On the Extended Hensel Construction and its Application to the Computation of Limit Points

Parisa Alvandi, Masoud Ataei, Marc Moreno Maza

University of Western Ontario

ISSAC 2017, 27 July, Kaiserslautern, Germany

Plan

1. Introduction
2. The Extended Hensel Construction
3. Yun-Moses Polynomials
4. Lifting the factors
5. Experimentation

Plan

Problem

Goal

Factorize $F(X, Y) \in \mathbb{C}[X, Y]$ into linear factors in X over $\mathbb{C}(\langle Y^* \rangle)$:

$$F(X, Y) = (X - \chi_1(Y))(X - \chi_2(Y)) \cdots (X - \chi_d(Y))$$

where each $\chi_i(Y)$ is a <u>Puiseux series</u>.

Puiseux Series

Series of the form

$$\chi_i(Y) = \sum_{k=a_i}^{\infty} c_k Y^{k/d_i}$$

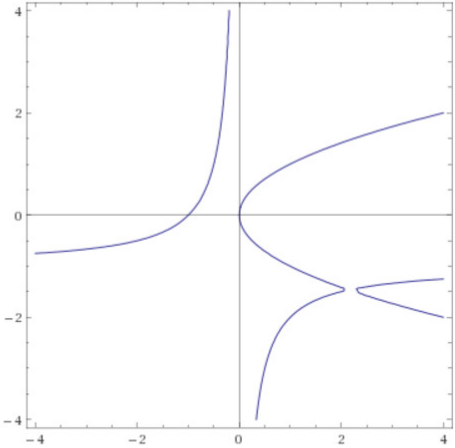
where

- $c_k \in \mathbb{C}$
- $a_i \in \mathbb{Z}$
- $d_i \in \mathbb{Z}_{>0}$

An example

$$F(X,Y) = Y^2X + Y^2 - YX^3 - YX^2 + Y - X^2.$$

- $\chi_1(Y) = \frac{-Y-1}{Y}$
- $\chi_2(Y) = Y$
- $\chi_3(Y) = -Y$



Another example

```
> P := PowerSeries([y, z]) :
  U := UnivariatePolynomialOverPowerSeries([y, z], x) :
  poly := y·x^3 + (−2 · y + z + 1)·x + y :
  U:−ExtendedHenselConstruction(poly, [0, 0], 3);
```

$$\left[\left[x = \frac{-RootOf(\_Z^2+y) + RootOf(\_Z^2+y)\, y - \frac{1}{2} RootOf(\_Z^2+y)\, z + \frac{1}{2} y^2}{y}\right],\right.$$

$$\left[x = \frac{RootOf(\_Z^2+y) - RootOf(\_Z^2+y)\, y + \frac{1}{2} RootOf(\_Z^2+y)\, z + \frac{1}{2} y^2}{y}\right],$$

$$\left.[x = -y]\right]$$

Previous works (1/2)

1. Extended Hensel Construction (EHC):
   - Introduction: F. Kako and T. Sasaki, 1999
   - Extensions:
     - M. Iwami, 2003,
     - D. Inaba, 2005,
     - D. Inaba and T. Sasaki 2007,
     - D. Inaba and T. Sasaki 2016.
2. Newton-Puiseux:
   - H. T. Kung and J. F. Traub, 1978,
   - D. V. Chudnovsky and G. V. Chudnovsky, 1986
   - A. Poteaux and M. Rybowicz, 2015.

Previous works (2/2)

- The Extended Hensel Construction (EHC) compute all branches concurrently
- while approaches based on Newton-Puiseux computes one branch after another.

For $F(X, Y) := -X^3 + Y\,X + Y$:

1. the EHC produces
   1. $\chi_1(Y) := Y^{\frac{1}{3}} + \frac{1}{3}\,Y^{\frac{2}{3}} + O(Y)$,
   2. $\chi_2(Y) := \frac{-1+\sqrt{-3}}{2}\,Y^{\frac{1}{3}} + \frac{1}{3}\,(\frac{-1-\sqrt{-3}}{2})\,Y^{\frac{2}{3}} + O(Y)$,
   3. $\chi_3(Y) := (\frac{-1-\sqrt{-3}}{2})\,Y^{\frac{1}{3}} + \frac{1}{3}\,(\frac{-1+\sqrt{-3}}{2})\,Y^{\frac{2}{3}} + O(Y)$.

Previous works (2/2)

- The Extended Hensel Construction (EHC) compute all branches concurrently
- while approaches based on Newton-Puiseux computes one branch after another.

For $F(X, Y) := -X^3 + Y\,X + Y$:

1. the EHC produces
   1. $\chi_1(Y) := Y^{\frac{1}{3}} + \frac{1}{3}\,Y^{\frac{2}{3}} + O(Y)$,
   2. $\chi_2(Y) := \frac{-1+\sqrt{-3}}{2}\,Y^{\frac{1}{3}} + \frac{1}{3}\,(\frac{-1-\sqrt{-3}}{2})\,Y^{\frac{2}{3}} + O(Y)$,
   3. $\chi_3(Y) := (\frac{-1-\sqrt{-3}}{2})\,Y^{\frac{1}{3}} + \frac{1}{3}\,(\frac{-1+\sqrt{-3}}{2})\,Y^{\frac{2}{3}} + O(Y)$.
2. Whereas Kung-Traub's method (based on Newton-Puiseux) computes
   1. $\chi_1(Y) := Y^{\frac{1}{3}} + \frac{1}{3}\,Y^{\frac{2}{3}} + O(Y)$,

Previous works (2/2)

- The Extended Hensel Construction (EHC) compute all branches concurrently
- while approaches based on Newton-Puiseux computes one branch after another.

For $F(X, Y) := -X^3 + Y\,X + Y$:

1. the EHC produces
   1. $\chi_1(Y) := Y^{\frac{1}{3}} + \frac{1}{3}\,Y^{\frac{2}{3}} + O(Y)$,
   2. $\chi_2(Y) := \frac{-1+\sqrt{-3}}{2}\,Y^{\frac{1}{3}} + \frac{1}{3}\,(\frac{-1-\sqrt{-3}}{2})\,Y^{\frac{2}{3}} + O(Y)$,
   3. $\chi_3(Y) := (\frac{-1-\sqrt{-3}}{2})\,Y^{\frac{1}{3}} + \frac{1}{3}\,(\frac{-1+\sqrt{-3}}{2})\,Y^{\frac{2}{3}} + O(Y)$.
2. Whereas Kung-Traub's method (based on Newton-Puiseux) computes
   1. $\chi_1(Y) := Y^{\frac{1}{3}} + \frac{1}{3}\,Y^{\frac{2}{3}} + O(Y)$,
   2. $\chi_2(Y) := \theta\,Y^{\frac{1}{3}} + \frac{\theta^2}{3}\,Y^{\frac{2}{3}} + O(Y)$,
   3. $\chi_3(Y) := \theta^2\,Y^{\frac{1}{3}} + \frac{\theta}{3}\,Y^{\frac{2}{3}} + O(Y)$,

   for $\theta \in \mathbb{C}$ such that $\theta^3 = 1, \theta^2 \neq 1, \theta \neq 1$, since $F(X, Y)$ is a Weierstrass polynomial.

Our contributions

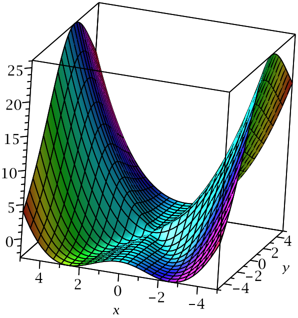
1. We show that the EHC requires only linear algebra and univariate polynomial arithmetic
2. We derive complexity estimates for the EHC
3. We obtain a competitive implementation against the original EHC and Kung-Traub's method
4. We apply the EHC to the problem of computing real limit points of quasi-components of regular chains.

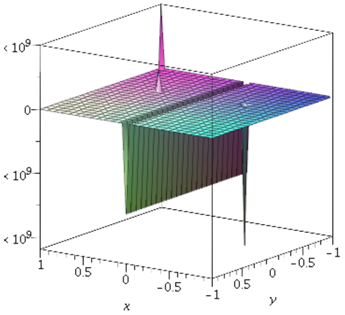
Application (1/2)

One immediate application of Extended Hensel Construction is finding the limit of a multivariate rational function i.e. for a rational function $q = \frac{P_1}{P_2}$ where P_1 and $P_2 \in \mathbb{F}[X_1, \ldots X_n]$, we can ask whether

$$\lim_{(X_1,\ldots,X_n)\to(x_1,\ldots,x_n)} q(X_1, \ldots, X_n)$$

exists and what it converges to.





Application (2/2)

```
> R := PolynomialRing([x, y, z]) :
  rc := Chain([y^(3)-2*y^(3) + y^(2) + z^(5), z^(4)*x + y^(3)-y^(2)], Empty(R), R) :
> LimitPoints(rc, R, coefficient = complex); Display(%, R);
```

[*regular_chain*, *regular_chain*]

$$\left[\begin{cases} x = 0 \\ y = 0 \\ z = 0 \end{cases}, \begin{cases} x = 0 \\ y - 1 = 0 \\ z = 0 \end{cases} \right]$$

```
> LimitPoints(rc, R, coefficient = real); Display(%, R);
```

[*regular_semi_algebraic_system*]

$$\left[\begin{cases} x = 0 \\ y - 1 = 0 \\ z = 0 \end{cases} \right]$$

```
> RegularChainBranches(rc, R, [z]);
```

$$\left[\left[z = T^2, y = \frac{1}{2} T^5 \left(-T^5 + 2\, RootOf\left(\_Z^2 + 1\right)\right), x = -\frac{1}{8} T^2 \left(-T^{20} + 6\, T^{15}\, RootOf\left(\_Z^2 + 1\right) + 10\, T^{10} + 8\right)\right], \left[z = T^2, y = -\frac{1}{2}\right.\right.$$

$$\left. + 2\, RootOf\left(\_Z^2 + 1\right)\right), x = \frac{1}{8} T^2 \left(T^{20} + 6\, T^{15}\, RootOf\left(\_Z^2 + 1\right) - 10\, T^{10} - 8\right)\Big], \left[z = T, y = T^5 + 1, x = -T\left(T^{10} + 2\, T^5 + 1\right.\right.$$

```
> RegularChainBranches(rc, R, [z], coefficient = real);
```

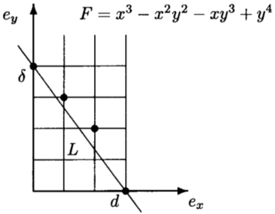
$$\left[\left[z = T, y = T^5 + 1, x = -T\left(T^{10} + 2\, T^5 + 1\right)\right]\right]$$

```
>
```

Plan

Newton Polynomial

- Let $F(x,y) \in \mathbb{C}[x,y]$ be square-free, monic in x and let $d := \deg_x(F)$.
- The "south-west-most" terms $c\,x^{e_x}y^{e_y}$ of $F(x,y)$ satisfy an equation of the form $e_x/d + e_y/\delta = 1$, with $\delta \in \mathbb{Q}$ and form the Newton polynomial $F^{(0)}(x,y)$ which is homogeneous $(x, y^{\delta/d})$ of degree d.
- Let $\hat{\delta}, \hat{d} \in \mathbb{Z}^{>0}$ such that: $\hat{\delta}/\hat{d} = \delta/d, \;\; \gcd(()\hat{\delta}, \hat{d}) = 1$.
$$F(x,y) \;\; = \;\; G_1^{(0)}(x,y) \cdots G_r^{(0)}(x,y)$$
where the $G_i^{(0)}(X,Y) := (X - \zeta_i Y^{\delta/d})^{m_i}$ are the initial factors.
- We define the ideal
$$S_k = \langle X^d Y^{(k+0)/\hat{d}}, X^{d-1} Y^{(k+\hat{\delta})/\hat{d}}, \ldots, X^0 Y^{(k+d\hat{\delta})/\hat{d}} \rangle, \qquad (1)$$
for $k = 1, 2, \ldots$.



Algorithm

Algorithm 1: EHC_Lift(F, k)

begin

Compute the Newton polynomial $F^{(0)}$ and $\hat{\delta}, \hat{d}$;

Compute $G_i^{(0)} = (X - \zeta_i Y)^{m_i}$, with $1 \leq i \leq r$;

Compute the Yun-Moses polynomial $W_i^{(\ell)}$ for $i = 1, \cdots, r$ and $\ell = 0, \ldots, d-1$;

for $j = 1, \ldots, k$ **do**

  Compute $\Delta F^{(j)}(X, Y) := F(X, Y) - \prod_{i=1}^{r} G_i^{(j-1)}$ mod $\bar{S}_{j+1}$;

  Compute $\Delta G_i^{(j)} = \sum_{\ell=0}^{m-1} W_i^{(\ell)} f_\ell^{(j)}$, for $i = 1, \cdots, r$;

  Let $G_i^{(j)} = G_i^{(j-1)} + \Delta G_i^{(j)}$ for $i = 1, \cdots, r$;

return $G_1^{(k)}, \ldots, G_r^{(k)}$;

Algorithm

Algorithm 2: EHC_Lift(F, k)

begin

Compute the Newton polynomial $F^{(0)}$ and $\hat{\delta}, \hat{d}$;

Compute $G_i^{(0)} = (X - \zeta_i Y)^{m_i}$, with $1 \leq i \leq r$;

Compute the Yun-Moses polynomial $W_i^{(\ell)}$ for $i = 1, \cdots, r$ and $\ell = 0, \cdots, d-1$;

for $j = 1, \ldots, k$ **do**

Compute $\Delta F^{(j)}(X, Y) := F(X, Y) - \prod_{i=1}^{r} G_i^{(j-1)} \mod \bar{S}_{j+1}$;

Compute $\Delta G_i^{(j)} = \sum_{\ell=0}^{m-1} W_i^{(\ell)} f_\ell^{(j)}$, for $i = 1, \cdots, r$;

Let $G_i^{(j)} = G_i^{(j-1)} + \Delta G_i^{(j)}$ for $i = 1, \cdots, r$;

return $G_1^{(k)}, \ldots, G_r^{(k)}$;

Algorithm

Algorithm 3: EHC_LiftF, k

begin

Compute the Newton polynomial $F^{(0)}$ and $\hat{\delta}, \hat{d}$;

Compute $G_i^{(0)} = (X - \zeta_i Y)^{m_i}$, with $1 \leq i \leq r$;

Compute the Yun-Moses polynomial $W_i^{(\ell)}$ for $i = 1, \cdots, r$ and $\ell = 0, \cdots, d-1$;

for $j = 1, \ldots, k$ **do**

Compute $\Delta F^{(j)}(X, Y) := F(X, Y) - \prod_{i=1}^{r} G_i^{(j-1)}$ mod $\bar{S}_{j+1}$;

Compute $\Delta G_i^{(j)} = \sum_{\ell=0}^{m-1} W_i^{(\ell)} f_\ell^{(j)}$, for $i = 1, \cdots, r$;

Let $G_i^{(j)} = G_i^{(j-1)} + \Delta G_i^{(j)}$ for $i = 1, \cdots, r$;

return $G_1^{(k)}, \ldots, G_r^{(k)}$;

Plan

Yun-Moses Polynomials (1/3)

Assume $G_1(X,Y),\dots,G_r(X,Y)$ are homogeneous polynomials. Regarding them as polynomials of $\mathbb{C}\langle Y\rangle[X]$, further assume

$$\gcd((\hat{G}_i,\hat{G}_j)) = 1 \text{ for } i \neq j,$$

Let $d := \deg(G_1(X,Y)\ \dots\ G_r(X,Y))$. Then, for each $\ell \in \{0,\dots,d-1\}$, there exists a unique set of polynomials $\{W_i^{(\ell)}(X,Y) \in \mathbb{C}\langle Y\rangle[X] \mid i = 1,\dots,r\}$ satisfying

$$W_1^{(\ell)}\left(\frac{G_1\cdots G_r}{G_1}\right) + \cdots + W_r^{(\ell)}\left(\frac{G_1\cdots G_r}{G_r}\right) = X^\ell Y^{d-\ell},$$

where $\deg_X(W_i^{(\ell)}(X,Y)) < \deg_X(G_i(X,Y)),\quad i = 1,\dots,r.$

Yun-Moses Polynomials (2/3)

Key observation

Let us fix $i := \lambda$. Writing $W_\lambda^{(\ell)} = \sum_{j=0}^{m_\lambda - 1} w_{\lambda,j}(\hat{Y}) X^j$, we have

$$\sum_{j=0}^{m_\lambda - 1} \frac{\partial^\mu}{\partial X^\mu} \left(X^j \frac{F^{(0)}}{G_\lambda^{(0)}} \right) \Big|_{X = \zeta_\lambda \hat{Y}} w_{\lambda,j}^{(\ell)} = \frac{\partial^\mu}{\partial X^\mu} (X^\ell \hat{Y}^{d-\ell}) \Big|_{X = \zeta_\lambda \hat{Y}} .$$

where ζ_λ is a root of $F^{(0)}(X, 1)$ and m_λ is its multiplicity

Consequences

- This is a system of linear equations $\mathcal{W}_\lambda \mathcal{X}_\lambda^{(\ell)} = \mathcal{B}_\lambda^{(\ell)}$.
- The matrix $\mathcal{W}_\lambda$ is a Wronskian matrix.

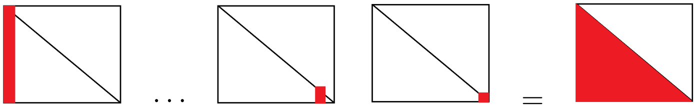
Yun-Moses Polynomials (3/3)

The inverse of $\mathcal{W}_\lambda$ is $\mathcal{W}_\lambda^{-1} = M_2 M_1$ where M_1 and M_2 are square matrices of order m_λ, defined as follows. The matrix M_1 writes $M_1 = M_{1(m_\lambda-1)} \cdots M_{11} M_{10}$ such that, for $j = 0, \cdots, m_\lambda - 1$, we have

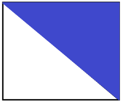
$$M_{1j} = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \frac{1}{j!f} & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & \binom{j+1}{j}\frac{-f'}{f} & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & \binom{m_\lambda-1}{j}\frac{-f^{(m_\lambda-1-j)}}{f} & 0 & \cdots & 1 \end{bmatrix}.$$

Hence, the matrix M_{1j} differs from the identity matrix only in its $(j+1)$-th column. The matrix M_2 is an upper triangular matrix $M_2 = [\gamma_{j,k}]$ with $\gamma_{j,k} = (-1)^{j+k}\binom{k}{k-j}\zeta_\lambda^{k-j}\hat{Y}^{k-j}$ if $j \leq k$ and $\gamma_{j,k} = 0$ if $j > k$, for $j, k \in \{0, 1, \ldots, m_\lambda - 1\}$.

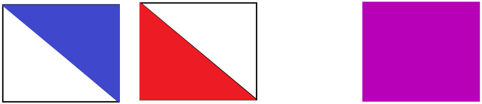
Matrix M_1



Matrix M_2



Matrix $\mathcal{W}_i^{-1} = M_2 M_1$



Complexity Result:

Theorem 1:

One can compute all the Yun-Moses polynomials $W_i^{(\ell)}$ $(0 \leq \ell \leq d-1,\ 1 \leq i \leq r)$, within

- $\mathcal{O}(d^3)$ operations in $\mathbb{C}$, or
- $\mathcal{O}(d^3\,\mathsf{M}(d))$ operations in the field of coefficients of $F(X, Y)$.

Plan

Algorithm

Algorithm 4: EHC_LiftF, k

begin

Compute the Newton polynomial $F^{(0)}$ and $\hat{\delta}, \hat{d}$;

Compute $G_i^{(0)} = (X - \zeta_i Y)^{m_i}$, with $1 \leq i \leq r$;

Compute the Yun-Moses polynomial $W_i^{(\ell)}$ for $i = 1, \cdots, r$ and $\ell = 0, \cdots, d-1$;

for $\underline{j = 1, \ldots, k}$ **do**

Compute

$\Delta F^{(j)}(X, Y) := F(X, Y) - \prod_{i=1}^{r} G_i^{(j-1)} \mod \bar{S}_{j+1}$;

Compute $\Delta G_i^{(j)} = \sum_{\ell=0}^{m-1} W_i^{(\ell)} f_\ell^{(j)}$, for $i = 1, \cdots, r$;

Let $G_i^{(j)} = G_i^{(j-1)} + \Delta G_i^{(j)}$ for $i = 1, \cdots, r$;

return $G_1^{(k)}, \ldots, G_r^{(k)}$;

Computing $\Delta F^{(j)}(X, Y)$

Goal

$$\Delta F^{(j)}(X, Y) := F(X, Y) - \prod_{i=1}^{r} G_i^{(j-1)} \mod \bar{S}_{j+1}$$

Oobservation

- $G_i^{(j-2)} := G_i^{(0)} + \Delta G_i^{(1)} + \cdots + \Delta G_i^{(j-2)}$
- $G_i^{(j-1)} := G_i^{(0)} + \Delta G_i^{(1)} + \cdots + \Delta G_i^{(j-2)} + \Delta G_i^{(j-1)}$

Hence, we aim at recycling terms in the product $\prod_{i=1}^{r} G_i^{(j-1)}$ mod $\bar{S}_{j+1}$ computed from previous iterations.

Notations

1. $P_2^{k+1} := \prod_{i=1}^{2} G_i^{(k)} \mod \overline{S}_{k+1}$
2. $P_j^{k+1} := \prod_{i=1}^{j} G_i^{(k)} \mod \overline{S}_{k+1}$, for $j = 3, \ldots, r$.

We want

$$P_r^{k+1} = \prod_{i=1}^{r} G_i^{(k)} \mod \overline{S}_{k+2}$$

Computing $\Delta F^{(j)}(X, Y)$

Initially define: $P_j^1 \equiv G_1^{(0)} \cdots G_j^{(0)} \mod S_2$, for $j = 2, \cdots, r$. and recursively compute:

$$P_2^{k+1} = P_2^k + (\Delta_1^0 \Delta_2^k + \Delta_1^k \Delta_2^0)\tilde{Y}^k + (\Delta_1^1 \Delta_2^k + \cdots + \Delta_1^k \Delta_2^1)\tilde{Y}^{k+1} = \prod_{i=1}^{2} G_i^{(k)}$$

Now for $j = 3, \ldots, r$, define

$$P_j^k \equiv P_{j-1}^k G_j^{(k-1)} \mod S_{k+1}$$

and assume q_j^{k+1} is recursively given by

$$q_j^{k+1} = p_{j-1}^{k+1,0} \Delta_j^k + q_{j-1}^{k+1} \Delta_j^0 \quad \text{with} \quad q_2^{k+1} = \Delta_2^k \Delta_1^0 + \Delta_2^0 \Delta_1^k. \quad (2)$$

where $p_{j-1}^{k+1,0}$ is the coefficient of $\tilde{Y}^0$ in P_{j-1}^{k+1}. We can compute

$$P_j^{k+1} = P_j^k + q_j^{k+1}\tilde{Y}^k + \left(p_{j-1}^{k+1,1}\Delta_j^k + \cdots + p_{j-1}^{k+1,k+1}\Delta_j^0\right)\tilde{Y}^{k+1} = \prod_{i=1}^{j} G_i^{(k)}$$

Computing $\Delta F^{(j)}(X, Y)$

$$
\begin{tikzcd}
G_1^{(0)}\,G_2^{(0)} & P_2^1 \arrow[r] \arrow[d] & P_2^2 \arrow[r] \arrow[d] & P_2^3 \arrow[r] \arrow[d] & P_2^4 \arrow[r] \arrow[d] & P_2^5 \arrow[r] \arrow[d] & \cdots \\
& P_3^1 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & \cdots \\
& P_4^1 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & \cdots \\
& P_5^1 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & \cdots \\
& P_6^1 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & \cdots
\end{tikzcd}
$$

Computing $\Delta F^{(j)}(X, Y)$

$$
\begin{tikzcd}
G_1^{(0)} G_2^{(0)} & P_2^1 \arrow[r] \arrow[d] & P_2^2 \arrow[r] \arrow[d] & P_2^3 \arrow[r] \arrow[d] & P_2^4 \arrow[r] \arrow[d] & P_2^5 \arrow[r] \arrow[d] & \cdots \\
P_2^1 G_3^{(0)} & P_3^1 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & \cdots \\
 & P_4^1 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & \cdots \\
 & P_5^1 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & \cdots \\
 & P_6^1 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & \cdots
\end{tikzcd}
$$

Computing $\Delta F^{(j)}(X, Y)$

$$
\begin{tikzcd}
G_1^{(0)} G_2^{(0)} & P_2^1 \arrow[r] \arrow[d] & P_2^2 \arrow[r] \arrow[d] & P_2^3 \arrow[r] \arrow[d] & P_2^4 \arrow[r] \arrow[d] & P_2^5 \arrow[r] \arrow[d] & \cdots \\
P_2^1 G_3^{(0)} & P_3^1 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & \cdots \\
P_3^1 G_4^{(0)} & P_4^1 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & \cdots \\
 & P_5^1 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & \cdots \\
 & P_6^1 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & \cdots
\end{tikzcd}
$$

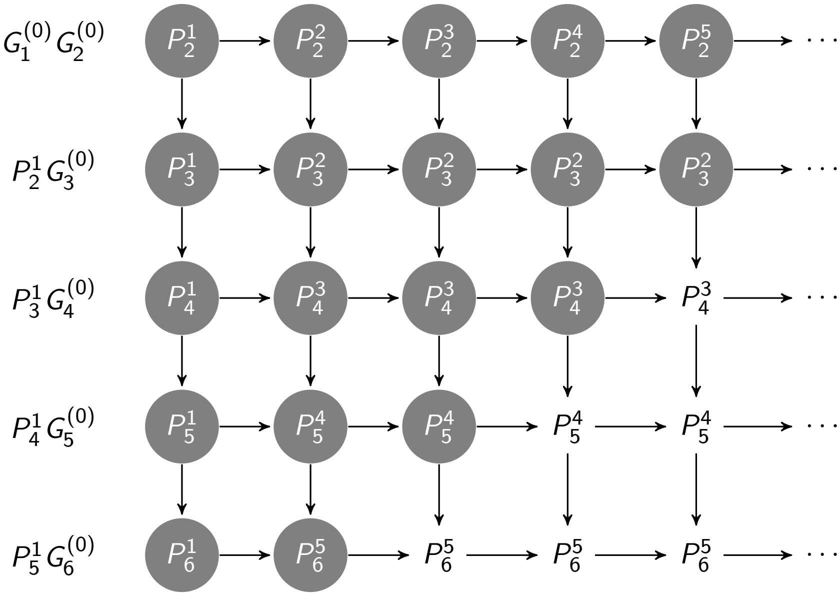
Computing $\Delta F^{(j)}(X, Y)$

$$
\begin{tikzcd}
G_1^{(0)}\, G_2^{(0)} & P_2^1 \arrow[r] \arrow[d] & P_2^2 \arrow[r] \arrow[d] & P_2^3 \arrow[r] \arrow[d] & P_2^4 \arrow[r] \arrow[d] & P_2^5 \arrow[r] \arrow[d] & \cdots \\
P_2^1\, G_3^{(0)} & P_3^1 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & \cdots \\
P_3^1\, G_4^{(0)} & P_4^1 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & \cdots \\
P_4^1\, G_5^{(0)} & P_5^1 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & \cdots \\
 & P_6^1 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & \cdots
\end{tikzcd}
$$

Computing $\Delta F^{(j)}(X, Y)$

$$\begin{tikzcd}
G_1^{(0)} G_2^{(0)} & P_2^1 \arrow[r] \arrow[d] & P_2^2 \arrow[r] \arrow[d] & P_2^3 \arrow[r] \arrow[d] & P_2^4 \arrow[r] \arrow[d] & P_2^5 \arrow[r] \arrow[d] & \cdots \\
P_2^1 G_3^{(0)} & P_3^1 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & \cdots \\
P_3^1 G_4^{(0)} & P_4^1 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & \cdots \\
P_4^1 G_5^{(0)} & P_5^1 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & \cdots \\
P_5^1 G_6^{(0)} & P_6^1 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & \cdots
\end{tikzcd}$$

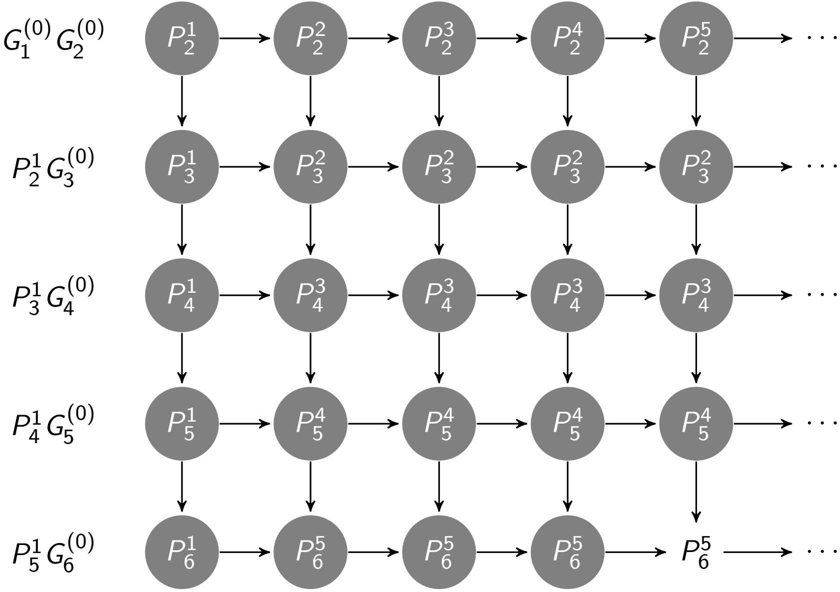
Computing $\Delta F^{(j)}(X, Y)$



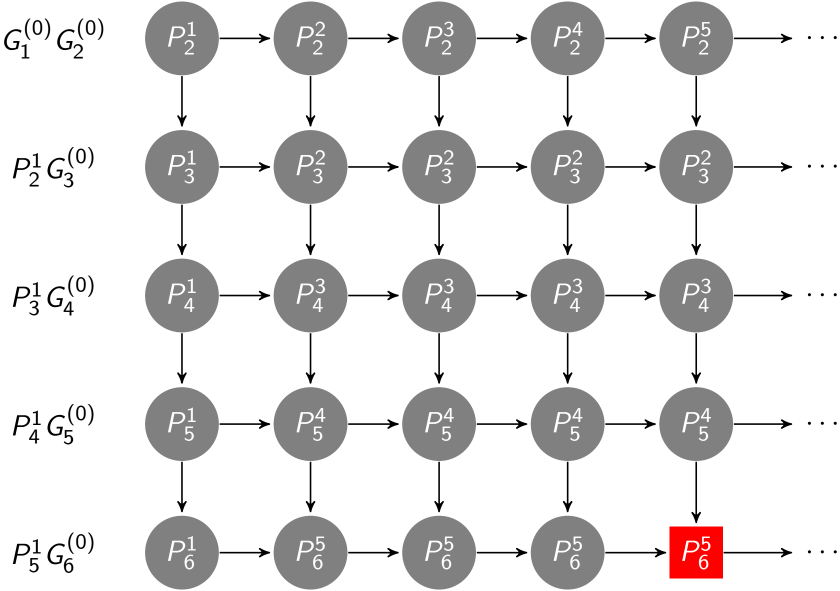
Computing $\Delta F^{(j)}(X, Y)$

$$
\begin{tikzcd}
G_1^{(0)} G_2^{(0)} & P_2^1 \arrow[r] \arrow[d] & P_2^2 \arrow[r] \arrow[d] & P_2^3 \arrow[r] \arrow[d] & P_2^4 \arrow[r] \arrow[d] & P_2^5 \arrow[r] \arrow[d] & \cdots \\
P_2^1 G_3^{(0)} & P_3^1 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & P_3^2 \arrow[r] \arrow[d] & \cdots \\
P_3^1 G_4^{(0)} & P_4^1 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & P_4^3 \arrow[r] \arrow[d] & \cdots \\
P_4^1 G_5^{(0)} & P_5^1 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & P_5^4 \arrow[r] \arrow[d] & \cdots \\
P_5^1 G_6^{(0)} & P_6^1 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & P_6^5 \arrow[r] & \cdots
\end{tikzcd}
$$

Computing $\Delta F^{(j)}(X, Y)$



Computing $\Delta F^{(j)}(X, Y)$



Complexity result:

Theorem 2:

he k-th iteration of Step 9 in the Algorithm 4 runs within

- $\mathcal{O}(k\, d\mathsf{M}(d))$ operations in $\mathbb{C}$,
- $\mathcal{O}(k\, d\mathsf{M}(d)^2)$ operations in the field of coefficients of $F(X, Y)$.

Comparative complexity results

Theorem 3:

Our enhancement of the EHC computes all the branches in $\mathcal{O}(k^2\, d\, \mathsf{M}(d))$ operations in $\mathbb{C}$, using a linear lifting scheme.

Kung-Traub, 1987

The first k iterations of Newton-Puiseux on an input bivariate polynomial of degree d computes all branches within

- $\mathcal{O}(d^2\, k\, \mathsf{M}(k))$ operations in $\mathbb{C}$ using a linear lifting scheme (Theorem 5.2 in their paper)
- $\mathcal{O}(d^2\, \mathsf{M}(k))$ operations in $\mathbb{C}$ using a quadratic lifting scheme (Corollary 5.1 in their paper)

D. V. Chudnovsky and G. V. Chudnovsky, 2015

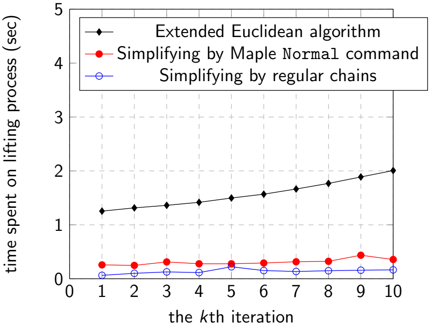
The latter estimate reported by Kung and Traub is improved to $\mathcal{O}(d^2\, k)$ operations in $\mathbb{C}$ for computing all the branches.

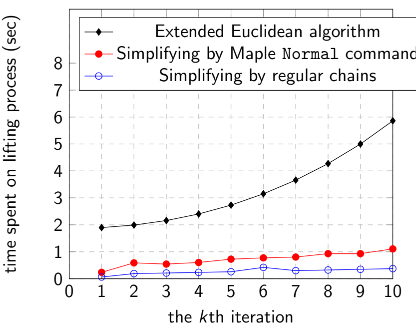
Remark

A quadratic lifting scheme for the EHC is work in progress.

Plan

Experimentation for lifting





The y axis is in square-root scale.

Experimentation for factoring (1/2)

Ex	MD	KT Lin		KT Quad		EHCWM			EHCEEA	
		KT10	KT20	KT10	KT20	EHC10	YM1	EHC20	EHC10	YM2
1	5	2.22	18.6	4.93	4.91	0.48	0.22	0.73	0.90	0.21
7	4	5.60	65.8	0.56	0.58	0.22	0.14	0.23	0.34	0.13
8	4	14.9	230	1.25	1.25	0.23	0.13	0.28	0.36	0.12
9	3	5.53	114	1.51	1.56	0.30	0.11	0.39	0.88	0.10
10	3	2.71	42.0	0.28	0.63	0.16	0.08	0.20	0.32	0.12
11	3	0.46	2.34	0.21	0.21	0.16	0.08	0.17	0.26	0.12
12	3	0.50	6.86	0.28	0.32	0.16	0.08	0.18	0.30	0.12
13	4	0.86	10.9	0.50	0.48	0.26	0.15	0.28	0.46	0.24
14	4	3.21	34.8	0.69	0.71	0.26	0.15	0.34	0.52	0.24
15	6	27.6	535	4.85	4.85	0.64	0.42	0.82	2.05	1.08
16	7	45.6	836	8.45	9.91	0.64	0.43	0.92	2.33	1.74

Table: Comparing EHC versus Kung-Traub's method

Experimentation for factoring (2/2)

Ex	MD	KT Lin		KT Quad		EHCWM			EHCEEA	
		KT10	KT20	KT10	KT20	EHC10	YM1	EHC20	EHC10	YM2
17	7	145	∞	23.4	23.2	0.78	0.43	3.37	4.12	1.77
19	4	0.14	0.16	0.16	0.14	0.39	0.26	0.45	0.51	0.15
20	4	2.79	7.98	0.77	0.82	0.26	0.15	0.29	0.50	0.24
21	4	8.58	143	1.96	1.93	0.23	0.12	0.31	0.47	0.16
24	5	2.90	24.8	1.11	1.11	0.26	0.15	0.35	0.49	0.17
25	7	1.83	9.45	0.90	1.00	0.46	0.31	0.50	0.73	0.42
26	8	2.35	12.3	3.09	3.29	0.66	0.53	0.74	2.18	1.80
27	8	60.8	2876	23.1	27.1	0.77	0.53	1.20	2.31	1.28
28	9	215	∞	73.8	123	1.88	1.03	2.11	7.03	4.92
30	17	∞	∞	∞	∞	39.8	6.70	41.3	53.8	16.5
31	32	∞	∞	∞	∞	599	24.9	∞	∞	∞
32	33	∞	∞	∞	∞	224	25.0	∞	∞	∞

Table: Comparing EHC versus Kung-Traub's method

Concluding remarks

- We have shown that the EHC requires only linear algebra and univariate polynomial arithmetic
- We have derive complexity estimates for the EHC which are comparable to those of Kung-Traub's method in linear lifting scheme
- Experimentally, this enhanced EHC is competitive and sometimes outperforms Kung-Traub's method in its linear and quadratic lifting schemes.
- A quadratic lifting scheme for the EHC is work in progress.
- Source code and experimental data are available in the `PowerSeries` library from `www.regularchains.org`.