Optimized Sparse Sampling Lattices

Peter Vouras, Mohamed Kashef (Hany), Sudantha Perera, Carnot Nogueira, Richard Candell, Kate A. Remley

***Abstract*—Sparse sampling approaches have been widely studied to achieve less complex measurement systems while maintaining detection performance. In this paper, we derive a new gradient implementation of an alternating projections algorithm that determines the optimal locations for spatial samples in a sparse array lattice. In a second phase of the sparse array design, an adaptive beamformer is used to further reduce the overall sidelobe level. Simulated results show a significant reduction in grating lobes. The approach described herein is useful in wideband synthetic aperture channel sounding applications where reducing the spatial sample set has the potential to significantly reduce data acquisition time.**

I. Introduction

Sparse sampling approaches have been an active area of research for many years in the phased array community. The allure of sparse sampling is that if detection and resolution performance can be maintained, then a sparse phased array will be cheaper and require less hardware complexity. In the synthetic aperture channel sounding community, sparse spatial sampling lattices are attractive because they can reduce the necessary data acquisition time. This paper derives a new gradient implementation of an alternating projections algorithm that searches for the optimal locations of samples in a sparse spatial lattice. In a second phase of the design, an adaptive beamformer is used to further reduce the overall sidelobe level in the beamformed output of the sparse array.

Other approaches investigated in the literature for sparse array design include simulated annealing and genetic algorithms. Simulated annealing is a stochastic optimization method analogous to the manner in which a metal cools and anneals [1]. The algorithm seeks to minimize a sparse array energy function which is set proportional to the peak sidelobe level. At each algorithm iteration, the location of array elements is randomized by moving one element at a time. The peak sidelobe level of the perturbed array is found and compared to the best solution of the last iteration. The new solution is accepted if it lowers the peak sidelobe level, or it may also be accepted with some finite probability if it raises the sidelobe level. In this way, the algorithm is less likely to be trapped in a local minimum. As the cost function is progressively minimized, the probability of accepting an inferior solution is reduced and ultimately the algorithm converges to a solution that may be close to optimal, provided the optimization parameters are well chosen.

P. V. is with the United States Department of Defense, Washington, DC, 20375 USA, e-mail: synthetic_aperture_twg@ieee.org.

M. K. and R. C. are with the National Institute of Standards and Technology (NIST), Gaithersburg, MD, 20899 USA e-mail: mohamed.kashef, richard.candell@nist.gov.

S. P., C. N. and K. A. R. are with the National Institute of Standards and Technology (NIST), Boulder, CO, 80303 USA e-mail: sudantha.perera, carnot.nogueira, kate.remley@nist.gov.

Simulated annealing has been applied to the optimization of sparse lattices in [2]–[5]

Genetic algorithms iteratively operate on the individuals in a population [6], [7]. Each member of the population represents a potential solution to the optimization problem. Initially, the population is randomly generated. The individuals are evaluated by means of a fitness function and then either retained or replaced. New individuals are created through either a cross-over operation or a mutation. Genetic optimization has been applied to the layout of sparse arrays in [8]–[10].

II. Part A. Sparse Array Element Optimization

The proposed approach for optimizing the spatial sample locations in a sparse lattice is to use a novel gradient implementation of alternating projections to minimize the mean squared error (MSE) between a beamspace vector of sidelobe levels taken from a filled array pattern and a corresponding beamspace vector of sparse array sidelobe levels as in,

$$\min_{\mathbf{x},\mathbf{y}} \quad \|\mathbf{b}(\mathbf{u},\mathbf{v}) - \mathbf{A}(\mathbf{x},\mathbf{y})\mathbf{s}(\mathbf{x},\mathbf{y})\|_2^2. \tag{1}$$

The $M\times 1$ vector $\mathbf{s}(\mathbf{x},\mathbf{y})$ corresponds to the output of a planar sparse array of size $M = M_1M_2$ with $M < N$. The components of the $\mathbf{x}$ and $\mathbf{y}$ position vectors correspond to the x and y coordinates of each array element,

$$\begin{aligned}\mathbf{x} &= [x_0, \quad x_1 \dots x_{M_1-1}]^T, \\ \mathbf{y} &= [y_0, \quad y_1 \dots y_{M_2-1}]^T.\end{aligned} \tag{2}$$

The beamspace vector $\mathbf{b}(\mathbf{u},\mathbf{v})$ contains complex amplitudes corresponding to the beampattern of an N-element filled array sampled over a regular grid $(\mathbf{u},\mathbf{v})$ of L angles in the sidelobe region,

$$\begin{aligned}\mathbf{b}(\mathbf{u},\mathbf{v}) &= [b(u_0,v_0) \quad b(u_1,v_1)\dots b(u_{L-1},v_{L-1})]^T, \\ \mathbf{u} &= [u_0, \quad u_1 \dots u_{L-1}]^T, \\ \mathbf{v} &= [v_0, \quad v_1 \dots v_{L-1}]^T.\end{aligned} \tag{3}$$

The sine space coordinates (u,v) correspond to

$$\begin{aligned}u &= \sin\theta\cos\phi, \\ v &= \sin\theta\sin\phi\end{aligned} \tag{4}$$

where θ and ϕ are spherical angle coordinates. The components of the beamspace vector $\mathbf{b}(\mathbf{u},\mathbf{v})$ correspond to L values of the beampattern $b(u,v)$ for a planar filled array with $N = N_1N_2$ elements given by,

$$b(u_l,v_l) = \sum_{k=0}^{N_1-1}\sum_{m=0}^{N_2-1} e^{-j\frac{2\pi}{\lambda}(x_ku_l+y_mv_l)} \tag{5}$$

for $0 \le l \le L-1$. Here x_k and y_m denote the x and y coordinates of the (k,m)th filled-array element.

The rows of the $L \times M$ matrix $\mathbf{A}(\mathbf{x},\mathbf{y})$ with $L > M$ correspond to the sparse array steering vectors $\mathbf{a}(u_l, v_l)$,

$$\mathbf{A}(\mathbf{x},\mathbf{y}) = \begin{bmatrix} \mathbf{a}(u_0,v_0)^T \\ \mathbf{a}(u_1,v_1)^T \\ \vdots \\ \mathbf{a}(u_{L-1},v_{L-1})^T \end{bmatrix} \tag{6}$$

for $0 \le l \le L-1$. The sparse array steering vector $\mathbf{a}(u_l, v_l)$ in the direction (u_l, v_l) is the $M \times 1$ vector given by,

$$[\mathbf{a}(u_l,v_l)]_{km} = \left[e^{-j\frac{2\pi}{\lambda}(x_k u_l + y_m v_l)}\right]^T \tag{7}$$

where $0 \le k \le M_1 - 1$ and $0 \le m \le M_2 - 1$.

The kth column of the matrix $\mathbf{A}(\mathbf{x},\mathbf{y})$ is the electrical angle vector corresponding to the kth array element. The lth component of an electrical angle vector is the phase shift corresponding to the lth beam-steering direction (u_l, v_l). Using electrical angle vectors $\mathbf{c}_{km}$, the matrix $\mathbf{A}(\mathbf{x},\mathbf{y})$ can be rewritten as

$$\mathbf{A}(\mathbf{x},\mathbf{y}) = \left[\begin{array}{c|c|c|c} \mathbf{c}_{00} & \mathbf{c}_{01} & \dots & \mathbf{c}_{M_1-1,M_2-1} \end{array}\right] \tag{8}$$

where

$$\mathbf{c}_{km} = \begin{bmatrix} e^{-j\frac{2\pi}{\lambda}(x_k u_0 + y_m v_0)} \\ e^{-j\frac{2\pi}{\lambda}(x_k u_1 + y_m v_1)} \\ \vdots \\ e^{-j\frac{2\pi}{\lambda}(x_k u_{L-1} + y_m v_{L-1})} \end{bmatrix}. \tag{9}$$

Proceeding by fixing $\mathbf{x}, \mathbf{y}$ and minimizing (1) with respect to $\mathbf{s}(\mathbf{x},\mathbf{y})$ yields the least-squares estimate

$$\widehat{\mathbf{s}}(\mathbf{x},\mathbf{y}) = \left[\mathbf{A}(\mathbf{x},\mathbf{y})^H \mathbf{A}(\mathbf{x},\mathbf{y})\right]^{-1} \mathbf{A}(\mathbf{x},\mathbf{y})^H \mathbf{b}(\mathbf{u},\mathbf{v}). \tag{10}$$

Substituting $\widehat{\mathbf{s}}(\mathbf{u},\mathbf{v})$ into (1) yields the minimization problem

$$\min_{\mathbf{x},\mathbf{y}} \quad \|\mathbf{b}(\mathbf{u},\mathbf{v}) - \mathbf{P}_{\mathbf{A}(\mathbf{x},\mathbf{y})}\mathbf{b}(\mathbf{u},\mathbf{v})\|_2^2 \tag{11}$$

with

$$\begin{aligned} \mathbf{P}_{\mathbf{A}(\mathbf{x},\mathbf{y})} &= \mathbf{A}(\mathbf{x},\mathbf{y})\left[\mathbf{A}(\mathbf{x},\mathbf{y})^H \mathbf{A}(\mathbf{x},\mathbf{y})\right]^{-1} \mathbf{A}(\mathbf{x},\mathbf{y})^H \\ &= \mathbf{A}(\mathbf{x},\mathbf{y})\mathbf{A}(\mathbf{x},\mathbf{y})^\dagger \end{aligned} \tag{12}$$

where $\dagger$ denotes the pseudoinverse. The matrix $\mathbf{P}_{\mathbf{A}(\mathbf{x},\mathbf{y})}$ is the projection matrix onto the range space of $\mathbf{A}(\mathbf{x},\mathbf{y})$. Thus, the estimates of the coordinates $(x_0,y_0),(x_0,y_1),\dots,(x_{M_1-1},y_{M_2-1})$ can be obtained by maximizing the function

$$\max_{\mathbf{x},\mathbf{y}} \quad J(\mathbf{x},\mathbf{y}) = \|\mathbf{P}_{\mathbf{A}(\mathbf{x},\mathbf{y})}\mathbf{b}(\mathbf{u},\mathbf{v})\|_2^2. \tag{13}$$

Using the property of the trace operator that $\mathbf{c}^H\mathbf{D}\mathbf{c} = \mathrm{tr}[\mathbf{D}\mathbf{c}\mathbf{c}^H]$ and the idempotent property of the orthogonal projection matrix $\mathbf{P}_{\mathbf{A}(\mathbf{x},\mathbf{y})}$, equation (12) can be rewritten as

$$\max_{\mathbf{x},\mathbf{y}} \quad J(\mathbf{x},\mathbf{y}) = \mathrm{tr}[\mathbf{P}_{\mathbf{A}(\mathbf{x},\mathbf{y})}\widehat{\mathbf{R}}] \tag{14}$$

where the beamspace covariance matrix is

$$\widehat{\mathbf{R}} = \mathbf{b}(\mathbf{u},\mathbf{v})\mathbf{b}(\mathbf{u},\mathbf{v})^H. \tag{15}$$

A geometric interpretation of (14) is that the estimates $\widehat{\mathbf{x}}, \widehat{\mathbf{y}}$ are obtained by searching over the array manifold $\mathbf{A}(\mathbf{x},\mathbf{y})$ for the M sparse-array electrical-angle vectors that form an M-dimensional subspace closest to the filled-array beamspace vector $\mathbf{b}(\mathbf{u},\mathbf{v})$. Here 'closeness' is measured by the Frobenius norm of the projection of the filled-array beamspace vector onto the sparse array electrical angle vectors. The function $J(\mathbf{x},\mathbf{y})$ is multidimensional and highly nonlinear which renders it difficult to solve directly. The next section describes a greedy optimization approach that transforms the original problem into an iterative sequence of two-dimensional optimization programs.

A. Alternating Projections Algorithm

The alternating projections (AP) algorithm maximizes the cost function $J(\mathbf{x},\mathbf{y})$ with respect to one pair of parameters (x_k, y_k) while holding the other parameters fixed. Since iterations of the AP algorithm perform a maximization at every step, the value of $J(\mathbf{x},\mathbf{y})$ can never decrease, so the algorithm is guaranteed to converge to a local maximum. Depending on the initial conditions, the local maximum may or may not coincide with the global maximum. Since $J(\mathbf{x},\mathbf{y})$ will, in general, have many local maxima, proper initialization is vital for the AP algorithm to converge to the global solution.

At the core of the AP algorithm is a projection matrix decomposition described as follows. Consider two arbitrary matrices $\mathbf{E}$ and $\mathbf{F}$ with the same number of rows. The projection matrix $\mathbf{P}_{[\mathbf{E},\mathbf{F}]}$ onto the column space of the augmented matrix $[\mathbf{E},\mathbf{F}]$ is equal to

$$\mathbf{P}_{[\mathbf{E},\mathbf{F}]} = \mathbf{P}_{[\mathbf{E},\mathbf{F}_\mathbf{E}]} \tag{16}$$

where

$$\mathbf{F}_\mathbf{E} = \mathbf{P}_\mathbf{E}^\perp \mathbf{F} = (\mathbf{I} - \mathbf{P}_\mathbf{E})\mathbf{F} \tag{17}$$

is the orthogonal complement of the projection of the vector space spanned by the columns of $\mathbf{F}$ onto the range space of $\mathbf{E}$. The column space of $\mathbf{F}_\mathbf{E}$ is orthogonal to the column space of $\mathbf{E}$ and their direct sum spans the column space of $[\mathbf{E},\mathbf{F}]$ so it follows that

$$\mathbf{P}_{[\mathbf{E},\mathbf{F}]} = \mathbf{P}_\mathbf{E} + \mathbf{P}_{\mathbf{F}_\mathbf{E}}. \tag{18}$$

Applying (16) and (18) to $\mathbf{P}_{\mathbf{A}(\mathbf{x},\mathbf{y})}$ yields

$$\begin{aligned} \mathbf{P}_{\mathbf{A}(\mathbf{x},\mathbf{y})} &= \mathbf{P}_{[\mathbf{A}(\widehat{\mathbf{x}}_k,\widehat{\mathbf{y}}_k),\mathbf{c}(x_k,y_k)]} \\ &= \mathbf{P}_{\mathbf{A}(\widehat{\mathbf{x}}_k,\widehat{\mathbf{y}}_k)} + \mathbf{P}_{\mathbf{c}(x_k,y_k)_{\mathbf{A}(\widehat{\mathbf{x}}_k,\widehat{\mathbf{y}}_k)}} \end{aligned} \tag{19}$$

where the $(M-1)\times 1$ vectors $\widehat{\mathbf{x}}_k$ and $\widehat{\mathbf{y}}_k$ are

$$\begin{aligned} \widehat{\mathbf{x}}_k &= [x_0, x_1, \dots, x_{k-1}, x_{k+1}, \dots, x_{M-1}]^T \\ \widehat{\mathbf{y}}_k &= [y_0, y_1, \dots, y_{k-1}, y_{k+1}, \dots, y_{M-1}]^T \end{aligned} \tag{20}$$

and the $L \times (M-1)$ matrix $\mathbf{A}(\widehat{\mathbf{x}}_k, \widehat{\mathbf{y}}_k)$ is

$$\begin{aligned} &\mathbf{A}(\widehat{\mathbf{x}}_k, \widehat{\mathbf{y}}_m) = \\ &[\mathbf{c}(x_0,y_0), ..., \mathbf{c}(x_{k-1},y_{m-1}), \mathbf{c}(x_{k+1},y_{m+1}), ...]. \end{aligned} \tag{21}$$

Rewriting the maximization problem in (13) to search around the kth array element location (x_k, y_k) at the $(p+1)$st algorithm iteration while holding all other array elements fixed yields

$$x_k^{(p+1)}, y_k^{(p+1)} = \arg\max_{x_k, y_k} \quad \text{tr}[\mathbf{P}_{[\mathbf{A}(\boldsymbol{\aleph}_k^{(p)}, \boldsymbol{\beth}_k^{(p)}), \mathbf{c}(x_k, y_k)]} \widehat{\mathbf{R}}]. \tag{22}$$

Equation (22) states that to obtain the coordinate estimates $x_k^{(l+1)}, y_k^{(l+1)}$ for the kth array element at the $(l+1)$st algorithm iteration, the parameters $\widehat{\mathbf{x}}_k^{(l)}, \widehat{\mathbf{y}}_k^{(l)}$ are held fixed while the parameters x_k, y_k are free to vary. Applying the matrix decomposition in (18) to (22) and ignoring the first term in the summation since it is constant yields the equivalent maximization problem

$$x_k^{(l+1)}, y_k^{(l+1)} = \arg\max_{x_k, y_k} \quad \text{tr}[\mathbf{P}_{\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}} \widehat{\mathbf{R}}]. \tag{23}$$

Using (17) and (12), the vector $\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}$ can be written as

$$\begin{aligned} \mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})} &= \left[\mathbf{I} - \mathbf{P}_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right] \mathbf{c}(x_k, y_k) \qquad (24) \\ &= \left[\mathbf{I} - \mathbf{A}(\widehat{\mathbf{x}}_k^{(l)}, \widehat{\mathbf{y}}_k^{(l)}) \mathbf{A}(\widehat{\mathbf{x}}_k^{(l)}, \widehat{\mathbf{y}}_k^{(l)})^{\dagger}\right] \mathbf{c}(x_k, y_k). \end{aligned}$$

Equation (24) shows that the vector $\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}$ is orthogonal to the projection of $\mathbf{c}(x_k, y_k)$ onto the column space of $\mathbf{A}(\widehat{\mathbf{x}}_k^{(l)}, \widehat{\mathbf{y}}_k^{(l)})$. Also by (12),

$$\begin{aligned} &\mathbf{P}_{\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}} = \qquad (25) \\ &= \frac{\left[\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right] \left[\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right]^H}{\left[\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right]^H \left[\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right]}. \end{aligned}$$

Define the unit norm vector

$$\mathbf{d}_k^{(l)} \equiv \mathbf{d}(x_k, y_k; \widehat{\mathbf{x}}_k^{(l)}, \widehat{\mathbf{y}}_k^{(l)}) = \frac{\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}}{\|\mathbf{c}(x_k, y_k)_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\|_2} \tag{26}$$

and substitute (25) into (23). By applying properties of the trace operator including $\text{tr}(\mathbf{AB}) = \text{tr}(\mathbf{BA})$, the optimization problem in equation (23) becomes

$$\begin{aligned} x_k^{(l+1)}, y_k^{(l+1)} &= \arg\max_{x_k, y_k} \quad \mathbf{d}_k^{(l)^H} \widehat{\mathbf{R}} \mathbf{d}_k^{(l)} \qquad (27) \\ &\equiv \arg\max_{x_k, y_k} \quad J^l(x_k, y_k). \end{aligned}$$

The entire AP algorithm can now be summarized as follows,

Algorithm 1 Sparse Array Alternating Projections Algorithm

Require: Initial values $x_0^{(0)}, \ldots, x_{M-1}^{(0)}$ and $y_0^{(0)}, \ldots, y_{M-1}^{(0)}$

1: Set algorithm iteration $l = 1$

2: Until $\left|x_k^{(l+1)} - x_k^{(l)}\right|^2 < \epsilon$ and $\left|y_k^{(l+1)} - y_k^{(l)}\right|^2 < \epsilon$ for all $k = 0, \ldots, M-1$, compute the location for the kth array element by solving $x_k^{(l+1)}, y_k^{(l+1)} = \arg\max_{x_k, y_k} \quad J^l(x_k, y_k)$

The primary contribution of this paper described in the next section is a gradient-based method to maximize the cost function $J^l(x_k, y_k)$ at each iteration.

III. Gradient-Based Implementation of Alternating Projections Algorithm

A. Derivation of Gradient Vector

In this section, an analytical expression for the gradient vector of the $(l+1)$st cost function specified in (27) is derived. To start, we rewrite the cost function as

$$\begin{aligned} &J^l(x_k, y_k) = \qquad (28) \\ &\frac{\mathbf{c}(x_k, y_k)^H \left[\mathbf{I} - \mathbf{P}_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right]^H \widehat{\mathbf{R}} \left[\mathbf{I} - \mathbf{P}_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right] \mathbf{c}(x_k, y_k)}{\mathbf{c}(x_k, y_k)^H \left[\mathbf{I} - \mathbf{P}_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right] \mathbf{c}(x_k, y_k)} \\ &\equiv \frac{\mathbf{c}(x_k, y_k)^H \mathbf{W} \mathbf{c}(x_k, y_k)}{\mathbf{c}(x_k, y_k)^H \mathbf{Q} \mathbf{c}(x_k, y_k)} \end{aligned}$$

by substituting (24) into (27) and note that the idempotent and self-adjoint properties of orthogonal projection matrices imply that $\left[\mathbf{I} - \mathbf{P}_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right]^H \left[\mathbf{I} - \mathbf{P}_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right] = \left[\mathbf{I} - \mathbf{P}_{\mathbf{A}(\boldsymbol{\aleph}_k^{(l)}, \boldsymbol{\beth}_k^{(l)})}\right]$.

The electrical angle vector $\mathbf{c}(x_k, y_k)$ defined in (9) can be written as the Kronecker product of two electrical angle vectors,

$$\begin{aligned} &\mathbf{c}(x_k, y_k) = \qquad (29) \\ &\left[e^{-j\frac{2\pi}{\lambda} x_k u_s} \middle| 0 \le s \le L_1 - 1\right]^T \otimes \left[e^{-j\frac{2\pi}{\lambda} y_k v_r} \middle| 0 \le r \le L_2 - 1\right]^T. \end{aligned}$$

where $L_1 L_2 = L$. Hereafter, to simplify notation, the subscript k denoting the kth array element will be dropped from the coordinates x_k, y_k. Next, we consider a change in element location corresponding to (δ_x, δ_y). Then,

$$\begin{aligned} &\mathbf{c}(x + \delta_x, y + \delta_y) = \qquad (30) \\ &\left[e^{-j\frac{2\pi}{\lambda} u_s (x + \delta_x)} \middle| 0 \le s \le L_1 - 1\right]^T \otimes \\ &\left[e^{-j\frac{2\pi}{\lambda} v_r (y + \delta_y)} \middle| 0 \le r \le L_2 - 1\right]^T. \end{aligned}$$

Define the diagonal matrices,

$$\begin{aligned} \boldsymbol{\Delta} x &= \delta_x \begin{bmatrix} \frac{2\pi}{\lambda} u_0 & 0 & 0 & 0 & 0 \\ 0 & \frac{2\pi}{\lambda} u_1 & 0 & 0 & 0 \\ 0 & 0 & \frac{2\pi}{\lambda} u_2 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \frac{2\pi}{\lambda} u_{L_1 - 1} \end{bmatrix} \qquad (31) \\ &\equiv \delta_x \mathbf{T}_u \end{aligned}$$

and

$$\begin{aligned} \boldsymbol{\Delta} y &= \delta_y \begin{bmatrix} \frac{2\pi}{\lambda} v_0 & 0 & 0 & 0 & 0 \\ 0 & \frac{2\pi}{\lambda} v_1 & 0 & 0 & 0 \\ 0 & 0 & \frac{2\pi}{\lambda} v_2 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \frac{2\pi}{\lambda} v_{L_2 - 1} \end{bmatrix} \qquad (32) \\ &\equiv \delta_y \mathbf{T}_v. \end{aligned}$$

Then using the identity

$$\boldsymbol{\Delta} x \oplus \boldsymbol{\Delta} y = \boldsymbol{\Delta} x \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{\Delta} y \tag{33}$$

and the properties of the matrix exponential, the perturbed electrical angle vector $\mathbf{c}(x+\delta_x, y+\delta y)$ can be written as

$$\begin{aligned}\mathbf{c}(x+\delta_x, y+\delta y) &= e^{-j\boldsymbol{\Delta}x} \otimes e^{-j\boldsymbol{\Delta}y}\mathbf{c}(x,y) \\ &= e^{-j(\boldsymbol{\Delta}x \oplus \boldsymbol{\Delta}y)}\mathbf{c}(x,y).\end{aligned} \tag{34}$$

At this point it is useful to clarify the overarching strategy for computing the gradient vector of the cost function $J(x,y)$ in (27); where the superscript iteration index l has been dropped for simplicity. A related approach is also described in [11]. The desired gradient vector of $J(x,y)$ to be computed is defined as $\nabla J = \begin{bmatrix}\partial J/\partial x & \partial J/\partial y\end{bmatrix}^T$. In terms of numerator and denominator functions, $J(x,y) = N(x,y)/D(x,y)$, so using the quotient rule for differentiation yields

$$\begin{aligned}\frac{\partial J}{\partial x} &= \frac{\frac{\partial N}{\partial x}D(x,y) - \frac{\partial D}{\partial x}N(x,y)}{D(x,y)^2} \\ \frac{\partial J}{\partial y} &= \frac{\frac{\partial N}{\partial y}D(x,y) - \frac{\partial D}{\partial y}N(x,y)}{D(x,y)^2}.\end{aligned} \tag{35}$$

It is clear that to apply the quotient rule for computing ∇J it is also necessary to compute $\nabla N = \begin{bmatrix}\partial N/\partial x & \partial N/\partial y\end{bmatrix}^T$ and $\nabla D = \begin{bmatrix}\partial D/\partial x & \partial D/\partial y\end{bmatrix}^T$. A useful fact is that since the numerator function N is continuously differentiable with respect to x and y, the directional derivative $N'(\mathbf{p};\mathbf{d})$ of N at the point $\mathbf{p} = \begin{bmatrix}x & y\end{bmatrix}^T$ in the direction $\mathbf{d} = \begin{bmatrix}\delta_x & \delta_y\end{bmatrix}^T$ is equal to [12]

$$N'(\mathbf{p};\mathbf{d}) = \nabla N(\mathbf{p})^T\mathbf{d}. \tag{36}$$

In the case at hand, ∇N is unknown and the quantity to be determined, but the directional derivative $N'(\mathbf{p};\mathbf{d})$ can also be calculated as the derivative with respect to t of the function $G_N(t) = N(\mathbf{p}+t\mathbf{d})$ evaluated at $t=0$,

$$N'(\mathbf{p};\mathbf{d}) = \left.\frac{d}{dt}G_N(t)\right|_{t=0} = \left.\frac{d}{dt}N(\mathbf{p}+t\mathbf{d})\right|_{t=0}. \tag{37}$$

Thus ∇N can be recovered by using (37) to compute the directional derivative $N'(\mathbf{p};\mathbf{d})$ and then writing the result in a form compatible with (36) to recover the gradient vector. The same procedure also applies to the denominator function $D(x,y)$ using the derivative with respect to t of the function $G_D(t) = D(\mathbf{p}+t\mathbf{d})$ evaluated at $t=0$.

Continuing along this track and starting with $D(x,y)$ yields,

$$\begin{aligned}G_D(t) &= D(x+t\delta_x, y+t\delta_y) = \\ &= \mathbf{c}(x+t\delta_x, y+t\delta_y)^H\mathbf{Q}\mathbf{c}(x+t\delta_x, y+t\delta_y) \\ &= \mathbf{c}(x,y)^H e^{j(\boldsymbol{\Delta}x\oplus\boldsymbol{\Delta}y)t}\mathbf{Q}e^{-j(\boldsymbol{\Delta}x\oplus\boldsymbol{\Delta}y)t}\mathbf{c}(x,y)\end{aligned} \tag{38}$$

and the desired directional derivative

$$\begin{aligned}&D'(\mathbf{p};\mathbf{d}) = j\mathbf{c}(x,y)^H\left[\boldsymbol{\Delta}x\oplus\boldsymbol{\Delta}y, \mathbf{Q}\right]\mathbf{c}(x,y) \\ &= j\text{tr}\left(\left[\boldsymbol{\Delta}x\oplus\boldsymbol{\Delta}y, \mathbf{Q}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right) \\ &= j\text{tr}\left(\left[\delta_y\mathbf{T}_v\oplus\delta_x\mathbf{T}_u, \mathbf{Q}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right) \\ &= j\text{tr}\left(\left(\left[\delta_x\mathbf{T}_u\otimes\mathbf{I}, \mathbf{Q}\right] + \left[\mathbf{I}\otimes\delta_y\mathbf{T}_v, \mathbf{Q}\right]\right)\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right) \\ &= j\left[\text{tr}\left(\left[\delta_x\mathbf{T}_u\otimes\mathbf{I}, \mathbf{Q}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right)\right. \\ &\quad \left.+\text{tr}\left(\left[\mathbf{I}\otimes\delta_y\mathbf{T}_v, \mathbf{Q}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right)\right] \\ &= j\left[\delta_x\text{tr}\left(\left[\mathbf{T}_u\otimes\mathbf{I}, \mathbf{Q}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right)\right. \\ &\quad \left.+\delta_y\text{tr}\left(\left[\mathbf{I}\otimes\mathbf{T}_v, \mathbf{Q}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right)\right]\end{aligned} \tag{39}$$

where the notation $[\mathbf{A},\mathbf{B}]$ denotes the Lie bracket, $[\mathbf{A},\mathbf{B}] = \mathbf{AB}-\mathbf{BA}$. Rewriting (39) in matrix form and comparing to (36) yields

$$\begin{aligned}&D'(\mathbf{p};\mathbf{d}) = \\ &\begin{bmatrix}-\text{imag}\left(\text{tr}\left(\left[\mathbf{T}_u\otimes\mathbf{I}, \mathbf{Q}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right)\right) \\ -\text{imag}\left(\text{tr}\left(\left[\mathbf{I}\otimes\mathbf{T}_v, \mathbf{Q}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right)\right)\end{bmatrix}^T\begin{bmatrix}\delta_x \\ \delta_y\end{bmatrix} \\ &\equiv \nabla D(\mathbf{p})^T\mathbf{d}.\end{aligned} \tag{40, 41}$$

Repeating the same argument for the numerator function $N(x,y)$ results in

$$\begin{aligned}&N'(\mathbf{p};\mathbf{d}) = \\ &\begin{bmatrix}-\text{imag}\left(\text{tr}\left(\left[\mathbf{T}_u\otimes\mathbf{I}, \mathbf{W}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right)\right) \\ -\text{imag}\left(\text{tr}\left(\left[\mathbf{I}\otimes\mathbf{T}_v, \mathbf{W}\right]\mathbf{c}(x,y)\mathbf{c}(x,y)^H\right)\right)\end{bmatrix}^T\begin{bmatrix}\delta_x \\ \delta_y\end{bmatrix} \\ &\equiv \nabla N(\mathbf{p})^T\mathbf{d}.\end{aligned} \tag{42, 43}$$

Now the components of ∇N and ∇D are clearly available to substitute into (35) to compute ∇J.

B. Conjugate Gradient Algorithm

The conjugate gradient algorithm for maximizing the cost function in (27) for the kth array element coordinates (x_k, y_k) at the lth iteration of the AP algorithm is

Algorithm 2 Conjugate Gradient Algorithm

Require: Initial array element coordinates x_k^0 and y_k^0

1: Set the initial search direction $\mathbf{d}_0 = \nabla J(x_k^0, y_k^0)$
2: Until $\|\nabla J(x_k^j, y_k^j)\|_2 \le \epsilon$, where j denotes the conjugate gradient iteration index, do the following:
3: Determine the step-size μ_j
4: Set $\mathbf{p}_{j+1} = \mathbf{p}_j + \mu_j\mathbf{d}_j$ where $\mathbf{p}_j = \begin{bmatrix}x_k^j & y_k^j\end{bmatrix}^T$
5: Set $\mathbf{g}_{j+1} = \nabla J(x_k^{j+1}, y_k^{j+1})$
6: Set $\mathbf{d}_{j+1} = \mathbf{g}_{j+1} + \alpha_j\mathbf{d}_j$
7: Set $\alpha_j = \frac{\mathbf{g}_{j+1}^T(\mathbf{g}_{j+1}-\mathbf{g}_j)}{\mathbf{g}_j^T\mathbf{g}_j}$
8: Set $j = j+1$

The step-size μ_j for the jth conjugate gradient iteration can be set equal to a constant value small enough to ensure algorithm convergence or it can be chosen via a one-dimensional line search. The preferred approach is to use Armijo's rule. Given an initial stepsize, $0<\rho<1$, Armijo's rule chooses the final stepsize μ_j to be the first value in the sequence $1, \rho, \rho^2, \rho^3, \ldots$ that satisfies the condition $J(\mathbf{p}_j + \mu_j\mathbf{d}_j) \ge J(\mathbf{p}_j) + \mu_j\alpha\nabla J(\mathbf{p}_j)^T\mathbf{d}_j$, for a fixed scalar $0<\alpha<0.5$. In other words, $\mu_j = \rho^m$ for some integer m.

C. Computation of Hessian Matrix

Curvature information for the cost function $J(x,y)$ is contained in the Hessian matrix $\mathbf{H}(x,y)$ defined as

$$\nabla^2 J(x,y) \equiv \mathbf{H}(x,y) = \begin{bmatrix}\frac{\partial^2 J(x,y)}{\partial x^2} & \frac{\partial^2 J(x,y)}{\partial x\partial y} \\ \frac{\partial^2 J(x,y)}{\partial y\partial x} & \frac{\partial^2 J(x,y)}{\partial y^2}\end{bmatrix}. \tag{44}$$

It is possible to compute the Hessian matrix $\mathbf{H}$ in a manner similar to the computation of the gradient vector ∇J by

using directional derivatives. Since $J(\mathbf{p})$ is a function with continuous second partial derivatives in any neighborhood of $\mathbf{p} = [x \quad y]^T$, then for any direction $\mathbf{d}$,

$$\frac{d^2}{dt^2} J(\mathbf{p} + t\mathbf{d}) = \mathbf{d}^H \mathbf{H}(\mathbf{p} + t\mathbf{d})\mathbf{d} \tag{45}$$

which implies that

$$\frac{d^2}{dt^2} J(\mathbf{p} + t\mathbf{d}) \mid_{t=0} = \mathbf{d}^H \mathbf{H}(\mathbf{p})\mathbf{d}. \tag{46}$$

Calculating the second order directional derivatives of $J(\mathbf{p} + t\mathbf{d})$ directly is a cumbersome process so a different approach will be used to compute $\mathbf{H}(\mathbf{p})$ based on the Taylor series expansion of $J(\mathbf{p} + \mathbf{d})$. Using a Taylor series, the function $J(\mathbf{p} + \mathbf{d})$ can be approximated by,

$$J(\mathbf{p} + \mathbf{d}) \approx J(\mathbf{p}) + \nabla J(\mathbf{p})^T \mathbf{d} + \frac{1}{2}\mathbf{d}^T \nabla^2 J(\mathbf{p})\mathbf{d} + \text{H.O.T} \tag{47}$$

$$= J(\mathbf{p}) + \frac{d}{dt} J(\mathbf{p} + t\mathbf{d})\bigg|_{t=0} + \frac{1}{2}\frac{d^2}{dt^2} J(\mathbf{p} + t\mathbf{d})\bigg|_{t=0} + \text{H.O.T.}$$

where H.O.T stands for higher order terms in the Taylor series. So the strategy employed to determine $\mathbf{H}(\mathbf{p}) = \nabla^2 J(\mathbf{p})$ is to compute the Taylor series expansion of $J(\mathbf{p} + \mathbf{d})$ and then collect the second order terms to arrive at a quadratic form with inner matrix equal to the Hessian.

Recall,

$$J(\mathbf{p} + t\mathbf{d}) = \frac{N(\mathbf{p} + t\mathbf{d})}{D(\mathbf{p} + t\mathbf{d})} \tag{48}$$

where

$$\begin{aligned} N(\mathbf{p} + t\mathbf{d}) &= N(x + t\delta_x, y + t\delta_y) = \\ &= \mathbf{c}(x, y)^H e^{j(\boldsymbol{\Delta}x \oplus \boldsymbol{\Delta}y)t} \mathbf{W} e^{-j(\boldsymbol{\Delta}x \oplus \boldsymbol{\Delta}y)t} \mathbf{c}(x, y) \\ D(\mathbf{p} + t\mathbf{d}) &= D(x + t\delta_x, y + t\delta_y) = \\ &= \mathbf{c}(x, y)^H e^{j(\boldsymbol{\Delta}x \oplus \boldsymbol{\Delta}y)t} \mathbf{Q} e^{-j(\boldsymbol{\Delta}x \oplus \boldsymbol{\Delta}y)t} \mathbf{c}(x, y). \end{aligned} \tag{49}$$

Let $\boldsymbol{\Delta} = \boldsymbol{\Delta}x \oplus \boldsymbol{\Delta}y$ and $\mathbf{c} = \mathbf{c}(x, y)$ to simplify notation. The terms necessary to compute the Taylor series expansion for the numerator function out to second order are

$$\begin{aligned} N(\mathbf{p} + t\mathbf{d})|_{t=0} &= \mathbf{c}^H \mathbf{W} \mathbf{c} \\ \frac{d}{dt} N(\mathbf{p} + t\mathbf{d})\bigg|_{t=0} &= j\mathbf{c}^H [\boldsymbol{\Delta}, \mathbf{W}] \mathbf{c} \\ \frac{d^2}{dt^2} N(\mathbf{p} + t\mathbf{d})\bigg|_{t=0} &= -\mathbf{c}^H [\boldsymbol{\Delta}, [\boldsymbol{\Delta}, \mathbf{W}]] \mathbf{c}. \end{aligned} \tag{50}$$

Combining terms as in (47) yields

$$N(\mathbf{p} + \mathbf{d}) = \mathbf{c}^H \left[\mathbf{W} + j [\boldsymbol{\Delta}, \mathbf{W}] - \frac{1}{2} [\boldsymbol{\Delta}, [\boldsymbol{\Delta}, \mathbf{W}]] + \ldots \right] \mathbf{c}. \tag{51}$$

Define the function

$$G(t) = \frac{1}{D(\mathbf{p} + t\mathbf{d})} = \frac{1}{\mathbf{c}^H e^{j\boldsymbol{\Delta}t} \mathbf{Q} e^{-j\boldsymbol{\Delta}t} \mathbf{c}}. \tag{52}$$

The derivatives necessary to compute the Taylor series expansion for $G(t)$ are

$$\begin{aligned} G'(t) &= -[D(\mathbf{p} + t\mathbf{d})]^{-2} \frac{d}{dt} D(\mathbf{p} + t\mathbf{d}), \\ G''(t) &= 2[D(\mathbf{p} + t\mathbf{d})]^{-3} \left[\frac{d}{dt} D(\mathbf{p} + t\mathbf{d}) \right]^2 \\ &\quad - [D(\mathbf{p} + t\mathbf{d})]^{-2} \frac{d^2}{dt^2} D(\mathbf{p} + t\mathbf{d}), \end{aligned} \tag{53}$$

which yields

$$\begin{aligned} G'(0) &= -j \frac{\mathbf{c}^H [\boldsymbol{\Delta}, \mathbf{Q}] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2}, \\ G''(0) &= \frac{\mathbf{c}^H [\boldsymbol{\Delta}, [\boldsymbol{\Delta}, \mathbf{Q}]] \mathbf{c}}{(\mathbf{c}\mathbf{Q}\mathbf{c})^2} - \frac{2(\mathbf{c}^H [\boldsymbol{\Delta}, \mathbf{Q}] \mathbf{c})^2}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3}. \end{aligned} \tag{54}$$

After combining terms, the Taylor series expansion of $1/D(\mathbf{p} + \mathbf{d})$ becomes

$$\begin{aligned} \frac{1}{D(\mathbf{p} + \mathbf{d})} &= G(0) + G'(0) + \frac{1}{2} G''(0) + \ldots \\ &= \frac{1}{\mathbf{c}^H \mathbf{Q} \mathbf{c}} - j \frac{\mathbf{c}^H [\boldsymbol{\Delta}, \mathbf{Q}] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2} \\ &\quad - \frac{\mathbf{c}^H [\boldsymbol{\Delta}, [\boldsymbol{\Delta}, \mathbf{Q}]] \mathbf{c}}{2(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2} - \frac{(\mathbf{c}^H [\boldsymbol{\Delta}, \mathbf{Q}] \mathbf{c})^2}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3} + \ldots. \end{aligned} \tag{55}$$

Multiplying together the Taylor series expansions for $N(\mathbf{p} + \mathbf{d})$ and $1/D(\mathbf{p} + \mathbf{d})$ and collecting second order terms yields

$$\begin{aligned} J(\mathbf{p} + \mathbf{d}) &= \frac{1}{D(\mathbf{p} + \mathbf{d})} \cdot N(\mathbf{p} + \mathbf{d}) \approx \\ &- \frac{\mathbf{c}^H [\boldsymbol{\Delta}, [\boldsymbol{\Delta}, \mathbf{W}]] \mathbf{c}}{2(\mathbf{c}^H \mathbf{Q} \mathbf{c})} + \\ &\frac{\frac{1}{2}(\mathbf{c}^H \mathbf{W} \mathbf{c})(\mathbf{c}^H [\boldsymbol{\Delta}, [\boldsymbol{\Delta}, \mathbf{Q}]] \mathbf{c}) + (\mathbf{c}^H [\boldsymbol{\Delta}, \mathbf{W}] \mathbf{c})(\mathbf{c}^H [\boldsymbol{\Delta}, \mathbf{Q}] \mathbf{c})}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2} \\ &- \frac{(\mathbf{c}^H \mathbf{W} \mathbf{c})(\mathbf{c}^H [\boldsymbol{\Delta}, \mathbf{Q}] \mathbf{c})^2}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3} + \text{other terms}. \end{aligned} \tag{56}$$

Recall,

$$\begin{aligned} \boldsymbol{\Delta} &= \boldsymbol{\Delta}x \oplus \boldsymbol{\Delta}y = \delta_x \mathbf{T}_u \oplus \delta_y \mathbf{T}_v \\ &= \boldsymbol{\Delta}x \otimes \mathbf{I} + \mathbf{I} \otimes \boldsymbol{\Delta}y = \delta_x \mathbf{T}_u \otimes \mathbf{I} + \mathbf{I} \otimes \delta_y \mathbf{T}_v. \end{aligned} \tag{57}$$

Substituting (57) into (56) and simplifying produces

$$
\begin{aligned}
J(\mathbf{p}+\mathbf{d}) \approx \delta_x^2 \Bigg[& -\frac{\mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, [\mathbf{T}_u \otimes \mathbf{I}, \mathbf{W}]\right] \mathbf{c}}{2(\mathbf{c}^H \mathbf{Q} \mathbf{c})} \\
& - \frac{(\mathbf{c}^H \mathbf{W} \mathbf{c})(\mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, \mathbf{Q}\right] \mathbf{c})^2}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3} \\
& + \frac{(\mathbf{c}^H \mathbf{W} \mathbf{c}) \mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, [\mathbf{T}_u \otimes \mathbf{I}, \mathbf{Q}]\right] \mathbf{c}}{2(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2} \Bigg] \\
& + \delta_y^2 \Bigg[-\frac{\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, [\mathbf{I} \otimes \mathbf{T}_v, \mathbf{W}]\right] \mathbf{c}}{2(\mathbf{c}^H \mathbf{Q} \mathbf{c})} \\
& + \frac{\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, [\mathbf{I} \otimes \mathbf{T}_v, \mathbf{Q}]\right] \mathbf{c}}{(\mathbf{c} \mathbf{Q} \mathbf{c})^2} \\
& - \frac{(\mathbf{c}^H \mathbf{W} \mathbf{c})(\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, \mathbf{Q}\right] \mathbf{c})^2}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3} \Bigg] \\
& + \delta_x \delta_y \Bigg[-\frac{\mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, [\mathbf{I} \otimes \mathbf{T}_v, \mathbf{W}]\right] \mathbf{c}}{2(\mathbf{c}^H \mathbf{Q} \mathbf{c})} \\
& - \frac{\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, [\mathbf{T}_u \otimes \mathbf{I}, \mathbf{W}]\right] \mathbf{c}}{2(\mathbf{c}^H \mathbf{Q} \mathbf{c})} \\
& + \frac{\mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, [\mathbf{I} \otimes \mathbf{T}_v, \mathbf{Q}]\right] \mathbf{c}}{(\mathbf{c} \mathbf{Q} \mathbf{c})^2} \\
& + \frac{\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, [\mathbf{T}_u \otimes \mathbf{I}, \mathbf{Q}]\right] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2} \\
& - \frac{2(\mathbf{c}^H \mathbf{W} \mathbf{c}) \mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, \mathbf{Q}\right] \mathbf{c} \mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, \mathbf{Q}\right] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3} \Bigg] \\
& + \text{other terms}.
\end{aligned} \tag{58}
$$

Equation (58) can be re-written in the form

$$
\begin{aligned}
J(\mathbf{p}+\mathbf{d}) &\approx \mathbf{d}^T \mathbf{H}(\mathbf{p}) \mathbf{d} + \text{ other terms} \\
&= [\delta_x \ \delta_y]^T \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_y \end{bmatrix} \\
&\quad + \text{other terms}
\end{aligned} \tag{59}
$$

where the elements of the desired Hessian matrix $\mathbf{H}(\mathbf{p})$ are given by

$$
\begin{aligned}
H_{11} = & -\frac{\mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, [\mathbf{T}_u \otimes \mathbf{I}, \mathbf{W}]\right] \mathbf{c}}{\mathbf{c}^H \mathbf{Q} \mathbf{c}} \\
& - \frac{2(\mathbf{c}^H \mathbf{W} \mathbf{c})(\mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, \mathbf{Q}\right] \mathbf{c})^2}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3} \\
& + \frac{(\mathbf{c}^H \mathbf{W} \mathbf{c}) \mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, [\mathbf{T}_u \otimes \mathbf{I}, \mathbf{Q}]\right] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2},
\end{aligned} \tag{60}
$$

$$
\begin{aligned}
H_{12} = & -\frac{\mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, [\mathbf{I} \otimes \mathbf{T}_v, \mathbf{W}]\right] \mathbf{c}}{\mathbf{c}^H \mathbf{Q} \mathbf{c}} \\
& + \frac{2\mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, [\mathbf{I} \otimes \mathbf{T}_v, \mathbf{Q}]\right] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2} \\
& - \frac{2(\mathbf{c}^H \mathbf{W} \mathbf{c}) \mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, \mathbf{Q}\right] \mathbf{c} \mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, \mathbf{Q}\right] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3},
\end{aligned} \tag{61}
$$

$$
\begin{aligned}
H_{21} = & -\frac{\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, [\mathbf{T}_u \otimes \mathbf{I}, \mathbf{W}]\right] \mathbf{c}}{\mathbf{c}^H \mathbf{Q} \mathbf{c}} \\
& + \frac{2\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, [\mathbf{T}_u \otimes \mathbf{I}, \mathbf{Q}]\right] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2} \\
& - \frac{2(\mathbf{c}^H \mathbf{W} \mathbf{c}) \mathbf{c}^H \left[\mathbf{T}_u \otimes \mathbf{I}, \mathbf{Q}\right] \mathbf{c} \mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, \mathbf{Q}\right] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3},
\end{aligned} \tag{62}
$$

$$
\begin{aligned}
H_{22} = & -\frac{\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, [\mathbf{I} \otimes \mathbf{T}_v, \mathbf{W}]\right] \mathbf{c}}{\mathbf{c}^H \mathbf{Q} \mathbf{c}} \\
& + \frac{2\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, [\mathbf{I} \otimes \mathbf{T}_v, \mathbf{Q}]\right] \mathbf{c}}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^2} \\
& - \frac{2(\mathbf{c}^H \mathbf{W} \mathbf{c})(\mathbf{c}^H \left[\mathbf{I} \otimes \mathbf{T}_v, \mathbf{Q}\right] \mathbf{c})^2}{(\mathbf{c}^H \mathbf{Q} \mathbf{c})^3}.
\end{aligned} \tag{63}
$$

D. Newton's Method

Using the derived Hessian matrix, Newton's method for solving the optimization program in (27) becomes

Algorithm 3 Newton's Method

Require: kth array element initial positions x_0 and y_0

1: Set the initial search direction $\mathbf{d}_0 = \mathbf{H}(x_0, y_0)^{-1} \nabla J(x_0, x_0)$
2: Until $\|\nabla J(x_j, y_j)\|_2 \leq \epsilon$, where j denotes iteration index, do the following:
3: Determine the step-size μ_j using a line-search method or set $\mu_j = 1$ for all j
4: Set $\mathbf{p}_{j+1} = \mathbf{p}_j + \mu_j \mathbf{d}_j$ where $\mathbf{p}_j = [x_j \quad y_j]^T$
5: Set $\mathbf{g}_{j+1} = \nabla J(x_{j+1}, y_{j+1})$
6: Set $\mathbf{d}_{j+1} = \mathbf{H}(x_{j+1}, y_{j+1})^{-1} \mathbf{g}_{j+1}$
7: Set $j = j + 1$
8: Repeat steps 1-7 for all array elements

IV. Part B. Optimal Adaptive Beamformer

In this section, we derive a beamformer solution that can be used to solve for complex weights to be applied to the sparse array elements in their new optimized locations to further reduce beampattern sidelobe levels. Consider the MSE cost function

$$
\begin{aligned}
& \min_{\mathbf{w}} \quad \|\mathbf{w}^H \mathbf{A} - \mathbf{f}\|^2 \\
& = \mathbf{w}^H \mathbf{A} \mathbf{A}^H \mathbf{w} - \mathbf{w}^H \mathbf{A} \mathbf{f}^H - \mathbf{f} \mathbf{A}^H \mathbf{w} + \mathbf{f} \mathbf{f}^H \\
& = \begin{bmatrix} 1 & \mathbf{w}^H \end{bmatrix} \begin{bmatrix} \mathbf{f} \mathbf{f}^H & -\mathbf{f} \mathbf{A}^H \\ -\mathbf{A} \mathbf{f}^H & \mathbf{A} \mathbf{A}^H \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{w} \end{bmatrix} \\
& \equiv \begin{bmatrix} 1 & \mathbf{w}^H \end{bmatrix} \mathbf{K} \begin{bmatrix} 1 \\ \mathbf{w} \end{bmatrix}.
\end{aligned} \tag{64}
$$

The 1-by-L vector $\mathbf{f}$ consists of complex filled array beampattern samples from the sidelobe region. The matrix A is the M-by-L manifold matrix corresponding to the sparse array whereby each column corresponds to a steering vector for a specified direction.

Applying a unity gain constraint on the mainbeam yields

$$
\begin{aligned}
& \min_{\mathbf{w}} \quad \begin{bmatrix} 1 & \mathbf{w}^H \end{bmatrix} \mathbf{K} \begin{bmatrix} 1 \\ \mathbf{w} \end{bmatrix} \\
& \text{such that} \quad \mathbf{w}^H \mathbf{a} = 1,
\end{aligned} \tag{65}
$$

where $\mathbf{a}$ is the steering vector corresponding to the desired mainbeam pointing direction. The linear constraint $\mathbf{w}^H\mathbf{a} = 1$ can be rewritten as,

$$\begin{bmatrix} 1 & \mathbf{0}^T \\ 0 & \mathbf{a}^H \end{bmatrix} \begin{bmatrix} 1 \\ \mathbf{w} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \equiv \mathbf{C}\widehat{\mathbf{w}} = \mathbf{b}. \tag{66}$$

The well-known global solution to the quadratic optimization program,

$$\min_{\widehat{\mathbf{w}}} \quad \widehat{\mathbf{w}}^H\mathbf{K}\widehat{\mathbf{w}} \quad \text{such that} \quad \mathbf{C}\widehat{\mathbf{w}} = \mathbf{b} \tag{67}$$

is given by

$$\widehat{\mathbf{w}} = \mathbf{K}^{-1}\mathbf{C}^H(\mathbf{C}\mathbf{K}^{-1}\mathbf{C}^H)^{-1}\mathbf{b}. \tag{68}$$

V. Numerical Results

This section describes simulated results that demonstrate the effectiveness of the proposed algorithm. We consider a sparse lattice with 81 spatial samples spaced at intervals 2λ apart on a 9-by-9 grid. Our goal is to reduce the sparse array grating lobes by at least 13 dB relative to the peak so as to be no worse than the sidelobes of a uniformly weighted filled array that has the same length and width dimensions as the sparse array.

The alternating projections algorithm proceeds by sampling the filled-array beampattern which also has a complex taper applied to reduce the ambient sidelobe level as shown in Fig. 1. Fig. 2 illustrates the trajectory of array element 1 as it moves across the xy-plane of the aperture according to the conjugate gradient iterations of the algorithm. Fig. 3 shows the final optimized array element locations after 16 passes through the entire array where each pass optimizes one element at a time. Fig. 4 illustrates the final beampattern at 40 GHz after the array element locations have been optimized and the adaptive beamformer described in Section IV has been applied. Fig. 5 illustrates a U-cut of the optimized beampattern and shows a reduction in grating lobe power by 14 dB relative to the peak of the mainbeam. Likewise, there is a reduction in grating lobe power by 12.9 dB in the V-principal plane.

A. Experimental Validation

In this section, we show the results of deploying the proposed approach in hybrid chamber measurements where a synthetic aperture is used in the setup detailed in [13]. In Fig. 6, we plot the measured grating lobes in the received power pattern with spatial aliasing due to the use of a 9-by-9 uniform sparse array. Note that the mainbeam and the grating lobes are wider than theoretical due to the impact of the measurement setup. Similarly, we also show the measured received power pattern of a 35-by-35 filled array in Fig. 6. The measured data in this case are post-processed using the same taper as in the simulated case to reduce the sidelobe levels. Finally, the optimized sparse array is used to get the received power pattern in Fig. 7 which validates the theoretical numerical results.

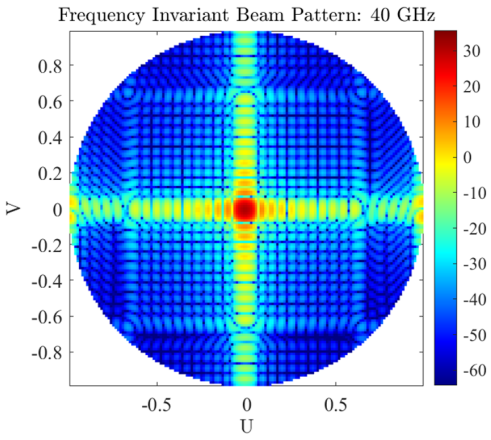

Fig. 1: Received power pattern of the filled array

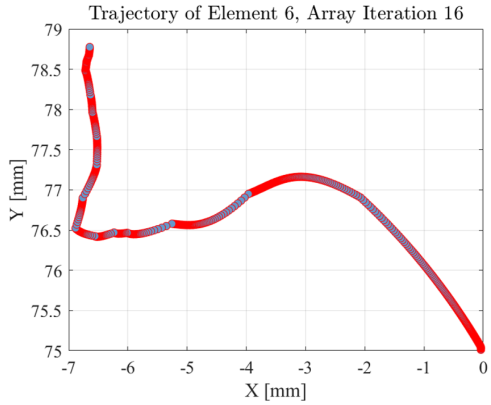

Fig. 2: Position trajectory of array element 1

VI. Conclusions

This paper derives a novel gradient implementation of an alternating projections algorithm followed by an adaptive beamformer. First, the AP algorithm optimizes the location of spatial samples in a sparse array lattice to mitigate the peak level of grating lobes. Then, the adaptive beamformer calculates an aperture taper that further reduces the overall sidelobe level. Simulation results show peak grating lobes in the optimized beampattern that are 14 and 12.9 dB below the

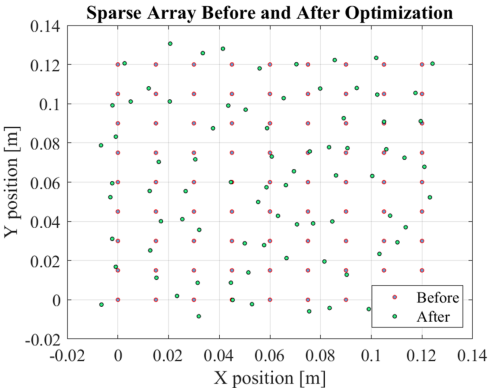

Fig. 3: Optimized sparse array lattice