu_uA,stdev_uA,mu_uB,stdev_uB,stdev_eps,stdev_de
lta,C1,C2,R,tol)

% tic
%% set up the problem

global debug;
debug = 0;

% % generate zero mean, unit variance normally distributed stratified
sample
% global normalizedSamples;
% numSamples = 10;
% normalizedSamples = icdf('normal',(0.5:numSamples)/numSamples,0,1);

% define the analyses
global analyses;
analyses.alternative = [1, 2, 1, 2];                    % analternative to
which applied
% analyses.stdev       = [0.25, 0.25, 0.0625, 0.0625];
% analyses.cost        = [0.0313, 0.0313, 0.3125, 0.3125];
analyses.stdev       = [stdev_eps, stdev_eps, stdev_delta, stdev_delta];
analyses.cost        = [C1, C1, C2, C2];

% define the prior mean and variance for each alternative
% infoState.mean  = [1, 0];
% infoState.stdev = [1.25, 1];
infoState.mean  = [mu_uA, mu_uB];
infoState.stdev = [stdev_uA, stdev_uB];
infoState.done  = zeros(length(analyses.alternative),1);

% define the risk aversion
% R = 2;

% define the tolerance for the numerical integration
% tol = 1e-4;

%% solve the decision problem
% 6 branches in top-level decision
[expUtil1] = evaluateDecision(1,infoState,0,R,tol);
[expUtil2] = evaluateDecision(2,infoState,0,R,tol);
[expUtil3] = evaluateDecision(3,infoState,0,R,tol);
[expUtil4] = evaluateDecision(4,infoState,0,R,tol);
```

```
expUtil_Select = utilFunction(infoState.mean, infoState.stdev, R);

% fprintf('|--------------|\n')
% fprintf('Select A %12.10f\n',expUtil_Select(1));
% fprintf('Select B %12.10f\n',expUtil_Select(2));
% fprintf('LQ A     %12.10f\n',expUtil1);
% fprintf('LQ B     %12.10f\n',expUtil2);
% fprintf('HQ A     %12.10f\n',expUtil3);
% fprintf('HQ B     %12.10f\n',expUtil4);
% fprintf('|--------------|\n');

SA = expUtil_Select(1);
SB = expUtil_Select(2);
A1A = expUtil1;
A1B = expUtil2;
A2A = expUtil3;
A2B = expUtil4;

% Plot
% A = normalizedSamples * infoState.stdev(1) + infoState.mean(1);
% B = normalizedSamples * infoState.stdev(2) + infoState.mean(2);
% samples = (0.5:numSamples)/numSamples;
% plot(A,samples,B,samples,...
%     result1,samples,result2,samples,...
%     result3,samples,result4,samples)
% legend('A','B','LQ A','LQ B','HQ A','HQ B','location','SouthEast')
% xlabel('Utility')
% toc
```

E_U_DNminus2.m

```
function [expectedUtility] = E_U_DNminus2( analysis, infoState, accumCost,
R, tol)

global analyses

% E_U_DNminus2() is a function that computes the expected utility of the
% second to last decision node by breaking the expectation integral into
% three parts.  In the first part, which corresponds to poor test values,
% the preferred alternative is to select the concept that wasn't tested. In
% the second part, which corresponds to intermediate test values, the
% preferred alternative is to perform the final test and then decide
% between the two concepts.  In the third part, which corresponds to good
% test values, the preferred alternative is to select the concept which was
% tested.  Depending on the prior distributions, any of these portions may
% be absent from the integral; e.g., if the prior distribution on the
% concept being tested is already much better than the prior distribution
% for the other concept, it may be the case that one will never select the
% other concept even if a low test value is returned.

% First find the thresholds between the three portions of the integral
% [Tstar, Tlower, Tupper] = findIntegrationLimits( analysis, infoState,
% accumCost, R, tol)
Tstar = [];
Tlower = [];
```

```
Tupper = [];

% fprintf(1,'FINDING ROOTS -------------------------------------------
\n',[])
%     fprintf(1,'current Analysis = %1.1f\n',analysis)
%     fprintf(1,'infoState.mean = %12.8f  %12.8f\n',infoState.mean)
%     fprintf(1,'infoState.stdev = %12.8f  %12.8f\n',infoState.stdev)
%     fprintf(1,'infoState.done = %12.8f %12.8f %12.8f
%12.8f\n',infoState.done)
%     fprintf(1,'accumCost = %4.3f\n',accumCost)
[Tstar, Tlower,
Tupper,xup,rel_error_up,xdown,rel_error_down,priorStdevZeroflag] =
findIntegrationLimitsNR( analysis, infoState, accumCost, R, tol);
% fprintf(1,'Tlower = %12.8f\n',Tlower)
% fprintf(1,'Tupper = %12.8f\n',Tupper)
% fprintf(1,'DONE FINDING ROOTS-------------------------------------
\n',[])


% Once the thresholds have been found, compute the integrals.  Analytical
% expressions are used for the first and third portions of the integral,
% while the intermediate portion is computed via numerical integration over
% E_U_Test_X.

% If Tstar is empty, then all three integrals must be computed; else, only
% the lower and upper integrals need to be computed.

LowerInt = [];
UpperInt = [];
InterInt = [];

% Extract variables from analysis and infostate
alternative = analyses.alternative(analysis);
analysisStdev = analyses.stdev(analysis);
priorStdev = infoState.stdev(alternative);
priorMean = infoState.mean(alternative);
stdevSumSq = priorStdev^2 + analysisStdev^2;

% perform the analysis
newInfoState = infoState;
newInfoState.stdev(alternative) = priorStdev * analysisStdev /
sqrt(stdevSumSq);
newInfoState.done(analysis) = 1;
newAccumCost = accumCost + analyses.cost(analysis);

% determine which analyses to perform next
nextAnalyses = find(not(newInfoState.done));
nextAnalysis = nextAnalyses(1);
nextAccumCost = newAccumCost + analyses.cost(nextAnalysis);
nextAlternative = analyses.alternative(nextAnalysis);
nextAnalysisStdev = analyses.stdev(nextAnalysis);

% Extract mu_uA, mu_uB, stdev_uA, and stdev_uB
stdev_uA = newInfoState.stdev(1);
stdev_uB = newInfoState.stdev(2);

% Define variables
```

```
if alternative==1  %A is being tested
    mu_uX=priorMean;
    mu_uY=newInfoState.mean(2);
    stdev_uX=stdev_uA;
    stdev_uY=stdev_uB;
    mu_uA_givenX = @(X) (priorMean*analysisStdev^2 +
X*priorStdev^2)/(stdevSumSq);
    mu_uB_givenX = @(X) newInfoState.mean(2);
elseif alternative==2  %B is being tested
    mu_uX=priorMean;
    mu_uY=newInfoState.mean(1);
    stdev_uX=stdev_uB;
    stdev_uY=stdev_uA;
    mu_uA_givenX = @(X) newInfoState.mean(1);
    mu_uB_givenX = @(X) (priorMean*analysisStdev^2 +
X*priorStdev^2)/(stdevSumSq);
else
    error('TestX must be either 1 for A or 2 for B.')
end

if priorStdevZeroflag == 1
    if R == 0
        EselectUp = (mu_uX-newAccumCost);
        EselectDown = (mu_uY-newAccumCost);
    else % R > 0
        EselectUp = 1/R*(1-exp(-R*(mu_uX -
newAccumCost)+stdev_uX^2*R^2/2));
        EselectDown = 1/R*(1-exp(-R*(mu_uY -
newAccumCost)+stdev_uY^2*R^2/2));
    end
    Etest =
E_U_Test_X(mu_uA_givenX(priorMean),mu_uB_givenX(priorMean),stdev_uA,stdev_u
B,...
        nextAccumCost,nextAnalysisStdev,R,nextAlternative);
    expectedUtility = max([EselectUp,EselectDown,Etest]);
else

    if isequal(Tstar,[])
        if R == 0
            LowerInt = (mu_uY-
newAccumCost)*normcdf(Tlower,priorMean,max(sqrt(stdevSumSq),tol));

            UpperInt = (analysisStdev^2*priorMean/stdevSumSq-
newAccumCost)*...
                (1-normcdf(Tupper,priorMean,max(sqrt(stdevSumSq),tol)))+...
                priorStdev^2/(sqrt(2*pi)*stdevSumSq)*...
                (sqrt(stdevSumSq)*exp(-(Tupper-
priorMean)^2/(2*stdevSumSq))...
                +priorMean*sqrt(pi/2)*...
                (1-erf((Tupper-priorMean)/(sqrt(2)*sqrt(stdevSumSq)))));

        else %R ~= 0
            LowerInt = 1/R*(1-exp(-R*(mu_uY -
newAccumCost)+stdev_uY^2*R^2/2))*...
                normcdf(Tlower,priorMean,max(sqrt(stdevSumSq),tol));
            UpperInt = 1/R*(1-
normcdf(Tupper,priorMean,max(sqrt(stdevSumSq),tol)))...
                - 1/R*(exp(stdev_uX^2*R^2/2 ...
                +R*newAccumCost+R^2*priorStdev^4/(2*(stdevSumSq))-
R*priorMean))...
                *(1-normcdf(Tupper,(priorMean-
R*priorStdev^2),max(sqrt(stdevSumSq),tol)));
        end
        if sqrt(stdevSumSq)<tol
            if priorMean <= Tupper && priorMean >= Tlower
                InterInt =
E_U_Test_X(mu_uA_givenX(priorMean),mu_uB_givenX(priorMean),stdev_uA,stdev_u
B,...
                nextAccumCost,nextAnalysisStdev,R,nextAlternative);
            else % priorMean is not between Tlower and Tupper
                InterInt = 0;
            end
        else %sqrt(stdevSumSq) >=tol
            % If the limits of integration are too far away from the
            % "interesting" things happening in the normal probability
            % density function, the adaptive quadrature will not find that
            % very small range of interesting-ness and will return a value
            % of zero for the integral.  Thus, if the limits of integration
            % are more than 14 times the standard deviation of the normal
            % distribution, we will replace them with +/- 7 times the
            % standard deviation of the normal distribution.
            if 14*sqrt(stdevSumSq)>=(Tupper - Tlower)
                % Active portion of pdf is much larger than integration
                % bounds, so don't change the bounds
                lowerbound = Tlower;
                upperbound = Tupper;
            else % The active portion of the pdf smaller than the
                % integration bounds, so the bounds should be changed to
                % only integrate over the active portion
                pdfMax = priorMean+7*sqrt(stdevSumSq);
                pdfMin = priorMean-7*sqrt(stdevSumSq);
                if Tupper <= pdfMax && Tupper >= pdfMin
                    upperbound = Tupper;
                else
                    upperbound = pdfMax;
                end
                if Tlower >= pdfMin && Tlower <= pdfMax
                    lowerbound = Tlower;
                else
                    lowerbound = pdfMin;
                end
            end
% ---------------------------------------------------------------------------
% INTEGRATE OVER MIDDLE BRANCH ------------------------------------------
%             InterInt = quad(@(sampleVal)
(normpdf(sampleVal,priorMean,sqrt(stdevSumSq)).*...
%
E_U_Test_X(mu_uA_givenX(sampleVal),mu_uB_givenX(sampleVal),stdev_uA,stdev_u
B,...
%                 nextAccumCost,nextAnalysisStdev,R,nextAlternative)),...
%                 lowerbound,upperbound,tol)
```

```
            InterInt =
Mid_Int(lowerbound,upperbound,infoState,analyses,analysis,R);

% ---------------------------------------------------------------------------
% ---------------------------------------------------------------------------
        end
    else % Tlower and Tupper should be empty
        if R == 0
            LowerInt = (mu_uY-
newAccumCost)*normcdf(Tstar,priorMean,max(sqrt(stdevSumSq),tol));

            UpperInt = (analysisStdev^2*priorMean/stdevSumSq-
newAccumCost)*...
                (1-normcdf(Tstar,priorMean,max(sqrt(stdevSumSq),tol)))+...
                priorStdev^2/(sqrt(2*pi)*stdevSumSq)*...
                (sqrt(stdevSumSq)*exp(-(Tstar-
priorMean)^2/(2*stdevSumSq))...
                +priorMean*sqrt(pi/2)*...
                (1-erf((Tstar-priorMean)/(sqrt(2)*sqrt(stdevSumSq)))));

        else % R ~= 0
            UpperInt = 1/R*(1-
normcdf(Tstar,priorMean,max(sqrt(stdevSumSq),tol)))...
                - 1/R*(exp(stdev_uX^2*R^2/2 ...
                +R*newAccumCost+R^2*priorStdev^4/(2*(stdevSumSq))-
R*priorMean))...
                *(1-normcdf(Tstar,(priorMean-
R*priorStdev^2),max(sqrt(stdevSumSq),tol)));

            LowerInt = 1/R*(1-exp(-R*(mu_uY -
newAccumCost)+stdev_uY^2*R^2/2))*...
                normcdf(Tstar,priorMean,max(sqrt(stdevSumSq),tol));
        end



    end

    expectedUtility = sum([LowerInt,UpperInt,InterInt]);
end
```

E_U_Test_X.m

```
function
E_test=E_U_Test_X(mu_uA,mu_uB,stdev_uA,stdev_uB,cost,stdev_eps,R,TestX)

%% E_test=E_U_Test(mu_uA,mu_uB,stdev_uA,stdev_uB,cost,stdev_eps,R)
%  E_U_Test_X() is a function that algebraically calculates the expected
%  value of performing a test on design alternative X defined by 'TestX'
If the test were
%  performed, a test value V_TX would be returned, and the prior beliefs
%  about the distribution of the utility of the alternative is updated
%  using Bayesian statistics.  An integral over all possible test values
%  was calculated to generate the expected utility of performing the test.
%
%
```

```
%  **Parameters**
%   Priors (mu_uA,mu_uB,stdev_uA,stdev_uB) of the distribution of the
%      possible values for two design alternatives 'A' and 'B'
%   Information about the cost 'cost' of testing the alternative and
%      information about the quality of the test characterized by
%      its standard deviation 'stdev_eps.'
%   The constant risk aversion 'R' of the decision maker.
%   A definition variable 'TestX' which is 1 if X='A' is the alternative
being
%   tested, and 2 if X='B' is the alternative being tested.
%%  Variable Definition
if TestX==1  %A is being tested
    mu_uX=mu_uA;
    mu_uY=mu_uB;
    stdev_uX=stdev_uA;
    stdev_uY=stdev_uB;
elseif TestX==2  %B is being tested
    mu_uX=mu_uB;
    mu_uY=mu_uA;
    stdev_uX=stdev_uB;
    stdev_uY=stdev_uA;
else
    error('TestX must be either 1 for A or 2 for B.')
end


%% Check for stdev_uX equal to zero
if stdev_uX == 0
    E_test = max(utilFunction(mu_uX-cost, stdev_uX, R),...
        utilFunction(mu_uY-cost, stdev_uY, R));
else

    %% Intermediate calculations

    stdev_Vtx=sqrt(stdev_uX^2+stdev_eps^2);
        %Standard deviation for distribution of uX given the value of the
test
    stdev_XgivenLQ=sqrt(stdev_uX^2*stdev_eps^2/stdev_Vtx^2);
        %Critical test value at which the E(uX)=E(uY) after testing X
    V_TXStar=(1/stdev_uX^2)*(stdev_Vtx^2*mu_uY-
stdev_eps^2*mu_uX+R/2*(stdev_uX^2*stdev_eps^2-stdev_Vtx^2*stdev_uY^2));

    %% Define Expectations for all risk considerations
    if R==0
        E_test=(mu_uY-cost).*normcdf(V_TXStar,mu_uX,stdev_Vtx)+...
            (stdev_eps^2.*mu_uX./stdev_Vtx^2-cost).*(1-
normcdf(V_TXStar,mu_uX,stdev_Vtx))+...
            stdev_uX^2/(sqrt(2*pi)*stdev_Vtx^2)*(stdev_Vtx.*exp(-(V_TXStar-
mu_uX).^2/(2*stdev_Vtx^2))+mu_uX.*sqrt(pi/2).*(1-erf((V_TXStar-
mu_uX)/(sqrt(2)*stdev_Vtx))));
    else
        E_test=1/R*(1-...
            (exp(-R.*(mu_uY-cost)+1/2*stdev_uY^2*R^2))...
            .*normcdf((V_TXStar-mu_uX)/stdev_Vtx)...
            -exp(1/2*stdev_XgivenLQ^2*R^2+R.*(R*stdev_uX^4/(2*stdev_Vtx^2)-
mu_uX+cost))...
            .*(1-normcdf((V_TXStar+R*stdev_uX^2-mu_uX)/stdev_Vtx)));
    end
end
```

evaluateDecision.m

```
function [expectedUtility] = E_U_DNminus2( analysis, infoState, accumCost,
R, tol)

global analyses

% E_U_DNminus2() is a function that computes the expected utility of the
% second to last decision node by breaking the expectation integral into
% three parts.  In the first part, which corresponds to poor test values,
% the preferred alternative is to select the concept that wasn't tested. In
% the second part, which corresponds to intermediate test values, the
% preferred alternative is to perform the final test and then decide
% between the two concepts.  In the third part, which corresponds to good
% test values, the preferred alternative is to select the concept which was
% tested.  Depending on the prior distributions, any of these portions may
% be absent from the integral; e.g., if the prior distribution on the
% concept being tested is already much better than the prior distribution
% for the other concept, it may be the case that one will never select the
% other concept even if a low test value is returned.

% First find the thresholds between the three portions of the integral
% [Tstar, Tlower, Tupper] = findIntegrationLimits( analysis, infoState,
% accumCost, R, tol)
Tstar = [];
Tlower = [];
Tupper = [];

% fprintf(1,'FINDING ROOTS -------------------------------------------
\n',[])
%     fprintf(1,'current Analysis = %1.1f\n',analysis)
%     fprintf(1,'infoState.mean = %12.8f  %12.8f\n',infoState.mean)
%     fprintf(1,'infoState.stdev = %12.8f  %12.8f\n',infoState.stdev)
%     fprintf(1,'infoState.done = %12.8f %12.8f %12.8f
%12.8f\n',infoState.done)
%     fprintf(1,'accumCost = %4.3f\n',accumCost)
[Tstar, Tlower,
Tupper,xup,rel_error_up,xdown,rel_error_down,priorStdevZeroflag] =
findIntegrationLimitsNR( analysis, infoState, accumCost, R, tol);
% fprintf(1,'Tlower = %12.8f\n',Tlower)
% fprintf(1,'Tupper = %12.8f\n',Tupper)
% fprintf(1,'DONE FINDING ROOTS---------------------------------------
\n',[])


% Once the thresholds have been found, compute the integrals.  Analytical
% expressions are used for the first and third portions of the integral,
% while the intermediate portion is computed via numerical integration over
% E_U_Test_X.

% If Tstar is empty, then all three integrals must be computed; else, only
% the lower and upper integrals need to be computed.
```

```
LowerInt = [];
UpperInt = [];
InterInt = [];

% Extract variables from analysis and infostate
alternative = analyses.alternative(analysis);
analysisStdev = analyses.stdev(analysis);
priorStdev = infoState.stdev(alternative);
priorMean = infoState.mean(alternative);
stdevSumSq = priorStdev^2 + analysisStdev^2;

% perform the analysis
newInfoState = infoState;
newInfoState.stdev(alternative) = priorStdev * analysisStdev /
sqrt(stdevSumSq);
newInfoState.done(analysis) = 1;
newAccumCost = accumCost + analyses.cost(analysis);

% determine which analyses to perform next
nextAnalyses = find(not(newInfoState.done));
nextAnalysis = nextAnalyses(1);
nextAccumCost = newAccumCost + analyses.cost(nextAnalysis);
nextAlternative = analyses.alternative(nextAnalysis);
nextAnalysisStdev = analyses.stdev(nextAnalysis);

% Extract mu_uA, mu_uB, stdev_uA, and stdev_uB
stdev_uA = newInfoState.stdev(1);
stdev_uB = newInfoState.stdev(2);

% Define variables
if alternative==1  %A is being tested
    mu_uX=priorMean;
    mu_uY=newInfoState.mean(2);
    stdev_uX=stdev_uA;
    stdev_uY=stdev_uB;
    mu_uA_givenX = @(X) (priorMean*analysisStdev^2 +
X*priorStdev^2)/(stdevSumSq);
    mu_uB_givenX = @(X) newInfoState.mean(2);
elseif alternative==2  %B is being tested
    mu_uX=priorMean;
    mu_uY=newInfoState.mean(1);
    stdev_uX=stdev_uB;
    stdev_uY=stdev_uA;
    mu_uA_givenX = @(X) newInfoState.mean(1);
    mu_uB_givenX = @(X) (priorMean*analysisStdev^2 +
X*priorStdev^2)/(stdevSumSq);
else
    error('TestX must be either 1 for A or 2 for B.')
end

if priorStdevZeroflag == 1
    if R == 0
        EselectUp = (mu_uX-newAccumCost);
        EselectDown = (mu_uY-newAccumCost);
    else % R > 0
        EselectUp = 1/R*(1-exp(-R*(mu_uX -
newAccumCost)+stdev_uX^2*R^2/2));
```

```
        EselectDown = 1/R*(1-exp(-R*(mu_uY -
newAccumCost)+stdev_uY^2*R^2/2));
    end
    Etest =
E_U_Test_X(mu_uA_givenX(priorMean),mu_uB_givenX(priorMean),stdev_uA,stdev_u
B,...
        nextAccumCost,nextAnalysisStdev,R,nextAlternative);
    expectedUtility = max([EselectUp,EselectDown,Etest]);
else

    if isequal(Tstar,[])
        if R == 0
            LowerInt = (mu_uY-
newAccumCost)*normcdf(Tlower,priorMean,max(sqrt(stdevSumSq),tol));

            UpperInt = (analysisStdev^2*priorMean/stdevSumSq-
newAccumCost)*...
                (1-normcdf(Tupper,priorMean,max(sqrt(stdevSumSq),tol)))+...
                priorStdev^2/(sqrt(2*pi)*stdevSumSq)*...
                (sqrt(stdevSumSq)*exp(-(Tupper-
priorMean)^2/(2*stdevSumSq))...
                +priorMean*sqrt(pi/2)*...
                (1-erf((Tupper-priorMean)/(sqrt(2)*sqrt(stdevSumSq)))));

        else %R ~= 0
            LowerInt = 1/R*(1-exp(-R*(mu_uY -
newAccumCost)+stdev_uY^2*R^2/2))*...
                normcdf(Tlower,priorMean,max(sqrt(stdevSumSq),tol));

            UpperInt = 1/R*(1-
normcdf(Tupper,priorMean,max(sqrt(stdevSumSq),tol)))...
                - 1/R*(exp(stdev_uX^2*R^2/2 ...
                +R*newAccumCost+R^2*priorStdev^4/(2*(stdevSumSq))-
R*priorMean))...
                *(1-normcdf(Tupper,(priorMean-
R*priorStdev^2),max(sqrt(stdevSumSq),tol)));
        end
        if sqrt(stdevSumSq)<tol
            if priorMean <= Tupper && priorMean >= Tlower
                InterInt =
E_U_Test_X(mu_uA_givenX(priorMean),mu_uB_givenX(priorMean),stdev_uA,stdev_u
B,...
                nextAccumCost,nextAnalysisStdev,R,nextAlternative);
            else % priorMean is not between Tlower and Tupper
                InterInt = 0;
            end
        else %sqrt(stdevSumSq) >=tol
            % If the limits of integration are too far away from the
            % "interesting" things happening in the normal probability
            % density function, the adaptive quadrature will not find that
            % very small range of interesting-ness and will return a value
            % of zero for the integral.  Thus, if the limits of integration
            % are more than 14 times the standard deviation of the normal
            % distribution, we will replace them with +/- 7 times the
            % standard deviation of the normal distribution.
            if 14*sqrt(stdevSumSq)>=(Tupper - Tlower)
                % Active portion of pdf is much larger than integration
                % bounds, so don't change the bounds
% ---------------------------------------------------------------------------
% INTEGRATE OVER MIDDLE BRANCH ---------USE GIVEN BOUNDS-------------------
            InterInt = quad(@(sampleVal)
(normpdf(sampleVal,priorMean,sqrt(stdevSumSq)).*...

E_U_Test_X(mu_uA_givenX(sampleVal),mu_uB_givenX(sampleVal),stdev_uA,stdev_u
B,...
                nextAccumCost,nextAnalysisStdev,R,nextAlternative)),...
                Tlower,Tupper,tol);
% ---------------------------------------------------------------------------
% ---------------------------------------------------------------------------
            else % The active portion of the pdf smaller than the
                % integration bounds, so the bounds should be changed to
                % only integrate over the active portion
                pdfMax = priorMean+7*sqrt(stdevSumSq);
                pdfMin = priorMean-7*sqrt(stdevSumSq);
                if Tupper <= pdfMax && Tupper >= pdfMin
                    upperbound = Tupper;
                else
                    upperbound = pdfMax;
                end
                if Tlower >= pdfMin && Tlower <= pdfMax
                    lowerbound = Tlower;
                else
                    lowerbound = pdfMin;
                end
% ---------------------------------------------------------------------------
% INTEGRATE OVER MIDDLE BRANCH ---------USE BOUNDS DETERMINED FROM PDF-----
            InterInt = quad(@(sampleVal)
(normpdf(sampleVal,priorMean,sqrt(stdevSumSq)).*...

E_U_Test_X(mu_uA_givenX(sampleVal),mu_uB_givenX(sampleVal),stdev_uA,stdev_u
B,...
                nextAccumCost,nextAnalysisStdev,R,nextAlternative)),...
                lowerbound,upperbound,tol);
% ---------------------------------------------------------------------------
% ---------------------------------------------------------------------------
            end
        end
    else % Tlower and Tupper should be empty
        if R == 0
            LowerInt = (mu_uY-
newAccumCost)*normcdf(Tstar,priorMean,max(sqrt(stdevSumSq),tol));

            UpperInt = (analysisStdev^2*priorMean/stdevSumSq-
newAccumCost)*...
                (1-normcdf(Tstar,priorMean,max(sqrt(stdevSumSq),tol)))+...
                priorStdev^2/(sqrt(2*pi)*stdevSumSq)*...
                (sqrt(stdevSumSq)*exp(-(Tstar-
priorMean)^2/(2*stdevSumSq))...
                +priorMean*sqrt(pi/2)*...
                (1-erf((Tstar-priorMean)/(sqrt(2)*sqrt(stdevSumSq)))));

        else % R ~= 0
            UpperInt = 1/R*(1-
normcdf(Tstar,priorMean,max(sqrt(stdevSumSq),tol)))...
```

```
                - 1/R*(exp(stdev_uX^2*R^2/2 ...
                +R*newAccumCost+R^2*priorStdev^4/(2*(stdevSumSq))-
R*priorMean))...
                *(1-normcdf(Tstar,(priorMean-
R*priorStdev^2),max(sqrt(stdevSumSq),tol)));

            LowerInt = 1/R*(1-exp(-R*(mu_uY -
newAccumCost)+stdev_uY^2*R^2/2))*...
                normcdf(Tstar,priorMean,max(sqrt(stdevSumSq),tol));
        end



    end

    expectedUtility = sum([LowerInt,UpperInt,InterInt]);
end
```

findIntegrationLimitsNR.m

```
function [Tstar, Tlower,
Tupper,xup,rel_error_up,xdown,rel_error_down,priorStdevZeroflag] =
findIntegrationLimitsNR( analysis, infoState, accumCost, R, tol)

global analyses

% This function finds the integration limits for the second-to-last
% decision node.

% Extract variables from analysis and infostate
alternative = analyses.alternative(analysis);
analysisStdev = analyses.stdev(analysis);
priorStdev = infoState.stdev(alternative);
priorMean = infoState.mean(alternative);
stdevSumSq = priorStdev^2 + analysisStdev^2;

if priorStdev == 0
    Tstar = [];
    Tlower = [];
    Tupper = [];
    xup = [];
    rel_error_up = [];
    xdown = [];
    rel_error_down = [];
    priorStdevZeroflag = 1;
else
    priorStdevZeroflag = 0;
    if alternative == 1
        otherAlternative = 2;
    elseif alternative ==2
        otherAlternative = 1;
    else
        error('Invalid input for variable alternative.  Must be either 1 or
2')
    end
    otherStdev = infoState.stdev(otherAlternative);
    otherMean = infoState.mean(otherAlternative);

    % perform the analysis
    newInfoState = infoState;
    newInfoState.stdev(alternative) = priorStdev * analysisStdev /
sqrt(stdevSumSq);
    newInfoState.done(analysis) = 1;
    newAccumCost = accumCost + analyses.cost(analysis);

    % determine which analyses to perform next
    nextAnalyses = find(not(newInfoState.done));
    nextAccumCost = newAccumCost + analyses.cost(nextAnalyses(1));
    nextAnalysis = nextAnalyses(1);
    nextAlternative = analyses.alternative(nextAnalysis);
    nextAnalysisStdev = analyses.stdev(nextAnalysis);


%     % Extract mu_uA, mu_uB, stdev_uA, and stdev_uB
%     mu_uA = newInfoState.mean(1);
%     mu_uB = newInfoState.mean(2);
%     stdev_uA = newInfoState.stdev(1);
%     stdev_uB = newInfoState.stdev(2);
%
%
%     % Define variables for solving for V_TXStar
%     if alternative==1  %A is being tested
%         mu_uX=mu_uA;
%         mu_uY=mu_uB;
%         stdev_uX=stdev_uA;
%         stdev_uY=stdev_uB;
%     elseif alternative==2  %B is being tested
%         mu_uX=mu_uB;
%         mu_uY=mu_uA;
%         stdev_uX=stdev_uB;
%         stdev_uY=stdev_uA;
%     else
%         error('TestX must be either 1 for A or 2 for B.')
%     end


    % % First, find the intersection of Select A and Select B.
    % stdev_Vtx=sqrt(stdev_uX^2+analysisStdev^2);
    % %Standard deviation for distribution of uX given the value of the
test
    % stdev_XgivenLQ=sqrt(stdev_uX^2*analysisStdev^2/stdev_Vtx^2);
    % Critical test value at which the E(uX)=E(uY) after testing X
%     V_TXStar=(1/priorStdev^2)*(stdevSumSq*mu_uY-
analysisStdev^2*priorMean+R/2*(priorStdev^2*analysisStdev^2-
stdevSumSq*stdev_uY^2));
    V_TXStar=(1/priorStdev^2)*(stdevSumSq*otherMean-
analysisStdev^2*priorMean+R/2*(priorStdev^2*analysisStdev^2-
stdevSumSq*otherStdev^2));
    % mu_uX given V_TXStar
    mu_uX_given_V_TXStar = (priorMean*analysisStdev^2 +
V_TXStar*priorStdev^2)/stdevSumSq;
```

```
    % Then test the utility of performing the next analysis at
mu_uX_given_V_TXStar
    nextInfoState = newInfoState;
    nextInfoState.mean(alternative) = mu_uX_given_V_TXStar;

    mu_uA = nextInfoState.mean(1);
    mu_uB = nextInfoState.mean(2);
    stdev_uA = nextInfoState.stdev(1);
    stdev_uB = nextInfoState.stdev(2);
    expUtilTest =
E_U_Test_X(mu_uA,mu_uB,stdev_uA,stdev_uB,nextAccumCost,nextAnalysisStdev,R,
nextAlternative);

    % Get the value of Select A (or SelecV_t B) at mu_given_V_TXStar
    % expUtilSelect = utilFunction(mu_uX_given_V_TXStar, stdev_XgivenLQ, R)
    expUtilSelect = utilFunction(mu_uA-newAccumCost, stdev_uA, R);
    expUtilSelect = utilFunction(mu_uB-newAccumCost, stdev_uB, R);


    if expUtilSelect >= expUtilTest
    %     then the test should not be performed
        Tlower = [];
        Tupper = [];
        Tstar = V_TXStar;
        xup = [];
        xdown = [];
        rel_error_up = [];
        rel_error_down = [];
    else
    %     use Newton Raphson find Tlower and Tupper
        Tstar = [];

        sigma_X = priorStdev;
        sigma_Y = otherStdev;
        sigma_delta = nextAnalysisStdev;
        sigma_epsilon = analysisStdev;
        mu_Y = otherMean;
        mu_X = priorMean;
        c_1 = newAccumCost;
        c_2 = nextAccumCost - newAccumCost;

        if R == 0;
            %RISK NEUTRAL
            if alternative == nextAlternative
                % SAME CONCEPT TESTED TWICE
                fuOverfprimeu = @(x) ((((mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 -
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2)) .* (1 + erf((-(-mu_Y .* sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + (sigma_X .^ 2) .* (sigma_epsilon .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* exp(-((-(-mu_Y .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* sqrt(0.2e1) .* (pi .*
(sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)
+ sigma_delta .^ 2)) .^ (-0.1e1 ./ 0.2e1) ./ 0.2e1 + (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2 .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) .* (1 - erf((-(-mu_Y .* sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + ((mu_Y - c_1 - c_2) .* (1 + erf((-(-mu_Y .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) -
mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2
./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./
0.2e1 - ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2
+ sigma_epsilon .^ 2)) + c_1) ./ ((sigma_X .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 .* (1 + erf((-(-mu_Y .* sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + 0.2e1 .* ((mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 -
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2)) .* pi .^ (-0.1e1 ./ 0.2e1) .*
exp(-((-(-mu_Y .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) - mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 +
sigma_epsilon .^ 2) - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2)) .^ 2 ./ (2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .*
sigma_delta .^ 2))) .* (-sigma_delta .^ 2 ./ sigma_epsilon .^ 2 - sigma_X
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)) .* ((2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .*
sigma_delta .^ 2) .^ (-0.3e1 ./ 0.2e1)) - (sigma_X .^ 2) .* (sigma_epsilon
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* (-(-mu_Y .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
```

```
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* exp(-((-(-mu_Y .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) - mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 +
sigma_epsilon .^ 2) - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2)) .^ 2 ./ (2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .*
sigma_delta .^ 2))) .* sqrt(0.2e1) .* (pi .* (sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^ 2)) .^ (-0.1e1
./ 0.2e1) + (sigma_X .^ 4 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .^ 2 .* (1 - erf((-(-mu_Y .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) - 0.2e1 .* (sigma_X .^ 2) .* (sigma_epsilon .^
2) ./ ((sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2) .* (mu_X .* sigma_epsilon
.^ 2 + x .* sigma_X .^ 2) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((-(-mu_Y .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) -
mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2
./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.3e1 ./ 0.2e1)) + (mu_Y
- c_1 - c_2) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((-(-mu_Y .* sigma_X .^ 2
.* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) -
(sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)));

                flOverfprimel = @(x) ((((mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 -
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2)) .* (1 + erf((-(-mu_Y .* sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + (sigma_X .^ 2) .* (sigma_epsilon .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* exp(-((-(-mu_Y .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* sqrt(0.2e1) .* (pi .*
(sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)
+ sigma_delta .^ 2)) .^ (-0.1e1 ./ 0.2e1) ./ 0.2e1 + (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2 .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) .* (1 - erf((-(-mu_Y .* sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + ((mu_Y - c_1 - c_2) .* (1 + erf((-(-mu_Y .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) -
mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2
./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./
0.2e1 - mu_Y + c_1) ./ ((sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_delta .^ 2 .* (1 + erf((-(-mu_Y .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + 0.2e1 .* ((mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 -
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2)) .* pi .^ (-0.1e1 ./ 0.2e1) .*
exp(-((-(-mu_Y .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) - mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 +
sigma_epsilon .^ 2) - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2)) .^ 2 ./ (2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .*
```

```
sigma_delta .^ 2))) .* (-sigma_delta .^ 2 ./ sigma_epsilon .^ 2 - sigma_X
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)) .* ((2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .*
sigma_delta .^ 2) .^ (-0.3e1 ./ 0.2e1)) - (sigma_X .^ 2) .* (sigma_epsilon
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* (-(-mu_Y .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* exp(-((-(-mu_Y .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) - mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 +
sigma_epsilon .^ 2) - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2)) .^ 2 ./ (2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .*
sigma_delta .^ 2))) .* sqrt(0.2e1) .* (pi .* (sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^ 2)) .^ (-0.1e1
./ 0.2e1) + (sigma_X .^ 4 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .^ 2 .* (1 - erf((-(-mu_Y .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) - 0.2e1 .* (sigma_X .^ 2) .* (sigma_epsilon .^
2) ./ ((sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2) .* (mu_X .* sigma_epsilon
.^ 2 + x .* sigma_X .^ 2) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((-(-mu_Y .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) -
mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2
./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.3e1 ./ 0.2e1)) + (mu_Y
- c_1 - c_2) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((-(-mu_Y .* sigma_X .^ 2
.* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)));
```

```
                f_lowerSameR0 = @(x) (((mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 -
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2)) .* (1 + erf((-(-mu_Y .* sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + (sigma_X .^ 2) .* (sigma_epsilon .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* exp(-((-(-mu_Y .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* sqrt(0.2e1) .* (pi .*
(sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)
+ sigma_delta .^ 2)) .^ (-0.1e1 ./ 0.2e1) ./ 0.2e1 + (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2 .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) .* (1 - erf((-(-mu_Y .* sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + ((mu_Y - c_1 - c_2) .* (1 + erf((-(-mu_Y .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) -
mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2
./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./
0.2e1 - mu_Y + c_1;

                f_upperSameR0 = @(x) (((mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 -
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2)) .* (1 + erf((-(-mu_Y .* sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + (sigma_X .^ 2) .* (sigma_epsilon .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* exp(-((-(-mu_Y .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* sqrt(0.2e1) .* (pi .*
(sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)
+ sigma_delta .^ 2)) .^ (-0.1e1 ./ 0.2e1) ./ 0.2e1 + (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2 .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) .* (1 - erf((-(-mu_Y .* sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - mu_Y .*
sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) + 2 .* sigma_delta .^ 2)) + ((mu_Y - c_1 - c_2) .* (1 + erf((-(-mu_Y .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) -
mu_Y .* sigma_delta .^ 2 + (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2
./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2) - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) .* (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./
0.2e1 - ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2
+ sigma_epsilon .^ 2)) + c_1;

%                 First check for a valid bracket for the upper root
                if f_upperSameR0(V_TXStar)*f_upperSameR0(V_TXStar + 100)>0
                    Tupper = Inf;
                    xup = [];
                    rel_error_up = [];
                else
                    % NEWTON-RAPHSON ALGORITHM
                    ii = 1;
                    xup(1) = V_TXStar;
                    rel_error_up(1) = 1; %Initialize error at one
        %             Find Tupper
                    while rel_error_up(ii) > tol
                        ii = ii+1;
                        xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
                        rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
                        if ii > 15
                            xup(ii)=Inf;
                            break
                        end
                    end
                    if xup(length(xup))>=V_TXStar
                        Tupper = xup(length(xup));
                    else
                        % restart NEWTON-RAPHSON ALGORITHM
                        ii = 1;
                        xup=[];
                        xup(1) = V_TXStar+3*analysisStdev;
                        rel_error_up = [];
                        rel_error_up(1) = 1; %Initialize error at one
              %           Find Tupper
                        while rel_error_up(ii) > tol
                            ii = ii+1;
                            xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
                            rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
                            if ii > 15
                                xup(ii)=Inf;
                                break
                            end
                        end
                        Tupper = xup(length(xup));
                    end
                end
%                 % NEWTON-RAPHSON ALGORITHM
%                 ii = 1;
%                 xup(1) = V_TXStar;
%                 rel_error_up(1) = 1; %Initialize error at one
%     %             Find Tupper
%                 while rel_error_up(ii) > tol
%                     ii = ii+1;
%                     xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
%                     rel_error_up(ii) = abs((xup(ii)-xup(ii-1))/xup(ii));
%                     if ii > 15
%                         xup(ii)=Inf;
%                         break
%                     end
%                 end
%                 if xup(length(xup))>=V_TXStar
%                     Tupper = xup(length(xup));
%                 else
%                     % restart NEWTON-RAPHSON ALGORITHM
%                     ii = 1;
%                     xup=[];
%                     xup(1) = V_TXStar+3*analysisStdev;
%                     rel_error_up = [];
%                     rel_error_up(1) = 1; %Initialize error at one
%         %             Find Tupper
%                     while rel_error_up(ii) > tol
%                         ii = ii+1;
%                         xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
%                         rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
%                         if ii > 15
%                             xup(ii)=Inf;
%                             break
%                         end
%                     end
%                     Tupper = xup(length(xup));
%                 end
%                 Check for a valid bracket for the lower root
                if f_lowerSameR0(V_TXStar)*f_lowerSameR0(V_TXStar - 100)>0
                    Tlower = -Inf;
                    xdown = [];
                    rel_error_down = [];
                else
                    jj = 1;
                    xdown(1) = V_TXStar;
                    rel_error_down(1) = 1; %Initialize error at one
        %             Find Tlower
                    while rel_error_down(jj) > tol
                        jj = jj+1;
                        xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-
1));
                        rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
                        if jj > 15
                            xdown(jj)=-Inf;
                            break
                        end
                    end
                    if xdown(length(xdown))<=V_TXStar
                        Tlower = xdown(length(xdown));
                    else %restart NEWTON RAPHSON ALGORITHM
                        jj = 1;
                        xdown = [];
                        xdown(1) = V_TXStar - 3*analysisStdev;
                        rel_error_down = [];
                        rel_error_down(1) = 1; %Initialize error at one
            %             Find Tlower
                        while rel_error_down(jj) > tol
                            jj = jj+1;
                            xdown(jj) = xdown(jj-1) -
flOverfprimel(xdown(jj-1));
                            rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
                            if jj > 15
                                xdown(jj)=-Inf;
                                break
                            end
                        end
                        Tlower = xdown(length(xdown));
                    end
                end
%                 jj = 1;
%                 xdown(1) = V_TXStar;
%                 rel_error_down(1) = 1; %Initialize error at one
%     %             Find Tlower
%                 while rel_error_down(jj) > tol
%                     jj = jj+1;
%                     xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-1));
%                     rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
%                     if jj > 15
%                         xdown(jj)=-Inf;
%                         break
%                     end
%                 end
%                 if xdown(length(xdown))<=V_TXStar
%                     Tlower = xdown(length(xdown));
%                 else %restart NEWTON RAPHSON ALGORITHM
%                       jj = 1;
%                       xdown = [];
%                       xdown(1) = V_TXStar - 3*analysisStdev;
%                       rel_error_down = [];
%                       rel_error_down(1) = 1; %Initialize error at one
%          %              Find Tlower
%                       while rel_error_down(jj) > tol
%                           jj = jj+1;
%                           xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-
1));
%                           rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
%                           if jj > 15
%                               xdown(jj)=-Inf;
%                               break
%                           end
%                       end
%                       Tlower = xdown(length(xdown));
%                   end
            else % test one concept, then test the next concept
                 % ONE CONCEPT AFTER ANOTHER
                fuOverfprimeu = @(x) (((mu_Y .* sigma_delta .^ 2 - (c_1 +
c_2) .* (sigma_Y .^ 2 + sigma_delta .^ 2)) .* (1 + erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) + (sigma_Y .^ 2) .* exp(-((-(-(mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^
2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)))
.* sqrt(0.2e1) .* (pi .* (sigma_Y .^ 2 + sigma_delta .^ 2)) .^ (-0.1e1 ./
0.2e1) ./ 0.2e1 + (sigma_Y .^ 2 .* mu_Y .* (1 - erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) + (((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1 - c_2) .* (1 + erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) + c_1) ./ (-0.2e1 .* (mu_Y .* sigma_delta .^ 2 - (c_1 + c_2) .*
(sigma_Y .^ 2 + sigma_delta .^ 2)) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((-(-
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 +
mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^
2 + 2 .* sigma_delta .^ 2))) .* (-sigma_X .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_Y .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ (sigma_Y .^ 2) .* ((2 .*
```

```
sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.3e1 ./ 0.2e1)) + (-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .* (-sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)
.* sigma_Y .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2) .* exp(-((-(-(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2 - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y)
.^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2))) .* sqrt(0.2e1) .* (pi
.* (sigma_Y .^ 2 + sigma_delta .^ 2)) .^ (-0.1e1 ./ 0.2e1) + 0.2e1 .* mu_Y
.* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((-(-(mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2 -
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)))
.* (-sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2 -
sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) .*
((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.3e1 ./ 0.2e1)) +
(sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* (1 + erf((-(-(mu_X
.* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) - c_1 - c_2) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2))) .* (-sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2) ./ (sigma_Y .^ 2) .* ((2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) - (sigma_X .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2)));

                f1Overfprime1 = @(x) (((mu_Y .* sigma_delta .^ 2 - (c_1 +
c_2) .* (sigma_Y .^ 2 + sigma_delta .^ 2)) .* (1 + erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) + (sigma_Y .^ 2) .* exp(-((-(-(mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^
2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)))
.* sqrt(0.2e1) .* (pi .* (sigma_Y .^ 2 + sigma_delta .^ 2)) .^ (-0.1e1 ./
0.2e1) ./ 0.2e1 + (sigma_Y .^ 2 .* mu_Y .* (1 - erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) + (((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1 - c_2) .* (1 + erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ 0.2e1 - mu_Y + c_1) ./ (-0.2e1
.* (mu_Y .* sigma_delta .^ 2 - (c_1 + c_2) .* (sigma_Y .^ 2 + sigma_delta
.^ 2)) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((-(-(mu_X .* sigma_epsilon .^ 2
+ x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2
- (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)))
.* (-sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2 -
sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./
(sigma_Y .^ 2) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.3e1
./ 0.2e1)) + (-(-(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2 - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y)
./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .* (-sigma_X .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2 - sigma_X .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) .* exp(-((-(-(mu_X
.* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2))) .* sqrt(0.2e1) .* (pi .* (sigma_Y .^ 2 + sigma_delta .^
2)) .^ (-0.1e1 ./ 0.2e1) + 0.2e1 .* mu_Y .* pi .^ (-0.1e1 ./ 0.2e1) .*
exp(-((-(-(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2
+ sigma_epsilon .^ 2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 +
mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^
2 + 2 .* sigma_delta .^ 2))) .* (-sigma_X .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_Y .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) .* ((2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.3e1 ./ 0.2e1)) + (sigma_X .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* (1 + erf((-(-(mu_X .* sigma_epsilon .^ 2 + x .*
sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2 -
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-
0.1e1 ./ 0.2e1)))) ./ 0.2e1 - ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1 - c_2) .* pi .^ (-0.1e1
./ 0.2e1) .* exp(-((-(-(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2 - (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y)
.^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2))) .* (-sigma_X .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^ 2 - sigma_X .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ (sigma_Y .^ 2)
.* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)));

                f_lowerNotSameR0 = @(x) ((mu_Y .* sigma_delta .^ 2 - (c_1 +
c_2) .* (sigma_Y .^ 2 + sigma_delta .^ 2)) .* (1 + erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) + (sigma_Y .^ 2) .* exp(-((-(-(mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^
2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)))
.* sqrt(0.2e1) .* (pi .* (sigma_Y .^ 2 + sigma_delta .^ 2)) .^ (-0.1e1 ./
0.2e1) ./ 0.2e1 + (sigma_Y .^ 2 .* mu_Y .* (1 - erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) + (((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1 - c_2) .* (1 + erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ 0.2e1 - mu_Y + c_1;

                f_upperNotSameR0 = @(x) ((mu_Y .* sigma_delta .^ 2 - (c_1 +
c_2) .* (sigma_Y .^ 2 + sigma_delta .^ 2)) .* (1 + erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) + (sigma_Y .^ 2) .* exp(-((-(-(mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_Y .^
2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)))
.* sqrt(0.2e1) .* (pi .* (sigma_Y .^ 2 + sigma_delta .^ 2)) .^ (-0.1e1 ./
0.2e1) ./ 0.2e1 + (sigma_Y .^ 2 .* mu_Y .* (1 - erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1))) ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) + (((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1 - c_2) .* (1 + erf((-(-(mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* sigma_Y .^ 2 - (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 + mu_Y .*
sigma_delta .^ 2) ./ sigma_Y .^ 2 - mu_Y) .* (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) + c_1;
% -------------------- ititial bracket test------------------------------
%                  Check for a valid bracket for the upper root
%                      upperRootFuncAtVTX = f_upperNotSameR0(V_TXStar)
%                      upperRootFuncAtVTXplus100sigmas =
f_upperNotSameR0(V_TXStar + 100*stdevSumSq)
```

```
                if f_upperNotSameR0(V_TXStar)*f_upperNotSameR0(V_TXStar +
100)>0
                    Tupper = Inf;
                    xup = [];
                    rel_error_up = [];
                else
        %             NEWTON-RAPHSON ALGORITHM
                    ii = 1;
                    xup(1) = V_TXStar;
                    rel_error_up(1) = 1; %Initialize error at one
        %             Find Tupper
                    while rel_error_up(ii) > tol
                        ii = ii+1;
                        xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
                        rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
                        if ii > 15
                            xup(ii)=Inf;
                            break
                        end
                    end
                    if xup(length(xup))>=V_TXStar
                        Tupper = xup(length(xup));
                    else
            %           restart NEWTON-RAPHSON ALGORITHM
                        ii = 1;
                        xup = [];
                        xup(1) = V_TXStar+3*analysisStdev;
                        rel_error_up = [];
                        rel_error_up(1) = 1; %Initialize error at one
            %             Find Tupper
                        while rel_error_up(ii) > tol
                            ii = ii+1;
                            xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
                            rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
                            if ii > 15
                                xup(ii)=Inf;
                                break
                            end
                        end
                        Tupper = xup(length(xup));
                    end
                end
% ------------------end ititial bracket test------------------------------
% -------------------no ititial bracket test------------------------------
% %             NEWTON-RAPHSON ALGORITHM
%                 ii = 1;
%                 xup(1) = V_TXStar;
%                 rel_error_up(1) = 1; %Initialize error at one
%     %             Find Tupper
%                 while rel_error_up(ii) > tol
%                     ii = ii+1;
%                     xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
%                     rel_error_up(ii) = abs((xup(ii)-xup(ii-1))/xup(ii));
%                     if ii > 15
%                         xup(ii)=Inf;
%                           break
%                       end
%                   end
%                   if xup(length(xup))>=V_TXStar
%                       Tupper = xup(length(xup));
%                   else
%         %             restart NEWTON-RAPHSON ALGORITHM
%                       ii = 1;
%                       xup = [];
%                       xup(1) = V_TXStar+3*analysisStdev;
%                       rel_error_up = [];
%                       rel_error_up(1) = 1; %Initialize error at one
%         %               Find Tupper
%                       while rel_error_up(ii) > tol
%                           ii = ii+1;
%                           xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
%                           rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
%                           if ii > 15
%                               xup(ii)=Inf;
%                               break
%                           end
%                       end
%
%                       Tupper = xup(length(xup));
%                  end
% ---------------end no ititial bracket test------------------------------
% -----------------------initial bracket test-----------------------------
%                Check for a valid bracket for the lower root
%                  lowerRootFuncAtVTX = f_lowerNotSameR0(V_TXStar)
%                  lowerRootFuncAtVTXminus100sigmas =
f_lowerNotSameR0(V_TXStar-100*stdevSumSq)
                if f_lowerNotSameR0(V_TXStar)*f_lowerNotSameR0(V_TXStar-
100)>0
                    Tlower = -Inf;
                    xdown = [];
                    rel_error_down = [];
                else
                    jj = 1;
                    xdown(1) = V_TXStar;
                    rel_error_down(1) = 1; %Initialize error at one
        %             Find Tlower
                    while rel_error_down(jj) > tol
                        jj = jj+1;
                        xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-
1));
                        rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
                        if jj > 15
                            xdown(jj)=-Inf;
                            break
                        end
                    end
                    if xdown(length(xdown))<=V_TXStar
                        Tlower = xdown(length(xdown));
                    else % restart NEWTON RAPHSON ALGORITHM
                        jj = 1;
                        xdown = [];
                        xdown(1) = V_TXStar - 3*analysisStdev;
                        rel_error_down = [];
                        rel_error_down(1) = 1; %Initialize error at one
            %             Find Tlower
                        while rel_error_down(jj) > tol
                            jj = jj+1;
                            xdown(jj) = xdown(jj-1) -
flOverfprimel(xdown(jj-1));
                            rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
                            if jj > 15
                                xdown(jj)=-Inf;
                                break
                            end
                        end
                        Tlower = xdown(length(xdown));
                    end
               end
% --------------------end initial bracket test----------------------------
% ---------------------no initial bracket test----------------------------
%                  jj = 1;
%                  xdown(1) = V_TXStar;
%                  rel_error_down(1) = 1; %Initialize error at one
%     %              Find Tlower
%                  while rel_error_down(jj) > tol
%                      jj = jj+1;
%                      xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-1));
%                      rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
%                      if jj > 15
%                          xdown(jj)=-Inf;
%                          break
%                      end
%                  end
%                  if xdown(length(xdown))<=V_TXStar
%                      Tlower = xdown(length(xdown));
%                  else % restart NEWTON RAPHSON ALGORITHM
%                      jj = 1;
%                      xdown = [];
%                      xdown(1) = V_TXStar - 3*analysisStdev;
%                      rel_error_down = [];
%                      rel_error_down(1) = 1; %Initialize error at one
%         %              Find Tlower
%                      while rel_error_down(jj) > tol
%                          jj = jj+1;
%                          xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-
1));
%                          rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
%                          if jj > 15
%                              xdown(jj)=-Inf;
%                              break
%                          end
%                      end
%                      Tlower = xdown(length(xdown));
%                  end
% -----------------end no initial bracket test-----------------------------
            end
        else % R > 0;
             % RISK AVERSE
            if alternative == nextAlternative
                % SAME CONCEPT TESTED TWICE
                fuOverfprimeu = @(x) (((1 - erf(((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R)
./ 0.2e1 - exp(((-2 .* R .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^
2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* (sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^ 2) + 2 .* R .*
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2) + R .^ 2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^
2 + R .^ 2 .* sigma_X .^ 4 .* sigma_epsilon .^ 4 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .^ 2) ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* (1 -
erf(((((sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) + sigma_delta .^ 2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X
.^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2 - 2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 -
((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2)) + (R .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1
./ 0.2e1)))) ./ R ./ 0.2e1 + (0.1e1 - exp(-(R .* (mu_Y - c_1 - c_2)) +
(sigma_Y .^ 2 .* R .^ 2) ./ 0.2e1)) .* (1 + erf(((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R
./ 0.2e1 - 0.1e1 ./ R .* (0.1e1 - exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 +
x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1)) + (R .^ 2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) ./ 0.2e1))) ./ (-pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2)) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./ R +
(sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .* exp(((-2 .* R .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) + sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .* (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) + R .^ 2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + R .^ 2 .* sigma_X .^ 4 .*
sigma_epsilon .^ 4 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2) ./ (2 .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) +
2 .* sigma_delta .^ 2))) .* (1 - erf(((((sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^ 2) .* (2 .* mu_Y -
sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^
2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .* (mu_X .* sigma_epsilon
.^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 +
sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)) + (R .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))) .* ((2 .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) +
2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) + exp(((-2 .* R .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) + sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .* (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) + R .^ 2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + R .^ 2 .* sigma_X .^ 4 .*
sigma_epsilon .^ 4 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2) ./ (2 .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) +
2 .* sigma_delta .^ 2))) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) +
sigma_delta .^ 2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^
2 - 2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2
+ sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) + (R .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2))) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2)) .* (-
sigma_delta .^ 2 ./ sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2)) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./
0.2e1)) ./ R + (0.1e1 - exp(-(R .* (mu_Y - c_1 - c_2)) + (sigma_Y .^ 2 .* R
.^ 2) ./ 0.2e1)) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
```

```
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2)) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./ R -
(sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* exp(-(R .* ((mu_X
.* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) - c_1)) + (R .^ 2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1));

               f1Overfprime1 = @(x) (((1 - erf(((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R)
./ 0.2e1 - exp(((-2 .* R .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^
2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* (sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^ 2) + 2 .* R .*
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2) + R .^ 2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^
2 + R .^ 2 .* sigma_X .^ 4 .* sigma_epsilon .^ 4 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .^ 2) ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* (1 -
erf(((((sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) + sigma_delta .^ 2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X
.^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2 - 2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 -
((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2)) + (R .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1
./ 0.2e1)))) ./ R ./ 0.2e1 + (0.1e1 - exp(-(R .* (mu_Y - c_1 - c_2)) +
(sigma_Y .^ 2 .* R .^ 2) ./ 0.2e1)) .* (1 + erf(((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R
./ 0.2e1 - 0.1e1 ./ R .* (0.1e1 - exp(-(R .* (mu_Y - c_1)) + (sigma_Y .^ 2
.* R .^ 2) ./ 0.2e1))) ./ (-pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) +
sigma_delta .^ 2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^
```

```
2 - 2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2
+ sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2)) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./ R +
(sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .* exp(((-2 .* R .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) + sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .* (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) + R .^ 2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + R .^ 2 .* sigma_X .^ 4 .*
sigma_epsilon .^ 4 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2) ./ (2 .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) +
2 .* sigma_delta .^ 2))) .* (1 - erf(((((sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^ 2) .* (2 .* mu_Y -
sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^
2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .* (mu_X .* sigma_epsilon
.^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon .^ 2 .* (sigma_X .^ 2 +
sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)) + (R .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))) .* ((2 .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) +
2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) + exp(((-2 .* R .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) + sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .* (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) + R .^ 2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2 + R .^ 2 .* sigma_X .^ 4 .*
sigma_epsilon .^ 4 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .^ 2) ./ (2 .*
sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) +
2 .* sigma_delta .^ 2))) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) +
sigma_delta .^ 2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^
2 - 2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2
+ sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) + (R .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2))) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2)) .* (-
sigma_delta .^ 2 ./ sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2)) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./
0.2e1)) ./ R + (0.1e1 - exp(-(R .* (mu_Y - c_1 - c_2)) + (sigma_Y .^ 2 .* R
.^ 2) ./ 0.2e1)) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
```

```
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .^ 2 ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2)) .* (-sigma_delta .^ 2 ./
sigma_epsilon .^ 2 - sigma_X .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2))
.* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./ R);

                f_lowerSame = @(x) ((1 - erf(((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R)
./ 0.2e1 - exp(((-2 .* R .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^
2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* (sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^ 2) + 2 .* R .*
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2) + R .^ 2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^
2 + R .^ 2 .* sigma_X .^ 4 .* sigma_epsilon .^ 4 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .^ 2) ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* (1 -
erf(((((sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) + sigma_delta .^ 2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X
.^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2 - 2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 -
((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2)) + (R .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1
./ 0.2e1)))) ./ R ./ 0.2e1 + (0.1e1 - exp(-(R .* (mu_Y - c_1 - c_2)) +
(sigma_Y .^ 2 .* R .^ 2) ./ 0.2e1)) .* (1 + erf(((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R
./ 0.2e1 - 0.1e1 ./ R .* (0.1e1 - exp(-(R .* (mu_Y - c_1)) + (sigma_Y .^ 2
.* R .^ 2) ./ 0.2e1));

                f_upperSame = @(x) ((1 - erf(((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R)
./ 0.2e1 - exp(((-2 .* R .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^
2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* (sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^ 2) + 2 .* R .*
(c_1 + c_2) .* (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + sigma_delta .^ 2) + R .^ 2 .* sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^
2 + R .^ 2 .* sigma_X .^ 4 .* sigma_epsilon .^ 4 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .^ 2) ./ (2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2))) .* (1 -
erf(((((sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon
.^ 2) + sigma_delta .^ 2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X
.^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .*
sigma_delta .^ 2 - 2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./
sigma_epsilon .^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 -
((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2)) + (R .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1
./ 0.2e1)))) ./ R ./ 0.2e1 + (0.1e1 - exp(-(R .* (mu_Y - c_1 - c_2)) +
(sigma_Y .^ 2 .* R .^ 2) ./ 0.2e1)) .* (1 + erf(((((sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) + sigma_delta .^
2) .* (2 .* mu_Y - sigma_Y .^ 2 .* R) + R .* sigma_X .^ 2 .* sigma_epsilon
.^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* sigma_delta .^ 2 - 2 .*
(mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* sigma_delta .^ 2) ./ sigma_X .^ 2 ./ sigma_epsilon
.^ 2 .* (sigma_X .^ 2 + sigma_epsilon .^ 2)) ./ 0.2e1 - ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2))) .* ((2 .* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R
./ 0.2e1 - 0.1e1 ./ R .* (0.1e1 - exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 +
x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1)) + (R .^ 2
.* sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2)) ./ 0.2e1));

%                 Check for a valid bracket for the upper root
%                 if f_upperSame(V_TXStar)*f_upperSame(V_TXStar +100
*analysisStdev)>0
%                     Tupper = Inf;
%                     xup = [];
%                     rel_error_up = [];
%                 else
%         %              NEWTON-RAPHSON ALGORITHM
%                     ii = 1;
%                     xup(1) = V_TXStar;
%                     rel_error_up(1) = 1; %Initialize error at one
%         %              Find Tupper
%                     while rel_error_up(ii) > tol
%                         ii = ii+1;
%                         xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
%                         rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
%                         if ii > 15
%                             xup(ii)=Inf;
%                             break
%                         end
%                     end
%                     if xup(length(xup))>=V_TXStar
%                         Tupper = xup(length(xup));
%                     else
%             %           restart NEWTON-RAPHSON ALGORITHM
%                         ii = 1;
%                         xup = [];
%                         xup(1) = V_TXStar + 3*analysisStdev;
%                         rel_error_up = [];
%                         rel_error_up(1) = 1; %Initialize error at one
%             %             Find Tupper
%                         while rel_error_up(ii) > tol
%                             ii = ii+1;
%                             xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-
1));
%                             rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
%                             if ii > 15
%                                 xup(ii)=Inf;
%                                 break
%                             end
%                         end
%                         Tupper = xup(length(xup));
%                     end
%                 end

    %             NEWTON-RAPHSON ALGORITHM
                ii = 1;
                xup(1) = V_TXStar;
                rel_error_up(1) = 1; %Initialize error at one
    %             Find Tupper
                while rel_error_up(ii) > tol
                    ii = ii+1;
                    xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
                    rel_error_up(ii) = abs((xup(ii)-xup(ii-1))/xup(ii));
                    if ii > 15
                        xup(ii)=Inf;
                        break
                    end
                end
                if xup(length(xup))>=V_TXStar
                    Tupper = xup(length(xup));
                else
        %           restart NEWTON-RAPHSON ALGORITHM
                    ii = 1;
                    xup = [];
                    xup(1) = V_TXStar + 3*analysisStdev;
                    rel_error_up = [];
                    rel_error_up(1) = 1; %Initialize error at one
        %             Find Tupper
```

```
                    while rel_error_up(ii) > tol
                        ii = ii+1;
                        xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
                        rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
                        if ii > 15
                            xup(ii)=Inf;
                            break
                        end
                    end
                    Tupper = xup(length(xup));
                end

%               Check for a valid bracket for the lower root
%                 if f_lowerSame(V_TXStar)*f_lowerSame(V_TXStar -
100*analysisStdev)>0
%                     Tlower = -Inf;
%                     xdown = [];
%                     rel_error_down = [];
%                 else
%                     jj = 1;
%                     xdown(1) = V_TXStar;
%                     rel_error_down(1) = 1; %Initialize error at one
%         %             Find Tlower
%                     while rel_error_down(jj) > tol
%                         jj = jj+1;
%                         xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-
1));
%                         rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
%                         if jj > 15
%                             xdown(jj)=-Inf;
%                             break
%                         end
%                     end
%                     if xdown(length(xdown))<=V_TXStar
%                         Tlower = xdown(length(xdown));
%                     else % restart NEWTON RAPHSON ALGORITHM
%                         jj = 1;
%                         xdown = [];
%                         xdown(1) = V_TXStar - 3*analysisStdev;
%                         rel_error_down = [];
%                         rel_error_down(1) = 1; %Initialize error at one
%             %             Find Tlower
%                         while rel_error_down(jj) > tol
%                             jj = jj+1;
%                             xdown(jj) = xdown(jj-1) -
flOverfprimel(xdown(jj-1));
%                             rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
%                             if jj > 15
%                                 xdown(jj)=-Inf;
%                                 break
%                             end
%                         end
%                         Tlower = xdown(length(xdown));
%                     end
%                  end

                jj = 1;
                xdown(1) = V_TXStar;
                rel_error_down(1) = 1; %Initialize error at one
    %             Find Tlower
                while rel_error_down(jj) > tol
                    jj = jj+1;
                    xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-1));
                    rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
                    if jj > 15
                        xdown(jj)=-Inf;
                        break
                    end
                end
                if xdown(length(xdown))<=V_TXStar
                    Tlower = xdown(length(xdown));
                else % restart NEWTON RAPHSON ALGORITHM
                    jj = 1;
                    xdown = [];
                    xdown(1) = V_TXStar - 3*analysisStdev;
                    rel_error_down = [];
                    rel_error_down(1) = 1; %Initialize error at one
        %             Find Tlower
                    while rel_error_down(jj) > tol
                        jj = jj+1;
                        xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-
1));
                        rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
                        if jj > 15
                            xdown(jj)=-Inf;
                            break
                        end
                    end
                    Tlower = xdown(length(xdown));
                end
            else % test one concept, then test the next concept
                 % ONE CONCEPT AFTER ANOTHER
                fuOverfprimeu = @(x) (((1 - erf(((((sigma_Y .^ 2 +
sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .*
sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1
- mu_Y) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./
0.2e1)))) ./ R) ./ 0.2e1 - exp(((-2 .* R .* mu_Y .* (sigma_Y .^ 2 +
sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .* (sigma_Y .^ 2 + sigma_delta .^
2) + R .^ 2 .* sigma_Y .^ 2 .* sigma_delta .^ 2 + R .^ 2 .* sigma_Y .^ 4)
./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2))) .* (1 - erf(((((sigma_Y .^
2 + sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .*
sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2) ./ 0.2e1 - mu_Y + (R .* sigma_Y .^ 2)) .* ((2 .* sigma_Y .^ 2
+ 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R ./ 0.2e1 + (0.1e1 -
exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^
2 + sigma_epsilon .^ 2) - c_1 - c_2)) + (sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R .^ 2) ./ 0.2e1)) .* (1 +
erf(((((sigma_Y .^ 2 + sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .*
sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^
2) .^ (-0.1e1 ./ 0.2e1)))) ./ R ./ 0.2e1 - (0.1e1 - exp(-(R .* ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) - c_1)) + (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* R .^ 2) ./ 0.2e1)) ./ R) ./ (-pi .^ (-0.1e1 ./
0.2e1) .* exp(-((((sigma_Y .^ 2 + sigma_delta .^ 2) .* (2 .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) - sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* R) + R .* sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta
.^ 2) ./ sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) .* (sigma_Y .^ 2 + sigma_delta .^ 2) .* (sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) ./ (sigma_Y .^ 2) .* ((2 .* sigma_Y
.^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./ R + exp(((-2 .* R .*
mu_Y .* (sigma_Y .^ 2 + sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .*
(sigma_Y .^ 2 + sigma_delta .^ 2) + R .^ 2 .* sigma_Y .^ 2 .* sigma_delta
.^ 2 + R .^ 2 .* sigma_Y .^ 4) ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^
2))) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_Y .^ 2 + sigma_delta .^
2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^
2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^
2 + sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .* sigma_delta .^ 2 - 2
.* mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1 - mu_Y + (R .*
sigma_Y .^ 2)) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)) .*
(sigma_Y .^ 2 + sigma_delta .^ 2) .* (sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) ./ (sigma_Y .^ 2) .* ((2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./ R + (sigma_X .^ 2) ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2) .* exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 + x
.* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1 - c_2)) +
(sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)
.* R .^ 2) ./ 0.2e1) .* (1 + erf(((((sigma_Y .^ 2 + sigma_delta .^ 2) .* (2
.* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .*
mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .* ((2 .*
sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ 0.2e1 +
(0.1e1 - exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1 - c_2)) + (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R .^ 2) ./
0.2e1)) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_Y .^ 2 + sigma_delta
.^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .* sigma_delta .^ 2 -
2 .* mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .^ 2 ./ (2
.* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)) .* (sigma_Y .^ 2 + sigma_delta .^
2) .* (sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) ./ (sigma_Y .^
2) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./
R - (sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* exp(-(R .*
((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) - c_1)) + (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) .* R .^ 2) ./ 0.2e1));
```

```
                flOverfprimel = @(x) (((1 - erf(((((sigma_Y .^ 2 +
sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .*
sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1
- mu_Y) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./
0.2e1)))) ./ R) ./ 0.2e1 - exp(((-2 .* R .* mu_Y .* (sigma_Y .^ 2 +
sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .* (sigma_Y .^ 2 + sigma_delta .^
2) + R .^ 2 .* sigma_Y .^ 2 .* sigma_delta .^ 2 + R .^ 2 .* sigma_Y .^ 4)
./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2))) .* (1 - erf(((((sigma_Y .^
2 + sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .*
sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2) ./ 0.2e1 - mu_Y + (R .* sigma_Y .^ 2)) .* ((2 .* sigma_Y .^ 2
+ 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R ./ 0.2e1 + (0.1e1 -
exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^
2 + sigma_epsilon .^ 2) - c_1 - c_2)) + (sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R .^ 2) ./ 0.2e1)) .* (1 +
erf(((((sigma_Y .^ 2 + sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .*
sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^
2) .^ (-0.1e1 ./ 0.2e1)))) ./ R ./ 0.2e1 - (0.1e1 - exp(-(R .* (mu_Y -
c_1)) + (sigma_Y .^ 2 .* R .^ 2) ./ 0.2e1)) ./ R) ./ (-pi .^ (-0.1e1 ./
0.2e1) .* exp(-((((sigma_Y .^ 2 + sigma_delta .^ 2) .* (2 .* (mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) - sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^
2) .* R) + R .* sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta
.^ 2) ./ sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2)) .* (sigma_Y .^ 2 + sigma_delta .^ 2) .* (sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) ./ (sigma_Y .^ 2) .* ((2 .* sigma_Y
.^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./ R + exp(((-2 .* R .*
mu_Y .* (sigma_Y .^ 2 + sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .*
(sigma_Y .^ 2 + sigma_delta .^ 2) + R .^ 2 .* sigma_Y .^ 2 .* sigma_delta
.^ 2 + R .^ 2 .* sigma_Y .^ 4) ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^
2))) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_Y .^ 2 + sigma_delta .^
2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^
2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^
2 + sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .* sigma_delta .^ 2 - 2
.* mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1 - mu_Y + (R .*
sigma_Y .^ 2)) .^ 2 ./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)) .*
(sigma_Y .^ 2 + sigma_delta .^ 2) .* (sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) ./ (sigma_Y .^ 2) .* ((2 .* sigma_Y .^ 2 + 2 .*
sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./ R + (sigma_X .^ 2) ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2) .* exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 + x
.* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1 - c_2)) +
(sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2)
.* R .^ 2) ./ 0.2e1) .* (1 + erf(((((sigma_Y .^ 2 + sigma_delta .^ 2) .* (2
.* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .*
mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .* ((2 .*
sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ 0.2e1 +
(0.1e1 - exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./
(sigma_X .^ 2 + sigma_epsilon .^ 2) - c_1 - c_2)) + (sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R .^ 2) ./
0.2e1)) .* pi .^ (-0.1e1 ./ 0.2e1) .* exp(-((((sigma_Y .^ 2 + sigma_delta
.^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X
.^ 2 + sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .* sigma_delta .^ 2 -
2 .* mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .^ 2 ./ (2
.* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2)) .* (sigma_Y .^ 2 + sigma_delta .^
2) .* (sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) ./ (sigma_Y .^
2) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)) ./
R);

                f_lowerNotSame = @(x) ((1 - erf(((((sigma_Y .^ 2 +
sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .*
sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1
- mu_Y) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./
0.2e1)))) ./ R) ./ 0.2e1 - exp(((-2 .* R .* mu_Y .* (sigma_Y .^ 2 +
sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .* (sigma_Y .^ 2 + sigma_delta .^
2) + R .^ 2 .* sigma_Y .^ 2 .* sigma_delta .^ 2 + R .^ 2 .* sigma_Y .^ 4)
./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2))) .* (1 - erf(((((sigma_Y .^
2 + sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .*
sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2) ./ 0.2e1 - mu_Y + (R .* sigma_Y .^ 2)) .* ((2 .* sigma_Y .^ 2
+ 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R ./ 0.2e1 + (0.1e1 -
exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^
2 + sigma_epsilon .^ 2) - c_1 - c_2)) + (sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R .^ 2) ./ 0.2e1)) .* (1 +
erf(((((sigma_Y .^ 2 + sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .*
sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^
2) .^ (-0.1e1 ./ 0.2e1)))) ./ R ./ 0.2e1 - (0.1e1 - exp(-(R .* (mu_Y -
c_1)) + (sigma_Y .^ 2 .* R .^ 2) ./ 0.2e1)) ./ R;

                f_upperNotSame = @(x) ((1 - erf(((((sigma_Y .^ 2 +
sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2)
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .* sigma_Y .^ 2 .*
sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./ sigma_Y .^ 2) ./ 0.2e1
- mu_Y) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./
0.2e1)))) ./ R) ./ 0.2e1 - exp(((-2 .* R .* mu_Y .* (sigma_Y .^ 2 +
sigma_delta .^ 2) + 2 .* R .* (c_1 + c_2) .* (sigma_Y .^ 2 + sigma_delta .^
2) + R .^ 2 .* sigma_Y .^ 2 .* sigma_delta .^ 2 + R .^ 2 .* sigma_Y .^ 4)
./ (2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^ 2))) .* (1 - erf(((((sigma_Y .^
2 + sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^ 2 + x .* sigma_X
.^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^ 2 .*
sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .*
sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2) ./ 0.2e1 - mu_Y + (R .* sigma_Y .^ 2)) .* ((2 .* sigma_Y .^ 2
+ 2 .* sigma_delta .^ 2) .^ (-0.1e1 ./ 0.2e1)))) ./ R ./ 0.2e1 + (0.1e1 -
exp(-(R .* ((mu_X .* sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^
2 + sigma_epsilon .^ 2) - c_1 - c_2)) + (sigma_X .^ 2 .* sigma_epsilon .^ 2
./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R .^ 2) ./ 0.2e1)) .* (1 +
erf(((((sigma_Y .^ 2 + sigma_delta .^ 2) .* (2 .* (mu_X .* sigma_epsilon .^
2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) - sigma_X .^
2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 + sigma_epsilon .^ 2) .* R) + R .*
sigma_Y .^ 2 .* sigma_delta .^ 2 - 2 .* mu_Y .* sigma_delta .^ 2) ./
sigma_Y .^ 2) ./ 0.2e1 - mu_Y) .* ((2 .* sigma_Y .^ 2 + 2 .* sigma_delta .^
2) .^ (-0.1e1 ./ 0.2e1)))) ./ R ./ 0.2e1 - (0.1e1 - exp(-(R .* ((mu_X .*
sigma_epsilon .^ 2 + x .* sigma_X .^ 2) ./ (sigma_X .^ 2 + sigma_epsilon .^
2) - c_1)) + (sigma_X .^ 2 .* sigma_epsilon .^ 2 ./ (sigma_X .^ 2 +
sigma_epsilon .^ 2) .* R .^ 2) ./ 0.2e1)) ./ R;
%                 Check for a valid bracket for the upper root
%                 if
f_upperNotSame(V_TXStar)*f_upperNotSame(V_TXStar+100*analysisStdev)>0
%                     Tupper = Inf;
%                     xup = [];
%                     rel_error_up = [];
%                 else
%         %             NEWTON-RAPHSON ALGORITHM
%                     ii = 1;
%                     xup(1) = V_TXStar;
%                     rel_error_up(1) = 1; %Initialize error at one
%         %             Find Tupper
%                     while rel_error_up(ii) > tol
%                         ii = ii+1;
%                         xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
%                         rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
%                         if ii > 15
%                             xup(ii)=Inf;
%                             break
%                         end
%                     end
%                     if xup(length(xup))>=V_TXStar
%                         Tupper = xup(length(xup));
%                     else
%             %           restart NEWTON-RAPHSON ALGORITHM
%                         ii = 1;
%                         xup = [];
%                         xup(1) = V_TXStar + 3*analysisStdev;
%                         rel_error_up = [];
%                         rel_error_up(1) = 1; %Initialize error at one
%             %             Find Tupper
%                         while rel_error_up(ii) > tol
%                             ii = ii+1;
%                             xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-
1));
%                             rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
%                             if ii > 15
%                                 xup(ii)=Inf;
%                                 break
%                             end
%                         end
%                         Tupper = xup(length(xup));
%                     end
%                 end

    %             NEWTON-RAPHSON ALGORITHM
```

```
                ii = 1;
                xup(1) = V_TXStar;
                rel_error_up(1) = 1; %Initialize error at one
    %             Find Tupper
                while rel_error_up(ii) > tol
                    ii = ii+1;
                    xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
                    rel_error_up(ii) = abs((xup(ii)-xup(ii-1))/xup(ii));
                    if ii > 15
                        xup(ii)=Inf;
                        break
                    end
                end
                if xup(length(xup))>=V_TXStar
                    Tupper = xup(length(xup));
                else
        %           restart NEWTON-RAPHSON ALGORITHM
                    ii = 1;
                    xup = [];
                    xup(1) = V_TXStar + 3*analysisStdev;
                    rel_error_up = [];
                    rel_error_up(1) = 1; %Initialize error at one
        %             Find Tupper
                    while rel_error_up(ii) > tol
                        ii = ii+1;
                        xup(ii) = xup(ii-1) - fuOverfprimeu(xup(ii-1));
                        rel_error_up(ii) = abs((xup(ii)-xup(ii-
1))/xup(ii));
                        if ii > 15
                            xup(ii)=Inf;
                            break
                        end
                    end
                    Tupper = xup(length(xup));
                end
%               Check for a valid bracket for the lower root
%                 if f_lowerNotSame(V_TXStar)*f_lowerNotSame(V_TXStar-
100*analysisStdev)
%                     Tlower = -Inf;
%                     xdown = [];
%                     rel_error_down = [];
%                 else
%                     jj = 1;
%                     xdown(1) = V_TXStar;
%                     rel_error_down(1) = 1; %Initialize error at one
%         %             Find Tlower
%                     while rel_error_down(jj) > tol
%                         jj = jj+1;
%                         xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-
1));
%                         rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
%                         if jj > 15
%                             xdown(jj)=-Inf;
%                             break
%                         end
%                     end
%                      if xdown(length(xdown))<=V_TXStar
%                          Tlower = xdown(length(xdown));
%                      else % restart NEWTON RAPHSON ALGORITHM
%                          jj = 1;
%                          xdown = [];
%                          xdown(1) = V_TXStar - 3*analysisStdev;
%                          rel_error_down = [];
%                          rel_error_down(1) = 1; %Initialize error at one
%             %              Find Tlower
%                          while rel_error_down(jj) > tol
%                              jj = jj+1;
%                              xdown(jj) = xdown(jj-1) -
flOverfprimel(xdown(jj-1));
%                              rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
%                              if jj > 15
%                                  xdown(jj)=-Inf;
%                                  break
%                              end
%                          end
%                          Tlower = xdown(length(xdown));
%                      end
%                  end

                jj = 1;
                xdown(1) = V_TXStar;
                rel_error_down(1) = 1; %Initialize error at one
    %             Find Tlower
                while rel_error_down(jj) > tol
                    jj = jj+1;
                    xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-1));
                    rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
                    if jj > 15
                        xdown(jj)=-Inf;
                        break
                    end
                end
                if xdown(length(xdown))<=V_TXStar
                    Tlower = xdown(length(xdown));
                else % restart NEWTON RAPHSON ALGORITHM
                    jj = 1;
                    xdown = [];
                    xdown(1) = V_TXStar - 3*analysisStdev;
                    rel_error_down = [];
                    rel_error_down(1) = 1; %Initialize error at one
        %             Find Tlower
                    while rel_error_down(jj) > tol
                        jj = jj+1;
                        xdown(jj) = xdown(jj-1) - flOverfprimel(xdown(jj-
1));
                        rel_error_down(jj) = abs((xdown(jj)-xdown(jj-
1))/xdown(jj));
                        if jj > 15
                            xdown(jj)=-Inf;
                            break
                        end
                    end
                    Tlower = xdown(length(xdown));
                end

            end
        end
    end
end
```

normedTwoOnewRiskAversionFunction.m

```
function [SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction(delta_mu,delta_stdev,stdev_eps,Ca,R)

muB = 0;
% y-axis is delta_stdev, with sigma A and sigma B held constant at 1 at the
% top and bottom of the plot, respectively.
% if delta_stdev>= 0
%     stdevA = 1;
%     stdevB = stdevA - delta_stdev;
% else
%     stdevB = 1;
%     stdevA = delta_stdev + stdevB;
% end


muA = delta_mu+muB;

% y-axis is sigma A, sigma B is constant at 1
stdevB = 1;
stdevA = delta_stdev;


[SA,SB,AA,AB] =
TwoOnewRiskAversionFunction(muA,stdevA,muB,stdevB,stdev_eps,Ca,R);
```

normedTwoOnewRiskAversionFunction_OneAnalysis.m

```
function [SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(delta_mu,delta_stdev,stde
v_eps,Ca,R)

muB = 0;
% y-axis is delta_stdev, with sigma A and sigma B held constant at 1 at the
% top and bottom of the plot, respectively.
% if delta_stdev>= 0
%     stdevA = 1;
%     stdevB = stdevA - delta_stdev;
% else
%     stdevB = 1;
%     stdevA = delta_stdev + stdevB;
% end


muA = delta_mu+muB;
```

```
% y-axis is sigma A, sigma B is constant at 1
stdevB = 1;
stdevA = delta_stdev;


[SA,SB,AA,AB] =
TwoOnewRiskAversionFunction_OneAnalysis(muA,stdevA,muB,stdevB,stdev_eps,Ca,
R);
```

plotBoundaries_new.m

```
function [areas]=plotBoundaries_new(R,stdev_eps,Ca)
tic

num_steps = 10;
yvect_up = linspace(1,2,num_steps);
yvect_down = linspace(1,0,num_steps);

[SA,SB,AA,AB] = normedTwoOnewRiskAversionFunction_new(0,1,stdev_eps,Ca,R);

if SA > AA
    areas = -1; % Two region plot, rather than 4 region. Vertical split.
%     find intersection of SA and SB
        y = yvect_up(1);

        SASBu(1) = fzero(@(x) scanYMid_new(x,y,stdev_eps,Ca,R),0);
        SASBd(1) = fzero(@(x) scanYMid_new(x,y,stdev_eps,Ca,R),SASBu(1));

    for ii = 2:num_steps
        y_up = yvect_up(ii);
        y_down = yvect_down(ii);

        SASBu(ii) = fzero(@(x)
scanYMid_new(x,y_up,stdev_eps,Ca,R),SASBu(ii-1));
        SASBd(ii) = fzero(@(x)
scanYMid_new(x,y_down,stdev_eps,Ca,R),SASBd(ii-1));

    end
figure
    plot(SASBu,yvect_up,'k',SASBd,yvect_down,'k')
    axis([-5 5 0 2])
%     legend('SA=SB','location','Southeast')
    xlabel('\mu_A')
    ylabel('\sigma_A')
    title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
            ', Cost = ',num2str(Ca),...
            ', R = ',num2str(R)]})
   hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-C',num2str(Ca),'-
',datestr(now,'HH.MM.SS'),'.fig'])
    close

else
    [SA1,SB1,AA1,AB1] =
normedTwoOnewRiskAversionFunction_new(5,1,stdev_eps,Ca,R);
    if SA1 > AA1
```

```
        areas = 1; % 4 region plot rather than two region
    %     find intersections of SA/AA,SA/AB,SB/AA,SB/AB
        SAAA = zeros(1,num_steps);
        SAAB = SAAA;
        SBAA = SAAA;
        SBAB = SAAA;

        y = yvect_up(1);
            SAAA(1) = fzero(@(x) scanYRup_new(x,y,stdev_eps,Ca,R),[0,5]);
            SAAB(1) = fzero(@(x)
scanYRdown_new(x,y,stdev_eps,Ca,R),[SAAA(1)]);
            SBAA(1) = fzero(@(x) scanYLup_new(x,y,stdev_eps,Ca,R),[-
1*SAAA(1)]);
            SBAB(1) = fzero(@(x)
scanYLdown_new(x,y,stdev_eps,Ca,R),[SBAB(1)]);



        %%
        for ii = 2:num_steps
        y_up = yvect_up(ii);
        y_down = yvect_down(ii);

            SAAA(ii) = fzero(@(x)
scanYRup_new(x,y_up,stdev_eps,Ca,R),SAAA(ii-1));
            SAAB(ii) = fzero(@(x)
scanYRdown_new(x,y_down,stdev_eps,Ca,R),SAAB(ii-1));
            SBAA(ii) = fzero(@(x)
scanYLup_new(x,y_up,stdev_eps,Ca,R),SBAA(ii-1));
            SBAB(ii) = fzero(@(x)
scanYLdown_new(x,y_down,stdev_eps,Ca,R),SBAB(ii-1));
        end
        figure

plot(SAAA,yvect_up,'k',SBAA,yvect_up,'k',SAAB,yvect_down,'k',SBAB,yvect_dow
n,'k',[SAAA(1),SBAA(1)],[1,1],'k')
        axis([-5 5 0 2])
%
legend('SA=AA','SB=AA','SA=AB','SB=AB','AA=AB','location','Southeast')
        xlabel('\mu_A')
        ylabel('\sigma_A')
        title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
                ', Cost = ',num2str(Ca),...
                ', R = ',num2str(R)]})
       hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-C',num2str(Ca),'-
',datestr(now,'HH.MM.SS'),'.fig'])
        close
        right = [[SAAA(linspace(num_steps,1,num_steps))]';SAAB'];
        left = [[SBAA(linspace(num_steps,1,num_steps))]';SBAB'];
        yvector =
[yvect_up([linspace(num_steps,1,num_steps)])';yvect_down'];
        csvwrite(['R',num2str(R),'-eps',num2str(stdev_eps),'-
C',num2str(Ca),'-',datestr(now,'HH.MM.SS'),'.csv'],[yvector,right,left])
    else
        areas = -2; %Two region plot.  Horizontal split. (Should be the
case for zero cost).
        AAAB = [-5,5];
        y = [1 1];
        figure
        plot(AAAB,y,'k')
        axis([-5 5 0 2])
%         legend('AA=AB','location','Southeast')
        xlabel('\mu_A')
        ylabel('\sigma_A')
        title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
                ', Cost = ',num2str(Ca),...
                ', R = ',num2str(R)]})
       hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-C',num2str(Ca),'-
',datestr(now,'HH.MM.SS'),'.fig'])
        close
    end

end

toc
```

plotBoundaries_OneAnalysis_new.m

```
function [areas]=plotBoundaries_OneAnalysis_new(R,stdev_eps,Ca)
tic

num_steps = 10;
yvect_up = linspace(1,2,num_steps);
yvect_down = linspace(1,0,num_steps);

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(0,1,stdev_eps,Ca,R);

if SA > AA
    areas = -1; % Two region plot, rather than 4 region. Vertical split.
%     find intersection of SA and SB
        y = yvect_up(1);

        SASBu(1) = fzero(@(x) scanYMid_OA_new(x,y,stdev_eps,Ca,R),0);
        SASBd(1) = fzero(@(x)
scanYMid_OA_new(x,y,stdev_eps,Ca,R),SASBu(1));

    for ii = 2:num_steps
        y_up = yvect_up(ii);
        y_down = yvect_down(ii);

        SASBu(ii) = fzero(@(x)
scanYMid_OA_new(x,y_up,stdev_eps,Ca,R),SASBu(ii-1));
        SASBd(ii) = fzero(@(x)
scanYMid_OA_new(x,y_down,stdev_eps,Ca,R),SASBd(ii-1));

    end
figure
    plot(SASBu,yvect_up,'k',SASBd,yvect_down,'k')
    axis([-5 5 0 2])
%     legend('SA=SB','SA=SB','location','Southeast')
    xlabel('\mu_A')
    ylabel('\sigma_A')
    title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
            ', Cost = ',num2str(Ca),...
            ', R = ',num2str(R)]})
   hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-C',num2str(Ca),'-
',datestr(now,'HH.MM.SS'),'.fig'])
    close

else
    [SA1,SB1,AA1,AB1] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(5,1,stdev_eps,Ca,R);
    if SA1 >= AA1
        areas = 1; % 4 region plot rather than two region
    %     find intersections of SA/AA,SA/AB,SB/AA,SB/AB
        SAAA = zeros(1,num_steps);
        SAAB = SAAA;
        SBAA = SAAA;
        SBAB = SAAA;
```

RunDoEs_TwoTwo.m

```
scriptStart = tic;
diary(['runDoEDiary-',datestr(now,'HH.MM.SS'),'.txt'])

% Run DoE's

levels = 21;
totalRuns = levels^2;

muB = 0;
stdevB = 1;
tol = 1e-5;

stdev_eps_vect = [1];
stdev_delta_vect = [0];
C1vect = [0.01];
C2vect = [0.1];
Rvect = [0.001];

mu_A = linspace(-5,5,levels);
stdev_A = linspace(0.1,1.9,levels);

% New normalization is symmetric about x-axis.  Stdev B is held at 1 for
% the lower two quadrants.  Stdev A is held at 1 for the upper two
% quadrants.  Mu A is varied within the plot; Stdev A and B are varied
% within the plot in the quadrants where they are not held constant.
% Separate plots are created for various values of Mu B, stdev_eps, Ca,
% and R.

for ee = 1:length(stdev_eps_vect);
    stdev_eps = stdev_eps_vect(ee);
    for dd = 1:length(stdev_delta_vect);
        stdev_delta = stdev_delta_vect(dd);
        for ff = 1:length(C1vect);
            C1 = C1vect(ff);
            for gg = 1:length(C2vect);
                C2 = C2vect(gg);
                for kk = 1:length(Rvect);
                R = Rvect(kk);

                    count = 0;
                    M = zeros(totalRuns,14);
                    timeElapsed = zeros(totalRuns,1);
                    for ii = 1:levels
                        for jj = 1:levels
                            count = count + 1;
                            fprintf(1,'-----------------------------------
\n',[])
                            fprintf(1,'Run %4.0f out of %4.0f\n', [count,
totalRuns])
                    % Run the calculation and store the inputs and outputs
in a file.
                            fprintf(1,'running
decisionProblem2_2Function(%8.6f, %8.6f, %6.2f, %6.2f, %8.6f, %8.6f, %8.6f,
%8.6f, %6.2f,
%1.0e)\n',[mu_A(ii),stdev_A(jj),muB,stdevB,stdev_eps,stdev_delta,C1,C2,R,to
l])
                            tic
                            [SA,SB,A1A,A1B,A2A,A2B] =
decisionProblem2_2Function(mu_A(ii),stdev_A(jj),muB,stdevB,stdev_eps,stdev_
delta,C1,C2,R,tol);
                            timeElapsed(count) = toc;
                            fprintf(1,'Elapsed Time: %12.8f seconds\n',
timeElapsed(count))
                            M(count,:) =
[mu_A(ii),stdev_A(jj),stdev_eps,stdev_delta,C1,C2,R,SA,SB,A1A,A1B,A2A,A2B,t
imeElapsed(count)];
                        end
                    end
                    fprintf(1,'-----------------------------------\n',[])
                    fprintf('Total Time for all runs: %12.8f
seconds\n',sum(timeElapsed))
                    fprintf(1,'---------End Block-----------------\n',[])
                    % Save data
                    csvwrite(['DoEData-R',num2str(R),...
                    '-eps',num2str(stdev_eps),...
                    '-delta',num2str(stdev_delta),...
                    '-C1',num2str(C1),...
                    '-C2',num2str(C2),...
                    '-',datestr(now,'HH.MM.SS'),'.csv'],M,0,0)

                    data = M;
                    num_blocks = size(data,1)/(levels^2);

                    for i = 1:num_blocks
                    % for i = 1:1
                    % Obtain one block of the experiment
                    % A block is a set of runs for which muA and stddevA
vary, but the other
                    % parameters stay constant
                    % separate data into inputs and responses
                    muA = reshape(data((totalRuns*(i-
1)+(1:(totalRuns))),1),[levels,levels]);
                    stdevA = reshape(data((totalRuns*(i-
1)+(1:(totalRuns))),2),[levels,levels]);
```

```
                    stdeveps= data((totalRuns*(i-1)+1),3);
                    stdevdelta= data((totalRuns*(i-1)+1),4);
                    C1= data((totalRuns*(i-1)+1),5);
                    C2 = data((totalRuns*(i-1)+1),6);
                    R = data((totalRuns*(i-1)+1),7);
                    SA = reshape(data((totalRuns*(i-
1)+(1:(totalRuns))),8),[levels,levels]);
                    SB = reshape(data((totalRuns*(i-
1)+(1:(totalRuns))),9),[levels,levels]);
                    A1A = reshape(data((totalRuns*(i-
1)+(1:(totalRuns))),10),[levels,levels]);
                    A1B = reshape(data((totalRuns*(i-
1)+(1:(totalRuns))),11),[levels,levels]);
                    A2A = reshape(data((totalRuns*(i-
1)+(1:(totalRuns))),12),[levels,levels]);
                    A2B = reshape(data((totalRuns*(i-
1)+(1:(totalRuns))),13),[levels,levels]);



                    figure
                    % Plot surfaces for each of six responses
                    % surf(muA,stdevA,HQB,1.0*ones(levels,levels))
                    hold on
                    surf(muA,stdevA,A2B,0.8*ones(levels,levels))
                    surf(muA,stdevA,A2A,1.0*ones(levels,levels))
                    surf(muA,stdevA,A1B,0.6*ones(levels,levels))
                    surf(muA,stdevA,A1A,0.4*ones(levels,levels))
                    surf(muA,stdevA,SB,0.2*ones(levels,levels))
                    surf(muA,stdevA,SA,0.0*ones(levels,levels))
                    colormap(hsv(6))
                    shading flat
                    legend('A2B','A2A','A1B','A1A','SB','SA')
                    xlabel('\mu_A')
                    ylabel('\sigma_A')
                    view(0,90)
                    title({'Maximum Expected Utilities of SA, SB, A1A, A1B,
A2A, and A2B';...
                        [' \sigma_\epsilon = ',num2str(stdeveps),...
                        ' ,\sigma_\delta = ',num2str(stdevdelta),...
                        ', Cost_1 = ',num2str(C1),...
                        ' ,Cost_2 = ',num2str(C2)]})
                    % save figure automatically, clear and move on to next
block
                    hgsave(['surf-R',num2str(R),...
                        '-eps',num2str(stdev_eps),...
                        '-delta',num2str(stdev_delta),...
                        '-C1',num2str(C1),...
                        '-C2',num2str(C2),...
                        '-',datestr(now,'HH.MM.SS'),'.fig'])

                    close

                    end

                end
            end
```

```
        end
    end
end
toc(scriptStart)
diary off
```

scanYLdown_new.m

```
function [diffSBAB] = scanYLdown_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] = normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);


diffSBAB = SB-AB;
```

scanYLdown_OA_new.m

```
function [diffSBAB] = scanYLdown_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSBAB = SB-AB;
```

scanYLup_new.m

```
function [diffSBAA] = scanYLup_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] = normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);


diffSBAA = SB-AA;
```

scanYLup_OA_new.m

```
function [diffSBAA] = scanYLup_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSBAA = SB-AA;
```

scanYMid_new.m

```
function [diffSASB] = scanYMid_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] = normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);
```

```
diffSASB = SA-SB;
```

scanYMid_OA_new.m

```
function [diffSASB] = scanYMid_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSASB = SA-SB;
```

scanYRdown_new.m

```
function [diffSAAB] = scanYRdown_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] = normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);


diffSAAB = SA-AB;
```

scanYRdown_OA_new.m

```
function [diffSAAB] = scanYRdown_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSAAB = SA-AB;
```

scanYRup_new.m

```
function [diffSAAA] = scanYRup_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] = normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);


diffSAAA = SA-AA;
```

scanYRup_OA_new.m

```
function [diffSAAA] = scanYRup_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSAAA = SA-AA;
```

Scenario1_TwoAnalyses_fromDoEData.m

```
% Script to run and save data on the VoI comparison for Scenario 1: One
% analysis


% Get samples and data from DoEData csv files
cd('C:\Users\stephanie\Documents\MATLAB\TwoTwo\TwoTwo_full_V4')
data2 = csvread('DoEData-R0.001-eps1-delta0-C10.01-C20.1-04.37.41.csv');
% data2 = csvread('DoEData-R1-eps1-delta0-C10.01-C20.1-10.28.36.csv');

% Full = [data1;data2];
Full = [data2];
samples = length(Full);
Full = [Full(:,1:2),0*ones(samples,1),1*ones(samples,1),Full(:,3:7),1e-
4*ones(samples,1),Full(:,8:size(Full,2))];

Trunc = zeros(size(Full));

X = [Full(:,1:2),Full(:,5:9)];
tol = Full(1,10);

stdev_1 = X(1,3);
stdev_2 = X(1,4);
C1 = X(1,5);
C2 = X(1,6);
R = X(1,7);

%%
% Run the full and truncated decision trees for each point, save the data
% to a csv file

for ii=1:samples

% %     Full
%     cd('C:\Users\stephanie\Documents\MATLAB\TwoTwo\TwoTwo_full_V4')
%     time1 = tic;
%     [SA,SB,A1A,A1B,A2A,A2B] =
decisionProblem2_2Function(X(ii,1),X(ii,2),0,1,X(ii,3),X(ii,4),X(ii,5),X(ii
,6),X(ii,7),tol);
%     elapsedFull = toc(time1);
%
%     Full(ii,:) =
[X(ii,1),X(ii,2),0,1,X(ii,3),X(ii,4),X(ii,5),X(ii,6),X(ii,7),tol,SA,SB,A1A,
A1B,A2A,A2B,elapsedFull];

%     Truncated
    cd('C:\Users\stephanie\Documents\MATLAB\TwoTwo\TwoTwo_TruncatedTree')
    time2 = tic;
    [SA_trunc,SB_trunc,A1A_trunc,A1B_trunc,A2A_trunc,A2B_trunc] =
TwoTwowRiskAversionFunction_OneAnalysis(X(ii,1),X(ii,2),0,1,X(ii,3),X(ii,4)
,X(ii,5),X(ii,6),X(ii,7),tol);
    elapsedTrunc = toc(time2);

    Trunc(ii,:) =
[X(ii,1),X(ii,2),0,1,X(ii,3),X(ii,4),X(ii,5),X(ii,6),X(ii,7),tol,SA_trunc,S
B_trunc,A1A_trunc,A1B_trunc,A2A_trunc,A2B_trunc,elapsedTrunc];

end
```

```
cd('C:\Users\stephanie\Documents\MATLAB\CompareVoI')
csvwrite(['Scenario1.TwoAnalyses.Full_',num2str(samples),'-
',datestr(now,'HH.MM.SS'),'.csv'],Full,0,0)
csvwrite(['Scenario1.TwoAnalyses.Trunc_',num2str(samples),'-
',datestr(now,'HH.MM.SS'),'.csv'],Trunc,0,0)

% test for the same preferred alternative, and record errors

Full_output = Full(:,11:16);
Trunc_output = Trunc(:,11:16);

[fullMaxes,fullLocations] = max(Full_output,[],2);
[truncMaxes,truncLocations] = max(Trunc_output,[],2);

VoIErrors = not(eq(fullLocations,truncLocations));
csvwrite(['Scenario1.OneAnalysis.VoIErrors_',num2str(samples),'-
',datestr(now,'HH.MM.SS'),'.csv'],VoIErrors,0,0)

%% Plot outputs

% plot(X(:,1),X(:,2),'o')
%%
ErrorsOnlyX = X(find(VoIErrors),:);

ErrorsOnlyFull = Full(find(VoIErrors),:);
ErrorsOnlyTrunc = Trunc(find(VoIErrors),:);
csvwrite(['Scenario1.TwoAnalyses.ErrorsOnlyFull_',num2str(samples),'-
',datestr(now,'HH.MM.SS'),'.csv'],ErrorsOnlyFull,0,0)
csvwrite(['Scenario1.TwoAnalyses.ErrorsOnlyTrunc_',num2str(samples),'-
',datestr(now,'HH.MM.SS'),'.csv'],ErrorsOnlyTrunc,0,0)

%%
plot(X(:,1),X(:,2),'k.','MarkerSize',4)
hold on
plot(ErrorsOnlyX(:,1),ErrorsOnlyX(:,2),'kx','LineWidth',3,'MarkerSize',10)
xlabel('\mu_A')
ylabel('\sigma_A')
legend('Agree','Disagree','Location','Best')
title({[' \sigma_1 = ',num2str(stdev_1),...
       ' \sigma_2 = ',num2str(stdev_2),...
       ', C_1 = ',num2str(C1),...
       ', C_2 = ',num2str(C2),...
       ', R = ',num2str(R)]})
hgsave(['R',num2str(R),'-eps',num2str(stdev_1),'-delta',num2str(stdev_2),'-
Cone',num2str(C1),'-Ctwo',num2str(C2),'-',datestr(now,'HH.MM.SS'),'.fig'])
```

TwoOnewRiskAversionFunction.m

```
function [SA,SB,AA,AB] = TwoOnewRiskAversionFunction(...

      mu_uA,stdev_uA,mu_uB,stdev_uB,stdev_eps,Ca,R,tol)


% New script including risk aversion
% tic
% -------------------------------------------------------------------------
% Define risk aversion.
% -------------------------------------------------------------------------
% R > 0  risk averse
% R < 0  risk seeking
% R = 0  risk neutral
% R = 0;

% -------------------------------------------------------------------------
% Define costs of analyses
% -------------------------------------------------------------------------
costs = [Ca,Ca];
% -------------------------------------------------------------------------
% Define distribution parameters
% -------------------------------------------------------------------------
% prior distributions on the utility of concepts A and B
% mu_uA =1;
% stdev_uA = 1;;
var_uA = stdev_uA^2;

% mu_uB = 0;
% stdev_uB = 1;
var_uB = stdev_uB^2;

% low quality analysis
% uLQ = uA + eps, where eps ~ N(0,10)
mu_eps = 0;
% stdev_eps = 0.125;
var_eps = stdev_eps^2;

% marginal distribution of low quality analysis
mu_uA_LQ = mu_uA + mu_eps;
var_uA_LQ = var_uA + var_eps;
stdev_uA_LQ = sqrt(var_uA_LQ);

mu_uB_LQ = mu_uB + mu_eps;
var_uB_LQ = var_uB + var_eps;
stdev_uB_LQ = sqrt(var_uB_LQ);
% -------------------------------------------------------------------------
% define conditional distributions
% -------------------------------------------------------------------------
% p(uI|uLQ) ~ N((uLQ*var_uI + mu_uI*var_eps)/(var_uI + var_eps),
%              (var_uI*var_eps)/(var_uI + var_eps))
mu_AgivenLQ = @ (uLQ) (uLQ*var_uA + mu_uA*var_eps)/(var_uA + var_eps);
var_AgivenLQ = (var_uA*var_eps)/(var_uA + var_eps);
stdev_AgivenLQ = sqrt(var_AgivenLQ);

mu_BgivenLQ = @ (uLQ) (uLQ*var_uB + mu_uB*var_eps)/(var_uB + var_eps);
var_BgivenLQ = (var_uB*var_eps)/(var_uB + var_eps);
stdev_BgivenLQ = sqrt(var_BgivenLQ);
% -------------------------------------------------------------------------

% -------------------------------------------------------------------------
% Evaluate decision alternatives
% -------------------------------------------------------------------------
% tol = 1.e-6;
```

```
SA = utilFunction(mu_uA,stdev_uA,R);
SB = utilFunction(mu_uB,stdev_uB,R);

AA = quadv(@(a) max(quad(@(b)((max(...
    utilFunction(mu_AgivenLQ(a)-
AnalysisCost([1,1],costs),stdev_AgivenLQ,R),...
    utilFunction(mu_BgivenLQ(b)-
AnalysisCost([1,1],costs),stdev_BgivenLQ,R)))...
    .*normpdf(b,mu_uB_LQ,stdev_uB_LQ)),...
    mu_uB_LQ-10*stdev_uB_LQ,mu_uB_LQ+10*stdev_uB_LQ, tol), (max(...
                utilFunction(mu_AgivenLQ(a)-
AnalysisCost([1,0],costs),stdev_AgivenLQ,R),...
                utilFunction(mu_uB-
AnalysisCost([1,0],costs),stdev_uB,R))))...
               .*normpdf(a,mu_uA_LQ,stdev_uA_LQ),...
          mu_uA_LQ-10*stdev_uA_LQ,mu_uA_LQ+10*stdev_uA_LQ, tol);

AB = quadv(@(b) max(quad(@(a)((max(...
    utilFunction(mu_AgivenLQ(a)-
AnalysisCost([1,1],costs),stdev_AgivenLQ,R),...
    utilFunction(mu_BgivenLQ(b)-
AnalysisCost([1,1],costs),stdev_BgivenLQ,R)))...
    .*normpdf(a,mu_uA_LQ,stdev_uA_LQ)),...
    mu_uA_LQ-10*stdev_uA_LQ,mu_uA_LQ+10*stdev_uA_LQ, tol), (max(...
                utilFunction(mu_uA-
AnalysisCost([0,1],costs),stdev_uA,R),...
                utilFunction(mu_BgivenLQ(b)-
AnalysisCost([0,1],costs),stdev_BgivenLQ,R))))...
               .*normpdf(b,mu_uB_LQ,stdev_uB_LQ),...
          mu_uB_LQ-10*stdev_uB_LQ,mu_uB_LQ+10*stdev_uB_LQ, tol);


% toc
```

TwoOnewRiskAversionFunction_OneAnalysis.m

```
function [SA,SB,AA,AB] =
TwoOnewRiskAversionFunction_OneAnalysis(mu_uA,stdev_uA,mu_uB,stdev_uB,stdev
_eps,Ca,R,tol)

% New script including risk aversion
% tic
% -------------------------------------------------------------------------
% Define risk aversion.
% -------------------------------------------------------------------------
% R > 0  risk averse
% R < 0  risk seeking
% R = 0  risk neutral
% R = 0;

% -------------------------------------------------------------------------
% Define costs of analyses
% -------------------------------------------------------------------------
costs = [Ca,Ca];
% -------------------------------------------------------------------------
% Define distribution parameters
% ------------------------------------------------------------------------
% prior distributions on the utility of concepts A and B
% mu_uA =1;
% stdev_uA = 1;;
var_uA = stdev_uA^2;

% mu_uB = 0;
% stdev_uB = 1;
var_uB = stdev_uB^2;

% low quality analysis
% uLQ = uA + eps, where eps ~ N(0,10)
mu_eps = 0;
% stdev_eps = 0.125;
var_eps = stdev_eps^2;

% marginal distribution of low quality analysis
mu_uA_LQ = mu_uA + mu_eps;
var_uA_LQ = var_uA + var_eps;
stdev_uA_LQ = sqrt(var_uA_LQ);

mu_uB_LQ = mu_uB + mu_eps;
var_uB_LQ = var_uB + var_eps;
stdev_uB_LQ = sqrt(var_uB_LQ);
% ------------------------------------------------------------------------
% define conditional distributions
% ------------------------------------------------------------------------
% p(uI|uLQ) ~ N((uLQ*var_uI + mu_uI*var_eps)/(var_uI + var_eps),
%                (var_uI*var_eps)/(var_uI + var_eps))
mu_AgivenLQ = @ (uLQ) (uLQ*var_uA + mu_uA*var_eps)/(var_uA + var_eps);
var_AgivenLQ = (var_uA*var_eps)/(var_uA + var_eps);
stdev_AgivenLQ = sqrt(var_AgivenLQ);

mu_BgivenLQ = @ (uLQ) (uLQ*var_uB + mu_uB*var_eps)/(var_uB + var_eps);
var_BgivenLQ = (var_uB*var_eps)/(var_uB + var_eps);
stdev_BgivenLQ = sqrt(var_BgivenLQ);
% ------------------------------------------------------------------------

% ------------------------------------------------------------------------
% Evaluate decision alternatives
% ------------------------------------------------------------------------
% tol = 1.e-6;

SA = utilFunction(mu_uA,stdev_uA,R);
SB = utilFunction(mu_uB,stdev_uB,R);

AA = quad(@(a) max(...
                utilFunction(mu_AgivenLQ(a)-
AnalysisCost([1,0],costs),stdev_AgivenLQ,R),...
                utilFunction(mu_uB-
AnalysisCost([1,0],costs),stdev_uB,R))...
               .*normpdf(a,mu_uA_LQ,stdev_uA_LQ),...
          mu_uA_LQ-10*stdev_uA_LQ,mu_uA_LQ+10*stdev_uA_LQ, tol);

AB = quad(@(b) max(...
                utilFunction(mu_uA-
AnalysisCost([0,1],costs),stdev_uA,R),...
                utilFunction(mu_BgivenLQ(b)-
AnalysisCost([0,1],costs),stdev_BgivenLQ,R))...
               .*normpdf(b,mu_uB_LQ,stdev_uB_LQ),...
          mu_uB_LQ-10*stdev_uB_LQ,mu_uB_LQ+10*stdev_uB_LQ, tol);

% toc
```

TwoTwowRiskAversionFunction_OneAnalysis.m

```
function [SA,SB,A1A,A1B,A2A,A2B] =
TwoTwowRiskAversionFunction_OneAnalysis(mu_uA,stdev_uA,mu_uB,stdev_uB,stdev
_eps,stdev_delta,C1,C2,R,tol)

% New script including risk aversion
% tic
% --------------------------------------------------------------------------
% Define risk aversion.
% --------------------------------------------------------------------------
% R > 0  risk averse
% R < 0  risk seeking
% R = 0  risk neutral
% R = 0;

% --------------------------------------------------------------------------
% Define costs of analyses
% --------------------------------------------------------------------------
costs = [C1,C1,C2,C2];
% --------------------------------------------------------------------------
% Define distribution parameters
% --------------------------------------------------------------------------
% prior distributions on the utility of concepts A and B
% mu_uA =1;
% stdev_uA = 1;;
var_uA = stdev_uA^2;

% mu_uB = 0;
% stdev_uB = 1;
var_uB = stdev_uB^2;

% Analysis 1 (LQ)
% uLQ = uA + eps, where eps ~ N(0,10)
mu_eps = 0;
% stdev_eps = 0.125;
var_eps = stdev_eps^2;

% marginal distributions of Analysis 1
mu_uA_LQ = mu_uA + mu_eps;
var_uA_LQ = var_uA + var_eps;
stdev_uA_LQ = sqrt(var_uA_LQ);

mu_uB_LQ = mu_uB + mu_eps;
var_uB_LQ = var_uB + var_eps;
stdev_uB_LQ = sqrt(var_uB_LQ);
```

```
% Analysis 2 (HQ)
% uHQ = uA + eps, where delta ~ N(0,10)
mu_delta = 0;
% stdev_eps = 0.125;
var_delta = stdev_delta^2;

% marginal distributions of Analysis 1
mu_uA_HQ = mu_uA + mu_delta;
var_uA_HQ = var_uA + var_delta;
stdev_uA_HQ = sqrt(var_uA_HQ);

mu_uB_HQ = mu_uB + mu_delta;
var_uB_HQ = var_uB + var_delta;
stdev_uB_HQ = sqrt(var_uB_HQ);
% ---------------------------------------------------------------------------
% define conditional distributions
% ---------------------------------------------------------------------------
% p(uI|uLQ) ~ N((uLQ*var_uI + mu_uI*var_eps)/(var_uI + var_eps),
%                (var_uI*var_eps)/(var_uI + var_eps))
% Analysis 1 (LQ)
mu_AgivenLQ = @ (uLQ) (uLQ*var_uA + mu_uA*var_eps)/(var_uA + var_eps);
var_AgivenLQ = (var_uA*var_eps)/(var_uA + var_eps);
stdev_AgivenLQ = sqrt(var_AgivenLQ);

mu_BgivenLQ = @ (uLQ) (uLQ*var_uB + mu_uB*var_eps)/(var_uB + var_eps);
var_BgivenLQ = (var_uB*var_eps)/(var_uB + var_eps);
stdev_BgivenLQ = sqrt(var_BgivenLQ);

% Analysis 2 (HQ)
mu_AgivenHQ = @ (uHQ) (uHQ*var_uA + mu_uA*var_delta)/(var_uA + var_delta);
var_AgivenHQ = (var_uA*var_delta)/(var_uA + var_delta);
stdev_AgivenHQ = sqrt(var_AgivenHQ);

mu_BgivenHQ = @ (uHQ) (uHQ*var_uB + mu_uB*var_delta)/(var_uB + var_delta);
var_BgivenHQ = (var_uB*var_delta)/(var_uB + var_delta);
stdev_BgivenHQ = sqrt(var_BgivenHQ);
% ---------------------------------------------------------------------------

% ---------------------------------------------------------------------------
% Evaluate decision alternatives
% ---------------------------------------------------------------------------
% tol = 1.e-6;

SA = utilFunction(mu_uA,stdev_uA,R);
SB = utilFunction(mu_uB,stdev_uB,R);

A1A = quad(@(a) max(...
                utilFunction(mu_AgivenLQ(a)-
AnalysisCost([1,0,0,0],costs),stdev_AgivenLQ,R),...
                utilFunction(mu_uB-
AnalysisCost([1,0,0,0],costs),stdev_uB,R))...
               .*normpdf(a,mu_uA_LQ,stdev_uA_LQ),...
          mu_uA_LQ-10*stdev_uA_LQ,mu_uA_LQ+10*stdev_uA_LQ, tol);

A1B = quad(@(b) max(...
                utilFunction(mu_uA-
AnalysisCost([0,1,0,0],costs),stdev_uA,R),...
```

```
                utilFunction(mu_BgivenLQ(b)-
AnalysisCost([0,1,0,0],costs),stdev_BgivenLQ,R))...
               .*normpdf(b,mu_uB_LQ,stdev_uB_LQ),...
          mu_uB_LQ-10*stdev_uB_LQ,mu_uB_LQ+10*stdev_uB_LQ, tol);

A2A = quad(@(a) max(...
                utilFunction(mu_AgivenHQ(a)-
AnalysisCost([0,0,1,0],costs),stdev_AgivenHQ,R),...
                utilFunction(mu_uB-
AnalysisCost([0,0,1,0],costs),stdev_uB,R))...
               .*normpdf(a,mu_uA_HQ,stdev_uA_HQ),...
          mu_uA_HQ-10*stdev_uA_HQ,mu_uA_HQ+10*stdev_uA_HQ, tol);

A2B = quad(@(b) max(...
                utilFunction(mu_uA-
AnalysisCost([0,0,0,1],costs),stdev_uA,R),...
                utilFunction(mu_BgivenHQ(b)-
AnalysisCost([0,0,0,1],costs),stdev_BgivenHQ,R))...
               .*normpdf(b,mu_uB_HQ,stdev_uB_HQ),...
          mu_uB_HQ-10*stdev_uB_HQ,mu_uB_HQ+10*stdev_uB_HQ, tol);

% toc
```

utilFunction.m

```
function expectedUtility = utilFunction(dollarsMean, dollarsStdev,
riskAversion)

% we need to account for risk aversion here.
% we assume that u = (1 - exp(-a*dollars))/a
% if dollars is normally distributed then the expected value is:
% E[u] = (1 - exp(-a*mean + 0.5*(a*stdev)^2))/a;
if riskAversion == 0
    expectedUtility = dollarsMean;
else
    expectedUtility = (1 - exp(-riskAversion.*dollarsMean +
0.5*(riskAversion.*dollarsStdev).^2))./riskAversion;

end
```

APPENDIX B: MATLAB CODE FOR THE PRESSURE VESSEL EXAMPLE

In this appendix, the Matlab code used to solve the pressure vessel design problem is presented. A list of the files along with page numbers are shown in Table B.1. Although the filenames are the same as those used in the Manager's Decision problem (included in Appendix A), there are differences in the content of the files themselves.

TABLE B.1. INVENTORY OF MATLAB FILES USED IN THE PRESSURE VESSEL DESIGN PROBLEM

Filename	Page
normedTwoOnewRiskAversionFunction_new.m	234
normedTwoOnewRiskAversionFunction_OneAnalysis_new.m	235
plotBoundaries_new.m	235
plotBoundaries_OneAnalysis_new.m	238
scanYLdown_new.m	240
scanYLdown_OA_new.m	240
scanYLup_new.m	240
scanYLup_OA_new.m	240
scanYMid_new.m	241
scanYMid_OA_new.m	241
scanYRdown_new.m	241
scanYRdown_OA_new.m	241
scanYRup_new.m	241
scanYRup_OA_new.m	242
TwoOnewRiskAversionFunction.m	242
TwoOnewRiskAversionFunction_OneAnalysis.m	244
utilFunction.m	246

normedTwoOnewRiskAversionFunction_new.m

```
function [SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction(delta_mu,delta_stdev,stdev_eps,Ca,R)

muB = 0;
```

```
% y-axis is delta_stdev, with sigma A and sigma B held constant at 1 at
the
% top and bottom of the plot, respectively.
% if delta_stdev>= 0
%     stdevA = 1;
%     stdevB = stdevA - delta_stdev;
% else
%     stdevB = 1;
%     stdevA = delta_stdev + stdevB;
% end


muA = delta_mu+muB;

% y-axis is sigma A, sigma B is constant at 1
stdevB = 1;
stdevA = delta_stdev;


[SA,SB,AA,AB] =
TwoOnewRiskAversionFunction(muA,stdevA,muB,stdevB,stdev_eps,Ca,R);
```

normedTwoOnewRiskAversionFunction_OneAnalysis_new.m

```
function [SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(delta_mu,delta_stdev,
stdev_eps,Ca,R)

muB = 0;
% y-axis is delta_stdev, with sigma A and sigma B held constant at 1 at
the
% top and bottom of the plot, respectively.
% if delta_stdev>= 0
%     stdevA = 1;
%     stdevB = stdevA - delta_stdev;
% else
%     stdevB = 1;
%     stdevA = delta_stdev + stdevB;
% end


muA = delta_mu+muB;

% y-axis is sigma A, sigma B is constant at 1
stdevB = 1;
stdevA = delta_stdev;


[SA,SB,AA,AB] =
TwoOnewRiskAversionFunction_OneAnalysis(muA,stdevA,muB,stdevB,stdev_eps
,Ca,R);
```

plotBoundaries_new.m

```
function [areas]=plotBoundaries_new(R,stdev_eps,Ca)
```

```
tic

num_steps = 10;
yvect_up = linspace(1,2,num_steps);
yvect_down = linspace(1,0,num_steps);

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_new(0,1,stdev_eps,Ca,R);

if SA > AA
    areas = -1; % Two region plot, rather than 4 region. Vertical
split.
%     find intersection of SA and SB
        y = yvect_up(1);

        SASBu(1) = fzero(@(x) scanYMid_new(x,y,stdev_eps,Ca,R),0);
        SASBd(1) = fzero(@(x)
scanYMid_new(x,y,stdev_eps,Ca,R),SASBu(1));

    for ii = 2:num_steps
        y_up = yvect_up(ii);
        y_down = yvect_down(ii);

        SASBu(ii) = fzero(@(x)
scanYMid_new(x,y_up,stdev_eps,Ca,R),SASBu(ii-1));
        SASBd(ii) = fzero(@(x)
scanYMid_new(x,y_down,stdev_eps,Ca,R),SASBd(ii-1));

    end
figure
    plot(SASBu,yvect_up,'k',SASBd,yvect_down,'k')
    axis([-5 5 0 2])
%     legend('SA=SB','location','Southeast')
    xlabel('\mu_A')
    ylabel('\sigma_A')
    title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
            ', Cost = ',num2str(Ca),...
            ', R = ',num2str(R)]})
   hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-C',num2str(Ca),'-
',datestr(now,'HH.MM.SS'),'.fig'])
    close

else
    [SA1,SB1,AA1,AB1] =
normedTwoOnewRiskAversionFunction_new(5,1,stdev_eps,Ca,R);
    if SA1 > AA1
        areas = 1; % 4 region plot rather than two region
    %     find intersections of SA/AA,SA/AB,SB/AA,SB/AB
        SAAA = zeros(1,num_steps);
        SAAB = SAAA;
        SBAA = SAAA;
        SBAB = SAAA;

        y = yvect_up(1);
            SAAA(1) = fzero(@(x)
scanYRup_new(x,y,stdev_eps,Ca,R),[0,5]);
```

```
            SAAB(1) = fzero(@(x)
scanYRdown_new(x,y,stdev_eps,Ca,R),[SAAA(1)]);
            SBAA(1) = fzero(@(x) scanYLup_new(x,y,stdev_eps,Ca,R),[-
1*SAAA(1)]);
            SBAB(1) = fzero(@(x)
scanYLdown_new(x,y,stdev_eps,Ca,R),[SBAB(1)]);


        %%
        for ii = 2:num_steps
        y_up = yvect_up(ii);
        y_down = yvect_down(ii);

            SAAA(ii) = fzero(@(x)
scanYRup_new(x,y_up,stdev_eps,Ca,R),SAAA(ii-1));
            SAAB(ii) = fzero(@(x)
scanYRdown_new(x,y_down,stdev_eps,Ca,R),SAAB(ii-1));
            SBAA(ii) = fzero(@(x)
scanYLup_new(x,y_up,stdev_eps,Ca,R),SBAA(ii-1));
            SBAB(ii) = fzero(@(x)
scanYLdown_new(x,y_down,stdev_eps,Ca,R),SBAB(ii-1));
        end
        figure

plot(SAAA,yvect_up,'k',SBAA,yvect_up,'k',SAAB,yvect_down,'k',SBAB,yvect
_down,'k',[SAAA(1),SBAA(1)],[1,1],'k')
        axis([-5 5 0 2])
%
legend('SA=AA','SB=AA','SA=AB','SB=AB','AA=AB','location','Southeast')
        xlabel('\mu_A')
        ylabel('\sigma_A')
        title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
                ', Cost = ',num2str(Ca),...
                ', R = ',num2str(R)]})
       hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-
C',num2str(Ca),'-',datestr(now,'HH.MM.SS'),'.fig'])
        close
        right = [[SAAA(linspace(num_steps,1,num_steps))]';SAAB'];
        left = [[SBAA(linspace(num_steps,1,num_steps))]';SBAB'];
        yvector =
[yvect_up([linspace(num_steps,1,num_steps)])';yvect_down'];
        csvwrite(['R',num2str(R),'-eps',num2str(stdev_eps),'-
C',num2str(Ca),'-
',datestr(now,'HH.MM.SS'),'.csv'],[yvector,right,left])
    else
        areas = -2; %Two region plot.  Horizontal split. (Should be the
case for zero cost).
        AAAB = [-5,5];
        y = [1 1];
        figure
        plot(AAAB,y,'k')
        axis([-5 5 0 2])
%         legend('AA=AB','location','Southeast')
        xlabel('\mu_A')
        ylabel('\sigma_A')
        title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
                ', Cost = ',num2str(Ca),...
                ', R = ',num2str(R)]})
       hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-
C',num2str(Ca),'-',datestr(now,'HH.MM.SS'),'.fig'])
        close
    end

end

toc
```

plotBoundaries_OneAnalysis_new.m

```
function [areas]=plotBoundaries_OneAnalysis_new(R,stdev_eps,Ca)
tic

num_steps = 10;
yvect_up = linspace(1,2,num_steps);
yvect_down = linspace(1,0,num_steps);

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(0,1,stdev_eps,Ca,R);

if SA > AA
    areas = -1; % Two region plot, rather than 4 region. Vertical
split.
%     find intersection of SA and SB
        y = yvect_up(1);

        SASBu(1) = fzero(@(x) scanYMid_OA_new(x,y,stdev_eps,Ca,R),0);
        SASBd(1) = fzero(@(x)
scanYMid_OA_new(x,y,stdev_eps,Ca,R),SASBu(1));

    for ii = 2:num_steps
        y_up = yvect_up(ii);
        y_down = yvect_down(ii);

        SASBu(ii) = fzero(@(x)
scanYMid_OA_new(x,y_up,stdev_eps,Ca,R),SASBu(ii-1));
        SASBd(ii) = fzero(@(x)
scanYMid_OA_new(x,y_down,stdev_eps,Ca,R),SASBd(ii-1));

    end
figure
    plot(SASBu,yvect_up,'k',SASBd,yvect_down,'k')
    axis([-5 5 0 2])
%     legend('SA=SB','SA=SB','location','Southeast')
    xlabel('\mu_A')
    ylabel('\sigma_A')
    title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
            ', Cost = ',num2str(Ca),...
            ', R = ',num2str(R)]})
   hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-C',num2str(Ca),'-
',datestr(now,'HH.MM.SS'),'.fig'])
    close

else
```

```
    [SA1,SB1,AA1,AB1] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(5,1,stdev_eps,Ca,R);
    if SA1 >= AA1
        areas = 1; % 4 region plot rather than two region
    %     find intersections of SA/AA,SA/AB,SB/AA,SB/AB
        SAAA = zeros(1,num_steps);
        SAAB = SAAA;
        SBAA = SAAA;
        SBAB = SAAA;

        y = yvect_up(1);
            SAAA(1) = fzero(@(x)
scanYRup_OA_new(x,y,stdev_eps,Ca,R),[0,5]);
            SAAB(1) = fzero(@(x)
scanYRdown_OA_new(x,y,stdev_eps,Ca,R),[SAAA(1)]);
            SBAA(1) = fzero(@(x) scanYLup_OA_new(x,y,stdev_eps,Ca,R),[-
1*SAAA(1)]);
            SBAB(1) = fzero(@(x)
scanYLdown_OA_new(x,y,stdev_eps,Ca,R),[SBAA(1)]);



        for ii = 2:num_steps
        y_up = yvect_up(ii);
        y_down = yvect_down(ii);

            SAAA(ii) = fzero(@(x)
scanYRup_OA_new(x,y_up,stdev_eps,Ca,R),SAAA(ii-1));
            SAAB(ii) = fzero(@(x)
scanYRdown_OA_new(x,y_down,stdev_eps,Ca,R),SAAB(ii-1));
            SBAA(ii) = fzero(@(x)
scanYLup_OA_new(x,y_up,stdev_eps,Ca,R),SBAA(ii-1));
            SBAB(ii) = fzero(@(x)
scanYLdown_OA_new(x,y_down,stdev_eps,Ca,R),SBAB(ii-1));
        end
        figure

plot(SAAA,yvect_up,'k',SBAA,yvect_up,'k',SAAB,yvect_down,'k',SBAB,yvect
_down,'k',[SAAA(1),SBAA(1)],[1,1],'k')
        axis([-5 5 0 2])
%
legend('SA=AA','SB=AA','SA=AB','SB=AB','AA=AB','location','Southeast')
        xlabel('\mu_A')
        ylabel('\sigma_A')
        title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
                ', Cost = ',num2str(Ca),...
                ', R = ',num2str(R)]})
       hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-
C',num2str(Ca),'-',datestr(now,'HH.MM.SS'),'.fig'])
        close
        right = [[SAAA(linspace(num_steps,1,num_steps))]';SAAB'];
        left = [[SBAA(linspace(num_steps,1,num_steps))]';SBAB'];
        yvector =
[yvect_up([linspace(num_steps,1,num_steps)])';yvect_down'];
        csvwrite(['R',num2str(R),'-eps',num2str(stdev_eps),'-
C',num2str(Ca),'-
',datestr(now,'HH.MM.SS'),'.csv'],[yvector,right,left])
```

```
    else
        areas = -2; %Two region plot.  Horizontal split. (Should be the
case for zero cost).
        AAAB = [-5,5];
        y = [1 1];
        figure
        plot(AAAB,y,'k')
        axis([-5 5 0 2])
%         legend('AA=AB','location','Southeast')
        xlabel('\mu_A')
        ylabel('\sigma_A')
        title({[' \sigma_\epsilon = ',num2str(stdev_eps),...
                ', Cost = ',num2str(Ca),...
                ', R = ',num2str(R)]})
       hgsave(['R',num2str(R),'-eps',num2str(stdev_eps),'-
C',num2str(Ca),'-',datestr(now,'HH.MM.SS'),'.fig'])
        close
    end

end

toc
```

scanYLdown_new.m

```
function [diffSBAB] = scanYLdown_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);


diffSBAB = SB-AB;
```

scanYLdown_OA_new.m

```
function [diffSBAB] = scanYLdown_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSBAB = SB-AB;
```

scanYLup_new.m

```
function [diffSBAA] = scanYLup_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);


diffSBAA = SB-AA;
```

scanYLup_OA_new.m

```
function [diffSBAA] = scanYLup_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSBAA = SB-AA;
```

scanYMid_new.m

```
function [diffSASB] = scanYMid_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);


diffSASB = SA-SB;
```

scanYMid_OA_new.m

```
function [diffSASB] = scanYMid_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSASB = SA-SB;
```

scanYRdown_new.m

```
function [diffSAAB] = scanYRdown_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);


diffSAAB = SA-AB;
```

scanYRdown_OA_new.m

```
function [diffSAAB] = scanYRdown_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSAAB = SA-AB;
```

scanYRup_new.m

```
function [diffSAAA] = scanYRup_new(x,y,stdev_eps,Ca,R)

```

```
[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_new(x,y,stdev_eps,Ca,R);


diffSAAA = SA-AA;
```

scanYRup_OA_new.m

```
function [diffSAAA] = scanYRup_OA_new(x,y,stdev_eps,Ca,R)

[SA,SB,AA,AB] =
normedTwoOnewRiskAversionFunction_OneAnalysis_new(x,y,stdev_eps,Ca,R);


diffSAAA = SA-AA;
```

TwoOnewRiskAversionFunction.m

```
function [SA,SB,AA,AB] =
TwoOnewRiskAversionFunction(mu_uA,stdev_uA,mu_uB,stdev_uB,stdev_eps,Ca,
matCosA,matCosB,percent,R,tol)

% New script including risk aversion
% tic
% ----------------------------------------------------------------------
----
% Define risk aversion.
% ----------------------------------------------------------------------
----
% R > 0  risk averse
% R < 0  risk seeking
% R = 0  risk neutral
% R = 0;

% ----------------------------------------------------------------------
----
% Define percentile for determining vessel thickness.
% ----------------------------------------------------------------------
----
percentile = percent;
% ----------------------------------------------------------------------
----
% Define costs
% ----------------------------------------------------------------------
----
costs = [Ca,Ca]; % of analysis
materialCostA = matCosA;
materialCostB = matCosB;
% ----------------------------------------------------------------------
----
% Define distribution parameters
% ----------------------------------------------------------------------
----
% prior distributions on the utility of concepts A and B
% mu_uA =1;
% stdev_uA = 1;;
var_uA = stdev_uA^2;

% mu_uB = 0;
% stdev_uB = 1;
var_uB = stdev_uB^2;

% low quality analysis
% uLQ = uA + eps, where eps ~ N(0,10)
mu_eps = 0;
% stdev_eps = 0.125;
var_eps = stdev_eps^2;

% marginal distribution of low quality analysis
mu_uA_LQ = mu_uA + mu_eps;
var_uA_LQ = var_uA + var_eps;
stdev_uA_LQ = sqrt(var_uA_LQ);

mu_uB_LQ = mu_uB + mu_eps;
var_uB_LQ = var_uB + var_eps;
stdev_uB_LQ = sqrt(var_uB_LQ);
% -----------------------------------------------------------------------
----
% define conditional distributions
% -----------------------------------------------------------------------
----
% p(uI|uLQ) ~ N((uLQ*var_uI + mu_uI*var_eps)/(var_uI + var_eps),
%               (var_uI*var_eps)/(var_uI + var_eps))
mu_AgivenLQ = @ (uLQ) (uLQ*var_uA + mu_uA*var_eps)/(var_uA + var_eps);
var_AgivenLQ = (var_uA*var_eps)/(var_uA + var_eps);
stdev_AgivenLQ = sqrt(var_AgivenLQ);

mu_BgivenLQ = @ (uLQ) (uLQ*var_uB + mu_uB*var_eps)/(var_uB + var_eps);
var_BgivenLQ = (var_uB*var_eps)/(var_uB + var_eps);
stdev_BgivenLQ = sqrt(var_BgivenLQ);
% -----------------------------------------------------------------------
----

% -----------------------------------------------------------------------
----
% Evaluate decision alternatives
% -----------------------------------------------------------------------
----
% tol = 1.e-6;

SA = utilFunction(mu_uA, stdev_uA, R, percentile, materialCostA);
SB = utilFunction(mu_uB, stdev_uB, R, percentile, materialCostB);

AA = quadv(@(a) max(quad(@(b)((max(...

utilFunction(mu_AgivenLQ(a),stdev_AgivenLQ,R,percentile,(materialCostA+
2*Ca)),...

utilFunction(mu_BgivenLQ(b),stdev_BgivenLQ,R,percentile,(materialCostB+
2*Ca))))...
    .*normpdf(b,mu_uB_LQ,stdev_uB_LQ)),...
    mu_uB_LQ-4*stdev_uB_LQ,mu_uB_LQ+4*stdev_uB_LQ, tol), (max(...
utilFunction(mu_AgivenLQ(a),stdev_AgivenLQ,R,percentile,(materialCostA+
Ca)),...

utilFunction(mu_uB,stdev_uB,R,percentile,(materialCostB+Ca)))))...
               .*normpdf(a,mu_uA_LQ,stdev_uA_LQ),...
          mu_uA_LQ-4*stdev_uA_LQ,mu_uA_LQ+4*stdev_uA_LQ, tol);

AB = quadv(@(b) max(quad(@(a)((max(...

utilFunction(mu_AgivenLQ(a),stdev_AgivenLQ,R,percentile,(materialCostA+
2*Ca)),...

utilFunction(mu_BgivenLQ(b),stdev_BgivenLQ,R,percentile,(materialCostB+
2*Ca))))...
    .*normpdf(a,mu_uA_LQ,stdev_uA_LQ)),...
    mu_uA_LQ-4*stdev_uA_LQ,mu_uA_LQ+4*stdev_uA_LQ, tol), (max(...

utilFunction(mu_uA,stdev_uA,R,percentile,(materialCostA+Ca)),...

utilFunction(mu_BgivenLQ(b),stdev_BgivenLQ,R,percentile,(materialCostB+
Ca)))))...
               .*normpdf(b,mu_uB_LQ,stdev_uB_LQ),...
          mu_uB_LQ-4*stdev_uB_LQ,mu_uB_LQ+4*stdev_uB_LQ, tol);

% toc
```

TwoOnewRiskAversionFunction_OneAnalysis.m

```
function [SA,SB,AA,AB] =
TwoOnewRiskAversionFunction_OneAnalysis(mu_uA,stdev_uA,mu_uB,stdev_uB,s
tdev_eps,Ca,R,tol)

% New script including risk aversion
% tic
% ----------------------------------------------------------------------
----
% Define risk aversion.
% ----------------------------------------------------------------------
----
% R > 0  risk averse
% R < 0  risk seeking
% R = 0  risk neutral
% R = 0;

% ----------------------------------------------------------------------
----
% Define costs of analyses
% ----------------------------------------------------------------------
----
costs = [Ca,Ca];
% ----------------------------------------------------------------------
----
% Define distribution parameters
% -----------------------------------------------------------------------
----
% prior distributions on the utility of concepts A and B
% mu_uA =1;
% stdev_uA = 1;;
var_uA = stdev_uA^2;

% mu_uB = 0;
% stdev_uB = 1;
var_uB = stdev_uB^2;

% low quality analysis
% uLQ = uA + eps, where eps ~ N(0,10)
mu_eps = 0;
% stdev_eps = 0.125;
var_eps = stdev_eps^2;

% marginal distribution of low quality analysis
mu_uA_LQ = mu_uA + mu_eps;
var_uA_LQ = var_uA + var_eps;
stdev_uA_LQ = sqrt(var_uA_LQ);

mu_uB_LQ = mu_uB + mu_eps;
var_uB_LQ = var_uB + var_eps;
stdev_uB_LQ = sqrt(var_uB_LQ);
% -----------------------------------------------------------------------
----
% define conditional distributions
% -----------------------------------------------------------------------
----
% p(uI|uLQ) ~ N((uLQ*var_uI + mu_uI*var_eps)/(var_uI + var_eps),
%                (var_uI*var_eps)/(var_uI + var_eps))
mu_AgivenLQ = @ (uLQ) (uLQ*var_uA + mu_uA*var_eps)/(var_uA + var_eps);
var_AgivenLQ = (var_uA*var_eps)/(var_uA + var_eps);
stdev_AgivenLQ = sqrt(var_AgivenLQ);

mu_BgivenLQ = @ (uLQ) (uLQ*var_uB + mu_uB*var_eps)/(var_uB + var_eps);
var_BgivenLQ = (var_uB*var_eps)/(var_uB + var_eps);
stdev_BgivenLQ = sqrt(var_BgivenLQ);
% -----------------------------------------------------------------------
----


% -----------------------------------------------------------------------
----
% Evaluate decision alternatives
% -----------------------------------------------------------------------
----
% tol = 1.e-6;

SA = utilFunction(mu_uA,stdev_uA,R);
SB = utilFunction(mu_uB,stdev_uB,R);

AA = quad(@(a) max(...
                utilFunction(mu_AgivenLQ(a)-
AnalysisCost([1,0],costs),stdev_AgivenLQ,R),...
                utilFunction(mu_uB-
AnalysisCost([1,0],costs),stdev_uB,R))...
```

```
                .*normpdf(a,mu_uA_LQ,stdev_uA_LQ),...
           mu_uA_LQ-10*stdev_uA_LQ,mu_uA_LQ+10*stdev_uA_LQ, tol);

AB = quad(@(b) max(...
                utilFunction(mu_uA-
AnalysisCost([0,1],costs),stdev_uA,R),...
                utilFunction(mu_BgivenLQ(b)-
AnalysisCost([0,1],costs),stdev_BgivenLQ,R))...
                .*normpdf(b,mu_uB_LQ,stdev_uB_LQ),...
           mu_uB_LQ-10*stdev_uB_LQ,mu_uB_LQ+10*stdev_uB_LQ, tol);

% toc
```

utilFunction.m

```
function expectedUtility = utilFunction(StrengthMean, StrengthStdev,
riskAversion, percentile, materialCost)

% Compute the expected utility of a pressure vessel given a normally
distributed material strength
% We assume that the thickness of the pressure vessel is determined
using the Xth percentile strength value. The length and radius are
fixed, and the vessel is a cylinder with hemi-spherical ends.

length = 50; % inches
radius = 4.5; % inches

% The vessel is designed to a pressure of 1000 atm. The thickness is
determined by setting the strength equal to the maximum (hoop) stress.

pressure = 1000; % atm
psiPerAtm = 14.6959488; % psi/atm
percentileStrength = norminv(percentile,StrengthMean,StrengthStdev); %
kpsi
thickness = pressure*radius*psiPerAtm./(percentileStrength); % inches
materialVolume = pi().*(length-2.*thickness).*thickness.*(2*radius-
thickness)+4/3*pi().*(radius^3-(radius-thickness).^3); % cubic inches
cubIncpercubMeter = 1.6387064*10^-5;

% The payoff function includes the sale price per vessel, the material
cost per vessel, and a failure indicator.

pricePerVessel = 200; % dollars
costPerFailure = 300; % dollars

if riskAversion == 0
    expectedUtility = (pricePerVessel -
materialCost.*cubIncpercubMeter.*materialVolume) -
costPerFailure*percentile;
else
% we assume that u = (1 - exp(-a*dollars))/a, where a is riskAversion
    expectedUtility = (percentile)*(1 - exp(-
riskAversion*(pricePerVessel -
materialCost.*cubIncpercubMeter.*materialVolume-
costPerFailure)))./riskAversion + (1-percentile)*(1 - exp(-
riskAversion*(pricePerVessel -
materialCost.*cubIncpercubMeter.*materialVolume)))/riskAversion;
end
```

REFERENCES

Altshuller, G. (2004). *And Suddenly the Inventor Appeared: TRIZ, the Theory of Inventive Problem Solving* (L. Shulyak, Trans.). Worcester, MA: Technical Innovation Center, Inc.

Baldwin, C. Y. and Clarke, K. B. (2000). *Design Rules* (Vol. 1). Cambridge, Massachusetts: The MIT Press.

Barker, J. (2005). "Abstraction and Modeling." In *Beginning Java Objects: From Concepts to Code,* (pp. 3-14) (2nd ed.). Berkeley, CA: Apress.

Bell, D. E., Raiffa, H. and Tversky, A. (Eds.). (1988). *Decision Making: Descriptive, Normative, and Prescriptive Interactions*. New York: Cambridge University Press.

Berger, J. O. (1985). *Statistical Decision Theory and Bayesian Analysis* (2nd ed.). New York: Springer.

Bradley, S. R. and Agogino, A. M. (1994). An Intelligent Real Time Design Methodology for Component Selection: An Approach to Managing Uncertainty. *ASME Journal of Mechanical Design*, 116(4), 980-8.

Braha, D. and Reich, Y. (2003). Topological structures for modeling engineering design processes. *Research in Engineering Design*, 14, 185-99.

Bras, B. and Mistree, F. (1991). Designing Design Processes in Decision-based Concurrent Engineering. *SAE Transactions, Journal of Materials and Manufacturing*, 100, 451-8.

Bras, B. A. and Mistree, F. (1993). Compromise Decision Support Problem for Axiomatic and Robust Design. *Advances in Design Automation*, 65, 359-69.

Chen, W., Allen, J. K., Mavris, D. and Mistree, F. (1996a). A Concept Exploration Method for Determining Robust Top-Level Specifications. *Engineering Optimization*, 26(2), 137-58.

Chen, W., Allen, J. K., Tsui, K.-L. and Mistree, F. (1996b). A Procedure for Robust Design: Minimizing Variations Caused by Noise Factors and Control Factors. *ASME Journal of Mechanical Design*, 118(4), 478-85.

Clemen, R. T. (1996). *Making Hard Decisions: An Introduction to Decision Analysis* (2nd ed.). Pacific Grove, CA: Duxbury Press.

Dasgupta, S. (1992). "Herbert Simon's 'Science of Design': Two Decades Later." First International Conference on Intelligent Systems Engineering, Edinburgh, UK. 171-8.

de Finetti, B. (1964). "Foresight. Its Logical Laws, Its Subjective Sources (translated)." In *Studies in Subjective Probability,* H. E. Kyburg and H. E. Smokler Eds.). New York: Wiley.

--- (1974). *Theory of Probability Volume 1: A Critical Introductory Treatment* (Vol. 1). New York: Wiley.

Finger, S. and Dixon, J. R. (1989a). A Review of Research in Mechanical Engineering Design. Part I: Descriptive, Prescriptive, and Computer-based Models of Design Processes. *Research in Engineering Design*, 1(1), 51-67.

--- (1989b). A Review of Research in Mechanical Engineering Design. Part II: Representations, Analysis and Design for the Life Cycle. *Research in Engineering Design*, 1(2), 121-37.

Fishman, G. S. (1996). *Monte Carlo: Concepts, Algorithms, and Applications*. New York: Springer.

Gander, W. and Gautschi, W. (2000). Adaptive Quadrature - Revisited. *Bit Numerical Mathematics*, 40(1), 84-101(18).

Gero, J. S. (1990). Design Prototypes: A Knowledge Representation Schema for Design. *AI Magazine*, 11(4), 26-36.

Ghanem, R. and Spanos, P. D. (1990). Polynomial Chaos in Stochastic Finite Elements. *ASME Journal of Applied Mechanics*, 57(1), 197-202.

Hatchuel, A. and Weil, B. (2009). C-K design theory: an advanced formulation. *Research in Engineering Design*, 19, 181-92.

Hazelrigg, G. A. (1996). *Systems Engineering: An Approach to Information-based Design*. Upper Saddle River, NJ: Prentice-Hall.

--- (1998). A Framework for Decision-Based Engineering Design. *ASME Journal of Mechanical Design*, 120, 653-8.

--- (2003). Validation of Engineering Design Alternative Selection Methods. *Engineering Optimization*, 35(2), 103-20.

Herstein, I. and Milnor, J. (1953). An Axiomatic Approach to Measurable Utility. *Econometrica*, 21(2), 291-7.

Howard, R. A. (1965). Bayesian Decision Models for Systems Engineering. *IEEE Transactions on Systems Science and Cybernetics*, SSC-1(1), 36-40.

--- (1966a). "Decision Analysis: Applied Decision Theory." Proceedings of the Fourth International Conference on Operational Research, New York (D. B. Hertz and J. Melese, Eds.). John Wiley, 55-71.

--- (1966b). Information Value Theory. *IEEE Transactions on Systems Science and Cybernetics*, SCC-2(1), 22-6.

--- (1968). The Foundations of Decision Analysis. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(3), 211-9.

Kolmogorov, A. N. (1956). *Foundations of the Theory of Probability* (2nd English ed.). New York: Chelsea Publishing Company. (Original work published in Grundbegriffe der Wahrscheinlichkeitrechnung).

Krishnamurty, S. (2006). "Normative Decision Analysis in Engineering Design." In *Decision Making in Engineering Design,* K. E. Lewis, W. Chen and L. C. Schmidt Eds.) (pp. 21-33). New York: ASME Press.

Kumar, D. K. D., Chen, W. and Kim, H. M. (2006). "Multilevel Optimization for Enterprise-Driven Decision-Based Product Design." In *Decision Making in Engineering Design,* K. Lewis, W. Chen and L. C. Schmidt Eds.) (pp. 203-14). New York: ASME Press.

Lawrence, D. B. (1999). *The Economic Value of Information*. New York: Springer.

Lee, B. D. and Paredis, C. J. J. (2010a).

Lee, B. D., Thompson, S. C. and Paredis, C. J. J. (2010b). "A Review of Methods for Design under Uncertainty from the Perspective of Utility Theory." ASME IDETC/CIE 2010, Montreal, Canada. American Society of Mechanical Engineers, Paper No. DETC2010-28721.

Lewis, K., Chen, W. and Schmidt, L. C. (Eds.). (2006a). *Decision Making in Engineering Design*. New York: ASME Press.

Lewis, K. E., Chen, W. and Schmidt, L. C. (Eds.). (2006b). *Decision Making in Engineering Design*. New York: ASME.

Ling, J. M., Aughenbaugh, J. M. and Paredis, C. J. J. (2005). "The Use of Imprecise Probabilities to Estimate the Value of Information in Design." Proceedings of the 2005 ASME International Mechanical Engineering Congress and Exposition, Orlando, FL. Paper No. IMECE2005-81181.

--- (2006). Managing the Collection of Information under Uncertainty using Information Economics. *ASME Journal of Mechanical Design*, 128(4), 980-90.

Luce, R. D. and Raiffa, H. (1957). *Games and Decisions*. New York: Wiley.

Marschak, J. (1950). Rational Behavior, Uncertain Prospects, and Measurable Utility. *Econometrica*, 18(2), 111-41.

Marston, M., Allen, J. K. and Mistree, F. (2000). The Decision Support Problem Technique: Integrating Descriptive and Normative Approaches in Decision Based Design. *Engineering Valuation & Cost Analysis, special edition on "Decision-Based Design: Status & Promise"*, 3, 107-29.

Marston, M. and Mistree, F. (1997). "A Decision-Based Foundation for Systems Design: A Conceptual Exposition." CIRP 1997 International Design Seminar Proceedings on Multimedia Technologies for Collaborative Design and Manufacturing, University of Southern California, Los Angeles, October 9-10, 1997. 1-11.

Matheson, J. E. (1968). The Economic Value of Analysis and Computation. *IEEE Transactions on Systems Science and Cybernetics*, SSC-4(3), 325-32.

Mistree, F., Smith, W. F., Bras, B., Allen, J. and Muster, D. (1990). Decision-based Design: A Contemporary Paradigm for Ship Design. *Transactions, Society of Naval Architects and Marine Engineers*, 98, 565-97.

Pahl, G. and Beitz, W. (1996). *Engineering Design: A Systematic Approach* (2nd ed.) (K. Wallace, L. Blessing and F. Baurt, Trans.). London: Springer-Verlag.

Panchal, J. H., Paredis, C. J. J., Allen, J. K. and Mistree, F. (2008). A Value-of-Information Based Approach to Simulation Model Refinement. *Engineering Optimization*, 40(3), 223-51.

Phadke, M. S. (1989). *Quality Engineering Using Robust Design*. Englewood Cliffs, NJ: Prentice Hall.

Radhakrishnan, R. and McAdams, D. A. (2005). A Methodology for Model Selection in Engineering Design. *ASME Journal of Mechanical Design*, 127(3), 378-87.

Rao, S. S. (1992). *Reliability-Based Design*. New York: McGraw-Hill.

Reich, Y. (1995). A Critical Review of General Design Theory. *Research in Engineering Design*, 7(1), 1-18.

Schlosser, J. and Paredis, C. J. J. (2007). "Managing Multiple Sources of Epistemic Uncertainty in Engineering Decision Making." Proceedings of the SAE 2007 World Congress, Detroit, MI, April 16-19.

Scott, M. J. and Antonsson, E. K. (1999). Arrow's Theorem and Engineering Design Decision Making. *Research in Engineering Design*, 11, 218-28.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press.

Simon, H. A. (1996). *The Sciences of the Artificial* (3rd ed.). Cambridge, MA: MIT Press.

--- (1997). *Models of Rationality, Vol. 3: Empirically Grounded Economic Reason* (Vol. 3). Cambridge, Massachusetts: MIT Press.

Suh, N. P. (2001). *Axiomatic Design: Advances and Applications*. New York: Oxford University Press.

SysML (2006).

Taguchi, G. (1986). *Introduction to Quality Engineering*. Tokyo: Asian Productivity Organization.

Taguchi, G. and Clausing, D. (1990). Robust Quality. *Harvard Business Review*, Jan/Feb, 65-75.

Thompson, S. C. and Paredis, C. J. J. (in press). An Investigation Into the Decision Analysis of Design Process Decisions. *ASME Journal of Mechanical Design*.

Thurston, D. L. (1991). A Formal Method for Subjective Design Evaluation with Multiple Attributes. *Research in Engineering Design*, 3(2), 105-22.

--- (2001). Real and Misconceived Limitations to Decision Based Design With Utility Analysis. *ASME Journal of Mechanical Design*, 123(2), 176-82.

Thurston, D. L., Carnahan, J. V. and Liu, T. (1994). Optimization of Design Utility. *ASME Journal of Mechanical Design*, 116, 801-8.

Tomiyama, T. and Yoshikawa, H. (1987). Extended General Design Theory. *Design theory for CAD; proceedings of the IFIP WG 5.2 Working Conference on Design Theory for CAD, Tokyo, Japan, 1-3 October, 1985*, 95-124.

Tribus, M. (1969). *Rational Descriptions, Decisions and Designs*. Elmsford, New York: Pergamon Press, Inc.

Tumer, I. Y., Barrientos, F. A. and Mehr, A. F. (2005). "Towards Risk Based Design (RBD) of Space Exploration Missions: A Review of RBD Practice and Research Trends at NASA." ASME 2005 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, Long Beach, California, USA. ASME, Paper No. DETC2005-85100.

Umeda, Y., Takeda, H., Tomiyama, T. and Yoshikawa, H. (1990). Function, behaviour, and structure. *Applications of Artificial Intelligence in Engineering*, 1, 177-94.

von Neumann, J. and Morgenstern, O. (1953). *Theory of Games and Economic Behavior* (3rd ed.). Princeton, NJ: Princeton University Press.

Walley, J. (1991). *Statistical Reasoning with Imprecise Probabilities*. New York: Chapman and Hall.

Wood, W. H. and Agogino, A. M. (2005). Decision-Based Conceptual Design: Modeling and Navigating Heterogeneous Design Spaces. *ASME Journal of Mechanical Design*, 127(1), 2-11.

Yoshikawa, H. (1981). "General Design Theory and a CAD System." Man-Machine Communication in CAD/CAM, Proceedings of the IFIP WG 5.2/5.3 Working Conference 1980, Tokyo (E. Sata and E. Warman, Eds.). North-Holland, Amsterdam, 35-58.