Trabajo de Fin de Grado
Ingeniería de Telecomunicación

Modelado de Comportamiento de Amplificadores de Potencia mediante Hill Climbing

Autor: Rafael García Noguer

Tutor: Juan Antonio Becerra González

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Departamento de Teoría de la Señal y Comunicaciones

Trabajo de Fin de Grado
Ingeniería de Telecomunicación

Modelado de Comportamiento de Amplificadores de Potencia mediante Hill Climbing

Autor:

Rafael García Noguer

Tutor:

Juan Antonio Becerra González

Profesor Sustituto Interino

Dpto. Teoría de la Señal y Comunicaciones
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla

Sevilla, 2020

Trabajo de Fin de Grado: Modelado de Comportamiento de Amplificadores de Potencia mediante Hill Climbing

Autor: Rafael García Noguer
Tutor: Juan Antonio Becerra González

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

Acknowledgement

Watashi no kazoku no tame ni

Rinko Kikuchi as Mako Mori, Pacific Rim (2013)

A mi familia, a Claudia y a mis hermanos de la ETSI.

Rafael García Noguer
Sevilla, 2020

Resumen

En este trabajo de fin de grado se presenta una introducción teórica breve sobre los amplificadores de potencia y sus no linealidades, así como la implementación del algoritmo Hill Climbing aplicado al modelado de comportamiento basado en series de Volterra de los amplificadores de potencia. A continuación, se presentan una serie de experimentos con el objetivo de estudiar las virtudes y los defectos de este tipo de algoritmo de búsqueda exhaustiva. Para los experimentos, se han utilizado muestras de señales 4G-LTE y 5G-NR, así como amplificadores operando entre -30 y -20 dBm.

Abstract

In this final degree work, a brief theoretical introduction about power amplifiers and their non-linearities followed by an implementation of the Hill Climbing algorithm applied to the behavioral modeling based on Volterra series of power amplifiers are presented. Some experiments aiming at studying the strength and weaknesses of this type of exhaustive search algorithm are also presented. For the experiments, 4G-LTE and 5G-NR samples, as well as amplifiers operating in the range of -30 to -20 dBm, have been used.

Contents

1 Introduction

To accept anything on trust, to preclude critical application and development, is a grievous sin; and in order to apply and develop, simple interpretation is obviously not enough.

Vladimir Ilyich Ulyanov

The rapid development of wireless communications has bring the necessity of new more-exquisitely designed devices which must be usually smaller and thinner but also must work with more complex modulations, shorter time responses and, of course, cheaper. Cellular coverage has also been required to be wider and access points to be invisible. The rising awareness about climate change has also put mobile communications in the spotlight: wireless communications must be energy efficient.

Power amplifiers (PAs) are responsible for a big part of the power consumption of wireless devices. They are active devices, so they require a power supply to function. They are present in cellular base stations, radars, planes, radio and television broadcast, sensor networks, audio players, etc. PAs are usually two-port devices which transfer an input signal from the input port to the output port increasing the signal amplitude. The difference between the output and the input signal amplitude is denoted as gain.

How efficiently amplifiers deliver this gain is key to reduce the energy waste in telecommunications. PAs are more efficient when they are put to work where the output power is close to the maximum supply power. This is known as the saturation zone, after which its behavior becomes nearly unpredictable and no higher output power is obtained. Part of the energy consumed by PAs will be transformed into heat escaping the enclosure of the apparatus. Some also will amplify non-fundamental spectrums of the input signal which may be undesired. PAs should operate linearly depending on the power it is supplied with, but this leakage of power is considered non-linear.

Progress in modulation techniques has brought new schemes such as Orthogonal Frequency Division Multiplexing (OFDM) which are featured by a high peak-to-average power ratio (PAPR). This magnitude is defined as the peak power of the modulated signal to the average power of the modulated signal. Signals with a constant envelope have a zero PAPR and are inmune to non-linear distortion, higher-than-zero PAPR increases the signal sensitivity to it [1].

Digital pre-distortion (DPD), crest factor reduction (CFR), envelope tracking (ET) and many other techniques are used to mitigate the PAPR as well as correct other non-linearities. The key of the DPD solution is the dynamic identification of the PA behavior and the consequent development of a PA model which resumes the nonlinear effects produced by the components of the amplifier such as nonlinear reactances in its transistors or charge storaged in its capacitors.

The relationship between the input and output signals is used to build a *behavioral model* that not necessarily need to be related to the physical phenomena that take place inside the system. Thus, this mainly exclusive mathematical relationship can result in a relatively simple model through techniques such as data-fitting.

Fitting is one of the key activities in modeling and consists on finding the parameters of a function to represent the data. For behavioral modeling, the physical relationship between inputs and outputs may not be needed. Polynomial functions can be adjusted in order to obtain acceptable-accuracy models. This polynomial it is achieved through error minimization, usually considering error the difference of distance between the data and the function result. These models can be very successfully created with relatively few parameters.

There are other computational models such as neural networks (NN), which have become very popular in the last few years due to the belief which says NN can approximate any nonlinear function. These machine-learning networks are groups of linear functions or *neurons* with common weighted inputs which output passes through a nonlinear function called activation function that modifies it, so it does not exceed certain thresholds. These networks can improve their performance thanks to a loss function and backforwarding, which evaluate and steadily correct the neurons coefficients during machine fitting. These neurons can be joint into layers put one after the other, usually smoothing the approximation of the objective function.

Another type of computational models very popular these days are genetic algorithms (GAs). GAs are also part of the machine-learning family of algorithms. They start calculating the fitness of a initial population of model solutions, randomly or partially randomly generated, running a selection of only the fittest models and generating a new population or generation by recombining the original fittest models or parents. This process shall repeat until the local optimum is found. As these algorithms are based on Darwin's theory of evolution, mutations are considered during the generation of every population. Mutation and recombination make the algorithm strong against premature convergence or plateauing, showing diversity as the best way to find the best solution.

The simulator that will be used in this final degree work will be MATLAB, a well-known numerical computing environment with his own programming language and available for numerous operating systems. It is a common tool used in the practical lessons of Seville's Engineering School and in its B.Sc. in Telecommunication Engineering.

Chapter 2 offers a brief state-of-the-art review about PAs and their non-linear behavior and Chapter 3 comes after the implementation of the Hill Climbing Heuristic presented in [2] and contains different experiments and analysis about this algorithm. Chapter 4 includes all the scripts of the implementation of the algorithm. Finally, Chapter 5 shows the conclusions and future lines of research this work could continue with and Chapter 6 holds the MatLab code which were needed for this work.

2 Nonlinearities in Power Amplifiers

To be radical is to grasp things by the root.

KARL MARX

Power amplifiers are electronic devices used to increase the magnitude voltage, current or power. In communications, the job of radio frequency (RF) PAs is to boost the modulated level so that the receiver is capable to detect the signal level. PAs must be capable of delivering a big amount of power to a load without losing efficiency while taking low-power input signal to a high-power output signal.

Amplifier designs in general are usually simple adaptations from linear system designs, but we must differentiate between small and large-signal amplifiers. Power amplifiers (PAs) are large-signal amplifiers that work with higher output power levels than small-signal ones. Small-signal approximations are generally not appropriate for the PAs [3, p. 551].

2.1 Efficiency is money

Efficiency is clearly one of our top priorities in order to design a PA. It is now time to learn more about these devices which have the leading role of this final degree work. PAs have been classified along the years. The alphabetical classification has been used for nearly a century back to the early ages of electronics [4, p. 159]. The most common classes will be described later in this work.

2.2 Class A-to-F PAs

Before describing the different classes of amplifiers, a short explanation about the operation modes of the amplifiers must be given. PAs can generally work in three different modes as follows: linear mode, when operating in a highly linear portion of their static characteristic curve; critical mode, when current ceases to flow but operation extends beyond the linear portion up to the saturation and cutoff regions; and non-linear mode, when current ceases to flow during a portion of each cycle [5, p. 4]. PA transistors must be provided with enough dc voltage, so it is maintained in the active region, where it works as an amplifier. This is the biasing condition. Modern communication systems perform at the edge of the linear portion of their characteristic curve, PA architectures must make sure maximum peak power is in range of this region and manage the impact this will have in efficiency [6, p. 7].

This classification starts with the Class A amplifier. This device is strongly linear when operating in the exact mid-point between the saturation and cutoff regions [7, p. 18]. However, this is essentially for smaller signal amplitudes, having only an energy efficiency of 50% at peak power and being considered too inefficient for cellular wireless communications. Class B amplifiers are widely used because of their low standby dc current requirements. A Class B PA transistor only conducts over half the period of the input signal, so two are used to entirely amplify the input signal [3, p. 535]. They still only have a maximum efficiency of about 78.6% and a very high biasing voltage. Class B PAs are also common for showing a type of distortion called crossover distortion, which bring non-linearities to the output. Class AB amplifiers are halfway between Class A and B amplifiers, with a low bias condition as in Class A but conducting only half-cycle like in Class B amplifiers. Theoretical efficiency can be over 70%. Class C is a high efficiency, low gain Class B amplifier

which presented no problem with distortions on constant envelope signals such as FM transmissions. Class D, E and F amplifiers, also denoted as switching amplifiers, are more efficient than the previous classes. They get close to 100% with the correct configuration, although they are strongly affected by harmonics. This problem requires of an effective tuning to avoid significant power losses. These amplifiers need to work close to the maximum output power to provide with the peak efficiency, and still they will be vulnerable to power back-offs [8, pp. 21-31]. A diagram which compares the main classes of amplifiers from A to T is in Figure 2.1.

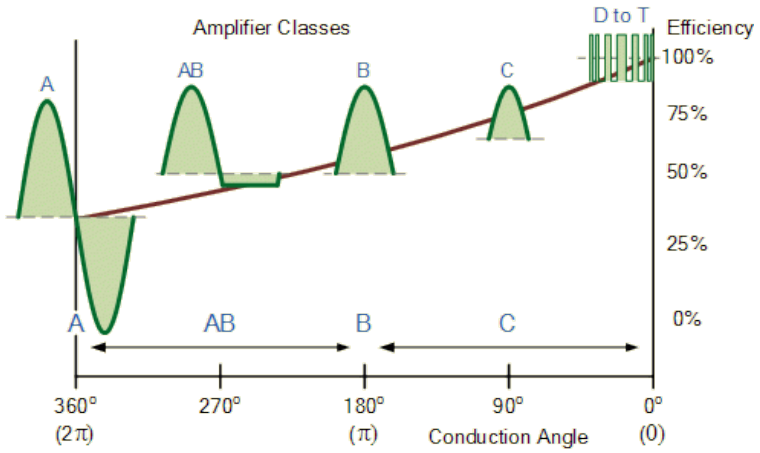

Figure 2.1 Comparison of the main classes of amplifiers [9].

2.3 More complex architectures

After shortly reviewing the main classes of amplifiers, it is evident that high efficiency comes at the prices of complexity. These will make difficult the task of the PA designer. More complex architectures or techniques are needed to ensure that our RF PA efficiency is high. Architectures and efficiency enhancement techniques excel among many and they will be explained briefly below:

2.3.1 Doherty PA

The Doherty PA (Figure 2.2) was proposed in the 1935 with the purpose of maintaining high efficiency over a wide range of input voltage of a linear power amplifier. It is built out of 2 amplifiers, one main power amplifier and an auxiliary power amplifier. The first one saturates at high input power losing gain, so that the second one switches on when this occurs. This makes possible that the amplifier is still efficient even 6-8 dB from the peak output power [7, p. 291] [8, p. 33] [10, p. 333].

2.3.2 Envelope Tracking

Envelope tracking technique goal is the same as for the Doherty PA. It works as a type of envelope elimination and restoration technique, although without a limiter amplifier. It consists of two paths: one has an envelope

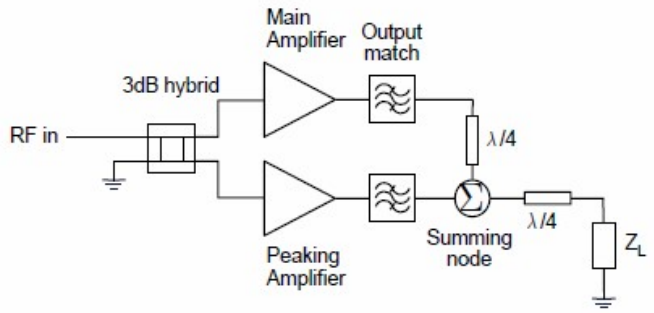

Figure 2.2 Doherty PA block diagram [8].

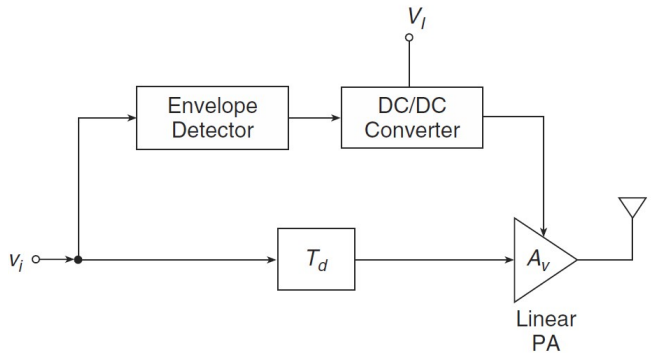

Figure 2.3 Block diagram of a PA with envelope tracking [10].

detector and a dc-dc switching-mode power converter, the other one has one delay block and a linear amplifier (Figure 2.3). The envelope detector obtains the amplitude modulation voltage out of the input voltage, and with the dc-dc converter this voltage is used to adjust the level of supply voltage of the linear amplifier. The efficiency of the amplifier increases with the envelope magnitude, so driving it into compression gives the highest efficiency. [7, pp. 311-314] [8, p. 36] [10, pp. 330-333].

2.4 Nonlinearities

After achieving high-efficiency devices, PAs still present some deficiencies which need to be handled somehow. This efficiency comes at the cost of operating in compression or even in saturation, which are very nonlinear regimes. That will lead to harmonic distortion, intermodulation products and the consequent power leakage. This happens because, even using linear or small-signal approximations, the physical components of the PA are not linear themselves: there will always be some level of distortion [8, p. 112].

2.4.1 1dB-compression point

The 1dB-compression point marks the beginning of the nonlinear operation region, as it is where the measured output power of a PA differs one decibel from the theoretical, linear and desired behavior. That nonlinear behavior is also known as amplitude modulation to amplitude modulation (AM-to-AM) conversion: as a modulation of the input signal nonlinearly modulates the output signal. Additionally, an amplitude modulation to phase modulation (AM-to-PM) conversion will occur: the input components of the amplifier start to show drive dependency and alter the phase of the amplifier gain from its linear value [7, p. 139] [8, pp. 112-125].

In Figure 2.4, it is clearly shown how the PA behavior stops from being linear when the slope of the curve which represents the relationship between the input and the output saturates. In this example, the amplifier enters a nonlinear region when it reaches roughly -35dBm of output power. The Gain-to-AM (Figure 2.5) plot is commonly represented alongside of the AM-to-AM plot. This graph quickly exhibit how the gain of the PA goes down as the input power increases.

2.4.2 Harmonic distortion

Harmonics or the harmonic content are frequency components placed at integral multiples of the fundamental frequency of a signal. Total harmonic distortion (THD) has two similar definitions. It basically consists on the ratio of power of a signal part of the harmonic content to the power of the full signal or the fundamental component of the signal, always expressed as a percentage [11, pp. 471-477]. Intermodulation distortion (IMD) is the nonlinear mixing of fundamental and harmonic components of the signal, producing new frequency components in the spectrum; cross-modulation distortion, produces new frequency components that already exists [8, p. 131].

Harmonic distortion makes the PAs add undesired nonlinearities which brings the necessity of nonlinear modeling and characterization. One of the biggest concerns when transmitting a RF signal is that it must not spread to adjacent channels. Wireless transmitters' PAs cause distortion in ways such as spreading the bandwidth out into adjacent channels. In this case, harmonic distortion causes the signal to replicate in harmonic frequencies of the carrier and intermodulation adds a spectral regrowth to the signal, spreading out

the bandwidth. Harmonic distortion usually can be dealt with bandpass filters, but the amplitude overlap produced by IMD cannot be solved with filters. There are legal regulations and standards for this signal spreading called spectral regrowth and for adjacent-channel interference, so it is an imperative requirement

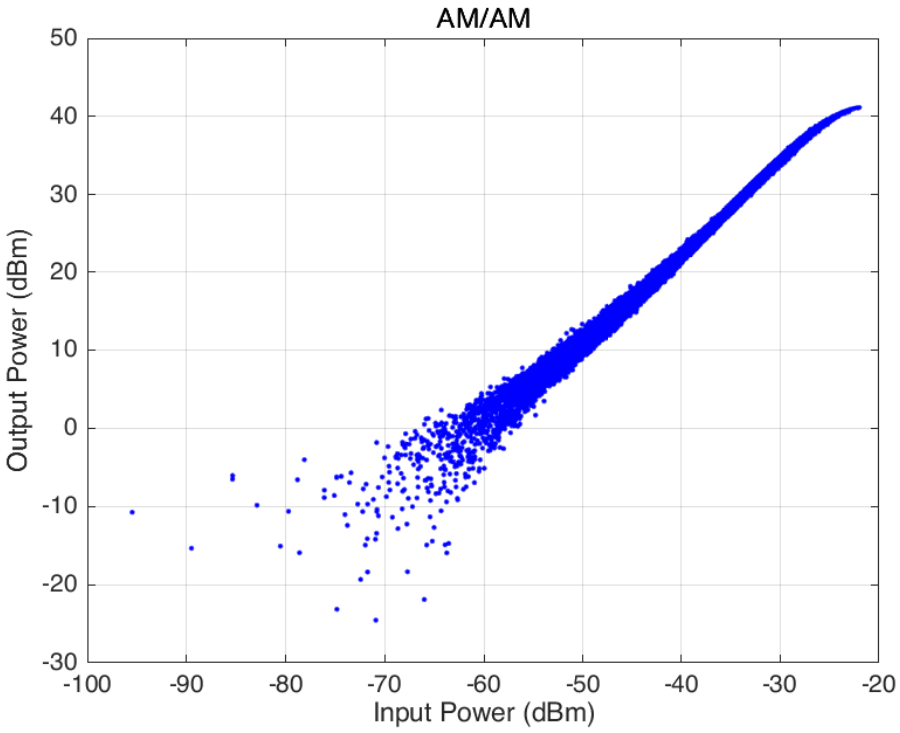

Figure 2.4 AM-to-AM response of the PA samples of the first experiment.

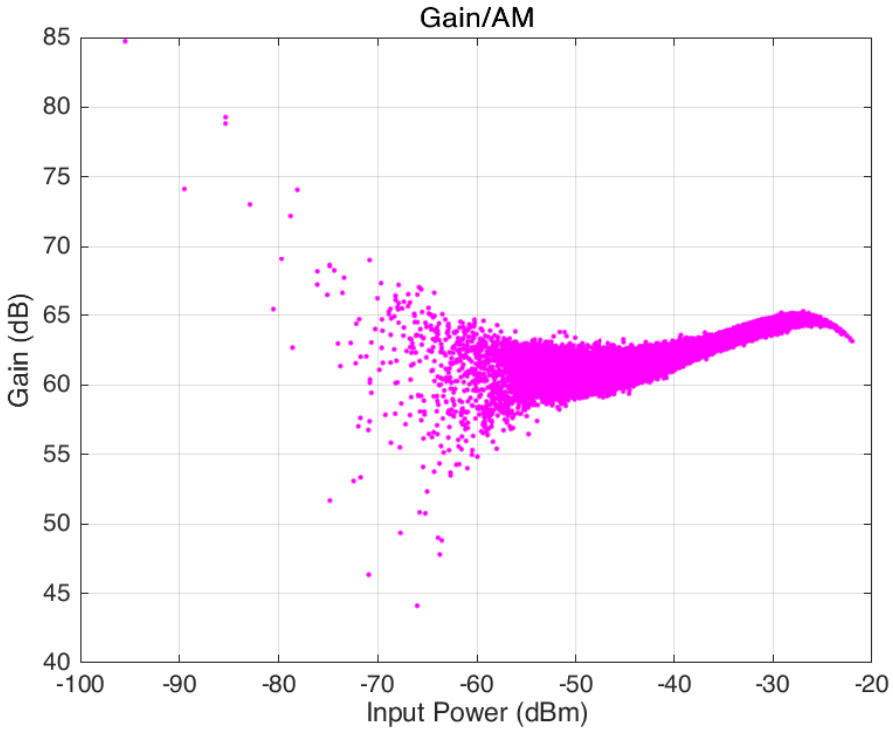

Figure 2.5 Gain-to-AM response of the PA samples of the first experiment.

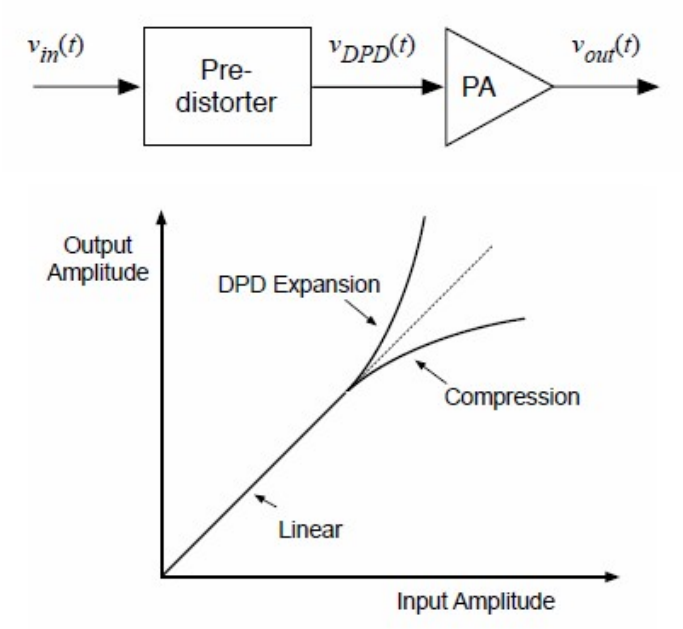

Figure 2.6 Predistorter and PA arranged. AM-to-AM responses along with the resulting output [8].

that must be taken into consideration when designing power amplifiers. This is a contemporary problem which solution engineers have been trying to improve with countless investigations during the current and past years [12, p. 5].

Dynamical systems are those whose response in the present is determined by its past input signals. This are also called systems with memory. This memory effects are a result of energy storage in a circuit formed by resistances and reactances, among others. These can be expressed as linear system's time delays or time derivatives. The amount of past time values needed for nonlinear systems such as PAs is finite, this is known as fading memory. The more we go back in time, the effect in the presents becomes less significant. Short-term memory effects are caused by the PAs capacitors, inductors and transmission lines; and long-term memory effects are caused by thermal-energy storage, charge traps and control circuits.

2.4.3 Digital pre-distortion

Digital pre-distortion is a preeminent technique of linearization in amplifiers. A pre-distorter is placed in front of the power amplifier adding distortion components to its input signal in a way nonlinearities are canceled at the output signal of the PA. There is an example of this in Figure 2.6 where AM-to-AM responses are represented along with the resulting linear output. DPD blocks model the amplifiers' nonlinearities out from samples of input and output data. The nonlinear model identification is key to develop robust DPD systems. Behavioral modeling is the field in which this work is going to focus on.

2.5 Modeling

A PA model describes the relationships between a low-power modulated input signal and a high-power modulated output signal plus distortions. Input and output signals can be currents or voltages running through the RF PA. A behavioral model is a simplified model that does not necessarily contain any reference to the inner physics of the system. Even though, this models are run quicker in simulators than more compact ones. Models have parameters which allow their configuration. In our case, our PA model may have to update his parameters constantly to adapt the DPD block to all sorts of modulated signals that our wireless communication device will transmit.

The process of finding the appropriate parameters for a known solution is called fitting. The input-output relationship can be translated into a mathematical function and parameters can be the coefficients of an approximation for this function. Choosing the right parameters will determine the fitness of the model. How the parameters *fit* will be judge by another function that will evaluate their accuracy or their error.

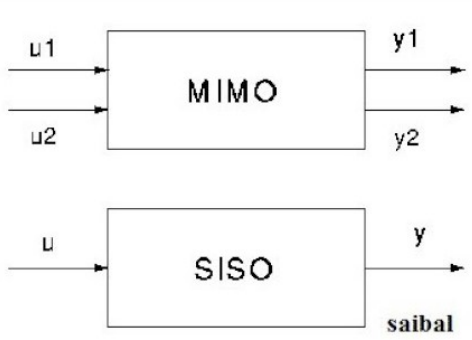

Figure 2.7 MIMO and SISO systems diagram [13].

Polynomial approximations are the best option for global approximation. These work with all the data and give a complete solution, not only for a region of it. Polynomial fit is measured by calculating the distance between the data points and the function estimated point (e.g. minimum squared error) [8, ch. 2].

2.5.1 Linear Systems

After this brief explanation about modeling, it is time to study linear and nonlinear systems in order to understand the behavior of RF PAs. This will help in the selection of a suitable model to solve DPD problems, for example, in wireless communications.

A linear system has an output proportional to its input. This is commonly expressed with the following equation.

$$y = kx, \tag{2.1}$$

where x is the input signal, y is the output signal and k is a weighing coefficient.

This happens for any combination of inputs and outputs and it is known as the principle of superposition:

$$y_1 + y_2 = k_1 x_1 + k_2 x_2 \tag{2.2}$$

These are single-input, single-output (SISO) systems and multiple input, multiple output (MIMO) systems (Figure 2.7). Only SISO systems will be reviewed, as notation will be simpler and everything will hold for the rest.

A linear system output or response is expressed has a sum of two terms: the zero-input response and the zero-state response. The zero-input response, as its name indicates, are the initial conditions when the input is zero. The zero-state response result only from the input when the initial conditions are zero:

$$\text{total response} = \text{zero-input response} + \text{zero-state response} \tag{2.3}$$

We must understand some of the basic types of systems, as other systems properties will be needed to understand the whole problem of modeling a PA. This section will follow the guidance of [14, ch. 1].

Time-invariant systems are those whose parameters do not change with time. The input-output relationship is the same every T seconds.

Instantaneous or memoryless systems are those which output at any instant only or mostly depends on the input at the same instant. Otherwise, the system is dynamic or with memory and its response is determined by the input over the past T seconds.

Causal or physical systems are those where the present instant output only depends on present and past instants inputs. Noncausal systems depend on future values too. It is necessary to operate with causal systems when working with real-time applications, but the study of noncausal systems can be important when the independent variable is not time but others such as space, or when approximate them as causal systems with delay (Figure 2.8).

Discrete-time systems are samples of continuous-time systems. The main objective of the first ones are giving the possibility of processing continuous-time systems with computers, as demonstrated in Figure 2.9. From now on, this work will focus on time-sampled measured data, so equations can be cast in a discrete form. We shall take continuous time descriptions of linear memory systems as linear difference equations:

$$y(t) + a_1 y(t-1) + a_2 y(t-2) + \ldots + a_n y(t-n) = b_0 u(t) + b_1 u(t-1) + \ldots + b_m u(t-m) \tag{2.4}$$

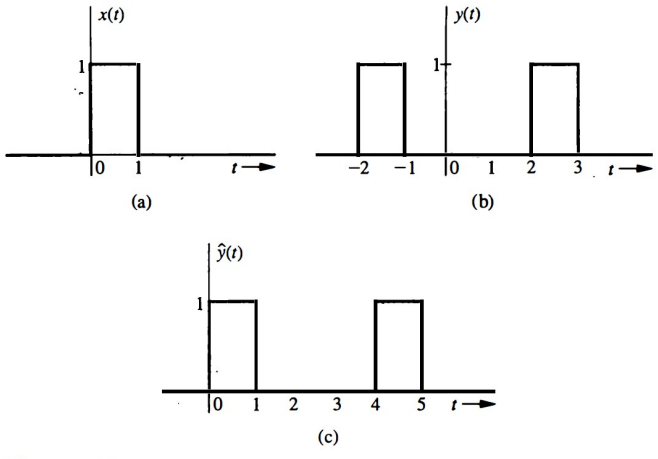

Figure 2.8 A noncausal system and its approximation as a delayed causal [14].

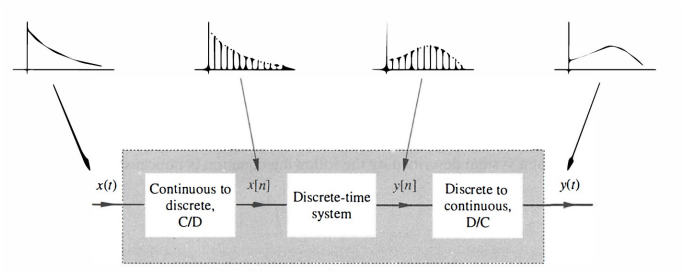

Figure 2.9 Continuous-time processing by a discrete-time system [14].

,

where $a_0 = 1$ for convenience. The delays m and n are memory depths in the input and the output signals, a_i and b_j are the parameters of the output y(t) and the input u(t) respectively.

2.5.2 Model Identification

The equation previous can be recast in terms of the current instant output y(t) and shortened by vectorizing all the components:

$$y(t) = \phi^T \theta \tag{2.5}$$

,

being the ϕ(t) regression vector and θ the vector of model parameters

$$\phi(t) = \left[\begin{array}{cccccccc} -y(t-1) & -y(t-2) & \ldots & -y(t-n) & u(t) & u(t-1) & \ldots & u(t-m) \end{array} \right]^T \tag{2.6}$$

$$\theta = \left[\begin{array}{cccccccc} a_1 & a_2 & \ldots & a_n & b_0 & b_1 & \ldots & b_m \end{array} \right] \tag{2.7}$$

Although the solution for y(t) could be recast as a sum of basis functions, where the basis functions are fundamental solutions to the system:

$$y(t) = \sum_{n=0}^{N} a_n f_n(u(t)) \tag{2.8}$$

These set of functions can be extended to nonlinear systems, being linear in the parameters a_n. This feature allows a wide variety of linear solutions techniques to be used, such as the least square technique. A least square criterion will be used in this work.

Given a model structure, with a N samples dataset, an estimation of y(t) can be obtain from a selection of parameters θ and of regressors of ϕ(t) and be denoted as $\hat{y}$(t). The fitness of the selection of model parameters

can be calculated from the average of the squares of the distances between the calculated and the measured point. This measures how far the estimated points fell of the measured data. This is the ℓ_2 norm

$$Error = \frac{1}{N}\sum_{t=1}^{N}(y(t)-\hat{y}(t|\theta))^2 = \frac{1}{N}\sum_{t=1}^{N}(y(t)-\phi^T(t)\theta)^2 \tag{2.9}$$

The derivative of this expression equate to zero clears the path in the search of a variable which helps minimizing the error:

$$\frac{2}{N}\sum_{t=1}^{N}(\phi(t)(y(t)-\phi^T(t)\theta)) = 0 \tag{2.10}$$

The model parameters θ becomes the solution that minimizes the distance between the estimated and the measured points:

$$\hat{\theta} = \frac{\sum_{t=1}^{N}\phi(t)y(t)}{\sum_{t=1}^{N}\phi(t)\phi^T(t)} \tag{2.11}$$

being $\hat{\theta}$ the parameters extracted from the measured data.
In this dissertation, the error will be measured my the normalized mean square error (NMSE):

$$NMSE = \frac{\sum_{t=1}^{N}(y(t)-\hat{y}(t))^2}{\sum_{t=1}^{N}y(t)^2} \tag{2.12}$$

Although, expressed in a logarithmic scale:

$$NMSE\big|_{dB} = 10\log_{10}NMSE \tag{2.13}$$

John Wood clearly defines the system identification as "*the mathematical framework for determining the dynamical properties of a given system*". This is basically the procedure that will be followed to build a model, out from a dataset, a model structure and an optimization criterion.

2.5.3 Modeling PA Nonlinearities

As described before, the objective of modeling is to find a smooth function that approximates the input-output relationship with the minimum error. It must be assumed that noise will not allow estimated points to hit exactly where data points are, but close and centered around them. It was also explained before how we could express the solution of our model as a weighted sum of basis functions. This could now be recast in terms of voltage amplitude, to better represent the characteristic curve of a PA.

$$v_{out}(t) = \sum_{n=1}^{N} a_n v_{in}^n(t) \tag{2.14}$$

Being a_n the linear parameters and N the degree of the polynomial. This can be written as a matrix:

$$\begin{pmatrix} v_{in}(t_1) & v_{in}^2(t_1) & \dots & v_{in}^N(t_1) \\ v_{in}(t_2) & v_{in}^2(t_2) & \dots & v_{in}^N(t_2) \\ v_{in}(t_3) & v_{in}^2(t_3) & \dots & v_{in}^N(t_3) \\ \dots & \dots & \ddots & \dots \\ v_{in}(t_T) & v_{in}^2(t_T) & \dots & v_{in}^N(t_T) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{pmatrix} = \begin{pmatrix} v_{out}(t_1) \\ v_{out}(t_2) \\ v_{out}(t_3) \\ \vdots \\ v_{out}(t_N) \end{pmatrix} \tag{2.15}$$

If the number of data points is the same as the polynomial degree, T equals N, this is a square matrix called Vandermonde matrix. It can be easily solved, although having such polynomial degree would not be a great idea. More degree is not always better, increasing it to a very high number would not make the model more

Figure 2.10 Vito Volterra [15].

precise. Fitness and accuracy end up saturating at certain degree. Generally, T will be much higher than N, having an overdetermined system, with more data than parameters. This can be cast as follows:

$$H\vec{x} = \vec{y} \tag{2.16}$$

Where H is an M x N matrix, with a number of data points bigger than coefficients: M > N. A solution for this matrix equation is found as follows:

$$H\vec{x} = \vec{y}, \tag{2.17}$$

$$H'H\vec{x} = H'H\vec{y}, \tag{2.18}$$

$$\vec{x} = (H'H)^{-1}H'\vec{y}, \tag{2.19}$$

this is the least square (LS) method. It will be used in one of the algorithm that will be discussed later. Although while modeling with voltage amplitude, only AM-to-AM conversion effects were being included in the model. To include AM-to-PM effects to the model, the coefficients of the polynomial shall be considered complex, so phase is also taken into the equation:

$$a_i \rightarrow |a_i| \angle \varphi = Re[a_i e^{j\varphi_i}] \tag{2.20}$$

2.6 Nonlinear Dynamical Models

After summarizing the mathematical background of this state-of-the-art review, a mathematical framework is now needed for building nonlinear dynamical models. The framework chosen for this task were Volterra series. This will be done in a sampled time domain. Volterra series are a very straightforward way to build linear-in-parameters polynomial models, this features were explained in past chapters.

Vito Volterra (Figure 2.10) was an Italian mathematician whose work led to the development of a new branch of mathematics: functional analysis. He studied at the University of Pisa and graduated with a Doctor of Physics degree in 1882. He was named Professor of Rational Mechanics at Pisa a year later. His theory of functionals led to new fields of analysis, such as the solution of differential and integral equations. Norbert Wiener became very interested of Volterra's theories through a student of the Italian: Paul Levy. Wiener was the first one known to use Volterra series to model PAs. The limitations in computational capacity made

Volterra series of higher degree wait until more powerful techniques were invented and computers were capable of handling their computational load.

2.6.1 Volterra Series

Volterra Series can be described out from Taylor series. An expression very similar to those used in previous chapters is a nice starting point:

$$y(t) = a_0 + \sum_{n=1}^{N} a_n u^n(t) \tag{2.21}$$

The a_n polynomial coefficients are extracted from the Taylor series of the input-output relationship, around some operating point u_0:

$$y(u(t)) = y(u)\big|_{u=u_0} + \frac{1}{2!}\frac{dy}{du}\bigg|_{u=u_0}(u-u_0) + \frac{1}{3!}\frac{d^2y}{du^2}\bigg|_{u=u_0}(u-u_0)^2 + \cdots = a_0 + a_1 u + a_2 u^2 + \ldots \tag{2.22}$$

In order to consider memory effects, y(t) must be a function of u(t) and the previous values of u:

$$y(t) = f(u,u_1,u_2,\ldots,u_n) \tag{2.23}$$

being

$$u = u(t), u_1 = u(t-\tau_1), u_2 = u(t-\tau_2), \ldots, u_n = u(t-\tau_n) \tag{2.24}$$

The Taylor series expression should be then:

$$\begin{aligned} y(u(t)) =& y(u)\big|_{u-u_0} + \frac{1}{2!}\frac{dy}{du}\bigg|_{u=u_0}(u-u_0) + \frac{1}{2!}\frac{dy}{du_1}\bigg|_{u=u_0}(u_1-u_0) + \ldots \\ &+ \frac{1}{3!}\frac{d^2y}{du^2}\bigg|_{u=u_0}(u-u_0)^2 + \frac{1}{3!}\frac{d^2y}{dudu_1}\bigg|_{u=u_0}(u-u_0)(u_1-u_0) + \ldots \end{aligned} \tag{2.25}$$

This is a multinomial series where memory terms are easily observed in $(u_1 - u_0)$ and in the cross-terms: $(u-u_0)(u_1-u_0)$. That is why Volterra series are usually referred to as Taylor series with memory. As mentioned numerous times earlier, for the model generation, discrete time-sampled data will be used. So, the expression which considers memory effects will be recast as follows:

$$y(t) = f_{poly}(u(t),u(t-1),\ldots,u(t-M)) \tag{2.26}$$

Where $f_{poly}()$ represents the Volterra polynomial function, t is the sampling instant and M is the memory depth of the model, measured as time steps or delays. This expression can be expanded in the others of the multinomial series:

$$\begin{aligned} y(t) =& \sum_{m_1}^{M} h_1(m_1)u(n-m_1) \\ &+ \sum_{m_1}^{M}\sum_{m_2}^{M} h_2(m_1,m_2)u(n-m_1)u(n-m_2) + \ldots \\ &+ \sum_{m_1}^{M}\cdots\sum_{m_k}^{M} h_2(m_1,\ldots,m_k)u(n-m_1)\ldots u(n-m_k) \\ &+ \sum_{n=1}^{K}\sum_{m_1}^{M}\cdots\sum_{m_k}^{M} h_n(m_1,\ldots,m_k)\prod_{j=1}^{k} u(n-m_j) \end{aligned} \tag{2.27}$$

,

where $h_n(m_1,m_2,\ldots,m_n)$ are the Volterra kernels, analogous to the coefficients of the instantaneous polynomial; N, the polynomial degree; and M, the memory depth. Usually, Volterra series keep the same memory depth for all the polynomial orders.

Kernels are nothing but system transfer functions which we use in a discrete form for our discrete nonlinear dynamical model. Morgan and colleagues [16] refer to them as filters, and with this interpretation we will be able to work with our data in their low-, band or high- pass components.

Accuracy of Taylor series models is best when close to the point around which is being made. A Volterra series model of PA should be centered at or close to the average signal power. From this point, the PA will typically run into compression, so memory and thermal effects will be included in the model. The solution given by the Volterra series can be improved also by adding more terms into the polynomial series: memory and cross-terms. This will lead to a smaller tolerance of the model. However, increasing the number of terms also increases the difficulty to find a solution for a model. Higher order Volterra series were a mere hypothesis until modern and multicore processors made their appearance.

There has been several attempts to reduce the number of terms when modeling power amplifiers. This is referred to in many articles as "pruning", as a metaphor where the terms are the leaves and branches of a Volterra series tree. Pruning techniques by themselves would be enough to write a few long final degree works. A Volterra series modeled can be pruned just by limiting the total number of coefficients in the polynomial or by following complex algorithms that evaluate the significance of every term. It is common to see even-order terms ignored when modeling PAs, although no pruning technique will be used in the first algorithm that will be explained later.

Simplified Volterra series models have been proposed over the years. The model on which this work is going to focus comes of the refinement of models, all derivatives from the Volterra series model.

The first of them is the Wiener model:

$$y_W(n) = \sum_{k=0}^{K} a_k \left[\sum_{m=0}^{M-1} h(m)u(n-m) \right]^k \tag{2.28}$$

In this model, kernels are integrated into the power series, so they are no longer linear in parameters. This makes it more difficult to identify the coefficients.

The second is the Hammerstein model:

$$y_H(n) = \sum_{m=0}^{M-1} h_m \left[\sum_{k=0}^{K} a_k u^k(n-m) \right] \tag{2.29}$$

This is a much simpler expression, where kernels can be easily identified as it is linear in parameters.

The Wiener-Hammerstein model is the combination of the two previous models:

$$y_{WH}(n) = \sum_{m_2}^{M-1} \sum_{k=0}^{K} a_k \left[\sum_{m_1=0}^{M-1} h(m_1)u(n-m_1-m_2) \right]^k \tag{2.30}$$

This model mimics the structure of a PA closer than the previous ones, although it still presents the nonlinear parameters like the Wiener model.

Choosing different filters for each different order and thus combining the different kernels and power series coefficients, the generalized Hammerstein can be cast:

$$y_{GH}(n) = \sum_{k=1}^{K} \sum_{m=0}^{M-1} a_{km} x^k(n-m) \tag{2.31}$$

Considering only the narrowband case, we arrive to the memory polynomial, formulated in [17]:

$$y_{MP}(n) = \sum_{k=0}^{K-1} \sum_{m=0}^{M-1} a_{km} x(n-m) \|x(n-m)\|^k \tag{2.32}$$

This model only includes what are called "diagonal" terms, where there are no cross-terms. The Generalized Memory Polynomial (GMP) proposed by Morgan et al. [16] introduces the cross-terms to the previous model. In this case, both positive and negative cross-terms are considered:

$$\begin{aligned} y_{GMP}(n) = & \sum_{k=0}^{K_a-1} \sum_{l=0}^{L_a-1} a_{kl} x(n-l)|x(n-l)|^k \\ & + \sum_{k=0}^{K_b} \sum_{l=0}^{L_b-1} \sum_{m=0}^{M_b} b_{klm} x(n-l)|x(n-l-m)|^k \\ & + \sum_{k=0}^{K_c} \sum_{l=0}^{L_c-1} \sum_{m=0}^{M_c} c_{klm} x(n-l)|x(n-l+m)|^k \end{aligned} \tag{2.33}$$

In this model we can identify: the a-part, where the diagonal terms are included; the b-part, where the lagging cross-terms are; and the c-part, where the leading cross-terms are. The GMP model is linear in parameters, so they can be simply estimated with any least square algorithm.

This model shall be used for the first experiment. The parameters that the first algorithm will try to optimize will be K_a,L_a,K_b,L_b,M_b,K_c,L_c,M_c.

3 The Hill Climbing Algorithm

An ounce of action is worth a ton of theory.

FRIEDRICH ENGELS

Although the model structure which is going to be used for modeling PA has already been fixed, an algorithm of optimization is still needed. This must contain a list of rules or operations which lead to a solution. In this work, the solution will be the configuration parameters that build the optimal model. For this task, the Hill-Climbing Heuristic (HC) was selected. The reason why this algorithm picked as the first one in this work, was to put into practice and review what Wang et al. explained in their work [2]. As it is mentioned in that paper, to their knowledge, it was the first time that an algorithm based on HC was implemented and studied for sizing GMP model structures of DPD. There was a stopping criterion selected in order to find the right combination of the model parameters for each experiment, and not become and endless iterative algorithm, as well as another algorithm to compare with the HC and a laboratory procedure in order to obtain empirical results out of the experiment. All this will be further explained inside this chapter.

3.1 The HC Algorithm

HC is a greedy search algorithm that makes an exhaustive search, evaluating one by one all the models until the best one to its criterion is found. When it finds the local optimum model inside a local solution space, this local optimum model will be used to search again around its local solution space for a *more optimum* model. Although HC algorithm has proven its efficacy, it does not necessarily find global optimum solutions. HC algorithms are very dependent of their starting point, and this sometimes leads to give a solution that is a local and not global one.

HC is an iterative algorithm that begins from an initial solution and then tries to find a better solution by comparing it to its *neighbors*. Every time a better solution is found among the neighbors of the current solution, it is taken as current solution. This process is repeated until no better solution is found. There must be a criterion that determines the fitness of a model, so the algorithm can sort the solutions in order to find the best. Two are proposed in [2], although only the additive criterion is studied in this final degree work.

This criterion aims to evaluate how good a model is. This is made through looking into the models' accuracy (NMSE_{dB}), has mentioned earlier. However, as explained in [8, ch. 6,pp. 165–168], accuracy of Volterra series models can be improved by increasing the number of terms of the model, so, the additive criterion also evaluates the model complexity, reflected in the number of coefficients of the model. To achieve this, the additive criterion is defined as a weighted sum of accuracy and complexity:

$$\mathrm{J}(x_i) = \mathrm{NMSE}_{dB}(x_i) + \mu \mathrm{NUMCOEF}(x_i) \tag{3.1}$$

where $J(x_i)$ is the merit value that represents the fitness of the model, $\mathrm{NMSE}_{dB}(x_i)$ is the normalized mean squared error in decibels and $\mathrm{NUMCOEF}(x_i)$ is the number of coefficients, all of the model structure i. Merit values become better as they become more negative, as the NMSE_{dB} is a negative value itself. The higher the accuracy and the lesser number of coefficients, the better.

The number of regressors of a GMP model is obtained with the following equation:

$$\text{NUMCOEF} = (K_a+1)(L_a+1)+K_b(L_b+1)M_b+K_c(L_c+1)M_c \tag{3.2}$$

Instead of performing an exhaustive search, the algorithm works with the merit value and the definition of a neighborhood used.

As explained in [2], all the possible GMP model structures are held inside the discrete space U. Each structure x_i consists of 8 coordinates: $K_{a,i}$,$L_{a,i}$,$K_{b,i}$,$L_{b,i}$,$M_{b,i}$,$K_{c,i}$,$L_{c,i}$,$M_{c,i}$, which are the parameters of the model. A merit value $J(x_i)$ is associated to each element x_i.

HC tests only elements which are successors or neighbors, which is how they will be referred to from now on, at each iteration. Neighbors are defined as an 8-tuple:

$$(K_{a,i}+\delta_1,L_{a,i}+\delta_2,K_{b,i}+\delta_3,L_{b,i}+\delta_4,M_{b,i}+\delta_5,K_{c,i}+\delta_6,L_{c,i}+\delta_7,M_{c,i}+\delta_8) \tag{3.3}$$

where

$$\delta_{1,\ldots,8} \in [-1,+1], \tag{3.4}$$

where the subspace of neighbors or neighborhood i is denoted S_i and is inside U: $S_i \in U$.

The HC algorithm searches for the neighbor with the minimum merit value at each iteration, takes it as the new current solution and starts again searching around its neighbors. There is an overlap between neighborhoods of subsequent iterations, however the algorithm does not evaluate elements which have been already evaluated. The search ends when the solution for two subsequent iteration is the same, so no better model structure has been found. An example of this is shown in Figure 3.1, where the red line is the path of optimum models selected at each iteration until the best one is found.

The maximum size of a neighborhood (SON) is $3^8 = 6561$, although this maximum will not be often reached. This happens because the overlapping elements between neighborhoods are not evaluated twice, but only in the neighborhood of the model evaluated first. Each iteration neighborhood is shown in a different color. The SON can be also seen in Figure 3.2 for the same simulation as the one mentioned before. The algorithm was set to find the best model with a maximum of 80 parameters. Representing the SON against the number of iterations allows to see how complex a search for a model was with the naked eye.

The obtention of the weighting coefficient μ still needs to be explained. Although there is already a proposal of how to obtain it in [2], the approach presented by Becerra et al [18] was used for the first experiment.

3.2 The Bayesian Information Criterion

The weighting coefficient is the angular stone of the algorithm, as it will dictate how the merit value is calculated. This is why it is also called stopping criterion. The one proposed at [18] is based on the Bayesian information criterion (BIC) by Schwarz in [19]. Schwarz presented an application of the Bayes' theorem to obtain a modification of the maximum likelihood principle.

In [18], the BIC expression depends on the error, the number of model coefficients and the number of samples used for the model identification. This makes the error dependent of the number of samples. The BIC is initially decreasing, and the best merit value will be found when the BIC function stops decreasing. The optimum model selected by the algorithm will be the one whose BIC value is an absolute minimum.

Following the same mathematical procedure applied in [18], μ will become μ_{BIC}:

$$\mu_{BIC} = \frac{10}{M}\log_{10} 2M, \tag{3.5}$$

where M is the total number of samples used for the search.

Thus, the additive criterion will be:

$$J_{BIC}(x_i) = NMSE_{dB}(x_i) + \mu_{BIC} NUMCOEF(x_i) \tag{3.6}$$

3.3 The DOMP algorithm

Another algorithm also by Becerra et al [18] was used to match results with the HC in the coming experiment. The Doubly Orthogonal Matching Pursuit (DOMP) [20] belongs to the vast literature on techniques applied on the DPD context. This method performs two orthogonalizations over the basis matrix. To better understand this algorithm, the OMP should be explained first:

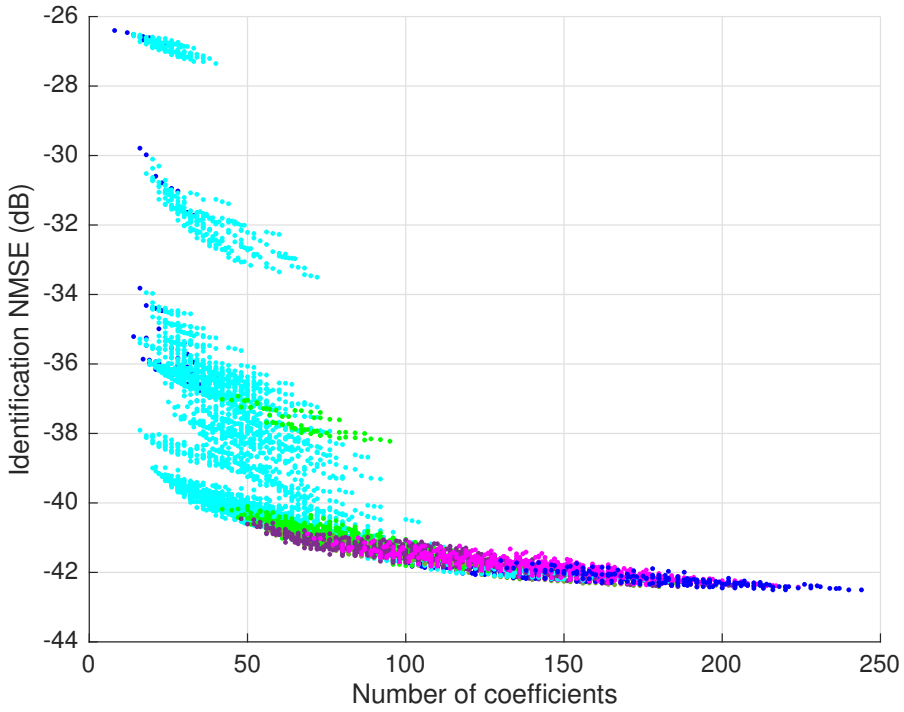

Figure 3.1 Example of a plot of subsequent neighborhoods in different colors.

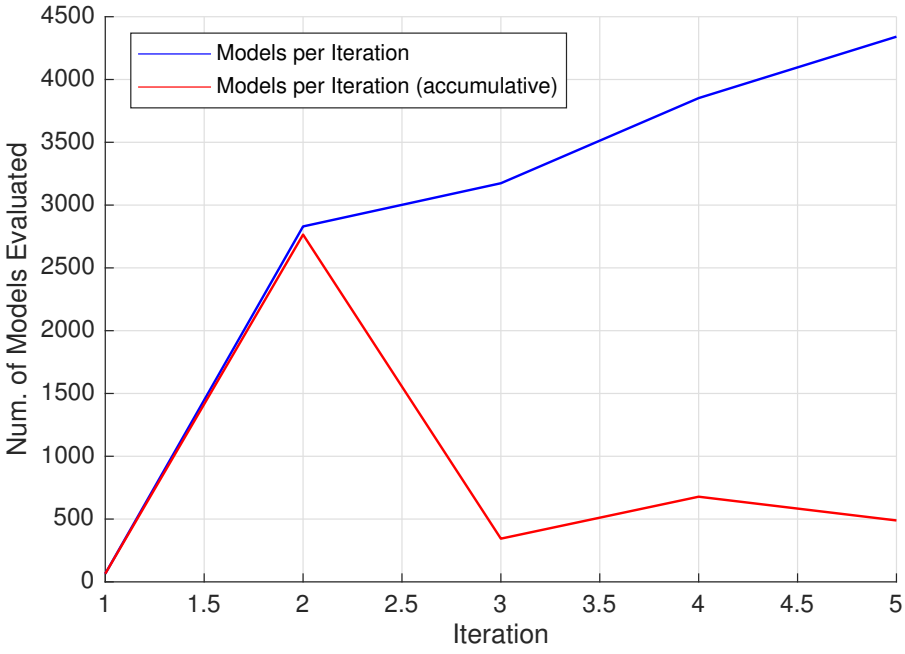

Figure 3.2 Number of models calculated per iteration or the size of all the neighborhoods in red and the accumulative count of models calculated per iteration in blue of a search limited to a maximum of 80 coefficients per model.

3.3.1 The OMP algorithm

The Orthogonal Matching Pursuit (OMP) is an sparse approximation algorithm which minimizes the sparsity of an input vector by finding the best matching projections of a dictionary or family of functions or vectors with the output. The error is measured through the residual, which is the difference between the output and the estimated output.

The equation to be solved is the following:

$$y = Xh \tag{3.7}$$

where X is the dictionary ($X \in \mathbb{C}^{MxN}$), y is the output signal ($y \in \mathbb{C}^{M}$) and h is the Volterra kernel vector.

During the first iteration, the column of X whose projection over y is larger is calculated and added to the basis matrix:

$$i^{(1)} = \underset{1 \leq n \leq N}{\arg\max} |X_n^H y| \tag{3.8}$$

Then, the column $i^{(1)}$ of X is added to the basis matrix:

$$S_{(1)} \leftarrow X_{i^{(1)}} \tag{3.9}$$

The least square technique is applied to obtain an estimation of the Volterra kernel vectors:

$$\hat{h}^{(1)} = S_{(1)}^{+} y \tag{3.10}$$

where $S_{(1)}^{+}$ is the pseudo-inverse matrix:

$$S_{(1)}^{+} = (S_{(1)}^H S_{(1)})^{-1} S_{(1)}^H \tag{3.11}$$

After this, the error is measured via the residual:

$$r^{(1)} = y - S_{(1)} \hat{h}^{(1)} = y - \hat{y}^{(1)} \tag{3.12}$$

The subsequent iterations will take the residual as a starting point. With this said, the *m-th* iteration would be:

$$i^{(m)} = \underset{1 \leq n \leq N}{\arg\max} |X_n^H r(m-1)| \tag{3.13}$$

$$S_{(m)} \leftarrow [S_{(m-1)} X_{i^{(m)}}] \tag{3.14}$$

$$\hat{h}^{(m)} = S_{(m)}^{+} y \tag{3.15}$$

$$r^{(m)} = y - S_{(m)} \hat{h}^{(m)} = y - \hat{y}^{(m)} \tag{3.16}$$

where $S_{(m)}^{+} = (S_{(m)}^H S_{(m)})^{-1} S_{(m)}^H$ and $m \geq 1$

3.3.2 The DOMP

The basis set is a matrix defined in [20] that contains the information about the orthogonalized regressors. It begins with $Z^{(0)} = X$. And during every iteration it is normalized by its $\ell_2 - norm$.

$$Z_{\{i\}} \leftarrow \frac{Z_{\{i\}}^{(m-1)}}{\|Z_{\{i\}}^{(m-1)}\|_2} \tag{3.17}$$

The DOMP performs the Gram-Schmidt orthogonalization by obtaining the vector projections of the selected regressor onto each one of the components of the basis matrix. Then, this projections are substracted from each regressor, ensuring they are orthogonal to the remaining of the basis set.

$$p^{(m)} = Z_{\{i^{(m)}\}}^{(m-1)^H} Z^{(m-1)} \tag{3.18}$$

$$Z^m = Z^{(m-1)} - p^{(m)} \otimes Z^{(m-1)}_{\{i^{(m)}\}} \tag{3.19}$$

Where $\otimes$ is the denotes the Kronecker product of the two matrices.

4 Experimental Designs and Results

In order to seek truth, it is necessary once in the course of our life to doubt, as far as possible, of all things.

RENÉ DESCARTES

4.1 BIC: Number of Samples

As it was explained before, the BIC rule theoretically decreases as the number of samples of the input and output signals M used to create the model increases. This could make the algorithm to identify as optimum models some with components with less effect on the error iteration after iteration [18]. However, this was put into practice in this work, comparing tests varying M from 5500 to roughly 70000 data samples.

The first thing which clearly increases with M is the time of calculation (ToC). In Figure 4.13, there is an example of this. The NMSE suffers a small increase of 1.4 dB (Figure 4.14), although the ToC goes up from 3 to 97 minutes. There is clearly no advantage on deliberately increasing the number of samples used for the exhaustive search of the optimum model. The optimum structure for each simulation varying only the number of samples is listed below, as well as a plot of every simulation from Figure 4.1 to 4.11.

For 5529 samples (1.5% of the signal's total length), after 4 iterations:

$$K_a = 5, L_a = 1$$
$$K_b = 1, L_b = 2, M_b = 4$$
$$K_c = 3, L_c = 3, M_c = 4$$
$$NMSE_{dB} = -41.3921$$
$$NUMCOEF = 72$$

For 7372 samples (2% of the signal's total length), after 15 iterations:

$$K_a = 6, L_a = 3$$
$$K_b = 1, L_b = 2, M_b = 4$$
$$K_c = 5, L_c = 1, M_c = 4$$
$$NMSE_{dB} = -40.8157$$
$$NUMCOEF = 80$$

For 8479 samples (2.3% of the signal's total length), after 14 iterations:

$$K_a = 5, L_a = 5$$
$$K_b = 1, L_b = 1, M_b = 4$$
$$K_c = 4, L_c = 1, M_c = 4$$
$$NMSE_{dB} = -41.2527$$

$$NUMCOEF = 76$$

For 11427 samples (3.1% of the signal's total length), after 33 iterations:

$$K_a = 5, L_a = 3$$
$$K_b = 2, L_b = 1, M_b = 4$$
$$K_c = 5, L_c = 1, M_c = 4$$
$$NMSE_{dB} = -40.6274$$
$$NUMCOEF = 80$$

For 13529 samples (3.67% of the signal's total length), after 38 iterations:

$$K_a = 5, L_a = 3$$
$$K_b = 1, L_b = 1, M_b = 4$$
$$K_c = 7, L_c = 1, M_c = 3$$
$$NMSE_{dB} = -40.4647$$
$$NUMCOEF = 74$$

For 15851 samples (4.3% of the signal's total length), after 15 iterations:

$$K_a = 5, L_a = 5$$
$$K_b = 1, L_b = 1, M_b = 4$$
$$K_c = 5, L_c = 1, M_c = 3$$
$$NMSE_{dB} = -40.9356$$
$$NUMCOEF = 74$$

For 18431 samples (5% of the signal's total length), after 14 iterations:

$$K_a = 3, L_a = 5$$
$$K_b = 10, L_b = 1, M_b = 2$$
$$K_c = 4, L_c = 1, M_c = 2$$
$$NMSE_{dB} = -40.6915$$
$$NUMCOEF = 80$$

For 23225 samples (6.3% of the signal's total length), after 33 iterations:

$$K_a = 3, L_a = 5$$
$$K_b = 1, L_b = 4, M_b = 4$$
$$K_c = 4, L_c = 2, M_c = 3$$
$$NMSE_{dB} = -41.1251$$
$$NUMCOEF = 80$$

For 33177 samples (9% of the signal's total length), after 34 iterations:

$$K_a = 3, L_a = 5$$
$$K_b = 2, L_b = 4, M_b = 4$$
$$K_c = 4, L_c = 1, M_c = 2$$
$$NMSE_{dB} = -40.7117$$
$$NUMCOEF = 80$$

For 55295 samples (15% of the signal's total length), after 10 iterations:

$$K_a = 6, L_a = 5$$
$$K_b = 1, L_b = 1, M_b = 2$$
$$K_c = 4, L_c = 1, M_c = 4$$
$$NMSE_{dB} = -40.5884$$
$$NUMCOEF = 78$$

For 70042 samples (19% of the signal's total length), after 18 iterations:

$$K_a = 3, L_a = 5$$
$$K_b = 10, L_b = 1, M_b = 2$$
$$K_c = 4, L_c = 1, M_c = 2$$
$$NMSE_{dB} = -39.9820$$
$$NUMCOEF = 80$$

Even if the number of samples is multiplied almost 13 times from the initial value of 5529 samples, the value of the NMSE does not differ much from the default one. NUMCOEF fluctuates arbitrarily from 72 to 80 regressors, although this difference is not significant. The small difference between the optimum paths of all the simulations is represented in Figure 4.12.

4.2 Size Matters in Exhaustive Searches

In order to better understand how this algorithm works, various simulations were done with further limits. The main reason for this was to empirically prove how always increasing the number of coefficients is going to bring better NMSE and how important is to choose the correct stopping rule.

The HC method used with the BIC stopping rule was limited to a maximum of 250 regressors per model. The model parameters were also delimited to a certain range in order to reduce the ToC, as they do in [2]. This has been empirically proven in this work, bringing down the ToC from days to few hours to few minutes. The ranges of the parameters for the upcoming experiments will be:

$$1 \leqslant K_a \leqslant 11, 1 \leqslant K_b \leqslant 10, 1 \leqslant K_c \leqslant 10$$
$$1 \leqslant L_a \leqslant 5, 1 \leqslant L_b \leqslant 5, 1 \leqslant L_c \leqslant 5$$
$$1 \leqslant M_b \leqslant 4, 1 \leqslant M_c \leqslant 4$$

It is interesting also how the size of the neighborhoods evolves in simulations with higher limits of coefficients. That is the case of Figure 4.15, where the limit was set to 400 coefficients. This type of simulation is impractical. The reason for considering this impractical is simple: a 1-dB-improvement in the NMSE should not come at the cost of 200 coefficients. However, seing how the algorithm works in the long run could help to understand its behavior. When the same search is delimited inside the ranges explained above, number of iterations drastically drops from 233 to 13. The time of execution is reduced from 1217.60 minutes or 20.29 hours to 19.14 minutes. This justifies the necessity of applying delimitation rules to the parameters.

Figure 4.17 belongs to the same simulation as Figure 4.15. The monotonically decreasing convex hull reveals at first glance that the optimum model is going to be found in close to the limit of coefficients. There is another plot using the merit value that allows the observer to see how the minimization process using the BIC rule affects to the exhaustive search for the optimum model. This is the case of Figure 4.18. The *belly* that appears between the 100 and the 350 coefficients-per-model range helps foreseeing where the optimum is. This idea is confirmed when the optimums path is overlapped.

The solution for this heavy and impractical simulation, after 233 iterations was:

$$K_a = 5, L_a = 9$$
$$K_b = 3, L_b = 1, M_b = 2$$
$$K_c = 4, L_c = 4, M_c = 7$$
$$NMSE_{dB} = -42.8269$$
$$NUMCOEF = 212$$

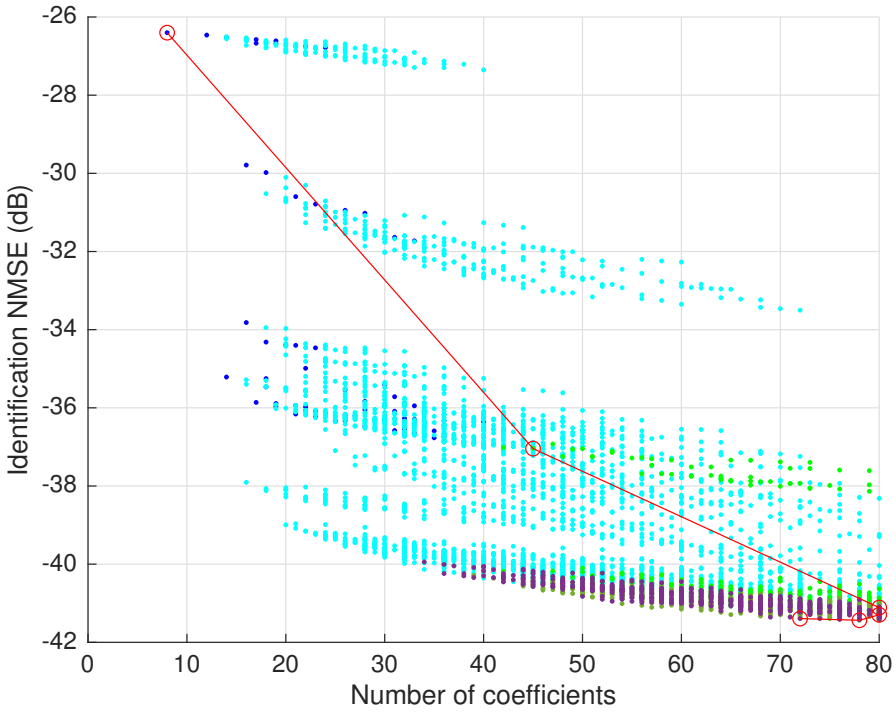

Figure 4.1 Plot of the HC when the number of samples is set to 5529. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

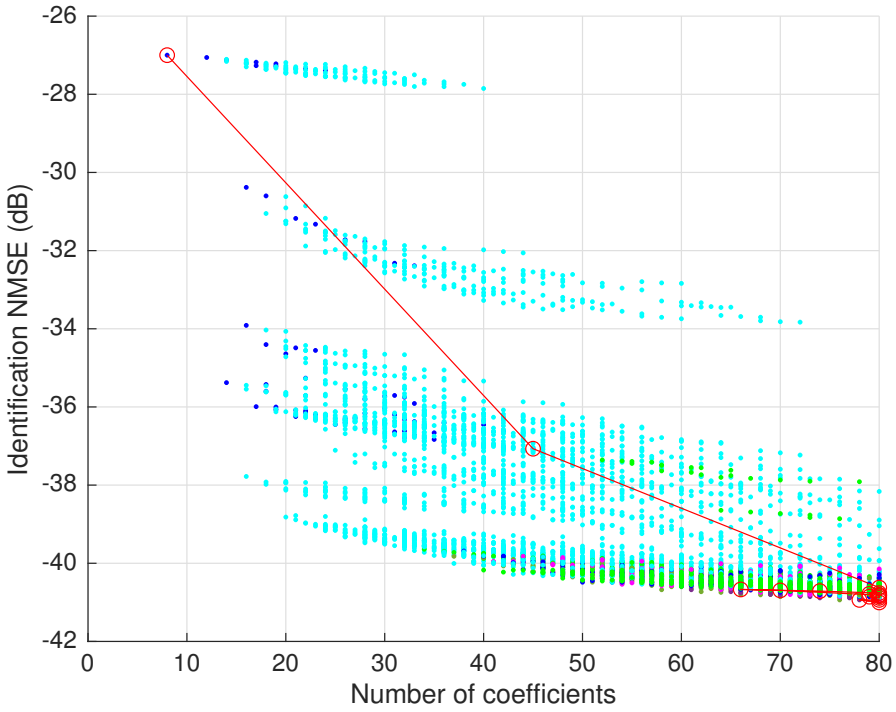

Figure 4.2 Plot of the HC when the number of samples is set to 7372. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

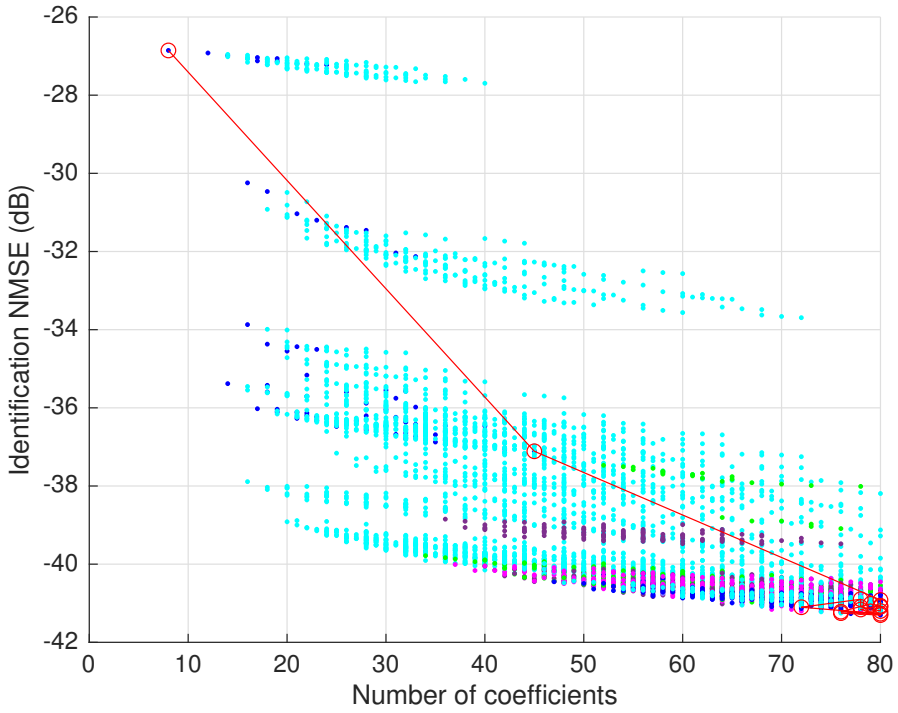

Figure 4.3 Plot of the HC when the number of samples is set to 8479. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

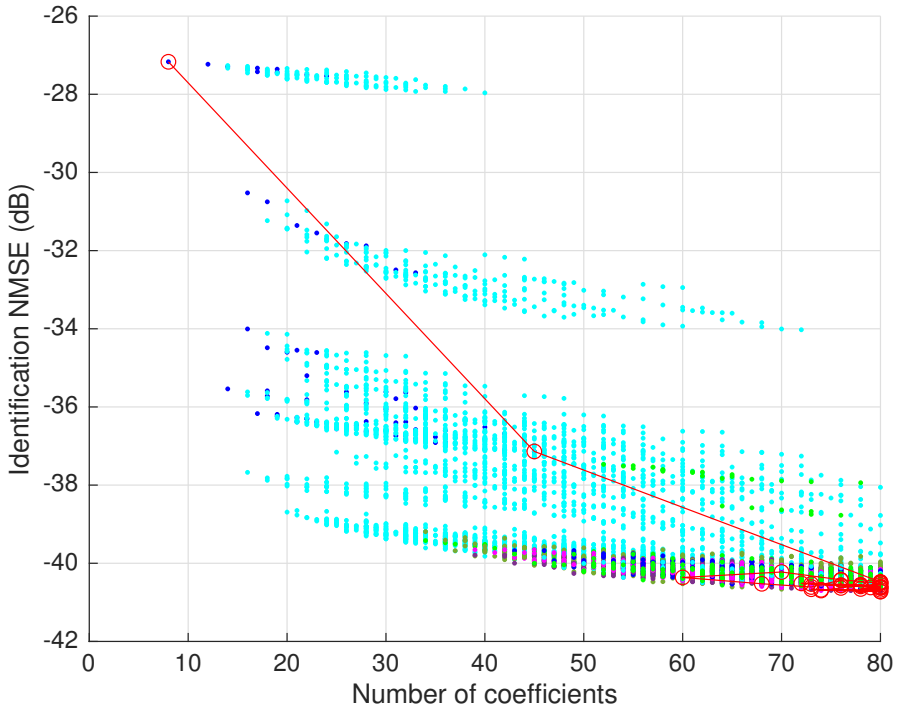

Figure 4.4 Plot of the HC when the number of samples is set to 11427. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

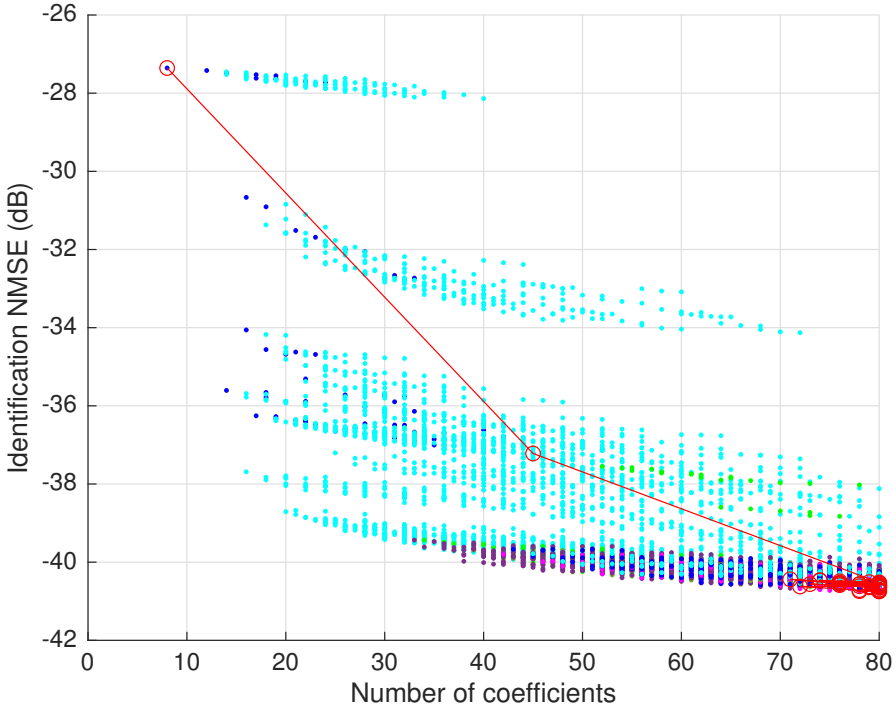

Figure 4.5 Plot of the HC when the number of samples is set to 13529. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

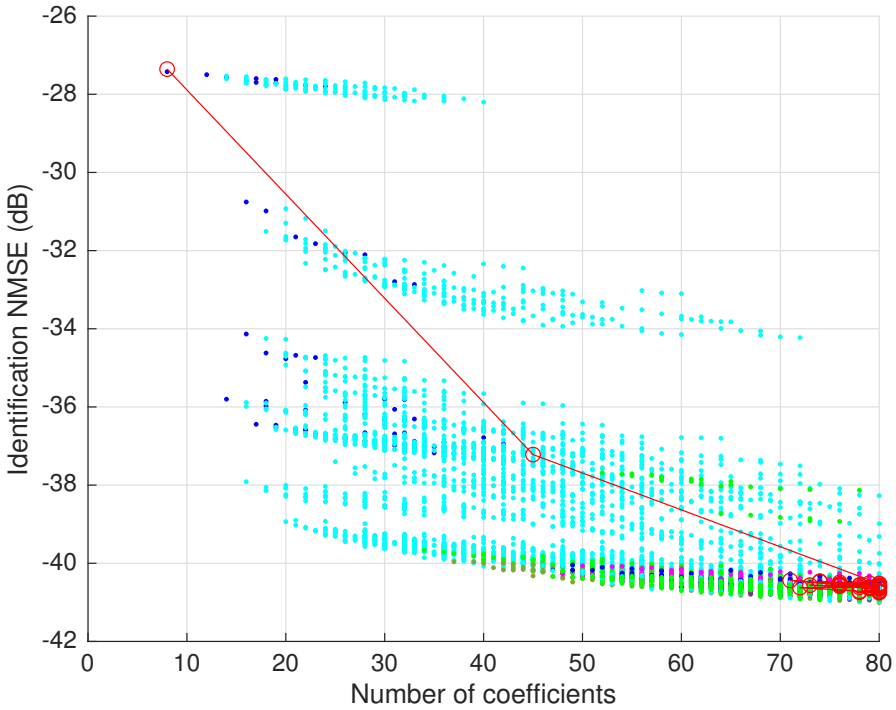

Figure 4.6 Plot of the HC when the number of samples is set to 15851. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

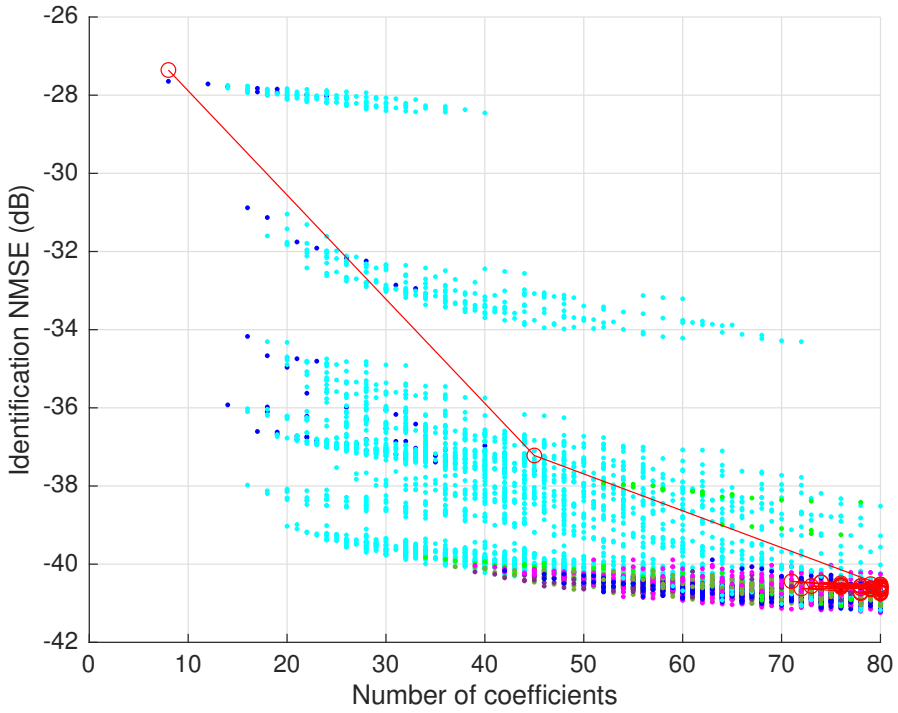

Figure 4.7 Plot of the HC when the number of samples is set to 18431. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

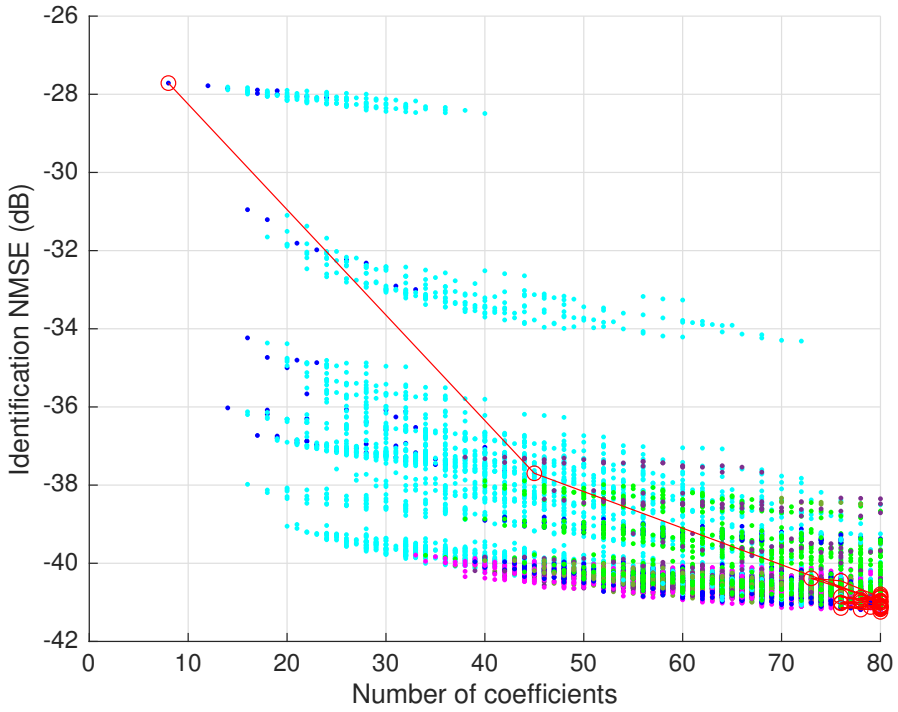

Figure 4.8 Plot of the HC when the number of samples is set to 23225. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

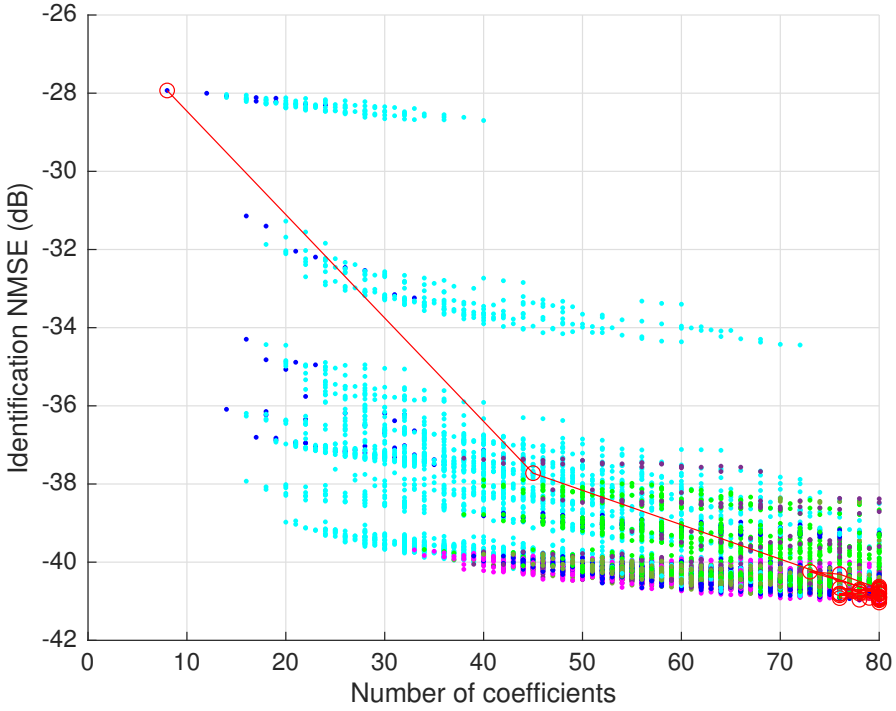

Figure 4.9 Plot of the HC when the number of samples is set to 33177. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

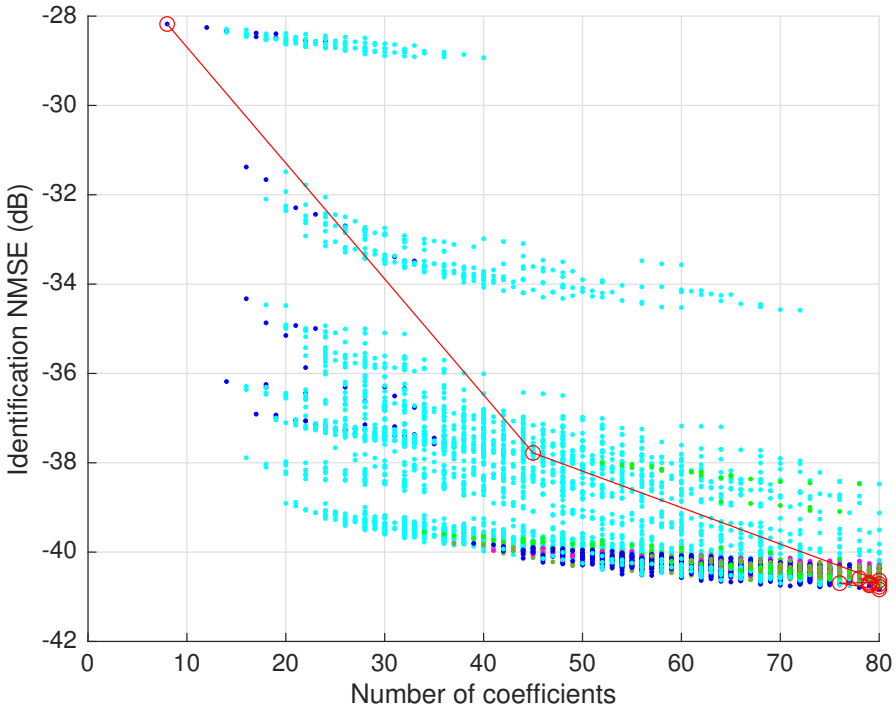

Figure 4.10 Plot of the HC when the number of samples is set to 55295. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

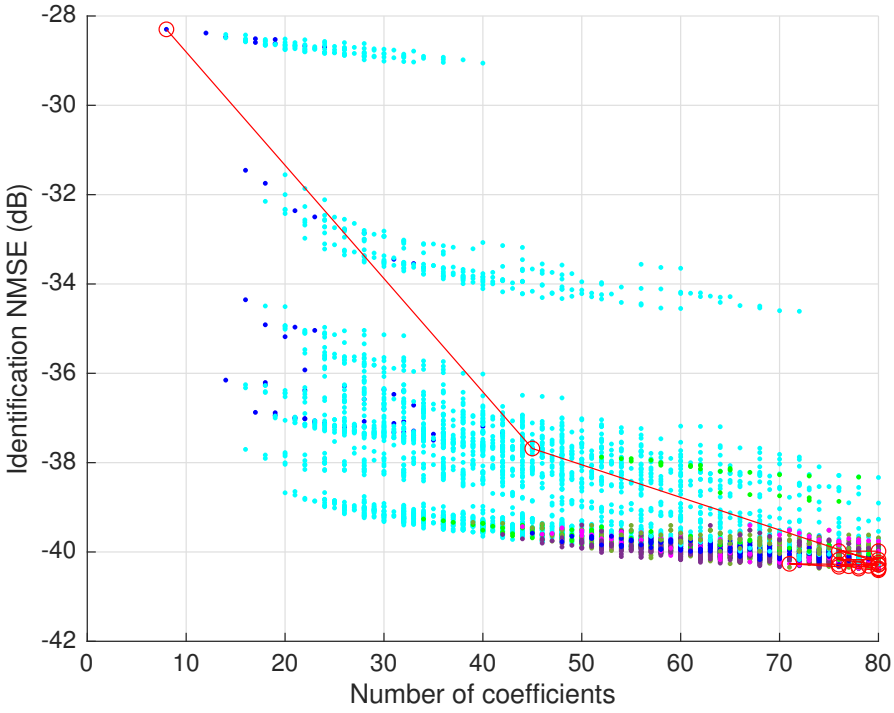

Figure 4.11 Plot of the HC when the number of samples is set to 70042. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

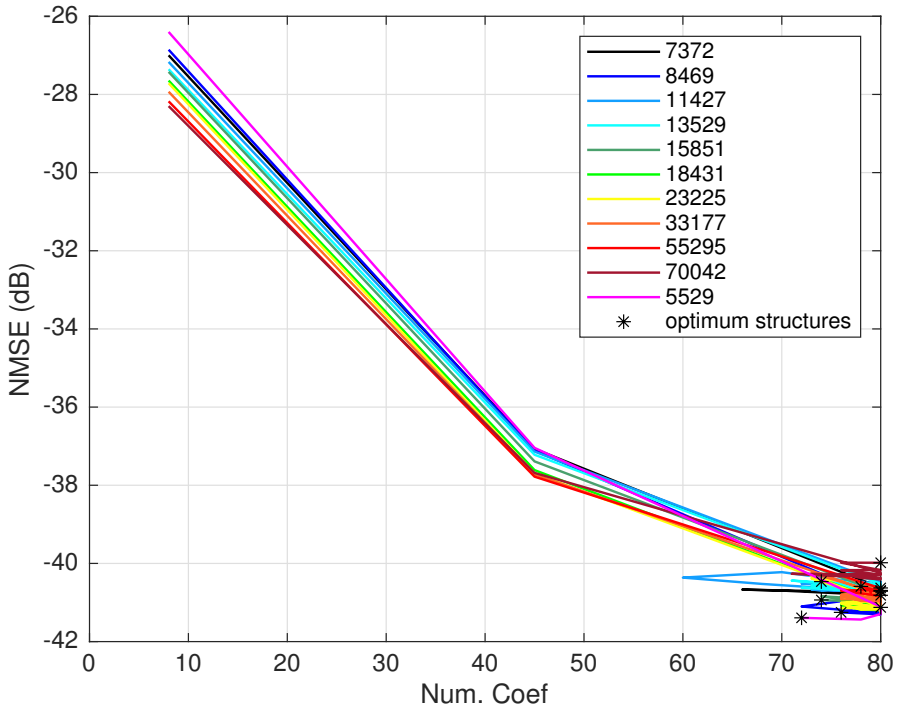

Figure 4.12 Optimum paths of all the simulations varying the M from 5529 to 70042 samples. Every optimum solution marked with a black asterisk.

As for the most of the simulations and experiments, the maximum number of coefficients is going to be much less than 400 coefficients. The relative minimum that appears at the bottom of the belly of Figure 4.18 using the BIC as stopping rule is unlikely going to be reached. The following subsections will be based on the argument that solutions will be usually found close to the limit of maximum coefficients allowed. The main goal of this section is the pursue of an improved-performace HC-based algorithm, specially in terms of ToC.

4.2.1 A Review of the Initial Model

For the next section, a initial model will be needed. The initial model is just a starting point, a first neighbor of the big iterative search. Just for convenience and to eliminate the variability that this could bring, the same will be used in the simulations in section 4.4. However, along this section, the configuration of the initial model will be tested.

$$K_a = 1, L_a = 1$$
$$K_b = 1, L_b = 1, M_b = 1$$
$$K_c = 1, L_c = 1, M_c = 1$$
$$NMSE_{dB} = -26.4006$$
$$NUMCOEF = 8$$

All the solutions found by the HC algorithm up until now were in the second half of the possible range of coefficients. An example of this is shown in 4.20. As one of the main purposes of this work is to find the swiftest algorithm for DPD, the next step was to find a starting point which saved the algorithm from calculating useless and less-complex models which may be discarded later. In other words, the next experiment's goal is to prove how the first neighbor determines the ToC and what recommended rules may be implemented to find an appropriate starting point, more complex than a simple starting point but closer to a solution.

The first simulation was configured with the initial model explained above and the maximum number of coefficients was 80, as in [2]. The improvement in NMSE passing the 80 coefficients is going to be considered impractial, as the The range delimited for the parameters are also detailed at the beginning of this section.

The result of the simulation is shown in Figure 4.19. The HC algorithm took 11.63 minutes to come up with the solution. A comparison with different simulations changing their initial models appears in Figure 4.20. The result are always in the range betweeen 72 and 80 coefficients, but the time of calculation was clearly reduced when the initial model was inside the range of 40 to 60 coefficients.

There were only 3 different solutions for the 11 simulations:

$$K_a = 5, L_a = 1$$
$$K_b = 1, L_b = 2, M_b = 4$$
$$K_c = 3, L_c = 3, M_c = 4$$
$$NMSE_{dB} = -41.3921$$
$$NUMCOEF = 72$$

$$K_a = 5, L_a = 5$$
$$K_b = 1, L_b = 1, M_b = 4$$
$$K_c = 4, L_c = 1, M_c = 4$$
$$NMSE_{dB} = -41.4955$$
$$NUMCOEF = 76$$

$$K_a = 5, L_a = 3$$
$$K_b = 1, L_b = 5, M_b = 4$$
$$K_c = 4, L_c = 1, M_c = 4$$
$$NMSE_{dB} = -41.5153$$
$$NUMCOEF = 80$$

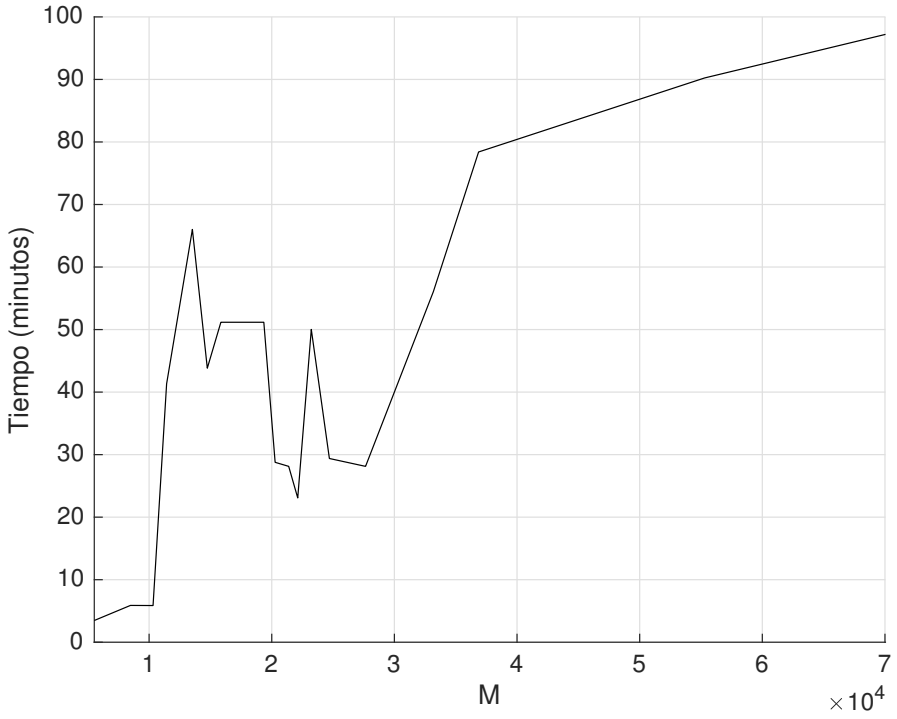

Figure 4.13 Time of calculation against the number of samples used in the optimum model identification.

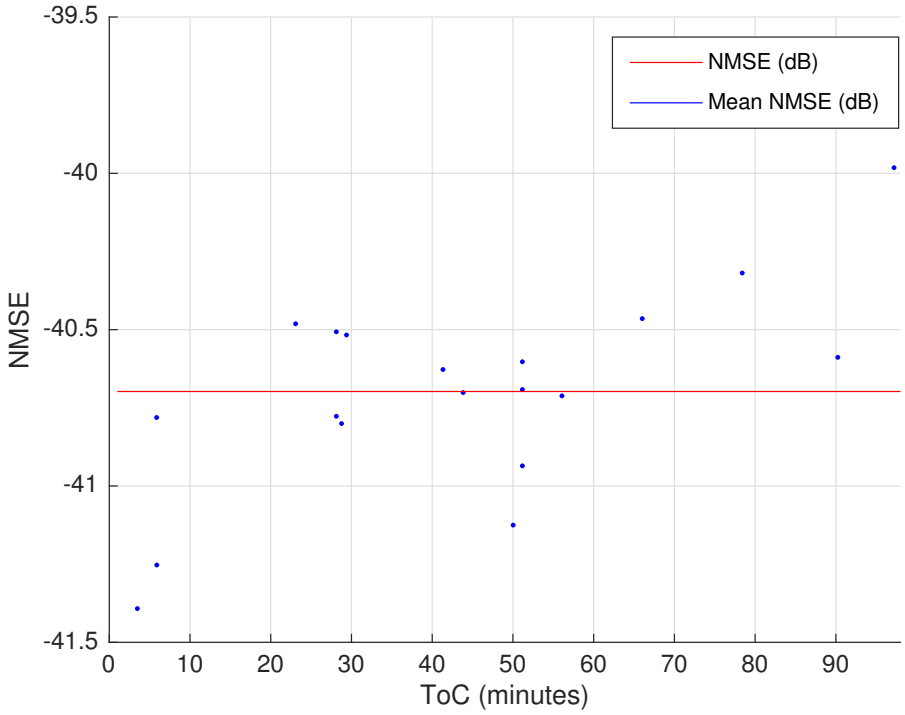

Figure 4.14 NMSE against time of calculation. The mean NMSE is represented as a constant blue line.

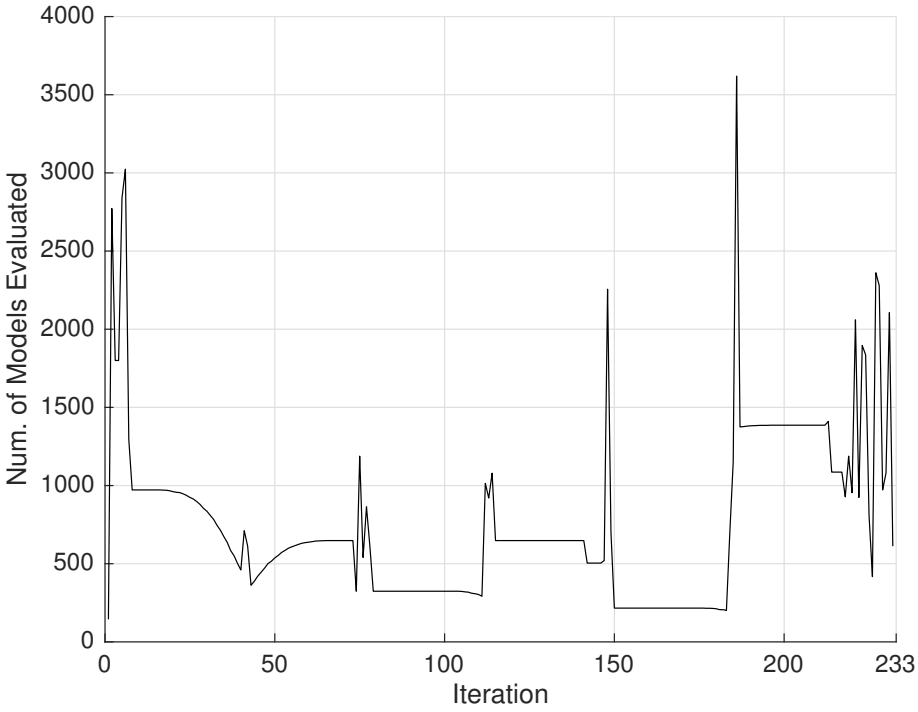

Figure 4.15 Number of models calculated per iteration or the size of all the neighborhoods of a search limited to a maximum of 400 coefficients per model.

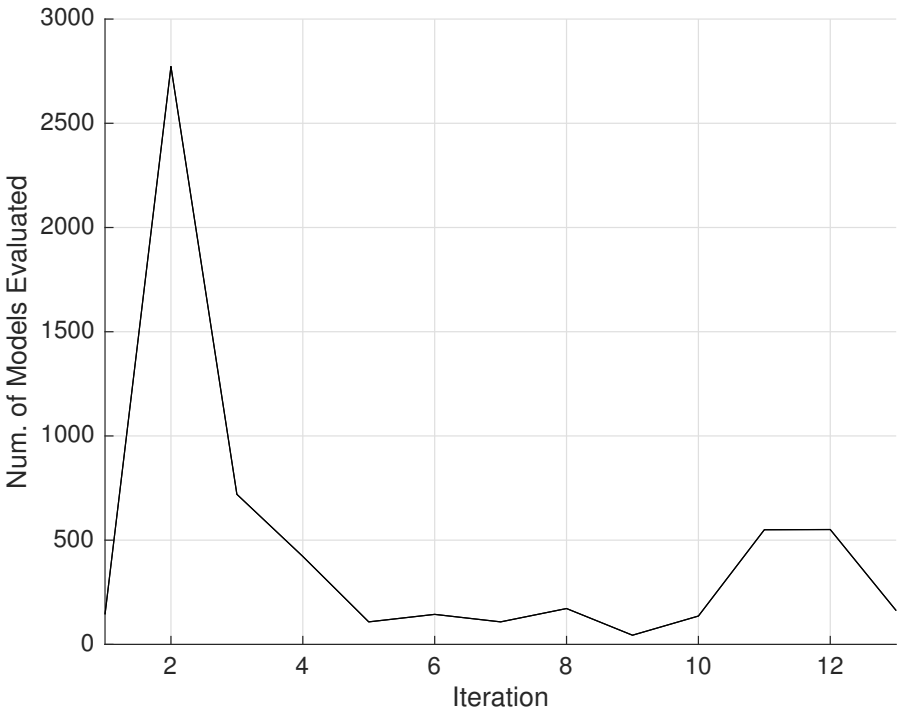

Figure 4.16 Number of models calculated per iteration or the size of all the neighborhoods of a search limited to a maximum of 400 coefficients per model.

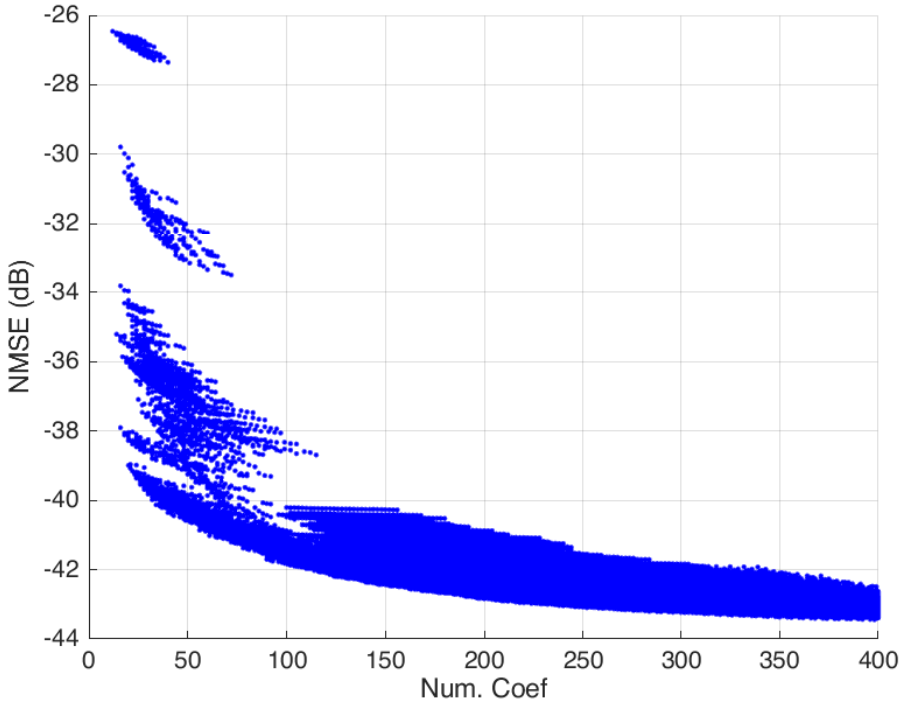

Figure 4.17 Plot of the first experiment simulation. Number of coefficients against merit value of the HC+BIC, limited to 400 coefficients..

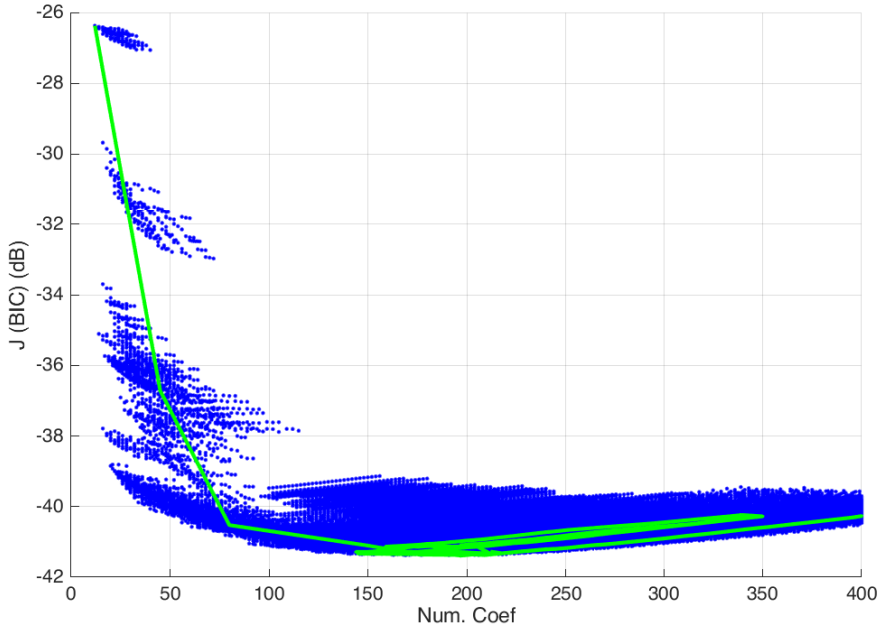

Figure 4.18 Plot of the first experiment simulation. Number of coefficients against merit value of the HC+BIC, limited to 400 coefficients. The optimums path followed by the HC algorithm in light green..

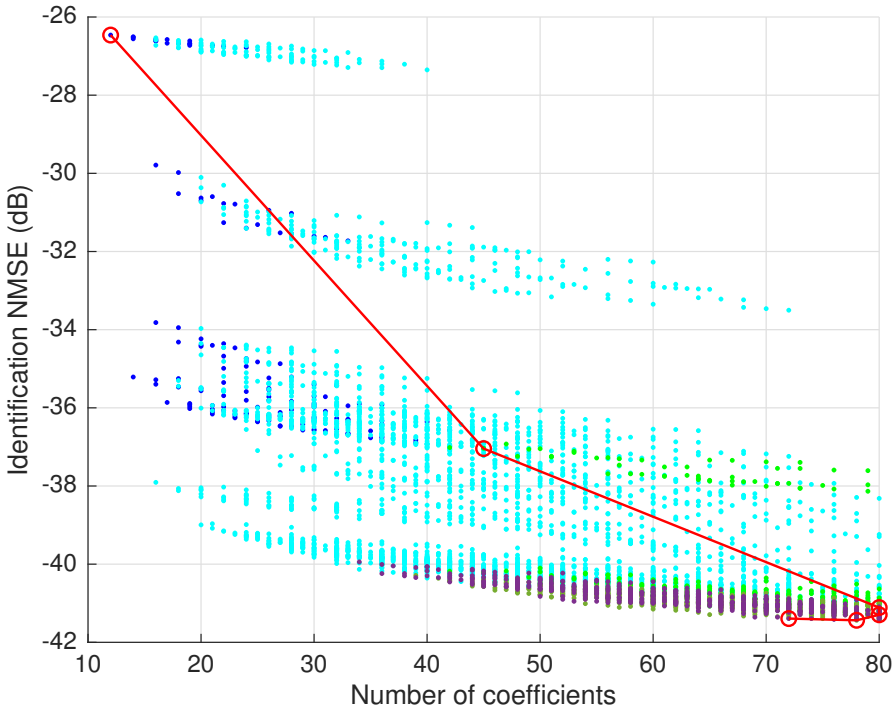

Figure 4.19 Plot of the first experiment simulation. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

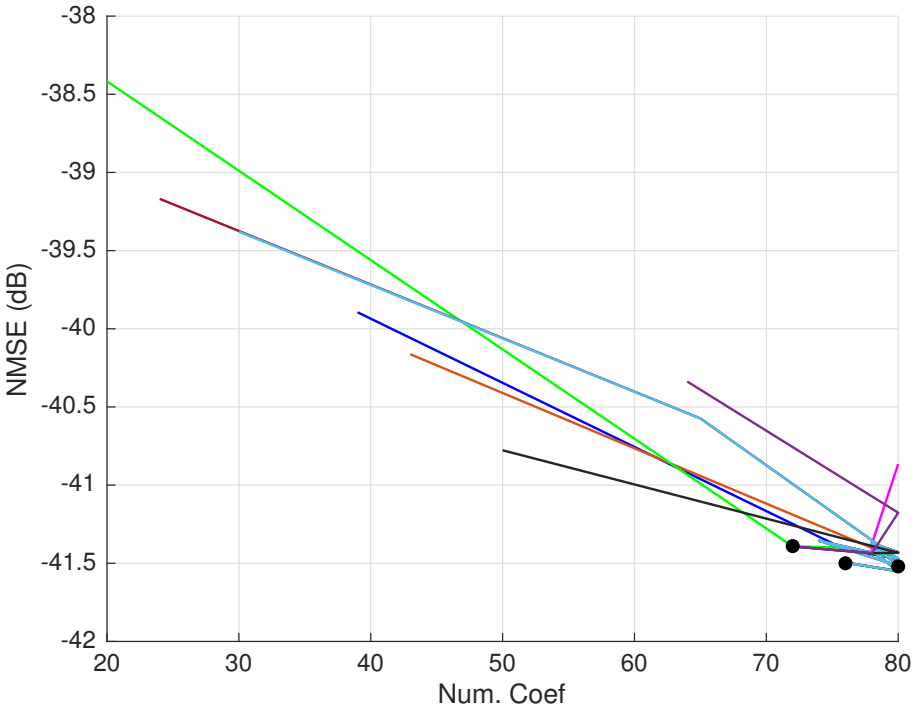

Figure 4.20 Comparison of the optimums path of several simulations limited to 80 coefficients and with the same conditions, but changing the initial model. The solutions for all the simulations represented as black dots.

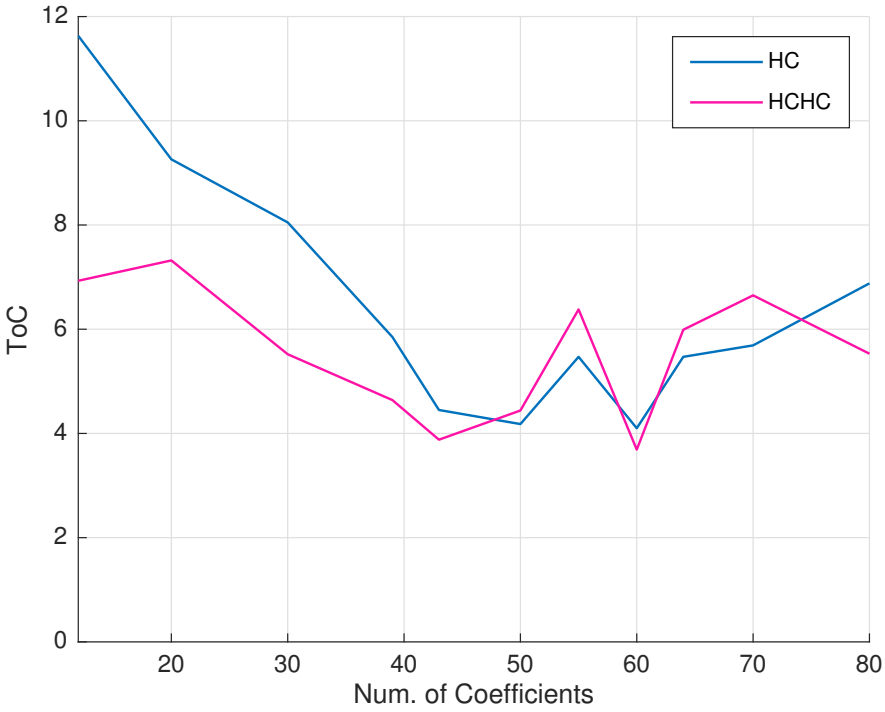

Figure 4.21 Comparison of the ToC of several simulations limited to 80 coefficients in blue and delimited between 40 and 80 in magenta, with the same conditions and varying the initial model.

4.2.2 Halved-Complexity HC

As a consequence of the results obtained in prior simulations, new ones were carried out delimiting the number of coefficients, not only the maximum but also the minimum. This was denoted as the Halved-Complexity HC (HCHC). The minimum number of coefficients would be half the maximum. A comparison between the HC doing an exhaustive search and the HCHC is shown in Figure 4.21. The mean ToC was reduced from 6.46 to 5.54 minutes. This proves how constraining the number of coefficients can be an effective way to reduce the computational load of the search. There are other proposals like [2], although this one does not add extra complexity to the algorithm.

4.3 Power Sweep

Another test was performed to analyse the behavior of the algorithm when the input power was gradually increased from -30 to -20 dBm with 1 decibel steps generating 11 different datasets. The number of samples (M) used on each iteration was 55000. The components of this testbench were:

- One SMW200A vector signal generator (VSG) from Rohde & Schwarz
- One BLD6G22L-50 Demo Board Doherty PA from NXP Semiconductors
- One FSW26 vector signal analyzer (VSA) from Rohde & Schwarz

The configuration of the testbench was:

- -30 to -20 dBm input power.
- 15 MHz LTE signal under test.
- 55000 signal samples used for model identification, which corresponds whith a $\mu_{BIC} = 0.0009166$ dB

The 11 datasets that were generated with the testbench were used as input arguments of the HC algoritm. All the final model structures are listed below (Figure 4.22 to 4.32).

For -30 dBm, after 47 iterations:

$$K_a = 4, L_a = 2$$
$$K_b = 3, L_b = 4, M_b = 4$$
$$K_c = 1, L_c = 1, M_c = 2$$
$$NMSE_{dB} = -34.4491$$
$$NUMCOEF = 79$$

For -29 dBm, after 11 iterations was:

$$K_a = 2, L_a = 5$$
$$K_b = 4, L_b = 1, M_b = 3$$
$$K_c = 1, L_c = 5, M_c = 6$$
$$NMSE_{dB} = -34.5978$$
$$NUMCOEF = 78$$

For -28 dBm, after 20 iterations was:

$$K_a = 2, L_a = 5$$
$$K_b = 4, L_b = 1, M_b = 3$$
$$K_c = 1, L_c = 5, M_c = 6$$
$$NMSE_{dB} = -34.5085$$
$$NUMCOEF = 78$$

For -27 dBm, after 28 iterations was:

$$K_a = 7, L_a = 4$$
$$K_b = 2, L_b = 3, M_b = 4$$
$$K_c = 1, L_c = 1, M_c = 4$$
$$NMSE_{dB} = -34.1366$$
$$NUMCOEF = 80$$

For -26 dBm, after 4 iterations was:

$$K_a = 1, L_a = 5$$
$$K_b = 3, L_b = 3, M_b = 4$$
$$K_c = 1, L_c = 3, M_c = 4$$
$$NMSE_{dB} = -32.1032$$
$$NUMCOEF = 76$$

For -25 dBm, after 11 iterations was:

$$K_a = 1, L_a = 5$$
$$K_b = 7, L_b = 1, M_b = 2$$
$$K_c = 1, L_c = 5, M_c = 6$$
$$NMSE_{dB} = -33.2911$$
$$NUMCOEF = 76$$

For -24 dBm, after 4 iterations was:

$$K_a = 1, L_a = 5$$
$$K_b = 3, L_b = 3, M_b = 4$$
$$K_c = 1, L_c = 3, M_c = 4$$
$$NMSE_{dB} = -32.5060$$
$$NUMCOEF = 76$$

For -23 dBm, after 4 iterations was:

$$K_a = 1, L_a = 5$$
$$K_b = 3, L_b = 3, M_b = 4$$
$$K_c = 1, L_c = 3, M_c = 4$$
$$NMSE_{dB} = -32.9390$$
$$NUMCOEF = 76$$

For -22 dBm, after 21 iterations was:

$$K_a = 1, L_a = 5$$
$$K_b = 2, L_b = 5, M_b = 4$$
$$K_c = 4, L_c = 1, M_c = 2$$
$$NMSE_{dB} = -31.1201$$
$$NUMCOEF = 76$$

For -21 dBm, after 19 iterations was:

$$K_a = 5, L_a = 5$$
$$K_b = 2, L_b = 1, M_b = 2$$
$$K_c = 1, L_c = 5, M_c = 6$$
$$NMSE_{dB} = -33.7730$$
$$NUMCOEF = 80$$

And for -20 dBm, after 4 iterations was:

$$K_a = 1, L_a = 5$$
$$K_b = 3, L_b = 3, M_b = 4$$
$$K_c = 1, L_c = 3, M_c = 4$$
$$NMSE_{dB} = -31.6498$$
$$NUMCOEF = 76$$

It is evident, judging by the results represented in Figure 4.34, how each 1-dB step from -30 to -20 dBm decreases the NMSE as the PA enters more a more in a nonlinear region. However, as it can be seen in Figure 4.33, the number of structures calculated tends to decrease when the input power is higher.

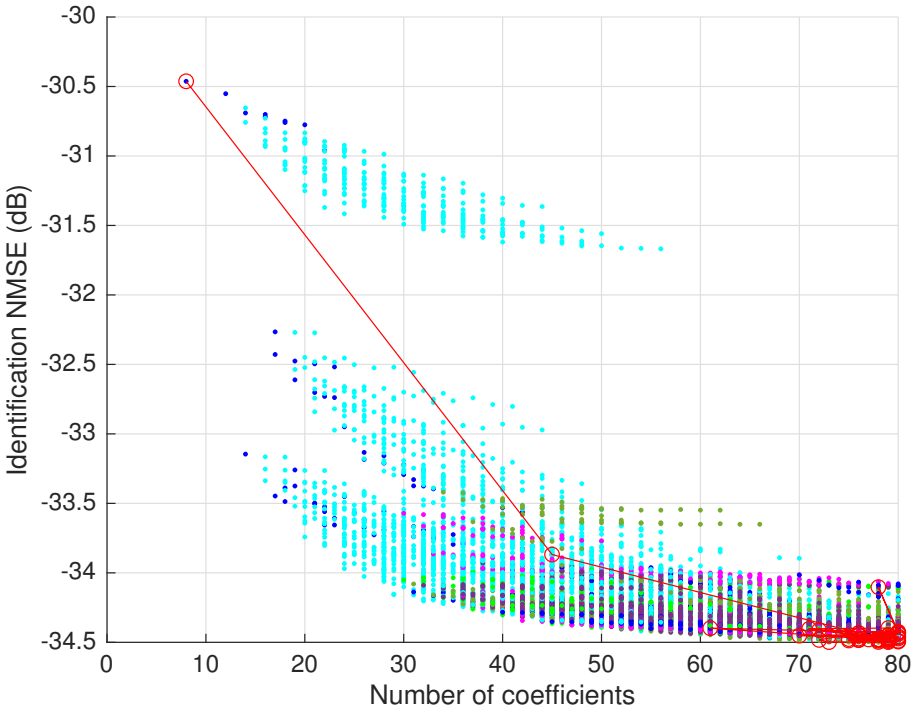

Figure 4.22 Plot of the testbench when the input power is set to -30 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

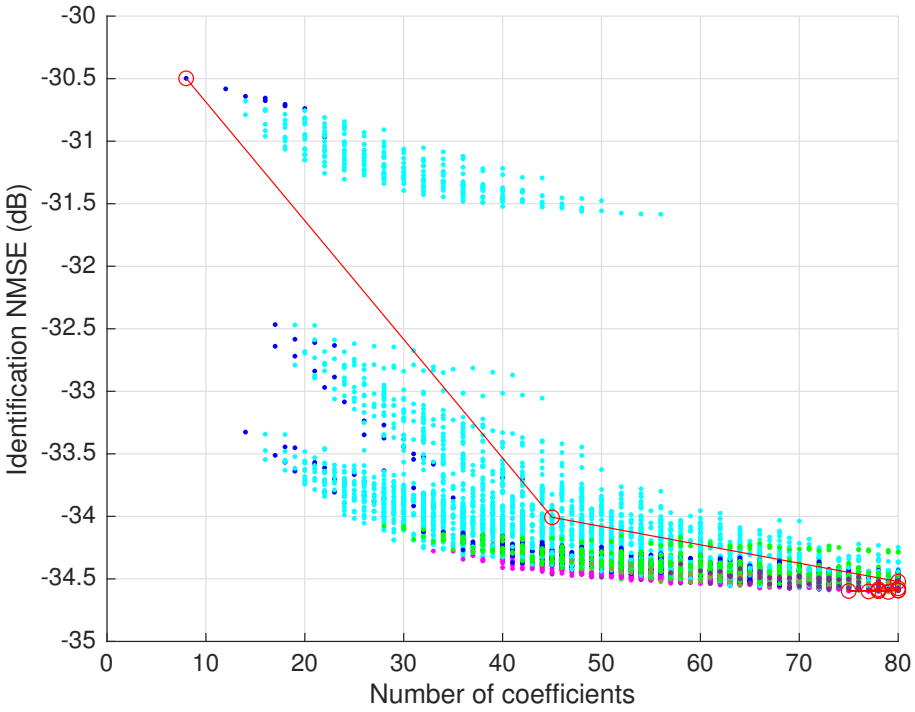

Figure 4.23 Plot of the testbench when the input power is set to -29 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

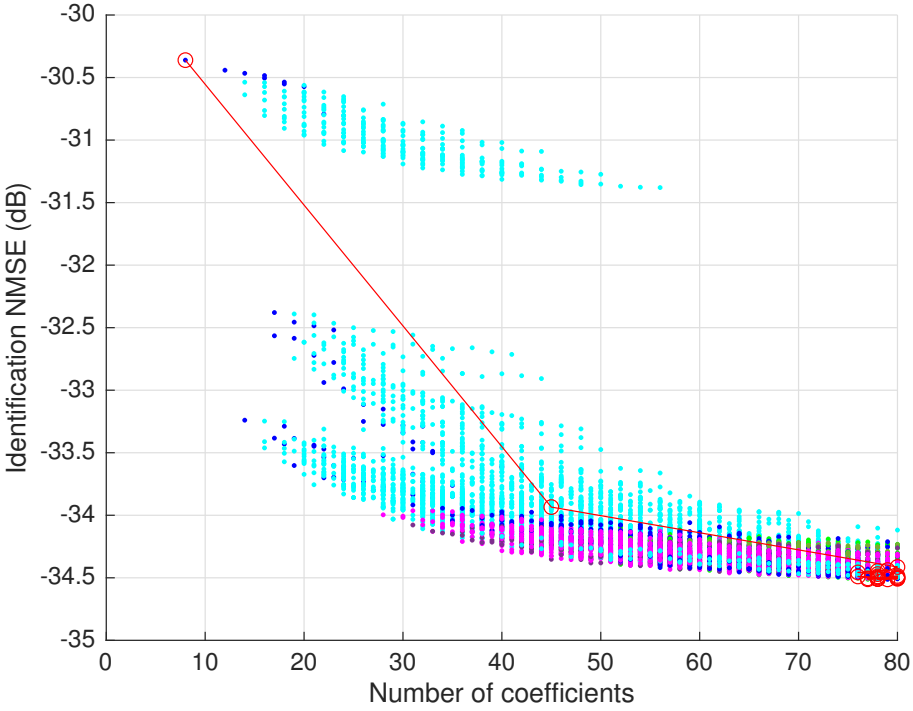

Figure 4.24 Plot of the testbench when the input power is set to -28 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

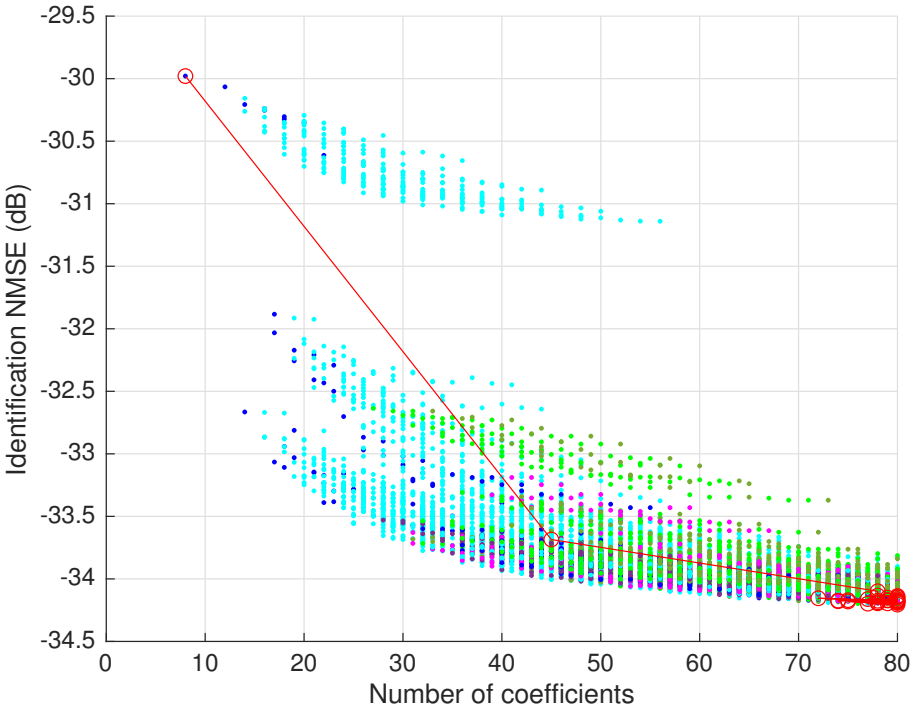

Figure 4.25 Plot of the testbench when the input power is set to -27 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

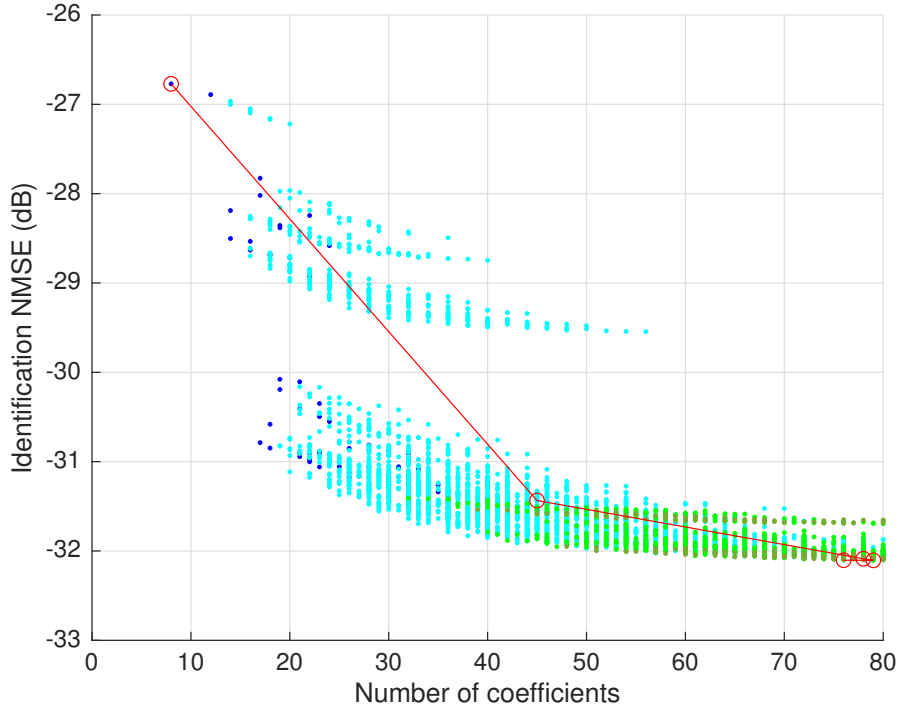

Figure 4.26 Plot of the testbench when the input power is set to -26 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

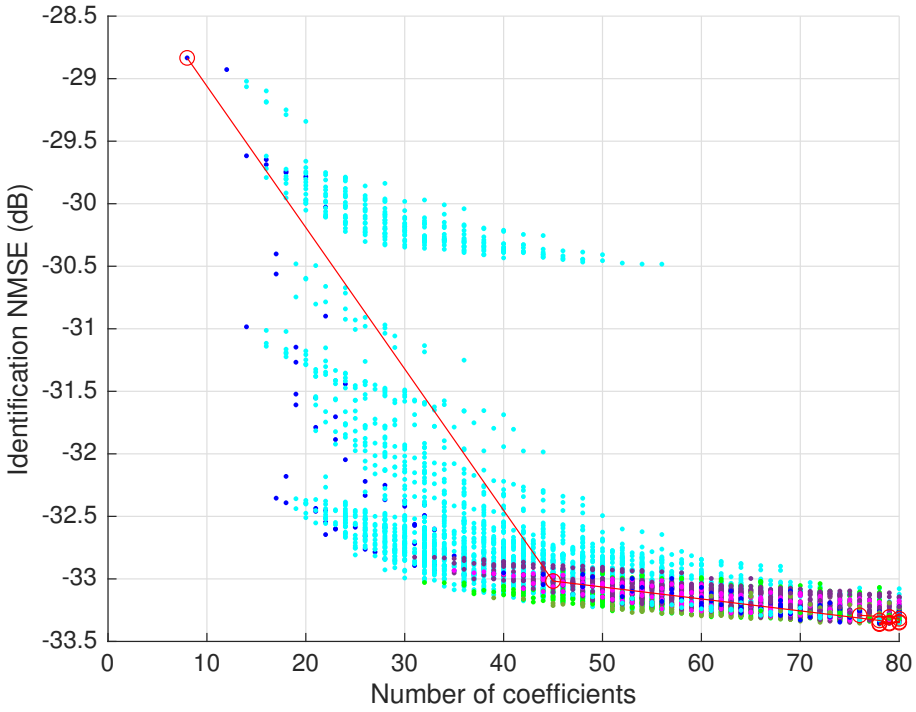

Figure 4.27 Plot of the testbench when the input power is set to -25 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

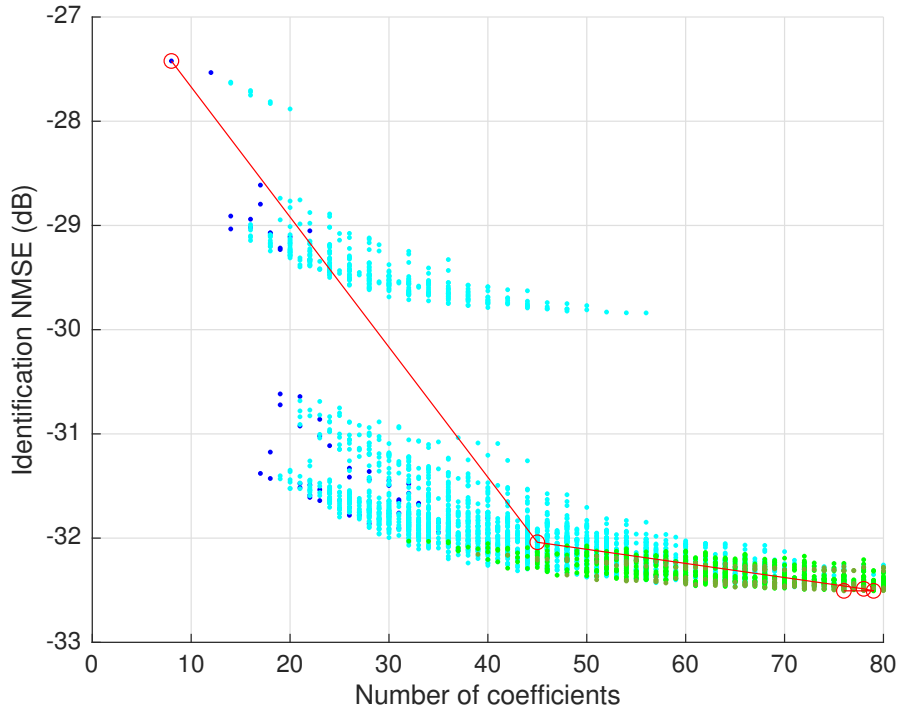

Figure 4.28 Plot of the testbench when the input power is set to -24 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

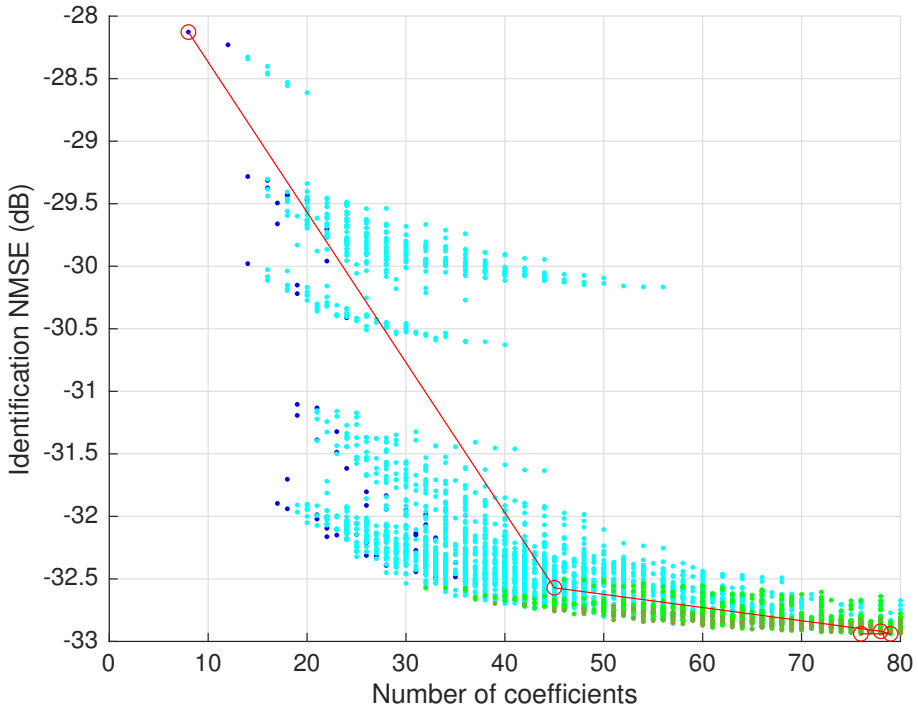

Figure 4.29 Plot of the testbench when the input power is set to -23 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

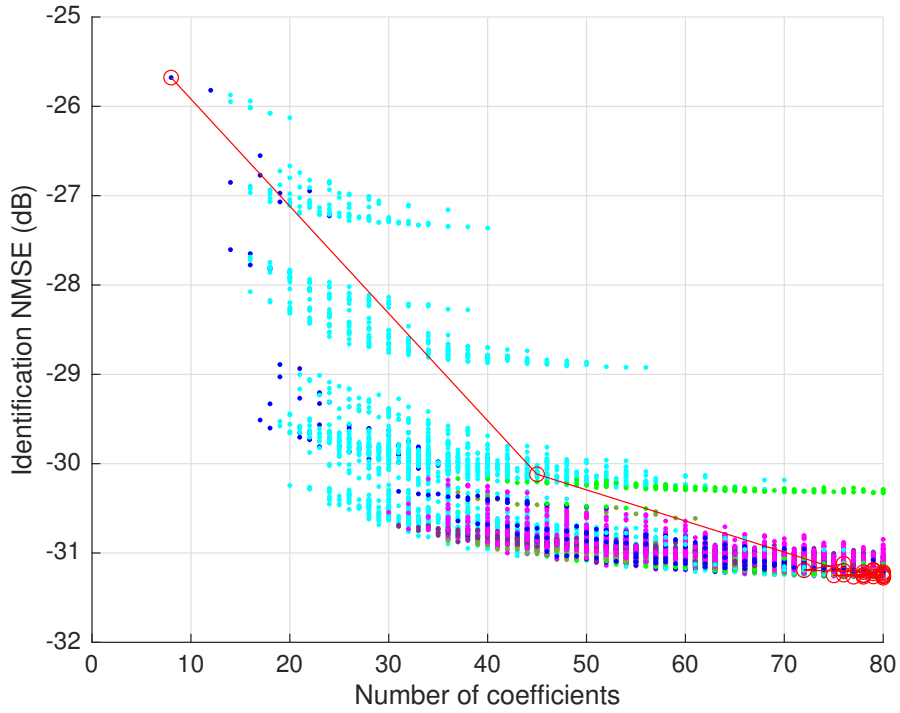

Figure 4.30 Plot of the testbench when the input power is set to -22 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

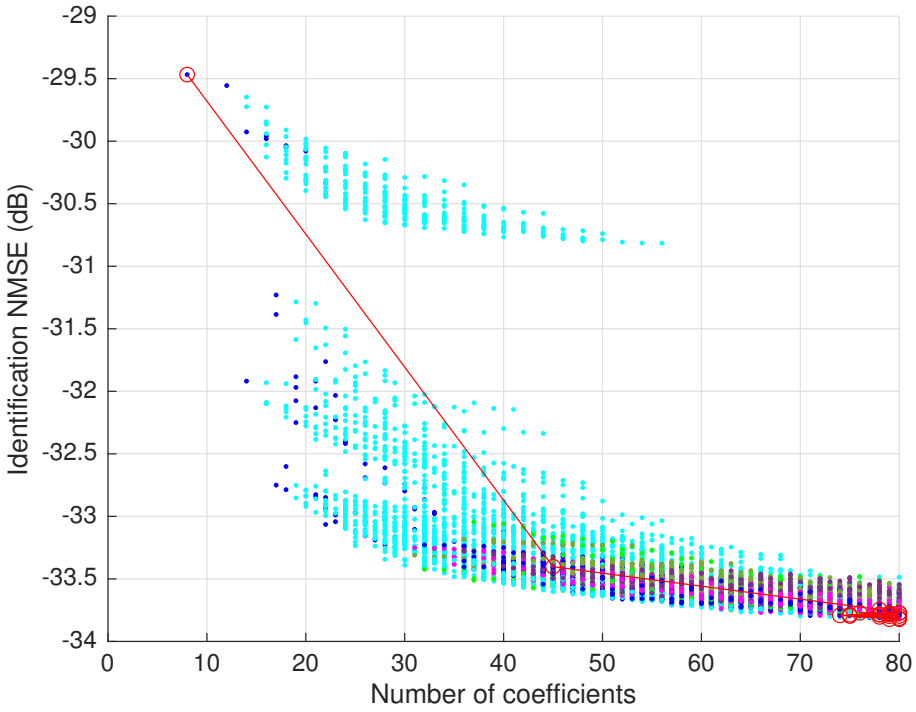

Figure 4.31 Plot of the testbench when the input power is set to -21 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

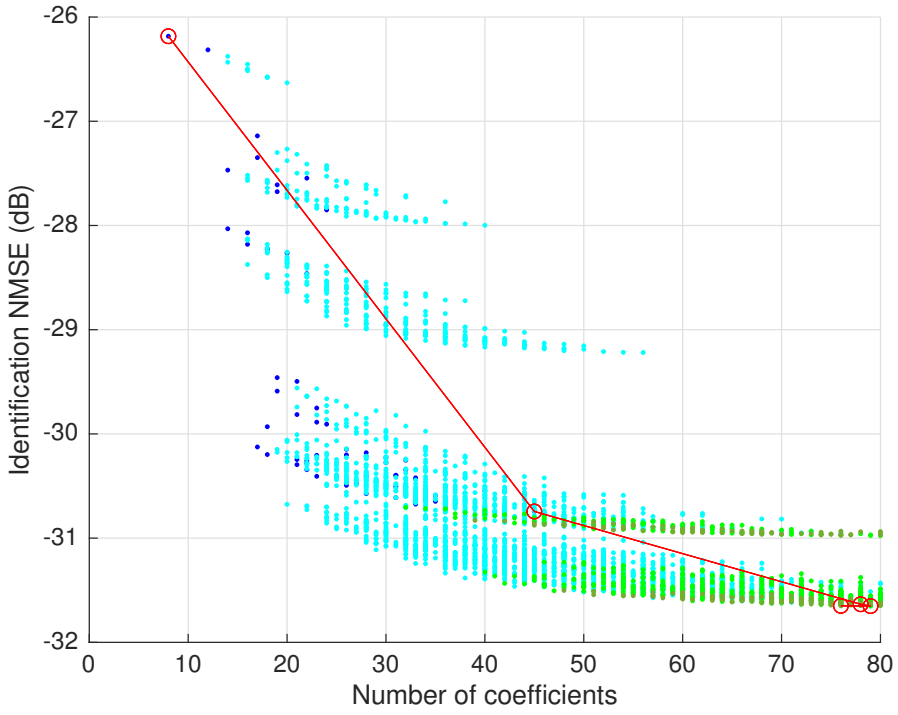

Figure 4.32 Plot of the testbench when the input power is set to -20 dBm. Number of coefficients against NMSE of the HC, limited to 80 coefficients. Neighborhoods are represented in different colors. The optimums path to the solution in red.

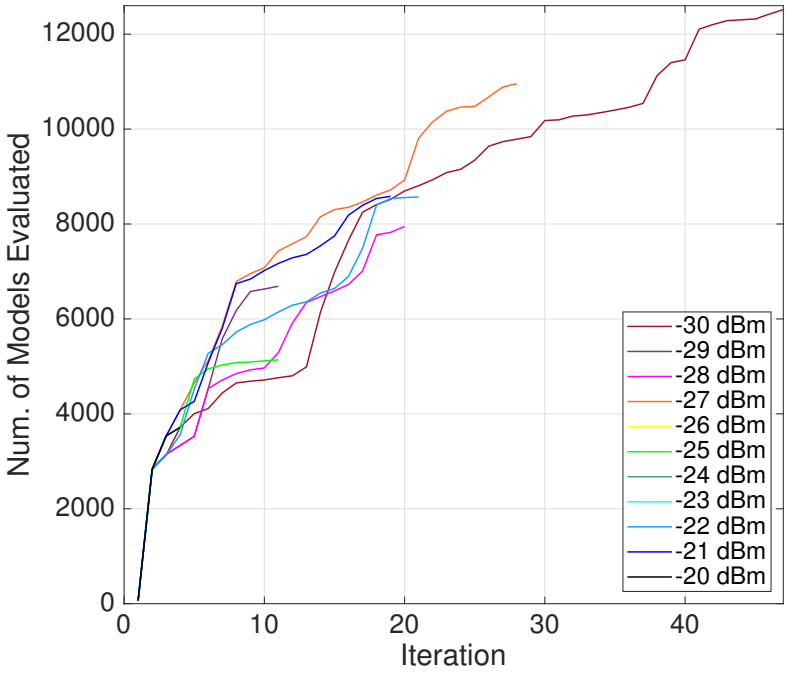

Figure 4.33 Number of models calculated per iteration as an accumulative sum, representing every step of the power sweep in a different color.

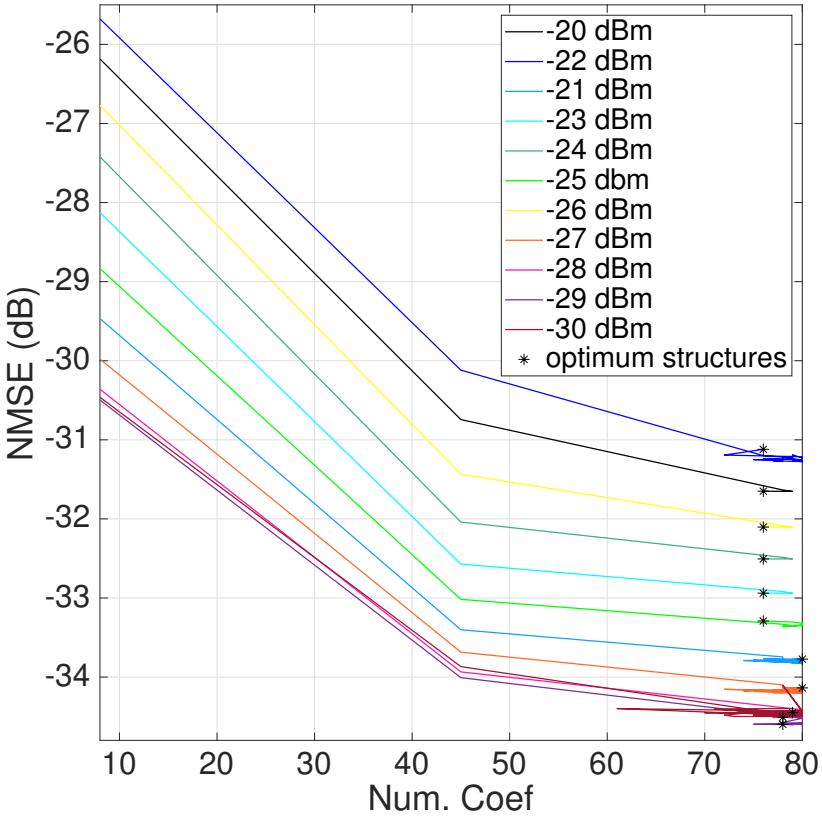

Figure 4.34 Optimum paths of all the simulations of the power sweep from -30 to -20 dBm. Every optimum solution marked with a black asterisk.

4.4 BIC: HC vs DOMP

The experiment that compares the performane of the HC and the DOMP algorithm will be described in this section.

4.4.1 Simulation Results

The parameters of the optimum model after 13 iterations found by the HC algorithm were:

$$K_a = 5, L_a = 2$$
$$K_b = 3, L_b = 4, M_b = 4$$
$$K_c = 4, L_c = 5, M_c = 4$$
$$NMSE_{dB} = -41.0694$$
$$NUMCOEF = 174$$

The DOMP beside the BIC stopping rule was applied to the model with the maximum number of parameters found by the HC:

$$K_a = 6, L_a = 3$$
$$K_b = 4, L_b = 5, M_b = 4$$
$$K_c = 5, L_c = 5, M_c = 4$$
$$NMSE_{dB} = -42.5029$$
$$NUMCOEF = 244$$

And the optimum model obtained by the DOMP with BIC was:

$$K_a = 6, L_a = 3$$
$$K_b = 4, L_b = 5, M_b = 4$$
$$K_c = 5, L_c = 4, M_c = 6$$
$$NMSE_{dB} = -42.2977$$
$$NUMCOEF = 124$$

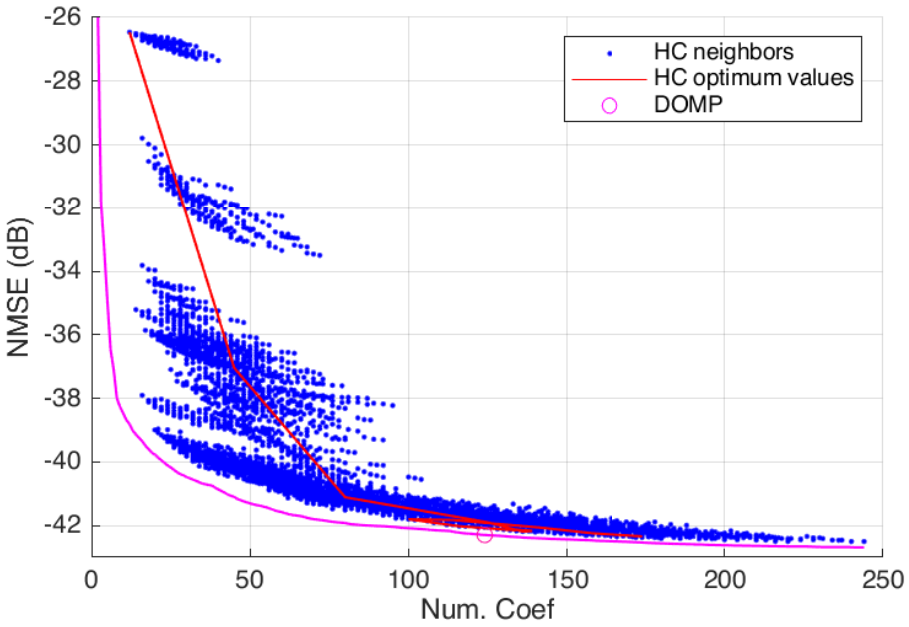

Figure 4.35 Plot of the first experiment simulation. Number of coefficients against NMSE of the HC and the DOMP.

The space of all the possible solutions calculated along the seven iterations that took the simulator to find the local optimum is represented in Figure 4.35. The DOMP is a more powerful algorithm and was able to obtain an optimum model with 50 coefficients less and roughly -1 dB compared to the HC algoritm. The DOMP takes also less time to estimate the best regressors of the optimum GMP model. The HC optimum was found in 21.59 minutes, while the DOMP only took 1.53 minutes to find it.

4.4.2 Empirical Results

First of all, the components of the testbench will be listed below:

- One SMU200A vector signal generator (VSG) from Rohde & Schwarz
- One PXA-N9030A vector signal analyzer (VSA) from Keysight
- One dc power supply
- One DUT: one cascade of two Mini-Circuits TVA-4W-422A+ preamplifiers and one continuous mode class J PA operating at a center frequency of 850 MHz

The configuration of the testbench was:

- 31 dBm output power characterized by 3.5 dB of gain compression
- 5G-NR signal under test with 20 MHz of bandwidth, 30 kHz carrier separation and 92.16 MHz of sampling rate.

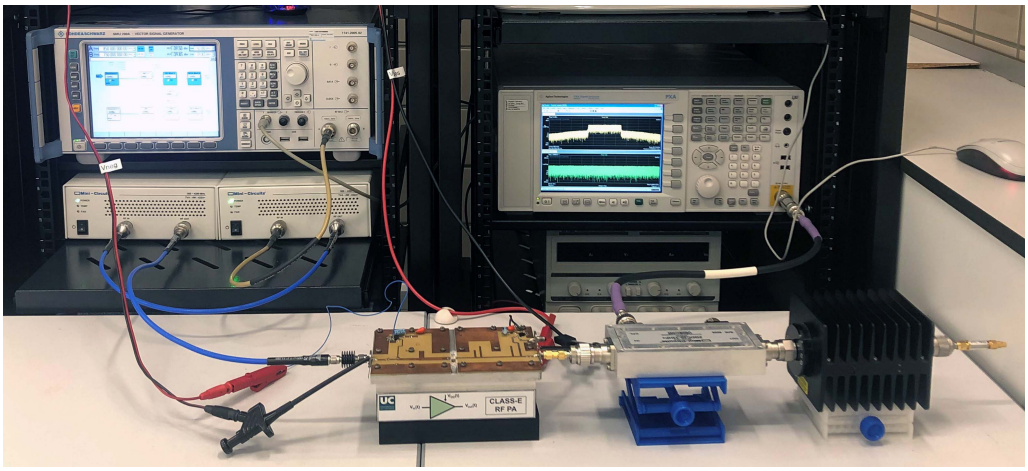

Figure 4.36 Testbench for the first esperiment.

- 5500 signal samples used for model identification, which corresponds whith a $\mu_{BIC} = 0.0073$ dB

All the setup is shown in Figure 4.36.

The optimum models for both HC and DOMP were used to generate a predistorted signal, whose linearization performance and power spectral density (PSD) are shown in Table 4.1, respectively. The effect of both techniques is evidenced by a reduction of the in-band distortion, achieving an enhancement of over 20 dB in NMSE and an error vector magnitude (EVM) of about 1% in the linearized signal. The adjacent channel power ratio (ACPR) improvement is illustrated by the decrease in spectral regrowth, achieving a level of below -52 dB in all the linearized cases. Note that the power added efficiency (PAE) of the PA is the same with and without DPD, which confirms that DPD techniques are capable of reducing nonlinearity while still keeping the same level of power efficiency.

These results can be seen graphically in Figure 4.37.

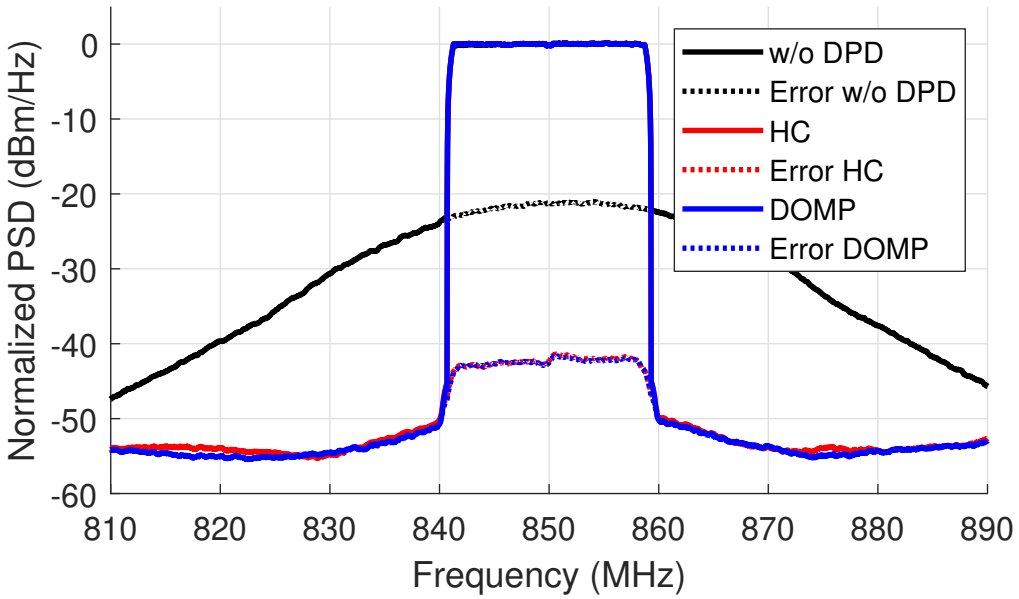

Figure 4.37 Normalized power spectral density of the PA output without DPD and with DPD regressed witht the number of coefficients indicated by the BIC.

Table 4.1 Measured performance in terms of NMSE, ACPR, EVM and number of coefficients for the PA with and without DPD.

Experiment	NMSE (dB)	ACPR -1/+1 (dBc)	EVM (%)	Number of coef.	PAE (%)
w/o DPD	-19.5	-29.1 / -27.0	8.6	-	24
HC DPD	-41.4	-53.4 / -52.8	1.0	206	24
DOMP DPD	-41.5	-53.8 / -53.1	1.1	120	24

5 Conclusion and Future Research Lines

To write well, express yourself like the common people, but think like a wise man.

ARISTOTLE

An extensive analysis of the application of Hill Climbing Heuristic to digital predistortion has been presented in this work [2]. The exhaustive search algorithm has honoured its name the HC algorithm has demonstrated to be heavy in terms of computational load. Very long simulations have been done with not too explanatory results. Its implementation has been also complicated, when other algorithms such as the DOMP [20] can be much more straightforward.

However, experimental results helped recognizing the utility of the structures obtained by the HC when compared with no DPD and the DOMP, although the complexity of the structure of this second algorithm is more reduced.

More research must be done about limited-complexity HC algorithms where the main focus should be on reducing the time of calculation needed to find the fittest model structure, as well as testing a wider range of stopping rules which could point to the appropriate criteria for this type of DPD model search.

In conclusion, this work should be considered an icebreaker in this area of study. Future research may find interesting to compare the HC algorithm discussed in this dissertation with genetic algorithms and other applications of machine learning to DPD.

6 Codes

Code 6.1 Main function of the algorithm, where the number of samples and the initial model are selected..

```
hold off, close all, clear all;

load('medida_cantabria_31_precalentamiento.mat')
x = xdpd-mean(xdpd);
y = ydpd-mean(ydpd);

n=0.225;

indices = sel_indices(x,y,n);
yp=y(indices);
xp=x(indices);

parameters=[1 1 1 1 1 1 1 1];

c=clock;
fecha=c(1)*10000+c(2)*100+c(3);
if c(4)==0
   hora="00"+c(5);
elseif c(4)<10
    hora="0"+((5)*100+c(5));
else
    hora=c(4)*100+c(5);
end

HC_Algorithm(parameters,fecha,hora,xp,yp);
```

Code 6.2 HC algorithm implementation. A model database matrix is used to hold all the unique model structures calculated..

```
function HC_Algorithm(coefs,fecha,hora,xp,yp)

M=length(yp);

numIteraciones=1;
modelDB=[];

tic;

model.type = 'GMP';
model.extension_periodica = 0;
model.grafica = 0;
model.h = [];
model.calculo = 'pinv';
model.pe = 1;
model.dc = 0;
model.cs = 0;
model.nmax = 200;

Ka=coefs(1); La=coefs(2);
Kb=coefs(3); Lb=coefs(4); Mb=coefs(5);
Kc=coefs(6); Lc=coefs(7); Mc=coefs(8);

model.Ka=[0:Ka];
model.La=ones(size(model.Ka))*La;
model.Kb=[1:Kb];
model.Lb=ones(size(model.Kb))*Lb;
model.Mb=ones(size(model.Kb))*Mb;
model.Kc=[1:Kc];
model.Lc=ones(size(model.Kc))*Lc;
model.Mc=ones(size(model.Kc))*Mc;

[model,toc] = model_gmp(yp, xp, model);

modelDB(1,:)=[numIteraciones,Ka,La,Kb,Lb,Mb,Kc,Lc,Mc,model.numcoef,model.nmse,
    HC_BIC_J(model.nmse,model.numcoef,M),toc];

modelDB_optimos=modelDB(1,:);
BM=modelDB(1,:);

while true

    oldJ=BM(12);

    [modelDB]=HC_Neighbors(model,modelDB,numIteraciones,yp,xp);

    [BM_aux]=HC_BestJ(modelDB,numIteraciones);

    if ~isempty(BM_aux)
        BM=BM_aux;

        modelDB_best_row=[numIteraciones,BM(2:13)];

        modelDB_optimos=cat(1,modelDB_optimos,modelDB_best_row);
```

```
        newJ=BM(12);

        model.Ka=[0:BM(2)];
        model.La=ones(size(model.Ka))*BM(3);
        model.Kb=[1:BM(4)];
        model.Lb=ones(size(model.Kb))*BM(5);
        model.Mb=ones(size(model.Kb))*BM(6);
        model.Kc=[1:BM(7)];
        model.Lc=ones(size(model.Kc))*BM(8);
        model.Mc=ones(size(model.Kc))*BM(9);
    end

    if oldJ==newJ
        break
    end

    numIteraciones=numIteraciones+1;

end

time=toc;

optimumModel=model;

disp("volcado de los modelos calculados en Excel.")

if length(modelDB(:,1))<100000
    disp("modelDB cabe en un Excel");
    writematrix(modelDB,"modelDB"+fecha+"_"+hora+".xls");
else
    disp("modelDB NO cabe en un Excel");
end

writematrix(modelDB_optimos,"modelDB_optimos"+fecha+"_"+hora+".xls");

end
```

Code 6.3 Function used to estimate all the possible neighbors of a designated model and add them to the model database. Here is where the complexity delimitations are implemented..

```
function [modelDB]=HC_Neighbors(model,modelDB,numIteraciones,y,x)
    MAXCOEF=80;

    for i=-1:1
        for j=-1:1
            for k=-1:1
                for l=-1:1
                    for m=-1:1
                        for n=-1:1
                            for o=-1:1
                                for p=-1:1
    Ka=model.Ka(end) + i;
    if Ka < 1
        Ka=1;
    elseif Ka>11
        Ka=11;
    end

    La=model.La(end) + j;
    if La < 1
        La = 1;
    elseif La>5
        La=5;
    end


    Kb=model.Kb(end) + k;
    if Kb < 1
        Kb = 1;
    elseif Kb>10
        Kb=10;
    end

    Lb=model.Lb(end) + l;
    if Lb < 1
        Lb = 1;
    elseif Lb>5
        Lb=5;
    end

    Mb=model.Mb(end) + m;
    if Mb<=Lb
        Mb=Lb+1;
    end
    if Mb<1
        Mb=1;
    elseif Mb>4
        Mb=4;
    end

    Kc=model.Kc(end) + n;
    if Kc < 1
        Kc = 1;
    elseif Kc>10
```

```
        Kc=10;
    end

    Lc=model.Lc(end) + o;
    if Lc < 1
        Lc = 1;
    elseif Lc>5
        Lc=5;
    end

    Mc=model.Mc(end) + p;
    if Mc<=Lc
        Mc=Lc+1;
    end
    if Mc<1
        Mc=1;
    elseif Mc>4
        Mc=4;
    end

    if Lb+Mb<Mc-1
        Lb=Lb+1;
    end

    modelDB_row=[Ka,La,Kb,Lb,Mb,Kc,Lc,Mc];
    NUMCOEF=(Ka+1)*(La+1)+Kb*(Lb+1)*Mb+Kc*(Lc+1)*Mc;
        if NUMCOEF<=MAXCOEF
            if ~any(ismember(modelDB(:,2:9),modelDB_row,'rows'))
                model2.type = 'GMP';
                model2.extension_periodica = 0;
                model2.grafica = 0;
                model2.h = [];
                model2.calculo = 'pinv';
                model2.pe = 1;
                model2.dc = 0;
                model2.cs = 0;
                model2.nmax = 200;

                model2.Ka=[0:Ka];
                model2.La=ones(size(model2.Ka))*La;
                model2.Kb=[1:Kb];
                model2.Lb=ones(size(model2.Kb))*Lb;
                model2.Mb=ones(size(model2.Kb))*Mb;
                model2.Kc=[1:Kc];
                model2.Lc=ones(size(model2.Kc))*Lc;
                model2.Mc=ones(size(model2.Kc))*Mc;

                [model2,toc] = model_gmp(y, x, model2);

                modelDB_row=[numIteraciones,Ka,La,Kb,Lb,Mb,Kc,Lc,Mc,model2.
                    numcoef,model2.nmse,HC_BIC_J(model2.nmse,model2.numcoef,
                    length(y)),toc];

                modelDB = cat(1,modelDB,modelDB_row);
            end
        end
```

```
                                end
                            end
                        end
                    end
                end
            end
        end
    end

end
```

Code 6.4 Function which calculates the J_{BIC} for every target model structure..

```
function [J]=HC_BIC_J(Yn,Rn,M)

    mu=(10/M)*log10(2*M);
    J=Yn+mu*Rn;

end
```

Code 6.5 Function which searches for the best J_{BIC} of every iteration..

```
function [bestModel]=HC_BestJ(modelDB,numIteraciones)


    modelDB_J=[];

    for i=1:length(modelDB(:,12))
        if modelDB(i,1)==numIteraciones
            modelDB_J=cat(1,modelDB_J,modelDB(i,:));
        end
    end

    if ~isempty(modelDB_J)
        [~,ind]=min(modelDB_J(:,12));
        bestModel=modelDB_J(ind,:);
    else
        bestModel=[];
    end


end
```

List of Figures

List of Tables

Listings

Bibliography

[1] G. E. Corazza, *Digital Satellite Communications*. Springer Science and Business Media, 2007.

[2] S. Wang, M. A. Hussein, O. Venard, and G. Baudoin, "A novel algorithm for determining the structure of digital predistortion models," vol. 67, pp. 7326–7340, August 2018.

[3] T. F. Schubert, Jr., and E. M. Kim, *Fundamentals of Electronics: Book 2 Amplifiers: Analysis and Design*. Morgan and Claypool, 2015.

[4] J. L. B. Walker, *Handbook of RF and Microwave Power Amplifiers*. Cambridge: Cambridge University Press, 2012.

[5] A. Grebennikov, *Radio Frequency and Microwave Power Amplifiers: Volume 1: Principles, Device Modeling and Matching Networks*. London: The Institution of Engineering and Technology, 2019.

[6] F. M. Ghannouchi, O. Hammi, and M. Helaoui, *Behavioral Modeling and Predistortion of Wideband Wireless Transmitters*. Chenai, India: John Wiley and Sons, 2015.

[7] S. C. Cripps, *RF Power Amplifiers for Wireless Communications*. Norwood, MA: Artech House microwave library, 2 ed., 2006.

[8] J. Wood, *Behavioral Modeling and Linearization of RF Power Amplifiers*. Norwood, MA: Artech House, 2014.

[9] "Amplifier classes." *https://www.electronics-tutorials.ws/amplifier/amplifier-classes.html*.

[10] M. K. Kazimierczuk, *RF Power Amplifiers*. John Wiley and Sons, 2008.

[11] T. L. Skvarenina, *The Power Electronics Handbook*. USA: The Power Electronics Handbook, 2002.

[12] A. Raghavan, N. Srirattana, and J. Laskar, *Modeling and Design Techniques for RF Power Amplifiers*. John Wiley and Sons, 2008.

[13] "Control desingh and system modelling." *http://homepages.ed.ac.uk/jwp/control06/controlcourse/restricted/course/third/course/modelling1.html*.

[14] B. P. Lathi, *Principles of Linear Systems and Signals*. New Delhi, India: Oxford University Press, 2009.

[15] "Vito volterra."

[16] D. R. Morgan, Z. Ma, M. G. Z. Haehyeong Kim, and J. Pastalan, "A generalized memory polynomial model for digital predistortion of rf power amplifiers," vol. 54, pp. 3852–3860, October 2006.

[17] L. Ding, G. T. Zhou, D. R. Morgan, J. S. K. Z. Ma, J. Kim, and C. R. Giardina, "A robust digital baseband predistorter constructed using memory polynomials," vol. 52, pp. 159–165, January 2004.

[18] J. A. Becerra, M. J. Madero-Ayora, R. G. Noguer, and C. Crespo-Cadenas, "On the optimum number of coefficients of sparse digital predistorters: A bayesian approach," vol. XX, March 2020.

[19] G. Schwarz, "Estimating the dimension of a model," vol. 6, March 1978.

[20] J. A. Becerra, M. J. Madero-Ayora, J. Reina-Tosina, C. Crespo-Cadenas, J. García-Frías, and Gonzalo-Arce, “A doubly orthogonal matching pursuit algorithm for sparse predistortion of power amplifiers,” vol. 28, August 2018.