

Solving Interval Constraints by Linearization in Computer-Aided Design

YAN WANG*

NSF Center for e-Design, University of Central Florida, 4000 Central Florida Blvd, Orlando, FL 32816, USA, e-mail: wangyan@mail.ucf.edu

and

BARTHOLOMEW O. NNAJI

NSF Center for e-Design, University of Pittsburgh, 1048 Benedum Hall, Pittsburgh, PA 15261, USA, e-mail: nnaji@engr.pitt.edu

(Received: 12 November 2004; accepted: 11 April 2005)

Abstract. Current parametric CAD systems require geometric parameters to have fixed values. Specifying fixed parameter values implicitly adds rigid constraints on the geometry, which have the potential to introduce conflicts during the design process. This paper presents a soft constraint representation scheme based on nominal interval. Interval geometric parameters capture inexactness of conceptual and embodiment design, uncertainty in detail design, as well as boundary information for design optimization. To accommodate under-constrained and over-constrained design problems, a double-loop Gauss-Seidel method is developed to solve linear constraints. A symbolic preconditioning procedure transforms nonlinear equations to separable form. Inequalities are also transformed and integrated with equalities. Nonlinear constraints can be bounded by piecewise linear enclosures and solved by linear methods iteratively. A sensitivity analysis method that differentiates active and inactive constraints is presented for design refinement.

1. Introduction

During the process of design, various design variables are specified, which include geometric variables (e.g. dimension, volume, and tolerance) and non-geometric ones (e.g. functional characteristics, tooling speed, and expected life). When design is realized by geometric form, these design variables are finalized and implemented as geometric parameters in parametric Computer-Aided Design (CAD) systems. Current CAD systems only allow geometric parameters to have fixed values, such as the position of a point in 3D space, the direction of a line, and the distance between two axes. Instead of simply assigning real values to design variables and the derived geometric parameters in CAD models, there are some advantages to give interval values to variables and parameters, which means that a variable or parameter can take any valid value between the lower and upper bounds of the interval.

* Corresponding author.

Fixed-value design variables and geometric parameters in CAD generate some problems. First, fixed values bring up conflicts easily in the design process. Specifying determined values of design variables and thereby geometric parameters implicitly adds rigid constraints of value range on geometric parameters at the very beginning of design implementation. The rigid constraints reduce the freedom of geometric entities in a CAD model to the minimal level. These dominant constraints may become the sources of conflicts at later stages. To resolve the conflicts, the values of some design variables have to be changed. This trial-and-error cycle will continue until no conflicts occur. If an interval instead of a fixed value is assigned to a design variable or a geometric parameter in CAD systems so that any real value within the interval is valid, the degrees of freedom of geometric shape are increased. As more constraints are imposed onto the designed object during the process, the freedom of geometric entities will be restricted gradually. The allowable interval values of design variables and thus geometric parameters are reduced by stages. There would be fewer chances that conflicts occur, and several cycles of modification can be saved.

Second, the requirement of fixed values for geometric parameters makes the development of Computer-Aided Conceptual Design (CACD) tools difficult. At the conceptual and embodiment design stages, actual values of design variables may not be known. Usually it is not important to specify fixed values of certain variables at the earlier design stages yet. Current CAD systems require that parameter values be fully specified and fixed, thus they are not effective tools for conceptual and embodiment design. It is quite challenging to develop a practically usable CACD tool based on the current scheme of fixed parameter values. Nevertheless, if the value of a parameter is specified as a range, the problem of parameter partial integrity can be solved, i.e., it is not necessary to fix all values of parameters. This increases the flexibility of geometric modeling. Inexactness of preference and specification is represented, and CACD is possible based on interval values.

Third, the specifications of valid value range are not captured by fixed-value variables. Current design optimization process usually occurs after all variables are specified at the detail design stage, while the original intention of feasible ranges of variables from upstream design activities is not transferable with the fixed-value scheme. Bounds have to be added separately for optimization purpose. However, with the interval representation, the inherent range information is directly applicable for optimization. Intervals appropriately represent design intent of feasibility, thus integrating the process from conceptual sketching to parameter optimization.

Variable and parameter intervals capture the uncertainty characteristics of design. In real-world situations, there are many uncertainty factors in CAD modeling. The dimensions and shape of the designed objects are computed and stored digitally in CAD systems. Representing an infinite number of real numbers by a finite number of bits requires approximation. Not all decimal numbers can be represented in binary format exactly. Rounding errors come from the approximation. Cancellation errors occur because of catastrophic and benign cancellation. The

precision of numbers in a computer depends on the word size and floating-point representation. Variation exists among different systems with different architectures. Uncertainty also comes from measurement as well as tolerance of human perception during the parameter specification.

In this paper, a nominal interval constraint representation (NICR) scheme based on nominal interval values is described to represent inexactness, uncertainty, and feasibility boundary of design information for CAD. It represents soft constraints, thus reducing the chances of conflicts during constraint imposition. It provides a generic numerical parameter scheme for different design phases. A piecewise linear enclosure is developed to transform nonlinear interval constraint systems and solved by a double-loop Gauss-Seidel iteration method, which accommodates under-constrained and over-constrained parametric design problems.

The paper is organized as follows. Section 2 reviews constraint-driven parametric design mechanisms, interval analysis in engineering design applications, as well as existing interval constraint solving methods. Section 3 introduces the NICR representation and notations. Section 4 describes the methods to solve under- and over-determined linear equations and the linearization method for nonlinear equations. Symbolic preconditioning is introduced to incorporate inseparable functions and inequalities. Section 5 describes a sensitivity analysis method to refine design. Finally, Section 6 gives a comprehensive example for these methods.

2. Background

2.1. CONSTRAINT-DRIVEN DESIGN IN PARAMETRIC CAD

Geometric constraints are fundamental constraints to be captured in engineering design. The study of geometric constraint representation can be traced back to the origin of CAD systems. Constrained geometries are sets of loci that satisfy certain constraints, thus they can be constructed systematically by computer systems. Different types of geometric constraint solving methods and associated representation for CAD have been proposed. Generally, there are four approaches. The numerical approach [27], [38], [66], [70], [87], [94] translates geometric constraints into a system of mathematical equations. These equations then can be solved numerically by Newton-Raphson or Homotopy methods directly, or by minimizing the total errors for all equations indirectly. The artificial intelligence approach [1], [2], [62], [69], [101], [111], [118] represents geometric constraints by facts and rules. Constraint problems are solved by the aid of geometric reasoning. The symbolic approach [14], [25], [60], [61] translates geometric constraints into a system of easily solvable nonlinear equations with symbolic algebraic methods, such as Göbner's bases or the Wu-Ritt method, before numerically solving them. The constructive approach [8], [13], [19], [24], [28], [39], [40], [42], [62], [67], [68], [93], [109] represents constraints as graphs internally. Constraint system is solved by either

top-down decomposition or bottom-up clustering of the constraint graphs with the aid of degrees of freedom analysis.

From a different perspective, the NICR presented here allows all numerical values of parameters including coordinates, dimensions, and others to be nominal interval numbers. Thus, *soft* constraints compared to traditional fixed-value *rigid* constraints can be represented.

2.2. INTERVAL ANALYSIS IN ENGINEERING DESIGN

Interval mathematics [5], [36], [46], [51], [79], [80], [89], [98] is a generalization in which interval numbers replace real numbers, interval arithmetic replaces real arithmetic, and interval analysis replaces real analysis. An interval $A = [a_L, a_U]$ is defined by a pair of real numbers, a_L and a_U , for lower and upper bounds.

Interval analysis has been applied in computer graphics, including rasterizing parametric surfaces [83], ray tracing of parametric surfaces [112] and implicit surfaces [47], collision detection of polyhedral objects [78] and surface models [22], [107], [108], [114].

In engineering design applications, Finch and Ward [23] applied interval analysis to eliminate infeasible design in set-based modeling. Rao and Berke [95] used interval arithmetic for imprecise structural analysis. Rao and Cao [96] applied interval analysis in design optimization of mechanical systems. Muhanna and Mullen [84]–[86] developed an element-by-element interval finite-element formulation method for uncertainty in solid and structural mechanics. Sharp enclosures on structural displacement and forces can be obtained with the consideration of interval dependency. Modares et al. [77] extended the method to analyze structural stability under uncertainty. Related to interval representation, probabilistic modeling, and fuzzy logic are also applied in engineering design.

In CAD applications, Sederberg and Farouki [104] used interval arithmetic in approximating Bezier curves. Maekawa and Patrikalakis [72], [73] used interval Bezier curves to solve shape interrogation problems. Hu et al. [43], [44] used rounded-interval arithmetic to ensure numerical robustness in Boolean operations and boundary evaluation. Tuohy et al. [113] applied interval methods for interpolating measured data with B-spline curves and surfaces. Wallner et al. [115] used intervals to bound errors in geometric construction. Chen and Lou [16] proposed methods to bound interval Bezier curve with lower degree interval Bezier curve. Lin et al. [71] investigated the boundary evaluation of interval Bezier curve. The above research concentrates on the improvement of geometric model's robustness, in which intervals embody rounding and cancellation errors during floating-point computation. In this paper, we propose soft constraint representation with intervals for conceptual and embodiment design. New methods of solving interval constraints are developed.

2.3. SOLVING INTERVAL CONSTRAINTS

2.3.1. *Linear Equality Systems*

Interval linear equation systems $\mathbf{AX} = \mathbf{B}$, where \mathbf{A} is interval matrix and \mathbf{B} and \mathbf{X} are interval vectors, can be solved by interval version of Gaussian elimination and Gauss-Seidel iteration [5], [36], [51], [89]. Hansen [30], [35] introduced a preconditioning procedure by multiplying both sides of the equations with an approximate inverse matrix of the center of \mathbf{A} to reduce the effect of dependence in Gaussian elimination. Hansen [29] and Bliet [12] developed an explicit bounding method to find interval hull of solution set. Rohn [99] gave a rigorous proof of the interval hull method. Ning and Kearfott [92] extended the interval hull method with a general formula for H-matrix coefficient linear systems, as also proved by Neumaier [88]. Shary [105], [106] introduced an algebraic approach to estimate outer and inner bounds of solution sets with the extension of Kaucher complete interval arithmetic [48]. Neumaier [90] unified the algebraic approach with fixed-point inverse method. Chiu and Lee [18] developed an incremental preconditioned interval Gauss-Seidel method.

2.3.2. *Nonlinear Equality Systems*

Interval nonlinear equation systems can be solved by different methods. The interval Newton method [4], [5], [79], [89] searches the roots of $f(X) = 0$ based on range estimation of the derivative $f'(X)$. Fixed-point contraction [6], [76], [102] solve $f(X) = X$ by iterative substitution of X with epsilon inflation. The Krawczyk operator [63], [64] removes the interval matrix inverse operation and improves on the feasibility problem of interval Newton method if intervals are not narrow. Wolfe [116] introduced inner iterations into the Krawczyk method to reduce the number of times to compute derivatives. Alefeld and Platzöder [7] modified the Krawczyk operator with the linear Gaussian algorithm. Hansen operator [33], [34] integrates preconditioning and Gauss-Seidel iteration into nonlinear equation solving. Kearfott [36], [50] proposed a linear programming approach for preconditioning to minimize interval widths. Hansen [32] developed symbolic preconditioning with cancellation of common algebraic terms. Kearfott and Walster [54] developed symbolic preconditioning with Taylor models. Chen [17] generalized the Krawczyk operator to non-smooth equations by using the mean-value theorem for non-smooth functions. Benhamou and Granvilliers [9] proposed a symbolic and numeric hybrid approach to accelerate convergence of the Newton's method based on Gröbner bases. Sufficient conditions for the existence and uniqueness of solutions for the Newton alike linearization operators were developed [36], [51], [82] with nonsingular linearization. Kearfott et al. extended the existence and uniqueness test to problems with singular Jacobi matrices [52], [53] and non-smooth functions [49] by computation of topological degrees in complex space. Recently, Moore [81] proposed preprocessing to reduce dimension of sparse systems in fixed point form.

Hansen and Walster [37] developed an algorithm to compute sharp bounds on the real roots of polynomials with interval coefficients. Zhang et al. [120] extended the Krawczyk operator for under constrained problems with generalized inverse operation for non-square matrices. Wolfe [117] applied generalized inverse matrix operation to extended Krawczyk operator with second derivatives. Kolev [55], [56] developed a top-down decomposition method to bound interval factorable functions with linear interval enclosures. This method is further extended with a bottom-up linear enclosure construction [57] and an initialization method to narrow the linear bound [59].

2.3.3. Polynomial Enclosure

Different polynomial forms have been proposed for interval functions to enclose the range of a function. Mean-value form [5], [79], [89] represents value ranges by estimate of interval derivatives. Slope form [3], [31], [58], [65], [103], [121] replaces interval derivatives with interval slope. Taylor form [91], [98] extends mean-value form to high-order derivatives. Horner form [15], [110] reduces overestimation of interval polynomial evaluation. Bernstein form [26], [41], [45], [97], [100] bounds polynomials by ranges of Bernstein coefficients. Taylor model [10], [11], [74], [75] changes coefficients of Taylor form from intervals to real numbers. Motivated by Taylor model and Kolev's work [55], we propose a piecewise linear enclosure for polynomials to solve interval geometric constraints.

3. Nominal Interval Constraints

In NICR, we define interval number X as $X = [x_L, x_N, x_U]$ which contains lower bound value x_L , nominal value x_N , and upper bound value x_U . The nominal value is usually corresponding to the specified fixed value in current CAD systems.

The introduction of the nominal value into an interval is necessary for CAD modeling. The nominal value represents user preference and the actual user specification if the parameter is fixed. It allows current CAD modeling system to adopt interval parameters so that intervals can be integrated with current fixed-value schemes and visualization methods. Furthermore, the nominal value is allowed to change within the range, allows more user interaction and captures the preference information. For example, a 2D point $P([1, 2, 3], [4, 5, 6])$ can be displayed at $(2, 5)$. When P is fixed, its coordinates are $([2, 2, 2], [5, 5, 5])$. A real number is a degenerated interval.

3.1. NOMINAL INTERVAL DEFINITIONS AND NOTATIONS

An n -dimensional real number space is denoted as \mathbf{R}^n . An n -dimensional interval number space is denoted as \mathbf{IR}^n . $X = [x_L, x_N, x_U] = \{x \mid x_L \leq x \leq x_U, x_L \leq x_N \leq x_U\}$, where $x_L \in \mathbf{R}$, $x_N \in \mathbf{R}$, $x_U \in \mathbf{R}$, and $X \in \mathbf{IR}$.

Given that $A = [a_L, a_N, a_U]$, $B = [b_L, b_N, b_U]$, and \wedge is logical *and*, we have the following relations:

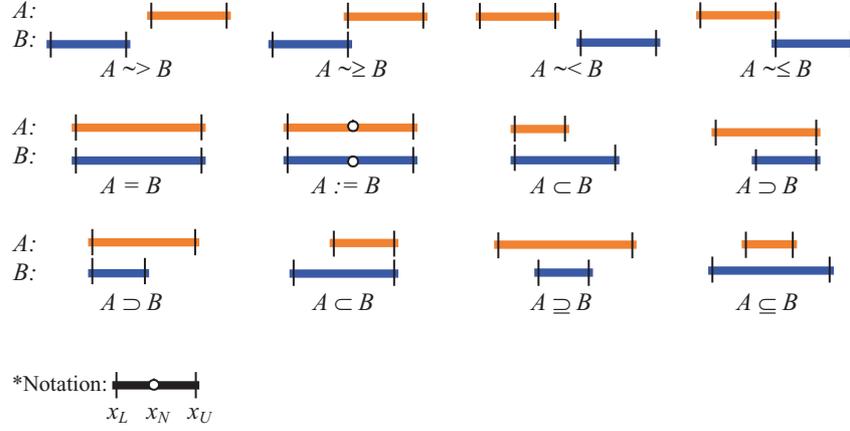


Figure 1. Relations between intervals.

- *equivalence*: $A = B \Leftrightarrow (a_L = b_L) \wedge (a_U = b_U)$.
- *nominal equivalence*: $A := B \Leftrightarrow (a_L = b_L) \wedge (a_N = b_N) \wedge (a_U = b_U)$.
- *strictly greater than or equal to*: $A \sim \geq B \Leftrightarrow a_L \geq b_U$.
- *strictly greater than*: $A \sim > B \Leftrightarrow a_L > b_U$.
- *strictly less than or equal to*: $A \sim \leq B \Leftrightarrow a_U \leq b_L$.
- *strictly less than*: $A \sim < B \Leftrightarrow a_U < b_L$.
- *inclusion*: $A \subseteq B \Leftrightarrow (a_U \leq b_U) \wedge (a_L \geq b_L)$, $A \subset B \Leftrightarrow (a_U < b_U) \wedge (a_L > b_L)$.

The relations of intervals are illustrated in Figure 1. $0 = [0, 0, 0]$ is *zero interval*. Interval A is *positive (negative)*, iff $A \sim > 0$ ($A \sim < 0$). If the nominal value of $A = [a_L, a_N, a_U]$ is not of concern, A can simply be denoted as $[a_L, a_U]$.

Interval $A = [a_L, a_N, a_U]$ is *empty*, denoted as $A = \emptyset$, iff $a_L > a_U$. A is *invalid* when $a_N > a_U$, or $a_L > a_N$, or A is empty. The basic arithmetic and set operations are:

- $A \cap B = \{x \mid x \in A \text{ and } x \in B, x \in \mathbf{R}\}$. If $A \cap B \neq \emptyset$, it can be derived by $A \cap B = [\max\{a_L, b_L\}, (\max\{a_L, b_L\} + \min\{a_U, b_U\}) / 2, \min\{a_U, b_U\}]$.
- $A \cup B = \{x \mid x \in A \text{ or } x \in B, x \in \mathbf{R}\}$. If $A \cap B \neq \emptyset$, it can be derived by $A \cup B = [\min\{a_L, b_L\}, (\min\{a_L, b_L\} + \max\{a_U, b_U\}) / 2, \max\{a_U, b_U\}]$.
- $A \setminus B = \{x \mid x \in A \text{ and } x \notin B, x \in \mathbf{R}\}$.
- $A + B = [a_L + b_L, a_N + b_N, a_U + b_U]$.
- $A - B = [a_L - b_U, a_N - b_N, a_U - b_L]$.
- $A \cdot B = [\min\{a_L b_L, a_L b_U, a_U b_L, a_U b_U\}, a_N b_N, \max\{a_L b_L, a_L b_U, a_U b_L, a_U b_U\}]$.
- $\frac{1}{B} = \left\{ \frac{1}{y} \mid y \in B, 0 \notin B \right\}$.

$$\bullet \frac{A}{B} = \begin{cases} A \cdot \frac{1}{B}, & (0 \notin B) \\ [-\infty, 0, +\infty], & (B = 0) \\ \left[\frac{a_U}{b_L}, \frac{a_U}{b_L}, +\infty \right], & (a_U \leq 0, b_L < 0, b_U = 0) \\ \left[-\infty, \frac{a_U}{b_U}, \frac{a_U}{b_U} \right] \cup \left[\frac{a_U}{b_L}, \frac{a_U}{b_L}, +\infty \right], & (a_U \leq 0, b_L < 0, b_U > 0) \\ \left[-\infty, \frac{a_U}{b_U}, \frac{a_U}{b_U} \right], & (a_U \leq 0, b_L = 0, b_U > 0) \\ [-\infty, 0, +\infty], & (a_L < 0, a_U > 0, b_L \leq 0, b_U \geq 0) \\ \left[-\infty, \frac{a_L}{b_L}, \frac{a_L}{b_L} \right], & (a_L \geq 0, b_L < 0, b_U = 0) \\ \left[-\infty, \frac{a_L}{b_L}, \frac{a_L}{b_L} \right] \cup \left[\frac{a_L}{b_U}, \frac{a_L}{b_U}, +\infty \right], & (a_L \geq 0, b_L < 0, b_U > 0) \\ \left[\frac{a_L}{b_U}, \frac{a_L}{b_U}, +\infty \right]. & (a_L \geq 0, b_L = 0, b_U > 0) \end{cases}$$

The width of an interval is $\text{wid}(A) = a_U - a_L$. $\text{wid}(\emptyset) = 0$. Some other notations are $\text{ubd}(A) = a_U$, $\text{lbd}(A) = a_L$, and $\text{nom}(A) = a_N$.

3.2. SAMPLING RELATION BETWEEN REAL GEOMETRY AND INTERVAL GEOMETRY

The intervals capture uncertainty of design. The value of a parameter, which is generated by computer or selected by human designer, is a sample of the corresponding set of values within the interval. One CAD interval model is allowed to generate different shapes because of parameter intervals. Implicitly, a CAD interval model defines a set of geometric shapes that automatically accommodate geometry variation.

Some *strict* relations \mathfrak{R} 's exist among intervals, which are related to real number samples. $X \mathfrak{R} Y \Leftrightarrow \forall x \in X, \forall y \in Y, x \mathfrak{R} y$.

- *strict equivalence*: $A \sim B \Leftrightarrow \forall x \in A, \forall y \in B, x = y$.
- *strictly greater than or equal to*: $A \sim \geq B \Leftrightarrow \forall x \in A, \forall y \in B, x \geq y$.
- *strictly greater than*: $A \sim > B \Leftrightarrow \forall x \in A, \forall y \in B, x > y$.
- *strictly less than or equal to*: $A \sim \leq B \Leftrightarrow \forall x \in A, \forall y \in B, x \leq y$.
- *strictly less than*: $A \sim < B \Leftrightarrow \forall x \in A, \forall y \in B, x < y$.

Besides strict relations, some *global* relations \mathfrak{J} 's exist in interval arithmetic evaluation and problem solving. $X \mathfrak{J} Y \Leftrightarrow \forall x \in X, \exists y \in Y, x \mathfrak{J} y$.

- *global equivalence*: $A = B \Leftrightarrow \forall x \in A, \exists y \in B, x = y$.
- *greater than or equal to*: $A \geq B \Leftrightarrow a_L \geq b_L$. Equivalently, $A \geq B \Leftrightarrow \forall x \in A, \exists y \in B, x \geq y$.

- *greater than*: $A > B \Leftrightarrow a_L > b_L$. Equivalently, $A > B \Leftrightarrow \forall x \in A, \exists y \in B, x > y$.
- *less than or equal to*: $A \leq B \Leftrightarrow a_U \leq b_U$. Equivalently, $A \leq B \Leftrightarrow \forall x \in A, \exists y \in B, x \leq y$.
- *less than*: $A < B \Leftrightarrow a_U < b_U$. Equivalently, $A < B \Leftrightarrow \forall x \in A, \exists y \in B, x < y$.

Strict inequalities are special cases of global inequalities. Global relations ensure the feasibility of interval arithmetic operations and solutions. The global relations make global solution and optimization of interval analysis possible. We assume global relations to be the default relations, such as the four basic arithmetic operations and function evaluation.

Interval vectors with same dimensions can be ranked and sorted.

- Interval vectors \mathbf{A}^I and \mathbf{B}^I are in a non-decreasing order, $\mathbf{A}^I \prec \mathbf{B}^I$, where $\mathbf{A}^I = (A_1, A_2, \dots, A_n)$, $\mathbf{B}^I = (B_1, B_2, \dots, B_n)$ if $A_n \leq B_n$, and $\neg(A_i < B_i) \rightarrow (A_{i-1} \leq B_{i-1})$ recursively apply, starting from $i = n$.
- Interval vectors \mathbf{A}^I and \mathbf{B}^I are in a non-increasing order, $\mathbf{A}^I \succ \mathbf{B}^I$, where $\mathbf{A}^I = (A_1, A_2, \dots, A_n)$, $\mathbf{B}^I = (B_1, B_2, \dots, B_n)$ if $A_n \geq B_n$, and $\neg(A_i > B_i) \rightarrow (A_{i-1} \geq B_{i-1})$ recursively apply, starting from $i = n$.
- $\max \text{wid}(\mathbf{A}^I) = \max_i(\text{wid}(A_i))$, where $\mathbf{A}^I = (A_1, A_2, \dots, A_n)$.
- $\min \text{wid}(\mathbf{A}^I) = \min_i(\text{wid}(A_i))$, where $\mathbf{A}^I = (A_1, A_2, \dots, A_n)$.

A *power interval* in an n -dimensional vector space of \mathbf{X}^I with a degree of m , denoted as $\mathbf{PX}^{(m,n)}$, is an ordered list of m non-overlapped interval vectors of n -dimensional, i.e., $\mathbf{PX}^{(m,n)} = \{\mathbf{X}_1^I, \mathbf{X}_2^I, \dots, \mathbf{X}_m^I\}$, where $\mathbf{X}_i^I \in \mathbf{IR}^n$ ($i = 1, \dots, m$), $\min \text{wid}(\mathbf{X}_i^I \cap \mathbf{X}_j^I) = 0$ ($i \neq j$), and $\mathbf{X}_i^I \prec \mathbf{X}_{i+1}^I$ ($i = 1, \dots, m - 1$).

3.3. VARIATIONAL GEOMETRY WITH INTERVAL CONSTRAINTS

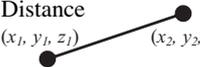
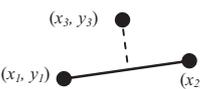
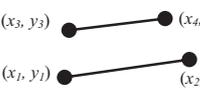
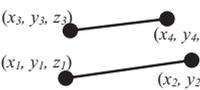
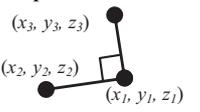
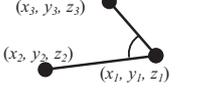
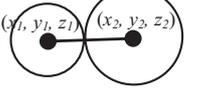
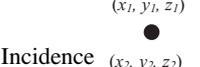
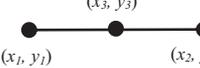
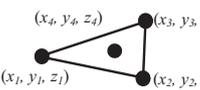
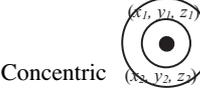
With the inherent capability of modeling uncertainty and inexactness, NICR has some special properties that make it different from current geometric modeling schemes. Changing geometric parameter values or adding extra geometric constraints lead to different geometries. For example, in Figure 2, the topology of a 2D rectangular shape may vary based on coordinates of four corner points within their allowable ranges.

A geometric model is a geometry vector

$$\mathbf{x} = [x_1, y_1, z_1, \dots, x_n, y_n, z_n]^T,$$

which are coordinates of n characteristic points in 3D Euclidean space, satisfying the constraints $f(\mathbf{x})$. The relations between these points can be linear or nonlinear equality or inequality. Commonly used geometric constraints can be represented in polynomial forms. Some examples are listed in Table 1. In feature-based parametric CAD systems, geometric constraints for shape construction usually are 2-dimensional. 3-dimensional constraints are normally used in assembly.

Table 1. Some examples of geometric constraints.

Relation	Constraint
Distance 	$(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 = d^2$
Distance 	$[(y_2 - y_1)(x_3 - x_1) - (x_2 - x_1)(y_3 - y_1)]^2$ $= d^2[(y_2 - y_1)^2 + (x_2 - x_1)^2]$
Parallel 	$(y_2 - y_1)(x_4 - x_3) - (x_2 - x_1)(y_4 - y_3) = 0$
Parallel 	$\begin{cases} (y_2 - y_1)(x_4 - x_3) - (x_2 - x_1)(y_4 - y_3) = 0, \\ (z_2 - z_1)(x_4 - x_3) - (x_2 - x_1)(z_4 - z_3) = 0, \\ (y_2 - y_1)(z_4 - z_3) - (z_2 - z_1)(y_4 - y_3) = 0 \end{cases}$
Perpendicular 	$(x_2 - x_1)(x_3 - x_1) + (y_2 - y_1)(y_3 - y_1) + (z_2 - z_1)(z_3 - z_1) = 0$
Angle 	$[(x_2 - x_1)(x_3 - x_1) + (y_2 - y_1)(y_3 - y_1) + (z_2 - z_1)(z_3 - z_1)]^2$ $= \cos^2 \alpha \cdot [(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2]$ $[(x_3 - x_1)^2 + (y_3 - y_1)^2 + (z_3 - z_1)^2]$
Tangent 	$(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 = (r_1 + r_2)^2$
Incidence 	$(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 = 0$
Incidence 	$(x_2 y_3 - x_3 y_2) + (x_3 y_1 - x_1 y_3) + (x_1 y_2 - x_2 y_1) = 0$
Incidence 	$(x_4 - x_1)[(y_2 - y_1)(z_3 - z_1) - (z_2 - z_1)(y_3 - y_1)]$ $+ (y_4 - y_1)[(z_2 - z_1)(x_3 - x_1) - (x_2 - x_1)(z_3 - z_1)]$ $+ (z_4 - z_1)[(x_2 - x_1)(y_3 - y_1) - (y_2 - y_1)(x_3 - x_1)] = 0$
Concentric 	$(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2 = 0$

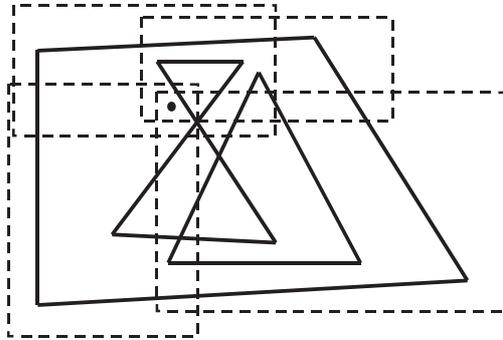
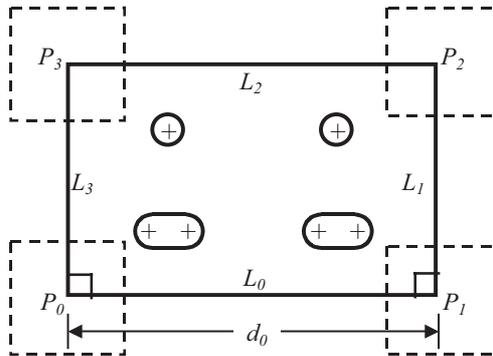


Figure 2. Interval constraint driven geometry exhibits inherent variational geometry.



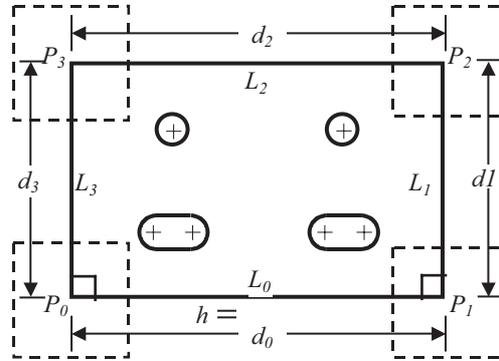
(a)

$$\begin{aligned} (x_1 - x_0)^2 + (y_1 - y_0)^2 &= d_0^2 \\ (x_0 - x_3)(x_1 - x_0)^2 + (y_0 - y_3)(y_1 - y_0)^2 &= 0 \\ (x_2 - x_1)(x_1 - x_0)^2 + (y_2 - y_1)(y_1 - y_0)^2 &= 0 \end{aligned}$$

(b)

Figure 3. An example of under-constrained geometry.

Current parametric modeling scheme has strict requirements on the number of constraints. The number of constraints should be equal to the number of variables, namely well-constrained. The concept of under-constrained and over-constrained geometry is not critical in interval representation. Soft constraints are applied to geometry implicitly at every step of specifications. The effect of adding more constraints is to reduce allowable regions of geometric entities. In the example of Figure 3, the rectangular contour of a mounting bracket is under-constrained if the available constraints are the distance between corner points P_0 and P_1 , the perpendicularity between lines L_0 and L_1 , and lines L_0 and L_3 .



(a)

$$\begin{aligned}
 x_0 &= a_0 \\
 y_0 &= b_0 \\
 y_0 - y_1 &= 0 \\
 (x_1 - x_0)^2 + (y_1 - y_0)^2 &= d_0^2 \\
 (x_2 - x_1)^2 + (y_2 - y_1)^2 &= d_1^2 \\
 (x_3 - x_2)^2 + (y_3 - y_2)^2 &= d_2^2 \\
 (x_0 - x_3)^2 + (y_0 - y_3)^2 &= d_3^2 \\
 (x_0 - x_3)(x_1 - x_0) + (y_0 - y_3)(y_1 - y_0) &= o_1 \\
 (x_2 - x_1)(x_1 - x_0) + (y_2 - y_1)(y_1 - y_0)^2 &= o_2 \\
 x_0 &\leq x_1
 \end{aligned}$$

(b)

Figure 4. An example of over-constrained geometry.

Over-determined or over-constrained situation is also allowed. As illustrated in Figure 4, if the geometric constraints in the bracket design are specified as: the position of P_0 ; distances between P_0 and P_1 , P_1 and P_2 , P_2 and P_3 , and P_3 and P_0 ; L_0 is perpendicular to L_1 as well as to L_3 ; and L_0 is horizontal, current CAD systems will complain that this geometry is over-constrained. However, in interval representation, only constraints that cause no feasible regions generate conflicts. Intervals loosen the current requirement on the number of constraints and give a different view of specification in CAD.

4. Solving Interval Constraints

To incorporate interval geometric modeling into current CAD systems, several fundamental issues related to geometric computation should be addressed. These include linear and nonlinear equation representation and solution, which are essential for transformation and assembly operation, surface intersection, and parametric shape construction, etc.

```

INPUT: Interval matrix A
       Interval vector B
OUTPUT: Interval vector X

Interval V
int i, j, k
REPEAT until stop criterion is met
  FOR each 1 <= i <= m
    FOR each 1 <= j <= n
      V = 0
      FOR each 1<=k<j
        V = V+Aik*Xk
      ENDFOR
      FOR each j+1<=k<=n
        V = V+Aik*Xk
      ENDFOR
      V = (Bi - V)/Aij
      Xj = Xj ∩ V
    ENDFOR
  ENDFOR

```

Figure 5. A double-loop Gauss-Seidel method.

4.1. INTERVAL LINEAR EQUATIONS

An interval linear system is

$$\mathbf{A}^I \mathbf{X}^I = \mathbf{B}^I, \quad (4.1a)$$

where $\mathbf{A}^I \in \mathbf{IR}^{m \times n}$, $\mathbf{X}^I \in \mathbf{IR}^n$, and $\mathbf{B}^I \in \mathbf{IR}^m$. Under-constrained ($m < n$) and over-constrained ($m > n$) linear systems are the major considerations in our context. Iteration methods have no requirement on the number of constraints. After the linearization process of geometric constraints, which will be discussed in Section 4.2, matrix \mathbf{A} is a real matrix. System (4.1a) becomes

$$\mathbf{A}\mathbf{X}^I = \mathbf{B}^I. \quad (4.1b)$$

A double-loop Gauss-Seidel method without preconditioning

$$\tilde{\mathbf{X}}_j^I = \frac{1}{\mathbf{A}_{ij}} \left(\mathbf{B}_i^I - \sum_{k \neq j} \mathbf{A}_{ik} \mathbf{X}_k^I \right) \cap \mathbf{X}_j^I \quad \text{for each } i \text{ and } j \quad (4.2)$$

is developed, as listed in Figure 5.

A second method to solve the linear system (4.1b) is to use Singular Value Decomposition (SVD) with linear least-square estimation. Suppose $\mathbf{A} = \mathbf{U}\mathbf{W}\mathbf{V}^T$, where $\mathbf{U}_{m \times n}$ and $\mathbf{V}_{n \times n}$ are column-orthogonal matrices, and $\mathbf{W}_{n \times n}$ is a diagonal matrix. With a preconditioner \mathbf{A}^T , we solve

$$\mathbf{A}^T \mathbf{A} \mathbf{X}^I = \mathbf{A}^T \mathbf{B}^I. \quad (4.3)$$

That is

$$\mathbf{X}^I = \mathbf{V} \left(\text{diag} \left(\frac{1}{w_j} \right) \right) \mathbf{U}^T \cdot \mathbf{B}^I.$$

In singular cases, replace $1/w_j$ with 0 if $w_j = 0$ and the corresponding X_j is not narrowed. If we define

$$\sum(\mathbf{A}^I, \mathbf{B}^I) := \{x \in \mathbf{R}^n \mid \exists A \in \mathbf{A}^I, \exists B \in \mathbf{B}^I, Ax = B\},$$

then

$$\sum(\mathbf{A}, \mathbf{B}^I) \subseteq \sum(\mathbf{A}^T \mathbf{A}, \mathbf{A}^T \mathbf{B}^I).$$

Here is an example. To solve linear system

$$\begin{bmatrix} 1 & 3 \\ 4 & 2 \\ 1 & -1 \\ -2 & 4 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} [0, 6] \\ [-6, 8] \\ [-1, 1] \\ [-4, 1] \end{bmatrix},$$

the double-loop Gauss-Seidel method without preconditioning has the result

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} [-1.8, 2.4] \\ [-0.8, 1.45] \end{bmatrix},$$

while the SVD preconditioning method gives

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} [-1.2135, 2.0976] \\ [-0.9269, 1.2013] \end{bmatrix}.$$

The hull of united solution set is

$$\begin{bmatrix} X_1 \\ X_2 \end{bmatrix} = \begin{bmatrix} [-0.3, 1.6666] \\ [-0.25, 1] \end{bmatrix}.$$

Some examples from [92] are computed using the double-loop Gauss Seidel method and results are compared in Table 2.

4.2. INTERVAL NONLINEAR EQUATIONS

To solve interval nonlinear equations

$$F_i(\mathbf{X}^I) = 0 \quad (i = 1, \dots, l), \quad (4.4)$$

where $\mathbf{X}^I = [X_1, X_2, \dots, X_n]^T \in \mathbf{IR}^n$, a piecewise linear enclosure algorithm is developed to solve geometric constraints. This algorithm is more general than Kolev's algorithm with the consideration of multiple solutions. The piecewise linear enclosure generates individual bounds for multiple roots.

Table 2. Comparisons of linear equation methods.

A^I	B^I	Gaussian elimination	Hansen [29]	Ning-Kearfott [92]	Double – loop Gauss – Seidel
$\begin{bmatrix} [4, 6] & [-1, 1] & [-1, 1] & [-1, 1] \\ [-1, 1] & [-6, -4] & [-1, 1] & [-1, 1] \\ [-1, 1] & [-1, 1] & [9, 11] & [-1, 1] \\ [-1, 1] & [-1, 1] & [-1, 1] & [-11, -9] \end{bmatrix}$	$\begin{bmatrix} [-2, 4] \\ [1, 8] \\ [-4, 10] \\ [2, 12] \end{bmatrix}$	$\begin{bmatrix} [-2.60, 3.10] \\ [-3.90, 1.50] \\ [-1.43, 2.15] \\ [-2.35, 0.60] \end{bmatrix}$	$\begin{bmatrix} [-2.50, 3.10] \\ [-3.90, 1.20] \\ [-1.40, 2.15] \\ [-2.35, 0.60] \end{bmatrix}$	$\begin{bmatrix} [-2.50, 3.10] \\ [-3.90, 1.20] \\ [-1.40, 2.15] \\ [-2.35, 0.60] \end{bmatrix}$	$\begin{bmatrix} [-2.50, 2.44] \\ [-3.70, 1.45] \\ [-1.38, 2.05] \\ [-2.25, 0.69] \end{bmatrix}$
$\begin{bmatrix} [3.7, 4.3] & [-1.5, -0.5] & [0, 0] \\ [-1.5, -0.5] & [3.7, 4.3] & [-1.5, -0.5] \\ [0, 0] & [-1.5, -0.5] & [3.7, 4.3] \end{bmatrix}$	$\begin{bmatrix} [-14, 14] \\ [-9, 9] \\ [-3, 3] \end{bmatrix}$	$\begin{bmatrix} [-6.38, 6.38] \\ [-6.40, 6.40] \\ [-3.40, 3.40] \end{bmatrix}$	$\begin{bmatrix} [-6.38, 6.38] \\ [-6.40, 6.40] \\ [-3.40, 3.40] \end{bmatrix}$	$\begin{bmatrix} [-6.38, 6.38] \\ [-6.40, 6.40] \\ [-3.40, 3.40] \end{bmatrix}$	$\begin{bmatrix} [-6.38, 6.38] \\ [-6.40, 6.40] \\ [-3.40, 3.40] \end{bmatrix}$
$\begin{bmatrix} [3.7, 4.3] & [-1.5, -0.5] & [0, 0] \\ [-1.5, -0.5] & [3.7, 4.3] & [-1.5, -0.5] \\ [0, 0] & [-1.5, -0.5] & [3.7, 4.3] \end{bmatrix}$	$\begin{bmatrix} [-14, 0] \\ [-9, 0] \\ [-3, 0] \end{bmatrix}$	$\begin{bmatrix} [-6.38, 0] \\ [-6.40, 0] \\ [-3.40, 0] \end{bmatrix}$	$\begin{bmatrix} [-6.38, 1.12] \\ [-6.40, 1.54] \\ [-3.40, 1.40] \end{bmatrix}$	$\begin{bmatrix} [-6.38, 6.67] \\ [-6.40, 2.77] \\ [-3.40, 2.40] \end{bmatrix}$	$\begin{bmatrix} [-6.38, 0] \\ [-6.40, 0] \\ [-3.40, 0] \end{bmatrix}$
$\begin{bmatrix} [3.7, 4.3] & [-1.5, -0.5] & [0, 0] \\ [-1.5, -0.5] & [3.7, 4.3] & [-1.5, -0.5] \\ [0, 0] & [-1.5, -0.5] & [3.7, 4.3] \end{bmatrix}$	$\begin{bmatrix} [0, 14] \\ [0, 9] \\ [0, 3] \end{bmatrix}$	$\begin{bmatrix} [0, 6.38] \\ [0, 6.40] \\ [0, 3.40] \end{bmatrix}$	$\begin{bmatrix} [-1.12, 6.38] \\ [-1.54, 6.40] \\ [-1.40, 3.40] \end{bmatrix}$	$\begin{bmatrix} [-1.67, 6.38] \\ [-2.77, 6.40] \\ [-2.40, 3.40] \end{bmatrix}$	$\begin{bmatrix} [0, 6.38] \\ [0, 6.40] \\ [0, 3.40] \end{bmatrix}$
$\begin{bmatrix} [3.7, 4.3] & [-1.5, -0.5] & [0, 0] \\ [-1.5, -0.5] & [3.7, 4.3] & [-1.5, -0.5] \\ [0, 0] & [-1.5, -0.5] & [3.7, 4.3] \end{bmatrix}$	$\begin{bmatrix} [2, 14] \\ [-9, 3] \\ [-3, 1] \end{bmatrix}$	$\begin{bmatrix} [-1.09, 4.29] \\ [-4.02, 1.24] \\ [-2.44, 0.773] \end{bmatrix}$	$\begin{bmatrix} [-0.995, 5.01] \\ [-4.64, 1.52] \\ [-2.69, 1.38] \end{bmatrix}$	$\begin{bmatrix} [-1.09, 4.29] \\ [-3.79, 1.24] \\ [-2.35, 0.773] \end{bmatrix}$	$\begin{bmatrix} [-0.995, 5.27] \\ [-3.79, 3.66] \\ [-2.35, 1.75] \end{bmatrix}$
$\begin{bmatrix} [3.7, 4.3] & [-1.5, -0.5] & [0, 0] \\ [-1.5, -0.5] & [3.7, 4.3] & [-1.5, -0.5] \\ [0, 0] & [-1.5, -0.5] & [3.7, 4.3] \end{bmatrix}$	$\begin{bmatrix} [2, 14] \\ [3, 9] \\ [-3, 1] \end{bmatrix}$	$\begin{bmatrix} [0.517, 6.25] \\ [0.450, 6.07] \\ [-0.881, 2.73] \end{bmatrix}$	$\begin{bmatrix} [-0.206, 6.25] \\ [-0.386, 6.07] \\ [-2.01, 2.73] \end{bmatrix}$	$\begin{bmatrix} [0.523, 6.25] \\ [0.499, 6.07] \\ [-0.743, 2.73] \end{bmatrix}$	$\begin{bmatrix} [0.523, 6.25] \\ [0.499, 6.07] \\ [-0.743, 2.73] \end{bmatrix}$
$\begin{bmatrix} [15, 17] & [-3, 3.01] & [-3, 3.01] & [-3, 3.01] \\ [-3, 3.01] & [15, 17] & [-3, 2.99] & [-3, 2.99] \\ [-3, 2.99] & [-3, 2.99] & [15, 17] & [-3, 3.01] \\ [-3, 3.01] & [-3, 3.01] & [-3, 2.99] & [15, 17] \end{bmatrix}$	$\begin{bmatrix} [-6, -2] \\ [4, 5] \\ [-2, 4] \\ [8, 10] \end{bmatrix}$	$\begin{bmatrix} [-1.03, 0.495] \\ [-0.347, 0.974] \\ [-0.770, 0.917] \\ [0.150, 1.25] \end{bmatrix}$	$\begin{bmatrix} [-1.03, 0.363] \\ [-0.233, 0.975] \\ [-0.752, 0.919] \\ [0.149, 1.25] \end{bmatrix}$		$\begin{bmatrix} [-1.03, 0.761] \\ [-0.372, 0.974] \\ [-0.785, 0.917] \\ [0.051, 1.25] \end{bmatrix}$

The following steps are needed to solve the system:

STEP 1: Transform each equation of (4.4) to separable form to eliminate dependency among variables.

STEP 2: Find the piecewise linear enclosure of each of the univariate nonlinear functions and form a piecewise linear equation system.

STEP 3: Solve the piecewise linear system by the algorithm of Section 4.1.

STEP 4: If stopping criterion is met, stop. Otherwise, repeat from STEP 2 to STEP 4.

STEP 1: According to Yamamura’s algorithm [119], functions that are composed of four basic arithmetic operations (+, −, ×, /), unary operations (sin, exp, log, sqrt, etc.), and the power operation (∧) can be transformed to the separable form by introducing extra variables. For example, $f_1 \times f_2$ can be transformed to $(y^2 - f_1^2 - f_2^2) / 2$ and $y = f_1 + f_2$ by introducing $y = f_1 + f_2$. Similarly, f_1 / f_2 can be transformed to $(y^2 - f_1^2 - 1 / f_2^2) / 2$ and $y = f_1 + 1 / f_2$; and $(f_1)^{f_2}$ is symbolically equivalent to $\exp(y_1)$, and $y_1 = (y_2^2 - (\log(f_1))^2 - f_2^2) / 2$, $y_2 = \log(f_1) + f_2$. In geometric modeling, polynomial constraints, as shown in Table 1, can be easily transformed to the separable form.

After the transformation, system (4.4) is equivalent to

$$\sum_{j=0}^n f_{ij}(X_j) = 0 \quad (i = 1, 2, \dots, m), \tag{4.5}$$

where $X_j \in \mathbf{IR}$.

STEP 2: Piecewise linear enclosure of $f_{ij}(x_j)$ is found with the initial power interval of $\mathbf{PX}^{(1,1)} = \{X_j^{(0)}\}$ for each i and j as follows. Let the j th initial variable $X_j^{(0)} = [x_{j0}, x_{jp}]$ and $\mathbf{PX}^{(p,1)} = \{[x_{j0}, x_{j1}], [x_{j1}, x_{j2}], \dots, [x_{j(p-1)}, x_{jp}]\}$ for a p -piecewise linearization ($p \geq 2$), we have slopes

$$a_{ijk} = \frac{f_{ij}(x_{j(k+1)}) - f_{ij}(x_{jk})}{x_{j(k+1)} - x_{jk}} \quad (k = 0, 1, \dots, p - 1). \tag{4.6}$$

The *piecewise linear enclosure* of $f_{ij}(x_j)$ is defined as

$$E_{ij}(x) = \begin{cases} a_{ij0}x + B_{ij0} & x \in [x_{j0}, x_{j1}), \\ a_{ij1}x + B_{ij1} & x \in [x_{j1}, x_{j2}), \\ \vdots & \vdots \\ a_{ij(p-1)}x + B_{ij(p-1)} & x \in [x_{j(p-1)}, x_{jp}], \end{cases} \tag{4.7}$$

where $a_{ijk} \in \mathbf{R}$ and $B_{ijk} \in \mathbf{IR}$ such that

$$f_{ij}(x) \in E_{ij}(x) \quad \text{for } \forall x \in X_j^{(0)}, \tag{4.8}$$

as illustrated in Figure 6. Piecewise linear enclosure is able to find multiple solutions.

Geometric constraints usually are polynomials. Therefore $f_{ij}(x)$ is continuous and differentiable within interval $X_j^{(0)}$. To find a B_{ijk} with the minimum width with the given a_{ijk} , derivative of $f_{ij}(x)$ can be obtained symbolically. The problem is reduced to solve real value nonlinear equation and find solutions of

$$f'_{ij}(x) = a_{ijk}. \tag{4.9}$$

Given that $f_{ij}(x)$ is a univariate polynomial function or a function with unary operations for most geometric constraints, equations (4.9) have at least one solution. Roots can be isolated within disjointed intervals individually based on Descartes' rule of signs before equations are solved. Descartes' bound gives the upper bound of the numbers of positive and negative roots of a polynomial.

Let $P(x)$ be a polynomial with real coefficients, the following transformations are defined:

- (*Reverse transformation*): $R[P(x)] = x^n P(1/x)$ where n is the degree of P .
- (*Translation transformation*): $T_t[P(x)] = P(x + t)$ for $t \in \mathbf{R}$.

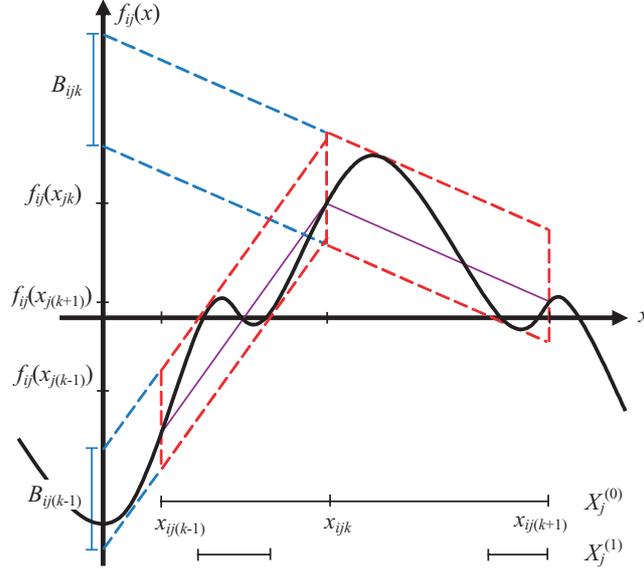


Figure 6. Piecewise linear enclosure of nonlinear interval function.

- (Homothetic transformation): $H_c[P(x)] = P(cx)$ for $c \in \mathbf{R}$.

Based on the algorithm of Collins et al. [20], [21], $P_{ij}(x)$ for $x \in X$ is transformed to $P_{ij}^0(x)$ for $x \in [0, 1]$ by $P_{ij}^0(x) = H_{b-a}[T_a[P_{ij}(x)]]$ where a is the lower bound of X while b is the upper bound of X . The roots of $P_{ij}(x)$ for $x \in X$ have one-to-one correspondence with the roots of $P_{ij}^0(x)$ for $x \in [0, 1]$. A list of root intervals or exact roots can be obtained by calling $RootIsolation(P_{ij}^0, 0, 0)$ listed in Figure 7. For each root interval or exact root with information of (depth, index) in the list, there is an corresponding $x \in \left[\frac{(b-a)\text{index}}{2^{\text{depth}}} + a, \frac{(b-a)(\text{index} + 1)}{2^{\text{depth}}} + a \right]$ for root intervals or $x = \frac{(b-a)\text{index}}{2^{\text{depth}}} + a$ for exact roots such that $P_{ij}(x) = 0$.

Interval $[x_{ijk}, x_{ij(k+1)})$ can be subdivided into small intervals bounding individual roots of (4.9). Let $g_{ijk}(x) = f'_{ij}(x) - a_{ijk}$. Solutions to (4.9) can be found by the Secant method

$$x^{(n+1)} = x^{(n)} - \frac{x^{(n)} - x^{(n-1)}}{g_{ijk}(x^{(n)}) - g_{ijk}(x^{(n-1)})} g_{ijk}(x^{(n)}) \quad n = 1, 2, 3, \dots \quad (4.10)$$

Suppose x_{ijks} ($s = 1, 2, \dots, S$) is the s th solution of (4.9), and $x_{ijk0} = x_{j0}$ as defined in (4.6). Let $B_{ijk} = [b_L^{ijk}, b_N^{ijk}, b_U^{ijk}]$, where

$$b_U^{ijk} = \max_s \{f_{ij}(x_{ijks}) - a_{ijk}x_{ijks}, s = 0, 1, \dots, S\}, \quad (4.11a)$$

$$b_N^{ijk} = f_{ij}(x_{ijk0}) - a_{ijk}x_{ijk0}, \quad (4.11b)$$

$$b_L^{ijk} = \min_s \{f_{ij}(x_{ijks}) - a_{ijk}x_{ijks}, s = 0, 1, \dots, S\}. \quad (4.11c)$$

```

INPUT: Polynomial P with n degree
       int depth
       int index
OUTPUT: RootIntervalList

IF P(0) = 0
    RootIntervalList.addExactRoot(depth, index)
ENDIF
IF P(1) = 0
    RootIntervalList.addExactRoot(depth, index+1)
ENDIF
Polynomial Q = T1[R(P)]
IF DecartesBound(Q) = 1
    RootIntervalList.addRootInterval(depth, index)
ELSEIF DecartesBound(Q) >= 2
    Polynomial P1 = 2nH1/2[P]
    RootIsolation(P1, depth+1, 2*index)
    Polynomial P2 = T1[P1]
    RootIsolation(P2, depth+1, 2*index+1)
ENDIF

```

Figure 7. RootIsolation procedure based on Descartes' rule of signs.

LEMMA 4.1.

$$\sum_{j=1}^n f_{ij}(X_j) \subseteq \sum_{j=1}^n E_{ij}(X_j) \quad \forall i = 1, 2, \dots, m. \quad (4.12)$$

Proof. From (4.8), we have for $\forall x \in [x_{ijk}, x_{ij(k+1)}]$,

$$f_{ij}(x) \in a_{ijk}x + B_{ijk} \quad (i = 1, \dots, m; j = 1, \dots, n). \quad (4.13)$$

Then, for $\forall x \in [x_{ij0}, x_{ijp}]$,

$$f_{ij}(x) \in \begin{cases} a_{ij0}x + B_{ij0} & x \in [x_{j0}, x_{j1}], \\ a_{ij1}x + B_{ij1} & x \in [x_{j1}, x_{j2}], \\ \vdots & \vdots \\ a_{ij(p-1)}x + B_{ij(p-1)} & x \in [x_{j(p-1)}, x_{jp}]. \end{cases} \quad (i = 1, \dots, m; j = 1, \dots, n) \quad (4.14)$$

Thus,

$$\sum_{j=1}^n f_{ij}(X_j) \subseteq \bigcup_{k=0}^{p-1} \sum_{j=1}^n (a_{ijk}X_j^k + B_{ij0}) \quad (i = 1, \dots, m), \quad (4.15)$$

where $X_j^k = [x_{jk}, x_{j(k+1)}]$ and $X_j = \bigcup_{k=0}^{p-1} X_j^k$. \square

STEP 3: Solving (4.5) thus is further reduced to solving piecewise linear enclosure

$$\sum_{j=1}^n (a_{ij}X_j + B_{ij}) = 0 \quad (i = 1, \dots, m)$$

iteratively, or

$$\sum_{j=1}^n a_{ij} X_j = - \sum_{j=1}^n B_{ij} \quad (i = 1, \dots, m), \quad (4.16)$$

where

$$a_{ij} = \begin{cases} a_{ij0} & x \in [x_{j0}, x_{j1}), \\ a_{ij1} & x \in [x_{j1}, x_{j2}), \\ \vdots & \vdots \\ a_{ij(p-1)} & x \in [x_{j(p-1)}, x_{jp}] \end{cases} \quad \text{and} \quad B_{ij} = \begin{cases} B_{ij0} & x \in [x_{j0}, x_{j1}), \\ B_{ij1} & x \in [x_{j1}, x_{j2}), \\ \vdots & \vdots \\ B_{ij(p-1)} & x \in [x_{j(p-1)}, x_{jp}]. \end{cases}$$

This linear system can be solved using the algorithm described in Section 4.1. Suppose Y_j is the j th variable solution of (4.16) in the t th iteration. By (4.17), the initial value of X_j in the $(t+1)$ -th iteration is calculated. If an empty interval is derived, the original system has no solution within the given intervals $(X_1^{(t)}, X_2^{(t)}, \dots, X_n^{(t)})$.

$$X_j^{(t+1)} = X_j^{(t)} \cap Y_j \quad \text{for } j = 1, 2, \dots, n. \quad (4.17)$$

STEP 4: When the stopping criterion is met, such as the width of intervals has no further improvement, or the intervals are sharp enough, the iteration is stopped. Otherwise, go back to (4.6) to find out the new linear enclosures within the updated intervals and repeat the procedure starting from STEP 2.

Remark 4.1. The piecewise linear enclosure is a special case of first-order Taylor model.

$$\mathbf{E}(\mathbf{F}, \mathbf{A}, \mathbf{X}^{\mathbf{I}}) = \mathbf{F}(0) + \mathbf{A}(\mathbf{X}^{\mathbf{I}} - 0) + \Delta^{\mathbf{I}} = \mathbf{A}\mathbf{X}^{\mathbf{I}} + \mathbf{B}^{\mathbf{I}}, \quad (4.18)$$

where $\mathbf{B}^{\mathbf{I}} = \mathbf{F}(0) + \Delta^{\mathbf{I}}$, $\mathbf{A}_{ij} = a_{ij}$, and $\mathbf{B}_i^{\mathbf{I}} = \sum_j B_{ij}$ as in (4.16).

THEOREM 4.1. *Let $K(\mathbf{X}^{\mathbf{I}})$ be the operator to solve the piecewise linear system $\mathbf{E}(\mathbf{F}, \mathbf{A}, \mathbf{X}^{\mathbf{I}}) = 0$. If $K(\mathbf{X}^{\mathbf{I}}) \subset \mathbf{X}^{\mathbf{I}}$, then there exists a point $\mathbf{x}^* = \mathbf{X}^{\mathbf{I}}$ such that $\mathbf{F}(\mathbf{x}^*) = 0$.*

Proof. From Lemma 4.1, we have $\mathbf{X}^{\mathbf{I}} \subseteq \mathbf{F}(\mathbf{X}^{\mathbf{I}}) \subseteq \mathbf{E}(\mathbf{F}, \mathbf{A}, \mathbf{X}^{\mathbf{I}})$. The combination of interval splitting in piecewise enclosure and Gauss-Seidel method ensures $K(\mathbf{X}^{\mathbf{I}}) \subset \mathbf{X}^{\mathbf{I}}$. Therefore, the Brouwer fixed point theorem implies that there is a point $\mathbf{x}^* = \mathbf{X}^{\mathbf{I}}$ such that $\mathbf{F}(\mathbf{x}^*) = 0$. \square

With extended interval arithmetic [36], [51] in which division by zero is allowed, interval Newton methods elegantly split intervals containing multiple solutions. The piecewise linear enclosure method works in the same way to divide intervals and bound unique solutions individually. A simple example is bisection of two-piece linear enclosure.

LEMMA 4.2 [55]. *If $f : X \rightarrow \mathbf{R}$ ($X \in \mathbf{IR}$) is a smooth function and its linear enclosure is $E(X) = aX + B$, then $\lim_{\text{wid}(X) \rightarrow 0} \frac{\text{wid}(B)}{(\text{wid}(X))^2} = \alpha$.*

Proof. Suppose that $X^{(T)} = [x_L, x_U] \subseteq X^{(0)}$ at T th iteration. Let $a = \frac{f(x_U) - f(x_L)}{x_U - x_L}$ and $B = [b_L, b_U]$. There exists an $x_s \in X^{(T)}$ such that $f'(x_s) = a$. Assume that $f(x)$ ($x \in X$) is convex. Other cases can be proved similarly. $b_U = f(x_L) - ax_L$ and $b_L = f(x_s) - ax_s$. Then $\text{wid}(B) = f(x_L) - f(x_s) - a(x_L - x_s)$.

Because of $f(x) = f(x_s) + f'(x_s)(x - x_s) + \frac{1}{2}f''(x_s)(x - x_s)^2 + \frac{1}{6}f'''(\xi)(x - x_s)^3$, we have $f(x_L) = f(x_s) + f'(x_s)(x_L - x_s) + \frac{1}{2}f''(x_s)(x_L - x_s)^2 + \frac{1}{6}f'''(\xi)(x_L - x_s)^3$. That is $f(x_L) - f(x_s) - a(x_L - x_s) = \frac{1}{2}f''(x_s)(x_L - x_s)^2 + \frac{1}{6}f'''(\xi)(x_L - x_s)^3$. Thus $\text{wid}(B) \approx \frac{1}{2}f''(x_s)(\text{wid}(X))^2 + O[(\text{wid}(X))^3]$ as $\text{wid}(X) \rightarrow 0$. Therefore, $\lim_{\text{wid}(X) \rightarrow 0} \frac{\text{wid}(B)}{(\text{wid}(X))^2} = \alpha$. \square

THEOREM 4.2. *Suppose $\mathbf{F} : \mathbf{X}^{\mathbf{I}} \rightarrow \mathbf{R}^n$ ($\mathbf{X}^{\mathbf{I}} \in \mathbf{IR}^n$) is a smooth function such that, if $\mathbf{F}(\mathbf{x}^*) = 0$, then $\mathbf{x}^* \notin \partial \mathbf{X}^{\mathbf{I}}$. $K(\mathbf{X}^{\mathbf{I}})$ is the operator to solve the piecewise linear system $\mathbf{E}(\mathbf{F}, \mathbf{A}, \mathbf{X}^{\mathbf{I}}) = 0$. If $\mathbf{A}^T \mathbf{A}$ is nonsingular and $K(\mathbf{X}^{\mathbf{I}}) \subset \mathbf{X}^{\mathbf{I}}$, then there exists unique solutions such that $\mathbf{F}(\mathbf{x}) = 0$.*

Proof. Because of the inherent interval splitting procedure, the piecewise linear enclosure method will isolate zeros in individual boxes during iteration. Let $(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T = \mathbf{P}$ as in (4.16) and $\underline{p} = \min_{i,j} |\mathbf{P}_{ij}|, \bar{p} = \max_{i,j} |\mathbf{P}_{ij}|$. For $\tilde{X}_k = \sum_{i=1}^m \mathbf{P}_{ki} \sum_{j=1}^n B_{ij}$, from Lemma 4.2, we have

$$\begin{aligned} \lim_{\text{wid}(X_k) \rightarrow 0} \frac{\text{wid}(\tilde{X}_k)}{(\text{wid}(X_k))^2} &= \lim_{\text{wid}(X_k) \rightarrow 0} \frac{m \times [\underline{p}, \bar{p}] \times \text{wid}\left(\sum_{j=1}^n B_{ij}\right)}{(\text{wid}(X_k))^2} \\ &= \lim_{\text{wid}(X_k) \rightarrow 0} \frac{mn[\underline{p}, \bar{p}]\text{wid}(B_{ij})}{(\text{wid}(X_k))^2} = [mn\alpha\underline{p}, mn\alpha\bar{p}], \end{aligned}$$

which indicates quadratic convergence. \square

Remark 4.2. Even if $\mathbf{A}^T \mathbf{A}$ is singular, the convergence of smooth functions is quadratic given the condition $K(\mathbf{X}^{\mathbf{I}}) \subset \mathbf{X}^{\mathbf{I}}$.

Remark 4.3. In (4.7), if a \hat{B}_{ijk} can be estimated without the calculation of derivative as described in STEP 2 such that $B_{ijk} \subseteq \hat{B}_{ijk}$, this algorithm can be simplified and $F(X)$ can be non-smooth functions. However, certain conditions need to be satisfied to ensure quadratic convergence. The interval splitting only provides linear convergence.

An example [51, p. 57] is used to show the quadratic onvergence of the algorithm, as shown in Table 3.

Table 3. An example to illustrate convergence.

<i>Constraints:</i>		<i>Iteration</i> (<i>T</i>)	$\sum_{j=0}^1 \text{wid}(X_j^{(T)})$	$\frac{\sum_{j=0}^1 \text{wid}(X_j^{(T)})}{\left(\sum_{j=0}^1 \text{wid}(X_j^{(T-1)})\right)^2}$
$\begin{cases} x_1^2 - 4x_2 = 0, \\ x_2^2 - 2x_1 + 4x_2 = 0 \end{cases}$				
<i>Initial values:</i>	<i>Solution:</i>	1	1.25	0.078125
$\begin{cases} X_1 = [-1, 1], \\ X_2 = [-1, 1] \end{cases}$	$\begin{cases} x_1 = 0, \\ x_2 = 0 \end{cases}$	2	0.78125	0.5
		3	0.22049	0.36125
		4	0.0170582	0.350878
		5	9.82161e-005	0.337535
		6	3.2185e-009	0.333647
		7	3.45293e-018	0.333335
		8	3.97423e-036	0.333333
		9	5.26484e-072	0.333333
		10	9.23951e-144	0.333333
		11	2.84562e-287	0.333333

4.3. INTERVAL INEQUALITIES

With preconditioning, inequalities can be solved by the methods for equations. Consider a set of linear or nonlinear inequalities

$$F_i(\mathbf{X}^I) \leq C_i \quad (i = 1, \dots, l), \tag{4.19}$$

where $\mathbf{X}^I \in \mathbf{IR}^n$ and $C_i \in \mathbf{IR}$, inequalities are transformed to equations

$$F_i(\mathbf{X}^I) + S_i = C_i \quad (i = 1, \dots, l), \tag{4.20}$$

where $S_i \in \mathbf{IR}$ is a slack variable with initial value of $[0, 0, +\infty]$. Similarly,

$$F_i(\mathbf{X}^I) \geq C_i \quad (i = 1, \dots, l) \tag{4.21}$$

can be transformed to

$$F_i(\mathbf{X}^I) + S_i = C_i \quad (i = 1, \dots, l), \tag{4.22}$$

where $S_i \in \mathbf{IR}$ has initial value of $[-\infty, 0, 0]$. Inequalities can be easily integrated into systems of equalities, which is a good property of interval constraint representation.

Table 4 shows an example when inequality is introduced in the example of Table 3.

5. Design Refinement

One important aspect related to interval representation is the over estimation of allowance. Design refinement is needed to generate more delicate design if desirable

Table 4. The result when inequality is introduced in the example of Table 3.

<i>Constraints:</i>	<i>Preconditioned:</i>	<i>Iteration</i>	$\sum_{j=0}^2 \text{wid}(X_j^{(T)})$	$\frac{\sum_{j=0}^2 \text{wid}(X_j^{(T)})}{\left(\sum_{j=0}^2 \text{wid}(X_j^{(T-1)})\right)^2}$
$\begin{cases} x_1^2 - 4x_2 = 0, \\ x_2^2 - 2x_1 + 4x_2 = 0, \\ x_1 \leq x_2 \end{cases}$	$\begin{cases} x_1^2 - 4x_2 = 0, \\ x_2^2 - 2x_1 + 4x_2 = 0, \\ x_1 - x_2 + s_1 = 0 \end{cases}$	(T)		
<i>Initial values:</i>	<i>Solution:</i>	1	0.75	7.44036e-007
$\begin{cases} X_1 = [-1, 1], \\ X_2 = [-1, 1], \\ X_3 = [0, 1000] \end{cases}$	$\begin{cases} x_1 = 0, \\ x_2 = 0, \\ x_3 = 0 \end{cases}$	2	0.046875	0.0833333
		3	0.000183105	0.0833333
		4	2.79397e-009	0.0833333
		5	6.50521e-019	0.0833333
		6	3.52648e-038	0.0833333
		7	1.03634e-076	0.0833333
		8	8.95001e-154	0.0833333
		9	6.67522e-308	0.0833333

details have not been achieved yet. There are two ways to refine design: *interval subdivision* and *constraint respecification*. Interval subdivision is to divide existing interval boxes into unions of subintervals to achieve a refined view of current design. Constraint respecification is to modify some of constraints or to add extra valid constraints to contract feasible regions.

5.1. INTERVAL SUBDIVISION

Interval subdivision (also called *subpaving*) substitutes an interval vector with multiple interval vectors such that the corresponding real space region is subdivided into multiple smaller regions to cover the actual solution set more compactly. Interval vector \mathbf{X} can be bisected recursively and subintervals are tested individually if they contain the actual solution set. The actual solution set then is approximated by the union of subinterval regions.

Consider a design problem $f(\mathbf{X}) = \mathbf{Y}$. The target is to find the actual solution set $\mathbf{S} \subseteq \mathbf{X}$ with the minimal size such that $f(\mathbf{S}) = \mathbf{Y}$. Interval arithmetic only gives a valid solution \mathbf{D} with $f(\mathbf{D}) \supseteq \mathbf{Y}$. If the valid solution is represented by power intervals, refinement is degree elevation of power intervals. If the original solution to a problem is found as an n -dimensional vector $\mathbf{X} = [X_1, X_2, \dots, X_n]$ with the corresponding power interval $\mathbf{P}_{(0)}^{(1,n)}$. One elevation operation will bisect \mathbf{X} , with each interval vector being deleted and new subintervals inserted. Feasibility of each new subinterval then is tested. For a m -step n -dimensional \mathbf{X} subdivision, the number of intervals generated is an order of $O(2^{m \times n})$, which is computationally inefficient. A better way to contract a solution is to modify or add valid constraints to generate a new solution with narrower feasible regions.

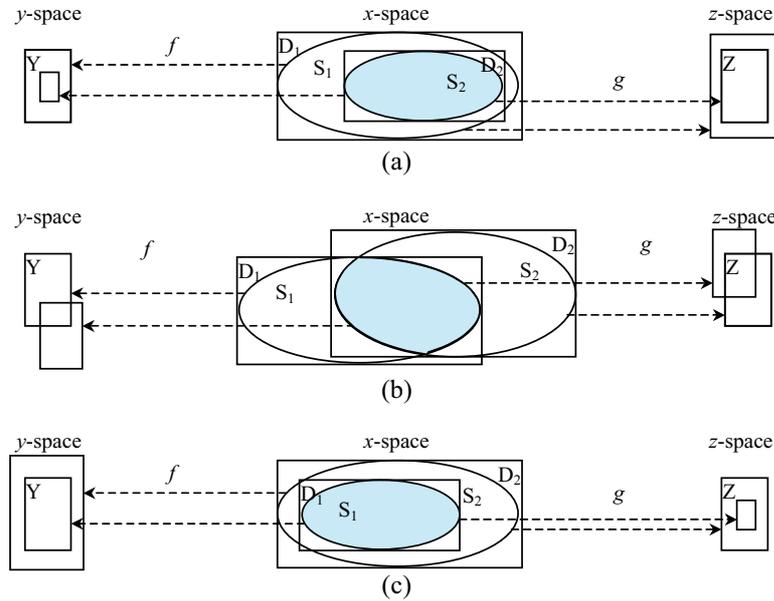


Figure 8. Relations of two constraint subsets.

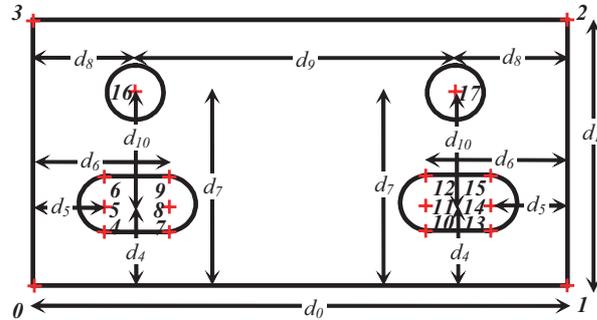
5.2. CONSTRAINT RESPECIFICATION

Constraint modification depends on sensitivity analysis, while adding constraints is largely dependent on user's specification. Feasibility and effectiveness should be considered simultaneously when changing constraints. One basic issue in sensitivity analysis is how to differentiate *active* and *inactive* constraints. Active constraints scope the actual range of solution, while inactive constraints have certain level of slackness. At the beginning of interval computation, all constraints are active if a sufficiently large initial region is given. As the iteration proceeds, some constraints become inactive. The decision of which constraints to be modified is based on the selection of active constraints.

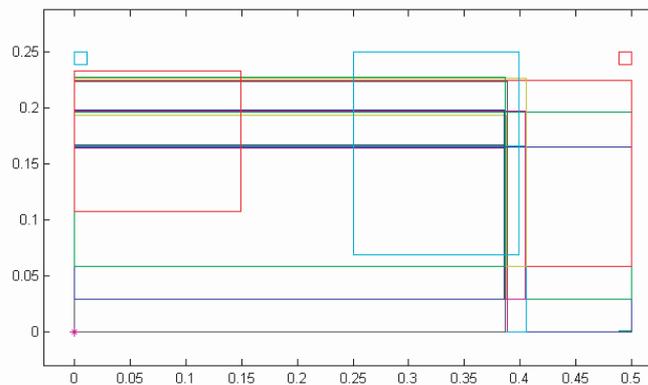
THEOREM 5.1. *For a constraint set $p = \{f(\mathbf{X}) = \mathbf{Y} \text{ and } g(\mathbf{X}) = \mathbf{Z}\}$, the subset $f(\mathbf{X}) = \mathbf{Y}$ with respect to a solution $\mathbf{D} \subset \mathbf{X}$ is inactive if $f(\mathbf{D}) \subset \mathbf{Y}$ and $g(\mathbf{D}) \supseteq \mathbf{Z}$.*

Proof. Suppose \mathbf{S}_1 and \mathbf{S}_2 are actual solution sets of f and g respectively, and \mathbf{S} is the actual solution set of p . Given that $f(\mathbf{S}_1) = \mathbf{Y}$ and $f(\mathbf{D}) \subset \mathbf{Y}$, because of the inclusion monotonicity, $\mathbf{S}_1 \supset \mathbf{D}$. Similarly, $\mathbf{D} \supseteq \mathbf{S}_2$. Thus, $\mathbf{S}_1 \supset \mathbf{S}_2$. \square

As illustrated by Figure 8, subset f is inactive and g is active in case (a); both f and g are active in case (b); and f is active and g is inactive in case (c).



(a) design problem with 50 constraints and 36 variables



(b) variational bounds of 18 characteristic points in the solution with 100 variables and 110 constraints in separable form

Figure 9. An over-constrained bracket design problem.

6. A Numerical Example

The NICR kernel is implemented in C++ with an object-oriented programming style. The kernel includes the fundamental structure and arithmetic operations of the nominal intervals. It also includes the implementation of the methods described in previous sections for solving linear and nonlinear constraint systems.

As a demonstration, the design of the bracket in Figure 4 is used as a numerical example. As shown in Figure 9(a), there are 18 characteristic points therefore 36 design variables in this problem. 50 constraints are assigned to the variables, which is over-constrained in the sense of traditional parametric modeling. Commercial CAD systems usually use constructive constraint solving methods. Large problems are decomposed into small ones. It is unusual and unrealistic to solve very large numerical systems directly. After the transformation to separable form, the bracket design problem has 100 variables and 110 constraints, as listed in Appendices. This

Table 5. Width reduction of 100 variables in the bracket example.

$Iter.$ (T)	$\sum_{j=0}^{99} \text{wid}(X_j^{(T)}) - \sum_{j=0}^{99} \text{wid}(X_j^{(T-1)})$	$Iter.$ (T)	$\sum_{j=0}^{99} \text{wid}(X_j^{(T)}) - \sum_{j=0}^{99} \text{wid}(X_j^{(T-1)})$
1	188.135	26	0.037213
2	12.0089	27	0.02771
3	3.09879	28	0.020351
4	3.87703	29	0.014782
5	3.16818	30	0.010643
6	1.83084	31	0.007607
7	1.28761	32	0.005405
8	0.892381	33	0.003821
9	0.541563	34	0.00269
10	0.502575	35	0.001887
11	0.394768	36	0.001319
12	0.313836	37	0.00092
13	0.286785	38	0.00064
14	0.231169	39	0.000445
15	0.156597	40	0.000308
16	0.127431	41	0.000213
17	0.130038	42	0.000147
18	0.138253	43	0.000102
19	0.128812	44	7.00e-05
20	0.115669	45	4.82e-05
21	0.108886	46	3.31e-05
22	0.09546	47	2.28e-05
23	0.079314	48	1.56e-05
24	0.06334	49	1.07e-05
25	0.049103	50	7.35e-06

problem is solved by the NICR kernel, and the result is shown in Figure 9(b). The reduction of total widths for the 100 variables is shown in Table 5 and Figure 10.

7. Conclusion

This paper presents a soft constraint representation scheme based on nominal interval for parametric CAD systems. Interval geometric parameters capture inexactness of conceptual and embodiment design, uncertainty in detail design, as well as boundary information for design optimization. To accommodate under-constrained and over-constrained design problems, a double-loop Gauss-Seidel method is developed to solve linear constraints. A symbolic preconditioning procedure transforms non-linear equations to separable form. Inequalities are also transformed and integrated with equalities.

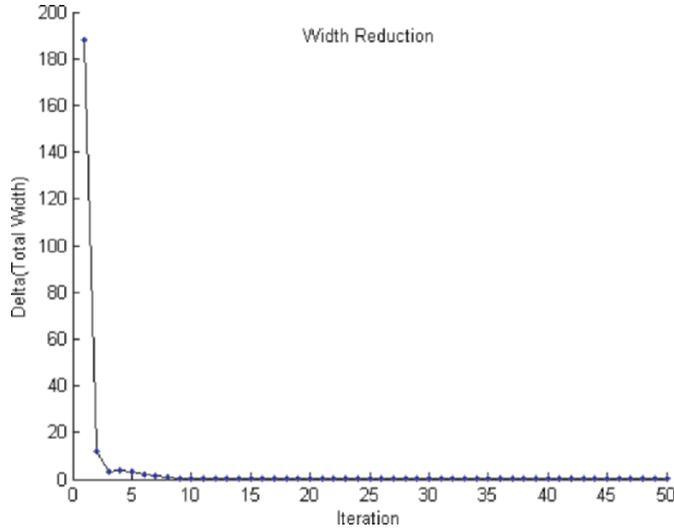


Figure 10. Width reduction in the bracket example.

Nonlinear constraints can be bounded by piecewise linear enclosures and solved by linear methods iteratively. A sensitivity analysis method that differentiates active and inactive constraints is presented for design refinement.

Acknowledgements

This research was supported in part by the Office of Naval Research, U.S. Department of Defense (Grant Number: N00014-02-1-0649). Authors would like to thank referees for the incisive comments and suggestions.

Appendix

A. 50 Constraints and 36 variables of the example in Section 6

$$\begin{aligned}
 x_0 &= a_0, & y_0 &= b_0, & y_0 - y_1 &= 0, \\
 (x_1 - x_0)^2 + (y_1 - y_0)^2 &= d_0^2, \\
 (x_2 - x_1)^2 + (y_2 - y_1)^2 &= d_1^2, \\
 (x_3 - x_2)^2 + (y_3 - y_2)^2 &= d_2^2, \\
 (x_0 - x_3)^2 + (y_0 - y_3)^2 &= d_3^2, \\
 (x_0 - x_3)(x_1 - x_0) + (y_0 - y_3)(y_1 - y_0) &= 0, \\
 (x_2 - x_1)(x_1 - x_0) + (y_2 - y_1)(y_1 - y_0) &= 0, \\
 x_0 \leq x_1, & \quad y_0 \leq y_3, & \quad x_0 \leq x_5, & \quad y_0 \leq y_5, \\
 [(y_1 - y_0)(x_5 - x_0) - (x_1 - x_0) + (y_5 - y_0)]^2 - d_4^2[(y_1 - y_0)^2 + (x_1 - x_0)^2] &= 0,
 \end{aligned}$$

$$\begin{aligned}
& [(y_3 - y_0)(x_5 - x_0) - (x_3 - x_0) + (y_5 - y_0)]^2 - d_5^2[(y_3 - y_0)^2 + (x_3 - x_0)^2] = 0, \\
& [(y_1 - y_0)(x_8 - x_0) - (x_1 - x_0) + (y_8 - y_0)]^2 - d_4^2[(y_1 - y_0)^2 + (x_1 - x_0)^2] = 0, \\
& [(y_3 - y_0)(x_8 - x_0) - (x_3 - x_0) + (y_8 - y_0)]^2 - d_6^2[(y_3 - y_0)^2 + (x_3 - x_0)^2] = 0, \\
& [(y_1 - y_0)(x_{11} - x_0) - (x_1 - x_0) + (y_{11} - y_0)]^2 - d_4^2[(y_1 - y_0)^2 + (x_1 - x_0)^2] = 0, \\
& [(y_2 - y_1)(x_{11} - x_0) - (x_2 - x_1) + (y_{11} - y_0)]^2 - d_6^2[(y_2 - y_1)^2 + (x_2 - x_1)^2] = 0, \\
& [(y_1 - y_0)(x_{14} - x_0) - (x_1 - x_0) + (y_{14} - y_0)]^2 - d_4^2[(y_1 - y_0)^2 + (x_1 - x_0)^2] = 0, \\
& [(y_2 - y_1)(x_{14} - x_0) - (x_2 - x_1) + (y_{14} - y_0)]^2 - d_5^2[(y_2 - y_1)^2 + (x_2 - x_1)^2] = 0, \\
& [(y_1 - y_0)(x_{16} - x_0) - (x_1 - x_0) + (y_{16} - y_0)]^2 - d_7^2[(y_1 - y_0)^2 + (x_1 - x_0)^2] = 0, \\
& [(y_3 - y_0)(x_{16} - x_0) - (x_3 - x_0) + (y_{16} - y_0)]^2 - d_8^2[(y_3 - y_0)^2 + (x_3 - x_0)^2] = 0, \\
& [(y_1 - y_0)(x_{17} - x_0) - (x_1 - x_0) + (y_{17} - y_0)]^2 - d_7^2[(y_1 - y_0)^2 + (x_1 - x_0)^2] = 0, \\
& [(y_2 - y_1)(x_{17} - x_0) - (x_2 - x_1) + (y_{17} - y_0)]^2 - d_8^2[(y_2 - y_1)^2 + (x_2 - x_1)^2] = 0, \\
& x_4 - x_5 = 0, \quad x_6 - x_5 = 0, \quad x_7 - x_8 = 0, \\
& x_9 - x_8 = 0, \quad x_{10} - x_{11} = 0, \quad x_{12} - x_{11} = 0, \\
& x_{13} - x_{14} = 0, \quad x_{15} - x_{14} = 0, \quad y_4 - y_7 = 0, \\
& y_6 - y_9 = 0, \quad y_{10} - y_{13} = 0, \quad y_{12} - y_{15} = 0, \\
& y_9 - y_{12} = 0, \quad y_7 - y_{10} = 0, \quad y_5 - y_4 - r = 0, \\
& y_6 - y_5 - r = 0, \quad y_8 - y_7 - r = 0, \quad y_9 - y_8 - r = 0, \\
& y_{11} - y_{10} - r = 0, \quad y_{12} - y_{11} - r = 0, \quad y_{14} - y_{13} - r = 0, \\
& y_{15} - y_{14} - r = 0, \quad y_{17} - x_{16} - d_9 = 0, \\
& y_{16} - y_5 - d_{10} = 0, \quad y_{17} - y_{11} - d_{10} = 0.
\end{aligned}$$

B. 110 Constraints and 100 variables of the example in Section 6 after the transformation to separable form

$$\begin{aligned}
& x_0 = a_0, \quad y_0 = b_0, \quad y_0 - y_1 = 0, \\
& (u_1)^2 + (v_1)^2 = d_0^2, \quad u_1 = x_1 - x_0, \quad v_1 = y_1 - y_0, \\
& (u_2)^2 + (v_2)^2 = d_1^2, \quad u_2 = x_2 - x_1, \quad v_2 = y_2 - y_1, \\
& (u_3)^2 + (v_3)^2 = d_2^2, \quad u_3 = x_3 - x_2, \quad v_3 = y_3 - y_2, \\
& (u_4)^2 + (v_4)^2 = d_3^2, \quad u_4 = x_3 - x_0, \quad v_4 = y_3 - y_0, \\
& .5(p_1)^2 - .5(u_4)^2 - .5(u_1)^2 + .5(q_1)^2 - .5(v_4)^2 - .5(v_1)^2 = 0, \\
& p_1 = u_4 + u_1, \quad q_1 = v_4 + v_1, \\
& .5(p_2)^2 - .5(u_2)^2 - .5(u_1)^2 + .5(q_2)^2 - .5(v_2)^2 - .5(v_1)^2 = 0, \\
& p_2 = u_2 + u_1, \quad q_2 = v_2 + v_1, \\
& x_0 \leq x_1, \quad y_0 \leq y_3, \quad x_0 \leq x_5, \quad y_0 \leq y_5, \\
& u_5 = x_5 - x_0, \quad v_5 = y_5 - y_0, \\
& [w_1]^2 - d_4^2[(v_1)^2 + (u_1)^2] = 0,
\end{aligned}$$

$$\begin{aligned}
w_1 &= .5(p_3)^2 - .5(v_1)^2 - .5(u_5)^2 - .5(q_3)^2 + .5(u_1)^2 + .5(v_5)^2, \\
p_3 &= v_1 + u_5, \quad q_3 = u_1 + v_5, \\
[w_2]^2 - d_5^2[(v_4)^2 + (u_4)^2] &= 0, \\
w_2 &= .5(p_4)^2 - .5(v_4)^2 - .5(u_5)^2 - .5(q_4)^2 + .5(u_4)^2 + .5(v_5)^2, \\
p_4 &= v_4 + u_5, \quad q_4 = u_4 + v_5, \\
u_6 &= x_8 - x_0, \quad v_6 = y_8 - y_0, \\
[w_3]^2 - d_4^2[(v_1)^2 + (u_1)^2] &= 0, \\
w_3 &= .5(p_5)^2 - .5(v_1)^2 - .5(u_6)^2 - .5(q_5)^2 + .5(u_1)^2 + .5(v_6)^2, \\
p_5 &= v_1 + u_6, \quad q_5 = u_1 + v_6, \\
[w_4]^2 - d_6^2[(v_4)^2 + (u_4)^2] &= 0, \\
w_4 &= .5(p_6)^2 - .5(v_4)^2 - .5(u_6)^2 - .5(q_6)^2 + .5(u_4)^2 + .5(v_6)^2, \\
p_6 &= v_4 + u_6, \quad q_6 = u_4 + v_6, \\
u_7 &= x_{11} - x_0, \quad v_7 = y_{11} - y_0, \\
[w_5]^2 - d_4^2[(v_1)^2 + (u_1)^2] &= 0, \\
w_5 &= .5(p_7)^2 - .5(v_1)^2 - .5(u_7)^2 - .5(q_7)^2 + .5(u_1)^2 + .5(v_7)^2, \\
p_7 &= v_1 + u_7, \quad q_7 = u_1 + v_7, \\
[w_6]^2 - d_6^2[(v_2)^2 + (u_2)^2] &= 0, \\
w_6 &= .5(p_8)^2 - .5(v_2)^2 - .5(u_7)^2 - .5(q_8)^2 + .5(u_2)^2 + .5(v_7)^2, \\
p_8 &= v_2 + u_7, \quad q_8 = u_2 + v_7, \\
u_8 &= x_{14} - x_0, \quad v_8 = y_{14} - y_0, \\
[w_7]^2 - d_4^2[(v_1)^2 + (u_1)^2] &= 0, \\
w_7 &= .5(p_9)^2 - .5(v_1)^2 - .5(u_8)^2 - .5(q_9)^2 + .5(u_1)^2 + .5(v_8)^2, \\
p_9 &= v_1 + u_8, \quad q_9 = u_1 + v_8, \\
[w_8]^2 - d_5^2[(v_2)^2 + (u_2)^2] &= 0, \\
w_8 &= .5(p_{10})^2 - .5(v_2)^2 - .5(u_8)^2 - .5(q_{10})^2 + .5(u_2)^2 + .5(v_8)^2, \\
p_{10} &= v_2 + u_8, \quad q_{10} = u_2 + v_8, \\
u_9 &= x_{16} - x_0, \quad v_9 = y_{16} - y_0, \\
[w_9]^2 - d_7^2[(v_1)^2 + (u_1)^2] &= 0, \\
w_9 &= .5(p_{11})^2 - .5(v_1)^2 - .5(u_9)^2 - .5(q_{11})^2 + .5(u_1)^2 + .5(v_9)^2, \\
p_{11} &= v_1 + u_9, \quad q_{11} = u_1 + v_9, \\
[w_{10}]^2 - d_8^2[(v_4)^2 + (u_4)^2] &= 0, \\
w_{10} &= .5(p_{12})^2 - .5(v_4)^2 - .5(u_9)^2 - .5(q_{12})^2 + .5(u_4)^2 + .5(v_9)^2, \\
p_{12} &= v_4 + u_9, \quad q_{12} = u_4 + v_9, \\
u_{10} &= x_{17} - x_0, \quad v_{10} = y_{17} - y_0, \\
[w_{11}]^2 - d_7^2[(v_1)^2 + (u_1)^2] &= 0,
\end{aligned}$$

$$\begin{aligned}
w_{11} &= .5(p_{13})^2 - .5(v_1)^2 - .5(u_{10})^2 - .5(q_{13})^2 + .5(u_1)^2 + .5(v_{10})^2, \\
p_{13} &= v_1 + u_{10}, & q_{13} &= u_1 + v_{10}, \\
[w_{12}]^2 - d_8^2[(v_2)^2 + (u_2)^2] &= 0, \\
w_{12} &= .5(p_{14})^2 - .5(v_2)^2 - .5(u_{10})^2 - .5(q_{14})^2 + .5(u_2)^2 + .5(v_{10})^2, \\
p_{14} &= v_2 + u_{10}, & q_{14} &= u_2 + v_{10}, \\
x_4 - x_5 &= 0, & x_6 - x_5 &= 0, & x_7 - x_8 &= 0, \\
x_9 - x_8 &= 0, & x_{10} - x_{11} &= 0, & x_{12} - x_{11} &= 0, \\
x_{13} - x_{14} &= 0, & x_{15} - x_{14} &= 0, & y_4 - y_7 &= 0, \\
y_6 - y_9 &= 0, & y_{10} - y_{13} &= 0, & y_{12} - y_{15} &= 0, \\
y_9 - y_{12} &= 0, & y_7 - y_{10} &= 0, & y_5 - y_4 - r &= 0, \\
y_6 - y_5 - r &= 0, & y_8 - y_7 - r &= 0, & y_9 - y_8 - r &= 0, \\
y_{11} - y_{10} - r &= 0, & y_{12} - y_{11} - r &= 0, & y_{14} - y_{13} - r &= 0, \\
y_{15} - y_{14} - r &= 0, & y_{17} - x_{16} - d_9 &= 0, \\
y_{16} - y_5 - d_{10} &= 0, & y_{17} - y_{11} - d_{10} &= 0.
\end{aligned}$$

References

1. Aldefeld, B.: Variation of Geometries Based on a Geometric-Reasoning Method, *Computer-Aided Design* **20** (3) (1988), pp. 117–126.
2. Aldefeld, B.: Rule-Based Approach to Variational Geometry, in: Smith, A. (ed.), *Knowledge Engineering and Computer Modelling in CAD, Proceedings of the 7th International Conference on the Computer as a Design Tool, September 2–5, 1986*, London, pp. 59–67.
3. Alefeld, G.: Bounding the Slope of Polynomial Operators and Some Applications, *Computing* **26** (1981), pp. 227–237.
4. Alefeld, G.: On the Convergence of Some Interval-Arithmetic Modifications of Newton's Method, *SIAM Journal on Numerical Analysis* **21** (2) (1984), pp. 363–372.
5. Alefeld, G. and Herzberger, J.: *Introduction to Interval Computations*, Academic Press, New York, 1983.
6. Alefeld, G. and Mayer, G.: Interval Analysis: Theory and Application, *Journal of Computational and Applied Mathematics* **121** (1–2) (2000), pp. 421–464.
7. Alefeld, G. and Platzöder, L.: A Quadratically Convergent Krawczyk-Like Algorithm, *SIAM Journal on Numerical Analysis* **20** (1) (1983), pp. 210–219.
8. Anantha, R., Kramer, G. A., and Crawford, R. H.: Assembly Modelling by Geometric Constraint Satisfaction, *Computer-Aided Design* **28** (9) (1996), pp. 707–722.
9. Benhamou, F. and Granvilliers, L.: Automatic Generation of Numerical Redundancies for Non-Linear Constraint Solving, *Reliable Computing* **3** (3) (1997), pp. 335–344.
10. Berz, M.: *Modern Map Methods in Particle Beam Physics*, Academic Press, San Diego, 1999.
11. Berz, M. and Hoffstätter, G.: Computation and Application of Taylor Polynomials with Interval Remainder Bounds, *Reliable Computing* **4** (1) (1998), pp. 83–97.
12. Blied, C.: *Computer Methods for Design Automation*, unpublished Ph.D. thesis, Massachusetts Institute of Technology, 1992.
13. Bouma, W., Fudos, I., Hoffmann, C., Cai, J., and Paige, R.: Geometric Constraint Solver, *Computer-Aided Design* **27** (6) (1995), pp. 487–501.
14. Buchberger, B., Collins, G., and Kutzler, B.: Algebraic Methods for Geometric Reasoning, *Annual Review of Computer Science* **3** (1988), pp. 85–120.
15. Ceberio, M. and Granvilliers, L.: Horner's Rule for Interval Evaluation Revisited, *Computing* **69** (1) (2002), pp. 51–81.

16. Chen, F. and Lou, W.: Degree Reduction of Interval Bezier Curves, *Computer-Aided Design* **32** (10) (2000), pp. 571–582.
17. Chen, X.: A Verification Method for Solutions of Nonsmooth Equations, *Computing* **58** (1997), pp. 281–294.
18. Chiu, C.-K. and Lee, J. H.-M.: Efficient Interval Linear Equality Solving in Constraint Logic Programming, *Reliable Computing* **8** (2) (2002), pp. 139–174.
19. Chyz, W.: *Constraint Management for CSG*, Unpublished Master Thesis, Massachusetts Institute of Technology, 1985.
20. Collins, G. E. and Akritas, A. G.: Polynomial Real Root Isolation Using Descartes's Rule of Signs, in: *Proceedings of the Third ACM Symposium on Symbolic and Algebraic Computation, August 10–12, 1976, Yorktown Heights, New York*, pp. 272–275.
21. Collins, G. E. and Johnson, J. R.: Quantifier Elimination and the Sign Variation Method for Real Root Isolation, in: *Proceedings of the ACM-SIGSAM 1989 International Symposium on Symbolic and Algebraic Computation, July 17–19, 1989 Portland, Oregon*, pp. 264–271.
22. Duff, T.: Interval Arithmetic and Recursive Subdivision for Implicit Functions and Constructive Solid Geometry, *Computer Graphics* **26** (2) (1992), pp. 131–138.
23. Finch, W. W. and Ward, A. C.: A Set-Based System for Eliminating Infeasible Designs in Engineering Problems Dominated by Uncertainty, in: *ASME Proceedings of DETC97/dtm-3886, Sept. 14–17, 1997, Sacramento, CA, USA*.
24. Fudos, I. and Hoffmann, C. M.: A Graph-Constructive Approach to Solving Systems of Geometric Constraints, *ACM Transactions on Graphics* **16** (2) (1997), pp. 179–216.
25. Gao, X.-S. and Chou, S.-C.: Solving Geometric Constraint Systems II: A Symbolic Approach and Decision of Rc-constructibility, *Computer Aided-Design* **30** (2) (1998), pp. 115–122.
26. Garloff, J.: The Bernstein Algorithm, *Interval Computations* (2) (1993), pp. 154–168.
27. Ge, J.-X., Chou, S.-C., and Gao, X.-S.: Geometric Constraint Satisfaction Using Optimization Methods, *Computer-Aided Design* **31** (14) (1999), pp. 867–879.
28. Gossard, D. C., Zuffante, R. P., and Sakurai, H.: Representing Dimensions, Tolerances, and Features in MCAE Systems, *IEEE Computer Graphics & Applications* **8** (3) (1988), pp. 51–59.
29. Hansen, E.: Bounding the Solution of Interval Linear Equations, *SIAM Journal on Numerical Analysis* **29** (5) (1992), pp. 1493–1503.
30. Hansen, E.: Interval Arithmetic in Matrix Computations, *SIAM Journal on Numerical Analysis* **2** (1965), pp. 308–320.
31. Hansen, E. R.: Interval Forms of Newton's Method, *BIT* **20** (1978), pp. 153–163.
32. Hansen, E.: Preconditioning Linearized Equations, *Computing* **58** (2) (1997), pp. 187–196.
33. Hansen, E. R. and Greenberg, R. I.: An Interval Newton Method, *Applied Mathematics and Computation* **12** (2–3) (1983), pp. 89–98.
34. Hansen, E. and Sengupta, S.: Bounding Solutions of Systems of Equations Using Interval Analysis, *BIT* **21** (1981), pp. 203–211.
35. Hansen, E. and Smith, R.: Interval Arithmetic in Matrix Computations, Part II, *SIAM Journal on Numerical Analysis* **4** (1) (1967), pp. 1–9.
36. Hansen, E. and Walster, G. W.: *Global Optimization Using Interval Analysis*, 2nd Edition, Marcel Dekker, New York, 2004.
37. Hansen, E. R. and Walster, G. W.: Sharp Bounds on Interval Polynomial Roots, *Reliable Computing* **8** (2) (2002), pp. 115–122.
38. Hillyard, R. C. and Braid, I. C., Analysis of Dimensions and Tolerances in Computer-Aided Mechanical Design, *Computer-Aided Design* **10** (3) (1978), pp. 161–166.
39. Hoffmann, C. M., Lomonosov, A., and Sitharam, M.: Decomposition Plans for Geometric Constraint Systems, Part I: Performance Measures for CAD, *Journal of Symbolic Computation* **31** (4) (2001), pp. 367–408.
40. Hoffmann, C. M., Lomonosov, A., and Sitharam, M.: Decomposition Plans for Geometric Constraint Systems, Part II: New Algorithms, *Journal of Symbolic Computation* **31** (4) (2001), pp. 409–427.
41. Hong, H. and Stahl, V.: Bernstein Form Is Inclusion Monotone, *Computing* **55** (1995), pp. 43–53.

42. Hsu, C. Y. and Bruderlin, B.: Constraint Objects—Integrating Constraint Definition and Graphical Interaction, in: *ACM Proceedings of the Second Symposium on Solid Modeling and Applications, 1993, Montreal, Quebec, Canada*, pp. 467–468.
43. Hu, C. Y., Maekawa, T., Patrikalakis, N. M., and Ye, X.: Robust Interval Algorithm for Surface Intersections, *Computer-Aided Design* **29** (9) (1997), pp. 617–627.
44. Hu, C. Y., Patrikalakis, N. M., and Ye, X.: Robust Interval Solid Modeling, Part II: Boundary Evaluation, *Computer-Aided Design* **28** (10) (1996), pp. 819–830.
45. Hungerbuhler, A. R. and Garloff, B. J.: Bounds for the Range of a Bivariate Polynomial over a Triangle, *Reliable Computing* **4** (1) (1998), pp. 3–13.
46. Jaulin, L., Kieffer, M., Didrit, O., and Walter, E.: *Applied Interval Analysis*, Springer, London, 2001.
47. Kalra, D. and Barr, A. H.: Guaranteed Ray Intersections with Implicit Surfaces, *Computer Graphics* **23** (3) (1989), pp. 297–304.
48. Kaucher, E.: Interval Analysis in the Extended Interval Space IR, *Computing Supplementum 2*, Springer, Heidelberg, 1980, pp. 33–49.
49. Kearfott, R. B.: On Existence and Uniqueness Verification for Non-Smooth Functions, *Reliable Computing* **8** (4) (2002), pp. 267–282.
50. Kearfott, R. B.: Preconditioners for the Interval Gauss-Seidel Method, *SIAM Journal on Numerical Analysis* **27** (3) (1990), pp. 804–822.
51. Kearfott, R. B.: *Rigorous Global Search: Continuous Problems*, Kluwer Academic Publishers, Dordrecht, 1996.
52. Kearfott, R. B. and Dian, J.: Existence Verification for Higher Degree Singular Zeros of Nonlinear Systems, *SIAM Journal on Numerical Analysis* **41** (6) (2003), pp. 2350–2373.
53. Kearfott, R. B., Dian, J., and Neumaier, A.: Existence Verification for Singular Zeros of Complex Nonlinear Systems, *SIAM Journal on Numerical Analysis* **38** (2) (2000), pp. 360–379.
54. Kearfott, R. B. and Walster, G. W.: Symbolic Preconditioning with Taylor Models: Some Examples, *Reliable Computing* **8** (6) (2002), pp. 453–468.
55. Kolev, L. V.: A New Method for Global Solution of Systems of Non-Linear Equations, *Reliable Computing* **4** (2) (1998), pp. 125–146.
56. Kolev, L. V.: An Improved Method for Global Solution of Non-Linear Systems, *Reliable Computing* **5** (2) (1999), pp. 103–111.
57. Kolev, L. V.: Automatic Computation of a Linear Interval Enclosure, *Reliable Computing* **7** (1) (2001), pp. 17–28.
58. Kolev, L.: Use of Interval Slopes for the Irrational Part of Factorable Functions, *Reliable Computing* **3** (1) (1997), pp. 83–93.
59. Kolev, L. V. and Nenov, I.: Cheap and Tight Bounds on the Solution Set of Perturbed Systems of Nonlinear Equations, *Reliable Computing* **7** (5) (2001), pp. 399–408.
60. Kondo, K.: Algebraic Method for Manipulation of Dimensional Relationships in Geometric Models, *Computer-Aided Design* **24** (3) (1992), pp. 141–147.
61. Kondo, K.: PIGMOD: Parametric and Interactive Geometric Modeller for Mechanical Design, *Computer-Aided Design* **22** (10) (1990), pp. 633–644.
62. Kramer, G. A.: A Geometric Constraint Engine, in: Freuder, E. C. and Mackworth, A. K. (eds), *Artificial Intelligence: Constraint-Based Reasoning* **58**, Elsevier, 1992, pp. 327–360.
63. Krawczyk, R.: Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken, *Computing* **4** (1969), pp. 187–201.
64. Krawczyk, R. and Neumaier, A.: An Improved Interval Newton Operator, *Journal of Mathematical Analysis and Applications* **118** (1) (1986), pp. 194–207.
65. Krawczyk, R. and Neumaier, A.: Interval Slopes for Rational Functions and Associated Centered Forms, *SIAM Journal on Numerical Analysis* **22** (3) (1985), pp. 604–615.
66. Lamure, H. and Michelucci, D.: Solving Geometric Constraints by Homotopy, in: *Proceedings of the Third ACM Symposium on Solid Modeling and Applications, May 17–19, 1995, Salt Lake City, Utah*, pp. 263–269.
67. Latham, R. S. and Middleditch, A. E.: Connectivity Analysis: A Tool for Processing Geometric Constraints, *Computer-Aided Design* **28** (11) (1996), pp. 917–928.

68. Lee, J. Y. and Kim, K.: A 2-D Geometric Constraint Solver Using DOF-Based Graph Reduction, *Computer-Aided Design* **30** (11) (1998), pp. 883–896.
69. Lee, J. Y. and Kim, K.: Geometric Reasoning for Knowledge-Based Parametric Design Using Graph Representation, *Computer-Aided Design* **28** (10) (1996), pp. 831–841.
70. Light, R. and Gossard, D.: Modification of Geometric Models Through Variational Geometry, *Computer-Aided Design* **14** (4) (1982), pp. 209–214.
71. Lin, H., Liu, L., and Wang, G.: Boundary Evaluation for Interval Bezier Curve, *Computer-Aided Design* **34** (9) (2002), pp. 637–646.
72. Maekawa, T. and Patrikalakis, N. M.: Computation of Singularities and Intersections of Offsets of Planar Curves, *Computer Aided Geometric Design* **10** (5) (1993), pp. 407–429.
73. Maekawa, T. and Patrikalakis, N. M.: Interrogation of Differential Geometry Properties for Design and Manufacture, *The Visual Computer* **10** (4) (1994), pp. 216–237.
74. Makino, K. and Berz, M.: Efficient Control of the Dependency Problem Based on Taylor Model Methods, *Reliable Computing* **5** (1) (1999), pp. 3–12.
75. Makino, K. and Berz, M.: Remainder Differential Algebras and Their Applications, in: Berz, M., Bischof, C., Corliss, G., and Griewank, A. (eds.), *Computational Differentiation: Techniques, Applications, and Tools*, SIAM, Philadelphia pp. 63–74.
76. Mayer, G.: Epsilon-Inflation in Verification Algorithms, *Journal of Computational and Applied Mathematics* **60** (1–2) (1995), pp. 147–169.
77. Modares, M., Mullen, R., Muhanna, R. L., and Zhang, H.: Buckling Analysis of Structures with Uncertain Properties and Loads Using an Interval Finite Element Method, in: Muhanna, R. L. and Mullen, R. L. (eds), *Proceedings of the 2004 NSF Workshop on Reliable Engineering Computing, September 15–17, 2004, Savannah, GA, USA*, pp. 317–327.
78. Moore, M. and Wilhelms, J.: Collision Detection and Response for Computer Animation, *Computer Graphics* **22** (4) (1988), pp. 289–298.
79. Moore, R. E.: *Methods and Applications of Interval Analysis*, SIAM, Philadelphia, 1979.
80. Moore, R. E. (ed.): *Reliability in Computing: The Role of Interval Methods in Scientific Computing*, Academic Press, Boston, 1988.
81. Moore, R. E.: Sparse Systems in Fixed Point Form, *Reliable Computing* **8** (4) (2002), pp. 249–265.
82. Moore, R. E. and Qi, L.: A Successive Interval Test for Nonlinear Systems, *SIAM Journal on Numerical Analysis* **19** (4) (1982), pp. 845–850.
83. Mudur, S. P. and Koparkar, P. A.: Interval Methods for Processing Geometric Objects, *IEEE Computer Graphics and Applications* **4** (2) (1984), pp. 7–17.
84. Muhanna, R. L. and Mullen, R. L.: Formulation of Fuzzy Finite-Element Methods for Solid Mechanics Problems, *Computer-Aided Civil and Infrastructure Engineering* **14** (1999), pp. 107–117.
85. Muhanna, R. L. and Mullen, R. L.: Uncertainty in Mechanics Problems—Interval-Based Approach, *ASCE Journal of Engineering Mechanics* **127** (6) (2001), pp. 557–566.
86. Muhanna, R. L., Mullen, R. L., and Zhang, H.: Interval Finite Element as a Basis for Generalized Models of Uncertainty in Engineering Mechanics, in: Muhanna, R. L. and Mullen, R. L. (eds), *Proceedings of the 2004 NSF Workshop on Reliable Engineering Computing, September 15–17, 2004, Savannah, GA, USA*, pp. 353–370.
87. Mullineux, G.: Constraint Resolution Using Optimisation Techniques, *Computers & Graphics* **25** (3) (2001), pp. 483–492.
88. Neumaier, A.: A Simple Derivation of the Hansen-Blik-Rohn-Ning-Kearfott Enclosure for Linear Interval Equations, *Reliable Computing* **5** (2) (1999), pp. 131–136.
89. Neumaier, A.: *Interval Methods for Systems of Equations*, Cambridge University Press, 1990.
90. Neumaier, A.: On Shary’s Algebraic Approach for Linear Interval Equations, *SIAM Journal on Matrix Analysis and Applications* **21** (4) (2000), pp. 1156–1162.
91. Neumaier, A.: Taylor Forms—Use and Limits, *Reliable Computing* **9** (1) (2003), pp. 43–79.
92. Ning, S. and Kearfott, R. B.: A Comparison of Some Methods for Solving Linear Interval Equations, *SIAM Journal on Numerical Analysis* **34** (4) (1997), pp. 1289–1305.

93. Owen, J. C.: Algebraic Solution for Geometry from Dimensional Constraints, in: *ACM Proceedings of the First Symposium on Solid Modeling Foundations and CAD/CAM Applications, 1991, Austin, Texas*, pp. 397–407.
94. Perez, A. and Serrano, D.: Constraint Based Analysis Tools for Design, in: *ACM Proceedings on the 2nd Symposium on Solid Modeling and Applications, 1993, Montreal, Quebec, Canada*, pp. 281–291.
95. Rao, S. S. and Berke, L.: Analysis of Uncertain Structural Systems Using Interval Analysis, *AIAA Journal* **35** (4) (1997), pp. 727–735.
96. Rao, S. S. and Cao, L.: Optimum Design of Mechanical Systems Involving Interval Parameters, *ASME Journal of Mechanical Design* **124** (2002), pp. 465–472.
97. Ratschek, H. and Rokne, J.: *Computer Methods for the Range of Functions*, Ellis Horwood, Chichester, 1984, Ch. 6.
98. Ratschek, H. and Rokne, J.: *New Computer Methods for Global Optimization*, Ellis Horwood, New York, 1988.
99. Rohn, J.: Cheap and Tight Bounds: the Recent Result by E. Hansen Can Be Made More Efficient, *Interval Computations* **4** (1993), pp. 13–21.
100. Rokne, J. G.: A Note on the Bernstein Algorithm for Bounds for Interval Polynomials, *Computing* **21** (1979), pp. 159–170.
101. Roller, D.: An Approach to Computer-Aided Parametric Design, *Computer-Aided Design* **23** (5) (1991), pp. 385–391.
102. Rump, S. M.: A Note on Epsilon-Inflation, *Reliable Computing* **4** (4) (1998), pp. 371–375.
103. Rump, S. M.: Inclusion of Zeros of Nowhere Differentiable n -Dimensional Functions, *Reliable Computing* **3** (1) (1997), pp. 5–16.
104. Sederberg, T. W. and Farouki, R. T.: Approximation by Interval Bezier Curves, *IEEE Computer Graphics and Applications* **12** (5) (1992), pp. 87–95.
105. Shary, S. P.: A New Technique in Systems Analysis under Interval Uncertainty and Ambiguity, *Reliable Computing* **8** (5) (2002), pp. 321–418.
106. Shary, S. P.: Algebraic Approach in the “Outer Problem” for Interval Linear Equations, *Reliable Computing* **3** (2) (1997), pp. 103–135.
107. Snyder, J.: *Generative Modeling for Computer Graphics and CAD: Symbolic Shape Design Using Interval Analysis*, Academic Press, Cambridge, 1992.
108. Snyder, J. M., Woodbury, A. R., Fleischer, K., Currin, B., and Barr, A. H.: Interval Methods for Multi-Point Collisions Between Time-Dependant Curved Surfaces, in: *ACM Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, New York, 1993, pp. 321–334.
109. Solano, L. and Brunet, P.: Constructive Constraint-Based Model for Parametric CAD Systems, *Computer-Aided Design* **26** (8) (1994), pp. 614–621.
110. Stahl, V., *Interval Methods for Bounding the Range of Polynomials and Solving Systems of Nonlinear Equations*, unpublished Ph.D. thesis, University of Linz, 1995.
111. Sunde, G.: Specification of Shape by Dimensions and Other Geometric Constraints, in: Wozny, M. J., McLaughlin, H. W., and Encarnacao, J. L. (eds), *Geometric Modeling for CAD Applications, IFIP WG 5.2 Working Conference on Geometric Modeling for CAD Applications, May 12–16, 1986, Rensselaerville, New York*, North-Holland, Amsterdam, 1988, pp. 199–213.
112. Toth, D. L.: On Ray Tracing Parametric Surfaces, *Computer Graphics* **19** (3) (1985), pp. 171–179.
113. Tuohy, S. T., Maekawa, T., Shen, G., and Patrikalakis, N. M.: Approximation of Measured Data with Interval B-Splines, *Computer-Aided Design* **29** (11) (1997), pp. 791–799.
114. Von Herzen, B., Barr, A. H., and Zatz, H. R.: Geometric Collisions for Time-Dependent Parametric Surfaces, *Computer Graphics* **24** (4) (1990), pp. 39–48.
115. Wallner, J., Krasauskas, R., and Pottmann, H.: Error Propagation in Geometric Constructions, *Computer-Aided Design* **32** (11) (2000), pp. 631–641.
116. Wolfe, M. A.: A Modification of Krawczyk’s Algorithm, *SIAM Journal on Numerical Analysis* **17** (3) (1980), pp. 376–379.
117. Wolfe, M. A.: On Bounding Solutions of Underdetermined Systems, *Reliable Computing* **7** (3) (2001), pp. 195–207.

118. Yamaguchi, Y. and Kimura, F.: A Constraint Modeling System for Variational Geometry, in: Wozny, M. J., Turner, J. U., and Preiss, K. (eds), *Geometric Modeling for Product Engineering, IFIP WG 5.2/NSF Working Conference on Geometric Modeling, September 18–22, 1988, Rensselaerville, New York*, North-Holland, Amsterdam, 1990, pp. 221–233.
119. Yamamura, K.: An Algorithm for Representing Functions of Many Variables by Superpositions of Functions of One Variable and Addition, *IEEE Transactions on Circuits and Systems—I: Fundamental Theory and Application* **43** (4) (1996), pp. 338–340.
120. Zhang, D., Li, W., and Shen, Z.: Solving Underdetermined Systems with Interval Methods, *Reliable Computing* **5** (1) (1999), pp. 23–33.
121. Zuhe, S. and Wolfe, M. A.: On Interval Enclosures Using Slope Arithmetic, *Applied Mathematics and Computation* **39** (1) (1990), pp. 89–105.