Discrete Mathematics 56 (1985) 217–225
North-Holland

FAST HENSEL'S LIFTING IMPLEMENTATION USING PARTIAL FRACTION DECOMPOSITION

D. LUGIEZ

LIFIA/IMAG, Grenoble, France

Received December 1984

1. Introduction

When factoring polynomials defined over $Z[x]$, one computes the factors modulo m from the factors modulo p, where p is a suitable prime number and m a power of p, greater than m_0 a bound on the absolute value of the coefficient of any factor of the polynomial. This is done using the Hensel lemma. We present a new way to perform this lifting process based on the idea of partial fraction decomposition (denoted by p.f.d. in the following).

Section 3 contains a presentation of the classical Hensel's method, Section 4 presents Kung and Tong's algorithm for p.f.d. and describes how Viry [6] suggests to use the p.f.d. for the lifting process. Section 5 describes the new algorithm based on Viry's idea and Kung and Tong's algorithm. An analysis of complexity is given and the advantages of the new algorithm are discussed.

Computing times help us to compare the new algorithm to the two different ways of performing the Hensel's lifting, the serial one and the parallel one. These measurements show that the new algorithm is more efficient.

2. Preliminaries

Let a be a primitive squarefree univariate polynomial defined over $Z[x]$, the bound m_0 is defined by

$$m_0 = 2^n \cdot |b_n| \cdot H(a), \quad \text{if } a = \sum_{i=1}^{n} b_i \cdot x^i, \quad \text{then } H(a) = \sum_{i=1}^{n} |b_i|$$

We set $n = \deg(n)$ the degree of the polynomial, and b_n is the leading coefficient of the polynomial.

Let p be a prime number which does not divide the resultant of a and its derivative, $m = p^k > 2 \cdot m_0$, and we suppose that $a = b_n \prod_{j=1}^{r} r_j \pmod{p}$ where the r_j are the r monic factors of the polynomial modulo p. They are irreducible and pairwise prime and they may be computed by Berlekamp's algorithm for example.

3. The Hensel lemma

We present only the quadratic version of the Hensel's lemma which is faster than the linear one in the general case. The Hensel lemma computes the factors modulo q^2 from the factors modulo q and this step is repeated until the bound m is reached. We have two versions of this lifting process, the serial one and the parallel one.

Serial Hensel lemma. We lift one factor at a time and this is repeated for each factor ($r-1$ times because the two last factors are lifted together). We do not present further this well known method, see Kaltofen (2) for details.

Parallel Hensel lemma. All the factors are lifted together, then the early detection of the true factors is allowed.

Actually the bound m is often too big and it is worthwhile to test for true factors as soon as a lower heuristic bound is reached, as shown by Wang [8]. Here we have chosen $m'' = m^{1/r}$.
For example

$$a = (2x+5)(2x-31)(2x+19), \quad m = 1\ 771\ 561, \qquad m'' = 121.$$

We present now the parallel algorithm as designed by Wang except for a slight modification. This algorithm computes the new approximation $r_i + q \cdot r_i'$ modulo q^2 of the factors from the approximation modulo q which is r_i.

The parallel Hensel's lifting algorithm

```
(1) Initialization:

        $q \leftarrow p, \quad r_1 \leftarrow b_n \cdot r_1, \quad w_i = \prod_{j \neq 1} r_j.$

Compute $a_i$ s.t.

        $\sum a_i \cdot w_i = 1 \pmod p, \qquad \deg(a_i) < \deg(r_i).$

(2) While $q < m$ do:
begin

        $s = (a - \prod r_i)/q, \quad r_1 \leftarrow b_n^{-1} \cdot r_1 \pmod q.$                    (2.1)

For $i = 1, \ldots, r$ do
  begin

        (correction step)  $r_1' \leftarrow a_i \cdot s \pmod{r_i}, \quad r_1' \leftarrow b_n^{-1} \cdot r_1'$     (2.1.1)
        (new factors)      $r_i \leftarrow r_i + q \cdot r_i', \quad r_1 \leftarrow b_n \cdot r_1$             (2.1.2)
        (true factors)     if $q^2 \geqslant m''$, then test for true factors.                (2.1.3)

  end
```

```
For $i = 1, \ldots, r$ do
  begin
        (new $w_i$)   $w_i \leftarrow \prod_{j \neq i} r_j \pmod{q^2}$                    (2.1.4)
  end
For $i = 1, \ldots, r$ do
  begin
        (correction step for coefficients)                                (2.1.5)
        $t = (1 - \sum a_i \cdot w_i)/q,\ a_i' = a_i \cdot t \pmod{(r_i, q)}$
        (new coefficients)   $a_i \leftarrow a_i + q \cdot a_i'$                       (2.1.6)
  end
        $q \leftarrow q^2$                                                     (2.2)
end
```

Hence we lift two equations $a = b_n \cdot \prod r_i \pmod{q}$ and $\sum a_i \cdot w_i = 1 \pmod{q}$

The second equation is a Bezout type equality which allows us to perform the lifting process. This equation comes from the fact that the factors r_i are pairwise prime.

The coefficients such that $\sum a_i \cdot w_i = 1 \pmod{p}$ are computed using the modular Euclid's algorithm and solutions of diophantine equations.

When $q > m''$ we test for true factors, but if b_n the leading coefficient is greater than m'' the test may fail. Wang [9] suggests a way to avoid this failure. When true factors are detected, we compute the new bound related to the new polynomial to be factored and if the current modulus q is greater than this bound the lifting process is terminated and we go to the step of the reconstruction of the factors. But if it does not happen, we have to restart the algorithm at the very beginning or to compute the new coefficients a_i using an ad-hoc algorithm which terminates when $\sum a_i \cdot w_i = 1 \pmod{q}$.

According to Wang [7] this parallel Hensel's lifting appears to be faster than the serial one in most cases although a proof of this statement remains to be given.

4. Partial fraction decomposition and the Hensel's lifting

We present Kung and Tong's algorithm to solve the p.f.d. problem. We have to compute the polynomials r_i' such that $b/\prod r_i = \sum r_i'/r_i$ and $\deg(r_i') < \deg(r_i)$, where the r_i are pairwise prime polynomials, b is a polynomial such that $\deg(b) < \sum \deg(r_i)$. All the polynomials are polynomials over $K[x]$, with K a field. The algorithm is as follows.

Algorithm for partial fraction decomposition

(1) $w_i = \prod_{j \neq i} r_j$, $d_i = w_i \pmod{r_i}$,

(2) compute a_i and e_i s.t.: $a_i \cdot d_i + e_i \cdot r_i = 1$, with $\deg(a_i) < \deg(r_i)$,
(3) $k_i = b \pmod{r_i}$,
(4) $r'_i = a_i \cdot k_i \pmod{r_i}$.

The coefficients a_i and e_i may be computed by Euclid's algorithm for instance.

Viry suggests to realize the Hensel's lifting using this algorithm as follows. The polynomial to be factored is supposed to be monic until the end of this section.

We start from $a = \prod r_i \pmod{q}$ and we are looking for r'_i such that $a = \prod (r_i + q \cdot r'_i) \pmod{q^2}$. This leads to the equation

$$\frac{b}{\prod_{i=1}^{r} r_i} = \sum_{i=1}^{r} \frac{r'_i}{r_i},$$

where $b = (a - \prod r_i)/q$.

As a is monic, and as the r_i are pairwise prime the r'_i are the solutions of the p.f.d. of the fraction $b/\prod r_i$; then Viry suggests to use Kung and Tong's algorithm to solve this equation, using modular arithmetic to avoid rational arithmetic. As the r_i are monic the divisions by r_i do not cause any difficulty, but the Euclid's algorithm may raise some, as Z_q is a ring and not a field. Thus problems of divisors of zero appear. Moreover this algorithm needs a to be monic and performs the costly Euclid's algorithm at each step. The new algorithm proposed avoid these difficulties.

5. The improved algorithm

5.1. *The algorithm*

The polynomial a is not necessary monic. We start from a factorization modulo q, $a = \prod r_i$ with r_i monic for $i > 1$ and pldcf $(r_1) =$ pldcf (a) $\pmod{q^2}$, and we are looking for r'_i such that $a = \prod (r_i + q \cdot r'_i) \pmod{q^2}$. This leads to the equation $b/\prod r_i = r'_i/r_i \pmod{q}$, where $b = (a - \prod r_i)/q$

We have $\deg(r'_i) < \deg(r_i)$ because pldcf (a) = pldcf $(r_1) \pmod{q^2}$. To solve this equation we use Kung and Tong's algorithm. But we use a modular version of Kung and Tong's algorithm, because this p.f.d. (mod q^2) is strongly related to the preceeding p.f.d. (mod q). We now describe the improved algorithm.

The improved algorithm

(1) (Initialization) $q \leftarrow p$, $r_1 \leftarrow b_n \cdot r_1 \pmod{p}$. For $i > 1$ $w_i = \prod_{j \neq i} r_j \pmod{p}$, $d_i = w_i \pmod{(r_i, p)}$. Compute a_i and e_i s.t.: $a_i \cdot d_i + e_i \cdot r_i = 1 \pmod{p}$, $\deg(a_i) < \deg(r_i)$.

(2) (Lifting Process) While $q < m$ do:
begin

$$a_s = (a - \prod r_i)/q \qquad (2.1)$$

```
For $i > 1$ do
  begin
    $k_i = a_s \pmod{r_i}$                                  (2.1.1)
    $r'_i = a_i \cdot k_i \pmod{r_i}$                           (2.1.2)
    $r_i \leftarrow r_i + q \cdot r'_i \pmod{q^2}$                  (2.1.3)
  end
  $a = (\prod_{j \neq 1} r_j) r_1 + a_s$                         (2.1.4)
  (true factors) if $q > m''$, then test for true factors.
  (termination) if $q > m$, then return.                        (2.3)
For $i > 1$ do
  begin
    $w_i \leftarrow \prod_{j \neq i} r_j \pmod{q^2}$, $d_i \leftarrow w_i \pmod{(r_i, q^2)}$   (2.3.1)
    $c_i = (1 - (a_i \cdot d_i + e_i \cdot r_i))/q \pmod{q}$         (2.3.2)
    $a'_i = a_i \cdot c_i \pmod{(r_i, q)}$                       (2.3.3)
    (new coefficient)  $a_i \leftarrow a_i + q \cdot a'_i \pmod{q^2}$
                       $e_i \leftarrow (1 - a_i \cdot d_i) \pmod{(r_i, q^2)}$   (2.3.4)
  end
      $q \leftarrow q^2$                                          (2.4)
end
```

The following comments must be made concerning this new method.

The coefficients a_i are the coefficients a_i related to the p.f.d. using Kung and Tong's algorithm. At each iteration they satisfy $a_i \cdot d_i + e_i \cdot r_i = 1$, with $d_i = w_i \pmod{r_i}$, hence we can compute r'_i using these coefficients as in Kung and Tong's algorithm.

If $\deg(d_i) = 0$, then we set $a_i = d_i \pmod{q}$, $e_i = 0$. We know that d_i exists. If not, then $d_i = 0 \pmod{p}$ and then r_i divides w_i which is impossible as the r_i are pairwise prime and as w_i is set as $\prod_{j \neq i} r_j$.

After the detection of true factors, we go back to the reinitialization step if the current modulus is not big enough. The new variables are identified with capital letters, and the old one with small letters. Then we compute W_i and D_i defined as previously and we must compute the new coefficients A_i and E_i. We have $R_i = r_i$; and $W_i = w_i/a_f$ where a_f is the product of the factors yet found.

We get $D_i = q'_i \cdot r_i + d_i$ from the previous equality between W_i and w_i.

As $a_i \cdot d_i + e_i \cdot r_i = 1$ and $A_i \cdot d_i + E_i \cdot R_i = 1$ we find that

$$A_i \cdot d_i + (E_i - q'_i \cdot A_i) \cdot r_i = a_f, \qquad \text{with } \deg(A_i) < \deg(r_i)$$

Then A_i may be computed as the solution of this diophantine equation because we know the a_i and the d_i such that $a_i \cdot d_i + e_i \cdot r_i = 1$

Finally we find that the reinitialization of the new algorithm is much easier than the reinitialization of Wang's algorithm.

All the divisions in $Z_q[x]$ are done with a monic divisor and do not raise any difficulty. The Euclid's algorithm is performed only one time for each factor in $Z_p[x]$ with p a small prime and then it is not too costly.

5.2 *Complexity analysis*

We do not use techniques as the Fast Fourier Transform. Thus the product of two polynomials of respective degree p and q requires $pq \cdot (\log d)^2$ operations where d is a bound on the absolute value of the coefficients. The average computing time to divide such polynomials is also of the same order $O(pq \cdot (\log d)^2)$. We suppose that all the factors have the same average degree n/r.

At each iteration the coefficients are bounded by the current modulus q, and it can be shown that the most consuming time iteration is the last one, see Musser [5] for details. Then the complexity of the whole algorithm is the complexity of this last iteration when we have $q = m$. Under these hypothesis the most important computation is the computation of the product $\prod r_i$.

We perform: $n/r(n/r + 2n/r + \cdots + n(r-1)/r) \leq n^2$ multiplications of coefficients. Then the average computing time is $O(n^2 \log q^2)$. The computation of either w_i or d_i requires also $O(n^2 \log q^2)$ because both the division and the multiplication of polynomial have the same complexity. All the other computations involve polynomials of lower degree and do not affect this complexity analysis. Then because at the last iteration $q = m$, with $m = 2^n \cdot d$, the complexity of the new algorithm is $O(n^4 \log d^2)$. We recall that the complexity of the Hensel lemma, both serial and parallel, is $O(n^4 \log d^2)$.

Then the complexity analysis gives the same results for both algorithms.

6 Comparison and measurement

The new algorithm is a parallel one with all the advantages related to this feature. No useless computation is done and the early detection of the true factors is allowed which is a crucial point to speed up the factorization of polynomials. The improved algorithm and the original parallel Hensel lemma seem to be very similar. This is not surprising because the Hensel lemma and the p.f.d. are equivalent, but the important point is that we use Kung and Tong's algorithm. Indeed the theoretical results are the same for both of them and only measurements will show us which of them is the fastest. Moreover the new algorithm will be very efficient when we have a good decomposition modulo p, i.e., no extraneous factors, because the true factors are readily detected and the step of reconstruction of the factors is avoided. Therefore the result of the first part (Berlekamp usually) is of basic importance for the lifting process. This raises the problem of the dependance of the factorization of polynomials modulo p on the selection of p a small or medium prime number.

The computations have been done using ALDES/SAC2 of Collins, Loos on a CII-HB DPS 68 under MULTICS.

The tested polynomials are always dense and most of the coefficients have the same average number of digits. In ALDES/SAC2 the Berlekamp's algorithm is performed for ten small primes to prevent from the occurrence of extraneous factors. The Hensel lemma is the serial one denoted IUPQHL without the possibility of the detection of extraneous factors.

Measurements show that the time for the Hensel lifting lies between 10% and 30% of the total time required for the factorization. The parallel Hensel lemma implemented is the previously described one.

In what follows a is the polynomial to be factored, T is the time spent for the Berlekamp's algorithm, t_i is the time spent for the lifting process plus the reconstruction step;

$i = 1$, for the serial algorithm,

$i = 2$, for the parallel Hensel lemma,

$i = 3$, for our proposed algorithm.

We include the time spent in the reconstruction step because of the early detection of true factors and also because the time spent in this reconstruction step may be neglected in practice (less than 1% of the total time) d is the average number of digits of the coefficients of the polynomial a.

All the times are given in milliseconds.

6.1. *Example of* 2 *factors modulo* p $(r = 2)$

(i) a is irreducible over $Z[x]$: It is the worst case for the parallel algorithm as no early detection of factors is done. Our improved algorithm is comparable to the Hensel lemma.

For example: If $a = x^4 + 1$ reducible modulo p for all prime number p, $T = 2\,411$ ms, $t_1 = 881$ ms, $t_2 = 1\,080$ ms, $t_3 = 892$ ms.

(ii) a is reducible over $Z[x]$. The true factors are detected early by the parallel algorithms which stop before the bound m is reached.

The following tables, Table 1 and Table 2, reproduce measurements for low degree polynomials.

Table 1
deg $(a) = 4$

d	3	3	5
T	3937	7652	3681
t_1	1698	1703	2053
t_2	1613	1703	2552
t_3	1594	1221	1414

Table 2
deg $(a) = 8$

d	4	4	4	4
T	21 865	12 483	17 810	22 856
t_1	3690	2469	3513	3427
t_2	2997	2391	2860	2751
t_3	1946	1968	2246	2124

6.2. *Example of more than two factors modulo p* $(r > 2)$

When extraneous factors do not exist the true factors are detected and the parallel algorithms stop.

But when the degree of a increases, the probability of getting extraneous factors also increase. However parallel algorithms remain very efficient when true factors and extraneous factors are mixed as it is usually the case. When it happens we detect the true factors and after this detection is completed we often find that the new bound computed is yet reached and we only have to perform the reconstruction step. Therefore in Table 3 and in Table 4 we do not distinguish between cases where extraneous factors are or are not found. It must be noted that the parallel methods, although superior to the serial one, are even more efficient when there is no extraneous factor. Table 3 and 4 show some typical measurements with the same definitions for the t_i's. T and d as in the previous table.

Table 3
$\deg(a) = 8$

d	2	3	4	5
T	16 468	18 117	18 785	17 860
t_1	5362	3317	4911	3154
t_2	4438	1790	4579	2450
t_3	3217	1616	3341	1955

Table 4
$\deg(a) = 12$

d	5	5	5
T	53 584	47 371	52 437
t_1	10 765	17 298	16 942
t_2	7172	13 650	12 484
t_3	5306	11 804	10 092

These measurements show that the two parallel algorithms are faster than the serial one. This is strongly related to the early detection of true factors using the heuristic bound of Wang.

The theoretical understanding of both methods, i.e., parallel and serial, is still incomplete. In particular there is no proof that the parallel one is better than the serial one. The use of heuristic bound, related to the early detection of true factors, may in fact well be a decisive fact in the implementation used either here or in other computer algebra systems. Despite this lack of proof there is a large agreement that the parallel methods are more efficient.

This explains why we do not implement our new method in a serial algorithm. This is straightforward and such an algorithm will be designed to have comparison of methods using parallel, serial, heuristic and normal bound. These comparisons will be found in Lugiez, These de 3° Cycle. Also we did not implement several refinements of the new algorithm such as the computations of d_i without computing w_i when the degree of r_i is one.

However this algorithm appears to be more efficient than the usual parallel Hensel lemma, both using the early detection of true factors. The fact that our new algorithm can be easily restarted after the detection of a true factor is also an important feature of our method.

Another one is that it can be extended to multivariate polynomials as it will be shown in a forthcoming paper.

Acknowledgment

I am grateful to J. Calmet for suggesting this problem.

References

[1] E.R. Berlekamp, Algebraic Coding Theory (McGraw-Hill, New York, 1968).
[2] E. Kaltofen, Factorization of Polynomials, in: Buchberger et al., eds., Computer Algebra, Comput. Supplementum.
[3] H.T. Kung and D.M. Tong, Fast Algorithm for partial fraction decomposition, SIAM J. Comput. 6 (1977) 582–592.
[4] M. Mignotte, An inequality about factors of polynomial, Math. Comp. 28 (1975) 1153–1157.
[5] D.M. Musser, Algorithm of factorization of polynomials, Ph.D. Thesis, University of Wisconsin, Wisconsin (1971).
[6] G. Viry, Polynomial's factorization over the integers, unpublished paper.
[7] G. Viry, Factorisation des polynomes à plusieurs variables, RAIRO Inform. Théor. 14, 209–223.
[8] P.S. Wang, Parallel p-adic construction in the univariate Factoring Algorithm, in: V.E. Lewis, ed., MACSYMA User's Conference (1979) 310–317.
[9] P.S. Wang, Early Detection of true factors in the factorization of unviariate polynomias, in: J.A. Van Hulzen, ed., Proc. SAME Conf., London (1983).