Journal of Global Optimization **7**: 115–125, 1995.
© 1995 *Kluwer Academic Publishers. Printed in the Netherlands.*

Computing Global Minima to Polynomial Optimization Problems Using Gröbner Bases

K. HÄGGLÖF, P. O. LINDBERG and L. SVENSSON
Department of Mathematics, Royal Institute of Technology, S - 100 44 Stockholm, Sweden.
e-mail:kroffa@math.kth.se, pol@math.kth.se and larss@math.kth.se

(Received: 2 Febr. 1995; accepted: 28 April 1995)

Abstract. The local optimality conditions to polynomial optimization problems are a set of polynomial equations (plus some inequality conditions). With the recent techniques of Gröbner bases one can find all solutions to such systems, and hence also find global optima. We give a short survey of these methods. We also apply them to a set of problems termed 'with exact solutions unknown' in the problem sets of Hock and Schittkowski. To these problems we give exact solutions.

Key words: Global optima, Gröbner bases, polynomial optimization problems.

1. Introduction

When the objective and the constraints of a mathematical programming problem are polynominal, then the necessary conditions for local optima is a system of polynomial equations (plus some nonnegativity conditions). All solutions to such systems can in fact be found (modulo space and time requirements) by use of recent techniques based on Gröbner bases, to be described below. Hence, by discarding solutions violating the nonnegativity conditions, one can find all local optima (plus some stationary points). By further evaluating the objective at these solutions, one can identify the global optima.

This application of Gröbner bases to local optimality conditions is straightforward, and we don't claim to be the first to note this. However, we want to draw the attention of the mathematical programming community (and the global optimizers in particular) to this possibility.

Thus, the current paper is a sort of tutorial introducing the optimizer to the use of Gröbner bases in this context.

In Section 2 we shortly review local optimality conditions, i.e. the Fritz–John and Karush–Kuhn–Tucker conditions, with special reference to Gröbner bases. Section 3 covers the relevant part of Gröbner base theory, without proofs though. It gives a concrete account of the computational machine, the Buchberger algorithm.

This account has enough detail, so that it can be applied by those who want to try. However, many algebraic programming systems, like Maple have routines for Gröbner base operations. Hence it is much easier to use those.

In Section 4 we apply these techniques to a specific example to show how it works.

In Appendix A we give global optima to optimization problems that are marked as practical problems (i.e. 'with exact solution unknown') in the problem sets of Hock & Schittkowski [4] and Schittkowski [6]. For some or all of these problems global optima may of course have been computed since 1981 or 1987 respectively.

The complete Maple code for the computations of appendix A can be found by world wide web at *www.optsyst.math.kth.se*. This makes it very easy for anyone to apply Gröbner bases to his/her own polynomial optimization problem.

2. Local Optima

Consider a standard mathematical programming problem :

$$\textbf{(P)}\left\{\begin{array}{rrl} \text{minimize} & f(\boldsymbol{x}) & \\ \text{s.t.} & g(\boldsymbol{x}) & \leq 0 \\ & h(\boldsymbol{x}) & = 0. \end{array}\right.$$

For this problem we have two classical optimality conditions, the Fritz–John conditions,

$$\textbf{(FJ)}\left\{\begin{array}{rl} \lambda_0\nabla f(\boldsymbol{x}) + \lambda_g\nabla g(\boldsymbol{x}) + \lambda_h\nabla h(\boldsymbol{x}) & = 0 \\ \lambda_g g(\boldsymbol{x}) & = 0 \\ h(\boldsymbol{x}) & = 0 \\ g(\boldsymbol{x}) & \leq 0 \\ \lambda_0, \lambda_g & \geq 0 \\ (\lambda_0, \lambda_g, \lambda_h) & \neq 0 \end{array}\right.$$

and the Karush–Kuhn–Tucker conditions, (corresponding to $\lambda_0 = 1$ in FJ),

$$\textbf{(KKT)}\left\{\begin{array}{rl} \nabla f(\boldsymbol{x}) + \lambda_g\nabla g(\boldsymbol{x}) + \lambda_h\nabla h(\boldsymbol{x}) & = 0 \\ \lambda_g g(\boldsymbol{x}) & = 0 \\ h(\boldsymbol{x}) & = 0 \\ g(\boldsymbol{x}) & \leq 0 \\ \lambda_g & \geq 0. \end{array}\right.$$

For the KKT conditions to be necessary for a local optimum, we further need some sort of constraint qualification. (For a thourough review of necessary conditions and constraint qualifications, see e.g. Bazaraa, Sherali & Shetty [1].)

In the early days of optimization, the FJ- and KKT-conditions were thought of as a means to solve optimization problems. Since it is usually extremely difficult to find all solutions to these systems, they have rather beeen used to 'verify' stationarity of candidate solutions arrived at by descent methods or similar.

However, in recent years there have appeared methods that can algebraically compute all solutions to polynomial systems, at least in principle. These methods are based on a concept called Gröbner bases. Given a polynomial system of equations, the central algorithm, the Buchberger algorithm, computes an equivalent system (or, if we use a more efficient, modified algorithm as described later, an equivalent set of systems). Under natural conditions, this new system is triangular in the way that it has one equation in only one variable, x_1 say; one or several equations in two variables, x_1 and x_2 say, etc. Using this one can first solve for x_1 in the first equation, getting all possible values of x_1. Then for each value of x_1, one can solve for x_2, and so on. Along the way one can check for nonnegativity and feasibility and terminate infeasible branches of the solution tree.

The drawback of this 'magic' method of course is that it is very time and space consuming. The time complexity grows as 2^{2^n} in the worst case, where n is the number of variables. However, it is 'only' 2^n in the case where there are finitely many solutions, which is the case we are interested in. (At present, there seems to exist no method to determine beforehand whether there are finitely many solutions. In fact, Gröbner bases would be the natural tool to find out.)

Still, one can often solve systems with a handfull of variables. When it works the results are beautiful, sometimes with closed form expressions of all solutions, as will be seen below.

These techniques give an alternative method to compute global optima to some problems, and which hence could be used for checking of other more numerical methods for global optimization. When there are closed form solutions, these can of course be computed numerically with arbitrary precision.

Finally note that the methods could also be applied to rational optimization problems as well, since these give rize to rational optimality conditions, which could be scaled to become polynomial. (At the cost of increasing the degrees of the polynomials involved, though.)

3. Gröbner Bases

In this section we give a brief introduction to the theory of Gröbner bases (for more details, see [2]). The theory was developed 1965 by Bruno Buchberger (W. Gröbner was his thesis advisor at the University of Innsbruch, Austria) and is roughly an algorithm that replaces a system of polynomial equations in several variables with a new system which has the same solutions but is easier to study.

The computation of Gröbner bases generalizes the Euclidian algorithm, Gaussian elimination and the Sylvester resultant.

The algorithm is implemented in many algebraic systems, e.g., Maple, Reduce, Scratchpad II, Macsyma and Macaulay.

We give below a short account of this area, giving all necessary definitions and results relevant for application to optimality conditions.

NOTATION. If $\boldsymbol{x} = (x_1, \ldots, x_n)$ is a list of formal variables, then

$$\boldsymbol{x}^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \ldots x_n^{\alpha_n}, \quad \alpha_i \in \{0, 1, 2 \ldots\} = \mathbb{N}$$

is called a *monomial* of *degree* $\mid \alpha \mid = \alpha_1 + \ldots + \alpha_n$. The set of monomials in $\boldsymbol{x}$ is denoted by $\mathrm{Mon}(\boldsymbol{x})$.

Let K be a *field* $(\mathbb{Q}, \mathbb{R}, \mathbb{C}, \ldots,)$. Finite sums like $\Sigma f_\alpha \boldsymbol{x}^\alpha$, where $f_\alpha \in K$, are called *polynomials* in $\boldsymbol{x}$ over K. The set of polynomials in $\boldsymbol{x}$ over K, we denote by $K[\boldsymbol{x}]$ (or $K[x_1, \ldots, x_n]$). An *ideal* I is a subset of $K[\boldsymbol{x}]$ such that

(i) $f, g \in I \Rightarrow f + g \in I$

(ii) $f \in K[\boldsymbol{x}]$ and $g \in I \Rightarrow fg \in I$.

If $G \subset K[\boldsymbol{x}]$, the smallest ideal containing G, denoted by $\langle G \rangle$, is

$$\langle G \rangle = \left\{ \sum_{\text{finite}} fg : f \in K[\boldsymbol{x}], g \in G \right\}.$$

Hilbert's basis theorem [2, Theorem 5.4, p. 75] states that every ideal I in $K[\boldsymbol{x}]$ is *finitely generated*, that is $I = \langle G \rangle$ for some finite set G in $K[\boldsymbol{x}]$.

DEFINITION. A *total ordering* $<$ *onMon*$(\boldsymbol{x})$ is called *admissible* if

(i) $1 < \boldsymbol{x}^\alpha \quad \forall \alpha \neq 0$

(ii) $\boldsymbol{x}^\alpha < \boldsymbol{x}^\beta \Rightarrow \boldsymbol{x}^\alpha \boldsymbol{x}^\gamma < \boldsymbol{x}^\beta \boldsymbol{x}^\gamma \quad \forall \alpha, \beta, \gamma \mathrm{in} \mathbb{N}^n$.

EXAMPLES OF ADMISSIBLE ORDERINGS

Lexicographic order with $x_1 > x_2 > \cdots > x_n$ is defined by: $\boldsymbol{x}^\alpha < \boldsymbol{x}^\beta$ if there exists some i such that $\alpha_i < \beta_i$ and $\alpha_j = \beta_j \; \forall j < i$.

Degree lexicographic order with $x_1 > x_2 > \cdots > x_n$ is defined by: $\boldsymbol{x}^\alpha < \boldsymbol{x}^\beta$ if $\mid \alpha \mid < \mid \beta \mid$ or if $\mid \alpha \mid = \mid \beta \mid$ and $\boldsymbol{x}^\alpha < \boldsymbol{x}^\beta$ lexicographically.

Degree reverse lexicographic order with $x_1 > x_2 > \cdots > x_n$ is defined by: $\boldsymbol{x}^\alpha < \boldsymbol{x}^\beta$ if $\mid \alpha \mid < \mid \beta \mid$ or if $\mid \alpha \mid = \mid \beta \mid$ and $\alpha_i > \beta_i$ for some i such that $\alpha_j = \beta_j \; \forall j > i$.

If $f(\boldsymbol{x}) = \Sigma f_\alpha \boldsymbol{x}^\alpha \in K[\boldsymbol{x}]$ we define the *leading monomial* as $\mathrm{lm}(f) = \boldsymbol{x}^\alpha$ if $(\boldsymbol{x}^\alpha < \boldsymbol{x}^\beta \Rightarrow f_\beta = 0)$ and $f_\alpha \neq 0$, and the *leading coefficient* as $\mathrm{lc}(f) = f_\alpha$ if $\mathrm{lm}(f) = \boldsymbol{x}^\alpha$.

Given polynomials f and g we write

$$f \rightarrow_g h$$

if $h = f - cx^\gamma g$ for some c in K and γ in $\mathbb{N}^n$ such that

$$x^\gamma \mathrm{lm}(g) \notin \mathrm{mon}(h) = \{x^\alpha : h_\alpha \neq 0\}.$$

For a subset G of $K[x]$ we write

$$f \rightarrow_G h$$

if $f \rightarrow_g h$ for some g in G. We say that f is *reduced* to h *modulo* G.

EXAMPLE. $G = \{x_1^2 + x_2^2 - 1,\ 3x_1x_2 - 1\} \subseteq \mathbb{R}[x_1, x_2]$ using lexicographic order we have $\mathrm{lm}(x_1^2 + x_2^2 - 1) = x_1^2$, $\mathrm{lm}(3x_1x_2 - 1) = x_1x_2$. If $f = 6x_1^3x_2 + x_1$, then $\mathrm{mon}(f) = \{x_1^3x_2, x_1\}$.

$$\begin{aligned} f &= 6x_1^3x_2 + x_1 \rightarrow_G 6x_1^3x_2 + x_1 - 6x_1x_2(x_1^2 + x_2^2 - 1) \\ &= x_1 - 6x_1x_2^3 + 6x_1x_2 = h. \end{aligned}$$

But h can also be reduced with G

$$h \rightarrow_G -6x_1x_2^3 + 6x_1x_2 + x_1 + 2x_2^2(3x_1x_2 - 1) = 6x_1x_2 + x_1 - 2x_2^2 = h_1$$

$$h_1 \rightarrow_G 6x_1x_2 + x_1 - 2x_2^2 - 2(3x_1x_2 - 1) = x_1 - 2x_2^2 + 2 = h_2$$

Now h_2 cannot be reduced with G and we write $h_2 \not\rightarrow_G$.

DEFINITION. We write $f \rightarrow_G^* h$ if there exist polynomials h_i such that

$$f \rightarrow_G h_1 \rightarrow_G h_2 \rightarrow \cdots \rightarrow_G h.$$

If moreover $h \not\rightarrow_G$ we say that h is a *normal form* of f with respect to G.

DEFINITION. The set of normal forms of f is denoted by $NF(f, G)$.

It can be shown that every reduction chain $f \rightarrow_G h_1 \rightarrow_G h_2 \rightarrow \cdots$ is stationary, that is $h_k \not\rightarrow_G$ for some k.

Hence $NF(f, G)$ is nonempty $\forall f, G$.

DEFINITION. G is called a *Gröbner basis* (or *Gröbner base*) if every polynomial has a unique normal form with respect to G (and the underlying admissible monomial ordering $<$)

$$\text{i.e.,} \quad |\,NF(f,G)\,| = 1 \quad \forall f \in K[x].$$

The above definition would have been of no use if we couldn't find an algorithm that to a given finite set F of polynomials computed a Gröbner basis G such that $\langle F\rangle = \langle G\rangle$. Such a G is called a *Gröbner basis for* F.

The most central and original definition towards such an algorithm is the concept of *S-polynomial.*

Let $f, g \in K[\boldsymbol{x}]$ and choose multiindices α and β such that $\boldsymbol{x}^{\alpha}\mathrm{lm(f)} = \boldsymbol{x}^{\beta}\mathrm{lm(g)} = \mathrm{LCM}(\mathrm{lm(f)}, \mathrm{lm(g)})$. ($LMC$ denotes the least common multiple.) Now define the *S-polynomial*

$$\mathrm{Spol}(f, g) = \boldsymbol{x}^{\alpha}\frac{f}{\mathrm{lc(f)}} - \boldsymbol{x}^{\beta}\frac{g}{\mathrm{lc(g)}}.$$

We can now state the most essential result in the theory of Gröbner bases.

THEOREM 1 (Buchberger). *A finite set G of polynomials is a Gröbner basis iff*

$$\mathrm{Spol(f,g)} \rightarrow^{*}_{\mathrm{G}} 0 \quad \forall \mathrm{f,g} \in \mathrm{G}.$$

COROLLARY 2 (Buchberger's algorithm). *The following algorithm computes a Gröbner basis G for a given finite set F of polynomials.*

```
$G := F$;
$B := \{(f_1, f_2) \in G \times G : f_1 \neq f_2\}$;
while $B \neq \emptyset$
  choose $(f, g)$ in $B$;
  $B := B \setminus \{(f, g)\}$;
  $\mathrm{Spol(f,g)} \rightarrow^{*}_{\mathrm{G}} \mathrm{h} \not\rightarrow_{\mathrm{G}}$;
  if $h \neq 0$ then
    $B := B \cup (G \times \{h\})$;
    $G := G \cup \{h\}$;
  end;
end;
```

REMARK. By removing all polynomials g in G which are reducible modulo $G \backslash g$, we get a *reduced* Gröbner basis which is unique if all leading coefficients are one.

Some Applications of Gröbner Bases

Let G be a Gröbner basis for F, with respect to some admissible ordering on monomials.

THEOREM 3 (Ideal membership). *Let G be a Gröbner base for $F \subseteq K[\boldsymbol{x}]$. A polynomial f in $K[\boldsymbol{x}]$ belongs to the ideal $\langle F \rangle$ iff f can be reduced to zero modulo G i.e.,*

$$f \in \langle F \rangle \leftrightarrow f \rightarrow^{*}_{G} 0.$$

THEOREM 4 (Elimination). *Given $F \subseteq K[\boldsymbol{x}, \boldsymbol{y}] = K[x_1 \cdots x_m, y_1 \cdots y_n]$ and an admissible ordering on* $\mathrm{Mon}(\boldsymbol{x}, \boldsymbol{y})$ *such that $\boldsymbol{x}^\alpha > \boldsymbol{y}^\beta$ $\forall \alpha \neq 0$. Then if G is a Gröbner basis for F, $G \cap K[\boldsymbol{y}]$ is a Gröbner basis for the elimination ideal $\langle F \rangle \cap K[\boldsymbol{y}]$.*

REMARK. If lexicographical order $x_1 > x_2 > \cdots > x_n$ is used, then it follows from the above theorem that the Gröbner basis is in a sort of triangular form. For instance, if there are only finitely many zeroes of F (in the algebraic closure of K, i.e. in $\mathbb{C}$ if $K = \mathbb{Q}$ or $\mathbb{R}$), then the Gröbner basis will have the following structure.

(i) It contains one polynomial $g_n(x_n)$ in $K[x_n]$

(ii) It contains one or several polynomials in $K[x_{n-1}, x_n]$, $g_{n-1,1}(x_{n-1}, x_n)$, $g_{n-1,2}(x_{n-1}, x_n), \ldots$ and so on.

Thus, to solve the original equation system we first solve for x_n in $g_n(x_n) = 0$, getting the solutions $\hat{x}_{n,1}, \hat{x}_{n,2}, \ldots$. Then, for fixed $\hat{x}_n$ we find the common roots of $g_{n-1,1}, g_{n-1,2}, \ldots$ getting solutions $\hat{x}_{n-1,1}, \hat{x}_{n-1,2}, \ldots$, and so on. In this way, we can compute all solutions to our equation system. Also, note that it is easy to eliminate e.g. negative solutions on the way.

The gröbner bases of nontrivial systems often contain polynomials with very large coeffients, say integers of the order 10^{30}. This of course slows down the computations and increases the memory requirements. One way to try to remedy this, is to try to factor the polynomials generated by Buchberger's algoritm. The system then splits into several systems, one for each factor. Then Buchberger's algoritm is applied to each of the systems, and so on. This approach in fact is the one used by the Maple routine `grobner[gsolve]`, which we have used in our computations. For more details, see [3].

4. Example

We choose a simple 'practical' problem, i.e. one 'with exact solutions unknown', from the the problem set [6]. To this problem, no 337, we apply the results above, in the form of the Gröbner base routines of Maple.

$$
\textbf{(P)} \left\{ \begin{array}{rrcl} \text{minimize} & f(\boldsymbol{x}) & = & 9x_1^2 + x_2^2 + 9x_3^2 \\ \text{s.t.} & g_1(\boldsymbol{x}) & = & 1 - x_1 x_2 \leq 0 \\ & g_2(\boldsymbol{x}) & = & 1 - x_2 \leq 0 \\ & g_3(\boldsymbol{x}) & = & x_3 - 1 \leq 0. \end{array} \right.
$$

For space reasons we will use the KKT-conditions rather than the FJ-conditions. Thus some stationary points not fulfilling a constraint qualifications might escape. The application of the FJ-conditions is totally parallel, but more voluminous. In the Maple code found at *www.optsyst.math.kth.se*, we give both options. This particular example however yields the same solution with the use of FJ-conditions and it hence is the global minimum.

$$\text{(KKT)}\left\{\begin{aligned} 18x_1 - \lambda_1 x_2 &= 0\\ 2x_2 - \lambda_1 x_1 - \lambda_2 &= 0\\ 18x_3 + \lambda_3 &= 0\\ \lambda_1(1 - x_1 x_2) &= 0\\ \lambda_2(1 - x_2) &= 0\\ \lambda_3(x_3 - 1) &= 0. \end{aligned}\right.$$

We use lexiographic order with $\boldsymbol{x} > \boldsymbol{\lambda}$. In this way we will get a single variable equation in λ_3 and so on. Thus, we can break early on nonnegativity constraints. Using the Maple routine `grobner[gbasis]`, we get the Gröbner base of this system. The corresponding equation system is:

$$(1)\left\{\begin{aligned} 18x_1 - \lambda_1 x_2 &= 0\\ 2x_2 - \lambda_1 - \lambda_2 &= 0\\ -36x_2 + 18\lambda_2 + \lambda_1 x_2 &= 0\\ -\lambda_2 + \lambda_2 x_2 &= 0\\ 18x_3 + \lambda_3 &= 0\\ \lambda_1^3 - 18\lambda_2^2 + 36\lambda_2 - 36\lambda_1 &= 0\\ -2\lambda_2 + \lambda_2^2 + \lambda_1\lambda_2 &= 0\\ \lambda_2^3 + 14\lambda_2^2 - 32\lambda_2 &= 0\\ 18\lambda_3 + \lambda_3^2 &= 0. \end{aligned}\right.$$

Using instead `grobner[gsolve]`, which uses the more efficient algorithm described briefly in the remark to theorem 4, the system above is reduced to a product of the following systems:

$$(1)\left\{\begin{aligned} x_1 &= 0\\ x_2 - 1 &= 0\\ x_3 - 1 &= 0\\ \lambda_1 &= 0\\ \lambda_2 - 2 &= 0\\ \lambda_3 + 18 &= 0 \end{aligned}\right.
(2)\left\{\begin{aligned} x_1 - 1 &= 0\\ x_2 - 1 &= 0\\ x_3 - 1 &= 0\\ \lambda_1 - 18 &= 0\\ \lambda_2 + 16 &= 0\\ \lambda_3 + 18 &= 0 \end{aligned}\right.
(3)\left\{\begin{aligned} 3x_1 - x_2 &= 0\\ x_2^2 - 3 &= 0\\ x_3 - 1 &= 0\\ \lambda_1 - 6 &= 0\\ \lambda_2 &= 0\\ \lambda_3 + 18 &= 0 \end{aligned}\right.
(4)\left\{\begin{aligned} 3x_1 + x_2 &= 0\\ x_2^2 + 3 &= 0\\ x_3 - 1 &= 0\\ \lambda_1 + 6 &= 0\\ \lambda_2 &= 0\\ \lambda_3 + 18 &= 0 \end{aligned}\right.$$

$$(5)\begin{cases} x_1 = 0 \\ x_2 = 0 \\ x_3 - 1 = 0 \\ \lambda_1 = 0 \\ \lambda_2 = 0 \\ \lambda_3 + 18 = 0 \end{cases} (6)\begin{cases} x_1 = 0 \\ x_2 - 1 = 0 \\ x_3 = 0 \\ \lambda_1 = 0 \\ \lambda_2 - 2 = 0 \\ \lambda_3 = 0 \end{cases} (7)\begin{cases} x_1 - 1 = 0 \\ x_2 - 1 = 0 \\ x_3 = 0 \\ \lambda_1 - 18 = 0 \\ \lambda_2 + 16 = 0 \\ \lambda_3 = 0 \end{cases} (8)\begin{cases} 3x_1 - x_2 = 0 \\ x_2^2 - 3 = 0 \\ x_3 = 0 \\ \lambda_1 - 6 = 0 \\ \lambda_2 = 0 \\ \lambda_3 = 0 \end{cases}$$

$$(9)\begin{cases} 3x_1 + x_2 = 0 \\ x_2^2 + 3 = 0 \\ x_3 = 0 \\ \lambda_1 + 6 = 0 \\ \lambda_2 = 0 \\ \lambda_3 = 0 \end{cases} (10)\begin{cases} x_1 = 0 \\ x_2 = 0 \\ x_3 = 0 \\ \lambda_1 = 0 \\ \lambda_2 = 0 \\ \lambda_3 = 0 \end{cases}$$

These ten systems are then solved separately, checking bounds and KKT-conditions along the way. In all cases, we recursively find the last one-variable equation, solve it and substitute its solution(s) in the remaining equations. Systems (1)–(5), (7) and (9) are terminated due to violation of nonnegativity conditions. The remaining systems hence gives the following points:

$$\begin{aligned} \boldsymbol{x}^{(6)} &= (0,1,0), & \boldsymbol{\lambda}^{(6)} &= (0,2,0) \\ \boldsymbol{x}^{(8)} &= (\tfrac{1}{\sqrt{3}},\sqrt{3},0), & \boldsymbol{\lambda}^{(8)} &= (6,0,0) \\ \boldsymbol{x}^{(10)} &= (0,0,0), & \boldsymbol{\lambda}^{(10)} &= (0,0,0). \end{aligned}$$

Checking these points with the remaining Kuhn–Tucker condition, $g(\boldsymbol{x}) \leq 0$, only one point remains : $\boldsymbol{x}^{(8)} = (\frac{1}{\sqrt{3}},\sqrt{3},0)$, $\boldsymbol{\lambda}^{(8)} = (6,0,0)$.

Computing the objective function value we have the following global minimum point (due to the fact that the FJ-conditions yield the same solution as mentioned above),

$$\begin{aligned} \boldsymbol{x}^* &= (\tfrac{1}{\sqrt{3}},\sqrt{3},0) \text{ or approximately,} & \boldsymbol{x}^* &= (0.57735027, 1.7320508, 0) \\ f(\boldsymbol{x}^*) &= 6 & f(\boldsymbol{x}^*) &= 6. \end{aligned}$$

5. Comments

The Gröbner base computations are space and time consuming. There are ways to partially circumvent this. Usually, the computation of Gröbner bases using degree reverse lexicographic ordering is faster than using lexicographic ordering. But, to get the triangulation according to the remark to Theorem 4, we need lexicographic ordering. However, one can speed up the process substantially by first computing a Gröbner base G using degree reverse lexiographic ordering, and then use that one to find polynomials in one variable in $\langle G\rangle$. These can then be used as above. For a more thorough coverage of such possibilities, see [5].

Problem	Solution	Optimal value
76 [4]	$(\frac{3}{11}, \frac{23}{11}, 0, \frac{6}{11})$	$-\frac{103}{22}$
315 [6]	$(\frac{3}{5}, \frac{4}{5})$	$-\frac{4}{5}$
316 [6]	$(5\sqrt{2}, -5\sqrt{2})$	$900 - 400\sqrt{2}$
317 [6]	RO	≈ 372.4666057
318 [6]	RO	≈ 412.7500540
319 [6]	RO	≈ 452.4043956
320 [6]	RO	≈ 485.5314623
323 [6]	$((\frac{1}{2} + \frac{1}{4}\sqrt{6})^{\frac{1}{3}} - \frac{1}{2(\frac{1}{2}+\frac{1}{4}\sqrt{6})^{\frac{1}{3}}},$ $(\frac{1}{2} + \frac{1}{4}\sqrt{6})^{\frac{2}{3}} + \frac{1}{4(\frac{1}{2}+\frac{1}{4}\sqrt{6})^{\frac{2}{3}}})$	$((\frac{1}{2} + \frac{1}{4}\sqrt{6})^{\frac{1}{3}} - \frac{1}{2(\frac{1}{2}+\frac{1}{4}\sqrt{6})^{\frac{1}{3}}})^2$ $+((\frac{1}{2} + \frac{1}{4}\sqrt{6})^{\frac{2}{3}} + \frac{1}{4(\frac{1}{2}+\frac{1}{4}\sqrt{6})^{\frac{2}{3}}})^2$ $-4((\frac{1}{2} + \frac{1}{4}\sqrt{6})^{\frac{1}{3}} - \frac{2}{(\frac{1}{2}+\frac{1}{4}\sqrt{6})^{\frac{1}{3}}}) + 4$
324 [6]	$(5\sqrt{10}, \frac{1}{2}\sqrt{10})$	5
325 [6]	$(\frac{1}{2} - \frac{1}{2}\sqrt{33}, -\sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{33}})$	$((\frac{1}{2} - \frac{1}{2}\sqrt{33})^2 - \sqrt{\frac{1}{2} + \frac{1}{2}\sqrt{33}})$
329 [6]	$(\frac{2819}{200}, 5 - \frac{1}{200}\sqrt{691239})$	$\frac{549353259}{8000000} + (-15 - \frac{1}{200}\sqrt{691239})^3$
335 [6]	$(\frac{1}{100\sqrt{24200011}}, \frac{22}{\sqrt{24200011}},$ $\frac{484001}{242000110})$	$-\frac{1}{1100000}\sqrt{24200011}$
336 [6]	$(\frac{20}{27} - \frac{407}{144369}\sqrt{5347},$ $\frac{10}{27} + \frac{323}{144369}\sqrt{5347},$ $-\frac{4}{27} - \frac{140}{144369}\sqrt{5347})$	$\frac{64}{27} - \frac{1}{27}\sqrt{5347}$
337 [6]	$(\frac{1}{\sqrt{3}}, \sqrt{3}, 0)$	6
338 [6]	$\approx (-0.3665301174,$ $-1.662075951,$ $2.845341009)$	≈ -10.99280625
340 [6]	$(\frac{3}{5}, \frac{3}{10}, \frac{3}{10})$	$-\frac{27}{500}$
341 [6]	$(4, 2\sqrt{2}, 2)$	$-16\sqrt{2}$
343 [6]	$\{(100\frac{\sqrt{419}}{t}, \frac{27}{167600}t^2, t) :$ $t \in [\frac{25}{9}\sqrt{419}, 125]\}$	$-\frac{2273913}{400000}$

Appendix: Test Examples

The method outlined in the paper has been coded in Maple. The code can be found by world wide web at *www.optsyst.math.kth.se*. It has been run on a set of test problems from [4] and [6]. These test problems are marked as 'practical problems', i.e. 'with exact solution unknown'. The listed global optima are computed using FJ-conditions. In the table below, RO denotes that the solution is exact, but is given in terms of roots of single variable polynomials. This implies that these solutions can be computed to arbitrary precision. In example 338, Maple was not able to correctly solve a third degree equation exactly. Approximate solution, however, gave the solution given. This is different from the one given in [6]. Also note example 343, where the global optimum is a curve, identified by our method. Except for examples 338 and 343 we have got the same solutions as [4] and [6] (up to numerical precision).

References

1. Bazaraa, M. S., Sherali, H. D. and Shetty, C. M., *Nonlinear Programming, Theory and Algorithms*, 2nd Edition, John Wiley and Sons, 1993.
2. Cox, D., Little, J. and O'Shea, D., *Ideals, Varieties and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra*, Springer-Verlag, 1992.
3. Czapor, S. R., *Soving Algebraic Equations: Combining Buchberger's Algorithm with Multivariate Factorization*, J. Symbolic Computation 7, pp. 49–53, Academic Press Limited, 1989.
4. Hock, W. and Schittkowski, K., *Test Examples for Nonlinear Programming Codes*, Springer-Verlag, 1981.
5. Hollman, J., *Theory and Applications of Gröbner Bases*, PhD thesis, Royal Institute of Technology, Stockholm, Sweden, 1992.
6. Schittkowski, K., *More Test Examples for NonLinear Programming Codes*, Springer-Verlag, 1987.