An. Şt. Univ. Ovidius Constanţa Vol. 14(1), 2006, 23–44

# Algorithm for the Gröbner region of a principal ideal

**Alexandru Bobe** *

**Abstract**

Gröbner basis theory is a fundamental tool of computational commutative algebra. The theory has been advanced by the introduction of techniques from combinatorics, polyhedral geometry and computational geometry. In particular, such techniques were used to create the concept of Gröbner region for an ideal of a polynomial ring. The purpose of this paper is to present algoritms for computing the Gröbner region of a principal ideal in two indeterminates, to implement in `Singular` [1] and also to visualize this object with `Mathematica` [2].

**Subject Classification: 90C57; 11P21; 62F25; 62G15.**

## 1 Introduction

Let $k$ be a field and let $k\,[X] = k\,[X_1, ..., X_n]$ be the polynomial ring in $n$ variables. Let $<$ be a term ordering and $I$ be an ideal in $k[X]$.

For $f \in k[X] - \{0\}$ we denote by:

$$in_<(f) = \textit{ the initial monomial } \text{of } f,$$

that is the leading monomial of $f$ with respect to $<$ and by

$$in_<(I) = (in_<(f) | f \in I, f \neq 0)$$

*the initial ideal* of $I$.

Key Words: Gröbner region; Ideal of a polynomial ring.
Received: December, 2005

*Work partially supported by the CEEX Programme of the Romanian Ministery of Education and Research, contract CEX 05-D11-11/2005

**Definition 1.1.** *A finite subset $G = \{g_1, ..., g_r\} \subset I$ is a **Gröbner basis** for $I$ with respect to $<$ if $in_<(I) = (in_<(g_1), ..., in_<(g_r))$.*

**Definition 1.2.** *Let $\omega \in \mathbb{R}^n$. For any polynomial $f = \sum\limits_{v \in \mathbb{N}^n} a_v X^v \in k[X]$ we define **the initial form** $in_\omega(f) = \sum\limits_{v' \in \mathbb{N}^n} a_{v'} X^{v'}$, where the vectors $v'$ maximize $\omega \cdot v'$ in $\{v \mid a_v \neq 0\}$ that is $\omega \cdot v' \geq \omega \cdot v$ for any $v$ with $a_v \neq 0$.*

**Definition 1.3.** *For an ideal $I \subset k[X]$ we define **the initial form of the ideal** $I$: $in_\omega(I) = (in_\omega(f) \mid f \in I)$.*

**Example 1.4.** *Let $I$ be the ideal generated by*

$$f(X_1, X_2) = 4X_1^6X_2^2 + 5X_1^5X_2^3 - X_1^4 + 3X_1^2X_2^4 + X_1^2 + X_1X_2 + X_2^3 + 7.$$

*We compute the initial form for $\omega = (2, 1)$ and $\omega = (1, 1)$. The vectors $v$ with $a_v \neq 0$ are: $v_1 = (6, 2)$, $v_2 = (5, 3)$, $v_3 = (4, 0)$, $v_4 = (2, 4)$, $v_5 = (2, 0)$, $v_6 = (1, 1)$, $v_7 = (0, 3)$, $v_8 = (0, 0)$.*

*If $\omega = (2, 1)$ then $\omega \cdot v_i = \{14, 13, 8, 8, 4, 3, 3, 0\}$, $i = \overline{1, 8}$ and the maximum of this list is 14 given by $\omega \cdot v_1$. So $in_\omega(f) = X_1^6X_2^2$, thus $in_\omega(I) = (X_1^6X_2^2)$. This is a monomial ideal.*

*For $\omega = (1, 1)$ then $\omega \cdot v_i = \{8, 8, 4, 6, 2, 2, 3, 0\}$, $i = \overline{1, 8}$ and the maximum of this list is 8 given by $\omega \cdot v_1$ and $\omega \cdot v_2$. So $in_\omega(f) = 4X_1^6X_2^2 + 5X_1^5X_2^3$, and $in_\omega(I) = (4X_1^6X_2^2 + 5X_1^5X_2^3)$. This is not a monomial ideal.*$\square$

**Definition 1.5.** *Let $\omega \in \mathbb{R}_+^n$ and $<$ be a term order. We define $<_\omega$ **the term ordering with respect to the weight vector** $\omega$, as follows: for $a, b \in \mathbb{N}^n$ we set $a <_\omega b \Longleftrightarrow \omega \cdot a < \omega \cdot b$ or ($\omega \cdot a = \omega \cdot b$ and $a < b$).*

**Lemma 1.6.** *(Proposition 1.8, [3]). For every ideal $I \subset k[X]$ we have $in_<(in_\omega(I)) = in_{<_\omega}(I)$.*

*Proof*: See [3].$\square$

**Example 1.7.** *Let $I = (f)$, $f$ from Example 1.4, $<$ be the lexicographical ordering with $X_1 > X_2$ and $\omega = (1, 1)$. Then*

$$in_<(in_\omega(I)) = in_<(4X_1^6X_2^2 + 5X_1^5X_2^3) = (X_1^6X_2^2)$$

*because $X_1^6X_2^2 > X_1^5X_2^3$. When we compute $in_{<_\omega}(I)$ like in Definition 1.5 we obtain $in_{<_\omega}(I) = (X_1^6X_2^2)$.*$\square$

**Corollary 1.8.** *(Corollary 1.9, [3]). If $\omega \geq 0$ and $G$ is a Gröbner basis of $I$ with respect to $<_\omega$, then $\{in_\omega(g) \mid g \in G\}$ is a Gröbner basis for $in_\omega(I)$ with respect to $<$.*

*Proof:* Let $G = \{g_1, g_2, ..., g_r\}$ be a Gröbner basis for $I$ with respect to $<_\omega$. From the definition of Gröbner basis and Lemma 1.6 it results

$$in_{<_\omega}(I) = in_<(in_\omega(I)) = (in_{<_\omega}(g_1), in_{<_\omega}(g_2), ..., in_{<_\omega}(g_r)) =$$

$$= (in_<(in_\omega(g_1)), in_<(in_\omega(g_2)), ..., in_<(in_\omega(g_r)))\,.$$

It follows

$$in_<(in_\omega(I)) = (in_<(in_\omega(g_1)), in_<(in_\omega(g_2)), ..., in_<(in_\omega(g_r)))\,,$$

i.e. the initial ideal of $in_\omega(I)$ is generated by the initial monomials of $in_\omega(g_1), in_\omega(g_2), ..., in_\omega(g_r)$ and this is exactly the definition of a Gröbner basis for $in_\omega(I)$. We can conclude that $\{in_\omega(g) \mid g \in G\}$ is a Groebner basis for $in_\omega(I)$ with respect to $<$ .□

**Corollary 1.9.** *(Corollary 1.10, [3]). If $\omega \geq 0$ and $in_\omega(I)$ is a monomial ideal, then $in_\omega(I) = in_{<_\omega}(I)$.*

We give an example of computing the notions in the above corollaries:

**Example 1.10.** *Let $I = (f)$, $f$ from Example 1.4, and $<$ be the lexicographical order ($X_1 > X_2$). Let $\omega = (2, 1)$. Then $in_\omega(I) = in_{<_\omega}(I)$.*

*The initial form $in_\omega(I) = (X_1^6 X_2^2)$ is a monomial ideal and is the same with $in_{<_\omega}(I)$. For $\omega = (1, 1)$ the initial form $in_\omega(I) = (4X_1^6X_2^2 + 5X_1^5X_2^3)$ is not a monomial ideal and is different from $in_{<_\omega}(I) = (X_1^6X_2^2)$.*□

The following proposition finds, for every term ordering $<$, a vector $\omega$ which represents the term ordering and it is easier in computations to use this vector instead of $<$:

**Proposition 1.11.** *(Proposition 1.11, [3]). For any term ordering $<$ and any ideal $I \subset k[X]$, there exists a non-negative integer vector $\omega \in \mathbb{N}^n$ such that $in_\omega(I) = in_<(I)$.*

*Proof*: See [3].□

**Definition 1.12.** *Let $\omega \in \mathbb{R}^n$ and $<$ be a term ordering such that $in_\omega(I) = in_<(I)$. $\omega$ is called **a term ordering for $I$ which represents the term order** $<$.*

**Definition 1.13.** *Let $I \subset k[X]$ be an ideal. We define the* ***Gröbner region*** *$GR(I)$ to be the set of all $\omega \in \mathbb{R}^n$ such that $in_\omega(I) = in_{\omega'}(I)$ for some $\omega' \geq 0$.*

*So we can write*

$$GR(I) = \{\omega \in \mathbb{R}^n \mid in_\omega(I) = in_{\omega'}(I) \ \textit{ for some } \ \omega' \geq 0\}\,. \qquad (1.1)$$

**Remark 1.14.** *$GR(I)$ contains the non-negative orthant $\mathbb{R}^n_+$.*

**Example 1.15.** *For $I = (f)$, $f$ from example 1.4 we compute $GR(I)$.*

*Each exponent vector $v_i$, with $a_{v_i} \neq 0$, determines the point $A_i$ in the plane like in the Figure 1. Their convex hull is a hexagon.*

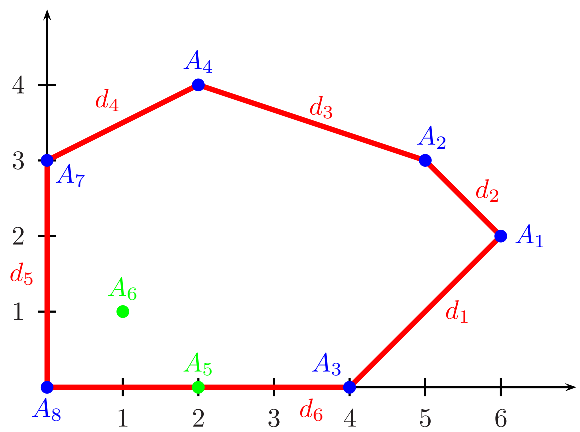



*Figure 1*

*We are interested in finding those vectors $\omega \in \mathbb{R}^2$ such that $in_\omega(I) = in_{\omega'}(I)$ for some $\omega' \in \mathbb{R}^2_+$, that is we want to find all the directions $\omega = (\omega_1, \omega_2) \in \mathbb{R}^2$ such that they maximize $\omega \cdot v_i$ and have non-empty intersection with the first quadrant of $\mathbb{R}^2$.*

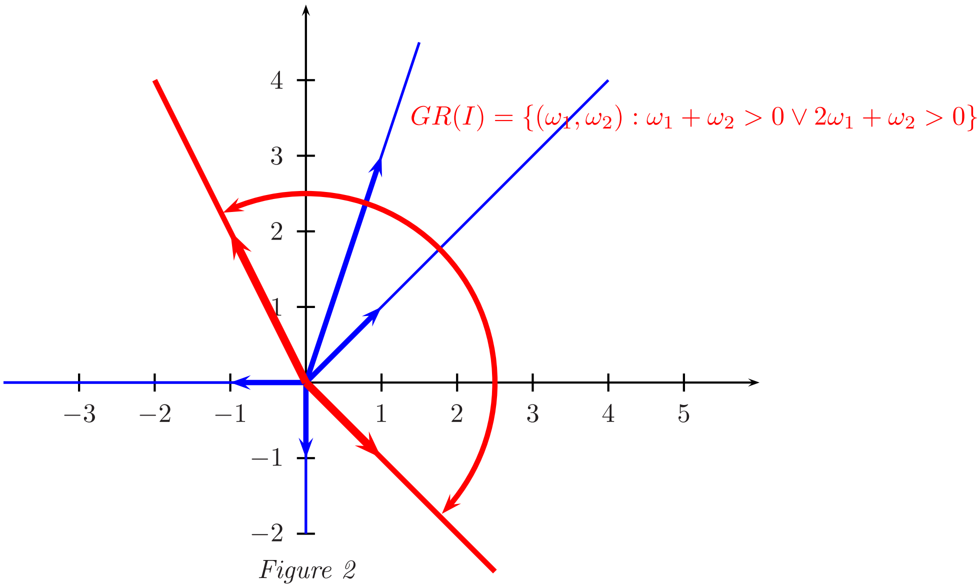



*Figure 2*

*So, we have to find the directions which are between the normal vectors of* $d_1$ *and* $d_4$. *In Figure 2 we have all the possible directions divided in classes. The Gröbner region is given by the region marked with circular arrow i.e.:*

$$GR(I) = \left\{(\omega_1, \omega_2) \in \mathbb{R}^2 \mid \omega_1 + \omega_2 > 0 \text{ or } 2\omega_1 + \omega_2 > 0\right\}.$$

*The other regions (unmarked) maximize the product* $\omega \cdot v_i$, *but the intersection with the first quadrant is empty.*□

## 2 Geometric interpretations and algorithms

Now we give an algorithm, along with its implementation in `Singular` and visualization in `Mathematica`, for computing the Gröbner region $GR(I)$ for a principal ideal $I = (f)$, $f \in k[X, Y]$.

Let $f = c_1 X^{a_1} Y^{b_1} + c_2 X^{a_2} Y^{b_2} + ... + c_n X^{a_n} Y^{b_n}$, $c_1, c_2, ..., c_n \neq 0$. Each vector $v_i = (a_i, b_i)$ determines the point $A_i(a_i, b_i)$ in the plane like in the Figure 1. For computing $GR(I)$ we are interested in finding those vectors $\omega \in \mathbb{R}^2$ such that $in_\omega(I) = in_{\omega'}(I)$ for some $\omega' \in \mathbb{R}^2_+$(first qudrant) i.e. we want to find the union of all sets with $\omega \in \mathbb{R}^2$ as elements such that fulfill the condition $in_\omega(I) = in_{\omega'}(I)$, for some $\omega' \in \mathbb{R}^2_+$. From definition of $in_\omega(I)$, the condition $in_\omega(I) = in_{\omega'}(I)$ means in fact to find those $\omega \in \mathbb{R}^2$ such that $\omega v_i$ is maximum, $i = \overline{1, n}$.

Let us consider

$$New(f) = conv(\{v_1, ..., v_n\}) = \left\{ \sum_{j=1}^{n} \lambda_j v_j \mid \lambda_j \in \mathbb{R}_+ \ , \ \sum_{j=1}^{n} \lambda_j = 1 \right\}$$

the *Newton polytope* for $f$.

From a theorem of linear programming ([4]) we know that an optimum value for $\omega v$ (a linear function) is obtained on the frontier of $New(f)$ and, more exactly, in a vertex of the frontier $H$ of $New(f)$. Using this result, our problem is to find for each vertex of $H$ the directions $\omega = (\omega_1, \omega_2) \in \mathbb{R}^2$ such that the problem

$$\begin{cases} \max(\omega_1 x + \omega_2 y) \\ v = (x, y) \in New(f) \end{cases}$$

riches its solution in the considered vertex. In this way we determine in fact the sets

$$S_j^\omega = \left\{ \omega \in \mathbb{R}^2 \mid \omega v \text{ rich its maximum in the point } A_j, \ v \in New(f) \right\},$$

$j = \overline{1, m}$, where $m =$ the number of vertices of the frontier $H$.

Then we set

$$GR(I) = \bigcup_{j=1}^{m} (S_j^\omega \cap \mathbb{R}_+^2).$$

Let us compute for example $S_2^\omega$ for $I = (f)$, $f$ from example 1.4. The Newton polytope of $f$ is given by the convex hull of the points $A_i$, $i = \overline{1, 8}$, which is the union of the segment lines $d_j$, $j = \overline{1, 6}$ (see Figure 1). As we saw, $S_2^\omega$ means the set of all $\omega = (\omega_1, \omega_2) \in \mathbb{R}^2$ such that $\omega v$ rich its maximum in the point $A_2$. So we have to solve the following problem:

$$\begin{cases} \max(\omega_1 x + \omega_2 y) \\ y \geq x - 4 \\ y \leq -x + 8 \\ y \leq -\frac{1}{3}x + \frac{14}{3} \\ y \leq \frac{1}{2}x + 3 \\ x \geq 0 \\ y \geq 0 \end{cases}.$$

If we consider the straight line $d : \omega_1 x + \omega_2 y = t$ then $n_d = (\omega_1, \omega_2)$ is the normal vector of this straight line. So, the normal vector of the sweep line $\omega_1 x + \omega_2 y = t$ which satisfies the problem is between $n_{d_2} = (1, 1)$ and $n_{d_3} = (1, 3)$ as we can see in Figure 2.

As we saw before, the first important object for computing the Gröbner region is the convex hull of a set of $n > 2$ points, which is the frontier of $New(f)$.

The basic idea (Graham, 1972) for finding the convex hull of $n$ points in two dimensions in $O(n \log n)$ time is simple: assume we are given a point $P$ on the hull. We use as $P$ the lowest point, which is clearly on the hull. If there are several points with the same minimum $y$-coordinate, we use the right most of the lowest as starting point of the sorting. In our example this point is $A_3$. Now we sort the points by angle, counter-clockwise about the point $P$. Collinearities raise an issue, but we suppose we use this rule: if $angle(A) = angle(B)$, then define $A < B$ if $\|A - P\| < \|B - P\|$ i.e. closer points are treated as earlier in the sorting sequence. For our example the sorted points are labeled: $A_1, A_2, A_4, A_7, A_6, A_5, A_8$. The points are now processed in their sorted order, and the hull grown incrementally around the set. At any step, the hull will be correct for the points examined so far, but of course points encountered later will cause earlier decisions to be reevaluated.

The hull is maintained in a circular list $S$ of points. Initially, the list containes all the points: $S = \{A_3, A_1, ..., A_3\}$. A point $A_i$ is deleted from $S$ if this point forms a right turn i.e. the determinant formed with $A_{i-1}, A_i, A_{i+1}$ is negative (ex: $A_6$) or zero (ex: $A_5$).

After we delete all the points with this property we obtain the convex hull $S = \{A_3, A_1, A_2, A_4, A_7, A_8, A_3\}$.

The algorithm in pseudocod is:

**Algorithm** *Convex hull* (see [5])
*Input*: $n > 2$ points $A_1, ..., A_n$.
*Output*: The convex hull of the points.
1. Find rightmost lowest point: label it $P$.
2. Sort all other points angulary about $P$:
    2.1. Break ties in favor of closeness to $P$.
    2.2. Label $A_1, A_2, ..., A_{n-1}$.
3. List $S = (P, A_1, A_2, ..., A_{n-1}, P)$.
4. $i = 1$.
While $i < n$
    if $A_i$ forms a right turn with $(A_{i-1}, A_{i+1})$
        then eliminate $A_i$ from $S$
5. Print $S$.

Now we want to find the straight lines which give us $GR(I)$. Those straight lines are determined by the exterior normal vectors of the sides of the convex hull. For each side $A_iA_{i+1}$ (where $A_i = (a_i, b_i)$) of the convex hull, the normal vector is $n_i = (b_{i+1} - b_i, a_i - a_{i+1})$. If we want to have always the exterior normal vector we have to verify if $n_i$ is oriented to the exterior of the convex hull i.e. the points $A_i$, $A_{i+1}$ and $N_i$ have negative orientation, where $N_i = (\alpha_i, \beta_i)$ is the translation of $n_i$ in $A_i$:

$$\begin{vmatrix} a_i & b_i & 1 \\ a_{i+1} & b_{i+1} & 1 \\ \alpha_i & \beta_i & 1 \end{vmatrix} < 0. \tag{2.1}$$

So, the normal exterior vector $n_i^e = n_i$ if the condition (2.1) is fulfilled, and $n_i^e = -n_i$, otherwise.

Having the normal exterior vectors $\{n_1^e, ..., n_m^e\}$, where $m =$ number of points in the convex hull, we choose those vectors with positive components: $\left\{n_i^e, n_{i+1}^e ..., n_j^e\right\}$ (the normal exterior vectors are consecutive because the sides of the convex hull are sorted). Then $GR(I)$ is the region between the straight lines passing through $(0, 0)$ and having as directing vectors $n_{i-1}^e$ and $n_{j+1}^e$.

In the case when we have $n \leq 3$, the Newton polytope and the Gröbner region are computed directly, as follows: for $n = 1$ the Newton polytope is the point itself and the Gröbner region is the entire $\mathbb{R}^2$; for $n = 2$ the Newton polytope is the segment line formed with this two points, and the Gröbner region can be $\mathbb{R}^2$ or a half plane depending on the position of the exterior normal vector of the straight line formed with the two points.

Now we can write the algorithm:

**Algorithm**: *Gröbner region*
*Input*: $I = (f)$, $f = c_1 X^{a_1} Y^{b_1} + ... + c_n X^{a_n} Y^{b_n}$, $c_1, c_2, ..., c_n \neq 0$.
*Output*: $GR(I)$.
1. For each monomial of $f$ do: $v_i := (a_i, b_i)$, $i = \overline{1, n}$.
2. If $n < 2$ then: $GR(I) = \mathbb{R}^2$; goto step 8.
3. If $n = 2$ then:
Compute the normal exterior vector $n^e = (a, b)$, $a > 0$. If $b > 0$ then:
   3.1. $GR(I) = \mathbb{R}^2$; goto step 8.
   3.2. Else: $GR(I) = \left\{(\omega_1, \omega_2) \in \mathbb{R}^2 \mid -b\omega_1 + a\omega_2 > 0\right\}$; goto step 8.
4. Compute $H := Convexhull(v_1, ..., v_n) = \{A_1, ..., A_m\}$, $m =$ the number of vertices in the convex hull.
5. Compute the normal exterior vectors of $H$: $V := \{n_1^e, ..., n_m^e\}$.
6. Choose from $V$ the vectors with positive components: $\left\{n_i^e, n_{i+1}^e ..., n_j^e\right\}$.
7. Let $n_{i-1}^e = (a_{i-1}, b_{i-1})$ and $n_{j+1}^e = (a_{j+1}, b_{j+1})$ be the directing vectors of the straight lines which give us $GR(I)$. Then

$$GR(I) = \left\{(\omega_1, \omega_2) \in \mathbb{R}^2 \mid -b_{i-1}\omega_1 + a_{i-1}\omega_2 > 0 \text{ or } b_{j+1}\omega_1 - a_{j+1}\omega_2 > 0\right\}$$

8. Print $GR(I)$.

## 3 The implementation

We'll implement in `Singular` the algorithms we saw before, and we'll also construct a string which, pasted in `Mathematica`, will give us the drawings of the desired objects.

First of all we discourse the cases when the number of monomials is less than 3. In this cases we will construct directly the Newton polytope and the Gröbner region. The procedure `showmat` prints a given matrix and the procedure `multind` computes the exponent vectors and deals with the particular cases. For this cases the program gives now all the strings that we'll paste in `Mathematica`. We'll discuss later the meanings of those strings.

```
LIB "matrix.lib";

proc showmat(intmat a,int n)
{
 int i,j;
 intmat b[n][2];
 for (i=1;i<=n;i++)
 {
  for (j=1;j<=2;j++)
  {
   b[i,j]=a[i,j];
  }
 }
 print(b);
}

proc multind(poly f)
{
 int m,j;
 while (f<>0)
 {
  m++;
  for (j=1;j<=2;j++)
  {
   a[m,j]=leadexp(f)[j];
  }
  f=f-lead(f);
 }
 if (m<2)
 {
```

```
 print("THE POLYNOMIAL HAS LESS THAN 2 MONOMIALS.");
 print("THE NEWTON POLYTOPE is:");
 showmat(a,m);
 print("THE GROEBNER REGION is: R^2 !");
 exit;
}
if (m==2)
{
 print("THE NEWTON POLYTOPE is:");
 showmat(a,m);
 print("");
 print("THE GROEBNER REGION is:");
 int vnx,vny;
 vnx=a[2,2]-a[1,2];
 vny=-(a[2,1]-a[1,1]);
 if (vnx*vny>0)
 {
  string s4;
  s4="GR(I)={(w1,w2) in R^2 | "+
  string(-vny)+"*w1+"+string(vnx)+"*w2>0"+
  " or "+
  string(vny)+"*w1+"+string(-vnx)+"*w2>0"+
  "}";
  s4;
  string s,s1,sg,s3;
  s="{"+string(a[1,1])+","+string(a[1,2])+"}"+
  ","+"{"+string(a[2,1])+","+string(a[2,2])+"}";
  s="{"+s+"}";
  s1="Show[Graphics[{
  RGBColor[0,0,1],PointSize[.03],Map[Point,"+s+"],
  RGBColor[1,0,0],Thickness[.01],Line["+s+"]},
  AspectRatio->Automatic,Axes->True]];";
  s1;
  sg="{"+string(vnx)+","+string(vny)+"}";
  sg="{"+sg+"}";
  s3="Show[Graphics[{
  RGBColor[1,0,0],Thickness[.01],
  Map[Line[{{0, 0}, #}] &,"+ sg+"]},
  AspectRatio->Automatic,Axes -> True]];";
  s3;
 }
```

```
  else
  {
   if (vnx<=0)
   {
    vnx=-vnx;
    vny=-vny;
   }
   string s4;
   s4="GR(I)={(w1,w2) in R^2 | "+
   string(-vny)+"*w1+"+string(vnx)+"*w2>0"+"}";
   s4;
   string s,s1,sg,s3;
   s="{"+string(a[1,1])+","+string(a[1,2])+"}"+
    ","+"{"+string(a[2,1])+","+string(a[2,2])+"}";
   s="{"+s+"}";
   s1="Show[Graphics[{
   RGBColor[0,0,1],PointSize[.03],Map[Point,"+s+"],
   RGBColor[1,0,0],Thickness[.01],Line["+s+"]},
   AspectRatio->Automatic,Axes->True]];";
   s1;
   sg="{"+string(vnx)+","+string(vny)+"}"+
   ","+"{"+string(-vnx)+","+string(-vny)+"}";
   sg="{"+sg+"}";
   s3="Show[Graphics[{
   RGBColor[1,0,0],Thickness[.01],
   Map[Line[{{0, 0}, #}] &,"+ sg+"]},
   AspectRatio->Automatic,Axes -> True]];";
   s3;
  }
  print("");
  print("In order to see the Newton polytope and the
  Groebner region paste the last 7 output lines into a
  Mathematica notebook and then evaluate the cell with
  SHIFT RETURN");
  exit;
 }
 print("");
 print("The EXPONENT VECTORS are:");
 showmat(a,m);
 n=m;
}
```

Following the algorithm for the convex hull we have to determine the rightmost lowest point and to bring this point in the first position with the procedure `interschimb`.

```
proc interschimb(int i,int j)
{
 int tempx=a[i,1];
 int tempy=a[i,2];
 a[i,1]=a[j,1];
 a[i,2]=a[j,2];
 a[j,1]=tempx;
 a[j,2]=tempy;
}

proc rmlowest(int n)
{
 print("");
 print("The NUMBER of the points is:");
 n;
 int i,poz;//position of the rightmost lowest point
 int ymin=a[1,2];
 poz=1;
 for (i=1;i<=n;i++)
 {
  if (a[i,2]<ymin)
  {
   ymin=a[i,2];
   poz=i;
  }
  else
  {
   if (a[i,2]==ymin)
   {
    if (a[i,1]>a[poz,1])
    {
     poz=i;
    }
   }
  }
 }
 print("");
 print("The RIGHTMOST LOWEST POINT is:");
```

```
 printf("(%s,%s)",a[poz,1],a[poz,2]);
 interschimb(1,poz);
 print("");
 print("The MATRIX before the sorting procedure is:");
 showmat(a,n);
}
```

Now we have to sort the points angulary with respect to the rightmost lowest point determined before. The procedure `detpuncte` forms a $3 \times 3$ submatix of the a given one and it computes the determinant.

```
proc detpuncte(intmat a,int i,int j,int k)
{
 intmat b[3][3]=a[i,1],a[i,2],1,
               a[j,1],a[j,2],1,
               a[k,1],a[k,2],1;
 return(det(b));
}

proc sortpoints(int n)
{
 int i;
 int temp;
 int t=0;
 while (t==0)
 {
  t=1;
  for (i=2;i<n;i++)
  {
   if (detpuncte(a,1,i,i+1)<0)
   {
    interschimb(i,i+1);
    t=0;
   }
   else
   {
    if ((detpuncte(a,1,i,i+1)==0) and (a[i,2]>a[i+1,2]))
    {
     interschimb(i,i+1);
    }
   }
  }
```

```
 }
 print("");
 print("The MATRIX with the POINTS SORTED angulary is:");
 showmat(a,n);
}
```

Having the matrix of the sorted point we can construct the circular list:

```
proc circlist(intmat a,int n)
{
 int i,j;
 for (i=1;i<=n;i++)
 {
  for (j=1;j<=2;j++)
  {
   c[i,j]=a[i,j];
  }
 }
 c[n+1,1]=a[1,1];
 c[n+1,2]=a[1,2];
 print("");
 print("The CIRCULAR LIST of the points is:");
 showmat(c,n+1);
}
```

The procedure `elimin` has the role to eliminate a given point from the circular list before. Using this procedure we can compute the Newton polytope for $f$ like in the algorithm given in section 2:

```
proc elimin(int i)
{
 int k;
 for (k=i;i<m;i++)
 {
  c[i,1]=c[i+1,1];
  c[i,2]=c[i+1,2];
 }
 m=m-1;
}

proc newtonpoly()
{
```

```
 print("");
 int i=1;
 while (i<m-1)
 {
  if (detpuncte(c,i,i+1,i+2)>0)
  {
   i=i+1;
  }
  else
  {
   print("ELIMINATE the POINT");
   printf("(%s,%s)",a[i+1,1],a[i+1,2]);
   elimin(i+1);
   if (i<>1)
   {
    i=i-1;
   }
  }
 }
 print("");
 print("THE NEWTON POLYTOPE is:");
 showmat(c,m);
}
```

Following the algorithm from section 2 we have to compute the exterior normal vectors. The procedure `deterv` computes the determinant of a choosen $3 \times 3$ matrix.

```
proc deterv(int j,int vx,int vy)
{
 intmat b[3][3]=c[j,1],c[j,2],1,c[j+1,1],c[j+1,2],1,vx,vy,1;
 return(det(b));
}

proc envectors(int m)
{
 int i;
 print("");
 print("THE EXTERIOR NORMAL VECTORS are:");
 int vx,vy;
 for (i=1;i<m;i++)
 {
```

```
  v[i,1]=c[i+1,2]-c[i,2];
  v[i,2]=-(c[i+1,1]-c[i,1]);
  vx=c[i+1,2]-c[i,2]+c[i,1];
  vy=c[i,1]-c[i+1,1]+c[i,2];
  if (deterv(i,vx,vy)>0)
  {
   v[i,1]=-v[i,1];
   v[i,2]=-v[i,2];
  }
 }
 showmat(v,m-1);
}
```

Now we have all the elements to implement the algorithm for the Gröbner region for any polynomial in two indeterminates. The variables `right` and `left` indicates the positions of the frontier half lines of the Gröbner region.

```
proc groebnerregion(intmat v,int m)
{
 int i=1;
 while (i<m)
 {
  if ((v[i,1]>0) and (v[i,2]>0))
  {
   right=i;
   i++;
   while ((v[i,1]>0) and (v[i,2]>0))
   {
    i++;
   }
   left=i;
   i++;
  }
  else
  {
   i++;
  }
 }
 right--;
 print("");
 print("The DIRECTING VECTORS of the FRONTIER HALF LINES
  of the Groebner region are:");
```

```
 if (right==0)
 {
  right=m-1;
 }
 printf("(%s,%s)",v[right,1],v[right,2]);
 printf("(%s,%s)",v[left,1],v[left,2]);
}
```

We can construct strings in `Singular` that will permit us to visualize the Newton polytope and the Gröbner region. This strings also containes some instructions for handling the input in `Mathematica`.

```
proc mathstrings(intmat v,int m)
{
 int i;
 print("");
 print("THE GROEBNER REGION is:");
 string s4;
 s4="GR(I)={(w1,w2) in R^2 | "+
  string(-v[right,2])+"*w1+"+string(v[right,1])+"*w2>0"+
  " or "+
  string(v[left,2])+"*w1+"+string(-v[left,1])+"*w2>0"+
  "}";
 s4;
 print("");
 string s,s1,s2,sg,s3;
 s="{"+string(c[1,1])+","+string(c[1,2])+"}";
 for (i=2;i<=m;i++)
 {
  s=s+","+"{"+string(c[i,1])+","+string(c[i,2])+"}";
 }
 s="{"+s+"}";
 s2="{"+string(a[1,1])+","+string(a[1,2])+"}";
 for (i=2;i<=n;i++)
 {
  s2=s2+","+"{"+string(a[i,1])+","+string(a[i,2])+"}";
 }
 s2="{"+s2+"}";
 s1="Show[Graphics[{
  RGBColor[0,1,0],PointSize[.03],Map[Point,"+s2+"],
  RGBColor[0,0,1],Map[Point,"+s+"],
  RGBColor[1,0,0],Thickness[.01],Line["+s+"]},
```

```
  AspectRatio->Automatic,Axes->True]];";
 s1;
 sg="{"+string(v[1,1])+","+string(v[1,2])+"}";
 for (i=2;i<m;i++)
 {
  sg=sg+","+"{"+string(v[i,1])+","+string(v[i,2])+"}";
 }
 sg="{"+sg+"}";
 s3="Show[Graphics[{
  RGBColor[0,0,1],Thickness[.01],Map[Line[{{0, 0}, #}] &,
  "+ sg+"],
  RGBColor[1,0,0],Thickness[.01],Line[{{0,0},
  {"+string(v[right,1])+","+string(v[right,2])+"}}],
  Line[{{0,0},{"+string(v[left,1])+","+string(v[left,2])+"}}]},
  AspectRatio->Automatic,Axes -> True]];";
 s3;
 print("");
 print("In order to see the Newton polytope and the
 GROEBNER REGION
 paste the last 10 output lines into a Mathematica
 notebook and then evaluate the cell with SHIFT RETURN
 The NEWTON POLYTOPE is the polygon in red.

 The GROEBNER REGION is the sector which contains the first
 quadrant between the two red half lines.");
}
```

The main program begins with the initialisation of the polynomial for which we want to compute the Newton polytope and the Gröbner region, and then we call the procedures discussed before, like in the algorithm from section 2.

```
int n;
ring r=0,(x,y),Dp;
intmat a[100][2];
poly f=4x6y2+5x5y3-x4+3x2y4+x2+xy+y3+7;
multind(f);
rmlowest(n);
sortpoints(n);
intmat c[n+1][2];
circlist(a,n);
int m=n+1;
```

```
newtonpoly();
intmat v[m][2];
envectors(m);
int right,left;
groebnerregion(v,m);
mathstrings(v,m);
```

Let us see now how this program works for the polynomial given at the beginning of the main program. Following the algorithm, the output gives us the results of each procedure and illustrates the important steps:

```
The EXPONENT VECTORS are:
     6     2
     5     3
     2     4
     4     0
     0     3
     2     0
     1     1
     0     0

The NUMBER of the points is:
8

The RIGHTMOST LOWEST POINT is:
(4,0)

The MATRIX before the sorting procedure is:
     4     0
     5     3
     2     4
     6     2
     0     3
     2     0
     1     1
     0     0

The MATRIX with the POINTS SORTED angulary is:
     4     0
     6     2
     5     3
     2     4
```

```
     0     3
     1     1
     2     0
     0     0

The CIRCULAR LIST of the points is:
     4     0
     6     2
     5     3
     2     4
     0     3
     1     1
     2     0
     0     0
     4     0

ELIMINATE the POINT
(2,0)
ELIMINATE the POINT
(1,1)

THE NEWTON POLYTOPE is:
     4     0
     6     2
     5     3
     2     4
     0     3
     0     0
     4     0

THE EXTERIOR NORMAL VECTORS are:
     2    -2
     1     1
     1     3
    -1     2
    -3     0
     0    -4

The DIRECTING VECTORS of the FRONTIER HALF LINES
  of the Groebner region are:
(2,-2)
```

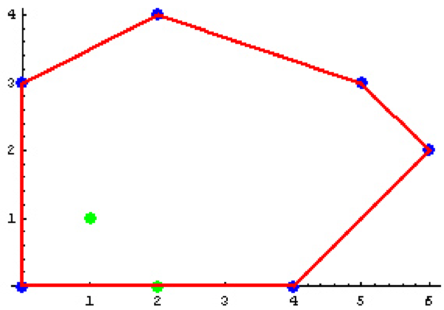
```
(-1,2)

THE GROEBNER REGION is:
GR(I)={(w1,w2) in R^2 | 2*w1+2*w2>0 or 2*w1+1*w2>0}
```

We can observe that the Gröbner region computed with this program is exactly the same with the one deduced theoreticaly in section 1.

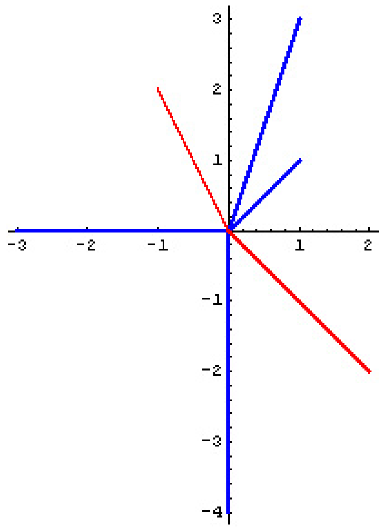
The output gives us also the strings in `Mathematica` language. If we follow the instructions given in the last rows of the output, we copy the last rows written in `Mathematica`:

```
Show[Graphics[{
  RGBColor[0,1,0],PointSize[.03],Map[Point,
  {{4,0},{6,2},{5,3},{2,4},{0,3},{1,1},{2,0},{0,0}}],
  RGBColor[0,0,1],Map[Point,
  {{4,0},{6,2},{5,3},{2,4},{0,3},{0,0},{4,0}}],
  RGBColor[1,0,0],Thickness[.01],
  Line[{{4,0},{6,2},{5,3},{2,4},{0,3},{0,0},{4,0}}]},
  AspectRatio->Automatic,Axes->True]];
Show[Graphics[{
  RGBColor[0,0,1],Thickness[.01],Map[Line[{{0, 0}, #}]
  &,{{2,-2},{1,1},{1,3},{-1,2},{-3,0},{0,-4}}],
  RGBColor[1,0,0],Thickness[.01],Line[{{0,0},{2,-2}}],
  Line[{{0,0},{-1,2}}]},
  AspectRatio->Automatic,Axes -> True]];
```

in a `Mathematica` notebook, and we'll have the drawings of the desired objects, like in the *Figure 1* and *Figure 2* from the first section. The Newton polytope and all the initial points computed with `Mathematica` are given in the figure below:




We can also see the Gröbner region computed with Mathematica, which is exactly the same as we deduced in section 1. As we said before, the Gröbner region is the sector which contains the first quadrant, between the half line from the fourth quadrant and the half line from the second quadrant:




## References


[1] G.M. Greuel, G. Pfister and H. Schönemann, *Singular 3.0. A Computer Algebra System for Polynomial Computations.* Centre for Computer Algebra, University of Kaiserslautern, 2001, http://www.singular.uni-kl.de.

[2] Wolfram Reasearch, *Mathematica 4.0*, http://www.wolfram.com.

[3] B. Sturmfels, *Gröbner Basis and Convex Polytopes*, AMS, 1996.

[4] G. Zoutendijk, *Mathematical Programming Methods*, North-Holland Publishing Company, 1976.

[5] F.P. Preparata, M.I. Shamos, *Computational Geometry. An Introduction*, Springer, 1985.


"Ovidius" University of Constanta
Department of Mathematics and Informatics,
900527 Constanta, Bd. Mamaia 124
Romania
e-mail: alex@univ-ovidius.ro