Integral Approximation of Rays and Verification of Feasibility

Waltraud Huyer (`waltraud.huyer@univie .ac.at`)
and Arnold Neumaier (`arnold.neumaier@univie.ac.at`)
Institut für Mathematik, Universität Wien, Strudlhofgasse 4, A-1090 Wien, Austria

Abstract. An algorithm is presented that produces an integer vector nearly parallel to a given vector. The algorithm can be used to discover exact rational solutions of homogeneous or inhomogeneous linear systems of equations, given a sufficiently accurate approximate solution.

As an application, we show how to verify rigorously the feasibility of degenerate vertices of a linear program with integer coefficients, and how to recognize rigorously certain redundant linear constraints in a given system of linear equations and inequalities. This is a first step towards the handling of degeneracies and redundandies within rigorous global optimization codes.

Keywords: ray, integer rounding, rational approximation, rigorous verification of feasibility, constraint satisfaction, global optimization, degeneracy, redundant constraints

2000 MSC Classification: primary 90C30, secondary 62F30, 65G20

1. Introduction

In rigorous constrained global optimization algorithms (see, e.g., the books by Kearfott [4], Ratschek & Rokne [7], and van Hentenryck et al. [8]), it is necessary to verify that close to an approximate, putative optimal point there is a feasible point satisfying all constraints with certainty. To do this, the existence theory associated with Krawczyk's operator (derived in [5, Chapter 5] or [6, Section 6.3]) may be used if the number of equality constraints equals the number of variables; see also [8, p.134].

Hansen [3, Section 12.3.6] extended this way of proceeding to the case where the number of equality constraints is less than the number of variables, and the point to be verified is moved into the interior of the domain defined by the inequality constraints. In the book by Kearfott [4, Section 5.2.1], these ideas are discussed in detail, and practically useful recipes are given for achieving the verification.

However, no method exists so far to handle the case where the number of active constraints is larger than the number of variables. Indeed, from a generic point of view, this situation is degenerate, and standard analytic techniques for proving existence fail. While in certain applications degeneracies are very rare, they occur much more frequently in applications where a large number of constraints, many of them possibly redundant, are automatically generated. Thus techniques of handling such degeneracies are desirable. Looking at specific instances of such degeneracies in practice, one frequently finds that they are due to the presence of redundant constraints. Removing (sufficiently many) redundant constraints would resolve the degeneracy, but verification of redundancy with mathematical rigor is nontrivial.

Another difficulty may occur for problems where the feasible set has a codimension (dimension of space minus dimension of feasible set) larger than the number of equality constraints. In such cases, it is impossible to shift a point slightly to move it away from all bounds induced by inequality constraints.

For example, suppose that we want to minimize $x_1^2+x_2^2+x_3^2$ subject to the constraints

$$\begin{aligned} 5x_1+4x_2-2x_3 &\geq 5, \\ x_1-2x_2-4x_3 &\geq -5, \\ -3x_1-x_2+3x_3 &\geq 0. \end{aligned} \tag{1}$$

None of the presently used techniques for rigorous global optimization can handle this problem. Indeed, they cannot even verify a single feasible point. Thus they cover the unbounded feasible set (usually in a bounded search region only) by a huge number of tiny boxes, and get as upper bound of the optimal objective function value the value $+\infty$. (On the other hand, the lower bound can be brought close to the true optimal value.)

The optimum is $f^*=\frac{81}{46}$, attained at the point

$$x^*=\frac{1}{46}(-6,33,51)^T.$$

This point cannot be represented exactly in binary arithmetic. The computed approximation to the solution is therefore slightly inaccurate. However, it is impossible to move x slightly to obtain a point in which the inequalities are satisfied strictly, since the feasible set degenerated to the line consisting of all points of the form

$$x=(10t,-9t+0.6,7t+1.2)^T.$$

On the other hand, the constraints are linear, of the form $Ax\geq b$, with integral coefficients. Therefore, if we know the rational form $x^*=\frac{p}{q}$ with an integral vector p and an integer $q>0$, we can verify rigorously in (rounding error free) integer arithmetic whether $Ap-qb\geq 0$, which is equivalent to the original constraint. Thus we can establish the feasibility of the optimal point in the above example.

This allows one to solve the problem rigorously. Indeed, after having established feasibility, we can add the restriction

$$x_1^2+x_2^2+x_3^2\leq\frac{81}{46},$$

valid for any possible globally optimal point. Together with the original constraints, the region left consists now of a single point only, so that constraint propagation techniques [8] combined with branch and bound will eliminate everything outside an arbitrarily small neighborhood of the solution. (Whether this is efficient is a different question.)

We therefore consider the problem to find, for given $x\in\mathbb{R}^n$, a real number $q>0$ such that

$$qx_i\approx p_i\in\mathbb{Z},\quad i=1,\dots,n, \tag{2}$$

whose error is small in some sense. In the above application, q is an integer. But this problem may also have other applications in which q can be any real number. Indeed, originally, we derived the algorithm discussed below for applications in a statistical estimation context, where the aim is to find good 'simple' hyperplanes (defined by an integral normal vector) with useful statistical properties. Here a direction (i.e., a ray in n-space) normal to the hyperplane is estimated from the data, which are subject to noise. Therefore the ray is only inaccurately known. Since directions are only determined up to multiplication with a positive scalar factor, one can try to get a nice, distinguished direction approximating the given direction by rounding suitable multiples of the given vector to nearest integers. This is again a problem of the form (2).

The paper is arranged as follows. In Section 2 we discuss some measures to assess the quality of a rounded direction (2) and propose an algorithm to find a good rounded vector. Section 3 contains some numerical examples to demonstrate the applicability of our algorithm.

In Section 4 we present some applications. In particular, we show how to verify rigorously the feasibility of degenerate vertices of a linear program with integer coefficients, and how to recognize rigorously certain redundant linear constraints in a given system of linear equations and inequalities. This is a first step towards the handling of degeneracies and redundandies within rigorous global optimization codes.

In the following $n = \mathrm{round}(a)$ denotes the nearest integer to a, with $\mathrm{sign}(n) = \mathrm{sign}(a)$ and $|a| - \frac{1}{2} < |n| \le |a| + \frac{1}{2}$.

2. The algorithm

Let $x \in \mathbb{R}^n$, $q > 0$ and $p \in \mathbb{Z}^n$ with

$$p_i := \mathrm{round}(qx_i). \tag{3}$$

A natural measure for the quality of the rounded vector p is given by

$$f_1(q) := \|x - p/q\|, \tag{4}$$

where $\|\cdot\|$ denotes one of the usual monotone norms on $\mathbb{R}^n$ and p depends on q via (3). Of course, f_1 can be made arbitrarily small for sufficiently large q since each real number can be approximated arbitrarily well by a rational number with a sufficiently large denominator.

The objective function

$$f_2(q) := \|qx - p\| = qf_1(q) \tag{5}$$

is a measure of the error made by rounding qx. It favors small values of q, where little rounding errors are made but the rounded directions are not very accurate. Moreover, when the components x_i are of different magnitude, rounding a small or a large number by the same amount contributes equally to $f_2(q)$.

In order to have a good tradeoff between choosing a too large and a too small q, we therefore consider the function

$$f(q) := f_1(q)f_2(q) = \|qx - p\|^2/q. \tag{6}$$

For $0 < q < (2\max|x_i|)^{-1}$ we have $p_i = 0$, $i = 1, \dots, n$, and $f(q) = q\|x\|^2 \to 0$ as $q \to 0$. Since rounding all components to zero is undesirable, we have to minimize f over $q \ge (2\max(|x_i|))^{-1}$.

Assume that q^* is such that $q^*x \in \mathbb{Z}^n$. Then $kq^*x \in \mathbb{Z}^n$ and $f_1(kq^*) = f_2(kq^*) = f(kq^*) = 0$ for all $k \in \mathbb{N}$, i.e., the objective functions (4)–(6) do not distinguish between the different multiples of q^*, and due to rounding errors we might even get $f(kq^*) < f(q^*)$ for some $k > 1$.

Thus it is desirable to find an objective function that favors the minimal q^* yielding a good approximation and, moreover, takes into account how accurately a coordinate x_i is known. Let $\sigma_i > 0$ be a measure for the variability of x_i, i.e., assume that we actually only know that the value of the ith component of x is in $[x_i - \sigma_i, x_i + \sigma_i]$. Instead of (6) we then consider

$$f(q) := \|\max(|qx - p|, q\sigma)\|^2/q \tag{7}$$

where $\sigma := (\sigma_1, \ldots, \sigma_n)^T$ and the maximum is taken componentwise.

Other possibilities would be to use either $f(q) = (\|qx-p\|+q\|\sigma\|)^2/q$ or $f(q) = (\|qx-p\|^2+q^2\|\sigma\|^2)/q$, but the disadvantage of these objective functions is that only $\|\sigma\|$ enters and they therefore do not distinguish between components of σ having different magnitude.

PROPOSITION 1. *We have*

$$f(q) \geq q\|\sigma\|^2$$

with equality iff

$$|x - p/q| \leq \sigma.$$

Thus we expect to obtain $p^*/q^* \in [x - \sigma, x + \sigma]$ for the minimizer q^* of the objective function (7) and $p^* := \mathrm{round}(q^*x)$, which is just the desired accuracy.

COROLLARY 1. *Assume that we have already computed* $f(q_{best}) = f_{best}$. *Then for* $q > f_{best}/\|\sigma\|^2$ *we have*

$$f(q) > f_{best}.$$

Corollary 1 implies that $[0, f_{\mathrm{best}}/\|\sigma\|^2]$ contains the minimizer q^* of f, and this interval becomes more and more confined as smaller function values are found. (This contrasts with the function f_1 given by (4).)

Assume that $q^*x \in \mathbb{Z}^n$ for a $q^* > 0$. Then $f(kq^*) = kq^*\|\sigma\|^2 = kf(q^*)$ for all $k \in \mathbb{N}$, i.e., the objective function (7) favors the smallest such factor q^* as desired. However, if $\|\sigma\|$ is too large, q^* might not be the global minimizer of f; cf. Sections 3 and 4.

The algorithm given by the following MATLAB program proved to be useful for finding an approximate minimizer of f, where `inorm` $= 1$, 2 or `inf` is a parameter to choose the 1-, 2- or ∞-norm:

```
fbest = inf;
for fac = sort(1./abs(x))
  q = 0; k = 0;
  while q <= fbest/norm(sigma,inorm)^2
    k = k + 1;
    q = k*fac;
    fq = f(q);
    if fq < fbest, fbest = fq; end
  end
end
```

For $q = k/|x_i|$, $k \in \mathbb{N}$, we have $p_i = qx_i$, and the ith component of $\max(|qx - p|, q\sigma)$ becomes small. We expect the local minima to be near the points $q = k/|x_i|$ and the local maxima near $q = (k + \frac{1}{2})/|x_i|$, $k \in \mathbb{N}_0$, where $|p_i - qx_i| = \frac{1}{2}$ is maximal.

It is meaningful to consider not only the minimizer q^* obtained by the algorithm but to keep a set of q-values yielding the m smallest function values as alternative candidates.

3. Numerical examples

In the figures in this section a base 10 logarithmic scale is used for the y-axis and a linear scale for the x-axis.

Example 1. We chose the rational vector $x = (\frac{1}{5}, \frac{1}{4}, \frac{1}{3}, \frac{1}{2})^T$ and computed $f(q)$ given by (7) for the 2-, 1- and ∞-norm and $\sigma_i = \sigma_0$, $i = 1, \dots, 4$. In Figure 1 $f(q)$ is plotted against q for $\sigma_0 = 10^{-2}$. This choice of σ_0 turned out to be too small to find the least common denominator 60, i.e., the algorithm already stopped at much smaller q-values, and the global minimizers of f were $q^* = 15.68$ for the 2- and 1-norm and $q^* = 20.39$ for the ∞-norm.

The optimal q-values found by the algorithm described in Section 2 were $q^* = 24$ in all three cases. The results for $\sigma_0 = 10^{-3}$ are depicted in Figure 2. Now the global minimizer is $q^* = 59.89$, and the algorithm of Section 2 found $q^* = 60$ in all three cases.

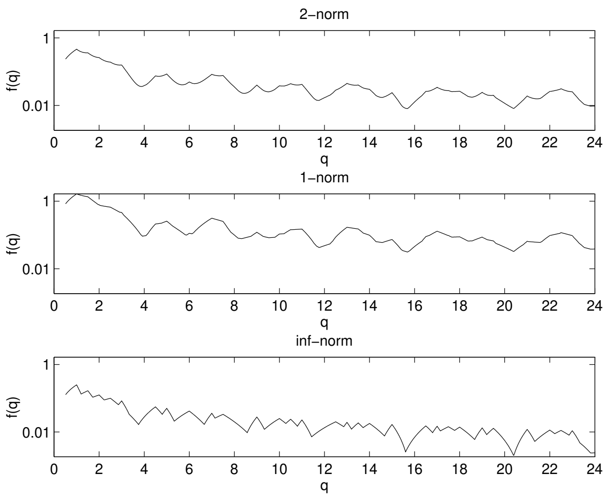

Figure 1. $f(q)$ for $\sigma_0 = 10^{-2}$ for Example 1

This example shows that the common denominator of a rational vector is not recovered if σ_0 is taken too small. Moreover, we see that the choice of the norm does not make so much difference concerning the minimizers. The functions $f(q)$ are rather similar for the 2- and the 1-norm, but for the ∞-norm we obtain additional break points and additional local minima and maxima. We therefore use the 2-norm in the remainder of this paper.

Example 2. A vector $x^* \in \mathbb{R}^{10}$ was generated by taking its entries from a uniform distribution on $[0, 1]$, and we applied our algorithm to x^* and eight random perturbations of x generated by $x = x^* + \sigma_0\varepsilon$, where the components of ε form a sample of a Gaussian distribution with mean 0 and variance 1, and took $\sigma_i = \sigma_0$, $i = 1, \dots, 10$, and $\sigma_0 = 10^{-3}$.

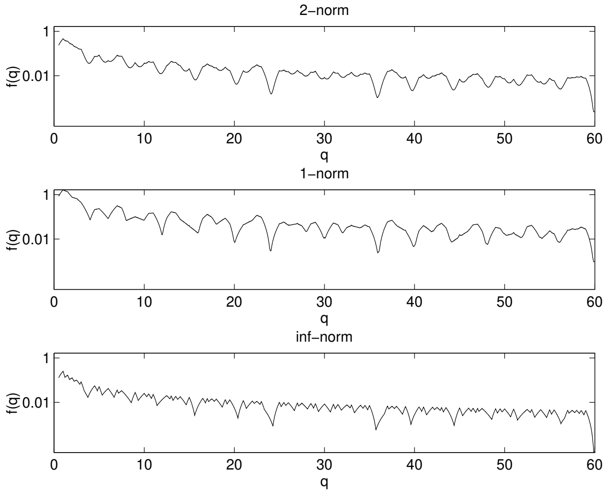

Figure 2. $f(q)$ for $\sigma_0 = 10^{-3}$ for Example 1

In Figure 3, $f(q)$ is shown for x^* and the eight perturbations, and we obtain 9 different global minimizers. However, some similarities are discernible, for example, a good local minimizer at $q \approx 65$, which was even a global minimizer for the eighth perturbation, where the algorithm already stopped at a q-value of about 135. In 7 of these 9 cases, the global minimizer roughly coincided with that obtained by computing f on a grid with step size $\Delta q = 0.01$.

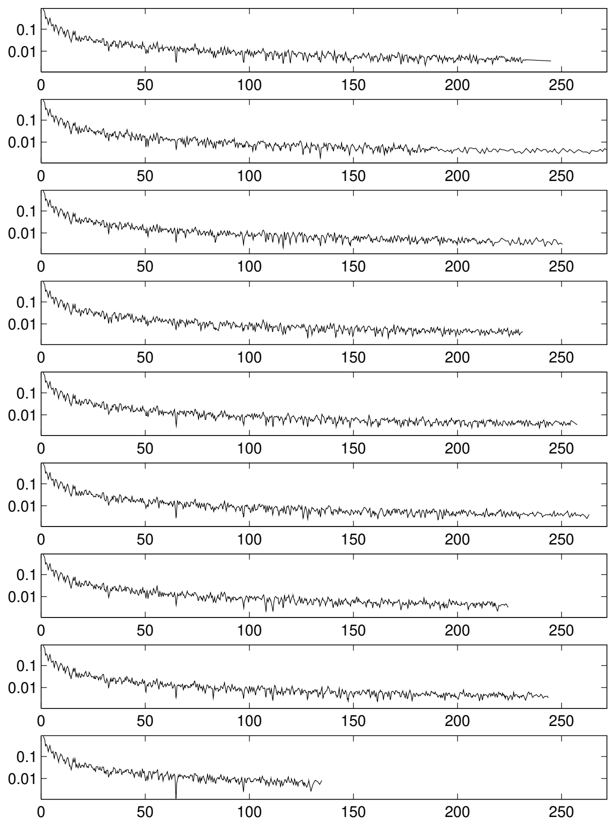

Figure 3. $f(q)$ for Example 2 for x^* and 8 random perturbations

Example 3. We chose a rational vector $x^* = 0.01p^*$, where $p^* = (22, 5, 68, 68, 93, 38, 52, 83, 3, 5)^T$, and generated 8 perturbations of x^* as in Example 2. The results are shown in Figure 4, and all functions have a clearly visible global minimizer at $q \approx 100$, recovering $p = p^*$.

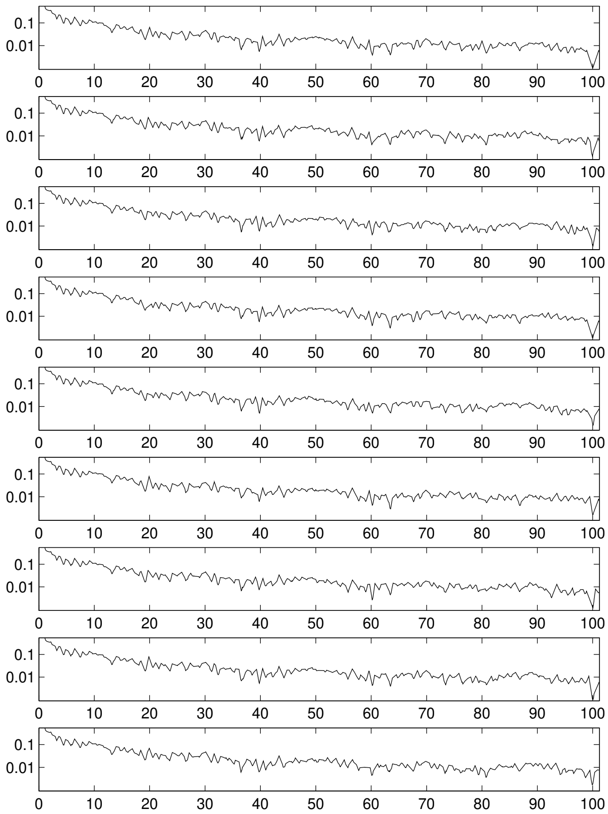

Figure 4. $f(q)$ for Example 3 for x^* and 8 random perturbations

4. Applications

We now apply the algorithm to some problems of numerical linear algebra for problems with integer coefficients.

The first problem is to determine whether a matrix with integer entries is singular. As is well-known, rounding errors in floating point calculations of the determinant usually result in a nonzero determinant even when A is singular; and since scaling of an $n \times n$ matrix by a uniform factor f changes the determinant by a factor f^n, the absolute value of the determinant says in higher dimensions very little about the closeness to singularity.

Therefore (see, e.g., GOLUB & VAN LOAN [2]) numerical (i.e., approximate) singularity is usually checked by means of a singular value decomposition. If the smallest singular value is tiny, there is a nearby singular matrix, and the corresponding singular vector x is an approximate null vector.

On the other hand, it is easily seen from an elimination procedure that an integral singular matrix always has an integral null vector. Checking the null vector property for an interger vector can be done error-free; thus we only need to find suitable candidates and test them.

By applying the rounding procedure of Section 2 to x (obtained with the MATLAB function `svd` in the two examples below) with $\sigma_i = \sigma_0$, $i = 1, \dots, n$, we may hope to find an exact null vector of A (if indeed A is singular).

As we have seen in Example 1, when a 'large' σ_0 is taken, the algorithm is rather quick, but it might stop before reaching the smallest q-value yielding a null vector with integer-valued entries. Only a sufficiently small σ_0 guarantees finding an integral null vector if there is any.

Example 4. Let

$$A = \begin{pmatrix} -5 & 2 & -2 & 2 & 0 \\ 7 & 6 & 1 & 2 & 0 \\ -1 & -1 & 1 & -1 & -4 \\ -7 & 3 & -2 & -1 & 8 \\ 3 & -5 & 1 & -1 & -2 \end{pmatrix}.$$

Our algorithm found the integer-valued null vector $x = (16, 9, -76, -45, -14)^T$ already for the rather large value $\sigma_0 = 10^{-3}$.

Example 5. Let

$$A = \begin{pmatrix} 3 & -3 & -5 & 5 & -1 & -1 & 4 & 0 & 3 & 2 \\ 1 & 1 & -1 & -4 & 0 & 0 & -2 & 1 & -1 & 0 \\ -2 & 2 & 0 & -2 & -1 & 0 & -1 & 1 & -2 & -1 \\ -1 & 3 & 1 & 0 & 0 & 3 & 0 & 2 & -2 & -2 \\ -2 & 0 & -2 & 0 & 1 & -3 & 2 & 4 & -1 & 1 \\ -6 & 7 & 3 & -4 & 1 & 4 & -3 & 3 & -6 & -1 \\ -2 & 0 & -1 & -2 & -6 & 2 & -3 & 0 & 1 & 0 \\ -2 & 2 & 1 & 0 & -3 & 2 & -3 & 1 & -3 & 1 \\ -5 & 3 & 0 & -3 & 3 & -1 & 2 & 2 & -2 & -2 \\ 2 & -2 & 4 & 3 & -1 & -3 & -1 & -4 & 3 & 0 \end{pmatrix}.$$

The null vector $x = (-1112, -229, -2309, 2125, 2130, -350, -4086, 299, 939, -2107)^T$ with rather large entries was not yet found for $\sigma_0 = 10^{-4}$ but was found for $\sigma_0 = 10^{-5}$.

In some applications, a rational approximation of a vector $x \in \mathbb{R}^n$ is needed. The algorithm described in this paper, however, does not always yield an integral q. In particular, this happens when the entries of x are rational but the numerators pertaining to the least common denominator contain a common divisor > 1. Therefore we consider the following extension of our algorithm. Let $p' \in \mathbb{Z}^n$ and $q' \in \mathbb{R}$ be determined by the algorithm described in Section 2 such that $q'x \approx p'$. With the aid of the MATLAB function `rats` we obtain a rational approximation r/s to q' and set $q := r$ and $p := sp'$. Then the vector p/q is the desired rational approximation to x. Our algorithm differs from MATLAB's `rats` in that a common denominator for all components is found. For example, let

$$x = (\pi, 2\pi, 3\pi, 4\pi, 5\pi, 6\pi, 7\pi, 8\pi, 9\pi, 10\pi),$$

where $\pi \approx 3.14159$ is the first positive zero of the sine function. Then `rats` gives the vector

$$\Big(\frac{355}{113}, \frac{333}{53}, \frac{1065}{113}, \frac{1420}{113}, \frac{1775}{113}, \frac{2130}{113}, \frac{2485}{113}, \frac{2840}{113}, \frac{1753}{62}, \frac{3550}{113}\Big),$$

which is no longer parallel to $(1, \ldots, 10)$. On the other hand, the algorithm described above with $\sigma_0 = 0.1$ yields

$$q = 113, \quad p = 355 \cdot (1, 2, 3, 4, 5, 6, 7, 8, 9, 10).$$

Our second application is finding a rational solution of a linear system of equations with integral coefficients. Consider the equation

$$Ax = b, \tag{8}$$

where $A \in \mathbb{Z}^{n \times n}$ is nonsingular and $b \in \mathbb{Z}^n$. By Cramer's rule, (8) possesses a solution $x \in \mathbb{Q}^n$ whose common denominator is a divisor of $|\det A|$, and we determine this rational solution by rounding suitable multiples of a computed approximation $x \approx A^{-1}b$ and checking whether (8) is satisfied. The same remarks on the choice of σ_i as above apply.

Example 6. Let

$$A = \begin{pmatrix} 6 & -2 & 3 & 0 & 6 \\ -4 & -1 & 0 & -3 & -9 \\ -3 & -1 & -6 & 0 & 1 \\ 3 & -9 & 5 & -8 & -8 \\ 7 & -2 & 4 & -2 & 5 \end{pmatrix}, \quad b = \begin{pmatrix} 3 \\ 1 \\ 0 \\ 4 \\ 0 \end{pmatrix}.$$

The solution

$$x = \frac{1}{175}(2363, -36, -1437, -1510, -1569)^T$$

of (8) was found for $\sigma_0 = 10^{-4}$.

Finally, we consider some problems in constraint satisfaction for linear constraints with integral coefficients.

Example 7. We return to the problem mentioned in the introduction of finding a feasible point near a floating point approximation $\tilde{x}$ of $x = \frac{1}{46}(-6, 33, 51)^T$ satisfying

$$\begin{aligned} 5x_1 + 4x_2 - 2x_3 &\geq 0, \\ x_1 - 2x_2 - 4x_3 &\geq -6, \\ -3x_1 - x_2 + 3x_3 &\geq 3. \end{aligned} \tag{9}$$

For the sake of demonstration, we take $\tilde{x}$ to be the decimal expansion of x^*, optimally rounded to e decimal digits. For $e \geq 7$ and $\sigma_0 = 10^{-2}$, the rational representation of x^* is recovered. For $e = 6$ and $\sigma_0 = 10^{-2}$, however, the vector $\frac{1}{200023}(-26090, 143495, 221765)^T$ is obtained. Thus, as to be expected, our method only works if the approximation it starts with is accurate enough.

Next we consider two linear programs

$$\min\{c^T x \mid Ax \geq b\} \tag{10}$$

with integer coefficients, where the optimal vertex x^* is degenerate. As mentioned in the introduction, there are two ways to proceed. Either we find a rational x (hopefully the true optimizer) close to an approximation $\tilde{x}$ of the optimal vertex and verify its feasibility in integer arithmetic. This is done in complete analogy to the Example 6.

Or we try to discover a redundant constraint, prove that it is redundant, and remove it from the system of linear inequalities, thus simplifying the problem. This requires that we represent one active constraint as a nonnegative linear combination of the remaining active constraints. But the latter is equivalent with finding a left null vector u of the extended matrix $E = [A_{K:}\ b_K]$, where K is the set of indices corresponding to active (or, in practice, nearly active) constraints (i.e., to (nearly) vanishing components of the residual $Ax - b$), such that u has exactly one negative entry (in the component corresponding to the redundant constraint). This is done in complete analogy to Examples 4 and 5.

Example 8. Let

$$A = \begin{pmatrix} -57 & -11 & -16 \\ -2 & 3 & 6 \\ -52 & -40 & -10 \\ -44 & 2 & 42 \\ 3 & 10 & 0 \\ 18 & 30 & 30 \end{pmatrix}, \quad b = \begin{pmatrix} 58 \\ -5 \\ 32 \\ 17 \\ 1 \\ -18 \end{pmatrix}, \quad c = \begin{pmatrix} -2 \\ -1 \\ 0 \end{pmatrix}.$$

The optimal solution is $x = \frac{1}{118}(-113, 71, -74)^T$, and given a good approximation to x, inspection of the residual shows that rows 1, 3, 4 and 6 are nearly active. This suggests a degenerate solution, which can be verified only if the exact rational representation of x can be retrieved from the computed approximation. For the sake of definiteness, we assume that the approximation is computed with MATLAB's \ as a least squares solution of the overdetermined linear system determined from the nearly active constraints,

$$\begin{pmatrix} -57 & -11 & -16 \\ -52 & -40 & -10 \\ -44 & 2 & 42 \\ 18 & 30 & 30 \end{pmatrix} x = \begin{pmatrix} 58 \\ 32 \\ 17 \\ -18 \end{pmatrix}. \tag{11}$$

Alternatively, one would like to discover a redundant constraint by finding a left null vector u of the matrix

$$E = \begin{pmatrix} -57 & -11 & -16 & 58 \\ -52 & -40 & -10 & 32 \\ 44 & -2 & -42 & 17 \\ 18 & 30 & 30 & -18 \end{pmatrix},$$

determined from (11) by the above recipe, with exactly one negative entry. For $\sigma_0 = 10^{-2}$, the algorithm gives neither a valid x nor a valid u. For $\sigma_0 = 10^{-3}$, a left null vector

$$u = (2, 19, -14, 27)^T$$

is obtained but not yet the correct rational representation of x. For $\sigma_0 = 10^{-4}$, the algorithm is successful for both x and u.

Example 9. Let

$$A = \begin{pmatrix} -54 & -16 & 23 \\ 27 & -4 & -61 \\ -60 & -42 & -38 \\ -52 & -26 & -8 \\ 59 & 42 & 4 \\ 49 & 12 & -17 \end{pmatrix}, \quad b = \begin{pmatrix} 35 \\ -91 \\ -58 \\ -13 \\ 9 \\ -27 \end{pmatrix}, \quad c = \begin{pmatrix} -11 \\ -54 \\ -154 \end{pmatrix}$$

Then the optimal solution is $x = \frac{1}{77}(-10, 22, 109)^T$ and the rows 1, 2, 3 and 6 are active. We proceed as in Example 8 and again, the algorithm fails for $\sigma_0 = 10^{-2}$. This time, for $\sigma_0 = 10^{-3}$, the correct rational representation of x is recovered but the algorithm is not successful in finding a valid u. For $\sigma_0 = 10^{-4}$, in addition to x, a left null vector $u = (250, 89, -60, 153)^T$ of

$$E = \begin{pmatrix} -54 & -16 & 23 & 35 \\ 27 & -4 & -61 & -91 \\ -60 & -42 & -38 & -58 \\ 49 & 12 & -17 & -27 \end{pmatrix}$$

is found.

In principle, the technique extends to polynomial constraints, but this will be considered elsewhere.

Acknowledgements

Part of this research was done within the COCONUT project [1], funded by the European Union under the project reference number IST-2000-26063.

References

1. *COCONUT, COntinuous CONstraints – Updating the Technology*, an IST Project funded by the European Union.
`http://www.mat.univie.ac.at/~neum/glopt/coconut/`
2. Golub, G. H. and van Loan, C. F.: *Matrix Computations*, 2nd ed., Johns Hopkins Univ. Press, Baltimore, 1989.
3. Hansen, R. E.: *Global Optimization using Interval Analysis*, Dekker, New York, 1992.
4. Kearfott, R. B.: *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht, 1996.
5. Neumaier, A.: *Interval Methods for Systems of Equations*, Cambridge Univ. Press, Cambridge, 1990.
6. Neumaier, A.: *Introduction to Numerical Analysis*, Cambridge Univ. Press, Cambridge, 2001.
7. Ratschek, H. and Rokne, J.: *New Computer Methods for Global Optimization*, Wiley, New York, 1988.
8. Van Hentenryck, P., Michel, L. and Deville, Y.: *Numerica. A Modeling Language for Global Optimization*, MIT Press, Cambridge, MA, 1997.