Fast algorithms for partial fraction decomposition

H. T. Kung
Carnegie Mellon University

D. M. Tong
NOTICE WARNING CONCERNING COPYRIGHT RESTRICTIONS:
The copyright law of the United States (title 17, U.S. Code) governs the making
of photocopies or other reproductions of copyrighted material. Any copying of this
document without permission of its author may be prohibited by law.
FAST ALGORITHMS FOR PARTIAL FRACTION DECOMPOSITION

H. T. Kung
Carnegie-Mellon University
Pittsburgh, PA 15213

D. M. Tong
The University of Akron
Akron, OH 44304

January 1976

This research was supported in part by the National Science Foundation under Grant MCS75-222-55 and the Office of Naval Research under Contract N00014-76-C-0370, NR 044-422.
ABSTRACT

The partial fraction decomposition of a proper rational function whose denominator has degree $n$ and is given in general factored form can be done in $O(n \log^2 n)$ operations in the worst case. Previous algorithms require $O(n^3)$ operations, and $O(n \log^2 n)$ operations for the special case where the factors appearing in the denominator are all linear.
1. INTRODUCTION

Let

\[ \frac{P(x)}{\prod_{i=1}^{k} Q_i(x)} \]

be a given fraction, where the \( P, Q_i \) are polynomials and the \( \ell_i \) are positive integral exponents such that

1. \( \deg P < \sum_{i=1}^{k} \ell_i \cdot \deg Q_i = n \), and

2. \( Q_1, \ldots, Q_k \) are relatively prime.

The general partial fraction decomposition problem (general PF problem) is to compute the coefficients of the polynomials \( C_{i,j} \) for \( i = 1, \ldots, k \) and \( j = 1, \ldots, \ell_i \) such that

\[ \frac{P(x)}{\prod_{i=1}^{k} Q_i(x)} = \sum_{i=1}^{k} \sum_{j=1}^{\ell_i} \frac{C_{i,j}(x)}{Q_i(x)} \]

with \( \deg C_{i,j} < \deg Q_i \) for all \( i,j \). The existence and uniqueness of the polynomials \( C_{i,j} \) are well known (see, e.g., van der Waerden [1953]). There are enormous applications of partial fractions in applied mathematics and in network theory (see, e.g., Henrici [1974] and Weinberg [1962]). This paper gives fast algorithms for solving the general partial fraction expansion problem when \( n \) is large.

Previous algorithms for the problem usually involve solving systems of linear equations (see Henrici [1974] for a nice summary). Hence they take \( O(n^3) \) arithmetic operations, or \( O(n^{2.81}) \) operations if Strassen's method (Strassen [1969]) is used. For the special case that the \( Q_i \) have either
degree one or two, many algorithms were known: see, e.g., Schwatt [1924],
Turnbull [1927], Hazony and Riley [1959], Pottle [1964], Pessen [1965],
Brugia [1965], Moad [1966], Valentine [1967], Wehrhahn [1967], Kami [1969]
and Linner [1974]. But these algorithms still take \(O(n^2)\) or more operations.

Recently Chin and Ullman [1975] showed that in case that all the \(Q_i\)
have degree one the problem can be done in \(O((n \log n)^{3/2})\) operations. This
bound was further improved by Chin in his thesis (Chin [1975]). He showed
that if the \(Q_i\) are all linear, then the problem can be done in \(O((\log k) \cdot (n \log n))\)
operations. However, the assumption that the \(Q_i\) are all linear factors is
crucial in his methods. Hence the problem of solving the general PF problem
(without assuming that the \(Q_i\) are linear) in \(O(n^2)\) operations is stated as an
unsolved problem in his thesis. Note that the general PF problem does occur
frequently in practice. For example, if we work over the field of real num-
bers, then the factors \(Q_i\) certainly can have either degree one or two. (See
also Grau [1971] and Henrici [1971] for more examples.) In this paper, we
show that the general PF problem can done in \(O((\log n) \cdot M(n))\) operations in
the worst case. \(M(n)\) is any upper bound on the number of operations needed
to multiply two \(n\)th degree polynomials, which satisfies some mild regularity
condition (see Section 2). In particular, if an FFT algorithm is used for
polynomial multiplication (see, e.g., Knuth [1969], Borodin and Munro [1975]),
then we have \(M(n) = O(n \log n)\), which satisfies the regularity condition, and
hence the general PF problem can be done in \(O(n \log^2 n)\) operations. Moreover,
we note that for the special case where the \(Q_i\) are all linear, our approach
will lead to Chin's \(O((\log k) \cdot (n \log n))\) algorithm.

Basic assumptions and preliminary lemmas used in this paper are intro-
duced in Section 2. In Section 3, the solution of the general PF problem is
reduced to the solution of two simpler problems, problem P1 and problem P2,
and precise statements of the main results of the paper are given. An algorithm, based on a new theorem (Theorem 4.1), for solving problem P1 is presented in Section 4. Section 5 contains an algorithm for solving problem P2. Finally, an important special case of problem P2 is solved in Section 6.
2. BASIC ASSUMPTIONS AND PRELIMINARY LEMMAS

We assume throughout the paper that polynomials are over some field $K$, are denoted by upper case letters, and are given in the form $P(x) = \sum p_i x^i$ where $p_i \in K$. To compute $P$ or $P(x)$ means to find all the coefficients of $P$. We assume that $M(n)$ is an upper bound on the number of operations needed to multiply two $n$th degree polynomials. Given relatively prime polynomials $A_1, A_2$ with $\deg A_1, \deg A_2 \leq n$, let $F(n)$ be an upper bound on the number of operations to find polynomials $F_1, F_2$ such that

$$F_2 \cdot A_1 + F_1 \cdot A_2 = 1$$

with $\deg F_1 < \deg A_1$ and $\deg F_2 < \deg A_2$. The existence and uniqueness of $F_1$ and $F_2$ are well-known (see, e.g., van der Waerden [1953]).

Let $Z^+$ be the set of all nonnegative integers and let $G: Z^+ \rightarrow Z^+$ be a nondecreasing function. We say $G$ satisfies Condition C, if

$$G(n) = n \cdot H(n)$$

for some nondecreasing function $H: Z^+ \rightarrow Z^+$. We assume that $M$ satisfies Condition C. Similar regularity conditions are usually assumed (see, e.g., Aho, Hopcroft and Ullman [1974, p.280], Brent and Kung [1976] and Moenck [1973b]).

There are many algorithms for polynomial multiplication. For example, the classical algorithm gives $M(n) = c_1 n^2$, binary splitting multiplication gives $M(n) = c_2 n^{1.585}$, and if the field $K$ is algebraically closed, then FFT multiplication gives $M(n) = c_3 n \log n$, where $c_1, c_2, c_3$ are positive constants (see e.g., Fateman [1974]). In all cases $M$ satisfies Condition C. In fact all we need in this paper are some consequences of Condition C. Hence it is possible to weaken our assumption on $M$, if one wishes to do so.
Let \( D(n) \) be the number of operations needed to divide a polynomial of degree \( 2n \) by a polynomial of degree \( n \). Then using Newton's method and the fact that \( M \) satisfies Condition C, one can show the following lemma (see, e.g., Borodin and Munro [1975] and Kung [1974]):

**Lemma 2.1**

\[
D(n) = O(M(n)).
\]

Using the algorithm EGCD in Moenck [1973a], which is a generalization of an algorithm due to Schönhage [1971] for integer GCDs, one can show the following lemma.

**Lemma 2.2**

\[
F(n) = O((\log n) \cdot M(n)).
\]

We shall assume that \( F \) satisfies the condition that

\[
\sum F(n_i) \leq F(\sum n_i)
\]

for any \( n_i \in \mathbb{Z}^+ \). Clearly, if \( F(n) = c \cdot (\log n) \cdot M(n) \) for some positive constant \( c \) as in Lemma 2.2, then \( F \) satisfies the condition. In fact, the required condition in \( F \) is satisfied as long as \( F \) satisfies Condition C.
3. PROBLEMS P1, P2 AND STATEMENT OF RESULTS

Consider the following two instances of the general PF problem defined in Section 1.

**Problem P1:** (This is the general PF problem with $\ell_i = 1$ for all $i$.)

Given the fraction $P/\prod_{i=1}^{k} R_i$ where the $R_i$ are relatively prime and

$$\text{deg } P < \sum_{i=1}^{k} \text{deg } R_i = n,$$

compute the polynomials $C_1, \ldots, C_k$ such that

$$P(x) = \prod_{i=1}^{k} \frac{C_i(x)}{R_i(x)}$$

with $\text{deg } C_i < \text{deg } R_i$ for all $i$.

The decomposition (3.1) is called the incomplete partial fraction decomposition by Henrici [1971, 1974]. Note also that efficient algorithms for solving problem P1 will furnish efficient procedures for factoring polynomials, as observed by Grau [1971].

**Problem P2:** (This is the general PF problem with $k = 1$.)

Given the fraction $P/Q^{\ell}$ where $\text{deg } P < \ell \cdot \text{deg } Q$ compute the polynomials $C_1, \ldots, C_{\ell}$ such that

$$\frac{P(x)}{Q^{\ell}(x)} = \sum_{j=1}^{\ell} \frac{C_j(x)}{Q^j(x)}$$

with $\text{deg } C_j < \text{deg } Q$ for all $j$. 

The following lemma essentially shows that fast algorithms for problems PI and P2 will lead to fast algorithms for the general PF problem. Define \( T(k,n), T_1(k,n) \) and \( T_2(l,\deg Q) \) to be the number of operations needed to solve the general PF problem, problem PI and problem P2, respectively.

**Lemma 3.1**

\[
T(k,n) \leq T_1(k,n) + \sum_{i=1}^{k} [T_2(l_i,\deg Q_i) + O(M(l_i,\deg Q_i))].
\]

**Proof**

The result follows from the observation that general PF problem can be solved in the following way:

1. Multiply \( Q_i(x) \) out for \( i = 1, \ldots, k \). Let the expansion of \( Q_i(x) \) be \( R_i(x) \) for all \( i \).

2. Solve problem PI for the fraction \( \prod_{i=1}^{k} R_i \) and obtain the polynomials \( C_i \) satisfying (3.1).

3. Solve problem P2 for the fractions \( C_i/Q_i, i = 1, \ldots, k \).

Note that each \( Q_i(x) \) can be computed in \( O(M(l_i,\deg Q_i)) \) operations by an algorithm in Brent [1975].

We summarize our results on \( T_1(k,n) \) and \( T_2(l,\deg Q) \) in the following:

(i) \( T_1(k,n) \leq F(n) + O((\log k) \cdot M(n)) \). (Theorem 4.2)

(ii) \( T_1(k,n) = O((\log k) \cdot (n \log n)) \), when the \( R_i(x) \) is given in the form \( (x-z_i^1)^{m_i} \) for all \( i \). (Theorem 4.3)

(iii) \( T_2(l,\deg Q) = O((\log l) \cdot M(l,\deg Q)) \). (Theorem 5.1)

(iv) \( T_2(l,\deg Q) = O(l \log l) \), when \( \deg Q \leq 2 \). (Theorems 6.1 and 6.2)
We have the following results for the general partial fraction expansion problem.

**Theorem 3.1**

The general PF problem can be done in $F(n) + O((\log k) \cdot M(n)) + O((\log \ell) \cdot M(n))$ operations, where $\ell = \max(\ell_1, \ldots, \ell_k)$.

**Proof**

Note that

$$
\sum_{i=1}^{k} (\log \ell_i) \cdot M(\ell_i \cdot \deg Q_i)
$$

$$
\leq (\log \ell) \sum_{i=1}^{k} \ell_i \cdot \deg Q_i \cdot H(\ell_i \cdot \deg Q_i)
$$

$$
\leq (\log \ell) \cdot n \cdot H(n) = (\log \ell) \cdot M(n).
$$

The result follows from (i), (iii) and Lemma 3.1.

**Corollary 3.1**

The general PF problem can be done in $O(n \log^2 n)$ operations.

**Proof**

Note that in Theorem 3.1, $k \leq n$ and $\ell \leq n$. The result follows from the theorem and Lemma 2.2 by letting $M(n) = O(n \log n)$.

$O(n \log^2 n)$ is the best asymptotic bound known for the general PF problem.

**Theorem 3.2**

The general PF problem can be done in $O((\log k) \cdot (n \log n))$ operations, if $Q_i(x) = x - z_i$, for $i = 1, \ldots, k$. 
Proof

The result follows from (ii), (iv) and Lemma 3.1.

The bound in Theorem 3.2 was obtained previously by Chin [1975]. We include it here just to show that his result will emerge as a special case in our general approach. See the remarks at the end of Section 4.
4. AN ALGORITHM FOR PROBLEM P1

We first assume that $P(x) = 1$ in problem P1. Thus we want to find $A_1, \ldots, A_k$ such that

$$\frac{1}{\prod_{i=1}^{k} R_i(x)} = \sum_{i=1}^{k} \frac{A_i(x)}{R_i(x)}$$

with $\deg A_i < \deg R_i$ for all $i$. Note that

$$\left[ \sum_{i=1}^{k} \left[ A_i(x) \prod_{j=1}^{k} R_j(x) \right] \right] = \prod_{i=1}^{k} R_i(x)$$

Define

$$R(x) = \prod_{i=1}^{k} R_i(x),$$

and for each $i = 1, \ldots, k$, define $B_i, D_i$ by

$$R(x) = B_i(x)R_i(x) + D_i(x)$$

where $\deg D_i < \deg R_i$. Note that $D_i(x) \neq 0$, since the $R_i$ are relatively prime.

Suppose that $\deg D_i \geq 1$, i.e., $D_i(x)$ is not a constant. Then (4.2) implies that $D_i$ and $R_i$ are relatively prime, since $R_i$ and $R$ are relatively prime.

Hence there exist unique polynomials $\Lambda_i$ and $E_i$ such that
Theorem 4.1

For $i = 1, \ldots, k$, if $D_i(x) = d_i$ for some constant $d_i$, then $A_i$ is the constant $1/d_i$, else $A_i = \bar{\lambda}_i$.

Proof

We classify the zeros of $R_i$ according to their multiplicities. Let $Z_m$ be the set of zeros of $R_i$ which have multiplicity $m$. (The zeros exist in an algebraically closed extension field of $K$.) Clearly, we have that

$$\sum_{m} m |Z_m| = \deg R_i,$$

where $|Z_m|$ is the number of elements in $Z_m$, and that if $z \in Z_m$ then

$$R_i^{(h)}(z) = 0 \text{ for } h = 0, \ldots, m-1.$$  

Taking derivatives of (4.1) and (4.3), and using (4.5), one can easily show that

$$\sum_{q=0}^{h} \binom{h}{q} A_i^{(q)}(z) \cdot \prod_{j=1}^{k} R_j^{(h-q)}(z) = \delta_{0,h},$$

for $z \in Z_m$ and $h = 0, \ldots, m-1$, where $\delta_{0,h} = 1$ if $h = 0$ and $\delta_{0,h} = 0$ otherwise. Note
that by (4.2) and (4.5),

\[
D_i^{(h-q)}(z) = R^{(h-q)}(z)
\]

\[
= \left( \prod_{j=1, j \neq i}^k R_j^{(h-q)}(z) \right)
\]

for \( z \in \mathbb{Z}_m \), \( h=0, \ldots, m-1 \) and \( q=0, \ldots, h \). Suppose that \( D_i(x) = d_i \) for some constant \( d_i \). Then by (4.6) and (4.8),

\[
A_i^{(h)}(z), d_i = \delta_{0,h}
\]

for \( z \in \mathbb{Z}_m \) and \( h=0, \ldots, m-1 \). Since by (4.4) the number of equations (4.9) is \( \deg R_i \) and since \( \deg A_i < \deg R_i \), the coefficients of \( A_i \) are uniquely determined by the equations. Since \( d_i \neq 0 \), these equations imply that \( A_i(x) \equiv 1/d_i \).

On the other hand, suppose that \( \deg D_i \geq 1 \). Then by (4.6), (4.7) and (4.8), we know that the coefficients of \( A_i \) and those of \( \tilde{A}_i \) satisfy the same system of equations. Because the \( R_i \) are relatively prime, the system is nonsingular. Since the system is of size \( \deg R_i \) and since both \( \deg A_i \) and \( \deg \tilde{A}_i \) are less than \( \deg R_i \), we conclude that \( A_i = \tilde{A}_i \).

By Theorem 4.1 the following algorithm can be used for computing \( A_i(x) \) for \( i=1, \ldots, k \).

\textbf{Algorithm 4.1}

1. Compute \( R(x) \).
2. Compute \( D_i(x) \) for \( i=1, \ldots, k \).
3. For \( i=1, \ldots, k \), if \( D_i(x) \equiv d_i \) for some constant \( d_i \) then set \( A_i(x) = 1/d_i \); else compute \( A_i(x) \) by solving (4.3).

In the following we study the number of operations needed by the algorithm.

It is well known that \( \prod_{i=1}^k R_i(x) \) can be computed by using a binary splitting scheme, which is illustrated as follows for the case \( k = 8 \):
Lemma 4.1

By using the binary splitting, $\prod_{i=1}^{k} R_{i}(x)$ and all the intermediate results such as $\prod_{i=1}^{k} R_{i}(x)$, $\prod_{i=5}^{k} R_{i}(x)$ can be computed in $O((\log k) \cdot M(n))$ operations.

Proof

Note that the sum of the degrees of all the polynomials at any level of the tree is $n$. Hence each level takes $M(n)$ operations, since $M$ satisfies Condition C. The result then follows from the fact that the height of the tree is $\lceil \log_2 k \rceil$.

Lemma 4.2

$R(x)$ can be computed in $O((\log k) \cdot M(n))$ operations.

Proof

We shall again use the binary splitting technique. We may assume that $k$ is a power of 2. It is easy to check that

$$
\sum_{j=1}^{k} (\prod_{R_{j}}) = \sum_{j=1}^{k/2} (\prod_{R_{j}}) + \sum_{j=k/2+1}^{k} (\prod_{R_{j}}) + \sum_{i=1}^{k} (\prod_{R_{i}}).
$$

This gives us a recursive algorithm for computing $R$. By Lemma 4.1, we may...
assume that all the products such as $\prod_{j>k/2} R_j$ and $\prod_{j\leq k/2} R_j$ needed by the algorithm have been precomputed. The result again follows from the fact that the sum of the degrees of all polynomials at any level of the associated binary tree is $n$. ■

Lemma 4.3

$D_1(x), \ldots, D_k(x)$ can be computed in $O((\log k) \cdot M(n))$ operations.

Proof

We may assume that $k$ is a power of 2. Note that if we use divisions to obtain $V_1$ and $V_2$ such that

$$R(x) = U_1(x) \cdot \prod_{j=1}^{k/2} R_j(x) + V_1(x),$$

$$R(x) = U_2(x) \cdot \prod_{j=k/2+1}^{k} R_j(x) + V_2(x),$$

where $\deg V_1 < \deg \prod_{j=1}^{k/2} R_j$ and $\deg V_2 < \deg \prod_{j=k/2+1}^{k} R_j$, then the problem of computing $D_i$ from $R$ for $i=1, \ldots, k$ is reduced to the problems of computing $D_i$ from $V_1$ for $i=1, \ldots, k/2$ and computing $D_i$ from $V_2$ for $i=k/2+1, \ldots, k$. This again gives us a recursive procedure. Using the fact that $D(n) = O(M(n))$ (Lemma 2.1), the lemma can be proved by the same argument as used in the proofs of Lemmas 4.1 and 4.2. ■

Lemma 4.4

$A_1(x), \ldots, A_k(x)$ can be computed in $F(n)$ operations.

Proof

Since $\deg D_i < \deg R_i$, the $A_i(x)$ and $E_i(x)$ satisfying (4.3) can be computed in $F(\deg R_i)$ operations. Hence all the $A_i$ can be computed in
By Lemmas 4.2, 4.3 and 4.4, we know that Algorithm 4.1 can be done in $F(n) + O((\log k) \cdot M(n))$ operations. After the $A_i$ have been computed, we can solve problem P1 without assuming $P(x) \equiv 1$ in $O((\log k) \cdot M(n))$ operations by the following method: For $i=1,...,k$,

1. compute $K_i(x)$ such that
   
   $P(x) = J_i(x)R_i(x) + K_i(x)$
   
   with $\deg K_i < \deg R_i$, for some $J_i$,

2. compute $L_i(x) = K_i(x)A_i(x)$ and $C_i(x)$ such that
   
   $L_i(x) = N_i(x)R_i(x) + C_i(x)$
   
   with $\deg C_i < \deg R_i$, for some $N_i$.

Note that

\[
\frac{P}{\prod R_i} = \left(\frac{P}{R_i}\right)A_i
= \left(\frac{J_i + K_i}{R_i}\right)A_i
= J_iA_i + \frac{L_i}{R_i}
= J_iA_i + N_i + \frac{C_i}{R_i}
\]

Since $P/\prod R_i$ is a proper fraction, $\sum J_iA_i + \sum N_i$ must be zero. Therefore the $C_i$ are the desired solution. Since $\deg P < n$, by the same argument as used in the proof of Lemma 4.3, $K_i(x)$ for $i=1,...,k$ can be computed in $O((\log k) \cdot M(n))$ operations. $A_i$ and $K_i$ have degree at most $\deg R_i$, so $C_i(x)$ can be computed in $O(M(\deg R_i))$ operations. This implies that $C_1(x),...,C_k(x)$ can be computed in $O(M(n))$ operations. Therefore, we have shown the following
Theorem 4.2

Problem P1 can be done in
\[ F(n) + O((\log k) \cdot M(n)) \]
operations.

We now consider the special case where the \( R_1(x) \) is given in the form
\( (x - z_i)^{m_i} \) for \( i = 1, \ldots, k \). In this case the \( A_i \) satisfying (4.3), i.e.,
\[ A_i(x)D_i(x) + E_i(x)(x - z_i)^{m_i} = 1, \]
can be computed easily in the following way. Let \( \hat{A}_i(x) = A_i(x + z_i) \), \( \hat{D}_i(x) = D_i(x + z_i) \),
etc. Then
\[ \hat{A}_i(x)\hat{D}_i(x) + \hat{E}_i(x)x^{m_i} = 1. \]
This implies that
\[ (4.10) \quad \hat{A}_i(x)\hat{D}_i(x) \equiv 1 \pmod{x^{m_i}}. \]

Hence we have the following algorithm for computing \( A_i \):

Algorithm 4.2

1. Compute \( \hat{D}_i(x) \) such that \( \hat{D}_i(x) = D_i(x + z_i) \).
2. Compute \( \hat{A}_i(x) \) from (4.10).
3. Compute \( A_i(x) \) such that \( A_i(x) = \hat{A}_i(x - z_i) \).

Step 1 is equivalent to evaluating \( D_i \) and all its derivatives at \( z_i \). Aho, Steiglitz and Ullman [1975] and Varil [1974] have independently shown that this can be done in \( O(m_i \log m_i) \) operations. Similarly, step 3 can be done in \( O(m_i \log m_i) \) operations. Step 2 involves a division, which is \( O(m_i \log m_i) \) operations by Lemma 2.1. Since \( \bigwedge_{i=1}^{k} m_i \log m_i = O(n \log n) \), by Theorem 4.2 with
\[ M(n) = O(n \log n) \] we have proved the following
Theorem 4.3

Problem $P_1$ with $R_i(x)$ given by $(x-z_i)^{m_i}$ for $i=1,\ldots,k$ can be done in $O((\log k) \cdot (n \log n))$ operations.

Suppose that we solve the general PF problem for $1/\prod_{i=1}^{k} (x-z_i) \hat{A}_i$ by solving problem $P_1$ for $1/\prod_{i=1}^{k} R_i(x)$ with $R_i(x) = (x-z_i)^{\hat{A}_i}$. Then we need not perform step 3 of Algorithm 4.2 since the solution of the general PF problem is given by the coefficients of the $\hat{A}_i$. It turns out that this is exactly Chin's $O((\log k) \cdot (n \log n))$ algorithm for solving the general PF problem for $1/\prod_{i=1}^{k} (x-z_i)^{\hat{A}_i}$. A similar observation can also be made for the case of solving the general PF problem for $P/\prod_{i=1}^{k} (x-z_i)^{\hat{A}_i}$ with $P(x) \neq 1$. 
5. AN ALGORITHM FOR PROBLEM P2

Note that using division, we have

\[
\frac{P}{Q^j} = \frac{1}{Q^{\left\lfloor \frac{j}{2} \right\rfloor}} \cdot \frac{P}{Q^{\left\lfloor \frac{j}{2} \right\rfloor}}
\]

\[
= \frac{1}{Q^{\left\lfloor \frac{j}{2} \right\rfloor}} \cdot \left( \frac{P_1}{Q^{\left\lfloor \frac{j}{2} \right\rfloor}} + \frac{P_2}{Q^{\left\lfloor \frac{j}{2} \right\rfloor}} \right)
\]

where \( \text{deg } P_1 < \left\lfloor \frac{j}{2} \right\rfloor \cdot \text{deg } Q \) and \( \text{deg } P_2 < \left\lfloor \frac{j}{2} \right\rfloor \cdot \text{deg } Q \). Thus, to solve problem P2 for the fraction \( \frac{P}{Q^j} \), it suffices to do the following:

1. Divide \( P \) by \( Q^{\left\lfloor \frac{j}{2} \right\rfloor} \) and obtain the quotient \( P_1 \) and the remainder \( P_2 \).
2. Solve problem P2 for the fractions \( \frac{P_1}{Q^{\left\lfloor \frac{j}{2} \right\rfloor}} \) and \( \frac{P_2}{Q^{\left\lfloor \frac{j}{2} \right\rfloor}} \).

This gives us a recursive procedure for solving problem P2. Assume that the expansions of the power such as \( Q^{\left\lfloor \frac{j}{2} \right\rfloor}(x) \) and \( Q^{\left\lfloor \frac{j}{2} \right\rfloor}(x) \) required by the recursive procedure have been precomputed. Let \( X(j) \) be the number of operations needed to solve problem P2. Then the recursive procedure gives

\[ X(j) \leq X(\left\lfloor \frac{j}{2} \right\rfloor) + X(\left\lfloor \frac{j}{2} \right\rfloor) + D(j \cdot \text{deg } Q) \]

for \( j > 1 \) and \( X(1) = 0 \). Note that

\[
\frac{1}{3} \leq \frac{\left\lfloor \frac{j}{2} \right\rfloor}{j} \leq \frac{2}{3}
\]

for any integer \( j \geq 2 \), and that by Lemma 2.1, \( D(j \cdot \text{deg } Q) = O(M(j \cdot \text{deg } Q)) \). We have

\[ X(j) \leq X(\alpha j) + X((1-\alpha) j) + O(M(\alpha j \cdot \text{deg } Q)) \]

where \( \alpha \) is a variable with its values in \([1/3, 2/3]\). The expansion of the recurrence corresponds to a binary tree.
such that \(X(\Delta)\) is bounded above by the total value of the nodes inside the tree. Using the fact that \(M\) satisfies Condition C, one can easily show that the sum of the values of the nodes at each level is \(O(M(\Delta \cdot \deg Q))\). Since \(\alpha \in [1/3, 2/3]\), the height of the tree is at most \(\lceil \log_{3/2} \Delta \rceil\). Hence

\[X(\Delta) = O((\log \Delta) \cdot M(\Delta \cdot \deg Q)).\]

Now we examine how to compute all the required powers of \(Q\). This can be done by using a recursion based on

\[Q^\Delta = Q^{\lceil \Delta/2 \rceil} \cdot Q^{\lfloor \Delta/2 \rfloor}.\]

The number of operations needed here clearly satisfies the same recurrence as \(X\), and hence is \(O((\log \Delta) \cdot M(\Delta \cdot \deg Q))\). We have proved the following

**Theorem 5.1**

Problem P2 can be done in

\[O((\log \Delta) \cdot M(\Delta \cdot \deg Q))\]

operations.
6. A SPECIAL CASE FOR PROBLEM P2

The following theorem can be found in Chin and Ullman [1975].

Theorem 6.1

Problem P2 can be solved in $O(\ell \log \ell)$ operations if $\deg Q = 1$.

In this section we extend the theorem to the case that $\deg Q = 2$. Our result is of interest when the underlying field $K$ is the field of real numbers, for in this case irreducible factors can have either degree one or two. We may assume that $Q$ is monic, since this will effect only $O(\ell)$ operations.

Let

$$Q(x) = x^2 + ax + b.$$  

By completing the square and letting $y = x + \frac{a}{2}$ and $c = b - \frac{a^2}{4}$, we have

$$\frac{P(x)}{Q'(x)} = \frac{P(y - \frac{a}{2})}{(y^2 + c)^{\frac{\ell}{2}}}.$$  

Write

$$P(y - \frac{a}{2}) = \sum_{i=0}^{2\ell-1} p_i y^i = (p_0 + p_2 y^2 + \cdots + p_{2\ell-2} y^{2\ell-2}) + y(p_1 + p_3 y^2 + \cdots + p_{2\ell-1} y^{2\ell-2})$$

$$= p_1(y^2) + y \cdot p_2(y^2),$$

where $\deg p_1 \leq \ell-1$ and $\deg p_2 \leq \ell-1$. Then

$$\frac{P(x)}{Q'(x)} = \frac{p_1(y^2)}{(y^2 + c)^{\frac{\ell}{2}}} + y \cdot \frac{p_2(y^2)}{(y^2 + c)^{\frac{\ell}{2}}}.$$
Hence we can solve problem P2 for \( P(x)/Q^\ell(x) \) by performing the following steps:

1. Compute \( P_0, \ldots, P_{2\ell-1} \).

2. Form \( P_1(z) = p_0 + p_2 z + \ldots + p_{2\ell-2} z^{\ell-1} \) and \( P_2(z) = p_1 + p_3 z + \ldots + p_{2\ell-1} z^{\ell-1} \).

Solve problem P2 for the fractions \( P_1(z)/(z+c)^\ell \) and \( P_2(z)/(z+c)^\ell \), and obtain

\[
\frac{P_1(z)}{(z+c)^\ell} = \sum_{i=1}^{\ell} \frac{f_i}{(z+c)^i}, \quad \frac{P_2(z)}{(z+c)^\ell} = \sum_{i=1}^{\ell} \frac{e_i}{(z+c)^i}.
\]

3. Since

\[
\frac{P(x)}{Q^\ell(x)} = \sum_{i=1}^{\ell} \frac{f_i}{Q^i(x)} + y \cdot \sum_{i=1}^{\ell} \frac{e_i}{Q^i(x)}
\]

\[
= \sum_{i=1}^{\ell} \frac{f_i}{Q^i(x)} + (x, \frac{e_i}{2}) \cdot \sum_{i=1}^{\ell} \frac{e_i}{Q^i(x)}
\]

\[
= \sum_{i=1}^{\ell} \frac{e_i x + f_i + e_i}{2} \cdot \frac{e_i}{Q^i(x)}
\]

we set \( c_i(x) = \frac{e_i x + f_i + e_i}{2} \) for \( i=1, \ldots, \ell \).

By the result of Aho, Steiglitz and Ullman [1975] and Vari [1974] step 1 can be done in \( O(\ell \log \ell) \) operations. By Theorem 6.1, step 2 can be done in \( O(\ell \log \ell) \) operations. Step 3 clearly uses \( O(\ell) \) operations. Thus, we have shown the following theorem.

**Theorem 6.2**

Problem P2 can be solved in \( O(\ell \log \ell) \) operations if \( \deg Q = 2 \).

It is an open problem whether Theorem 6.2 holds if \( \deg Q > 2 \).
REFERENCES

Aho, A. V., J. E. Hopcroft and J. D. Ullman, 1974, The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, Mass.

Aho, A. V., K. Steiglitz and J. D. Ullman, 1975, "Evaluating Polynomials at Fixed Sets of Points," SIAM J. Comput. 4, 533-539.

Borodin, A. and I. Munro, 1975, The Computational Complexity of Algebraic and Numerical Problems, American Elsevier, New York, Ch. 4.

Brent, R. P., 1975, "Multiple-Precision Zero-Finding Methods and Complexity of Elementary Function Evaluation," in Analytic Computational Complexity, ed. by J. F. Traub, Academic Press, New York, Sec. 13.

Brent, R. P. and H. T. Kung, 1976, "Fast Algorithms for Manipulating Formal Power Series," Report, Computer Science Department, Carnegie-Mellon University.

Brugia, O., 1965, "Noniterative Method for the Partial Expansion of a Rational Function with High Order Poles," SIAM Review 7, 381-387.

Chin, F. Y., 1975, "Complexity of Numerical Algorithms for Polynomials," Ph.D. thesis, Department of Electrical Engineering, Princeton University, October 1975.

Chin, F. Y. and J. D. Ullman, 1975, "Asymptotic Complexity of Partial Fraction Expansion," Report, Computer Science Laboratory, Department of Electrical Engineering, Princeton University.

Fateman, R. J., 1974, "Polynomial Multiplication, Powers and Asymptotic Analysis: Some Comments," SIAM J. Comput. 3, 196-213.

Grau, A. A., 1971, "The Simultaneous Newton Improvement of a Complete Set of Approximate Factors of a Polynomial," SIAM J. Numer. Anal. 8, 425-438.

Hazony, D. and J. Riley, 1959, "Evaluating Residues and Coefficients of High Order Poles," IRE Trans. on Automatic Control, AC-4, 132-136.

Henrici, P., 1971, "An Algorithm for the Incomplete Decomposition of a Rational Function into Partial Fraction," Z. Angew. Math. Phys. 22, 751-755.

Henrici, P., 1974, Applied and Computational Complex Analysis, Vol. 1, Wiley-Interscience, New York, Ch. 7.

Karni, S., 1969, "Easy Partial Fraction Expansion with Multiple Poles," Proc. IEEE (letters), 57, 231-232.

Knuth, D. E., 1969, The Art of Computer Programming, Vol. 2, Addison-Wesley, Reading, Mass., Sec. 4.6.4.

Kung, H. T., 1974, "On Computing Reciprocals of Power Series," Numer. Math. 22, 341-348.

Linnér, L. J. P., 1974, "The Computation of the kth Derivative of Polynomials and Rational Functions in Factored Form and Related Matters," IEEE Trans. on Circuits and Systems, CAS-21, 233-236.
**Title:** FAST ALGORITHMS FOR PARTIAL FRACTION DECOMPOSITION

**Authors:**
- H.T. Kung
- D. M. Tong

**Performing Organization:**
- Carnegie-Mellon University
- Computer Science Dept.
- Pittsburgh, PA 15213

**Contract or Grant Numbers:**
- N00014-76-C-0370
- Nr044-422

**Monitoring Agency Name & Address:**
- Office of Naval Research
- Arlington, VA 22217

**Report Date:** January 1976

**Number of Pages:** 26

**DISTRIBUTION STATEMENT:**
Approved for public release; distribution unlimited.

**KEY WORDS:**
- Partial fraction decomposition
- Proper rational function
- Algorithm

**ABSTRACT:**
The partial fraction decomposition of a proper rational function whose denominator has degree n and is given in general factored form can be done in \(O(n \log^2 n)\) operations in the worst case. Previous algorithms require \(O(n^3)\) operations, and \(O(n \log^2 n)\) operations for the special case where the factors appearing in the denominator are all linear.