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 \( \beta : \mathbb{C}^m \times \mathbb{C}^n \rightarrow \mathbb{C}^p \), the bilinear complexity [9], [10] of \( \beta \) 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.

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 matrix-vector product can be computed by Fourier transform. The circulant matrix represented by \( a \) is one of circulant, Toeplitz/Hankel, symmetric, Toeplitz-plus-Hankel, or sparse matrices.

Algorithm 1: Circulant matrix-vector product

1. Represent the circulant matrix \( A = (a_{ij}) \) by \( \omega = (a_1, a_2, \ldots, a_n)^\top \) and the column vector by \( v = (v_1, v_2, \ldots, v_n)^\top \).
2. Compute \( W a \) and represent it by \( \tilde{a} = (\tilde{a}_1, \tilde{a}_2, \ldots, \tilde{a}_n)^\top \).
3. Compute \( nW^{-1} v \) and represent it by \( \tilde{v} = (\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_n)^\top \).
4. Compute \( \tilde{z} = (\tilde{a}_1 \tilde{v}_1, \ldots, \tilde{a}_n \tilde{v}_n)^\top \).
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
\[ a_{ij} = a_{i+j-p}, \quad 1 \leq i, j, i+p, j+p \leq n. \]

We represent an \( n \times n \) Toeplitz matrix \( A = (a_{ij}) \) by \( (a_1, a_2, \ldots, a_{2n-1}) \in \mathbb{C}^{2n-1} \)
\[ a_{ij} = a_{j-i+n}. \]
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 $(a_n, \ldots, a_{2n-1}, b, a_1, \ldots, a_{n-1})$ 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
1: Express the Toeplitz matrix $A$ as $(a_1, \ldots, a_{2n-1})$ and the vector as $v = (v_1, \ldots, v_n)^T$.
2: Compute $b = -\sum_{i=1}^{2n-1} a_i$.
3: Construct $c = (a_{2n}, \ldots, a_{2n-1}, b, a_1, \ldots, a_{n-1}) \in \mathbb{C}^{2n}$.
4: Compute $\tilde{v} = (v_1, \ldots, v_n, 0, \ldots, 0)^T \in \mathbb{C}^{2n}$.
5: Compute the product $\tilde{z} = (z_1, \ldots, z_{2n})^T$ of the circulant matrix determined by $c$ with $\tilde{v}$ by Algorithm 1.
6: $z = (z_1, \ldots, z_n)^T$ is the product of $A$ and $v$.

An $n \times n$ matrix $H$ is called a Hankel matrix if $H_{ij} = h_{i+j}$ for some Hankel matrix $H$. We represent a symmetric matrix $S = (s_{ij})$ as $(s_1, \ldots, s_N) \in \mathbb{C}^N$ where $N = \binom{n+1}{2}$ and the index of $s_k$ is

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

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

Algorithm 3 Hankel matrix-vector product
1: Express $T$ as $(t_1, \ldots, t_{2n-1})$.
2: Apply Algorithm 2 to the Toeplitz matrix represented by $T$ and $v$ to obtain $(z_1, \ldots, z_n)$.
3: $(z_n, z_{n-1}, \ldots, z_1)$ 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 $(s_1, \ldots, s_N) \in \mathbb{C}^N$ where $N = \binom{n+1}{2}$ and the index of $s_k$ is

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

Algorithm 4 Symmetric matrix-vector product
1: $S$ is an $n \times n$ symmetric matrix. Set $S_1 = S$. Set $m = \lceil n/2 \rceil$. Set $v_1 = v$ and $z = 0$.
2: for $k = 1, \ldots, m$
do
3: Construct Hankel matrix $H_k$ determined by first row and last column of $S_k$.
4: Compute $w_k = H_k v_k$ by Algorithm 3.
5: Update $z = z + w_k$.
6: Construct $S_{k+1}$ by deleting first and last columns and first and last rows of $S_k - H_k$.
7: Construct $v_{k+1}$ by deleting first and last entry of $v_k$.
8: end for
9: $z = (z_1, \ldots, z_n)^T$ 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 $H + a E$ and a Toeplitz matrix $T - a E$ where $E$ is the $n \times n$ matrix with all entries equal to one.

Algorithm 5 Toeplitz-plus-Hankel matrix-vector product
1: Express $X$ as $H + T$ with Hankel matrix $H$ and Toeplitz matrix $T$.
2: Express $T$ as $(t_1, \ldots, t_{2n-1})$ and $H$ as $(h_1, \ldots, h_{2n-1})$.
3: Compute $b = -\sum_{j=1}^{2n-1} t_j$.
4: 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 $\omega = e^{k\pi i/n}$.
5: Update $H = H + a E$ and $T = T - a E$.
6: Compute $z_H = Hv$ by Algorithm 3 and $z_T = Tv$ by Algorithm 2, respectively.
7: 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 $\Omega \subseteq \{1, \ldots, n\} \times \{1, \ldots, n\}$ 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 $\Omega$, 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

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

i.e., $A \otimes B$ 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 $A = A_1 \otimes \cdots \otimes A_p$. In particular, if $A_1, \ldots, A_p$ are structured matrices (circulant, Toeplitz,
where

The bilinear map \( X \) defined by the Algorithms 6–11 are obtained from the tensor decompositions

\[\text{BCCB, circulant-block or Hankel, symmetric and Toeplitz-plus-Hankel},\]

Then

\[\mathcal{A} = (a \xi_1 + \xi_2) + b(\eta_1 + \eta_2) \]

\[b(\xi_1 + \xi_2) + a(\eta_1 + \eta_2) = \frac{1}{2} (\alpha + \beta),\]

Define

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

The bilinear map \( \beta: (X_1 \times \cdots \times X_p) \times C^{n_1 \cdots n_p} \rightarrow C^{n_1 \cdots n_p} \),

defined by the \( p \) levels structured matrix-vector product, has structural tensor \( \mu_{\beta_2} \). 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 \( \mu_{\beta_i} \) 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 \( 2 \times 2 \) 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

\[v = (x, y, z, w)^T = \begin{bmatrix} x \\ y \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} z \\ w \end{bmatrix} \oplus \begin{bmatrix} 0 \\ 1 \end{bmatrix} .\]

We want to compute the product of \( A \oplus B \) and \( v \). By definition we have

\[A \oplus B = \begin{bmatrix} ab & BD \\ BD & aB \end{bmatrix} = \begin{bmatrix} ac & ad & bc & bd \\ ad & ac & bd & bc \\ bc & bd & ac & ad \\ bd & bc & ad & ac \end{bmatrix},\]

and

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

where

\[\xi_1 = \frac{1}{2} ((cx + dy) + (dx + cy)),\]

\[\xi_2 = \frac{1}{2} ((cx + dy) - (dx + cy)),\]

\[\eta_1 = \frac{1}{2} ((cz + dw) + (dz + cw)),\]

\[\eta_2 = \frac{1}{2} ((cz + dw) - (dz + cw)).\]
and \( \vartheta \) is another linear map sending \( m(b', v') \) to \( Bv, \varphi, \psi \), and \( \vartheta \) 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 \( \varphi, \psi \), and \( \vartheta \) concretely as matrices.

We will present the algorithms for \( p \) levels structured matrix-vector product \textit{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 \( \varphi, \psi \), and \( \vartheta \) for any \( p - 1 \) levels structured matrix. Given a \( p \) levels structured matrix \( A_1 \otimes \cdots \otimes A_p \) and a column vector \( v \) of size \( N = \prod_{i=1}^{p} n_i \), we write \( A_1 \otimes \cdots \otimes A_p \) as \( A \otimes B \) where \( A = A_1 \) and \( B = A_2 \otimes \cdots \otimes A_p \). Set \( N_1 \) to be \( N/n_1 \).

Let \( A \) be a circulant matrix. Let \( \omega_k = e^{2\pi i n/k} \), \( k = 0, 1, \ldots, n-1 \) be the \( n \)th 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 \( a = (a_1, \ldots, a_n) \) and express \( v \) by a column vector \( v = (v_{1,1}, \ldots, v_{1,N_1}, v_{2,1}, \ldots, v_{2,N_1}, \ldots, v_{n_1,1}, \ldots, v_{n_1,N_1})^T \).
2. Express \( \varphi \) as \( (\varphi_1, \ldots, \varphi_r) \) where \( \varphi_j \) is a linear functional on \( X_2 \otimes \cdots \otimes X_p \) and \( r = \prod_{i=2}^{p} \dim(X_i) \).
3. Express \( \psi \) as \( (\psi_1, \ldots, \psi_r) \) where \( \psi_j \) is a linear functional on \( C^{N_1} \).
4. Compute \( \tilde{a} = Wa \) and denote it by \( \tilde{a} = (\tilde{a}_1, \ldots, \tilde{a}_n)^T \).
5. Denote \( \tilde{v}_i = (\tilde{v}_{i,1}, \ldots, \tilde{v}_{i,N_1})^T, i = 1, \ldots, n_1 \).
6. \textbf{for} \( s = 1, \ldots, n_1 \) \textbf{do}
7. \quad \textbf{for} \( t = 1, \ldots, r \) \textbf{do}
8. \quad Compute \( w_{st} = \tilde{a}_s \varphi_t(B) \sum_{k=1}^{n_1} \omega_k^{s-1} \psi_t(v_k) \).
9. \quad \textbf{end for}
10. \textbf{end for}
11. Represent \( (w_{st}) \) as a column vector \( w = (w_{1,1}, \ldots, w_{1,r}, w_{2,1}, \ldots, w_{2,r}, \ldots, w_{n_1,1}, \ldots, w_{n_1,r})^T \).
12. Compute \( (W \otimes \vartheta)v \), which is the product \( (A \otimes B)v \).

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

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

\[
(C \otimes B) \begin{bmatrix} v \\
0 \\
\end{bmatrix} = \begin{bmatrix} (A \otimes B)v \\
(A' \otimes B)v \\
\end{bmatrix}.
\]

Hence to compute \( (A \otimes B)v \), it suffices to compute \( (C \otimes B) [v \ 0] \) 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 \( (v_1, \ldots, v_N)^T \).
2. Compute \( b = \sum_{i=1}^{2n-1} a_i \).
3. Construct \( c = (a_1, \ldots, a_{2n-1}, b, a_1, \ldots, a_{n-1}) \in C^{2n+1} \) representing a \( 2n_1 \times 2n_1 \) circulant matrix \( C \).
4. Construct \( \hat{v} = (v_1, \ldots, v_N, 0, 0) \in C^{2N} \).
5. Compute \( \hat{z} = (C \otimes B)\hat{v} \) by Algorithm 6 and express \( \hat{z} \) as \( (z_1, \ldots, z_{2N})^T \).
6. \( (z_1, \ldots, z_N)^T \) is the product of \( (A \otimes B) \) 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 10 $p$ levels Toeplitz-plus-Hankel matrix-vector product
1: Express $A$ as $H + T$ with Hankel matrix $H$ and Toeplitz matrix $T$.
2: Express $T$ as $(t_1, \ldots, t_{2n_1-1})$ and $H$ as $(h_1, \ldots, h_{2n_1-1})$.
3: Compute $b = -\sum t_j$.
4: Find $a \in \mathbb{C}$ such that
   $$a = \sum_{j=0}^{n_1-1} \omega_1^{j} t_{n_1+j} + \omega_1^{n_1} b + \sum_{j=1}^{n_1-1} \omega_1^{n_1+j} t_j$$
   where $\omega_1 = e^{k \pi i/n_1}$.
5: Update $H = H + aE$ and $T = T - aE$.
6: Compute $z_H = (H \odot B)v$ by Algorithm 8 and $z_T = (T \odot B)v$ by Algorithm 7, respectively.
7: Compute $z = z_H + z_T$ which is the product of $A$ and $v$.

Algorithm 11 $p$ levels sparse matrix-vector product
1: Express $A$ by its entries $A = (a_{ij})$.
2: Express $v$ by a column vector
   $$v = (v_1, \ldots, v_{1,N_1}, v_{2,1}, \ldots, v_{2,N_1}, \ldots, v_{n_1,1}, \ldots, v_{n_1,N_1})^\top$$.
3: Express $\varphi$ as $(\varphi_1, \ldots, \varphi_r)^\top$ where $\varphi_j$ is a linear functional on $X_2 \otimes \cdots \otimes X_p$ and $r = \prod_{i=2}^p \dim(X_i)$.
4: Express $\psi$ as $(\psi_1, \ldots, \psi_r)$ where $\psi_j$ is a linear functional on $\mathbb{C}^{N_1}$.
5: Denote $v_i = (v_1, \ldots, v_{i,N_1})^\top, i = 1, \ldots, n_1$.
6: for $s = 1, \ldots, n_1$ do
7:     for $t = 1, \ldots, r$ do
8:       Compute
       $$w_{s,t} = \varphi_t(B) \sum_{(k,s) \notin \Omega} a_{ks} \psi_t(v_k)$$
9:     end for
10: end for
11: Compute
    $$z_{ij} = (I \otimes \varnothing)(w_{s,t})$$.
12: $(z_{1,1}, \ldots, z_{1,n_1}, z_{2,1}, \ldots, z_{2,n_1}, \ldots, z_{N_1,1}, \ldots, z_{N_1,n_1})^\top$
is $(A \otimes B)v$.

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 $\tilde{v}$; 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 $\tilde{v}$; 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 $w_{s,t}$ costs one multiplication; note that computation of the coefficient $\hat{a}_s \varphi_t(B)$ does not cost any multiplication as $\hat{a}_s \varphi_t(B)$ is a linear combination of the entries of $A \otimes B$).

(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 $4(n-3)r$ 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


