Algebraic Factoring and Rational Function Integration

by

Barry M. Trager

Laboratory for Computer Science, MIT
(formerly Project MAC)

Abstact

This paper presents a new, simple, and efficient algorithm for factoring polynomial's in several variables over an algebraic number field. The algorithm is then used iteratively to construct the splitting field of a polynomial over the integers. Finally the factorization and splitting field algorithms are applied to the problem of determining the transcendental part of the integral of a rational function. In particular, a constructive procedure is given for finding the least degree extension field in which the integral can be expressed.

1. Introduction

For many applications in symbolic mathematics it is necessary to explicitly describe all the roots of a polynomial. One approach is to compute the roots numerically to some predetermined accuracy. This is the approach taken in numerical analysis packages but is generally avoided in symbolic manipulations as the basic routines such as greatest common divisor and factoring would then be useless. Another attempt is to try to express the roots in terms of radicals, but this often cannot be done, and even when it can it leads to great problems when simplifications are required. The approach taken here is to describe an extension field of the rationals by the minimal polynomial for some primitive element and then to express all the roots of the polynomials in terms of that element. If an irreducible polynomial over k is normal then all of its roots are rationally expressible in terms of any one of them. Thus the degree of its splitting field is the same as the degree of the polynomial. For other polynomials, however, the degree of its splitting field may be as high as n!. This degree growth tends to make many "computable" problems practical impossibilities. It then becomes very important to operate in as low degree an extension field as possible.

2. Norms and Algebraic Factoring

We assume the existence of some base field of characteristic zero (e.g. the rationals) over which we have constructive procedures for factoring polynomials. We also assume the capability is present to perform basic polynomial operations (e.g. division with remainder) in both the base field and some extension field of finite degree. Our approach is to map a polynomial over an extension field to one of higher degree in the base field such that there is an exact correspondence between their factorizations in their respective fields. We use this correspondence to reconstruct the factorization of the original polynomial over the extension field.

2.1. Definitions

A number α which is algebraic over k satisfies an irreducible polynomial with coefficients in k. $k(\alpha)$ is the field obtained by adjoining α to k. Let $f_\alpha(x)$ be the unique monic, irreducible equation of degree n which α satisfies. The conjugates of α over k are the remaining distinct roots of f_α, α_2, α_3, ..., α_n. If β is any element of $k(\alpha)$ then β can be represented uniquely as a polynomial in α, of degree less than n, with coefficients from k, $P_\beta(\alpha)$. (See [5 pp. 91-94]) The conjugates of β considered as an element of $k(\alpha)$ are $P(\alpha_2)$, $P(\alpha_3)$, ..., $P(\alpha_n)$. If $\beta = P(\alpha)$ we will let β_i represent $P(\alpha_i)$.

A very useful mapping from $k(\alpha)$ to k is the Norm. Norm(β) is the product of all the conjugates of β relative to $k(\alpha)$ over k. If we want to emphasize the fields involved we may also write $\text{Norm}_{[k(\alpha)/k]}(\beta)$. Since the norm is symmetric in the α_i, by the fundamental theorem on symmetric functions it can be expressed in terms of the coefficients of P_α and thus lies in k.

We can extend the definition of norm to polynomials in several variables with coefficients in $k(\alpha)$. Any such function can be expressed as $G(x_1,x_2,\ldots,x_k,\alpha)$ where G is a polynomial in several variables with coefficients in k. Then the Norm(G) is the product of $G(x_1,x_2,\ldots,x_k,\alpha_i)$ over the n conjugates of α.

The definition given above for the Norm of G coincides with the definition of Resultant($P_\alpha(y)$,$G(x_1,\ldots,x_k,y)$) as presented in [11 p. 86]. Collins [1] presents a modular resultant algorithm which can be employed to effectively compute norms. Even if the original polynomial is sparse, its norm is likely to be dense; thus a modular algorithm is probably the optimal choice. Note that norm is a multiplicative function by definition, i.e. Norm(A*B)=Norm(A)*Norm(B). Thus we can extend the norm to rational functions by defining Norm(A/B)=Norm(A)/Norm(B).

We are now ready to prove an extremely important property of norms. (See [14 pp. 19-20])

Theorem 2.1: *If $f(x,\alpha)$ is an irreducible polynomial over $k(\alpha)$ then the Norm(f) is a power of an irreducible polynomial over k.*

Proof: Assume Norm(f)=C(x)*D(x) where gcd(C,D)=1. Let $f_1 = f(x,\alpha_1)$, then Norm(f) = Product(f_i). The polynomial $f=f_1$ divides Norm(f) and since f is irreducible, either f|C or f|D. Assume for concreteness that f|C, i.e. $C=f_1*g_1$. The fields $k(\alpha_1)$ and $k(\alpha_j)$ are canonically isomorphic under a mapping ϕ_j which sends α_1 to α_j and is the identity on k. ϕ_j can be extended to the ring of polynomials in x over those fields and still remain an isomorphism. Since C(x) has all its coefficients in k it is invariant under ϕ_j, but f_1 and g_1 are mapped to f_j and g_j respectively. Thus the equation $C=f_1*g_1$ becomes $C=f_j*g_j$ under ϕ_j. Therefore $f_j|C$ for $1\le j\le n$, but gcd(C,D)=1 implies that $\gcd(f_j,D)=1$ for all j, and in turn gcd(product(f_j),D)=1. But we assumed that D|Norm(f)=product(f_j), so we have shown that D=1. Thus the Norm(f) cannot be split into two relatively prime factors and can only be some power of an irreducible polynomial. ∎

A simple application of the above theorem is for finding minimal polynomials for elements of $k(\alpha)$. If β is an element of $k(\alpha)$ then, as above, $\beta=Q(\alpha)$. Thus $x-\beta$ divides the Norm(x-Q(α))=B(x) and therefore $B(\beta)=0$. The problem remaining is to determine which irreducible factor of B(x) is actually the minimal polynomial of β. The polynomial $x-\beta$ is linear and thus obviously irreducible. By the above theorem we see that B(x) can only be a power of the minimal polynomial for β, $B(x)=P_\beta(x)^k$. $P_\beta(x)$ can be found directly by calculating the gcd(B(x),B'(x)) where B'(x) is the derivative of B(x).

We now turn to the problem of factoring polynomials with coefficients in $k(\alpha)$ assuming we have this capability over k. The ability to perform basic arithmetic in $k(\alpha)$ allows one to compute gcd's of polynomials over $k(\alpha)$ by the Euclidean Algorithm. Thus we can perform a square free decomposition (see [15]) on any polynomial and reduce the factoring problem to square free polynomials.

Our approach to the factorization of $f(x,\alpha)$ is first to make a linear substitution for x so that the norm(f) is square free. We then factor the norm(f) over k. $Norm(f)=G_1(x)\ G_2(x)\ \ldots\ G_r(x)$ with each G_i distinct and irreducible over k. We claim that $g_i(x,\alpha)=\gcd(f,G_i)$ is irreducible over $k(\alpha)$ for all i and that f=product(g_i).

Theorem 2.2: *Let $f(x,\alpha)$ be a polynomial over $k(\alpha)$ such that the Norm(f) is square free. Let $\Pi(G_i(x))$ be a complete factorization of the Norm(f) over k. Then $\Pi(\gcd(f(x,\alpha),G_i(x)))$ is a complete factorization of f over $k(\alpha)$.*

Proof: Let $g_i=\gcd(f(x,\alpha),G_i(x))$, then we must show that each g_i is irreducible and that all the irreducible factors of f are among the g_i. Let v(x) be an irreducible factor of f over $k(\alpha)$. By the previous theorem Norm(v) must be a power of an irreducible polynomial over k, but v|f implies Norm(v)|Norm(f) and the Norm(f) is square free. Therefore the Norm(v) is irreducible and must be one of the G_i. Since the Norm(f) is equal to the product of the norms of each of the irreducible factors of f, each G_j must be the norm of some irreducible factor of f. Assume both $v_1(x)$ and $v_2(x)$ divide $\gcd(f,G_i)$ where v_1 and v_2 are distinct irreducible factors of f. $v_1|G_i$ implies $norm(v_1)|norm(G_i)$, but $G_i(x)$ is a polynomial over k and its norm is $G_i(x)^n$. The $norm(v_1)$ is irreducible over k and divides a power of the irreducible polynomial $G_i(x)$, thus the $norm(v_1)=G_i(x)$. Similarly the $norm(v_2)=G_i(x)$. But $(v_1*v_2)|f$ implies $Norm(v_1*v_2)=G_i(x)^2|Norm(f)$ and this contradicts the assumption that the Norm(f) was square free. Therefore the $\gcd(f,G_i(x))$ must be irreducible for all i. ∎

The only missing step in the previously outlined factoring procedure is finding a linear substitution that makes Norm(f) square free. We claim that Norm(f(x+sα)) is square free for some s in k. We will prove this in two stages, first for f(x) a polynomial over k and then extend the result to polynomials over $k(\alpha)$.

Theorem 2.3: *If f(x) is a square free polynomial with coefficients in k, then there are only a finite number of s in k such that Norm(f(x-sα)) has a multiple root.*

Proof: Let the roots of f(x) be $\beta_1,\beta_2,\ldots,\beta_m$, all distinct; then the roots of $f(x-s\alpha_j)$ are $\beta_1+s\alpha_j,\ldots,\beta_m+s\alpha_j$. Let G(x) = Norm(f(x-sα)) = $\text{product}_i f(x-s\alpha_i)$. Thus the roots of G are $s\alpha_k+\beta_i$ for $k\le n, i\le m$. G can have a multiple root only if $s = (\beta_j-\beta_i)/(\alpha_k-\alpha_m)$ where $k\ne m$. Therefore there are only a finite number of such values. ∎

Lemma 2.4: *If $f(x,\alpha)$ is a square free polynomial with coefficients in $k(\alpha)$ then there exists a square free polynomial g(x) over k such that $f|g$.*

Proof: Let G(x) = Norm(f(x,α)), let $\Pi g_i(x)^i$ be a square free decomposition of G. Then g(x) = $\Pi g_i(x)$ is a polynomial over k. Since f is square free and we have only discarded the multiple factors of G(x), g(x) is divisible by f. ∎

Corollary 2.5: *If $f(x,\alpha)$ is a square free polynomial over $k(\alpha)$ then there are only a finite number of s in k such that Norm(f(x-sα)) has a multiple root.*

Proof: Let g(x) be a polynomial over k as in the lemma. By theorem 2.3 there are only a finite number of s such that Norm(g(x-sα)) has multiple roots. But f|g implies Norm(f(x-sα,α)) divides Norm(g(x-sα)) and thus any multiple root of the former is a multiple root of the latter. ∎

We are now ready to present the entire factoring algorithm, but we will split off the norm computation as a subroutine for later use by other procedures.

Algorithm sqfr_norm

input: f(x,α) a square free polynomial over $k(\alpha)$

output: a positive integer s, g(x,α)=f(x-sα,α), R(x)=Norm(g(x,α)) a square free polynomial over k.

(1) s=0,g(x,α)=f(x,α) [initialize]

(2) R(x)=resultant(P_α(y),g(x,y),y) [Norm,(resultant taken with respect to y)]

(3) If degree(gcd(R(x),R'(x))=0 then return (s,g,R) [sqfr check]

(4) s=s+1, g(x,α)=g(x-α,α),go to (2) ∎

Algorithm alg_factor

input: f(x,α) a square free polynomial over $k(\alpha)$

output: a list of irreducible factors over $k(\alpha)$

(1) (s,g,R)=sqfr_norm(f(x,α))

(2) L=factor(R(x)) [over k, returns list of factors]

(3) If length(L)=1 then return (f) [original poly was irreducible]

(4) For each h_i(x) in L Do

(4.1) $h_i(x,\alpha) = \gcd(h_i(x),g(x,\alpha))$

(4.2) $g(x,\alpha) = g(x,\alpha)/h_i(x,\alpha)$ [performed over $k(\alpha)$]

(4.3) $h_i(x,\alpha) = h_i(x+s\alpha,\alpha)$ [undoes linear transformation]

(5) return (L) ∎

This factoring algorithm is similar to the one presented in van der Waerden [11 pp. 136-7] but computationally more efficient. If one wants to factor a univariate polynomial of degree d over an extension field of degree n, van der Waerden's approach requires computing a norm which is bivariate of degree *nd* in each variable and then factoring it over k. The algorithm presented above leads to the computation and factoring of a univariate norm of degree *nd*. It appears we have the additional cost of finding a linear transformation which makes the norm square free. However, the first step in factoring a bivariate polynomial over the k is to find a substitution for one of the variables which makes the result square free. Thus there is no actual additional cost and this algorithm is always superior to van der Waerden's.

In [13] Paul Wang gives another algorithm for factorization over algebraic number fields. His approach is an extension of his algorithm for factoring over the integers [12]. Wang's algorithm utilized van der Waerden's technique when the minimal polynomial for the algebraic number factored over primes, e.g. x^4+1. He now uses our improved algorithm in this case. In other cases, Wang's algorithm appears somewhat faster than ours, but his is restricted to algebraic numbers. Our algorithm can also be used on algebraic functions and thus has the advantages of increased generality as well as simplicity. We expect to analyze the computing times in both algorithms in the near future.

3. Primitive Elements

Next we will present an algorithm for computing a primitive element for a tower of extension fields. All of the algorithms we have presented have assumed that the extension field we operate in can be described by the adjunction of a single element to our base field k. If our current extension field is $k(\alpha)$ and β is algebraic over $k(\alpha)$ with minimal polynomial $Q_\beta(x,\alpha)$ then $k(\alpha,\beta)$ is the field obtained by adjoining β to $k(\alpha)$. We seek some γ which is algebraic over k such that $k(\gamma)=k(\alpha,\beta)$. The following theorem will prove very useful.

Lemma 3.1: *Let $P_\alpha(x)$ be the minimal polynomial for α over k and β be a root of Q(x,α), a square free polynomial. If $\mathrm{Norm}_{[k(\alpha)/k]}(Q_\beta(x))$ is square free then $\gcd(P_\alpha(x),Q(\beta,x))$ is linear.*

Proof: α is clearly a root of both $P_\alpha(x)$ and $Q(\beta,x)$ since $Q(\beta,\alpha)=0$. Let the other roots of P(x) be α_j for j=2,...,n. If $Q(\beta,\alpha_j)=0$ then β is a root of both $Q(x,\alpha)$ and $Q(x,\alpha_j)$, but Norm(Q) is $\Pi Q(x,\alpha_i)$ and then β is a multiple root of the norm. Since the norm is square free this cannot happen and the only common root of P(x) and Q(β,x) is α. Therefore the gcd(P(x),Q(β,x)) is linear. ∎

Theorem 3.2: *Let $Q_\beta(x,\alpha)$ be the minimal polynomial for β over k(α) and $P_\alpha(x)$ be the minimal polynomial for α over k. If $\mathrm{Norm}_{[k(\alpha)/k]}(Q_\beta(x))$ is square free then $k(\alpha,\beta) = k(\beta)$.*

Proof: We only need to show that α is representable in $k(\beta)$. By the lemma the gcd(Q(β,x),P(x))=x-c, i.e. is linear. c is the only common root of Q(β,x) and P(x), so c=α. But Q(β,x) and P(x) are both polynomials over $k(\beta)$ and so their gcd is over $k(\beta)$. Thus α=c is in $k(\beta)$. ∎

Given an arbitrary β the norm($Q_\beta(x,\alpha)$) may not be square free, but algorithm sqfr_norm will find an integer s and a polynomial g(x,α) such that R(x) = Norm(g(x,α)) is square free. Since Q(x,α) is irreducible over $k(\alpha)$ and sα is in $k(\alpha)$, g(x,α) = Q(x-sα,α) is irreducible over $k(\alpha)$. If we let γ be a root of g(x,α) over $k(\alpha)$ and thus a root of R(x) over k then by theorem 3.2 $k(\alpha,\gamma) = k(\gamma)$. But $\gamma = \beta+s\alpha$ so $k(\alpha,\gamma) = k(\alpha,\beta)$. Thus γ is the primitive element we were looking for and R(x) is its minimal polynomial over k.

The algorithm we present for calculating primitive elements is essentially the same as that presented by Loos [4]. We differ in allowing the minimal polynomial for β to have coefficients over $k(\alpha)$ instead of requiring it to be over k. Loos does not guarantee, as we do, that the polynomial returned be irreducible. To achieve this result we must require that the minimal polynomial for β be irreducible over $k(\alpha)$ not just irreducible over k. In fact, if we examine our algebraic factoring algorithm, we can determine the conditions under which the resultant of two irreducible (over k) polynomials will factor. This will happen if and only if each of the polynomials factors over the extension field determined by the other polynomial. By symmetry, it is sufficient if one of the polynomials factors over the other's extension field.

Algorithm primitive_element

input: $P_\alpha(x)$ the minimum polynomial for α over k

$Q_\beta(x,\alpha)$ the minimum polynomial for β over $k(\alpha)$

output: R(x) the minimum polynomial for γ over k where $k(\alpha,\beta) = k(\gamma)$.

$A(\gamma)$ is a representation of α in $k(\gamma)$

$B(\gamma)$ is a representation of β in $k(\gamma)$

(1) (s,g,R) = sqfr_norm(Q(x,α),P(x))

(2) α=linsolve(gcd(g(γ,x),P(x))) [arithmetic over $k(\gamma)$ where γ denotes a root of R(x), linsolve($ax-b$) returns b/a]

(3) $\beta=\gamma-s\alpha$

(4) return (R(x),γ,α,β) ∎

4. Splitting Fields

The last algorithms to be presented in this section calculate splitting fields. We restrict our considerations to irreducible polynomials since the composite field from many such extensions can be found by repeated application of the primitive_element algorithm. Our basic approach will be to alternately adjoin a root of an irreducible factor of the polynomial to the current extension field and then refactor the polynomial in the new extension field. As linear factors are discovered they are put on a separate list and their coefficients are updated as the extension field changes. If n is the degree of the original polynomial, in the worst case $n-1$ iterations will occur and the primitive element for the splitting field will be of degree n!.

The norm computation at step 4.1 serves a dual purpose. It is used to find a minimal polynomial for the new extension field at step 4.3.2, but it is also the first stage for the algebraic factoring performed in steps 4.2 and 4.3. Thus in the context of splitting field calculations, our factoring algorithm becomes even more efficient.

Algorithm split_field

input: P(x) a polynomial irreducible over k

output: $R_\gamma(x)$ the defining polynomial for the splitting field of P(x) and a list of the roots of P(x) over $k(\gamma)$.

(1) roots = [], polys = [P(x)]

(2) minpoly = P(x), newminpol = P(x), index = 1, $\beta=\gamma$ [γ is a root of minpoly]

(3) replace polys[index] by polys[index]/(x-β), add β to roots, Newfactors = [], k = 1

(4) for each $P_i(x)$ in polys do

(4.1) (g,s,R) = sqfrnorm(P_i(x),minpoly)

(4.2) L = factor (R(x))

(4.3) for each $Q_j(x)$ in L do

(4.3.1) f(x,γ) = gcd(g(x,γ),Q_j(x))

[f is an irred. factor of P_i in $k(\gamma)$]

(4.3.2) if Deg(Q_i)>Deg(newminpol) then do

(4.3.1.1) newminpol = Q_j(x), index=k, new_s=s, Bpoly(x,γ) = f(x,γ)

(4.3.3) g(x,γ) = g(x,γ)/f(x,γ)

(4.3.4) f(x,γ) = f(x+sγ,γ)

(4.3.5) If Deg(f(x,γ)) = 1

(4.3.5.1) then add linsolve(f(x,γ)) to roots

(4.3.5.2) else add f(x,γ) to Newfactors, k=k+1

[let new_γ be a root of newminpol, now we operate in k(new_γ)]

(5) α = linsolve(gcd(minpoly,Bpoly(new_γ,x)))

(6) β = new_γ-new_sα

(7) subst α for γ in roots [update for new extension]

(8) If Newfactors = [] then return (newminpol,roots)

(9) subst α for γ in Newfactors

(10) polys = Newfactors, minpoly = newminpol, γ = new_γ, go to 3. ∎

For comparison, we now present a simpler version of the above algorithm which avoids performing any factoring. If the splitting field is actually of degree n! this approach will actually be faster since the attempt at factorization in step 4.2 would always fail and thus be a waste of time. The essence of the splitting field calculation is repeated primitive

element calculations. In the algorithm above factorization is attempted in the hope that the degree of the splitting field is much less than n!. The algorithm below always returns a polynomial of degree n! known as the resolvent [2]. The irreducible factors of this polynomial over k are all of the same degree, all normal, and are all defining polynomials for the splitting field of P(x).

Algorithm resolvent

input: P(x) a polynomial irreducible over k

output: R(x) a polynomial of degree n! such that any irreducible factor of R defines a splitting field for P

(1) minpoly = P(x), $\beta=\gamma$ [γ is a root of minpoly]

(2) $P(x,\gamma) = P(x)/(x-\beta)$

(3) If degree(P)=0 then Return minpoly

(4) (g,s,R) = sqfrnorm(P(x,γ),minpoly)

[Let new_γ be a root of R(x), now operate in k(new_γ)]

(5) α = linsolve(gcd(minpoly,P(new_γ,x)))

(6) β = new_γ - sα

(7) subst α for γ in P(x,γ)

(8) minpoly = R(x), γ = new_γ, go to 2. ∎

5. Rational Function Integration

We now turn to an application of the algorithms presented in the previous sections. The problem of the symbolic integration of rational functions in the context of algebraic manipulation has been investigated by Manove et al [8], Moses [9], Horowitz [3], Tobey [10], and Mack [6]. By Liouville's theorem the integral of a rational function can be expressed as a rational function plus a sum of complex constants times logs of rational functions.

$$\int R(x)dx = v_0(x) + \sum c_i \log(v_i(x))$$

Algorithms for obtaining the rational part of the integral are well known and reasonably efficient [3], [6], but as far as the author knows, no one has actually presented practical and relatively general algorithms for obtaining the transcendental part. Horowitz limited his investigation to algorithms for obtaining the rational part. Manove and by extension Moses' SIN factored the denominator over the integers and obtained logarithmic terms if the factors were linear or quadratics. Tobey examined the transcendental problem and concluded "There is no generally valid algebraic algorithm for obtaining in a symbolic form the transcendental part of the integral of a rational function." He did however present some algorithms for obtaining the transcendental part in special cases. He also presented as an unsolved problem the problem of obtaining the least degree extension field in which the integration can be done.

Starting where Horowitz leaves off, we are interested in integrating a rational function S(x)/T(x) where T(x) is square free and degree(S(x)) < degree(T(x)). If we were willing to operate in the splitting field of the denominator then we could perform a partial fraction expansion to get $S(x)/T(x) = \sum c_i/(x-\theta_i)$ where each c_i is an element of the splitting field and θ_i is a root of the denominator. In fact c_i is just the residue of R(x) at θ_i. This would lead to an expression for the integral, but could require operating in an extension field of very high degree. Thus for the sake of efficiency and to promote a more intelligible result, we propose to find the minimum degree extension field in which the result can be expressed. We claim that any field which contains all the residues is sufficient for expressing the integral. This is a significant result since the residues may be contained in a field of much lower degree than the splitting field of the denominator.

Our approach to this problem is first to constructively find the extension field E determined by all of the residues of the integrand. Then we factor the denominator of the integrand over this field and perform a partial fraction decomposition. This breaks the original integral into a sum of integrals of proper rational functions where each denominator is irreducible over E. We claim that each integral in the sum can be expressed as a single log term.

First we will find a simple expression for the residues in terms of the roots of the denominator T(x). Let θ be a root of the square free polynomial T(x). Since θ must then be a simple root, the residue of S(x)/T(x) at θ is $S(\theta)/T'(\theta)$. Since T(x) is square free, the gcd(T(x),T'(x))=1. Thus by the extended Euclidean algorithm we can find polynomials A(x) and B(x) over k such that A(x) T(x) + B(x) T'(x) = S(x), with the degree(B(x)) < degree(T(x)). $T(\theta) = 0$ implies $B(\theta) = S(\theta)/T'(\theta)$. Since θ was an arbitrary root of T(x) we have established the following:

Lemma 5.1: *Let S(x) and T(x) be polynomials over* k, *where T(x) is square free. Then there exists a polynomial B(x) over* k *such that for any root θ of T(x) the residue of S(x)/T(x) at θ is $B(\theta)$.*

Theorem 5.2: *Let S(x)/T(x) be a quotient of polynomials where T(x) is irreducible over some ground field k. If all the residues are contained in k then all the nonzero residues are equal.*

Proof: Let B(x) be the polynomial described in the lemma. Let the roots of T(x) be θ_i for i=1,...,n. We will operate in the splitting field E of T(x). Since T(x) is irreducible, its Galois group $G(E/k)$ is transitive and there is an automorphism ϕ_j sending θ_1 to θ_j for each j and leaving k unchanged. Let $c = B(\theta_1)$, the residue at θ_1. If we apply ϕ_j to this equation, the left side is unchanged since c is in k and the right side is mapped to $B(\theta_j)$ since the coefficients of B(x) are in k. Thus $c = B(\theta_j)$ for each j and all the residues are equal. ∎

Corollary 5.3: *With S(x) and T(x) as in Theorem 5.2, $\int S(x)/T(x)dx$ is expressible over k.*

Proof: Let c be the common residue of the θ_j, then the integrand has the partial fraction decomposition, $S(x)/T(x) = \sum c/(x-\theta_j)$. This integrates to the following:

$$\int S(x)/T(x)dx = \sum c\log(x-\theta_i) = c \log \Pi(x-\theta_i) = c \log T(x) \ .$$ ∎

All that remains is to transform the integrand so that the corollary applies. If θ is any root of the denominator $T(x)$ then $B(\theta)$ is the residue of $S(x)/T(x)$ at γ, therefore the resultant of $x-B(y)$ and $T(y)$ with respect to y is a polynomial whose roots are precisely the residues of $S(x)/T(x)$. Using the algorithms presented in the last section we can compute the splitting field of this polynomial and therefore the least degree extension which contains all the residues. Then we factor the denominator of the integrand over this extension field. If we performed a partial fraction decomposition on the resulting rational function, each term in the sum would satisfy the hypotheses of the corollary and thus be directly integrable. But the actual computation of the partial fraction decomposition is unnecessary. Since we know the form of the result as $\sum c_i\log(f_i(x,\gamma))$ where the f_i's are the irreducible factors of the denominator, all we need to be able to do is compute each c_i. But c_i is $B(x)$ evaluated at any root of $f_i(x,\gamma)$. The resultant$(B(x),f_i(x,\gamma),x)$ is the product of $B(x)$ evaluated at each of the roots of f and thus equals c_i^k where k is the degree of f. Therefore $g(y) =$ resultant$(y-B(x),f_i(x,\gamma),x)$ is $(y-c_i)^k$, and $y-c_i$ is $g(x)/Gcd(g(x),g'(x))$.

Algorithm ratint

input: T(x) a square free polynomial, S(x) a polynomial of lower degree than T(x)

output: R(x) the minimal polynomial for the splitting field of the residues, γ such that $R(\gamma) = 0$, and $I(x,\gamma)$ the integral expressed in terms of γ.

(1) (A,B) = Extended_Euclidean(T(x),T'(x),S(x))

(2) R(x) = split_field (resultant(x-B(y),T(y),y))

(3) L = alg_factor(T(x),R(x),γ)

(4) L = map(Int_log,L) [applies function Int-log to each element in L]

(5) Return(R(x),γ,sum(L)) ∎

Algorithm int-log

input: $D(x,\gamma)$ an irreducible polynomial over $k(\gamma)$

output: c log $D(x,\gamma)$

(1) c(y) = resultant(y-B(x),D(x,γ),x)

(2) c = linsolve(c(y)/Gcd(c(y),c'(y))) [common residue expressed in terms of γ.]

(3) return (c log $D(x,\gamma)$) ∎

On page III-10 of his thesis, Tobey presents a rational function which he demonstrates is integrable over Q(sqrt(2)). He asks how one determines a priori the extension of least degree in which the integral can be expressed. Using the MACSYMA [7] system and the ideas presented in this section, we solve his problem below.

```
(C1) INTEGRATE(S(X)/T(X),X);/* THIS IS TOBEY'S INTEGRAL */
(D1)

  /      13       8      7      6      3      2
  [ 7 X   + 10 X  + 4 X  - 7 X  - 4 X  - 4 X  + 3 X + 3
  I ------------------------------------------------- DX
  ]      14       8      7      4      3    2
  /   X   - 2 X  - 2 X  - 2 X  - 4 X  - X  + 2 X + 1

(C2) (ALGEBRAIC:TRUE,TELLRAT(T(C)))$ /* LET C BE A ROOT OF
                                         T(X) */

(C3) B(C):=''(RATSIMP(N(C)/DIFF(D(C),C))); /* B(X) IS THE
            POLY COMPUTED IN STEP 1 OF RATINT */
(D3) B(C) :=

     12    11   10    9    8    7    6      2
    C   - C  + C  - C  + C  - C  - C  - 2 C  - 2 C + 2
    ------------------------------------------------
                            2

(C4) RESULTANT(Y-B(X),D(X),X);
           14           13           12           11
(D4) 16384 Y   - 114688 Y   + 315392 Y   - 401408 Y

          10           9           8          7          6
+ 164864 Y   + 121856 Y  - 109312 Y  - 23552 Y  + 27328 Y

        5         4         3        2
+ 7616 Y  - 2576 Y  - 1568 Y  - 308 Y  - 28 Y - 1

(C5) SQFR(%); /* SQUARE FREE DECOMPOSITION ISN'T NECESSARY
      BUT MAKES STRUCTURE MORE EVIDENT IN THIS EXAMPLE */
                        2       7
(D5)                (4 Y  - 4 Y - 1)

(C6) MP(X):=''(SUBST(X/2,Y,PART(%,1))); /* THIS IS THE
       MINIMAL POLY. FOR THE RESIDUES, MONICIZED FOR
       EFFICIENCY. */
                          2
(D6)              MP(X) := X  - 2 X - 1

(C7) FACTOR(T(X),MP(ALG)); /* ALGEBRAIC FACTORIZATION OF
       DENOMINATOR OF INTEGRAND */
      7             2              7              2
(D7) (X  + (1 - ALG) X  - ALG X - 1) (X  + (ALG - 1) X

                                        + (ALG - 2) X - 1)

(C8) (F1:PART(%,1),F2:PART(%,2)); /* F1 IS ITH FACTOR */
           7             2
(D8)      X  + (ALG - 1) X  + (ALG - 2) X - 1

(C9) TELLRAT(MP(ALG))$ /* NEXT WE CALCULATE RESIDUES OVER
                                  k(ALG) */

(C10) RESULTANT(Y-B(X),F1,X);
          7              6                    5
(D10) 128 Y  - 448 ALG Y  + (1344 ALG + 672) Y

                          4                        3
 + (- 2800 ALG - 1120) Y  + (3360 ALG + 1400) Y

                         2
 + (- 2436 ALG - 1008) Y  + (980 ALG + 406) Y

 - 169 ALG - 70
```

```
(C11) SOLVE(X,Y);
SOLUTION

                              ALG
(E11)                    Y = ---
                               2
MULTIPLICITY 7
(D11)                     [E11]

(C12) C1:EV(Y,X);/* C1 IS THE RESIDUE AT ANY ROOT OF F1 */

                              ALG
(D12)                         ---
                               2

(C13) SOLVE(RESULTANT(Y-B(X),F2,X),Y);
SOLUTION

                                ALG - 2
(E13)                    Y = - -------
                                  2
MULTIPLICITY 7
(D13)                     [E13]

(C14) C2:EV(Y,X); /* C2 IS RESIDUE AT ANY ROOT OF F2 */
                              ALG - 2
(D14)                       - -------
                                 2

(C15) C1*LOG(F1)+C2*LOG(F2); /* FINALLY WE CAN EXPRESS
                                THE INTEGRAL */
                7                 2
     ALG LOG(X  + (1 - ALG) X  - ALG X - 1)
(D15) ------------------------------------
                        2

                                 7                2
       (ALG - 2) LOG(X  + (ALG - 1) X  + (ALG - 2) X - 1)
     - ---------------------------------------------------
                                 2

(C16) (ALGEBRAIC:FALSE,SOLVE(MP(ALG))); /* SINCE MP(X) IS
     QUADRATIC, WE CAN EXPRESS THE ANSWER IN RADICALS */
SOLUTION

(E16)                 ALG = 1 - SQRT(2)

(E17)                 ALG = SQRT(2) + 1
(D17)                     [E16, E17]

(C18) EV(D15,E17);
(D18)

                    7              2
(SQRT(2) + 1) LOG(X  - SQRT(2) X  - (SQRT(2) + 1) X - 1)
-----------------------------------------------------
                           2

                      7              2
  (SQRT(2) - 1) LOG(X  + SQRT(2) X  + (SQRT(2) - 1) X - 1)
- -----------------------------------------------------
                           2
```

We have shown that any integral can be expressed over the field generated by the residues of the integrand. Now we must justify our claim that this field is the minimal one, i.e. we will show that if an integral can be expressed over a field k, then all the residues of the integrand lie in k.

Let $v_0(x) + \sum c_i \log(v_i(x))$ be the integral of a rational function, and let all the coefficients be contained in some field k. Then the integrand can be expressed as $v'_0 + \sum c_i v'_i/v_i$. The residue of v'_0 is zero everywhere, while the residue of v'/v is always a rational integer. Thus all the residues of the integrand can be expressed as integer linear combinations of the c_i and thus must lie in k. Combined with our earlier result, we have that the field determined by the residues is both necessary and sufficient for expressing the integral.

6. Extensions and Comments

The algorithms presented in this paper were designed to operate over an arbitrary base field. If we are interested in factoring univariate polynomials over algebraic number fields then we let our base field be the rational numbers. Given the capability to factor multivariate polynomial norms over Q as in [12], we can extend to factoring multivariate polynomials over algebraic number fields. If we allow our minimal polynomials to have polynomial coefficients then we can factor polynomials over algebraic function fields.

As algebraic manipulation systems expand their problem domains, the need for performing operations with quantities satisfying algebraic relationships will increase. The basic arithmetic operations can be performed by merely using the side relations to keep the expressions reduced. We have extended factoring to these domains by mapping the problem to a simpler domain while still preserving its structure. Then we were able to lift the factorization back to the original expression. That finding such unramified morphisms can lead to efficient algorithms for algebraic manipulation has been amply demonstrated by the recent development of modular and p-adic techniques [15].

I would like to thank Joel Moses, Richard Zippel, and David Barton for many enlightening discussions. Work herein was supported in part by the United States Energy Research and Development Administration contract E(11-1)-3070.

References

[1] Collins, G.E., "The Calculation of Multivariate Polynomial Resultants", *JACM*, vol. 18, no. 4, Oct. 1971, pp. 515-532.

[2] Gaal, L., *Classical Galois Theory with Examples*, Markham, Chicago, 1971, reprinted by Chelsea, New York.

[3] Horowitz, E.,*Algorithms for Symbolic Integration of Rational Functions*, Ph.D. Thesis, U. of Wisconsin, 1970.

[4] Loos, R. G. K., "A Constructive Approach to Algebraic Numbers", Computer Science Dept., Stanford University, Palo Alto, Calif.

[5] MacDuffee, C., *An Introduction to Abstract Algebra*, Dover, 1966.

[6] Mack, D., *On Rational Integration*, Computer Science Dept., Utah Univ., UCP-38, 1975.

[7] *MACSYMA Reference Manual*. Mathlab Group, Project MAC, M.I.T., Cambridge, Mass., November 1975.

[8] Manove, M., Bloom, S., and Engelman, C., "Rational functions in MATHLAB", *Proc. IFIP Conf. on Symbolic Manipulation Languages*, Pisa, Italy, 1968.

[9] Moses, J., "Symbolic Integration: The Stormy Decade", *Communications of the ACM*, vol 14, no 8, pp. 548-560, 1971.

[10] Tobey, R.G., *Algorithms for Antidifferentiation of Rational Functions*, Ph.D. Thesis, Harvard, 1967.

[11] van der Waerden, B.L., *Modern Algebra*, vol 1, tr. Fred Blum, Frederick Ungar Publishing Co., New York, 1953.

[12] Wang, P.S. and Rothschild, L.P., "Factoring Multivariate Polynomials Over the Intgers," *Mathematics of Computation*, vol 29, no. 131, pp 935-950, 1975.

[13] Wang, P.S., "Factoring Multivariate Polynomials over Algebraic Number Fields", *Mathematics of Computation*, vol. 30, no. 134, April 1976.

[14] Weyl, Hermann, *Algebraic Theory of Numbers*, Princeton University Press, 1940.

[15] Yun, D.Y.Y., *The Hensel Lemma in Symbolic Manipulation*, Ph.D. Thesis, M.I.T., MAC TR-138, 1974.