*remote sensing*

MDPI

*Article*

# On the Efficient Implementation of Sparse Bayesian Learning-Based STAP Algorithms

**Kun Liu [1], Tong Wang [1,*], Jianxin Wu [2], Cheng Liu [1] and Weichen Cui [1]**

[1] National Lab of Radar Signal Processing, Xidian University, Xi'an 710071, China
[2] School of Electronics and Communication Engineering, Sun Yat-sen University, Guangzhou 510275, China
* Correspondence: twang@mail.xidian.edu.cn

**Abstract:** Sparse Bayesian learning-based space–time adaptive processing (SBL-STAP) algorithms can achieve superior clutter suppression performance with limited training sample support in practical heterogeneous and non-stationary clutter environments. However, when the system has high degrees of freedom (DOFs), SBL-STAP algorithms suffer from high computational complexity, since the large-scale matrix calculations and the inversion operations of large-scale covariance matrices are involved in the iterative process. In this article, we consider a computationally efficient implementation for SBL-STAP algorithms. The efficient implementation is based on the fact that the covariance matrices that need to be updated in the iterative process of the SBL-STAP algorithms have a Hermitian Toplitz-block-Toeplitz (HTBT) structure, with the result being that the inverse covariance matrix can be expressed in closed form by using a special case of the Gohberg–Semencul (G-S) formula. Based on the G-S-type factorization of the inverse covariance matrix and the structure of the used dictionary matrix, we can perform almost all operations in the SBL-STAP algorithms by 2-D FFT/IFFT. As a result, compared with the original SBL-STAP algorithms, even for moderate data sizes, the proposed algorithms can directly reduce the computational load by about two orders of magnitudes without any performance loss. Finally, simulation results validate the effectiveness of the proposed algorithms.

**Keywords:** clutter suppression; Gohberg–Semencul factorization; space–time adaptive processing; sparse Bayesian learning



**Citation:** Liu, K.; Wang, T.; Wu, J.; Liu, C.; Cui, W. On the Efficient Implementation of Sparse Bayesian Learning-Based STAP Algorithms. *Remote Sens.* **2022**, *14*, 3931. https://doi.org/10.3390/rs14163931

Academic Editor: Pedro Melo-Pinto

Received: 13 July 2022
Accepted: 11 August 2022
Published: 13 August 2022



## 1. Introduction

Space–time adaptive processing (STAP) [1–4] adopts two-dimensional joint adaptive filtering in the space and time domains to achieve effective filtering of clutter, and it is a key technology for radar clutter suppression and target detection using various types of motion platforms. The self-adaptation of STAP technology is reflected in the accurate perception of the external clutter environment, which relies on the real-time acquisition of the clutter plus noise covariance matrix (CNCM) of the cell under test (CUT). However, the CNCM is usually unknown in practical applications and need to be estimated on the basis of the independent and identically distributed (IID) training samples. To achieve an output signal-to-clutter-plus-noise ratio (SCNR) loss within 3 dB, according to the well-known Reed–Mallett–Brennan (RMB) rule [5], the number of IID training samples required to estimate CNCM should be greater than twice the system's degrees of freedom (DOFs). In fact, airborne radars usually work in heterogeneous and non-stationary clutter environments, and it is difficult to obtain enough IID training samples.

Sparse recovery (SR) techniques [6–23] can use limited training samples to reconstruct signals with high precision, and this feature is exactly in line with the requirement of using as few observation samples as possible to accurately describe the clutter characteristics in the STAP. Thus, SR-based STAP (SR-STAP) techniques [24–29] have inherent advantages for CNCM estimation in fast-changing clutter environments. In recent years, many SR techniques have been applied to airborne radar clutter suppression processing to improve

the detection performance of slow moving targets under the condition that there are not enough training samples in practice. The greedy algorithms [10–13] iteratively select atoms from the dictionary and calculate the corresponding sparse coefficients, so that the difference between the linear combination of these atoms and the observed data is gradually reduced. The convex optimization (CVX) algorithms [14–19] relax the $\ell_0$-norm optimization problem into a CVX problem, and use the properties of the CVX function to obtain the sparse coefficient vector. The focal underdetermined system solver (FOCUSS) algorithms [20,21] use iterative $\ell_p$-norm ($0 < p < 1$) optimization to approximate $\ell_0$-norm optimization and transform the $\ell_0$-norm optimization problem into a weighted minimum norm least square problem. The iterative adaptive approach (IAA) [22,23] is a non-parametric algorithm based on iterative weighted least squares approach. Although these SR techniques have great advantages in combination with STAP, they suffer from some drawbacks: the greedy algorithms may fail when there has a strong correlation between the atoms of the dictionary, resulting in poor sparse coefficient solutions. The performance of CVX algorithms and FOCUSS algorithms is closely related to the choice of regularization parameters and the IAA algorithms easily suffer from severe performance degradation in the non-ideal case.

Sparse Bayesian learning (SBL) [30–35], proposed by Tipping, is a popular SR technique for signal reconstruction. Compared with other SR algorithms, SBL does not require setting regularization parameters and can be used to obtain a sparser global optimal solution. Moreover, SBL can still achieve favorable performance when the dictionary possesses a high coherence. Due to their superior performance, SBL-based STAP (SBL-STAP) algorithms with multiple measurement vectors (MMV) [36–43] have been widely researched. However, in MMV-based SBL-STAP (MSBL-STAP) algorithms, an iterative procedure that converges very slowly is utilized to reconstruct the CNCM. Additionally, inversion operations of the large-scale covariance matrices and several large-scale matrix calculations are involved in each iteration, which is quite computationally expensive for MSBL-STAP algorithms in practical applications. To tackle this problem, many efficient MSBL-STAP methods have been developed. In [44], a fast tensor-based three-dimensional MSBL-STAP (TMSBL-STAP) algorithm was proposed, in which the large-scale matrix calculation was decomposed into small-scale matrix calculation by utilizing the Kronecker structure of the data. However, this algorithm only relieves a small part of the computational burden. In [45], by combining a simple approximation term, a fast-converging MSBL-STAP (MFCSBL-STAP) algorithm was proposed to improve the convergence of MSBL-STAP algorithm. In [46], an MSBL-STAP algorithm based on the iterative reweighed $\ell_{2,1}$-norm (IR$\ell_{2,1}$-MSBL-STAP) was proposed, and the experiments showed that the algorithm had great convergence performance. Compared with the basic MSBL-STAP algorithms, these two algorithms can greatly improve the convergence speed and exhibit a comparable or even a better reconstruction accuracy. However, they ignore the core problem that there exist large-scale covariance matrix inversion operations and large-scale matrix calculations in each iteration of these MSBL-STAP algorithms, which have high computational complexity.

In this article, we propose several efficient MSBL-STAP algorithms based on the G-S factorization [47] for airborne radar in the case of uniformly spaced linear array (ULA) and a constant pulse repetition frequency (PRF). In our proposed algorithms, based on the fact that the inverse of the Hermitian Toplitz-block-Toeplitz (HTBT) matrix has low displacement ranks [48] and can be written in a G-S factorization-based form, an equivalent G-S factorization-based method is utilized to efficiently calculate the inverse covariance matrices in the iterative process of the MSBL-STAP algorithms. Then, by utilizing the property whereby the dictionary matrix is the Kronecker product of two Fourier matrices and the obtained G-S-type factors of the inverse covariance matrix, many large-scale matrices in the iterative process of the MSBL-STAP algorithms can be efficiently computed by using 2D-FFT/IFFT [49].

The main contributions of this paper can be listed as follows:

(a) The algorithm efficiency is the focus of this paper; in this paper, several computationally effective MSBL-STAP algorithms based on G-S factorization are proposed. In

our proposed algorithms, utilizing the G-S factorization of the inverse of the Hermitian Toplitz-block-Toeplitz matrix and the structure of the dictionary matrix, almost all the processing procedures of the original MSBL-STAP algorithms can be implemented with fast FFT/IFFT. Compared with the original MSBL-STAP algorithms, these proposed algorithms can directly reduce the computational complexity by several orders of magnitude without any performance loss.

(b) A detailed comparison is presented to show the computational complexity of the proposed computationally efficient MSBL-STAP algorithms and the original MSBL-STAP algorithms and other SR-STAP algorithms.

(c) A detailed comparative analysis of our proposed algorithms, including the convergence speed, the clutter suppression performance and the target detection performance, with the original MSBL-STAP algorithms and other SR-STAP algorithms is carried out.

The rest of the paper is organized as follows. In Section 2, the general space–time sparse signal model is introduced. In Section 3, a brief review of the traditional MSBL-STAP algorithms is provided. In Section 4, we give a detail introduction of the proposed algorithms. In Section 5, simulation results are provided to demonstrate the computational efficiency, the clutter suppression performance and the target detection performance of the proposed algorithms. Final conclusions are given in Section 6.

Notation: Boldface lowercase letters denote vectors and boldface uppercase letters denote matrices. $\mathbb{R}^+$ represents the nonnegative real field and $\mathbb{C}$ represents the complex field. $(\bullet)^*$, $(\bullet)^T$ and $(\bullet)^H$ represent the complex conjugate, transpose and conjugate transpose, respectively. The symbol $\otimes$ denotes the Kronecker product. 0 represents a zero vector/matrix. $\mathbf{I}_{NK}$ denotes the $NK \times NK$ identity matrix. $diag(\bullet)$ represents a diagonal matrix with entries of a vector on the diagonal or a vector is made up of all the elements on the diagonal of a matrix. The symbol $\triangleq$ denotes a definition. $\|\bullet\|_F$ denotes the Frobenius norm. The symbol $\sim$ above a matrix represents reversing the elements in a matrix first by row, then by column. $\mathcal{F}_{2-D}(\bullet)_{N,K}$ and $\mathcal{IF}_{2-D}(\bullet)_{N,K}$ denote the $N$- and $K$-point 2-D FFT and IFFT operations.

## 2. Signal Model

Consider an airborne pulsed-Doppler radar system employing a side-looking ULA consisting of $N$ elements. The interelement spacing is $d = \lambda/2$, where $\lambda$ is the wavelength. $K$ pulses are transmitted at a constant PRF during a coherent processing interval (CPI). Then, the space–time sparse signal model in the MMV case can be written as

$$\mathbf{Y} = \mathbf{DX} + \mathbf{N} \tag{1}$$

where $\mathbf{Y} = \left[\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_L\right] \in \mathbb{C}^{NK \times L}$ is the received clutter plus noise data, $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L] \in \mathbb{R}^{N_s K_d \times L}$ is the unknown angle-Doppler profile to be recovered with each row representing a possible clutter component, $\mathbf{N} = [\mathbf{n}_1, \mathbf{n}_2, \ldots, \mathbf{n}_L] \in \mathbb{C}^{NK \times L}$ is the zero mean noise matrix with covariance matrix $\sigma^2\mathbf{I}$, $\sigma^2$ is the noise power and $\mathbf{I}$ is the identity matrix, $\mathbf{D} = \mathbf{S_t} \otimes \mathbf{S_s} = \left[\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{N_s K_d}\right] \in \mathbb{C}^{NK \times N_s K_d}$ is the space–time dictionary matrix, $\mathbf{v}_m (m = 1, 2, \ldots, N_s K_d)$ is the spatial-temporal steering vector of the $m$th grid point of the whole angle-Doppler plane, $N_s = \rho_s N (\rho_s > 1)$ is the number of normalized spatial frequency bins and $K_d = \rho_d K (\rho_d > 1)$ is the number of normalized Doppler frequency bins, $\mathbf{S_s} = [\mathbf{s}_{s,1}, \mathbf{s}_{s,2}, \ldots, \mathbf{s}_{s,N_s}] \in \mathbb{C}^{N \times N_s}$ and $\mathbf{S}_t = [\mathbf{s}_{t,1}, \mathbf{s}_{t,2}, \ldots, \mathbf{s}_{t,K_d}] \in \mathbb{C}^{K \times K_d}$ are two Fourier matrices, $\mathbf{s}_{s,n}$ and $\mathbf{s}_{t,k}$ are the spatial and temporal steering vectors, given by

$$\mathbf{s}_{s,n} = \left\{1, \exp\left[-j2\pi f_{s,n}\right], \ldots, \exp\left[-j2\pi(N-1) f_{s,n}\right]\right\}^T,\ n = 1, 2, \ldots, N_s \tag{2}$$

$$\mathbf{s}_{t,k} = \left\{1, \exp\left[-j2\pi f_{d,k}\right], \ldots, \exp\left[-j2\pi(K-1) f_{d,k}\right]\right\}^T,\ k = 1, 2, \ldots, K_d \tag{3}$$

where $f_{s,n} = (n-1)/N_s$ is the normalized spatial frequency of the $n$th angle grid point and $f_{d,k} = (k-1)/K_d$ is the normalized Doppler frequency of the $k$th Doppler grid point.

## 3. A Brief Review of the Traditional MSBL-STAP Algorithms

In this section, firstly, we give a brief review of the basic MSBL-STAP algorithm proposed by Duan [36]; then, we directly give the pseudocodes of two fast-converging MSBL-STAP algorithms for the brevity of the article. According to the signal model in (1), the Gaussian likelihood function of the measurement can be denoted as

$$p\left(\mathbf{Y}|\mathbf{X};\sigma^2\right) = \left(\pi\sigma^2\right)^{-NKL}\exp\left(-\sigma^{-2}\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2\right) \tag{4}$$

Suppose that each column in $\mathbf{X}$ obeys a complex Gaussian prior, i.e.,

$$\mathbf{x}_l \sim N(0,\mathbf{\Gamma}),\ l = 1,2,\ldots,L \tag{5}$$

where $\mathbf{\Gamma}$= diag($\boldsymbol{\gamma}$), $\boldsymbol{\gamma} = \left[\gamma_1,\gamma_2,\ldots,\gamma_{N_sK_d}\right]^T$ are the unknown variance parameters which controlling the prior covariance of $\mathbf{x}_l$. Then, we can obtain the prior probability density function (PDF) of $\mathbf{X}$

$$p(\mathbf{X};\mathbf{\Gamma}) = \pi^{-N_sK_dL}|\mathbf{\Gamma}|^{-L}\exp\left(-\sum_{l=1}^{L}\mathbf{x}_l^H\mathbf{\Gamma}^{-1}\mathbf{x}_l\right) \tag{6}$$

If the above prior distributions are obtained, we can obtain the posterior PDF of $\mathbf{X}$ by using Bayesian estimation methods [30]:

$$p\left(\mathbf{X}|\mathbf{Y};\mathbf{\Gamma},\sigma^2\right) = \pi^{-N_sK_dL}|\mathbf{\Sigma}|^{-L}\exp\left[\sum_{l=1}^{L}-(x_l-\boldsymbol{\mu}_l)^H\mathbf{\Sigma}^{-1}(x_l-\boldsymbol{\mu}_l)\right] \tag{7}$$

where $\boldsymbol{\mu} = [\boldsymbol{\mu}_1,\boldsymbol{\mu}_2,\ldots,\boldsymbol{\mu}_L]$ and $\mathbf{\Sigma}$ are the posteriori mean matrix and the posteriori covariance matrix respectively, given by

$$\mathbf{\Sigma} = \mathbf{\Gamma} - \mathbf{\Gamma}\mathbf{D}^H\mathbf{R}^{-1}\mathbf{D}\mathbf{\Gamma} \tag{8}$$

$$\boldsymbol{\mu} = \mathbf{\Gamma}\mathbf{D}^H\mathbf{R}^{-1}\mathbf{Y} \tag{9}$$

where $\mathbf{R} = \sigma^2\mathbf{I}_{NK} + \mathbf{D}\mathbf{\Gamma}\mathbf{D}^H$ is the covariance matrix to be inverted in each iteration of the basic MSBL-STAP algorithm. Then, we use the expectation-maximization (EM) [30] method to estimated $\gamma_m$ ($m = 1,2,\ldots,N_sK_d$) and $\sigma^2$, which are unknown hyperparameters in $\boldsymbol{\mu}$. We have

$$\gamma_m^{t+1} = \frac{1}{L}\sum_{l=1}^{L}\left(\mu_{l,m}^t\right)^2 + \mathbf{\Sigma}_{m,m}^t \tag{10}$$

$$\left(\sigma^2\right)^{t+1} = \frac{(1/L)\|\mathbf{Y}-\mathbf{D}\boldsymbol{\mu}^t\|_F^2}{NK - \sum_{m=1}^{N_sK_d}\left(1-\mathbf{\Sigma}_{m,m}^t/\gamma_m^t\right)} \tag{11}$$

where the superscript $t$ indicates the $t$th iteration, $\mu_{l,m}^t$ is the $m$th component of $\boldsymbol{\mu}_l^t$, $\mathbf{\Sigma}_{m,m}^t$ is the $m$th component of the main diagonal of $\mathbf{\Sigma}^t$. In fact, by updating and iterating $\boldsymbol{\mu}$, we can finally obtain the optimal sparse solution $\hat{\mathbf{X}}$ when a predefined convergence criterion is satisfied:

$$\frac{\|\boldsymbol{\mu}^t - \boldsymbol{\mu}^{t-1}\|}{\boldsymbol{\mu}^t} < \delta \tag{12}$$

where $\delta$ is a small enough positive value. Then, the angle-Doppler profile $\hat{\mathbf{X}}$ can be given by

$$\hat{\mathbf{X}} = \boldsymbol{\mu} \tag{13}$$

Then, we can estimate the CNCM by the formula

$$\mathbf{R}_{c+n} = \frac{1}{L}\sum_{l=1}^{L}\sum_{m=1}^{N_sK_d}\left|\hat{x}_l^m\right|^2 \mathbf{v}_m\mathbf{v}_m^H + \alpha\sigma^2\mathbf{I}_{NK} \tag{14}$$

where $\hat{x}_l^m$ is the $m$th element of the $l$th column of $\hat{\mathbf{X}}$ and $\alpha$ is a positive loading factor. Finally, we can obtain the optimal STAP weight vector based on the linearly constrained minimum variance (LCMV) principle, given by

$$\mathbf{w}_{\text{opt}} = \frac{\mathbf{R}_{c+n}^{-1}\mathbf{v}_t}{\mathbf{v}_t^H\mathbf{R}_{c+n}^{-1}\mathbf{v}_t} \tag{15}$$

where $\mathbf{v}_t$ is the spatial-temporal steering vector of the target.

The MFCSBL-STAP algorithm [45] proposed by Wang and the IR$\ell_{2,1}$-MSBL-STAP [46] algorithm proposed by Liu significantly accelerate the convergence speed of the basic MSBL-STAP algorithm proposed by Duan [36]. For the sake of brevity in this article, here we will not describe these two algorithms in detail. Instead, we give the pseudocodes of these two algorithms in Tables 1 and 2.

**Table 1.** Pseudocode of MFCSBL-STAP algorithm.

**Input**: training samples $\mathbf{Y}$, dictionary matrix $\mathbf{D}$.

**Initialize**:

$\boldsymbol{\gamma}^0 = \mathbf{1}$, $\left(\sigma_0^2\right)^0 = 1$, $\boldsymbol{\Gamma}^0 = diag(\boldsymbol{\gamma}_0)$, $\mathbf{R}^0 = \left(\sigma^2\right)^0\mathbf{I}_{NK} + \mathbf{D}\boldsymbol{\Gamma}^0\mathbf{D}^H$, $\mathbf{R}_{\text{ML}} = \mathbf{Y}\mathbf{Y}^H/L$.

**Repeat:**

$\boldsymbol{\mu}^t = \boldsymbol{\Gamma}^t\mathbf{D}^H\left(\mathbf{R}^t\right)^{-1}\mathbf{Y}$

$\gamma_m^{t+1} = \left(\gamma_m^t\right)^2\left|\mathbf{v}_m^H\left(\mathbf{R}^t\right)^{-1}\mathbf{R}_{\text{ML}}\left(\mathbf{R}^t\right)^{-1}\mathbf{v}_m\right|$, $(m = 1, 2, \ldots, N_sK_d)$

$\left(\sigma^2\right)^{t+1} = \left(1/L\right)\|\mathbf{Y} - \mathbf{D}\boldsymbol{\mu}^t\|_F^2 \Big/ \left[NK - \sum_{m=1}^{N_sK_d}\gamma_m^t\mathbf{v}_m^H\left(\mathbf{R}^t\right)^{-1}\mathbf{v}_m\right]$ $(m = 1, 2, \ldots, N_sK_d)$

$\mathbf{R}^{t+1} = \left(\sigma^2\right)^{t+1}\mathbf{I}_{NK} + \mathbf{D}\boldsymbol{\Gamma}^{t+1}\mathbf{D}^H$

**The iterative procedure terminates when the iteration termination condition in (12) is satisfied.**

Get the estimated angle-Doppler profile $\hat{\mathbf{X}}$ using (13).

Reconstruct the CNCM using (14) and compute the optimal STAP weight vector using (15).

**Table 2.** Pseudocode of IR$\ell_{2,1}$-MSBL-STAP algorithm.

**Input**: training samples $\mathbf{Y}$, dictionary matrix $\mathbf{D}$.

**Initialize**:

$\boldsymbol{\gamma}^0 = \mathbf{1}$, $\left(\sigma_0^2\right)^0 = 1$, $\boldsymbol{\Gamma}^0 = diag(\boldsymbol{\gamma}_0)$, $\mathbf{R}^0 = \left(\sigma^2\right)^0\mathbf{I}_{NK} + \mathbf{D}\boldsymbol{\Gamma}^0\mathbf{D}^H$.

**Repeat:**

$\boldsymbol{\mu}^t = \boldsymbol{\Gamma}^t\mathbf{D}^H\left(\mathbf{R}^t\right)^{-1}\mathbf{Y}$

$\boldsymbol{\Sigma}^t = \boldsymbol{\Gamma}^t - \boldsymbol{\Gamma}^t\mathbf{D}^H\left(\mathbf{R}^t\right)^{-1}\mathbf{D}\boldsymbol{\Gamma}^t$

$\gamma_m^{t+1} = \left[\mathbf{v}_m^H\left(\mathbf{R}^t\right)^{-1}\mathbf{v}_m\right]^{-\frac{1}{2}}\left[\frac{1}{L}\sum_{l=1}^{L}\left(\mu_{l,m}^t\right)^2\right]^{\frac{1}{2}}$, $(m = 1, 2, \ldots, N_sK_d)$

$\left(\sigma^2\right)^{t+1} = \left[\left(1/L\right)\|\mathbf{Y} - \mathbf{D}\boldsymbol{\mu}^t\|_F^2\right] \Big/ \left(NK - \sum_{m=1}^{N_sK_d}\left(1 - \boldsymbol{\Sigma}_{m,m}^t/\gamma_m^t\right)\right)$ $(m = 1, 2, \ldots, N_sK_d)$

$\mathbf{R}^{t+1} = \left(\sigma^2\right)^{t+1}\mathbf{I}_{NK} + \mathbf{D}\boldsymbol{\Gamma}^{t+1}\mathbf{D}^H$

**The iterative procedure terminates when the iteration termination condition in (12) is satisfied.**

Get the estimated angle-Doppler profile $\hat{\mathbf{X}}$ using (13).

Reconstruct the CNCM using (14) and compute the optimal STAP weight vector using (15).

## 4. Proposed Algorithms

From the procedures of the basic MSBL-STAP algorithm, the MFCSBL-STAP algorithm and the IR$\ell_{2,1}$-MSBL-STAP algorithm, it can be observed that we need to calculate the

product of many large-scale matrices and the inverse of the covariance matrix $\mathbf{R}$ in each iteration. The size of $\mathbf{R}$ is $NK \times NK$, i.e., the computational complexity in each iteration of these MSBL-STAP algorithms is at least $o\left((NK)^3\right)$. Thus, the MSBL-STAP algorithms have a computational complexity that grows rapidly with the DOFs of the STAP system, which hinders its application in many practical problems with even moderately large data sets. To tackle this question, in this section, an efficient G-S factorization-based implementation for these MSBL-STAP algorithms is proposed.

From (1), we know that the space–time dictionary matrix $\mathbf{D}$ is the Kronecker product of two Fourier matrices $\mathbf{S_t}$ and $\mathbf{S_s}$. The $kN_s + n + 1$ column of $\mathbf{D}$ can be written as

$$\mathbf{D}(\omega_k, \omega_n) = \mathbf{s}_t(\omega_k) \otimes \mathbf{s}_s(\omega_n) \tag{16}$$

where

$$\mathbf{s}_t(\omega_k) = \left\{1, \exp\left[-j\omega_k\right], \ldots, \exp\left[-j(K-1)\omega_k\right]\right\}^T \tag{17}$$

$$\mathbf{s}_s(\omega_n) = \left\{1, \exp\left[-j\omega_n\right], \ldots, \exp\left[-j(N-1)\omega_n\right]\right\}^T \tag{18}$$

where $\omega_k = 2\pi k/K_d, k = 0, 1, \ldots, K_d - 1$, $\omega_n = 2\pi n/N_s, n = 1, 2, \ldots, N_s$. The covariance matrix $\mathbf{R}$ in each iteration of the MSBL-STAP algorithms can be represented by

$$\mathbf{R} = \sigma^2\mathbf{I} + \mathbf{Q} \tag{19}$$

where $\mathbf{Q} = \mathbf{D\Gamma D}^H$ and can be represented by

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_0 & \mathbf{Q}_1^H & \cdots & \mathbf{Q}_{K-1}^H \\ \mathbf{Q}_1 & \mathbf{Q}_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{Q}_1^H \\ \mathbf{Q}_{K-1} & \cdots & \mathbf{Q}_1 & \mathbf{Q}_0 \end{bmatrix} \tag{20}$$

From (20), we know that $\mathbf{Q}$ is a HTBT matrix [50,51], and each submatrix $\mathbf{Q}_{j_1}$ can be calculated by

$$\mathbf{Q}_{j_1} = \mathbf{M}_0 + e^{-j2\pi\frac{1}{K_d}j_1}\mathbf{M}_1 + \cdots + e^{-j2\pi\frac{K_d-1}{K_d}j_1}\mathbf{M}_{K_d-1} \tag{21}$$

where $j_1 = 0, 1, \ldots, K-1$, $\mathbf{M}_k = \mathbf{S}_s\mathbf{\Lambda}_k\mathbf{S}_s^H$, $k = 0, 1, \ldots, K_d - 1$, $\mathbf{\Lambda}_k$= daig($\overline{\gamma}_k$), $\overline{\gamma}_k$ is the $k$th column vector of the matrix $\overline{\mathbf{\Gamma}}$, given $\{\gamma_m\}_{m=1}^{N_sK_d}$, $\overline{\mathbf{\Gamma}}$ is a $N_s \times K_d$ matrix

$$\overline{\mathbf{\Gamma}} = \begin{bmatrix} \gamma_1 & \gamma_{N_s+1} & \cdots & \gamma_{(K_d-1)N_s+1} \\ \gamma_2 & \gamma_{N_s+2} & \cdots & \gamma_{(K_d-1)N_s+2} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{N_s} & \gamma_{2N_s} & \cdots & \gamma_{K_dN_s} \end{bmatrix} \tag{22}$$

From (21) and (22), we know that the submatrix $\mathbf{Q}_{j_1}$ is an $N \times N$ Toeplitz matrix, which can be represented by

$$\mathbf{Q}_{j_1} = \begin{bmatrix} q_{j_1,0} & q_{j_1,-1} & \cdots & q_{j_1,-N+1} \\ q_{j_1,1} & q_{j_1,0} & \ddots & \vdots \\ \vdots & \ddots & \ddots & q_{j_1,-1} \\ q_{j_1,N-1} & \cdots & q_{j_1,1} & q_{j_1,0} \end{bmatrix} \tag{23}$$

From (20) and (23), we find that if we want to obtain the matrix $\mathbf{Q}$, we only need to obtain the elements of the first row and first column of each submatrix $\mathbf{Q}_{j_1}$, and the total number

of required elements to construct $\mathbf{Q}$ is $(2N-1)K$, which is far less than $(NK)^2$. Utilizing the definition of $\mathbf{Q}$, we get

$$q_{j_1,j_2} = \sum_{k=0}^{K_d-1} \sum_{n=0}^{N_s-1} \gamma_{kN_s+n+1} e^{-j2\pi \frac{j_2}{N_s} n} e^{-j2\pi \frac{j_1}{K_d} k} \tag{24}$$

where $j_2 = 0, 1, \ldots, N-1$. According to (24), we find that $\left\{q_{j_1,j_2}\right\}$ and $\left\{\gamma_{kN_s+n+1}\right\}$ form a Fourier transform pair. Thus, by performing 2-D FFT to the $\underline{\mathbf{\Gamma}}$, we can efficiently obtain the matrix $\mathbf{Q}$. Since this term $\mathbf{Q} = \mathbf{D}\boldsymbol{\Gamma}\mathbf{D}^H$ appears in all the three MSBL-STAP algorithms mentioned above, this fast implementation to calculate $\mathbf{Q}$ is applicable to all of them.

Then, we can calculate the covariance matrix $\mathbf{R}$ using (19), and it is easy to know that $\mathbf{R}$ is also a HTBT matrix with the same structure as $\mathbf{Q}$ in (20). Next, we detail the G-S decomposition [47,52] of $\mathbf{R}^{-1}$ and show how to efficiently calculate $\mathbf{R}^{-1}$. It follows from (20) that $\mathbf{R}$ has the following structure:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_0 & \mathbf{R}_{K-1}^H \\ \mathbf{R}_{K-1} & \mathbf{R}_{K-1,N} \end{bmatrix} \tag{25}$$

$$= \begin{bmatrix} \mathbf{R}_{K-1,N} & \tilde{\mathbf{R}}_{K-1}^* \\ \tilde{\mathbf{R}}_{K-1}^T & \mathbf{R}_0 \end{bmatrix} \tag{26}$$

where

$$\mathbf{R}_{K-1} = \left[\mathbf{R}_1^T, \mathbf{R}_2^T, \ldots, \mathbf{R}_{K-1}^T\right]^T \tag{27}$$

$$\tilde{\mathbf{R}}_{K-1}^T = [\mathbf{R}_{K-1}, \mathbf{R}_{K-2}, \ldots, \mathbf{R}_1] \tag{28}$$

It can be seen that $\mathbf{R}_{K-1,N}$ is a $(K-1) \times (K-1)$ HTBT matrix. Define the $N \times N$ exchange matrix $\mathbf{S}_N$ as

$$\mathbf{S}_N \triangleq \begin{bmatrix} & & 1 \\ & \iddots & \\ 1 & & \end{bmatrix} \tag{29}$$

According to (27) and (28), we get

$$\tilde{\mathbf{R}}_{K-1} = \mathbf{S}_{(K-1)N} \mathbf{R}_{K-1} \mathbf{S}_N \tag{30}$$

Applying the formula for the inverse of a partitioned matrix [53] to the right side of (25) and (26), we get

$$\mathbf{R}^{-1} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{K-1,N}^{-1} \end{bmatrix} + \begin{bmatrix} \mathbf{I}_N \\ \mathbf{A}_{K-1} \end{bmatrix} \mathbf{W}_N^{-1} \left[\mathbf{I}_N \mathbf{A}_{K-1}^H\right] \tag{31}$$

$$= \begin{bmatrix} \mathbf{R}_{K-1,N}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{B}_{K-1}^* \\ \mathbf{I}_N \end{bmatrix} \mathbf{V}_N^{-1} \left[\mathbf{B}_{K-1}^T \mathbf{I}_N\right] \tag{32}$$

where

$$\mathbf{A}_{K-1} = -\mathbf{R}_{K-1,N}^{-1} \mathbf{R}_{K-1} \tag{33}$$

$$\mathbf{W}_N = \mathbf{R}_0 - \mathbf{R}_{K-1}^H \mathbf{R}_{K-1,N}^{-1} \mathbf{R}_{K-1} \tag{34}$$

$$\mathbf{B}_{K-1}^* = -\mathbf{R}_{K-1,N}^{-1} \tilde{\mathbf{R}}_{K-1}^* \tag{35}$$

$$\mathbf{V}_N = \mathbf{R}_0 - \tilde{\mathbf{R}}_{K-1}^T \mathbf{R}_{K-1,N}^{-1} \tilde{\mathbf{R}}_{K-1}^* \tag{36}$$

Due to the persymmetric property [54] of the HTBT matrix and its inverse, we have

$$\mathbf{S}_{(K-1)N} \mathbf{R}_{K-1,N} \mathbf{S}_{(K-1)N} = \mathbf{R}_{K-1,N}^T \tag{37}$$

$$\mathbf{S}_{(K-1)N} \mathbf{R}_{K-1,N}^{-1} \mathbf{S}_{(K-1)N} = \mathbf{R}_{K-1,N}^{-T} \tag{38}$$

$$\mathbf{R}_{K-1,N}^{-1}\mathbf{S}_{(K-1)N} = \mathbf{S}_{(K-1)N}\mathbf{R}_{K-1,N}^{-*} \tag{39}$$

Substituting (38) and (39) into (35) and (36), we get

$$\mathbf{B}_{K-1}^{*} = \tilde{\mathbf{A}}_{K-1}^{*} \tag{40}$$

$$\mathbf{V}_{N} = \tilde{\mathbf{W}}_{N}^{T} \tag{41}$$

The detailed derivation of (39)–(41) is shown in Appendix A. Then, substituting (40) and (41) into (32), $\mathbf{R}^{-1}$ can be reformulated as

$$\mathbf{R}^{-1} = \begin{bmatrix} \mathbf{R}_{K-1,N}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{A}}_{K-1}^{*} \\ \mathbf{I}_N \end{bmatrix} \tilde{\mathbf{W}}_{N}^{-T} \begin{bmatrix} \tilde{\mathbf{A}}_{K-1}^{T}\mathbf{I}_N \end{bmatrix} \tag{42}$$

Define a $K \times K$ lag-1 shifting matrix $\mathbf{J}_K$, which has the following form

$$\mathbf{J}_K \triangleq \begin{bmatrix} 0 & & & \\ 1 & 0 & & \\ & \ddots & \ddots & \\ & & 1 & 0 \end{bmatrix} \tag{43}$$

The block matrix has the fact that

$$\mathbf{J}_{K,N} \begin{bmatrix} \mathbf{R}_{K-1,N}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \mathbf{J}_{K,N}^{T} = \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_{K-1,N}^{-1} \end{bmatrix} \tag{44}$$

where $\mathbf{J}_{K,N} = \mathbf{J}_K \otimes \mathbf{I}_N$. Based on (31), (42) and (44), we obtain the displacement representation [48] of $\mathbf{R}^{-1}$, given by

$$\begin{aligned} \nabla\mathbf{R}^{-1} &= \mathbf{R}^{-1} - \mathbf{J}_{K,N}\mathbf{R}^{-1}\mathbf{J}_{K,N}^{T} \\ &= \begin{bmatrix} \mathbf{I}_N \\ \mathbf{A}_{K-1} \end{bmatrix} \mathbf{W}_{N}^{-1} \begin{bmatrix} \mathbf{I}_N & \mathbf{A}_{K-1}^{H} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{A}}_{K-1}^{*} \end{bmatrix} \tilde{\mathbf{W}}_{N}^{-T} \begin{bmatrix} \mathbf{0} & \tilde{\mathbf{A}}_{K-1}^{T} \end{bmatrix} \end{aligned} \tag{45}$$

Let

$$\mathbf{T} = \begin{bmatrix} \mathbf{I}_N \\ \mathbf{A}_{K-1} \end{bmatrix} \mathbf{W}_{N}^{-1/2} = [\mathbf{t}_0, \mathbf{t}_1, \ldots, \mathbf{t}_{N-1}] \tag{46}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{0} \\ \tilde{\mathbf{A}}_{K-1}^{*} \end{bmatrix} \left(\tilde{\mathbf{W}}_{N}^{-1/2}\right)^{T} = [\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_{N-1}] \tag{47}$$

where $n = 0, 1, \ldots, N-1$, $\mathbf{t}_n \in \mathbb{C}^{NK\times 1}$ and $\mathbf{p}_n \in \mathbb{C}^{NK\times 1}$ are the $(n+1)$th column vectors of $\mathbf{T}$ and $\mathbf{P}$, respectively. Substituting (46) and (47) into (45), $\nabla\mathbf{R}^{-1}$ can be rewritten as

$$\begin{aligned} \nabla\mathbf{R}^{-1} &= \mathbf{T}\mathbf{T}^{H} - \mathbf{P}\mathbf{P}^{H} \\ &= \sum_{n=0}^{N-1} \left(\mathbf{t}_n\mathbf{t}_n^{H} - \mathbf{p}_n\mathbf{p}_n^{H}\right) \end{aligned} \tag{48}$$

For the block matrix $\mathbf{R}^{-1}$, it has the fact that

$$\left(\mathbf{J}_{K,N}\right)^{K} \mathbf{R}^{-1} \left(\mathbf{J}_{K,N}^{T}\right)^{K} = \mathbf{0} \tag{49}$$

Using (48) and (49), $\mathbf{R}^{-1}$ can be written as

$$\begin{aligned} \mathbf{R}^{-1} &= \sum_{k=0}^{K-1} \left(\mathbf{J}_{K,N}\right)^{k} \left(\nabla\mathbf{R}^{-1}\right) \left(\mathbf{J}_{K,N}^{T}\right)^{k} \\ &= \sum_{n=0}^{N-1} \sum_{k=0}^{K-1} \left(\mathbf{J}_{K,N}\right)^{k} \left(\mathbf{t}_n\mathbf{t}_n^{H} - \mathbf{p}_n\mathbf{p}_n^{H}\right) \left(\mathbf{J}_{K,N}^{T}\right)^{k} \end{aligned} \tag{50}$$

Let $\mathbf{U} = \left[\mathbf{U}_0^T, \mathbf{U}_1^T, \ldots, \mathbf{U}_{K-1}^T\right]^T \in \mathbb{C}^{NK\times M}$, we define a Toplitz-block matrix $\mathcal{L}_{K,N}(\mathbf{U}, \mathbf{J}_{K,N})$ as

$$\begin{aligned}\mathcal{L}_{K,N}(\mathbf{U}, \mathbf{J}_{K,N}) &\triangleq \left[\mathbf{U}, \mathbf{J}_{K,N}\mathbf{U}, \ldots, \left(\mathbf{J}_{K,N}\right)^{K-1}\mathbf{U}\right] \\ &= \begin{bmatrix} \mathbf{U}_0 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{U}_1 & \mathbf{U}_0 & \cdots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{0} \\ \mathbf{U}_{K-1} & \mathbf{U}_{K-2} & \cdots & \mathbf{U}_0 \end{bmatrix}_{NK\times KM}\end{aligned} \tag{51}$$

Then, $\mathbf{R}^{-1}$ can be reformulated as

$$\mathbf{R}^{-1} = \sum_{n=0}^{N-1}\left[\mathcal{L}_{K,N}(\mathbf{t}_n, \mathbf{J}_{K,N})\mathcal{L}_{K,N}^H(\mathbf{t}_n, \mathbf{J}_{K,N}) - \mathcal{L}_{K,N}(\mathbf{p}_n, \mathbf{J}_{K,N})\mathcal{L}_{K,N}^H(\mathbf{p}_n, \mathbf{J}_{K,N})\right] \tag{52}$$

Equation (52) is termed as a two-dimensional (2-D) G-S formula, where $\mathbf{t}_n$ and $\mathbf{p}_n$ are the G-S decomposition factors of $\mathbf{R}^{-1}$. From (46), (47) and (52), it is clear that once we obtain the matrices $\mathbf{A}_{K-1}$ and $\mathbf{W}_N$, we can compute the matrices $\mathbf{T}$ and $\mathbf{P}$, and then we can compute $\mathbf{R}^{-1}$ by using (52). Meanwhile, the matrices $\mathbf{A}_{K-1}$ and $\mathbf{W}_N$ can be calculated using a 2-D Levinson–Durbin (L-D)-type algorithm with $o\left(8N^3\left(K^2 - K + 2\right)\right)$ flops. By extending the L-D algorithm in the one-dimensional case in [55], we can obtain the L-D algorithm in the 2-D case. Here, we give the procedures of the 2-D L-D algorithm.

(1) Calculate the initial values

$$\mathbf{A}_1 = -\mathbf{R}_0^{-1}\mathbf{R}_1 \tag{53}$$

$$(\mathbf{R}_{K-1})_1 = \mathbf{R}_1 \tag{54}$$

$$\mathbf{W}_N^{(1)} = \mathbf{R}_0 - \mathbf{R}_1^H\mathbf{R}_0^{-1}\mathbf{R}_1 \tag{55}$$

(2) Repeat: $k = 2, 3, \ldots, K-1$

$$\mathbf{H}_{k-1} = \tilde{\mathbf{A}}_{k-1}^T(\mathbf{R}_{K-1})_{k-1} + \mathbf{R}_k \tag{56}$$

$$\mathbf{A}_k = \begin{bmatrix}\mathbf{A}_{k-1} \\ \mathbf{0}\end{bmatrix} - \begin{bmatrix}\tilde{\mathbf{A}}_{k-1}^* \\ \mathbf{I}_N\end{bmatrix}\left(\tilde{\mathbf{W}}_N^{(k-1)}\right)^{-T}\mathbf{H}_{k-1} \tag{57}$$

$$\mathbf{W}_N^{(k)} = \mathbf{W}_N^{(k-1)} - \mathbf{H}_{k-1}^H\left(\tilde{\mathbf{W}}_N^{(k-1)}\right)^{-T}\mathbf{H}_{k-1} \tag{58}$$

$$(\mathbf{R}_{K-1})_k = \begin{bmatrix}(\mathbf{R}_{K-1})_{k-1} \\ \mathbf{R}_k\end{bmatrix} \tag{59}$$

(3) Output: $\mathbf{A}_{K-1}$ and $\mathbf{W}_N^{(K-1)}$.

The detailed derivation of (56)–(58) is shown in Appendix B.

Since the term $\mathbf{R}^{-1} = \left(\sigma^2\mathbf{I}_{NK} + \mathbf{D}\Gamma\mathbf{D}^H\right)^{-1}$ is involved in the procedures of all the three MSBL-STAP algorithms mentioned above, this rapid way of calculating $\mathbf{R}^{-1}$ is applicable to all of them.

Let $\boldsymbol{\varepsilon}$ denote the vector which is made up of all the elements on the diagonal of the matrix $\boldsymbol{\Sigma}$ given in (8), i.e.,

$$\boldsymbol{\varepsilon} = diag(\boldsymbol{\Sigma}) \tag{60}$$

Then, based on the G-S factorization of $\mathbf{R}^{-1}$ in (52), we can efficiently calculate the vector $\boldsymbol{\varepsilon}$ given in (60) and the mean matrix $\boldsymbol{\mu}$ given in (9) in the iterative process of the MSBL-STAP algorithms. Firstly, we give the efficient way to compute $\boldsymbol{\varepsilon}$.

Let

$$\mathbf{Z} = \mathbf{D}^H\mathbf{R}^{-1}\mathbf{D} \tag{61}$$

Then, according to (8) and (61), we can rewrite the covariance matrix $\mathbf{\Sigma}$ as

$$\mathbf{\Sigma} = \mathbf{\Gamma} - \mathbf{\Gamma Z \Gamma} \tag{62}$$

During the iteration process of the MSBL-STAP algorithms, in fact, we only use these elements on the diagonal of the covariance matrix $\mathbf{\Sigma}$, i.e., we only need to compute the vector $\boldsymbol{\varepsilon}$ given in (60). As a result, we only need to obtain these values on the diagonal of $\mathbf{Z}$. Let $\mathbf{z} = \left[z_0, \ldots, z_{N_s-1}, \ldots, z_{(K_d-1)N_s}, \ldots, z_{K_dN_s-1}\right]^T$, where $\mathbf{z}$ is a vector consists of all the elements on the diagonal of the matrix $\mathbf{Z}$. Utilize the structure of the matrix $\mathbf{D}$, the $(kN+n+1)$th element of $\mathbf{z}$ can be written as

$$\begin{aligned} \mathbf{z}_{kN+n+1} &= \mathbf{D}^H(\omega_k, \omega_n)\mathbf{R}^{-1}\mathbf{D}(\omega_k, \omega_n) \\ &= \sum_{m_1=-K+1}^{K-1} \sum_{m_2=-N+1}^{N-1} c_{m_1,m_2} e^{-jm_2\omega_n} e^{-jm_1\omega_k} \\ &= \sum_{m_1=-K+1}^{K-1} \sum_{m_2=-N+1}^{N-1} c_{m_1,m_2} e^{-jm_2 n/N_s} e^{-jm_1 k/K_d} \end{aligned} \tag{63}$$

where $c_{m_1,m_2}$ is the sum of all elements on the $m_2$th main diagonal of all block matrices on the $m_1$th main diagonal of the HTBT matrix $\mathbf{R}^{-1}$. In addition, we can write $\{c_{m_1,m_2}\}_{m_1=\text{-}K+1,\ldots,0}^{m_2=-N+1,\ldots,N-1}$ as

$$\mathbf{c} = \begin{bmatrix} \left[c_{-K+1,-N+1}, c_{-K+1,-N+2}, \ldots, c_{-K+1,N-1}\right]^T \\ \left[c_{-K+2,-N+1}, c_{-K+2,-N+2}, \ldots, c_{-K+2,N-1}\right]^T \\ \vdots \\ \left[c_{0,-N+1}, c_{0,-N+2}, \ldots, c_{0,N-1}\right]^T \end{bmatrix} \tag{64}$$

By utilizing the G-S factorization of $\mathbf{R}^{-1}$ in (52), $\mathbf{c}$ can be represented by

$$\mathbf{c} = \sum_{n=0}^{N-1} \mathcal{L}_{K,2N-1}\left(\overline{\mathbf{T}}_n, \mathbf{J}_{K,2N-1}\right)\mathbf{t}_n^* - \mathcal{L}_{K,2N-1}\left(\overline{\mathbf{P}}_n, \mathbf{J}_{K,2N-1}\right)\mathbf{p}_n^* \tag{65}$$

where

$$\overline{\mathbf{T}}_n = \begin{bmatrix} \mathcal{L}_{N,2N-1}\left(\bar{\tilde{\mathbf{t}}}_n^{K-1}, \mathbf{J}_{2N-1}\right) \\ \mathcal{L}_{N,2N-1}\left(2\bar{\tilde{\mathbf{t}}}_n^{K-2}, \mathbf{J}_{2N-1}\right) \\ \vdots \\ \mathcal{L}_{N,2N-1}\left(K\bar{\tilde{\mathbf{t}}}_n^{0}, \mathbf{J}_{2N-1}\right) \end{bmatrix}_{(2N-1)K\times N} \tag{66}$$

$$\overline{\mathbf{P}}_n = \begin{bmatrix} \mathcal{L}_{N,2N-1}\left(\bar{\tilde{\mathbf{p}}}_n^{K-1}, \mathbf{J}_{2N-1}\right) \\ \mathcal{L}_{N,2N-1}\left(2\bar{\tilde{\mathbf{p}}}_n^{K-2}, \mathbf{J}_{2N-1}\right) \\ \vdots \\ \mathcal{L}_{N,2N-1}\left(K\bar{\tilde{\mathbf{p}}}_n^{0}, \mathbf{J}_{2N-1}\right) \end{bmatrix}_{(2N-1)K\times N} \tag{67}$$

where

$$\mathcal{L}_{N,2N-1}\left(\bar{\tilde{\mathbf{t}}}_n^{K-1}, \mathbf{J}_{2N-1}\right) = \left[\bar{\tilde{\mathbf{t}}}_n^{K-1}, \mathbf{J}_{2N-1}\bar{\tilde{\mathbf{t}}}_n^{K-1}, \ldots, \left(\mathbf{J}_{2N-1}\right)^{N-1}\bar{\tilde{\mathbf{t}}}_n^{K-1}\right] \tag{68}$$

$$\mathcal{L}_{N,2N-1}\left(\bar{\tilde{\mathbf{p}}}_n^{K-1}, \mathbf{J}_{2N-1}\right) = \left[\bar{\tilde{\mathbf{p}}}_n^{K-1}, \mathbf{J}_{2N-1}\bar{\tilde{\mathbf{p}}}_n^{K-1}, \ldots, \left(\mathbf{J}_{2N-1}\right)^{N-1}\bar{\tilde{\mathbf{p}}}_n^{K-1}\right] \tag{69}$$

and

$$\bar{\tilde{\mathbf{t}}}_n^k = \begin{bmatrix} \tilde{\mathbf{t}}_n^k \\ \mathbf{0} \end{bmatrix}_{2N-1} \tag{70}$$

$$\overline{\tilde{\mathbf{p}}}_n^k = \begin{bmatrix} \tilde{\mathbf{p}}_n^k \\ \mathbf{0} \end{bmatrix}_{2N-1} \tag{71}$$

where $\tilde{\mathbf{t}}_n^k = \mathbf{S}_N \mathbf{t}_n^k$ and $\tilde{\mathbf{p}}_n^k = \mathbf{S}_N \mathbf{p}_n^k$, $k = 0, 1, \ldots, K-1$, $\mathbf{t}_n^k$ is the $k$th block vector of $\mathbf{t}_n$ and $\mathbf{p}_n^k$ the $k$th block vector of $\mathbf{p}_n$.

Since D is the Kronecker product of two Fourier matrices and $\mathbf{R}^{-1}$ is a TBT matrix, the evaluation of $\mathbf{z}_{kN+n+1} = \mathbf{D}^H(\omega_k, \omega_n)\mathbf{R}^{-1}\mathbf{D}(\omega_k, \omega_n)$ can be transformed to calculate the coefficients of a bivariate polynomial on the unit sphere [56,57], and these polynomial coefficients can be computed using (65), which is a summation of the TBT matrix-vector products. In addition, the 2-D convolution can be utilized to obtain the summation of the TBT matrix-vector products. For STAP, by using FFT and IFFT, the 2-D convolution in the spatial-temporal domain can be transformed into a dot product in the beam-Doppler domain; thus, we can conclude that $\mathbf{c}$ can be efficiently calculated using 2-D FFT and IFFT. Once we obtain the matrix $\mathbf{c}$, we can obtain $\mathbf{z}_{kN+n+1}$ by performing $K_d$- and $N_s$-point 2-D FFT on the polynomial coefficients $c_{m_1,m_2}$. Since $\mathbf{R}^{-1}$ is a Hermitian matrix, we have $c_{-m_1,-m_2} = c^*_{m_1,m_2}$, where $m_1 = -K+1, \ldots, 0$ and $m_2 = -N+1, \ldots, N-1$. Then, given $\{c_{m_1,m_2}\}_{m_1=-K+1,\ldots,K-1}^{m_2=-N+1,\ldots,N-1}$, we have

$$\overline{\mathbf{Z}} = \mathcal{F}_{2-D}(\mathbf{C}) \tag{72}$$

where

$$\overline{\mathbf{Z}} = \begin{bmatrix} z_0 & \cdots & z_{(K_d-1)N_s} \\ \vdots & \ddots & \vdots \\ z_{N_s-1} & \cdots & z_{K_dN_s-1} \end{bmatrix} \tag{73}$$

$$\mathbf{C} = \begin{bmatrix} \mathbf{C}_0 & \mathbf{0} & \mathbf{C}_2 \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{C}_1 & \mathbf{0} & \mathbf{C}_3 \end{bmatrix} \tag{74}$$

$$\mathbf{C}_0 = \begin{bmatrix} c_{0,0} & \cdots & c_{K-1,0} \\ \vdots & \ddots & \vdots \\ c_{0,N-1} & \cdots & c_{K-1,N-1} \end{bmatrix} \tag{75}$$

$$\mathbf{C}_1 = \begin{bmatrix} c_{0,-N+1} & \cdots & c_{K-1,-N+1} \\ \vdots & \ddots & \vdots \\ c_{0,-1} & \cdots & c_{K-1,-1} \end{bmatrix} \tag{76}$$

$$\mathbf{C}_2 = \begin{bmatrix} c_{-K+1,0} & \cdots & c_{-1,0} \\ \vdots & \ddots & \vdots \\ c_{-K+1,N-1} & \cdots & c_{-1,N-1} \end{bmatrix} \tag{77}$$

$$\mathbf{C}_3 = \begin{bmatrix} c_{-K+1,-N+1} & \cdots & c_{-1,-N+1} \\ \vdots & \ddots & \vdots \\ c_{-K+1,-1} & \cdots & c_{-1,-1} \end{bmatrix} \tag{78}$$

Since the update of the covariance matrix $\mathbf{\Sigma}$ is not involved in the procedures of the MFCSBL-STAP algorithm, this part only suitable for the basic MSBL-STAP algorithm and the IR$\ell_{2,1}$-MSBL-STAP algorithm.

Then, we give an efficient way to compute the mean matrix $\boldsymbol{\mu}$, which can be divided into three steps

$$\mathbf{\Theta} = \mathbf{R}^{-1}\mathbf{Y} \tag{79}$$

$$\mathbf{\Phi} = \mathbf{D}^H\mathbf{\Theta} \tag{80}$$

$$\boldsymbol{\mu} = \mathbf{\Gamma}\mathbf{\Phi} \tag{81}$$

First, substituting (65) into (79), we get

$$\theta_l = \sum_{n=0}^{N-1} \left[ \mathcal{L}_{K,N}(\mathbf{t}_n, \mathbf{J}_{K,N}) \mathcal{L}_{K,N}^H(\mathbf{t}_n, \mathbf{J}_{K,N}) - \mathcal{L}_{K,N}(\mathbf{p}_n, \mathbf{J}_{K,N}) \mathcal{L}_{K,N}^H(\mathbf{p}_n, \mathbf{J}_{K,N}) \right] \mathbf{y}_l \tag{82}$$

where $\theta_l$ and $\mathbf{y}_l$ are the $l$th column vectors of matrices $\mathbf{\Theta}$ and $\mathbf{Y}$. From (82), we observe that each column of $\mathbf{\Theta}$ can be computed by the sum of Toeplitz-block matrix-vector products, which can be calculated by the 1-D convolution. In addition, we can efficiently calculate the 1-D convolution through FFT and IFFT. Since $\mathbf{D}$ is the product of two Fourier matrices, the $(kN + n + 1)$th element of the $\boldsymbol{\varphi}_l$ can be written as

$$\begin{aligned} \varphi_{l,kN+n+1} &= \sum_{m_1=0}^{K-1} \sum_{m_2=0}^{N-1} \theta_{l,m_1N+m_2+1} e^{jm_2\omega_n} e^{jm_1\omega_k} \\ &= \sum_{m_1=0}^{K-1} \sum_{m_2=0}^{N-1} \theta_{l,m_1N+m_2+1} e^{-jm_2n/N_s} e^{-jm_1k/K_d} \end{aligned} \tag{83}$$

where $\theta_{l,m_1N+m_2+1}$ is the $(m_1N + m_2 + 1)$th value of $\theta_l$. Thus, given $\left\{\theta_{l,m}\right\}_{m=1,\ldots,NK}^{l=1,\ldots,L}$, let

$$\overline{\theta}_l = \begin{bmatrix} \theta_{l,1} & \theta_{l,N+1} & \cdots & \theta_{l,(K-1)N+1} \\ \theta_{l,2} & \theta_{l,N+2} & \cdots & \theta_{l,(K-1)N+2} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{l,N} & \theta_{l,2N} & \cdots & \theta_{l,KN} \end{bmatrix} \tag{84}$$

According to (83) and the definition of IFFT, we observe that $\varphi_{l,kN+n+1}$ can be efficiently calculated by using $K$- and $N$-points 2-D IFFT, i.e.,

$$\overline{\boldsymbol{\varphi}}_l = \mathcal{IF}_{2-D}\left(\overline{\theta}_l\right) \tag{85}$$

where

$$\overline{\boldsymbol{\varphi}}_l = \begin{bmatrix} \varphi_{l,1} & \varphi_{l,N+1} & \cdots & \varphi_{l,(K-1)N+1} \\ \varphi_{l,2} & \varphi_{l,N+2} & \cdots & \varphi_{l,(K-1)N+2} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{l,N} & \varphi_{l,2N} & \cdots & \varphi_{l,KN} \end{bmatrix} \tag{86}$$

Finally, $\boldsymbol{\mu}$ can be computed using (81). Since the update of the mean matrix $\boldsymbol{\mu}$ is involved in the procedures of all the three MSBL-STAP algorithms mentioned above; thus, this rapid way to calculate $\boldsymbol{\mu}$ is applicable to all of them.

We denote the proposed efficient implementation of the basic MSBL-STAP algorithm based on the G-S factorization as GS-MSBL-STAP. The procedures of the GS-MSBL-STAP algorithm are summarized as follows.

Step 1: Give the initial values $\boldsymbol{\gamma}^0 = \mathbf{1}$, $\left(\sigma^2\right)^0 = 1$

Step 2: Give the $\boldsymbol{\gamma}^t$ and $\left(\sigma^2\right)^t$, Using (19)–(24), obtain the first $N$ columns of the covariance matrix $\mathbf{R}^t$ by applying 2-D FFT, with $o\left(5K_sN_d \log_2(K_sN_d)\right)$ flops.

Step 3: Given the first $N$ columns of $\mathbf{R}^t$, compute the $\left(\mathbf{R}^{-1}\right)^t$ through 2-D L-D algorithm, with $o\left(8N^3\left(K^2 - K + 2\right)\right)$ flops.

Step 4: Utilizing (60)–(86), calculate the vector $\boldsymbol{\varepsilon}^t$ and the mean matrix $\boldsymbol{\mu}^t$ by applying 2-D FFT and IFFT, with $o\left(5NK^2 \log_2(NK)\right) + o\left(5LK_dN_s \log_2(K_dN_s)\right)$ flops.

Step 5: Update $\boldsymbol{\gamma}^{t+1}$ and $\left(\sigma^2\right)^{t+1}$ using (10) and (11).

Step 6: Repeat step 2 to step 5 until the predefined convergence criteria is satisfied.

Step 7: obtain the estimated angle-Doppler profile $\hat{\mathbf{X}}$ using (13)

Step 8: Compute the CNCM using (14)

Step 9: Compute the optimal STAP weight vector Using (15).

We denote the proposed efficient implementations of the MFCSBL-STAP algorithm and IR$\ell_{2,1}$-MSBL-STAP algorithm based on the G-S factorization GS-MFCSBL-STAP and GS-IR$\ell_{2,1}$-MSBL-STAP, respectively. Since they have almost the same procedures as the GS-MSBL-STAP, here we will not describe these two algorithms in detail for the sake of article brevity.

## 5. Numerical Simulation

In this section, numerical experiments based on simulated data and measured data are conducted to assess the computational efficiency, the clutter suppression performance, and the target detection performance of the proposed computationally efficient GS-based MSBL-STAP algorithms. The simulation parameters of the radar system are listed in Table 3. The dictionary resolution scales are set to be $\rho_s = 4$ and $\rho_d = 4$. The number of used training samples is 10. We use the metric of the signal to interference plus noise ratio (SINR) loss as a measure to assess the performance of the proposed algorithms, which is calculated by the ratio of the output SINR and the signal to noise ratio (SNR) achieved by a matched filter in a noise-only environment, i.e.,

$$\mathrm{SINR}_{\mathrm{loss}} = \frac{\sigma^2}{NK} \frac{\left|\mathbf{w}_{opt}^H \mathbf{v}_t\right|}{\mathbf{w}_{opt}^H \mathbf{R}_{ideal} \mathbf{w}_{opt}} \tag{87}$$

where $\mathbf{R}_{ideal}$ is the clairvoyant CNCM.

**Table 3.** Simulation parameters of the radar system.

| Parameter | Value |
|---|---|
| Bandwidth | 2.5 M |
| Wavelength | 0.3 m |
| Pulse repetition frequency | 2000 Hz |
| Platform velocity | 150 m/s |
| Platform height | 9 km |
| Element number | 8 |
| Pulse number | 8 |
| CNR | 40 dB |

### *5.1. Simulated Data*

First, we detail the computational complexity of the proposed GS-MSBL-STAP algorithm, GS-MFCSBL-STAP algorithm and GS-IR$\ell_{2,1}$-MSBL-STAP algorithm for a single iteration and compare them with the original MSBL-STAP algorithms and other classical SR-STAP algorithms, including MCVX-STAP [14], MOMP-STAP [12], MFOCUSS-STAP [20], MIAA-STAP [58], MSBL-STAP [36], MFCSBL-STAP [45] and IR$\ell_{2,1}$-MSBL-STAP [46]. The computational complexity is measured by the number of floating-point operations. For simplicity, the low-order terms are omitted. The results are given in Table 4, where the sparse level $r_s$ of the MOMP-STAP algorithm is set to be equal to the clutter rank.

**Table 4.** Computational complexity comparison.

| Algorithm | The Number of Floating-Point Operations for a Single Iteration |
|---|---|
| MCVX-STAP | $o\left(8(N_sK_dL)^3\right)$ |
| MOMP-STAP | $o\left(8NKN_sK_dL + 8r_s^3 + 16NKr_s^2 + 8NKLr_s\right)$ |
| MFOCUSS-STAP | $o\left(8NKN_sK_dL + 8(NK)^3 + 8NK(N_sK_d)^2 + 16(NK)^2N_sK_d\right)$ |
| MIAA-STAP | $o\left(8NKN_sK_dL + 8(NK)^3 + 32(NK)^2N_sK_d + 16NKN_sK_d\right)$ |
| MSBL-STAP | $o\left(8NKN_sK_dL + 8(NK)^3 + 32NK(N_sK_d)^2 + 24(NK)^2N_sK_d\right)$ |
| MFCSBL-STAP | $o\left(8NKN_sK_dL + 8(NK)^3 + 16NK(N_sK_d)^2 + 40(NK)^2(N_sK_d)\right)$ |
| IR$\ell_{2,1}$-MSBL-STAP | $o\left(8NKN_sK_dL + 8(NK)^3 + 32NK(N_sK_d)^2 + 32(NK)^2N_sK_d\right)$ |
| GS-MSBL-STAP | $o\left(8N^3\left(K^2-K+2\right) + 5(L+1)N_sK_d\log_2(N_sK_d) + 5(NK)^2\log_2(NK) + 8N_sK_dNKL\right)$ |
| GS-MFCSBL-STAP | $o\left(8N^3\left(K^2-K+2\right) + 5(L+1)N_sK_d\log_2(N_sK_d) + 8NKN_sK_dL + 24(NK)^2N_sK_d\right)$ |
| GS-IR$\ell_{2,1}$-MSBL-STAP | $o\left(8N^3\left(K^2-K+2\right) + 5(L+1)N_sK_d\log_2(N_sK_d) + 5(NK)^2\log_2(NK) + 8NKN_sK_dL + 8(NK)^2N_sK_d\right)$ |

In fact, during the process of the MSBL-STAP algorithms, for each iteration, the computational complexities are mainly related to the update of the posterior mean matrix $\boldsymbol{\mu}$, the posterior variance matrix $\boldsymbol{\Sigma}$ and the estimated clutter covariance matrix $\mathbf{R}$. We can easily observe that these three terms $\boldsymbol{\Sigma} = \boldsymbol{\Gamma} - \boldsymbol{\Gamma}\mathbf{D}^H\mathbf{R}^{-1}\mathbf{D}\boldsymbol{\Gamma}$, $\boldsymbol{\mu} = \boldsymbol{\Gamma}\mathbf{D}^H\mathbf{R}^{-1}\mathbf{Y}$ and $\mathbf{R} = \sigma^2\mathbf{I}_{NK} + \mathbf{D}\boldsymbol{\Gamma}\mathbf{D}^H$ contain large-scale matrix inversion operations and large-scale matrix multiplication operations. Traditional MSBL-STAP algorithms directly compute these three terms. Thus, they suffer from high computational complexity. In the process of the proposed GS-based MSBL-STAP algorithms, utilizing the structure of the dictionary matrix $\mathbf{D}$ and the G-S factorization of the $\mathbf{R}^{-1}$, all the three terms can be rapidly computed by using 2-D FFT and IFFT. Thus, compared with the Traditional MSBL-STAP algorithms, the computational complexities of the proposed GS-based MSBL-STAP algorithms are significantly reduced. In Table 4, the number of floating-point operations of different SR-STAP algorithms for a single iteration are listed, and it can be observed that the MCVX-STAP algorithm has a high degree of computational complexity, which grows rapidly with the product of the number of training samples and the number of the atoms in the space–time dictionary matrix and the MOMP-STAP algorithm has the lowest computational complexity among the SR-STAP algorithms. Compared with the MSBL-STAP algorithm, the MFCSBL-STAP algorithm and the IR$\ell_{2,1}$-MSBL-STAP algorithm, it can also be observed that the proposed GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm and GS-IR$\ell_{2,1}$-MSBL-STAP algorithm have lower computational complexities.

Figure 1 provides a more direct illustration of the computational complexities of the different SR-STAP algorithms. It shows the number of floating-point operations for a single iteration as the function of the number of the system DOFs. From Figure 1, it can be intuitively observed that the proposed GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm have lower computational complexities than the CVX-STAP algorithm, the MFOCUSS-STAP algorithm, the MSBL-STAP algorithm, the MFCSBL-STAP algorithm, and the IR$\ell_{2,1}$-MSBL-STAP algorithm. Additionally, with the growth in the number of system DOFs, it can be found that the proposed computationally efficient GS-based MSBL-STAP algorithms have lower growth rates than the other SR-STAP algorithms. Table 5 presents the computational complexities of various SR-STAP algorithms under different system DOFs. In Table 5, it can be observed that when the number of system DOFs is 128, compared with the MSBL-STAP algorithm, the MFCSBL-STAP algorithm and the IR$\ell_{2,1}$-MSBL-STAP algorithm, using the proposed GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm can reduce the computational load by about one order of magnitude. Meanwhile, when the number of system DOFs is 512, using the proposed GS-based MSBL-STAP algorithms can reduce the computational load by about two orders of magnitude.

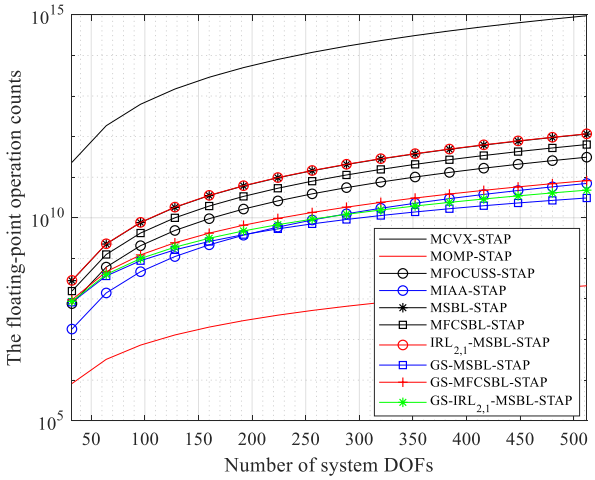


**Figure 1.** Computational complexities of different SR-STAP algorithms.

**Table 5.** Computational complexities of various SR-STAP algorithms under different system DOFs.

| Algorithm \ System DOFs | The Number of Floating-Operations for Single Iteration | 128 | 256 | 512 |
|---|---|---|---|---|
| MCVX-STAP | | $1.484 \times 10^{13}$ | $1.187 \times 10^{14}$ | $9.499 \times 10^{14}$ |
| MOMP-STAP | | $1.296 \times 10^{7}$ | $5.206 \times 10^{7}$ | $2.111 \times 10^{8}$ |
| MFOCUSS-STAP | | $4.861 \times 10^{9}$ | $3.884 \times 10^{10}$ | $3.105 \times 10^{11}$ |
| MIAA-STAP | | $1.107 \times 10^{9}$ | $8.791 \times 10^{9}$ | $7.006 \times 10^{10}$ |
| MSBL-STAP | | $1.802 \times 10^{10}$ | $1.441 \times 10^{11}$ | $1.152 \times 10^{12}$ |
| MFCSBL-STAP | | $9.962 \times 10^{9}$ | $7.964 \times 10^{10}$ | $6.369 \times 10^{11}$ |
| IR$\ell_{2,1}$-MSBL-STAP | | $1.828 \times 10^{10}$ | $1.462 \times 10^{11}$ | $1.169 \times 10^{12}$ |
| GS-MSBL-STAP | | $1.619 \times 10^{9}$ | $7.072 \times 10^{9}$ | $3.070 \times 10^{10}$ |
| GS-MFCSBL-STAP | | $2.424 \times 10^{9}$ | $1.351 \times 10^{10}$ | $8.223 \times 10^{10}$ |
| GS-IR$\ell_{2,1}$-MSBL-STAP | | $1.888 \times 10^{9}$ | $9.220 \times 10^{9}$ | $4.788 \times 10^{10}$ |

The cost function $C = \ln|\mathbf{R}_{c+n}| + \mathrm{Tr}\left(\mathbf{R}_{c+n}^{-1}\mathbf{R}_{ideal}\right)$ can be used to evaluate the convergence performance of different SR-STAP algorithms [45]. Figure 2 plots the value of cost function versus the number of iterations curves of different SR-STAP algorithms. We find that the IR$\ell_{2,1}$-MSBL-STAP algorithm and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm converge to their steady-state values after about 12 iterations, and the MFCSBL-STAP algorithm and the GS-MFCSBL-STAP algorithm converge to their steady-state values after about 15 iterations. The MIAA-STAP algorithm and the MFOCUSS-STAP algorithm converge to their steady-state values after about 20 iterations and 50 iterations, respectively. The MSBL-STAP algorithm and the GS-MSBL-STAP algorithm converge very slowly, requiring more than 200 iterations to reach their steady-state values. We also find that the proposed computationally efficiently GS-based MSBL-STAP algorithms do not change the convergence of the original MSBL-STAP algorithms. In Table 6, the average running times of different SR-STAP algorithms are compared. The results were obtained using MATLAB 2018b and a computer with Intel(R) Xeon(R) E5-2620 CPU @ 2.40GHz 2.39 GHz. According to Figure 2, we set the number of iterations of the MFOCUSS-STAP algorithm, the MIAA-STAP algorithm, the MSBL-STAP algorithm, the MFCSBL-STAP, the IR$\ell_{2,1}$-MSBL-STAP algorithm, the GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm as 50, 20, 200, 15, 12, 200, 15 and 12, respectively. From Table 6, it can be determined that the average running times of the proposed computationally efficient GS-based MSBL-STAP algorithms are far shorter than the original MSBL-STAP algorithms, thus validating the computational efficiency of our proposed algorithms.

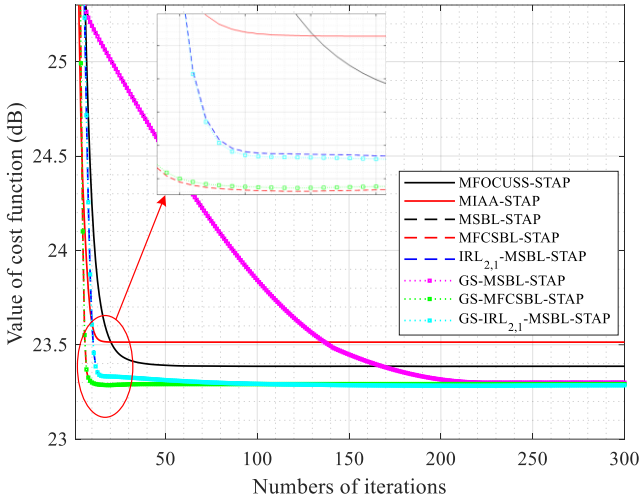


**Figure 2.** Value of cost function versus the number of iterations.

**Table 6.** Average running time comparison.

| Algorithm | Running Time |
|---|---|
| MCVX-STAP | 900.4931 s |
| MOMP-STAP | 0.0254 s |
| MFOCUSS-STAP | 3.8556 s |
| MIAA-STAP | 0.7614 s |
| MSBL-STAP | 15.1402 s |
| MFCSBL-STAP | 1.4835 s |
| IR$\ell_{2,1}$-MSBL-STAP | 1.8533 s |
| GS-MSBL-STAP | 1.3409 s |
| GS-MFCSBL-STAP | 0.3610 s |
| GS-IR$\ell_{2,1}$-MSBL-STAP | 0.1914 s |

Next, we compare the clutter suppression performance of different SR-STAP algorithms. The specific simulation parameters of different SR-STAP are set as follows. The diagonal loading factor for the LSMI-STAP algorithm is 10 dB to the noise power [56]. The iteration termination thresholds of the MOMP-STAP algorithm [12], the MFOCUSS-STAP algorithm [20], the MIAA-STAP algorithm [58], the MSBL-STAP algorithm [36], the MFCSBL-STAP algorithm [45], the IR$\ell_{2,1}$-MSBL-STAP algorithm [46], the GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm are all set to the same value, i.e., $\delta = 0.0001$. The regularization parameter for the MFOCUSS-STAP algorithm is set as $p = 0.8$. Figure 3 plots the recovered clutter capon spectrums of the different SR-STAP algorithms. From Figure 3b–d, it can be observed that the clutter spectrum of the LSMI-STAP is very poor, the clutter spectrum of the MOMP-STAP is discontinuous, and the clutter spectrum of MFOCUSS-STAP has a slight spectrum expansion. The reason for this is that the CNCM cannot be well estimated using LSMI-STAP algorithms when the number of the training samples is small, and the steering vectors selected by MOMP-STAP and MFOCUSS-STAP cannot precisely span the true clutter subspace due to the limitations of the algorithm itself. From Figure 3e, it can be observed that the clutter spectrum of the MIAA-STAP has a high level of noise; this is because the space–time dictionary matrix $\mathbf{D}$ is not an orthogonal matrix; rather, the atoms in the space–time dictionary $\mathbf{D}$ are usually highly coherent. From Figure 3f–k, it can also be observed that the recovered clutter spectrums of the MSBL-STAP, the MFCSBL-STAP, the IR$\ell_{2,1}$-MSBL-STAP, the GS-MSBL-STAP, the GS-MFCSBL-STAP and the GS-IR$\ell_{2,1}$-MSBL-STAP are very close to the optimal spectrum. This shows that the MSBL-STAP algorithms can achieve superior clutter suppression performance, and the GS-based MSBL-STAP algorithms proposed by us can still achieve superior clutter suppression performance with less computational complexity.

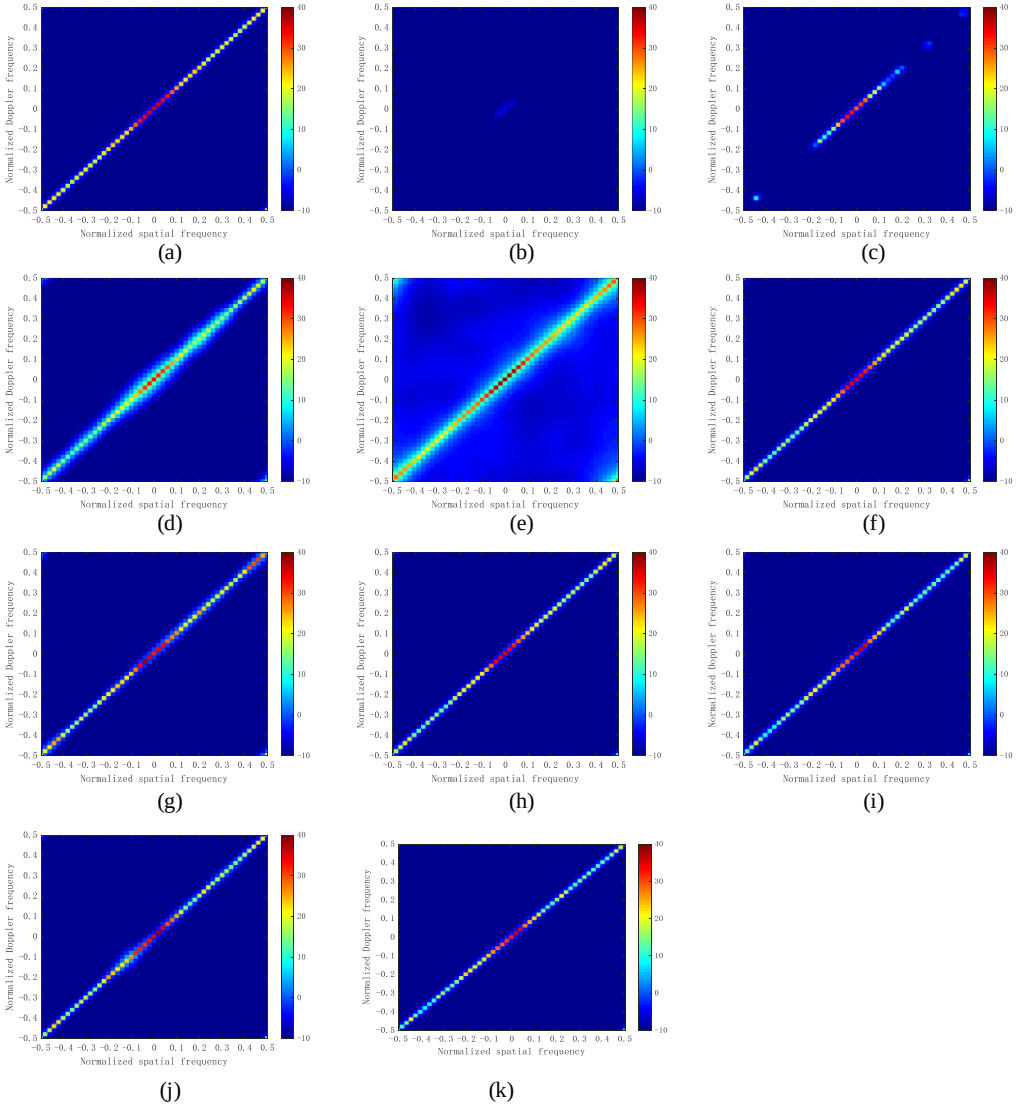


**Figure 3.** Clutter capon spectrums of different SR-STAP algorithms. (**a**) OPT; (**b**) LSMI-STAP; (**c**) MOMP-STAP; (**d**) MFCOUSS-STAP; (**e**) MIAA-STAP; (**f**) MSBL-STAP; (**g**) MFCSBL-STAP; (**h**) IR$\ell_{2,1}$-MSBL-STAP; (**i**) GS-MSBL-STAP; (**j**) GS-MFCSBL-STAP; (**k**) GS-IR$\ell_{2,1}$-MSBL-STAP.

Figure 4 depicts the SINR loss curves of different SR-STAP algorithms. All simulation results for SINR loss are averaged over 100 independent Monte Carlo trails. From Figure 4, it can be observed that the GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm achieve the same performance as the MSBL-STAP algorithm, the MFCSBL-STAP algorithm, and the IR$\ell_{2,1}$-MSBL-STAP algorithm. This further indicates that the proposed computationally efficient GS-based MSBL-STAP algorithms do not change the clutter suppression performance of the original MSBL-STAP algorithms.

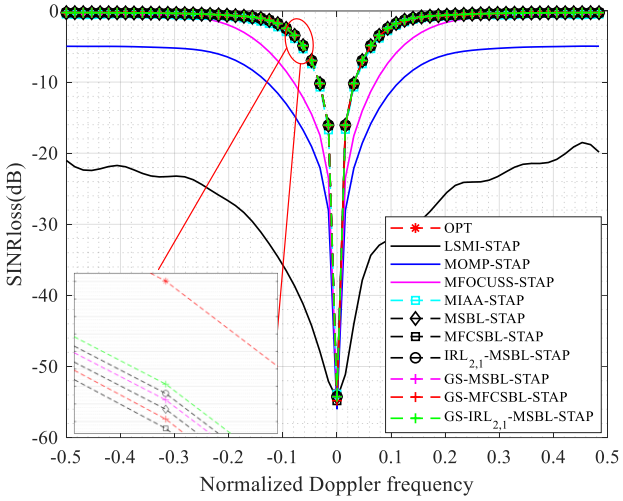


**Figure 4.** SINR loss comparison of different SR-STAP algorithms.

Then, we evaluate the target detection performance of different SR-STAP algorithms by the probability of detection probability (PD) versus SNR curves and the receive operating character (ROC) (i.e., PD versus the probability of false alarm (PFA)) curves, which are obtained by using the adaptive matched filter (AMF) detector [59]. The detection threshold and the probability of detection estimates are based on $10^4$ samples. Besides, all the PD versus SNR curves and ROC curves are acquired based on 1000 independent Monte Carlo trails.

The PD versus SNR curves of different SR-STAP algorithms are depicted in Figure 5. the PFA is set as $10^{-3}$, the target is assumed to be in the main beam direction with a Doppler frequency of 0.1 in Figure 5a and 0.3 in Figure 5b. As depicted in Figure 5a,b, the target detection performance of the MSBL-STAP algorithm, the MFCSBL-STAP, the IR$\ell_{2,1}$-MSBL-STAP algorithm, the GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm are close to the optimal performance, which indicates that the MSBL-STAP algorithms have superior target detection performance whether in the mainlobe region ($f_{dt} = 0.1$) or in the sidelobe region ($f_{dt} = 0.3$). We find that the target detection performance of MIAA-STAP algorithm is slightly worse than the MSBL-STAP algorithms, the reason for this is that the recovered clutter spectrum of the MIAA-STAP has a relatively high level of noise. By comparing the PD versus SNR curves of MFCOUSS-STAP algorithm, the MOMP-STAP algorithm, and the LSMI-STAP algorithm in Figure 5a,b, we find that these algorithms have worse target detection performance in the mainlobe region ($f_{dt} = 0.1$), which indicates that these algorithms have poor ability to detect slow moving targets. From Figure 5a,b, it can also be observed that the GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm have the same performance as the MSBL-STAP algorithm, the MFCSBL-STAP algorithm and the IR$\ell_{2,1}$-MSBL-STAP algorithm, which shows that the proposed computationally efficient GS-based MSBL-STAP algorithms do not change the target detection performance of the original MSBL-STAP algorithms.

The ROC curves of different SR-STAP algorithms are depicted in Figure 6. The SNR is set as $-10$ dB in Figure 6a,c and $-2$dB in Figure 6b,d. The target is assumed to be in the main beam direction with a Doppler frequency of 0.1 in Figure 6a,b and 0.3 in Figure 6c,d. As depicted in Figure 6a–d, the MIAA-STAP algorithm, the MSBL-STAP algorithm, the MFCSBL-STAP, the IR$\ell_{2,1}$-MSBL-STAP algorithm, the GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm can achieve superior target detection performance whether for fast moving targets (see Figure 6a,b) or for slow-moving targets (see Figure 6c,d). It can also be observed that the MFCOUSS-STAP algorithm, the MOMP-STAP algorithm, and the LSMI-STAP algorithm have worse target detection performance than other SR-STAP algorithms. Additionally, the target detection performance increases to various degrees with the improvement of the SNR for all SR-STAP

algorithms. Similarly, it can be seen from Figure 6 that the proposed computationally efficient GS-based MSBL-STAP algorithms do not change the target performance of the original MSBL-STAP algorithms.

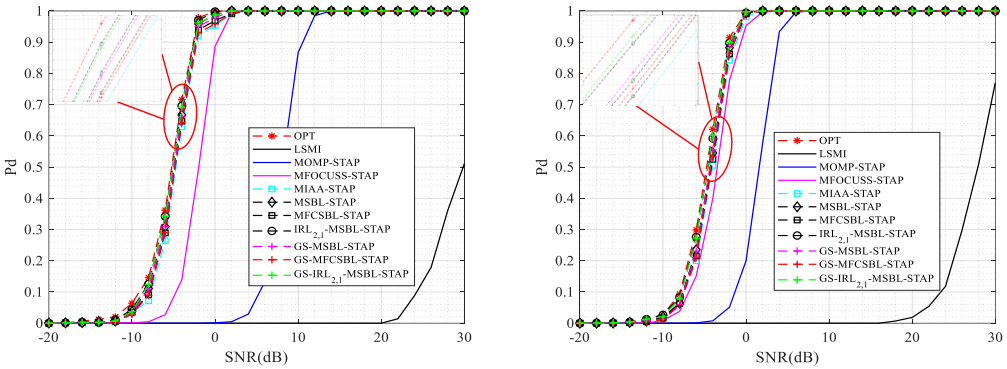


**Figure 5.** PD versus SNR curves of different SR-STAP algorithms. (**a**) the normalized target Doppler frequency $f_{dt} = 0.1$; (**b**) the normalized target Doppler frequency $f_{dt} = 0.3$.

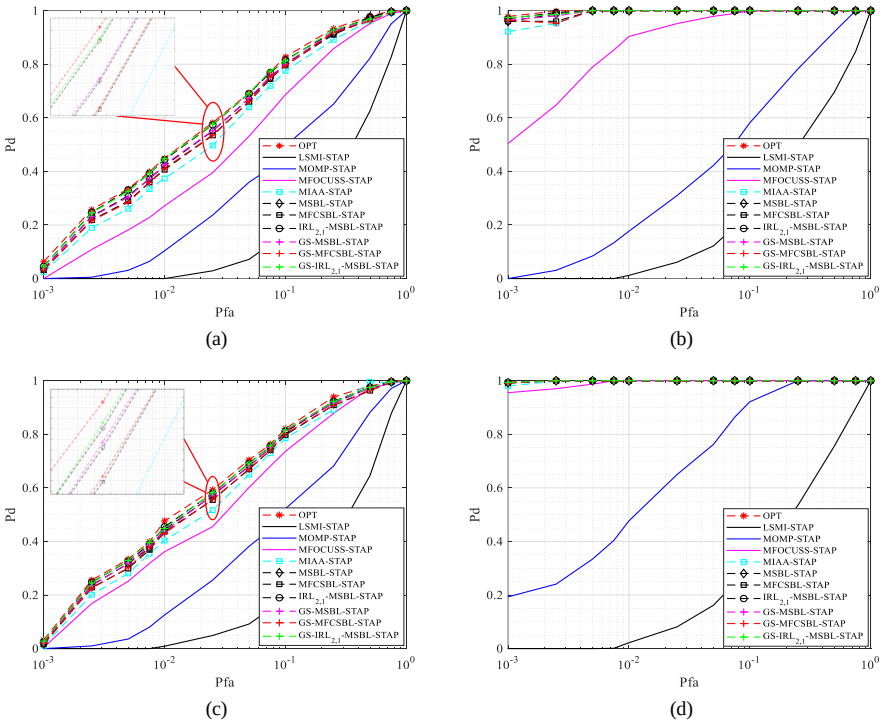


**Figure 6.** ROC curves of different SR-STAP algorithms. (**a**) $f_{dt}$= 0.1, SNR = − 10 dB; (**b**) $f_{dt}$= 0.1, SNR = − 2 dB; (**c**) $f_{dt}$= 0.3, SNR = − 10 dB; (**d**) $f_{dt}$= 0.3, SNR = − 2 dB.

Figure 7 depicts the average SINR loss versus the number of training samples. As shown in Figure 7, when the number of the used training samples is greater than 6, the MSBL-STAP algorithm, the MFCSBL-STAP, the IR$\ell_{2,1}$-MSBL-STAP algorithm, the GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm can achieve a near-optimal performance. For the MIAA-STAP algorithm, at least eight training samples are needed to achieve near-optimal performance. For the MFCOUSS-STAP algorithm, the MOMP-STAP algorithm, and the LSMI-STAP algorithm, more training samples are needed to achieve a steady performance.

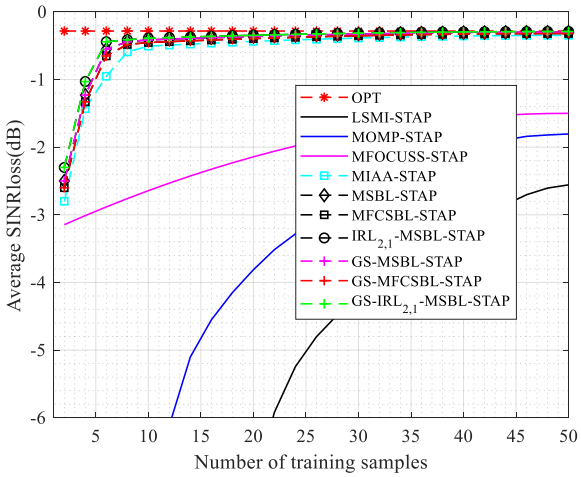


**Figure 7.** Average SINR loss versus the number of training samples.

### *5.2. Measured Data*

In this section, we apply the proposed GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP algorithm to the publicly available Mountain-Top set, i.e., t38pre01v1 CPI6 [60]. For this data set, the received antenna array consists of 14 elements and 16 coherent pulses are transmitted in a CPI. The PRF is 625 Hz and a 500 kHz linear frequency modulated pulse is used for transmitting. There are 403 range cells in this data file and the clutter is located around 245$^{\circ}$ relative to true north, and the target is located at 275$^{\circ}$ relative to true north. The target is located in the 147th range cell, with a normalized Doppler frequency of 0.25. The estimated clutter capon spectrum using all 403 training samples is given in Figure 8. Specifically, firstly, we use all 403 range cells to estimate the clutter-plus-noise covariance matrix; then, utilizing the general formula for capon spectrum estimation, the estimated capon spectrum of the mountain-top data can be obtained. From Figure 8, it can be observed that the mountain-top data have serious heterogeneity.

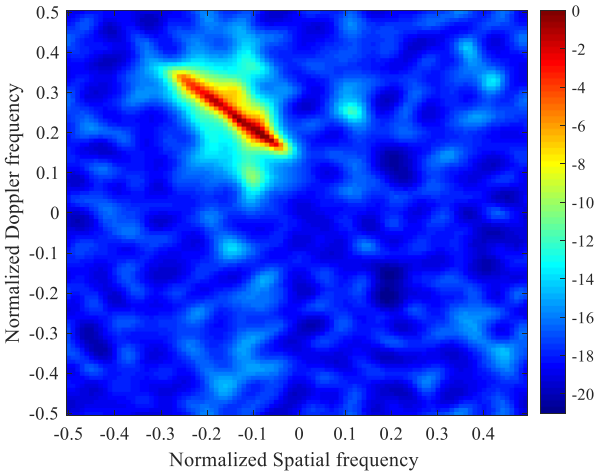


**Figure 8.** Estimated clutter capon spectrum with all 403 samples.

Table 7 shows the average running time of different SR-STAP methods with the measured data. From Table 7, it can be observed that, compared with the original MSBL-STAP algorithms, the average running time of the proposed GS-based MSBL-STAP algorithms is significantly reduced when processing the measured data. Figure 9 depicts the STAP output for the EFA algorithm, and different SR-STAP algorithms in range cells from 130 to 165, where the curves of the MSBL-STAP algorithm, the MFCSBL-STAP algorithm, the IR$\ell_{2,1}$-MSBL-STAP algorithm are omitted, since they are visually identical to those of the GS-MSBL-STAP algorithm, the GS-MFCSBL-STAP algorithm, and the GS-IR$\ell_{2,1}$-MSBL-STAP

algorithm, respectively. Ten snapshots out of 20 snapshots located next to the CUT are selected as the training data. From Figure 9, it can be observed that the target (located at 147th range cell) can be detected by all the SR-STAP algorithms even though only 10 snapshots are selected as training samples. However, since the CNCM cannot be accurately estimated, the traditional EFA algorithm cannot find the target. By comparing the STAP output of different SR-STAP algorithms, it can be observed that the proposed GS-MSBL-STAP algorithm, GS-MFCSBL-STAP algorithm, and GS-IR$\ell_{2,1}$-MSBL-STAP algorithm have better detection performance than the other SR-STAP algorithms. Moreover, since the proposed algorithms greatly reduce the computational complexity, they are more favorable for practical applications.

**Table 7.** Average running time of different SR-STAP methods with measured data.

| Algorithm | Running Time |
|---|---|
| MFOCUSS-STAP | 41.9740 s |
| MIAA-STAP | 2.3135 s |
| MSBL-STAP | 43.8430 s |
| MFCSBL-STAP | 4.7761 s |
| IR$\ell_{2,1}$-MSBL-STAP | 6.7693 s |
| GS-MSBL-STAP | 2.3971 s |
| GS-MFCSBL-STAP | 0.8380 s |
| GS-IR$\ell_{2,1}$-MSBL-STAP | 0.4277 s |

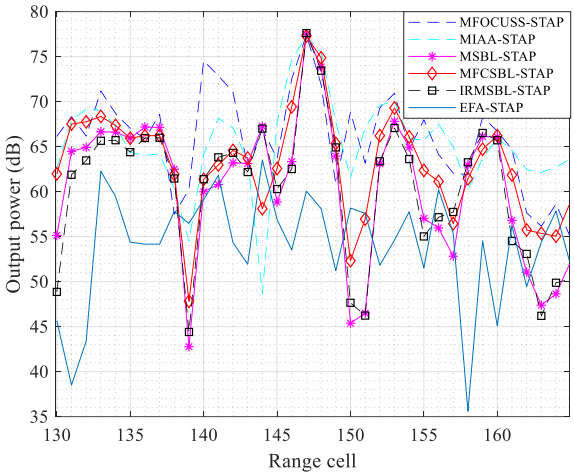


**Figure 9.** STAP output power against the range cell for different algorithms.

## 6. Conclusions

In this work, we developed several computationally efficient GS-based MSBL-STAP algorithms. Since the covariance matrix to be updated in the iterative process of the original MSBL-STAP algorithms is an HTBT matrix, the inverse of the covariance matrix can be decomposed using G-S factorization. Then, by exploiting the TBT/Toeplitz structural characteristics and the property whereby the space–time dictionary matrix **D** is the Kronecker product of two Fourier matrices, the computational complexity of the original MSBL-STAP algorithms can be significantly reduced by using the 2D-FFT/IFFT. The simulation results validate that the proposed efficient MSBL-STAP algorithms can significantly reduce the computational complexity while obtain superior clutter suppression performance and target detection performance. However, the efficient algorithms we proposed are only suitable for the case of ULA and a constant PRF with uniformly sampled spatial frequencies and Doppler frequencies. When these conditions are not met, the space–time dictionary matrix **D** is no longer the Kronecker product of two Fourier matrices, and the estimated covariance matrix in each iteration of the MSBL-STAP algorithms will not be an HTBT matrix; as a result, the efficient MSBL-STAP algorithms proposed in this article will no longer be

applicable. Thus, in our future work, extending the proposed efficient implementation of the original MSBL-STAP algorithms under other conditions is worthy of research.

**Author Contributions:** Conceptualization, K.L. and T.W.; investigation, K.L. and C.L.; methodology, K.L. and J.W.; project administration, T.W.; software, K.L.; supervision, J.W.; visualization, K.L.; writing—original draft, K.L.; writing—review and editing, W.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by National Key R&D Program of China, grant number 2021YFA1000400.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Appendix A

The proofs of (39)–(41) are given in Appendix A. By utilizing the property $\mathbf{S}_{(K-1)N}\mathbf{S}_{(K-1)N} = \mathbf{I}_{(K-1)N}$, we get

$$\begin{aligned} \mathbf{R}_{K-1,N}^{-1}\mathbf{S}_{(K-1)N} &= \mathbf{S}_{(K-1)N}\mathbf{S}_{(K-1)N}\mathbf{R}_{K-1,N}^{-1}\mathbf{S}_{(K-1)N} \\ &= \mathbf{S}_{(K-1)N}\mathbf{R}_{K-1,N}^{-T} \\ &= \mathbf{S}_{(K-1)N}\left(\mathbf{R}_{K-1,N}^{H}\right)^{-T} \\ &= \mathbf{S}_{(K-1)N}\mathbf{R}_{K-1,N}^{-*} \end{aligned} \tag{A1}$$

Substituting (38) and (39) into (35) and (36), we get

$$\begin{aligned} \mathbf{B}_{K-1}^{*} &= -\mathbf{R}_{K-1,N}^{-1}\tilde{\mathbf{R}}_{K-1}^{*} \\ &= -\mathbf{R}_{K-1,N}^{-1}\mathbf{S}_{(K-1)N}\mathbf{R}_{K-1}^{*}\mathbf{S}_{N} \\ &= -\mathbf{S}_{(K-1)N}\mathbf{R}_{K-1,N}^{-*}\mathbf{R}_{K-1}^{*}\mathbf{S}_{N} \\ &= \mathbf{S}_{(K-1)N}\mathbf{A}_{K-1}^{*}\mathbf{S}_{N} \\ &= \tilde{\mathbf{A}}_{K-1}^{*} \end{aligned} \tag{A2}$$

$$\begin{aligned} \mathbf{V}_{N} &= \mathbf{R}_{0} - \tilde{\mathbf{R}}_{K-1}^{T}\mathbf{R}_{K-1,N}^{-1}\tilde{\mathbf{R}}_{K-1}^{*} \\ &= \mathbf{R}_{0} - \mathbf{S}_{N}\mathbf{R}_{K-1}^{T}\mathbf{S}_{(K-1)N}\mathbf{R}_{K-1,N}^{-1}\mathbf{S}_{(K-1)N}\mathbf{R}_{K-1}^{*}\mathbf{S}_{N} \\ &= \mathbf{S}_{N}\mathbf{S}_{N}\mathbf{R}_{0}\mathbf{S}_{N}\mathbf{S}_{N} - \mathbf{S}_{N}\mathbf{R}_{K-1}^{T}\mathbf{S}_{(K-1)N}\mathbf{R}_{K-1,N}^{-1}\mathbf{S}_{(K-1)N}\mathbf{R}_{K-1}^{*}\mathbf{S}_{N} \\ &= \mathbf{S}_{N}\left(\mathbf{S}_{N}\mathbf{R}_{0}\mathbf{S}_{N} - \mathbf{R}_{K-1}^{T}\mathbf{R}_{K-1,N}^{-T}\mathbf{R}_{K-1}^{*}\right)\mathbf{S}_{N} \\ &= \mathbf{S}_{N}\mathbf{W}_{N}^{T}\mathbf{S}_{N} = \tilde{\mathbf{W}}_{N}^{T} \end{aligned} \tag{A3}$$

## Appendix B

The proofs of (56)–(58) are given in Appendix B. Given $\mathbf{A}_1$ and $\mathbf{W}_N^{(1)}$, according to (33) and (34), we get

$$\begin{aligned} &\mathbf{A}_k = -\left(\mathbf{R}_{K-1,N}^{-1}\right)_k^{-1}(\mathbf{R}_{K-1})_k \\ &= -\left\{\begin{bmatrix} \left(\mathbf{R}_{K-1,N}^{-1}\right)_{k-1}^{-1} & 0 \\ 0 & 0 \end{bmatrix} + \begin{bmatrix} \tilde{\mathbf{A}}_{k-1}^{*} \\ \mathbf{I}_N \end{bmatrix}\left(\tilde{\mathbf{W}}_N^{(k-1)}\right)^{-T}\begin{bmatrix} \tilde{\mathbf{A}}_{k-1}^{T} & \mathbf{I}_N \end{bmatrix}\right\}\begin{bmatrix} (\mathbf{R}_{K-1})_{k-1} \\ \mathbf{R}_k \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}_{k-1} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{A}}_{k-1}^{*} \\ \mathbf{I}_N \end{bmatrix}\left(\tilde{\mathbf{W}}_N^{(k-1)}\right)^{-T}\left(\tilde{\mathbf{A}}_{k-1}^{T}(\mathbf{R}_{K-1})_{k-1} + \mathbf{R}_k\right) \\ &= \begin{bmatrix} \mathbf{A}_{k-1} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{A}}_{k-1}^{*} \\ \mathbf{I}_N \end{bmatrix}\left(\tilde{\mathbf{W}}_N^{(k-1)}\right)^{-T}\mathbf{H}_{k-1} \end{aligned} \tag{A4}$$

$$
\begin{aligned}
&\mathbf{W}_N^{(k)} - \mathbf{W}_N^{(k-1)} \\
&= -(\mathbf{R}_{K-1})_k^H \left(\mathbf{R}_{K-1,N}^{-1}\right)_k^{-1} (\mathbf{R}_{K-1})_k + (\mathbf{R}_{K-1})_{k-1}^H \left(\mathbf{R}_{K-1,N}^{-1}\right)_{k-1}^{-1} (\mathbf{R}_{K-1})_{k-1} \\
&= (\mathbf{R}_{K-1})_k^H \mathbf{A}_k - (\mathbf{R}_{K-1})_{k-1}^H \mathbf{A}_{k-1} \\
&= \left[(\mathbf{R}_{K-1})_{k-1}^H \mathbf{R}_k\right] \left\{ \begin{bmatrix} \mathbf{A}_{k-1} \\ \mathbf{0} \end{bmatrix} - \begin{bmatrix} \tilde{\mathbf{A}}_{k-1}^* \\ \mathbf{I}_N \end{bmatrix} \left(\tilde{\mathbf{W}}_N^{(k-1)}\right)^{-T} \left(\tilde{\mathbf{A}}_{k-1}^T (\mathbf{R}_{K-1})_{k-1} + \mathbf{R}_k\right) \right\} - (\mathbf{R}_{K-1})_{k-1}^H \mathbf{A}_{k-1} \\
&= -\left((\mathbf{R}_{K-1})_{k-1}^H \tilde{\mathbf{A}}_{k-1}^* + \mathbf{R}_k^*\right) \left(\tilde{\mathbf{W}}_N^{(k-1)}\right)^{-T} \left(\tilde{\mathbf{A}}_{k-1}^T (\mathbf{R}_{K-1})_{k-1} + \mathbf{R}_k\right) \\
&= -\mathbf{H}_{k-1}^H \left(\tilde{\mathbf{W}}_N^{(k-1)}\right)^{-T} \mathbf{H}_{k-1}
\end{aligned} \tag{A5}
$$

$$
\mathbf{W}_N^{(k)} = \mathbf{W}_N^{(k-1)} - \mathbf{H}_{k-1}^H \left(\tilde{\mathbf{W}}_N^{(k-1)}\right)^{-T} \mathbf{H}_{k-1} \tag{A6}
$$

where

$$
\mathbf{H}_{k-1} = \tilde{\mathbf{A}}_{k-1}^T (\mathbf{R}_{K-1})_{k-1} + \mathbf{R}_k \tag{A7}
$$

## References


1. Ward, J. *Space-Time Adaptive Processing for Airborne Radar*; MIT Lincoln Laboratory: Lexington, KY, USA, 1994.
2. Klemm, R. *Principles of Space-Time Adaptive Processing*; The Institution of Electrical Engineers: London, UK, 2002.
3. Guerci, J.R. *Space-Time Adaptive Processing for Radar*; Artech House: Norwood, MA, USA, 2003.
4. Brennan, L.E.; Mallett, J.D.; Reed, I.S. Theory of Adaptive Radar. *IEEE Trans. Aerosp. Electron. Syst.* **1973**, *9*, 237–251. [CrossRef]
5. Reed, I.S.; Mallett, J.D.; Brennan, L.E. Rapid Convergence Rate in Adaptive Arrays. *IEEE Trans. Aerosp. Electron. Syst.* **1974**, *10*, 853–863. [CrossRef]
6. Baraniuk, R.G. Compressive sensing. *IEEE Signal Proc. Mag.* **2007**, *24*, 118–121. [CrossRef]
7. Trzasko, J.; Manduca, A. Relaxed Conditions for Sparse Signal Recovery With General Concave Priors. *IEEE Trans. Signal Process.* **2009**, *57*, 4347–4354. [CrossRef]
8. Davies, M.E.; Gribonval, R. Restricted Isometry Constants where $\ell$p sparse recovery can fail for $0 < p \leq 1$. *IEEE Trans. Inf. Theory* **2009**, *55*, 2203–2214.
9. David, M.E.; Eldar, Y.C. Rank awareness in joint sparse recovery. *IEEE Trans. Inf. Theory* **2012**, *58*, 1135–1146.
10. Mallat, S.G.; Zhang, Z. Matching pursuits with time-frequency dictionaries. *IEEE Trans. Signal Process.* **1993**, *41*, 3397–3415. [CrossRef]
11. Davis, G.; Mallat, S.; Avellaneda, M. Adaptive greedy approximations. *J. Constr. Approx.* **1997**, *13*, 57–98. [CrossRef]
12. Tropp, J.A.; Gilbert, A.C. Signal recovery from random measurements via orthogonal matching pursuit. *IEEE Trans. Inf. Theory* **2007**, *53*, 4655–4666. [CrossRef]
13. Donoho, D.L.; Tsaig, Y.; Drori, I.; Starck, J.-L. Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit. *IEEE Trans. Inf. Theory* **2012**, *58*, 1094–1121. [CrossRef]
14. Tropp, J.A. Just relax: Convex programming methods for identifying sparse signals in noise. *IEEE Trans. Inf. Theory* **2006**, *52*, 1030–1051. [CrossRef]
15. Donoho, D.L.; Elad, M.; Temlyakov, V.N. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Trans. Inf. Theory* **2005**, *52*, 6–18. [CrossRef]
16. Koh, K.; Kim, S.J.; Boyd, S. An interior-point method for large-scale $\ell$1-regularized logistic regression. *J. Mach. Learn. Res.* **2007**, *1*, 606–617.
17. Daubechies, I.; Defrise, M.; De, M.C. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Commun. Pure Appl. Math. J. Issued Courant Inst. Math. Sci.* **2004**, *57*, 1413–1457. [CrossRef]
18. Wright, S.J.; Nowak, R.D.; Figueiredo, M.A.T. Sparse reconstruction by separable approximation. *IEEE Trans. Signal Process.* **2009**, *57*, 2479–2493. [CrossRef]
19. Donoho, D.L.; Tsaig, Y. Fast Solution of l1-Norm Minimization Problems When the Solution May Be Sparse. *IEEE Trans. Inf. Theory* **2008**, *54*, 4789–4812. [CrossRef]
20. Gorodnitsky, I.F.; Rao, B.D. Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm. *IEEE Trans. Signal Process.* **1997**, *45*, 600–616. [CrossRef]
21. Cotter, S.; Rao, B.; Engan, K.; Kreutz-Delgado, K. Sparse solutions to linear inverse problems with multiple measurement vectors. *IEEE Trans. Signal Process.* **2005**, *53*, 2477–2488. [CrossRef]
22. Yardibi, T.; Li, J.; Stoica, P.; Xue, M.; Baggeroer, A.B. Source localization and sensing: A nonparametric iterative adaptive approach based on weighted least squares. *IEEE Trans. Aerosp. Electron. Syst.* **2010**, *46*, 425–443. [CrossRef]
23. Rowe, W.; Li, J.; Stoica, P. Sparse iterative adaptive approach with application to source localization. In Proceedings of the IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing, St. Martin, France, 15–18 December 2013; pp. 196–199.

24. Yang, Z.C.; Li, X.; Wang, H.Q.; Jiang, W.D. On clutter sparsity analysis in space-time adaptive processing airborne radar. *IEEE Geosci. Remote Sens. Lett.* **2013**, *10*, 1214–1218. [CrossRef]
25. Duan, K.Q.; Yuan, H.D.; Xu, H.; Liu, W.J.; Wang, Y.L. Sparisity-based non-stationary clutter suppression technique for airborne radar. *IEEE Access* **2018**, *6*, 56162–56169. [CrossRef]
26. Yang, Z.C.; Wang, Z.T.; Liu, W.J. Reduced-dimension space-time adaptive processing with sparse constraints on beam-Doppler selection. *Signal Process.* **2019**, *157*, 78–87. [CrossRef]
27. Zhang, W.; An, R.X.; He, Z.S.; Li, H.Y. Reduced dimension STAP based on sparse recovery in heterogeneous clutter environments. *IEEE Trans. Aerosp. Electron. Syst.* **2020**, *56*, 785–795. [CrossRef]
28. Li, Z.Y.; Wang, T. ADMM-Based Low-Complexity Off-Grid Space-Time Adaptive Processing Methods. *IEEE Access* **2020**, *8*, 206646–206658. [CrossRef]
29. Su, Y.Y.; Wang, T.; Li, Z.Y. A Grid-Less Total Variation Minimization-Based Space-Time Adaptive Processing for Airborne Radar. *IEEE Access* **2020**, *8*, 29334–29343. [CrossRef]
30. Tipping, M.E. Sparse Bayesian learning and the relevance vector machine. *J. Mach. Learn.* **2001**, *1*, 211–244.
31. Wipf, D.P.; Rao, B.D. Sparse Bayesian learning for basis selection. *IEEE Trans. Signal Process.* **2004**, *52*, 2153–2164. [CrossRef]
32. Wipf, D.P.; Rao, B.D. An empirical Bayesian strategy for solving the simultaneous sparse approximation problem. *IEEE Trans. Signal Process.* **2007**, *55*, 3704–3716. [CrossRef]
33. Ji, S.; Xue, Y.; Carin, L. Bayesian compressive sensing. *IEEE Trans. Signal Process.* **2008**, *56*, 2346–2356. [CrossRef]
34. Baraniuk, R.G.; Cevher, V.; Duarte, M.F.; Hegde, C. Model-based compressive sensing. *IEEE Trans. Inf. Theory* **2010**, *56*, 1982–2001. [CrossRef]
35. Zhang, Z.; Rao, B.D. Extension of SBL algorithms for the recovery of block sparse signals with intra-block correlation. *IEEE Trans. Signal Process.* **2013**, *61*, 2009–2015. [CrossRef]
36. Duan, K.Q.; Wang, Z.T.; Xie, W.C.; Chen, H.; Wang, Y.L. Sparsity-based STAP algorithm with multiple measurement vectors via sparse Bayesian learning strategy for airborne radar. *IET Signal Process.* **2017**, *11*, 544–553. [CrossRef]
37. Sun, Y.; Yang, X.; Long, T.; Sarkar, T.K. Robust sparse Bayesian learning STAP method for discrete interference suppression in nonhomogeneous clutter. In Proceedings of the IEEE Radar Conference, Seattle, WA, USA, 8–12 May 2017; pp. 1003–1008.
38. Wu, Q.; Zhang, Y.D.; Amin, M.G.; Himed, B. Space-Time Adaptive Processing and Motion Parameter Estimation in Multistatic Passive Radar Using Sparse Bayesian Learning. *IEEE Trans. Geosci. Remote Sens.* **2016**, *54*, 944–957. [CrossRef]
39. Li, Z.H.; Guo, Y.D.; Zhang, Y.S.; Zhou, H.; Zheng, G.M. Sparse Bayesian learning based space-time adaptive processing against unknown mutual coupling for airborne radar using middle subarray. *IEEE Access* **2019**, *7*, 6094–6108. [CrossRef]
40. Liu, H.; Zhang, Y.; Guo, Y.; Wang, Q.; Wu, Y. A novel STAP algorithm for airborne MIMO radar based on temporally correlated multiple sparse Bayesian learning. *Math. Probl. Eng.* **2016**, *2016*, 3986903. [CrossRef]
41. Liu, C.; Wang, T.; Zhang, S.; Ren, B. A Fast Space-Time Adaptive Processing Algorithm Based on Sparse Bayesian Learning for Airborne Radar. *Sensors* **2022**, *22*, 2664. [CrossRef]
42. Liu, K.; Wang, T.; Wu, J.; Chen, J. A Two-Stage STAP Method Based on Fine Doppler Localization and Sparse Bayesian Learning in the Presence of Arbitrary Array Errors. *Sensors* **2022**, *22*, 77. [CrossRef]
43. Cui, N.; Xing, K.; Duan, K.; Yu, Z. Knowledge-aided block sparse Bayesian learning STAP for phased-array MIMO airborne radar. *IET Radar Sonar Navig.* **2021**, *15*, 1628–1642. [CrossRef]
44. Cui, N.; Xing, K.; Duan, K.; Yu, Z. Fast Tensor-based Three-dimensional Sparse Bayesian Learning Space-Time Adaptive Processing Method. *J. Radars* **2021**, *10*, 919–928.
45. Wang, Z.T.; Xie, W.; Duan, K.; Wang, Y. Clutter suppression algorithm based on fast converging sparse Bayesian learning for airborne radar. *Signal Process.* **2017**, *130*, 159–168. [CrossRef]
46. Liu, C.; Wang, T.; Zhang, S.; Ren, B. Clutter suppression based on iterative reweighted methods with multiple measurement vectors for airborne radar. *IET Radar Sonar Navig.* 2022. *early view*. [CrossRef]
47. Xue, M.; Xu, L.; Li, J. IAA spectral estimation: Fast implementation using the Gohberg–Semencul factorization. *IEEE Trans. Signal Process.* **2011**, *59*, 3251–3261.
48. Kailath, T.; Sayed, A.H. Displacement structure: Theory and applications. *SIAM Rev.* **1995**, *37*, 297–386. [CrossRef]
49. Blahut, R.E. *Fast Algorithms for Signal Processing*; Cambridge University Press: London, UK, 2010.
50. Noor, F.; Morgera, S.D. Recursive and iterative algorithms for computing eigenvalues of Hermitian Toeplitz matrices. *IEEE Trans. Signal Process.* **1993**, *41*, 1272–1280. [CrossRef]
51. Jain, J.R. An efficient algorithm for a large Toeplitz set of linear equations. *IEEE Trans. Acoust. Speech Signal Process.* **1980**, *27*, 612–615. [CrossRef]
52. Glentis, G.O.; Jakobsson, A. Efficient implementation of iterative adaptive approach spectral estimation techniques. *IEEE Trans. Signal Process.* **2011**, *59*, 4154–4167. [CrossRef]
53. Harville, D.A. *Matrix Algebra from a Statistician's Perspective*; Springer: New York, NY, USA, 1998.
54. Wax, M.; Kailath, T. Efficient inversion of Toeplitz-block Toeplitz matrix. *IEEE Trans. Acoust. Speech Signal Process.* **1983**, *31*, 1218–1221. [CrossRef]
55. Musicus, B. Fast MLM power spectrum estimation from uniformly spaced correlations. *IEEE Trans. Acoust. Speech Signal Process.* **1985**, *33*, 333–1335. [CrossRef]

56. Glentis, G.O. A Fast Algorithm for APES and Capon Spectral Estimation. *IEEE Trans. Signal Process.* **2008**, *56*, 4207–4220. [CrossRef]
57. Jakobsson, A.; Marple, S.L.; Stoica, P. Computationally efficient two-dimensional Capon spectrum analysis. *IEEE Trans. Signal Process.* **2000**, *48*, 2651–2661. [CrossRef]
58. Yang, Z.; Li, X.; Wang, H.; Jiang, W. Adaptive clutter suppression based on iterative adaptive approach for airborne radar. *Signal Process.* **2013**, *93*, 3567–3577. [CrossRef]
59. Robey, F.; Fuhrmann, D.; Kelly, E.; Nitzberg, R. A CFAR adaptive matched filter detector. *IEEE Trans. Aerosp. Electron. Syst.* **1992**, *28*, 208–216. [CrossRef]
60. Titi, G.W.; Marshall, D.F. The ARPA/NAVY Mountaintop Program: Adaptive signal processing for airborne early warning radar. In Proceedings of the 1996 IEEE International Conference on Acoustics, Speech, and Signal Processing, Atlanta, GA, USA, 9 May 1996.