Approximate Factorization of Complex Multivariate Polynomials

Erich Kaltofen

North Carolina State University

http://www.kaltofen.us

Joint work with Shuhong Gao, John May, Zhengfeng Yang, and Lihong Zhi

May and Yang received the ACM SIGSAM’s ISSAC 2004 Distinguished Student Author Award for this work
Factorization of noisy polynomials over the complex numbers [my ’98 “Challenges”]

\[ 81x^4 + 16y^4 - 648z^4 + 72x^2y^2 - 648x^2 - 288y^2 + 1296 = 0 \]

\[ (9x^2 + 4y^2 + 18\sqrt{2}z^2 - 36)(9x^2 + 4y^2 - 18\sqrt{2}z^2 - 36) = 0 \]

\[ 81x^4 + 16y^4 - 648.003z^4 + 72x^2y^2 + .002x^2z^2 + .001y^2z^2 - 648x^2 - 288y^2 - .007z^2 + 1296 = 0 \]
“D. Izraelevitz at Massachusetts Institute of Technology has already implemented a version of algorithm 1 using complex floating point arithmetic. Early experiments indicate that the linear systems computed in step (L) tend to be **numerically ill-conditioned**. How to overcome this numerical problem is an important question which we will investigate.”
The Approximate Factorization Problem [LATIN ’94]

Given \( f \in \mathbb{C}[x, y] \) irreducible, find \( \tilde{f} \in \mathbb{C}[x, y] \) s.t. \( \deg \tilde{f} \leq \deg f \), \( \tilde{f} \) factors, and \( \|f - \tilde{f}\| \) is minimal.
The Approximate Factorization Problem [LATIN ’94]

Given $f \in \mathbb{C}[x,y]$ irreducible, find $\tilde{f} \in \mathbb{C}[x,y]$ s.t. $\deg \tilde{f} \leq \deg f$, $\tilde{f}$ factors, and $\|f - \tilde{f}\|$ is minimal.

Problem depends on choice of norm $\| \cdot \|$, and notion of degree.

We use 2-norm, and multi-degree: $\text{mdeg } f = (\deg_x f, \deg_y f)$
The Approximate Factorization Problem [LATIN ’94]

Given $f \in \mathbb{C}[x,y]$ irreducible, find $\tilde{f} \in \mathbb{C}[x,y]$ s.t. $\deg \tilde{f} \leq \deg f$, $\tilde{f}$ factors, and $\|f - \tilde{f}\|$ is minimal.

Problem depends on choice of norm $\| \cdot \|$, and notion of degree.

We use 2-norm, and multi-degree: $\text{mdeg } f = (\deg_x f, \deg_y f)$

Degree bound is important:

$(1 + \delta x)f$ is reducible but for $\delta < \varepsilon / \|f\|$, 

$$\| (1 + \delta x)f - f \| = \| \delta x f \| = \delta \| f \| < \varepsilon$$
State of the Approximate Factorization

- No polynomial time algorithm (except for constant degree factors [Hitz, Kaltofen, Lakshman ’99])
State of the Approximate Factorization

- No polynomial time algorithm (except for constant degree factors [Hitz, Kaltofen, Lakshman ’99])

- Several algorithms and heuristics to find a nearby factorizable $\hat{f}$ if $f$ is “nearly factorizable” [Corless et al. ’01 & ’02, Galligo and Rupprecht ’01, Galligo and Watt ’97, Huang et al. ’00, Sasaki ’01,...]
State of the Approximate Factorization

• No polynomial time algorithm (except for constant degree factors [Hitz, Kaltofen, Lakshman ’99])

• Several algorithms and heuristics to find a nearby factorizable $\hat{f}$ if $f$ is “nearly factorizable” [Corless et al. ’01 & ’02, Galligo and Rupprecht ’01, Galligo and Watt ’97, Huang et al. ’00, Sasaki ’01,...]

• There are lower bounds for $\min ||f - \hat{f}||$ (“irreducibility radius”) [Kaltofen and May ISSAC 2003]
Our ISSAC’04 Results

- A new practical algorithm to compute approximate multivariate GCDs
Our ISSAC’04 Results

- A new practical algorithm to compute approximate multivariate GCDs
- A practical algorithm to find the factorization of a nearby factorizable polynomial given any $f$

especially “noisy” $f$:

Given $f = f_1 f_2 + f_{\text{noise}}$, we find $\tilde{f}_1, \tilde{f}_2$ s.t. $\|f_1 f_2 - \tilde{f}_1 \tilde{f}_2\| \approx \|f_{\text{noise}}\|$ even for large noise: $\|f_{\text{noise}}\|/\|f\| \geq 10^{-3}$
Maple Demonstration
Ruppert’s Theorem

\[ f \in \mathbb{K}[x,y], \ \text{mdeg } f = (m,n) \]

\[ \mathbb{K} \text{ is a field, algebraically closed, and characteristic } 0 \]

Theorem. \( f \) is reducible \( \iff \exists g, h \in \mathbb{K}[x,y], \) non-zero,

\[ \frac{\partial g}{\partial y f} - \frac{\partial h}{\partial x f} = 0 \]

\[ \text{mdeg } g \leq (m-2,n), \text{ mdeg } h \leq (m,n-1) \]
Ruppert’s Theorem

\[ f \in \mathbb{K}[x, y], \ mdeg \ f = (m, n) \]

\( \mathbb{K} \) is a field, algebraically closed, and characteristic 0

**Theorem.** \( f \) is reducible \( \iff \exists g, h \in \mathbb{K}[x, y], \) non-zero,

\[
\frac{\partial g}{\partial y} f - \frac{\partial h}{\partial x} f = 0
\]

\[ \text{mdeg } g \leq (m - 2, n), \ \text{mdeg } h \leq (m, n - 1) \]

PDE \( \sim \) linear system in the coefficients of \( g \) and \( h \)
Gao’s PDE based Factorizer

Change degree bound: \( \text{mdeg } g \leq (m - 1, n), \text{mdeg } h \leq (m, n - 1) \)

so that: \( \text{# linearly indep. solutions to the PDE } = \text{# factors of } f \)

Require square-freeness: \( \text{GCD}(f, \frac{\partial f}{\partial x}) = 1 \)
Gao’s PDE based Factorizer

Change degree bound: \( \text{mdeg } g \leq (m - 1, n), \text{mdeg } h \leq (m, n - 1) \)

so that: # linearly indep. solutions to the PDE = # factors of \( f \)

Require square-freeness: \( \text{GCD}(f, \frac{\partial f}{\partial x}) = 1 \)

Let

\[
G = \text{Span}_\mathbb{C}\{g \mid [g, h] \text{ is a solution to the PDE}\}.
\]

Any solution \( g \in G \) satisfies \( g = \sum_{i=1}^{r} \lambda_i \frac{\partial f_i}{\partial x} f_i \) with \( \lambda_i \in \mathbb{C} \), so

\[
f = f_1 \cdots f_r = \prod_{\lambda \in \mathbb{C}} \gcd(f, g - \lambda \frac{\partial f}{\partial x})
\]

(\( f_i \) the distinct irreducible factors of \( f \))

With high probability \( \exists \) distinct \( \lambda_i \) s.t. \( f_i = \gcd(f, g - \lambda_i \frac{\partial f}{\partial x}) \)
Gao's PDE based Factorizer

Algorithm

**Input:** $f \in \mathbb{K}[x, y]$, $\mathbb{K} \subseteq \mathbb{C}$

**Output:** $f_1, \ldots, f_r \in \mathbb{C}[x, y]$

1. Find a basis for the linear space $G$, and choose a random element $g \in G$.

2. Compute the polynomial $E_g = \prod_i (z - \lambda_i)$ via an eigenvalue formulation

   If $E_g$ not squarefree, choose a new $g$

3. Compute the factors $f_i = \gcd(f, g - \lambda_i \frac{\partial f}{\partial x})$ in $\mathbb{K}(\lambda_i)$.

In exact arithmetic the extension field $\mathbb{K}(\lambda_i)$ is found via univariate factorization.
Adapting to the Approximate Case

The following must be solved to create an approximate factorizer from Gao’s algorithm:

1. Computing approximate GCDs of bivariate polynomials;

2. Determining the numerical dimension of $G$, and computing an approximate solution $g$;

3. Computing a $g$ s.t. the polynomial $E_g$ has no clusters of roots.
Determining the Number of Approximate Factors

Let $\text{Rup}(f)$ be the matrix from Gao’s algorithm

Recall:

$$\# \text{ of factors of } f = \text{Nullity}(\text{Rup}(f))$$
Determining the Number of Approximate Factors

Let $\text{Rup}(f)$ be the matrix from Gao’s algorithm

Recall:

$$\text{# of factors of } f = \text{Nullity(} \text{Rup}(f)) \text{)$$

$\text{Rup}(f)$ has nullity $r$ if

$$\sigma_m \geq \ldots \geq \sigma_{r+1} \neq 0 \text{ and } \sigma_r = \ldots = \sigma_1 = 0.$$ 

Say $\text{Rup}(f)$ has nullity $r$ with tolerance $\varepsilon$ if:

$$\sigma_m \geq \ldots \geq \sigma_{r+1} > \varepsilon \geq \sigma_r \geq \ldots \geq \sigma_1$$

Find a “best” $\varepsilon$ from the largest gap
choose $\varepsilon = \sigma_r$ s.t. $\sigma_{r+1}/\sigma_r$ is maximal
Determining the Number of Approximate Factors

If $f$ is irreducible
largest gap in the sing. values of $\text{Rup}(f) \sim \# \text{ of approx. factors}$

Recall:

$$G = \text{Span}_\mathbb{C}\{g \mid [g,h] \in \text{Nullspace}(\text{Rup}(f))\}$$

If $r$ is position of the largest gap in the sing. values of $\text{Rup}(f)$,
approx. version of $G$ is Span of last $r$ sing. vectors of $\text{Rup}(f)$
Approximate Factorization

Input: \( f \in \mathbb{C}[x,y] \) abs. irreducible, approx. square-free

Output: \( f_1, \ldots, f_r \) approx. factors of \( f \), and \( c \)

1. Compute the SVD of \( \text{Rup}(f) \), determine \( r \), its approximate nullity, and choose \( g = \sum a_ig_i \), a random linear combination of the last \( r \) right singular vectors

2. compute \( E_g \) and its roots via an eigenvalue computation

3. For each \( \lambda_i \) compute the approximate GCD
\[
f_i = \gcd(f, g - \lambda_if)\]
and find an optimal scaling:
\[
\min_c \| f - c \prod_{i=1}^r f_i \|
\]
Approx. GCD: Generalized Sylvester Matrix

A pair \( g, h \in \mathbb{K}[x, y] \) has GCD of degree at least \( k \) iff
\[ \exists \text{ non-zero solutions } u, v \in \mathbb{K}[x, y] \text{ to:} \]
\[ \frac{g}{h} = \frac{v}{u}, \quad \text{tdeg}(u) \leq \text{tdeg}(h) - k, \quad \text{tdeg}(v) \leq \text{tdeg}(g) - k \]
or
\[ ug - vh = 0, \quad \text{tdeg}(u) \leq \text{tdeg}(h) - k, \quad \text{tdeg}(v) \leq \text{tdeg}(g) - k \]

Equation gives a linear system in the coefficients of \( u \) and \( v \)

Denote the matrix of the system \( \text{Syl}_k(g, h) \)
Computing the Approximate GCD

**Input:** $g$ and $h$ relatively prime

**Output:** $d \not\in \mathbb{K}$, approx. GCD of $g$ and $h$

1. Find $p$ from the largest gap in the singular values of $\text{Syl}_1(g, h)$

2. Find $k \in \mathbb{Z}$ which solves $\min_k \left| p - \binom{k+2}{2} \right|

3. Find $[u, v]$, the right singular vector corresponding to smallest singular value of $\text{Syl}_k(g, h)$
   [compute with an iterative method]

4. Find a $d$ to minimize $\|h - du\|_2^2 + \|g - dv\|_2^2$, using least squares ("Approximate division")

Also possible to add iterative improvement à la Zeng&Dayton’04
Notes on the Repeated Factor Case

We say $f$ is approximately square-free if:

dist. to nearest reducible poly. < dist. to nearest non-square-free poly.
Notes on the Repeated Factor Case

We say $f$ is approximately square-free if:

dist. to nearest reducible poly. < dist. to nearest non-square-free poly.

We handle the repeated factor case differently than usual:
without iterating approximate GCDs:

Compute the approximate quotient $\bar{f}$ of $f$ and $\gcd(f, \frac{df}{dx})$ and
factor the approximately square-free kernel $\bar{f}$
Notes on the Repeated Factor Case

We say $f$ is approximately square-free if:

dist. to nearest reducible poly. $< \text{dist. to nearest non-square-free poly.}$

We handle the repeated factor case differently than usual:
without iterating approximate GCDs:

Compute the approximate quotient $\bar{f}$ of $f$ and $\gcd(f, \frac{\partial f}{\partial x})$ and
factor the approximately square-free kernel $\bar{f}$

Determine multiplicity of approximate factors $f_i$ by comparing
the degrees of the approximate GCDs:

$$\gcd(f_i, \partial^k f / \partial x^k)$$
Table of Benchmarks

<table>
<thead>
<tr>
<th>Example</th>
<th>tdeg$(f_i)$</th>
<th>$\frac{\sigma_{r+1}}{\sigma_r}$</th>
<th>$\frac{\sigma_r}{|R(f)|_2}$</th>
<th>coeff. error</th>
<th>backward error</th>
<th>time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nagasaka’02</td>
<td>2, 3</td>
<td>11</td>
<td>$10^{-3}$</td>
<td>$10^{-2}$</td>
<td>$1.08e–2$</td>
<td>14.631</td>
</tr>
<tr>
<td>Kaltofen’00</td>
<td>2, 2</td>
<td>$10^9$</td>
<td>$10^{-10}$</td>
<td>$10^{-4}$</td>
<td>$1.02e–9$</td>
<td>13.009</td>
</tr>
<tr>
<td>Sasaki’01</td>
<td>5, 5</td>
<td>$10^9$</td>
<td>$10^{-10}$</td>
<td>$10^{-13}$</td>
<td>$8.30e–10$</td>
<td>5.258</td>
</tr>
<tr>
<td>Sasaki’01</td>
<td>10, 10</td>
<td>$10^5$</td>
<td>$10^{-6}$</td>
<td>$10^{-7}$</td>
<td>$1.05e–6$</td>
<td>85.96</td>
</tr>
<tr>
<td>Corless et al’01</td>
<td>7, 8</td>
<td>$10^7$</td>
<td>$10^{-8}$</td>
<td>$10^{-9}$</td>
<td>$1.41e–8$</td>
<td>19.628</td>
</tr>
<tr>
<td>Corless et al’02</td>
<td>3, 3, 3</td>
<td>$10^8$</td>
<td>$10^{-10}$</td>
<td>0</td>
<td>$1.29e–9$</td>
<td>9.234</td>
</tr>
<tr>
<td>Zeng’04</td>
<td>$(5)^3, 3, (2)^4$</td>
<td>$10^7$</td>
<td>$10^{-9}$</td>
<td>$10^{-10}$</td>
<td>$2.09e–7$</td>
<td>73.52</td>
</tr>
</tbody>
</table>
# Table of Benchmarks

<table>
<thead>
<tr>
<th>Example</th>
<th>$tdeg(f_i)$</th>
<th>$\frac{\sigma_{r+1}}{\sigma_r}$</th>
<th>$\frac{\sigma_r}{|R(f)|_2}$</th>
<th>coeff. error</th>
<th>backward error</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random ($f_i \in \mathbb{Z}$)</td>
<td>9, 7</td>
<td>486</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
<td>2.14e–4</td>
<td>43.823</td>
</tr>
<tr>
<td>&quot;</td>
<td>6, 6, 10</td>
<td>$10^3$</td>
<td>$10^{-6}$</td>
<td>$10^{-5}$</td>
<td>2.47e–4</td>
<td>539.67</td>
</tr>
<tr>
<td>&quot;</td>
<td>4, 4, 4, 4, 4</td>
<td>273</td>
<td>$10^{-6}$</td>
<td>$10^{-5}$</td>
<td>1.31e–3</td>
<td>3098.</td>
</tr>
<tr>
<td>&quot;</td>
<td>3, 3, 3</td>
<td>1.70</td>
<td>$10^{-3}$</td>
<td>$10^{-1}$</td>
<td>7.93e–1</td>
<td>29.25</td>
</tr>
<tr>
<td>&quot;</td>
<td>18, 18</td>
<td>$10^4$</td>
<td>$10^{-7}$</td>
<td>$10^{-6}$</td>
<td>3.75e–6</td>
<td>3173.</td>
</tr>
<tr>
<td>&quot;</td>
<td>12, 7, 5</td>
<td>8.34</td>
<td>$10^{-4}$</td>
<td>$10^{-3}$</td>
<td>8.42e–3</td>
<td>4370.</td>
</tr>
<tr>
<td>Not Sqr Free</td>
<td>5, $(5)^2$</td>
<td>$10^3$</td>
<td>$10^{-5}$</td>
<td>$10^{-5}$</td>
<td>6.98e–5</td>
<td>34.28</td>
</tr>
<tr>
<td>3 variables</td>
<td>5, 5</td>
<td>$10^4$</td>
<td>$10^{-5}$</td>
<td>$10^{-5}$</td>
<td>1.72e–5</td>
<td>332.99</td>
</tr>
<tr>
<td>$f_i \in \mathbb{C}$</td>
<td>6, 6</td>
<td>$10^6$</td>
<td>$10^{-8}$</td>
<td>$10^{-7}$</td>
<td>2.97e–7</td>
<td>30.034</td>
</tr>
</tbody>
</table>
# Table of Benchmarks

<table>
<thead>
<tr>
<th>Example</th>
<th>tdeg($f_i$)</th>
<th>$\frac{\sigma_{r+1}}{\sigma_r}$</th>
<th>$\frac{\sigma_r}{|R(f)|_2}$</th>
<th>coeff. error</th>
<th>backward error</th>
<th>time(sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random ($f_i \in \mathbb{Z}$)</td>
<td>9, 7</td>
<td>486</td>
<td>$10^{-4}$</td>
<td>$10^{-4}$</td>
<td>2.14e–4</td>
<td>43.823</td>
</tr>
<tr>
<td>''</td>
<td>6, 6, 10</td>
<td>$10^3$</td>
<td>$10^{-6}$</td>
<td>$10^{-5}$</td>
<td>2.47e–4</td>
<td>539.67</td>
</tr>
<tr>
<td>''</td>
<td>4, 4, 4, 4, 4</td>
<td>273</td>
<td>$10^{-6}$</td>
<td>$10^{-5}$</td>
<td>1.31e–3</td>
<td>3098.</td>
</tr>
<tr>
<td>''</td>
<td>3, 3, 3</td>
<td>1.70</td>
<td>$10^{-3}$</td>
<td>$10^{-1}$</td>
<td>7.93e–1</td>
<td>29.25</td>
</tr>
<tr>
<td>''</td>
<td>18, 18</td>
<td>$10^4$</td>
<td>$10^{-7}$</td>
<td>$10^{-6}$</td>
<td>3.75e–6</td>
<td>3173.</td>
</tr>
<tr>
<td>''</td>
<td>12, 7, 5</td>
<td>8.34</td>
<td>$10^{-4}$</td>
<td>$10^{-3}$</td>
<td>8.42e–3</td>
<td>4370.</td>
</tr>
<tr>
<td>Not Sqr Free</td>
<td>5, (5)$^2$</td>
<td>$10^3$</td>
<td>$10^{-5}$</td>
<td>$10^{-5}$</td>
<td>6.98e–5</td>
<td>34.28</td>
</tr>
<tr>
<td>3 variables</td>
<td>5, 5</td>
<td>$10^4$</td>
<td>$10^{-5}$</td>
<td>$10^{-5}$</td>
<td>1.72e–5</td>
<td>332.99</td>
</tr>
<tr>
<td>$f_i \in \mathbb{C}$</td>
<td>6, 6</td>
<td>$10^6$</td>
<td>$10^{-8}$</td>
<td>$10^{-7}$</td>
<td>2.97e–7</td>
<td>30.034</td>
</tr>
</tbody>
</table>
More than two variables: direct approach

- PDEs can be generalized to many variables

\[
\frac{\partial g}{\partial y_i} f - \frac{\partial h_i}{\partial x} f = 0, \forall 1 \leq i \leq k
\]

\[
\deg g \leq \deg f, \quad \deg h_i \leq \deg f, \quad \forall 1 \leq i \leq k,
\]

\[
\deg_x g \leq (\deg_x f) - 1, \quad \deg_y h_i \leq (\deg_y f) - 1, \quad \forall 1 \leq i \leq k.
\]
More than two variables: interpolation

- Our multivariate implementation together with Wen-shin Lee’s numerical *sparse* interpolation code quickly factors polynomials arising in engineering *Stewart-Gough platforms*

Polynomials were 3 variables, factor multiplicities up to 5, coefficient error $10^{-16}$, and were provided to us by Jan Verschelde
Stewart Platform Example

Drexler’s 1992 nano Stewart platform
Current Investigations

- Use *Gauss-Newton optimization* at the end to improve nearness of computed approximate answers
Current Investigations

• Use *Gauss-Newton optimization* at the end to improve nearness of computed approximate answers

• Replace SVD techniques with Structured SVD/Total Least Squares [Park et al., Chu et al.]
Current Investigations

- Use *Gauss-Newton optimization* at the end to improve nearness of computed approximate answers

- Replace SVD techniques with Structured SVD/Total Least Squares [Park et al., Chu et al.]

- More generally, use *blackbox matrix* SVD algorithms

  \[ R_u(f) \cdot v \] costs 4 polynomial multiplications

  Should make very large problems possible
Current Investigations

• Use Gauss-Newton optimization at the end to improve nearness of computed approximate answers

• Replace SVD techniques with Structured SVD/Total Least Squares [Park et al., Chu et al.]

• More generally, use blackbox matrix SVD algorithms

  \[ \text{R}u(f) \cdot v \text{ costs 4 polynomial multiplications} \]

  Should make very large problems possible

• Also need sparse interpolation for “very noisy” inputs to handle sparse multivariate problems
Code + Benchmarks at:

http://www.mmrc.iss.ac.cn/~lzhi/Research/appfac.html

or

http://www.kaltofen.us

(click on “Software”)