Algorithms for structured matrix-vector product of optimal bilinear complexity

Ke Ye
Computational and Applied Mathematics Initiative
Department of Statistics
University of Chicago
Email: kye@galton.uchicago.edu

Lek-Heng Lim
Computational and Applied Mathematics Initiative
Department of Statistics
University of Chicago
Email: lekheng@galton.uchicago.edu

***Abstract*—We present explicit algorithms for computing structured matrix-vector products that are optimal in the sense of Strassen, i.e., using a provably minimum number of multiplications. These structures include Toeplitz/Hankel/circulant, symmetric, Toeplitz-plus-Hankel, sparse, and multilevel structures. The last category include BTTB, BHHB, BCCB but also any arbitrarily complicated nested structures built out of other structures.**

I. Introduction

Given a bilinear map $\beta : \mathbb{C}^m \times \mathbb{C}^n \to \mathbb{C}^p$, the *bilinear complexity* [9], [10] of β is the least number of multiplications needed to evaluate $\beta(x, y)$ for $x \in \mathbb{C}^m$ and $y \in \mathbb{C}^n$. This notion of bilinear complexity is the standard measure of computational complexity for matrix inversion and matrix multiplication [7], [6], [12], [13].

This article is an addendum to our work in [14] where we proposed a generalization of the *Cohn–Umans method* [3], [4] and used it to study the bilinear complexity of structured matrix-vector product. We did not derive any actual algorithms in [14]. The purpose of this present work is to provide explicit algorithms for structured matrix-vector product obtained by our generalized Cohn–Umans method in [14]. All algorithms in this paper have been shown to be the fastest possible in terms of bilinear complexity. The proofs may be found in [14] and involve determining the *tensor ranks* of these structured matrix-vector products.

Here is a list of structured matrices discussed in this article:

- §II Circulant matrices.
- §III Toeplitz/Hankel matrices.
- §IV Symmetric matrices.
- §V Toeplitz-plus-Hankel matrices.
- §VI Sparse matrices.
- §VII Multilevel structured matrices $A_1 \otimes \cdots \otimes A_p$ where each A_i is one of circulant, Toeplitz/Hankel, symmetric, Toeplitz-plus-Hankel, or sparse matrices.

The algorithms for circulant [5] and Toeplitz [1] matrices are known but those for other structured matrices are new (as far as we know). In particular, the multilevel structured matrices in §VII include arbitrarily complicated nested structures, e.g., block BCCB matrices whose blocks are Toeplitz-plus-Hankel, a 3-level structure.

We analyze the bilinear complexities of all algorithms in §VIII. Readers should bear in mind that bilinear complexity does not count scalar multiplications. For example, the bilinear map $\beta : \mathbb{C}^2 \times \mathbb{C}^2 \to \mathbb{C}$, $\beta((a, b), (c, d)) = (2a+b)(3c-d)$ has bilinear complexity one. For those familiar with tensor rank [8], the bilinear complexity of β is just the tensor rank of the *structure tensor* $\mu_\beta \in \mathbb{C}^2 \otimes \mathbb{C}^2 \otimes \mathbb{C}^2$ corresponding to β [2], [14].

II. Circulant matrix

An $n \times n$ circulant matrix $A = (a_{ij})$ is a matrix with

$$a_{ij} = a_{i+p,j+p}, \quad 1 \le i, j, i+p, j+p \le n,$$
$$a_{1j} = a_{n+2-j,1}, \quad 2 \le j \le n.$$

The circulant matrix represented by $a = (a_1, \dots, a_n) \in \mathbb{C}^n$ is one whose first row is a. It is well-known [5] that the circulant matrix-vector product can be computed by Fourier transform. We restate this algorithm for completeness. Let $\omega_k = e^{2k\pi i/n}$, $k = 0, \dots, n-1$ and define the Fourier matrix

$$W = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & \omega_1 & \cdots & \omega_{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \omega_1^{n-1} & \cdots & \omega_{n-1}^{n-1} \end{bmatrix}. \tag{1}$$

Algorithm 1 Circulant matrix-vector product

1: Represent the circulant matrix A by $a = (a_1, a_2, \dots, a_n)^\mathsf{T}$ and the column vector by $v = (v_1, v_2, \dots, v_n)^\mathsf{T}$.
2: Compute Wa and represent it by $(\tilde{a}_1, \dots, \tilde{a}_n)^\mathsf{T}$.
3: Compute $nW^{-1}v$ and represent it by $(\tilde{v}_1, \dots, \tilde{v}_n)^\mathsf{T}$.
4: Compute $\tilde{z} = (\tilde{a}_1\tilde{v}_1, \dots, \tilde{a}_n\tilde{v}_n)^\mathsf{T}$.
5: Compute $z = W\tilde{z}$, which is the product of A and v.

III. Toeplitz/Hankel matrix

An $n \times n$ Toeplitz matrix $A = (a_{ij})$ is a matrix with

$$a_{ij} = a_{i+p,j+p}, \quad 1 \le i, j, i+p, j+p \le n.$$

We represent an $n \times n$ Toeplitz matrix $A = (a_{ij})$ by $(a_1, a_2, \dots, a_{2n-1}) \in \mathbb{C}^{2n-1}$

$$a_{ij} = a_{j-i+n}.$$

Every $n \times n$ Toeplitz matrix A may be regarded as a block of some $2n \times 2n$ circulant matrix C whose first row is $(a_n, \dots, a_{2n-1}, b, a_1, \dots, a_{n-1})$ and $b \in \mathbb{C}$ is arbitrary. Using this embedding, we obtain Algorithm 2 for Toeplitz matrix-vector product [1], [14] .

Algorithm 2 Toeplitz matrix-vector product

Express the Toeplitz matrix A as $(a_1, \dots, a_{2n-1})$ and the vector as $v = (v_1, \dots, v_n)^{\mathsf{T}}$.
Compute $b = -\sum_{i=1}^{2n-1} a_i$.
Construct $c = (a_n, \dots, a_{2n-1}, b, a_1, \dots, a_{n-1}) \in \mathbb{C}^{2n}$.
Construct $\tilde{v} = (v_1, \dots, v_n, 0, \dots, 0)^{\mathsf{T}} \in \mathbb{C}^{2n}$.
Compute the product $\tilde{z} = (z_1, \dots, z_{2n})^{\mathsf{T}}$ of the circulant matrix determined by c with $\tilde{v}$ by Algorithm 1.
$z = (z_1, \dots, z_n)^{\mathsf{T}}$ is the product of A and v.

An $n \times n$ matrix H is called a Hankel matrix if JH is a Toeplitz matrix where

$$J = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 \end{bmatrix}. \tag{2}$$

We represent an $n \times n$ Hankel matrix $H = (h_{ij})$ as $(h_1, h_2, \dots, h_{2n-1}) \in \mathbb{C}^{2n-1}$ where

$$h_{ij} = h_{2n+1-i-j}, \quad 1 \leq i, j \leq n.$$

Algorithm 3 computes the product of a Hankel matrix and a column vector v.

Algorithm 3 Hankel matrix-vector product

Express $T = (h_1, h_2, \dots, h_{2n-1})$.
Apply Algorithm 2 to the Toeplitz matrix represented by T and v to obtain $(z_1, \dots, z_n)$.
$(z_n, z_{n-1}, \dots, z_1)$ is the product of H and v.

IV. Symmetric matrix

Algorithm 4 computes the product of a symmetric matrix $S = (s_{ij})$ where $s_{ij} = s_{ji}$ and a column vector v. We represent a symmetric matrix $s = (s_{ij})$ as $(s_1, \dots, s_N) \in \mathbb{C}^N$ where $N = \binom{n+1}{2}$ and the index of s_k is

$$k = (i-1)n - \binom{i-1}{2} + j, \quad 1 \leq i \leq j \leq n.$$

V. Toeplitz-plus-Hankel matrix

An $n \times n$ Toeplitz-plus-Hankel matrix is a matrix which can be expressed as the sum of an $n \times n$ Hankel matrix and an $n \times n$ Toeplitz matrix. If X is an $n \times n$ Toeplitz-plus-Hankel matrix and

$$X = H + T$$

Algorithm 4 Symmetric matrix-vector product

S is an $n \times n$ symmetric matrix. Set $S_1 = S$. Set $m = \lceil n/2 \rceil$. Set $v_1 = v$ and $z = 0$.
for $k = 1, \dots, m$ **do**
  Construct Hankel matrix H_k determined by first row and last column of S_k.
  Compute $w_k = H_k v_k$ by Algorithm 3.
  Update $z = z + w_k$.
  Construct S_{k+1} by deleting first and last columns and first and last rows of $S_k - H_k$.
  Construct v_{k+1} by deleting first and last entry of v_k.
end for
$z = (z_1, \dots, z_n)^{\mathsf{T}}$ is the product of S and v.

for some Hankel matrix H and some Toeplitz matrix T, then for any $a \in \mathbb{C}$ we have a decomposition of X into the sum of a Hankel matrix $H + aE$ and a Toeplitz matrix $T - aE$ where E is the $n \times n$ matrix with all entries equal to one.

Algorithm 5 Toeplitz-plus-Hankel matrix-vector product

Express X as $H + T$ with Hankel matrix H and Toeplitz matrix T.
Express T as $(t_1, \dots, t_{2n-1})$ and H as $(h_1, \dots, h_{2n-1})$.
Compute $b = -\sum_{j=1}^{2n-1} t_j$.
Compute $a \in \mathbb{C}$ as

$$a = \frac{\sum_{j=0}^{n-1} \omega^j t_{n+j} + \omega^n b + \sum_{j=1}^{n-1} \omega^{n+j} t_j}{2n}$$

where $\omega = e^{k\pi i/n}$.
Update $H = H + aE$ and $T = T - aE$.
Compute $z_H = Hv$ by Algorithm 3 and $z_T = Tv$ by Algorithm 2, respectively.
Compute $z = z_H + z_T$, which is the product of X and v.

VI. Sparse matrix

An $n \times n$ sparse matrix $A = (a_{ij})$ with sparsity pattern $\Omega \subseteq \{1, \dots, n\} \times \{1, \dots, n\}$ is one where

$$a_{ij} = 0 \quad \text{for all } (i, j) \in \Omega.$$

For example, an upper triangular matrix is a sparse matrix with sparsity pattern $\Omega = \{(i, j) : 1 \leq i \leq j \leq n\}$. For sparse matrices associated with Ω, the matrix-vector product has optimal bilinear complexity $\#\Omega$ realized by the usual matrix-vector product algorithm [14].

VII. Multilevel structured matrix

Let $A = (a_{ij}) \in \mathbb{C}^{n \times n}$ and $B = (b_{ij}) \in \mathbb{C}^{m \times m}$. The Kronecker product [11] of A and B is defined as

$$A \circledast B = (a_{ij} B) \in \mathbb{C}^{mn \times mn},$$

i.e., $A \circledast B$ is an $m \times m$ block matrix whose (i, j)th block is the $n \times n$ matrix $a_{ij} B$. We may iterate the definition to obtain a p levels matrix $A = A_1 \circledast \cdots \circledast A_p$. In particular, if $A_1, \dots, A_p$ are structured matrices (circulant, Toeplitz, Hankel, symmetric and Toeplitz-plus-Hankel), then A is called a p levels structured matrix.

Let $X_1 \subseteq \mathbb{C}^{n_1 \times n_1}, \dots, X_p \subseteq \mathbb{C}^{n_p \times n_p}$ be subspaces of structured matrices. Then $X_1 \circledast \dots \circledast X_p \subseteq \mathbb{C}^{n_1 \cdots n_p \times n_1 \cdots n_p}$ is the set of all p levels structured matrices $A_1 \circledast \dots \circledast A_p$ where $A_1 \in X_1, \dots, A_p \in X_p$.

Algorithms 6–11 are based on the following idea. Let $\beta_i : X_i \times \mathbb{C}^{n_i} \to \mathbb{C}^{n_i}$ be the bilinear map defined by the matrix-vector product for matrices in X_i. Assume that the bilinear complexity of β_i is r_i. Then the structural tensor [14] $\mu_{\beta_i} \in X_i^* \otimes (\mathbb{C}^{n_i})^* \otimes \mathbb{C}^{n_i}$ of β_i has a tensor decomposition

$$\mu_{\beta_i} = \sum_{j=1}^{r_i} u_j \otimes v_j \otimes w_j.$$

The bilinear map $\beta : (X_1 \circledast \dots \circledast X_p) \times \mathbb{C}^{n_1 \cdots n_p} \to \mathbb{C}^{n_1 \cdots n_p}$, defined by the p levels structured matrix-vector product, has structural tensor $\mu_\beta = \mu_{\beta_1} \otimes \dots \otimes \mu_{\beta_p}$. In [14] we showed that if X_i is Toeplitz, Hankel, symmetric, or Toeplitz-plus-Hankel, the bilinear complexity is equal to the dimension of X_i and we obtain a machinery to decompose μ_{β_i} explicitly. Essentially, Algorithms 6–11 are obtained from the tensor decompositions of structural tensors.

A. Illustrative example

As an example, let us consider the case where $p = 2$ and A, B are 2×2 circulant matrices. This gives a block-circulant-circulant-block or BCCB matrix. We set

$$A = \begin{bmatrix} a & b \\ b & a \end{bmatrix}, \quad B = \begin{bmatrix} c & d \\ d & c \end{bmatrix},$$

and

$$v = (x, y, z, w)^{\mathsf{T}} = \begin{bmatrix} x \\ y \end{bmatrix} \circledast \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} z \\ w \end{bmatrix} \circledast \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

We want to compute the product of $A \circledast B$ and v. By definition we have

$$A \circledast B = \begin{bmatrix} aB & bB \\ bB & aB \end{bmatrix} = \begin{bmatrix} ac & ad & bc & bd \\ ad & ac & bd & bc \\ bc & bd & ac & ad \\ bd & bc & ad & ac \end{bmatrix},$$

and

$$(A \circledast B)v = \begin{bmatrix} a(\xi_1 + \xi_2) + b(\eta_1 + \eta_2) \\ a(\xi_1 - \xi_2) + b(\eta_1 - \eta_2) \\ b(\xi_1 + \xi_2) + a(\eta_1 + \eta_2) \\ b(\xi_1 - \xi_2) + a(\eta_1 - \eta_2) \end{bmatrix},$$

where

$$\begin{aligned} \xi_1 &= \frac{1}{2}((cx + dy) + (dx + cy)), \\ \xi_2 &= \frac{1}{2}((cx + dy) - (dx + cy)), \\ \eta_1 &= \frac{1}{2}((cz + dw) + (dz + cw)), \\ \eta_2 &= \frac{1}{2}((cz + dw) - (dz + cw)). \end{aligned}$$

Observe that

$$\begin{bmatrix} a(\xi_1 + \xi_2) + b(\eta_1 + \eta_2) \\ b(\xi_1 + \xi_2) + a(\eta_1 + \eta_2) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \alpha + \beta \\ \alpha - \beta \end{bmatrix},$$

where

$$\begin{aligned} \alpha &= (a + b)[(\xi_1 + \xi_2) + (\eta_1 + \eta_2)] \\ &= (a + b)[(\xi_1 + \eta_1) + (\xi_2 + \eta_2)], \\ \beta &= (a - b)[(\xi_1 + \xi_2) - (\eta_1 + \eta_2)] \\ &= (a - b)[(\xi_1 - \eta_1) + (\xi_2 - \eta_2)]. \end{aligned}$$

Similarly, we have

$$\begin{bmatrix} a(\xi_1 - \xi_2) + b(\eta_1 - \eta_2) \\ b(\xi_1 - \xi_2) + a(\eta_1 - \eta_2) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \gamma + \tau \\ \gamma - \tau \end{bmatrix},$$

where

$$\begin{aligned} \gamma &= (a + b)[(\xi_1 - \xi_2) + (\eta_1 - \eta_2)] \\ &= (a + b)[(\xi_1 + \eta_1) - (\xi_2 + \eta_2)], \\ \tau &= (a - b)[(\xi_1 - \xi_2) - (\eta_1 - \eta_2)] \\ &= (a - b)[(\xi_1 - \eta_1) - (\xi_2 - \eta_2)]. \end{aligned}$$

Lastly, we observe that

$$\begin{aligned} \xi_1 + \eta_1 &= \frac{1}{2}(c + d)[(x + y) + (z + w)], \\ \xi_1 - \eta_1 &= \frac{1}{2}(c + d)[(x + y) - (z + w)], \\ \xi_2 + \eta_2 &= \frac{1}{2}(c - d)[(x - y) + (z - w)], \\ \xi_2 - \eta_2 &= \frac{1}{2}(c - d)[(x - y) - (z - w)]. \end{aligned}$$

By above computations, we see that one may compute $(A \circledast B)v$ using four multiplications, i.e., it is sufficient to compute

$$\begin{aligned} w_{11} &= (a + b)(c + d)[(x + y) + (z + w)], \\ w_{12} &= (a + b)(c - d)[(x - y) + (z - w)], \\ w_{21} &= (a - b)(c + d)[(x + y) - (z + w)], \\ w_{22} &= (a - b)(c - d)[(x - y) - (z - w)]. \end{aligned}$$

Note that since the entries of $A \circledast B$ are given as inputs, evaluating terms like $(a + b)(c + d) = ac + ad + bc + bd$ does not cost any multiplication (as we already have ac, ad, bc, bd as inputs).

B. General case

We now generalize the above calculations to obtain an algorithm for p levels structured matrix-vector product. In order to treat all cases at one go, our presentation in this section is slightly more abstract. Given a p levels structured matrix $B \in X_1 \circledast \dots \circledast X_p$ and a vector v of appropriate size, our algorithm, when applied to B and v, takes the form:

$$(B, v) \xrightarrow{(\varphi, \psi)} (b', v') \xrightarrow{m} m(b', v') \xrightarrow{\vartheta} Bv,$$

where φ is a linear map sending B to a vector b', ψ is a linear map sending v to a vector v', m is pointwise multiplication, and ϑ is another linear map sending $m(b', v')$ to Bv. φ, ψ, and ϑ depend only on the structure of B (i.e., on $X_1, \ldots, X_p$) but not on the values of B and v. For any given structure, we can represent the linear maps φ, ψ, and ϑ concretely as matrices.

We will present the algorithms for p levels structured matrix-vector product *inductively*, by calling the corresponding $p-1$ levels algorithms. Also, they will be built upon Algorithms 2, 3, 4, and 5 for the relevant structured matrix-vector product.

Suppose we have algorithms for $p-1$ levels structured matrix-vector product, i.e., we may evaluate the linear maps φ, ψ, and ϑ for any $p-1$ levels structured matrix. Given a p levels structured matrix $A_1 \circledast \cdots \circledast A_p$ and a column vector v of size $N = \prod_{i=1}^{p} n_i$, we write $A_1 \circledast \cdots \circledast A_p$ as $A \circledast B$ where $A = A_1$ and $B = A_2 \circledast \cdots \circledast A_p$. Set N_1 to be N/n_1.

Let A be a circulant matrix. Let $\omega_k = e^{2k\pi i/n}$, $k = 0, 1, \ldots, n-1$ be the nth roots of unity and let $W = (\omega_k^j)_{j,k=0}^{n-1}$ be the Fourier matrix in (1). We have Algorithm 6.

Algorithm 6 p levels circulant matrix-vector product

1: Express A by a column vector $a = (a_1, \ldots, a_{n_1})^\mathsf{T}$ and express v by a column vector
$$v = (v_{1,1}, \ldots, v_{1,N_1}, v_{2,1}, \ldots, v_{2,N_1}, \ldots, v_{n_1,1}, \ldots, v_{n_1,N_1})^\mathsf{T}.$$
2: Express φ as $(\varphi_1, \ldots, \varphi_r)^\mathsf{T}$ where φ_j is a linear functional on $X_2 \circledast \cdots \circledast X_p$ and $r = \prod_{i=2}^{p} \dim(X_i)$.
3: Express ψ as $(\psi_1, \ldots, \psi_r)$ where ψ_j is a linear functional on $\mathbb{C}^{N_1}$.
4: Compute $\tilde{a} = Wa$ and denote it by $(\tilde{a}_1, \ldots, \tilde{a}_{n_1})^\mathsf{T}$.
5: Denote $v_i = (v_{i,1}, \ldots, v_{i,N_1})^\mathsf{T}, i = 1, \ldots, n_1$.
6: **for** $s = 1, \ldots, n_1$ **do**
7:     **for** $t = 1, \ldots, r$ **do**
8:         Compute
$$w_{st} = \tilde{a}_s \varphi_t(B) \sum_{k=1}^{n_1} \omega_{k-1}^{s-1} \psi_t(v_k).$$
9:     **end for**
10: **end for**
11: Represent (w_{st}) as a column vector
$$w = (w_{11}, \ldots, w_{1,r}, w_{21}, \ldots, w_{2,r}, \ldots, w_{n_1,1}, \ldots, w_{n_1,r})^\mathsf{T}.$$
12: Compute $(W \circledast \vartheta)w$, which is the product $(A \circledast B)v$.

If we apply Algorithm 6 to the case where A, B are 2×2 circulant matrices, we obtain $w_{11}, w_{12}, w_{21}, w_{22}$ as in Section VII-A. To compute the product of $A \circledast B$ and v, we express A as $(a, b)^\mathsf{T}$, B as $(c, d)^\mathsf{T}$, and v as $(x, y, z, w)^\mathsf{T}$. Hence $v_1 = (x, y)^\mathsf{T}$ and $v_2 = (z, w)^\mathsf{T}$. By Algorithm 1 the linear map $\varphi = (\varphi_1, \varphi_2)^\mathsf{T}$ is given by $\varphi_1((\alpha, \beta)^\mathsf{T}) = \alpha + \beta$ and $\varphi_2((\alpha, \beta)^\mathsf{T}) = \alpha - \beta$, and ψ is the map given by $\psi_1((\alpha, \beta)^\mathsf{T}) = \alpha + \beta$ and $\psi_2((\alpha, \beta)^\mathsf{T}) = \alpha - \beta$, where $(\alpha, \beta)^\mathsf{T}$ is any column vector of size two. Lastly, the linear map ϑ is given by left multiplication by $\left[\begin{smallmatrix} 1 & 1 \\ 1 & -1 \end{smallmatrix}\right]$.

Let A be a Toeplitz matrix. As before, there exists a circulant matrix C of the form
$$C = \begin{bmatrix} A & A' \\ A' & A \end{bmatrix},$$
and
$$(C \circledast B) \begin{bmatrix} v \\ 0 \end{bmatrix} = \begin{bmatrix} (A \circledast B)v \\ (A' \circledast B)v \end{bmatrix}.$$
Hence to compute $(A \circledast B)v$, it suffices to compute $(C \circledast B)\left[\begin{smallmatrix} v \\ 0 \end{smallmatrix}\right]$ and this can be done using Algorithm 6. We obtain Algorithm 7.

Algorithm 7 p levels Toeplitz matrix-vector product

1: Express A as a vector $a = (a_1, \ldots, a_{2n-1})$ and v as $(v_1, \ldots, v_N)^\mathsf{T}$.
2: Compute $b = -\sum_{i=1}^{2n_1-1} a_i$.
3: Construct $c = (a_n, \ldots, a_{2n_1-1}, b, a_1, \ldots, a_{n_1-1}) \in \mathbb{C}^{2n_1}$ representing a $2n_1 \times 2n_1$ circulant matrix C.
4: Construct $\tilde{v} = (v_1, \ldots, v_N, 0, \ldots, 0) \in \mathbb{C}^{2N}$.
5: Compute $\tilde{z} = (C \circledast B)\tilde{v}$ by Algorithm 6 and express $\tilde{z}$ as $(z_1, \ldots, z_{2N})^\mathsf{T}$.
6: $(z_1, \ldots, z_N)^\mathsf{T}$ is the product of $(A \circledast B)$ and v.

Now for square Hankel matrices A and B we observe that
$$JA \otimes B = (J \otimes I)(A \otimes B),$$
where J is the matrix in (2). Algorithm 8 follows.

Algorithm 8 p levels Hankel matrix-vector product

1: Compute the $Z = (JA \otimes B)v$ by Algorithm 7.
2: Compute $z = (J \otimes I)Z$ and z is $(A \otimes B)v$.

The algorithms for p levels symmetric matrix (Algorithm 9), p levels Toeplitz-plus-Hankel matrix (Algorithm 10), p levels sparse matrix (Algorithm 11) are obtained via similar considerations.

Algorithm 9 p levels symmetric matrix-vector product

1: A is an $n_1 \times n_1$ symmetric matrix. Compute $m = \lceil n_1/2 \rceil$. Set $v_1 = v$ and $z = 0 \in \mathbb{C}^N$.
2: **for** $k = 1, \ldots, m$ **do**
3:     Construct H_k determined by first row and last column of A_k.
4:     Compute $w_k = (H_k \circledast B)v_k$ by Algorithm 8.
5:     Update $z = z + w_k$.
6:     Construct A_{k+1} by deleting first and last columns and first and last rows of $A_k - H_k$.
7:     Construct v_{k+1} by deleting first N_1 and last N_1 entries of v_k.
8: **end for**
9: $z = (z_1, \ldots, z_N)^\mathsf{T}$ is the product of S and v.

Algorithm 10 p levels Toeplitz-plus-Hankel matrix-vector product

Express A as $H + T$ with Hankel matrix H and Toeplitz matrix T.
Express T as $(t_1, \dots, t_{2n_1-1})$ and H as $(h_1, \dots, h_{2n_1-1})$.
Compute $b = -\sum_{j=1}^{2n_1-1} t_j$.
Find $a \in \mathbb{C}$ such that

$$a = \frac{\sum_{j=0}^{n_1-1} \omega_1^j t_{n_1+j} + \omega_1^{n_1} b + \sum_{j=1}^{n_1-1} \omega_1^{n_1+j} t_j}{2n_1}$$

where $\omega_1 = e^{k\pi i/n_1}$.
Update $H = H + aE$ and $T = T - aE$.
Compute $z_H = (H \circledast B)v$ by Algorithm 8 and $z_T = (T \circledast B)v$ by Algorithm 7, respectively.
Compute $z = z_H + z_T$ which is the product of A and v.

Algorithm 11 p levels sparse matrix-vector product

Express A by its entries $A = (a_{ij})$.
Express v by a column vector

$$v = (v_{1,1}, \dots, v_{1,N_1}, v_{2,1}, \dots, v_{2,N_1}, \dots, v_{n_1,1}, \dots, v_{n_1,N_1})^\mathsf{T}.$$

Express φ as $(\varphi_1, \dots, \varphi_r)^\mathsf{T}$ where φ_j is a linear functional on $X_2 \circledast \cdots \circledast X_p$ and $r = \prod_{i=2}^p \dim(X_i)$.
Express ψ as $(\psi_1, \dots, \psi_r)$ where ψ_j is a linear functional on $\mathbb{C}^{N_1}$.
Denote $v_i = (v_{i1}, \dots, v_{i,N_1})^\mathsf{T}, i = 1, \dots, n_1$.
for $s = 1, \dots, n_1$ **do**
    for $t = 1, \dots, r$ **do**
        Compute

$$w_{s,t} = \varphi_t(B) \sum_{(k,s) \notin \Omega} a_{ks} \psi_t(v_k)$$

    end for
end for
Compute

$$(z_{ij}) = (I \circledast \vartheta)(w_{st}).$$

$(z_{1,1}, \dots, z_{1,n_1}, z_{2,1}, \dots, z_{2,n_1}, \dots, z_{N_1,1}, \dots, z_{N_1,n_1})^\mathsf{T}$ is $(A \circledast B)v$.

VIII. Bilinear complexity

As we have shown in [14], all 11 algorithms presented in this article are of optimal bilinear complexity, i.e., requires a minimum number of multiplications. We give the multiplication counts below.

(i) Algorithm 1 for $n \times n$ circulant matrix-vector product costs n multiplications (from the computation of $\tilde{z}$; note that the other multiplications in the algorithm are scalar multiplications and do not count towards bilinear complexity).
(ii) Algorithms 2 and 3 for $n \times n$ Toeplitz/Hankel matrix-vector products each costs $2n - 1$ multiplications (from the computation of $\tilde{z}$; by our special choice of b we saved one multiplication).
(iii) Algorithm 4 for $n \times n$ symmetric matrix-vector product costs $\binom{n+1}{2}$ multiplications (each w_k costs $2[n - 2(k-1)] - 1$ multiplications and so the total number of multiplications is $\binom{n+1}{2}$).
(iv) An $N \times N$ p levels structured matrix-vector product costs $\prod_{i=1}^p \dim X_i$ multiplications. Let $r = \prod_{i=2}^p \dim(X_i)$.
(v) Algorithm 6 costs $n_1 r$ multiplications (each w_{st} costs one multiplication; note that computation of the coefficient $\tilde{a}_s \varphi_t(B)$ does not cost any multiplication as $\tilde{a}_s \varphi_t(B)$ is a linear combination of the entries of $A \circledast B$).
(vi) Algorithms 7 and Algorithm 8 each costs $(2n_1 - 1)r$ multiplications.
(vii) Algorithm 9 costs $\binom{n+1}{2} r$ multiplications.
(viii) Algorithm 10 costs $(4n - 3)r$ multiplications.
(ix) Algorithm 11 costs $\#\Omega \times r$ multiplications.

Acknowledgment

The authors thank Nikos Pitsianis and Xiaobai Sun for helpful discussions. LHL and KY are supported by AFOSR FA9550-13-1-0133, DARPA D15AP00109, NSF IIS 1546413, DMS 1209136, and DMS 1057064. In addition, KY's work is also partially supported by NSF CCF 1017760.

References

[1] D. Bini and M. Capovani, "Tensor rank and border rank of band Toeplitz matrices", *SIAM J. Comput.*, **16** (1987), no. 2, pp. 252-258.
[2] P. Bürgisser, M. Clausen, and M. A. Shokrollahi, *Algebraic Complexity Theory*, Grundlehren der Mathematischen Wissenschaften, **315**, Springer-Verlag, Berlin, 1997.
[3] H. Cohn, R. Kleinberg, B. Szegedy, and C. Umans, "Group-theoretic algorithms for matrix multiplication," *Proc. IEEE Symp. Found. Comput. Sci.* (FOCS), **46** (2005), pp. 379–388.
[4] H. Cohn and C. Umans, "A group-theoretic approach to fast matrix multiplication," , *Proc. IEEE Symp. Found. Comput. Sci.* (FOCS), **44** (2003), pp. 438–449.
[5] G. Golub and C. Van Loan, *Matrix Computations*, 4th Ed., John Hopkins University Press, Baltimore, MD, 2013.
[6] F. Le Gall, "Powers of tensors and fast matrix multiplication," *Proc. Internat. Symp. Symbolic Algebr. Comput.* (ISSAC), **39** (2014), pp. 296–303.
[7] V. Strassen, "Gaussian elimination is not optimal," *Numer. Math.*, **13** (1969), pp. 354–356.
[8] V. Strassen, "Rank and optimal computation of generic tensors," *Linear Algebra Appl.*, **52/53** (1983), pp. 645–685.
[9] V. Strassen, "Relative bilinear complexity and matrix multiplication," *J. Reine Angew. Math.*, **375/376** (1987), pp. 406–443.
[10] V. Strassen, "Vermeidung von Divisionen," *J. Reine Angew. Math.*, **264** (1973), pp. 184–202.
[11] C. F. Van Loan, "The ubiquitous Kronecker product," *J. Comput. Appl. Math.*, **123** (2000), no. 1–2, pp. 85–100.
[12] V. V. Williams, "Multiplying matrices in $O(n^{2.373})$ time," *preprint*, (2014), http://theory.stanford.edu/~virgi/matrixmult-f.pdf.
[13] S. Winograd, "Some bilinear forms whose multiplicative complexity depends on the field of constants," *Math. Systems Theory*, **10** (1976/77), no. 2, pp. 169–180.
[14] K. Ye and L.-H. Lim, "Fast structured matrix computations: tensor rank and Cohn–Umans method", *preprint*, (2015), http://arxiv.org/abs/1601.00292.