Generalized subresultants and generalized subresultant algorithm

Petr Glotov
pglotov@yahoo.com

Abstract
In this paper we present the notions of trail (pseudo-)division, generalized subresultants and generalized subresultant algorithm.

1 Trail pseudo-division

We will work in some polynomial ring $K[x]$. First, we define full polynomial as one with non-zero trail coefficient. All through the paper we will deal with full polynomials. Let we have two full polynomials $f$ and $g$. Now we describe the process of trail (pseudo-)division. The usual (pseudo-)division can be illustrated in the following scheme, where the coefficients of both polynomials are written from the left to the right by decreasing powers of $x$ and the second, third, etc. lines subtracts one after one from the first after some multiplications ("*" mean some coefficient, no other comments are needed there):

\[
\begin{array}{cccccc}
* & * & * & * & * & * & f \\
* & * & * & & & x^2 g \\
* & * & * & & xg \\
* & * & * & g \\
\end{array}
\]

We introduce the trail (pseudo-)division, which can be analogously illustrated:

\[
\begin{array}{cccccc}
* & * & * & * & * & * & f \\
* & * & * & & & g \\
* & * & * & & xg \\
* & * & * & & x^2 g \\
\end{array}
\]

Here the eliminations perform in the trail part of "bigger" polynomial $f$ by
the trail part of the "smaller" one \( g \). In each step one get polynomial from the ideal \((f, g)\) (it has zero in some lower terms). After removing of some maximal possible power of \( x \) \( (x \notin (f, g), \text{ if } \deg_x(\gcd(f, g)) > 0 \) as \( f \) and \( g \) are full and if \( \gcd(f, g) = 1 \) then resulting polynomial is obviously belongs to \((f, g)\)) one can get the polynomial of degree less than \( g \) has.

This way of division is usefull in the case of pseudo-division. In the usual pseudo-division the first polynomial \( f \) is multiplied by some degree of the leading coefficient of the second one \( g \). In the trail division it is multiplied by some degree of the trail coefficient of \( g \). If it is "less" in some sense than the leading coefficient then the resulting trail pseudo-remainder will have "smaller" coefficients than the usual one. Analogously in the case of division (not pseudo) the dividing of \( g \) can be performed by "smaller" term. The \( \text{tprem}(f, g) \) will denote the trail pseudo remainder of \( f \) and \( g \). The remark: in general we can change the "place" of elimination (e.g first vanish the leading coefficient, then the trail, again the trail, etc.).

If \( h \) is full polynomial than \( h^* \) will denote the reverted polynomial (e.g. \( (5x^4 + 4x^3 + 3x^2 + 2x + 1)^* = x^4 + 2x^3 + 3x^2 + 4x + 5 \)). The following formula is valid up to multiplying by some power of \( x \): \( \text{tprem}(f, g) = (\text{prem}(f^*, g^*))^* \).

To prove this formula one need to place the mirror near the scheme for trail pseudo-remainder. Then in the mirror one will see the process of finding usual pseudo-remainder of reverted polynomials.

Our next goal is to develop the algorithm analogous to the subresultant algorithm for gcd computation[[1]] with using trail pseudo-remainders. For this purpose we fix here the generalized algorithm for pseudo-remainders \( \text{genPRem} \): this algorithm gets as input two full polynomials \( f \) and \( g \), \( \deg_x g \leq \deg_x f \). In output it produces the polynomial \( r \) together with the following six values: \( r, \delta, \lambda, g, \bar{g}, w \). \( r \) is the full part of trail or usual pseudo-remainder depending on which way is better (usual pseudo-remainder algorithm doesn’t exclude "superfluous" powers of \( x \)); \( \delta = \deg_x f - \deg_x g; \lambda = \text{trailDeg}_x(\text{prem}(f, g)) \) if usual pseudo-division is used and \( \text{trailDeg}_x(\text{prem}(f^*, g^*)) \) if trail pseudo-division was performed; \( g \) and \( \bar{g} \) are \( \text{lc}_x g \) and \( \text{tc}_x g \) or \( \text{tc}_x g \) and \( \text{lc}_x g \) depending on the way of division: first pair in the usual case and the second in the trail one; \( w \) is marker of kind of division: \( \text{lead or trail} \). Formally we can write it in the following way:

Algorithm \( \text{genPRem} \)
Input: \( u, v \) are full polynomials, \( \deg_x(u) \geq \deg_x(v) \)
Output: the generalized pseudo-remainder
if relativeSize(lcₓ(u)) ≤ relativeSize(tcₓ(u)) then
    \( w := \text{prem}(u, v); \)
    return\( (w/x^{\text{trailDeg}_{x}(w)}, \text{deg}_{x}(u) - \text{deg}_{x}(v), \text{trailDeg}_{x}(w), \text{lc}_{x}(u), \text{tc}_{x}(u), \text{lead}) \)
else
    \( w := \text{prem}(u^*, v^*); \)
    return\( ((w/x^{\text{trailDeg}_{x}(w)})^*, \text{deg}_{x}(u) - \text{deg}_{x}(v), \text{trailDeg}_{x}(w), \text{tc}_{x}(u), \text{lc}_{x}(u), \text{trail}) \)
fi;

Here relativeSize is a integer characteristic of some term which says how big it is. For example, the amount of memory which takes the term can be used.

2 Generalized subresultant algorithm and generalized subresultants

Let \( f, g \) be the initial full polynomials, \( \text{deg}_{x}f \geq \text{deg}_{x}g \). Let \( u_1 = \tilde{u}_1 = \bar{u}_1 = f, u_2 = \tilde{u}_2 = \bar{u}_2 = g, \tilde{u}_3, \bar{u}_4, \ldots \) be the sequence of generalized remainder: \( \tilde{u}_i = \text{genPRem}(\bar{u}_{i-2}, \tilde{u}_{i-1}) \). Of course, the elements of this sequence contains removable factors, we need this sequence just to define the sequence \( \delta_i \): we denote \( \delta_i = \text{deg}_{x}u_{i+1} - \text{deg}_{x}\tilde{u}_i, S_{m}^{n} = \sum_{i=m}^{n} \delta_i, \lambda_i = \lambda - \text{value of} \ \text{genPRem}(\tilde{u}_{i-1}, \bar{u}_i) \). As in the subresultant algorithm we will investigate the determinants of matrices which consist of coefficients of polynomials \( x^}{a}f, x^}{b}g \): let

\[
M_k = \begin{pmatrix}
x^k f \\
x^{k-1} f \\
\vdots \\
f \\
x^{k+\delta_1} g \\
x^{k+\delta_1-1} g \\
\vdots \\
g
\end{pmatrix} = \begin{pmatrix}
* & * & * & * & \cdots & * & * \\
* & * & * & * & \cdots & * & * \\
\vdots \\
* & * & * & \cdots & * & * & * \\
* & * & \cdots & * & * \\
\vdots \\
* & * & \cdots & * & * & * \\
* & * & \cdots & * & * & * \\
\vdots \\
\end{pmatrix}
\]

where \( k < \text{deg}_{x}g \). We denote by \( (u_1, u_2)^j \) the polynomial whose coefficients are obtained by fixing some \( a < \text{rows}(M_j) \) columns in the left part of \( M_j \).
From this formula we see, for example, that $\bar{u}_h$ we want to determine how $u_{\lambda}$ in our considerations, we know that there is some $a$. $(u_1, u_2) S_2^k$ will be denoted as $\bar{u}_{k+2}$. Our goal is to express such polynomial via taking generalized pseudo-remainders. Most of equations bellow will be true up to the sign – the sign is not important in our considerations and it’s checking is redundant. The following relation will be usefull for us: it describes what is happened when we perform the generalized pseudo-division in the matrix:

$$
(\bar{u}_1, \bar{u}_2) S_2^k = \frac{\bar{g}_2 \bar{g}_2 \delta_{1+\delta_{2-\lambda_2}}}{\bar{g}_2} (u_2, u_3) S_3^k = \frac{\bar{g}_2}{\bar{g}_2} \frac{1}{\bar{g}_2 \bar{g}_2} (u_2, u_3) S_3^k.
$$

(1)

From this formula we see, for example, that $\bar{u}_4 = (u_1, u_2) S_2^2 = \frac{\bar{g}_2^2}{\bar{g}_2^2} \frac{1}{\bar{g}_2 \bar{g}_2} u_4$, where $u_4 = \text{genPRem}(\bar{u}_2, \bar{u}_3)$. Let $u_i$ denote genPRem($\bar{u}_{i-2}, \bar{u}_{i-1}$). We want to determine how $u_i$ linked with $\bar{u}_i$. Let us fix the number $k$. Then we can write down the following sequence of equations:

$$
\bar{u}_{k+1} = (\bar{u}_1, \bar{u}_2) S_2^{k-1} = G_{4}^{k+1}(\bar{u}_2, \bar{u}_3) S_3^{k-1} = \cdots
$$

$$
G_{i+1}^{k+1}(\bar{u}_{i-1}, \bar{u}_i) S_i^{k-1} = \cdots = G_{k+1}^{k+1}(\bar{u}_{k-1}, \bar{u}_k) 0 = G_{k+1}^{k+1} u_{k+1}.
$$

Now we proceed the same transformations with $k$ instead of $k - 1$ and simultaneously we will express $G_{i+2}^{k+2}$ via $G_{i+1}^{k+1}$ using the $\Pi$:

$$
\bar{u}_{k+2} = (\bar{u}_1, \bar{u}_2) S_2^k = \frac{1/G_3^2 \delta_k}{g_2^{\delta_{1+\delta_2-\lambda_2}}} G_{4}^{k+1}(\bar{u}_2, \bar{u}_3) S_3^k = \cdots
$$

$$
= \left( \prod \frac{1/G_j^2}{g_{j-1}^{\delta_{j-2+1}}} \right) \frac{\delta_k}{g_k^{\delta_k}} \frac{1/G_{k+1}^{k+1}}{g_{k-1}^{\delta_{k-1}}} G_{k+1}^{k+1}(\bar{u}_k, \bar{u}_{k+1}) 0 = G_{k+2}^{k+2} u_{k+2}.
$$

Hence

$$
G_{k+2} = \frac{\bar{g}_k^{\lambda_k} \lambda_k}{g_k^{\delta_k}} \frac{1}{g_k^{\delta_k}} \left( \frac{1}{\prod G_j^2 g_{j-1}^{\delta_{j-2+1}}} \right) \frac{1}{G_{k+1}^{k+1} g_k^{\delta_{k-1}}} \delta_k.
$$

Let us denote the expression with product as $h_{k+2}$:

$$
h_{k+2} = \prod G_j^2 g_{j-1}^{\delta_{j-2+1}} G_{k+1}^{k+1} g_k^{\delta_{k-1}}.
$$

(2)
\[ h_{k+2} \text{ is } "\text{integer}" \text{ as it is equal to the determinant with } "\text{integer}" \text{ entries:} \]

\[
(\bar{u}_1, \bar{u}_2)_{S^k_2-1} = G^3 \delta^{k+1}_4 G^{k+1}_4 (\bar{u}_3, \bar{u}_4)_{S^k_4-1} = \ldots
\]

\[
= \prod G^i_j G_{j-1}^{k-2} G_{k-1}^{k+1}(\bar{u}_{k-3}, \bar{u}_{k-2})_{S^{k-2}_k-1} = \prod G^i_j G_{j-1}^{k-2} G_{k-1}^{k+1}(\bar{u}_{k-2}, \bar{u}_{k-1})_{S^{k-1}_k-1} = \ldots
\]

and taking the leading or trailing coefficient we get \( h_{k+2} \).

We can remark here that from the (3) it follows that \((\bar{u}_1, \bar{u}_2)_{S^{k-1}_2-1} \sim \bar{u}_k\) and as we know one of its coefficient, we can compute it from the \( \bar{u}_k \).

Analyzing the view of matrices \( M_i, S^{k-1}_2 < i < S^k_2 - 1 \) (namely, the presence of zero’s on the “leading” or “trailing” diagonals) we see that we can fix the columns in such a way that \((\bar{u}_1, \bar{u}_2)^i = 0 \) for that \( i \), so the structure of the sequence of \((\bar{u}_1, \bar{u}_2)^i \) is analogue to the one of usual subresultants.

From the (2) it follows the law of \( h_k \) transformation:

\[
h_{k+2} = h_{k+1} g_{k-1} G_{k+1}^k g_k \delta_{k-1} = h_{k+1} g_{k-1} \frac{1}{h_{k+1}^\delta_{k-1}} \frac{1}{g_{k-1}^\delta_{k-1}} \frac{g_k}{g_{k-1}^\delta_{k-1}} = \frac{g_k}{g_{k-1}^{\delta_{k-1}}} \frac{1}{h_{k+1}^\delta_{k-1}}.
\]

From the considerations above we can derive the algorithms for computing the gcd and resultants. Bellow we present the algorithm for gcd computation. (we present in the style a la Algorithm C from [1]):

Algorithm C’

Input: \( f, g \) are polynomials
Output: the gcd of \( f \) and \( g \)

C’1. [Reduce to full and primitive.] \((u,v):=(f,g), \, d:=\gcd(\text{cont}(u), \text{cont}(v))\), \( e:=\min(\text{trailDeg}_x(u), \text{trailDeg}_x(v)) \), replace \((u,v)\) by \((\text{primpart}(u)/x^{\text{trailDeg}_x(u)}, \text{primpart}(v)/x^{\text{trailDeg}_x(v)})\). If \( \deg_x(u) < \deg_x(v) \) then replace \((u,v)\) by \((v,u)\). Set \( h:=1, g:=1, \bar{g}=1, G:=1, \bar{G}=1 \).

C’2. [General pseudo-remainder.] Apply \( \text{genPRem}(u,v) \) and assign \( r, \delta, \lambda, g_2, \bar{g}_2, w \). If \( r=0 \), then return \( dx^w v/\text{cont}(v) \).

C’3. [Adjust remainder.] \( u:=v; \, v:=(r \bar{G})/(G g^\delta) \, g:=g_2; \, \bar{g}:={\bar{g}_2}; \, h:={\bar{G} g^\delta}/(G h^{\delta-1}); \, G:=g^\lambda; \, \bar{G}:={\bar{g}^\lambda}; \) go to C’2

5
In the algorithm for computing the resultant of two full polynomials the algorithm is almost the same, but one need to return the value of $h$.

For the non-full polynomials the following formula for resultant can be used: $\text{res}_x(xu,v) = tc_x(v)\text{res}_x(u,v)$ (up to the sign, of course).

## 3 Implementation

The algorithms for gcd and resultant computing above was implemented with the Axiomxl computer algebra system, which allows to get an efficient executing code. As a coefficient ring it was used the ring of polynomials $\mathbb{Z}[y]$. In Axiomxl there are two different structures for dense and sparse polynomials. As a relativeSize it was used the degree for dense polynomials and number of non-zero terms for sparse polynomials. The results of testing is the following: in the case of dense polynomials the algorithm is not slower than the usual subresultant algorithm; on some examples it is times faster than the usual subresultant algorithm.

## 4 One property of generalized subresultants.

Here we change the notation and will notate the generalized subresultants as $S^*_k$ to underline the analogues with usual subresultants. $S^*_k$ means that we get the generalized subresultant from the matrix for the usual subresultant $S_k$. The well known property of usual subresultants is that there formal leading coefficients (principal resultants) $\text{flc}_x(S_k)$ allows one to check the degree of gcd [2]. The generalized subresultants have the same property, namely, the following lemma can be proved:

**Lemma.** Let $S_k^*$ be the sequence of generalized subresultants of two full polynomials $A$ and $B$. Then $\deg_x(\gcd(A, B)) = d$ iff $(\text{flc}_x(S_0^*)$ or $\text{ftc}_x(S_d^*)) = \cdots = (\text{flc}_x(S_{d-1}^*) or \text{ftc}_x(S_{d-1}^*)) = 0$ and $\text{flc}_x(S_d^*) \neq 0$ (then also $\text{ftc}_x(S_d^*) \neq 0$ and back); here $\text{flc}_x$ and $\text{ftc}_x$ are formal leading and trailing coefficient, they are some determinants.

The content and proof of the lemma is almost analogous to the corollary 7.7.9 from [2]. We just make here some remarks. Everywhere in the previous to the corollary 7.7.9 lemma’s in [2] the PSC$_i$ appears it can be substituted by formal leading or trailing coefficient of the generalized subresultants. The big role in the proof plays the equation $A(x)T_j(x) + B(x)U_j(x) = C_j(x)$,
where there is some conditions on the degrees of $T_j(x)$, $U_j(x)$ and $C_j(x)$ and which holds when formal leading coefficient of $S_j$ is vanishes. In the our case this equation will be of the form $A(x)T_j(x) + B(x)U_j(x) = x^*C_j(x)$, where $x^*$ means some power of $x$.

5 Ackowlegment

I would like to thank M.Bronstein for usefull discussions, help with Axiomxl and Axiomxl itself and E.V.Zima for providing me with the copy of [1]. This paper was written in the year of 2000 I think.

References

[1] D.E.Knuth The art of computer programming. Volume 2 Seminumerical Algorithms. pp. 428-434.

[2] B.Mishra Algorithmic Algebra Springer-Verlag 1993.