J. Symbolic Computation (1997) **24**, 45–50

A Note on Subresultants and the Lazard/Rioboo/Trager Formula in Rational Function Integration

THOM MULDERS

Institute of Scientific Computing,
ETH Zurich,
Zurich, Switzerland

An ambiguity in a formula of Lazard, Rioboo and Trager, connecting subresultants and rational function integration, is indicated and examples of incorrect interpretations are given.

1. Introduction

In Lazard and Rioboo (1990) the authors present a formula connecting the logarithmic part of the integral of a rational function and certain subresultants. They also give an algorithm based on this formula. The same algorithm was implemented independently by Trager in SCRATCHPAD II (now AXIOM). In this paper it will be shown that this formula is ambiguous and that a wrong interpretation of this formula and the corresponding algorithm can lead to wrong results. In fact the formula has been wrongly interpreted in Geddes *et al.* (1992) and the wrong interpretation has been implemented in AXIOM 2.0.

2. The Formula

Let K be a field of characteristic 0 and $P(x), Q(x) \in K[x]$ such that $\deg(P(x)) < \deg(Q(x))$ and $Q(x)$ square-free. Consider the differentiation $' = \frac{d}{dx}$ on $K(x)$. Let $S(y) \in K[y]$ be the resultant of $Q(x)$ and $P(x) - yQ'(x)$ w.r.t. x and let

$$S(y) = c \prod_{i \in I} S_i(y)^i$$

be the square-free factorization of $S(y)$. Let $R_i(x, y) \in K[x, y]$ denote the remainder of degree i in x appearing in the computation of $S(y)$ by the subresultant algorithm. The formula stated in Lazard and Rioboo (1990) is then

$$\int \frac{P(x)}{Q(x)}\,dx = \sum_{i \in I} \sum_{b : S_i(b) = 0} b \log(R_i(x, b)). \tag{2.1}$$

The ambiguity of this formula lies in the fact that it is not clear what is meant by the above-mentioned remainder of degree i.

3. Subresultants

In this section the facts about subresultants and the Subresultant Polynomial Remainder Sequence (SPRS) that we need will be stated. For a nice treatment of these matters the reader is referred to Brown (1978).

Let $u_0, u_1 \in R[X] \setminus \{0\}$ for some integral domain R and $d_0 = \deg(u_0) > \deg(u_1) = d_1$. Denote by $T_k(u_0, u_1)$ (or T_k when u_0 and u_1 are clear from the context) the kth subresultant of u_0 and u_1. The sequence of all subresultants of u_0 and u_1 then looks as follows:

$$\begin{array}{ccccccccccc}
T_{d_0} = u_0, & T_{d_0-1} = u_1, & 0, & \ldots, & 0, & T_{d_1}, & T_{d_1-1}, & 0, & \ldots, & 0, & T_{d_2}, \\
\downarrow & \downarrow & & & & \downarrow & \downarrow & & & & \downarrow \\
\deg = d_0 & \deg = d_1 & & & & \deg = d_1 & \deg = d_2 & & & & \deg = d_2
\end{array}$$

$$\begin{array}{ccccccccccc}
T_{d_2-1}, & 0, & \ldots, & 0, & T_{d_3}, & T_{d_3-1}, & 0, & \ldots, & 0, & T_{d_4}, & \ldots \\
\downarrow & & & & \downarrow & \downarrow & & & & \downarrow & \\
\deg = d_3 & & & & \deg = d_3 & \deg = d_4 & & & & \deg = d_4 &
\end{array}$$

and $T_{d_i-1} \sim T_{d_{i+1}}$ (where $f \sim g$ means that there are $a, b \in R \setminus \{0\}$ such that $af = bg$). The remainders that are computed in the SPRS of u_0 and u_1 are

$$T_{d_0}, T_{d_0-1}, T_{d_1-1}, T_{d_2-1}, \ldots, T_{d_k-1},$$

where d_{k+1} is the degree of the last remainder $\neq 0$ in any polynomial remainder sequence of u_0 and u_1. In case R is a unique factorization domain we have $T_{d_k-1} \sim \gcd(u_0, u_1)$.

Concerning specialization we have the following fact. When $\phi\colon R \to S$ (R and S integral domains) is a homomorphism such that $\deg(\phi(u_0)) = \deg(u_0)$ and $\phi(u_1) \neq 0$ then for $0 \leq k \leq d_0$ we have $\phi(T_k(u_0, u_1)) \sim T_k(\phi(u_0), \phi(u_1))$.

When d is the degree of a remainder in the SPRS of $\phi(u_0)$ and $\phi(u_1)$ we have that $\deg(T_d(\phi(u_0), \phi(u_1))) = d$ and $T_d(\phi(u_0), \phi(u_1)) \sim \phi(T_d(u_0, u_1))$. Since $\deg(T_d(u_0, u_1)) \leq d$ it follows that $\deg(T_d(u_0, u_1)) = d$. From this fact we see the following relationship between the sequences $T_k(u_0, u_1)$ $(= T_k)$ and $T_k(\phi(u_0), \phi(u_1))$ $(= \tilde{T}_k)$

$$\begin{array}{cccccccccccccc}
\ldots, & T_d, & T_{d-1}, & 0, & \ldots, & 0, & T_e, & T_{e-1}, & 0, & \ldots, & \ldots, & T_f, & T_{f-1}, & \ldots \\
& \updownarrow & \downarrow & & & & \downarrow & \downarrow & & & & \updownarrow & \downarrow & \\
\ldots, & \tilde{T}_d, & \tilde{T}_{d-1}, & 0, & \ldots, & 0, & 0, & 0, & 0, & \ldots, & 0, & \tilde{T}_f, & \tilde{T}_{f-1}, & \ldots
\end{array}$$

where $\deg(T_{d-1}) = e$ and $\deg(\tilde{T}_{d-1}) = f \leq e$. Here $\updownarrow$ means that the degree remains the same under ϕ and $\downarrow$ means that the degree may drop under ϕ.

4. Different Interpretations of the Formula

In equation (2.1) one needs the remainder of degree i in the subresultant algorithm. As is clear from the previous section there can be two subresultants of degree i in the sequence of all subresultants (T_i and T_{j-1} for some $j > i$), while the subresultants in the SPRS have all different degrees. So, if one interprets "the subresultant algorithm" as an algorithm which computes all subresultants then it is not clear which subresultant one has to take. If one interprets "the subresultant algorithm" as the SPRS then there is no choice.

Looking at the proof in Lazard and Rioboo (1990), what one needs is

$$R_i(x,b) \sim \gcd(P(x) - bQ'(x), Q(x)),$$

when i is the degree of the gcd. Taking $u_0 = Q(x)$, $u_1 = P(x) - yQ'(x)$ and $\phi(f(x,y)) = f(x,b)$ it is clear that

$$\gcd(\phi(u_0), \phi(u_1)) \sim T_i(\phi(u_0), \phi(u_1)) \sim \phi(T_i(u_0, u_1))$$

so we can take $R_i(x,y) = T_i(u_0,u_1)$.

If we would take the other choice, i.e. $R_i(x,y) = T_{j-1}(u_0,u_1)$ where $j > i$, we might have (as the following example shows) that $R_i(x,b) = 0$, leading to a wrong result. This choice is implied when we use the SPRS to compute the resultant (which is done in Geddes *et al.* (1992) and in the AXIOM implementation).

Notice that any polynomial equivalent ($\sim$) to $T_i(u_0,u_1)$ which is not mapped to 0 under ϕ would also be suitable as a logarithmic part and thus, since $T_{j-1}(u_0,u_1) \sim T_i(u_0,u_1)$, any polynomial equivalent to $T_{j-1}(u_0,u_1)$ which is not mapped to 0 under ϕ would be suitable.

Example 4.1. *In the following example we will see that the wrong choice of the remainder will lead to a wrong result. Let*

$$P(x) = x^4 + x^3 + x^2 + x + 1$$

and

$$Q(x) = x^5 + x^4 + 2x^3 + 2x^2 + 2x - 2 + 4\alpha$$

where $\alpha = \sqrt{-1+\sqrt{3}}$. *The remainders in the SPRS of* $Q(x)$ *and* $P(x) - yQ'(x)$ *are* $Q(x), P(x) - yQ'(x), R_3(x,y), R_1(x,y)$ *and* $S(y)$ *where*

$$\begin{aligned} R_3(x,y) = {} & (16y^2 - 8y + 1)x^3 + (24y^2 - 10y + 1)x^2 + (36y^2 - 12y + 1)x \\ & +(100\alpha - 52)y^2 - (40\alpha - 21)y + 4\alpha - 2 \end{aligned}$$

and

$$\begin{aligned} R_1(x,y) = {} & \left\{(320\alpha - 288)y^3 - (224\alpha - 184)y^2 + (52\alpha - 40)y - 4\alpha + 3\right\}x \\ & -(224\alpha - 96)y^3 + (104\alpha - 32)y^2 - (12\alpha + 2)y + 1. \end{aligned}$$

Here $R_1(x,y)$ *is in fact the second subresultant. Now* $S(y)$ *is square-free and* $y = \frac{1}{4}$ *is a root (of multiplicity* 1*) of* $S(y)$ *and* $R_1(x,y)$ *is the remainder of degree* 1 *in the SPRS so the formula implies in the integral a term* $\frac{1}{4}\log(R_1(x,\frac{1}{4}))$. *However* $R_1(x,\frac{1}{4}) = 0$, *as one can easily check.*

Computing this example in AXIOM 2.0 will lead to a runtime-error.

5. Some Algorithmic Solutions

We have seen that when using the SPRS to compute the resultant $S(y)$, one cannot just take the remainder of the right degree in that sequence (i.e. T_{j-1} in the previous section) for the logarithmic part of the integral. However, we have seen that any polynomial equivalent ($\sim$) to T_{j-1} which is not mapped to 0 under ϕ would suit. Let us call the final expression in the logarithmic part L, i.e. the final result will have a sum $\sum b \log(L(x, b))$ in it. We will now give some methods to compute such an L.

1. A possible solution is to take for L the primitive part of T_{j-1}. The advantage of this approach is that one gets small coefficients, but the computation of this primitive part might be quite time consuming, especially when algebraic numbers are involved. Doing this we get in our example

$$L = \big\{(320\alpha - 288)y^2 - (144\alpha - 112)y + 16\alpha - 12\big\}x - (224\alpha - 96)y^2 + (48\alpha - 8)y - 4.$$

2. Another possibility is to take T_i for L. T_i can be computed by multiplying T_{j-1} by a constant (i.e. an element $C \in K(y)$), known during the computation of the SPRS. Since both T_{j-1} and T_i are elements of $K[y][x]$, we can perform the multiplication by C by first multiplying by the numerator of C (an element of $K[y]$) followed by exact division by the denominator of C (also an element of $K[y]$). The advantage of this approach is that it is simple and fast. However, the result one gets will be larger in general. In our example this approach yields

$$\begin{aligned} L = \big\{&(6400\alpha^2 - 11520\alpha + 5184)y^4 - (5760\alpha^2 - 9664\alpha + 4032)y^3 \\ &+(1936\alpha^2 - 3072\alpha + 1216)y^2 - (288\alpha^2 - 440\alpha + 168)y + 16\alpha^2 - 24\alpha + 9\big\}x \\ &-(4480\alpha^2 - 5952\alpha + 1728)y^4 + (2976\alpha^2 - 3456\alpha + 816)y^3 \\ &-(656\alpha^2 - 592\alpha + 56)y^2 + (48\alpha^2 - 8\alpha - 22)y - 4\alpha + 3. \end{aligned}$$

3. Yet another possibility is to divide out any factor that might map T_{j-1} to 0 under ϕ. This can be done by repeatedly computing $\gcd(\mathrm{lc}(T_{j-1}), S_i(y))$ and dividing out this gcd. This is essentially the method applied in Bronstein (1997) and seems to be a compromise of the previous methods. In our example we get the same L as by taking primitive parts. Notice that when a gcd g as mentioned above is $\neq 1$ and $\neq S_i(y)$ we might split the sum in the final result even further, according to the factorization $S_i = g\,(S_i/g)$ (i.e. $\sum_{b:S_i(b)=0}$ becomes $\sum_{b:g(b)=0} + \sum_{b:(S_i/g)(b)=0}$). This might be useful in any further processing of the result.
4. Finally, when one splits the result as described above, for the part corresponding to g one could also use the remainder R in the SPRS of next higher degree (see the diagram in the previous section). In fact one only needs to consider the terms up to degree i of R since one knows that higher terms will be mapped to 0 by ϕ. Subsequently one can do the same with R, i.e. compute $\gcd(\mathrm{coeff}(R, i), g)$ and so on. In our example this means that for $b = \frac{1}{4}$ we can take $R_3(x, y)$ and delete the degree 2 and 3 terms (since we know that their coefficients $16y^2 - 8y + 1$ and $24y^2 - 10y + 1$ are 0 when evaluated in $\frac{1}{4}$). We then get for the zeros $\neq \frac{1}{4}$ of $S(y)$ the same L as in the previous method and for $\frac{1}{4}$ we get

$$L = (36y^2 - 12y + 1)x + (100\alpha - 52)y^2 - (40\alpha - 21)y + 4\alpha - 2.$$

It now depends on what kind of result one wants. If one is only interested in some expression for the integral, whatever its size may be, the second approach is certainly the fastest.

However, one might want a smaller expression for the result or one might want to process the result even further. Examples of further processing are making L monic or replacing complex logarithms by real functions (see Rioboo, 1991). In both cases subsequent gcd computations have to be done, in which case it is better for L to be small, so that the third and fourth method are useful.

Below we give the result of all these methods when making L monic. We also give the times it took to compute these results, excluding the time to compute the SPRS but including the time to make L monic. The computations where done in MAPLE V.3 on a SUN Sparcstation 5 with 110 MHz CPU and 64 Mb main memory. For the first three methods we get the same result, i.e.

$$\begin{aligned}&\sum_{b:S(b)=0} b\log\Bigg(x+\frac{1}{76\,090\,729}\{\\&-(3\,166\,362\,384\,896\alpha^3-9\,589\,146\,863\,104\alpha^2+5\,561\,008\,828\,160\alpha+25\,820\,852\,224)b^4\\&+(2\,572\,043\,436\,032\alpha^3-7\,559\,719\,452\,928\alpha^2+4\,149\,998\,074\,624\alpha+152\,749\,784\,768)b^3\\&-(782\,864\,496\,864\alpha^3-2\,219\,904\,899\,648\alpha^2+1\,116\,206\,990\,848\alpha+94\,889\,180\,832)b^2\\&+(105\,949\,632\,800\alpha^3-288\,577\,921\,016\alpha^2+128\,975\,650\,144\alpha+19\,331\,983\,176)b\\&-5\,377\,952\,768\alpha^3+14\,063\,435\,544\alpha^2-5\,525\,914\,060\alpha-1\,188\,274\,675\}\Bigg)\end{aligned}$$

The first method took 8.6 CPU seconds, the second 21.4 and the third 8.2.

For the last method we get the result

$$\begin{aligned}&\sum_{b:y-\frac{1}{4}=0} b\log(x+a)+\sum_{b:(S(y)/(y-\frac{1}{4}))=0} b\log\Bigg(x+\frac{1}{76\,090\,729}\{\\&(197\,271\,647\,360\alpha^3-367\,859\,305\,600\alpha^2-20\,758\,546\,496\alpha+133\,384\,145\,600)b^3\\&-(110\,333\,223\,456\alpha^3-191\,732\,358\,112\alpha^2-37\,918\,237\,872\alpha+85\,408\,371\,744)b^2\\&+(20\,807\,178\,288\alpha^3-33\,322\,639\,792\alpha^2-12\,257\,297\,780\alpha+16\,950\,073\,192)b\\&-1\,326\,544\,080\alpha^3+1\,941\,306\,416\alpha^2+1\,013\,686\,888\alpha-1\,062\,973\,459\}\Bigg)\end{aligned}$$

and it took 8.3 CPU seconds to compute it.

The high time for the second method is due to the fact that the initial expression for L is big, so that inverting L modulo $S(y)$ is expensive. We see that when one wants to make L monic it is worthwhile to make L small so that the other methods are preferable. Also notice that the fourth method yields the 'nicest' result.

Remark 5.1. As is observed in Geddes *et al.* (1992) one can use subresultants in a similar manner when integrating transcendental functions. Again some caution has to be taken when interpreting the formula. See also Bronstein (1997) on this.

Acknowledgement

I would like to thank D. Lazard and the referee for their useful comments on an earlier draft of this paper.

References

Bronstein, M. (1997). *Symbolic Integration I*. Springer-Verlag.
Brown, W.S. (1978). The subresultant PRS algorithm. *ACM TOMS* **4**(3), 237–249.

Geddes, K.O., Czapor, S.R., Labahn, G. (1992). *Algorithms for Computer Algebra*. Kluwer Academic Publishers.

Lazard, D., Rioboo, R. (1990). Integration of Rational Functions: Rational Computation of the Logarithmic Part. *J. Symbolic Comput.* **9**, 113–116.

Rioboo, R. (1991). *Quelques aspects du calcul exact avec des nombres réels*, Thèse de Doctorat de l'Université de Paris 6, Informatique.

Originally received 13 May 1996
Accepted 23 February 1997