ENGINEERING ANALYSIS WITH UNCERTAINTIES AND COMPLEXITIES, USING REASONING APPROACHES

Ye Zhou[1], Siegfried. F. Stiemer[2]

ABSTRACT

Conventional computation methods generally limit practicing engineers from using complex formulations or considering uncertainties in general. A method is needed that can be implemented regardless of the uncertainty or linearity of the design parameters and their constraints. Methods such as qualitative reasoning provide an effective and sound technique for solving complex and uncertain scenarios. Uncertainties in engineering designs can be formulated as variables in the application domain and processed by numerical constraint reasoning. This paper describes a software platform, built upon numerical constraint reasoning for engineering applications. The capability of representing design parameters and outcomes in a two-dimensional solution space provides a practical way for engineers to leverage their existing knowledge and experience. The software expresses the results of the analysis in variable ranges and diagrams showing a two-dimensional design space. Qualitative reasoning can assist in the difficult process of making appropriate engineering assumptions and judgments, when carrying out complicated analysis procedures. In addition, interval constraint analysis can be used to derive controlling parameters and design space, therefore giving engineers a good overall understanding of a problem when practical experience is not available.

KEY WORDS

Computer applications, conceptual design, decision support system, constraints, qualitative reasoning.

INTRODUCTION

Design engineers often need to avoid detailed analysis, such as finite element methods, due to resource constraints. Instead, they utilize simplified design methods specified by codes or standards, which generally rely heavily on judgment calls and experience with complex problems. In addition to the complexity of the analysis itself, many of the input parameters required for a detailed analysis may have a large degree of uncertainty during the design stage. Handling this uncertainty requires engineering experience and knowledge. This paper describes a software framework referred to as Qualitative Engineering System 2 (QES2).

[1] Research Engineer, AMEC Dynamic Structures Ltd.,1515 Kingsway Avenue, Port Coquitlam, BC V3C1S2, Canada. Email: ye.zhou@amec.com

[2] Professor, Department of Civil Engineering, University of British Columbia, 6250 Applied Science Lane, Vancouver, BC V6T1Z4, Canada. Email: sigi@civil.ubc.ca

QES2 is capable of analyzing engineering problems where the input with uncertainties is expressed in the form of equations, constraints, and design variable intervals, and the results are depicted in the form of a solution space.

COMPUTATIONAL METHODS

Qualitative reasoning (QR) promises high potential for use in the development of practical engineering design tools. Some recent developments show examples of building useful engineering tools with QR techniques (Hickey 2001, Kuipers 1994, Rossi 2000); however, these examples concentrate mainly on non-numerical problems. In contrast, this paper focuses on developing numerical applications of QR, and improving its efficiency by adopting new algorithms.

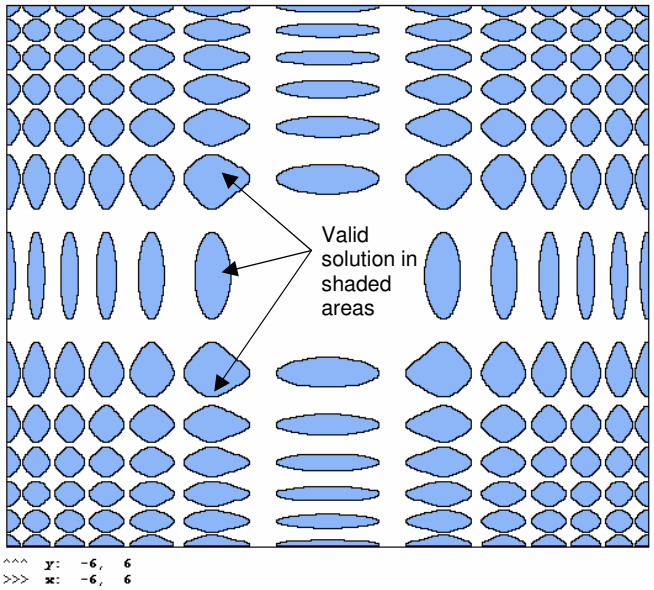

Figure 1: 2D solution space of $sin(x^2)+sin(y^2) < -0.25$

QUALITATIVE REASONING WITH UNCERTAINTIES

Any engineering design task can usually be decomposed into a set of relationships and constraints, which can be easily represented in terms of inequalities. Since application of relevant geometric and engineering principles is always carried out within the scope of such functional criteria, most important engineering decision-making involves judgments regarding inequalities.

Inequality constraints define solutions in the form of solution spaces. Single point solutions are sought in engineering due to the fact that complete solution spaces are too difficult to compute and manage. Using solution spaces can be extremely helpful during engineering decision-making. Figure 1 shows an example of the 2D solution space of $sin(x^2)+sin(y^2) < -0.25$. One can easily see that many non-connected solution spaces are possible, and that slight deviation of one parameter value can move the design point from a permitted region to a non-permitted region. Complex multi-dimensional solution spaces are

more common in real engineering analysis that are even more difficult to understand when only single numerical answers are available to the designer.

Qualitative reasoning is capable of deriving the complete solution space from a set of constraints. The key technique used in qualitative reasoning is constraint satisfaction. Engineering tasks are well suited to formulation as Constraint Satisfaction Problems (CSP), which are defined by a set of variables subject to constraints. The variables correspond to the relevant parameters of the design formulas. The constraints express design criteria by equalities or inequalities. The CSP approach uses search methods that detect single variable assignments that satisfy all the constraints, and then provide a description of solution spaces, i.e. the set of the entire solutions.

Consistency techniques are possible methods used for solving CSPs. They provide filters that remove inconsistent values (i.e. values that cannot be part of a solution) from the search space, and thus make the search more efficient. Consistency techniques can be used as preprocessors in order to simplify a CSP prior to solving. This approach has led to the identification of restrictions on the constraint syntax, the topology of the constraint network, and the solution space described by constraints. Most techniques for CSPs involve decomposing constraints into a tree, with nodes representing variables and branches representing constraints. Solving a CSP is a process of reasoning through such a tree. Reasoning in the direction from end nodes to tree root is known as backtracking.

While applications of CSPs usually deal with discrete or binary data, numeric CSPs (also known as continuous constraint systems or interval constraint systems) are the primary focus of interest for engineering applications. Interval constraints were first introduced by J. G. Cleary (Cleary 1987) in the 1980s to address the error of floating-point numerical computations in the Prolog programming language. Interval constraint processing combines propagation and search techniques developed from artificial intelligence with methods from interval analysis, also known as interval propagation. Given a set of constraints C involving variables $v_1 \ldots v_n$ and a set of floating-point intervals $V_1 \ldots V_n$ representing the domains of possible values, a reasoning procedure isolates a set of regions $R_1 \ldots R_n$ approximating the constraint system solution. To compute such a set, a search procedure navigates through the initial intervals $V_1 \ldots V_n$, alternating pruning and branching steps. The pruning step employs a relational form of interval arithmetic developed by R.E. Moore in the 1960s (Moore 1966). Given a set of constraints imposed on real numbers, interval arithmetic is used to compute local approximations of the solution space for a given constraint. It results in the discarding of values from the initial variable domains. These domain modifications are propagated through the entire constraint set until reaching a stable state. This is closely related to the notion of arc consistency, which is a well-known concept in artificial intelligence.

A system of constraint interval arithmetic consists of three distinct layers. The top layer is concerned with the conversion from the external source language to an internal data structure or constraint network. The middle layer handles the interval iteration and relates the properties of the primitive operations to that of the constraint network as a whole. The bottom layer is the theory for the implementation of primitive functions such as addition, subtraction, multiplication, etc. This paper focuses on development of the top layer for the application of integrating engineering computations of metal fatigue design, as well as the middle layer for the effective implementation of intensive engineering calculations that

involve a large number of variables and constraints. A number of computational primitives necessary for engineering calculations are also developed for the bottom layer.

ADVANCEMENT IN NUMERIC QUALITATIVE REASONING

The advantage of using CSPs is that consistency techniques represent an approximation of both input variables and solution spaces instead of single point values. In this context, the following methods have been implemented in the research presented here:

- The approximation of solution spaces is achieved by an improved local consistency method for numerical variables providing good results in pruning and execution time. This is achieved by using a local consistency operator for numerical constraints, which is superior in pruning power to existing methods.
- A novel search method using local consistency for numerical variables is implemented. In contrast to most existing approaches for solving mixed CSPs that are based on a cooperation between constraint solvers, this method integrates the local consistency methods for numerical variables into the search process, and also makes use of the mixed constraints to prune the search space.

The research described by this paper resulted in the QES2 software framework capable of accommodating complexity and uncertainty in engineering calculations by using techniques in qualitative reasoning and adaptive graphing. Recent advances in qualitative analysis have begun to apply reasoning techniques to continuous domains, but still lack efficiency in dealing with large quantities of arbitrary numerical constraints, such as those given in engineering problems (Gedig 1995). The contribution towards solving this problem reduces such inefficiency by retrofitting existing qualitative methods with interval arithmetic reasoning and optimized consistency algorithms. This presents a major step in the direction of applying qualitative techniques to everyday engineering.

Below is a summary of the main areas in which the research makes advances:

- The techniques of qualitative reasoning, interval arithmetic, and adaptive graphing are integrated. This provides advancement over existing tools, which are inefficient for handling large numerical problems and are incapable of accommodating problems of generic forms with complexities and uncertainties.
- This paper introduces higher order consistency techniques, as well as additional rule sets, to the standard interval-based techniques, which use simultaneous constraints over logical and numerical operators to enable stability in solving large numerical problems. Common functions in engineering calculations, such as the *if*() operator, are also developed and expand upon standard interval arithmetic libraries.

ALGORITHMS

Most constraint reasoning developments have had difficulties in achieving suitability to general applications. Consequently, reasoning for binary and discrete variables is neglected in QES2 to streamline the handling of numerical constraints. In addition, the introduction of

the high order consistency enhances the stability and efficiency for reasoning over numerical constraints.

The focal idea of interval arithmetic constraints is to view computing problems as constraint systems that relate a set of variables or functions in real numbers. The variables whose values are to be computed are initially unbounded in the model. The goal of the computation, or reasoning, is to shrink the intervals of the variables in such a way that no solution to the original system is removed. The shrinking, or variable narrowing, is done iteratively by applying various contraction operators that typically apply only a subset of the entire constraint set. The implementation of interval constraints focuses on the development of contraction operators, which apply a narrowing set of constraints to the variables without removing any solutions, while also retaining as much efficiency as possible. Consistency techniques are used to select contraction operators in a sequence that may potentially accelerate the solution convergence.

Constraint interval arithmetic is a relatively new approach to the problem of deriving numerical results from algebraic models. Since it is simultaneously a numerical computation technique and a proof technique, it bypasses the traditional dichotomy between numerical calculations and symbolic proofs. The combination of proof and calculation is used to handle practical problems, which neither method can handle alone. The underlying semantic model is based on the properties of monotone contraction operators on a lattice, an algebraic setting in which fixed point semantics take a particular sequential form.

COMPUTATION AS PROOF

The other aspect of constraint interval arithmetic that uniquely distinguishes it from traditional numerical techniques is that the computations represent proofs of the non-existence of solutions. Interval proofs can be used to refer to real numbers since only bounded precision constants actually appear in the proofs, and the constraint system itself has no notion of real numbers in the full mathematical sense. In practical applications, this one-directional bounding technique is not contradictory to the consistency techniques described in the preceding text. The absolute consistency is a requirement in strict logical sense, but one-directional bounding is sufficient in most applications of engineering numerical calculations.

POTENTIAL PITFALLS

For general constraint networks containing loops, it is very difficult to predict performance theoretically even for specific problems, just as it is generally difficult to predict the number of iterations to convergence in conventional fixed-point iterations. It is also unclear how to formulate a useful complexity measure for constraint interval arithmetic in general. Since termination is governed by the actual precision being used, a complexity model must take the precision into account. Worst-case performance for fixed precision in general has an upper bound that depends exponentially on the number of variables but is independent of the problem being solved. This bound, which is on the order of the number of different states of the system, is so large as to be practically useless, and there are some relatively simple problems that come close to reaching it. For example:

$abs(X) \leq M$, $abs(Y) \leq M$,

$A \times X + B \times Y == C$
$A \times X + B \times Y == C'$

where C and C' are disjoint intervals which differ by some small δ. This eventually results in failure, but effectively does so by counting from $-M$ to $+M$ by steps of size δ, which can be an enormously large number. On the other hand, in practice, industrial sized problems with hundreds of variables and constraints will in most cases converge to a fixed point in a reasonable amount of time. If the constraints are inconsistent, failure usually occurs fairly quickly unless the initial conditions are very near the boundary of the failure region.

CONSISTENCY ENFORCEMENT

Narrowing a system of variables through a set of constraints is the process of consistency enforcement. These are key techniques to all numerical constraint solvers, including QES2. They govern the efficiency, stability and generality of a constraint solver.

Research in the last decade shows that interval arithmetic is best applicable in real number reasoning (Benhamou 1997, Rossi 2000) and has been adopted by the QES2 solver. Enforcing local consistency through interval arithmetic offers generality to numerical problems, but if used alone may be slow and plagued by complexity issues described in the previous section. The efficiency and stability may be potentially improved by applying high-order and global consistencies. However, most of the latest implementations of high-order consistency require significant increases in reasoning time that may offset the improvement in global reasoning stability. None of the current implementations of global consistency techniques are capable of handling general numerical problems (Sam 1995).

In the reasoning solver of QES2, local consistency is enforced by recursive iterations through a binary tree of converted constraints, similar to hull consistency in many other numerical constraint solvers. In contrast to other solvers, QES2 discards consistency manipulators in binary and discrete domains, consequently increasing the efficiency by preserving only interval arithmetic manipulators. To support global efficiency and avoid complexities, QES2 uses adaptive 2^{nd}-order consistency enforcement, as well as a direct partial enforcement on global consistency. Apart from being compatible with all reasoning scenarios targeted by other developments on consistency techniques, the introduction of these two methods yields dramatic improvements on global efficiency and stability for most numerical engineering problems. Such improvements are the result of sacrificing absoluteness in strict logic applicability, while focusing on numerical solving by the same methodology as finite element structural analysis, as described in the next section.

The QES2 solver always starts a reasoning process by converting the constraint set into a binary tree, in which the enforcement of first-order consistency, or hull consistency, can proceed. A narrowing iteration consists of a series of recursive requests calling the root node, propagated down to the end nodes, and then backtracked with narrowing functions that finish at the root node. The narrowing functions are first supported by interval arithmetic operations, composing the enforcement of 1^{st}-order local consistency. The second step of a narrowing function is an adaptive 2^{nd}-order local consistency check, which is realized by evaluating the level of further narrowing provided by the upper level consistency against a preset parameter nr, defined as:

$$nr = \frac{\text{Amount narrowed in the first constraint among all 2nd order constraints}}{\text{Amount narrowed in the 1st order constraint}}$$

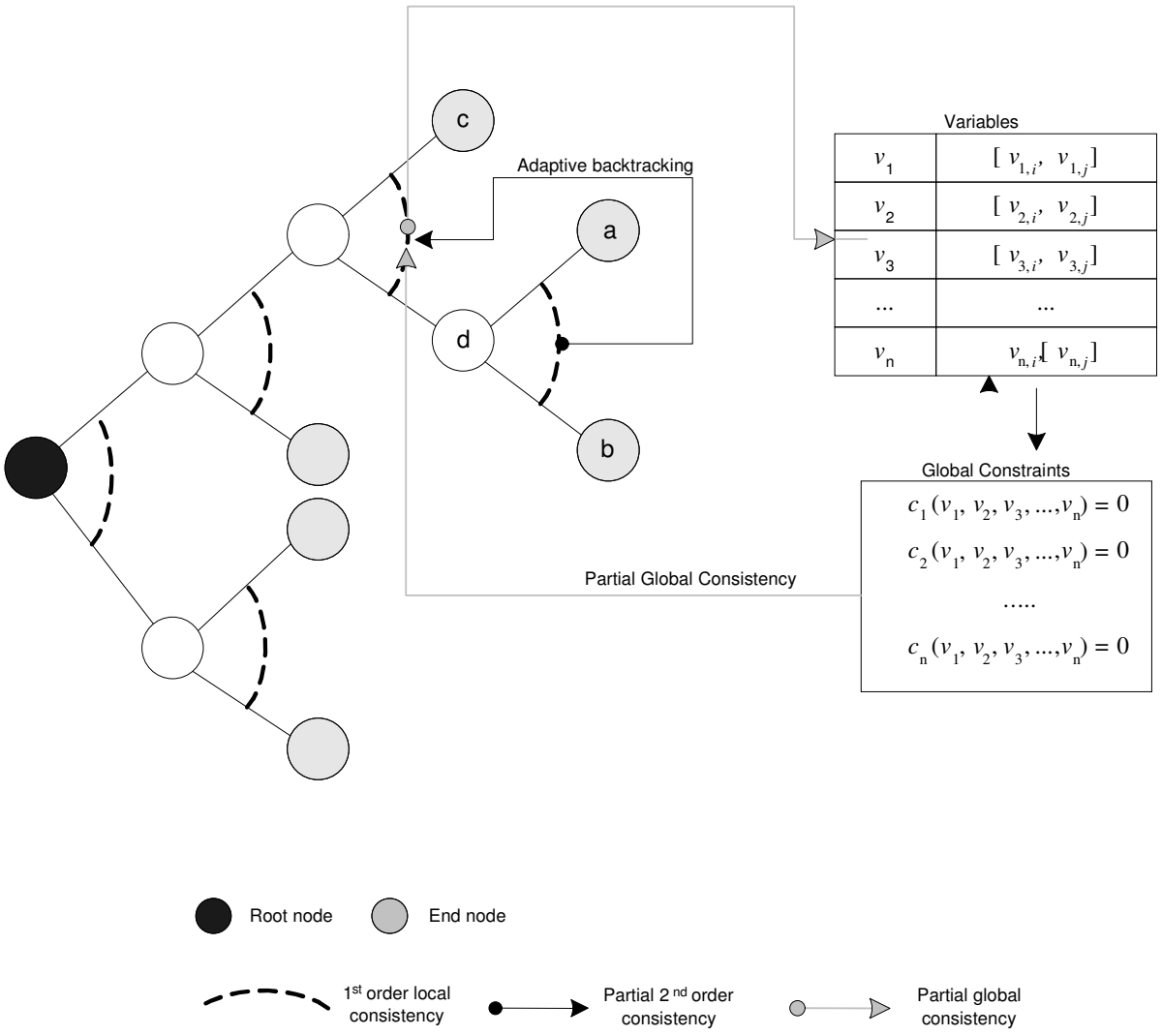

Figure 2: Adaptive 2nd-order and partial global consistency enforcement

With conventional interval constraint reasoning, the only means to accomplish global consistency is by enforcing local consistencies over all constraints. Since converged approximation of variables is well accepted in engineering calculations, QES2 implements the enforcement of global consistency at all instances on narrowing over the 1st-order consistency. Figure 2 illustrates such a process. While narrowing on the consistency between node *c* and *d*, the current table of variables is substituted into the global constraint set for a consistency check. The normal narrowing process continues when the consistency exists. For the case that the global consistency is not achieved, bisection is called in the current narrowing step before proceeding to the next local consistency level. The outer bounds of the

precision level used with the reasoning solver are the global consistency check to ensure logical soundness. In order to avoid potentially excessive computation associated with the iterative global-local consistency interaction, the process of global consistency checking is a one-way process, i.e. a maximum of a single iteration is allowed. While the enforcement over partial global consistency works on the assumption of approximation, it does eliminate the complexity issues of local consistencies acting alone.

ANALYSIS WITH QES2

QES2 is a software implementation of the theories and algorithms described in the preceding sections. QES2 enables an engineering design or analysis to account for uncertainties in the form of ranges of input design parameters, and illustrate analysis results in the form of plots of solution spaces. QES2 is based upon the techniques of numerical constraint reasoning, interval arithmetic and adaptive plotting.

QES2 calls for the following requirements when modelling an engineering problem:

- The problem must be described, explicitly or implicitly, by equations that define the relationships between the design variables. The components of the equation system, or the equation system as a whole, can be linear or nonlinear, determinate or indeterminate.
- All design variables can be described numerically, in the form of a single real number, or as an interval between two real numbers.

By solving the given problem with numerical constraint reasoning, QES2 produces sound solutions, where the highest achievable numerical accuracy is limited only by the hosting software and hardware platforms. Upon completion of a successful analysis, QES2 presents the results in the following forms:

- The design variable results are expressed as intervals between two real numbers. If an interval of a design variable is too narrow to be physically meaningful, the value of the variable may be considered to be a single value; otherwise, the variable is valid inside of the interval.
- A 2-D graph with specified resolutions can be plotted between two design variables. Provided that the variables are valid as intervals, instead of as a single real number, the plot will be an area, or solution space, instead of a line. The 2-D graph further improves the representation of the output by including discontinuous solution spaces.
- Multiple sets of solution spaces can be plotted on the same graph, showing the relationships between a pair of variables under different sets of constraints.

During the solving process, QES2 may fail to derive a solution due to numerical reasoning difficulties, although this is rarely encountered. All engineering problems are physically meaningful and can provide additional constraints to overcome reasoning difficulties.

EXAMPLES

Engineering calculations often involve solving and understanding multi-variable equations. Complex nonlinear equations are difficult or impossible to solve, and even more difficult to interpret. The following formulation is a three-variable equation set for solving the force/displacement relationship of a simple truss, including an arbitrarily introduced nonlinearity. QES2 allows interpretation of nonlinear solutions with parameter variations by plotting a solution space (Figure 3).

$$29.5\times10^{6}/600\times(A_{11}\times X1+A_{12}\times\sin(X2)+A_{13}*X3) = P1;$$
$$29.5\times10^{6}/600\times(A_{21}\times X1+A_{22}\times X2^{2}+A_{23}*X3) = P2;$$
$$29.5\times10^{6}/600\times(A_{31}\times X1+A_{32}\times X2^{3}+A_{33}*X3) = P3;$$

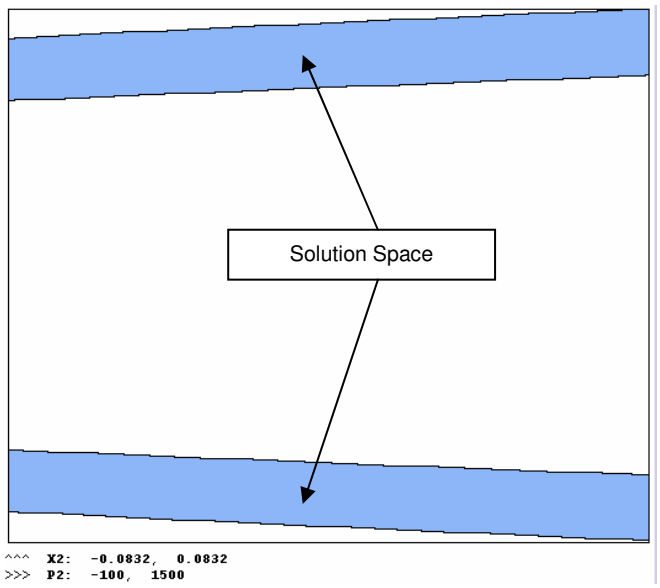

Figure 3: QES2 solution space plot of a multi-variable, nonlinear equation set

CONCLUSIONS

The QES2 computer software framework applies the latest developments in qualitative constraint reasoning to practical engineering problems. Using these techniques, QES2 is free of the constraints that limit conventional approaches, with the capability of solving complex engineering problems regardless of their formulations, explicitness or linearity.

Qualitative reasoning techniques are effective in solving problems that can be expressed with constraints. Constraints are able to account for uncertainty in engineering design by using numerical intervals for input parameters. Being a proving process, qualitative reasoning also possesses the following characteristics that are beneficial for engineering applications:

- *Faithfulness to the mathematical model*. Qualitative reasoning regards numerical computations as computer-generated proofs that true real numbers are contained within a certain pair of computer floating-point numbers.
- *Soundness of results.* The reasoning operations result in intervals that contain all values that are logically possible from arithmetic operations.
- *No restriction on forms of formulations.*
- *Accommodation of uncertainties*. Engineers are able to take into account uncertainties by specifying interval bounds for the design parameters.

This paper approached the reasoning techniques in a way that is analogous to the approximation of finite element modelling used in structural analysis; with proper modelling, the finite element method is able to approximate the true solution with increased accuracy as element size decreases. The reasoning process in QES2 generates solution spaces of design variables through adaptive plotting. This allows a logical, rather than analytical, treatment of continuous solution space, thereby avoiding the speed limitations of reasoning with singularities and other analytical anomalies. The accuracy in the resulting solution space plot is limited by the resolution of the presentation media, such as the computer monitor, and the accuracy is guaranteed by the nature of logical reasoning.

REFERENCES

Benhamou, F. and Older, W.J. (1997) "Applying interval arithmetic to real, integer, and Boolean constraints." *Journal of Logic Programming*, 32:1-24.

Cleary, J.G. (1987) "Logical arithmetic." *Future Computing Systems*, 2(2) 1987.

Gedig, M.H. (1995) A Framework for Qualitative and Semi-Quantitative Analysis in Engineering Design and Evaluation. Masters Thesis, University of British Columbia.

Hickey, T. (2001) "Metalevel Interval Arithmetic and Verifiable Constraint Solving." *Journal of Functional and Logic Programming*, Vol. 2001, no 7, Oct. European Association for Programming Languages and Systems.

Krawczyk, R. (1986) "A class of interval-Newton operators." *Computing*, 37. Springer-Verlag. 1986

Kuipers, B.J. (1994) Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge. MIT Press, Cambridge, MA.

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

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

Rossi, F. (2000) Constraint (Logic) Programming: A Survey on Research and Applications." *New Trends in Constraints*. Springer-Verlag (Berlin) 2000

Sam, J. (1995) *Constraint Consistency Techniques for Continuous Domains*. Thesis No. 1423, Ecole Polytechnique Federale de Lausanne (Lausanne).

Zhou, Y. (2003) Engineering Qualitative Analysis and its Application on Fatigue Design of Steel Structures. Ph.D. Thesis, University of British Columbia.