Adaptive Antenna Array Using the RLS Algorithm

1st Jacob Kirschner
Department of Electrical Engineering
The Colorado School of Mines
Golden, Colorado, United States
jkirschner@mymail.mines.edu

2nd Mason Wilie
Department of Electrical Engineering
The Colorado School of Mines
Golden, Colorado, United States
wilie@mymail.mines.edu

***Abstract*—The Recursive Least Squares is an estimation algorithm that can be used to implement an adaptive antenna array, capable of nulling unwanted interference signals while leaving desired signals unchanged. This allows for a more robust and capable system that is applicable to both the power sensitive, extremely high frequency communications systems as well as increasing the safety and effectiveness of cancer treatment through hyperthermia.**

***Index Terms*—Recursive Least Squares, antenna arrays, adaptive filters, machine learning, 5G, DSP**

I. Introduction

A Singular antenna element has a fixed radiation pattern which is described by physical properties such as length and shape. However, the use of multiple antennas allows for the ability to change the radiation pattern by manipulating the signal that is fed into or received by each element. This resulting change is known as the array factor, providing the ability for things such as beam steering or null unwanted interference and is the basis behind antenna arrays.

An Adaptive or smart array is a form of digital finite impulse response (FIR) filter with weights at each antenna element that change the amplitude and phase of the incoming or outgoing signal. These weights can be dynamically updated to produce a desired array factor in changing conditions and are calculated using the received signal, known desired signal, and an estimation algorithm, in this case the Recursive Least Squares (RLS).

II. The Recursive Least Squares Algorithm

The recursive least squares algorithm (RLS) was first discovered by Gauss, rediscovered by Plackett in 1950, and finally applied in 1960 by Kalman with the growing popularity and use for control theory [1]. It is based on the well known Least Squares (LS) algorithm, both attempting to minimize the sum of the squared error between sampled data and estimation curve, though the RLS algorithm diverges much faster, at the cost of being much more computationally complicated [2].

A. Derivation

The block diagram for the RLS algorithm is shown in figure 1, with $f(n)$ representing the received signal, $H(z)$ the transfer function of the system, $y(n)$ the output, $d(n)$ the desired output, $e(n)$ the error, and $b(n)$ the updated filter coefficients.

The goal of the RLS algorithm is to minimize the squared error, $\varepsilon(n)$ between the desired signal $d(n)$ and the output signal $y(n)$ for each iteration n, shown in equation 1. λ represents the *forgetting factor*, which is usually chosen to be $0.98 < \lambda < 1$ and determines how much weight is given to past data points. If $y(n)$ is described by equation 2, where b_k represents the k^{th} filter weight and $f(n-k)$ represents the k^{th} input signal, a new equation for the sum of the squared error is produced, equation 3 with expanded form shown in equation 4.

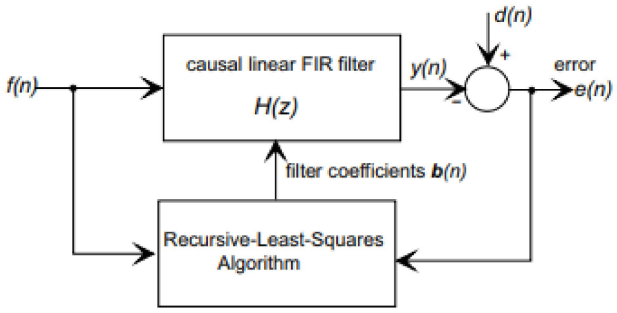

Fig. 1. The block diagram representation of the RLS algorithm [3].

$$\varepsilon(n) = \sum_{i=0}^{n} \lambda^{n-i}(d(i) - y(i))^2 \tag{1}$$

$$y(n) = \sum_{k=0}^{M-1} b_k f(n-k) \tag{2}$$

$$\varepsilon(n) = \sum_{i=0}^{n} \lambda^{n-i}\left(d(i) - \sum_{k=0}^{M-1} b_k f(i-k)\right)^2 \tag{3}$$

$$\begin{aligned}\varepsilon(n) &= \sum_{i=0}^{n} \lambda^{n-i} d^2(i) \\ &+ \sum_{i=0}^{n} \lambda^{n-i} \sum_{k=0}^{M-1}\sum_{m=0}^{M-1} b_k b_m f(n-k) f(n-m) \\ &- 2\sum_{i=0}^{n} \lambda^{n-i} \sum_{k=0}^{M-1} b_k f(n-k) d(i)\end{aligned} \tag{4}$$

To minimize equation 4, the derivative of $\varepsilon(n)$ is taken with respect to the filter weights and set to zero. This produces equation 5, with matrix representation shown in equation 6.

$$\sum_{i=0}^{n} \lambda^{n-i} \sum_{k=0}^{M-1} \sum_{m=0}^{M-1} b_k f(n-k) f(n-m) = \sum_{i=0}^{n} \lambda^{n-i} \sum_{k=0}^{M-1} f(n-k) d(i) \tag{5}$$

$$\mathbf{R}(n)\mathbf{b}(n) = \mathbf{P}(n) \tag{6}$$

To solve equation 6 for the filter weights $\mathbf{b}(n)$, $\mathbf{R}^{-1}(n)$ is multiplied on both sides to produce equation 7, where $\mathbf{b}(n)$ is a column vector of filter weights, $\mathbf{R}(n)$ and $\mathbf{P}(n)$ are described by equations 8 and 9 respectively, $\mathbf{f}(n)$ is a column vector of system outputs, and $d(n)$ is the desired output.

$$\mathbf{b}(n) = \mathbf{R}^{-1}(n)\mathbf{P}(n) \tag{7}$$

$$\mathbf{R}(n) = \sum_{i=0}^{n} \lambda^{n-i} \mathbf{f}(i)\mathbf{f}^T(i) \tag{8}$$

$$\mathbf{P}(n) = \sum_{i=0}^{n} \lambda^{n-i} \mathbf{f}(i) d(i) \tag{9}$$

B. Generating an Implementable Form

The RLS algorithm does not solve for $\mathbf{b}(n)$ in one step, but through an iterative process of incrementally updating the $\mathbf{R}(n)$ matrix. This process is shown in equation 10, where $\mathbf{R}(n-1)$ is the previous $\mathbf{R}(n)$ matrix.

$$\mathbf{R}(n) = \lambda \mathbf{R}(n-1) + \mathbf{f}(n)\mathbf{f}^T(n) \tag{10}$$

Notice that equation 7 depends on $\mathbf{R}^{-1}(n)$ not $\mathbf{R}(n)$, therefore an equation for $\mathbf{R}^{-1}(n)$ in terms of $\mathbf{R}^{-1}(n-1)$ is necessary as taking the inverse of $\mathbf{R}(n)$ each iteration is computationally intensive and would be entirely too slow. This is accompished using the *Matrix Inversion Lemma*, which states that "if a square non-singular nxn matrix $\mathbf{A}$, with known inverse $\mathbf{A}^{-1}$, is updated with an additive term the new inverse is given by" [3] equation 11.

$$\begin{aligned} &\left(\mathbf{A} + \mathbf{BCB}^T\right)^{-1} \\ =&\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{B}\left(\mathbf{C}^{-1} + \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}\right)^{-1}\mathbf{B}^T\mathbf{A}^{-1} \end{aligned} \tag{11}$$

By setting $\mathbf{A} = \mathbf{R}^{-1}(n-1)$, $\mathbf{B} = \mathbf{f}(n)$, and $\mathbf{C} = \lambda^{-1}$, equation 12 is generated.

$$\mathbf{R}^{-1}(n) = \lambda^{-1}\left(\mathbf{R}^{-1}(n-1) - \frac{\mathbf{R}^{-1}(n-1)\mathbf{f}(n)\mathbf{f}^T(n)\mathbf{R}^{-1}(n-1)}{\lambda + \mathbf{f}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{f}(n)}\right) \tag{12}$$

We can define a vector of Kalman gains $\mathbf{k}(n)$ as equation 13 in order to simplify equation 12. The simplified version is shown in equation 14.

$$\mathbf{k}(n) = \frac{\mathbf{R}^{-1}(n-1)\mathbf{f}(n)}{\lambda + \mathbf{f}^T(n)\mathbf{R}^{-1}(n-1)\mathbf{f}(n)} \tag{13}$$

$$\mathbf{R}^{-1}(n) = \lambda^{-1}\left[\mathbf{R}^{-1}(n-1) - \mathbf{k}(n)\mathbf{f}^T\mathbf{R}^{-1}(n-1)\right] \tag{14}$$

Simmilarly, $\mathbf{P}(n)$ can be defined as the incrementally updating equation 15.

$$\mathbf{P}(n) = \lambda \mathbf{P}(n-1) + d(n)\mathbf{f}(n) \tag{15}$$

With these new definitions for $\mathbf{R}^{-1}(n)$ and $\mathbf{P}(n)$, equation 7 can be rewritten for an iterative calculation of $\mathbf{b}(n)$ as equation 16, where $e(n)$ represents the error between the output signal $y(n)$ and the desired signal $d(n)$.

$$\mathbf{b}(n) = \mathbf{b}(n-1) + \mathbf{k}(n)e(n) \tag{16}$$

In summary, actual implementation of the RLS algorithm requires equations 13, 14, and 16 to be calculated with each iteration.

Equations in this section largely originate from reference [3].

C. Choosing a Forgetting Factor

The forgetting factor, λ decides how much weight is put on previous iterations and is one of the things that sets the Recursive Least Squares algorithm apart from the Least Squares algorithm, therefore it is important to choose a reasonable value. Figure 2 plots the normalized magnitude of the array factor found using the RLS algorithm at the desired angle in respect to the lambda value chosen, where the desired value is 0 dB. It is clear that the smaller lambda is, the more variable and unreliable the system is. The graph shows that the range of $0.8 < \lambda < 1$ is the most consistently successful range, which is why lambda is usually chosen to fall within it.

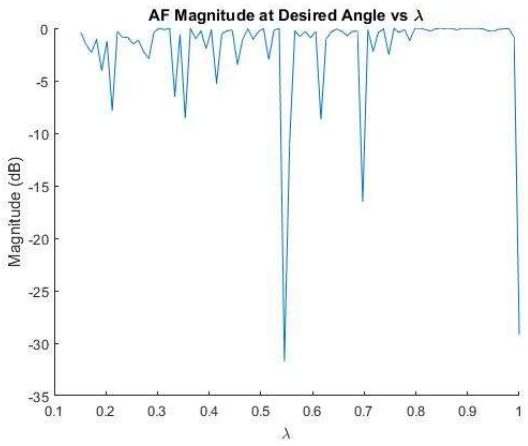

Fig. 2. Magnitude of the array factor at the desired angle in respect to the forgetting factor λ.

III. Application of RLS to Antenna Arrays

Given an antenna array with N elements, a desired signal S incident at angle θ_d and two interference signals incident at angles θ_{I1} and θ_{I2}, the RLS algorithm can be used to null the interference signals, while leaving the desired signal unaffected.

A. Correlation Matrix

To initialize the algorithm, a value for the first instance of $\mathbf{R}(n)$ must be chosen. The best choice is a matrix populated by the correlation values between the antenna elements, which in this case is equivalent to the covariance. Equation 17 shows the calculation of athe covariance between two elements, where $Cov(X,Y)$ represents the covariance between X and Y, X_i the value of the i^{th} element of the data set X, $\overline{X}$ the mean of the data set X, Y_i the i^{th} element of the data set Y, $\overline{Y}$ the mean of the data set Y, and M the number of values in each data set [4].

$$Cov(X,Y) = \frac{1}{M-1}\sum_{i=1}^{M}(X_i - \overline{X})(Y_i - \overline{Y}) \tag{17}$$

In the case of an antenna array, the data sets X and Y represent the received signals at two antenna elements. Therefore, if the antenna array contains N elements, we can define the N×N correlation matrix of the array to be,

$$\begin{bmatrix} C_{11} & C_{12} & \cdots & C_{1N} \\ C_{21} & C_{22} & \cdots & C_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ C_{N1} & C_{N2} & \cdots & C_{NN} \end{bmatrix}$$

where C_{ij} represents the correlation between the i^{th} and j^{th} elements of the array. This method requires a sufficient number of samples at each array element to be known. In Matlab, the covariance of a data set can be calculated using *cov(X)*, where *X* represents the N×M vector of samples at each antenna element.

B. Running the Algorithm

Once the initial value for $\mathbf{R}^{-1}(n)$ is known, the first iteration of the algorithm can occur. The first steps of implementation are to calculate the error $e(n)$ between the received signal $y(n)$ and the desired signal $d(n)$, shown in equation 18, and to calculate the Kalman gains $\mathbf{k}(n)$ shown in equation 13, where $\mathbf{f}(n)$ represents the un-weighted signal from each array element. Because these two equations are independent of each other, they can be computed in parallel.

$$e(n) = d(n) - y(n) \tag{18}$$

Once these values are found, $\mathbf{R}(n)$ and the new filter weights $\mathbf{b}(n)$ can be found using equations 14 and 16 respectively. Again, these equations are independent of each other and therefore can be calculated simultaneously.

Because it would be impractical and computationally intensive to use all samples that each element receives, sampling of $y(n)$ is required. Choosing a sampling method requires the engineer to decide between speed of the algorithm and accuracy, as more samples would create a more accurate system while increasing computational time.

C. Calculating the Array Factor

When the final element weights are calculated, the array factor can be found. For a linear antenna array oriented along the y-axis, equation 19 can be used where $b(n)$ represents the complex weight on the n^{th} element of the array, d the spacing between the elements in terms of the wavelength, n the element number, and θ the angle. The vector form of equation 19 is shown in equation 20, where **w** is an N×1 column vector of element weights, **n** is an N×1 column vector of element numbers $[0, 1, ..., N-1]^T$, and $\boldsymbol{\theta}$ is a 1×Q row vector where Q represents the number of samples of the array factor **AF**, which itself is a 1×Q row vector.

$$AF = \sum_{n=0}^{N-1} b(n)e^{-j2\pi dn*sin(\theta)} \tag{19}$$

$$\mathbf{AF} = \mathbf{w}^T e^{-j2\pi d\mathbf{n}*sin(\boldsymbol{\theta})} \tag{20}$$

IV. Simulation

In order to test the algorithm, a Matlab simulation of an antenna array was needed as access to an antenna array was not possible at the time of testing.

A. Setup

To model the scenario, one desired and two interference signals were created. Arbitrarily, the desired signal was chosen to be a sine wave with an amplitude of 1, shown in figure 3, and the interference signals chosen to be random numbers between -5 and 5 using the Matlab function rand(), shown in figure 4.

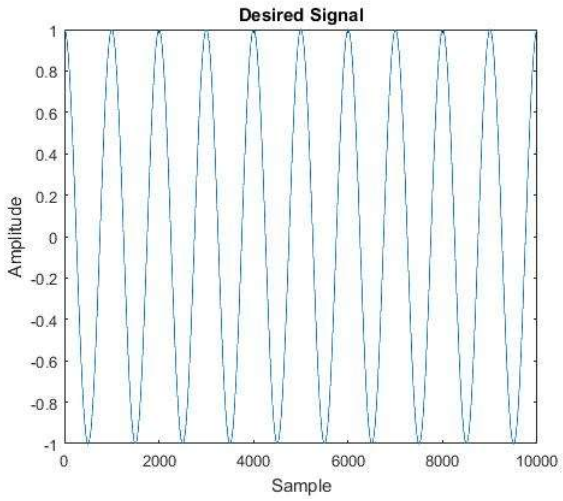

Fig. 3. Plot of the signal which is desired to recover.

In order to give the signals an incident angle, a steering vector was needed for each one. This steering vector is shown in equation 21, where **a** is the N×1 steering vector, d is the distance between elements in terms of the wavelength, **n** is an N×1 column vector of element numbers $[0, 1, ..., N-1]^T$, and θ is the incident angle of the signal [5]. The signals

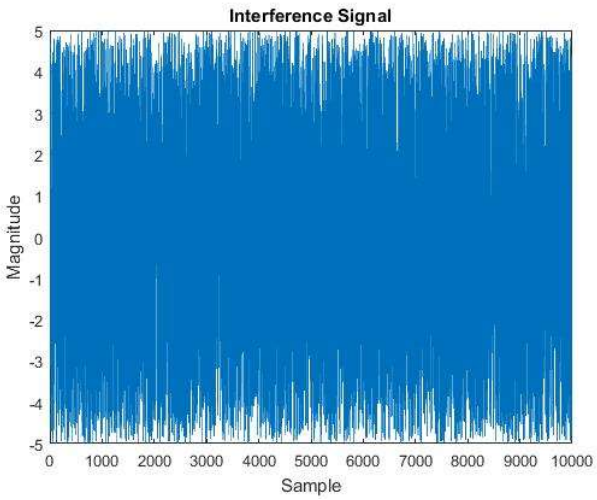

Fig. 4. Undesired signal which will be incident at a different angle than the desired signal.

are then multiplied by their corresponding steering vectors as well as summed at each individual antenna element, in addition some Gaussian white noise is added to each element. This produces N different signals, the recieved signals at each antenna element. The total recieved signal which the antanna array must recover the signal shown in figure 3 from is shown in figure 5

$$\mathbf{a} = e^{-j2\pi d\mathbf{n} * sin(\theta)} \tag{21}$$

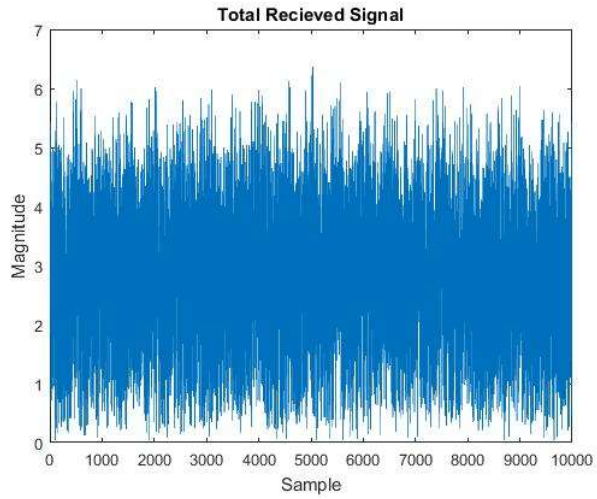

Fig. 5. Total signal seen by the antenna array.

B. Results

After running the simulated recieved signals through the Recursive Least Squares algorithm, the desired signal was successfully recovered. The recovered signal, displayed in figure 6 is almost indistinguishable from the original signal, and therefore it can be concluded that the RLS algorithm successfully calculated the correct element weights needed to null the interference signals while keeping the desired signal intact. The error between the received signal $y(n)$ and the desired signal $d(n)$ at each iteration is shown in figure 7 and shows how quickly the RLS algorithm diverges.

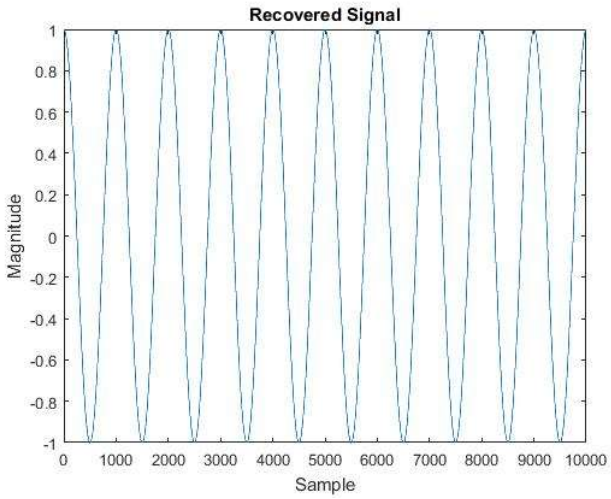

Fig. 6. Signal recovered from the received signal using the weights calculated by the RLS algorithm.

Using the final calculated element weights, an array factor can be found using equation 20. Represented in figure 8, the array factor shows what directions the antenna array will receive the signal at and clearly displays that the desired signal is received and the interference signals are nulled.

C. Translating to Radiation Pattern

The array factor only conveys information about how the geometry of the antenna array will effect the overall radiation pattern. To find the far-field radiation pattern of the array, one must multiply the array factor by the radiation pattern of a

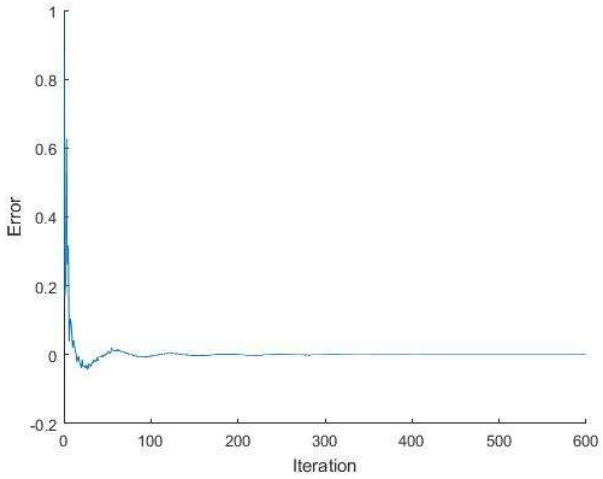

Fig. 7. Error between received and desired signal at each iteration of the RLS algorithm.

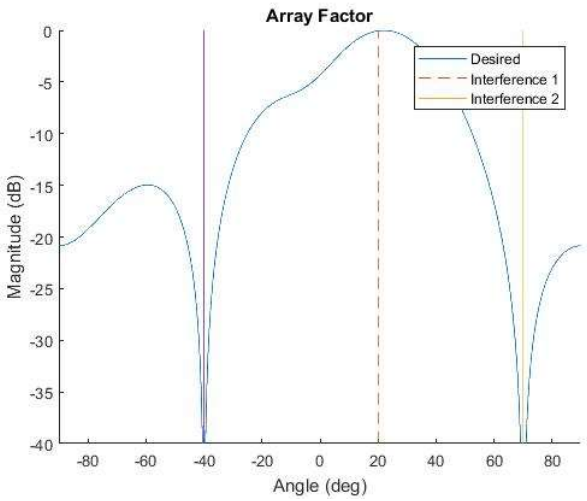

Fig. 8. Array factor using the weights found from the RLS algorithm.

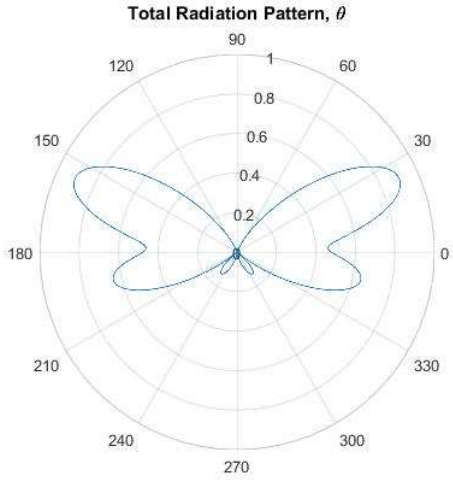

Fig. 10. Total radiation pattern of the array with weights calculated by the RLS algorithm and elemental pattern described as a Hertzian dipole.

single element in the array. This assumes that every element in the array is identical.

As an example, the radiation pattern for the array used above will be found assuming that every element in the array is a Hertzian dipole oriented parallel to the y-axis, with Electric field radiation pattern described by equation 22 [10] and elevation plane shown by figure 9.

$$\mathbf{E} = \hat{\boldsymbol{\theta}} j\eta_0 \frac{kIdcos(\theta)}{4\pi r} e^{-jkr} \tag{22}$$

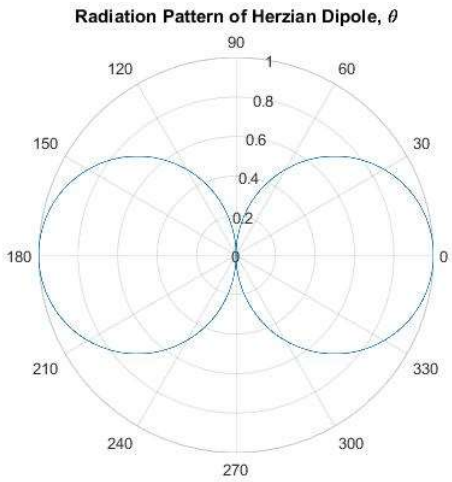

Fig. 9. Electric field radiation pattern of a Hertzian dipole in the elevation plane.

Finally, multiplying the array factor by the elemental electric field radiation pattern produces the actual electric field radiation pattern, displayed in figure 10.

D. Extension to a Two Dimensional Array

Two dimensional arrays allow for much more freedom than their one dimensional counterparts, as they allow for beam steering in more than one direction. To test the RLS algorithm on a two dimensional antenna array, a similar though more complex process is followed. The 2-dimensional array in test is shown in figure 11 and is located in the x-z plane.

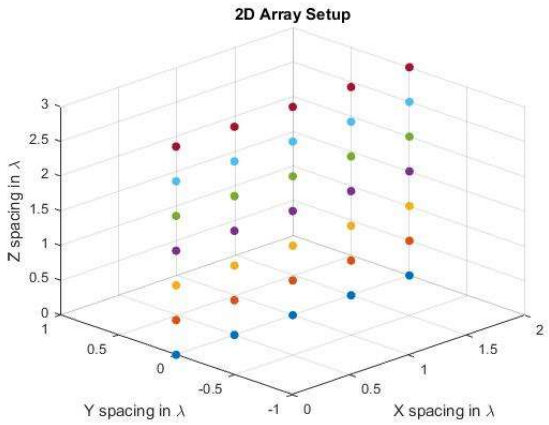

Fig. 11. Graphical representation of the 2 dimensional antenna array, oriented in the x-z plane.

The fist step of this process is to create new steering vectors for the desired and interference signals that depends on the spherical dimensions θ and ϕ. To do this, the 2-dimensional array is thought of as a 1-dimensional array in the z-direction comprised of 1-dimensional arrays in the x direction. If this is done, the total steering vector of the N×M array is simply the product of the steering vectors of each array, shown in equation 23. In this equation, d is the spacing between elements in terms of the wavelength, M is the number of elements in the x direction, N is the number of elements in the y direction, and θ and ϕ are the incident angles.

$$a_z = \sum_{i=0}^{M-1} e^{-j2\pi di*cos(\theta)}$$

$$a_x = \sum_{p=0}^{N-1} e^{-j2\pi dp*cos(\phi)}$$

$$a = a_z * a_x = \sum_{i=0}^{M-1}\sum_{p=0}^{N-1} e^{-j2\pi d[i*cos(\theta)+p*cos(\phi)]} \quad (23)$$

The desired signal is multiplied by the steering vectors obtained from the directions of the interference signal and the desired signal, then summed together with additional Gaussian noise. The final signal obtained is what each element of the antenna array receives and will be used in the recursive least squares algorithm.

The resulting array factor is shown in figure 12, where the interference direction is shown in red and the desired direction in green. It can be seen that this simulation of the recursive least squares algorithm for determining the weights of a 2-dimensional antenna array was successful, though this is not always the case.

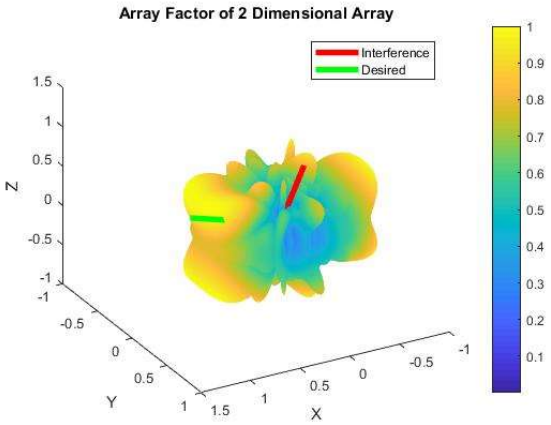

Fig. 12. Array factor resulting from a successful simulation of the RLS algorithm on a 2-dimensional array.

Due to the randomness of the interference in the system, it was often seen that the array factor would vary drastically between simulations. Many times, the simulation results in an array factor that nulls both incoming signals, shown in figure 13, instead of solely the interference. A solution to this problem was not found.

E. Possible Improvements

One of the main drawbacks in any adaptive antenna array system is the computational time required to determine the optimal weights to achieve desired system response. If the system is changing too rapidly, the weights might be useless by the time they are found. This problem is apparent using

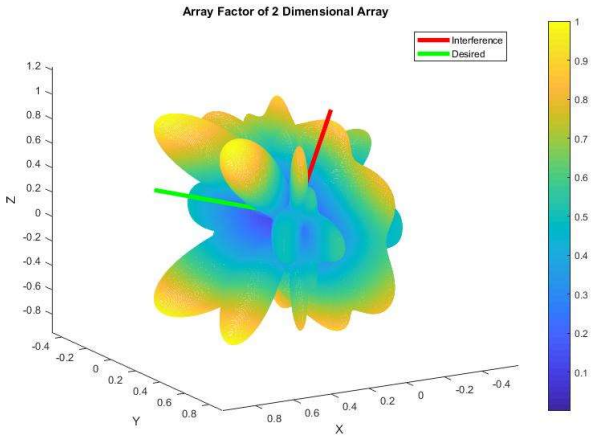

Fig. 13. Array factor resulting from an unsuccessful simulation of the RLS algorithm on a 2-dimensional array.

a sequential coding language like Matlab or Embedded C which only allows for one calculation to be done at a time. As discussed previously, equations 18 and 13, equations 12 and 7 can be computed simultaneously. Therefore not allowing for parallel computing is wasting time. One way to fix this is by using hardware such as graphical processing units which are compatible with parallel processing. Using these, the computational time is virtually cut in half as you only have to weight for two groups of equations to be calculated instead of four.

Another useful hardware tool which would further decrease computational time is the field programmable gate array (FPGA). FPGAs utilize lookup tables, allowing for virtually instantaneous and completely parallel computations of updated parameters per iteration [7]. Therefore the compute time required to calculate weights is comparable to the clock speed times the desired number of iterations. Both of these tools increase the realizable complexity of smart antennas.

V. Applications

A. High Frequency Wireless Communications

With the growing number of devices connected to wireless networks, frequency bands are becoming crowded and overused. To combat this, the FCC has recently freed space in the extremely high frequency band to house the fifth generation of wireless communications. Because of the magnitude of space freed, wireless carriers will be alloted a bandwidth of 200 MHz or higher compared to the current fourth generation at 20 MHz per carrier, allowing for a drastic increase in data transfer rates [6]. Though the frequencies allotted are less cluttered, the increase in frequency introduces a new problem.

The Friss Transmission Formula is shown in equation 24. It shows the relationship between the power at the receiver P_R and the power at the transmitter P_T in terms of the gain of the transmitter G_T, the gain of the receiver G_R, the distance from the source R, as well as the wavelength λ of

the transmitted electromagnetic wave. As the frequency of operation increases the wavelength decreases, with relationship defined by equation 25. Together these equations prove that, as the frequency of the transmitted signal increases with a constant transmitting power P_T, the power received decreases. This relationship between frequency and power received with constant gains, transmitting power, and distance from the source is seen in figure 14.

$$\frac{P_R}{P_T} = \frac{G_T G_R \lambda^2}{(4\pi R)^2} \tag{24}$$

$$\lambda = \frac{c}{f} \tag{25}$$

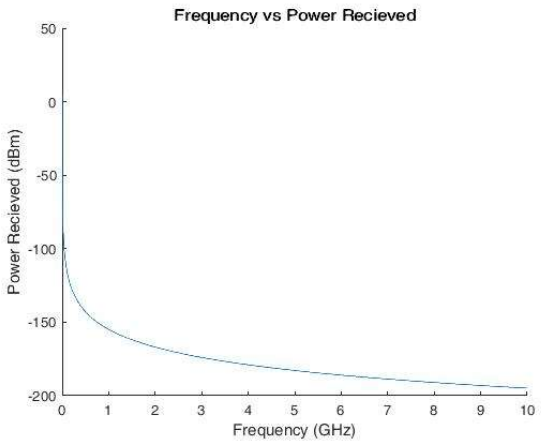

Fig. 14. Friss formula graphed keeping all but the frequency of operation constant.

This low received signal power creates a need for high gain receivers that are not easily affected by noise from any source other than the desired transmitter. The adaptive antenna array using the recursive least squares algorithm discussed in this report would fulfill this need, dynamically nulling unwanted signals while leaving the desired signal unaffected with a high enough power level to recover information.

B. Cancer Treatment Hot-Spot Prevention

One method found for eliminating tumors in the human body is through hyperthermia, the condition where a certain part of the body has a higher temperature than normal. This can be accomplished through the use of high power electromagnetic waves, which transfer energy into the tumors heating them and eventually destroying the cells. With advancements in nanopartical technology, allowing for the ability to effectively change the electromagnetic properties of the cancer cells [8], this technique is becoming more attractive.

One issue with electromagnetic induced hyperthermia is side-lobe radiation creating unwanted hot-spots on the patients body, which can cause discomfort or damage to their skin and internal tissue. By placing nulls in the direction of the created hot-spots, and using an adaptive algorithm such as the recursive least squares, one can calculate the array weights required to reduce the intensity of the radiation towards these hot-spots, allowing for a more effective and less evasive treatment [9].

VI. Conclusion

The recursive least squares algorithm is found to be a viable method for the calculation of element weights for an adaptive antenna array, successfully nulling unwanted interference while keeping desired signals intact. Once the weights are calculated, the array factor and in turn the total radiation pattern of the antenna array can be determined. This method proves to be successful for one-dimensional arrays, as well as a possibility for the extension to two and three-dimensional arrays.

With the use of this adaptive nulling technology the expansion of wireless communications into the extremely high frequency band is possible, as power loss and signal to noise ratio are of high concern due to the high losses at these wavelengths. As technologies such as graphical processing units and field programmable gate arrays continue to advance, adaptive arrays and other forms of machine learning will become commonplace and used in even the smallest of devices.

Acknowledgment

We would like to thank Dr. Randy Haupt for giving us such an interesting and relevant project that will give us a ground to stand on as we begin our careers in the industry of antennas and wireless communications.

References

[1] D. Pollock, "Recursive Estimation in Econometrics", Journal of Computational Statistics and Data Analysis, vol. 44, pp. 37-75, 2003.
[2] S. Patel, S. Panchal and H. Mewada, "Comparative Study of LMS and RLS Algorithms for Adaptive Filter Design with FPGA", Progress In Science in Engineering Research Journal, vol. 02, no. -, pp. 185-192, 2014.
[3] MIT OpenCourseWare, "Introduction to Recursive-Least-Squares (RLS) Adaptive Filters", The Massachusetts Institute of Technology Department of Mechanical Engineering, Cambridge, MA, 2008.
[4] "6.5.4.1. Mean Vector and Covariance Matrix", Itl.nist.gov.
[5] R. Haupt, Antenna arrays, 1st ed. Hoboken, N.J.: Wiley-IEEE Press, 2010, p. 476.
[6] Cellular Telecommunications and Internet Association, "High Band Spectrum: The Key to Unlocking the Next Generation of Wireless", Washington, DC, 2018.
[7] "FPGA Fundamentals", Ni.com, 2012. [Online]. Available: http://www.ni.com/white-paper/6983/en/.
[8] Amin Md., Z. (2014). Application of Electromagnetic Waves in Cancer Treatment by Hyperthermia. Scientific Research.
[9] A. Fenn and G. King, "Adaptive Nulling in the Hyperthermia Treatment of Cancer", The Lincoln Laboratory Journal, vol. 5, no. 2, pp. 223 - 240, 1992.
[10] S. Hum, "Ideal (Hertzian) Dipole", ECE422: Radio and Microwave Wireless Systems, 2018.