arXiv:2602.09577v1 [cs.SC] 10 Feb 2026

关于半域的计算探索

Jean-Guillaume Dumas* Stefano Lia† John Sheekey‡

2026年2月11日

摘要

有限半体是有限域上的除代数，其中乘法不一定满足结合律。本文研究小型半域和有限域扩张中乘法的复杂度。对于这一运算，所需基域乘法的数量就是张量秩，即乘法复杂度。其他基域运算包括加法和常数数乘，我们将其统称为加法复杂度。在递归使用时，张量秩决定指数，而其他运算决定相关渐近复杂度下界中的常数。对于较小的扩张，这两种度量具有相近的重要性。

本文确定了特征 2 和 3 的一些半域及有限域的张量秩。我们还提出了关于其加法复杂度的新的上界和下界，并给出了新的相关算法，在总体复杂度方面改进了当前最先进的结果。为此，我们考虑了用于对具有给定参数的线性码进行编码的短直线程序。

本研究部分获得了PHC ULYSSES 计划（Campus France / Taighde Éireann – Research Ireland）的资助，资助编号为50216SG；此外还获得了法国国家研究署的资助，资助编号为ANR-22-PECY-0010。

1 引言

在有限域上的代数中进行乘法运算的计算复杂度，是一个已被广泛研究的问题。著名的例子包括矩阵乘法、多项式乘法以及有限域扩张中的乘法。近年来，非结合除代数（半域）也受到关注，其应用包括编码理论 [8], 密码学 [31] 或图论 [25]。

研究最为关注的通常是乘法复杂度（张量秩），尤其是在递归实现方面。此外，例如对于小型有限域上的多项式乘法，理论上最快的实现方式可能是预先穷举计算所有可能多项式的运算结果，并将其存入表中。

然而，在嵌入式密码学中，至少自 [19], 以来，确保侧信道抗性的一种方法就是避免分支和查表，因为它们可能泄露有关秘密信息的大量信息（例如通过计时或缓存攻击，参见 [18, 2]）。在这种情况下，所谓的恒定时间实现（即不依赖于相同长度输入的实际取值）通常会退而采用多项式算术。因此，尽可能减少域运算的总次数十分重要。另一个值得关注的领域是在更大的有限域中实现密码学运算，并将其构造为扩张塔。在这种情况下，基域运算可以通过查表实现，但扩张也会直接使用模多项式运算。同样，为了获得快速实现，减少乘法和加法的次数也至关重要；例如参见 [14]。

通过插值（Toom-Cook 方法），当 $q \geq 2n-1$ 时，域 $\mathbb{F}_{q^n}$ 中乘法的张量秩以上界 $2n-1$ [33] 为限。当 q 较小时，该问题在一般情况下仍未解决。该

*格勒诺布尔阿尔卑斯大学 让·孔茨曼实验室，法国国家科学研究中心第5224联合研究单位，法国格勒诺布尔F38058。 Jean-Guil-laume.Dumas@univ-grenoble-alpes.fr

†于默奥大学，数学与数理统计系，瑞典于默奥901 87。 Stefano.Lia@umu.se

‡都柏林大学学院，数学与统计学院，爱尔兰都柏林贝尔菲尔德4区。 John.Sheekey@ucd.ie

算法（出自 [1, 9] ）随后可以找到最优公式（直线程序， SLP），用以计算有限域上的双线性映射。现在，在 [23] 中已经证明，存在乘法复杂度低于同阶域的半域。然而，由于半域乘法通常不能递归应用，仍需考虑总体复杂度。基域的其他运算是加法和常数缩放，我们将其合称为加法复杂度。据我们所知，关于加法复杂度的了解要少得多，即便最优的 SLP针对某些 Toom-Cook 公式的结果已在 [3, 27]中给出。

本文考察低维、低特征的代数，运用计算和理论工具，改进若干情形下已知的最佳复杂度。

贡献。 具体而言，我们首先在第 2 节介绍双线性映射作为张量和矩阵子空间的不同表示，并改进加法复杂度的随机化搜索。接着，我们在第 3 节确定阶 243 的代数的张量秩。由于有限域或半域中的乘法可通过矩阵子空间编码，我们还在第 4.1 节研究小型线性码的生成矩阵上的矩阵-向量积的最短直线程序上界，其中线性码定义在 $\mathbb{F}_2$ 和 $\mathbb{F}_3$ 上。我们在第 5 节继续探索加法复杂度，定义一种用于计算模另一多项式的多项式乘法的折叠技术。最后，我们在第 6 节确定 81阶代数的加法复杂度，并在第 7 节改进阶 32 和阶 243 的代数的加法复杂度。由此，我们还得以改进 $\mathbb{F}_3$ 上 5 次扩域中的乘法算法。

2 预备知识

让我们将 $\mathbb{F}^n$视为 作为域 $\mathbb{F}$上的列向量空间，并将$(\mathbb{F}^n)^*$ 作为行向量空间。

定义1. 考虑 $T \in (\mathbb{F}^n)^* \otimes (\mathbb{F}^n)^* \otimes F^n$。 T的张量秩是使得存在分解 $T = \sum_{i=1}^{r} L_i \otimes R_i \otimes P_i$的最小整数 r。

定义2. 张量分解 $T = \sum_{i=1}^{r} L_i \otimes R_i \otimes P_i$ 的 LRP 表示由三个矩阵构成，即 $\mathbb{F}^{r\times n}$中的 L、 $\mathbb{F}^{r\times n}$中的 R 以及 $\mathbb{F}^{n\times r}$中的 P，使得 L_i 是 L的第 i行， R_i 是R的第 i行，且 P_i 是 P的第 i列。

L 是某个 LRP表示的左矩阵， R 是右矩阵， P 是后矩阵（或乘积矩阵）。我们将记为 $T = T(L, R, P)$。

由张量（分解）可以在 $\mathbb{F}^n$ 上定义一种双线性但不一定满足结合律的乘法。

$$x \star_T y := \sum_{i=1}^{r} (L_i \cdot x)(R_i \cdot y)P_i.$$

我们可以用矩阵 L、 R、 P 将其表示为

$$x \star_T y = P(Lx \odot Ry),$$

其中， $\odot$ 是向量的哈达玛（逐元素）乘积。因此，计算 $x\star_T y$ 的总体复杂度由张量秩 r 以及计算三个矩阵-向量积 Lx、 Ry 和 P（$Lx \odot Ry$）所需的运算次数决定。换言之，我们要优化张量分解的秩以及相关直线程序的复杂度。

2.1 域与半域

一个 $\mathbb{F}$-双线性乘法 $\star$ 在 $\mathbb{F}^n$ 上定义了一个预半域，当且仅当 $x \star y = 0$ 当且仅当 $x = 0$ 或 $y = 0$。等价地，以下形式的每个线性方程 $a\star x = b$或 $y\star a = b$都有唯一解。注意，我们不假设 $\star$满足任何结合律或交换性。如果 $\star$ 具有单位元，我们称其为半域。如果它进一步同时满足结合律和交换性，则它是一个域。背景介绍请参见[22] 。

对于某些较小参数，有限半体已完成分类，并且已有许多构造；例如参见 [8, 表 4], [29], [32], [22]。从张量的角度研究半域始于[24] ，并在 [20]中得到进一步发展。

在 [23] 中，我们证明了存在张量秩（乘法复杂度）低于同阶域的半域。这表明，在无需满足域的结合律的应用中，半域或许能带来更好的性能。在这项工作中，我们考察小阶域和半域的整体复杂度。

2.2 同位与自同位

我们通过可逆线性映射定义张量上的作用。尽管该定义不依赖于分解，但直接使用分解对我们来说会更方便。

定义 3。设 $T = \sum_{i=1}^{r} L_i \otimes R_i \otimes P_i$，并令 (X, Y, Z) 为一组三个可逆线性映射。则我们定义

$$T^{(X,Y,Z)} := \sum_{i=1}^{r} (L_i X) \otimes (R_i Y) \otimes (Z P_i)$$

若存在三元组 (X, Y, Z) 使得 $S = T^{(X,Y,Z)}$，则称两个张量 T和 S为同位的。若 $T^{(X,Y,Z)} = T$，则称三元组 (X, Y, Z) 是 T的自同位。

注意， $T(L, R, P)^{(X,Y,Z)} = T(LX, RY, ZP)$。张量 T定义一个（预）半域，当且仅当 $T^{(X,Y,Z)}$ 也定义一个（预）半域。同位是半域和张量的一种自然等价概念，此外还有 S_3 因子置换的作用。

[17] 的方法包括这样一种可能性：利用同位将 T 替换为另一个加法复杂度较低的张量；当递归使用乘法时，可以节省开销，因为基变换（X, Y, Z）的代价较低。然而，对于固定代数的情形，这样做的代价总是更高，因此我们需要考虑固定代数的加法复杂度，而不是利用同位。我们可以善用同位。

2.3 线性循环秩分解表示与直线程序

尽管代数的乘法复杂度在同位下保持不变，加法复杂度却不是如此。我们必须考虑 LRP 表示，并优化所需运算的次数。也就是说，我们要寻找长度最短的直线程序（SLP）。

例 4。 例如，用 $\odot$ 表示哈达玛积，下面这个用于计算次数为 1 的多项式乘法的 Karatsuba 算法变体，仅需 3 次基域乘法， $c_0 + c_1 X + c_2 X^2 = (a_0 + a_1 X)(b_0 + b_1 X) = a_0 b_0 + (a_0 b_0 + (a_0 - a_1)(b_1 - b_0) + a_1 b_1) X + a_1 b_1 X^2$，可由式 (1)表示。

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}}_{P} \left(\underbrace{\begin{bmatrix} 1 & 0 \\ 1 & -1 \\ 0 & 1 \end{bmatrix}}_{L} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} \odot \underbrace{\begin{bmatrix} 1 & 0 \\ -1 & 1 \\ 0 & 1 \end{bmatrix}}_{R} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} \right) \tag{1}$$

该 PLinOpt 软件 [11, 12, 13], 生成直线程序（SLP），并通过若干常见子表达式消除技术优化其算术成本（加法和标量乘法） [13,算法 1], ，同时进行核优化 [13, 算法 2]。

我们在此改进算法 1中的核优化。其思路是 [13, 算法 2] 将输出计算分成两个部分。从矩阵出发，选出一组线性无关的行，以及一组依赖于第一组的行。首先，为第一组生成一个优化后的 SLP ；它会 SLP计算输出的一个子集。然后，不再优化第二组，而是优化将其表示为第一组线性组合的表达式

，即输出子集的第一部分。如果线性依赖比初始行更稀疏或具有更规整的结构，那么第二个 SLP 可能会更快。 PLinOpt 随后会搜索第一组行的最佳随机选取方案。我们在此对该技术提出两项改进：

- 允许第一组中包含额外的标准基向量：计算它们不需要任何运算；并允许第二组的行由输入和第一组输出的组合表示（而原算法只能用输入或第一组输出来表示它们）；特别是，现在即使初始矩阵不满秩或列数多于行数，该方法也适用；
- 允许第一组的行数多于矩阵的秩：这同样会增加可能性，但也会扩大搜索范围。

这些技术在算法 1 中得到了形式化，并已证明对有限域和半域有用（例如，它能够在第 SLP 节 7.3 中找到一个比 [13, 算法 2] 先前找到的结果少一次运算的结果）。算法 1 现在也允许将核技术应用于列数多于行数的矩阵。

算法 1 矩阵的混合随机核分解 输入：M 属于 $\mathbb{F}^{m\times n}$，且满足 $r=$ 秩 M。输出：计算 $\vec{u}\leftarrow M\cdot\vec{v}$ （$\vec{v}\in\mathbb{F}^n$）的直线程序。1：随机选择 $k\in 0..r$；2：$G\in\mathbb{F}^{k\times n}\leftarrow$ 随机选取 M 的 k 个线性无关的行；3：$H\in\mathbb{F}^{(n-k)\times n}\leftarrow$ 随机选取维数为 n 的典范基中的 $n-k$ 个不同向量，且与 G 线性无关；4：随机选择 $j\in 0..(m-k)$；5：$S\in\mathbb{F}^{j\times n}\leftarrow$ 随机选取 $M\setminus G$ 的 j 行；6：$R\in\mathbb{F}^{(m-j-k)\times n}\leftarrow M\setminus(G\cup S)$；{$M=\begin{bmatrix}G\\R\\S\end{bmatrix}$, 最多进行行置换。}7：对 $\begin{bmatrix}G\\R\end{bmatrix}$，$P_1\leftarrow$ SLP；{通过 [13, 算法 1]}8：对 $X\in\mathbb{F}^{j\times n}$，求解 $S=X\begin{bmatrix}G\\H\end{bmatrix}$；9：对 X，$P_2\leftarrow$ SLP；10：返回 P_1，随后将 P_2 应用于：

{Via [13, Alg. 1]}

- 第一个 k 输出 P_1的；{正在计算 $G\cdot\vec{v}$}
- $\vec{v}$对应于 H的 $n-k$ 值。

2.4 张量和分解所产生的矩阵子空间

从三阶张量 T的 LRP 表示中，我们可以用一些自然的方式定义二阶张量（或矩阵）的子空间。我们将 $L_i\otimes R_i$ 与秩一矩阵 $L_i^\intercal R_i$对应起来。

$$W(L,R):=\langle L_i\otimes R_i : i\in[1,r]\rangle,$$
$$U(L,R,P):=\left\langle\sum_{i=1}^{r}P_{ji}(L_i\otimes R_i) : j\in[1,n]\right\rangle.$$

显然，我们有

$$U(L,R,P)\leq W(L,R)\leq\mathbb{F}^{n\times n},$$

注意，$U(L,R,P)$ 不依赖于特定的 LRP 表示，因此我们可以定义 $U(T)$。注意，这就是张量 T 的收缩空间之一。当且仅当 $U(T)$ 是 n 维的，且 $U(T)$ 的每个非零元素都可逆时，张量 T 定义一个域或（预）半域乘法；这样的

空间称为半体扩张集。注意，这一概念与最小距离为 n 的最大秩距离（MRD）码的概念相同。

另一方面，空间 $W(L,R)$ 确实取决于具体表示。众所周知，可以通过寻找由秩一张量张成的最小维数空间 W 来计算张量秩，使得 $U(T) \leq W$。由这样的空间 W，我们可以构造 L 和 R：选取 W 的一组秩一矩阵基 $\{B_i\}$，令 $L_i^\intercal$（分别为 R_i）为 B_i 的列空间（分别为行空间）的一个基元素，并令 P_j 为 $U(T)$ 的一组基中第 j 个元素相对于 $\{B_i\}$ 的坐标向量。

还要注意，对于任意可逆的 X,Y,Z，都有 $U(LX,RY,ZP) = X^\intercal U(L,R,P)Y$，并且 $W(LX,RY) = X^\intercal W(L,R)Y$。在这种情况下，我们称 $U(LX,RY,ZP)$ 与 $U(L,R,P)$ 等价。此外，对于任意$r \times r$置换矩阵 σ 以及任意可逆的 $r \times r$ 对角矩阵 D_1, D_2，都有 $W(\sigma D_1 L, \sigma D_2 R) = W(L,R)$，因为 σ 的作用是置换张成集 $L_i \otimes R_i$，而乘以对角矩阵则会将 $L_i \otimes R_i$ 乘以一个标量。因此，若 $L' = \sigma D_1 LX$ 且 $R' = \sigma D_2 LY$，则 $W(L,R)$ 与 $W(L',R')$ 等价。

3 乘法复杂度

3.1 张量秩

Brockett 和 Dobkin 的下述定理 [4], 结合 [15], 中关于已知最佳线性码的数据，给出了任意域或半域的张量秩下界。若 T 是一个阶为 q^n的域或半域的乘法张量，则 $L^\intercal$、$R^\intercal$和 P 都必须是 r 对 T的秩分解中的生成矩阵，且对应一个 $[r,n,n]_q$线性码。令 $A_q(n,n)$ 表示在 n 维、最小距离至少为 n 的条件下，汉明度量上的线性码的最短长度，其中底域为 $\mathbb{F}_q$。

定理 5（[4]）。在 $\mathbb{F}_q$ 上，阶为 q^n 的域或半域的张量秩至少为 $A_q(n,n)$。

当 $q \geq 2n-1$ 时，我们有 $A_q(n,n) = 2n-1$，而域 $\mathbb{F}_{q^n}$ 的张量秩为 $2n-1$。当 q 较小时，一般情形下这个问题仍未解决。

这里汇总了关于 $\mathbb{F}_2$ 和 $\mathbb{F}_3$ 上张量秩的已知结果。对于次数为 3 的扩张，这个问题已在 [21] 中得到彻底解决。关于 $\mathbb{F}_2$ 上的多项式乘法，请参见 [6]。

表 1：已知张量秩表

Set	张量秩	参考
$\mathbb{F}_{2^4}$	9	[16],[7]
$\mathbb{S}_{2^4}$ #2, #3	9	[23]
$\mathbb{F}_{3^4}$,$\mathbb{S}_{3^4}$ #9	9	[16],[20]
$\mathbb{S}_{3^4}$ #1、. . .、#8、#10、#11	8	[23]
$\mathbb{F}_{2^5} = \mathbb{S}_{2^5}$ #5	13	[26]
$\mathbb{S}_{2^5}$ #1,. . .,#4,#6	13	小节 7.2
$\mathbb{F}_{3^5}$	11	定理 6
$\mathbb{S}_{3^5}$ #2, #4	10	定理 7
$\mathbb{S}_{3^5}$ #3、#5、#6	11	定理 7
$\mathbb{S}_{3^5}$ #7, ..., #12	$\geq$11	定理 7
$\mathbb{F}_{2^6}$	15	[15],[5]
$\mathbb{F}_{3^6}$	$\in [12,15]$	[15],[5]
$\mathbb{F}_{2^7}$	$\in [18,22]$	[15],[5]
$\mathbb{F}_{2^8}$	$\in [20,24]$	[15],[5]

3.2 阶 243 的域与半域

阶为 243 [30] 的有限半体共有 12 个同位类。根据定理5，任意此类代数的张量秩至少为 $A_3(5,\ 5)\ =10$。此外，$[10,5,5]_3$ 码 [34]只有一个等价类；令 C表示其中一个这样的码，并令 G 为 C的生成矩阵。因此，相应张量的任意秩 10 分解都必须满足：$L^\intercal$、$R^\intercal$、P 是与 C等价的码的生成矩阵；也就是说，$L=\sigma DGX$ 其中 σ是某个置换矩阵，D是某个可逆对角矩阵，且 X是某个可逆矩阵。

定理 6。含有 3^5 个元素的有限域的张量秩为 11。

证明。由 [1]，可知，$\mathbb{F}_{3^5}$ 的张量秩至多为 11。因此，只需证明其张量秩不可能为 10。

我们为 $\sigma\in S_{10}$ 构造空间 $W(G,\sigma G)$。然而，我们可以进一步缩减考察范围，因为若 $\sigma DGX\ =\ G$，则 $W(G,G)$ 与 $W(G,\sigma G)$ 等价。因此，我们只需检查 S_{10} 的某个子群的陪集代表元。用 MAGMA 直接计算可知，只需检查 2520 个代表元，数量减少了 720 倍。

对于每个这样的空间，我们使用 [29]，的方法变体，穷尽搜索半体扩张集，并按同位进行分类。我们没有发现域 $\mathbb{F}_{3^5}$，从而证明了这一断言。□

接下来，我们将 [1] 的算法推广到有限半体的情形。该算法搜索满足 $L\ =\ R$ 的 LRP 分解。我们将其称为部分对称张量秩。任何具有这种分解的半体必然是交换的，因此我们将该算法应用于 [30] 中标记为 #1、$\cdots$#7 的那些半体。

定理 7. 具有 243 个元素的交换半域的部分对称张量秩如表 2 所列。

表 2：通过 [1] 得到的具有 243 个元素的半域的部分对称张量秩

Set	部分对称张量秩	# 解（基）
$\mathbb{F}_{3^5}$	11	121
$\mathbb{S}_{3^5}$ #2	10	11
$\mathbb{S}_{3^5}$ #3	11	121
$\mathbb{S}_{3^5}$ #4	10	1
$\mathbb{S}_{3^5}$ #5	11	22
$\mathbb{S}_{3^5}$ #6	11	31
$\mathbb{S}_{3^5}$ #7	12	1 310 980

这并不排除存在其他张量秩为 10 的半域；然而，以下结果表明这种情况并不会发生。

定理 8. 秩为 10 的 3^4 阶半域恰有两个类别。

证明。 沿用定理 6的证明，我们发现，与标记为 #2 和 #4 的半域所对应的扩张集等价的扩张集，确实会作为子空间出现在 $W(G,\ \sigma G)$ 中，除此之外没有其他情况。□

我们也可以利用这一设置，进一步证明不存在 $L=R$ 的部分对称算法来计算 $\mathbb{F}_{3^6}$中的乘法。这一结论已在 [1], 中通过计算得出，计算耗时 4.50×10^7 秒；而这里完全不需要进行搜索。

引理 9. 域 $\mathbb{F}_{3^6}$ 和任意阶为 3^6，的半域的部分对称张量秩至少为 13。

证明。 我们要求 $L^\intercal = R^\intercal$ 和 P 是 $[12,6,6]_3$ 码的生成矩阵。唯一的$[12,6,6]_3$ 码（在等价意义下）是扩展三元戈莱码 [28]。因此，我们可以假设$L^\intercal$ 是该码的任意生成矩阵；例如，

$$L^\intercal = \begin{bmatrix} 100000011111010000 \\ 101221001000110122 \\ \\ 000100121012000010 \\ 122101000001112210 \end{bmatrix}$$

直接计算可得 $\sum_{i=1}^{12} L_i^\intercal L_i = 0$，因此 $U(L,L)$ 是一个维数为 11 的子空间。这就要求 P 生成一个 $[11,6,6]_3$ 码，但这样的码不存在 [15]。 □

4 线性码的加法复杂度

4.1 最短直线程序

由于我们知道，域或维数为 n 的半域在 $\mathbb{F}_q$ 上的任何 LRP 分解中出现的矩阵都必须定义一个 $[r,n,\geq n]$ 码，因此有必要了解计算具有此性质的矩阵的矩阵-向量积所需直线程序的最小复杂度。

定义 10。我们用 SSLP$_q(r,k,d)$（最短直线程序）表示计算 Lv 所需的加法和标量乘法的最小数量，其中 $L^\intercal$ 生成一个 $[r,k,d]_q$ 汉明度量码。

注意，对于 $\mathbb{F}_2$ 和 $\mathbb{F}_3$，所有运算都可以通过加法实现，并将取负视为平凡运算。

引理 11。使用 r 次乘法，在 $\mathbb{F}_q$ 上计算次数为 n 的域或半域中的乘法所需的最少运算次数至少为 $rM + (3\text{SSLP}_q(r,n,n) + n - r)A$。

证明。这由 SSLP$_q$ 将 $L^\intercal$、 $R^\intercal$ 和 P 视为码的生成矩阵的定义以及转置原理得出：计算 Pv恰好比计算$P^\intercal u$ 多需要 $n-r$次运算。 □

注意，当 r 相对于 n 较大时，这可能是平凡的；例如，我们有 SSLP$_q(n^2,n,n) = 0$，它由重复码实现，该码作为多项式乘法标准算法的 L 和 R 矩阵出现。

对于较小的参数，我们可以通过例如算法1 和 PLinOpt，以给出 SSLP$_q(r,\ n,\ n)$的值的上界；穷举搜索仅在极小的情形下才可行。

表 3：若干线性码的最短直线程序表

$[r,k,d]_q$	SSLP	参考文献
$[8,4,4]_2$	6	定理18
$[9,4,4]_2$	5	引理 22
$[10,4,4]_2$	4	引理 23
$[13,5,5]_2$	8	引理 21
$[8,4,4]_3$	6	定理 18
$[9,4,4]_3$	5	引理 22
$[10,4,4]_3$	4	引理 23
$[10,5,5]_3$	$\in [5,12]$	引理 16 和表7
$[11,5,5]_3$	≤ 12	表 7
$[12,5,5]_3$	≤ 12	表 7
$[13,5,5]_3$	≤ 11	表 7

我们可以搜索示例 穷举或随机寻找一个高效的 $[r,k,d]_q$ 码的直线程序，如下所示 .

- 从由 $\mathbb{F}_q^k$ 中的标准基向量构成的矩阵开始。
- 添加一行，使其为前两行的和或差，或重复前面的一行。
- 重复此过程，直到获得最小距离至少为 n 的码，或达到选定的最大行数 $r_{max} \geq r$ 。

- 计算通过选取每个包含 r 行的多重集所得到的码的最小距离。如果该距离至少为 d，则 SSLP$_q(r,k,d)$ $\leq r_{max} - k -$ #重复行。

请注意，整体复杂度最低不一定由乘法复杂度最低的算法实现，也不意味着存在 $[r,n,n]_q$ 码就必然存在张量秩为 r 的算法。线性码的 SSLP 达到最小值，也不一定意味着它能出现在某个域或半域的 LRP表示中。

4.2 计算小参数下的 SSLP$_q(r,k,d)$

本节将证明关于某些码的最短直线程序的一些理论结果。我们先介绍一些一般性结果。

引理12. 重量为 v s 的向量无法用少于 $s-1$ 次加法计算。

证明. 假设在计算 v的过程中，向量 v_1、…、 v_l 已通过前 l 次加法计算得到。 S_1、…、 S_l这些 v_1、…、 v_l 的支撑集在包含关系下构成偏序。对每个 l，定义 $R_l = \{S_{i_1}, \ldots, S_{i_t}\}$为 S_1、…、 S_l中的极大元集合。假设 v通过 k次加法得到，即 $v = v_k$，且 k 为可能的最小值，也就是说， v 无法通过少于 k 次加法得到。在这种情况下，必须有 $R_k = \{\text{Supp}(v_k)\}$ 。确实，若不然，就会有一些加法被用于计算我们不需要为计算 v 而计算的坐标，这样我们就能用严格少于 k 次加法计算出 v 。现在，每次加法都会使 R_l的大小增加、减少或保持不变。由于$\{S_1, \ldots, S_l\} \subseteq \{S_1, \ldots, S_{l+1}\}$，且只进行了另一次加法，因此仅有以下几种可能：

1. $|R_l| = |R_{l+1}|$。在这种情况下， S_{l+1}要么包含 $\{S_1, \ldots, S_l\}$之一，要么被其包含。使用这种加法，最多计算出一个新的坐标。如果情况并非如此， $S_l = \{i,j\}$，其中i， $j \notin \cup_{i=1}^{l} S_i$。由于不可能出现 $S_i \subseteq \{i,j\} = S_{l+1}$的情况， $S_{l+1} \in R_{l+1}$，这与 $|R_l| = |R_{l+1}|$矛盾。2. $|R_{l+1}| = |R_l| + 1$。在这种情况下， S_{l+1}不包含于任何一个 $\{S_1, \ldots, S_l\}$中，并且至多有两个坐标不属于 $\cup_{i=1}^{l} S_i$。 3. $|R_{l+1}| = |R_l| - 1$。在这种情况下， S_{l+1}是两个元素 S、 $S' \in R_l$的不交并。此外，无法计算出任何新坐标，即 $\cup_{i=1}^{l} S_i = \cup_{i=1}^{l+1} S_i$。

由于 $R_k = \{\text{Supp}(v_k)\}$，对于除第一次以外的任意一次加法，设为第 i次，且满足 $|R_i| = |R_{i-1}| + 1$，必定存在某个操作，设为第 j次，且满足 $|R_j| = |R_{j-1}| - 1$。令 k_l表示截至第 l次加法，在某个向量中计算出的坐标数，即 $k_l = |\cup_{i=1}^{l} S_i|$。显然， $k_0 = 0$ 且 $k_1 = 2$。设第一次加法之后，第一类加法还有 a 次，第二类有 b 次，第三类有 c 次。于是 $k_{a+b+c+1} \leq a + 2b$。由于 $b = c$， $k_{a+b+c+1} \leq a + b + c$，这就证明了该结论。 □

推论 13. 若向量 v 的重量为 s ，且恰好通过 $s-1$ 次加法计算得到，那么计算出的所有 $s-1$ 个中间向量除符号外，共享相同的非零项。换言之，支撑集任意两个中间向量的支撑集交集所对应的子矩阵秩为一。

证明。仿照引理12的证明，记 $s - 1 = 1 + a + b + c$，其中 a、 b、 c分别表示第一次加法之后执行的类型1、2和3加法的次数。于是，每次类型（1）加法都必须实际计算一个新坐标，而每次类型（2）加法都必须将两个新坐标相加；否则，在这两种情况下，我们都会得到重量$(v) \leq s - 1$。这意味着，类型（1）加法（例如第 $l+1$步）形式为 $v_i + E_j$，其中 $i \leq l$、 E_i 是典范基中的一个向量，并且第 i个坐标不属于 $\cup_{i=1}^{l}$ Supp(v_i)。类似地，类型（2）加法（例如第 $l+1$步）形式为 $E_i + E_j$，其中$\{i,j\} \cap (\cup_{i=1}^{l}\text{Supp}(v_i)) = \emptyset$。 □

以下结果针对 $[2k,k,k]_q$码，其中 $q\in\{2,3\}$。注意，根据Griesmer界，若存在一个满足 $q\in\{2,3\}$ 的 $[2k,k,k]_q$码，则 $q=2$ 且 $k\le 4$，或者 $q=3$ 且 $k\le 6$。

引理14. 设 $G=(I_k|M)$ 是一个 $[2k,k,k]_q$码的生成矩阵，其中 I_k 是 $k\times k$单位矩阵，且 $q\in\{2,3\}$。那么，如果 M 有重复的列，则无法通过 $2k-3$ 次加法计算出 G 的各列。此外，典范基中没有向量会在 G 中重复。

证明. 第二个结论很容易验证，因此我们只证明第一个结论。反设我们可以用 $2k-3$ 次加法计算出 G，并且不失一般性地假设前两列相等，即 $M^1=M^2$。由于 G 中没有重量小于 k 的码字， M 中没有任何一行含有两个 0；而且不存在指标 i、j，使得 M 的三列在行 i 和 j 中的坐标相同（否则它们的差的重量小于 k）。此外， M 中没有任何一列含有两个 0。事实上，如果 M 的某一列在行 i、j 上的值为 0，那么 M 的其余 $k-1$ 列在行 i、j 上都将有非零项，而行 i、j 的某个线性组合的重量至多为 $k-1$。特别地，由于 M^1 重复出现，其重量为 k。由于 M 中没有行或列能含有两个零项， M 各行中至多有 $k-2$ 个零项，因此 M 中至少有两行的重量为 k。将 G 中对应的行作线性组合，可得到一个重量小于 k 的码字，矛盾。 □

引理15. 设 $G=(I_k|M)$ 是一个 $[2k,k,k]_q$码的生成矩阵，其中 I_k是 $k\times k$单位矩阵，且 $q\in\{2,3\}$。则无法通过 $2k-3$ 次加法计算出 G 的列。等价地，计算 G 至少需要 $2k-2$ 次运算。

证明. 反设我们可以通过 $2k-3$ 次加法计算出 G。按照引理14的证明方法，可以假设 M 的每一行都没有两个 0 项；不存在两个不同的指标 i、j，使得三个不同的列 M 在行 i、j 中具有相同的坐标；并且 M 的每一列都没有两个零项。因此，每一列的重量至少为 $k-1$，根据引理12，计算每一列至少需要 $k-2$次加法。假设 M的每一列的重量均为 k。那么，根据引理12，使用前 $k-1$ 次加法至多能计算出 M 的一列，而计算 M 中剩余的 $k-1$ 列，每列还至少需要一次加法（这些列的重量都达到最大值，但必须彼此不同——由其元素的符号可知）。因此，至少需要 $k-1+k-1=2k-2$ 次加法，这就导出了矛盾。因此可以假设 M 至少有一列含有零项，并且在第 $k-2$次加法时， M 的一列已经计算出来。不失一般性，设用前 $k-2$次加法计算出的 $k-2$ 个向量为 P^1、.....、P^{k-3}、$P^{k-2}=M^1=\begin{pmatrix}\star & \dots & \star & 0\end{pmatrix}^\intercal$，其中 $\star$表示一个非零标量。根据推论13，所有向量 P^1、.....、P^{k-2}在相差一个非零标量倍数的意义下，共享所有非零坐标。由于前 $k-2$ 次加法中只计算出了列 M^1，接下来的每一次加法都必须计算出 M的一列新列。由于 M 的任何一行都没有两个零项，下一次加法必须将一个非零项作为最后一个坐标相加。特别地，M^1和 M^2除了一个项之外，其余各项都相同（至多相差符号）。每个 M^i要么由 P^1、.....、P^{k-3}、$P^{k-2}=M^1$、M^2中的两个向量组合得到，要么由其中一个向量与典范基的一个向量组合得到。无论哪种情况，它们都必须与 M^1至少共享两个坐标。现在，根据上述性质， M^3与 M^1至少共享两个非零项，因此也与 M^2共享两个非零项，这与不存在两个不同的指标 i、j，使得 M的三列在行 i、j中具有相同项这一事实矛盾。

□

引理16. G的生成矩阵是一个 $[2k,k,k]_q$码，其中 $q\in\{2,3\}$ 且 $k\ge 4$，它没有重复列。

证明. 设 C 是由 G生成的 $[2k,k,k]_q$ 码。反设 G 有两列相同。对该码关于这两列进行缩短；也就是说，取 C 中在这两个位置上均为零的所有元素构成的余维为1的子码。删除这两个位置后，得到一个 $[2k-2,k-1,\ge k]_q$码。特别地，这是一个MDS码。对于 $q\in\{2,3\}$，维数至少为3的MDS码的最小距离为 $d\le 2$，因此不存在这样的码。 □

定理 17. Th 码的生成矩阵 G 需要至少 6 次加法，其中 $q\in\{2,3\}$。 *s.*

证明。 如果 G 是系统形式的，引理 15 即可证明该结论。反设 G 可以通过 5 次加法计算得到。根据引理 16，剩下的唯一可能是 G 的 3 列是典范基的向量，并且每次加法都计算出 G的一列。不失一般性，设 $G = (I_{4|3} \mid M)$，其中 $I_{4|3}$ 是一个 4×3 矩阵，其列为典范基的前三个向量。

那么， M 的至少一列，记为 M^1，至少有两个零元（否则 5 次加法中的第一次无法实现满列，而剩下的 4 次加法只能实现 5 个不同的输出）。首先，若 M^1 最后一行的元素为零，则不失一般性，$M^1 = \begin{pmatrix} 1 & \square & 0 & 0 \end{pmatrix}^\intercal$，其中 $\square$ 表示一个未知元素，可能为零。由于 G 生成一个最小距离为 4 的码， M 的任何一行都没有 3 个零元。特别地，对于 $i = 2$、3、4、5，我们可以假设 $M^i = \begin{pmatrix} \square & \square & \square & 1 \end{pmatrix}^\intercal$。若 $M^2 = \begin{pmatrix} \star & \star & 0 & 1 \end{pmatrix}^\intercal$，其中 $\star$ 表示一个未知的非零元素，则 M^i、 $i = 3$、4、5 中每一个的第三个坐标都必须非零。在这种情况下，第三行与第四行的一个线性组合的重量将小于 4。若 M^i、 $i = 3$、4、5 中任何一个的第三个坐标为零，同样的推理也适用。因此，M^i、 $i = 3$、4、5 中至少有两个具有相同的第三个坐标，从而第三行与第四行的一个线性组合的重量将小于 4，这同样是不可能的。

其次，不失一般性，需要考虑的最后一种情况是 $M^1 = \begin{pmatrix} \star & 0 & 0 & 1 \end{pmatrix}^\intercal$。在 M 的其余列中，第四个坐标为 0 的至多有一列。假设有一列如此，不失一般性，设它是 M^2。首先假设 $M^2 = \begin{pmatrix} 0 & y & 1 & 0 \end{pmatrix}^\intercal$。注意， M^i、 $i = 3$、4、5 满足 $M^i = M^1 + something$，因为它们在第四个坐标上为 1。如果 $M^i = M^1 \pm E^1$，那么通过组合第二行和第三行，我们可以得到一个重量小于 4 的码字。如果 $M^i = M^1 \pm E^j$、 $j = 2$、3，那么第一行加上或减去第四行的重量小于 4。现在假设 $M^2 = \begin{pmatrix} y & 0 & 1 & 0 \end{pmatrix}^\intercal$。记住， M^i、 $i = 3$、4、5 满足 $M^i = M^1 + something$，因为它们在第四个坐标上为 1。由于目前第二行中只有一个非零项，它们都必须是 $M^i = M^1 \pm E^2$ 的形式。在这种情况下，组合第二行和第四行将得到重量小于 4 的结果。因此， G 无法通过 5 次加法计算得出。

□

定理 18. 当 $q \in \{2, 3\}$ 时， $[8, 4, 4]_q$ 码的生成矩阵的最短直线程序为 6；也就是说，$\mathsf{SSLP}_q(8, 4, 4) = 6$, $q \in \{2, 3\}$。

证明。 根据定理 17，$\mathsf{SSLP}_q(8,4,4) \geq 6$，其中 $q \in \{2, 3\}$。另一方面，式（2）中的以下矩阵可以通过 6 次加法实现：

```
o0:=i0; o1:=i1; o2:=i2; o3:=i3; t4:=i0+i2; t5:=i1+i3;
o4:=i1+t4; o5:=i0+t5; o6:=i3+t4; o7:=i2+t5;
```

$$\begin{bmatrix} 1&0&0&0&1&1&1&0 \\ 0&1&0&0&1&1&0&1 \\ 0&0&1&0&1&0&1&1 \\ 0&0&0&1&0&1&1&1 \end{bmatrix}^\intercal \tag{2}$$ □

两个其他线性码的 SSLP_2，它们定义在 $\mathbb{F}_2$ 上，维数为 $4, [9, 4, 4]_2$，距离为 $[10, 4, 4]_2,$，见附录 A。

5 扩张的折叠

我们现在转向小型有限域扩张。与其将有限域乘法实现为先进行多项式乘法、再进行多项式取余，不如直接考虑完整算法的 L、 R、 P表示。这通过对乘法的 P 矩阵进行折叠而得到：考虑LRP 表示，对次数为 d 的不可约多项式取模，就是将第$d + 1$行之后的每一行加到前面的各行上（乘以模该不可约多项式后的相应系数）。细节见算法 2。显然，看待算法 2 的另一种方式是：它是模 $I(X)$约化的（Sylvester）$(d+1)\times($ $2d+1)$矩阵（即由第 3 行中的系数列增广的单位矩阵）与乘法的$(2d + 1) \times r$ P矩阵之间的矩阵乘法。由此可直接得到有限域乘法的一个 SLP。现在的问题是找到生成 SLP 且运算次数最少的不可约多项式。

算法 2 折叠扩域乘法

输入： P，次数为 d 的多项式乘法的后矩阵；
输入：次数为 $d+1$ 的不可约多项式 $I(X)$。
输出： $P^{(I)}$，模 $I(X)$ 的多项式乘法的后矩阵。

```
令 P^(I)_{0..d,*} = P_{0..d, *};                                  {前 d+1 行}
对 i = d+1 到 2d 执行
    M = X^i 对 I(X) 取模 = Σ_{j=0}^{d} m_j X^j;
    对 j = 0 到 d 执行
        P^(I)_{j,*} += m_j P_{i, *};                  将 X^i 的系数 { 加到第 j 行, ·m_j }
    结束循环
结束循环
```

6 加法复杂度；4 次扩张

6.1 81阶有限域

对于 $\mathbb{F}_{3^4}$，张量秩为 9，递归使用两层 Karatsuba 算法。根据例 4，LRP 为： $L^{\otimes^2} \in \mathbb{F}_3^{9\times 4}$，而 $P^{\otimes^2} \in \mathbb{F}_3^{7\times 9}$ 的折叠则通过算法 2 在 $\mathbb{F}_3^{4\times 9}$ 中完成。我们测试了 $\mathbb{F}_3$ 上全部 18 个 4 次不可约多项式，发现最佳者为 $1+X+X^2+X^3+X^4$，得到一个 SLP，其后矩阵仅需 11 次运算，见 PLinOpt/data/3o3o3_F81-Karatsuba-9-21_{L,R,P}.s{ms,lp}。因此，加法复杂度上界为 $5+5+(6+(9-4)) = 21$，详见表 4。这意味着我们可以在 $\mathbb{F}_{3^4}$ 中进行乘法，总共只需 30 次运算。

注 19. 可能存在一种 LRP 分解，使 $\mathbb{F}_{3^4}$ 只需 $5+5+10=20$次加法，因此总共需要 29 次运算。我们计算了所有使用 5 次加法并给出 $[9,4,4]_3$码的可能矩阵。忽略置换后，这类矩阵共有 4104 个（其中有两个在码等价意义下不同）。目前，对所有 L/R对进行穷举搜索的代价过高。

6.2 81阶半域

由 [23], 可知，存在张量秩为 8 的 81 阶半域。根据引理 11和定理18，我们得到 $30 = 8+6+6+(6+8-4)$ 次运算是使用 8 次乘法的算法所需运算次数的下界。以下例子表明，该下界可以达到。

定理 20. 存在一个阶为 3^4 的预半域，其乘法复杂度最优，为 8；对于使用 8 次乘法的算法，其加法复杂度也达到最优。

证明。 参见 PLinOpt/data/3o3o3_S81_8_22_{L,R,P}.s{ms,lp}以获取 LRP 预半域的矩阵。可利用附录 D 中的矩阵验证，它与 $\mathbb{S}_{3^4}$#1 同位，标记遵循[10]。

不计符号时，这些 LRP 矩阵与式（2）中的矩阵相似。因此 Lx 与 Ry 均可各用 6 次加法实现，并且根据转置原理， Pz 可用最优运算次数 $6+(8-4) = 10$ 实现。这达到了引理 11 的界。 □

总的来说，我们得到表 4 ，总结了 81 个元素的扩张的已知复杂度界限。再次说明，可能存在一种算法，能用 9 次乘法和 20 次加法计算预半域，但我们尚未找到这样的例子。

表 4：81 个元素扩张中的乘法

	rank	ADD	总计
$\mathbb{F}_{3^4}$ 标准模 X^4+X-1	16	0+0+15	16M+15A
$\mathbb{F}_{3^4}$ Karatsuba 算法$^{\otimes\ 2}$（模 $1+X+X^2+X^3+X^4$）	9	5+5+11	9M+21A
$\mathbb{S}_{3^4}$ 这里	8	6+6+10	8M+22A

7 加法复杂度；5 次扩域

7.1 计算 4 次多项式的乘积

如第 5 节所示，为了实现 5 次扩域中的乘法，我们考虑其 LRP表示，这种表示是通过折叠 P 矩阵得到的。因此，我们首先考察 4 次多项式的乘法。

对此，标准算法的秩为 25，而 Toom-Cook-5插值方法的张量秩为 $2d-1=9$，但要求域中至少有 9 个不同的点。对于非常小的域，可以采用 Karatsuba方法的变体，论文 [26] 提出了一种秩为 13 的算法。文献 [14]随后比较了这些方法的总运算次数 SLP。

我们构造了 LRP 矩阵，并使用PLinOpt对其进行了优化。我们找到的程序所需运算次数略少于 [14]中给出的程序。蒙哥马利方法的算法和 SLP 见附录B，文件位于： PLinOpt/data/4o4o8_Montgomery-13-58_{L,R,P}.s{ms,lp}。现在，Toom-5方法的算法已达到最优，相关内容见 [3]，，文件位于：PLinOpt/data/4o4o8_Toom5_{L,R,P}.s{ms,lp}。表5列出了得到的运算次数（秩表示有限域元素乘法的次数，ADD表示加法次数，SCA表示常数乘法次数，3个元素之和则列出了 L、 R 或 P 矩阵中各自的具体情况）。对于这两种方法的L矩阵和R矩阵， PLinOpt 分别得到 $11ADD$ 和 $19ADD+11SCA$，这与 [26, 3]的预期一致。对于P矩阵， [26] 指出：“处理13个乘积的输出时还会出现其他重复子表达式，不过很难统计。” PLinOpt 得到一个SLP ，包含31次加法和5次数乘，比 [14]少4次运算。另一方面，对于Toom-5方法， PLinOpt的部分搜索仅找到 $35ADD+28SCA$ 次运算，而通过穷举搜索找到的 [3, A.3],的最优 SLP 需要 $32ADD+21SCA$ 次运算。

表5：4次多项式乘法

	rank	ADD	SCA	总计	EGI [14]
标准	25	0+0+16	-	25M+16A	25M+16A
Montg.	13	11+11+31	0+0+5	13M+58A	13M+62A
Toom-5	9	19+19+32	11+11+21	9M+113A	9M+140A

注意，在 [14], 运算次数的统计方式略有不同，即对某些运算组合赋予了不同的权重。表中列出的数值是通过对每项运算[1]进行计数得到的。

[1] [14] 对数乘和移位采用了不同的计数方式：对于 L 和 R，运算次数为 $11ADD+8SHIFTS+4SCAL$；对于 P，则为 $28ADD+11SHIFTS+11SCAL+28A$ $SHIFT\approx\frac{2}{3}ADD$ $SHIFT$ $ADD\approx$。不过，他们假设实际中，然后 &，并且，因此他们的 $\frac{5}{4}ADD$ $SCAL\approx 2ADD$ $\approx 9M+137A$ 计数实际上是 。

7.2 32 阶域与半域

对于$\mathbb{F}_{2^5}$，我们考察了 $\mathbb{F}_2$上的 6 个五次不可约多项式。我们发现 $1+X^2+X^5$ 对标准算法更有意义，而 $1+X+X^2+X^4+X^5$ 对 Montgomery 算法更好。相关算法可在PLinOpt/data/4o4o4_F32_Standard_{L,R,P}.s{ms,lp}以及PLinOpt/data/4o4o4_F32_Montgomery_{L,R,P}.s{ms,lp}中找到。总体而言，如表6所示，两种算法的运算次数相同，但 Montgomery 版本的基域乘法次数更少。

其余 32 阶半域的张量秩均为 13。表示方式太多，无法穷尽检查；有限搜索也未找到任何能与 LRP 相媲美的例子；只找到一个多用 3 次运算的例子，见 $\mathbb{F}_{2^5}$，如表 PLinOpt/data/4o4o4_S32_13_#2_{L,R,P}.s{ms,lp}的最后一行所示6。

引理 21. 有 $\mathrm{SSLP}_2(13,5,5) = 8$。因此，在 32 阶域或半域中进行乘法所需的运算总数至少为 $13M+32A$。

证明. 可以很容易地验证，下面的矩阵生成一个 $[13,5,5]_2$ 码。手工即可轻松找到一个使用 8 次加法来计算其转置的直线程序：

```
[1 0 0 0 0 0 0 1 0 0 1 1 1]
```

```
010000001101100100
010011100001011001
0010000110110100
```

```
o0:=i0; o1:=i1; o2:=i2; o3:=i3; o4:=i4; o5:=i4+i3; o6:=i2+i3;
o7:=i4+i0; o8:=i4+i1; o9:=o6+i1; o10:=o7+o2; o11:=o8+o10;
o12:=o11+o6.
```

少于 8 次加法至少需要有一列重复。穷举搜索表明这是不可能的。

□

表 6：特征 2 下的 4 次折叠

	rank	ADD	总计
标准模 2	25	0+0+16	25M+16A
模 2 蒙哥马利	13	9+9+19	13M+37A
标准 $\mathbb{F}_{2^5}$ (模 X^5+X^2+1)	25	0+0+24	25M+24A
$\mathbb{F}_{2^5}$ 通过模 2 蒙哥马利 (模 $X^5+X^4+X^2+X+1$)	13	9+9+18	13M+36A
$\mathbb{S}_{2^5}$ #2	13	12+9+18	13M+39A

7.3 阶 243 有限域

我们进一步研究了通过 [1] 的算法为 $\mathbb{F}_{3^5}$ 找到的秩 11 算法。我们注意到，该张量的各向同性群的阶为 $5\cdot 242^2$，其中有一个阶为 $5\cdot 242$ 的群保持这些对称算法不变。该群在从 [1] 找到的 121 个算法上具有传递性。这意味着，对于 $\mathbb{F}_{243}$ 的某个实例返回的全部 121 个对称公式，在 L 和 R 上具有相同的加法复杂度，因此我们只需针对每个不可约多项式优化一个示例。我们找到的运算总数最少为 $11+12+12+(14+11-5) = 11M+44A = 55$，而相关算法及 SLP（可用 11 次乘法在 $\mathbb{F}_{3^5}$ 中相乘）见附录 C，文件位于： PLinOpt/data/4o4o4_F243-11-44_{L,R,P}.s{ms,lp}；蒙哥马利方法对应的算法则位于： PLinOpt/data/4o4o4_F243-Montgomery-13-42_{L,R,P}.s{ms,lp}。有趣的是，采用其最佳不可约多项式（$X^5+X^4-X^3-X^2-1$）时，Montgomery方法使用了更多乘法，即 13 次而非 11 次，但 L 和 R 各少一次运算，因此运算总数相同。因此，只要（基域）乘法比（基域）加法更昂贵，我们仅使用 11 次乘法的新算法就更可取。我们所能找到的秩 12 中间算法的最佳 SLP 使用 $14+14+(12+(12-5)) = 47$ 次加法，因此并无实际优势。这些结果列于表 7。

7.4 阶 243 的半域

我们也研究了 243 阶半域的算法，同样使用[1]的算法。我们重点考察了张量秩为 10 的半域：

- 我们能找到的关于 $\mathbb{S}_{243}$ #4（表 2 中部分对称秩为 10）的最佳算法给出：$10+12+12+$（$15+10-5$）$=10M+44A$ 次运算；
- 而尝试 $\mathbb{S}_{243}$ #2 的 11 个基的全部 121 个可能公式后，最佳方案给出：$10+13+13+$（$12+10-5$）$=10M+43A$ 次运算。

相关内容见表 7，后一个算法可在 PLinOpt/data/4o4o4_S243_10-43__{L,R,P}.s{ms,lp}中找到，另见附录 E。

表 7：含 243 个元素的扩张中的乘法

	rank	ADD	总计
$\mathbb{F}_{3^5}$ 标准模 X^5-X+1	25	0+0+24	25M+24A
$\mathbb{F}_{3^5}$ 蒙哥马利（模 $X^5+X^4-X^3-X^2-1$）	13	11+11+20	13M+42A
$\mathbb{F}_{3^5}$ 此处（模 $X^5-X^4+X^3+X^2+X+1$）	12	14+14+19	12M+47A
$\mathbb{F}_{3^5}$ 此处（模 X^5-X+1）	11	12+12+20	11M+44A
$\mathbb{S}_{3^5}$ 此处	10	13+13+17	10M+43A

8 结论与开放问题

本文进一步推进了域与半域的张量秩研究，并开启了半域加法复杂度的研究。我们在许多情形下改进了现有最优结果，并比较了不同代数上新的已知最佳复杂度。作为其中一部分，我们研究了具有特定参数的线性码的最短直线程序。仍有若干自然的开放问题有待解决。

- 给出SSLP$_q(r,k,q)$的一般下界，并为更多小参数建立精确结果。
- 确定是否存在一个针对阶为3_4 的域或半域、且仅需$9M+20A$ 次运算的算法。
- 确定对于阶为 3_5 的半域，是否存在一种算法，需要 $10M$ 且少于 $43A$ 次运算，或 $11M$ 且少于 $44A$ 次运算。

参考文献

[1] Razvan Barbulescu，J´er´emie Detrey、Nicolas Estibals 和 Paul Zimmermann。寻找双线性映射的最优公式。¨载于 Ferruh Ozbudak 和 Francisco Rodr´ıguez-Henr´ıquez 编，《有限域算术》第四届国际研讨会 WAIFI 2012，德国波鸿，2012年7月16日至19日。论文集，第 7369 卷，《计算机科学讲义》，第 168–186 页。施普林格，2012 年。doi:10.1007/978-3-642-31662-3_12。

[2] Daniel J. Bernstein、Niels Duif、Tanja Lange、Peter Schwabe 和 Bo-Yin Yang。高速高安全性数字签名。《密码工程杂志》，2(2):77–89，2012年9月。网址：http://link.springer.com/10.1007/s13389-012-0027-1，doi:10.1007/s13389-012-0027-1。

[3] Marco Bodrato 和 Alberto Zanoni。整数与多项式乘法：迈向最优 Toom-Cook 矩阵。载于2007年国际符号与代数计算研讨会论文集，ISSAC '07，第17–24页，美国纽约州纽约，2007年。计算机协会。doi:10.1145/1277548.1277552。

[4]Roger W. Brockett 和 David Dobkin。《双线性型集合的最优求值》。《线性代数及其应用》，19(3):207–235，1978年。URL：https://www.sciencedirect.com/science/article/pii/0024379578900125，doi:10.1016/0024-3795(78)90012-5。.[5] Murat Cenk 和 Ferruh Ozbudak。《有限域中的乘法》。《复杂性杂志》，26(2):172–186，2010年。doi:10.1016/j.jco.2009.11.002。[6]Shaoshi Chen 和 Manuel Kauers。《多项式乘法的翻转图》，2025年2月。arXiv:2502.06264 [cs]。URL：http://arxiv.org/abs/2502.06264，doi:10.48550/arXiv.2502.06264。[7] D. V. Chudnovsky 和 G. V. Chudnovsky。《有限域上的代数复杂度与代数曲线》。《复杂性杂志》，4(4):285–316，1988年。doi:10.1016/0885-064X(88)90012-X。[8]El´ıas F. Combarro、I. F. R´ua 和 J. Ranilla。《半域计算探索的新进展》。《国际计算机数学杂志》，88(9):1990–2000，2011年。doi:10.1080/00207160.2010.548518。[9] Svyatoslav Covanov。《有限域中双线性映射最优公式的改进求解方法》。《理论计算机科学》，790:41–65，2019。doi:10.1016/j.tcs.2019.04.029。[10]Ulrich Dempwolff。《81阶半域平面》。《几何学杂志》，89(1-2):1–16，2008年。doi:10.1007/s00022-008-1995-2。[11] Jean-Guillaume Dumas、Bruno Grenet、Cl´ement Pernet 和 Alexandre Sedoglavic。PLinOpt：用于处理线性与双线性程序的 C++ 程序集合，2026年1月。v4.1，9.8 kSLOC。URL：https://github.com/jgdumas/plinopt。[12] Jean-Guillaume Dumas、Cl´ement Pernet 和 Alexandre Sedoglavic。《Strassen 算法并非精度最优》。载于 Shaoshi Chen 编，《ISSAC'2024：2024年国际符号与代数计算研讨会论文集》，美国北卡罗来纳州罗利，第254–263页，纽约，2024年7月。ACM 出版社。URL：https://hal.science/hal-04441653，doi:10.1145/3666000.3669697。[13] Jean-Guillaume Dumas、Cl´ement Pernet 和 Alexandre Sedoglavic。《迈向递归矩阵乘法快速且精确算法的自动生成》。《符号计算杂志》，134:102524，2026年。doi:10.1016/j.jsc.2025.102524。

[14]Nadia El Mrabet、Aurore Guillevic 和 Sorina Ionica。《5次有限域扩张中的高效乘法》。载于 Abderrahmane Nitaj 和 David Pointcheval 编，密码学进展——AFRICACRYPT 2011，第188–205页，柏林、海德堡，2011年。Springer Berlin Heidelberg。doi:10.1007/978-3-642-21969-6_12。

[15] Markus Grassl。线性码与量子码最小距离的界。在线可访问于http://www.codetables.de，2007。访问日期：2026-01-29。[16] Anatolii Karatsuba与Y. Ofman。自动机上的多位数乘法。载于苏联物理学报告，第7卷，第595–596页，1963。[17] E. Karstadt与O. Schwartz。矩阵乘法，略快一些。载于C. Scheideler与M. Hajiaghayi编，第29届 ACM 算法与体系结构并行性研讨会论文集，SPAA'17，第101–110页，纽约， NY， USA，2017年7月24–26日。 ACM。doi:10.1145/3087556.3087579。[18]Emilia K¨ asper。OpenSSL中的快速椭圆曲线密码学。载于David Hutchison、Takeo Kanade、Josef Kittler、Jon M. Kleinberg、Friedemann Mattern、John C. Mitchell、Moni Naor、Oscar Nierstrasz、C. Pandu Rangan、Bernhard Steffen、Madhu Sudan、Demetri Terzopoulos、Doug Tygar、Moshe Y. Vardi、Gerhard Weikum、George Danezis、Sven Dietrich与Kazue Sako编，金融密码学与数据安全，第7126卷，第27–39页。施普林格，柏林海德堡，柏林，海德堡，2012。丛书名：《计算机科学讲义》。URL: http://link.springer.com/10.1007/978-3-642-29889-9_4，doi:10.1007/978-3-642-29889-9_4。[19] Paul C. Kocher。Diffie-Hellman、RSA、DSS及其他系统实现上的时序攻击。载于Gerhard Goos、Juris Hartmanis、Jan Van Leeuwen与Neal Koblitz编，密码学进展——CRYPTO '96，第1109卷，第104–113页。施普林格，柏林海德堡，柏林，海德堡，1996。丛书名：《计算机科学讲义》。URL: http://link.springer.com/10.1007/3-540-68697-5_9，doi:10.1007/3-540-68697-5_9。[20] Michel Lavrauw。有限半体与非奇异张量。设计、码与密码学，68(1-3):205–227，2013。doi:10.1007/s10623-012-9710-6。[21] Michel Lavrauw、Andrea Pavan与Corrado Zanella。关于 $3\times3\times3$张量的秩。线性与多重线性代数，61(5):646–652，2013。doi:10.1080/03081087.2012.701299。[22] Michel Lavrauw与Olga Polverino。有限半体。伽罗瓦几何当前研究主题，第131–160页，2011年。[23] Michel Lavrauw与John Sheekey。16阶与81阶半域的张量秩。《线性代数及其应用》，643:99–124，2022。doi:10.1016/j.laa.2022.02.015。[24] Robert A. Liebler。关于非奇异张量及相关射影平面。专门几何，11(4):455–464，1981。doi:10.1007/BF00181205。[25] John P. May、David Saunders与Zhendong Wan。高效矩阵秩计算及其在强正则图研究中的应用。载于2007年国际符号与代数计算研讨会论文集，ISSAC '07，第277–284页，纽约，NY，美国，2007。计算机协会。doi:10.1145/1277548.1277586。[26] Peter L. Montgomery。五、六、七项Karatsuba类公式。《IEEE计算机汇刊》，54(3):362–369，2005。doi:10.1109/TC.2005.49。[27] Roy Nissim、Oded Schwartz与Yuval Spiizer。除法障碍：Toom-Cook插值中的最优界与结构极限。载于Kasper Green Larsen与Barna Saha编，2026年ACM-SIAM离散算法研讨会（SODA）论文集，第5241–5254页，费城，PA，2026年。工业与应用数学学会。doi:10.1137/1.9781611978971.190。

[28]Vera Pless。论戈莱码的唯一性。《组合理论杂志》，5:215–228，1968。doi:10.1016/S0021-9800(68)80067-5。

[29] I. F. R´ua, El´ıas F. Combarro 和 J. Ranilla。64 阶半域的分类。《代数杂志》，322(11):4011–4029，2009。doi:10.1016/j.jalgebra.2009.02.020。

[30]I.F. R´ua, El´ías F. Combarro。243 元可除代数的确定。《有限域及其应用》，18(6):1148–1155，2012。网址：https://www.sciencedirect.com/science/article/pii/S1071579712000615，doi:10.1016/j.ffa.2012.07.001。

[31] Ignacio F. R´ua and El´ías F. Combarro。密码不确定度：关于基于有限´半域的代换盒的一些实验。载于 Eduardo Gil、Eva Gil、Juan Gil 和 Mar´ía Angeles Gil（编），《不确定性的数学：献给 Pedro Gil 的致敬文集》，第 485–492 页，Cham，2018。施普林格国际出版。doi:10.1007/978-3-319-73848-2_45。

[32] John Sheekey. MRD码：构造与联系。载于 组合学与有限域——差集、多项式、伪随机性及其应用，第23卷，拉东计算与应用数学丛书，第255–285页。德古意特，柏林， [2019] ©2019。 doi:10.1515/9783110642094-013。

[33] Andrei L Toom. 实现整数乘法的功能元件方案的复杂度，发表于《苏联数学》（《苏联科学院通报》译文），第4卷，1963年。

[34] M. van Eupen. 三元线性码的四个不存在性结果。《IEEE信息论汇刊》，41(3):800–805，1995年。doi:10.1109/18.382031。

维数和距离为4的SSLP

我们在此给出维数和距离为4的其他线性码的SSLP。

引理 22。SSLP$_q$(9, 4, 4) $= 5$，对于 $q \in \{2, 3\}$。

证明。我们考虑一个 $[9, 4, 4]$ 码的生成矩阵 G 。若它有一个零列，则由定理 18，它至少需要 6 次运算。因此，若要少于 5 次运算，它必须至少有 5 个不需要运算的列。不失一般性，我们可以假设其中四个是标准基向量，其余为重复列。首先，假设有 2 个重复列。则不失一般性且至多相差符号，$G = [I_4 \mid E^1 \mid E^1 \mid M]$或 $G = [I_4 \mid I_{4|2} \mid M]$，其中 $I_{4|2}$是其列为典范基前两个向量的 4×2矩阵。为使重量为 4， M 的最后两行必须全为加一或减一，并且这 3 个中至少有两个在这两行之间是共享的。于是 G 最后两行之差的重量不大于 3。现在， $G = [I_4 \mid E^1 \mid \vec{v} \mid M]$。其次，若 4 个互异的列，即 $\vec{v}$以及 M的那些列，都有 3 个非零元，则它们不能用少于 5 次运算算出（任一列需要 2 次，然后其余 3 列至少各需 1 次运算）。第三，若 $\vec{v} = E^1 + E^2$，则同样 M 的最后两行必须全为 1，且 G 最后两行之差的重量不超过 3。现在不失一般性，$G = [I_4 \mid E^1 \mid E^2 + E^3 \mid M]$，并且 M的最后一行全为非零元，以使 G 的最后一行重量为 4。 M 的其余各行必须至少还有 2 个其他非零元，并且必须在其中互不相同，否则它们之差的重量过小。因此，不失一般性， G 如下：

$$\begin{bmatrix} 1&0&0&0&\star&0&\square&\star&\star \\ 0&1&0&0&0&\star&\star&\square&\star \\ 0&0&1&0&0&\star&\star&\star&\square \\ 0&0&0&1&0&0&\star&\star&\star \end{bmatrix}$$

现在，运算次数最少的情况是星号为 1、方块为 0。在这种情况下，它确实表示一个 $[9, 4, 4]$ 码，定义在 $\mathbb{F}_2$ 或 $\mathbb{F}_3$ 上，而其转置已经需要 5 次运算：

$$\begin{bmatrix} 1&0&0&0&1&0&0&1&1 \\ 0&1&0&0&0&1&0&0&1 \\ 0&0&1&0&0&1&1&0&1 \\ 0&0&0&1&1&0&1&1&1 \end{bmatrix}$$

```
01000110100100
1110000100111
```

```
o0:=i0; o1:=i1; o2:=i2; o3:=i3; o4:=i0; t4:=i0+i3; o5:=i1+i2;
o6:=i3+o5; o7:=i2+t4; o8:=i1+t4;
```

□

引理 23. SSLP(10, 4, 4) $= 4$，对于 $q \in \{2, 3\}$。

证明。 如果 $[10, 4, 4]$ 码的生成矩阵含有一个零列，那么根据引理 22，至少需要 5 次运算。要少于 4 次运算，它因此必须至少有四个标准基列和两个重复列。一方面，假设有 3 个重复列。不失一般性，令 $G = [I_4 \mid E^i \mid E^j \mid E^k \mid M]$，相差至多一个符号。若 i、j、k 中有两个相等，则 M至少有两行全为非零元素，而 G 中相同行的差的重量不大于 3。因此，不失一般性，$G = [I_4 \mid I_{4|2} \mid M]$ 且 M 的最后一行全为非零元素。于是 M 的前 3 行互不相同，且重量至少为 2；不失一般性， G 如下：

$$\begin{bmatrix} 1&0&0&0&\star&0&0&\square&\star&\star \\ 0&1&0&0&0&\star&0&\star&\square&\star \\ 0&0&1&0&0&0&\star&\star&\star&\square \\ 0&0&0&1&0&0&0&\star&\star&\star \end{bmatrix}$$

同样，运算次数最少的情况是星号为 1、方块为 0。在这种情况下，其转置已经至少需要 5 次运算：

```
0100010101001000
0111000010001111
```

```
o0:=i0; o1:=i1; o2:=i2; o3:=i3; o4:=i0; o5:=i1; o6:=i2;
t4:=i3+i0; o7:=i1+i2+i3; o8:=t4+i2; o9:=t4+i1;
```

On

另一方面，若只有两行重复行，则至少需要 4 次运算。现在，下面的矩阵也表示一个 $[10, 4, 4]$ 码，定义在 $\mathbb{F}_2$ 或 $\mathbb{F}_3$上。其转置只需 4 次运算：

```
0100001110001000
1110000011110000
```

```
o0:=i0; o1:=i1; o2:=i2; o3:=i3; o4:=i3; o5:=i3+i2; o6:=o5+i1;
o8:=i1+i0; o7:=o8+i2; o9:=i0;
```

□

B 秩 13 次 4 多项式乘法

这里给出在任意环上进行一般 4 次多项式乘法的算法：PLinOpt/data/4o4o8_蒙哥马利该算法计算 $c_0 + c_1X + c_2X^2 + \ldots + c_8X^8 = (a_0 + a_1X + \ldots + a_4X^4)(b_0 + b_1X + \ldots + b_4X^4$，需要 13 次乘法、53 次加减法，以及 5 次乘以 2 或 3 的数乘。

列表 g 1: 左侧 & 右侧 g侧 SLP, 13 p乘积 ；发布 SLP

```
l7:=a0+a1; l8:=a0-a4; l6:=a4+a3; l3:=l7-l6; x7:=a2+l7; l0:=l6+x7; x9:=l6+a2; l1:=a0-x9; l2:=x7-a4;
l4:=l1+a4; l5:=l2-a0; l9:=a4; l10:=a3; l11:=a1; l12:=a0;

r7:=b0+b1; r8:=b0-b4; r6:=b4+b3; r3:=r7-r6; y7:=b2+r7; r0:=r6+y7; y9:=r6+b2; r1:=b0-y9; r2:=y7-b4;
r4:=r1+b4; r5:=r2-b0; r9:=b4; r10:=b3; r11:=b1; r12:=b0;

p0:=l0*r0; p1:=l1*r1; p2:=l2*r2; p3:=l3*r3; p4:=l4*r4; p5:=l5*r5; p6:=l6*r6; p7:=l7*r7; p8:=l8*r8;
p9:=l9*r9; p10:=l10*r10; p11:=l11*r11; p12:=l12*r12;

z15:=p1+p10-p8; z16:=p2+p11-p8; k2:=p6-p10; k3:=p7-p11; k4:=z15-p4; k5:=z16-p5; k9:=p12+p9;
k10:=k2+z15-p3; k11:=p3+p0-k3-z16; c7:=k2-p9; c1:=k3-p12; c6:=k4+k9-k2; c2:=k5+k9-k3;
c5:=k11-(k4+p9)*2-p12*3; c4:=k4+k5+k9*3+k10-k11; c3:=p0-k10-(k5+p12)*2-p9*3; c0:=p12; c8:=p9;
```

C 在 $\mathbb{F}_{3^5}$ 中的秩 11 乘法

这里给出目前已知的最佳算法，只需 11 次乘法即可计算 $\mathbb{F}_{3^5}$ 中的元素，其中多项式分别模 3 和 $1 - X + X^5$ 进行计算：PLinOpt/data/4o4o4_F243-11-44_{L,R,P}.s{ms,lp}。随后给出相应的 SLP。它计算 $c_0 + c_1X + \ldots + c_4X^4 \equiv (a_0 + a_1X + \ldots + a_4X^4)(b_0 + b_1X + \ldots + b_4X^4) \bmod 1 - X + X^5 mod 3$，需要 11 次乘法和 20 次加法。

列表 2：左与右 SLP；11 个乘积；发布 SLP

```
l7:=a3-a4; x6:=a1+a0; l1:=x6-a2; l5:=a1-a4; l0:=a0+a2; l2:=a0-a3; l3:=l1-a3; l6:=a2-a3+l5; l8:=a1-l7;
l9:=x6-l7; l10:=a2+l7; l4:=a4;

r7:=b3-b4; y6:=b1+b0; r1:=y6-b2; r5:=b1-b4; r0:=b0+b2; r2:=b0-b3; r3:=r1-b3; r6:=b2-b3+r5; r8:=b1-r7;
r9:=y6-r7; r10:=b2+r7; r4:=b4;
```

```
p0:=l0*r0; p1:=l1*r1; p2:=l2*r2; p3:=l3*r3; p4:=l4*r4; p5:=l5*r5; p6:=l6*r6; p7:=l7*r7; p8:=l8*r8;
p9:=l9*r9; p10:=l10*r10;

n1:=p7+p6; k16:=p3-n1; q5:=p5+k16; q2:=p2+k16; z7:=p1-q5; n5:=p8+q2+p1; c3:=z7-p10-p4; c1:=q2+p0-z7;
n7:=p0-p10-n5; c0:=p4-p7-p3-q5+n5; c4:=p9-n7; c2:=n7+n1-q5;
```

D $\mathbb{S}_{3^4}$ 中的最优秩-8乘法 #1

这是以下对象所能达到的最佳算法： 一个 81 阶预半域，见 PLinOpt/data/3o3o3_S81_8_22_{L,R,P}.s{ms ,

Listing 3: Left & Right SLP; 8 products; Post SLP

```
x4:=a1+a3; x5:=a0+a2; l4:=a2+x4;  y4:=b1+b2; y5:=b0+b3; r1:=b0+y4;
l5:=a3+x5; l6:=a0+x4; l7:=a1+x5;  r2:=b2+y5; r3:=b3+y4; r4:=b1+y5;
l0:=a0; l1:=a1; l2:=a2; l3:=a3;   r0:=b3; r5:=b0; r6:=b1; r7:=b2;

p0:=l0*r0; p1:=l1*r1; p2:=l2*r2; p3:=l3*r3; p4:=l4*r4; p5:=l5*r5; p6:=l6*r6; p7:=l7*r7;

n1:=p4+p3; z4:=p2+p1; c3:=p1+p0+n1; c0:=p5-p2-n1; c2:=p7+p3+z4; c1:=p6-p4+z4;
```

以下是将张量 $T(L,R,P)$ 变换为半域 $\mathbb{S}_{3^4}$ #1 的张量所需的基变换矩阵来自 [10]。

$$X=\begin{pmatrix}2&2&1&1\\0&1&1&2\\0&2&1&2\\2&2&2&1\end{pmatrix},\quad Y=\begin{pmatrix}2&1&1&0\\1&0&1&1\\0&1&1&1\\2&2&0&2\end{pmatrix},\quad Z=\begin{pmatrix}2&0&1&2\\1&2&2&0\\1&1&0&1\\0&2&1&0\end{pmatrix}$$

E $\mathbb{S}_{3^5}$ 中的秩-10乘法 #2

以下是我们为阶 243 的半域找到的最佳算法，详见 PLinOpt/data/4o4o4_S243_10-43__{L,R,P}.s{m

Listing 4: Left & Right SLP; 10 products; Post SLP

```
l2:=a1+a4; x8:=a3+a4; x9:=a3-a4; l3:=a0+l2; l0:=a0-a1; l4:=a2+l3; l1:=l4-a4; l5:=a3+l2; l6:=a2-x8;
l7:=l1-x8; l8:=a0+x9; l9:=a2+a1+x9;

r2:=b1+b4; y8:=b3+b4; y9:=b3-b4; r3:=b0+r2; r0:=b0-b1; r4:=b2+r3; r1:=r4-b4; r5:=b3+r2; r6:=b2-y8;
r7:=r1-y8; r8:=b0+y9; r9:=b2+b1+y9;

p0:=l0*r0; p1:=l1*r1; p2:=l2*r2; p3:=l3*r3; p4:=l4*r4; p5:=l5*r5; p6:=l6*r6; p7:=l7*r7; p8:=l8*r8;
p9:=l9*r9;

z5:=p9-p4; z8:=p5+p2; z7:=p5-p2; z6:=p4-p1; c0:=p8+p7-p4+z7; c4:=z7-p6+z6; c1:=p6-p3+p0+z6;
c2:=z8-p3-z5; c3:=z8-p7-p1+z5;
```