BEAMFORMING: FUNDAMENTALS TO IMPLEMENTATION

Version 1.0

Luc Langlois, Director | Products & Emerging Technologies /5G | Avnet

CONTENTS

Introduction 3
1. Basic concepts of beamforming 5
1.1 Array steering vector 7
1.2 Applying beam weights in RFSoC programmable logic 9
2. Beamforming in Xilinx Zynq UltraScale+ RFSoC 12
2.1 HDL-optimized minimum-variance distortionless-response (MVDR) beamformer . . . 14
2.1.1Computing the MVDR beam weights in programmable logic 14
3. Further study 16
3.1 Beamforming for 5G communications and MU-MIMO 16
4. Conclusion 17
Appendix A 18

FIGURES

Figure 1 – Beamforming in wireless communication 3
Figure 2 – Beamforming signal chain with Zynq UltraScale+ RFSoC 4
Figure 3 – Relative distances from wavefront to antenna elements 7
Figure 4 – Array steering vector 8
Figure 5 – Digital beamforming applying weights as sum-of-products 9
Figure 6 – Beamforming based on equation (2) (see Appendix A) 10
Figure 7 – Beamforming with Phased Array ToolBox from MathWorks (Appendix A) 11
Figure 8 – Simplified MVDR beamformer in RFSoC (RF front-end not shown) 14
Figure 9 – Implementation in Zynq RFSoC of MVDR beamformer from MathWorks 15
Figure 10 – Channel matrix H for beamforming in wireless communication 16

INTRODUCTION

The advent of the inter-connected world brings innovations ranging from ubiquitous consumer broadband to ultra-low latency wireless vehicular communication and massive machine-type communications (mMTC), making electromagnetic spectrum for radio propagation a more valuable commodity than ever. The efficient use of spectrum in wireless applications is thus of prime concern, prompting advances in phased array antenna systems and beamforming techniques. This paper presents an introduction to fundamental beamforming theory for phased array systems and its implementation in Xilinx Zynq® UltraScale+™ RFSoC.

The essence of beamforming is described in Wikipedia :

> *Beamforming or spatial filtering is a signal processing technique used in sensor arrays for directional signal transmission or reception.[1] This is achieved by combining elements in an antenna array in such a way that signals at particular angles experience constructive interference while others experience destructive interference. Beamforming can be used at both the transmitting and receiving ends in order to achieve spatial selectivity. The improvement compared with omnidirectional reception/ transmission is known as the directivity of the array.*

Figure 1 - Beamforming in wireless communication

Xilinx Zynq UltraScale+ RFSoC provides essential functionality for beamforming. Integrated multi-channel direct-RF data converters enable agile frequency planning for fully digital beamforming without external RF mixing circuitry at carrier frequencies up to 6 GHz. For millimeter wave (mmWave) applications, the integrated numerically controlled oscillator (NCO) and complex mixer within each data converter block can accurately place an intermediate frequency (I/F) entirely in the digital domain for conversion to/from the analog domain toward an external RF mixing stage.

Matching converter latency across data converter channels is a critical prerequisite for beamforming. The multi-tile synchronization (MTS) feature in Zynq UltraScale+ RFSoC can be used to achieve relative and deterministic multi-tile and multi-device alignment.[ii]Data converter tiles are fully programmable at runtime through the RFdc driver API functions for both bare metal and Linux running on the Arm® Cortex®-A53 64-bit dual-core processor and Cortex-R5F dual-core real-time processor.

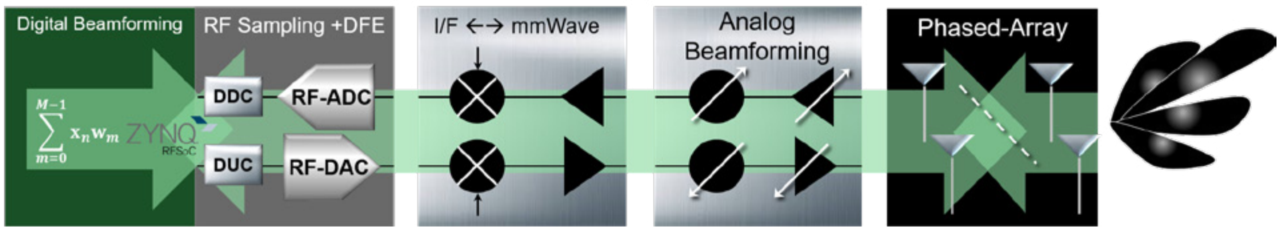


Figure 2 - Beamforming signal chain with Zynq UltraScale+ RFSoC

Whether for RF propagation through antenna arrays or for audio applications with multiple microphones, beamforming may seek various objectives. The beams can be made to have high gain and low sidelobes, or controlled beamwidth (directivity) or even generate simultaneous independent beams for spatial multiplexing.[iii]Adaptive beamforming techniques dynamically adjust the array pattern to optimize some characteristic of the received signal while rejecting interference and noise.

We shall limit our scope to consider two main components of beamforming:

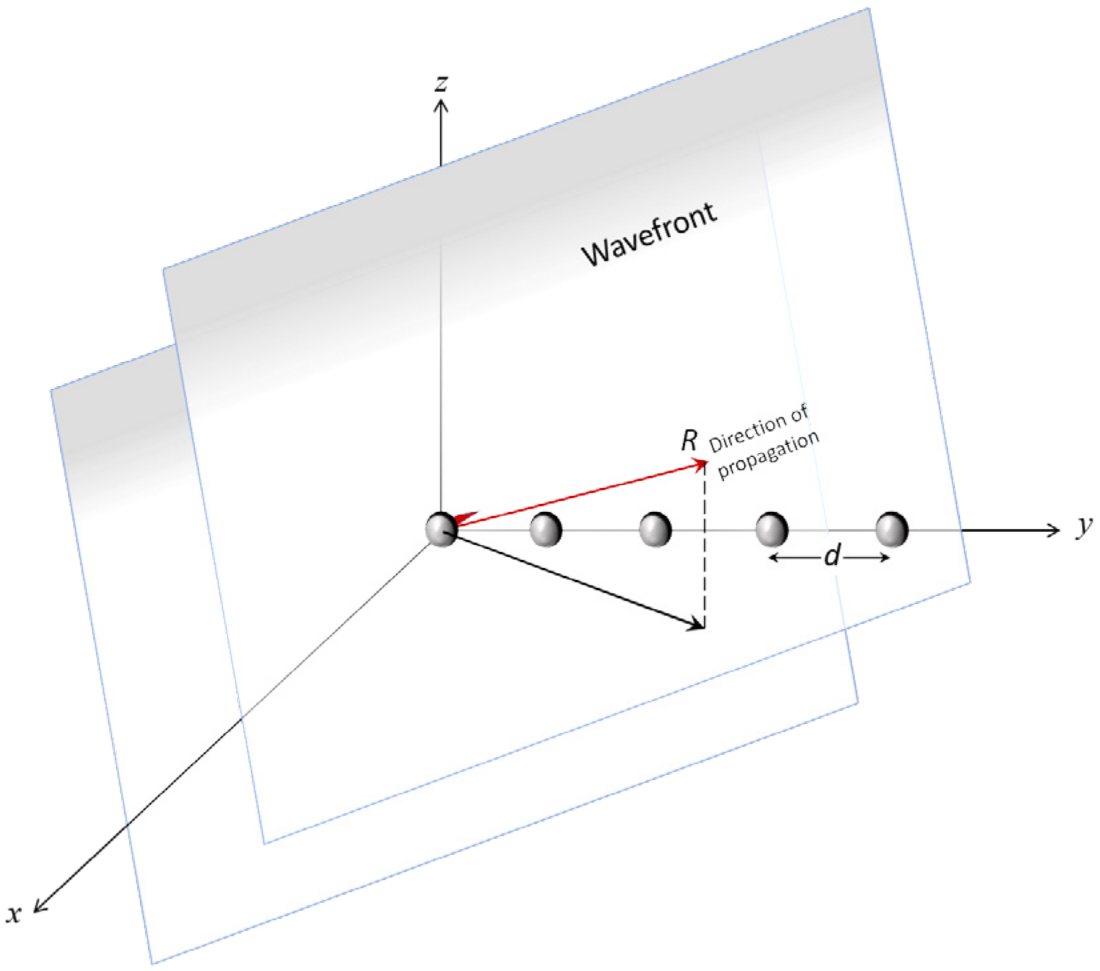
- Computing the beam weights, primarily an optimization problem toward realizing one or several of the above objectives
- Applying the beam weights, whether in the digital or analog domain or both, known respectively as digital / analog / hybrid beamforming

The latter is the topic in Section1. The former, more complex is the focus in Section 2.

1. BASIC CONCEPTS OF BEAMFORMING

To help grasp the methods of beamforming, we first define some basic concepts in 3D space. In this text we adhere to conventions of spherical coordinates as defined by Phased Array System Toolbox™ from MathWorks[iv].

Consider a uniform linear array of antenna elements (ULA) spaced *d* meters apart along the y-axis of an orthogonal coordinate system. The ULA is irradiated by a plane wave emitted from a distant single point-source in the far field situated behind and beneath the ULA, such that all points on a given wavefront depicted as emerging from the page have equal instantaneous power. The direction of propagation is represented by vector *R*, normal to the wavefront.

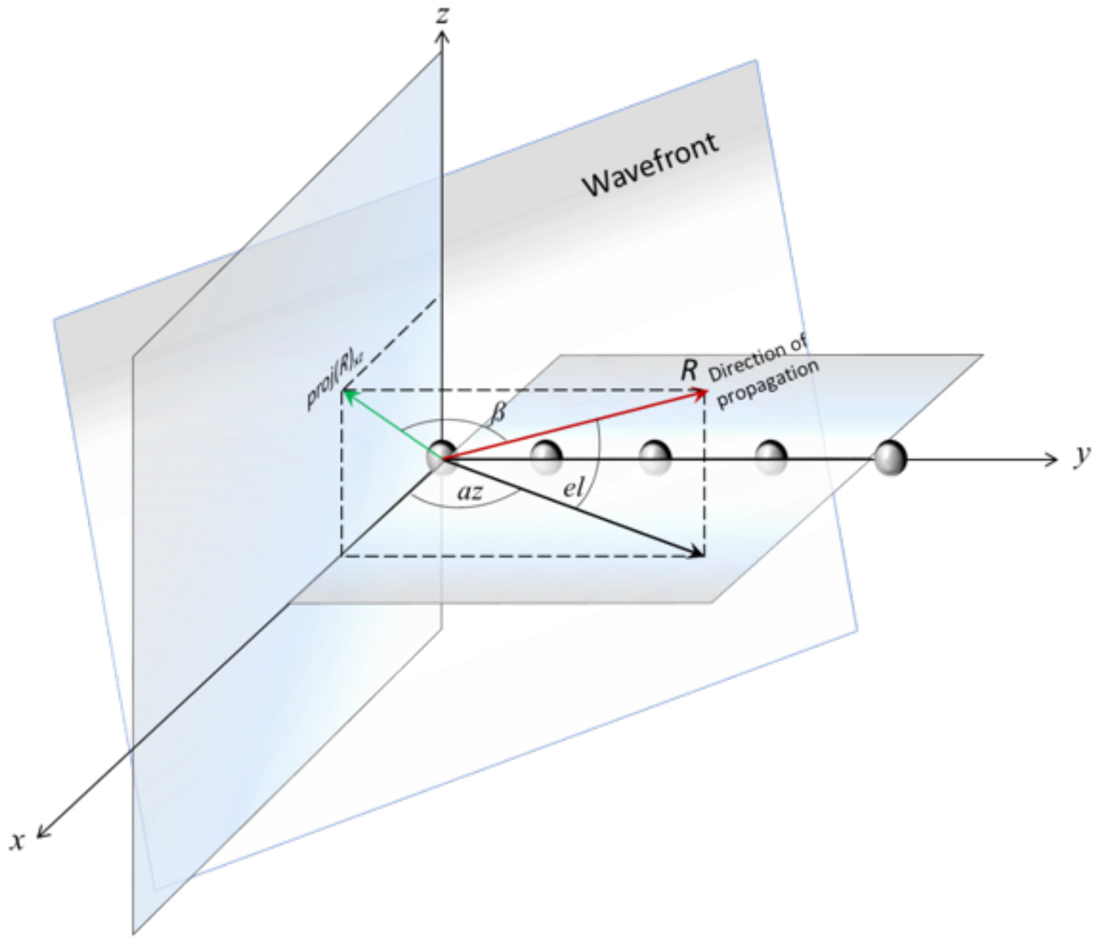


We choose the element at the origin of the coordinate system [0 0 0] as the point of reference known as the phase center, from which the phase offsets of all other elements in the array will be measured. By convention, two angles serve to define the direction of propagation of the wavefront:

- Azimuth angle (*az*) is taken from the x-axis to the projection of the propagation vector R onto the xy plane
- Elevation angle (*el*) is taken from the projection of R onto the xy plane to *R*

Broadside angles are useful when describing the response of a ULA. The array response depends directly on the broadside angle *β* and not on the azimuth and elevation angles. Follow these steps to visualize the broadside angle *β*:

- Project the vector *R* onto the plane normal to the ULA, i.e. the xz axis. Call this vector proj(*R*)xz
- Form an imaginary plane spanning proj(*R*)xz and the direction of propagation vector *R*
- The broadside angle (*β*) is taken from proj(*R*)xz to the vector *R* while traversing the aforementioned imaginary plane.



1.1 Array steering vector

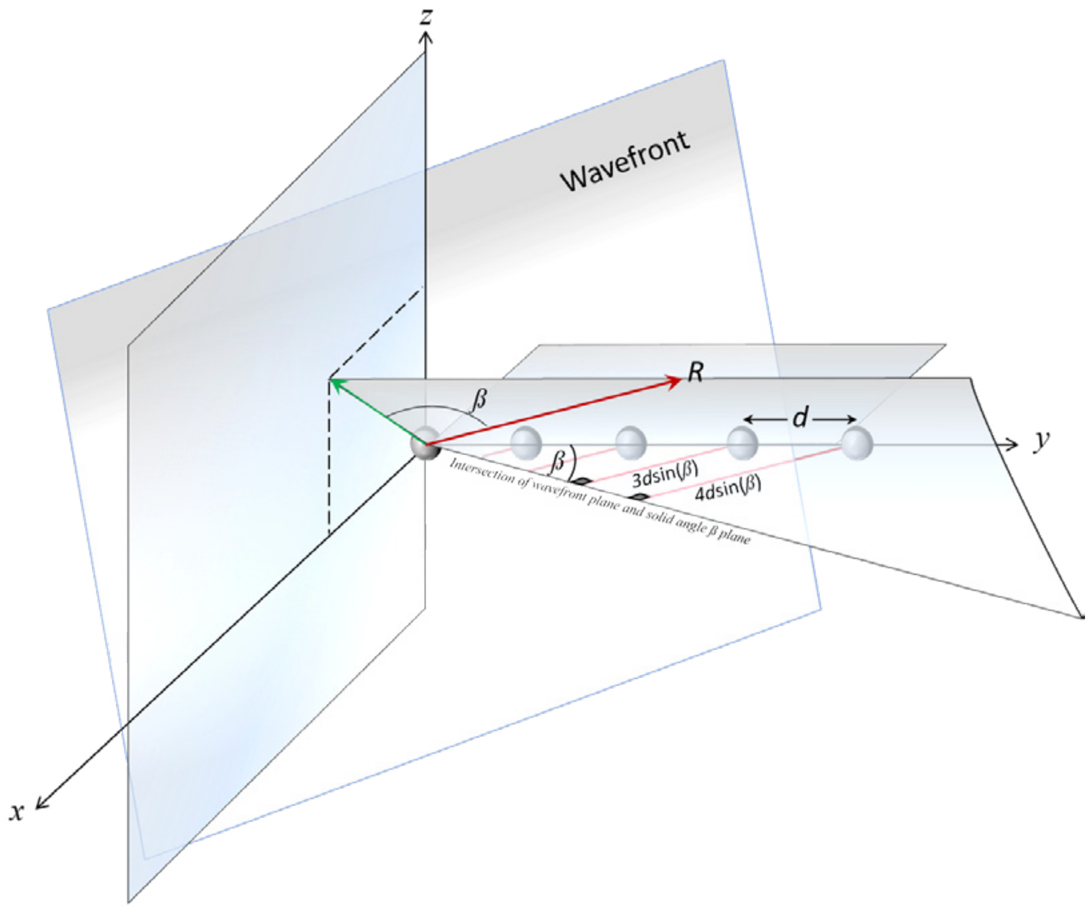


Figure 3 - Relative distances from wavefront to antenna elements

Consider a plane wave propagating in free space with carrier $e^{j2\pi f_0 t}$ modulated by a signal of interest s(*t*). [1]The distance travelled by a given wavefront to successive array elements depends on the angle of arrival (AoA), broadside angle *β*, creating a time delay $\tau = d\sin(\beta)/c$, where c is the speed of the wave, nominally the speed of light. The vector of signals ***s(t)*** induced into the antenna feed network from the array elements is given by equation (1).

$$\mathbf{s}(t) = \alpha s(t - m\tau)e^{j2\pi f_0 (t - m\tau)} \quad (1)$$

(1) If the carrier frequency $f_0 < 6$ GHz the signal can be mixed to baseband and decimated through the digital down-converter within the direct-RF ADC tile of the RFSoC device. At millimeter wave carrier frequencies, the down-conversion can be implemented through the Otava DTRX2 Dual Transceiver mmWave Radio Card

Assuming a narrowband signal of interest with bandwidth *<< c/(M-1)d Hz*[v], then *$s(t-m\tau)\approx s(t)$*, and neglecting any interference and noise, the signal vector **s**(*t*) from the M elements of the array (after down-conversion) can be expressed as shown below, where **a**(*β*) is known as the array steering vector and λ the wavelength of the carrier in free space.

$$\mathbf{s}(t) = s(t)\underbrace{\begin{bmatrix} 1 \\ e^{-j\ 2\pi d\sin(\beta)/\lambda} \\ e^{-j\ 4\pi d\sin(\beta)/\lambda} \\ e^{-j\ 6\pi d\sin(\beta)/\lambda} \\ \vdots \\ e^{-j\ (M-1)2\pi d\sin(\beta)/\lambda} \end{bmatrix}}$$

$$\text{Array steering vector} \longrightarrow \boldsymbol{a}(\beta)$$

Figure 4 - Array steering vector

The goal of beamforming is to find a set of complex weights **w** to optimize some parameter such as signal-to-noise ratio. In the absence of noise and interference, the trivial solution is the set of weights that will null the phase of each complex exponential term in the steering vector **a**(*β*), resulting in signals that are perfectly in-phase with each other *($e^{j0}=1$)* across the array, thereby maximizing the signal of interest. This can be accomplished by multiplying each component signal of **s**(t) by a complex weight w which is the conjugate of each term of **a**(*β*).[2] Summing these in-phase signals will result in constructive interference that will amplify the signal at the angle of arrival *β*. Expressed mathematically this is the product of the input signal s(t) impinging on the array and the inner product of the steering vector **a**(*β*) with its conjugate transpose, or Hermitian as per equation (2).

$$\mathrm{y(t)} = \mathrm{s}(t)\cdot \mathbf{w}^H \boldsymbol{a}(\beta), \text{with } \mathbf{w}^H = \boldsymbol{a}^H \qquad (2)$$

Note that $\mathbf{w}^H$ is a row vector; **a**(*β*) a column vector. Their inner product $<\mathbf{w}^H,\mathbf{a}(\beta)>$ is a scalar gain factor that is applied to the signal s(t) at the angle of arrival *β*. Setting **w** equal to **a** provides gain factor of M, the number of antennas in the array.

(2) This is known as a phase-scanned linear array, optimal for narrowband signals with bandwidth significantly less than the carrier frequency. By contrast a time-scanned linear array works by applying time delays to each component of **s**(t).

1.2 Applying beam weights in RFSoC programmable logic

Similar to the topology of a FIR filter where the tapped delay line is replaced by the antenna array, beamforming is often referred to as *spatial filtering*. The inner product of 2 vectors amounts to a sum-of-products, which can be performed in fully parallel multiply-and-add configuration as shown in Figure 5, or sequentially using complex multiply-accumulators (CMACC) – not shown. Multiple implementations of CMACC are supported using DSP48E2 hardened blocks in RFSoC, with flexible trade-offs between performance and resource utilization.[vi] The hardened DSP48E2 slices in RFSoC are capable of aggregate computation performance up to 7,613 GMACC/s.[vii]

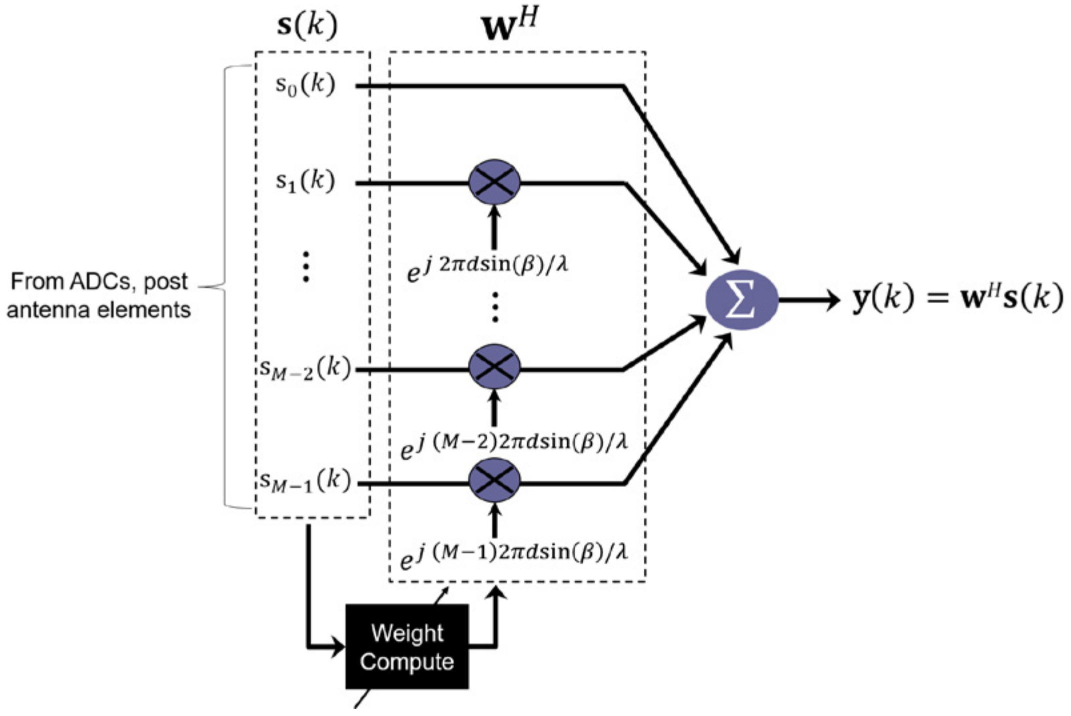

$\mathbf{s}(k)$
$\mathbf{w}^H$
$s_0(k)$
$s_1(k)$
$e^{j\,2\pi d\sin(\beta)/\lambda}$
From ADCs, post antenna elements
$s_{M-2}(k)$
$e^{j\,(M-2)2\pi d\sin(\beta)/\lambda}$
$s_{M-1}(k)$
$e^{j\,(M-1)2\pi d\sin(\beta)/\lambda}$
Σ
$\mathbf{y}(k) = \mathbf{w}^H\mathbf{s}(k)$
Weight Compute

Figure 5 - Digital beamforming applying weights as sum of products

In a digital beamformer receiver architecture, each component of the signal vector **s**(t) arriving through the RF feed network from the M antenna elements of the array is digitized by a dedicated data converter, hence the index 'k' denoting discrete-time sampled signals, used hitherto in place of continuous time signals without loss of generality.

Xilinx Zynq UltraScale+ RFSoC devices include up to 16 channels of integrated RF-ADCs and RF-DACs with up to 6GHz of direct RF bandwidth for full sub-6 GHz digital beamforming[viii]. Each digitized component signal **s**(*k*) can be processed through a dedicated digital down-converter (DDC) directly within the RF-ADC tile, comprising complex mixer and decimation filters, all fully programmable[ix]. Upon emerging from the DDC into the programmable logic at a decimated sampling rate sufficient to respect Nyquist and mixed down to baseband, the signal vector **s**(*t*) is routed toward a beamformer weight computation engine implemented in programmable logic.

For carrier frequencies above 6 GHz, an external RF mixing stage may be added for down-conversion to an intermediate frequency (I/F) < 6 GHz.

Beamforming demonstrations with example MATLAB code are included in appendix A, starting from first-principles based on equation (2) and progressing towards algorithms from Phased Array System Toolbox from MathWorks for designing and simulating sensor array and beamforming systems. These introductory code examples illustrate fundamental concepts of beamforming by applying complex weights to a ULA with ideal CW tone signals free of interference and noise.

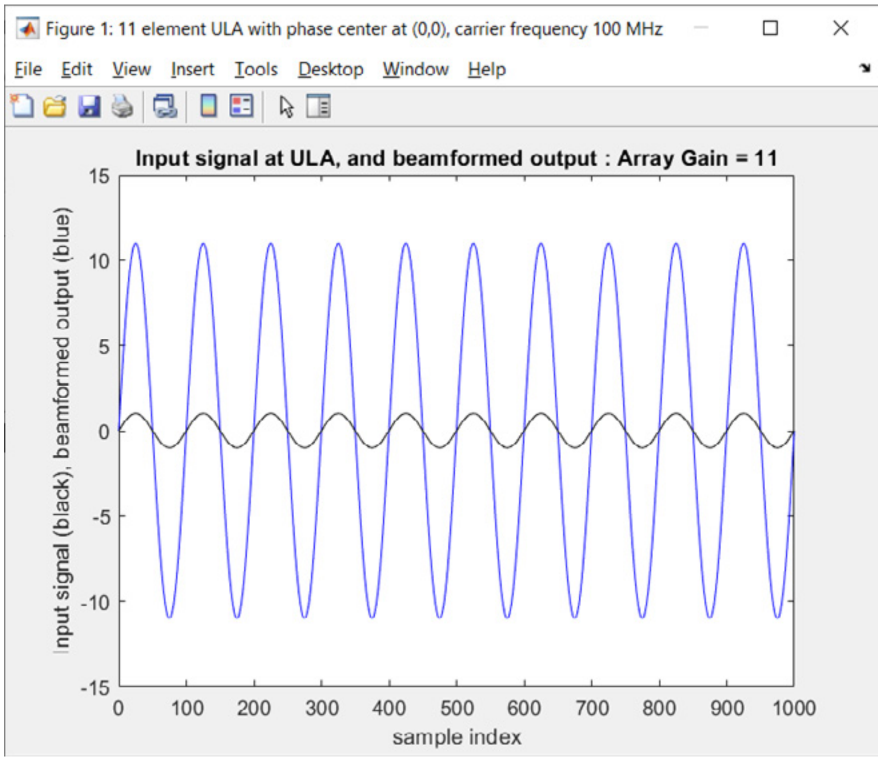
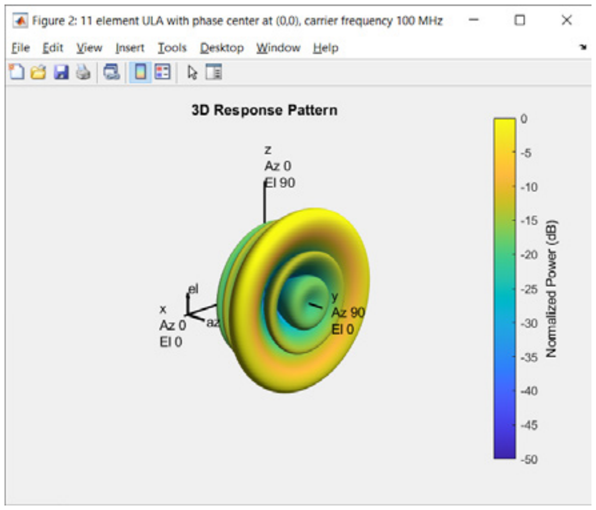


Figure 6 - Beamforming based on equation (2) (see Appendix A)

Phased Array System Toolbox from MathWorks provides algorithms and apps for designing and simulating sensor array and beamforming systems in wireless communication, radar, sonar, acoustic and medical imaging applications. From the code examples of Appendix A, Figure 7 shows the response of an 11-element ULA in 3D space with azimuth 30° and free-range elevation angle and an azimuth-cut with fixed elevation angle of 0°.



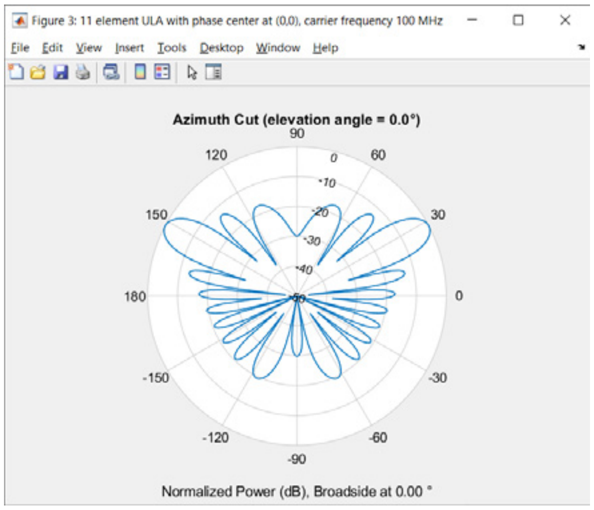


Figure 7 - Beamforming with Phased Array ToolBox from MathWorks (Appendix A)

2. BEAMFORMING IN XILINX ZYNQ ULTRASCALE+ RFSOC

In the preceding section we neglected any noise or interference impinging upon the ULA along with the signal of interest. In the real world such impairments will of course be present such that the signal from the ULA can be represented as

$$\mathbf{x}(k) = \mathbf{s}(k) + \mathbf{i}(k) + \mathbf{n}(k) \quad (3)$$

where $\mathbf{x}(k) \in \mathbb{C}^M$ is the discrete-time array observation vector at sample index k. $\mathbf{s}(k)$, $\mathbf{i}(k)$ and $\mathbf{n}(k)$ denote the Mx1 column vectors of the signal of interest (SOI), interference and noise respectively, which are assumed uncorrelated with the SOI, and $\mathbf{s}(k)=s(k)\mathbf{a}(\beta_s)$, where β_s is the desired (broadside) source angle-of-arrival (AoA).[3] M is the number of elements in the antenna array.

An adaptive beamforming technique known as minimum variance distortionless response (MVDR) can constructively combine the RF signal received through a phased-array antenna to increase the SOI power while simultaneously creating nulls to mitigate interference and noise.[x] Applications include long-range surveillance radar, active jammer rejection and multibeam antennas for space communications.[xi] MVDR beamforming also finds uses in speech enhancement and audio noise reduction from microphone arrays.

The task of retrieving the signal of interest through MVDR beamforming is an optimization problem in which we seek to maximize an objective function, such as the signal-to-interference-and-noise ratio (SINR). SINR is a ratio of average power: in the numerator the SOI power; in the denominator the combined interference and noise power. Calculating power involves taking the square of a signal and averaging over some time period, hence the index `k' as the discrete time sample index, i.e. after sampling through the ADC.

3 Note the distinction between vectors in boldface and scalars in standard typeface. The scalar $s(k)$ is the electromagnetic signal arriving over the air in subsequent wavefronts at the antenna array, sampled at time `k'. $\mathbf{s}(k)$ is the Mx1 column vector of outputs from the M antenna elements, after sampling through ADCs. Observe that SINR in equation (4) boils down to a ratio of scalar values.

We seek the Mx1 complex weight vector $\mathbf{w} \in \mathbb{C}^M$ that will maximize the beamformer output SINR:

$$\text{SINR} \triangleq \frac{\boldsymbol{E}[|\mathbf{w}^H \boldsymbol{s}|^2]}{\boldsymbol{E}[|\mathbf{w}^H(\boldsymbol{i}+\boldsymbol{n})|^2]} = \frac{\boldsymbol{\sigma}^2|\mathbf{w}^H\boldsymbol{a}|^2}{\mathbf{w}^H\boldsymbol{R}_{i+n}\mathbf{w}} \quad (4)$$

where σ^2 is the (scalar) average power of the signal of interest, $\boldsymbol{R}_{i+n}$ is the interference-plus-noise covariance matrix, and $\boldsymbol{E}[\cdot]$ denotes the expectation operator, which is approximated by time averaging.[xii]

Maximizing the SINR in the MVDR beamformer involves minimizing the denominator of equation (4) while keeping the numerator fixed. The solution is given (without proof) as:

$$\mathbf{w}_{\text{MVDR}} = \alpha \boldsymbol{R}_{\boldsymbol{i+n}}^{-\mathbf{1}} \boldsymbol{a}(\beta_s) \ (5)$$

Where $\alpha = 1/\boldsymbol{a}^H(\beta_s)\boldsymbol{R}_{i+n}^{-1}\boldsymbol{a}(\beta_s)$ is a normalization constant that does not affect the output SINR (4) and, therefore will be omitted until applied in the final step.[xiii]

In the real world the individual signals comprising vector **x** don't emerge from the antenna array neatly separated from the interference and noise; rather the 3 components are contained in **x** through superposition as per eq (3). In practice the interference-plus-noise covariance matrix is unknown and can be estimated by the data sample covariance matrix $\widehat{\boldsymbol{R}}$, which is calculated as a moving average of ***K*** snapshots of the array:

$$\widehat{\boldsymbol{R}} \triangleq \frac{1}{K}\sum_{k=0}^{K-1} \mathbf{x}(k)\mathbf{x}^H(k) \quad (6)$$

2.1 HDL-Optimized Minimum-Variance Distortionless Response (MVDR Beamformer

A simplified block diagram of an MVDR (digital) beamformer is shown below, implemented in Xilinx Zynq UltraScale+ RFSoC:

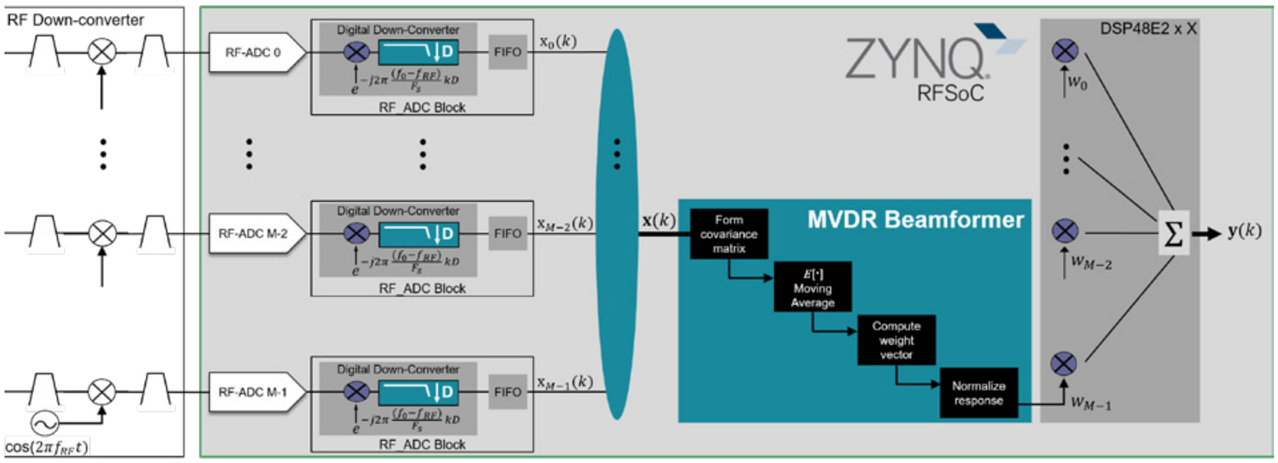


Figure 8 - Simplified MVDR beamformer in RFSoC (RF front-end not shown)

2.1.1 Computing the MVDR beam weights in programmable logic

Derivation of the beam weights in programmable logic calls for efficient computation techniques optimized for fixed-point math. While the solution to the MVDR beam weights in equation (5) is mathematically correct, it is not practical to perform direct matrix inversion in hardware. A more suitable approach is QR decomposition of the covariance matrix followed by back substitution of the upper triangular matrix **R** to solve for the vector of beam weights $\mathbf{w}_{\mathrm{MVDR}}$.

The QR factorization of a K-by-M matrix A produces an K-by-M upper triangular matrix **R** with all zeros below the diagonal and an K-by-K orthogonal matrix **Q** such that **A** = **QR**. A complex M-by-M matrix **Q** is orthogonal if $\mathbf{Q}^H\mathbf{Q} = \mathbf{I}$, the identity matrix.
Re-writing equation (5) with QR decomposition applied to data covariance matrix $\widehat{\boldsymbol{R}}$:

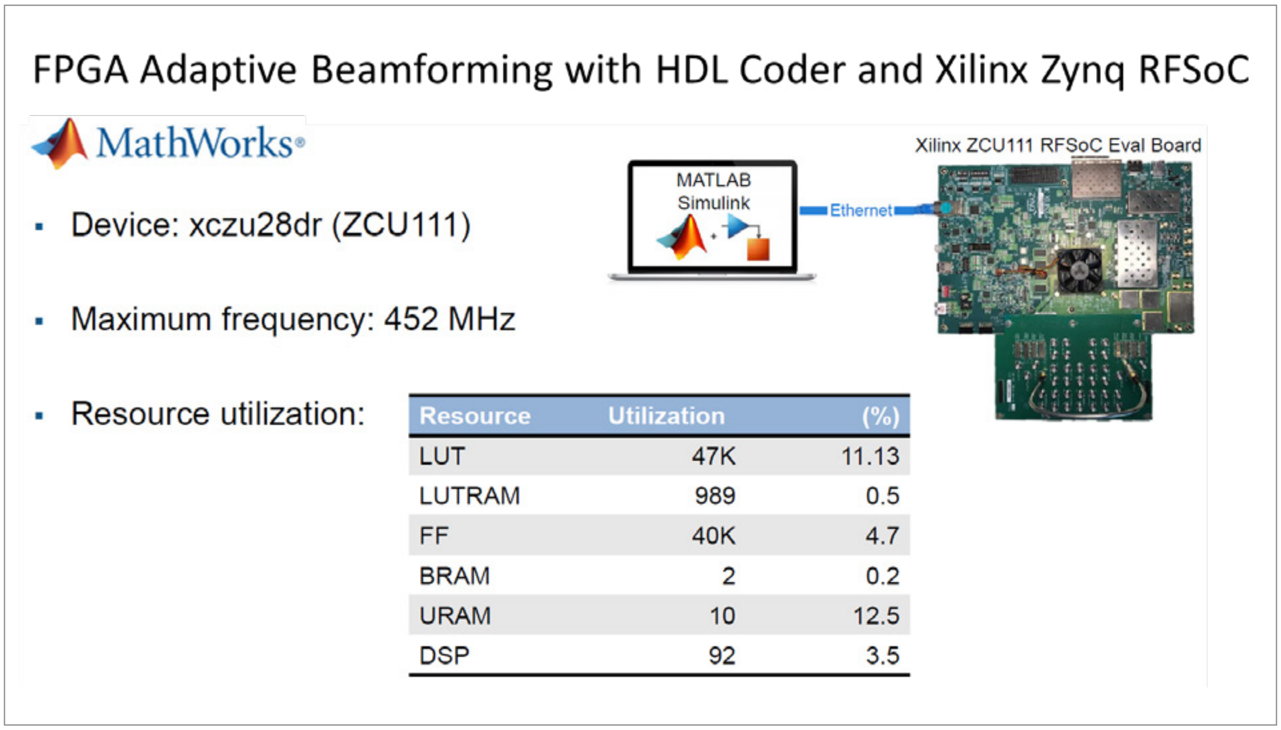
$$\widehat{\mathbf{R}}\mathbf{w}_{\mathrm{MVDR}} = \boldsymbol{a}(\beta_s) \quad (7)$$

$$\mathbf{QR}\mathbf{w}_{\mathrm{MVDR}} = \boldsymbol{a}(\beta_s)$$

$$\mathbf{R}\mathbf{w}_{\mathrm{MVDR}} = \mathbf{Q}^H\boldsymbol{a}(\beta_s) \quad (8)$$

With R upper triangular, we can employ simple back-substitution to solve for the weights $\mathbf{w}_{\mathrm{MVDR}}$ [xiv]

The CORDIC algorithm is used extensively to implement QR decomposition in fixed-point arithmetic because it can apply orthogonal Givens rotations using only shift and add operations.[xv] [xvi] MathWorks offers several beamforming and radar designs with Simulink models and MATLAB reference code to showcase high-level simulation and HDL designs of array processing algorithms. Source code is available for a functional demonstration of MVDR beamforming using DAC to ADC cabled loopback targeting Xilinx Zynq UltraScale+ RFSoC ZCU111 Evaluation Kit.[xvii] Various implementations of complex CORDIC arrays including systolic, partial-systolic and burst forms are presented with trade-offs of DSP48 resource utilization, maximum system clock frequency, latency and throughput.



FPGA Adaptive Beamforming with HDL Coder and Xilinx Zynq RFSoC

- Device: xczu28dr (ZCU111)
- Maximum frequency: 452 MHz
- Resource utilization:

Resource	Utilization	(%)
LUT	47K	11.13
LUTRAM	989	0.5
FF	40K	4.7
BRAM	2	0.2
URAM	10	12.5
DSP	92	3.5

Figure 9 – Implementation in Zynq RFSoC of MVDR beamformer from MathWorks

3. FURTHER STUDY

3.1 Beamforming for 5G communications and MU-MIMO

The MVDR beamformer algorithm featured in the previous section relies solely on the covariance matrix at the receiver to derive the beamforming weights that maximize SINR. Wireless communication standards such as 5G take a different approach by exploiting the spatial diversity of multi-path signal trajectories to implement multiplexing of several independent data streams called "layers" onto the MIMO channel. The objective is increased capacity in terms of bits/s/Hz. Such techniques are especially crucial for transmission of millimeter waves through large uniform rectangular arrays (URA) that steer beams in both azimuth and elevation to overcome the propagation loss at these shorter wavelengths.

With dynamic channel conditions in busy urban environments where multiple, mobile users are communicating with a 5G base station (gNB), the standard calls for channel-state information (CSI) to be sent from the receiver to the transmitter[xviii]. A channel matrix denoted by **H** holds the complex phase and gain relationship between each transmit and receive pair of antenna elements and forms the basis for adaptive beam weight computation. The rank of the channel matrix **H** reflects the multi-path environment between the transmitter and receiver arrays, and is a first-order but crude measure of the capacity of the channel.

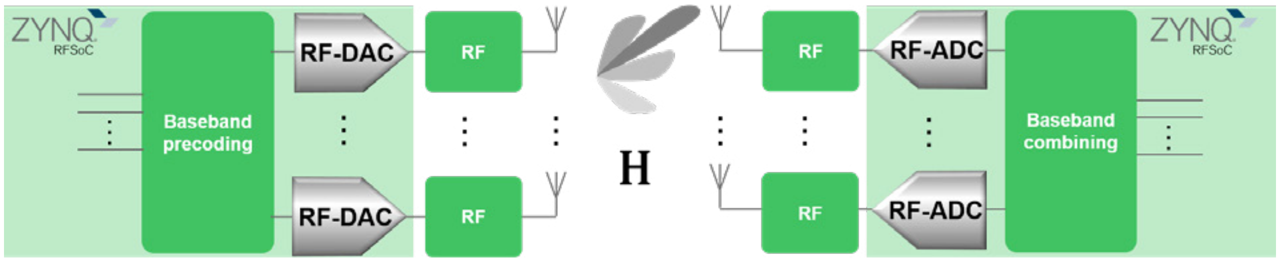


*Figure 10 - Channel matrix **H** for beamforming in wireless communication*

Beamforming algorithms for multi-user multiple-input-multiple-output (MU-MIMO) in 5G NR is a rich subject beyond the scope of this work. Comprehensive references are available with details of the underlying mathematical framework[xix]. Beamforming in the context of MU-MIMO technology calls for the intensive compute capabilities of Xilinx AI Engine within Versal™ AI Core devices. An efficient implementation for applying beamforming weights in multiple 5G layers on AI Engine arrays is described with architectures consisting of three large matrix-multiplication kernels with slight differences, scalable to various matrix sizes and throughput requirements.[xx]

4. CONCLUSION

This paper introduced some fundamental concepts of beamforming in phased array systems. Section 1 established the conventions of wavefronts impinging upon a uniform linear array in a 3D coordinate system and the concept of steering vector with an ideal signal of interest unimpaired by noise or interference. In Section 2 we focused on the popular Minimum-Variance-Distortionless-Response (MVDR) beamforming technique with a practical MATLAB / HDL coder-based implementation from MathWorks targeting RFSoC using efficient QR decomposition techniques in fixed-point arithmetic.

Adaptive beamforming is a fast-evolving and essential technology in a wide range of wireless applications. The literature is vast and demands meticulous study in order to realize efficient implementations in advanced systems. Xilinx Zynq UltraScale+ RFSoC and Versal AI offer unmatched levels of integration and performance for a range of beamforming applications. The references cited in the bibliography provide a sound basis for further study.

APPENDIX A

```
% Demonstrate basic concepts of ideal beamformer with ULA
%
%  1)  Define ULA from first principles
%  1a) perform beamforming of narrowband signal `s'
%  1b) plot beamformed output signal in the time-domain and verify beamforming gain
%
%  2) Repeat with Phased Array ToolBox from MathWorks
%
%  Assumptions:
%    signal `s' is free of any noise and interference impinging on the ULA
%    ... see MVDR beamformer for beamforming techniques to deal with these effects
%    antenna elements have ideal isotropic response
%    propagation speed of the wavefront impinging on the ULA = c, speed of light

% Ref; MATLAB Help `Spherical Coordinates'

%%
% Luc Langlois, Avnet
% May 2021
%
%% define parameters of Uniform Linear Array (ULA)
M = 11;         % number of antenna elements
fc = 100e6;     % carrier frequency (Hz)
lambda = physconst(`LightSpeed')/fc; % carrier wavelength (lambda)
delta = 0.5;    % dimensionless fraction of the carrier wavelength (lambda)
Fs = 100*fc;    % sampling frequency (Hz)
az = 30;        % azimuth and elevation angles of arrival (in degrees) of impinging signal of interest, as defined in Phased
Array Toolbox Help / `Spherical coordinates'
el = 20;

% create Uniform Linear Array from first principles
[steering_vector, ~, Beta] = ULA_basic(1, fc, az, el, M, delta, (M-1)/2);

s = sin(2*pi*fc/Fs*[0:1000])';  % narrow band signal modulating carrier of frequency fc, sampled at I/F post any RF down-
conversion

% create array outputs (time is along the rows dimension, array element number along the columns dimension)
s_vector = zeros(length(s),M);
for m = 1:M
   s_vector(:,m) = s*steering_vector(m);
end

% beamforming
y = s_vector*steering_vector'; % inner product <array output vector, Hermitian of steering vector>
```

```
gain = (max(abs(y))/max(abs(s)));

% plot beamformed output signal and verify beamforming gain
figure(`Name', strcat(string(M), " element ULA with phase center at (0,0), carrier frequency ", string(fc*1e-6), " MHz"));
plot([0:length(y)-1],real(y), `Color', `blue');
hold on;
plot([0:length(s)-1],real(s), `Color', `black');
title(strcat("Input signal at ULA, and beamformed output : Array Gain = ", string(gain)));
xlabel(`sample index')
ylabel(`Input signal (black), beamformed output (blue)');

if gain == M
   disp(strcat("Beamforming gain matches expected value = ", string(gain)," (number of antenna elements)"));
else
   disp(strcat("Beamforming gain does NOT match expected value ", string(gain)," (number of antenna elements)"));
end

%% Create Uniform Linear Array (ULA) from MathWorks Phased Array ToolBox
array = phased.ULA(M, `ElementSpacing', lambda*delta);

s_vector = collectPlaneWave(array,s,[az;el],fc,physconst(`LightSpeed')); % vector `s' output from sensor array, when the
received plane wave signal(s) indicated by `s' arrive(s) at the array from the direction(s) specified in AoA

a_vector = phased.SteeringVector(`SensorArray',array);

y = s_vector*(a_vector(fc, [az; el])').'; % beamform using Hermitian of steering vector

% Generate 1-D beamformer response by sweeping the weights vector across azimuth, at fixed elevation angle
beamformer_response = zeros(1,180);
for az_ang = 1:180
   w_vector = (a_vector(fc, [az_ang; el]));
   beamformer_response(az_ang) = w_vector'*a_vector(fc, [az; el]);
end

% plot beamformer response
figure(`Name', strcat(string(M), " element ULA with phase center at (0,0), carrier frequency ", string(fc*1e-6), " MHz"));
plot([1:length(beamformer_response)],abs(beamformer_response), `-o', `Color', `black');
title(strcat("Beamformer response with steering vector [azimuth, elevation] = ", string(az), ",", string(el), " degrees"));
xlabel(`azimuth angle')
ylabel(`Beamformer response (magnitude)');
%
pattern(array,fc, [1:180], `weights',  a_vector(fc, [30; 0]),  `CoordinateSystem','polar', `Type','powerdb','PlotStyle','overlay')
%

figure(`Name', strcat(string(M), " element ULA with phase center at (0,0), carrier frequency ", string(fc*1e-6), " MHz"));
pattern(array,fc,[-180:180],0,...
   `weights',  a_vector(fc, [30; 20]),...
   `CoordinateSystem','polar',...
```

```
    `Type','powerdb',...
    `PlotStyle','overlay')
%
% % use phased.SteeringVector() method to compute the steering vector (aka `array response vector'); then verify against
theoretical computation of the steering vector
% % Motivation: gain an understanding of phased array fundamental principles by studying functions within MathWorks
Phased Array ToolBox
%
% array = phased.ULA(`NumElements',NumElements, `ElementSpacing', delta*lambda);
% steervec = phased.SteeringVector(`SensorArray',array);
% sv = steervec(fc,[az;el])
% sv_angle = angle(sv)

%%
function[sv, sv_angle, Beta] = ULA_basic(R, fc, az, el, NumElements, delta, first_element_offset)
    % Create a uniform linear array from basic principles

    % Input arguments

    % R, az, el
    %   magnitude, azimuth and elevation angles of arrival (in degrees) of impinging signal of interest, as define in Phased
Array Toolbox Help / `Spherical coordinates'
    %   NOTE: the convention defined by Phased Array Toolbox Help / `Spherical coordinates' is not exactly that which is
commonly defined by https://en.wikipedia.org/wiki/Spherical_coordinate_system

    % delta
    %   dimensionless fraction of the carrier wavelength (lambda)

    % first_element_offset
    %   normalized in units of carrier wavelength lambda
    %   ie 0.75 => element_zero is positioned at -3*lambda/4 on the ArrayAxis, by default the y axis
    %   phased.ULA() places the elements symmetrically about the ArrayAxis. The phase center is considered to be at the
origin of the coordinate system.
    %   `first_element_offset' is used in the theoretical computation of the steering vector in order to match the Element
Positions of the standard Phased.ULA

    c = physconst(`LightSpeed');
    lambda = c/fc; % units = meters/cycle

    %%
    % Ref: MATLAB Help / `Spherical coordinates'
    %   Broadside angles are useful when describing the response of a uniform linear array (ULA).
    %   The array response depends directly on the broadside angle and not on the azimuth and elevation angles.
    %   Start with a ULA and draw a plane orthogonal to the ULA axis. For example, if the ULA axis is the y axis, then the
aforementionned plane is the xz plane.
    %   The signal direction is represented by a vector anchored at the origin, normal to the plane wave wavefront that
passes through the origin.
    %   The broadside angle, ?, is the angle between the plane normal to the ULA axis and the signal direction.
    %   To compute the broadside angle:
    %      a) Construct a line from any point on the signal path to the plane normal to the ULA axis, in a direction parallel to the ULA axis.
    %      b) The point where this line reaches the plane is the projection of the signal direction vector (x_s,y_s,z_s) onto that plane. If the ULA axis is the y axis, then this projection is simply (x_s,0,z_s)
    %      c) The angle between the signal direction vector and its projection onto the plane normal to the ULA is the broadside angle and lies in the interval [–90°,90°]. The broadside angle is positive when measured toward the positive direction of the array axis. Zero degrees indicates a signal path orthogonal to the array axis. ±90° indicates paths along the array axis. All signal paths having the same broadside angle form a cone around the ULA axis.

    % ref: To convert (az,el,R) to rectangular coordinates:
    x = R*cosd(el)*cosd(az);
    y = R*cosd(el)*sind(az);
    z = R*sind(el);

    % The conversion from azimuth angle, az, and elevation angle, el, to broadside angle, ?, is
    Beta = asind(sind(az)*cosd(el)); % in degrees

    %% theoretical computation of the steering vector for narrowband signal from far-field with plane wave propagating at the speed of light `c' ...
    % Ref:
    % delay tau = (normal distance from wavefront to element)/c           % units of seconds
    %           = (delta + first_element_offset)*lambda*sind(Beta)*f_0/c;  % first_element_offset in dimensionless units of `delta'; lambda is the wavelength; f_0 is the carrier frequency
    % but f_0   = c(m/s)/lambda(m/cycle)
    % →   tau = (delta + first_element_offset)*sind(Beta);
    sv = exp(-1i*2*pi*sind(Beta)*delta*([0:NumElements-1] - first_element_offset));
    sv_angle = angle(sv);

    % To Do: calculate the element coordinates and compare against getElementPosition(array)

end
```

[i]Beamforming
https://en.wikipedia.org/wiki/Beamforming

[ii]Zynq UltraScale+ RFSoC RF Data Converter v2.4 Gen 1/2/3 (p 169)
https://www.xilinx.com/support/documentation/ip_documentation/usp_rf_data_converter/v2_4/pg269-rf-data-converter.pdf

[iii]David Tse and Pramod Viswanath (2005), *Fundamentals of Wireless Communication* (Chapter 7: MIMO I: spatial multiplexing and channel modeling). Cambridge University Press
www.cambridge.org/9780521845274

[iv]Phased Array System Toolbox™ from MathWorks
https://www.mathworks.com/products/phased-array.html

[vi]Brian D. Jeffs, *Beamforming; A brief introduction.* Dept. of Electrical and Computer Engineering, Brigham Young University
https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.401.667&rep=rep1&type=pdf

[vii]Xilinx UltraScale Architecture DSP Slice User Guide UG579 (v1.10) September 22, 2020 (pp 23, 45-48)

[viii]Xilinx DS 890, UltraScale Architecture and Product Data Sheet: Overview https://www.xilinx.com/support/documentation/data_sheets/ds890-ultrascale-overview.pdf

[ix]Zynq UltraScale+ RFSoC https://www.xilinx.com/products/silicon-devices/soc/rfsoc.html

[x]Zynq UltraScale+ RFSoC RF Data Converter v2.4 Gen 1/2/3 (pp 57-76)
https://www.xilinx.com/support/documentation/ip_documentation/usp_rf_data_converter/v2_4/pg269-rf-data-converter.pdf

[xi]Adaptive Beamforming Tutorial Part 2: Preserving the Signal of Interest
https://grittyengineer.com/adaptive-beamforming-tutorial-part-2-preserving-the-signal-of-interest/

[xii]M. Barrett and R. Arnott, *Adaptive antennas for mobile communications, Electronics and Communication Engineering Journal,* vol. 6, no. 4, pp. 203–214, 1994.

[xii]Yujie Gu and Amir Leshem, *Robust Adaptive Beamforming Based on Interference Covariance Matrix Reconstruction and Steering Vector Estimation, IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 60, NO. 7, JULY 2012*

[xiii]*Sergiy A.Vorobyov, Principles of minimum variance robust adaptive beamforming design*
https://doi.org/10.1016/j.sigpro.2012.10.021

[xiv]Solving Linear Equation Systems
http://qucs.sourceforge.net/tech/node99.html

[xv]Perform QR Factorization Using CORDIC https://www.mathworks.com/help/fixedpoint/ug/perform-qr-factorization-using-cordic.html

[xiv]Minzhen Ren, *CORDIC-BASED GIVENS QR DECOMPOSITION FOR MIMO* DETECTORS, In Partial Fulfillment of the Requirements for the Degree Master of Science in the School of Electrical and Computer Engineering, Georgia Institute of Technology, December 2013

[xvii]FPGA Adaptive Beamforming and Radar Examples with HDL Coder and Zynq RFSoC https://github.com/mathworks/FPGA-Adaptive-Beamforming-and-Radar-Examples

[xviii]5G Explained: Signals for Channel Sounding in 5G NR, MathWorks, https://www.youtube.com/watch?v=3r8Ny5IZFUU

[xix]David Tse and Pramod Viswanath (2005), *Fundamentals of Wireless Communication* (Chapter 7: MIMO I: spatial multiplexing and channel modeling). Cambridge University Press, Section 7.1.2
www.cambridge.org/9780521845274

[xx]Application Note: Versal™ AI Core Devices, *Beamforming Implementation on AI Engine*, XAPP1352 (v1.0) January 11, 2021 http://xilinx.eetrend.com/files/2021-01/wen_zhang_/100060989-119416-xapp1352-beamforming-ai-engine.pdf

Reach Further® Together

ABOUT AVNET

Avnet is a global technology solutions provider with an extensive ecosystem delivering design, product, marketing and supply chain expertise for customers at every stage of the product lifecycle. We transform ideas into intelligent solutions, reducing the time, cost and complexities of bringing products to market. For nearly a century, Avnet has helped its customers and suppliers around the world realize the transformative possibilities of technology.

Learn more about Avnet at www.avnet.com

Avnet
2211 S. 47th Street
Phoenix, AZ 85034
1-800-332-8638
avnet.com