COMPUTING THE DETERMINANT OF A MATRIX WITH POLYNOMIAL ENTRIES BY APPROXIMATION

XIAOLIN QIN, ZHI SUN, TUO LENG, YONG FENG

Abstract. Computing the determinant of a matrix with the univariate and multivariate polynomial entries arises frequently in the scientific computing and engineering fields. In this paper, an effective algorithm is presented for computing the determinant of a matrix with polynomial entries using hybrid symbolic and numerical computation. The algorithm relies on the Newton's interpolation method with error control for solving Vandermonde systems. It is also based on a novel approach for estimating the degree of variables, and the degree homomorphism method for dimension reduction. Furthermore, the parallelization of the method arises naturally.

1. Introduction

In the scientific computing and engineering fields, such as computing multipolynomial resultants [1], computing the implicit equation of a rational plane algebraic curve given by its parametric equations [2], and computing Jacobian determinant in multi-domain unified modeling [3], computing the determinant of a matrix with polynomial entries (also called symbolic determinant) is inevitable. Therefore, computing symbolic determinants is an active area of research [4–14]. There are several techniques for calculating the determinants of matrices with polynomial entries, such as expansion by minors [10], Gaussian elimination over the integers [11, 12], a procedure which computes the characteristic polynomial of the matrix [13], and a method based on evaluation and interpolation [5–9]. The first three algorithms belong to symbolic computations. As is well known, symbolic computations are principally exact and stable. However, they have the disadvantage of intermediate expression swell. The last one is the interpolation method, which as an efficient numerical method has been widely used to compute resultants and determinants, etc.. In fact, it is not approximate numerical computations but big number computations, which are also exact computations and only improve intermediate expression swell problem. Nevertheless, the main idea of black box approach takes an external view of a matrix, which is a linear operator on a vector space [14]. Therefore, it is particularly suited to the handling of large sparse or structured matrices over finite fields. In this paper, we propose an efficient approximate interpolation approach to remedy these drawbacks.

Mathematics subject classification (2010): 15A15, 41A05, 65Y10, 68W30.

Keywords and phrases: symbolic determinant; approximate interpolation; dimension reduction; Vandermonde systems; error controllable algorithm.
Hybrid symbolic-numerical computation is a novel method for solving large scale problems, which applies both numerical and symbolic methods in its algorithms and provides a new perspective of them. The approximate interpolation methods are still used to get the approximate results [14–16]. In order to obtain exact results, one usually uses exact interpolation methods to meliorate intermediate expression swell problem arising from symbolic computations [5, 7, 8, 16]. Although the underlying floating-point methods in principle allow for numerical approximations of arbitrary precision, the computed results will never be exact. Recently, the exact computation by intermediate of floating-point arithmetic has been an active area of solving the problem of intermediate expression swell in [15, 18–21]. The nice feature of the work is as follows: The initial status and final results are accurate, whereas the intermediate of computation is approximate. The aim of this paper is to provide a rigorous and efficient algorithm to compute symbolic determinants by approximate interpolation. In this paper, we restrict our study to a non-singular square matrix with polynomial entries and the coefficients of polynomial over the integers.

The rest of this paper is organized as follows. Section 2 first constructs the degree matrix of symbolic determinant on variables and gives theoretical support to estimate the upper bounds degree of variables, and then analyzes the error controlling for solving Vandermonde systems of equations by Newton’s interpolation method, finally proposes a reducing dimension method based on degree homomorphism. Section 3 proposes a novel approach for estimating the upper bound on degree of variables in symbolic determinant, and then presents algorithms of dimension reduction and lifting variables and gives a detailed example. Section 4 gives some experimental results. The final section makes conclusions.

2. Preliminary results

Throughout this paper, \(\mathbb{Z}\) and \(\mathbb{R}\) denote the set of the integers and reals, respectively. There are \(v\) variables named \(x_i\), for \(i = 1\) to \(v\). Denote the highest degree of each \(x_i\) by \(d_i\). Denoted by \(\Phi_{m,n}(\mathbb{F})\) the set of all \(m\) by \(n\) matrices over field \(\mathbb{F} = \mathbb{R}\), and abbreviate \(\Phi_{n,n}(\mathbb{F})\) to \(\Phi_n(\mathbb{F})\).

2.1. Estimating degree of variables

In this subsection, a brief description to Chio’s expansion is proposed. We also give the Theorem for estimating the upper bound on degree of variables in symbolic determinant.

**Lemma 1.** ([22]) Let \(A = [a_{ij}]\) be an \(n \times n\) matrix and suppose \(a_{11} \neq 0\). Let \(K\) denote the matrix obtained by replacing each element \(a_{ij}\) in \(A\) by \(\begin{vmatrix} a_{11} & a_{1j} \\ a_{i1} & a_{ij} \end{vmatrix}\). Then
Without loss of generality, we call it the degree matrix \( |A| = |K|/a_{11}^{n-2} \). That is, 

\[
\begin{vmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{21} & a_{22} & \cdots & a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{n1} & a_{n2} & \cdots & a_{nn}
\end{vmatrix}
\]

Remark 1. The proof of Lemma \( \square \) is clear. Multiply each row of \( A \) by \( a_{11} \) except the first, and then perform the elementary row operations, denote \( Op(2-a_{21} \cdot 1), Op(3-a_{31} \cdot 1), \cdots, Op(n-a_{n1} \cdot 1) \), where ‘1’, ‘2’, ‘⋯’, ‘n’ represents for the row index. We get 

\[
a_{11}^{-1} \cdot |A| = 
\begin{vmatrix}
  a_{11} & a_{12} & \cdots & a_{1n} \\
  a_{11}a_{21} & a_{11}a_{22} & \cdots & a_{11}a_{2n} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{11}a_{n1} & a_{11}a_{n2} & \cdots & a_{11}a_{nn}
\end{vmatrix} = a_{11}|K|.
\]

We observe that \( K \) is \( (n-1) \times (n-1) \) matrix, then the above procedure can be repeated until the \( K \) is \( 2 \times 2 \) matrix. It is a simple and straightforward method for calculating the determinant of a numerical matrix.

Lemma 2. Given two polynomials \( f(x_1) \) and \( g(x_1) \), the degree of the product of two polynomials is the sum of their degrees, i.e., 

\[
\text{deg}(f(x_1) \cdot g(x_1), x_1) = \text{deg}(f(x_1), x_1) + \text{deg}(g(x_1), x_1).
\]

The degree of the sum (or difference) of two polynomials is equal to or less than the greater of their degrees, i.e., 

\[
\text{deg}(f(x_1) \pm g(x_1), x_1) \leq \max\{\text{deg}(f(x_1), x_1), \text{deg}(g(x_1), x_1)\},
\]

where \( f(x_1) \) and \( g(x_1) \) are the univariate polynomials over field \( \mathbb{F} \), and \( \text{deg}(f(x_1), x_1) \) represents the highest degree of \( x_1 \) in \( f(x_1) \).

Let \( M = [M_{ij}] \) be an \( n \times n \) matrix and suppose \( M_{ij} \) is a polynomial with integer coefficients consisting of variables \( x_1, x_2, \cdots, x_v \), where the order of \( M \) is \( n \geq 2 \). Without loss of generality, we call it the degree matrix \( \Omega_1 = (\sigma_{ij})^\square \) for \( x_1 \) defined as:

\[\Omega_1, \Omega_2, \cdots, \Omega_v \] denote the degree matrix of \( x_1, x_2, \cdots, x_v \), respectively.
\[ \sigma_{ij} = \begin{cases} 
\text{highest degree of } x_i \text{ appears in the element } M_{ij}, \text{i.e., } \deg(M_{ij}, x_1), \\
0, \text{ if } x_i \text{ does not occur in } M_{ij}. 
\end{cases} \]

So, we can construct the degree matrix from \( M \) for all variables, respectively.

**Theorem 1.** \( M \) is defined as above. Suppose the \( 2 \times 2 \) degree matrix can be obtained from \( M \) for \( x_i (1 \leq i \leq v) \), denotes

\[
\Omega_i = \begin{bmatrix}
\sigma_{(n-2)(n-1)} & \sigma_{(n-2)n} \\
\sigma_{(n-2)n} & \sigma_{nn}
\end{bmatrix},
\]

then

\[
\maxdeg = \max\{\sigma_{(n-2)(n-1)} + \sigma_{nn}, \sigma_{(n-1)n} + \sigma_{n(n-1)}\}.
\]

That is, the maximum degree of variable is no more than

\[
\maxdeg = \sum_{i=3}^{n} (i-2)\sigma_{(n-i)(n-i+1)},
\]

where \( \sigma_{(n-2)(n-1)} = \deg(M_{(n-1)(n-1)}, x_i) \).

**Proof.** Considering the order \( n \) of symbolic determinant

\[
|M| = \begin{vmatrix}
M_{11} & M_{12} & \cdots & M_{1n} \\
M_{21} & M_{22} & \cdots & M_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
M_{n1} & M_{n2} & \cdots & M_{nn}
\end{vmatrix},
\]

by Chio’s expansion is from Remark[1] then

\[
|M| = \frac{1}{M_{11}^{n-2}} \begin{vmatrix}
M_{22}^{(1)} & M_{23}^{(1)} & \cdots & M_{2n}^{(1)} \\
M_{32}^{(1)} & M_{33}^{(1)} & \cdots & M_{3n}^{(1)} \\
\vdots & \vdots & \ddots & \vdots \\
M_{n2}^{(1)} & M_{n3}^{(1)} & \cdots & M_{nn}^{(1)}
\end{vmatrix},
\]

where

\[
M_{22}^{(1)} = M_{11}M_{22} - M_{12}M_{21}, M_{32}^{(1)} = M_{11}M_{32} - M_{12}M_{31}, \ldots, M_{nn}^{(1)} = M_{11}M_{nn} - M_{1n}M_{n1}.
\]

\[ \sigma_{ij}^{(1)} \text{ is defined by the same way for the rest of this paper.} \]
By Lemma\(^2\) for \(x_i\) we get
\[
\deg(M, x_i) \leq \max\{\sigma_{(n-1)(n-1)}^{(n-2)}, \sigma_{(n-1)n}^{(n-2)}, \sigma_{n(n-1)}^{(n-2)}\} - (n-2)\sigma_{11}
\]
\[-(n-3)\sigma_{22}^{(1)} - \cdots - \sigma_{(n-2)(n-2)}^{(n-3)} = \maxdeg - \sum_{i=3}^{n}(i-2)\sigma_{(n-i)(n-i+1)}^{(n-i)},
\]
where
\[
\maxdeg = \max\{\sigma_{(n-1)(n-1)}^{(n-2)}, \sigma_{mn}^{(n-2)}, \sigma_{(n-1)n}^{(n-2)} + \sigma_{n(n-1)}^{(n-2)}\}.
\]

The proof of Theorem\(^1\) is completed. It can be applied to all variables.

**Remark 2.** We present a direct method for estimating the upper bound on degrees of variables by computation of the degree matrices. Our method only needs the simple recursive arithmetic operations of addition and subtraction. Generally, we can obtain the exact degrees of all variables in symbolic determinant in practice.

### 2.2. Newton’s interpolation with error control

Let \(M\) be defined as above. Without loss of generality, we consider the determinant of a matrix with bivariate polynomial entries, and then generalize the results to the univariate or multivariate polynomial. A good introduction to the theory of interpolation can be seen in [23].

**Definition 1.** The Kronecker product of \(A = [a_{i,j}] \in \Phi_{m,n}(\mathbb{F})\) and \(B = [b_{ij}] \in \Phi_{p,q}(\mathbb{F})\) is denoted by \(A \otimes B\) and is defined to the block matrix
\[
A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix} \in M_{mp,nq}(\mathbb{F}).
\]
Notice that \(A \otimes B \neq B \otimes A\) in general.

**Definition 2.** With each matrix \(A = [a_{ij}] \in \Phi_{m,n}(\mathbb{F})\), we associate the vector \(\text{vec}(A) \in \mathbb{F}^{mn}\) defined by
\[
\text{vec}(A) \equiv [a_{11}, \cdots, a_{m1}, a_{12}, \cdots, a_{m2}, \cdots, a_{1n}, \cdots, a_{mn}]^T,
\]
where \(T\) denotes the transpose of matrix or vector.

Let the determinant of \(M\) be \(f(x_1, x_2) = \sum_{i,j} a_{ij}x_1^i x_2^j\) which is a polynomial with integer coefficients, and \(d_1, d_2\) be the bounds on the highest degree of \(f(x_1, x_2)\) in \(x_1, x_2\), respectively. We choose the distinct scalars \((x_{ij}, x_{2j})\) \((i = 0, 1, \cdots, d_1; j = 0, 1, \cdots, d_2)\).\(^3\)

\(^3\) \(d_1, d_2\) are defined by the same way for the rest of this paper.
and obtain the values of $f(x_1, x_2)$, denoted by $f_{ij} \in \mathbb{R}$ ($i = 0, 1, \ldots, d_1; j = 0, 1, \ldots, d_2$). The set of monomials is ordered as follows:

$$(1, x_1, x_1^2, \cdots, x_1^{d_1}) \times (1, x_2, x_2^2, \cdots, x_2^{d_2}),$$

and the distinct scalars in the corresponding order is as follows:

$$(x_{10}, x_{11}, \cdots, x_{1d_1}) \times (x_{20}, x_{21}, \cdots, x_{2d_2}).$$

Based on the bivariate interpolate polynomial technique, which is essential to solve the following linear system:

$$(V_{x_1} \otimes V_{x_2}) \text{vec}(a) = \text{vec}(F), \quad (2)$$

where the coefficients $V_{x_1}$ and $V_{x_2}$ are Vandermonde matrices:

$$V_{x_1} = \begin{pmatrix} 1 & x_{10} & x_{10}^2 & \cdots & x_{10}^{d_1} \\ 1 & x_{11} & x_{11}^2 & \cdots & x_{11}^{d_1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{1d_1} & x_{1d_1}^2 & \cdots & x_{1d_1}^{d_1} \end{pmatrix}, \quad V_{x_2} = \begin{pmatrix} 1 & x_{20} & x_{20}^2 & \cdots & x_{20}^{d_2} \\ 1 & x_{21} & x_{21}^2 & \cdots & x_{21}^{d_2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{2d_2} & x_{2d_2}^2 & \cdots & x_{2d_2}^{d_2} \end{pmatrix},$$

and

$$a = \begin{pmatrix} a_{00} & a_{01} & \cdots & a_{0d_2} \\ a_{10} & a_{11} & \cdots & a_{1d_2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{d_10} & a_{d_11} & \cdots & a_{d_1d_2} \end{pmatrix}, \quad F = \begin{pmatrix} f_{00} & f_{01} & \cdots & f_{0d_2} \\ f_{10} & f_{11} & \cdots & f_{1d_2} \\ \vdots & \vdots & \ddots & \vdots \\ f_{d_10} & f_{d_11} & \cdots & f_{d_1d_2} \end{pmatrix}.$$

Marco et al. [5] have proved in this way that the interpolation problem has a unique solution. This means that $V_{x_1}$ and $V_{x_2}$ are nonsingular and therefore $V = V_{x_1} \otimes V_{x_2}$, then the coefficient matrix of the linear system (2) is nonsingular. The following lemma shows us how to solve the system (2).

**Lemma 3.** ([24]) Let $\mathbb{F}$ denote a field. Matrices $A \in \Phi_{m,n}(\mathbb{F})$, $B \in \Phi_{d,p}(\mathbb{F})$, and $C \in \Phi_{m,q}(\mathbb{F})$ are given and assume $X \in \Phi_{n,p}(\mathbb{F})$ to be unknown. Then, the following equation:

$$(B \otimes A) \text{vec}(X) = \text{vec}(C) \quad (3)$$

is equivalent to matrix equation:

$$AXB^T = C. \quad (4)$$

Obviously, equation (4) is equivalent to the system of equations

\[
\begin{cases}
AY = C \\
BX = Y^T.
\end{cases}
\quad (5)
\]
Notice that the coefficients of system (2) are Vandermonde matrices, the reference [25] by the Newton’s interpolation method presented a progressive algorithm which is significantly more efficient than previous available methods in $O(d_i^2)$ arithmetic operations in Algorithm [1].

**Algorithm 1 (Björck and Pereyra algorithm)**

Input: a set of distinct scalars $(x_i, f_i)(0 \leq i \leq d_1)$;  
Output: the solution of coefficients $a_0, a_1, \ldots, a_{d_1}$.

1. **Step 1:** $c_{i}^{(0)} := f_i(i = 0, 1, \ldots, d_1)$  
   for $k = 0$ to $d_1 - 1$ do  
   $c_{i}^{(k+1)} := \frac{c_{i}^{(k)} - c_{i-k}^{(k)}}{x_i - x_{i-k-1}}(i = d_1, d_1 - 1, \ldots, k + 1)$  
   end for

2. **Step 2:** $a_{i}^{(d_1)} := c_{i}^{(d_1)}(i = 0, 1, \ldots, d_1)$  
   for $k = d_1 - 1$ to 0 by $-1$ do  
   $a_{i}^{(k)} := a_{i}^{(k+1)} - x_k a_{i+1}^{(k+1)}(i = k, k + 1, \ldots, d_1 - 1)$  
   end for

3. Return $a_i := a_i^{(0)}(i = 0, 1, \ldots, d_1)$.

In general, we can compute the equation (2) after choosing $d_1 + 1$ distinct scalars $(x_{10}, x_{11}, \ldots, x_{1d_1})$ and $d_2 + 1$ distinct scalars $(x_{20}, x_{21}, \ldots, x_{2d_2})$, then obtain their corresponding exact values $(f_{00}, f_{01}, \ldots, f_{0d_2}, \ldots, f_{10}, f_{11}, \ldots, f_{1d_2}, \ldots, f_{d_10}, f_{d_11}, \ldots, f_{d_1d_2})$. However, in order to improve intermediate expression swell problem arising from symbolic computations and avoid big integer computation, we can get the approximate values of $f(x_1, x_2)$, denoted by $(\tilde{f}_{00}, \tilde{f}_{01}, \ldots, \tilde{f}_{0d_2}, \tilde{f}_{10}, \tilde{f}_{11}, \ldots, \tilde{f}_{1d_2}, \tilde{f}_{d_10}, \tilde{f}_{d_11}, \ldots, \tilde{f}_{d_1d_2})$.

Based on Algorithm [1] together with Lemma [3] we can obtain the approximate solution $\bar{a} = [\bar{a}_{ij}](i = 0, 1, \ldots, d_1; j = 0, 1, \ldots, d_2)$. So an approximate bivariate polynomial $\bar{f}(x_1, x_2) = \sum_{i, j} \bar{a}_{ij} x_1^i x_2^j$ is only produced. However, we usually need the exact results in practice. Next, our main task is to bound the error between approximate coefficients and exact values, and discuss the controlling error $\varepsilon$ in Algorithm [1]. The literature [15] gave a preliminary study of this problem. Here, we present a necessary condition on error controlling $\varepsilon$ in floating-point arithmetic. In Step 1 of Algorithm [1] it is the standard method for evaluating divided differences $c_k^{(k)} = f[x_0, x_1, \ldots, x_k]$. We consider the relation on the $f_{ij} - \tilde{f}_{ij}$ with $a_{ij} - \bar{a}_{ij}$ and the propagation of rounding errors in divided difference schemes. We have the following theorem to answer the above question.

**Lemma 4.** $c_i$ and $f_i$ are defined as in Algorithm [1] $\bar{c}_i$ and $\tilde{f}_i$ are their approximate values by approximate interpolation, $\lambda = \min\{|x_{2i} - x_{2j}| : i \neq j\}(0 < \lambda < 1)$. Then

$$|c_i - \bar{c}_i| \leq \left(\frac{2}{\lambda}\right)^{d_2} \max\{|f_i - \tilde{f}_i|\}.$$
Proof. From Algorithm 1, we observe that Step 1 is recurrences for $c_i^{(k+1)}$, $(k = 0, 1, \cdots, d_2 - 1, i = d_2, d_2 - 1, \cdots, k + 1)$, whose form is as follows:

$$c_i^{(d_2)} = \frac{1}{\lambda} (c_i^{(d_2-1)} - c_i^{(d_2-1)}) .$$

However, when we operate the floating-point arithmetic in Algorithm 1, which is recurrences for $\tilde{c}_i^{(k+1)}$, which form is as follows:

$$\tilde{c}_i^{(d_2)} = \frac{1}{\lambda} (\tilde{c}_i^{(d_2-1)} - \tilde{c}_i^{(d_2-1)}) .$$

Therefore,

$$|c_i^{(d_2)} - \tilde{c}_i^{(d_2)}| = \frac{1}{\lambda} |c_i^{(d_2-1)} - \tilde{c}_i^{(d_2-1)} + \tilde{c}_i^{(d_2-1)} - c_i^{(d_2-1)}| \leq \frac{1}{\lambda} (|c_i^{(d_2-1)} - \tilde{c}_i^{(d_2-1)}| + |c_i^{(d_2-1)} - \tilde{c}_i^{(d_2-1)}|) .$$

The bounds are defined by the following recurrences,

$$|c_i^{(d_2)} - \tilde{c}_i^{(d_2)}| \leq \frac{2}{\lambda} |c_i^{(d_2-1)} - \tilde{c}_i^{(d_2-1)}| \leq \cdots \leq (\frac{2}{\lambda})^{d_2} \max\{|f_i - \tilde{f}_i|\} .$$

This completes the proof of the lemma.

**Theorem 2.** Let $\varepsilon = \max\{|f_{ij} - \tilde{f}_{ij}|\}$, $\lambda = \min\{|x_{1i} - x_{1j}|, |x_{2i} - x_{2j}| : i \neq j\}$ ($0 < \lambda < 1$). Then

$$\max\{|a_{ij} - \tilde{a}_{ij}|\} \leq (\frac{2}{\lambda})^{d_1} (\frac{2}{\lambda})^{d_2} \varepsilon .$$

Proof. From equation (2), it holds that

$$V \text{vec}(\tilde{a} - a) = \text{vec}(F - F) ,$$

where $V = V_{x_1} \otimes V_{x_2}$. By Lemma 3, the above equation is equivalent to the following equation:

$$V_{x_2} (\tilde{a} - a) V_{x_1}^T = \tilde{F} - F .$$

Thus, it is equivalent to

$$V_{x_2} z = \tilde{F} - F \quad (6a)$$

$$V_{x_1} (\tilde{a} - a)^T = z^T \quad (6b)$$

where $z = [z_{ij}]$. Matrix equation (6a) is equivalent to

$$V_{x_2} z_{i} = \tilde{F}_{i}. - F_{i}, \quad i = 1, 2, \cdots, d_2 + 1 \quad (7)$$

where $z_{i}$ stands for the $i$-th column of $z$ and $F_{i}$ the $i$-th row of matrix $F$. 

8
From Lemma 4 and Algorithm 1 it holds that

$$\max_{j=0}^{d_2} |z_{ji}| < \left( \frac{2}{\lambda} \right)^{d_2} |f_i - \tilde{f}_i|, \text{ for every } i.$$  

Hence, we conclude that

$$\max_{i,j} |z_{ji}| < \left( \frac{2}{\lambda} \right)^{d_2} |f_i - \tilde{f}_i|.$$  

Let $\delta = \left( \frac{2}{\lambda} \right)^{d_2} |f_i - \tilde{f}_i|$, argue equation (6b) in the same technique as do above, we deduce that

$$\max_{i,j} |a_{ij} - \tilde{a}_{ij}| \leq \left( \frac{2}{\lambda} \right)^{d_1} \left( \frac{2}{\lambda} \right)^{d_2} \varepsilon.$$  

The proof is finished.

In order to avoid the difficulty of computations, we restrict our study to the coefficients of polynomial over $\mathbb{Z}$. So we need to solve the Vandermonde system and take the nearest integer to each component of the solution. The less degree of bounds on variables we obtain, the less the amount of computation is for obtaining approximate multivariate polynomial. Once an upper bound $d_1$ and $d_2$ are gotten, we choose $(d_1 + 1) \cdot (d_2 + 1)$ interpolate nodes and calculate

$$\varepsilon = 0.5 \left( \frac{\lambda}{2} \right)^{d_1 + d_2}.$$  

Then, compute the values $\tilde{f}_{ij} \approx f(x_{i1},x_{i2})$ for $i = 0, 1, \cdots, d_1, j = 0, 1, \cdots, d_2$ with an error less than $\varepsilon$. By interpolation method, we compute the approximate interpolation polynomial $\tilde{f}(x_1,x_2)$ with coefficient error less than 0.5.

As for the generalization of the algorithm to the case $v > 2$, we can say that the situation is completely analogous to the bivariate case. It comes down to solving the following system:

$$(V_{x_1} \otimes V_{x_2} \cdots \otimes V_{x_v}) \text{vec}(a) = \text{vec}(F).$$  

(9)

Of course, we can reduce the multivariate polynomial entries to bivariate ones on symbolic determinant. For more details refer to Section 2.3.

We can analyze the computational complexity of the derivation of above algorithm. For the analysis of floating-point arithmetic operations, the result is similar with the exact interpolation situation [15]. However, our method can enable the practical processing of symbolic computations in applications.

**Remark 3.** Our result is superior to the literature [15]. Here we make full use of advantage of arbitrary precision of floating-point arithmetic operations on modern computer and symbolic computation platform, such as Maple. In general, it seems as if at least some problems connected with Vandermonde systems, which traditionally have been considered too ill-conditioned to be attached, actually can be solved with good precision.
2.3. Reducing dimension method

As the variables increased, the storage of computations expands severely when calculated high order on symbolic determinant. The literature [26] is to map the multivariate problem into a univariate one. For the general case, the validity of the method is established by the following lemma.

**Lemma 5.** ([26]) In the polynomial ring $R[x_1, x_2, \ldots, x_v], v > 2$. The mapping:

$$\phi : R[x_1, x_2, \ldots, x_v] \rightarrow R[x_1]$$

$$\phi : x_i \mapsto x_1^{n_i}, 1 \leq i \leq v$$

where $n_v > n_{v-1} > \cdots > n_1 = 1$ is a homomorphism of rings.

Let $d_i(f(x_1, x_2, \ldots, x_v))$ be the highest degree of the polynomial $f(x_1, x_2, \ldots, x_v)$ in variable $x_i$. The following lemma relates the $n_i$ of the mapping to $d_i$ and establishes the validity of the inverse mapping.

**Lemma 6.** ([26]) Let $\psi$ be the homomorphism of free $R$-modules defined by:

$$\psi : R[x_1] \rightarrow R[x_1, x_2, \ldots, x_v]$$

$$\psi : x_1^k \mapsto \begin{cases} 1 & \text{if } k = 0, \\ \psi(x_1^q) \cdot x_1^r & \text{otherwise} \end{cases}$$

where $n_i+1 > k \geq n_i, k = q \cdot n_i + r, 0 \leq r < n_i$ and $n_v > \cdots > n_1 = 1$.

Then for all $f(x_1, x_2, \cdots, x_v) \in R[x_1, x_2, \ldots, x_v], \psi(\phi(f)) = f$, and for all $i$ if and only if

$$\sum_{j=1}^{i} d_j(f)n_j < n_{i+1}, 1 \leq i < v. \quad (10)$$

**Remark 4.** We apply the degree homomorphism method to reduce dimension for computing the determinant of a matrix with multivariate polynomial entries, which is distinguished from the practical fast polynomial multiplication [26]. We note that relation (10) satisfying is isomorphic to their univariate images. Thus any polynomial ring operation on entries of symbolic determinant, giving results in the determinant, will be preserved by the isomorphism. In this sense $\phi$ behaves like a ring isomorphism on the symbolic determinant of polynomials. Another way to view the mapping given in the theorems is:

$$\phi : x_i \mapsto x_1^{n_i}, 2 \leq i \leq v.$$

### 3. Derivation of the algorithm

The aim of this section is to describe a novel algorithm for estimating the degree of variables on symbolic determinant, and the degree homomorphism method for dimension reduction.
3.1. Description of algorithm

Algorithm 2 is to estimate the degree of variables on symbolic determinant by computation of the degree matrix, and Algorithm 3 and 4 are used to reduce dimension and lift variables.

Algorithm 2 (Estimating degree of variables algorithm)

Input: given the order $n$ of symbolic determinant $M$, list of variables $vars$;

Output: the exact or upper bounds on degree of variables.

Step 1: Select variable from $vars$ respectively, and repeat the following steps

1: \textbf{loop}
2: Obtain the degree matrix $\Omega = (\sigma_{ij})(1 \leq i, j \leq n)$ from $M$;
3: \textbf{if} order($\Omega$) = 2 \textbf{then}
4: \hspace{1em} maxdeg := max\{ $\sigma_{11} + \sigma_{22}, \sigma_{12} + \sigma_{21}$\}
5: \textbf{else}
6: \hspace{1em} for $i = 1$ to $n - 1$ do
7: \hspace{2.5em} for $j = 1$ to $n - 1$ do
8: \hspace{4em} temp := $\sigma_{i1} + \sigma_{1j}$
9: \hspace{4em} $\sigma_{ij}$ := max\{ $\sigma_{ij} + \sigma_{11}, temp$\}
10: \hspace{2.5em} end for
11: \hspace{1em} end for
12: \textbf{else}
13: \hspace{1em} for $i = 1$ to $n - 2$ do
14: \hspace{2.5em} maxdeg := maxdeg $-$ $\sigma_{11}$
15: \hspace{1em} end for
16: \hspace{1em} Return maxdeg
17: \textbf{end loop}

Theorem 3. Algorithm 2 works correctly as specified and its complexity is $O(n^2)$, where $n$ is the order of symbolic determinant.

Proof. Correctness of the algorithm follows from Theorem 1. The number of arithmetic operations required to execute $(n - 1) \times (n - 1)$ additions and simultaneous comparisons, and remain $n - 2$ substructions and one comparison by using degree matrix. Therefore, the total arithmetic operations are $n^2 - n$, that is $O(n^2)$. 
Algorithm 3 (Reducing dimension algorithm)

Input: given the order \( n \) of symbolic determinant \( M \), list of variables \( \text{vars} \);
Output: the order \( n \) of symbolic determinant \( M' \) with bivariate polynomial entries.

Step 1: Call Algorithm 2 to evaluate the bounds on degree of the variables in \( M \), denoted by \( d_i (1 \leq i \leq v) \).

Step 2: Reducing dimension
1: Divide the \( \text{vars} \) into the partitions: \([x_1, x_2, \ldots, x_t], [x_{t+1}, x_{t+2}, \ldots, x_v]\);
2: \textbf{for } \( i = t - 1 \) to 1 by \(-1\) \textbf{do} \( D_i := \prod_{j=i+1}^{t} (d_j + 1), x_i \leftarrow x_i^{D_i} \)
3: \textbf{end for}
4: \textbf{for } \( i = v - 1 \) to \( t + 1 \) by \(-1\) \textbf{do} \( D_i := \prod_{j=i+1}^{v} (d_j + 1), x_i \leftarrow x_i^{D_i} \)
5: \textbf{end for}

Step 3: Obtain the symbolic determinant \( M' \) on variables \( \text{vars} = [x_t, x_v] \);

Step 4: Return \( M' \).

REMARK 5. The beauty of this method is in a substitution trick. In Algorithm 3, \( t = \text{ceil}(\frac{n}{2}) \), where \( \text{ceil}(c) \) is a function which returns the smallest integer greater than or equal the number \( c \). We note that the lexicographic order \( x_v \succ x_{v-1} \succ \cdots \succ x_1 \) and divide the \( \text{vars} \) into two parts. Then the symbolic determinant can be translated into the entries with bivariate polynomial. It can be highly parallel computation when the variables are more than three.

REMARK 6. To sum up, based on Algorithm 2 to estimate bounds on degree of variables, Algorithm 3 to reduce dimension for multivariate case, Algorithm 1 to solve the Vandermonde coefficient matrix of linear equations with error controlling, and finally Algorithm 4 to lift variables for recovering the multivariate polynomial.

In this paper, we consider the general symbolic determinant, which is not sparse. Applying the substitutions to the matrix entries as described above and assuming the monomial exists in the determinant then the bivariate form of unknown polynomial is a highest degree of

\[
D = \sum_{i=1}^{\text{ceil}(\frac{n}{2})} (d_i \cdot \prod_{k=i+1}^{\text{ceil}(\frac{n}{2})} (d_k + 1)). \tag{11}
\]

While this upper bound on degree of variable is often much larger than needed, which is the worst case and thus is suitable to all cases.
Algorithm 4 (Lifting variables algorithm)

Input: given the set of monomial on $x_t, x_v$ in $L$;
Output: the polynomial with $x_1, x_2, \cdots, x_v$.

Step 1: Obtain the corresponding power set on $x_t, x_v$, respectively;

Step 2: Lifting variables

1: Call Algorithm 3 extract the power $D_i (1 \leq i \leq t-1, t+1 \leq i \leq v-1)$;

2: while $\text{nops}(L) \neq \text{NULL}$ do
3: $\text{temp} := \text{deg}(x_t)$
4: for $i = 1$ to $t-1$ by 1 do
5: $d_i := \text{iquo}(\text{temp}, D_i), \text{temp} := \text{irem}(\text{temp}, D_i)$
6: end for
7: $d_i := \text{temp}, \text{temp} := \text{deg}(x_v)$
8: for $i = t+1$ to $v-1$ by 1 do
9: $d_i := \text{iquo}(\text{temp}, D_i), \text{temp} := \text{irem}(\text{temp}, D_i)$
10: end for
11: $d_i := \text{temp}$
12: end while

Step 3: Obtain the new set of monomial $L'$ on $x_1, x_2, \cdots, x_v$;

Step 4: Return $L'$.

3.2. A small example in detail

Example 1. For convenience and space-saving purposes, we choose the symbolic determinant is three variables and order 2 as follows.

$$|M| = \begin{vmatrix}
5x_1^2 - 3x_1x_2 + 2x_3^2 & -9x_1 - 3x_2^2 - x_3^2 \\
-x_1 + x_2 + 3x_2x_3 & x_3 - 4x_2^2 
\end{vmatrix},$$

At first, based on Algorithm 2 we estimate the degree on $x_1, x_2, x_3$. For the variable $x_1$, we get

$$\Omega_1 = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}.$$ 

Then

$$\max\{2 + 0, 1 + 1\} = 2.$$ 

Therefore, the maximum degree of the variable $x_1$ is 2. As the same technique for $x_2, x_3$, we can get 3 and 3.

Call Algorithm 3 by substituting $x_1 = x_2^4$, we get

$$|M'| = \begin{vmatrix}
5x_2^8 - 3x_1^5 + 2x_3^2 & -9x_1^4 - 3x_2^2 - x_3^2 \\
-x_2^4 + x_2 + 3x_2x_3 & x_3 - 4x_2^2 
\end{vmatrix}.$$
Then, based on Algorithm \[2\] we again estimate the degree on \(x_2, x_3\) for \([10, 3]\).

Based on the derivation of algorithm in Section 3.1 and Algorithm [1] computing exact polynomial \(f(x_2, x_3)\) as follows: Choose the different floating-point interpolation nodes by using the distance between two points 0.5; \(\lambda = 0.5\), compute \(\varepsilon = 0.745 \times 10^{-8}\) from Theorem [2] Compute the approximate interpolate datum \(\tilde{f}_{ij}\) such that \(|f_{ij} - \tilde{f}_{ij}| < \varepsilon\). We get the following approximate bivariate polynomial:

\[
4.99995826234x_2^8x_3 - 20.00000018736x_2^{10} + 24.0010598569x_2^5x_3 + 12.0025760656x_2^7 + 2.00000000000x_3^3 - 8.000948268634x_2^2x_3^2 - 9.0004533172ox_3^5 + 9.01977448800x_2^5 - 3.00897542075x_2^6 + 3.02270681750x_3^3 + 9.00076124850x_2^3x_3 - 1.00207248277x_2^4x_3 + 1.00018098282x_2^3x_3^2 + 2.99986559933x_2x_3^3.
\]

Next, based on Algorithm [4] we lift the variables to obtain the following multivariate polynomial:

\[
4.99995826234x_2^7x_3 - 20.00000018736x_2^5x_3 + 24.0010598569x_1x_2x_3 + 12.0025760656x_2^3x_1 + 2.00000000000x_3^3 - 8.000948268634x_2^2x_3^2 - 9.0004533172ox_3^5 + 9.01977448800x_1x_2^5 - 3.00897542075x_2x_1^2 + 3.02270681750x_3^3 + 9.00076124850x_2^3x_3 - 1.00207248277x_1x_3^3 + 1.00018098282x_2x_3^2 + 2.99986559933x_2x_3^3.
\]

Finally, we easily recover the integer coefficients of above approximate polynomial to the nearest values as follows:

\[
5x_1^2x_3 - 20x_1^2x_2 + 24x_1x_2x_3 + 12x_1x_3 + 2x_3^2 - 8x_2^2x_2 - 9x_1^2 + 9x_1x_2 - 3x_2^2x_1 + 3x_2^2 + 9x_3^2x_3 - x_2^2x_1 + x_2^2x_2 + 3x_2x_2.
\]

### 4. Experimental results

Our algorithms are implemented in Maple. The following examples run in the same platform of Maple under Windows and AMD Athlon(tm) 2.70 Ghz, 2.00 GB of main memory(RAM). Figure 1, 2 presents the Time and RAM of computing for symbolic determinants to compare our method with symbolic method(\textit{det}, see Maple’s help), and exact interpolation method [5, 7, 8]. Figure 1 compared with time for computing, Figure 2 compared with memory consumption for computing, the \textit{order} of \(x\)-coordinate represents for the order of symbolic determinants.

From Figure 1, 2, we have the observations as follows:

- In general, the \textit{Time} and \textit{RAM} of algorithm \textit{det} are reasonable when the \textit{order} is less than nine, and two indicators increase very rapidly when the \textit{order} is to nine. However, two indicators of interpolation algorithm is steady growth.

- Compared with the exact interpolation method, the approximate interpolation algorithm has the obvious advantages on the \textit{Time} and \textit{RAM} when the \textit{order} is more than eight.
Figure 1: Computing time for symbolic determinant with different algorithms

Figure 2: Computing memory for symbolic determinant with different algorithms

Remark 7. All examples are randomly generated using the command of Maple. The symbolic method has the advantage of the low order or sparse symbolic determinants, such as expansion by minors, Gaussian elimination over the integers. However, a purely symbolic algorithm is powerless for many scientific computing problems, such as resultants computing, Jacobian determinants and some practical engineering always involving high-order symbolic determinants. Therefore, it is necessary to introduce numerical methods to improve intermediate expression swell problem arising from symbolic computations.

5. Conclusions

In this paper, we propose a hybrid symbolic-numerical method to compute the symbolic determinants. Meanwhile, we also present a novel approach for estimating
the bounds on degree of variables by the extended numerical determinant technique, and introduce the reducing dimension algorithm. Combined with these methods, our algorithm is more efficient than exact interpolation algorithm for computing the high order symbolic determinants. It can be applied in scientific computing and engineering fields, such as computing Jacobian determinants in particular. Thus we can take fully advantage of approximate methods to solve large scale symbolic computation problems.

6. Acknowledgement

This research was partly supported by China 973 Project NKBRPC-2011CB302402, the National Natural Science Foundation of China(Grant No. 61402537, 91118001, 11171053), the West Light Foundation of the Chinese Academy of Sciences, and China Postdoctoral Science Foundation funded project (Grant No. 2012M521692).

REFERENCES

[1] D. A. COX, J. LITTLE, D. O'SHEA, Using Algebraic Geometry, 2nd edn. Springer-Verlag, Berlin Heidelberg, 2005.
[2] S. DELVAUX, A. MARCO, J. J. MARTÍNEZ, et al., Fast computation of determinants of Bézout matrices and application to curve implicitization, Linear. Algebra. Appl. 430 1 (2009), 27–33.
[3] X. L. QIN, W. Y. WU, Y. FENG, et al., Structural analysis of high-index DAE for process simulation, Int. J. Model. Simul. Sci. Comput. 4 4 (2013), 1342008-1–16.
[4] E. HOROWITZ, S. SAHNI, On computing the Exact Determinant of Matrices with Polynomial Entries, J. ACM. 22 1(1975), 38–50.
[5] A. MARCO, J. J. MARTÍNEZ, Parallel computation of determinants of matrices with polynomial entries, J. Symb. Comput. 37 6(2004), 749–760.
[6] H. K. WIMMER, Polynomial matrices with Hermitian coefficients and a generalization of the Eneström-Kakeya theorem, Oper. Matrices, 2(2008), 443–454.
[7] Y. LI, An effective hybrid algorithm for computing symbolic determinants, Appl. Math. Comput. 215 7(2009), 2495–2501.
[8] L. Y. CHEN, Z. B. ZENG, Parallel computation of determinants of matrices with multivariate polynomial entries, Sci. China Inform. Sci. 56 11(2013), 1–16.
[9] D. KRESSNER, Bivariate matrix functions, Oper. Matrices 8 2(2014), 449–466.
[10] W. M. GENTLEMAN, S. C. JOHNSON, Analysis of Algorithms, A case Study: Determinants of Matrices with Polynomial Entries, ACM T. Math. Software, 2 (1976), 232–241.
[11] T. SASAKI, H. MURAO, Efficient Gaussian Elimination Method for Symbolic Determinants and Linear Systems, ACM T. Math. Software, 8 3(1982), 277–289.
[12] E. KALTOFEN, On Computing Determinants of Matrices Without Divisions, Proc. ISSAC 1992, ACM Press, New York, 342–349, 1992.
[13] J. D. LIPSON, Symbolic methods for the computer solution of linear equations with applications to flowgraphs, Proc. 1968 Summer Institute on Symbolic Mathematical Computation, 233–303, 1969.
[14] L. CHEN, W. EBERLY, E. KALTOFEN, et al., Efficient matrix preconditioners for black box linear algebra, Linear. Algebra. Appl. (343–344) (2002), 119–146.
[15] Y. FENG, X. L. QIN, J. Z. ZHANG, et al., Obtaining exact interpolation multivariate polynomial by approximation, J. Syst. Sci. Complex. 24 4(2011), 803–815.
[16] E. KALTOFEN, Z. F. YANG, On exact and approximate interpolation of sparse rational functions, Proc. ISSAC 2007, ACM Press, New York, 203–210, 2007.
[17] M. GIESBRECHT, G. LABAHN, W-S. LEE, Symbolic-numeric Sparse Interpolation of Multivariate Polynomials, Proc. ISSAC 2006, ACM Press, New York, 116–123, 2006.
[18] G. CHÉZE, A. GALLIGO, From an approximate to an exact absolute polynomial factorization, J. Symb. Comput. 41 (2006), 682–696.
[19] J. Z. Zhang, Y. Feng, Obtaining exact value by approximate computations, Sci. China Math. 50 (2007), 1361–1368.

[20] E. Kaltofen, B. Li, Z. F. Yang, et al., Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients, J. Symb. Comput. 47 (2012), 1–15.

[21] X. L. Qin, Y. Feng, J. W. Chen, et al., A complete algorithm to find exact minimal polynomial by approximations, Int. J. Comput. Math. 89 (2012), 2333–2344.

[22] E. Howard, Elementary Matrix Theory, Dover Publications, New York, 1966.

[23] C. D. Boor, Polynomial Interpolation in Several Variables, In Studies in Computer Science (in Honor of Samuel D. Conte), eds. R. DeMillo and J. R. Rice, Plenum Press, New York, 87–119, 1994.

[24] R. A. Horn, C. R. Johnson, Topics in Matrix Analysis, Cambridge University Press, Cambridge, 1991.

[25] A. Björck, V. Pereyra, Solution of Vandermonde Systems of Equations, Math. Comput. 24 (1970), 893–903.

[26] R. T. Moenck, Practical Fast Polynomial Multiplication, Proc. ACM Symposium on Symbolic and Algebraic Computation, 136–148, 1976.

Xiaolin Qin, Department of Mathematics, Sichuan University, Chengdu 610064, China
Chengdu Institute of Computer Applications, CAS, Chengdu 610041, China
e-mail: qinxl@casit.ac.cn

Zhi Sun, Chengdu Institute of Computer Applications, CAS, Chengdu 610041, China
e-mail: sunzhi0190163.com

Tuo Leng, School of Computer Engineering and Science, Shanghai University, Shanghai 200072, China
e-mail: lengtuo2004@hotmail.com

Yong Feng, Chongqing Institutes of Green and Intelligent Technology, CAS, Chongqing 401122, China
e-mail: yongfeng@casit.ac.cn

Corresponding Author: Xiaolin Qin