Approximate Factorization of Complex Multivariate Polynomials My Lessons Learned

Erich Kaltofen

Massachusetts Institute of Technology

google->kaltofen

Gröbner and me

1974–1977 I was an undergraduate student of Bruno Buchberger (6 “Scheine” [individual grades])

Gröbner and me

1974–1977 I was an undergraduate student of Bruno Buchberger (6 “Scheine” [individual grades])

1982–1983 Promoted Buchberger’s algorithm at the University of Toronto with somewhat delayed success

Gröbner and me

1974–1977 I was an undergraduate student of Bruno Buchberger (6 “Scheine” [individual grades])

1982–1983 Promoted Buchberger’s algorithm at the University of Toronto with somewhat delayed success

1987–1990 Supervised my first Ph.D. student Lakshman Y. N. “On the Complexity of Computing Gröbner Bases for Zero Dimensional Polynomial Ideals”

Gröbner and me

1974–1977 I was an undergraduate student of Bruno Buchberger (6 “Scheine” [individual grades])

1982–1983 Promoted Buchberger’s algorithm at the University of Toronto with somewhat delayed success

1987–1990 Supervised my first Ph.D. student Lakshman Y. N. “On the Complexity of Computing Gröbner Bases for Zero Dimensional Polynomial Ideals”
Greetings from Lakshman

My 1998 Challenge Problem 5

Buchberger's algorithm [1967]
S-polynomial construction and reduction correspond to row-reduction in comparable matrices

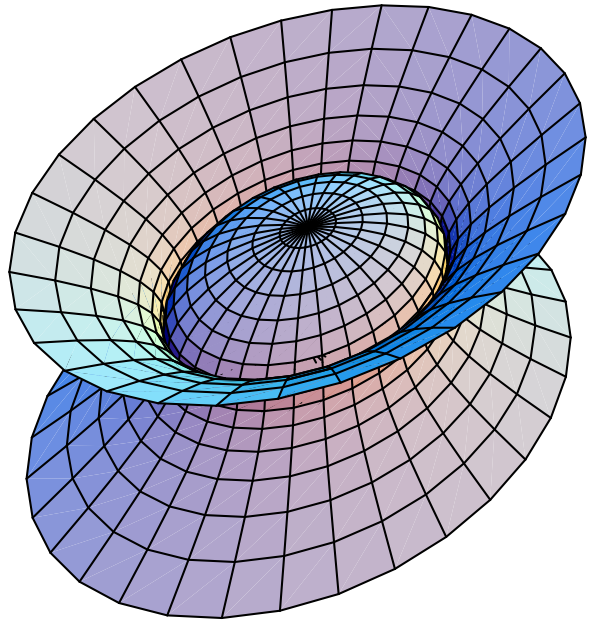
Faugère's [1997] method: use sparse "symbolic" LU matrix decomposition for performing these row reductions.

Problem 5
Compute Gröbner bases approximately by iterative methods for solving systems, such as Gauss&Seidel, conjugate gradient, Newton,...

A solution plugs into numerical software and computes some bases faster than the exact approach; the structure of the bases may be determined, e.g., by modular arithmetic

Factorization of “noisy” polynomials over the complex numbers [my 1998 Challenge Problem 1]

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



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

$$81x^4 + 16y^4 - 648.003z^4 + 72x^2y^2 + .002x^2z^2 + .001y^2z^2$$
$$- 648x^2 - 288y^2 - .007z^2 + 1296 = 0$$

Conclusion on my exact algorithm [JSC 1(1)'85]

"*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.*"

Public abstract of my 1989 NSF proposal

"*Motivated by geometric design, we propose to develop an algorithm for computing the irreducible components of an algebraic curve or surface given by its polynomial defining implicit equation with floating point coefficients. Attacked by computer algebra methods, the problem is to computing the* ***approximate factorization*** *of a rational bivariate polynomial.*"

The Approximate Factorization Problem [Kaltofen '89; Sasaki '89]

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 [Kaltofen '89; Sasaki '89]

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: $\mathrm{mdeg}\ f = (\deg_x f, \deg_y f)$

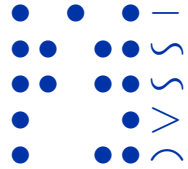
The Approximate Factorization Problem [Kaltofen '89; Sasaki '89]

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.

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

$$\|(1+\delta x)f - f\| = \|\delta x f\| = \delta \|f\| < \varepsilon$$

My NSF Workshop on *Integrated Symbolic–Numeric Computing* at ISSAC 1992



8:15am B. Donald, “Geometric algorithms, rational rotations, and computational topology.”

9:30am R. Liska, “Symbolic computation techniques in the finite difference method for solving PDEs.”

10:30am A. Letichevsky, “Rule-based programming in APS.”

11:30am L. Wauthier, “An example of computer algebra application in artificial satellite theory.”

1:30pm P. Moore, “A symbolic interface to adaptive algorithms for parabolic systems.”

2:30pm J. Renegar, “Condition numbers and complexity theory.”

3:45pm K. Broughan, “Senac: a software environment for numeric and algebraic computation.”

What’s in a Name?

- Integrated Symbolic-Numeric Computing [ISSAC 1992]

What's in a Name?

- Integrated Symbolic-Numeric Computing [ISSAC 1992]
- Symbolic-Numeric Algebra for Polynomials [SNAP'96, JSC special issue]

What's in a Name?

- Integrated Symbolic-Numeric Computing [ISSAC 1992]
- Symbolic-Numeric Algebra for Polynomials [SNAP'96, JSC special issue]
- Symbolic and Numerical Scientific Computation [SNSC'99]

What's in a Name?

- Integrated Symbolic-Numeric Computing [ISSAC 1992]
- Symbolic-Numeric Algebra for Polynomials [SNAP'96, JSC special issue]
- Symbolic and Numerical Scientific Computation [SNSC'99]
- Hybrid Symbolic-Numeric Computation [Computer Algebra Handbook 2002]

What's in a Name?

- Integrated Symbolic-Numeric Computing [ISSAC 1992]
- Symbolic-Numeric Algebra for Polynomials [SNAP'96, JSC special issue]
- Symbolic and Numerical Scientific Computation [SNSC'99]
- Hybrid Symbolic-Numeric Computation [Computer Algebra Handbook 2002]
- Symbolic-Numeric Computation [SNC 2005]

What's in a Name?

- Integrated Symbolic-Numeric Computing [ISSAC 1992]
- Symbolic-Numeric Algebra for Polynomials [SNAP'96, JSC special issue]
- Symbolic and Numerical Scientific Computation [SNSC'99]
- Hybrid Symbolic-Numeric Computation [Computer Algebra Handbook 2002]
- Symbolic-Numeric Computation [SNC 2005]
- Approximate Algebraic Computation [AAC@ACA'05]

What's in a Name?

- Integrated Symbolic-Numeric Computing [ISSAC 1992]
- Symbolic-Numeric Algebra for Polynomials [SNAP'96, JSC special issue]
- Symbolic and Numerical Scientific Computation [SNSC'99]
- Hybrid Symbolic-Numeric Computation [Computer Algebra Handbook 2002]
- Symbolic-Numeric Computation [SNC 2005]
- Approximate Algebraic Computation [AAC@ACA'05]
- Approximate Commutative Algebra [ApCoA'06]

What's in a Name?

- Integrated Symbolic-Numeric Computing [ISSAC 1992]
- Symbolic-Numeric Algebra for Polynomials [SNAP'96, JSC special issue]
- Symbolic and Numerical Scientific Computation [SNSC'99]
- Hybrid Symbolic-Numeric Computation [Computer Algebra Handbook 2002]
- Symbolic-Numeric Computation [SNC 2005]
- Approximate Algebraic Computation [AAC@ACA'05]
- Approximate Commutative Algebra [ApCoA'06]

- Applicable Algebra and Error Correcting Codes [AAECC]

What's in a Name?

- Integrated Symbolic-Numeric Computing [ISSAC 1992]
- Symbolic-Numeric Algebra for Polynomials [SNAP'96, JSC special issue]
- Symbolic and Numerical Scientific Computation [SNSC'99]
- Hybrid Symbolic-Numeric Computation [Computer Algebra Handbook 2002]
- Symbolic-Numeric Computation [SNC 2005]
- Approximate Algebraic Computation [AAC@ACA'05]
- Approximate Commutative Algebra [ApCoA'06]

- Applicable Algebra and Error Correcting Codes [AAECC]
- Journal of Symbolic Computation

Previous Work on Approximate Factorization

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

Previous Work on 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,...]

Previous Work on 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 $\min \| f - \tilde{f} \|$ (“irreducibility radius”)
  [Kaltofen and May ISSAC ’03; Nagasaka CASC ’04, ’05]

Our ISSAC'04, ASCM'05, 2005, 2006 Results [joint with John May, Zhengfeng Yang, Lihong Zhi (and Shuhong Gao ISSAC'04)]

- Several practical algorithms to compute approximate multivariate GCDs

Our ISSAC'04, ASCM'05, 2005, 2006 Results [joint with John May, Zhengfeng Yang, Lihong Zhi (and Shuhong Gao ISSAC'04)]

- Several practical algorithms to compute approximate multivariate GCDs

- Practical algorithms to find the factorization of a nearby factorizable polynomial given any f

  especially "noisy" f:
  Given $f = \underline{f_1 \cdots f_s} + f_{\text{noise}}$,
  we find $\overline{f_1}, \ldots \overline{f_s}$ s.t. $\| f_1 \cdots f_s - \overline{f_1} \cdots \overline{f_s} \| \approx \| f_{\text{noise}} \|$

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

Maple Demonstration

Lesson #1: The solution of some problems takes time

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: $\operatorname{mdeg} g \leq (m-1,n), \operatorname{mdeg} h \leq (m,n-1)$

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 $g \in G$ satisfies $g = \sum_{i=1}^{r} \lambda_i \frac{\partial f_i}{\partial x} \frac{f}{f_i}$ with $\lambda_i \in \mathbb{C}$, so

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

(f_i the distinct irreducible factors of f)

With high probability $\exists$ distinct λ_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: $f_1, \ldots, f_s \in \mathbb{C}[x,y]$

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 Bivariate 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. Randomize s.t. the polynomial E_g has no clusters of roots;
4. Compute approximate squarefree factorization.

Approximate Factorization

Input: $f \in \mathbb{C}[x,y]$ abs. irreducible, approx. square-free
Output: $f_1, \ldots, f_s$ approx. factors of f.

1. Compute the SVD of $\mathrm{Rup}(f)$, determine s, its approximate nullity, and choose $g = \sum a_i g_i$, a random linear combination of the last s right singular vectors

2. Compute E_g and its roots via an eigenvalue computation

3. For each λ_i compute the approximate GCD $f_i = \gcd(f, g - \lambda_i f)$

4. Optimize $\| f - f_1 \cdots f_s \|_2$ via Gauss-Newton iterative refinement.

When to optimize?

The Matrix A_g

Let $\{g_i\}_{i=1}^{s}$ be a basis for G, $g \in_R G$
Let $A_g = (a_{i,j})$ be the $s \times s$ matrix s.t.

$$g g_i \equiv \sum_{j=1}^{s} a_{i,j} g_j \frac{\partial f}{\partial x} (\text{mod } f) \text{ in } \mathbb{C}(y)[x].$$

The Matrix A_g

Let $\{g_i\}_{i=1}^{s}$ be a basis for G, $g \in_R G$
Let $A_g = (a_{i,j})$ be the $s \times s$ matrix s.t.

$$g g_i \equiv \sum_{j=1}^{s} a_{i,j} g_j \frac{\partial f}{\partial x} (\text{mod } f) \text{ in } \mathbb{C}(y)[x].$$

Then, $\text{CharPoly}(A_g) = E_g$
With high probability the eigenvalues of A_g are distinct so:

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

is a complete factorization of f over $\mathbb{C}$

Generalization to Several Variables

- PDEs can be generalized to many variables

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

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

$$\deg_x g \leq (\deg_x f) - 1, \quad \deg_{y_i} h_i \leq (\deg_{y_i} f) - 1, \forall 1 \leq i \leq k.$$

Approximate GCD Problem

Given two polynomials $f, g \in \mathbb{C}[y_1, y_2, \ldots, y_r]$, with total degree $\deg(f) = m$ and $\deg(g) = n$. For a positive integer k with $k \leq \min(m, n)$, we wish to compute Δf, $\Delta g \in \mathbb{C}[y_1, y_2, \ldots, y_r]$ such that $\deg(\Delta f) \leq m$, $\deg(\Delta g) \leq n$, and

- $\deg(\gcd(f + \Delta f, g + \Delta g)) \geq k$,
- $\|\Delta f\|_2^2 + \|\Delta g\|_2^2$ is minimized.

Approximate GCD Problem

Given two polynomials $f, g \in \mathbb{C}[y_1, y_2, \ldots, y_r]$, with total degree $\deg(f) = m$ and $\deg(g) = n$. For a positive integer k with $k \leq \min(m, n)$, we wish to compute Δf, $\Delta g \in \mathbb{C}[y_1, y_2, \ldots, y_r]$ such that $\deg(\Delta f) \leq m$, $\deg(\Delta g) \leq n$, and

- $\deg(\gcd(f + \Delta f, g + \Delta g)) \geq k$,
- $\|\Delta f\|_2^2 + \|\Delta g\|_2^2$ is minimized.

REMARK: The problem has a globally minimal real or complex solution.

The ε-GCD Problem

Problem formulation:
find highest degree approximate GCD within distance ε.

The ε-GCD Problem

Problem formulation:
find highest degree approximate GCD within distance ε.

Should solve our problem by binary search/bisection on distance.
But for certain ε the ε-GCD problem is “ill-posed.”

Lesson #2: Be careful about what numerical algorithms claim they can do.

Current methods use global numerical optimization techniques.

- Gauss-Newton and Lagrangian multipliers

Current methods use global numerical optimization techniques.

- Gauss-Newton and Lagrangian multipliers
- STLN [Park, Zhang, and Rosen ’99: Zhi’s talk]

Current methods use global numerical optimization techniques.

- Gauss-Newton and Lagrangian multipliers
- STLN [Park, Zhang, and Rosen ’99: Zhi’s talk]
- STLS [Lemmerling, Mastronardi, and Van Huffel 2000]

Current methods use global numerical optimization techniques.

- Gauss-Newton and Lagrangian multipliers
- STLN [Park, Zhang, and Rosen ’99: Zhi’s talk]
- STLS [Lemmerling, Mastronardi, and Van Huffel 2000]
- CTLS and Riemannian SVD [Mastronardi, Lemmerling, and Van Huffel 2000]

Current methods use global numerical optimization techniques.

- Gauss-Newton and Lagrangian multipliers
- STLN [Park, Zhang, and Rosen ’99: Zhi’s talk]
- STLS [Lemmerling, Mastronardi, and Van Huffel 2000]
- CTLS and Riemannian SVD [Mastronardi, Lemmerling, and Van Huffel 2000]
- Structured condition numbers [Rump 2003]

Lesson #3: a. Contemporary numerical analysis is amazing.

b. Symbolic problems are difficult cases.

c. Methods must be adapted.

The two disciplines are converging.

Lesson #4: Approximate symbolic problems can be computationally more complex than exact ones.

Floating point input data may slow us down.

End or Handbook section [Corless, Kaltofen, Watt]

The challenge of hybrid symbolic numeric algorithms is to explore the effects of imprecision, discontinuity, and ***algorithmic complexity*** *by applying mathematical optimization, perturbation theory, and inexact arithmetic and other tools in order to solve mathematical problems that today are not solvable by numerical or symbolic methods alone.*

End or Handbook section [Corless, Kaltofen, Watt]

The challenge of hybrid symbolic numeric algorithms is to explore the effects of imprecision, discontinuity, and ***algorithmic complexity*** *by applying mathematical optimization, perturbation theory, and inexact arithmetic and other tools in order to solve mathematical problems that today are not solvable by numerical or symbolic methods alone.*

[Zhi 2003, ASCM'03]

"Displacement structure in computing approximate GCD of univariate polynomials."

[Li, Yang and Zhi SNC'05]

End or Handbook section [Corless, Kaltofen, Watt]

The challenge of hybrid symbolic numeric algorithms is to explore the effects of imprecision, discontinuity, and ***algorithmic complexity*** *by applying mathematical optimization, perturbation theory, and inexact arithmetic and other tools in order to solve mathematical problems that today are not solvable by numerical or symbolic methods alone.*

"Interactive Supercomputing"

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 $\overline{f}$ of f and $\gcd(f, \frac{\partial f}{\partial x})$
("deflation") and factor the approximately square-free kernel $\overline{f}$

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 $\overline{f}$ of f and $\gcd(f, \frac{\partial f}{\partial x})$
("deflation") and factor the approximately square-free kernel $\overline{f}$

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

Ex.	$\deg(f_i)$	coeff. error	backward error	bkwd err. w/ G-N iter	*time(sec)*	*iters*	*impr.*
1	2,3	10^{-2}	1.08e–2	1.02e–3	7.764	7	10.6×
2	5,5	10^{-13}	1.07e–12	1.18e–13	6.813	2	9.0×
3	10,10	10^{-7}	9.95e–7	2.87e–7	157.09	3	3.4×
4	7,8	10^{-9}	1.94e–8	2.38e–9	50.222	16	8.2×
5	3,3,3	0	1.24e–13	6.44e–14	19.517	1	2.4×
6	6,6,10	10^{-5}	1.47e–4	7.24e–6	1329.4	4	20.3×
7	9,7	10^{-4}	2.18e–4	7.07e–5	74.157	4	3.1×
8	4,4,4,4,4	10^{-5}	3.34e–3	8.56e–6	5345.5	4	390.6×
9	3,3,3	10^{-1}	8.03e–1	1.06e–1	33.062	16	7.6×
10	12,7,5	10^{-5}	3.16e–4	8.02e–6	1766.7	4	39.4×

Table of Benchmarks Continued

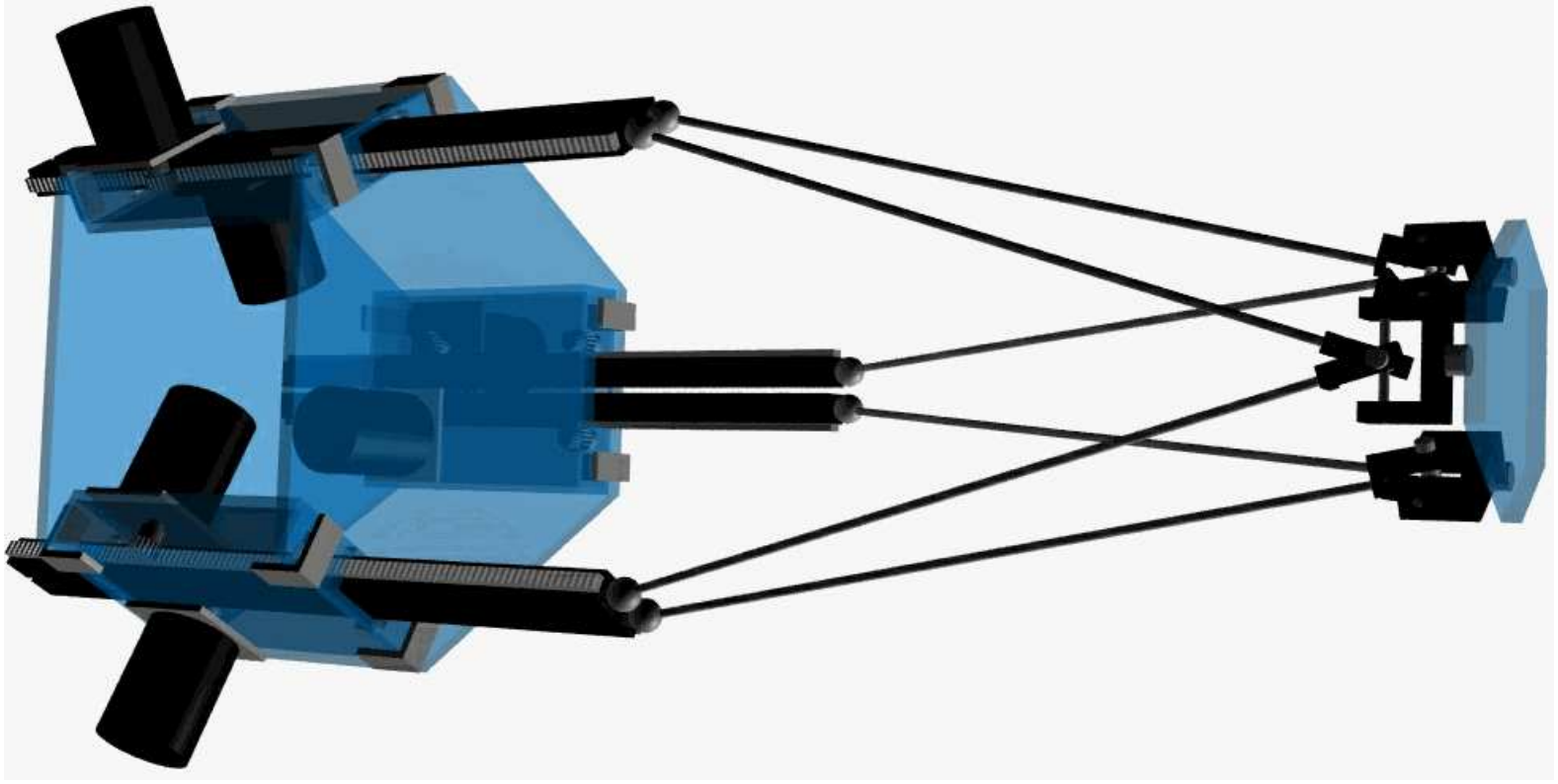
Ex.	$\deg(f_i)$	coeff. error	backward error	bkwd err. w/ G-N iter	*time(sec)*	*iters*	*impr.*
11	12,7,5	10^{-5}	7.77e–5	7.66e–6	2737.6	4	10.2×
12	12,7,5	10^{-3}	5.82e–3	7.66e–4	4288.7	6	7.6×
13	$5,(5)^2$	10^{-5}	6.84e–5	6.52e–6	46.751	3	10.5×
14	$(5)^3,3,(2)^4$	10^{-10}	2.60e–8	3.93e–9	136.39	2	6.6×
15	5,5	10^{-5}	1.55e–5	7.91e–6	559.30	3	2.0×
15a	2,2	10^{-5}	4.62e–13	3.23e–14	2.871	2	14.4×
15b	2,3	10^{-2}	7.44e–4	3.78e–4	6.687	4	2.0×
16	18,18	10^{-6}	4.50e–6	6.65e–7	5945.9	3	6.8×
17	18,18	10^{-6}	4.03e–6	6.61e–7	10348.	3	6.1×
18	6,6	10^{-7}	2.97e–7	5.10e–8	31.829	2	3.8×

More than two variables by sparse 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}, are from [Sommese, Verschelde, Wampler 2004]

Stewart Platform Example



Josh Targownik’s bypass surgery motorized manipulator

Current Factorization Investigations

- Use structured total least squares, possibly with constraints.

Current Factorization Investigations

- Use structured total least squares, possibly with constraints.

- Investigation of displacement operators for generalized Sylvester and Ruppert matrices.

Current Factorization Investigations

- Use structured total least squares, possibly with constraints.
- Investigation of displacement operators for generalized Sylvester and Ruppert matrices.
- Compute nearest factorization with a given degree pattern, e.g., all linear factors.

Current Factorization Investigations

- Use structured total least squares, possibly with constraints.
- Investigation of displacement operators for generalized Sylvester and Ruppert matrices.
- Compute nearest factorization with a given degree pattern, e.g., all linear factors.
- Apply sparse interpolation to handle sparse multivariate problems

Code + Benchmarks at:

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

or

`google->kaltofen` (click on “Software”)

Danke schön!