A Fast Algorithm to Solve Delay Vandermonde Systems in Phased-Array Digital Receivers

Sirani M. Perera, Arjuna Madanayake, *Member, IEEE,* Austin Ogle, Daniel Silverio and Jacky Q. Huang,

***Abstract*—Phased-array multi-beam RF beamformers require calibration of receivers used in an array of elements before the signals can be applied to an analog beamforming network. This paper presents a fast $\mathcal{O}(n^2)$ algorithm for solving linear systems having a delay Vandermonde matrix (DVM) for the analog beamforming matrix. The structure of the DVM enables to obtain a fast algorithm to efficiently reverse multi-beams at the DVM output such that calibration of the input low noise amplifiers can be achieved.**

The arithmetic complexity of the proposed DVM system solving algorithm is preferable over the standard matrix inversion consuming $\mathcal{O}(n^3)$ operations. Numerical experiments are presented for forward accuracy of the proposed algorithm computed at the calibration frequency. Moreover, numerical results are shown to compare the order of the arithmetic complexity and the execution time of the proposed algorithm. Signal flow graphs are presented for 4- and 8-element multi-beam phased-arrays.

***Index Terms*—Matrix inversion, Sparse matrices, Complexity of algorithms, Performance of algorithms, Discrete Fourier transform, Fast algorithms, Antenna arrays, Phased-array radar, Millimeter wave communication.**

NOTATION AND TERMINOLOGY

We introduce the following notations and terminologies.

n number of elements in array
$x(t) \in \mathbb{R}$ scalar time function
$\mathbf{x}(t)$ vector of time functions
T inter-sample period
$\mathbf{x}(k)$ vector at discrete time index k where $t = kT$
$\mathbf{A}$ linear transform matrix
$Q(x(t))$ quantization of $x(t)$
$Q(\mathbf{x})(t)$ quantization of vector $\mathbf{x}(t)$
$Q(\mathbf{x})(k)$ sampled and quantized vector
$Q(\mathbf{Ax})(t)$ quantized vector-matrix product
$Q(\mathbf{Ax})(k)$ sampled and quantized vector-matrix product
$V(\alpha)$ Vandemonde matrix defined by nodes α^k
f time frequency
ω circular frequency
g_i gain of receiver number $i = 1, 2, ..., n$
g_i^{-1} calibration coefficients
$\#a\mathbb{C}$ number of complex additions
$\#m\mathbb{C}$ number of complex multiplications
$\#a\mathbb{R}$ number of real additions
$\#m\mathbb{R}$ number of real multiplications

S. M. Perera is with the Department of Mathematics, Embry-Riddle Aeronautical University, Daytona Beach, FL, 32703 USA e-mail: pereras2@erau.edu (see https://faculty.erau.edu/Sirani.Perera).

A. Madanayake is with Florida International University.

A. Ogle, D. Silverio, and J. Q. Huang were with Embry-Riddle Aeronautical University.

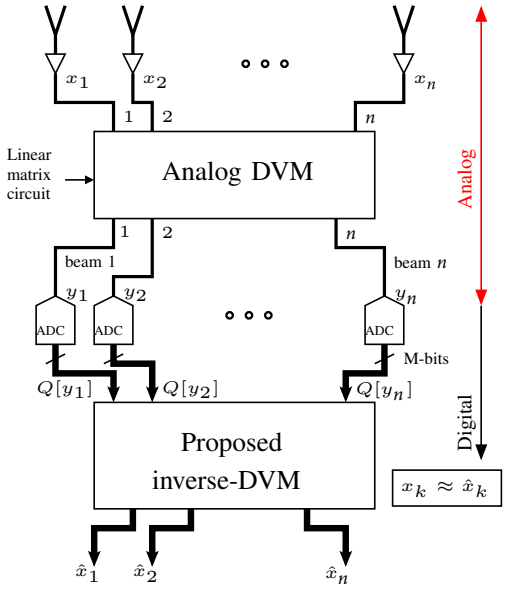

Fig. 1: Analog multi-beam DVM beamformer and digital inverse-DVM computational architecture where an array of antennas are amplified and beamformed using an analog DVM circuit. The DVM system is solved using a fast algorithm implemented in a systolic array processor.

I. INTRODUCTION

Emerging communication networks, such as 5G/6G wireless systems depend on high signal-to-noise ratio (SNR) channels for desired high data rates (10-100 Gbps, typically) [1]. A key technology for improving the SNR of mm-wave wireless channels is the use of analog-digital hybrid beamformers [1]–[4]. The delay Vandemonde matrix is an $n \times n$ matrix $V(\alpha)$ describing a network of true-time-delay linear combinations of the antenna outputs that are functions of $\alpha \in \mathbb{C}, |\alpha| = 1$ that is suitable for analog realization of n simultaneous beams at low complexity [1]–[5].

Fig. 1 shows the architecture of the proposed analog-DVM n-beam beamformer followed by a digital inverse-DVM algorithm realized within a software implementation that allows calibration of the individual receiver gains after analog-DVM beamforming has been realized. Calibration of the array receiver gains is important due to component variations, fabrication tolerances, and environmental factors that affect the receiver gains causing them to deviate from their expected values.

The use of analog realizations of the DVM via a suitable

fast algorithm implemented in analog radio-frequency integrated circuits (RF-IC) potentially allows the improvement of the channel SNR along n simultaneous directions of wave propagation. Alternatively, a passive Rotman lens microwave circuit can be applied to approximate the analog-DVM multi-beam beamformer [9].

Unlike fully-digital multi-beam beamformers [10]–[19], the analog-DVM based analog-digital hybrid multi-beam approach achieves sub-array beamforming in analog/microwave domain using active or passive realization of the DVM matrix, followed by secondary digital beamforming when needed. In fully-digital multi-beam beamforming systems, the number of antennas is equal to the number of degrees of freedom; however, single-beam phased-array analog-digital hybrid systems allow trading-off of degrees of freedom in return for power efficient analog/microwave domain sub-array beamforming. In analog DVM multi-beam systems, there is no loss of degrees of freedom because N-elements can provide up to N-orthogonal beams. However, only the required DVM beams will be sampled by their corresponding ADCs thus enabling receiving of particular beams as and when needed.

The analog/microwave DVM circuit (or alternatively, a passive Rotman lens) realizes a matrix-vector multiplication $\mathbf{y} = \mathbf{Ax}$ where $\mathbf{x} = [x_1(t)\ x_2(t)\ ...\ x_n(t)]^T$ is the vector of antenna array outputs obtained from the receivers, $\mathbf{A}$ is the $n-$ beam beamformer matrix (e.g., DVM matrix) and $\mathbf{y} = [y_1(t)\ y_2(t)\ ...\ y_n(t)]^T$ is the beam outputs that are to be sampled by $n-$ number of ADCs. An overview of an analog DVM 4-beam multi-beam beamformer is shown in Fig. 2. The transmission line segments are used to achieve $n = 4$ fixed beams. The increase of the SNR along the n directions. The ADCs quantize (operation $Q(\cdot)$) the beamformed analog signal to $m-$bits (in a uniformly quantized flash ADC) and then sample to discrete time at sample rate $F_s = T^{-1}$ for sample period T, resulting in the vector of digitized beams (denoted $[Q(y_1(kT))\ Q(y_2(kT))\ ...\ Q(y_n(kT))]^T$ for time index $k \in \mathbb{Z}^+$ that are used in the software defined radio (SDR) for demodulation and error control, with unpacked raw data being moved to the upper layers of the communications protocol for subsequent processing.

Non-ideal gains in the receivers modify the model to vector of analog time functions $\mathbf{y}(t) = \mathbf{AGx}(t)$ where $\mathbf{G}$ is an $n \times n$ diagonal matrix that contains the gains $g_1, g_2, ..., g_n$ of the receivers. Ideally, $\mathbf{G} = \mathbf{I}$ but in practice, the elemental receiver gains may differ by several dB making $\mathbf{G} \neq \mathbf{I}$. These deviations come from physical properties and fabrication non-idealities of the antennas and receiver electronics that cannot be avoided. Calibration is the operation that introduces a set of n software programmable amplifiers that have gains $\mathbf{G}^{-1}$ such that the output vector of time functions applied to ADCs becomes $\mathbf{y}(t = kT) = \mathbf{AG^{-1}Gx}(t = kT)$ for which the inverse matrix is also a diagonal matrix having elements $g_1^{-1}, g_2^{-1}, ..., g_n^{-1}$, and the sampled beams become discrete time sequences $Q(\mathbf{Ax})(k)$ for every $k \in \mathbb{Z}$. The beamformer has to be calibrated periodically to ensure all of the RF channels have the expected gain before reaching the inputs of the analog DVM network. However, since digitization occurs after analog beamforming modeled by $\mathbf{y}(t) = \mathbf{AGx}(t)$, it is not possible to isolate antenna channels $\mathbf{Gx}(t)$ at ADC inputs, at times $t = kT$, without inverting the analog-DVM matrix in a digital signal processing (DSP) step that in turn gives $A^{-1}Q(\mathbf{AGx})(k)$. For high-precision ADCs (e.g., $m \geq$12 bits) and the above non-linear matrix operation is reasonably close to the desired linear matrix operation given in a time sequence $\mathbf{A}^{-1}\mathbf{AGx}(k)$ that in turn furnishes the vector time sequence $\approx \mathbf{Gx}(k)$ needed for calibrating receivers.

The multi-beamformers resulting in an analog DVM are wideband in nature [5]–[8]. However, the gain controls for each antenna/receiver consists of software controlled amplifiers resulting in the desired calibration $\mathbf{G}^{-1}$ where each elemental calibration gain is flat for the frequency band of interest. Therefore, the calibration plane-wave signals are assumed to be a tone centered the dominant frequency of interest, such as the system carrier frequency in a wireless communication system. For purposes of calibration, the input signal are therefore narrowband, and consists of sinusoidal tones being presented to the antenna array and analog DVM beamformer [5]–[8].

In this paper, we solve the problem of recovering the original analog input signals $Q(\mathbf{AGx})$, albeit with some inaccuracies $\mathbf{A}^{-1}\nu$ due to the prevalence of ADC signal quantization in the signal path, and where ν is the quantization noise time-function vector, in discrete-time domain given as $\nu = \mathbf{AGx} - Q(\mathbf{AGx})$, where $t = Tk$. In general, the larger the number of bits used in the ADC, the more reasonable the results of the DVM inverse operation (i.e., effectively, the multiplication of ADC output sampled-time function vector is $Q(\mathbf{AGx})$ by A^{-1} (at time index k) to get us an approximation of analog time function vector $\mathbf{Gx}$) as achieved in the proposed fast algorithm. The paper covers the algorithm for inverting the DVM from a mathematical standpoint. The algorithm is implemented in software as the intended platform is SDR.

In section II, we state background information on DVM-based multibeam beamforming, and historical results in solving Vandermonde systems. In section III, we will present a factorization of the inverse of the DVM. In section IV, we will state the algorithm for solving the corresponding linear system of equations. Next, in section V, numerical results are presented for the forward error bound of the proposed algorithm. Within the same section execution time of the proposed algorithm is presented having nodes of the DVM matrix on the unit circle. In section VI, we present signal flow graphs of the proposed algorithms having 4-point and 8-point inputs. Finally section VII concludes the paper.

II. DVM-Based Multi-Beam Arrays

The DVM is the super class of the Discrete Fourier Transform (DFT) matrix and can be used to realize wideband and narrowband multi-beam systems [5]–[8]. Compared with the DFT matrix, the DVM also has nodes on the unit circle. But the nodes of the DVM are not the primitive roots of unity, and so the DVM is lacking unitary and periodicity properties. Therefore we won't be able to compute the inverse of the DVM in the way that we compute the inverse DFT (which is

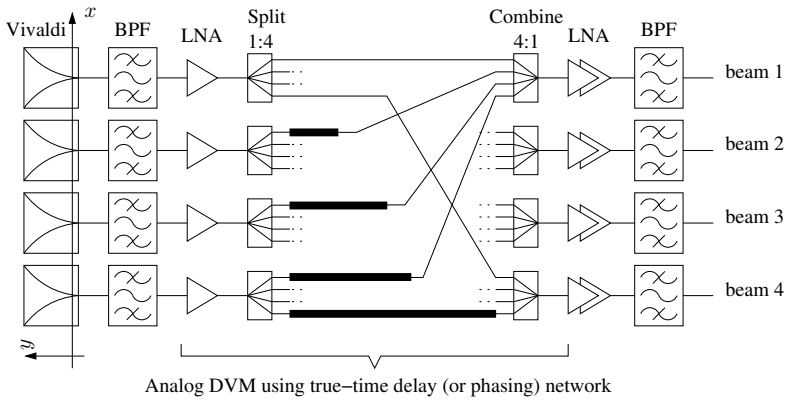

Fig. 2: Analog DVM circuit. Shown here is a typical 4-element 4-beam analog DVM multi-beam beamformer architecture. The true-time delays - shown here as impedance matched transmission line segments - achieves wideband performance that is optimal for wide sense stationary additive white Gaussian noise. For the narrowband case, true-time delays can be replaced with analog phasing elements resulting in an analog phased-array architecture.

the conjugate transpose). On the other hand, DFT and DVM are Vandermonde structured matrices. Thus, we will use the Vandermonde structure of the DVM to derive a fast algorithm to solve the DVM system, and then compare the numerical results with the DFT systems. In the linear system $\mathbf{y} = \mathbf{AGx}$, we define the DVM s.t. $\mathbf{A} = V(\alpha) = \left[\alpha^{ik}\right]_{i,k=0}^{n-1,n-1}$ where $\alpha = e^{-j\omega_t\tau}$ are the functions of temporal frequency variable ω_t s.t. $\omega_t = 2\pi f$ with frequency f, delay τ, and $j^2 = -1$.

A. DVM Model for True-Time Delay Multi-Beam Beamformer

The DVM can process wideband signals via the use of true-time-delays in the forward DVM. However, for narrowband signals such as 5G wireless systems, true time delays can be replaced with phase-shifters [20], [21]. The coefficients α^k take the form of a complex function of frequency $e^{-j\omega k\tau}$ where there are k delays, each of duration τ in place of the regular complex coefficients used in a matrix. The coefficients are therefore functions of the temporal frequency ω. The forward DVM matrix is realized in analog domain using microwave/mm-wave circuit components, such as networks of transmission lines or all-pass filter networks. This paper assumes the special case, which is common in practice, where the modulated signals are bandpass and are centered at a particular carrier frequency ω_c. When the input array signal corresponds to a calibration tone, the DVM forward coefficients can be approximated by constants $\alpha^k \in \mathbb{C}$ where the temporal frequency is the *constant* $\omega = 2\pi f$.

When the forward DVM consists of such constants (despite being implemented as true-time-delay circuits capable of processing wideband signals) due to calibration tone having frequency f Hz, the inverse-DVM operation can be realized in the discrete domain using numerical techniques. The special case where the forward DVM matrix is processing a calibration tone can be handled using a linear system solver. In the paper, we describe an efficient realization of DVM inverse using numerical techniques.

B. Prior Mathematical Work in Solving Vandermonde Systems

It is well known that the Gaussian elimination can be used to solve the system of linear equations, having a coefficient matrix that is the classical Vandermonde matrix with nodes $\{d_1, d_2, \cdots, d_n\}$, of the form:

$$\begin{bmatrix} 1 & d_1 & d_1^2 & \cdots & d_1^{n-1} \\ 1 & d_2 & d_2^2 & \cdots & d_2^{n-1} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & d_n & d_n^2 & \cdots & d_n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

$$V \cdot \mathbf{x} = \mathbf{y}. \tag{1}$$

If we didn't consider the structure of the matrix V, then the Gaussian elimination to solve the system (1) would cost $\mathcal{O}(n^3)$ arithmetic operations. On the other hand, the condition number of the matrix V rapidly increases with size, leading to instability of the system [22], [23]. Thus, it was shown by Björck-Pereyra (see e.g. [24]–[26]) that one can explore the structure of V to speed-up the computation i.e. to solve the system in $\mathcal{O}(n^2)$ operations. Further, it was shown that the Björck-Pereyra algorithm is not only faster, but it is often more accurate than the standard methods [27] for the forward stability and [28] for the backward stability analyses.

A more general class of the matrix V having the Vandermonde structure is made by replacing the monomials (note that, the matrix V can be generated by the monomials $\{1, d, d^2, \cdots, d^{n-1}\}$ evaluated at the roots $\{d_1, d_2, \cdots, d_n\}$ of $p_n(d)$) with the set of polynomials $\mathbf{P} := \{p_0(d), p_1(d), p_2(d), \cdots, p_{n-1}(d), p_n(d)\}$, and is called the polynomial Vandermonde matrix, say $V_{\mathbf{P}}$. This matrix has the form:

$$V_{\mathbf{P}} = \begin{bmatrix} p_0(d_1) & p_1(d_1) & p_2(d_1) & \cdots & p_{n-1}(d_1) \\ p_0(d_2) & p_1(d_2) & p_2(d_2) & \cdots & p_{n-1}(d_2) \\ \vdots & \vdots & \vdots & & \vdots \\ p_0(d_n) & p_1(d_n) & p_2(d_n) & \cdots & p_{n-1}(d_n) \end{bmatrix}, \tag{2}$$

where $\{d_1, d_2, \cdots, d_n\}$ are the roots of the polynomial $p_n(d)$. Thus, the Björck-Pereyra algorithm for solving a linear system of equations corresponding to monomials can be extended

to a polynomial system of equations. Table I provides such fast algorithms to solve systems of linear equations covering special cases of the system $\mathbf{P}$.

TABLE I: Fast $\mathcal{O}(n^2)$ algorithms for solving linear systems

Coefficient Matrix	Algorithms solving the system $\mathbf{P}$
Classical Vandermonde matrix	Björck-Pereyra algorithm [24]
Chebyshev Vandermonde matrix	Reichel-Opfer algorithm [29]
Three-term Vandermonde matrix	Higham algorithm [30]
Szegö-Vandermonde matrix	Bella et al. algorithm [31]
Quasiseparable Vandermonde matrix	Bella and et al. algorithm [32]

In this paper, we propose a fast algorithm to solve a system linear of equations having coefficient matrix that is the DVM when $V(\alpha)$ is generated by the set of distinct complex nodes $\{1, \alpha, \alpha^2, \cdots, \alpha^{n-1}\}$.

The receiver outputs are approximately computed by the proposed algorithm, which provides the outputs as $V(\alpha)^{-1}Q(V(\alpha)\mathbf{Gx})+V(\alpha)^{-1}\nu$, where $V(\alpha)^{-1}\nu$ is the computational error vector due to quantization at the ADCs. In our analysis, we assume ideal time sampling with no quantization non-linearity in the ADC in order to explore the numerical properties of the proposed algorithm in its "best case" form; that is we assume the ideal (linearized and quantization noise-free) output vector $V(\alpha)^{-1}V(\alpha)\mathbf{Gx}$ - to be computed repeatedly for every time sample k. The resulting inverse-DVM serves as an upper bound on the eventual performance of the total system when ADC quantization non-linearity is present. In general, the number of bits in the ADCs are chosen to support the SNR of the beams, and this is considered sufficient for enabling calibration of the receiver gains.

III. Factorization for the Inverse of the Delay Vandermonde Matrix

In this section, we present a sparse factorization for the inverse delay Vandermonde matrix (based on a set of complex nodes $\{1, \alpha, \alpha^2, \cdots, \alpha^{n-1}\}$) analogous to the classical Björck-Pereyra algorithm (stated in Appendix A and based on a set of real nodes $\{d_1, d_2, \cdots, d_n\}$). It is known that the inverse of the Vandermonde matrices can be calculated efficiently by using the recurrence relations of Horner polynomials (stated in Appendix B). Thus, we use a similar approach to solve $V(\alpha) \cdot \mathbf{x} = \mathbf{y}$ while introducing a system of Horner polynomials. Let us say $\widehat{\mathbf{W}} = \{\hat{w}_0(\omega), \hat{w}_1(\omega), \ldots, \hat{w}_{n-1}(\omega), \hat{w}_n(\omega)\}$ (defined in Appendix B), is the system of Horner polynomials associated with the given system of polynomials $\mathbf{W} = \{1, \omega, \omega^2, \ldots, \omega^{n-1}, w_n(\omega)\}$, where $w_n(\omega) = (\omega - 1)(\omega - \alpha)(\omega - \alpha^2)\cdots(\omega - \alpha^{n-1})$. We use the Horner polynomials, the recurrence relations among the polynomials in the set $\widehat{\mathbf{W}}$ (proved in Appendix B, Proposition B.1), and an auxiliary result (proved in Appendix C) to solve the linear system $V(\alpha) \cdot \mathbf{x} = \mathbf{y}$.

In the following, we state a decomposition of the inverse of the delay Vandermode matrix. From now onwards, we consider all missing elements in any matrix as zeros entries.

Theorem III.1 Let $\mathbf{W} = \{1, \omega, \omega^2, \cdots, \omega^{n-1}\}$ be the system of complex-valued functions associated with the delay Vandermode matrix $V(\alpha)$. Furthermore, let $\alpha^{0:n-1} = \{1, \alpha, \alpha^2 \cdots, \alpha^{n-1}\}$, where $\alpha = e^{-j\omega_t\tau}$, $\omega_t = 2\pi f$, frequency f, delay τ, and $j^2 = -1$ be a set of n distinct points. Then the inverse of the DVM admits a recursive decomposition

$$\left[V(\alpha^{0:n-1})\right]^{-1} = \tilde{U}_1 \cdot \begin{bmatrix} 1 & 0 \\ 0 & \left[V(\alpha^{1:n-1})\right]^{-1} \end{bmatrix} \cdot \tilde{D}_1 \cdot \tilde{L}_1 \quad (3)$$

with

$$\tilde{U}_1 = \begin{bmatrix} 1 & -1 & & & \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & 1 & -1 \\ & & & & 1 \end{bmatrix}, \quad (4)$$

$$\tilde{D}_1 = \begin{bmatrix} 1 & & & & \\ & \frac{1}{\alpha-1} & & & \\ & & \frac{1}{\alpha^2-1} & & \\ & & & \ddots & \\ & & & & \frac{1}{\alpha^{n-1}-1} \end{bmatrix},$$

$$\text{and } \tilde{L}_1 = \begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & & 1 & & \\ \vdots & & & \ddots & \\ -1 & & & & 1 \end{bmatrix}.$$

Proof. Multiplying $V(\alpha^{0:n-1})$ by a sparse lower triangular matrix, we get

$$\begin{bmatrix} 1 & & & & \\ -1 & 1 & & & \\ -1 & & 1 & & \\ \vdots & & & \ddots & \\ -1 & & & & 1 \end{bmatrix} V(\alpha^{0:n-1})$$

$$= \begin{bmatrix} 1 & & & & \\ & \alpha-1 & & & \\ & & \alpha^2-1 & & \\ & & & \ddots & \\ & & & & \alpha^{n-1}-1 \end{bmatrix} \quad (5)$$

$$\left[\begin{array}{c|c} 1 & 0 \\ \hline 0 & \bar{\mathbf{W}} \end{array}\right] \left[\begin{array}{c|ccc} 1 & 1 & \cdots & 1 \\ \hline 0 & & I & \end{array}\right]$$

where the matrix $\bar{\mathbf{W}} = \left[\frac{\alpha^{ik}-1}{\alpha^i-1}\right]_{i,k=1}^{n-1}$ consists of divided differences. By the Appendix B (i.e. the definition of the complex-valued Horner polynomials of the given set of complex-valued functions $\mathbf{W}$ (14) and the Proposition B.1), the matrix $\bar{\mathbf{W}}$ is associated with the system of complex-valued Horner polynomials and can further be factored into:

$$\bar{\mathbf{W}} = V(\alpha^{1:n-1}) \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ & 1 & 1 & \cdots & 1 \\ & & \ddots & \ddots & \vdots \\ & & & 1 & 1 \\ & & & & 1 \end{bmatrix} \cdot \quad (6)$$

Using Lemma C.1 (showed in Appendix C) and inverting (5), we get the result. □

Corollary III.2 Inverse of the delay Vandermonde matrix $V(\alpha)^{-1}$ admits the factorization

$$V(\alpha)^{-1} = \tilde{U}_1\tilde{U}_2\cdots\tilde{U}_{n-1}\cdot\tilde{D}_{n-1}\tilde{L}_{n-1}\tilde{D}_{n-2}\tilde{L}_{n-2}\cdots\tilde{D}_1\tilde{L}_1 \tag{7}$$

where

$$\tilde{U}_m = \begin{bmatrix} I_{m-1} & & & & & \\ & 1 & -\alpha^{m-1} & & & \\ & & 1 & \ddots & & \\ & & & \ddots & -\alpha^{m-1} & \\ & & & & 1 & \end{bmatrix},$$

$$\tilde{D}_m = \begin{bmatrix} I_m & & & & \\ & \frac{1}{\alpha^m-\alpha^{m-1}} & & & \\ & & \ddots & & \\ & & & \frac{1}{\alpha^{n-1}-\alpha^{m-1}} & \end{bmatrix},$$

$$\text{and } \tilde{L}_m = \begin{bmatrix} I_{m-1} & & & & \\ & 1 & & & \\ & -1 & 1 & & \\ & \vdots & & \ddots & \\ & -1 & & & 1 \end{bmatrix} \tag{8}$$

for $m = 1, 2, \cdots n-1$.

Proof. The recursive nature of the formula (3) allows the decomposition (7). □

Remark III.3 In Corollary III.2, **(a).** α^k are complex numbers as opposed to real nodes in Appendix A. **(b).** The matrices $\tilde{L}_m$'s in (8) are different from the matrices L_k's in (12).

IV. DVM System Solving Algorithm and Complexity

This section introduces a fast algorithm based on the recursive factorization formula stated in Theorem III.1. After the algorithm is established we will derive the arithmetic complexity on computing the system of linear equations associated with the coefficient matrix $V(\alpha)$.

A. A Fast DVM System Solving Algorithm

In the following, we will state a fast algorithm to solve the delay Vandermonde system $V(\alpha)\cdot\mathbf{x} = \mathbf{y}$ by using the factorization in (7).

Algorithm IV.1 DVMS$(n, \tau, f,$ **y**$)$
Input: n, τ, f, and $\mathbf{y} \in \mathbb{R}^n$ or $\mathbb{C}^n$
Steps:

1) Set $\alpha = e^{-2\pi f\tau j}$, $s = 2n$, $\mathbf{v} = \begin{bmatrix}1 & \alpha & \cdots & \alpha^{n-1}\end{bmatrix}^T$, and $\mathfrak{A} = \begin{bmatrix}\mathbf{y} & 0_{n\times s-1}\end{bmatrix}$.

2) For $k = 1, 2, \cdots, n-1$,
   For $i = 1, 2, \cdots, n$,
   if $(i > k)$
   $\mathfrak{A}(i, k+1) = \frac{\mathfrak{A}(i,k)-\mathfrak{A}(k,k)}{\mathbf{v}(i)-\mathbf{v}(k)}$,
   else
   $\mathfrak{A}(i, k+1) = \mathfrak{A}(i, k)$.

3) For $k = n, n+1, \cdots, s-1$,
   For $i = 1, 2, \cdots, n$,
   if $(i < s-k)$ or $(i = n)$
   $\mathfrak{A}(i, k+1) = \mathfrak{A}(i, k)$,
   else
   $\mathfrak{A}(i, k+1) = \mathfrak{A}(i, k) - \mathfrak{A}(i+1, k)\cdot\mathbf{v}(s-k)$.

4) Take $\mathfrak{A}(:, s) = \mathbf{x}$.

Output: $\mathbf{x} = V(\alpha)^{-1}\cdot\mathbf{y}$

B. Arithmetic Complexity of the Algorithm

The arithmetic complexity, measured as the required number of additions and multiplications, needed to compute the proposed DVM system solving algorithm, will be derived in this section. Here counts are calculated based on the multiplication of two α having two real additions and four real multiplications, and subtraction of two complex numbers having two real additions. We do not count multiplication by ± 1.

Lemma IV.2 Let $n(\geq 4)$ be an integer. The complex arithmetic cost in computing the DVM system solving algorithm **DVMS**$(n, \tau, f, \mathbf{y})$ having $\mathbf{y} \in \mathbb{C}^n$ is given by

$$\begin{aligned} \#a\mathbb{C}(DVMS, n) &= \frac{3}{2}n(n-1), \\ \#m\mathbb{C}(DVMS, n) &= (n-1)^2. \end{aligned} \tag{9}$$

Proof. To set up the vector $\mathbf{v}$, through Step 1 of the algorithm, requires no complex additions and $\frac{n(n-1)}{2}$ complex multiplications. These counts are based on pre-computation and correlated with the design-time computations. Also, the aim of Step 1 is to compute the output of the algorithm effectively. In Step 2 of the algorithm, we calculate the product of $\tilde{D}_m$'s and $\tilde{L}_m$'s by a vector for $m = 1, 2, \ldots, n-1$. If $i \leq k$, there is no cost involved as it is an assignment of entries in $\mathfrak{A}$ from a column to the next column. When $i > k$, Step 2 involves $n(n-1)$ complex additions and $\frac{n(n-1)}{2}$ complex multiplications (this is calculated based on $|\alpha| = 1$). In Step 3 of the algorithm, we calculate the product of $\tilde{U}_m$'s by a vector for $m = 1, 2, \ldots, n-1$. If $i < m-k$ or $i = n$, there is no cost involved as it is an assignment of entries in $\mathfrak{A}$ from a column to the next column. The else part of Step 3 involves $\frac{n(n-1)}{2}$ complex additions and $\frac{n(n-1)}{2}$ complex multiplications for $k = n, n+1, \ldots, s-1$. But when $k = s-1$, this section involves no complex multiplication (i.e. multiplication by 1) for $i = 1, 2, \ldots, n-1$. Thus, Step 3 involves $\frac{n(n-1)}{2}$ complex additions and $\frac{(n-1)(n-2)}{2}$ complex multiplications. In Step 4 of the algorithm, we extract the last column of $\mathfrak{A}$, hence no cost is involved. Hence, the proposed algorithm requires $\frac{3}{2}n(n-1)$ complex additions and $(n-1)^2$ complex multiplications. □

Lemma IV.3 Let $n(\geq 4)$ be an integer. The real arithmetic cost in computing the DVM system solving algorithm **DVMS**$(n, \tau, f, \mathbf{y})$ having $\mathbf{y} \in \mathbb{C}^n$ is given by

$$\begin{aligned} \#a\mathbb{R}(DVMS, n) &= (5n-2)(n-1), \\ \#m\mathbb{R}(DVMS, n) &= 4(n-1)^2. \end{aligned} \tag{10}$$

Proof. To set up the vector $\mathbf{v}$, through Step 1 of the algorithm, requires n^2-3n-2 real additions and $2(n^2-3n-2)$ real multiplications. These counts are based on pre-computation and correlated with the design-time computations. Also, the aim of Step 1 is to calculate the algorithm effectively. Steps 2 and 3 follow the similar lines as in Lemma IV.2. Based on the multiplication and subtraction of two complex numbers and taking $|\alpha|=1$, the Steps 2 and 3 involve $3n(n-1)$ real additions and $2n(n-1)$ real multiplications, and $2(n-1)^2$ real additions and $2(n-1)(n-2)$ real multiplications, respectively. In Step 4 of the algorithm, we extract the last column of $\mathfrak{A}$, hence no cost is involved. Therefore, real addition and multiplication complexity of the proposed algorithm are $(5n-2)(n-1)$ and $4(n-1)^2$, respectively. □

By following Lemma IV.2 and IV.3, the proposed DVM system solving algorithm requires $\mathcal{O}(n^2)$ complex and also real arithmetic complexity as opposed to the brute-force calculation of $\mathbf{x}=V(\alpha)^{-1}\cdot\mathbf{y}$ with $\mathcal{O}(n^3)$ arithmetic complexity.

V. Numerical Results

In this section we present execution time and forward error of the proposed DVM system solving algorithm. We compare the execution time of the proposed algorithm with the Gaussian elimination. Also, the forward error of the proposed system solving algorithm will be compared with the MATLAB built-in function *ifft*, for the primitive roots of unity.

A. Performance of the Algorithm

Algorithmic complexity gives us information on the performance of algorithm in time and different platform. An algorithm can show different execution time even with the same inputs based on processor speed, instruction set, disk speed, brand of compiler, etc. Thus, we compute the time complexity of the algorithm using Python to observe its connection to the arithmetic complexity. Also, we have compared the time complexity of the proposed algorithm with the Gaussian elimination method using Python. Figure 3 shows the execution time, in microseconds-μs, of the DVM system solving algorithm and the Gaussian elimination with the size of the matrices varying from 4×4 to 128×128, with size increment 4 and nodes on the unit circle, not necessary the primitive roots of unity. Execution time results were implemented using Python with CPU running at 1.80 GHz and i7-8550U intel core processor.

When the size of matrices increased, it shows from Figure 3 that the proposed algorithm executes much faster than the Gaussian elimination. Although the algorithmic complexity depends on other mentioned factors, we have observed from Lemma IV.3, Lemma IV.2, and Figure 3, that the order of arithmetic complexity coincides with the order of the time complexity.

B. Forward Error of the Algorithm

We show the numerical results for the forward error of the proposed algorithm using MATLAB (R2014a version) with the machine precision $2.23e-16$. The forward error is calculated to address the accuracy of the proposed algorithm. We use the relative forward error formula defined by:

$$e=\frac{\|\mathbf{x}-\hat{\mathbf{x}}\|_2}{\|\mathbf{x}\|_2}$$

where $\hat{\mathbf{x}}$ is the computed solution and $\mathbf{x}$ is the exact solution. Since the DFT matrix is a special case of the DVM, we take the opportunity to compare the forward error of the proposed algorithm with the MATLAB built-in function *ifft* as the exact solution, when nodes of the DVM, i.e., $\{\alpha^k\}_{k=0}^{N-1}$ are primitive roots of unity. The *ifft*$(\mathbf{y})$ computes the product of the inverse of the DFT matrix by the vector $\mathbf{y}$, using the fast Fourier transform algorithm. Also, the *ifft* takes the input vector in the Galois field GF(2^m) with the length 2^m-1 s.t. $m\in\mathbb{Z}$ and $1\leq m\leq 7$. To show the accuracy of the proposed algorithm beyond the length of the input vector in the *ifft*, we execute the propose algorithm for the size of matrices up to 128×128. The third column in Table II shows the forward error in computing the proposed DVM system solving algorithm IV.1 with variable precisions, i.e., with single precision $\hat{\mathbf{x}}$ and double precision $\mathbf{x}$. The fourth column of the table denotes the comparison of the proposed algorithm (taken as a computed solution) with the MATLAB built-in function *ifft* (taken as a true solution). Figures in this column are obtained with the double precision computed solution, i.e. $\hat{\mathbf{x}}=V(\alpha)^{-1}\cdot\mathbf{y}$ and solution obtained using the function *ifft*, i.e. $\mathbf{x}=$ *ifft*$(\mathbf{y})$. In these experiments, we have used distinct nodes $\{1,\alpha,\cdots,\alpha^{n-1}\}$ s.t. $\alpha\in\mathbb{C}$ and $|\alpha|=1$ with random input vector $\mathbf{y}$ to solve the system $V(\alpha)\cdot\mathbf{x}=\mathbf{y}$.

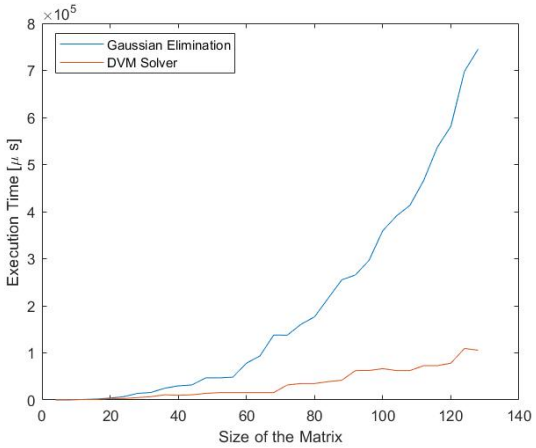

Fig. 3: Execution time of the proposed DVM system solving algorithm vs the Gaussian elimination to solve $V(\alpha)\cdot\mathbf{x}=\mathbf{y}$.

Table II shows that the proposed DVM system solving algorithm yields a very high relative forward accuracy compared with the built-in function *ifft*. Moreover, the proposed algorithm yields constant error order 10^{-8} with respect to variable precisions. Furthermore, the proposed system solving algorithm can be executed for any $\alpha\in\mathbb{C}$ s.t. $|\alpha|=1$ as opposed to the *ifft* function (which can be executed for the primitive roots of unity). From the last column, one can observe the increment of the order of error with the increment

TABLE II: Forward error comparison of the DVM system solving with variable precisions

α	n	Proposed DVMS	IFFT with Proposed DVMS
$e^{-\frac{j\pi}{2}}$	4	3.0188e-08	2.3175e-16
$e^{-\frac{j\pi}{4}}$	4	2.111e-08	-
	6	1.4368e-08	-
	8	1.4028e-08	1.1723e-15
$e^{-\frac{j\pi}{8}}$	4	1.9051e-08	-
	8	2.2695e-08	-
	12	2.4446e-08	-
	16	1.2626e-08	3.6853e-14
$e^{-\frac{j\pi}{16}}$	8	1.5253e-08	-
	16	2.8177e-08	-
	24	2.9418e-08	-
	32	2.0378e-08	7.6324e-10
$e^{-\frac{j\pi}{18}}$	8	2.2843e-08	-
	16	1.5216e-08	-
	32	2.3872e-08	-
	36	2.8917e-08	-
$e^{-\frac{j\pi}{32}}$	8	2.9641e-08	-
	16	1.8514e-08	-
	32	2.2397e-08	-
	64	3.4169e-08	4.5080e-02
$e^{-\frac{j\pi}{64}}$	16	2.3713e-08	-
	32	1.8683e-08	-
	64	2.7164e-08	-
	128	1.9896e-08	-

of the size of the matrices. Also, we recall that the *ifft* works for the length of the input vectors up to $n \leq 127$. In summary, the proposed algorithm shows favorable results when compared to the *ifft*.

VI. Signal Flow Graphs

To illustrate the connection between the algebraic operations used to execute the proposed algorithm IV.1 and to show the simplicity of the algorithm, we provide signal flow graphs (SFGs) having 4-point and 8-point inputs. As shown in the flow graphs, in each graph the signal flows from the left to the right with the dotted lines representing the multiplication by -1. Figures 4 and 5 show the proposed 4-point and 8-point DVM system solving algorithm IV.1, respectively. As shown in the flow graphs, each block is drawn using a combination of adders and multiplier. Thus, the visible counts are different from the real addition and multiplication counts in (10). But these visible counts are same as the complex addition and multiplication counts (9). Note that for the visible addition counts, we have to count the differences in each block.

The SFG is implemented in practice using a digital signal processor based on a parallel systolic array architecture. The design and implementation of the analog and digital hardware and related electronic system will be described in a subsequent contribution. Suffice it to say a Xilinx radio frequency system on chip (RF SoC) platform (ZCU 1285) is the baseline for digital realization. The design, simulation, implementation and test of the electromagnetic interface, microwave front-ends, and digital sub-systems including the hardware description language realization of the proposed algorithm on the RF SoC chip will be covered in a future submission to IEEE Aerospace and Electronic Systems.

VII. Conclusion

In this paper, we have derived a fast algorithm to solve the system of linear equations having the DVM as the coefficient matrix. This algorithm was derived by using the recurrence relations among the Horner polynomials and resulted in $\mathcal{O}(n^2)$ arithmetic complexity as opposed to $\mathcal{O}(n^3)$ arithmetic complexity. The arithmetic complexity and time complexity were coincident with $\mathcal{O}(n^2)$ order of complexity. Numerical results for the forward error bounds indicated very good performance even for large systems of equations with difference delays. Finally, signal flow graphs were presented for the proposed factorization to solve the system of equations associated with the reduced complexity.

The inverse DVM operation is important for array calibration and when the raw antenna outputs are required in the SDR for realization of non-DVM RF beams for various applications. A typical example is the use of adaptive beamforming where optimization algorithms are used for computing the array weights other than DVM weights for cases where optimal statistical beamforming algorithms are necessary. The proposed algorithms are well-suited for real-time implementation using custom digital hardware processors based on systolic array architectures. The current work is focused on the narrowband case to reduce the complexity of the DVM system solver while utilizing the structure of the DVM. The true wideband DVM case requires the algorithm to be extended to frequency dependent coefficients, which can be realized as finite impulse response filters. This is reserved for future work.

Appendix A
The classical Björck-Pererya factorization formula

The authors in [24] have derived a representation for the inverse of the classical Vandermonde matrix, i.e. $V_{n\times n}^{-1}$ in (1), as the product of bidiagonal matrices of the form

$$V^{-1} = U_1 \dots U_{n-1} L_{n-1} \dots L_1, \tag{11}$$

and used the above result to solve the system of linear equations $V \cdot \mathbf{x} = \mathbf{y}$ such that

$$\mathbf{x} = U_1 \dots U_{n-1} L_{n-1} \dots L_1 \mathbf{y},$$

with $\mathcal{O}(n^2)$ arithmetic operations. This is a significant improvement of magnitude compared to the well known Gaussian elimination, which required $\mathcal{O}(n^3)$ operations, in general. The complexity advantage is due to sparse structures of U_k and L_k

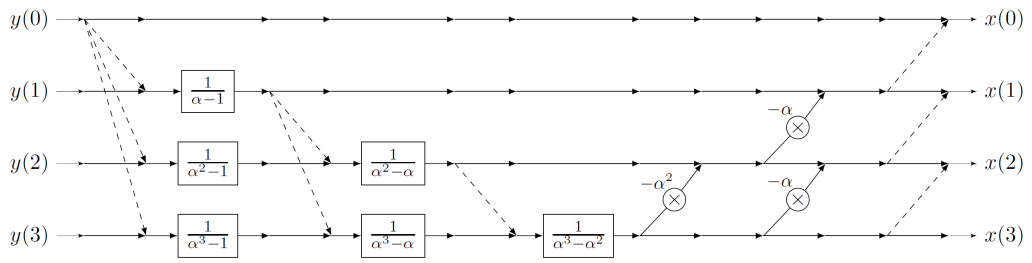

Fig. 4: SFG for 4-point DVM system solving algorithm IV.1.

as shown below:

$$U_k = \begin{bmatrix} I_{k-1} & & & & \\ & 1 & -d_k & & \\ & & 1 & \ddots & \\ & & & \ddots & -d_k \\ & & & & 1 \end{bmatrix},$$

$$L_k = \begin{bmatrix} I_k & & & \\ & \frac{1}{d_{k+1}-d_1} & & \\ & & \ddots & \\ & & & \frac{1}{d_n-d_{n-k}} \end{bmatrix} \tag{12}$$

$$\begin{bmatrix} I_{k-1} & & & & \\ & 1 & & & \\ & -1 & 1 & & \\ & & \ddots & \ddots & \\ & & & -1 & 1 \end{bmatrix}.$$

Appendix B
Horner polynomials and recurrence relations

By following [33], the real-valued Horner polynomials $\widehat{\mathbf{P}} = \{\hat{p}_0(d), \hat{p}_1(d), \cdots, \hat{p}_n(d)\}$ for a given system of real-valued polynomials $\mathbf{P} = \{p_0(d), p_1(d), \cdots, p_n(d)\}$ satisfying $\deg p_k = k$ can be defined via the relation

$$\frac{p_n(d) - p_n(e)}{d - e} = \begin{bmatrix} p_0(d) & p_1(d) & p_2(d) & \cdots & p_{n-1}(d) \end{bmatrix} \cdot \begin{bmatrix} \hat{p}_{n-1}(e) \\ \hat{p}_{n-2}(e) \\ \vdots \\ \hat{p}_1(e) \\ \hat{p}_0(e) \end{bmatrix} \tag{13}$$

where $\hat{p}_n(d) = p_n(d)$. It is well known that classical Horner polynomials associated with monomials $\{1, d, d^2, \cdots, d^n\}$ (nodes of V is based on this set) exist. This shows us such real-valued Horner polynomials exist for a given system of polynomials.

In the following, we will use the system of real-valued Horner polynomials introduced in [33], to define a complex-valued Horner polynomials for a given system of complex-valued functions.

If $\mathbf{W}$ is the system of $n+1$ complex-valued functions $\{1, \omega, \omega^2, \ldots, \omega^{n-1}, w_n(\omega)\}$, where $w_n(\omega) = (\omega - 1)(\omega - \alpha)(\omega - \alpha^2)\cdots(\omega - \alpha^{n-1})$, and α^k's are delays, then the complex-valued Horner polynomials $\widehat{\mathbf{W}} = \{\hat{w}_0(\omega), \hat{w}_1(\omega), \ldots, \hat{w}_{n-1}(\omega), \hat{w}_n(\omega)\}$ can be defined via the relation

$$\frac{w_n(\omega) - w_n(z)}{\omega - z} = \begin{bmatrix} 1 & \omega & \omega^2 & \cdots & \omega^{n-1} \end{bmatrix} \cdot \begin{bmatrix} \hat{w}_{n-1}(z) \\ \hat{w}_{n-2}(z) \\ \vdots \\ \hat{w}_1(z) \\ \hat{w}_0(z) \end{bmatrix}, \tag{14}$$

where

$$\begin{aligned} \hat{w}_n(\omega) &= w_n(\omega) \\ &= (\omega - 1)(\omega - \alpha)(\omega - \alpha^2)\cdots(\omega - \alpha^{n-1}) \\ &= \omega^n + a_{n-1}\omega^{n-1} + \cdots + a_0. \end{aligned} \tag{15}$$

In the following, we will obtain recurrence relations among the complex-valued Horner polynomials in (14).

Proposition B.1 For a given complex-valued function $w_n(\omega)$ (15), the complex-valued Horner polynomials $\{\hat{w}_k(\omega)\}_{k=0}^n$ exist and satisfy the recurrence relations

$$\begin{aligned} \hat{w}_0(\omega) &= 1, \\ \hat{w}_k(\omega) &= \omega \cdot \hat{w}_{k-1}(\omega) + a_{n-k}, \quad k = 1, 2, \cdots, n-1, \end{aligned} \tag{16}$$

and

$$\hat{w}_n(\omega) = \omega \cdot \hat{w}_{n-1}(\omega), \tag{17}$$

where a_k's and α^k's are related via (15).

Proof. Let's consider the divided difference for the complex-valued function $w_n(\omega)$ such that

$$\begin{aligned} w_n[z, \omega] &= \frac{w_n(z) - w_n(\omega)}{z - \omega} \\ &= \sum_{i=1}^{n} a_i \frac{z^i - \omega^i}{z - \omega} \\ &= \sum_{i=1}^{n} a_i \left(z^{i-1} + z^{i-2}\omega + \cdots + z\omega^{i-2} + \omega^{i-1} \right). \end{aligned} \tag{18}$$

Now define complex-valued polynomials $\{\hat{w}_k(\omega)\}_{k=0}^{n-1}$ by

$$w_n[z, \omega] = \hat{w}_n(\omega) + \hat{w}_{n-1}(\omega) \cdot z + \cdots + \hat{w}_0(\omega) \cdot z^{n-1}. \tag{19}$$

Thus by following (14), polynomials $\{\hat{w}_k(\omega)\}_{k=1}^{n-1}$ defined via (19) exist and are called Horner polynomials for the basis $\{\omega^k\}_{k=0}^n$. Furthermore, by equating the coefficients of z^k in

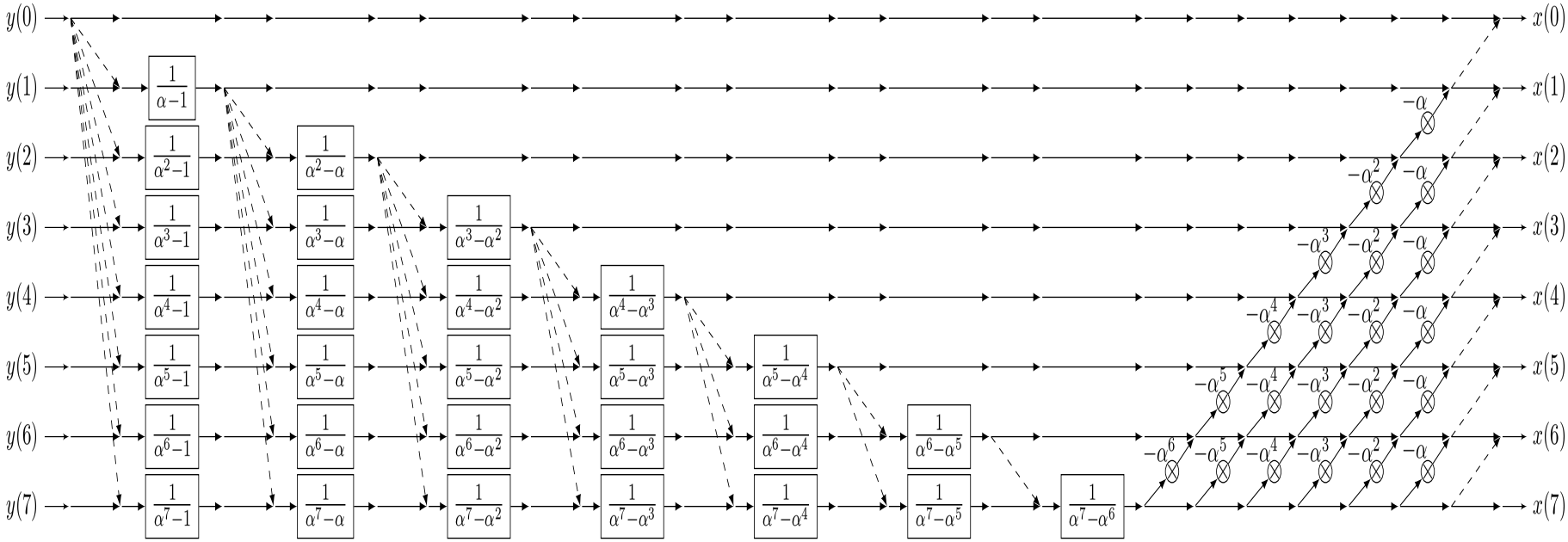

Fig. 5: SFG for 8-point DVM system solving algorithm IV.1.

(18) and (19) for $k = 0, 1, \cdots, n$, it is possible to state the complex-valued Horner polynomials as

$$\hat{w}_k(\omega) = \omega^k + a_{n-1}\omega^{k-1} + \cdots + a_{n-k+1}\omega + a_{n-k} \quad \text{for } k = 0, 1, \cdots, n. \tag{20}$$

Additionally, by writing down each polynomial $\hat{w}_k(\omega)$ in (20) for $k = 0, 1, \cdots, n$ explicitly, it can be seen that the complex-valued Horner polynomials $\{\hat{w}_k(\omega)\}_{k=0}^{n}$ satisfy the recurrence relations (16). □

In Section III, we have used the Proposition B.1 to solve the linear system $V(\alpha) \cdot \mathbf{x} = \mathbf{y}$.

APPENDIX C
AUXILIARY RESULT

The following auxiliary result will be used to obtain the sparse factorization for $V(\alpha)^{-1}$.

Lemma C.1 Let $\alpha \in \mathbb{C}$ and $B = \begin{bmatrix} 1 & -\alpha & & & \\ & 1 & -\alpha & & \\ & & \ddots & \ddots & \\ & & & 1 & -\alpha \\ & & & & 1 \end{bmatrix}$, then

$$B^{-1} = \begin{bmatrix} 1 & \alpha & \alpha^2 & \cdots & \alpha^{n-1} \\ & 1 & \alpha & \cdots & \alpha^{n-2} \\ & & \ddots & \ddots & \vdots \\ & & & 1 & \alpha \\ & & & & 1 \end{bmatrix} \tag{21}$$

Proof. The product of the i^{th} row in B, say B_i, and k^{th} column in B^{-1}, say B_k^{-1}, is given by $0 \cdot \alpha^{k-1} + 0 \cdot \alpha^{k-2} + \cdots + 0 \cdot \alpha^{k-i+1} + 1 \cdot \alpha^{k-i} - \alpha \cdot \alpha^{k-i-1}$. Thus, if $i = k$, $B_i \cdot B_k^{-1} = 1$, else $B_i \cdot B_k^{-1} = 0$. Hence the result. □

REFERENCES

[1] Rappaport, T. S., Xing, Y., Kanhere, O., Ju, S., Madanayake, A., Mandal, S., et al, *Wireless Communications and Applications Above 100 GHz: Opportunities and Challenges for 6G and Beyond*, IEEE Access 7: 78729-78757, 2019.

[2] Rappaport, T. S., Sun, S., Mayzus, R., Zhao, H., Azar, Y., Wang, K., et. al., *Millimeter Wave Mobile Communications for 5G Cellular: It Will Work!,* IEEE Access 1: 335-349, 2013.

[3] Babur, G., Manokhin, G. O., Geltser, A. A., and Shibelgut, A. A., *Low-Cost Digital Beamforming on Receive in Phased Array Radar,* IEEE Transactions on Aerospace and Electronic Systems 53(3): 1355-1364, 2017.

[4] Morsali, A., Norouzi, S., and Champagne, B., *Single RF Chain Hybrid Analog/Digital Beamforming for Mmwave Massive-mimo,* 2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP):1-5, Ottawa, ON, Canada, 2019.

[5] Perera, S. M., Ariyarathna, V., Udayanga, N., Madanayake, A., Wu, G., Belostotski, L., et al, *Wideband N-beam Arrays with Low-Complexity Algorithms and Mixed-Signal Integrated Circuits*, IEEE Journal of Selected Topics in Signal Processing 12(2): 368-382, 2018.

[6] Perera, S. M., Madanayake, A., and Cintra, R., *Efficient and Self-Recursive Delay Vandermonde Algorithm for Multi-beam Antenna Arrays*, IEEE Open Journal of Signal Processing 1(1): 64-76, 2020.

[7] Perera, S. M., Madanayake, A., and Cintra, R., *Radix-2 Self-recursive Algorithms for Vandermonde-type Matrices and True-Time-Delay Multi-Beam Antenna Arrays*, IEEE Access 8: 25498-25508, 2020.

[8] Ariyarathna, V., Udayanga, N., Madanayake, A., Perera, S. M., Belostotski, L. , and Cintra, R. J., *Design methodology of an analog 9-beam squint-free wideband IF multi-beamformer for mmW applications*, In: Proceedings of IEEE 2017 Moratuwa Engineering Research Conference (MERCon): 236-241, IEEE, 2017.

[9] Rotman, W., and Turner, R. F., *Wide Angle Microwave Lens for Line Source Applications,* IEEE Transaction on Antennas and Propagation, 11(6) pp. 623-632, 1963.

[10] Zheng, K., Dhananjay, A., Mezzavilla, M., Madanayake, A., Bharadwaj, S., Ariyarathna, V., et al, *Software-defined Radios to Accelerate mmWave Wireless Innovation*, 2019 IEEE International Symposium on Dynamic Spectrum Access Networks (DySPAN): 1-4, Newark, NJ, USA, 2019.

[11] Pulipati, S., Ariyarathna, V., Dhananjay, A., Eltayeb, M. E., Mezzavilla, M., Jornet, J. M., et al, *Xilinx RF-SoC-based Digital Multi-Beam Array Processors for 28/60 GHz Wireless Testbeds*, 2020 Moratuwa Engineering Research Conference (MERCon): 254-259, Moratuwa, Sri Lanka, 2020.

[12] Zhao, R., Woodford, T., Wei, T., Qian, K., and Zhang, X., *M-Cube: A Millimeter-Wave Massive MIMO Software Radio*, The 26th Annual International Conference on Mobile Computing and Networking (MobiCom '20), September 21-25, 2020, London, United Kingdom. ACM, New York, NY, USA.

[13] Rajapaksha, N., Madanayake, A., and Bruton, L.T., *Systolic array architecture for steerable multibeam VHF wave-digital RF apertures,* IEEE Transactions on Aerospace and Electronic Systems 51 (1): 669-687, January 2015.

[14] Wijayaratna, S., Madanayake, A., Wijenayake, C., and Bruton, L. T., *Digital VLSI architectures for beam-enhanced RF aperture arrays,* IEEE Transactions on Aerospace and Electronic Systems 51(3): 1996-2011, July 2015.

[15] Wijayaratna, S., Madanayake, A., Wijenayake, C., and Bruton, L. T., *Digital VLSI architectures for beam-enhanced RF aperture arrays,* IEEE Transactions on Aerospace and Electronic Systems 51(3): 1996-2011, July 2015.

[16] Coutinho,V. A., Ariyarathna, V., Coelho, D. F. G., Pulipati, S. K., Cintra, R. J., Madanayake, A., et al, *A Low-SWaP 16-Beam 2.4 GHz Digital Phased Array Receiver Using DFT Approximation,* IEEE Transactions on Aerospace and Electronic Systems 56(5): 3645-3654, Oct. 2020.

[17] Madanayake, A., Ariyarathna, V., Madishetty, S., Pulipati, S., Cintra, R. J., Coelho, D., et al, *Towards a Low-SWaP 1024-Beam Digital Array: A 32-Beam Subsystem at 5.8 GHz*, IEEE Transactions on Antennas and Propagation 68 (2): 900-912, Feb. 2020.

[18] Madanayake, A., Cintra, R. J., Akram, N., Ariyarathna, V., Mandal, S., Coutinho, V. A., et al, *Fast Radix-32 Approximate DFTs for 1024-Beam Digital RF Beamforming*, IEEE Access 8: 96613-96627, 2020.

[19] Pulipati, S., Ariyarathna, V., Silva, U. D., Akram, N., Alwan, E. , Madanayake, A., et al, *A Direct-Conversion Digital Beamforming Array Receiver with 800 MHz Channel Bandwidth at 28 GHz using Xilinx RF SoC*, 2019 IEEE International Conference on Microwaves, Antennas, Communications and Electronic Systems (COMCAS): 1-5, Tel-Aviv, Israel, 2019.

[20] Park, S., Alkhateeb, A., and Heath, R. W., *Dynamic subarray architecture for wideband hybrid precoding in millimeter wave massive MIMO systems,* 2016 IEEE Global Conference on Signal and Information Processing (GlobalSIP):600-604, Washington, DC, 2016.

[21] Gao, X., Dai, L., Han, S., Chih-Lin, I., and Heath, R. W., *Energy-Efficient Hybrid Analog and Digital Precoding for MmWave MIMO Systems With Large Antenna Arrays,* IEEE Journal on Selected Areas in Communications 34 (4): 998-1009, 2016.

[22] Walter, G., *Norm Estimates for Inverses of Vandermonde Matrices*, Numerische Mathematik 23(4): 337-347, 1975.

[23] Pan, V. Y., *How Bad Are Vandermonde Matrices?*, SIAM Journal of Matrix Analysis 37(2):676-694, 2016.

[24] Björck, A. and Pereyra, V., *Solution of Vandermonde Systems of Equations*, Mathematics of Computation (American Mathematical Society) 24 (112): 893-903, 1970.

[25] Golub, G. and Van Loan, C., *Matrix Computations third ed.*. Johns Hopkins University Press, Maryland, 1996.

[26] Olshevsky, V., *Pivoting for structured matrices and rational tangential interpolation*, Fast Algorithms for Structured Matrices: Theory and Applications CONM/323:1-75, AMS Publications, 2003.

[27] Higham, N. J., *Error analysis of the Björck-Pereyra algorithms for solving Vandermonde systems*, Numer. Math. 50(5): 613-632, 1987.

[28] Boros, T., Kailath, T., and Olshevsky, V., *Fast Björck-Pereyra-type algorithm for parallel solution of Cauchy linear equations* Linear Algebra Appl., 302-303:265-293, 1999.

[29] Reichel, L. and Opfer, G. *Chebyshev-Vandermonde systems*, Math. Comp. 57:703-721, 1991.

[30] Higham, N. J., *Stability analysis of algorithms for solving confluent Vandermonde-like systems*, SIAM J. Matrix Anal. Appl.11(1):23-41, 1990.

[31] Bella, T., Eidelman Y., Gohberg, I., Koltracht, I., and Olshevsky, V., *A Björck-Pereyra-type algorithm for Szegö-Vandermonde matrices based on properties of unitary Hessenberg matrices*, Linear Algebra and Applications 420(2-3): 634-647, 2007.
[32] Bella, T., Eidelman, Y., Gohberg, I., Koltracht, I., and Olshevsky, V., *A fast Björck-Pereyra-type algorithm for solving Hessenberg-quasiseparable-Vandermonde systems*, SIAM. J. Matrix Anal. and Appl. 31(2): 790-815, 2009.
[33] Kailath, T. and Olshevsky, V., *Displacement structure approach to polynomial Vandermonde and related matrices*, Linear Algebra Appl. 261:49-90, 1997.

Sirani Mututhanthrige Perera received the Ph.D. in Mathematics from the University of Connecticut, USA in 2012. She was the first Sri Lankan selected by the Center for Mathematical Sciences at the University of Cambridge, UK with full Shell Centenary Scholarship (under The Cambridge Commonwealth Trust). She obtained the Part III Tripos in Mathematics (Hons.) from the University of Cambridge, UK in 2006. Perera received the B.Sc. in Mathematics (First Class Hons.) from the University Sri Jayewardenepura, Sri Lanka in 2004. Since 2015, she has been working as an assistant professor in Mathematics at Embry-Riddle Aeronautical University (ERAU), USA. She is working in the field of Numerical Linear Algebra and Scientific Computing. Her research scope includes structured matrices, matrices with displacement structure, FFT, fast and stable algorithms, complexity and performance of algorithms, machine learning algorithms, signal processing, and image processing. Perera is a member of SIAM.

Arjuna Madanayake is an Associate Professor of Electrical and Computer Engineering at Florida International University (FIU) in Miami, Florida. He completed his Ph.D. and M.Sc. both in Electrical Engineering from the University of Calgary, Canada, in 2008 and 2004, and a B.Sc in Engineering (First Class Hons.) from the University of Moratuwa, Sri Lanka, in 2001. His research interests are in one- and multi-dimensional signal processing, antenna arrays, mm-wave systems, microwave engineering, analog circuits, integrated circuit design, FPGA architectures and VLSI realizations of algorithms. His research is supported by the Defense Advanced Research Projects Agency (DARPA), the Office of Naval Research (ONR), and the US National Science Foundation (NSF).

Austin Ogle received his B.S in Engineering Physics Magna Cum Laude in 2018 and an M.S. in Engineering Physics with Distinction in 2019, both from Embry-Riddle Aeronautical University. While attending Embry-Riddle, Austin was selected for the Boren Scholarship to Kyrgyzstan and was recently selected for a Fulbright Grant to Kazakhstan.

Daniel Silverio graduated Summa Cum Laude with a Bachelor of Science in Electrical Engineering from Embry-Riddle Aeronautical University in May 2019. Daniel subsequently completed a Master of Science in Electrical and Computer Engineering with Distinction from Embry-Riddle Aeronautical University in December 2019.

Jacky Qi Huang received the Bachelor of Science degree in Aerospace Engineering and Computational Mathematics from the Embry-Riddle Aeronautical University (Daytona Beach, FL) in 2016. He is currently pursuing Master's degrees in Aerospace Engineering and Systems Engineering at The University of Texas at Arlington (Arlington, TX). He previously worked at Northrop Grumman Corporation.