J. Symbolic Computation (1990) **9**, 113–115

Integration of Rational Functions: Rational Computation of the Logarithmic Part

D. LAZARD AND R. RIOBOO†

LITP & GRECO de Calcul Formel, Université Pierre et Marie Curie, 4 Place Jussieu, F-75252 Paris Cedex 05

(*Received* 18 *November* 1987)

A new formula is given for the logarithmic part of the integral of a rational function, one that strongly improves previous algorithms and does not need any computation in an algebraic extension of the field of constants, nor any factorisation since only polynomial arithmetic and GCD computations are used. This formula was independently found and implemented in SCRATCHPAD by B. M. Trager.

The Result

Let $P(x)$ and $Q(x)$ be polynomials in x such that Q is square free and $|P|<|Q|$ ($|\,|$ denotes the degree). In most cases P and Q are relatively prime, but we do not need this.

The classical integration formula for P/Q is

$$\int \frac{P(x)}{Q(x)}\,\mathrm{d}x = \sum_{a|Q(a)=0} \frac{P(a)}{Q'(a)} \log(x-a). \tag{1}$$

This formula is not satisfactory under a computing point of view because it introduces more algebraic quantities than necessary. The number $P(a)/Q'(a)$ is called the *residue* of the root a of Q. We introduce the notion of multiplicity of a residue b that is the number of roots a having b as a residue.

Trager (1976) introduced a new formula involving less algebraic numbers: let $S(y)$ be the resultant $S(y)=\mathrm{Res}_x[P(x)-yQ'(x), Q(x)]$; we have

$$\int \frac{P(x)}{Q(x)}\,\mathrm{d}x = \sum_{b|S(b)=0} b \log(R_b(x)), \tag{2}$$

where $R_b(x)=\gcd[P(x)-bQ'(x), Q(x)]$, the gcd being computed in the extension of the field of the coefficients of P/Q by the algebraic b.

The result is simpler than in formula (1); for example if $P=Q'$ it gives the desired result: $\log[Q(x)]$. But the computation of (2) needs several gcd calculations in algebraic extensions, whereas in fact R_b does not depend on the value of b but only on the minimal polynomial of b.

We introduce now our formula. Let

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

be the square free factorisation of S; here c is a constant (the content of S if P and Q have

† Supported by Electronique Serge Dassault, 55 Quai Marcel Dassault, F-92214 Saint Cloud.

0747–7171/90/020113+03 $03.00/0

integer coefficients), and the $S_i(y)$ are non-constant square free polynomials (they are primitive in the case of integer coefficients). Let $R_i(x, y)$ be the remainder of degree i in x appearing in the computation of S by the subresultant algorithm. Our formula is

$$\int \frac{P(x)}{Q(x)}\,\mathrm{d}x = \sum_{i\in I} \sum_{b|S_i(b)=0} b \log (R_i(x, b)). \tag{3}$$

In other words, the polynomials R_b of the formula (2) are polynomials in x and b that depend only on their degree in x and are obtained as a by-product of the resultant computation.

After the submission of this paper, we found it was implemented in the SCRATCHPAD II Computer Algebra System. Trager has discovered it before us, but has only implemented it and not published.

The Proof

PROPOSITION 1. *The multiplicity of b as a root of $S(y) = \mathrm{Res}_x(P(x) - yQ'(x), Q(x))$ is the same as its multiplicity as a residue.*

This follows immediately from the classical formula

$$\mathrm{Res}_x(P(x) - yQ'(x), Q(x)) = \pm c \prod_{a|Q(a)=0} (P(a) - yQ'(a)),$$

where c is a power of the leading coefficient of Q (a constant relative to x and y).

PROPOSITION 2. *If b is a residue then*

$$\gcd(P(x) - bQ'(x), Q(x)) = c \prod (x - a),$$

where c is a constant and the product is taken over all the roots a of Q that have b as residue. Moreover, if the multiplicity of b is i then the above polynomial has degree i.

The roots of the gcd are exactly the roots a of $Q(x)$ such that $Q'(a)b - P(a) = 0$, i.e. b is the residue of $a (Q'(a) \neq 0)$. The polynomial $Q(x)$ being square free, the same is true for the gcd and the equality is proved. The fact that the degree is i is exactly the definition of the multiplicity of a residue.

PROPOSITION 3. *If b is a residue of multiplicity i then*

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

where $R_i(x, y)$ is as in formula (3).

It is a well-known property of the subresultant algorithm that a specialisation of the remainder of degree i is the remainder of degree i of the subresultant algorithm applied to the specialisation of the input polynomials.

With these three propositions, formula (3) is deduced immediately from formula (1) by grouping terms with the same residue.

The Algorithm

We give now the algorithm corresponding to formula (3) for the main case of integer coefficients.

Input: P, Q polynomials in x such that P is square free and degree(P) < degree(Q).
Output a primitive of P/Q.
begin
1. $S := \text{resultant}(P - y * \textit{diff}(Q, x), Q, x)$
   $R(i) :=$ remainder of degree i in this computation.
2. $\text{sqfr}(S) = c \prod_{i \in I} S(i)^i$, where $S(i)$ are non-constant, primitive, square free, and pairwise relatively prime polynomials in y and c is an integer.
3. for i in I do {*optional*}
   begin
   $r(i) := \text{coeff}(R(i), x^i)$; {*polynomial in y*}
   let $d = A * r(i) + B * S(i)$ given by the extended gcd algorithm; {*d is an integer*}
   $R(i) := \text{remainder}(A * R(i), S(i))$;
   $R(i) := \text{primpart}(R(i))$;
   end;

return$\{\sum_i \sum_{b|S_i(b)=0} b * \log[\text{subst}(b, y, R(i))]\}$
end.

The effect of optional step 3 is to provide the following canonical form for the polynomials R_i: the leading coefficient in x does not depend on y; the degree in y is less than the degree of S_i; as a polynomial in x and y R_i has integer coefficients and is primitive. This computation supposes that $\gcd(r_i, S_i)$ is a constant. If it were not true for some b, the degree of $R_i(x, b)$ in x would be less than i; this would be a contradiction with propositions (2) and (3).

REMARK 1. All computations are done without algebraic numbers, even if they appear in the result. This explains the title of this paper.

REMARK 2. This formula was discovered during the implementation by one of us of rational functions integration in system AMP.

REMARK 3. In the result returned by this algorithm the summation over i is expanded, but the summations over b remain implicit.

REMARK 4. Subsequent simplifications may be done, namely by computing the bs that are rational or Gaussian, and expressing (after a simple substitution) the corresponding terms as logarithms and arctangents of rational polynomials

References

Buchberger, B., Collins, G. E., Loos, R. (1982). *Symbolic and Algebraic Computation.* (Computing Suppl. 4.) New York: Springer.
Davenport, J., Siret, Y., Tournier, E. (1987). *Calcul Formel, Systèmes et Algorithmes de Manipulation Algébriques.* Paris: Masson.
Drouffe, J. M. (1986). *AMP Reference Manual Version 6.6.* Gif-sur-Yvette, France: CEN Saclay.
Trager, B. M. (1976). *Algebraic Factoring and Rational Function Integration.* Proc. 1976 Symp. on Symbolic and Algebraic Computation, pp. 219–226. New York: ACM Inc.