Proceedings,
The 2013 ASEE North Midwest Section Conference,
October 17-18, 2013, Fargo, North Dakota.

MATLAB® Simulation Tool for Antenna Array Pattern Development

Jon J. Smith and Sima Noghanian
University of North Dakota, Department of Electrical Engineering
jon.j.smith1980@gmail.com, sima.noghanian@engr.und.edu

Abstract

Adaptive antenna arrays are widely used in wireless communication and radar. The understanding of array design and optimization often needs sophisticated signal processing methods to estimate the Degree of Arrival (DoA). DoA estimation algorithms often require numerical analysis. We propose an educational tool that can be utilized for adaptive antenna array simulation and DoA estimation.

There were two major factors behind the motivation of this work. First, development of antenna arrays relies heavily on software packages focused on the individual antenna element design. Each software package that is used requires a license and may not match software in other institutions. Executing simulations and viewing their data within a common environment is needed. Second, executing DoA estimation algorithms is too cumbersome to be implemented within these simulation tools.

We propose a new program that treats array elements as single points and only utilizes their relative locations and orientations to ease calculation by algorithms. This program is written in MATLAB® as it is used by most institutions. The program utilizes a Graphical User Interface (GUI) to visualize the array developed. Then the program may also be utilized to call antenna simulation software, such as free simulation software, Numerical Electromagnetics Code (NEC) to obtain each antenna element radiation patterns.

We provide a modeling and simulation suite that users can develop new functions for its library under three categories: geometry-development functions, signal-generating functions, and DoA algorithm functions. Users may develop a shareable library of functions that can be run in a common environment.

Literature Review

The developed program can be used to design and analyze antenna arrays with different arrangements and in different environments. Ultimately, this program's purpose is to process the signals through DoA algorithms. Different DoA algorithms are available that can be developed as MATLAB® functions. However, at this point, only three algorithms have been developed: Delay-and-Sum, Capon and MUltiple SIgnal Classification (MUSIC). There are two major types of DoA estimation[2]: spectral-based and parametric methods. Parametric techniques require extensive searching to find an estimation of angle of arrival. Generally these methods are computationally complex, but very accurate. These methods will not be discussed further here.

Spectral-based methods look at the individual parameters and add the transformed or weighted results of antenna weighted inputs based on the parameter of interest's value. They give results of a spectral-like function (pattern). Spectral-based methods come in two forms: beamforming techniques and Sub-space based methods. In beamforming, the beam is "steered" to look through the entire range of values of the desired parameter by adjusting the weighting of sensors applied to incident signals. The most common beamforming method uses Bartlett beamformer and is called the Delay-and-Sum method. Essentially the weights are constrained to a complex magnitude of 1 and the phase is adjusted. The weighting essentially mathematically delays the signals and adds the results to maximize the output. This is done for all values of the angle parameter. Peaks within the spectrum are then considered estimates of the DoA. Delay-and-Sum is often used for uniform linear arrays (ULA), however, there is no requirement that this be the case. The algorithm is purely based on the displacement between sensors. The major drawback of this method is that it requires a large number of elements to achieve good resolution. It generally has low resolution and wide beamwidths. If two signals have DoA's with a beamwidthh of eachother, the method cannot separate them in the spectral function. The Capon beamformer overcomes this by minimizing the contribution of both signals and noise coming from directions other than the desired angle. However, some noise cancellation in the direction of arrival is sacrificed for "nulling" in directions of other signals. Capon, like Delay-and-Sum, depends on array geometry, as well as Signal to Noise Ratio (SNR). The Capon method is also known as the Minimum Variance Distortionless Response (MVDR). It, however, has the major drawback of significant signal cancellation due to coherence of multiple signals present at the sensors. The other technique aside from beamforming is subspace-based methods. One of the most popular of these methods is MUSIC. MUSIC has greater resolution than Capon and has the ability to find multiple signals present without cancellation[2-7].

Problem Statement

Today, with so much advancement in computer technology resulting in much faster data processing, engineers more often perform primarily simulation work before any actual "pen and paper" design is ever even begun. Especially with fast processor speeds and ability to divide processing jobs among multiple processors, many simulations are just as easily run on an engineer's desktop PC or laptop as can be run on a dedicated mainframe with limited availability. This is particularly true for small simulation packages with reasonably small data sets. In other words, if the engineer wants to run a simulation on some small piece of data that won't take too long, then it is not worth his or her time to reserve a dedicated computer for limited time for such a small or simple simulation.

There are numerous simulation packages available for almost any type simulation in just about any discipline you desire (flow calculations, chemical reactions, states of matter, heat transfer, electrical circuit calculations, etc.). Many of these simulation packages come in suites. Many, however, are very specialized simulation programs for a specific aspect of design. Therefore, more specialized programs tend to be sold as standalone programs requiring purchase of an individual license. This can become very expensive.

In educational environment, this is a major concern. In many introductionary and advanced courses offered in the field of antenna and array design, as well as wireless communications,

students are required to deal with design and simulation tools. In the educational setups, there are two general drawbacks to most available simulation software packages:

1. The packages are independent programs that require separate licenses that must be purchased,
2. From institution to institution, a different program may be utilized to run the same simulation types leading to complications in sharing and analyzing data which results from differing design input formats and output data formats.

In the field of electromagnetics (EM) there are numerous simulation packages available that deal with the design and simulation of high and low frequency circuits and antennas, or the interaction of objects with electromagnetic waves and scattering of waves. Therefore, these interactions can be categorized into either "passive" such as reflection, refraction or absorption of waves, or "active" such as a radiation of antenna elements in a certain pattern based in the geometry and material composition of the antenna elements and the geometrical arrangement of multiple radiating elements when an array of elements is designed. Many of the EM packages will also consider the environment surrounding the object and how those environmental variables affect radiant energy propagation. In general, these simulation packages can be very complex and thus very expensive. A few examples of these software packages are Ansoft Designer, Ansoft HFSS, FEKO, CST Microwave Studio, Remcom® Wireless Insite®, Remcom® XFDTD® WIPL-D, etc.

In designing of antenna, antenna array design imposes problems that involve both EM simulation and signal processing. In array design most EM simulation tools have a drawback. Most of these programs focus on the design and simulation of the single elements and do not offer an efficient simulation tool to include the array geometry and array signal processing together in one package. If an engineer desires to develop a certain array pattern and array processing algorithm that work together to acquire incident signals when arriving from certain angles, the engineer would have to completely design the entire array (i.e. specify its geometry, material composition, surrounding material environment, EM environment, etc). The problem can be arranged by copying and repeating the antenna element into structure a number of times. For every change in the array arrangement or feed arrangement a separate simulation must be run and the simulation must account for every other point's contribution and the field quantity. Typically, when calculations are performed using these types of simulation packages, there are large numbers of spatial and time data points and the geometrical design of the elements has great deal of complexity. The program might not be able to handle the large number of unknowns, or it might take a long simulation run time to run each case of feed arrangement.

Ideally, in design of an antenna array system, the design consists of two phases

- Antenna element design
- Array geometry and signal processing algorithm design

In design of the individual elements, with each variation in the design, the engineers can run separate simulations to reach the ideal pattern desired. In this case, use of many of the existing programs is desirable since high resolution results are required in the element design. The single

antenna element design takes far less time. In the array design, on the other hand, there will be a large number of trial and error simulations to evaluate various array geometries and signal processing algorithms. Therefore, there is a need to integrate the antenna element simulation with a simple array simulation to reduce the amount of run time. The purpose of these simulations is to observe the basic functionality of arrays and algorithms.

Solution Statement

To overcome the drawbacks stated above, we worked on development of simulation tool based on MATLAB®. MATLAB® is a simulation software package that allows the user to write their own programs within the confines of the MATLAB® development environment or within a MATLAB® run-time environment, if installed. MATLAB® is a program utilized by most engineering and science institutions and, therefore, any program written within it can be used on any machine that holds a MATLAB® license. Therefore, there will be no need to purchase an additional license. Also, utilizing the same program within the same environment allows much easier sharing and analysis of data among institutions with the added advantage of users having the ability to develop their own programs within MATLAB® to perform additional analyses on the data.

There are, however, some disadvantages in running simulations within the MATLAB® environment. The most critical disadvantage is that the run time is much slower in the MATLAB® environment (in comparison to programs written in low level languages such as C++). This is due to the overhead involved in interpreting MATLAB® code for execution at runtime as compared to a compiled program that executes the code directly as machine code. This increased execution time, results in significantly longer times for the entire simulation run.

Although MATLAB® offers a complier that enables us to convert the code to an executable code, since each built-in function of MATLAB® is written for general purposes and is not optimized for the problem at hand, the run time in general is longer than custom written programs in low level languages.

Our goal was to develop a program that focuses on the development and processing of the array as compared to the detailed design of the individual elements that comprise the array. To keep the run time within a reasonable range simplicity is paramount in the design of the program.

Essentially our purpose of this program was to develop the array pattern (or array factor) as opposed to the combination of the element and array patterns.

Simulation Environment Model

As mentioned in the introduction, the major drawback of executing array simulations in most antenna modeling software packages is that it focuses on mainly design details and high resolution data (i.e. large number of data points) that require intensive data processing to calculate field quantities.

In the program each antenna element is simplified to a single, dimensionless point in the simulation space (what is known as point source). A point source is an isotropic receiver or radiator. When it acts as a receiver, the induced current on a single element is the same regardless of the direction of arrival of the incident waveform. The induced current will only vary based on the waveform properties, like amplitude, frequency, etc. When the element acts as a radiator, the transmitted waveform will be transmitted equally in all direction away from the element. Only the waveform properties like amplitude and frequency will vary with equivalent variations in the excitation current signal[3,4].

In more complex, multiple data point elements, induced currents must be calculated from incident electromagnetic waves; and transmitted electromagnetic waves must be calculated from excitation current. Although the array elements are treated as isotropic, the user will have the option to include the polarization of the antenna, e.g. for a linear antenna signals arriving from directions perpendicular to the orientation of the element will have zero reception and signals arriving from directions parallel to the orientation of the element will have 100% reception of the signal. The relationship of the reception is based on the difference between the angle of arrival of the incident signal and the orientation of the element. A simple cosine of the angular difference describes this relationship[3].

Program Assumptions

In this program propagation is only accounted for at the points in time and at locations in space where the signal is present. Therefore, if a signal is transmitted from a distant source to the array, there is a time lapse from the time that signal is transmitted to the time it arrives at the array. The program calculates a time delay and then, after that delay, performs necessary calculation and signal processing.

Additionally, the program treats the environment as homogenous. The MATLAB® program focuses purely on geometrical configuration of the array and performs its analysis in an ideal uniform environment. The only environmental variable the user defines is the speed of propagation which can be set as any value desired by the user. It should also be pointed out that no considerations are made with regards to material composition of array elements either.

Far -Field

Like in most antenna design, there is always an assumption that receivers are always in the far-field of the transmitter. The Near-field/Far-field boundary is based on the wavelength (and thus frequency and speed of propagation). The program will not alert the user if their receiver is or is not in the far-field, therefore, the user must be aware of this assumption. The modeled waveforms that propagate in the simulation space of this program are far-field waveforms only. Therefore, array patterns will vary in both the zenith and azimuth angles but will not vary with radial distance from the array, provided that radial distance is in the far-field. Therefore, in transmitter mode, when setting the radius for pattern development, user has to make sure that pattern is in the far-field[3,4].

Plane Wave

A spatial waveform at a sufficient distance from its source can be approximated as a plane wave. Typically, in the far-field a plane wave can be assumed. DoA algorithms are based on plane wave reception by elements. Modeling of a propagating plane wave, as opposed to a spherical wave that resembles a plane wave due to travel, is much simpler to model and remove unnecessary complexity to program execution[3,4].

Constant Amplitude

As an incident plane wave passes over the array's elements in receiver mode, the amplitudes within the signal do not vary, or more explicitly do not decrease with the distance from the source. The assumption here is that the largest dimension across the array is extremely small compared to the distance from the source to the array, and therefore, the change in amplitude is relatively insignificant.

Overall Program Operation

This MATLAB® program's primary purpose is to develop various array geometries and test those geometries ability to effectively transmit or receive electromagnetic signals using an assortment of different algorithms. In order to accomplish this, the program performs three major tasks: (1) develop an array, (2) define a signal set and associated signal environment, and (3) select and apply an appropriate algorithm to process the defined signals. Each of these tasks has a separate associated Graphical User Interface (GUI) window to accomplish the task and all are completed in the order given above. The overall program operates in one of two modes: **transmitter mode** or **receiver mode**, as are discussed in greater detail below. Regardless of the mode of operation, all three major tasks defined above are applicable and must be completed for a successful simulation.

Array Development

The first task in completing a successful simulation is to build an array. As mentioned above, each of the three main tasks of this program have their own GUI window. The array development GUI is the only of the three windows that has absolutely no dependence on the mode of operation and therefore is exactly the same in either mode. In fact, the mode of operation is not even specified until the signal development GUI. Development of an array under this screen is a fairly simple process. The screen is divided into three columns (Fig. 1).

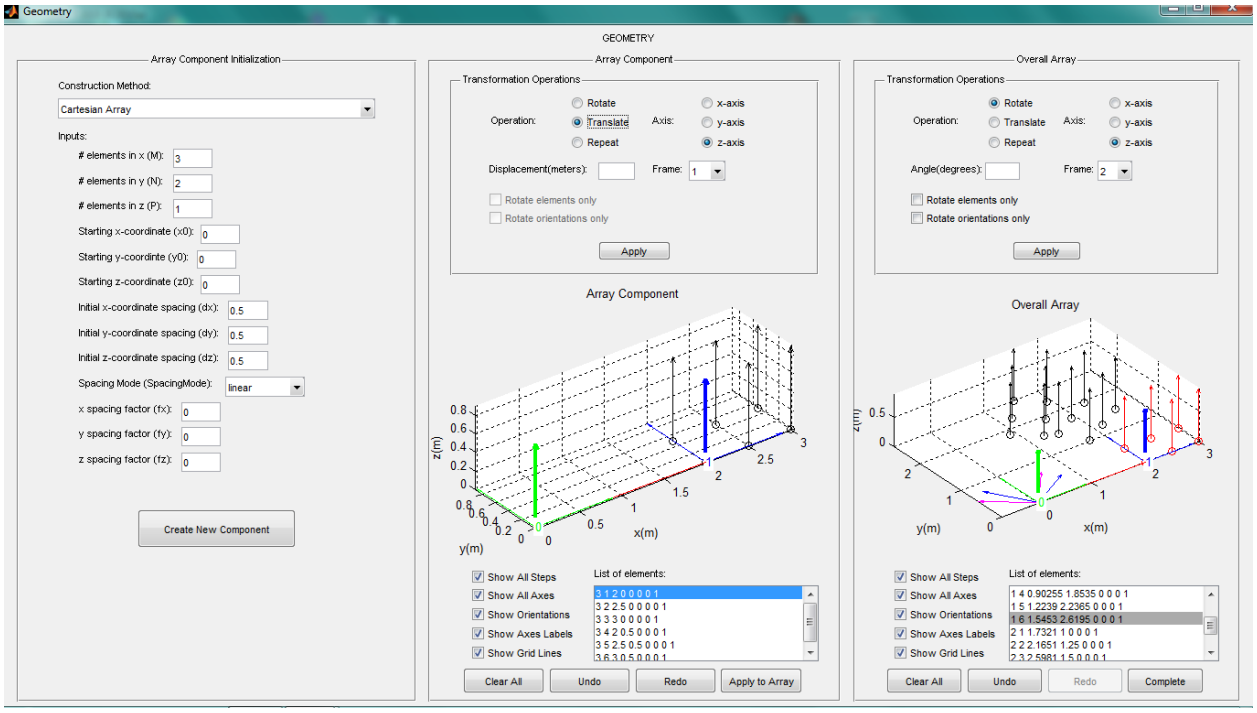

Fig. 1. Geometry screen used for designing array geometries

The leftmost column is where the user calls one of the array development functions from the array library. These functions can create any type of array the user likes. There are a limited number of array types in the library. However, a user can develop his or her own function. Any array generated in this leftmost column is displayed and available for modification in the center column of the GUI. The center column is for development of the array components. An array component is a piece of the overall array developed on its own and then applied to the overall array, in the rightmost column. An array component can be used to generate the overall array as soon as it is created or it can be modified through rotations and translations to achieve the desired array orientation. A repeat function is also planned to be available, to repeat the same array pattern multiple times (The repeat function is currently unavailable). These transformations are applied separately to the array component until it is ready to be applied to the overall array. Additionally, the rotate and translate functions can be applied to any previous frame of reference to which the array component was referenced. Each time a translation or rotation is applied to the array component, the current frame of reference also rotates or translates, as well. The user can also control what attributes of the array are displayed, such as references axes and axis labeling, transformation path, grid display and orientation of the individual elements. Note element orientations are useful when elements are treated as non-isotropic sources. All transformation operations that are conducted on an array component can be undone or redone utilizing pushbuttons of the same names. An array component can also be completely cleared out using the clear pushbutton. The component can also be overwritten by simply creating a new component in the leftmost column. Finally, once the user is ready to apply the array component to the overall array, the “Apply to Array” pushbutton can be pressed.

As mentioned above, the array component is used to generate an array which is displayed and modified in the rightmost column. Each array component will have its own associated component IDentification (ID) number along with unique numbers for the individual elements of that component. Each time a new component is added to the overall array, the component ID of the next component array will be increased by one. In the center column, "List of Elements" contains all of the elements of the currently displayed array component (sub-array). The first number of any entry is the component ID and the second number is the element ID. The third, fourth and fifth numbers are x, y and z locations of the element in the simulation space. The sixth, seventh and eighth values are unit vectors describing the spatial orientation along the absolute x, y and z axes of the element. Note that all elements in this box should have the same component ID and different element IDs.

As stated before, the rightmost column is the location of the overall array which may contain multiple component arrays (sub-arrays). This is the array that will ultimately be utilized in the simulation. The layout of this column is almost identical to the center column for the array component. This column displays the overall array and provides the same transformation operations to the user as the center column. The user has the same controls over what attributes of the overall array are displayed. In addition, the overall array plot also displays the current component array on top of the overall array. The overall array is displayed in black and the component array is red. In order to remove the component array from the overall array plot, the component array must be cleared in the center column which removes it from both displays. Like the component array, the user can undo or redo any operations performed on the overall array, like rotations and translations, but also application of a new component to the array which is performed in the center column. In fact, anytime an undo or redo operation removes or adds a component, the component ID value for the next component array is decreased or increased by one. The clear option also exists for the overall array. Once you click the "Clear All" pushbutton, the action cannot be undone. One difference from the component array, however, is that the overall array has a "Complete" button instead of an "Apply to Array" button. Pressing this button signifies that the user has completed the creation of his or her array and the program now moves to the signal GUI. One other difference between the component array and the overall array windows is that in the List of Elements in the overall array window all elements of all components are listed. Each component's elements have their own component ID. Multiple elements in the list may have the same element ID; however, no two elements will have the same combination of component ID and element ID.

Signal Development

After completion of the array design, the user must now specify under what mode the array will operate and specify the signal that will either be transmitted or received by the array. The signal development GUI is also divided into three columns (Fig. 2).

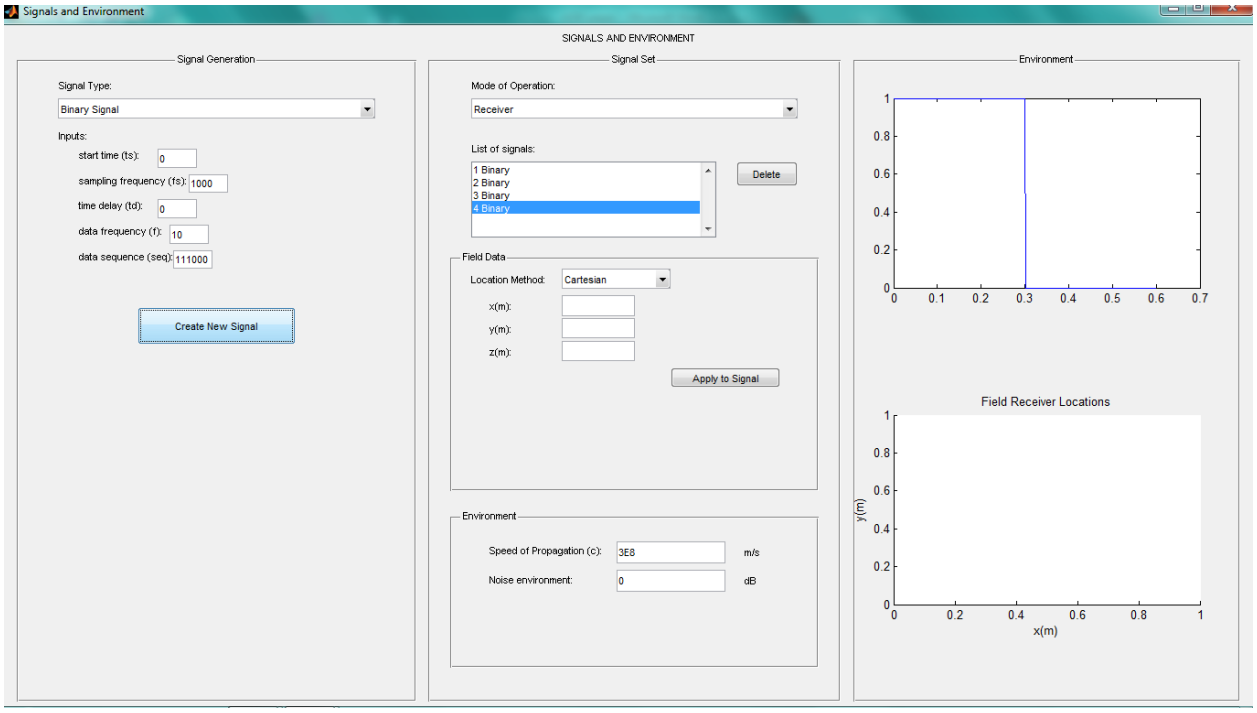

Fig. 2. Signals and Environment screen used to specify signals and their locations

The first column, like the array development GUI, is utilized to select and generate the signal type. Once a signal's input parameters are specified, that signal is added to the list of signals. A user can add as many signals of different type as needed. Next, the user must specify under which mode of operation he or she desires to perform the simulation, **receiver mode** or **transmitter mode**. The modes of operation are described further below. The Field Data sub-panel will change based on the selected mode of operation. Under receiver mode, an "Apply to Signal" button appears that allows the user to associate a location in the simulation space with a signal (as shown in Fig. 2). That location and signal number are then displayed in the rightmost column. The user can also delete already created signals from the list, if desired, using the "Delete" button. Also, if the user selects/highlights a signal in the "List of Signals", that signals time-based waveform will be displayed in the rightmost column. The Environment sub panel specifies only two major attributes of the overall environment, speed of signal propagation and noise. The rightmost column displays the selected signal in the top axes and the map of the signal source locations in the bottom axes.

Signal Processing / Algorithms

The final screen is for signal processing of received signals or pattern development based on use of user selected algorithms. The functions for the algorithms are selected from a library just like for arrays and signals. However, in this case, algorithms for DoA (receiver mode) are different than algorithms for pattern development (transmitter mode). Once the user selects the appropriate algorithm, the simulation can be run. At this point, a screen is in development to display simulation results. Currently, it is up to the user to format the results for display.

Modes of Operation

The MATLAB® program essentially performs only one of two basic analyses defined by a mode of operation:

1. Transmitter Mode: The user-defined array transmits signal(s) that are then measured at user defined points at given field locations in order to develop a field pattern. In this mode, the measurement locations are located in the far-field of the array and its transmission elements, typically a constant distance from the array and uniformly spaced angles around the array to develop a field pattern.
2. Receiver Mode: The user-defined array receives signals transmitted from radiating sources located at user-selected locations such that the array is in the far-field of the radiating sources. In this mode, the array elements are themselves measurement devices or sensors that intake data from the incident electromagnetic wave. This data is then processed by an algorithm to determine a direction of arrival of the pattern.

For the two main modes of operation, array development is exactly the same whether designing an array for receiver or transmitter functionality. Therefore, regardless of the mode of operation the user chooses, the user will design an array under the same exact GUI. Under both modes of operation, the user must also specify a signal or group of signals and an appropriate algorithm or method of signal reception to process transmitted or received signals. However, these two aspects of the program operate differently for the two modes of operation.

Signal development, unlike array development, differs depending on the mode of operation. Under both modes of operation, a single signal or multiple signals can be specified. The signals can have any characteristics the user defines according the existing signal function library or a signal function created by the user. Since the array elements are isotropic and do not possess any attributes specific to an antenna type, the user can transmit signals of any frequency, amplitude, sequence, etc. Signals can overlap in time, be widely separated in time, and possess differing pulse shapes, modulation schemes, different frequencies and wide-varying bandwidths. Essentially, these transmitters are ideal and have no frequency response or any real-world physical constraints on them. This allows the program to focus completely on array processing. This program takes a rather simple approach to defining the simulation environment's parameters. The program treats the entire environment as materially homogenous; therefore, all signals propagate through the environment equally (i.e. same speed of propagation). In either mode of operation, the user defines the speed of propagation for the entire environment. Where the above aspects of signal development are equivalent for both modes, the modes differ significantly with regards to location of receivers and transmitters. In transmitter mode, all signals originate from one location, the array. The array is a transmitter and already has its position in simulation space defined. However, the receivers placed in the array's far field have their locations defined within the signal development GUI. This data is prompted by the GUI and not the signal function. In receiver mode, signals can originate from many locations, except from the array. The array is the only receiver; and like in transmitter mode, its location is already defined. The signal sources which are transmitters are located as specified by the user such that the array is located in their far field. Each signal given can have its own location or multiple signals can be specified at the same location. This is completely at the user's discretion. The

prompting of location data for the transmitters is a function of the signal GUI based on the selected mode of operation, just like in transmitter mode. Therefore, in either modes of operation, a signal function from the library (existing or user-built) can be used to define a signal to be transmitted, whether from a field source or the array itself. However, the defining of transmitter or receiver location is different between the two modes of operation. Another way to look at it is that in transmitter mode user-defined signals are transmitted by the array, and therefore, have no data location associated with them (i.e. no defining data location for signals). The receivers defined in the far field do have data location assigned to them. In receiver mode, the user-defined signals are transmitted by transmitters (i.e. signal sources) for which the array is in their far field and each individual signal defined is assigned location data. In this case the array is the receiver and has no signals associated with it. Therefore, both modes have transmitter(s) and receiver(s), both modes define location "in the far-field" at a distance from the array, whether in the array's far field or vice-versa, and in both modes the array's location is always given. Specification of the signal environment (i.e. noise environment) also varies between the modes of operation. In this program, noise is defined in one of two ways: (1) noise added to a baseband signal prior modulation and transmission at and (2) noise added to a received signal at a receiver (i.e. additive noise to the already modulated and transmitted signal as received). The first type of noise is defined by a given signal's function from the signal function library, if that particular function even defines noise. That particular type of noise is defined (when it is defined) identically in either modes of operation. It is the latter type of noise that differs between the two modes of operation. Regardless of the mode of operation, noise type (2) affects receivers. Therefore, in transmitter mode, the noise environment is defined at the receivers defined in the far-field of the receiver. The noise is defined, however, for the entire environment, not for the individual receivers. This will likely be changed at a later time to allow defining a noise environment at each individual receiver. Recall that these "receivers" are typically defined at a constant radius and at equally spaced angles in order to develop an array pattern. Therefore, it may be desirable to define no noise at all, if one's purpose is pure pattern development. In receiver mode, the array itself is the only receiver and there would only be a single noise environment definition anyway. Typically, in this mode noise should more often be considered to test an array and its DoA algorithm's ability to locate a signal in a noisy environment.

Signal processing of array inputs also differs between modes of operation. How the signals at the array elements, transmitted or received, are manipulated in order to optimize transmission or reception of the signals of interest is controlled by similar processes. Both involve control and tuning of signal parameters such as phase angle, frequency, and amplitude to direct or steer a beam that transmits or "listens" for signals. However, the processes are algorithmically different depending on the mode of operation and require separate libraries of separate functions, unlike the previous GUI's for array and signal functions. In transmitter mode, only transmission algorithms will be available for selection from the menu. In receiver mode, only DoA algorithms will be available for selection from the menu. Like with the array development GUI and the signal development GUI, selection of a specific algorithm from the menu will prompt for associated data for that algorithm. Unlike, the GUI above, this GUI does not have any GUI-specific prompts, it only prompts for data based on the selected algorithm from the library.

DoA Algorithms

Although this program can operate in two modes in which the array acts as either a transmitter or as a receiver, it is the latter that was the original intention of the program. A simulation environment was developed in which signals propagated from distant sources to arrive at the array at which time the array's DoA algorithm would then process those signals. The capability still exists to implement the array as a transmitter to develop an array pattern, however, that function is secondary to the DoA processing performed in receiver mode. In fact, no processing algorithms have yet been developed for transmitter mode.

The origin of this MATLAB® program was to take inputs from NEC and then process those inputs using the signal processing algorithms of the MATLAB® program's library. NEC, like the MATLAB® program, specifies elements which are then excited by either applying a current signal to the element to act like a transmitter; or by an incident EM signal exciting the element to act like a receiver. In the case in which the element is excited by the incident wave, the MATLAB® program would take the current values induced at the elements and process those signals in the MATLAB® program to determine the signal's DoA. It is still the intention of the MATLAB® program to perform this function.

The MATLAB® program currently has three known DoA algorithms: Delay and Sum, Capon and MUSIC. Delay and Sum can be implemented using a single snapshot of time, however, it will have low resolution. Capon and MUSIC require much more data to create highly accurate results, especially in a multiple signal environment. This is why the simulation environment was developed to allow for multiple signal sources to propagate over the array over a user-defined time range.

Conclusions and Future Work

In this paper the functionalities of a MATLAB® based array simulator was discussed. This simulator currently uses isotropic sources and is able to characterize any user defined array configuration and input signals. The capability to import antenna element information form an EM simulation software such as NEC is left as a future work.

This software is in the final stages of preparation and the authors are planning to introduce it in different related courses offered in the Electrical Engineering Department of University of North Dakota, such as "Introduction to Antenna Systems", "Antenna Theory" and "Phased Array Antennas". In the previous years students in these classes either were using commercial software for simple array design, or used MATLAB® for developing simple DoA programs. Having this complete program available with the source code now provides the instructors the tools to design more sophisticated assignments and projects for better understanding of realistic array antennas used in wireless systems.

Acknowledgments

Authors would like to acknowledge the financial support of North Dakota NASA EPSCoR (National Aeronautics and Space Administration Experimental Program to Stimulate

Competitive Research) and the support of University of North Dakota Electrical Engineering Department.

Bibliography

1. G.J. Burke, and A.J. Poggio, *Numerical Electromagnetics Code (NEC) – Method of Moments*, Lawrence Livermore Laboratory, Livermore, CA, January 1981
2. H. Krim, and M. Viberg, "Two decades of array signal processing research: the parametric approach," *IEEE Signal Processing Magazine*, vol.13, no. 4, pp. 67-94, July 1996.
3. C. A. Balanis, *Antenna Theory: Analysis and Design, 3rd Ed.* John Wiley and Sons, Inc. Hoboken, NJ, 2005.
4. C. A. Balanis, and Panayiotis Ioannids, *Introduction to Smart Antennas*, Morgan and Claypool, 2007
5. J. Capon, "High-resolution frequency-wavenumber spectrum analysis," *Proceedings of the IEEE*, vol. 57, no. 8, pp. 1408-1418, Aug. 1969.
6. R. G. Lorenz, S. P. Boyd, "Robust minimum variance beamforming," *Proceedings of the IEEE*, vol. 53, no. 5, pp. 1684-1696, May 2005.
7. P. Hacker, B. Yang, "Single snapshot DOA estimation," *Advances in Radio Science*, vol. 8, pp. 251-256, 2010.