Universidade de Aveiro
2022

Bárbara
Cecílio Matos

Simulador de Sistemas de Transmissão

Transmission System Simulator

Universidade de Aveiro
2022

Bárbara Cecílio Matos

Simulador de Sistemas de Transmissão

Transmission System Simulator

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia Eletrónica e Telecomunicações, realizada sob a orientação científica do Doutor Telmo Reis Cunha, Professor associado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor Filipe Miguel Esturrenho Barradas, Professor Investigador do Instituto de Telecomunicações da Universidade de Aveiro.

o júri / the jury

presidente / president	Prof. Doutor Manuel Alberto Reis de Oliveira Violas Professor auxiliar da Universidade de Aveiro
vogais / examiners committee	Prof. Doutor Manuel Cândido Duarte dos Santos Professor auxiliar da Faculdade de Engenharia da Universidade do Porto
	Prof. Doutor Telmo Reis Cunha Professor associado da Universidade de Aveiro

agradecimentos / acknowledgements

Esta dissertação significa o fim de uma etapa muito especial da minha vida, pois traz com ela o resumo de cinco anos cheios de emoções. Como tal, gostava de referir alguns dos nomes que mais foram relevantes, não só na entrega da mesma, mas também no decorrer de todo este tempo. Começo por agradecer ao Professor Telmo Cunha, meu orientador de Dissertação, pois sem ele este trabalho não teria sido desenvolvido. Agradeço, também, aos meus pais, que mesmo não estando comigo diariamente, me permitiram chegar onde cheguei e me deram força para alcançar todos os meus objetivos. Um obrigada muito especial aos amigos Joana, Hugo, Gustavo, Inês Rocha e Bruno por terem calhado na minha turma no primeiro ano e também à Inês Araújo, ao Frederico e à Ana por se terem juntado. Por último, mas não menos importante, muito obrigada, Diogo, por todas as ideias, sugestões e por todo o apoio.

Palavras Chave

5G, Antenna array, Power Amplifier, Digital predistorter, Mutual coupling.

Resumo

A procura de sistemas altamente lineares, eficientes e que consigam dar resposta a muitos utilizadores, tem sido uma das grandes preocupações dos operadores de telecomunicações. Para dar resposta a todas estas necessidades, começaram, recentemente, a ser desenvolvidos sistemas de comunicações 5G. A transição para frequências mais altas associada a estes sistemas traz consigo novos desafios no design dos transmissores, como por exemplo o facto de serem necessários arrays de antenas. Estes, para além de poderem ter um grande número de elementos, provocam, também, a existência de acoplamento eletromagnético entre eles, tornando-se, desta forma, muito difícil testá-los a nível do circuito. Surge, assim, esta dissertação com o objetivo de desenvolver uma plataforma onde seja possível testar, de forma rápida e amiga do utilizador, este tipo de sistemas. No segundo capítulo são apresentadas mais aprofundadamente algumas características dos arrays de antenas, reforçando a necessidade da plataforma. O terceiro capítulo apresenta o desenvolvimento da mesma, servindo como manual de instruções para alguém que, futuramente, lhe queira acrescentar funcionalidades. No quarto capítulo são apresentados os resultados obtidos e é feito um tutorial para um futuro utilizador da plataforma. Por fim são apresentadas as conclusões e o trabalho futuro.

Keywords 5G, Antenna array, Power Amplifier, Digital predistorter.

Abstract

The search for highly linear, efficient systems that can respond to many users has been one of the major concerns of telecommunications operators. In order to respond to all these needs, 5G communication systems have recently started to be developed. The transition to higher frequencies associated with these systems brings with it new challenges in transmitter design, such as the need for antenna arrays. These, in addition to being able to have a large number of elements, also cause the existence of electromagnetic coupling between them, making it very difficult to test them at the circuit level. Thus, this dissertation arises with the objective of developing a platform where it is possible to test, in a fast and user-friendly way, this type of systems. In the second chapter, some characteristics of antenna arrays are presented, reinforcing the need for the platform. The third chapter presents its development, serving as an instruction manual for someone who, in the future, wants to add new features to it. In the fourth chapter, the obtained results are presented and a tutorial is made for a future user of the platform. Finally, the conclusions and future work are presented.

Contents

List of Figures

List of Tables

Glossary

DPD	digital predistorter
GUI	graphical user interface
MIMO	multiple-input-multiple-output
PA	power amplifier
RF	radio frequency
1G	first generation
3G	third generation
4G	fourth generation
5G	fifth-generation

CHAPTER 1

Introduction

The search for highly linear systems that are more energy efficient and have very wide bandwidth has been one of telecommunications operators' greatest concerns. Considering that fifth-generation (5G) communication systems are continuously being developed, the concern to have mobile systems with greater capacity and efficiency, serving more users and having a lower cost has greatly increased over the last few years [1].

Comparing with previous technologies, one of the biggest differences in 5G networks is the shift to higher frequencies (usually in the range from 28 to 90GHz) where there is a much higher propagation attenuation and a lower radiation power is reached [2]. Hence, the need to use high-directivity antennas to ensure that the signal power sent to a certain receiver is sufficient, and considering that transmission architectures have been driven for multiple-input-multiple-output (MIMO) systems, which make use of multiple radio frequency (RF) power amplifiers (PAs) to drive an antenna array, to implement these systems becomes one of the biggest challenges of millimeter wave antenna arrays [3].

When transitioning to these types of systems, the isolation between the array and the power amplifier is often removed. As the array elements are electromagnetically coupled, the output signal from each antenna interferes with the neighboring antennas and, therefore, the behavior of each PA does not depend only on the input signal, but also on the coupling signal coming from the neighboring antennas [3].

1.1 Motivation and Context

In addition to these challenges in building antenna arrays, they can also be associated with some disadvantages, related to requirements for the signal-to-noise-and-distortion ratio and the decrease of available bands in the radiofrequency spectrum [4].

With the significant increase in the bandwidth of the signals to be transmitted, the simulation of radio-frequency amplifiers has become a long and exhausting task. In the development of transmission systems, it is necessary to run multiple simulations at the electronic circuit level, as it is necessary to analyze, beforehand, the expected behavior of these circuits (for example, the level of distortion, the energy efficiency, etc.), which is a task that consumes a long time. Furthermore, with the shift to higher frequencies associated with 5G communications and consequent need to have smaller antennas arranged in arrays, to obtain the necessary gain overcoming the attenuation effects [5], it becomes necessary to implement techniques that aim to significantly reduce the simulation time required by electronic circuit simulators.

Thus, the motivation to create a platform that is as complete and user-friendly as possible, capable of meeting these needs, arises.

1.2 Objectives

The main purpose of this dissertation is to develop a platform which allows simulating and obtaining the behavior of transmission systems based on mathematical models previously extracted from the electronic circuits that implement amplifiers.

With the use of mathematical models that imitate the input-output behavior of circuits, it is possible to avoid the simulation at the electronic circuit level. This task, which manually would take a long time, can be simulated in the platform in a few minutes.

This platform should have a wide range of features, allowing the execution of simulations and also including configurable pre-distortion algorithms.

More specifically, the platform must give the user the possibility to specify an antenna array (define its number of rows and columns), as well as define, for each antenna, its input signal and the mathematical models that mimic the behavior of its power amplifier and digital predistorter (DPD). It should also be noted that the PA model must include the coupling signals from the other antennas and, for this, the platform must also consider the necessary S parameters. The possibility to change the fundamental frequency of the system and to change the distance between each antenna should also be given to the user. At the end of the simulation, it should be possible to observe the behavior of each individual antenna as well as the simulated antenna array.

All of this must be done in a computationally efficient way and be user-friendly at the same time.

1.3 Structure

- **Chapter 1:** Introduction, motivation and the structure of this dissertation.
- **Chapter 2:** Characteristics associated with antenna arrays' systems and how can the platform be useful to solve them.
- **Chapter 3:** Development of the platform to simulate the transmission systems; this chapter provides the reader the architecture followed for the platform implementation and serves as a support manual for those who may desire to add it new features and functionalities.
- **Chapter 4:** Platform simulation and results; this chapter serves as a support manual for those who want to use the platform.
- **Chapter 5:** Conclusions and future work.

CHAPTER 2

Array transmitters and their characteristics

The evolution of mobile communication systems has been seen in the last decades, from the first generation (1G) to the fourth generation (4G), in which the most recent generations have always emerged with significant performance upgrades.[5]

Recently, mainly to respond to the need for higher capacity and efficiency, 5G communication systems began to be developed. One of the biggest differences in this latest generation of mobile communications is the shift to higher frequencies. As it is possible to see in figure 2.1, 5G frequency bands (also known as millimeter-wave bands) can provide much wider bandwidths than 3G and 4G frequency bands and therefore can support higher data rates.

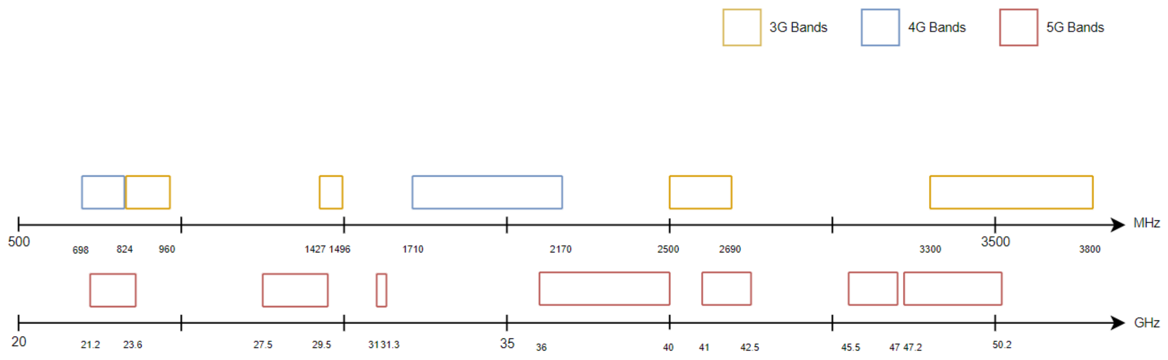


Figure 2.1: Frequency bands in 3G, 4G and 5G [5].

Unfortunately, the transition to these millimeter-wave systems is a new challenge on transmitter design. With the increase of the operation frequency and consequently of the path loss, it becomes desirable to have smaller antennas arranged in arrays in order to obtain the required gain overcoming attenuation effects.

This chapter intends to present some antenna arrays characteristics, reinforcing the idea that a platform that can bypass them is needed.

2.1 Antenna arrays

An antenna array is simply a set of antennas that work together as a single one. Unlike the antennas used in 4G that radiate the signal in all directions, wasting a lot of energy towards the directions where no receiver is located, 5G directional antenna arrays only cover part of the space [6]. Using an antenna array allows the transmitter to focus the radiated beam towards the directions where the receivers are located, clearly using the available energy in a much more efficient way. Besides this advantage and the fact that, due to the small wavelength of millimeter-waves, it is possible to accommodate a large number of antennas in a small physical space [2], this system has also some technical issues that need to be taken into account.

2.2 Circuit-level analysis

One of the problems that array transmitter design faces is the fact that circuit-level simulation easily becomes impractical to use. Not only millimeter-wave circuits need to rely on very accurate simulation (as laboratory probing is very difficult at these frequencies), but mostly the fact of requiring several amplifying circuits (one for each antenna in the array) to build an array makes it impossible to perform the usual circuit-level simulations, as the number of circuits and electronic components (models) becomes too high (either the circuit-level simulation consumes too much time, or even becomes impossible to simulate) [1]. Moreover, as in any design process, several iterations are commonly required until the final circuit can be considered to be finalized, whereas several trimming and small modifications are introduced from iteration to iteration – highly time consuming simulation would simply make this process unbearable in practice. To overcome this problem, a system-level analysis, where all the important blocks such as PAs, DPDs, antennas, etc are previously modeled by dedicated mathematical formulations, is necessary.

This is the first aspect where the platform can be useful, since the goal is to obtain estimates of the behavior of transmission systems based on mathematical models previously extracted from the electronic circuits that implement the amplifiers, which, in the design stage, can be extracted from circuit-level simulation of a single amplifier unit. System-level simulation is computationally very efficient, being able to reduce the simulation time of a complex array transmitter to the order of seconds or minutes, whereas in the circuit-level simulation domain it would require several hours or even days (if the simulator would be able to converge at all).

2.3 Isolators

Conventional array transmitters, designed for frequencies on the units of GHz, commonly use an isolator between the amplifier output and the antenna, as illustrated in figure 2.2. This isolator, typically a circulator with the third port connected to a 50 Ohm resistive load, directs the electromagnetic wave coming from the amplifier to the antenna, while the wave being reflected back from the antenna is directed to the 50 Ohm load, where it is dissipated. This way, the amplifier output is always seeing a constant load, regardless of signals being reflected back from the antenna (which may be due to antenna mismatching, reflections on objects in the transmitter vicinity, or coming from the signals emitted from the other elements in the array — this case is described in the next Sub-section).

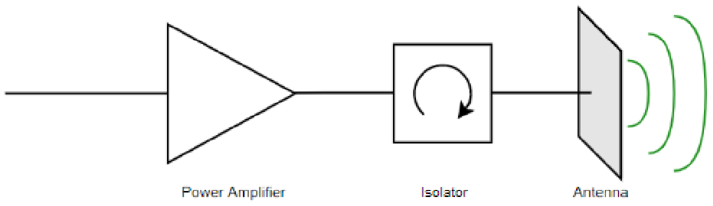


Figure 2.2: Isolator block representation.

Commercial isolators are commonly built from ferromagnetic elements, as these need to be non-reciprocal devices, being quite bulky and expensive components. When the need to use integrated phased array transmitter systems arises and one of the goals is to minimize the size of each element of the array, it becomes no longer practicable to include them [7]. With the technology moving towards millimeter-wave frequencies, circuit and antenna size reduces accordingly, but that reduction is not being accompanied by isolators, creating the necessity to remove these from the array elements.

2.4 The mutual coupling problem

Without the inclusion of isolators, a problem arises in antenna array applications — the mutual coupling. Mutual coupling is an electromagnetic phenomenon which consists of the interaction of antennas that are close to each other, so it is a common problem in antenna arrays [8].

As the elements of the array are electromagnetically coupled, the waves that feed the PAs of the antennas also come from the outputs of the other antennas, causing an apparent variable load to be created at the output of each PA. In these systems, to define the behavior of the PA, its input signal is not enough, because the signal changes according to the coupled signal.

To better understand this phenomenon, in figure 2.3 are presented two power amplifiers, each one connected to an antenna. Each power amplifier has an input signal, *x1* and *x2*, and an output signal, *y1* and *y2*. In addition to these signals, there is another signal affecting the amplifier (therefore, seen topologically also as an input to the amplifier), which corresponds to part of the output signal from the neighboring antenna, which arises due to the electromagnetic coupling existing between the two antennas. Represented as *x12* is the signal from the second antenna that enters the first one.

This additional signal, usually named coupling signal, will affect the behavior of the amplifier, as demonstrated in [3]. In short, assuming that signal $x12$ is correlated with signals $x1$ (and, therefore, with $y1$), the amplifier will observe an apparent modification of its load, and it is widely known that the amplifier performance changes rather significantly with load variations.

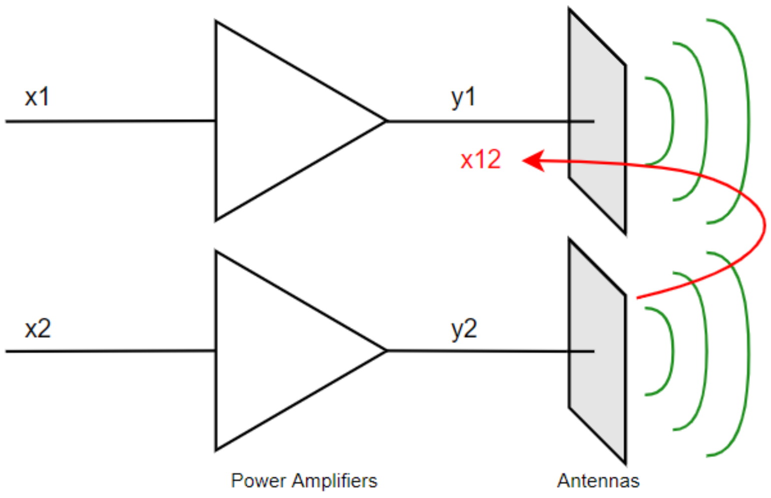


Figure 2.3: Mutual coupling representation.

Thinking about simulation through mathematical models, in this case the function that implements the power amplifier would have to be of the type $f(x_1, x_{12})$ where $x1$ is the conventional input signal of the amplifier, and $x12$ is the coupling signal originated from the signals transmitted by the antennas in the vicinity of that of the considered element. So, considering the antennas in the previous figure, the equations would look like this:

$$\begin{cases} y_1 = f(x_1, x_{12}) \\ y_2 = f(x_2, x_{21}) \end{cases} \tag{2.1}$$

where $x_{12}(t) = \mathscr{F}^{-1}(S_{12}(w)\,\mathscr{F}(y_2(t)))$ and $x_{21}(t) = \mathscr{F}^{-1}(S_{21}(w)\,\mathscr{F}(y_1(t)))$. Putting everything together in one equation, the expression would be as follows:

$$\begin{cases} y_1(t) = f(x_1, \mathscr{F}^{-1}(S_{12}(w)\,\mathscr{F}(y_2(t)))) \\ y_2(t) = f(x_2, \mathscr{F}^{-1}(S_{21}(w)\,\mathscr{F}(y_1(t)))) \end{cases} \tag{2.2}$$

and S_{12} and S_{21} are the S parameters that describe the linear relations, in the frequency domain, between incident and reflected waves at the two ports of the electromagnetic system comprehending the two antennas.

As it is possible to see in the equation system 2.2, a recursive problem arises at this point, since to get the output of the first antenna, it is necessary to know the output of the second antenna and vice versa. To solve this recursive problem, two distinct methods were analyzed, as described next, which resort to iterative processes.

2.4.1 Newton-Raphson's method

The Newton-Raphson's method is a quick way to find the root of a real-valued function, $g(x, y) = 0$, and it uses the idea that a continuous and differentiable function can be approximated by a line tangent to the function. In figure 2.4 an example of a graphical representation of the method is shown [9].

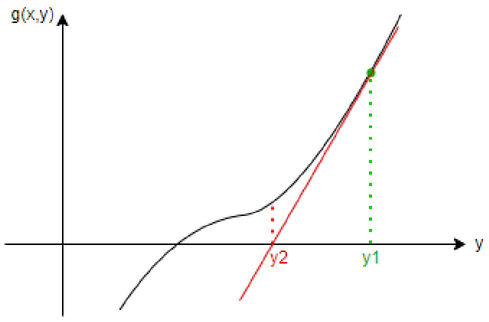


Figure 2.4: Representation of Newton-Raphson's method.

Assuming that y1 is the first tentative solution (randomly selected, for instance), the tangent at the point $g(x, y)$ corresponding to $y1$ is drawn in direction to the y axis. The point where the axis is intersected is the second solution, $y2$. The process is repeated (where every iteration starts from the solution of the previous iteration) until a point sufficiently close to the final solution is reached, for example through a predefined threshold.

To find the tangent line at a point, the derivative of the function at that point is calculated, following the equation:

$$\left.\frac{dg(x,\, y)}{dy}\right|_{y=y_k} = \frac{g(x,\, y_k)}{y_k - y_{k+1}} \Leftrightarrow y_{k+1} = y_k - \frac{g(x,\, y_k)}{\left.\frac{dg(x,y)}{dy}\right|_{y=y_k}} \tag{2.3}$$

Since in the case of this work, the method will be applied to an array of elements, it will be considered in matrix form:

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ ... \\ Y_N \end{bmatrix} = \begin{bmatrix} f_1(x_1, c_{12}(y_2) + c_{13}(y_3) + ... + c_{1N}(y_N)) \\ f_2(x_2, c_{21}(y_1) + c_{23}(y_3) + ... + c_{2N}(y_N)) \\ ... \\ f_N(x_N, c_{N1}(y_1) + c_{N2}(y_2) + ... + c_{N,N-1}(y_{N-1})) \end{bmatrix} \tag{2.4}$$

where c are the S-parameters in the time domain, to be multiplied by the output signals of each antenna, y, and $c_{ij}(y_j)$, with $i, j = 1, ..., N$, are the time domain operators corresponding to the application of the S parameters between antenna pairs (together forming the coupling signal affecting the amplifier of each element).

Equation 2.4 can be represented in the condensed form $Y = F(X, Y)$, where X is the matrix containing all the samples of all the signals being input to the array elements (in each row are the successive samples of one of the input signals — it is assumed that the input signals of all elements follow the same sampling scheme), Y is the equivalent matrix for the signals being output by each antenna, and $F(X, Y)$ is the matrix aggregating the dual-input-single-output models of the amplifiers. This simple equation can be put in the form $G(X, Y) = Y - F(X, Y) = 0$, to which can be directly applied the Newton-Raphson iterative method:

$$Y_{k+1} = Y_k - J_k^{-1}\, G(X, Y_k) \tag{2.5}$$

where J_k is the Jacobian of the matrix $G(X,\, Y_k)$ for each iteration.

This method, although very good in its approximation to the final solution, has proven to be difficult to implement and computationally inefficient if no analytic expression is available for the Jacobian matrix, as was the case in the performed implementation (it was being determined numerically, by applying a small deviation to each output signal being calculated, one at a time). The computational efficiency associated with the calculation of the Jacobian is further compromised as the number of samples of the input signals increases, which is common in wide bandwidth communications. So, with the initial idea of having a robust simulator, another alternative was afterwards explored.

2.4.2 Iterative learning control method

In this second approach, the simplicity of the method, in terms of computational efficiency, was privileged, but without relaxing the accuracy of the result. In this sense, knowledge of a method which was published in [10] for performing digital predistortion of a power amplifier was considered. This method iteratively finds the signal for which a recursive process converges, using as update the error which is being minimized. This method can be directly transported to the recursive problem of equation 2.4

$$Y_{k+1} = Y_k + \lambda \left(G(X, Y_k) - G(X, Y_{k-1}) \right) \tag{2.6}$$

where k refers to the iteration, λ is a factor that controls the convergence speed of the process, usually named learning rate and $G(X, Y)$ is the matrix already explained, composed by all power amplifiers models which receives as arguments the matrices X and Y mentioned above.

It should be noticed that in common cases, this method should present a good convergence towards the correct solution, since the coupling signal produces a deviation of the amplifier behavior, but such deviation is usually not so significant that it would make the amplifier behavior drastically different (from the case where no coupling is present). Therefore, if the deviation introduced by the coupling signal is sufficiently small, this method should, in principle, converge to the solution.

To test the validity of this iterative method in an array of antennas where there is coupling between elements, a virtual array of antennas was created through the Simulink platform, with predefined mathematical functions for the dual-input-single-output model of the amplifiers, and also for the antennas' S parameters. Figure 2.5 shows the referred virtual array.

The objective of this test was to set a reference array system, where everything is known and which is simulated in an independent platform (Simulink, in this case, which has widely recognized accuracy in solving time domain simulations). The same setup is established in the simulation platform developed within this work, and the solutions of both platforms are, then, compared.

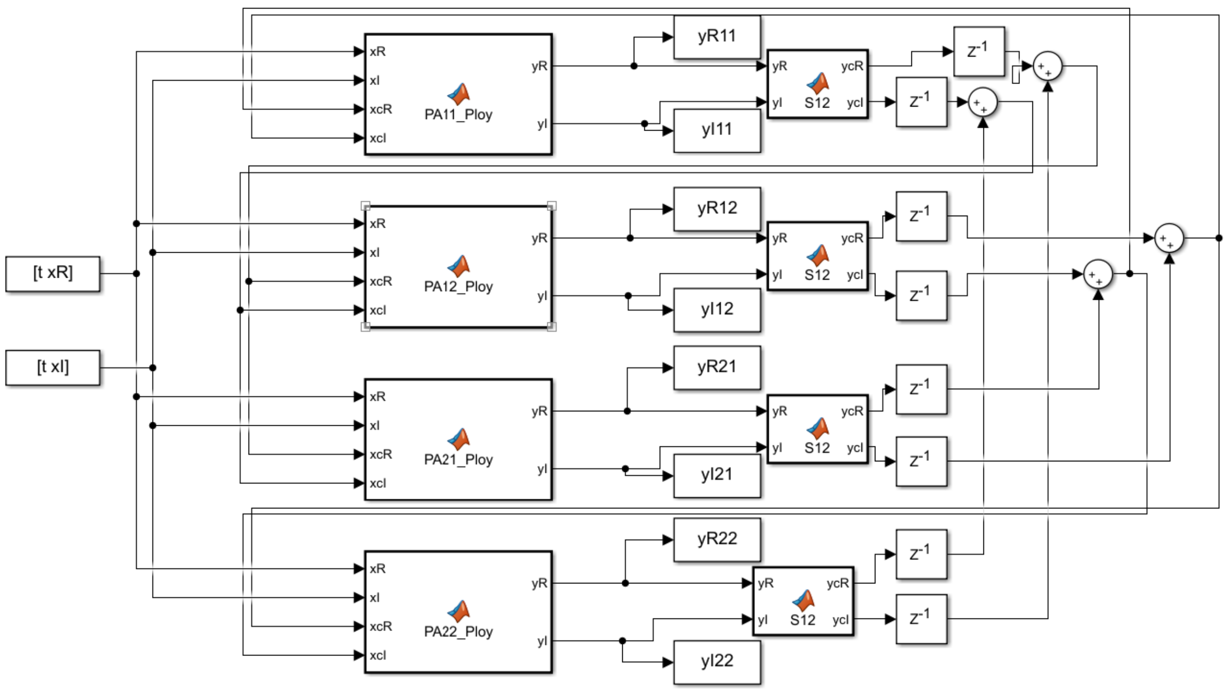


Figure 2.5: Representative diagram of the virtual antenna array, made in Simulink.

In this diagram, complex envelope signals were separated into real component, xR, and imaginary component xI, because Simulink does not consider the simulation of complex valued signals.

Due to the low complexity of the simulated array, with 4 elements in a 2x2 arrangement, the S parameters were considered only between adjacent elements, and not among diagonal ones. Figure 2.6 shows the considered coupling scheme.

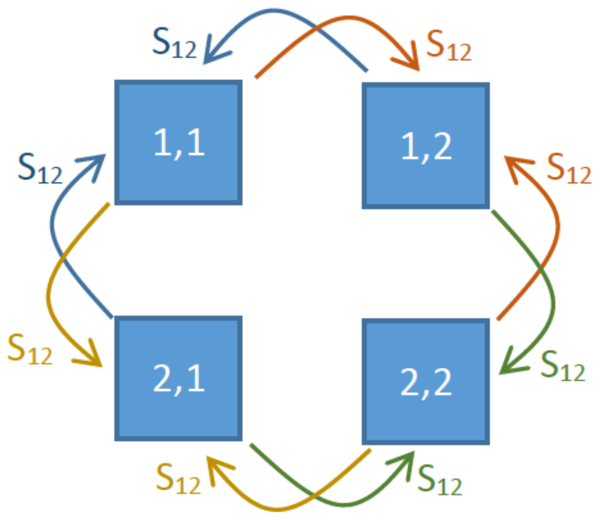


Figure 2.6: Coupling scheme considered in the test array.

All the blocks that simulate the amplifiers (PAs) use the same mathematical model to simulate their behavior:

$$y(n) = \left(a1 + a2\left|x(n)\right| + a3\left|x(n)\right|^2 + b\left|x_c(n-1)\right|\right)x(n) \tag{2.7}$$

with:

$$a1 = 1,\ a2 = -0.2,\ a3 = 0.1,\ b = -0.2 \tag{2.8}$$

being $x(n)$ the sample sequence of the input signal, $x_c(n)$ the sample sequence of the coupling signal coming from neighboring antennas, and $y(n)$ the sample sequence of the amplifier output signal.

The delay of one sample represented in equation 2.7, with respect to the coupling signal, was actually implemented as a characteristic associated with the S parameters, allowing the amplifier model implemented in Simulink to be completely static (something that is convenient to facilitate the simulation in Simulink). In the developed platform, this delay was incorporated into the amplifier function as in equation 2.7, facilitating the implementation of the frequency-domain S parameters, which become a simple gain for all frequencies, of magnitude equal to 0.2. Thus, the coupling signal coming from an antenna is equal to the signal arriving at that antenna from the PA, multiplied by 0.2.

After implementing the iterative method, whose development is presented in the next chapter, and using the same input signal, the output signal from one of the antennas was obtained and compared with the corresponding signal from the same antenna in the Simulink simulation, observing that the results from the two simulation platforms are identical.

In figure 2.7, on the left side, is presented the comparison between the antenna (1,1) output signals, obtained in Simulink (theoretical) and in the developed platform (simulated) already considering the coupling signals. On the right is shown the corresponding error signal.

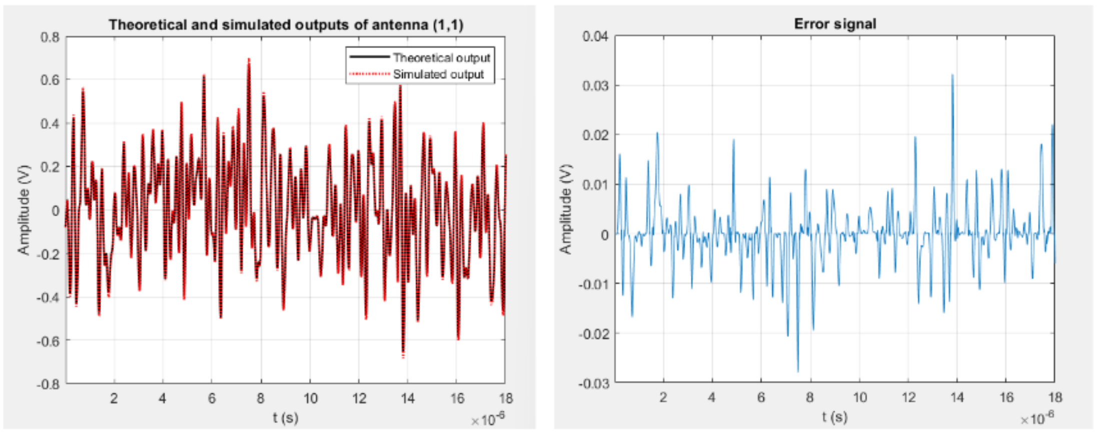


Figure 2.7: Antenna (1,1) output signals obtained by Simulink (theoretical) and in the platform (simulated) and the corresponding error signal.

In order to be able to see the previous signals more clearly, they have been approximated to a shorter time interval. The results obtained are then shown in the figure 2.8.

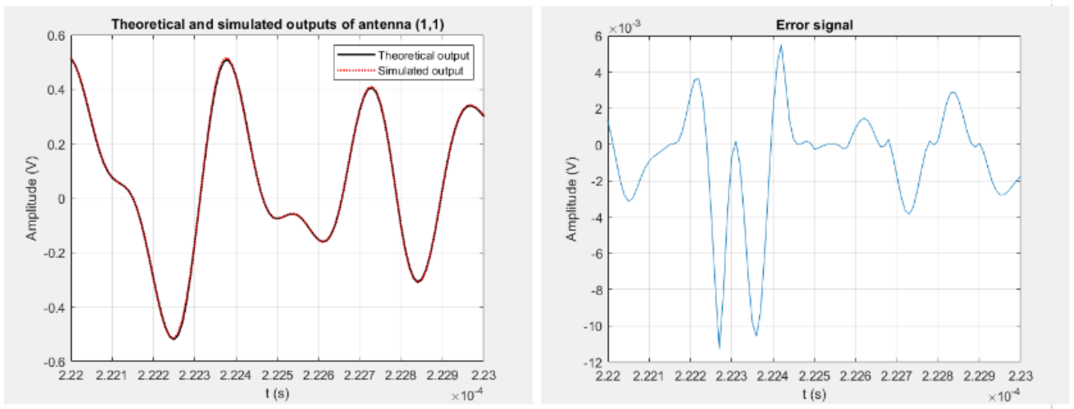


Figure 2.8: Antenna (1,1) output signals obtained by Simulink (theoretical) and in the platform (simulated) and the corresponding error signal.

Verifying that the error signal is in the order of mV and that the signals in the left image are overlapped, it was concluded that the predictions of the iterative method developed on the platform coincide with the results obtained in Simulink, and, consequently, this method can be used to calculate the coupling signals between elements.

The iterative simulation algorithm is a crucial part of the simulation platform that was developed, allowing the determination of the signals appearing at each node of the array transmitter. With these signals, several performance metrics and graphical results can be generated and displayed to the user, providing essential information on the overall (and particular) characteristics of the transmitter. The detailed design and development of the simulation platform is described in the next chapter. In the fourth chapter, several tests and results are shown, evidencing the potentialities of the conceived simulation tool.

CHAPTER 3

Platform development

In this section, the implementation details of the simulation platform are described, with two purposes in mind: i) to provide the reader the architecture followed for such implementation; ii) to serve as a support manual for those who may desire to add new features and functionalities to the platform.

The Transmission System Simulator was developed through the program MATLAB, more specifically the App Designer tool. This tool is an interactive development environment for designing app layouts and programming their behavior.

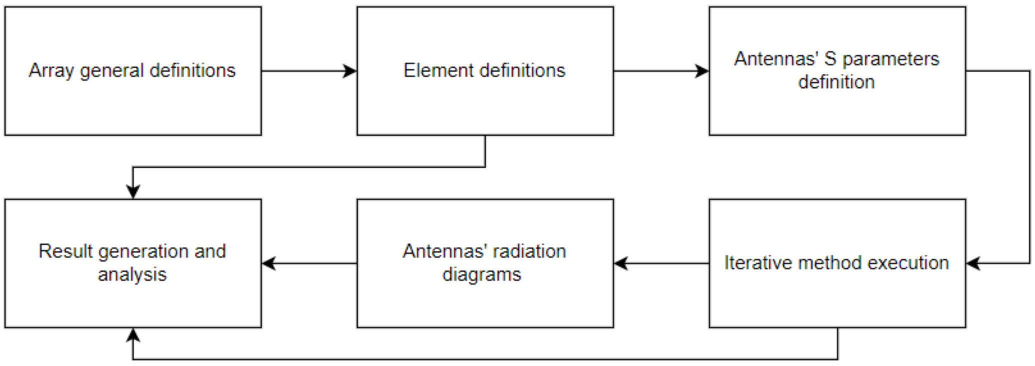
In addition, all the MATLAB toolboxes that were installed during the development of the platform are presented below, as they are necessary to perform some calculations and/or use some functions presented throughout the chapter:

- Signal Processing Toolbox
- Antenna Toolbox
- Communications Toolbox
- RF Toolbox
- SerDes Toolbox

In App Designer, the possibility to drag certain components to the desired location of the graphical user interface (GUI) is given to the user. Thus, the code that defines these components, such as their name, position, alignment, etc, is automatically created. When a component is intended to only appear after clicking somewhere else, then the code that defines that component must be created manually.

It is also possible to create callback functions, which are functions associated with the components, whose code is executed when they are clicked on. It is through these functions that the entire application works.

Moving to the graphical interface itself, the first function that is executed when the interface is opened is the *startupFcn()* function. In this function, the size of the graphical interface is defined, so that it adjusts to the resolution of the screen. After that, the procedure shown in the figures' 3.1 diagram is followed, so it will be further explained throughout the chapter.

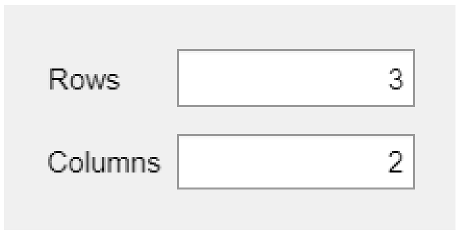


Array general definitions
Element definitions
Antennas' S parameters definition
Result generation and analysis
Antennas' radiation diagrams
Iterative method execution

Figure 3.1: General diagram of the interface behaviour.

3.1 Array general definitions

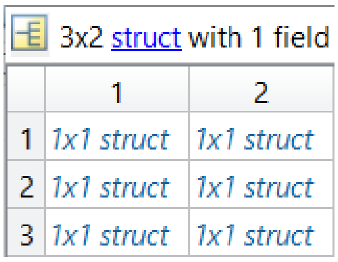
The first step of the diagram is to choose the size of the antenna array. Thus, two edit fields were created to insert numerical values, one to define the number of rows and another for the number of columns in the array (figure 3.2). In this case, both fields point to the same callback function, in which the functions that create the antenna array — *buildAntennaArray()* — and insert some more important components — *insertComponents()* — are called.



Rows	3
Columns	2

Figure 3.2: Edit fields for an example of 3 rows and 2 columns.

When the function *buildAntennaArray()* is executed, a variable of type *struct* (*AntArr*) is created. Each cell of this structure is also a *struct* itself. This way, it is possible to store in each cell the information related to each antenna throughout the program. In figure 3.3 it is possible to observe the appearance of the structure in case of the three rows and two columns selected for the antenna array size.



3x2 struct with 1 field

	1	2
1	1x1 struct	1x1 struct
2	1x1 struct	1x1 struct
3	1x1 struct	1x1 struct

Figure 3.3: *AntArr* structure.

Besides this variable, a matrix of buttons is also created in the graphical interface, each one corresponding to an antenna, as it is possible to see in figure 3.4.

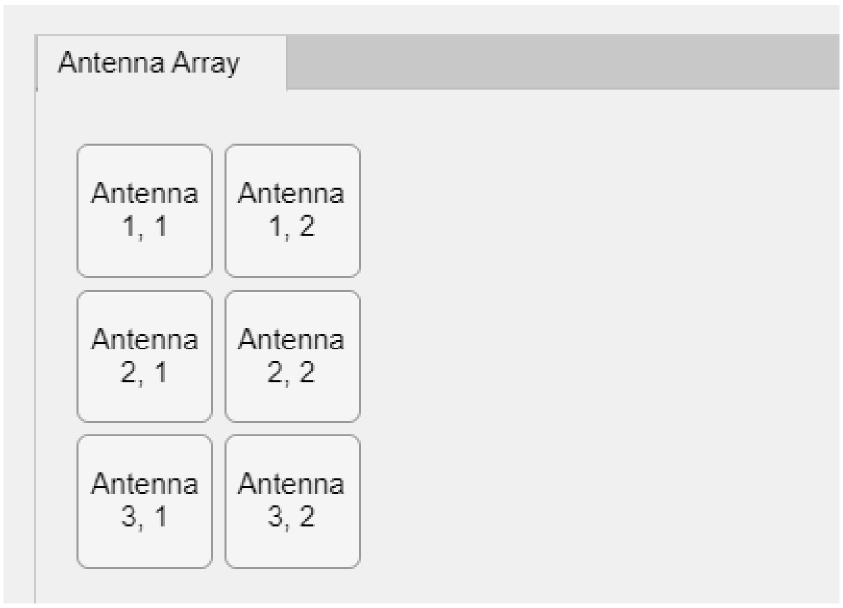


Figure 3.4: Antenna buttons in the interface.

The *insertComponents()* function is the one that inserts the remaining components in the GUI (Appendix A). For this reason, this function is only executed one time.

3.2 Element definitions

When one of the buttons of the button matrix of figure 3.4 is pressed, the *AntennaButton-Pushed()* callback function is executed. This function creates another page in the interface (Appendix B) where it is possible to upload the antenna input signal — pressing the *Input* button —, as well as the mathematical model of its power amplifier — pressing the *PA* button. It is also possible to upload the model of a digital predistorter — pressing the *Predistorter* button —, as well as remove it, making it unitary again — pressing the *Unitary DPD* button. There is also a button called *Update*, which is not relevant for now. In the figure 3.5 the buttons referred to can be seen.

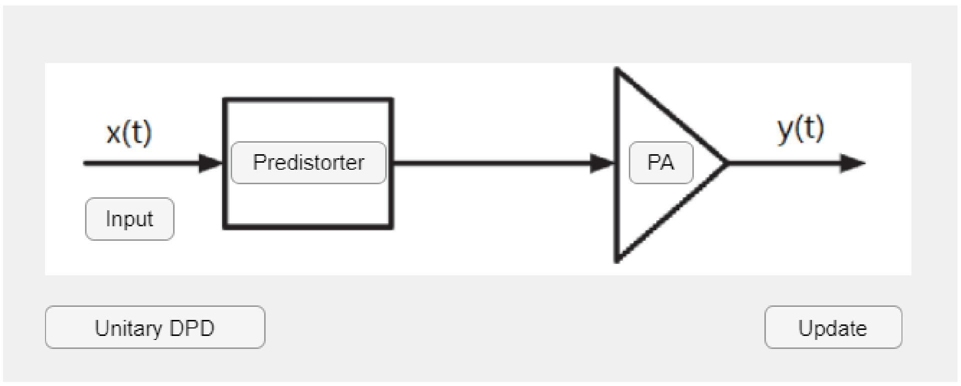
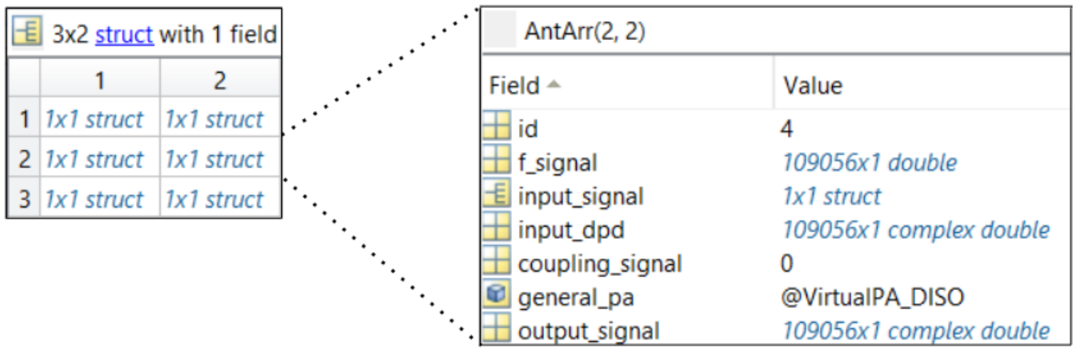


Figure 3.5: Antenna's interface.

When entering the input signal and the remaining mathematical models, the *AntArr* structure is uploaded with new information. In figure 3.6 this structure is presented again, but now after uploading the input signal and the power amplifier which, in this example, is named *VirtualPA_DISO*, and predistorter models.



3x2 struct with 1 field

	1	2
1	1x1 struct	1x1 struct
2	1x1 struct	1x1 struct
3	1x1 struct	1x1 struct

AntArr(2, 2)

Field	Value
id	4
f_signal	109056x1 double
input_signal	1x1 struct
input_dpd	109056x1 complex double
coupling_signal	0
general_pa	@VirtualPA_DISO
output_signal	109056x1 complex double

Figure 3.6: *AntArr* structure after defining an input signal and PA and DPD models.

As already mentioned, by pressing the *Input* button, the input signal from the antenna is uploaded. This input signal is a *.mat* file that contains two variables, each one a column vector. The first column, t, contains the time instants (in seconds) at which each sample of the input signal occurs, and the second column, x, has the respective sample values, which are complex-valued numbers (as the input signal is the baseband complex envelope of the RF modulated signal being transmitted). This file is stored in the *input_signal* variable of the *AntArr* structure, or in other words, in the *AntArr.input_signal* variable.

As soon as each input signal file is uploaded into the simulator, the frequency grid corresponding to the spectrum of that signal is calculated and stored in the variable *AntArr.f_signal*. This step is relevant for the simulation stage, as the signals produced by each PA model are then processed by the S-parameters of the antenna elements, and this processing is performed directly in the frequency domain. The input signal frequency grid is calculated by:

```
N = length(input_signal.x);
h = input_signal.t(2) - input_signal.t(1);
df = 1./(N*h);
AntArr.f_signal = [0:N-1]'*df - 1/(2*h);
```

When a predistorter model is selected, the input signal is passed through this model and the resulting signal is stored in the structure's *input_dpd* variable. This signal is then the input signal of the PA function, stored in the structure's *general_pa* variable. This function, which takes as arguments an input signal x and a coupling signal xc, considers, at this moment, that the coupling signal (*coupling_signal*) is zero, as shown in figure 3.6. Thus, the output signal of each element is calculated, assuming that there is no coupling between antennas, and stored in the variable *output_signal*. This signal is saved for later comparison with the signal version considering coupling. When a model of a predistorter is not selected, then a copy of the x component of the *input_signal* is stored in the variable *input_dpd*.

The *id* field is just a specific identifier of each antenna.

After obtaining all the signals related to each individual element, some graphic results are obtained, which will be presented in the next chapter. To represent these graphics, namely the *Transfer Function* and the *AM/AM gain*, it is necessary to calculate: i) the *input* and *output* signals' power:

$$P_x = x\, e^{j\omega} * x\, e^{-j\omega} = x * x^* = |x|^2 \tag{3.1}$$

where x is the signal and x^* is the respective conjugate; convert it to dBm:

$$P_{dBm} = 10\, log_{10}(\frac{P_x}{1mW}) = 10\, log_{10}(|x|^2) - 10\, log_{10}(0.001) = 20\, log_{10}(|x|) + 10\, log_{10}(1000) \tag{3.2}$$

and ii) each element gain, given by:

$$gain = \frac{P_{out}}{P_{in}}. \tag{3.3}$$

Since it is an exhaustive process to upload the signals for all antennas individually, in the case of a large array of antennas, it is also possible to define the same signals and functions for all antennas, by clicking on the *Equal antennas* check box.

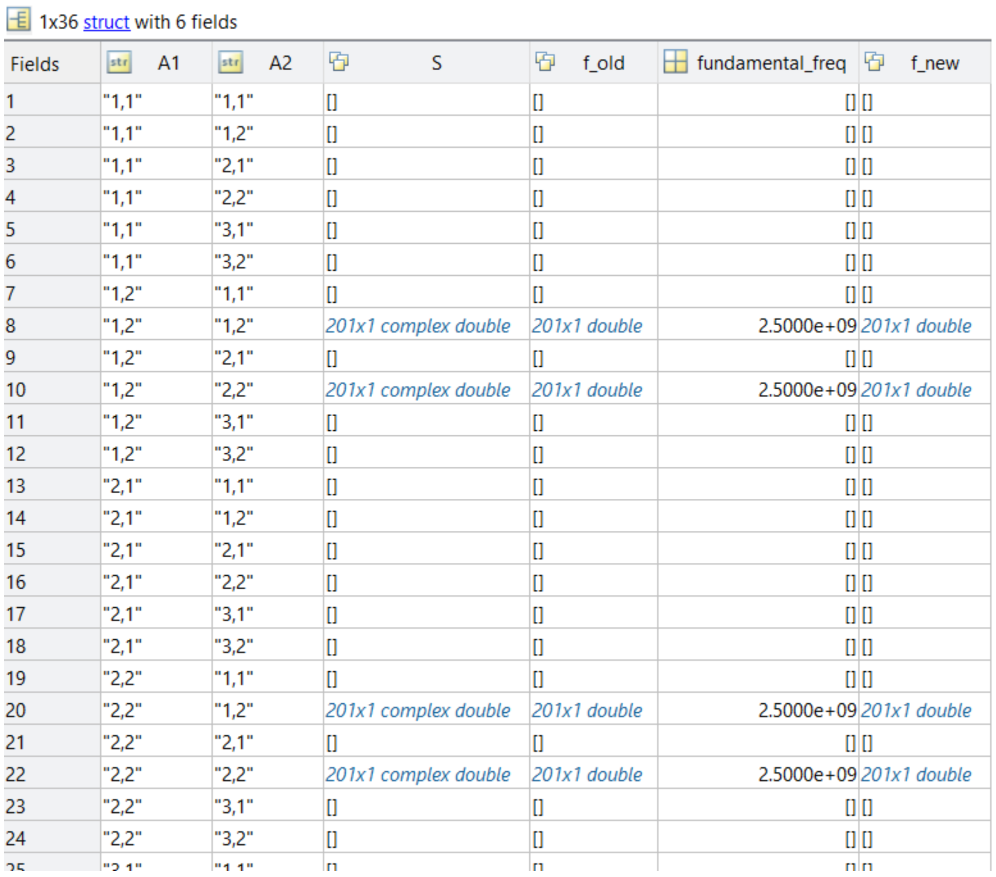
3.3 Antennas' S parameters definition

Once the signals of all antennas are defined, the choice of S parameters is next. For this, there are two options. The one that should be used first is the *S Parameters* button. This button allows the user to select multiple files, whose filename has the following format: *name_ant1(i1),ant1(i2)_ant2(i1),ant2(i2)*, where *ant1* corresponds to the first antenna and *ant2* to the second antenna that are being related by these files. The indices *i1* and *i2* correspond to the position of the antennas in the array. For example, by selecting the file *Sparameters*_1, 2_2, 1, a first antenna which in the antenna array is at position "1,2" (A1, in the figure 3.7), would be related with a second antenna which in the antenna array is at position "2,1" (A2, in the same figure), so it would correspond to the line number 9.

By pressing this button, the callback function *SParametersPushedButton()* is executed. Following the indications provided in the filenames, the frequency-domain S parameter relations are stored in the *sfiles_info* structure — figure 3.7 —, according to the data provided in the standard S2P file format (an interpreter for this format was incorporated into the simulator).

As this structure includes the relation between all the antennas in the array, which corresponds to the number of permutations of all antennas, two by two ($n!/(n-2)! = 6*5 = 30$), plus the cases in which each antenna relates to itself (6, because there are 6 antennas), corresponding to parameters S11, its size is 36. For the example used in figure 3.7, only one file was uploaded, to understand which are the structure's fields where values will be stored:

- Sparameters_1,2_2,2.s2p


1x36 struct with 6 fields

Fields	A1	A2	S	f_old	fundamental_freq	f_new
1	"1,1"	"1,1"	[]	[]	[]	[]
2	"1,1"	"1,2"	[]	[]	[]	[]
3	"1,1"	"2,1"	[]	[]	[]	[]
4	"1,1"	"2,2"	[]	[]	[]	[]
5	"1,1"	"3,1"	[]	[]	[]	[]
6	"1,1"	"3,2"	[]	[]	[]	[]
7	"1,2"	"1,1"	[]	[]	[]	[]
8	"1,2"	"1,2"	*201x1 complex double*	*201x1 double*	2.5000e+09	*201x1 double*
9	"1,2"	"2,1"	[]	[]	[]	[]
10	"1,2"	"2,2"	*201x1 complex double*	*201x1 double*	2.5000e+09	*201x1 double*
11	"1,2"	"3,1"	[]	[]	[]	[]
12	"1,2"	"3,2"	[]	[]	[]	[]
13	"2,1"	"1,1"	[]	[]	[]	[]
14	"2,1"	"1,2"	[]	[]	[]	[]
15	"2,1"	"2,1"	[]	[]	[]	[]
16	"2,1"	"2,2"	[]	[]	[]	[]
17	"2,1"	"3,1"	[]	[]	[]	[]
18	"2,1"	"3,2"	[]	[]	[]	[]
19	"2,2"	"1,1"	[]	[]	[]	[]
20	"2,2"	"1,2"	*201x1 complex double*	*201x1 double*	2.5000e+09	*201x1 double*
21	"2,2"	"2,1"	[]	[]	[]	[]
22	"2,2"	"2,2"	*201x1 complex double*	*201x1 double*	2.5000e+09	*201x1 double*
23	"2,2"	"3,1"	[]	[]	[]	[]
24	"2,2"	"3,2"	[]	[]	[]	[]

Figure 3.7: S parameters structure (*sfiles_info*).

Although all provided S parameters can be considered by the simulator, only s12 are stored in this structure, because these are the ones required to calculate the coupling signals between neighbor antennas. As a standard two-port S2P file contains the data of the four S parameters that relate the two reflected waves with the two incident waves (S11, S12, S21 and S22), through just one file, it is possible to save four different sets of S parameters. For the previous example, with the file *Sparameters_1,2_2,2.s2p* uploaded, it is possible to save four different sets of values in the structure, because:

- if A1 = 1,2 and A2 = 2,2, they are the file's s12 parameters.
- if A1 = 2,2 and A2 = 1,2, they are the file's s21 parameters.
- if A1 = A2 e A1 = 1,2, they are the file's s11 parameters.
- if A1 = A2 e A2 = 2,2, they are the file's s22 parameters.

In addition to the S parameters, this structure also has a field called *f_old*, which contains the frequency array of the parameters, a field called *fundamental_freq* (fundamental frequency), which is, by default, the center frequency of the parameters, and a field called *f_new*, which corresponds to the subtraction of the frequency array *f_old* with the fundamental frequency *fundamental_freq*. This field is important, because later on it is necessary to analyze these parameters at the baseband.

The fundamental frequency can be changed by the user in the editable text box *Fundamental Frequency (Hz)*, which executes the code of its callback function.

In addition to the *sfiles_info* structure, the values of the structure fields *S* and *f_new* are also stored in the *AntArr* structure, in the form of Map containers. A Map container, in MATLAB, works like a dictionary in Python. It stores data values in a Map object, which is a data structure that associates each value with a corresponding key. For example, executing the function *SParametersCont()* it is created in the antenna *AntArr(2,2)* a field called *sfiles*, which has some keys and the corresponding values, where the keys are the antennas that in the previous structure have a relation with this one and the values are the corresponding S parameters, as illustrated in 3.1.

Table 3.1: S parameters' Map Container of *AntArr* cell (2,2).

Keys	Values
1,2	201×1 double
2,2	201×1 double

It should be noted that the other antennas and the respective values of the S parameters do not appear, because as it is possible to see in the *sfiles_info* structure, figure 3.7, the S-parameters do not exist for the remaining combination, i.e., in this example the files that relate the antenna 2,2, with the antennas 1,1, 2,1, 3,1 and 3,2 were not uploaded.

Following exactly the same procedure, the function *frequenciesCont()* creates a second Map container in which the *f_new* values of the *sfiles_info* structure are stored in the *sfilesfreq* variable of *AntArr* (*AntArr.sfilesfreq*), each one corresponding to its key.

The second way to define the S parameters is to select the *S Parameters Manual* checkbox. By selecting it, it is possible to choose, in the interface's buttons matrix, two antennas (A1 and A2) and then upload a *.s2p* file that relates them. Here, the file does not have a specific name. The *sfiles_info* structure is updated with the new information, and consequently the *AntArr* structure is updated as well.

Another field that is also defined by default when the S parameters are uploaded is the *Distance between elements* edit field. This edit field, which can be changed by the user, is set to 0.5*λ and is used to define the center position coordinates of each antenna in the array. This field executes its callback function, which calls the *antennasPosition()* inside.

The position of the antennas in the array is defined according to the center of the array, which is at position (0,0,0). In the figure 3.8 it is possible to see that according to the chosen number of rows and columns, the center of the array correspond to the center of one antenna, or not.

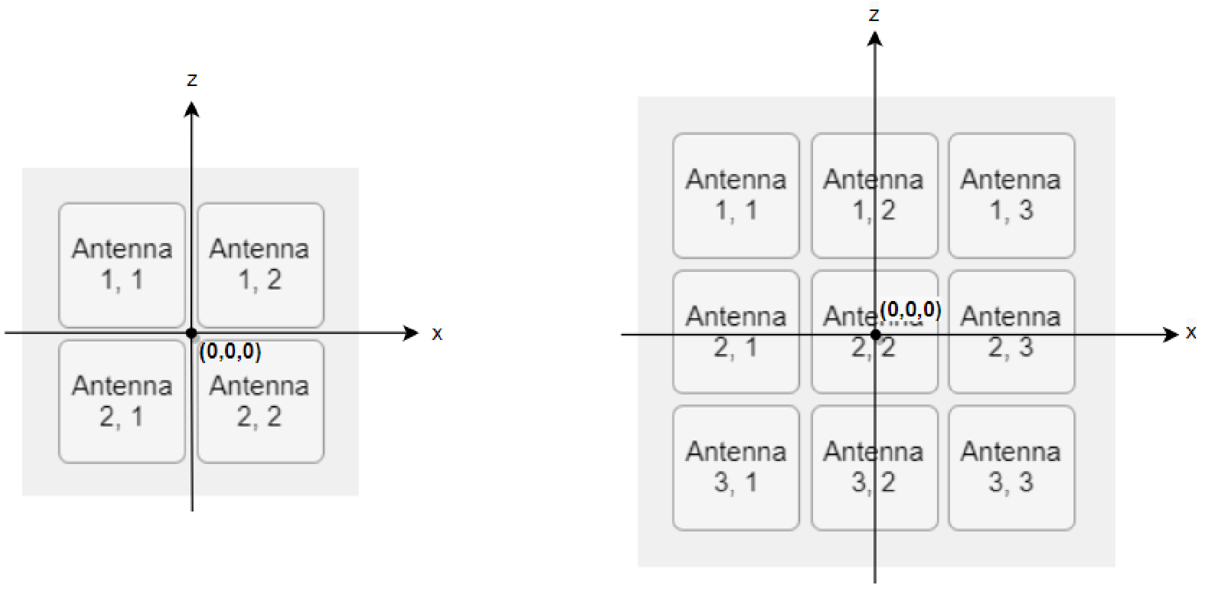


Figure 3.8: Coordinate system definition (the y axis is orthogonal to the x and z axis, following the right-hand orientation rule) and array placement in the reference frame.

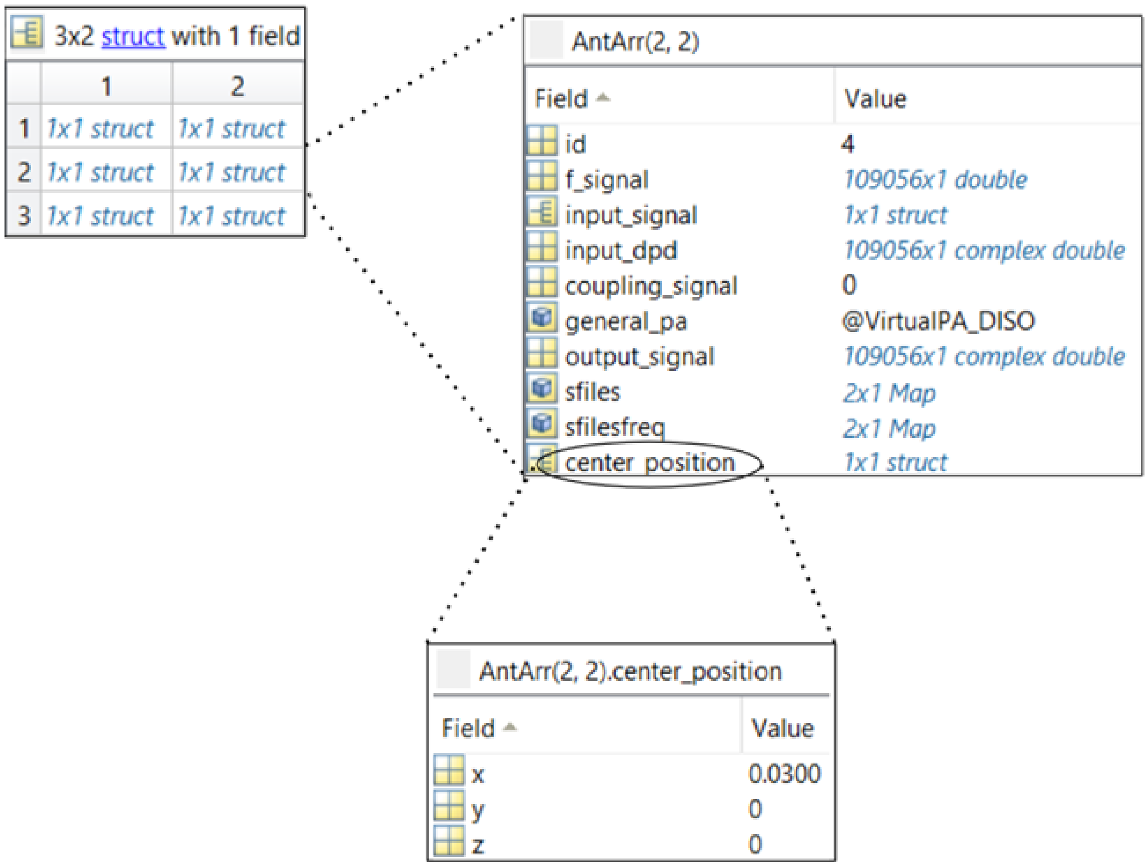
It is assumed that the array is positioned in the $y = 0$ plane and, therefore, the formula defined to calculate the values of the remaining coordinates of the center was:

$$x = dist \times \left(col - \left(\frac{maxCol}{2} \right) - 0.5 \right) \tag{3.4}$$

$$z = -\left(dist \times \left(row - \left(\frac{maxRow}{2} \right) - 0.5 \right) \right) \tag{3.5}$$

where *dist* is the distance between elements defined in the *Distance between elements* edit field, *maxCol* and *maxRow* are the number of columns and rows of the array, respectively, and the *col* and *row* are the value of the column and the value of the row of the antenna whose center is being calculated.

At this point, it can be seen in figure 3.9 that the *AntArr* structure, in addition to the previously defined fields, also has the *sfiles*, *sfilesfreq* and *center_position* fields.



3x2 struct with 1 field

	1	2
1	1x1 struct	1x1 struct
2	1x1 struct	1x1 struct
3	1x1 struct	1x1 struct

AntArr(2, 2)

Field	Value
id	4
f_signal	109056x1 double
input_signal	1x1 struct
input_dpd	109056x1 complex double
coupling_signal	0
general_pa	@VirtualPA_DISO
output_signal	109056x1 complex double
sfiles	2x1 Map
sfilesfreq	2x1 Map
center_position	1x1 struct

AntArr(2, 2).center_position

Field	Value
x	0.0300
y	0
z	0

Figure 3.9: *AntArr* structure after calculating S parameters and the center coordinates of each element.

As the distance was previously defined for 0.5λ and

$$\lambda = \frac{c}{f} \tag{3.6}$$

where c is the speed of light in the air, which is assumed equal to that in vacuum ($2.9979x10^8$ m/s), and f is the fundamental frequency (2.5 GHz, in the case of this illustrative example), the value of the position x will be equal to half the distance between the antennas, which corresponds to the 0.03 value (in meters) shown in the figure 3.9 and z will be zero.

3.4 Iterative method execution

Having already defined the output signals of all elements without considering the coupling between them, as well as the S parameters, it is necessary to solve the iterative method studied in the previous chapter to calculate the interference between neighboring antennas. This interference, after being calculated in each antenna, will correspond to the signal *AntArr.coupling_signal*, which so far is set to zero.

Of the two methods studied in the previous chapter, it was chosen the second one, which applies the equation already explained in the same chapter:

$$Y_{k+1} = Y_k + \lambda\left(G(X, Y_k) - G(X, Y_{k-1})\right) \tag{3.7}$$

All the power amplifiers models used in this method are saved in the *AntArr.general_pa* variable, which takes, as already said, as arguments an input signal, x, and a coupling signal, xc. In this method, the coupling signal will be composed by output signal's, y, past samples.

The button that executes the iterative method in its callback function is, by default, disabled. This button is only enabled after defining the output signals of each individual antenna, as well as the S parameters, as the method only works if these are correctly defined. It is also necessary to change at least one of the two edit fields *Number of iterations* and *Error threshold*. These two fields constitute the stopping conditions of the referred method.

When this method is executed, a first coupling signal is calculated for each antenna. Then, the function stored in the variable *general_pa*, which at first considered the coupling signal to be zero, uses, now, the calculated coupling signal and the equation 3.7 to calculate the new output signal (already considering coupling). This output signal is used to calculate a new coupling signal and this cycle is repeated, until one of the convergence conditions (error threshold or maximum number of iterations) previously defined is met.

The error e_k in each k element is calculated in all iterations and corresponds to the subtraction of the output signal in consecutive iterations of the method:

$$e_k = y_{k+1} - y_k \tag{3.8}$$

being y the output signal. The quadratic error E_k of the signal is then calculated:

$$E_k = e_k^T \times e_k \tag{3.9}$$

where e_k^T is the e_k transposed. Then, the error of the entire array is the sum of all squared errors.

To be able to calculate the coupling signal in each antenna, it is necessary to multiply (in the frequency domain) the output signal of all its neighboring antennas with the set of S parameters that relate these neighboring antennas with the antenna where the signal is being calculated.

To perform this multiplication, it is necessary to have both signals in the desired format. For this, it is necessary to:

- transfer the S parameters to the baseband
- transform the output signals of the elements into the frequency domain
- match the frequency grid of the signals to that of the S parameters to be able to do a point-to-point multiplication.

Transferring the signals to the baseband is trivial, as it consists in subtracting the fundamental frequency to the frequency grid provided in the S2P files (these files already contain the unilateral magnitude and phase of the S parameters). This operation was already done and and the result was stored in the *sfilesfreq* field of *AntArr* structure.

To transform the output signal from the time domain to the frequency domain, the $fft()$ function is used. However, it is also used the function $fftshift()$ that rearranges the Fourier transform by shifting the zero-frequency component to the center of the array. This way, the frequency array is correctly organized.

To move to the third task, an array of an *output_signal* from an antenna and an array of S parameters, both in the frequency domain, are represented in figure 3.10.

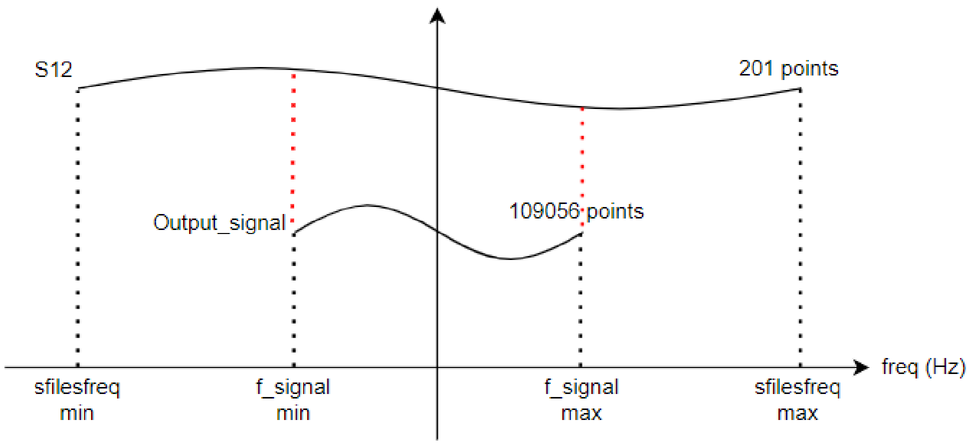


Figure 3.10: Representation of an output signal and an S parameter measurement, both in frequency domain

As it can be seen in the image, the *output_signal* has 109056 points and its frequency range is much smaller than the frequency range of the S parameters. Then, it is necessary to make an interpolation of the S parameters, so that from *f_signal min* to *f_signal max* they have the same number of points as the output signal. To do this, the function *interp*1() was used.

The pseudo code for these last three points is presented below:

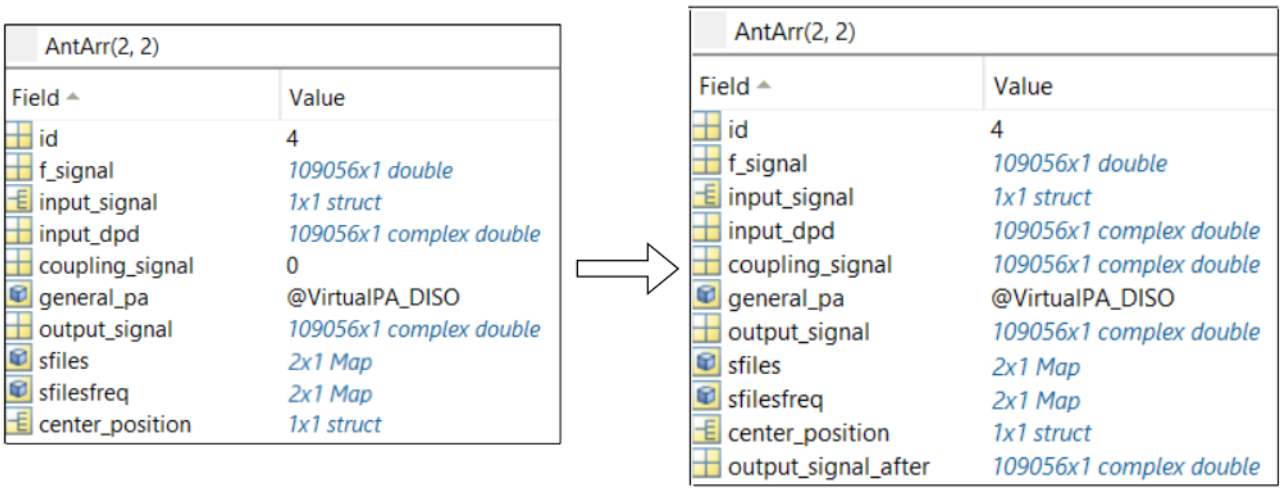
```
N_initial = length(AntArr.sfiles);
N_final = length(AntArr.output_signal);
f_sPar = AntArr.sfilesfreq;

% Interpolation
x = linspace(f_sPar(1), f_sPar(end), N_initial);
x_new = linspace(AntArr.f_signal(1), AntArr.f_signal(end), N_final);
s_param = AntArr.sfiles;
s_param_new = interp1(x, s_param, x_new, 'spline');

% Multiplication of the signals with both arrays in frequency
coup_signal_f = (s_param_new.*fftshift(fft(AntArr.output_signal)/N_final));
```

After this calculation an initial coupling signal, in the frequency domain, is obtained. This signal is then passed back to the time domain with the *ifft* and *ifftshift* functions. This last undoes the result of *fftshift*:

```
    coupling_signal_t = ifft(ifftshift(coup_signal_f)*N_final);
```

Thus, as a summary of the iterative method behavior, the coupling signals of all elements are used by the DISO models of the elements' power amplifiers to recalculate the output signal generated by each element. These will lead to new coupling signals between antenna elements, calculated in the same way as described above. This process is repeated in an iterative way, until convergence is achieved (i.e., until the integrated difference between the signals of two consecutive iterations is smaller than the specified error threshold or the maximum number of iterations is achieved).

At the end of the iterative process, the structure *AntArr* is updated again, this time with the values of the *coupling_signal* calculated for each antenna and with the value of the output signal of each antenna, already considering the coupling signal (*AntArr.output_signal_after*). Figure 3.11 shows the data of the *AntArr* structure for the antenna (2,2) before and after the iterative method.



AntArr(2, 2)

Field	Value
id	4
f_signal	109056x1 double
input_signal	1x1 struct
input_dpd	109056x1 complex double
coupling_signal	0
general_pa	@VirtualPA_DISO
output_signal	109056x1 complex double
sfiles	2x1 Map
sfilesfreq	2x1 Map
center_position	1x1 struct

AntArr(2, 2)

Field	Value
id	4
f_signal	109056x1 double
input_signal	1x1 struct
input_dpd	109056x1 complex double
coupling_signal	109056x1 complex double
general_pa	@VirtualPA_DISO
output_signal	109056x1 complex double
sfiles	2x1 Map
sfilesfreq	2x1 Map
center_position	1x1 struct
output_signal_after	109056x1 complex double

Figure 3.11: *AntArr* structure before and after the iterative method.

3.5 Antennas' radiation diagrams

The next step is to upload the antenna radiation diagram. To do that, the button *Radiation Diagram* should be pressed. This button executes its callback function, which uploads a $.ffs$ file. This version of the implemented simulation platform assumes that all antenna elements are geometrically equal. Therefore, the uploaded radiation diagram is equally applied to each of the antennas in the array. The information with the values of the θ and ϕ angles and the respective values of the electric field, coming from the file, are stored in a structure called *data*.

After uploading the radiation diagram (which is the same for all antennas), the radiation diagram of the antenna array is also displayed. For this, a spherical cloud of points is created and for each point, all the antennas of the array are processed, calculating the angles ϕ and θ of each antenna in relation to each point, as well as the respective magnitude of the electric field. This value is summed for all antennas of the array and thus, repeating this process for all points, the radiation diagram of the array is obtained.

To calculate the angles, the MATLAB function *atan*2() is used. This function follows the convention that $atan2(x, x)$ returns 0 when x is mathematically zero (either 0 or -0). Then, interpolating the values previously stored in the *data* structure, the components of the electric field in ϕ and θ are obtained for each antenna. The interpolation was made with the MATLAB *polyvaln*() function. Having these values, the magnitude of the electric field in each antenna is calculated according to the formula:

$$|E| = \sqrt{|E_\phi|^2 + |E_\theta|^2} \tag{3.10}$$

Next, the pseudocode of this section is presented, where x, y and z are the values of the coordinates of each point subtracted from the values of the central coordinates of each antenna.

```
E_sum = [];
for col = 1:numberOfColumns
    for row = 1:numberOfRows
        phi = atan2d(z,x);                      % Azimuth
        theta = atan2d(sqrt(x.^2+z.^2),y);      % Elevation

        phi(phi<0)=phi(phi<0)+360;              % Move -180:180 to 0:360

        % Interpolation
        E_phi_re = polyvaln(pol_phi_re,[phi theta]);
        E_phi_im = polyvaln(pol_phi_im,[phi theta]);
        E_theta_re = polyvaln(pol_theta_re,[phi theta]);
        E_theta_im = polyvaln(pol_theta_im,[phi theta]);

        E = sqrt(E_theta_re.^2 + E_theta_im.^2 + E_phi_re.^2 + E_phi_im.^2);
        reshape(E);                             % E is now a matrix
    end
    % Sum the E matrices of each antenna
    E_sum = E_sum + E;
end
```

After the radiation diagram and the central coordinates of the antenna array are defined, the opportunity to define the position of a possible receiver is given to the user, and the signal captured by the receiver, at that particular location, is calculated. Thus, there are three editable fields in the GUI that can be modified (figure 3.12).

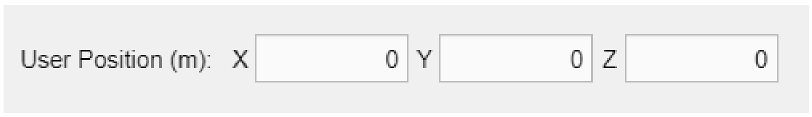

User Position (m): X 0 Y 0 Z 0

Figure 3.12: Edit fields to change the receiver position.

The callback function of these three fields is the same, which means that by changing any of the fields, the same function, called $electricField()$, is executed. In this function, the antenna array is processed so that the azimuth and elevation angles (ϕ and θ, respectively) from each antenna element to the receiver position are calculated, as shown in the figure 3.13. Then, with the interpolation mentioned above, the components E_ϕ and E_θ of the electric field are also calculated.

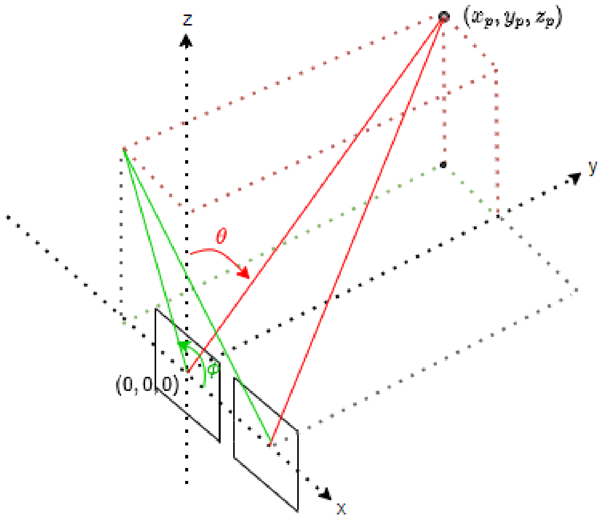


Figure 3.13: Array of antennas with the representation of azimuth, ϕ, and elevation, θ, angles.

In the end, the signal captured by the receiver corresponds to the sum of each antenna's electric field multiplied by the output signal of each antenna, already considering the coupling signal ($AntArr.output\_signal\_after$).

In the next chapter are presented illustrative results of this entire process.

CHAPTER 4

Simulation and results

In this chapter, a simulation of the entire interface is made, as well as some graphical results are presented. The objective is that it serves as a support manual for those who intend to use the platform, getting to know all its features.

To facilitate the organization and the reader's perception, the subtopics presented in this chapter will be the same as those in chapter 3.

4.1 Array general definitions

As already described in the previous chapter, the first step in the GUI is to define the desired configuration for the array by selecting the number of rows and columns of the array in their respective editable fields, *Rows* and *Columns* (figure 4.1). This simulation will be done with a 2x2 antenna array, to go according to the simulated antenna array (in the Simulink tool) presented in chapter 2.

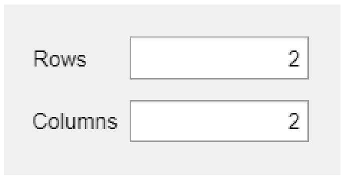
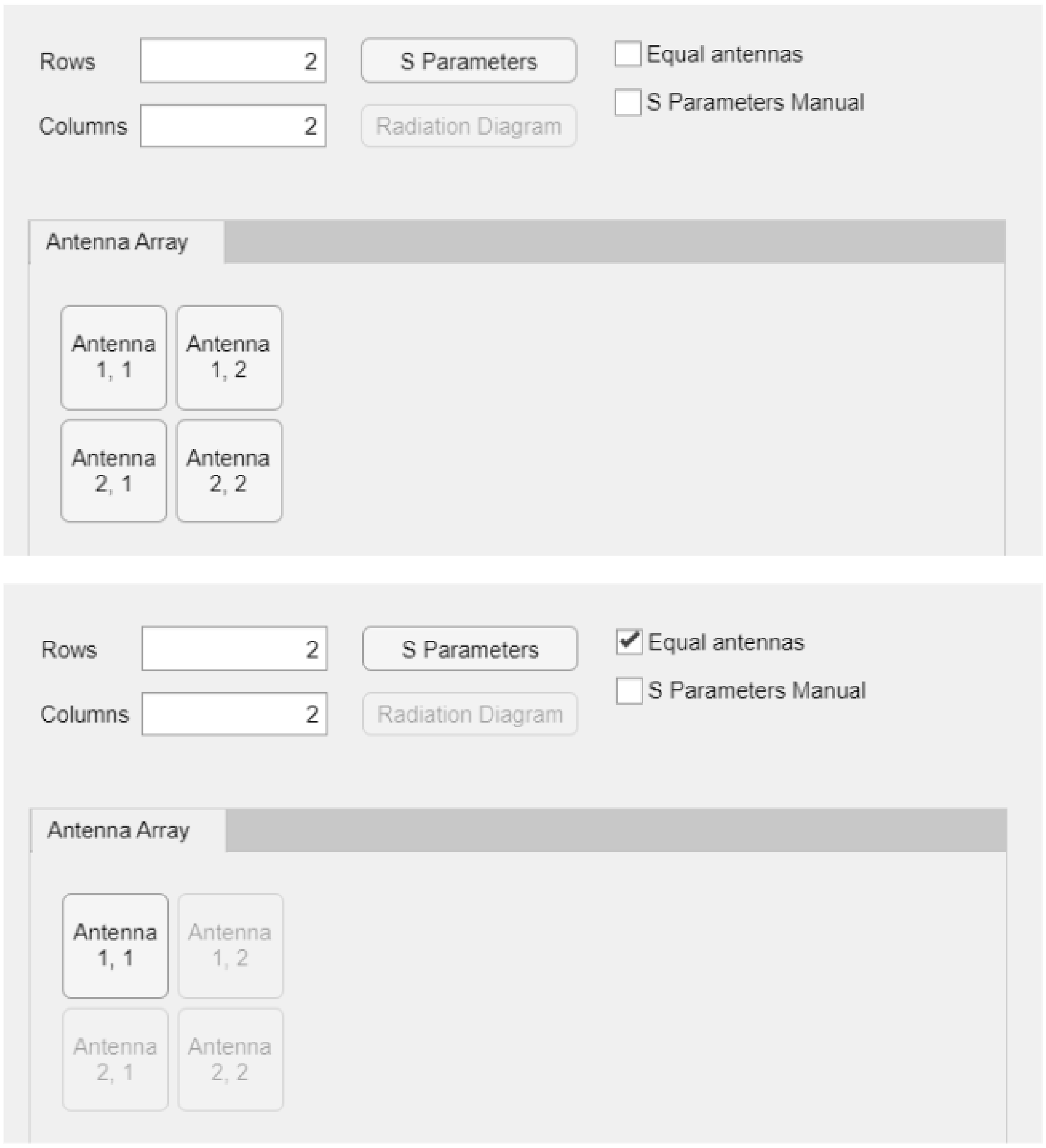

Rows 2
Columns 2

Figure 4.1: Edit fields for an example of 2 rows and 2 columns.

Following this step, the antenna button matrix is created, as well as the rest of the platform components.

The next step is to select the signals of each antenna by clicking on the button of the desired antenna in the buttons' matrix. Note that if the goal is to select the same signals for all antennas, then the *Equal antennas* checkbox should be selected first. By selecting this checkbox, all buttons in the antenna array become disabled except the first one. All signals that are set to this antenna will be assigned to the other antennas as well. In figure 4.2 it can be seen the behavior of the interface by selecting or not the referred checkbox.


Rows 2
Columns 2
☐ Equal antennas
☐ S Parameters Manual

Rows 2
Columns 2
☑ Equal antennas
☐ S Parameters Manual

Figure 4.2: In the top image, the *Equal antennas* checkbox has not been selected, so all buttons are enabled. When it is selected, as it is possible to see in the bottom image, only the first button is enabled.

4.2 Element definitions

After selecting the desired antenna, it proceeds to the definition of its signals. In this case the antenna (1,1) will be selected, because it is considered that the checkbox *Equal antennas* has been selected.

Figure 4.3 shows the interface of the antenna (1,1), where it is possible to select the input signal and the amplifier and predistorter models.

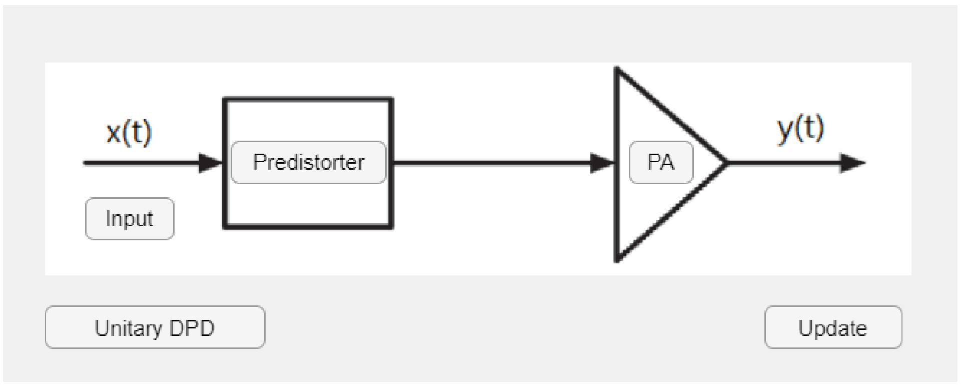


Figure 4.3: Antenna (1,1) interface.

By pressing the *Input* button, the user can select a *.mat* file with two column vectors, one, t, contains the time instants (in seconds) at which each sample of the input signal occurs, and the second, x, has the respective sample values, which are complex-valued numbers, corresponding to the complex envelope of the modulated signal that is intended to be transmitted through the respective element of the array.

After selecting the input signal, a digital predistorter or power amplifier models can be selected by pressing the corresponding button. It should be noted that in all buttons in which a different model is uploaded or removed, the output signal (without considering interactions between antennas) is always calculated, and some graphics are presented.

Let's see an example: selecting a power amplifier model by pressing the PA button, the graphics shown in figure 4.4 were obtained.

After this, the predistorter model suitable for the considered power amplifier was selected — *Predistorter* button. The graphics that are generated when this DPD model was loaded are presented in figure 4.5.

By pressing the *Unitary DPD* button, the visible graphics return to those of figure 4.4, because the DPD is unitary again, as the button indicates.

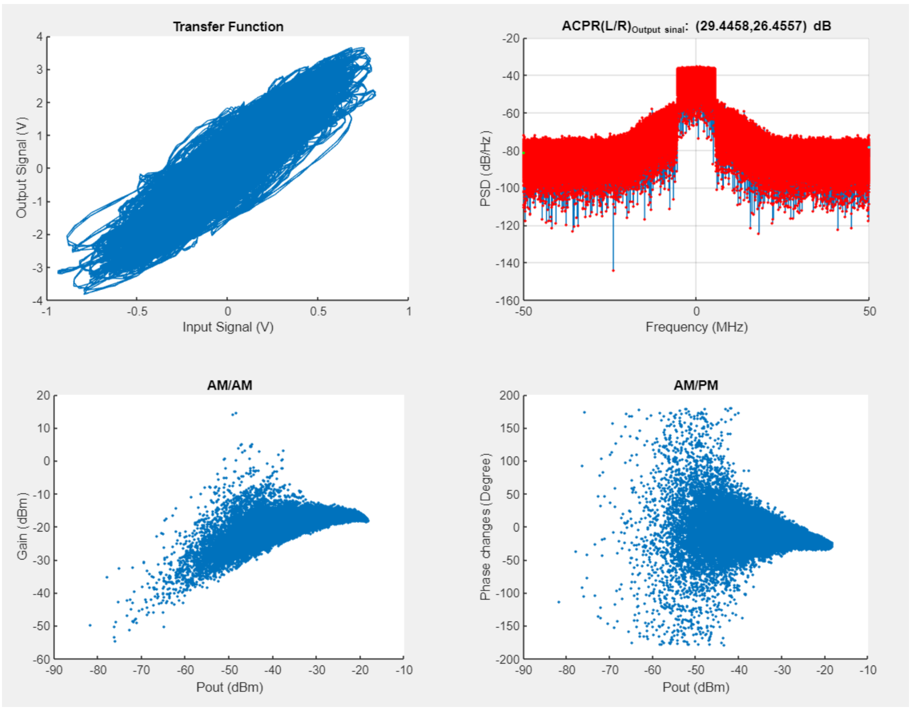


Figure 4.4: Graphics obtained after loading a power amplifier model.

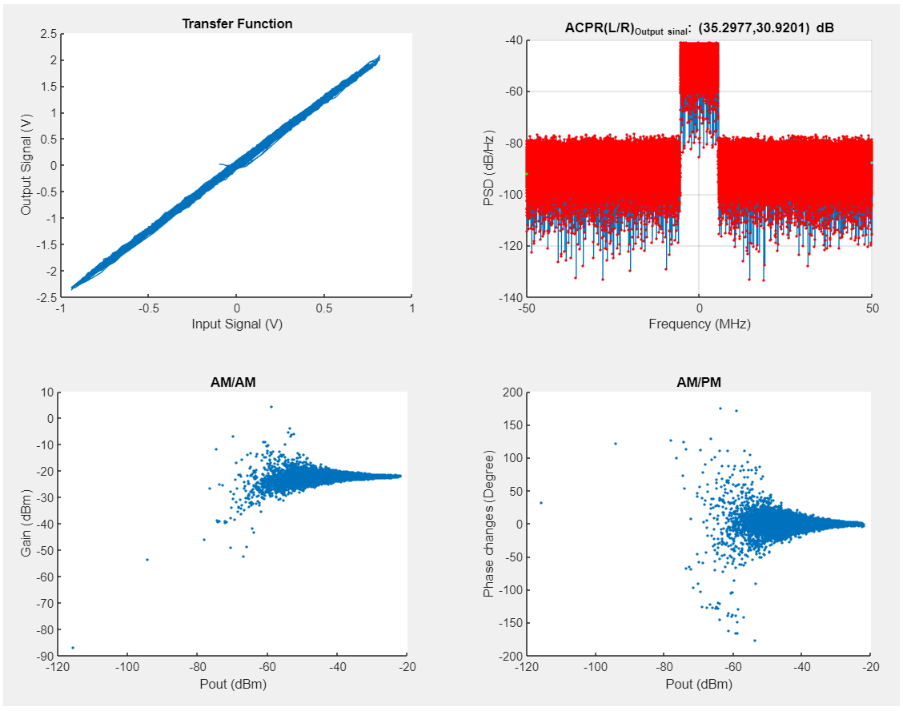


Figure 4.5: Graphics obtained with a power amplifier model and its corresponding digital predistorter.

To continue the simulation, the DPD model was removed, and the PA model selected was the one following the equation 2.7 from the chapter 2.

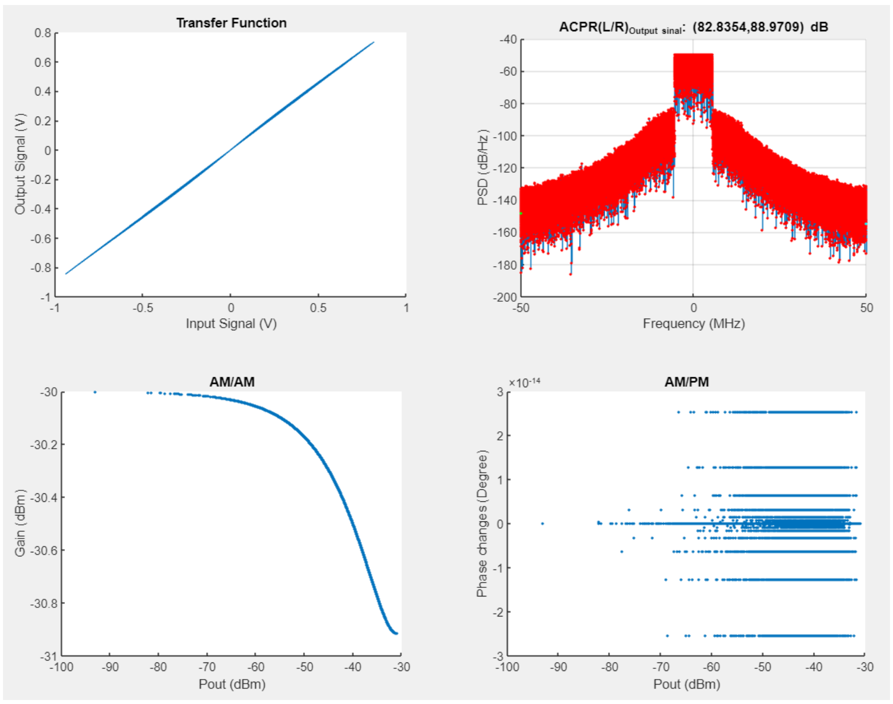


Figure 4.6: Signals obtained for the PA model defined in the chapter 2, equation 2.7.

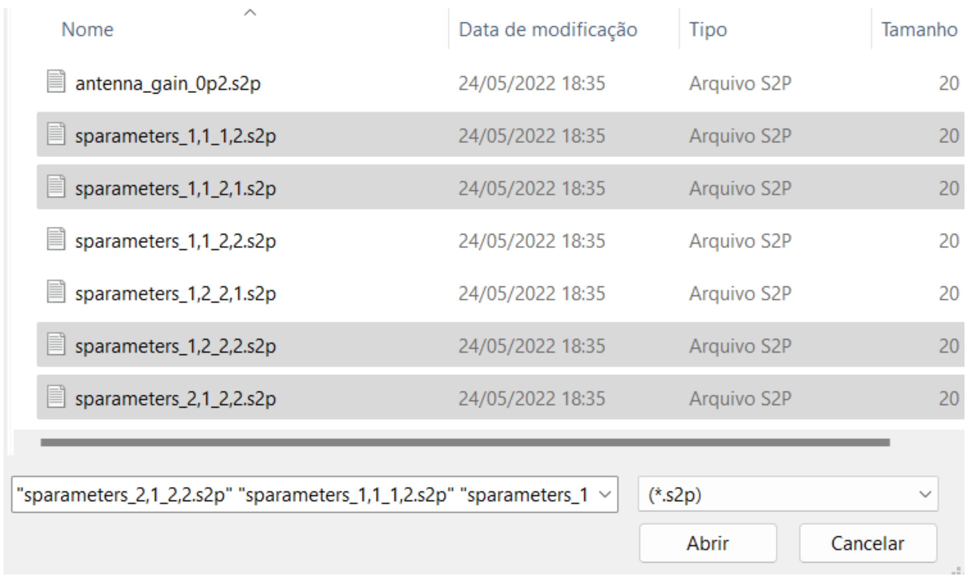
It should be noted that the considered model does not change the phase of the input signal, only the amplitude, and this explains the result of the AM/PM plot.

The last feature of this part of the platform that defines the signals for each element is the *Update* button. Imagine this example, where all antennas were loaded with the same signals (because the *Equal Antennas* checkbox was selected). Considering that after defining the signals, the user closes the page corresponding to the antenna (1,1), and no longer has access to the plots presented above. Then, the user can re-click a specific the antenna and, by pressing the *Update* button, the latest graphics information is updated.

4.3 Antennas' S parameters definition

Once the signals of all the antennas are defined, the selection of the S parameters is next. To select the S parameters there are two options: the *S Parameters* button and the *S Parameters Manual* checkbox — figure 4.2. With the *S Parameters* button, it is possible to define the relationships of several antennas at the same time, by uploading multiple files, while with the checkbox, it is only possible to define the relationship of two antennas at a time. For this reason, it is advisable to start with the button.

So, by clicking the *S Parameters* button, the user can choose multiple files with the format $name\_ant1(1), ant1(2)\_ant2(1), ant2(2).sp2$, as already indicated in the previous chapter. Once again, to fit this simulation with the Simulink's example, the S parameters will be uploaded according to the figure 2.6 of the chapter 2, i.e., it is assumed, in this example, that the electromagnetic coupling between antenna (1,1) and antenna (2,2) and between antenna (1,2) and antenna (2,1) — placed diagonally with respect to each other — are negligible. In the figure 4.7 are the chosen files.



Nome	Data de modificação	Tipo	Tamanho
antenna_gain_0p2.s2p	24/05/2022 18:35	Arquivo S2P	20
sparameters_1,1_1,2.s2p	24/05/2022 18:35	Arquivo S2P	20
sparameters_1,1_2,1.s2p	24/05/2022 18:35	Arquivo S2P	20
sparameters_1,1_2,2.s2p	24/05/2022 18:35	Arquivo S2P	20
sparameters_1,2_2,1.s2p	24/05/2022 18:35	Arquivo S2P	20
sparameters_1,2_2,2.s2p	24/05/2022 18:35	Arquivo S2P	20
sparameters_2,1_2,2.s2p	24/05/2022 18:35	Arquivo S2P	20

"sparameters_2,1_2,2.s2p" "sparameters_1,1_1,2.s2p" "sparameters_1

(*.s2p)

Figure 4.7: Uploaded S parameters' files.

Note, once again, that the files *sparameters_1,1_2,2.s2p* and *sparameters_1,2_2,1.s2p* were not selected.

Imagine that, at a later stage, the user wants to exchange the S parameter file between the antennas 1,1 and 1,2. So, selecting the *S Parameters Manual* checkbox and then both antennas, 1,1 and 1,2, it is possible to choose another *.s2p* file. In this case, the file has no specific name. It could be, for example, the *antenna_gain_0p2.s2p* file in the previous figure.

When the S parameter files are inserted, there are two edit fields filled in by default values: the *Fundamental Frequency (Hz)*, defined with the value of the S parameters' central frequency (it is assumed that all files have the same frequency range); and the *Distance between elements*, set to 0.5λ. Once these values appear, the user can change them.

4.4 Iterative method execution

After defining the signals in all antennas, as well as choosing the S parameters, the implemented iterative method can be executed, to consider the coupling between elements. It should be noticed that for the method to work, all antennas must have their input and output signals correctly defined.

Before pressing the *Coupling Relations* button to start the method, the user must select at least one of the stopping conditions *Error threshold* or *Number of iterations*, clicking on their respective edit fields.

When one of the stopping conditions is reached, the iterative method is terminated and the graphic of the error, as a function of the number of iterations, is displayed on the interface — figure 4.8.

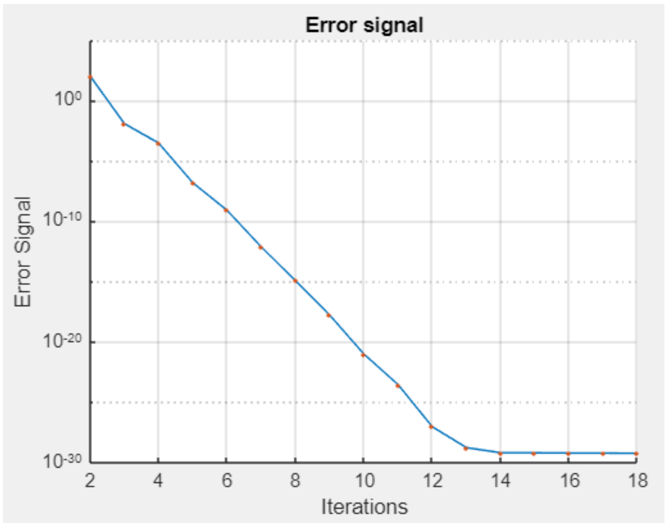
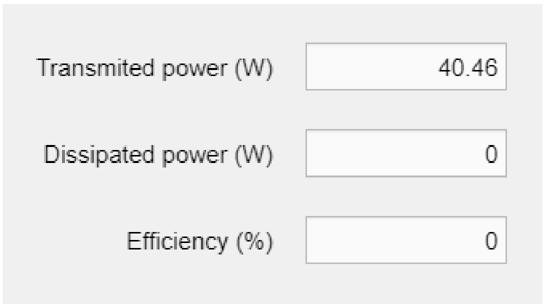


Figure 4.8: Iterative method's error signal.

The error signal unit is volt (V), as it is the subtraction of consecutive output signals from the antennas, which are samples of voltage signals and the y-axis has been plotted on a logarithmic scale, just to see the smoother method conversion. This signal has such a large variation, because its large number of samples was not considered.

As it is possible to see, the error is only considered from the second iteration. This happens because, in the part $G(X, Y_{k-1})$ of the iterative method's equation, in the first iteration, there are no previous samples of the output signal (Y) defined, to use as arguments of the power amplifier functions. So, in the first iteration the Y_k argument already has a value, but Y_{k-1} is zero. It is only on the second iteration that all the necessary values exist, and then the error decreases.

In addition to this graphic, the value of the power transmitted by the array is also displayed in the *Transmited Power* edit field (figure 4.9).


Transmited power (W) 40.46

Dissipated power (W) 0

Efficiency (%) 0

Figure 4.9: Antenna array's transmitted power.

Despite the checkbox label being presented in *Watt* (*W*), in this specific case, the value presented is not in *Watt*, because in this simulation, the amplitudes of the input and output signals were normalized.

The other two edit fields — *Dissipated Power* and *Efficiency* — still do not have any values, because for that, it would be necessary to incorporate a model of the evolution of the current supplied by the power supplies to the amplifiers on the platform and this version still does not consider this model, although its incorporation would be simple to accommodate (it would just require the amplifier model to be a dual input - dual output one, where one of the outputs is the amplifier output envelope signal, and the other would be the baseband current being supplied to the amplifier — this modification would not require any structural change in the developed user interface).

4.5 Antennas' radiation diagrams

Once the iterative method is concluded, the selection of the antenna radiation diagrams begins. This version of the platform assumes that all antenna elements are geometrically equal. Thus, by pressing the *Radiation Diagram* button, the user can select a radiation diagram (file in $.ffs$ format) that will be applied equally to all antennas in the array.

Once the file is selected, the radiation diagram of the antenna array is presented to the user. This diagram is shown in the figure 4.10.

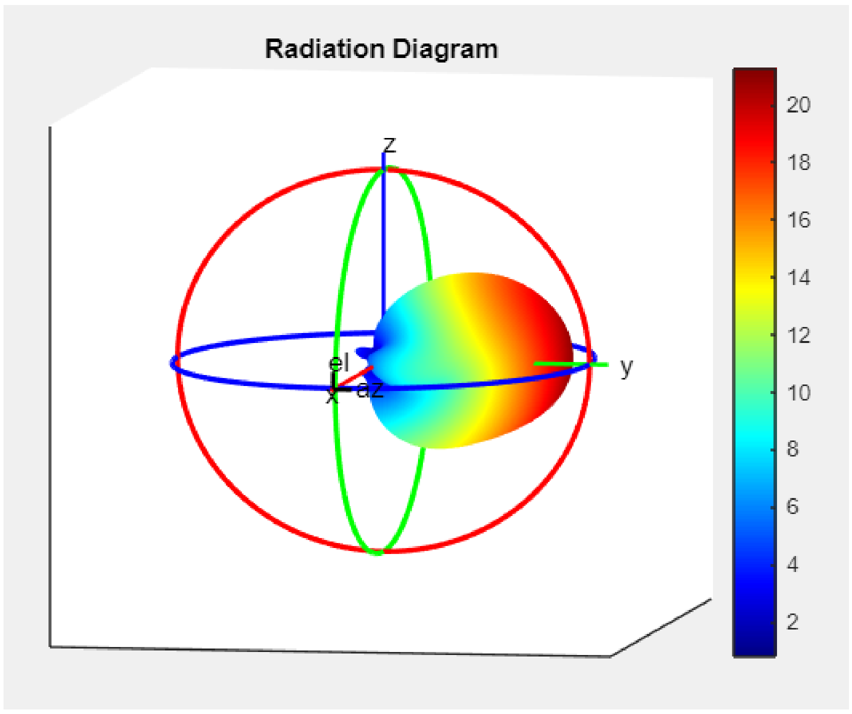


Figure 4.10: Radiation diagram of the antenna array.

The color bar shown in the previous figure correspond to the electric field intensity. It is important to mention that, in this simulator, the effect of distance on the antenna radiation diagram was not considered and, consecutively, the value of the electric field magnitude does not decrease with increasing distance to the array.

Finally, after defining the antenna radiation diagram, the user can also define the position of a possible receiver, changing the edit fields *User Position*, shown in the figure 4.11.

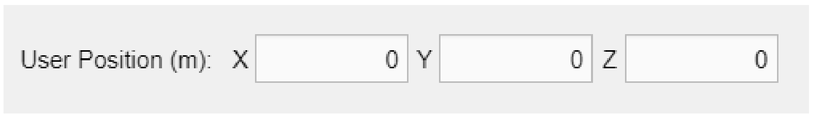


Figure 4.11: Edit fields to change the receiver position.

When having any of the previous edit fields non-zero, for example, changing the Y value to 15, it is displayed the signal that is reaching the receiver at that location (as shown in figure 4.12), whereas the transmitter array is always assumed to be centered with the origin of the reference frame, perpendicularly to the y axis.

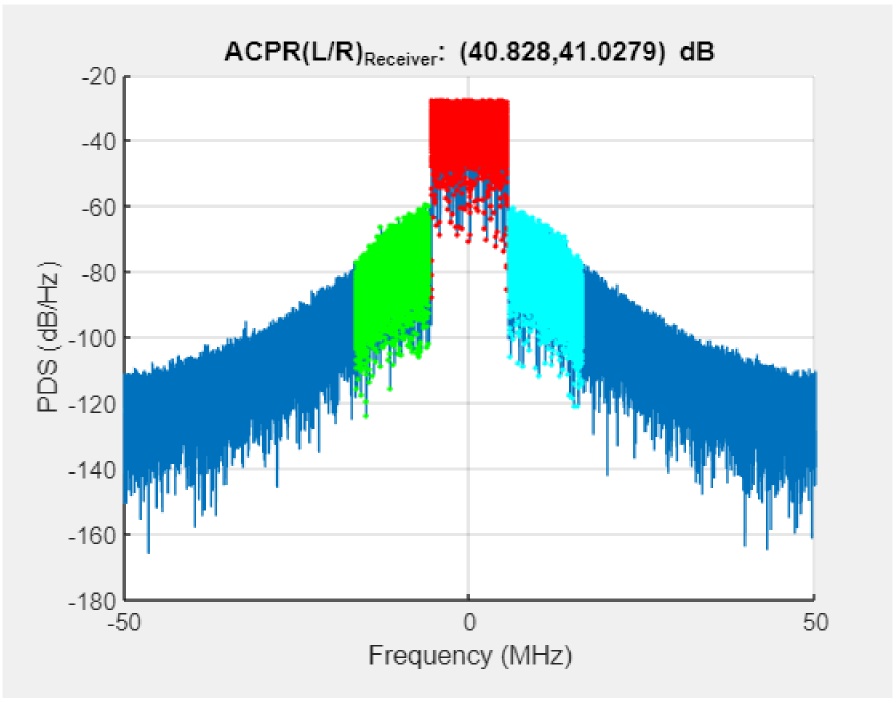
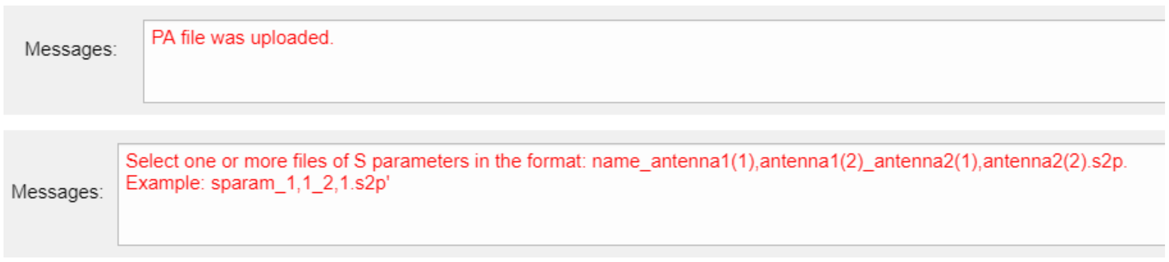


Figure 4.12: Power signal received by the user.

Throughout the entire simulation, there are also two message boxes, one on each page of the interface, where some messages are presented to the user, such as errors or tips. Two examples are shown in figure 4.13.



Messages:

PA file was uploaded.

Messages:

Select one or more files of S parameters in the format: name_antenna1(1),antenna1(2)_antenna2(1),antenna2(2).s2p.
Example: sparam_1,1_2,1.s2p'

Figure 4.13: Examples of messages presented to the user.

The first message is an information that states that the power amplifier model was correctly uploaded. The second one refers to an error, in which the *SParameters* button is selected, but then no file is chosen. In appendices C and D is a view of the entire interface, with the results obtained.

CHAPTER 5

Conclusions and future work

In this work it was developed a transmission system simulator in the platform MATLAB, with the App Designer tool. Its goal is to obtain estimates of the system behavior based on mathematical models, that mimic the baseband behavior of the amplifiers behind each antenna of the array transmitter, operating those models from the system-level perspective. Consequently, a simulation at the electrical circuit level is avoided whenever a change in the system is required, for example, the signal to be transmitted, or to test different amplifiers and/or antennas in the design stage of an array transmitter.

This platform has several features, starting with the possibility of defining the size of the antenna array, as well as the input signals and the models of their amplifiers and predistorters. It also considers the inclusion of the antennas' S parameters, as well as their radiation diagrams. Through this, it is possible to calculate the coupling effects between the elements, as well as to obtain graphics for each individual antenna and for the antenna array. All this achieved in a user-friendly way. In the previous simulation, the iterative method ran in 2.83 seconds (with all the signals having more than one hundred thousand samples), also showing that the application is quite fast and efficient.

Despite all the features that are already considers, it would be important, for future work, to upgrade the platform to incorporate an evolution model of the current supplied by the power supplies to the amplifiers. As such, it would be possible to estimate the total power consumed by the array and, based on the power transmitted by the antennas, estimate the dissipated power and overall energy efficiency (very important in assessing the array's operation). Figure 4.9 presents two edit fields, *Dissipated power (W)* and *Efficiency (%)*, which, despite already being in the interface, are not yet capable of presenting values, without this upgrade.

In addition to this, it would also be important to consider the effect of distance on the antenna radiation diagram and on the calculation of the signal received by the user. Both of these features were implemented for a normalized fixed distance and, consequently, the value of the magnitude of the electric field does not decrease if the distance to the array increases.

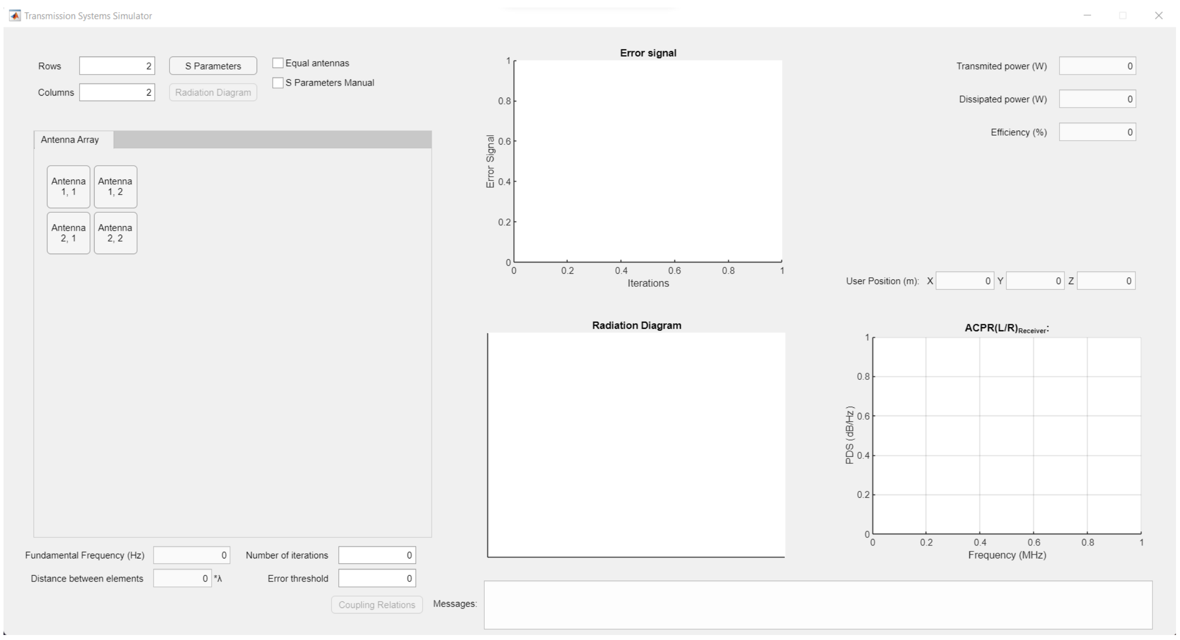
Another functionality that would be important to implement is the calculation of the coefficients of the DPD model that best fit a certain PA. In the simulator it is already possible to include DPD models, but it would be important to be able to calculate the coefficients of a given DPD model which would lead to the best linearization of the respective amplifier.

Although the application was developed with speed and efficiency in mind, having already been highly optimized, it is always possible to further optimize certain algorithms. With the addition of new functionalities, optimization must always be a priority, so as to maintain the application response time viable.

Appendices

APPENDIX A

Transmission System Simulator



Rows 2

Columns 2

Transmited power (W) 0

Dissipated power (W) 0

Efficiency (%) 0

User Position (m): X 0 Y 0 Z 0

Fundamental Frequency (Hz) 0

Number of iterations 0

Distance between elements 0 *λ

Error threshold 0

Figure A.1: Transmission System Simulator.

APPENDIX B

Antenna (1,1) interface

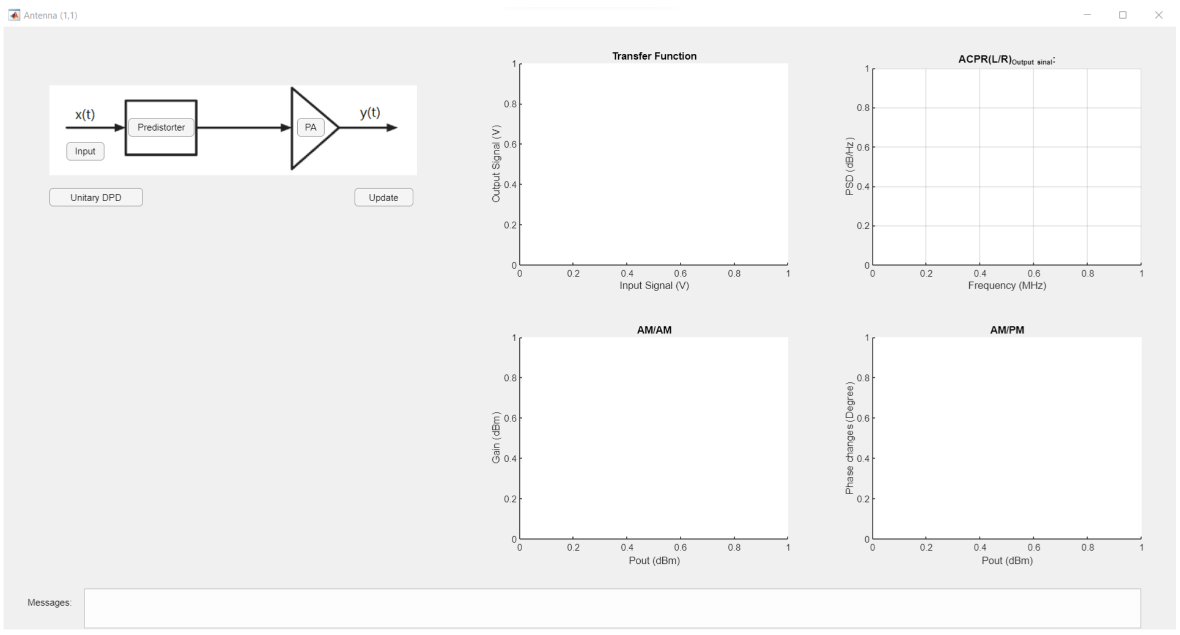
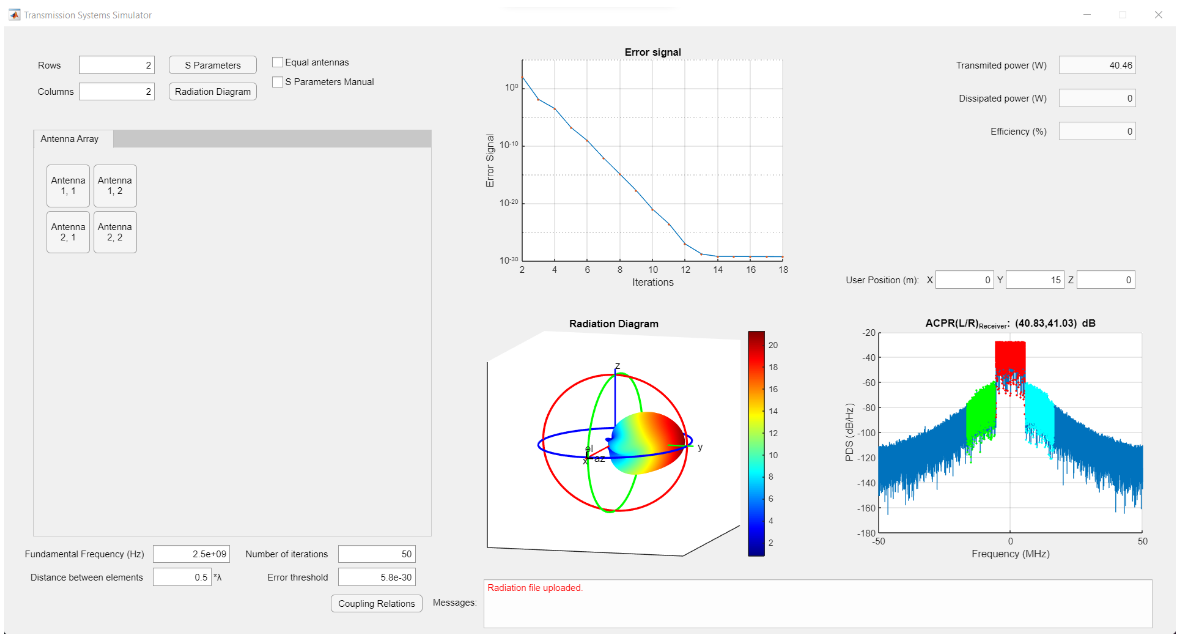


Figure B.1: Antenna's (1,1) interface.

APPENDIX C

Transmission System Simulator with results


Transmission Systems Simulator

Rows: 2
Columns: 2
☐ Equal antennas
☐ S Parameters Manual

Fundamental Frequency (Hz): 2.5e+09
Distance between elements: 0.5 *λ
Number of iterations: 50
Error threshold: 5.8e-30

Transmited power (W): 40.46
Dissipated power (W): 0
Efficiency (%): 0

User Position (m): X 0 Y 15 Z 0

ACPR(L/R)$_{Receiver}$: (40.83,41.03) dB

Messages: Radiation file uploaded.

Figure C.1: Transmission System Simulator with the results presented in chapter 4.

APPENDIX D

Antenna (1,1) interface with results

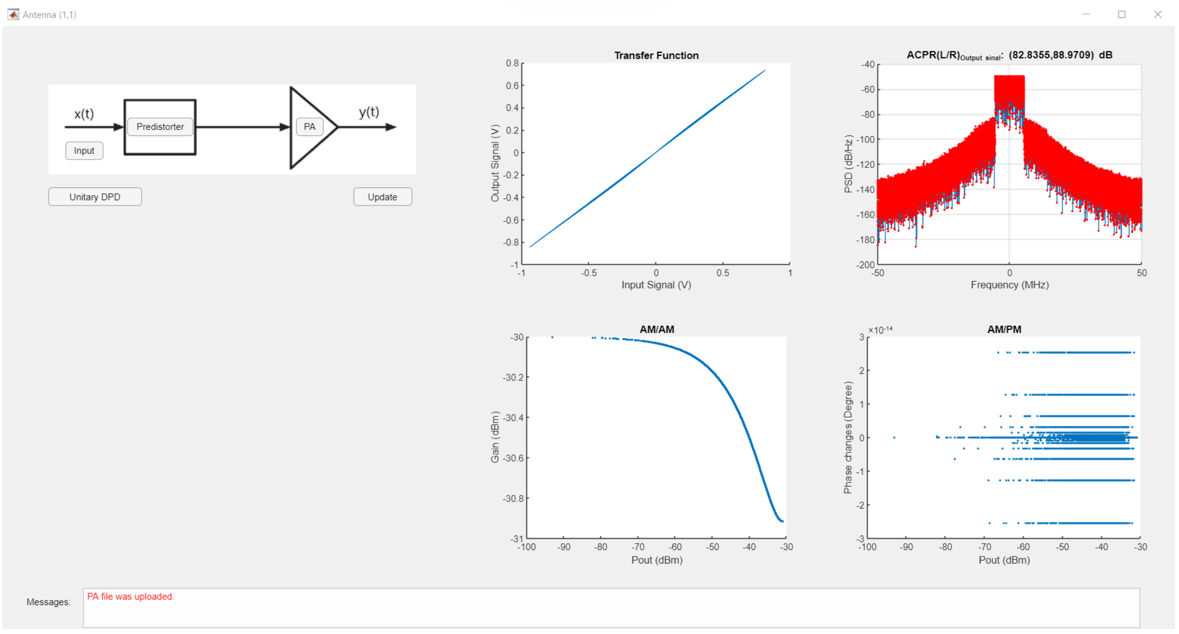


Figure D.1: Antenna's (1,1) interface with the results presented in chapter 4.

References

[1] C. Fager, T. Eriksson, F. Barradas, K. Hausmair, T. Cunha, and J. C. Pedro, “Linearity and efficiency in 5G transmitters: New techniques for analyzing efficiency, linearity, and linearization in a 5G active antenna transmitter context,” *IEEE Microwave Magazine*, vol. 20, no. 5, pp. 35–49, 2019. DOI: 10.1109/MMM.2019.2898020.

[2] J. A. Zhang, X. Huang, V. Dyadyuk, and Y. J. Guo, “Massive hybrid antenna array for millimeter-wave cellular communications,” *IEEE Wireless Communications*, vol. 22, no. 1, pp. 79–87, 2015. DOI: 10.1109/MWC.2015.7054722.

[3] F. M. Barradas, P. M. Tomx00E9; J. M. Gomes, T. R. Cunha, P. M. Cabral, and J. C. Pedro, “Power, linearity, and efficiency prediction for mimo arrays with antenna coupling,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 65, no. 12, pp. 5284–5297, 2017. DOI: 10.1109/TMTT.2017.2766067.

[4] P. M. Lavrador, T. R. Cunha, P. M. Cabral, and J. Pedro, “The linearity-efficiency compromise,” *IEEE Microwave Magazine*, vol. 11, no. 5, pp. 44–58, 2010. DOI: 10.1109/MMM.2010.937100.

[5] N. Ojaroudiparchin, M. Shen, and G. Pedersen, “A 28 ghz fr-4 compatible phased array antenna for 5G mobile phone applications,” in *2015 International Symposium on Antennas and Propagation (ISAP)*, ser. International Symposium on Antennas and Propagation (ISAP). Proceedings. IEEE, 2015, ISBN: 978-4-8855-2303-8.

[6] N. Ojaroudiparchin, M. Shen, S. Zhang, and G. Pedersen, “A switchable 3d-coverage phased array antenna package for 5G mobile terminals,” English, *I E E E Antennas and Wireless Propagation Letters*, vol. 15, pp. 1747–1750, Feb. 2016, ISSN: 1536-1225. DOI: 10.1109/LAWP.2016.2532607.

[7] C. Fager, K. Hausmair, K. Buisman, K. Andersson, E. Sienkiewicz, and D. Gustafsson, “Analysis of nonlinear distortion in phased array transmitters,” in *2017 Integrated Nonlinear Microwave and Millimetre-wave Circuits Workshop (INMMiC)*, 2017, pp. 1–4. DOI: 10.1109/INMMIC.2017.7927314.

[8] H. T. Hui, “Decoupling methods for the mutual coupling effect in antenna arrays: A review,” Department of Electrical and Computer Engineering, National University of Singapore, 10 Kent Ridge Crescent Singapore 119260, Singapore, 2007.

[9] A. Ben-Israel, “A newton-raphson method for the solution of systems of equations,” *Journal of Mathematical Analysis and Applications*, vol. 15, no. 2, pp. 243–252, 1966, ISSN: 0022-247X. DOI: https://doi.org/10.1016/0022-247X(66)90115-6. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0022247X66901156.

[10] J. Chani-Cahuana, P. N. Landin, C. Fager, and T. Eriksson, “Iterative learning control for rf power amplifier linearization,” *IEEE Transactions on Microwave Theory and Techniques*, vol. 64, no. 9, pp. 2778–2789, 2016. DOI: 10.1109/TMTT.2016.2588483.