Methods For Interval Linear Equations

Eldon Hansen

Abstract. We discuss one known and five new interrelated methods for bounding the hull of the solution set of a system of interval linear equations. Each method involves a polynomial amount of computing; but requires considerably more effort than Gaussian elimination. However, each method can yield sharper results for appropriate problems. For certain problems, our methods can obtain sharp bounds for one or more (and perhaps all) components of the hull of the solution set.

1. Introduction

Consider a set of linear equations

$$Ax = b \tag{1.1}$$

where $A = [\underline{A}, \overline{A}]$ is an interval matrix of order n and $b = [\underline{b}, \overline{b}]$ is an interval vector of n components. The problem of determining the interval hull h of the solution set is NP-hard. (See [5].)

In practice, one can accept non-sharp bounds on h obtained by applying an interval version of Gaussian elimination. Only a polynomial amount of computing of order $O(n^3)$ is needed. Unfortunately, bounds obtained in this way can be far from sharp because of growth of interval widths caused by dependence. (See [2] or [4].). In fact, the method can fail even when A is regular. See [6].

The author introduced preconditioning (see [1]) to reduce the effect of dependence in Gaussian elimination. In this procedure, the equation $Ax = b$ is multiplied by an approximate inverse B of the center of A. Unfortunately, preconditioning generally enlarges the solution set. Thus, the hull h^P of the preconditioned system $BAx = Bb$ contains h; but is generally larger than h.

Later, the author found that the hull of the preconditioned system could be determined exactly (except for roundoff). In [4], the procedure to do so is called the "hull method". The hull method requires somewhat more computing then applying Gaussian elimination to the preconditioned system. Nevertheless, it is the recommended method. The hull method fails only if the preconditioned matrix BA is irregular.

Another way to get bounds on h is to use a version of Gaussian elimination which involves use of what the author called "parameter dependent monotonicity". A procedure of this kind has been described by the author [3]. It requires more computing that ordinary Gaussian elimination; but yields sharper results. It does not involve preconditioning which enlarges the solution set.

In this paper, when we refer to "crude bounds" on the hull h, we mean bounds obtained by a method such as described above. The term is meant to imply that the bounds are

obtained by a relatively efficient method; and the bounds include the exact result, but are not sharp.

In subsequent sections, we introduce six interrelated methods for bounding h. They require a polynomial amount of computing; but considerably more computing than is needed to get crude bounds. Each requires solving linear programming problems. A weakness of our methods is that they are not applicable if the crude methods fail to produce bounds.

For a small fraction of possible linear systems, the methods in Sections 3 and 4 provide the exact (except for roundoff) hull of the solution set. For a slightly larger fraction of systems, the method of Section 4 provides exact bounds on one or more components of h.

We define and discuss sign-definiteness in the next section. All our methods use this property. In Section 5, we describe ways to precondition a system of equations to produce a desired kind of sign-definiteness. The methods are described in Sections 3, 4, and 6 through 9. In Section 10, we note that our methods can be used to bound the inverse of an interval matrix. Section 11 provides a suggested procedure for deciding how to use our methods. Section 12 discusses some special problems for which our methods are especially suited.

2. Sign-definiteness

We shall need a concept defined as follows:

$$A\widetilde{x} = [\underline{A}\widetilde{x}, \overline{A}\widetilde{x}] \tag{2.1}$$

That is, we are able to express $A\widetilde{x}$ in terms of endpoints of elements of A because we know the signs of the components of x. Equation (1.1) can therefore be written $[\underline{A}\widetilde{x}, \overline{A}\widetilde{x}] = [\underline{b}, \overline{b}]$. Since $\widetilde{x}$ must be such that these intervals intersect. it follows that

$$\underline{A}\widetilde{x} \leq \overline{b} \text{ and } \overline{A}\widetilde{x} \geq \underline{b}. \tag{2.2}$$

To find a component $\underline{h}_i$ (and $\overline{h}_i$) for a given $i = 1, ..., n$, we can minimize (and maximize) $\widetilde{x}_i$ subject to the constraints (2.2). This linear programming problem can be solved by a polynomial amount of computing.

We were able to formulate this problem because, in (2.1), we could express $A\widetilde{x}$ as $[\underline{A}\widetilde{x}, \overline{A}\widetilde{x}]$. In the general case in which x is SD and $\widetilde{x} \in x$, we have

$$a_{ij}\widetilde{x}_j = \begin{cases} [\underline{a}_{ij}\widetilde{x}_j, \overline{a}_{ij}\widetilde{x}_j] \text{ if } x_j \geq 0, \\ [\overline{a}_{ij}\widetilde{x}_j, \underline{a}_{ij}\widetilde{x}_j] \text{ if } x_j < 0. \end{cases} . \tag{2.3}$$

Therefore,

$$A\widetilde{x} = [\widetilde{A}_1\widetilde{x}, \widetilde{A}_2\widetilde{x}] \tag{2.4}$$

where the real (i.e., non-interval) matrices $\widetilde{A}_1$ and $\widetilde{A}_2$ are formed from appropriate endpoints of elements of the interval matrix A.

It is this feature which enables us to formulate three of the methods in this paper. A similar feature enables us to formulate the other three methods. See Section 4. In particular,

see Equation (4.1). In all six of our methods, we determine or produce a vector or subvector which is SD and use this fact to obtain an algorithm for bounding h. We rely on the crude methods to provide essential information about SD.

Each algorithm involves solving a set of linear programming problems. Thus, they require considerably more computing than the procedures in Section 1 for getting crude bounds.

3. First method

Assume we have computed crude bounds x^B on h by a method such as those described in Section 1. If x^B is SD, then h is SD; and we can use the known method of Section 2 to compute h sharply. If x^B lies in just a few orthants, a reasonable procedure is to use the method to obtain the part of h in each of these orthants. The narrowest interval vector containing all such results is h.

If h extends into all 2^n orthants, then this approach entails solving 2^n separate problems each involving $2n$ linear programming problems. This amount of effort is prohibitive even for moderate values of n. This is an example of an exponential amount of computing used to solve the NP-hard problem of determining h.

4. Second method

It is possible to define a linear programming problem in which the primal variables are components of x and the dual variables are elements of the inverse of A. Since our second method concentrates on the inverse, the method can be considered as a kind of dual of the first method. In this sense, the third and fourth methods below are duals as are the fifth and sixth, respectively.

Given an interval matrix A, let P denote the interval matrix which is the hull of the set of inverses of real matrices in A. Then for any real $\widetilde{A} \in A$, there exists $\widetilde{P} \in P$ such that $\widetilde{P}\widetilde{A} = I$. Note that this does not imply that for any $\widetilde{P} \in P$, there exists $\widetilde{A} \in A$ such that $\widetilde{P}\widetilde{A} = I$.

Let p_i^T denote the ith row of P. If p_i is SD, then we know how to express both $\widetilde{p}_i^T A$ and $\widetilde{p}_i^T b$ for any real $\widetilde{p}_i \in p_i$. For example, if $p_i \geq 0$, then

$$\widetilde{p}_i^T A = [\widetilde{p}_i^T \underline{A},\, \widetilde{p}_i^T \overline{A}] \quad \text{and} \quad \widetilde{p}_i^T b = [\widetilde{p}_i^T \underline{b},\, \widetilde{p}_i^T \overline{b}]. \tag{4.1}$$

Note that

$$x_i = \{\widetilde{p}_i^T \widetilde{b} : \widetilde{p}_i^T \widetilde{A} = e_i^T,\ \widetilde{A} \in A,\ \widetilde{b} \in b\}$$

where e_i denotes the ith column of the identity matrix. Therefore, $\underline{h}_i$ is the solution of the linear programming problem

$$\min\ p_i^T \underline{b} \tag{4.2}$$

subject to $p_i^T \underline{A} \leq e_i^T$ and $p_i^T \overline{A} \geq e_i^T$; and $\overline{h}_i$ is the maximum of $p_i^T \overline{b}$ subject to the same constraints.

More generally, if p_i is just SD rather than nonnegative, we know how to express $p_i^T A$ and $p_i^T b$ in a way similar to (4.1). See (2.3). Therefore, we can compute sharp values of $\underline{h}_i$ and $\overline{h}_i$.

If all the elements of P are SD, we could compute all the components of h sharply by solving $2n$ linear programming problems. However, we now show that the results can sometimes be obtained more simply.

If we differentiate the equation $AP = I$ with respect to an element a_{rs} of A, we find that

$$\frac{\partial P_{ij}}{\partial a_{rs}} = -P_{ir}P_{sj} \quad (i, j, r, s = 1, ..., n). \tag{4.3}$$

When P is SD, we know the signs of these derivatives. Therefore, we can determine the real matrix in A whose inverse is $\underline{P}$ and the real matrix in A whose inverse is $\overline{P}$. For example, if $P \geq 0$, then the derivatives given by (4.3) are negative. In this case, $\underline{P} = \overline{A}^{-1}$ and $\overline{P} = \underline{A}^{-1}$.

For certain conditions on b, the results of the optimization problem (4,2) (and the corresponding max) can be expressed simply. For example, if $P \geq 0$, we find

$$h = \begin{cases} [\overline{A}^{-1}\underline{b}, \underline{A}^{-1}\overline{b}] & \text{if } b \geq 0 \\ [\underline{A}^{-1}\underline{b}, \underline{A}^{-1}\overline{b}] & \text{if } 0 \in b \\ [\underline{A}^{-1}\underline{b}, \overline{A}^{-1}\overline{b}] & \text{if } b \leq 0 \end{cases} .$$

This special case is known. See page 108 of [6].

For this example, we are able to compute h as the solution of two noninterval systems. It requires that P be positive and that b be rather special. For our method P need only be SD, and b is arbitrary. However, our method can require much more computing.

To formulate the linear programming problem (4.2) we must know the signs of all the components of row p_i^T of P. Using a method described in Section 1, we can compute crude bounds on the ith row p_i^T by solving $A^T p = e_i$. If the bounds are SD, then the exact row p_i^T is SD. In this case, we can obtain sharp bounds on x_i using the above method.

If desired, we can compute bounds on all the rows of P by solving $A^T P^T = I$ and then compute bounds on all the columns of P by solving $AP = I$. The intersection of the two results will generally be sharper than either result. This enhances the chance of proving that p_i is SD.

Let P^B denote a bound on P obtained by a method such as that just described. Note that we can obtain crude bounds on h by computing $P^B b$; but the bounds need not be sharp even if P^B is sharp. Suppose p_i fails to be SD in only a few of its components. We can divide each of these components into its negative and positive parts and solve for x_i for each combination of cases in which p_i is SD. Then h is the hull of the union of results. Compare the similar statement in Section 3.

Let $(p_i^B)^T$ denote the ith row of P^B. In problem (4.2), the unknown real row vector p_i must be bounded by the interval row vector p_i^B. If the method used to solve the linear programming problem (4.2) can benefit from additional constraints, we can use $p_{ij} \in p_{ij}^B$ $(j = 1, ..., n)$.

5. Preconditioning

We now consider methods of preconditioning which allow a preconditioned equation to be solved exactly in part or in whole. One type of method involves preconditioning a vector so that it is SD. See the methods in Sections 6 and 7. Another type eliminates the variables which are not SD so that the problem can be partially solved. The latter type can be regarded as preconditioning so that certain quantities are zeroed. It involves operations by an interval matrix; so it is perhaps misleading to call it preconditioning. See the methods in Sections 8 and 9.

We now consider the first type of preconditioning. Assume an interval vector $x = [\underline{x}, \overline{x}]$ is not SD. Define a real vector q with components

$$q_i = \begin{cases} -\underline{x}_i & \text{if } \underline{x}_i < 0 < \overline{x}_i \text{ and } |\underline{x}_i| \le \overline{x}_i \\ 0 & \text{if } \underline{x}_i \ge 0 \text{ or } \overline{x}_i \le 0 \\ -\overline{x}_i & \text{if } \underline{x}_i < 0 < \overline{x}_i \text{ and } |\underline{x}_i| > \overline{x}_i \end{cases} \qquad (i = 1, ..., n).$$

Then the interval vector $y = x + q$ is the nearest SD interval vector to x in some sense.

Assume x is not SD. However, assume that at least one component of x is strictly SD. By "strictly SD", we mean that the the interval is SD and neither endpoint is zero. An interval having an endpoint which is zero is SD, but not strictly SD. Let j be the index such that x_j is the component of largest mignitude. The mignitude of x_j is

$$\text{mig}(x_j) = \begin{cases} \underline{x}_j & \text{if } \underline{x}_j > 0 \\ -\overline{x}_j & \text{if } \overline{x}_j < 0 \\ 0 & \text{otherwise} \end{cases} .$$

Define the vector $v(j)$ with components

$$v_i(j) = \begin{cases} q_i/\underline{x}_j & \text{if } x_j > 0 \\ q_i/\overline{x}_j & \text{if } x_j < 0 \end{cases}$$

and define the matrix

$$V_j = I + v(j)e_j^T. \tag{5.1}$$

Then $V_j x$ is SD.
Note that

$$V_j^{-1} = I - \tfrac{1}{2}v(j)e_j^T \tag{5.2}$$

Therefore, V_j^{-1} is known exactly when V_j has been determined.

In what follows, we do not actually use the interval vector (such as x) which we precondition so as to be SD. Instead, we use an unspecified real vector $\widetilde{x} \in x$. However, if we precondition so that the interval vector x is SD, then the sign of any component of $\widetilde{x} \in x$ has the sign we impose on the corresponding component of x.

Suppose we precondition a matrix A by multiplying by a real matrix $\widetilde{B}$. The product $M = \widetilde{B}A$ can be irregular even when A is regular and $\widetilde{B}$ is nonsingular. The four methods we describe below all use some kind of preconditioning and in two cases $\widetilde{B}$ becomes an interval matrix. In each method, we assume that $\widetilde{B}$ is such that M is regular.

A virtue of the preconditioning method just described is that the preconditioning matrix differs from the identity in only one row. Contrast this with preconditioning using the inverse of the center of A. See Section 1.

The matrix V_j^{-1} in (5.2) will be used as a preconditioner. The smaller the norm of the vector $v(j)$ used to define V_j^{-1}, the closer V_j^{-1} is to the identity. The enlargement of the solution set by preconditioning is less when V_j^{-1} is nearer the identity.. If more than one component of x is strictly SD, it is sometimes possible to define a matrix similar to V_j^{-1} which is nearer the identity. We omit the details.

6. Third method

Our third method is obtained by introducing preconditioning into our first method. Suppose we have obtained crude bounds x^B on the solution to $Ax = b$ and find that at least one component of x^B is SD. Then the corresponding component of the hull h is SD. Therefore we can determine a matrix V as in Section 5 such that Vx^B is SD. This assures that Vh is SD.

Define $y = Vx$ and $M = AV^{-1}$. Then the solution y of $My = b$ is SD and its hull can be found using the first method (in Section 3). We then obtain x as $x = V^{=1}y$.

Presumably any kind of preconditioning can enlarge the solution set. It is natural to compare the method using this kind of preconditioning with a method in Section 1 used to get the crude bounds x^B. The latter method requires considerably less computing.

To get x^B, we can precondition by multiplying by an approximate inverse of the center A_c of A. The closer A_c is to the identity, the less the preconditioning step tends to enlarge the solution set. The closer x^B is to being SD, the less the method just described enlarges the solution set. (A measure of how far x^B is from SD is the norm of the vector $v(j)$ in Section 5.) The amount to which preconditioning enlarges the solution set depends on how far the preconditioner is from the identity matrix. Therefore, the comparative sharpness of results when preconditioning by A_c^{-1} or by V^{-1} depends strongly on the nature of the problem. A similar statement holds for the methods discussed below.

7. Fourth method

In the third method, we introduced a preconditioning procedure which produced an equation whose solution was SD. Therefore, we could apply the first method. In the same way, we can precondition so that the new equation can be solved by the second method (in Section 4). That is, the preconditioning is such that a row of the inverse of the generated matrix is SD.

The exact inverse P of A will usually be regular when A is regular. Assume it is. Also assume that the bound P^B (obtained as in Section 4) on P is regular. Then for any $i = 1, ..., n$, at least one component of the ith row $(p_i^B)^T$ of P^B must be SD. From Section 5, we can determine a matrix V such that $(p_i^B)^T V$ is SD. This implies that $p_i^T V$ is SD.

Assume that we have determined V such that row i of PV is SD. To precondition $Ax = b$, we multiply by the matrix V^{-1}. (Note that V^{-1} is exactly known from (5.2) when V is known.) The new coefficient matrix is $M = V^{-1}A$. Row i of the inverse of M is SD.

We can compute the ith component of the hull of $V^{-1}Ax = V^{-1}b$ using the second method (see Section 4).

It is not necessary to verify that row i of the inverse of M is SD. If it were computed to not be SD, it would still be correct to proceed as if it were.

Note that the result x_i will generally not be a sharp bound on the corresponding component h_i of the hull since preconditioning by V^{-1} tends to enlarge the solution set.

There are two other reasons why a solution obtained by this method can fail to be sharp. First, the computed bounds on the inverse will generally not be sharp. The preconditioning matrix V is determined so that a row of the bound P^B is SD. Since P^B is not a sharp bound on P, the matrix V generally causes an "overshoot" when changing a non-SD element to SD. Therefore, preconditioning A by $V^{=1}$ causes too large a change in A.

The other cause of loss of sharpness is more subtle. It occurs because P contains matrices which are not inverses of matrices in A. The loss of sharpness is similar in nature to that just described.

8. Fifth method

Assume that one or more component of the crude bound x^B is SD. Then the corresponding component(s) of the hull h are SD. For simplicity, assume that for some integer k, we have $\underline{h}_i < 0 < \overline{h}_i$ for $i = 1, ..., k$ and h_i is SD for $i = k+1, ..., n$. Partition A, x, and b conformally so that the equation $Ax = b$ takes the form

$$\begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} c \\ d \end{bmatrix}. \tag{8.1}$$

Here $x^T = (y^T, z^T)$ where y has k components and z has $n - k$ components. The hull of the solution set of this system is such that the interval solution z is SD.

Perform interval Gaussian elimination; but stop when A_3 is zeroed The result is an equation of the form

$$\begin{bmatrix} A_1' & A_2' \\ 0 & A_4' \end{bmatrix} \begin{bmatrix} y \\ z \end{bmatrix} = \begin{bmatrix} c' \\ d' \end{bmatrix} \tag{8.2}$$

This equation can be written as the system

$A_1'y + A_2'z = c'$,

$A_4'z = d'$.

Since z is SD, we can compute z sharply from the equation $A_4'z = d'$ using the first method (in Section 3). We can then compute bounds on y by backsolving $A_1'y + A_2'z = c'$.

When we perform the interval Gaussian elimination to obtain (8.2) from (8.1), interval widths will tend to grow; and we should precondition. Suppose we precondition by multiplying $Ax = b$ by an approximate inverse of the center of A. Then there is no point in using the method just described because we can determine the hull of such a preconditioned system sharply using the hull method. See [4].

Instead, we should precondition by an approximate inverse of the center of

$$\begin{bmatrix} A_1 & 0 \\ A_3 & I \end{bmatrix}$$

where I denotes an identity matrix of order $n-k$. This tends to enlarge the solution set by less than preconditioning by an approximate inverse of the center of the entire matrix A.

9. Sixth method

We now consider a method which can be regarded as a kind of dual of the fifth method.

Suppose we compute crude bounds on the inverse P of A as described in Sections 1 and 4. To simplify discussion we fix our attention on the first row of P. We also simplify by assuming that P_{1j} is SD for $j = 1, ..., k$ and that $\underline{P}_{1j} < 0 < \overline{P}_{1j}$ for $j = k+1, ..., n$.

Partition A as

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_4 \end{bmatrix}$$

where A_1 is k by k and A_4 is $n-k$ by $n-k$. We can perform Gaussian elimination on A in such a way that A_2 becomes zero. This is achieved by multiplying by a matrix of the form

$$B = \begin{bmatrix} I & B_2 \\ 0 & B_4 \end{bmatrix}.$$

This matrix need not be explicitly generated. However, the operations to obtain BA must also be perfomed on b so that the new equation is $BAx = Bb$.

The first row of the inverse PB^{-1} of BA is such that its first k components are the same as those of P and, by assumption, are SD. The last $n-k$ components of the first row of PB^{-1} are zero (and hence SD). Since the first row of the inverse of BA is SD, we can determine the first component of the solution of $BAx = Bb$ by the second method (in Section 4).

Other components of x can be bounded in a similar way.

10. Using the inverse

Suppose we have a matrix P' which bounds the exact inverse P of A. In Section 4, we discussed how to obtain P'. Note that $P'b$ bounds the hull h of the solution set. The bound on h would generally not be sharp even if P' were the exact inverse P. This is because P can contain matrices which are not inverses of any matrix in A. However, this provides another bound on h which can be intersected with bounds obtained by methods such as those we have described.

We can bound P more sharply than by the way described in Section 4 by using methods such as ours to solve the equations which P must satisfy. Thus, we can solve for the ith column of P by solving $Ax = e_i$. We can solve for the ith row of P by solving $A^T x = e_i$. If we solve for both the rows and the columns, we can intersect the two results.

The wider the vector b in $Ax = b$ the wider the solution set. From this point of view, the equation $Ax = e_i$ is ideal in that the right hand vector e_i is real and all components

are zero except one. This suggests that there can be advantages in using a method which computes the inverse of A.

11. Choosing a method

In practice, we must decide whether to use any of the methods we have described; and, if so,which one(s). We first note that if the center A_c of A is "near" the identity matrix, then its inverse $B = A_c^{-1}$ is near the identity. In this case, there is little need to use any of our methods. We can use B to precondition the system without unduly enlarging the solution set. We can then use the hull method to determine the hull of the preconditioned system. The question of what is meant by "near the identity matrix" will be left to a user. Alternatively, the center of A might be near a matrix which is the identity with rows and columns permuted.

Suppose the center A_c of A is near the identity matrix. Then the center of A^{-1} is near the identity. That is, its off-diagonal elements are likely to contain zero. Therefore, a row of A^{-1} is unlikely to be SD. In this case, it is unlikely that our second method is applicable; and there is probably little point in using the fourth or sixth method.

For any problem, a reaonable first step is to obtain bounds x^B on the hull h using the hull method. One can also use the method from [3] and find the intersection of the two methods Thereafter, we might use the following procedural steps. They involve a great deal of computing; but the work is polynomially bounded.
(1) If A_c is near the (perhaps permuted) identity matrix, accept the results of the hull method as a sufficiently sharp solution. Thus, go to step (10).
(2) If x^B is SD in all but a few components, solve for h using the first method (in Section 3). Then go to step (10).
(3) Use the hull method to obtain bounds on A^{-1} by solving $Ax = e_i$ for $i = 1, ..., n$. (The method in [3] can also be used.) Also solve $A^T x = e_i$ for $i = 1, ..., n$ to bound A^{T-1}. Then intersect the two bounds on A^{-1}. Denote the resulting bound on A^{-1} by P^B.
(4) For $i = 1, ..., n$, if all but a few components of row i of P^B are SD, solve for h_i using the second method (in Section 4). If all components of h are obtained in this way, go to step (10).
(5) Compute the bound $P^B b$ on h. Intersect it with x^B and the result from step (4).
(6) If at least one component of h has been shown to be SD, the third method (in Section 6) is applicable. Use it to bound h. Intersect the solution with the result of step (5).
(7) Use the fourth method (in Section 7) to bound h_i for $i = 1, ..., n$. Skip any value of i for which h_i was obtained sharply in step (4). Intersect the result with the result from step (6).
(8) If at least one component of h has been found to be SD, use the fifth method (in Section 8) to bound h. Intersect the result with the result from previous steps.
(9) Use the sixth method (in Section 9) to bound h. Intersect the result with the result from previous steps.
(10) Stop.

The amount of work to apply this procedure is not particularly excessive if the crude bounds reveal that h or A^{-1} is SD. If this is not the case, our procedure is useful only if sharpness is so important that a considerable amount of computing is warranted.

There are cases in which there is no need to use our methods. If A is an M-matrix, then interval Gaussian elimination will obtain the hull h sharply. See [5].

12. Some special cases

The essential requirement in our methods is that we are able to express certain products in which a factor is unknown except for its sign. For example, in the first method, we needed to be able to express Ax when x is unknown. If x_j is SD, we can use (2.3) to express $a_{ij}x_j$ in terms of the endpoints of a_{ij}. But suppose that for a given value of j, the element a_{ij} is real (i.e., a degenerate interval) for all $i = 1, ..., n$. Then the "endpoints" of a_{ij} are equal; and $a_{ij}x_j$ is expressed in terms of their coincident value. Therefore, x_j need not be SD.

If all but a few columns of A are real, the first method can be used to determine the hull h with a reasonable amount of computing. If a given column of A is real, the corresponding component of x does not have to be made SD in the third method, nor does it have to be eliminated in the fifth method.

Similar statements can be made for the dual methods. Now, however, we must be able to express both p^TA and p^Tb for a row p^T of P. Consider the matrix $R = (P \quad b)$ which is the matrix P augmented with the vector b as an added column. If all but a few rows of R are real, the second method can determine the hull. If a row (or rows) of R is real, the corresponding component of a row of P need not be SD in the fourth and sixth methods.

Even if every element of A except one is real, then every element of P can be a nondegenerate interval. It is unlikely that we shall know that a row of R is real. Therefore, the dual methods generally do not "simplify" in this way.

13. A non-polynomial method

The methods we have described fail if the crude methods fail to obtain bounds on the solution. In this section, we describe how the hull of the solution set of $Ax = b$ can be obtained as the solution of an optimization problem. The optimization problem is not a linear programming problem; so the work to solve it is not polynomially bounded. We include it as an alternative for two reasons. First, it does not require that some other method provide crude bounds. Second, it is similar to the methods we have described. The difference is that the formulation of the optimization problem includes nonlinear constraints.

Equation (2.3) can be written as

$$a_{ij}x_j = [\min\{\underline{a}_{ij}x_j, \overline{a}_{ij}x_j\}, \max\{\underline{a}_{ij}x_j, \overline{a}_{ij}x_j\}]. \tag{14.1}$$

Since the maximum of two function can be exressed as their average plus half their difference, we have

$$\max\{\underline{a}_{ij}x_j, \overline{a}_{ij}x_j\} = \tfrac{1}{2}(\underline{a}_{ij}x_j + \overline{a}_{ij}x_j) + \tfrac{1}{2}|\overline{a}_{ij}x_j - \underline{a}_{ij}x_j| = m_{ij}x_j + \tfrac{1}{2}w_{ij}|x_j| \tag{14.2}$$

where $m_{ij} = \frac{1}{2}(\underline{a}_{ij} + \overline{a}_{ij})$ and $w_{ij} = \overline{a}_{ij} - \underline{a}_{ij}$. Similarly,

$$\min\{\underline{a}_{ij}x_j, \overline{a}_{ij}x_j\} = m_{ij}x_j - \tfrac{1}{2}w_{ij}|x_j|. \tag{14.3}$$

Row i of the equation $Ax = b$ can be written

$\sum_{j=1}^{n} a_{ij}x_j = b_i \quad (i = 1, ..., n).$

From (14.1), (14.2), and (14.3), we therefore obtain

$\sum_{j=1}^{n} [m_{ij}x_j - \frac{1}{2}w_{ij}|x_j|,\ m_{ij}x_j + \frac{1}{2}w_{ij}|x_j|] = b_i.$

A point x is in the solution set of $Ax = b$ only if the intervals in the left and right members intersect. This imposes the constraints

$$\sum_{j=1}^{n} (m_{ij}x_j - \tfrac{1}{2}w_{ij}|x_j|) \le \overline{b}_i, \tag{14.4a}$$

and

$$\sum_{j=1}^{n} (m_{ij}x_j + \tfrac{1}{2}w_{ij}|x_j|) \ge \underline{b}_i. \tag{14.4b}$$

for $i = 1, ..., n$.

We can obtain the kth component of the hull by minimizing and maximizing x_k subject to the constraints (14.4). The function $|x_j|$ is not differentiable at $x_j = 0$. We can obtain constraints which are differentiable if we replace $|x_j|$ by x_{j+n} and add the constraints

$x_{j+n}^2 - x_j^2 = 0 \quad (j = 1, ..., n).$

The problem now becomes: For $k = 1, ..., n$,

minimize and maximize x_k

subject to

$\sum_{j=1}^{n} (m_{ij}x_j - \frac{1}{2}w_{ij}x_{j+n}) \le \overline{b}_i \quad (i = 1, ..., n)$

$\sum_{j=1}^{n} (m_{ij}x_j + \frac{1}{2}w_{ij}x_{j+n}) \ge \underline{b}_i \quad (i = 1, ..., n)$

$x_{j+n}^2 - x_j^2 = 0 \quad (j = 1, ..., n).$

14. References

[1] Hansen, E. R., Interval arithmetic in matrix computations, part I, SIAM J. Numer. Anal. 2, 308-320, 1965.

[2] Hansen, E. R., Global Optimization Using Interval Analysis, Marcel Dekker, 1992.

[3] Hansen, E. R., Sharpening interval computations, paper presented at the First Scandinavian workshop on interval methods and their application, Copenhagen, 2003.

[4] Hansen, E. R. and Walster, G. W. Global Optimization Using Interval Analysis, (second ed.), Marcel Dekker, 2004.
[5] Heindl, G., Kreinovich, V., and Lakeyev, A. (1998), Solving linear interval systems is NP-hard even if we exclude overflow and underflow, Reliable Computing, 4, 383-388.
[6] Neumaier, A., Interval Methods for Systems of Equations, Cambridge University Press, London, 1990.