Rigorous filtering using linear relaxations

Ferenc Domes, Arnold Neumaier
Faculty of Mathematics, University of Vienna
Nordbergstrasse 15, A-1090 Vienna, Austria

August 12, 2010

Abstract. This paper presents rigorous filtering methods for continuous constraint satisfaction problems based on linear relaxations. Filtering or pruning stands for reducing the search space of constraint satisfaction problems. Discussed are old and new approaches for rigorously enclosing the solution set of linear systems of inequalities, as well as different methods for computing linear relaxations. This allows custom combinations of relaxation and filtering. Care is taken to ensure that all methods correctly account for rounding errors in the computations.
Although most of the results apply more generally, strong emphasis is given to relaxing and filtering quadratic constraints, as implemented in the GloptLab environment, which internally exploits a quadratic structure. Demonstrative examples and tests comparing the different linear relaxation methods are also presented.
Keywords. linear relaxations, filtering, pruning, continuous constraints, quadratic constraint satisfaction problems, rounding error control, verified computation, quadratic programming, branch and bound, global optimization.

1 Introduction

1.1 Context

This paper considers rigorous filtering methods based on computing linear relaxations for continuous constraint satisfaction problems. A constraint satisfaction problem is the task of finding one or all points satisfying a given family of equations and/or inequalities, called constraints. Many real word problems are continuous constraint satisfaction problems, often high-dimensional ones. Applications include robotics (Grandon et al. [11], Merlet [20]), localization and map building (Jaulin [13], Jaulin et al. [14], biomedicine Cruz & Barahona [6]), and the protein folding problem (Krippahl & Barahona [16]). In practice, constraint satisfaction problems are solved by a combination of a variety of techniques, often involving constraint propagation with either some form of stochastic search or a branch and bound scheme for a complete search. These techniques are mainly complemented by filtering or pruning techniques based on techniques borrowed from optimization, such as linear or convex relaxations (see, e.g., Neumaier [23]).
Filtering or pruning stands for reducing the search space of constraint satisfaction problems. There are many filtering techniques which are usually combined with branch and bound methods and provide more or less reduction of the search space. If applied to quadratic constraints, the classical filtering algorithms based upon local consistencies like 2B-consistency or Box-consistency (see, e.g., Benhamou et al. [4]) do not take advantage of the special properties of quadratic forms and therefore often results are poorer than desirable. 3B-consistency is more effective, but the practice shows that for quadratic problems they usually tend to be slow due to the exhaustive branching needed to achieve the required precision. Hull consistency techniques like the classical HC4 (Benhamou et al. [3]) or the newly developed OCTUM (Chabert & Jaulin [5]) show promising results, but still do not use the special structure of quadratic problems.

Another approach — originally developed for global optimization — is to compute linear relaxations for a problem, and then use these to reduce the search space of the original problem. As the name suggests, by computing a relaxation, we may also lose some structural information of the original problem. However this loss is often complementary to that in the consistency techniques described before. Here only linear relaxations are considered, higher degree relaxations and convex relaxations are discussed in the literature; for example, affine and convex functions for non-convex multivariate polynomials in GARLOFF et al. [10]. Constructing relaxations by using the right method can effectively approximate the structure of the original constraint; in fact, the criteria for a good relaxation is having a small distance (in some vague sense) to the original constraints. The resulting linear system usually contains more variables/constraints than the original problem but is much easier to solve. A classical method, called RLT (reformulation-linearization technique) by SHERALI & ADAMS [29] is used by LEBBAH et al. [18] in the QUAD algorithm; another interesting approach was given by KOLEV [15]. Since the last two fit the main scope of this paper they will be discussed in detail. These methods require the rigorous solution of linear programs, as for example given for programs with uncertain data JANSSON & RUMP [12].

1.2 Software

A number of software packages for solving constraint satisfaction problems make extensive use of linear relaxations. The ICOS solver by LEBBAH [17] is a free software package for solving nonlinear and continuous constraints, based on constraint programming, relaxation and interval analysis techniques. The prize winning, commercial solver BARON by SAHINIDIS & TAWARMALANI [26] — a highly developed, approximate global optimization solver — uses a special linear relaxation technique called the sandwich method, while the COCONUT Environment [27, 28] applies both linear relaxations using slopes and reformulation-linearization on Directed Acyclic Graphs (DAGs).
Note that solving constrained global optimization problems by branch and bound is in practice reduced to solving a sequence of constraint satisfaction problems, each obtained by adding a constraint $f(x) \leq f_{\text{best}}$ to the original constraints, where f is the objective function and f_{best} the function value of the best feasible point found so far. Thus all techniques for solving constraint satisfaction problems have immediate impact on global optimization. This widens the scope of the possible applications of the methods presented.

1.3 Outline

The paper is organized as follows. In Sections 2–5 rigorous techniques for enclosing the solution set of linear systems of inequalities are discussed, while Sections 6–7 are about creating linear relaxations for quadratic constraint satisfaction problems. In detail, Section 2 considers finding the interval hull of a bounded polyhedron by means of solving a single linear optimization problem. In Section 3 the concept of a Gauss-Jordan preconditioner is introduced. In Section 4 it is shown how the preconditioner can be used to find cheap bounds for a linear system of inequalities. In Section 5 a more costly approach is given. Here for the most promising directions two linear programs are solved approximately and the approximate solutions are verified. Section 6 gives step-by-step instructions how linear relaxations for multivariate quadratic expressions can be generated, how the method of LEBBAH et al. [18] and KOLEV [15] can be unified, and how new relaxation techniques for bilinear terms can be computed. In Section 7, linear relaxations and filtering for constraint satisfaction problems are considered, by discussing how the linear methods can be combined in order to effectively reduce the search space of a quadratic constraint satisfaction problem. The integration of the methods in GloptLab (see Domes [7]) environment is explained in detail. In Section 8 two demonstrative examples are given while Section 9 presents some test results and comparison of different relaxation techniques.

1.4 Notation

$\mathbf{a} = [\underline{a}, \overline{a}]$ with $\underline{a} \leq \overline{a}$ denotes a real interval with a possibly infinite lower bound $\underline{a}$ and a possibly infinite upper bound $\overline{a}$. A bound is **large**, if its absolute value is greater than a configurable constant μ (whose default value is 10^6 in GloptLab). Decisions are often based on "if a bound is large" rather than on "if a bound is infinite". An interval is **large** if both of its bounds are large. The expressions

$$\operatorname{wid}(\mathbf{a}) := \overline{a} - \underline{a}$$

denotes the **width**,

$$\operatorname{mid}(\mathbf{a}) := (\underline{a} + \overline{a})/2$$

denotes the **midpoint**,

$$\langle \mathbf{a} \rangle := \begin{cases} \min(|\underline{a}|, |\overline{a}|) & \text{if } 0 \notin [\underline{a}, \overline{a}], \\ 0 & \text{otherwise,} \end{cases}$$

denotes the **mignitude** and

$$|\,\mathbf{a}| := \max(|\underline{a}|, |\overline{a}|)$$

denotes the **magnitude** of an interval $\mathbf{a}$. An interval is called **thin** or **degenerate** if its width is zero. An interval is called **narrow** if its width is less than a configurable constant η (whose default value is 10^{-6} in GloptLab). Decisions are often based on "if an interval is narrow" rather than on "if an interval is thin". The **sign of the interval a** is defined by

$$\operatorname{sign} \mathbf{a} = \begin{cases} 0 & \text{if } \underline{a} = \overline{a} = 0, \\ 1 & \text{if } \underline{a} > 0, \\ -1 & \text{if } \overline{a} < 0, \\ [-1, 1] & \text{if } \underline{a} < 0 < \overline{a}. \end{cases}$$

Note that this is not an interval extension of a real sign function.
An **interval vector** $\mathbf{x} = [\underline{x}, \overline{x}] \in \overline{\mathbb{IR}}^n$ or a **box** is the Cartesian product of the closed real intervals $\mathbf{x}_i := [\underline{x}_i, \overline{x}_i]$, representing a (bounded or unbounded) axiparallel box in $\mathbb{R}^n$. The values $-\infty$ and ∞ are allowed as lower and upper bounds, respectively, to take care of one-sided bounds on variables. $\overline{\mathbb{IR}}^n$ denotes the set of all n-dimensional boxes. A box is large or narrow when all its components are large or narrow. Operations defined for intervals (like width, midpoint, mignitude and magnitude) are interpreted component-wise when applied to boxes. The condition $x \in \mathbf{x}$ is equivalent to the collection of simple bounds

$$\underline{x}_i \leq x_i \leq \overline{x}_i \quad (i = 1, \ldots, n),$$

or, with inequalities on vectors and matrices interpreted component-wise, to the two-sided vector inequality $\underline{x} \leq x \leq \overline{x}$. Apart from two-sided constraints, this includes with $\mathbf{x}_i = [a, a]$ variables x_i fixed at a particular value $x_i = a$, with $\mathbf{x}_i = [a, \infty]$ lower bounds $x_i \geq a$, with $\mathbf{x}_i = [-\infty, a]$ upper bounds $x_i \leq a$, and with $\mathbf{x}_i = [-\infty, \infty]$ free variables.

The n-dimensional identity matrix is denoted by I_n and the n-dimensional zero matrix is denoted by 0_n. The jth row vector of a matrix A is denoted by $A_{j:}$, the kth column vector by $A_{:k}$ and the number of nonzero entries by $\mathrm{nnz}(A)$. The set $\neg N$ denotes the complement of a set N. The number of elements of a set N is denoted by $|N|$. Let $I \subseteq \{1, \ldots, n\}$ and $J \subseteq \{1, \ldots, m\}$ be index sets and let $n_I := |I|$, $n_J := |J|$. Let x be an n-dimensional vector, then x_I denotes the n_I-dimensional vector built from the components of x selected by the index set I. Let A be an $m \times n$ matrix, then $A_{:I}$ denotes the $m \times n_I$ matrix built from the rows of A selected by the index sets I. Similarly, $A_{J:}$ denotes the $n_J \times n$ matrix built from the columns of A selected by the index sets J. $(A^T)^{-1}$ is denoted by A^{-T}. For vectors and matrices the comparison operators $=$, $\neq$, $<$, $>$, $\leq$, $\geq$ and the absolute value $|A|$ of a matrix A are interpreted component-wise.
We also consider a quadratic expression $p(x)$ in $x = (x_1, \ldots, x_n)^T$ such that the evaluation at any $x \in \mathbf{x}$ is a real number. The box $p(\mathbf{x})$ is called an **interval enclosure** of $p(x)$ in the box $\mathbf{x}$ if $p(x) \in p(\mathbf{x})$ holds for all $x \in \mathbf{x}$. There are a number of methods for defining $p(\mathbf{x})$, for example interval evaluation or centered forms (for details, see, e.g., Neumaier [22]). If for all $y \in p(\mathbf{x})$ an $x \in \mathbf{x}$ exists such that $p(x) = y$, then $p(\mathbf{x})$ is called the **range**. If this only holds for $y = \inf p(\mathbf{x})$ and $y = \sup p(\mathbf{x})$, then $p(\mathbf{x})$ is the **interval hull** $\square\{p(x) \mid x \in \mathbf{x}\}$. Another – and somewhat trickier – alternative is is to compute the upper and the lower bound of the range separately, without the use of interval arithmetic, by using monotonicity properties of the operations. To get rigorous results when using floating point arithmetic, one needs here directed rounding. However, not all expressions can be bounded from below or above using directed rounding only; and detailed considerations are needed in each particular case. For an expression p, $\nabla\{p\}$ denotes the result obtained when first the rounding mode is set to downward rounding, then p is evaluated, and by $\Delta\{p\}$ the result obtained when first the rounding mode is set to upward rounding, then p is evaluated. Assumed is that negating an expression is done without error; thus, e.g., $\Delta\{-(x-y)\} = -\Delta\{x-y\}$ holds. Careful arrangement allows in many cases to replace downward rounded expressions by equivalent upward rounded expressions. For example, $\nabla\{x-y\} = \Delta\{-(y-x)\}$. If this is possible, one can achieve correct results using only upward rounding (thus saving rounding mode switches), while in Intlab's interval arithmetic (see Rump [25]), the rounding mode is switched often, slowing down the computations.
During the first four sections of this paper linear systems are discussed. Linear systems of two-sided inequalities are given by

$$Ex \in \mathbf{b},\ x \in \mathbf{x}, \tag{1}$$

where E is an $m \times n$ real matrix, $\mathbf{b} := [\underline{b}, \overline{b}]$ is an m–dimensional box and $\mathbf{x} := [\underline{x}, \overline{x}]$ is an n–dimensional box. The linear expressions comprise component-wise linear enclosures $E_{i:}x \in \mathbf{b}_i$. This includes equality constraints if $\mathbf{b}_i$ is a thin interval with $\underline{b}_i = \overline{b}_i$, inequality constraints if one bound of $\mathbf{b}_i$ is infinite, and two-sided inequalities if both bounds are finite. Similarly, the n bounds on the variables are interpreted as enclosures $x_j \in \mathbf{x}_j$ with $j = 1, \ldots, n$. Again, fixed variables and one-sided bounds on the variables are included as special cases.

2 Bounding a polyhedron

Geometrically the linear system (1) defines a polyhedron. If the polyhedron is bounded and nonempty, the method presented in this section finds a finite enclosure of the polyhedron, by solving a single linear program. By using this method only *large bounds* are reduced; for a fixed, large constant μ (in our implementation $\mu = 10^6$) and a given interval $\mathbf{a}$ the lower bound $\underline{a}$ is large iff $\underline{a} \le -\mu$ and the upper bound $\overline{a}$ is large iff $\overline{a} \ge \mu$. Choosing μ too small may result in improvement of small bounds but it is not recommended since the gain is not significant enough compared to other methods presented in this paper. In oder to avoid numerical problems, in this section we also assume that we have found a suitable scaling vector $\omega \in \mathbb{R}^m$ for the constraints and a suitable scaling vector $\rho \in \mathbb{R}^n$ for the variables (for finding these we refer to Domes & Neumaier [8]).

2.1 Completing one-sided bound constraints

Let

$$Ex \in \mathbf{b},\ x \in \mathbf{x},$$

be a linear system as given in (1). We partition the components of the box $\mathbf{b}$ in only lower bounded (B_-), only upper bounded (B_+) and bounded (B_f) ones. We also partition the components of the box $\mathbf{x}$ in unbounded (X_∞), only lower bounded (X_-), only upper bounded (X_+) and bounded (X_f) ones. According to this for $i \in \{1,\dots,m\}$ and $j \in \{1,\dots,n\}$, we define the index sets

$$\begin{array}{ll} B_- := \{i \mid \underline{b}_i > -\omega_i\mu,\ \overline{b}_i \ge \omega_i\mu\}, & X_+ := \{j \mid \underline{x}_j \le -\rho_j\mu,\ \overline{x}_j < \rho_j\mu\}, \\ B_+ := \{i \mid \underline{b}_i \le -\omega_i\mu,\ \overline{b}_i < \omega_i\mu\}, & X_f := \{j \mid \underline{x}_j > -\rho_j\mu,\ \overline{x}_j < \rho_j\mu\}, \\ B_f := \{i \mid \underline{b}_i > -\omega_i\mu,\ \overline{b}_i < \omega_i\mu\}, & X_\pm := X_+ \cup X_-, \\ X_\infty := \{j \mid \underline{x}_j \le -\rho_j\mu,\ \overline{x}_j \ge \rho_j\mu\}, & X_b := X_\pm \cup X_f, \\ X_- := \{j \mid \underline{x}_j > -\rho_j\mu,\ \overline{x}_j \ge \rho_j\mu\}, & X_u := X_\pm \cup X_\infty. \end{array} \tag{2}$$

Multiplying (1) by a vector $y \in \mathbb{R}^m$ (choosen later) leads to the enclosure

$$y^T Ex \in y^T \mathbf{b}.$$

Bringing the terms containing the variables with index in X_f and X_∞ to the right hand side, substituting their bounds and evaluating the results by using interval arithmetic, leads to

$$(y^T E_{:X_\pm})x_{X_\pm} \in \mathbf{d} := y^T\mathbf{b} - (y^T E_{:X_f})\mathbf{x}_{X_f} - (y^T E_{:X_\infty})\mathbf{x}_{X_\infty}. \tag{3}$$

Therefore if

$$(E^T y)_{X_\infty} = 0. \tag{4}$$

then by (3) follows that

$$(y^T E_{:X_\pm})x_{X_\pm} \ge \underline{d} = \inf(y_{B_-}^T \mathbf{b}_{B_-}) + \inf(y_{B_+}^T \mathbf{b}_{B_+}) + \inf(y_{B_f}^T \mathbf{b}_{B_f}) - \sup(y^T E_{:X_f}\mathbf{x}_{X_f}). \tag{5}$$

The following proposition shows which conditions must hold such that (5) yields finite bounds on the half–bounded variables:

Proposition. 2.1 *If we can find an y such that the conditions*

$$\begin{array}{ll} y_i > 0 & \text{if } i \in B_-, \\ y_i < 0 & \text{if } i \in B_+, \\ (E^T y)_j = 0 & \text{if } j \in X_\infty, \\ (E^T y)_j < 0 & \text{if } j \in X_-, \\ (E^T y)_j > 0 & \text{if } j \in X_+, \end{array} \tag{6}$$

hold, then for each $k \in X_-$

$$x_k \leq c_k := (\underline{d} - y^T E_{:X_-^k} \underline{x}_{X_-^k} - y^T E_{:X_+} \overline{x}_{X_+})/(y^T E_{:,k}) \tag{7}$$

is satisfied, and for each $k \in X_+$

$$x_k \geq c_k := (\underline{d} - y^T E_{:X_-} \underline{x}_{X_-} - y^T E_{:X_+^k} \overline{x}_{X_+^k})/(y^T E_{:,k}) \tag{8}$$

holds. The bounds c_k *are finite.*

Proof. Since $y_i > 0$ for all $i \in B_-$ and $y_i < 0$ for all $i \in B_+$ by definition of B_- and B_+ the terms $y_i^T \mathbf{b}_i$ have finite lower bounds for all $i \in B_\pm$. Since $(E^T y)_{X_\infty} = 0$ and $\mathbf{x}_{X_f}$ is bounded by definition the inequality (5) holds and $\underline{d}$ is finite. By definition the bounds $\underline{x}_{X_-}$ are finite and by (6) the terms $(E^T y)_j = y^T E_{:j} < 0$ for all $j \in X_-$, therefore we have finite approximation

$$y^T E_{:X_-} x_{X_-} \leq y^T E_{:X_-} \underline{x}_{X_-}. \tag{9}$$

Similarly, the bounds $\overline{x}_{X_+}$ are finite and $(E^T y)_j = y^T E_{:,j} > 0$ for all $j \in X_+$, therefore we have finite approximation

$$y^T E_{:X_+} x_{X_+} \leq y^T E_{:X_+} \overline{x}_{X_+}. \tag{10}$$

For any $k \in X_-$ and $X_-^k := X_- \setminus \{k\}$ since by (6) the inequality $y^T E_{:,k} x_k < 0$ holds, considering (5) and (10) we have

$$\begin{aligned} & y^T E_{:,k} x_k + y^T E_{:X_-^k} x_{X_-^k} + y^T E_{:X_+} x_{X_+} \geq \underline{d} \\ & \quad \Rightarrow \quad y^T E_{:,k} x_k + y^T E_{:X_-^k} \underline{x}_{X_-^k} + y^T E_{:X_+} \overline{x}_{X_+} \geq \underline{d} \\ & \quad \Rightarrow \quad y^T E_{:,k} x_k \geq \underline{d} - y^T E_{:X_-^k} \underline{x}_{X_-^k} - y^T E_{:X_+} \overline{x}_{X_+}, \end{aligned}$$

which by (6) implies (7) with a finite bound c_k. Therefore $\mathbf{x}'_k = [\underline{x}_k, c_k]$ is finite. By similar considerations for any $k \in X_+$ and $X_+^k := X_+ \setminus \{k\}$ since $y^T E_{:,k} x_k > 0$ we have (8) and $\mathbf{x}'_k = [c_k, \overline{x}_k]$ is finite. □

Now we have the necessary conditions on y which allow to find bounds on $x_{X_\pm}$. We note that (7) and (8) are automatically obtained by standard constraint propagation on (3) resulting in finite bounds for the half–bounded variables and improving the bounds which are already finite. If the polyhedron has a finite hull, the constraint propagation succeeds. The constraint propagation method quadratic (and linear) constraints introduced by Domes & Neumaier [9] can be used for this task.
To find tight bounds on $x_{X_\pm}$ the entries of y should not be larger than necessary. This is achieved by solving the linear program with the objective

$$\text{minimize} \sum_{i \in B_-} \omega_i y_i - \sum_{i \in B_+} \omega_i y_i, \tag{11}$$

where ω is the constraint scaling vector, subject to the constraints given by (6). Solving the linear program we either obtain a solution $y \in \mathbb{R}^m$, or the linear program is infeasible. In the latter case and the polyhedron is empty or unbounded:

Proposition. 2.2 *Suppose that* $\mu = \infty$ *in* (2). *If the constraints* (6) *are inconsistent then the polyhedron defined by* (1) *is empty or unbounded.*

Proof. Let x^0 be a point satisfying (1). If no such x^0 can be found the polyhedron is empty. If this is not the case then that x^0 satisfies (1) is equivalent to

$$\begin{array}{ll} (Ex^0)_{B_-} \geq \underline{b}_{B_-}, & x^0_{X_-} \geq \underline{x}_{X_-}, \\ (Ex^0)_{B_+} \leq \overline{b}_{B_+}, & x^0_{X_+} \leq \overline{x}_{X_+}, \\ (Ex^0)_{B_f} \in \mathbf{b}_{B_f}, & x^0_{X_f} \in \mathbf{x}_{X_f}. \end{array}$$

Therefore if a $z \in \mathbb{R}^n$ with $z \neq 0$ satisfies

$$\begin{array}{ll} (Ez)_{B_-} \geq 0, & z_{X_-} \geq 0, \\ (Ez)_{B_+} \leq 0, & z_{X_+} \leq 0, \\ (Ez)_{B_f} = 0, & z_{X_f} = 0, \end{array} \tag{12}$$

then

$$\begin{array}{ll} E(x^0+\lambda z)_{B_-} = (Ex^0)_{B_-} + \lambda(Ez)_{B_-} \geq \underline{b}_{B_-}, & (x^0+\lambda z)_{X_-} = x^0_{X_-} + \lambda z_{X_-} \geq \underline{x}_{X_-}, \\ E(x^0+\lambda z)_{B_+} = (Ex^0)_{B_+} + \lambda(Ez)_{B_+} \leq \overline{b}_{B_+}, & (x^0+\lambda z)_{X_+} = x^0_{X_+} + \lambda z_{X_+} \leq \overline{x}_{X_+}, \\ E(x^0+\lambda z)_{B_f} = (Ex^0)_{B_f} \in \mathbf{b}_{B_f}, & (x^0+\lambda z)_{X_f} = x^0_{X_f} \in \mathbf{x}_{X_+}, \end{array}$$

and thus all $x \in L := \{x^0 + \lambda z \mid \lambda \geq 0\}$ satisfy (1). Since the set L describes a line segment of infinite length and L is contained in the polyhedron defined by (1) the polyhedron must be unbounded.
For any $y \in \mathbb{R}^m$ satisfying (6) and $z \in \mathbb{R}^n$, $z \neq 0$ satisfying (12)

$$\begin{aligned} 0 \leq & \sum_{i \in B+} y_i(Ez)_i + \sum_{i \in B-} y_i(Ez)_i + \sum_{i \in B_f} y_i(Ez)_i \\ = & \sum_{j \in X\infty} (E^Ty)_j z_j + \sum_{j \in X+} (E^Ty)_j z_j + \sum_{j \in X-} (E^Ty)_j z_j + \sum_{j \in Xf} (E^Ty)_j z_j < 0. \end{aligned} \tag{13}$$

Therefore (6) and (12) cannot be solved simultaneously. The Motzkin's transposition theorem (see Motzkin [21]) implies that exactly one of (6) and (12) is satisfied. Therefore if the constraints (6) are inconsistent then (12) holds and the polyhedron defined by (1) is unbounded. □

In reality we only have an approximate solution $\tilde{y}$ of (11) which usually does not need to satisfy (4). Therefore using a matrix C (chosen in the next subsection) and a vector z (chosen below) we construct the corrected solution

$$y = \tilde{y} - C^T z, \tag{14}$$

such that (4) it satisfied. If we substitute (14) into (4) we see that y satisfies (4) if $\tilde{y}E_{:X_\infty} - z^T CE_{:X_\infty} = 0$. Thus we choose z such that

$$(CE_{:X_\infty})^T z = E^T_{:X_\infty}\tilde{y},$$

holds and therefore

$$y = \tilde{y} - C^T((CE_{:X_\infty})^{-T}(E_{:X_\infty}\tilde{y})) \tag{15}$$

satisfies (4). In floating point arithmetic we have to take the rounding errors into account therefore we evaluate (15) using interval arithmetic and obtain a box $\mathbf{y}$ such that for an $y \in \mathbf{y}$ equality (4) is satisfied.

2.2 Bounding free variables

From this point on we assume that all one-sided unbounded constraints are bounded by the method presented in the previous subsection and thus the set $X_\pm$ is empty. Therefore we use (3) with $\mathbf{y}$ instead of y and choose C such that we find finite bounds on the free variables x_{X_∞}:

Proposition. 2.3 *Let C be a preconditioner for $E_{:X_\infty}$ such that $CE_{:X_\infty} \approx I$. Suppose y satisfies*

$$\begin{array}{ll} y_i > 0 & \text{if } i \in B_-, \\ y_i < 0 & \text{if } i \in B_+, \\ (E^T y)_j = 0 & \text{if } j \in X_\infty. \end{array} \tag{16}$$

Let $u^+, u^- \in \mathbb{R}^m$ be vectors with

$$u_j^+ \leq \min\{-C_{ij}/y_j \mid i \in X_\infty\}, \quad u_j^- \geq \max\{-C_{ij}/y_j \mid i \in X_\infty\}, \tag{17}$$

and

$$C^+ := C + u^+ y^T, \quad C^- := C + u^- y^T, \tag{18}$$

then

$$CE_{X_\infty:} x_{X_\infty} \in \mathbf{z} \tag{19}$$

for a bounded box

$$\mathbf{z} := [\inf(C^-\mathbf{b} - (C^- E_{:X_b})\mathbf{x}_{X_b}), \ \sup(C^+\mathbf{b} - (C^+ E_{:X_b})\mathbf{x}_{X_b})]. \tag{20}$$

Proof. By (18) and (3), the equation

$$CE_{:X_\infty} x_{X_\infty} = (C^\pm - u^\pm y^T) E_{:X_\infty} x_{X_\infty} = C^\pm E_{:X_\infty} x_{X_\infty} - u^\pm y^T E_{:X_\infty} x_{X_\infty} = C^\pm E_{:X_\infty} x_{X_\infty}$$

holds. On the other hand, (1) implies

$$E_{:X_\infty} x_{X_\infty} + E_{:X_b} x_{X_b} \in \mathbf{b},$$

so that

$$\begin{array}{l} CE_{:X_\infty} x_{X_\infty} = C^- E_{:X_\infty} x_{X_\infty} \geq \inf(C^-(\mathbf{b} - E_{:X_b}\mathbf{x}_{X_b})), \\ CE_{:X_\infty} x_{X_\infty} = C^+ E_{:X_\infty} x_{X_\infty} \leq \sup(C^+(\mathbf{b} - E_{:X_b}\mathbf{x}_{X_b})), \end{array}$$

proving (19).
Since $\mathbf{x}_{X_b}$ is bounded, $E_{:X_b}\mathbf{x}_{X_b}$ is also bounded, we have to show that $\sup(C^+\mathbf{b})_j < \infty$ and $\inf(C^-\mathbf{b})_j > -\infty$ for all $j \in \{1, \dots, n\}$. By (17) we have

$$u_j^+ \leq (-C_{ij}/y_j) \text{ for all } i \in X_\infty. \tag{21}$$

If $y_i < 0$ then $\overline{b}_i$ is finite after the construction of y. Using this together with (18) implies that $u_j^+ y_i \geq -C_{ij}$ and with (21) results in

$$C_{ij}^+ = C_{ij} + u_j^+ y_i \geq 0.$$

Therefore

$$\sup(C_{ij}^+ \mathbf{b}_i) = C_{ij}^+ \overline{b}_i < \mu. \tag{22}$$

On the other hand, if $y_i > 0$ then $\underline{b}_i > -\mu$ implies that $C_{ij}^+ = C_{ij} + u_j^+ y_i \leq 0$, again ending up in

$$\sup(C_{ij}^+ \mathbf{b}_i) = C_{ij}^+ \underline{b}_i < \mu. \tag{23}$$

Since

$$\sup(C^+\mathbf{b})_j = \sum_{i=1}^{m} \sup(C^+_{ij}\mathbf{b}_i),$$

for all $j \in \{1, \dots, n\}$, by inequalities (22) and (23) we obtain

$$\sup(C^+\mathbf{b})_j < \mu.$$

The proof for the lower bounds is similar, with (17) and (18) implying that

$$\inf(C^-_{ij}\mathbf{b}_k) = C^+_{ij}\underline{b}_i > -\mu \tag{24}$$

if $y_i > 0$ or

$$\inf(C^-_{ij}\mathbf{b}_k) = C^+_{ij}\overline{b}_i > -\mu \tag{25}$$

if $y_i < 0$, proving that $\inf(C^-\mathbf{b})_j$ is finite for all j. □

By the above proposition

$$x_{X_\infty} \in (CE_{:X_\infty})^{-1}\mathbf{z} \tag{26}$$

gives finite bounds on the free variables x_{X_∞}.

2.3 Bounding a polyhedron

Summarizing the results of both subsections we are ready to give the following algorithm for bounding a polyhedron:

Algorithm: 2.4 (Bounding a polyhedron)
Purpose: Obtain rigorous finite bounds $\mathbf{x}$ on the variables and improve the bounds $\mathbf{b}$ of the linear program (1)*.*

1. *In* (11) *we replace the sharp inequalities by non-sharp ones: $y_i > 0$ and $y_i < 0$ is replaced by $y_i \geq \omega^{-1}$ and $y_i \leq -\omega^{-1}$ respectively, where $\omega \in \mathbb{R}^m$ is the scaling vector for the constraints. Similarly $(E^Ty)_j > 0$ and $(E^Ty)_j < 0$ is replaced by $(E^Ty)_j \geq \rho^{-1}$ and $(E^Ty)_j \leq -\rho^{-1}$ respectively, where $\rho \in \mathbb{R}^n$ the scaling vector for the variables.*

2. *We solve the linear program* (11) *by using an approximate linear solver:*

   (a) *If the linear program is feasible, we obtain the approximate solution $\tilde{y}$ and compute $\mathbf{y}$ according to* (15) *using interval arithmetic.*

   (b) *If the linear program is infeasible, the polyhedron is empty or unbounded. The algorithm ends.*

3. *Using constraint propagation on* (3) *we obtain finite bounds $\mathbf{x}'_{X_\pm}$ on the half-bounded variables $x_{X_\pm}$.*

4. *We compute u^+, u^-, C^+, C^- and $\mathbf{z}$ as defined in Proposition 2.3. Since x_{X_b} is already bounded the proposition holds and evaluating* (26) *yields finite bounds $\mathbf{x}'_{X_\infty}$ on x_{X_∞}.*

5. *Substituting the new components $\mathbf{x}'_{X_\pm}$ and $\mathbf{x}'_{X_\infty}$ for the corresponding components of the bound constraints in* (1) *and applying constraint propagation to* (1) *results in the new bounds $\mathbf{x}$ on the variables.*

6. *We compute the new bounds $\mathbf{b}' := \mathbf{b} \cap E\mathbf{x}$ for the constraints.*

3 Gauss-Jordan preconditioning

In this section we discuss an extension of Gauss-Jordan elimination, used for preconditioning interval linear systems of equations. We first discuss the original method then modify it to suit our applications.

Discussion of the original method. The Gauss-Jordan inversion is similar to Gaussian elimination but computes the inverse of a matrix. The iterative algorithm starts with an $m \times n$ matrix B (with $n \geq m$) and transforms the leading $m \times m$ sub-matrix of B to an identity matrix. The transformation is done by permuting rows and columns, multiplying whole rows with constants, and subtracting multiplies of a row from other rows. Formally, the Gauss-Jordan elimination algorithm finds an $m \times (n-m)$ matrix L, an $m \times m$ transformation matrix G and an $n \times n$ permutation matrix P such that

$$GBP = [I_m, L]. \tag{27}$$

In practice only the matrices P and L are computed explicitly.

Algorithm: 3.1 (Gauss-Jordan elimination for $m \times n$ matrices with pivot search)

1. *Given is the $m \times n$ matrix B. The permutation matrix P is initially set to the $n \times n$ identity matrix matrix I_n.*
2. *For $k = 1 \dots m$ do:*
   - (a) *Find the (pivot) element p_k; the entry in $B_{k:m,k:n}$ having the maximum absolute value.*
   - (b) *If $|p_k| \ll 1$, B is numerically singular, terminate the algorithm and return an error message.*
   - (c) *Shift the pivot to B_{kk} by exchanging the rows and columns of B; the k-th row $B_{k:}$ is exchanged with the row of the pivot and the k-th column $B_{:k}$ is exchanged with the column of the pivot.*
   - (d) *Exchange the same columns in the permutation matrix P as in B.*
   - (e) *Divide all nonzero entries in the k-th column of B by the pivot element p_k;*

     $$\lambda_i := \begin{cases} B_{ik}/p_k & \text{if } B_{ik} \neq 0, \\ 0 & \text{otherwise,} \end{cases} \quad p_k = B_{kk}. \tag{28}$$

   - (f) *Overwrite the rows $B_{i:}$ of B with*

     $$B'_{i:} := \begin{cases} \lambda_k B_{k:} & \text{if } i = k, \\ B_{i:} - \lambda_i B_{k:} & \text{otherwise,} \end{cases}$$

     making the kth column of B' to the kth column of the identity matrix I_m.
3. *Since the matrix B now has the form $B = [I_m, L]$, return the matrix L and the column permutation matrix P.*

Example. 3.2 *We apply the above algorithm for*

$$B = \begin{pmatrix} 3 & 3 & 1 & 0 \\ 2 & 6 & 0 & 1 \end{pmatrix} \text{ and } P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

- *($k = 1$) The maximal element of B is $B_{22} = 6$, which is chosen as the first pivot p_1. In B we exchange the first row with the second one, and in B and P we exchange the first column with the second one, which results in*

  $$B = \begin{pmatrix} 6 & 2 & 0 & 1 \\ 3 & 3 & 1 & 0 \end{pmatrix} \text{ and } P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

  Then the multiplier $\lambda_2 = 3/6 = 1/2$ is computed. We divide the first row with the pivot, subtract $\lambda_2 B_{1:} = (3 \;\; 1 \;\; 0 \;\; 1/2)$ from the second one, and get

  $$B = \begin{pmatrix} 1 & \frac{1}{3} & 0 & \frac{1}{6} \\ 0 & 2 & 1 & -\frac{1}{2} \end{pmatrix}.$$

- *($k = 2$) The pivot p_2 is $B_{22} = 2$ which is the maximum element of the submatrix $B_{2,2:4}$. In this case the pivot is at the correct position, therefore no exchange of the rows or columns of B or P is needed. Since $\lambda_1 = (1/3)/2 = 1/6$ from the first row we subtract $\lambda_1 B_{2:} = (0 \;\; 1/3 \;\; 1/6 \;\; -1/12)$, then divide the second one with the pivot, and get*

  $$B = \left[\; I_2 \;\; L \;\right], \quad L := \begin{pmatrix} -\frac{1}{6} & \frac{1}{4} \\ \frac{1}{2} & -\frac{1}{4} \end{pmatrix} \text{ and } P = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{29}$$

The algorithm is finished, the matrices L and P from (29) are returned.

The aim of the original Gauss-Jordan inversion is to find the inverse of the $m \times m$ matrix A. The first version of the Gauss-Jordan inversion was numerically unstable since it did not use pivoting. If we set $B := [A, I_m]$ in the Gauss-Jordan elimination Algorithm 3.1 we get the Gauss-Jordan inversion with pivoting, by (27) we have

$$G[A, I_m]P = [I_m, L], \tag{30}$$

and after m iterations the Algorithm 3.1 results in the matrix $[I_m, L]$ and the column permutation P. Note that for this selection of B even if A is singular B is regular by construction and thus Algorithm 3.1 never returns an error message.

Proposition. 3.3 *If the column permutation matrix P returned by Algorithm 3.1 consists only of permutations of the first m columns, then the matrix $C := \hat{P}L$ with $\hat{P} := P_{1:m,1:m}$ is the inverse of A.*

Proof. Since P consist only of permutations of the first m columns it must have the form

$$P = \begin{pmatrix} \hat{P} & 0 \\ 0 & I_m \end{pmatrix}. \tag{31}$$

Then by (30)

$$G[A, I_m]P = [I_m, L] \Rightarrow GA\hat{P} = I_m,\ GI_m = L \Rightarrow LA\hat{P} = I_m \Rightarrow (\hat{P}L)A = I_m. \tag{32}$$

Proving that A is regular and $\hat{P}L$ is the inverse of A. □

Example. 3.4 *In Example 3.2, we had*

$$B = [A, I_2] \text{ with } A = \begin{pmatrix} 3 & 3 \\ 2 & 6 \end{pmatrix}.$$

By (29)*, P has the form of* (31) *with* $\hat{P} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}$*, thus the matrix*

$$C = \hat{P}L = B = \begin{pmatrix} \frac{1}{2} & -\frac{1}{4} \\ -\frac{1}{6} & \frac{1}{4} \end{pmatrix},$$

is the inverse of A.

Scaling: Note that if the matrix A is regular, but not scaled correctly, entries from the wrong part of B may be chosen as pivot elements. In this case even though A is regular, (31) does not hold, and the inverse of A cannot be found by the algorithm. This would happen if in the above example we would divide the entries of A by 10. Since this would only scale the matrix A, it would be still invertible. In this case in the first step of the algorithm $B_{31} = 1$ would be chosen as the pivot therefore the prerequisites of Proposition 3.3 would not be met. To solve this problem, Algorithm 3.1 can be improved by applying *suitable scaling* to A and I_m. We choose a diagonal row scaling matrix $U \in \mathbb{R}^{m\times m}$, a diagonal column scaling matrix $V \in \mathbb{R}^{m\times m}$ and an additional scaling constant δ and set

$$B := [UAV, \delta I_m].$$

in Algorithm 3.1 and obtain

$$G[UAV, \delta I_m]P = [I_m, L]. \tag{33}$$

Theorem. 3.5 *If the column permutation matrix P returned by Algorithm 3.1 consists only of permutations of the first m columns, then the matrix $C := \delta^{-1}V\hat{P}LU$ is the inverse of A.*

Proof. Since P consist only permutations of the first n columns we have

$$P = \begin{pmatrix} \hat{P} & 0 \\ 0 & I_m \end{pmatrix}, \tag{34}$$

then by (33)

$$GUAV\hat{P} = I_m,\ G\delta I_m = L \Rightarrow \delta^{-1}LUAV\hat{P} = I_m \Rightarrow (\delta^{-1}V\hat{P}LU)A = I_m, \tag{35}$$

holds, proving the assumption. □

We suggest the following choice of the scaling matrices U and V and the scaling constant δ: By the scaling method presented in Domes & Neumaier [8] we find matrices U and V such that the entries of UAV are between zero and one but not too close to zero. The second part of $B = [UAV, \delta I_m]$ consists of an $m \times m$ identity matrix, scaled with the constant δ. Setting δ as a very small positive number (e.g., $\delta := \sqrt{\varepsilon}$) prevents that – even for well conditioned matrices – some elements of the second part of B are chosen as pivots.
Extension of the Gauss-Jordan inversion. If the matrix A is not square or regular, there is no (two-sided) inverse, but preconditioner for A can be found, such that for a suitable chosen index set J the equality $CA_{:J} = I$ holds.

Theorem. 3.6 *Let $A, L \in \mathbb{R}^{m\times n}$, $U, G \in \mathbb{R}^{m\times m}$, $V \in \mathbb{R}^{n\times n}$, $P \in \mathbb{R}^{(n+m)\times(n+m)}$ and $\delta > 0$.*
(1) Suppose that

$$G[UAV, \delta I_m] = [I_m, L]P, \tag{36}$$

then

$$G = \delta^{-1}(P_{MM} + LP_{NM}), \tag{37}$$

where $M = \{1, \dots, m\}$ and $N = \{m+1, \dots, m+n\}$.
(2) If V is an invertible diagonal matrix, P is a permutation matrix and $R \subseteq \{1, \dots, n\}$ is an index set of size $r \le m$ such that

$$P_{RM}P_{MR} = I_r \tag{38}$$

holds, then

$$C := V_{RR}P_{RM}GU \tag{39}$$

satisfies

$$CA_{:R} = I_r. \tag{40}$$

Proof. We write P in block form as

$$P = \begin{pmatrix} P_{MN} & P_{MM} \\ P_{NN} & P_{NM} \end{pmatrix} \tag{41}$$

with $P_{MN} \in \mathbb{R}^{m\times n}$, $P_{NN} \in \mathbb{R}^{n\times n}$, $P_{MM} \in \mathbb{R}^{m\times m}$ and $P_{NM} \in \mathbb{R}^{n\times m}$. Then we have

$$[I_m, L]P = [I_mP_{MN} + LP_{NN}, I_mP_{MM} + LP_{NM}]. \tag{42}$$

From (42) and (36) the identities

$$GUAV = P_{MN} + LP_{NN} \text{ and } G = \delta^{-1}(P_{MM} + LP_{NM})$$

follow. The second equality proves assumption (1). To prove (2) we multiply the second equality with the matrix $W := V_{R:}P_{NM}$ from the left and with a vector x from the right side and obtain

$$WGUAVx = WP_{MN}x + WLP_{NN}x.$$

Without loss of generality, we assume that $R = \{1, \dots, r\}$. Choosing

$$x_i := \begin{cases} z_i/V_{ii} & \text{for } i \in R, \\ 0 & \text{otherwise,} \end{cases}$$

we find $x_R = V_{RR}^{-1} z_R$ and $AVx = A_{:R} z_R$. Since P is a permutation matrix, from (38) follows that all columns of P_{MR} contain a one; therefore the columns P_{NR} have to be zero columns. Since V^{-1} is diagonal, the matrix V_{RR}^{-1} only contains nonzero elements in the jth rows when $j \in R$. We summarize and obtain

$$\begin{array}{rcll} (P_{NR})_{ij} & = & 0 & \text{if} \quad j \in R \\ (V_{RR}^{-1})_{jk} & = & 0 & \text{if} \quad j \notin R. \end{array} \tag{43}$$

Then by (43) follows that

$$(P_{NR} V_{RR}^{-1})_{ik} = \sum_{j=1}^{n} (P_{NR})_{ij} (V_{RR}^{-1})_{jk} = 0, \quad \text{for all } i, j.$$

Therefore $P_{NR} V_{RR}^{-1} z_R = 0$ and

$$V_{RR} P_{RM} GUA z_R = V_{RR} P_{RM} P_{MR} V_{RR}^{-1} z_R.$$

By (38) we get

$$V_{RR} P_{RM} GUA z_R = z_R.$$

implying (39) and (40). □

Using the results of the above proposition we generalize Algorithm 3.1:

Algorithm: 3.7 (Gauss-Jordan preconditioner)

1. *Given is the matrix $A \in \mathbb{R}^{m \times n}$, the diagonal row scaling matrix $U \in \mathbb{R}^{m \times m}$ and the diagonal column scaling matrix $V \in \mathbb{R}^{n \times n}$.*
2. *We set $u = n + m$, $K = (1, \ldots, u)$ and $B = [UAV, \delta I_m] \in \mathbb{R}^{m \times u}$.*
3. *For $k = 1 \ldots m$ do:*
   - *(a) Find the pivot $p_k := B_{ij}$ having the maximum absolute value from the submatrix $B_{k:m,k:u}$.*
   - *(b) Exchange the kth row $B_{k,:}$ of B with the row $B_{i,:}$ of the pivot and the kth column $B_{:,k}$ of B with the column $B_{:,j}$ of the pivot.*
   - *(c) Exchange K_k with K_j in the index list K.*
   - *(d) Compute each λ_i as given in (28).*
   - *(e) For each $i \neq k$ overwrite the row $B_{i:}$ with $B_{i:} - \lambda_i B_{k:}$.*
   - *(f) Overwrite $B_{k:}$ with $B_{k:}/p_k$.*
4. *The row permutation matrix can be set by using the found index set K:*

$$P = \begin{pmatrix} P_{MN} & P_{MM} \\ P_{NN} & P_{NM} \end{pmatrix} = (I_{n+m})_{:K}.$$

5. *The matrix $G = \delta^{-1}(P_{MM} + LP_{NM})$ is computed.*
6. *Generate the index set R:*

$$R = \{j \in R_{1:m} \mid K_j \leq n\}$$

7. *Since for $\hat{P}_{RM} := P_{MR}^T$ condition* (38) *in Theorem 3.6 holds, the preconditioner,*
$$C = V_{RR} P_{RN}^T GU$$
with $CA_{:R} = I_r$ is obtained.

8. *We have found an index set R and a matrix $C \in \mathbb{R}^{r\times m}$ such that* (36) *holds. Return the matrix C and the index set R.*

The preconditioner found by Algorithm 3.7 can be used for solving under- or overdetermined linear equation systems. If the matrix A is square and has full rank, Algorithm 3.7 returns the inverse of A.

Example. 3.8 *Let*
$$A = \begin{pmatrix} \frac{1}{2} & \frac{1}{6} & \frac{1}{6} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{pmatrix},\ U = \begin{pmatrix} 6 & 0 \\ 0 & 8 \end{pmatrix},\ V = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{pmatrix}, \quad \textit{and } \delta = 1.$$
Then the matrix
$$B = [UAV, \delta I_m] = \begin{pmatrix} 3 & 1 & 3 & 1 & 0 \\ 2 & 4 & 6 & 0 & 1 \end{pmatrix},$$
is similar to the matrix in Example 3.4; the same pivots will be chosen. After two iterations, we get
$$B = \begin{pmatrix} 1 & 0 & \frac{1}{2} & -\frac{1}{6} & \frac{1}{4} \\ 0 & 1 & -\frac{1}{2} & \frac{1}{2} & -\frac{1}{4} \end{pmatrix}, \quad K = (3, 1). \tag{44}$$
Then we have $R = \{1, 3\}$,
$$G = \delta^{-1}(P_{MM} + LP_{NM}) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} \frac{1}{2} & -\frac{1}{6} & \frac{1}{4} \\ -\frac{1}{2} & \frac{1}{2} & -\frac{1}{4} \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -\frac{1}{6} & \frac{1}{4} \\ \frac{1}{2} & -\frac{1}{4} \end{pmatrix},$$
$$C = V_{RR} P_{RM} GU = \begin{pmatrix} 1 & 0 \\ 0 & 3 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -\frac{1}{6} & \frac{1}{4} \\ \frac{1}{2} & -\frac{1}{4} \end{pmatrix} \begin{pmatrix} 6 & 0 \\ 0 & 8 \end{pmatrix} = \begin{pmatrix} 3 & -2 \\ -3 & 6 \end{pmatrix},$$
and find that $CA_{:R} = I_r$ holds.

The above considerations about a suitable scaling (in the sense of making the algorithm choose all linear independent columns as pivot columns by correctly choosing U, V and δ) applies again. If P permutes some of the last m columns, then either A has non-maximal numerical rank or the scaling was not chosen suitably.

Lemma. 3.9 *If the matrix R returned by Algorithm 3.7 was computed by using exact arithmetic and suitable scaling then $r := |R|$ is the rank of A .*

Proof. Without the loss of generality we assume that in the kth iteration step the first part $B_{k,1:n}$ of the pivot row and the first part $B_{i,1:n}$ of another row ($k < i$) are linearly dependent. Therefore $B_{i,1:n} = cB_{k,1:n}$ holds for some constant c. Since $\lambda_i = B_{ik}/B_{kk} = c$ the entries $B_{i,1:n}$ will be overwritten with $B_{i,1:n} - cB_{k,1:n} = 0$. From this point on the first n entries of this row only contain zeros, and sooner or later a pivot has to be selected from the lower right $m \times m$ part of B. Since this happens for each linear dependent row the number of pivots selected from the first part of B gives us the rank of the matrix A. □

Since in our implementation inexact arithmetic is used, due to the rounding errors, we only get the numerical rank, which (if the scaling is suitable) is correct for the most non-degenerate matrices.

4 Linear contraction

Based on the Gauss-Jordan method discussed in Section 3, we present a simple technique for reducing the bounds of x of the linear system (1).
First a Gauss-Jordan preconditioner for the matrix E is computed; we choose suitable scaling matrices U and V and a scaling factor δ then apply Algorithm 3.7 for the matrices E, U, V and the scaling factor δ.
A possible choice for the scaling was given in Section 3. For this application a better alternative is to specify the scaling matrices U and V such that the rows of E matching the constraints having tighter bounds are preferred as pivot rows. Similarly, by this scaling the columns of E matching the variables with tighter bounds, are preferred as pivot columns. According to this we set

$$d := \max\{\underline{x}_i,\ \overline{x}_j \mid i = 1 \dots n,\ j = 1 \dots n\},\quad \mathbf{z} = \mathbf{x} \cap [-d, d]^n$$

and for the scaling matrices

$$U = \mathrm{diag}(u),\quad V = \mathrm{diag}(v) \tag{45}$$

with

$$u = ((\overline{b} - \underline{b}) + \delta|\mathbf{b} - E\mathbf{x}|)\ \text{ and }\ v = (\overline{z} - \underline{z})/(\max\{\overline{z}_i - \underline{z}_i \mid i = 1 \dots n\}).$$

For the scaling constant (as in Section 3) we choose $\delta = \sqrt{\varepsilon}$.
The algorithm returns an index list R with $|R| = r$ and a matrix $C \in \mathbb{R}^{r \times m}$ such that $CE_{:R} = I_r$. We set $K := \neg R$, multiply (1) with the matrix C and obtain

$$CE_{:R}x_R + CE_{:K}x_K \in C\mathbf{b},\ x_R \in \mathbf{x}_R,\ x_K \in \mathbf{x}_K. \tag{46}$$

Since $CE_{:R} = I_r$, if we substitute the bounds for x_K we get

$$x_R \in \hat{\mathbf{b}},\ \hat{\mathbf{b}} := (C\mathbf{b} - CE_{:K}\mathbf{x}_K),\ x_R \in \mathbf{x}_R.$$

and if we cut $x_R \in \mathbf{b}'$ with the original bounds $\mathbf{x}_R$ on the variables x_R we end up in

$$x_R \in \hat{\mathbf{x}},\ \hat{\mathbf{x}} := \hat{\mathbf{b}} \cap \mathbf{x}_R. \tag{47}$$

If the matrix E is square and has full rank ($n = m = r$) then we get

$$x \in \hat{\mathbf{x}},\ \hat{\mathbf{x}} := C\mathbf{b} \cap \mathbf{x}.$$

In inexact arithmetic, the computation of the preconditioner C is not rounding error free, and thus only $CE_{:R} \approx I_r$ holds. This modifies (47) to

$$Mx_R \in \hat{\mathbf{b}},\ M := CE_{:R},\ x_R \in \mathbf{x}_R.$$

Since the off diagonal entries of M are tiny we have

$$x_R \in \hat{\mathbf{x}} \text{ with } \hat{\mathbf{x}}_i := (M_{ii}^{-1}(\hat{\mathbf{b}}_i - \sum_{j=1,\ j\neq i}^{r} M_{ij}\mathbf{x}_i)) \cap \mathbf{x}_i,\ M := CE_{:R},\ \hat{\mathbf{b}} := (C\mathbf{b} - CE_{:K}\mathbf{x}_K). \tag{48}$$

Again, if the matrix E is square and of full rank, then $CE \approx I_m$ and we get the bounds

$$x \in \hat{\mathbf{x}} \text{ with } \hat{\mathbf{x}}_i := \frac{(C\mathbf{b})_i - \sum_{j=1,\ j\neq i}^{r}(CE)_{ij}\mathbf{x}_i}{(CE)_{ii}} \cap \mathbf{x}_i.$$

By using this method we obtain new bounds for the variables x_R.
An alternative to the above method is to use constraint propagation on (46). Constraint propagation for quadratic (and linear) systems is discussed in [9]. This alternative costs more computational time but it also yields new bounds on the variables x_K not only on x_R. Both approaches are useful; the decision which alternative is preferable is based on the dimension of the problem.

5 Linear bounding

Another simple, efficient, but costly method for improving the bounds on the variables in linear systems is presented in this section.
Consider the linear system (1) of n variables and m two-sided inequalities and choose $k \le n$ variables, where the most reduction is expected. Alternatively all n variables can be selected. Then for each selected variable x_i solve two linear programs (one for each sign) given by

$$\begin{array}{ll} \min & f(x) := \pm x_i \\ \text{s.t.} & Ex \in \mathbf{b},\ x \in \mathbf{x}. \end{array} \tag{49}$$

Let $\hat{x}^i_+$ and $\hat{x}^i_-$ be the solutions of (49) for the different signs and let y^i_+ and y^i_- the multiplier vectors of the solutions not containing the multipliers corresponding to the bound constraints. If all $2k$ linear programs are solved, the multipliers are collected in a $2k \times m$ matrix

$$Y \in \mathbb{R}^{2k\times m},\ \ Y_{:,2i-1} = y^i_+,\ \ Y_{:,2i} = y^i_- \quad \text{for all } i = 1,\dots,k.$$

The matrix Y is used to precondition the linear system (1) resulting in a new system of $2k$ inequalities

$$\hat{\mathbf{E}}x \in \hat{\mathbf{b}},\ x \in \mathbf{x},\ \hat{\mathbf{E}} := Y[E,E],\ \hat{\mathbf{b}} := Y\mathbf{b}. \tag{50}$$

To ensure mathematical rigor, the interval coefficient matrix $\hat{\mathbf{E}}$ and the box $\hat{\mathbf{b}}$ must be computed by using interval arithmetic. Since each solution $x^j_\pm$ of (50) with corresponding multipliers $y^j_\pm$ must satisfy the first order optimality conditions the equality

$$\nabla_x \mathcal{L}(x^j_\pm, y^j_\pm) = \pm e^j - (y^j_\pm)^T E = 0$$

holds and each row $\hat{\mathbf{E}}_{k:}$ of $\hat{\mathbf{E}}$ contains only one dominant entry $\hat{\mathbf{E}}_{kj}$ and all other entries should be thin intervals containing zero. Therefore (50) can be solved row-wise for each row $k = 1,\dots,2k$, by substituting the bounds $\mathbf{x}_i$ for the variables with nearly zero coefficients and bringing the corresponding interval terms to the right hand side. This results in a new bound

$$x_j \in \hat{\mathbf{x}}_j,\ \hat{\mathbf{x}}_j := \mathbf{x}_j \cap (\hat{b} - \sum_{i=1,i\ne j}^{n} \hat{\mathbf{E}}_{ki}\mathbf{x}_i),$$

on the variable x_j. Alternately, constraint propagation for quadratic (and linear) systems is discussed in [9] can be used to solve (50).

6 Linear relaxations for quadratic expressions

The following sections elaborate techniques for creating linear relaxations of quadratic constraint satisfaction problems.

Let $p(x) : \mathbb{R}^n \to \mathbb{R}$ be a mapping. The function $u(x)$ is called an *underestimator* of $p(x)$ over the box $\mathbf{x}$ if $u(x) \leq p(x)$ holds for all $x \in \mathbf{x}$. Similarly, the function $v(x)$ is called an *overestimator* of $p(x)$ over the box $\mathbf{x}$, if $p(x) \leq v(x)$ holds for all $x \in \mathbf{x}$. If both an underestimator $u(x)$ and an overestimator $v(x)$ is given then

$$p(x) \in [u(x), v(x)] \text{ for all } x \in \mathbf{x}$$

is an *enclosure* of $p(x)$ over the box x.

Theorem. 6.1 *Let $p(x), h(x) : \mathbb{R}^n \to \mathbb{R}$ be mappings, let $\mathbf{c}$, $\mathbf{d}$ be intervals and let $x \in \mathbf{x}$. If*

$$p(x) \in \mathbf{c} \quad \Rightarrow \quad h(x) \in \mathbf{d} \tag{51}$$

for all $x \in \mathbf{x}$ then

$$h(x) - p(x) \in [\underline{d} - \underline{c},\ \overline{d} - \overline{c}] \tag{52}$$

is satisfied for all $x \in \mathbf{x}$. In this case, the two-sided inequality $h(x) \in \mathbf{d}$ is called a relaxation *of $p(x) \in \mathbf{c}$ over the box $\mathbf{x}$.*

Proof. ($\Rightarrow$) By (51), for a real number r (choosen later) the inequality $h(x) \geq p(x) + r$ must hold for all $x \in \mathbf{x}$. Since $p(x) \in \mathbf{c}$, $h(x) \geq p(x)+r \geq \underline{c}+r$. Since $h(x) \in \mathbf{d}$, $h(x) \geq \underline{d}$ must also hold. Choose r as minimal and get $\underline{c}+r = \underline{d}$ ending up in $h(x) \geq p(x)+r = p(x)+\underline{d}-\underline{c}$. This given the lower inequality $\underline{d}-\underline{c} \leq h(x)-p(x)$ of (52). The upper inequality $h(x)-p(x) \leq \overline{d}-\overline{c}$ can be obtained in the same way.
($\Leftarrow$) By (52) the inequality $\underline{d} - \underline{c} \leq h(x) - p(x)$ holds for all $x \in \mathbf{x}$. Bringing $p(x)$ to the left hand side results in $p(x)+\underline{d}-\underline{c} \leq h(x)$. By (62) $\underline{c} \leq p(x)$ and thus $\underline{d} = \underline{c}+\underline{d}-\underline{c} \leq h(x)$. The inequality $h(x) \leq \overline{d}$ can be obtained similarly. Therefore (51) holds, proving the assumption. $\square$

In the following subsections give a step-by-step explanation how linear relaxations for quadratic expressions are generated; in Subsection 6.1 linear relaxations for univariate, quadratic expressions are construct, then in Subsection 6.2 separable, multivariate, quadratic expressions are handled, finally in Subsection 6.3 the most general case of generating linear relaxations for multivariate, not necessary separable, quadratic expressions is discussed.

6.1 Linear relaxations for univariate quadratic expressions

Without loss of generality, an arbitrary univariate quadratic expressions, can be written in the form

$$q(x) \in \mathbf{c}, \quad q(x) := ax^2 + bx, \quad x \in \mathbf{x}, \tag{53}$$

where a and b are constant and $\mathbf{c}$ and $\mathbf{x}$ are intervals. We assume without the loss of generality that $a > 0$ since for $a = 0$ we already have a linear expression, with no need of relaxing, and for $a < 0$ all the observations below hold with trivial modifications.
For univariate functions Kolev [15] proposes linear relaxations of the form

$$ex \in \mathbf{d} \text{ for } x \in \mathbf{x} \text{ with } q(x) \in \mathbf{c}, \tag{54}$$

where e is a constant and $\mathbf{d}$ is an interval (see, Figure 1). Kolev states that this relaxation is optimal if $\mathbf{w}$ has minimal width and uses a generalized representation of intervals to compute e and $\mathbf{d}$.

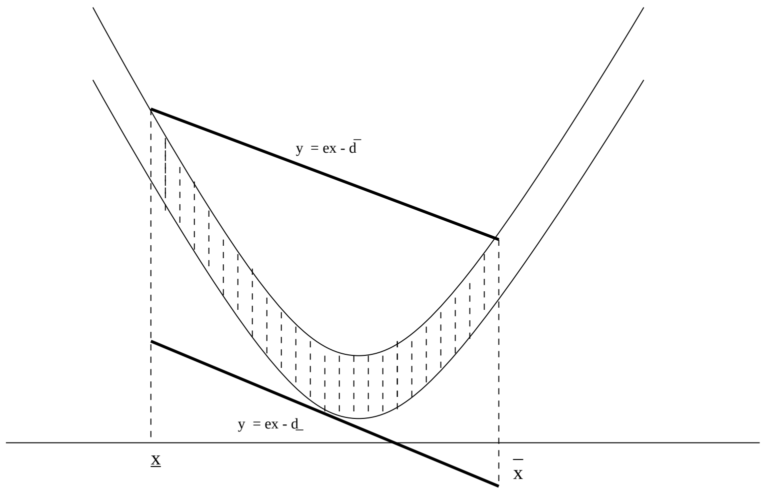


Figure 1: Linear relaxations by Kolev

Another approach is the QUAD algorithm of LEBBAH et al. [18], where linear under- and overestimators are used to generate linear relaxations. Since $a > 0$, we obtain for any $z \in \mathbf{x}$ linear underestimators

$$L_z(x) := l'(z)(x - z) + l(z) \text{ where } l(x) := q(x) - \underline{c},$$

(in [18], the two tangents of $l(x)$ for $z = \underline{x}$ and $z = \overline{x}$ are chosen) and the linear overestimator

$$L(x) := \frac{q(\overline{x}) - q(\underline{x})}{\overline{x} - \underline{x}} x + \frac{u(\underline{x})\overline{x} - u(\overline{x})\underline{x}}{\overline{x} - \underline{x}} \text{ where } u(x) := q(x) - \overline{c},$$

(the secant of $u(x)$ between the points $(\underline{x}, u(\underline{x}))$ and $(\overline{x}, u(\overline{x}))$) (see, Figure 2).

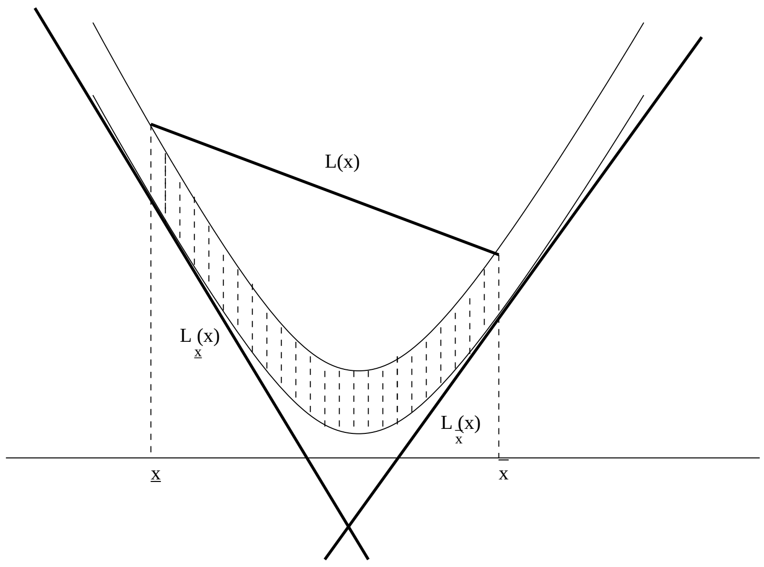


Figure 2: Linear relaxations by Lebbah & Rueher

Note that while L is the best choice for linear overestimator, the choice and the number of the underestimators L_z are arbitrary. The two underestimators suggested by [18] could

be replaced by a single one (e.g., $L_z(\mathrm{mid}(\mathbf{x}))$) or refined by adding more (e.g., $L_z(\mathrm{mid}(\mathbf{x}))$ would be a good choice). The latter can be made adaptive to satisfy a given error bound, and is then called the sandwich method (see TAWARMALANI & SAHINIDIS [30]), it used in the solver BARON.
According to this if Z is a finite set with values in the box $\mathbf{x}$ and $n_z = |Z|$, then the system

$$L_z(x) \geq 0, \quad L(x) \leq 0, \quad z \in Z, \quad x \in \mathbf{x}, \tag{55}$$

of $n_z + 1$ linear inequalities is a linear relaxation of (53).
To give an uniform representation for the two methods we choose the form (54) where e is now a k–dimensional vector $\mathbf{d}$ is a k–dimensional box. The relaxation by Kolev is included in this form for $k = 1$ while the inequalities by Lebbah & Rueher can be embedded by setting

$$e_i = \begin{cases} q'(X_i) & \text{for} \quad i = 1, \ldots, n_z, \\ \frac{q(\overline{x}) - q(\underline{x})}{\overline{x} - \underline{x}} & \text{if} \quad i = n_z + 1, \end{cases} \quad \mathbf{d}_i = \begin{cases} [q'(X_i)X_i - q(X_i) + \underline{c}, \infty] & \text{for} \quad i = 1, \ldots, n_z, \\ [-\infty, \frac{q(\overline{x})\underline{x} - q(\underline{x})\overline{x}}{\overline{x} - \underline{x}} - \overline{c}] & \text{if} \quad i = n_z + 1. \end{cases}$$

6.2 Linear relaxations for separable quadratic expressions

We consider an arbitrary separable quadratic expression, which we write without loss of generality in the form

$$p(x) \in \mathbf{c}, \quad p(x) := a^T x^2 + b^T x, \quad x \in \mathbf{x}, \tag{56}$$

where x^2 is the component-wise square of x, a and b are n–dimensional vectors, $\mathbf{c}$ is an interval and $\mathbf{x}$ is an n–dimensional box. We assume that $a \neq 0$ since otherwise we already would have a linear expression, with no need of relaxing. The results of the univariate case can be directly applied to the multivariate case with slight modifications:
For a function of n variables, we consider linear relaxations of the form

$$e^T x \in \mathbf{d}, \quad x \in \mathbf{x}, \tag{57}$$

where e is an n–dimensional vector and $\mathbf{d}$ is an interval. Since (57) is a linear relaxation of (56) by (52)

$$e^T x - q(x) \in [\underline{d} - \underline{c}, \overline{d} - \overline{c}],$$

holds. For this special case of a separable quadratic expression this simplifies to

$$u^T x - a^T x^2 \in [\underline{d} - \underline{c}, \overline{d} - \overline{c}], \quad u := e + b.$$

If we choose a suitable slope vector e, the exact range $\mathbf{t}$ of the quadratic expression on the left hand side is easy to compute rigorously (see DOMES & NEUMAIER [9]). This results in the equality $\mathbf{t} = [\underline{d} - \underline{c}, \overline{d} - \overline{c}]$ from which the interval $\mathbf{c}$ follows directly. Possible selections of the slope vector could be the derivate $2a \cdot x + b$ of $q(x)$ (where $\cdot$ denotes the componentwise product) in a suitable chosen point $z \in \mathbf{x}$ (midpoint, upper or lower bound, or midpoint of a promising region of $\mathbf{x}$). Another useful selection for the slope vector is the the secant slope $\frac{q(\overline{x}) - q(\underline{x})}{\overline{x} - \underline{x}}$, of between the points $(\underline{x}, q(\underline{x}))$ and $(\overline{x}, q(\overline{x}))$).
To integrate the method of Lebbah & Rueher [18] for the multivariate case, we generate the linear relaxations for each univariate quadratic expression separately. Let

$$Q := \{k \in \{1, \ldots, n\} \mid a_k \neq 0\}, \quad L := \{k \in \{1, \ldots, n\} \mid b_k \neq 0\}, \tag{58}$$

with $n_q = |Q|$ and $n_l = |L|$ be the index sets then (56) can be written as

$$\begin{aligned}&\textstyle\sum_{k\in Q} y_k + \sum_{k\in L} b_k x_k \in \mathbf{c}, \quad x \in \mathbf{x},\\ &y_k = a_k x_k^2 \text{ for all } k \in Q.\end{aligned}$$

To generate the linear relaxations of the n_q univariate quadratic expressions $a_k x_k^2$ we apply the results of the previous section; we choose the set Z, compute the n_z+1 linear inequalities

$$e^k x_k \in \mathbf{d}^k, \text{ for } x_k \in \mathbf{x}_k,$$

for the n_q quadratic expressions separately, whereby e^k is now a (n_z+1)–dimensional vector and $\mathbf{d}^k$ is a (n_z+1)–dimensional box. Let Z be a finite set with values in $\mathbf{x}$ and $n_z = |Z|$ then the linear relaxation can be given as a system of $n_y(n_z+1)+1$ inequalities:

$$\begin{aligned}&\textstyle\sum_{k\in Q} y_k + \sum_{k\in L} b_k x_k \in \mathbf{c}, \quad x \in \mathbf{x}, \quad y_k - e_i^k x_k \in \mathbf{d}_i^k, \text{ for all } k \in Q.\\ &e_i^k = \begin{cases} 2a_k X_i & \text{for} \quad i = 1,\dots,n_z,\\ a_k(\overline{x}+\underline{x}) & \text{if} \quad i = n_z+1,\end{cases} \quad \mathbf{d}_i^k = \begin{cases} [-a_k X_i^2, \infty] & \text{for} \quad i = 1,\dots,n_z,\\ [-\infty, -a_k\overline{x}_k\underline{x}_k] & \text{if} \quad i = n_z+1.\end{cases}\end{aligned}$$

In order to give an uniform representation for the two methods, we propose the general form

$$Ex \in \mathbf{d}, \quad x \in \mathbf{x}, \tag{59}$$

with $E \in \mathbb{R}^{h\times n}$ and $\mathbf{d}$ is an h–dimensional box. The relaxation by Kolev is included in this form for $h = 1$ while the inequalities by Lebbah & Rueher can be embedded by setting the $h = n_y(n_z+1)+1$ components and increasing the number of variables to $n + n_q$.

6.3 Linear relaxations for quadratic expressions

We consider an arbitrary multivariate quadratic expressions

$$p(x) \in \mathbf{c}, \quad p(x) := \sum_{k\in Q} a_k x_k^2 + \sum_{(j,k)\in B} b_{jk} x_j x_k + \sum_{k\in L} b_k x_k, \quad x \in \mathbf{x}, \tag{60}$$

where the $a_k x_k^2$ are the quadratic, the $b_{jk}x_j x_k$ are the bilinear, the $b_k x_k$ are the linear terms and $\mathbf{c}$ is an interval. The sets Q and L are are from (58), while

$$B := \{(j,k) \in \{1,\dots,n\} \times \{1,\dots,n\} \mid b_{jk} \neq 0\}, \quad n_b = |B|,$$

and we assume that B is non-empty.
We discuss two different methods to deal with the bilinear entries, the first one is based on the results of Domes & Neumaier [9] and removes the bilinear terms by modifying the quadratic or linear coefficients of (60) while the second one from McCormick [19] adds four linear inequalities for each bilinear term.
In Section 5 of [9] two different methods are presented for separating the constrains; approximation of the bilinear terms by quadratic, by linear and by constant ones. The choice is made for each bilinear term $b_{jk}x_j x_k$ separately and the decision is based on the bounds of the variables x_k and x_j.
Case 1: If both $\mathbf{x}_k$ and $\mathbf{x}_j$ are bounded we approximate the bilinear term by linear terms, obtaining

$$b_{jk}x_j x_k - b_{jk}z_j x_k - b_{jk}z_k x_j \in [\min_i \nabla u_i, \max_i \Delta u_i]$$

where $z = \operatorname{mid} x$ and

$$\begin{array}{ll} u_1 = b_{jk}((\underline{x}_j - z_j)\underline{x}_k - z_k\underline{x}_j), & u_2 = b_{jk}((\overline{x}_j - z_j)\underline{x}_k - z_k\overline{x}_j), \\ u_3 = b_{jk}((\underline{x}_j - z_j)\overline{x}_k - z_k\underline{x}_j), & u_4 = b_{jk}((\overline{x}_j - z_j)\overline{x}_k - z_k\overline{x}_j). \end{array}$$

This modifies the linear and the constant constraint coefficients to

$$b'_k := b_{jk}z_j + b_k, \quad b'_j := b_{jk}z_k + b_j, \quad \mathbf{c}' := [\underline{c} - \max_i \Delta u_i, \ \underline{c} - \min_i \nabla u_i].$$

Case 2: If the interval $\mathbf{x}_k$ or the interval $\mathbf{x}_j$ is unbounded we eliminate the bilinear terms $b_{jk}x_jx_k$ by adding the quadratic term

$$d_{jk}(x_j - v_{jk}x_k)^2 \text{ with } v_{jk} := \operatorname{sign}(b_{jk})\sqrt{\frac{a_k}{a_j}}, \quad d_{jk} := \frac{b_{jk}}{2v_{jk}},$$

to the constraint. This results in the new quadratic coefficients

$$a'_k := a_k + d_{jk}, \quad a'_j := a_j + \frac{b_{jk}v_{jk}}{2}. \tag{61}$$

This results in a separable quadratic expression (56), which can be relaxed by using the methods discussed in the Section 6.2.
The convex envelope of a function $f(x)$ over the box $\mathbf{x}$ is the tightest convex underestimating function for $f(x)$ for $x \in \mathbf{x}$. Al-Khayyal [1] and McCormick [19] developed an efficient relaxation technique to obtain the convex envelope for the bilinear terms x_jx_k. This requires that $\mathbf{x}_j$ and $\mathbf{x}_k$ are bounded. In this case the convex envelope of $\mathbf{x}_j\mathbf{x}_k$ is convex polyhedral (see Rikun [24]), and its convex and concave parts can be given as

$$\begin{array}{l} \operatorname{Conv}(x_jx_k) := \max\{\underline{x}_kx_j + \underline{x}_jx_k - \underline{x}_k\underline{x}_j, \ \overline{x}_kx_j + \overline{x}_jx_k - \overline{x}_k\overline{x}_j\}, \\ \operatorname{Conc}(x_jx_k) := \min\{\overline{x}_kx_j + \underline{x}_jx_k - \overline{x}_k\underline{x}_j, \ \underline{x}_kx_j + \overline{x}_jx_k - \underline{x}_k\overline{x}_j\}. \end{array}$$

Therefore, a linear relaxation of the bilinear terms can be given by substituting a new variables y_{jk} for every x_jx_k, and adding the following linear constraints:

$$\begin{array}{ll} y_{jk} \geq \underline{x}_kx_j + \underline{x}_jx_k - \underline{x}_k\underline{x}_j, & y_{jk} \geq \overline{x}_kx_j + \overline{x}_jx_k - \overline{x}_k\overline{x}_j, \\ y_{jk} \leq \overline{x}_kx_j + \underline{x}_jx_k - \overline{x}_k\underline{x}_j, & y_{jk} \leq \underline{x}_kx_j + \overline{x}_jx_k - \underline{x}_k\overline{x}_j. \end{array}$$

Androulakis et al. [2] showed that the maximum difference between variable y_{jk} and the bilinear term x_jx_k depends on the widths of $\mathbf{x}_j$ and $\mathbf{x}_k$ and can be given as $\frac{1}{4}(\overline{x}_j - \underline{x}_j)(\underline{x}_k - \underline{x}_k)$. Therefore, algorithms using convex envelopes to underestimate bilinear terms seek maximal domain reduction, making preprocessing methods helpful in uncovering implicit bounds.
In their Quad algorithm, Lebbah & Rueher [18] used McCormick's convex and concave envelopes to relax the bilinear terms. This results in $4n_b$ additional inequalities which can added to the representation (57), increasing the total number of inequalities to $h = n_y(n_z + 1) + 4n_b + 1$ and the number of variables to $n + n_q + n_b$. The method of Domes & Neumaier does not generate additional inequalities but McCormick's method may yield relaxations of higher quality.

7 Polynomial constraint satisfaction problems

We consider continuous constraint satisfaction problems of the form

$$G(x) \in \mathbf{F}, \quad x \in \mathbf{x}, \quad G(x) \in \mathbf{G}(x). \tag{62}$$

The m general constraint are interpreted as componentwise enclosures $G_i(x) \in \mathbf{F}_i$ $(i = 1,\dots,m)$. This form includes equality constraints if $\mathbf{F}_i = [\underline{F}_i, \overline{F}_i]$ is a degenerate interval $(\underline{F}_i = \overline{F}_i)$, inequality constraints if one of the bounds is infinite and two–sided constraints $\underline{F}_i \leq G_i(x) \leq \overline{F}_i$ if both bounds are finite. For allowing uncertainties in the constraint coefficients, we allow $G(x)$ to vary in the given interval function $\mathbf{G}(x)$. Similarly, the n bound constraints are interpreted as enclosures $x_j \in \mathbf{x}_j$ with $j = 1,\dots,n$. Again, fixed variables and one-sided bounds on the variables are included as special cases. Each $x \in \mathbf{x}$ for which the constraints of (62) are satisfied, is called a feasible point or a solution of the constraints satisfaction problem. The set of all feasible points is called the feasible domain. If the function $G(x)$ has only quadratic, bilinear and linear terms (62) is called a quadratic constraint satisfaction problems. If $G(x)$ is only linear in the variables we end up in the linear system given by (1). A linear system of the form of (1) can be obtained by relaxing (62):

Theorem. 7.1 *Every feasible point of the constraint satisfaction problem* (62) *satisfies* (1) *iff for all $x \in \mathbf{x}$ and $G(x) \in \mathbf{G}(x)$ the inequalities*

$$Ex - G(x) \in [\underline{b} - \underline{F},\ \overline{b} - \overline{F}] \tag{63}$$

hold. In this case, the linear system (1) *is a linear relaxation of* (62).
If (63) *holds, then*

$$Ex \in \mathbf{b}' \textit{ with } \mathbf{b}' = \mathbf{b} \cap E\mathbf{x} \tag{64}$$

and

$$G(x) \in \mathbf{F}' \textit{ with } \mathbf{F}' = \mathbf{F} \cap G(\mathbf{x}) \cap [\inf(E\mathbf{x}) - \overline{b}' + \overline{F},\ \sup(E\mathbf{x}) - \underline{b}' + \underline{F}] \tag{65}$$

holds for all $x \in \mathbf{x}$ and $G(x) \in \mathbf{G}(x)$.

Proof. That the linear system (1) is a linear relaxation of (62) follows directly from Theorem 6.1, with $p(x) := G(x)$, $\mathbf{c} := \mathbf{F}$, $h(x) := Ex$, and $\mathbf{d} := \mathbf{b}$ and for all $G(x) \in \mathbf{G}(x)$. By (51), every feasible point of (62) satisfies (1).
In addition to this $Ex \in E\mathbf{x}$ holds for all $x \in \mathbf{x}$ and by (1) $Ex \in \mathbf{b}$ also holds for all $x \in \mathbf{x}$ proving (64).
Since (64) is a linear relaxation of (62) by Theorem 7.1 the two-sided inequality (63) holds, implying that

$$G(x) \in [Ex - \overline{b}' + \overline{F},\ Ex - \underline{b}' + \underline{F}]. \tag{66}$$

Since $Ex \in E\mathbf{x}$ for all $x \in \mathbf{x}$, with (66) implies that

$$G(x) \in [\inf(E\mathbf{x}) - \overline{c} + \overline{F},\ \sup(E\mathbf{x}) - \underline{c} + \underline{F}]. \tag{67}$$

for all $x \in \mathbf{x}$ and for all $G(x) \in \mathbf{G}(x)$. From this, (65) follows since both $G(x) \in G(\mathbf{x})$, and $G(x) \in \mathbf{F}$ holds for all $x \in \mathbf{x}$ and for all $G(x) \in \mathbf{G}(x)$. □

If $G(x)$ is quadratic in x, the linear relaxations of (62) can be computed according to the results of the previous section. If for each constraint the quadratic terms are relaxed by the method of Kolev and the bilinear terms are eliminated by our approach, the resulting linear system (1) has m inequalities and n variables. If the approach of Lebbah & Rueher for the quadratic terms is combined with our approach for eliminating the bilinear terms, the resulting linear system has at most $m(3n+4)$ inequalities and $2n$ variables. The original method of Lebbah & Rueher results in a linear system of at most $m(7n + 4)$ inequalities and $3n$ variables. The methods discussed above can now be applied to the linear relaxation

in order to obtain tighter bounds on the variables. The following corollary shows how the relaxation and the new bounds on the variables can be used to tighten the bounds of the constraints of (62).
If we have tightened the bounds on the variables, the bounds of the relaxation can be tightened with (64). With (65) we may also tighten the bounds on the general constraints of the original constraint satisfaction problem.
The Gauss-Jordan preconditioner Algorithm 3.7 from Section 3 can be also applied to precondition a quadratic system. In GloptLab (see [7]), the quadratic constraints are represented as

$$Aq(x) \in \mathbf{F}, \quad x \in \mathbf{x}, \quad A \in \mathbf{A}, \tag{68}$$

where $A \in \mathbb{R}^{m \times n^2+n}$ is a (generally sparse) matrix, $\mathbf{A}$ represents the bounds for the constraint coefficients, $\mathbf{x}$ is n-dimensional and $\mathbf{F}$ is m-dimensional. The linear, quadratic, and bilinear monomials occurring in at least one of the constraints (but not the constant term) are collected into the $n^2 + n$ dimensional column vector

$$q(x) := (x_1, \dots, x_n,\ x_1^2, \dots, x_1x_n, \dots, x_nx_1, \dots, x_n^2)^T.$$

For this system the Gauss-Jordan preconditioner algorithm 3.7 can be applied.
All our methods can be applied after suitable preprocessing to arbitrary algebraic constraints. We can always transform a polynomial constraint to a collection of quadratic constraints by introducing explicit intermediate variables, and the same holds for constraints involving roots, provided that we also add nonnegativity constraints to the intermediate variables representing the roots. Rewriting an algebraic constraint satisfaction problem as an equivalent problem with linear and quadratic constraints only increases the number of variables, but allows one to apply the methods discussed in this paper. Of course, all techniques can be applied to the subset of quadratic (or algebraic) constraints in an arbitrary constraint satisfaction problem.
How the different techniques presented in this paper can be applied and combined to solve quadratic constraint satisfaction problems is visualized in Figure 3.

8 Examples

In this section two examples are given in order to demonstrate how the linearization techniques can be combined by filtering methods. The first example is a quadratic constraint satisfaction problem, to which linearization by Lebbah and Kolev are applied and then the arising linear system solved by both linear contraction (see, Section 4) and linear bounding (see, Section 5).

Example. 8.1 *Let*

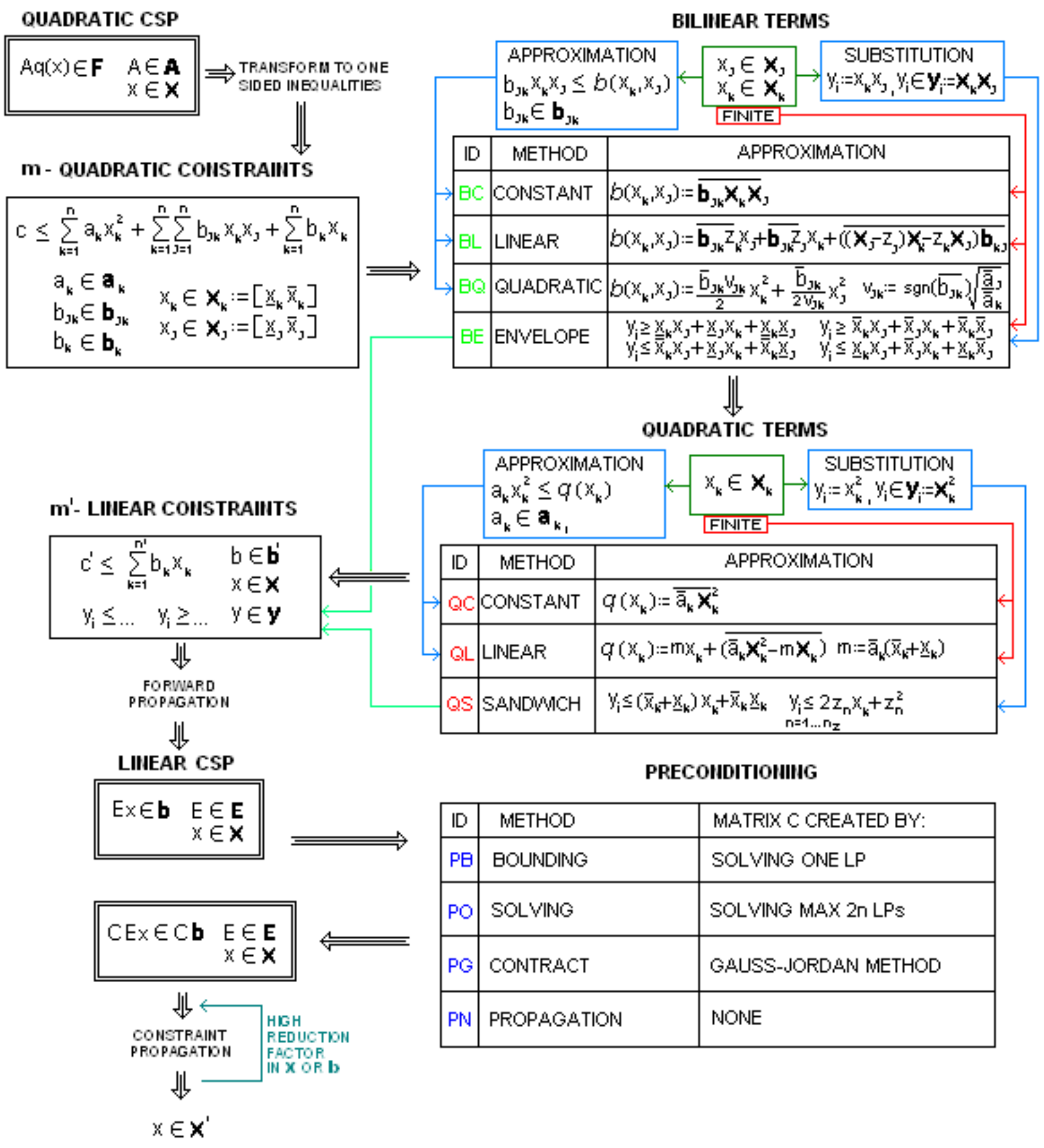
$$x_1^2 + x_2^2 \le 25, \quad x_1 \in \mathbf{x}_1 = [4, 5], \quad x_2 \in \mathbf{x}_2 = [0, 5]. \tag{69}$$

We linearize the quadratic expression (69)*. Since both quadratic terms x_1^2 and x_2^2 have positive coefficients we compute the tangents*

$$t(x_i) := mx_i + d, \quad t(x_i) \le x_i^2, \quad m = 2\tilde{x}_i, \quad d = \tilde{x}_i^2 - m\tilde{x}_i$$

for $i = 1, 2$ at the midpoints $\tilde{x}_1 = 4.5$ and $\tilde{x}_2 = 2.5$ of the intervals $\mathbf{x}_1$ and $\mathbf{x}_2$ obtaining

$$t(x_1) = 9x_1 - 20.25 \le x_1^2,\ t(x_2) = 5x_2 - 6.25 \le x_2^2. \tag{70}$$



$$c \le \sum_{k=1}^{n} a_k x_k^2 + \sum_{k=1}^{n}\sum_{j=1}^{n} b_{jk} x_k x_j + \sum_{k=1}^{n} b_k x_k$$
$a_k \in \mathbf{a}_k$, $b_{jk} \in \mathbf{b}_{jk}$, $b_k \in \mathbf{b}_k$, $x_k \in \mathbf{X}_k := [\underline{x}_k, \overline{x}_k]$, $x_j \in \mathbf{X}_j := [\underline{x}_j, \overline{x}_j]$

$b_{jk} x_k x_j \le b(x_k, x_j)$, $b_{jk} \in \mathbf{b}_{jk}$; $x_j \in \mathbf{X}_j$, $x_k \in \mathbf{X}_k$; $y_i := x_k x_j$, $y_i \in \mathbf{y}_i := \mathbf{X}_k \mathbf{X}_j$

ID	METHOD	APPROXIMATION
BC	CONSTANT	$b(x_k,x_j) := \overline{\mathbf{b}_{jk}\mathbf{X}_k\mathbf{X}_j}$
BL	LINEAR	$b(x_k,x_j) := \overline{\mathbf{b}_{jk}z_k}x_j + \overline{\mathbf{b}_{jk}z_j}x_k + \overline{((\mathbf{X}_j - z_j)\mathbf{X}_k - z_k\mathbf{X}_j)\mathbf{b}_{kj}}$
BQ	QUADRATIC	$b(x_k,x_j) := \frac{\overline{b}_{jk}v_{jk}}{2}x_k^2 + \frac{\overline{b}_{jk}}{2v_{jk}}x_j^2 \quad v_{jk} := \mathrm{sgn}(\overline{b_{jk}})\sqrt{\frac{\overline{a}_j}{\overline{a}_k}}$
BE	ENVELOPE	$y_i \ge \underline{x}_k x_j + \underline{x}_j x_k + \underline{x}_k \underline{x}_j \quad y_i \ge \overline{x}_k x_j + \overline{x}_j x_k + \overline{x}_k \overline{x}_j$ $y_i \le \overline{x}_k x_j + \underline{x}_j x_k + \overline{x}_k \underline{x}_j \quad y_i \le \underline{x}_k x_j + \overline{x}_j x_k + \underline{x}_k \overline{x}_j$

$a_k x_k^2 \le q(x_k)$, $a_k \in \mathbf{a}_k$; $x_k \in \mathbf{X}_k$; $y_i := x_k^2$, $y_i \in \mathbf{y}_i := \mathbf{X}_k^2$

ID	METHOD	APPROXIMATION
QC	CONSTANT	$q(x_k) := \overline{\overline{a}_k \mathbf{X}_k^2}$
QL	LINEAR	$q(x_k) := mx_k + \overline{(\overline{a}_k\mathbf{X}_k^2 - m\mathbf{X}_k)} \quad m := \overline{a}_k(\overline{x}_k + \underline{x}_k)$
QS	SANDWICH	$y_i \le (\overline{x}_k + \underline{x}_k)x_k + \overline{x}_k\underline{x}_k \quad \underset{n=1\dots n_z}{y_i \le 2z_n x_k + z_n^2}$

$$c' \le \sum_{k=1}^{n'} b_k x_k \quad b \in \mathbf{b}' \quad x \in \mathbf{X} \quad y_i \le \dots \quad y_i \ge \dots \quad y \in \mathbf{y}$$

ID	METHOD	MATRIX C CREATED BY:
PB	BOUNDING	SOLVING ONE LP
PO	SOLVING	SOLVING MAX 2n LPs
PG	CONTRACT	GAUSS-JORDAN METHOD
PN	PROPAGATION	NONE

Figure 3: Rigorous filtering using linear relaxations.

By **Lebbah's method** *we substitute the new variables* $x_3 \in \mathbf{x}_1^2$ *and* $x_4 \in \mathbf{x}_2^2$ *for the terms* x_1^2 *and* x_2^2 *and get the linear relaxation*

$$\begin{array}{l} x_3 + x_4 \le 25, \\ 9x_1 - x_3 \le 20.25 \\ 5x_2 - x_4 \le 6.25 \\ x_1 \in [4,5], \quad x_2 \in [0,5], \quad x_3 \in [16,25], \quad x_4 \in [0,25], \end{array} \tag{71}$$

of the constraint satisfaction problem (69).

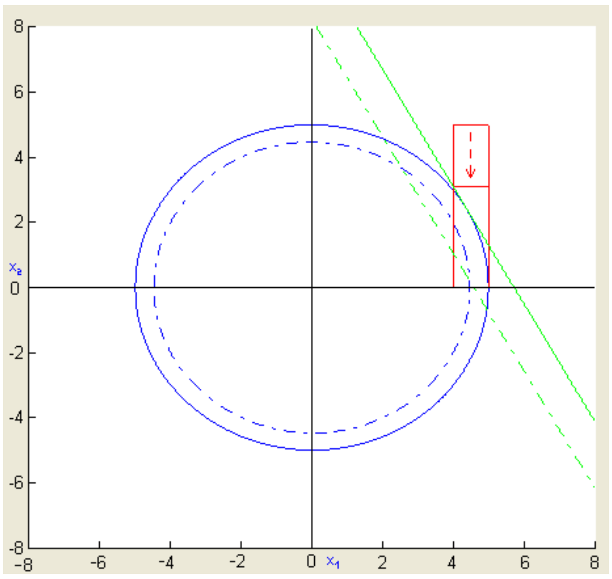


Figure 4: Example of solving a quadratic constraint satisfaction problem by linear relaxation. The arrow indicates the reduction of the bound $\overline{x}_2$.

Then we can use linear contraction to obtain tighter bounds on x_2: In the first step we have

$$\begin{array}{lcl} x_3 + x_4 \le 25,\ 16 \le x_3,\ 0 \le x_4 & \Rightarrow & x_3 \le 25,\ x_4 \le 9 \\ 9x_1 - x_3 \le 20.25,\ 4 \le x_1 & \Rightarrow & 15.75 \le x_3 \\ 5x_2 - x_4 \le 6.25,\ 0 \le x_2 & \Rightarrow & -6.25 \le x_4, \end{array}$$

getting improved bounds $x_4 \in [0, 9]$ and in the second step

$$5x_2 - x_4 \le 6.25,\quad -9 \le -x_4,\ 0 \le x_2 \quad \Rightarrow \quad x_2 \in [0, 3.05].$$

If we use the linear bounding and minimize x_1, $-x_1$, x_2, and $-x_2$ subject to the constraints (71) we obtain the approximate multiplier matrix Y which has all zero rows expect for the last row $Y_{4:} = \begin{pmatrix} 0.2 & 0 & 0.2 \end{pmatrix}$ corresponding to the objective $-x_2$. The matrix representation of (71) can be given as

$$Ex \ge c, \quad E = \begin{pmatrix} 0 & 0 & -1 & -1 \\ -9 & 0 & 1 & 0 \\ 0 & -5 & 0 & 1 \end{pmatrix}, \quad c = \begin{pmatrix} -25 & -20.25 & -6.25 \end{pmatrix}^T.$$

Since the first three rows of Y are zero, the first three of the inequalities $YE \ge Yc$ are trivial and the last one is

$$5x_2 + 0.2x_3 + 2.7 \cdot 10^{-17} x_4 \le 6.25$$

can be solved by substituting the lower bounds for x_3 and x_4, obtaining $x_2 \le 3.005$.

We use **Kolev's method** *to linearize the quadratic expression (69) by substituting (70) into it, obtaining*

$$9x_1 - 20.25 + 5x_2 - 6.25 \le x_1^2 + x_2^2 \le 25,$$

ending up in

$$9x_1 + 5x_2 \le 51.5 \quad x_1 \in [4, 5], \quad x_2 \in [0, 5].$$

From there a single step of linear contraction

$$9x_1 + 5x_2 \leq 51.5 \quad 4 \leq x_1 \quad \Rightarrow \quad x_2 \leq 3.1,$$

yields improved bounds on x_2.
If we use the linear bounding method we obtain the approximate multiplier matrix Y which has all zero rows expect for the last row $Y_{4:} = \begin{pmatrix} 0.2 & 1.8 \end{pmatrix}$ resulting in

$$x_2 \leq 10.3 - 1.8x_1 \leq 10.3 - 1.8\underline{x}_1 = 3.1.$$

The second example extends the first one by solving a simple system of separable constraint satisfaction problem.

Example. 8.2 *Let*

$$x_1^2 + x_1x_2 + x_2^2 \leq 25, \quad x_1 \in \mathbf{x}_1, \ \mathbf{x}_1 = [4,5], \quad x_2 \in \mathbf{x}_2, \ \mathbf{x}_2 = [0,5]. \tag{72}$$

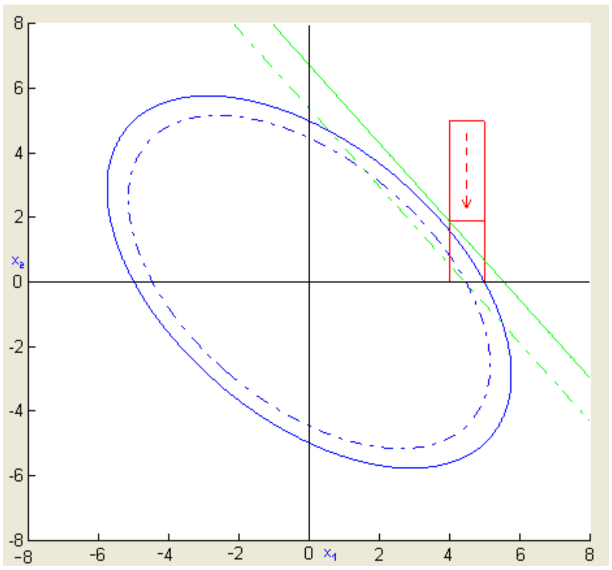


Figure 5: Example of solving a separable constraint satisfaction problem by linear relaxation. The arrow indicates the reduction of the bound $\overline{x}_2$.

By **Lebbah's method** *we compute the McCormick relaxations for the bilinear terms and approximate the quadratic terms as in the previous example, obtaining*

$$\begin{aligned}
&x3 + x_4 + x_5 \leq 25,\\
&4x_2 - x_3 \leq 0,\\
&5x_1 + 5x_2 - x_3 \leq 25,\\
&9x_1 - x_4 \leq 20.25,\\
&5x_2 - x_5 \leq 6.25,\\
&x_1 \in [4,5], \quad x_2 \in [0,5], \quad x_3 \in [0,25], \quad x_4 \in [16,25], \quad x_5 \in [0,25].
\end{aligned} \tag{73}$$

Solving (73) *with linear contraction results in $x_2 \leq 2.25$ while by solving with linear bounding we get $x_2 \leq 1.6944$.*

By **Kolev's method** *we first separate* (72) *by approximating the bilinear term* x_1x_2 *by linear terms obtaining*

$$x_1^2 + x_2^2 + 2.5x_1 + 4.5x_2 \leq 37.5,$$

then we approximate the quadratic terms as in the previous example, ending up in

$$11.5x_1 + 9.5x_2 \leq 64. \tag{74}$$

Solving (74) *with either linear contraction or with the linear bounding we get* $x_2 \leq 1.8947$.

9 Test Results

In this section we compare different linearization techniques. We use the following linear relaxation methods to reduce the bounds of the different test problems:

`identifier`	`bilinear`	`quadratic`
`LinCL`	constant	linear
`LinLL`	linear	linear
`LinQL`	quadratic	linear
`LinLN`	linear	new inequalities
`LinEL`	envelope	linear
`LinEN`	envelope	new inequalities

whereby the `bilinear` column describes the technique for approximating the bilinear terms and the `quadratic` column describes the technique for approximating the quadratic terms. After the linearization the three most promising variables are chosen, and linear solving are used tho reduce the bound constraints (for details see Section 5). Each method is applied to all test problems of a test set, one by one. If a method fails to reduce the bound constraints for some of the test problems, these will be solved again with the same method but with tighter bound constraints. With each retry the width of the bound constraints are reduced by 33% but the retries are counted and the solution times are summed up. The table below shows the average solution times (in seconds), the minimum, the average and the maximum number of retries as well as the gain:

$$g := \frac{1}{n}\sum_{i=1}^{n}\frac{\mathrm{wid}(\mathbf{x}'_i)}{\mathrm{wid}(\mathbf{x}_i)}$$

where $\mathbf{x}$ are the original bounds on the n variables, and $\mathbf{x}'$ the reduced bounds.
The first test consisted of 200 two dimensional, quadratic, randomly generated problems. Each problems had two equality constraints which intersected at least in the origin. The bound constraints for each variable were set between -10 and 10.

Linearization Test Results.

`dimension`	`n = 2`		
`method`	`time`	`retry`	`gain`
`LinCL`	0.136	[0 1.345 3]	0.127
`LinLL`	0.159	[0 1.345 3]	0.127
`LinQL`	0.110	[0 0.515 3]	0.187
`LinLN`	0.128	[0 0.305 3]	0.186
`LinEL`	0.132	[0 0.6 3]	0.201
`LinEN`	0.102	[0 0 0]	0.388

The second test consisted several quadratic, randomly generated problems with equality constraints which intersected at least in the origin. The test parameters and the test results are shown in the following tables.

Test case parameters.

name	probs	variables		constraints		
		#	bounds	#	relations	types
Test 1	20	2	$[-20, 20]$	2	equalities	ellipsoids
Test 2	20	5	$[-1, 1]$	5	equalities	ellipsoids
Test 3	20	10	$[-0.1, 0.1]$	10	equalities	ellipsoids

Linearization Test Results.

`dimen`	`n = 2`			`n = 5`			`n = 10`		
`method`	`time`	`retry`	`gain`	`time`	`retry`	`gain`	`time`	`retry`	`gain`
`LinCL`	0.134	[0 2.3 3]	0.107	0.260	[0 1 2]	0.019	0.185	[0 0 0]	0.054
`LinLL`	0.166	[0 2.3 3]	0.107	0.734	[0 1 2]	0.019	2.276	[0 0 0]	0.054
`LinQL`	0.137	[0 1.25 3]	0.174	0.287	[0 0.3 2]	0.040	0.511	[0 0 0]	0.070
`LinLN`	0.248	[0 1.65 3]	0.159	0.511	[0 0.3 1]	0.020	2.360	[0 0 0]	0.068
`LinEL`	0.140	[0 0.75 3]	0.225	0.193	[0 0 0]	0.095	0.568	[0 0 0]	0.153
`LinEN`	0.107	[0 0 0]	0.422	0.232	[0 0 0]	0.192	0.630	[0 0 0]	0.183

The third test consisted of 100 two dimensional and 100 ten dimensional, quadratic, randomly generated problems. Each problem had a single inequality constraint describing the boundary and the interior of an ellipsoid through the origin. The bound constraints were set to $[0, 3]$ for the two and $[0, 0.3]$ for the ten dimensional problems. No retries were allowed and the number of problems where the methods did not improve the bound constraints is listed in the column `no gain`. The column `gain2` is the average of the gain of all problems where the methods produced an improvement. Note that for the methods where new inequalities or envelopes were computed we use the linear solving method while the linear contraction for the other ones. This choice reflects some of our experience gained from experimenting with different combinations linearization and solution techniques.

Linearization Test Results.

`dimension`	`n = 2`				`n = 10`			
`method`	`time`	`gain`	`no gain`	`gain2`	`time`	`gain`	`no gain`	`gain2`
`LinearCLContract`	0.020	0.589	7	0.633	0.024	0.148	24	0.194
`LinearLLContract`	0.017	0.587	5	0.618	0.102	0.181	18	0.221
`LinearQLContract`	0.018	0.634	2	0.647	0.048	0.275	16	0.327
`LinearLNSolve`	0.085	0.720	1	0.728	0.459	0.211	14	0.245
`LinearELSolve`	0.083	0.665	4	0.693	0.647	0.286	11	0.321
`LinearENSolve`	0.075	0.845	1	0.853	0.685	0.341	11	0.384

As the test show introducing new variables instead of approximating the bilinear terms by constant linear or quadratic ones and the quadratic terms by constant or linear ones yields more gain but is slower even in low dimensions.

Acknowledgment

This research was supported through the research grant FSP 506/003 of the University of Vienna.

References

[1] F. Al-Khayyal. Jointly constrained biconvex programming and related problems: An overview. *Comput. Math. Appl*, 19:53–62, 1990.

[2] I. P. Androulakis, C. D. Maranas, and C. A. Floudas. αBB: a global optimization method for general constrained nonconvex problems. *Journal of Global Optimization*, 7:337–363, 1995.

[3] F. Benhamou, F. Goualard, L. Granvilliers, and J. F. Puget. Revising hull and box consistency. In *International Conference on Logic Programming*, pp. 230–244, 1999. URL `citeseer.ist.psu.edu/benhamou99revising.html`.

[4] F. Benhamou, D. McAllister, and P. Van Hentenryck. CLP(intervals) revisited. In *Proc. International Symposium on Logic Programming*, pp. 124–138. MIT Press, 1994.

[5] G. Chabert and L. Jaulin. Hull consistency under monotonicity. In *Principle and practices of constraint programming - CP2009*, pp. 188–195, 2009.

[6] J. Cruz and P. Barahona. Constraint reasoning in deep biomedical models. *Journal of Artificial Intelligence in Medicine*, 34:77–88, 2005. URL `http://ssdi.di.fct.unl.pt/~pb/papers/ludi_constraints.pdf`.

[7] F. Domes. GloptLab – a configurable framework for the rigorous global solution of quadratic constraint satisfaction problems. *Optimization Methods and Software*, 24: 727–747, 2009. URL `http://www.mat.univie.ac.at/~dferi/publ/Gloptlab.pdf`.

[8] F. Domes and A. Neumaier. A scaling algorithm for polynomial constraint satisfaction problems. *Journal of Global Optimization*, 43:327–345, 2008. URL `http://www.mat.univie.ac.at/~dferi/publ/Scaling.pdf`.

[9] F. Domes and A. Neumaier. Constraint propagation on quadratic constraints. *Constraints*, 15:404–429, 2010. ISSN 1383–7133. URL `http://www.mat.univie.ac.at/~dferi/publ/Propag.pdf`.

[10] J. Garloff, C. Jansson, and A. Smith. Lower bound functions for polynomials. *Journal of Computational and Applied Mathematics*, 157:207–225, 2003. URL `citeseer.ist.psu.edu/534450.html`.

[11] C. Grandon, D. Daney, and Y. Papegay. Combining CP and interval methods for solving the direct kinematic of a parallel robot under uncertainties. IntCP 06 Workshop, 2006. URL `ftp://ftp-sop.inria.fr/coprin/daney/articles/intcp06.pdf`.

[12] C. Jansson and S. M. Rump. Rigorous solution of linear programming problems with uncertain data. *ZOR Methods and Models of Operations Research*, 35:87111, 1991. URL `http://www.ti3.tu-harburg.de/paper/rump/JaRu91.pdf`.

[13] L. Jaulin. Interval constraints propagation techniques for the simultaneous localization and map building of an underwater robot, 2006. URL `http://www.mat.univie.ac.at/~neum/glopt/gicolag/talks/jaulin.pdf`.

[14] L. Jaulin, M. Kieffer, I. Braems, and E. Walter. Guaranteed nonlinear estimation using constraint propagation on sets. *International Journal of Control*, 74:1772–1782, 1999. URL `https://www.ensieta.fr/e3i2/Jaulin/observer.pdf`.

[15] L. V. Kolev. Automatic computation of a linear interval enclosure. *Reliable Computing*, 7(1):17–28, 2001.

[16] L. Krippahl and P. Barahona. PSICO: Solving protein structures with constraint programming and optimization. *Constraints*, 7:317–331, 2002. URL `http://ssdi.di.fct.unl.pt/~pb/papers/ludi_constraints.pdf`.

[17] Y. Lebbah. iCOs – Interval COnstraints Solver, 2003. URL `http://ylebbah.googlepages.com/icos`.

[18] Y. Lebbah, C. Michel, and M. Rueher. A rigorous global filtering algorithm for quadratic constraints. *Constraints*, 10:47–65, 2005. URL `http://ylebbah.googlepages.com/research`.

[19] G. McCormick. Computability of global solutions to factorable non-convex programs part I Convex underestimating problems. *Math. Prog.*, 10:147175, 1976.

[20] J-P. Merlet. Solving the forward kinematics of a Gough-type parallel manipulator with interval analysis. *International Journal of Robotics Research*, 23(3):221–235, 2004. URL `http://www-sop.inria.fr/coprin/equipe/merlet/Papers/IJRR2004.pdf`.

[21] T. S. Motzkin. *Beitrge zur Theorie der linearen Ungleichungen*. PhD thesis, Basel, Jerusalem, 1936.

[22] A. Neumaier. *Interval methods for systems of equations*, vol. 37 of *Encyclopedia of Mathematics and its Applications*. Cambridge Univ. Press, Cambridge, 1990.

[23] A. Neumaier. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica*, 1004:271–369, 2004.

[24] A. D. Rikun. A convex envelope formula for multilinear functions. *Journal of Global Optimization*, 10:425437, 1997.

[25] S. M. Rump. INTLAB – INTerval LABoratory, 1998 - 2008. URL `http://www.ti3.tu-harburg.de/~rump/intlab/`.

[26] N. V. Sahinidis and M. Tawarmalani. *BARON 7.2.5: global optimization of mixed-integer nonlinear programs,* User's Manual, 2005. URL `http://www.gams.com/dd/docs/solvers/baron.pdf`.

[27] H. Schichl, M. C. Markót, A. Neumaier, Xuan-Ha Vu, and C. Keil. The COCONUT Environment, 2000-2010. Software. URL `http://www.mat.univie.ac.at/coconut-environment`.

[28] H. Schichl and A. Neumaier. Interval Analysis on Directed Acyclic Graphs for Global Optimization. *Journal of Global Optimization*, 33(4):541–562, 2005.

[29] H. Sherali and W. Adams. *A Reformulation-Linearization Technique for Solving Discrete and Continuous Nonconvex Problems.* Kluwer Academic Publ., 1999.

[30] M. Tawarmalani and N. Sahinidis. Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. *Math. Program.*, 99(3), 2004.