CONCEPTUAL DESIGN USING PROBABILISTIC INTERVAL CONSTRAINT SATISFACTION

by

NATHAN LOEWEN

B.A.Sc., University of British Columbia, 2001
M.A.Sc., University of British Columbia, 2003

A THESIS SUBMITTED IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

THE FACULTY OF GRADUATE STUDIES

(Civil Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

November 2009

© Nathan Loewen, 2009

Abstract

Dealing with uncertainty is one of the primary challenges engineers face in the conceptual design phase of a project. Engineers must make decisions with regards to uncertain design parameters that influence performance and cost. It is well recognized that decisions made in the concept phase of a project have far greater performance and economic impacts than decisions made during the detailed phase, yet most engineering analysis tools have limited capabilities to carry out risk analysis, sensitivity analysis, optimization, and design space mapping.

This thesis proposes a methodology for coupling probabilistic techniques with recent advances in semi-quantitative analysis, specifically the application of interval analysis to numerical constraint satisfaction problems. The result is a method of analysis referred to as Probabilistic Constraint Satisfaction that can be used to analyze problems with probabilistic input and generate a probabilistic design space as an output. The methodology involves subdivision of the design space to a requested resolution and then testing the consistency of the constraint satisfaction problem at each subdivided location. This approach is very robust and is capable of solving a wide range of problems regardless of linearity or the availability of an explicit solution. Probabilistic data is integrated through the use of a solver which determines the valid interval for each of the probabilistic variables at a given point in the design space. The valid intervals are subsequently used to determine the probability of a feasible solution occurring at that point in the design space.

The characteristics and capabilities of this methodology are illustrated through several engineering examples and a case study that involves the conceptual design of a novel radio antenna concept. The selected case study has several features that are common to conceptual design problems for novel and complex projects including design parameters with continuous domains, trade-offs between performance targets and cost, uncertain costing data, and limited existing experience to base the design on.

Table of Contents

Abstract ii
Table of Contents iii
List of Tables vi
List of Figures vii
Acknowledgments x
1 INTRODUCTION 1
2 LITERATURE REVIEW 4
2.1 Uncertainty, Risk, and Conceptual Design 4
2.2 Existing Tools & Techniques 6
2.2.1 Computational Methods 6
2.2.1.1 Optimization 6
2.2.1.2 Finite Element Analysis 7
2.2.2 Simulation 7
2.2.3 Reliability Methods 8
2.2.4 Artificial Intelligence 10
2.2.4.1 Expert Systems 10
2.2.4.2 Fuzzy Logic 10
2.2.4.3 Qualitative Reasoning 11
2.2.5 Interval Arithmetic 11
3 BACKGROUND 12
3.1 Qualitative Reasoning 12
3.2 Constraint Satisfaction Problems 14
3.2.1 Backtracking 14
3.2.2 Consistency Techniques 14
3.3 Interval Analysis 15
3.4 Interval Constraint Satisfaction Problems 16
3.4.1 Developments in Research and Applications 17
3.4.2 Solution Visualization 18
4 THEORY 21

4.1 Overview of Methodology 21
4.2 Interval Narrowing Solution Method 25
4.2.1 Problems with a Single Probabilistic Variable 25
4.2.2 Problems with Multiple Probabilistic Variables 29
4.3 Grid-Search Solution Method 35
4.4 Comparison of Solution Methods 36
4.4.1 Hybrid Method 44
4.4.2 Comparison to Monte Carlo 45
5 APPLICATION 47
5.1 Software Platform 47
5.2 Application Architecture 48
5.2.1 Input Module 50
5.2.1.1 Input Parameters 50
5.2.1.2 Constraints 51
5.2.1.3 Output Parameters 51
5.2.2 Solution Module 51
5.2.2.1 Probabilistic Analysis Module 52
5.2.2.2 Output Matrix 52
5.2.3 Post-Processing Module 53
5.2.3.1 Plotting Functions 54
5.3 Example Problems 54
5.3.1 Example 1: 2D Probabilistic Solution Space 54
5.3.2 Example 2: 3D Probabilistic Solution Space 56
5.3.3 Example 3: 3D Isosurface Plots 58
5.3.4 Example 4: Reliability Solution Space 61
5.3.5 Example 5: Multi-Span Beam 64
5.3.5.1 Problem Description 64
5.3.5.2 Calculations 66
5.3.5.3 Analysis Methodology 71
5.3.5.4 Results 73
6 CASE STUDY 80

6.1 Introduction 80
6.1.1 Reflector Panel Design 81
6.1.2 Receiver Design 85
6.2 Problem Description 87
6.2.1 Panel Cost 88
6.2.2 Actuator Cost 89
6.2.3 Receiver Cost 90
6.2.4 Total Cost 91
6.2.5 Probabilistic Parameters 92
6.3 Analysis and Results 94
6.4 Case Study Summary 98
7 CONCLUSIONS 99
8 References 102
Appendix A: Source Code for Example Problem 2 106
Appendix B: Source Code for Convergence Example 109

List of Tables

Table 1: Probability over all *z* for *y*=[0,1] with and with-out depth-wise subdivision 34

Table 2: Comparison of solution methods for determining valid ranges of probabilistic variables 37

Table 3: Probability of solution for *n* subdivisions..... 43

List of Figures

Figure 1: Timeline of costs committed and costs incurred for a typical design-build project 2
Figure 2: PDF of parameter *b* for example problem 26
Figure 3: Probabilistic solution space for example problem $y <= x+b$ 26
Figure 4: Probabilistic solution space for example problem $y <= x+b$ approximated via interval constraint satisfaction with 1x1 cells 27
Figure 5: Probabilistic solution space for example problem $y <= x+b$ approximated via interval constraint satisfaction with 0.5x0.5 cells 28
Figure 6: PDF of parameter *r* for example problem 30
Figure 7: Values of narrowed intervals of *r* and *z* at each cell for example problem $x^2 + y^2 + z^2 <= r^2$ 30
Figure 8: Probabilistic solution space for example problem $x^2 + y^2 + z^2 <= r^2$ considering only a single probabilistic variable *r* 31
Figure 9: PDFs of parameters *r* and *z* for example problem 32
Figure 10: Probabilistic solution space for example problem $x^2 + y^2 + z^2 <= r^2$ showing the individual and combined probabilities of the valid intervals of *r* and *z* 32
Figure 11: Probabilistic solution space for example problem $x^2 + y^2 + z^2 <= r^2$ with probabilistic variable *z* subdivided and plotted on the vertical axis over $y = [0,1]$ .. 34
Figure 12: PDF of variable *x* with exact valid interval plotted 39
Figure 13: Valid interval approximated by interval narrowing 39
Figure 14: Valid interval approximated by grid search with dx=1, and consistency checking at interval midpoint 40
Figure 15: PDFs of variables *x* and *y* 41
Figure 16: Valid intervals of *x* and *y* after interval narrowing 41
Figure 17: Valid interval of *y* for test interval of x=[1:1.5] 42
Figure 18: Valid interval of y for test interval of x=[1.5:2] 42
Figure 19: Architecture of software application 49
Figure 20: Probabilistic solution space for example 1 55

Figure 21: Probabilistic solution space for example 1 with bounding solution space lines overlain. 56
Figure 22: Solution space to example 2, hollow sphere 57
Figure 23: Probabilistic solution space to example 2 for different resolutions 58
Figure 24: Probabilistic solution space for example 3 59
Figure 25: Isosurfaces corresponding to solutions with 90%, 50%, and 10% probability of success; note that the contour colour in this case is equal to the z-value 60
Figure 26: Probabilistic solution space for example 4 62
Figure 27: Probabilistic solution space for example 4, with lower contour value cut-off to isolate solution space with reliability of $\Phi(\beta = 1) = 0.841$, $\Phi(\beta = 2) = 0.977$ and $\Phi(\beta = 3) = 0.9987$ 63
Figure 28: Layout of example problem 5 64
Figure 29: Section properties used in example problem 5 65
Figure 30: Sample calculations using spreadsheet calculation method 71
Figure 31: Probability of meeting weight budget with target weight = 30 tonnes 73
Figure 32: Probability of meeting weight budget with target weight = 30 tonnes with results plotted at resolutions of 1x1, 2x2, 5x5, 10x10, 15x15, and 20x20. 74
Figure 33: Maximum (top) and minimum (bottom) mass distributions 76
Figure 34: Probability-weighted mass distribution 77
Figure 35: Probability-weighted mass distribution for flange mass (top), web mass (center), and column mass (bottom) 78
Figure 36: Maximum calculated load capacity [kN/m] for designs which met all constraints, including target weight = 30 tonnes 79
Figure 37: LAR concept overview showing ground based adaptive reflector structure and airborne receiver 81
Figure 38: Prototype reflector structure with two triangular reflector panels (silver, 10m side length) and one secondary supporting structure (red, 20m side length)..... 82
Figure 39: Prototype hydraulically driven actuator with 6m stroke 83
Figure 40: Prototype reflector structure supported on three actuators 84
Figure 41: A 350m diameter reflector structure tiled with 20m triangular support structures 85

Figure 42: Prototype aerostat system shown in launching configuration 86
Figure 43: Prototype aerostat system airborne with one of the six control tether lines shown in the foreground 86
Figure 44: Sample spreadsheet calculations for case study problem..... 92
Figure 45: Probabilistic solution space with contour representing the probability of meeting the constraint $C_{TOT}<C_{TARGET}$=\$1100/m^2 95
Figure 46: Solution space surface plots for 10th, 25th, 50th, 75th and 90th percentile surfaces with contour plotting the allowable deflection ⊿ in mm. 96
Figure 47: Probability-weighted cost breakdown in \$/m^2 showing total cost (top left), panel cost (top right), actuator cost (bottom left) and receiver cost (bottom right).. 97
Figure 48: Solution space to example problem 2..... 106

Acknowledgments

I wish to express my gratitude to my advisor Dr. Siegfried F. Stiemer for encouraging me to continue my studies through a Ph.D. degree, and for providing advice and direction during my research. I also wish to thank Mr. David Halliday, Vice President of Empire Dynamic Structures Ltd for allowing me the opportunity to pursue my studies during my employment there.

The generous financial assistance of the Natural Sciences and Engineering Research Council of Canada is greatly appreciated.

I am also very thankful to my wife, Tara, and my family for all of their support and encouragement during my studies.

1 INTRODUCTION

Dealing with uncertainty is one of the primary challenges engineers face in the design phase of a project. Typical areas where engineers encounter uncertainty are loading conditions, material properties, and expected cost of a project. Some of the uncertain parameters may have sufficient databases available to develop statistical distributions, whereas for other parameters engineers must rely on judgment and experience. Existing tools for dealing with uncertainty typically require statistical distributions for uncertain parameters and are limited in the types of equation systems they are capable of solving, and are not often implemented by practicing engineers.

Further sources of vagueness are encountered during the conceptual design phase of projects when many design parameters impacting design and cost are poorly defined or changing. Engineers must make decisions with regard to these design parameters that control risk related to safety, performance, and cost. It is well recognized that decisions made in the concept phase of a project have far greater performance and economic impacts than decisions made during the detailed phase, as illustrated in Figure 1. However, this is disproportionate to the focus of most engineering analysis tools which require rigid input and output data structures, precise design parameters, and have limited capabilities to carry out risk analysis, sensitivity analysis, optimization, and map the available design space.

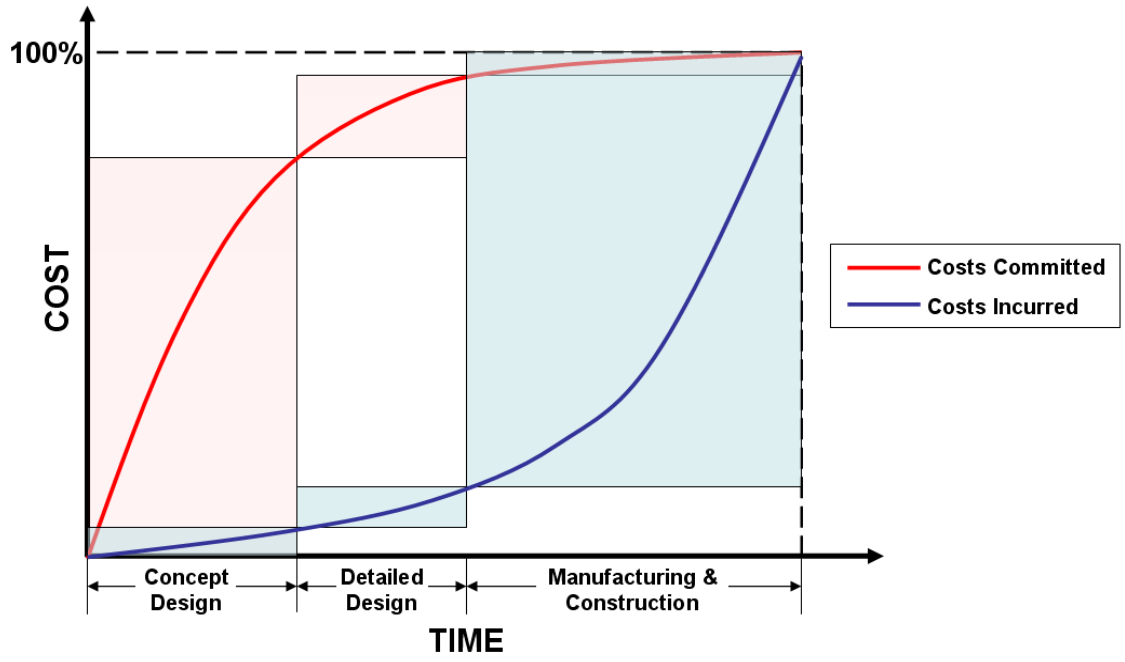

Figure 1: Timeline of costs committed and costs incurred for a typical design-build project

The goal of the research presented herein is to develop a tool to assist in high-level computations where uncertainty is present; this is a scenario typically encountered during engineering conceptual design. The tool must be general enough to handle a wide range of problems, have a flexible user interface, be computationally robust in solving both linear and non-linear systems of equations, and handle uncertainties in a variety of forms. This paper describes how recent advances in semi-quantitative analysis can be coupled with probabilistic data to produce such a tool.

This thesis begins with a review of common engineering methods used in conceptual design and briefly describes their strengths and weaknesses. A background is then given on the development of semi-quantitative methods, particularly the use of solving numeric constraint satisfaction problems with interval parameters.

The original work of this thesis combining semi-quantitative methods with probabilistic analysis is then presented. The method of analysis is referred to as Probabilistic Constraint Satisfaction and can be used to analyze problems with probabilistic input and

generates a probabilistic design space as an output. The methodology involves subdivision of the design space to a requested resolution and then tests the consistency of the constraint satisfaction problem at each subdivided location. This approach is very robust and is capable of solving a wide range of problems regardless of linearity or the availability of an explicit solution. Probabilistic data is integrated through the use of a solver which determines the valid interval for each of the probabilistic variables at a given point in the design space. Different solution methods for determining the valid intervals are developed and discussed including interval narrowing techniques, grid-search techniques and hybrid techniques. The valid intervals are subsequently used to determine the probability of a feasible solution occurring at that point in the design space.

The software implementation of the methodology is then outlined, which consists of a series of input, solution and output modules that can be reconfigured to apply to a wide range of problems. Several simple example problems are presented to demonstrate the range of the methodology capabilities, including various data that can be extracted from the solution process and also various post-processing methods that are available. An elementary structural engineering problem is also presented to demonstrate the basic application and output data interpretation for engineering applications.

A case study is then presented that is based on an industrial research project and involves the conceptual design of a novel concept for a very large array of radio antennae. The case study is representative of conceptual design problems with large continuous design parameter domains, uncertain design parameters including performance targets and costing data, and very limited existing experience from similar projects to draw from. The application of the methodology developed in this thesis is compared with more standard existing methodologies.

The thesis concludes with a discussion of the research results and prospect for future work in this area.

2 LITERATURE REVIEW

The following section gives an overview of the general process of conceptual design and how uncertainty and risk are defined and dealt with in typical engineering scenarios. A brief review of some of the existing tools and techniques for analyzing these types of problems is then given, and their advantages and shortcomings are discussed.

2.1 Uncertainty, Risk, and Conceptual Design

Conventionally, engineers have carried out design under conditions of uncertainty and risk using a combination of judgment based on experience, code-based techniques, rules of thumb, and empirical data. When these do not apply due to either a very complex or unique design, additional tools are implemented.

Uncertainty and risk are two closely related factors that must be addressed in conceptual design. Hubbard (2007) distinguishes between uncertainty and risk as follows:

- Uncertainty is a state of having limited knowledge where it is impossible to exactly describe the existing state or future outcome, and there is more than one possible outcome.
- Risk is a state of uncertainty where some possible outcomes have an undesired effect or significant loss.

Ayyub and Chao (1998) identify uncertainties as stemming from either ambiguity or vagueness. Uncertainties stemming from ambiguity are due to randomness or lack of knowledge, and are typically analyzed with probability theory. Uncertainties stemming from vagueness are due to human factors and inappropriate problem definitions, and are typically analyzed with fuzzy set theory (described below).

Estimating statistical data associated with uncertain parameters is handled in several ways depending on the data available. If sufficient data sets are available then well defined statistical methods are available to define statistical parameters describing the

probability distributions of the data. However, in many engineering applications reliable data sets are not available or must be extrapolated to new domains. Examples of this include unit cost data applied to new projects, cost escalation factors, or the performance relationship between two design parameters. Tversky and Kahneman (1974) describe ways in which people commonly estimate probabilities, and identify the techniques and biases they intuitively apply. Three heuristics are described that people use to estimate probabilities and their associated problems:

- Representativeness: describes how people assess probability of events or classify events based on comparison to existing evidence
- Availability: describes how people assess probability of events based on recalling occurrences of similar events
- Adjustment and anchoring: describes how initial biases in the initial problem formulation influence a persons assessment in the probability of an event

Conceptual design involving uncertainty is typically carried out by senior engineers or architects based largely on experience and judgment developed through a long history of similar projects. Research in the past years has investigated the various methods that design experts employ when carrying out conceptual design activities. Since most of the design methods are informal and are learned through experience and not taught, they are often difficult to identify and analyze. To address this systems have been developed to formalize the analysis of design protocols to gain an understanding of how designers design (Gero and McNeill 1998), how drawings are used in the design process (Purcell and Gero 1998), and how sketching is used by expert architects during early conceptual design (Bilda et al 2006).

One of the challenges in conceptual design is transferring the acquired experience of design experts to less experienced designers. Recent research is making attempts to address this issue. For example, Cerulli et al (2001) developed a system for storing knowledge of design process and decisions in an object-oriented CAD system with a goal of encouraging a collaborative process in the design phase without inhibiting design creativity, and the developed system was subject to successful field trials.

2.2 Existing Tools & Techniques

An ideal analytical tool to assist in conceptual design would have the following traits:

- The user interface would allow simple problem formulation that could be configured to a wide range of problems.
- The output would identify optimal solutions based on performance functions, as well as identify the available design space for alternate solutions
- The tool would be capable of handling uncertain parameters, and indicate sensitivity of the solution to their values
- The computational platform would be capable of solving any system of equations regardless of linearity

A brief review existing tools and techniques applied to conceptual design problems is given below, and their strengths and weaknesses with regards to the above list are noted.

2.2.1 Computational Methods

Engineering problems can typically be represented by a system of equations that may be linear or non-linear. Linear algebra is a very well developed branch of mathematics concerned with solving systems of linear equations and can be expanded to systems of non-linear equations using algebraic or iterative algorithms. These methods are typically used to solve problems that have a single solution, i.e. they are neither over-constrained nor under-constrained, and have deterministic input data. Since engineering design problems typically involve solution *spaces* (i.e. a range of solutions available within certain limits) and probabilistic data, basic computational methods must be coupled with other techniques to fully analyze these types of problems.

2.2.1.1 Optimization

Optimization procedures can be used to search the solution space of a system of equations for the maximum or minimum value of a user-defined objective function.

Linear programming is a technique used to optimize problems where all of the constraints and the objective function are linear.

Stochastic methods provide a more general optimization procedure that is not limited to linear problems. These methods implement random starting points in the design space and then iteratively search for local maxima or minima to the objective function. The results of many random starting points are compared to determine the global maxima or minima. These types of optimization procedures have proved to be useful in both conceptual and final design phases, but they are not capable of implementing probabilistic data, and are therefore more useful for problems with well-defined constraints. An additional limitation is that they only give the analyst discrete snapshots of the solution, and don't provide an overall representation of the available solution space.

2.2.1.2 Finite Element Analysis

Finite element analysis has traditionally been used to provide detailed analytical models of structures once initial design concepts are well developed. Advances in commercial finite element analysis packages have made parametric modeling and optimization more readily available to practicing designers and when used correctly can provide useful design aids during conceptual design.

Other recent research has incorporated sensitivity analysis into finite element analysis (Scott and Haukaas 2008). Sensitivity analysis is generally used to determine the sensitivity of a design outcome to uncertain parameters, thus providing a measure of design risk. These advances promote the integration of finite element analysis, reliability analysis, and optimization.

2.2.2 Simulation

Simulation techniques can be used to analyze systems of equations. The most common method of implementing probabilistic data in simulation is the use of Monte Carlo

techniques due to its generality – it simply uses brute computational force to build up a probabilistic solution space. This can lead to long simulation times, but this is becoming less of a problem as computational power continues to increase with time. While Monte Carlo remains a powerful tool for generating probabilistic solution spaces, it has some drawbacks:

- The procedure is totally naïve; it randomly searches the entire solution space even including regions that contain no solution. This can cause major inefficiencies in solution spaces that sparsely populate the overall design space.
- It needs a very large number of iterations to find design points with a very high or very low probability of occurrence, which are often of interest in engineering problems which require a very high degree of reliability.
- It does not reliably find zero-width features, e.g. it cannot find lines or points in a 2-dimensional solution space since it is unlikely for the initial random points to be located on the zero-width lines to a very high precision.

2.2.3 Reliability Methods

Reliability methods provide the basis for modern design codes, in particular in load and resistance factor design (LRFD) where reliability methods are used to calibrate the various factors. Both the load conditions and the material properties are not known with sufficient precision to be expressed by deterministic values. By using reliability methods, an overall specified probabilistic level of safety can be achieved by using probability distributions to calculate the corresponding load factors to apply.

A widely used method is the first order reliability method (FORM) procedure. The basic analysis procedure is as follows:

- Define a performance function $g(\mathbf{X})$, where $g<=0$ indicates failure and $g>0$ indicates success. Specify all random variables $\mathbf{X}$ and assign probability distributions and correlation coefficients.
- Convert the problem into standard normal space (i.e. convert from **X**-space to **Y**-space such that all entries of **Y** have a mean of zero and unit variance)

- Determine the point on the limit state surface g=0 in standard normal space that has the highest probability density (this is the point nearest the origin, referred to as the design point **Y***)
- Approximate the limit state surface as a plane through the design point, and integrate the probability distribution of the g-function in the region of failure to estimate the probability of failure. Higher-order approximations can be used through this point; approximating the limit state surface as a quadratic is referred to as Second Order Reliability Method (SORM).

FORM analysis can be applied to many general engineering problems, and is gaining momentum as the basis of performance-based design. Some drawbacks with its use as a conceptual design tool are as follows:

- Highly non-linear limit state functions may not be well approximated by either first-order or second-order approximations. Additionally, it is often difficult to determine the design point about which to linearize the limit state function, and much effort has been devoted to developing computational methods to solve this problem.
- The output estimates the reliability at a single design point. Multiple analyses are required to build up a probabilistic solution space.
- Many practical engineering problems may lack probabilistic data with the distribution forms (e.g. normal, lognormal) required to carry out probabilistic calculations.

Methods have been developed that utilize sensitivity and importance measures that are available from FORM analysis to rank design parameters based on their cost-effectiveness in improving the reliability of the design (Haukaas 2006). The methodology distinguishes between design parameters and tolerance parameters, which enables assessment of improving reliability by allocating resources to allow tighter tolerances on certain parameters where this proves to be cost effective.

2.2.4 Artificial Intelligence

The field of artificial intelligence (AI) in general terms seeks to use non-human means (i.e. computers) to make intelligent decisions. Several fields of AI have been applied to engineering problems including expert systems, fuzzy logic, neural networks, image recognition, and qualitative physics. Recent applications of artificial intelligence to engineering problems have been made in the geotechnical field where problems are often characterized by uncertain or incomplete information. Cheng et al (2008) applied an Evolutionary Fuzzy Neural Inference Model (EFNIM) to geotechnical engineering problems, which combined a genetic algorithm, fuzzy logic, and neural network. Another engineering application of artificial neural networks has been in the field of bridge risk assessment (Elhag et al 2007).

2.2.4.1 Expert Systems

Expert systems are essentially computer programs that attempt to mimic the decision behaviour of a human expert in a certain field. They contain a knowledge base of rules that are able to infer conclusions based on user-input. Expert systems are not practical as concept design tools due to their inflexibility since they require the development of a domain-specific knowledge base and inference rules that are applied to specific problems.

2.2.4.2 Fuzzy Logic

Fuzzy logic is an approach to logical reasoning that assigns a degree of truthfulness to statements. Traditional Boolean logic applies either “true” or “false” values to statements (typically represented by a value of either 0 or 1), whereas fuzzy logic allows the degree of truthfulness of a statement to be represented by a continuum of truthfulness values between 0 and 1. The intent of this is to allow the reasoning process to more closely mimic reality, since in real-world problems the absolute truth or falsity of a statement is rarely known. Fuzzy logic can be used to expand the capabilities of expert systems to indicate degrees of uncertainty, but cannot be applied to the general continuous numerical domains typically encountered in many engineering problems.

2.2.4.3 Qualitative Reasoning

Another lesser-known field of artificial intelligence known as qualitative reasoning shows promise in application to engineering problems. When combined with constraint satisfaction techniques, qualitative reasoning can be used to carry out numerical calculations; this is described in further detail in section 3.

2.2.5 Interval Arithmetic

The representation of variables as intervals and their subsequent manipulation is a well used and understood methodology. The general techniques involved in interval analysis date back to the work of Moore (1966). Interval analysis was originally used to calculate the error bounds on floating-point computations made on computers. Interval arithmetic is capable of handling a wide variety of linear and non-linear problems, and algorithms have been developed for calculating common engineering functions. Section 3.3 contains a more detailed description of the principles of interval analysis.

3 BACKGROUND

This chapter provides a general overview of qualitative reasoning, followed by a more focused description of constraint satisfaction problems (CSPs) and interval analysis. Combining the techniques of CSPs and interval analysis is then described, and provides the starting point for the original work of this thesis presented in the subsequent chapter.

3.1 Qualitative Reasoning

The field of artificial intelligence known as qualitative physics involves reasoning about systems that are subject to the fundamental laws of physics (Iwasaki, 1989). This provides a formal, systematic method of carrying out qualitative analysis for science and engineering applications. Research in this field was spurred by the observation that human experts are capable of reasoning successfully and easily about physical systems with incomplete or uncertain data based on knowledge of the fundamental principles of the system (Hayes, 1979). Expert systems (described above) were unable to perform in this manner, which was attributed to the fact that they are based on “shallow knowledge” of a system – knowledge that is very domain specific and is not capable of adapting to even slight deviations outside of the programmed domain. Qualitative physics on the other hand utilizes “deep knowledge” of a system that is based on the underlying physical principles which govern the shallow knowledge, and is therefore more capable of adapting to problems with uncertain or incomplete data. Since most engineering problems are based on well-established fundamental laws, as opposed to domain-specific empirical data or rules-of-thumb, they can be generally well suited to qualitative reasoning.

Data used in qualitative reasoning is less precise than data used in conventional numerical analysis. Qualitative variables may contain information such as the variable signs, relative magnitudes, and gradients, which can be used to carry out useful qualitative reasoning. Qualitative variables may also be represented as a series of intervals where the qualitative assignment to the variable is the name given to the interval

within which the variable's numerical value falls. For example, a numerical value could be given a qualitative assignment of negative, zero, or positive. The extension of qualitative intervals to semi-quantitative forms can be made by assigning numerical endpoints to the interval (this is described further in section 3.4 below).

Qualitative equations can be derived based on knowledge of the dependency of variables, or knowledge of the fundamental physical principles. The degree of precision of the functions used in the qualitative equations is limited only by the user's knowledge of principles governing the equation. A more realistic, precise equation will yield a more precise result, but even a very imprecise equation can still yield valuable results and insights into the system behavior.

Solving a system of qualitative equations involves finding a set of qualitative variable values that are consistent with the system of qualitative equations, meaning that all the qualitative equations are satisfied. This is a type of constraint satisfaction problem, where the constraints are the qualitative equations. The simplest general method of solving constraint satisfaction problems is an exhaustive generate-and-test approach, where all combinations of qualitative variable values are assigned, and the consistency of the qualitative equations is checked. This method is highly inefficient, and more efficient algorithms are described in the next section on constraint satisfaction problems.

In general, qualitative analysis will yield multiple solutions to a set of equations. Since the formulation of the problem is based on fundamental principles of the system, each solution represents a potential configuration or behavior of the system, all of which are systematically identified by the analysis. This may provide the analyst/designer with insight into solution spaces that may not be initially intuitive, which has particular importance in conceptual design applications. Qualitative reasoning has been applied to industrial problems, as demonstrated in Kuipers (1994).

3.2 Constraint Satisfaction Problems

The general formulation of a constraint satisfaction problem (CSP) consists of the following components:

- A set of *n* variables: $\{V_1, V_2, \ldots, V_n\}$
- Domain of variable values for each *n* variables: $\{D_1, D_2, \ldots, D_n\}$
- A set of *t* constraints, defining a relationship between some subset of the variables: $\{C_1, C_2, \ldots, C_t\}$

An instantiation is an assignment of specific domain values to some subset of the variables. An instantiation is said to be locally consistent if all of the constraints which are involved in the subset of variables are satisfied for the domain values used in the instantiation. Locally consistent instantiations are partial solutions to the CSP, and these are used with different methods to find the total solution. Two general techniques used for solving CSPs discussed here are backtracking and consistency algorithms.

3.2.1 Backtracking

Backtracking is the brute-force approach to solving CSPs. Backtracking involves a systematic, sequential instantiation of the variables. When a given constraint has all of its variables instantiated, the consistency of that constraint is checked. If the constraint is locally consistent, the instantiation of the remaining variables proceeds and additional constraints are checked. If the constraint is locally inconsistent, the backtracking procedure moves to the next sequential instantiation of the variables involved in that constraint, and then re-tests the local consistency. The process continues to iterate and will systematically find all of the globally consistent variable instantiations.

3.2.2 Consistency Techniques

Backtracking on its own is can be highly inefficient due to its systematic, exhaustive approach. Consistency techniques can be used as a preprocessor to a backtracking search to improve the efficiency of solving CSPs. Consistency techniques essentially provide a

check to see if there are values in the variable domains for which a given constraint cannot be locally consistent. These values can then be "pruned" from the variable domain, thus simplifying the subsequent backtrack search.

Consistency techniques are classified by the number of variables that are involved in the constraints checked in the consistency algorithm (Mackworth, 1977):

- **Node consistency:** the consistency of unary constraints (i.e. constraints that involve 1 variable) is checked. This is the simplest consistency technique.
- **Arc consistency:** the consistency of binary constraints (i.e. constraints that involve 2 variables) is checked. Since the pruning of variables may impact adjacent binary constraints, multiple passes through the system are required to achieve global arc consistency. Methods of mitigating this inefficiency have been proposed by Waltz (1975) and Mackworth (1977).
- **k-consistency:** a generalized form of consistency is "*k*-consistency", where for any *k* variables values exist that simultaneously satisfy all of the constraints related to the *k* variables. This algorithm can be used to solve a CSP on its own (i.e. $k=n$, where n is the total number of variables in the problem) or as a preprocessor to a backtrack search. Freuder (1978) outlines the basic constraint propagation mechanism, and presents a method for organizing constraints from lowest order and creating higher order constraints from combinations of lower order constraints through a "synthesis algorithm", and solves moving upwards.

The extent to which one implements consistency techniques as a preprocessor to backtrack search (or as solution methods in themselves) will impact the efficiency of the total solution process. The optimization of the CSP solution efficiency is not a focus of this research.

3.3 Interval Analysis

Practical engineering problems deal with quantitative variables, even if there is a very high degree of uncertainty surrounding those variables. The engineering usefulness of

the qualitative analysis techniques presented above can be greatly increased by adding some quantitative knowledge, regardless of how vague or imprecise it is. One method of implementing uncertain quantitative knowledge is to represent variables as intervals.

In interval analysis, variables are represented by intervals with an upper and lower bound as follows: $x=[x_L,x_U]$ where x_L and x_U are the lower and upper bounds of the interval. Analysis is carried out by operating on the end points of the intervals, and the solution consists of an interval representation of the "hull" of the solution space which contains all possible output values given the range of input values in the input intervals. The following example illustrates basic interval addition:

$$[1,2] + [3,4] = [4,6].$$

In addition to basic arithmetic operations, algorithms have been developed for all functions typically used in engineering calculations such as exponential, logarithmic and trigonometric functions. Libraries of interval functions are available for many common computational platforms such as Java, Matlab (Hargreaves 2002) and Maple.

Intervals provide an intuitive and simple method of representing uncertain data. While interval variables represent only a crude approximation of true statistical probability distributions, they have advantages over more complex representations. Statistical functions such as Gaussian distributions involve more complicated and arbitrary assumptions that are more difficult to justify than simple intervals, and lead to much more complicated computations (Davis, 1987). The advantages of using interval variables to represent uncertain data are particularly relevant in an engineering environment where the analyst is typically faced with limited statistical data and resources, and the strict statistical accuracy of the solution is of secondary importance.

3.4 Interval Constraint Satisfaction Problems

A system of constraints using interval variables can be formulated and solved using the procedures described for CSPs above. The general difference between solving qualitative

(discrete) vs. semi-quantitative (continuous) CSPs is that the domains of the interval variables are continuous, and therefore the "pruning" of the domains is carried out by updating the values of the interval bounds, as opposed to eliminating discrete qualitative values in the domain.

The formulation of the CSP uses two general types of constraints:

- Relational constraints: includes equalities and inequalities that can be implicit or explicit in form; this includes the objective function that defines the criteria for success/failure can be included as constraints
- Numerical constraints: numerical interval bounds on parameters

3.4.1 Developments in Research and Applications

Engineering research aimed at extending the capabilities of qualitative reasoning has combined the techniques used in CSPs and interval analysis as described above. This work has developed robust methods of solving constraint satisfaction problems with interval parameters, culminating in the Qualitative Engineering Software (QES) program by Gedig (1995), which was expanded to the QES2 program by Zhou (2007).

These applications are based on an architecture that allows reasoning with combinations of both qualitative and semi-quantitative data, where the semi-quantitative data is represented by intervals. The program then solves numeric CSPs by assigning each variable a dynamic interval domain that is propagated through the set of constraints. In general, this is an iterative process that converges asymptotically towards a solution.

QES utilizes the bounds consistency technique referred to as arc B(w)-consistency developed by Lhomme (1993). Two types of constraint propagation are used:

- **Propagation through arithmetic relations:** for constraints involving arithmetic relations, if the value of one variable in a constraint expression is changed, the values of the other variables involved in that constraint must be recalculated using interval arithmetic procedures.

- **Propagation through ordinal relations:** for constraints involving equalities and inequalities, the intersection of the intervals involved is subtracted from the appropriate variable to achieve consistency.

The interval methods and numeric constraint satisfaction techniques implemented have important key strengths. Firstly, the interval methods imply a guarantee that the solution will be bounded within the final domains of the intervals after the propagation is complete (or stated alternatively, they prove the non-existence of solutions outside of the final domain). Secondly, an explicit solution to the problem does not need to be found. This allows the formulation of the problem and the functions used to be very flexible.

Constraint inference techniques are used to complement constraint propagation (Simmons 1986). Three different types of constraint inference are used in QES. "Relation arithmetic" and "constant elimination arithmetic" define commonsense arithmetic relations that get lost in strict interval arithmetic, but provide valuable information that can be used to improved narrowing efficiency. Graph search is a technique used to determine new ordinal relations from the initial set of ordinal relations by searching of paths between expressions.

Further development in the form of the QES2 program combined the interval CSP techniques described above with adaptive graphing techniques. Additionally, higher order consistency techniques were implemented to improve the stability when solving larger problems. Application to a metal-fatigue problem demonstrated engineering applicability (Zhou 2003).

3.4.2 Solution Visualization

Adaptive graphing provides a valuable addition to the analysis of numeric CSPs by allowing more detailed analysis and visualization of the solution space. In general, the solution space generated by a single numeric CSP consists of an *n*-dimensional box, where *n* is the number of variables in the CSP. The dimensions of the box are given by the upper and lower bounds of each of the *n* variables after the CSP solution process has

narrowed the interval bounds. It is guaranteed that a solution exists within the box, and no solutions exist outside of the box (i.e. the box is a "hull" of the solution). In practical applications, the designer will benefit from more information about the shape of the solution space than a simple box. This is achieved by subdividing the solution space box from the initial CSP solution into smaller boxes, and then checking the consistency of those boxes. The smaller boxes are then used to build up a more detailed *n*-dimensional solution space. The number of *n*-dimensional boxes that need to be checked for consistency directly influences the computation time and increases as m^n where *m* is the number of divisions of the original box. In practice fewer boxes are required since multiple runs of increasing resolution are used (e.g. progressively set m=2, 4, 8, etc.), and then boxes that contained no solution in coarser runs do not need to be rechecked in finer resolution runs. The consistency check of each box in the subset is a proof of existence of a solution within that box using the numeric CSP solution methods for interval variables described above.

QES2 implements the adaptive graphing techniques described above to generate a 2-dimensional solution space hull for any two variables within the *n*-dimensional solution space, i.e. the 2-dimensional solution space plot contains all combinations of the two plotted variables that can possibly contain a solution to the CSP. QES2 allows any 2 of the *n* variables to be plotted as the axis of the solution space, and the trade off between plot resolution and run-time can be manually adjusted. Depending on the CSP and the specified resolution of the solution plotting process, the solution space may converge from an area to a line or point.

Visualizing solution spaces in practice is limited to either 2-dimensional graphs, 3-dimensional graphs (implementing contours or a rendering of a solid), or 4-dimensional graphs (implementing a sliced solid plus contours, multiple solids or a time axis). In complex cases this limits the number of variables which can be visualized in the solution space. Another engineering field where higher-dimensionality solution space visualization is used is in the construction industry for visualization of construction

sequences where automated methods have been developed for generating 4D construction model directly from a 3D design models (De Vries and Harink 2005).

4 THEORY

The following section describes the original contributions made by this research, namely the development of an analysis method referred to as Probabilistic Constraint Satisfaction (PCS) which extends the interval CSP method described above by introducing probabilistic variables. The goals of this research were to extend the capabilities of interval CSPs by conceiving of a method to implement a probabilistic representation of the problem input and output while maintaining flexibility and simplicity of problem formulation and robustness of the solution process. Additional goals were to implement capabilities for post-processing of higher-dimensionality and probabilistic solution spaces, and highlight through an industrial case study the capability of these techniques to analyze engineering design problems.

4.1 Overview of Methodology

This section gives an overview of the PCS methodology including the assumptions, techniques, and characteristics and interpretation of the results. The PCS methodology solves a CSP where some of the variables are probabilistic and have a known or approximated probability distribution function (PDF) associated with them and generates a probabilistic design space.

In the description of the methodology developed, there are two primary classifications of variables:

- Design variables: these are the variables the designer has control over the values, and wants to select values that yield an optimal solution. For conceptual design problems, designers typically have a few top-level design variables they wish to optimize, and other lower-level design variables are functions of the top-level design variables based on engineering relationships.
- Probabilistic variables: these are the variables for which the designer does not have control over, and there is uncertainty regarding their value.

Variables are classified in a secondary way for the purpose of describing the algorithms used in the PCS methodology in the most general way possible.

- Plotting variables: these are the variables that are plotted on the solution space axes, and thus there are typically only 2 or 3 plotting variables depending on the dimensionality of the output plot. The plotting variables typically correspond to top-level design variables.
- Narrowing variables: these are the variables for which a range of values is possible, and the range is narrowed to contain as closely as possible only the values yielding a feasible solution. Probabilistic variables are typically represented by narrowing variables. Design variables that are not included in the plotting variable set can be included in the narrowing variable set to gain insight into their valid ranges as a function of the selected plotting variables.

The following notation is used in the description of the methodology below:

- n: total number of variables
- n_{DES}: number of design variables
- n_{PROB}: number of probabilistic variables
- n_{PLOT}: number of plotting variables (typically $n_{PLOT} = 2$ or 3)
- n_{NARROW}: number of narrowing variables

The following relationships are noted:

$$n = n_{DES} + n_{PROB}$$

$$n = n_{PLOT} + n_{NARROW}$$

In general, it is most useful for the designer to set the design variables to be the plotting variables, and the probabilistic variables to be the narrowed variables. This provides the designer with a solution space that plots the range of available design choices, and shows the result of an objective function or the probability of meeting an objective function as a function of the plotting variables.

The PCS methodology is now summarized in the following steps:

1. **Problem definition:** the user defines the problem, specifically:
    - The range of interest of the design variables
    - The PDFs of the probabilistic variables
    - The constraining equations and constants governing the problem
    - The plotting variables defining the output plot axes and plot resolution (cell size)
    - The objective function to be plotted on the output plot

2. **Variable narrowing:** at each output cell the valid range of the narrowing variables is determined. Two general solution techniques are given below and described further in the following sections, as well as a hybrid solution method:
    - **Interval narrowing method**: the interval constraint satisfaction techniques described in the preceding chapter prove whether or not a solution possibly exists in a given cell with a single test of the CSP. If a solution exists, further narrowing of the narrowing variables yields the hull of their valid interval. The result is a matrix of *n*-dimensional boxes containing the region of valid solutions, where the n_{PLOT}-dimensions of the boxes are defined by the cell dimensions and the n_{NARROW}-dimensions of the boxes are given by the narrowed variable intervals.
    - **Grid search method**: the grid search solution method utilizes a systematic subdivision of the narrowing variables to generate a matrix of *n*-dimensional boxes and test the consistency of the CSP at a representative point, such as the midpoint, of each box. Again, the result is a matrix of *n*-dimensional boxes containing the region of valid solutions, where the n_{PLOT}-dimensions of the boxes are defined by the cell dimensions and the n_{NARROW}-dimensions are approximated based on the consistency tests.

3. **Probability calculation:** the probability of a solution occurring in each *n*-dimensional box calculated in the previous step can then be calculated. For each

n-dimensional box the probability that a solution exists in that box is calculated by integrating the PDF of each probabilistic variables over the narrowed interval of that variable. The probabilities of each of the probabilistic variables are then multiplied together incorporating any correlations to get the total probability of a solution occurring in that box.

4. **Output plot generation:** A probabilistic n_{PLOT}-dimensional solution space can now be constructed by summing depth-wise through the matrix of *n*-dimensional boxes. The result is a contoured design space, where the axis labels are the plotting variables of interest as defined by the user, and the contour value is the probability that a solution occurs within each cell based on the valid ranges of the probabilistic variables and their associated probabilistic data. The resolution (cell size) is defined by the size of the n_{PLOT} -dimensional cells, and can be made coarser or finer as desired by the user.

The PCS methodology as described above has the following attributes.

- Any probabilistic distribution shape for which the cumulative probability can be calculated over a given interval can be implemented. The shape of the PDF is irrelevant (normal, lognormal, rectangular, triangular, etc.) since the probability associated with each variable's interval can be calculated deterministically for common PDF shapes
- The magnitude of the probability of a given region does not influence the solution procedure or runtime; i.e. areas of very high or very low probability require the same solution time to generate similarly accurate results.

The following sections go into further detail describing the methodology and traits associated with the two main solution methods described above: the interval narrowing method, and the grid-search method. A comparison of methods is then made, and a hybrid method is proposed that takes advantage of the strengths of both methods.

4.2 Interval Narrowing Solution Method

The starting point for the interval narrowing solution method is based on two key capabilities of the solution of interval CSPs through interval narrowing as described in the preceding chapter:

- For a problem with *n*-variables, the initial *n*-dimensional solution space can be narrowed to an *n*-dimensional box that forms the hull of the solution to the CSP (i.e. it is guaranteed that no solutions lie outside of the box).
- The *n*-dimensional box can be subsequently subdivided, the remaining boxes can be proven to either exclude or possibly contain a solution to the CSP, and these boxes can be subsequently narrowed iteratively.

A series of examples is used below to illustrate the interval narrowing analysis procedure used in the PCS method, and to illustrate the characteristics of the output. The examples selected are as simple as possible so that the solution can be verified by inspection in order to demonstrate the methodology.

4.2.1 Problems with a Single Probabilistic Variable

Consider the simple constraint satisfaction problem given by the following single constraint:

$$y <= x+b$$

Suppose *b* is a probabilistic parameter with a PDF represented by the step function shown in Figure 2. The parameters *x* and *y* are considered the design variables, i.e. the designer has control over the values of these variables and wants to select values that yield a solution with a sufficiently high probability of occurrence. The desired output is a 2-dimensional solution space plot with contours representing the probability of a solution being available in a given region. For this simple problem the solution space can be easily generated by plotting the valid solution areas for various values of *b*, and calculating the associated probability of each area thereby generating the probability contours as shown Figure 3. This is the exact solution to this problem.

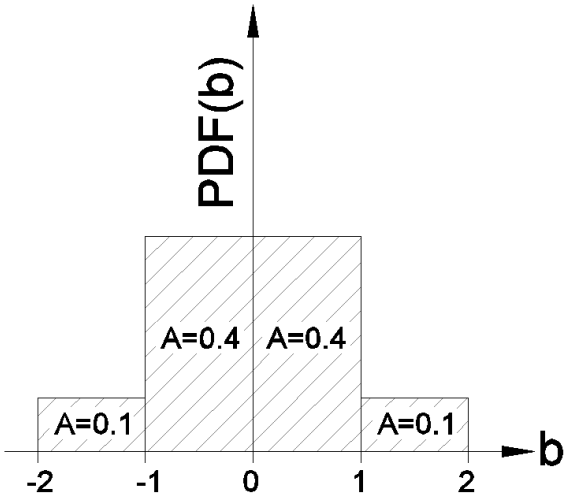

Figure 2: PDF of parameter *b* for example problem

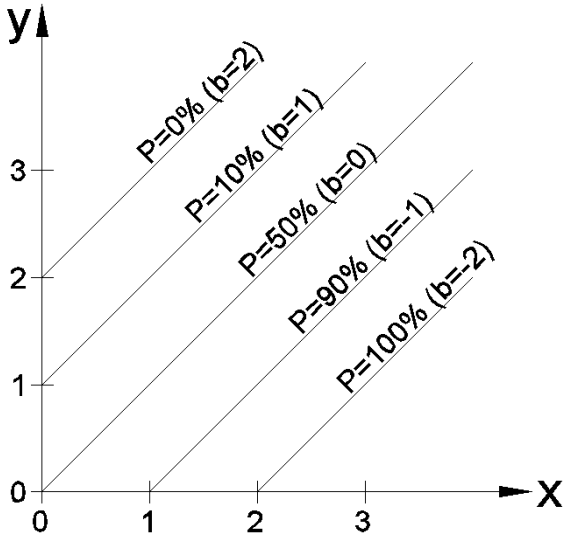

Figure 3: Probabilistic solution space for example problem *y* <= *x*+*b*

In Figure 3 above the plotted lines and their associated probability represent points that have an equal probability of occurrence, i.e. they are contours on the probabilistic surface. For example, points that fall between the P=10% and P=50% contours have a

probability of occurrence between 10% and 50%. Points that are below the P=100% contour have a probability of occurrence of 100%.

Approaching the problem using the techniques of probabilistic interval narrowing is now described. For the first pass, it is assumed the size of each cell will be (dy,dx) = (1,1). For each cell, the techniques of interval narrowing are then used to determine the possible values of b which make a solution in a given cell valid. For example, in the cell (x, y)=(0:1, 0:1) solutions are possible for values of $b=(-1,\infty)$. By integrating the PDF of b, we determine that the probability of b occurring in this interval is 90%. Since there are no other probabilistic variables, the probability associated with this cell is 90%. Carrying out the same procedure for the remaining cells, the solution space shown in Figure 4 can be generated.

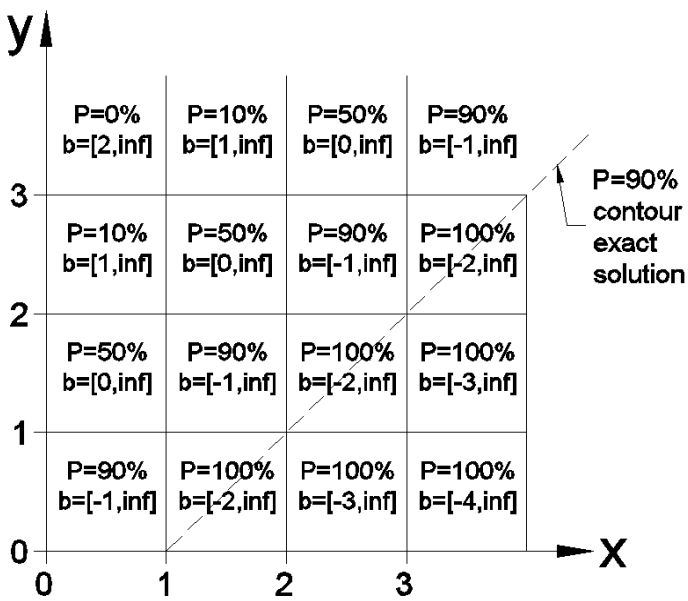

Figure 4: Probabilistic solution space for example problem $y <= x+b$ approximated via interval constraint satisfaction with 1x1 cells

It can be seen in Figure 4 that only the bottom right corner of the 90% cells touch the actual 90% contour from Figure 3. When the interval of *b* is narrowed for a given cell, the entire intervals of *x* and *y* that define that cell are used to determine the largest interval of *b* that provides a valid solution. Therefore, in this case the probability associated with a given cell can be interpreted as the maximum probability that a point within that cell contains a solution. Generally, the plotted probability is not the maximum but rather an upper bound and this will be illustrated in further examples. As the plot resolution is increased (cell size is decreased), the solution space generated with interval narrowing converges to the actual solution, and therefore the interpretation of the cell probability becomes less important at higher resolutions. This can be seen in Figure 5 which plots the solution with the cell size reduced to (*dx*,*dy*) = (0.5,0.5). The 90% cells now lie closer to the actual 90% contour.

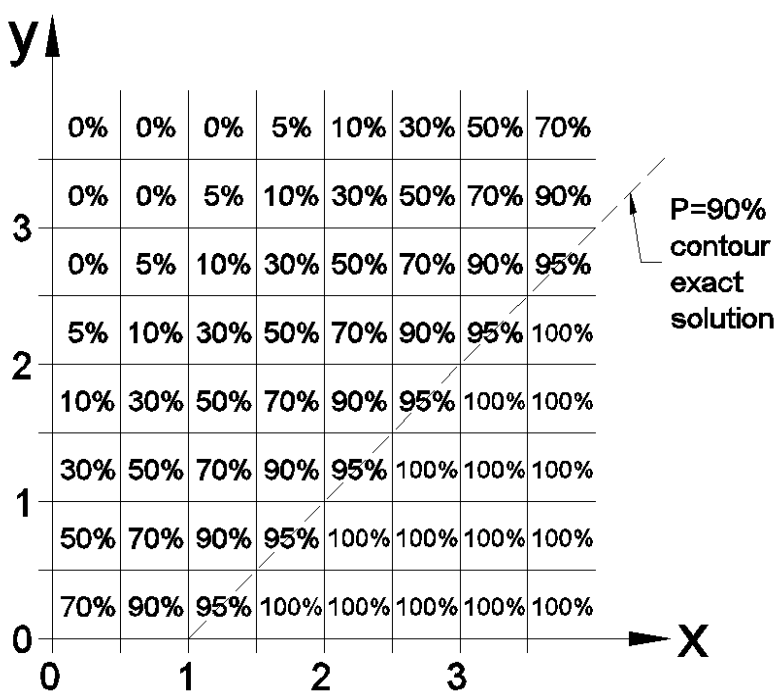

Figure 5: Probabilistic solution space for example problem *y* <= *x*+*b* approximated via interval constraint satisfaction with 0.5x0.5 cells

The user can formulate the problem so that the solution space can represent either the regions of success or failure, so it follows that the probabilistic design space can represent an upper bound on the probability of either success or failure. In many engineering problems a reliable margin of safety is required, and therefore it is more useful to find the upper bound on the probability of failure. In other situations involving the cost optimization or comparison of design concepts, it may be more intuitive to find the upper bound on the probability of success. In any case the problem formulation is very flexible and the user must decide through experience how best to define the problem.

4.2.2 Problems with Multiple Probabilistic Variables

The following example illustrates another property of the solution space generated by the interval narrowing PCS process, which is that the plotted probability represents an upper bound to the probability that a solution exists within a given region. This was demonstrated in the previous example with respect to cell size (i.e. the probability associated with a given cell is the upper bound of the exact solution). The example below demonstrates how the upper bound property is required for problems involving multiple probabilistic variables.

Consider the following CSP, consisting of a single equation representing a solid sphere:

$$x^2 + y^2 + z^2 <= r^2$$

Initially the only probabilistic variable is *r* with its PDF shown in Figure 6. The first quadrant of the solution space with parameters *x* and *y* plotted on the axes is shown in Figure 7 and Figure 8. Figure 7 gives the upper and lower bounds of the parameters *r* and *z* after interval narrowing is carried out at each cell, and Figure 8 shows the probabilistic solution space generated by integrating the PDF of the probabilistic variable *r* in each cell. Note that the narrowed values of *z* are available as output even though in this case it is not required for calculating the cell probabilities since *z* is not a probabilistic variable.

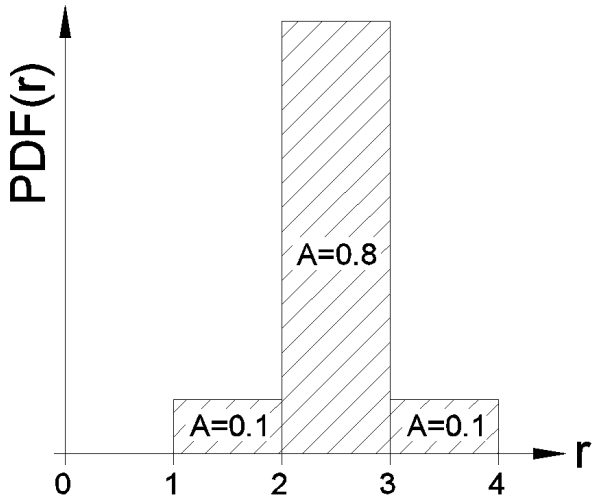

Figure 6: PDF of parameter *r* for example problem

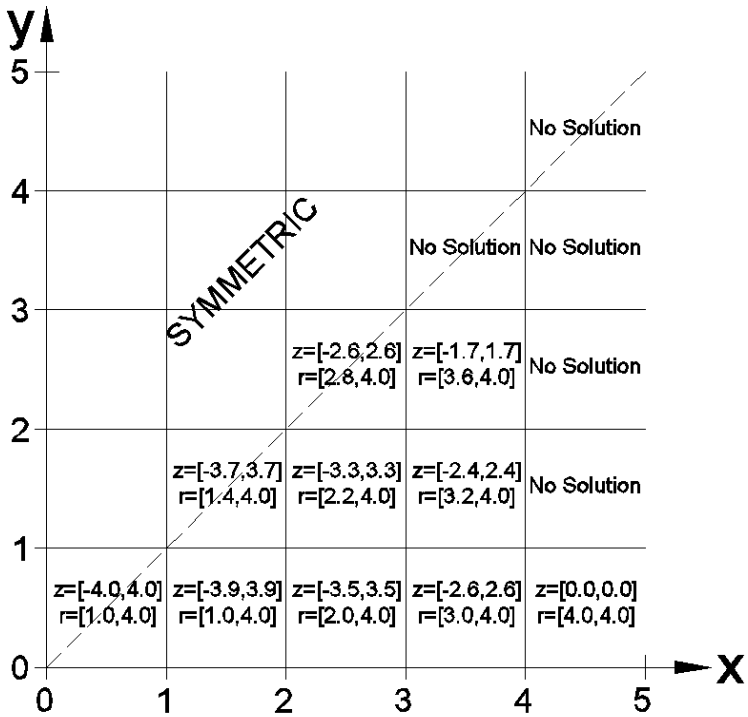

Figure 7: Values of narrowed intervals of *r* and *z* at each cell for example problem $x^2 + y^2 + z^2 <= r^2$

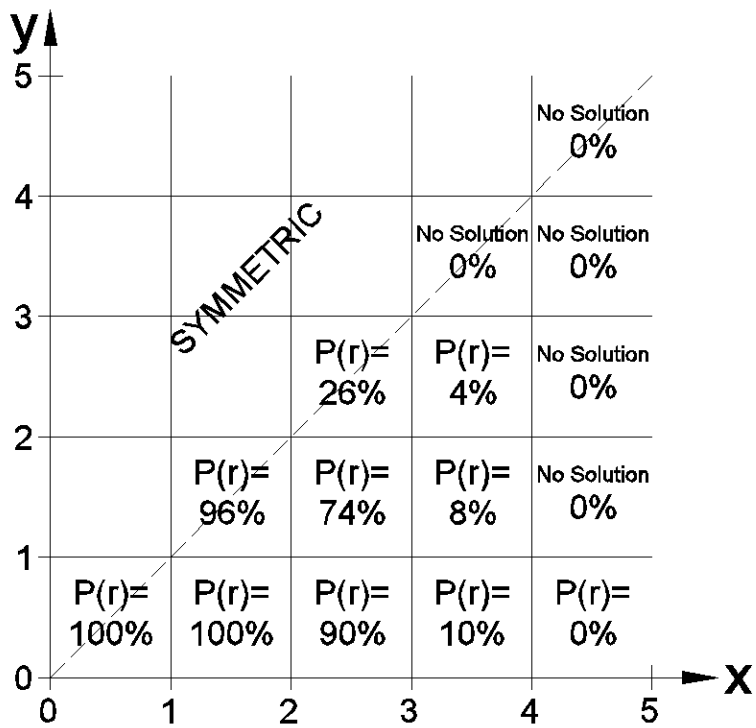

Figure 8: Probabilistic solution space for example problem $x^2 + y^2 + z^2 <= r^2$ considering only a single probabilistic variable *r*

The example is now extended by making *z* a probabilistic variable defined by the PDF shown in Figure 9. The probabilistic solution space is then developed by taking the narrowed intervals of *r* and *z* at each cell (as shown in Figure 7), and integrating the respective PDFs over these intervals. The resulting probabilistic solution space is plotted in Figure 10.

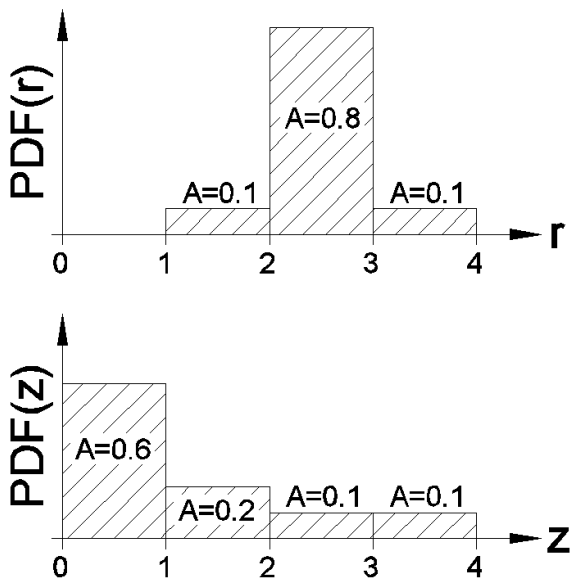

Figure 9: PDFs of parameters *r* and *z* for example problem

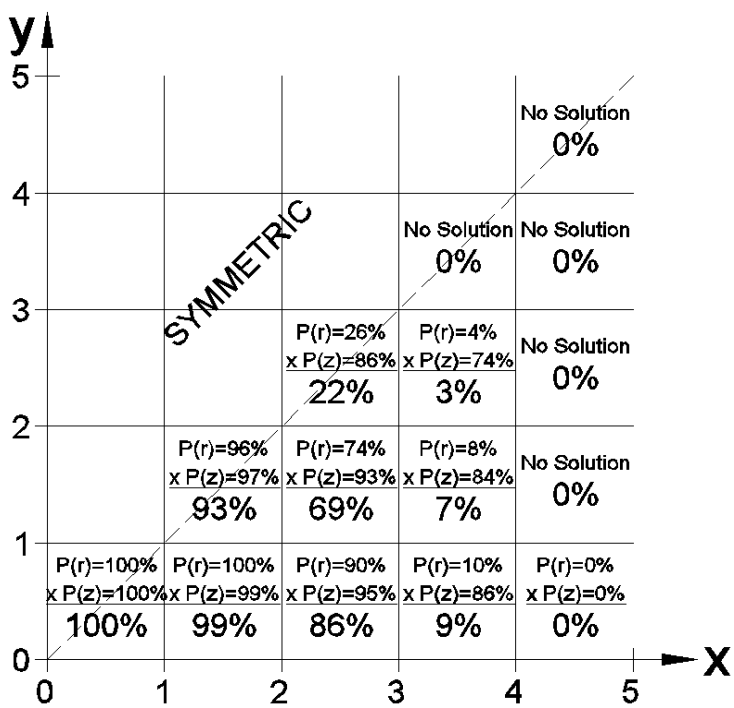

Figure 10: Probabilistic solution space for example problem $x^2 + y^2 + z^2 <= r^2$ showing the individual and combined probabilities of the valid intervals of *r* and *z*

The narrowed intervals of *r* and *z* represent the "hull" of the solution space for a given cell, i.e. it is guaranteed that no solution exists outside of the narrowed intervals. This does not mean that every point in the narrowed *r*-*z* space contains a solution. Since the subsequent calculation of probability integrates the PDFs for *r* and *z* over the entire narrowed interval, it overestimates the probability that a solution actually occurs in that region, again providing an upper bound to the probability.

To further illustrate this we can add an additional constraint y=[0,1] and now plot *z* on the vertical axis (this is essentially taking a cross section through the solution space shown in Figure 10). The narrowed intervals for *r* can then be calculated for each specific interval of *z*, and therefore the probabilities can be calculated more accurately since we now know for each interval of *z* what interval of *r* provides a valid solution. The *z*-*x* probabilistic solution space is plotted in Figure 11. In the case of this example, further subdivision will converge to the actual probabilities in each box (in general this is not the case since more probabilistic variables may be introduced, or the interval hull may have gaps in it containing no solution; this is expanded on below). By summing the probabilities for each column of width dx, we can calculate the total probability that a solution occurs in the cell (dx, y = [0,1]). This provides a more accurate approximation to the probability (still an upper bound) when compared to the original upper bound probability calculation carried out without subdividing *z*, and the results are compared in Table 1.

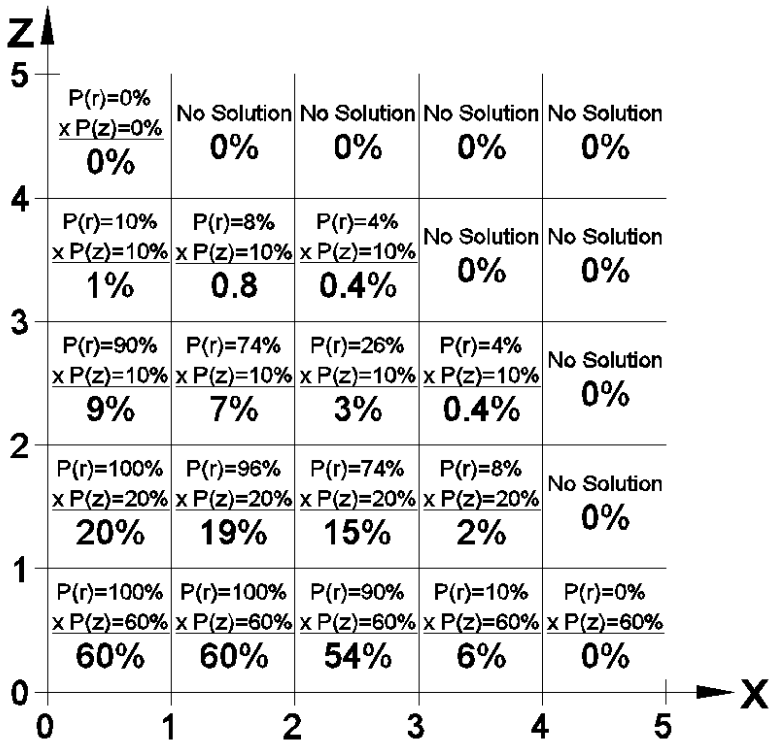

Figure 11: Probabilistic solution space for example problem $x^2 + y^2 + z^2 <= r^2$ with probabilistic variable *z* subdivided and plotted on the vertical axis over *y* = [0,1]

Table 1: Probability over all *z* for *y*=[0,1] with and with-out depth-wise subdivision

Method	***x*=[0,1]**	***x*=[1,2]**	***x*=[2,3]**	***x*=[3,4]**	***x*=[4,5]**
With subdivisions of *dz*=1 (from summing columns in Figure 11)	90%	87%	72%	8%	0%
Without subdivisions of *z* (from Figure 10 for *y*=[0,1])	100%	99%	86%	9%	0%

In the example problem in Figure 11, the actual probability is closely approximated since it is limited to two probabilistic variables and a solution space that doesn't contain any "holes". As more probabilistic variables are added and the solution space becomes more non-linear, the original upper bound probability calculated for 2-dimensional cells may become potentially more inaccurate. A solution to this is subdividing the solution space

depth-wise which essentially generates a matrix of higher-dimensionality cells for which the intervals defining each dimension are user-specified, and this is the approach taken by the grid-search solution method described below.

4.3 Grid-Search Solution Method

The grid search solution method involves a systematic subdivision of the narrowing variables to approximate their valid interval range. The basic approach is to generate a matrix of *n*-dimensional boxes, where the number of subdivisions along both the n_{PLOT} and n_{NARROW} dimensions is defined by the user. At each of the boxes, a consistency test of the CSP is carried out by instantiating the variables at a representative point, such as the box midpoint, and a truth value is assigned to the box based on whether or not the CSP is satisfied. The probability of the probabilistic variables occurring along the subdivided intervals can then be calculated and summed depth wise to generate the output plot in the same way that is used in the interval narrowing procedure.

The previous 4-variable example problem demonstrated the potential value of adding a depth-wise grid-search element to the analysis. The plot in Figure 10 originally generated a matrix of n_{PLOT} =2-dimensional cells, and the intervals of the remaining n_{NARROW} =2 variables were narrowed at each cell. The plot in Figure 11 essentially expanded the analysis to include a depth-wise grid search by fixing the value of one of the depth-wise narrowing variables to the user-defined range y = [0,1] and subsequently the interval of the remaining one variable r was narrowed. The result indicated the increased accuracy achieved by using a depth-wise grid search.

This grid-search method has two important properties when being compared to the interval narrowing method:

- The probabilistic solution space converges to the actual probability as the cell resolution is increased, as opposed to providing an upper bound.
- No interval narrowing is required since all of the intervals for each narrowing variable are user-defined through subdivision, and all that needs to be done is to

verify whether or not it is possible that a solution exists in the box by checking the consistency of the CSP with the variables instantiated at the box mid-point.

4.4 Comparison of Solution Methods

The interval narrowing method and grid-search method each have their advantages and disadvantages when used to solve the narrowed intervals of the probabilistic variables. These are summarized in the table below, and described in the following text.

Also shown in the table and discussed below is a hybrid method that utilizes aspects of both interval narrowing and grid-search. The well-known technique of Monte Carlo simulation is also shown for comparison.

Table 2: Comparison of solution methods for determining valid ranges of probabilistic variables

Method	Interval Narrowing	Grid-Search	Hybrid: Interval Narrowing & Grid Search	Monte Carlo
Initial solution space search	Single narrowing run of interval CSP	Naive	Single narrowing run of interval CSP	Naive
Dimensionality of results matrix	n_{PLOT}	$n = n_{PLOT} + n_{NARROW}$	User defined: $n_{PLOT} < n_{USER} \leq n$	n_{PLOT}
Size of results matrix	$(s_{PLOT})^{n_{PLOT}}$	$(s_{PLOT})^{n_{PLOT}} \cdot (s_{NARROW})^{n_{NARROW}}$	$(s_{PLOT})^{n_{PLOT}} \cdot (s_{NARROW})^{n_{USER}}$	$(s_{PLOT})^{n_{PLOT}}$
Number of analysis per results matrix entry	1 (includes interval narrowing iteration)	1 (check of CSP consistency)	1 (includes interval narrowing iteration for $(s_{PLOT})^{n_{PLOT}}$ boxes)	Varies, typically 10^1 to 10^7
Info on design variables	Guarantee of bounded solution at each entry	List of valid tested points, systematic	Guarantee of bounded solution at each entry	List of valid tested points, random
Robustness of solution space	Guarantee of non-existence of solution outside solution space	Point sampling, systematic points	Guarantee of non-existence of solution outside solution space	Point sampling, random points
Capabilities of finding discontinuities in solution space	Not found	Depends on selected resolution, systematic	Depends on selected resolution, systematic	Depends on number of analysis, random
Convergence of solution	Upper bound on solution	Converges to exact solution	Upper bound on solution, converges to exact solution	Converges to exact solution

s_{PLOT} = the number of subdivisions along the plotted variables

s_{NARROW} = the number of subdivisions along the narrowed variables

The main drawback of the grid search approach is that the number of result matrix entries is on the order of s^n where s is the number of variable subdivisions since systematic instantiation of all variables is required to test truthfulness. Therefore, the grid-search technique results in combinatorial explosion in problems with a high number of probabilistic variables. However, only a single test of truthfulness at the box midpoint is required for each box.

The strength of the grid-search method is that it converges to the exact solution as the number of subdivisions of the narrowing variables is increased. This is due to its capability of detecting discontinuities or "gaps" in the intervals of the narrowing variables, and also its capability to assess consistency of the constraints while considering various segments of the intervals of multiple variables in combination with each other, as opposed to assessing the intervals in their entirety. The differences in the capabilities of the interval narrowing and grid-search methods with regards to the subsequent probability calculations are highlighted in the following examples.

Consider a problem which has a solution where the probabilistic variable x is found to be valid over the following discontinuous intervals:

$$1.2 < x < 2.8 \text{ and } 3.7 < x < 5.3$$

Assume x has a Gaussian distribution associated with it with a mean of $\mu_x = 3$ and a standard deviation of $\sigma_x = 1$. Using basic statistics, the probability of x occurring within the valid interval given above is taken by integrating the associated PDF over the valid interval as depicted in Figure 12 below.

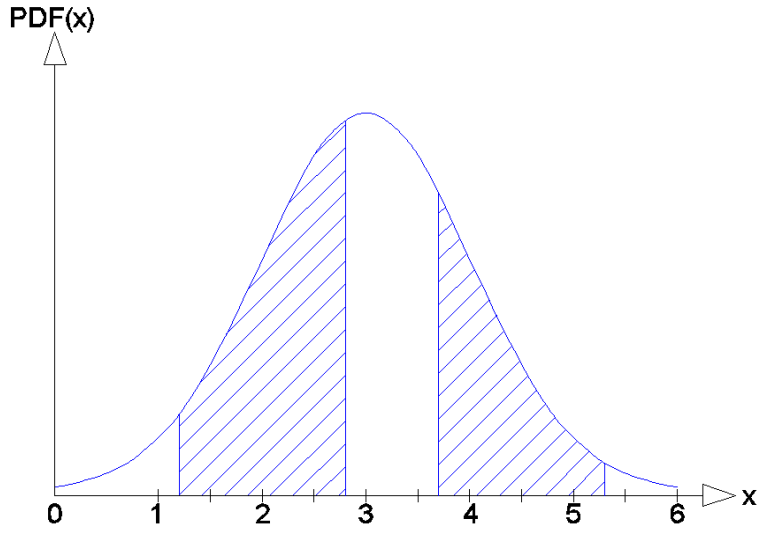

Figure 12: PDF of variable *x* with exact valid interval plotted

If interval narrowing is used to estimate the valid interval of the variable *x* and its associated probability it is not capable of detecting the gap in the valid interval, and thus overestimates the total interval and the probability that *x* will occur in that interval as depicted in Figure 13 below. This is consistent with the idea that interval narrowing provides a "hull" of the solution space and thus an upper bound on the estimate of the probability.

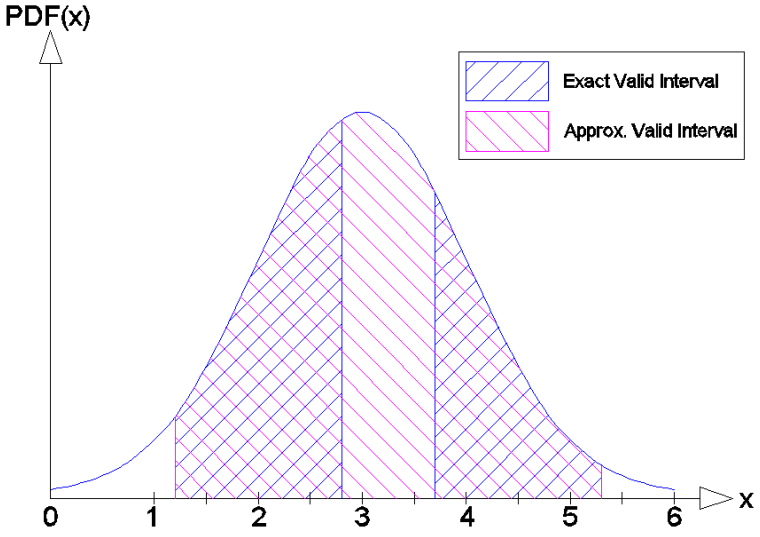

Figure 13: Valid interval approximated by interval narrowing

If grid search is used to estimate the valid interval of x it will detect the gap provided the grid search resolution is fine enough. An example is shown in Figure 14 below for the case where the intervals dx = [0,1], [1,2], …. [5,6] are checked by testing the value of x at the midpoint of the interval against the governing constraint given above. It is apparent that the grid search method may either overestimate or underestimate the interval and its corresponding probability, but it will converge to the exact solution as the resolution is increased.

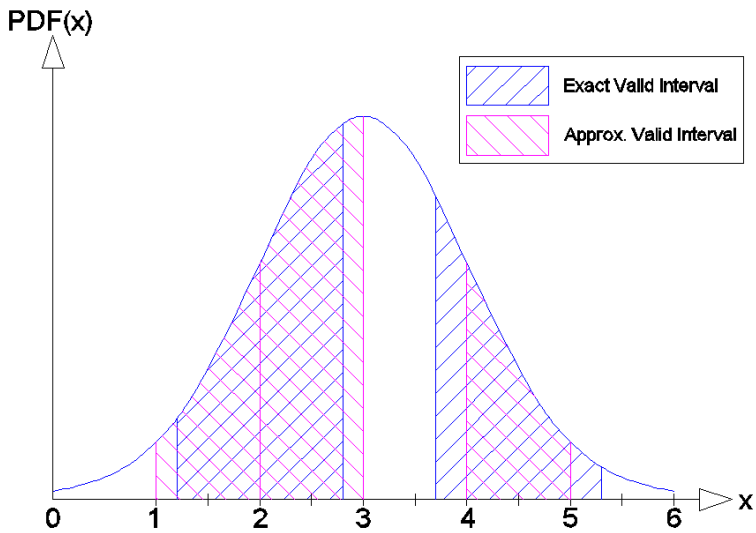

Figure 14: Valid interval approximated by grid search with *dx*=1, and consistency checking at interval midpoint

The next example illustrates the other drawback of using interval narrowing on its own which is that it assesses the intervals of multiple variables in their entirety and cannot assess the validity of the solution while considering only various segments of the intervals. To illustrate this consider a problem with a single constraint:

$$x \geq y$$

where x and y are probabilistic variables with triangular distributions as shown in Figure 15 below.

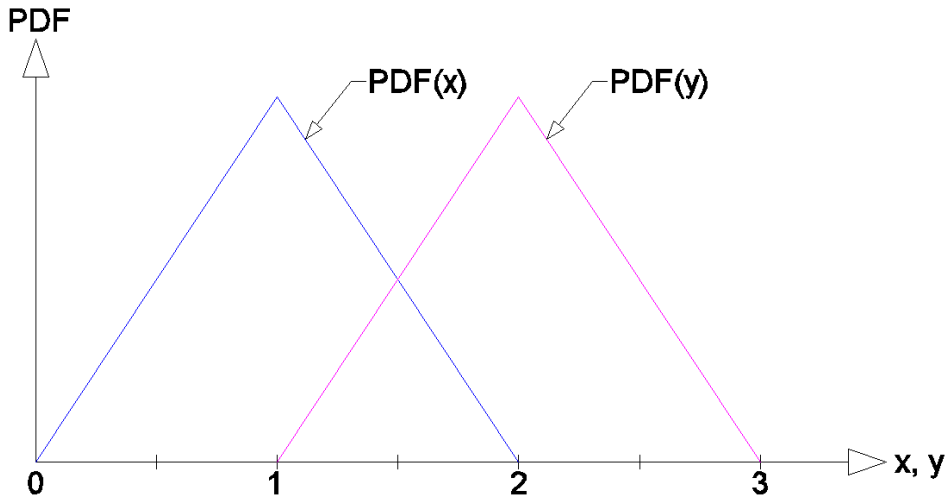

Figure 15: PDFs of variables *x* and *y*

Figure 16 below now shows the intervals of *x* and *y* after interval narrowing that satisfy the constraint given above. An upper bound of the probability of a solution existing is then calculated as follows (it will be shown below that the result, while providing an upper bound, is quite inaccurate in this case):

$$P_x = P(1 < x < 2) = 0.5$$
$$P_y = P(1 < y < 2) = 0.5$$
$$P = P_x \cdot P_y = 0.25$$

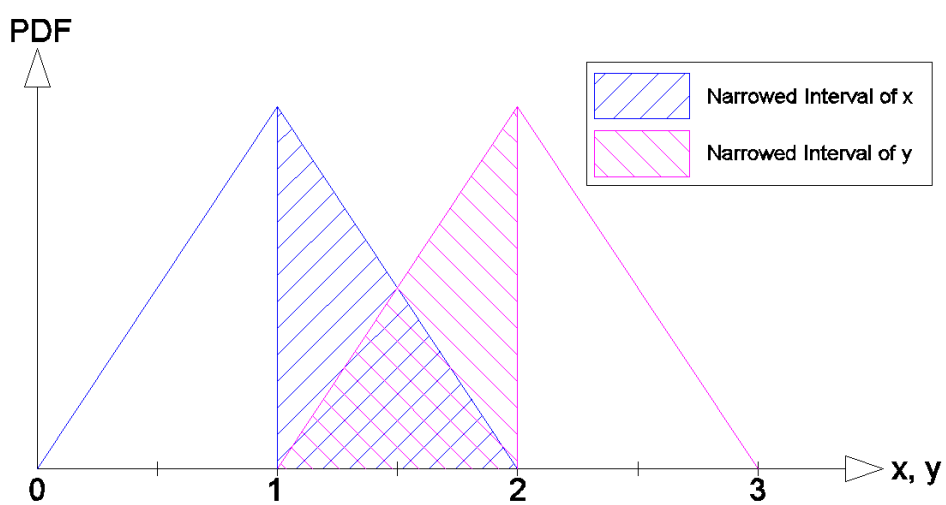

Figure 16: Valid intervals of *x* and *y* after interval narrowing

Now the result is calculated using the grid search method. To illustrate this the total intervals of *x* and *y* are subdivided into *n*=4 sub-intervals. The intervals of *x*=[0,0.5], [0.5,1], [1,1.5] and [1.5,2] are tested systematically at their midpoints in combination with the intervals of *y*=[1,1.5], [1.5,2], [2,2.5] and [2.5,3]. Only the last two instantiated intervals of *x* yield a valid tested interval for *y* as illustrated in Figure 17 and Figure 18

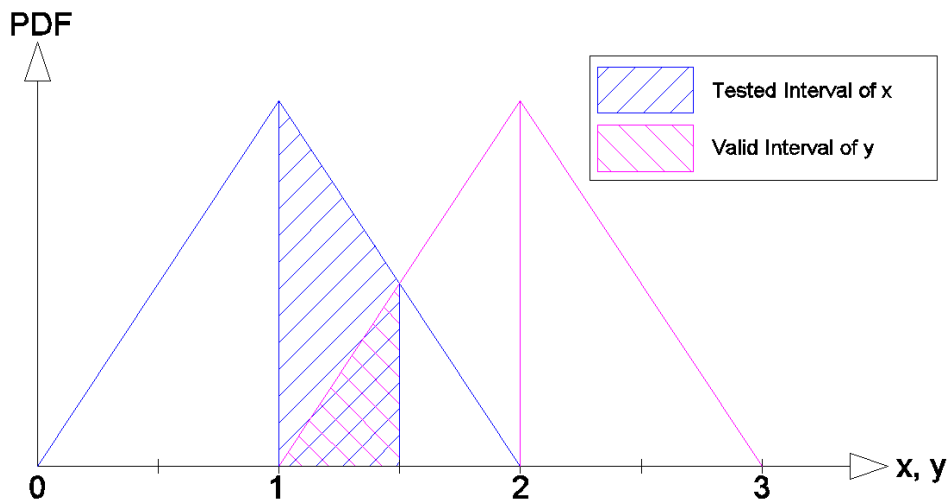

Figure 17: Valid interval of *y* for test interval of *x*=[1:1.5]

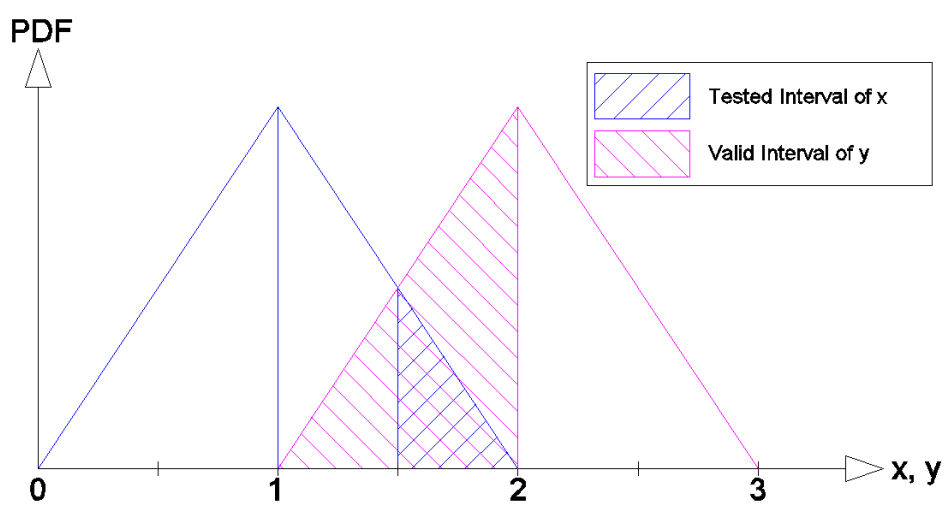

Figure 18: Valid interval of y for test interval of x=[1.5:2]

Using the results for the valid intervals in combination as illustrated above the probability of a solution existing can be calculated as follows:

$$P = \sum P_x \cdot P_y$$
$$P = P(1 < x < 1.5) \cdot P(1 < y < 1.5) + P(1.5 < x < 2) \cdot P(1 < y < 2)$$
$$P = 0.375 \cdot 0.125 + 0.125 \cdot 0.5$$
$$P = 0.1094$$

This result is substantially lower in this case than the upper bound provided by the interval narrowing method. In this case n=4 subdivisions were used on the initial intervals. Table 3 below shows the estimated probability for various values of n, where n is the number of subdivisions of variables x and y. It can be seen that the grid search method converges asymptotically to the exact solution as n increases. The Matlab source code for producing this result is given in Appendix B.

Table 3: Probability of solution for *n* subdivisions

n	***P***
$\mathbf{2^0 = 1}$	0
$\mathbf{2^1 = 2}$	0.25
$\mathbf{2^2 = 4}$	0.1094
$\mathbf{2^3 = 8}$	0.0684
$\mathbf{2^4 = 16}$	0.0535
$\mathbf{2^5 = 32}$	0.0472
$\mathbf{2^6 = 64}$	0.0444
$\mathbf{2^7 = 128}$	0.0430
$\mathbf{2^8 = 256}$	0.0423
$\mathbf{2^9 = 512}$	0.0420
$\mathbf{2^{10} = 1024}$	0.0418
$\mathbf{2^{11} = 2048}$	0.0417

The above examples illustrated some of the drawbacks of interval narrowing that can be addressed by the grid search method. The next section discusses hybrid approaches that take advantage of the intrinsic strengths of both methods.

4.4.1 Hybrid Method

Hybrid solution methods involving interval narrowing combined with grid search methods are capable of utilizing the advantages of both methodologies. The most logical approach to yield efficient and robust results involves using the interval narrowing solver for the initial pass to narrow the intervals of the probabilistic variables. Grid search method can then be used to further subdivide the initial narrowed interval to search for holes and variable instantiation combinations. The checking of the CSP consistency at each of the subdivided boxes could then be made using interval CSP checks, as opposed to the box mid-point testing used in the grid search technique described above, thus yielding more robust results. If gaps are found upon subdivision, further iterations of the interval narrowing process would converge accurately to the edges of the gaps.

The hybrid approach takes advantage of key fundamental aspects of both the interval narrowing and grid-search methods. From interval narrowing it draws the following traits:

- The initial solution space can be narrowed to focus subsequent runs only on the region of valid solutions
- The narrowed design variable intervals provide a logical proof that the valid solutions are bounded within the narrowed interval

From the grid-search method it draws the following traits:

- The solution converges to the exact solution
- It is capable of locating discontinuities in the design space due to gaps or variable combination effects

The hybrid approach also provides the opportunity for the user to specify the number of variables to be subject to further subdivision, denoted by n_{USER} in Table 2 above where $n_{PLOT} < n_{USER} \leq n$. For many cases, especially those involving monotonically increasing or decreasing functions, the simplest case of setting $n_{USER} = n_{PLOT}$ (which reverts to the interval narrowing method) likely yields insightful and reasonably accurate results for first pass analysis. At the other end of the spectrum for more rigorous analysis one can set $n_{USER} = n$.

Various strategies can be implemented during the analysis to maximize understanding of the problem with minimum problem formulation and computation time. For example:

- The resolution and number of probabilistic variables can be increased with additional runs
- The user can individually control resolution of various parameters, e.g. can make s_{PLOT} large and s_{NARROW} small, where s_{PLOT} and s_{NARROW} are the number of subdivisions along the plotting and narrowing variables, respectively

4.4.2 Comparison to Monte Carlo

A well known general solution method capable of handling uncertainty is Monte-Carlo simulation. A Monte Carlo simulation could be used to solve the associated probability with each cell in the solution space by randomly picking values for the probabilistic variables based on PDFs and carrying out a sufficient number of simulations to adequately estimate the probability of a solution occurring in each cell.

The PCS method developed through this thesis has several fundamental advantages over the Monte Carlo method. The Monte Carlo method results in a list of randomly selected points where existence of a solution was tested. The soundness of this list is dependent on the density of tested points, and valid solution spaces may be missed especially at lower resolutions. The interval narrowing method provides a much stronger result that is a guarantee of the upper and lower bounds of the valid solution space for each variable at each box based on the narrowed interval of that variable.

Monte Carlo techniques begin with a naive search of the solution space, thus wasting computational effort analyzing invalid regions. Alternatively, some additional methodology must be used to solve the initial solution space. The interval narrowing approach has the advantage that the initial narrowed solution to the CSP bounds the region of possible solutions.

Monte Carlo analysis requires that at each cell a large number of simulations are required to get a reasonably accurate probability for the cell. The number of runs is especially high when dealing with very high or very low probability events. For example, if a box is considered where the probability of failure is on the order of 10^{-6}, the Monte Carlo simulation would need to get at least a few failures to establish a reasonable probability of failure, so the number of runs would be on the order of 10^{6} to 10^{7}. Examples of very low probability problems are routinely encountered in fields such as structural engineering, where structures are designed to a very high level of reliability so failures are extremely rare. By contrast, when looking at a box that has a 50/50 probability of failure, the number of runs would be on the order of 10^{1} to get a reasonably accurate result. The interval narrowing and grid search approaches are not impacted by this effect since the runtime required to determine the variable intervals that are contained in the solution is independent from the probability associated with those intervals.

5 APPLICATION

This chapter describes the software application of the original theory of this thesis as presented above, and includes a description of the following:

- Selection of software platform
- Application architecture
- Description of input, solution, and output modules

The chapter concludes with a series of examples to demonstrate the functionality of the software and the range of capability.

5.1 Software Platform

Matlab was selected as the software platform used for the development of theory application and proof-of-concept for this thesis. It provides a programming and computing environment with many built-in features and add-ons to facilitate easy development of computational tools.

Matlab has several key features that made it the preferred platform for this thesis work:

- Capability of handling multi-dimensional arrays which is required for manipulating the solution space probabilistic data set and plotting data.
- Built-in plotting capabilities that enable variety of visualizations of 2D, 3D, and 4D (sliced 3D with contours) data.
- Statistical toolboxes which access functions used to carry out statistical calculations.

Matlab also has an integral interface to Java, and a built-in Java Virtual Machine. User-defined Java classes are developed in a Java development environment that is external to Matlab. Once these have been compiled, methods can be invoked on Java objects directly from Matlab. This feature allows access to existing interval CSP solution and narrowing algorithms that have been developed on a Java platform.

Features of Matlab that may be used to advantage in further development of applications include simple GUI creation, and availability of an interval analysis toolbox. The INTLAB (INTerval LABoratory) is available as a free download from the Institute for Reliable Computing at the Hamburg University of Technology (http://www.ti3.tu-harburg.de/~rump/intlab/). A background on interval arithmetic and introduction to INTLAB is given in Hargreaves (2002).

5.2 Application Architecture

The application architecture contains three major modules: an input module, solution module, and a post-processing module. These modules and their inter-relations are depicted in Figure 19 and described in the following sections.

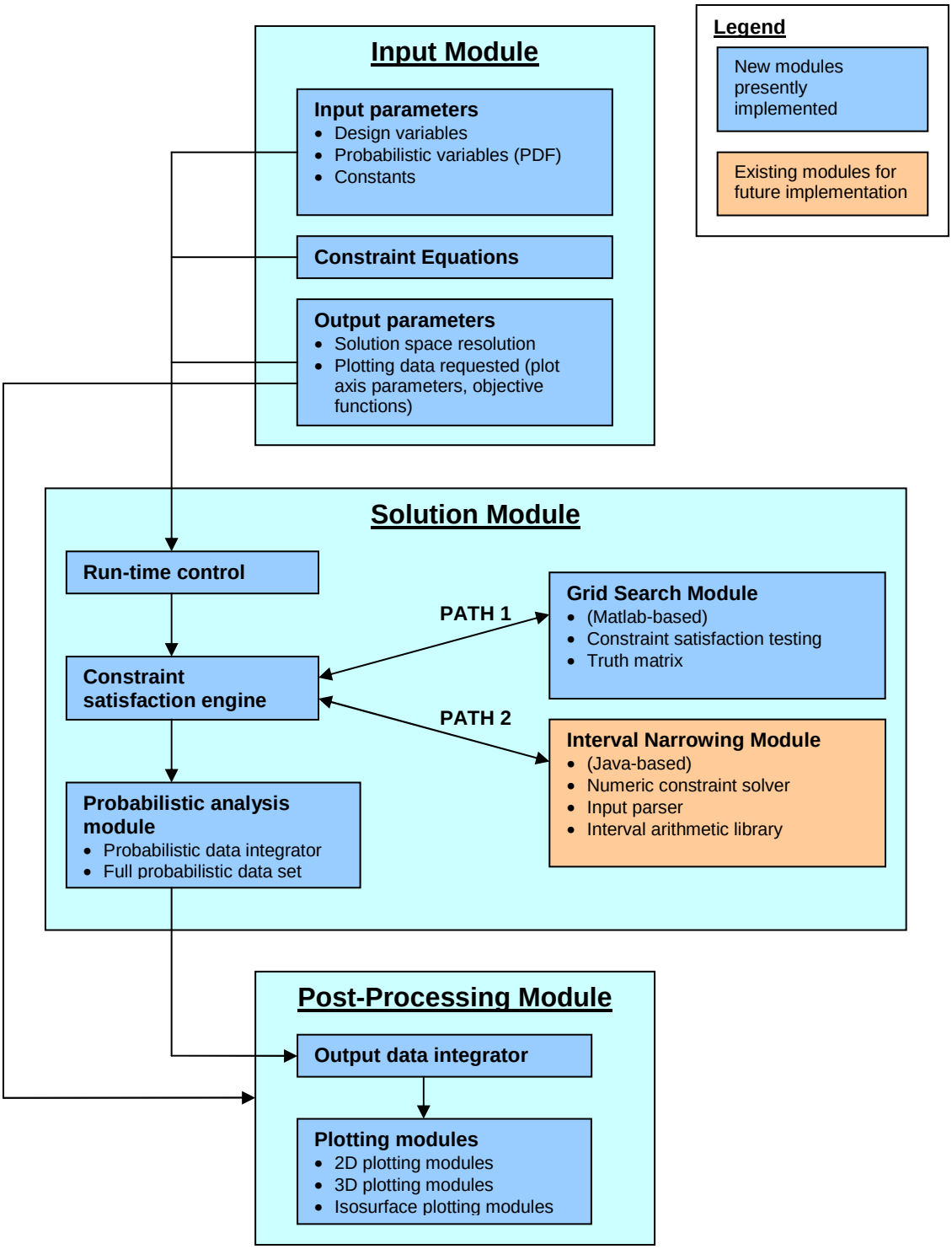

Figure 19: Architecture of software application

5.2.1 Input Module

The input module is where the user defines the data needed for the solution to execute. The major components are as described below.

5.2.1.1 Input Parameters

Three types of input parameters are used:

- Design variables: these are the parameters for which the designer must select values for when carrying out the initial design. In this application, they are expressed as an interval containing the entire range of parameters that the designer considers feasible points. There is no limit to the number of design variables that can be included in the analysis; however, the design variables should be selected carefully so that the output can be interpreted since design space plots are limited to 3-dimensional plots plus a contour value. The syntax for defining the design variable bounding values and resolution of the output plotting space is as follows:

```
Design_Variable = [low value, high value, no. boxes]
```

- Probabilistic variables: these are the parameters which have uncertain values and are assigned a probabilistic distribution. The designer does not have direct control of these parameters at the time of analysis, either because of lack of knowledge, or because of inherent uncertainty in the parameter. There is no limit to the number of probabilistic variables that can be implemented in the analysis. Probabilistic variables are assigned a distribution type and key parameters which quantify the distribution. For the example problem, a symmetric triangular PDF is defined by the following syntax:

```
Probabilistic_Variable = [low value, high value, no. boxes]
```

- Constants: these are assigned to fixed-value parameters. Assigning parameter names to constants enables easier coding of calculations, and simplifies future changes.

5.2.1.2 Constraints

In the context of CSPs, constraints are simply the equations which define the problem, and can use all typical engineering functions, equalities, inequalities, and logical functions. The constraints may take the form of engineering calculations that operate on input parameters, performance functions such as building code checks (i.e. the design must meet a code requirement to be considered valid) or other feasibility checks (for example the design must be within a maximum estimated cost to be considered valid).

5.2.1.3 Output Parameters

The output parameters specified by the user include the parameters to plot on the solution space axes, the solution space resolution, and any special plotting data requested in post-processing, such as an objective function, or probability-weighted results.

5.2.2 Solution Module

The solution module contains the solver module and the probabilistic analysis module. The output of the solution module is the full probabilistic data set over the requested design variable space.

Two solver engines are described below.

- Grid-search solver module: this solver module was developed in Matlab as part of the thesis work. The module carries out the calculations required by the CSP, verifies the existence of solutions, and estimates the intervals of the probabilistic variables for each box within the design space that are consistent with the problem constraints. This is achieved by subdividing the solution space depth-wise for the probabilistic variables to test the range for which they yield valid solutions. In addition to the fundamental advantages of the grid-search method described in the preceding chapter it allows stand-alone functionality within Matlab at the expense of increased computational time, particularly in the case of multiple probabilistic variables.
- Interval narrowing solver module: this is the Java-based module developed as part of the QES2 software framework by Zhou 2003. It utilizes the interval CSP

solution approach described in the background section of this thesis to check validity of the solution space boxes, and returns the narrowed intervals of the probabilistic variables that are consistent with the problem constraints. While this module has yet to be interfaced with the Matlab framework, the proven algorithms can provide accurate narrowed intervals of probabilistic variables without the need to subdivide the solution space depth-wise, as is the case with the grid search method, and thus provides a more efficient solver.

5.2.2.1 Probabilistic Analysis Module

The probabilistic analysis modules take the valid interval data from the solution module and incorporate the data associated with the probabilistic variables. This is done by calling various statistical functions which return the probability of each of the probabilistic variables occurring over the valid interval, based on the type of PDF associated with each of the probabilistic variables. At this point, calculations related to correlations between the probabilistic variables can also be implemented if desired.

5.2.2.2 Output Matrix

The output of the solution module is a matrix containing the full probabilistic data set. The dimensionality and size of the output matrix depends on the type of solver engine used. For the grid-search solver, the matrix dimensionality is the number of design variables plus the number of probabilistic variables. The size of the matrix in dimensions associated with the design variables is the number of cells in the solution space requested along each dimension. The size of the matrix in dimensions associated with the probabilistic variables is the number of subdivisions requested in the grid-search method when subdividing depth-wise along the probabilistic variables.

For the interval narrowing solver, the output matrix dimensionality is equal to the number of design variables plus one. The size of the matrix in dimensions associated with the design variables is the number of cells in the solution space requested along each dimension. The additional dimension contains data on the upper and lower bounds of

each of the probabilistic variables, and thus the size of the additional dimension is two times the number of probabilistic variables.

5.2.3 Post-Processing Module

The post-processing module manipulates the data contained in the full output matrix described above to generate the requested solution space plotting data. This is done through routines that integrate the depth-wise data in the output matrix, which reduces the dimensionality of the output matrix to the solution space plotting dimension. For example, to generate plotting data for the probability of a solution occurring in a given cell, the probabilities associated with each probabilistic variable are multiplied depth-wise through the matrix incorporating any correlations to get the probability associated with that cell. Other possible outputs are probability-weighted functions which return the depth-wise integration of the solution probability multiplied by some objective function value to get the probability-weighted objective function value at each cell. The post-processed data is stored in various matrices which can be called in plotting functions to generate output plots.

The coding takes advantage of Matlab's ability to handle multi-dimensional arrays through built-in functions. For example, the following subroutine sums the probabilities depth-wise through an *n*-dimensional matrix to reduce it to an n_{PLOT}-dimensional matrix that can be then be plotted over the solution space. The subroutine uses the built in *sum* function which sums all the values over a specified dimension of a multi-dimensional array.

```
function matrix_out = sumdepth(matrix_in,ntot,nplot)
      for i=ntot:-1:nplot+1;
            matrix_out=sum(matrix_in,i);
            matrix_in=matrix_out;
      end
```

Similarly, the *max* function can be used as follows:

```
function matrix_out = getmax(matrix_in,ntot,nplot)
      for i=ntot:-1:nplot+1;
            matrix_out=max(matrix_in,[],i);
```

```
            matrix_in=matrix_out;
        end
```

5.2.3.1 Plotting Functions

Various plotting functions are available in Matlab, including 2D contour plots, 3D surface plots, volume slicing, and isosurfaces which isolate a 3D surface containing equal-value data of a specified output parameter. The plotting functions operate on the processed matrixes described above and arrays containing the design variable data related to the plot axes.

5.3 Example Problems

The following basic examples illustrate different features of the theory, software application, results interpretation, and post-processing capabilities.

5.3.1 Example 1: 2D Probabilistic Solution Space

The first example problem plots the solution space for the following inequality using the equation of a line:

$$y \geq mx + b$$

The parameters *m* and *b* are selected to be probabilistic parameters with the following distributions:

- *m* has a symmetric triangular distribution from -2 to 2
- *b* has a symmetric triangular distribution from 2 to 8

The range of the selected solution space for plotting is defined by the following intervals:

$$x = [-5,5]$$

$$y = [0,10]$$

The resulting solution space is plotted in Figure 20 below, with a 20 by 20 resolution selected for the solution space. Parameters *x* and *y* are plotted on the respective axes, and the contour represents the probability of a valid solution at a given location.

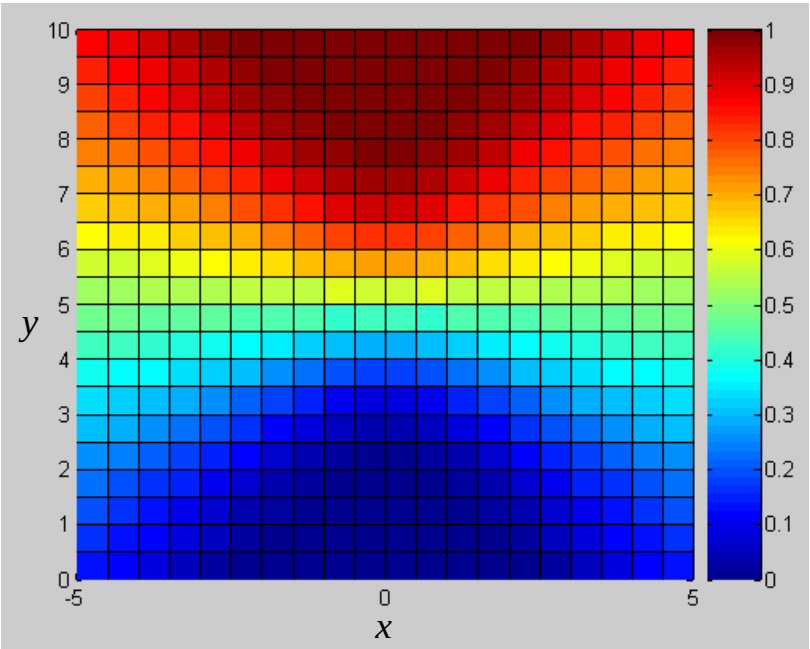

Figure 20: Probabilistic solution space for example 1

In order to aid interpretation of the data, Figure 21 below shows the solution space overlain with the lines plotting the bounding lines corresponding to the maximum and minimum values of the probabilistic parameters *m* and *b*, and indicating the region of 100% and 0% solution probability.

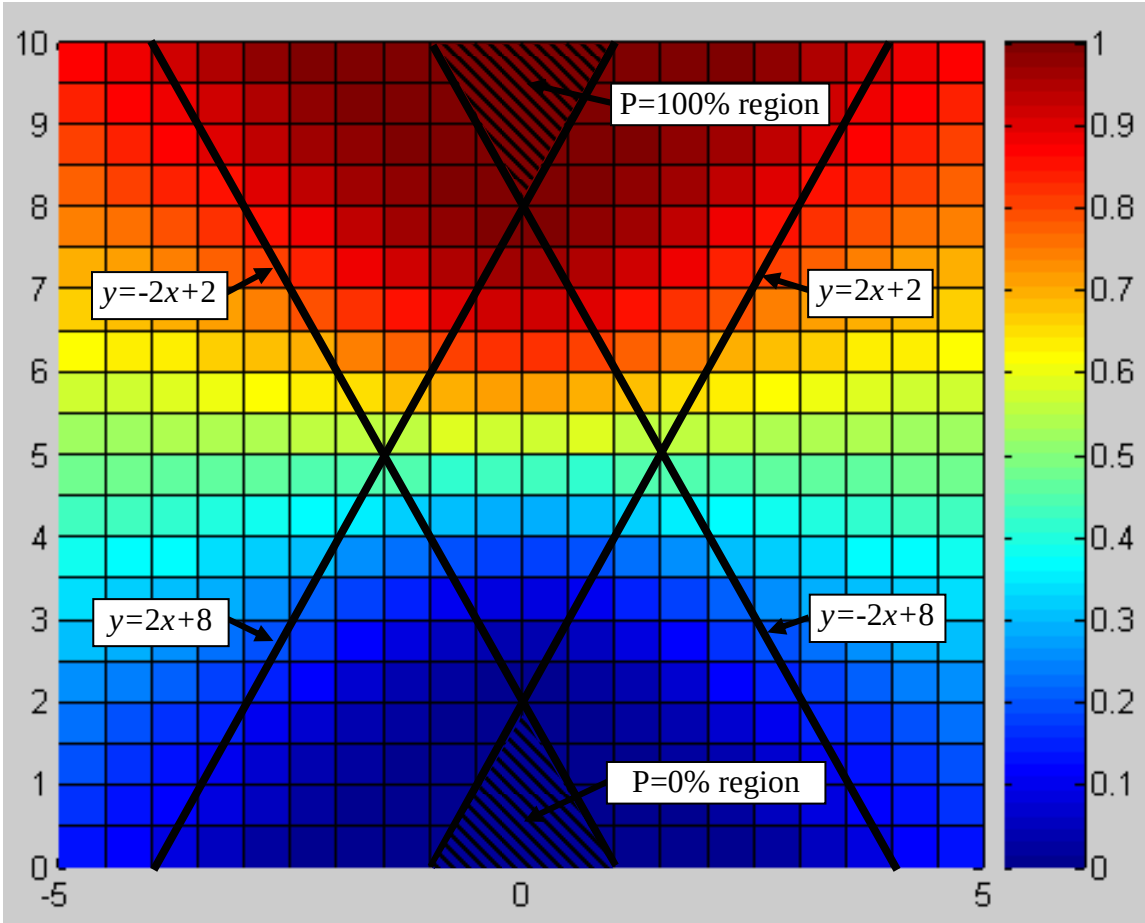

Figure 21: Probabilistic solution space for example 1 with bounding solution space lines overlain.

5.3.2 Example 2: 3D Probabilistic Solution Space

The next example illustrates how a 3D probabilistic solution space may be represented. The following constraint equations represent a hollow sphere:

$$x^2 + y^2 + z^2 \geq r_i^2$$

$$x^2 + y^2 + z^2 \leq r_o^2$$

The parameters r_i and r_o are the inner and outer radii of the hollow sphere, represented by the following probability distributions:

- r_i has a symmetric triangular distribution from 2 to 3.5
- r_o has a symmetric triangular distribution from 3.5 to 5

The probabilistic solution space is plotted over the following range:

$$x = [0,5]$$

$$y = [0,5]$$

$$z = [0,4]$$

The solution space to the hollow sphere contains "holes" that can be visualized by using volume slicing plots, as shown in Figure 22 below. Again, the *x*, *y* and *z* parameters are plotted on their respective axes and the contour represents the probability of a solution existing at a given point.

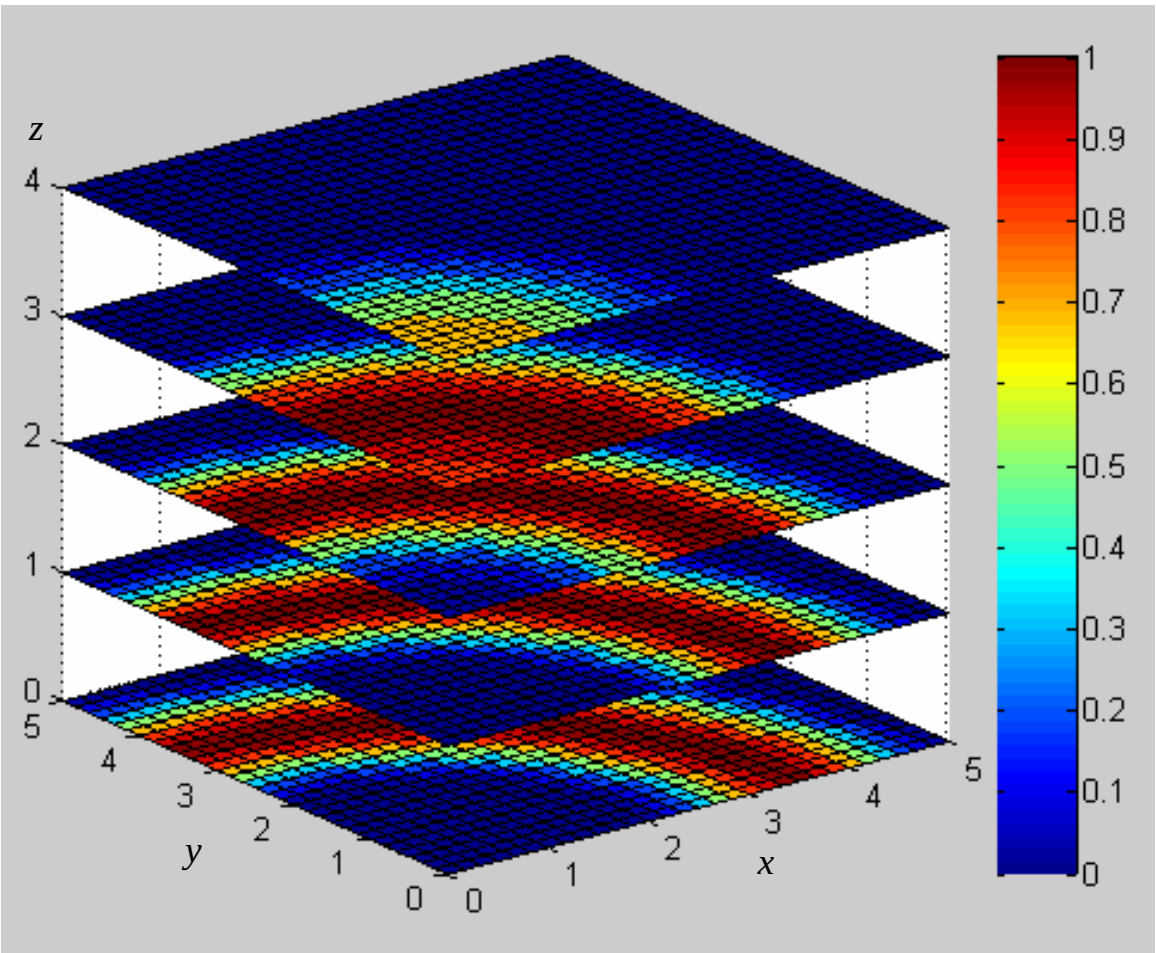

Figure 22: Solution space to example 2, hollow sphere

The resolution of the solution space in Figure 22 is (30,30,5) for the (*x*,*y*,*z*) coordinates. Figure 23 below plots the same solution space at lower resolutions, illustrating that useful data can be obtained from the plots even at significantly lower plot resolutions for initial runs.

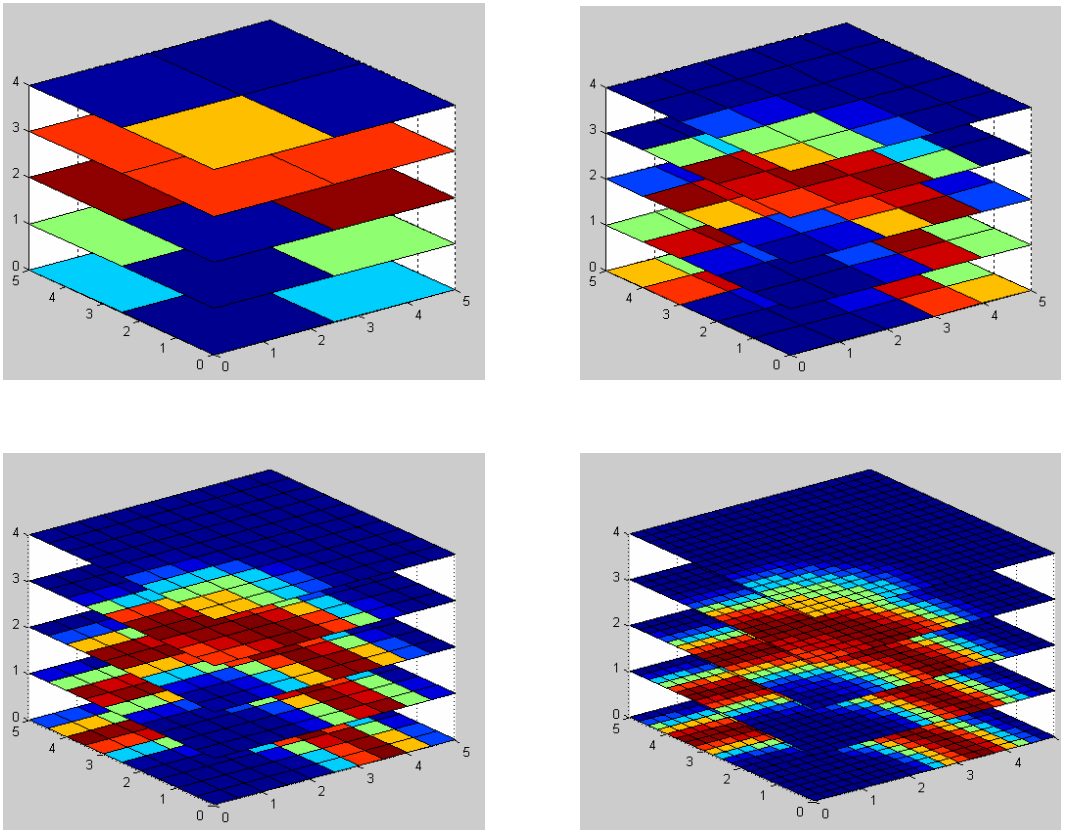

Figure 23: Probabilistic solution space to example 2 for different resolutions

5.3.3 Example 3: 3D Isosurface Plots

The following example illustrates how isosurfaces can be used to plot design space surfaces that have equal probability of success. For this example the following surface equation is used:

$$z \le (a \cdot x^3 + b \cdot x^2 + c \cdot x) \cdot \sin(\pi \cdot y / k)$$

Parameters *a* and *b* are the probabilistic parameters with the following distributions:

- *a* has a symmetric triangular distribution from 0.18 to 0.26

- *b* has a symmetric triangular distribution from -2.5 to -3.5.

Parameters *c* and *k* are set as constants, where *c*=10 and *k*=10. The probabilistic solution space is plotted over the range (*x*,*y*,*z*)=(0:10, 0:10, 0:15) with the result shown in Figure 24.

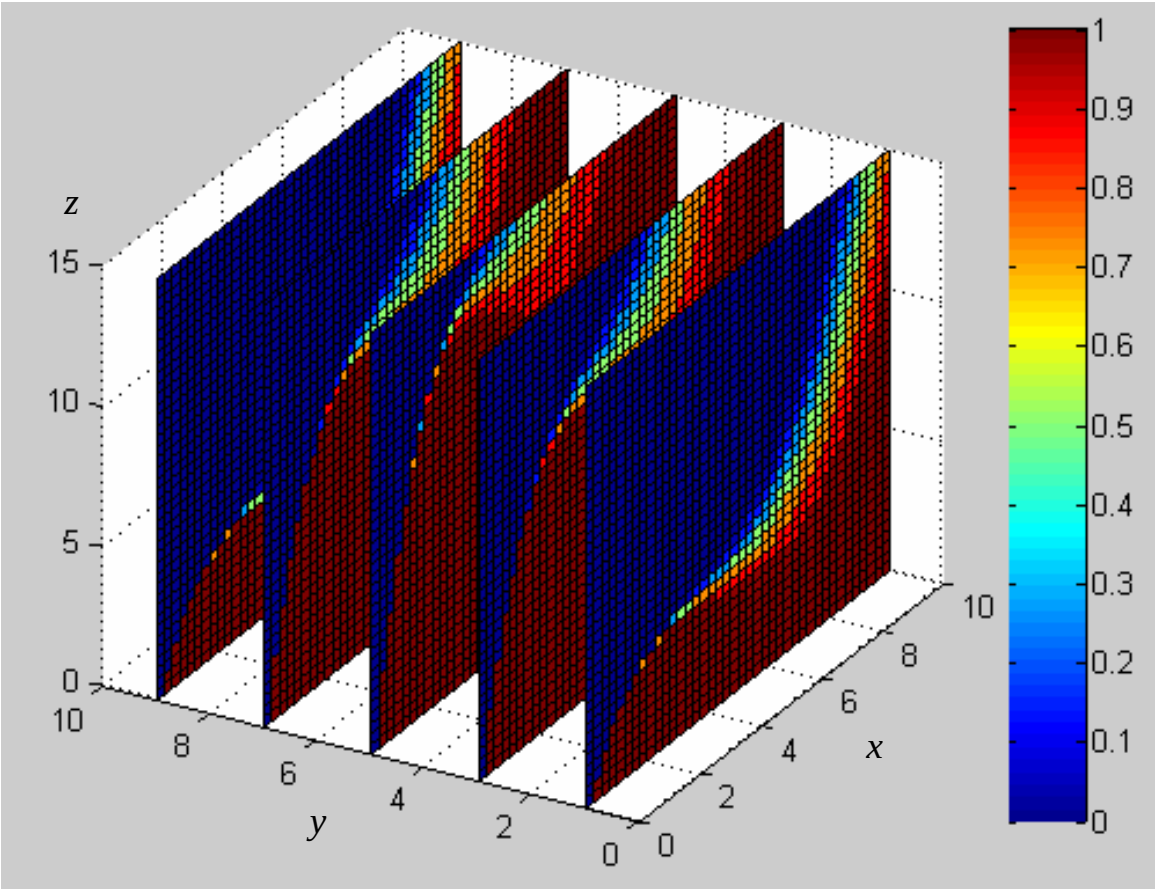

Figure 24: Probabilistic solution space for example 3

The solution space can also be represented by surfaces corresponding to solution points that have the same probability of success. These are referred to as isosurfaces, and are plotted in Figure 25 below for the cases of 10%, 50%, and 90% probability of success.

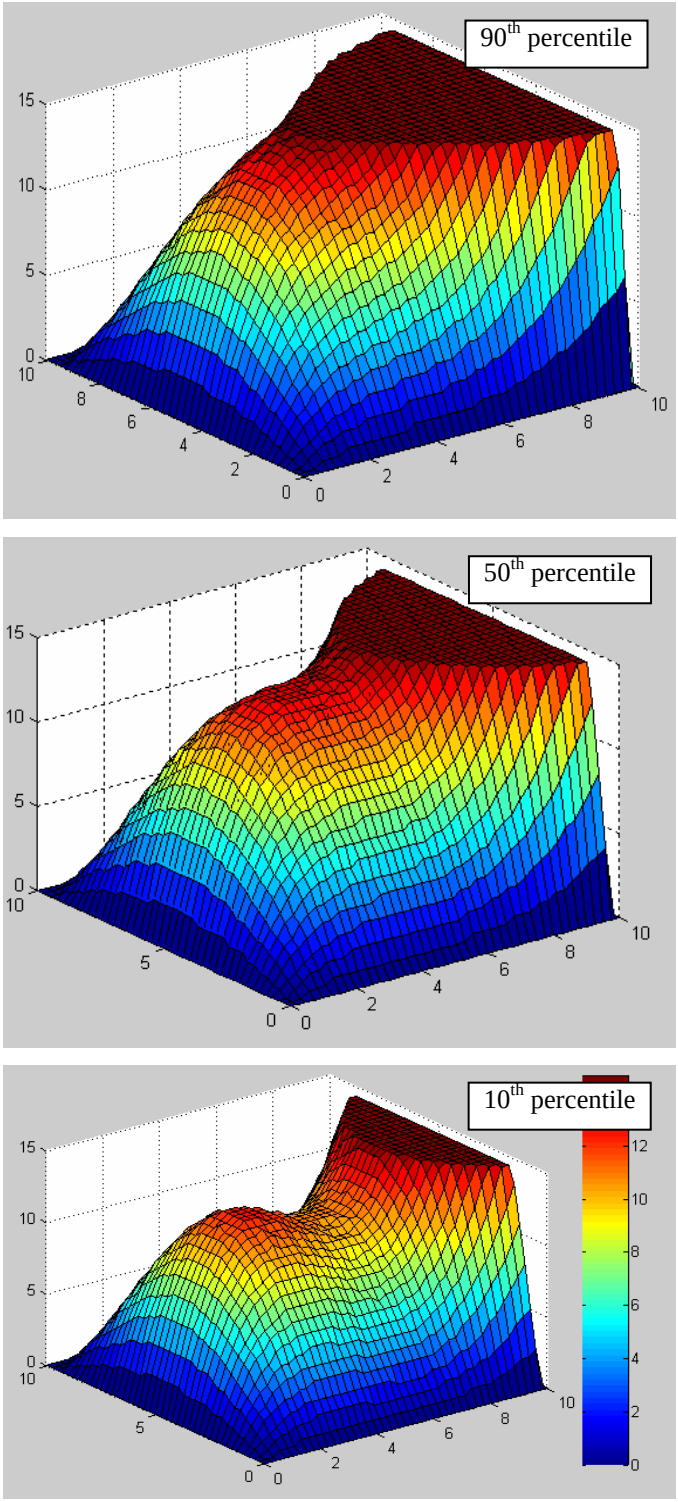

Figure 25: Isosurfaces corresponding to solutions with 90%, 50%, and 10% probability of success; note that the contour colour in this case is equal to the z-value

5.3.4 Example 4: Reliability Solution Space

Similar to example 3, the solution data can be presented in a way that isolates solutions of a given probability of success, or reliability. The following example problem represents the simple stress analysis of a simply supported beam subject to a uniformly distributed load. The problem is expressed below using the concept of a "*g*-function" used in reliability analysis, where $g \leq 0$ defines failures, and $g > 0$ defines success. The constraint equations are written as follows:

$$g > 0$$

$$g = Mr - Mf$$

$$Mr = S \cdot (\phi \cdot f_y) \quad \text{the factored moment resistance}$$

$$Mf = w_f \cdot L^2 / 8 \quad \text{the factored applied moment}$$

where $(\phi \cdot f_y)$ is the factored yield strength, S is the beam section modulus, w_f is the factored distributed load, and L is the beam span. The probabilistic parameters are as follows:

- $(\phi \cdot f_y)$ has a normal PDF with μ =300Mpa and σ =25Mpa
- w_f has a normal PDF with μ =25kN/m and σ =10kN/m

The solution space is plotted in Figure 26 over the range L=[5m,10m] and S=[200x10^{3}mm^3, 1500x10^{3}mm^3].

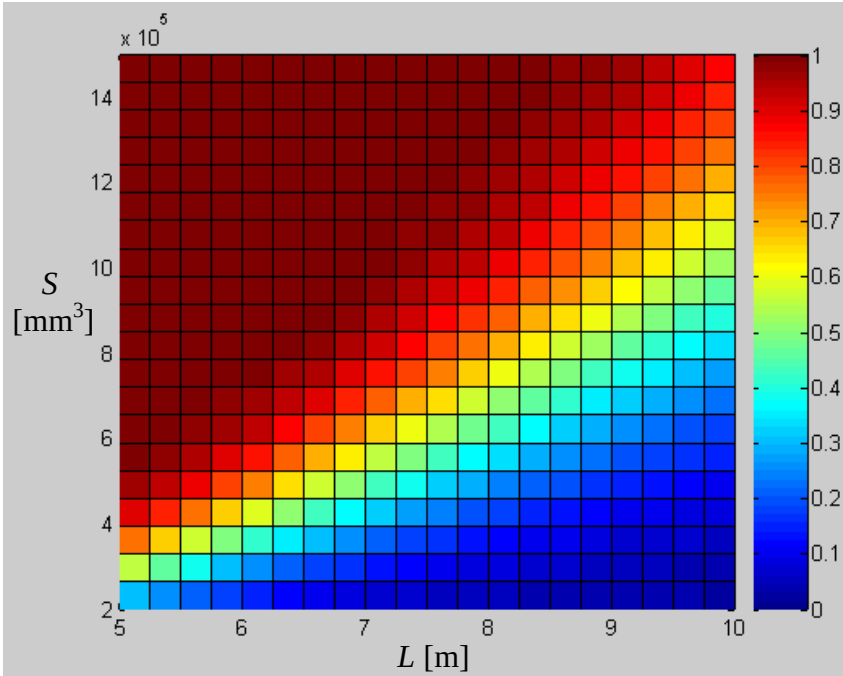

Figure 26: Probabilistic solution space for example 4

For the case of a linear g-function with probabilistic variables with normal distributions, the probability of success can be found analytically as follows:

$$P_s = \Phi(\mu_g / \sigma_g) = \Phi(\beta)$$

where μ_g and σ_g are the mean and standard deviation of the *g*-function, and Φ is the standard normal CDF. The parameter β is referred to as the reliability index, and is often used as a means to quantify the reliability in structural and mechanical analysis. For the above example the solutions that achieve the reliability corresponding to a given value of β can be plotted by varying the scale of the contour on the plot as shown in Figure 27 below.

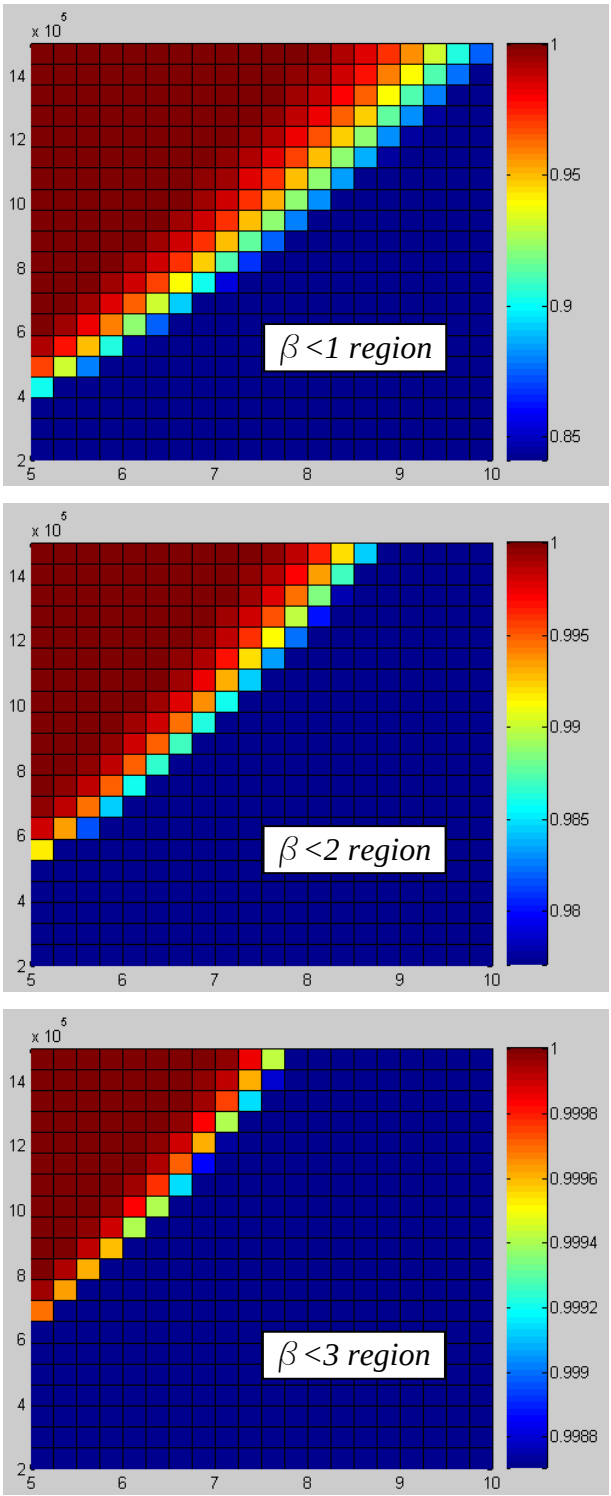

Figure 27: Probabilistic solution space for example 4, with lower contour value cut-off to isolate solution space with reliability of $\Phi(\beta = 1) = 0.841$, $\Phi(\beta = 2) = 0.977$ **and** $\Phi(\beta = 3) = 0.9987$

5.3.5 Example 5: Multi-Span Beam

5.3.5.1 Problem Description

The following example illustrates an application of the software to a relatively simple structural engineering problem: the initial sizing of a roof beam over multiple spans. The basic problem layout and parameters are shown in Figure 28 below.

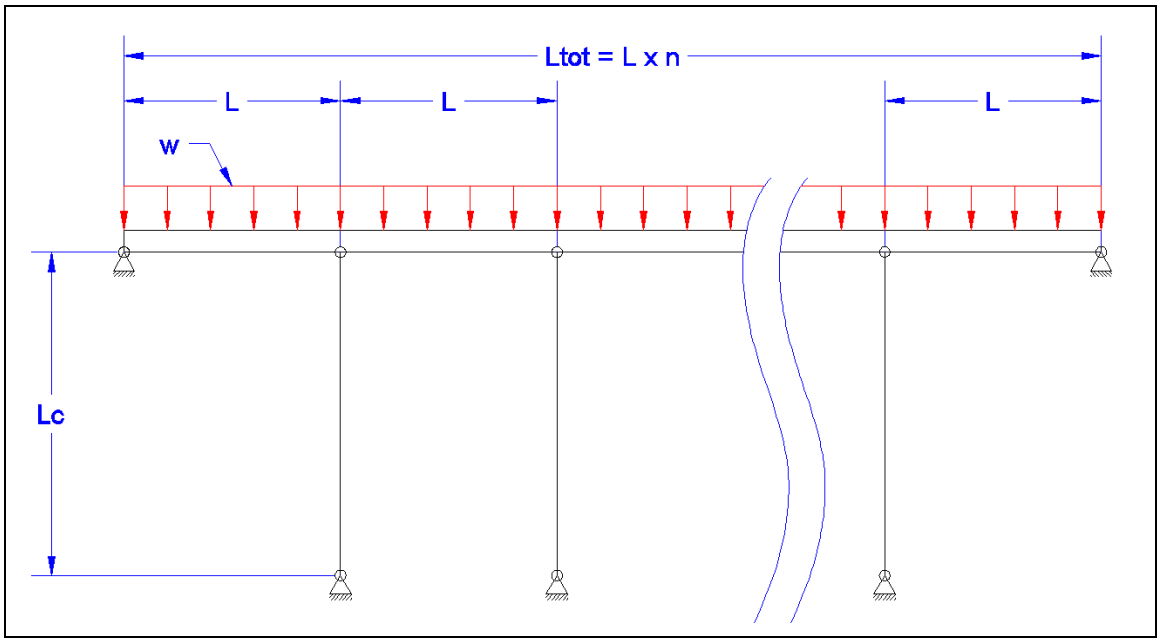

Figure 28: Layout of example problem 5

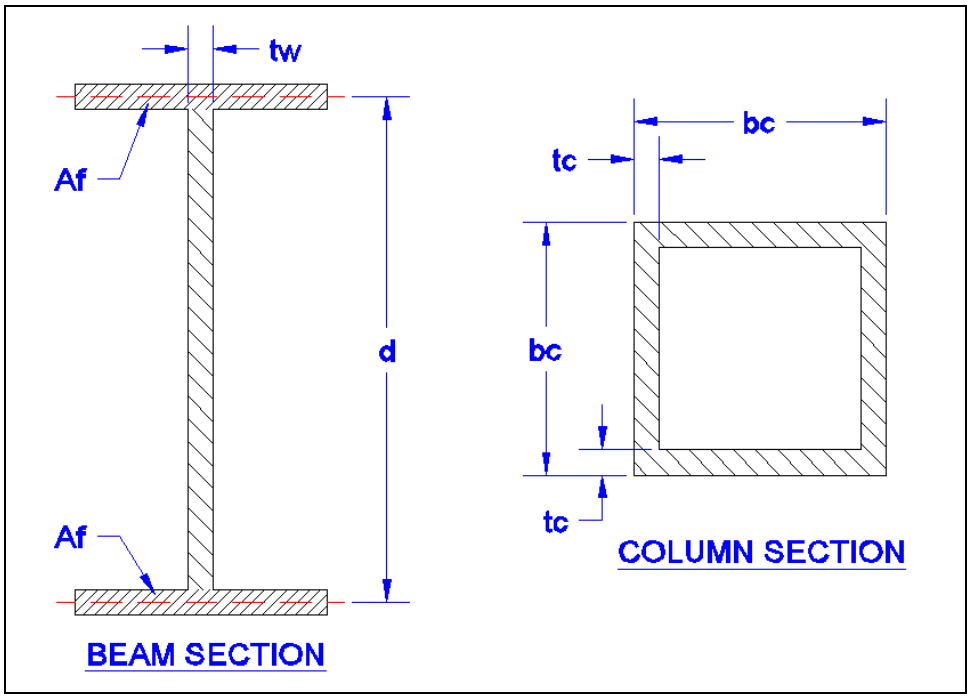

Figure 29: Section properties used in example problem 5

Simplifications have been made to the calculations which would be included in the final engineering analysis. This is done to maintain the simplicity of the problem for the purpose of illustrating the application of the theory, and it is also representative of the rough conceptual-level calculations that a design engineer may carry out during the very first stages of conceptual design.

There are two design variables selected for this problem:

- The span between columns
- The depth of the beam

There are two probabilistic variables selected for this problem:

- The factored distributed load on the roof beam
- The allowable deflection of the roof beam

The major constraints related to the engineering requirements are as follows:

- Bending strength of the beam (factored bending resistance must be greater than factored bending moment)
- Shear strength of the beam (factored shear resistance must be greater than factored shear force)
- Stiffness of the beam (calculated deflection must be less than allowable deflection)
- Strength of the columns (factored compressive resistance must be greater than factored load)

The output parameter of primary interest is the total mass of the structural steel in the beams and columns, which has a direct relationship on the cost of the structural steel. Therefore an additional constraint is added that the resulting mass must be less than a target mass. Such a constraint could be developed by considering the project budget, and using historical cost per unit weight data for the material, fabrication, and erection costs.

Other constraints that may be applied but aren't included in this example problem may include:

- Minimum spacing of columns, which may be required based on architectural requirements
- Maximum depth of beam, which may be required based on architectural requirements or accommodation of mechanical equipment

5.3.5.2 Calculations

The initial structural analysis is based on the Canadian steel design code CAN/CSA-S16.1-01 Limit States Design of Steel Structures (CSA 2001). Approximations are made that are representative of initial conceptual design calculations.

<u>Beam Flange Area</u>

The beam flange area based on strength requirements is calculated as follows:

$$M_R \geq M_F$$

where M_R and M_F are the factored moment resistance and applied moment respectively, and are calculated as follows:

$$M_F = \frac{w_F \cdot L^4}{10} \text{ (for interior spans, end spans to be reinforced)}$$

$$M_R = \phi \cdot F_y \cdot S \text{ (assuming the beam is laterally braced)}$$

where

w_F = factored applied load

ϕ = resistance factor

F_Y = yield strength

S = beam section modulus, approximated by $S = A_F \cdot d$ neglecting the web contribution

The beam flange area based on deflection requirements is found as follows:

$$\Delta \leq \Delta_{ALLOW}$$

where Δ and Δ_{ALLOW} are the calculated and allowable deflection respectively, and Δ is calculated as follows:

$$\Delta = \frac{w \cdot L^4}{384 \cdot E \cdot I} \text{ (for interior spans, end spans to be reinforced)}$$

where

w = unfactored applied load

E = elastic modulus

I = beam moment of inertia, approximated by $I = A_F \cdot d^2 / 2$ neglecting the web contribution

The above equations are easily reconfigured so that the minimum beam flange area can be explicitly calculated for any given values of design parameters (L, d) and probabilistic parameters (w, Δ_{ALLOW}). This is not the case for the calculation of the beam shear

capacity and the column compression capacity which both involve non-linear equations that are described below. In common engineering practice these types of calculations are carried out iteratively by initially selecting a member size, checking it against the code requirements, and then reselecting and testing subsequent member sizes as necessary. Similarly, in the present software application the minimum required member sizes are found by narrowing the interval bounds of the governing member geometry to find a valid solution.

<u>Beam Web Area</u>

The beam web area is calculated based on the shear strength requirements as follows:

$$V_R \geq V_F$$

where V_R and V_F are the factored shear resistance and applied shear respectively. The factored shear resistance is given as:

$$V_R = \phi \cdot A_w \cdot F_S$$

where A_w is the shear area taken as $A_w = t_w \cdot d$ and F_S is given by:

If $\frac{d}{t_w} \leq 439\sqrt{\frac{k_v}{F_y}}$ then $F_S = 0.66 \cdot F_y$

If $439\sqrt{\frac{k_v}{F_y}} < \frac{d}{t_w} \leq 502\sqrt{\frac{k_v}{F_y}}$ then $F_S = F_{CRI}$

If $502\sqrt{\frac{k_v}{F_y}} < \frac{d}{t_w} \leq 621\sqrt{\frac{k_v}{F_y}}$ then $F_S = F_{CRI} + k_a \cdot (0.50 \cdot F_y - 0.866 \cdot F_{CRI})$

If $621\sqrt{\frac{k_v}{F_y}} < \frac{d}{t_w}$ then $F_S = F_{CRE} + k_a \cdot (0.50 \cdot F_y - 0.866 \cdot F_{CRE})$

where F_{CRI} and F_{CRE} are the inelastic and elastic buckling strengths respectively, calculated as follows:

$$F_{CRI} = 290 \cdot \frac{\sqrt{F_y \cdot k_v}}{(d / t_w)}$$

$$F_{CRE} = \frac{180000 \cdot k_v}{(d / t_w)^2}$$

k_v and k_a are the shear buckling coefficient and aspect coefficient respectively, calculated as follows:

$$\text{If} \quad a / d < 1 \qquad \text{then} \qquad k_v = 4 + \frac{5.34}{(a / d)^2}$$

$$\text{If} \quad a / d \geq 1 \qquad \text{then} \qquad k_v = 5.34 + \frac{4}{(a / d)^2}$$

$$k_a = \frac{1}{\sqrt{1 + (a / d)^2}}$$

where a is the distance between web stiffeners.

<u>Column Area</u>

The column area is calculated based on the axial compressive strength requirements as follows:

$$C_R \geq C_F$$

where C_R and C_F are the factored compressive resistance and applied compressive force respectively. The factored compressive resistance is calculated as follows:

$$C_R = \phi \cdot A_C \cdot F_y \cdot (1 + \lambda^{2 \cdot nc})^{-1/nc}$$

$$\lambda = \frac{kL_C}{r_C} \cdot \sqrt{\frac{F_y}{\pi^2 \cdot E}}$$

where

A_C = column area

kL_C = column effective length

r_C = column radius of gyration

nc = compressive strength exponent, taken as 1.34 for fabricated sections or cold formed hollow structural sections

The column cross sectional geometry is shown in Figure 29. For initial calculations, it is assumed that the column width b_C drives the overall column geometry, and that the plate thickness t_C is proportioned to meet class 3 slenderness requirements per CAN/CSA-S16-01:

$$t_C \geq b_C \cdot \frac{\sqrt{F_y}}{670}$$

The column area, moment of inertia, and radius gyration can then be calculated from the single governing parameter b_C as follows:

$$A_C = t_C \cdot (2 \cdot b_C + 2 \cdot (b_C - 2 \cdot t_C))$$

$$I_C = 2 \cdot t_C \cdot b_C \cdot \left(\frac{b_C - t_C}{2}\right)^2 + 2 \cdot \frac{t_C \cdot (b_C - 2 \cdot t_C)^3}{12} + 2 \cdot \frac{b_C \cdot t_C^{\ 3}}{12}$$

$$r_C = \sqrt{I_C / A_C}$$

Figure 30 below provides summarizes the input values, calculations, and output checks in a traditional spreadsheet format that is commonly used by practicing engineers.

INPUT						
DESIGN VARIABLES						
Length of span	L	=			5	[m]
Depth of beam	d	=			500	[mm]
PROBABILISTIC VARIABLES						
Uniformly distributed load	w	=			50	[kN/m]
Allowable deflection ratio (span to deflection)	dall	=			400	[ ]
DESIGN VARIABLES ITERATIVELY SOLVED						
-the following two variables should be minimized, but the explicit calculation is difficult; they are calculated iteratively in the code through interval narrowing						
Web thickness	tw	=			6.35	[mm]
Width of column	bc	=			300	[mm]
CONSTANTS						
Total span	Ltot	=			100	[m]
Effective length of columns	kLc	=			20	[m]
Live load factor	lf	=			1.6	[ ]
Dead load factor	df	=			1.2	[ ]
Factored yield strength	Fy	=			350	[Mpa]
Resistance factor	phi	=			0.9	[ ]
Elastic modulus	E	=			200000	[Mpa]
Density of steel	p	=			7850	[kg/m^3]
CALCULATIONS						
BEAM MOMENT CAPACITY						
-assume flanges are laterally braced, calculate required flange area explicitly						
-dead load is small compared to live load and neglected						
Factored moment	Mf	=	(w*lf)*L^2/10	=	200	[kN-m]
Required flange area for strength requirements	Afb	=	Mf/(phi*Fy*d)*1e6	=	1270	[mm^2]
Required flange area for deflection requiremen	Afd	=	(w*L^4/384)/(E*d^2/2)*(dall/L)*1e9	=	260	[mm^2]
Required flange area	Af	=	max(Afb,Afd)	=	1270	[mm^2]
BEAM SHEAR CAPACITY						
Factored shear load	Vf	=	(w*lf)*L/2	=	200	[kN]
Shear area	Aw	=	d*tw	=	3175	[mm^2]
Distance between stiffeners	a	=	2*d	=	1000	[mm]
Shear buckling coefficient	kv	=	if(a/d<1,4+5.34/(a/d)^2,5.34+4/(a/d)^2)	=	6.34	[ ]
Aspect coefficient	ka	=	1/sqrt(1+(a/d)^2)	=	0.45	[ ]
Critical stress inelastic	Fcri	=	290*sqrt(Fy*kv)/(d/tw)	=	173	[Mpa]
Critical stress elastic	Fcre	=	180000*kv/(d/tw)^2	=	184	[Mpa]
Shear strength	Fs	=	if(d/tw<=439*sqrt(kv/Fy),0.66*Fy,if(and(d/tw>439*sqrt(kv/Fy),d/tw<=502*sqrt(kv/Fy)),Fcri,if(and(d/tw>502*sqrt(kv/Fy),d/tw<=621*sqrt(kv/Fy)),Fcri+ka*(0.50*Fy-0.866*Fcri),if(d/tw>621*sqrt(kv/Fy),Fcri+ka*(0.50*Fy-0.866*Fcre),"ERR"))))	=	185	[Mpa]
Factored shear resistance	Vr	=	phi*Aw*Fs/1000	=	527	[kN]
Shear utilization	ut_v	=	Vf/Vr	=	**0.38**	[ ]
COLUMN COMPRESSION CAPACITY						
-assume dead load small compared to live load						
Factored compression load	Cf	=	(w*lf)*L	=	400	[kN]
-assume square column of width bc, *and width/thickness ratio for column at class 3 limit*						
Thickness of column (to meet class 3 limit)	tc	=	bc*sqrt(Fy)/670	=	8.4	[mm]
Column area	Ac	=	tc*(2*bc+2*(bc-2*tc))	=	9772	[mm]
Column moment of inertia	Ic	=	2*tc*bc*((bc-2*tc)/2)^2+2*tc*(bc-2*tc)^3/12+2*bc*tc^3/12	=	1.3E+08	[mm^4]
Column radius of gyration	rc	=	sqrt(Ic/Ac)	=	116	[mm]
Compressive strength exponent	nc	=			1.34	[ ]
Compressive strength	CrA	=	phi*Fy*(1+(kLc*1000/rc*sqrt(Fy/(pi()^2*E)))^(2*nc))^(-1/nc)	=	56	[Mpa]
Factored compression resistance	Cr	=	CrA*Ac/1000	=	545	[kN]
Compression utilization	ut_c	=	Cf/Cr	=	**0.73**	[ ]
MASS SUMMARY						
Number of spans	ns	=	Ltot/L	=	20	[ ]
Flange mass	mf	=	2*Af*Ltot*p/1e9	=	2.0	[tonne]
Web mass	mw	=	Aw*Ltot*p/1e9	=	2.5	[tonne]
Column mass	mc	=	(ns+1)*Ac*kLc*p/1e9	=	32.2	[tonne]
Total mass	m	=	mf+mw+mc	=	**36.7**	[tonne]

Figure 30: Sample calculations using spreadsheet calculation method

5.3.5.3 Analysis Methodology

The input parameters (design variables, probabilistic variables, constants) and calculations as described above are coded into the input module. For the design

variables, the bounding values and resolution of the output plotting space are defined as follows:

```
% Design variable data [low value, high value, no. boxes]
L=[2,25,20];
d=[50,3000,20];
```

For the probabilistic variables, the probabilistic parameters must be entered. For the example problem, triangular PDFs are defined as follows:

```
% Probabilistic variable data
% For triangular PDF: [low value, high value, no. boxes]
w=[25,50,16];
dall=[300,600,16];
```

Note that the third entry indicated the plot resolution (number of boxes) is only required if the grid-search solution module is used. If the interval CSP solution module is used, no manual subdivision of the probabilistic variables is required since the interval CSP method iteratively narrows the valid interval of the probabilistic variables prior to integrating the PDF over the valid interval range.

The coding of the equations into the input module is similarly straightforward, and follows much the same coding that would be used in a typical spreadsheet calculation such as that shown in Figure 30 above. For example, the following sample code below calculates the total mass using the same notation as the spreadsheet calculation, and then applies a test to see if the instantiated variables provide a solution that meets the required target mass.

```
% Total mass
mf=2*Af*Ltot*p/1e9;     % mass of beam flanges
mw=Aw*Ltot*p/1e9;       % mass of beam web
mc=(ns+1)*Ac*kLc*p/1e9; % mass of columns
m=mf+mw+mc;             % total mass
mtarget=30;             % target mass
if m<=mtarget;
      truth=1;          % parameter to test solution validity
else
      truth=0;
end
```

5.3.5.4 Results

The primary result of the calculation process is a contour plot of the probability of a successful design over the design space, which is plotted in Figure 31 below. The two axes represent the design variables: the x-axis is the beam depth *d* [mm], and the y-axis is the beam span *L* [m]. The contour represents the probability of achieving a design in a given region, namely that the strength and stiffness requirements are met, and that the target mass is achieved.

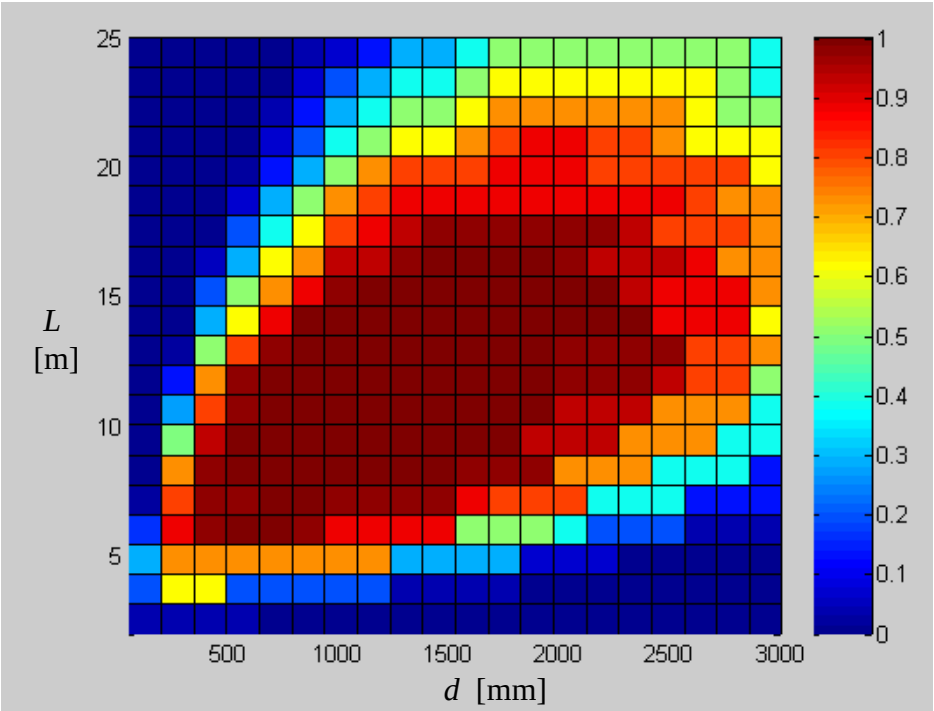

Figure 31: Probability of meeting weight budget with target weight = 30 tonnes

The results are plotted again for various output resolutions in Figure 32 below. These plots indicate how useful design information can be taken from lower resolution plots and then subsequent analyses can increase the plotting resolution or reselect the design space of interest.

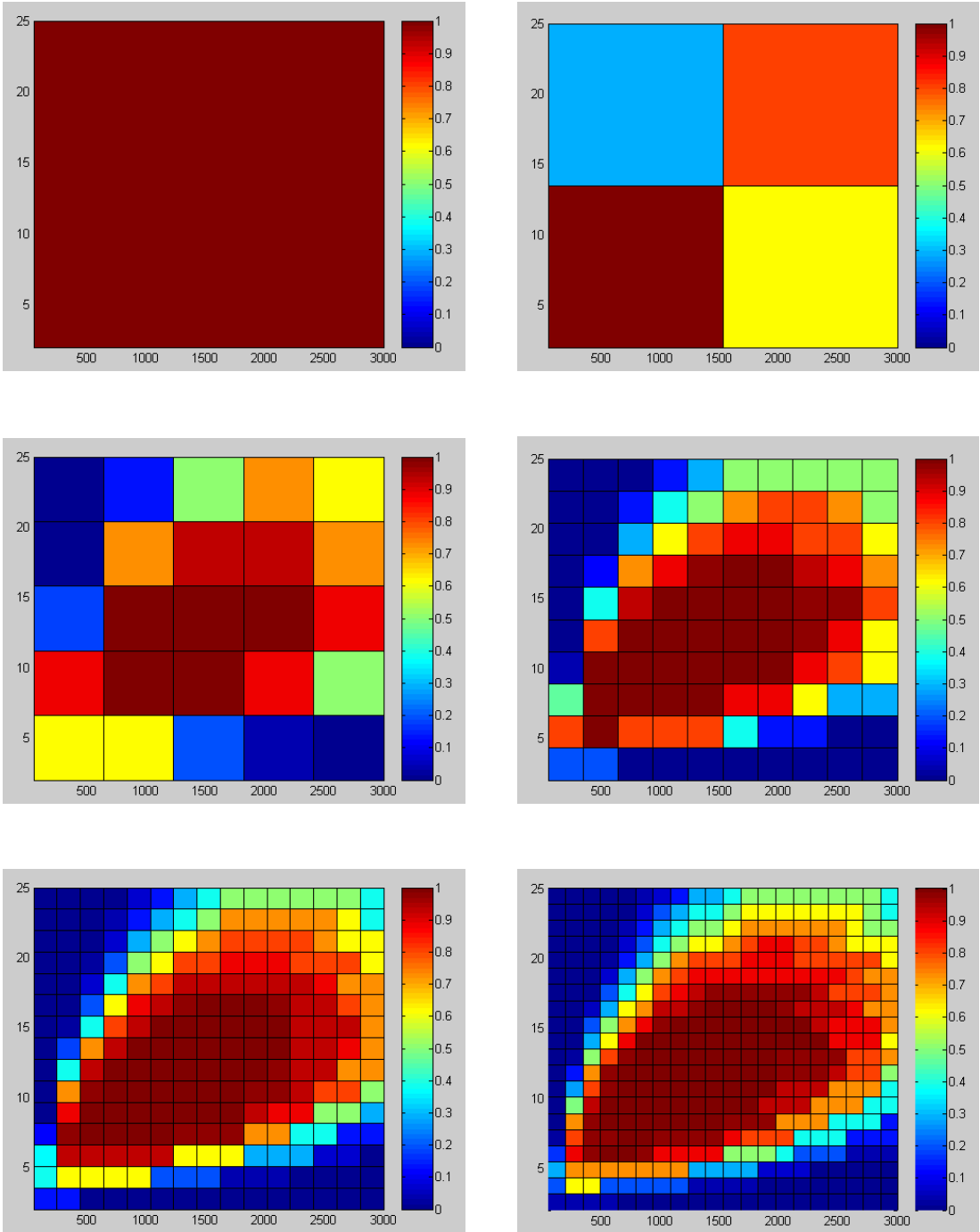

Figure 32: Probability of meeting weight budget with target weight = 30 tonnes with results plotted at resolutions of 1x1, 2x2, 5x5, 10x10, 15x15, and 20x20.

In addition to the main results described above there are many additional results that are available as a result of the solution process that may be of interest to the designer. In general, these require no additional calculation or processing time, only that the designer must specify that the required results are recorded during the solution process as they are calculated, so that they can be loaded for use during post-processing. For example, the designer would likely be interested not only in meeting the mass target, but also in minimizing the mass of the design. This data can be extracted from the analysis in several ways. The first would be to look at the distributions of minimum and maximum masses that were achieved over the design space. These distributions correspond to the calculated mass of the design which meets all of the structural requirements with regards to strength and stiffness, calculated with the values of the probabilistic variables at the most favorable and least favorable values in their distributions. These minimum and maximum mass distributions are plotted in Figure 33 below.

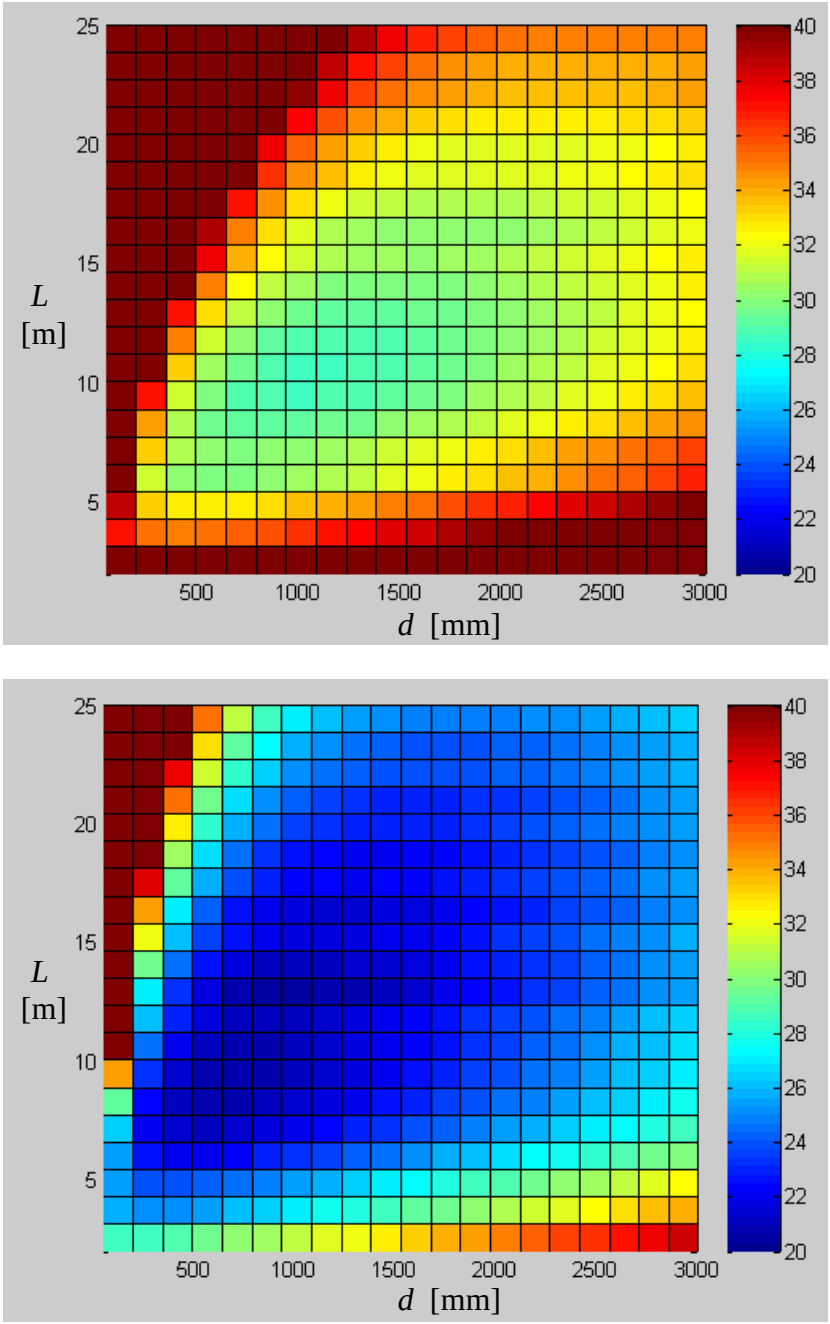

Figure 33: Maximum (top) and minimum (bottom) mass distributions

A second way of reviewing the mass data is to plot the probability-weighted mass distribution. This corresponds to integrating depth-wise into the plot (i.e. into the higher dimensions of the results matrix) the probability of the probabilistic variables over a given interval multiplied by the resulting mass over that interval, and yields the

distribution of the mean mass over the design space. This is plotted in Figure 34 below for the example problem.

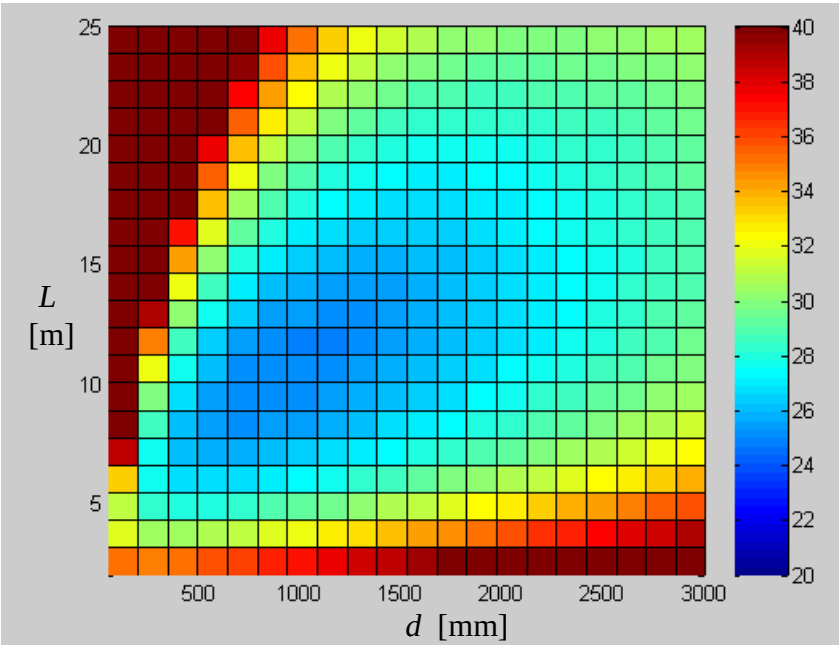

Figure 34: Probability-weighted mass distribution

Figure 35 plots the probability-weighted mass broken down for the major structural components: the beam flanges, the beam web, and the columns.

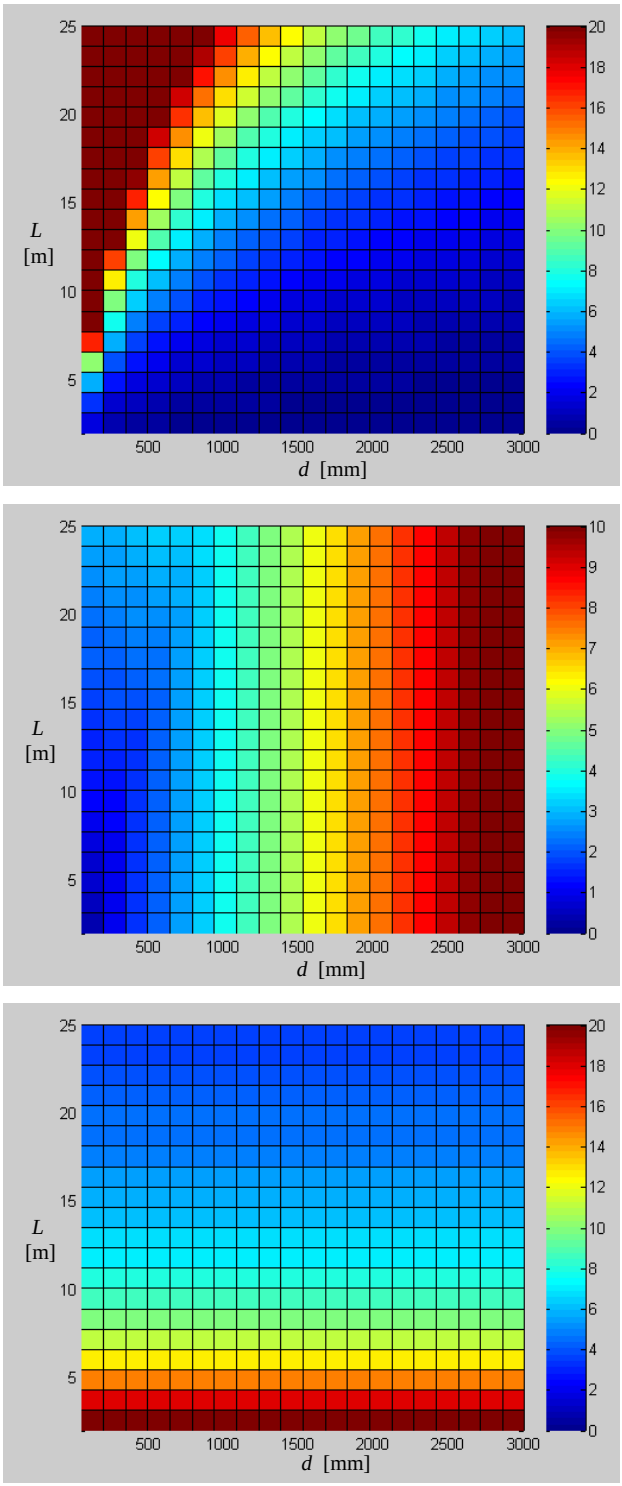

Figure 35: Probability-weighted mass distribution for flange mass (top), web mass (center), and column mass (bottom)

Also of interest to the designer are the values of the probabilistic variables for which a valid solution was found in the design space. For example, Figure 36 below plots the maximum applied load for which a valid solution was found. The solutions found in the lower portion of the design space were in general capable of carrying the maximum load value assigned to the probabilistic load variable, whereas the longer span solutions only met the constraints for lower values of the applied load.

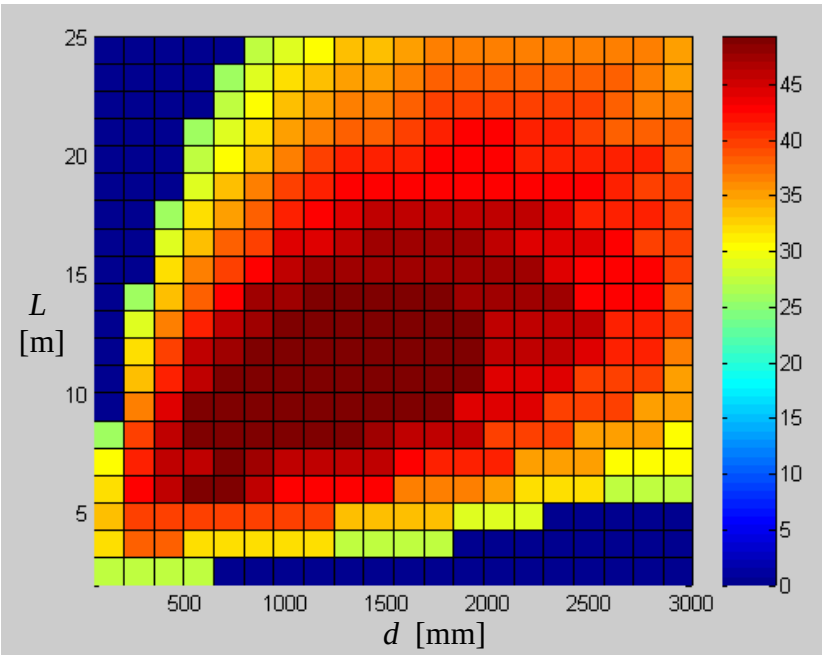

Figure 36: Maximum calculated load capacity [kN/m] for designs which met all constraints, including target weight = 30 tonnes

6 CASE STUDY

The following case study is presented to illustrate the application of the theory and software developed as part of this thesis to an industrial research project. The project involves the conceptual design of a novel concept for a very large array of radio antennae to be used for astronomical research. The case study is representative of conceptual design problems with large continuous design parameter domains, uncertain design parameters including performance targets and costing data, and very limited existing experience from similar projects to draw from.

6.1 Introduction

The Large Adaptive Reflector (LAR) is a novel concept for a radio telescope design proposed by the Dominion Radio and Astrophysical Observatory (DRAO) in Penticton, BC (Carlson et al 2000). The LAR concept was originally proposed as a candidate antenna solution for the Square Kilometer Array (SKA) - an ongoing international astronomy project seeking to build an array of radio antennas with a combined collecting area of one square kilometer.

The goal of the LAR concept is to provide an antenna design with a low cost per unit collecting area, enabling the development of radio antenna arrays with very large collecting areas. There are two key aspects of the design:

1. An **adaptive reflector structure** that can be configured to a wide continuous range of parabolic shapes that focus radio signals from any point in the sky
2. An **airborne receiver** that can be positioned to the focal point of the radio signal reflected from the adaptive reflector structure.

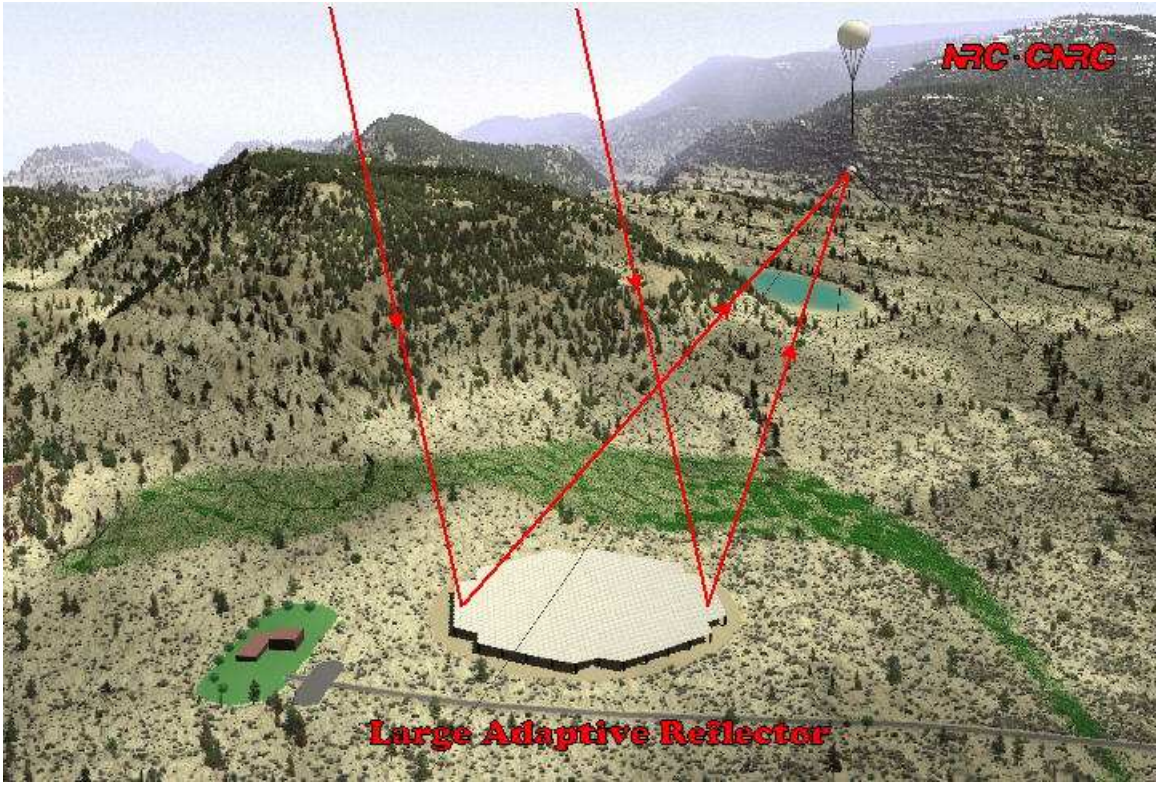

Figure 37: LAR concept overview showing ground based adaptive reflector structure and airborne receiver

6.1.1 Reflector Panel Design

The surface of the reflector is given by the following equation:

$$z = \frac{x^2 \cdot \cos\alpha + \dfrac{y^2}{\cos\alpha}}{4 \cdot R \cdot \left[1 + x \cdot \left(\dfrac{\sin\alpha}{2 \cdot R}\right)\right]}$$

R is the focal length of the telescope (fixed), α is the zenith angle of the observing point, (i.e. the angle of the observing point away from vertical), x and y are the coordinates on the reflector surface that are parallel and orthogonal to the ray path respectively, and z is the required vertical displacement of the reflector surface above a reference plane.

The reflector is adaptive in that it must reconfigure to this surface height for various observing points. This is achieved through an articulating structure that supported on a grid-work of actuators which raise and lower the individual panels. The stroke of the actuators must be sufficient to meet the required surface configurations.

Conceptual designs were developed for the reflector structure including a fiber reinforced concrete design (Kuerschner 1999) and a steel design (Loewen 2003). A prototype section of the reflector structure was constructed in 2004 that consisted of two steel reflector panels, a secondary supporting structure, and three hydraulically driven actuators.

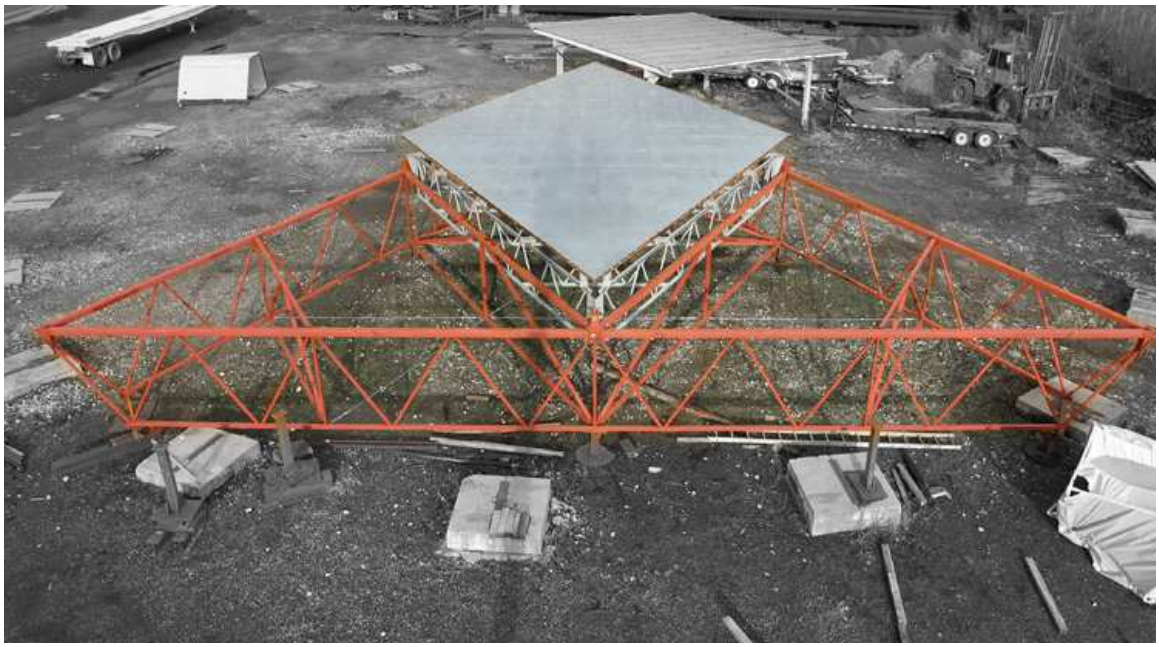

Figure 38: Prototype reflector structure with two triangular reflector panels (silver, 10m side length) and one secondary supporting structure (red, 20m side length)

Figure 39: Prototype hydraulically driven actuator with 6m stroke

Figure 40: Prototype reflector structure supported on three actuators

The actual reflector structure would be composed of a circular area that is tiled with triangular panel structures such as those that were prototyped. An example tiling pattern is shown in the figure below.

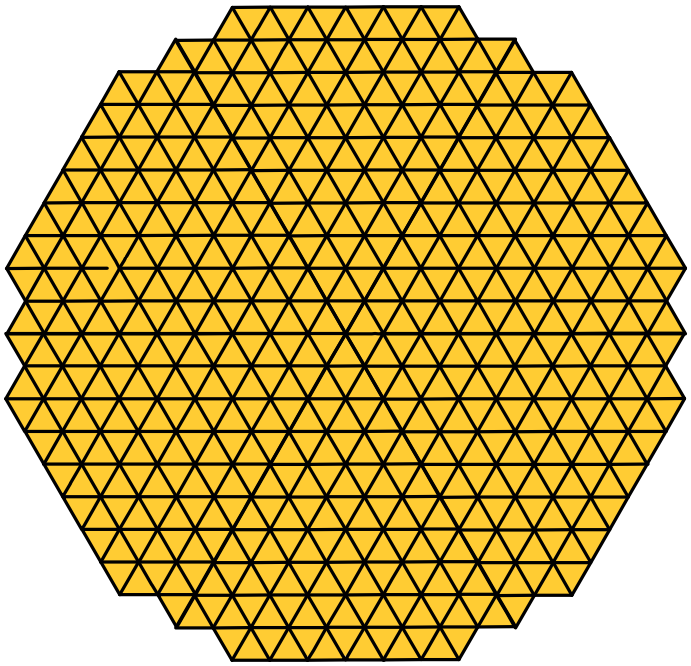

Figure 41: A 350m diameter reflector structure tiled with 20m triangular support structures

6.1.2 Receiver Design

The airborne receiver design consists of a receiver unit that is suspended by a helium-filled aerostat. The receiver unit position is controlled via six tether lines running to ground-based winches that are distributed around the reflector structure.

A prototype of the airborne receiver unit was developed by DRAO to demonstrate the stability and controllability of the system.

Figure 42: Prototype aerostat system shown in launching configuration

Figure 43: Prototype aerostat system airborne with one of the six control tether lines shown in the foreground

6.2 Problem Description

The prototype LAR components provide a baseline design solution and present the opportunity for optimization of the major design parameters for a full-scale antenna system in order to minimize the cost per unit area. The prototyping cost data provides a useful data set for extrapolating costs of the overall system while looking at varying the major design parameters.

Three of the major tradeoffs in optimizing the concept selected for further investigation in the case study are as follows:

- **Size of reflector structure**: since the total sum of the reflector area for the SKA must be one square kilometer there is a tradeoff between the size of the individual reflectors and the quantity of reflectors. If a larger reflector structure is selected, the total number of reflectors decreases resulting in a reduced total number of airborne receivers. However, as the reflector size increases the range of the required surface shapes at the perimeter of the reflector become more extreme and thus the actuators require a longer stroke on average.
- **Size of panels**: the selected size of the individual reflector panels involves a tradeoff between a fewer number of large panels versus a greater number of small panels. Since actuators are required at the panel vertices the larger panels result in a lower number of actuators. The panel structure must also be stiff enough to retain a specified surface accuracy under wind load. As panels become larger and span longer distances between the supporting actuators, they must become heavier in order to maintain the required stiffness and surface form. Therefore, smaller panels will result in a lighter overall structure, and the total load on the actuators will decrease.
- **Surface accuracy:** the surface accuracy of the reflector panels is one of the major factors governing the astronomy performance of the telescope. Higher surface accuracy enables the reflector to focus radio signals at higher wavelengths thus increasing the range of data that can be collected. However, there is a tradeoff

between required surface accuracy and structural stiffness, which in turn influences the structural weight and the load on the actuators.

The three design variables selected which represent the major tradeoffs described above are as follows:

D = diameter of reflector structure

L = size of reflector panels (side length of equilateral triangle)

Δ= allowable deflection of the panel surface under operating conditions

The methodology for estimating the cost of the major antenna components, namely the reflector panels, the actuators, and the receivers, as a function of these primary variables is described below. Note that the costing data shown is for exemplary purposes only, and does not reflect actual current industrial costs.

6.2.1 Panel Cost

The panel cost is estimated by first scaling the mass of the prototype panel design. The total mass per unit area of the reflector panels is as follows:

$$m_{PANEL} = m_{TRUSS} + m_{SURF}$$

where m_{TRUSS} and m_{SURF} are the unit masses of the panel trusses and panel surface materials per unit area, respectively. The panel surface mass per unit area is assumed to be constant, regardless of panel size. The truss unit mass can be scaled from the prototype mass as follows:

$$m_{TRUSS}(L,\Delta) = m_{TRUSS_B} \cdot \left(\frac{L}{L_B}\right)^2 \cdot \frac{\Delta_B}{\Delta}$$

where m_{TRUSS_B}, L_B and Λ_B are the properties associated with the baseline prototype design. This scaling law is based on the assumption that the design scales geometrically, thus maintaining a constant span to depth ratio L/d of the trusses, and that the design of

the trusses is governed by the maximum allowed deflection under performance conditions. This can be expressed via the following proportionalities:

$\Delta \propto w \cdot L^4 / I$	where w is load per unit length of truss
$w \propto L$	since the tributary area of each truss increases with L
$d \propto L$	where d is the truss depth which scales proportionally to L based on geometric scaling assumption
$I \propto A \cdot d^2$	where I is the truss moment of inertia and A is the truss chord area
$m \propto A \cdot L / L^2$	where m is the truss mass per unit panel area

Combining the above yields the following:

$$\Delta \propto L^5 / (A \cdot L^2) \propto L^3 / A \qquad \text{rearranged as} \quad A \propto L^3 / \Delta$$

$$m \propto L^2 / \Delta$$

The last expression demonstrates the proportionality used in the equation for scaling the mass of the panel trusses based on the prototype mass. The cost per unit area of the panel is then calculated as follows

$$C_{PANEL} = c_{TRUSS} \cdot m_{TRUSS} + C_{SURF}$$

where c_{TRUSS} is the cost per unit mass of the truss (including materials, fabrication, and erection) and C_{SURF} is the cost per unit area of the surface materials.

6.2.2 Actuator Cost

The cost of each actuator is again scaled from the prototype actuator costs:

$$C_{ACT_EA} = \left[C_{ACT_0} + \frac{dC}{dS} \cdot (S_{ACT} - S_{ACT_B}) \right] \cdot \left(\frac{\phi}{\phi_B} \right)^{\phi_{EXP}}$$

where C_{ACT_0} is the baseline prototype actuator costs, dC/dS is the estimated additional cost per unit stroke of actuator, and S_{ACT} and S_{ACT_B} are the strokes of the scaled actuator and baseline actuator designs. The required actuator stroke is calculated as follows:

$$S_{ACT} = D^2 \cdot \frac{\frac{1}{\cos(\alpha_{MAX})} \cdot \left[1 + \frac{D \cdot \sin(\alpha_{MAX})}{4 \cdot R}\right] - \cos(\alpha_{MAX})}{4 \cdot R \cdot \left[1 + \frac{D \cdot \sin(\alpha_{MAX})}{4 \cdot R}\right]}$$

where R is the focal length of the telescope, α_{MIN} is the minimum zenith angle required for astronomical observations, and D is the reflector diameter.

ϕ and ϕ_B are the diameters of the hydraulic actuation cylinders for the scaled and baseline designs respectively. The cost of the actuators scales as a function of the diameters to a cost scaling exponent ϕ_{EXP}. The diameter of the cylinders is governed by either required cylinder area needed to generate the required force from hydraulic pressure, or by the required cylinder diameter to prevent buckling, typically taken as 1/40th of the actuator stroke. The scaled cylinder diameter can therefore be represented by the following equation:

$$\phi = MAX\left(\phi_B \cdot \sqrt{\frac{P}{P_B}}, S/40\right)$$

where P and P_B are the actuator load due to the weight of the panels in the scaled and baseline design. The actuator load is calculated as follows given that each actuator has a tributary area of two panels since it supports a third of the weight of each of the six panels located at the vertices:

$$P = m_{PANEL} \cdot (\sqrt{3} \cdot L^2 / 2)$$

The cost of the actuators per unit reflector area is calculated as follows:

$$C_{ACT} = C_{ACT_EA} / (\sqrt{3} \cdot L^2 / 2)$$

6.2.3 Receiver Cost

The receiver cost per unit area is calculated as follows:

$$C_{REC} = C_{REC_EA} \cdot n_{REC} / A_{TOT}$$

where C_{REC_EA} is the cost per receiver, A_{TOT} is the total area of the antenna array (i.e. one square kilometer) and n_{REC} is the number of receivers as given by the following:

$$n_{REC} = \frac{A_{TOT}}{A_{RECEIVER}} = \frac{A_{TOT}}{\pi \cdot D^2 / 4}$$

6.2.4 Total Cost

Finally, the total cost of the antenna per unit area is given by the following equation:

$$C_{TOT} = C_{PANEL} + C_{ACT} + C_{REC}$$

where C_{PANEL}, C_{ACT} and C_{REC} are the costs per unit collecting area for the reflector panels, actuators, and receiver respectively. For this case study, it is assumed that there is a target cost per unit area which must be met in order for a design to be considered feasible, and this is expressed by the following constraint:

$$C_{TOT} < C_{TARGET}$$

The calculations are summarized in the figure below using a traditional spreadsheet format, summarizing the selected equations and parameter values.

INPUT							
DESIGN VARIABLES							
	Reflector diameter	D	=			300	[m]
	Panel side length	L	=			20.0	[m]
	Allowable operating deflection	del	=			0.5	[mm]
PROBABILISTIC VARIABLES							
	Panel truss costs (material & fab) per unit weight	ctruss	=		=	6.00	[$/kg]
	Cost per unit stroke	dCdS	=			15,000	[$/m]
	Scaling exponent for actuator diameter	Dexp	=			2.00	[ ]
	Cost of receiver per unit	Crec_ea	=			17,500,000	[$]
PROTOTYPE VARIABLES							
	Reflector diameter - prototype	D_b	=			300.0	[m]
	Panel side length - prototype	L_b	=			20.0	[m]
	Allowable operating deflection - prototype	del_b	=			0.5	[mm]
PROTOTYPE PANEL DATA							
	Mass of panel trusses per unit area - prototype	mtruss_b	=			90	[kg/m^2]
	Mass of surface materials per unit area	msurf	=			16	[kg/m^2]
	Panel surface cost per unit area	Csurf	=			150	[$/m^2]
PROTOTYPE ACTUATOR DATA							
	Actuator cost - prototype	Cact_b	=			100,000	[$]
	Actuator stroke - prototype	Sact_b	=	(D_b/2)^2*(1/cos(zmax)*(1+D_b*sin(zmax)/(4*R_b))-cos(zmax))/(4*R_b*(1+D_b*sin(zmax)/(4*R_b)))	=	11.5	[m]
	Actuator load - prototype	Pact_b	=	(msurf+mtruss_b)*(sqrt(3)*L_b^2/2)*9.81	=	359.2	[kN]
	Actuator diameter - prototype	Dact_b	=			350	[mm]
CONSTANTS							
	Focal ratio	ratio	=			2.5	[ ]
	Focal length	R	=	ratio*D	=	750	[m]
	Focal length - prototype	R_b	=	ratio*D_b	=	750	[m]
	Total area of array	Atot	=			1.00E+06	[m^2]
	Maximum zenith angle	zmax	=	(3.1416/180)*60	=	1.05	[rad]
CALCULATIONS							
	Area of reflector	Ar	=	pi()*D^2/4	=	70686	[m^2]
	Number of receivers required	nr	=	Atot/Ar	=	14.1	[ ]
PANEL MASS & COST CALCULATION							
	Mass of panel trusses per unit area	mtruss	=	mtruss_b*(L/L_b)^2*(del_b/del)	=	90	[kg/m^2]
	Panel truss cost per unit area	Ctruss	=	ctruss*mtruss	=	540	[$/m^2]
	Panel total cost per unit area	Cpanel	=	Ctruss+Csurf	=	690	[$/m^2]
ACTUATOR QUANTITY & COST CALCULATION							
	Actuator stroke	Sact	=	(D/2)^2*(1/cos(zmax)*(1+D*sin(zmax)/(4*R))-cos(zmax))/(4*R*(1+D*sin(zmax)/(4*R)))	=	11.5	[m]
	Actuator load	Pact	=	(msurf+mtruss)*(sqrt(3)*L^2/2)*9.81/100	=	359	[kN]
	Actuator diameter	Dact	=	max(Dact_b*sqrt(Pact/Pact_b),Sact_b*1	=	350	[mm]
	Actuator cost each	Cact_ea	=	(Cact_b+dCdS*(Sact-Sact_b))*(Dact/Dact_b)^Dexp	=	100,000	[$]
	Actuator cost per unit area	Cact	=	Cact_ea/(sqrt(3)*L^2/2)	=	289	[$/m^2]
RECEIVER/CORRELATOR QUANTITY & COST CALCULATION							
	Cost of receiver total per unit area	Crec	=	Crec_ea*nr/Atot	=	248	[$]
TOTAL COST		CTOT	=	Cpanel+Cact+Crec	=	1,226	[$/m^2]

Figure 44: Sample spreadsheet calculations for case study problem

6.2.5 Probabilistic Parameters

The following parameters have the greatest uncertainty with respect to estimating the overall cost, and were selected to be modeled as probabilistic variables in the analysis:

- c_{TRUSS}: cost of panel truss structure per unit mass, modeled with a standard normal PDF with a mean of $6/kg and a standard deviation of $2/kg

- *dC/dS*: cost per unit stroke of actuator, modeled with a standard normal PDF with a mean of $15000/m and a standard deviation of $5000/m
- ϕ_{EXP} : actuator cost scaling exponent for actuator diameter, modeled with a symmetric triangular PDF ranging from 1.5 to 2.5
- C_{REC_EA}: cost of receiver unit each, modeled with a symmetric triangular PDF ranging from $10M to $25M.

The above probability distributions are exemplary only, and in reality the estimates of the costing data would be developed by various means.

For fabricated components, unit cost values can be derived based on prototype data and also from databases from steel fabricators. Fabricators often track cost data from historical projects for use in estimating future projects. Typical quantities that are tracked include the following:

- Historical, current, and forecasted material costs and labour rates
- Person-hours per tonne of steel fabricated
- Person-hours per unit weld length
- Person-hours and material requirements per unit area for cleaning and painting
- Historical, current, and forecasted shipping costs per unit weight or unit volume
- Person-hours per tonne of steel erected

Data often exists for a wide range of fabrication projects, and the unit cost values described above vary widely for different types of projects. The similarities between past and present projects must be carefully assessed to determine which unit cost values to use. Having the capability to represent this uncertainty by applying a probability distribution to the unit cost data provides a valuable tool to assess risk.

In the case of actuator costing data it may be based on a combination of prototype data and vendor quotations for the hydraulic systems. Vendors may be engaged to provide cost estimates for a range of actuator data which could be used to verify the scaling laws, and multiple vendor quotations could be used to estimate the scatter in the costing data.

6.3 Analysis and Results

The coding of the constraining equations and probabilistic parameters described above into the Matlab framework is straightforward, and the general procedure is described in section 5.2 above.

The following sets of results plot the solution space with the design variables D (reflector diameter), L (panel size) and Δ (allowable deflection) on the x, y and z plot axes respectively. The selected plot resolution in (D, L, Δ) coordinates is (12,12,5)

The first results plotted in Figure 45 show the solution space where the contour represents the probability of the cost per unit area meeting the constraint $C_{TOT}<C_{TARGET}$=\$1100/m^2. The results indicate that for values of allowable deflection Δ in the range of 0.6mm to 1.0mm there is a large portion of the L-D design space that has a high likelihood of meeting the target cost, whereas for lower values of allowable deflection the feasible design space becomes more limited, and the probability of meeting the cost targets becomes lower.

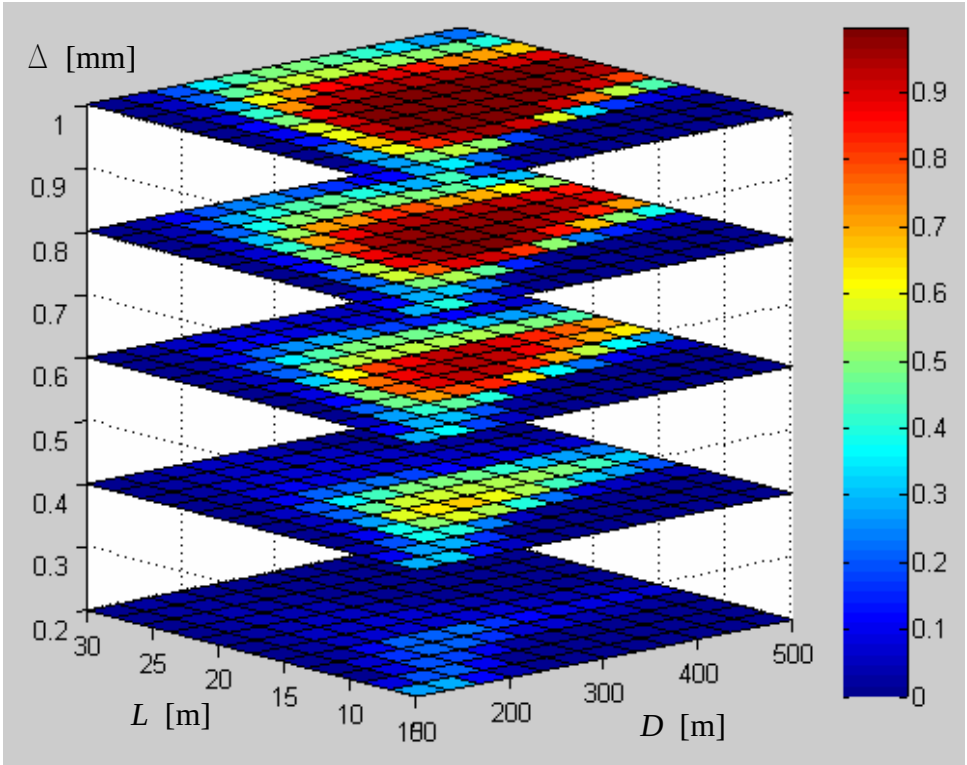

Figure 45: Probabilistic solution space with contour representing the probability of meeting the constraint $C_{TOT}<C_{TARGET}$=\$1100/m^2

Figure 46 below represents the data in a different way. In this case the results are presented as a 2D plot with the design variables *D*[m], *L*[m] plotted on the *x* and *y* axes, and the allowable deflection Δ [mm] plotted on the contour. Various plots are shown which represent the solution space surface for various probabilities of success. For example, considering the 90th percentile plot below the design space located at (*D*,*L*)=(250m:350m, 15m:17.5m) will have a 90% chance of meeting the cost target for a design with an allowable deflection of Δ=0.6mm which is shown on the contour.

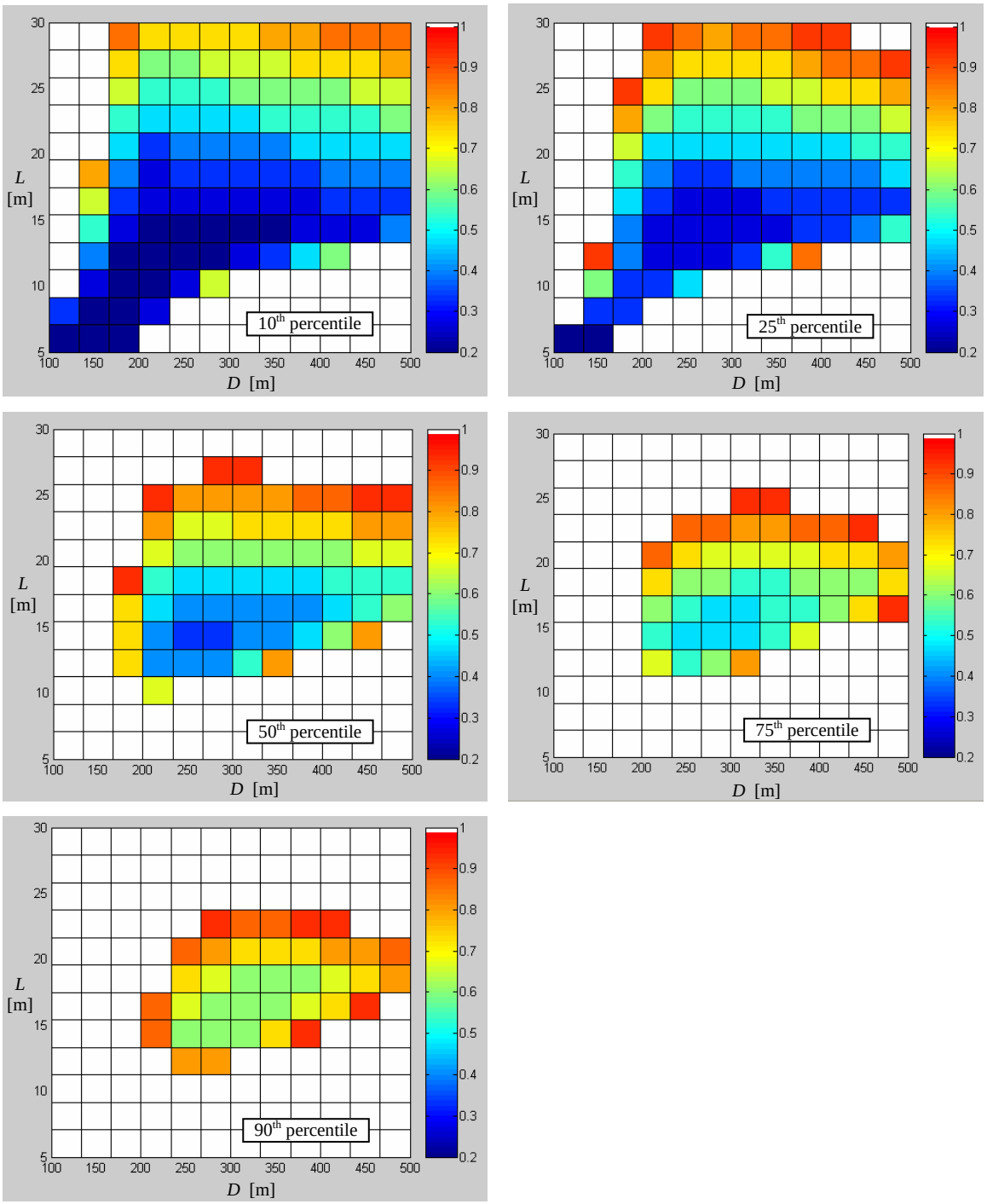

Figure 46: Solution space surface plots for 10th, 25th, 50th, 75th and 90th percentile surfaces with contour plotting the allowable deflection Δ in mm.

Additional data that is available as a function of the solution process. For example, Figure 47 below gives the cost breakdown for the major components over the design

space. The plotted costs are the probability-weighted costs which are the median costs in this case. This is calculated by summing the cost depth-wise through the full results matrix and at each entry multiplying the probability of the probabilistic variables occurring at the intervals contained in that entry.

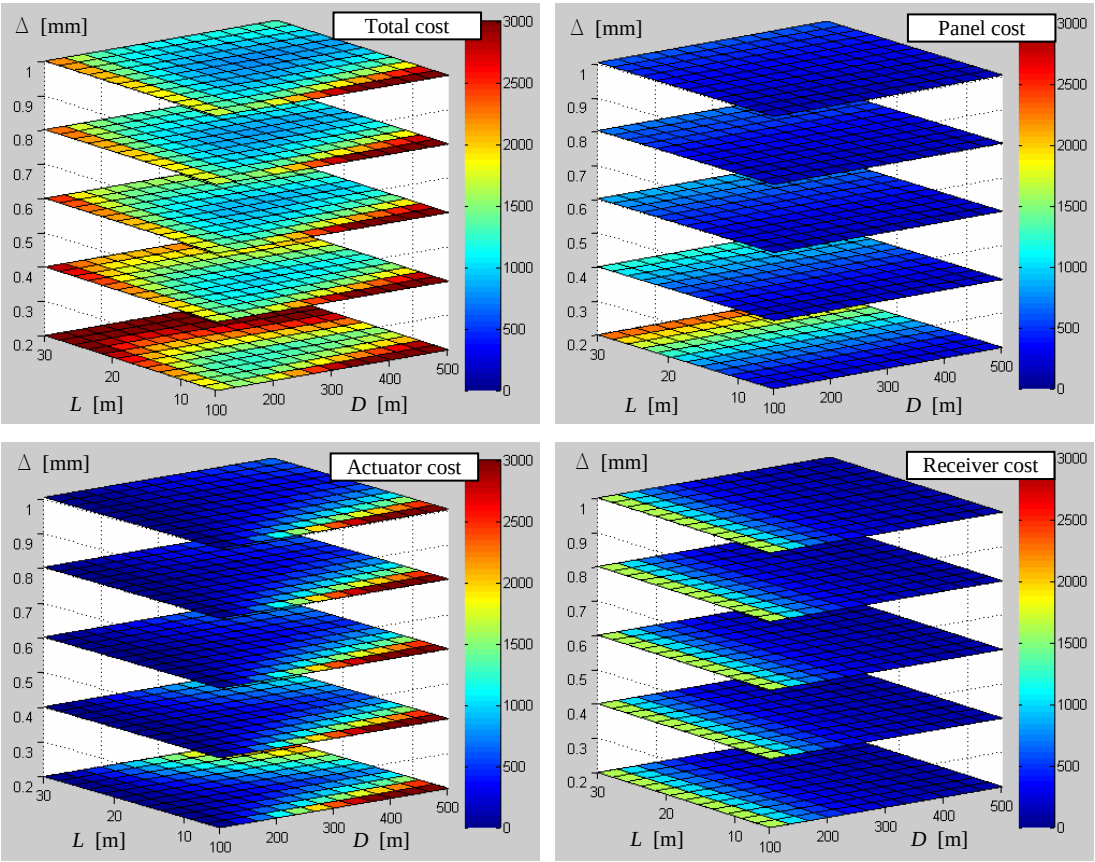

Figure 47: Probability-weighted cost breakdown in $/m^2 showing total cost (top left), panel cost (top right), actuator cost (bottom left) and receiver cost (bottom right)

The results plotted in Figure 47 indicate the trends in the component costs as a function of the design variables. The panel costs increase exponentially with panel size L since the longer spans require heavier panel truss structures to maintain the required surface accuracy. The panel costs also increase as the allowable deflection Δ is reduced since stiffer, heavier panel trusses are required to limit the deflection.

The actuator costs can be seen to generally increase with both reflector diameter D and panel size L. Increasing the reflector diameter requires actuators with a larger stroke since the required high and low points at the perimeter of the reflector increase in magnitude. Increasing the panel size requires actuators with a larger capacity since the panels become heavier. Lower values of allowable deflection Δ also result in increased panel weight, and therefore increased load on the actuators.

The receiver costs vary only as a function of reflector size D since if larger reflectors are used, fewer total reflectors are required, and thus fewer receivers are required.

6.4 Case Study Summary

The above case study demonstrates the application of the theory outlined in this thesis to an industrial problem involving the preliminary exploration of a design space using probabilistic parameters. A problem of this type can be coded relatively quickly into the software modules that have been developed as part of this thesis. For the above problem which utilizes three design variables and four probabilistic variables, the run-time to generate results at the resolution shown in the figures above is on the order of 100 seconds on a typical laptop computer. The resulting output can be used to identify design space solutions with various probabilities of meeting the project budget, carry out cost minimization, identify performance and cost tradeoffs, and indicate trends and dependencies in the component cost breakdown.

7 CONCLUSIONS

The research presented in this thesis describes a methodology referred to as Probabilistic Constraint Satisfaction that has high potential as a design aid, particularly during conceptual design. The basic theory and its application through a software framework were presented. The methodology combines probabilistic analysis with recent advances in numerical constraint satisfaction problems that use interval parameters. The outcome is a methodology that allows flexible problem formulation as a system of constraints, has the capability to handle probabilistic input of any distribution shape, and generates a map of the feasible design space with contours indicating solution reliability.

The generality of the methodology makes it applicable to any field requiring decision and risk analysis. In civil engineering, applications include concept design (solution space mapping and performance optimization), cost and risk minimization, and reliability analysis.

The methodology involves subdivision of the design space to a requested resolution and then testing the consistency of the constraint satisfaction problem at each subdivided location. This approach is very robust and is capable of solving a wide range of problems regardless of linearity or the availability of an explicit solution. Probabilistic data is integrated through the use of a solver which determines the valid interval for each of the probabilistic variables at a given point in the design space. The valid intervals are subsequently used to determine the probability of a feasible solution occurring at that point in the design space.

Two solution methods were presented for determining the valid intervals of probabilistic variables: the interval narrowing method, and the grid search method. The interval narrowing method combines interval arithmetic and constraint satisfaction problem techniques. Its advantages are that it greatly reduces the size of the output matrix since interval bounds on the parameters are solved internally in the interval narrowing process method, as opposed to requiring further subdivision of the design space which

exponentially increases the size of the output matrix. The iterations carried out by the interval narrowing process provide a logical proof that the solution is bounded within the narrowed interval. Furthermore, the process can be used to provide an initial narrowing of the design variables so that the initial design space bounds the feasible design space, and subsequent design space subdivisions are focused in this region.

The grid-search method provides a more exhaustive search of the design space. Advantages of the grid-search method over the interval narrowing method are its capability to identify gaps in the design space and to systematically test combinations of multiple probabilistic variable instantiations for consistency. The result is that the grid search method converges to the exact solution as resolution is increased, as opposed to the interval narrowing method which converges to an upper bound. Hybrid solutions involving an interval CSP solver combined with the grid search method were also discussed as a way to utilize the advantages of both methodologies. One combined approach that would likely yield efficient results would be to use the interval CSP method for the initial pass to narrow the intervals of the probabilistic variables at a point in the design space, and then further subdivide this interval using the grid search method to search for holes and variable instantiation combinations.

The applicability of this methodology to a wide range of general problems was demonstrated, including the following capabilities:

- Solution of linear and non-linear problems with no numerical instability
- Handling of multiple probabilistic variables with varying distribution types
- Representation of 2-dimensional, 3-dimensional, and 4-dimensional probabilistic solution spaces

The examples presented indicated how the methodology could be applied to structural engineering problems with probabilistic input parameters and reliability-based problems where the reliability is plotted as a function of the selected design variables. A case study involving the conceptual design of a novel radio antenna concept demonstrated the applicability of the methodology to an industrial problem. The selected case study had

several features that are often common to conceptual design problems for unique projects including design parameters with continuous domains, trade-offs between performance targets and cost, uncertain costing data, and limited existing experience to base the design on.

Areas for future work include further development of the software application to utilize the alternate solution methods discussed. This includes integrating existing Java-based interval narrowing methods into the Matlab-based framework developed for this thesis, and also development and testing of hybrid interval narrowing and grid search methods which show high potential of providing the best combination of efficiency and results which converge to the exact solution. Efficiency of these methods could be compared with each other, and also to Monte-Carlo simulation run-times. Additional statistical modules could be added to represent a wider range of probability distribution functions, and also correlations between multiple probabilistic variables could be implemented. Other potential software developments include a user interface to simplify the coding of new problems, and the development of a stand-alone application external to commercial software.

A useful extension of the theory would to incorporate a sensitivity analysis to determine the sensitivity of the model output to changes in the various input parameters, including the statistical parameters. Other research could focus on the application of the theory to various engineering problems.

8 References

Ayyub, B.M., and Chao, R.J. (1998). *Uncertainty Modeling and Analysis in Civil Engineering*, edited by B. Ayyub, CRC Press: 1-32

Bilda, Z, Gero, JS and Purcell, T. (2006). To sketch or not to sketch? That is the question. *Design Studies* 27(5): 587-613

Canadian Standard Association (2001). *CAN/CSA-S16.1-01 Limit States Design of Steel Structures.* Canadian Standard Association (Toronto)

Carlson B., et al. (2000). The LAR: A 200-m diameter, wideband, cm-m wave radio telescope. *Proceedings, SPIE International Symposium on Astronomical Telescopes and Instrumentation 2000*

Cerulli, C., Peng, C. and Lawson, B. (2001). Capturing histories of design processes for collaborative building design development - field trial of the ADS prototype. *Proceedings of the Ninth International Conference on Computer Aided Architectural Design Futures 2001*, Eindhoven, 8-11 July 2001, pp. 427-437

Cheng, M.Y., Tsai, H.C., Ko, C.H., and Chang, W.T. (2008). Evolutionary fuzzy neural inference system for decision making in geotechnical engineering, *Journal of Computing in Civil Engineering*, 22(4): 272-280

Davis, E. (1987). Constraint propagation with interval labels, *Artificial Intelligence*, 32: 281-331

De Vries, B. and Harink, J. (2005). Construction analysis during the design process, *Proceedings of the 11th International Conference on Computer Aided Architectural Design Futures*, Vienna (Austria) 20–22 June 2005, pp. 413-422

Elhag, T. and Wang, Y.M. (2007). Risk assessment for bridge maintenance projects: neural networks versus regression techniques, *Journal of Computing in Civil Engineering*, 21(6): 402-409

Freuder, E.C. (1978). Synthesizing constraint expressions. Communications of the ACM, 21:958-966

Gedig, M.H. and Stiemer, S.F. (2003). A computer application to study engineering projects at the early stages of development, *Electronic Journal of Structural Engineering*, 3: 43-66

Gedig, M.H. and Stiemer, S.F. (2003). Qualitative and semi-quantitative reasoning techniques for engineering projects at conceptual stage, *Electronic Journal of Structural Engineering*, 3: 67-88.

Gedig, M.H. (1995). A framework for qualitative and semi-quantitative analysis in engineering design and evaluation, *Masters Thesis*, University of British Columbia.

Gero, J.S. and McNeill, T. (1998). An approach to the analysis of design protocols, *Design Studies* 19(1): 21-61

Haukaas, T. (2006). Cost-benefit importance vectors for design improvement decisions, *Proceedings of the 2nd Forum of the International Forum on Engineering Decision Making*, Lake Louise, Canada, April 26-29

Hayes, P.J. (1979). The naïve physics manifesto, in , Michie, D. (Ed.) *Expert Systems in the Microelectronic Age*, Edinburgh University Press, Edinburgh

Hargreaves, G.I. (2002). Interval analysis in Matlab, *Numerical Analysis Report No. 416, Manchester Centre for Computational Mathematics*, Manchester, England

Iwasaki, Y. (1989). Qualitative physics. In Barr, A., Cohen, P.R., and Feigenbaum, E.A. (Eds.), *The Handbook of Artificial Intelligence*, Vol IV. Addison-Wesley, Reading, MA, USA.

Kuerschner, K. (1999). Conceptual design of Large Adaptive Reflector panel for the SKA. *Dip. Ing*, Dept of Civil Engineering, University of Stuttgart, Stuttgart

Kuipers, B.J. (1994). Qualitative reasoning: modeling and simulation with incomplete knowledge. MIT Press, Cambridge, MA.

Lhomme, O. (1993). Consistency techniques for numeric CSPs, in *Proceedings of Thirteenth Joint Conference on Artificial Intelligence*, IJCAII 1993, Morgan Kaufmann, San Mateo, CA. pp. 232-238

Loewen, N.L. (2003). Conceptual design of reflector structure for the Large Adaptive Reflector, *M.A.Sc. Thesis*, Department of Civil Engineering, University of British Columbia

Mackworth, A.K. (1977). Consistency in networks of relations, *Artificial Intelligence* 8: 99-118

Moore, R.E. (1966). *Interval Analysis*. Prentice-Hall, Englewood Cliffs, NJ, USA.

Purcell, T. and Gero, J. S. (1998). Drawings and the design process: a review of protocol studies in design and other disciplines and related research in cognitive psychology, *Design Studies* 19(4): 389-430

Scott, M. and Haukaas, T. (2008). Software framework for parameter updating and finite-element response sensitivity analysis, *Journal of Computation in Civil Engineering* 22(5): 281-291

Simmons, R. (1986). Commonsense arithmetic reasoning, *Proceedings of Fifth National Conference on Artificial Intelligence*, pp. 118-124.

Tversky, A. and Kahneman, D. (1974). Judgement under uncertainty: heuristics and biases, *Science* 185: 1124-31

Waltz, D.L. (1975). Understanding line drawings of scenes with shadows, in Winston, P.H. (Ed.), *The Psychology of Computer Vision*, pp. 19-92

Zhou, Y. and Stiemer, S.F. (2007). Engineering analysis with uncertainties and complexities using reasoning approaches, *Journal of Computing in Civil Engineering*, 5: 353-362

Zhou, Y. (2003). Engineering qualitative analysis and its application on fatigue design of steel structures, *Ph.D Thesis*, Department of Civil Engineering, University of British Columbia

Appendix A: Source Code for Example Problem 2

The following source code outlines the basic coding for example problem 2, which uses an integral grid-search method to check the satisfaction of the single constraint involved in this problem. The problem formulation for this example is straightforward, and the components are representative of more complex problems. The code as listed produces the output plot shown in the figure below:

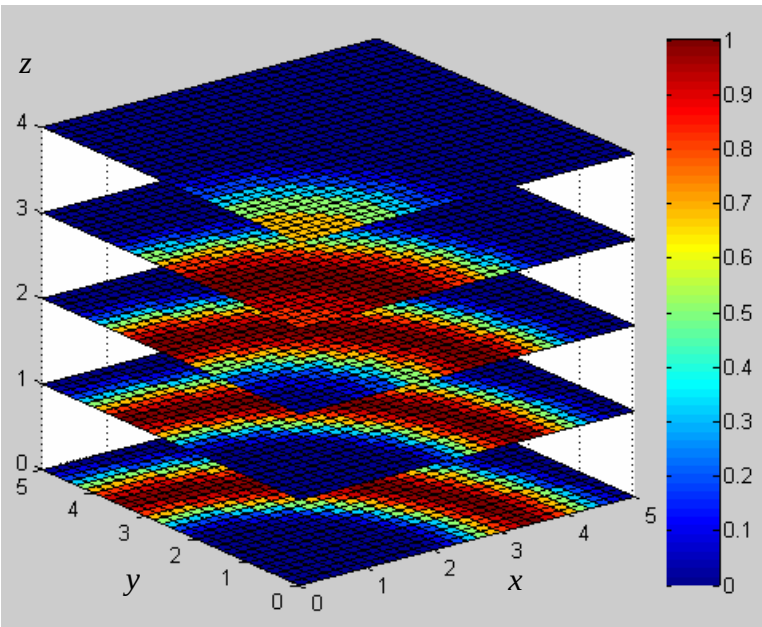

Figure 48: Solution space to example problem 2

```
% DEFINE DESIGN & PROBABILISTIC VARIABLE DATA
% Design variable data [low value, high value, number of boxes]
dv1=[0,5,20];
dv2=[0,5,20];
dv3=[0,5,20];

% Probabilistic variable data (symmetric triangular PDFs)
% [low value, high value, number of boxes]
pv1=[2,3,5];
pv2=[4,5,5];

% Generate data arrays at required size
```

```
dv1a=dv1(1,1)+(dv1(1,2)-dv1(1,1))*(0:dv1(1,3))'/dv1(1,3);
dv2a=dv2(1,1)+(dv2(1,2)-dv2(1,1))*(0:dv2(1,3))'/dv2(1,3);
dv3a=dv2(1,1)+(dv3(1,2)-dv3(1,1))*(0:dv3(1,3))'/dv3(1,3);
pv1a=pv1(1,1)+(pv1(1,2)-pv1(1,1))*(0:pv1(1,3))'/pv1(1,3);
pv2a=pv2(1,1)+(pv2(1,2)-pv2(1,1))*(0:pv2(1,3))'/pv2(1,3);

% FILL FULL PROBABILITY MATRIX
n1=dv1(1,3);
n2=dv2(1,3);
n3=dv3(1,3);
n4=pv1(1,3);
n5=pv2(1,3);

B=zeros(n1+1,n2+1,n3+1,n4+1,n5+1);

% Loop through design variables
for i1=1:n1;
for i2=1:n2;
for i3=1:n3;
    % Loop through probabilistic variables
    for j1=1:n4;
    for j2=1:n5;
        % Instantiate variables at box midpoint
        x=(dv1a(i1)+dv1a(i1+1))/2;
        y=(dv2a(i2)+dv2a(i2+1))/2;
        z=(dv3a(i3)+dv2a(i3+1))/2;
        ri=(pv1a(j1)+pv1a(j1+1))/2;
        ro=(pv2a(j2)+pv2a(j2+1))/2;

        % Check problem constraints
        if (x^2+y^2+z^2)>=ri^2;
            if (x^2+y^2+z^2)<=ro^2;
                truth=1;
            else
                truth=0;
            end
        else
            truth=0;
        end

        % Calculate probability over instantiated intervals
        P1=cdf_tri(pv1a(j1+1),pv1(1,2),pv1(1,1))- …
         …cdf_tri(pv1a(j1),pv1(1,2),pv1(1,1));
```

```
        P2=cdf_tri(pv2a(j2+1),pv2(1,2),pv2(1,1))- …
         …cdf_tri(pv2a(j2),pv2(1,2),pv2(1,1));
        B(i1,i2,i3,j1,j2)=P1*P2*truth;
    end
    end
end
end
end

% FILL PLOT-SPACE PROBABILITY MATRIX

% Sum probability matrix depth-wise
A4=sum(B,5);
A3=sum(A4,4);
A2=sum(A3,3);

% PLOT RESULTS
slice(y,x,z,A3,[],[],[0,1,2,3,4]);

% TRIANGULAR CDF SUBROUTINE
function c = cdf(d,dmax,dmin)
    if d<dmin
       c=0;
    elseif d<(dmax+dmin)/2
       c=((d-dmin)/(dmax-dmin))^2*2;
    elseif d>=(dmax+dmin)/2
       c=1-((dmax-d)/(dmax-dmin))^2*2;
    else
       c=1;
    end
```

Appendix B: Source Code for Convergence Example

The following is the Matlab code for the example problem presented in Section 4.4 demonstrating the convergence of the grid-search solution as the degree of subdivision increases.

```
for nexp=0:9;   % Loop through exponent values
ndiv=2^nexp;    % Number of subdivisions

    % DEFINE DESIGN & PROBABILISTIC VARIABLE DATA
    % Probabilistic variable data
    % [low value, high value, no. boxes]
    pv1=[0,2,ndiv];
    pv2=[1,3,ndiv];

    % Generate data arrays at required size
    pv1a=pv1(1,1)+(pv1(1,2)-pv1(1,1))*(0:pv1(1,3))'/pv1(1,3);
    pv2a=pv2(1,1)+(pv2(1,2)-pv2(1,1))*(0:pv2(1,3))'/pv2(1,3);

    % FILL FULL PROBABILITY MATRIX
    B=zeros(ndiv,ndiv);
    for j1=1:ndiv;
    for j2=1:ndiv;

        % Instantiate variables at current interval midpoint
        x=(pv1a(j1)+pv1a(j1+1))/2;
        y=(pv2a(j2)+pv2a(j2+1))/2;

        % Test constraint
        if x>=y;
            truth=1;
        else
            truth=0;
        end

        % Calculate probabilities
        P1=cdf_tri(pv1a(j1+1),pv1(1,2),pv1(1,1))- …
         …cdf_tri(pv1a(j1),pv1(1,2),pv1(1,1));
        P2=cdf_tri(pv2a(j2+1),pv2(1,2),pv2(1,1))- …
         …cdf_tri(pv2a(j2),pv2(1,2),pv2(1,1));
```

```
        B(j1,j2)=P1*P2*truth;
    end
    end

% CALCULATE TOTAL PROBABILITY
B1=sum(B,2);
Prob=sum(B1,1);

output(nexp+1,1)=nexp;
output(nexp+1,2)=ndiv;
output(nexp+1,3)=Prob;
end

output
```

The resulting output matrix printout contains the data given in Table 3. Column 1 contains the exponent value, column 2 contains the number of subdivisions, and column 3 contains the probability.

```
output =
         0    1.0000         0
    1.0000    2.0000    0.2500
    2.0000    4.0000    0.1094
    3.0000    8.0000    0.0684
    4.0000   16.0000    0.0535
    5.0000   32.0000    0.0472
    6.0000   64.0000    0.0444
    7.0000  128.0000    0.0430
    8.0000  256.0000    0.0423
    9.0000  512.0000    0.0420
   10.0000 1024.0000    0.0418
   11.0000 2048.0000    0.0417
```