

Sparse multivariate polynomial factorization: A high-performance design and implementation.

Michael Monagan

Centre for Experimental and Constructive Mathematics
Simon Fraser University
British Columbia

This is joint work with Baris Tuncer.

Factoring polynomials using Wang's Hensel lifting

```
> e1 := (a = f*g);
```

$$\begin{aligned} e1 &:= x^5 + 20x^2y^6z + 5x^4y^3z + 30xy^4z^3 + 12xy^4z^2 + 4x^3y^3 \\ &\quad + 3x^3yz^2 + 18y^2z^4 + 6x^2yz^2 \\ &= (x^2 + 5xy^3z + 3yz^2)(x^3 + 4xy^3 + 6yz^2) \end{aligned}$$

```
> e2 := eval(e1,z=3);
```

$$\begin{aligned} e2 &:= x^5 + 60x^2y^6 + 15x^4y^3 + 4x^3y^3 + 918xy^4 + 27x^3y + 54x^2y + 1458y^2 \\ &= (x^2 + 15xy^3 + 27y)(x^3 + 4xy^3 + 54y) \end{aligned}$$

```
> e3 := eval(e2,y=-5);
```

$$\begin{aligned} e3 &:= x^5 - 1875x^4 - 635x^3 + 937230x^2 + 573750x + 36450 \\ &= (x^2 - 1875x - 135)(x^3 - 500x - 270) \end{aligned}$$

Notes: Let h be any factor of a and let $B > \max(\|h\|_\infty, \|a\|_\infty)$.
Multivariate Hensel Lifting (MHL) is done modulo $m = p^L > 2B$.

Wang's Multivariate Hensel Lifting (MHL)

Input $a \in \mathbb{Z}_p[x_1, \dots, x_j]$, $\alpha = (\alpha_2, \dots, \alpha_j)$, $(f_0, g_0) \in \mathbb{Z}_p[x_1, \dots, x_{j-1}]$ s.t.

- (i) $a(x_1, \dots, x_{j-1}, \alpha_j) = f_0 g_0$ and
- (ii) $\gcd(f_0(x_1, \alpha), g_0(x_1, \alpha)) = 1$ in $\mathbb{Z}_p[x_1]$.

Idea: $f = f_0 + f_1(x_j - \alpha_j) + f_2(x_j - \alpha_j)^2 + \dots + f_{df}(x_j - \alpha_j)^{df}$

Wang's Multivariate Hensel Lifting (MHL)

Input $a \in \mathbb{Z}_p[x_1, \dots, x_j]$, $\alpha = (\alpha_2, \dots, \alpha_j)$, $(f_0, g_0) \in \mathbb{Z}_p[x_1, \dots, x_{j-1}]$ s.t.

- (i) $a(x_1, \dots, x_{j-1}, \alpha_j) = f_0 g_0$ and
- (ii) $\gcd(f_0(x_1, \alpha), g_0(x_1, \alpha)) = 1$ in $\mathbb{Z}_p[x_1]$.

Idea: $f = f_0 + f_1(x_j - \alpha_j) + f_2(x_j - \alpha_j)^2 + \dots + f_{df}(x_j - \alpha_j)^{df}$

Initialize: $(f, g) := (f_0, g_0)$ and $error := a - fg$

For $k = 1, 2, \dots$, while $\deg(f, x_j) + \deg(g, x_j) < \deg(a, x_j)$ do

$c_k := \text{coeff}(error, (x_j - \alpha_j)^k)$

If $c_k \neq 0$ then

Solve the MDP $f_k g_0 + g_k f_0 = c_k$ for $f_k, g_k \in \mathbb{Z}_p[x_1, \dots, x_{j-1}]$.

Set $f := f + f_k(x_j - \alpha_j)^k$ and $g := g + g_k(x_j - \alpha_j)^k$

Set $error := a - fg$

If $error = 0$ output (f, g) else output FAIL.

Implemented in Magma, Maple, Macsyma, Mathematica and Singular.

Algorithms for Computer Algebra, Geddes, Czapor, and Labahn, 1992.

Wang's Multivariate Hensel Lifting (MHL)

Input $a \in \mathbb{Z}_p[x_1, \dots, x_j]$, $\alpha = (\alpha_2, \dots, \alpha_j)$, $(f_0, g_0) \in \mathbb{Z}_p[x_1, \dots, x_{j-1}]$ s.t.

- (i) $a(x_1, \dots, x_{j-1}, \alpha_j) = f_0 g_0$ and
- (ii) $\gcd(f_0(x_1, \alpha), g_0(x_1, \alpha)) = 1$ in $\mathbb{Z}_p[x_1]$.

Idea: $f = f_0 + f_1(x_j - \alpha_j) + f_2(x_j - \alpha_j)^2 + \dots + f_{df}(x_j - \alpha_j)^{df}$

Initialize: $(f, g) := (f_0, g_0)$ and $error := a - fg$

For $k = 1, 2, \dots$, while $\deg(f, x_j) + \deg(g, x_j) < \deg(a, x_j)$ do

$c_k := \text{coeff}(error, (x_j - \alpha_j)^k)$

If $c_k \neq 0$ then

Solve the MDP $f_k g_0 + g_k f_0 = c_k$ for $f_k, g_k \in \mathbb{Z}_p[x_1, \dots, x_{j-1}]$.

Set $f := f + f_k(x_j - \alpha_j)^k$ and $g := g + g_k(x_j - \alpha_j)^k$

Set $error := a - fg$

If $error = 0$ output (f, g) else output FAIL.

Implemented in Magma, Maple, Macsyma, Mathematica and Singular.

Algorithms for Computer Algebra, Geddes, Czapor, and Labahn, 1992.

Wang's algorithm is highly sequential. $O(d^n)$ if the α_j 's are non-zero.

The Taylor Coefficients : MT CASC 2016

$$f = x^3 - xyz^2 + y^3z^2 + z^4 - 2$$

The solutions f_k to the MDP are Taylor coefficients of $f(z)$ about $z = \alpha_3$.

If $\alpha_3 = 0$ then $f(z) = \underbrace{(x^3 - 2)}_{f_0} + \underbrace{(y^3 - xy)}_{f_2} z^2 + \underbrace{1}_{f_4} z^4$.

If $\alpha_3 = 2$ then

$$f(z) = \underbrace{(x^3 + 4y^3 - 4xy + 14)}_{f_0} + \underbrace{(4y^3 - 4xy + 32)}_{f_1}(z - 2) + \underbrace{(y^3 - xy + 24)}_{f_2}(z - 2)^2 + \underbrace{8}_{f_3}(z - 2)^3 + \underbrace{1}_{f_4}(z - 2)^4$$

The Taylor Coefficients : MT CASC 2016

$$f = x^3 - xyz^2 + y^3z^2 + z^4 - 2$$

The solutions f_k to the MDP are Taylor coefficients of $f(z)$ about $z = \alpha_3$.

$$\text{If } \alpha_3 = 0 \text{ then } f(z) = \underbrace{(x^3 - 2)}_{f_0} + \underbrace{(y^3 - xy)}_{f_2} z^2 + \underbrace{1}_{f_4} z^4.$$

If $\alpha_3 = 2$ then

$$f(z) = \underbrace{(x^3 + 4y^3 - 4xy + 14)}_{f_0} + \underbrace{(4y^3 - 4xy + 32)}_{f_1}(z - 2) + \underbrace{(y^3 - xy + 24)}_{f_2}(z - 2)^2 + \underbrace{8}_{f_3}(z - 2)^3 + \underbrace{1}_{f_4}(z - 2)^4$$

Lemma 1 [MT 2016] If α_j is chosen at random from a sufficiently large set then

$$\text{supp}(f_0) \supseteq \text{supp}(f_1) \supseteq \cdots \supseteq \text{supp}(f_k) \quad \text{w.h.p.}$$

MTSHL : Monagan and Tuncer's Sparse Hensel Lifting

Solve $f_k g_0 + g_k f_0 = c_k$ for $f_k, g_k \in \mathbb{Z}_p[x_1, \dots, x_{n-1}]$.

Let $f_{k-1} = \sum_{i=1}^{df} a_i(x_2, \dots, x_{n-1})x_1^i$ and $\text{supp}(a_i) = \{M_{i1}, \dots, M_{ir_i}\}$,
and $g_{k-1} = \sum_{i=1}^{dg} b_i(x_2, \dots, x_{n-1})x_1^i$ and $\text{supp}(b_i) = \{N_{i1}, \dots, N_{is_i}\}$.

Assume $f_k = \sum_{i=0}^{df} \left(\sum_{j=1}^{r_i} a_{ij} M_{ij} \right) x_1^i$ and $g_k = \sum_{i=0}^{dg} \left(\sum_{j=1}^{s_i} b_{ij} N_{ij} \right) x_1^i$.

Let $s = \max(\#a_i, \#b_i)$. Let $\beta = (\beta_2, \dots, \beta_{n-1}) \in \mathbb{Z}_p^{n-2}$.

For $1 \leq j \leq s$ solve $\sigma_j g_0(\beta^j) + \tau_j f_0(\beta^j) = c_k(\beta^j)$ for $\sigma_j, \tau_j \in \mathbb{Z}_p[x_1]$.

Needs $\gcd(g_0(\beta^j), f_0(\beta^j)) = 1 \forall 1 \leq j \leq s$.

Solve the shifted Vandermonde systems

$$\left\{ \begin{array}{l} \text{coeff}(f_k(\beta^j), x_n^i) = \text{coeff}(\sigma_j, x_n^i) \text{ for } 1 \leq j \leq \#a_i \\ \text{coeff}(g_k(\beta^j), x_n^i) = \text{coeff}(\tau_j, x_n^i) \text{ for } 1 \leq j \leq \#b_i \end{array} \right\} \text{ for } 0 \leq i < df,$$
$$\left\{ \begin{array}{l} \text{coeff}(f_k(\beta^j), x_n^i) = \text{coeff}(\sigma_j, x_n^i) \text{ for } 1 \leq j \leq \#a_i \\ \text{coeff}(g_k(\beta^j), x_n^i) = \text{coeff}(\tau_j, x_n^i) \text{ for } 1 \leq j \leq \#b_i \end{array} \right\} \text{ for } 0 \leq i < dg.$$

Needs $M_{ij}(\beta) \neq M_{ik}(\beta) \forall i, j \neq k$ and $N_{ij}(\beta) \neq N_{ik}(\beta) \forall i, j \neq k$.

$a_j(x_1, x_2, \dots, x_j)$
 $f_{j-1}(x_1, x_2, \dots, x_{j-1})$
 evaluate x_3, \dots, x_{j-1} for $1 \leq k \leq s$

\downarrow
 $a_j(x_1, x_2, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

$f_{j-1}(x_1, x_2, \beta_3^k, \dots, \beta_{j-1}^k)$
 evaluate x_2 for $1 \leq l \leq \deg(a_j, x_2)$

\downarrow
 $a_j(x_1, \gamma_m, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

$f_{j-1}(x_1, \gamma_m, \beta_3^k, \dots, \beta_{j-1}^k)$

Hensel
 lift x_j

\longrightarrow

$f_j(x_1, x_2, x_3, \dots, x_j)$
 $g_j(x_1, x_2, x_3, \dots, x_j)$

\uparrow

interpolate x_3, \dots, x_{j-1}
 $f_j(x_1, x_2, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

$g_j(x_1, x_2, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

\uparrow

dense interpolate x_2

$f_j(x_1, \gamma_m, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

Hensel
 lift x_j

\longrightarrow

$g_j(x_1, \gamma_m, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

$a_j(x_1, x_2, \dots, x_j)$
 $f_{j-1}(x_1, x_2, \dots, x_{j-1})$
 evaluate x_3, \dots, x_{j-1} for $1 \leq k \leq s$

\downarrow
 $a_j(x_1, x_2, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

$f_{j-1}(x_1, x_2, \beta_3^k, \dots, \beta_{j-1}^k)$
 evaluate x_2 for $1 \leq l \leq \deg(a_j, x_2)$

\downarrow
 $a_j(x_1, \gamma_m, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

$f_{j-1}(x_1, \gamma_m, \beta_3^k, \dots, \beta_{j-1}^k)$

Hensel
 lift x_j

\longrightarrow

$f_j(x_1, x_2, x_3, \dots, x_j)$
 $g_j(x_1, x_2, x_3, \dots, x_j)$

\uparrow

interpolate x_3, \dots, x_{j-1}
 $f_j(x_1, x_2, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

$g_j(x_1, x_2, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

\uparrow

dense interpolate x_2

$f_j(x_1, \gamma_m, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

Hensel
 lift x_j

\longrightarrow

$g_j(x_1, \gamma_m, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$

Parallelized evaluations $a(x_1, x_2, \beta_3^k, \dots, \beta_{j-1}^k, x_j)$ using Cilk C
 Do evaluations $x_2 = \gamma_m$ and **bivariate Hensel lifts** in parallel.
 Optimize the bivariate Hensel lifts in $\mathbb{Z}_p[x_1, x_j] \bmod (x_j - \alpha_j)^k$.

Benchmark 1 – many variables, modest degree

On a server with 2 Intel Xeon E2660 8 core CPUs – 2.2 GHz(base) – 3.0 GHz(turbo)

n	d	t	Maple 2018		New times (1 core)			New times (16 cores)		
			best	worst	total	(hensel)	(eval)	total	(hensel)	(eval)
6	7	500	0.411	28.84	0.098	(0.015)	(0.042)	0.074	(0.019)	(0.008 – 5.2x)
6	7	1000	1.140	58.46	0.414	(0.025)	(0.247)	0.180	(0.027)	(0.030 – 8.2x)
6	7	2000	3.066	99.88	1.593	(0.041)	(1.132)	0.285	(0.042)	(0.121 – 9.4x)
6	7	4000	7.173	162.49	5.072	(0.069)	(4.070)	0.814	(0.074)	(0.380 – 10.7x)
6	7	8000	15.61	NA	12.75	(0.122)	(10.95)	1.896	(0.130)	(0.939 – 11.7x)
9	7	500	1.171	7564.9	0.105	(0.013)	(0.040)	0.101	(0.024)	(0.010 – 4.0x)
9	7	1000	3.704	10010.4	0.524	(0.025)	(0.297)	0.233	(0.026)	(0.030 – 11.4x)
9	7	2000	13.43	NA	2.838	(0.042)	(1.973)	0.483	(0.045)	(0.193 – 10.2x)
9	7	4000	51.77	NA	18.35	(0.078)	(14.84)	2.325	(0.083)	(1.350 – 11.0x)
9	7	8000	NA	NA	116.6	(0.139)	(102.5)	11.50	(0.145)	(7.947 – 12.9x)

Table 1: Timings (real time in seconds) for Hensel lift of x_n .

Legend: $n = \#$ variables, $d = \deg(f, x_j) = \deg(g, x_j)$, $t = \#f = \#g$.

Benchmark 2: higher degree

n	d	t	Maple 2018		New time (1 core)			New time (16 cores)		
			best	worst	total	(hensel)	(eval)	total	(hensel)	(eval)
6	10	500	0.571	92.49	0.099	(0.029)	(0.025)	0.081	(0.024 – 1.2x)	(0.006)
6	15	500	0.751	7956.5	0.134	(0.070)	(0.016)	0.093	(0.034 – 2.1x)	(0.006)
6	20	500	0.919	48610.1	0.238	(0.168)	(0.017)	0.130	(0.065 – 2.6x)	(0.005)
6	40	500	1.615	NA	1.207	(1.128)	(0.015)	0.282	(0.203 – 5.6x)	
6	60	500	3.343	NA	4.580	(4.486)	(0.015)	0.732	(0.631 – 7.1x)	(0.011)
6	80	500	4.485	NA	13.76	(13.65)	(0.016)	1.674	(1.554 – 8.8x)	(0.012)
6	10	2000	5.237	976.94	1.775	(0.089)	(1.067)	0.374	(0.055 – 1.6x)	(0.121)
6	15	2000	7.166	23128.5	1.616	(0.221)	(0.706)	0.413	(0.107 – 2.1x)	(0.061)
6	20	2000	9.195	NA	1.635	(0.451)	(0.480)	0.431	(0.150 – 3.0x)	(0.040)
6	40	2000	15.98	NA	4.008	(2.993)	(0.260)	0.854	(0.505 – 5.9x)	(0.038)
6	60	2000	42.32	NA	14.25	(13.15)	(0.292)	1.926	(1.500 – 8.8x)	(0.052)
6	80	2000	57.33	NA	26.34	(25.25)	(0.217)	3.340	(2.839 – 8.9x)	(0.050)

Table 2: Timings (real time in seconds) for increasing degree d .

Legend: $n = \#$ variables, $d = \deg(f, x_j) = \deg(g, x_j)$, $t = \#f = \#g$.

Software Design Issues

For high performance we prefer arrays of machine integers

Data structure for $f = 3x^4 + 5x^2y^3z^4 - 2yz^6 + 7$ in $\mathbb{Z}_p[x, y, z]$.

array of coefficients (64 bit integers):

3	5	-2	7
---	---	----	---

array of monomials (64 bit integers):

4 0 0	2 3 4	0 1 6	0 0 9
-------	-------	-------	-------

For polynomials in more variables use `__int128_t` and `__uint128_t` in gcc.

Software Design Issues

For multi-core performance using Cilk C we like “in-place” algorithms which reduce memory management.

Solve $f_k g_0 + g_k f_0 = c_k$ given for $f_k, g_k \in \mathbb{Z}_p[x_1]$ given $sg_0 + tf_0 = 1$.

```
for( k=1; k<da; k++ ) {  
    ...  
    polcopy64s( ck, dck, T );           // T = ck  
    dT = poldiv64s( T, f0, dt, df0, p ); // T = Tk mod f0  
    dfk = polmul64s( T, s, dT, ds, fk, p ); // fk = T * s  
    dfk = poldiv64s( fk, f0, dfk, df0, p ); // fk = fk mod f0  
    ...  
}
```

All routines for $\mathbb{Z}_p[x_1]$ and $\mathbb{Z}_p[x_1, \dots, x_n]$ do not allocate memory. The array T is reused in the next iteration.

Thank you for attending!