Componentwise Determination of the Interval Hull Solution for Linear Interval Parameter Systems*

L. V. Kolev
Dept. of Theoretical Electrotechnics, Faculty of Automatics, Technical University of Sofia, 1000 Sofia, Bulgaria
lkolev@tu-sofia.bg

Abstract

In this paper, the problem of determining the interval hull (IH) solution $\boldsymbol{x}^*$ to a linear interval parameter system $A(p)x = b(p)$, $p \in \boldsymbol{p}$ is revisited. A new iterative method for computing $\boldsymbol{x}^*$ is suggested, which is based on individually finding each interval component $\boldsymbol{x}_k^*$ of $\boldsymbol{x}^*$. Each component $\boldsymbol{x}_k^* = [\underline{x}_k^*, \overline{x}_k^*]$ is in turn found by separately determining the lower end-point $\underline{x}_k^*$ and upper end-point $\overline{x}_k^*$ of $\boldsymbol{x}_k^*$, respectively. The lower end-point $\underline{x}_k^*$ is located by an iterative method which, at each iteration, makes use of a respective outer solution $\boldsymbol{x}$ and an upper bound x_k^u on $\underline{x}_k^*$. The upper end-point $\overline{x}_k^*$ is located in a similar manner using relevant outer solutions $\boldsymbol{x}^*$ and lower bounds $\boldsymbol{x}_k^l$ on $\overline{x}_k^*$. In both cases, appropriate modified monotonicity conditions are checked and used. Such an approach results in better performance compared to similar methods employing standard monotonicity conditions. The method is capable of determining the solution $\boldsymbol{x}^*$ if the modified monotonicity conditions are satisfied for all components of p; otherwise, it only provides a two-sided enclosure $[\underline{x}_k, x_k^u]$ ($[x_k^l, \overline{x}_k]$) of $\underline{x}_k^*(\overline{x}_k^*)$. The method is extended to a more general setting where the problem is to determine the IH $\boldsymbol{y}^*$ of an output variable vector y which depends on x and p, $p \in \boldsymbol{p}$. A numerical example illustrating the new method is also given.

Keywords: linear interval parameter systems, interval hull solution, modified monotonicity conditions
AMS subject classifications: 65L15,65G40

1 Introduction

Let $p = (p_1, ..., p_m)$ be a real m-dimensional vector belonging to a given interval vector $\boldsymbol{p} = (\boldsymbol{p}_1, ..., \boldsymbol{p}_m)$. Also, let $A(p)$ and $b(p)$ be, respectively, a real rectangular $(n_1 \times n_2)$

*Submitted: January 22, 2013; Final revision: February 26, 2014; Accepted: March 4, 2014

matrix and an n_1-dimensional vector whose elements depend on p. As is well known, a (real) linear interval parameter (LIP) system is defined as the family of linear algebraic systems

$$A(p)x = b(p) \tag{1.1a}$$

$$a_{ij}(p) = a_{ij}(p_1, ..., p_m), \quad b_i(p) = b_i(p_1, ..., p_m) \tag{1.1b}$$

where $a_{ij}(p) = a_{ij}(p_1, ..., p_m)$ and $b_i(p) = b_i(p+1, ..., p_m)$ are given functions from R^m to R and

$$p_\mu \in \boldsymbol{p}_\mu, \quad \mu = 1, ..., m. \tag{1.1c}$$

Systems of this type are encountered in many practical applications (e.g., [3, 5, 11, 13, 16, 18].

The united solution set of (1.1) is the collection of all point solutions of (1.1a), (1.1b) over $\boldsymbol{p}$, i.e. the set $\sum(a(p), b(p), \boldsymbol{p}) = \{x : A(p)x = b(p), p \in \boldsymbol{p}\}$.

This set has a rather complex form [1] even in the simplest case of affine (linear) functions

$$a_{ij}(p) = \alpha_{ij} + \sum_{\mu=1}^{m} a_{ij\,\mu} p_\mu, \tag{1.2a}$$

$$b_i(p) = \beta_i + \sum_{\mu=1}^{m} \beta_{i\mu} p_\mu, \tag{1.2b}$$

(systems characterised by the linear parametric dependence (1.2) will be referred to as LPD systems) and $n_1 = n_2$. Therefore (under the assumption that $\sum(A(p), b(p), \boldsymbol{p})$ is a bounded set), the following "interval solutions" to (1.1) will be considered in this paper:

(i) interval hull (IH) solution $\boldsymbol{x}^*$: the smallest interval vector containing $\sum(A(p), b(p), \boldsymbol{p})$;

(ii) outer interval (OI) solution $\boldsymbol{x}$: any interval vector enclosing $\boldsymbol{x}^*$, i.e. $\boldsymbol{x}^* \subseteq \boldsymbol{x}$;

(iii) inner estimation of the hull (IEH) solution ζ: an interval vector such that $\boldsymbol{\zeta} \subseteq \boldsymbol{x}^*$.

Most of the known results are obtained for the case of determining OI solutions of square LIP systems ($n_1 = n_2$). Various iterative [2, 8, 10, 12, 14] ([2] being a special case of [14]) and direct [4, 24] methods for determining OI solutions associated with LPD systems have been suggested. Two different methods for treating nonlinear parametric dependency problems have been proposed in [7] and [14], respectively. The general case of $n_1 \neq n_2$ has also been considered for the case of interval and parametric matrices (e.g., [15]). Methods for obtaining IEH solutions have been proposed in [5, 10, 25]. The latter solutions are also computed as a byproduct by the method of [14].

Determining the IH solution $\boldsymbol{x}^*$ of an LIP system is an NP-hard problem even in the case of LPD systems. Thus, the solution $\boldsymbol{x}^*$ can be obtained with reasonable amount of computations only if certain restrictive requirements are additionally imposed. Such an approach for determining $\boldsymbol{x}^*$ has been adopted by many authors (e.g., [5, 7, 17, 18, 21, 26]) where certain monotonicity conditions are to be fulfilled. In the earlier publications on the topic, the monotonicity conditions are defined over the whole initial domain $\boldsymbol{p}$ of the interval parameters. The idea to check monotonicity conditions valid for certain nested subdomains of $\boldsymbol{p}$ has been suggested, seemingly for the first time, in [5] (see also [6] and [7]). Independently of [5], the same idea was proposed slightly later in [17] but was implemented in a better way (accounting to a fuller extent for the interdependencies between all uncertain parameters involved, verified

evaluation of the partial derivatives used). The methods in [5, 7, 17] are iterative which permits to reduce the monotonicity requirements successively at each iteration. A global optimization method has been used in [27, 28] for approximating or computing $\boldsymbol{x}^*$, respectively.

In the present paper, a new iterative method for determining $\boldsymbol{x}^*$ is suggested, which is an improvement over the methods of [5] and [17]. The idea behind the new method is to determine $\boldsymbol{x}^*$ in a componentwise manner, computing separately the lower end-point $\underline{x}_k^*$ and upper end-point $\overline{x}_k^*$ of each component $\boldsymbol{x}_k^*$. In computing $\underline{x}_k^*$, use is made of both the outer interval approximation and the kth component ζ_k of the approximation ζ with respect to $\underline{x}_k^*$ at each iteration of the iterative process. Thus, appropriate modified monotonicity conditions are introduced which are less restrictive compared to the standard monotonicity conditions previously used. Therefore, the fulfillment of the modified monotonicity conditions speeds up the location of $\underline{x}_k^*$. The same approach is used for computing $\overline{x}_k^*$. The method is capable of determining the solution $\boldsymbol{x}^*$ if the modified monotonicity conditions are satisfied for all components of p; otherwise, it only provides a two-sided enclosure of $\underline{x}_k^*$ $(\overline{x}_k^*)$. It is also shown that the new method can be extended to the more general problem of determining the IH $\boldsymbol{y}^*$ of an outcome variable vector $\boldsymbol{y}$ depending on both x and p, $p \in \boldsymbol{p}$. In a less general form (treating a specific problem arising in mechanical engineering), a method for computing $\boldsymbol{y}^*$, based on the standard monotonicity approach, has already been considered in [13]; in the framework of an interval finite element formulation, methods for bounding $\boldsymbol{y}^*$ can be found in [19] and [20].

The paper is organized as follows. The formulation of the problems considered and the basic approach to solving them are given in Section 2. The main results obtained are reported in the next section. In Section 4, several algorithms implementing the results from the previous section are presented. The new method is illustrated by way of a numerical example in Section 5. The paper ends up with several concluding remarks.

2 Problem Formulation and Basic Approach

We shall distinguish between two forms of the IH solution problem: standard form and generalized form. The standard IH solution (SIHS) problem is formulated as follows: given the pair $\{A(p), b(p)\}$ and the interval vector $\boldsymbol{p}$, find the IH solution $\boldsymbol{x}^*$ to (1.1).

The LIP system (1.1a) to (1.1c) defines implicitly a non-linear mapping $\mathbb{N} : \boldsymbol{p} \in R^m \to R^{n_1}$. Thus, the united solution set $\sum(A(p), b(p), \mathbf{p})$ can be viewed as the image $\mathbb{N}(\boldsymbol{p})$ of $\boldsymbol{p}$ under the mapping $\mathbb{N}$. Let $\square(\Sigma)$ denote the interval hull of $\sum(A(p), b(p), \boldsymbol{p})$. Thus, the IH solution x^* can be defined as follows

$$x^* = \square(\mathbb{N}(\boldsymbol{p})). \tag{2.1}$$

To introduce the generalized IH solution (GIHS) problem, we need to define an additional mapping

$$y = f(x, p) \tag{2.2}$$

where $f : \mathbb{N}(\boldsymbol{p}) \times \boldsymbol{p} \in R^{n_2+m} \to R^{n'}$, $1 \leq n' \leq n_2$. It specifies the transformation of the "state" variable vector x and the "input" parameter vector p into an "output" variable vector y. Let $\sum(A(p), b(p), f, \boldsymbol{p})$ denote the united solution set of (1.1) and (2.2). Thus

$$\boldsymbol{y}^* = \square(\Sigma(A(p), b(p), f, \boldsymbol{p}). \tag{2.3}$$

The generalized IH solution (GIHS) problem is formulated as follows: given the triple $\{A(p), b(p), f\}$ and the interval vector $\boldsymbol{p}$, find the corresponding IH solution $\boldsymbol{y}^*$.

On account of (1.1) and (2.2), the GIHS problem can be viewed as that of determining the interval hull of the image of p under the composition $f\mathbb{N}$.

A specific GIHS problem related to a given pair $\{A(p), b(p)\}$ is actually defined by specifying the function $f(x, p)$ in (2.2). In its most general form, the GIHS setting encompasses a large class of various problems depending on the type of f used. If $f(x, p)$ is a nonlinear function, determining each end-point $\underline{y}_k^*$ or $\overline{y}_k^*$ of the kth component $\boldsymbol{y}_k^* = [\underline{y}_k^*, \overline{y}_k^*]$ of $\boldsymbol{y}^*$ is a perturbed global optimization problem with linear equality constraints. If $f(x, p)$ is linear in x, then the corresponding problem is a parametric linear programming problem. In many practical applications (e.g., [3, problems 3.9, 3.10], f is a scalar nonlinear function $f(x)$ $(n' = 1)$ independent of p. An illustrative example is the case of determining the magnitude v of a single complex variable V_k involved in a $(n \times n)$ complex-valued LIP system [3]

$$GV = J. \tag{2.4}$$

The latter system can be rewritten equivalently as a $(2n \times 2n)$ real-valued LIP system by introducing a $2n$-dimensional real state vector x. In x, the first n components correspond to the respective real parts of V_j while the next n components correspond to the respective imaginary parts of V_j. Thus, for this example

$$y = x_k^2 + x_{k+n}^2. \tag{2.5}$$

If f is independent of p and

$$f = E \tag{2.6}$$

(E is the identity matrix), then the GIHS problem reduces to the SIHS problem. If we are interested in finding the range of a single component x_k of x, then (2.2) becomes

$$y_k = e_k^T x \tag{2.7}$$

(e_k^T is the transposed kth column of E).

In this section, the basis of a method for determining the IH solution $\boldsymbol{x}^*$ in the simpler case of the SIHS problem defined by (1.1), (2.6) will be presented. It is based on a componentwise approach: compute $\boldsymbol{x}_k^*$ n_2 times for $k = 1, ..., n_2$. Also, for a fixed k:

(i) first, the lower end-point $\underline{x}_k^*$ of the kth component $\boldsymbol{x}_k^* = [\underline{x}_k^*, \overline{x}_k^*]$ of $\boldsymbol{x}^*$ is located;

(ii) next, the upper end-point $\overline{x}_k^*$ of the kth component $\boldsymbol{x}_k^*$ of $\boldsymbol{x}^*$ is found.

Determination of $\underline{x}_k^*$

On account of (2.7), the value of $\underline{x}_k^*$ will be determined as the solution of the following global optimization problem

$$\underline{x}_k^* = \min e_k^T x \tag{2.8a}$$

subject to the constraint

$$A(p)x = b(p), \quad p \in \boldsymbol{p}. \tag{2.8b}$$

An iterative method for solving (2.8) will be suggested in the next section. It is based on the use of an interval enclosure $\boldsymbol{d}_l^{(k)}$ for the derivative $\partial x_k / \partial p_l$ of x_k with respect

to $\boldsymbol{p}_l$, $l = 1, ..., m$ for a given $\boldsymbol{p}$ (related to a current iteration). To obtain $\boldsymbol{d}_k^{(l)}$, we first differentiate (2.8b) in p_l to get

$$\sum_{j=1}^{n} a_{ij}(p)\frac{\partial x_j}{\partial p_l} = \frac{\partial b_i(p)}{\partial p_l} - \sum_{j=1}^{n} a_{ij}(p)\frac{\partial a_{ij}(p)}{\partial p_l}x_j, \quad i = 1, ...n_2, \tag{2.9a}$$

$$p \in \boldsymbol{p}. \tag{2.9b}$$

System (2.9) is rewritten as

$$A(p)d_l = \gamma_l(p) - \eta_l(p)x(p), \quad p \in \boldsymbol{p} \tag{2.10}$$

where $\gamma_l(p)$ is a column vector and $\eta_l(p)$ is a matrix. Let $\boldsymbol{d}_l$ denote an outer solution to (2.10). Obviously, the enclosure $\boldsymbol{d}_l^{(k)}$ sought is the k-th component of $\boldsymbol{d}_l$. If

$$0 \notin \boldsymbol{d}_l^{(k)}, \tag{2.11}$$

we can reduce the interval $\boldsymbol{p}_l$ to a point p_l. Indeed, on account of (2.9) and (2.10)

$$\frac{\partial x_k}{\partial p_l}(p) \in \boldsymbol{d}_l^{(k)}, \quad p \in \boldsymbol{p}. \tag{2.12}$$

Hence, (2.11) guarantees that x_k is monotone in $\boldsymbol{p}$ with respect to p_l. Therefore,

$$p_l = \begin{cases} \underline{p}_l, & \text{if} \quad \boldsymbol{d}_l^{(k)} \geq 0 \\ \overline{p}_l, & \text{if} \quad \boldsymbol{d}_l^{(k)} \leq 0. \end{cases} \tag{2.13}$$

Such an approach has already been used in [5], [17] and [26] for square matrices.

The interval $\boldsymbol{d}_l^{(k)}$ could be computed by applying the approach of [5] and [17]. It consists of ignoring the dependence of x on p in (2.10) and treating x as an independent variable q beloning to $\boldsymbol{x}$. Thus, (2.10) becomes

$$A(p)d_l = \gamma(p) - \eta(p)q, \quad p \in \boldsymbol{p}, \quad q \in \boldsymbol{x} \tag{2.14}$$

where $\boldsymbol{x}$ is an OI solution to (2.8b). It should be stressed that this approach is based uniquely on $\boldsymbol{p}$ and the outer solution $\boldsymbol{x}$ to (2.8b).

A better approach to assessing $\boldsymbol{d}_l$ applicable both in the context of the SIHS and GIHS problem will be suggested in the next section. It resorts to also employing the k-th component ζ_k of the inner estimation ζ related to (2.8b).

Remark 2.1 *Seemingly the best approach (not considered so far) to exploiting the dependence of both d_l and x on p in* (2.10) *is to find an outer solution to the following LIP system of size $(2n_1 \times 2n_2)$:*

$$A(p)x = b(p), \tag{2.15a}$$

$$A(p)d_l = \gamma_l(p) - \eta_l(p)x, \tag{2.15b}$$

$$p \in \boldsymbol{p}. \tag{2.15c}$$

This opportunity will not be considered in this paper.

In the sequel, we shall compare various interval vectors with respect to their widths. The width $w(\boldsymbol{p})$ of an interval vector $\boldsymbol{p}$ is a real vector whose components $w_i(\boldsymbol{p})$ are defined as the width of the ith component $\boldsymbol{p}_i$ of $\boldsymbol{p}$, i.e. $w_i(\boldsymbol{p}) = \overline{p}_i - \underline{p}_i$. An interval vector $\boldsymbol{p}$ will be called narrower than another interval vector $\boldsymbol{p}'$ if $w_k(\boldsymbol{p}) < w_k(\boldsymbol{p}')$ for at least one index k. The vector $\boldsymbol{p}$ will be referred to as strictly narrower than the vector $\boldsymbol{p}'$ if all components $w_i(\boldsymbol{p})$ are narrower than the respective components $w_i(\boldsymbol{p}')$.

3 Main Results

3.1 The SIHS Problem: Lower End-point of the Range

The new approach is based on the simultaneous use of both the outer solution $\boldsymbol{x}$ to (2.8b) and an upper bound x_k^u on the lower end-point x_k^*. The bound x_k^u is given by the lower end-point $\underline{\zeta}_k$ of the k-th component $\boldsymbol{\zeta}_k = [\underline{\zeta}_k, \overline{\zeta}_k]$ of the IEH solution $\boldsymbol{\zeta}$ to (2.8b) or by a specialized method (e.g., [5]). Since x_k'' is an upper bound on $\underline{x}_k^*$ and $\underline{x}_k$ is a lower bound on $\underline{x}_k^*$

$$\underline{x}_k \leq \underline{x}_k^* \leq x_k^u. \tag{3.1}$$

Thus,

$$x_k^* \in \tilde{\boldsymbol{x}}_k \tag{3.2a}$$

where

$$\tilde{\boldsymbol{x}}_k = [\underline{x}_k, x_k^u]. \tag{3.2b}$$

Now we introduce a modified outer solution vector $\tilde{\boldsymbol{x}}$ with components

$$\tilde{\boldsymbol{x}}_i = \begin{cases} \boldsymbol{x}_i, & \text{if} \quad i \neq k \\ \tilde{\boldsymbol{x}}_i, & \text{if} \quad i = k. \end{cases} \tag{3.2c}$$

The new approach consists of replacing (2.14) with the following system:

$$A(p)d_l = \gamma_l(p) - \eta_l(p)q, \quad p \in \boldsymbol{p}, \tag{3.3a}$$

$$q \in \widetilde{\boldsymbol{x}}. \tag{3.3b}$$

Since

$$\widetilde{\boldsymbol{x}}_k \subset \boldsymbol{x}_k, \tag{3.4a}$$

$$\widetilde{\boldsymbol{x}} \subseteq \boldsymbol{x}. \tag{3.4b}$$

Thus, the use of $\tilde{\boldsymbol{x}}$ instead of $\boldsymbol{x}$ in (3.3b) imposes a constraint on (3.3a). Exploiting this restriction via some constraint satisfaction technique (e.g., the second stage of the forward and backward sweep method) will lead to a narrower $\boldsymbol{x}'$ and, hence, to a narrower parameter interval vector $\boldsymbol{p}'$. Accordingly, we have to consider the following modified LIP system

$$A(p)d_l = \gamma_l(p) - \eta_l(p)q, \tag{3.5a}$$

$$p \in \boldsymbol{p}', \quad q \in \boldsymbol{x}', \tag{3.5b}$$

Now we can formulate the following result.

Lemma 3.1 *If*

$$\boldsymbol{p}' \subset \boldsymbol{p}, \quad \boldsymbol{x}' \subset \boldsymbol{x}, \tag{3.6a}$$

then

$$\widetilde{\boldsymbol{d}}_l(\boldsymbol{p}', \boldsymbol{x}') \subset \boldsymbol{d}_l(\boldsymbol{p}, \boldsymbol{x}). \tag{3.6b}$$

Otherwise, if

$$\boldsymbol{p}' \subseteq \boldsymbol{p}, \quad \boldsymbol{x}' \subseteq \boldsymbol{x}, \tag{3.7a}$$

then

$$\widetilde{\boldsymbol{d}}_l(\boldsymbol{p}', \boldsymbol{x}') \subseteq \boldsymbol{d}_l(\boldsymbol{p}, \boldsymbol{x}). \tag{3.7b}$$

Proof. The assertions of the lemma follow directly from the premises (3.6a), (3.7a) and the inclusion isotonicity property of the interval operations needed to compute $\widetilde{\boldsymbol{d}}_l(\boldsymbol{p}', \boldsymbol{x}')$ and $\boldsymbol{d}_l(\boldsymbol{p}, \boldsymbol{x})$, respectively.

It is seen from Lemma 2.1 that the outer interval solution $\widetilde{\boldsymbol{d}}_l(\boldsymbol{p}', \mathbf{x}')$ of (3.5) is never wider and may be narrower than the outer interval solution $\boldsymbol{d}_l(\boldsymbol{p}, \boldsymbol{x})$ of (2.14). Moreover, the former vector $\widetilde{\boldsymbol{d}}_l(\boldsymbol{p}', \boldsymbol{x}')$ is proved to be strictly narrower than the latter one $\boldsymbol{d}_l(\boldsymbol{p}, \boldsymbol{x})$ if (3.6a) holds.

As is well known, condition (2.11) guarantees that the function $x_k(p)$ is monotone within $\boldsymbol{p}$ with respect to the l-th component p_l of p. It has been used in [5] and [17] at each iteration ν of the respective iterative method for different interval vectors $\boldsymbol{p} = \boldsymbol{p}^{(\nu)}$ associated with the ν-th current iteration. To express the dependence of $\boldsymbol{d}_l^{(k)}$ on $\boldsymbol{p}^{(\nu)}$, (2.11) will be written in the form

$$0 \notin \boldsymbol{d}_l^{(k)}(\boldsymbol{p}^{(\nu)}, \boldsymbol{x}(\boldsymbol{p}^{\nu}). \tag{3.8}$$

In [17], the requirements (3.8) have been called a global monotonicity condition for $\nu = 1$ (first iteration) and local monotonicity condition for $\nu > 1$ (since $\boldsymbol{p}^{(1)}$ is the whole initial parameter domain while each subsequent $\boldsymbol{p}^{(2)}$, $\boldsymbol{p}^{(3)}$, etc. is a smaller and smaller subdomain of $\boldsymbol{p}^{(1)}$). In a similar way, the satisfaction of the requirement

$$0 \notin \widetilde{\boldsymbol{d}}_l^{(k)}(\boldsymbol{p}'^{(\nu)}, x(\boldsymbol{p}'^{(\nu)})) \tag{3.9}$$

is a sufficient condition for x_k to be monotone within $\mathbf{p}'$ with respect to p_l. On account of (3.6a) and (3.7a), the latter type of monotonocity condition (3.9) will be referred to as a modified monotonocity condition (compared to (3.8)).

As is well known, the united solution set $\Sigma(A(p), b(p), \boldsymbol{p})$ has a very complex form so, in the general case, it can touch its interval hull $\boldsymbol{x}^*$ at arbitrary points located on the $2n_2$ faces of $\boldsymbol{x}^*$. Therefore, each $\underline{x}_k^*$ (and, in a similar manner, each $\overline{x}_k^*$) is the image of a corresponding point $p^{(s)}$ lying on a certain face of $\boldsymbol{p}$. In a special case, however, $p^{(s)}$ may be a vertex $p^{(\nu)}$ of $\boldsymbol{p}$ (a vertex of $\boldsymbol{p}$ is a specific combination of end-points of $\boldsymbol{p}_j$, $j = 1, ..., m$). This property will be referred to as the *vertex property with respect to* $\underline{x}_k^*$ (or $\overline{x}_k^*$). If $\underline{x}_k^*$ does not possess the vertex property, the corresponding vector $p^{(s)}$ (providing $\underline{x}_k^*$) will have at least one component $p_i^{(s)}$ such that

$$p_i^{(s)} \in \mathrm{int}(\boldsymbol{p}_i) \tag{3.10}$$

(int stands for interior). One of the advantages of the present approach is that it is capable of establishing that the solution $\underline{x}_k^*$ sought does not possess the vertex property.

Lemma 3.2 *Let* $\boldsymbol{p}'^{(\nu)}$ *be the interval vector resulting from the application of a constraint technique at the* ν*-th iteration. If for at least one iteration* ν *and one index* i

$$\boldsymbol{p}'^{(\nu)} \in \mathrm{int}(\boldsymbol{p}_i), \tag{3.11}$$

then the solution $\underline{x}_k^*$ *is not reached on a vertex.*

Proof. It follows from the fact that (3.11) entails (3.10).

Remark 3.1 *In the present paper, it is assumed that condition* (3.11) *does not occur for all indices* i *and all iterations* ν*. Thus, the present approach will be developed only for that case where the vertex property is valid. The general case of non-vertex solutions will be considered in a separate publication.*

At this point, we need the following procedure designed to detect (if possible) a component p_l such that x_k is monotone along that component within $\boldsymbol{p}'$ (in the case of locating the lower end-point $\underline{x}_k^*$). To simplify the presentation, the arguments in $\boldsymbol{d}_l^{(k)}$ and $\widetilde{\boldsymbol{d}}_l^{(k)}$ will henceforth be omitted.

Procedure Pl. Given k, the functions $a_{ij}(p)$, $b_i(p)$ and the initial parameter vector $\boldsymbol{p}^{(0)}$, set $l = 1$, $\boldsymbol{p} = \boldsymbol{p}^{(0)}$ and carry out the following sequence of steps.

Step 1. Compute an outer solution $\boldsymbol{x}$ to (2.8b) using an appropriate method (accounting for the specific structure of the functions $a_{ij}(p)$ and $b_i(p)$).

Step 2. Compute an upper bound x_k^u, using an appropriate method (accounting for the specific structure of the functions $a_{ij}(p)$ and $b_i(p)$.

Step 3. Form by (3.2) the reduced-width interval $\widetilde{\boldsymbol{x}}$.

Step 3′. Apply a constraint satisfaction technique (e.g., the second stage of the forward and backward sweep method) to (3.5a) to get the modified interval vectors $\boldsymbol{x}'$ and $\boldsymbol{p}'$.

Step 4. Form the LIP system (3.5) and find an outer solution $\widetilde{\boldsymbol{d}}_l$ whose k-th component is $\widetilde{\boldsymbol{d}}_l^{(k)}$.

Step 5. Check the monotonicity condition (3.9). If (3.9) is fulfilled, go to Outcome O_2. Otherwise, go back to Step 4 with $l = l + 1$ if $l \leq m$; else proceed to the next line.

Outcome O_1: the procedure has not detected the modified monotonicity property for x_k along any p_l.

Outcome O_2: a variable p_l has been detected along which x_k is guaranteed to be locally monotone.

Remark 3.2 *To facilitate the readability of Procedure Pl it has been tacitly assumed that the method used in Step 1 and Step 4 is capable of determining the outer solution $\boldsymbol{x}$ or $\widetilde{\boldsymbol{d}}_l$, respectively. If this is not the case, the procedure will terminate with outcome O_1.*

We now present the basic algorithm for locating the lower end-point $\underline{x}_k^*$. It is iterative and consists of repeatedly calling Procedure Pl at most m times.

Algorithm Al. Let ν denote the number of the current iteration of the algorithm; let m^0 be the length of the initial vector $\boldsymbol{p}^{(0)}$. Set $\nu = 0$, $\boldsymbol{p} = \boldsymbol{p}^{(0)}$ and $m = m^0$.

Step 1. Let $\nu = \nu + 1$. Call procedure Pl. If the outcome is O_1, go to Termination T_1. If the outcome is O_2, a derivative interval $\widetilde{\boldsymbol{d}}_l^{(k)}$ that does not contain zero has been found. Thus, the corresponding parameter $\boldsymbol{p}_l$ can be reduced to a real number p_l^* using the formula

$$p_l^* = \begin{cases} \underline{p}_l, & \text{if} \quad \widetilde{\boldsymbol{d}}_l^{(k)} \geq 0 \\ \overline{p}_l, & \text{if} \quad \widetilde{\boldsymbol{d}}_l^{(k)} \leq 0. \end{cases} \tag{3.12}$$

Step 2. Form a new interval vector $\boldsymbol{p}'$ with components

$$\boldsymbol{p}_i' = \begin{cases} \boldsymbol{p}_i, & \text{if} \quad i \neq l \\ p_l^*, & \text{if} \quad i = l. \end{cases} \tag{3.13a}$$

Rewrite $\boldsymbol{p}'$ in the partitioned form

$$\boldsymbol{p}' = (p_l^*, \boldsymbol{p}) \tag{3.13b}$$

where $\boldsymbol{p}$ of length $m' = m - 1$ regroups the non-degenerate components of $\boldsymbol{p}'$. If $m' = 0$, go to Termination T2; otherwise let $m = m'$ and proceed to the next step.

Step 3. Substitute (3.13) into (1.1b), (1.1c) to get the modified functions $a'_{ij}(p)$ and $b'_i(p)$. Rename $a'_{ij}(p)$ and $b'_i(p)$ as $a_{ij}(p)$ and $b_i(p)$ and go back to Step 1.

Termination T_1: only a two-sided bound $[\underline{x}_k, \overline{x}_k^u]$ on the lower end-point $\underline{x}_k^*$ has been found.

Termination T_2: the algorithm has succeeded in determining a real vector p^* whose components are given by (3.12) such that its image provides the lower end-point $\underline{x}_k^*$.

Remark 3.3 *As is easily seen, $\nu \leq m$ if the algorithm terminates in T_1 or $\nu = m$ if it terminates in T_2.*

On account of Algorithm Al, we have the following result.

Theorem 3.1 *For given $\{A(p), b(p)\}$, $\boldsymbol{p}^{(0)}$ and k, assume that algorithm Al terminates in T_2 with p^* whose components are defined by* (3.12). *Then the following assertions are valid:*

(i) the global solution $\underline{x}_k^*$ of problem (2.8) *is reached at a vertex p^ν for m iterations of Al;*

(ii) the vertex p^ν sought is defined by p^* in (3.12) *and is unique;*

(iii) the numerical complexity of Algorithm Al is polynomial in n_1, n_2, and m.

Proof. (i). As Procedure Pl terminates in outcome O_2 for each iteration ν of Algorithm Al, the assertions are proved by induction. Thus, we first set $\nu = 1$. Let $\underline{x}^*$ denote the real vector associated with the parameter solution vector p^s to the minimization problem (2.8). On account of (3.1), (3.2) and the fact that $\boldsymbol{x}$ is an outer solution of (2.8b), it follows that $\underline{x}_k^* \in \widetilde{\boldsymbol{x}}_k$, $\underline{x}_i^* \in \widetilde{\boldsymbol{x}}_i$ so $\underline{x}^* \in \widetilde{\boldsymbol{x}}$; also $p^s \in \boldsymbol{p}$. Since any constraint satisfaction technique deletes only such parts of the initial $\boldsymbol{x}$ and $\boldsymbol{p}$ that do not contain $\underline{x}^*$ and p^s, we have $\underline{x}^* \in \boldsymbol{x}'$, $p^s \in \boldsymbol{p}'$. Hence, (3.9) is a sufficient condition for x_k to be monotone within $\boldsymbol{p}'$ with respect to p_l. Therefore, the l-th component p_l^s of the solution p^s is given by (3.12).

The same argument is valid for all $\nu > 1$. Thus, it has been shown that p_l^s is given by (3.12) for each l. Hence, the global solution of (2.8) is actually attained at a particular combination of end-points $\underline{p}_i$ and $\overline{p}_i$, i.e. at the vertex $p^{(\nu)}$, which proves the validity of assertion (i).

(ii). The validity of assertion (ii) is a corollary of assertion (i) since each p_l^s is determined in a unique manner by (3.12).

(iii). Since Algorithm Al terminates in exit T_2 , procedure Pl is called m times. For each ν, the outer solution $\boldsymbol{x}$, the upper solution x_k^u, the reduced-width vectors $\boldsymbol{x}'$, $\boldsymbol{p}'$ and the outer solution $\widetilde{\boldsymbol{d}}_l$ can be computed by a polynomial time algorithm. The respective number of computations is estimated by $P_1(n_1, n_2, m)$, $P_2(n_1, n_2, m)$, $P_3(n_1, n_2, m)$, and $P_4(n_1, n_2, m)$, where each $P_i(n_1, n_2, m)$, is a polynomial expression in n_1, n_2 and m (whose actual form depends on the specific method used). Thus, the numerical complexity of the present algorithm is $P(n_1, n_2, m) = m \sum_{i=1}^{4} P_i(n_1, n_2, m)$, which completes the proof of the theorem.

3.2 The SIHS Problem: Upper End-point of the Range

The upper end-point $\overline{x}_k^*$ is determined in, essentially, the same manner as the lower end-point $\underline{x}_k^*$. We list here the main distinctions. The global minimization problem (2.8a) is now replaced with the global maximization problem

$$\overline{x}_k^* = \max e_k^T x \tag{3.14a}$$

subject to the constraint

$$A(p)x = b(p), \quad p \in \boldsymbol{p}. \tag{3.14b}$$

The solution $\overline{x}_k^*$ is found using an iterative algorithm referred to as Algorithm Au. Its structure is similar to that of Al. At each iteration, however, use is now made of an outer solution $\boldsymbol{x}$ and a lower bound $x_k^l \le \overline{x}_k^*$. The lower bound x_k^l is given by a local optimization technique or the upper end $\overline{\zeta}_k$ of the k-th component $\boldsymbol{\zeta}_k = [\underline{\zeta}_k, \overline{\zeta}_k]$ of the IEH solution $\boldsymbol{\zeta}$ to (3.14b). Thus, the interval

$$\widetilde{x}_k = [x_k^l, \overline{x}_k] \tag{3.15}$$

is used to obtain the modified interval vector $\widetilde{\boldsymbol{x}}$ with components given in (3.4). Algorithm Au employs the procedure Pu where the main difference is that formula (3.12) now becomes

$$p_l^* = \begin{cases} \overline{p}_l, & \text{if} \quad \boldsymbol{d}_l'^{(k)} \ge 0 \\ \underline{p}_l, & \text{if} \quad \boldsymbol{d}_l'^{(k)} \le 0. \end{cases} \tag{3.16}$$

Obviously, Theorem 1 remains valid for the case of problem (3.14) if it is reformulated substituting algorithm Au for algorithm Al.

3.3 The GIHS Problem

The extension of the present approach to the GIHS problem (1.1), (2.2) is straightforward. This will be shown for the case of the lower end-point $\underline{y}_k^*$ of the k-th range $\boldsymbol{y}_k$. Indeed, the value of $\underline{y}_k^*$ is found as the global solution of the following minimization problem

$$\underline{y}_k^* = \min e_k^T y \tag{3.17a}$$

subject to the constraint

$$A(p)x = b(p), \quad p \in \boldsymbol{p}, \tag{3.17b}$$

$$y = f(x, p). \tag{3.17c}$$

The only difference is that now we need the derivatives of y_k with respect to p_l. From (3.17c)

$$\frac{\partial y_k}{\partial p_l}(p) = \frac{\partial f_k}{\partial p_l}(x, p) + \sum_{j=1}^{n} \frac{\partial f_k}{\partial x_j}(x, p) \frac{\partial x_j}{\partial p_l}(x, p). \tag{3.18}$$

Let

$$\boldsymbol{D}_l^{(k)} = \boldsymbol{\gamma}_{kl} + \sum_{j=1}^{n} \boldsymbol{\eta}_{kj} \widetilde{\boldsymbol{d}}_l^{(j)} \tag{3.19}$$

where $\widetilde{\boldsymbol{d}}_i^{(j)}$ are computed as in Step 4 of Procedure Pl of the SIHS problem while $\boldsymbol{\gamma}_{kl}$ and $\boldsymbol{\eta}_{kl}$ are the interval extension of the corresponding terms in (3.18). For instance,

$$\boldsymbol{\gamma}_{kj} = \frac{\partial f_k}{\partial x_j}(\boldsymbol{x}, \boldsymbol{p}) \tag{3.20}$$

where $\boldsymbol{x}$ is an outer solution to (3.17b). A better but more expensive way to compute $\boldsymbol{\gamma}_{kl}$ is to determine the hull solution $\boldsymbol{x}^*$ to (3.17b) and use it in (3.20) instead of $\boldsymbol{x}$. We shall illustrate this by way of example (2.5). In that case

$$\frac{\partial y}{\partial p_l}(p) = \frac{\partial f_k}{\partial p_l}(x,p) + \sum_{j=1}^{n} \frac{\partial f_k}{\partial x_j}(x,p)\frac{\partial x_j}{\partial p_l}(x,p) \tag{3.21a}$$

so

$$\boldsymbol{D}_l = 2\boldsymbol{x}_k \widetilde{\boldsymbol{d}}_l^{(k)} + 2\boldsymbol{x}_{k+n}\widetilde{\boldsymbol{d}}_l^{(k+n)} \tag{3.21b}$$

or

$$\boldsymbol{D}_l = 2\boldsymbol{x}_k^* \widetilde{\boldsymbol{d}}_l^{(k)} + 2\boldsymbol{x}_{k+n}^*\widetilde{\boldsymbol{d}}_l^{(k+n)}. \tag{3.21c}$$

Obviously

$$\frac{\partial y_k}{\partial p_l}(p) \in \boldsymbol{D}_l^{(k)}, \quad p \in \boldsymbol{p} \tag{3.22}$$

so the global or local monotonicity condition for y_k to be monotone with respect to p_l is the requirement

$$0 \notin \boldsymbol{D}_l^{(k)} = [\underline{d}_l^{(k)}, \overline{d}_l^{(k)}]. \tag{3.23}$$

As in the SIHS case, it is expedient to introduce and use the corresponding modified monotonicity condition. Therefore, we need an upper bound d_{kl}^u on $\frac{\partial y^k}{\partial p_l}(p)$, $p \in \boldsymbol{p}$ which can be found applying some local minimization method (e.g., the simple algorithm in [5]) to the right-hand side of (3.21a). Thus, the modified monotonicity condition for y_k with respect to p_l is

$$0 \notin \widetilde{\boldsymbol{D}}_l^{(k)} = [\underline{D}_l^{(k)}, d_{kl}^u]. \tag{3.24}$$

For simplicity (as in the case of SIHS problem), the short notations $\boldsymbol{D}_l^k$ and $\widetilde{\boldsymbol{D}}_l^k$ (where the respective arguments $\boldsymbol{p}^{(\nu)}$, $\boldsymbol{x}(\boldsymbol{p}^\nu)$ or $\boldsymbol{p}'^{(\nu)}$, $\boldsymbol{x}(\boldsymbol{p}'^\nu)$ have been omitted) are used. If (3.24) is satisfied for some l, the respective interval parameter $\boldsymbol{p}_l$ can be reduced to a point p_l^* using the formula

$$p_l^* = \begin{cases} \underline{p}_l, & \text{if} \quad \widetilde{\boldsymbol{D}}_l^{(k)} \geq 0 \\ \underline{p}_l, & \text{if} \quad \widetilde{\boldsymbol{D}}_l^{(k)} \leq 0. \end{cases} \tag{3.25}$$

The solution y_k^* sought can be found using the same computational scheme as in the SIHS case. Thus, as soon as an index l is detected such that the interval component $\boldsymbol{p}_l$ has been reduced to a point p_l^*, a new iteration is initiated with a reduced-width vector $\boldsymbol{p}' = (p_l^*, \boldsymbol{p})$.

There are, however, several minor modifications to be introduced in Procedure Pl and Algorithm Al. For instance, the constraint satisfaction technique is now applied to the equality

$$\widetilde{\boldsymbol{D}}_l^{(k)} = \boldsymbol{\gamma}_{kj} + \sum_{j=1}^{n} \boldsymbol{\eta}_{kl}\widetilde{\boldsymbol{d}}_l^{(j)}. \tag{3.26}$$

Let these modified versions be denoted by Procedure Pl.g and Algorithm Al.g. We have the following result.

Theorem 3.2 *For given $\{A(p), b(p), f\}$, $\boldsymbol{p}^{(0)}$ and k, assume that Algorithm Al.g terminates in T_2 with p^* whose components are defined by* (3.25). *Then the following assertions are valid:*

(i) the global solution $\underline{y}_k^*$ *of problem* (3.17) *is attained at a vertex* $p^{(\nu)}$ *for* m *iterations of Al.g;*

(ii) the vertex $p^{(\nu)}$ *sought is defined by* p^* *and is unique;*

(iii) the numerical complexity of Algorithm Al.g is polynomial in n_1, n_2 *and* m.

The proof of this theorem is similar to that of Theorem 3.1.

4 Numerical Aspects

4.1 Efficiency of an Interval Method

Let $P(\boldsymbol{p})$ denote an interval analysis problem defined for a given interval vector $\boldsymbol{p}$. Also, let $M(P(\boldsymbol{p}))$ denote an interval method capable of solving $P(\boldsymbol{p})$. Such a method will be referred to as a method applicable to problem P for $\boldsymbol{p}$. To assess the degree of applicability of $M(P(\boldsymbol{p}))$ for intervals $\boldsymbol{p}$ of various widths, we first introduce a one-parameter family of intervals $\boldsymbol{p}(\rho)$, $(\rho \in R)$ as follows: any two ρ_1, ρ_2 and the corresponding $\boldsymbol{p}(\rho_1)$, $\boldsymbol{p}(\rho_2)$ satisfy the relationship

$$\rho_1 < \rho_2 \Longleftrightarrow \boldsymbol{p}(\rho_1) \subset \boldsymbol{p}(\rho_2) \tag{4.1}$$

where the inclusion is proper. The simplest (but not unique) way to construct $\boldsymbol{p}(\rho)$ is to use a centre p^0 enclosed by a symmetric box of variable width, i.e.

$$p(\rho) = p^0 + \rho[-r^0, r^0], \tag{4.2a}$$

$$p(1) = p^0 + [-r^0, r^0] = \boldsymbol{p}^s, \tag{4.2b}$$

where $\boldsymbol{p}^s$ is a given (start) interval vector. (An alternative for constructing $\boldsymbol{p}(\rho)$ is given in [9, formula (5.4)].) Now the following measure for applicability of a given method M to a certain problem $P(\boldsymbol{p}(\rho))$, the so-called applicability radius $r_a(M)$, is defined as follows:

$$r_a(M) = \sup\left\{\rho :\ M \text{ is applicable to } P(\boldsymbol{p}(\rho)) \text{ for } \ \boldsymbol{p}(\rho) = p^0 + \rho[-r^0, r^0]\right\}. \tag{4.3}$$

The concept of applicability radius has been suggested earlier in [9] in the context of a method for determining the regularity radius of an interval matrix.

The radius $r_a(M)$ is a measure for the capacity of a given method to solve a class of problems. It also permits us to compare the relative efficiency of two methods M_1 and M_2 to solve the same problem P. Indeed, if $r_a(M_1) < r_a(M_2)$, then M_2 is numerically more efficient since M_1 fails to solve the problem at hand earlier (for an interval vector $\boldsymbol{p}(r(M_1))$ of smaller width) compared to M_2.

If the width $\boldsymbol{p}^{(0)}$ of a given problem $P(\boldsymbol{p}^{(0)})$ is such that both methods M_1 and M_2 are applicable to $P(\boldsymbol{p}^{(0)})$, then it is natural to assess their numerical efficiency by the total number of arithmetic operations $N_t(M_1)$ and $N_t(M_2)$ needed by the respective method. Obviously, there exists a relationship between $N_t(M)$ and $r_a(M)$ for a given method. As a general rule, a method M_2 that is more expensive than method M_1 (that is, $N_t(M_2) > N_t(M_1)$) is expected to have a larger radius of applicability $r_a(M_2)$ than $r_a(M_1)$. These observations will be confirmed by the theoretical considerations given below in §4.3, §4.4 and the numerical evidence related to the example considered in Section 5.

4.2 Modifications of the Basic Algorithms

The basic algorithms Al and Au developed in the previous section will be here referred to as Algorithm Al.V1 and Au.V1, respectively. In this subsection, two simplifications of the basic algorithms (denoted by Al.V2 and Al.V3 or Au.V2 and Au.V3) will be presented. The first modification of Algorithm Al.V1 (Au.V1) consists of simplifying Procedure Pl (Procedure Pu). Since Algorithms Al.V2 and Au.V2 are similar, only the versions of Algorithm Al.V2 and Procedure Pl.V2 will be considered below.

Procedure Pl.V2. The modification of Pl.V1 into Procedure Pl.V2 consists of omitting Step 3′ (where a constraint propagation is used in order to modify the initial interval vectors $\widetilde{\boldsymbol{x}}$ and $\boldsymbol{p}$ to vectors $\boldsymbol{x}'$ and $\boldsymbol{p}'$ of reduced widths).

Algorithm Al.V2 is the same as Algorithm Al.V1. The second modification denoted by Algorithm Al.V3 is obtained by simplifying Algorithm Al.V2. In this case, both Algorithm Al.V2 and Procedure Pl.V2 are changed to get the modified versions Al.V3 and Pl.V3.

Procedure Pl.V3. In the previous version Pl.V2, the iterations are terminated as soon as a modified monotonicity condition has been detected along a parameter p_l. In the present version, we continue the iterations until $l = m$, hoping to reach new parameters satisfying the respective modified monotonicity condition, modifying only the right-hand side of LIP (3.3a). More specifically, Procedure Pl.V3 includes the following steps

Step 1. For $l = 1$ to $l = m$, do:

a) form the LIP (3.3a), upgrading its right-hand side alone, and find the k-th component $\boldsymbol{d}_l^{(k)}$ of its outer solution;

b) if the modified monotonicity condition (3.9) is fulfilled, reduce the l-th interval component to a point p_l^* using (3.12).

Step 2. Let n_m denote the number of times the monotonicity condition (3.9) has been satisfied. If $n_m = 0$, go to Outcome O_1; otherwise if $n_m = m$, go to Outcome O_2; else go to Outcome O_3.

Outcome O_1: no interval component $\boldsymbol{p}_l$ has been reduced to a point.

Outcome O_2: all components $\boldsymbol{p}_l$, $l = 1, ..., m$ have been reduced to points

Outcome O_3: n_m components $\boldsymbol{p}_l$, $0 < n_m < m$ have been reduced to points so a reduced-width interval vector $\boldsymbol{p}' = (p', \boldsymbol{p})$ has been obtained where the length of $\boldsymbol{p}$ is $m - n_m$.

This procedure is used in the version Al.V3.

Algorithm Al.V3. Let ν, $\boldsymbol{p}^{(0)}$ and m_0 have the same meaning as in Algorithm Al.V1. The new version comprises the following steps.

Step 1. Compute an outer solution $\boldsymbol{x}$ to (2.8b) using an appropriate method.

Step 2. Compute an upper bound x_k^u using an appropriate method.

Step 3. Form by (3.2) the reduced-length interval $\widetilde{\boldsymbol{x}}$.

Step 4. Call Procedure Pl.V3.

Step 5. If its outcome is O_1, go to Termination T_1. In case the outcome is O_2, go to Termination T_2. If the outcome is O_3, then let $m = m - n_m$, $\boldsymbol{p} = \boldsymbol{p}'$ and go back to Step 1.

Termination T_1: The simplified Algorithm Al.V3 is not capable of solving the IH problem considered: only a two-sided enclosure $[\underline{x}_k, x_k^u]$ on $\underline{x}_k^*$ has been obtained.

Termination T_2: The simplified Algorithm Al.V3 has succeeded in solving the IH problem considered.

Remark 4.1 *A fourth modification V4 is possible when Step 3′ from version V1 is restored in Algorithm Al.V3. This version V4 will not be considered here.*

Remark 4.2 *If Algorithm Al.V2 or Algorithm Al.V3 ends in termination T_2, it can be proved that the respective algorithm has the properties enumerated in Theorem 3.1 with the exception that for Algorithm Al.V3 the number of iterations m in assertion (i) should be replaced by m' and $m' < m$.*

The version Al.V3 was motivated by the following considerations. It differs, essentially, from version Al.V1 in that the same vectors $\boldsymbol{x}$ and $\boldsymbol{p}$ are used within a current ν-th iteration, i.e. for ν fixed and l variable. This is an attempt to reach several new local monotonicity conditions for the fixed ν, skipping the more costly operations of finding a new outer interval solution $\boldsymbol{x}$ and a new upper bound x_k^u.

4.3 Numerical Characteristics of the Present Method

We first consider the numerical costs related to the various algorithms of the present method. It is easily seen that Algorithm Al.V1 is more expensive than Algorithm Al.V2; in a similar manner, Algorithm Al.V2 is more expensive than Algorithm Al.V3. Indeed, the total number of arithmetic operations N_t (Al.V1) needed by Al.V1 is given by the expression

$$N_t(\mathsf{Al.V1}) = \sum_{\nu=1}^{m} \left(\sum_{i=1}^{4} P_i^{(\nu)}(n_1, n_2, m+1-\nu) \right) \tag{4.4}$$

where $P_i^{(\nu)}(n_1, n_2, m+1-\nu)$ is the number of operations required to determine the respective term $P_i^{(\nu)}$ at the ν-th iteration (as mentioned in the proof of part (iii) of Theorem 3.1, the value of the index i corresponds to the operations associated with computing an outer solution $\boldsymbol{x}$, an upper bound x_k^u, the reduced width vectors $\boldsymbol{x}'$ and $\boldsymbol{p}'$ in Step 3′ and an outer solution $\boldsymbol{d}_l$). Obviously,

$$P_i^{(\nu)}(n_1, n_2, m+1-\nu) > P_i^{(\nu+1)}(n_1, n_2, m-\nu) \tag{4.5}$$

since $P_i^{(\nu)}(n_1, n_2, m+1-\nu)$ refers to computations involving $m+1-\nu$ interval parameters while $P_i^{(\nu+1)}(n_1, n_2, m-\nu)$ is associated with the same kind of computations involving, however, $m-\nu$ interval parameters. If $P_i^{(\nu)}$, $\nu > 1$ in (4.4) is replaced with P_i^1, we obtain the upper bound for $N_t(Al.V1)$ given in Theorem 3.1.

On account of (4.4) and (4.5):

$$N_t(\mathsf{Al.V1}) > N_t(\mathsf{Al.V2}) \tag{4.6}$$

since the term $P_3^{(\nu)}$ associated with Step 3′ in Procedure Pl.V2 (constraint propagation) is missing in Algorithm Al.V2. Also

$$N_t(\mathsf{Al.V2}) \geq N_t(\mathsf{Al.V3}) \tag{4.7}$$

since now

$$N_t(\mathsf{Al.V3}) = \sum_{\nu=1}^{m'} \left(\sum_{i=1}^{3} P_i^{(\nu)}(n_1, n_2, m'+1-\nu) \right) \tag{4.8}$$

where typically $m' < m$.

We now compare the various algorithms with regard to their radii of applicability. In view of (4.6) and (4.7) it is expected that

$$r_a(\mathsf{Al.V1}) \geq r_a(\mathsf{Al.V2}) \geq r_a(\mathsf{Al.V3}). \tag{4.9}$$

We shall show only the validity of the first relation (the argument for the second is similar). Suppose that the width of the interval vector $\boldsymbol{p}(\rho)$ is such that ρ is slightly larger than $r_a(\mathsf{Al.V2})$ so Algorithm AlV2 is not applicable. In that case, we can try the more expensive Algorithm Al.V1 to solve the problem considered. While Algorithm Al.V2 relies only on the modified monotonicity conditions related to the current parameter domain $\boldsymbol{p}$ and the associated initial domain $\widetilde{\boldsymbol{x}}$ of the phase variables, Algorithm Al.V1 uses additionally the constraint propagation capable of contracting $\boldsymbol{x}$ and $\boldsymbol{p}$ to narrower vectors $\boldsymbol{x}'$ and $\boldsymbol{p}'$. Hence, the updated modified monotonicity conditions associated now with $\boldsymbol{x}'$ and $\boldsymbol{p}'$ will be easier to satisfy, leading to a larger radius of applicability for Algorithm Al.V1.

4.4 Comparison with Other Methods

We now compare the new method (referred to as method MK) with two other methods of polynomial complexity, namely [17] (referred to as method MP) and [5] (referred to as method M0) in a qualitative manner.

First, we consider the methods MK and MP. As mentioned in Section 2, the method MP is also an iterative method capable of determining the lower and upper end of the IH solution component x_k^*. To be able to carry out such a comparison, we assume that both methods compute the necessary outer interval solutions in the same manner. To be specific, only the case of computing $\underline{x}_k^*$ will be considered here; so we consider a corresponding algorithm of method MP which will be referred to as Algorithm Al.P. We start by comparing Algorithm Al.P with the simplest version of method MK, namely Algorithm Al.V3. The two algorithms differ in that the upper bound x_k^u is not computed and used in Algorithm Al.P. Hence, the cruder monotonocity condition (3.8) is exploited in the latter algorithm to reduce the width of the initial parameter vector $\boldsymbol{p}^{(0)}$. In contrast, the present Algorithm Al.V3 is based on the use of the more effective modified monotonicity criterion (3.9). Therefore, in view of Lemma 3.1 the new version Al.V3 is never worse and is expected to be actually more efficient than the known version Al.P with regard to the radius of applicability of the two algorithms. Indeed, it is natural to suppose that, at each iteration, the standard monotonicity condition (3.8) is more difficult to satisfy than the respective modified monotonicity condition (3.9). This is due to the fact that the outer solution $\boldsymbol{x}$ is involved in computing while the modified vector $\widetilde{\boldsymbol{x}}$ is used to evaluate $\widetilde{\boldsymbol{d}}_l^{(k)}$. Since $\widetilde{\boldsymbol{x}} \subseteq \boldsymbol{x}$ by (3.4b) and $\widetilde{\boldsymbol{d}}_l^{(k)} \subseteq \boldsymbol{d}_l^{(k)}$ by (3.7b), the overestimation of $\boldsymbol{d}_l^{(k)}$ (with regard to the true range of the respective derivative $\partial x_k/\partial p_l$ over the current $\boldsymbol{p}$) will, in general, be larger than the overestimation of $\widetilde{\boldsymbol{d}}_l^{(k)}$. This, in turn, results in easier violation of conditions (3.8) than conditions (3.9). Hence, it is expected that most often

$$r_a(\texttt{Al.P}) < r_a(\texttt{Al.V3}). \tag{4.10}$$

It should also be stressed that if the general-purpose method of [14] is used to find $\boldsymbol{x}$ and ζ, the upper bound $x_k^u = \underline{\zeta}_k$ needed in version Al.V3 is obtained with no additional computational cost. In that case, the total number of arithmetic operations

$N_t(MK)$ and $N_t(MP)$ needed by the respective method will be the same. If, a local optimization technique is used to locate x_k^u, then obviously $N_t(MK) > N_t(MP)$. On the other hand, that bigger cost may be compensated by a relatively larger radius of applicability $r_a(\mathsf{Al.MK})$ since always $\underline{\zeta}_k > \underline{x}_k^*$ while often (especially for relatively narrow $\boldsymbol{p}$) $x_k^u = \underline{x}_k^*$.

In cases where the problem at hand has a specific structure, a specialized method such as [10] or [24] should be employed in either method MP or method MK. The specialized methods, however, do not provide any inner estimation vector $\boldsymbol{\zeta}$ so a local optimization technique is to be used to locate x_k^u. Obviously, $N_t(MK) > N_t(MP)$ in that case. Again, it might pay, in the long run, to accept that bigger computational cost since on the average method MK will provide a relatively larger radius of applicability.

Next we compare the new approach with the method M0 of polynomial complexity [5]. As mentioned in Section 2, the method M0 is also a componentwise method, separately determining (for a fixed k) the lower and upper end of the IH solution component $\boldsymbol{x}_k^*$. The corresponding algorithm for computing $\underline{x}_k^*$ will here be referred to as Algorithm Al.V0.

Algorithm Al.V0. In fact, this algorithm is a simplified version of Algorithm Al.V3. From this point of view, it consists of skipping Steps 2 and 3 in Procedure Pl.V3, that is, we do not compute and use the inner bound x_k^u. Thus, only the outer interval solution $\boldsymbol{x}$ from Step 1 of Procedure Pl.V3 is employed in the subsequent computations. Hence, the cruder global monotonocity condition (3.8) is exploited to reduce the width of the initial parameter vector $\boldsymbol{p}^{(0)}$.

It is obvious that, for reasons similar to those considered in the comparison of MK and MP, algorithm Al.V3 should outperform Al.V0. In particular, it is expected that

$$r_a(\texttt{Al.V0}) < r_a(\texttt{Al.V3}). \tag{4.11}$$

The latter inequality is confirmed numerically in Section 5. Furthermore, better results should be obtained if the more expensive Algorithm Al.V2 or Al.V1 is used rather than Algorithms Al.K or Al.V0.

It should also be stressed that, unlike all known methods, the best version Al.V1 of the present method is capable of detecting those coordinates of the initial domain $\boldsymbol{p}$ along which the vertex property of the solution $\underline{x}_k^*$ sought is violated (Lemma 3.2).

5 Numerical Example

We illustrate the method suggested by way of an example. All LIP systems considered below are square and defined by the affine functions (1.2). They are given equivalently in the form

$$A(p) = A^{(0)} + \sum_{\mu=1}^{m} A^{(\mu)} p_\mu, \tag{5.1a}$$

$$b(p) = b^{(0)} + Bp, \tag{5.1b}$$

where $A^{(0)}$, $A^{(\mu)}$, $\mu = 0, 1, ..., m$ are $n \times n$ real matrices while B is a $n \times m$ real matrix. The outer interval solutions $\boldsymbol{x}$ and $\widetilde{\boldsymbol{d}}_l^{(k)}$ were determined using the direct method of [4]. The upper bound x_k^u or lower bound x_k^l were computed using the simple iterative method of [5]. Algorithms Al.V3 and Au.V3 were employed to determine the lower or upper end-point of the k-th component $\boldsymbol{x}_k^*$ of $\boldsymbol{x}$. The algorithms were programmed in the MATLAB environment using the INTLAB toolbox [23] to carry

out the interval calculations. The programs were run on a 1.7 GHz PC computer. The linear parametric system is given by the matrix [25]

$$A(p) = \begin{bmatrix} p_1 & p_2+1 & -p_3 \\ p_2+1 & -3 & p_1 \\ 2-p_3 & 4p_2+1 & 1 \end{bmatrix} \tag{5.2a}$$

and the vector

$$b(p) = \begin{bmatrix} 2p_1 \\ p_3-1 \\ -1 \end{bmatrix}. \tag{5.2b}$$

Thus, $n = 3$ and $m = 3$. It is seen that

$$A^{(0)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -3 & 0 \\ 2 & 1 & 1 \end{bmatrix}, A^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, A^{(2)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 4 & 0 \end{bmatrix}, A^{(3)} = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \tag{5.2c}$$

and

$$b^{(0)} = \begin{bmatrix} 0 \\ -1 \\ -1 \end{bmatrix}, \quad B = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}. \tag{5.2d}$$

The initial interval vector $\boldsymbol{p}^s = (\boldsymbol{p}_1, ..., \boldsymbol{p}_3)$ is given by (4.2) with

$$p^0 = (0.5\ 0.5\ 0.5)^{\mathrm{T}}, \qquad r^0 = (0.5\ 0.5\ 0.5)^{\mathrm{T}}. \tag{5.3a}$$

In the following three subsections, we use the data (5.2), (5.3) to illustrate the application of the present method to solving the SIHS problem. The solution of a GIHS problem is treated in the fourth subsection.

5.1 Fixing an Index

In this instance, the parameter vector $p = (p_1, ..., p_3)$ belongs to $\boldsymbol{p} = (\boldsymbol{p}_1, ..., \boldsymbol{p}_3)$ defined by (4.2a) and (5.3a) for $\rho = 0.1$, i.e.

$$\boldsymbol{p} = p^0 + 0.1[-r^0, r^0]. \tag{5.3b}$$

The problem considered here is to determine the range $\boldsymbol{x}_3^*$ so the fixed index is $k = 3$.

We first determine the lower end-point $\underline{x}_3^*$. Application of Algorithm Al.V3 yields

$$\underline{x}_3^* = -1.7786. \tag{5.4}$$

The algorithm takes two iterations to reach $\underline{x}_3^*$. At the first iteration, Procedure Pl.V3 succeeds in reducing two interval variables $\boldsymbol{p}_1$ and $\boldsymbol{p}_3$ to points $\overline{p}_1^*$ and $\underline{p}_3^*$, respectively. The interval parameter $\boldsymbol{p}_2$, however, remains unchanged. Thus, we form the reduced-width vector

$$\boldsymbol{p}' = (\overline{p}_1, \boldsymbol{p}_2, \underline{p}_3)^{\mathrm{T}}. \tag{5.5}$$

Now $\boldsymbol{p}'$ is substituted into (5.2) to get a modified system of the form

$$A'(\boldsymbol{p}_2) = A^{(0)} + A^{(1)}\overline{p}_1 + A^{(3)}\underline{p}_3 + A^{(2)}\boldsymbol{p}_2 = A' + A^{(2)}\boldsymbol{p}_2 \tag{5.6a}$$

$$b'(\boldsymbol{p}_2) = b^{(0)} + B^{(1)}\overline{p}_1 + B^{(3)}\underline{p}_3 + B^{(2)}\boldsymbol{p}_2 = b' + B^{(2)}\boldsymbol{p}_2 \tag{5.6b}$$

Table 1: Data on Algorithm Al.V3 for $k = 3$ and $\rho = 0.1$

ν	$\underline{x}_3$	x_3^u	$n_{\rm in}$	$\boldsymbol{p}'$	$\underline{x}_3^*$
1	-1.7982	-1.7785	2	$(0.55\ p_2\ 0.45)^T$	
2	-1.7800	-1.7785	2	$(0.55\ 0.55\ 0.45)^T$	-1.7786

($B^{(j)}$ denotes the corresponding jth column of matrix B). At this point, the second iteration of Algorithm Al.V3 is initiated. This time, Procedure Pl.V3 is applied to (5.6) after $\boldsymbol{p}_2$ has been renamed $\boldsymbol{p}$. Now the last interval parameter $\boldsymbol{p} = \boldsymbol{p}_2$ is successfully reduced to the end-point $\overline{p}_2$. Thus, it is seen that for the problem considered Algorithm Al.V3 terminates in outcome T_2 with

$$p^* = (0.55 \quad 0.55 \quad 0.45)^T. \tag{5.7}$$

Finally, the global solution (5.4) of (5.2) is obtained after solving

$$A(p^*)x = b(p^*) \tag{5.8}$$

for $\check{x}$ and letting $x_3^* = \check{x}_3$. In Table 1, data concerning the algorithm used and the results obtained are given.

The current iteration number of Algorithm Al.V3 is denoted by ν; n_{in} denotes the number of iterations needed by the local optimization (LO) method used to compute the upper (inner) bound x_k^u. The lower end $\underline{x}_k$ of the outer enclosure $[\underline{x}_k, \overline{x}_k]$ at the corresponding iteration ν is given in the second column of the table. In the next two columns, the values of x_k^u and n_{in} are listed for each ν. The parameter vectors of reduced width are presented in the fifth column. The last column includes the approximate value of $\underline{x}_3^*$ (for $\nu = 2$). The optimal parameter vector p^* is given in the second entry of the fifth column.

Remark 5.1 *The complexity of the Algorithm Al.V3 can be assessed by the number N_{ls} of $(n \times n)$ linear systems solved. We shall distinguish between N_{ip} and N_{pp}: number of interval parameter (IP) systems and number of point (noninterval) parameter systems, respectively, since the former systems are harder to solve than the latter ones. To simplify the analysis, we assume that solving (2.8b) once and (3.3a) m_ν times (m_ν being the number of interval parameters at each ν-th iteration) can be equated to solving one single IP system since $A(p)$ is the same for all $m_\nu+1$ systems. This is a reasonable assumption if $m \leq n$, which is often the case in practice [5]. Then, as is easily seen, N_{ip} varies between $\underline{N}_{ip} = 1$ (in the best case) and $\underline{N}_{ip} = m$ (in the worst case). The number N_{pp} is given by the sum of $n_{in}^{(\nu)}$ over ν where $n_{in}^{(\nu)}$ is the number of iterations needed by the LO method chosen to find the bound $\underline{x}_k^\nu$ on $\underline{x}_k^*$ at the ν-th iteration. For the LO method of [5] used in the present example, $n_{in}^{(\nu)}$ varies between $n_{in}^{(\nu)} = 2$ (minimum value) and $n_{in}^{(\nu)} = m_\nu + 1$ (maximum admissible value). Thus $\underline{N}_{pp} = 2$ in the best case; in the worst case, $\overline{N}_{pp} = (m+1)+(m-1)+\ldots+2+1 = (m+1)(m+2)/2$. Therefore, $N_{ls} = N_{ip}+N_{pp}$ varies between $\underline{N}_{ip}+\underline{N}_{pp} = 3$ and $\overline{N}_{ip}+\overline{N}_{pp} = m+(m+1)(m+2)/2$.*

It is interesting to note that the number N_{ip} related to versions V2 and V1 of Algorithm Al is the same and equal to $\overline{N}_{ip}$ associated with version V3.

Similar results are obtained in determining the upper end-point $\overline{x}_3^*$ using Algorithm Au.V3. These are reported in Table 2 (n_{in} denoting the number of iterations needed by the LO method used to compute the upper (inner) bound x_k^l).

Table 2: Data on Algorithm Au.V3 for $k = 3$ and $\rho = 0.1$

ν	$\overline{x}_3$	x_3^l	n_{in}	$\boldsymbol{p}'$	$\overline{x}_3^*$
1	-1.3447	-1.3824	2	$(0.45\ p_2\ 0.55)^T$	
2	-1.3823	-1.3824	2	$(0.45\ 0.45\ 0.55)^T$	-1.3823

Remark 5.2 *The approximate four digit values for $\underline{x}_3$, x_3^u and $\underline{x}_3^*$ reported in Table 5.1 and $\overline{x}_3$, x_3^l and $\overline{x}_3^*$ in Table 5.2 are obtained after appropriate directed roundings. Thus, downward rounding has been used to represent $\underline{x}_3$, x_3^* and x_3^l while upward rounding is needed for $\overline{x}_3$,$\overline{x}_3^*$ and x_3^u. It should be borne in mind that the appropriate roundings of $\underline{x}_k$, x_k^u and $\overline{x}_k$, x_k^l are mandatory when rigorously implementing the present method in order to provide reliable monotonic properties and final results for $\underline{x}_k^*$ and $\overline{x}_k^*$.*

5.2 Computing all of $\boldsymbol{x}$

The problem in this subsection is to determine the whole IH vector $\boldsymbol{x}^*$ associated with problem (5.2), (5.3). According to the present paper's approach, $\boldsymbol{x}^*$ is found in a componentwise manner, i.e. by separately computing each end-point of each range $\boldsymbol{x}_k^*$. The data for $k = 3$ have been given in Tables 1 and 2 from the previous example. Thus, algorithm Al.V3 and Au.V3 remain to be applied for $k = 1$ and $k = 2$. Tables 3 and 4 summarize the relevant results.

Table 3: Data on Al.V3 and Au.V3 for $k = 1$ and $\rho = 0.1$

Al.V3	ν	n_{in}	p^*	$\underline{x}_1^*$
	1	2	$(0.45\ 0.55\ 0.55)^T$	0.1826
Au.V3	ν	n_{in}	p^*	$\overline{x}_1^*$
	1	2	$(0.55\ 0.45\ 0.45)^T$	0.4052

Table 4: Data for Al.V3 and Au.V3 for $k = 2$ and $\rho = 0.1$

Al.V3	ν	n_{in}	p^*	$\underline{x}_2^*$
	2	2	$(0.55\ 0.45\ 0.55)^T$	0.0277
Au.V3	ν	n_{in}	p^*	$\overline{x}_2^*$
	3	2	$(0.45\ 0.45\ 0.45)^T$	0.0654

5.3 Determining the Applicability Radius

We now consider the problem of determining the applicability radius r_a of algorithm Al.V3 of the present method for the case of $k = 2$. With this in mind, we introduce

Table 5: Data on the applicability radii of algorithms Al.V3 and Al.V0 for $k = 2$

algorithm	ν	r_a	p^*	$\underline{x}_2^*$
Al.V3	3	0.165	$(0.5825,\ 0.4175,\ 0.5825)^T$	0.0137
Al.V0	3	0.104	$(0.5520,\ 0.4480,\ 0.5520)^T$	0.0269

the family (4.2)

$$\boldsymbol{p}(\rho) = p^0 + \rho \boldsymbol{p}^s \tag{5.9}$$

where p^0 and $\boldsymbol{p}^s$ are given by (5.3a). In accordance with the definition (4.3), we estimate $r_a(Al)$ of the respective algorithm approximately by letting ρ increase with an increment $\Delta\rho$ until inapplicability is reached. The data concerning Algorithm Al.V3 are given in the second row of Table 5.

Around the "critical" value of r_a, the increment of ρ was chosen to be $\Delta\rho = 0.001$. Thus, $r_a(\text{Al.V3}) = 0.165$ means that Algorithm Al.V3 becomes inapplicable for $\rho = r_a(\text{Al.V3}) + \Delta\rho = 0.166$.

We now compare $r_a(\text{Al.V3})$ with the applicability radius of the method M0 from [5] (in fact, version M3). To make the comparison invariant with respect to the way the enclosures $\boldsymbol{d}_l^{(k)}$ and $\widetilde{\boldsymbol{d}}_l^{(k)}$ are computed, $\boldsymbol{d}_l^{(k)}$ was evaluated in the same manner as $\widetilde{\boldsymbol{d}}_l^{(k)}$. This modified version of M0 is denoted by Algorithm Al.V0. The numerical evidence shows that approximately Al.V0 fails to be applicable for $\rho = 0.105$ (third row of Table 5). It is seen that, in accordance with the theoretical consideration from Section 4.1,

$$r_a(\text{Al.V3}) > r_a(\text{Al.V0}). \tag{5.10}$$

Thus, it has been shown that Algorithm Al.V3 of the present method is capable of solving LIP problems of larger uncertainties than Algorithm Al.V0 of the previous method M0 [5].

5.4 Solving a GIHS Problem

In this final subsection, we illustrate the application of the present method to solving a GIHS problem. The parametric system is again (5.2), (5.3) while the output variable vector y is the scalar

$$y(p) = \sum_{k=1}^{3} x_k^2(p). \tag{5.11}$$

If we interpret the components x_k of x as the projections of the vector x onto the axes in Euclidian space, then (5.11) has a clear geometrical meaning: y is the square of the length of the vector x. Thus, the GIHS defined by (5.2), (5.3) and (5.11) consists of determining the range $\boldsymbol{y}^*$ of y over $\boldsymbol{p}$. On account of (5.11),

$$\frac{\partial y}{\partial p_l}(p) = 2\left[\sum_{k=1}^{3} x_k(p)\frac{\partial x_k}{\partial p_l}(p)\right]. \tag{5.12}$$

Now we bound $\frac{\partial y}{\partial p_l}$ using different enclosures $\boldsymbol{x}_k$ and $\boldsymbol{d}_l^{(k)}$ for $x_k(p)$ and $\frac{\partial x_k}{\partial p_l}(p)$ over $\boldsymbol{p}$, respectively. First, we use data related to the standard (global) monotonicity

conditions. Therefore, we use the outer bounds $\boldsymbol{x}_k$ and $\boldsymbol{d}_l^{(k)}$ obtained at the first iteration of algorithms Al.V0 and Au.V0. We have (after canceling the factor of 2)

$$\boldsymbol{D}_l = \sum_{k=1}^{3} \boldsymbol{x}_k \boldsymbol{d}_l^{(k)}, \tag{5.13}$$

so

$$\boldsymbol{D}_1 = [0.3533, 2.0732] > 0, \rangle \boldsymbol{D}_2 = [0.1419, 0.6657] > 0, \rangle \ \boldsymbol{D}_3 = [-5.3009, -1.6776] > 0. \tag{5.13a}$$

Hence, on account of (5.11) to (5.13a)

$$\underline{y}^* = \sum_{k=1}^{3} x_k^2(p^{(l)}) = 1.9108 \tag{5.14a,)}$$

where

$$p^{(l)} = (\underline{p}_1, \underline{p}_2, \overline{p}_3)^T = (0.45\ 0.45\ 0.55)^T. \tag{5.14b}$$

The corresponding vector $x(p^{(l)})$ is the solution of $A(p^{(l)})x = b(p^{(l)})$. In a similar manner

$$\overline{y}^* = \sum_{k=1}^{2} x_k^2(p^{(u)}) = 3.1631 \tag{5.15a}$$

where

$$p^{(u)} = (\overline{p}_1, \overline{p}_2, \underline{p}_3)^T = (0.55,\ 0.55,\ 0.45)^T \tag{5.15b}$$

and $x(p^{(u)})$ is the solution of $A(p^{(u)})x = b(p^{(u)})$. Thus

$$\boldsymbol{y}^* = [1.91083.1631]. \tag{5.16}$$

A narrower interval $\boldsymbol{D}_l^*$ bounding $\frac{\partial y}{\partial p_l}(p)$, $p \in \boldsymbol{p}$ is obtained if the $\boldsymbol{x}_k$ in (5.13) are replaced with $\boldsymbol{x}_k^*$ (since $\boldsymbol{x}_k^* \subset \boldsymbol{x}_k$):

$$\boldsymbol{D}_l^* = \sum_{k=1}^{3} \boldsymbol{x}_k^* \boldsymbol{d}_l^{(k)}. \tag{5.17}$$

The data for $\boldsymbol{D}_l$ and $\boldsymbol{D}_l^*$ are reported in the second and third row, respectively, of Table 6.

Next, we bound $\frac{\partial y}{\partial p_l}$ using the modified monotonicity approach. We start with the case of determining modified monotonicity conditions $\widetilde{\boldsymbol{D}}_l^{(l)}$ related to determining the lower end-point $\underline{y}^*$ of the range $\boldsymbol{y}^*$. In that case, $\boldsymbol{x}_k$ will be replaced with $\widetilde{\boldsymbol{x}}_k = [\underline{x}_k, x_k^u]$; the corresponding modified derivatives $\widetilde{\boldsymbol{d}}_l^{(k)}$ (computed at the first iteration of Al.V3) will replace the previous $\boldsymbol{d}_l^{(k)}$. The values of $\widetilde{\boldsymbol{D}}_l^{(l)}$ for that case are given in the fourth row of Table 6.

In a similar manner, we determine modified monotonicity conditions $\widetilde{\boldsymbol{D}}_l^{(u)}$ related to the upper end-point $\overline{y}^*$ of $\boldsymbol{y}^*$. Now $\boldsymbol{x}_k$ and $\widetilde{\boldsymbol{d}}_l^{(k)}$ will be replaced with $\widetilde{\boldsymbol{x}}_k = [x_k^l, \overline{x}_k]$ and $\widetilde{\boldsymbol{d}}_l^{(k)}$, respectively. The values for $\widetilde{\boldsymbol{D}}_l^{(u)}$ are given in the fifth row of Table 6.

As expected, the intervals $\widetilde{\boldsymbol{D}}_l^{(l)}$ and $\widetilde{\boldsymbol{D}}_l^{(u)}$ provide more effective (easer to satisfy) monotonicity conditions as compared to $\boldsymbol{D}_l$ or even $\boldsymbol{D}_l^*$. Indeed, the lower end-points $\underline{D}_l^{(l)}$, $l = 1$ and $l = 2$, of the respective intervals $\widetilde{\boldsymbol{D}}_l^{(l)}$ are much higher then the lower

Table 6: Data for $\boldsymbol{D}_l$, $\widetilde{\boldsymbol{D}}_l^*$, $\widetilde{\boldsymbol{D}}_l^{(l)}$ and $\boldsymbol{D}_l^{(u)}$

l	1	2	3
$\mathbf{D}_l$	$[0.3533, 2.0732]$	$[0.1419, 0.6657]$	$[-5.3009, -1.6776]$
$\boldsymbol{D}_l^*$	$[0.4232, 2.0383]$	$[0.1560, 0.6550]$	$[-5.2275, -1.8185]$
$\widetilde{\boldsymbol{D}}_l^{(l)}$	$[0.7211, 1.6215]$	$[0.3156, 0.6300]$	$[-4.6656, -3.5115]$
$\widetilde{\boldsymbol{D}}_l^{(u)}$	$[0.8561, 1.6565]$	$[0.1758, 0.4961]$	$[-3.5709, -2.5061]$

end-points of $\boldsymbol{D}_l$ and $\boldsymbol{D}_l^*$. Also, the upper end-point $\overline{D}_3^{(l)}$ is much lower then the upper end-points of $\boldsymbol{D}_3$ and $\boldsymbol{D}_3^*$. Similarly, the upper-end points $\overline{D}_l^{(u)}$, $l = 1$ and $l = 2$, are much lower and the lower end-point $\underline{D}_3^{(u)}$ is much higher than the respective end-points of D_l and $\boldsymbol{D}_l^*$. Thus, the applicability radius of the new method is bound to be larger than the applicability radius of previous methods also in the context of GIHS problems.

6 Conclusion

A unified method for determining the interval hull solutions to various problems associated with linear interval parameter systems (systems of non-linear (1.1) or linear parametric dependencies (1.2), problems of standard (SIHS) or generalized (GIHS) form, square or rectangular systems) has been suggested. In all the cases, it reduces to iteratively solving global optimization problems of the type (2.8) and (3.17) to find the lower end-point $\underline{x}_k^*$ and upper end-point $\overline{x}_k^*$ of the phase variable component $\boldsymbol{x}_k^*$ or the end-points $\underline{y}_k^*$ and $\overline{y}_k^*$ of the output variable component $\boldsymbol{y}_k^*$, respectively. Each global solution is attained in an efficient manner using the respective modified monotonicity conditions (3.9) or (3.24) that have been proved in Lemma 3.1 to be not worse and, if the sufficient conditions (3.6a) are valid, to be better than the standard monotonicity conditions (3.8) or (3.23). The method, basically, is capable of determining the global solution sought if it has the vertex property by checking whether the above modified monotonicity conditions are satisfied for each iteration: Theorems 3.1 and 3.2. Otherwise, it only provides a two-sided enclosure of $\underline{x}_k^*$ ($\overline{x}_k^*$) or $\underline{y}_k^*$ ($\overline{y}_k^*$).

Three versions of the algorithms designed for determining $\underline{x}_k^*$ ($\overline{x}_k^*$) have been developed in Sections 3 and 4. It is shown that their complexity is polynomial in the size n_1 and n_2 of the linear interval parameter problem studied and the length m of the interval parameter vector $\boldsymbol{p}$. A numerical example provided in Section 5 confirms the characteristics of the algorithms demonstrated theoretically in Section 4 and illustrates the fact that the present method outperforms similar methods of polynomial numerical complexity [5], [17] with regard to its radius of applicability (4.3) (capacity to solve problems of larger uncertainty intervals).

Future research will concentrate on extending the present approach to problems where the vertex property is not valid, i.e. where the global optimum $\underline{x}_k^*$ of problem (2.8) (or $\overline{x}_k^*$ of problem (3.7)) is attained at a point p^* in $\boldsymbol{p}$ that is not a vertex.

Acknowledgements

The author wishes to express his gratitude to two of the anonymous referees for their valuable comments and suggestions.

References

[1] G. Alefeld, V. Kreinovich and G. Mayer. On the solution sets of particular classes of linear interval systems. *J. Comput. Appl. Math.* 152:1–15, 2003.

[2] C. Jansson. Interval linear systems with symmetric matrices, skew-symmetric matrices and dependencies in the right-hand side, *Computing* 46:265–274, 1991.

[3] L. Kolev. *Interval Methods for Circuit Analysis*. World Scientific, Singapore, New Jersey, London, 1993.

[4] L. Kolev. Outer solution of linear systems whose elements are affine functions of interval parameters, *Reliable Computing*, 8:493–501, 2002.

[5] L. Kolev. Worst-case tolerance analysis of linear DC and AC electric circuits, *IEEE Trans. on Circuits and Systems - I: Fundamental Theory and Appl.*, 49:1693–1701, 2002.

[6] L. V. Kolev. Solving linear systems whose elements are nonlinear functions of interval parameters. *Proceedings of the International Symposium of Scientific Computing, Computer Arithmetic, and Validated Numerics , SCAN 2002*, September 24-27, Paris, France, 2002, p. 45.

[7] L.V. Kolev. Solving linear systems whose elements are nonlinear functions of interval parameters. *Numerical Algorithms*, 37:199–212, 2004.

[8] L. Kolev. A method for outer interval solution of linear parametric systems. *Reliable Computing*, 10:227–239, 2004.

[9] L. V. Kolev. A method for determining the regularity radius of interval matrices. *Reliable Computing*, 16:1–26, 2011.

[10] A. Neumaier and A. Pownuk. Linear systems with large uncertainties, with applications to truss structures. *Reliable Computing*, 13:149–172, 2007.

[11] K. Okumura. An application of interval operation to electric network analysis. In *Bullletin of the Japan Society for Industrial & Applied Mathematics* 2:115–127, 1993.

[12] E. Popova. On the solution of parameterised linear systems. In *Scientific Computing, Validated Numerics, Interval Methods*, Kluwer Academic Publishers, Boston, pages 127–138, Kraemer, W. and Wolff von Gudenberg, J. (Eds.), 2001.

[13] Popova, E.D., Datcheva, M., Iankov, R., Schanz, T.: Mechanical models with interval parameters. In K. Geurlebeck, L. Hempel, C. Keonke (Eds.): *Proceedings of 16th International Conference on the Applications of Computer Science and Mathematics in Architecture and Civil Engineering*, 2003.

[14] E. Popova. Generalization of the parametric fixed-point iteration, *PAAM* 4:680–681, 2004.

[15] E. Popova. Improved solution enclosures for over- and underdetermined interval linear systems. In *Lecture Notes in Computer Science 3743*, pages 305–312, I.Lirkov, S. Margenov, J. Wasniewski (Eds.), 2006.

[16] E. Popova, R. Yankov and Z. Bonev. Bounding the response of mechanical structures with uncertainties in all the parameters. In *Proceedings of the NSF Workshop on Reliable Engineering Computing*, Savannah, Georgia USA, 2006, pages 245–265, R.L.Muhanna, R.L.Mullen (Eds.), 2006.

[17] E. Popova. Computer-assisted proofs in solving linear parametric problems. In *Conference Post-Proceedings of SCAN 2006*, IEEE Computer Society Press, 2007, page 35, 2007.

[18] Pownuk, A., Efficient method of solution of large scale engineering problems with interval parameters based on sensitivity analysis, *Proceedings of the NSF workshop on Reliable Engineering Computing*, September 15-17, 2004, Savannah, Georgia, USA, pp. 305–316, 2004.

[19] Rama Rao, M.V., Mullen, R.L. and Muhanna, R.L. Primary and derived variables with the same accuracy in interval finite elements. In M. Beer, R. L. Muhanna, R. L. Mullen (Eds.), *Fourth International Workshop on Reliable Engineering Computing (REC 2010)*, Research Publishing Services, National University of Singapore, 2010.

[20] Rama Rao, M.V., Mullen, R.L. and Muhanna, R.L. (2011) A new interval finite element formulation with the same accuracy in primary and derived variables, *Int. J. Reliability and Safety*, 5(3–4):336–357, 2011.

[21] J. Rohn. Linear interval equations with dependent coefficients. In *International Conference on Interval Methods for Numerical Computation*, Oberwohlfach, West Germany, 1990.

[22] J. Rohn. A method for handling dependent data in interval linear systems. Technical Report No. 911, Institute of Computer Science, Academy of Science of the Czech Republic, July 2004.

[23] S. Rump. INTLAB - INTerval LABoratory, in Tibor Csende, ed., *Developments in Reliable Computing* , pp. 77–105. Kluwer, Dordrecht, Netherlands, 1999.

[24] I. Skalna. A method for outer interval solution of parameterized systems of linear equations. *Reliable Computing*, 12:107–120, 2006.

[25] I. Scalna. Evolutionary optimization method for approximating the solution set hull of parametric linear systems, *Lecture Notes in Computer Science 4310*, pp. 361–368, 2006.

[26] I. Skalna. On checking the monotonicity of parametric interval solution of linear structural systems. *Lecture Notes in Computer Science 4967* pp. 1400–1409, 2008.

[27] I. Skalna, A. Pownuk: On using global optimization method for approximating interval hull solution of parametric linear systems. In *Proceedings of Third International Workshop on Reliable Engineering Computing (REC08)*, Georgia Institute of Technology, Feb 20-22, 2008, Savannah, Georgia, USA, 2008.

[28] I. Skalna and A. Pownuk, A global optimisation method for computing interval hull solution for parametric linear systems, *International Journal of Reliability and Safety* 3 (1–3):235–245, 2009.