ISSAC'04

Approximate Factorization of Multivariate Polynomials via Differential Equations

Shuhong Gao	Erich Kaltofen, John May	Zhengfeng Yang, Lihong Zhi
Clemson	NCSU	AMSS

http://www.math.ncsu.edu/˜kaltofen/software/appfac/

Approximate Factorization Problem [Kaltofen '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.

Approximate Factorization Problem [Kaltofen '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: $\text{mdeg } f = (\deg_x f, \deg_y f)$

Approximate Factorization Problem [Kaltofen '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: $\operatorname{mdeg} f = (\deg_x f, \deg_y f)$

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

State of the Approximate Factorization

- No polynomial time algorithm
- 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

- 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 $\min \| f - \tilde{f} \|$
  [Kaltofen and May ISSAC 2003]

Our Results

- A new practical algorithm to compute approximate multivariate GCDs

Our 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 $\bar{f}_1, \bar{f}_2$ s.t. $\| f_1 f_2 - \bar{f}_1 \bar{f}_2 \| \approx \| f_{\text{noise}} \|$

  even for large noise: $\| f_{\text{noise}} \| / \| f \| \geq 10^{-3}$

Maple Demonstration

Ruppert's Theorem

$f \in \mathbb{K}[x,y]$, 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,

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

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

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,

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

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

PDE $\rightsquigarrow$ 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: # linearly indep. solutions to the PDE = # factors of f

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

Gao's PDE based Factorizer

Change degree bound: $\mathrm{mdeg}\, g \le (m-1, n), \mathrm{mdeg}\, h \le (m, n-1)$

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

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

Let

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

Any solution $g \in G$ gives a factorization:

$$f = \prod_{\lambda \in \mathbb{C}} \mathrm{gcd}(f, g - \lambda f_x)$$

with high probability $\exists$ distinct λ_i s.t. $f_i = \mathrm{gcd}(f, g - \lambda_i f_x)$
f_i's distinct irreducible factors of f

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 $f_i = \gcd(f, g - \lambda_i f_x)$ in $\mathbb{K}(\lambda_i)$.

In exact arithmetic the extention 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 $\mathrm{Rup}(f)$ be the matrix from Gao's algorithm
Recall:

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

$\mathrm{Rup}(f)$ has nullity r if
$\sigma_m \geq \ldots \geq \sigma_{r+1} \neq 0$ and $\sigma_r = \ldots = \sigma_1 = 0$.

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

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

Find a "best" ε from the largest gap
choose $\varepsilon = \sigma_r$ s.t. σ_{r+1}/σ_r is maximal

Determining the Number of Approximate Factors

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

Recall:

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

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

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 $\mathrm{Rup}(f)$, determine r, its approximate nullity, and choose $g = \sum a_i g_i$, 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 λ_i compute the approximate GCD $f_i = \gcd(f, g - \lambda_i f)$ and find an optimal scaling: $\min_c \| f - c \prod_{i=1}^{r} f_i \|$

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 $\gcd(f, f_x)$ 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. $<$ 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 $\gcd(f, f_x)$ 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

Example	$\mathrm{tdeg}(f_i)$	$\frac{\sigma_{r+1}}{\sigma_r}$	$\frac{\sigma_r}{\|R(f)\|_2}$	coeff. error	backward error	*time*(*sec*)
Nagasaka'02	$2,3$	11	10^{-3}	10^{-2}	1.08e–2	14.631
Kaltofen'00	$2,2$	10^9	10^{-10}	10^{-4}	1.02e–9	13.009
Sasaki'01	$5,5$	10^9	10^{-10}	10^{-13}	8.30e–10	5.258
Sasaki'01	$10,10$	10^5	10^{-6}	10^{-7}	1.05e–6	85.96
Corless et al'01	$7,8$	10^7	10^{-8}	10^{-9}	1.41e–8	19.628
Corless et al'02	$3,3,3$	10^8	10^{-10}	0	1.29e–9	9.234
Zeng'04	$(5)^3,3,(2)^4$	10^7	10^{-9}	10^{-10}	2.09e–7	73.52

Table of Benchmarks

Example	$\mathrm{tdeg}(f_i)$	$\frac{\sigma_{r+1}}{\sigma_r}$	$\frac{\sigma_r}{\|R(f)\|_2}$	coeff. error	backward error	*time(sec)*
Random ($f_i \in \mathbb{Z}$)	$9,7$	486	10^{-4}	10^{-4}	2.14e–4	43.823
"	$6,6,10$	10^3	10^{-6}	10^{-5}	2.47e–4	539.67
"	$4,4,4,4,4$	273	10^{-6}	10^{-5}	1.31e–3	3098.
"	$3,3,3$	1.70	10^{-3}	10^{-1}	7.93e–1	29.25
"	$18,18$	10^4	10^{-7}	10^{-6}	3.75e–6	3173.
"	$12,7,5$	8.34	10^{-4}	10^{-3}	8.42e–3	4370.
Not Sqr Free	$5,(5)^2$	10^3	10^{-5}	10^{-5}	6.98e–5	34.28
3 variables	$5,5$	10^4	10^{-5}	10^{-5}	1.72e–5	332.99
$f_i \in \mathbb{C}$	$6,6$	10^6	10^{-8}	10^{-7}	2.97e–7	30.034

Table of Benchmarks

Example	$\mathrm{tdeg}(f_i)$	$\frac{\sigma_{r+1}}{\sigma_r}$	$\frac{\sigma_r}{\lVert R(f)\rVert_2}$	coeff. error	backward error	*time*(*sec*)
Random ($f_i \in \mathbb{Z}$)	$9, 7$	486	10^{-4}	10^{-4}	2.14e–4	43.823
"	$6, 6, 10$	10^3	10^{-6}	10^{-5}	2.47e–4	539.67
"	$4, 4, 4, 4, 4$	273	10^{-6}	10^{-5}	1.31e–3	3098.
"	$3, 3, 3$	1.70	10^{-3}	10^{-1}	7.93e–1	29.25
"	$18, 18$	10^4	10^{-7}	10^{-6}	3.75e–6	3173.
"	$\mathbf{12}, \mathbf{7}, \mathbf{5}$	8.34	10^{-4}	$\mathbf{10^{-3}}$	**8.42e–3**	4370.
Not Sqr Free	$5, (5)^2$	10^3	10^{-5}	10^{-5}	6.98e–5	34.28
3 variables	$5, 5$	10^4	10^{-5}	10^{-5}	1.72e–5	332.99
$f_i \in \mathbb{C}$	$6, 6$	10^6	10^{-8}	10^{-7}	2.97e–7	30.034

More than two variables

- Everything can be generalized to many variables directly (w/o projecting to 2-variables)

More than two variables

- Everything can be generalized to many variables directly (w/o projecting to 2-variables)

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

  Polynomials were 3 variables, factor mult. up to 5, coefficient error 10^{-16}, and were provided by to us Jan Verschelde

Future Work

- Factorization algorithm can be modified to use only iterative blackbox methods to compute singular values/vectors

  $\mathrm{Rup}(f) \cdot \mathbf{v}$ costs 4 polynomial multiplications

  Should make very large problems possible

Future Work

- Factorization algorithm can be modified to use only iterative blackbox methods to compute singular values/vectors

  $\mathrm{Rup}(f) \cdot \mathbf{v}$ costs 4 polynomial multiplications

  Should make very large problems possible

- Replace SVD techniques with Structured SVD/Total least squares

Future Work

- Factorization algorithm can be modified to use only iterative blackbox methods to compute singular values/vectors

  $\mathrm{Rup}(f) \cdot \mathbf{v}$ costs 4 polynomial multiplications

  Should make very large problems possible
- Replace SVD techniques with Structured SVD/Total least squares
- Find robust "noisy" sparse interpolation to handle sparse multivariate problems

Code + Benchmarks at:

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

or

`http://www.math.ncsu.edu/~kaltofen/`

click on “Software”