On the Computation Cost of the Determinant of $\Phi(\gamma, s) := \gamma^2 I - G^{\sim}(s)G(s)$ in the Guaranteed Accuracy $\mathcal{L}_\infty$-norm Computation

Masaaki Kanno

CUED/F-INFENG/TR.492
September 2004

On the Computation Cost of the Determinant of $\Phi(\gamma, s) := \gamma^2 I - G^{\sim}(s)G(s)$ in the Guaranteed Accuracy $\mathcal{L}_\infty$-norm Computation

Masaaki Kanno
University of Cambridge
Department of Engineering
Trumpington Street
Cambridge CB2 1PZ
United Kingdom
E-mail: `mk303@eng.cam.ac.uk`

September 3, 2004

Abstract

This report is concerned with the computation cost of the determinant of a (bivariate) polynomial matrix required in the guaranteed accuracy $\mathcal{L}_\infty$-norm computation. The obtained computation cost is in terms of word operations, unlike most results available in the literature where the computation cost is provided in terms of arithmetic operations. The proposed method employs multivariate Lagrange interpolation and the computation cost in terms of word operations is shown to be polynomial in the dimensions of the system, the orders of transfer functions in the elements and the sizes of the coefficients of the polynomials.

1 Introduction

Several algorithms for the solution of systems and control problems with guaranteed accuracy have been developed in [9]. Those algorithms require exact rational (or multiprecision integer) arithmetic supported in computer algebra systems, as opposed to floating point arithmetic which is almost always used in numerical computation. In such cases computation cost should be expressed in terms of word operations to estimate the realistic growth of computation time, rather than in terms of arithmetic operations. One of the problems considered in [9] is the $\mathcal{L}_\infty$-norm computation. Potentially the most time-consuming part of the $\mathcal{L}_\infty$-norm computation algorithm is the computation of the determinant of a bivariate rational function matrix. This reduces to computing the determinant of a bivariate polynomial matrix after clearing the denominator of each matrix element. In this report the cost of this matrix determinant computation needed for the guaranteed accuracy $\mathcal{L}_\infty$-norm computation is analysed.

The determinant is regarded as a fundamental characteristic of a square matrix and fast computation algorithms and cost analyses of algorithms for polynomial matrices have been presented: in [6] two algorithms, Gaussian elimination and expansion by minors, are considered; in addition an algorithm based on evaluation and interpolation and a procedure which computes the characteristic polynomial are investigated in [8]; [7] relates (univariate) polynomial matrix multiplication and determinant computation; [10] presents an algorithm which is based on multivariate Lagrange interpolation and which is suitable for parallelization. Nevertheless most computation costs are provided in terms of arithmetic operations. An exception is [2], which deals with multivariate polynomial resultants, namely, not determinants of general matrices.

The method employed in this report uses multivariate Lagrange polynomial interpolation and thus is in line with the one used by [10], but the resulting computation cost is given in terms of word operations. In Section 2, some notation and preliminary results are presented. Section 3 briefly reviews the theorem given in [9] which is needed for the guaranteed accuracy $\mathcal{L}_\infty$-norm computation algorithm. Section 4 presents Horner's rule, which evaluates a polynomial at a particular value and also computes the quotient and remainder on division of a polynomial by a monic linear term, and derives its computation cost in terms of word operations. Section 5 reviews Lagrange interpolation and also derives its computation cost. Then, in Section 6, an algorithm of computing the determinant of a para-Hermitian matrix of bivariate polynomials (of a special form) is proposed and its computation cost is analysed. Based on the result in Section 6, the computation cost of the matrix determinant needed for the guaranteed accuracy $\mathcal{L}_\infty$-norm computation is estimated in Section 7. The main result derived in this report is summarized in Theorem 10. Section 8 presents a numerical example whose purpose is to show how the suggested algorithm works.

2 Notation and Preliminary Results

In this section some notation unusual in the control field but commonly used in the computer algebra field is firstly reviewed. Then some basic results used for the analysis of the computation cost and/or of the lengths of the outputs of algorithms are presented.

When considering the computation cost of an algorithm, it is customary not to obtain the exact cost but to find the 'rate of growth' with respect to the input. The cost is then expressed 'up to a constant factor'. The 'big Oh' notation is used for this purpose:

Definition 1 ([4, Definition 25.7, p. 684][5, Definition 25.7, p. 710])
(i) A partial function $f : \mathbb{N} \to \mathbb{R}$, that is, one that need not be defined for all $n \in \mathbb{N}$, is called eventually positive *if there is a constant $N \in \mathbb{N}$ such that $f(n)$ is defined and strictly positive for all $n \geq N$.*
(ii) Let $g : \mathbb{N} \to \mathbb{R}$ be eventually positive. Then, $O(g)$ is the set of all eventually positive functions $f : \mathbb{N} \to \mathbb{R}$ for which there exist $N, c \in \mathbb{N}$ such that $f(n)$ and $g(n)$ are defined and $f(n) \leq cg(n)$ for all $n \geq N$.

In some situations the presence of many logarithmic factors is cumbersome. The 'soft Oh' notation is used in those cases in order to simplify the expression (but not to lose essential information):

Definition 2 ([4, Definition 25.8, p. 685][5, Definition 25.8, p. 711]) *Let $f, g : \mathbb{N} \to \mathbb{R}$ be eventually positive. Then we write $f \in O^{\sim}(g)$ if there are constants $N, c \in \mathbb{N}$ such that $f(n) \leq g(n)(\log_2(3 + g(n)))^c$ for all $n \geq N$.*

With this notation, $n \log n \operatorname{loglog} n \in O^{\sim}(n)$. See [4, Section 25.7, pp. 684-685][5, Section 25.7, pp. 710-711] for some features of the notion.

Addition of two integers obviously requires one integer addition but, when an (exact) addition is carried out on a computer, the computation time (or cost) depends on how large these numbers are. In general the actual computation cost of an arithmetic operation is affected by the 'sizes' of the numbers involved as well as the arithmetic operation itself. The size of an integer a is measured by its length $\lambda(a)$:

$$\lambda(a) \quad := \quad \begin{cases} 0 & \text{if } \lambda = 0 \,, \\ \lfloor \log_{2^{64}} |a| \rfloor + 1 & \text{if } \lambda \neq 0 \,, \end{cases}$$

where the floor function $\lfloor x \rfloor$ gives the largest integer less than or equal to x. This reflects the word number required to store the integer. The base 2^{64} only affects the constant factor. Thus this is not important for computation cost analysis and we do not explicitly specify it from now on. We can see that $\lambda(a) \in O(\log |a|)$ (if $a \neq 0$).

In the case of polynomial computations, as well as the degrees of the polynomials, the lengths of the coefficients have an effect on the computation cost. We measure the size of the coefficients by the max-norm of a polynomial: if $f = \sum_{0\le i\le n} a_i x^i \in \mathbb{Z}[x]$, then its max-norm is defined as $\| f \|_\infty := \max_{0\le i\le n} |a_i|$. In this report we further extend the definition for bivariate polynomials, that is, if $f = \sum_{i,j} a_{ij} x^i y^j \in \mathbb{Z}[x, y]$, then $\| f \|_\infty := \max_{i,j} |a_{ij}|$.

Let $f, g \in \mathbb{Z}[x]$ and suppose that $\| f \|_\infty \le a$, $\|g\|_\infty \le b$, $\deg f \le c$ and $\deg g \le d$. Without loss of generality we can assume that $c \le d$. Then the following relationships are easy to derive:

$$\begin{aligned} \deg(fg) &\le c + d \,, \\ \| fg\|_\infty &\le ab(c + 1) \,. \end{aligned}$$

Using these we can deduce that, for f_i such that $\| f_i \|_\infty \le a$ and $\deg f_i \le c$,

$$\deg \prod_{1\le i\le n} f_i \le nc \,, \tag{1}$$

$$\left\| \prod_{1\le i\le n} f_i \right\|_\infty \le a^n (c + 1)^{n-1} \,. \tag{2}$$

Addition (and subtraction) of two integers whose lengths are at most n can be computed with $O(n)$ word operations. Also addition of two polynomials of degree up to n with coefficients in a ring uses $O(n)$ ring operations.

Compared to addition, multiplication needs more cost and multiplication appears everywhere in practical computations. The effort to reduce the computation cost of multiplication has been made. For two integers whose lengths are bounded by n (resp., two polynomials of degree up to n with coefficients in a ring), the so-called classical multiplication method requires at most $2n^2$ word operations (resp., ring operations) to multiply them. Some methods have been proposed to achieve asymptotically faster multiplication. The fastest method currently available seems to be the method of Schönhage & Strassen, which achieves $O(n \log n \operatorname{loglog} n)$ operations [4, Chapter 8, pp. 209-241][5, Chapter 8, pp. 219-251]. In the computation cost estimation of larger problems, in order to abstract from the cost of the underlying multiplication algorithm, the notation of multiplication time $\mathsf{M}(\cdot)$ has been introduced [4, Definition 8.26, p. 232][5, Definition 8.26, p. 242]: multiplication of two integers of lengths at most n (resp., two polynomials of degrees up to n with coefficients in a ring) requires at most $\mathsf{M}(n)$ word operations (resp., ring operations).

The computation cost of division is the same as that of multiplication up to a constant and it is $O\big(\mathsf{M}(n)\big)$ word operations (resp., ring operations) for integers of lengths at most n (resp., polynomials of degrees up to n) (see [4, Theorems 9.6 & 9.8, pp. 247-248][5, Theorems 9.6 & 9.8, pp. 257-258]).

In the following we will estimate the computation costs of the product of two factorials of integers and of operations on polynomials in terms of word operations. Here we note that, if a computation requires $O(f)$ ring operations and the computation cost of each ring operations in terms of word operations is in $O(g)$, then the total computation cost is $O(fg)$ word operations.

Lemma 1 *We can compute $(2n)!n!$ in $O\big(\mathsf{M}(n\log n)\log n\big)$ word operations. Further its length is bounded by $3n\log 2n$.*

Proof
Firstly consider the length. Since

$$\log(n!) \;\le\; \log(n^n) \;=\; n\log n$$

and

$$\log\big((2n!)\big) \;\le\; \log\big((2n)^{2n}\big) \;=\; 2n\log 2n\ ,$$

the length of $(2n)!n!$ is bounded by $3n\log 2n$. The computation of $(2n)!$ requires $O\big(\mathsf{M}(n\log n)\log n\big)$ word operations [4, Exercise 10.8, p. 292][5, Exercise 10.8, p. 304]. We note that $n!$ can be computed while $(2n)!$ is calculated and thus we can get $n!$ without extra effort. Multiplication of $(2n)!$ and $n!$ requires $\mathsf{M}(2n\log 2n)$ word operations. In total it takes $O\big(\mathsf{M}(n\log n)\log n\big)$ word operations to compute $(2n)!n!$.

$\square$

Now the computation cost of expanding a product of linear terms is analysed.

Lemma 2 *Let $p_n \in \mathbb{Z}[s]$ be defined as $p_n := \prod_{0\le i\le n}(s-i)$ (or $p_n := \prod_{1\le i\le n}(s-i)$). Then the length of $\|p_n\|_\infty$ is bounded by $(n-1)\log(n+1)$. Further, p_n can be computed in $O\big(\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\big)$ word operations.*

Proof
First it should be noted that $\prod_{0\le i\le n}(s-i) = s\prod_{1\le i\le n}(s-i)$, so, $\big\|\prod_{0\le i\le n}(s-i)\big\|_\infty = \big\|\prod_{1\le i\le n}(s-i)\big\|_\infty$. Also, after computing $\prod_{1\le i\le n}(s-i)$, to compute $\prod_{0\le i\le n}(s-i)$ can be done by shifting the coefficients, whose cost will be $O\big(n\log\big\|\prod_{1\le i\le n}(s-i)\big\|_\infty\big)$.

So, $p_n = \prod_{1\le i\le n}(s-i)$ is considered. It is shown that $\|p_n\|_\infty$ is bounded by $(n+1)^{n-1}$. By noting that $\|p_1\|_\infty = 1$, it is derived that

$$\begin{aligned}\|p_n\|_\infty &= \|(s-n)p_{n-1}\|_\infty = \|sp_{n-1}-np_{n-1}\|_\infty\\ &\le (n+1)\,\|p_{n-1}\|_\infty \le (n+1)n\,\|p_{n-2}\|_\infty \le \cdots\\ &\le (n+1)\cdots 3\,\|p_1\|_\infty = \frac{(n+1)!}{2}\,\|p_1\|_\infty \le (n+1)^{n-1}\ .\end{aligned}$$

Therefore the length $\log \|p_n\|_\infty$ is bounded by $(n-1)\log(n+1)$.

To compute p_n, at most $\mathsf{M}(n)\log n$ operations in $\mathbb{Z}$ are needed [4, Lemma 10.4, p. 281][5, Lemma 10.4, p. 293]. It uses $O\big(\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\big)$ word operations since $\log\|p_n\|_\infty \le (n-1)\log(n+1) \in O(n\log n)$ and all the integers appearing in the computation are bounded by this. This is over $n\log\|p_n\|_\infty \in O\big(n^2\log n\big)$ so it also holds true for $\prod_{0\le i\le n}(s-i)$, which establishes the claim.

$\square$

It is not difficult to see that

$$\Bigg\|\prod_{\substack{0\le i\le n\\ i\ne j}}(s-i)\Bigg\|_\infty \quad\le\quad (n+1)^{n-1}\;.$$

We need a similar but slightly different result later on.

Lemma 3 *Let $p_n \in \mathbb{Z}[s]$ be defined as $p_n := \prod_{0\le i\le n}(s-i^2)$ (or $p_n := \prod_{1\le i\le n}(s-i^2)$). Then the length of $\|p_n\|_\infty$ is bounded by $(n-1)\log(n^2+1)$. Further, p_n can be computed in $O\big(\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\big)$ word operations.*

Proof

Similar to the proof above, $p_n = \prod_{1\le i\le n}(s-i^2)$ is considered. It is shown that $\|p_n\|_\infty$ is bounded by $(n^2+1)^{n-1}$. Again, by noting that $\|p_1\|_\infty = 1$, it is derived that

$$\begin{aligned}
\|p_n\|_\infty &= \big\|(s-n^2)p_{n-1}\big\|_\infty = \big\|sp_{n-1} - n^2 p_{n-1}\big\|_\infty\\
&\le (n^2+1)\,\|p_{n-1}\|_\infty \le (n^2+1)\big((n-1)^2+1\big)\,\|p_{n-2}\|_\infty \le \cdots\\
&\le (n^2+1)\cdots(2^2+1)\,\|p_1\|_\infty = \prod_{2\le i\le n}(i^2+1)\,\|p_1\|_\infty \le (n^2+1)^{n-1}\;.
\end{aligned}$$

Therefore the length $\log\|p_n\|_\infty$ is bounded by $(n-1)\log(n^2+1) \in O(n\log n)$. The fact that this also bounds all the integers appearing in the computation further proves the computation cost in word operations.

$\square$

Similarly we can see that

$$\Bigg\|\prod_{\substack{0\le i\le n\\ i\ne j}}(s-i^2)\Bigg\|_\infty \quad\le\quad (n^2+1)^{n-1}\;.$$

Here some results of the computation of the determinant of a matrix of real numbers/integers are quoted.

Theorem 4 (Hadamard's inequality [4, Theorem 16.6, p. 451][5, Theorem 16.6, p. 465]) *Let $B \in \mathbb{R}^{n\times n}$ and $A \in \mathbb{R}$ such that all the entries of B are at most A in absolute value. Then,*

$$|\det B| \quad \leq \quad n^{\frac{n}{2}} A^n \ .$$

Theorem 5 ([4, Theorem 5.12, p. 104][5, Theorem 5.12, p. 111]) *The determinant of a matrix $B \in \mathbb{Z}^{n\times n}$ with all entries less than A in absolute value can be computed with*

$$O\Big(\big(n^4 \log(nA) + n^3 \log^2(nA)\big)\left(\log^2 n + (\operatorname{loglog} A)^2\right)\Big) \ \text{ or } \ O^{\sim}\big(n^4 \log A + n^3 \log^2 A\big)$$

word operations.

Remark: The same computation cost as Theorem 5 (up to logarithmic factors) is reported in [1].
$\diamond$

3 Guaranteed Accuracy $\mathcal{L}_\infty$-norm Computation

In this section the theorem on which the guaranteed accuracy $\mathcal{L}_\infty$-norm computation algorithm suggested in [9] is based is briefly reviewed. Let $G \in \mathcal{RL}_\infty$ and $\Phi_\gamma(s) := \gamma^2 I - G^{\sim}(s)G(s)$ where $G^{\sim}(s) := G^T(-s)$. (This notation and the 'soft Oh' notation may slightly be confusing, but it should be clear from the context.) Then, $\det \Phi_\gamma(s)$ is a (real) rational function in s^2, so we substitute x for s^2 in $\det \Phi_\gamma(s)$ and write as $g_\gamma(x)$, i.e., $g_\gamma(s^2) = \det \Phi_\gamma(s)$. Write $g_\gamma(x) = \frac{n_\gamma(x)}{d_\gamma(x)}$ where $n_\gamma(x)$ and $d_\gamma(x)$ are polynomials in x whose coefficients are polynomials in γ (γ^2, in fact) and, when seen as polynomials in x and γ, are coprime over $\mathbb{R}[x, \gamma]$. Further define

$$h_\gamma^s(x) \quad := \quad \frac{n_\gamma(x)}{\mathrm{GCD}\left(n_\gamma(x), \frac{\partial}{\partial x} n_\gamma(x)\right)} \tag{3}$$

(where the greatest common divisor (GCD) is that of the polynomials in x and γ). We then have the following result.

Theorem 6 *Suppose that $G \in \mathcal{RL}_\infty$ and let $h_\gamma^s(x)$ be defined as above. Then the $\mathcal{L}_\infty$-norm of G is one of the following quantities:*

i) $\overline{\sigma}\{G(0)\}$,
ii) $\overline{\sigma}\{G(j\infty)\}$,
iii) a real root of the discriminant of $h_\gamma^s(x)$ (with respect to x).

Moreover each of the above quantities is a (real) root of a real polynomial.

The computation of the determinant of $\Phi_\gamma(s)$ is potentially the most time-consuming part in the entire guaranteed accuracy $\mathcal{L}_\infty$-norm computation algorithm. In this report we analyse the computation cost of the determinant of a matrix of the form of $\Phi_\gamma(s)$.

In fact we investigate the computation cost of the determinant of a matrix which is, roughly speaking, obtained after clearing denominators in $\Phi_\gamma(s)$. So a matrix whose elements are polynomials in γ and s is dealt with. First we prove some prerequisite results in the following two sections.

4 Horner's Rule

There are two ways of representing a univariate polynomial of degree n on a computer [4, Sections 5.1-2, pp. 92-95][5, Sections 5.1-2, pp. 98-101]. One way is an obvious one and it uses a list of $(n+1)$ coefficients. The other way is to express it by its values at $(n+1)$ different points. To convert a polynomial in the former form into the latter form is just evaluation of the polynomial at $(n+1)$ points and it can be executed by using Horner's rule. When given its values at $(n+1)$ different points, the coefficients can be obtained by means of Lagrange interpolation. This section reviews Horner's rule and also estimates its computation cost in terms of word operations. Lagrange interpolation is treated in the next section.

Suppose a polynomial $f = \sum_{0\le i\le n} a_i x^i \in F[x]$ where F is a field. To evaluate the value at $u \in F$, Horner's rule

$$f(u) = \Big(\cdots\big((a_n u + a_{n-1})u + a_{n-2}\big)u + \cdots + a_1\Big)u + a_0$$

uses n multiplications and n additions in F [4, p. 93][5, p. 99].

Horner's rule in fact computes the quotient and the remainder on division of f by $x - u$ [4, Exercise 5.3, p. 124][5, Exercise 5.3, p. 131]. This can easily be seen from the following:

$$\begin{aligned}
f(x) &= a_n x^{n-1}(x-u) + (a_n u + a_{n-1})x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_0 \\
&= \left\{a_n x^{n-1} + (a_n u + a_{n-1})x^{n-2}\right\}(x-u) \\
&\quad + \big((a_n u + a_{n-1})u + a_{n-2}\big)x^{n-2} + \cdots + a_0 \\
&= \cdots \\
&= \Big\{a_n x^{n-1} + (a_n u + a_{n-1})x^{n-2} + \cdots \\
&\quad + \Big(\cdots\big((a_n u + a_{n-1})u + a_{n-2}\big)u + \cdots + a_1\Big)\Big\}(x-u) \\
&\quad + \Big(\cdots\big((a_n u + a_{n-1})u + a_{n-2}\big)u + \cdots + a_1\Big)u + a_0 \,.
\end{aligned}$$

That is, Horner's rule is equivalent to computing the coefficients of the quotient from the highest degree and the final value is the remainder.

It is trivially true that the computation of $f(u)$ for $f \in \mathbb{Z}[x]$, $\deg f = n$ and $u \in \mathbb{Z}$ by way of Horner's rule requires n multiplications and n additions in $\mathbb{Z}$. Now we estimate the computation cost in terms of word operations. We further suppose that $\|f\|_\infty \le A$ and $|u| \le B$ where $A, B \ge 1$. First we find a bound for the absolute values of integers that appear in the calculation. From the assumption,

$$\begin{aligned}
|a_n| &\le A\ , \\
|a_n u + a_{n-1}| &\le |a_n u| + |a_{n-1}| \le 2AB\ , \\
&\ \ \cdots \\
|f(u)| &= |a_n u^n + a_{n-1}u^{n-1} + \cdots + a_0| \\
&\le |a_n u^n| + |a_{n-1}u^{n-1}| + \cdots + |a_0| \le (n+1)AB^n\ . \qquad (4)
\end{aligned}$$

So the lengths of integers appearing in the calculation are bounded by $\big(\log(n+1) + \log A + n \log B\big)$ and thus one multiplication costs at most $\mathsf{M}\big(\log(n+1)+\log A+n\log B\big)$ word operations and one addition costs $O(\log n + \log A + n \log B)$ word operations. So we have proven the following result.

Lemma 7 *For $f \in \mathbb{Z}[x]$, $\deg f = n$, $\|f\|_\infty \le A$ and $u \in \mathbb{Z}$, $|u| \le B$ where $A, B \ge 1$, the computation of $f(u)$, or equivalently, the computation of the quotient and the remainder on division of f by $x - u$ (in $\mathbb{Z}[x]$), can be carried out by means of Horner's rule using $O\big(n\mathsf{M}(\log n + \log A + n \log B)\big)$ word operations.*

5 Lagrange Interpolation

Let $f \in F[x]$ be a polynomial of degree n where F is a field. Suppose that $u_0, u_1, \ldots, u_n \in F$ and that we know the values v_i of f at u_i: $f(u_i) = v_i \in F$. Then we can recover the coefficients of f by the following formula (the so-called Lagrange interpolation) [4, pp. 93-94][5, pp. 99-100]:

$$f = \sum_{0\le i\le n} v_i \prod_{\substack{0\le j\le n \\ j\ne i}} \frac{x - u_j}{u_i - u_j}\ . \qquad (5)$$

In the case of $f \in \mathbb{Z}[x]$, if $u_0, u_1, \ldots, u_n \in \mathbb{Z}$, then $f(u_i) = v_i \in \mathbb{Z}$. The above formula can also be used and the resulting f will be obtained in $\mathbb{Z}[x]$. It should be noted that, if the above formula is simply used, we have to leave $\mathbb{Z}[x]$ during the calculation since $v_i \prod_{0\le j\le n, j\ne i} \frac{x-u_j}{u_i-u_j}$ is not necessarily in $\mathbb{Z}[x]$.

However it may be possible to stay in $\mathbb{Z}[x]$. In terms of computation cost it is wise to use small integers, so, $u_i = i$, $0 \le i \le n$, are used. Then, (5) can be written as

$$f = \frac{\displaystyle\sum_{0\le i\le n} v_i\alpha_i \prod_{\substack{0\le j\le n\\ j\ne i}} (x-j)}{n!\left\lfloor \frac{n}{2}\right\rfloor!} \tag{6}$$

where

$$\alpha_i = (-1)^{n+i} \prod_{n-i+1\le j\le n} j \prod_{i+1\le j\le \lfloor \frac{n}{2}\rfloor} j \ , \ \ \alpha_{n-i} = (-1)^n\alpha_i \ , \ \ 0 \le i \le \left\lfloor \frac{n}{2}\right\rfloor \ .$$

Notice that $\alpha_i \in \mathbb{Z}$ and thus the numerator of (6) can be computed with operations on $\mathbb{Z}[x]$. Since $f \in \mathbb{Z}[x]$, all the coefficients in the numerator are divisible by the denominator in $\mathbb{Z}$ and what is required is merely $(n+1)$ (exact) integer divisions.

The same formula can also be used when $f \in R[x]$ where $R = \mathbb{Z}[y]$. The same u_i are used and $f(u_i) = f(i) = v_i \in R = \mathbb{Z}[y]$ in this case. Given v_i, the coefficients of f can be computed by way of (5) or (6). Similar to the above, (5) in general requires computation in $\mathbb{Q}[y]$ while in (6) the numerator can be computed in $R[x]$, the denominator is in $\mathbb{Z}$ and the division can be done exactly in $R[x]$.

We now estimate the computation cost of Lagrange interpolation when $f \in R[x]$ where $R = \mathbb{Z}[y]$.

Theorem 8 *Let $f \in R[x]$ where $R = \mathbb{Z}[y]$, $\deg_x f = n$, $f = \sum_{0\le i\le n} a_i x^i$, $a_i \in \mathbb{Z}[y]$ and $\deg_y a_i \le m$. Given $v_i = f(u_i) \in \mathbb{Z}[y]$, $0 \le i \le n$, where $u_i = i$, and a bound A for $\|v_i\|_\infty$ such that $\|v_i\|_\infty \le A$ for all i, Lagrange interpolation of recovering the coefficients of f (by means of formula (6)) uses $O\Big(mn^2\mathsf{M}\big(\log\max\{(n+1)^{\frac{3}{2}n-1}, A\}\big) + \mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\Big)$ word operations.*

Proof

Firstly consider the computation of the denominator. Lemma 1 states that $n!\lfloor\frac{n}{2}\rfloor!$ can be computed in $O\Big(\mathsf{M}\big(\frac{n}{2}\log\frac{n}{2}\big)\log\frac{n}{2}\Big)$ or $O\big(\mathsf{M}(n\log n)\log n\big)$ word operations. Also we can see that the length of $n!\lfloor\frac{n}{2}\rfloor!$ is bounded by $\frac{3}{2}n\log n$.

Then consider the numerator. There are $\big(\lfloor\frac{n}{2}\rfloor + 1\big)$ different α_i. For each α_i, there are $\lfloor\frac{n}{2}\rfloor$ integers of lengths at most $\log n$ to be multiplied. Therefore the computation of each α_i uses $O\Big(\mathsf{M}\big(\lfloor\frac{n}{2}\rfloor\log n\big)\log\lfloor\frac{n}{2}\rfloor\Big)$ or $O\big(\mathsf{M}(n\log n)\log n\big)$ word operations [4, Exercise 10.8, p. 292][5, Exercise 10.8, p. 304]. For all α_i, the number of word operations required is $O\big(n\mathsf{M}(n\log n)\log n\big)$. Also the length of α_i is bounded by $\lfloor\frac{n}{2}\rfloor\log n$.

To compute $p_i := \prod_{0\le j\le n, j\ne i}(x-j)$, we first compute $p := \prod_{0\le j\le n}(x-j)$ and then p_i can be computed as $p_i = p/(x-i)$. From Lemma 2, $O\big(\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\big)$ word operations are needed to compute p. This has to be done once. Then we can use Horner's rule to compute p_i and we can see that, for each p_i, $O\big(n\mathsf{M}(n\log n)\big)$ word operations are used (Lemmas 2 & 7), noting that $\log A$ and B in Lemma 7 are $(n-1)\log(n+1)$ and n here, respectively. There are $(n+1)$ different p_i, so for all p_i we need $O\big(n^2\mathsf{M}(n\log n)\big)$ word operations. So, in total, $O\big(n^2\mathsf{M}(n\log n)+\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\big)$ word operations are used.

Then consider $\alpha_i p_i$. Since α_i is in $\mathbb{Z}$ and p_i is a polynomial of degree n in $\mathbb{Z}[x]$, we need $(n+1)$ integer multiplications for each $\alpha_i p_i$. The length of α_i is at most $\lfloor \frac{n}{2}\rfloor \log n$ and, from the comment just below Lemma 2, $\log\|p_i\|_\infty \le (n-1)\log(n+1)$. So to compute one $\alpha_i p_i$ requires $(n+1)\mathsf{M}\big((n-1)\log(n+1)\big)$ or $O\big(n\mathsf{M}(n\log n)\big)$ word operations. There are $(n+1)$ different $\alpha_i p_i$ and, as a whole, we need $O\big(n^2\mathsf{M}(n\log n)\big)$ word operations. Furthermore,

$$\log\|\alpha_i p_i\|_\infty \;\le\; \left\lfloor\frac{n}{2}\right\rfloor\log n+(n-1)\log(n+1) \;\le\; \left(\frac{3}{2}n-1\right)\log(n+1)\,. \tag{7}$$

To compute each $v_i\alpha_i p_i$, we have to multiply a polynomial in $\mathbb{Z}[y]$ (namely, v_i) and a polynomial in $\mathbb{Z}[x]$ (namely, $\alpha_i p_i$). Since $\deg_y v_i \le m$ and $\deg_x \alpha_i p_i = n$, there are at most $(m+1)(n+1)$ multiplications in $\mathbb{Z}$ to be executed. Since $\|v_i\|_\infty \le A$ and (7), we need $O\Big(mn\mathsf{M}\big(\log\max\{(n+1)^{\frac{3}{2}n-1},\, A\}\big)\Big)$ word operations. To compute all $v_i\alpha_i p_i$, it has to be computed for $0\le i\le n$, so, $O\Big(mn^2\mathsf{M}\big(\log\max\{(n+1)^{\frac{3}{2}n-1},\, A\}\big)\Big)$ word operations are used.

Adding up all $v_i\alpha_i p_i$ needs at most $(m+1)n(n+1)$ additions in integers, since each $v_i\alpha_i p_i$ has at most $(m+1)(n+1)$ coefficients when seen as a polynomial in $\mathbb{Z}[x,y]$ and there are $(n+1)$ different $v_i\alpha_i p_i$. Also,

$$\begin{aligned}\log\Big\|\sum_{0\le i\le n} v_i\alpha_i p_i\Big\|_\infty &\le \log\big\{(n+1)\max_i\|v_i\alpha_i p_i\|_\infty\big\}\\ &\le \log(n+1)+\log A+\left(\frac{3}{2}n-1\right)\log(n+1)\\ &\le \frac{3}{2}n\log(n+1)+\log A\,.\end{aligned} \tag{8}$$

The cost of integer addition is linear in the length of the integers, so, as a whole, we need $O\big(mn^2(n\log n+\log A)\big)$ word operations.

Division of the numerator by the denominator requires $(m+1)(n+1)$ integer divisions. The bound (8) of the coefficients in the numerator is larger than the bound of the denominator ($\frac{3}{2}n\log n$), so the computation cost is determined by the length of the numerator and we need $O\big(mn\mathsf{M}(n\log n+\log A)\big)$ word operations.

Removing comparatively 'low' costs, the whole computation cost is estimated to be $O\Big(mn^2\mathsf{M}\big(\log\max\{(n+1)^{\frac{3}{2}n-1},\, A\}\big)+\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\Big)$ word operations.

□

Now consider the case where $f \in \mathbb{Z}[x]$ is in fact a polynomial in x^2, i.e., the coefficients of all the odd degrees are zero and only those of even degrees can be nonzero. Here we suppose that f is $2n$-th order. As in the above, we use $u_i = i$, $-n \leq i \leq n$, and write $f(u_i) = v_i \in \mathbb{Z}$. Then, by taking into account that $v_{-i} = f(u_{-i}) = f(u_i) = v_i$, the following formula can be obtained which is equivalent to (5):

$$f = \frac{\begin{aligned}&(-1)^n v_0 \prod_{n+1\leq j\leq 2n} j \prod_{1\leq j\leq n} (x^2 - j^2) \\ &\qquad + \sum_{1\leq i\leq n} (-1)^{n+i}\, 2v_i \prod_{n-i+1\leq j\leq n} j \prod_{n+i+1\leq j\leq 2n} j \prod_{\substack{0\leq j\leq n\\ j\neq i}} (x^2 - j^2)\end{aligned}}{(2n)!n!}. \tag{9}$$

Again this can be computed without leaving $\mathbb{Z}[x]$ (in fact, without leaving $\mathbb{Z}[x^2]$).

We now estimate the computation cost of Lagrange interpolation when $f \in \mathbb{Z}[x^2]$. As will be seen in Section 6, our determinant computation algorithm executes this Lagrange interpolation for several times for the same u_i (but for different sets of v_i). The following estimation is for such a case.

Theorem 9 *Let* $f_k \in \mathbb{Z}[x^2]$, $\deg_x f_k \leq 2n$ *(i.e.,* $\deg_{x^2} f_k \leq n$*) for* $0 \leq k \leq m$. *Given* $v_{i,k} = f_k(u_i) \in \mathbb{Z}$, $0 \leq i \leq n$, $0 \leq k \leq m$, *where* $u_i = i$, *and a bound* A *for* $|v_{i,k}|$ *such that* $|v_{i,k}| \leq A$ *for all* i, k, *Lagrange interpolation of recovering the coefficients of* f_k, $0 \leq k \leq m$, *(by means of formula (9)) uses* $O\Big(mn^2\mathsf{M}\big(\log\max\big\{2^n(n^2+1)^{\frac{3}{2}n-1}, A\big\}\big) + \mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\Big)$ *word operations.*

Proof

To find all f_k, Lagrange interpolation has to be carried out for $(m+1)$ times, but the only difference in (9) for different k is the values of $v_{i,k}$. Therefore, $(2n)!n!$, $\prod_{n+1\leq j\leq 2n} j$, $\prod_{n-i+1\leq j\leq n} j \prod_{n+i+1\leq j\leq 2n} j$ and $\prod_{0\leq j\leq n, j\neq i}(x^2 - j^2)$ have to be computed only once. We first consider the computation costs of these.

Consider the computation of the denominator. Lemma 1 states that $(2n)!n!$ can be computed in $O\big(\mathsf{M}(n\log n)\log n\big)$ word operations and that its length is bounded by $3n\log 2n$.

To compute $\prod_{n+1\leq j\leq 2n} j$ or $\prod_{n-i+1\leq j\leq n} j \prod_{n+i+1\leq j\leq 2n} j$ for a particular i, we have to multiply n integers of lengths at most $\log 2n$. So its computation cost is $O\big(\mathsf{M}(n\log 2n)\log n\big)$ or $O\big(\mathsf{M}(n\log n)\log n\big)$ word operations [4, Exercise 10.8, p. 292][5, Exercise 10.8, p. 304]. For all $\prod_{n+1\leq j\leq 2n} j$ and $\prod_{n-i+1\leq j\leq n} j \prod_{n+i+1\leq j\leq 2n} j$, $1 \leq i \leq n$, the number of word operations required is $O\big(n\mathsf{M}(n\log n)\log n\big)$. Also their lengths are bounded by $n\log 2n$.

To compute $p_i := \prod_{0\leq j\leq n, j\neq i}(x^2 - j^2)$, we first compute $p := \prod_{0\leq j\leq n}(x^2 - j^2)$ and then p_i can be computed as $p_i = p/(x^2 - i^2)$. From Lemma 3, $O\big(\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\big)$ word operations are needed to compute p. This has to be done once. Then we can use Horner's rule to compute p_i and we can see that, for each p_i, $O\big(n\mathsf{M}(n\log n)\big)$ word operations are used (Lemmas 3 & 7), noting that $\log A$ and B in Lemma 7 are $(n-1)\log(n^2+1)$ and n^2 here, respectively. There are $(n+1)$ different p_i, so for all p_i we need $O\big(n^2\mathsf{M}(n\log n)\big)$ word operations. So, in total, $O\big(n^2\mathsf{M}(n\log n)+\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\big)$ word operations are used.

Then consider the multiplication of $\prod_{n+1\le j\le 2n} j$ or $\prod_{n-i+1\le j\le n} j\prod_{n+i+1\le j\le 2n} j \in \mathbb{Z}$ and p_i. Since p_i is a polynomial of degree n in x^2, we need $(n+1)$ integer multiplications. The length of $\prod_{n+1\le j\le 2n} j$ or $\prod_{n-i+1\le j\le n} j\prod_{n+i+1\le j\le 2n} j$ is at most $n\log 2n$ and, from the comment just below Lemma 3, $\log\|p_i\|_\infty \le (n-1)\log(n^2+1)$. So to compute $\prod_{n+1\le j\le 2n} j\, p_0$ or each $\prod_{n-i+1\le j\le n} j\prod_{n+i+1\le j\le 2n} j\, p_i$ requires $(n+1)\mathsf{M}\big((n-1)\log(n^2+1)\big)$ or $O\big(n\mathsf{M}(n\log n)\big)$ word operations. This should be done for $0\le i\le n$ and, as a whole, we need $O\big(n^2\mathsf{M}(n\log n)\big)$ word operations. Furthermore, $\log\left\|\prod_{n+1\le j\le 2n} j\, p_0\right\|_\infty$ and $\log\left\|\prod_{n-i+1\le j\le n} j\prod_{n+i+1\le j\le 2n} j\, p_i\right\|_\infty$ are bounded by

$$n\log 2n+(n-1)\log(n^2+1) \quad\le\quad \left(\frac{3}{2}n-1\right)\log(n^2+1)+n\log 2\,. \tag{10}$$

Those computed above are common for all f_k and hence should be computed once. The following part is dependent on $v_{i,k}$ and thus should be computed for each f_k, i.e., should be repeated for $(m+1)$ times.

For a particular k, to compute $v_{0,k}\prod_{n+1\le j\le 2n} j\, p_0$ or $v_{i,k}\prod_{n-i+1\le j\le n} j\prod_{n+i+1\le j\le 2n} j\, p_i$ for a particular i, we have to multiply an integer and a polynomial of degree n in $\mathbb{Z}[x^2]$. There are at most $(n+1)$ multiplications in $\mathbb{Z}$ to be executed. Since $|v_i|\le A$ and (10), we need $O\Big(n\mathsf{M}\big(\log\max\{2^n(n^2+1)^{\frac{3}{2}n-1}, A\}\big)\Big)$ word operations. It has to be computed for all i, $0\le i\le n$, so, in total,

$$O\Big(n^2\mathsf{M}\big(\log\max\{2^n(n^2+1)^{\frac{3}{2}n-1}, A\}\big)\Big) \tag{11}$$

word operations are used. We note that multiplying an integer by 2 can be achieved by shifting one bit and its cost is negligible here.

Adding up all the terms in the numerator needs at most $n(n+1)$ additions in integers. Also the length of the max-norm of the numerator is bounded by

$$\begin{aligned}
&\log(n+1)+\log 2A+\left(\frac{3}{2}n-1\right)\log(n^2+1)+n\log 2\\
&\qquad\le\quad \frac{3}{2}n\log(n^2+1)+(n+1)\log 2+\log A \\
&\qquad\in\quad O(n\log n+\log A)\ .
\end{aligned} \tag{12}$$

The cost of integer addition is linear in the length of the integers, so, as a whole, we need

$$O\bigl(n^2(n\log n+\log A)\bigr) \tag{13}$$

word operations.

Division of the numerator by the denominator requires $(n+1)$ integer divisions. The bound (12) of the coefficients in the numerator and the bound of the denominator $(3n\log 2n)$ are both in $O(n\log n+\log A)$, so we need

$$O\bigl(n\mathsf{M}(n\log n+\log A)\bigr) \tag{14}$$

word operations.

Here it is repeated that the computation costs (11), (13) and (14) are for a particular f_k and they have to be multiplied by m when estimating the computation costs for all f_k.

Removing comparatively 'low' costs, the whole computation cost is estimated to be $O\Bigl(mn^2\mathsf{M}\bigl(\log\max\bigl\{2^n(n^2+1)^{\frac{3}{2}n-1}, A\bigr\}\bigr)+\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\Bigr)$ word operations.

□

6 Computation of the Determinant of a Para-Hermitian Matrix of a Special Form

Let $\Psi(\gamma,s)=(\psi_{ij})$ be in $\mathbb{Z}[\gamma,s]^{n\times n}$ and suppose that Ψ is para-Hermitian with respect to s, namely, $\Psi^T(\gamma,-s)=\Psi(\gamma,s)$. Further we assume that $\deg_s\psi_{ij}\le m$, that $\deg_\gamma\psi_{ii}=1$, $\deg_\gamma\psi_{ij}=0$ for $i\ne j$ and that $\|\psi_{ij}\|_\infty\le A$. In this section we will estimate the computation cost (in terms of word operations) to find the determinant of Ψ. Write $\chi(\gamma,s)=\det\Psi\in\mathbb{Z}[\gamma,s]$. First it should be pointed out that

$$\chi(\gamma,s) \;=\; \chi(\gamma,-s)$$

and hence χ is a polynomial in s^2. The degrees of χ with respect to γ and s are bounded by n and mn, respectively. In fact, χ is a polynomial in s^2 of degree at most $\lfloor\frac{mn}{2}\rfloor$.

Remark: In practice we may be able to have a smaller bound for the degree of χ with respect to s (or s^2) [4, Exercise 5.32, p. 127][5, Exercise 5.32, p. 134] and in the actual computation we should use that bound instead of $\lfloor\frac{mn}{2}\rfloor$. However, $\lfloor\frac{mn}{2}\rfloor$ can be a tight bound and it is therefore suitable for the purpose of the cost analysis. ◇

We propose to compute χ in the following way, using Lagrange interpolation twice:

1. Compute $\chi(\gamma_i,s_j)\in\mathbb{Z}$ for $\gamma_i=i$, $0\le i\le n$, $s_j=j$, $0\le j\le\lfloor\frac{mn}{2}\rfloor$.

2. For each i, $0 \le i \le n$, compute $\chi(\gamma_i, s) \in \mathbb{Z}[s]$ from $\chi(\gamma_i, s_j)$, $0 \le j \le \left\lfloor \frac{mn}{2} \right\rfloor$, using Lagrange interpolation.

3. Compute $\chi(\gamma, s) \in \mathbb{Z}[\gamma, s]$ from $\chi(\gamma_i, s)$, $0 \le i \le n$, again using Lagrange interpolation.

To estimate the computation cost of Step 1, we first point out that the computation of all $\chi(\gamma_i, s_j)$ requires evaluation of all the elements of Ψ at $\gamma = \gamma_i = i$, $0 \le i \le n$, $s = s_j = j$, $0 \le j \le \left\lfloor \frac{mn}{2} \right\rfloor$, and the computation of the determinants of resulting integer matrices. To evaluate all the elements of Ψ, we use the fact that Ψ can be written as

$$\Psi(\gamma, s) \quad = \quad \Psi_0(s) + \gamma \Psi_\Delta(s)$$

where Ψ_0 and Ψ_Δ are matrices whose elements are polynomials in s only and Ψ_Δ is diagonal. Consider evaluating $\Psi(\gamma_i, s_j)$, $0 \le i \le n$, for a particular s_j. Since s_j is fixed, $\Psi_0(s_j)$ and $\Psi_\Delta(s_j)$ have to be computed once each. Then the relationships

$$\begin{aligned} \Psi(0, s_j) &= \Psi_0(s_j) \ , \\ \Psi(\gamma_i, s_j) &= \Psi(\gamma_i - 1, s_j) + \Psi_\Delta(s_j) \ , \quad 1 \le i \le n \ , \end{aligned}$$

can be used, which only requires (integer) additions. To evaluate an element of $\Psi_0(s_j)$ or a diagonal element of $\Psi_\Delta(s_j)$, we can use Horner's rule and it can be computed in

$$O\Big(m\mathsf{M}\big(m(\log m + \log n) + \log A\big)\Big)$$

word operations, noting that m, A and $\left\lfloor \frac{mn}{2} \right\rfloor$ here correspond to 'n', 'A' and 'B' in Lemma 7, respectively. There are $(n^2 + n)$ elements in total, so the computation cost for evaluation of $\Psi_0(s_j)$ and $\Psi_\Delta(s_j)$ is done in

$$O\Big(mn^2\mathsf{M}\big(m(\log m + \log n) + \log A\big)\Big)$$

word operations. Moreover the length of each element is bounded by

$$m \log \left\lfloor \frac{mn}{2} \right\rfloor + \log(m+1) + \log A \quad \in \quad O\big(m(\log m + \log n) + \log A\big)$$

(cf. (4)). To compute all $\Psi(\gamma_i, s_j)$, $0 \le i \le n$, (for a particular s_j,) we further need n^2 integer additions, noting that $\Psi_\Delta(s_j)$ is diagonal. The lengths of integers appearing during the calculation are bounded by

$$m \log \left\lfloor \frac{mn}{2} \right\rfloor + \log(m+1) + \log n + \log A \quad \in \quad O\big(m(\log m + \log n) + \log A\big) \ ,$$

so the computation cost is

$$O\big(mn^2(\log m + \log n) + n^2 \log A\big)$$

word operations.

Therefore evaluating $\Psi(\gamma_i, s_j)$, $0 \le \gamma_i \le n$, for a particular s_j requires

$$O\Big(mn^2 \mathsf{M}\big(m(\log m + \log n) + \log A\big)\Big)$$

word operations and, for all s_j, $0 \le s_j \le \lfloor \frac{mn}{2} \rfloor$,

$$O\Big(m^2n^3 \mathsf{M}\big(m(\log m + \log n) + \log A\big)\Big)$$

or

$$O^{\sim}\big(m^2n^3 \mathsf{M}(m + \log A)\big)$$

word operations are needed.

Now consider the determinant of $\Psi(\gamma_i, s_j)$. When evaluated at $\gamma = \gamma_i = i$, $0 \le i \le n$, and $s = s_j = j$, $0 \le j \le \lfloor \frac{mn}{2} \rfloor$, the absolute value of ψ_{ij} is bounded by

$$A\Big\{\Big(\Big\lfloor \frac{mn}{2} \Big\rfloor\Big)^m + \Big(\Big\lfloor \frac{mn}{2} \Big\rfloor\Big)^{m-1} + \cdots + 1\Big\}(n+1) \quad \le \quad A(m+1)(n+1)\Big(\Big\lfloor \frac{mn}{2} \Big\rfloor\Big)^m \ . \tag{15}$$

By using this bound and Theorem 5, the computation cost of a particular $\chi(\gamma_i, s_j) = \det \Psi(\gamma_i, s_j)$ is estimated to be

$$O^{\sim}\big(n^4(m + \log A) + n^3(m + \log A)^2\big) \ ,$$

word operations, or equivalently,

$$O^{\sim}\big(m^2n^3 + mn^4 + n^4 \log A + n^3 \log^2 A\big)$$

word operations. Since there are $(n+1)\big(\lfloor \frac{mn}{2} \rfloor + 1\big)$ different $\Psi(\gamma_i, s_j)$, in total,

$$O^{\sim}\big(m^3n^5 + m^2n^6 + mn^6 \log A + mn^5 \log^2 A\big)$$

word operations are needed. Furthermore a bound for $|\chi(\gamma_i, s_j)|$ can be found from Hadamard's inequality (Theorem 4) and (15):

$$|\chi(\gamma_i, s_j)| \quad \le \quad n^{\frac{n}{2}} \Big\{A(m+1)(n+1)\Big(\Big\lfloor \frac{mn}{2} \Big\rfloor\Big)^m\Big\}^n \ . \tag{16}$$

Theorem 9 can be used to estimate the computation cost in Step 2. A bound for $|\chi(\gamma_i, s_j)|$ has already been found in (16), which is 'A' in Theorem 9. Also, 'm' and 'n' in Theorem 9 should be replaced by n and $\lfloor \frac{mn}{2} \rfloor$, respectively. The two arguments of the max function are

$$2^{\lfloor \frac{mn}{2} \rfloor} \left\{ \left(\left\lfloor \frac{mn}{2} \right\rfloor \right)^2 + 1 \right\}^{\frac{3}{2} \lfloor \frac{mn}{2} \rfloor - 1}$$

and the right hand side of (16). It is not obvious which is the bigger, but, taking their logarithms:

$$\begin{aligned}
&\left(\frac{3}{2} \left\lfloor \frac{mn}{2} \right\rfloor - 1 \right) \log \left\{ \left(\left\lfloor \frac{mn}{2} \right\rfloor \right)^2 + 1 \right\} + \left\lfloor \frac{mn}{2} \right\rfloor \log 2 \\
&\qquad\qquad\qquad \in \ O\big(mn(\log m + \log n)\big) , \\
&\frac{n}{2} \log n + n \left(\log A + \log(m+1) + \log(n+1) + m \log \left\lfloor \frac{mn}{2} \right\rfloor \right) \\
&\qquad\qquad\qquad \in \ O\big(mn(\log m + \log n) + n \log A\big) ,
\end{aligned}$$

we can conclude that

$$\begin{aligned}
\log \max &\left\{ 2^{\lfloor \frac{mn}{2} \rfloor} \left\{ \left(\left\lfloor \frac{mn}{2} \right\rfloor \right)^2 + 1 \right\}^{\frac{3}{2} \lfloor \frac{mn}{2} \rfloor - 1}, n^{\frac{n}{2}} \left\{ A(m+1)(n+1) \left(\left\lfloor \frac{mn}{2} \right\rfloor \right)^m \right\}^n \right\} \\
&\qquad\qquad\qquad \in \ O\big(mn(\log m + \log n) + n \log A\big) .
\end{aligned}$$

As a result the computation cost of Step 2 is

$$\begin{aligned}
O\Big(& m^2 n^3 \mathsf{M}\big(mn(\log m + \log n) + n \log A\big) \\
& + \mathsf{M}\big(mn(\log m + \log n)\big) \mathsf{M}(mn) \, (\log m + \log n) \Big)
\end{aligned}$$

word operations, or

$$O^{\sim}\big(m^2 n^3 \mathsf{M}(mn + n \log A) + \mathsf{M}(mn)^2 \big)$$

word operations. Furthermore the lengths of the max-norms of the obtained $\chi(\gamma_i, s)$ can be bounded by (12), and hence we get

$$\begin{aligned}
&\frac{3}{2} \left\lfloor \frac{mn}{2} \right\rfloor \log \left\{ \left(\left\lfloor \frac{mn}{2} \right\rfloor \right)^2 + 1 \right\} + \frac{n}{2} \log n \\
&+ n \left(\log A + \log(m+1) + \log(n+1) + m \log \left\lfloor \frac{mn}{2} \right\rfloor \right) + \left(\left\lfloor \frac{mn}{2} + 1 \right\rfloor \right) \log 2 \\
&\qquad\qquad\qquad \in \ O\big(mn(\log m + \log n) + n \log A\big) . \qquad (17)
\end{aligned}$$

The estimation of the computation cost of Step 3 uses Theorem 8. We have already found a bound for $\log \|\chi(\gamma_i, s)\|_\infty$ in (17), which is the logarithm of 'A' in Theorem 8. Also note that

Step	Cost
Step 1	$O^{\sim}\big(m^2n^3\mathsf{M}(m+\log A)+m^3n^5+m^2n^6+mn^6\log A+mn^5\log^2 A\big)$
Step 2	$O^{\sim}\big(m^2n^3\mathsf{M}(mn+n\log A)+\mathsf{M}(mn)^2\big)$
Step 3	$O^{\sim}\big(mn^3\mathsf{M}(mn+n\log A)+\mathsf{M}(n)^2\big)$

Table 1: Computation Costs of Steps in Determinant Computation (1)

Step	Cost
Step 1	$O^{\sim}\big(m^4n^3+m^3n^5+m^2n^6+mn^6\log A+(m^2n^3+mn^5)\log^2 A\big)$
Step 2	$O^{\sim}\big(m^4n^5+m^2n^5\log^2 A\big)$
Step 3	$O^{\sim}\big(m^3n^5+mn^5\log^2 A\big)$
Total	$O^{\sim}\big(m^4n^5+m^2n^6+mn^6\log A+m^2n^5\log^2 A\big)$

Table 2: Computation Costs of Steps in Determinant Computation (2) (when $\mathsf{M}(n)=2n^2$: Classical method)

'm' and 'n' in Theorem 8 are $\lfloor \frac{mn}{2} \rfloor$ and n here, respectively. If we take the logarithms of the two arguments of the max function, it is seen that they are in

$$O(n\log n)$$

and

$$O\big(mn(\log m+\log n)+n\log A\big)\ ,$$

respectively. We can conclude that Step 3 uses

$$O\Big(mn^3\mathsf{M}\big(mn(\log m+\log n)+n\log A\big)+\mathsf{M}(n\log n)\,\mathsf{M}(n)\log n\Big)$$

word operations, or

$$O^{\sim}\big(mn^3\mathsf{M}(mn+n\log A)+\mathsf{M}(n)^2\big)$$

word operations.

The computation costs of those steps are summarized in Table 1. Also the computation cost when $\mathsf{M}(n)=2n^2$ and $\mathsf{M}(n)\in O(n\log n\,\mathrm{loglog}\,n)\in O^{\sim}(n)$ are shown in Table 2 and Table 3, respectively. When the classical multiplication method is used, the most (asymptotically) costly steps are the first step (for large n) and the first and second steps (for large m). For large A, all

Step	Cost
Step 1	$O^{\sim}\big(m^3n^5 + m^2n^6 + (m^2n^3 + mn^6)\log A + mn^5 \log^2 A\big)$
Step 2	$O^{\sim}\big(m^3n^4 + m^2n^4 \log A\big)$
Step 3	$O^{\sim}\big(m^2n^4 + mn^4 \log A\big)$
Total	$O^{\sim}\big(m^3n^5 + m^2n^6 + (m^2n^4 + mn^6)\log A + mn^5 \log^2 A\big)$

Table 3: Computation Costs of Steps in Determinant Computation (3) (when $\mathsf{M}(n) \in O(n \log n \operatorname{loglog} n) \in O^{\sim}(n)$: Schönhage & Strassen method)

the steps can be expensive. The total cost is quartic in m, sextic in n and quadratic in $\log A$. In the case of the Schönhage & Strassen method, Step 1 is the most expensive one (also Step 2 for large m) and the total cost is cubic in m, sextic in n and quadratic in $\log A$.

Remark: A similar estimate will result if Lagrange interpolations are done first to get $\chi(\gamma, s_j)$ and then to get $\chi(\gamma, s)$ in the method proposed at the beginning of this section. $\diamond$

Even though the proposed method is not claimed to be the most efficient method, it is guessed that the method can be efficient in practice. Although the presentation of the method relies on polynomial manipulation, actual implementation may only have to keep the values of coefficients and only operations on numbers may be required, just as in [10]. Therefore direct polynomial manipulation (or symbolic operations), which is expensive in practice, can be avoided. Furthermore the method is suitable for parallelization since it is in line with the one proposed in [10]. If one can prepare a number of processors, the computing time can greatly be reduced.

There is a remark in [8, p. 44] regarding the determinant computation: '[...] Thus we can already surmise that though the evaluation-interpolation method may work quite well on completely dense polynomials of reasonably small degree and number of variables, other methods will become superior as the matrix's polynomials become more sparse.' It can be true in general (their surmise is based on the number of arithmetic operations, however), but the determinant we consider here is in general a dense polynomial and therefore the interpolation approach seems sensible. There is another remark in [3, p. 103]: 'It seems that in the case of a matrix of polynomials in several variables, Cramer's method is clearly much faster than any method based on Gaussian elimination. For the case of a matrix of integers or of polynomials in one variable, Bareiss' method[1] seems to be the most efficient.' It is not clearly mentioned how costly those methods are. If Bareiss's method is efficient for matrices of univariate polynomials, then we

[1]A variation of Gaussian elimination.

may use Bareiss's method to compute $\chi(\gamma_i, s)$ and then carry out one Lagrange interpolation to calculate $\chi(\gamma, s)$.

7 Computation of $\det \Phi'(\gamma, s)$

Now we are in a position to estimate the computation cost of the determinant needed in the guaranteed accuracy $\mathcal{L}_\infty$-norm computation. When $G(s) \in \mathcal{R}\mathcal{L}_\infty$ is given and all the coefficients in $G(s)$ are rational numbers (or finite decimals), then, by clearing all the denominators (or decimal points), we can have all the elements of $G(s)$ in $\mathbb{Z}(s)$. So let $G(s) = \left(g_{ij}\right) \in \mathbb{Z}(s)^{m\times n}$ and suppose that $g_{ij}(s)$ are at most d-th order. Write $g_{ij}(s) = \frac{n_{ij}(s)}{d_{ij}(s)}$ where $n_{ij}(s), d_{ij}(s) \in \mathbb{Z}[s]$. Then, $\deg n_{ij}(s)$, $\deg d_{ij}(s) \leq d$. Further suppose that $\|n_{ij}\|_\infty, \|d_{ij}\|_\infty \leq A$.

Define

$$\Phi(\gamma, s) \ := \ \gamma^2 I - G^\sim(s)G(s) \ \in \ \mathbb{Z}(\gamma, s)^{n\times n} .$$

We actually consider the computation cost of the determinant of

$$\begin{aligned} \Phi'(\gamma, s) \ = \ \left(\phi'_{ij}\right) \ &:= \ T^\sim(s)\Phi(\gamma, s)T(s) \\ &= \ \gamma^2 T^\sim(s)T(s) - T^\sim(s)G^\sim(s)G(s)T(s) \end{aligned} \tag{18}$$

where

$$T(s) \ = \ \operatorname{diag}\left(\prod_{1\leq k\leq m} d_{k1}(s), \prod_{1\leq k\leq m} d_{k2}(s), \cdots, \prod_{1\leq k\leq m} d_{kn}(s)\right) \ \in \ \mathbb{Z}[s]^{n\times n} . \tag{19}$$

First we note that

$$\Phi'^T(\gamma, -s) \ = \ T^\sim(s)\Phi^T(\gamma, -s)T(s) \ = \ T^\sim(s)\Phi(\gamma, s)T(s) \ = \ \Phi'(\gamma, s)$$

and hence that $\Phi'(\gamma, s)$ is para-Hermitian with respect to s. Also, noting that $T^\sim(s)T(s)$ is diagonal, we can see only diagonal elements of $\Phi'(\gamma, s)$ are polynomials in γ (in fact, γ^2) and that $\deg_{\gamma^2} \phi'_{ii} = 1$ and $\deg_{\gamma^2} \phi'_{ij} = 0$ for $i \neq j$. Further the (i, j)-element of $G(s)T(s)$ can be written as

$$n_{ij}(s) \prod_{\substack{1\leq k\leq m \\ k\neq i}} d_{kj}(s) \ \in \ \mathbb{Z}[s] , \tag{20}$$

which implies that

$$\Phi'(\gamma, s) \ \in \ \mathbb{Z}[\gamma, s]^{n\times n}$$

and further that

$$\det \Phi'(\gamma, s) \quad \in \quad \mathbb{Z}[\gamma, s] \ .$$

Therefore the form of $\Phi'(\gamma, s)$ is the same as that of $\Psi(\gamma, s)$ in Section 6. (Note that γ^2 here corresponds to γ in Section 6.) It should be pointed out that, since

$$\det \Phi'(\gamma, s) \quad = \quad \det T^{\sim}(s) \cdot \det T(s) \cdot \det \Phi(\gamma, s)$$

and $\det T^{\sim}(s)$, $\det T(s) \in \mathbb{Z}[s]$, it can be concluded that $\det\ \Phi'(\gamma, s)$ is divisible by the numerator of $\det \Phi(\gamma, s)$ and moreover that the quotient is in $\mathbb{Z}[s]$. The 'extra' factor will be removed when computing $h^s_\gamma(x)$ in (3) and does not cause a problem for the $\mathcal{L}_\infty$-norm computation (except for the possible extra computation cost).

Remark: In fact it is sufficient to use

$$T(s) \quad = \quad \mathrm{diag}\Big(\mathrm{LCM}\big(d_{k1}(s)\big), \mathrm{LCM}\big(d_{k2}(s)\big), \cdots, \mathrm{LCM}\big(d_{kn}(s)\big)\Big) \ ,$$

instead of $T(s)$ in (19), to clear denominators in Φ. So in the actual computation the above $T(s)$ should be used. Nevertheless, $T(s)$ in (19) is the worst case of such $T(s)$ and it is suitable for the purpose of the cost analysis. $\diamond$

In order to use the result in Section 6, we need to find a bound for the degree of ϕ'_{ij} (with respect to s) and also a bound for $\left\|\phi'_{ij}\right\|_\infty$. Firstly we consider the diagonal elements of $T(s)$. From (1)-(2),

$$\deg_s \prod_{1 \le k \le m} d_{kj}(s) \quad \le \quad md$$

and

$$\left\| \prod_{1 \le k \le m} d_{kj}(s) \right\|_\infty \quad \le \quad A^m (d+1)^{m-1} \ .$$

Hence bounds for the degree and the moduli of the coefficients of each element of $T^{\sim}(s)T(s)$ are $2md$ and $(md+1)A^{2m}(d+1)^{2(m-1)}$, respectively. Similarly, from (20), for each element of $G(s)T(s)$, the degree is bounded by md and the moduli of the coefficients are bounded by $A^m(d+1)^{m-1}$. They further imply that the degree and the moduli of the coefficients of each element of $T^{\sim}(s)G^{\sim}(s)G(s)T(s)$ are bounded by $2md$ and $m \cdot A^m(d+1)^{m-1} \cdot A^m(d+1)^{m-1}(md+1) = m(md+1)A^{2m}(d+1)^{2(m-1)}$, respectively.

As a result a degree bound for ϕ'_{ij} is

$$2md \ \text{ in } \ s \ . \tag{21}$$

The moduli of the coefficients of ϕ'_{ij}, $\|\phi'_{ij}\|_\infty$, is bounded by

$$m(md+1)A^{2m}(d+1)^{2(m-1)}\ , \tag{22}$$

or its length $\log\|\phi'_{ij}\|_\infty$ is bounded by

$$\begin{aligned}\log m + \log(md+1) + 2m\log A + 2(m-1)\log(d+1)\\ \in \quad O\big(m(\log A + \log d)\big)\ .\end{aligned}$$

(Remember the definition of $\|\cdot\|_\infty$ for bivariate polynomials.)

Now we can use the result in Section 6. In Section 6, $\Psi(\gamma, s) = (\psi_{ij})$ is in $\mathbb{Z}[\gamma, s]^{n\times n}$, $\deg_s \psi_{ij} \le m$ and $\|\psi_{ij}\|_\infty \le A$. So the correspondence of $\Psi(\gamma, s)$ in Section 6 and $\Phi'(\gamma, s)$ in this section is as follows:

$$\begin{array}{rcl}\Psi(\gamma,s) \text{ in Section 6} & \longleftrightarrow & \Phi'(\gamma,s) \text{ in this section}\\ m & \longleftrightarrow & 2md\\ n & \longleftrightarrow & n\\ A & \longleftrightarrow & m(md+1)A^{2m}(d+1)^{2(m-1)}\\ \log A & \longleftrightarrow & O\big(m(\log A+\log d)\big)\end{array}$$

If we use the classical method for multiplication, i.e., $\mathsf{M}(n) = 2n^2$, the result in Table 2 can be used. Using the above correspondence, we can derive the following theorem, which is the main result of this report.

Theorem 10 *Let $G(s) = (g_{ij}) \in \mathbb{Z}(s)^{m\times n}$. Write $g_{ij}(s) = \frac{n_{ij}(s)}{d_{ij}(s)}$ where $n_{ij}(s), d_{ij}(s) \in \mathbb{Z}[s]$. Suppose that $\deg n_{ij}(s)$, $\deg d_{ij}(s) \le d$. Further suppose that $\|n_{ij}\|_\infty$, $\|d_{ij}\|_\infty \le A$. Then, when the classical method for multiplication is employed, the computation of the determinant of $\Phi'(\gamma, s)$ as is defined in (18)-(19) requires*

$$O^\sim\Big(m^2n^5d\left\{m^2d^3 + nd + n\log A + m^2d\log^2 A\right\}\Big) \tag{23}$$

word operations. Namely, the computation cost in terms of word operations is

4th order in m,

6th order in n,

4th order in d,

2nd order in $\log A$.

In the case of square $G(s)$, i.e., when $m = n$, the computation cost of $\det \Phi'(\gamma, s)$ *is estimated to be*

$$O^{\sim}\left(n^9 d^2 (d^2 + \log^2 A)\right)$$

word operations and this is

9th order in n,
4th order in d,
2nd order in $\log A$.

Remark: Since $\|G(s)\|_\infty = \|G^T(s)\|_\infty$, we can use $G^T(s)$ to compute the $\mathcal{L}_\infty$-norm of $G(s)$. When computing degree bound (21) and coefficient bound (22), we do not use the relationship between m and n (e.g., $m < n$, etc.), so we can just swap m and n to get the computation cost

$$O^{\sim}\left(m^5 n^2 d \left\{md + n^2 d^3 + m \log A + n^2 d \log^2 A\right\}\right)$$

word operations. Judging from the powers of m and n, if $G(s) \in \mathbb{Z}(s)^{m \times n}$ is a wide matrix (i.e., $m < n$), it is in general sensible to first transpose it and then compute the determinant. The first sentence of Theorem 10 should thus be read: *Let $G(s) = \left(g_{ij}\right) \in \mathbb{Z}(s)^{m \times n}$ where $m \geq n$ (if $G(s)$ is a wide matrix, use $G^T(s)$).* $\diamond$

8 Numerical Example

In this section the algorithm proposed in this report is demonstrated by a numerical example. The purpose is to show how the algorithm works, not to show the validity of the computation cost analysis.

We use the plant

$$G(s) = \begin{bmatrix} \frac{s+1}{s^2-4} & 0 \\ 1 & \frac{1}{s+3} \end{bmatrix}$$

χ		s				
		0	1	2	3	4
γ^2	0	1	0	-3	-8	-15
	1	-24	-9	12	-33	-264
	2	239	126	27	-58	-2529

Table 4: Values of χ at $\gamma^2 = \gamma_i^2 = i, s = s_j = j$

as an example. In this case,

$$\begin{aligned}
\Phi(\gamma, s) &= \gamma^2 I - G^{\sim}(s)G(s) \\
&= \begin{bmatrix} \gamma - 1 + \frac{s^2-1}{s^4-8s^2+16} & -\frac{1}{s+3} \\ \frac{1}{s-3} & \gamma + \frac{1}{s^2-9} \end{bmatrix}, \\
T(s) &= \begin{bmatrix} s^2-4 & 0 \\ 0 & s+3 \end{bmatrix}, \\
\Phi'(\gamma, s) &= T^{\sim}(s)\Phi(\gamma, s)T(s) \\
&= \begin{bmatrix} (s^4 - 8s^2 + 16)\gamma^2 - s^4 + 9s^2 - 17 & -s^2 + 4 \\ -s^2 + 4 & (-s^2 + 9)\gamma^2 - 1 \end{bmatrix}.
\end{aligned}$$

So, m and n (in Section 6) are $m = 4$ and $n = 2$, respectively. Write $\chi(\gamma, s) = \det \Phi'(\gamma, s)$. Then, $\deg_{\gamma^2} \chi = 2$ and $\deg_s \chi \le 8$ (or $\deg_{s^2} \chi \le 4$). The values that χ takes when evaluated at $\gamma^2 = \gamma_i^2 = i, 0 \le i \le 2, s = s_j = j, 0 \le j \le 4$ are shown in Table 4.

Now the first Lagrange interpolation is carried out to obtain $\chi(\gamma_i, s)$. We use formula (9) and, for $\gamma_0^2 = 0$,

$$\begin{aligned}
\chi(\gamma_0, s) &= \frac{\begin{array}{l} 1 \cdot 1680(s^2-1)(s^2-2^2)(s^2-3^2)(s^2-4^2) \\ \quad - 2 \cdot 0 \cdot 1344 \cdot s^2(s^2-2^2)(s^2-3^2)(s^2-4^2) \\ \quad + 2 \cdot (-3) \cdot 672 \cdot s^2(s^2-1)(s^2-3^2)(s^2-4^2) \\ \quad - 2 \cdot (-8) \cdot 192 \cdot s^2(s^2-1)(s^2-2^2)(s^2-4^2) \\ \quad + 2 \cdot (-15) \cdot 24 \cdot s^2(s^2-1)(s^2-2^2)(s^2-3^2) \end{array}}{8!4!} \\
&= \frac{\begin{array}{l} 1680(s^8 - 30s^6 + 273s^4 - 820s^2 + 576) \\ \quad - 4032(s^8 - 21s^6 + 84s^4 - 64s^2) \\ \quad + 3072(s^8 - 21s^6 + 84s^4 - 64s^2) \\ \quad - 720(s^8 - 14s^6 + 49s^4 - 36s^2) \end{array}}{967680} \\
&= \frac{-967680s^2 + 967680}{967680} = -s^2 + 1 .
\end{aligned}$$

Similarly we get

$$\begin{aligned}\chi(\gamma_1, s) &= -2s^4 + 17s^2 - 24\ ,\\ \chi(\gamma_2, s) &= -2s^6 + 30s^4 - 141s^2 + 239\ .\end{aligned}$$

The second Lagrange interpolation yields

$$\begin{aligned}\chi(\gamma, s) &= \frac{1\cdot\chi(\gamma_0, s)(\gamma^2-1)(\gamma^2-2) - 2\cdot\chi(\gamma_1, s)\gamma^2(\gamma^2-2) + 1\cdot\chi(\gamma_2, s)\gamma^2(\gamma^2-1)}{2!1!}\\ &\quad (-s^2+1)(\gamma^4-3\gamma^2+2)\\ &= \frac{-2(-2s^4+17s^2-24)(\gamma^4-2\gamma^2)+(-2s^6+30s^4-141s^2+239)(\gamma^4-\gamma^2)}{2}\\ &= \frac{(-2s^6+34s^4-176s^2+288)\gamma^4+(2s^6-38s^4+212s^2-338)\gamma^2-2s^2+2}{2}\\ &= (-s^6+17s^4-88s^2+144)\gamma^4+(s^6-19s^4+106s^2-169)\gamma^2-s^2+1\ .\end{aligned}$$

Of course this coincides with the result obtained from a direct calculation.

9 Conclusion

This report has shown that the matrix determinant computation used for the guaranteed accuracy $\mathcal{L}_\infty$-norm computation can be carried out in polynomial time in the dimensions of the system, the orders of transfer functions in the elements and the sizes of the coefficients of the polynomials. It should be emphasized that the time is measured in terms of word operations.

The computation cost is 6th order in the larger dimension of the system, or 9th order in the dimension in the case of a square system. Although it may be expensive, this is the cost required for the guaranteed computation.

References

[1] J. Abbott, M. Bronstein, and T. Mulders. Fast deterministic computation of determinants of dense matrices. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC '99*, pages 197–204, Vancouver, British Columbia, Canada, July 1999.

[2] G. E. Collins. The calculation of multivariate polynomial resultants. *Journal of the Association for Computing Machinery*, 18(4):515–532, October 1971.

[3] J. H. Davenport, Y. Siret, and E. Tournier. *Computer Algebra: Systems and Algorithms for Algebraic Computation*. Academic Press, London, 2nd edition, 1993.

[4] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, 1st edition, 1999.

[5] J. von zur Gathen and J. Gerhard. *Modern Computer Algebra*. Cambridge University Press, Cambridge, 2nd edition, 2003.

[6] W. M. Gentleman and S. C. Johnson. Analysis of algorithms, a case study: Determinants of matrices with polynomial entries. *ACM Transactions on Mathematical Software*, 2(3):232–241, September 1976.

[7] P. Giorgi, C.-P. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In *Proceedings of the International Symposium on Symbolic and Algebraic Computation, ISSAC 2003*, pages 135–142, Philadelphia, Pennsylvania, USA, August 2003.

[8] E. Horowitz and S. Sahni. On computing the exact determinant of matrices with polynomial entries. *Journal of the Association for Computing Machinery*, 22(1):38–50, January 1975.

[9] M. Kanno. *Guaranteed Accuracy Computations in Systems and Control*. PhD thesis, University of Cambridge, 2003.

[10] A. Marco and J.-J. Martínez. Parallel computation of determinants of matrices with polynomial entries. *Journal of Symbolic Computation*, 37(6):749–760, June 2004.