What About Toom-Cook Matrices Optimality ?

Marco Bodrato and Alberto Zanoni

Centro "Vito Volterra" – Università di Roma Tor Vergata
Via Columbia 2 – 00133 Rome, Italy
{bodrato, zanoni}@volterra.uniroma2.it

Abstract. Karatsuba and Toom-Cook are well-known methods used to multiply efficiently two long integers. There have been different proposal about the interpolating values used to determine the matrix to be inverted and the sequence of operations to invert it. A definitive word about which is the optimal matrix (values) and the (number of) basic operations to invert it seems still not to have been said. In this paper we present some particular examples of useful matrices and a method to generate automatically, by means of optimised exhaustive searches on a graph, the best sequence of basic operations to invert them.

AMS Subject Classification: 11A05, 11A25, 11K65, 11Y70

Keywords and phrases: Polynomial multiplication, Karatsuba, Toom-Cook, interpolation

1 Introduction

Starting with the works of Karatsuba [3], Toom [5] and Cook [1], who found methods to lower asymptotic complexity for polynomial multiplication from $O(n^2)$ to $O(n^{1+\epsilon})$ with $0 < \epsilon \leqslant \log_2 3$, many efforts have been done in finding optimised implementations in arithmetic software packages [4], [7], [2].

The family of so-called Toom-Cook methods is an infinite set of algorithms (called Toom-3, Toom-4, etc. - Karatsuba may be identified with Toom-2). Each of them may be viewed as a polynomial interpolation problem, for which the base points are not specified a priori, from which a matrix to be inverted rise. We indicate the matrix related to Toom-n method with $A_n \in GL(\mathbb{Z}, 2n-1)$.

Moreover, a set O of basic operations (typically sums, subtractions, bit shiftings, multiplication and division by small numbers, etc.) is given. Practically, this is a set of very efficiently implemented basic functions in a certain computer language, and the idea is to use them to invert A_n step by step.

A particular implementation of Toom-n method must then specify

1. The interpolation points v_i (Toom-n requires $2n-1$ values), which determine the matrix A_n.
2. The sequence of operations in O needed to invert the matrix (we indicate it with *inversion sequence*, or IS, for short, when we want to emphasise the dimension, we will use IS_n).

In the scientific literature there is still not a definitive word on which is the best matrix to be used and which is the corresponding sequence of basic operations to invert it. For Karatsuba method, a exhaustive search on a "reasonable" set of interpolation points is very easy, while already for Toom-3 things are less clear. For example, only recently the GMP library (from version 4.2) changed implementation for Toom-3, choosing a different IS_3, more efficient than the one in the precedent release.

It is not trivial at all to prove the optimality of a matrix $\overline{A}_n$ with respect to the related Toom-n method, because in principle there are infinite possibilities for the v_i values, and a combinatorial number of inversion sequences, and different optimality criteria. Some heuristics for v_i choice are present in Zuras' paper [8], but, to the best of our knowledge, the final word has still not been said.

In this paper we introduce some optimality criteria to measure goodness of inversion sequences, and present an algorithm to automatically search for an optimal IS starting from a given matrix A_n. We considered a theoretical model following an approach based on:

- Minimal use of extra memory (no temporary variables)
- Only exact integer divisions
- No matrix support grow (a zero entry remains always zero)
- No multiplication of a line times an integer

2 A short review on Toom-n methods

Toom-n is a class of recursive methods to multiply two polynomials of degree m with complexity lower than $O(m^2)$. In particular, a very standard analysis tells that Toom-n method has asymptotic complexity $O(m^{log_n(2n-1)})$, so that in principle it would be possible to approach linear complexity for $n \to \infty$.

In practise, only the methods with very small values of n (as 2, 3, 4) are used, because of the asymptotically better $O(n \log n \log\log n)$–complexity Schönhage-Strassen method [6]. The thresholds indicating the convenience of one method in comparison with another one depend very much on the implementation, but also on the choice of A_n and of the performed IS_n.

All these methods can be used to multiply also integers, just fixing a number $1 < B \in \mathbb{N}$ (the *base*) and considering integers expansion in base B, so that to each integer u we can univocally associate a polynomial $u_B(x)$:

$$\mathbb{Z} \ni u = \sum_{k=0}^{\log_B i} u_k B^k \qquad \Longrightarrow \qquad u_B(x) = \sum_{k=0}^{\log_B i} u_k x^k \in \mathbb{Z}[x]$$

Integer multiplication reduces to polynomial multiplication, followed by an evaluation:

$$u \cdot v = z \qquad \Longrightarrow \qquad u_B(x) \cdot v_B(x) = z_B(x) \qquad \Longrightarrow \qquad z = z_B(x)|_{x=B}$$

2.1 The classical point of view

As indicated above, usually one makes the assumption that the two factors have the same degree. If it is not the case, padding the lower-degree polynomial with zero coefficients is supposed. To compute $c(x) = a(x) \cdot b(x) \in \mathbb{K}[x]$ such that

$$a(x) = \sum_{i=0}^{d} a_i x^i \qquad ; \qquad b(x) = \sum_{i=0}^{d} b_i x^i$$

we set up an interpolation problem. Of course, for every $v \in \mathbb{K}$, $c(v) = a(v) \cdot b(v)$. Provided $\mathbb{K}$ is not too small, we may choose $2d+1$ different values $v_i \in \mathbb{K}$, and compute the sequences $\mathbf{a} = (a(v_0), \ldots, a(v_{2d}))$ and $\mathbf{b} = (b(v_0), \ldots, b(v_{2d}))$, that let us to compute the product sequence

$$\mathbf{w} = (w(v_0), \ldots, w(v_{2d})) = (a(v_0) \cdot b(v_0), \ldots, a(v_{2d}) \cdot b(v_{2d}))$$

Recovering the set of coefficients $\mathbf{c} = (c_0, \ldots, c_{2d})$ is now just a classical interpolation problem solving

$$A_n \mathbf{c} = \mathbf{w} \quad \Longrightarrow \quad \begin{pmatrix} v_0^{2d} & \cdots & v_0 & 1 \\ v_1^{2d} & \cdots & v_1 & 1 \\ \vdots & & & \vdots \\ v_{2d}^{2d} & \cdots & v_{2d} & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_2 \\ \vdots \\ c_{2d} \end{pmatrix} = \begin{pmatrix} w_0 \\ w_2 \\ \vdots \\ w_{2d} \end{pmatrix} \quad \Longrightarrow \quad \mathbf{c} = A_n^{-1} \mathbf{w}$$

For integer multiplication, suppose the factors have l digits if considered in a certain base B. Now consider them in the new base $B' = B^{l/n}$: what practically happens is to consider n parts of l/n B-digits each. The i^{th} part is nothing but the coefficient a_i (b_i).

Now perform the above indicated computations: the whole process requires $2d+1 = 2n-1$ multiplications of numbers with n times less digits[1], with the additional overhead of $a(v_i), b(v_i)$ values computation and the inversion of the resulting matrix A_n. Note that only matrices with odd order are obtained this way.

2.2 A temporary useful point of view

In this paper, for performance testing, we will also consider "intermediate" versions of Toom-Cook methods, which consider $a(x), b(x)$ with different degrees d_1, d_2, without zero padding. Indicating with $n_1 = d_1+1, n_2 = d_2 + 1$ the number of necessary subdivisions, the principle remains basicly the same, but now the obtained matrices – indicated with A_{n_1,n_2} – can have also even order.

Examples : (without loss of generality we suppose $n_1 > n_2$)

[1] Recursive application of the method to $a(v_i) \cdot b(v_i)$ products results in the asymptotic complexity estimate $O(m^{\log_n(2n-1)})$.

$\boxed{n_2 = 1}$	Here $b(x) = b_0$, and it is more convenient to compute directly the product coefficients $c_i = a_i \cdot b_0$.
$\boxed{(n_1, n_2) = (3, 2)}$	Here $c(x)$ has degree 3, and $A_{3,2}$ has order 4. In section 7 we will refer to this method as Toom-2.5
$\boxed{(n_1, n_2) = (4, 3)}$	Here $c(x)$ has degree 5, and $A_{4,3}$ has order 6. In section 7 we will refer to this method as Toom-3.5
$\boxed{(n_1, n_2) = (4, 2)}$	In this case $A_{4,2}$ has order 5, and $\overline{A}_3$ may well be chosen as an optimal choice $\overline{A}_{4,2}$
$\boxed{\left.\begin{matrix} n_1 - n_2 \\ d_1 - d_2 \end{matrix}\right\} \equiv 0 \ (2)}$	Generalisation of the precedent cases: a "fall back" to the classical Toom-$(d_1 + d_2 - 1)$ method: $\overline{A}_{n_1+n_2-3}$ may well be chosen as an optimal choice for $\overline{A}_{n_1,n_2}$

While the (3,2) and (4,3) models have most a theoretical value, the (4,2) case suits very well in practical cases. Infact, a program/library implementing both Toom-3 and Toom-4 methods *should* use Toom-3 when one factor has 4 parts and the other one just 2.

3 The matrices A_n

Even if Toom-n works for whatever choice of the v_i, it is better to choose them in order to minimize the matrix invertion overhead as much as possible. The inverse is usually computed by a sequence of elementary row operations à la Gauss, and therefore we should search the "shortest" (least number of elementary operations) and easiest (fastest elementary operations) way to compute A_n^{-1}.

It is possible to consider rational v_i values even when working only with integers. Infact, if $v_i = N_i/D_i$, we have

$$a(v_i) \cdot b(v_i) = c(v_i) \implies \left(\sum_{k=0}^{d} a_k \left(\frac{N_i}{D_i}\right)^k\right) \cdot \left(\sum_{k=0}^{d} b_k \left(\frac{N_i}{D_i}\right)^k\right) = \sum_{k=0}^{2d} c_k \left(\frac{N_i}{D_i}\right)^k$$

and multiplying by D_i^{2d} we may get rid of the denominators, so that:

$$\left(\sum_{k=0}^{d} a_k N_i^k D_i^{2d-k}\right) \cdot \left(\sum_{k=0}^{d} b_k N_i^k D_i^{2d-k}\right) = \sum_{k=0}^{2d} c_k N_i^k D_i^{2d-k}$$

which means that the i^{th} line of A_n will be $(N_i^{2d}, N_i^{2d-1} D_i, \ldots, N_i D_i^{2d-1}, D_i^{2d})$. In particular,

- integer values $v_i = N_i$ generate lines $(N_i^{2d}, N_i^{2d-1}, \ldots, N_i, 1)$
- integer reciprocals $v_i = 1/D_i$ generate $(1, D_i, \ldots, D_i^{2d-1}, D_i^{2d})$

We use ∞ as interpolation "value" to indicate $a_d \cdot b_d$ product computation, which in a certain sense represents interpolation on the reciprocal of zero, or, more precisely,

$$a(\infty) = \lim_{x \to \infty} \frac{a(x)}{x^d} = a_d \qquad ; \qquad b(\infty) = \lim_{x \to \infty} \frac{b(x)}{x^d} = b_d$$

Following the literature and for obvious efficiency, we will always consider the interpolating values $v_0 = 0$ and $v_{2d} = \infty$, so that A_n will have the shape

$$A_n = \begin{pmatrix} 1 & 0 & \cdots & 0 & 0 \\ \mathrm{X} & \mathrm{X} & \cdots & \mathrm{X} & \mathrm{X} \\ \vdots & & \vdots & & \vdots \\ \mathrm{X} & \mathrm{X} & \cdots & \mathrm{X} & \mathrm{X} \\ 0 & 0 & \cdots & 0 & 1 \end{pmatrix} \quad \text{with X indicating not zero entries} \tag{1}$$

and $c_0 = w_0$, $c_{2d} = w_{2d}$. Our inversion analysis will then focus mainly on the inner lines.

The well-known result about Vandermonde determinants applied to Toom matrices gives the following:

Proposition 1. *For a Vandermonde matrix A_n generated by the $r = 2n-1$ values $\left\{\infty, \frac{N_2}{D_2}, \ldots, \frac{N_{r-1}}{D_{r-1}}, 0\right\}$ (Toom matrix) we have*

$$\det(A_n) = \left(\prod_{i=2}^{r-1} N_i D_i\right) \cdot \prod_{1<i<j<r} (N_i D_j - N_j D_i)$$

Proof. Starting from the classical Vandermonde formula, and remembering that in order to have integer entries we cleared the denominators in every line of A_n multiplying it by D_i^{r-1}, we have

$$\det(A_n) = \left(\prod_{i=1}^{r} D_i^{r-1}\right) \cdot \prod_{i<j} \left(\frac{N_i}{D_i} - \frac{N_j}{D_j}\right) = \left(\prod_{i=1}^{r} D_i^{r-1}\right) \cdot \prod_{i<j} \left(\frac{N_i D_j - N_j D_i}{D_i D_j}\right) =$$
$$= \left(\prod_{i=1}^{r} D_i^{r-1}\right) \cdot \frac{\prod_{i<j}(N_i D_j - N_j D_i)}{\prod_{i<j} D_i D_j}$$

In the denominator $\prod_{i<j} D_i D_j$ every term D_k appears $r-k$ times in the first place ($k = i, j = k+1, \ldots, r$) and $k-1$ times in the second ($j = k, i = 1, \ldots, k-1$), so that the result is $\prod_{i=1}^{r} D_i^{r-1}$. Canceling out, we obtain

$$= \left(\prod_{i=1}^{r} D_i^{r-1}\right) \cdot \frac{\prod_{i<j}(N_i D_j - N_j D_i)}{\prod_{i=1}^{r} D_i^{r-1}} = \prod_{1 \leqslant i<j \leqslant r} (N_i D_j - N_j D_i)$$

The value 0 corresponds to the choice $N_r = 0, D_r = 1$, while ∞ to $N_1 = 1, D_1 = 0$ (it is easy to prove that the above formula is also valid in this latter case). Considering these two cases ($i = 1, j = r$) apart, we have

$$= \left(D_r \prod_{j=2}^{r-1} D_j\right)\left(\prod_{1<i<j<r} (N_i D_j - N_j D_i)\right)\left(N_1 \prod_{i=2}^{r-1} N_i\right)$$

which gives the desired result.

Lemma 1. *Let $\alpha = \pm 2^a \neq \beta = \pm 2^b$, with $a, b \in \mathbb{Z}$. The number $\gamma = \alpha - \beta$ is a power of 2 only if $\alpha = -\beta$ or $\alpha = 2\beta$ or $\alpha = \beta/2$.*

Proof. There are two cases:

1. $\alpha\beta > 0$: we have $\pm\gamma = 2^a - 2^b = 2^b(2^{a-b} - 1)$, and $2^{a-b} - 1$ is a power of 2 iff $|a - b| \leqslant 1$. Excluding the case $a = b$, corresponding to $\alpha = \beta$, only the cases $\alpha = 2\beta$, $\alpha = \beta/2$ remain.
2. $\alpha\beta < 0$: we have $\pm\gamma = 2^a + 2^b = 2^b(2^{a-b} + 1)$, and $2^{a-b} + 1$ is a power of 2 iff $a = b$, that is $\alpha = -\beta$

Proposition 2. *For each Toom matrix A_n with $n \geqslant 3$, its determinant is not a power of 2.*

Proof. By contradiction, we should have that $\det(A_n) = \left(\prod_{i=2}^{r-1} N_i D_i\right) \prod_{1<i<j<r} (N_i D_j - N_j D_i)$ should be a power of 2, according to proposition 1. Looking at the first factor, this means that all the N_i, D_i should be powers of 2, say $N_i = \pm 2^{e_i}, D_i = 2^{f_i}$ and the same for all the factors $(N_i D_j - N_j D_i)$. Supposing that $(N_i, D_i) = 1$ ($e_i \cdot f_i = 0$), we have

$$(N_i D_j - N_j D_i) = D_i D_j \frac{(N_i D_j - N_j D_i)}{D_i D_j} = D_i D_j \left(\frac{N_i}{D_i} - \frac{N_j}{D_j}\right) = \pm 2^f (x_i - x_j)$$

We can fix $x_1 = 2^a$, any power of 2. Then, by lemma 1, we should choose the other x_i in the set $\{-2^a, 2^{a+1}, 2^{a-1}\}$, but for any two of them we will never have $x_i - x_j$ being a power of two.

4 Optimality criteria

In the following, for a square matrix M we indicate with $M[i, j]$ its entry in position (i, j) and with $M^{(i)}$ its i^{th} line. We begin with some definitions:

Definition 1. *The* **support** *of $M^{(i)}$ is the set $s(M^{(i)})$ of column indexes $j \in \mathbb{N}$ such that $M[i, j] \neq 0$. The* **support** *of M is the set $s(M)$ of pairs $(i, j) \in \mathbb{N} \times \mathbb{N}$ such that $M[i, j] \neq 0$. The cardinality of a line support $s(M^{(i)})$ is indicated with $^{\#}M^{(i)}$, and of the matrix support $s(M)$ with $^{\#}M$.*

Definition 2. *We note with* $gcd_i(M) = gcd(M^{(i)})$ *the greatest common divisor of all coefficients in* $M^{(i)}$.

Inversion sequences are sequences of basic operations changing the initial matrix and producing as last matrix the identity matrix. Gauss' method is e.g. a classical algorithm to produce effective IS, reducing at each step the *maximum* possible cardinality of still-to-be-analysed lines support. However, it uses as basic operations, apart from sums and subtractions, multiplications and division *at each step*, and from a computer point of view this is not quite optimal when looking for efficiency.

Gauss' method merit is that it is very efficient for *general* matrices, but in our case, where matrices have a very particular form, it may be possible to find different SI which are, in terms of nature and number of used basic operations, more efficient.

We propose here some optimality criteria, that guarantee termination of our exhausting algorithm. For A_n matrices as specified in equation (1), we have

$$^{\#}A_n = 1 + (2n-1-2)(2n-1) + 1 = ((2n-1)^2 - 2(2n-1) + 1) + 1 = (2n-1-1)^2 + 1 = 4(n-1)^2 + 1$$

We now list the used criteria: we suppose that the basic row operations involve lines $M^{(i_1)}, M^{(i_2)}$, and that $M^{(i_1)}$ is overwritten with the result. We indicate with $\widetilde{M}$ the matrix *after* the execution of the operation.

A) Support reduction criterion : $^{\#}\widetilde{M}^{(i)} < {}^{\#}M^{(i)}$. The resulting line must have at least one more coefficient with value 0.

B) Alignment criterion : $\widetilde{M}[i_1, j_1]/M[i_2, j_1] = \widetilde{M}[i_1, j_2]/M[i_2, j_2]$. The resulting line must have more entries differing from the corresponding ones in another line by an equal multiplicative factor.

Criterion **(A)** is sufficient to guarantee termination of the exhaustive research algorithm that we'll describe in the following section. The solution actually implemented by GMP applies once criterion **(B)**.

Gauss' method uses a support lexicographic ordering. Supposing no row and column permutation is necessary, at step k we have that $n_2 = k$ for all following lines whose k^{th} entry is not 0. Their support is reduced (coefficient is zet to 0) starting from the leftmost entry. Note that Gauss' method can lead to intermediate step in which the matrix is more dense than the precedent one, while we do not permit this.

5 The Toom graph

We note the identity matrix of order s with I_s; if the order is not relevant, simply with I.

Let an appropriate set of interpolation values $\{v_i\}$ be given, that is, let A_n be known from the beginning. We perform an exhaustive search on all the possible IS, according to the criteria introduced in section 4. This can be modeled by a oriented and weighted graph (N, E) (which we call Toom graph), in which each node $\nu \in N$ is represented by the matrix M_ν. We call α the distinguished "initial" node, such that $M_\alpha = A_n$, and similarly ω the corresponding "final" node for I. Edges are described below.

In order to have a sufficiently easy theoretical modelisation of the graph procedure analysis, we will consider as basic operations the following ones (i, i_1, i_2 are row indexes and $c, c_1, c_2 \in \mathbb{Z}$):

1. $M_{\nu'}^{(i_1)} = \widetilde{M}_\nu^{(i_1)} = c_1 M_\nu^{(i_1)} + c_2 M_\nu^{(i_2)}$
2. $M_{\nu'}^{(i)} = \widetilde{M}_\nu^{(i)} = \dfrac{M_\nu^{(i_1)}}{c} = \left(\dfrac{M_\nu[i,1]}{c}, \dfrac{M_\nu[i,2]}{c}, \cdots, \dfrac{M_\nu[i,2n-1]}{c}\right)$

where each of the divisions by c is exact. The edges $\varepsilon = (\nu_1, \nu_2) \in E$ are identified by the quadruple (i_1, i_2, c_1, c_2) or by the pair (i, c), respectively. With this in mind, IS can simply be considered as edges sequences, or, in graph terminology, as paths joining α to ω.

To consider the different computational cost of row elementary operations depending on c, c_1, c_2 values in graph analysis, we introduce some constant weights, named in table 1 (possibly swapping $|c_1|$ and $|c_2|$ values). `COMBINATION_WEIGHT` is practically the basic cost of an addition/subtraction of two long integers. An appropriated tuning of these constants will result in choosing a particular criterion of graph visit.

Definition 3. *The* **weight** $w(\varepsilon)$ *of an edge* $\varepsilon \in E$ *is the constant defined by table 1 according to the values of* c_1, c_2 *(or simply* c*, in case* 2*). The weight* w_{IS} *of a IS* $= (\varepsilon_1, \ldots, \varepsilon_l)$*, where* $\varepsilon_i = (\nu_i, \nu_{i+1})$ *and* $M_{\nu_{l+1}} = I$*, is* $w_{IS} = w(\varepsilon_1) + \cdots + w(\varepsilon_l)$*. An IS is* **minimal** *if its weight is minimal among the weights of all possible IS connecting* ν_1 *to* ν_{l+1}*. The weight* $w(M)$ *of a matrix* M *is* w_{IS} *of a minimal IS for* M*. The weight* $w(\nu)$ *of a node* ν *is defined as* $w(\nu) = w(M_\nu)$.

1.

$\lvert c_1\rvert$	$\lvert c_2\rvert$	**Operation**	**Weight**
1	1	$+$ or $-$	`COMBINATION_WEIGHT`
$= 2^k > 1$	1	$\pm$, shifting	'' + `_1_2_WEIGHT`
$> 1, \neq 2^k$	1	$\pm$, multiplication	'' + `_1_X_WEIGHT`
$= 2^k > 1$	$> 1, \neq 2^k$	$\pm$, shifting and multiplication	'' + `_2_X_WEIGHT`
> 1	> 1	general linear combination	'' + `_X_Y_WEIGHT`

2.

$\lvert c\rvert$	**Operation**	**Weight**
1	(may be unitary $-$)	~ 0
$= 2^k > 1$	shifting	`SHIFT_WEIGHT`
$> 1, \neq 2^k$	/	`DIVISION_WEIGHT`

Table 1. Weight constants for elementary row operations

Obviously, $w(I) = 0$.

Example (Karatsuba graph): Let $(v_0 = \infty, v_1 = 1, v_2 = 0)$ identify Karatsuba matrix A_2. We have

$$\begin{array}{ccc} A_2 & \xrightarrow{\varepsilon_1} & M_1 \\ {\scriptstyle\varepsilon_2}\downarrow & & \downarrow{\scriptstyle\varepsilon_3} \\ M_2 & \xrightarrow{\varepsilon_4} & I_2 \end{array} \qquad \text{with } A_2 = \begin{pmatrix} 1&0&0 \\ 1&1&1 \\ 0&0&1 \end{pmatrix},\ M_1 = \begin{pmatrix} 1&0&0 \\ 0&1&1 \\ 0&0&1 \end{pmatrix},\ M_2 = \begin{pmatrix} 1&0&0 \\ 1&1&0 \\ 0&0&1 \end{pmatrix}$$

Example (Knuth graph): Let $(v_0 = \infty, v_1 = -1, v_2 = 0)$ identify the Knuth matrix A'_2. We have

$$\begin{array}{ccc} A'_2 & \xrightarrow{\varepsilon_1} & M'_1 \\ {\scriptstyle\varepsilon_2}\downarrow & & \downarrow{\scriptstyle\varepsilon_3} \\ M'_2 & \xrightarrow{\varepsilon_4} & I_2 \end{array} \qquad \text{with } A'_2 = \begin{pmatrix} 1&0&0 \\ 1&-1&1 \\ 0&0&1 \end{pmatrix},\ M'_1 = \begin{pmatrix} 1&0&0 \\ 0&1&-1 \\ 0&0&1 \end{pmatrix},\ M'_2 = \begin{pmatrix} 1&0&0 \\ 1&-1&0 \\ 0&0&1 \end{pmatrix}$$

These almost trivial graphs show that in the Toom graph generated by a fixed matrix A_n there are many minimal IS, and, that different matrices may have equivalent minimal IS. We call A_n-graph the graph generated by a matrix A_n. In the following, we consider Toom graphs generated by A_n (or A_{n_1,n_2}) matrices.

6 Travelling through Toom graphs

What we're really trying to do is practically to solve a minimum weight path problem between α and ω.

Toom graphs are very big. Beginning with Toom-3, it is easy to work with dozens of thousands nodes, and, in order to cope both with memory limitation and time reduction, we developed some strategies not to consider/add some nodes to the graphs. Note that a node ν (and therefore all of its neighbors) might be visited many times, since in general there are many different sequences ("paths") of elementary row operations joining ν with another node ν'.

In general, two opposite approaches are possible to automatically visit the graph by means of a computer program: the *purely functional* one, which does not need the graph to be really constructed, and a second one for which the graph is actually built.

- It is clear that a recursing function f visiting a graph G would have no computer memory occupation problems (Gauss' method gives immediately a reasonable upper bound l for IS lenghts), but this approach would be extremely lengthy in time. Infact, in this case G would be implicitly represented by the stack of f nested callings, with length $\leqslant K \times l$ for a certain fixed factor K, and the amount of needed memory can be precisely estimated from the very beginning (avoiding a computational "explosion" because of space lacking). But there is a "time" disadvantage. When an istance of f ends its work finishing analysing a node ν, all the information about the subgraph G' having ν as initial node is lost. If ν appears again, reached by another path, G' must be completely analysed again.
- On the other hand, f could effectively memorize the nodes of a graph while analysing it, in order to recognise if a node has already been inserted, avoiding thus to loose time. Unfortunately, as said, these graphs are huge, and without a tuned analysis management they grow so much as to rapidly fill all the available memory.

We try to stay someway in the middle, keeping only *some* nodes for *some* time, so as to avoid until possible – possibly before filling up all the available memory – to repeat analysing nodes, but we also apply the below explained considerations in order to skip "trivial" graph analysis, when the path(s) from the current node to the identity node are trivial or can be sufficiently easily estimated.

6.1 A guided tour of the graph

Let $f(\alpha)$ be, as indicated before, the program visiting the graph having α as its initial node. Keeping in mind that we want to obtain IS with lowest possible weight, we introduce a second argument for f, the DESIRED_WEIGHT (DW). Its meaning is related to the fact that we're interested only in IS with $w_{IS} \leqslant DW$, marking as "not interesting" the nodes ν with $w(M_\nu) > DW$.[2]

We implement this marking not as a boolean variable updating, but as two weights (to be compared) we associate to every node ν. These weights are a lower and upper bound of $w(\nu)$, respectively, and are indicated with $m_w(\nu)$ and $\mathcal{M}_w(\nu)$ (for min and max). The function f returns as result the pair $(m_w(\nu), \mathcal{M}_w(\nu))$. The underlying idea is that making use of DW, it is not always really necessary to know precisely the exact weight of a node to classify a matrix as interesting or not.

In the analysis made by $f(\nu, t)$ – where t is the desired weight – for a node ν the following may happen:

$\boxed{m_w(\nu) = \mathcal{M}_w(\nu)}$: this means that ν has already been completely analysed, and that $w(\nu) = m_w(\nu) = \mathcal{M}_w(\nu)$. Then f immediately returns the pair $(m_w(\nu), \mathcal{M}_w(\nu))$, with no further computations.

$\boxed{t < m_w(\nu)}$: then f does not need to recurse over and over again, because, so to say, a sufficient number of operations to invert M_ν is not permitted. Then f immediately returns the pair $(m_w(\nu), \mathcal{M}_w(\nu))$. Note that this may happen even if $w(M_\nu)$ has still not been determined exactly, in particular even if ν has never been found before in the graph.

In the above cases, no further analysis is carried beyond ν, and this make the graph analysis faster. Suppose now $f(\nu, t)$ is executing, and let ν' be a neighbor of ν, joined to ν by the edge ε, and $t' = t - w(\varepsilon)$. The recursive call on ν' will be $f(\nu', t')$, and we call $w = (m_w(\nu'), \mathcal{M}_w(\nu'))$ the returned value.

At this point, the "parent" istance of f receiving w must try the other neighbors of ν, but before doing this, t is updated: $t = \min\{t, m_w(\nu') + w(\varepsilon)\}$. This is a very important step, because if f succeeded in inverting $M_{\nu'}$ with a IS having weight $m_w(\nu') = \mathcal{M}_w(\nu') < t'$, the new value t for DW will be strictly smaller than the precedent one, and all the paths that will be analyzed from now will have a stronger restriction, so that pruning is likely to become more and more effective.

6.2 Heuristics for weight estimates

It is clear that the better estimate one is able to give for a matrix weight $w(M)$, the shorter and more efficient the graph analysis will be, both in space and time requirements. we present here some of the ideas we implemented in order to give a as best as possible estimate.

A first trivial estimate for $m_w(\nu)$ and $\mathcal{M}_w(\nu)$ can be obtained from the support of M_ν and of the single lines and columns, considering that ${}^{\#}I_m = m$.

Definition 4. *A matrix row $M^{(i)}$ is a* **singleton** *line if all its entries but one are zero, and the not zero entry has value $v = 1$ or $v = -1$. It is* **almost singleton** *if $|v| \neq 1$. Similarly for columns. A singleton line is indicated with SL_j, an almost singleton line with ASL_j, where j is the position of v.*

Proposition 3. *Let $M_\nu \in GL(m, \mathbb{Z})$ with ${}^{\#}M_\nu = s$, ns_l the number of its non-singleton lines, and ns_c the number of its non-singleton columns. Then $m_w(\nu) \geqslant \max\{ns_l, ns_c\}$ and $\mathcal{M}_w(\nu) \leqslant (s - m) + ns_l$*

Proof. The common idea is that every non-singleton line must be made singleton. For m_w it is sufficient to consider the best possible case (just one operation for each line to update), while for $\mathcal{M}_w$ consider the worst possible case, in which each operation of type 1 produces just one more zero, so there are $s - m$ of them, followed by a final division for each non-singleton line, to make it monic.

[2] In a certain sense, DW works as a threshold, useful to prune Toom graphs analysis.

Obviously, the more accurate the estimates are, the less nodes are considered/build in the graph, because the function f has more probabilities to obtain sufficient information from the estimates. Weight estimation can be performed line by line, just summing up the corresponding values. There is an interesting particular case:

Proposition 4. *Let $M_\nu \in GL(m,\mathbb{Z})$, and $M_\nu^{(i)} = (a, 0, \dots 0, b, 0, \dots, 0, c)$, with b in j^{th} position, different from the first and the last one. Then this line-weight estimate can be computed exactly.*

Proof. The entries a and c can be set to zero only using respectively the first and last line (both singleton). The minimal (partial) IS depends on a, b and c values. A default strategy could then be:

- Set a to zero using the first line.
- Set c to zero using the last line.
- If $b \neq 1$, divide the i^{th} line by b.

but the three operations might be executed in a different order, depending on the values of weights associated to the basic operations (table 1): for example, it could be more convenient first to divide and then to subtract twice

$$(3, 0, \dots 0, 3, 0, \dots, 0, 3) \Rightarrow (1, 0, \dots 0, 1, 0, \dots, 0, 1) \Rightarrow (0, \dots 0, 1, 0, \dots, 0, 1) \Rightarrow S_j$$

or division could be better done in the middle:

$$(1, 0, \dots 0, 3, 0, \dots, 0, 3) \Rightarrow (0, 0, \dots 0, 3, 0, \dots, 0, 3) \Rightarrow (0, \dots 0, 1, 0, \dots, 0, 1) \Rightarrow S_j$$

The various sequence possibilities correspond to different coefficients in basic operations (different weight constants are used). A very small number of coefficient values comparisons is sufficient to obtain the result, without having to recurse more on this line entries.

Another interesting case: exactly one not singleton line. Here the global analysis is precise, too.

Proposition 5. *Let $M_\nu \in GL(m,\mathbb{Z})$ with $^{\#}\{i \mid M_\nu^{(i)}$ is not singleton$\} = 1$. Then the weight estimate can be computed exactly.*

Proof. In this case $m-1$ basic operation of type 1 (and possibly 1 division) are surely needed, because at most one new 0 can appear at each step. Similar considerations as the ones appearing in the precedent case apply here, in order to recognise which is the best sequence order.

Similar considerations may be done, *mutatis mutandis*, reasoning with columns.

7 Implementation and results

The authors developed code in `C++` based on the STL library to make experiments. In this section we expose some of the obtained results. The basic operations are indicated between matrices using the following notation:

$i \pm= j$	: j^{th} line is destructively added to or subtracted from i^{th} line
$i -= (c)j$	: j^{th} line multiplied by c is destructively subtracted from i^{th} line
$(c)i \pm= j$	: j^{th} line is destructively added or subtracted from i^{th} line multiplied by c
$i /= (c)$	: the entries in i^{th} line are all destructively divided by c
$i \gg (c)$	: the entries in i^{th} line are all destructively divided by 2^c

7.1 Toom-2.5

This case could also be treated by hand, and we present it as a checking test. Starting from the matrix defined by the resonably good values $\{\infty, -1, 1, 0\}$ we obtained the (expected) IS indicated below. Note that $\det(A_{3,2}) = 2$, and therefore at least a division by 2 (shifting) is necessary. The nodes that were really inserted in the Toom-graph are 44.

$$A_{3,2} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \overset{2-=3}{\Longrightarrow} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 \\ 1 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \underset{3-=1}{\overset{2\gg(1)}{\Longrightarrow}} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & -1 & 1 & -1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \underset{2-=4}{\overset{3+=2}{\Longrightarrow}} I_4$$

7.2 Toom-3

$\boxed{GMP\ \textbf{solution}}$: We first present the IS used by GMP library starting from the matrix obtained by using $\{\infty, 2, -1, 1, 0\}$, and compute its weight w_{GMP} according to our definitions.

$$A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 16 & 8 & 4 & 2 & 1 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \overset{2+=(2)3}{\Longrightarrow} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 18 & 6 & 6 & 0 & 3 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underset{3+=4}{\overset{2+=(3)5}{\Longrightarrow}} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 18 & 6 & 6 & 0 & 6 \\ 2 & 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\overset{2/=(6)}{\Longrightarrow} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 3 & 1 & 1 & 0 & 1 \\ 2 & 0 & 2 & 0 & 2 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underset{3\gg(1)}{\overset{2-=(2)1}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0 \\ 1&1&1&0&1 \\ 1&0&1&0&1 \\ 1&1&1&1&1 \\ 0&0&0&0&1 \end{pmatrix} \overset{4-=2}{\Longrightarrow} \begin{pmatrix} 1&0&0&0&0 \\ 1&1&1&0&1 \\ 1&0&1&0&1 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \end{pmatrix} \overset{2-=3}{\Longrightarrow} \begin{pmatrix} 1&0&0&0&0 \\ 0&1&0&0&0 \\ 1&0&1&0&1 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \end{pmatrix} \underset{3-=5}{\overset{3-=1}{\Longrightarrow}} I_5$$

$$w_{GMP} = 8 \cdot \texttt{COMBINATION\_WEIGHT} + \texttt{DIVISION\_WEIGHT} + \texttt{SHIFT\_WEIGHT} + \\ + \texttt{\_1\_X\_WEIGHT} + 2{\cdot}(\texttt{\_1\_2\_WEIGHT})$$

Surprisingly enough, just by applying criterion **(A)** our program was able to compute solutions having a weight less than GMP's. We present here two of them, both obtained from the matrix defined by the interpolating values $\left\{\infty, -1, 1, \frac{1}{2}, 0\right\}$. Because of matrix central symmetry, the same IS (with adapted line and column indexes) can be applied to the matrix used by GMP. The general weight we obtain is

$$8{\cdot}\texttt{COMBINATION\_WEIGHT} + \texttt{DIVISION\_WEIGHT} + \texttt{SHIFT\_WEIGHT} + \\ \min(\texttt{\_1\_X\_WEIGHT}, \texttt{SHIFT\_WEIGHT}) + \texttt{\_1\_2\_WEIGHT}$$

and depending on `SHIFT_WEIGHT` and `_1_X_WEIGHT` values, the below indicated solutions are obtained.

$\boxed{1^{st}\ \textbf{solution}}$: `SHIFT_WEIGHT` > `_1_X_WEIGHT`

Note that $\det(A_3) = 12$, and the division is "split" into a shifting and a division by 6. By using estimates, the nodes that were really inserted in the Toom-graph are only 11205. Considering also criterion **(B)** – applied in the IS used by GMP – we have that the graph get bigger, reaching 123213 nodes, but the optimal found solution $\overline{IS}$ does not change.

Moreover, *independently from weight values*, one has $w(\overline{IS}) < w_{GMP}$.

$$A_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 4 & 8 & 16 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \overset{4-=2}{\Longrightarrow} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 3 & 3 & 9 & 15 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \overset{2-=3}{\Longrightarrow} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & -2 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 3 & 3 & 9 & 15 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underset{2/=(-1)}{\overset{3-=1}{\Longrightarrow}}$$

$$\tilde{A}_3 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 3 & 3 & 9 & 15 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underset{2\gg(1)}{\overset{4-=(3)3}{\Longrightarrow}} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 6 & 12 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \underset{3-=2}{\overset{4/=(6)}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0 \\ 0&1&0&1&0 \\ 0&0&1&0&1 \\ 0&0&0&1&2 \\ 0&0&0&0&1 \end{pmatrix} \underset{4-=(2)5}{\overset{3-=5}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0 \\ 0&1&0&1&0 \\ 0&0&1&0&0 \\ 0&0&0&1&0 \\ 0&0&0&0&1 \end{pmatrix} \overset{2-=4}{\Longrightarrow} I_5$$

$\boxed{2^{nd}\ \textbf{solution}}$: SHIFT_WEIGHT $<$ _1_X_WEIGHT

Here only 11862 nodes are generated. The initial operations are the same as above: the different behavior begins from the fourth matrix $\tilde{A}_3$.

$$\tilde{A}_3 = \begin{pmatrix} 1&0&0&0&0\\ 0&2&0&2&0\\ 0&1&1&1&1\\ 0&3&3&9&15\\ 0&0&0&0&1 \end{pmatrix} \underset{2\gg(1)}{\overset{4/=(3)}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0\\ 0&1&0&1&0\\ 0&1&1&1&1\\ 0&1&1&3&5\\ 0&0&0&0&1 \end{pmatrix} \overset{4-=3}{\Longrightarrow} \begin{pmatrix} 1&0&0&0&0\\ 0&1&0&1&0\\ 0&1&1&1&1\\ 0&0&0&2&4\\ 0&0&0&0&1 \end{pmatrix} \underset{3-=2}{\overset{4\gg(1)}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0\\ 0&1&0&1&0\\ 0&0&1&0&1\\ 0&0&0&1&2\\ 0&0&0&0&1 \end{pmatrix} \underset{4-=(2)5}{\overset{3-=5}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0\\ 0&1&0&1&0\\ 0&0&1&0&0\\ 0&0&0&1&0\\ 0&0&0&0&1 \end{pmatrix} \overset{2-=4}{\Longrightarrow} I_5$$

With our program we tested all the matrices obtained from interpolating points $\{\infty, a, b, c, 0\}$, where $a, b, c \in \left\{\pm 1, \pm 2, \ldots, \pm 16, \pm\frac{1}{2}, \pm\frac{1}{3}, \ldots, \pm\frac{1}{16}\right\}$. No IS with weight smaller than the two presented above was found. Paths with the same weight are substantially equivalent to these ones and were found starting from interpolating points $\{\infty, 1, -1, \pm 2, 0\}$ and $\left\{\infty, 1, -1, \pm\frac{1}{2}, 0\right\}$.

7.3 Toom-3.5

Starting from the matrix defined by the values $\{\infty, 2, -2, 1, -1, 0\}$ we obtained the sequence of operations indicated below, with weight

$$12 \cdot \texttt{STEP\_WEIGHT} + 2 \cdot \texttt{DIVISION\_WEIGHT} + 2 \cdot \texttt{SHIFT\_WEIGHT} + 2 \cdot \texttt{\_1\_2\_WEIGHT}$$

$$A_{4,3} = \begin{pmatrix} 1&0&0&0&0&0\\ 32&16&8&4&2&1\\ -32&16&-8&4&-2&1\\ 1&1&1&1&1&1\\ -1&1&-1&1&-1&1\\ 0&0&0&0&0&1 \end{pmatrix} \underset{5-=4}{\overset{3-=2}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0&0\\ 32&16&8&4&2&1\\ -64&0&-16&0&-4&0\\ 1&1&1&1&1&1\\ -2&0&-2&0&-2&0\\ 0&0&0&0&0&1 \end{pmatrix} \underset{3\gg(1)}{\overset{2-=6}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0&0\\ 32&16&8&4&2&0\\ -32&0&-8&0&-2&0\\ 1&1&1&1&1&1\\ -2&0&-2&0&-2&0\\ 0&0&0&0&0&1 \end{pmatrix}$$

$$\underset{5/=(-1)}{\overset{\substack{2+=3\\4-=6}}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0&0\\ 0&16&0&4&0&0\\ -32&0&-8&0&-2&0\\ 1&1&1&1&1&0\\ 2&0&2&0&2&0\\ 0&0&0&0&0&1 \end{pmatrix} \underset{5\gg(1)}{\overset{3+=5}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0&0\\ 0&16&0&4&0&0\\ -30&0&-6&0&0&0\\ 1&1&1&1&1&0\\ 1&0&1&0&1&0\\ 0&0&0&0&0&1 \end{pmatrix}$$

$$\underset{4-=5}{\overset{3/=(-6)}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0&0\\ 0&16&0&4&0&0\\ 5&0&1&0&0&0\\ 0&1&0&1&0&0\\ 1&0&1&0&1&0\\ 0&0&0&0&0&1 \end{pmatrix} \underset{2-=(4)4}{\overset{3-=(4)1}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0&0\\ 0&12&0&0&0&0\\ 1&0&1&0&0&0\\ 0&1&0&1&0&0\\ 1&0&1&0&1&0\\ 0&0&0&0&0&1 \end{pmatrix} \underset{2/=(12)}{\overset{5-=3}{\Longrightarrow}} \begin{pmatrix} 1&0&0&0&0&0\\ 0&1&0&0&0&0\\ 1&0&1&0&0&0\\ 0&1&0&1&0&0\\ 0&0&0&0&1&0\\ 0&0&0&0&0&1 \end{pmatrix} \underset{4-=2}{\overset{3-=1}{\Longrightarrow}} I_6$$

8 Conclusions

We presented a method to determine an optimal sequence of basic operations inverting a particular matrix appearing in Toom multiplication methods through an optimized exhaustive research on graphs. Weights for each basic operation are introduced, which determine the inversion cost for the matrices appearing in the intermediate steps. An interval-like estimate analysis permit to know in advance that a path in the graph is not optimal, saving thus time and space.

Intermediate version of Toom methods are also presented, to show more results. As an interesting result, we were able to determine two new inversion sequences for Toom-3 matrix, which seem, to the best of our knowledge, to be new, and good alternatives to the one proposed in the actual version of GMP library.

References

1. Cook, S.A. *On the Minimum Computation Time of Functions* Thesis, Harvard University, pp. 51-77 (1966)
2. The GNU Multiplication Library (GMP) documentation `http://www.swox.com/gmp/#DOC`
3. Karatsuba, A., Ofman, Yu. *Multiplication of multidigit numbers on automata* Soviet Physics-Doklady, 7, 595-596 (1963); translation from Dokl. Akad. Nauk SSSR, 145:2, 293-294, (1962)
4. Knuth, D.E. *The Art of Computer Programming, Vol. 2, Second Edition* Addison-Wesley, Reading Mass., Chapter 4, Section 3.3, pp. 278-301 (1981)
5. Toom, A.L. *The Complexity of a Scheme of Functional Elements Realizing the Multiplication of Integers* Soviet Mathematics, Vol. 3, pp. 714-716 (1963)
6. Schonhage, A., Strassen, V. *Schnelle Multiplikation großer Zahlen* Computing 7 pp. 281-292 (1971)
7. St Denis, T., Rasmussen, M., Rose, G. *Multi-precision math* (tommath library documentation) `math.libtomcrypt.com/files/tommath.pdf`
8. Zuras, D. *On Squaring and Multiplying Large Integers* 11^{th} IEEE Symposium on Computer Arithmetic pp. 260-271 (1993)

Appendix A : Toom-4, Toom-4.5 and Toom-5

We present here some found IS for the indicated methods, that were obtained not in a completely automatic way. Because of limited computing resources, some initial steps were done manually, while the general automatical analysis starts some matrix later.

Toom-4

Starting from the matrix defined by the values $\left\{\infty, 2, 1, -1, \frac{1}{2}, -\frac{1}{2}, 0\right\}$ we obtained the sequence of operations indicated below, with weight

$$\begin{gathered}\texttt{18}\cdot\texttt{STEP\_WEIGHT}+\texttt{3}\cdot\texttt{DIVISION\_WEIGHT}+\texttt{SHIFT\_WEIGHT}+\\ \min\left(\texttt{\_1\_X\_WEIGHT}\ ,\ \texttt{SHIFT\_WEIGHT}\right)+\texttt{2}\cdot\texttt{\_1\_X\_WEIGHT}+\texttt{4}\cdot\left(\texttt{\_1\_2\_WEIGHT}\right)\end{gathered}$$

Depending on `SHIFT_WEIGHT` and `_1_X_WEIGHT` values, we present the two corresponding solutions.

$\boxed{1^{st}\ \textbf{solution}}$: `SHIFT_WEIGHT` > `_1_X_WEIGHT`

$$A_4=\begin{pmatrix}1&0&0&0&0&0&0\\64&32&16&8&4&2&1\\1&1&1&1&1&1&1\\1&-1&1&-1&1&-1&1\\1&2&4&8&16&32&64\\1&-2&4&-8&16&-32&64\\0&0&0&0&0&0&1\end{pmatrix}\overset{\substack{2+=5\\6-=5\\4-=3}}{\Longrightarrow}\begin{pmatrix}1&0&0&0&0&0&0\\65&34&20&16&20&34&65\\1&1&1&1&1&1&1\\0&-2&0&-2&0&-2&0\\1&2&4&8&16&32&64\\0&-4&0&-16&0&-64&0\\0&0&0&0&0&0&1\end{pmatrix}\overset{\substack{5-=1\\5-=(64)7\\4\gg1}}{\Longrightarrow}\begin{pmatrix}1&0&0&0&0&0&0\\65&34&20&16&20&34&65\\1&1&1&1&1&1&1\\0&-1&0&-1&0&-1&0\\0&2&4&8&16&32&0\\0&-4&0&-16&0&-64&0\\0&0&0&0&0&0&1\end{pmatrix}$$

$$\overset{\substack{3+=4\\(2)5+=6}}{\Longrightarrow}\begin{pmatrix}1&0&0&0&0&0&0\\65&34&20&16&20&34&65\\1&0&1&0&1&0&1\\0&-1&0&-1&0&-1&0\\0&0&8&0&32&0&0\\0&-4&0&-16&0&-64&0\\0&0&0&0&0&0&1\end{pmatrix}\overset{\substack{2-=(65)3\\4/=(-1)\\6/=(-1)\\3-=7\\3-=1}}{\Longrightarrow}\begin{pmatrix}1&0&0&0&0&0&0\\0&34&-45&16&-45&34&0\\0&0&1&0&1&0&0\\0&1&0&1&0&1&0\\0&0&8&0&32&0&0\\0&4&0&16&0&64&0\\0&0&0&0&0&0&1\end{pmatrix}\overset{\substack{2+=(45)3\\5-=(8)3}}{\Longrightarrow}\begin{pmatrix}1&0&0&0&0&0&0\\0&34&0&16&0&34&0\\0&0&1&0&1&0&0\\0&1&0&1&0&1&0\\0&0&0&0&24&0&0\\0&4&0&16&0&64&0\\0&0&0&0&0&0&1\end{pmatrix}\overset{\substack{5/=(24)\\6-=2\\2-=(16)4}}{\Longrightarrow}$$

$$\tilde{A}_4=\begin{pmatrix}1&0&0&0&0&0&0\\0&18&0&0&0&18&0\\0&0&1&0&1&0&0\\0&1&0&1&0&1&0\\0&0&0&0&1&0&0\\0&-30&0&0&0&30&0\\0&0&0&0&0&0&1\end{pmatrix}\overset{\substack{2/=(18)\\3-=5}}{\Longrightarrow}\begin{pmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&1&0\\0&0&1&0&0&0&0\\0&1&0&1&0&1&0\\0&0&0&0&1&0&0\\0&-30&0&0&0&30&0\\0&0&0&0&0&0&1\end{pmatrix}\overset{\substack{4-=2\\6+=(30)2}}{\Longrightarrow}\begin{pmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&1&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&60&0\\0&0&0&0&0&0&1\end{pmatrix}\overset{6/=(60)}{\Longrightarrow}\begin{pmatrix}1&0&0&0&0&0&0\\0&1&0&0&0&1&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&0&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{pmatrix}\overset{2-=6}{\Longrightarrow}I_7$$

$\boxed{2^{nd}\ \textbf{solution}}$: `SHIFT_WEIGHT` $<$ `_1_X_WEIGHT`

The initial operations are the same as above: the difference begins from the matrix $\tilde{A}_4$.

$$\tilde{A}_4 = \begin{pmatrix} 1 & 0\,0\,0\,0 & 0\,0 \\ 0 & 18\,0\,0\,0\,18 & 0 \\ 0 & 0\,1\,0\,1 & 0\,0 \\ 0 & 1\,0\,1\,0 & 1\,0 \\ 0 & 0\,0\,0\,1 & 0\,0 \\ 0 & -30\,0\,0\,0\,30 & 0 \\ 0 & 0\,0\,0\,0 & 0\,1 \end{pmatrix} \overset{\substack{2/=(18)\\3-=5\\6/=(30)}}{\Longrightarrow} \begin{pmatrix} 1 & 0\,0\,0\,0\,0 & 0 \\ 0 & 1\,0\,0\,0\,1 & 0 \\ 0 & 0\,1\,0\,0\,0 & 0 \\ 0 & 1\,0\,1\,0\,1 & 0 \\ 0 & 0\,0\,0\,1\,0 & 0 \\ 0 & -1\,0\,0\,0\,1 & 0 \\ 0 & 0\,0\,0\,0\,0 & 1 \end{pmatrix} \overset{\substack{4-=2\\6+=2}}{\Longrightarrow} \begin{pmatrix} 1\,0\,0\,0\,0\,0 & 0 \\ 0\,1\,0\,0\,0\,1 & 0 \\ 0\,0\,1\,0\,0\,0 & 0 \\ 0\,0\,0\,1\,0\,0 & 0 \\ 0\,0\,0\,0\,1\,0 & 0 \\ 0\,0\,0\,0\,0\,2 & 0 \\ 0\,0\,0\,0\,0\,0 & 1 \end{pmatrix} \overset{6\gg(1)}{\Longrightarrow} \begin{pmatrix} 1\,0\,0\,0\,0\,0 & 0 \\ 0\,1\,0\,0\,0\,1 & 0 \\ 0\,0\,1\,0\,0\,0 & 0 \\ 0\,0\,0\,1\,0\,0 & 0 \\ 0\,0\,0\,0\,1\,0 & 0 \\ 0\,0\,0\,0\,0\,1 & 0 \\ 0\,0\,0\,0\,0\,0 & 1 \end{pmatrix} \overset{2-=6}{\Longrightarrow} I_7$$

Toom-4.5

Starting from the matrix defined by the values $\left\{\infty, -1, -2, \frac{1}{2}, 1, 2, -\frac{1}{2}, 0\right\}$ we obtained the sequence of operations indicated below, with weight

$$22 \cdot \texttt{STEP\_WEIGHT} + 4 \cdot \texttt{DIVISION\_WEIGHT} + \texttt{SHIFT\_WEIGHT} + 3 \cdot \texttt{\_1\_X\_WEIGHT} + 6 \cdot \texttt{\_1\_2\_WEIGHT}$$

$$A_{5,4} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ -128 & 64 & -32 & 16 & -8 & 4 & -2 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & -2 & 4 & -8 & 16 & -32 & 64 & -128 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \overset{\substack{2+=5\\7+=4\\3-=6}}{\Longrightarrow} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 & 2 & 0 & 2 \\ -256 & 0 & -64 & 0 & -16 & 0 & -4 & 0 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 2 & 0 & 8 & 0 & 32 & 0 & 128 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\overset{\substack{4-=2\\(2)6+=3\\2\gg(1)}}{\Longrightarrow} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ -256 & 0 & -64 & 0 & -16 & 0 & -4 & 0 \\ 1 & 0 & 4 & 6 & 16 & 30 & 64 & 126 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 128 & 0 & 32 & 0 & 8 & 0 & 2 \\ 2 & 0 & 8 & 0 & 32 & 0 & 128 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \overset{\substack{5-=2\\6-=(128)2\\(2)3+=7}}{\Longrightarrow} \begin{pmatrix} 1\,0 & 0 & 0\,0 & 0 & 0 & 0 \\ 0\,1 & 0 & 1\,0 & 1 & 0 & 1 \\ -510\,0 & -120 & 0\,0 & 0 & 120 & 0 \\ 1\,0 & 4 & 6\,16 & 30 & 64 & 126 \\ 1\,0 & 1 & 0\,1 & 0 & 1 & 0 \\ 0\,0 & 0 & -96\,0 & -120 & 0 & -126 \\ 2\,0 & 8 & 0\,32 & 0 & 128 & 0 \\ 0\,0 & 0 & 0\,0 & 0 & 0 & 1 \end{pmatrix}$$

$$\overset{\substack{4+=6\\2-=8}}{\Longrightarrow} \begin{pmatrix} 1\,0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0\,1 & 0 & 1 & 0 & 1 & 0 & 0 \\ -510\,0 & -120 & 0 & 0 & 0 & 120 & 0 \\ 1\,0 & 4 & -90 & 16 & -90 & 64 & 0 \\ 1\,0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0\,0 & 0 & -96 & 0 & -120 & 0 & -126 \\ 2\,0 & 8 & 0 & 32 & 0 & 128 & 0 \\ 0\,0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \overset{\substack{(2)4-=7\\6+=(126)8\\7-=(32)5}}{\Longrightarrow} \begin{pmatrix} 1\,0 & 0 & 0\,0 & 0 & 0\,0 \\ 0\,1 & 0 & 1\,0 & 1 & 0\,0 \\ -510\,0 & -120 & 0\,0 & 0 & 120\,0 \\ 0\,0 & 0 & -180\,0 & -180 & 0\,0 \\ 1\,0 & 1 & 0\,1 & 0 & 1\,0 \\ 0\,0 & 0 & -96\,0 & -120 & 0\,0 \\ -30\,0 & -24 & 0\,0 & 0 & 96\,0 \\ 0\,0 & 0 & 0\,0 & 0 & 0\,1 \end{pmatrix}$$

$$\overset{\substack{(5)7-=3\\4/=(-180)\\6/=(-24)\\3+=(510)1}}{\Longrightarrow} \begin{pmatrix} 1\,0 & 0\,0\,0\,0 & 0\,0 \\ 0\,1 & 0\,1\,0\,1 & 0\,0 \\ 0\,0 & -120\,0\,0\,0 & 120\,0 \\ 0\,0 & 0\,1\,0\,1 & 0\,0 \\ 1\,0 & 1\,0\,1\,0 & 1\,0 \\ 0\,0 & 0\,4\,0\,5 & 0\,0 \\ 360\,0 & 0\,0\,0\,0 & 360\,0 \\ 0\,0 & 0\,0\,0\,0 & 0\,1 \end{pmatrix} \overset{\substack{3/=(-120)\\7/=(360)\\6-=(4)4\\2-=4}}{\Longrightarrow} \begin{pmatrix} 1\,0\,0\,0\,0\,0 & 0\,0 \\ 0\,1\,0\,0\,0\,0 & 0\,0 \\ 0\,0\,1\,0\,0\,0 & -1\,0 \\ 0\,0\,0\,1\,0\,1 & 0\,0 \\ 1\,0\,1\,0\,1\,0 & 1\,0 \\ 0\,0\,0\,0\,0\,1 & 0\,0 \\ 1\,0\,0\,0\,0\,0 & 1\,0 \\ 0\,0\,0\,0\,0\,0 & 0\,1 \end{pmatrix} \overset{\substack{5-=7\\4-=6\\7-=1}}{\Longrightarrow} \begin{pmatrix} 1\,0\,0\,0\,0\,0 & 0\,0 \\ 0\,1\,0\,0\,0\,0 & 0\,0 \\ 0\,0\,1\,0\,0\,0 & -1\,0 \\ 0\,0\,0\,1\,0\,0 & 0\,0 \\ 0\,0\,1\,0\,1\,0 & 0\,0 \\ 0\,0\,0\,0\,0\,1 & 0\,0 \\ 0\,0\,0\,0\,0\,0 & 1\,0 \\ 0\,0\,0\,0\,0\,0 & 0\,1 \end{pmatrix} \overset{\substack{3+=7\\5-=3}}{\Longrightarrow} I_8$$

Toom-5

Starting from the matrix defined by the values $\left\{\infty, -2, \frac{1}{2}, 4, 2, -1, 1, -\frac{1}{2}, 0\right\}$ we obtained the sequence of operations indicated below, with weight

$$32 \cdot \texttt{STEP\_WEIGHT} + 5 \cdot \texttt{DIVISION\_WEIGHT} + 2 \cdot \texttt{SHIFT\_WEIGHT} + 6 \cdot \texttt{\_1\_X\_WEIGHT} + 8 \cdot \texttt{\_1\_2\_WEIGHT}$$

$$A_5 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 256 & -128 & 64 & -32 & 16 & -8 & 4 & -2 & 1 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 \\ 4^8 & 4^7 & 4^6 & 4^5 & 256 & 64 & 16 & 4 & 1 \\ 256 & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -2 & 4 & -8 & 16 & -32 & 64 & -128 & 256 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} 6-=7 \\ 2-=5 \\ \Longrightarrow \\ 4-=9 \\ 4-=(2^{16})1 \\ 8-=3 \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -256 & 0 & -64 & 0 & -16 & 0 & -4 & 0 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 \\ 0 & 4^7 & 4^6 & 4^5 & 256 & 64 & 16 & 4 & 0 \\ 256 & 128 & 64 & 32 & 16 & 8 & 4 & 2 & 1 \\ 0 & -2 & 0 & -2 & 0 & -2 & 0 & -2 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & -4 & 0 & -16 & 0 & -64 & 0 & -256 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{matrix} 6 \gg (1) \\ (2)5+=2 \\ \Longrightarrow \\ 2/=(-1) \\ 8/=(-1) \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 256 & 0 & 64 & 0 & 16 & 0 & 4 & 0 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 \\ 0 & 4^7 & 4^6 & 4^5 & 256 & 64 & 16 & 4 & 0 \\ 512 & 0 & 128 & 0 & 32 & 0 & 8 & 0 & 2 \\ 0 & -1 & 0 & -1 & 0 & -1 & 0 & -1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 4 & 0 & 16 & 0 & 64 & 0 & 256 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} 7+=6 \\ \Longrightarrow \\ 6/=(-1) \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 256 & 0 & 64 & 0 & 16 & 0 & 4 & 0 \\ 1 & 2 & 4 & 8 & 16 & 32 & 64 & 128 & 256 \\ 0 & 4^7 & 4^6 & 4^5 & 256 & 64 & 16 & 4 & 0 \\ 512 & 0 & 128 & 0 & 32 & 0 & 8 & 0 & 2 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 4 & 0 & 16 & 0 & 64 & 0 & 256 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{matrix} 3-=7 \\ 5-=(512)7 \\ \Longrightarrow \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 256 & 0 & 64 & 0 & 16 & 0 & 4 & 0 \\ 0 & 2 & 3 & 8 & 15 & 32 & 63 & 128 & 255 \\ 0 & 4^7 & 4^6 & 4^5 & 256 & 64 & 16 & 4 & 0 \\ 0 & 0 & -384 & 0 & -480 & 0 & -504 & 0 & -510 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 4 & 0 & 16 & 0 & 64 & 0 & 256 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} (2)3-=8 \\ 7-=1 \\ \Longrightarrow \\ 7-=9 \\ 8+=2 \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 256 & 0 & 64 & 0 & 16 & 0 & 4 & 0 \\ 0 & 0 & 6 & 0 & 30 & 0 & 126 & 0 & 510 \\ 0 & 4^7 & 4^6 & 4^5 & 256 & 64 & 16 & 4 & 0 \\ 0 & 0 & -384 & 0 & -480 & 0 & -504 & 0 & -510 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 260 & 0 & 80 & 0 & 80 & 0 & 260 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{matrix} 5+=3 \\ 8-=(80)6 \\ \Longrightarrow \\ 3-=(510)9 \\ 4-=2 \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 256 & 0 & 64 & 0 & 16 & 0 & 4 & 0 \\ 0 & 0 & 6 & 0 & 30 & 0 & 126 & 0 & 0 \\ 0 & 16128 & 4096 & 960 & 256 & 48 & 16 & 0 & 0 \\ 0 & 0 & -378 & 0 & -450 & 0 & -378 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 180 & 0 & 0 & 0 & 0 & 0 & 180 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} (3)3+=5 \\ 8/=(180) \\ \Longrightarrow \\ 5+=(378)7 \\ 2 \gg (2) \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 64 & 0 & 16 & 0 & 4 & 0 & 1 & 0 \\ 0 & 0 & -360 & 0 & -360 & 0 & 0 & 0 & 0 \\ 0 & 16128 & 4096 & 960 & 256 & 48 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & -72 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{matrix} 6-=2 \\ 5/=(-72) \\ \Longrightarrow \\ 3/=(-360) \\ 2-=8 \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 63 & 0 & 16 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 16128 & 4096 & 960 & 256 & 48 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -63 & 0 & -15 & 0 & -3 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{matrix} 7-=3 \\ \Longrightarrow \\ 4-=(256)5 \\ 3-=5 \end{matrix} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 63 & 0 & 16 & 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 16128 & 4096 & 960 & 0 & 48 & 16 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -63 & 0 & -15 & 0 & -3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{matrix}4-=(4096)3\\4-=(16)7\\\Longrightarrow\\4+=(256)6\end{matrix}\begin{pmatrix}1&0&0&0&0&0&0&0\\0&63&0&16&0&4&0&0\\0&0&1&0&0&0&0&0\\0&0&0&-2880&0&-720&0&0\\0&0&0&0&1&0&0&0\\0&-63&0&-15&0&-3&0&0\\0&0&0&0&0&0&1&0\\0&1&0&0&0&0&0&1\\0&0&0&0&0&0&0&1\end{pmatrix}\begin{matrix}6+=2\\\Longrightarrow\\(180)2+=4\end{matrix}\begin{pmatrix}1&0&0&0&0&0&0&0\\0&11340&0&0&0&0&0&0\\0&0&1&0&0&0&0&0\\0&0&0&-2880&0&-720&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&1&0\\0&1&0&0&0&0&0&1\\0&0&0&0&0&0&0&1\end{pmatrix}$$

$$\begin{matrix}2/=(11340)\\4+=(720)6\\\Longrightarrow\end{matrix}\begin{pmatrix}1&0&0&0&0&0&0&0&0\\0&1&0&0&0&0&0&0&0\\0&0&1&0&0&0&0&0&0\\0&0&0&-2160&0&0&0&0&0\\0&0&0&0&1&0&0&0&0\\0&0&0&1&0&1&0&0&0\\0&0&0&0&0&0&1&0&0\\0&1&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&0&1\end{pmatrix}\begin{matrix}4/=(-2160)\\6-=4\\\Longrightarrow\\8-=2\end{matrix}I_9$$