SNAP Package for *Mathematica* and Its Applications*

Kosaku Nagasaka
nagasaka@main.h.kobe-u.ac.jp

Division of Mathematics and Informatics,
Department of Science of Human Environment,
Faculty of Human Development, Kobe University, JAPAN.

Abstract

SNAP(Symbolic Numeric Algebra for Polynomials) Package for *Mathematica* is a *Mathematica* package which provides various functions to compute approximate algebraic properties, approximate GCD of polynomials for example. For practical situations, the package does not have enough functionalities yet for now, since the package is an on-going project. At this time, our aim is showing how an unified tolerance mechanism that we introduce for the package works. Using the mechanism, we can continue approximate calculations under certified tolerances.

1 Introduction

Recently, there are a lot of results in the area called Symbolic Numeric Algebra, especially for polynomials (for example, approximate GCD for univariate polynomials [11, 12, 1, 4] and approximate factorization for bivariate polynomials [3, 10, 13, 15]). We think that those results have been getting practical qualities, and we should combine them and implement it into integrated one computer algebra system. In fact, *Maple* has such a special package called "SNAP" (Symbolic-Numeric Algorithms for Polynomials).

We have been developing our "SNAP" package for *Mathematica* , and it is an abbreviation for Symbolic Numeric **Algebra** for Polynomials. By Algebra, we mean that continuous applicabilities of approximate operations: for example, computing an approximate GCD between an empirical polynomial and a polynomial which is the nearest singular polynomial computed by SNAP functions, of another empirical polynomial. This continuous applicability is more important for practical computations.

Our aim of this package is providing practical implementations of SNAP functions with an unified tolerance mechanism for polynomials like *Mathematica*'s floating-point number or Kako and Sasaki's effective floating-point number [8]. Our idea is very simple. We only have to

*This research is partly helped by Grants-in-Aid for Scientific Research, Ministry of Education, Culture, Sports, Science and Technology, JAPAN, #16700016.

add 1) new data structures representing such polynomials with tolerances, 2) basic calculation routines for the structures and 3) SNAP functions for the structures. We note that the package does not have enough functionalities yet for now, since the package is an on-going project. At this time, simple tolerance representations, l^2-norm, l^1-norm and l^∞-norm for coefficients vector of polynomials, are only implemented.

2 Tolerance Mechanism

We introduce the following data structures for empirical polynomials. They are very simple but there is no system in which we can use the following structures unconsciously like as *Mathematica*'s floating-point number and Kako and Sasaki's effective floating-point number [8].

Definition 1 We define the following approximate polynomial structures (like a set), for the given polynomial $f(x) \in \mathbb{C}[x]$ and tolerance $\varepsilon \in \mathbb{R}$.

$$P_2(f,\varepsilon) = \{\tilde{f} \mid \tilde{f} \in \mathbb{C}[x],\ \deg_x \tilde{f} \le \deg_x f,\ \|f-\tilde{f}\|_2 \le \varepsilon\}. \tag{2.1}$$

$$P_1(f,\varepsilon) = \{\tilde{f} \mid \tilde{f} \in \mathbb{C}[x],\ \deg_x \tilde{f} \le \deg_x f,\ \|f-\tilde{f}\|_1 \le \varepsilon\}. \tag{2.2}$$

$$P_\infty(f,\varepsilon) = \{\tilde{f} \mid \tilde{f} \in \mathbb{C}[x],\ \deg_x \tilde{f} \le \deg_x f,\ \|f-\tilde{f}\|_\infty \le \varepsilon\}. \tag{2.3}$$

◁

Lemma 1 *We have the following properties, for polynomials $g(x)$ and $h(x)$ of degrees n and m, respectively.*

$$\begin{aligned}\forall \tilde{g} \in P_2(g,\varepsilon_g),\ \forall \tilde{h} \in P_2(h,\varepsilon_h),&\\ \tilde{g}+\tilde{h} \in P_2(g+h, \varepsilon_g+\varepsilon_h),&\\ \tilde{g}\times\tilde{h} \in P_2(gh, \sqrt{c}\cdot(\|h\|_2\,\varepsilon_g + \|g\|_2\,\varepsilon_h + \varepsilon_g\varepsilon_h)),&\end{aligned} \tag{2.4}$$

$$\begin{aligned}\forall \tilde{g} \in P_1(g,\varepsilon_g),\ \forall \tilde{h} \in P_1(h,\varepsilon_h),&\\ \tilde{g}+\tilde{h} \in P_1(g+h, \varepsilon_g+\varepsilon_h),&\\ \tilde{g}\times\tilde{h} \in P_1(gh, \|h\|_1\,\varepsilon_g + \|g\|_1\,\varepsilon_h + \varepsilon_g\varepsilon_h),&\end{aligned} \tag{2.5}$$

$$\begin{aligned}\forall \tilde{g} \in P_\infty(g,\varepsilon_g),\ \forall \tilde{h} \in P_\infty(h,\varepsilon_h),&\\ \tilde{g}+\tilde{h} \in P_\infty(g+h, \varepsilon_g+\varepsilon_h),&\\ \tilde{g}\times\tilde{h} \in P_\infty(gh, c\cdot(\|h\|_\infty\,\varepsilon_g + \|g\|_\infty\,\varepsilon_h + \varepsilon_g\varepsilon_h)),&\end{aligned} \tag{2.6}$$

where $c = \min\{n,m\}+1$. ◁

Proof The former properties of each expression is directly proved by the norm's triangle inequality. The latter ones are proved by the following properties of these norms.

$$\begin{aligned}&\|p_1p_2\|_2 \le \sqrt{\min\{n,m\}+1}\,\|p_1\|_2\|p_2\|_2 = \sqrt{c}\,\|p_1\|_2\|p_2\|_2,\\ &\|p_1p_2\|_1 \le \|p_1\|_1\|p_2\|_1,\\ &\|p_1p_2\|_\infty \le (\min\{n,m\}+1)\,\|p_1\|_\infty\|p_2\|_\infty = c\,\|p_1\|_\infty\|p_2\|_\infty,\end{aligned}$$

where p_1 and p_2 are polynomials of degrees n and m, respectively. □

Corollary 1 *We have the following properties, for $g(x)$ of degree n.*

$$\begin{array}{l} P_2(g,\varepsilon_g) \subseteq P_1(g,\sqrt{n+1}\varepsilon_g), \\ P_2(g,\varepsilon_g) \subseteq P_\infty(g,\varepsilon_g), \\ P_1(g,\varepsilon_g) \subseteq P_2(g,\varepsilon_g), \\ P_1(g,\varepsilon_g) \subseteq P_\infty(g,\varepsilon_g), \\ P_\infty(g,\varepsilon_g) \subseteq P_2(g,\sqrt{n+1}\varepsilon_g), \\ P_\infty(g,\varepsilon_g) \subseteq P_1(g,(n+1)\varepsilon_g), \end{array} \tag{2.7}$$

◁

According to the above, related basic calculation routines are implemented in the package.

3 Approximate Operations

We implemented basic approximate operations: a polynomial division (quotient and remainder), the nearest singular polynomial, an approximate GCD and a root finding. Since our aim is showing merits of SNAP package with an unified tolerance mechanism, we implemented old but well-known algorithms for those calculations. Recent algorithms will be implemented in the future, and we have been working on it.

Let $g(x)$ and $h(x)$ be polynomials of degrees n and m $(\leq n)$ with tolerances ε_g and ε_h, respectively, be

$$\begin{array}{l} g(x) = g_n x^n + g_{n-1}x^{n-1} + \cdots + g_1 x + g_0,\ g_i \in \mathbb{C}, \\ h(x) = h_m x^m + h_{m-1}x^{m-1} + \cdots + h_1 x + h_0,\ h_i \in \mathbb{C}. \end{array}$$

In the rest of this paper, $\tilde{*}$ denotes an arbitrary representative element of $P_2(*,\varepsilon_*)$, $P_1(*,\varepsilon_*)$ and $P_\infty(*,\varepsilon_*)$, respectively.

3.1 Polynomial Division

Let $q(x)$, $r(x)$, $\tilde{q}(x)$ and $\tilde{r}(x)$ be polynomials satisfying

$$\begin{array}{l} g(x) = q(x)h(x) + r(x),\ \ \deg_x q = n-m,\ \deg_x r < m, \\ \tilde{g}(x) = \tilde{q}(x)\tilde{h}(x) + \tilde{r}(x),\ \ \deg_x \tilde{q} = \deg_x \tilde{g} - \deg_x \tilde{h},\ \deg_x \tilde{r} < \deg_x \tilde{h}, \end{array}$$

and H be the following non-singular matrix of size $(n+1)\times(n+1)$, whose elements are coefficients of $h(x)$.

$$H = \begin{pmatrix} h_m & 0 & \cdots & \cdots & 0 & 0 \\ h_{m-1} & h_m & \ddots & \ddots & \vdots & \vdots \\ \vdots & h_{m-1} & \ddots & \ddots & \vdots & \vdots \\ h_0 & \vdots & \ddots & \ddots & 0 & \vdots \\ \vdots & \vdots & \ddots & \ddots & h_m & 0 \\ 0 & \cdots & \cdots & \cdots & h_{m-1} & h_m \end{pmatrix}.$$

By the lemma 2.7.1 and the theorem 2.7.2 in [5] (see Appendix) and their proofs, we have the following corollaries.

Corollary 2 *If the following expression holds,*

$$\sigma_2\sqrt{n+1}\ \varepsilon_h < 1,\ \ \sigma_2 = \|H^{-1}\|_2,$$

we have

$$\begin{array}{l} \forall \tilde{g} \in P_2(g, \varepsilon_g),\ \forall \tilde{h} \in P_2(h, \varepsilon_h),\ \tilde{q} \in P_2(q, \varepsilon_q),\ \tilde{r} \in P_2(r, \varepsilon_r), \\ \varepsilon_q = \sigma_2(\varepsilon_g + \sqrt{n+1}\varepsilon_h(\|q\|_2 + \sigma_2\varepsilon_g)/(1 - \sigma_2\sqrt{n+1}\varepsilon_h)), \\ \varepsilon_r = \varepsilon_g + \sqrt{\lceil \frac{n}{2} \rceil + 1} \cdot (\|h\|_2\ \varepsilon_q + \|q\|_2\ \varepsilon_h + \varepsilon_h\varepsilon_q). \end{array} \tag{3.1}$$

◁

Corollary 3 *If the following expression holds,*

$$\sigma_1\varepsilon_h < 1,\ \ \sigma_1 = \|H^{-1}\|_1,$$

we have

$$\begin{array}{l} \forall \tilde{g} \in P_1(g, \varepsilon_g),\ \forall \tilde{h} \in P_1(h, \varepsilon_h),\ \tilde{q} \in P_1(q, \varepsilon_q),\ \tilde{r} \in P_1(r, \varepsilon_r), \\ \varepsilon_q = \sigma_1(\varepsilon_g + \varepsilon_h(\|q\|_1 + \sigma_1\varepsilon_g)/(1 - \sigma_1\varepsilon_h)), \\ \varepsilon_r = \varepsilon_g + \|h\|_1\ \varepsilon_q + \|q\|_1\ \varepsilon_h + \varepsilon_h\varepsilon_q. \end{array} \tag{3.2}$$

◁

Corollary 4 *If the following expression holds,*

$$\sigma_\infty(n+1)\ \varepsilon_h < 1,\ \ \sigma_\infty = \|H^{-1}\|_\infty,$$

we have

$$\begin{array}{l} \forall \tilde{g} \in P_\infty(g, \varepsilon_g),\ \forall \tilde{h} \in P_\infty(h, \varepsilon_h),\ \tilde{q} \in P_\infty(q, \varepsilon_q),\ \tilde{r} \in P_\infty(r, \varepsilon_r), \\ \varepsilon_q = \sigma_\infty(\varepsilon_g + (n+1)\varepsilon_h(\|q\|_\infty + \sigma_\infty\varepsilon_g)/(1 - \sigma_\infty(n+1)\varepsilon_h)), \\ \varepsilon_r = \varepsilon_g + (\lceil \frac{n}{2} \rceil + 1) \cdot (\|h\|_\infty\ \varepsilon_q + \|q\|_\infty\ \varepsilon_h + \varepsilon_h\varepsilon_q). \end{array} \tag{3.3}$$

◁

We note that these quotients and remainders and ε-divisors [2, 12] are different, we give literal quotients and remainders. ε-divisor and ε-quotient have been implemented as part of the approximate GCD. Moreover, we can use better bounds for divisions if the degree constraints in the structure definitions, are not inequalities but equalities.

3.2 Nearest Singular Polynomial

The nearest singular polynomial [16, 9], of $f(x)$ is the nearest polynomial $\tilde{f}(x)$ which has a double root, minimizes $\| f(x) - \tilde{f} \|$ and has the same degree as $f(x)$. We implemented the algorithm by Zhi et al. [16, 17]. However, the similar problem, the nearest polynomial with constrained roots [7, 6] has not been implemented. We will implement it in the future.

3.3 Approximate GCD

As in recent studies [11, 12, 1, 4], computing an approximate GCD is important and there are many results. The problem is very simple, for the given polynomials $g(x)$ and $h(x)$ and the tolerance ε, finding polynomial $f(x)$ which maximize its degree and satisfies

$$f(x) \mid \tilde{g}(x), \;\; f(x) \mid \tilde{h}(x), \;\; \tilde{g}(x) \in P_*(g, \varepsilon \|g\|_*), \; \tilde{h}(x) \in P_*(h, \varepsilon \|h\|_*),$$

where $*$ denotes 2, 1 and ∞, respectively. $f(x)$ is called ε-GCD of polynomials $g(x)$ and $h(x)$ with tolerance ε. We implemented the algorithm by Pan [12], for the 2-norm case only.

3.4 Root Finding

Although finding the roots of polynomials is not a special operation, we have to take care of perturbations of the roots within the given approximate polynomial structure, since each element of a structure has different zero points from others. In this point of view, there is an useful study by Terui and Sasaki [14], based on Smith's celebrated theorem.

According to their results, we have the following corollary.

Corollary 5 *For any polynomial $\tilde{g}(x) \in P_*(g, \varepsilon_g)$, we have*

$$|\zeta_i - \tilde{\zeta}_{\pi(i)}| \leq n \frac{|g(\zeta_i)| + \varepsilon_g \sum_{j=0}^{n} |\zeta_i|^j}{(|g_n| + \varepsilon_g) |\prod_{j=1, \neq i}^{n} (\zeta_i - \zeta_j)|},$$

where $\zeta_1, \ldots, \zeta_n$ and $\tilde{\zeta}_{\pi(1)}, \ldots, \tilde{\zeta}_{\pi(n)}$ are the roots of $g(x)$ and $\tilde{g}(x)$, respectively, and $\pi(i)$ is a permutation on $\{1, \ldots, n\}$, which minimizes the maximum distance between the roots: $\max_i |\zeta_i - \tilde{\zeta}_{\pi(i)}|$. ◁

4 Implementation

According to the above discussions, we have implemented the structures, basic operations on the structures, functions which transform a structure into another, compute approximate polynomial quotient and remainder, the nearest singular polynomial and an approximate polynomial GCD, and some additional functions which adjust accuracy of numerical roots calculated by the built-in function `NSolve` and so on. We show some coding examples briefly.

Example 1 For the structures and basic operations, we have used *Mathematica*'s built-in function `UpSetDelayed`. The following simple definition shows us what `UpSetDelayed` provides.

```
SNAP[g_, epsg_] + SNAP[h_, epsh_] ^:= SNAP[g + h, epsg + epsh]
```

After evaluating the above, we can see follows.

```
 In[2]:= SNAP[1, 0.001] + SNAP[2, 0.002]
Out[2]:= SNAP[3, 0.003]
```

◁

Example 2 One of our aim is providing SNAP functions unconsciously like as *Mathematica*'s floating-point number and Kako and Sasaki's effective floating-point number [8], so we have used *Mathematica*'s `Format` features. The following simple definition shows us what `Format` provides.

```
Format[Literal[SNAP[f_, epsf_]]] := f
```

After evaluating the above, we can see follows.

```
 In[4]:= SNAP[1, 0.001] + SNAP[2, 0.002]
Out[4]:= 3
 In[5]:= FullForm[%]
Out[5]//FullForm=
         SNAP[3, 0.003‘]
```

◁

5 Examples

In this section, we give some examples of our SNAP package.

```
  In[1]:= << SNAP‘
  In[2]:= g=SNAP[xˆ5+5.503xˆ4+9.765xˆ3+7.647xˆ2+2.762x+0.37725]
 Out[2]:= 0.37725 + 2.762x + 7.647x^2 + 9.765x^3 + 5.503x^4 + x^5
  In[3]:= h=SNAP[xˆ4 - 2.993xˆ3 - 0.7745xˆ2 + 2.0070x + 0.7605]
 Out[3]:= 0.7605 + 2.007x - 0.7745x^2 - 2.993x^3 + x^4
  In[4]:= FullForm[g]
 Out[4]//FullForm=
          SNAP[Plus[0.37725‘, Times[2.762‘, x], Times[7.647‘,
           Power[x, 2]], Times[9.765‘, Power[x, 3]],
           Times[5.503‘, Power[x, 4]], Power[x, 5]],
           1.537912594467717‘*ˆ-15, 5, AbsolutePolynomial2Norm]
  In[5]:= Tolerance[g]
 Out[5]:= 1.53791 × 10^-15
```

As in the above, `SNAP[]` generates an object representing our structure in which a tolerance is generated from their maximum `Accuracy` if omitted. We can see their actual data by `FullForm` and their tolerances by `Tolerance[]`.

By the method studied by Terui and Sasaki [14], our package can bound radii of existence domains of the roots, of the given approximate or empirical polynomial. In this case, we bound the roots of the nearest singular polynomial which is computed by another SNAP function.

```
  In[6]:= ng=NearestSingularPolynomial[g]
 Out[6]:= 0.377204 + 2.76202x + 7.64699x^2 + 9.76501x^3 + 5.503x^4 + x^5
  In[7]:= SNAPQ[ng]
 Out[7]:= True
```

```
 In[8]:= x /. NSolve[ng==0,x]
```
$$\text{Out[8]:= } \{-3.00005313, -0.999351282, -0.516441372, -0.49, -0.49\}$$
```
 In[9]:= Tolerance[%]
```
$$\text{Out[9]:= } \{9.877 \times 10^{-14}, 1.336 \times 10^{-12}, 4.781 \times 10^{-10}, 0.003549, 0.003549\}$$
```
 In[10]:= PolynomialGCD[ng, h]
Out[10]:= 1
 In[11]:= g2 = SetTolerance[ng, 0.3];
 In[12]:= PolynomialGCD[g2, h]
```
$$\text{Out[12]:= } 0.380209 + 1.50293x + 1.48876x^2$$

As in the above, after loading our package, `PolynomialGCD[]` can treat SNAP structures and can compute an approximate GCD corresponding to their tolerances. In this case, the original polynomials ng and h do not have any approximate GCD in their tolerances, hence we enlarged their tolerances up to 0.3 by `SetTolerance[]`, to get their approximate GCD.

6 Conclusion

In this paper, we introduce our SNAP package for *Mathematica*, which gives us a lot of practical functionalities, though it has been on preliminary stages. There are many approximate algorithms and ideas that should be implemented. For example, the structure provided by bounding radii of existence domains of roots, as the following definition, should be implemented, however, we've not done.

Definition 2 We define the following approximate polynomial structure, for the given polynomial $f(x) \in \mathbb{C}[x]$, $n = \deg_x f$ and tolerance $\varepsilon \in \mathbb{R}$.

$$P_{rr}(f, \varepsilon) = \{\tilde{f} \mid \tilde{f} \in \mathbb{C}[x], \ \deg_x \tilde{f} = n, \ \min_{\pi \in \Sigma_n} \max_i |\zeta_i - \tilde{\zeta}_{\pi(i)}| \leq \varepsilon\}, \tag{6.1}$$

where ζ_i and $\tilde{\zeta}_{\pi(i)}$ are the roots of $f(x)$ and $\tilde{f}(x)$, respectively, and Σ_n denotes the set of permutations on $\{1, \ldots, n\}$. ◁

This definition provides the following useful property, however it is complicated for the plus operation.

$$\forall \tilde{g} \in P_{rr}(g, \varepsilon_g), \ \forall \tilde{h} \in P_{rr}(h, \varepsilon_h), \ \ \tilde{g} \times \tilde{h} \in P_{rr}(gh, \max\{\varepsilon_g, \varepsilon_h\}).$$

Moreover, there are a lot of problems which are solved for ideal inputs and approximate outputs with some tolerances, but not solved for inputs with non-zero tolerances. We have to study those problems to make our package more useful.

However, our project has just been started, algorithms and ideas that should be implemented, will be done as soon as possible, in the near future. We note that our SNAP package will be downloadable with online help documents in December 2004, and will be demonstrated in the conference, with recent features.

Appendix

We cite some related lemma and theorem, where $\kappa(A)$ denotes the condition number of matrix A: $\kappa(A) = \|A^{-1}\|\|A\|$.

Lemma 2 (Lemma 2.7.1 in [5]) *Suppose*

$$\begin{array}{cc} Ax = b, & A \in \mathbb{R}^{n\times n},\ 0 \neq b \in \mathbb{R}^n, \\ (A + \Delta A)y = b + \Delta b, & \Delta A \in \mathbb{R}^{n\times n}, \Delta b \in \mathbb{R}^n, \end{array}$$

with $\|\Delta A\| \leq \varepsilon\ \|A\|$ *and* $\|\Delta b\| \leq \varepsilon\ \|b\|$. *If* $\varepsilon\kappa(A) = r < 1$, *then* $A + \Delta A$ *is non-singular and*

$$\frac{\|y\|}{\|x\|} \leq \frac{1+r}{1-r}.$$

◁

Theorem 1 (Theorem 2.7.2 in [5]) *If the above conditions hold, then*

$$\frac{\|y - x\|}{\|x\|} \leq \frac{2\varepsilon}{1-r}\kappa(A).$$

◁

References

[1] B. Beckermann and G. Labahn. When are two numerical polynomials relatively prime? *J. Symb. Comput.*, 267:677–689, 1998.

[2] R. M. Corless, P. M. Gianni, B. M. Trager, and S. M. Watt. The singular value decomposition for polynomial systems. In *Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation, ISSAC '95*, pages 195–207, 1995.

[3] R. M. Corless, M. W. Giesbrecht, M. Hoeij, I. S. Kotsireas, and S. M. Watt. Towards factoring bivariate approximate polynomials. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation, ISSAC 2001*, pages 85–92, 2001.

[4] I. Z. Emiris, A. Galligo, and H. Lombardi. Certified approximate univariate gcds. *J. Pure Appl. Algebra*, 117&118:229–251, 1997.

[5] G. H. Golub and C. F. V. Loan. *Matrix Computations Third Edition, Johns Hopkins Series in the Mathematical Sciences*. The Johns Hopkins University Press, Baltimore, 1996.

[6] M. A. Hitz and E. Kaltofen. Efficient algorithms for computing the nearest polynomial with constrained roots. In *Proceedings of the 1998 International Symposium on Symbolic and Algebraic Computation, ISSAC '98*, pages 236–243, 1998.

[7] M. A. Hitz, E. Kaltofen, and Y. N. Lakshman. Efficient algorithms for computing the nearest polynomial with a real root and related problems. In *Proceedings of the 1999 International Symposium on Symbolic and Algebraic Computation, ISSAC '99*, pages 205–212, 1999.

[8] F. Kako and T. Sasaki. Proposal of "effective floating-point number" for approximate algebraic computation. *preprint*, 1997.

[9] N. Karmarkar and Y. N. Lakshman. Approximate polynomial greatest common divisors and nearest singular polynomials. In *Proceedings of the 1996 International Symposium on Symbolic and Algebraic Computation, ISSAC '96*, pages 35–39, 1996.

[10] Z. Mou-tan and R. Unbehauen. Approximate factorization of multivariate polynomials. *Signal Processing*, 14:141–152, 1988.

[11] V. Y. Pan. Approximate polynomial gcds, padé approximation, polynomial zeros and bipartite graphs. In *Proc. Ninth Annual ACM-SIAM Symp. on Discrete Algorithms*, pages 68–77, 1998.

[12] V. Y. Pan. Computation of approximate polynomial gcds and an extension. *Inform. and Comput.*, 167(2):71–85, 2001.

[13] T. Sasaki. Approximate multivariate polynomial factorization based on zero-sum relations. In *Proceedings of the 2001 International Symposium on Symbolic and Algebraic Computation, ISSAC 2001*, pages 284–291.

[14] A. Terui and T. Sasaki. "approximate zero-points" of real univariate polynomial with large error terms. *IPSJ J.*, 41:974–989, 2000.

[15] Y. H. W. Wu, H. J. Stetter, and L. Zhi. Pseudofactors of multivariate polynomials. In *Proceedings of the 2000 International Symposium on Symbolic and Algebraic Computation, ISSAC 2000*, pages 161–168, 2000.

[16] L. Zhi and W. Wu. Nearest singular polynomials. *J. Symbolic Comput.*, 26(6):667–675, 1998.

[17] L. Zhi, W. Wu, M.-T. Noda, and H. Kai. Hybrid method for computing the nearest singular polynomials. *MM Research Preprints*, 20:229–239, 2001.