On k-way split multiplication algorithms

Mehmet Özgün Cihangir*and Oğuz Yayla

Institute of Applied Mathematics
Middle East Technical University, Ankara, Türkiye
{ozgun.cihangir,oguz}@metu.edu.tr

August 6, 2026

Abstract

Efficient polynomial multiplication and matrix-vector operations are fundamental to computational algebra and modern cryptography. In lattice-based post-quantum cryptography (PQC), schemes utilizing Number Theoretic Transform (NTT)-unfriendly rings require highly optimized subquadratic multiplication algorithms. In this paper, we establish a rigorous mathematical framework for generalized k-way split polynomial multiplication and Toeplitz Matrix-Vector Product (TMVP) algorithms over arbitrary fields. First, we construct generalized k-way Schoolbook and Karatsuba multiplication algorithms, deriving exact closed-form recurrence relations and arithmetic complexities for any integer k. Second, we introduce a novel k-way TMVP algorithm utilizing optimal evaluation points and matrix row-reversal techniques. We mathematically prove that this generalized formulation strictly achieves the theoretical interpolation lower bound, requiring exactly $2k-1$ subproblems and yielding a subquadratic asymptotic complexity of $O(n^{\log_k(2k-1)})$. Furthermore, we determine the optimal consecutive application sequence of k-way Karatsuba and Schoolbook algorithms for any input size n, proving that the peak efficiency is driven entirely by the prime factorization of n. Finally, we establish exact algebraic crossover thresholds, demonstrating that our generalized TMVP formulas and optimal algorithmic sequences significantly outperform state-of-the-art unequal k-way splits and classical combinations in the literature, providing minimum arithmetic operation counts for $k \in \{5, 6, 8, 12\}$ and inputs of power-of-two and power-of-three dimensions.

Keywords: Polynomial multiplication, Toeplitz matrix-vector product, Algebraic complexity, Subquadratic algorithms, Post-quantum cryptography, Divide and conquer

*Corresponding author and he is partially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK) 2210-E graduate scholarship program.

1 Introduction

The arithmetic complexity of polynomial multiplication and matrix-vector operations constitutes a foundational pillar in computer algebra, applied mathematics, and algorithmic design [22]. Historically, the naive schoolbook method of multiplying two n-term polynomials requires $O(n^2)$ arithmetic operations. The quest to overcome this quadratic barrier began with the seminal work of Karatsuba and Ofman [15], which reduced the complexity to $O(n^{\log_2 3})$ by recursively splitting the multiplicand polynomials into two equal parts. This divide-and-conquer principle was subsequently generalized by Toom and Cook [20, 8], proving that polynomials split into k parts can be multiplied using exactly $2k-1$ sub-multiplications, leading to subquadratic time complexities. However, the efficiency of these algorithms in practice heavily depends on the choice of interpolation points and the exact sequence of evaluation and interpolation operations, as the hidden constants in the asymptotic notation can become significant bottlenecks [4].

A significant portion of our work is dedicated to mathematically formalizing the generalized k-way split polynomial multiplication. While specific split algorithms for small integers (such as 2-way, 3-way, or 4-way splits) have been extensively studied and optimized in the literature [14], providing a closed-form recurrence and an explicit algorithmic construction for an arbitrary $k \in \mathbb{N}$ remains a distinct mathematical challenge. In this paper, we establish the exact algebraic construction for generalized k-way Schoolbook and Karatsuba polynomial multiplication algorithms. By strictly defining the recursive arithmetic operations for any given k, we provide a rigorous framework that not only establishes the precise number of additions and multiplications required but also mathematically solves the problem of algorithmic sequencing. Specifically, we prove the optimal consecutive application sequence of these k-way splits for an input size n based entirely on its prime factorization.

Alongside standard polynomial multiplication, a highly elegant and algebraically robust alternative is the Toeplitz Matrix-Vector Product (TMVP). A Toeplitz matrix, characterized by constant descending diagonals, can represent one of the multiplicand polynomials, while the other polynomial is represented as a vector [10]. This representation has drawn significant attention because TMVP-based algorithms inherently integrate the polynomial modular reduction step into the multiplication process. Furthermore, TMVP operations can be performed division-free, which avoids the modular inverse calculations that traditionally plague Toom-Cook methods over certain finite fields and rings [11]. By employing divide-and-conquer strategies, an n-dimensional TMVP can be split into multiple smaller subproblems, yielding highly efficient subquadratic space and time complexities [7, 12].

Despite the extensive literature on subquadratic TMVP multipliers, designing a generalized k-way split TMVP algorithm that strictly achieves the theoretical minimum of $2k-1$ subproblems without relying on complex field extensions or ad-hoc heuristic evaluation points is a formidable task. To reach higher dimensions, researchers have often resorted to unequal matrix splittings, such as 2×3 or 4×3 approaches [16], or have restricted their focus exclusively to binary ex-

tension fields [10]. In this paper, we mathematically prove that for *any* integer k, an optimal k-way TMVP formula utilizing exactly $2k-1$ subproblems can be constructed for general fields. By utilizing optimal evaluation points and matrix row-reversal techniques, our generalized formulation strictly reaches the theoretical interpolation lower bound for k-split structures.

While the theoretical optimization of these algorithms is mathematically significant in its own right, these advancements also offer immediate practical benefits to the broader landscape of modern cryptography. With the impending arrival of large-scale quantum computers, the National Institute of Standards and Technology (NIST) has spearheaded the standardization of Post-Quantum Cryptography (PQC) [2, 19]. Lattice-based cryptography has emerged as the cornerstone of this transition, leading to the selection of CRYSTALS-Kyber for key encapsulation, along with CRYSTALS-Dilithium and Falcon for digital signatures [6]. The operational efficiency of these selected schemes heavily relies on prime moduli and specific parameters that perfectly accommodate the Number Theoretic Transform (NTT). Conversely, other prominent third-round finalists such as Saber [9] and NTRU [13] utilize NTT-unfriendly quotient rings, such as those with power-of-two moduli. Although these specific schemes were not ultimately selected for the initial NIST standardization, the algebraic structures they employ remain a vibrant area of academic research. They are highly relevant for alternative cryptographic designs and future algorithms that may prioritize different security assumptions, side-channel resistance, or specific hardware constraints over NTT compatibility. In such NTT-unfriendly and parameter-restricted environments, polynomial multiplication remains the primary computational bottleneck [17, 18]. Our theoretically optimal k-way TMVP algorithms, alongside our newly proven optimal sequences of Karatsuba and Schoolbook applications, provide a mathematically rigorous and generalized solution to minimize arithmetic operation counts across this diverse spectrum of cryptographic systems.

Furthermore, generalized TMVP formulations are highly applicable to code-based cryptography. A compelling target is the Hamming Quasi-Cyclic (HQC) algorithm, which was recently selected by the National Institute of Standards and Technology (NIST) to be standardized as a Key-Encapsulation Mechanism (KEM) [3]. The operational efficiency of HQC relies heavily on dense polynomial multiplications in the quotient ring $\mathcal{R} = \mathbb{F}_2[x]/\langle x^n - 1\rangle$ for specific prime dimensions n [1]. Because circulant multiplication in such quasi-cyclic codes is intrinsically equivalent to a Toeplitz matrix-vector product, integrating k-way TMVP splits which require exactly $2k-1$ subproblems can eliminate the modular reduction overhead entirely. Consequently, applying our theoretically optimal subquadratic algorithms provides a mathematically rigorous pathway to minimize both space and time complexities for future hardware and software implementations of this new post-quantum standard.

1.1 Our Contributions

In this work, we present generalized polynomial multiplication and TMVP formulas that reach theoretical lower bounds, and we strictly prove their optimal deployment sequences. Specifically, our main contributions are:

- **Generalized k-way Split Polynomial Multiplication:** In Section 3, we advance the theoretical framework of polynomial arithmetic by constructing generalized k-way Schoolbook and Karatsuba multiplication algorithms. We derive their exact closed-form recurrence relations for any arbitrary $k \in \mathbb{N}$, formally establishing their arithmetic complexities as $O(n^2)$ and $O(n^{\log_k(k(k+1)/2)})$, respectively.

- **Optimal k-way TMVP with exactly $2k-1$ Subproblems:** In Section 4, we construct a novel k-way split TMVP algorithm utilizing optimal evaluation points and matrix row-reversal techniques. We mathematically prove that this generalized formulation requires exactly $2k-1$ subproblems, strictly meeting the theoretical lower bound for interpolation-based splits and yielding a subquadratic asymptotic complexity of $O(n^{\log_k(2k-1)})$.

- **Efficiency Sequence based on Prime Factorization:** In Section 5, we theoretically establish the optimal consecutive application sequence of Schoolbook and Karatsuba algorithms for any fixed input size n. We prove that an algorithmic sequence achieves peak efficiency when Karatsuba is recursively applied corresponding to the prime factors of n, dictating precise transitional rules based on factor parity.

- **Crossover Thresholds and Literature Comparison:** In Section 6, we provide precise algebraic calculations to identify the crossover points where our proposed algorithms outperform existing methods. Our TMVP formulas require fewer arithmetic operations than the state-of-the-art unequal splits (e.g., [11], [12], [16]) for dimensions $k = 5, 6, 8, 12$ beyond specifically calculated input sizes. Furthermore, our optimal algorithmic sequence demonstrates superior efficiency compared to classical Karatsuba and Toom-Cook implementations [14] for power-of-two and power-of-three input dimensions.

1.2 Outline

The remainder of this paper is organized as follows. Section 2 introduces the preliminary definitions of Schoolbook and Karatsuba polynomials, Toeplitz matrices, and basic 2-way and 3-way TMVP concepts. Section 3 establishes the mathematical construction and complexity analysis of the generalized k-way split Schoolbook and Karatsuba algorithms. Section 4 presents our novel k-way TMVP algorithm that requires exactly $2k-1$ subproblems. Section 5 details the efficiency order relations and mathematically proves the optimal sequence for consecutive algorithm applications. Section 6 compares our algorithmic complexities and exact crossover

thresholds with the state-of-the-art literature. Finally, Section 7 concludes the paper and outlines future research directions.

2 Basic Definitions

Throughout this section, some preliminary definitions about multiplication algorithms and special types of matrices will be given. In Sections 2.1 and 2.2, we give the definition of the schoolbook polynomial multiplication method and the definitions of the famous 2-way Schoolbook and Karatsuba polynomial multiplication algorithms. In Section 2.3, we give the definitions of Toeplitz, Vandermonde and semi-Vandermonde matrices. Finally, in Section 2.4, the definitions of the 2-way and 3-way Toeplitz matrix-vector product formulas are given.

2.1 Schoolbook Multiplication and Schoolbook 2-way Polynomial Multiplication Algorithm

We first define the schoolbook formula for multiplying two polynomials.

Definition 2.1 (Schoolbook Polynomial Multiplication)**.** Let $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ and $b(x) = b_0 + b_1x + \cdots + b_{n-1}x^{n-1}$ be two polynomials over a field $\mathbb{F}$. The *schoolbook polynomial multiplication* $c(x)$ of $a(x)$ and $b(x)$ is defined as

$$c(x) = a(x)b(x) = c_0 + c_1x + \cdots + c_{2n-2}x^{2n-2},$$

where the coefficients $c_0, c_1, \ldots, c_{2n-2}$ are obtained by computing

$$\begin{aligned}
c_0 &= a_0b_0 \\
c_1 &= a_0b_1 + a_1b_0 \\
c_2 &= a_0b_2 + a_1b_1 + a_2b_0 \\
&\ldots \\
c_{n-1} &= a_0b_{n-1} + a_1b_{n-2} + \cdots + a_{n-2}b_1 + a_{n-1}b_0 \\
c_n &= a_1b_{n-1} + a_2b_{n-2} + \cdots + a_{n-2}b_2 + a_{n-1}b_1 \\
&\ldots \\
c_{2n-3} &= a_{n-2}b_{n-1} + a_{n-1}b_{n-2} \\
c_{2n-2} &= a_{2n-2}b_{2n-2}.
\end{aligned}$$

We now give the definition of the 2-way Schoolbook polynomial multiplication algorithm in [14] for computing the product of two n-term polynomials.

Definition 2.2. [2-way Schoolbook Polynomial Multiplication Algorithm] Let $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ and $b(x) = b_0 + b_1x + \cdots + b_{n-1}x^{n-1}$ be two polynomials over a field $\mathbb{F}$. Let us split each of $a(x)$ and $b(x)$ into two equal parts in the following way:

$$a(x) = A_0(x) + A_1(x)x^{n/2}$$

and

$$b(x) = B_0(x) + B_1(x)x^{n/2},$$

where $A_0(x)$, $A_1(x)$, $B_0(x)$, $B_1(x)$ are $n/2$-term polynomials. One can compute the product $a(x)b(x)$ as follows:

$$a(x)b(x) = C_0(x) + C_1(x)x^{n/2} + C_2(x)x^n, \tag{1}$$

where

$$\begin{aligned} C_0(x) &= A_0(x)B_0(x) \\ C_1(x) &= A_0(x)B_1(x) + A_1(x)B_0(x) \\ C_2(x) &= A_1(x)B_1(x). \end{aligned} \tag{2}$$

The algorithm for computing the product $c(x) = a(x)b(x)$ by plugging the results of (2) to Equation (1) is called *2-way Schoolbook polynomial multiplication algorithm.*

In Definition 2.2, to compute C_0, C_1 and C_2 after finding A_0, A_1, B_0, B_1, we recursively apply 2-way Schoolbook polynomial multiplication algorithm for computing the $n/2$-term polynomial multiplications $A_0(x)B_0(x)$, $A_0(x)B_1(x)$, $A_1(x)B_0(x)$ and $A_1(x)B_1(x)$ in the above equations for $C_0(x)$, $C_1(x)$ and $C_2(x)$, until we eventually end up with multiplicand polynomials of degree one and obtain $C_0(x)$, $C_1(x)$ and $C_2(x)$ by combining all these results appropriately. To multiply two polynomials using 2-way Schoolbook polynomial multiplication algorithm, the following addition and multiplication operations will be executed:

- *Multiplications:* There are four $n/2$-term polynomial multiplications (i.e., A_0B_0, A_0B_1, A_1B_0, A_1B_1). So, it takes $4M(n/2)$ operations as for multiplication (we use the same algorithm recursively until we reach a single-term polynomial).

- *Additions:* The addition of the $n-1$-term polynomials A_0B_1 and A_1B_0 takes $n-1$ double-word additions. Also, in the construction step of $a(x)b(x)$, we need to add C_0 and $C_1x^{n/2}$, which have $n/2-1$ overlapping terms, and so costs $n/2-1$ double-word additions. A similar reasoning for $C_1x^{n/2}$ and C_2x^n gives $n/2-1$ more overlapping terms, which also costs $n/2-1$ double-word additions. Note that the overlapping terms of the first addition consist of terms of degree $n/2, n/2+1, \ldots, n-2$, whereas the second addition has overlapping terms of degree $n, n+1, \ldots, 3n/2-2$, which means that these three polynomials do not have a common overlapping term. So the total additions are $(n-1)+(n/2-1)+(n/2-1) = 2n-3$ double-word additions.

In conclusion, we have the following recursive formula:

$$M(n) = 4M(n/2) + (2n-3)A_d, \tag{3}$$

where A_d denotes the cost of double-word addition. Setting $M(1) = M$, where M is the cost of one multiplication, taking $n = 2^l$ and solving (3) gives

$$M(2^l) = 4^lM + (4^l - 2 \cdot 2^l + 1)A_d$$

or

$$M(n) = n^2M + (n^2 - 2n + 1)A_d.$$

If we work on an implementation platform assuming equal unit costs of multiplication and double-word addition, we take $M = A_d = 1$ in (3). In this case, taking $n = 2^l$ and solving (3) gives

$$M(2^l) = 2^{2l+1} - 2^{l+1} + 1$$

or

$$M(n) = 2n^2 - 2n + 1.$$

for all n, i.e., 2-way Schoolbook polynomial multiplication algorithm has a time complexity of $O(n^2)$. The pseudocode of 2-way Schoolbook Polynomial Multiplication Algorithm is given in Algorithm 1:

Algorithm 1 2-way Schoolbook Polynomial Multiplication Algorithm (TwoWaySBPM)

Input: Two n-term polynomials $a(x)$ and $b(x)$
Output: $c(x) = a(x)b(x)$
$A_0 \leftarrow$ first n/2 terms of $a(x)$ (terms having degree from 0 to $n/2-1$, inclusive)
$A_1 \leftarrow$ last n/2 terms of $a(x)$ (terms having degree from $n/2$ to $n-1$, inclusive)
$B_0 \leftarrow$ first n/2 terms of $b(x)$ (terms having degree from 0 to $n/2-1$, inclusive)
$B_1 \leftarrow$ last n/2 terms of $b(x)$ (terms having degree from $n/2$ to $n-1$, inclusive)
$C_0 \leftarrow$ TwoWaySBPM(A_0, B_0)
$D_0 \leftarrow$ TwoWaySBPM(A_0, B_1)
$D_1 \leftarrow$ TwoWaySBPM(A_1, B_0)
$C_2 \leftarrow$ TwoWaySBPM(A_1, B_1)
$C_1 \leftarrow D_0 + D_1$
$c(x) \leftarrow C_0 + C_1x^{n/2} + C_2x^n$

2.2 Karatsuba Polynomial Multiplication Algorithm

We now give the definition of the 2-way Karatsuba polynomial multiplication algorithm in [14] for computing the product of two n-term polynomials.

Definition 2.3. [2-way Karatsuba Polynomial Multiplication Algorithm] Let $a(x) = a_0 + a_1x + \cdots + a_{n-1}x^{n-1}$ and $b(x) = b_0 + b_1x + \cdots + b_{n-1}x^{n-1}$ be two polynomials over a field $\mathbb{F}$. Let us split each of $a(x)$ and $b(x)$ into two equal parts in the following way:

$$a(x) = A_0(x) + A_1(x)x^{n/2}$$

and

$$b(x) = B_0(x) + B_1(x)x^{n/2},$$

where $A_0(x)$, $A_1(x)$, $B_0(x)$, $B_1(x)$ are $n/2$-term polynomials. One can compute the product $a(x)b(x)$ as follows:

$$a(x)b(x) = C_0(x) + C_1(x)x^{n/2} + C_2(x)x^n, \tag{4}$$

where

$$\begin{aligned} C_0(x) &= A_0(x)B_0(x) \\ C_1(x) &= [A_0(x) + A_1(x)][B_0(x) + B_1(x)] - A_0(x)B_0(x) - A_1(x)B_1(x) \\ C_2(x) &= A_1(x)B_1(x) \end{aligned} \tag{5}$$

The algorithm for computing the product $c(x) = a(x)b(x)$ by plugging the results of (5) to Equation (4) is called *2-way Karatsuba polynomial multiplication algorithm.*

In Definition 2.3, to compute C_0, C_1 and C_2 after finding A_0, A_1, B_0, B_1, we recursively apply 2-way Karatsuba polynomial multiplication algorithm for computing the $n/2$-term polynomial multiplications $A_0(x)B_0(x)$, $A_1(x)B_1(x)$ and $[A_0(x) + A_1(x)][B_0(x) + B_1(x)]$ in the above equations for $C_0(x)$, $C_1(x)$ and $C_2(x)$, until we eventually end up with multiplicand polynomials of degree one and obtain $C_0(x)$, $C_1(x)$ and $C_2(x)$ by combining all these results appropriately. To multiply two polynomials using 2-way Karatsuba polynomial multiplcation algorithm, the following addition and multiplication operations will be executed:

- *Multiplications:* There are three $n/2$-term polynomial multiplications (i.e., A_0B_0, A_1B_1, $(A_0 + A_1)(B_0 + B_1)$, A_0B_1, A_1B_0). So, it takes $3M(n/2)$ operations as for multiplication.

- *Additions:* The addition of the $n/2$-term polynomials A_0 and A_1 takes $n/2$ single-word additions, similarly for B_0 and B_1 we have $n/2$ single-word additions. The addition of the $n-1$-term polynomials $(A_0 + A_1)(B_0 + B_1)$, A_0A_1 and B_0B_1 takes $2(n-1) = 2n-2$ double-word additions. Also, in the construction step of $a(x)b(x)$, we need to add C_0 and $C_1x^{n/2}$, which have $n/2-1$ overlapping terms, and so costs $n/2-1$ double-word additions. A similar reasoning for $C_1x^{n/2}$ and C_2x^n gives $n/2-1$ more overlapping terms, so this also costs $n/2-1$ double-word additions. Note that the overlapping terms of the first addition consist of terms of degree $n/2, n/2+1, \ldots, n-2$, whereas the second addition has overlapping terms of degree $n, n+1, \ldots, 3n/2-2$, which means that these three polynomials do not have a common overlapping term. So the total additions are $n/2 + n/2 = n$ single-word additions and $(2n-2) + (n/2-1) + (n/2-1) = 3n-4$ double-word additions.

In conclusion, we have the following recursive formula:

$$M(n) = 3M(n/2) + nA_s + (3n-4)A_d, \tag{6}$$

where A_s denotes the cost of single-word multiplication. Setting $M(1) = M$, where M is the cost of one multiplication, taking $n = 2^l$ and solving (6) gives

$$M(2^l) = 3^lM + 2(3^l - 2^l)A_s + (4 \cdot 3^l - 6 \cdot 2^l + 2)A_d$$

or

$$M(n) = n^{\log_2 3}M + 2(n^{\log_2 3} - n)A_s + (4n^{\log_2 3} - 6n + 2)A_d.$$

If we work on an implementation platform assuming equal unit costs of multiplication, single-word and double-word additions, we take $M = A_d = A_s = 1$ in (6). In this case, taking $n = 2^l$ and solving (6) gives

$$M(2^l) = 7 \cdot 3^l - 8 \cdot 2^l + 2$$

or

$$M(n) = 7n^{\log_2 3} - 8n + 2.$$

for all n, i.e., Karatsuba polynomial multiplication algorithm has a time complexity of $O(n^{\log_2 3})$. The pseudocode of Karatsuba Polynomial Multiplication Algorithm is given in Algorithm 2:

Algorithm 2 Karatsuba Polynomial Multiplication Algorithm (Karatsuba)

Input: Two n-term polynomials $a(x)$ and $b(x)$
Output: $c(x) = a(x)b(x)$

$A_0 \leftarrow$ first n/2 terms of $a(x)$ (terms having degree from 0 to $n/2-1$, inclusive)
$A_1 \leftarrow$ last n/2 terms of $a(x)$ (terms having degree from $n/2$ to $n-1$, inclusive)
$B_0 \leftarrow$ first n/2 terms of $b(x)$ (terms having degree from 0 to $n/2-1$, inclusive)
$B_1 \leftarrow$ last n/2 terms of $b(x)$ (terms having degree from $n/2$ to $n-1$, inclusive)
$C_0 \leftarrow$ Karatsuba(A_0, B_0)
$C_2 \leftarrow$ Karatsuba(A_1, B_1)
$A_2 \leftarrow A_0 + A_1$
$B_2 \leftarrow B_0 + B_1$
$K \leftarrow$ Karatsuba(A_2, B_2)
$C_1 \leftarrow K - C_0 - C_2$
$c(x) \leftarrow C_0 + C_1x^{n/2} + C_2x^n$

2.3 Toeplitz and Vandermonde Matrices

Here we shall give the definitions of Toeplitz matrices and Vandermonde matrices. We will make use of Vandermonde matrices in order to construct a k-way split Toeplitz matrix-vector product formula for any positive integer k, as explained in Section 4.

Definition 2.4 (Toeplitz matrix)**.** Let $m, n \in \mathbb{N}$. A matrix $T = (t_{i,j})_{m\times n}$ is called a *Toeplitz matrix* if $T_{i,j} = T_{i+1,j+1}$ for all $1 \leq i \leq m-1$ and $1 \leq j \leq n-1$.

Definition 2.5 (Vandermonde matrix)**.** Let $n \in \mathbb{N}$. A *Vandermonde matrix* $V \in$

$M_{n\times n}(\mathbb{F})$ over $\mathbb{F}$ is a matrix of the form

$$V = \begin{bmatrix} 1 & v_1 & v_1^2 & v_1^3 & \dots & v_1^{n-1} \\ 1 & v_2 & v_2^2 & v_2^3 & \dots & v_2^{n-1} \\ 1 & v_3 & v_3^2 & v_3^3 & \dots & v_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & v_n & v_n^2 & v_n^3 & \dots & v_n^{n-1} \end{bmatrix}_{n\times n},$$

where $v_i \in \mathbb{F}$ for all $i = 1, 2, \dots, n$.

Definition 2.6 (Semi-Vandermonde matrix)**.** Let $n \in \mathbb{N}$. A *semi-Vandermonde matrix* $V \in M_{n\times n}(\mathbb{F})$ is a matrix of the form

$$V = \begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 \\ 1 & v_1 & v_1^2 & v_2^3 & \dots & v_2^{n-1} \\ 1 & v_2 & v_2^2 & v_3^3 & \dots & v_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & v_{n-2} & v_{n-2}^2 & v_{n-2}^3 & \dots & v_{n-2}^{n-1} \\ 0 & 0 & 0 & 0 & \dots & 1 \end{bmatrix}_{n\times n},$$

where $v_i \in \mathbb{F}$ for all $i = 1, 2, \dots, n-2$.

In semi-Vandermonde matrices, the first row now has the form [1,0,...,0] and the last row [0,0,...,1], while all other rows stay unchanged in Vandermonde form.

2.4 2-way and 3-way Toeplitz Matrix-Vector Product Algorithms (TMVP)

Having defined Toeplitz matrices, we are now ready to define 2-way and 3-way Toeplitz matrix-vector product (TMVP) formulas that are given in [11], which, in Section 4, will be generalized to the general case of k-way TMVPs.

Definition 2.7. [The 2-way Split Toeplitz Matrix-Vector Product Algorithm] Let T be a Toeplitz matrix of dimensions $n \times n$, and V be a vector of dimensions $n \times 1$. Moreover, assume that we have

$$T = \begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \end{bmatrix}, V = \begin{bmatrix} V_0 \\ V_1 \end{bmatrix}$$

for Toeplitz matrices T_0, T_1, T_2 of dimensions $n/2 \times n/2$ and vectors V_0, V_1 of dimensions $n/2 \times 1$. The product $T \cdot V$ can be found via the following:

$$T \cdot V = \begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} = \begin{bmatrix} Q_1 + Q_2 \\ Q_1 + Q_3 \end{bmatrix}, \tag{7}$$

where

$$\begin{aligned} Q_1 &= T_1(V_0 + V_1) \\ Q_2 &= (T_0 - T_1)V_1 \\ Q_3 &= (T_2 - T_1)V_0. \end{aligned}$$

Then we calculate the smaller TMVPs of dimension $n/2$ in the equations for Q_1, Q_2 and Q_3 in a recursive manner using the divide-and-conquer method. Afterwards, we eventually find Q_1, Q_2 and Q_3 and recombine these results appropriately according to (7). This method is called *2-way split Toeplitz matrix-vector product algorithm*, or 2-way TMVP in short.

For an integer n, let $M(n)$ denote the total number of operations which occurs in the 2-way split TMVP. This algorithm will execute the following operations:

- *Multiplications:* There are 3 TMVP multiplications of dimension $n/2$, which results in 3M(n/2).
- *Matrix additions:* There are 2 Toeplitz matrix additions of dimension $n/2$, which gives $2(n-1)A_s$, where A_s is the cost of single-word addition, since a Toeplitz matrix of dimension $n/2$ can be identified by only $n-1$ entries.
- *Vector additions:* There is one vector addition of dimension $n/2$, which gives $n/2 \cdot A_s$.
- *Recombination:* After calculating Q_1, Q_2, Q_3, necessary recombination is performed to obtain the final result. This step requires two additions of Q_i's, which give $2(n/2)A_d = nA_d$, where A_d is the cost of double-word addition.

Combining all these costs, one obtains the following recurrence relation for $M(n)$:

$$M(n) = 3M(n/2) + (5n/2 - 2)A_s + nA_d. \tag{8}$$

Solving (8) for $M(n)$, where $n = 2^l$, gives

$$M(n) = n^{\log_2 3}M + (4n^{\log_2 3} - 5n + 1)A_s + 2(n^{\log_2 3} - n)A_d.$$

If we take $M(1) = M = A_s = A_d = 1$ in (8), then we have

$$M(n) = 7n^{\log_2 3} - 7n + 1$$

which yields a computational complexity of

$$O(n^{\log_2 3}) \approx O(n^{1.58}).$$

The pseudocode of 2-way TMVP algorithm is given by Algorithm 3:

Definition 2.8 (The 3-way Split Toeplitz Matrix-Vector Product Algorithm)**.** Let T be a Toeplitz matrix of dimensions $n \times n$, and V be a vector of dimensions $n \times 1$. Moreover, assume that we have

$$T = \begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{bmatrix}, V = \begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix}$$

Algorithm 3 2-way Split Toeplitz Matrix-Vector Product algorithm [TwoWayTMVP]

Input: T, V
Output: $R = T \cdot V$
$T_0 \leftarrow$ top right $n/2 \times n/2$ block of T
$T_1 \leftarrow$ top left $n/2 \times n/2$ block of T
$T_2 \leftarrow$ bottom left $n/2 \times n/2$ block of T
$V_0 \leftarrow$ first $n/2$ entries of V
$V_1 \leftarrow$ last $n/2$ entries of V
$A_0 \leftarrow V_0 + V_1$
$B_0 \leftarrow T_0 - T_1$
$B_1 \leftarrow T_2 - T_1$
$Q_1 \leftarrow \text{TwoWayTMVP}(T_1, A_0)$
$Q_2 \leftarrow \text{TwoWayTMVP}(B_0, V_1)$
$Q_3 \leftarrow \text{TwoWayTMVP}(B_1, V_0)$
$R_0 \leftarrow Q_1 + Q_2$
$R_1 \leftarrow Q_1 + Q_3$
$R \leftarrow \begin{bmatrix} R_0 \\ R_1 \end{bmatrix}$

for Toeplitz matrices T_0, T_1, T_2, T_3, T_4 of dimensions $n/3 \times n/3$ and vectors V_0, V_1, V_2 of dimensions $n/3 \times 1$. The product $T \cdot V$ can be found via the following equation:

$$T \cdot V = \begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} Q_3 + Q_4 + Q_6 \\ Q_2 - Q_4 + Q_5 \\ Q_1 - Q_2 - Q_3 \end{bmatrix}, \tag{9}$$

where

$$\begin{aligned} Q_1 &= (T_4 + T_3 + T_2)V_0 \\ Q_2 &= T_3(V_0 - V_1) \\ Q_3 &= T_2(V_0 - V_2) \\ Q_4 &= T_1(V_1 - V_2) \\ Q_5 &= (T_3 + T_2 + T_1)V_1 \\ Q_6 &= (T_2 + T_1 + T_0)V_2 \end{aligned}$$

Then we calculate the smaller TMVPs of dimension $n/3$ in the equations for Q_1, Q_2, ..., Q_6 in a recursive manner using the divide-and-conquer method. Afterwards, we eventually find $Q_1, Q_2, \ldots, Q_6$ and recombine these results appropriately according to (9). This method is called *3-way split Toeplitz matrix-vector product algorithm*, or 3-way TMVP in short.

For an integer n, let $M(n)$ denote the total number of operations which occurs in the 3-way split TMVP. This algorithm will execute the following operations:

- *Multiplications:* There are 6 TMVP multiplications of dimension $n/3$, which results in 6M(n/3).
- *Matrix additions:* There are 5 Toeplitz matrix additions of dimension $n/3$, which gives $5(2n/3-1)A_s$, where A_s is the cost of single-word addition, since a Toeplitz matrix of dimension $n/3$ can be identified by only $2(n/3)-1$ entries.
- *Vector additions:* There are 3 vector additions of dimension $n/3$, which gives $3(n/3)A_s = nA_s$.
- *Recombination:* After calculating Q_1, Q_2, Q_3, Q_4, Q_5, Q_6 necessary recombination is performed to obtain the final result. This step requires six additions of Q_i's, which give $6(n/3)A_d = 2nA_d$, where A_d is the cost of double-word addition.

Combining all these costs, one obtains the following recurrence relation for $M(n)$:

$$M(n) = 6M(n/3) + (13n/3 - 5)A_s + 2nA_d. \tag{10}$$

Solving (10) for $M(n)$, where $n = 3^l$, gives

$$M(n) = n^{\log_3 6}M + (\frac{10}{3}n^{\log_3 6} - \frac{13}{3}n + 1)A_s + 2(n^{\log_3 6} - n)A_d.$$

If we take $M(1) = M = A_s = A_d = 1$ in (10), then we have

$$M(n) = \frac{19}{3}n^{\log_3 6} - \frac{19}{3}n + 1$$

which yields a computational complexity of

$$O(n^{\log_3 6}) \approx O(n^{1.63}).$$

The pseudocode of 3-way TMVP algorithm is given in Algorithm 4:

Definitions 2.7 and 2.8 describe how to perform a n-dimensional TMVP by employing three smaller $n/2$-dimensional TMVPs (2-way split TMVP) and employing six smaller $n/3$-dimensional TMVPs (3-way split TMVP), respectively. A 4-way split TMVP formula that uses 7 TMVP subproblems and a 5-way split TMVP formula that uses 13 TMVP subproblems can also be found in [11]. We will generalize these cases by constructing a k-way split TMVP algorithm for any given $k \in \mathbb{N}$, in Section 4.

3 Results on k-way Split Polynomial Multiplication Algorithms

In this section, we prove that for any $k \in \mathbb{N}$, there exists a k-way split Schoolbook algorithm and a k-way split Karatsuba algorithm for multiplying two n-term polynomials that invoke k^2 and $\frac{k(k+1)}{2}$ sub-multiplications of n/k-term polynomials, respectively.

Algorithm 4 3-way Split Toeplitz Matrix-Vector Product Algorithm [ThreeWayTMVP]

Input: T, V

Output: $R = T \cdot V$

$T_0, T_1, T_2, T_3, T_4 \leftarrow n/3 \times n/3$-dimensional subblocks of T as described above

$V_0, V_1, V_2 \leftarrow$ partition of V into $n/3$-dimensional vectors as described above

$A_0 \leftarrow V_0 - V_1$

$A_1 \leftarrow V_0 - V_2$

$A_2 \leftarrow V_1 - V_2$

$S_1 \leftarrow T_2 + T_3$

$B_0 \leftarrow S_1 + T_4$

$B_1 \leftarrow S_1 + T_1$

$S_2 \leftarrow T_2 + T_1$

$B_2 \leftarrow S_2 + T_0$

$Q_1 \leftarrow \text{ThreeWayTMVP}(B_0, V_0)$

$Q_2 \leftarrow \text{ThreeWayTMVP}(T_3, A_0)$

$Q_3 \leftarrow \text{ThreeWayTMVP}(T_2, A_1)$

$Q_4 \leftarrow \text{ThreeWayTMVP}(T_1, A_2)$

$Q_5 \leftarrow \text{ThreeWayTMVP}(B_1, V_1)$

$Q_6 \leftarrow \text{ThreeWayTMVP}(B_2, V_2)$

$R_0 \leftarrow Q_3 + Q_4 + Q_6$

$R_1 \leftarrow Q_2 - Q_4 + Q_5$

$R_2 \leftarrow Q_1 - Q_2 - Q_3$

$$R \leftarrow \begin{bmatrix} R_0 \\ R_1 \\ R_2 \end{bmatrix}$$

3.1 The Existence of a k-way Schoolbook Polynomial Multiplication Algorithm

We show the existence and construction of a k-way split Schoolbook polynomial multiplication algorithm, that is, a polynomial multiplication algorithm that divides the main multiplicand polynomials into k equal parts in each recursive step of the application, and derive a recurrence relation and a complexity formula for the newly constructed k-way Schoolbook polynomial multiplication algorithm.

Theorem 3.1. *For each $k \in \mathbb{N}$, there exists a k-way split Schoolbook polynomial multiplication algorithm S_k satisfying the recursive relation*

$$S_k(n) = k^2 S_k(n/k) + (k-1)(2n-k-1)A_d.$$

In particular, its asymptotic complexity is $O(n^2)$.

Proof. Let $A(x)$ and $B(x)$ be n-term polynomials. Write

$$A(x) = A_0(x) + yA_1(x) + \cdots + y^{k-1}A_{k-1}(x)$$

and

$$B(x) = B_0(x) + yB_1(x) + \cdots + y^{k-1}B_{k-1}(x),$$

where $A_0, A_1, \ldots, A_{k-1}, B_0, B_1, \ldots, B_{k-1}$ are n/k-term polynomials, and $y = x^{n/k}$. We can compute

$$\begin{aligned} C(x) &= A(x)B(x) \\ &= A_0B_0 + y(A_0B_1 + A_1B_0) + \cdots + y^{k-1}(A_0B_{k-1} + A_1B_{k-2} + \cdots + A_{k-1}B_0) \\ &+ y^k(A_1B_{k-1} + A_2B_{k-2} + \cdots + A_{k-1}B_1) + \cdots + y^{2k-2}A_{2k-2}B_{2k-2} \\ &= C_0 + C_1y + C_2y^2 + \cdots + C_{2k-2}y^{2k-2} \end{aligned}$$

This proves the existence of the k-way Schoolbook polynomial multiplication algorithm. We now compute the minimum amount of operations needed to employ this algorithm, categorized as follows:

- *Multiplications:* There are k^2 many multiplications of the form A_iB_j, where A_i and B_j are n/k-term polynomials in the above, resulting in $k^2S_k(n/k)$.
- *Additions (of A_i's and B_j's):* Addition of two polynomials of the form A_iB_j takes $2n/k - 1$ double-word additions. Accordingly, since there are $(k-1)^2$ such additions, this yields a cost of $(k-1)^2(2n/k-1)A_d$.
- *Additions from overlaps:* The polynomial A_0B_0 has terms of degree from 0 to $2n/k - 2$, $y(A_0B_1 + A_1B_0)$ has terms of degree from n/k to $3n/k - 2$, and so on, until $y^{2k-2}A_{2k-2}B_{2k-2}$ has terms of degree from $2n - 2n/k$ to $2n - 2$. There are $n/k - 1$ overlapping terms for each pair of consecutive summands in the resulting product polynomial. Since there are $2k-1$ summands in the product, the number of consecutive summands is $2k - 2$ and so this results in $2(k-1)(n/k-1)A_d$.

Combining all of these, we obtain

$$\begin{aligned} S_k(n) &= k^2 S_k(n/k) + (k-1)^2(2n/k-1)A_d + 2(k-1)(n/k-1)A_d \\ &= k^2 S_k(n/k) + (k-1)(2n-k-1)A_d. \end{aligned} \tag{11}$$

Setting $n = k^l$ and $S_k(1) = M$, solving (11) for $S_k(n)$ gives

$$S_k(k^l) = k^{2l}M + (k^{2l} - 2k^l + 1)A_d,$$

or

$$S_k(n) = n^2 M + (n^2 - 2n + 1)A_d.$$

Taking $S_k(1) = M = A_d = 1$ in (11) gives

$$S_k(n) = 2n^2 - 2n + 1.$$

Hence, the asymptotic complexity of the k-way Schoolbook polynomial multiplication algorithm is $O(n^2)$. □

3.2 The Existence of a k-way Karatsuba Polynomial Multiplication Algorithm

We show the existence and construction of a k-way split Karatsuba polynomial multiplication algorithm, that is, a polynomial multiplication algorithm that divides the main multiplicand polynomials into k equal parts in each recursive step of the application, and derive a recurrence relation and a complexity formula for the newly constructed k-way Karatsuba polynomial multiplication algorithm.

Theorem 3.2. *For each $k \in \mathbb{N}$, there exists a k-way split Karatsuba polynomial multiplication algorithm K_k satisfying the recursive relation*

$$K_k(n) = \frac{k(k+1)}{2} K_k(n/k) + (k-1)(3n - \frac{3k}{2} - 1)A_d + n(k-1)A_s.$$

In particular, its asymptotic complexity is $O(n^{\log_k(\frac{k(k+1)}{2})})$.

Proof. Let $A(x)$, $B(x)$ and $A_0, A_1, \ldots, A_{k-1}, B_0, B_1, \ldots, B_{k-1}$ be as above. Observe that we can write the sums of terms of the form A_iB_j, which lie in the product polynomial $C(x) = A(x)B(x)$, in the following way:

$$\begin{aligned} C_0 &= A_0B_0 \\ C_1 &= (A_0 + A_1)(B_0 + B_1) - A_0B_0 - A_1B_1 \\ C_2 &= (A_0 + A_1 + A_2)(B_0 + B_1 + B_2) - (A_0 + A_1)(B_0 + B_1) \\ &\quad - (A_1 + A_2)(B_1 + B_2) + 2A_1B_1 \\ C_3 &= (A_0 + A_1 + A_2 + A_3)(B_0 + B_1 + B_2 + B_3) \\ &\quad - (A_0 + A_1 + A_2)(B_0 + B_1 + B_2) - (A_1 + A_2 + A_3)(B_1 + B_2 + B_3) \\ &\quad + 2(A_1 + A_2)(B_1 + B_2) - A_1B_1 - A_2B_2 \\ &\vdots \\ C_{2k-3} &= (A_{k-1} + A_{k-2})(B_{k-1} + B_{k-2}) - A_{k-1}B_{k-1} - A_{k-2}B_{k-2} \\ C_{2k-2} &= A_{k-1}B_{k-1} \end{aligned}$$

and so on. In other words, if we fix an index $0 \leq j \leq k-1$, we have

$$C_j = \sum_{i=0}^{j} A_i \sum_{i=0}^{j} B_i - \sum_{i=0}^{j-1} A_i \sum_{i=0}^{j-1} B_i - \sum_{i=1}^{j} A_i \sum_{i=1}^{j} B_i + 2\sum_{i=1}^{j-1} A_i \sum_{i=1}^{j-1} B_i + \ldots, \quad (12)$$

where the coefficients of each term of the form $\sum A_i \sum B_i$ in the right hand side of (12) create a sequence

$$1, -1, -1, 2, -1, -1, 2, -1, -1, 2, ...$$

with the subsequence $-1, -1, 2$ repeating eventually. This proves the existence of the k-way Karatsuba polynomial multiplication algorithm. We now compute the minimum amount of operations needed to employ this algorithm, categorized as follows:

- *Multiplications:* There are $k(k+1)/2$ many multiplications of n/k-term polynomials in the above, resulting in $\frac{k(k+1)}{2} K_k(n/k)$.
- *Additions of A_i's and B_j's:* To obtain the sums of A_i's and sums of B_i's within parentheses, $k(k-1)$ additions of n/k-term polynomials, or $k(k-1)\frac{n}{k} = n(k-1)$ single-word additions, are sufficient. Also, addition of two polynomials of the form $A_i(x)B_j(x)$ takes $2n/k - 1$ double-word additions, and we also need to perform additions on the terms of the form $\sum_{i \in I \subset \mathbb{N}} A_i \sum_{i \in I \subset \mathbb{N}} B_i$. A short computation implies that $\frac{1}{2}(k-1)(3k-2)$ such double-word additions are sufficient, resulting in an overall total of $\frac{1}{2}(k-1)(3k-2)(\frac{2n}{k} - 1)A_d + n(k-1)A_s$.
- *Additions from overlaps:* This is identical to the analysis in the proof about k-way Schoolbook polynomial multiplication algorithm. So this results in $2(k-1)(n/k-1)A_d$.

Therefore, the resulting recursive relation is

$$K_k(n) = \frac{k(k+1)}{2} K_k(n/k) + (k-1)(3n - \frac{3k}{2} - 1)A_d + n(k-1)A_s. \quad (13)$$

Setting $n = k^l$, $K_k(1) = M$ and solving (13) for $K_k(n)$ gives

$$K_k(n) = n^{\log_k \frac{k(k+1)}{2}} M + 2(n^{\log_k \frac{k(k+1)}{2}} - n)A_s + [\frac{3k+10}{k+2} n^{\log_k \frac{k(k+1)}{2}} - 6n + \frac{3k+2}{k+2}]A_d.$$

If we take $K_k(1) = M = A_s = A_d = 1$ in (13), then we have

$$K_k(n) = \frac{6k+16}{k+2} n^{\log_k \frac{k(k+1)}{2}} - 8n + \frac{3k+2}{k+2}.$$

Hence, the asymptotic complexity of the k-way Karatsuba polynomial multiplication algorithm is $O(n^{\log_k(\frac{k(k+1)}{2})})$, as desired. □

4 Results on k-way Split Toeplitz Matrix-Vector Product Algorithm

In this section, we will prove and construct a formula for k-way split Toeplitz matrix-vector product (TMVP) which yields exactly $2k-1$ TMVP-subproblems of smaller dimensions. We will also derive a recurrence relation upon recursive utilization of the newly constructed k-way TMVP algorithm. Then, in Section 4.2, we give certain examples of k-way TMVPs with $2k-1$ TMVP subproblems based on the technique used in proving the existence of a k-way TMVP formula, for small values of k.

4.1 The Existence of a k-way Toeplitz Matrix-Vector Product Algorithm with Exactly $2k-1$ TMVP Sub-Problems

For a Toeplitz matrix T over a field $\mathbb{F}$ and a vector V over $\mathbb{F}$ of dimension n, and for any integer k, we will prove that there exists a k-way split TMVP formula which employs at most $2k-1$ smaller TMVP subproblems of dimension n/k, by imitating the methods described in [11].

Theorem 4.1. *Let $T \in \mathbb{F}^{n\times n}$ be a $n \times n$ Toeplitz matrix, and $V \in \mathbb{F}^n$ be a $n \times 1$ vector, and $k \in \mathbb{N}$. Then there exists a k-way split TMVP algorithm T_k for calculating $T \cdot V$, in which only $2k-1$ TMVP subproblems of dimension n/k are created upon recursion, which satisfies the recurrence relation*

$$T_k(n) = (2k-1)T_k(\frac{n}{k}) + (k^2-k-1)\frac{n}{k}A_s + f(k)(2\frac{n}{k}-1)A_s + (k^2-3)\frac{n}{k}A_d$$

for some function $f : \mathbb{N} \to \mathbb{N}$ satisfying $f(k) \leq (2k-2)(2k-3)$. In particular, its asymptotic complexity is $O(n^{\log_k(2k-1)})$.

Proof. Let $a(x) = a_0 + a_1x + \cdots + a_{k-1}x^{k-1}$ and $b(x) = b_0 + b_1x + \cdots + b_{k-1}x^{k-1}$ be two polynomials of degree $k-1$. Let

$$c(x) = a(x)b(x) = c_0 + c_1x + \cdots + c_{2k-2}x^{2k-2}.$$

From Definition 2.1, the coefficients c_j for $j \in \{0, 1, \ldots, 2k-2\}$ can be found by (14):

$$\begin{aligned}
c_0 &= a_0b_0 \\
c_1 &= a_0b_1 + a_1b_0 \\
c_2 &= a_0b_2 + a_1b_1 + a_2b_0 \\
&\cdots \\
c_{k-1} &= a_0b_{k-1} + a_1b_{k-2} + a_2b_{k-3} + \cdots + a_{k-3}b_2 + a_{k-2}b_1 + a_1b_0 \qquad (14) \\
&\cdots \\
c_{2k-4} &= a_{k-3}b_{k-1} + a_{k-2}b_{k-2} + a_{k-1}b_{k-3} \\
c_{2k-3} &= a_{k-2}b_{k-1} + a_{k-1}b_{k-2} \\
c_{2k-2} &= a_{k-1}b_{k-1}
\end{aligned}$$

We can express this linear system as

$$\vec{c} = M_c \cdot \vec{ab}, \tag{15}$$

where

$$\vec{c} = [c_0, c_1, \ldots, c_{2k-2}]^T, \vec{ab} = [a_0b_0, a_0b_1, a_0b_2, \ldots, a_{k-1}b_{k-1}]^T$$

and M_c is the coefficient matrix of the right hand side with respect to the basis vector $\vec{a}$. Note that M_c has dimensions $(2k-1) \times k^2$. Let $P_1, P_2, \ldots, P_m$ be products of the form

$$(a_{j_1} + a_{j_2} + \cdots + a_{j_r})(b_{j_1} + b_{j_2} + \cdots + b_{j_r})$$

for some subset $\{j_1, j_2, \ldots, j_r\} \subseteq \{1, 2, \ldots, k-1\}$. As happens with the vectors c_i, if we denote

$$\vec{P} = [P_1, P_2, \ldots, P_m]^T$$

then

$$\vec{P} = M_p \cdot \vec{ab} \tag{16}$$

where M_p is the coefficient matrix of the expressions $P_1, P_2, \ldots, P_m$ with respect to the basis vector $\vec{ab}$. Note that M_p has dimensions $m \times k^2$.

Claim 1. *Assume that there is a matrix M of dimensions $(2k-1) \times m$, such that $\vec{c} = M \cdot \vec{P}$. Then there exists a k-way split TMVP algorithm that creates m TMVP subproblems of dimension n/k upon recursion.*

Proof. Suppose that there exists a matrix M with $2k-1$ rows and m columns, satisfying

$$\vec{c} = M \cdot \vec{P}, \tag{17}$$

Then, using the ideas and steps given in [11], one can build a k-way TMVP algorithm as follows using the methods from [21] and [22]: There are exactly $2k-1$ equations in (14). We multiply the first equation by T_{2k-2}, the second by T_{2k-3}, and so on, until we multiply the last equation by T_0 and sum all the resulting equations side-by-side and group the left hand side by b_i's. We have the following:

$$\begin{aligned}
&(T_{2k-2}a_0 + T_{2k-3}a_1 + \cdots + T_{k-1}a_{k-1})b_0 \\
&+ (T_{2k-3}a_0 + T_{2k-4}a_1 + \cdots + T_{k-2}a_{k-1})b_1 \\
&+ \ldots \\
&+ (T_{k-1}a_0 + T_{k-2}a_1 + \cdots + T_0a_{k-1})b_{k-1} \\
&= T_{2k-2}c_0 + T_{2k-3}c_1 + \cdots + T_0c_{2k-2}
\end{aligned} \tag{18}$$

Let $\vec{T}$ be defined as

$$\vec{T} := \begin{bmatrix} T_{2k-2} & T_{2k-1} & \ldots & T_0 \end{bmatrix}$$

(18) can then be written as

$$\begin{bmatrix} T_{2k-2}a_0 + T_{2k-3}a_1 + \cdots + T_{k-1}a_{k-1} \\ T_{2k-3}a_0 + T_{2k-4}a_1 + \cdots + T_{k-2}a_{k-1} \\ \vdots \\ T_{k-1}a_0 + T_{k-2}a_1 + \cdots + T_0a_{k-1} \end{bmatrix}^T \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{k-1} \end{bmatrix} = \vec{T} \cdot M \cdot \vec{P} \tag{19}$$

Let $P_i = P_i(a)P_i(b)$ for $i = 1, 2, \ldots, m$. We can now express the right hand side of (19) as

$$\vec{T} \cdot M \cdot \vec{P} = ((\vec{T} \cdot M) \odot \vec{P}(a)^T) \cdot \vec{P}(b),$$

where the operator $\odot$ stands for componentwise multiplication, $\vec{P}(a) = [P_1(a), \ldots, P_m(a)]^T$ and $\vec{P}(b) = [P_1(b), \ldots, P_m(b)]^T$. Let

$$\vec{Q} := (\vec{T} \cdot M) \odot \vec{P}(a)^T$$

and M_a and M_b be matrices of dimension $m \times k$ that satisfy

$$\vec{P}(a) := M_a \cdot \vec{a}, \vec{P}(b) := M_b \cdot \vec{b},$$

where

$$\vec{a} := [a_0, \ldots, a_{k-1}]^T, \vec{b} := [b_0, \ldots, b_{k-1}]^T.$$

Note that the entries of the vector $\vec{Q}$ constitute the building blocks of the entries of the wanted TMVP, and are formed by multiplying a linear combination of T_i's by a linear combination of a_i's. Plugging the $\odot$-expression found for $\vec{T} \cdot M \cdot \vec{P}$ and the definitions of Q and $\vec{P}$ to (19), we obtain

$$\begin{bmatrix} T_{2k-2}a_0 + T_{2k-3}a_1 + \cdots + T_{k-1}a_{k-1} \\ T_{2k-3}a_0 + T_{2k-4}a_1 + \cdots + T_{k-2}a_{k-1} \\ \vdots \\ T_{k-1}a_0 + T_{k-2}a_1 + \cdots + T_0 a_{k-1} \end{bmatrix}^T = Q \cdot M_b. \tag{20}$$

Taking the transpose of both sides of (20) gives

$$\begin{bmatrix} T_{2k-2}a_0 + T_{2k-3}a_1 + \cdots + T_{k-1}a_{k-1} \\ T_{2k-3}a_0 + T_{2k-4}a_1 + \cdots + T_{k-2}a_{k-1} \\ \vdots \\ T_{k-1}a_0 + T_{k-2}a_1 + \cdots + T_0 a_{k-1} \end{bmatrix} = M_b^T \cdot Q^T. \tag{21}$$

Now turning the rows of both sides of (21) and rewriting the left hand side of it in the matrix form gives

$$\begin{bmatrix} T_{k-1} & T_{k-2} & \ldots & T_0 \\ T_k & T_{k-1} & \ldots & T_1 \\ \vdots & \vdots & \ddots & \vdots \\ T_{2k-2} & T_{2k-3} & \ldots & T_{k-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix} = N_b^T \cdot Q^T, \tag{22}$$

where

$$\begin{aligned} \vec{Q} = (\vec{T} \cdot M) \odot \vec{P}(a)^T &= (\vec{T} \cdot M) \odot (M_a \cdot \vec{a})^T \\ &= (\begin{bmatrix} T_{2k-2} & T_{2k-1} & \ldots & T_0 \end{bmatrix} \cdot M) \odot (M_a \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix})^T \end{aligned}$$

and N_b^T is formed from M_b^T by turning its rows upside down, that is, the first row becomes the last, second becomes the second-last, and so on. The left hand side of (22) is a Toeplitz matrix-vector product. The right hand side of (22) is a vector whose entries are linear combinations of the entries $Q_1, Q_2, \ldots, Q_m$ of Q. If we regard each T_i as a Toeplitz matrix and each a_i as a vector, it suffices to compute the entries of $N_b^T \cdot Q^T$, all of which are sums of $Q_1, Q_2, \ldots, Q_m$. Moreover, by definition, each Q_i is a TMVP subproblem. This means that computing m TMVPs is sufficient for us to find the initial product. This proves the existence of a k-way Toeplitz matrix-vector product algorithm that uses m TMVP subproblems. □

Claim 2. *If the matrix M exists as described in Claim 1, then $m \geq 2k-1$.*

Proof. From Equations (15), (16) and (17), we have

$$M \cdot M_p = M_c$$

Now, M_p has m rows, and the $2k-1$ rows of M_c are linearly independent by the construction of M_c, since each term of the form $a_i b_j$ for $i, j \in \{0, \ldots, k-1\}$ is contained in exactly one of $c_0, \ldots, c_{2k-2}$. Therefore

$$m = \text{rank}(M_p) \geq \text{rank}(M \cdot M_p) = \text{rank}(M_c) = 2k-1$$

so $m \geq 2k-1$. □

Claim 3. *If $m = 2k-1$, then there exists a $m \times 1$-vector P and a $(2k-1) \times m$-matrix M such that $\vec{c} = M \cdot \vec{P}$.*

Proof. For $m = 2k-1$, the matrix M becomes a square matrix and it is invertible, since M_c is invertible as all of its rows are linearly independent. We can easily see that the matrix

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \ldots & 0 \\ 1 & 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & -1 & 1 & -1 & 1 & \ldots & 1 \\ 1 & 2 & 4 & 8 & 16 & \ldots & 2^{2k-2} \\ 1 & -2 & 4 & -8 & 16 & \ldots & 2^{2k-2} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & k-1 & (k-1)^2 & (k-1)^3 & (k-1)^4 & \ldots & (k-1)^{2k-2} \\ 0 & 0 & 0 & 0 & 0 & \ldots & 1 \end{bmatrix}^{-1}_{(2k-1)\times(2k-1)}$$

and the vector $P = (M^{-1} \cdot U \cdot \vec{a}) \odot (M^{-1} \cdot U \cdot \vec{b})$, where $U := \begin{bmatrix} I_{k\times k} \\ 0_{(k-1\times k-1)} \end{bmatrix}_{2k-1\times k}$

are as required. More precisely, one can write

$$
\begin{aligned}
P_1 &= a_0 b_0 \\
P_2 &= (a_0 + a_1 + a_2 + \cdots + a_{k-1})(b_0 + b_1 + b_2 + \cdots + b_{k-1}) \\
P_3 &= (a_0 - a_1 + a_2 + \cdots + (-1)^{k-1} a_{k-1})(b_0 - b_1 + b_2 + \cdots + (-1)^{k-1} b_{k-1}) \\
P_4 &= (a_0 + 2a_1 + 4a_2 + \cdots + 2^{k-1} a_{k-1})(b_0 + 2b_1 + 4b_2 + \cdots + 2^{k-1} b_{k-1}) \\
P_5 &= (a_0 - 2a_1 + 4a_2 + \cdots + (-2)^{k-1} a_{k-1})(b_0 - 2b_1 + 4b_2 + \cdots + (-2)^{k-1} b_{k-1}) \\
P_6 &= (a_0 + 3a_1 + 9a_2 + \cdots + 3^{k-1} a_{k-1})(b_0 + 3b_1 + 9b_2 + \cdots + 3^{k-1} b_{k-1}) \\
&\cdots \\
P_{m-1} &= (a_0 + (k-1)a_1 + \cdots + (k-1)^{k-1} a_{k-1})(b_0 + (k-1)b_1 + \cdots + (k-1)^{k-1} b_{k-1}) \\
P_m &= a_{k-1} b_{k-1}.
\end{aligned}
$$

This finishes the proof of Claim 3. □

That there exists a k-way TMVP algorithm as required immediately follows from Claims 1 and 3. Now we prove the second part of the theorem by establishing a recursive complexity relation for the k-way TMVP algorithm. To do this, we find the cost of each of the addition and multiplication operations that the k-way TMVP algorithm executes.

- *Multiplications:* Multiplication operations stem only from the TMVP subproblems of dimension n/k created upon the division of the main TMVP problem, which there are $2k-1$ of them. Therefore, multiplications have a total cost of $(2k-1)T_k(n/k)$.

- *Additions:* We are now ready to calculate the amount of single-word and double-word additions that are present in the k-way TMVP algorithm. Single-word additions comprise the matrix addition of the smaller submatrices, namely, $T_0, \ldots, T_{2k-2}$, and the smaller vectors created in the recursive step, namely, $a_0, \ldots, a_{k-1}$. Double-word additions consist of products of the form TA, where T is a linear expression of T_i's and A is a linear expression of a_i's, and are generated in the reconstruction step. Detailed investigation of additions are given in the categories below:

  - *Single-word additions coming from linear combinations of T_i's:* We remember that the linear combination of T_i's that are wanted to establish the final TMVP result lie in the vector $\vec{T} \cdot M$. Note that since the first and last row of M has only one nonzero entry, and other entries of M may or may not be zero. Therefore, the number of additions of two T-matrices that are required is upper-bounded by $(2k-2)(2k-3)$. Each of these additions take $2(n/k-1)A_s$ time.
  - *Single-word additions coming from linear combinations of a_i's:* The linear combination of a_i's that are wanted to establish the final TMVP

result lie in the vector

$$M_a \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & \cdots & 1 \\ 1 & -1 & 1 & \cdots & (-1)^{k-1} \\ 1 & 2 & 4 & \cdots & 2^{k-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & k-1 & (k-1)^2 & \cdots & (k-1)^{k-1} \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{k-1} \end{bmatrix}$$

$$= \begin{bmatrix} a_0 \\ a_0 + a_1 + \cdots + a_{k-1} \\ a_0 - a_1 + \cdots + (-1)^{k-1} a_{k-1} \\ a_0 + 2a_1 + \cdots + 2^{k-1} a_{k-1} \\ a_0 - 2a_1 + \cdots + (-2)^{k-1} a_{k-1} \\ a_0 + (k-1)a_1 + \cdots + (k-1)^{k-1} a_{k-1} \\ a_{k-1} \end{bmatrix}$$

Grouping the like-combinations of sum of terms of this vector allows us that $k^2 - k - 1$ a_i-additions are needed. Each of these additions take $n/k \cdot A_s$ time.

- *Double-word additions coming from linear combinations of Q_i's:* The linear combination of Q_i's that are wanted to establish the final TMVP result lie in the vector

$$N_b^T \cdot Q^T = \begin{bmatrix} 0 & 1 & (-1)^{k-1} & 2^{k-1} & \cdots & (k-1)^{k-1} & 1 \\ 0 & 1 & (-1)^{k-2} & 2^{k-2} & \cdots & (k-1)^{k-2} & 0 \\ 0 & 1 & (-1)^{k-3} & 2^{k-3} & \cdots & (k-1)^{k-3} & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & -1 & 2 & \cdots & k-1 & 0 \\ 1 & 1 & 1 & 1 & \cdots & 1 & 0 \end{bmatrix} \begin{bmatrix} Q_1 \\ Q_2 \\ Q_3 \\ \vdots \\ Q_{2k-1} \end{bmatrix}$$

Similarly, grouping the like-combinations of sum of terms of this vector allows us that $k^2 - 3$ Q_i-additions are needed. Each of these additions take $n/k \cdot A_d$ time.

Now that we have established all sorts of operations that are needed, the recursive relation for the complexity of the k-way TMVP algorithm is given in Equation (23):

$$T_k(n) = (2k-1)T_k(\frac{n}{k}) + (k^2 - k - 1)\frac{n}{k}A_s + f(k)(2\frac{n}{k} - 1)A_s + (k^2 - 3)\frac{n}{k}A_d \quad (23)$$

where $f : \mathbb{N} \to \mathbb{N}$ satisfies $f(k) \leq (2k-2)(2k-3)$. In an implementation platform where single and double word additions have equal costs, A_s and A_d will be taken as equal. If $A_s = A_d = 1$, then solving (23) for $n = k^l$ gives us

$$T_k(n) = n^{\frac{\ln(2k-1)}{\ln(k)}} \left[\frac{4k^2 - 10 + 3f(k)}{2k-2}\right] - \frac{2k^2 - k - 4 + 2f(k)}{k-1}n + \frac{f(k)}{2k-2}$$

Observe that since $f(k)$ is less than or equal to a quadratic function of k, the coefficients of $n^{\frac{\ln(2k-1)}{\ln(k)}}$ and n are less than or equal to linear functions of k. Therefore, applying the k-way TMVP algorithm to inputs of dimension n takes

$$O(n^{\log_k(2k-1)})$$

amount of time. This proves the complexity statement, and so Theorem 4.1. $\square$

4.2 Examples of k-way Toeplitz Matrix-Vector Product Algorithms for $k = 2$ and $k = 3$

Now, we give some examples of k-way TMVP constructions by assigning small positive integer values to k and following the steps in the Theorem 4.1. To do this, we first construct a vector $\vec{Q}$ and a matrix M_b, as defined in the proof of the Theorem 4.1. Then we find N_b and plug Q and N_b in (22) to obtain an equivalent formation for a k-way TMVP formula.

For $k = 2$, we have that

$$M = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & -1 \\ 0 & 0 & 1 \end{bmatrix}, U = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, M_b = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$$

and

$$Q = \begin{bmatrix} (T_2 - T_1)a_0 \\ T_1(a_0 + a_1) \\ (T_0 - T_1)a_1 \end{bmatrix}, N_b^T = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}, N_b^T Q^T = \begin{bmatrix} Q_2 + Q_3 \\ Q_1 + Q_2 \end{bmatrix}.$$

Therefore, we have

$$\begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} Q_2 + Q_3 \\ Q_1 + Q_2 \end{bmatrix}, \tag{24}$$

where

$$\begin{aligned} Q_1 &= (T_2 - T_1)a_0 \\ Q_2 &= T_1(a_0 + a_1) \\ Q_3 &= (T_0 - T_1)a_1. \end{aligned} \tag{25}$$

Applying the same discussions to $k = 3$, we have

$$\begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \\ T_4 & T_3 & T_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} Q_2 + Q_3 + 4Q_4 + Q_5 \\ Q_2 - Q_3 + 2Q_4 \\ Q_1 + Q_2 + Q_3 + Q_4 \end{bmatrix}, \tag{26}$$

where

$$
\begin{aligned}
Q_1 &= (T_4 - \frac{1}{2}T_3 - T_2 + \frac{1}{2}T_1)a_0 \\
Q_2 &= (T_3 + \frac{1}{2}T_2 - \frac{1}{2}T_1)(a_0 + a_1 + a_2) \\
Q_3 &= (-\frac{1}{3}T_3 + \frac{1}{2}T_2 - \frac{1}{6}T_1)(a_0 - a_1 + a_2) \\
Q_4 &= (-\frac{1}{6}T_3 + \frac{1}{6}T_1)(a_0 + 2a_1 + 4a_2) \\
Q_5 &= (2T_3 - T_2 - 2T_1 + T_0)a_2
\end{aligned}
\tag{27}
$$

The formulas in Equations (24) and (26) are 2-way and 3-way TMVP formulas, respectively. The equations defining Q_i's for the 2-way TMVP formula (24) and the 3-way TMVP formula (26) are given in Equations (25) and (27), respectively.

5 Efficiency Comparison Among Applications of Polynomial Multiplication Algorithms in Sequence

In this section, we prove some basic results about the efficiency order relations between combinations of Schoolbook and Karatsuba polynomial multiplication algorithms that are applied in sequence to an input of fixed size n. In order to do this, we need certain preliminary notations. Let S_k and K_k denote the k-way Schoolbook and Karatsuba polynomial multiplication algorithms, respectively, as constructed and defined in Section 3.

Definition 5.1. For the k-way formulas defined in Section 3,

- A_k denotes the k-way multiplication algorithm A that takes a k-sized input,
- $A_k \to B_l$ denotes the algorithm that first applies k-way algorithm A and then applies l-way algorithm B, and that takes a kl-sized input,
- we write $A >_n B$ or $B <_n A$ (respectively, $A =_n B$) if algorithm A has better (respectively, equal) performance than algorithm B in terms of operations needed, when both are applied to an n-sized input.
- we write $A \geq_n B$ if $A >_n B$ or $A =_n B$, and $A \leq_n B$ if $A <_n B$ or $A =_n B$.

Lemma 1. *Let $p, q \in \mathbb{N}$, and A, B and C be three algorithms. If $A_p >_p B_p$ (respectively, $A_p =_p B_p$), then $C_q \to A_p >_{pq} C_q \to B_p$ (respectively, $C_q \to A_p =_{pq} C_q \to B_p$).*

Proof. Both steps of the two sides of the latter inequality first apply a q-way algorithm C to a pq-sized input, and at the last step, p-way algorithms A and B are applied to the same polynomials produced from the first steps of both hand sides. The result follows. □

Lemma 2. *Let $p \geq 2$ be a positive integer. Then $S_p >_p K_p$.*

Proof. We have $S_p(p) = p^2 M(p/p) + (p-1)(2p-p-1) = 2p^2 - 2p + 1$ and $K_p(p) = \frac{p(p+1)}{2} M(p/p) + (p-1)(4p - \frac{3p}{2} - 1) = 3p^2 - 3p + 1$. Since $p \geq 2$, the result follows. □

Each of the five parts in Lemma 3 can be established by comparing the values of the number of operations on each of these algorithms by plugging their corresponding values to the formulas.

Lemma 3. *Let $p, q \geq 2$ be positive integers. Then*

a) $K_p \to S_q \geq_{pq} K_p \to K_q \geq_{pq} S_p \to K_q$.

b) If $q \geq 3$, then $K_p \to S_q \geq_{pq} S_p \to S_q \geq_{pq} S_p \to K_q$.

c) If $q = 2$, then $S_p \to S_q \geq_{pq} K_p \to S_q \geq_{pq} S_p \to K_q$.

d) $K_p \to K_q, K_q \to K_p \geq_{pq} K_{pq}$.

e) $S_p \to S_q =_{pq} S_q \to S_p =_{pq} S_{pq}$.

f) If $p < q$, then $K_p \to S_q \geq_{pq} K_q \to S_p \geq_{pq} K_p \to K_q \geq_{pq} K_q \to K_p \geq_{pq} S_q \to K_p \geq_{pq} S_p \to K_q$.

Proof. We only prove part a, the other parts can be proven similarly. We have

$$\begin{aligned}(K_p \to S_q)(pq) &= \frac{p(p+1)}{2} S_q(q) + (p-1)(4pq - \frac{3p}{2} - 1) \\ &= \frac{p(p+1)}{2}(2q^2 - 2q + 1) + (p-1)(4pq - \frac{3p}{2} - 1) \\ &= p^2q^2 + 3p^2q + pq^2 - p^2 - 5pq + p + 1.\end{aligned}$$

Similar calculations show that

$$(K_p \to K_q)(pq) = \frac{3}{2}p^2q^2 + \frac{5}{2}p^2q + \frac{3}{2}pq^2 - p^2 - \frac{11}{2}pq + p + 1$$

and

$$(S_p \to K_q)(pq) = 3p^2q^2 - p^2q - 2pq + 1.$$

Now observe that $K_p \to S_q \geq_{pq} K_p \to K_q$ if and only if

$$p^2q^2 - p^2q + pq^2 - pq = pq(p+1)(q-1) \geq 0,$$

which is always true if $q \geq 2$. The other inequality is similar to prove. □

Lemma 4 provides identities about ordering sequences of three or more Schoolbook k-way or Karatsuba k-way polynomial multiplication algorithms by efficiency.

Lemma 4. *Let $p, q, r \geq 2$ be positive integers, and A be any algorithm. Then*

a) If $A_q(q) > 4q - 1$, then $K_p \to A_q >_{pq} S_p \to A_q$.

b) If $p < q$, then $K_p \to K_q \to A_r >_{pqr} K_q \to K_p \to A_r$.

c) If $p < q$, then $K_p \to S_q \to S_2 >_{2pq} K_q \to S_p \to S_2$.

d) $K_p \to K_q \to A_r, K_q \to K_p \to A_r \geq_{pqr} K_{pq} \to A_r$.

e) $S_p \to S_q \to A_r =_{pqr} S_q \to S_p \to A_r =_{pqr} S_{pq} \to A_r$.

Proof. We only prove part a, other parts can be proved similarly. We have $K_p \to A_q >_{pq} S_p \to A_q$ if and only if

$$\frac{p(p+1)}{2}A_q(q) + (p-1)(4pq - \frac{3p}{2} - 1) < p^2 A_q(q) + (p-1)(2pq - p - 1)$$

if and only if $(p-1)(2pq - \frac{p}{2}) < \frac{p(p-1)}{2}A_q(q)$ if and only if $A_q(q) > 4q - 1$, so the result follows. □

One can prove Theorem 5.2 by applying the results of Lemmas 1, 2, 3 and 4 repeatedly.

Theorem 5.2. *Let $n \in \mathbb{N}$, and let $p_1, p_2, \ldots, p_m$ be the list of prime factors of n in non-decreasing order, where each prime divisor p of n is included in this list as many times as the highest power of p that divides n. Then we have the following:*

a) For any permutation σ of $(1, 2, \ldots, m)$, we have

$$S_{p_1} \to S_{p_2} \to \cdots \to S_{p_m} =_n S_{p_{\sigma(1)}} \to S_{p_{\sigma(2)}} \to \cdots \to S_{p_{\sigma(m)}} =_n S_n$$

and, moreover, we have the following inequality:

$$K_{p_1} \to K_{p_2} \to \cdots \to K_{p_m} \geq_n K_{p_{\sigma(1)}} \to K_{p_{\sigma(2)}} \to \cdots \to K_{p_{\sigma(m)}} \geq_n K_n,$$

with equality when σ is the identity permutation of the symmetric group of m elements.

b) If we fix a permutation σ of $(1, 2, \ldots, m)$, we have that

$$\begin{aligned} K_{p_1} \to K_{p_2} \to \cdots \to K_{p_{m-1}} \to S_{p_m} &\geq_n K_{p_{\sigma(1)}} \to K_{p_{\sigma(2)}} \to \cdots \to K_{p_{\sigma(m-1)}} \to S_{p_{\sigma(m)}} \\ &\geq_n K_{p_m} \to K_{p_{m-1}} \to \cdots \to K_{p_2} \to S_{p_1}. \end{aligned}$$

and

$$\begin{aligned} S_{p_m} \to S_{p_{m-1}} \to \cdots \to S_{p_2} \to K_{p_1} &\geq_n S_{p_{\sigma(1)}} \to S_{p_{\sigma(2)}} \to \cdots \to S_{p_{\sigma(m-1)}} \to K_{p_{\sigma(m)}} \\ &\geq_n S_{p_1} \to S_{p_2} \to \cdots \to S_{p_{m-1}} \to K_{p_m} \end{aligned}$$

c) *If we fix a permutation σ of $(1, 2, \ldots, m)$, and if $p_{\sigma(m)} \geq 3$ (respectively, $p_{\sigma(m)} = 2$), then we have that*

$$K_{p_{\sigma(1)}} \to K_{p_{\sigma(2)}} \to \cdots \to K_{p_{\sigma(m-1)}} \to S_{p_{\sigma(m)}} \geq_n (A_1)_{p_{\sigma(1)}} \to (A_2)_{p_{\sigma(2)}} \to \cdots \to (A_m)_{p_{\sigma(m)}}$$
$$\geq_n S_{p_{\sigma(1)}} \to S_{p_{\sigma(2)}} \to \cdots \to S_{p_{\sigma(m-1)}} \to K_{p_{\sigma(m)}},$$

(respectively,

$$K_{p_{\sigma(1)}} \to K_{p_{\sigma(2)}} \to \cdots \to S_{p_{\sigma(m-1)}} \to S_{p_{\sigma(m)}} \geq_n (A_1)_{p_{\sigma(1)}} \to (A_2)_{p_{\sigma(2)}} \to \cdots \to (A_m)_{p_{\sigma(m)}}$$
$$\geq_n S_{p_{\sigma(1)}} \to S_{p_{\sigma(2)}} \to \cdots \to S_{p_{\sigma(m-1)}} \to K_{p_{\sigma(m)}}),$$

where $A_i \in \{S, K\}$ for all $i = 1, \ldots, m$.

Proof. a) We have that $S_a \to S_b =_{ab} S_{ab}$ for any positive integers a and b, by Lemma 3.e. This, together with Lemma 1, implies that

$$\underbrace{S_{p_{\sigma(1)}} \to \cdots \to S_{p_{\sigma(m)}}}_{m \text{ terms}} =_n \underbrace{S_{p_{\sigma(1)}} \to \cdots \to S_{p_{\sigma(m-1)}p_{\sigma(m)}}}_{m-1 \text{ terms}} =_n \cdots =_n S_{p_{\sigma(1)}\cdots p_{\sigma(m)}} =_n S_n.$$

We have that $K_a \to K_b \geq_{ab} K_{ab}$ for any positive integers a and b, by Lemma 3.d. This, together with Lemma 1, implies that

$$\underbrace{K_{p_{\sigma(1)}} \to \cdots \to K_{p_{\sigma(m)}}}_{m \text{ terms}} \geq_n \underbrace{K_{p_{\sigma(1)}} \to \cdots \to K_{p_{\sigma(m-1)}p_{\sigma(m)}}}_{m-1 \text{ terms}} \geq_n \cdots \geq_n K_{p_{\sigma(1)}\cdots p_{\sigma(m)}} =_n K_n.$$

b) Suppose that we fix a permutation σ of $(1, 2, \ldots, m)$. The first chain of inequalities follows from repeated uses of Lemma 3.f and Lemma 4.b, and the second chain of inequalities follows from repeated uses of Lemma 3.f and Lemma 4.e.

c) In both cases, for the second inequality, assume that

$$(A_1)_{p_{\sigma(1)}} \to (A_2)_{p_{\sigma(2)}} \to \cdots \to (A_m)_{p_{\sigma(m)}}$$

is the worst sequence, where each $A_i \in \{S, K\}$. If $A_i = K$ for some $i = 1, 2, \ldots, m-1$, then we can change A_i to S and obtain a worse sequence, by Lemma 1 and Lemma 4.a. Thus $A_i = S$ for all $i = 1, 2, \ldots, m-1$. If $A_m = S$, then we can change A_m to K and obtain a worse sequence, by Lemmas 1 and 2. Thus $A_m = K$. To establish the first inequality, we consider the cases in question separately:

- *Case 1:* Assume first that $p_{\sigma(m)} \geq 3$. Assume that

$$(A_1)_{p_{\sigma(1)}} \to (A_2)_{p_{\sigma(2)}} \to \cdots \to (A_m)_{p_{\sigma(m)}}$$

is the best sequence, where each $A_i \in \{S, K\}$. If $A_i = S$ for some $i = 1, 2, \ldots, m-2$, then we can change A_i to K and obtain a better sequence,

by Lemma 1 and Lemma 4.a. Thus $A_i = K$ for all $i = 1, 2, \ldots, m-2$. If $A_m = K$, then we can change A_m to S and obtain a better sequence, by Lemmas 1 and 2. Thus $A_m = S$. If $A_{m-1} = S$, then we can change A_{m-1} to K and obtain a better sequence, by Lemma 1 and Lemma 3.b. Thus $A_{m-1} = K$. This yields the first inequality.

- *Case 2:* Now assume that $p_{\sigma(m)} = 2$. Assume that

$$(A_1)_{p_{\sigma(1)}} \to (A_2)_{p_{\sigma(2)}} \to \cdots \to (A_m)_{p_{\sigma(m)}}$$

is the best sequence, where each $A_i \in \{S, K\}$. If $A_i = S$ for some $i = 1, 2, \ldots, m-2$, then we can change A_i to K and obtain a better sequence, by Lemma 1 and Lemma 4.a. Thus $A_i = K$ for all $i = 1, 2, \ldots, m-2$. If $A_m = K$, then we can change A_m to S and obtain a better sequence, by Lemmas 1 and 2. Thus $A_m = S$. If $A_{m-1} = K$, then we can change A_{m-1} to S and obtain a better sequence, by Lemma 1 and Lemma 3.c. Thus $A_{m-1} = S$. This yields the first inequality.

This finishes the proof of the theorem. □

Theorem 5.3 establishes the best possible sequence of algorithms which comes from applying certain k-way Schoolbook or Karatsuba polynomial multiplication algorithms consecutively in order to multiply two n-term polynomials. In particular, the sequence becomes most efficient when all of its components other than the last one or two invoke the Karatsuba polynomial multiplication algorithm, and the last ones invoke Schoolbook polynomial multiplication algorithm.

Theorem 5.3. *Let $n \in \mathbb{N}$, and let $p_1, p_2, \ldots, p_m$ be the list of prime factors of n in non-decreasing order, where each prime divisor p of n is included in this list as many times as the highest power of p that divides n. Then the most efficient sequence of algorithms of the form*

$$(A_1)_{p_{\sigma(1)}} \to (A_2)_{p_{\sigma(2)}} \to \cdots \to (A_m)_{p_{\sigma(m)}}$$

is obtained by taking

$$A_1 = A_2 = \cdots = A_{m-1} = K, A_m = S,$$

unless $p_i = 2$ for all $i = 1, \ldots, m$, in which case we take

$$A_1 = A_2 = \cdots = A_{m-2} = K, A_{m-1} = A_m = S.$$

In either case, to obtain the best sequence, the permutation σ must be the identity permutation of the symmetric group of m elements.

Proof. We consider two cases.

- *Case 1:* Assume that $p_i \geq 3$ for some $i = 1, 2, \ldots, m$. Assume also that

$$(A_1)_{p_{\sigma(1)}} \to (A_2)_{p_{\sigma(2)}} \to \cdots \to (A_m)_{p_{\sigma(m)}}$$

is the best sequence, where each $A_i \in \{S, K\}$ and σ is a permutation of $(1, 2, \ldots, m)$. Note that no matter what the permutation σ is, we must have $A_1 = A_2 = \cdots = A_{m-2} = K$ and $A_m = S$, by Theorem 5.2.c. If $p_{\sigma(m)} \geq 3$, then $A_{m-1} = K$ by Theorem 5.2.c. Now suppose that $p_{\sigma(m)} = 2$ and $A_{m-1} = S$. In this case, if $p_{\sigma(m-1)} \geq 3$, then, by Lemma 1 and Lemma 3.e,

$$A \to S_{p_{\sigma(m-1)}} \to S_{p_{\sigma(m)}} =_n A \to S_{p_{\sigma(m)}} \to S_{p_{\sigma(m-1)}} \leq_n A \to K_{p_{\sigma(m)}} \to S_{p_{\sigma(m-1)}},$$

where A is the string of the remaining first $m-2$ terms, so we have a better sequence with $A_{m-1} = K$. Thus we must have $A_{m-1} = K$ as well. Finally, suppose both $p_{\sigma(m-1)} = p_{\sigma(m)} = 2$. Let j be the biggest index in $1, 2, \ldots, m-2$ such that $p_{\sigma(j)} \geq 3$. By using Lemma 4.b and Lemma 4.c repeatedly, we may bring $p_{\sigma(j)} \geq 3$ to be the subscript of the first S-term (the second-to-last term), and then proceed as above to obtain a better sequence with the second-to-last term being a K-term. Thus we must have $A_{m-1} = K$ as well. We now find the best sequence of the form

$$K_{p_{\sigma(1)}} \to K_{p_{\sigma(2)}} \to \cdots \to K_{p_{\sigma(m-1)}} \to S_{p_{\sigma(m)}}.$$

By Lemma 1, Lemma 3.f and Lemma 4.b, it is not hard to see that σ must be the identity permutation of the symmetric group of m elements, since if $p_{\sigma(j_1)} > p_{\sigma(j_2)}$ for some $j_1 < j_2$, then we may swap the corresponding terms and obtain a better sequence.

- *Case 2:* Assume that $p_i = 2$ for all $i = 1, 2, \ldots, m$. Our goal is to find the best sequence among the sequences of the form

$$(A_1)_2 \to (A_2)_2 \to \cdots \to (A_m)_2.$$

As above, we must have $A_1 = A_2 = \cdots = A_{m-2} = K$ and $A_m = S$ by Theorem 5.2.c. If $A_{m-1} = K$, by Lemma 1 and Lemma 3.c,

$$K_2 \to K_2 \to \cdots \to K_2 \to S_2 \to S_2 \geq_n K_2 \to K_2 \to \cdots \to K_2 \to K_2 \to S_2,$$

so we have $A_{m-1} = S$.

This finishes the proof of the theorem. □

6 Comparison with Literature

In this section, we compare our findings regarding the k-way TMVP algorithm from Section 4.1, the k-way Schoolbook polynomial multiplication algorithm from Section 3.1, and the k-way Karatsuba polynomial multiplication algorithm from Section 3.2 with established algorithms in the literature. In Section 6.1, for specific values of k, we compare the k-way TMVP algorithm established in Section 4.1 with various k-way TMVP algorithms available in the literature. To accomplish this,

the 5-way TMVP algorithm by Gokce [11], the unequal split algorithms by Paksoy [16], and the multiway splitting method by Hasan and Negre [12] will serve as benchmarks. The total arithmetic complexity and the precise number of operations will be utilized for comparison. Subsequently, in Section 6.2, for inputs of fixed sizes $n = 2^l$ and $n = 3^l$ (for $l \in \mathbb{N}$), the most efficient sequence of algorithms for n-term polynomial multiplications as defined in Theorem 5.3 is compared with the recursive applications of the 2-way and 3-way split polynomial multiplication algorithms described in [14].

6.1 Comparison of New TMVPs with Known TMVPs

For the values $k = 5, 6, 8$, and 12, we now compare our constructions of k-way Toeplitz matrix-vector products from Section 4 with the k-way TMVP formulas in the literature, evaluating the efficiency of these algorithms.

- **TMVP-5:** Let $T = \begin{bmatrix} T_4 & T_3 & T_2 & T_1 & T_0 \\ T_5 & T_4 & T_3 & T_2 & T_1 \\ T_6 & T_5 & T_4 & T_3 & T_2 \\ T_7 & T_6 & T_5 & T_4 & T_3 \\ T_8 & T_7 & T_6 & T_5 & T_4 \end{bmatrix}$ be a Toeplitz matrix and $V = \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix}$ be a vector. The product $T \cdot V$ can be calculated as proposed in [11]:

$$\begin{aligned} T \cdot V &= \begin{bmatrix} T_4 & T_3 & T_2 & T_1 & T_0 \\ T_5 & T_4 & T_3 & T_2 & T_1 \\ T_6 & T_5 & T_4 & T_3 & T_2 \\ T_7 & T_6 & T_5 & T_4 & T_3 \\ T_8 & T_7 & T_6 & T_5 & T_4 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} \\ &= \begin{bmatrix} Q_5 + Q_8 - Q_9 - Q_{11} + Q_{12} + Q_{13} \\ Q_4 + Q_9 - Q_{10} - Q_{12} - Q_{13} \\ Q_3 + Q_7 + Q_8 - Q_{10} + Q_{11} - Q_{13} \\ Q_2 - Q_6 + Q_{11} - Q_{12} - Q_{13} \\ Q_1 + Q_6 + Q_7 + Q_{10} + Q_{12} + Q_{13} \end{bmatrix}, \end{aligned} \tag{28}$$

where

$$
\begin{aligned}
Q_1 &= (T_8 + T_7 - T_6 + T_5 + T_4 - T_3)V_0 \\
Q_2 &= (T_7 + T_6 + T_4 + T_3)V_1 \\
Q_3 &= (-T_6 - T_5 + 2T_4 - T_3 - T_2)V_2 \\
Q_4 &= (T_5 + T_4 + T_2 + T_1)V_3 \\
Q_5 &= (-T_5 + T_4 + T_3 - T_2 + T_1 + T_0)V_4 \\
Q_6 &= (-T_7 - T_4)(V_0 - V_1) \\
Q_7 &= (T_6 + T_3)(V_0 + V_2) \\
Q_8 &= (T_5 + T_2)(V_2 + V_4) \\
Q_9 &= (-T_4 - T_1)(V_3 - V_4) \\
Q_{10} &= (-T_5 - T_4)(V_0 - V_2 - V_3) \\
Q_{11} &= (-T_4 - T_3)(V_1 + V_2 - V_4) \\
Q_{12} &= (-T_5 - T_3)(V_0 - V_1 - V_3 + V_4) \\
Q_{13} &= (T_5 + T_4 + T_3)(V_0 - V_1 - V_2 - V_3 + V_4)
\end{aligned}
$$

The equation for $T \cdot V$ given in (28) is a 5-way TMVP formula. Assuming unit costs for additions and multiplications ($A_s = A_d = M = 1$), the recursive complexity of this algorithm is given by

$$M(n) = 13M(n/5) + 13n - 18.$$

Conversely, our 5-way TMVP formula, obtained by setting $k = 5$ in Theorem 4.1 and utilizing the results from Section 4, possesses a recursive complexity of

$$M(n) = 9M(n/5) + 124n/5 - 41.$$

Consequently, our 5-way TMVP algorithm proves to be more efficient when $n \geq 125$, as detailed in Table 1.

- **TMVP-6**: Let $T = \begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \end{bmatrix}$ be a Toeplitz matrix and $V = \begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix}$ be a vector. The product $T \cdot V$ can be calculated via the following unequal split [16]:

$$T \cdot V = \begin{bmatrix} T_2 & T_1 & T_0 \\ T_3 & T_2 & T_1 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}Q_0 - \frac{1}{2}Q_1 + Q_2 \\ \frac{1}{2}Q_0 + \frac{1}{2}Q_1 + Q_3 \end{bmatrix}, \tag{29}$$

where

$$
\begin{aligned}
Q_0 &= (T_1 + T_2)(V_0 + V_1 + V_2) \\
Q_1 &= (T_1 - T_2)(V_0 - V_1 + V_2) \\
Q_2 &= (T_0 - T_2)V_2 \\
Q_3 &= (T_3 - T_1)V_0.
\end{aligned}
$$

Note that the matrix T in (29) possesses an unequal number of rows and columns when considering the T_i blocks as entries. Furthermore, if T is an $n \times n$ square matrix, then each T_i block has dimensions $(n/2) \times (n/3)$, meaning they are not square matrices. This specific unequally split TMVP formulation in (29) is termed a (2×3)-*way TMVP formula*. Similarly, by defining $T = \begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \\ T_3 & T_2 \end{bmatrix}$ and $V = \begin{bmatrix} V_0 \\ V_1 \end{bmatrix}$, a corresponding (3×2)-*way TMVP formula* can be constructed as given in [16]:

$$T \cdot V = \begin{bmatrix} T_1 & T_0 \\ T_2 & T_1 \\ T_3 & T_2 \end{bmatrix} \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} = \begin{bmatrix} \frac{1}{2}Q_0 + \frac{1}{2}Q_1 + Q_2 \\ \frac{1}{2}Q_0 - \frac{1}{2}Q_1 \\ \frac{1}{2}Q_0 + \frac{1}{2}Q_1 + Q_3 \end{bmatrix}, \tag{30}$$

where

$$\begin{aligned} Q_0 &= (T_1 + T_2)(V_0 + V_1) \\ Q_1 &= (T_1 - T_2)(V_0 - V_1) \\ Q_2 &= (T_0 - T_2)V_1 \\ Q_3 &= (T_3 - T_1)V_0. \end{aligned}$$

Given the (2×3)-way TMVP formula in (29), the (3×2)-way TMVP formula in (30), a square matrix T of dimension n (where n is a multiple of 6), and a vector V of length n, one can sequentially apply the (2×3)-way TMVP formula to yield smaller TMVPs involving an $(n/2) \times (n/3)$ matrix and an $(n/3) \times 1$ vector. Subsequently applying the (3×2)-way TMVP formula to these smaller matrices and vectors produces a composite 6-way TMVP algorithm, whose complexity is given by:

$$M(n) = 16M(n/6) + 47n/3 - 20.$$

In contrast, the complexity of our proposed 6-way TMVP formula (by setting $k = 6$ in Theorem 4.1) is given by:

$$M(n) = 11M(n/6) + 65n/2 - 66.$$

Additionally, there exists another implementation of a 6-way formula by Hasan and Negre [12]. Note that this is a particular case of the k-way TMVP formula initially proposed for $\mathbb{F}_2$ in [12], which is integrated into non-binary fields here for a comprehensive comparison. Let T and V be defined as before. The method in [12] for computing the product $T \cdot V$ exhibits the following complexity:

$$M(n) = \frac{k(k+1)}{2} M(n/k) + k(k-1)\left(\frac{7n}{2k} - 1\right).$$

Setting $k = 6$ yields:

$$M(n) = 21M(n/6) + 35n/2 - 30.$$

As a result, our 6-way TMVP algorithm proves to be more efficient when $n \geq 254$, as summarized in Table 1.

- **TMVP-8 and TMVP-12:** Utilizing similar methodologies, one can construct (2×4)-way, (3×4)-way, (4×2)-way, and (4×3)-way TMVPs [16]. By employing these unequal TMVPs in sequential combinations (e.g., (2×4)-way $\rightarrow$ (4×2)-way and (3×4)-way $\rightarrow$ (4×3)-way), 8-way and 12-way TMVPs can be derived. The resulting TMVP-8 algorithm has a complexity of:

$$M(n) = 25M(n/8) + 237n/8 - 54,$$

while the TMVP-12 algorithm has a complexity of:

$$M(n) = 36M(n/12) + 32n - 84.$$

In contrast, our proposed TMVP-8 algorithm exhibits a complexity of:

$$M(n) = 15M(n/8) + 385n/8 - 134,$$

and our TMVP-12 algorithm exhibits a complexity of:

$$M(n) = 23M(n/12) + 319n/4 - 342.$$

Finally, generalizing the method from [12] yields complexities of:

$$M(n) = 36M(n/8) + 49n/2 - 56$$

for $k = 8$, and

$$M(n) = 78M(n/12) + 77n/2 - 132$$

for $k = 12$. Consequently, our 8-way TMVP formula proves superior for $n \geq 59$, and our 12-way TMVP formula is superior for $n \geq 3689$, as demonstrated in Table 1.

According to the solutions of these recursive relations, Table 1 details the comparison results between our k-way TMVP algorithms and existing k-way TMVP algorithms found in the literature. Among the 5-, 6-, 8-, and 12-way TMVP algorithms presented, our novel constructions necessitate the lowest total number of arithmetic operations for $n \geq 125$, $n \geq 254$, $n \geq 59$, and $n \geq 3689$, respectively, as highlighted in the final row of Table 1.

6.2 Comparison of Polynomial Algorithms

For input sizes n of types $n = 2^l$ and $n = 3^l$, we compare the complexities and number of operations for known polynomial multiplication algorithms, including Karatsuba like-3 way, Optimized Karatsuba and Toom Cook, with the best sequence of algorithms defined in Section 5 corresponding to these input sizes. Recall from Theorem 5.3 that the most efficient sequence of algorithms is

k	5	6	8	12
[11]	$\frac{61}{8}n^{\log_5 13} - \frac{65}{8}n + \frac{3}{2}$			
[16]		$\frac{136}{15}n^{\log_6 16} - \frac{47}{5}n + \frac{4}{3}$	$\frac{863}{68}n^{\log_8 25} - \frac{237}{17}n + \frac{9}{4}$	$\frac{73}{5}n^{\log_{12} 36} - 16n + \frac{12}{5}$
[12]		$\frac{13}{2}n^{\log_6 21} - 7n + \frac{3}{2}$	$\frac{32}{5}n^{\log_8 36} - 7n + \frac{8}{5}$	$\frac{44}{7}n^{\log_{12} 78} - 7n + \frac{12}{7}$
This work	$\frac{215}{8}n^{\log_5 9} - 31n + \frac{41}{8}$	$\frac{167}{5}n^{\log_6 11} - 39n + \frac{33}{5}$	$\frac{325}{7}n^{\log_8 15} - 55n + \frac{67}{7}$	$\frac{797}{11}n^{\log_{12} 23} - 87n + \frac{171}{11}$
Better if	$n \geq 125$	$n \geq 254$	$n \geq 59$	$n \geq 3689$

Table 1: Comparison table for the number of operations for different small values of k: Our k-way TMVP algorithms versus existing k-way TMVP algorithms in the literature

$B_2 := K_2 \to K_2 \to \cdots \to K_2 \to S_2 \to S_2$ when the input size is of the form $n = 2^l$, and $B_3 := K_3 \to K_3 \to \cdots \to K_3 \to S_3$ when the input size is of the form $n = 3^l$. Applying these sequences of algorithms to these inputs yields

$$B_2(n = 2^l) = \frac{55}{9}3^l - 8 \cdot 2^l + 2$$

and

$$B_3(n = 3^l) = \frac{87}{15}6^l - 8 \cdot 3^l + \frac{11}{5}.$$

On the other hand, the Karatsuba 3-way, Toom-Cook 3-way and Optimized Karatsuba 3-way algorithms ([14]) have the complexity formulas $M(n) = \frac{34}{5}n^{\log_3 6} - 8n + \frac{11}{5}$, $M(n) = 13n^{\log_3 5} - 15n + 3$ and $M(n) = \frac{98}{15}n^{\log_3 6} - \frac{22}{3}n + \frac{9}{5}$, respectively. Similarly, the Karatsuba 2-way, Toom-Cook 2-way (which is identical to Karatsuba 2-way) and Optimized Karatsuba 2-way algorithms ([14]) have the complexity formulas $M(n) = 7n^{\log_2 3} - 8n + 2$ and $M(n) = \frac{13}{2}n^{\log_2 3} - 7n + \frac{3}{2}$, respectively. According to these formulas, the comparison results are given in Table 2. For inputs of size $n = 2^l$, our best sequence always requires the lowest amount of operations for $n \geq 2$ or $l \geq 1$. For inputs of size $n = 3^l$, the 3-way Toom-Cook split algorithm uses the overall lowest amount of operations for $n \geq 84$, and among the remaining three algorithms for the input set $n = 3^l$, our best sequence is the most operation-saving algorithm for $n \geq 3$ or $l \geq 1$.

	$n = 2^l$	$n = 3^l$
Karatsuba ([15])	$7 \cdot 3^l - 8 \cdot 2^l + 2$	$\frac{34}{5}6^l - 8 \cdot 3^l + \frac{11}{5}$
Optimized Toom-3 / Toom-Cook (Bodrato and Zanoni) ([20, 8, 5])	$7 \cdot 3^l - 8 \cdot 2^l + 2$	$13 \cdot 5^l - 15 \cdot 3^l + 3$
Optimized Karatsuba ([15, 14])	$\frac{13}{2}3^l - 7 \cdot 2^l + \frac{3}{2}$	$\frac{98}{15} \cdot 6^l - \frac{22}{3} \cdot 3^l + \frac{9}{5}$
Proposed Sequence (this work)	$\frac{55}{9}3^l - 8 \cdot 2^l + 2$	$\frac{87}{15}6^l - 8 \cdot 3^l + \frac{11}{5}$
Our work is better than Toom-Cook if	$l \geq 1$	$1 \leq l \leq 4$
Our work is better than Optimized Karatsuba if	$l \geq 1$	$l \geq 1$

Table 2: Comparison of the number of operations where input sizes are $n = 2^l$ and $n = 3^l$: Our best sequence versus algorithms in the literature

Remark 6.1. The Toom-3 operation count is based on an optimized interpolation sequence related to the work of Bodrato and Zanoni ([5]). Their weighted cost model distinguishes additions, shifts, exact divisions, and multiplications by small constants, whereas Table 2 uses a simplified operation-counting convention.

7 Conclusion and Future Work

In this paper, we presented generalized k-way split polynomial multiplication algorithms and a novel k-way split Toeplitz matrix-vector product (TMVP) algorithm. By utilizing optimal evaluation points and interpolations, we proved that a k-way TMVP can be computed using exactly $2k-1$ smaller TMVP subproblems, which corresponds to the theoretical lower bound for such interpolation-based methods. This yields a subquadratic asymptotic time complexity of $O(n^{\log_k(2k-1)})$. We rigorously derived the recurrence relations, closed-form expressions, and exact arithmetic operation counts for these algorithms. Furthermore, we analyzed the consecutive application of k-way Schoolbook and Karatsuba algorithms, mathematically proving the optimal algorithm sequence for any input size n based entirely on its prime factorization.

Our theoretical comparisons with the existing literature demonstrate that the proposed k-way TMVP formulas outperform currently known specific k-way algorithms (such as 5, 6, 8, and 12-way unequal splits) in terms of total arithmetic operations once the input size n exceeds certain explicitly calculated crossover thresholds. Similarly, our optimal algorithmic sequence for polynomial multiplication yields state-of-the-art efficiency for inputs of size $n = 2^l$ and $n = 3^l$.

7.1 Future Directions

The theoretical bounds and algorithmic constructions established in this work open several promising avenues for future research:

- **Hardware and Software Implementations:** While our generalized algorithms provide lower total operation counts asymptotically, the overhead originating from scalar multiplications (such as powers of $2, 3, \ldots, k-1$) and matrix additions requires careful architectural management. Future work should focus on parallelized hardware (FPGA/ASIC) and optimized software implementations to observe the practical performance and measure the empirical crossover points of these algorithms.

- **Applications in Post-Quantum Cryptography (PQC):** Polynomial multiplication over quotient rings and Toeplitz matrix-vector products are fundamental arithmetic bottlenecks in many PQC algorithms, particularly in lattice-based (e.g., ML-KEM/Kyber) and code-based cryptographic schemes. Integrating our optimal k-way algorithms into these cryptographic suites could yield significant performance improvements in key generation, encryption, and decapsulation phases.

- **Generalization to Other Structured Matrices:** The matrix row-reversal techniques and evaluation-interpolation methods utilized here to reduce the TMVP subproblems can potentially be adapted to other dense structured matrices, such as Hankel and Circulant matrices, to construct similarly optimal k-way split algorithms.

References

[1] Carlos Aguilar-Melchor et al. Hamming quasi-cyclic (HQC) algorithm specifications and supporting documentation, 2025. Submission to the NIST Post-Quantum Cryptography Standardization Project.

[2] Gorjan Alagic et al. Status report on the third round of the NIST post-quantum cryptography standardization process. Technical Report NISTIR 8413, US Department of Commerce, NIST, 2022.

[3] Gorjan Alagic et al. Status report on the fourth round of the NIST post-quantum cryptography standardization process. Technical Report NISTIR 8545, US Department of Commerce, NIST, 2024.

[4] Marco Bodrato. Towards optimal Toom-Cook multiplication for univariate and multivariate polynomials in characteristic 2 and 0. In *International Workshop on the Arithmetic of Finite Fields*, pages 116–133. Springer, 2007.

[5] Marco Bodrato and Alberto Zanoni. Integer and polynomial multiplication: towards optimal Toom-Cook matrices. In *International Symposium on Symbolic and Algebraic Computation*. ACM, 2007.

[6] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018.

[7] Murat Cenk, Christophe Negre, and M Anwar Hasan. Improved three-way split formulas for binary polynomial and Toeplitz matrix vector products. *IEEE Transactions on Computers*, 62(7):1345–1361, 2013.

[8] Stephen A Cook. *On the Minimum Computation Time of Functions*. PhD thesis, Harvard University, Cambridge, MA, USA, 1966.

[9] Jan-Pieter D'Anvers, Angshuman Karmakar, Sujoy Sinha Roy, and Frederik Vercauteren. Saber: Module-LWR based key exchange, CPA-secure encryption and CCA-secure KEM. In *International Conference on Cryptology in Africa*, pages 282–305. Springer, 2018.

[10] Haining Fan and M Anwar Hasan. A new approach to subquadratic space complexity parallel multipliers for extended binary fields. *IEEE Transactions on Computers*, 56(2):224–233, 2007.

[11] Anıl Burak Gökçe. A new efficient TMVP algorithm and an application for post-quantum cryptography. 2023.

[12] M. Anwar Hasan and C. Negre. Multiway splitting method for Toeplitz matrix vector product. *IEEE Transactions on Computers*, 62(7):1467–1471, 2012.

[13] Jeffrey Hoffstein, Jill Pipher, and Joseph H Silverman. NTRU: A ring-based public key cryptosystem. In *International Algorithmic Number Theory Symposium*, pages 267–288. Springer, 1998.

[14] Murat Burhan İlter and Murat Cenk. Efficient big integer multiplication in cryptography. *International Journal of Information Security Science*, 6(4):1–16, 2017.

[15] Anatoly Karatsuba and Yuri Ofman. Multiplication of many-digit numbers by automatic computers. *Doklady Akademii Nauk SSSR*, 145(2):293–294, 1962. Translation in Physics-Doklady, 7, 595–596 (1963).

[16] İrem Keskinkurt Paksoy. New TMVP-based multiplication algorithms for polynomial quotient rings and application to post-quantum cryptography. 2022.

[17] İrem Keskinkurt Paksoy and Murat Cenk. TMVP-based multiplication for polynomial quotient rings and application to saber on ARM Cortex-M4. *Cryptology ePrint Archive, Report 2020/1302*, 2020.

[18] İrem Keskinkurt Paksoy and Murat Cenk. Faster NTRU on ARM Cortex-M4 with TMVP-based multiplication. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 69(10):4083–4092, 2022.

[19] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. *Proceedings 35th Annual Symposium on Foundations of Computer Science*, pages 124–134, 1994.

[20] Andrei L Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3:714–716, 1963.

[21] A. Venkatesan and S. V. Kumar. Efficient implementation of fast convolution in ASIP. In *Recent Advances in Space Technology Services and Climate Change 2010 (RSTS and CC-2010)*, pages 203–207. IEEE, 2010.

[22] S. Winograd. *Arithmetic Complexity of Computations*. Society For Industrial and Applied Mathematics, U.S., 1980.