Analyses and Implementations of Chordality-Preserving Top-Down Algorithms for Triangular Decomposition

Mingyu Dong and Chenqi Mou(✉)

LMIB–School of Mathematical Sciences, Beihang University, Beijing 100191, China
{mingyudong,chenqi.mou}@buaa.edu.cn

Abstract. When the input polynomial set has a chordal associated graph, top-down algorithms for triangular decomposition are proved to preserve the chordal structure. Based on these theoretical results, sparse algorithms for triangular decomposition were proposed and demonstrated with experiments to be more efficient in case of sparse polynomial sets. However, existing implementations of top-down triangular decomposition are not guaranteed to be chordality-preserving due to operations which potentially destroy the chordality. In this paper, we first analyze the current implementations of typical top-down algorithms for triangular decomposition in the Epsilon package to identify these chordality-destroying operations. Then modifications are made accordingly to guarantee new implementations of such algorithms are chordality-preserving. In particular, the technique of dynamic checking is introduced to ensure that the modifications also keep the computational efficiency. Experimental results with polynomial sets from biological systems are also reported.

Keywords: Triangular decomposition · Chordal graph · Sparsity · Implementation

1 Introduction

Symbolic computation, also called computer algebra, is an interdisciplinary subject of mathematics and computer science which studies how to solve mathematical problems in terms of symbolic objects by using algorithms and their implementations [11]. As an indispensable method in symbolic computation, triangular decomposition transforms any multivariate polynomial set into finitely many triangular sets or systems which are in the triangular shape with respect to their greatest variables and thus much easier to solve, making operations with polynomial systems like solving them algorithmically feasible [1,16,32,37].

After the introduction of characteristic set, a special kind of triangular set, by Ritt [28,29], solid development on the theories, methods, and algorithms of

This work was partially supported by the National Natural Science Foundation of China (NSFC 11971050) and Beijing Natural Science Foundation (Z180005).

F. Boulier et al. (Eds.): CASC 2022, LNCS 13366, pp. 124–142, 2022.
https://doi.org/10.1007/978-3-031-14788-3_8

triangular decomposition has been witnessed in the last decades [2,3,7–9,12,13,17–19,30,31,38], accompanied by many successful applications of triangular decomposition in scientific and engineering areas, e.g., in automated reasoning of geometric theorems [9,36], stability analysis of biological systems [27,35], and cryptography [5,13,15] etc. Well-known implementations for triangular decomposition include the Epsilon package for top-down triangular decomposition for MAPLE [33], the RegularChains library for regular decomposition in MAPLE [20], the wsolve package for characteristic decomposition [34], and the built-in implementations of triangular decomposition in SINGULAR [14].

This paper focuses on top-down algorithms for triangular decomposition which preserve the chordal structure. The connections between chordal graphs and triangular sets were first established by Cifuentes and Parrilo in their study on the chordal network of polynomial systems by associating a graph to a polynomial set [10]. They also showed that algorithms due to Wang [30,31] are more efficient when the input polynomial set has a chordal associated graph. Their works inspired Mou and his collaborators to study chordal graphs in top-down algorithms for triangular decomposition: they proved that such algorithms preserve the chordal structure and thus are also sparsity-preserving, explaining the experimental observations by Cifuentes and Parrilo [22]. Then based on these theoretical results top-down algorithms for sparse triangular decomposition were proposed and applied to solve large polynomial systems arising from stability analysis of biological systems [23–25]. Furthermore, algorithms for incremental triangular decomposition and for cylindrical algebraic decomposition were also proved to preserve the chordal structure, leading to more efficient algorithm variants in the sparse case [6,21].

Though those algorithms for triangular decomposition are proved to be chordality-preserving at the algorithmic level, in real computation with their existing implementations, instances where chordality is destroyed are reported. This means that in the implementations of these top-down algorithms for triangular decomposition, there exist procedures or operations which are against the overall top-down strategy of the algorithm, introducing unwanted relationships between the variables in the polynomial sets. In this paper, we first analyze the current implementations of top-down algorithms for triangular decomposition in the Epsilon package to identify the operations which destroy the chordal structure and then modify them accordingly to have real chordality-preserving implementations for top-down triangular decomposition. Furthermore, we introduce the technique of dynamic checking to make the best use of simplification which speeds up the computation of triangular decomposition considerably while keeping the implementations chordality-preserving. The effectiveness and efficiency of our modified implementations were demonstrated with experiments with benchmark polynomial systems in the ODEbase database for biological models[1].

To our best knowledge, the implementations we present in this paper are the first ones for chordality-preserving top-down triangular decomposition in the community of symbolic computation. It is planned to incorporate these

[1] https://odebase.cs.uni-bonn.de/ODEModelApp.

implementations into the next release of the Epsilon package. We believe that the four operations we identify in the original implementations in the Epsilon package to potentially destroy the chordality-preserving property of an implementation of top-down triangular decomposition are also useful as references to those who plan to apply chordal graphs in top-down elimination methods like those for cylindrical algebraic decomposition.

2 Preliminaries

Let $\mathbb{K}$ be a computable field. Denote by $\mathbb{K}[\boldsymbol{x}]$ the polynomial ring in the variables $x_1, \ldots, x_n$ over $\mathbb{K}$. We fix a variable ordering $x_1 < \cdots < x_n$ unless otherwise specified. For a polynomial $F \in \mathbb{K}[\boldsymbol{x}]$, the greatest variable that effectively appears in it is called the *leading variable* of F and denoted by $\mathrm{lv}(F)$. Let $x_k = \mathrm{lv}(F)$. Then F can also be regarded as a univariate polynomial in x_k, with coefficients from $\mathbb{K}[x_1, \ldots, x_{k-1}]$, and accordingly it can be written as $F = \sum_{i=0}^{d_k} C_i x_k^i$, where $C_i \in \mathbb{K}[x_1, \ldots, x_{k-1}]$, $d_k = \deg(F, x_k)$, and $C_{d_k} \neq 0$. The leading coefficient C_{d_k} here is called the *initial* of F, denoted by $\mathrm{ini}(F)$, and plays an important role in the theory of triangular decomposition.

2.1 Triangular Set and Triangular Decomposition

Definition 1. Let $\mathcal{T} = [T_1, \ldots, T_r]$ be an ordered polynomial set in $\mathbb{K}[\boldsymbol{x}]$. If none of $T_1, \ldots, T_r$ is constant and $\mathrm{lv}(T_1) < \cdots < \mathrm{lv}(T_r)$, then $\mathcal{T}$ is called a *triangular set* in $\mathbb{K}[\boldsymbol{x}]$.

Clearly the following polynomial set forms a triangular set in $\mathbb{K}[x_1, \ldots, x_4]$

$$[x_2 + x_1, (x_2^2 - x_1^2 + 2)x_3, (x_3 + x_2)x_4 + x_3 - 1]. \tag{1}$$

One can impose additional conditions on the polynomials and their initials in a triangular set to make it even stronger and have more desirable properties. Commonly used triangular sets include regular sets (or called regular chains) [8, 17, 38], simple sets [3, 26, 31], irreducible sets [32, Sect. 4.1], and normal sets [32, Sect. 5.2], etc.

Let $\mathcal{P}$, $\mathcal{Q} \subset \mathbb{K}[\boldsymbol{x}]$ be two polynomial sets. We are interested in the zeros defined by $\mathcal{P}$ as equations and $\mathcal{Q}$ as inequations. To be specific, we study the system of equations $P = 0$ and inequations $Q \neq 0$ for all $P \in \mathcal{P}$ and $Q \in \mathcal{Q}$ and denote this system by $\mathcal{P} = 0$ and $\mathcal{Q} \neq 0$ accordingly. Let $\overline{\mathbb{K}}$ be the algebraic closure of $\mathbb{K}$. Then we denote by $\mathsf{Z}(\mathcal{P})$ the common zeros of the polynomials in $\mathcal{P}$ in $\overline{\mathbb{K}}$ and denote $\mathsf{Z}(\mathcal{P}/\mathcal{Q}) := \mathsf{Z}(\mathcal{P}) \setminus \mathsf{Z}(S)$, where $S = \prod_{Q \in \mathcal{Q}} Q$. As one may find, the definition $\mathsf{Z}(\mathcal{P}/\mathcal{Q})$ is indeed the zero set of $\mathcal{P} = 0$ and $\mathcal{Q} \neq 0$.

Definition 2. Let $(\mathcal{T}, \mathcal{U})$ be a pair of polynomial sets in $\mathbb{K}[\boldsymbol{x}]$. Then it is called a *triangular system* if $\mathcal{T}$ is a triangular set, say $\mathcal{T} = [T_1, \ldots, T_r]$, and for each $i = 2, \ldots, r$ and any $\overline{\boldsymbol{x}}_{i-1} \in \mathsf{Z}([T_1, \ldots, T_{i-1}]/\mathcal{U})$, we have $\mathrm{ini}(T_i)(\overline{\boldsymbol{x}}_{i-1}) \neq 0$.

Definition 3. Let $\mathcal{P}$ and $\mathcal{Q}$ be two finite polynomial sets in $\mathbb{K}[\boldsymbol{x}]$. Then the process to compute finite many triangular systems $(\mathcal{T}_1, \mathcal{T}_1), \ldots, (\mathcal{T}_s, \mathcal{U}_s)$ such that $\mathsf{Z}(\mathcal{P}/\mathcal{Q}) = \bigcup_{i=1}^{s} \mathsf{Z}(\mathcal{T}_i/\mathcal{U}_i)$ is called *triangular decomposition* of $\mathcal{P}$ and $\mathcal{Q}$.

When the triangular set $\mathcal{T}$ in a system $(\mathcal{T}, \mathcal{U})$ is regular, simple, irreducible, or normal, the corresponding triangular system is also called so. Triangular decomposition to different kinds of triangular systems is also named after the resulting triangular systems. For example, in this paper, we are interested in top-down algorithms for regular and simple decomposition which decomposes a polynomial set into regular sets and simple sets, respectively.

Top-down algorithms for triangular decomposition refer to those handle the polynomials in a decreasing order with respect to their leading variables so that when handling the polynomials with a certain leading variable, those with strictly greater leading variables keep the same and newly generated polynomials in the process are only of smaller leading variables. The readers are referred to [23] for a formal definition of top-down triangular decomposition.

2.2 Sparse Triangular Decomposition Based on Chordal Graphs

Consider an undirected graph $G = (V, E)$, where $V := \{x_1, \ldots, x_n\}$ and E are, respectively, the sets of its vertices and edges. An edge in E connecting two vertices x_i and x_j is denoted by (x_i, x_j). Since G is an undirected graph, we have $(x_i, x_j) = (x_j, x_i)$. Let S be a non-empty subset of V. Then the subgraph with its vertices in S and edges consisting of the edges in E whose endpoints belong to S is called the *induced subgraph* of G with respect to S and denoted by $G[S]$. If an induced subgraph $G[S]$ is complete, meaning that all the vertices are connected with edges, then we say that S forms a *clique* in G.

Definition 4. Let $G = (V, E)$ be an undirected graph with $V = \{x_1, \ldots, x_n\}$ and $x_{i_1} < x_{i_2} < \cdots < x_{i_n}$ be an ordering of all the vertices. If for each $j = i_1, i_2, \cdots, i_n$, the set $\{x_j\} \cup \{x_k : x_k < x_j \text{ and } (x_k, x_j) \in E\}$ forms a clique in G, then this vertex ordering is called a *perfect elimination ordering* of G. If G has some perfect elimination ordering, then it is said to be *chordal*.

There exist effective algorithms, e.g., the MCS (maximum cardinality search) algorithm [4], to test whether a given graph is chordal, returning also a perfect elimination ordering in the affirmative case. The specifications of the MCS algorithm are formulated in Algorithm 1 for later references.

Algorithm 1: MCS algorithm $(B, \boldsymbol{\sigma}) := \mathsf{MCS}(G)$

Input: G, a graph
Output: $(B, \boldsymbol{\sigma})$: if G is chordal, then $B = \mathsf{True}$ and $\boldsymbol{\sigma}$ is one perfect elimination ordering of G, otherwise $B = \mathsf{False}$ and $\boldsymbol{\sigma} = \emptyset$

For a non-chordal graph $G = (V, E)$, one can make it chordal by adding a set of new edges E'. The process to find a minimal set E' of edges for G so that $G' := (V, E \cup E')$ is chordal is called *chordal completion* of G. We denote this process

by $G' = \mathsf{ChordalComp}(G)$. Here by minimal we mean that any strict subset E'' of E' cannot make $G'' = (V, E \cup E'')$ chordal. The resulting supergraph G' is also called a chordal completion of G if no ambiguity may happen.

In Fig. 1 below, the subgraph (a) is not chordal and (b) is a chordal completion of it with a perfect elimination ordering $x_1 < x_2 < x_3 < x_4 < x_5$. Note that (c) is not a chordal completion of (a) even though it is a chordal graph, for the set of added edges is not minimal.

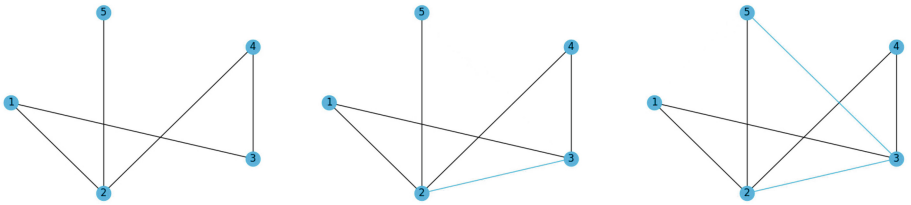

Fig. 1. A chordal graph and its chordal completion (Color figure online)

For a polynomial $F \in \mathbb{K}[\boldsymbol{x}]$, denote by $\mathrm{supp}(F)$ the set of the variables F contains; similarly for a polynomial set $\mathcal{F} \subset \mathbb{K}[\boldsymbol{x}]$ we define $\mathrm{supp}(\mathcal{F}) = \{\mathrm{supp}(F) : F \in \mathcal{F}\}$. In the following way a graph is associated to $\mathcal{F}$.

Definition 5. Let $\mathcal{F} \subset \mathbb{K}[\boldsymbol{x}]$ be a polynomial set. Set $V = \mathrm{supp}(\mathcal{F})$ and $E = \{(x_i, x_j) : \text{there exists } F \in \mathcal{F} \text{ such that } x_i, x_j \in \mathrm{supp}(F)\}$. Then the graph $G = (V, E)$ is called the *associated graph* of $\mathcal{F}$, denoted by $G(\mathcal{F})$.

The associated graph of the polynomial set (1) is illustrated as below in Fig. 2. One can see that the associated graph of a polynomial set $\mathcal{F}$ reflects how the variables in $\mathcal{F}$ are interconnected, and, thus, the definition below for variable sparsity via associated graphs is natural.

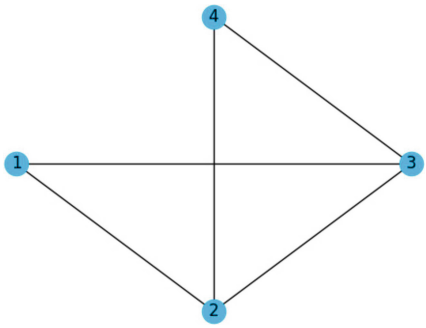

Fig. 2. Associated graph of (1)

Definition 6. Let $\mathcal{F} \subset \mathbb{K}[\boldsymbol{x}]$ be a polynomial set and $G(\mathcal{F}) = (V, E)$ be its associated graph. Then the *variable sparsity* of $\mathcal{F}$ is defined to be $s_v := |E|/\binom{2}{|V|}$.

It is proved that if the input polynomial set has a chordal associated graph and a perfect elimination ordering of this graph is used as the variable ordering, top-down algorithms for triangular decomposition preserve the variable sparsity of the polynomial sets in the process of decomposition and, thus, the following algorithmic framework for sparse triangular decomposition is proposed in [23] and verified to be more efficient for polynomial systems which are sparse with respect to their variables [24]. In Algorithm 2 below, RegDec() represents a top-down algorithm for regular decomposition. It is called an algorithmic framework because by simply replacing RegDec() with any other top-down algorithm for triangular decomposition, say one for simple decomposition, one will have a sparse version of that top-down algorithm.

Algorithm 2: Top-down algorithm for sparse regular decomposition $(\boldsymbol{\sigma}, \psi) = \mathsf{SparseRegDec}(\mathcal{F})$

Input: $\mathcal{F} \in \mathbb{K}[\boldsymbol{x}]$, a sparse polynomial set with respect to the variables
Output: $(\boldsymbol{\sigma}, \Psi)$, a perfect elimination ordering $\boldsymbol{\sigma}$ and triangular decomposition Ψ of $\mathcal{F}$ with respect to $\boldsymbol{\sigma}$

```
(B, σ) := MCS(G(F));
if B = False then
    G' := ChordalComp(G(F));
    (B, σ) := MCS(G');
Ψ := RegDec(F, σ);
return (σ, Ψ);
```

3 Chordality in Top-Down Triangular Decomposition

In this paper, we focus on three typical top-down algorithms for triangular decomposition which have been implemented in the Epsilon package: they are the algorithms for regular decomposition (denoted by RegSer), for simple decomposition (by SimSer), and for triangular decomposition (by TriSer). These algorithms, detailed in [32] and proved to be chordality-preserving [23,25], rely heavily on subresultants for polynomial elimination in the decomposition and, thus, share a similar underlying structure.

We applied sparse algorithms for triangular decomposition in the framework of Algorithm 2 with the implementations in the Epsilon package of these three algorithms to polynomial systems arising from biological models in the database ODEbase in our experiments. For each studied polynomial system $\mathcal{F}$, a finite number of triangular systems $(\mathcal{T}_1, \mathcal{U}_1), \ldots, (\mathcal{T}_s, \mathcal{U}_s)$ will be computed after the triangular decomposition, and we want to check whether all the associated graphs $G(\mathcal{T}_1), \ldots, G(\mathcal{T}_s)$ are indeed subgraphs of the input chordal graph that is either $G(\mathcal{F})$ or its chordal completion. In fact, in our experiments, instances with extra added edges not contained in the input chordal graph were reported.

Example 1. Consider the following polynomial system in Model BM483

$$\{100x_3 - x_1 - 10^{-5}x_1^2x_4 + 10^{-5}x_1x_4 + 0.2x_6, -5\times10^{-6}x_2^2x_3 + 5\times10^{-6}x_2x_3 + 0.1x_5$$
$$100x_4 - x_2 - 10^{-5}x_2^2x_3 + 10^{-5}x_2x_3 + 0.2x_5, -5\times10^{-6}x_1^2x_4 + 5\times10^{-6}x_1x_4 + 0.1x_6,$$
$$5\times10^{-6}x_2^2x_3 - 5\times10^{-6}x_2x_3 - 0.1x_5, 5\times10^{-6}x_1^2x_4 - 5\times10^{-6}x_1x_4 - 0.1x_6,$$
$$x_1 - 100x_3, x_2 - 100x_4\}.$$

The associated graph of this polynomial set is shown below in Fig. 3 and it is a chordal graph with one perfect elimination ordering $x_1 < x_3 < x_6 < x_4 < x_2 < x_5$.

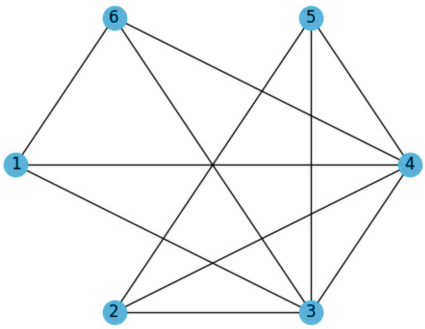

Fig. 3. A chordal associated graph

With this perfect elimination ordering as the variable order, the RegSer function in Epsilon package returns the following two triangular systems

$$([x_1 - 100x_3, x_1^2x_4 - x_1x_4 - 20000x_6, x_2 - 100x_4, 100x_4^2x_1 - x_1x_4 - 20000x_5],$$
$$\{x_1, x_1 - 1\}), ([x_1^2 - x_1, x_1 - 100x_3, x_6, x_2 - 100x_4, 100x_4^2x_1 - x_1x_4 - 20000x_5], \emptyset).$$

Their associated graphs are shown in (a) and (b) of Fig. 4 respectively.

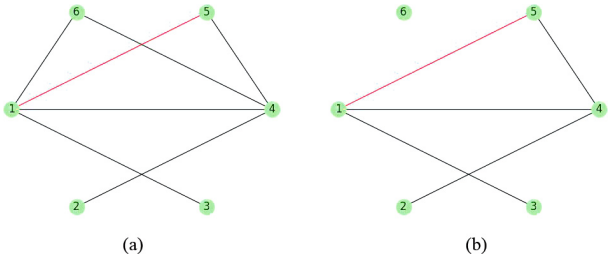

Fig. 4. Associated graph (Color figure online)

As one may find, the red edge in these two graphs does not appear in Fig. 3. This means that, even though the top-down algorithm RegSer for regular decomposition is proved theoretically to preserve the chordal structure of the input polynomial set, there may still exist specific operations in the actual implementation of this algorithm which can destroy the chordality-preserving property. In fact, out of 40 polynomial systems we picked from the ODEbase database, our experiments found 7 to produce extra added edges in some associated graphs of the returned triangular systems output by the RegSer function in the Epsilon package.

4 When is the Chordality Destroyed?

In this section, we report our study and analyses on the source codes of the implementations of the functions RegSer, SimSer, and TriSer for top-down triangular decomposition in the Epsilon package. In total, we found the following 4 operations which may destroy the chordality in these implementations.

4.1 Simplifying a Polynomial Set with Its Binomials

Let $\mathcal{P}$ be a polynomial set appearing in the process of triangular decomposition which represents the equations $\mathcal{P} = 0$. In case there exists some simple polynomial in $\mathcal{P}$ like a binomial, we can simplify $\mathcal{P}$ with this simple polynomial. This operation is formulated as Simplify($\mathcal{P}$) as follows.

Simplify($\mathcal{P}$)
1. Consider each polynomial T in the polynomial set $\mathcal{P}$ representing $\mathcal{P} = 0$.
2. If T is a binomial, write $T = c_1\boldsymbol{t}_1 + c_2\boldsymbol{t}_2$, with the term $\boldsymbol{t}_1$ greater than $\boldsymbol{t}_2$.
3. For any polynomial $P \in \mathcal{P}\backslash\{T\}$, replace every occurrence $\boldsymbol{t}_1$ in P, if any, with $-\frac{c_2}{c_1}\boldsymbol{t}_2$.

Example 2. Take the polynomial set (1) as input, we have
Input: Simplify($\{x_2 + x_1, (x_2^2 - x_1^2 + 2)x_3, (x_3 + x_2)x_4 + x_3 - 1\}$)
Output: $\{x_2 + x_1, 2\,x_3, -x_1x_4 - 1\}$

Clearly after the operation the output polynomial set is simpler. However, with the associated graph of the output polynomial set shown in Fig. 5, one can find an added edge in red compared with Fig. 2.

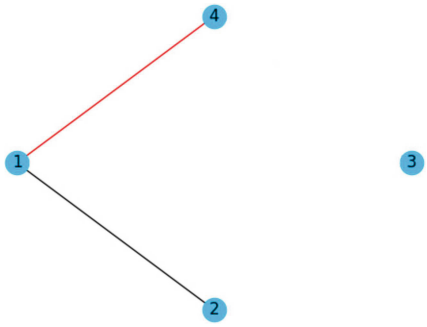

Fig. 5. Associated graph with one added edge after simplification (Color figure online)

Example 3. When computing regular decomposition of the polynomial system in Model BM483 in ODEbase database with the function RegSer in the Epsilon package with the variable ordering $x_1 < x_3 < x_6 < x_4 < x_2 < x_5$. A triangular set

$$\mathcal{T} = [x_1 - 100x_3, x_2 - 100x_4, 100x_3x_4^2 - x_3x_4 - 200x_5, \\ x_1^2x_4 - x_1x_4 + 100000x_1 - 10000000x_3 - 20000x_6]$$

is simplified with Simplify(). With the binomial $x_1 - 100x_3$, the substitution $x_3 = x_1/100$ into other polynomials in $\mathcal{T}$ results in a new triangular set

$$\mathcal{T}' = [x_1 - 100x_3, x_2 - 100x_4, 100x_1x_4^2 - x_1x_4 - 20000x_5, x_1^2x_4 - x_1x_4 - 20000x_6].$$

One can find that the vertices x_1 and x_5 appear in $100x_1x_4^2 - x_1x_4 - 20000x_5$ in $\mathcal{T}'$ now, introducing a new edge (x_1, x_5) in $G(\mathcal{T}')$.

4.2 Simplifying a Polynomial System with Binomials

In the process of top-down triangular decomposition, when the handling down to the variable x_{k+1} from x_n has finished, there are many stored polynomial systems $(\mathcal{P}, \mathcal{Q})$ such that for $i = n, \ldots, k+1$, the number of polynomials in $\mathcal{P}$ with leading variable x_i is at most 1. This means that triangular decomposition has been done for $\mathcal{P}$ down to x_{k+1}. At this point, the polynomials in both $\mathcal{P}$ and $\mathcal{Q}$ may be simplified with a simple polynomial $T \in \mathcal{P}$ if $\mathrm{lv}(T) \leq x_k$. This operation, formulated as Filter$((\mathcal{T}, \mathcal{U}), x_k)$ below, is similar to Simplify() in Sect. 4.1, except that the substitution here is by the other term of the binomial.

Filter$((\mathcal{P}, \mathcal{Q}), x_k)$
1. Consider each polynomial T in the polynomial set $\mathcal{P}$ representing $\mathcal{P} = 0$.
2. If T is a binomial and $\mathrm{lv}(T) \leq x_k$, write $T = c_1\boldsymbol{t}_1 + c_2\boldsymbol{t}_2$, with the term $\boldsymbol{t}_1$ greater than $\boldsymbol{t}_2$.
3. For any polynomial P in $\mathcal{P} \cup \mathcal{Q} \setminus \{T\}$, replace every occurrence $\boldsymbol{t}_2$ in P, if any, with $-\frac{c_1}{c_2}\boldsymbol{t}_1$.

Example 4. The following example shows that this operation can potentially destroy the chordality.
Input: Filter$((\{x_4 + x_2, x_3 + x_1 + x_2, x_4^2 - x_2^2 + x_3\}, \{x_1, x_2, x_4\}), x_4)$
Output: $(\{x_4 + x_2, x_3 + x_1 - x_4, x_3\}, \{x_1, x_4\})$

One can find that both the equation and inequation sets become simpler after this operation. The associated graphs of the input and output polynomial sets are shown in (a) and (b) of Fig. 6 respectively, with the newly added edges colored in red.

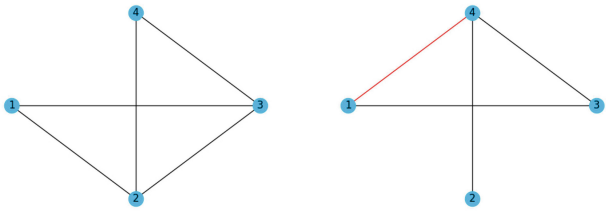

Fig. 6. Associated graph with one added edge after simplification (Color figure online)

4.3 Reducing Inequation Polynomials with a Polynomial in the Triangular Set

In top-down triangular decomposition, reduction like the pseudo-division and subresultant (see, e.g., Sects. 1.2–1.3 of [32] for the definitions of these fundamental operations in triangular decomposition) is applied to the polynomials whose leading variables are the one of interest, say x_k. After the reduction, only one polynomial T whose leading variable equals x_k is left, and this polynomial T is an element in the triangular set. Whenever such a polynomial T is found, one can perform reduction on all the current polynomials in $\mathcal{Q}$ representing inequations $\mathcal{Q} \neq 0$ in the decomposition to simplify $\mathcal{Q}$. This operation is formulated as Reduce$(\mathcal{Q}, T)$ below, in which prem(Q, T) computes the pseudo-remainder of Q with respect to T in lv(T).

Reduce$(\mathcal{Q}, T)$
1. For each $Q \in \mathcal{Q}$, replace Q with prem(Q, T).

Example 5. We report our experimental results with the function TriSer applied to the polynomial system in Model BM335 in ODEbase database, where the operation Reduce() introduced an extra edge in the process of decomposition.

Figure 7(a) is the associated graph of the input polynomial system in Model BM335 and it is not chordal. Then chordal completion is applied to it with

MCS(), resulting in a chordal graph as (b), with added edges colored in blue, and the following perfect elimination ordering

$$x_2 < x_6 < x_1 < x_5 < x_4 < x_3 < x_7 < x_{23} < x_{12} < x_8 < x_{29} < x_{22} < x_9 < x_{18} < x_{10} < x_{27} < x_{11} < x_{25} < x_{13} < x_{26} < x_{28} < x_{14} < x_{21} < x_{24} < x_{17} < x_{16} < x_{20} < x_{19} < x_{15}.$$

Then performing pseudo division of $Q = -943230000000000x_1x_5 - 437100x_8$ in the inequation polynomial set $\mathcal{Q}$ by $T = x_2(101200000000x_1x_5 + 403x_6)$ results in $\text{prem}(Q, T, x_5) = 41317575x_2x_6 - 4808100x_2x_8$. One can find that in the resulting graph shown in (c), an edge in red connecting x_2 and x_8 is added and it is not included in the chordal graph (b).

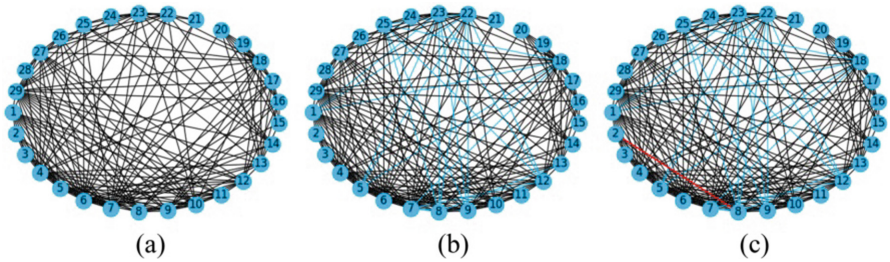

Fig. 7. One added edge with reduction on inequation polynomials (Color figure online)

4.4 Reducing a Triangular System with a Polynomial in the Triangular Set

In top-down triangular decomposition, whenever a triangular system $(\mathcal{T}, \mathcal{U})$ is constructed, one can simplify it by performing pseudo-division on all the polynomials in $\mathcal{T}$ and $\mathcal{U}$ by any polynomial in $\mathcal{T}$. This operation is formulated as ReduceTS$((\mathcal{T}, \mathcal{U}))$ below.

ReduceTS$((\mathcal{T}, \mathcal{U}))$
1. For each polynomial $T \in \mathcal{T}$, replace P with $\text{prem}(P, T)$ for each $P \in \mathcal{T} \cup \mathcal{U} \setminus \{T\}$.

Example 6.
Input: ReduceTS$([x_2^2 + x_1, x_3^5 - x_1, x_3^{10}x_4 + x_2], \{x_3\}))$
Output: $([x_2^2 + x_1, x_3^5 - x_1, x_1^2x_4 + x_2], \{x_3\})$

The associated graphs of the input and output polynomial sets are shown in (a) and (b) of Fig. 8 respectively, with one added edge colored in red.

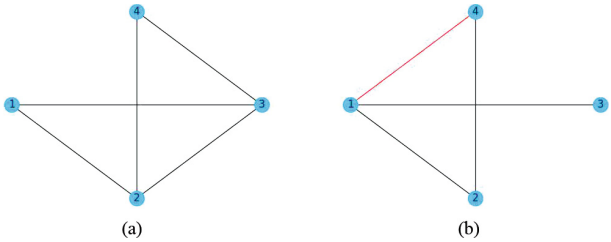

Fig. 8. Associated graph with one added edge after reduction (Color figure online)

4.5 Analysis on the Four Operations

As one can easily find, the occurrences of added edges in the functions in Sects. 4.1–4.2 are all due to simplification via substitution. It is worth mentioning that neither algebraic simplification with binomials in Sects. 4.1–4.2 nor reducing inequation polynomials with a polynomial in the triangular set in Sect. 4.3 appears in the original descriptions of top-down algorithms for triangular decomposition. These operations are found in the implementations of such algorithms only for the efficiency consideration. Similarly, reducing a triangular system is not included in the original descriptions of top-down algorithms for triangular decomposition either. This operation in the implementation is to make a triangular system *perfect*, a stronger notion than our target triangular system. To be short, this operation is for the quality of the output after triangular decomposition. To conclude, the existences of all these four identified "bad" operations in the implementations do not affect the correctness of the implementations.

5 Chordality-Preserving Implementations and Experiments

As analyzed above, all the four identified operations which potentially destroy the chordal structure in triangular decomposition do not affect the correctness of the implementations. Then one straightforward method to construct chordality-preserving implementations of top-down algorithms for triangular decomposition is merely removing the related codes.

5.1 Removing Chordality-Destroying Operations

Take our modifications to the function RegSer for regular decomposition in Epsilon package for example. We simply removed related codes of the four chordality-destroying operations in the implementations, resulting in a new function which we name RegSerC. Then we tested this new function with the benchmark polynomial systems from ODEbase database to see whether it indeed preserves chordality and to compare its efficiency against the original RegSer function.

The experimental results are summarized in Table 1, and all the experiments in this paper were carried out on a Macbook Pro laptop with a 2 GHz quad core i5 CPU and 16 GB 3733 MHz MHz LPDDR4 memory under the operating system MacOS Catalina 10.15.5. In this table, the timings (CPU time in seconds), number of branches in the computed triangular decomposition, and number of branches with added edges compared with the input chordal graph are recorded in the columns "Time", "#Bran", and "#Edge", respectively. In particular, the number "XXX-i" in the column "No." means that the corresponding ID in the ODEbase is BIOMD0000000XXX with the ith perfect elimination ordering (the specific ordering is not provided due to its length) and a dash "—" in the column "Time" means that the corresponding computation does not finish within 2 h.

From this table, we have the following observations: (1) There is no added edge reported with the new function RegSerC for all the finished computation, meaning that (at least) experimentally this function is chordality-preserving; (2) There are considerable efficiency decreases with this new function against the original RegSer one, for all tested systems. Take Model BM332 for example, in total we tested it with 5 perfect elimination orderings: the RegSer function finishes the computation around 312 s on average, while for 4 out of 5 variable orderings, the new RegSerC function cannot finish within 2 h. For the remaining ordering, the computation time with RegSerC is 6.71 times of that with RegSer.

5.2 Further Optimization with Dynamic Checking

Simply removing related codes of chordality-destroying operations can indeed guarantee that the chordality is preserved but unfortunately it also diminishes the efficiency of the implementations. This means that algebraic simplification and reduction are quite effective to improve the computational efficiency. Or in other words, we should keep as much algebraic simplification and reduction as possible in the implementations while still preserving the chordality. Following this strategy we introduce the technique of dynamic checking to test whether some extra edge will be added if some specific algebraic simplification or reduction is performed. This test is in fact quite easy: for example, if algebraic simplification is applied to a polynomial F with a binomial T to have a new polynomial F', then a simple comparison of $G(F')$ to the input chordal graph would tell us whether some extra edge would be added.

As an example, we formulate the technique of dynamic checking with Simplify in Sect. 4.1. Other algebraic simplification and reduction with dynamic checking are the same and we omit their formal descriptions.

Algorithm 3: Algebraic simplification with dynamic checking

Input: A polynomial set $\mathcal{F} \subset \mathbb{K}[\boldsymbol{x}]$, the input chordal graph G
Output: A polynomial set $\mathcal{F}' \subset \mathbb{K}[\boldsymbol{x}]$ after simplification

```
for T ∈ F do
    if T is a binomial then
        Write T = c1 t1 + c2 t2, with the term t1 greater than t2;
        for F ∈ F \ {T} do
            F' := polynomial obtained by replacing t1 in F by −(c2/c1) t2;
            if G(F') ⊆ G then
                F := F';
return F;
```

Table 1. Experiments with chordality-preserving top-down implementations for regular decomposition

No.	#Var	RegSer			RegSerCO			RegSerC		
		Time	#Bran	#Edge	Time	#Bran	#Edge	Time	#Bran	#Edge
220-1	58	91.73	1728	288	99.99	1728	0	273.55	1728	0
220-2	58	256.41	5184	1944	307.00	5184	0	450.24	5184	0
332-1	78	328.96	2125	985	376.91	2176	0	—	—	—
332-2	78	248.33	2082	998	272.08	2083	0	—	—	—
332-3	78	244.08	3130	1692	284.24	3160	0	—	—	—
332-4	78	215.94	2719	1030	245.37	2719	0	—	—	—
332-5	78	522.94	3680	1312	642.25	3708	0	2423.99	3732	0
333-1	54	22.82	314	72	25.37	302	0	503.07	319	0
333-2	54	26.52	251	72	28.38	251	0	80.02	299	0
333-3	54	29.85	393	96	35.24	393	0	—	—	
333-4	54	11.93	197	87	14.45	197	0	75.35	260	0
333-5	54	22.97	269	44	26.04	269	0	151.04	366	0
334-1	74	321.06	2223	117	353.48	2223	0	—	—	—
334-2	74	235.91	1640	1081	279.44	1640	0	—	—	—
334-3	74	502.42	3183	1175	544.60	3183	0	—	—	—
334-4	74	257.60	2313	590	274.76	2313	0	1719.56	2552	0
335-1	34	8.22	262	90	8.22	262	0	14.91	205	0
335-2	34	7.03	256	81	7.86	256	0	12.32	196	0
362	34	14.68	500	165	14.80	500	0	30.36	450	0
431-1	27	5.21	66	21	5.29	67	0	18.11	78	0
431-2	27	3.05	42	7	3.45	42	0	10.07	58	0
475-1	23	5.50	40	8	4.43	38	0	8.10	38	0
475-2	23	2.02	42	6	2.20	42	0	4.14	48	0
478-1	33	5.82	67	20	5.88	80	0	9.24	80	0
478-2	33	1.30	34	7	1.44	34	0	1.64	34	0
504-1	75	83.15	142	142	94.04	240	0	2055.42	586	0
504-2	75	112.63	295	284	119.07	319	0	1972.26	2778	0
599-1	30	40.54	46	36	39.75	46	0	—	—	—
599-2	30	—	—	—	—	—	—	—	—	—
599-3	30	30.60	50	30	31.07	50	0	—	—	—

Denote by RegSerCO the new function integrated with algebraic simplification and reduction with dynamic checking. We experimented with RegSerCO with the same polynomial systems, comparing with RegSer and RegSerC, and the experimental results are recorded in Table 1 too. It can be seen that this new function RegSerCO also preserves the chordal structure in the process of triangular decomposition as RegSerC, at the same time, the efficiency loss is under control compared with RegSer: the computation time with RegSerCO is about 1.15 times on average of that with RegSer.

5.3 Chordality-Preserving Implementations for SimSer and TriSer Functions

We did similar modifications to the two functions SimSer and TriSer in the Epsilon package by introducing algebraic simplification and reduction with dynamic

Table 2. Experiments with chordality-preserving top-down implementations for simple decomposition

	SimSer			SimSerCO		
No.	Time	#Bran	#Edge	Time	#Bran	#Edge
220-1	141.06	2016	228	152.78	2016	0
220-2	329.39	6048	2160	395.78	6048	0
332-1	262.13	2082	998	303.69	2083	0
332-2	321.88	3628	2068	405.18	3590	0
333-1	23.53	314	72	24.04	302	0
333-2	27.52	251	83	27.52	251	0
333-3	42.63	450	118	47.25	450	0
333-4	15.98	218	89	18.36	218	0
333-5	31.86	283	54	36.22	283	0
335-1	7.97	262	90	8.85	262	0
335-2	8.03	256	81	8.05	256	0
362-1	26.56	635	213	26.83	636	0
362-2	23.67	874	442	26.64	861	0
362-3	21.99	602	209	22.97	602	0
431-1	7.30	74	23	7.35	72	0
431-2	3.31	45	9	3.26	45	0
431-3	2.76	28	11	2.89	28	0
475-1	5.25	38	4	5.03	36	0
475-2	4.86	42	6	2.86	42	0
475-3	3.94	72	24	4.62	74	0
475-4	2.36	42	18	2.61	42	0
478-1	7.62	97	22	8.00	110	0
478-2	1.64	43	8	1.72	43	0
478-3	1.60	40	0	1.76	40	0

checking to have two new functions SimSerCO and TriSerCO respectively. The results of our experiments with these two new functions are recorded in Tables 2 and 3, respectively.

It can be seen that both these two new functions SimSerCO and TriSerCO preserve the chordality of the input polynomial systems and keep the same level of efficiency compared with the original functions: computation with SimSerCO is 1.17 times on average of that with SimSer (slightly slower), and computation with TriSer is 0.94 times on average of that with TriSer (slightly faster).

Table 3. Experiments with chordality-preserving top-down implementations for triangular decomposition

	TriSer			TriSerCO		
No.	Time	#Bran	#Edge	Time	#Bran	#Edge
220-1	1710.22	1728	288	1566.92	1728	0
220-2	—	—	—	—	—	—
332-1	4193.24	1902	977	3948.40	1883	0
332-2	—	—	—	—	—	—
333-1	48.01	248	72	48.86	236	0
333-2	46.20	241	74	49.54	178	0
333-3	99.65	343	84	105.40	343	0
333-4	19.61	162	85	24.64	162	0
333-5	37.62	196	44	41.06	196	0
335-1	19.57	245	102	20.00	245	0
335-2	17.89	244	81	19.38	244	0
362-1	52.49	393	158	54.59	394	0
362-2	69.67	483	225	71.99	448	0
362-3	49.82	392	163	58.70	389	0
431-1	8.33	64	26	8.78	55	0
431-2	3.47	41	7	3.85	41	0
431-3	2.81	28	12	2.74	26	0
475-1	4.86	40	8	4.54	38	0
475-2	2.24	42	6	2.42	42	0
475-3	3.84	54	26	4.37	54	0
475-4	1.74	42	18	2.09	42	0
478-1	4.86	40	8	4.54	38	0
478-2	2.24	42	6	2.42	42	0
478-3	2.20	38	0	2.13	38	0

6 Concluding Remarks and Future Work

In our experiments with sparse triangular decomposition, instances are found such that existing implementations of top-down triangular decomposition destroy the chordal structure of the input polynomial system, which is inconsistent with the proved theoretical results of the chordality-preserving property of such algorithms. The main contribution of this paper is the real chordality-preserving implementations of top-down triangular decomposition based on the Epsilon package, and they are, to our best knowledge, the first chordality-preserving ones for this kind of triangular decomposition. In order to achieve this, we first analyze the current implementations in the Epsilon package to identify four chordality-destroying operations. Corresponding modifications to these four operations with dynamic checking lead to chordality-preserving implementations. Experimental results with polynomial sets from biological systems show that these implementations are indeed chordality-preserving and their efficiency is comparable to original implementations.

In the future, more implementations for top-down triangular decomposition, like those for irreducible decomposition in which factorization over algebraic field extensions is essential, are planned to be investigated and further transformed into chordality-preserving ones. Furthermore, since the choice of a specific perfect elimination ordering also influences the computational efficiency of sparse triangular decomposition, we also plan to study the underlying reasons for the influence.

Acknowledgments. The authors would like to thank Prof. Dongming Wang for his insightful comments on the implementations in Epsilon package and the referees for their helpful comments resulting in improvements on the previous version of this paper.

References

1. Aubry, P., Lazard, D., Moreno Maza, M.: On the theories of triangular sets. J. Symb. Comput. **28**(1–2), 105–124 (1999)
2. Aubry, P., Moreno Maza, M.: Triangular sets for solving polynomial systems: a comparative implementation of four methods. J. Symb. Comput. **28**(1), 125–154 (1999)
3. Bächler, T., Gerdt, V., Lange-Hegermann, M., Robertz, D.: Algorithmic Thomas decomposition of algebraic and differential systems. J. Symb. Comput. **47**(10), 1233–1266 (2012)
4. Berry, A., Blair, J., Heggernes, P., Peyton, B.: Maximum cardinality search for computing minimal triangulations of graphs. Algorithmica **39**(4), 287–298 (2004)
5. Chai, F., Gao, X.S., Yuan, C.: A characteristic set method for solving Boolean equations and applications in cryptanalysis of stream ciphers. J. Syst. Sci. Complex. **21**(2), 191–208 (2008)
6. Chen, C.: Chordality preserving incremental triangular decomposition and its implementation. In: Bigatti, A.M., Carette, J., Davenport, J.H., Joswig, M., de Wolff, T. (eds.) ICMS 2020. LNCS, vol. 12097, pp. 27–36. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-52200-1_3

7. Chen, C., Golubitsky, O., Lemaire, F., Moreno Maza, M., Pan, W.: Comprehensive triangular decomposition. In: Ganzha, V.G., Mayr, E.W., Vorozhtsov, E.V. (eds.) CASC 2007. LNCS, vol. 4770, pp. 73–101. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75187-8_7
8. Chen, C., Moreno Maza, M.: Algorithms for computing triangular decompositions of polynomial systems. J. Symb. Comput. **47**(6), 610–642 (2012)
9. Chou, S.-C., Gao, X.-S.: Ritt-Wu's decomposition algorithm and geometry theorem proving. In: Stickel, M.E. (ed.) CADE 1990. LNCS, vol. 449, pp. 207–220. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-52885-7_89
10. Cifuentes, D., Parrilo, P.: Chordal networks of polynomial ideals. SIAM J. Appl. Algebra Geom. **1**(1), 73–110 (2017)
11. Cox, D., Little, J., O'Shea, D.: Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra. Undergraduate Texts in Mathematics, Springer, New York (1997). https://doi.org/10.1007/978-3-319-16721-3
12. Della Dora, J., Dicrescenzo, C., Duval, D.: About a new method for computing in algebraic number fields. In: Caviness, B.F. (ed.) EUROCAL 1985. LNCS, vol. 204, pp. 289–290. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-15984-3_279
13. Gao, X.S., Huang, Z.: Characteristic set algorithms for equation solving in finite fields. J. Symb. Comput. **47**(6), 655–679 (2012)
14. Greuel, G.M., Pfister, G., Bachmann, O., Lossen, C., Schönemann, H.: A Singular Introduction to Commutative Algebra. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-662-04963-1
15. Huang, Z., Lin, D.: Attacking bivium and trivium with the characteristic set method. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 2011. LNCS, vol. 6737, pp. 77–91. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21969-6_5
16. Hubert, E.: Notes on triangular sets and triangulation-decomposition algorithms I: polynomial systems. In: Winkler, F., Langer, U. (eds.) SNSC 2001. LNCS, vol. 2630, pp. 1–39. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45084-X_1
17. Kalkbrener, M.: A generalized Euclidean algorithm for computing triangular representations of algebraic varieties. J. Symb. Comput. **15**(2), 143–167 (1993)
18. Lazard, D.: A new method for solving algebraic systems of positive dimension. Discret. Appl. Math. **33**(1–3), 147–160 (1991)
19. Lazard, D.: Solving zero-dimensional algebraic systems. J. Symb. Comput. **13**(2), 117–131 (1992)
20. Lemaire, F., Moreno Maza, M., Xie, Y.: The RegularChains library in MAPLE. ACM SIGSAM Bull. **39**(3), 96–97 (2005)
21. Li, H., Xia, B., Zhang, H., Zheng, T.: Choosing the variable ordering for cylindrical algebraic decomposition via exploiting chordal structure. In: Proceedings of ISSAC 2021, pp. 281–288 (2021)
22. Mou, C., Bai, Y.: On the chordality of polynomial sets in triangular decomposition in top-down style. In: Proceedings ISSAC 2018, pp. 287–294 (2018)
23. Mou, C., Bai, Y., Lai, J.: Chordal graphs in triangular decomposition in top-down style. J. Symb. Comput. **102**, 108–131 (2021)
24. Mou, C., Ju, W.: Sparse triangular decomposition for computing equilibria of biological dynamic systems based on chordal graphs. In: IEEE/ACM Transactions Computational Biology and Bioinformatics (2022)

25. Mou, C., Lai, J.: On the chordality of simple decomposition in top-down style. In: Slamanig, D., Tsigaridas, E., Zafeirakopoulos, Z. (eds.) MACIS 2019. LNCS, vol. 11989, pp. 138–152. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-43120-4_12
26. Mou, C., Wang, D., Li, X.: Decomposing polynomial sets into simple sets over finite fields: the positive-dimensional case. Theoret. Comput. Sci. **468**, 102–113 (2013)
27. Niu, W., Wang, D.: Algebraic approaches to stability analysis of biological systems. Math. Comput. Sci. **1**(3), 507–539 (2008)
28. Ritt, J.: Differential Equations from the Algebraic Standpoint. AMS (1932)
29. Ritt, J.: Differential Algebra. AMS (1950)
30. Wang, D.: An elimination method for polynomial systems. J. Symb. Comput. **16**(2), 83–114 (1993)
31. Wang, D.: Decomposing polynomial systems into simple systems. J. Symb. Comput. **25**(3), 295–314 (1998)
32. Wang, D.: Elimination Methods. Texts and Monographs in Symbolic Computation, Springer Science & Business Media, New York (2001). https://doi.org/10.1007/978-3-7091-6202-6
33. Wang, D.: Epsilon: A library of software tools for polynomial elimination. In: Mathematical Software, pp. 379–389. World Scientific (2002)
34. Wang, D.: wsolve: A Maple package for solving system of polynomial equations (2004). http://www.mmrc.iss.ac.cn
35. Wang, D., Xia, B.: Stability analysis of biological systems with real solution classification. In: Proceedings of ISSAC 2005, pp. 354–361 (2005)
36. Wu, W.T.: Basic principles of mechanical theorem proving in elementary geometries. J. Autom. Reason. **2**(3), 221–252 (1986)
37. Wu, W.T.: A zero structure theorem for polynomial-equations-solving and its applications. In: Davenport, J.H. (ed.) EUROCAL 1987. LNCS, vol. 378, pp. 44–44. Springer, Heidelberg (1989). https://doi.org/10.1007/3-540-51517-8_84
38. Yang, L., Zhang, J.: Searching dependency between algebraic equations: an algorithm applied to automated reasoning. In: Artificial Intelligence in Mathematics, pp. 147–156 (1994)