Dagstuhl 2004

Approximate Factorization of Multivariate Polynomials via Differential Equations

Erich Kaltofen

North Carolina State University

Webpage: Google→kaltofen

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

Approximate Factorization Problem [Kaltofen '94]

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

Approximate Factorization Problem [Kaltofen '94]

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

Problem depends on choice of norm $\|\cdot\|$, **and degree**:

For $f = x^2 + y^2 - 1$, the 2-norm, and total degree:

$$\tilde{f} = (x-1)(x+1), \; \|f - \tilde{f}\|_2 = 1.$$

Approximate Factorization Problem [Kaltofen '94]

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

Problem depends on choice of norm $\|\cdot\|$, **and degree**:

For rectangular degrees we get closer to $f = x^2 + y^2 - 1$:

$$\hat{f} = (0.4906834y^2 + 0.8491482x - 0.9073464)(x + 1.214778)$$

$$= 0.596072y^2 + 0.849148x^2 + 0.490683xy^2 + 0.124180x - 1.102225,$$

$$\|f - \hat{f}\|_2 \approx 0.6727223.$$

Approximate Factorization Problem [Kaltofen '94]

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

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

- There are currently no polynomial time algorithms to find the closest polynomial which factors.
  Best today: Constant degree factors [Hitz, Kaltofen, Lakshman ISSAC '99]

State of the Approximate Factorization

- There are currently no polynomial time algorithms to find the closest polynomial which factors.
  Best today: Constant degree factors [Hitz, Kaltofen, Lakshman ISSAC '99]
- There are several algorithms and heuristics to find factorizable polynomials that are close to "nearly factorizable" polynomials.

State of the Approximate Factorization

- There are currently no polynomial time algorithms to find the closest polynomial which factors.
  Best today: Constant degree factors [Hitz, Kaltofen, Lakshman ISSAC ’99]
- There are several algorithms and heuristics to find factorizable polynomials that are close to “nearly factorizable” polynomials.
- There is a method for finding lower bounds on the distance to the nearest polynomial which factors [Kaltofen and May ISSAC 2003].

State of the Approximate Factorization

- There are currently no polynomial time algorithms to find the closest polynomial which factors.
  Best today: Constant degree factors [Hitz, Kaltofen, Lakshman ISSAC '99]
- There are several algorithms and heuristics to find factorizable polynomials that are close to "nearly factorizable" polynomials.
- There is a method for finding lower bounds on the distance to the nearest polynomial which factors [Kaltofen and May ISSAC 2003].
- Improved lower bounds together with improved approximate factorizers may yield the nearest polynomial which factors similarly to TSP problems.

Wolfgang M. Ruppert's Theorem

$f \in \mathbb{K}[x,y]$, $\deg f = (\deg_x f, \deg_y f) = (m,n)$.

$\mathbb{K}$ is a field, algebraically closed, and characteristic 0.

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

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

$$\deg g \leq (m-2,n)\,,\ \deg h \leq (m,n-1).$$

Wolfgang M. Ruppert's Theorem

$f \in \mathbb{K}[x,y]$, $\deg f = (\deg_x f, \deg_y f) = (m,n)$.

$\mathbb{K}$ is a field, algebraically closed, and characteristic 0.

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

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

$$\deg g \le (m-2,n)\,,\ \deg h \le (m,n-1).$$

Bounds on the degrees of g and h eliminate the solution

$$g = \frac{\partial f}{\partial x}\,,\ h = \frac{\partial f}{\partial y}.$$

Wolfgang M. Ruppert's Theorem

$f \in \mathbb{K}[x,y]$, $\deg f = (\deg_x f, \deg_y f) = (m,n)$.

$\mathbb{K}$ is a field, algebraically closed, and characteristic 0.

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

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

$$\deg g \leq (m-2,n)\,,\ \deg h \leq (m,n-1).$$

The PDE leads to a set of equations linear in the coefficients of g and h.

Gao's Factorizer based on Ruppert's Theorem

Change the degree bound: $\deg g \leq (m-1, n)$

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

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

Gao's Factorizer based on Ruppert's Theorem

Change the degree bound: $\deg g \leq (m-1, n)$
linearly indep. solutions to the PDE = # factors of f
Require square-freeness: $\mathrm{GCD}(f, \frac{\partial f}{\partial x}) = 1$
If $f = f_1 \cdots f_r$, then let

$$E_i = \frac{f}{f_i}\frac{\partial f_i}{\partial x} \in \mathbb{C}[x,y]$$

and

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

Any solution $g \in G$ satisfies $g = \sum_i^r \lambda_i E_i$. If the λ_i's are distinct then

$$f_i = \gcd(f, g - \lambda_i f_x).$$

Gao's Factorizer based on Ruppert's Theorem

Algorithm

1. Find a basis for the linear space G, and choose a random element $g \in G$.
2. Compute the $r \times r$ matrix A_g using g and the basis of G.
3. Factor $\mathrm{CharPoly}(A_g) = \prod_i \phi_i$ over $\mathbb{Q}$.
   $\mathbb{Q}[z]/\phi_i(z)$ are the extensions in which the factors lie.
4. Compute (the equivalence class of) a factor
   $f_i = \gcd(f, g - \alpha_i f_x)$ with $\alpha_i \equiv z \in \mathbb{Q}[z]/\phi_i(z)$.

Adapting to the Approximate Case

The following problems must be solved in order to create an approximate factorizer from Gao's algorithm:

1. Computing approximate GCDs of bivariate polynomials;
2. Determining the numerical dimension of the solution space of the PDE, and computing approximate solutions to the PDE;
3. Computing a matrix A_g which has no clusters of eigenvalues.

The Generalized Sylvester Matrix

A pair $g, h \in \mathbb{K}[x,y]$ has a GCD of degree at least k iff there are non-zero solutions $u, v \in \mathbb{K}[x,y]$ to:

$$ug + vh = 0, \ \deg u \leq \deg(h) - k, \ \deg v \leq \deg(g) - k.$$

This system is linear in the coefficients of u and v.
The matrix of this system, $\mathrm{Syl}_k(g,h)$ is the bivariate generalization of the Sylvester matrix of g and h when $k = 1$.

Determining the Degree of the GCD

In exact arithmetic: the degree of $\gcd(g,h)$ can be easily determined from the rank deficiency of $\mathrm{Syl}_1(g,h)$.

Numerically, rank deficiency is determined by the singular values σ_i of $\mathrm{Syl}_1(g,h) = U\Sigma V$.

$\mathrm{Syl}_1(g,h)$ is **exactly** rank p if $\sigma_1 \geq \cdots \geq \sigma_p > 0$ and $\sigma_{p+1} = \cdots = \sigma_m = 0$.

Determining the Degree of the GCD

In exact arithmetic: the degree of $\gcd(g,h)$ can be easily determined from the rank deficiency of $\mathrm{Syl}_1(g,h)$.

Numerically, rank deficiency is determined by the singular values σ_i of $\mathrm{Syl}_1(g,h) = U\Sigma V$.

$\mathrm{Syl}_1(g,h)$ is **exactly** rank p if $\sigma_1 \geq \cdots \geq \sigma_p > 0$ and $\sigma_{p+1} = \cdots = \sigma_m = 0$.

We call $\mathrm{Syl}_1(g,h)$ **numerical** rank p if for a chosen tolerance ε:

$$\sigma_1 \geq \cdots \geq \sigma_p > \varepsilon \geq \sigma_{p+1} \geq \cdots \geq \sigma_m.$$

If ε is not given, we will find an ε from the largest gap. That is $\varepsilon = \sigma_{p+1}$ so that σ_p/σ_{p+1} is maximal.

Computing the Exact GCD

Let $k \leq \deg(\gcd(g,h))$, then the smallest degree solution to

$$ug + vh = 0, \ \deg u \leq \deg(h) - k, \ \deg v \leq \deg(g) - k$$

is $u = h_1 = h/\gcd(g,h)$, $v = g_1 = -g/\gcd(g,h)$.

Computing the Exact GCD

Let $k \leq \deg(\gcd(g,h))$, then the smallest degree solution to

$$ug + vh = 0, \ \deg u \leq \deg(h) - k, \ \deg v \leq \deg(g) - k$$

is $u = h_1 = h/\gcd(g,h)$, $v = g_1 = -g/\gcd(g,h)$.

We can determine $k = \deg(\gcd(g,h))$ from the rank of $\mathrm{Syl}_1(g,h)$. For that k,

$$\mathrm{Nullspace}(\mathrm{Syl}_k(g,h)) = \mathrm{Span}_{\mathbb{K}}\{[h_1, g_1]\}.$$

Computing the Exact GCD

Let $k \leq \deg(\gcd(g,h))$, then the smallest degree solution to

$$ug + vh = 0, \ \deg u \leq \deg(h) - k, \ \deg v \leq \deg(g) - k$$

is $u = h_1 = h / \gcd(g,h)$, $v = g_1 = -g / \gcd(g,h)$.

We can determine $k = \deg(\gcd(g,h))$ from the rank of $\mathrm{Syl}_1(g,h)$.
For that k,

$$\mathrm{Nullspace}(\mathrm{Syl}_k(g,h)) = \mathrm{Span}_{\mathbb{K}}\{[h_1, g_1]\}.$$

The GCD, d, is found by:
1. solving for null-vector $[h_1, g_1]$ of $\mathrm{Syl}_k(g,h)$
2. dividing: $d = h/h_1 = g/g_1$.

Computing the Approximate GCD

Input: g and h relatively prime
Output: $d \notin \mathbb{C}$ approx. GCD of g and h, and $\varepsilon > 0$ tolerance

1. Form $\mathrm{Syl}_1(g,h)$
2. Determine k, the degree of the approximate GCD of g and h by finding the largest gap in the singular values of S and inferring the degree from the numerical rank
3. Set $\varepsilon = \sigma_{p+1}$
4. Form $\mathrm{Syl}_k(g,h)$ [has approximate rank 1]
5. Singular vector corresponding the smallest singular value of $\mathrm{Syl}_k(g,h)$ is the approximate null-vector $[u,v]$ [can compute with an iterative method]
6. Find a d that minimizes $\|h - d\,u\|_2^2 + \|g + d\,v\|_2^2$, using least squares [Approximate Division]

Determining the Number of Approximate Factors

Denote the matrix from Gao's algorithm $\mathrm{Rup}(f)$. Recall

$$\#\text{ of factors of } f = \mathrm{Dim}(\mathrm{Nullspace}(\mathrm{Rup}(f))).$$

If f is irreducible, find the number of approximate factors with the approximate rank of $\mathrm{Rup}(f)$.

Determining the Number of Approximate Factors

Denote the matrix from Gao's algorithm $\mathrm{Rup}(f)$. Recall

$$\#\text{ of factors of } f = \mathrm{Dim}(\mathrm{Nullspace}(\mathrm{Rup}(f))).$$

If f is irreducible, find the number of approximate factors with the approximate rank of $\mathrm{Rup}(f)$.

Like approximate GCD, approximate rank of $\mathrm{Rup}(f)$ is determined by the largest gap in the singular values.
Recall:

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

An approximate basis for G can be found from the singular vectors corresponding to the smallest singular values of $\mathrm{Rup}(f)$.

Determining the Number of Approximate Factors

Denote the matrix from Gao's algorithm $\mathrm{Rup}(f)$. Recall

$$\# \text{ of factors of } f = \mathrm{Dim}(\mathrm{Nullspace}(\mathrm{Rup}(f))).$$

If f is irreducible, find the number of approximate factors with the approximate rank of $\mathrm{Rup}(f)$.

Note:

$\mathrm{Rup}(f)$ always has at least one null-vector, $[f_x, f_y]$, so $r \geq 1$. Since we are looking for approximate factors we will choose the largest gap with $r > 1$.

The Matrix A_g

Let $\{g_i\}_{i=1}^{r}$ be a basis for G.
Let $g = \sum s_i g_i$ where $s_i \in S \subset \mathbb{C}$ are chosen randomly, and independently.
$A_g = [a_{i,j}]$ is the unique $r \times r$ matrix such that

$$g\, g_i \equiv a_{i,j}\, g_j\, f_x \pmod{f} \ \text{ in } \mathbb{C}(y)[x].$$

The Matrix A_g

Let $\{g_i\}_{i=1}^r$ be a basis for G.
Let $g = \sum s_i g_i$ where $s_i \in S \subset \mathbb{C}$ are chosen randomly, and independently.
$A_g = [a_{i,j}]$ is the unique $r \times r$ matrix such that

$$g\, g_i \equiv a_{i,j}\, g_j\, f_x (\mathrm{mod}\ f) \ \text{ in } \mathbb{C}(y)[x].$$

Then:

$$f = \prod_{\lambda \in \text{Eigenvalues}(A_g)} \gcd(f, g - \lambda f)$$

is a complete factorization of f over $\mathbb{C}$ with probability at least $1 - r(r-1)/(2|S|)$.

Approximate Factorization

Input: $f \in \mathbb{Q}[x,y]$ absolutely irreducible, approximately square-free
Output: $f_1,\ldots,f_r$ approximate factors of f

1. Compute the SVD of $\mathrm{Rup}(f)$, and determine r, its rank deficiency.
2. Set $g_1,\ldots,g_r$ equal to the last r singular vectors of $\mathrm{Rup}(f)$.
3. Choose $s_i \in S \subset \mathbb{C}$ randomly and independently, set $g = \sum s_i g_i$
4. Compute A_g, and $\lambda_1,\ldots,\lambda_r$ its eigenvalues.
   If any $|\lambda_i - \lambda_j|$ is too small compute a new random g.
5. For each λ_i compute the approximate GCD $f_i = \mathrm{gcd}(f, g - \lambda_i f)$.

Notes on the non-Square-free Case

Let δ be the r^{th} smallest singular value of $\mathrm{Rup}(f)$.
Let ε be the tolerance of $\gcd(f, f_x)$.
If $\varepsilon \leq \delta$, f is not approximately square-free.

One way to handle the non-square-free case:
Compute the approximate quotient $\bar{f}$ of f and f_x and factor the approximately square-free kernel $\bar{f}$.

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

$$\gcd(f_i, \partial^k f / \partial x^k).$$

Table of Tests

Ex.	$\deg(f_i)$	$\frac{\sigma_{r+1}}{\sigma_r}$	σ_r	coeff. err.	backward err.'s	T_1(s)	T(s)
Nagasaka '02	$2,3$	11	10^{-3}	10^{-1}	0.13×10^{-1}	0.031	2.406
Sasaki '01	$5,5$	10^{9}	10^{-8}	10^{-12}	0.11×10^{-8}	0.656	5.156
Sasaki '01	$10,10$	10^{5}	10^{-6}	10^{-5}	0.11×10^{-5}	22.7	115.1
Corless etal '01	$7,8$	10^{8}	10^{-8}	10^{-5}	0.14×10^{-7}	5.0	24.83
Corless etal '02	$3,3,3$	10^{8}	10^{-10}	0	0.11×10^{-8}	0.312	9.422
Random	$6,6,10$	10^{5}	10^{-8}	10^{-5}	0.62×10^{-6}	40.83	558.3
"	$15,7,2$	371	10^{-4}	10^{-2}	0.33×10^{-3}	67.17	2801
"	$4,4,4,4,4$	10^{6}	10^{-7}	10^{-5}	0.50×10^{-7}	23.98	978.4
"	$3,3,3$	16	10^{-3}	10^{-1}	0.156	0.359	41.97
"	$8,8$	173	10^{-3}	10^{-2}	0.65×10^{-3}	7.266	66.56
"	$12,7,5$	529	10^{-5}	10^{-2}	0.20×10^{-3}	66.69	1461
"	$12,7,5$	53	10^{-4}	10^{-1}	0.28×10^{-2}	66.75	1534
"	$8,8$	9	10^{-2}	10^{-1}	0.14×10^{-1}	7.157	153.9
"	$18,18$	10^{6}	10^{-8}	10^{-6}	0.64×10^{-6}	725.2	4979

Fin