Pontifícia Universidade Católica
do Rio de Janeiro

Daniel Lerner

Uso de modelos de redes neurais artificiais para detecção de falhas no processo *Tennessee Eastman*

Dissertação de Mestrado

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Materiais e Processos Químicos e Metalúrgicos da PUC-Rio como requisito parcial para obtenção do grau de Mestre em Engenharia Química e de Materiais.

Orientador: Prof. Brunno Ferreira dos Santos

Rio de Janeiro

Agosto de 2018

Pontifícia Universidade Católica
do Rio de Janeiro

Daniel Lerner

Uso de modelos de redes neurais artificiais para detecção de falhas no processo Tennessee Eastman

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Materiais e Processos Químicos e Metalúrgicos da PUC-Rio como requisito parcial para obtenção do grau de Mestre em Engenharia Química e de Materiais. Aprovada pela Comissão Examinadora abaixo assinada.

Prof. Brunno Ferreira dos Santos
Orientador
Departamento de Engenharia Química e de Materiais – PUC-Rio

Prof[a]. Amanda Lemette Teixeira Brandão
Departamento de Engenharia Química e de Materiais – PUC-Rio

Prof. Tiago Dias Martins
Departamento de Engenharia Química – UNIFESP

Prof. Marcio da Silveira Carvalho
Coordenador(a) Setorial do Centro Técnico Científico - PUC-Rio

Rio de Janeiro, 30 de agosto de 2018

Daniel Lerner

Graduou-se em Engenharia Química pela Pontifícia Universidade Católica do Rio de Janeiro (Rio de Janeiro, Brasil).

Ficha Catalográfica

Lerner, Daniel

Uso de modelos de redes neurais artificiais para detecção de falhas no processo Tennessee Eastman / Daniel Lerner ; orientador: Brunno Ferreira dos Santos – 2018.

89 f. : il. color. ; 30 cm

Dissertação (mestrado)–Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Engenharia Química e de Materiais, 2018.

Inclui bibliografia

1. Engenharia Química e de Materiais – Teses. 2. Detecção de falhas. 3. Processo Tennessee Eastman. 4. Redes neurais artificiais. 5. Rede de Elman. 6. Echo state network. I. Santos, Brunno Ferreira dos. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Engenharia Química e de Materiais. III. Título.

CDD: 620.11

Agradecimentos

Aos meus pais que tanto amo, Miguel e Marcia, por me educarem com muita dedicação e carinho, sempre acreditando no meu potencial e apoiando minhas decisões.

Ao meu irmão David, pelas conversas e jantares como escapatória de momentos mais estressantes.

À minha namorada Gisele, com quem amo compartilhar a vida, nos momentos alegres e tristes. Obrigado por estar sempre lá para me ajudar, seja na realização deste trabalho ou em ensinamentos do dia a dia.

Ao Professor Brunno Ferrira dos Santos, pela excelente orientação e paciência na realização deste trabalho. Além dos ensinamentos, incentivos e segurança transmitida, fatores que me fortaleceram como profissional e pessoa.

Aos meus amigos de infância, Raul, Rô, Manela, Acher, Ilann, Alan, Bila, Debora, Mix, Vax, pelas cervejas, altinhas, viagens e lembranças inesquecíveis.

Aos meus amigos de PUC-Rio, Hauesler, Pedrinho, Ruas, Braço, Diego, Julia, Rafa, Marcelle, pelas risadas e choros que compartilhamos nas aulas. Em particular, aqueles que pertecem ao meu grupo de pesquisa ou que ajudaram de alguma forma durante o mestrado, Isabelle, Serpa, Zanone, Juliana, Carol e Vitinho.

Aos amigos que a vida me trouxe, Melgaço, Gewerc, Galc, Kadinho, Higor, Eric, Michel, Spritzer, Ana, Sabrina, Clarice, Vitória.

À todos aqueles que não foram citados aqui, mas que contribuíram de alguma forma ao longo deste caminho.

Por último, gostaria de agradecer à PUC-Rio pelos anos maravilhosos que passei nessa universidade, a qual posso chamar de minha segunda casa.

Resumo

Lerner, Daniel; Santos, Brunno Ferreira dos; **Uso de modelos de redes neurais artificiais para detecção de falhas no processo Tennessee Eastman.** Rio de Janeiro, 2018. 89p. Dissertação de Mestrado - Departamento de Engenharia de Materiais e de Processos Químicos e Metalúrgicos, Pontifícia Universidade Católica do Rio de Janeiro.

A humanidade está vivenciando a 4ª Revolução Industrial, caracterizada pela implementação global da internet, utilização de inteligência artificial e automatização dos processos. Este último é de grande importância para indústria química, uma vez que seu desenvolvimento possibilitou um aumento significativo da quantidade de dados armazenados diariamente, o que gerou uma demanda para análise desses dados. Este enorme fluxo de informações tornou o sistema cada vez mais complexo com uma aleatoriedade de falhas no processo que se identificadas poderiam ajudar a melhorar o processo e evitar acidentes. Uma solução ainda pouco comum na indústria, porém com grande potencial para identificar estas falhas de processo com excelência, é a emergente inteligência artificial. Para lidar com esta questão, o presente trabalho realiza a detecção e identificação de falhas em processos industriais através da modelagem de redes neurais artificias. O banco de dados foi obtido através do uso do *benchmark* de processo *Tennessee Eastman*, implementado no Software *Matlab* 2017b, o qual foi projetado para simular uma planta química completa. A enorme quantidade de dados gerados pelo processo tornou possível a simulação em um contexto de *Big Data*. Para modelagem dos dados, foram tanto aplicadas redes neurais tradicionais *feedforward*, quanto redes recorrentes: Rede de Elman e *Echo State Network*. Os resultados apontaram que as redes *feedforward* e de *Elman* obtiveram melhores desempenhos analisados pelo coeficiente de determinação (R^2). Assim, o primeiro modelo obteve melhor topologia com 37x60x70x1, algoritmo de treinamento *trainlm*, funções de ativação *tansig* para as duas camadas intermediárias e camada de saída ativada pela *purelin* com R^2 de 88,69%. O modelo da rede de *Elman* apresentou sua melhor topologia com 37x45x55x1, algoritmo de treinamento *trainlm*, funções de ativação *tansig* para as duas camadas intermediárias e camada de saída ativada pela função *purelin* com R^2 de 83,63%. Foi concluido que as redes analisadas podem ser usadas em controle preditivo de falhas em processos industriais, podendo ser aplicadas em plantas químicas no futuro.

Palavras-chave

Detecção de falhas; Processo *Tennessee Eastman*; Redes Neurais Artificiais; Rede de *Elman*; *Echo State Network*;

Abstract

Lerner, Daniel; Santos, Brunno Ferreira dos (Advisor); **Use of artificial neural network models for fault detection and diagnosis of Tennessee Eastman Process.** Rio de Janeiro, 2018. 89p. Dissertação de Mestrado - Departamento de Engenharia de Materiais e de Processos Químicos e Metalúrgicos, Pontifícia Universidade Católica do Rio de Janeiro.

Humanity is experiencing the 4th Industrial Revolution, characterized by the global implementation of the internet, use of artificial intelligence and automation of processes. The last one is of great importance for the chemical industry, since its development allowed a significant increase in the amount of data stored daily, which generated a demand for the analysis of this data. This enormous flow of information made the system more and more complex with a randomness of process faults that if identified could help improve the process and prevent accidents. A solution not yet common in industry, but with great potential to identify these process faults with excellence, is the emergent artificial intelligence. To deal with this issue, the present work performs fault detection and diagnosis in industrial processes through artificial neural networks modeling. The database was obtained using the benchmark of processes Tennessee Eastman, implemented in *Matlab* 2017b Software, which is designed to simulate a complete chemical plant. The huge amount of data generated by the process made it possible to simulate in a Big Data context. For data modeling, were applied both traditional feedforward neural networks as well as recurrent networks: Elman Network and Echo State Network. The results indicated that the feedforward and Elman networks obtained better performances analyzed by the determination coefficient (R^2). Thus, the first model obtained the best topology with 37x60x70x1, *trainlm* as training algorithm, *tansig* as activation functions for the two intermediate layers and output layer activated by the *purelin* function with R^2 of 88.69%. The Elman network model presented its best topology with 37x45x55x1, *trainlm* as training algorithm, *tansig* as activation functions for the two intermediate layers and output layer activated by *purelin* function with R^2 of 83.63%. It was concluded that the analyzed networks can be used in predictive control of fault in industrial processes and can be applied in chemical plants in the future.

Keywords

Fault Detction; *Tennessee Eastman* Process; Artificial Neural Networks; Elman Network; Echo State Network;

Sumário

Lista de imagens

Lista de tabelas

Abreviaturas e Siglas

A – Substância química gasosa hipotética

B – Substância química gasosa inerte hipotética

BP – *Backpropagation*

C – Substância química gasosa hipotética

D – Substância química gasosa hipotética

E – Substância química gasosa hipotética

ESN – *Echo State Network*

G – Substância química líquida hipotética

GA – Algoritmo genético

F – Substância química líquida hipotética

H – Substância química líquida hipotética

IA – Inteligência artificial

RC – *Reservoir computing*

RNA – Rede Neural Artificial

TE – *Tennessee Eastman*

VBA – *Visual Basic*

XMEAS – Variável medida

XMV – Variável controlada

Tudo que temos de decidir é o que fazer com o tempo que nos é dado.

Gandalf, *O Senhor dos Anéis.*

1. Introdução

1.1. Motivação

Com o passar dos anos as indústrias sofreram transformações em suas formas de atuação, o que reverberou em inovações e reestruturação social e econômica. A primeira revolução industrial ficou caracterizada pela utilização do carvão, usado como fonte de energia, e pelo uso de máquinas, em substituição a produções artesanais. A segunda, por sua vez, teve como grande marco a descoberta do eletromagnetismo e o petróleo como nova fonte de energia. Já a terceira, o uso intenso de eletrônicos e da Tecnologia de Informação (TI) foi uma revelação que permitiu a automação da manufatura. O que culminou na quarta revolução industrial ou Revolução 4.0, em andamento atualmente, trazendo ao mundo a internet, a inteligência artificial (IA) e a automatização dos processos (AZEVEDO, 2017).

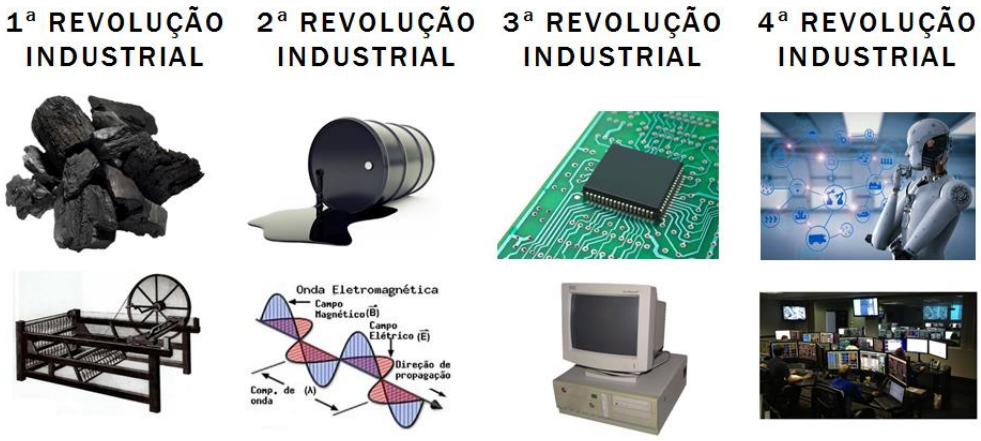


Figura 1: Inovações que influenciaram em cada uma das revoluções industriais (PONTO FINAL MACAU, 2017; CITANDO MODA SITE BLOG, 2017; QUATRO RODAS, 2016; SUPER TESLAS, 2014; SHUTTER STOCK, 2018; PENSAMENTO VERDE, 2013; REAMP, 2018; GRUPO MULT, 2018).

A revolução 4.0 gerou avanços tecnológicos em diversas áreas, particularmente na indústria química esta se manifestou através da automatização dos processos devido ao barateamento de sensores e estocagem de dados e do aumento da instrumentação disponível. Isso aumentou muito a quantidade de dados históricos e on-line disponíveis, gerando uma demanda para analise desses dados (SOARES, 2017). Devido a este enorme fluxo de dados gerado diariamente, os sistemas se tornam cada vez mais complexos e geram problemas difíceis de resolver, ainda mais com a aleatoriedade de falhas no processo.

O comportamento anormal do processo pode estar relacionado a um erro de equipamento, condições de processo ou ainda um equívoco de um operador. A detecção de falhas indica quando esta ocorreu, enquanto a identificação de falhas diagnostica de onde ele é proveniente e qual seu tipo de falha (RAD, *et al.* 2015). Estas são uma das principais preocupações dos engenheiros de processo e operadores de instalações, pois podem ajudar a melhorar o processo, aumentar a produtividade da planta, aumentar o tempo de vida útil dos equipamentos e evitar acidentes com consequências ao meio ambiente e a vida de seres humanos (LAU, *et al.* 2012).

Uma solução ainda pouco comum na indústria química, porém com grande potencial para identificar estas falhas de processo com excelência, é a emergente inteligência artificial. Apesar do termo da tecnologia ter sido detalhada pela primeira vez em 1955 pelo professor John McCarthy durante um congresso em New Hampshire, ela só se tornou mundialmente utilizável com a chegada da Revolução 4.0 (MCCARTHY, 1955). Para a indústria química, a técnica de inteligência artificial com maior potencial de aderência, é a Rede Neural artificial (RNA), isso devido à sua habilidade de aprender a partir das experiências, melhorando seu desempenho e adaptando-se às mudanças no ambiente, tornando possível a leitura e modelagem dos dados de um processo mesmo em um cenário de *Big Data*. Trabalhos como GAO e OVASKA (2002), LAU (2012) e JAMIL (2015) apresentaram bons resultados para RNA e por isso motivam pesquisadores e engenheiros de processo ao seu estudo e utilização.

Para modelar um processo industrial é necessário um extenso banco de dados com informações dos mais diversos processos e variáveis, felizmente em 1993 DOWNS e VOGEL propuseram o problema de controle *Tennessee Eastman* (TE) que mais tarde se tornou um *benchmark* em estudo de controle e simulação

de processo. Baseado em um problema de um processo industrial real, o TE é capaz de simular uma planta química completa, tornando-a uma ótima ferramenta para detecção e identificação de falhas de processo.

1.2. Objetivo

O objetivo do presente trabalho é o estudo de modelos preditivos por redes neurais artificiais para detecção e identificação de falhas em uma planta química completa através do processo *Tennessee Eastman*.

Os objetivos específicos do trabalho foram:

- Estudar o benchmark de processo Tennessee Eastman para geração do conjunto de dados num contexto de *Big Data*, característico de uma planta química;
- Utilizar algoritmos de *Levenberg Marquardt* e regularização *Bayesiana* para treinamento de rede neural *feedforward;*
- Utilizar algoritmos de *Levenberg Marquardt* e regularização *Bayesiana* para treinamento de rede neural de *Elman;*
- Utilizar *Echo State Network* para modelar o conjunto de dados.
- Comparar o desempenho dos diferentes modelos neurais

1.3. Estrutura da Dissertação

A dissertação está organizada em sete capítulos. O primeiro capítulo introduz a necessidade de automatização de processos nas indústrias químicas, que pode ser satisfeita através da modelagem por meio de técnicas de redes neurais e a simulação da planta química realizada através do processo *Tennessee Eastman*. Também apresenta a motivação e objetivo do trabalho.

No capítulo 2, os conceitos estudados no trabalho são embasados através de uma revisão bibliográfica. Primeiramente se esclarece o conceito de detecção e identificação de falha, assim como o *benchmark* de processo *Tennessee Eastman*. Em seguida, entra em tema o tipo de inteligência artificial utilizada no trabalho para modelagem de dados, as redes neurais artificiais. Estas são definidas e sua estrutura, topologia e treinamento são explicados, adentrando as concepções de rede de *Elman*, *Echo State Network* e algoritmo de *Levenberg Marquardt* e regularização *Bayesiana*.

O capítulo 3 traz os materiais e métodos utilizados na dissertação. Aqui se explica como foi construído o banco de dados por meio do problema TE, quais foram as configurações das redes que modelaram o processo e quais métodos foram utilizados para avaliar o desempenho destas.

No capítulo 4, são expostos os resultados das diversas RNAs estudadas, bem como a discussão dos mesmos. Também são apresentados métodos para melhorar a leitura das redes.

O capítulo 5 serve de conclusão para a dissertação ao formular as devidas considerações e sugerir possíveis melhorias para trabalhos futuros.

E finalmente são apresentadas as referências utilizadas como base neste trabalho. Ainda, uma seção adicional para anexos se encontram ao fim.

2. Revisão Bibliográfica

Neste capítulo são abordados os conceitos fundamentais a respeito de detecção e identificação de falhas em processos industriais, o processo Tennessee Eastman, o funcionamento das redes neurais *feedforward* e recorrentes, Elman e Echo State, e algoritmo de *Levenberg Marquardt* e regularização *Bayesiana*.

2.1. Big Data

Com a chegada da Revolução 4.0, o custo de armazenamento, sensores e comunicação abaixou muito, levando a um bombardeamento de dados em todos os setores (SRINIVASA e BHATNAGAR, 2012). Para tratar de conjuntos de dados muito grandes e complexos, o analista Doug Laney, da companhia de pesquisa e consultoria Gartner, criou o conceito de *Big Data*. Segundo Laney, o *Big Data* é um grande volume, alta velocidade e/ou grande variedade de informações que exigem formas inovadoras e econômicamente efetivas de processamento de informações que permitem uma melhor percepção, tomada de decisões e automação de processos (GARTNER).

O potencial do *Big Data* é incalculável. Suas funções incluem coleta, armazenamento, pré-processamento, visualização e, essencialmente, análise estatística de dados (TORRECILLA e ROMO, 2018). Em particular na indústria química, ele é usado para analisar os dados de processo, visando o aperfeiçoamento do mesmo.

2.2. Detecção e identificação de falhas

Conforme explicado, a indústria química passou a investir na automatização de seus processos, visando à detecção e identificação de falhas para melhorar o processo e evitar acidentes que envolvam o meio ambiente ou seres humanos. Falhas são consideradas como alterações de estado não autorizadas de pelo menos uma propriedade característica do sistema de seu comportamento aceitável, usual

ou da condição padrão, que podem levar a avarias e erros (ISERMANN, 2013; GERMANO; 2016).

As falhas podem ser causadas por erros de equipamentos, condições de processo ou ainda por operadores. A detecção de uma falha apenas aponta que um erro aconteceu, mas quando a identificação de falhas trabalha em conjunto, este erro também é diagnosticado, apresentando o local e motivo da falha. A figura 2, apresenta uma pesquisa realizada na literatura contabilizando ano a ano, nos últimos 15 anos, o número de trabalhos que estão relacionados à detecção de falhas (*fault detection*), sendo possível identificar a crescente pesquisa por este tema.

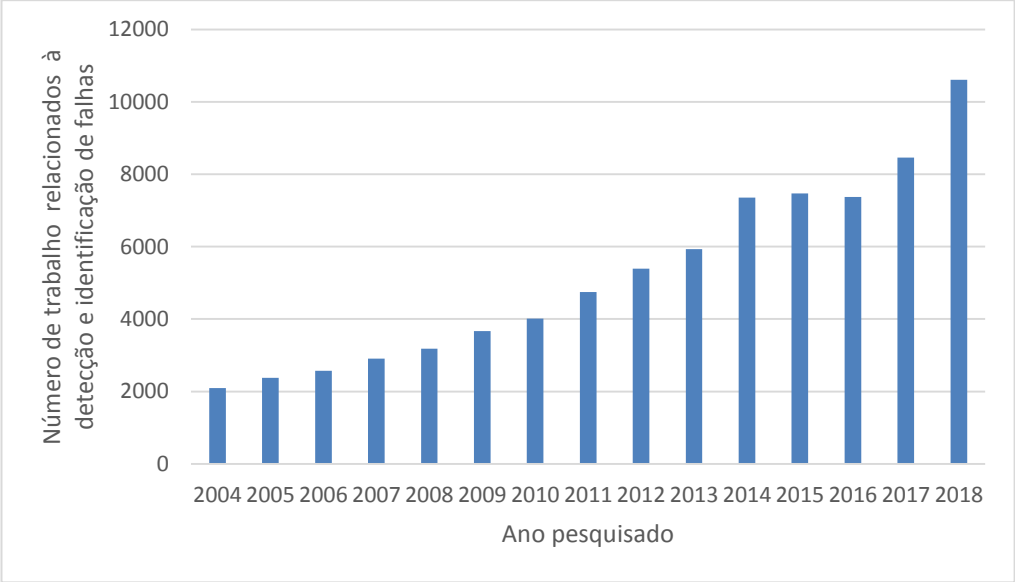


Figura 2: Pesquisa realizada no CAPES sobre a quantidade de trabalhos relacionados à detecção de falhas.

A análise de uma planta química é um problema de alto nível de dificuldade devido a sua complexidade. O ideal para detecção e identificação de falhas em processos industriais seria trabalhar com dados de falhas gerados em um processo industrial real, contudo companhias não costumam disponibilizar publicamente essas informações por motivos de confidencialidade do processo (GERMANO *et al.*, 2016). Para suprir estas finalidades, foi criado o processo *Tennesse Eatsman*, o qual será utilizado como estudo de caso no presente trabalho.

2.3. Principais técnicas de identificação de falhas

Neste tópico serão apresentadas as principais técnicas encontradas na literatura para detectar e identificar falhas em processos industriais e trabalhos que a utilizaram.

Uma das técnicas mais utilizadas é o *principal componente analysis* (PCA), o PCA é uma técnica de redução de dimensionalidade ideal em termos de captura da variação dos dados. O PCA determina um conjunto de vetores ortogonais, chamados de Índices de carga, que podem ser ordenados pela quantidade de variância explicada nas direções do vetor de carregamento (CHIANG, *et al.* 1999). Os métodos de PCA podem ser estendidos para levar em conta as correlações em série, aumentando cada vetor de observação com as observações anteriores e empilhando a matriz de dados. Esta abordagem de aplicação de PCA é referida aqui como PCA dinâmico (DPCA). Já o *canonical variate analysis* (CVA) é uma técnica de redução de dimensionalidade que é ótima em termos de maximizar uma estatística de correlação entre dois conjuntos de variáveis (RUSSEL, *et al.* 2000).

O *Fisher's discriminant analysis* (FDA) é uma técnica de redução de dimensionalidade ideal em termos de maximizar a separabilidade de tipos de falha. Ele determina um conjunto de vetores de projeção que maximizam a dispersão entre as classes enquanto minimizam a dispersão dentro de cada classe. Outro método utilizado é o *discriminant partial least squares* (DPLS) é uma técnica de redução de dimensionalidade para maximizar a covariância entre o preditor e o previsto para cada componente. O DPLS modela esse relacionamento usando uma série de ajustes de mínimos quadrados locais (CHIANG, *et al.* 1999).

O *signed directed graph-based* (SDG) pode ser usado para descrever o comportamento do processo sob várias condições normais e anormais. Eles essencialmente capturam o fluxo de informações na relação causa-efeito e a direção do efeito. Uma das vantagens mais importantes do uso de modelos baseados em digraph é que eles não exigem muita informação quantitativa (MAURYA, *et al.* 2004). Por sua vez, a metodologia *Multi-scale principal componente analysis (*MSPCA) consiste em decompor cada variável com base na função de onduletas. O modelo PCA é então determinado independentemente para os coeficientes em cada escala. Os modelos em escalas importantes são então

combinados de uma maneira recursiva em escala eficiente para produzir o modelo em escala múltipla.

A lógica *fuzzy* é um conjunto de regras SE-ENTÃO que têm capacidade de aprendizado para aproximar funções não-lineares. Quando este conceito é combinado com redes neurais artificiais (que será detalhado ao longo do presente trabalho), se forma o sistema neuro-*fuzzy* (ANFIS), que tem o potencial para capturar os benefícios de ambos em um único sistema (LAU, *et al.* 2012). A seguir, serão trazidos os principais trabalhos da literatura a usar estas técnicas.

CHIANG *et al.* (1999) desenvolveram um critério de informação que determina automaticamente a ordem de redução de dimensionalidade para análise discriminante de *Fisher* e para mínimos quadrados parciais discriminativos para identificar falhas em processos químicos, com a finalidade de avaliá-los em relação a análise de componentes principais. Os dados foram coletados pelo *benchmark* de processo *Tennessee Eastman*.

RUSSEL *et al.* (2000) aplicaram três técnicas de dimensionalidade para detectar falhas durantes operações de processos industriais. A primeira foi a altamente utilizada PCA. As outras duas foram as técnicas que ainda levam em conta as correlações em série: a análise de componentes principais dinâmico e a análise de variáveis canônicas. O simulador de processos TE é novamente utilizado para gerar o banco de dados.

MAURYA *et al.* (2004) utilizaram dois estudos de caso para analisar o *signed directed graph-based* em nível de fluxograma de processos. O primeiro estudo de caso trata da previsão da resposta inicial e da sua aplicação em identificação de falhas no fluxograma TE usando um modelo de parâmetro agrupado do processo. O segundo caso analisa o estado estacionário e detecção de falhas de um processo de reação-separação.

LAU *et al.* (2012) desenvolveram uma estrutura de diagnóstico de falhas on-line para um processo dinâmico incorporando análise de componentes principais multi-escala para extração de recursos e sistema de inferência neuro-fuzzy adaptativo para aprender a correlação de sintomas de falhas dos dados históricos do processo. A estrutura MSPCA-ANFIS proposta é testada no processo *Tennessee Eastman*.

RATO e REIS (2013) desenvolveram um conjunto de estatísticas multivariadas baseadas na DPCA e na geração de resíduos decorrelados, que

apresentam baixos níveis de autocorrelação e, portanto, melhor posicionamento para implementar o controle estatístico de processos (SPC) de forma mais consistente e estável (DPCA-DR). O desempenho do monitoramento dessas estatísticas foi comparado com o de outras metodologias alternativas para o conhecido *benchmark* do processo TE.

MORELLO *et al.* (2015) propôs a utilização de um algoritmo denominado STRASS, que visa detectar os recursos mais relevantes para cada tipo de falha de processos, pois a presença de recursos irrelevantes pode afetar o desempenho do classificador de falhas. O algoritmo usa a correlação *k-way* entre recursos e classes para selecionar recursos relevantes. Os dados são coletados através do problema TE.

RAD e YAZDANPANAH (2015) propõem um classificador de perceptron multicamada local supervisionado (SLMLP) integrado com modelos de análise de componentes independentes (ICA) para detecção e identificação de falhas de sistemas industriais. O interesse deste trabalho é melhorar o desempenho da rede neural única (SNN) dividindo o espaço do padrão de falhas em alguns sub-espaços menores utilizando a técnica de agrupamento *Expectation-Maximization* (EM) e desencadeando o classificador local correto ao criar um agente supervisor.

GERMANO *et al.* (2016) utiliza o *Typicality and Eccentricity Data Analytics* (TEDA) para detectar falhas em processos industriais usando o clássico *benchmark* de processo TE.

2.4. Problema Tennessee Eastman

O processo *Tennessee Eastman* é um problema de controle proposto pela *Eastman Chemical Company*, baseado em um processo industrial real. Mesmo modificando os componentes, a cinética, o processo e as condições de operação para proteger a natureza proprietária do processo, o TE se tornou um *benchmark* em estudo de controle e simulação de processo, por ser um modelo não-linear de um sistema de multicomponentes bastante complexo (DOWNS e VOGEL, 1993).

BATHELT *et al.* (2015) propuseram uma implementação em *Matlab* do problema TE ao estabelecer alterações ao modelo original, adicionando mais perturbações, melhorando a reprodutibilidade e aumentando o número de medidores do sistema. A figura 3 ilustra o processo atualizado.

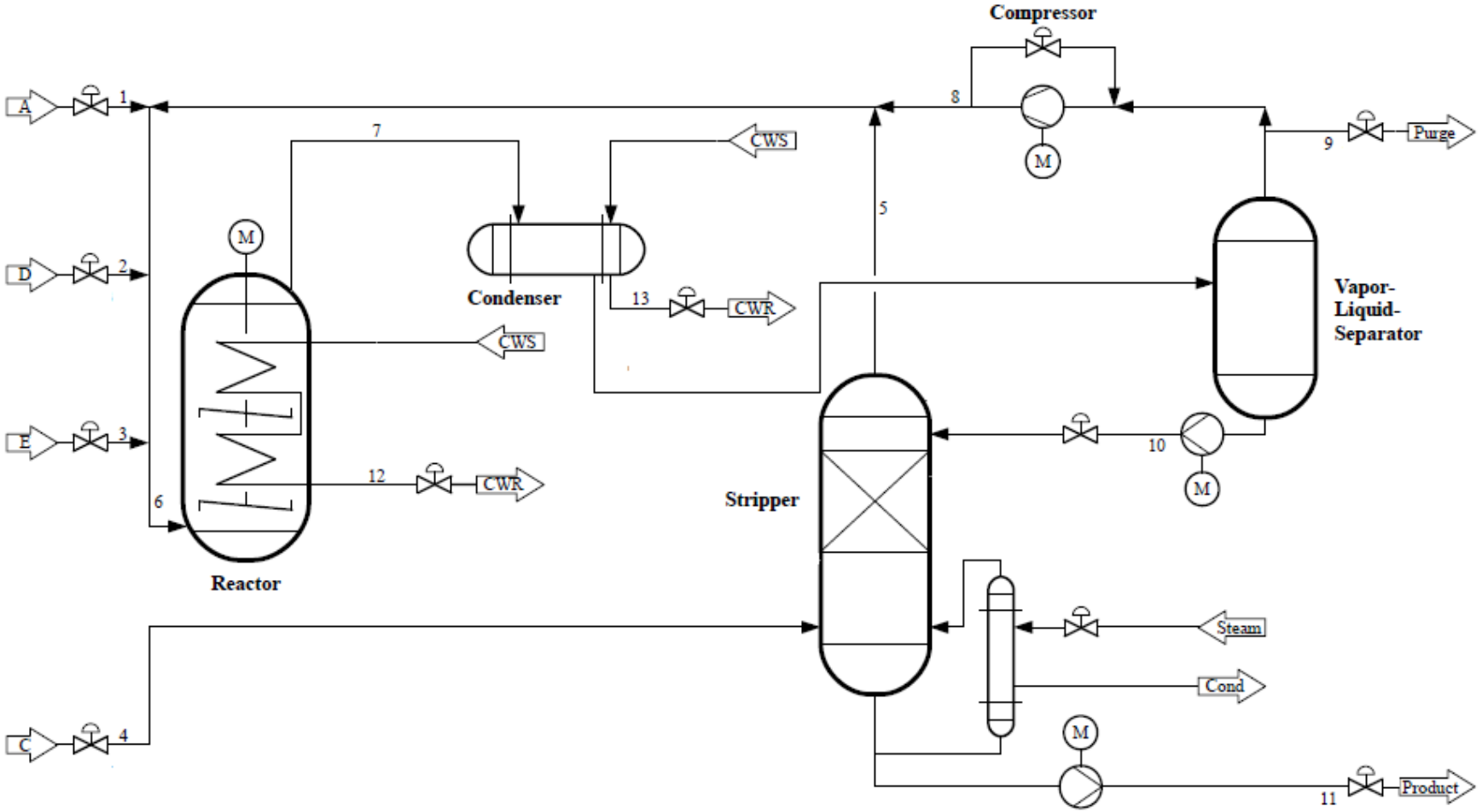


Figura 3: Fluxograma atualizado do processo *Tenessee Eastman.*

O processo é composto por um reator e quatro operações unitárias principais: condensador, compressor, separador e *stripper*. Os reagentes gasosos A, C, D e E e o inerte B são alimentados ao reator, onde são formados os produtos líquidos G e H, assim como o subproduto F. As reações são:

$$A_{(g)} + C_{(g)} + D_{(g)} \rightarrow G_{(liq)} \quad (1)$$

$$A_{(g)} + C_{(g)} + E_{(g)} \rightarrow H_{(liq)} \quad (2)$$

$$A_{(g)} + E_{(g)} \rightarrow F_{(liq)} \quad (3)$$

$$3D_{(g)} \rightarrow 2F_{(liq)} \quad (4)$$

As reações são irreversíveis, exotérmicas e aproximadamente de primeira ordem em relação às concentrações dos reagentes. As taxas de reação seguem a expressão de Arrhenius. A energia de ativação para produzir G é maior do que a da reação de H. As reações são catalisadas por um catalisador não-volátil dissolvido na fase líquida.

O reator é equipado com uma serpentina de arrefecimento, para remover o calor da reação, e um agitador com velocidade manipulável. A saída do reator é condensada e, em seguida, alimentada em um separador vapor-líquido. A saída de topo do separador é reciclada para o reator através de um compressor, com certa fração purgada para evitar acúmulo de subproduto e inerte no processo. A saída de fundo do separador move-se para um *stripper*, que é alimentado pela corrente 4 utilizada como corrente de vapor para remover os reagentes restantes, reciclando-os em direção ao reator. Os produtos saem na base do *stripper* e são separados em uma seção de refino que não está incluída no problema (DOWNS & VOGEL, 1993).

No problema TE, existem seis modos de operação de processo em três taxas de massa G/H diferentes, conforme apresentado na tabela 1. Neste trabalho apenas o modo 1 foi utilizado por ser o modo padrão do processo.

Tabela 1: Modos de operação do *Tennesse Eastman*.

Modo	Razão G/H	Taxa de produção
1	50/50	7038 kg/h G&H
2	10/90	1048 kg/h G & 12669 kg/h H
3	90/10	10000 kg/h G & 1111 kg/h H
4	50/50	Máximo
5	10/90	Máximo
6	90/10	Máximo

O processo tem um total de 85 variáveis, separadas em dois blocos: variáveis manipuláveis (XMV) e variáveis medidas (XMEAS). As variáveis manipuláveis se encontram na tabela 2.

Tabela 2: Variáveis manipuláveis do *Tennessee Eastman*.

Variável	Descrição	Unidade
XMV(1)	Vazão de entrada de D (corrente 2)	kg/h
XMV(2)	Vazão de entrada de E (corrente 3)	kg/h
XMV(3)	Vazão de entrada de A (corrente 1)	kscmh
XMV(4)	Vazão de entrada de total (corrente 4)	kscmh
XMV(5)	Válvula de reciclo do compressor	%
XMV(6)	Válvula de purga (corrente 9)	%
XMV(7)	Vazão de saída de líquido do separador (corrente 10)	m^3/h
XMV(8)	Saída de líquido do stripper (corrente 11)	m^3/h
XMV(9)	Válvula do vapor do stripper	%
XMV(10)	Vazão da água de resfriamento do reator	m^3/h
XMV(11)	Vazão da água de resfriamento do condensador	m^3/h
XMV(12)	Velocidade do agitador	rpm

As variáveis referentes a XMEAS (1) a XMEAS (41) são originais do processo criado por DOWNS e VOGEL (1993). A tabela 3 apresenta as medições contínuas do processo (como temperaturas, pressões, vazão, etc), enquanto a tabela 4 traz as medidas de composição dos componentes com tempo morto equivalente ao intervalo de amostragem.

Tabela 3: Variáveis medidas continuamente do *Tennessee Eastman*.

Variável	Descrição	Unidade
XMEAS(1)	Entrada de A (corrente 1)	kscmh
XMEAS(2)	Entrada de D (corrente 2)	kg/h
XMEAS(3)	Entrada de E (corrente 3)	kg/h
XMEAS(4)	Entrada de total (corrente 4)	kscmh
XMEAS(5)	Corrente de reciclo (corrente 8)	kscmh
XMEAS(6)	Entrada do reator (corrente 6)	kscmh
XMEAS(7)	Pressão do reator	kpa
XMEAS(8)	Nível do reator	%
XMEAS(9)	Temperatura do reator	^{0}C
XMEAS(10)	Vazão de purga (corrente 9)	kscmh
XMEAS(11)	Temperatura do separador	^{0}C
XMEAS(12)	Nível do separador	%
XMEAS(13)	Pressão do separador	kPa
XMEAS(14)	Corrente de fundo do separador (corrente 10)	m^3/h
XMEAS(15)	Nível do stripper	%
XMEAS(16)	Pressão do stripper	kPa
XMEAS(17)	Vazão de fundo do stripper	m^3/h
XMEAS(18)	Temparatura do stripper	^{0}C
XMEAS(19)	Vazão de vapor do stripper	kg/h
XMEAS(20)	Trabalho do compressor	kw
XMEAS(21)	Temperatura da saída da água de resfriamento do reator	^{0}C
XMEAS(22)	Temperatura da saída da água de resfriamento do condensador	^{0}C

Tabela 4: Variáveis medidas da composição dos componentes do *Tennessee Eastman*.

Variável	Descrição	Corrente	Intervalo de amostragem (minutos)	Unidade
XMEAS(23)	Componente A	6	6	% molar
XMEAS(24)	Componente B	6	6	% molar
XMEAS(25)	Componente C	6	6	% molar
XMEAS(26)	Componente D	6	6	% molar
XMEAS(27)	Componente E	6	6	% molar
XMEAS(28)	Componente F	6	6	% molar
XMEAS(29)	Componente A	9	6	% molar
XMEAS(30)	Componente B	9	6	% molar
XMEAS(31)	Componente C	9	6	% molar
XMEAS(32)	Componente D	9	6	% molar
XMEAS(33)	Componente E	9	6	% molar
XMEAS(34)	Componente F	9	6	% molar
XMEAS(35)	Componente G	9	6	% molar
XMEAS(36)	Componente H	9	6	% molar
XMEAS(37)	Componente D	11	15	% molar
XMEAS(38)	Componente E	11	15	% molar
XMEAS(39)	Componente F	11	15	% molar
XMEAS(40)	Componente G	11	15	% molar
XMEAS(41)	Componente H	11	15	% molar

As variáveis de XMEAS (42) a XMEAS (73) foram adicionadas ao modelo atualizado por BATHELT *et al.* (2015) e são apresentadas na tabela 5.

Tabela 5: Variáveis medidas adicionadas ao *Tennessee Eastman*.

Variável	Descrição	Unidade
XMEAS(42)	Temperatura da entrada de A (corrente 1)	^{0}C
XMEAS(43)	Temperatura da entrada de D (corrente 2)	^{0}C
XMEAS(44)	Temperatura da entrada de E (corrente 3)	^{0}C

XMEAS(45)	Temperatura da entrada de total (corrente 4)	^{0}C
XMEAS(46)	Temperatura de entrada da água de resfriamento do reator	^{0}C
XMEAS(47)	Vazão da água de resfriamento do reator	m^3/h
XMEAS(48)	Temperatura de entrada da água de resfriamento do condensador	^{0}C
XMEAS(49)	Vazão da água de resfriamento do condensador	m^3/h
XMEAS(50)	Composição de A na corrente 1	% molar
XMEAS(51)	Composição de B na corrente 1	% molar
XMEAS(52)	Composição de C na corrente 1	% molar
XMEAS(53)	Composição de D na corrente 1	% molar
XMEAS(54)	Composição de E na corrente 1	% molar
XMEAS(55)	Composição de F na corrente 1	% molar
XMEAS(56)	Composição de A na corrente 2	% molar
XMEAS(57)	Composição de B na corrente 2	% molar
XMEAS(58)	Composição de C na corrente 2	% molar
XMEAS(59)	Composição de D na corrente 2	% molar
XMEAS(60)	Composição de E na corrente 2	% molar
XMEAS(61)	Composição de F na corrente 2	% molar
XMEAS(62)	Composição de A na corrente 3	% molar
XMEAS(63)	Composição de B na corrente 3	% molar
XMEAS(64)	Composição de C na corrente 3	% molar
XMEAS(65)	Composição de D na corrente 3	% molar
XMEAS(66)	Composição de E na corrente 3	% molar
XMEAS(67)	Composição de F na corrente 3	% molar
XMEAS(68)	Composição de A na corrente 4	% molar
XMEAS(69)	Composição de B na corrente 4	% molar
XMEAS(70)	Composição de C na corrente 4	% molar
XMEAS(71)	Composição de D na corrente 4	% molar
XMEAS(72)	Composição de E na corrente 4	% molar
XMEAS(73)	Composição de F na corrente 4	% molar

Além das variáveis, o problema original ainda contém 20 falhas e a versão de BATHELT *et al.* (2015) disponibiliza outras 8. Elas são adicionadas ao processo com entradas do modelo, um vetor com booleanos e numéricos. A tabela 6 apresenta as perturbações do processo.

Tabela 6: Perturbações programadas do *Tennessee Eastman*.

Perturbação	Descrição	Tipo
IDV(1)	Razão de entrada A/C composição de B constante (corrente 4)	Degrau
IDV(2)	Composição de B, razão A/C constante (corrente 4)	Degrau
IDV(3)	Temperatura de entrada de D (corrente 2)	Degrau
IDV(4)	Temperatura de entrada da água de resfriamento do reator	Degrau
IDV(5)	Temperatura de entrada da água de resfriamento do condensador	Degrau
IDV(6)	Perda de entrada de A (corrente 1)	Degrau
IDV(7)	Queda de pressão de C, disponibilidade reduzida (corrente 4)	Degrau
IDV(8)	Composição da entrada de A, B e C (corrente 4)	Variação randômica
IDV(9)	Temperatura de entrada de D (corrente 2)	Variação randômica
IDV(10)	Temperatura de entrada de C (corrente 4)	Variação randômica
IDV(11)	Temperatura de entrada da água de resfriamento do reator	Variação randômica
IDV(12)	Temperatura de entrada da água de resfriamento do condensador	Variação randômica
IDV(13)	Cinética da reação	Desvio lento

IDV(14)	Válvula de água de resfriamento do reator	Agarramento
IDV(15)	Válvula de água de resfriamento do condensador	Agarramento
IDV(16)	Desconhecido	
IDV(17)	Desconhecido	
IDV(18)	Desconhecido	
IDV(19)	Desconhecido	
IDV(20)	Desconhecido	
IDV(21)	Temperatura de entrada de A (corrente 1)	Variação randômica
IDV(22)	Temperatura de entrada de E (corrente 3)	Variação randômica
IDV(23)	Pressão de entrada de A (corrente 1)	Variação randômica
IDV(24)	Pressão de entrada de D (corrente 2)	Variação randômica
IDV(25)	Pressão de entrada de E (corrente 3)	Variação randômica
IDV(26)	Pressão de entrada de A e C (corrente 4)	Variação randômica
IDV(27)	Flutuação de pressão da unidade de recirculação de água de resfriamento do reator	Variação randômica
IDV(28)	Flutuação de pressão da unidade de recirculação de água de resfriamento do condensador	Variação randômica

2.5. Inteligência Artificial

O ser humano moderno vive rodeado pela inteligência artificial sem mesmo que ele perceba. Ela está presente na manhã, ao utilizar o aplicativo *Waze* para conduzir ao local de trabalho. De tarde, ao usar o *Google Tradutor* para traduzir um texto que lhe ajude a solucionar uma questão em seu escritório. E a noite, ao relaxar enquanto assisti a uma série da *Netflix* na televisão.

Estas três tecnologias utilizam da IA para ajudar as pessoas, seja ao facilitar, ou entreter o seu dia a dia. Já pensou como funciona uma tradução instantânea de uma imagem? Como a máquina identifica que há um texto na imagem? Como ela sabe em que idioma está escrito? Onde ela deve inserir o trecho já traduzido? E se o objeto for circular, curvando a frase? Ou se houver algo na frente bloqueando parcialmente o texto? Este é o caso do *Google Tradutor*. Para realizar esta tarefa, foram adicionadas dezenas de milhares de imagens com e sem textos para o algoritmo aprender a identificar a presença de um texto. Depois, o algoritmo realiza o reconhecimento do idioma escrito e o traduz simultaneamente.

O Facebook é outro exemplo de tecnologia que utiliza IA, no caso para reconhecimento facial em imagens. Sites e aplicativos que realizam transições bancárias, como o Itaú, garantem a segurança do usuário através de IA. Já o Gyant, é uma tecnologia voltada à saúde que, através de uma conversa natural, pede os sintomas do usuário para determinar a provável causa do que a acomete.

Na indústria química, a IA se manifesta desde os anos 1990, com aplicação de redes neurais para fazer aproximação de processos para fins de controle automático, quimometria e identificação de falhas (SOARES, 2017).

Em relação à detecção e identificação de falhas no processo, há diversos métodos estudados que pode-se encontrar na literatura, conforme exposto no capítulo 2. No presente trabalho, será utilizado o método de redes neurais artificiais para esse fim e, por isso, é apresentada sua definição, estrutura, topologia e treinamento.

2.6. Redes Neurais Artificiais

2.6.1. Definição

O conceito de rede neural artificial deriva da célula fundamental para o processamento de atividades do cérebro humano: o neurônio. Uma das funções básicas executadas por um neurônio é a combinação de sinais recebidos dos neurônios anteriores, conectados em grande parte aos dendritos. Caso a combinação dos sinais recebidos seja acima do limiar de excitação do neurônio, um impulso elétrico é produzido e propagado através do axônio para os neurônios seguintes. A estrutura individual desses neurônios, a topologia de suas conexões e o comportamento conjunto desses elementos de processamento naturais formam a base para o estudo das RNAs (BRAGA, *et al*. 2007). A figura 4 ilustra a estrutura de um neurônio biológico.

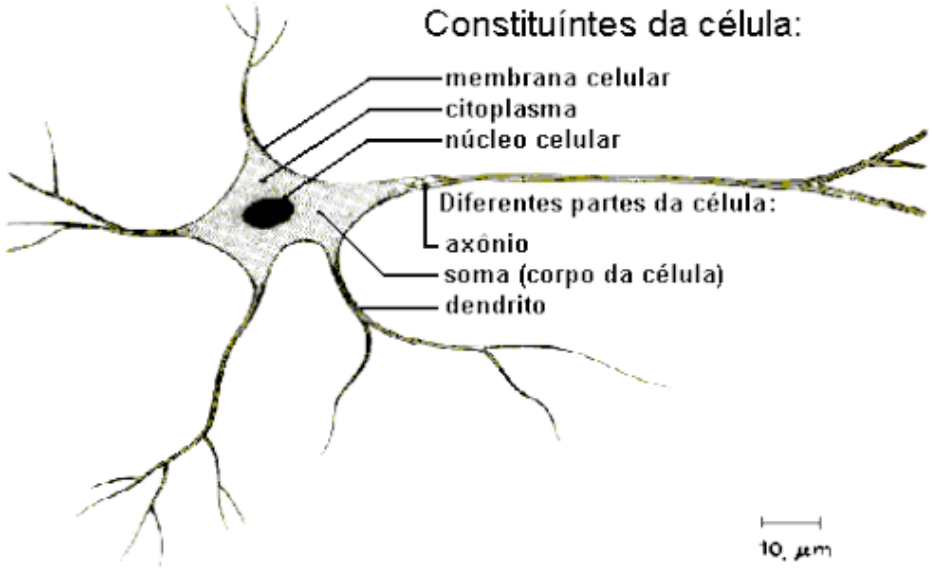


Figura 4: Estrutura do neurônio biológico (BRAGA, *et al*. 2007).

As RNAs tentam reproduzir as funções das redes biológicas, buscando implementar seu comportamento funcional e sua dinâmica. Através de um conjunto de dados iniciais, a RNA é treinada, a partir de um algoritmo, e aprende como o sistema em estudo se comporta. Esse conhecimento é então aplicado a novas situações de entrada para determinar a saída apropriada. A finalidade é estabelecer uma relação linear ou não, entre o conjunto de dados de entrada e uma correspondente saída (SANTOS, 2015).

Dentre as diversas tarefas que as RNAs podem realizar, as principais são: classificação, categorização, aproximação, previsão e otimização. Existe uma gama imensa de setores que lucram com a utilização de RNAs para suprir esses objetivos. Alguns exemplos são: setor financeiro, setor elétrico, automação e controle, modelagem de sistemas industriais, bioinformática, comércio eletrônico e comunicações (BRAGA, *et al*. 2007).

2.6.2. Estrutura da RNA

Como no modelo biológico, o neurônio artificial tem *n* terminais de entrada (que representam dendritos) que recebem os valores x_1, x_2, ..., x_n (representando ativações dos neurônios anteriores) e um terminal de saída *y* (axônio). Para representar o comportamento das sinapses, os terminais de entrada dos neurônios têm pesos acoplados w_1, w_2, ..., w_n. O efeito de uma sinapse particular *i* no neurônio pós-sináptico é dado por $x_i w_i$. A soma ponderada de todas as sinapses é realizada e se adiciona o limiar de ativação *b* (também chamado de bias), uma constante que serve para aumentar ou diminuir a soma e transladar uma função de ativação.

A função de ativação é responsável por gerar a saída y do neurônio a partir dos valores dos vetores de peso $w=(w_{k1}, w_{k2}, ..., w_{kn})^T$ e de entrada $x=(x_1, x_2, ..., x_n)^T$ (Braga, *et al*. 2007). A figura 5 apresenta o modelo de um neurônio artificial.

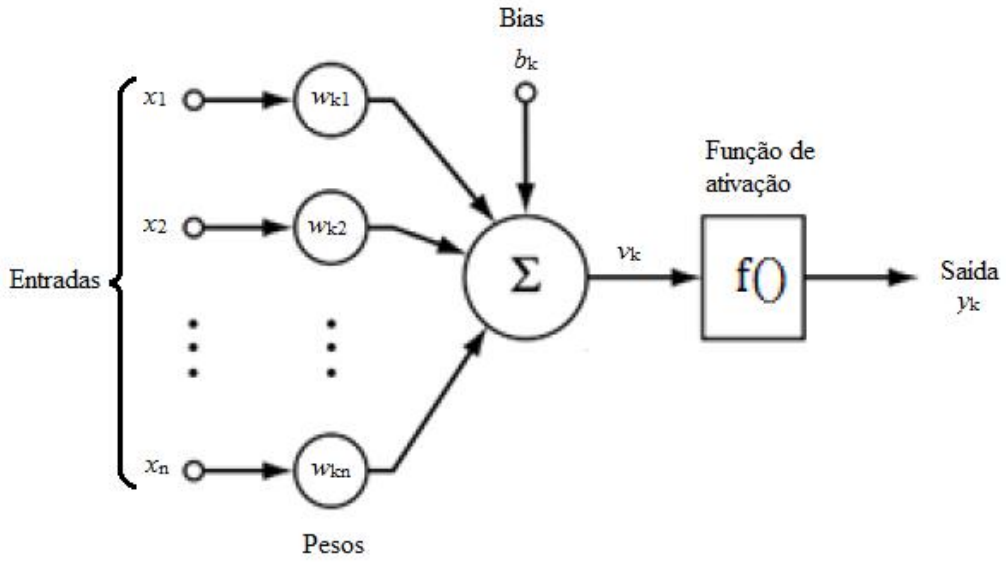


Figura 5: Estrutura do neurônio artificial (BRAGA, *et al.* 2007).

A equação que representa a saída de uma rede simples é

$$v_i = f(w_i * x + b_i) \quad (5)$$

$$y_j = w_j^{out} * v + b_j^{out} \quad (6)$$

Onde w_i é o vetor de pesos do neurônio i, b_i é o bias do neurônio i, y_i é a saída j da rede, $w^{out}{}_j$ é o vetor de pesos da saída j e $b^{out}{}_j$ é o bias da saída j; f é a função não linear aplicada a cada elemento.

Algumas das funções de ativação mais comuns são sigmoidal (sig), tangente hiperbólica (tanh), degrau e função de base radial (RBF) como a função gaussiana. A figura 6 mostra os gráficos destes exemplos, enquanto as equações 7-10 apresentam suas equações. No presente trabalho foi utilizada uma função de ativação do tipo sigmoidal.

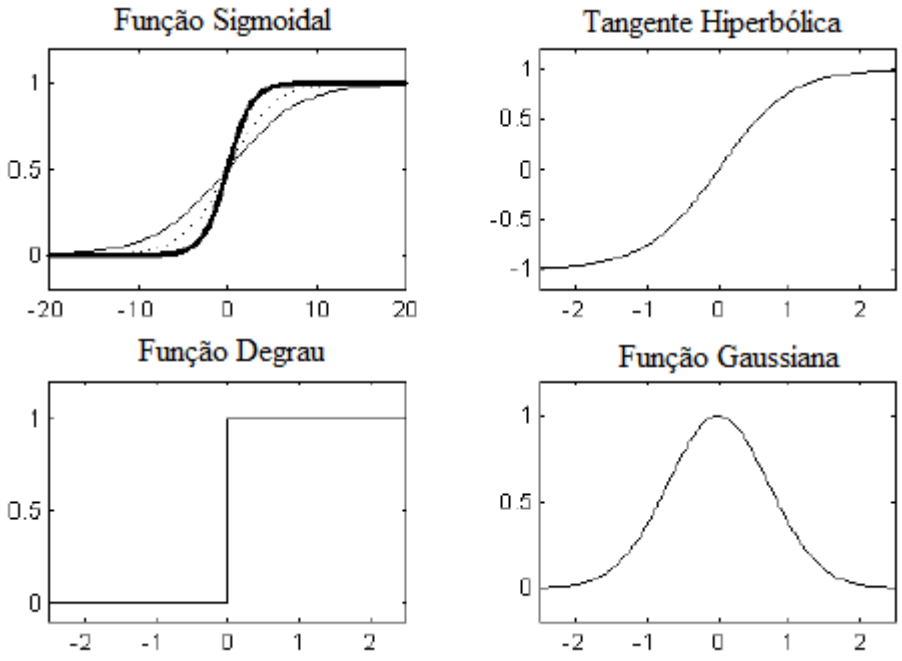


Figura 6: Funções de ativação mais comuns (BRAGA, *et al*. 2007).

$$sig(x) = \frac{1}{1 + e^{-\beta x}} \quad (7)$$

$$\tanh(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \quad (8)$$

$$degrau(x) = \begin{cases} 1 \; if \; x \geq 0 \\ 0 \; if \; x < 0 \end{cases} \quad (9)$$

$$RBF(x) = e^{-x^2} \quad (10)$$

Independente da função de ativação utilizada, quando isolados os neurônios têm habilidades limitadas. Porém, quando estes se agrupam, conectando-se em uma rede, os neurônios artificias desenvolvem a capacidade de resolver questões de alta complexidade.

Nas RNAs, os neurônios são organizados em camadas: a camada de entrada recebe os dados de entrada, as camadas internas ou ocultas representam as diversas nuances que os dados de treinamento podem ter e a camada de saída finaliza o processo de previsão ou classificação e os resultados são apresentados com um pequeno erro de estimativa. O número de camadas intermediárias é variável e, em teoria, quanto mais destas camadas, mais ajustada ficará a rede (KAYRI, 2016). A figura 7 ilustra a estrutura de uma RNA.

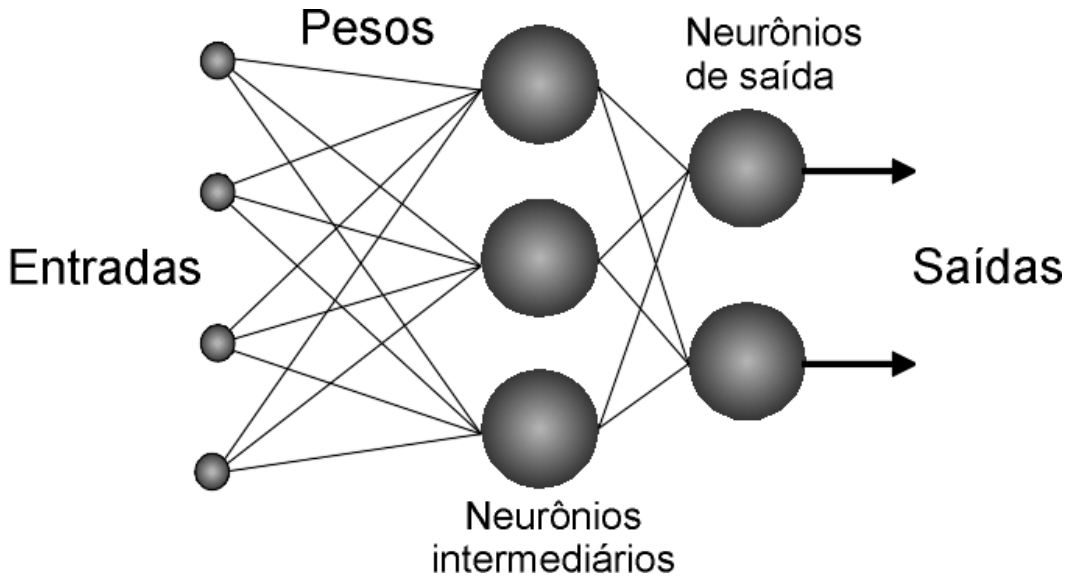


Figura 7: Estrutura de uma rede neural artificial.

2.6.3. Topologia da RNA

A topologia de uma RNA é o modo como diferentes quantidades de neurônios artificiais se interconectam e organizam para formar as mais variadas arquiteturas de camadas de rede. Existem três tipos de interconexões entre os neurônios: Intracamada, intercamada e recorrente. Na topologia de intracamada a saída de um neurônio alimenta outros neurônios da mesma camada. Nas conexões intercamadas cada nódulo se conecta a outros de diferentes camadas. Já nas redes

recorrentes, a saída de um neurônio alimenta a ele próprio. A figura 8 ilustra esses três tipos de interconexões de nódulos (FRANCISCO, 2000).

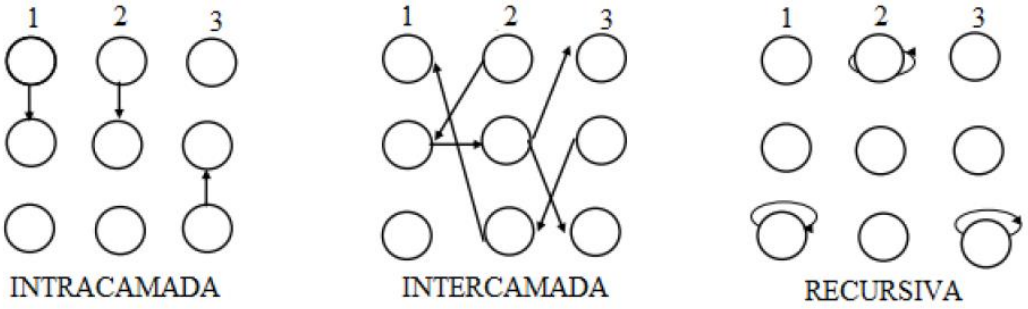


Figura 8: Opções de conexões entre neurônios em RNA (FRANCISCO, 2000).

Conforme FRANCISCO (2000), a conexão intercamada é particularmente importante para aplicações de engenharia. Dentro deste tipo de interconexão, existem duas opções: alimentação direta e retro-alimentação, conforme destacado na figura 9.

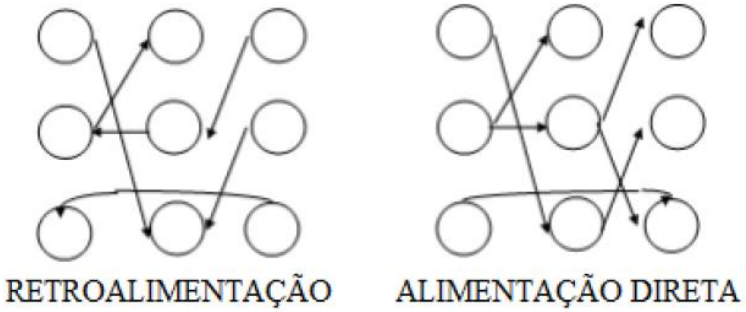


Figura 9: Opções de conexões intercamadas (FRANCISCO, 2000).

Ainda segundo os autores, modelagens dinâmicas de equipamentos tem por objetivo mapear uma resposta baseada em informações de entrada e, para isso, deve-se utilizar a conexão de intercamadas com alimentação direta, podendo esta topologia de redes ser chamada de *Perceptron* ou *feedforward*. Dentro desta topologia, ainda pode-se separar em duas classes de arquiteturas: Redes *feedforward* com camada única, as quais possuem conexões unidirecionais e apenas uma camada de entrada e uma camada de saída, e redes *feedforward* com

múltiplas camadas, que apresentam uma ou mais camadas intermediárias. A figura 10 ilustra essas classes.

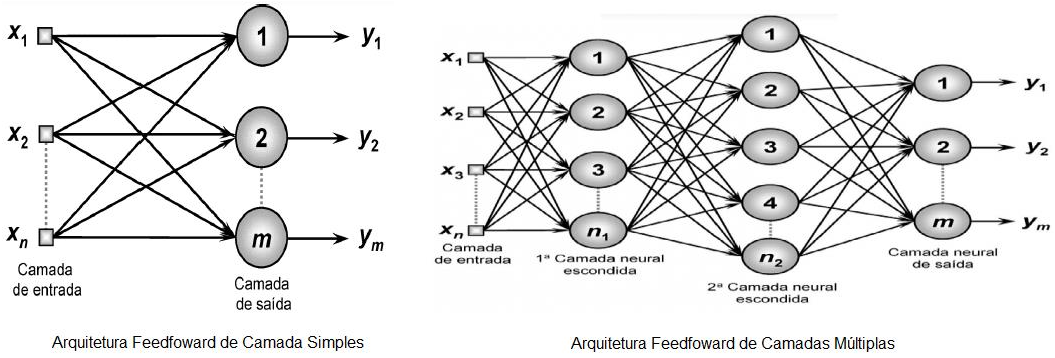


Figura 10: Classes de conexões intercamadas com alimentação direta (EMBARCADOS, 2016).

Além dos modos de conexões entre neurônios, a quantidade utilizada dos mesmos também é de grande importância, uma vez que determina a capacidade de uma rede resolver problemas de determinada complexidade. Quanto maior o número de neurônios, maior a complexidade da rede e maior a sua abrangência em termos de soluções possíveis. Contudo, não se pode dizer que haja na literatura um método para determinação exata da quantidade de neurônios a ser usada para na resolução de um problema.

Se forem usados poucos neurônios, pode acontecer o efeito de *underfitting*, ou seja, a rede não consegue detectar adequadamente os sinais em um conjunto de dados. Todavia, se a quantidade for exagerada, encontra-se o efeito de *overfitting*, onde há neurônios demais a serem treinados por um número limitado de informação contida no banco de dados (BRAGA, *et al*. 2007). Assim, o número ideal de neurônios ocultos é um dever difícil por depender de três questões externas: Complexidade de correlação entre variáveis dependentes e independentes sendo manipuladas por uma rede neural, número de conjuntos de dados de treinamento e teste disponíveis e quantidade de ruído que existe no conjunto de dados. Em geral, para encontrar a quantidade ideal de neurônios é preciso testar diversas quantidades de neurônios até alcançar o melhor resultado (YOUSEFI, *et al*. 2015).

2.6.4. Treinamento da RNA

Um fator que distingue as RNAs é o seu poder de aprender com exemplos. A chamada 'etapa de aprendizagem' consiste em um processo iterativo de ajuste de parâmetros da rede, os pesos das conexões, no qual se armazena o conhecimento adquirido do ambiente externo no final do processo. Basicamente, o que altera de um algoritmo de aprendizagem para outro é o modo como o ajuste dos parâmetros é calculado. O erro quadrático médio da resposta da rede em relação ao conjunto de dados fornecido pelo ambiente é utilizado como critério de desempenho pelos algoritmos de correção de erros. O ajuste gradual dos parâmetros é o que gera uma melhoria gradativa no desempenho da rede, uma vez que o erro da camada de saída é minimizado (BRAGA, *et al*. 2007). A equação 11 apresenta a equação geral de ajuste:

$$e(p) = \frac{1}{2}\sum_{i=1}^{p}(\gamma_d^i - \gamma_i)^2 \tag{11}$$

Onde *e(p)* é o erro quadrático médio, *p* é o número de exemplos de treinamento, $\gamma^i{}_d$ é a saída desejada para o vetor de entrada x_i e γ é a saída corrente da rede para o vetor x_i.

O treinamento de uma RNA pode ser classificado em dois grandes grupos: aprendizado não supervisionado e aprendizado supervisionado. No aprendizado não supervisionado, o treinamento ocorre apenas pela entrada de dados na rede, fazendo com que este somente seja possível quando existem valores iguais nos dados de entrada, contudo, se não houver, o aprendizado não será realizado (REGO, 2017).

O método mais comum para treinamento é o aprendizado supervisionado, onde há um supervisor responsável por observar a saída calculada e compara-la com a saída desejada para, deste modo, corrigir o comportamento da rede e continuar com o treinamento.

2.6.4.1. Algoritmo *Backpropagation*

O exemplo mais conhecido para aprendizado supervisionada de uma RNA é o algoritmo de *backpropagation* (BP). O BP é um procedimento de treinamento para redes neurais *feedforward* que, primeiramente, propaga os valores de entrada

para uma camada oculta e, em seguida, propaga as sensibilidades de volta para tornar o erro menor, através da atualização dos pesos no final do processo (KAYRI, 2016). A propagação para a camada oculta acontece em três etapas:

1. O banco de dados é apresentado às entradas da rede e as saídas dos neurônios da primeira camada oculta são calculadas;
2. Estas saídas alimentarão as entradas da camada seguinte. Da mesma forma, as saídas dos neurônios desta última camada são calculadas e o processo se repete até chegar na camada de saída;
3. As saídas geradas pela camada de saída são comparadas às saídas desejadas para o banco de dados fornecido, calculando assim o erro correspondente.

Por sua vez, a propagação em sentido reverso acontece em 4 passos (BRAGA, *et al*. 2007):

1. Utilizando o erro da camada de saída calculado anteriormente, é possível ajustar os pesos através do gradiente descendente do erro;
2. Utilizando o peso das conexões entre as camadas, que serão multiplicados pelos erros correspondentes, é possível propagar os erros dos neurônios da camada de saída para a camada anterior. Assim, tem-se um valor de erro estimado para cada neurônio da camada oculta que representa uma medida da influência de cada neurônio da camada oculta no erro de saída da camada a frente;
3. Assim como o procedimento anterior, os erros calculados para os neurônios da camada oculta são utilizados para ajusatar os pesos pelo gradiente descendente;
4. O processo se repete até chegar ao ajuste dos pessos da camada de entrada, finalizando o ajuste de todos os pesos da rede.

O desempenho do processo de aprendizado deste algoritmo é influenciado por dois parâmetros: taxa de aprendizado e o termo de momentum. A taxa de aprendizado (η) é um termo constante que varia de 0 a 1. Uma taxa muito pequena implica em mudanças mais suaves nos valores de pesos e limiares, porém aumenta o tempo necessário para o aprendizado e, além disso, pode causar um problema de mínimo local, chamado de *trapping*. Por outro lado, um número muito alto leva a

mudanças mais bruscas e pode causar uma oscilação em torno do mínimo global (REGO, 2017).

O momentum (α) é uma técnica utilizada para agilizar o processo de treinamento. Ele é um peso extra que é adicionado aos fatores ponderais, enquanto estes são ajustados. Isso faz com que a variação dos pesos ponderais seja acelerada e, consequentemente, a velocidade de treinamento aumenta (FRANCISCO, 2000). A equação 12 apresenta o cálculo para a atualização dos pesos.

$$\Delta w_{ij}^{t+1} = \eta s_i e_j + \alpha w_{ij}^{t} \tag{12}$$

Onde Δw_{ij}^{t+1} e w_{ij}^{t} são as variações do peso do neurônio j em relação à conexão i no instante t+1 e t respectivamente, η é a taxa de aprendizado, α é o termo de momentum, s_i é o valor de entrada pela conexão i do neurônio j e e_j é o valor do erro calculado do neurônio j.

JAMIL *et al.* (2015) é um exemplo do uso de RNAs *feedforward* com algoritmo de treinamento *backpropagation*. Sua rede tinha como objetivo detectar e identificar falhas na linha de transmissão de energia elétrica.

Apesar de ser muito utilizado, o algoritmo de *backpropagation* demora muito para convergir e tende a ficar preso em mínimos locais, o que torna seu rendimento ruim para resolver problemas maiores e mais complexos. Para solucionar estes problemas, foram desenvolvidos métodos de regularização para algoritmos de *backpropagation*. Dentre essas variações, as mais utilizadas são a de *Levenberg Marquardt* (LM) e regularização *Bayesiana* (RB), pois são capazes de obter erros quadráticos médios mais baixos do que qualquer outro algoritmo de treinamento.

2.6.4.2. Algoritmo de *Levenberg Marquardt* e Regularização *Bayesiana*

O LM foi desenvolvido especialmente para uma convergência mais rápida em algoritmos de *backpropagation*. Essencialmente, o BR tem uma função objetiva que inclui a soma de quadrados e a soma de pesos para minimizar os erros de estimativa e alcançar um bom modelo (KAYRI, 2016).

Como os métodos Quasi-Newton, o algoritmo de Levenberg Marquardt foi projetado para abordar a velocidade de treinamento de segunda ordem sem precisar computar a matriz Hessiana. Quando a função de desempenho tem a forma de uma soma de quadrados (como é típico em redes *feedforward*), a matriz Hessiana pode ser aproximada conforme a equação 13 e o gradiente pode ser calculado como na equação 14.

$$H = J^T J \tag{13}$$

$$g = J^T e \tag{14}$$

Onde *J* é a matriz jacobiana que contém as primeiras derivadas dos erros de rede em relação aos pesos e bias, J^T é a matriz jacobiana transposta e *e* é um vetor de erros da rede. A matriz Jacobiana pode ser calculada através de uma técnica padrão de *backpropagation* que é muito menos complexa do que computar a matriz Hessiana.

O algoritmo de Levenberg-Marquardt usa essa aproximação para a matriz hessiana na seguinte atualização semelhante a Newton:

$$x_{k+1} = x_k - (H + \mu I)^{-1} g \tag{15}$$

Sendo *I*, a matriz identidade, quando o escalar μ é zero, este é apenas o método de Newton, usando a matriz Hessiana aproximada. Assim, μ é diminuído após cada etapa bem-sucedida (redução na função de desempenho) e é aumentada apenas quando um passo experimental aumenta a função de desempenho. Desta forma, a função de desempenho é sempre reduzida em cada iteração do algoritmo (MATLAB).

A regularização bayesiana minimiza uma combinação linear de erros e pesos quadrados. Também modifica a combinação linear para que, no final do treinamento, a rede resultante tenha boas qualidades de generalização. A regularização ocorre dentro do algoritmo de *Levenberg Marquardt*. O *backpropagation* é usado para calcular o Jx Jacobiano do desempenho em relação aos pesos e bias da variável x. Cada variável é ajustada de acordo com *Levenberg Marquardt*:

$$JJ = Jx * Jx \tag{16}$$

$$Je = Jx * E \tag{17}$$

$$dx = \frac{-(JJ + I * mu)}{Je} \tag{18}$$

Onde *E* são todos os erros. O valor adaptativo *mu* é aumentado até que a mudança mostrada acima resulte em um valor de desempenho reduzido. A mudança é então feita para a rede e *mu* é diminuída (MATLAB).

2.6.5. Redes recorrentes

As redes recorrentes possuem grande importância no presente trabalho. Por possuírem conexões internas, onde as saídas de uma camada alimentam a ela mesma ou a anterior, são criadas memórias dentro desta rede, adquirindo assim um comportamento dinâmico. Redes recorrentes são excelentes para previsão de séries temporais por poderem possuir um atraso no tempo. O algoritmo de *backpropagation through time* é uma extensão do algoritmo de backpropagation padrão onde é acrescido à topologia da rede uma camada a cada instante de tempo. Ao final da sequência de entrada, os valores esperados são apresentados e o sinal gradiente é calculado retropropagando o sinal do erro no tempo (SOARES, 2017; BRAGA, *et al*. 2007).

2.6.5.1. Rede de *Elman*

Elman teve um papel fundamental ao introduzir as memorias nas RNAs. Em sua rede, a rede de Elman, além das zonas de entrada, intermediaria e saída, existe o diferencial da camada de contexto. Enquanto as camadas de entrada e saída se comunicam com o ambiente externo, as camadas intermediárias e de contexto não o fazem. As unidades de entrada são apenas unidades de armazenamento que passam os sinais sem modificá-los. As unidades de saída são unidades lineares que somam os sinais que recebem. As unidades intermediárias podem ter funções de ativação lineares ou não-lineares, e as unidades de contexto são usadas apenas para memorizar as ativações anteriores das unidades intermediárias, podendo ser consideradas como atraso no tempo em um passo (BRAGA, *et al*. 2007). A figura 11 ilustra a estrutura da rede de *Elman* com sua unidade de contexto.

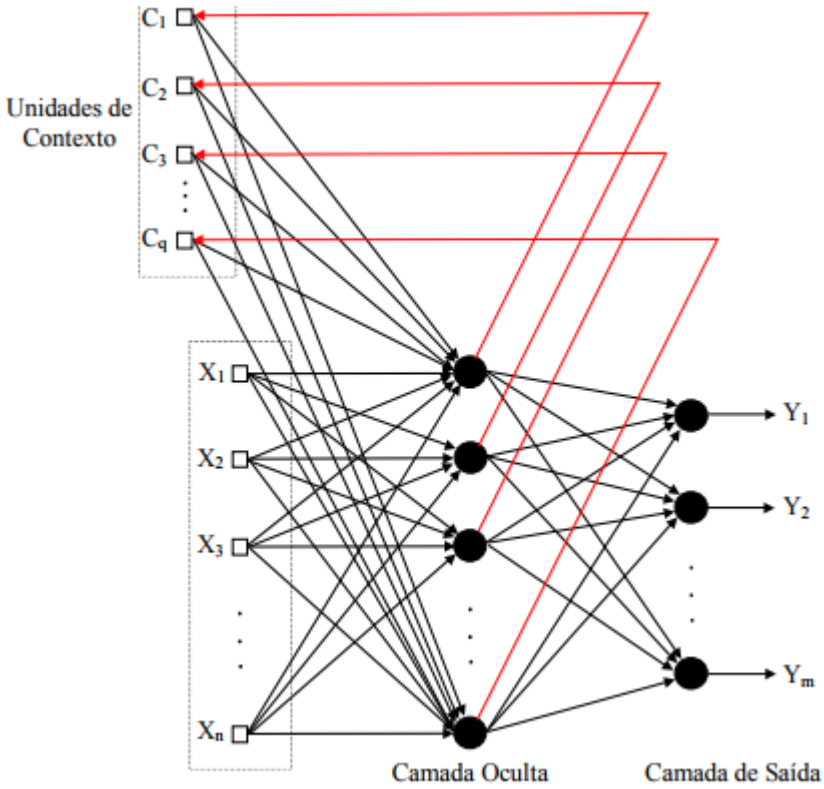


Figura 11: Estrutura de uma rede de *Elman* (BRAGA, *et al*. 2007).

Em um intervalo de tempo específico *k*, as ativações das unidades intermediárias (em *k*-1) e as entradas correntes (em *k*) são utilizadas como entradas da rede. Em um primeiro estágio *feedforward*. Essas entradas são propagadas para frente para produzir as saídas. Posteriormente, a rede é treinada com algoritmo de aprendizagem *backpropagation* padrão. Após esse passo de treinamento, as ativações intermediárias no tempo k são reintroduzidas através das ligações recorrentes nas unidades de contexto, sendo salvas nessas unidades para o próximo passo do treinamento (*k*+1) (BRAGA, *et al*. 2007).

Um trabalho interessante nesse contexto foi o de GAO e OVASKA (2002), que propuseram um esquema de detecção de falhas em motores baseado em redes neurais de *Elman*. Para melhorar a precisão da aproximação e obter um melhor desempenho de detecção, introduziram uma estratégia de treinamento auxiliada pelo algoritmo genético (GA) para rede de *Elman.*

2.6.5.2. Echo State Network

Além de atrasos temporais, outra abordagem para construção de redes recorrentes é a *reservoir computing* (RC). O RC apresenta não só as camadas de entrada e saída, como uma camada escondida, denominada reservatório, onde há um conjunto de neurônios que se interconectam de maneira recorrente e são gerados aleatoriamente.

A RC forneceu ao estudo de RNA dois novos modelos de rede de dois diferentes ambientes: *Liquid State Machines*, a qual surgiu da área de neurociência computacional, e *Echo State Network* (ESN), que é proveniente dos estudos de *Machine Learning*. Ambas as redes têm em comum características gerais que dispensam a adaptação supervisionada de todos os pesos de interconexões, sendo suficiente o treinamento supervisionado apenas dos pesos de saída, para obtenção de excelente desempenho em muitas tarefas. Este conceito torna a tarefa de aprendizado de rede mais simples e rápido (MARTINS, 2016).

As ESNs foram propostas pelo Dr. Hebert Jaeger em 2001 no artigo intitulado *The "echo state" approach to analysing and training recurrent neural network*, que foi corrigido em 2010 por ele mesmo em (JAEGER, 2010). O modelo das ESN foi motivado pela ineficiência dos algoritmos anteriores para o treinamento das redes neurais recorrentes, as quais são caracterizadas pela presença de ao menos um laço de realimentação em seus neurônios. As redes recorrentes apresentam um comportamento dinâmico não linear, já que elas podem manter a ativação ou saída mesmo na ausência da entrada, sendo esta característica chamada de "Memória Dinâmica" (GUACAS, 2016; RUEDA, 2014). A figura 12 ilustra a topologia de uma ESN.

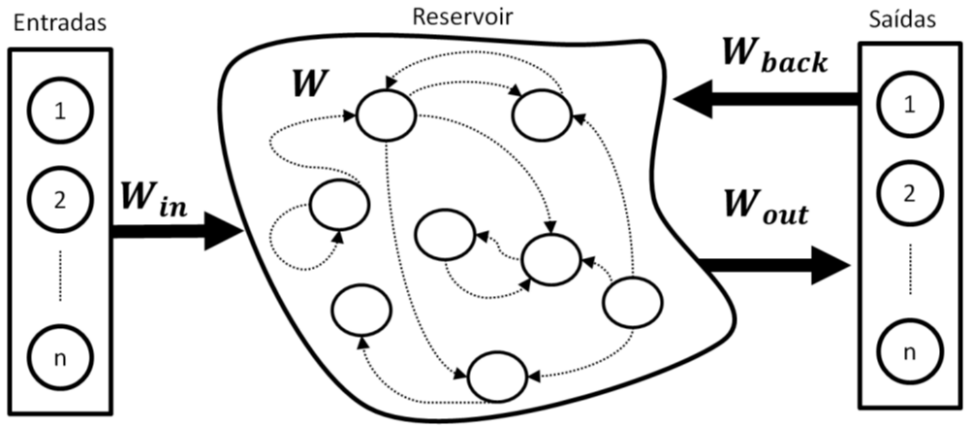


Figura 12: Topologia geral de uma rede *Echo State* (GUACAS, 2016).

Na figura 12, *W*, W_{in}, W_{out} e W_{back} são os pesos das conexões do reservatório, da camada de entrada, da camada de saída e da unidade de retroalimentação dos neurônios da camada de saída para os neurônios do reservatório. A matriz Wout é a matriz de pesos que deve ser atualizada durante o processo de treinamento da rede, enquanto as outras são inicializadas de forma aleatória no momento de criação da rede e não sofrem alterações (JAEGER, 2010).

O reservatório é o principal componente da ESN. Ele é gerado de forma aleatório no momento da criação da rede. As conexões internas podem ser entre um neurônio e outro ou de um neurônio para si próprio. Geralmente as ESN contém uma grande quantidade de neurônios (da ordem de 10-1500 ou mais) (Rueda, 2014). As características do reservatório dependem de três hiperparâmetros principais configurados inicialmente:

1. Raio espectral: Valor usado para escalar ou normalizar pesos do reservatório, normalmente menor que 1 para evitar um comportamento caótico da rede e saturações nas ativações dos neurônios;
2. Percentual de interconexão: Indica a porcentagem de neurônios que serão conectados aleatoriamente dentro do reservatório, entre um valor mínimo de 20%, para garantir um reservatório disperso, e 100% das conexões (JAEGER, 2010);
3. Tamanho do reservatório: Defini o número de unidades ou neurônios que formará o reservatório.

No reservatório, geralmente as funções de ativação são do tipo sigmoide ou tangente hiperbólica, enquanto que a função de ativação do neurônio da saída normalmente é linear. O estado do reservatório é um vetor de tamanho N, onde N é número de neurônios no reservatório. Esse vetor contém as ativações de cada neurônio em qualquer instante de tempo e é calculado pela equação 19.

$$x(n+1) = f(W_{in}u(n+1) + Wx(n) + W_{back}y(n)) \qquad (19)$$

Onde:

x(n + 1): Estado do reservatório no instante *n* + 1.

x(n): Estado do reservatório no instante *n.*

u(n + 1): Entradas da rede no instante *n* + 1.

y(n): Saídas da rede no instante *n*.

W_{in}: Matriz de pesos de entrada da rede.

W: Matriz de pesos do reservat

W_{back}: Matriz de pesos de retroalimentação.

f( ): Função de ativação

Para calcular a saída da rede é usada a equação 20.

$$y(n) = x(n)W_{out} \tag{20}$$

Onde:

y(n): Saída da rede no instante *n*.

x(n): Estado do reservatório no instante *n*.

W_{out}: Matriz de pesos de saída.

É importante enfatizar que a única matriz de pesos modificada durante a etapa de treinamento é a W_{out}, todas as outras, *W*, W_{in} e W_{back}, permanecem fixas desde o momento da criação da rede.

No trabalho proposto por MARTINS (2016), a ESN é usada para identificação automática de sistemas dinâmicos com o fim de aplica-los na identificação do nível do gerador de uma usina nuclear PWR. Assim como GAO e OVASKA (2002), este trabalho utilizou GA para treinamento da rede, otimizando a mesma.

3. Materiais e Métodos

Para iniciar o estudo experimental foi necessário primeiramente analisar o problema *Tennessee Eastman* para entender seu funcionamento. Foi utilizado uma adaptação do TE proposta por BATHELT *et al*. (2015), disponível e implementada em Simulink 6.0, ferramenta do Software *Matlab* 2017b. Todos os modelos e simulações foram desenvolvidos em um computador Windows 10 64 bits, com processador Intel Core i5-2310 CPU 2.9 GHz e memória 8 Gb. Os principais blocos construídos no Simulink para a simulação, podem ser visualizados no anexo 1.

Foram utilizadas as 12 variáveis de medição e as 41 variáveis manipuladas descritas no problema original de DOWNS e VOGEL (1993), totalizando 53 variáveis analisadas no processo.

O banco de dados foi formado a partir de 27 perturbações mais o padrão inicial do modelo com um intervalo de amostragem de 0,01 horas em 72 horas, totalizando 7201 linhas de dados por perturbação para 53 colunas de variáveis, sendo 12 de variáveis manipuláveis, 41 de variáveis medidas continuamente e a saída do programa que apenas indica o tipo de falha. Assim, foram obtidos 201628 vetores de dados (7201x28), um verdadeiro cenário de *Big Data*. A figura 13 apresenta um diagrama de blocos que representa a construção do bando de dados do processo. As simulações realizadas no TE tiveram tempo equivalente a 84 dias de processos industriais (28 simulações com 72 horas cada).

Para categorização dos dados em treinamento, teste e validação foram construídos códigos em Visual Basic (VBA) que a cada 10 linhas os enviavam para 3 diferentes abas na relação de 6, 2 e 2 linhas respectivamente.

Contudo, devido à grande quantidade de dados, que já passavam de 10 milhões (201628x53), a máquina não teve capacidade computacional de realizar a separação inteira e após algumas tentativas descobriu-se que seria necessário dividir o código referente ao treinamento em 6 partes independentes, de 5 em 5 perturbações. Para o caso de teste e validação foi preciso apenas fragmenta-los em 2 cada. Os códigos desenvolvidos encontram-se no anexo 2 a 4.

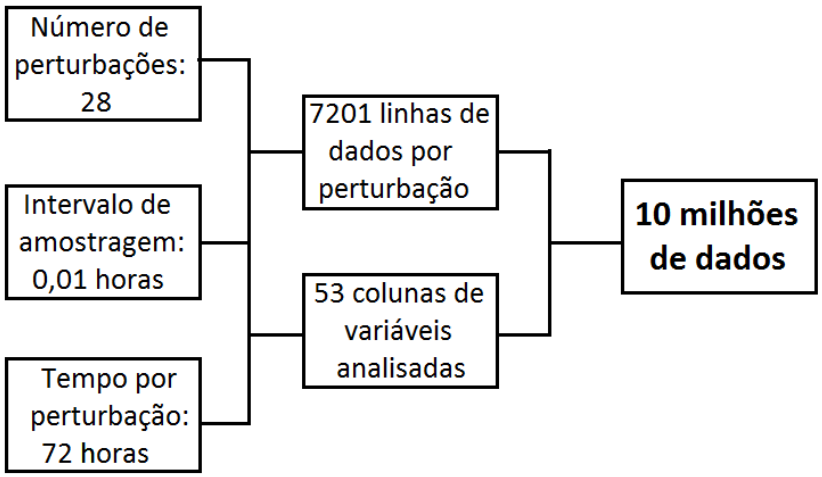


Figura 13: Representação da construção do banco de dados do processo através de um diagrama de blocos.

Contudo, devido à grande quantidade de dados, que já passavam de 10 milhões (201628x53), a máquina não teve capacidade computacional de realizar a separação inteira e após algumas tentativas descobriu-se que seria necessário dividir o código referente ao treinamento em 6 partes independentes, de 5 em 5 perturbações. Para o caso de teste e validação foi preciso apenas fragmenta-los em 2 cada. Os códigos desenvolvidos encontram-se no anexo 2 a 4.

Após a classificação dos dados, foi dado início a modelagem por redes neurais artificiais para obtenção de um resultado satisfatório para detectar e identificar falhas do processo. Os dados foram transferidos e organizados do Excel para arquivos "*.dat"*, podendo assim serem lidos pelo *Matlab*. O programa desenvolvido no *Sofware* apresenta as seguintes funções:

1. Leitura dos arquivos de dados;
2. Normalização dos dados entre -1 e 1;
3. Criação da rede neural;
4. Definição do número de iterações;
5. Valores iniciais dos pesos e bias (aleatório);
6. Treinamento da rede neural com o conjunto de dados;
7. Simulação da variável de saída a partir das entradas do conjunto de teste;
8. Desnormalização da variável de saída (detecção de falha);
9. Geração dos gráficos para análise dos erros;
10. Fim do programa.

A rede neural apresenta sensibilidade à escala dos dados, por isso se torna necessária sua normalização. Se os valores dos dados forem muito destoantes, a rede pode erroneamente atribuir uma maior importância a valores com maiores magnitudes.

A modelagem do conjunto de dados foi realizada através de três tipos de RNAs: *Feedforward*, *Elman* e *Echo State*. Os testes realizados para cada uma das redes serão apresentados a seguir.

3.1. Configurações para Redes Feedforward

Para o caso das redes *feedforward*, as configurações foram testadas variando quantidade de neurônios nas camadas intermediárias, quantidade de camadas intermediárias, função de ativação das camadas intermediárias e algoritmo de treinamento, conforme segue:

1. Escolha do número de neurônios: Em redes com apenas uma camada intermediária variou-se somente entre 80 ou 100 neurônios. Já as redes com mais de uma camada intermediária proporcionam espaço para uma maior variedade de topologias. Foram variados entre 60 e 100 neurônios nas duas camadas, sempre em múltiplos de cinco (60, 65, 70, etc);
2. Escolha da função de ativação: As funções de ativação tangente hiperbólica e logística foram testadas, mantendo-se fixo os números de neurônios e algoritmo de treinamento;
3. Escolha do algoritmo de treinamento: Os algoritmos de treinamento de *Levenberg Marquardt* e regularização *Bayesianas* foram testados, enquanto manteve-se fixo o número de neurônios e a função de ativação.

3.2. Configurações para Redes de Elman

Em respeito às redes de *Elman*, as possibilidades de diversidade do número de neurônios nas camadas intermediárias teve de ser menor devido ao grande esforço computacional que a rede recorrente apresenta. As variações de número de neurônios, função de ativação e algoritmo de treinamento foram realizadas da seguinte maneira:

1. Escolha do número de neurônios: Foram testadas apenas redes de duas camadas variando entre 40, 45, 50 ou 55 neurônios por camada;
2. Escolha da função de ativação: As mesmas funções das redes *feedforward* foram testadas, tangente hiperbólica e logística, mantendo-se fixo o número de neurônios e algoritmo de treino;
3. Escolha do algoritmo de treinamento: Novamente se utilizaram os algoritmos de *Levenberg Marquardt* e regularização *Bayesiana*, enquanto foi mantido fixo o número de neurônios e a função de ativação.

3.3. Configurações para Redes Echo State

No caso das redes *Echo State*, os parâmetros variados nas configurações testadas foram: o tamanho do reservatório, o raio espectral e o percentual de interconexão. Para cada um dos parâmetros foram analisados diversos valores até se chegar a duas opções cada de valores ótimos, ou seja, com os melhores desempenhos.

1. Escolha do tamanho do reservatório: O número de neurônios no reservatório variou de 20 a 1000, valores ótimos de 100 e 200 neurônios.
2. Escolha do raio espectral: Este parâmetro foi variado nos dois extremos, tanto perto de zero, quanto de um, obtendo valores ótimos de 0,55 e 0,99999.
3. Escolha do percentual de interconexão: Da mesma forma que o raio espectral, o percentual de interconexão vareou nos extremos, chegando a valores ótimos de 0,65 e 0,755.

3.4. Método de Desempenho

O desempenho das configurações dos modelos neurais foi definido através do coeficiente de determinação (R^2) e da soma dos quadrados dos erros (SSE), que representam a precisão do modelo. O cálculo do R^2 é apresentado na equação 21, enquanto a equação 22 mostra a SSE. Para avaliação da capacidade de generalização do modelo, foram analisados os gráficos de dispersão da validação (dados reais versus preditos).

$$R^2 = \frac{\sum_{i=1}^{n}(\hat{\gamma}_i - \bar{\gamma})^2}{\sum_{i=1}^{n}(\hat{\gamma}_i - \bar{\gamma})^2 + \sum_{i=1}^{n}(\bar{\gamma} - \hat{\gamma}_i)^2} \quad (21)$$

γ_i: valor observado;

$\hat{\gamma}_i$: valor estimado;

$\bar{\gamma}$: valor médio;

$$SSE = \frac{1}{2}\sum_{i=1}^{n}(\gamma_{di} - \gamma_i)^2 \quad (22)$$

γ_{di}: valor desejado de saída;

γ_i: valor calculado de saída;

Para avaliar o coeficiente de determinação, primeiramente deve-se checar a diferença entre o R^2 do treinamento e do teste. Se os valores estiverem muito longe, em ordem de grandeza, de um para o outro, significa que a rede não possui boa capacidade de predição e seu dimensionamento foi comprometido. Já se o R^2 do treinamento for baixo, entende-se que a rede não obteve boa capacidade de aprendizado e, por consequência, não conseguirá relacionar as variáveis de maneira correta.

O modelo SSE avalia a soma quadrada da distância entre os pontos da dispersão (experimental) e a reta de melhor ajuste de modelo (calculado). Sendo assim, quanto menor esse valor, mais próximos os valores calculados estarão dos valores experimentais, indicando uma boa predição.

Para modelagem das redes, o número de neurônios na camada intermediária foi escolhido através do cálculo do número de parâmetros, uma relação matemática que leva em consideração o número de neurônios em cada camada e a quantidade de dados disponíveis para treinamento, evitando problemas de dimensionalidade na rede. As equações 23 e 26 apresentam essa relação para as redes *feedforward*, com utilização de uma e duas camadas intermediárias, respectivamente, e as equações 19 e 20 para as redes recorrentes (GROSSI, 2017).

$$n_{CI\,1} = \frac{n_{TR} - n_s}{n_e + n_s + 1} \quad (23)$$

$$n_{CI\,2} = \frac{n_{TR} - n_s - n_{CI\,1} * (1 + n_e)}{n_{CI\,1} + n_s + 1} \quad (24)$$

$$n_{CI\,1} = \frac{n_{TR} - n_s}{n_e + n_{CT\,1} + n_s + 1} \quad (25)$$

$$n_{CI\,2} = \frac{n_{TR} - n_s - n_{CI\,1} * (1 + n_e + n_{CT\,2})}{n_{CI\,1} + n_{CT\,2} + n_s + 1} \quad (26)$$

n_{TR}: número de parâmetros de treinamento;

n_{CI}: número de neurônios de camada intermediária;

n_s: número de neurônios de camada de saída;

n_e: número de neurônios de camada de entrada;

n_{CT}: número de neurônios de contexto;

Se o número de parâmetros for inferior à quantidade de dados disponíveis para treinamento, a rede não terá problemas de dimensionalidade.

4. Resultados e Discussões

Conforme detalhado no capítulo 3, para realizar a detecção e identificação das falhas no conjunto de dados gerado pelo processo TE, foram modeladas três tipos de RNAs: *Feedforward*, *Elman* e *Echo State*. As configurações de redes testadas, bem como seus resultados serão trazidas para analisar o desempenho dos tipos de rede.

4.1. Rede Neural Feedforward

A primeira etapa de modelagem foi utilizar uma rede do tipo *feedforward*, para adquirir uma noção do tempo que o programa levaria para simular a rede e, principalmente, a coerência dela com os dados. O programa desenvolvido, encontra-se no anexo 5.

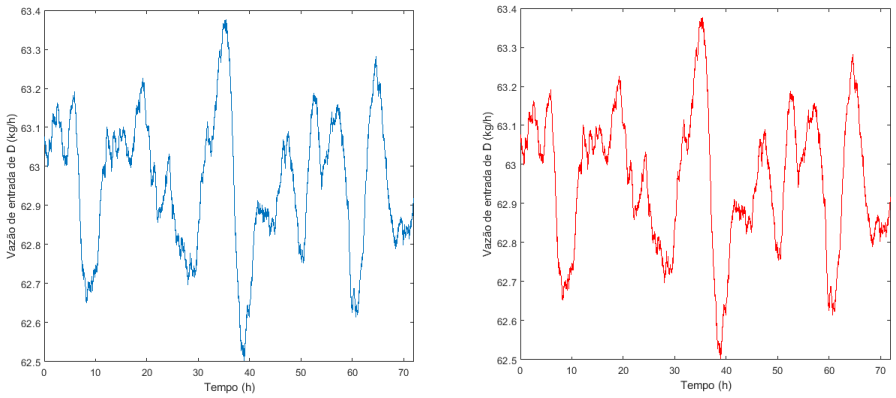
Assim que foi simulada a primeira configuração de rede, sendo uma das redes mais simples possível com apenas uma camada intermediária e 70 neurônios, já foi possível perceber que, após mais de três horas de corrida, este seria um trabalho que levaria meses para se concluir. A tabela 7 apresenta esta parte inicial do trabalho que se utiliza de todos os dados em rede *feedforward* alterando-se o número de neurônios, número de camadas internas, algoritmo de treinamento e função de ativação das camadas intermediárias. A função de ativação da camada de saída utilizada para todas as redes foi linear (*purelin*). Intitulou-se Caso 1 esta situação que tem como dados de entrada da rede o banco de dados originais.

Tabela 7: Configurações modeladas para RNA *feedforward* aplicada ao Caso 1.

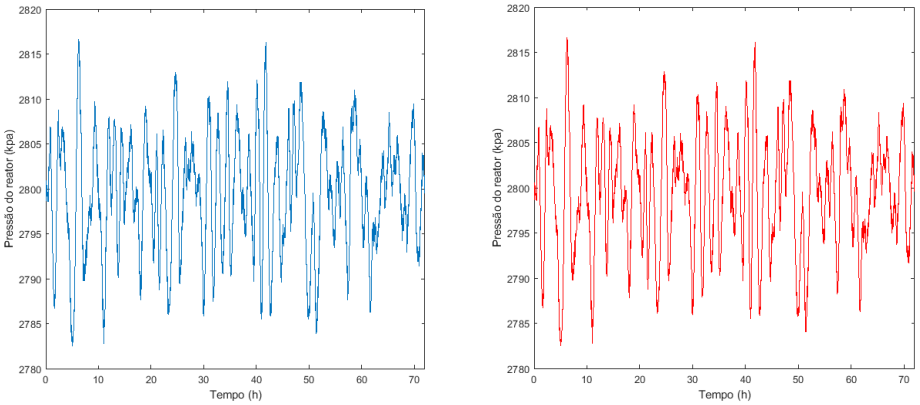
Topologia	Algoritmo de treinamento	Funções de ativação das camadas intermediárias	Tempo	SSE	R^2 (treinamento)	R^2 (teste)
53X70X1	trainlm	tansig	03:11:28	2,02E+04	0,73006	0,72726
53X75X1	trainlm	tansig	03:08:49	2,38E+04	0,67196	0,66977
53X80X1	trainlm	logsig	03:40:23	2,08E+04	0,72124	0,71913
53X80X1	trainbr	tansig	03:52:27	2,50E+04	0,65074	0,64909
53X90X1	trainlm	tansig	04:52:51	1,49E+04	0,80983	0,80468
53X60X60X1	trainlm	tansig/tansig	09:06:37	1,16E+04	0,85604	0,85273
53X80X70X1	trainlm	tansig/tansig	11:46:22	1,33E+04	0,83300	0,82974

A abreviação *trainlm* no *Matlab* se refere ao algoritmo de treinamento de *Levenberg Marquardt*, enquanto o *trainbr* concerne à regularização *Bayesiana*. Enquanto *tansig* e *logsig* fazem referência às funções de ativação tangente hiperbólica e logística, respectivamente.

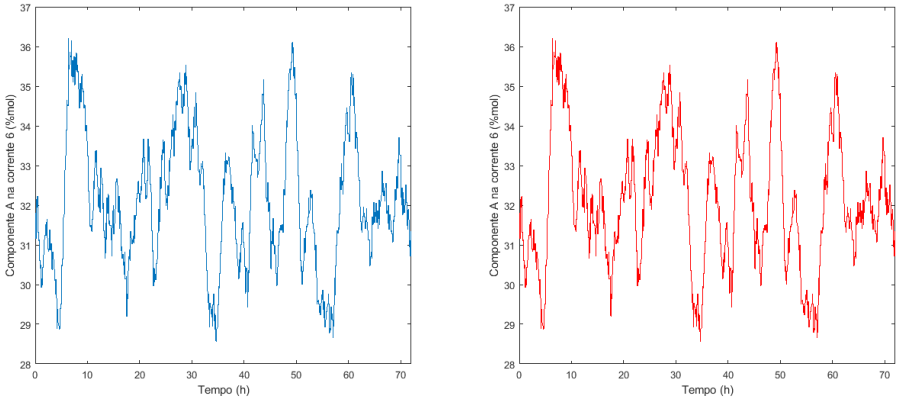
Em vista do extenso tempo de corrida que as redes tomaram, foram propostos dois caminhos para redução do mesmo sem que houvesse alteração em seu padrão. Em primeiro lugar, por meio de um programa construído em VBA, foram retiradas metade das linhas de dados da seguinte maneira: mantiveram-se as linhas ímpares enquanto as pares foram excluídas. A segunda solução foi realizar uma análise da influência das variáveis e remover um terço destas, deletando aquelas que geravam menor influência nos resultados. O banco de dados do primeiro método será aqui chamado de Caso 2, enquanto o segundo método terá o nome de Caso 3. A figura 14 apresenta a comparação do efeito da retirada de 50% dos dados na curva de três variáveis do processo, uma variável manipulável, outra medida continuamente e a última medida da composição de um componente.



a)



b)



c)

Figura 14: Comparação entre o banco de dados com 100% (em azul) e 50% (em vermelho) dos dados gerados pelo programa TE, sendo que a) representa uma variável manipulável, b) medida continuamente e c) medida da composição de um componente.

As soluções foram de fato eficazes e ambas conseguiram reduzir o tempo de corrida em torno da metade da original. Observou-se também que havia três variáveis que mantinham valores constantes e, por isso, foram retiradas do banco com 50% dos dados, já que sua permanência não alterava o resultado. Foi dado ainda um passo além e criou-se um novo banco de dados com a união dos dois caminhos tomados, 50% de dados e um terço de variáveis, o que fez o programa simular ainda mais rápido. Este último método foi chamado de Caso 4. Ao final de semanas de testes e dezenas de configurações de redes, foram encontradas redes com excelente desempenho, conforme apresentado nas tabelas 8, 9 e 10.

Tabela 8: Configurações modeladas para RNA *feedforward* aplicada ao Caso 2.

Topologia	Algoritmo de treinamento	Funções de ativação das camadas intermediárias	Tempo	SSE	R^2 (treinamento)	R^2 (teste)
50X80X1	trainlm	tansig	01:45:11	9,56E+03	0,74663	0,74096
50X100X1	trainlm	tansig	02:16:27	8,13E+03	0,79031	0,78241
50X60X60X1	trainlm	tansig/tansig	03:42:35	4,14E+03	0,89940	0,89074
50X70X70X1	trainlm	tansig/tansig	05:26:54	2,20E+05	0,82317	0,81707
50X90X90X1	trainlm	tansig/tansig	12:08:18	7,04E+03	0,82153	0,8142
50X100X100X1	trainlm	tansig/tansig	17:08:03	6,17E+03	0,84569	0,83756
50X60X60X1	trainlm	logsig/tansig	04:06:26	6,14E+03	0,84652	0,8422
50X60X60X1	trainlm	logsig/logsig	03:26:44	9,22E+03	0,75801	0,75449
50X60X60X1	trainlm	tansig/logsig	03:56:29	4,87E+03	0,88034	0,85908
50X70X60X1	trainlm	tansig/tansig	05:24:25	5,74E+03	0,85731	0,85016
50X60X70X1	trainlm	tansig/tansig	04:53:55	5,32E+03	0,86871	0,86195
50X60X70X1	trainbr	tansig/tansig	05:28:45	1,28E+04	0,64141	0,63856
50X70X60X1	trainbr	tansig/tansig	05:47:48	1,21E+04	0,66308	0,66079
50X65X65X1	trainlm	tansig/tansig	05:26:51	4,48E+03	0,89063	0,88104
50X65X65X1	trainbr	tansig/tansig	05:50:52	1,23E+04	0,65905	0,65717

Tabela 9: Configurações modeladas para RNA *feedforward* aplicada ao Caso 3.

Topologia	Algoritmo de treinamento	Funções de ativação das camadas intermediárias	Tempo	SSE	R^2 (treinamento)	R^2 (teste)
37X80X1	trainlm	tansig	01:44:50	2,25E+04	0,69321	0,69074
37X60X60X1	trainlm	logsig/tansig	05:11:05	1,09E+04	0,86476	0,85960
37X60X60X1	trainlm	tansig/logsig	05:32:10	1,04E+04	0,87162	0,86447
37X70X70X1	trainlm	tansig/tansig	13:19:20	1,33E+04	0,83255	0,82812
37X80X80X1	trainlm	tansig/tansig	17:04:18	1,54E+04	0,80279	0,79874

Tabela 10: Configurações modeladas para RNA *feedforward* aplicada ao Caso 4.

Topologia	Algoritmo de treinamento	Funções de ativação das camadas intermediárias	Tempo	SSE	R^2 (treinamento)	R^2 (teste)
37X80X1	trainlm	tansig	00:53:16	1,08E+04	0,70745	0,70158
37X100X1	trainlm	tansig	01:12:15	1,01E+04	0,73160	0,72473
37X60X60X1	trainlm	tansig/tansig	01:22:27	1,15E+04	0,68468	0,67981
37X60X70X1	trainbr	tansig/tansig	05:27:01	1,23E+04	0,65801	0,65554
37X70X70X1	trainlm	tansig/tansig	05:56:16	5,77E+03	0,85653	0,84797
37X90X90X1	trainlm	tansig/tansig	09:29:06	7,43E+03	0,81126	0,80208
37X100X100X1	trainlm	tansig/tansig	14:42:51	7,84E+03	0,80095	0,7915
37X60X60X1	trainlm	logsig/tansig	02:58:50	5,62E+03	0,86060	0,85214
37X60X60X1	trainlm	logsig/logsig	02:58:06	8,37E+03	0,78340	0,77837
37X60X60X1	trainlm	tansig/logsig	02:56:23	5,77E+03	0,85672	0,85009
37X60X70X1	trainlm	tansig/tansig	03:41:21	4,10E+03	0,90026	0,88692
37X70X60X1	trainlm	tansig/tansig	04:14:05	9,35E+03	0,75390	0,75026
37X60X70X1	trainbr	tansig/tansig	04:05:30	1,39E+04	0,59897	0,59794
37X70X60X1	trainbr	tansig/tansig	04:33:38	1,15E+04	0,68467	0,68148
37X65X65X1	trainlm	tansig/tansig	03:52:23	6,94E+03	0,82448	0,81774
37X65X65X1	trainbr	tansig/tansig	04:17:19	1,24E+04	0,65558	0,65338

As diversas RNAs modeladas foram analisadas pelos parâmetros de R^2 e SSE e a melhor configuração de rede *feedforward* é apresentada na tabela 11 e o gráfico referente ao seu treinamento na figura 15. Por apresentar um valor alto para o R^2 do treinamento, 90,0026%, pode-se afirmar que esta rede possui uma boa capacidade de aprendizado. Visto que o R^2 do teste também atingiu um valor alto, 88,692%, é possível afirmar que sua capacidade de predição é igualmente

boa. Devido ao fato dos R^2 de treinamento e teste terem desvio menor de 2%, conclui-se que a rede não teve problema de dimensionamento.

Tabela 11: Configuração da rede *feedforward* que apresentou os melhores resultados.

Tipo de rede	***Feedforward***
Banco de dados utilizado	Caso 4
Topologia	37X60X70X1
Algoritmo de treinamento	trainlm
Funções de ativação das camadas intermediárias	tansig/tansig
SSE	4,10E+03
R^2 (treinamento)	0,90026
R^2 (teste)	0,88692
R^2 (validação)	0,88316

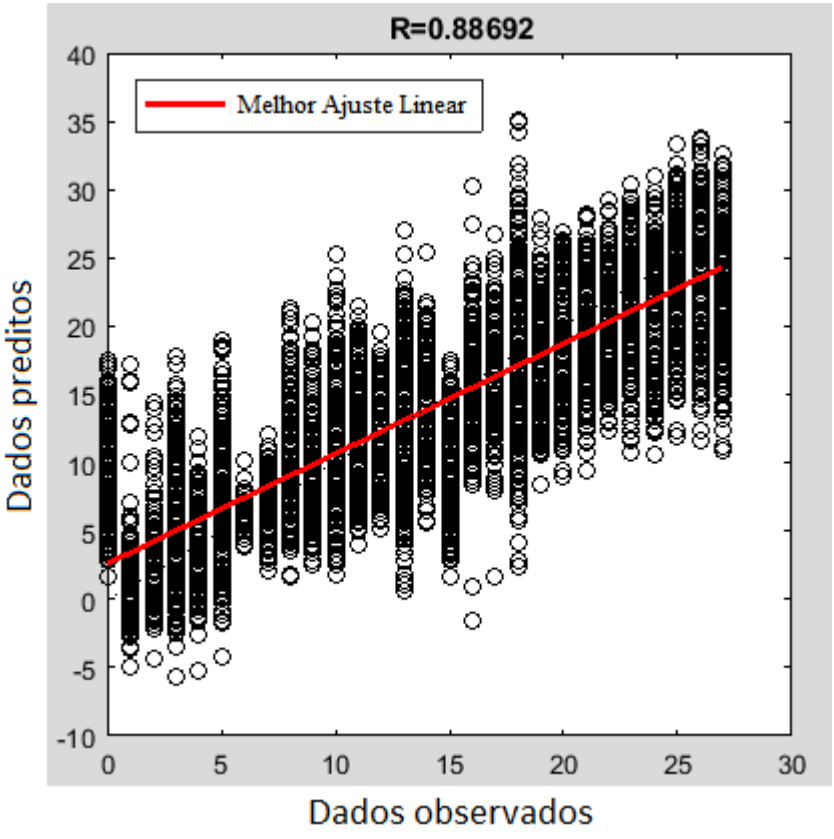


Figura 15: Gráfico para o teste da rede *feedforward*.

Outro parâmetro a ser avaliado é o erro SSE, que avalia a dispersão dos pontos do gráfico e teve valor de 4100 para esta rede. A princípio o SSE parece alto, contudo é preciso levar em consideração a enorme quantidade de dados analisados. Cada dado gera um erro e quando todos estes são somados é natural

obter um SSE alto. A figura 16 apresenta a dispersão da validação, onde prova-se que o modelo obteve uma boa capacidade de generalização dos dados, errando apenas nas detecções das falhas do tipo 1 (quando não há falha, apenas o comportamento padrão do processo) e do tipo 16 (desconhecido).

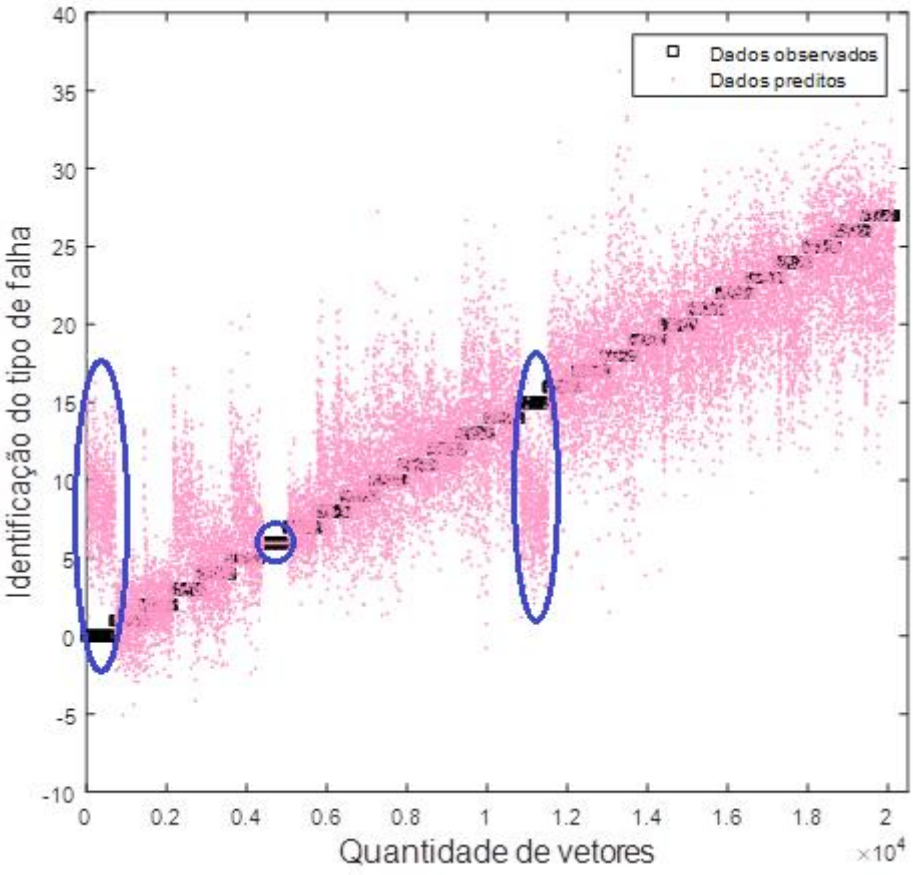


Figura 16: Gráfico de dispersão da validação para rede *feedforward* que apresentou os melhores resultados.

É possível concluir que a rede *feedforward* obteve boa adaptação ao conjunto de dados, apresentando boa capacidade de aprendizado, predição e generalização. A rede apenas não conseguiu predizer dois tipos de falhas, sendo que a primeira provavelmente ocorreu pelo motivo de não estar acontecendo de fato uma falha, apenas as condições padrões do processo.

No trabalho realizado por RICKER (1995), também é analisada a detecção e identificação de falhas no processo TE através de uma rede neural *feedforward*. Assim como no presente trabalho, RICKER provou que as redes de *feedforward* podem ser usadas com sucesso para predição de falhas em processos industriais.

4.2. Rede Neural de Elman

Após os resultados satisfatórios das redes feedforward, a próxima etapa se compunha no uso de redes recorrentes. A primeira utilizada foi a rede de Elman, que utiliza unidades de contexto apenas para memorizar as ativações anteriores das unidades intermediárias e pode ser consideradas como atraso no tempo em um passo. Este atraso aumenta consideravelmente o cálculo computacional e, consequentemente, esta rede levará um tempo muito maior para cada simulação, o que tornou obrigatório o uso apenas do menor banco de dados, pois em outros casos o programa não tinha capacidade computacional de simular as corridas. As configurações e resultados das redes de Elman estão apresentadas na tabela 12. O programa desenvolvido para redes de Elman é o mesmo que para redes feedforward, apresentando apenas uma pequena diferença, como explicado no anexo 5.

Tabela 12: Configurações modeladas para RNA de *Elman* aplicada ao Caso 4.

Topologia	Algoritmo de treinamento	Funções de ativação das camadas intermediárias	Tempo	SSE	R^2 (treinamento)	R^2 (teste)
37X50X50X1	trainlm	tansig/tansig	15:22:18	1,17E+04	0,67665	0,67366
37X40X40X1	trainlm	tansig/logsig	10:51:23	8,97E+03	0,76532	0,76186
37X40X40X1	trainbr	tansig/tansig	08:10:47	1,27E+04	0,64622	0,64394
37X45X45X1	trainlm	tansig/tansig	16:44:06	6,95E+03	0,82400	0,82096
37X45X45X1	trainbr	tansig/tansig	11:40:09	1,34E+04	0,61888	0,61675
37X45X45X1	trainlm	tansig/logsig	17:31:11	8,42E+03	0,78190	0,77819
37X45X45X1	trainlm	logsig/tansig	11:15:49	1,25E+04	0,65077	0,64856
37X50X50X1	trainlm	logsig/tansig	23:47:03	1,44E+04	0,57932	0,57881
37X50X50X1	trainlm	tansig/logsig	16:59:13	6,70E+03	0,83119	0,82267
37X40X50X1	trainlm	tansig/tansig	17:06:08	1,21E+04	0,66500	0,66159
37X50X40X1	trainlm	tansig/tansig	12:18:13	8,54E+03	0,77833	0,77393
37X55X45X1	trainlm	tansig/tansig	23:59:59	1,07E+04	0,71012	0,70698
37X45X55X1	trainlm	tansig/tansig	17:09:11	6,29E+03	0,84237	0,83637
37X55X45X1	trainlm	tansig/logsig	23:59:59	8,53E+03	0,77869	0,77563
37X45X55X1	trainlm	tansig/logsig	17:23:41	8,30E+03	0,78546	0,78161
37X50X50X1	trainbr	tansig/logsig	18:08:08	1,25E+04	0,64962	0,64807

Foram analisadas todas as redes de *Elman* modeladas, de acordo com os parâmetros R^2 e SSE, e a melhor configuração é apresentada na tabela 13 e o gráfico referente ao seu treinamento na figura 17. Já que o R^2 do treinamento obteve um valor alto, 84,237%, essa rede teve boa capacidade de aprendizado. Da mesma forma, o R^2 do teste também atingiu um valor alto, 83,637%, o que permitiu afirmar que sua capacidade de predição é igualmente boa. Como os R^2 de treinamento e teste tiveram valores próximos, conclui-se que a rede não teve problema de dimensionamento.

Tabela 13: Configuração da rede de *Elman* que apresentou os melhores resultados.

Tipo de rede	***Elman***
Banco de dados utilizado	Caso 4
Topologia	37X45X55X1
Algoritmo de treinamento	trainlm
Funções de ativação das camadas intermediárias	tansig/tansig
SSE	6,29E+03
R^2 (treinamento)	0,84237
R^2 (teste)	0,83637
R^2 (validação)	0,83419

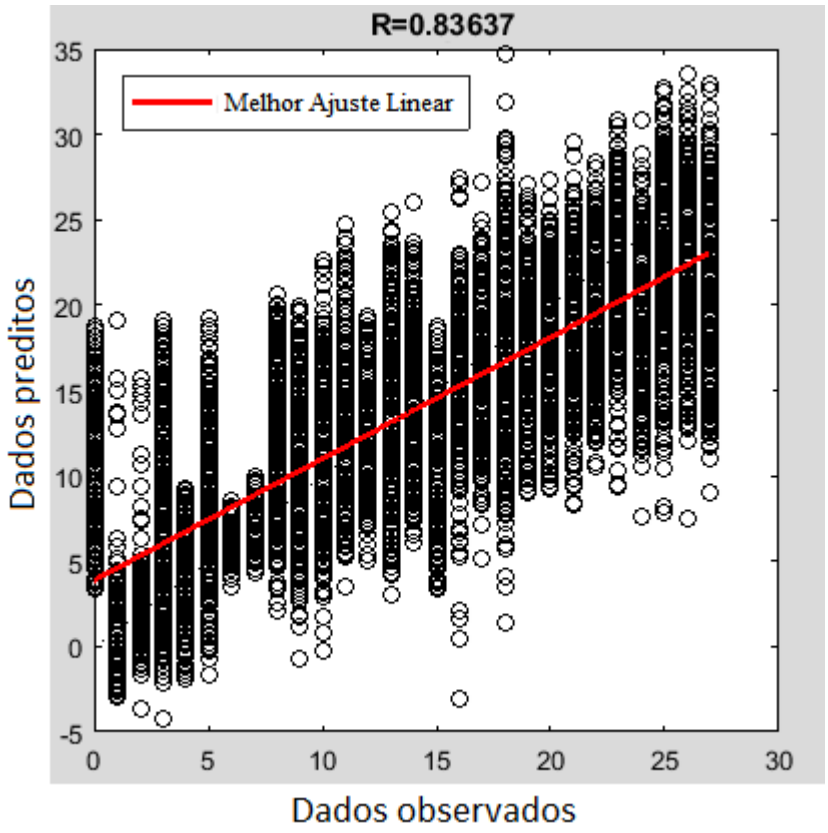


Figura 17: Gráfico para o teste da rede de *Elman*.

O parâmetro do erro SSE avaliou a dispersão dos pontos do gráfico e, para essa rede, obteve o valor de 629. Da mesma maneira que a rede *feedforward*, a de *Elman* também apresentou um valor alto de SSE devido à quantidade de dados modelados. A figura 18, que apresenta a dispersão da validação, revelou mais uma vez a capacidade de generalização dos dados, errando nas mesmas detecções de falhas do modelo *feedforward*, o tipo 1 (comportamento padrão do sistema) e 16 (desconhecido).

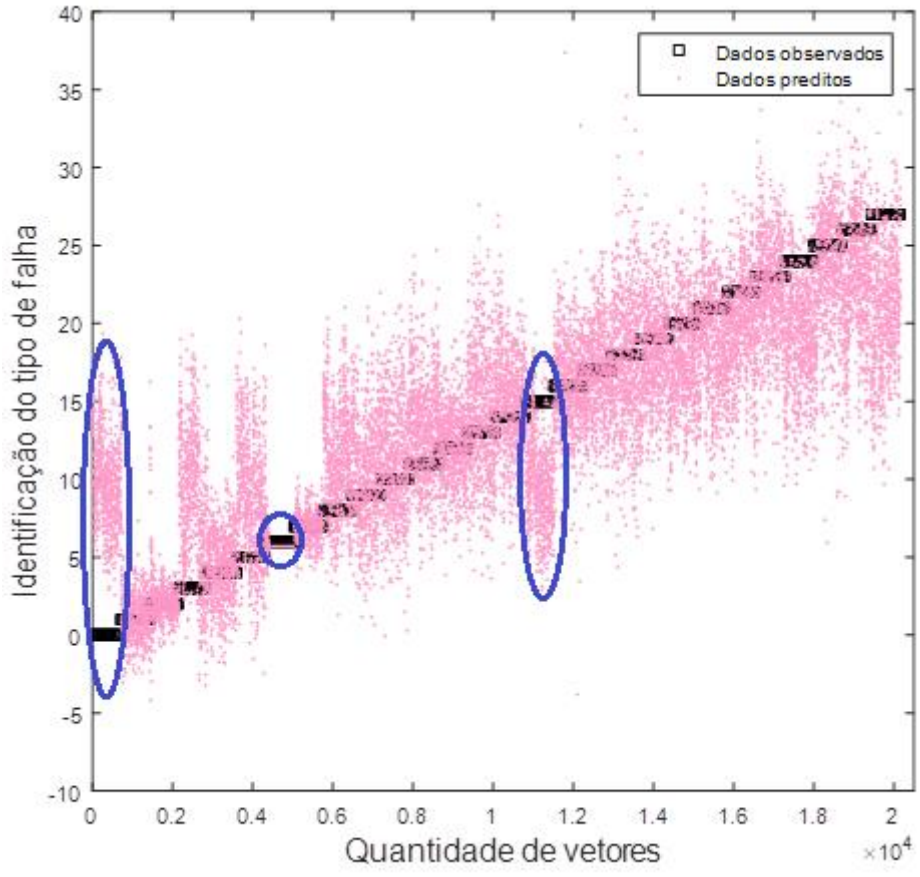


Figura 18: Gráfico de dispersão da validação para rede de *Elman* que apresentou os melhores resultados.

Em suma, a rede neural de *Elman* modelou bem o conjunto de dados, apresentando boa capacidade de aprendizado, predição e generalização. Contudo, o resultado da rede *feedforward* foi ainda superior a esta. O problema de predição de dois específicos tipos de falhas da modelagem da rede *feedforward* se repete aqui na de *Elman*.

No trabalho de AHMAD e HAMID (2001), o processo TE também é utilizado para construção do banco de dados, porém apenas as variáveis de pressão, temperatura e água de resfriamento do reator são analisados, não se situando num contexto de *Big Data*, como na presente dissertação. O desempenho da rede de *Elman* modelada por AHMAD e HAMID obteve bons resultados, assim como no presente trabalho.

4.3. Rede Echo State

Finalizada a análise da primeira rede recorrente, continuou-se para a seguinte, a *Echo State Network*. Como explicado no capítulo 3, por se tratar de uma rede recorrente da área de *reservoir computing*, as ESN contém, além das camadas de entrada e saída, uma camada escondida, denominada reservatório. O reservatório é o componente fundamental da ESN e suas características dependem de três hiperparâmetros principais: raio espectral, percentual de interconexão e tamanho do reservatório. Desta forma, vários modelos de rede foram testados, conforme apresentado na tabela 14. Todos os modelos contêm 37 variáveis na camada de entrada e 1 na camada de saída. O tempo de simulação não foi apresentado na tabela devido à rapidez da rede, que levou no máximo quinze minutos por simulação. Isso acontece devido ao fato das regressões nas ESNs serem muito mais simples e menos custosas computacionalmente em relação às outras duas analisadas. O programa desenvolvido para ESN, disponibilizado por RIBEIRO *et al*. (2018) e proposto por JAEGER (2001), encontra-se no anexo 6.

Tabela 14: Hiperparâmetros e resultados das redes *Echo State* aplicados ao Caso 4.

Tamanho do reservatório	Raio Espectral	Percentual de interconexão	R^2 (treinamento)	R^2 (teste)
100	0,55	0,65	0,53751	0,49861
200	0,55	0,65	0,55076	0,29925
100	0,55	0,755	0,53233	0,51167
200	0,99999	0,755	0,53008	0,51552
100	0,99999	0,755	0,52007	0,51423
200	0,99999	0,65	0,52921	0,4631

Das ESN modelas, a que obteve melhor configuração, de acordo com os parâmetros R^2 e SSE, é apresentada na tabela 15 e o gráfico referente ao seu treinamento na figura 19. Como o valor do R^2 do treinamento obteve um valor baixo, 53,008%, a rede não teve capacidade de aprender suficientemente bem com os dados do processo. O mesmo aconteceu para a capacidade de predição, já que o valor do R^2 do teste também atingiu um valor baixo, 51,552%. A

dimensionalidade da rede foi claramente correta, já que os valores dos R^2 de treinamento e teste se mantiveram próximos.

Tabela 15: Configuração da rede de *Echo State* que apresentou os melhores resultados.

Tipo de rede	***Echo State***
Banco de dados utilizado	Caso 4
Topologia	37X200X1
Raio Espectral	0,99999
Percentual de interconexão	0,755
R^2 (treinamento)	0,53008
R^2 (teste)	0,51552
R^2 (validação)	0,50762

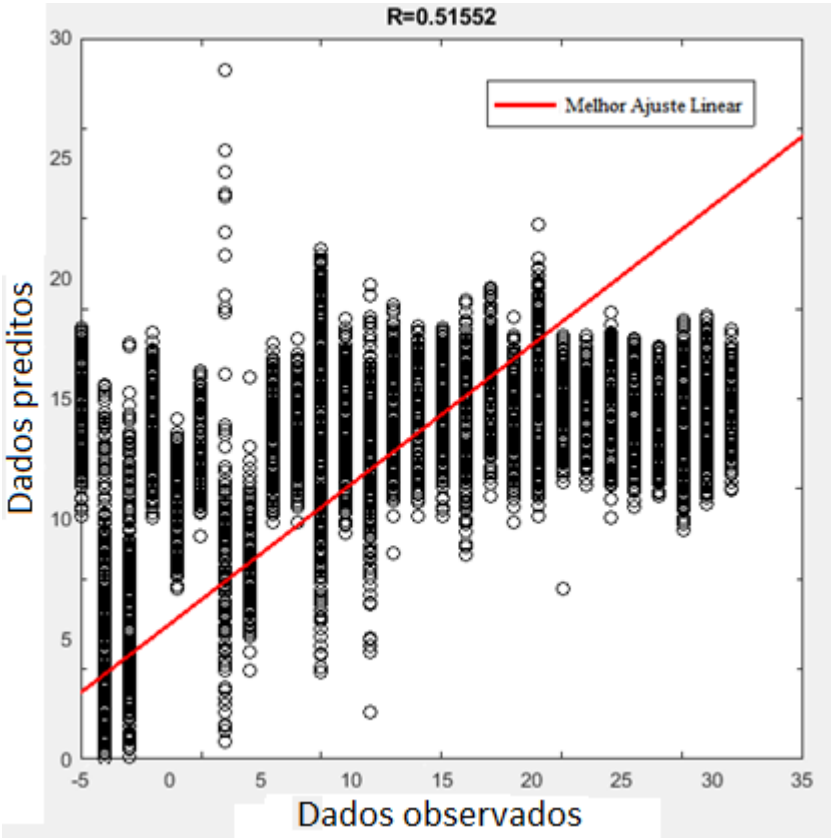


Figura 19: Gráfico para o treinamento da rede *Echo State*.

Pode-se concluir que a rede *Echo State*, apesar da inigualável velocidade de simulação, não foi capaz de alcançar um resultado satisfatório para capacidade de aprendizado e predição, tornando dispensável a análise do SSE. A figura 20, gráfico de dispersão da validação, também prova que a rede não conseguiu generalizar o conjunto de dados, apresentando um comportamento praticamente

linear e aproximando quase todos os resultados de apenas um tipo de falha. Contudo algumas falhas obtiveram um bom R^2, entre elas a falha do tipo 16, a mesma que não havia obtido bom resultado tanto para feedforward, quanto para Elman, o que significa que seria interessante realizar a identificação da maioria das falhas pelas redes anteriores e incrementar o reconhecimento da falha do tipo 16 por meio da rede Echo State.

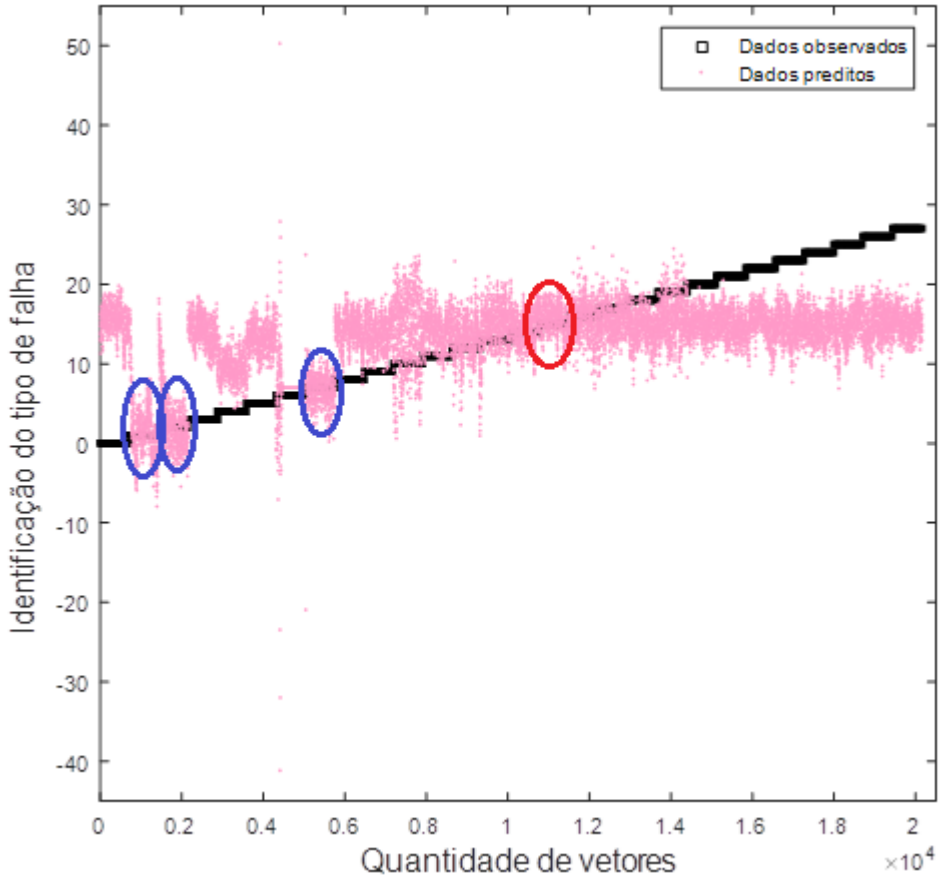


Figura 20: Gráfico de dispersão da validação para rede *Echo State* que apresentou os melhores resultados.

Na dissertação de SOARES (2017), o extenso banco de dados também é formado pelo processo TE e modelado por uma ESN para detectar e identificar falhas no processo. Contudo, diferente do presente trabalho, o dele revelou bons resultados de predição, o que significa que o método aqui utilizado pode ser melhorado no futuro. O desempenho ruim da ESN provavelmente se deve ao fato de não utilizar os mesmos algoritmos de treinamento que as redes *feedforward* e de *Elman*, *Levenberg Marquardt* e regularização *Bayesiana*.

5. Conclusões e sugestões para trabalhos futuros

Neste trabalho foi avaliado o desempenho de redes neurais artificiais na detecção e identificação de falhas de processos. Foi realizado um estudo de caso através do benchmark de processo *Tennessee Eastman*, um problema de controle baseado em um processo industrial real, que apresenta um modelo não-linear de um sistema multicomponente bastante complexo.

O conjunto de dados gerado pelo processo foi de enorme volume (dez milhões de dados), reproduzindo um contexto de *Big Data*. Este banco de dados foi modelado por meio de redes do tipo *feedforward* e recorrentes.

As redes *feedforward* apresentaram boa capacidade de aprendizado, predição e dimensionalidade de dados, através da analise dos parâmetros coeficiente de determinação para treinamento e teste (R^2) e soma dos quadrados dos erros (SSE), que obtiveram os valores 90,026%, 88,692% e 4100, respectivamente.

As redes de *Elman* também apresentaram boa capacidade de aprendizado, predição e dimensionalidade de dados, obtendo para os parâmetros R^2 de treinamento e teste e SSE, os valores de 84,237%, 83,637%, e 629, respectivamente. É importante notar que devido à capacidade computacional da máquina trabalhada, a rede de *Elman* encontrou limite de topologia em uma rede de duas camadas intermediárias com 55 neurônios na primeira e 45 na segunda.

As *Echo State Network*, apesar de serem incomparáveis quanto à velocidade de simulação, não foram capazes de alcançar resultados satisfatórios para capacidade de aprendizado e predição, já que obtiveram valores de R^2 de treinamento e teste baixos, 53,008% e 51,552%, respectivamente. Devido a esses valores, provou-se que a rede não alcançou desempenho suficiente para adaptação da maior parte dos dados modelados e, por consequência, foi dispensável a análise do SSE.

Foi comprovado que as redes *feedforward* e de *Elman* adaptaram-se bem ao conjunto de dados modelados, sendo que a rede *feedforward* foi a responsável pelas melhores configurações de redes neurais, analisada pelos resultados dos

parâmetros R^2 e SSE. Ainda sim, apesar das redes apresentarem ótimas generalizações dos dados, o gráfico de dispersão comprovou que errou na detecção de dois tipos de falhas específicas: no padrão do sistema sem alterações e em um desconhecido. Contudo, a rede Echo State, apesar de não ter obtido um resultado satisfatório na maior parte dos resultados, foi capaz de realizar a identificação da falha do tipo desconhecido que nem a feedforward nem a Elman conseguiram, o que gerou a possibilidade de se trabalhar com a união de redes. Desta maneira, a melhor configuração seria através do uso da rede feedforward para identificação de todas as falhas menos a do tipo 16, a qual identificada pela rede Echo State.

O resultado desta dissertação indica que as redes *feedforward* e de *Elman* podem ser usados com boa robustez em controle preditivo de falhas em processos industriais, podendo ser aplicadas em plantas químicas no futuro.

Para dar continuidade ao trabalho, sugere-se:

- Aplicação de algoritmos genéticos (GA) para otimização do treinamento das redes neurais;
- Avaliar a utilização de uma rede neural não supervisionada;
- Utilizar máquinas mais potentes para aumentar a possibilidade de topologias das redes neurais, principalmente da rede de *Elman*.

6. Referências Bibliográficas

AZEVEDO, MARCELO TEIXEIRA DE. Transformação Digital na Indústria: Indústria 4.0 e a Rede de Água Inteligente do Brasil. Universidade de São Paulo, 2017. Tese (Mestrado).

BATHELT, A. N., RICKER, L, JELALI, M. Revision of the Tennessee Eastman process model. 9th INTERNATIONAL SYMPOSIUM ON ADVANCED CONTROL OF CHEMICAL PROCESSES, 309-314, 2015.

BRAGA, A. P., CARVALHO, A. P. L. F., LUDERMIR, T. B. Redes Neurais Artificiais, Rio de Janeiro: LTC Editora, 2007.

CITANDO MODA SITE BLOG. https://citandomodasiteblog.wordpress.com/2017/01/28/1a-revolucao-industrial-1760-1830/ Acessado em: 16-08-2018.

CHIANG, L. H., RUSSELL, E. L., BRAATZ, R. D. Fault diagnosis in chemical processes using Fisher discriminant analysis, discriminant partial least squares, amd principal component analysis. Journal of Chemometrics and Intelligent Laboratory Systems, 2000, p. 243-252, 1999.

DOWNS, J. J. e VOGEL, E. F. A plant-wide industrial process problem. Journal of Computers & Chemical engineering. v. 17, n. 3, p. 245-255, 1993.

EMBARCADOS. https://www.embarcados.com.br/redes-neurais-artificiais/ Acessado em: 02-08-2018.

FRANCISCO, CLAUDIO DE OLIVEIRA. Modelagem e simulação de um secador industrial de gelatin através de redes neurais artificiais. Universidade Estadual de Campinas, 2000. Tese (Mestrado).

GAO X. Z. e OVASKA S. J. Genetic algorithm training of Elman network in motor fault detection. Journal of Neural Computing & Applications, v. 11, p. 37-44, 2002.

GARTNER. Big Data. https://www.gartner.com/it-glossary/big-data. Acessado em: 10-08-2018.

GERMANO, A. L., GUEDES, L. A., COSTA, B. S. J., BEZERRA, C. G. Detecção de falhas no processo Tennessee Eastman utilizando métricas de tipicidade e excentricidade. XXI CONGRESSO BRASILEIRO DE AUTOMÁTICA, Vitória, ES, 2016

GROSSI, CAROLINE DIAS. Modelagem com redes neurais para predição do crescimento microbiano em reator batelada. Pontifícia Universidade Católica do Rio de Janeiro, 2017. Trabalho de Conclusão de Curso.

GRUPO MULT. http://www.grupomult.com.br/a-importancia-da-integracao-de-sistemas-de-producao-na-industria-4-0/ Acessado em: 16-08-2018.

GUACAS, DANIEL ALBERTO MENESES. Otimização de Echo StateNeural Networkcom Algorítmos Genéticos para a Ponderação Dinâmica de Previsores. Pontifícia Universidade Católica do Rio de Janeiro, 2016. Tese (Mestrado).

ISERMANN, ROLF. Fault diagnosis of diesel engines. Mechanical Engineering, New York, v. 135, n. 12, p. 11-6B, 7B, 8B, 9B, 10B, 11B, 12B, 13B, 14B,15B, 14B1, 12, 2013.

JAEGER, HERBERT. The “echo state” approach to analysing and training recurrent neural network – with na Erratum note. GMD Report 148, 1-47, 2010

JAMIL, M., SHARMA, S. K., SINGH, R. Fault detection and classification in eletrical power transmission system using artificial neural network. Journal of SpringerPlus, 2015.

JÚNIOR, ALDAYR DANTAS DE ARAUJO. Predição não-linear de curvas de produção de petróleo via redes neurais recursivas. Universidade Federal do Rio Grande do Norte, 2010. Tese (Mestrado).

KAYRI, MURAT. Predictive abilities of Bayesin Regularization and Levenberg-Marquardt algorithms in artificial neural networks: a comparative empirical study on social data. Journal of Mathematical and Computational Applications, 2016.

LAU, C. K., GHOSH, K., HUSSIAN, M. A., HASSAN, C. R. C. Fault diagnosis of Tennessee Eastman process with multi-scale PCA and ANFIS. Journal of Chemometrics and Intelligent Laboratory Systems, p. 1-14, 2012.

MARTINS, GLAUCO PEREIRA DE MORAES. Modelo híbrido baseado em redes neurais Echo State Network otimizadas por Algoritmos Evolucionários para indetificação automática de sistemas dinâmicos e a sua aplicação à identificação do nível do gerador de vapor de uma usina nuclear PWR. Pontifícia Universidade Católica do Rio de Janeiro, 2016. Tese (Mestrado).

MATLAB. Trainbr. https://www.mathworks.com/help/nnet/ref/trainbr.html. Acessado em: 08-08-2018.

MATLAB. Trainlm. https://www.mathworks.com/help/nnet/ref/trainlm.html. Acessado em: 08-08-2018.

MAURYA, M. R., RENGASWAMY, R., VENKATASUBRAMANIAN, V. Applications of signed digraphs-based analysis for fault diagnosis of chemical process flowsheets. Journal of Engineering Applications of Artificial Intelligence, v. 17, p. 501-518, 2004.

MCCARTHY, J., MINSKY, M. L., ROCHESTER, N., SHANNON, C. E. A proposal for the Dartmouth summer research project on artificial intelligence. DARTMOUTH ARTIFICIAL INTELLIGENCE CONFERENCE, 1955.

MORELLO, B. C., MALINOWSKI, S., SENOUSSI, H. Feature selection for fault detection system: application to the Tennessee Eastman process. Journal of Applied Intelligence, v. 44, p. 111-122, 2016.

PENSAMENTO VERDE. https://www.pensamentoverde.com.br/dicas/confira-dicas-computador-antigo/ Acessado em: 16-08-2018.

PONTO FINAL MACAU. https://pontofinalmacau.wordpress.com/2017/03/01/consumo-de-carvao-caiu-47-por-cento-em-2016/ Acessado em: 16-08-2018.

QUATRO RODAS. https://quatrorodas.abril.com.br/noticias/barril-de-petroleo-vale-menos-que-o-proprio-tambor/ Acessado em: 16-08-2018.

RAD, M. A. A. e YAZDANPANAH, M. J. Designing supervised local neural network classifiers based on EM clustering for fault diagnosis of Tennessee Eastman process. Journal of Chemometrics and Intelligent Laboratory Systems, v. 146, p. 149-157, 2015.

RATO, T. J. e REIS, M. S. Fault detectionin the Tennesse Eastman benchmark process using dynamic principal components analysis based on decorrelated residuals (DPCA-DR). Journal of Chemometrics and Intelligent Laboratory Systems, v. 125, p. 101-108, 2013.

REAMP. http://www.reamp.com.br/blog/2018/01/inteligencia-artificial-criara-mais-empregos-do-que-elimina-ate-2020/ Acessado em: 16-08-2018.

REGO, ARTUR SERPA DE CARVALHO. Otimização dos parâmetros da deslignização do bagaço de cana-de-açucar com peróxido de hidrogênio alcalino através do modelo neural. Pontifícia Universidade Católica do Rio de Janeiro, 2017. Tese (Mestrado).

RIBEIRO, A. M., GROSSI, C. D., SANTOS, B. F., SANTOS, R. B., FILETI, A. M. F. Leak Detection Modeling Of a Pipeline Using Echo State Neural Networks. Journal of Computer Aided Chemical Engineering, v. 43, p. 1231-1236, 2018.

RICKER, N. LAWRENCE, Decentralized control of the Tennessee Eastman challenge process. Journal of Process Control, v. 6, p. 205-221, 1995.

RUEDA, CAMILO VELASCO. EsnPredictor: Ferramenta de previsão de series temporais baseada em Echo State Network optimizadas por algoritmos genéticos e Particle Swarm Optimazation. Pontifícia Universidade Católica do Rio de Janeiro, 2014. Tese (Mestrado).

RUSSEL, E. L., CHIANG, L. H., BRAATZ, R. D. Fault detection in industrial processes using canonical variate analysis and dynamic principal component analysis. Journal of Chemometrics and Intelligent Laboratory Systems, v. 51, p. 81-93, 2000.

SANTOS, REJANE BARBOSA. Detecção e localização de vazamentos em tubulações de gás de baixa pressão por meio de sensors acústicos e processamento matemático de sinais. Universidade Estadual de Campinas, 2015. Tese (Doutorado).

SHUTTER STOCK. https://www.shutterstock.com/image-photo/photo-two-black-microchip-green-printed-1180375837 Acessado em: 16-08-2018.

SOARES, FELIPO DOVAL ROJAS. Técnicas de *Machine Learning* Aplicadas a Inferência e Detecção e Diagnóstico de Falhas de Processos Químicos Industriais em Contexto *Big Data*. Universidade Federal do Rio de Janeiro, 2017. Tese (Mestrado).

SRINIVASA, S. e BHATNAGAR, V. Big Data Analytics, New Delhi, India: Springer, 2012.

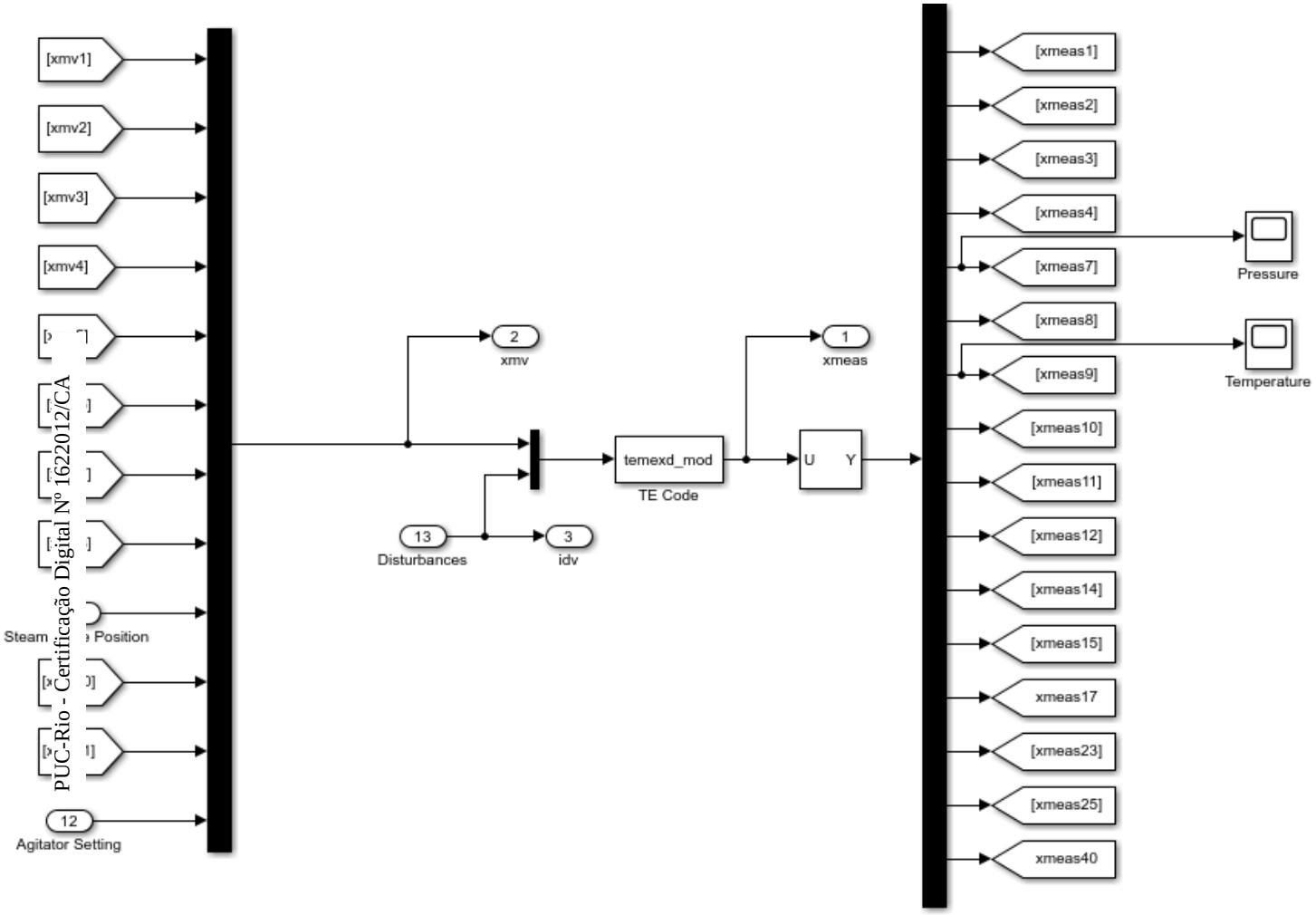
SUPER TESLAS. http://superteslas.blogspot.com/2014/04/eletromagnetismo-o-que-e.html Acessado em: 16-08-2018.

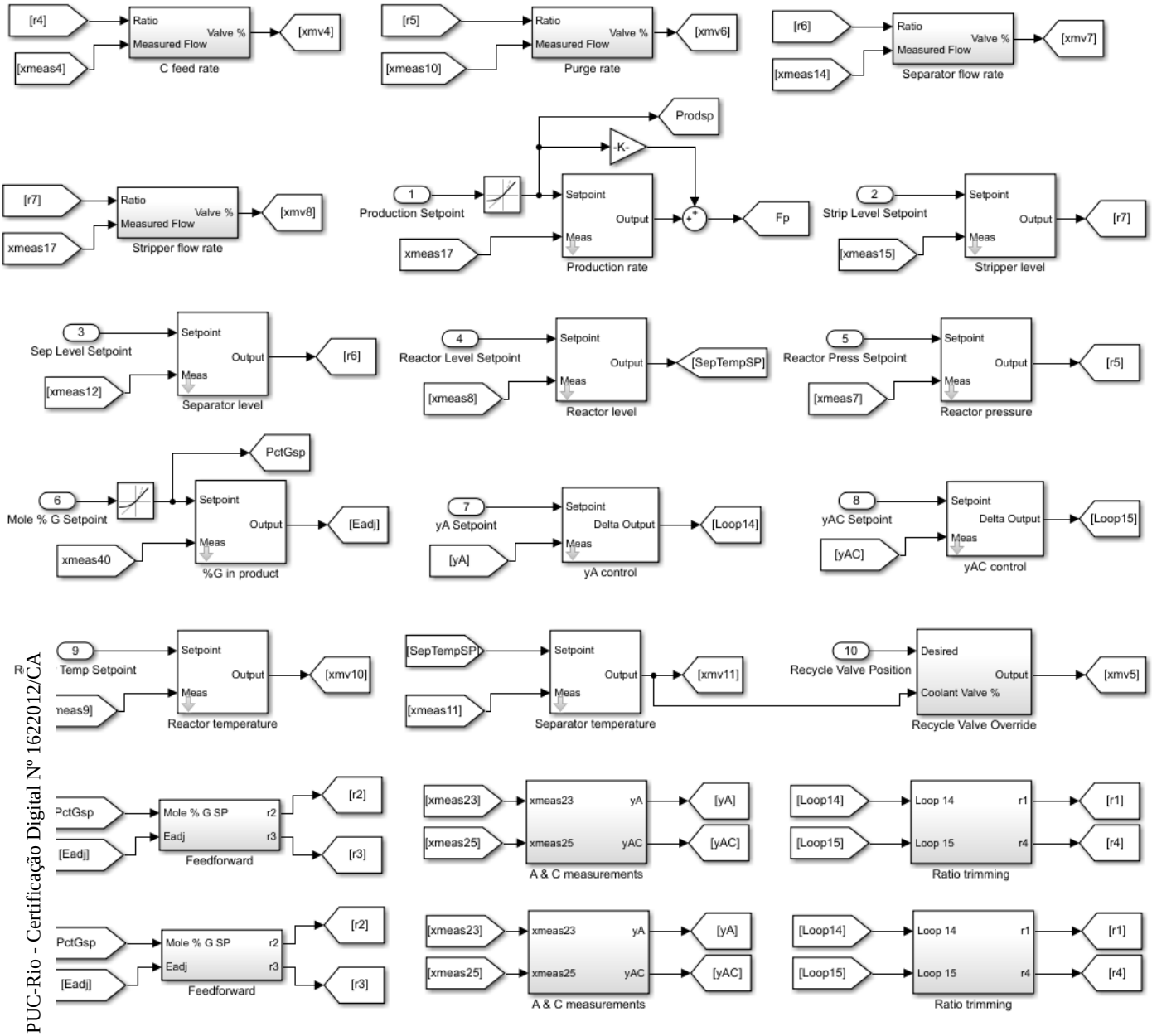
TORRECILLA, J. L. e ROMO, J. Data learning from Big Data. Journal of Statistics and Probability Letters. v. 136, p. 15-19, 2018.

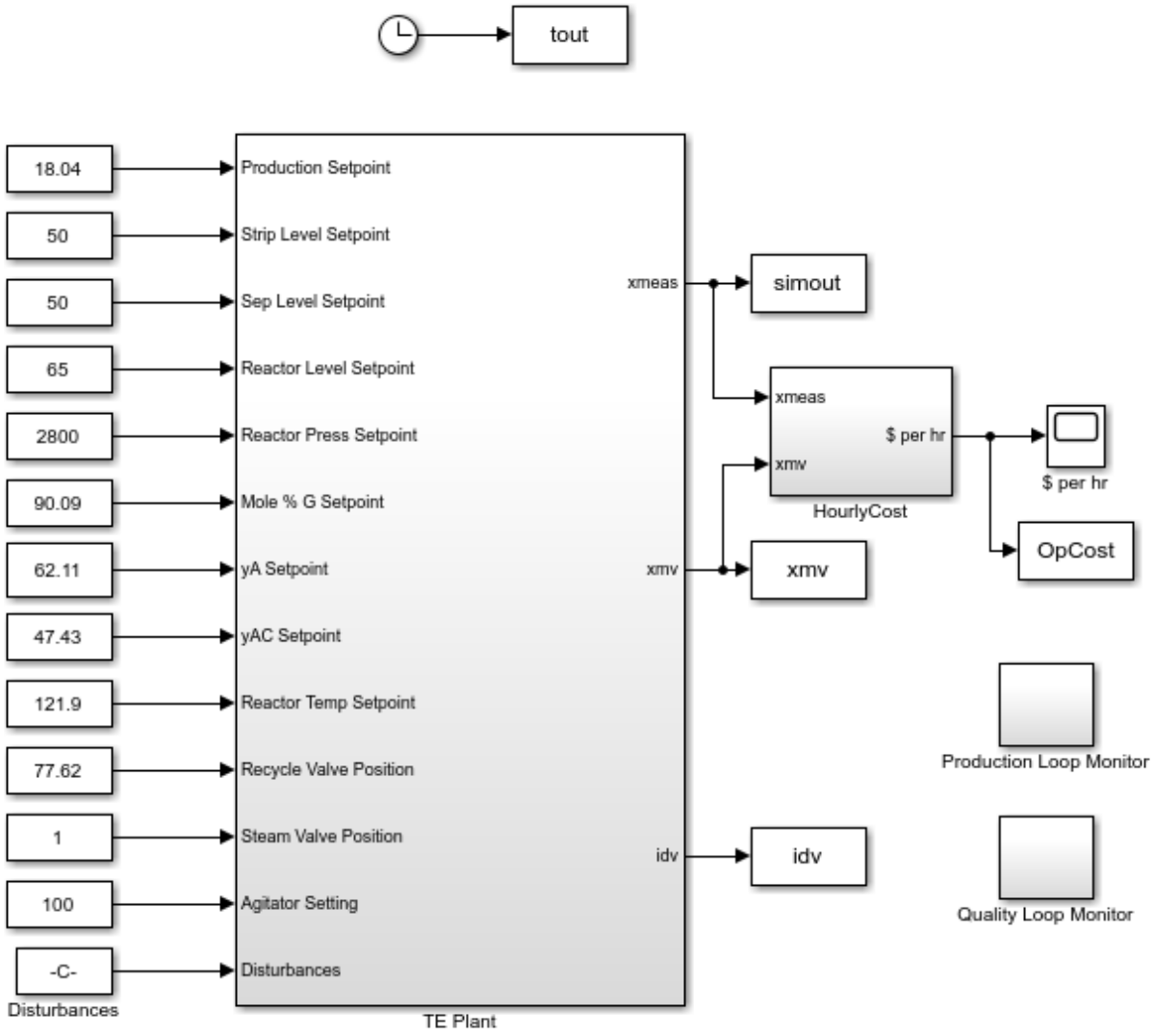
YOUSEFI F., MOHAMMADIYAN, S., KARIMI, H. Application of artificial neural network and PCA to predict the termal conductivities of nanofluids. Journal of Heat and Mass Transfer, v. 52, p. 2141-2154, 2016.

7. Anexos

7.1. Blocos do processo Tennessee Eastman contruídos no Simulink







7.2. Programa de separação dos dados de treinamento no software Excel

```
Sub Treino1()
Dim V(120000, 100) As Double
Dim Aba, i, j, x, y, t, c As Integer

Application.ScreenUpdating = False
y = -1

'O programa é grande demais e não aguenta ser rodado inteiro, por isso é necessário fazer pausas de 5 em 5 abas transcritas. Continue na próxima macro.
For Aba = 1 To 5
   y = -1
   'Área responsavel pela coleta de dados
   Worksheets("Dado" & Aba).Activate
   For j = 1 To 81
```

```
    x = 0
    y = y + 1
    t = 0
    For i = 3 To 7203
       If t < 6 Then
          V(x, y) = Cells(i, j)
          x = x + 1
          t = t + 1
       Else
          i = i + 3
          V(x, y) = Cells(i, j)
          t = 0
       End If
    Next i
    Next j
    'Área responsavel pela transferência dos dados
    Worksheets("Treino").Activate
    'Para localizar onde pararam os dados da aba anterior
    x = 2
    y = 1
    Do While Cells(x, y) <> ""
       x = x + 1
       c = x
    Loop
    'Transferência de dados
    y = -1
    For j = 1 To 81
    x = 0
    y = y + 1
    For i = c To c + 4320
       Cells(i, j) = V(x, y)
       x = x + 1
    Next i
    Next j
Next Aba

Application.ScreenUpdating = True

End Sub
```


7.3. Programa de separação dos dados de teste no software Excel

```
Sub Teste1()
Dim V(40000, 100) As Double
Dim Aba, i, j, x, y, t, c As Integer

Application.ScreenUpdating = False
y = -1
```

```
'O programa é grande demais e não aguenta ser rodado inteiro, por isso é necessário fazer pausas de 14 em 14 abas transcritas. Continue na próxima macro.
For Aba = 1 To 14
   y = -1
   'Área responsavel pela coleta de dados
   Worksheets("Dado" & Aba).Activate
   For j = 1 To 81
   x = 0
   y = y + 1
   t = 0
   For i = 9 To 7203
      If t < 2 Then
         V(x, y) = Cells(i, j)
         x = x + 1
         t = t + 1
      Else
         i = i + 8
         V(x, y) = Cells(i, j)
         x = x + 1
         t = 1
      End If
   Next i
   Next j
   'Área responsavel pela transferência dos dados
   Worksheets("Teste").Activate
   'Para localizar onde pararam os dados da aba anterior
   x = 2
   y = 1
   Do While Cells(x, y) <> ""
      x = x + 1
      c = x
   Loop
   'Transferência de dados
   y = -1
   For j = 1 To 81
   x = 0
   y = y + 1
   For i = c To c + 1439
      Cells(i, j) = V(x, y)
      x = x + 1
   Next i
   Next j
Next Aba

Application.ScreenUpdating = True

End Sub
```

7.4. Programa de separação dos dados de validação no software Excel

```
Sub Validacao1()
Dim V(40000, 100) As Double
Dim Aba, i, j, x, y, t, c As Integer

Application.ScreenUpdating = False
y = -1

'O programa é grande demais e não aguenta ser rodado inteiro, por isso é necessário fazer pausas de 14 em 14 abas transcritas. Continue na próxima macro.
For Aba = 1 To 14
   y = -1
   'Área responsavel pela coleta de dados
   Worksheets("Dado" & Aba).Activate
   For j = 1 To 81
   x = 0
   y = y + 1
   t = 0
   For i = 11 To 7203
      If t < 2 Then
         V(x, y) = Cells(i, j)
         x = x + 1
         t = t + 1
      Else
         i = i + 8
         V(x, y) = Cells(i, j)
         x = x + 1
         t = 1
      End If
   Next i
   Next j
   'Área responsavel pela transferência dos dados
   Worksheets("Validacao").Activate
   'Para localizar onde pararam os dados da aba anterior
   x = 2
   y = 1
   Do While Cells(x, y) <> ""
      x = x + 1
      c = x
   Loop
   'Transferência de dados
   y = -1
   For j = 1 To 81
   x = 0
   y = y + 1
   For i = c To c + 1439
      Cells(i, j) = V(x, y)
      x = x + 1
```

```
    Next i
    Next j
Next Aba

Application.ScreenUpdating = True

End Sub
```

7.5. Programa de treinamento de redes neurais artificiais do tipo feedforward e Elman no software Matlab

```
clear all;
clc;
M = load('train50menosV.dat'); %carrega o arquivo de dados%
M = M'; %calcula a matriz transposta%
entrada = M(1:36,:); %define os dados de entrada da rede%
saida = M(37,:); %define os dados de saída da rede%
[entradan,minentrada,maxentrada,saidan,minsaida,maxsaida]=premnmx(
entrada,saida);
%define os parâmetros máximos e mínimos das matrizes de entrada e
%saída
%e faz normalização%
%net.numinputs = size(entrada(:,:),5); %n° variáveis de entrada%
%net.numlayers = 2; %n° de camadas sem a camada de entrada%
net =
newelm(minmax(entradan(:,:)),[45,55,1],{'logsig','tansig','purelin
'},'trainbr'); %newff=feedforward, newelm=Elman

%cria a rede, definindo número de neuronios e função de ativação
%das camadas%
%intermediária e de s   aída%
net.trainParam.epochs = 100; %n° de passos%
net.trainParam.goal = 1e-4; %convergência desejada%
net.initFcn = 'initlay'; %função que inicia os pesos e bias%
net.performFcn = 'sse'; %função objetivo a ser minimizada%
net.trainParam.min_grad = 1e-100; %mínimo gradiente%
%net.trainParam.mu_max = 1e+400; %máximo MU%
net = init(net);
[net,tr] = train(net,entradan(:,:),saidan(:,:)); %realiza o
%treinamento da rede%
%pesos e bias da rede determinados e guardados em 'net'%
Y = sim(net,entradan(:,:)); %simula com os dados de entrada do
X = postmnmx(Y,minsaida,maxsaida); %desnormaliza dados de saida%
figure(1);
[m,b,r]=postreg(X(1,:),saida(1,:)); %gráfico da saída real versus
%calculada%
figure(2);
plot(saida(1,:),'-k') %grafica a saida real do arquivo de
%treinamento%
hold on
plot(X(1,:),'or'); %grafica a saida calculada pela rede%
xlabel('Sample','FontSize',15); %rótulo eixo x, fonte tamanho
%20%
ylabel('Brand','FontSize',15); %rótulo eixo y, fonte tamanho 20%
legend('saída real','saída calculada','northeast'); %legenda no
canto
%superior direito%
hold off
```

```
save 'rna15' net; %salva a rede criada%

load rna15; %carrega a rede gravada
M = load('train50menosV.dat'); %carrega arquivo de dados para
%treinamento, para pegar a mesma normalização no treinamento e no
%teste%
M = M'; %transposta%
intre = M(1:36,:); %dados de entrada da rede%
outtre = M(37,:); %dados de saida da rede%
[intren,minintre,maxintre,outtren,minouttre,maxouttre]=premnmx(int
re,outtre);
%normaliza dados%
N = load('test50menosV.dat'); %carrega arquivo de dados para
teste%
N = N';
in = N(1:36,:); %dados de entrada da rede%
out = N(37,:); %dados de saida da rede%
[inn] = tramnmx(in,minintre,maxintre); %normaliza dados%
Y = sim(net,inn(:,:)); %simula com os dados de entrada do arquivo
%de teste%
X = postmnmx(Y,minouttre,maxouttre); %desnormaliza dados de saida%
figure(3);
plot(out(1,:),'-k') %grafica a saida real do arquivo de teste%
hold on
plot(X(1,:),'om'); %grafica a saida calculada pela rede%
xlabel('Sample','FontSize',15); %rótulo eixo x, fonte tamanho
%20%
ylabel('Brand','FontSize',15); %rótulo eixo y, fonte tamanho 20%
legend('observed','predicted','northeast'); %legenda no canto
%superior direito%
colordef white;
hold off
figure(4);
[m,b,r]=postreg(X(1,:),out(1,:)); %grafico da saida real versus
%calculada%
xlabel('T','FontSize',20); %rótulo eixo x, fonte tamanho 20%
ylabel('A','FontSize',20); %rótulo eixo y, fonte tamanho 20%
whitebg([1 1 1]);
colordef white;
net.IW{1}
net.LW{2}
net.LW{6}
net.b{1}
net.b{2}
net.b{3}
```

7.6. Programa de treinamento de redes neurais artificiais do tipo Echo State no software Matlab

```
close all;
clear all;
clc;
colordef white; %cor de fundo dos gráficos

%Carregamento dos dados:
arq_tr = load('train50menosV.dat'); %dados do treinamento
arq_test = load('test50menosV.dat'); %dados do teste
arq_valid = load('validation50menosV.dat'); %dados da validação
```

```
%Separação dos dados em entradas e saídas
in_tr = arq_tr(:,1:36);
out_tr = arq_tr(:,37);
in_test = arq_test(:,1:36);
out_test = arq_test(:,37);
in_valid = arq_valid(:,1:36);
out_valid = arq_valid(:,37);

%Normalização dos dados
[in_tr_norm, ft_in_tr_norm] = mapminmax(in_tr', 0, 1);
[out_tr_norm, ft_out_tr_norm] = mapminmax(out_tr', 0, 1);
[in_test_norm, ft_in_test_norm] = mapminmax(in_test', 0, 1);
[out_test_norm, ft_out_test_norm] = mapminmax(out_test', 0, 1);
[in_valid_norm, ft_in_valid_norm] = mapminmax(in_valid', 0, 1);
[out_valid_norm, ft_out_valid_norm] = mapminmax(out_valid', 0, 1);


%-------------------------------------------------------------------
-------------------------------------------------

%Criando a ESN

nInputUnits = 36; %número de neurônios de entrada
nOutputUnits = 1; %número de neurônios de saída

% Hiperparâmetros
nInternalUnits = 200; %número de neurônios do reservatório
conect = 0.755; %Nível de conectividade entre os neurônios do
reservatório
sWin = 0.65; %fator de ajuste dos pesos de entrada
sWf = 0.0001; %fator de ajuste dos pesos de saída
raioEsp = 0.99999;
leakyRate = 0.99999;

esn = generate_esn(nInputUnits, nInternalUnits, nOutputUnits,
conect, sWin, sWf,
'spectralRadius',raioEsp,'inputScaling',ones(nInputUnits,1),'input
Shift',zeros(nInputUnits,1), ...
    'teacherScaling',[1],'teacherShift',[0],'feedbackScaling',
ones(nOutputUnits,1), ...
    'type', 'leaky_esn', 'leakage', leakyRate,
'reservoirActivationFunction', 'tanh', 'outputActivationFunction',
'identity', 'methodWeightCompute', 'pseudoinverse',
'inverseOutputActivationFunction', 'identity');
    %plain_esn usa neurônios estáticos. Para ela, leakage=1.
    %leaky_esn usa neurônios dinâmicos. Tenho que ajustar leakage
entre 0-1

%Ajuste dos pesos do reservatório de acordo com o raio spectral
escolhido
esn.internalWeights = esn.spectralRadius *
esn.internalWeights_UnitSR;

%-------------------------------------------------------------------
-------------------------------------------------

% Treinamento da ESN
nForgetPoints = 0 ; %quantidade de pontos iniciais que serão
descartados
```

```
[trainedEsn stateMatrix] = ...
    train_esn(in_tr_norm', out_tr_norm', esn, nForgetPoints);


%-------------------------------------------------------------------
-------------------------------------------------

%Simulação da rede treinada

[predictedTrainOutput, last_state_tr] = test_esn(in_tr_norm',
trainedEsn, nForgetPoints);
[predictedTestOutput, last_state_test] = test_esn(in_test_norm',
trainedEsn, nForgetPoints, 'startingState', last_state_tr ) ;
[predictedValidOutput, last_state_test] = test_esn(in_valid_norm',
trainedEsn, nForgetPoints, 'startingState', last_state_tr ) ;


%-------------------------------------------------------------------
-------------------------------------------------


%Desnormalização dos dados
out_calc_tr_desnorm = mapminmax('reverse', predictedTrainOutput,
ft_out_tr_norm);
out_calc_test_desnorm = mapminmax('reverse', predictedTestOutput,
ft_out_tr_norm);
out_calc_valid_desnorm = mapminmax('reverse',
predictedValidOutput, ft_out_tr_norm);

%-------------------------------------------------------------------
-------------------------------------------------

% GRÁFICOS

%Gráficos do Treinamento ------------------------------------------
----------------------------------------

%Criação da reta de ajuste linear para o treinamento
[coef_curve1, error1] = polyfit(out_calc_tr_desnorm,out_tr,1);
%define os coeficientes da reta de ajuste 1
Y_curve1 = polyval(coef_curve1, out_calc_tr_desnorm, error1);
%define a variável y da reta de ajuste
X_curve1 = out_calc_tr_desnorm; %define a variável x da reta de
ajuste

%Criação da reta modelo
x= 0 : 1;
y = x;

%Cálculo do R2 e dos coeficientes de erro da reta 1
r2_curve1 = calc_r2(out_calc_tr_desnorm,out_tr);
sse_curve1 = sse(trainedEsn, out_tr, out_calc_tr_desnorm);
mse_curve1 = immse(out_tr, out_calc_tr_desnorm);
rmse_curve1 = sqrt(mse_curve1);

%Gráficos do treinamento
figure(1)
subplot(1,2,1);
```

```
plot(out_calc_tr_desnorm,out_tr,'or'); %gráfico da saída real x
saída calculada
hold on
plot(X_curve1,Y_curve1,'-k'); %reta de melhor ajuste dos pontos
(reta calculada)
plot(x,y,'--b'); %reta modelo (y = x)
title1 = strcat('Title - Train   R^2 =', num2str(r2_curve1));
title(title1,'FontSize',11); %título do primeiro gráfico
xlabel('X - Train','FontSize',9); %legenda do eixo x: valores
calculados na simulação
ylabel('Y - Train','FontSize',9); %legenda do eixo y: valores
fornecidos pelo banco de dados
hold off

subplot(1,2,2);
plot(out_tr,'-k'); %grafico dos valores fornecidos pelo banco de
dados
hold on
plot(out_calc_tr_desnorm,'vm'); %gráfico dos valores calculados na
simulação
xlabel('Samples - Train','FontSize',9); %legenda do eixo x: número
de ensaios
ylabel('Variable - Train','FontSize',9); %legenda do eixo y:
variável de saída
legend('original value - Train','sim value - Train'); %legenda do
gráfico
title('Variable vs Samples','FontSize',11); %título do gráfico
hold off

savefig 'graph_train.fig'

%Gráficos do Teste ----------------------------------------------
---------

%Criação da reta de ajuste linear para o teste
[coef_curve2, error2] = polyfit(out_calc_test_desnorm,out_test,1);
%define os coeficientes da reta de ajuste 1
Y_curve2 = polyval(coef_curve2, out_calc_test_desnorm); %define a
variável y da reta
X_curve2 = out_calc_test_desnorm; %define a variável x da reta

%Cálculo do R2 e dos coeficientes de erro da reta 2
r2_curve2 = calc_r2(out_calc_test_desnorm,out_test);
sse_curve2 = sse(trainedEsn, out_test, out_calc_test_desnorm);
mse_curve2 = immse(out_test, out_calc_test_desnorm);
rmse_curve2 = sqrt(mse_curve2);

%Gráficos do teste
figure(2)
subplot(1,2,1);
plot(out_calc_test_desnorm,out_test,'or'); %gráfico da saída real
x saída calculada
hold on
plot(X_curve2,Y_curve2,'-k'); %reta de melhor ajuste dos pontos
(reta calculada)
plot(x,y,'--b'); %reta modelo (y = x)
title1 = strcat('Title - Test   R^2 =', num2str(r2_curve2));
title(title1,'FontSize',11); %título do primeiro gráfico
xlabel('X - Test','FontSize',9); %legenda do eixo x: valores
calculados na simulação
```

```
ylabel('Y - Test','FontSize',9); %legenda do eixo y: valores
fornecidos pelo banco de dados
hold off

subplot(1,2,2);
plot(out_test,'-k'); %grafico dos valores fornecidos pelo banco de
dados
hold on
plot(out_calc_test_desnorm,'vm'); %gráfico dos valores calculados
na simulação
xlabel('Samples - Test','FontSize',9); %legenda do eixo x: número
de ensaios
ylabel('Variable - Test','FontSize',9); %legenda do eixo y:
variável de saída
legend('original value - Test','sim value - Test'); %legenda do
gráfico
title('Variable vs Samples','FontSize',11); %título do gráfico
hold off

savefig 'graph_test.fig'

%Gráficos da Validação ----------------------------------------
-------------

%Criação da reta de ajuste linear para a validação
[coef_curve3, error3] =
polyfit(out_calc_valid_desnorm,out_valid,1); %define os
coeficientes da reta de ajuste 1
Y_curve3 = polyval(coef_curve3, out_calc_valid_desnorm); %define a
variável y da reta
X_curve3 = out_calc_valid_desnorm; %define a variável x da reta

%Cálculo do R2 e dos coeficientes de erro da reta 3
r2_curve3 = calc_r2(out_calc_valid_desnorm,out_valid);
sse_curve3 = sse(trainedEsn, out_valid, out_calc_valid_desnorm);
mse_curve3 = immse(out_valid, out_calc_valid_desnorm);
rmse_curve3 = sqrt(mse_curve3);

%Gráficos da Validação
figure(3)
subplot(1,2,1);
plot(out_calc_valid_desnorm,out_valid,'or'); %gráfico da saída
real x saída calculada
hold on
plot(X_curve3,Y_curve3,'-k'); %reta de melhor ajuste dos pontos
(reta calculada)
plot(x,y,'--b'); %reta modelo (y = x)
title1 = strcat('Title - Valid   R^2 =', num2str(r2_curve3));
title(title1,'FontSize',11); %título do primeiro gráfico
xlabel('X - Valid','FontSize',9); %legenda do eixo x: valores
calculados na simulação
ylabel('Y - Valid','FontSize',9); %legenda do eixo y: valores
fornecidos pelo banco de dados
hold off

subplot(1,2,2);
plot(out_valid,'-k'); %grafico dos valores fornecidos pelo banco
de dados
hold on
```

```
plot(out_calc_valid_desnorm,'vm'); %gráfico dos valores calculados
na simulação
xlabel('Samples - Valid','FontSize',9); %legenda do eixo x: número
de ensaios
ylabel('Variable - Valid','FontSize',9); %legenda do eixo y:
variável de saída
legend('original value - Valid','sim value - Valid'); %legenda do
gráfico
title('Variable vs Samples','FontSize',11); %título do gráfico
hold off

savefig 'graph_valid.fig'

%Salva os valores encontrados em todas as variáveis
save 'name.mat'
```