Univariate Polynomial Factorization in Maple via Combinatorial Trial Division

Alejandro Erickson, Michael Monagan, Ha Le
Department of Mathematics, Simon Fraser University

NSERC CRSNG

Abstract

We present a Maple implementation ('factorrpoly') for factoring polynomials in $\mathbb{Z}[x]$. For example, on input of $3x^5-8x^4-12x^3+14x^2+25x+14$, the algorithm outputs the factorization $(3x^2-2x-7)(x^3-2x^2-3x-2)$.
The implementation uses "recden", a recursive dense polynomial representation. We first apply the Cantor Zassenhaus distinct degree factorization algorithm to factor the polynomial modulo a prime p then we lift the factors using a quadratic parallel Hensel lift with trial division of combinations of 1 and 2 modular factors at each step. The algorithm finishes with a combinatorial algorithm, checking the necessary combinations of lifted factors to find real factors.

Example

Input

$f_1 := 5517+620x+6149x^2+1458x^3+6944x^4$
$f_2 := 4583+9893x+9799x^2+528x^3+9000x^4+206x^5$
$f_3 := 2412+9747x+9733x^2+608x^3+607x^4+1804x^5+9993x^6$
$f_4 := 903950282169-4996375028x^2+9509358x^4-6756x^6+x^8$
Input expanded product: $f := f_1f_2f_3f_4$

Pseudotrace and Timing of factorrpoly(f)

Modular factorization:	modulus	number of factors	time (seconds)
Total time for this part	13	10	0.265
is 0.662 sec. Choose	19	9	0.147
mod 19 because	23	9	0.129
it has less factors.	29	11	0.121

Enter parallel Hensel lift with 9 images mod19.
Bound $= 3440393296418922708871826903271405036503040 \simeq 19^{32}$
Find sequence of lifting steps to prevent overshooting bound.
lift solution from $\mathbb{Z}_{19}[x]$ to $\mathbb{Z}_{19^2}[x]$
  from 19^2 to 19^4 time $= 0.287$, trial divide in $\mathbb{Z}[x]$, mod by 19^3
  from 19^3 to 19^6 time $= 0.219$, trial divide, mod by 19^5
  from 19^5 to 19^{10} time $= 0.318$, trial divide, mod by 19^9
  from 19^9 to 19^{18} time $= 0.254$, trial divide
found factor in $\mathbb{Z}[x]$: $4583+9893x+9799x^2+528x^3+9000x^4+206x^5$
found factor in $\mathbb{Z}[x]$: $5517+620x+6149x^2+1458x^3+6944x^4$
mod by 19^{17}
lift from 19^{17} to 19^{34} time $= 0.270$, no trial division.
Time for parallel Hensel lift: 1.424
Good factors (in $\mathbb{Z}[x]$): 2, Bad factors ($\mathbb{Z}_{19^{34}}[x]$ only): 7
Multiply bad factors mod 19^{34} and trail divide in $\mathbb{Z}[x]$ to find good factors.
Try all combinations of bad factors starting with groups of size 1.
Combined 3 found factor: $2412+9747x+9733x^2+608x^3+607x^4+1804x^5+9993x^6$
Combined remaining 4 (is a factor): $903950282169-4996375028x^2+9509358x^4-6756x^6+x^8$
Time for Combinations: 0.083
Total time for 'factorrpoly': 2.169
Time for Maple's 'factor' : 0.161

default

The Technical Bit

The Parallel Quadratic Hensel Lift

The basic Hensel lift takes $A \in \mathbb{Z}[x] \ni \gcd(A \pmod p, A' \pmod p) = 1$, for a prime, p, and $u_0, w_0 \in \mathbb{Z}_p[x]$ such that $A - u_0w_0 = 0 \pmod p$. At the kth step of the lift, $A - u^{(k)}w^{(k)} = 0 \pmod{p^k}$ where $u^{(k)} = u_0 + u_1p + u_2p^2 + \cdots + u_{k-1}p^{k-1} \pmod{p^k}$. By Hensel's lemma, the updates u_k and w_k are given by the solution to,

$$\left(\frac{A-u^{(k)}w^{(k)}}{p^k}\right)_{\in\mathbb{Z}[x]} = w_ku^{(k)} + u_kw^{(k)} \pmod p$$

In 'factorrpoly' a parallel, quadratic variation is used so that there are less steps and we avoid redundant lifting. Now,

$$u^{(k)} = u_0 + u_1p + u_2p^2 + u_3p^{2^2} + \cdots + u_{k-1}p^{2^{k-2}} \pmod{p^{2^{k-1}}}$$

Problems

1. We may potentially to lift the coefficients almost to the bound squared.
Solution: Lift via a *nearly* quadratic sequence of prime powers (see example). Consider the positive sequence given by $B_{k+1} = \lceil\sqrt{B_k}\rceil$, $B_0 = B$. Our prime powers are given by the minimum e_k such that $p^{e_k} \geq B_k$. Mod our p-adic images at the kth step by p^{e_k}. In practice, this is usually $p^{2e_{k-1}-1}$ or $p^{2e_{k-1}}$ and we lift, at most, to pB.

2. For N images the diophantine equation becomes,

$$\left(\frac{A-a_1^{(k)}a_2^{(k)}\cdots a_N^{(k)}}{p^{e_k}}\right)_{\in Z[x]} =$$

$$a_{1k}a_2^{(k)}\cdots a_N^{(k)} + a_1^{(k)}a_{2k}a_3^{(k)}\cdots a_N^{(k)} + \cdots + a_1^{(k)}\cdots a_{N-1}^{(k)}a_{Nk} \pmod{p^{e_k}}$$

which prevents us from using the euclidean algorithm because the non prime modulus introduces zero divisors.
Solution: For two factors, ignoring problem one we solve

$$1 = s^{(k)}u^{(k)} + t^{(k)}w^{(k)} \pmod{p^{2^{k-1}}} \quad (1)$$

for $k = 1$ and lift solutions, $s^{(k)}, t^{(k)}$ to $\mathbb{Z}_{p^{2^{k-1}}}[x]$ at each step. Using $s^{(k)} = s^{(k-1)} + p^{2^{k-2}}s_{k-1}$ (likewise for t, u, w) we expand (1) to arrive at

$$1 - (s^{(k-1)}u^{(k-1)} + t^{(k-1)}w^{(k-1)}) - p^{2^{k-2}}(s^{(k-1)}u_{k-1} + t^{(k-1)}w_{k-1}) =$$
$$p^{2^{k-2}}(s_{k-1}u^{(k-1)} + t_{k-1}w^{(k-1)}) \pmod{p^{2^{k-1}}}$$

Divide both sides by $p^{2^{k-2}}$ and set $d \leftarrow RHS$. Now let $s_{k-1} \leftarrow ds^{k-1}, t_{k-1} \leftarrow dt^{k-1}$ then constrain the degrees of these such that $\deg(s_{k-1}) \leq \deg(u^{(k-1)})$ to update $s^{(k)}$. We find w_{k-1} and u_{k-1} in the same fashion.

Fast Trial Division

Since we are lifting modular images they often do not correspond to factors in $\mathbb{Z}[x]$. We find the "real" factors by trial dividing combinations of images over the integers. If we do this fast enough it is worthwhile to trial divide combinations of 1 and 2 images part way up the lift. Two different trial divisions are used in 'factorrpoly' to avoid unnecessary rational reconstruction. The algorithm used in the Hensel lift is as follows:

procedure TrialDivision(A,L,p,B,'Q','M')
  # A is a polynomial in $\mathbb{Z}[x]$
  # L is a list of monic images mod p^k
  # p is a prime, B is a bound on the coefficients of the factors of A
  $N \leftarrow |L|$
  if lcoeff$(a) = 1$ **then** $m \leftarrow \Pi_{i=1}^{N}$ tcoeff$(L_i) \pmod{p^k}$
    if $m \not| $ tcoeff(a) **then return** false
  $m \leftarrow \Pi L$, $q \leftarrow$ a prime $\ni q \neq p$
  if lcoeff$(a) \neq 1$ **then** $m \leftarrow$ integer ratrecon(m)
    if $m =$FAIL **then return** false
    else $m \leftarrow$ integer primitive part(m)
  if $\|m\|_\infty > B$ **then return** false
  if lcoeff$(m) \not|$ lcoeff(a) **then return** false
  if $m \not| a \pmod q$ **then return** false
  if $m|a$ **then** $Q \leftarrow$ quotient, $M \leftarrow m$, **return** true **end**

'factorrpoly'

In brief the rest of the algorithm is as follows:
1. Input $A \in \mathbb{Z}[x]$
2. Remove integer content
3. Find square free factorizations mod $\lfloor\log(\deg A)\rfloor$ primes
4. Pick "best" prime (see example)
5. Lift monic modular images, trial dividing at each step, use rational reconstruction if necessary
6. Trial divide combinations of the remaining "bad" factors until all factors in $\mathbb{Z}[x]$ have been found
7. Return factors in $\mathbb{Z}[x]$ from each square free factor and content from step 2

Results and Timing

For a polynomial f with 2 equal size factors the algorithm is $O(d^2M(L))$. $L = O(\log_{2^{32}}\|f\|_\infty)$, M is maple's integer multiplication and $d = \deg(f)$.
'factorrpoly' is significantly faster than 'factor' in Maple 9.5 for random monic inputs with very long coefficients. For 3 factors with degree 30:
factorrpoly= 5.281 sec, factor= 12.710, sec, Coeff length = 3138 digits
factorrpoly= 14.460 sec, factor= 75.840, sec, Coeffs length = 6209 digits
'factorrpoly' is comparable to the factorization in Maple V, the last version which uses a similar algorithm.

Future Work and Recden

Recden is the underlying data structure used by 'factorrpoly' and polynomials are stored as follows:

```
POLYNOMIAL([<characterstic>,[x1,x2],[<finite field
extension>],[[<list rep in x2>],[<list rep in x2>],
<polys in x2 are coeffs of x1>])
```

This dense representation (ie zeros are stored) facilitates operations on multivariate polynomials in finite and infinite fields and rings. The package is currently under development and in the future will be implemented in C so as to replace the aging modp1 and modp2 packages.
Parts of 'factorrpoly' will be used in Ha Le's implementation of Mark van Hoeij's knapsack factorization.

References
K.O. Geddes, S.R. Czapor, G. Labahn.
Algorithms for Computer Algebra. Kluwer Academic Publishers, Boston, 1992.