Parallel Algorithms for Factoring Multivariate Polynomials Represented by Black Boxes

Tian Chen and Michael Monagan

Department of Mathematics,
Simon Fraser University,
British Columbia, Canada

Goal + Outline

Our goal is to develop **faster parallelizable** algorithms to factor multivariate polynomials with coefficients in the ring of integers.

1. The Sparse and the Black Box Representation of a Polynomial
2. The Algorithm CMSHL and Modification for the Black Box
3. Complexity Analysis
4. A hybrid Maple + C implementation
5. Future Work

The sparse and the black box representation of a polynomial

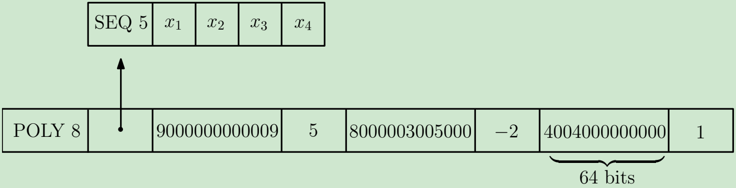
The **sparse representation** of $f \in \mathbb{Z}[x_1, \cdots, x_n]$ consists of a list of coefficients a_k and exponents $(e_{k_1}, \cdots, e_{k_n})$ such that

$$f = \sum_{k=1}^{t} a_k \cdot x_1^{e_{k_1}} \cdots x_n^{e_{k_n}}, a_k \neq 0, a_k \in \mathbb{Z}.$$

It is a natural, readable and **explicit** representation.

Example

Maple's packed monomial representation for $f = 5x_4^9 - 2x_2^3x_3^5 + x_1^4$.



The sparse and the black box representation of a polynomial

The **black box representation** of $f \in \mathbb{Z}[x_1, \cdots, x_n]$ is a program that accepts inputs a prime p and $\boldsymbol{\alpha} \in \mathbb{Z}_p^n$ and outputs $f(\boldsymbol{\alpha})$ mod p.

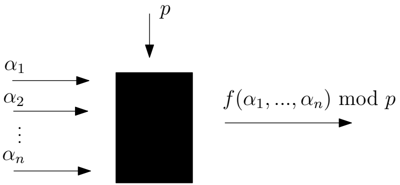


Figure: The black box representation of $f \in \mathbb{Z}[x_1, ..., x_n]$.

It is one of the most space efficient **implicit** representations (Kaltofen and Trager (1990)).

Given the black box representation of $f \in \mathbb{Z}[x_1, \cdots, x_n]$, its sparse representation can be computed in random polynomial time via *sparse interpolation*.

Background/Motivation

Sparse Hensel lifting

- Wang (1975), (1978): Mulitvariate Hensel lifting (MHL). Recovers the factors one variable at a time. Solves MDP $\sigma_i g_{j-1} + \tau_i f_{j-1} = c_i$ for $\sigma_i, \tau_i \in \mathbb{Z}_p[x_1, ..., x_{j-1}]$ one variable at a time. (exponential)
- Zippel (1981), Kaltofen (1985): Sparse Hensel lifting (SHL).
- Monagan and Tuncer (2016): Solves MDP via sparse interpolation (MTSHL).
- Monagan and Tuncer (2018): Bivariate Hensel lifts to get σ_i, τ_i.
- Chen and Monagan (2020): No multivariate polynomial arithmetic. No expression swell. Highly parallelizable. (CMSHL)
  Dominating cost is in polynomial evaluations –> Consider black box

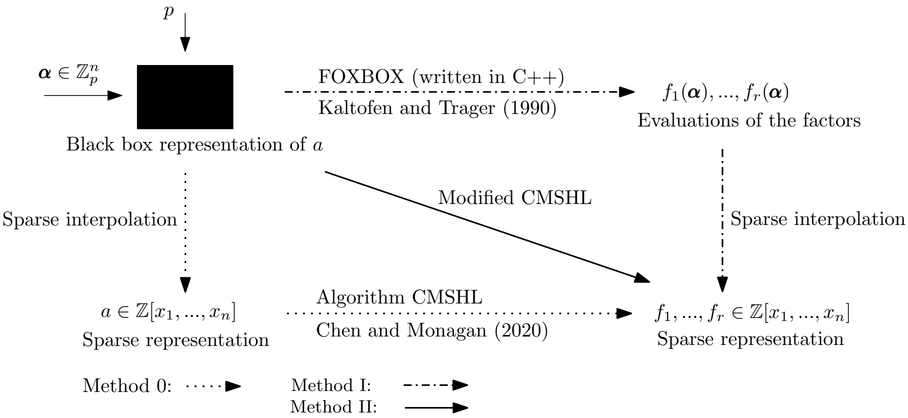
Black box factorization

- Kaltofen and Trager (1990): Black box factorization that outputs the evaluation of the factors.
- Diaz and Kaltofen (1998): FOXBOX. Implemented in C++.

There has not been any black box factorization algorithm developed since Kaltofen and Trager (1990).

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

Develop new black box factorization algorithms that output the factors in sparse representation directly via algorithm CMSHL (highly parallelizable).



Example: Computing the determinant of a Toeplitz matrix

Let T_n denote an $n \times n$ symmetric Toeplitz matrix

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

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	$\#\det(T_n)$	$\#f_i$	s
7	427	30, 56	8
8	1628	167, 167	38
9	6090	294, 153	50
10	23797	931, 931	229
11	90296	1730, 849	337
12	350726	5579, 5579	1465
13	1338076	10611, 4983	2297
14	5165957	34937, 34937	9705

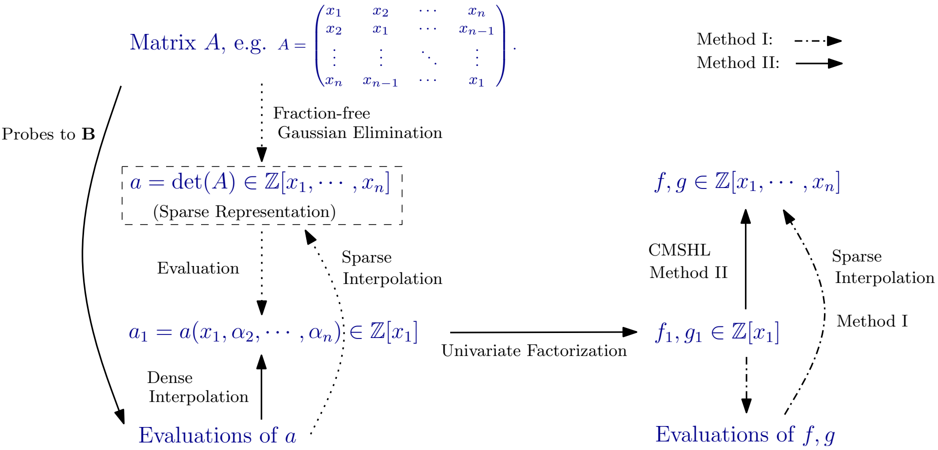
Table: Number of terms of $\det(T_n)$ and its factors.

Method II:

- Since $\#f_i \ll \#\det(T_n)$, Method II saves space by not storing $a = \det(T_n)$.
- Method II is potentially more efficient than Method I since $s \ll \#f_{\max}$ (see analysis).

Computing the factors of $a = \det(T_n) \in \mathbb{Z}[x_1, ..., x_n]$.

Method 0: Get sparse representation of a, then use SHL.
Method I: Kaltofen and Trager (1990) (modified).
Method II: Modified CMSHL (Chen and Monagan (2020)).



Computing the factors of $a = \det(T_n) \in \mathbb{Z}[x_1, ..., x_n]$.

Example (Method II: Modified CMSHL ($n = 4$))

- Choose $\boldsymbol{\alpha} = (3, 5, 4)$.

Computing the factors of $a = \det(T_n) \in \mathbb{Z}[x_1, ..., x_n]$.

Example (Method II: Modified CMSHL ($n = 4$))

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

Computing the factors of $a = \det(T_n) \in \mathbb{Z}[x_1, ..., x_n]$.

Example (Method II: Modified CMSHL ($n = 4$))

- Choose $\boldsymbol{\alpha} = (3, 5, 4)$.
- $a(x_1, \boldsymbol{\alpha}) = x_1^4 - 93x_1^2 + 420x_1 - 416 = (x_1^2 - 7x_1 + 8)(x_1^2 + 7x_1 - 52) \in \mathbb{Z}[x_1]$.
- Choose $p = 101$. After the first step of Hensel lifting, we get

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

Computing the factors of $a = \det(T_n) \in \mathbb{Z}[x_1, ..., x_n]$.

Example (Method II: Modified CMSHL ($n = 4$))

- Choose $\boldsymbol{\alpha} = (3, 5, 4)$.
- $a(x_1, \boldsymbol{\alpha}) = x_1^4 - 93x_1^2 + 420x_1 - 416 = (x_1^2 - 7x_1 + 8)(x_1^2 + 7x_1 - 52) \in \mathbb{Z}[x_1]$.
- Choose $p = 101$. After the first step of Hensel lifting, we get

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

- After the second step, we get

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

Example (Method II: Modified CMSHL ($n = 4$))

- Choose $\boldsymbol{\alpha} = (3, 5, 4)$.
- $a(x_1, \boldsymbol{\alpha}) = x_1^4 - 93x_1^2 + 420x_1 - 416 = (x_1^2 - 7x_1 + 8)(x_1^2 + 7x_1 - 52) \in \mathbb{Z}[x_1]$.
- Choose $p = 101$. After the first step of Hensel lifting, we get

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

- After the second step, we get

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

- At the final step, we obtain the true factors

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

Algorithm CMSHL: distributed: two factors (CASC 2020)

Input: p, $\alpha_j \in \mathbb{Z}_p$, $a_j \in \mathbb{Z}_p[x_1, ..., x_j]$, $f_{j-1}, g_{j-1} \in \mathbb{Z}_p[x_1, ..., x_{j-1}]$.
Output: f_j, g_j s.t. $a_j = f_j g_j$ and $f_j(x_j = \alpha_j) = f_{j-1}$, $g_j(x_j = \alpha_j) = g_{j-1}$ or FAIL.

```
Let f_{j-1} = x_1^{df} + Σ_{i=0}^{df-1} σ_i(x_2, ..., x_{j-1}) x_1^i with σ_i = Σ_{k=1}^{s_i} c_{ik} M_{ik}
    g_{j-1} = x_1^{dg} + Σ_{i=0}^{dg-1} τ_i(x_2, ..., x_{j-1}) x_1^i with τ_i = Σ_{k=1}^{t_i} d_{ik} N_{ik},
Pick β = (β_2, ..., β_{j-1}) ∈ Z_p^{j-2} at random and evaluate monomials at β:
    {S_i = {m_{ik} = M_{ik}(β), 1 ≤ k ≤ s_i}, 0 ≤ i < df}; {T_i = {n_{ik} = N_{ik}(β), 1 ≤ k ≤ t_i}, 0 ≤ i < dg}.
Check if monomial evaluations are distinct. If not, return FAIL.
Let s be the maximum of s_i and t_i.
for k from 1 to s in parallel do
    Let Y_k = (x_2 = β_2^k, ..., x_{j-1} = β_{j-1}^k).
    A_k, F_k, G_k ← a_j(x_1, Y_k, x_j), f_{j-1}(x_1, Y_k), g_{j-1}(x_1, Y_k). ....... Eval: O(s(#f + #g + #a))
    if gcd(F_k, G_k) ≠ 1 then return FAIL end if // unlucky evaluation
    f_k, g_k ← BivariateHenselLift(A_k, F_k, G_k, α_j, p). .................. BHL: O(s(d_1^2 d_j + d_1 d_j^2))
end for
Let f_k = x_1^{df} + Σ_{l=1}^{μ} α_{kl} M̃_l(x_1, x_j) for 1 ≤ k ≤ s, where μ ≤ d_1 d_j.
for l from 1 to μ in parallel do
    i ← deg(M̃_l, x_1).
    Solve the s_i × s_i linear system for c_{lk}: { Σ_{k=1}^{s_i} m_{ik}^n c_{lk} = α_{nl} for 1 ≤ n ≤ s_i }
end for
Construct f_j ← x_1^{df} + Σ_{l=1}^{μ} ( Σ_{k=1}^{s_i} c_{lk} M_{ik}(x_2, ..., x_{j-1}) ) M̃_l(x_1, x_j).
Similarly, construct g_j. .................................... VSolve: O(s d_j(#f + #g))
if a_j = f_j g_j then return (f_j, g_j) else return FAIL end if
```

Algorithm CMSHL: distributed: two factors (CASC 2020)

Theorem (Chen and Monagan (2020))

Let p be a large prime, $d = \deg(a)$, $d_1 = \deg(a, x_1)$, $d_j = \deg(a, x_j)$ and s_j be the number s defined in line 4 of Algorithm CMSHL and $T_{fg_{j-1}} = \max(\#f_{j-1}, \#g_{j-1})$. With a failure probability less than

$$\frac{d\, s_j(T_{fg_{j-1}} + d\, s_j) + 2d^2 T_{fg_{j-1}}}{p - d + 1},$$

the number of arithmetic operations in $\mathbb{Z}_p$ for the j^{th} Hensel lifting step via Algorithm CMSHL in the worst case is

$$\mathcal{O}(d_j s_j (\underbrace{\#f_{j-1} + \#g_{j-1}}_{\text{VSolve}} + \underbrace{d_1^2 + d_1 d_j}_{\text{BHL}}) + \underbrace{s_j \#a_j}_{\text{Eval}}).$$

Usually $\#f_j \ll \#a_j$, $\#g_j \ll \#a_j$ and Eval dominates the cost of CMSHL.

Algorithm CMSHL: black box: multi-factors

Let $f_{\rho,j-1} = x_1^{df_\rho} + \sum_{i=0}^{df_\rho - 1} \sigma_{\rho,i}(x_2, \ldots, x_{j-1}) x_1^i$ for $1 \le \rho \le r$, where $\sigma_{\rho,i} = \sum_{k=1}^{s_{\rho,i}} c_{\rho,ik} M_{\rho,ik}$ and $M_{\rho,ik}$ are the monomials in $\sigma_{\rho,i}$.
Pick $\boldsymbol{\beta} = (\beta_2, \cdots, \beta_{j-1}) \in \mathbb{Z}_p^{j-2}$ at random.
$\{\mathcal{S}_\rho = \{S_{\rho,i} = \{m_{\rho,ik} = M_{\rho,ik}(\boldsymbol{\beta}), 1 \le k \le s_{\rho,i}\}, 0 \le i \le df_\rho - 1\}, 1 \le \rho \le r\}$.
if any $|S_{\rho,i}| \ne s_{\rho,i}$ **then return** FAIL **end if**
Let s be the maximum of $s_{\rho,i}$.
for k from 1 to s **do**
    Let $Y_k = (x_2 = \beta_2^k, \cdots, x_{j-1} = \beta_{j-1}^k)$.
    $A_k \leftarrow a_j(x_1, Y_k, x_j)$. // via probes to **B** and dense interpolation ... $\mathcal{O}(sd_1d_j \cdot C(\text{probe B}))$
    $F_{1,k}, \cdots, F_{r,k} \leftarrow f_{1,j-1}(x_1, Y_k), \cdots, f_{r,j-1}(x_1, Y_k)$. .... $\mathcal{O}(s(\#f_1 + \cdots + \#f_r))$
    if $\gcd(F_{\rho,k}, F_{\phi,k}) \ne 1$ for any $\rho \ne \phi$ $(1 \le \rho, \phi \le r)$ **then return** FAIL **end if**
    $f_{1,k}, \cdots f_{r,k} \leftarrow$ *BivariateHenselLift*$(A_k, F_{1,k}, \cdots, F_{r,k}, \alpha_j, p)$. .... $\mathcal{O}(d_1 d_j^2 + d_1^2 d_j)$
end for
Let $f_{\rho,k} = x_1^{df_\rho} + \sum_{l=1}^{\mu_\rho} \alpha_{\rho,kl} \tilde{M}_{\rho,l}(x_1, x_j)$ for $1 \le k \le s$ where $\mu_\rho \le d_1 d_j$, for $1 \le \rho \le r$.
for ρ from 1 to r **do**
    for l from 1 to μ_ρ **do**
        $i \leftarrow \deg(\tilde{M}_{\rho,l}, x_1)$. Solve the $s_{\rho,i} \times s_{\rho,i}$ linear system for $c_{\rho,lk}$:
        $\left\{ \sum_{k=1}^{s_{\rho,i}} m_{\rho,ik}^n c_{\rho,lk} = \alpha_{\rho,nl} \text{ for } 1 \le n \le s_{\rho,i} \right\}$.
    end for .... $\mathcal{O}(sd_j(\#f_1 + \cdots + \#f_r))$
    Construct $f_{\rho,j} \leftarrow x_1^{df_\rho} + \sum_{l=1}^{\mu_\rho} \left(\sum_{k=1}^{s_{\rho,i}} c_{\rho,lk} M_{\rho,ik}(x_2, \ldots, x_{j-1}) \right) \tilde{M}_{\rho,l}(x_1, x_j)$.
end for
if $a_j = \prod_{\rho=1}^r f_{\rho,j}$ **then return** $f_{1,j} \cdots, f_{r,j}$ **else return** FAIL **end if**

Complexity Analysis

- Interested in the number of probes to the black box **B**.
- Preliminary estimates show that Method II requires a lot fewer probes to the black box than Method I since $s \ll \#f_{\text{max}}$ (Chen and Monagan (2020)). (s is the number of bivariate images needed in algorithm CMSHL and $\#f_{\text{max}}$ is the maximum number of terms among the factors of a.)

n	6	7	8	9	10	11	12	13	14
$\#f_{\text{max}}$	32	56	167	294	931	1730	5579	10611	34937
s	6	8	38	50	229	337	1465	2297	9705

Table: $\#f_{\text{max}}$ and s for $\det(T_n)$.

Complexity Analysis

- Interested in the number of probes to the black box **B**.
- Preliminary estimates show that Method II requires a lot fewer probes to the black box than Method I since $s \ll \#f_{\max}$ (Chen and Monagan (2020)). (s is the number of bivariate images needed in algorithm CMSHL and $\#f_{\max}$ is the maximum number of terms among the factors of a.)

n	6	7	8	9	10	11	12	13	14
$\#f_{\max}$	32	56	167	294	931	1730	5579	10611	34937
s	6	8	38	50	229	337	1465	2297	9705

Table: $\#f_{\max}$ and s for $\det(T_n)$.

- Method I:
  $O(nd_1 d_{\max}\#f_{\max})$ probes **B** using sparse interpolation (Zippel (1990)), plus $O(nd_{\max}\#f_{\max})$ times the cost of univariate polynomial factorization.
- Method II:
  $O(nd_1 d_{\max} s)$ probes to the black box **B**.
  Total cost:
  $O\left((n-2)\left(sd_{\max}\left(\sum_{i=1}^{r}\#f_i + d_1^2 + d_1 d_{\max}\right) + sd_1 d_{\max}\mathrm{C}(\text{probe B})\right)\right).$

A Hybrid Maple + C Implementation of Method II

n	10	11	12	13	14
total time	9.817	23.268	111.592	321.486	2836.336
total probes	109139	267465	894358	2180399	6981462
Maple Det	0.566	5.095	359.249	3127.827	N/A
Maple factor	1.687	3.836	50.143	364.842	N/A
Maple total	2.253	8.931	409.392	3492.669	–
Hensel Lift x_n total	2.069	3.446	26.936	61.186	867.773
s (Hensel lift x_n)	522	814	3174	5223	19960
BB (C)	0.475	0.843	4.403	8.381	43.931
BB+Interp	1.103	1.671	8.542	16.214	83.621
Eval f, g (Maple)	0.109	0.327	7.229	19.165	416.548
BHL (Maple)	0.4331	0.750	3.107	5.522	27.707
table (Maple)	0.163	0.459	4.556	13.772	224.573
VSolve (C)	0.239	0.170	3.209	5.973	112.495

Table: Timings for computing $\det(T_n)$ using CMSHL black box algorithm.

Future Work

- Next Goal: Compute $\det(T_n)$ for $n = 15$
- Possible speed ups for Method II:
  - Eval f, g in C
  - BB + Interp
  - Integrate BHL in C
- Implement Method I in Maple
- A detailed complexity analysis with failure probabilities
- Non-monic factors

Future Work

- Next Goal: Compute $\det(T_n)$ for $n = 15$
- Possible speed ups for Method II:
  - Eval f, g in C
  - BB + Interp
  - Integrate BHL in C
- Implement Method I in Maple
- A detailed complexity analysis with failure probabilities
- Non-monic factors

Thank you for attending!

References

- 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)
- Diaz, A., Kaltofen E.: FOXBOX: A system for manipulating symbolic objects in black box representation. In Proceedings of ISSAC '98, pp. 30–37. ACM (1998)
- Kaltofen, E.: Sparse Hensel lifting. In Proceedings of EUROCAL '85, Springer LNCS **204**, 4–17 (1985)
- Kaltofen E., Trager, B.M.: 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)
- Monagan, M., Tuncer, B.: Using Sparse Interpolation in Hensel Lifting. In Proceedings of CASC 2016, LNCS **9890**, 381–400. Springer (2016)
- 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)

References

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

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)