On Fast Matrix Inversion via Fast Matrix Multiplication

Zak Tonks
Department of Computer Science, University of Bath
z.p.tonks@bath.ac.uk
January 7, 2019

Abstract
Volker Strassen first suggested an algorithm [Str69] to multiply matrices with worst case running time less than the conventional $O(n^3)$ operations in 1969. He also presented a recursive algorithm with which to invert matrices, and calculate determinants using matrix multiplication. James R. Bunch & John E. Hopcroft improved upon this in 1974 [BH74] by providing modifications to the inversion algorithm in the case where principal submatrices were singular, amongst other improvements. We cover the case of multivariate polynomial matrix inversion, where it is noted that conventional methods that assume a field will experience major setbacks. Initially, there existed a presentation of a fraction free formulation of inversion via matrix multiplication along with motivations in [TDS17], however analysis of this presentation was rudimentary. We hence provide a discussion of the true complexities of this fraction free method arising from matrix multiplication, and arrive at its limitations.

1 Background
Throughout, we discuss matrices over a ring of polynomials, say $R[x_1, \ldots, x_m]$, where $m > 1$. Until stated otherwise, $A$ is a matrix of polynomials, and we assume $A$ to be square, of dimension $n = 2^m$ for some $m \in \mathbb{N}$. We will refer to this as being of “binary size”, and respectively matrices of “non binary size” are those that do not fit this description. In particular matrices of binary size can be divided up into four equal quadrants, and as such $A_{i,j}$, $i, j \in \{1, 2\}$ refers to the canonical quadrant unless stated otherwise. When we refer to the size of a matrix, we mean either of its dimensions (as every matrix traversed is square unless stated otherwise). Hence if we say “a matrix of half the size” we mean a matrix of size $\frac{n}{2} \times \frac{n}{2}$. If we say $A$ is of degree $d$, we mean that all its entries are polynomials of degree $d$, and when we refer to to the degree of a polynomial, we refer to its total degree in all variables, rather than any variable in particular. We assume the result of any multiplication or addition of two polynomials to be expanded, ie. we distribute multiplication over addition. We do not assume polynomial arithmetic to be unit cost, and note that the cost of multiplication of two multivariate polynomials to addition of the same pair should be high, and this ratio even increases with degree.
For the time being, we assume every entry of any one matrix to have exactly degree $d$,
\[\text{ie. “uniformly of degree }d\].

### 1.1 Fast Matrix Multiplication and Inversion

Strassen-like matrix multiplication has been of interest since Strassen’s first presentation in 1969. In particular these algorithms are able to provide methods to multiply matrices in $O(n^\omega)$ ring operations where $\omega < 3$. Strassen’s first $\omega$ was $\log_2 7 \approx 2.8$, arising from the fact that only 7 recursions on matrix multiplications of half the size were required as opposed to 8. The current best $\omega$ is $\approx 2.37$, via Le Gall [Le 14]. Many of these methods are known to be extremely inefficient compared to naive matrix multiplication for small matrix sizes, although this is usually for the floating point case. For the purposes of this paper, we are interested in any such algorithm where $\omega < 3$, and note that analysis for the ideal cross over point for recursion in the symbolic case for specific polynomials, between the naïve and Strassen–Winograd [Win71] algorithms was covered in [TDS17].

Strassen also introduced methodology for inversion of a matrix via matrix multiplication, arising from the following:

\[
A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} = \begin{pmatrix} I & 0 \\ A_{2,1}A_{1,1}^{-1} & I \end{pmatrix} \begin{pmatrix} A_{1,1} & 0 \\ 0 & \Delta \end{pmatrix} \begin{pmatrix} I & A_{1,1}^{-1}A_{1,2} \\ 0 & I \end{pmatrix}
\]

where $\Delta = A_{2,2} - A_{2,1}A_{1,1}^{-1}A_{1,2}$, and so

\[
A^{-1} = \begin{pmatrix} A_{1,1}^{-1} + A_{1,1}^{-1}A_{1,2}\Delta^{-1}A_{2,1}A_{1,1}^{-1} & -A_{1,1}^{-1}A_{1,2}\Delta^{-1} \\ -\Delta^{-1}A_{2,1}A_{1,1}^{-1} & \Delta^{-1} \end{pmatrix}
\]

Bunch & Hopcroft followed this with a completeness result in 1974 [BH74], which covered the case where an $A_{1,1}$ or $\Delta$ received during the algorithm is singular, while the original input matrix $A$ is non singular (and so inversion or taking a determinant should make sense).

Multivariate polynomial matrices appear, for example, in Gröbner basis calculations, where we must upper triangulize dense matrices of sparse polynomials. This is essentially the case we wish to cover, and we note that methods involving interpolation will work poorly here.

### 1.2 Fraction Free Inversion

However, Section 1.1 assumes a field, as inversion is analogous with fractions - the inverse of a matrix of polynomials is in general a matrix of rational functions. In performing these algorithms symbolically, this means we are forced to take GCDs (Greatest Common Divisors) of potentially large polynomials in order to simplify results of inversion. The worst case complexity of taking the GCD is polynomial in the degree of the polynomials.
and exponential in the number of variables $[\text{Bro71}]$. As far as we are concerned the number of variables may as well be fixed, but the degree of numerators and denominators will increase per recursion as a result of the matrix arithmetic performed in any one recursion.

Our original full presentation of a fraction free modification to Strassen’s inversion follows, as was given in $[\text{TDS17}]$, where we investigated the ideal cross over point between standard matrix multiplication and Strassen’s methodology in the purely symbolic case. We omit lines indicating that the $A_{i,j}$, $B_{i,j}$ are the canonical submatrices of $A$, $B$ respectively. The operators $+$ and $\cdot$ are the operators acting on $R^{n \times n} \times R^t$ essentially canonical as per, for example, the rational numbers.

**Algorithm 1** First fraction free formulation of Strassen inversion of a matrix $A$ over a ring $R$. The matrices $A_{1,1}$ and $\Delta$ are required to be non singular at every iteration.

**Input:** $(A, d_1) \in R^{n \times n} \times R^t$

**Output:** $(B, d_2) \in R^{n \times n} \times R^t$ such that $A \cdot B = B \cdot A = d_1d_2I$, and as such $B$ is $d_1\text{Adj}(A)$, and $d_2$ is $\text{Det}(A)$.

*begin algorithm* FFInversion($A, d_1$)

if $n = 1$

    $B \leftarrow (d_1)$, $d_2 \leftarrow A_{1,1}$

else

    $(A_{1,1}^{adj}, a_{1,1}^{adj}) \leftarrow \text{FFInversion}(A_{1,1}, 1)$

   $(\Delta, \delta) \leftarrow (A_{2,2}, 1) + (-A_{2,1}, 1) \cdot (A_{1,2}^{adj}, a_{1,2}^{adj}) \cdot (A_{1,1}, 1)$

   $(\Delta^{adj}, \delta^{adj}) \leftarrow \text{FFInversion}(\Delta, \delta)$

   $(\Lambda, \lambda) \leftarrow (\Delta^{adj}, \delta^{adj}) \cdot (A_{2,1}, 1) \cdot (A_{1,2}^{adj}, a_{1,2}^{adj})$

   $(B'_{1,1}, b_{1,1}) \leftarrow (A_{1,1}^{adj}, a_{1,1}^{adj}) \cdot [(I_{\frac{n}{2}} \times \frac{n}{2}, 1) + (A_{1,2}, 1) \cdot (\Lambda, \lambda)]$

   $(B'_{1,2}, b_{1,2}) \leftarrow (-A_{1,1}^{adj}, a_{1,1}^{adj}) \cdot (A_{1,2}, 1) \cdot (\Delta^{adj}, \delta^{adj})$

   $d_2 \leftarrow \text{Det}(A)$

   $d \leftarrow d_1d_2$

   $B_{1,1} \leftarrow \frac{d}{b_{1,1}} B'_{1,1}$

   $B_{1,2} \leftarrow \frac{d}{b_{1,2}} B'_{1,2}$

   $B_{2,1} \leftarrow \frac{d}{\delta^{adj}} \Lambda$

   $B_{2,2} \leftarrow \frac{d}{\Delta^{adj}} \Delta^{adj}$

*end if*

Note that Algorithm is the “direct translation” of Strassen’s inversion process. The main difference is the definition of $\Delta$, which in the classical case is $A_{2,2} = A_{2,1}A_{1,1}^{-1}A_{1,2} = A_{2,2} - \frac{A_{2,1}A_{1,1}^{adj}A_{1,2}}{a_{1,1}^{adj}}$, but here we multiply through by $a_{1,1}^{adj}$ (where $a_{1,1}^{adj}$ is actually the determinant of $A_{1,1}$) to obtain the fraction free object $a_{1,1}^{adj}A_{2,2} = A_{2,1}A_{1,1}^{adj}A_{1,2}$. When doing fraction free inversion, it makes most sense to talk about the pair $(\text{Adj}(A), \text{Det}(A))$, as
a result of the fact the intention was to produce inverses, and Det(A)A⁻¹ = Adj(A). As such when we discuss “fraction free inverse”, we will largely mean this pair, especially in the context of inverses via the presented methods.

Under this formulation, (1) becomes:

\[
A = \begin{pmatrix} I & 0 & 0 \\ A_{2,1}A_{1,1} & I \\ a_{adj} & 0 & I \end{pmatrix} \begin{pmatrix} A_{1,1} & 0 & 0 \\ 0 & \Delta \\ 0 & 0 & I \end{pmatrix} \begin{pmatrix} I & A_{adj} & 0 \\ a_{adj} & 0 & 0 \\ a_{adj} & 0 & I \end{pmatrix}
\]

(3)

where \( \Delta = a_{adj}A_{2,2} - A_{2,1}A_{1,1}A_{1,2}, \) \( a_{adj} = Det(A_{1,1}), \) and \( \delta_{adj} = Det(\Delta). \) Then (2) becomes:

\[
A_{adj} = Det(A) \begin{pmatrix} a_{adj} & 0 & 0 \\ a_{adj} & a_{adj} & 0 \\ \Delta & 0 & 0 \end{pmatrix} \begin{pmatrix} -\frac{a_{adj}A_{1,2}}{a_{adj}} & 0 & 0 \\ 0 & -\frac{\delta_{adj}}{a_{adj}} & 0 \\ 0 & 0 & \frac{\Delta_{adj}}{\delta_{adj}} \end{pmatrix}
\]

\[
= \begin{pmatrix} a_{adj}^{\frac{n}{2}-3} & (a_{adj}^3 - \delta_{adj}^2)A_{1,1}^2A_{1,2}A_{2,1} & -a_{adj}^{\frac{n}{2}-2}A_{1,1}A_{1,2}A_{2,1}A_{1,1} \\ 0 & -a_{adj}^{\frac{n}{2}-2}A_{1,1}A_{1,2}A_{2,1} & a_{adj}^{\frac{n}{2}-1} \Delta_{adj} \\ 0 & 0 & \Delta_{adj} \end{pmatrix}
\]

via the fact discussed in Section 1.2.1 immediately below.

1.2.1 Immediate Improvements

As postulated in [TDS17], we have that Det(A) arises naturally in the course of computation as \( \frac{\delta_{adj}}{(\hat{a}_{adj})^\frac{n}{2}-1}. \) Immediately, this saves on computations used to calculate Det(A), which could previously be obtained in \( O(n^\omega) \) operations via the fraction free formula \( Det(\Delta_{adj})Det(A_{1,1}^d) \), which lends itself to a recursive definition requiring adjugates of submatrices, and hence mutually dependent on Algorithm [1]. Removing the need to calculate this via any method removes a significant amount of work, but only up to a constant in terms of the \( O \)-complexity of the algorithm.

However, this formula actually does more - by substituting \( \frac{\delta_{adj}}{(\hat{a}_{adj})^\frac{n}{2}-1} \) for \( d_2 \) in all subsequent steps of Algorithm [1] and working out exact expressions for the pairs calculated via the new operators, we see that significant cancellations occur in calculating the \( B_{i,j} \), and in fact we can obtain much more concise expressions for these matrices and polynomials. In fact, the full definitions of the operators \( \cdot \) and \( + \) on pairs \( (M,c) \) in \( R^{n\times n} \times R^l \) begin to seem superfluous, and in fact we may as well calculate every object directly. Throughout the course of the next section, we see that even the \( b_{i,j} \) become redundant, as we require less exact divisions.

We may as well not recurse via this type of method all the way down to size 1. If \( A \) is a \( 2 \times 2 \) matrix, the well known method for computing the adjugate is a matter of
negating or swapping submatrices of $A$ (the cofactor matrix), and the determinant can be computed in 2 multiplications and an addition (of size 1 matrices ie. polynomials). In contrast, computing the equivalent pair at this dimension via Algorithm 1 can already be seen to require at least as many operations just in computing the relevant $\Delta$ and $\Lambda$ (4 multiplications and an addition).

With a view to these improvements, Algorithm 2 is a superior implementation that considers the above. Note that the claim that $d_2$ is $\text{Det}(A)$ is inductive on $n$.

Algorithm 2 Second fraction free formulation of Strassen inversion of a matrix $A$ over a ring $R$. The matrices $A_{1,1}$ and $\Delta$ are required to be non singular at every iteration.

**Input:** $(A, d_1) \in R^{n \times n} \times R$

**Output:** $(B, d_2) \in R^{n \times n} \times R$ such that $A \cdot B = B \cdot A = d_1 d_2 I$, and as such $B$ is $d_1 \text{Adj}(A)$, and $d_2$ is $\text{Det}(A)$.

**begin algorithm** $\text{FFInversion2}(A, d_1)$

if $n = 1$ then

$B \leftarrow (d_1), d_2 \leftarrow A_{1,1}$

else if $n = 2$ then

$B \leftarrow \begin{pmatrix} A_{2,2} & -A_{1,2} \\ -A_{2,1} & A_{1,1} \end{pmatrix}$, $d_2 \leftarrow A_{1,1} A_{2,2} - A_{1,2} A_{2,1}$

else

$(A_{1,1}^{adj}, a_{1,1}^{adj}) \leftarrow \text{FFInversion2}(A_{1,1,1})$

$\Delta \leftarrow a_{1,1}^{adj} A_{2,2} - A_{2,1} A_{1,1}^{adj}$

$(\Delta^{adj}, \delta^{adj}) \leftarrow \text{FFInversion2}(\Delta, a_{1,1}^{adj})$

$d_2 \leftarrow \frac{\delta^{adj}}{|a_{1,1}^{adj}|^{2/3}}$

$\Delta^{adj'} \leftarrow \frac{1}{|a_{1,1}^{adj}|^{2/3}} \Delta^{adj}$

$\Lambda \leftarrow \frac{1}{\delta^{adj}} \Delta^{adj'} A_{2,1} A_{1,1}^{adj}$

$\epsilon \leftarrow A_{1,1}^{adj} A_{1,2}$

$B_{1,1} \leftarrow \frac{d_1}{\delta^{adj}} (d_2 A_{1,1}^{adj} + \epsilon \Lambda)$

$B_{1,2} \leftarrow \frac{d_1}{\delta^{adj}} (\epsilon \Delta^{adj'})$

$B_{2,1} \leftarrow -d_1 \Lambda$

$B_{2,2} \leftarrow d_1 \Delta^{adj'}$

end if

2 Analysis of Algorithm 2

We consider Algorithm 2 from here onwards. In particular this means we recurse down to size 2, and thus have one less recursion level. So we never obtain $1 \times 1$ “matrices”, which is a somewhat trivial case that would have consequences with regards to “matrix content”, which will be defined in due course. Note substitution of $\text{Det}(A)$ has resulted
in cancellations, which results in fewer required divisions (especially on the $B_{i,j}$).

2.1 Degree Bloat in Intermediate Matrices

Note that if $A$ is a matrix of size $n$ and degree $d$, then $\text{Adj}(A)$ will be a matrix of degree $(n-1)d$, and $\text{Det}(A)$ will be a single polynomial of degree $nd$, as they are sums of $n-1$ and $n$ products respectively. Informally, this implies that working with adjugates means that we will introduce matrices of at least degree $O(nd)$, but we should avoid performing arithmetic on them, possibly by performing cancellations where possible. Indeed our fraction free formulation relies on the multiplication, and introducing multiplications of polynomials of degree anything other than $d$ invalidates any assumption that the cost of multiplication is fixed (and hence the ratio of cost of multiplication to addition, which was a large focus of our previous work).

First, note that if we say a matrix is degree $O(d)$, then we mean all its elements are polynomials of degree $O(d)$. Further, we must make rigorous a concept analogous to “content” for polynomials viewed recursively over their variables. For the following two definitions, assume $C$ to be the result of some arithmetic and/or inversion operations from the submatrices of an original matrix $A$, where $C$ is of a binary size possibly less than $n$, but more than 1 - it makes no sense to talk about the content of a matrix of size 1, as this would be trivial, as every element of the matrix is divisible by itself. This $A$ should be assumed to be “random” in some sense, or ideally “symbolic”, which will be defined in due course. For the purposes of the following Definition 1, when we say $C_{i,j}$, we mean the $i,j$th element of $C$, rather than the canonical submatrix that we usually mean throughout.

**Definition 1.** Define the content of a matrix to be the gcd of all its elements, analogous to the same concept for a polynomial. ie. $\text{content}_{\gcd}(C) = \gcd(C_{1,1}, \gcd(C_{1,2}, \gcd(\ldots, \gcd(C_{2,1}, \gcd(\ldots, \gcd(C_{n,n-1}, C_{n,n}))))))$)

**Definition 2.** Define the systematic matrix content, $\text{content}_{\text{sys}}(C)$ of a matrix to be the content that we can deduce the matrix to have. That is, if $C$ is a matrix that is the result of some arithmetic and inversion operations from some symbolic matrix $A$ (such that every entry of $A$ is a distinct symbol $a_{i,j}$), then $c$ is the systematic matrix content of $C$ if $c$ is the polynomial of largest degree such that every element of $C$ is divisible by $c$. In this way, the systematic matrix content of a symbolic matrix is equal to $\text{content}_{\gcd}(C)$.

The distinction between the above definitions is important. It is clear that $\text{content}_{\text{sys}}(C) \mid \text{content}_{\gcd}(C)$. When discussing deterministic algorithms, Definition 2 allows us to talk of factors that we know will always be there in any one intermediate matrix. Content in the context of Definition 1 somewhat implies that we must take pairwise GCDs of elements in $A$ to compute this object. This content may be more than we bargained for, and depending on the structure of the original matrix $A$, we may deduce “accidental” factors that wouldn’t be there for every starting matrix $A$.

In many scenarios, $\text{content}_{\text{sys}}(C) = \text{content}_{\gcd}(C)$. One situation where this is not the case is when $A$ had content to begin with, ie. $\text{degree(content}(A)) > 0$. Then
there is a discrepancy between $\text{content}_{sys}(A)$ and $\text{content}_{gcd}(A)$ that will manifest in, for example the matrix $\Delta$, which is a non linear combination of elements in $A$. In any case, whenever we refer to “content” further, we mean systematic matrix content in the context of Definition 2, and so take $\text{content} = \text{content}_{sys}$. If we mean content in terms of Definition 1, we will use $\text{content}_{gcd}$.

Note that for some polynomial $c$ and some matrix $A$, $\text{Adj}(cA) = c^{n-1}\text{Adj}(A)$. This is especially important if we are to view $c$ in terms of matrix content of $A$ - we can, at the very least, calculate $\text{Adj}(A)$ from a “primitive” $A$, and as such multiply only the polynomials of lowest degree possible in the context of matrix multiplication in the process of Algorithm 2. This is the motivation behind Definition 2. In fact, it will be seen further that the term $c^{n-1}$ is quite often the factor that gets divided through (as can be seen frequently in Algorithm 1), and so in practice we can discard it.

2.2 Analysis of Degree Bloat in Intermediate Matrices

Forward, we assume that $d_2$ from Algorithm 2 is $\text{Det}(A)$.

**Remark 3.** Where we claim a matrix to have content, this can be verified by computation, for example in Maple, from an original symbolic matrix. Thus, we defer proof of claims of matrix content to computation. We note that the deterministic nature of all computation involved ensures that in the context of systematic matrix context, this will always hold for that particular matrix. A matrix divided through by its content will always result in an integral matrix, i.e. fraction free / the divisions are exact.

**Theorem 4.** All intermediate matrices at the top level for Algorithm 2 are of degree $O(nd)$.

**Proof.** We assume $n \geq 4$ such that $\Delta$ makes sense in the context of Algorithm 2. The fraction free inversion of $(A_{1,1}, 1)$ returns $(A_{1,1}^{adj}, a_{1,1}^{adj}) = (\text{Adj}(A_{1,1}), \text{Det}(A_{1,1}))$. $A_{1,1}^{adj}$ is of degree $d(\frac{n}{2} - 1)$ as $A_{1,1}$ is of size $\frac{n}{2}$, and similarly $a_{1,1}^{adj}$ is a single polynomial of degree $\frac{nd}{2}$. $\Delta$ is $a_{1,1}^{adj}A_{2,2} - A_{2,1}a_{1,1}^{adj}A_{1,2}$, which is of degree $d(\frac{n}{2} + 1)$, and $\delta = a_{1,1}^{adj}$. We then calculate the fraction free inverse of $(\Delta, \delta)$ as $(\Delta^{adj}, \delta^{adj})$. $\Delta^{adj}$ is the adjugate of $\Delta$, multiplied through by a factor of $a_{1,1}^{adj}$ (as $\delta = a_{1,1}^{adj}$ appears as $d_1$ in Algorithm 2 here), so $\Delta^{adj}$ has degree $d(\frac{n}{2} + 1)(\frac{n}{2} - 1) + \frac{nd}{2}$. Note that now we divide through by $a_{1,1}^{adj}$, which is degree $\frac{nd}{4} - \frac{nd}{2}$. Thus $\Delta^{adj}$ ends up being degree $(n - 1)d$ (the degree of the adjugate). All ensuing operations at the top level is matrix arithmetic on matrices of degree $O(nd)$, and as such all resulting matrices are $O(nd)$. Note that the cancellations involving $a_{1,1}^{adj}$ are $O(nd)$ cancellations, and so only end up subtracting a factor of $nd$ up to a constant in the degrees of the matrices involved.

Note that additionally in this formulation, we can cancel an extra factor of $a_{1,1}^{adj}$ from $\Lambda$, such that $\Lambda$ becomes degree $d(n - 1)$. 

7
The above demonstrates that adjugates and $\Delta$ are both inherently “degree $nd$” objects, in that calculating them will introduce a factor of $n$ to the degrees of the polynomials in arithmetic. So the ratio of cost of addition to multiplication certainly isn’t fixed. However in particular, Algorithm 2 being inherently recursive should exacerbate this issue, as we must take “the $\Delta$ of $\Delta$”, and “the adjugate of $\Delta$” and so on.

At first, one should be most concerned about the chain of $\Delta$s that ensues upon calculation of inversion of a suitably large matrix $A$, which we shall investigate as a case study into a special case of the bigger picture.

**Theorem 5.** Let $A \in \mathbb{R}^{n \times n}$. Let $\Delta_0$ denote the $\Delta$ calculated from the submatrices of $A$, and for all $1 \leq k \leq \log_2 n - 1$, let $\Delta_k$ denote the $\Delta$ calculated from the submatrices of $\Delta_{k-1}$. Then $\Delta_k$ is of degree $O(n^{k+1}d)$ for all $0 \leq k \leq \log_2 n - 1$.

**Proof.** Note that $0 \leq k \leq \log_2 n - 1$ ensures that $\Delta_k$ actually exists in the context of Algorithm 2. $\Delta_{\log_2 n-1}$ is of size 4, and as such the last $\Delta$ to fully be inverted by recursion for Algorithm 2, $\Delta_0$ is of degree $d(\frac{n}{2} + 1) \in O(nd)$ by Theorem 4. Now assume that $\Delta_i$ is of degree $O(n^{i+1}d)$ for some $0 \leq i \leq \log_2 n - 1$. We have $\Delta_{i+1}$ is of degree $O(n^{i+1}d)$, and so the associated determinant $\delta_{i+1,i+1}$ is of degree $O(n^{i+2}d)$. $\Delta_{i+1}$ is similarly of degree $O(n^{i+2}d)$, and $\Delta_{i+2,1}, \Delta_{i+1,2}$ are all of degree $O(n^{i+1}d)$. Hence the object $\Delta_{i+1} = \delta_{i+1,i+1} \Delta_{i+2,2} - \Delta_{i+1,2} \Delta_{i+1,i+1}$ is of degree $O(n^{i+2}d)$. By induction, $\Delta_k$ is of degree $O(n^{k+1}d)$ for all $1 \leq k \leq \log_2 n - 1$.

**Remark 6.** In recursions on Algorithm 2 on $A$, for matrices of size $\frac{n}{2^i}, i \geq 1$ there are $2^i$ matrices that are the results of taking various ‘$A_{1,1}$’ and ‘$\Delta$’ operations that we must invert, however the worst case really is inversions involving as many ‘$\Delta$’ operations as possible, as per Theorem 5. In contrast, if we were to talk about ‘$A_{1,1k}$’ in the same manner as Theorem 5, then $A_{1,1k}$ would be of degree $d$ for all $k \geq 1$. Section 2.4 discusses the degree of “mixed cases”, which is the generalisation.

Further, when we say “iteration $k$”, we mean all inversions on matrices of size $\frac{n}{2^k}$, but in particular this recursion level involves $\Delta_{k-1}$ from Theorem 5 and this matrix is certainly the worst case in terms of degree. The following Theorem generalises Theorem 4.

**Theorem 7.** The matrices in arithmetic in iteration $k$ of Algorithm 2 (on $\Delta_{k-1}$) are of degree $O(n^{k+1}d)$.

**Proof.** At iteration $k$ on $\Delta$, we calculate $\Delta_k$, which via Theorem 5 is of degree $O(n^{k+1}d)$. Inversion of this gives us $\Delta_k^{adj}$, but in fact we cancel it to receive $\Delta_k^{adj}$, and in a similar manner to in Theorem 4 this is of degree $O(n^{k+1}d)$. All further arithmetic at this level is on matrices of degree $O(n^{k+1}d)$ and thus leads to matrices of degree $O(n^{k+1}d)$.

**Theorem 8.** Fraction free inversion via Algorithm 3 on a matrix of size $n$ and degree $d$ involves arithmetic on matrices of degree $O(n^{\log_2 n - 1}d)$.  

8
Proof. Algorithm 2 calls itself recursively on $\Delta_i$ matrices (ie. we are in the “else” case of Algorithm 2) $\log_2 n - 2$ times if we stop recursion on Algorithm 2 at size 2 and calculate adjugates via the transpose of the cofactor matrix. Using Theorem 7, we can deduce that the matrices in arithmetic at size 2 (ie. recursion level $\log_2 n - 2$) are of degree $O(n^{\log_2 n - 1} d)$.

Remark 9. The equivalent results from Theorems 7 and 8 for Algorithm 1 would have the degrees at $O(n^{k+2}d)$ and $O(n^{\log_2 n + 1} d)$ respectively.

Proof. Consider Algorithm 1, and observe that Theorem 5 holds the same in this case. In Theorem 7, we do not perform cancellation on $\Delta_{k \text{adj}}$, and so as the adjugate of $\Delta_k$ which is of degree $O(n^{k+1}d)$, it is of degree $O(n^{k+2}d)$, and then all arithmetic is now on matrices of this degree. For Algorithm 1 we instead perform $\log_2 n - 1$ recursions due to recursing down to matrices of size 1. In total, the equivalent result for Theorem 8 has that we perform arithmetic on matrices of degree $O(n^{\log_2 n + 1} d)$.

2.3 Cancellations in Intermediate Matrices

Having to perform arithmetic on matrices of degree $O(n^{\log_2 n - 1} d)$ is beyond the realms of being competitive - however in this section we discuss what cancellations can be provided to mitigate this degree increase. This requires some discussion of mixed cases, which so far we have neglected. But we first present the “pure” $\Delta$ case, ie. always choosing the right subtree from Figure 1.

Remark 10. Assume from here onwards that when computing any intermediate matrix, we do so from a cancelled down matrix, ie. from a matrix with a systematic matrix content of 1. In the context of Section 2.1, $\Delta_k$ would be calculated from a $\Delta_{k-1}$ that has been divided through elementwise by the content defined in due course.

Notation 11. The notation $A_{i,1...j}$ where $i, j \in \{1,...,n\}$ means the upper left submatrix with $i$ rows and $j$ columns. As an example, $A_{1,1}$ would be $A_{1,\frac{n}{2},1:1\frac{n}{2}}$ in this notation. Additionally, let $A_{i,0,1...j}$ be 1 for $i, j < 1$.

Theorem 12. If $0 \leq i \leq \log_2(n) - 2$, $\Delta_i$ has content $\text{Det}(A_{1, (2^i - 1)\frac{n}{2},1, (2^i - 1)\frac{n}{2},1})^{\frac{n}{2^i + 1}}$. After cancelling through by this content, $\Delta_i$ is of degree $(\frac{(2^{i+1} - 1)n}{2^{i+1}} + 1)d \in O(nd)$, for all $0 \leq i \leq \log_2(n) - 2$.

Proof. Note that $0 \leq i \leq \log_2(n) - 2$ ensures that $\Delta_i$ is a matrix of size at least 2, such that the definition of matrix content makes sense.

As above, we calculate each $\Delta_i$ from a $\Delta_{i-1}$ that has been divided through by the content from 12 and thus has a resulting content of 1. The theorem is true for $i = 0$ as by computation on a symbolic matrix of any size we can verify it has a content of 1, which is equivalent to $A_{1,0,1...0}$ under Notation 11 and Theorem 5 has that $\Delta_0$ is of degree $d(\frac{n}{2} + 1)$. 

9
Now assume $\Delta_i$ is of degree $\left(\frac{(2i+1-1)n}{2+i+2}\right) + 1$ for some $i \geq 0$. Then $\Delta_i$ is of size $\frac{n}{2+i}$, and the $A_{1,1}$ for $\Delta_i$ has the same degree as $\Delta_i$, and is of size $\frac{n}{2+i}$. So Det$(A_{1,1})$ is of degree $\frac{n}{2+i}$ $\left(\frac{(2i+1-1)n}{2+i+2}\right) + 1$), and so $\Delta_{i+1}$ is of degree $\left(\frac{n}{2+i+2}\right) + 1$). Expanding this, we get $\left(\frac{(2i+1-1)n^2}{2+i+3} + \frac{(2i+2-1)n}{2+i+2}\right) + 1$). We know via Theorem 12 that we can divide through by Det$(A_{1,1})$ so we achieve degree $\frac{n}{2+i+2}$. As such after cancellation $\Delta_{i+1}$ is of degree $\left(\frac{n}{2+i+2}\right) + 1$), which is precisely the induction statement but with $i$ incremented by 1, evidently $\Delta_{i+1}$ is of degree $O(nd)$ as a result, and thus by mathematical induction the statement holds for all $i \geq 1$.

**Theorem 13.** If $0 \leq i \leq \log_2(n) - 3$, $\Delta_i^{adj}$, when computed from a corresponding $\Delta_i$ of matrix content 1, has content Det$(A_{1,1}, (2i+1-1)n, (2i+1-1)n)$ $\frac{n}{2+i+2}$. After dividing through by this content, $\Delta_i^{adj}$ achieves degree $O(nd)$.

**Proof.** Note that $0 \leq i \leq \log_2(n) - 3$ ensures that $\Delta_i^{adj}$ is a matrix of size at least 4. $\Delta_i^{adj}$, is the result of transposition of the cofactor matrix of $\Delta_i$, and as such no arithmetic is performed, and as such its content is certainly 1.

Using Theorem 12, we have that after appropriate cancellations $\Delta_i$ is of degree $\left(\frac{(2i+1-1)n}{2+i+2}\right) + 1$), and size $\frac{n}{2+i}$. Computing $\Delta_i^{adj}$ we get that it is of degree $\left(\frac{n}{2+i} - 1\right)\left(\frac{(2i+1-1)n}{2+i+2}\right) + 1$). Expanding this gives $\left(\frac{(2i+1-1)n^2}{2+i+3} + \frac{(2i+2-1)n}{2+i+2}\right) + 1$). Now via Theorem 13, we can divide through by Det$(A_{1,1}, (2i+1-1)n, (2i+1-1)n)$ $\frac{n}{2+i+2}$. This gives $\left(\frac{(2i+1-1)n^2}{2+i+3} + \frac{(2i+2-1)n}{2+i+2}\right)$. After cancellation $\Delta_i^{adj}$ is of degree $\left(\frac{2i+2-2i+1+1}{2+i+2}\right) = (n - 1)d \in O(nd)$.

### 2.4 Mixed Cases

The red path in figure 1 is an example of a “mixed case”. In ad-hoc functional notation, the red path gives us $M = \Delta(A_{1,1}, (A))$. Note that with this notation, $\Delta_k$ from 2.1 becomes $\Delta_k(A)$, and from here onwards we omit the brackets. Essentially, we require the adjugate of $A_{1,1}$ in order to invert $\Delta_0$, which is required in order to invert $A$. The case study traversed in Section 2.1 would have us believe that we should treat the inversion of this $A_{1,1}$ as if it were a completely new matrix - indeed it has a trivial systematic matrix content as per the first $\Delta_0$ (see Theorem 14), but then this leads us to believe that $M$ has no systematic content (as it is a $\Delta_0$). Except $M$ is the result of two net $\Delta$ operations, and thus is of degree $O(n^2d)$. We want $O(nd)$.

**Theorem 14.** If $M$ is a matrix produced directly from a recent $A_{1,1}$ operation, then content$_{sys}(M) = 1$. Otherwise, keeping in mind that we calculate each matrix from matrices of respective systematic matrix content 1, if $M = \Delta \phi_1 \ldots \phi_k A$ where $\phi_j \in \{A_{1,1}, \Delta\}$ for all $1 \leq j \leq k$, and $A$ is a matrix of binary size $n$ at least $2^{k+2}$, then...
Figure 1: A representation of the binary tree for the inversion of a matrix $A$ (that should be of size at least 16 here) via Algorithm 2. We simplify notation such that “the $A_{1,1}$ of $\Delta_0$” is just labeled as “$A_{1,1}$”. Note that the tree is left-oriented, that is for any node, the inversion of the left subtree is required before the inversion of the right subtree can be computed, which is required for the inversion of the node (computation of $\Delta$ requires the pair $(A_{1,1}^{adj}, A_{1,1}^{adj})$). The red path shows an example of a “mixed case”.

\[
\text{content}_{sys}(M) = \text{Det}(A_{1,1}^{(2^2-1)n,1}, A_{1,1}^{(2^2-1)n,2})^{\frac{1}{2^k+1}}, \text{ where } i \text{ is the number of } \phi_j \text{ that are } \Delta. 
\]

Upon cancelling through by this, we achieve degree at most $O(nd)$. 

**Proof.** If $M$ is a matrix produced directly from a recent $A_{1,1}$ operation (ie. $\phi_1 = A_{1,1}$), then we are looking at an unmodified submatrix of a matrix that is assumed to have been cancelled through, and so we can deduce no further content, so $\text{content}_{sys}(M) = 1$. Hence it suffices to work with sequences of operations beginning with $\Delta$, as we do further.

To prove the claim, we work with generic intermediate matrices of the form $M = \Delta A_{1,1}^{m_1} \Delta A_{1,1}^{m_{J-1}} \Delta \ldots \Delta A_{1,1}^{m_{J}} \Delta A$, where $J > 0$, each $m_i \geq 0$ for all $1 \leq i \leq J$, and $A$ is a matrix of size at least $2^{J+1+\sum_{i=1}^{J} m_i}$. In the case where $m_i = 0$ for all $1 \leq i \leq J$, this is identical to the case of Theorem 12. We induct on $J$ over $\mathbb{N}_0$. 

For $J = 0$, we obtain $\Delta A$, and this is of degree $(\frac{n}{2} + 1)d$, which is of degree $O(nd)$. 

Now, for the purposes of induction, assume $J \geq 0$, and $M = \Delta A_{1,1}^{m_1} \Delta \ldots \Delta A_{1,1}^{m_J} \Delta A$ is of degree

\[
\frac{1 + \sum_{j=1}^{J} 2^{J+1-j+\sum_{i=j}^{J} m_i}n d}{2^{J+1+\sum_{i=1}^{J} m_i}} + 1 - O(n^{k_0}d),
\]

for some $k_0 \geq 2$ and of size $2^{-J-1-\sum_{i=1}^{J} m_i} n$. Then consider $\Delta A_{1,1}^{m_{J+1}} M$, where $m_{J+1} \geq 0$. This is of size $2^{-J-1-\sum_{i=1}^{J+1} m_i} n$, and hence of degree

\[
\left(\frac{n}{2^{J+2+\sum_{i=1}^{J+1} m_i}} + 1\right) \left(\frac{1 + \sum_{j=1}^{J} 2^{J+1-j+\sum_{i=j}^{J} m_i}n}{2^{J+1+\sum_{i=1}^{J} m_i}} + 1\right) d.
\]

11
The coefficient of \( n^2d \) is
\[
1 + \sum_{j=1}^{J} 2^{j} \frac{\sum_{i=j}^{J} m_i}{2^{2J+3} + \sum_{i=1}^{J+1} m_i + \sum_{i=1}^{J} m_i}.
\]
and the theorem suggests we can cancel through by
\[
\text{Det}(A, (2^{J+1} - 1)n, \ldots , (2^{J+1} - 1)n^{J+1}) n^{2J-2 - \sum_{i=1}^{J} m_i},
\]
which is of degree
\[
\frac{(2^{J+1} - 1)n^2d}{2^{\sum_{i=1}^{J+1} m_i + 2J+3}}.
\]
After cancellation, the coefficient of \( n^2d \) becomes:
\[
1 + \sum_{j=1}^{J} 2^{j} \frac{\sum_{i=j}^{J} m_i + j - 2^{J+1} \sum_{i=1}^{J} m_i + 2 \sum_{i=1}^{J} m_i}{2^{2J+3} + \sum_{i=1}^{J+1} m_i}
\]
If we are to achieve degree \( O(nd) \), this must be non positive.
\[
\Rightarrow 1 + \sum_{j=1}^{J} 2^{j} \frac{\sum_{i=j}^{J} m_i + j - 2^{J+1} \sum_{i=1}^{J} m_i + 2 \sum_{i=1}^{J} m_i - 1}{2^{2J+3} + \sum_{i=1}^{J+1} m_i} \leq 0
\]
Manipulating the left hand side, we receive:
\[
\leq 1 + \sum_{j=1}^{J} 2^{j} \left( \sum_{i=j}^{J} m_i + j - 2^{J+1} \sum_{i=1}^{J} m_i + 2 \sum_{i=1}^{J} m_i \right)
\]
\[
= 1 + 2 \sum_{i=1}^{J} m_i \left( \sum_{j=1}^{J} 2^j - 2^{J+1} + 1 \right)
\]
Using that \( \sum_{j=1}^{J} 2^j = 2^{J+1} - 2 \),
\[
= 1 + 2 \sum_{i=1}^{J} m_i \left( 2^{J+1} - 2^{J+1} + 1 \right)
\]
\[
= 1 - 2 \sum_{i=1}^{J} m_i
\]
which is certainly non positive as \( m_i \geq 0 \) for all \( 1 \leq i \leq J + 1 \). Hence the \( n^2d \) term is at most 0, and thus we achieve \( O(nd) \). We verify that we match the coefficient of \( nd \) from the inductive step, which is:
\[
\frac{1}{2^{\sum_{i=1}^{J} m_i + j}} + \frac{1 + \sum_{j=1}^{J} 2^{j+1} \sum_{i=j}^{J} m_i}{2^{J+1} + \sum_{i=1}^{J+1} m_i} + 1
\]
\[
= \frac{1 + 2^{m_{J+1}} + \sum_{j=1}^{J} 2^{\sum_{i=j}^{J+1} m_i + j + 1}}{2^{J+2} + \sum_{i=1}^{J+1} m_i} + 1
\]
12
\[ \frac{1 + \sum_{j=1}^{J+1} 2 \sum_{i=1}^{m_i} m_i + j}{2^{J+2+\sum_{i=1}^{m_i} m_i}} \]

which is the induction statement about the degree, but with \( J + 1 \). Hence by induction \( M \) achieves degree \( O(nd) \) for all \( J \geq 0 \), and so all mixed case \( \Delta \) achieve degree \( O(nd) \).

**Remark 15.** Theorem 12 is consistent with Theorem 14 by taking or \( m_i = 0 \) for all \( i \) in the latter to get the former, NOT by taking \( J = 0 \). It is important to note \( J \) is commensurate with the net number of \( \Delta \) operations undertaken, whereas the \( m_i \) represent the number of \( A_{1,1} \) operations undertaken. Note that we know from Theorem 12 that the cancellations induced in the pure \( \Delta \) case completely coincide with the \( O(n^2d) \) term from the degree, ie. the ‘\( \leq \)’ from the above theorem is actually an ‘\( = \)’ in this case.

**Remark 16.** We explain the presence of \( k_0 \) in the above proof. Note that in the case that the cancellation “overshoots”, ie. the ‘\( \leq \)’ is really a ‘\( < \)’, this results in a negative \( O(n^2d) \) term in the degree. Further, this will actually increase by a factor of \( n \) every inductive step (every time \( J \) increases with an associated \( m_J > 0 \), ie. there has been an additional tangible chain of \( A_{1,1} \) operations) by virtue of the multiplication by the term induced by taking the last \( \Delta \) operation. As such, a mixed case may actually be of a lower degree than the pure \( \Delta \) case of the same size by a term of \( O(n^kd) \) where \( k_0 \) is proportional to the number of non-zero \( m_i \). As above, Theorem 12 implies no such increasing negative \( O(n^kd) \) term for the pure \( \Delta \) case.

**Remark 17.** 16 is the first matrix size where we obtain a mixed case \( \Delta \) of degree an order higher than \( O(nd) \) that we then cancel. If \( A \) is a matrix of size 16, this admits the sequence of operations \( \Delta A_{1,1} \Delta A \), which has two separate contiguous sequences of \( \Delta \). This results in a matrix of degree \( O(n^2d) \) (precisely, degree \( \frac{n^2}{8} + \frac{3n}{4} + 1)d = 45d \), and theorem 14 suggests we can cancel through by a factor of degree \( \frac{n^2}{16} \). A Maple worksheet examining this case can be found at [Ton18].

The equivalent discussion for cancellation of mixed case adjugates (ie. a “mixed case” \( \Delta_{adj} \)) is not traversed here. The existence of cancellations for mixed case \( \Delta \) and “pure” \( \Delta_{adj} \) suggests that cancellations for mixed case \( \Delta_{adj} \) are likely, but the formula and degree of this cancellation are both unknown, too whether it results in a cancellation down to \( O(nd) \).

2.5 Limitations of Cancellations

As the degree of both elements of the intended output of a fraction free inversion algorithm is \( O(nd) \), it is good news that it appears so far that we can work with elements of optimal degree after cancellation. However one has to think about the cancellations themselves, and that requiring these cancellations induces further problems.

The above theorems describe factorisations possible in the binary case - that is the original matrix \( A \) was of binary size, and subsequently all intermediate matrices are of
binary size. But the factors found are determinants of submatrices of the original $A$, and these submatrices are (not usually) of binary size. In some sense, it would be elegant and perhaps convenient to recurse on Algorithm 2 to acquire these determinants, in some manner similar to that of Bareiss-Dodgson for Gaussian Elimination [Dod66] [Bar68]. This requires an extension of Algorithm 2 to be able to accept and work with matrices of non-binary size. This can be done via padding the original matrix in a manner similar to that for Strassen-like multiplication. And indeed such a padding solution can also be employed to be able to multiply intermediate matrices that are of odd size, but actually a better solution is probably “peeling” as presented in [Hus+96] in the context of Strassen-like multiplication.

We briefly discuss padding for inversion. If $A$ is of odd size, we introduce a row of 0s, and then a further column of 0s, and add a 1 on the leading diagonal. This conditions the input matrix to be of even size, and so ensures that we can divide the matrix into quarters. Note that this matrix has the same determinant as the input, and we can “trim” the resulting adjugate to receive the adjugate of the actual input. But this has interesting implications on the intermediate matrices depending on if we pad “upper left” or “lower right” - in particular the elements we introduce are of maximum degree 0, and so our choice of padding has interesting consequences on the $\Delta_i$ matrices - in particular the former choice leads to $\Delta_i$ matrices of uniform degree, whereas the latter choice results in $\Delta_i$ that are more diagonal (ie. we get some 0s off the leading diagonal in the far right and bottom left of the matrix), but of potentially higher maximum degree. A Maple worksheet examining the dichotomy of cases can be found at [Ton18].

In particular however, with respect to the obtained cancellations we now require up to four recursions on the algorithm per iteration - the two original recursions on $A_{1,1}$ and $\Delta$, and up to two further recursions to provide cancellation factors for $\Delta$ and $\Delta_{adj}$. Furthermore, these two latter recursions may be on submatrices that are close to the whole original matrix $A$. In particular, the pure $\Delta_i$ or $\Delta_i^{adj}$ for larger $i$ may require recursions on matrices of size $\approx n$, which isn’t in the spirit of the rest of the algorithms that require recursions purely on matrices of half the size - a disaster for worst case complexity.

An alternative to the additional recursions is to instead calculate the content in the spirit of Definition 1. That is, we take pairwise GCDs of elements, but we can in fact be more intelligent and terminate taking the GCDs when we reach the degree we expect the content to have. This is a compromise between systematic and non-systematic matrix content, and ensures in best terms possible that we don’t catch factors that we wouldn’t normally deduce to be there. We can also randomise our selection of elements to take GCDs of in the same spirit as calculating content of polynomials. There is a consideration however that in the situation where $A$ was not uniformly of degree $d$, the above theorems on cancellations only provide upper bounds for the degree that we expect.

**Theorem 18.** Assuming that every $\Delta^{adj}$ matrix admits cancellation (including those arising from mixed cases), and via the above suggestion that we take pairwise GCDs
to cancel $\Delta_i$ and $\Delta_i^{\text{adj}}$, we require taking $O(n)$ GCDs of polynomials of degree at least $O(n^2d)$ in $m$ variables.

Proof. At iteration $i$, $1 \leq i \leq \log_2 n - 1$, we have $2^i$ principal submatrices that are the results of taking various $A_{1,1}$ and $\Delta$ operations. Half of these are the result of a $\Delta$ operation at this level. Exactly one is the result of just one net $\Delta$ operation, while the rest admit cancellation. This results in $2^{i-1} - 1$ matrices of a $\Delta$ description that admit cancellation, $1 \leq i \leq \log_2 n - 2$. Note that we assume each $\Delta$, $\Delta^{\text{adj}}$ is calculated from a matrix of degree $O(nd)$, and as such before cancellation is degree $O(n^2d)$. Also noting that we must take at least one GCD per matrix in order to obtain its content, we take at least $\sum_{i=1}^{\log_2 n - 1} 2^{i-1} - 1 + \sum_{i=1}^{\log_2 n - 2} 2^{i-1} = \frac{3}{2}2^{\log_2 n - 1} - \log_2 n - 1 = \frac{3n}{4} - \log_2 n - 1 \in O(n)$ GCDs.

Remark 19. Interestingly, the worst case (in terms of number of GCDs) is at most $O(n^2)$ GCDs of polynomials of degree at least $O(n^2d)$ if we have to take pairwise GCDs of every element in every intermediate matrix to find the respective content. At iteration $i$, each matrix has $\frac{n^2}{2^{i-2}}$ elements, so the sum is $n^2(\sum_{i=1}^{\log_2 n - 1} \frac{2^{i-1} - 1}{2^{i-2}} + \sum_{i=1}^{\log_2 n - 2} \frac{1}{2^{i-2}}) = \frac{4}{3}(2n^2 - 9n + 4) \in O(n^2)$. Anecdotally, the author finds that the content can usually be found in exactly one step, ie. exactly one GCD (of any two elements) coincides with $\text{content}_{\gcd}(A)$. Therefore, we expect the average case to be at least $O(n)$ GCDs.

3 Other methods

We acknowledge a fraction free formulation arising from work summarised & reviewed in [SY10], regarding arithmetic circuits & formulae. Informally, Strassen’s formulations can be followed in such a manner that we separate the numerator and denominator of each entry of the result during calculation. In particular however this assumes unit cost for arithmetic to be efficient, and so does not avoid the issue we present. In particular one very large GCD must be taken per entry of the result upon completion of all relevant arithmetic. We also note that this formulation attributes a denominator to every entry of the result at all times, whereas our formulation associates a denominator (implicitly or otherwise) to every matrix throughout the algorithm.

Giorgi, Jeannerod and Villard present a method for inversion of polynomial matrices in [GJV03], but this is for matrices over $K[x]$, ie. polynomials in one variable, where we cover the case of $m > 1$. This claims a running time of $\tilde{O}(n^3d)$, where $\tilde{O}$ is the usual “soft $O$” which drops logarithmic terms. Similar claims are made for determinants, but with a running time of $\tilde{O}(n^{\omega}d)$, via Storjohann [Sto02]. While their solutions do use fast matrix multiplication, the main part of the solution in [GJV03] revolves around “$\sigma$-bases”, which don’t appear to generalise to the case of multiple variables.

One notes that one could use Kronecker substitutions [AR14] to reduce the case of multivariate polynomial matrices to the case of univariate polynomial matrices, and then use the method discussed above. However, this will increase the degree of polynomials considerably, and it seems unlikely that this outperforms a direct approach.
4 Conclusions and Further Questions

Indeed with respect to matrices of multivariate polynomials, a fraction free solution as a translation of Strassen’s original inversion certainly exists, but has several caveats. An adaptation of Strassen’s inversion requires using an adaptation of the matrix $\Delta$, which in the classical case is in general a matrix of rational functions, but in the fraction free case becomes a degree $O(nd)$ object. This is a result of adjugates / determinants being “degree $nd$” objects. These modified $\Delta$ matrices have a significant amount of “content”, which we find to be redundant. While this content is directly computable via recursion, it is most likely better to remove this content via taking GCDs “intelligently”. We note that for the univariate case covered in [GJV03], with each recursion they obtain two matrices of half the size of degree 2$d$, whereas for our multivariate case we obtain two matrices of half the size of degree at least $O(nd)$ after cancellation, which greatly increases the worst case complexity.

In total, the following remain unexplored:

- The equivalent of Theorem 14 for adjugates (ie. mixed case $\Delta^{adj}$) is still unknown. The fact that the pure case generalises to the mixed case for $\Delta$ seems to make this hopeful, but it is unknown what the factors will be, nor their degree - one would hope this takes the matrices down to degree $O(nd)$ to match optimal degree as per the rest.

- What one does to adapt Algorithm 2 with the cancellations arising from the theorems in this work. In doing so one should essentially require none of the intermediate divisions that presently appear towards getting the $B_{i,j}$, where instead these occur directly on $\Delta$ and $\Delta^{adj}$ in order to ensure the ensuing arithmetic is all on degree $O(nd)$ objects at every iteration. This corresponds to the claim that we find the content to be “redundant”, where as per Algorithm 2 the factors that are divided through are then unused further in the algorithm.

- When cancellations occur, it is most likely best to do so via “intelligent GCDs”. In practice this may require some heuristics to ensure you get the correct content as to end up with correct output - it is easy to verify that you obtain the correct cancellation degree in easy cases, such as matrices of uniform degree of a binary size, such as is assumed in this paper. But in general polynomial matrices may not be of uniform degree nor binary size, and this complicates trying to compute such factors intelligently.

- Whether such an approach is feasible with respect to any cases obtained in practice for polynomial matrix inversion. Certainly the recursion issue seems a damning one, where one may require recursions on matrices approximately as large as the input, but this becomes more prevalent at very large matrix sizes. On smaller examples (which are likely more abundant), this approach may very optimistically beat the classical case where arithmetic would occur on higher degree polynomials than this work would achieve.
References

[AR14] A. Arnold and D.S. Roche. “Multivariate sparse interpolation using randomized Kronecker substitutions”. In: CoRR abs/1401.6694 (2014). arXiv:1401.6694. URL: http://arxiv.org/abs/1401.6694.

[Bar68] E.H. Bareiss. “Sylvester’s Identity and Multistep Integer-Preserving Gaussian Elimination”. In: 22 (1968), pp. 565–578.

[BH74] J. Bunch and J. Hopcroft. “Triangular factorization and inversion by fast matrix multiplication”. In: 28 (1974), pp. 231–236.

[Bro71] W.S Brown. “On Euclid’s Algorithm and the Computation of Polynomial Greatest Common Divisors”. In: J. ACM 18 (1971), pp. 178–504.

[Dod66] C.L. Dodgson. “Condensation of determinants, being a new and brief method for computing their algebraic value”. In: Proc. Roy Soc. Ser. A 15 (1866), pp. 150–155.

[GJV03] P. Giorgi, C-P. Jeannerod, and G. Villard. “On the Complexity of Polynomial Matrix Computations”. In: Proceedings ISSAC 2003. 2003, pp. 135–142.

[Hus+96] S. Huss-Lederman et al. Implementation of Strassen’s Algorithm for Matrix Multiplication. Pittsburgh, Pennsylvania, USA, Oct. 1996. DOI: 10.1145/369028.369096. URL: https://doi.org/10.1145/369028.369096.

[Le 14] F. Le Gall. “Powers of Tensors and Fast Matrix Multiplication”. In: Proceedings ISSAC 2014. 2014, pp. 296–303.

[Sto02] A. Storjohann. “High-Order Lifting and Integrality Certification”. In: Journal of Symbolic Computation 36 (2002), pp. 613–648.

[Str69] V. Strassen. “Gaussian Elimination is not optimal”. In: Numer. Math. 13 (1969), pp. 354–356.

[SY10] A. Shpilka and A. Yehudayoff. “Arithmetic Circuits: A survey of Recent Results and Open Questions”. In: Foundations and trends in Theoretical Computer Science 5 (2010), pp. 207–388.

[TDS17] Z. Tonks, J.H. Davenport, and G.K. Sankaran. “Fast Matrix Operations in Computer Algebra”. In: Post-Proceedings SYNASC 2017. 2017.

[Ton18] Z. Tonks. ‘On Fast Matrix Inversion’ Cancellations Demonstration Worksheet. University of Bath Research Data Archive. 2018. URL: https://doi.org/10.15125/BATH-00577.

[Win71] S. Winograd. “On multiplication of 2 x 2 matrices”. In: Linear algebra and its applications 4 (1971), pp. 381–388.