A New Black Box Algorithm for Factoring Multivariate Polynomials in Maple

Tian Chen

Department of Mathematics,
Simon Fraser University, Canada

Collaborators:
Prof. Michael Monagan (SFU)
Dr. Juergen Gerhard (Maplesoft)

Multivariate Polynomial Factorization

Problem $\mathcal{P}$: Given a polynomial $a \in \mathbb{Z}[x_1, \cdots, x_n]$, compute the irreducible factors of a with coefficients in $\mathbb{Z}$.

Note that the integer content is not factored.
E.g., $6x^2 - 6y^2 = 6(x + y)(x - y)$.

- My research centers on the **design**, **analysis** and **implementation** of algorithms to solve problem $\mathcal{P}$.

A Brief History of Multivariate polynomial factorization

Sparse Hensel lifting

- Yun (1974), Wang (1975), (1978): **Multivariate Hensel lifting**.
  (can be exponential in the number of variables).
- Zippel (1981), Kaltofen (1985): Sparse Hensel lifting (SHL).
- Monagan and Tuncer (2016), (2018): MTSHL.
- Chen and Monagan (2020): CMSHL. No expression swell, no multivariate polynomial arithmetic, highly parallelizable.
  Dominating cost is evaluating the input polynomial
  –> black box representation

A Brief History of Multivariate polynomial factorization

Black box factorization

- Kaltofen and Trager (1990): First computes the black boxes of the factors, then uses sparse polynomial interpolation to recover the sparse representation of the factors.
- Rubinfeld and Zippel (1994): For factoring $a \in \mathbb{Z}[x_1, \cdots, x_n]$.
- Chen and Monagan (2022), (2023): A modular algorithm. Output factors in the sparse representation directly. Requires significantly fewer probes to the black box than Rubinfeld and Zippel's algorithm.

Key Milestone Contributions

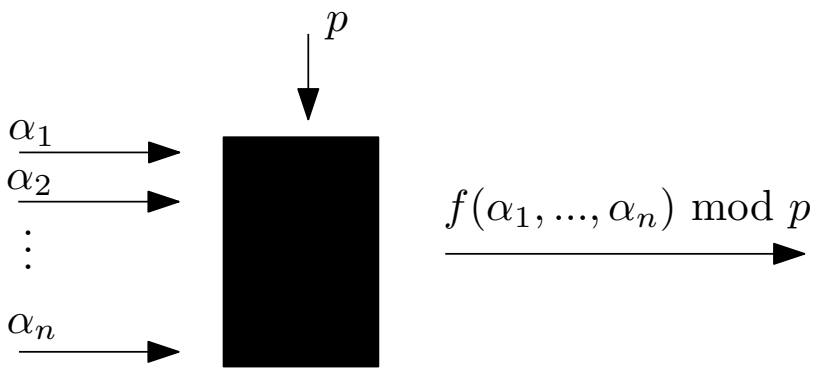
1. T. Chen and M. Monagan. The complexity and parallel implementation of two sparse multivariate Hensel lifting algorithms for polynomial factorization. In Proceedings of CASC 2020, LNCS **12291**, 150–169. Springer (2020)
2. T. Chen and M. Monagan. Factoring multivariate polynomials represented by black boxes: A Maple + C Implementation. *Math. Comput. Sci.* **16**,18 (2022)
3. T. Chen and M. Monagan. A new black box factorization algorithm - the non-monic case. In Proceedings of ISSAC 2023, pp. 173–181. ACM (2023)
4. T. Chen. Sparse Hensel lifting algorithms for multivariate polynomial factorization. PhD Thesis. Simon Fraser University (2024)

Black box representation of a polynomial

The **sparse representation** of $f \in \mathbb{Z}[x_1, \cdots, x_n]$ consists of a list of coefficients $c_k \in \mathbb{Z}$ and distinct exponent vectors $(e_{k_1}, \cdots, e_{k_n}) \in \mathbb{N}^n$ such that

$$f = \sum_{k=1}^{\#f} c_k \cdot x_1^{e_{k_1}} \cdots x_n^{e_{k_n}}.$$

A **modular black box representation** of $f \in \mathbb{Z}[x_1, \cdots, x_n]$ is a computer program B that on input $\boldsymbol{\alpha} \in \mathbb{Z}^n$ and a prime p outputs $\mathrm{B}(\boldsymbol{\alpha}, p) = f(\boldsymbol{\alpha}) \bmod p$.



Example: Black box for the determinant of a Toeplitz matrix

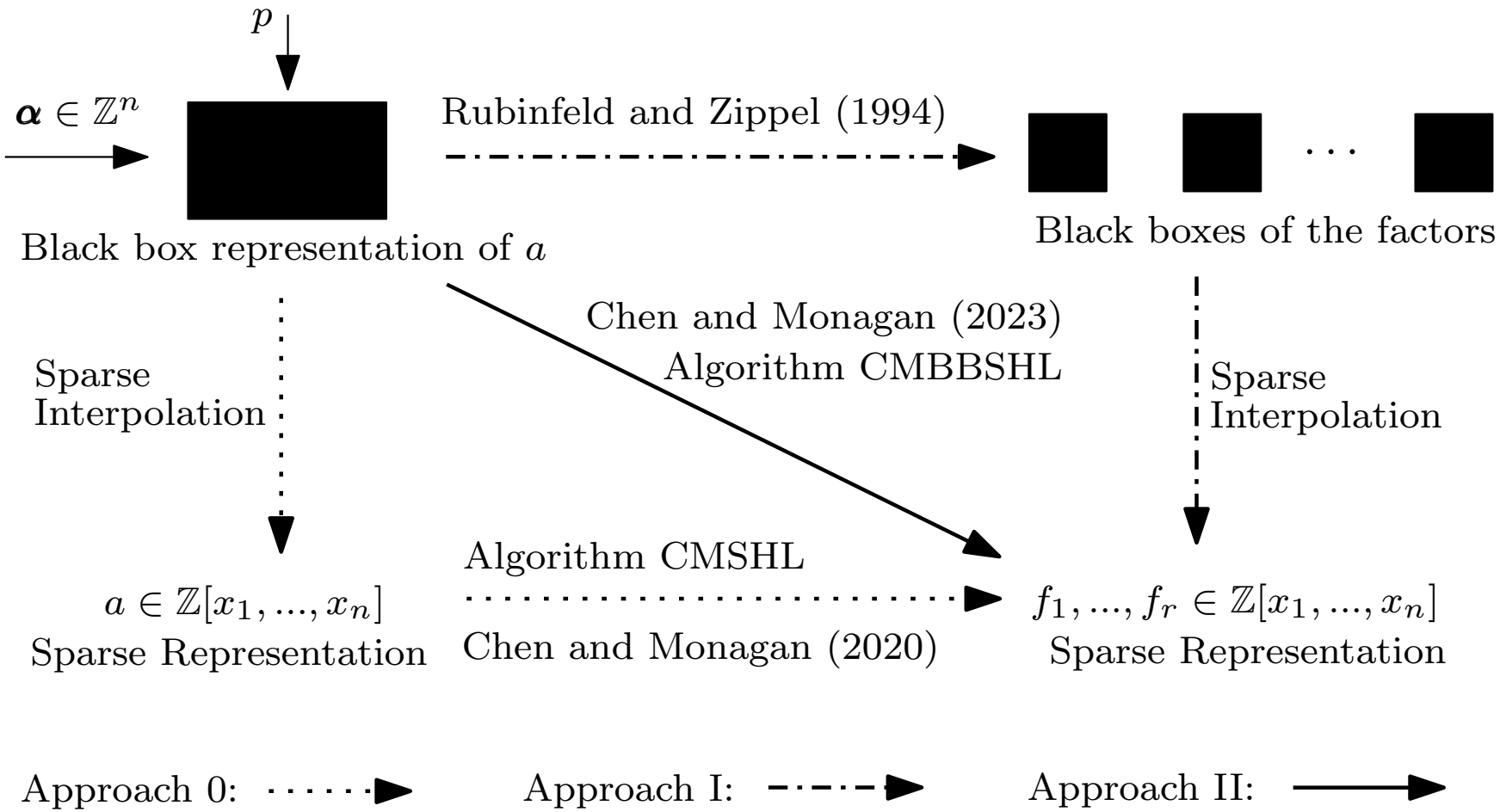
Let T_n be an $n \times n$ symmetric Toeplitz matrix. The modular black box representation of $\det(T_n)$ can be coded in Maple as a procedure:

```
B := proc( alpha::{Array,list}, p::prime )
    local n := numlems(alpha), i,j,Tn;
    Tn := Matrix(n,n);
    for i to n do
        for j to n do
            Tn[i,j] := alpha[abs(i-j)+1];
        od;
    od;
    return Det(Tn) mod p;
end:
> B ( [1,2], 7 );
```

4

Factoring a $\in \mathbb{Z}[x_1, \cdots, x_n]$ represented by a black box

Given a polynomial $a \in \mathbb{Z}[x_1, \cdots, x_n]$ represented by a black box, we aim to compute its factors in the sparse representation.



Example: Computing the determinant of a Toeplitz matrix

Let

$$T_n = \begin{pmatrix} x_1 & x_2 & x_3 & \cdots & x_n \\ x_2 & x_1 & x_2 & & \\ x_3 & x_2 & x_1 & & \\ \vdots & & & \ddots & \vdots \\ x_n & & & \cdots & x_1 \end{pmatrix}.$$

For example, $\det(T_4) =$
$(x_1^2 - x_1x_2 - x_1x_4 - x_2^2 + 2x_2x_3 + x_2x_4 - x_3^2)(x_1^2+x_1x_2+x_1x_4-x_2^2-2x_2x_3+x_2x_4-x_3^2)$.

n	$\#\det(T_n)$	$\#f_i$
8	1628	167, 167
9	6090	294, 153
10	23797	931, 931
11	90296	1730, 849
12	350726	5579, 5579
13	1338076	10611, 4983
14	5165957	34937, 34937
15	19732508	66684, 30458
16	–	221854, 221854

Table: Number of terms of $\det(T_n)$ and its factors.

Algorithm CMBBSHL (Approach II):

- Space efficient since $\#f_i \ll \#\det(T_n)$.
- Fewer probes to the black box than Rubinfeld and Zippel's algorithm.

The number of probes to the black box

	Approach I	Kaltofen & Trager	Rubinfeld & Zippel
Zippel's S.I.	# probes	$\mathcal{O}(n\delta_{\max} d^2 \# f_{\max})$	$\mathcal{O}(rn^2\delta_{\max}^2 d_1 T_{\max})$
	# univariate fac.	$\mathcal{O}(n\delta_{\max} \# f_{\max})$	$\mathcal{O}(rn^2\delta_{\max}^2 T_{\max})$
Ben-Or/ Tiwari	# probes	$\mathcal{O}(d^2 \# f_{\max})$	$\mathcal{O}(rn\delta_{\max} d_1 T_{\max})$
	# univariate fac.	$\mathcal{O}(\# f_{\max})$	$\mathcal{O}(rn\delta_{\max} T_{\max})$

Approach II	CMBBSHL
# probes	$\mathcal{O}(nd_1 d_{\max} s_{\max})$
# univariate fac.	1

Algorithm CMBBSHL requires the least number of probes since $T_{\max} \geq s_{\max}$ and $r\delta_{\max} \geq d_{\max}$.

Example (Computing the factors of $\det(T_4)$)

- Let $p = 101$ and choose $\boldsymbol{\alpha} = (3, 5, 4)$.

Example (Computing the factors of $\det(T_4)$)

- Let $p = 101$ and choose $\boldsymbol{\alpha} = (3, 5, 4)$.
- $a(x_1, \boldsymbol{\alpha}) = x_1^4 - 93x_1^2 + 420x_1 - 416 = (x_1^2 - 7x_1 + 8)(x_1^2 + 7x_1 - 52) \in \mathbb{Z}[x_1]$.

Example (Computing the factors of $\det(T_4)$)

- Let $p = 101$ and choose $\boldsymbol{\alpha} = (3, 5, 4)$.
- $a(x_1, \boldsymbol{\alpha}) = x_1^4 - 93x_1^2 + 420x_1 - 416 = (x_1^2 - 7x_1 + 8)(x_1^2 + 7x_1 - 52) \in \mathbb{Z}[x_1]$.
- The first Hensel lifting step recovers x_2 and we get

$$f_2 = x_1^2 - x_1x_2 - x_2^2 - 4x_1 + 14x_2 - 25,$$
$$g_2 = x_1^2 + x_1x_2 - x_2^2 + 4x_1 - 6x_2 - 25.$$

Example (Computing the factors of $\det(T_4)$)

- Let $p = 101$ and choose $\boldsymbol{\alpha} = (3, 5, 4)$.
- $a(x_1, \boldsymbol{\alpha}) = x_1^4 - 93x_1^2 + 420x_1 - 416 = (x_1^2 - 7x_1 + 8)(x_1^2 + 7x_1 - 52) \in \mathbb{Z}[x_1]$.
- The first Hensel lifting step recovers x_2 and we get

$$f_2 = x_1^2 - x_1x_2 - x_2^2 - 4x_1 + 14x_2 - 25,$$
$$g_2 = x_1^2 + x_1x_2 - x_2^2 + 4x_1 - 6x_2 - 25.$$

- The second Hensel lifting step recovers x_3 and

$$f_3 = x_1^2 - x_1x_2 - x_2^2 + 2x_2x_3 - x_3^2 - 4x_1 + 4x_2,$$
$$g_3 = x_1^2 + x_1x_2 - x_2^2 - 2x_2x_3 - x_3^2 + 4x_1 + 4x_2.$$

Example (Computing the factors of $\det(T_4)$)

- Let $p = 101$ and choose $\boldsymbol{\alpha} = (3, 5, 4)$.
- $a(x_1, \boldsymbol{\alpha}) = x_1^4 - 93x_1^2 + 420x_1 - 416 = (x_1^2 - 7x_1 + 8)(x_1^2 + 7x_1 - 52) \in \mathbb{Z}[x_1]$.
- The first Hensel lifting step recovers x_2 and we get

$$f_2 = x_1^2 - x_1x_2 - x_2^2 - 4x_1 + 14x_2 - 25,$$
$$g_2 = x_1^2 + x_1x_2 - x_2^2 + 4x_1 - 6x_2 - 25.$$

- The second Hensel lifting step recovers x_3 and

$$f_3 = x_1^2 - x_1x_2 - x_2^2 + 2x_2x_3 - x_3^2 - 4x_1 + 4x_2,$$
$$g_3 = x_1^2 + x_1x_2 - x_2^2 - 2x_2x_3 - x_3^2 + 4x_1 + 4x_2.$$

- At the final step, we recover x_4 and obtain the true factors

$$f = x_1^2 - x_1x_2 - x_1x_4 - x_2^2 + 2x_2x_3 + x_2x_4 - x_3^2,$$
$$g = x_1^2 + x_1x_2 + x_1x_4 - x_2^2 - 2x_2x_3 + x_2x_4 - x_3^2.$$

Benchmark: Computing the factors of $\det(T_n)$

n	10	11	12	13	14	15	16
CMBBSHL	6.299	14.679	43.927	106.838	403.089	1020.001	4876.827
# probes	109,139	267,465	894,358	2,180,399	6,981,462	17,175,949	53,416,615
Maple det	0.306	1.754	8.429	49.080	315.842	$>$ 72gb	N/A
Maple fac	1.91	3.48	23.11	57.75	509.82	7334.50	N/A
Maple tot	2.22	5.23	31.54	106.83	825.66	-	-
Magma det	1.89	5.10	36.12	327.79	2108.42	$>$ 72gb	N/A
Magma fac	1.21	7.58	158.97	583.39	13,640.79	$>$ 72gb	N/A
Magma tot	3.10	12.68	195.09	911.18	15,749.21	-	-

Table: CPU timings in seconds for computing $\det(T_n)$ using the fast Vandermonde solver. N/A: Not attempted.

Benchmark 2: Dixon matrices

Table: Timings (in seconds) for computing the determinant of Dixon matrices.

	heron3d	heron4d	robotarms (b_1)	robotarms (t_1)	heron5d
n	7	11	8	8	16
$N \times N$	13×13	63×63	16×16	16×16	399×399
$d_j = \deg(a, x_j)$ $(1 \leq j \leq n)$	19,12,12, 8,8,8,4	89,26,26, 12,12,12 8,8,8,8,8	16,64,128,44 36,16,16,16	16,64,32,56, 32,128,16,16	2159,328,328,144, 144,144,64,64, 64,64,32,32, 32,32,32,16
r	6	4	8	7	8
$\#f_i$ $(1 \leq i \leq r)$	3,23,3, 3,1,3	22,1, 6,131	1,1,2,39, 2,1,4,7	2,1,30,6, 2,7,7	823,130,22,3 3,3,3,1
e_i $(1 \leq i \leq r)$	1,2,1, 1,7,1	2,37, 7,4	8,24,48,8 24,4,4,4	48,16,8,8, 16,4,4	8,8,20,46 46,46,1831
$\#\det(A)$	525	37666243	O/M*	O/M*	N/A
$\max \lambda_\rho$	1	1	1	2	1
CMBBSHL tot	0.685	43.809	169.851	350.809	165208.747
probes tot	5701	201183	99652	131250	36008392
$\mathrm{pp}(a)$ fac	0.683	43.804	18.972	43.178	165106.278
probes $\mathrm{pp}(a)$	5699	201181	11448	16626	-
Maple det	0.614	O/M	N/A	N/A	N/A
Maple minor	0.006	0.383	O/M	O/M	N/A
Maple fac	0.084	O/M	N/A	N/A	N/A
Maple tot	0.620	-	-	-	-

N/A: Not attempted. O/M: Out of memory.

O/M*: Out of memory when expanding the factors in Maple.

Integration into Maple

A Maple and C implementation.

```
> with(BlackBoxFactor):
> BBfactor( B, X ):
```

Steps inside `BBfactor`:

```
1). Compute the partial degrees of the polynomial (w.h.p.)
2). Compute the irreducible factors (w.h.p.)
          CMBBSHLcont
              CMBBSHL
                  CMBBSHL_stepj (major subroutines coded in C)
3). Check the answer by selecting another prime.
```

Thank you!!

CMBBSHL: Hensel lifting x_j (non-monic, non-square-free)

[Algorithm 13, Chen (2024)]

Input: The modular black box B : $\mathbb{Z}^n \times \{p\} \to \mathbb{Z}_p$ s.t. B$(\boldsymbol{\beta}, p) = a(\boldsymbol{\beta}) \bmod p$, $(\hat{f}_{\rho,j-1}, 1 \le \rho \le r) \in \mathbb{Z}_p[x_1, \cdots, x_{j-1}]^r$, $\boldsymbol{\alpha} \in \mathbb{Z}^{n-1}$, a prime p, $d_i = \deg(a, x_i)$ for $1 \le i \le n$ (pre-computed), $X = [x_1, \cdots, x_n]$, $j \in \mathbb{Z}$ s.t. $\mathrm{sqf}(a_j(x_j = \alpha_j)) = \prod_{\rho=1}^r \lambda_\rho \prod_{\rho=1}^r \hat{f}_{\rho,j-1}$.

Output: $(\hat{f}_{\rho,j}, 1 \le \rho \le r) \in \mathbb{Z}_p[x_1, \cdots, x_j]^r$ s.t. (i) $\mathrm{sqf}(a_j) = \prod_{\rho=1}^r \lambda_\rho \prod_{\rho=1}^r \hat{f}_{\rho,j}$, (ii) $\hat{f}_{\rho,j}(x_j = \alpha_j) = \hat{f}_{\rho,j-1}$ for all $1 \le \rho \le r$; Otherwise, **return** FAIL.

1: Let $\hat{f}_{\rho,j-1} = \sum_{i=0}^{df_\rho} \sigma_{\rho,i}(x_2, ..., x_{j-1}) x_1^i$ $(1 \le \rho \le r)$ where $\sigma_{\rho,i} = \sum_{k=1}^{s_{\rho,i}} c_{\rho,ik} M_{\rho,ik}$.
2: Pick $\boldsymbol{\beta} = (\beta_2, \cdots, \beta_{j-1}) \in (\mathbb{Z}_p \backslash \{0\})^{j-2}$ at random.
3: Evaluate (for $1 \le \rho \le r$): $\mathcal{S}_\rho = \{\mathcal{S}_{\rho,i} = \{m_{\rho,ik} = M_{\rho,ik}(\boldsymbol{\beta}), 1 \le k \le s_{\rho,i}\}, 0 \le i \le df_\rho\}$.
4: **if** any $|\mathcal{S}_{\rho,i}| \ne s_{\rho,i}$ **then return** FAIL **end if** // monomial evals must be distinct
5: Let s be the maximum of $s_{\rho,i}$. // Compute s images of the factors in $\mathbb{Z}_p[x_1, x_j]$:
6: **for** k from 1 to s**do**
7: Let $Y_k = (x_2 = \beta_2^k, \cdots, x_{j-1} = \beta_{j-1}^k)$.
8: $A_k \leftarrow a_j(x_1, Y_k, x_j) \in \mathbb{Z}_p[x_1, x_j]$. ..... $\mathcal{O}(sd_1 d_j \mathrm{C}(\text{probe B})) + \mathcal{O}(s(d_1^2 d_j + d_1 d_j^2))$
9: **if** $\deg(A_k, x_1) \ne d_1$ **or** $\deg(A_k, x_j) \ne d_j$ **then return** FAIL **end if**
10: $g_k \leftarrow \gcd(A_k, \frac{\partial A_k}{\partial x_1}) \bmod p \in \mathbb{Z}_p[x_1, x_j]$. ..... $\mathcal{O}(s(d_1^2 d_j + d_1 d_j^2))$
11: **if** $\deg(g_k, x_1) \ne d_1 - \sum_{\rho=1}^r df_\rho$ **then return** FAIL **end if**
12: $A_{sf} \leftarrow \mathrm{quo}\,(A_k, g_k) \bmod p$. // $A_{sf} = \mathrm{sqf}(A_k) \bmod p$, up to a constant in $\mathbb{Z}_p$.
13: $A_{sfm} \leftarrow A_{sf}/(\mathrm{LC}(\mathrm{LC}(A_{sf}, x_1), x_j)) \bmod p$. // make $\mathrm{LC}(A_{sf}, x_1)$ monic in x_j.
14: $F_{\rho,k} \leftarrow \hat{f}_{\rho,j-1}(x_1, Y_k) \in \mathbb{Z}_p[x_1]$ for $1 \le \rho \le r$. ..... $\mathcal{O}(s(\sum_{\rho=1}^r \#\hat{f}_{\rho,j-1}))$
15: **if** any $\deg(F_{\rho,k}) < df_\rho$ (for $1 \le \rho \le r$) **then return** FAIL **end if**

16: **if** $\gcd(F_{\rho,k}, F_{\phi,k}) \neq 1$ for any $1 \leq \rho < \phi \leq r$ **then return** FAIL **end if**

17: $\hat{f}_{\rho,k} \leftarrow \text{BivariateHenselLift}(A_{sfm}(x_1, x_j), F_{\rho,k}(x_1), \alpha_j, p)$. . . . . $\mathcal{O}(s(\tilde{d}_1 \tilde{d}_j^2 + \tilde{d}_1^2 \tilde{d}_j))$

18: **end for**

19: Let $\hat{f}_{\rho,k} = \sum_{l=1}^{t_\rho} \alpha_{\rho,kl} \tilde{M}_{\rho,l}(x_1, x_j) \in \mathbb{Z}_p[x_1, x_j]$ for $1 \leq k \leq s$, for $1 \leq \rho \leq r$ $(t_\rho = \#\hat{f}_{\rho,k})$.

20: **for** ρ from 1 to r **do**

21: **for** l from 1 to t_ρ **do**

22: $i \leftarrow \deg(\tilde{M}_{\rho,l}, x_1)$.

23: Solve the linear system for $c_{\rho,lk}$: $\left\{ \sum_{k=1}^{s_{\rho,i}} m_{\rho,ik}^t c_{\rho,lk} = \alpha_{\rho,tl} \text{ for } 1 \leq t \leq s_{\rho,i} \right\}$.

24: **end for** . . . . . $\mathcal{O}(s\tilde{d}_j(\sum_{\rho=1}^{r} \#\hat{f}_{\rho,j-1}))$

25: Construct $\hat{f}_{\rho,j} \leftarrow \sum_{l=1}^{t_\rho} \left(\sum_{k=1}^{s_{\rho,i}} c_{\rho,lk} M_{\rho,ik}(x_2, \ldots, x_{j-1}) \right) \tilde{M}_{\rho,l}(x_1, x_j)$.

26: **end for**

27: Pick $\beta = (\beta_2, \cdots, \beta_j) \in \mathbb{Z}_p^{j-1}$ at random until $\deg(\hat{f}_{\rho,j}(x_1, \beta)) = df_\rho$ for all $1 \leq \rho \leq r$.

28: $A_\beta \leftarrow a_j(x_1, \beta) \bmod p$ via probes to **B** and Lagrange interpolation.

29: **if** $\hat{f}_{\rho,j}(x_1, \beta) \mid A_\beta$ for all $1 \leq \rho \leq r$ **then return** $(\hat{f}_{\rho,j}, 1 \leq \rho \leq r)$ **else return** FAIL **end if**

References


M. Ben-Or and P. Tiwari. A deterministic algorithm for sparse multivariate polynomial interpolation. In Proceedings of STOC '88, pp. 301–309. ACM (1988)


Chen, T., Monagan, M.: The complexity and parallel implementation of two sparse multivariate Hensel lifting algorithms for polynomial factorization. In Proceedings of CASC 2020, LNCS **12291**: 150–169. Springer (2020)


Chen, T., Monagan, M.: Factoring multivariate polynomials represented by black boxes: A Maple + C Implementation. *Math. Comput. Sci.* **16**,18 (2022)


Chen, T., Monagan, M.: A new black box factorization algorithm - the non-monic case. In Proceedings of ISSAC 2023, pp. 173–181. ACM (2023)


Von zur Gathen, J., Gerhard, J.: *Modern Computer Algebra*. Cambridge University Press (2013)


E. Kaltofen. Sparse Hensel lifting. In Proceedings of EUROCAL '85, LNCS **204**, 4–17. Springer (1985)


E. Kaltofen and B. M. Trager. Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symb. Cmpt.* **9**(3), 301–320. Elsevier (1990)


G. Lecerf. Improved dense multivariate polynomial factorization algorithms. *J. Symbolic Computation* **42**:477–494 (2007)


M. Monagan and B. Tuncer. Using Sparse Interpolation in Hensel Lifting. In Proceedings of CASC 2016, LNCS **9890**, 381–400. Springer (2016)


M. Monagan and B. Tuncer. Sparse multivariate Hensel lifting: a high-performance design and implementation. In Proceedings of ICMS 2018, LNCS **10931**, 359–368. Springer (2018)


Monagan, M., Tuncer, B.: Sparse multivariate Hensel lifting: a high-performance design and implementation. In Proceedings of ICMS 2018, LNCS **10931**, 359–368. Springer (2018)


Monagan, M., Tuncer, B.: The complexity of sparse Hensel lifting and sparse polynomial factorization. *J. Symb. Cmpt.* **99**, 189–230. Elsevier (2020)


R. Rubinfeld and R. E. Zippel. A new modular interpolation algorithm for factoring multivariate polynomials. In Proceedings of Algorithmic Number Theory, First International Symposium, ANTS-I (1994)


Schwartz, J.: Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, **27**(4), 701–717 (1980)


Wang, P.S., Rothschild, L.P.: Factoring multivariate polynomials over the integers. *Math. Comp.* **29**, 935–950 (1975)


Wang, P.S.: An improved multivariate polynomial factoring algorithm. *Math. Comp.* **32**, 1215–1231 (1978)


Yun, D.Y.Y.: The Hensel Lemma in algebraic manipulation. Ph.D. Thesis (1974)


H. Zassenhaus. On Hensel factorization I. *J. Number Theory*, **1**(3), 291–311 (1969)


Zippel, R.E.: Probabilistic algorithms for sparse polynomials. LNCS **72**, 216–226 (1979)


Zippel, R.E.: Newton's iteration and the sparse Hensel algorithm. In Proceedings of the ACM Symposium on Symbolic Algebraic Computation, pp. 68–72 (1981)


Zippel, R.E.: Interpolating polynomials from their values. *J. Symb. Cmpt.* **9**(3), 375–403 (1990)