WHITE PAPER

Hybrid Beamforming for Massive MIMO Phased Array Systems

MathWorks®

Introduction

This paper demonstrates how you can use MATLAB® and Simulink® features and toolboxes to:

1. Design and synthesize complex antenna elements and MIMO phased arrays and subarrays
2. Partition hybrid beamforming systems intelligently across RF and digital domains
3. Validate spatial signal processing algorithm concepts
4. Verify link-level designs using high-fidelity simulations
5. Evaluate the impacts of failed or imperfect elements and subarrays
6. Eliminate design problems before building hardware

A fundamental goal of MATLAB and Simulink products for this application is to provide a direct path to expand the level of fidelity of the model over the many phases of project development. This includes tasks such as bringing measured data into the model for the antenna pattern and the propagation paths. It also includes expanding the level of fidelity of the RF chain by bringing in models of RF components in the context of multidomain simulation with Simulink.

Note: In the examples below, we use Phased Array System Toolbox™, Antenna Toolbox™, RF Blockset, RF Toolbox™, Communications System Toolbox™, and Global Optimization Toolbox to complete the associated workflows.

Challenges Designing Massive MIMO Arrays for 5G

As 5G standards continue to evolve, the goals for higher data rates, lower latency network access, and more energy efficient implementations are clear. Higher data rates drive the need for greater bandwidth systems. The available bandwidth in the spectrum up through 6 GHz is not sufficient to satisfy these requirements. This has moved the target operating frequency bands up into the millimeter wave range for the next generation of wireless communication systems.

Intelligent Array Design with Beamforming

Smaller wavelengths at these higher frequency bands enable implementations with more antenna elements per system within small form factors. Signal path and propagation challenges associated with operating at these frequencies also increases. For example, the attenuation due to gas absorption for a 60-GHz waveform is more than 10 dB/km, while a 700-MHz waveform experiences an attenuation on the order of 0.01 dB/km. You can offset these losses with intelligent array design and the use of spatial signal processing techniques, including beamforming. This type of processing is enabled by massive MIMO arrays and can be used directly to provide higher link-level gains to overcome path loss and undesirable interference sources.

To achieve the most control and flexibility with beamforming in an active array design, it is desirable to have independent weighting control over each antenna array element. This requires a transmit/

receive (T/R) module dedicated to each element. For array sizes that are typical of a massive MIMO communication system, this type of architecture is difficult to build due to cost, space, and power limitations. For example, having a high performance ADC and DAC for every channel (along with the supporting components) can drive the cost and power beyond allocated design budgets. Similarly, having variable gain amplifiers in the RF chain for each channel increases the system cost.

Hybrid Beamforming

Hybrid beamforming is a technique you can use to partition beamforming between the digital and RF domains. System designers can implement hybrid beamforming to balance flexibility and cost trade-offs while still fielding a system that meets the required performance parameters. Hybrid beamforming designs are developed by combining multiple array elements into subarray modules. A transmit/receive (T/R) module is dedicated to a subarray in the array and therefore fewer T/R modules are required in the system. The number of elements, and the positioning within each subarray, can be selected to ensure system-level performance is met across a range of steering angles.

Using the transmit signal chain as our first example, each element within a subarray can have a phase shift applied directly in the RF domain, while digital beamforming techniques based on complex weighting vectors can be applied on the signals that feed each subarray. Digital beamforming allows control of the signal for both amplitude and phase on signals aggregated at the subarray level. For cost and complexity reasons, the RF control is typically limited to applying phase shifts to each of the elements.

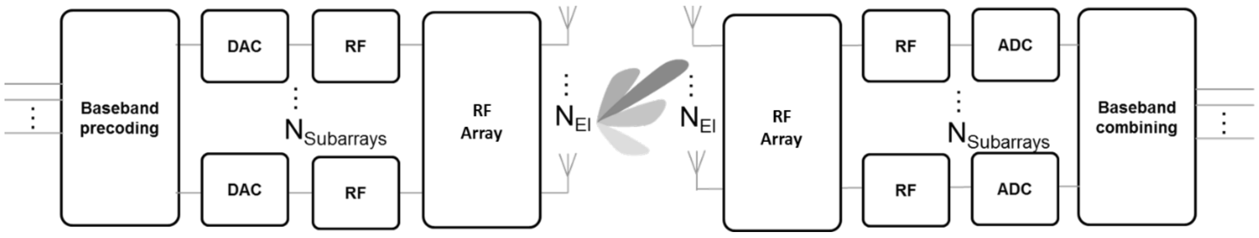


Figure 1. Hybrid beamforming architecture.

Systems such as the one shown in Figure 1 are complex to develop. You can use modeling techniques to design and evaluate massive MIMO arrays and the corresponding RF and digital architectures needed to help manage their complexity. With these techniques, you can reduce risk and validate design approaches at the earliest stages of a project. We will first look at an array design example.

We have selected parameters for each of the examples that are common in the 5G wireless community but all of the examples shown can be modified to match your desired configuration.

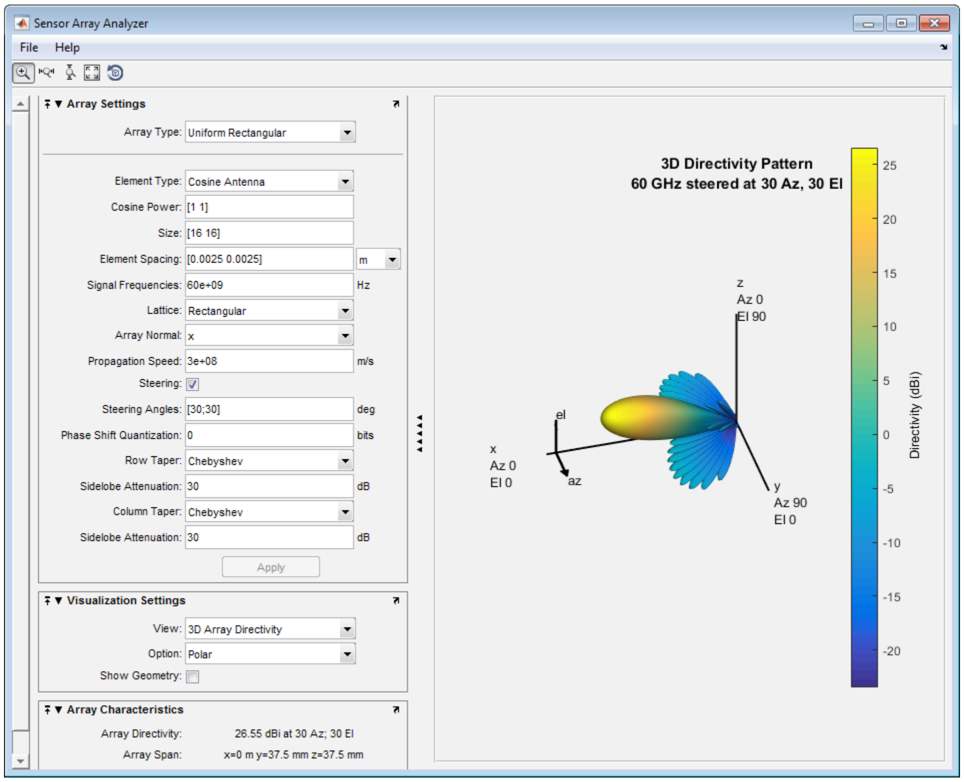
Designing the Array

There are many factors to consider when designing an array. Typical array designs include parameters such as array geometry, element spacing, the lattice structure of the elements, and element tapering. In addition, the effects of mutual coupling are important to characterize before the final design is implemented. Once an initial configuration of the array design is complete, architectural partitioning can be iteratively evaluated against the overall system performance goals.

With millimeter wave systems, the area of the array is reduced in proportion to the wavelength size. As an example, an antenna array designed at millimeter wave frequencies can be up to 100 times smaller than an array designed to operate at microwave frequencies. By building an array with a larger number of antenna elements, you can achieve a high beamforming gain. The highly directive beam helps to offset the increased path loss at higher frequencies of operations, as beams are steered to a specific direction.

To start the array design process, the Sensor Array Analyzer app, which is available with Phased Array System Toolbox, can be launched from the MATLAB prompt:

```
>> sensorArrayAnalyzer
```



Array Settings

Array Type:	Uniform Rectangular	
Element Type:	Cosine Antenna	
Cosine Power:	[1 1]	
Size:	[16 16]	
Element Spacing:	[0.0025 0.0025]	m
Signal Frequencies:	60e+09	Hz
Lattice:	Rectangular	
Array Normal:	x	
Propagation Speed:	3e+08	m/s
Steering:	☑	
Steering Angles:	[30;30]	deg
Phase Shift Quantization:	0	bits
Row Taper:	Chebyshev	
Sidelobe Attenuation:	30	dB
Column Taper:	Chebyshev	
Sidelobe Attenuation:	30	dB

Visualization Settings

View:	3D Array Directivity
Option:	Polar
Show Geometry:	☐

Array Characteristics

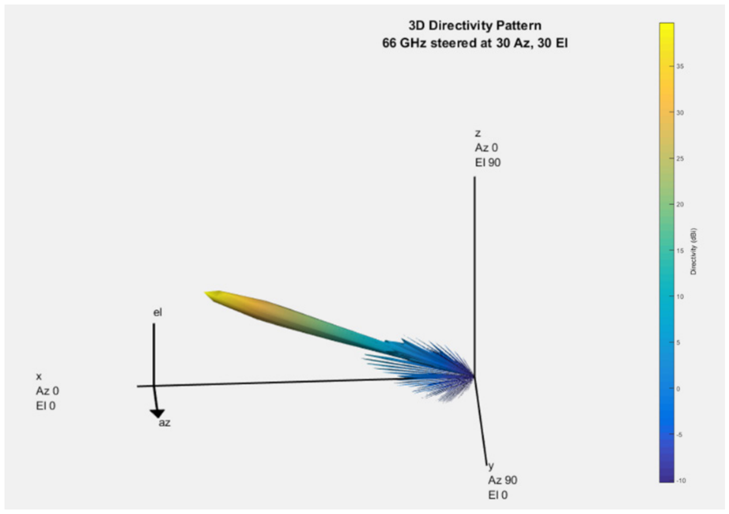
Array Directivity:	26.55 dBi at 30 Az; 30 El
Array Span:	x=0 m y=37.5 mm z=37.5 mm

Figure 2. Sensor Array Analyzer app for Array Design.

You can edit all design parameters that define an array directly on the left side of the app in the Array Settings window shown in Figure 2. The parameters include array size, array geometry, element spacing, and tapering.

From the app, you can easily visualize the resulting geometry, 2D and 3D Directivity, and the Grating Lobe Diagrams.

To achieve steering in azimuth and elevation, you can design a uniformly spaced planar array. Figure 3 shows an example of a 64x64 uniform rectangular array that was designed within the Sensor Array Analyzer app. The large number of elements provides a high level of directivity. The design shown below also has tapering applied to the rows and columns of the array to reduce sidelobe levels. As is the case with all design choices, the larger antenna gains achieved with narrower beams must be balanced with the fact that MIMO systems are based on scattering environments that also depend on broader beam patterns to maximize channel capacity. This trade-off can also be assessed during the interactive design process.



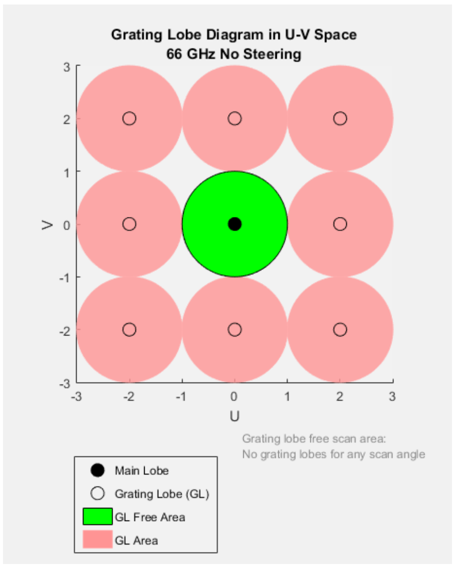


Figure 3. Beam pattern and grating lobe diagram for 66 GHz 64x64 element design.

The image displayed in the right side of Figure 3 shows that with half wavelength spacing between the elements, there are no grating lobes present across the full range of steering directions as expected. It is important to understand the impacts here because it may be necessary to increase the spacing between the elements to mitigate the effects of mutual coupling. This is an important design consideration that needs to be accounted for. Fortunately, at higher frequencies where half wavelength spacing is small to start with, an increase in element separation by 10% of a wavelength only requires a change of less than 0.5 mm at 66 GHz. Figure 4 shows the trade-off that must be considered using a grating lobe diagram with a 10% increase in the spacing between the elements. For this example, grating lobes are only present with azimuth and elevation angles outside +/- 54.9 degrees. This can be traded off against the array with less space between elements (and more mutual coupling effects).

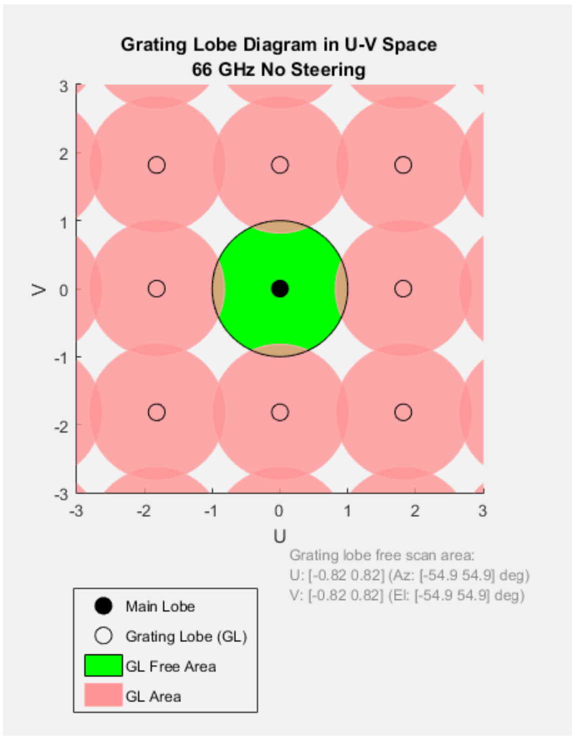
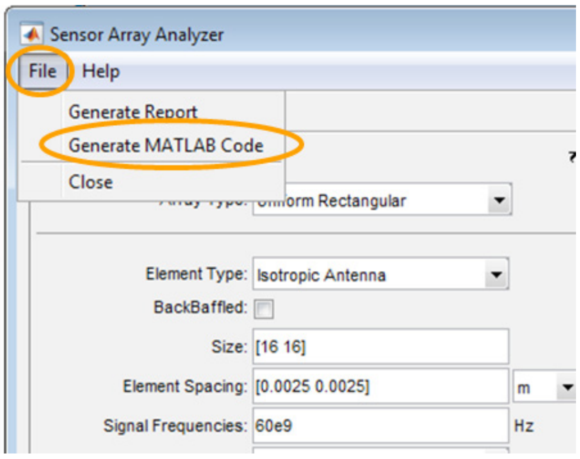


Figure 4. Grating lobe diagram with element spacing larger than half a wavelength

When the process of designing the array is complete, you can generate MATLAB code from the app and either use directly in your model or as a starting point for further customization, as shown in Figure 5.



Array Type: Uniform Rectangular
Element Type: Isotropic Antenna
BackBaffled: ☐
Size: [16 16]
Element Spacing: [0.0025 0.0025] m
Signal Frequencies: 60e9 Hz

Figure 5. Generating MATLAB code from the Sensor Array Analyzer.

Extending the Model Fidelity: Antenna and RF

In the previous example, an ideal antenna element was used to model the array pattern. There are multiple ideal elements choices available to get started with, including an omnidirectional and a cosine element. The element being used in the next example is no longer ideal and is based on a patch antenna designed for 66 GHz resonance. The full MATLAB example for this type of antenna element design is located *here.*

We have extracted some of the key code sections to show how an antenna can be quickly designed in Antenna Toolbox. We use a patch micro-strip element that resonates at 66GHz in our example. The resulting pattern is also shown below in Figure 7.

We start with a patch element in the Antenna Toolbox library and directly modify the patch parameters to operate at 66GHz. The code sample and patch structure (shown in Figure 6) are shown below.

```
p = patchMicrostrip;
p.Length     = 0.49*lambda;
p.Width      = 1.5*0.49*lambda;
p.Height     = 0.01*lambda;
p.GroundPlaneLength = lambda;
p.GroundPlaneWidth  = lambda;
```

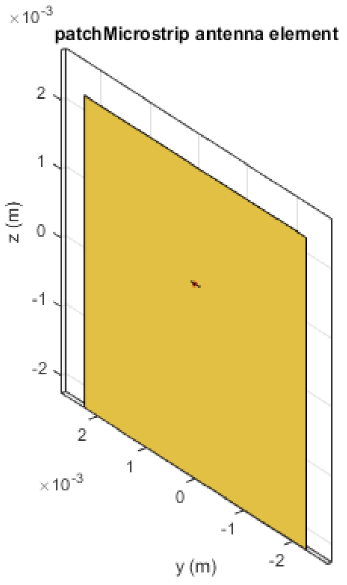
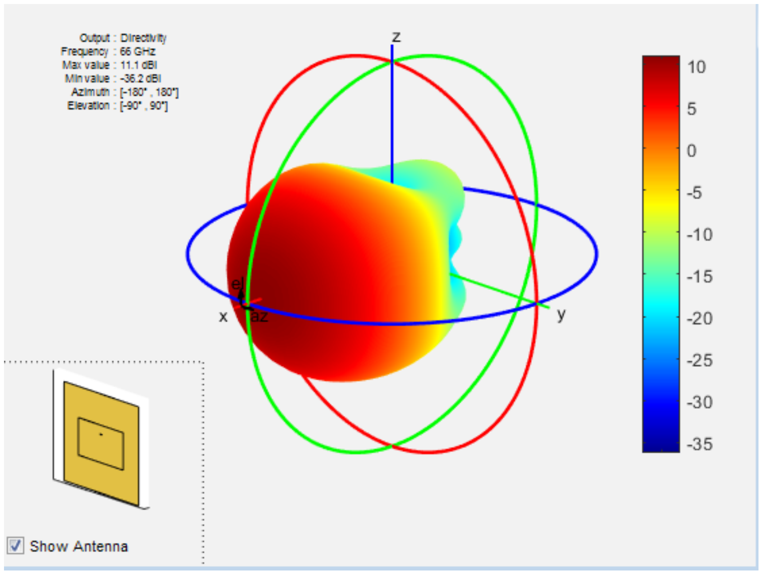


Figure 6. Patch Microstrip Element.

We generate the pattern of the patch element in free space using the full wave EM solver in Antenna Toolbox where F0 = 66e9:

```
P _ isolated = pattern(p, F0);
figure
pattern(p, F0);
```



```
Output : Directivity
Frequency : 66 GHz
Max value : 11.1 dBi
Min value : -36.2 dBi
Azimuth : [-180° , 180°]
Elevation : [-90° , 90°]
```

Figure 7. Element pattern generated using a full wave EM solver in Antenna Toolbox.

Note that we modified the patch element parameters directly in the code above, but there is also a dedicated function in Antenna Toolbox which you can use to generate the parameters directly for any library element and frequency combination. In this example it would be:

```
 p = design(patchMicrostrip,66e9)
```

Next, we build a uniform linear array (ULA) which serves as the subarray in this example. We then create a full array based on a collection of multiple subarrays. From the code shown above, we generate the pattern, P_isolated, for each element in the subarray. P_isolated is defined as a pattern across a range of azimuth and elevation angles.

We model an 8x1 element uniform linear array, where each element has a pattern response from the patch element. Eight subarrays are then replicated to form an 8x8 array using the MATLAB code shown below. Note that the taper for the elements in each subarray can be applied directly within the subarray. Hamming weights are added to reduce the level of sidelobes in the resulting pattern.

```
%% Definition of custom antenna element for phased .ULA arrays

patchElement = phased.CustomAntennaElement;

patchElement.AzimuthAngles = (-180:5:180);

patchElement.ElevationAngles = (-90:5:90);

patchElement.RadiationPattern = P _ isolated;


%Phased array design using pattern superposition of the isolated element


numElementsA = 8; % number antenna element in each subarray

numElementsS = 8;  % number of subarrays in an array

% subarray design (patches stacked vertically)

sULA = phased.ULA('NumElements',numElementsA,...

       'Element',patchElement,...

       'ElementSpacing', lambda/Spacing,...

       'ArrayAxis','z','Taper', hamming(8));

% array design (subarrays stacked horizontally)

aURA = phased.ReplicatedSubarray('Subarray',sULA,...

       'GridSize',[1 numElementsS],...

       'SubarraySteering','Phase',...

       'PhaseShifterFrequency', F0,...

       'GridSpacing', lambda/Spacing);
```

Phased Array System Toolbox makes it easy to build a large array by replicating subarrays using the phased.ReplicatedSubarray System object, as shown in the code above.

The resulting array structure can be visualized as follows, where each subarray (8 elements x 1 subarray) is shown on the left of Figure 8. The full array (1 subarray replicated in 8 columns) is shown on the right.

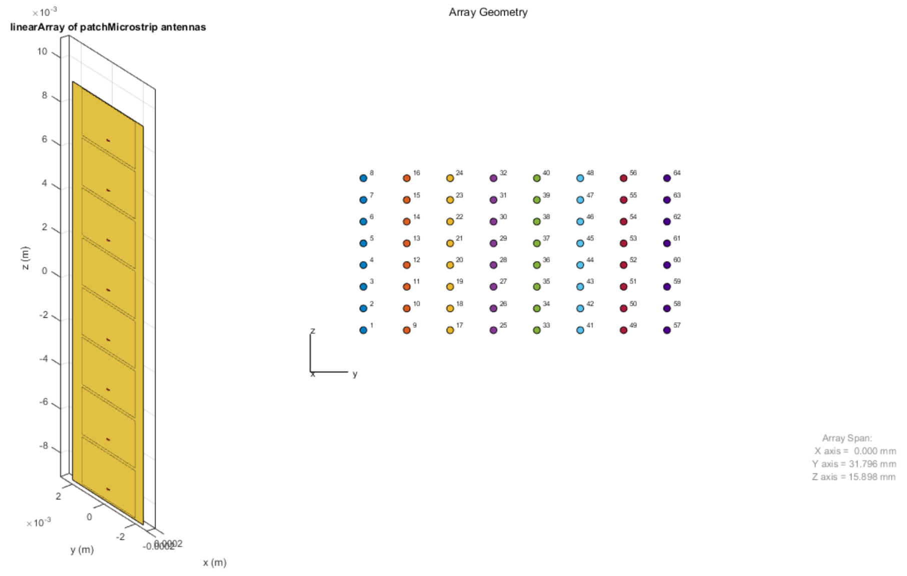


Figure 8. 8x1 ULA subarray and corresponding full array.

The full array is shown in Figure 9.

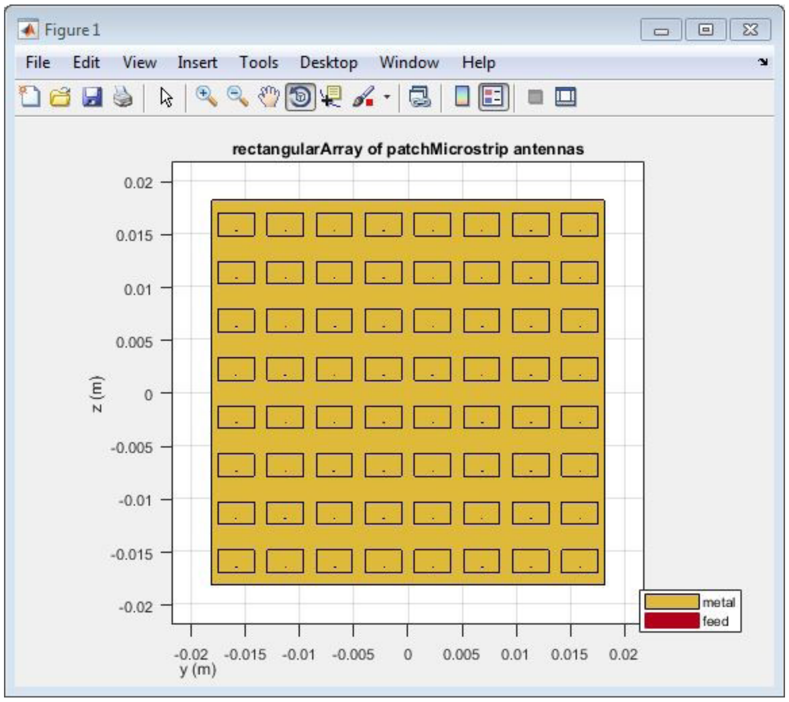


Figure 1
File Edit View Insert Tools Desktop Window Help
rectangularArray of patchMicrostrip antennas
z (m)
y (m)
metal
feed

Figure 9. Corresponding full array.

From the hybrid beamforming perspective, you can pass each of the signals that drive elements within the 8x1 array through a phase shifter for steering in the elevation plane. We show how this can be modeled in the RF domain in the following section. In addition, each of the signals that feed the eight subarrays can be controlled via digital beamforming techniques to steer the beam in the azimuth direction.

The resulting beam pattern for this array configuration, which has been calculated using superposition, is shown in Figure 10. With this combination of RF and digital beamforming, you can achieve more granularity in the steering angle in the azimuth direction. Figure 11 provides a comparison of the subarray pattern computed with superposition and Method of Moments.

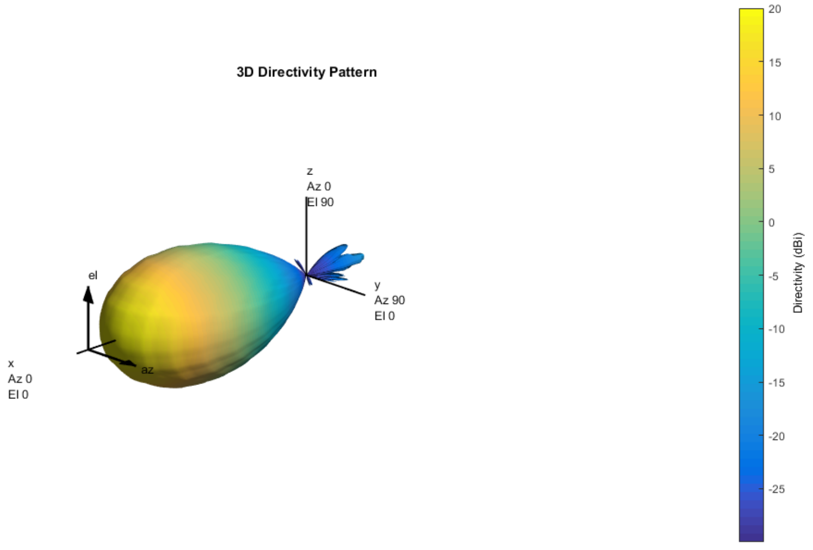


Figure 10. Array pattern generated using superposition techniques with Phased Array System Toolbox.

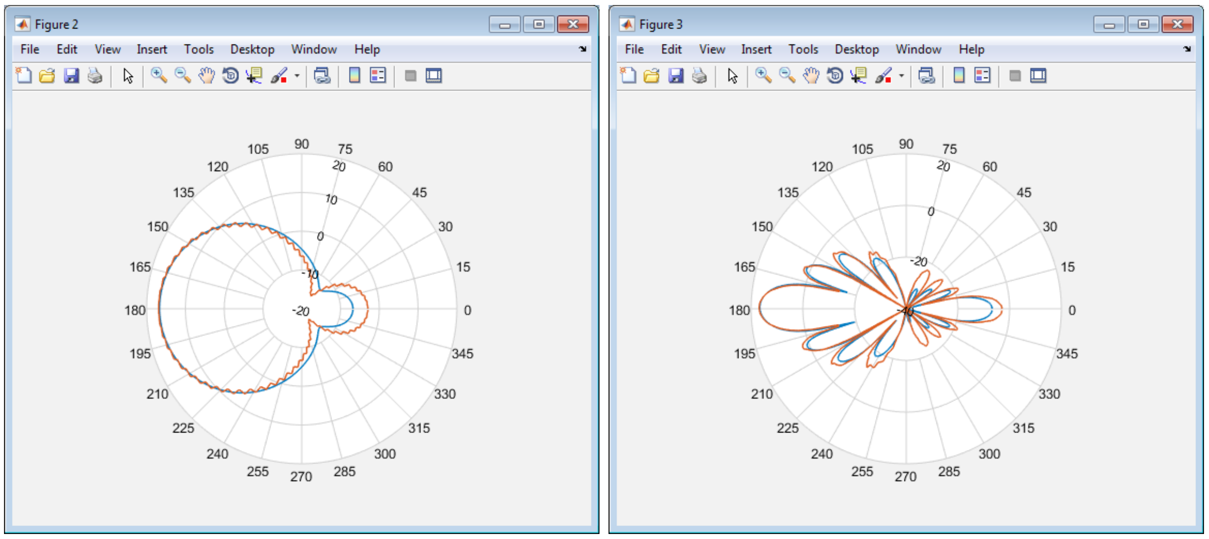


Figure 11. Comparison of the subarray pattern computed with superposition and Method of Moments.

In this example, we start with a partition of our architecture for the transmit chain with phase shifts (applied in the RF domain) and complex weights (applied in the digital domain). For basic analysis, you can generate the weights using MATLAB and Phased Array System Toolbox, as shown in the code below.

```
% complex weights used as part of digita; baseband precoding
wT _ digital = steervec(subpos,[tp.steeringAngle;0]);

% analog phase shift values used as part of RF precoding
wT _ analog = exp(1i*angle(steervec(subelempos,[tp.steeringAngle;0])));

%%
% From the system perspective, the effect of the hybrid beamforming can
% be represented by hybrid weights as shown below.
wT _ hybrid = kron(wT _ digital,wT _ analog);
```

Combined with the array design parameters built up earlier, the digital weights and the RF phase shifts generated in the MATLAB code above can be applied using an architecture model in Simulink, which can then be part of a multi-domain system simulation (as shown in Figure 12).

In this block diagram, you can see that the phase shifts are provided as inputs to each of the subarrays, which are then applied to the RF signals. The digital beamforming weights are used to shape the signals feeding each of the subarrays.

RF Blockset is used within Simulink to perform the circuit envelope simulation (note that the circuit envelope allows you to achieve fast simulation). RF Blockset contains a library of RF components such as amplifiers, mixers, filters, couplers splitters, and other typical parts that you can use to create an RF chain. This is done to increase the level of fidelity of the model.

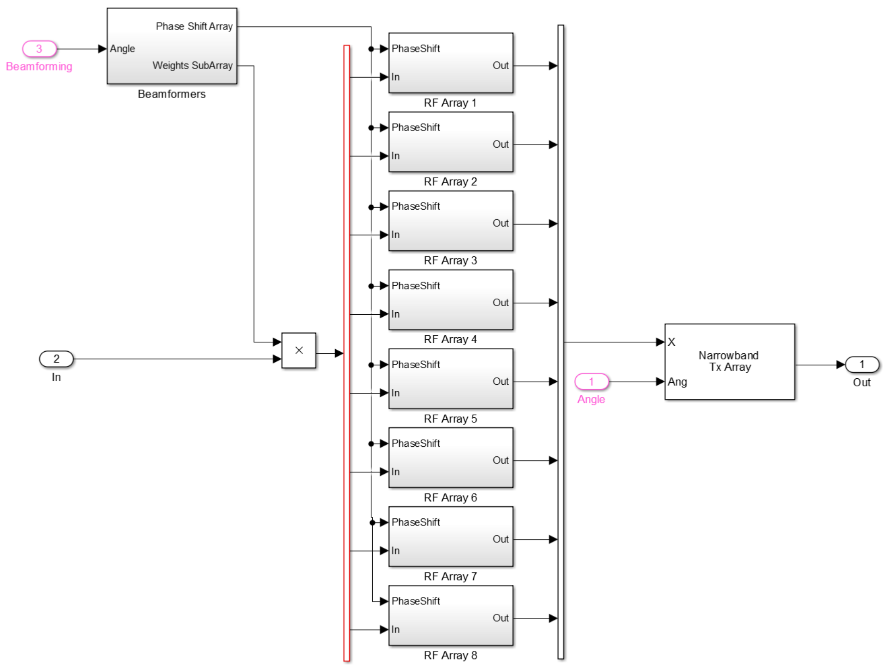


Figure 12. Multi-domain hybrid architecture in Simulink and RF Blockset.

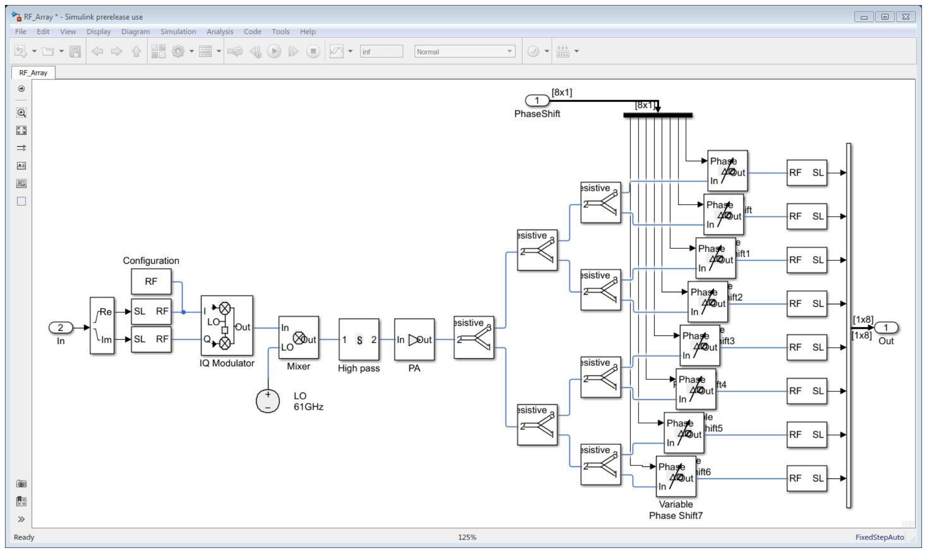
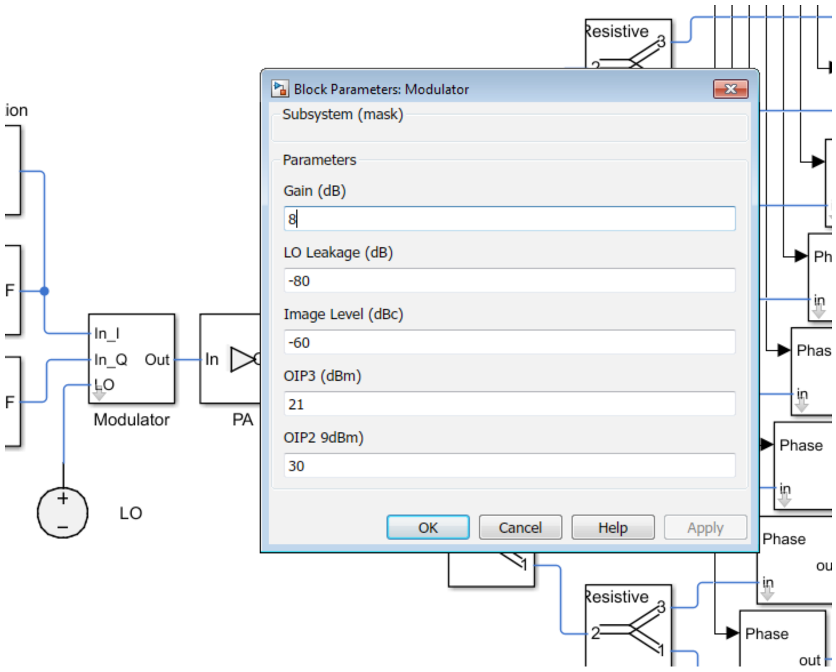


Figure 13. RF transmit chain using RF Blockset blocks to control phase shifters.

Figure 13 provides a detailed view into a single RF Array block from Figure 12. The RF phase shifters shown in Figure 13 perform the beamforming in the elevation plane, while the baseband weights provide the beamforming in the azimuth plane.

You can configure each of the blocks with parameters taken from the data sheet of the supplier. The power amplifier and the modulator blocks are shown below, in Figures 14 and 15, to illustrate this capability.



Subsystem (mask)

Parameters

Gain (dB)
8

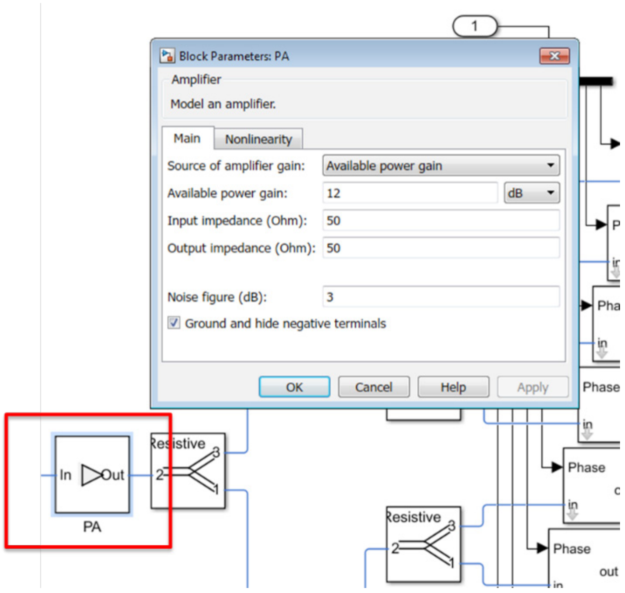
LO Leakage (dB)
-80

Image Level (dBc)
-60

OIP3 (dBm)
21

OIP2 9dBm)
30

Figure 14. Example of modulator.



Amplifier

Model an amplifier.

Source of amplifier gain: Available power gain

Available power gain: 12 dB

Input impedance (Ohm): 50

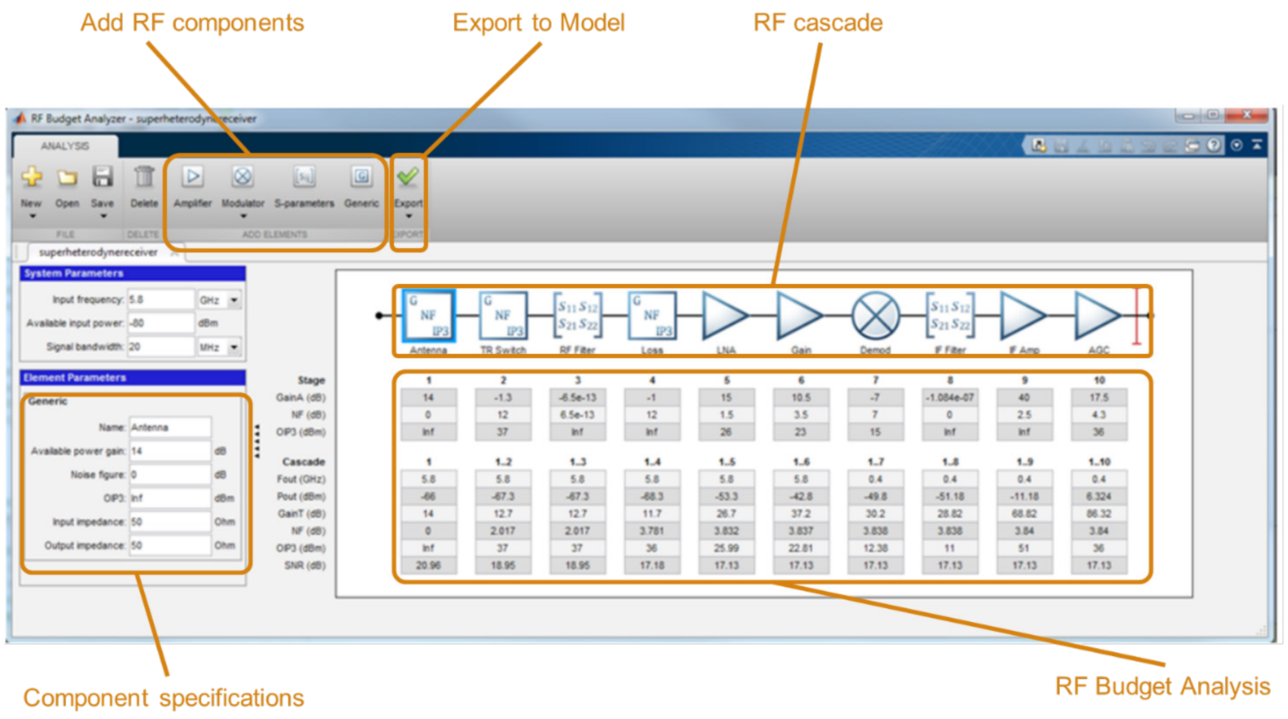
Output impedance (Ohm): 50

Noise figure (dB): 3

☑ Ground and hide negative terminals

Figure 15. Example of power amplifier.

An alternative approach to creating the RF chain in the model involves using the RF Budget Analyzer, which is part of RF Toolbox, (shown in Figure 16). Here, you can build up your RF chain directly in the app, including devices represented with S-parameters, mixers, amplifiers, and filters. This app provides an intuitive interface to build up your link budget directly. You can export the resulting cascade directly to the system model using the **Export** option on the tool strip.



System Parameters

Input frequency:	5.8	GHz
Available input power:	-80	dBm
Signal bandwidth:	20	MHz

Element Parameters

Generic

Name:	Antenna	
Available power gain:	14	dB
Noise figure:	0	dB
OIP3:	Inf	dBm
Input impedance:	50	Ohm
Output impedance:	50	Ohm

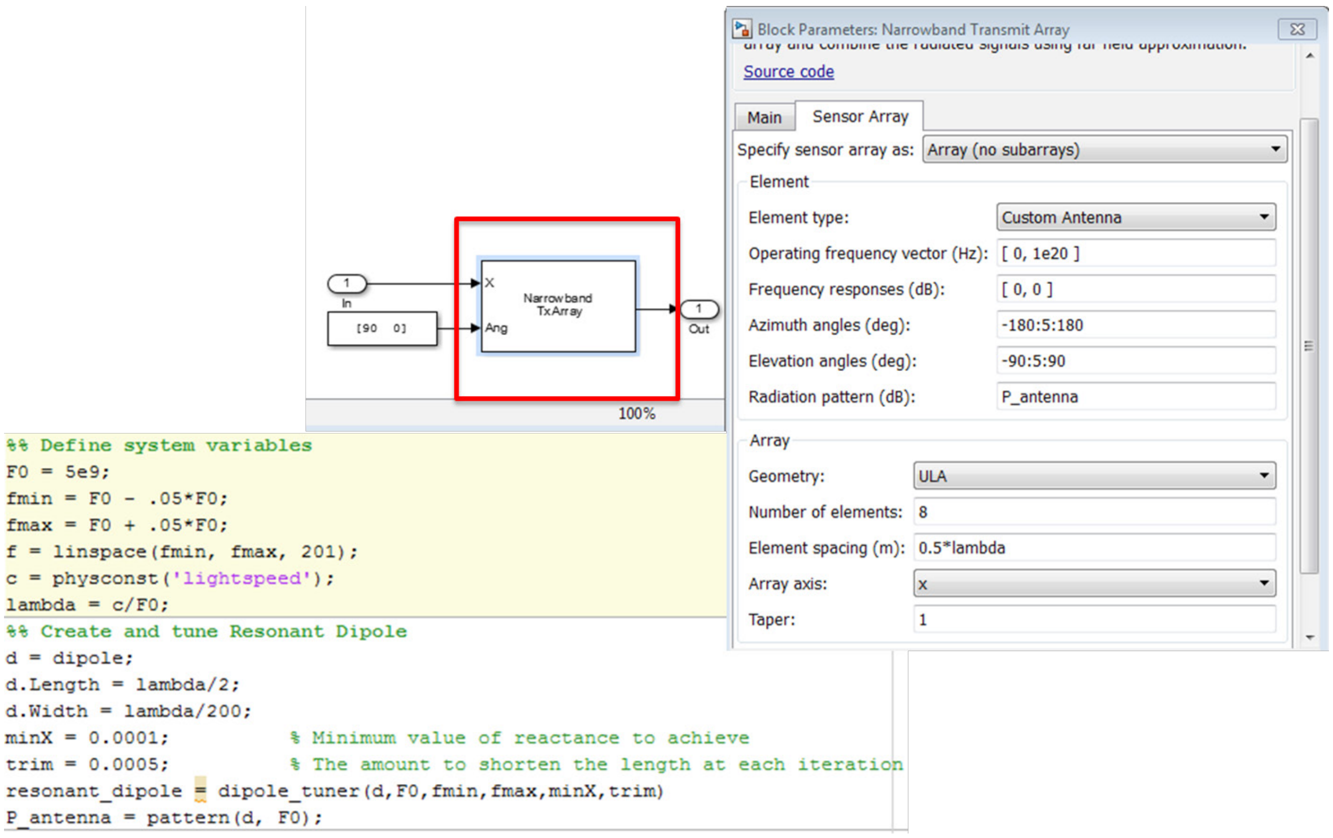
Stage	1	2	3	4	5	6	7	8	9	10
GainA (dB)	14	-1.3	-6.5e-13	-1	15	10.5	-7	-1.084e-07	40	17.5
NF (dB)	0	12	6.5e-13	12	1.5	3.5	7	0	2.5	4.3
OIP3 (dBm)	Inf	37	Inf	Inf	26	23	15	Inf	Inf	36

Cascade	1	1..2	1..3	1..4	1..5	1..6	1..7	1..8	1..9	1..10
Fout (GHz)	5.8	5.8	5.8	5.8	5.8	5.8	0.4	0.4	0.4	0.4
Pout (dBm)	-66	-67.3	-67.3	-68.3	-53.3	-42.8	-49.8	-51.18	-11.18	6.324
GainT (dB)	14	12.7	12.7	11.7	26.7	37.2	30.2	28.82	68.82	86.32
NF (dB)	0	2.017	2.017	3.781	3.832	3.837	3.838	3.838	3.84	3.84
OIP3 (dBm)	Inf	37	37	36	25.99	22.81	12.38	11	51	36
SNR (dB)	20.96	18.95	18.95	17.18	17.13	17.13	17.13	17.13	17.13	17.13

Figure 16. RF Budget Analyzer.

It is worth noting that the final block in our example also includes the detailed model of the array described earlier. The pattern (represented as **P_antenna**) includes the effects of mutual coupling and is used directly in the array as a Custom Antenna. **P_antenna** is defined as a radiation pattern across azimuth and elevation angles. Note that a pattern measured from an actual element could also be imported into the model in this same way.

Also, the array parameters of an 8-element ULA are also included in this same block, as shown in Figure 17 below.



Block Parameters: Narrowband Transmit Array

Source code

Main | Sensor Array

```
Specify sensor array as: Array (no subarrays)
Element
Element type: Custom Antenna
Operating frequency vector (Hz): [ 0, 1e20 ]
Frequency responses (dB): [ 0, 0 ]
Azimuth angles (deg): -180:5:180
Elevation angles (deg): -90:5:90
Radiation pattern (dB): P_antenna
Array
Geometry: ULA
Number of elements: 8
Element spacing (m): 0.5*lambda
Array axis: x
Taper: 1
```

```
%% Define system variables
F0 = 5e9;
fmin = F0 - .05*F0;
fmax = F0 + .05*F0;
f = linspace(fmin, fmax, 201);
c = physconst('lightspeed');
lambda = c/F0;
%% Create and tune Resonant Dipole
d = dipole;
d.Length = lambda/2;
d.Width = lambda/200;
minX = 0.0001;          % Minimum value of reactance to achieve
trim = 0.0005;          % The amount to shorten the length at each iteration
resonant_dipole = dipole_tuner(d,F0,fmin,fmax,minX,trim)
P_antenna = pattern(d, F0);
```

Figure 17. Using the Array Design within the Simulink block.

Incorporating Optimization Techniques to Improve the Beam Pattern

So far, we have been focused on setting up a specific configuration and partitioning between digital and RF subsystems. We can continue to build our system link-level model and see how performance changes as the steering weights change, as well as how frequency may impact the performance.

This can be a manually intensive process if there are a large number of elements and a wide frequency band of operations.

Alternatively, we can take advantage of optimization techniques from Optimization Toolbox™ and Global Optimization Toolbox to iteratively understand how array element spacing and element tapering can be tuned to achieve the desired performance for a hybrid beamforming system.

Figure 18 below demonstrates how this can be accomplished. For example, you can try to match a specific beam pattern or you may want to drive the attributes of a beam pattern in a specific direction (e.g. lower sidelobes, more narrow beamwidth, etc.).

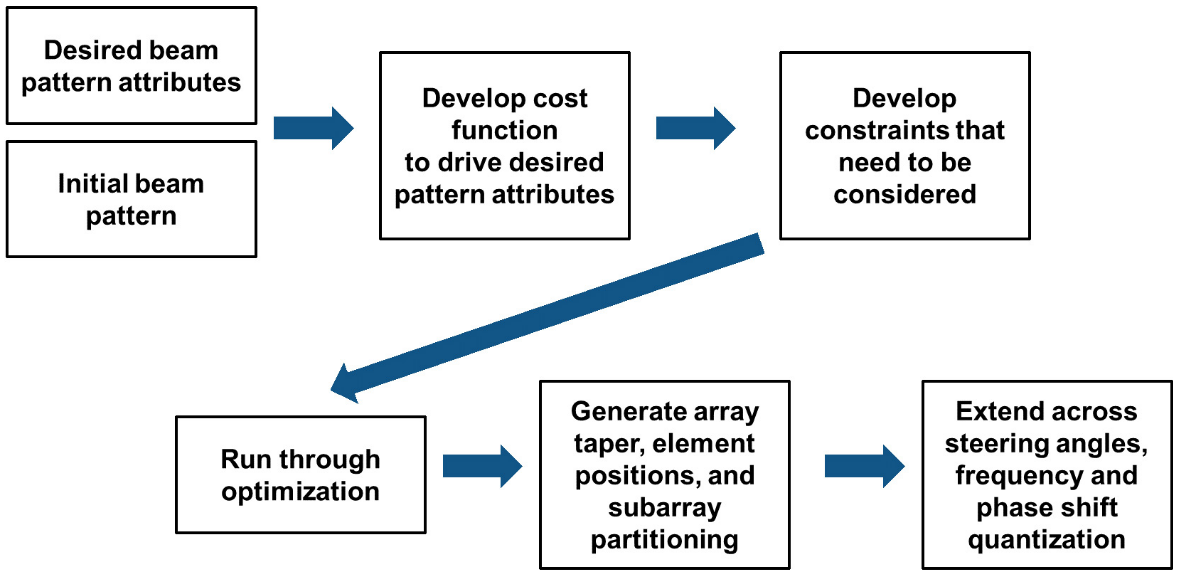

Desired beam pattern attributes
Initial beam pattern
Develop cost function to drive desired pattern attributes
Develop constraints that need to be considered
Run through optimization
Generate array taper, element positions, and subarray partitioning
Extend across steering angles, frequency and phase shift quantization

Figure 18. Array Synthesis Workflow.

Global Optimization Toolbox provides solvers that can be used when there are many locally optimal solutions (or when the functions are not smooth). In our example, we are looking to get the best performance across a set of steering angles and frequencies, which translates to the need for multiple optimal solutions.

Constraints for the outputs, for example weights and element positions, can be set up as part of the optimization. This can include parameters that bound the number of elements per subarray, where the elements are located in the subarray. It can also include accounting for the effects of phased shift quantization. You can use this general capability to ensure that the design coming out of optimization is actually buildable.

Once the array is set up, you can determine the beam patterns across both the azimuth and elevation angles. You can then use this data directly to extract the key metrics associated with the pattern. This example focuses on the main lobe, sidelobes, and beamwidth, but many other parameters could be considered.

For a detailed example, see the code available ***here***.

Using the Model for Life Cycle Analysis and Calibration Framework Development

Before reviewing ways to assess link-level performance, it is interesting to note that you can use the model to support a variety of specific "what-if" analysis exercises that relate to more detailed design trade-offs and life cycle planning. For example, with the resulting modeling framework in place, you can find the best implementation for array thinning. You can evaluate the relative impact of failed elements in the array. This is important for determining maintenance cycles. For an array that is not staffed 24/7, multiple failures can be tolerated before a site is visited and the failures are repaired. The beam pattern in Figure 19 shows the degradations in the beam pattern with 15% of the elements failed.

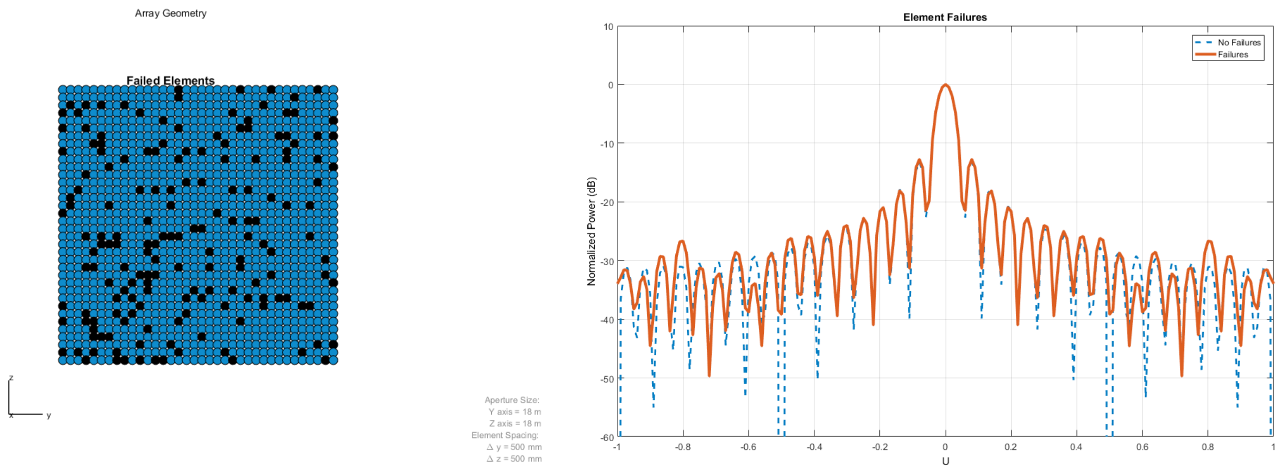


Figure 19. Failed Element Analysis.

You can perform similar analysis at the subarray level as well. An example is shown below in Figure 20, where the array is built up from 6x6 subarrays. The resulting beam pattern is also shown with 10 of the 36 subarrays in a failed state. Again, you can use this type of data to determine how many sub-arrays should be implemented. It can also be used in a way similar to the maintenance concept described earlier.

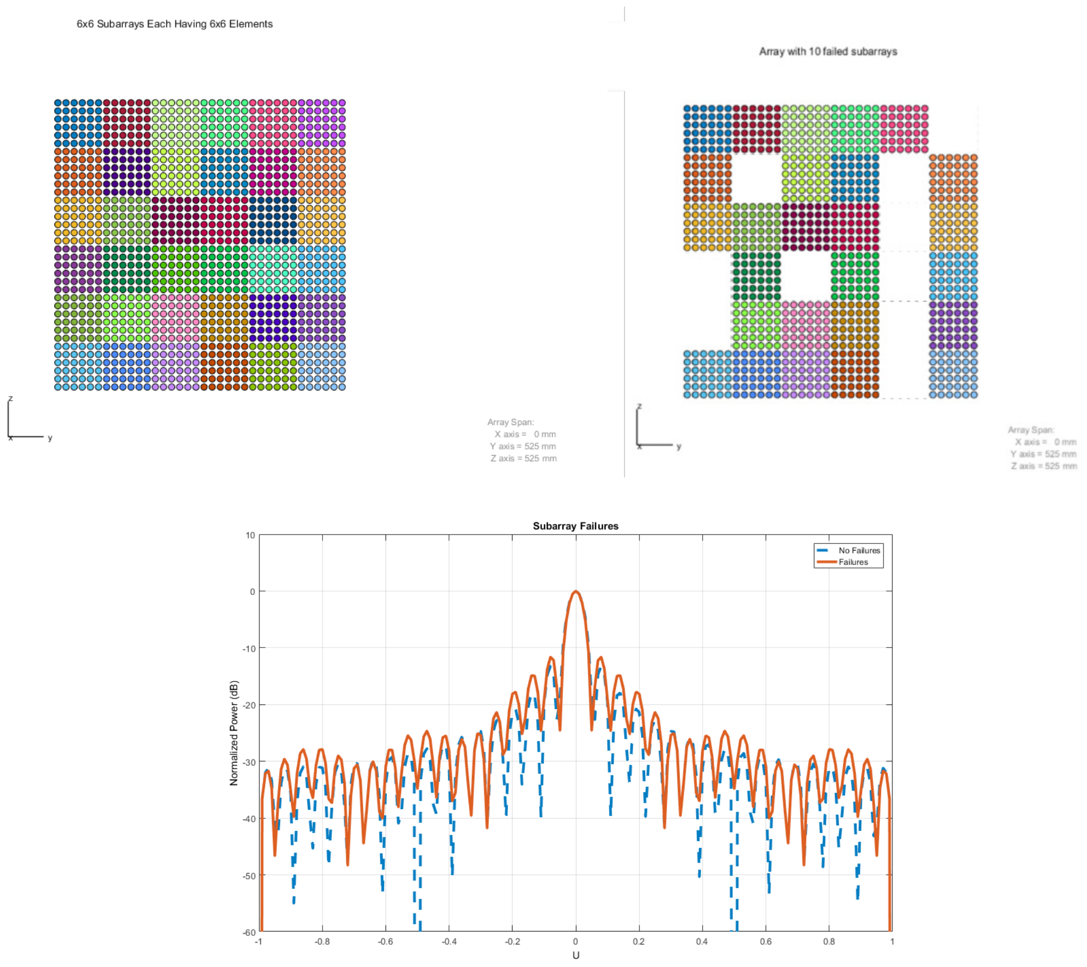


Figure 20. Failed Subarray Architecture.

Assessing Link-level Performance

Once the array, subarray, and beamforming design are completed, you can implement a larger system around the array and subarrays. You can setup scenarios and signal processing algorithms, including beamforming and DOA integration. There are multiple ways to visualize the link-level performance, including the constellation diagrams shown below. The ***link to this example*** demonstrates how this can be accomplished.

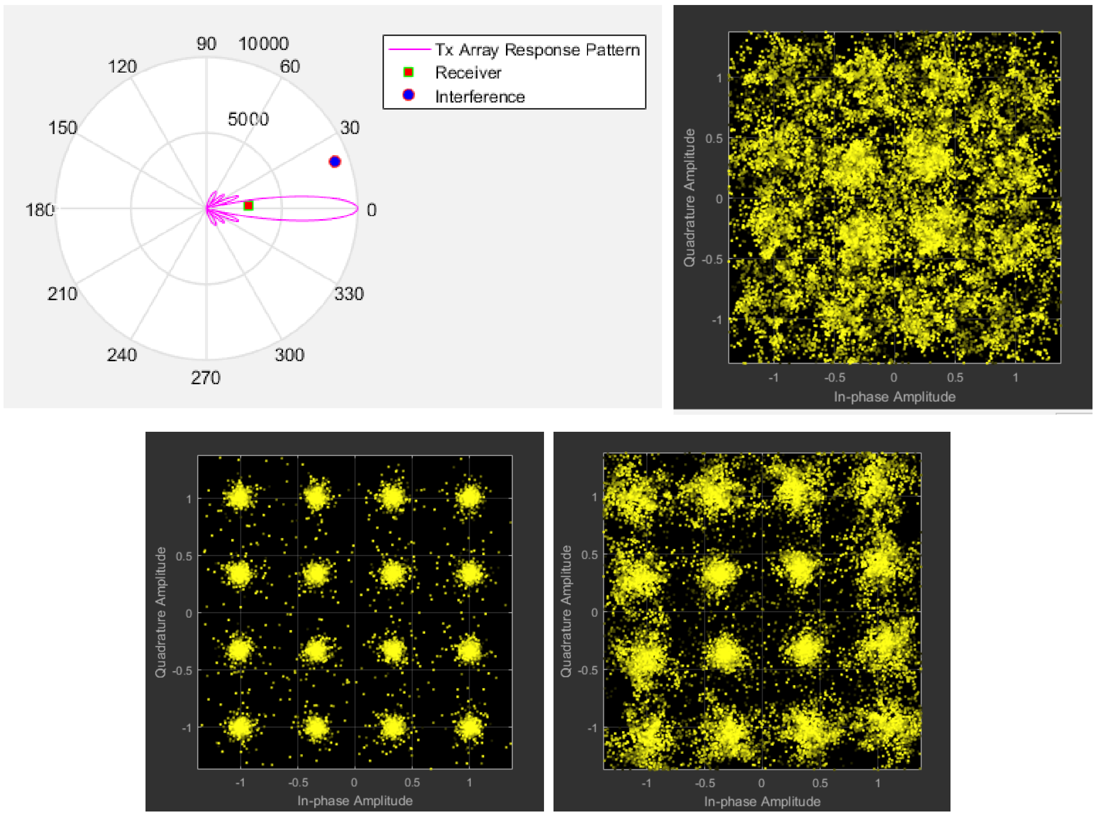


Figure 21. Link-level performance assessment.

Multi-Beam Hybrid System Architectures

The model can be extended to support multi-user beamforming systems. You can use the baseband beamforming blocks described above to create multiple beams from the array to cover multiple users concurrently, as shown in Figure 22. You can also use this beamforming to account for the path variations between the transmitter and desired user locations.

You can combine the resulting signals to perform the RF beamforming and serve the different users in a sector at specific distances from the base station.

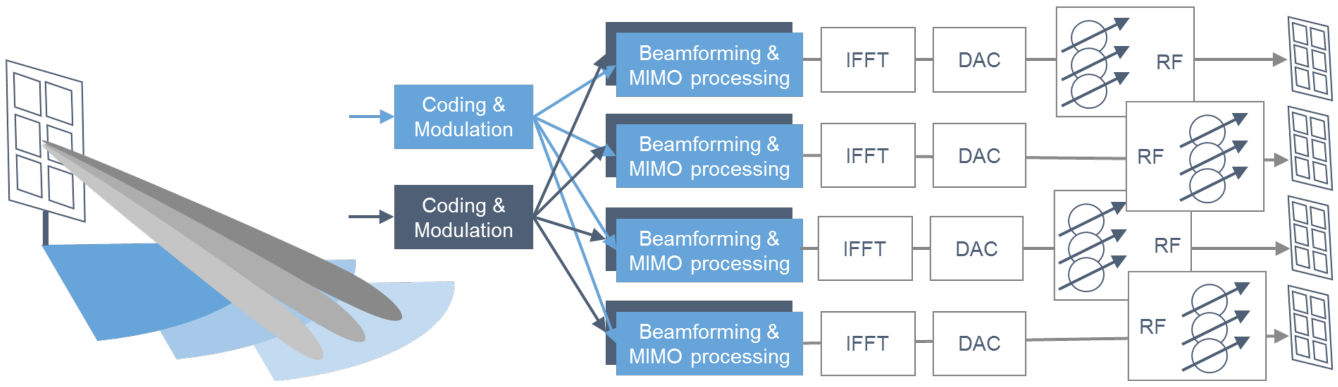


Figure 22. Multi-beam hybrid system architecture.

Summary

MIMO arrays and the corresponding RF and digital architectures are critical components of 5G designs. These components also drive the related hybrid beamforming systems. A balance must be reached in these systems to meet system performance goals and system-level cost objectives. Modeling and simulation techniques can help reduce the risk associated with this complex workflow. Higher levels of fidelity can be added across the project life cycle to bring the model in line with the end system implementation.

Developing a hybrid beamformer and evaluating algorithm alternatives is only the first step toward achieving the required performance of a wireless communications system. To assess performance, the beamformer must be integrated into a system-level model and evaluated over a collection of parameter, steering, and channel combinations.

Modeling these beamforming algorithms in the context of an entire system, including RF, antenna, and signal processing components, can help verify design choices at the earliest phases of the project and reduce the associated challenges.

In MATLAB you can design antenna elements with a full wave EM solver. You can use the resulting element patterns directly in your complex array design models. It is easy to iterate the key parameters of the array, including the geometry, tapering, element spacing, and the lattice structure between elements. As you design the array, you have access to a full set of visualizations including 2D and 3D directivity and grating lobe diagrams. Alternatively, you can start with a beam pattern and synthesize an array which generates this desired pattern.

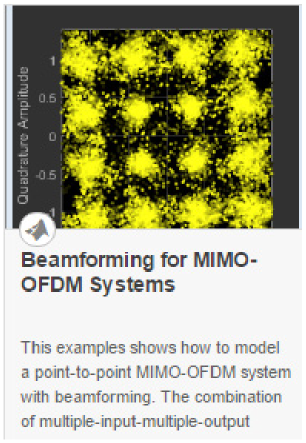
You have a library of spatial signal processing algorithms available to be used in your system model, including narrowband and wideband beamforming and direction of arrival (DOA) estimation algorithms.

You can use individual components, such as baseband receive and transmit systems, and connect within MATLAB to form a system link-level model where your designs can be evaluated and tested in the context of overall system performance. Many supporting components are also provided, including model propagation effects due to atmospheric conditions (e.g. rain, gas, fog), multi-path reflections, and platform motion modeling.

In addition to the system modeling described above, you can use the phased array model to evaluate areas such as equipment life cycle planning (failed element and subarray impacts) or to help with the development of calibration frameworks (to correct for imperfect elements and arrays).

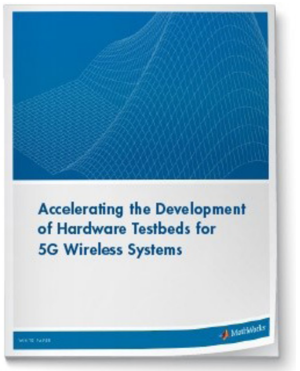
Expanding the level of fidelity in the RF domain can be valuable as your project evolves. You have multiple ways to do this with your antenna and RF designs.

Learn More



Beamforming for MIMO-OFDM Systems

This examples shows how to model a point-to-point MIMO-OFDM system with beamforming. The combination of multiple-input-multiple-output



Accelerating the Development of Hardware Testbeds for 5G Wireless Systems

Get code example:

Beamforming for MIMO-OFDM Systems

Download white paper:

Accelerating the Development of Hardware Testbeds for 5G Wireless Systems

Visit resource center:

Wireless Communications Design with MATLAB