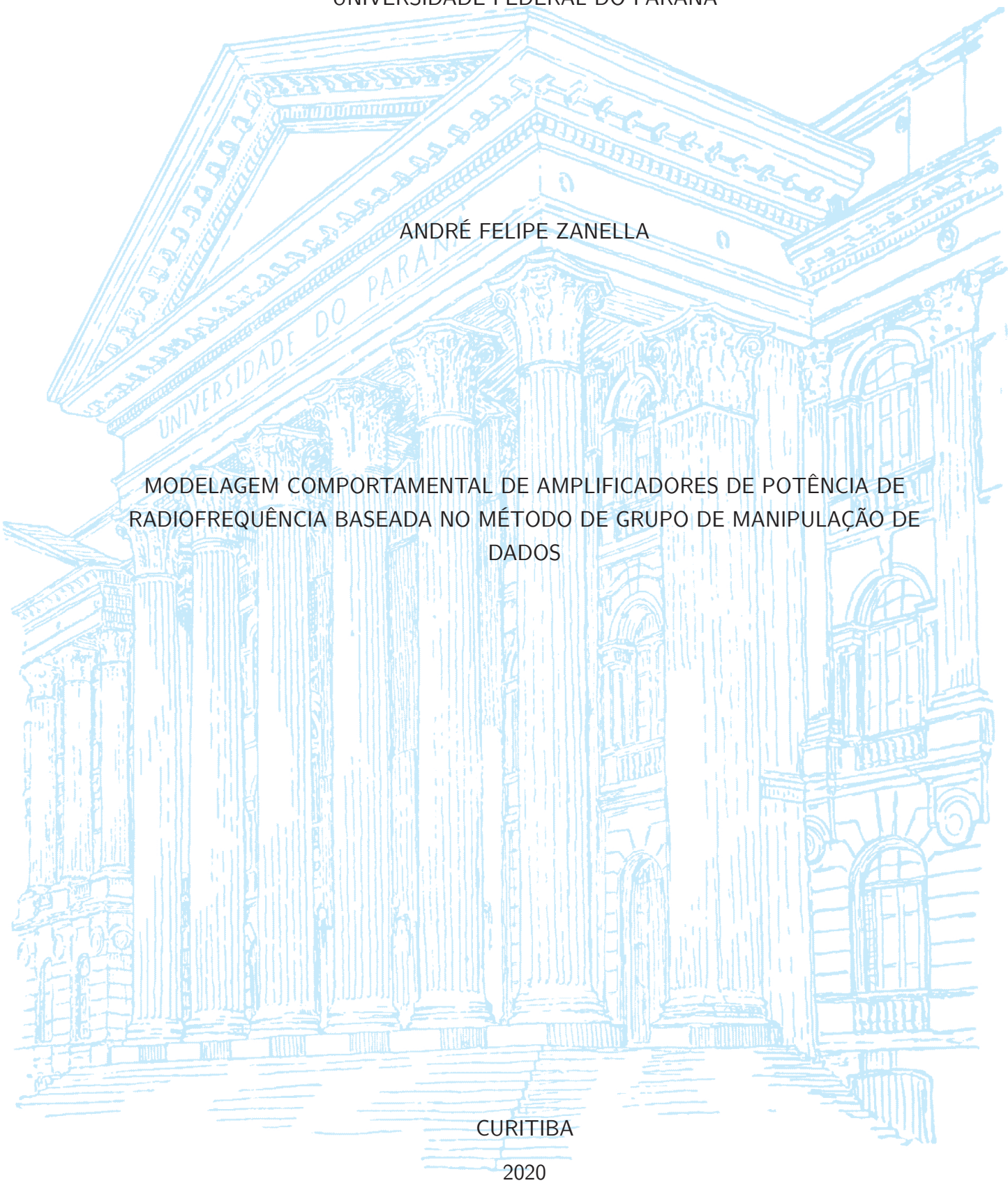
UNIVERSIDADE FEDERAL DO PARANÁ

ANDRÉ FELIPE ZANELLA



MODELAGEM COMPORTAMENTAL DE AMPLIFICADORES DE POTÊNCIA DE RADIOFREQUÊNCIA BASEADA NO MÉTODO DE GRUPO DE MANIPULAÇÃO DE DADOS

CURITIBA

2020

ANDRÉ FELIPE ZANELLA

MODELAGEM COMPORTAMENTAL DE AMPLIFICADORES DE POTÊNCIA DE RADIOFREQUÊNCIA BASEADA NO MÉTODO DE GRUPO DE MANIPULAÇÃO DE DADOS

Dissertação apresentada ao curso de Pós-graduação em Engenharia Elétrica, Setor de Tecnologia, Universidade Federal do Paraná, como requisito parcial à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Prof. Dr. Eduardo Gonçalves de Lima

CURITIBA

2020

Catalogação na Fonte: Sistema de Bibliotecas, UFPR
Biblioteca de Ciência e Tecnologia

Z28m Zanella, André Felipe
Modelagem comportamental de amplificadores de potência de radiofrequência baseada no método de grupo de manipulação de dados [recurso eletrônico] / André Felipe Zanella. – Curitiba, 2020.

Dissertação - Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Engenharia Elétrica, 2020.

Orientador: Eduardo Gonçalves de Lima.

1. Modelagem. 2. Amplificadores de potência. 3. Radiofrequência. I. Universidade Federal do Paraná. II. Lima, Eduardo Gonçalves de. III. Título.

CDD: 621.044

Bibliotecária: Vanusa Maciel CRB- 9/1928

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em ENGENHARIA ELÉTRICA da Universidade Federal do Paraná foram convocados para realizar a arguição da Dissertação de Mestrado de **ANDRÉ FELIPE ZANELLA** intitulada: **Modelagem Comportamental de Amplificadores de Potência de Radiofrequência Baseada no Método de Grupo de Manipulação de Dados**, sob orientação do Prof. Dr. EDUARDO GONÇALVES DE LIMA, que após terem inquirido o aluno e realizada a avaliação do trabalho, são de parecer pela sua APROVAÇÃO no rito de defesa.

A outorga do título de mestre está sujeita à homologação pelo colegiado, ao atendimento de todas as indicações e correções solicitadas pela banca e ao pleno atendimento das demandas regimentais do Programa de Pós-Graduação.

CURITIBA, 28 de Outubro de 2020.

Assinatura Eletrônica
28/10/2020 16:03:31.0
EDUARDO GONÇALVES DE LIMA
Presidente da Banca Examinadora

Assinatura Eletrônica
29/10/2020 08:43:39.0
ANDRÉ AUGUSTO MARIANO
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Assinatura Eletrônica
28/10/2020 16:03:40.0
GUILHERME DE SANTI PERON
Avaliador Externo (UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ)

Assinatura Eletrônica
28/10/2020 16:02:22.0
BERNARDO REGO BARROS DE ALMEIDA LEITE
Avaliador Interno (UNIVERSIDADE FEDERAL DO PARANÁ)

Este trabalho é dedicado ao Ensino Superior Público, Gratuito e Universal brasileiro.

AGRADECIMENTOS

Agradeço inicialmente aos meus pais, Clemente e Dulcemara, e também ao meu padrinho Cassio, pelo suporte que me permitiu dedicar aos estudos. Também agradeço a minha namorada Thais, pelo auxílio diário que sempre me proporcionou.

Agradeço ao Prof. Dr. Eduardo Gonçalves de Lima, por ter sido a minha porta de entrada à pesquisa e fazer parte de todas as grandes conquistas durante minha graduação e mestrado.

Também agradeço a todos os membros do GICS-UFPR por todo o suporte técnico, além de todos os colegas e professores que passaram pela minha vida durante estes dois anos que precederam a conclusão desta dissertação.

"É inimaginavelmente difícil fazer isso, ficar consciente e vivo no mundo adulto durante cada dia. O que significa que um grande clichê acaba sendo verdadeiro: a educação realmente é o trabalho de uma vida. E ele começa: agora."

(David Foster Wallace, This is water)

RESUMO

Este trabalho aborda a modelagem comportamental de um RFPA não linear, adequado para um DPD de banda base, a ser utilizado na linearização do sistema. O foco principal é a eficácia do algoritmo GMDH para esta tarefa. Trata-se de um modelo de rede polinomial auto-organizado, capaz de se construir e selecionar suas melhores características, a fim de obter um modelo com boa precisão e carga computacional reduzida, o que é interessante na implementação de um DPD em sistemas embarcados e de baixo custo. Além disso, para o RGMDH, é utilizada uma topologia comprovada para redes que processam números reais, para modelar o comportamento complexo de entrada / saída do RFPA. Já para o CGMDH, é proposto um modelo que reduz o número total de coeficientes do modelo, visto que não é necessária uma topologia para representação dos dados complexos na forma real. Ao modelar a função inversa de um RFPA não linear, o modelo baseado no RGMDH reduziu o erro quadrático médio normalizado em pelo menos 7 dB, quando comparado aos modelos de rede neural tradicionais do MLP e usando um número de coeficientes próximo a 150. Ao reduzir o total de coeficientes para próximo de 45, para comparar o RGMDH com o melhor resultado obtido para o CGMDH, o modelo com dados reais obteve um resultado 2,54 dB melhor que o modelo de dados complexos, e 10 dB melhor que o melhor resultado obtido com MLPs de coeficientes reduzidos. Por fim, nota-se que tanto o RGMDH quanto o CGMDH obtiveram uma melhor precisão do que ANNs MLP na modelagem comportamental de RFPAs, com as duas variações do GMDH obtendo desempenho próximas para o RFPA modelado, para uma complexidade computacional similar.

Palavras-chave: GMDH, DPD, modelagem, pré-distorção digital, auto-organizado

ABSTRACT

This work addresses the behavioral modeling of a RFPA suitable for a DPD utilized in linearization of the system. The main focus given is in the effectiveness of the GMDH algorithm for this task. This is a self-organized polynomial network model, capable of building itself and selecting its best features, in order to achieve a model with good accuracy and reduced computational burden, which is interesting when implementing the DPD in embedded and low cost systems. In addition to that, a proven topology for networks that process real numbers is utilized, in order to model the complex input/output behavior of the RFPA. When modeling the inverse function of a nonlinear RFPA, the RGMDH based model reduced the normalized mean square error by at least 7 dB, when compared to traditional MLP neural network models and using a number of coefficients close to 150. When reducing the number of coefficients close to 45, in order to compare the RGMDH model with the best result obtained with the CGMDH model, the model processing real data had an NMSE result 2,54 dB lower than the model processing complex data, and 10 dB lower than the best result obtained with the MLP models with reduced coefficients. It can be concluded that both the RGMDH and CGMDH obtained a better performance than the MLP models in the behavioral modeling of a RFPA, with both variations of GMDH having a similar performance for the utilized RFPA dataset, with a similar computational complexity.

Keywords: GMDH, DPD, modeling, digital predistortion, self-organized.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E DE SIGLAS

ANN	Rede Neural Artificial
ASIC	Circuito Integrado de Aplicação Específica
CGMDH	Método do Grupo de Manipulação de Dados Complexos
CPU	Unidade Central de Processamento
CVNN	Rede Neural de Valores Complexos
DC	Corrente Contínua
DPD	Pré-distorsor Digital
DSP	Processador de Sinais Digital
FPGA	*Field Programmable Gate Arrays*
GMDH	*Group Method of Data Handling)*
GPU	Unidade de Processamento Gráfico
HDL	Linguagem de Descrição de *Hardware*
HW	*Hardware*
LUT	*Look-up Tables*
MLP	*Perceptron* Multicamadas
MMQ	Método dos Mínimos Quadrados
MSE	Erro Quadrático Médio
NMSE	Erro Quadrático Médio Normalizado
OFDMA	*Orthogonal frequency-division multiple access*
PAPR	Razão entre as Potências de Pico e Média
RBF	Função de Base Radial
RF	Radiofrequência
RFPA	Amplificador de Potência de Radiofrequência
RGMDH	Método do Grupo de Manipulação de Dados Reais

VHDL Linguagem de Descrição de *Hardware* VHSIC

WCDMA *Wide-Band Code-Division Multiple Access*

LISTA DE SÍMBOLOS

dB	Decibel
θ_n	Fase do Sinal de Entrada do RFPA
α_n	Fase do Sinal de Saída do RFPA
∞	Infinito
a_n	Módulo do Sinal de Entrada do RFPA
b_n	Módulo do Sinal de Saída do RFPA
M	Ordem de Memória do Modelo

SUMÁRIO

1 INTRODUÇÃO

A taxa elevada de transmissão demandada em padrões de comunicação móvel (3G, 4G e 5G) traz um desafio interessante no projeto de sistemas de transmissão sem fio. Devido ao espectro de frequências cada vez mais limitado em sistemas com um número massivo de usuários, as agências regulamentadoras impõem restrições rígidas quanto à largura de banda de transmissor, evitando que canais adjacentes sejam invadidos por este durante a operação. Para um projetista de circuitos, isto torna-se uma exigência de linearidade na transmissão do sinal pelo sistema. Todavia, um Amplificador de Potência de Radiofrequência (RFPA) pode apenas prover uma linearidade de sinal quando operado em uma região que requer menor potência de entrada, resultando num sistema com baixa eficiência energética (CRIPPS, 2000). Visto que o aumento do consumo energético resulta em aumento do custo de instalação e operação de uma estação rádio-base, ou do aumento de consumo de bateria em sistemas isolados, a eficiência do sistema torna-se crítica. A fim de evitar que o RFPA torne-se gargalo no consumo do sistema, é costumeiro operá-lo em sua região não linear, pois esta requer níveis maiores de potência de entrada, quando comparada à região linear. Para isto, torna-se elementar a utilização de métodos de linearização do sistema, para garantir respeito ao espectro em que o sistema operará.

Uma abordagem interessante para atender simultaneamente a demanda por linearidade e eficiência energética em sistemas de comunicação sem fio é a inclusão de um bloco de linearização na cadeia de transmissão (LIMA, 2009). O uso de um Pré-distorsor Digital (DPD) para executar o papel linearizador é uma boa opção focada em custo baixo de desenvolvimento e implementação. O desafio torna-se obter um modelo que atinja os requisitos de desempenho mantendo uma complexidade baixa, para que seu consumo energético final não seja superior aos ganhos de eficiência obtidos pelo RFPA operando na região não linear, ou que o *hardware* utilizado tenha custos muito elevados, dificultando sua implementação em projetos em massa. Além disto, é interessante que este modelo seja capaz de se atualizar durante a operação, a fim de compensar qualquer novo efeito não visto anteriormente durante a fase de concepção do DPD, evitando problemas durante operação.

2 OBJETIVOS

2.1 OBJETIVO GERAL

O objetivo deste trabalho é apresentar uma metodologia nova para o desenvolvimento de DPDs focados na linearização de RFPAs. Esta será uma arquitetura utilizando o Método do Grupo de Manipulação de Dados (GMDH), já explorada anteriormente na literatura, mas pouco visto na área de telecomunicações. Este método possui um processo de treinamento que resultará, para cada conjunto de dados únicos, não apenas valores de seus coeficientes, mas também uma estrutura de rede neural única e já otimizada, compondo um modelo de arquitetura auto organizada e capaz de se otimizar já durante o treinamento, e finalizando com um custo computacional menor quando comparado com redes neurais tradicionais. Isto auxiliará a reduzir o viés do projetista, além de se encaixar melhor aos diferentes tipos de dados e RFPAs que poderão exigir linearização, visto sua capacidade de mudar seu formato para cada base de dados nova.

2.2 OBJETIVOS ESPECÍFICOS

Desta forma os objetivos específicos deste trabalho serão:

- Apresentar um modelo de Rede com GMDH capaz de realizar a modelagem inversa de um RFPA e resultar em um DPD apto para a aplicação desejada;
- Apresentar uma variação do GMDH capaz de trabalhar com dados no domínio complexo, e validá-los contra o GMDH com dados no domínio real;
- Montar modelos com técnicas já consolidadas de redes neurais para compará-los aos modelos de GMDH propostos a fim de validar seu desempenho na modelagem inversa do RFPA.

2.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho será organizado da seguinte forma. O Capítulo 3 apresenta uma revisão dos princípios básicos de elementos de telecomunicações em radiofrequência; o Capítulo 4 apresentará conceitos importantes da Modelagem Comportamental de RFPAs, assim como sua implementação em *hardware*. O Capítulo 5 apresenta técnicas de modelagem baseadas em funções polinomiais; o Capítulo 6 apresenta técnicas de modelagem baseadas em redes neurais artificiais; o Capítulo 7 mostrará os conceitos do GMDH, assim como a forma de usá-lo para a modelagem de RFPAs; os Capítulos 8, 9 e 10 apresentarão os resultados para a modelagem

de um RFPA utilizando o GMDH com valores no domínio real e imaginário, assim como a comparação dos resultados obtidos em relação a métodos já consolidados.

3 ELEMENTOS DE RADIOFREQUÊNCIA

3.1 AMPLIFICADORES DE POTÊNCIA DE RF

Independente do processo de comunicação a ser analisado, um sistema de comunicações sempre possuirá três elementos básicos: transmissor, meio e receptor (HAYKIN, 2006). O transmissor e receptor estão localizados em pontos distintos do espaço, enquanto o meio é o canal físico que interliga ambos. Em um RFPA, o transmissor tem o papel de converter o sinal da mensagem produzido pela fonte de informação em uma forma adequada à transmissão em um canal (HAYKIN, 2006). Devido à propriedade natural de qualquer meio de atenuar a energia de um sinal que se propaga por este, além da presença de distorções, imperfeições e ruídos presentes no meio que serão adicionados ao sinal, o transmissor deve entregar o sinal transmitido com um nível mínimo de energia. Assim, o receptor poderá receber o sinal de informação, acrescido de ruídos do meio, e reconstruir de uma forma reconhecível o sinal original para o usuário. O elemento de interesse e primeiro a ser comentado neste trabalho será o PA de banda simples.

Em um sistema de transmissão, o RFPA é o elemento presente no transmissor responsável por entregar um ganho de energia ao sinal já modulado, a fim de que o sinal transmitido pela antena chegue ao circuito receptor com um nível mínimo de energia para ser demodulado, após sofrer as atenuações presentes no meio de propagação do sinal. Um modelo simples de RFPA é visto na Figura 1. Este recebe energia de uma fonte de corrente contínua (DC) externa, transferindo-a para o sinal de RF.

O balanceamento de potências se dará na forma da Equação (3.1):

$$P_{OUT} = P_{IN} + P_{DC} - P_{DIS}, \tag{3.1}$$

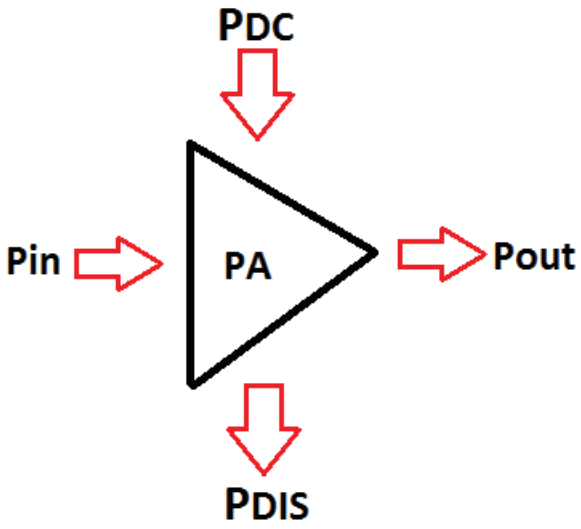
onde P_{OUT} é a potência contida no sinal de saída do RFPA, P_{DC} é a potência fornecida pela fonte DC, P_{IN} é a potência contida no sinal de entrada do RFPA, vinda do bloco de modulação, e por fim P_{DIS} representa as perdas por dissipação interna do circuito do RFPA.

O ganho de potência de um RFPA é o quociente entre a potência de saída do PA e a potência de entrada, conforme a Equação (3.2):

$$G = \frac{P_{OUT}}{P_{IN}}. \tag{3.2}$$

O ganho é um dos parâmetros mais desejados num amplificador de potência. Ele mostrará quantas vezes a potência de saída é maior em relação à entrada, sendo essencial no

FIGURA 1 – Representação básica de um RFPA.



FONTE: O autor (2020)

cálculo de propagação de um *link* de comunicação, para que potência suficiente chegue ao receptor.

A eficiência do circuito é outro conceito fundamental quando estuda-se o RFPA. Representada por

$$\eta = \frac{P_{OUT}}{P_{DC}}, \tag{3.3}$$

em que a Equação (3.3) mostra a eficiência da conversão de energia entregue pela fonte DC em potência disponível na saída do RFPA. Quanto menores forem as perdas de dissipação, maior será a eficiência.

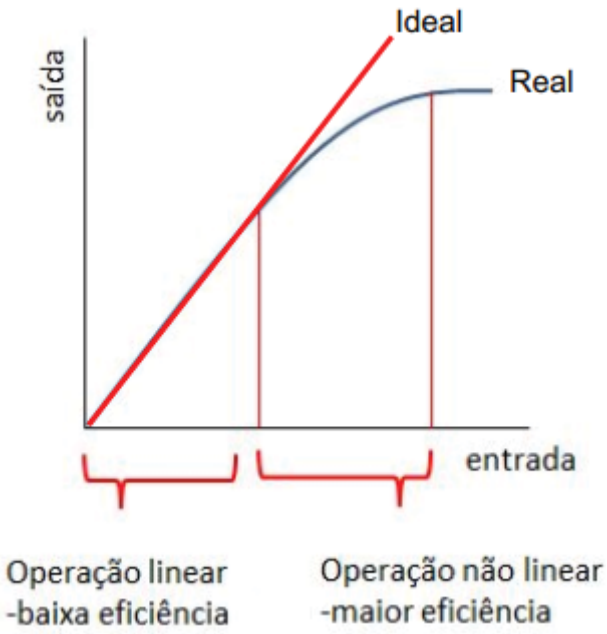
3.2 LINEARIDADE *VERSUS* EFICIÊNCIA

Um RFPA apresenta um comportamento inverso de sua eficiência contra sua linearidade (CRIPPS, 2000). Ou seja, para que ele tenha uma maior eficiência energética, haverá um custo de perda de linearidade, com o oposto também sendo verdade. Com isto, ele possuirá 2 zonas possíveis de operação, conforme visto na Figura 2.

A primeira região é chamada de Região de Operação Linear. Nesta, a curva característica de ganho do PA segue uma relação linear entre potência de entrada e saída. Opera-se nesta região quando os níveis de potência de entrada são baixos. Ao operar o RFPA longe de sua região de saturação dos transistores, obtém-se uma menor eficiência energética do sistema. Em sistemas de comunicação móvel, como celulares e dispositivos de monitoramento

remoto, um aumento no consumo de energia do circuito de radiofrequência (RF) implica em uma diminuição no tempo de uso da bateria (indesejável pelo usuário) ou na necessidade de um aumento na capacidade dela (resultando em custos elevados). Protocolos novos de comunicação com foco em eficiência espectral, como *Orthogonal frequency-division multiple access* (OFDMA) e *Wide-Band Code-Division Multiple Access* (WCDMA) (utilizados na tecnologia 3G), buscam conciliar o aumento do número de usuários do sistema de comunicações com taxas de transmissão e limitações de banda. Estes padrões possuem índices elevados de razão entre as potências de pico e média (PAPR), tipicamente na ordem de 10 dB, mesmo após implementação de técnicas para sua redução (ROBLIN et al., 2013). Isto inviabiliza operar o RFPA em sua região linear.

FIGURA 2 – Regiões de operação do RFPA.

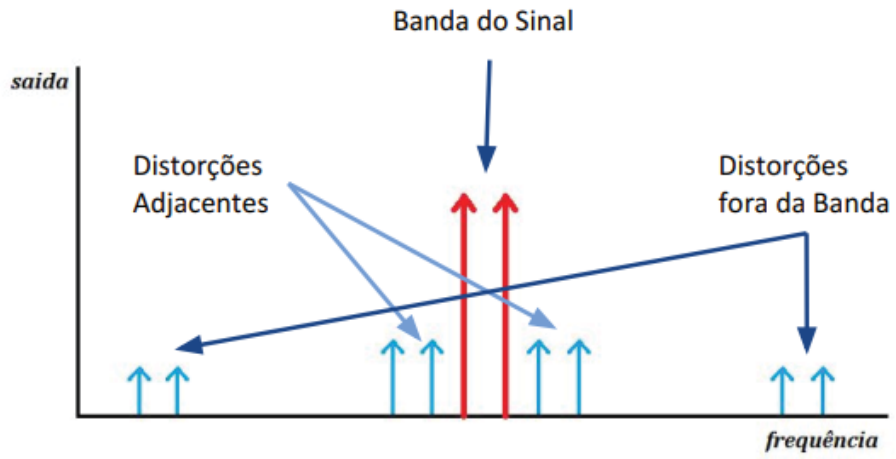


FONTE: O autor (2020)

A segunda região é a Região de Saturação. Nesta, o ganho do PA exibe uma característica não linear na sua relação de entrada/saída. Ao operar com níveis superiores de potência de entrada haverá uma forte compressão no ganho do circuito. Isto resultará em distorções que levam a um aumento da largura de banda do sinal, invadindo canais adjacentes do espectro, conforme observado na Figura 3. Porém, a troca é um aumento na eficiência do circuito, algo de extremo interesse em sistemas móveis.

Ao operar na região de saturação, são introduzidas então distorções no domínio da frequência. As distorções fora da banda do sinal, vistas na Figura 3, são facilmente removidas ao aplicar um filtro passa-banda centrado na frequência da portadora do sinal a ser transmitido. Porém, as distorções adjacentes à banda do sinal já não podem ser filtradas por filtros realísticos. Estas podem causar a invasão de canal entre usuários de canais adjacentes,

FIGURA 3 – Saída de um PA operando em saturação, em função da frequência.



FONTE: O autor (2020)

fenômeno comumente chamado de intermodulação. Para operar na Região de Saturação, é necessário então introduzir elementos que irão linearizar o RFPA, conseguindo assim reduzir as distorções de bandas adjacentes que não são filtradas pelo passa-banda.

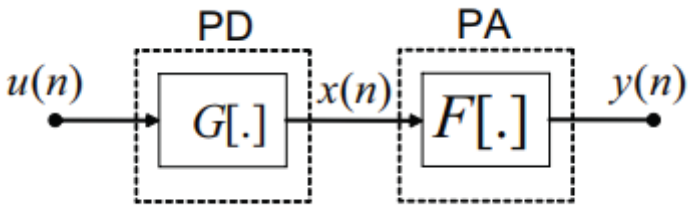
3.3 PRÉ-DISTORSOR EM BANDA BASE E MODELAGEM COMPORTAMENTAL

Conforme falado na seção anterior, a fim de atenuar as distorções adjacentes e modulações cruzadas resultantes da operação na região de saturação do RFPA, é necessário introduzir um elemento linearizador ao bloco do transmissor. Uma abordagem comum e que apresenta um interessante custo-benefício é a utilização de um Pré-Distorsor Digital em cascata ao RFPA. A pré-distorção pode ser realizada de duas maneiras: em radiofrequência (RF) ou em banda base. Quando realizada em rádio frequência, a pré-distorção torna-se uma difícil tarefa, por exigir circuitos analógicos e, consequentemente, modelos matemáticos de elevada acurácia e complexidade em sua modelagem. Uma alternativa então é realizá-la em banda base, ou seja, com sinais contidos em frequências muito próximas de 0 Hz. Com isto, a pré-distorção pode ser feita com circuitos digitais, como *Field Programmable Gate Arrays* (FPGA), reduzindo os custos com projeto e utilizando componentes de produção massificada e custo inferior quando comparado a um circuito dedicado. Conectar um DPD em cascata significa deixar seu bloco antes do bloco do RFPA, a fim que o DPD introduza uma distorção inversa a do RFPA que cancele seus efeitos não lineares ao operar na região de saturação, trazendo um ganho de eficiência ao sistema.

Quando trata-se um DPD como uma topologia inversa do modelo comportamental do RFPA, é necessário então aplicar conceitos de modelagem comportamental (LIMA, 2009). Ao trabalhar com uma topologia de cascata, a saída do DPD será conectada em série com a entrada do PA, como pode ser visto na Figura 4. Nela, $u(n)$ é o sinal modulado com a

informação desejada. Este passará por uma função arbitrária $G[.]$ que descreve o modelo do DPD, em que $x(n) = G[u]$. O PA será representado pela função arbitrária $F[.]$, sendo sua entrada a saída do DPD, e sua saída sendo $y(n) = F[x]$.

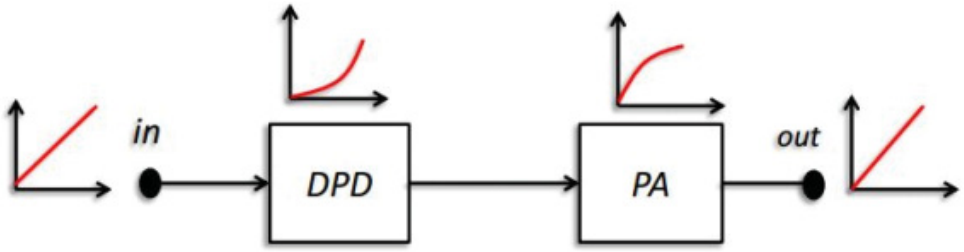
FIGURA 4 – Conexão em cascata de um DPD com um RFPA.



FONTE: (LIMA, 2009)

A fim de garantir que a relação entre u(n) e y(n) da Figura 4 seja linear, é necessário que a função G[.] seja a inversa da função F[.]. Ou seja, a curva do DPD deverá possuir as características inversas da curva característica do PA que deseja-se linearizar. Isto é melhor observado na Figura 5. Torna-se imprescindível uma modelagem precisa da curva característica do RFPA, a fim de obter uma maior fidelidade na modelagem comportamental do DPD e um melhor resultado na linearização.

FIGURA 5 – Princípio de funcionamento da pré-distorção em cascata.



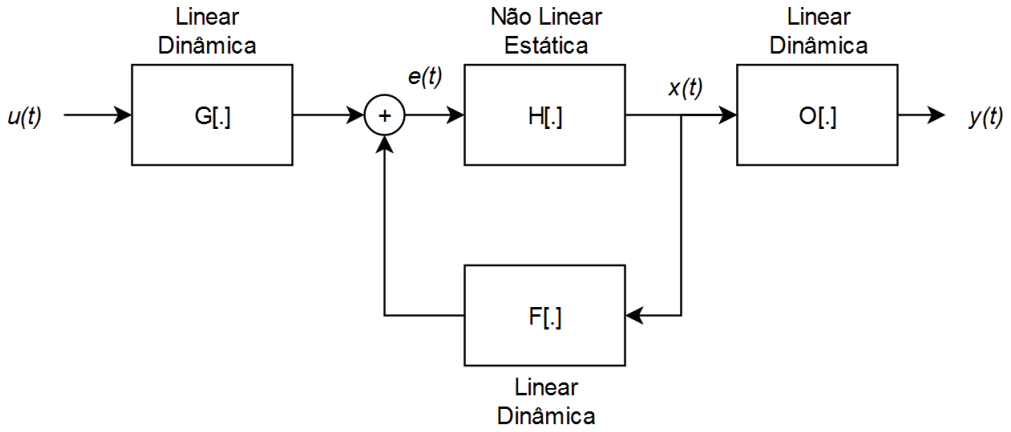
FONTE: (PAIXÃO RIBA, 2007)

Numa abordagem em que o modelo matemático do RFPA apresenta memória, a modelagem do DPD também deverá levar em conta efeitos de memória, a fim de compensá-los (LIMA, 2009).

4 MODELAGEM COMPORTAMENTAL DE UM RFPA

A relação entre o sinal de entrada e saída de um RFPA foi descrita em (PEDRO; CARVALHO; LAVRADOR, 2003), na forma de uma topologia com realimentação que pode ser vista na Figura 6. Aqui, $u(t)$ representa a entrada do RFPA e $y(t)$ representa a saída. O bloco representado pela função $G[.]$ será a rede seletiva de casamento de frequência da entrada, que tem como função garantir que apenas sinais na banda passante correspondente gerem sinais na saída do sistema, enquanto o bloco $O[.]$ será a rede seletiva de casamento de frequência da saída, que irá restringir que todos os sinais vistos nesta estejam dentro da banda passante. O *loop* fechado entre saída e entrada, representado pelos blocos com função $H[.]$ e $F[.]$ é capaz de modelar os efeitos não lineares e dinâmicos, com ordem de truncamento ilimitada.

FIGURA 6 – Diagrama de um RFPA com *Feedback*



FONTE: O autor (2020), adaptado de (LIMA, 2009)

As funções que serão utilizadas nos blocos $H[.]$, $G[.]$, $O[.]$ e $F[.]$ são a essência do problema de modelagem, e serão melhor abordadas em capítulos seguintes. O importante agora é ressaltar que o modelo da Figura 6 contém uma grande quantidade de informações, tendendo a possuir também um elevado número de coeficientes necessários para descrever de maneira competente o modelo.

De fato, no grande espectro de métodos disponíveis, como funções polinomiais e redes neurais artificiais, a atividade de modelar os valores complexos que descrevem a relação não linear entre a entrada $u(t)$ e a saída $y(t)$ é uma tarefa árdua. A fim de obter uma acurácia mínima necessária, estes modelos tendem a crescer e necessitar de um elevado número de coeficientes a serem determinados durante o processo de treinamento e posteriormente armazenados durante a implementação em *hardware*. Além disto, o número de operações necessárias a serem feitas também pode se tornar um gargalo da função escolhida para representar o bloco não linear

do sistema, visto que poderá haver limitações também nas unidades de processamento que realizarão estas.

Muitas das informações que fazem o modelo tornar-se computacionalmente complexo também são consideradas essenciais para a eficácia da modelagem, como a ordem de truncamento do modelo polinomial, ou mesmo a ordem de memória necessária para descrever a influência das entradas passadas no valor instantâneo da saída (LIMA, 2009).

Um algoritmo preciso de modelagem, especialmente aqueles focados em modelos para telecomunicações, deverá ser capaz de estimar valores complexos (ou seja, composto de parte real e imaginária) que representam o sinal de envoltória, e ser capaz de estimar os comportamentos não lineares vistos em amplitude e fase deste sinal, quando for prever os valores de saída do RFPA. Além disto, é interessante neste caso que estes modelos sejam aplicados juntos de técnicas de redução de complexidade computacional, a fim de reduzir sua demanda por recursos de processamento, reduzir o tamanho do modelo para consumir menos memória de armazenamento, resultando em um modelo leve que possa ser embarcado em *hardwares* simples. Este ponto afeta totalmente os custos finais de um projeto para produção em massa.

4.1 IMPLEMENTAÇÃO DA MODELAGEM COMPORTAMENTAL EM *HARDWARE*

Ao longo dos anos, variadas técnicas de implementação de modelos matemáticos na forma de circuitos eletrônicos foram apresentadas. A essência é utilizar um circuito que seja capaz de representar no mundo real os valores obtidos nas simulações destes modelos matemáticas, com uma precisão desejada e também levando conta o consumo energético para a realização de tais operações.

Cada um destes modelos de circuito variam drasticamente em complexidade de implementação, complexidade de fabricação, nível de generalização, eficiência energética e custo total do projeto. As próximas seções entrarão em detalhes de algumas das mais populares técnicas de implementação em *hardware*, assim como apontando as problemáticas para a implementação de um DPD.

4.2 CIRCUITOS INTEGRADOS DE APLICAÇÃO ESPECÍFICA

Os Circuitos Integrados de Aplicação Específica (ASIC - do inglês *Application-specific Integrated Circuit*) são circuitos integrados customizados para um caso de uso específico, sendo considerados um *hardware* ultra especializado.

Estes funcionam no princípio de implementar funções lógicas fixas e estáticas, dedicadas a uma tarefa específica, como por exemplo, a conversão de um sinal, modulação ou até mesmo a pré-distorção. Ao implementar funções físicas, o projetista tem a liberdade de ir ao mais baixo nível de circuito, otimizando os transistores utilizados, a fim de reduzir tanto o tamanho

físico do circuito, seu total de elementos e, mais importante, otimizar sua eficiência energética ao máximo.

Sua premissa é simples: transformar um modelo matemático em uma representação de circuito específico (ou seja, que realiza apenas esta tarefa). Porém, esta poderá tornar-se uma tarefa árdua, especialmente conforme o modelo torna-se mais e mais complexo de implementar, especialmente falando em questões de latência do circuito (o tempo necessário para obter a resposta na saída após receber a entrada).

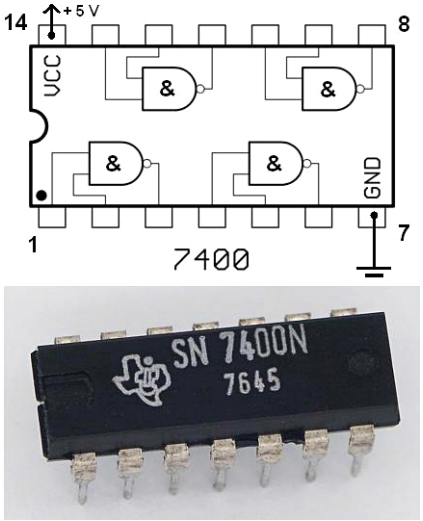
Além disto, por tornar-se um projeto de uso específico, que poderá ter especificações a nível de transistores, seus custos de design e fabricação tornar-se-ão muito elevados, desencorajando muitas aplicações que buscam baixo custo.

4.3 CIRCUITOS INTEGRADOS GENERALISTAS

Localizados no polo oposto do espectro, em relação aos ASICs, estão os ICs generalistas. Este é um termo amplo, mas que no geral abrange qualquer tipo de circuito que pode ter suas funções lógicas modificadas para atender diferentes tipos de tarefa. Ou seja, ao invés do ASIC construído e otimizado para uma tarefa específica, a função destes ICs é atender ao maior número de aplicações diferentes.

O termo torna-se abrangente em complexidade. Pode-se englobar nele famílias de ICs lógicos baseados em tecnologia CMOS, como os da série 4000, ou em transistores de junção bipolar como os ICs da família 7400, visto na Figura 7. Estes são circuitos simples, que implementam uma série de funções com um pequeno número de portas lógicas e transistores, mas já abrangem um bom número de tarefas.

FIGURA 7 – IC da Família 7400, junto do seu esquema de portas lógicas baseadas em transistores TBJ.



FONTE: Texas Instruments (2020)

Mas o termo também abrange *hardwares* mais complexos, como as unidades centrais de processamento (CPU) ou unidades de processamento gráfico (GPU). Estas podem executar uma vasta quantidade de tarefas, baseadas em código e algoritmo escritos por seus projetistas.

Porém, esta característica generalista traz malefícios, com atenção especial para este trabalho: seu consumo energético. O fato destes serem desenhados para abranger o máximo de tarefas faz com que sua eficiência energética não seja otimizada aos níveis que chegam os ASICs, e para problemas de *hardware* embarcado para a pré distorção de um RFPA, acabam podendo exigir mais energia do que é salvo operando o RFPA em sua região não linear.

4.4 FPGAS

Uma FGPA é um circuito integrado capaz de ser customizado pelo usuário após sua fabricação. Diferentemente de uma CPU, que é um processador capaz de organizar diferentes funções de acordo com o seu código, uma FPGA é capaz de reconfigurar a disposição dos seus circuitos de acordo com o desejo do projetista. Torna-se então um meio termo interessante entre ASICs e ICs generalistas.

FPGAs contêm uma matriz de blocos lógicos programáveis, organizados através de uma hierarquia que permite reconfigurar suas conexões a fim de gerar novas configurações que implementarão novas funções, como portas lógicas (NOR, XOR, etc...), flip-flops ou até mesmo elementos de memória (RAM, ROM). A Figura 8 mostra o XC2064, a primeira FPGA comercialmente viável, fabricada pela Xilinx em 1985. Esta possuia 64 blocos lógicos programáveis e duas *look-up tables* (LUT), responsáveis por indexar valores a fim de reduzir operações em tempo real.

FIGURA 8 – FPGA Xilinx XC2064.

FONTE: Xilinx (2020)

Esta recombinação de *hardware* é feita através da Linguagem de Descrição de *Hardware* (HDL), que possui duas populares versões: A Linguagem de Descrição de *Hardware* VHSIC (VHDL) (hoje comumente associada com soluções da fabricante Xilinx) e Verilog (comumente associado com soluções da Intel/Altera). Como seu nome diz, esta é uma linguagem de computador utilizada para descrever a estrutura e comportamento de um circuito eletrônico, especialmente circuitos lógicos digitais. Após escrito, este será compilado e executado em uma FPGA, a fim de fazê-la assumir a configuração desejada pelo usuário.

Além disto, um dos seus grandes diferenciais em relação às ASICs é a possibilidade da FPGA poder ser reprogramada em campo. Isto é algo pouco possível para circuitos especializados, mas possível para uma FPGA. Isto é interessante para este trabalho, considerando por exemplo um DPD que atualize seus coeficientes com o tempo para adequar qualquer variação na relação não linear do RFPA devido parâmetros externos.

Com estes pontos, pode-se entender que o uso de FGPAs torna-se uma das melhores soluções em questão de velocidade de implementação e otimização de custos, para o problema de um DPD Digital baseado em modelos matemáticos.

4.5 LIMITAÇÕES IMPOSTAS POR FGPAS E PONTOS DE ATENÇÃO DA IMPLEMENTAÇÃO DE MODELOS MATEMÁRICOS EM *HARDWARE*

Como falado nas seções anteriores, FPGAs tornam-se ferramentas interessantes para a implementação de um DPD Digital. Porém, limitações físicas impostas por diferentes fabricantes e modelos precisam de uma atenção extra já na fase de concepção do projeto, a fim de garantir que a complexidade do modelo seja atendida pelo IC selecionado.

Uma FPGA é composta por um conjunto de portas lógicas, portas I/O (entrada/saída), canais de roteamento e blocos de funções específicas, como Processadores de Sinal Digital (DSP) , Multiplicadores, LUTs e memórias voláteis e não voláteis. Pelo fato destes itens serem os responsáveis por representar no circuito as operações aritméticas necessárias ao algoritmo do DPD, é importante notar que poderão ocorrer limitações no número de operações a serem feitas, sejam elas em paralelo, ou caso sejam serializadas, qual o impacto surge em questão de latência do sinal.

Um caso simples a ser notado é a quantidade de multiplicações a serem realizadas de maneira paralela, em um ciclo de *clock*. Caso o algoritmo exija mais do que a FPGA provê em blocos lógicos, parte das operações precisará ser realizada no ciclo seguinte, aumentando a latência do sinal, um comportamento que pode ser indesejado em aplicações em tempo real. No geral, o número de multiplicações em paralelo disponíveis em uma FPGA costuma ser um valor limitado, e uma atenção especial é necessária quando um modelo for descrito em *hardware*.

Além disto, o processo de modelagem de um RFPA costuma abranger sempre a determinação de coeficientes de um modelo, que representarão seu comportamento. Quando implementado em *Hardware* (HW), todos estes coeficientes precisarão ser armazenados. Não apenas isto, mas conforme a necessidade de acurácia do modelo, o número de bits que cada coeficiente ocupará também irá variar de acordo com o tipo de dado que este será (*float*, *double*, ...).

A fim de reduzir o número de multiplicações, é possível implementar algoritmos na FPGA que substituem uma multiplicação por uma série de somas e outras operações. Estes utilizarão as DSPs da FGPA, que costumam ser mais abundantes que multiplicadores. Porém,

será um processo adicional a ser implementado, que acabará resultando em mais consumo energético. Outra forma de reduzir multiplicações é implementar LUTs que representem os valores esperados de tais multiplicações. Esta maneira pode ser efetiva para reduzir a quantidade de multiplicações necessárias em paralelo, mas para modelos com memória limitada, poderá tornar-se um dificultador, visto que ainda será necessário armazenar os coeficientes.

Em geral, o ideal quando buscamos um modelo matemático a ser implementado em uma FPGA é uma relação de desempenho *versus* complexidade. O primeiro ponto sempre a buscar é a acurácia do modelo, mas o seu número de multiplicações e seu número total de coeficientes a serem estimados tornam-se fatores de diferenciação, visto que estes podem impactar na aplicação do modelo matemático e impossibilitar a usabilidade deste.

5 MODELAGEM COMPORTAMENTAL DE RFPAS COM FUNÇÕES POLINOMIAIS

Uma das principais problemáticas no trabalho de linearização de um RFPA é a técnica utilizada para modelar seu comportamento não linear, a fim de poder gerar o modelo matemático de um DPD. Mais especificamente, definir um modelo que seja capaz de representar a largura na frequência dos sinais de um PA operando com distorções devido à região não linear (a fim de aumentar sua eficiência energética). Uma das principais abordagens escolhida é o uso de funções polinomiais como representante do modelo.

Este capítulo irá abordar alguns dos principais polinômios utilizados para a modelagem comportamental de RFPAs, assim como os benefícios e problemáticas de cada um.

5.1 ALGORITMOS ORIENTADOS EM BLOCOS

Esta seção busca mostrar alguns modelos orientados em bloco, utilizados para a identificação de sistemas não lineares. Este método parte da ideia de que para poder modelar comportamentos não lineares de um sistema, modelos matemáticos deverão ser feitos para cada um destes a fim de modelá-los em partes, em uma espécie de decomposição do comportamento, com a combinação destes modelos parciais (ou blocos) sendo então capaz de modelar o sistema completo.

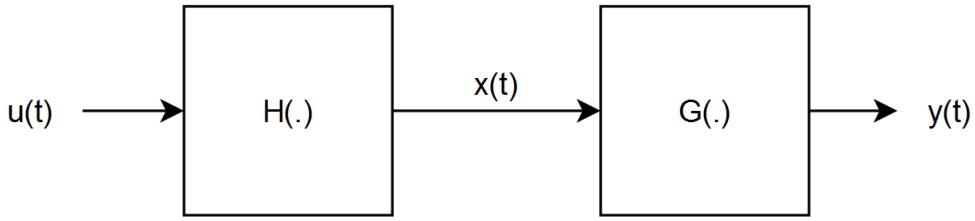
Dois dos modelos mais costumeiros na literatura de variadas áreas são os modelos de Hammerstein e Wiener, que serão descritos nas subseções a seguir.

5.1.1 Modelo de Hammerstein

A utilização de um Modelo de Hammerstein para a modelagem de um RFPA foi proposta em (ISAKSSON; WISELL; RÖNNOW, 2006) e pode ser visualizada na Figura 9. Nele, um sistema não linear com entrada $u(t)$ e saída $y(t)$ será identificado através de um algoritmo orientado a blocos. Nele, dois blocos serão os responsáveis por representar o sistema: o primeiro bloco será uma função $H(.)$, que é uma função não linear e irá transformar $u(t)$ em $x(t)$. Em seguida, os valores x(t) serão modelados através de uma função de transferência $G(.)$, que será responsável por gerar a saída $y(t)$.

O que caracteriza este modelo é a posição em que o bloco da não linearidade estará. Para Hammerstein, a não linearidade do sistema é tratada como efeitos de um ganho em relação à entrada. Ou seja, ele operará de modo instantâneo logo na entrada do sistema. Considerando que a função não linear $H(.)$ pode ser representada por polinômios, ela poderá

FIGURA 9 – Diagrama de blocos para um Modelo de Hammerstein.



FONTE: O autor (2020)

ser descrita da forma:

$$x(t) = \alpha_1 u(t) + \alpha_2 u^2(t) + ... + \alpha_m u^m(t) \tag{5.1}$$

em que m representa a ordem do polinômio. Além disso, a função de transferência $G(.)$ que mapeará a saída $y(t)$ com relação ao valor intermediário $x(t)$ é dada por:

$$\begin{aligned} &y(t) + a_1 y(t-1) + ... + a_n y(t-n) = \\ &b_1 x(t-1) + b_2 x(t-2) + ... + b_n x(t-n) \end{aligned} \tag{5.2}$$

em que $y(t)$ é o valor que deseja ser previsto e n representa quantos instantes passados serão considerados no modelo, ou seja, a memória. Igualando a Equação (5.1) com a Equação (5.2), pode-se obter a função de transferência total do diagrama de blocos de um Modelo de Hammerstein, que será:

$$\begin{aligned} &y(t) + a_1 y(t-1) + ... + a_n y(t-n) = \\ &\alpha_1 b_1 u(t) + ... + \alpha_1 b_m u^m(t) + ...+ \\ &\alpha_n b_1 u(t-n) + \alpha_n b_m u^m(t-n) \end{aligned} \tag{5.3}$$

Desta forma, pode-se reescrever a Equação (5.3) em forma de função de transferência da saída $y(t)$ com a entrada $u(t)$:

$$y(t) = \frac{B(q^{-1})}{A(q^{-1})} \sum_{i=1}^{m} \alpha_i u^i(t) \tag{5.4}$$

em que:

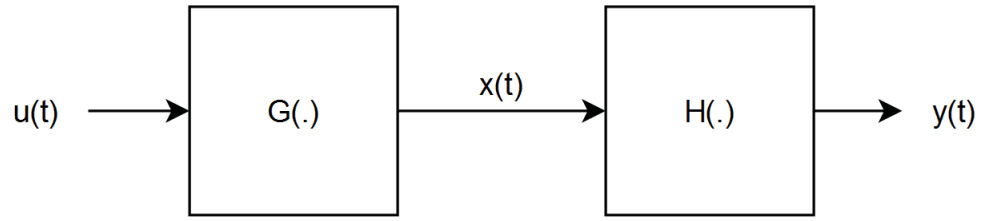
$$\begin{aligned} &A(q^{-1}) = 1 + a_1 q^{-1} + ... + a_n q^{-k} \\ &e \\ &B(q^{-1}) = b_1 q^{-1} + ... + b_n q^{-j} \end{aligned} \tag{5.5}$$

sendo k e j as ordens de cada polinômio.

5.1.2 Modelo de Wiener

O Modelo de Wiener é muito similar ao Modelo de Hammerstein. Propostas do uso deste modelo para a modelagem comportamental de RFPAs são vistas em (MUHA et al., 1999), (CLARK et al., 1998) e (CHRISIKOS et al., 1998). Como pode ser visto na Figura 10, sua principal diferença é a troca na ordem entre os blocos da não linearidade e a função de transferência.

FIGURA 10 – Diagrama de blocos para um Modelo de Wiener.



FONTE: O autor (2020)

Desta forma, o Modelo de Wiener irá descrever a não linearidade do sistema como efeitos de um ganho sobre a saída do modelo. Desta forma, a entrada inicialmente será mapeada pela função de transferência $G(.)$, resultando em:

$$x(t) = a_1u(t-1) + a_2u(t-2) + ... + a_nu(t-n) \tag{5.6}$$

Então, a não linearidade do sistema será descrita a partir de:

$$y(t) = \alpha_1x(t) + \alpha_2x^2(t) + ... + \alpha_mx^m(t) \tag{5.7}$$

Por fim, pode-se igualar as Equações (5.6) e (5.7), obtendo:

$$\begin{aligned} y(t) &= \alpha_1(a_1u(t-1) + a_2u(t-2) + ... + a_nu(t-n)) \\ &+ \alpha_2(a_1u(t-1) + a_2u(t-2) + ... + a_nu(t-n))^2 \\ &+ ... + \alpha_m(a_1u(t-1) + a_2u(t-2) + ... + a_nu(t-n))^m \end{aligned} \tag{5.8}$$

5.2 SÉRIE DE VOLTERRA

Uma série de Volterra é caracterizada como a expansão de um sistema não linear, dinâmico e invariante no tempo. Mais precisamente, ela é vista como a convolução de um sistema

unidimensional, a fim de poder capturar as dinâmicas deste sistema ao tempo, comumente expressado como efeitos de memória deste sistema.

Matematicamente, ela torna-se uma série de Taylor capaz de modelar efeitos de memória, visto que esta é capaz apenas de descrever pontos instantâneos. Ou seja, uma combinação linear de uma série de Taylor com uma integral de convolução. Seja um sistema linear e invariante no tempo, caracterizado por um impulso $h(t)$ e com saída $y(t)$. A sua convolução unidimensional desta com um sinal de entrada $x(t)$ será:

$$y(t) = \int_{-\infty}^{\infty} h(\tau)x(t-\tau)d\tau \tag{5.9}$$

Se o sistema da Equação (5.9) tiver uma resposta ao impulso finita de duração M, a integral poderá ter como limite superior M. Se o sistema for causal, então o limite inferior pode ser substituído por zero. Pode-se considerar no restante deste texto que o sistema linear e invariante no tempo será sempre causal e finito, com duração M.

Da mesma forma que foi vista com a convolução, pode-se considerar uma série de Taylor de um sinal não linear e sem memória, em torno do ponto $x_o(t)$, na forma:

$$y(t) = \sum_{p=0}^{\infty} c_p[x(t) - x_o(t)]^p \tag{5.10}$$

Para a modelagem de um RFPA, o termo c_o será sempre zero, visto que a saída do PA sempre será nula quando sua entrada é nula, e por isto este termo poderá ser omitido daqui em diante. Outro ponto interessante de notar é que a série poderá ser truncada, ou seja, limitada a um valor de limite superior P, o que será considerado daqui para frente. Obviamente, ao truncar a série, perde-se acurácia dela para representar o modelo, em troca de reduzir seu número de termos.

A combinação linear das Equações (5.9) e (5.10) nos dará a Série de Volterra:

$$y(t) = \sum_{p=1}^{P} \int_0^M ... \int_0^M h_p(\tau_1, ..., \tau_p)x(t-\tau_1)...x(t-\tau_p)d\tau_1...d\tau_p \tag{5.11}$$

A expressão $h_p(\tau_1, ..., \tau_p)$ é chamada de *kernels* de Volterra, ou *kernels* do modelo. Determinar os valores destes para que a Equação (5.11) se aproxime da relação entrada/saída desejada torna-se o problema em torno da modelagem comportamental de um sistema não linear. A Equação (5.11) também pode ser escrita da forma:

$$y(t) = \sum_{p=1}^{P} H_p[x(n)] \tag{5.12}$$

em que $H_p[x(n)]$ será um operador não linear de Volterra.

Porém, até agora todo o tema abordado foi considerando um sistema contínuo no tempo. Isto não será a realidade na modelagem de dados, visto que as medições obtidas para linearização de um RFPA sempre serão de natureza discreta, onde os dados são obtidos a partir da amostragem no tempo do elemento. Com isto, a forma discreta da série de Volterra vista na Equação (5.11) pode ser descrita como:

$$y[n] = \sum_{p=1}^{P} \sum_{q_1=0}^{M} \sum_{q_2=q_1}^{M} ... \sum_{q_p=q_{p-1}}^{M} h_p(q_1, q_2, ..., q_p) \prod_{j=1}^{p} x[n - q_j] \tag{5.13}$$

em que $y[n]$ e $x[n]$ são a saída e entrada discretas, respectivamente. A ordem de truncamento é representada por P, a memória do modelo é representada por M e os *kernels* são representados por $h_p(q_1, q_2, ..., q_p)$.

Para a modelagem de um RFPA, é interessante converter a Equação (5.13) em seu equivalente em passa-baixas para a resposta do sistema a um impulso, que descreverá a relação entre os sinais complexos da envoltória de entrada $\tilde{x}[n]$ e saída $\tilde{y}[n]$. A série de Volterra em sua representação discreta e em passa baixa foi elaborada em (BENEDETTO; BIGLIERI; DAFFARA, 1979) e será representada por:

$$\begin{aligned} \tilde{y}[n] = & \sum_{p=1}^{P} \sum_{q_1=0}^{M} \sum_{q_2=q_1}^{M} ... \sum_{q_p=q_{p-1}}^{M} \sum_{q_{p+1}=0}^{M} \sum_{q_{p+2}=q_{p+1}}^{M} ... \sum_{q_{2p-1}=q_{2p-2}}^{M} \\ & \tilde{h}_{2p-1}(q_1, q_2, ..., q_{2p-1}) \prod_{j1=1}^{p} \tilde{x}[n - q_{j1}] \prod_{j2=1}^{2p-1} \tilde{x}^*[n - q_{j2}] \end{aligned} \tag{5.14}$$

em que $2p - 1 = P$ é a ordem de truncamento da série e $\tilde{h}_{2p-1}(q_1, q_2, ..., q_{2p-1})$ são os kernels de Volterra em seu equivalente em passa baixa. o valor $\tilde{x}^*[n]$ será o complexo conjugado de $\tilde{x}[n]$.

A principal propriedade da Equação (5.14) é o fato de seus coeficientes (ou *kernels*) serem lineares nos parâmetros. Isto significa que a extração dos coeficientes do modelo poderá ser feita com algoritmos lineares, com complexidade computacional reduzida, como o dos Mínimos Quadrados.

5.2.1 Problemáticas da Série de Volterra

Um dos principais problemas da implementação de uma Série de Volterra, especialmente quando o intuito é utilizá-la em implementações em HW que possam impor restrições de memória de armazenamento, é o número de coeficientes que esta poderá gerar. Considerando os coeficientes $\tilde{h}_{2p-1}(q_1, q_2, ..., q_{2p-1})$ da Equação (5.14), com uma ordem de truncamento 2P-1 e memória M, o número total de coeficientes N de uma série pode ser calculado por

(ISAKSSON; WISELL; RÖNNOW, 2006):

$$N = \sum_{p=0}^{P-1} \frac{[(M+p)!]^2(M+p+1)}{(M!)^2(p!)^2(p+1)} \tag{5.15}$$

Extrair um grande número de coeficientes não só traz contras no total de memória consumida pelo modelo, mas também introduz complexidade no cálculo dos coeficientes durante a identificação do sistema. Isto prejudica, por exemplo, implementações que requerem atualização dos valores dos coeficientes em tempo real, exigindo *hardwares* de maior capacidade e que elevarão o custo do projeto.

Além disto, conforme o número de coeficientes cresce, técnicas lineares de extração de parâmetros, como o Método dos Mínimos Quadrados (MMQ), tem seu desempenho comprometido. Isto ocorre visto que a técnica tem como um dos seus passos calcular uma matriz de autocorrelação inversa.

5.3 APROXIMAÇÃO DE VOLTERRA COM FUNÇÕES UNIDIMENSIONAIS

Uma maneira atrativa de reduzir o número de coeficientes de uma série de Volterra, sem gerar grandes impactos em seu desempenho, são aproximações do modelo. Nesta seção, será vista a sua aproximação por funções unidimensionais.

Se considerarmos uma função multidimensional discreta, em que sua dimensionalidade é dada no conceito de uma memória de tamanho M, teremos:

$$y[n] = f(x[n], x[n-1], ..., x[n-M]) \tag{5.16}$$

Esta função de dimensão $M+1$ pode ser representada na forma de uma combinação de $M+1$ funções unidimensionais, na forma:

$$y[n] = \sum_{m=0}^{M} g_m(x[n-M]) \tag{5.17}$$

em que g_m serão as $M+1$ funções unidimensionais. Como visto na seção anterior, pode-se expandir a função vista na Equação (5.16) através de uma série de Taylor e através de algumas manipulações, obter uma série de Volterra discreta vista na Equação (5.13) ou seu equivalente passa-baixa vista na Equação (5.14). O mesmo pode ser feito para a função unidimensional da Equação (5.17), e será descrito a seguir.

Se for expandida cada um dos termos $g_m(x[n-M])$ através de uma série de Taylor,

com um truncamento em P, obtemos:

$$y[n] = \sum_{p=1}^{P} \sum_{m=0}^{M} a_{pm} x^p[n-M] \tag{5.18}$$

em que a_{pm} serão os coeficientes da série de Taylor. A representação da Equação (5.18) em passa-baixa será:

$$y[n] = \sum_{p=1}^{P} \sum_{m=0}^{M} \tilde{b}_{2p-1,m} |\tilde{x}[n-m]|^{2p-2} \tilde{x}[n-m] \tag{5.19}$$

em que $\tilde{x}[n]$ e $\tilde{y}[n]$ serão a entrada e saída da envoltória complexa do RFPA, $\tilde{b}_{2p-1,m}$ são os coeficientes complexos da série, e a ordem de truncamento será igual a $2P-1$.

A Equação (5.19), que representa a função unidimensional em seu equivalente passa-baixa também tem a propriedade de ser linear em seus parâmetros, similar a série de Volterra em passa-baixa da Equação (5.14), o que novamente será atrativo. $|\tilde{x}[n-m]|$ será o módulo de $x[n-m]$.

5.4 PROPRIEDADES IMPORTANTES DE SÉRIES DE VOLTERRA E SUAS VARIAÇÕES

Algumas importantes propriedades da Série de Volterra e suas derivações foram descritas na literatura, em particular quando estas serão aplicadas para a modelagem comportamental de RFPAs. Esta seção irá descrever algumas destas, que serão importantes durante este trabalho.

Um interessante fundamento, descrito inicialmente em (BENEDETTO; BIGLIERI; DAFFARA, 1979), refere-se ao fato de que apenas termos ímpares (ou seja, aqueles elevados a números ímpares, como 1, 3 e 5) irão contribuir para o desempenho de um modelo Volterra durante a modelagem comportamental de um RFPA. Isto é justificado ao assumir que, para um sinal de envoltória complexa que carregará a informação, a largura de banda deste será muito inferior quando comparada com a frequência da portadora, e com isso apenas os termos ímpares contribuirão para o desempenho de um modelo passa-baixa.

Além disto, o número de entradas complexas conjugadas (tanto instantânea como em valores passados) no equivalente em passa-baixa de Volterra sempre será um a menos do número de entradas complexas não conjugadas.

6 REDES NEURAIS PARA A MODELAGEM COMPORTAMENTAL

Este capítulo tem como objetivo uma introdução ao uso de redes neurais para a identificação de sistemas, assim como avanços feitos no estudo de redes neurais na aplicação da identificação e modelagem de RFPAs.

O uso de redes neurais artificiais (ANN) tem tido um crescimento nas últimas décadas, tanto em problemas de classificação como problemas de modelagem. Para o segundo caso, o tipo mais popular vem sendo as redes *feed-forward*, com um foco maior nos modelos *Perceptron* Multicamadas (MLP) e a rede com Função de Base Radial (RBF). Utilizando técnicas corretas de treinamento, estas serão capazes de representar de maneira satisfatória a dinâmica não linear da função de transferência de RFPAs (LIU; BOUMAIZA; GHANNOUCHI, 2004), (PEDRO; MAAS, 2005).

6.1 A TOPOLOGIA DE UMA ANN FEEDFORWARD

Uma rede neural de topologia *FeedForward* é definida como aquela em que os dados trafegarão pela rede em apenas um sentido: da primeira camada de entradas até a última camada de saída.

Uma ANN do tipo *FeedForward* com duas camadas ocultas pode ser vista na Figura 11. Nela, o conjunto $[u_1, u_2, ..., u_n]$ serão as *n* entradas do modelo, que serão utilizadas para estimar o conjunto $[y_1, y_2, ..., y_m]$ de *m* saídas. Como dito, é possível ver que os neurônios da camada de entrada são conectados diretamente na primeira camada oculta, que seguirá sendo conectada na segunda camada oculta, para então os neurônios serem conectados para formar a camada de saída. O dado trafegará da esquerda para a direita sempre.

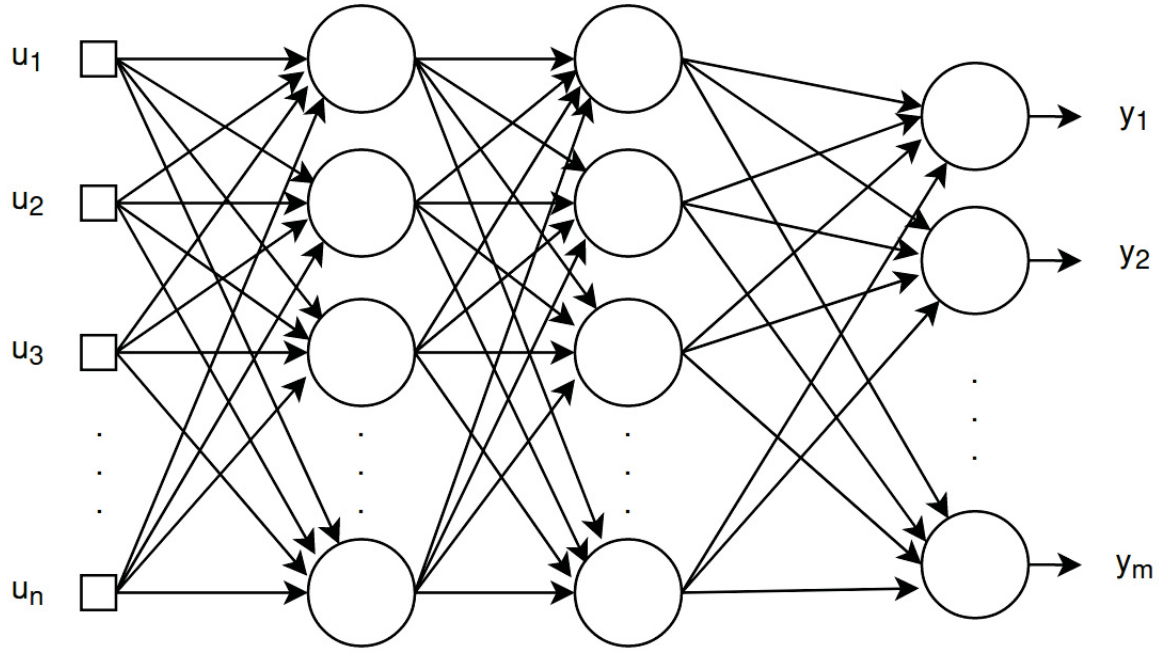
Outro ponto importante de ressaltar sobre a Figura 11 é sua característica totalmente conectada. Isto significa que cada nó tem como entrada os resultados dos neurônios da camada anterior. Ou seja, todo neurônio da segunda camada oculta é conectado a todos os neurônios da primeira camada oculta.

6.2 REDE NEURAL MULTICAMADAS PERCEPTRON

Após ver a estrutura total de uma rede *feedforward*, que será utilizada por redes neurais de diferentes formas, será apresentado um dos principais tipos de ANNs, a Perceptron Multi Camadas.

Esta é caracterizada pelo formato do seu neurônio, com todos estes sendo conectados de acordo com a Figura 11. O neurônio de uma rede MLP pode ser visto na Figura 12, onde o conjunto de n elementos $[x_1, x_2, ..., x_n]$ serão as múltiplas entradas de cada neurônio, que

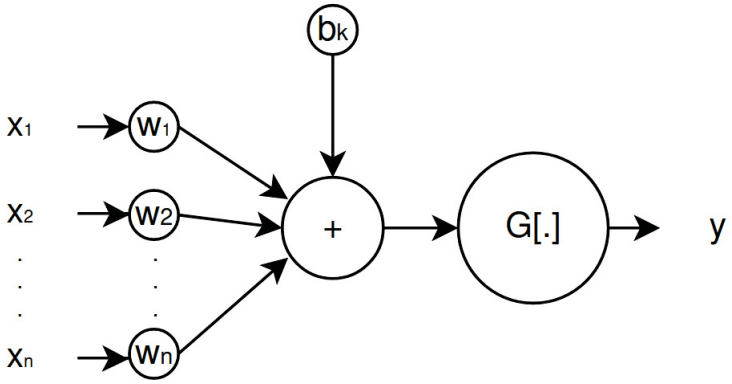
FIGURA 11 – Topologia de uma ANN FeedForward.



FONTE: O autor (2020)

após passar por uma função de ativação G[.], terá como resultado de saída y.

FIGURA 12 – Neurônio utilizado em uma ANN MLP.



FONTE: O autor (2020)

Na Figura 12, o conjunto $[w_1, w_2, ..., w_n]$ será o conjunto de n pesos dado as entradas, e b_k é o bias. Cada entrada é multiplicada pelos pesos e somada com o bias, antes de passar pela função de ativação G[.]. Este valor parcial que será entregue a G[.] e que representa as entradas já com seus pesos será chamado de v_k daqui em diante. O processo de treinamento nada mais será do que definir o conjunto de pesos $[w_1, w_2, ..., w_n]$ e bias b_k para cada neurônio que melhor represente o sistema desejado.

Cada neurônio pode ser visto como um nó do sistema. Um neurônio nada mais é do

que uma abstração de um neurônio biológico, que apresentará um potencial de passagem da célula. Ou seja, de maneira análoga a um transistor operando como chave, valores abaixo de um valor potencial previamente estabelecido resultarão em um valor de saída do neurônio, enquanto valores acima deste potencial farão com que o neurônio apresente um comportamento diferente em sua saída.

O que definirá quando o neurônio apresentará cada valor e o formato dos valores de sua saída, assim como refletirá a habilidade de uma ANN MLP em modelar sistemas, será a sua função de ativação G[.].

6.2.1 Funções de Ativação de uma MLP

Um dos modelos mais simples de função de ativação é a Função Degrau. Esta será descrita por:

$$G[.] = \begin{cases} 1, \text{se } v_k \geq 0 \\ 0, \text{se } v_k < 0 \end{cases} \tag{6.1}$$

em que caso o valor intermediário v_k for maior que 0, a saída do neurônio será igual a 1. Caso contrário, a saída será igual a zero. Por mais simples de ser aplicada, esta função não trará bons resultados caso aplicada em uma rede neural para regressão de dados, visto que este problema não é de característica binária, como pode ocorrer com um problema de classificação. Para a regressão, haverá infinitos valores entre 0 e 1 que precisarão também ser representados.

A fim de introduzir uma função não binária, um modelo comumente utilizado é a chamada Função Sigmoide, também conhecida por Função Logística, definida por:

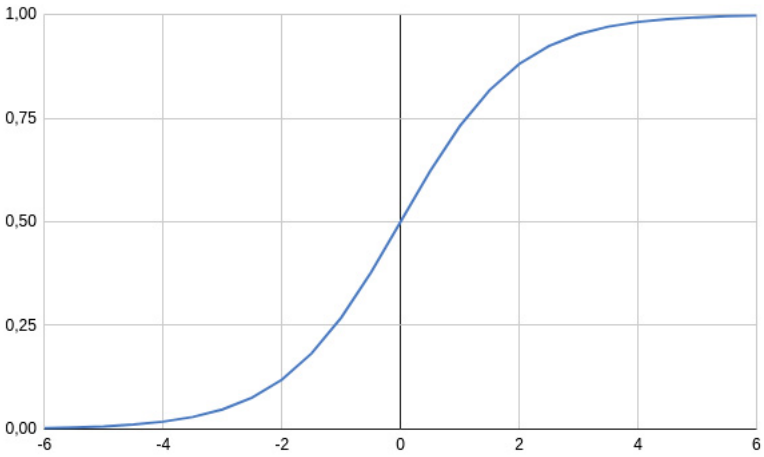
$$G[.] = \sigma(x) = \frac{1}{1 + e^{-x}} \tag{6.2}$$

A Equação (6.2) pode ser melhor vista na Figura 13, para entrada no intervalo $[-6; 6]$. Pode-se perceber que, para valores tendendo a ∞ , a saída da função sigmoide tende a 1, enquanto para valores de entrada tendendo a $-\infty$, a saída tenderá a zero.

A função sigmoide já contribuirá com um comportamento não linear, visto que oferece um degrau suave com valores intermediários entre 0 e 1. Porém, esta função tem um revés: ela não apresentará valores de saída negativos. Embora para alguns modelos isto torna-se vantajoso, para o caso de RFPAs, isto virá a ser um problema, visto que a envoltória complexa de saída poderá assumir valores negativos, especialmente no momento em que formos modelar a fase.

Para superar o problema de não haver valores negativos de saída, uma boa alternativa

FIGURA 13 – Comportamento de uma Função Sigmoide.



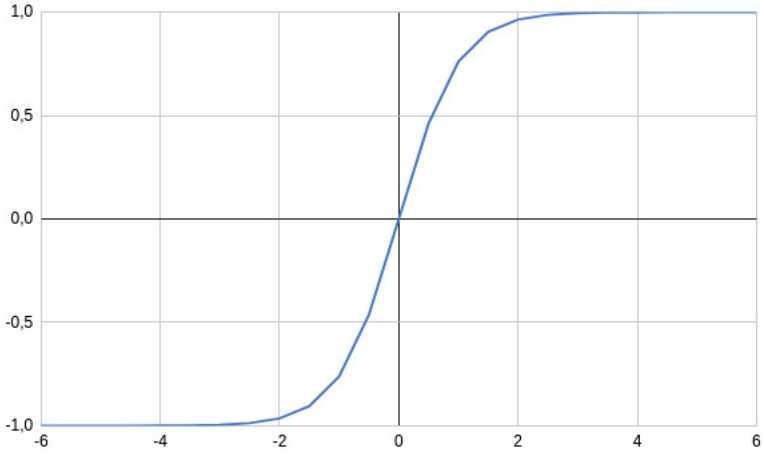
FONTE: O autor (2020)

será a função tangente hiperbólica. Esta é definida por:

$$G[.] = tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{6.3}$$

O comportamento da Equação (6.3) para uma entrada no intervalo $[-6; 6]$ pode ser visto na Figura 14. Nota-se que o comportamento não linear da Equação (6.3) é similar em forma ao comportamento da Equação (6.2). Porém, quando a entrada da função assumir valores negativos, sua saída refletirá um valor negativo e, conforme a entrada tender a $-\infty$, sua saída tenderá a -1.

FIGURA 14 – Comportamento de uma Função Tangente Hiperbólica.



FONTE: O autor (2020)

6.2.2 Exigências para uma função de Ativação em Rede Neural

Anteriormente, foram apresentadas três funções, e comentado um pouco de sua eficácia em problemas de regressão. Obviamente, estas não são as únicas funções que poderiam ser usadas para a ativação de um neurônio perceptron.

O principal parâmetro para determinar o uso de uma função de ativação é o seu desempenho para o conjunto de dados. Porém, além do desempenho, algumas regras precisam ser seguidas para que uma função de ativação possa ser utilizada em uma ANN MLP de maneira eficaz (HAYKIN, 1998). Estas são:

- **Não linearidade**: Quando a função de ativação utilizada for não linear, é possível provar que uma rede neural com duas camadas ocultas pode ser um aproximador universal (considerando um número ilimitado de neurônios por camada) (CYBENKO, 1989);
- **Limites Definidos e Finitos**: quando os limites de uma função de ativação são finitos e definidos, os métodos de treinamento tendem a ser mais estáveis;
- **Continuamente Diferenciável**: a função deverá possuir uma derivada clara e contínua em todos os seus pontos de atuação, a fim de que o principal método de treinamento, o gradiente descendente (a ser comentado nas próximas seções) tenha uma bom desempenho (ARORA, 2006).

6.3 TREINAMENTO DE UMA REDE NEURAL MLP

Por mais que não seja o foco principal deste trabalho, é importante comentar um pouco sobre o processo de treinamento de uma MLP, a fim de poder caracterizar e recordar deste processo e sua complexidade durante os próximos capítulos.

O processo de treinamento de uma ANN MLP também é comumente chamado de processo de aprendizado. Ele consistirá em ajustar os pesos e coeficientes de cada um dos neurônios do modelo, a fim de minimizar o erro de previsão em relação aos alvos (com este valor sendo obtido pela função erro). Alguns algoritmos já foram propostos na literatura mas, de maneira disparada, o mais comum é o método do *backpropagation*.

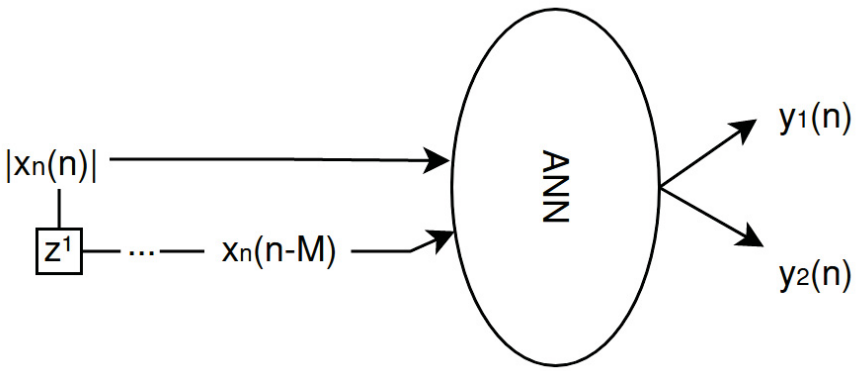
O *backpropagation* consistirá em computar o gradiente da função erro de um par fixo de entrada/saída x_n, y_m, ajustando seus coeficientes wn a fim de minimizar o erro entre a saída prevista com y_m. O objetivo então torna-se encontrar o conjunto de coeficientes que minimizará este erro. Um algoritmo comumente utilizado é a descida do gradiente. O *backpropagation* então servirá para determinar o melhor caminho para encontrar o caminho que minimizará o erro.

6.4 REDES NEURAIS COM VALORES REAIS PARA A MODELAGEM COMPORTAMENTAL DE RFPAS

As redes neurais comentadas nas seções anteriores deste capítulo podem ser utilizadas para a modelagem de RFPAs em passa-baixa. Porém, a grande parte dos algoritmos de ANNs trabalha apenas com valores reais, tanto em seus dados como em seu processamento interno. Por isto, torna-se essencial utilizar topologias que possam adequar conjuntos de dados complexos para operar com estas ANNs de valores reais. Esta seção irá trazer algumas das topologias propostas na literatura, além de comentar sobre seu desempenho e complexidade para a modelagem comportamental de RFPAs.

Inicialmente, pode-se aplicar como entradas da rede apenas a amplitude da envoltória complexa do sinal de entrada, e com as saídas da rede (e consequentemente os alvos durante o treinamento), sendo a função que descreverá as distorções AM-AM e AM-PM do RFPA (ISAKSSON; WISELL; RÖNNOW, 2005). Esta pode ser vista na Figura 15, em que $|x(n)|$ representa o módulo da envoltória do sinal, M é a ordem da memória e y_1 e y_2 são as distorções AM-AM e AM-PM do RFPA, respectivamente. Embora este modelo possa obter desempenho similar a um Modelo de Hammerstein na modelagem de um RFPA, ele deixa de considerar a influência das componentes de fase atual e passadas da envoltória do sinal de entrada e seus valores passados, resultando em importantes informações não consideradas pelo modelo.

FIGURA 15 – Arquitetura para uma ANN com o módulo da envoltória do sinal como entrada, e duas saídas: as distorções AM-AM e AM-PM do RFPA.

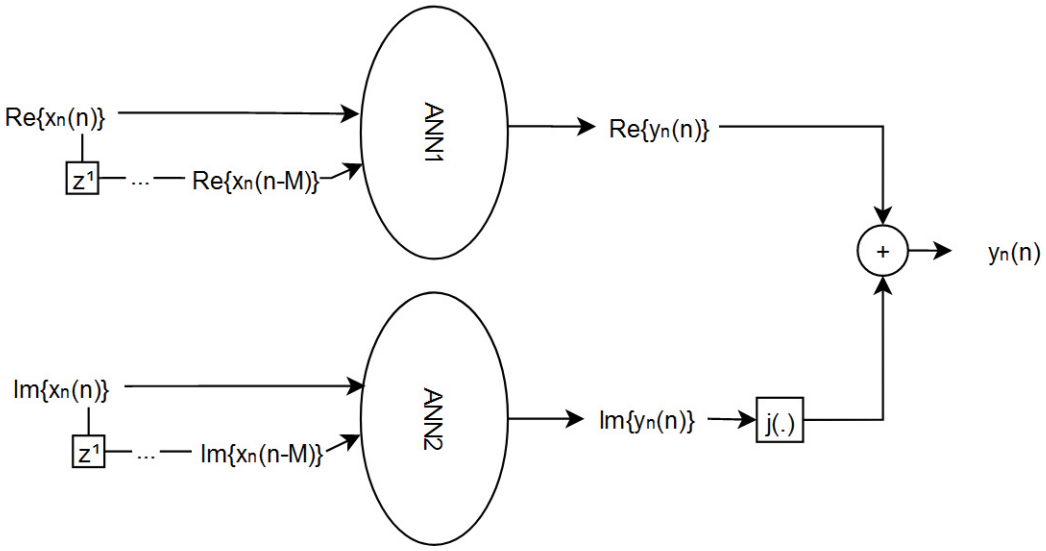


FONTE: O autor (2020), adaptado de (ISAKSSON; WISELL; RÖNNOW, 2005).

A fim de incluir também a influência da componente de fase para gerar a saída do RFPA, uma simples alternativa proposta em (IBUKAHLA et al., 1997) seria separar a envoltória em componentes real e imaginária, assim como incluir duas ANNs na arquitetura: a primeira é responsável por prever o valor da parte real da envoltória de saída (tendo como entrada a parte real da envoltória), enquanto a segunda funcionaria de forma idêntica, mas com o valor da parte imaginária. Por fim, as saídas das duas redes poderão ser somadas para obter o resultado final, conforme visto na Figura 16. Este modelo sofre de um revés: a convergência é prejudicada

devido dois treinamentos independentes ocorrerem simultaneamente (BENVENUTO; PIAZZA; UNCINI, s.d.). Além disto, devido o fato de serem duas redes separadas, toda a complexidade computacional do modelo será dobrada, seja no número de coeficientes que deverão ser armazenados na memória do HW utilizado, assim como no total de operações executadas visto que o processo de treinamento ou atualização dos pesos ocorrerá sempre duas vezes.

FIGURA 16 – Arquitetura para ANNs, em que a parte real e imaginária são separadas em redes distintas.



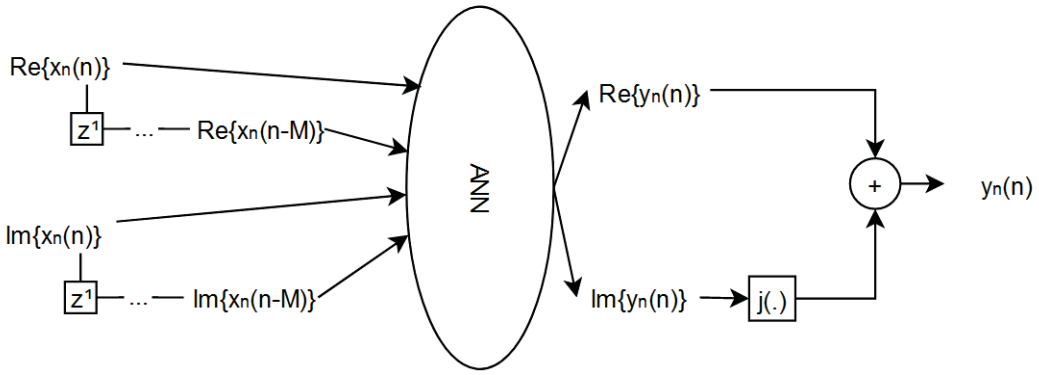
FONTE: O autor (2020), adaptado de (IBUKAHLA et al., 1997).

Porém, em (LIU; BOUMAIZA; GHANNOUCHI, 2004), um modelo similar ao da Figura 16 foi proposto, obtendo um melhor desempenho. Este poderá ser visto na Figura 17, com sua principal diferença sendo o fato de que a parte real e imaginária da envoltória de saída serão estimadas por uma rede única. Isto deverá evitar os problemas de convergência do modelo da Figura 16, além de evitar um aumento da complexidade computacional, sem afetar o desempenho.

Até agora, todos os modelos envolviam ter como entradas ou apenas o valor absoluto da envoltória, ou sua parte real e imaginária. Em (LIMA; CUNHA; PEDRO, 2011), uma nova forma de tratar os valores de entrada foi proposta. Esta consiste em uma rede única, similar a Figura 17, mas agora tendo como entradas o módulo $a_n(n)$ e valores passados da envoltória, além do seno e cosseno da diferença de fase da entrada. As saídas a serem estimadas por esta rede são o módulo $b_n(n)$ e a fase $\phi_n(n)$. Para poder estimar a envoltória de saída do RFPA, é necessário somar à fase a ser prevista $\phi_n(n)$ o valor da componente de fase de entrada $\theta_n(n)$, visto que os valores de entrada consistiam em uma diferença de fase de θ.

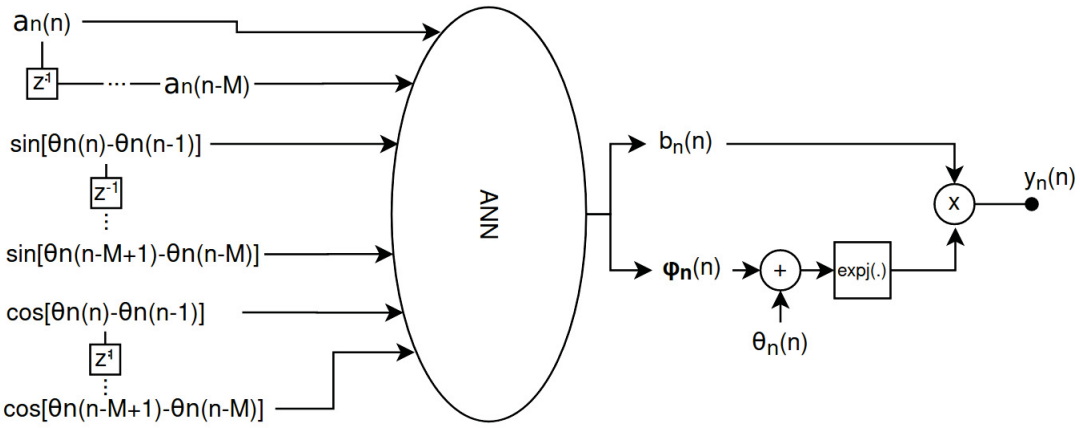
Porém, um estudo comparativo feito em (FREIRE; FRANÇA; LIMA, 2014) mostrou que um melhor custo-benefício foi obtido entre desempenho e complexidade computacional

FIGURA 17 – Arquitetura para ANNs, em que a parte real e imaginária são estimadas separadamente, porém em uma rede única.



FONTE: O autor (2020), adaptado de (LIU; BOUMAIZA; GHANNOUCHI, 2004).

FIGURA 18 – Arquitetura para ANNs, considerando módulo e seus valores passados, assim como o seno e cosseno da diferença de fase da envoltória de entrada.

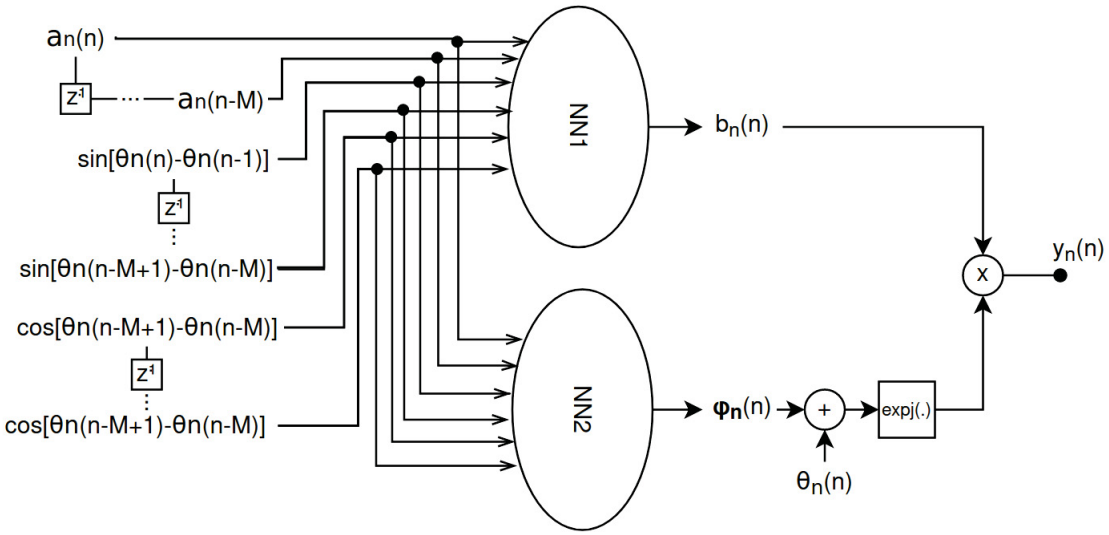


FONTE: O autor (2020), adaptado de (LIMA; CUNHA; PEDRO, 2011).

de um novo modelo, visto na Figura 19, em relação ao modelo da Figura 18. Ambas terão as mesmas entradas, com a diferença das redes serem divididas. Ou seja, uma irá prever a saída $b_n(n)$ e a outra será responsável pela fase $\phi_n(n)$.

Aqui, um ponto interessante na discussão de topologias foi atingido. Embora trabalhos passados mostraram que treinar duas redes separadamente resultava em uma pior convergência, além de trazer um maior número de operações para treinar aumentando a complexidade computacional do modelo, a partir da rede da Figura 19, foi começado a ser visto um custo-benefício interessante do desempenho contra complexidade computacional, mostrando que embora estes modelos sejam maiores, a estimativa de redes neurais com valores reais com dados de uma envoltória complexa de um RFPA é melhor ao separar os valores a serem previstos

FIGURA 19 – Arquitetura para ANNs, considerando módulo e seus valores passados, assim como o seno e cosseno da diferença de fase da envoltória de entrada.



FONTE: O autor (2020), adaptado de (FREIRE; FRANÇA; LIMA, 2014).

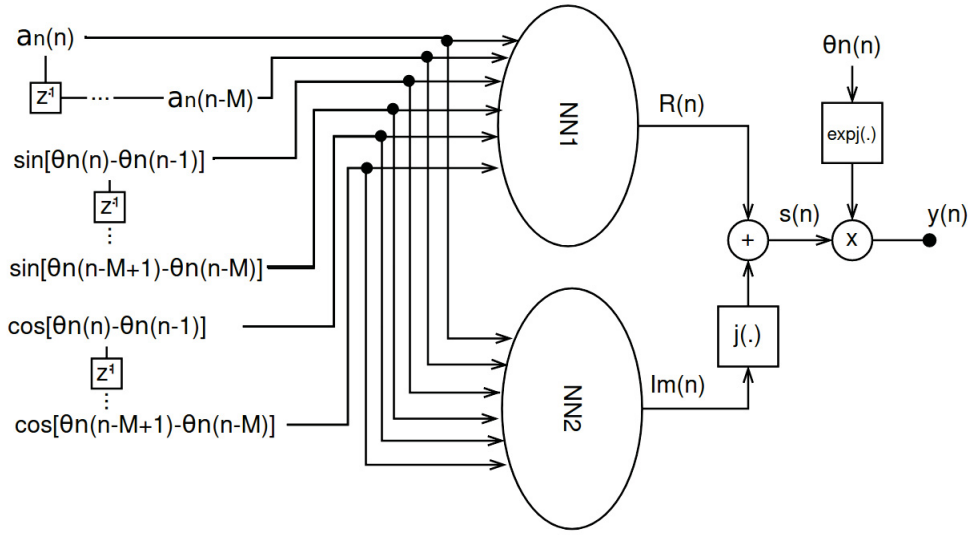
em duas ANNs distintas, fazendo previsões parciais antes de montar o valor final de saída da envoltória.

Com estas considerações, o último modelo a ser comentado nesta seção é visto na Figura 20. Neste, a grande diferença em relação ao modelo da Figura 19 são os valores a serem estimados por cada uma das redes. Esta nova topologia foi apresentada em (FREIRE; FRANCA; LIMA, 2015), em que uma rede é responsável por realizar a estimativa da parte real, enquanto a outra é responsável pela estimativa da parte imaginária da saída, decrescida da fase $\theta_n(n)$ (que será depois somada nestes valores para gerar a saída final).

6.5 REDES NEURAIS COM VALORES COMPLEXOS

Grande parte das metodologias inicialmente propostas para Redes Neurais previa que estas trabalhariam apenas com dados de natureza real, tanto nas camadas de entrada/saída, assim como nas suas camadas ocultas. Porém, alguns problemas mostraram uma necessidade de que estas ANNs fossem capazes de lidar com valores complexos nestes dois estágios. Para isto, foram propostas Redes Neurais Artificiais baseadas em Valores Complexos (CVNN) (HIROSE, 2012), (AIZENBERG, 2011), (SURESH; SUNDARARAJAN; SAVITHA, 2013). O uso destas CVNNs que trabalharão com dados complexos em suas camadas ocultas mostrou alguns benefícios em relação à sua contra parte real: o tempo de convergência tende a ser entre duas e três mais veloz que redes neurais baseadas em números reais; o número de parâmetros necessários tende a reduzir em metade; os modelos tendem a ser mais generalistas e ter menos problemas com *overfitting* que ANNs tradicionais (NITTA, 1997), (NITTA, 2004), (HIROSE; YOSHIDA, 2012). Além disto, para sistemas que geram conjuntos de dados no domínio

FIGURA 20 – Arquitetura para ANNs, considerando módulo e seus valores passados, assim como o seno e cosseno da diferença de fase da envoltória de entrada.



FONTE: O autor (2020), adaptado de (FREIRE; FRANCA; LIMA, 2015).

complexo, torna-se mais simples trabalhar com modelos que não exijam a conversão destes para o domínio real, além de simplificar a topologia necessária para a ANN, ao reduzir o número total de coeficientes a serem estimados (não significando necessariamente um ganho de precisão, mas sim uma redução no tempo de treinamento). Porém, grande parte destes resultados foram vistos em problemas de classificação, com poucos estudos trazendo os benefícios destas redes complexas em problemas de regressão.

Para avançar neste tema, alguns conceitos e suas problemáticas serão importantes. Uma função de ativação de um neurônio necessita ter uma relação não linear de entrada/saída, ter limites definidos e ser derivável em todos os pontos do plano que esta operar (seja este real ou complexo) (HAYKIN, 1998). Porém, o teorema de Liouville diz que no domínio complexo, uma função complexa inteira e limitada é uma função constante. Isto traz problemas para a aplicação em redes neurais, visto que uma função constante não torna-se uma boa função de ativação, por não possuir o comportamento não linear.

Neste trabalho, duas metodologias podem ser vistas: a primeira será a CVNN Parcial, que terá valores complexos em suas entradas e saídas, porém seus coeficientes e função de ativação estarão no domínio real; já a segunda será a CVNN Completa, que não só poderá ter seus valores de entrada complexos, mas também terá as camadas ocultas no domínio complexo, ou seja, o processo de treinamento resultará em coeficientes complexos, assim como as funções de ativação poderão ser complexas.

6.5.1 CVNNs Parciais

Neste modelo, um conjunto de dados no domínio complexo é utilizado para identificar o sistema, porém este conjunto de entrada e alvos precisarão de topologias que os transformem para o domínio real, sem perda de informação, com a arquitetura da ANN podendo ser então da forma de uma ANN MLP, inclusive em seu processo de treinamento. Esta divisão pode tanto ser feita na forma retangular (parte real e parte imaginária), como da forma polar (componente de módulo e ângulo). Com isto, pode-se permanecer com tanto funções de ativações como coeficientes no plano real, não sendo necessárias novas técnicas nas camadas ocultas da rede. Isto resulta em uma rede com mais camadas e saídas do que sua contraparte real, e consequentemente também uma necessidade de mais neurônios e coeficientes a serem estimados, resultando em modelos maiores e mais custosos para serem treinados. As topologias vistas anteriormente na Seção 6.4 se encaixam no conceito de CVNN Parcial, visto que o problema de modelagem de um RFPA trata com valores complexos, e estes foram modelados em sistemas baseados em valores reais.

Além disto, a técnica de gradiente descendente utilizada no treinamento ainda é baseada em valores reais, não sendo capaz de completamente refletir o gradiente do valor complexo, o que pode resultar em perda de acurácia, principalmente introduzindo distorção de fase nos valores estimados (KIM; ADALI, 2002). Por fim, a eficácia do treinamento e convergência do modelo dependem de uma inicialização adequada do modelo, com valores corretos de início, além de ajustes na taxa de aprendizado (YANG; HO; SIU, 2007), (ZHANG; ZHANG; WU, 2009), (SAVITHA et al., 2009).

Para evitar o problema de distorção de fase que pode ocorrer durante o treinamento do modelo com coeficientes reais durante o gradiente descendente, é possível utilizar coeficientes complexos com funções de ativações reais (JIANPING, D.; SUNDARARAJAN, N.; SARATCHANDRAN, 2000), (JIANPING, Deng; SUNDARARAJAN, Narasimhan; SARATCHANDRAN, 2002), (BENVENUTO; PIAZZA, 1992). Para isto, o produto entre o valor complexo de erro com o seu conjugado é usado para a atualização dos coeficientes durante o erro. Porém, ainda há uma perda da correlação entre parte real e imaginária, visto que a parte real do erro é usada para atualizar sua componente real, e o mesmo é feito para a parte imaginária, e o gradiente utilizado no aprendizado não se torna uma real representação dos valores complexos, ainda resultando em perdas de acurácia.

6.5.2 CVNNs Completas

A fim de superar os problemas vistos anteriormente com CVNNs Parciais, as CVNNs Completas utilizarão não apenas coeficientes complexos, mas também funções de ativação complexas, que respeitem as exigências de uma ANN (característica não linear, ser limitada e ser derivável em todos os pontos do plano que esta operar). A principal mudança será no fato de que as entradas poderão ser complexas, não sendo necessário separar em valores

reais. Isto instantaneamente reduzirá o número necessário de entradas, saídas e neurônios do modelo, reduzindo sua complexidade computacional quando comparado ao modelo Parcial. Porém, a princípio, o seu neurônio continuará operando da mesma forma, com complexidade computacional semelhante, apenas necessitando das adaptações em sua função de ativação e seu treinamento, a fim de adequá-los aos dados no domínio imaginário.

Como comentado, um dos grandes problemas estará em selecionar funções de ativação complexas. Estas deverão respeitar as seguintes definições (GEORGIOU; KOUTSOUGERAS, 1992):

- $f_a(z) = f_a(x + jy) = u(x, y) + jv(x, y)$, sendo $u(x, y)$ e $v(x, y)$ não lineares e limitadas em x e y;
- As derivadas parciais $\frac{\partial u}{\partial x}$, $\frac{\partial u}{\partial y}$, $\frac{\partial v}{\partial x}$ e $\frac{\partial u}{\partial x}$ existem e são limitadas;
- $f_a(z)$ não é completa, ou seja, não é integrável em todos os pontos finitos do plano complexo;
- $\frac{\partial u}{\partial x}\frac{\partial v}{\partial y} \neq \frac{\partial v}{\partial x}\frac{\partial u}{\partial y}$, com exceção de $\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} = 0$ e $\frac{\partial v}{\partial x} = \frac{\partial u}{\partial y} = 0$.

Estas condições podem ser reduzidas ao seguinte (KIM; ADALI, 2002): Em um plano C de um domínio complexo limitado, uma função de ativação totalmente complexa e não linear $f_a(z)$ precisa ser analítica e limitada neste.

Com isto, (KIM; ADALI, 2002) propõe algumas funções que poderão ser usadas como ativação de uma CVNN Completa e que podem ser vistas na Tabela 1, assim como pontos de singularidades destas. Caso o algoritmo de aprendizagem utilize esta região de singularidade, haverá problemas de convergência do modelo. Não apenas isto, mas este processo também deverá levar em conta que a distribuição dos dados, o valor de inicialização dos coeficientes e a taxa de aprendizagem também deverão respeitar estes pontos de singularidade.

TABELA 1 – Funções de ativação para uma CVNN Completa e seus pontos de singularidade

Função de Ativação	***Singularidade***
$tanh$	$(1/2 + n)\pi$
tan	$(1/2 + n)\pi$
$atanh$	± 1
$atan$	$\pm i$
$asinh$	$\geq i$
$asin$	≥ 1

FONTE: O autor (2020)

6.5.3 Processo de aprendizado de CVNNs

No processo de aprendizado das CVNNs, os algoritmos são baseados nos utilizados em ANNs, exigindo modificações para suportarem valores complexos. Ou seja, será baseado no processo de minimização do erro durante as etapas.

Para uma MLP complexa, pode-se utilizar a técnica de *Back Propagation* e descida do gradiente adaptada para valores complexos (LEUNG; HAYKIN, 1991), assim como algumas otimizações destas baseadas nas condições descritas na seção anterior para a função de ativação (KIM; ADALI, 2002), (GEORGIOU; KOUTSOUGERAS, 1992).

A maior parte dos textos até aqui descreveram MLPs complexas. Porém, para o caso de ANNs RBF com valores complexos (CHEN; MCLAUGHLIN; MULGREW, 1994a), (CHEN; MCLAUGHLIN; MULGREW, 1994b), o processo de treinamento também torna-se uma adaptação daquele utilizado em ANNs de valores reais.

7 GMDH: MÉTODO DE GRUPO DE MANIPULAÇÃO DE DADOS

O Método de Grupo de Manipulação de Dados (do inglês *Group Method of Data Handling* é uma família de algoritmos para organização, identificação e regressão de sistemas com grande conjunto de dados multiparamétricos, ou seja, com múltiplas entradas que representarão diferentes informações sobre o sistema a ser identificado. Suas origens datam do começo dos anos 70 e, durante as décadas seguintes, múltiplas áreas o estudaram, resultando em interessantes variações e evoluções do método.

Porém, uma área que nunca estudou o uso deste método foi a identificação e modelagem aplicada à telecomunicações. Este capítulo será dedicado a introduzir o método, seus conceitos e aplicações, assim como introduzir propostas que o viabilizarão para a área de modelagem comportamental para microeletrônica. Sua divisão se dará da seguinte forma. Primeiramente, será comentado sobre o histórico do modelo, e como este se diferencia de outras técnicas de identificação e modelagem de sistemas. Em seguida, será apresentado o funcionamento do modelo baseado em dados reais, assim como variações propostas. Por fim, será comentado o modelo baseado em dados complexos, assim como modificações propostas para adequá-lo para a modelagem de RFPAs.

7.1 HISTÓRICO E DIFERENCIAIS DO MODELO

Um dos primeiros diferenciais do método, quando originalmente proposto (IVAKHNENKO, 1971), foi o de minimizar intervenções humanas em problemas de identificação e modelagem de sistemas com grandes estruturas de dados e parâmetros, ou seja, um elevado número de entradas a serem consideradas e que influenciarão o resultado de saída. Inspirado por modelos caixa-preta e seleção genética, o GMDH buscou simplificar a concepção de modelos matemáticos e reduzir a influência humana em sistemas com poucas informações e múltiplas variáveis de entrada. Um ponto interessante é o fato de este ser um dos primeiros modelos reportados de redes neurais profundas, ainda no início da década de 70, em uma época em que as discussões de redes neurais ainda não estavam focadas no número de camadas.

Em trabalhos passados, uma variedade de aplicações mostraram o potencial do GMDH como uma técnica de aprendizado profundo auto organizada. O método também tem o costume de ser chamado de Redes Neurais Polinomiais, devido suas funções de ativação poderem ser da forma polinomial, e costumam trazer bons resultados na modelagem de sistemas com elevada ordem de não linearidade, mesmo quando pequenos conjuntos de treinamentos são utilizados nestas, graças a sua capacidade de auto organização, crescimento em profundidade e combinação de entradas para gerar complexidade (IVAKHNENKO, 1971),(IVAKHNENKO, 1978).

Isto significa que é possível construir um modelo de redes neurais profundas já otimizando e reduzindo sua complexidade computacional, além de ter tempos de treinamentos muito mais rápidos. Esta característica já foi mostrada em outros campos (NGUYEN; NGUYEN-XUAN; LEE, 2020),(JEDDI; SHARIFIAN, 2020),(SATTARI; ROSHANI; HANUS, 2020),(LIN et al., 2020),(DORN et al., 2012), mas ainda não foi testada na modelagem comportamental de RFPAs ou outros componentes para telecomunicações.

As próximas subseções comentarão um pouco sobre algumas palavras chaves do método já comentadas neste texto, que serão algumas de suas características definidoras.

7.1.1 Auto organização

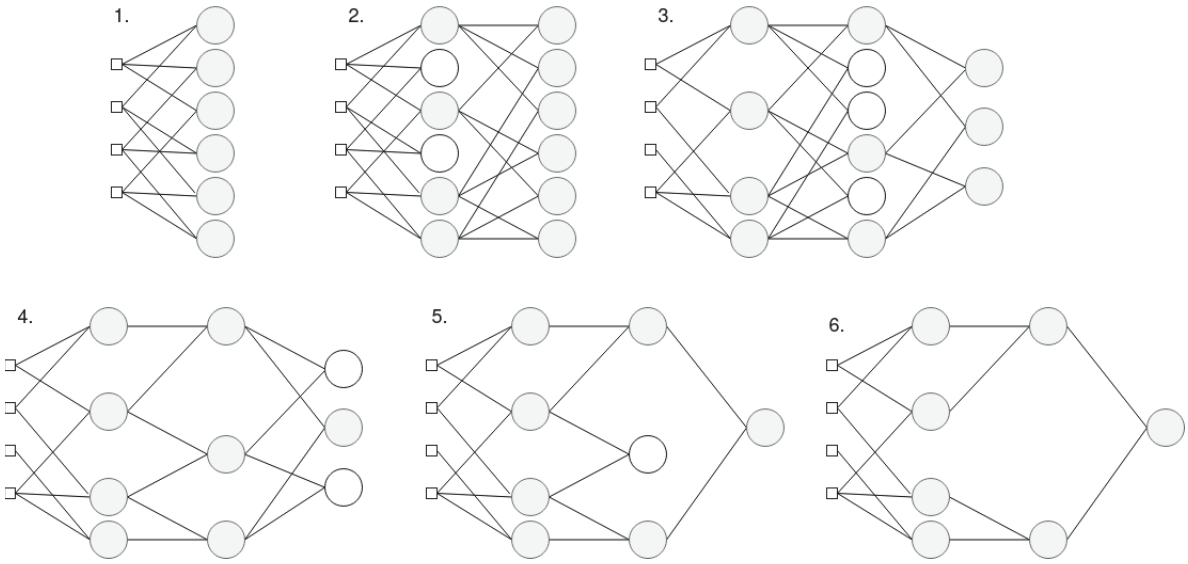
Uma das principais características do GMDH é sua natureza auto organizada. Isto significa que a estrutura do modelo, seja em seu número de nós ou número de camadas, não é definida pela projetista do modelo. Estes serão definidos durante o treinamento pelo próprio modelo, seguindo alguns parâmetros externos de desempenho desejada pelo projetista.

De fato, o treinamento do GMDH torna-se um processo único e exclusivo, que define sua característica auto organizadora. O modelo inicia apenas com um número N de entradas. Estas são combinadas duas a duas para gerar os nós da primeira camada. Após o treinamento e definição dos coeficientes destes nós, uma etapa de avaliação é realizada, a fim de selecionar os nós que melhor contribuem com o resultado parcial de saída. Esta métrica de avaliação é pré determinada, e estes modelos gerados durante as camadas intermediárias são chamados de modelo parcial. O processo de geração de uma nova camada recombinando as saídas da camada anterior, treinamento dos nós, avaliação e seleção dos melhores nós é repetido até que um parâmetro pré estabelecido seja satisfeito. Este parâmetro pode ser o desempenho total do modelo, a evolução do erro camada por camada, ou um número máximo de camadas, por exemplo.

Todo o processo descrito agora é independente de intervenção humana e feito de maneira autônoma, através de um algoritmo iterativo que continuará até um critério de parada pré-definido seja atingido, e pode ser melhor visto na Figura 21. O número de nós de uma nova camada é definido após serem selecionados os melhores nós da camada anterior, e a última camada do modelo sempre deverá possuir apenas um nó, visto que o GMDH sempre possuirá apenas uma saída e cada nó necessariamente também possui apenas uma saída.

É interessante notar na Figura 21 como o número de camadas e nós total do modelo é definido durante o treinamento, e o grande influenciador do treinamento é o conjunto de dados de entrada e alvos a serem atingidos. Com isto, cada sistema a ser identificado e seu conjunto de dados resultará em um modelo único, mesmo que seu número de entradas e métricas a serem atingidas seja o mesmo.

FIGURA 21 – Evolução da estrutura da Rede durante o Processo de Treinamento do GMDH.



FONTE: O autor (2020)

7.1.2 Seleção automática

A seleção automática é outra característica interessante do GMDH e que contribui para a auto organização. Em várias metodologias e algoritmos de identificação e regressão de sistemas, seja polinomial ou redes neurais, o usual é executar o treinamento que determinará os coeficientes primeiramente, para em seguida ter um algoritmo segregado de otimização para selecionar os melhores coeficientes que compõem o modelo e reduzir sua complexidade computacional.

No GMDH, o processo de treinamento já possuirá um otimizador integrado. Antes de gerar uma nova camada, os nós da camada recém treinada serão avaliados por parâmetros e funções pré estabelecidas, e apenas aqueles que atingirem as exigências serão selecionados para serem recombinados e compor a próxima camada. Nós não selecionados serão descartados e não influenciarão na acurácia final do modelo.

7.1.3 Amplo Espaço Combinatório

Como os nós do GMDH são definidos como a combinação dois a dois das entradas (no caso da primeira camada) ou da saída da camada anterior, seu espaço combinatório torna-se extremamente amplo. A combinação de todos os possíveis parâmetros é considerada como possivelmente benéfica ao desempenho, de maneira similar a um algoritmo de árvore, considerando o maior número de possibilidades que poderão trazer melhorias à acurácia do modelo.

Em realidade, o espaço combinatório pode tornar-se tão amplo que, no caso de um conjunto de dados com 10 entradas, a primeira camada terá 45 nós e, consequentemente, a

segunda camada possuirá 990 nós (considerando que nenhum nó foi descartado). Este valor continuará crescendo de maneira exponencial conforme novas camadas sejam adicionadas. É um elevado número de coeficientes a serem calculados, indo na contramão do intuito do método em reduzir a complexidade computacional versus métodos já bem estabelecidos para a modelagem comportamental de RFPAs. Por isto, a questão da seleção automática de nós e otimização de estrutura camada por camada durante o treinamento torna-se essencial e indispensável para reduzir complexidade computacional e evitar que o modelo cresça de maneira descontrolada.

7.1.4 Profundidade

A questão do amplo espaço combinatório não é apenas beneficiado pelo formato de combinação dois a dois das entradas para gerar nós, mas também da possibilidade do modelo crescer infinitamente em número de camadas. O tema de redes profundas (ou no inglês *deep networks*, também erroneamente referenciadas pelo nome do seu processo de treinamento, *deep learning*) está em alta, sendo contra produtivo listar a quantidade de pesquisas e avanços da comunidade científica deste tema neste trabalho, visto que modelos desta complexidade não são o foco. O importante a notar aqui é: métodos de redes profundas trazem imensos benefícios ao desempenho do modelo, mas com um grande revés: seu treinamento é extremamente exaustivo ao *hardware*, demandando grandes quantias de tempo (podendo chegar a dias, no estado da arte de redes neurais convolucionais para classificação) e recursos computacionais (muitas vezes na ordem de múltiplos servidores exclusivamente dedicados à tarefa de treinamento).

Isto é totalmente o oposto do que busca-se atingir neste trabalho. Embora o crescimento de redes em profundidade, mesmo que em apenas uma camada, beneficia muito mais em desempenho do que o crescimento em número de nós (ELDAN; SHAMIR, 2016), o aumento de complexidade computacional de métodos atuais de redes neurais não é bem-vindo em aplicações de modelagem comportamental a ser aplicada em *hardwares* embarcados, que é o caso para a linearização de RFPA, visto que estes são limitados tanto em poder de processamento, capacidade de armazenamento em memória e consumo energético. Por isto, o GMDH torna-se um modelo atraente como alternativa às técnicas de redes neurais profundas: ele oferecerá profundidade em camadas e amplo espaço combinatório, mas com um treinamento simples com um otimizador integrado.

7.2 FUNDAMENTOS DO MODELO CLÁSSICO DO GMDH

O GMDH é uma família de algoritmos de modelagem em casos de sistemas com um elevado número de variáveis de entrada com uma saída única. O modelo também é chamado de Rede Neural Polinomial devido Ivakhnenko ter utilizado como função de ativação dos nós funções polinomiais de segunda ordem. Porém, o GMDH não necessariamente precisa ser limitado a estas funções, com novas iterações do método propondo polinômios de ordens

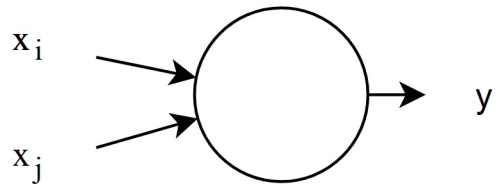
superiores e outras formas de funções de ativação. Mais especificamente, o adequado não é definir o GMDH como uma Rede Neural, por mais que sua estrutura final se assemelhe a uma rede *feedforward* e possuir funções de ativação não lineares. Porém, ainda será citado sua estrutura com o termo Rede, apenas removendo o termo Neural.

Um dos principais benefícios do GMDH sobre ANNs estará no fato de seus coeficientes serem lineares e que poderão ser calculados por métodos muito mais simples do que aqueles utilizados em redes neurais, como o MMQ.

7.2.1 O nó do GMDH

Primeiramente, será visto como funcionará um nó dentro do GMDH. Este poderá ser visto na Figura 22. Seu principal diferencial aos modelos de nós utilizados em ANNs (que também podem ser chamados de nó) é o fato deste ser limitado a apenas 2 entradas, enquanto em modelos MLP, como visto anteriormente na Figura 12, o nó acaba sendo formado pela combinação de todas as entradas do modelo e resultando em uma saída. Este fato torna-se interessante, pois a largura da rede, ou seja, o número de nós de uma camada, será determinado pelo número de valores de entrada da camada anterior, e não pelo desejo do projetista do sistema (igual ocorre em ANNs).

FIGURA 22 – Nó utilizado no GMDH.



FONTE: O autor (2020)

7.2.2 Funções de Ativação do GMDH

Em seguida, é interessante entender como a função de ativação funcionará no GMDH, e como este se diferencia contra as funções de ativação em ANNs. A função de ativação originalmente proposta por (IVAKHNENKO, 1971) é uma função polinomial, resultando então na denominação do neurônio polinomial (embora neste trabalho, será apenas referenciado como nó). A função de ativação originalmente proposta é descrita como:

$$y = a_0 + \sum_{i=1}^{m} a_i x_i + \sum_{i=1}^{m} \sum_{j=1}^{m} a_{ij} x_i x_j, \tag{7.1}$$

conhecida também como o Polinômio de Kolmogorov-Gabor. a_0 é o coeficiente que representa o *bias* do nó. a_i e a_{ij} representam os coeficientes de primeira e segunda ordem do polinômio, respectivamente. x_i e x_j representam as entradas *i* e *j* do nó.

Cada nó então terá como sua função de ativação a Equação (7.1), com cada um recebendo um par de entradas x_i e x_j resultantes da combinação dois a dois de todas as entradas resultantes para esta camada. Fica a cargo do projetista do sistema definir a ordem polinomial da função de ativação de cada nó utilizado em seu modelo, conforme a necessidade do sistema. A questão do impacto da ordem da função de ativação será extensivamente estudada nos capítulos seguintes deste trabalho.

Desta forma, um nó representado pela Equação (7.1), limitado a segunda ordem, será representado por:

$$y = a_0 + a_1 x_i + a_2 x_i^2 + a_3 x_j + a_4 x_j^2 + a_5 x_i x_j. \tag{7.2}$$

Também fica a cargo do projetista do sistema propor diferentes funções para o nó. É importante notar que as limitações impostas para uma função de ativação do GMDH são muito menores do que as vistas para ANNs no Capítulo 6.

Para entender melhor como utilizar polinômios de ordens diferentes como funções de ativação de um mesmo modelo poderá auxiliar o desempenho conforme camadas são adicionadas, uma maneira simples é alterar as ordens da função de acordo com a profundidade da rede. Mais especificamente, uma hipótese válida é que nós de camadas iniciais poderiam ter resultados melhores quando são representados por uma função de ordem igual a 3, enquanto nós das camadas posteriores apresentam melhor comportamento com uma função de primeira ordem, visto que aqui já terão sido introduzidas não linearidades, sendo apenas necessário fazer funções similares a um *buffer* para combinar os resultados obtidos nas camadas anteriores para poder gerar uma saída, além de evitar um *overfit*.

7.2.3 Treinamento do Modelo e Auto Organização de sua Estrutura

O processo de treinamento será constituído em ajustar os coeficientes da função de ativação de cada nó, camada por camada. Mesmo este polinômio tendo uma relação não linear entre entrada e saída, se sua função de ativação for um polinômio linear em seus coeficientes, ou seja, uma função derivada da Equação (7.1), seus coeficientes poderão ser determinados por técnicas lineares de regressão, como o MMQ. Isto traz um ótimo benefício comparado com outros modelos de ANNs *feed-forward*, como MLPs, que utilizam a técnica de *back propagation* para ajustar seus coeficientes durante o treinamento, em um processo muito mais exaustivo computacionalmente. Isto ocorre visto que o MMQ é computacionalmente mais simples de ser implementado. Isto não só torna o processo de treinamento mais veloz, mas também permite

este ser implementado em *hardwares* mais simples, sendo vantajoso também para aplicações que exigem a atualização em tempo real dos seus coeficientes.

7.2.3.1 Método dos Mínimos Quadrados

O MMQ, também chamado de Mínimos Quadrados Ordinários, é uma das técnicas matemáticas mais amplamente utilizadas na estimação e extração de parâmetros de sistemas. Como seu nome sugere, a técnica consiste em ajustar um conjunto de dados a fim de minimizar a soma dos quadrados das diferenças entre o valor estimado e o valor a ser atingido, através de um processo iterativo.

Partindo de uma função similar à Equação (7.1), que poderá ter termos de múltiplas ordens (por exemplo x_i^2, x_j^3,...) e termos que representam a covariância entre duas entradas (como $x_i x_j$, $x_i^2 x_j$,...). Estes todos são termos não lineares, que não necessariamente poderiam ser utilizados em um método como o MMQ.

Porém, é possível fazer uma substituição, assumindo que o resultado de termos não lineares é uma variável linear. Ou seja, $x_i^2 = w_1$ ou $x_i^2 x_j = w_2$. Desta forma, pode-se transformar a Equação (7.1) em:

$$y = a_0 + \sum_{i=1}^{m} a_i f(x_i, x_j) = a_0 + \sum_{i=1}^{m} a_i w_i \tag{7.3}$$

em que a identidade $f(x_i, x_j) = w_i$ representa que w será uma função das entradas x_i e x_j de um nó. Assim, pode-se dizer que a Equação (7.3) é totalmente linear em seus termos, e esta poderá ter seus coeficientes determinados pelo MMQ.

O MMQ então traz a tarefa de determinar os coeficientes a_i que farão o conjunto de dados representados por w se aproximar ao máximo dos valores alvo y. Isto se dará da seguinte forma: seja $y = y_{est} = a_0 + \sum_{i=1}^{m} a_i w_i$ os valores estimados do modelo, e y_{ref} os alvos a serem atingidos durante o treinamento. O MMQ consiste em minimizar a função:

$$S = \sum_{z=1}^{n} e_z^2 = \sum_{z=1}^{n} (y_{ref,z} - y_{est,z})^2 \tag{7.4}$$

em que S será o vetor de i coeficientes estimados, n representa o tamanho do vetor das saídas de referência e estimada (estes deverão ter tamanhos iguais), $y_{ref,z}$ e $y_{est,z}$ são os valores de referência e estimado no instante z, respectivamente. A diferença entre valor estimado e previsto é chamada de resíduo, e representada por e.

Porém, a Equação (7.4) ainda não representa bem a solução para determinar os coeficientes da Equação (7.3), pois esta apenas servirá para problemas do tipo $y = ax + b$. Na hora em que tanto y como x se tornam vetores, o MMQ ficará da forma $Y = AX + b$, com Y sendo um vetor com n linhas e 1 coluna e X sendo uma matriz de n linhas por z colunas, com

n sendo o número de valores instantâneos e i sendo o número de variáveis ou coeficientes a determinar. A será um vetor de tamanho n por 1 de coeficientes, e será o objetivo do problema. Este vetor de coeficientes S poderá ser calculado da forma matricial:

$$S = (X^T X)^{-1} X^T Y \tag{7.5}$$

em que X^T será a matriz transposta de X e a potência -1 em $(X^T X)^{-1}$ representará a matriz inversa. Aqui entrará um pequeno ponto de atenção: devido o fato de haver o cálculo de uma inversa para a definição dos coeficientes de um nó, torna-se novamente interessante cuidar com o número de termos gerados por cada nó, assim como seu número total destes elementos, a fim de evitar que o processo de treinamento torne-se demasiadamente exigente. Ou seja, conforme funções de ativação com um maior número de coeficientes sejam utilizadas, o processo do MMQ para cada nó torna-se mais demandante, devido ao cálculo desta matriz inversa de tamanho diretamente proporcional ao número de coeficientes a serem estimados.

7.2.3.2 Definição da estrutura

Como comentado anteriormente, durante o treinamento a rede irá se construir e otimizar automaticamente, no processo chamado de auto organização. Isto foi facilmente visto na Figura 21. A auto organização durante o treinamento consistirá em construir modelos parciais, no formato de camadas de nós formados pela combinação duas e duas de cada uma das N entradas da camada anterior, até que a métrica de parada do treinamento seja atingida.

Após a construção de cada modelo parcial, um processo de seleção dos melhores nós deste ocorrerá. Este processo pode ser facilmente modificado, mas a forma originalmente proposta consiste em calcular o Erro Quadrático Médio (MSE) de cada um dos nós, e através de um processo de filtro pré definido, selecionar os que obtiveram melhores resultados nestes modelos parciais para então serem recombinados e formarem a próxima camada. Este processo de construir modelos parciais, camada por camada, e escolher os melhores nós seguirá até que uma métrica de parada final seja atingida. No momento em que o critério de parada, como por exemplo o número máximo de camadas for atingido, caso ainda existam dois ou mais nós na camada final (ou seja, dois nós independentes com resultados diferentes de saída), será selecionado apenas o nó com o melhor desempenho, para que este componha a última camada do modelo e seu resultado gere a saída final deste modelo. Ou seja, a última camada do GMDH sempre deverá possuir apenas um nó.

Este torna-se o grande diferencial do GMDH perante outras arquiteturas de ANNs. Este é um processo completamente autônomo e sem intervenção humana, com a arquitetura final da rede sendo única de acordo com o conjunto de dados e os parâmetros pré estabelecidos. Ao construir modelos parciais, ao invés de calcular todos os coeficientes da rede em um processo único, o processo de treinamento será separado em blocos menores, onde a operação de cada

um deles será calcular os coeficientes de cada nó e depois verificar seu desempenho.

O número de nós de uma camada genérica *K* é obtido pela combinação linear:

$$N_{k+1} = \frac{N_k(N_k - 1)}{2}, \tag{7.6}$$

onde N_k poderá ser o número de entradas iniciais do modelo ou o número de saídas da camada anterior, combinadas duas a duas. Como comentado, o processo de criar novas camadas será repetido até que um critério de parada seja atingido. O número total de nós do modelo não será necessariamente o valor da Equação (7.6) multiplicado pelo número de camadas, visto que o processo de seleção reduzirá o total de nós do modelo.

Outro ponto interessante é o número de entradas a ser usado no modelo. Um modelo com apenas 3 entradas não torna-se interessante, visto que a primeira camada terá apenas 3 nós, que consequentemente serão o número máximo de nós em todas as camadas. Sistemas com poucas entradas não são ideais para o GMDH, pois não se beneficiarão da sua auto organização e amplo espaço combinacional.

7.2.4 Avaliação de desempenho e Cálculo do Erro

Para um GMDH baseado em valores reais, uma alternativa simples e eficaz é avaliar o desempenho dos nós gerados pelos modelos parciais de cada camada a partir do MSE. Este será representado por:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \overline{Y_i})^2 \tag{7.7}$$

A Equação (7.7), quando usada de métrica de desempenho do treinamento, é capaz de acompanhar o valor médio da diferença entre valor desejado e valor previsto, elevado ao quadrado. Quanto mais próximo de zero, melhor é a acurácia da previsão.

Durante o processo de treinamento, é possível definir por qual métrica um estimador como o MSE será utilizado para selecionar os melhores nós. Isto poderá ser feito simplesmente selecionando um valor mínimo por camada para que um nó seja escolhido para ser recombinado e gerar a próxima camada.

É importante notar que para uma nova camada ser possível, são necessários pelo menos 2 nós em uma camada anterior, para que esta nova camada seja gerada com apenas um nó. E, para haver um mínimo de diversidade em combinações no modelo (e não apenas introduzir maiores ordens de não linearidade), 3 nós são necessários em camadas anteriores, com o ideal sendo um número igual ou superior a 4. Este ponto faz com que seja necessária uma atenção neste filtro selecionado para avaliar a função de erro, a fim que este não seja severo demais, eliminando nós demais e prejudicando o espaço amostral ou até mesmo interrompendo o treinamento do modelo precocemente, antes do desempenho ideal ser atingida.

Por isto, um critério misto torna-se mais atraente para a avaliação de desempenho dos modelos parciais. Por exemplo, permitir que uma porcentagem dos nós com menor MSE formem a próxima camada.

Não só deverá ser usada uma métrica de avaliação de modelos parciais, mas também é necessário uma métrica de avaliação do modelo global. Muitas vezes, pode não ser interessante adicionar camadas demais, com um desempenho satisfatória sendo atingida com menos camadas e a evolução da redução do MSE fique estagnada. Por isto, avaliar o modelo parcial sob esta métrica global torna-se necessário, para que caso ela seja atingida, ou a evolução do treinamento fique estagnada, este seja interrompido automaticamente.

7.3 VARIAÇÕES NO MODELO CLÁSSICO

Como o modelo inicial proposto em (IVAKHNENKO, 1971) possuía uma estrutura feedforward e limitado apenas a ter como função de ativação de todos os seus nós uma forma única da Equação (7.1), mudanças no GMDH clássico foram propostas a fim de expandir o grande potencial de sua estrutura auto organizada, especialmente buscando formas novas de conectar as camadas a fim de aumentar a diversidade de combinações do modelo.

Por exemplo, a fim de otimizar a estrutura da rede, pode-se utilizar funções de ativação de diferentes ordens de não linearidade conforme a rede for ganhando profundidade. Isto é interessante inicialmente a fim de reduzir o número de coeficientes em camadas posteriores do modelo, visto que nem sempre um polinômio cúbico traria melhorias significativas ao modelo, comparado com uma função linear. Por exemplo, pode ser proposto que sejam avaliadas para cada nó as seguintes funções de ativação, sendo estas variações da Equação (7.1) para a ordem:

- **Linear**: $y_L = a_0 + a_1 x_i + a_2 x_j$
- **Linear com Covariância**: $y_{LC} = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j$
- **Quadrático**: $y_Q = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2$
- **Cúbico**: $y_C = y_Q + a_6 x_i^3 + a_7 x_j^3 + a_8 x_i^2 x_j + a_9 x_i x_j^2$

É importante ressaltar que as funções de ativação não precisam ser restritas a funções polinomiais. Funções de outras naturezas (como aquelas utilizadas em ANNs) poderiam ser utilizadas, mas haverá a necessidade de adaptar seu algoritmo de treinamento, para que este se adéque as particularidades da estimação de coeficientes de cada função.

Para cada nó, pode ser avaliado o desempenho de cada uma destas funções, a fim de que se uma função com menos coeficientes trouxer um desempenho muito similar a uma função de mais coeficientes, possa ser escolhida aquela que reduzirá o tamanho do modelo. Mas, também pode ser possível escolher múltiplos nós para a combinação de duas entradas

específicas, cada um com uma função de ativação diferente. Se o processo de treinamento apontar este cenário como benéfico para a acurácia global do modelo, ele deverá ser considerado.

Uma possibilidade que ocorre no modelo clássico, durante o processo de avaliação dos modelos parciais, foi de na primeira camada todos os nós que continham uma entrada específica serem descartados pela métrica de seleção estabelecida. Porém, descartar completamente uma entrada logo na primeira camada pode ser maléfico para alguns conjuntos de dados, sendo interessante tentar mantê-la para avaliação de desempenho em camadas posteriores, a fim de garantir a real contribuição desta. Por isto, torna-se interessante permitir que o algoritmo considere sempre que uma nova camada for gerada as entradas não utilizadas no começo, além dos resultados da camada anterior.

7.4 GMDH BASEADO EM VALORES COMPLEXOS

Originalmente, o GMDH foi planejado apenas para realizar a regressão e a classificação de sistemas baseados em dados reais. Este modelo clássico, referenciado como Método do Grupo de Manipulação de Dados Real (RGMDH), significando que o método suporta apenas valores reais. Porém, como descrito em capítulos anteriores, o problema da modelagem comportamental de um RFPA envolve a identificação e a modelagem de um sistema composto por entradas e saídas do tipo complexo. Para modelos polinomiais, trabalhar com dados complexos não exigiu grandes modificações, sendo apenas necessário adaptações no MMQ utilizado em seu treinamento. Para sistemas com dados complexos, é intuitivo trabalhar com polinômios e técnicas prontas para estes, não sendo necessárias adaptações, já sendo este um processo muito bem estabelecido na literatura. Já para ANNs, as adaptações são mais severas, não apenas englobando o processo de treinamento, mas também na definição das funções de ativação do modelo, resultando em uma tarefa nada simples.

Porém, foram vistos significativos ganhos de desempenho e redução computacional quando utilizados modelos que aceitam valores complexos, embora sua implementação torne-se difícil e com grandes limitações a serem consideradas no projeto, como limites das zonas de operação de cada função de ativação, afetando a efetividade do treinamento e limitando as opções do projetista.

Portanto, torna-se vantajoso estudar as contribuições do GMDH quando este processe valores complexos. Esta é uma área hoje pouco explorada, com apenas trabalhos na área de classificação, limitado a um sistema com entrada e saída real, com o processamento nas camadas ocultas feito no domínio complexo (XIAO et al., 2020), seguindo a lógica anteriormente vista para ANNs de valores complexos. Por isto, é de grande interesse estudar e entender possíveis benefícios de um Método do Grupo de Manipulação de Dados Complexos (CGMDH).

Adicionando ao problema, pouco foi estudado sobre variações nas funções de ativação a serem utilizadas com o CGMDH. Mais precisamente, como já visto, ao modelar um sistema

complexo e, mais especificamente, um RFPA com funções polinomiais, foi provado ser benéfico fazer alterações na Série de Volterra utilizada, a fim de adequar às peculiaridades matemáticas deste sistema. O processo de treinamento, assim como a função de erro utilizada também exigirão um novo olhar, sendo comentado ao decorrer desta seção.

7.4.1 Funções de ativações para modelagem de um RFPA com o CGMDH

Como comentado para o RGMDH, as funções de ativações a serem utilizados em cada nó são variações da Série de Volterra, o chamado polinômio de Kolmogorov-Gabor anteriormente visto na Equação (7.1). Embora estas sejam boas funções generalistas, os avanços no uso de Série de Volterra na modelagem de RFPAs podem ser implementados na função de ativação de um CGMDH, visto que haverá similaridades em sua natureza.

Ao analisar a função de ativação a ser utilizada neste modelo, pode-se utilizar as abordagens já consolidadas para a modelagem comportamental de RFPAs, desenvolvidas em (LIMA, 2009). Para isto, esta subseção analisará um nó singular, utilizando o que foi discutido na Seção 5, ao invés do modelo total com múltiplas camadas.

Como a memória do modelo poderá ser implementada a partir das entradas (ou seja, um vetor de entrada com n e seus $n - M$ instantes passados), pode-se utilizar as derivações da Série de Volterra sem memória como função de ativação do CGMDH. Além disto, como este trabalha analisa especificamente a modelagem comportamental de um RFPA, pode-se também implementar algumas das melhorias propostas na Série de Volterra sem memória para este caso específico.

De acordo com (BENEDETTO; BIGLIERI; DAFFARA, 1979), apenas termos ímpares contribuem para o desempenho do modelo, quando utilizado uma Série de Volterra em Passa Baixa para modelar um RFPA. Isto ocorre pois quaisquer outros termos estarão fora da região do filtro aplicado. Além disso, o número de entradas conjugadas complexas na entrada do modelo será sempre um a menos do número de entradas não conjugadas na série. Com isto, é proposto como função de ativação:

$$\begin{aligned} y(x) =& a_0x_1 + a_1x_2 + a_2x_1x_1x_1* + a_3x_1x_1x_2* + \\ & a_4x_1x_2x_1* + a_5x_1x_2x_2* + a_6x_2x_2x_1* + a_7x_2x_2x_2*, \end{aligned} \tag{7.8}$$

onde x_1* e x_2* são os conjugados de x_1 e x_2, respectivamente. A Equação (7.8) é nada mais que uma Série de Volterra sem memória, com ordem de truncamento igual a 3.

Como comentado, os termos de segunda ordem são descartados pois não contribuirão à acurácia do modelo. Além disso, não são inclusos termos acima da terceira ordem na função de ativação, pois a ideia da técnica do GMDH é a construção de modelos parciais. Por isto, ordens superiores de não linearidade poderão ocorrer conforme novas camadas forem adicionadas,

sendo que os nós que introduzirem estas ordens superiores apenas serão utilizados caso estes contribuam significativamente com o desempenho, senão serão descartados, a fim de reduzir a complexidade computacional. Além disso, o número de termos conjugados de cada coeficiente sempre será um menor do que o número de termos não conjugados.

7.4.2 Treinamento do CGMDH para Identificação e Regressão de um Sistema

Esta subseção entrará em detalhes no processo utilizado para a determinação dos coeficientes do modelo CGMDH, expondo suas peculiaridadess e adaptações em relação ao modelo baseado em dados reais.

7.4.2.1 Treinamento e estimação dos coeficientes com um MMQ Complexo

Quando era trabalhado apenas com dados reais no RGMDH, os coeficientes do modelo eram facilmente calculados utilizando o MMQ, visto na Equação (7.5), em que X e Y eram as matrizes de entrada e alvo, respectivamente, compostas apenas de dados reais, com S sendo o vetor de coeficientes reais, X^T e X^{-1} as matrizes transposta e inversa de X, respectivamente.

Para o CGMDH, X e Y tornarão-se matrizes que conterão os N valores complexos de entrada $\tilde{x}_n$ e de alvo $\tilde{y}_n$, respectivamente. O objetivo será calcular a matriz de coeficientes complexos C. Este problema se dará da seguinte forma:

$$\tilde{y} = \tilde{c_0} + \sum_{i=1}^{n} \tilde{c_i} f(\tilde{x_i}, \tilde{x_j}) = \tilde{c_0} + \sum_{i=1}^{n} \tilde{c_i} \tilde{z_i} \tag{7.9}$$

em que $C = [\tilde{c_0}, \tilde{c_1}, ..., \tilde{c_n}]$ será a matriz de coeficientes complexos a serem calculados durante o treinamento. A função $\tilde{z_i} = f(\tilde{x_i}, \tilde{x_j})$ nada mais é do que a função complexa que representa a interação entre as duas entradas de cada nó. Esta função torna-se necessária visto que haverá termos não lineares ocorrendo na função de ativação vista na Equação (7.8), sendo preciso substituir os termos não lineares por uma variável linear, permitindo então o uso do MMQ para calcular os coeficientes.

O próximo passo será adaptar a Equação (7.5) a fim de suportar valores complexos. Mais especificamente, a matriz transposta X^T torna-se o centro das atenções. Para dados complexos, ao invés de realizar o cálculo da matriz transposta, é necessário realizar o cálculo do conjugado transposto da matriz X de entrada, também chamado de matriz hermitiana e representada por X^H ou X^*. Considerando que $\overline{X}$ será o conjugado da matriz complexa X, a matriz hermitiana de X será calculada por:

$$X^H = (\overline{X})^T = \overline{X^T} \tag{7.10}$$

Com isto, o MMQ complexo será calculado por:

$$C = (X^H X)^{-1} X^H Y \tag{7.11}$$

7.4.2.2 Cálculo do Erro

O uso do MSE para cálculo do erro com valores complexos não torna-se algo direto. Se não houver alterações no formato do algoritmo, é comum o gradiente de erro acompanhar apenas valores de magnitude, ignorando a evolução do erro da fase.

7.4.2.3 Erro Logarítmico

Para evitar estes problemas, que serão críticos para o treinamento deste modelo, algumas soluções podem ser propostas. Em (SURESH; SUNDARARAJAN; SAVITHA, 2013), é proposta uma nova função de erro que incluirá tanto os valores de módulo como de fase do modelo em seu cálculo. Esta é definida na forma

$$E_{new} = \frac{1}{2}[\log(\frac{y}{y_{ref}}) \log(\overline{\frac{y}{y_{ref}}})] \tag{7.12}$$

em que $\log(\frac{y}{y_{ref}})*$ será o Conjugado Cúbico de $\log(\frac{y}{y_{ref}})$. Esta função de erro logarítmica também pode ser manipulada para deixar explícito os valores de módulo e fase. Se $y = Ae^{j\theta}$ e $y_{ref} = A_{ref}e^{j\theta_{ref}}$, então a Equação (7.12) torna-se

$$E_{new} = \frac{1}{2}[\log(\frac{A}{A_{ref}})^2 + (\theta - \theta_{ref})^2] \tag{7.13}$$

sendo as Equações (7.12) e (7.13) equivalentes. Desta forma, o erro estará claramente mensurando tanto componentes de módulo como fase. Este formato da Equação (7.13) também facilita a adição de pesos para cada componente,

$$E_{new} = \frac{1}{2}[k_1 \log(\frac{A}{A_{ref}})^2 + k_2(\theta - \theta_{ref})^2] \tag{7.14}$$

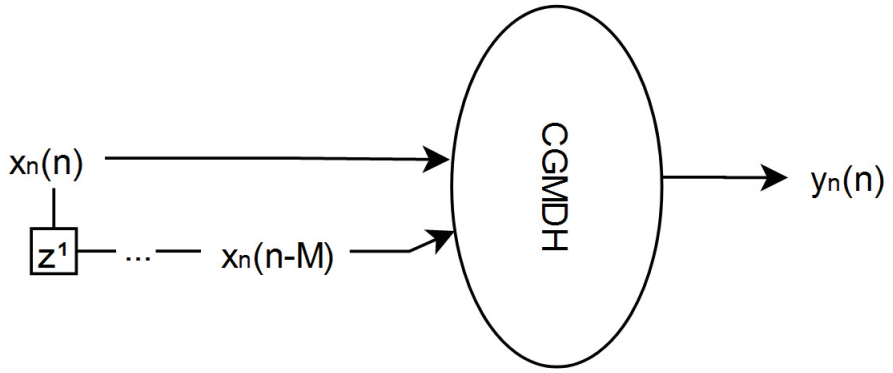
o que poderá facilitar caso o processo de treinamento tenda a convergir mais rapidamente de acordo com o fator de cada componente.

7.4.3 Estrutura do modelo CGMDH para a modelagem de um RFPA

Por fim, será revista a forma com que ficará a estrutura do CGMDH, durante a modelagem de um RFPA. Agora, como é possível trabalhar diretamente com dados complexos, estimando tanto o módulo como a fase de saída dos dados em um único valor, não torna-se mais necessário utilizar uma topologia de rede vista na Seção 6.4 e utilizadas tanto para ANNs reais como para o RGMDH.

A forma com que a topologia de um CGMDH será pode ser vista na Figura 23. Seu grande diferencial contra topologias a serem utilizadas em modelos com dados reais é a necessidade de se utilizar apenas uma rede para a modelagem. Isto significa reduzir em cerca de metade os coeficientes necessários do modelo, sem haver a princípio impactos na acurácia da modelagem.

FIGURA 23 – Topologia a ser utilizada para um CGMDH utilizado na modelagem de um RFPA.



FONTE: O autor (2020)

De fato, mais do que impactos na acurácia, um dos motivos mais interessantes do uso do CGMDH é o fato do modelo ser menor: possuir menos coeficientes a serem estimados, menos números a serem guardados em memória, e a princípio sem necessitar descartar qualquer informação. Validar seu desempenho contra o RGMDH, em um cenário em que o número de coeficientes seja similar, trará resultados interessantes. Estes vão ser vistos nos próximos capítulos deste trabalho.

8 RESULTADOS DE SIMULAÇÃO COM A MODELAGEM DE RFPAS UTILIZANDO RGMDH E ANNS DE VALORES REAIS

Este capítulo irá descrever a metodologia utilizada para a definição e construção dos modelos de redes utilizados para obter os resultados de modelagem comportamental de um RFPA, tanto do tipo ANN como do tipo RGMDH, focando nos passos do ajuste do treinamento e de peculiaridades de cada um destes para a modelagem com dados do tipo real.

É importante ressaltar que para esta dissertação, todos os resultados obtidos neste capítulo e nos seguintes foram a partir de simulações computacionais, e nenhum modelo foi utilizado em aplicações reais.

Para todas as simulações, foram feitos algoritmos em Python, na versão 3.8.3. As ANNs MLP foram montadas utilizando a biblioteca Tensorflow, enquanto a rede GMDH foi obtida através de modificações do autor no algoritmo aberto GmdhPy.

Os dados utilizados para a modelagem neste capítulo e durante os próximos foram obtidos através de um analisador de sinal vetorial Rohde & Schwarz, que realizou a medição de entrada e saída de forma discreta no tempo, com uma amostragem de 61,44 MHz, de um RFPA GaN HEMT classe AB, estimulado por um sinal WCDMA com duas portadoras, com largura de banda de 8,84 MHz e centrado em 900 MHz, com a potência de saída média de 26 dBm. Os dados utilizados estão normalizados.

Das 33.780 amostras obtidas deste RFPA, 24.180 (71,5%) foram utilizadas para o treinamento dos modelos, enquanto 9.600 amostras (28,5 %) foram utilizadas para validação da modelagem.

8.1 MÉTRICA DE AVALIAÇÃO DOS MODELOS: NMSE

Uma forma comumente utilizada para validar a acurácia de modelos matemáticos é o Erro Quadrático Médio Normalizado (NMSE). A expressão utilizada por ele é a que segue:

$$NMSE = 10 * log_{10} \frac{\sum_{n=1}^{N} |e(n)|^2}{\sum_{n=1}^{N} |yref(n)|^2}, \tag{8.1}$$

Nesta Equação (8.1), a diferença entre valor previsto e valor desejado é definido por $e(n) = y_{ref}(n) - y_{test}(n)$, sendo $y_{ref}(n)$ o valor desejado na saída, representado pelos dados de validação do modelo e $y_{test}(n)$ a saída medida no instante n. O valor de N será o número total de amostras do modelo. Quanto menor for o valor do NMSE do modelo, mais preciso será o modelo na previsão de dados.

8.2 DADOS DE ENTRADA, ALVOS E TOPOLOGIA DAS REDES COM VALORES REAIS

O primeiro passo na modelagem de dados complexos utilizando modelos que processam apenas dados reais é o ajuste deste conjunto de dados, visto que diferentes formas já foram propostas, e isto afetará o desempenho geral do treinamento.

Inicialmente, os dados utilizados neste trabalho consistem no sinal de envoltória complexa da entrada e saída de um RFPA. Por isto, torna-se necessário convertê-los para valores puramente reais, mas sem perder nenhuma informação importante contida na representação complexa.

Um sinal de envoltória complexo $\tilde{x}_n$ no instante de tempo n aplicado na entrada de RPFA poderá ser decomposto em amplitude a_n e fase θ_n , da forma:

$$\tilde{x}_n = a_n exp(j\theta_n), \tag{8.2}$$

com sua parte real sendo representada por $R_{x_n} = a_n cos(\theta_n)$ e parte imaginária representada por $I_{x_n} = a_n sin(\theta_n)$.

De maneira similar, a envoltória complexa medida na saída de um RFPA, ao ser excitado pelo sinal representado pela Equação (8.2), pode ser descrita com:

$$\tilde{y}_n = b_n exp(j\alpha_n), \tag{8.3}$$

com b_n sendo o módulo do sinal e α_n sendo a fase da saída, com a parte real representada por $R_{y_n} = b_n cos(\alpha_n)$ e parte imaginária representada por $I_{y_n} = b_n sin(\alpha_n)$.

Como descrito no Capítulo 3, a fim de otimizar a eficiência energética do RFPA, este deverá operar em sua região não linear. Isto significa introduzir não linearidades na relação entre a envoltória de entrada $\tilde{x}_n$ e saída $\tilde{y}_n$, sejam eles na forma de uma relação não linear entre os módulos a_n e b_n (distorção AM-AM), ou com os valores instantâneos do módulo de entrada a_n resultarem em uma fase α_n diferente de θ_n (distorção AM-PM).

Por isto, torna-se crítico que a amplitude e fase da saída dependerão dos valores instantâneos da entrada. Além disto, foi comentado já neste trabalho que RFPAs operando na região não linear também apresentam efeitos de memória, ou seja, valores passados influenciarão nos valores instantâneos. Por isto, torna-se necessário ter como entrada não apenas o módulo a_n no instante n, mas também seus M valores passados. Com isto, pode-se ter um vetor de módulo de entrada na forma:

$$A = [a_n, a_{n-1}, ..., a_{n-M}] \tag{8.4}$$

com M sendo a ordem de memória a ser escolhida. Também foi visto em trabalhos passados

que para uma boa modelagem dos efeitos PM-AM e PM-PM do RFPA (FREIRE; FRANCA; LIMA, 2015), é interessante usar o seno e o cosseno da diferença de fase de dois instantes de tempo consecutivos, assim como seus consequentes valores passados para representar a memória da fase. Com isto, pode-se obter mais dois vetores de entrada do modelo:

$$P_{sin} = [sin(\theta_n - \theta_{n-1}), sin(\theta_{n-1} - \theta_{n-2}), ..., sin(\theta_{n-M+1} - \theta_{n-M})] \tag{8.5}$$

$$P_{cos} = [cos(\theta_n - \theta_{n-1}), cos(\theta_{n-1} - \theta_{n-2}), ..., cos(\theta_{n-M+1} - \theta_{n-M})] \tag{8.6}$$

Concluindo, para um modelo que terá como entradas valores complexos, mas que processará apenas valores reais, pode-se decompor a envoltória complexa de entrada em 3 vetores: A, P_{sin} e P_{cos}, com o tamanho destes vetores sendo determinado pela memória M e pela quantidade de dados obtidos para a modelagem.

A forma com que os alvos serão representados será totalmente influenciada pela topologia da rede escolhida. Retomando o que foi comentado no Capítulo 6, esta topologia é essencial ao modelar um sistema baseado em valores complexos, utilizando modelos que apenas aceitam valores reais. Estas topologias foram extensamente comentadas na Seção 6.4, mostrando a evolução em acurácia e treinamento de cada uma.

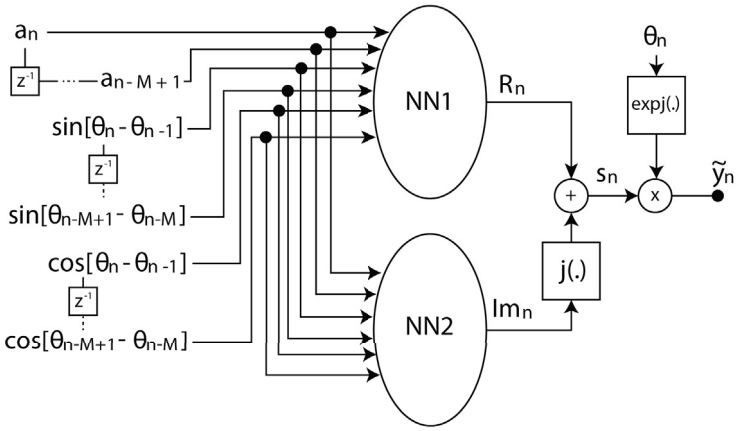
O modelo escolhido para a modelagem com redes de valores reais neste trabalho pode ser visto na Figura 24. Aqui, duas redes completamente separadas serão utilizadas, em que uma será responsável por estimar a parte real, enquanto a outra estimará a parte imaginária da saída subtraída da fase de entrada no instante n. As entradas serão os valores absolutos do sinal de envoltória, seus valores nos instantes de tempo anteriores *n-m*, com $m = 1, 2, ...M$, assim como o seno e cosseno da diferença de fase do sinal de envoltória no instante atual n e instante passado $n - 1$, assim como uma segunda ordem de memória com a diferença de fase entre os instantes $n - 1$ e $n - 2$.

Como a fase da entrada não é representada diretamente, mas sim em seno e cosseno da diferença de fase entre dois momentos consecutivos, os alvos da rede não serão diretamente a fase α_n do sinal de saída $\tilde{y}_n$, mas sim:

- Alvo da Parte Real: $b_n cos(\alpha_n - \theta_n)$
- Alvo da Parte Imaginária: $b_n sin(\alpha_n - \theta_n)$

Para recompor a saída $\tilde{y}_n$, será necessário somar o valor da fase da entrada θ_n no mesmo instante de tempo, conforme visto no diagrama de blocos da saída das redes da Figura 24.

FIGURA 24 – Diagrama de blocos de uma Rede Neural para Modelagem de um RFPA.



FONTE: O autor (2020), adaptado de (FREIRE; FRANCA; LIMA, 2015)

Agora que a definição das entradas foi feita, dos alvos e da topologia a ser utilizada nesta seção, é possível avançar para o ajuste dos modelos para obter os melhores treinamentos e consequentemente a representação com melhor acurácia do RFPA, sempre levando em conta a complexidade computacional.

8.3 TREINAMENTO DO MODELO GMDH COM DADOS REAIS PARA A MODELAGEM DE UM RFPA

A ideia de comparação neste trabalho será o desempenho para modelos com complexidade computacional similar. Isto será considerado da seguinte forma: modelos que tenham um número muito próximo de coeficientes serão considerados de complexidade computacional similar, visto que consumirão uma quantidade próxima de memória de armazenamento do HW em que serão implementados. Além disso, pode-se estimar indiretamente o número de operações necessários para a previsão de dados baseado no número de coeficientes, por isto este torna-se um parâmetro simples de equiparação.

Como o GMDH é um modelo auto organizável, não é possível a priori saber o seu número exato de coeficientes (igual ocorre em modelos de ANNs). Ou seja, é necessário executar todo seu treinamento primeiramente para saber qual o tamanho do modelo final. Por isto, será iniciado o treinamento dos modelos com o GMDH em vez das ANNs. Utilizando a topologia de rede proposta na Figura 24, e uma memória $M = 1$, resta agora definir as funções de ativação dos nós do GMDH.

Conforme comentado no Capítulo 7, as funções de ativação originalmente propostas para o modelo eram do tipo polinomial, mais especificamente o polinômio de Kolmogorov-Gabor e suas derivações visto na Equação (7.1). Como este permite que manipulemos a ordem para obter funções com números diferentes de coeficientes, serão escolhidos alguns valores diferentes de ordem para serem as funções de ativação. Estes serão:

- **Linear**: $y_L = a_0 + a_1 x_i + a_2 x_j$
- **Linear com Covariância**: $y_{LC} = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j$
- **Quadrático**: $y_Q = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2$
- **Cúbico**: $y_C = y_Q + a_6 x_i^3 + a_7 x_j^3 + a_8 x_i^2 x_j + a_9 x_i x_j^2$

Como já havia sido comentado, o modelo terá múltiplas funções de ativação e, para cada combinação de duas entradas, serão gerados nós com cada uma destas funções, a fim de validar a contribuição de cada no modelo parcial. Ou seja, existe a possibilidade de que duas entradas gerem dois nós, cada um com uma função de ativação diferente. Além disto, a vantagem destas funções de ordens diferentes é vista conforme o modelo ganha profundidade. Já que a maior ordem de uma função de ativação escolhida neste capítulo é de terceira, combinações ao longo das camadas vão gerar ordens superiores para o modelo. Por isto, camadas posteriores poderiam ter não linearidades desnecessárias, que gerariam um *overfitting* ao modelo. Por isto, permitir que o modelo possa escolher funções de ordem inferior (como a Linear e a Linear com Covariância) acaba fazendo com que as camadas finais sejam apenas uma combinação dos múltiplos resultados obtidos nos modelos parciais para então gerar a saída final, evitando introduzir operações desnecessárias.

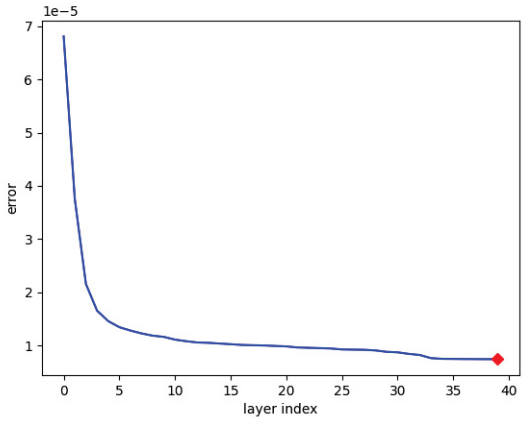
8.3.1 RGMDH Sem restrições de camadas

A primeira versão do modelo gerada será chamada de RGMDH sem Restrições, concebido deixando seu treinamento sem restrições no número máximo de camadas. Deseja-se mais do que tudo entender como a redução do erro do modelo evolui conforme novas camadas são adicionadas, a fim de encontrar o ponto ideal de número de camadas para melhor desempenho. Portanto, não foram colocadas restrições em largura da rede (número de nós máximo por camada) nem profundidade (número de camadas). Apenas foi determinado o critério de parada $\epsilon = 0,001$, ou seja, caso a diferença do erro relativo da camada atual em relação ao menor erro desta for menor que 0,001, o treinamento poderá ser finalizado. Enquanto não atingir esta métrica, a rede continuará adicionando camadas. Vale relembrar que a métrica de erro utilizada no RGMDH é o MSE.

As Figuras 25 e 26 representam a evolução do erro MSE por camada da rede, conforme novas camadas foram acrescentadas. Como pode-se ver, a rede responsável pela parte real chegou a um total de 40 camadas, enquanto a rede responsável pela parte imaginária teve 27 camadas totais. Porém, pode ser visto que para esta primeira rede, o erro teve uma grande evolução em sua redução até a 5ª camada, seguido de uma evolução menor até a 10ª camada e pouquíssima melhora, na casa de 10^{-5}, após a 10ª camada, enquanto a segunda rede teve uma evolução similar, com grandes resultados até a 5º camada, e pouca evolução após a 10ª, na ordem $2,510^{-4}$. Ou seja, muitas camadas foram adicionadas após a 10ª, gerando um aumento

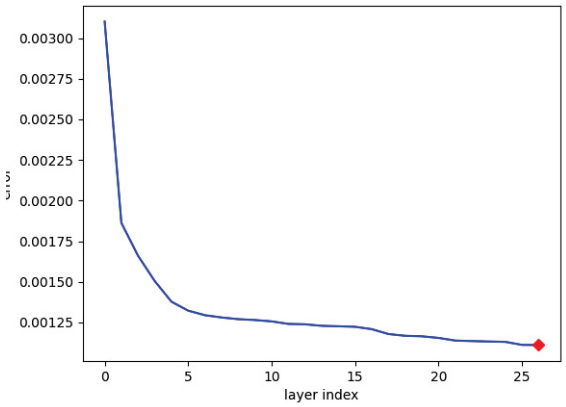
muito significativo no número de coeficientes, porém contribuindo muito pouco na redução do erro do modelo. O NMSE total deste modelo foi de -47,17 dB .

FIGURA 25 – Evolução do erro por camada para o treinamento do RGMDH Sem Restrições para a Rede que irá prever os valores reais da saída.



FONTE: O autor (2020)

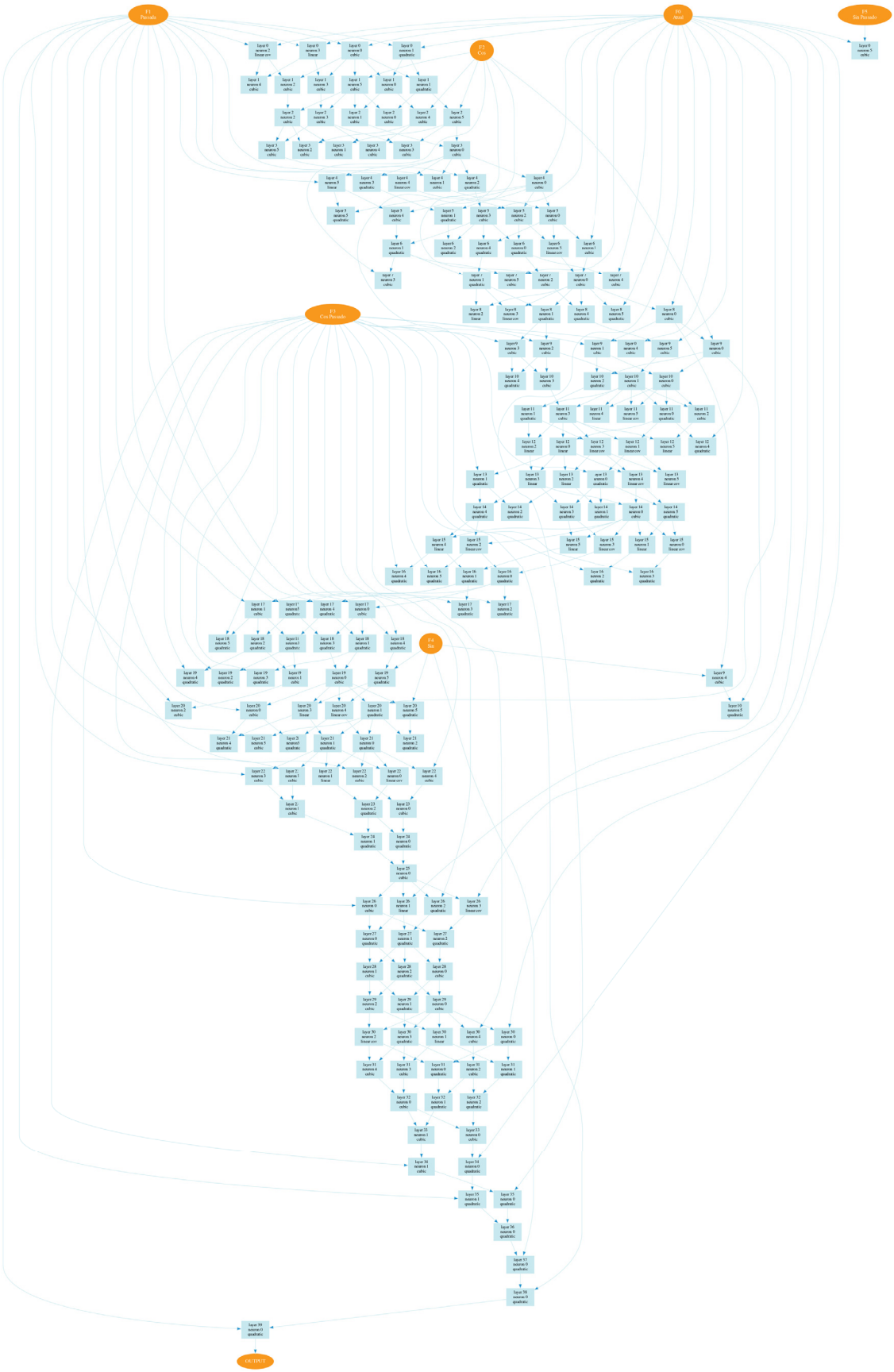
FIGURA 26 – Evolução do erro por camada para o treinamento do RGMDH Sem Restrições para a Rede que irá prever os valores imaginários da saída.



FONTE: O autor (2020)

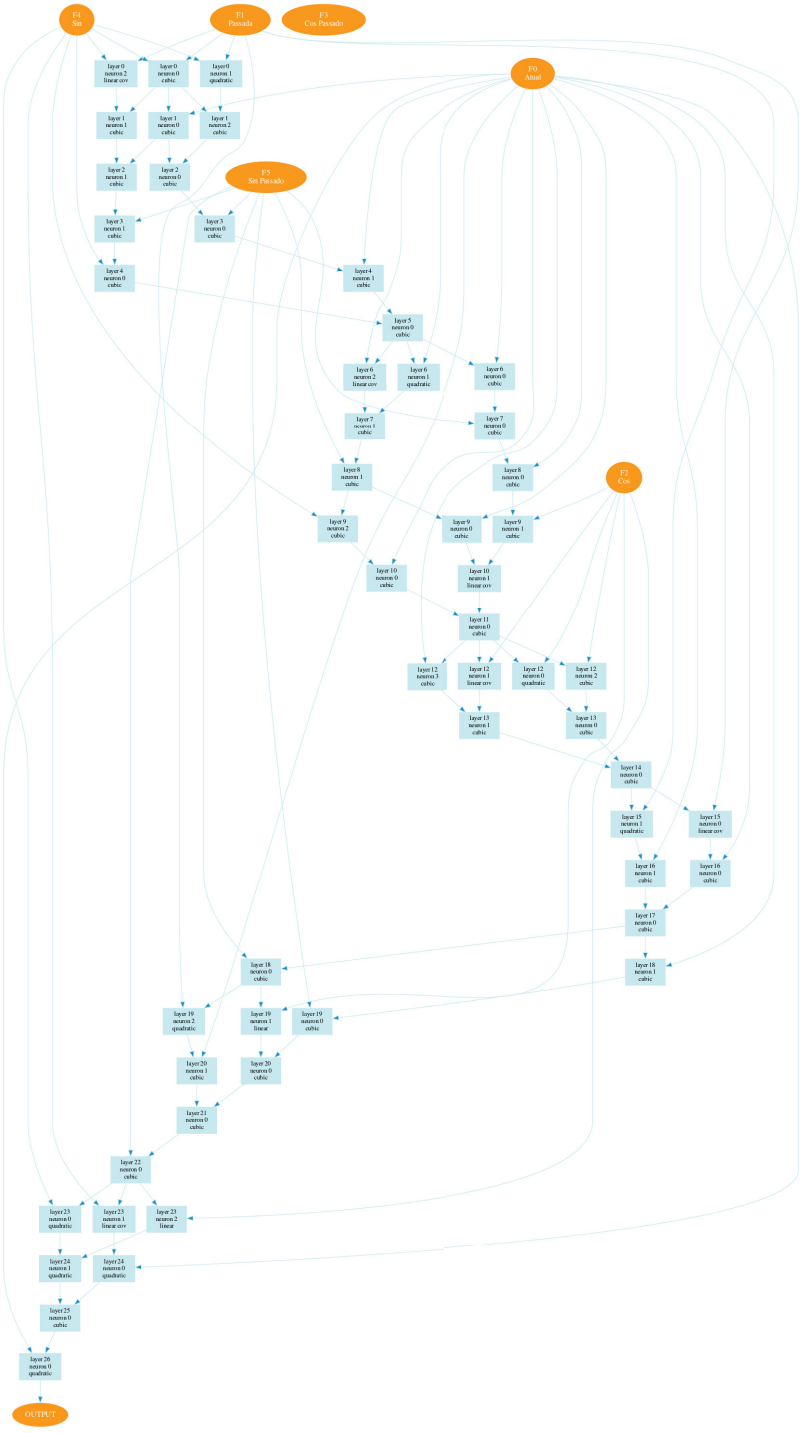
Este número elevado de camadas também significa que o número de nós e, consequentemente, o número de coeficientes destas duas redes foi extremamente alto. Sem entrar em grandes detalhes no número total, a forma com que o treinamento organizou a estrutura de ambas pode ser visto nas Figuras 27 e 28. Conforme novas camadas foram adicionadas, o número de nós cresceu, principalmente para a rede que estimará a parte real, que chegou a 40 camadas. Por mais válido que seja, os resultados do RGMDH Sem Restrições não são interessantes para nosso intuito de implementá-lo em HWs, onde haverá limites de processamento e armazenamento.

FIGURA 27 – Arquitetura da Rede para estimar parte real da saída gerada durante o treinamento do RGMDH Sem Restrições.



FONTE: O autor (2020)

FIGURA 28 – Arquitetura da Rede para estimar parte imaginária da saída gerada durante o treinamento do RGMDH Sem Restrições.



FONTE: O autor (2020)

É importante também notar que, por mais que as Figuras 27 e 28 já contenham um elevado número de nós, o treinamento destas envolveu muito mais, já que nós de cada camada que não contribuíram significativamente para os modelos parciais foram eliminados no processo de auto organização. Mais especificamente, a rede real possui 177 nós, enquanto a

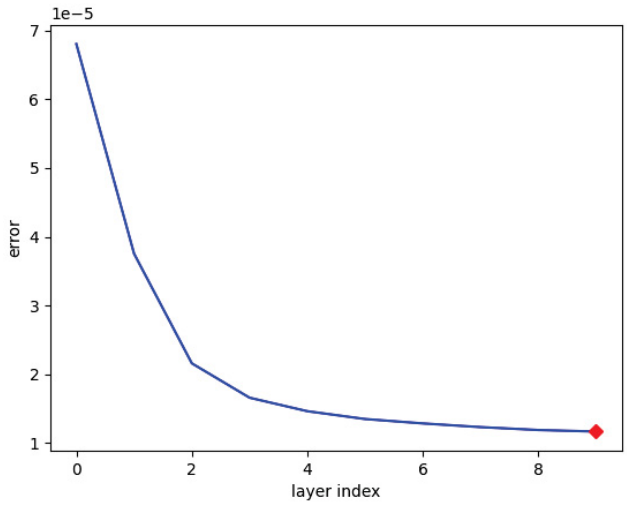
rede responsável pela parte imaginária possui 54 nós. Ou seja, esta primeira versão do GMDH sem nenhuma restrição totalizou 231 nós escolhidos, um número bem elevado, ainda mais caso tivesse que ser estimado o total de coeficientes do modelo (cada nó tem no mínimo 3 coeficientes a serem estimados, caso este tenha como função de ativação a função linear).

8.3.2 RGMDH com limites de camadas

Uma primeira forma de simplificar o modelo em seu tamanho é limitando o número máximo de camadas no treinamento. Aqui, as Figuras 25 e 26 tornam-se essenciais. Como havia sido comentado anteriormente, pode-se ver que conforme as 10 primeiras camadas são adicionadas, o erro é minimizado drasticamente para ambas as redes. Porém, após a décima, a evolução da minimização do erro é pequena e pouco significativa. Por isto, o próximo passo será impor uma limitação de 10 camadas para as redes que irão prever parte real e imaginária.

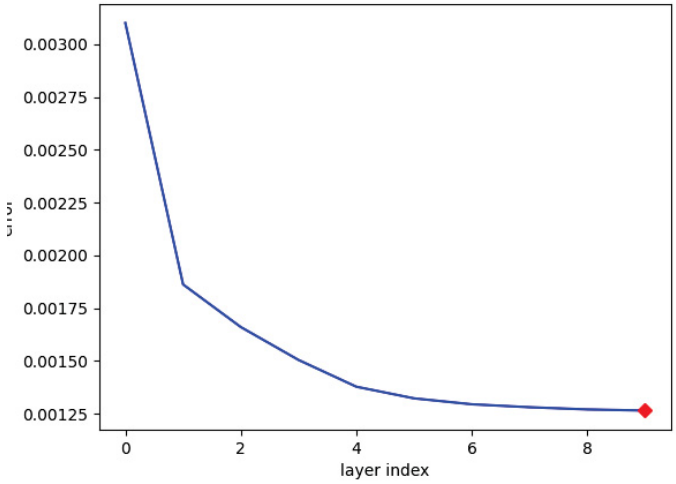
Realizando os treinamentos da mesma forma do RGMDH Sem Restrições, será gerado o modelo RGMDH com 10 camadas. A evolução da redução do erro pode ser vista nas Figuras 29 e 30. O NMSE de validação deste modelo completo foi -49,92 dB. Como pode-se notar nos dois gráficos, novamente existe uma região de platô no erro com a evolução das camadas, mais especificamente após a adição da 5ª camada, significando que poderá haver espaços para reduzir o número de camadas e o número de coeficientes sem haver grandes impactos no NMSE de validação deste modelo.

FIGURA 29 – Evolução do erro por camada para o treinamento do RGMDH com 10 camadas, para a Rede que irá prever os valores reais da saída.



FONTE: O autor (2020)

FIGURA 30 – Evolução do erro por camada para o treinamento do RGMDH com 10 camadas, para a Rede que irá prever os valores imaginários da saída.



FONTE: O autor (2020)

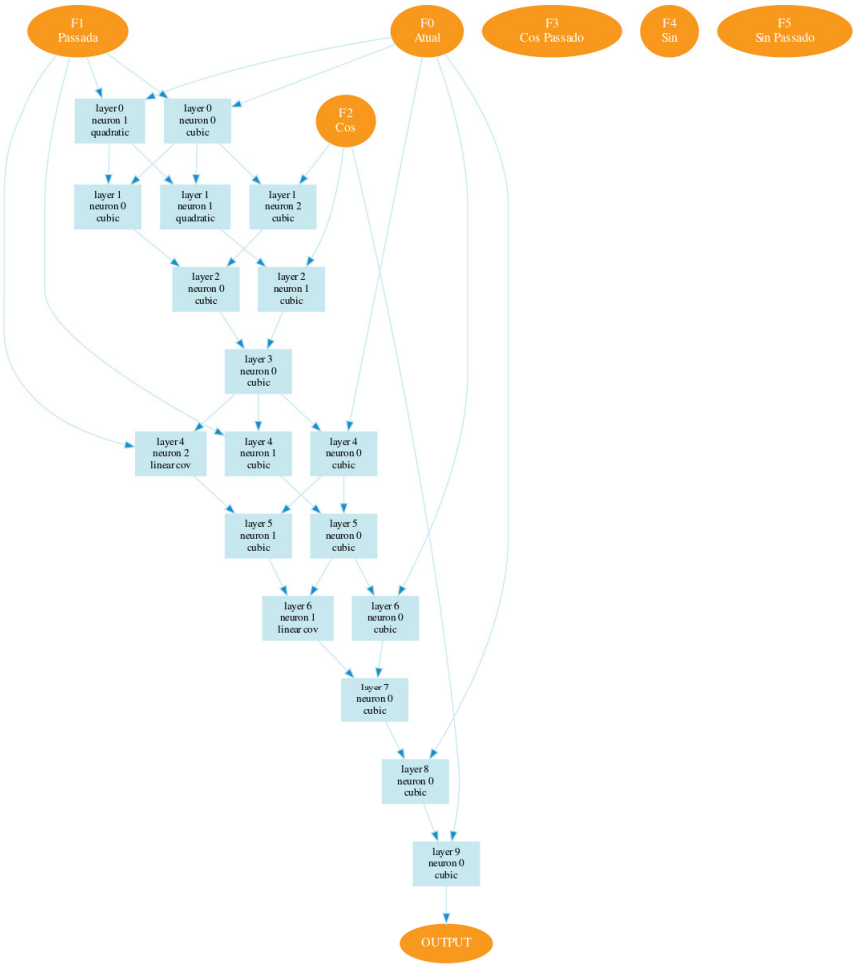
As Figuras 31 e 32 mostram como ficaram os modelos auto organizados gerados durante o treinamento, com a limitação de 10 camadas por rede. Alguns pontos interessantes de cada um desses modelos podem ser notados e comentados. A rede real possui um total de 18 nós, enquanto a rede imaginária totalizou 21 nós, distribuídos ao longo das 10 camadas de cada uma. Já pode ser visto aqui uma boa redução no número de nós nestes modelos, totalizando 39, ou seja, 200 nós a menos comparados com os 239 do RGMDH Sem Restrições. Isto diretamente significará também uma redução drástica no total de coeficientes a serem estimados e posteriormente armazenados.

Além disto, é possível já ver o processo de treinamento completamente descartando entradas que não contribuíram significativamente para a acurácia do modelo final. Para a rede responsável pela parte real, foram utilizadas apenas o módulo instantâneo, o módulo no instante $n-1$ e o cosseno da diferença da fase instantânea e no instante $n-1$. Já para a rede responsável pela parte imaginária, foram utilizados o módulo instantâneo, o módulo no instante $n-1$, o seno da diferença de fase instantânea e fase em $n-1$ e o seno da diferença de fase em $n-1$ e fase em $n-2$, sendo que foram descartados o cosseno da diferença entre fase instantânea e fase em $n-1$, assim como o cosseno da diferença entre a fase em $n-1$ e $n-2$, significando que estes não contribuem significativamente ao desempenho da modelagem dos valores complexos de saída da rede.

Outro ponto interessante que havia sido comentado em capítulos anteriores e pode ser visto na prática é o fato de algumas entradas apenas contribuírem em camadas posteriores à primeira. Isto é visto na Figura 31 com o cosseno da diferença de fase instantânea e $n-1$, assim como na Figura 32 com o módulo instantâneo e o seno da diferença entre a fase em

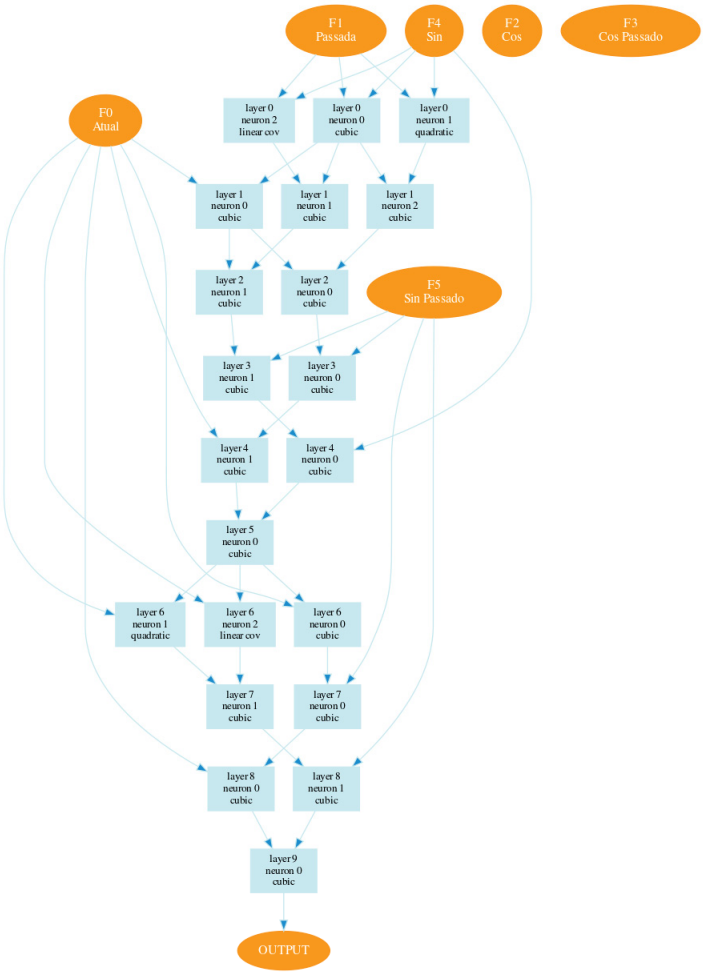
$n-1$ e $n-2$. Se esta característica não fosse inclusa no modelo, ambas estas entradas já teriam sido descartadas logo na primeira camada, e muito provavelmente haveria uma perda significativa de desempenho.

FIGURA 31 – Arquitetura da Rede para estimar parte real da saída gerada durante o treinamento do RGMDH com 10 camadas.



FONTE: O autor (2020)

FIGURA 32 – Arquitetura da Rede para estimar parte imaginária da saída gerada durante o treinamento do RGMDH com 10 camadas.

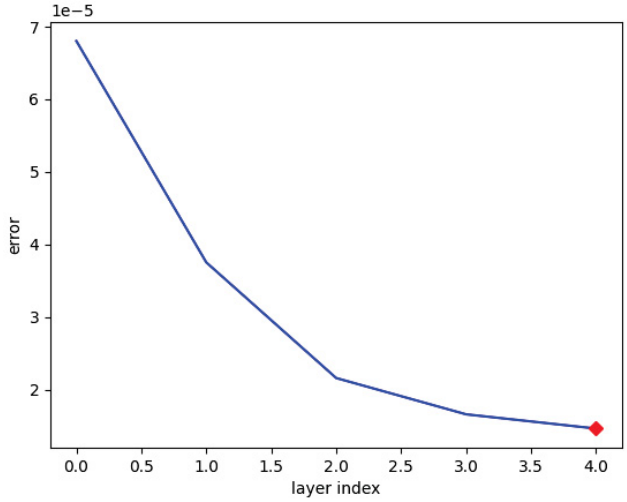


FONTE: O autor (2020)

Em seguida, ira ser tentado reduzir um pouco mais o número de camadas, para tentar chegar na melhor relação entre camadas e NMSE. Para isto, ambos os modelos do RGMDH serão limitados a 5 camadas durante seu treinamento e validar seu NMSE e número total de nós e coeficientes. Ao reduzir para 5 camadas, a evolução da redução do erro por camada pode ser vista nas Figuras 33 e 34. Pode-se ver uma redução significativa conforme novas camadas são adicionadas, sem haver uma clara região em que o erro atinge um platô. Além disto, o modelo também ganhou em acurácia em relação ao GMDH de 10 camadas, muito provavelmente graças a evitar um *overfitting* no treinamento.

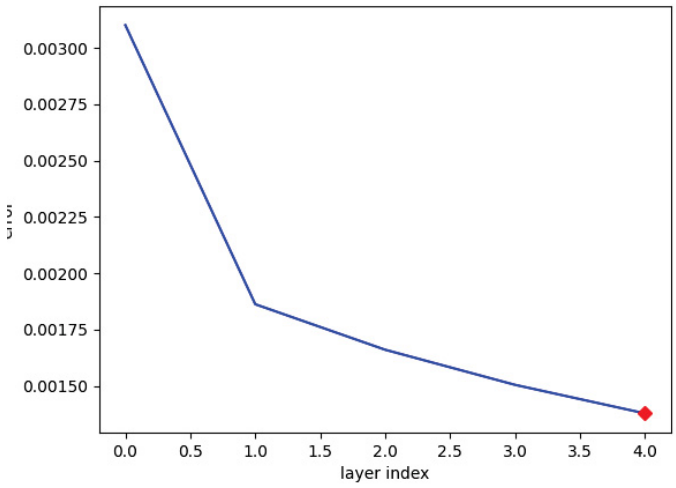
É interessante notar que o NMSE de validação obtido do modelo neste formato foi de -50,17 dB, melhor do que o modelo RGMDH com 10 camadas. Isto ocorreu muito provavelmente ao evitar um *overfitting* no treinamento, que deverá ter ocorrido no RGMDH com 10 camadas e foi evitado no caso reduzido com apenas 5 camadas.

FIGURA 33 – Evolução do erro por camada para o treinamento do RGMDH com 5 camadas, para a Rede que irá prever os valores reais da saída.



FONTE: O autor (2020)

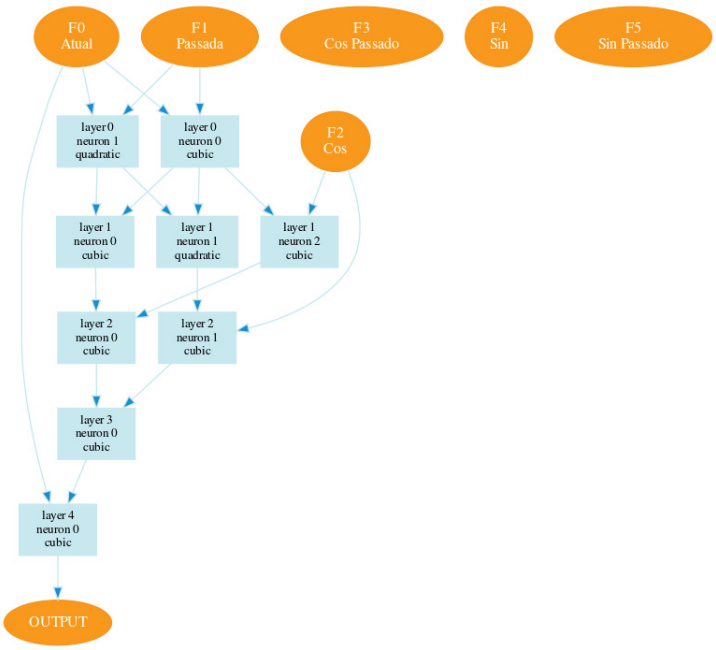
FIGURA 34 – Evolução do erro por camada para o treinamento do RGMDH com 5 camadas, para a Rede que irá prever os valores imaginários da saída.



FONTE: O autor (2020)

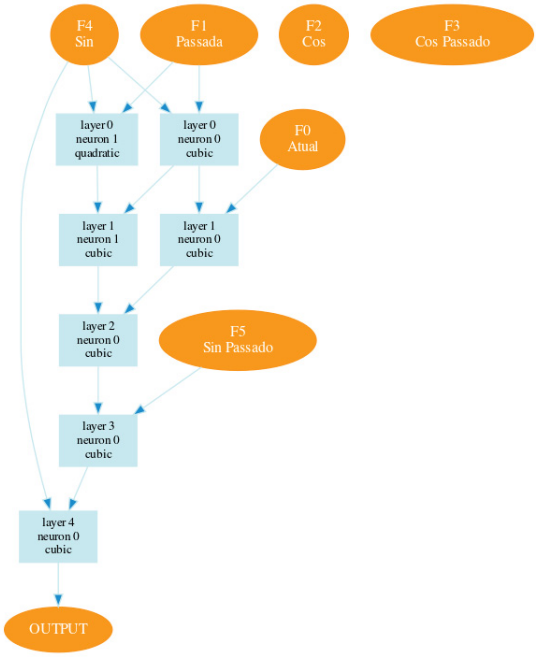
Os modelos auto organizáveis podem ser vistos nas Figuras 35 e 36. O total de nós destes foram de 10 para a rede real e 7 para a rede imaginária, ao longo das 5 camadas de cada, totalizando 17 nós para o modelo final. Novamente é visto tanto o caso de algumas entradas serem descartadas, assim como algumas serem utilizadas em camadas posteriores à primeira.

FIGURA 35 – Arquitetura da Rede para estimar parte real da saída gerada durante o treinamento do RGMDH com 5 camadas.



FONTE: O autor (2020)

FIGURA 36 – Arquitetura da Rede para estimar parte imaginária da saída gerada durante o treinamento do RGMDH com 5 camadas.



FONTE: O autor (2020)

Finalizando esta subseção, a Tabela 2 apresenta os resultados obtidos conforme foi variado os números de camadas do RGMDH. O ponto de maior interesse aqui é que, conforme foi reduzido o número de camadas, o NMSE de validação do modelo diminuiu. Existe uma explicação simples para isto, que torna-se essencial para o treinamento do GMDH com qualquer formato de dados, mas também torna-se verdadeiro para redes neurais. Quando não é imposto um limite de camadas, e o erro de parada não é configurado de acordo, a rede acaba com *overfitting*, ou seja, ela torna-se muito especializada em prever os valores de treinamento, não sendo generalista o suficiente e tendo problemas para prever valores de validação. Ao observar a evolução do erro por número de camadas, é possível determinar empiricamente o número ideal de camadas do modelo para o conjunto de dados, conforme foi feito neste trabalho.

TABELA 2 – NMSE do RGMDH para diferentes números de camadas

RGMDH com variação de camadas	
Total de camadas	***NMSE***
RGMDH Sem limites	-47,17 dB
RGMDH com 10 camadas	-49,92 dB
RGMDH com 5 camadas	-50,17 dB

FONTE: O autor (2020)

8.3.3 Variação do desempenho do RGMDH com diferentes funções de ativação

Agora, torna-se interessante entender o número total de coeficientes presentes neste modelo de 5 camadas, visto que existem diversas funções de ativação nos nós. Apenas o RGMDH de 5 camadas seguirá a ser utilizado, visto que este obteve o melhor NMSE na subseção anterior. A rede real é majoritariamente formada por nós com funções de ativação de terceira ordem, com alguns nós de segunda ordem, totalizando 82 coeficientes a serem estimados. Já a rede imaginária contará também apenas com funções de segunda e terceira ordem, totalizando 66 coeficientes. Ou seja, o modelo RGMDH com 5 camadas terá 148 coeficientes a serem estimados ao longo dos seus 17 nós.

Fixando o número de camadas em 5, tem-se como objetivo entender o impacto das diferentes funções de ativações para o NMSE de validação do modelo. Mudar a função de ativação também terá um efeito direto no total de coeficientes a serem estimados no modelo, visto que a rede sempre terminará seu treinamento em 5 camadas, mas diferentes funções têm diferentes números de coeficientes.

A Tabela 3 mostra qual o NMSE, número de nós e coeficientes obtidos fixando apenas uma função de ativação em cada modelo. É fácil perceber que, conforme ordens são adicionadas, o NMSE melhora. Especialmente falando, a melhora é muito mais drástica indo dos modelos lineares (de 1ª ordem) para os modelos não lineares (a partir de 2ª ordem) como função de

ativação. Porém, pode ser visto que o número de coeficientes é multiplicado aproximadamente por 2 conforme a função de ativação é alterada.

TABELA 3 – NMSE, nós e coeficientes de um RGMDH com 5 camadas, para diferentes funções de ativação

RGMDH com 5 camadas			
Função de Ativação	***nós***	***Coeficientes***	***NMSE***
Linear	8	24	-37,90 dB
Linear com Covariância	12	48	-37,97 dB
Quadrático	16	96	-42,81 dB
Cúbico	19	190	-44,13 dB

FONTE: O autor (2020)

Ainda assim, é notado que um RGMDH puramente composto por funções de ativação de 3ª ordem obteve mais coeficientes e um pior NMSE que o modelo misto obtido na seção anterior (148 coeficientes com NMSE de -50,17 dB). Isto mostra a eficácia do GMDH quando múltiplas funções de ativação são permitidas no modelo

8.4 TREINAMENTO DOS MODELOS MLP

O total de coeficientes do modelo RGMDH que obteve o melhor NMSE foi de 148, referente ao RGMDH de 5 camadas com múltiplas funções de ativação permitidas. Este número será então usado como base para termos modelos de ANNs MLP para benchmark do desempenho, visto que o desempenho de ANNs para modelagem de RFPAs já é bem consolidada. As entradas seguirão o mesmo comentado na Subseção 8.2.

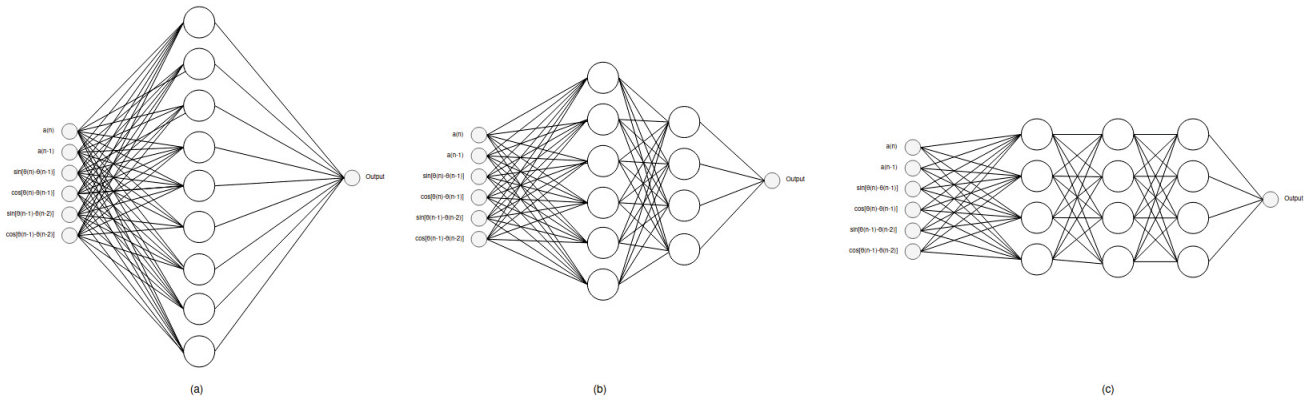
Para as MLPs, três variações serão avaliadas, todas utilizando a topologia da Figura 24. Primeiramente, um modelo com uma única camada será testado, resultando em uma rede rasa porém larga (com um maior número de nós). A segunda rede testada será seu oposto, sendo uma rede com duas camadas, mais profunda porém com menos nós por camada. Por fim, uma rede de 3 camadas será experimentada, com um número menor ainda de nós por camada. Todas as redes terão um total similar de coeficientes à rede GMDH da seção anterior, a fim de equipará-las na complexidade computacional. Trabalhos passados mostraram que, mesmo em um tamanho pequeno, crescer uma rede neural *feed-forward* no número de camadas tende a deixar sua estrutura mais diversa, mesmo que apenas em uma camada (ELDAN; SHAMIR, 2016), por isto a escolha de diferentes profundidades de MLP.

A fim de atingir uma complexidade computacional similar, mensurada pelo número de coeficientes, a rede com apenas uma camada, chamada daqui para frente de MLP 1, deverá ter 9 nós em sua camada única, totalizando 73 coeficientes por rede e 146 coeficientes no total do

modelo a serem estimados. Já a segunda rede, com duas camadas, será chamada de MLP 2 e deverá ter 6 nós em sua primeira camada e 4 nós na segunda, totalizando 75 coeficientes por rede e 150 coeficientes no total do modelo. Por fim, a terceira rede, com 3 camadas e chamada de MLP 3, terá cada uma destas com 4 neurônios, totalizando 73 coeficientes por rede e 146 coeficientes totais no modelo para serem estimados.

A arquitetura das três redes utilizadas pode ser observada na Figura 37. As MLPs terão todas a função de ativação na forma de tangente hiperbólica, com o processo de treinamento utilizando MSE como função de perda e algoritmo de otimização RMSprop.

FIGURA 37 – Estrutura dos Modelos de Redes MLP após o treinamento: (a) Rede MLP 1, com uma camada;(b) Rede MLP 2, com duas camadas; (c) Rede MLP 3, com três camadas



FONTE: O autor (2020)

Importante atentar que, para cada uma das redes, os modelos vistos na Figura 37 serão repetidos para as redes que estimarão as partes real e imaginária da saída, referente à Figura 24. Ou seja, não haverá distinção entre as redes responsáveis pela parte real e imaginária.

A Tabela 4 apresenta o NMSE de validação das três redes da Figura 37 para a modelagem inversa de um RFPA. Pode-se notar que as três apresentaram resultados muito próximos, com a MLP 2 obtendo a melhor acurácia. Porém, todas ainda estão mais de 15 dB distante do desempenho da RGMDH com 5 camadas.

TABELA 4 – NMSE, nós e coeficientes de um GMDH com 5 camadas, para diferentes funções de ativação

MLP com dados reais			
Modelo ANN	***nós***	***Coeficientes***	***NMSE***
MLP 1	18	146	-33,06 dB
MLP 2	20	150	-34,40 dB
MLP 3	32	146	-32,01 dB

FONTE: O autor (2020)

9 RESULTADOS DE SIMULAÇÃO COM A MODELAGEM DE RFPAS UTILIZANDO GMDH DE VALORES COMPLEXOS

No Capítulo 8, foi comentado sobre a modelagem comportamental de RFPAs utilizando modelos que trabalham apenas com dados reais. Porém, como comentado nas seções 6.5 e 7.4, é de interesse tentar trabalhar com o conjunto de dados complexos, sem necessitar de algoritmos ou topologias que resultem em representações destes dados complexos em valores puramente reais.

Este capítulo irá tratar de modelos que processam dados complexos, provenientes de um sistema representado por dados neste formato. Isto significa que todo o processo de treinamento, determinação de coeficientes, cálculo e otimização de erro lidará com valores compostos por parte real e parte imaginária.

9.1 DADOS DE ENTRADA, ALVOS E TOPOLOGIA DAS REDES COM VALORES COMPLEXOS

Diferente do que foi visto na Subseção 8.2, desta vez não há necessidade de utilizar o módulo ou componentes de fase como entrada, visto que ambas estas características são representadas integralmente pelo valor complexo. Ou seja, ao poder trabalhar com dados complexos, a única variação a princípio a ser acrescentada na entrada torna-se a componente de memória, os valores passados da entrada que influenciam a saída no instante atual.

A fim de tentar manter o número de entradas do CGMDH similar ao número utilizado no GMDH Real, será feito com que a memória M seja igual a 5. Com isto, o total será de 6 entradas para o modelo complexo, composto pelo instante atual e os 5 instantes anteriores desta.

Além disto, os alvos durante o treinamento também serão alterados. Anteriormente, havia sido descritas algumas maneiras diferentes de utilizar alvos quando era trabalhado apenas com dados reais para representar um sistema complexo. Isto significou em ter sido necessário utilizar duas redes neurais, uma prevendo a parte real e a outra prevendo a parte imaginária dos valores de saída. Com isto, os alvos eram a parte real e a parte imaginária da saída a ser prevista, subtraída da fase instantânea da entrada. Novamente, isto não é necessário quando trabalha-se com valores complexos. Os alvos da rede serão a saída complexa apenas. Isto também significa que poderá ser utilizado apenas uma rede na topologia do modelo, reduzindo pela metade o número de coeficientes tecnicamente necessários para o modelo.

9.2 FUNÇÕES DE ATIVAÇÃO DO CGMDH

Outro ponto interessante para ser observado no CGMDH é a escolha das funções de ativação, e como estas influenciarão no resultado final. Neste modelo com dados complexos, também serão utilizadas as funções de ativação polinomiais vistas no RGMDH, mais especificamente:

- **Linear**: $y_L = a_0 + a_1 x_i + a_2 x_j$
- **Linear com Covariância**: $y_{LC} = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j$
- **Quadrático**: $y_Q = a_0 + a_1 x_i + a_2 x_j + a_3 x_i x_j + a_4 x_i^2 + a_5 x_j^2$
- **Cúbico**: $y_C = y_Q + a_6 x_i^3 + a_7 x_j^3 + a_8 x_i^2 x_j + a_9 x_i x_j^2$

Mas, também é de interesse observar se o uso de funções específicas para valores complexos trarão benefícios ao modelo. A função a ser usada, referida daqui em diante de Conjugado Cúbico, será:

$$\begin{aligned} y(x) =& a_0 x_1 + a_1 x_2 + a_2 x_1 x_1 x_1 * + a_3 x_1 x_1 x_2 * + \\ & a_4 x_1 x_2 x_1 * + a_5 x_1 x_2 x_2 * + a_6 x_2 x_2 x_1 * + a_7 x_2 x_2 x_2 *, \end{aligned} \tag{9.1}$$

em que serão utilizados apenas os termos de ordem ímpar do polinômio (visto que apenas estes contribuem para a modelagem do RFPA), além de considerar também o complexo conjugado no modelo. O total de coeficientes a serem estimados para esta função de ativação será 8, estando então em sua complexidade computacional entre as funções polinomiais de segunda e terceira ordem usadas no RGMDH. Ou seja, mesmo adicionando o valor do complexo conjugado aos termos, será uma função de terceira ordem com menos coeficientes do que seu contraparte real, visto que descartamos todos os termos de segunda ordem dela, já que estes não contribuem ao resultado final da modelagem do RFPA.

9.3 TREINAMENTO DO MODELO CGMDH

De forma similar ao que foi feito com o RGMDH, o CGMDH será inicialmente treinado a fim de identificar o número ideal de camadas para a modelagem do RFPA. O treinamento iniciará não impondo limites ao número de camadas do modelo, e analisar a evolução do erro por camada, a fim de identificar em qual destas o erro para de evoluir de maneira significativa. Após isso, será imposto gradualmente limites de camadas, observando como o NMSE geral do modelo se comporta, a fim de determinar a melhor relação entre camadas, coeficientes e NMSE possível para o CGMDH.

Conforme sejá avaliada a evolução do treinamento por camada, será analisado como a mudança da função de ativação, entre as 5 citadas na seção anterior, impactará para a acurácia geral do modelo, através do NMSE. O intuito aqui é observar se os avanços vistos em técnicas de modelagem comportamental com funções polinomiais para RFPAs poderá ser transferido para a função de ativação do CGMDH. Além disto, na literatura não foram encontradas menções de variações de função de ativação do GMDH para quando tratamos de dados complexos.

9.3.1 Avaliação de funções de ativação para o CGMDH

O processo de seleção da função de ativação a ser utilizada será iniciado sem impor limites às camadas. A Tabela 5 apresenta o NMSE das redes de acordo com sua função de ativação, assim como o número de camadas obtidas. Será chamado de Funções Reais o modelo que permite que as quatro funções de ativação reais sejam utilizadas (igual foi feito nos resultados obtidos no Capítulo 8). O critério de parada estabelecido para estes resultados foi igual ao do RGMDH, ou seja, $\epsilon = 0,001$.

TABELA 5 – Número de Camadas e NMSE do CGMDH para diferentes funções de ativação, sem impor limites no número máximo de camadas.

CGMDH Variando Função de Ativação		
Função Ativação	***Camadas***	***NMSE***
Linear	10	-36,80 dB
Linear com Covariância	7	-36,78 dB
Quadrático	50	-37,01 dB
Cúbico	50	-37,11 dB
Funções Reais	3	-36,43 dB
Conjugado Cúbico	4	-41,76 dB

FONTE: O autor (2020)

Antes de entrarmos em detalhes sobre as camadas, uma coisa já pode ser notada: o NMSE da função de ativação Conjugado Cúbico está claramente abaixo do restante das funções especializadas em dados reais. Já pode ser afirmado de antemão que esta função será a mais interessante a ser utilizada daqui para frente no restante das otimizações do modelo com dados complexos.

Sobre o número de camadas, ou mais precisamente, sobre sua inconstância durante as diferentes funções de ativação na Tabela 5, alguns comentários podem ser feitos. Primeiramente, as funções de segunda e terceira ordem (Quadrático e Cúbico, respectivamente) tiveram um número desproporcional de camadas em relação ao restante. Pode-se supor que provavelmente o modelo não está evoluindo de maneira adequada conforme foram adicionadas camadas. Por

isto o modelo permaneceu adicionando-as, com o intuito de minimizar o erro, mas não sendo capaz de minimizá-lo de maneira efetiva, necessitando então de 50. Além disso, seu erro foi pouco melhor do que as funções Lineares, um comportamento oposto do que foi visto com dados reais na Tabela 3.

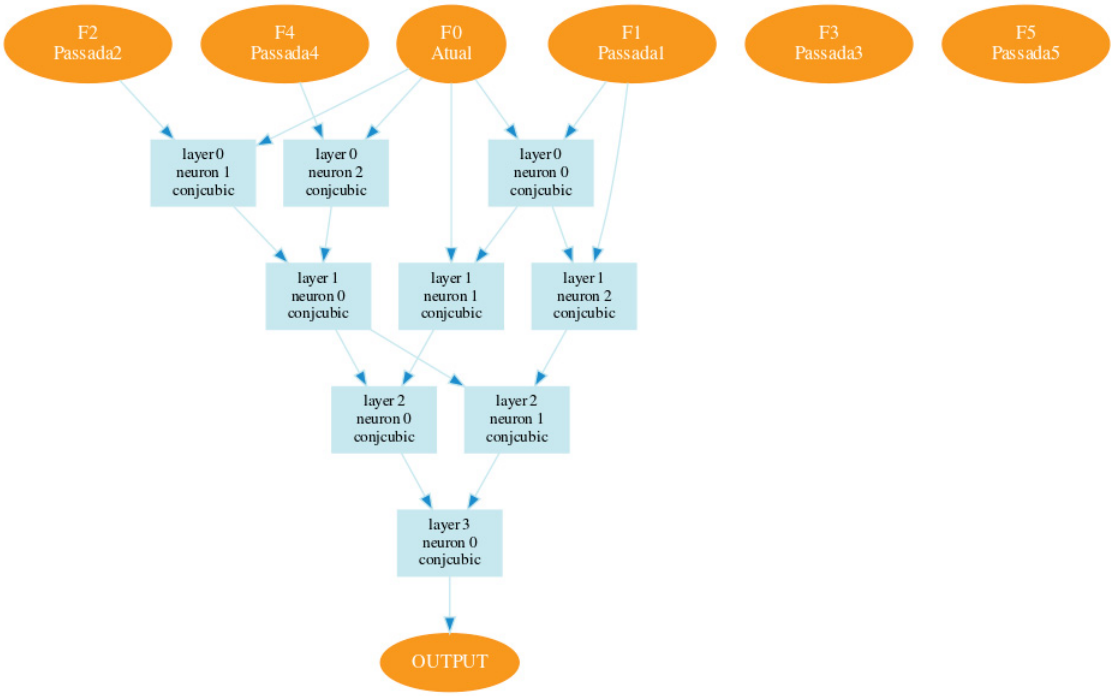
Com isto em mente, as próximas seções trabalharão o CGMDH apenas com a função Conjugado Cúbico, a fim de entender melhor o modelo gerado e como será possível otimizá-lo.

9.3.2 Otimizando o CGMDH com Função de Ativação Conjugado Cúbico

Para gerar o modelo de CGMDH com função de ativação Conjugado Cúbico da Equação (9.1), foi utilizada uma memória M = 5, ou seja, totalizando 6 entradas no modelo, contando a entrada instantânea além das 5 passadas. Obviamente, estas entradas são de natureza puramente complexa, contendo nelas já toda a informação necessária de módulo e fase.

Agora, alguns comentários serão postos a fim de entender o modelo gerado que teve o resultado mostrado na Tabela 5. O modelo auto organizável gerado durante o treinamento desta, sem haver sido imposto um limite no número máximo de camadas, e com o treinamento parando quando o critério $\epsilon = 0,001$ fosse atingido, pode ser visto na Figura 38.

FIGURA 38 – Arquitetura da Rede CGMDH para estimar os valores complexos de saída, sem impor limites ao número máximo de camadas durante o treinamento.

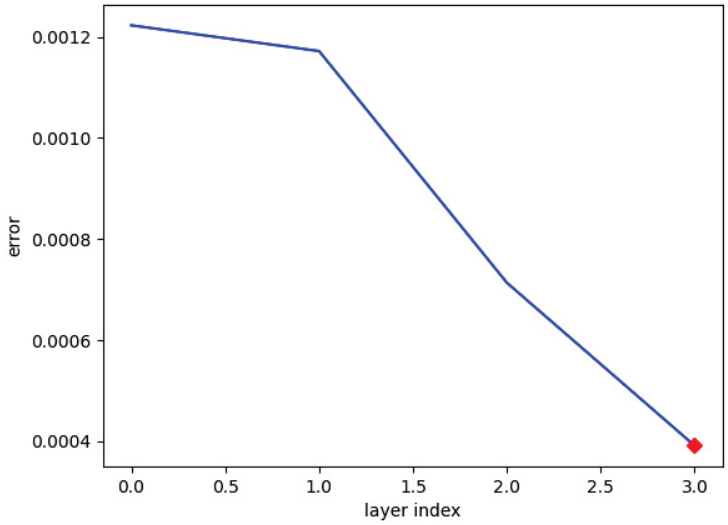


FONTE: O autor (2020)

Como já foi comentado, o NMSE desta foi de -41,76 dB. Seu número total de nós foi 9, com um total de 72 coeficientes a serem estimados. Um ponto interessante notado aqui foi o fato das entradas nos instantes $M = 3$ e $M = 5$ serem totalmente descartadas do modelo. A escolha do algoritmo de treinamento em não utilizar estes instantes de memória a princípio é arbitrária, e não pode ser concluído ainda ser um comportamento esperado de qualquer modelagem de RFPA.

A evolução do erro por camadas é visto na Figura 39. Pode-se ver que o erro evolui de maneira satisfatória camada por camada. Mas, ainda há espaços para entender se o modelo poderá melhorar sua precisão conforme forem adicionadas novas camadas.

FIGURA 39 – Evolução do erro por camada do CGMDH com Função de ativação Conjugado Cúbico, sem impor limites ao número máximo de camadas.



FONTE: O autor (2020)

Mesmo alterando alguns fatores de treinamento, como o critério de parada ϵ, o modelo ainda escolheu sempre parar na quarta camada. Por isto, não faz sentido manter uma análise igual a realizada no capítulo da modelagem com valores reais, visto que o modelo sem limitações já gerou um resultado satisfatório para o número de coeficientes e camadas.

De fato, como adicionar camadas não trouxe melhorias à acurácia, a única maneira encontrada para conseguir reduzir o NMSE foi introduzir algumas funções de ativação para atuar em conjunto com a Conjugado Cúbico. Estes resultados são vistos na Tabela 6.

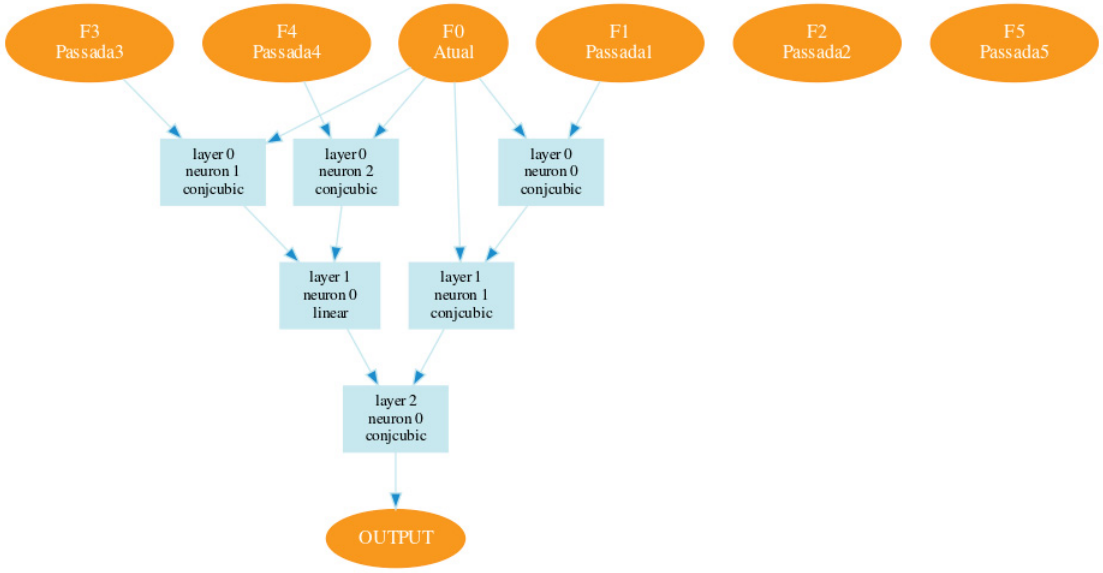
TABELA 6 – NMSE, número de Camadas e Coeficientes de um CGMDH com a Função Conjugado Cúbico acrescida de outras funções de ativação

CGMDH com Funções de Ativação Mistas			
Função Ativação	***Camadas***	***Coeficientes***	***NMSE***
Conjugado Cúbico	4	72	-41,76 dB
Conjugado Cúbico + Linear	3	43	-41,15 dB
Conjugado Cúbico + Linear Covariância	3	44	-43,11 dB

FONTE: O autor (2020)

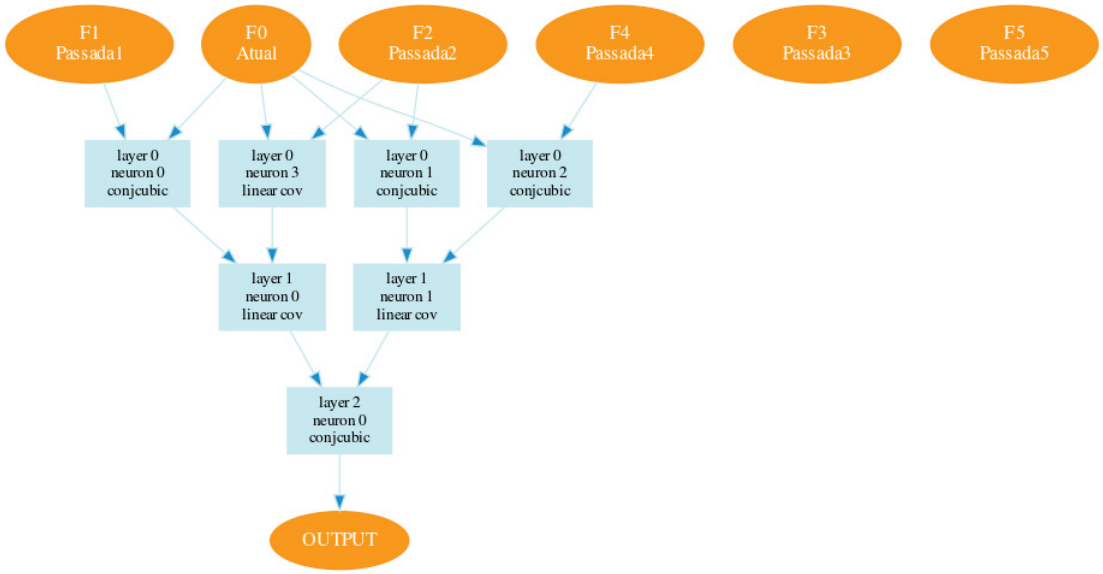
Os modelos auto organizados gerados durante o treinamento para a combinação da Conjugado Cúbico com a função Linear e Linear com Covariância podem ser vistos nas Figuras 40 e 41, respectivamente.

FIGURA 40 – Arquitetura da Rede CGMDH para estimar os valores complexos de saída, sem impor limites ao número de camadas máximo durante o treinamento e utilizando as funções de ativação Complexo Conjugado e Linear.



FONTE: O autor (2020)

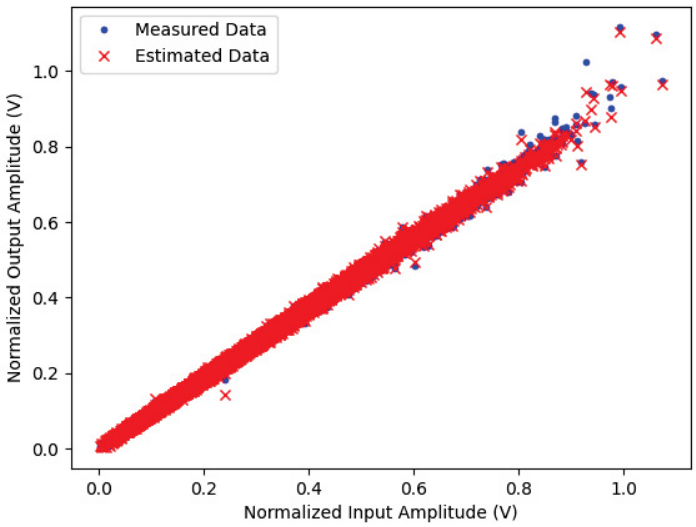
FIGURA 41 – Arquitetura da Rede CGMDH para estimar os valores complexos de saída, sem impor limites ao número de camadas máximo durante o treinamento e utilizando as funções de ativação Complexo Conjugado e Linear com Covariância.



FONTE: O autor (2020)

Por fim, a Figura 42 representa o gráfico AM-AM do CGMDH com as funções de ativação Conjugado Cúbico e Linear Covariância. Os pontos azuis são os dados reais do RFPA, e os dados em vermelho são o resultado da previsão do modelo. Pode-se notar que mesmo pontos mais dispersos do modelo, presentes em valores próximo a 1 na relação de entrada-saída, tiveram uma boa acurácia na previsão do CGMDH.

FIGURA 42 – Gráfico AM-AM da modelagem inversa de um RFPA utilizando o CGMDH, com funções de ativação Conjugado Cúbico e Linear Covariância. Em azul, os dados reais do RFPA, e em vermelho os dados previstos pelo modelo.



FONTE: O autor (2020)

9.4 VALIDAÇÃO DE FUNÇÕES DE ERRO/AVALIAÇÃO PARA O CGMDH

Conforme comentado nas Subseções 7.4.2.2 e 7.4.2.3, o MSE nem sempre é a melhor métrica para validação da evolução do erro de treinamento de um modelo baseado em dados complexos, por não considerar completamente as evoluções do erro da modelagem da fase. Por isto, foi proposto validar uma nova função que leve em conta tanto magnitude como fase dos valores previstos. Esta foi vista na Equação (7.13), e chamada de erro logaritmo. Esta seção busca entender as diferenças o desempenho total do modelo (medida através do NMSE), comparando o erro logaritmo com o MSE.

Para isto, será comparado os resultados de NMSE obtido nas seções anteriores, que utilizavam o MSE como métrica de avaliação de desempenho da evolução do treinamento, com novos modelos gerados utilizando o erro logaritmo. Idealmente, o número de coeficientes entre modelos gerados deverá ser igual, a fim de comparar modelos de complexidade computacional similar. É importante notar que, como a forma de avaliação do treinamento será alterada, modelos completamente novos serão obtidos.

A Tabela 7 traz o tamanho dos modelos e sua acurácia, para diferentes funções de ativação. O intuito aqui é tentar aproximar o total de coeficientes com aqueles vistos na Tabela 6, que utilizava o MSE, para cada função de ativação. Pode-se ver nela resultados mistos. Para o modelo que utilizava apenas a função Conjugado Cúbico, o erro teve uma melhora de 0,68 dB, assim como a mista com a função real Linear teve uma melhora de 0,40 dB. Porém, para o modelo com o misto com a função Linear Covariância, o NMSE total do modelo teve uma piora de 1,61 dB. Além disto, mesmo com a pequena melhora nos dois primeiros modelos citados nesta seção, o melhor resultado obtido de NMSE continua os -43,11 dB do CGMDH com a avaliação utilizando o MSE e funções de ativação Conjugado Cúbico e Linear Covariância.

TABELA 7 – NMSE, número de Camadas e Coeficientes de um CGMDH com diferentes funções de ativação, utilizando o erro logaritmo para avaliar a evolução do treinamento

CGMDH com Avaliação de Erro Logaritmo			
Função Ativação	***Camadas***	***Coeficientes***	***NMSE***
Conjugado Cúbico	4	80	-42,44 dB
Conjugado Cúbico + Linear	3	63	-41,55 dB
Conjugado Cúbico + Linear Covariância	3	64	-41,50 dB

FONTE: O autor (2020)

9.5 CONCLUSÕES PARCIAIS SOBRE O TREINAMENTO E GERAÇÃO DE MODELOS COM O CGMDH

Com as Figuras 40 e 41, em conjunto das Tabelas 6 e 7, algumas conclusões interessantes do treinamento podem ser ditas. Primeiro, pode ser notado que, ao introduzir a função de ativação Linear com Covariância ao treinamento, não apenas foi obtido uma redução de 1,3 dB no NMSE, mas também uma redução de quase 40% no número de coeficientes do modelo. Não só isso, mas esta função, ao ser escolhida na segunda camada do modelo, foi capaz de trazer melhorias ao resultado a ponto de reduzir uma camada em relação ao modelo que utilizou apenas a função Conjugado Cúbico.

A função Linear com Covariância tem apenas 4 coeficientes a serem estimados, em comparação aos 8 da função Conjugado Cúbico. Isto explica a grande redução de coeficientes visto na Tabela 6, um fato benéfico para a implementação física do modelo.

Também foi avaliado o uso de uma nova função de erro, específica para um modelo com dados complexos. Os resultados obtidos foram mistos, com algumas funções se beneficiando deste, e outras perdendo acurácia. Mas, no geral, o melhor resultado deste capítulo continua a ser obtido utilizando o MSE como avaliação da evolução do treinamento. Porém, ainda é interessante considerar o uso do MSE e do erro logaritmo em treinamento, visto que este

trabalho não traz conclusões decisivas neste assunto sobre qual trará maiores benefícios ao treinamento do CGMDH.

O principal ponto notado também em relação ao CGMDH não é sua incapacidade em melhorar a acurácia em relação ao RGMDH, mas sim a possibilidade de grande redução de complexidade do modelo, graças a um número reduzido de coeficientes a serem estimados durante o treinamento e armazenados posteriormente em uma implementação. Torna-se interessante então, como último passo deste trabalho, validar diretamente os modelos com dados reais e dados complexos, a fim de validá-los em cenários similares para entender o real custo-benefício entre acurácia e complexidade computacional. Este assunto será abordado no capítulo seguinte.

10 COMPARAÇÕES ENTRE A MODELAGEM COMPORTAMENTAL DE RFPAS UTILIZANDO REDES COM DADOS REAIS E COMPLEXOS

Nos dois últimos capítulos, foi desenvolvido o treinamento e seus ajustes para que os modelos matemáticos tivessem resultados satisfatórios na modelagem comportamental de RFPAs, tanto para dados reais quanto para dados complexos. Porém, estes dois ainda não foram comparados diretamente, a fim de validar o custo-benefício destes em relação à acurácia e complexidade computacional. Este capítulo apresentará esta comparação, a fim de destacar as diferenças entre estes, mais especificamente visando comparar o RGMDH com o CGMDH.

10.1 SUMÁRIO DOS RESULTADOS PRELIMINARES OBTIDOS

A Tabela 8 resume os resultados de interesse obtidos nos dois últimos capítulos. Estão presentes os 3 modelos propostos de ANNs com dados reais, o RGMDH limitado a 5 camadas e o CGMDH com funções Conjugado Cúbico e Linear com Covariância. Fica claro algo destoante entre estes 5 modelos: o número de coeficientes do CGMDH é menos de um terço do restante. Embora o CGMDH tenha tido resultados inferiores ao RGMDH, sua diferença no número de coeficientes deixa a questão de como seria a comparação de desempenho entre ambos nivelando este quesito.

TABELA 8 – Resultados de NMSE e número de coeficientes dos modelos gerados até agora, para a modelagem inversa de um RFPA

Resultados do NMSE e Coeficientes, para múltiplos modelos		
Modelo	***Coeficientes***	***NMSE***
MLP 1	146	-33,06 dB
MLP 2	150	-34,40 dB
MLP 3	146	-32,01 dB
RGMDH com 5 camadas	148	-50,17 dB
CGMDH com Conjugado Cúbico + Linear Covariância	44	-43,11 dB

FONTE: O autor (2020)

10.2 RGMDH COM REDUÇÃO DE COEFICIENTES

Agora, será utilizado o modelo RGMDH de 5 camadas gerado no Capítulo 8 e tentar simplificá-lo para obter um número de coeficientes similares ao CGMDH da Tabela 8. Ou seja, reduzir os 148 coeficientes para um número próximo de 44. Algumas maneiras podem ser

testadas para entender a melhor forma de reduzir o tamanho do modelo com o menor impacto à acurácia.

Uma alternativa inicial é limitar o número de camadas das duas redes necessárias para o RGMDH. Ambas as redes, vistas anteriormente nas Figuras 35 e 36, eram formadas apenas por funções de ativação de segunda e terceira ordem. Eliminar um neurônio, através da limitação de camadas, já significa reduzir 10 coeficientes de cada rede (o total de uma função de terceira ordem).

A Tabela 9 mostra o total de camadas por rede, coeficientes totais do modelo e NMSE da modelagem inversa de um RFPA, a fim de identificar a melhor opção quando é reduzido o número de camadas do modelo. Foram utilizadas as funções de ativação Linear, Linear com Covariância, Quadrática e Cúbica, sendo essa combinação a que obteve melhores resultados ao RGMDH. Os mesmos limites foram impostos tanto para a rede responsável pela parte real, quanto para a rede responsável pela parte imaginária. É interessante destacar que, para os modelos com 4 e 3 camadas, o treinamento selecionou apenas nós com função de segunda e terceira ordem, com 6 e 10 coeficientes por neurônio, respectivamente. Então, por mais que tenhamos reduzido as camadas, cada neurônio ainda contribuirá significativamente ao valor total de coeficientes a serem estimados.

Para a rede com 2 camadas, o treinamento também selecionou automaticamente apenas nós com função de ativação de segunda e terceira ordem. Porém, como uma rede possuiu apenas 2 nós de terceira ordem, enquanto a outra possuía 2 nós de terceira e um de segunda ordem, foi possível reduzir seus coeficientes para 48.

TABELA 9 – Resultados de NMSE e total de coeficientes dos modelos RGMDH para diferentes limitações de camadas, para a modelagem inversa de um RFPA

RGMDH com Limitações de Camadas		
Camadas por Rede	***Coeficientes Total***	***NMSE***
4	128	-49,47 dB
3	88	-40,46 dB
2	46	-45,54 dB

FONTE: O autor (2020)

10.3 ANNS COM REDUÇÃO DE COEFICIENTES

De maneira similar ao que foi feito com o RGMDH, será simplificada a estrutura dos modelos de ANNs, a fim de tentar chegar a um número similar de 44 coeficientes. Os resultados desta redução são observados na Tabela 10. O Modelo MLP 4 será similar ao MLP 1, em que será utilizada apenas uma camada para a rede. Para reduzir o número total de coeficientes

do modelo, foi necessário utilizar apenas 3 neurônios nesta camada única, totalizando 50 coeficientes. Já a rede MLP 5 tenta conseguir um pouco de profundidade ao modelo, com duas camadas. Como o número total de coeficientes é reduzido, foi obtido apenas 2 neurônios na primeira camada e 2 neurônios na segunda, totalizando 46 coeficientes.

TABELA 10 – Resultados de NMSE e número de coeficientes das ANNs com valores reais, em modelos de coeficientes reduzidos.

ANN MLP com Redução de Coeficientes		
Modelo	***Coeficientes Total***	***NMSE***
MLP 4	50	-35,10 dB
MLP 5	46	-30,35 dB

FONTE: O autor (2020)

Nota-se que a redução de coeficientes impactou no desempenho das ANNs, que já haviam obtidos resultados inferiores ao GMDH com um número maior de coeficientes.

10.4 COMPARAÇÃO DIRETA ENTRE RGMDH, CGMDH E MLP COM NÚMERO SIMILAR DE COEFICIENTES

Por fim, será compilado os resultados obtidos nos 3 modelos matemáticos apresentados neste trabalho. A Tabela 11 apresenta o NMSE de cada um destes modelos, assim como o total de coeficientes a serem estimados por cada um destes.

TABELA 11 – Resultados de NMSE e número de coeficientes dos 3 modelos matemáticos para modelagem inversa de um RFPA.

ANN MLP com Redução de Coeficientes		
Modelo	***Coeficientes Total***	***NMSE***
RGMDH com 2 Camadas	46	-45,54 dB
CGMDH com Conjugado Cúbico + Linear Covariância	44	-43,11 dB
MLP 4	50	-35,10 dB
MLP 5	46	-30,35 dB

FONTE: O autor (2020)

Já é possível ver que ambos os modelos baseados em GMDH obtiveram resultados melhores do que aqueles baseados em ANN MLP, quando comparados com um número similar de coeficientes. Comparando os modelos de melhor desempenho, é visto que o RGMDH teve um NMSE alguns dB abaixo do CGMDH, com um número similar de coeficientes totais a

serem estimados nos modelos. Porém, o desempenho para apenas este conjunto de dados não deverá significar que o RGMDH seria o modelo superior, sendo interessante no futuro analisar a diferença de desempenho entre estes para outros RFPAs.

11 CONCLUSÕES E SUGESTÕES PARA TRABALHOS FUTUROS

Este capítulo irá trazer um apanhado dos resultados obtidos neste trabalho, trazendo comentários adicionais sobre o que foi obtido nas simulações dos modelos propostos. Além disto, irão ser discutidos próximos passos e perspectivas futuras após as conclusões deste trabalho.

11.1 CONCLUSÕES E COMENTÁRIOS SOBRE OS RESULTADOS DO GMDH NA MODELAGEM COMPORTAMENTAL DE RFPAS

Este trabalho apresentou a abordagem do GMDH para o problema da identificação e modelagem de um sistema composto por um RFPA. Seu intuito principal era validar como o GMDH se comportaria contra um método já estabelecido e bem pesquisado, as ANNs MLP, neste problema. A hipótese principal é de que, para modelos com acurácia similar, o GMDH seria mais simples computacionalmente do que MLPs, visto a natureza polinomial de suas funções de ativação resultar em um processo de treinamento simplificado, quando comparado ao *backpropagation* utilizado em MLPs. Além disto, o treinamento do GMDH já possui um algoritmo de otimização de coeficientes. Ou seja, já serão selecionados apenas os nós que melhor contribuem ao resultado total do modelo. Para obter isto em MLPs, é necessário um algoritmo a parte, que costuma ser executado apenas após o treinamento do modelo ter sido completado.

Ao avaliar o RGMDH, modelo que trabalhará apenas com dados reais, foi observado como a evolução da acurácia deste modelo evoluiu conforme foram impostas restrições nas camadas. Isto ocorreu pois foi evitado o *overfitting* do modelo durante o treinamento, ou seja, este ficar especializado nos dados de treinamento e não ser generalista o suficiente, o que foi observado pela menor acurácia com os dados de validação. Além disto, reduzir o número de camadas máximo do RGMDH significou simplificar computacionalmente o modelo, reduzindo drasticamente o total de nós que necessitam ter seus coeficientes estimados. Também foram avaliadas diferenças de acurácia para funções de ativação de diferentes ordens de complexidade. Nota-se que o valor do NMSE decresceu de maneira significativa quando foram empregadas funções não lineares (Quadrática e Cúbica), em relação ao modelos utilizando apenas funções lineares, com todos limitados a 5 camadas. Mas, é importante também notar que o número de coeficientes do modelo tendeu a dobrar conforme a ordem de não linearidade do modelo subiu. No caso da mudança de uma função de primeira ordem que considera apenas a covariância entre as entradas para uma função de segunda ordem, o total de coeficientes cresceu de 48 para 96 (+100% de crescimento), com uma redução de 4,84 dB no NMSE. Porém, comparando o uso de uma função de segunda ordem para uma de terceira ordem, o total de coeficientes subiu para 190 (+97%), com redução do NMSE de 1,32 dB.

Porém, o melhor desempenho vista do RGMDH foi obtida quando o misto entre funções lineares e não lineares foi utilizado. Para um modelo limitado a 5 camadas, o NMSE obtido foi de -50,17 dB, uma melhora de 6,04 dB em relação ao modelo que utilizou exclusivamente funções de terceira ordem. Além do mais, este modelo totalizou 148 coeficientes a serem estimados, 42 a menos que o modelo exclusivamente de terceira ordem. Analisando a topologia dos modelos gerados, nota-se que as primeiras camadas utilizam funções não lineares, enquanto as camadas finais tendem a utilizar funções lineares. O uso misto entre estas funções torna-se eficaz visto que, nas camadas finais do modelo, o intuito principal é apenas agregar os resultados obtidos nas primeiras camadas, a fim de poder obter a saída. Reduzir a ordem de não linearidade também evita que esta recombinação termine afetando o resultado final, evitando também o *overfitting*.

Por fim, ao comparar o RGMDH com outro modelo de rede que processa dados exclusivamente reais, a MLP, ao equiparar o número de coeficientes em aproximadamente 148 e gerar 3 modelos diferentes de MLP, variando tanto largura da rede como profundidade, foi visto que nenhum dos 3 conseguiu chegar próximo da acurácia do RGMDH, estando todos com mais de 15 dB de diferença no NMSE da validação de modelagem inversa do RFPA.

Também foi proposto o modelo CGMDH, a fim de processar dados complexos, sem a necessidade de transformá-los para uma representação real. O princípio deste modelo é reduzir o total de coeficientes a serem estimados ao utilizar apenas uma rede que fará a previsão do valor complexo, já que para dados reais era necessário duas redes estimando parte real e imaginária separadamente, além de evitar a perda de informações ao prever valores em redes separadas. Além disto, baseado em avanços vistos em modelos de funções polinomiais, foi proposta uma nova função de ativação para o CGMDH, baseada em características de RFPAs ao serem modelados por séries de Volterra. Esta foi chamada de Conjugado Cúbico. Ao comparar o CGMDH com as funções de ativação real com esta nova função para dados complexos, foi vista uma melhoria de mais de 4 dB desta nova função em relação àquelas utilizadas anteriormente no RGMDH. Também foi analisado o uso de funções mistas de ativação, já que a combinação de funções lineares e não lineares teve bons resultados no RGMDH. Para o CGMDH, foi obtido um NMSE de -43,11 dB utilizando as funções Conjugado Cúbico e Linear Covariância, uma melhora de 1,35 dB em relação ao modelo apenas com Conjugado Cúbico. O total de coeficientes deste modelo foi de 44, também uma grande redução aos 148 vistos no melhor resultado do RGMDH, embora o desempenho do CGMDH esteja cerca de 7 dB abaixo do RGMDH. Para finalizar as análises do CGMDH, também foi proposta uma alternativa ao MSE durante a avaliação da evolução do erro de treinamento. Esta função logarítmica obteve resultados mistos, tendo uma melhora no NMSE quando utilizando apenas a função Conjugado Cúbico e em conjunto com a função linear, mas uma piora na acurácia para o conjunto com a Linear Covariância. Além disto, o número de coeficientes cresceu cerca de 50% para todos os modelos. Por isto, não é possível ter resultados conclusivos de um impacto real desta nova função de avaliação de erro de treinamento para o CGMDH.

Ao comparar os resultados obtidos entre RGMDH e CGHDM, a fim de compreender as diferenças de acurácia entre estes, foi vista a necessidade de reduzir o total de coeficientes do RGMDH de 148 para cerca de 44, a fim de equipará-los computacionalmente. O mesmo foi feito com os modelos de MLP, a fim de ter uma base entre as técnicas já consolidadas e o proposto por este trabalho. Foi possível reduzir o total de coeficientes do RGMDH para 46, obtendo um NMSE de -45,54 dB. Já os dois modelos de MLP propostos, o primeiro com uma única camada e o segundo com duas, também tiveram redução para 50 e 46, respectivamente. Porém, o melhor NMSE de uma MLP com esta redução foi de -35,10 dB. No geral, fica claro o benefício de acurácia do GMDH, tanto em sua variante para dados reais como a variante de dados complexos, em relação a MLPs. Melhores resultados foram obtidos tanto para modelos com um total maior de coeficientes, como para modelos com uma redução drástica da quantidade de coeficientes. Comparando ambos os GMDH, a variante real obteve um NMSE 2,54 dB melhor que o CGMDH, com uma complexidade computacional similar.

11.2 PRÓXIMOS PASSOS PARA O GMDH NA MODELAGEM COMPORTAMENTAL DE RFPAS

Não é possível afirmar que o RGMDH obterá sempre melhores resultados que o CGMDH na modelagem de RFPAs. Como o treinamento é totalmente influenciado pelo conjunto de dados utilizados, gerando modelos completamente diferentes, torna-se interessante propor para trabalhos futuros a implementação destes dois modelos de GMDH para a modelagem inversa de diferentes tipos de RFPA, a fim de validar a capacidade de ser um algoritmo generalista e com pouca interferência humana no processo de otimização. Mas, é possível afirmar de maneira positiva que o GMDH mostrou-se um modelo mais atrativo para a modelagem da característica inversa de um RFPA sob análise do que MLPs, em modelos de complexidade computacional similar e número de coeficientes reduzidos. No futuro, também torna-se interessante entender as diferenças entre o GMDH com relação a funções polinomiais agregadas a técnicas de redução de coeficientes, visto que ambos apresentam semelhanças em seus princípios.

O fato do GMDH obter um desempenho interessante da modelagem inversa, empregando um reduzido número de coeficientes também deixa-o como um modelo atraente para a implementação em HW. Trabalhos futuros poderão abordar sua implementação em uma FPGA, a fim de servir como o bloco DPD de um sistema de linearização de um RFPA real. Esta será a forma crucial para validar este modelo em aplicações de mundo real. Além disto, também é interessante estudar o desempenho de uma FPGA atualizando em tempo real os pesos do GMDH. Como o treinamento do GMDH é a princípio menos intensivo de realizar do que o de MLPs, um modelo *online* torna-se mais propício a ter sucesso. Porém, atenção precisará ser dada para a quantidade de multiplicações e inversões de matrizes a serem feitas em paralelo, visto que este é um real limitante em FPGAs e, caso não sejam impostos limites no tamanho de crescimento do GMDH, como número de camadas e total de nós por camada a

serem avaliados, pode ocorrer problemas de latência elevada e alta carga computacional na atualização em tempo real destes coeficientes.

REFERÊNCIAS

AIZENBERG, Igor. **Complex-Valued Neural Networks with Multi-Valued Neurons**. [S.l.]: Springer Berlin Heidelberg, 2011. v. 353. ISBN 978-3-642-20352-7. DOI: 10.1007/978-3-642-20353-4. Citado 1 vez na página 48.

ARORA, Jasbir S. Jan A. Snyman, Practical Mathematical Optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms. **Structural and Multidisciplinary Optimization**, v. 31, 3 2006. ISSN 1615-147X. DOI: 10.1007/s00158-005-0595-0. Citado 1 vez na página 44.

BENEDETTO, S.; BIGLIERI, E.; DAFFARA, R. Modeling and performance evaluation of nonlinear satellite links-a volterra series approach. **IEEE Transactions on Aerospace and Electronic Systems**, AES-15, 4 1979. ISSN 00189251. DOI: 10.1109/TAES.1979.308734. Citado 3 vezes nas páginas 37, 39, 64.

BENVENUTO, N.; PIAZZA, F. On the Complex Backpropagation Algorithm. **IEEE Transactions on Signal Processing**, v. 40, 4 1992. ISSN 19410476. DOI: 10.1109/78.127967. Citado 1 vez na página 50.

BENVENUTO, N.; PIAZZA, F.; UNCINI, A. A neural network approach to data predistortion with memory in digital radio systems. In: ISBN 0-7803-0950-2. DOI: 10.1109/ICC.1993.397263. Citado 1 vez na página 46.

CHEN, S.; MCLAUGHLIN, S.; MULGREW, B. Complex-valued radial basic function network, Part I: Network architecture and learning algorithms. **Signal Processing**, v. 35, 1 1994. ISSN 01651684. DOI: 10.1016/0165-1684(94)90187-2. Citado 1 vez na página 52.

CHEN, S.; MCLAUGHLIN, S.; MULGREW, B. Complex-valued radial basis function network, Part II: Application to digital communications channel equalisation. **Signal Processing**, v. 36, 2 1994. ISSN 01651684. DOI: 10.1016/0165-1684(94)90206-2. Citado 1 vez na página 52.

CHRISIKOS, G. et al. A nonlinear ARMA model for simulating power amplifiers. In: ISBN 0-7803-4471-5. DOI: 10.1109/MWSYM.1998.705095. Citado 1 vez na página 35.

CLARK, C.J. et al. Time-domain envelope measurement technique with application to wideband power amplifier modeling. **IEEE Transactions on Microwave Theory and Techniques**, v. 46, 12 1998. ISSN 00189480. DOI: 10.1109/22.739245. Citado 1 vez na página 35.

CRIPPS, S.C. RF power amplifiers for wireless communications. **IEEE Microwave Magazine**, v. 1, 1 2000. ISSN 1527-3342. DOI: 10.1109/mmw.2000.823830. Citado 2 vezes nas páginas 19, 23.

CYBENKO, G. Approximation by superpositions of a sigmoidal function. **Mathematics of Control, Signals, and Systems**, v. 2, 4 1989. ISSN 09324194. DOI: 10.1007/BF02551274. Citado 1 vez na página 44.

DORN, Márcio et al. A GMDH polynomial neural network-based method to predict approximate three-dimensional structures of polypeptides. **Expert Systems with Applications**, v. 39, 15 2012. ISSN 09574174. DOI: 10.1016/j.eswa.2012.04.046. Citado 1 vez na página 54.

ELDAN, Ronen; SHAMIR, Ohad. The power of depth for feedforward neural networks. In: v. 49. Citado 2 vezes nas páginas 56, 83.

FREIRE, Luiza B.C.; FRANÇA, Caroline de; LIMA, Eduardo G. de. Low-pass equivalent behavioral modeling of RF power amplifiers using two independent real-valued feed-forward neural networks. **Progress In Electromagnetics Research C**, v. 52, 2014. ISSN 15309681. DOI: 10.2528/PIERC14070207. Citado 1 vezes nas páginas 46, 48.

FREIRE, Luiza B.Chipansky; FRANCA, Caroline De; LIMA, Eduardo G. De. A modified real-valued feed-forward neural network low-pass equivalent behavioral model for RF power amplifiers. **Progress In Electromagnetics Research C**, v. 57, 2015. ISSN 19378718. DOI: 10.2528/pierc15022802. Citado 2 vezes nas páginas 48, 49, 70, 71.

GEORGIOU, George M.; KOUTSOUGERAS, Cris. Complex Domain Backpropagation. **IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing**, v. 39, 5 1992. ISSN 10577130. DOI: 10.1109/82.142037. Citado 2 vezes nas páginas 51, 52.

HAYKIN, Simon. **An Introduction to Analog and Digital Communications**. [S.l.]: Wiley, 2006. ISBN 978-0-471-43222-7. Citado 2 vez na página 22.

HAYKIN, Simon. **Neural Networks: A Comprehensive Foundation**. [S.l.]: Pearson, jan. 1998. ISBN 0132733501. Citado 2 vezes nas páginas 44, 49.

HIROSE, Akira. **Complex-valued neural networks**. [S.l.]: Springer, jan. 2012. P. 3–175. ISBN 9783642276316. Citado 1 vez na página 48.

HIROSE, Akira; YOSHIDA, Shotaro. Generalization characteristics of complex-valued feedforward neural networks in relation to signal coherence. **IEEE Transactions on Neural Networks and Learning Systems**, v. 23, 4 2012. ISSN 2162237X. DOI: 10.1109/TNNLS.2012.2183613. Citado 1 vez na página 48.

IBUKAHLA, M. et al. Neural networks for modeling nonlinear memoryless communication channels. **IEEE Transactions on Communications**, v. 45, 7 jul. 1997. ISSN 00906778. DOI: 10.1109/26.602580. Citado 1 vezes nas páginas 45, 46.

ISAKSSON, Magnus; WISELL, David; RÖNNOW, Daniel. A comparative analysis of behavioral models for RF power amplifiers. In: v. 54. DOI: 10.1109/TMTT.2005.860500. Citado 2 vezes nas páginas 33, 38.

ISAKSSON, Magnus; WISELL, David; RÖNNOW, Daniel. Wide-band dynamic modeling of power amplifiers using radial-basis function neural networks. In: v. 53. DOI: 10.1109/TMTT.2005.855742. Citado 1 vez na página 45.

IVAKHNENKO, A. G. Polynomial Theory of Complex Systems. **IEEE Transactions on Systems, Man and Cybernetics**, v. 1, 4 1971. ISSN 21682909. DOI: 10.1109/TSMC.1971.4308320. Citado 4 vezes nas páginas 53, 57, 62.

IVAKHNENKO, A. G. The group method of data handling in long-range forecasting. **Technological Forecasting and Social Change**, v. 12, 2-3 1978. ISSN 00401625. DOI: 10.1016/0040-1625(78)90057-4. Citado 1 vez na página 53.

JEDDI, Sima; SHARIFIAN, Saeed. A hybrid wavelet decomposer and GMDH-ELM ensemble model for Network function virtualization workload forecasting in cloud computing. **Applied Soft Computing Journal**, v. 88, 2020. ISSN 15684946. DOI: 10.1016/j.asoc.2019.105940. Citado 1 vez na página 54.

JIANPING, D.; SUNDARARAJAN, N.; SARATCHANDRAN, P. Complex-valued minimal resource allocation network for nonlinear signal processing. **International journal of neural systems**, v. 10, 2 2000. ISSN 01290657. DOI: 10.1142/s0129065700000090. Citado 1 vez na página 50.

JIANPING, Deng; SUNDARARAJAN, Narasimhan; SARATCHANDRAN, P. Communication channel equalization using complex-valued minimal radial basis function neural networks. **IEEE Transactions on Neural Networks**, v. 13, 3 2002. ISSN 10459227. DOI: 10.1109/TNN.2002.1000133. Citado 1 vez na página 50.

KIM, Taehwan; ADALI, Tülay. Fully complex multi-layer perceptron network for nonlinear signal processing. **Journal of VLSI Signal Processing Systems for Signal, Image, and Video Technology**, v. 32, 1-2 2002. ISSN 13875485. DOI: 10.1023/A:1016359216961. Citado 4 vezes nas páginas 50–52.

LEUNG, Henry; HAYKIN, Simon. The Complex Backpropagation Algorithm. **IEEE Transactions on Signal Processing**, v. 39, 9 1991. ISSN 19410476. DOI: 10.1109/78.134446. Citado 1 vez na página 52.

LIMA, Eduardo G.; CUNHA, Telmo R.; PEDRO, José C. A physically meaningful neural network behavioral model for wireless transmitters exhibiting PM-AM/PM-PM distortions. **IEEE Transactions on Microwave Theory and Techniques**, v. 59, 12 2011. ISSN 00189480. DOI: 10.1109/TMTT.2011.2171709. Citado 1 vezes nas páginas 46, 47.

LIMA, Eduardo Gonçalves de. **Behavioral Modeling and Digital Base-Band Predistortion of RF Power Amplifiers**. 2009. Tese (Doutorado) – Politecnico di Torino. Citado 5 vezes nas páginas 19, 25–28, 64.

LIN, Luan et al. A novel efficient model for gas compressibility factor based on GMDH network. **Flow Measurement and Instrumentation**, v. 71, 2020. ISSN 09555986. DOI: 10.1016/j.flowmeasinst.2019.101677. Citado 1 vez na página 54.

LIU, Taijun; BOUMAIZA, Slim; GHANNOUCHI, Fadhel M. Dynamic behavioral modeling of 3G power amplifiers using real-valued time-delay neural networks. **IEEE Transactions on Microwave Theory and Techniques**, v. 52, 3 2004. ISSN 00189480. DOI: 10.1109/TMTT.2004.823583. Citado 2 vezes nas páginas 40, 46, 47.

MUHA, M. S. et al. Validation of power amplifier nonlinear block models. In: v. 2. DOI: 10.1109/mwsym.1999.779870. Citado 1 vez na página 35.

NGUYEN, Tan N.; NGUYEN-XUAN, H.; LEE, Jaehong. A novel data-driven nonlinear solver for solid mechanics using time series forecasting. **Finite Elements in Analysis and Design**, v. 171, 2020. ISSN 0168874X. DOI: 10.1016/j.finel.2019.103377. Citado 1 vez na página 54.

NITTA, Tohru. An extension of the back-propagation algorithm to complex numbers. **Neural Networks**, v. 10, 8 1997. ISSN 08936080. DOI: 10.1016/S0893-6080(97)00036-1. Citado 1 vez na página 48.

NITTA, Tohru. Orthogonality of Decision Boundaries in Complex-Valued Neural Networks. **Neural Computation**, v. 16, 1 2004. ISSN 08997667. DOI: 10.1162/08997660460734001. Citado 1 vez na página 48.

PAIXÃO RIBA, Otávio Augosto da. **Modelagem Comportamental de Amplificadores de Potência de Banda Dupla Baseada em Aproximações Bi-dimensionais e Produto de Tabelas de Busca**. 2007. Diss. (Mestrado) – Universidade Federal do Paraná. Citado 0 vez na página 26.

PEDRO, José C.; MAAS, Stephen A. **A comparative overview of microwave and wireless power-amplifier behavioral modeling approaches**. v. 53. [S.l.: s.n.], 2005. DOI: 10.1109/TMTT.2005.845723. Citado 1 vez na página 40.

PEDRO, José Carlos; CARVALHO, Nuno Borges; LAVRADOR, Pedro Miguel. Modeling nonlinear behavior of band-pass memory less and dynamic systems. In: v. 3. DOI: 10.1109/mwsym.2003.1210584. Citado 1 vez na página 27.

ROBLIN, Patrick et al. Concurrent linearization: The state of the art for modeling and linearization of multiband power amplifiers. **IEEE Microwave Magazine**, v. 14, 7 2013. ISSN 15273342. DOI: 10.1109/MMM.2013.2281297. Citado 1 vez na página 24.

SATTARI, Mohammad Amir; ROSHANI, Gholam Hossein; HANUS, Robert. Improving the structure of two-phase flow meter using feature extraction and GMDH neural network. **Radiation Physics and Chemistry**, v. 171, 2020. ISSN 18790895. DOI: 10.1016/j.radphyschem.2020.108725. Citado 1 vez na página 54.

SAVITHA, R. et al. A new learning algorithm with logarithmic performance index for complex-valued neural networks. **Neurocomputing**, v. 72, 16-18 2009. ISSN 09252312. DOI: 10.1016/j.neucom.2009.06.004. Citado 1 vez na página 50.

SURESH, Sundaram; SUNDARARAJAN, Narasimhan; SAVITHA, Ramasamy. **Supervised Learning with Complex-valued Neural Networks**. [S.l.: s.n.], 2013. DOI: 10.1007/978-3-642-29491-4_9. Citado 2 vezes nas páginas 48, 66.

XIAO, Jin et al. Circular Complex-Valued GMDH-Type Neural Network for Real-Valued Classification Problems. **IEEE Transactions on Neural Networks and Learning Systems**, 2020. ISSN 21622388. DOI: 10.1109/TNNLS.2020.2966031. Citado 1 vez na página 63.

YANG, Sheng-Sung; HO, Chia-Lu; SIU, Sammy. Sensitivity Analysis of the Split-Complex Valued Multilayer Perceptron Due to the Errors of the i.i.d. Inputs and Weights. **IEEE Transactions on Neural Networks**, v. 18, 5 set. 2007. ISSN 1045-9227. DOI: 10.1109/TNN.2007.894038. Citado 1 vez na página 50.

ZHANG, Huisheng; ZHANG, Chao; WU, Wei. Convergence of batch split-complex backpropagation algorithm for complex-valued neural networks. **Discrete Dynamics in Nature and Society**, v. 2009, 2009. ISSN 10260226. DOI: 10.1155/2009/329173. Citado 1 vez na página 50.