UNIVERSIDADE FEDERAL DO PARANÁ

JÚLIO CÉSAR ROYER

METODOLOGIA PARA A DEFINIÇÃO DE ALERTAS DINÂMICOS PARA SUBPRESSÃO EM FUNDAÇÕES DE BARRAGENS DE CONCRETO: O CASO DA BARRAGEM PRINCIPAL DE ITAIPU

CURITIBA
2016

JÚLIO CÉSAR ROYER

METODOLOGIA PARA A DEFINIÇÃO DE ALERTAS DINÂMICOS PARA SUBPRESSÃO EM FUNDAÇÕES DE BARRAGENS DE CONCRETO: O CASO DA BARRAGEM PRINCIPAL DE ITAIPU

Tese apresentada ao Programa de Pós-Graduação em Métodos Numéricos em Engenharia do Setor de Tecnologia da UFPR, como requisito parcial para a obtenção do grau de Doutor em Métodos Numéricos em Engenharia, na área de concentração de Programação Matemática e na linha de pesquisa de Métodos Estatísticos aplicados à Engenharia.

Orientador: Prof. Dr. Volmir Eugênio Wilhelm
Coorientadora: Dra. Josiele Patias

CURITIBA
2016

Royer, Júlio César
Metodologia para a definição de alertas dinâmicos para subpressão em fundações de barragens de concreto: o caso da barragem principal de Itaipu / Júlio César Royer. – Curitiba, 2016
125 f. : il., tabs.

Tese (doutorado) – Universidade Federal do Paraná, Setor de Tecnologia, Programa de Pós-Graduação em Métodos Numéricos em Engenharia.
Orientador: Volmir Eugênio Wilhelm
Coorientadora: Josiele Patias
Bibliografia: 83-91

1. Barragens de concreto. 2. Barragens e açudes - Fundações. 3. Redes neurais – Computação. I. Wilhelm, Volmir Eugênio. II. Patias, Josiele. III. Título

CDD 627.82

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
PRÓ-REITORIA DE PESQUISA E PÓS-GRADUAÇÃO
Setor CIÊNCIAS EXATAS
Programa de Pós Graduação em MÉTODOS NUMÉRICOS EM ENGENHARIA
Código CAPES: 40001016030P0

TERMO DE APROVAÇÃO

Os membros da Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em MÉTODOS NUMÉRICOS EM ENGENHARIA da Universidade Federal do Paraná foram convocados para realizar a arguição da Tese de Doutorado de **JULIO CESAR ROYER**, intitulada: "**Metodologia para a definição de alertas dinâmicos para subpressão em fundações de barragens de concreto: o caso da barragem principal de ITAIPU**", após terem inquirido o aluno e realizado a avaliação do trabalho, são de parecer pela sua APROVAÇÃO.

Curitiba, 29 de Março de 2016.

Prof VOLMIR EUGÊNIO WILHELM (UFPR)
(Presidente da Banca Examinadora)

Prof ANSELMO CHAVES NETO (UFPR)

Prof LUIZ ALBINO TEIXEIRA JUNIOR (UNILA)

Prof PAULO HENRIQUE SIQUEIRA (UFPR)

Participação Por Vídeo Conferência
Prof RAFAEL MORAIS DE SOUZA (UFMG)

AGRADECIMENTOS

Ao meu orientador, professor Dr. Volmir Eugênio Wilhelm, pela amizade, disponibilidade e proatividade em buscar soluções para os problemas mais diversos durante a elaboração da tese.

À minha coorientadora, Dra. Josiele Patias, pela amizade, esclarecimentos técnicos e disponibilidade para orientações.

Ao professor Dr. Luiz Albino Teixeira Júnior, pelas valiosas contribuições, sem as quais o trabalho não teria alcançado o mesmo nível.

Aos professores Dra. Liliana Madalena Gramani e Dr. Anselmo Chaves Neto, pelo empenho na organização e coordenação da turma especial de doutorado interinstitucional, DINTER, na cidade de Foz do Iguaçu.

Ao Centro de Estudos Avançados em Segurança de Barragens, CEASB, pelo suporte financeiro à realização do DINTER.

Ao Instituto Federal do Paraná, IFPR, pelo afastamento das atividades docentes, permitindo a dedicação integral ao doutorado.

À Fundação Araucária, pelo suporte financeiro parcial ao projeto.

Aos colegas de turma, pela ajuda com o aprendizado dos conceitos da matemática, pelo apoio nos momentos mais difíceis, e principalmente pela amizade e pelo clima de intercooperação, que tornou esse período mais leve.

À minha esposa, Valdirene, pelo amor, pelo apoio e pela compreensão nos momentos de dedicação integral à pesquisa.

RESUMO

Uma questão importante em segurança de barragens de concreto é a monitoração da subpressão em suas fundações e a sua manutenção dentro dos limites de projeto, ao longo do seu tempo de operação. A supbressão é a pressão de baixo para cima gerada pela água que se infiltra pela porosidade ou descontinuidades da fundação rochosa sob a barragem. Se exceder os limites de segurança pode levar ao deslizamento ou tombamento da estrutura. As subpressões estão sujeitas a oscilações por mudanças de temperatura ou nível do reservatório, entre outras causas. A monitoração dos níveis de subpressão é realizada com o auxílio de limites de alertas fixos para cada instrumento. Por exemplo, em Itaipu o limite amarelo é estabelecido como o máximo histórico para o instrumento, e o limite vermelho é definido com base no limite de segurança do projeto. Assim, essa definição de limites de alertas fixos, não ajuda a identificar aumentos inesperados quando deveria estar próximo do mínimo sazonal. No caso de um incremento significativo de subpressão não justificada por variações correspondentes nas variáveis causais associadas, é importante que seja acionado um alerta assim que possível, auxiliando dessa forma a equipe de engenheiros responsáveis pela segurança a estudar o caso e se necessário executar as tarefas de manutenção a tempo de reverter um quadro que poderia colocar em risco a segurança da barragem. Os modelos de regressão múltipla de subpressão em Itaipu não conseguem alcançar um nível de explicação aceitável da variação sazonal da subpressão. As séries temporais de subpressão no contato entre o concreto e a rocha apresentam frequentes variações de regime, tornando sua modelagem mais complicada. Diante disso, esta tese propõe um novo método preditivo híbrido, incluindo modelo ARIMAX, Análise Espectral Singular (SSA), Wavelet, Redes Neurais Artificiais (RNAX) e Bootstrap, cujo objetivo é produzir predições, e intervalos de previsão a eles associados, de leituras futuras de subpressão em barragens de concreto, levando em consideração séries temporais de variáveis causais. Em síntese, a previsão pontual é a média entre um previsor ARIMAX que capta estrutura de dependência linear, e um previsor não linear iterativo, que combina redução de ruídos por SSA, decomposição espectral Wavelet e previsão por Redes Neurais Artificiais com busca automática de parâmetros ótimos. A determinação do intervalo preditivo usa Bootstrap, e os limites superiores do intervalo de previsão são interpretados como limites amarelos dinâmicos. O método proposto é aplicado em três séries temporais de subpressão de piezômetros instalados na barragem principal de Itaipu, localizada em Foz do Iguaçu, Paraná, Brasil. Os resultados alcançados mostram uma melhoria significativa quando comparado aos métodos tradicionais de previsão utilizados na literatura.

Palavras-chave: Subpressão; fundações de barragens de concreto; ARIMAX; SSA; Wavelet; Redes Neurais Artificiais.

ABSTRACT

An important point in concrete dam safety is to monitor the uplift pressure on its foundations, and keep it below the design limits during all operation life time. Uplift pressure is the bottom upward pressure generated by water that seeps through rock foundation porosity or discontinuities under the dam. If the design safety limits are exceeded, it may lead to slippage or tipping of the structure. The uplift pressure oscillates depending on changes in temperature or reservoir level, among other causes. Monitoring of uplift pressure levels is performed with the aid of fixed warning thresholds for each instrument. For example, in Itaipu the yellow threshold is established as the high value recorded for the instrument, and the red limit is set based on the design's safety limit. Thus, this definition of fixed warnings thresholds does not help to identify unexpected increases when it should be close to the minimum. In the case of a significant increase of uplift pressure not justified by corresponding variation in the associated causal variables, it is important to trigger an alert as soon as possible, thus aiding the dam safety engineers team to study the case and to perform any maintenance task in time to reverse a possible danger situation for the dam safety. Multiple regression models in Itaipu were not able to reach an acceptable explanation level of the uplift pressure seasonal variation. The uplift pressure time series in the concrete-rock contact have frequent regimen changes, making its modeling more complicated. Therefore, this thesis proposes a new hybrid predictive method, including ARIMAX model Spectral Analysis Singular (SSA), Wavelet, Artificial Neural Networks (RNAX) and Bootstrap, whose goal is to produce predictions, and prediction intervals associated with them, of future uplift readings in concrete dams, considering causal variables time series. In short, the point forecast is the average between an ARIMAX forecaster that captures linear dependence structure, and an iterative non-linear predictor that combines noise reduction by SSA, Wavelet spectral decomposition and forecast by Artificial Neural Networks with automatic search of optimal parameters. Bootstrap determines the predictive interval, and the upper limits of this interval are interpreted as dynamic yellow limits. The proposed method is applied in three uplift pressure time series, for piezometers installed in the Itaipu main dam, located in Foz do Iguaçu, Paraná, Brazil. The results obtained show a significant improvement when compared to traditional prediction methods in the literature.

Keywords: Uplift pressure; concrete dam foundations; ARIMAX; SSA; Wavelet; Artificial Neural Networks.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

As barragens hidráulicas (de concreto, de enrocamento ou de terra) consistem em estruturas construídas no curso de um rio com o objetivo de gerar um reservatório (ou lago), ao seu lado a montante (rio acima), para captação de recursos hídricos, com diversas finalidades, tais como: irrigação, abastecimento de água, controle de enchentes, navegação, pesca ou produção de energia hidrelétrica. Para esta última, em particular, as barragens podem ser classificadas em dois grandes grupos: as de concreto, que podem ser do tipo gravidade, gravidade aliviada, arco ou contrafortes; e as convencionais de enrocamento ou de terra. Na Figura 1, são apresentados os tipos de barragens mencionados.

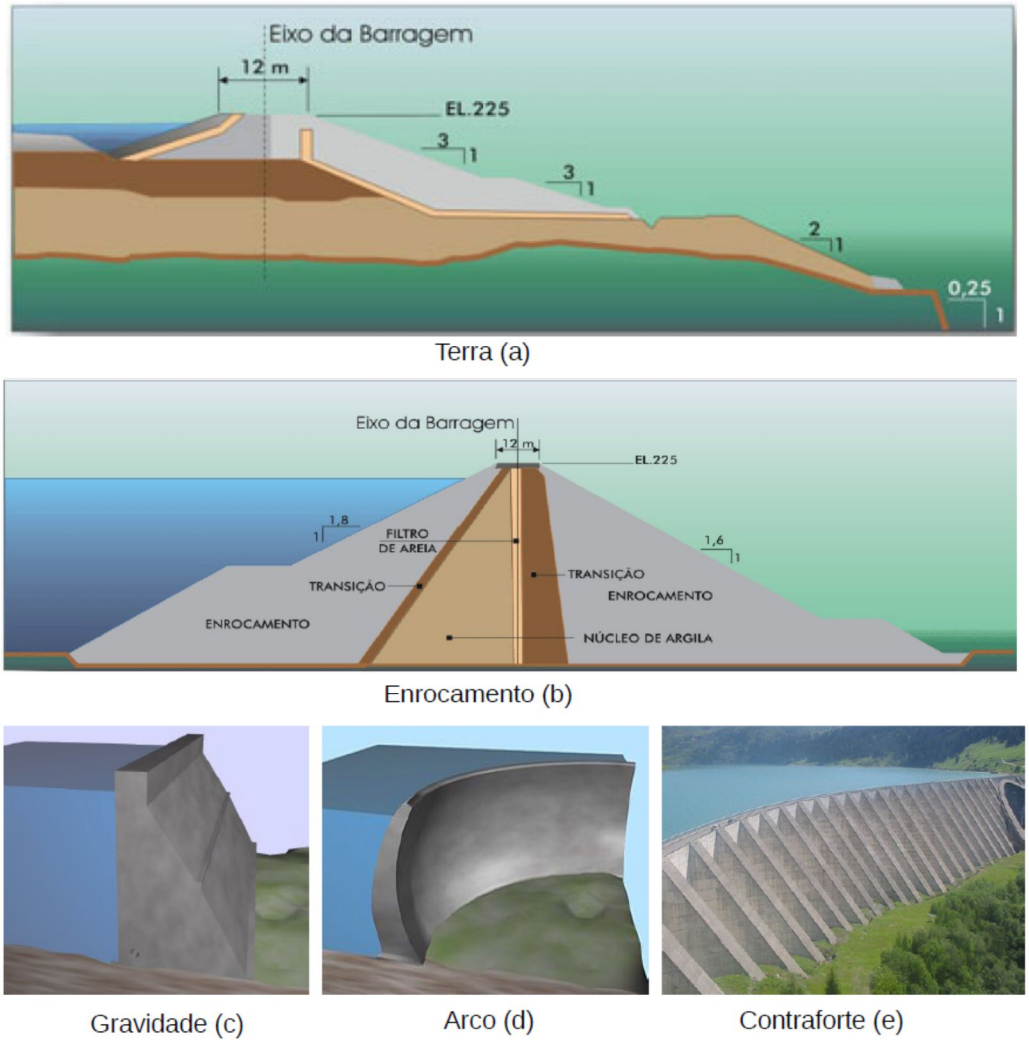


FIGURA 1 - TIPOS DE BARRAGENS DE ACORDO COM O MATERIAL EMPREGADO.

FONTE: Adaptado de Itaipu (2016)(a e b), Wikimedia (2014)(c), Wikipedia (2014)(d) E Touristlink (2015)(e)

Uma usina hidrelétrica, principalmente quando de grande porte, é, de fato, um empreendimento de considerável relevância para o desenvolvimento econômico-social de um país, e a sua manutenção e monitoramento requerem capacidade técnica-científica, exigindo o envolvimento de profissionais de áreas distintas e alta capacidade e perspicácia.

O tema “segurança de barragens”, em particular, consiste em um assunto amplamente debatido em todo o mundo. O Brasil, por exemplo, se mostra atento à “saúde” de suas barragens, de maneira que, em 20 de setembro de 2010, foi promulgada a lei no. 12.334 (BRASIL, 2010), que estabelece a regulamentação da Política Nacional de Segurança de Barragens destinadas à acumulação de água para quaisquer usos, à disposição final ou temporária de rejeitos, e à acumulação de resíduos industriais. Além disso, foi criado o Sistema Nacional de Informações sobre Segurança de Barragens (SNISB); e, em seguida, publicada a resolução no. 144 de 10 de julho de 2012 (CNRH, 2012), estabelecendo diretrizes para a implantação da política nacional de segurança de barragens, aplicação de seus instrumentos e atuação do sistema nacional de informação sobre segurança de barragens. Atualmente, as empresas responsáveis por barragens brasileiras adotam políticas de segurança baseadas na referida lei, de maneira a monitorar a segurança através de um sistema de instrumentação instalado em pontos estratégicos da barragem, o que permite amplo acompanhamento da saúde estrutural da barragem, diante da ocorrência de diversos eventos aleatórios (como temperatura do ambiente, temperatura da água, nível do reservatório, variação pluviométrica, subpressão) que afetam a sua dinâmica temporal.

Um dos fatores de risco em segurança de barragens é a *subpressão*, que é a pressão exercida de baixo para cima pela água presente na fundação, e é aferida por meio de um instrumento chamado “piezômetro”. No caso particular de barragens de concreto, é gerada pela água que percola (ou seja, infiltra) pela estrutura rochosa do leito do rio sobre a qual a barragem se apoia. A subpressão alivia o preso da estrutura, e a presença da água ainda tem o efeito de reduzir o coeficiente de atrito entre as superfícies de descontinuidades presentes na fundação. Se a pressão da água sob a fundação da barragem (subpressão) for suficientemente grande, pode provocar um deslizamento da barragem, com consequências catastróficas (CDBD, 1999).

A fim de minimizar tal problema, durante a construção de barragens normalmente são usadas duas técnicas (ANDRADE, 1988): (i) cortinas de injeção de calda de cimento, para diminuir a permeabilidade nas regiões de rochas fraturadas ou com descontinuidades permeáveis, e (ii) cortinas de drenagem, para aliviar a subpressão. A quantidade de água coletada pelos drenos deve ser monitorada constantemente, juntamente com outros parâmetros tais como temperatura, pressão da água na fundação, deformação das rochas que suportam a barragem, inclinação da barragem, deformação da barragem e deslocamento da barragem. A subpressão também pode ser afetada por alterações nos caminhos de percolação de água, seja pela colmatação que provoca o entupimento dos drenos próximos, seja pela erosão, que pode aumentar o fluxo de água (OSAKO, 2002).

Existem ainda outros fatores com potencial para afetar a subpressão, tais como o nível do reservatório, que é a fonte da pressão hidrostática, e a quantidade de precipitações na região da barragem, cuja influência não é facilmente identificada pelos gráficos das séries temporais. Com efeito, faz-se necessário o desenvolvimento de um modelo preditivo a fim de se explicar e predizer o futuro do comportamento temporal da subpressão, com base na série de tempo histórica, para verificar se há indícios relevantes de preocupação. Uma abordagem deste tipo poderia acusar, rapidamente e com maior precisão, se algum piezômetro está registrando comportamento significativamente anormal, sugerindo uma investigação daquele local específico. Trata-se de uma técnica útil para otimizar o trabalho dos analistas responsáveis pela segurança da barragem. Um exemplo de ocorrência que pode causar comportamento atípico é a obstrução de drenos por depósitos de carbonato de Cálcio, indicado por uma redução na vazão de um dreno acompanhado por um aumento da subpressão que é indicada por um piezômetro próximo ao dreno - sugerindo procedimentos de desobstrução do dreno ou perfuração de outro (OSAKO, 2002).

1.1 Problema

O problema da subpressão em fundações de barragens de concreto exige o seu monitoramento constante, e a execução de eventuais procedimentos de manutenção, antecipadamente, caso seus valores passem a ficar perigosamente elevados. Em grandes barragens, como é o caso de Itaipu, acompanhar de perto

todos os instrumentos é tarefa complicada, dada a quantidade de instrumentos instalados. Isso torna necessário um sistema de previsão de subpressão, capaz de identificar comportamentos fora do esperado, levando-se em conta variáveis causais com influência sobre a subpressão.

1.2 Objetivos

O objetivo geral deste trabalho é desenvolver uma metodologia para identificar comportamentos de subpressão fora da faixa de valores esperados.

Os objetivos específicos são:

1. Estabelecer a metodologia para identificar as variáveis independentes (temperatura ambiente, temperatura da água, e nível do reservatório) com influência significativa sobre a variável dependente (subpressão);
2. Construir um método para realizar uma previsão pontual híbrida da subpressão levando em consideração as séries históricas da subpressão e das variáveis independentes;
3. A partir da previsão pontual da subpressão, calcular o intervalo de previsão, servindo de base para a definição de valores dinâmicos para os alertas do comportamento da subpressão.

1.3 Justificativa

O presente trabalho se justifica pela ausência de trabalhos disponíveis na literatura para a definição de limites de alerta para subpressão em barragens de concreto, que funcionem na realidade de Itaipu. O trabalho de referência para estabelecimento de critérios para alertas de subpressão (KUPERMAN, MORETTI, *et al.*, 2003), utiliza técnicas de regressão múltipla, que não produzem resultados aceitáveis para a previsão de subpressão nos piezômetros da feição contato concreto-rocha, na barragem principal de Itaipu. É essa a feição que apresenta as maiores oscilações, e, portanto, que necessita maior atenção das equipes de segurança de barragens.

1.4 Estrutura do Trabalho

Este trabalho possui 5 capítulos, incluindo a Introdução. O capítulo 2 apresenta uma fundamentação teórica, com uma introdução ao escoamento em maciços rochosos fraturados, a caracterização geológica das fundações das barragens de Itaipu, uma introdução ao modelo ARIMAX e às técnicas SSA, Wavelet, RNA e Bootstrap; o capítulo 3 descreve o material usado e o método desenvolvido; no capítulo 4 são apresentados e discutidos os resultados obtidos; por fim, o capítulo 5 apresenta a conclusão e proposta de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Na seção 2.1, são tratados de forma sucinta os conhecimentos de Engenharia necessários para compreender o contexto real no qual se insere a subpressão (no contato concreto-rocha) em uma barragem hidráulica de concreto – mais especificamente, a da usina de Itaipu. Por sua vez, na seção 2.2, é feita uma breve revisão dos métodos numéricos que integram o método preditivo híbrido proposto nesta tese - descrito no capítulo 3 -, cujo objetivo é a predição pontual e intervalar das subpressões no contato concreto-rocha, na barragem de concreto em Itaipu.

2.1 Conhecimentos de engenharia

A Usina Hidrelétrica de Itaipu, localizada no rio Paraná, possui um barramento composto por vários tipos de barragens: barragem de terra, de enrocamento e de concreto, todas instrumentadas e monitoradas. Em Itaipu essa monitoração é realizada através de mais de 2200 instrumentos instalados ao longo de toda a extensão da barragem, lidos a intervalos regulares, com a periodicidade variando de acordo com o instrumento. Atualmente 270 destes instrumentos, em pontos considerados críticos, são também lidos automaticamente a cada 30 minutos, simultaneamente (ITAIPU, 2009).

O sistema atualmente em uso em Itaipu utiliza valores estáticos como parâmetros para alertas de cada instrumento, como o valor máximo já registrado para aquele instrumento (alerta amarelo), ou o limite de projeto (alerta vermelho). Quando acontece alguma variação, como por exemplo, uma vazão acima do máximo já registrado, os engenheiros precisam confrontar manualmente os dados das diversas grandezas medidas, e fazer uma análise para identificar se isso oferece risco ou não. Mas esses limites fixos não oferecem uma boa acurácia, pois uma leitura próxima ao máximo histórico em uma época do ano em que deveria estar próxima ao mínimo é motivo de preocupação, mas o sistema não detecta, e ao mesmo tempo, uma leitura levemente acima do valor máximo histórico aciona o alerta, mas pode não ser motivo de preocupação, se associada a uma temperatura

média ambiental ou temperatura da água também abaixo do valor mínimo histórico para o período.

Um estudo do histórico de leitura destes instrumentos sugere que há forte relação entre a subpressão e a temperatura ambiental média de três meses antes (FIORINI, 2001). Do mesmo modo, a teoria do escoamento em maciços rochosos fraturados afirma que a temperatura da água que passa pelas descontinuidades e a compressão de cada camada da fundação também influenciam a vazão e a subpressão (ANDRADE, 1984; 1988), (MASCARENHAS, 2009). Por outro lado, é plausível que a compressão de certos trechos da fundação seja uma consequência da temperatura ambiente e da geometria da barragem, especialmente no caso de barragem de concreto de gravidade aliviada, como é o caso da barragem principal de Itaipu.

As barragens de concreto, em particular, normalmente são assentadas sobre fundações rochosas que apresentam descontinuidades em sua formação, decorrentes de origem singenética, ações sísmicas, químicas ou intemperismo; além disso, estes maciços rochosos podem apresentar diferentes graus de porosidade e permeabilidade. Cada uma dessas características varia de um local para outro, de modo que, no processo de valoração de riscos inerentes à estrutura de concreto, se faça uma avaliação adequada quanto às condições de permeabilidade, e resistência à compressão e ao cisalhamento (CDBD, 1999).

A água que se infiltra pela porosidade natural da rocha da fundação e por suas descontinuidades exerce uma pressão (a qual é referida como “subpressão”) sob a estrutura de concreto, aliviando o peso da estrutura, podendo alterar o coeficiente de atrito, de maneira a reduzir a resistência ao cisalhamento (OSAKO, 2002). Tal fenômeno aumenta significativamente o risco à estrutura da barragem, se este ultrapassar valores definidos em projeto. Com o passar do tempo, as condições hidráulicas da fundação podem se alterar, por conta das ações mecânicas de erosão ou depósito de materiais nos caminhos seguidos pela água. Com efeito, as condições hidráulicas necessitam ser monitoradas durante toda a vida útil do empreendimento (CDBD, 1999).

Em maciços formados por rochas de baixa permeabilidade, as investigações geológicas e geotécnicas usualmente apontam uma permeabilidade hidráulica para as descontinuidades várias ordens de grandeza maiores do que a permeabilidade do maciço rochoso contínuo (ANDRADE, 1988), (MASCARENHAS, 2009). Desta forma,

no cálculo da estimativa de vazão e subpressão de um empreendimento, os valores de percolação pelo maciço rochoso passam a ser desprezíveis, conforme destaca Andrade (1984).

O modelo mais utilizado faz a substituição do modelo de fratura irregular real (Figura 2A) por um modelo teórico de uma descontinuidade plana (Figura 2B), com uma abertura nominal teórica ε tal que o escoamento do modelo teórico seja equivalente ao escoamento observado nos ensaios realizados na fratura real.

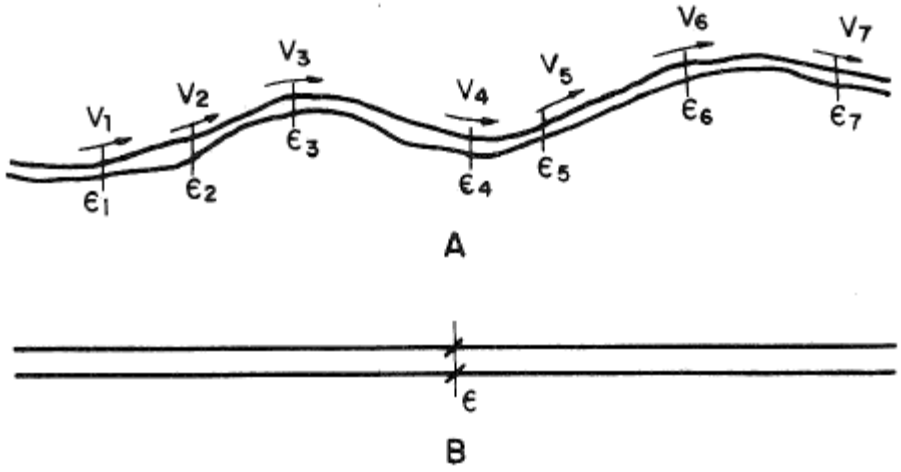


FIGURA 2 - DESCONTINUIDADE NATURAL (A); E SUA REPRESENTAÇÃO PLANA REGULAR (B)
FONTE: Andrade (1988).

Para a caracterização hidráulica do maciço rochoso, é necessário determinar as famílias de descontinuidades e suas orientações, as descontinuidades em cada família, como também a permeabilidade e as descontinuidades no maciço rochoso. As permeabilidades das descontinuidades podem ser obtidas através de ensaios *in situ* ou em laboratório, e estas dependem das aberturas das descontinuidades e do seu material de preenchimento (MASCARENHAS, 2009).

Além da condutividade hidráulica do maciço rochoso da fundação, a subpressão em um ponto específico da fundação depende dos seguintes aspectos: (i) fatores de projeto da obra, como distância entre o ponto e o dreno mais próximo, geometria e diâmetro dos drenos, profundidade do ponto até a boca do dreno, distância entre o ponto e o reservatório e a pressão exercida pelo peso da estrutura e de (ii) fatores que mudam no decorrer do tempo, como a pressão hidrostática (nível do lago), entupimento dos drenos, variação da pressão exercida pelo peso da

estrutura em função de variações térmicas ambientais, erosão ou fechamento por depósito de materiais nas descontinuidades condutoras de água na fundação, temperatura e viscosidade da água e variações das aberturas das descontinuidades em função de dilatações e contrações térmicas ocasionadas pela variação da temperatura da água que passa por essas descontinuidades (ANDRADE, 1984; 1988), (MASCARENHAS, 2009).

De acordo com Andrade (1988), uma diminuição de 1ºC na temperatura da água que passa por uma descontinuidade na fundação rochosa de uma barragem (por exemplo, de 20ºC para 19ºC) pode provocar um aumento na ordem de 110% no fluxo de água por essa descontinuidade, acarretando forte influência da subpressão na região na qual se fez a aferição.

2.1.1 A Barragem de Itaipu

A Usina Hidrelétrica de Itaipu se situa no rio Paraná, na divisa do Brasil com o Paraguai, a 14 Km do centro da cidade de Foz do Iguaçu, e 20 Km do centro de Ciudad Del Este. Sua estrutura de barramento possui 7.806m, com coroamento na cota de 225,0m em relação ao nível do mar, e é composta por barragens de terra, enrocamento e concreto (gravidade, gravidade aliviada e contrafortes), com a identificação e extensão abaixo, e que pode ser observada na Figura 3 (ITAIPU, 2009):

1.	Barragem de terra direita	872m
2.	Vertedouro (Trecho A)	374m
3.	Barragem lateral direita (blocos de contraforte – Trecho D)	968m
4.	Barragem de ligação direita (blocos de contraforte – Trecho E)	182m
5.	Barragem principal (blocos de gravidade aliviada – Trecho F)	612m
6.	Estrutura de desvio (blocos de gravidade maciços – Trecho H)	170m
7.	Barragem de ligação esquerda (blocos de contraforte – Trecho I)	350m
8.	Barragem de enrocamento	1.984m
9.	Barragem de terra esquerda	2.294m

FIGURA 3 - TIPOS DE BARRAGENS DA USINA HIDRELÉTRICA DE ITAIPU.

FONTE: Adaptado de Google Maps (2014)

O foco deste trabalho é no trecho F (barragem principal), bloco F19/20, um dos blocos de maior altura, com 185m da fundação até o topo, e bloco F5/6, com 135m da fundação até o topo, ambos localizados no leito do rio Paraná e extensamente instrumentados. O reservatório de Itaipu, em seu nível normal alcança a cota de 220m acima do nível do mar. Já o nível do rio Paraná, a jusante da usina se situa próximo da cota de 102m acima do nível do mar.

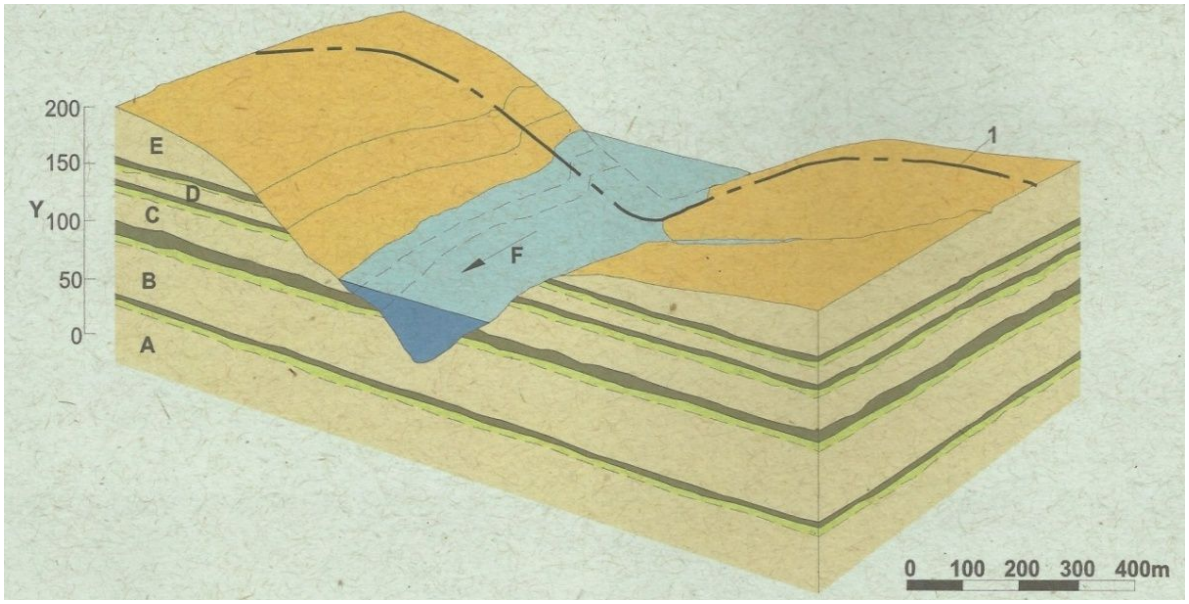
A área da Usina Hidrelétrica de Itaipu, incluindo o reservatório, está na bacia superior do Paraná, assentada sobre grandes derrames basálticos do período

Jurássico inferior, pertencendo à Formação Serra Geral, com as seguintes características (Itaipu, 2008):

1. Derrames basálticos essencialmente horizontais, de 20 a 60m de espessura;
2. Camadas de brecha entre os derrames basálticos, de 1 a 30m de espessura, na maior parte heterogêneas, normalmente mais fracas, mais deformáveis e mais permeáveis do que o basalto, formadas pela consolidação de depósitos de solo, areia e detritos pela ocasião do novo derrame.
3. Descontinuidades em planos paralelos aos derrames basálticos, normalmente localizadas no contato entre derrames ou na base da zona de transição entre basalto vesicular amigdaloide e basalto compacto.
4. Permeabilidade horizontal várias vezes maior do que a permeabilidade vertical.

Os derrames basálticos são relativamente uniformes e de baixa permeabilidade, sendo em geral compostos por basaltos sãos. Apresentam-se, em geral, densos da parte central para baixo, e com uma camada vesículo-amigdaloidal na parte superior, que durante a formação esteve sujeita a um resfriamento mais rápido e com aprisionamento de gases, formando pequenas bolhas, algumas preenchidas com minerais, outras vazias. Em geral há também uma descontinuidade entre as camadas de basalto denso e de basalto vesículo-amigdaloidal de cada derrame, que apresenta alta permeabilidade, se comparada às camadas inferior e superior.

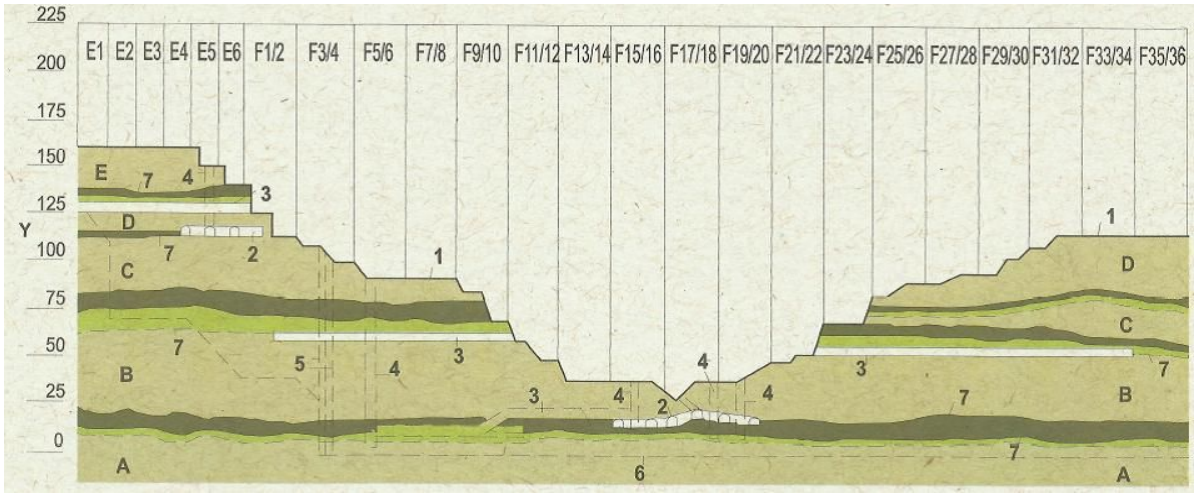
A Figura 4 ilustra a composição geológica do local do projeto de Itaipu, mostrando os 5 principais derrames investigados.



Y: Elevação (m)	Basalto denso
1: Eixo da Barragem	Brecha
A,B,C,D,E: Derrames basálticos	Basalto vesículo-amigdaloidal
F: Direção do fluxo	

FIGURA 4 - GEOLOGIA NA ÁREA DO PROJETO DE ITAIPU

FONTE: Itaipu (2008).



Y: Elevação (m)	E1...E6: Barragem lateral direita
1: Perfil de escavação	F1/2...F35/36: Barragem principal
2: Chavetas	A,B,C,D,E: Derrames basálticos
3: Túneis de exploração	
4: Poços de prospecção	Basalto denso
5: Cortina de injeção transversal	Brecha
6: Limite da cortina de injeção	Basalto vesículo-amigdaloidal
7: Descontinuidades	

FIGURA 5 - FUNDAÇÃO DA BARRAGEM PRINCIPAL – SEÇÃO GEOLÓGICA LONGITUDINAL.

FONTE: Itaipu (2008).

A Figura 5 ilustra a fundação dos trechos E e F da barragem principal, e a respectiva localização e altura dos blocos F-5/6 e F-19/20, sob os quais estão localizados os piezômetros estudados nesta tese.

2.2 Conhecimentos de métodos numéricos

Do ponto de vista estatístico, as aferições no tempo das subpressões ao longo das estruturas são séries temporais estocásticas. Com efeito, elas são passíveis de serem explicadas e preditas no tempo por meio de métodos preditivos. Uma série temporal pode ser vista como uma realização de um conjunto de variáveis aleatórias, chamado “processo estocástico”, e, em linhas gerais, são temporalmente dependentes (como, por exemplo, as séries de tempo de subpressão).

A fim de modelar uma série histórica, Box e Jenkins (1970) propuseram uma abordagem que consiste em ajustar modelos estatísticos *Autorregressivos Integrados de Médias Móveis* (ARIMA), simples e sazonais, a um conjunto de observações históricas. Essa abordagem considera três estágios básicos: identificação do modelo, estimação dos parâmetros e verificação do modelo ajustado. Os modelos de Box e Jenkins são amplamente utilizados, porém são capazes de capturar somente estruturas de autodependência linear, conforme destaca Zhang (2003). Para se capturar informações de autodependência não linear, o autor ainda sugere a utilização de Redes Neurais Artificiais (RNA). Firmino, Mattos Neto e Ferreira (2014) propõe uma combinação das previsões oriundas de modelos ARIMA e de RNAs para se produzir previsões híbridas, dotadas, simultaneamente, de informações de estruturas lineares e não lineares presentes nos dados.

Zhang (2003), Zou e Yang (2004), Faria e Mubwandarikwa (2008), Martins (2011), Cavalieri e Ribeiro (2011) e Firmino, Mattos Neto e Ferreira (2014) são exemplos de autores que desenvolveram pesquisas que mostram empiricamente a eficácia preditiva das previsões híbridas oriundas de combinação de modelos. Em sua pesquisa clássica, Clemen (1989) afirmou que as combinações de previsões são métodos práticos e eficazes, em regra. Firmino, Mattos Neto e Ferreira (2014) obtiveram resultados que reforçam a tese de que, com a utilização combinação de métodos de previsão, ganhos de acurácia consideráveis em relação a outros previsores individuais (como, por exemplo, modelos ARIMA e de RNAs) são obtidos.

Outra técnica que vem sendo usada com sucesso nas previsões de séries temporais é a abordagem *Wavelet*. Embora os primeiros estudos no campo das *Wavelet*s tenham ocorridos em 1909 (HAAR, 1910), somente em 1985, Stephane Mallat deu às *Wavelet*s um grande impulso através do seu trabalho em processamento digital de imagens, construindo a primeira *Wavelet* não-trivial (suave) (LIMA, 2002). Daubechies (1988) construiu um conjunto de bases ortonormais de *Wavelet* suaves, passando a ser uma das principais referências para as atuais aplicações da técnica. Lima (2011) utilizou *Wavelet* no seu trabalho de previsão de séries econômicas, separando a parte determinística dos ruídos. Teixeira Jr *et al.* (2012) optaram por um método de combinação em que as componentes *Wavelet* de uma série temporal constituem os padrões de entrada de uma RNA *feedforward* MLP (*multi-layer perceptron*), cuja saída fornece a previsão da série temporal. Karthikeyan e Kumar (2013); Tiwari e Chatterjee (2010); Kisi (2010); Kisi e Cimen (2011); Nalley *et al*. (2012) também utilizaram *Wavelet* em seus trabalhos de previsão de séries temporais e obtiveram ganhos preditivos relevantes.

Assumindo que uma série temporal, denotada por $y_t\ (t = 1, \dots, T)$, tenha uma forte componente estocástica (como o são as de subpressão), é fortemente recomendada a sua filtragem, antes de sua efetiva modelagem. Nota-se que y_t é tal que, para cada instante t, pode ser teoricamente decomposta como: $y_t = \tilde{f}(t) + e_t$, onde: (i) e_t é um estado (ou realização) de uma variável aleatória independente e identicamente distribuída (iid), com média zero e variância constante; e (ii) $\tilde{f}(t)$ é a componente determinística (que, de fato, é passível de ser prevista pontualmente ao contrário de e_t) de y_t. Entre as técnicas de filtragem de séries temporais existentes, destaca-se o encolhimento (shrinkage) Wavelet, sendo David L. Donoho e Iain M. Johnstone os precursores nesta área com os trabalhos: Donoho e Johnstone (1994, 1998, 1999), entre outros. A filtragem Wavelet pode ser abordada com uma regressão não paramétrica, onde a função $f(x_i)$ no modelo $y_i = f(x_i) + \varepsilon_i$ é obtida dos dados da resposta y_i, sem que seja assumida uma estrutura paramétrica para $f(x_i)$ (ABRAMOVICH et al. 2000). Uma das soluções para esta abordagem é a expansão Wavelet do sinal e eliminação, manutenção ou atenuação dos coeficientes das componentes de detalhes mediante regras de limiarização. Outra técnica de filtragem de sinal bastante utilizada é a análise espectral singular (SSA – singular spectrum analysis), que se trata de uma abordagem relativamente nova que

incorpora elementos de análise clássica de séries temporais, estatística multivariada, geometria multivariada, sistemas dinâmicos e processamentos de sinais (ELSNER e TSONIS, 1996). A partir do método SSA, uma série temporal pode ser decomposta em componentes que podem ser agrupadas em três grupos distintos: tendência, harmônica e ruído, sendo possível a transformação da série temporal observada em uma série com componente estocástica de menor magnitude.

Diante o exposto, percebe-se uma grande diversidade de métodos e aplicações das séries temporais, sendo a melhoria da acurácia das previsões o cerne dos estudos encontrados nesta área. O presente trabalho está inserido neste contexto e se propõe a apresentar um método de previsão para séries temporais envolvendo *SSA* na filtragem das séries temporais e *Wavelet* na decomposição ortogonal do resultado da filtragem, modelagem neural e *Bootstrap* na estimativa da incerteza que envolve o processo de geração das previsões de subpressões e explicar as suas relações com condições de contorno disponíveis.

2.2.1 Análise Espectral Singular

A análise espectral singular (SSA – *Singular Spectrum Analysis*) consiste em uma abordagem de tratamento de sinais que integra elementos da análise clássica de séries temporais, estatística multivariada, sistemas dinâmicos, análise funcional e processamento de sinais. O método SSA tem por base a decomposição de uma série temporal em função de três componentes fisicamente interpretáveis - referidas por tendência, oscilatórias (ou harmônicas) e ruído. Quanto à série temporal, não é necessário que esta seja estacionária ou que tenha parâmetros probabilísticos conhecidos, o que torna a referida técnica aplicável a diversas áreas. A abordagem SSA pode ser aplicada para algumas finalidades, como: mapear tendências, suavisar séries temporais, encontrar componentes sazonais e ciclos, redução de ruídos (GOLYANDINA e ZHIGLJAVSKY, 2013), (HASSANI, 2007). Nesta tese a SSA, porém, foi utilizada para a redução de ruídos em uma série de tempo de subpressão a ser modelada.

Operacionalmente, o método SSA pode ser dividido em duas etapas básicas: decomposição e reconstrução.

2.2.1.1 Decomposição

Seja $y_t\ (t = 1, \dots, N)$ uma série temporal de tamanho igual a N. A etapa da decomposição de $y_t\ (t = 1, \dots, N)$ pode ser subdividida em outras duas subetapas: a de incoporporação e a de decomposição em valores singulares (SVD – Singular Value Decomposition).

Na subetapa de incorporação, a matriz trajetória X é construida, usando uma janela de tamanho L que percorre toda a série temporal. A cada passo uma coluna da matriz trajetória, com L linhas, é composta. Assim, o total de colunas K é tal que $K = N - L + 1$.

$$X = [X_1, \dots, X_K] = (x_{ij})_{i,j=1}^{L,K} = \begin{pmatrix} y_1 & y_2 & \cdots & y_K \\ y_2 & y_3 & \cdots & y_{K+1} \\ \vdots & \vdots & \ddots & \vdots \\ y_L & y_{L+1} & \cdots & y_N \end{pmatrix}$$

O tamanho da janela L pertence ao intervalo $2 \leq L \leq N - 1$. A escolha de L é tópico de pesquisa, sendo que Golyandina, Nekrutkin e Zhigljavsky (2001) sugere que L seja próximo a $\frac{N}{2}$. Já Khan e Poskitt (2013) destaca que o tamanho da janela pode ser significativamente menor, com resultados melhores, e sugere, para tanto, uma faixa de valores, que é dada por: $L = (\ln N)^c$, com $1{,}5 \leq c \leq 3$, e $L \leq \frac{N}{2}$.

Na subetapa SVD, a matriz trajetória X é decomposta em uma soma de matrizes E_i, com $i = 1, \dots, d$, onde d é o número de autovalores λ não nulos:

$$X = \sum_{i=1}^{d} E_i = \sum_{i=1}^{d} \sqrt{\lambda_i} U_i V_i^T \qquad (2.2.1.1)$$

Onde λ_i e U_i são, respectivamente, os i-ésimos autovalores e autovetores da matriz $S = XX^T$, enquanto que o vetor V_i é tal que $V_i = \frac{X^T U_i}{\sqrt{\lambda_i}}$, para $i = 1, \dots, d$. O trio λ_i, U_i e V_i é comumente referido por autotripla, e os elementos da sequência λ_i $(i = 1, \dots, d)$ de autovalores são ordenados em ordem decrescente (PESSANHA, GUEDES, *et al.*, 2014), (MENEZES, SOUZA e PESSANHA, 2015).

2.2.1.2 Reconstrução

A etapa da reconstrução, por sua vez, também pode ser dividida em duas subetapas, a saber: a de agrupamento e a de média diagonal. Na subetapa do agrupamento, as componentes $\mathrm{E_i}\ (\mathrm{i} = 1, \dots, \mathrm{d})$ são agrupadas de acordo com suas características em m grupos, sendo que $\mathrm{m} < d$. Como consequência, para um grupo formado por p componentes, cujos índices são $\mathrm{I} = \{\mathrm{I_1}, \dots, \mathrm{I_p}\}$, a matriz resultante $\mathrm{W_I}$ é dada por:

$$\mathrm{W_I} = \mathrm{E_{I_1}} + \cdots + \mathrm{E_{I_p}}.$$

De igual modo, são computadas m matrizes resultantes para os grupos dados, genericamente, por $\mathrm{I_1}, \dots, \mathrm{I_m}$. Assim, a expressão (2.2.1.1) pode ser reescrita tal como:

$$\mathrm{X} = \mathrm{W_{I_1}} + \cdots + \mathrm{W_{I_m}}.$$

Na subetapa da média diagonal, porém, cada matriz $\mathrm{W_{I_i}}$, em que $\mathrm{i} = 1, \dots, \mathrm{m}$, referente à expressão acima, é convertida em uma subsérie temporal de tamanho N, chamada “componente SSA”. O procedimento é realizado com o cálculo da média dos elementos $\mathrm{w_{jk}}$, onde a soma $\mathrm{j} + \mathrm{k}$ produz o mesmo valor. Dessa maneira, seja $\mathrm{y_{t^{(i)}}}\ (\mathrm{t} = 1, 2, \dots, \mathrm{N})$ uma componente SSA de $\mathrm{y_t}\ (\mathrm{t} = 1, 2, \dots, \mathrm{N})$ gerada a partir da matriz elementar $\mathrm{W_{I_i}}$, e $\mathrm{w_{jk^{(i)}}}$ o elemento da linha j e coluna k de $\mathrm{W_{I_i}}$. Considere $\mathrm{L^*} = \min(\mathrm{L}, \mathrm{K})$ e $\mathrm{K^*} = \max(\mathrm{L}, \mathrm{K})$. O cálculo de $\mathrm{y_{t^{(i)}}}\ (\mathrm{t} = 1, 2, \dots, \mathrm{N})$, através do procedimento de média diagonal, é definido pela expressão (2.2.1.2) (CASSIANO, MENEZES e PESSANHA, 2014), a seguir:

$$\mathrm{y_t^{(i)}} = \begin{cases} \dfrac{\sum_{\mathrm{j}=1}^{\mathrm{t}} \mathrm{w_{j,t-j+1}^{(i)}}}{\mathrm{t}}, \text{para } 1 \leq \mathrm{t} \leq \mathrm{L^*} \\ \dfrac{\sum_{\mathrm{j}=1}^{\mathrm{L^*}} \mathrm{w_{j,t-j+1}^{(i)}}}{\mathrm{L^*}}, \text{para } \mathrm{L^*} \leq \mathrm{t} \leq \mathrm{K^*} \\ \dfrac{\sum_{\mathrm{j}=\mathrm{t}-\mathrm{K^*}}^{\mathrm{N}-\mathrm{K^*}+1} \mathrm{w_{j,t-j+1}^{(i)}}}{\mathrm{N} - \mathrm{K^*} + 1}, \text{para } \mathrm{K^*} \leq \mathrm{t} \leq \mathrm{N} \end{cases} \qquad (2.2.1.2)$$

A contribuição da componente W_{I_i} na composição da série temporal final pode ser avaliada pela razão $\frac{\sum_{j=1}^{p_i}(\lambda_{I_{ij}})^{1/2}}{\sum_{l=1}^{d}(\lambda_l)^{1/2}}$ de autovalores. Uma componente SSA $y_{t^{(i)}}$ $(t = 1, 2, \ldots, N)$ gerada também deve ser testada para verificar se, de fato, se trata de ruído branco; e, neste caso, não é considerada na reconstrução da série temporal supracitada, de forma a gerar uma versão menos ruidosa desta.

2.2.2 Análise Wavelet

A análise Wavelet é uma técnica que permite a decomposição espectral de um sinal. Enquanto na análise de Fourrier o sinal é decomposto em uma série de ondas senoidais de frequências diferentes, na análise Wavelet o sinal é decomposto usando-se informações de escala (compressão e dilatação) e translação de uma função Wavelet (MORETTIN, 1997).

Para entender a base teórica da análise Wavelet, inicialmente, é necessário definir três conceitos fundamentais, a saber: espaço de Hilbert, base ortonormal e série de Fourier. Por espaço de Hilbert, entende-se como sendo um espaço produto interno completo. Em particular, a coleção l^2 de todas as sequências infinitas de números complexos quadraticamente somáveis ou de energia finita (isto é, $l^2 := \{y_t : \mathbb{Z} \to \mathbb{C} : \sum_{t\in\mathbb{Z}} |y_t|^2 < \infty\}$ - em que $\mathbb{Z}$ e $\mathbb{C}$ denotam, respectivamente, o conjunto dos números inteiros e o dos complexos) - equipada de um produto interno $\langle\,;\rangle$ - ou seja, $(l^2, \langle;\rangle)$ -, consiste em um caso exemplo de espaço de Hilbert (KUBRUSLY, 2001).

De acordo com Kubrusly e Levan (2006), um subconjunto $\{h_n\}_{n\in\mathbb{Z}}$ de l^2 é uma base ortonormal, se os axiomas seguintes são assegurados: ortogonalidade (ou seja, $\langle h_n, h_m\rangle = 0$, sempre que $n \neq m$); (ii) normalização (ou seja, $\|h_n\| = 1$, para todo $n \in \mathbb{Z}$); e (iii) completamento (ou seja, $\langle x, h_m\rangle = 0$), se $x = 0$. Com efeito, baseado no Teorema da série de Fourier, enunciado em Kubrusly (2001), se $\{h_n\}_{n\in\mathbb{Z}}$ é uma base ortonormal para o espaço de Hilbert $(l^2, \langle;\rangle)$, então, para todo vetor y_t em l^2, existe uma expansão (única) tal como em (2.2.2.1).

$$y_t = \sum_{n\in\mathbb{Z}} \langle x, y_t\rangle\, h_n \qquad (2.2.2.1)$$

A expansão em (2.2.2.1) é referida por série de Fourier do vetor y_t sobre o espaço de Hilbert $(l^2, \langle ; \rangle)$. Daqui em diante, $(l^2, \langle ; \rangle)$ é sempre representado, por simplicidade, por l^2.

Assim, tome o espaço de Hilbert l^2. Um elemento $\omega(.) \in l^2$ - com um produto interno $\langle ; \rangle : l^2 \longrightarrow \mathbb{C}$ - é uma função Wavelet, se a coleção de funções $\omega_{m,n}(.) := 2^{\frac{m}{2}}\, \omega(2^m(.) - n)$, onde $n, m \in \mathbb{Z}$, forma uma base ortonormal. De acordo com Levan e Kubrusly (2003), um vetor y_t em l^2 admite a sua decomposição através de uma expansão tal como em (2.2.2.1), definida em termos de uma base ortonormal Wavelet $\{\omega_{m,n}(.)\}_{(m,n)\in\mathbb{Z}\times\mathbb{Z}}$, conforme em (2.2.2.2).

$$y_t = \sum_{m\in\mathbb{Z}} \sum_{n\in\mathbb{Z}} \langle y_t, \omega_{m,n}(.) \rangle\, \omega_{m,n}(.) \tag{2.2.2.2}$$

O índice m, em (2.2.2.2), é chamado de parâmetro de escala (ou de resolução) e o n, de parâmetro de translação (MALLAT, 2009). De acordo com Levan e Kubrusly (2003), a projeção de y_t sobre $\omega_{m,n}(.)$ pode ser interpretada como uma variação de detalhes de $f(.)$, na escala m e translação n. Baseado em Mallat (2009), tem-se que o subespaço fechado $W_m(\omega) := \left(\text{span}\{\omega_{m,n}(.)\}_{n\in\mathbb{Z}}\right)^-$ de l^2 é chamado de subespaço de detalhes na escala m. Por sua vez, a projeção ortogonal de $f(.)$ sobre $W_m(\omega)$, denotada por $y_{t,W_m(\omega)}$, é definida pela soma parcial descrita em (2.2.2.3).

$$y_{t,W_m(\omega)} := \sum_{n\in\mathbb{Z}} \langle y_t, \omega_{m,n}(.) \rangle\, \omega_{m,n}(.) \tag{2.2.2.3}$$

De acordo com Teixeira Jr et al. (2012), a projeção ortogonal $y_{t,W_m(\omega)}$, em (2.2.2.3), pode ser interpretada como uma componente de detalhes de y_t, na escala m sobre $W_m(\omega)$. Com efeito, dada a identidade, em (2.2.2.2), segue que y_t pode ser interpretada como a soma de todas suas componentes de detalhes $y_{t,W_m(\omega)}$, em todas as escalas inteiras m, sobre o subespaço fechado $((\sum_{n\in\mathbb{Z}} W_m(\omega))^-, \langle ; \rangle)$ de l^2. De acordo com Kubrusly e Levan (2006), tem-se que $(\sum_{n\in\mathbb{Z}} W_m(\omega))^- = l^2$.

Por sua vez, um elemento $\phi(.) \in l^2$ - com um produto interno $\langle ; \rangle : l^2 \longrightarrow \mathbb{C}$ - é chamado de função (Wavelet) escala, se a família de funções $\phi_{m,n}(.) := 2^{\frac{m}{2}}\, \phi(2^m(.) - n)$, onde $n, m \in \mathbb{Z}$, é tal que: $\langle \phi_{m',n'}(.), \phi_{j,k}(.) \rangle = 0$, sempre que $m' = j$ e $n' \neq k$; e $\langle \phi_{m',n'}(.), \phi_{j,k}(.) \rangle \neq 0$, caso contrário. O subespaço (linear) fechado $V_m(\phi) := \left(\text{span}\{\phi_{m,n}(.)\}_{n\in\mathbb{Z}}\right)^-$ de l^2 é chamado de subespaço de aproximação na escala m; e a projeção ortogonal de y_t sobre $V_m(\phi)$, denotada por $y_{t,V_m(\phi)}$, é definida pela soma parcial em (2.2.2.4) (MALLAT, 2009).

$$y_{t,V_m(\phi)} := \sum_{n\in\mathbb{Z}} \langle y_t, \phi_{m,n}(.) \rangle\, \phi_{m,n}(.) \tag{2.2.2.4}$$

De acordo com Teixeira Jr, Menezes, et al., (2013), $y_{t,V_m(\phi)}$ pode ser interpretada como uma componente de aproximação de y_t, na escala m, sobre $V_m(\phi)$. Por transformada Wavelet sobre l^2, entende-se como sendo um produto interno usual $\langle ; \rangle : l^2 \longrightarrow \mathbb{C}$ entre uma função $y_t \in l^2$ e uma função Wavelet $\omega_{m,n}(.) \in W_m(\omega)$ ou uma função escala $\phi_{m,n}(.) \in V_m(\phi)$, onde $(m, n) \in \mathbb{Z} \times \mathbb{Z}$. De acordo com Mallat (2009), as transformadas Wavelet podem ser classificadas em dois conjuntos disjuntos: o de coeficientes de detalhes, denotado por $\{d_{m,n}\}_{(m,n)\in\mathbb{Z}\times\mathbb{Z}}$, e o de coeficientes de aproximação, denotado por $\{a_{m,n}\}_{(m,n)\in\mathbb{Z}\times\mathbb{Z}}$. Para cada par ordenado $(m, n) \in \mathbb{Z} \times \mathbb{Z}$, as transformadas Wavelet $d_{m,n}$ e $a_{m,n}$ são calculadas, respectivamente, por: $d_{m,n} := \langle y_t, \omega_{m,n}(.) \rangle = \sum_{t\in\mathbb{Z}} y_t\, \omega_{m,n}(t)$; e $a_{m,n} := \langle y_t, \phi_{m,n}(.) \rangle = \sum_{t\in\mathbb{Z}} y_t\, \phi_{m,n}(t)$.

2.2.2.1 Decomposição Wavelet

Em Kubrusly e Levan (2006), é provado que $V_{m_0}(\phi) = \left(\sum_{-\infty}^{m_0-1} W_m(\omega)\right)^-$, sendo que o parâmetro m_0 é tal que $m_0 \in \mathbb{Z}$. Com efeito, visto que $l^2 = \left(\sum_{m\in\mathbb{Z}} W_m(\omega)\right)^-$ e $V_{m_0}(\phi) = \left(\sum_{-\infty}^{m_0-1} W_m(\omega)\right)^-$, segue que, com base no Teorema da Estrutura Ortogonal e no Teorema de Fourier (ambos enunciados em Kubrusly (2001)), o espaço de Hilbert l^2 pode ser ortogonalmente expandido tal como em (2.2.2.5).

$$l^2 = V_{m_0}(\phi) + \left(\sum_{m=m_0}^{+\infty} W_m(\omega) \right)^{-} \quad (2.2.2.5)$$

Ou ainda, de forma equivalente, um vetor y_t em l^2 pode ser decomposto de acordo com a Equação (2.2.2.6).

$$y_t = y_{t,V_m(\phi)} + \sum_{m=m_0}^{+\infty} y_{t,W_m(\omega)} \quad (2.2.2.6)$$

Considerando as componentes Wavelet $y_{t,V_m(\phi)}$ e $y_{t,W_m(\omega)}$, bem como as Equações (2.2.2.1) e (2.2.2.6), tem-se que a Série de Fourier do vetor y_t em l^2, em função de uma base ortonormal Wavelet $\{\phi_{m_0,n}(.)\}_{n\in\mathbb{Z}} \cup \{\omega_{n,m}(.)\}_{(m,n)\in\{m\}_{m_0}^{+\infty}\times\mathbb{Z}}$, é defina tal como:

$$y_t = \sum_{n\in\mathbb{Z}} a_{m_0,n}\, \phi_{m_0,n}(.) + \sum_{m=m_0}^{+\infty} \sum_{n\in\mathbb{Z}} d_{m,n}\, \omega_{m,n}(.) \quad (2.2.2.7)$$

Sendo que: $a_{m,n} = \sum_{t\in\mathbb{Z}} y_t\, \phi_{m,n}(t)$; $d_{m,n} = \sum_{t\in\mathbb{Z}} y_t\, \omega_{m,n}(t)$; e o parâmetro m_0 é um ponto inteiro que toma valor no intervalo $m_0 \leq m < +\infty$.

A título exemplificativo, na Figura 6, têm-se as curvas das funções Wavelet da família de Symlet, com momento nulo igual a 20 (notação: “sym 20”) (MORETTIN, 1997). Na parte superior, tem-se o gráfico da função escala; e, na inferior, o gráfico da a função Wavelet.

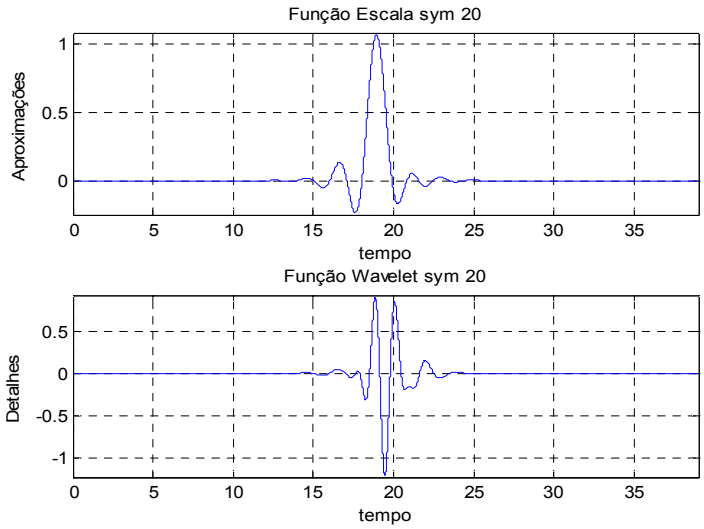


FIGURA 6 - FUNÇÕES WAVELET DA FAMÍLIA SYMELET 20.
FONTE: O autor.

A Figura 7, apresenta as componentes de aproximação e de detalhe da série temporal mensal de vazão de afluentes da região Sul do Brasil (TEIXEIRA JR, 2013) de uma decomposição ortogonal Wavelet de nível 1, construída em termos de uma base ortonormal sym 20 (DAUBECHIES, 1988).

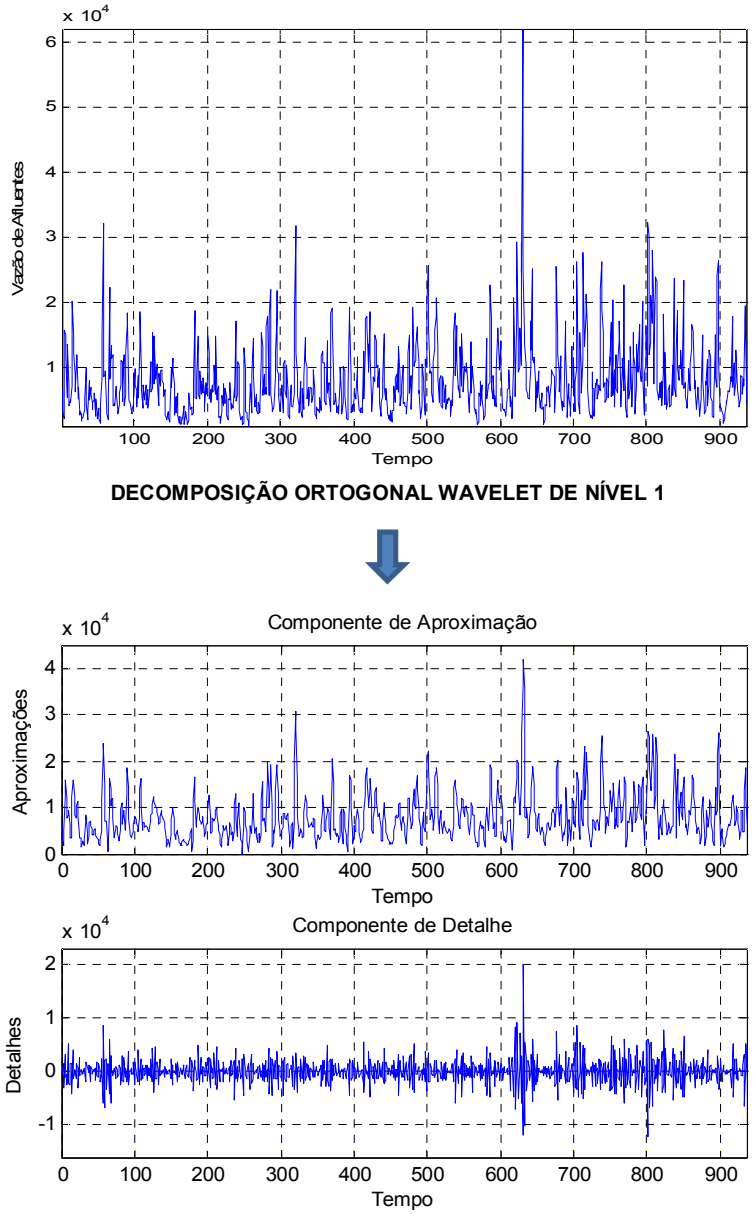


FIGURA 7 - DECOMPOSIÇÃO WAVELET DE NÍVEL 1, COM A BASE ORTOGONAL WAVELET SYM 20, DA SÉRIE TEMPORAL MENSAL DE VAZÃO DE AFLUENTES DA REGIÃO SUL DO BRASIL.

FONTE: Teixeira Jr (2013).

2.2.3 Modelos ARIMAX

Os modelos estatísticos lineares clássicos chamados de Autorregressivos Integrados de Médias Móveis (ARIMA) são aplicáveis à modelagem de séries temporais fracamente estacionárias ou, equivalentemente, estacionárias de segunda ordem (ou seja, cuja média e variância são constantes no tempo). De acordo com

Hamilton (1994), se $y_t\,(t = 1,\ldots,T)$ for uma série temporal não estacionária na média, então pode-se torná-la estacionária (quando isso for possível) a partir de sua diferenciação, com a aplicação do operador de diferenças, ∇^d, tal como, $\nabla^d y_t = W_t = (1-B)^d y_t$; onde: $\nabla^d = (1-B)^d$, sendo d um parâmetro que toma valor inteiro positivo; e $W_t(t = d+1, \ldots T)$ sendo a versão da série temporal subjacente diferenciada d vezes.

Neste caso, a nova série de tempo $W_t\,(t = 1,\ldots,T)$ é definida como um processo $ARMA(p,q)$ e é, usualmente, representada pela formulação matemática $\phi(B)W_t = \theta(B)\varepsilon_t$, sendo que: ε_t $(t = d+1,\ldots,T)$ é uma realização de um processo estocástico Independente e Identicamente Distribuído (i.i.d.); e os parâmetros p e q representam, respectivamente, a ordem dos polinômios autorregressivo $\phi(B)$ - (AR(p)) - e de médias móveis $\theta(B)$ - (MA(q)) -, donde: $\phi(B) = (1 - \phi_1 B - \cdots - \phi_p B^p)$ e $\theta(B) = (1 - \theta_1 B - \cdots - \theta_q B^q)$, sendo que $\phi_i(i = 1, \ldots, p)$ e $\theta_i(i = 1, \ldots, q)$ consistem em listas de números complexos que satisfazem às condições de estacionariedade e invertibilidade (BOX, JENKINS e REINSEL, 2008).

Em outras palavras, pode-se dizer que o modelo ARIMA (p, d, q) é tal que combina linearmente valores passados das entradas $W_t(t = d+1, \ldots, T)$ e choques aleatórios a_t descorrelacionados, de média zero e covariância constante, conforme a Equação (2.2.3.1).

$$W_t = \phi_1 W_{t-1} + \phi_2 W_{t-2} + \cdots + \phi_p W_{t-p} + e_t + \theta_1 e_{t-1} + \theta_2 e_{t-2} + \cdots + \theta_q e_{t-q} \qquad (2.2.3.1)$$

Ou, de forma mais simples,

$$\phi(B)\nabla^d y_t = \theta(B)e_t. \qquad (2.2.3.2)$$

Um processo estacionário $W_t(t = 1,\ldots,T)$, o qual pode ser obtido por meio do operador diferença $\nabla^d y_t$ $(t = 1,\ldots,T)$, em que a série original $y_t(t = 1,\ldots,T)$ é não-estacionária (mas passível de ser tornada estacionária), é chamado de não-estacionário homogêneo (BOX, JENKINS e REINSEL, 2008).

O modelo ARIMA (p, d, q) da equação (2.2.3.2) pode ser representado em termos de valores prévios de W_t e do valor atual e prévios de a_t, que é a forma usual do modelo e é útil para o cálculo de previsões conhecido por forma de equações de diferenças:

$$W_t = \phi_1 W_{t-1} + \phi_2 W_{t-2} + \cdots + \phi_{p+d} W_{t-p-d} + e_t - \theta_1 e_{t-1} - \theta_2 e_{t-2} - \cdots - \theta_q e_{t-q} \quad (2.2.3.3)$$

Em termos de modelagem, a análise do perfil dos gráficos das Funções de Autocorrelação (FAC) e de Funções de Autocorreções Parciais (FACP) pode indicar ordens p, d e q plausíveis para a obtenção de modelos iniciais (HAMILTON, 1994). Após a fase da identificação do modelo, passa-se à próxima etapa que é a estimação dos parâmetros, momento em que é necessário o uso de métodos iterativos não lineares de mínimos quadrados (Levenberg–Marquardt (MARQUARDT, 1963)).

Uma vez obtidas as estimativas dos parâmetros de um modelo Arima (p, d, q), segue-se a sua validação que ocorre por meio de testes de diagnósticos, para a verificação de pressupostos, como, por exemplo: teste de Box-Pierce, teste do periodograma acumulado, teste da autocorrelação cruzada, teste BDS, teste Arch, teste de Multiplicador de Lagrange, estatística de Durbin-Watson e teste de Ljung-Box (HAMILTON, 1994).

Se uma série temporal $y_t (t = 1, \dots, T)$, porém, exibe estrutura de autodependência simples e sazonal, o modelo ARIMA adequado é dado, genericamente, em (2.2.3.4).

$$\phi(B)\Phi(B^S)\nabla_S^D \nabla^d y_t = \theta(B)\Theta(B^S)a_t \quad (2.2.3.4)$$

onde: $\phi(B)$ e $\theta(B)$ são, respectivamente, os polinômios da parte autorregressiva e da parte de médias móveis da componente simples do modelo ARIMA sazonal (SARIMA); $\Phi(B^S) = (1 - \Phi_1 B^S - \cdots - \Phi_P B^{PS})$ e $\Theta(B^S) = (1 - \Theta_1 B^S - \cdots - \Theta_Q B^{QS})$ são, nesta ordem, a parte autorregressiva e de médias móveis da componente de sazonalidade da série temporal subjacente; $\nabla_S^D = (1 - B^S)^D$ é o operador de diferença sazonal de ordem D; $\nabla^d = (1 - B)^d$ é o operador de diferença não sazonal de ordem d; S é o período sazonal, $\phi_k \in \mathbb{C}$ e $\theta_j \in \mathbb{C}$ são os coeficientes complexos dos polinômios não sazonais e $\Phi_m \in \mathbb{C}$ e $\Theta_n \in \mathbb{C}$ são os coeficientes complexos dos polinômios sazonais (LÜTKERPOHL, 2007).

Agora, sendo $y_t (t = 1, \dots, T)$ uma série temporal estacionária (ou não estacionária homogênea) que apresenta estrutura de autodependência linear

(GUPTA e WILTON, 1987). Considere que $\left(\left(x_{1,t}\right)_{t=1}^{T}, \ldots, \left(x_{(r+1),t}\right)_{t=1}^{T}\right)$ é a sequência de $(r+1)$ vetores de realização de $(r+1)$ variáveis exógenas estacionárias da série temporal y_t $(t = 1, \ldots, T)$. Com base em Box & Tiao (1975) e Pankratz (1991), cada realização y_t pode ser representada por meio de um modelo ARIMAX (p, d, q), em que o modelo matemático genérico é formulado conforme segue abaixo:

$$\nabla^d y_t = \phi_0 + \sum_{j=1}^{p} \phi_j y_{t-j} + \sum_{i=1}^{q} \theta_i e_{t-i} + \sum_{i=1}^{r+1} \sum_{l_i=0}^{L_i} \alpha_{il_i} x_{i,t-l_i} + e_t. \qquad (2.2.3.5)$$

Onde: B é o operador de retardo definido por $B^k y_t := y_{t-k}$, com $k \in Z$; $\nabla^d = (1-B)^d$ é o operador diferença, com d representando sua ordem; $(\phi_i)_{i=0}^{p}$, $(\theta_j)_{j=1}^{q}$ são as listas ordenadas de parâmetros complexos, onde $\phi_p \neq 0$ e $\theta_q \neq 0$ e $\left(\alpha_{il_i}\right)_{l_i=0}^{L_i}$ é a i-ésima lista ordenada de vetor de parâmetros complexo associado à componente exógena (estes três vetores de parâmetros precisam satisfazer tanto a invertibilidade quanto as condições de estacionariedade de segunda ordem) (HAMILTON, 1994), (LÜTKERPOHL, 2007); e_t é a inovação no tempo t que consiste na realização de uma variável aleatória ε_t em que ε_t $(t = 1, \ldots, T)$ é um processo estocástico com média zero e variância constante σ^2; e p e q são, respectivamente, as ordens das partes autorregressivas (AR) $\sum_{j=1}^{p} \phi_j y_{t-j}$ (notação: AR(p)) e das partes médias móveis (MA) $\sum_{j=i}^{q} \theta_i e_{t-i}$ (notação: (MA(q)). L_i é a maior ordem do lag em $\sum_{l_i=0}^{L_i} \alpha_{il} x_{i,t-l}$ da variável exógena $x_{i,t}$, onde i=1, ..., (r+1).

Particularmente, nota-se que, se $\alpha_{il} = 0$ para todo $i = 0, \ldots, (r+1)$ e todo $l_i = 0, \ldots, L_i$ então o modelo (2.2.3.5) se torna o tradicional modelo ARIMA (p, d, q). Por sua vez, um modelo estocástico SARIMAX (p, d, q) x $(P, D, Q)_S$, também conhecido como modelo multiplicativo ARIMAX (p, d, q), pode ser usado para modelagem de série temporal sazonal (HAMILTON, 1994) e consiste, genericamente, de um modelo ARIMAX com componentes sazonais.

2.2.4 Redes Neurais Artificiais

As redes neurais artificiais (ou, simplesmente, RNA' s) com uma ou várias camadas escondidas são sistemas constituídos por unidades de processamento

simples, usualmente chamadas de neurônios artificiais (ABELÉM, 1994), (HAYKIN, 2001). As RNA' s podem ser utilizadas para aproximar classes de funções contínuas de suporte compacto (CYBENKO, 1989). Os neurônios artificiais de uma RNA são normalmente organizados em camadas (TAFNER, XEREZ e RODRIGUES FILHO, 1996) interligadas por uma grande quantidade de conexões, chamados pesos sinápticos, os quais servem para ponderar numericamente os padrões de entrada recebidos (RUSSEL e NORVIG, 1995). Ainda de acordo com Tafner, Xerez e Rodrigues Filho (1996), o funcionamento das RNA's é inspirado no cérebro humano, com sua estrutura massivamente paralela, capaz de realizar operações similares (aprendizado, associação, generalização e abstração, etc.), utilizando conhecimento experimental previamente obtido a partir da observação dos padrões (ou sinais). A Figura 8 ilustra um esquema simplificado da estrutura de um neurônio biológico, que serve de inspiração para um neurônio artificial de uma RNA.

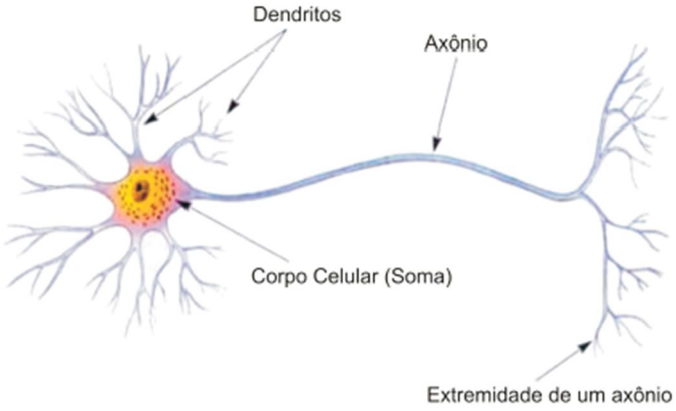


FIGURA 8 - ESQUEMATIZAÇÃO SIMPLIFICADA DE UM NEURÔNIO BIOLÓGICO.
FONTE: Paiva (2015).

De acordo com Tafner, Xerez e Rodrigues Filho (1996), um neurônio biológico possui, basicamente, os seguintes elementos: um corpo celular, que contém o núcleo da célula; diversos dendritos, através dos quais impulsos elétricos são recebidos; e um axônio, por onde são enviados impulsos elétricos. Esses impulsos elétricos passam de um neurônio para o próximo, através das chamadas sinapses, que são pontos de contato acionados por impulsos elétricos e por reações químicas, mediadas pela presença de neurotransmissores, entre o axônio de um neurônio e os dendritos de outros neurônios próximos, compondo, assim, uma rede neural biológica de transmissão e processamento de informações.

Segundo Haykin (2001), um neurônio artificial é constituído pelos elementos descritos, em (i), (ii) e (iii), e ilustrados na Figura 9.

(i) Um conjunto de pesos sinápticos (denotado por $\{\{W_{kn}\}_{k=1}^{K}\}_{n=1}^{N}$), e um conjunto de padrões de entrada (denotado por $\{x_t\}_{t=1}^{T}$), onde W_{kn} é o peso sináptico associado ao n-ésimo valor do padrão de entrada x_t (observado no instante t) e ao k-ésimo neurônio artificial. Cada valor do padrão de entrada x_t é multiplicado pelo respectivo peso sináptico W_{kn}, ou seja, $x_{t,n}W_{kn}$, para todo $n \in \{1, \dots, N\}$, $t \in \{1, \dots, T\}$ e para todo $k \in \{1, \dots, K\}$;

(ii) Um somador (denotado por Σ) para realizar a operação soma dos padrões de entrada em $\{x_t\}_{t=1}^{T}$ ponderados (multiplicados) pelos respectivos pesos sinápticos no conjunto $\{\{W_{kn}\}_{k=1}^{K}\}_{n=1}^{N}$. Isto é, $\sum_{n=1}^{N} x_{t-n}W_{kn} + b_k$, onde: N é o tamanho da janela neural; x_{t-n} é o padrão de entrada, em $t-n$; W_{kn} é o peso sináptico associado ao padrão de entrada x_{t-n} e ao k-ésimo neurônio artificial; e b_k é o intercepto (ou nível, ou bias), que serve para aumentar ou diminuir a entrada líquida da função de ativação, podendo, desse modo, ser positivo ou negativo. A soma ponderada $\sum_{n=1}^{N} x_{t-n}W_{kn} + b_k$ é, usualmente, chamada de regra de propagação (HAYKIN, 2001); e

(iii) Uma função de ativação, denotada por φ(.), para restringir a amplitude da saída y_k de um neurônio artificial, dada a soma ponderada $\sum_{n=1}^{N} x_{t-n}W_{kn} + b_k = v_k$. Isto é, $y_k = \varphi(v_k)$, onde $k \in \{1, \dots, K\}$. Além disso, as funções de ativação, de acordo com Herrera e Lozano (1998), servem para a identificação de relações lineares ou não lineares entre os padrões de entrada e os de saída em modelos neuronais.

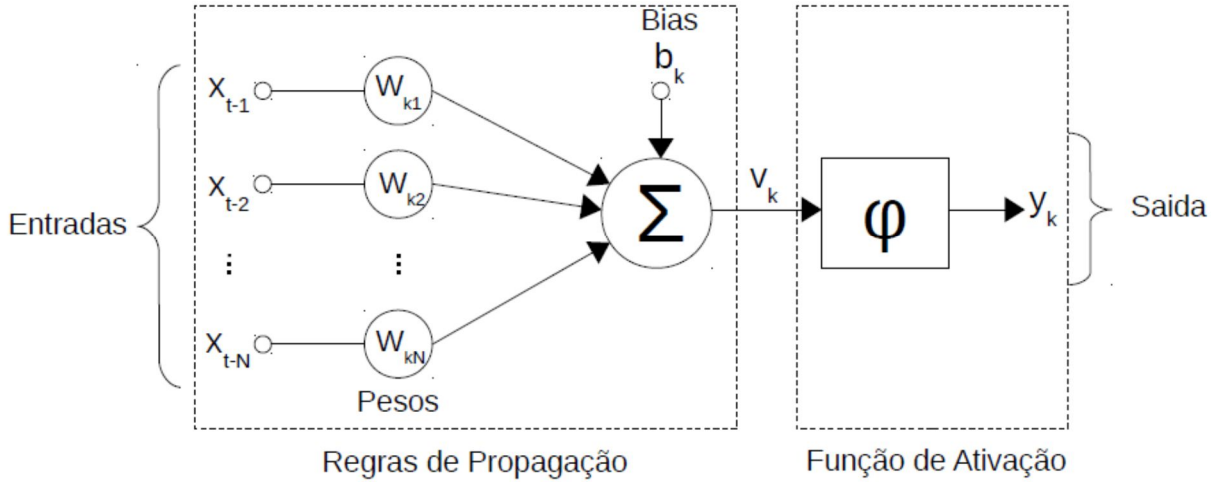


FIGURA 9 - ARQUITETURA BÁSICA DE UM NEURÔNIO ARTIFICIAL.
FONTE: Adaptado de Haykin (2001).

2.2.4.1 Componentes Básicas de uma RNA

Segundo Kovacks (2002), as RNA' s (básicas) podem ser classificadas de acordo com as seguintes características: (i) função de ativação dos neurônios artificiais; (ii) a arquitetura (forma como os neurônios estão interligados); e (iii) O treinamento neural.

(i) Função de Ativação

Dependendo do uso que se pretende fazer da rede neural, uma função de ativação pode ter um resultado melhor do que outra. Por exemplo, se o objetivo da rede é fazer um agrupamento de elementos (clusterização), a função de ativação com melhor resultado provavelmente será diferente de outra rede destinada a fazer previsões de séries temporais. Para Campos (2010), os quatro tipos de funções de ativação mais utilizados em aplicações envolvendo a projeção de séries temporais, são: *linear*, *degrau*, *logística*, *tangente hiperbólica*.

(a) **Função Linear:**

$$y_k = \sum_{t=1}^{n} x_t W_{kt} + b_k. \qquad (2.2.4.1)$$

A função linear, geralmente, é utilizada nos neurônios da camada de saída e é indicada quando as sequências de padrões de saída associados a estes não são conjuntos limitados.

(b) **Função Degrau (ou Limiar):**

$$y_k = \begin{cases} 1, \text{se } v_k \geq 0 \\ 0, \text{se } v_k < 0 \end{cases}. \quad (2.2.4.2)$$

O conjunto imagem da função degrau (ou limiar) é constituído pelos valores 0 ou 1. No entanto, por vezes, é necessário que a sua imagem seja constituída pelos valores -1 ou +1, de modo que, neste caso, a função de limiar pode ser particularmente chamada de função de sinal (HAYKIN, 2001).

(c) **Função Sigmoide Logística:**

$$y_k = \frac{1}{1 + \exp\{-\alpha v_k\}}, \text{onde } \alpha \in \mathbb{R}. \quad (2.2.4.3)$$

A função sigmoide logística aceita valores no intervalo limitado e fechado $[0,1]$ e faz uma transformação monotônica crescente, onde $\sum_{t=1}^{n} x_t W_{kt} + b_k = v_k$ é a regra de propagação do k-ésimo neurônio artificial.

(d) **Tangente Hiperbólica:**

$$y_k = \tanh(\alpha v_k), \text{onde } \alpha \in \mathbb{R}. \quad (2.2.4.4)$$

A função tangente hiperbólica, denotada por $\tanh(.)$, é utilizada quando se deseja que a saída de um neurônio artificial tome valores no intervalo limitado e fechado $[-1, 1]$.

(ii) Arquitetura

A arquitetura de uma RNA diz respeito ao modo como os seus neurônios artificiais estão interconectados. Segundo Russell e Norvig (1995), os dois tipos básicos de arquitetura de RNA' s são a feedforward (não recorrente) (a) e a recorrente (b).

(a) **RNA, com arquitetura *feedforward*:** uma RNA com arquitetura *feedforward* é aquela que não possui conexões entre neurônios da mesma camada ou entre um neurônio de uma camada com outro neurônio de uma camada anterior. A arquitetura básica de uma RNA *feedforward* possui: um conjunto de nós de entradas, que servem para distribuir os padrões de entrada para os seus neurônios artificiais; uma camada de saída, cujos neurônios fornecem as saídas finais da RNA; e camadas intermediárias, cujas saídas dos neurônios são as entradas dos neurônios da camada seguinte (HAYKIN, 2001). Na Figura 10 é apresentado um exemplo de RNA *feedforward*, com uma camada escondida (intermediária ou oculta) e um neurônio na camada de saída.

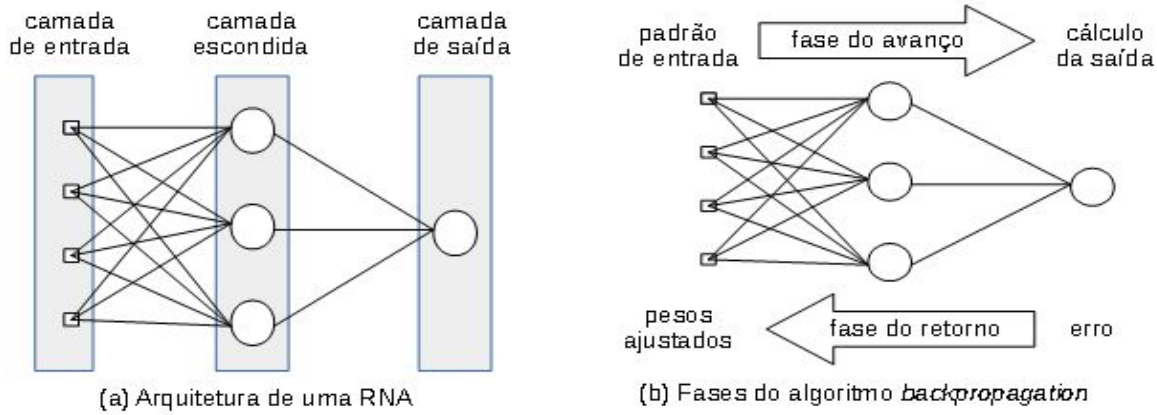


FIGURA 10 - RNA MLP FEEDFORWARD E ALGORITMO BACKPROPAGATION.
FONTE: Adaptado de Haykin (2001).

(b) **RNA' s, com arquitetura recorrente:** as RNA' s com arquitetura recorrente são aquelas que podem apresentar interligações entre neurônios da mesma camada ou ligações partindo de uma camada posterior para camadas anteriores (HAYKIN, 2001).

Segundo Russell e Norvig (1995), a arquitetura de uma RNA possui, basicamente, os seguintes parâmetros: tamanho da janela de entrada, número de camadas escondidas, número de neurônios por camada (entrada, escondidas e de saída), tipo de função de ativação dos neurônios e os padrões de saída.

(iii) Treinamento neural

O treinamento neural de uma RNA consiste em um processo para realizar o ajuste numérico dos pesos sinápticos (usando um algoritmo de otimização plausível), de modo a minimizar (ou maximizar) uma função-objetivo (RAGSDALE, 2010). Para isso, é comum a utilização de um único conjunto de sinais como padrões de entrada e de saída, simultaneamente (a chamada amostra de treino) (CAMPOS, 2010). Usualmente, o treinamento de uma RNA tem por objetivo a minimização do erro quadrático médio (MSE – *Mean Square Error*) entre os padrões de saída produzidos pela RNA e os desejados (HAYKIN, 2001).

Para Herrera e Lozano (1998), é importante que os padrões de entrada e de saída utilizados sejam representativos em relação a uma população de interesse, e que o número de sinais disponíveis seja suficientemente grande para que se consiga uma modelagem adequada (MORETTIN e TOLOI, 2006). Segundo Haykin (2001), existem dois tipos básicos de treinamento neural:

(a) **Treinamento supervisionado:** os padrões de treino são constituídos pelos conjuntos de entrada e de saída desejados. Durante o treinamento, as entradas são apresentadas às RNA' s e, com isso, são calculadas as saídas neuronais. Na sequência, cada saída da RNA é comparada com a respectiva saída desejada, produzindo, desta forma, um sinal de erro (calculado pela diferença entre as duas saídas). A cada ciclo, o algoritmo de treinamento atualiza os pesos sinápticos, com base nos valores de erros obtidos nos padrões de treino, de modo que as saídas da RNA sejam as mais próximas possíveis das saídas desejadas. Esse é o tipo de treinamento geralmente utilizado em previsão de séries temporais; e

(b) **Treinamento não supervisionado:** os padrões de treinamento não contêm padrões de saída desejados, não havendo, portanto, comparações para produzir sinais de erro. O treinamento não supervisionado geralmente visa extrair as propriedades estatísticas do conjunto de padrões de entrada, de modo a formar agrupamentos disjuntos constituídos de padrões similares, com base em uma topologia (KUBRUSLY, 2001), (PALIT e POPOVIC, 2005).

Depois da etapa do treinamento neural, um conjunto de padrões de validação e um conjunto de padrões de teste (da mesma população, mas que não foram apresentados antes) são apresentados à RNA treinada, a fim de validá-la e testá-la (CAMPOS, 2010), (ABELÉM, 1994). Existem vários critérios para a escolha de uma RNA, mas, no caso das RNA' s com treinamento supervisionado (que são as de interesse nesta pesquisa), o erro quadrático médio (nas amostras de treino, validação e teste) é o mais utilizado (RUSSEL e NORVIG, 1995). No caso específico desta pesquisa, são treinados vários modelos de redes neurais com parâmetros diferentes, e o critério para a escolha do melhor modelo é o menor erro quadrático médio sobre a amostra de validação. Os resultados dos erros quadráticos médios sobre a amostra de teste são usados como benchmark de comparação com outras técnicas clássicas usadas na literatura.

2.2.4.2 Padronização

A fim de manter as magnitudes dos padrões de entrada e saída de uma RNA (com treinamento supervisionado), dentro de uma faixa de valores, é fundamental padronizar os padrões. Caso contrário, é possível que os padrões com maiores magnitudes distorçam os valores dos pesos sinápticos no processo de treinamento neural ou produzam problemas associados à paralisia neural (HAYKIN, 2001). Para isso, os padrões devem ser padronizados dentro de um intervalo limitado compatível com o domínio das funções de ativação dos respectivos neurônios da estrutura neuronal. As padronizações de sinais mais comuns, para redes neurais aplicadas à modelagem de séries temporais, são as que os transformam em pontos nos intervalos limitados e fechados [0,1] ou [-1,1]; ou, ainda, em pontos padronizados (ou

seja, elementos de um conjunto de sinais cuja média é igual a 0 e desvio-padrão, igual a 1) (CAMPOS, 2010).

Neste sentido, os métodos de padronizações de padrões MINIMAX, premnmx, z-escore e sigmoidal são, respectivamente, definidos nos itens (i), (ii), (iii) e (iv), a seguir. Em todos os casos, assume-se uma RNA com treinamento supervisionado, e define-se previamente que $\{x_t\}_{t=1}^{n}$ seja, simultaneamente, o conjunto dos padrões de entrada e de saída, e que $\{x_t^*\}_{t=1}^{n}$ seja a sua versão normalizada.

(i) **Método MINIMAX:** utiliza o valor absoluto máximo do conjunto $\{x_t\}_{t=1}^{n}$ de sinais, denotado por $x_{max} = \text{Max}\{\text{abs}(x_t)\}_{t=1}^{n}$, onde $\text{abs}(x_t)$ denota o valor absoluto de x_t. A padronização dos pontos da sequência $\{x_t\}_{t=1}^{n}$ é realizada com a divisão dos sinais pelo valor absoluto máximo, definida em (2.2.4.5), levando a valores no intervalo $[-1, 1]$.

$$x_t^* = \frac{x_t}{x_{max}} \tag{2.2.4.5}$$

(ii) **Método premnmx:** usa os valores máximo e mínimo do conjunto de padrões $\{x_t\}_{t=1}^{n}$, denotados, respectivamente, por $x_{max} = \text{Max}\{x_t\}_{t=1}^{n}$ e $x_{min} = \text{Min}\{x_t\}_{t=1}^{n}$. A padronização de cada ponto x_t é realizada através da transformação definida em (2.2.4.6), que produz valores no intervalo $[-1, 1]$.

$$x_t^* = 2 \times \left[\frac{x_t - x_{min}}{x_{max} - x_{min}}\right] - 1 \tag{2.2.4.6}$$

(iii) **Método do z-escore:** realiza a padronização do conjunto $\{x_t\}_{t=1}^{n}$ em torno de sua média, denotada por $\bar{x}$, e dividindo pelo seu desvio padrão amostral, denotado por S_x, seguindo a transformação definida em (2.2.4.7). A média do conjunto de padrões padronizados $\{x_t^*\}_{t=1}^{n}$ é igual a 0 e o desvio-padrão amostral é igual a 1.

$$x_t^* = \frac{x_t - \bar{x}}{S_x} \tag{2.2.4.7}$$

(iv) **Método Sigmoidal:** também utiliza a média e o desvio-padrão do conjunto $\{x_t\}_{t=1}^{n}$ de padrões, denotados, respectivamente, por $\bar{x}$ e S_x. A padronização de cada ponto x_t em $\{x_t\}_{t=1}^{n}$ é realizada seguindo a transformação definida em (2.2.4.8), produzindo valores no intervalo $[0, 1]$.

$$x_t^* = \frac{1}{1 + \exp\{\frac{x_t - \bar{x}}{S_x}\}} \qquad (2.2.4.8)$$

2.2.4.3 RNA' s multicamadas feedforward, com uma camada escondida

A Figura 10(a) exemplifica a arquitetura de uma RNA perceptron multicamada (MLP – *MultiLayer Perceptron*), com 4 neurônios na camada de entrada (janela = 4), uma camada escondida com 3 neurônios e um neurônio na camada de saída.

Um dos principais algoritmos de treinamento neural é o *backpropagation*. Esse algoritmo realiza o ajuste numérico dos pesos sinápticos, através de um processo de otimização com duas fases básicas: a *forward* e a *backward*, ilustradas na Figura 10 (b). Na fase *forward*, é calculada a resposta fornecida pela RNA a partir de um conjunto de padrões de entrada, que depende do tamanho da janela neural. Na fase *backward*, o erro (diferença) entre a resposta da RNA e a resposta desejada é utilizado no processo de atualização dos pesos sinápticos. Ao longo do treinamento, os padrões de entrada e as respectivas respostas desejadas são apresentados à RNA, de modo a minimizar o erro quadrático médio, da amostra de treino, entre os sinais de saída da RNA e os sinais de saída desejados.

2.2.4.4 Geração de previsões através de redes neurais artificiais

A previsão de valores futuros de uma série temporal, com o uso de uma RNA MLP, *feedforward,* com treinamento supervisionado, com uma camada escondida e um neurônio na camada de saída, é possível porque as suas observações (sinais) podem ser interpretadas como pontos de alguma função e, de acordo com a seção 2.2.4.3, tais RNA' s servem para aproximar este tipo de

mapeamento. Em uma série temporal com estrutura de dependência temporal autoregressiva (linear ou não linear), o conjunto de padrões de entrada é formado pelos valores passados da própria série temporal que se deseja modelar (HAMILTON, 1994). O padrão de saída desejado, neste caso, é o próximo valor da série, ou seja, o valor a ser previsto. Desta forma, define-se previamente o tamanho da janela neural L e o horizonte de previsão h desejado.

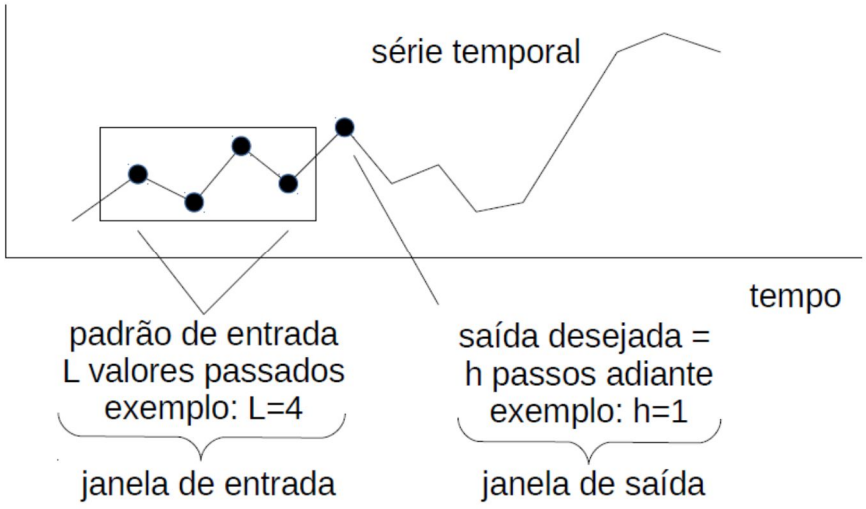


FIGURA 11 – USO DE UMA RNA COM JANELA 4 PARA A PREVISÃO, 1 PASSO À FRENTE, DE UMA SÉRIE TEMPORAL

FONTE: Adaptado de Haykin (2001).

A Figura 11 ilustra como é realizada a previsão, 1 passo à frente, para uma série temporal com estrutura de autodependência com defasagem igual a 4 (ou seja, o tamanho da janela L é igual a 4), por uma RNA *feedforward*. Observe que o vetor de padrões de entrada é composto pelos 4 últimos valores da série, antes do valor de saída a ser previsto. Para prever o próximo valor, basta deslocar a janela da RNA um passo à frente (CAMPOS, 2010).

2.2.4.5 Redes neurais artificiais com variáveis exógenas – RNAX

Além de considerar os valores anteriores da própria série temporal de subpressão, uma rede neural pode levar em conta também valores das séries

temporais de variáveis exógenas (ou causais). Neste caso, para diferenciar das redes neurais tratadas até aqui, esse tipo de rede neural será referenciada por RNAX. A composição dos padrões de entrada leva em consideração os valores correntes (e possivelmente passados) das séries temporais das variáveis causais, e os valores anteriores da série temporal de saída.

A título de ilustração, supondo-se que a série temporal mensal de subpressão $\mathrm{y_t}\ (t = 1, \ldots, \mathrm{T})$ seja influenciada por outras duas variáveis causais $\mathrm{x1_t}\ (t = 1, \ldots, \mathrm{T})$, sem defasagem de tempo, e $\mathrm{x2_t}\ (t = 1, \ldots, \mathrm{T})$, com defasagem de três meses. Considerando ainda que o tamanho da janela neural de entrada seja $L = 4$, ou seja, o valor de $\mathrm{y_t}$ é influenciado por $\mathrm{y_{t-1}}$, $\mathrm{y_{t-2}}$, $\mathrm{y_{t-3}}$ e $\mathrm{y_{t-4}}$. Assim, a composição dos padrões de entrada e saída segue o padrão definido em (2.2.4.9)

$$\begin{pmatrix} x1_5 & x2_2 & y_1 & y_2 & y_3 & y_4 \\ x1_6 & x2_3 & y_2 & y_3 & y_4 & y_5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x1_T & x2_{T-3} & y_{T-4} & y_{T-3} & y_{T-2} & y_{T-1} \end{pmatrix} \qquad \begin{pmatrix} y_5 \\ y_6 \\ \vdots \\ y_T \end{pmatrix} \tag{2.2.4.9}$$

Padrões de entrada Padrões de saída

Cabe destacar a perda dos graus de liberdade ocasionada pelo tamanho da janela de 4 valores. Assim, são usados os valores dos 4 primeiros meses para prever o quinto mês, do mês 2 ao mês 5 para prever o sexto mês e assim sucessivamente. Da mesma forma, entram no padrão de treinamento para a previsão do mês 5 (y_5), as variáveis causais $x1_5$ (mês corrente), e $x2_2$ (com defasagem de três meses). Generalizando, pode-se escrever:

$$\hat{y}_t = f(x1_t, x2_{t-3}, y_{t-4}, y_{t-3}, y_{t-2}, y_{t-1}), (t = 5, \ldots, T) \tag{2.2.4.10}$$

2.2.5 Combinação de métodos preditivos individuais

Dada uma série temporal $\mathrm{y_t}\ (t = 1, \ldots, \mathrm{T})$ (por exemplo, uma média mensal de subpressão), que se quer modelar, a fim de se gerar previsões, dentro e fora da amostra. Considere que se tenha disponível um conjunto com M métodos preditivos plausíveis, o qual é denotado por $\{\mu_\mathrm{m}\}_{\mathrm{m=1}}^{\mathrm{M}}$, onde $\mathrm{M} \in \mathbb{N}$, sendo $\mathbb{N}$ o conjunto dos

números inteiros positivos. Segundo Faria e Mubwandarikwa (2008), há, basicamente, duas maneiras para se determinar um previsor para a série temporal supracitada:

1. escolher um dos métodos preditivos em $\{\mu_m\}_{m=1}^{M}$, baseado em algum critério de seleção; ou
2. escolher K métodos preditivos em $\{\mu_m\}_{m=1}^{M}$ e, em seguida, combiná-los (de forma linear ou não linear).

Para a combinação de métodos preditivos, é importante se considerar dois aspectos (FLORES e WHITE, 1988): o primeiro se refere à escolha dos K métodos preditivos com base em $\{\mu_m\}_{m=1}^{M}$; e o segundo diz respeito à forma de combiná-los. Em relação ao primeiro aspecto, para os autores, as previsões produzidas pelos métodos preditivos base podem ser classificadas em três categorias: objetivas, subjetivas e mistas (ou seja, as resultantes da combinação de previsões objetivas e subjetivas). As previsões objetivas são aquelas produzidas por previsores com base teórica estatística ou matemática; enquanto que as previsões subjetivas são as que dependem de julgamento humano - geralmente opinião de especialistas (individuais ou em grupo) (WERNER e RIBEIRO, 2006). Cabe destacar que, nesta tese, somente foram utilizadas combinações de previsões objetivas. Quanto ao segundo aspecto, embora exista um consenso entre os pesquisadores de que a combinação de previsores individuais, em geral, produza ganhos preditivos na projeção de séries temporais, não há um consenso sobre qual a melhor maneira de se fazê-la (WALLIS, 2011), (CAVELERI e RIBEIRO, 2011).

Para a definição genérica de combinação de previsões, considere uma série temporal $y_t\,(t = 1,\ldots,T)$, a qual se deseja modelar, a fim de se gerar previsões, dentro e fora da amostra, através de uma função de combinação de métodos preditivos base. Seja ∇^K o conjunto de todas as previsões (dentro e fora da amostra) produzidas pelos K métodos preditivos individuais escolhidos, pelo tomador de decisão, em $\{\mu_m\}_{m=1}^{M}$, sendo que $K \leq M$; e tome um mapeamento $\hat{y}_C: \begin{cases} \nabla^k \to \mathbb{R} \\ [\hat{y}_{t,k}]_{k=1}^{K} \in \nabla^k \to \hat{y}_{t,C} \in \mathbb{R} \end{cases}$ de combinação de previsões (genérico) que transforma, de acordo com sua definição, um vetor $[\hat{y}_{t,k}]_{k=1}^{K} \in \nabla^k$ de previsões em uma previsão

combinada $\hat{y}_{t,C} \in \mathbb{R}$, para todo $t = 1, \dots, T + h$, onde $h \in \mathbb{N}$ é o horizonte de previsão. Nesta tese, em particular, é utilizada uma combinação não linear iterativa como $\hat{y}_C$ a qual é detalhada no capítulo 3.

2.2.6 A técnica *Bootstrap*

O método *Bootstrap* (EFRON, 1979), é um procedimento estatístico computacionalmente intensivo, não paramétrico que consiste na reamostragem de mesmo tamanho e com reposição dos dados da amostra original, com o objetivo de se estimar uma medida de interesse. O princípio básico do método *Bootstrap* estabelece que quando não se tem informações sobre a verdadeira distribuição de uma variável aleatória, é possível obter-se uma distribuição empírica a partir de reamostragens de uma amostra observada. Partindo-se do princípio que esta amostra possui informações da distribuição da variável aleatória, B amostras daquela permitirão estimar os parâmetros da população da qual foi retirada.

Seja $X = (X_1, X_2, \dots, X_n)$ uma amostra aleatória de uma distribuição F e $\hat{\theta} = s(X)$ estimador de um parâmetro θ desta distribuição. Uma amostra *Bootstrap* $X^* = (X_1^*, X_2^*, \dots, X_n^*)$ é obtida da amostra original X rearranjando os seus n elementos. Assim, uma das possibilidades para X^* é $X^* = (X_1^* = X_n, X_2^* = X_1, \dots, X_n^* = X_2)$, mantendo a probabilidade de escolha em X de $1/n$ para cada uma das suas observações. Desta forma determinam-se B amostras *Bootstrap* independentes da distribuição empírica $\hat{F}$, obtendo-se então o estimador *Bootstrap* $\hat{\theta}^{*(b)} = s(X^{*(b)})$, com $b = 1,2, \dots, B$, do parâmetro θ. O erro padrão do estimador *Bootstrap* é representado por (2.2.6.1).

$$\widehat{se}_{boot}(\hat{\theta}) = \left[\frac{1}{B-1} \sum_{b=1}^{B} \left(\hat{\theta}^{*(b)} - \hat{\theta}^{*} \right)^2 \right]^{1/2} \qquad (2.2.6.1)$$

Onde: $\hat{\theta}^{*} = \frac{1}{B} \sum_{b=1}^{B} \hat{\theta}^{*(b)}$.

A escolha do número B de amostras *Bootstrap* pode tomar por base a observação da variação do desvio padrão do estimador *Bootstrap*. O valor de B mais

adequado é indicado por uma baixa variabilidade ou estabilidade do valor desse estimador (MONTGOMERY e RUNGER, 2012).

Em se tratando de previsão de séries temporais, existem duas maneiras para se aplicar o *Bootstrap* em uma série de tempo: (a) *Bootstrap* nos resíduos; (b) método *moving blocks*. Sendo o primeiro mais usado que o segundo em aplicações de *Bootstrap* em séries temporais. Mais detalhes sobre a técnica *Bootstrap* podem ser encontrados em Freedman e Peters (1984), Efron e Tibshirani (1986) e Souza e Neto (1996).

2.2.6.1 Intervalos de previsão

Para aplicar modelos de regressão é necessário realizar a avaliação das incertezas inerentes aos estimadores de parâmetros e às variáveis aleatórias. Os intervalos de confiança e de previsão são duas medidas estatísticas usadas nessa avaliação. O primeiro está relacionado às variações associadas aos parâmetros do modelo de regressão, ao passo que o segundo é mais amplo e considera tanto as variações devidas ao modelo de regressão, quanto as variações relativas aos erros preditivos.

Considere os padrões $(y_t, x_t)\,(t = 1, \ldots, T)$ e o modelo de regressão neural $y(x) = f(x, w) + e(x)$, onde: $y(x)$ são os alvos; w são os parâmetros populacionais do modelo desconhecido $f(x, w)$; e $e(x)$ o ruído (variável aleatória) com média zero. Seja $g(x, \hat{w})$, onde $\hat{w}$ são os estimadores de w, a aproximação de $f(x, w)$, que é interpretada como a média dos alvos (y), dada a entrada (input) x.

Para problemas de regressão podem ser considerados dois diferentes aspectos (HESKES, 1997): a acurácia da estimativa da regressão verdadeira e a acurácia das estimativas relativas às saídas observadas. O primeiro aspecto é relativo à diferença $f(x, w) - g(x, \hat{w})$ e leva à construção do intervalo de confiança. Já o segundo aspecto está relacionado à variação $y(x) - g(x, \hat{w})$, associada ao intervalo de previsão, conforme (2.2.6.2). Observe que o intervalo de confiança está contido no intervalo de previsão.

$$y(x) - g(x, \hat{w}) = f(x, w) - g(x, \hat{w}) + e(x) \qquad (2.2.6.2)$$

Considerando (2.2.6.2), a variância total associada com o resultado da modelagem pode ser expresso como em (2.2.6.3) (KHOSRAVI, NAHAVANDI, *et al.*, 2014).

$$\sigma_y^2(x) = \sigma_g^2(x) + \sigma_e^2(x) \tag{2.2.6.3}$$

Onde $\sigma_g^2(x)$ é a incerteza do modelo de regressão ajustado e $\sigma_e^2(x)$ a medida da variância do ruído $e(x)$. Tibshirani (1996) sugere, no contexto das regressões, dois tipos de amostragens *Bootstrap* que permitem determinar o intervalo de confiança: *Bootstrap pairs* e *Bootstrap* residual. A seguir, um resumo dos passos que constituem o algoritmo da amostragem *Bootstrap pairs*:

(1) Através de reamostragem das n observações da amostra de treino, representada por $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, gerar B amostras *Bootstrap* de tamanho n, denotadas por $\{(x_1^{*b}, y_1^{*b}), (x_2^{*b}, y_2^{*b}), \dots, (x_n^{*b}, y_n^{*b})\}$, com $b = 1, \dots, B$ e $(20 \leq B \leq 200)$;

(2) Para cada amostra *Bootstrap*, minimize $\sum_{i=1}^{n}[y_i^{*b} - g(x_i^*, w)]^2$ para obter $\hat{w}^{*b}$; e

(3) Estimar o erro padrão do i-ésimo valor previsto usando o seguinte estimador:

$$\left\{\frac{1}{B-1}\sum_{b=1}^{B}[g(x_i, \hat{w}^{*b}) - m(x_i)]^2\right\}^{1/2} \tag{2.2.6.4}$$

onde $m(x_i) = \frac{1}{B}\sum_{b=1}^{B} g(x_i, \hat{w}^{*b})$.

O algoritmo do *Bootstrap* residual apresenta alguns passos em comum com o *Bootstrap pairs*. A principal diferença reside na construção das séries "Bootstrapadas". Segue o procedimento para execução do *Bootstrap* residual:

(1) Determinar $\hat{w}$ a partir dos dados observados, fazendo $r_i = y_i - g(x_i, \hat{w})$, com $i = 1, \dots, n$;

(2) Gerar B amostras *Bootstrap*, cada uma com tamanho n, representadas por $r_1^{*b}, r_2^{*b}, \dots, r_n^{*b}$, com $b = 1, 2, \dots, B$, a partir da reamostragem de $r_1, r_2, \dots, r_n$. Faça $y_i^{*b} = g(x_i, \hat{w}) + r_i^{*b}$;

(3) Determine o valor de $\hat{w}^{*b}$ para cada amostra *Bootstrap*, minimizando $\sum_{i=1}^{n}\left[y_i^{*b} - g(x_i^*, w)\right]^2$; e

(4) Estimar o erro padrão do i-ésimo valor previsto, usando:

$$\left\{\frac{1}{B-1}\sum_{b=1}^{B}\left[g\left(x_i, \hat{w}^{*b}\right) - m(x_i)\right]^2\right\}^{1/2} \tag{2.2.6.5}$$

onde $m(x_i) = \frac{1}{B}\sum_{b=1}^{B} g\left(x_i, \hat{w}^{*b}\right)$.

De acordo com Khosravi *et al* (2014), o intervalo de previsão pode ser expresso por (2.2.6.5).

$$m(x) - t_{1-\frac{\alpha}{2},gl}\sqrt{\sigma_y^2(x)} \le y(x) \le m(x) + t_{1-\frac{\alpha}{2},gl}\sqrt{\sigma_y^2(x)} \tag{2.2.6.5}$$

Onde: gl são graus de liberdade; e α o nível de significância em uma distribuição t de Student. Os autores definiram gl como sendo a diferença entre o número de amostras de treinamento e o número de parâmetros do modelo RNA. Seguindo nesta linha, o intervalo de confiança pode ser escrito na forma representada em (2.2.6.6).

$$m(x) - t_{1-\frac{\alpha}{2},gl}\sqrt{\sigma_g^2(x)} \le f(x, w) \le m(x) + t_{1-\frac{\alpha}{2},gl}\sqrt{\sigma_g^2(x)} \tag{2.2.6.6}$$

Carney; Cunningham; Bhagwan (1999) obtiveram intervalo de confiança e previsão a partir de uma combinação de diversos resultados de previsão de RNA com *Bootstrap*. Os autores sugerem dividir o conjunto com B redes *Bootstrap* em M subconjuntos de tamanho β, executando os seguintes passos:

(1) Determinar $m^{*i}(x)$ em cada um dos M subconjuntos formados por β redes e montar o conjunto $E = \left\{m^{*i}(x)\right\}_{i=1}^{M}$;

(2) Obter B^* reamostragens *Bootstrap* de E, formando o conjunto $G = \{E_j^*\}_{j=1}^{B^*}$, onde $E_j^* = \{m^{*j1}(x), \dots, m^{*jM}(x)\}$;

(3) Calcular a variância em cada um dos B^* conjuntos E_j^*: $\sigma_j^{2*} = \frac{1}{M}\sum_{k=1}^{M}\left(m^{*jk}(x) - m^{*j}(x)\right)^2$; e

(4) Fazer a média das variâncias: $\sigma_g^2 = \frac{1}{B^*}\sum_{j=1}^{B^*}\sigma_j^{2*}$.

Tendo agora uma melhor estimativa para variância da distribuição de probabilidade $P(f(x,w)|m^*(x))$, constrói-se o IC para $f(x,w)$, conforme (2.2.6.7).

$$m^*(x) - z_{1-\alpha}\sigma_g \leq f(x,w) \leq m^*(x) + z_{1-\alpha}\sigma_g \qquad (2.2.6.7)$$

Onde: $m^*(x)$ é obtido a partir das B amostragens *Bootstrap* e $z_{1-\alpha}$ provêm da distribuição normal padrão com nível de significância α. Nix e Weigend (1995) sugerem uma técnica para estimar a média e a variância de uma distribuição de probabilidades de alvos em um problema de regressão por redes neurais artificiais. Os autores propõem treinar uma rede neural, usando, na formação da função custo, a máxima verossimilhança, conforme descrita em (2.2.6.8).

$$C = \frac{1}{2}\sum_i\left[\frac{(d(x_i)-y(x_i))^2}{\sigma_e^2(x_i)} + \ln(\sigma_e^2(x_i))\right] \qquad (2.2.6.8)$$

Onde $d(x)$ são os valores observados (alvos) e $y(x) \approx f(x)$ no modelo $d(x) = f(x) + e(x)$. Esta proposta é frequentemente adaptada para calcular a variância do ruído ($\sigma_e^2(x)$), sendo utilizada em vários trabalhos, tais como Heskes (1997), Carney; Cunningham; Bhagwan (1999) e Khosravi *et al.*, (2014).

Bartkiewicz (2012) realizou previsões através de redes neuro-fuzzy e estimou a variância do ruído ($\sigma_e^2(x)$) treinando uma rede com padrões $\{x_i, |e_i|\}$, para $i = 1, \dots n$, sendo $|e_i| = |y_i - g(x_i, \hat{w})|$. Na comparação entre o intervalo de previsão determinado através da sua proposta com intervalos calculados através de outras técnicas, o autor concluiu que os seus resultados foram satisfatórios.

2.3 Revisão de Literatura

Há uma escassez de trabalhos publicados envolvendo previsões de séries temporais de subpressão, ou mesmo de definição de limites de alerta para subpressão.

Kuperman *et al.* (2003) propõe o uso de regressão múltipla, definindo o alerta como o limite do intervalo de confiança da previsão, quando R^2 ajustado maior que 0,7, ou seja, se as variáveis causais conseguem explicar mais de 70% da variação sazonal da subpressão. Caso contrário, é estabelecido um intervalo de valores que contenham 95% das leituras da série temporal de subpressão e o alerta passa a ser o limite superior do intervalo estabelecido.

Para o caso das leituras de subpressão dos piezômetros de Itaipu, pode-se observar claramente uma diferença de comportamento entre os piezômetros localizados na feição contato concreto-rocha, que apresentam a maior oscilação sazonal de subpressão, e os piezômetros localizados em feições mais profundas, cujas subpressões se apresentam cada vez mais estáveis conforme a profundidade aumenta. Para os casos de piezômetros mais profundos, a abordagem utilizada por Kuperman *et al.* (2003) pode ser aplicada sem problemas. Entretanto, para os piezômetros de contato concreto-rocha analisados, a regressão múltipla não é capaz de explicar 70% da variação da subpressão, e outra técnica torna-se necessária.

3 MATERIAIS E MÉTODOS

Neste capítulo são descritos os materiais utilizados, incluindo dados e software e o método desenvolvido para atender aos objetivos propostos na tese.

3.1 Materiais utilizados

3.1.1 Descrição dos dados disponíveis

O instrumento usado para medir a subpressão é chamado de Piezômetro. Esse instrumento é composto de um duto inserido em um furo na fundação rochosa. Na ponta do duto existem furos, que permitem a entrada de água dentro do duto. Essa parte do furo em que o duto está inserido é preenchida com areia, e localizada em alguma feição que se deseja monitorar a subpressão. Acima e abaixo dessa feição, o furo é preenchido com calda de cimento, de modo a não permitir que infiltrações de água de outras feições entrem no tubo. Pelo princípio dos vasos comunicantes, a água que se infiltra por essa feição entra pelo tubo e sobre até equilibrar a pressão. A aferição da pressão é realizada medindo-se a altura da coluna d'água no interior do tubo.

A unidade de medida utilizada neste trabalho para a subpressão é a cota piezométrica, especificada em metros sobre o nível do mar. Cada piezômetro tem uma cota de instalação. Por exemplo, se o piezômetro está instalado na cota 45, significa que ele está 45 metros acima do nível do mar. Se a partir dele é constatada uma leitura de uma coluna de água com 10 metros de altura, ela é registrada com a cota de 55 metros sobre o nível do mar.

Os intervalos entre as leituras realizadas nos piezômetros sofreram variações significativas devido a vários fatores. No período de enchimento do reservatório, por exemplo, as leituras dos piezômetros foram mais frequentes, chegando em alguns casos a leituras diárias. Com o passar do tempo e a observação do comportamento real da subpressão, as leituras passaram a ser feitas a intervalos maiores. Esse intervalo também passou a ser diferente de acordo com o

comportamento da subpressão, sendo semanal para os piezômetros com maior oscilação da subpressão, e quinzenais ou mesmo mensais para os outros piezômetros localizados em feições com variação pequena, ou mesmo praticamente nula na subpressão. Outro fator que pode afetar a frequência de leitura de piezômetros é a ocorrência de procedimentos de manutenção, tais como desentupimentos de drenos ou perfuração de novos drenos.

A temperatura do ar está disponível em leituras médias diárias, desde o período da execução da obra, e o nível do reservatório também está disponível em médias diárias a partir da data do seu enchimento. Todas as temperaturas estão registradas em graus Celsius.

Ao longo do tempo, a série histórica de subpressão dos piezômetros de contato concreto-rocha apresenta variações significativas de regime, o que torna sua análise mais complexa. Há um momento crítico que é o período do enchimento do reservatório, e depois um período de acomodação. Os dados desse período inicial não foram considerados na análise, pois as condições atuais não são as mesmas do período inicial de operação do reservatório.

Foram realizadas visitas técnicas às fundações da barragem, entrevistas com os engenheiros, coleta de dados das leituras dos instrumentos, pesquisa em documentos técnicos e estudo de obras técnicas sobre hidrogeotecnia de maciços rochosos fraturados aplicada a fundações de barragens de concreto. Com isso foram identificadas possíveis influências de:

a) temperatura ambiente, atribuída à dilatação térmica diferente a que estão sujeitas a face de jusante da barragem (seca) e a face de montante (em contato com o reservatório), ocasionando diferenças na tensão imposta à fundação nas proximidades do pé de montante. No caso específico desta barragem, a influência não é imediata, devido à propagação mais lenta da temperatura na estrutura de concreto de vários metros de espessura (FIORINI, PORTO, *et al.*, 2001);
b) temperatura da água que passa pelos piezômetros que monitoram a pressão, nos casos em que essa informação está disponível, em função da abertura e obstrução das descontinuidades na fundação ocasionadas pelo efeito da dilatação e contração térmica provocada pela passagem de água em diferentes temperaturas. Uma variação de 20ºC para 19ºC na

água que passa por uma descontinuidade da fundação pode provocar uma alteração de 110% na vazão que passa por ela, alterando significativamente a subpressão na região (ANDRADE, 1988);

c) nível do reservatório, que influencia diretamente a pressão hidrostática na fundação e a pressão exercida sobre a face de montante da barragem, possivelmente alterando as tensões no pé de montante (MASCARENHAS, 2005). Em função da pequena variação do nível do reservatório, essa influência não é percebida visualmente pela análise dos gráficos, mas a variável foi mantida para ser verificada estatisticamente;
d) intervenções pontuais sobre o sistema, tais como limpeza de drenos ou sismos, no caso de alguns piezômetros. A limpeza de drenos, procedimento de manutenção executado poucas vezes foi capaz de reduzir a subpressão significativamente em alguns piezômetros, e isso independe das outras variáveis independentes. A região onde está instalada a barragem de Itaipu não é sujeita a abalos sísmicos, mas a barragem possui sismógrafos, e alguns grandes tremores na região dos Andes podem provocar alterações de comportamento nos piezômetros. Essas alterações podem ser apenas pontuais, ou uma mudança permanente no nível de subpressão, para mais ou para menos. Para manter o modelo de modo mais próximo ao comportamento real da subpressão é necessário verificar se há esse tipo de intervenção pontual e modelar pela adição de novas variáveis que representem a sua influência. Para o caso dos piezômetros estudados neste trabalho, não foi detectada intervenção significativa, em função de sua localização.

A barragem de Itaipu possui leituras manuais desde a fase de sua construção com dados a partir de 1980. Em 2005 foi instalado um sistema para a coleta automática de dados de 270 sensores a cada 30 minutos, incluindo piezômetros (ITAIPU, 2008). Os leitores automáticos dos piezômetros fornecem também a temperatura da água que está passando por ele. Mesmo com a leitura automática, a leitura manual foi mantida, e é considerada mais confiável. Assim decidiu-se pelo uso das séries de dados manuais de subpressão, junto com as leituras automáticas de temperatura da água, pelo período em que todas as variáveis estivessem disponíveis, ou seja, de 2006 a 2015.

Nesta pesquisa foram identificadas:

a) uma correlação forte entre a temperatura ambiente e a subpressão, com uma defasagem de aproximadamente 3 meses, com a temperatura máxima no verão levando a uma subpressão mínima no outono, e a temperatura mínima no inverno induzindo a uma subpressão máxima na primavera. Como a temperatura ambiente varia bastante de um dia para o outro, e a leitura da subpressão só está disponível semanalmente, e ainda assim apresenta variação mais suave, é mais razoável utilizar temperaturas médias de períodos maiores, para detectar sua influência na subpressão;
b) possível influência da temperatura da água que passa pelo piezômetro, influenciando a variação de subpressão quase que simultaneamente à variação de temperatura;
c) resfriamento rápido da temperatura da água na fundação quando a temperatura ambiente cai abaixo de 20ºC aproximadamente, atribuído à convecção da água do reservatório;
d) aquecimento gradual e lento da temperatura da água na fundação, e sem ultrapassar 21.1ºC quando a temperatura sobe, atribuído à condução de calor pela água do reservatório;
e) não encontrada visualmente influência do nível do reservatório no nível de subpressão. Sua influência foi considerada estatisticamente insignificante na variação da subpressão, e a partir de então os modelos passaram a não mais incluir essa variável.

As leituras de subpressão utilizadas são provenientes de leituras manuais, executadas desde a instalação dos instrumentos, durante a construção da barragem. Pelo fato de a obra ter passado por momentos críticos, como a etapa de enchimento do reservatório, seguida pelos primeiros anos de operação, quando não se conhecia o comportamento da obra, as leituras eram realizadas com maior frequência. Em seguida, com o acúmulo de informações, as leituras passaram a ser feitas com intervalos maiores. A periodicidade de leituras também passou a variar de instrumento para instrumento, sendo lidos a intervalos menores os que apresentam maior variação. Para o piezômetro em estudo há leituras com frequência diária, duas vezes por semana e semanal. Optou-se pela frequência semanal das leituras,

disponíveis em todo o período. Detalhes menores passaram a ser corrigidos, como a eliminação de leituras extras realizadas durante procedimentos de manutenção, leituras duplicadas e a estimativa das raras leituras ausentes pela média entre os vizinhos. Essa estimativa de valores mostrou-se consistente com as leituras automáticas, que também se mantiveram próximas à média das leituras das semanas anterior e seguinte. Em função do cálculo de sazonalidade, também foi definido um número fixo de 52 leituras por ano, pois o número apresentava pequenas variações dependendo do dia da semana em que o ano se iniciou.

Para cada uma das variáveis independentes (temperatura ambiente, temperatura da água na fundação e nível do reservatório) foram estabelecidas a média diária, a média dos últimos 7 dias, a média dos últimos 15 dias, a média dos últimos 30 dias e a média dos últimos 90 dias, de modo a verificar quais as séries que melhor representam a influência da variável na leitura da subpressão. Para cada data com leitura de subpressão disponível foram acrescentadas as médias de cada variável independente para cada um dos períodos acima. Identificou-se que as curvas de médias mensais apresentaram correlação mais forte entre a subpressão e as variáveis independentes, entre todos os períodos pesquisados. Foram acrescentadas também as leituras das médias mensais de temperatura ambiente com defasagem de 1, 2, 3 e 4 meses, uma vez que há o indício visual a partir do gráfico das séries de que a subpressão é afetada pela temperatura ambiente ocorrida 3 meses antes.

3.1.2 Descrição do software utilizado

Para a realização do processamento dos dados foi utilizado o programa estatístico *R* (R CORE TEAM, 2015). A escolha do programa foi motivada por se tratar de um software livre, sem custo de licença, pela sua ampla disponibilidade de pacotes estatísticos e pela sua farta documentação e facilidade de programação.

Junto com os módulos básicos, instalados automaticamente com o *R*, foram utilizados os pacotes *forecast* (HYNDMAN, 2015), (HYNDMAN e KHANDAKAR, 2008), *Rssa* (GOLYANDINA e KOROBEYNIKOV, 2014), (KOROBEYNIKOV, 2010), *waveslim* (WHITCHER, 2015), *RSNNS* (BERGMEIR e BENITEZ, 2012).

A produção de alguns gráficos foi realizada com o auxílio do software Microsoft Excell.

3.2 Método Proposto

Seja $y_t(t = 1,\ldots,T)$ uma série temporal de tamanho igual a T. De acordo com Zhang (2003), cada estado y_t pode ser decomposto como segue: $y_t = CL_t + CN_t$, sendo que CL_t e CN_t representam, respectivamente, as componentes linear e não linear do estado y_t. Com efeito, uma previsão de y_t, denotada por $\hat{y}_t$, é algebricamente dada por $\hat{y}_t = \widehat{CL}_t + \widehat{CN}_t$, onde: $\widehat{CL}_t$ é a previsão de CL_t, que é produzida por um previsor linear; e $\widehat{CN}_t$ é a predição de CN_t, gerada por um modelo não linear.

Aqui, a previsão linear $\hat{L}_t$ e a não linear $\hat{N}_t$ são, respectivamente, produzidas por um modelo linear ARIMAX, descrito na seção 2.2.3, e por previsor não linear, referido por Combinação Neural SSA-Wavelet (CNSSAW) iterativa, proposta em Royer et al (2016). Além disso, a previsão híbrida do estado y_t, que é representada por $\hat{y}_t$, é definida por $\hat{y}_t = \frac{\hat{L}_t + \hat{N}_t}{2}$; os intervalos de previsão associados a cada $\hat{y}_t$ são calculados via *Bootstrap*, descrito na seção 2.2.6. Do ponto de vista aplicado, o método híbrido aqui proposto visa gerar previsões mensais associadas a intervalos de previsão para as séries temporais de leituras de subpressão em Barragens de concreto, na usina de Itaipu. Assim, o mesmo pode ser operacionalizado conforme os cinco passos, abaixo:

Passo 1: as condições de contorno (ou variáveis exógenas) são modeladas e previstas pontualmente através de modelos ARIMA automáticos, utilizando o pacote *forecast* (HYNDMAN, 2015) do software *R* (R CORE TEAM, 2015);

Passo 2: a série temporal de supressão é modelada e predita h passos à frente por meio de um modelo ARIMAX, o qual utiliza as condições de contorno completadas pelas suas respectivas previsões no Passo 1. Assim, são produzidas as previsões, h passos à frente, da componente linear, $(L_t)_{t=1}^{T}$, da série temporal

subjacente. Neste caso, a identificação de um modelo preditivo ARIMAX plausível segue os seguintes passos, adaptados de Pankratz (1991):

a) analisar estatisticamente as séries de dados e dos fatores que podem influenciar a variável subpressão, com consultas aos especialistas;
b) traçar os gráficos das séries, identificando possíveis correlações entre as variáveis independentes e a subpressão, suas prováveis defasagens de tempo, e observando a ocorrência de *outliers*, que podem ocorrer devido a erros de leitura, ou influência de eventos históricos pontuais (intervenções);
c) regularizar a série temporal de leitura da subpressão, estabelecendo um intervalo regular para as leituras e eliminando eventuais duplicações de registros;
d) estabelecer médias de leituras diárias, semanais, quinzenais, mensais e trimestrais, para todas as variáveis independentes, e também nas leituras de subpressão, para identificar qual o intervalo que mais explica a variação da subpressão. Inclusão de colunas com as leituras anteriores $(t-1, t-2, \ldots, t-k)$ para as variáveis onde existe suspeita de influência de leituras anteriores;
e) realizar testes de modelagem ARIMA sobre a série histórica, e verificar se o modelo encontrado é satisfatório; e
f) realizar testes de significância através da regressão múltipla com todas as variáveis independentes, e para cada intervalo de leitura média. Verificar se a regressão alcançou nível aceitável de explicação da variação da subpressão pelo teste do R^2 ajustado, e se os resíduos se apresentam como ruído branco. Caso o teste do R^2 ajustado não seja adequado e/ou os resíduos a_t não sejam ruído branco, chamar a série de resíduos de N'_t, aplicar a modelagem de séries temporais ARIMA na série N'_t, iniciando por um modelo AR de baixa ordem, estimar os coeficientes da regressão e minimizar o erro a_t. Caso os resíduos a_t sejam ruído branco, o modelo é aceito. Caso contrário, examina-se a função de autocorrelação (fac) e a função de autocorrelação parcial (facp) dos resíduos para tentar novamente com outro modelo ARIMA, até obter ruído branco como resíduo. Este passo foi automatizado pelo uso do pacote *forecast*

(HYNDMAN e KHANDAKAR, 2008), do software R (R CORE TEAM, 2015).

Durante os itens a, b, c e d, foram testadas as influências das seguintes variáveis exógenas: temperatura do ar (atual e até 6 meses de defasagem), nível do reservatório, temperatura da água que passa pelo piezômetro, e nível de precipitação - sendo que somente a temperatura do ar com defasagem de 2 a 4 meses e a temperatura atual da água que passa pelo piezômetro se mostraram estatisticamente significativas.

Passo 3: a CNSSAW iterativa é usada na modelagem da supressão a fim de se capturar estruturas de autodependência não linear, considerando pré-processamento nos dados via filtragens SSA e decomposição Wavelet. O previsor automático CNSSAW é baseado em um laço computacional de técnicas numéricas e é operacionalizado de acordo com as etapas a, b, c e d, abaixo:

a) $y_t\ (t = 1, \dots, T)$ é filtrado via método SSA, descrito na seção 2.2.1, de forma a gerar $y'_t\ (t = 1, \dots, T)$ - que é uma versão menos ruidosa daquela;
b) uma decomposição Wavelet de nível r (conforme definido na seção 2.2.2), definido em termos de uma base ortogonal Wavelet (por exemplo, Haar, Daubechies, Minimum-Bandwidth, Fejér-Korovkin, Battle-Lemarie, e Symlet (Mallat, 2009; Daubechies, 1992; and Morris & Peravali, 1999) de $y'_t\ (t = 1, \dots, T)$ é realizada, produzindo uma CW de aproximação de nível m', denotada por $y_{A_{m'},t}\ (t = 1, \dots, T)$, onde $m' \in \mathbb{Z}$, e r WCs de detalhes nos níveis $m',\ m'+1,\ \dots,\ m'+(r-1)$, representadas por $y_{D_m,t}\ (t = 1, \dots, T)$, respectivamente, onde $r \in \mathbb{Z}$.

c) Cada WC do Item b é modelado, individualmente, por k diferentes RNAXs (como as definidas na seção 2.2.4), onde $k > 5$. Com efeito, são geradas as seguintes sequências de previsões: $\hat{y}_{A_{m'},t,i}$, $\hat{y}_{D_{m'},t,i}$, $\dots, \hat{y}_{D_{m'+(r-1)},t,i}\ (t = t'+1, \dots, T+h)$, onde: $i = 1, \dots, k$; e t' representa o número de graus de liberdade perdidos até esta etapa. Note que $\hat{y}_{A_{m'},t,i}$ e $y_{D_m,t,i}$, onde, $m = m', \dots, m'+(r-1)$, consistem, respectivamente, das

previsões do estado $y_{A_m,t}$, as quais são produzidas pela *i-ésima* RNA, e do estado $y_{D_m,t}$, gerada pela *i-ésima* RNA. Deste modo, uma vez que seja considerada uma Wavelet de decomposição de nível r de $y_t\ (t = 1, \dots, T)$, $(r + 1) \times k$ RNAs distintas são necessárias, ao todo, para a modelagem das CWs.

d) as 5 melhores previsões de cada WC são combinadas não linearmente por meio de outra RNA (como as da seção 2.2.4), genericamente indicada por $f(.)$, a fim de produzir as suas previsões combinadas dotadas de informações não lineares. Do ponto de vista matemático, tem-se que:

$$f_{A_m'}\left(\hat{y}_{A_m',t,1}, \hat{y}_{A_m',t,2}, \dots, \hat{y}_{A_m',t,5}, \hat{w}\right) = \hat{y}_{A_m',t,C}; \text{ e}$$

$$f_{D_m}\left(\hat{y}_{D_m,t,1}, \hat{y}_{D_m,t,2}, \dots, \hat{y}_{D_m,t,5}, \hat{w}\right) = \hat{y}_{D_m,t,C}.$$

Em que $\hat{y}_{A_m',t,C}$ e $\hat{y}_{D_m,t,C}$ são, respectivamente, a combinação de previsões de $y_{A_m,t}$ e $y_{D_m,t}$. De fato, as RNAs $f_{A_m'}(.)$ e $f_{D_m}(.)$ fornecem as predições $\hat{y}_{A_m',t,C}$ e $\hat{y}_{D_m,t,C}$, respectivamente.

e) as previsões combinadas das CWs do item b são, simplesmente, somadas, gerando-se as previsões de $N_t(t = 1, .., T)$, denotadas por $\hat{N}_t\ (t = t'' + 1, \dots, T + h)$, onde t'' representa os graus de liberdade perdidos até aqui. Com efeito, tem-se que:

$$\hat{N}_t = \hat{y}_{A_m',t,C} + \sum_{m=m'}^{m'+(r-1)} \hat{y}_{D_m,t,C}.$$

Quanto aos itens acima, destaca-se que eles são executados, em conjunto e de forma iterativa, por meio de um algoritmo computacional (esquematizado na Figura 12), o qual testa vários valores para os parâmetros da CNSSAW. Como objetivo, utiliza-se a minimização da estatística Erro Quadrático Médio (MSE – *Mean Square Error*) da amostra de validação. Mais especificamente, os parâmetros da CNSSAW iterativa são dados por: tamanho da janela SSA, no item a; nível de

decomposição Wavelet (*r*) e a base ortogonal Wavelet, no item b; e o comprimento da janela (*j*), o número de neurônios artificiais na camada escondida (*n*) e o número de iterações para treinamento (*i*) de cada ANN, nos itens c e d. Observa-se que, nos itens c e d, outros parâmetros das RNAs (como funções de ativação, normalização e algoritmo de treinamento) também poderiam ser usados como "variáveis de decisão" a serem otimizadas. No entanto, a fim de evitar períodos muito longos de ajuste numérico, apenas o comprimento da janela, o número de nós da camada oculta e a quantidade de iterações de treinamento foram escolhidos para se trabalhar como variáveis de decisão a serem ajustados numericamente.

Alguns aspectos no algoritmo computacional foram contabilizados e também merecem algum detalhamento.

Em primeiro lugar, tem-se que, uma vez que pode ser realizado um número enorme de diferentes configurações entre um nível de Wavelet de decomposição *r* e uma base ortonormal Wavelet, e que não há uma solução analítica para se determinar qual é a melhor combinação, segue que é necessário um grande número de testes empíricos, dentro de uma faixa previamente definida, para a obtenção da melhor.

Em segundo lugar, quanto aos parâmetros da RNA, não existe regra sistemática a fim de se decidir o seu valor ótimo, de tal modo que todos os valores pertencentes a uma determinada gama, também previamente definida na programação, são numericamente testados.

Em terceiro lugar, a escolha de um número apropriado de nós escondidos *n* e a seleção do número de observações desfasadas, *j* (a dimensão do vector de entrada), definem outra importante tarefa na modelagem RNA. O valor do parâmetro *j* é, talvez, o parâmetro mais importante a ser avaliado num modelo de RNA, visto que desempenha um papel direto na determinação da estrutura ótima de autodependência não linear dos dados de subpressão a serem modelados. No entanto, também não existe uma teoria a partir da qual se pode calculá-lo analiticamente, de maneira que a seleção de um valor ótimo para *j* ocorre iterativamente.

Em quarto lugar, devido ao efeito “*overfitting*”, comumente encontrados em modelagens por RNAs, uma amostra de validação é utilizada para se escolher as

RNAs com maior poder de generalização fora da amostra (no caso, na amostra de teste). O termo "*overfitted*" significa que uma RNA possui bom ajuste para os dados na amostra de treinamento e de validação, mas tem baixa capacidade de generalização no período fora da amostra de treinamento (*out-of-sample* ou amostra de teste).

Em quinto lugar, para cada configuração de parâmetros de rede neural testado, uma previsão é gerada e armazenada para serem escolhidas as 5 melhores previsões para serem combinadas formando a previsão final para cada componente Wavelet. Entretanto, considerando um maior rigor estatístico, somente são consideradas válidas para serem combinadas as previsões que produzirem um resíduo considerado ruído branco, sobre a amostra de treinamento. Para isso é necessário testar se os resíduos são independentes e identicamente distribuídos (i.i.d), e com média zero, o que é feito aplicando o teste BDS (BROCK, DECHERT e SCHEINKMAN, 1987), que tem como hipótese nula (H_0) a classificação do resíduo como i.i.d. e hipótese alternativa (H_1) a presença de dependência linear ou não linear nos resíduos.

A Figura 12 ilustra o fluxograma da previsão, iniciando pela filtragem da série temporal de subpressão usando SSA, conforme item a do passo 3. O número mínimo e máximo de componentes SSA (l) é definido de acordo com o tamanho da série temporal informada N, seguindo a fórmula analítica $(\ln \mathrm{N})^{1,5} \leq \mathrm{l} \leq (\ln \mathrm{N})^{3}$ (KHAN e POSKITT, 2013).

Para cada um dos valores de l, é realizada uma decomposição SSA, gerando l componentes, que são agrupadas em 4 grupos exclusivos, por meio de clusterização hierárquica, sendo os três primeiros grupos reagrupados para recompor a série histórica com a redução de ruídos, e o último grupo (ruídos) é desprezado. A série filtrada é, então, usada para chamar a rotina que faz a previsão da série de tempo filtrada usando a decomposição Wavelet e as RNAs. Tal operacionalização é ilustrada na Figura 13.

Na Figura 13, é mostrado o fluxograma da decomposição Wavelet, descrito item b, incluindo os laços para se procurar os melhores parâmetros de decomposição (a saber, nível de decomposição e base ortonormal) e a chamada para a rotina da RNA que retorna as previsões.

Nesta parte, os parâmetros de entrada utilizados são: *série* (série temporal de subpressão), *t.ar* (série temporal de temperatura do ar), *t.agua* (série temporal de temperatura da água), *ji* e *jf* (janela inicial e janela final, os limites para o tamanho da janela da rede neural), *ni* e *nf* (número inicial e final de neurônios na camada escondida) e *ii* e *if* (número máximo de iterações inicial e final para treinamento de rede neural). Há ainda as variáveis internas: *f* (índice que indica qual filtro – ou base ortonormal Wavelet – está sendo usado), *r* (nível de decomposição Wavelet, testado e 1 a 3), filtros (um vetor com as identificações de todos os filtros Wavelet suportados), *Apr r* é uma componente Wavelet (CW) de aproximação de nível *r*, *Det 1, ..., Det r* (CW de detalhe de nível *1* até *r*), *previsão APR r, previsão Det 1, ..., previsão Det r* (são as previsões para as respectivas CW de aproximação e de detalhe), *previsãoComp[f,r]* (matriz que armazena as previsões das somas das CW de aproximação e de detalhe para cada filtro *f* e nível de decomposição *r*), numFiltros (indica a quantidade de filtros suportados) e Previsão (contém a melhor previsão encontrada).

O comando RNA (.) indica uma chamada ao procedimento que faz a previsão de uma série temporal usando Redes Neurais Artificiais (RNAX), ilustrado na Figura 14. Cabe ressaltar que esse procedimento é chamado levando todos os parâmetros recebidos, o que foi omitido para maior clareza visual.

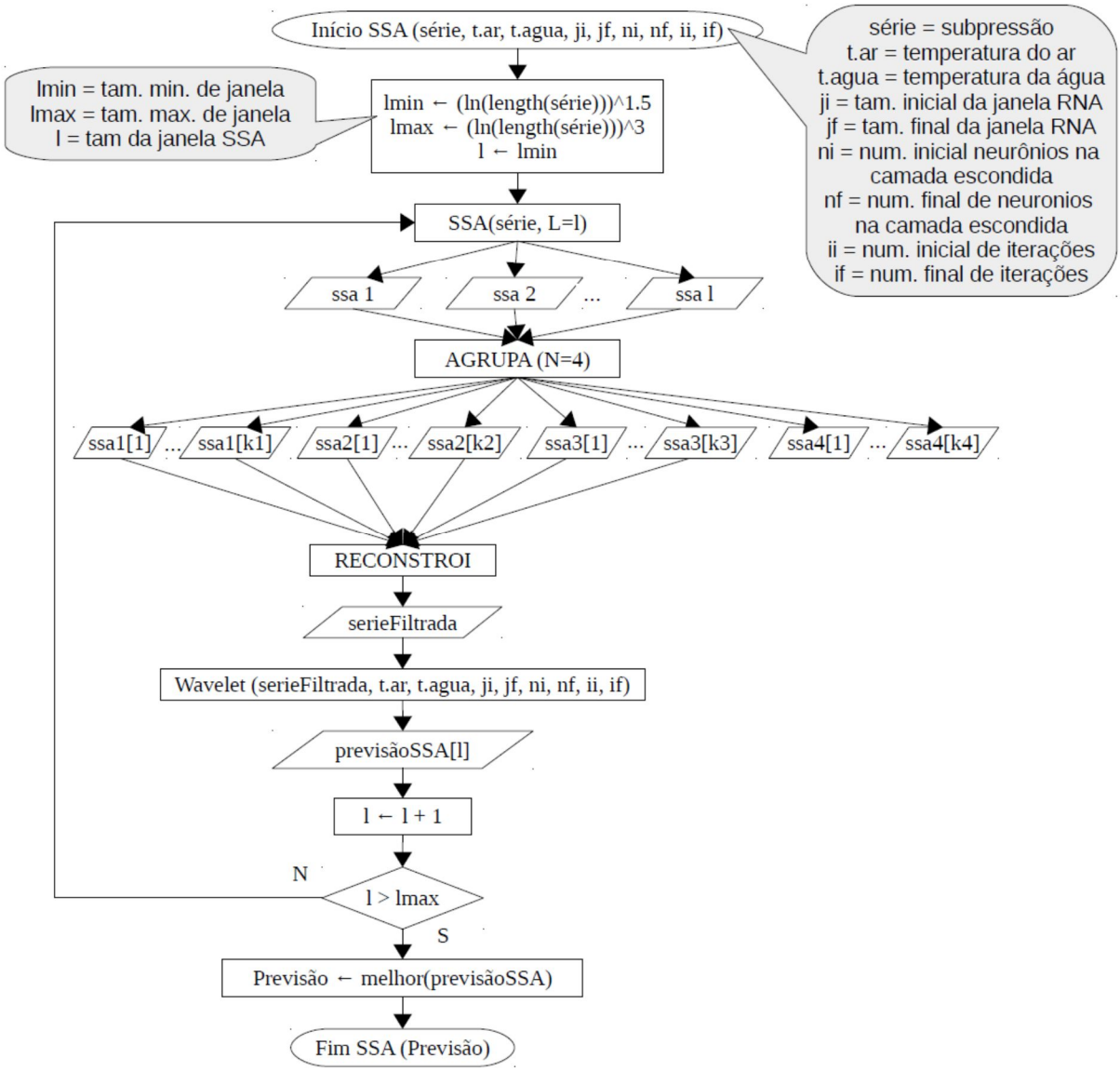


FIGURA 12 – PROCEDIMENTO DE FILTRAGEM DE RUÍDOS POR SSA.

FONTE: O autor

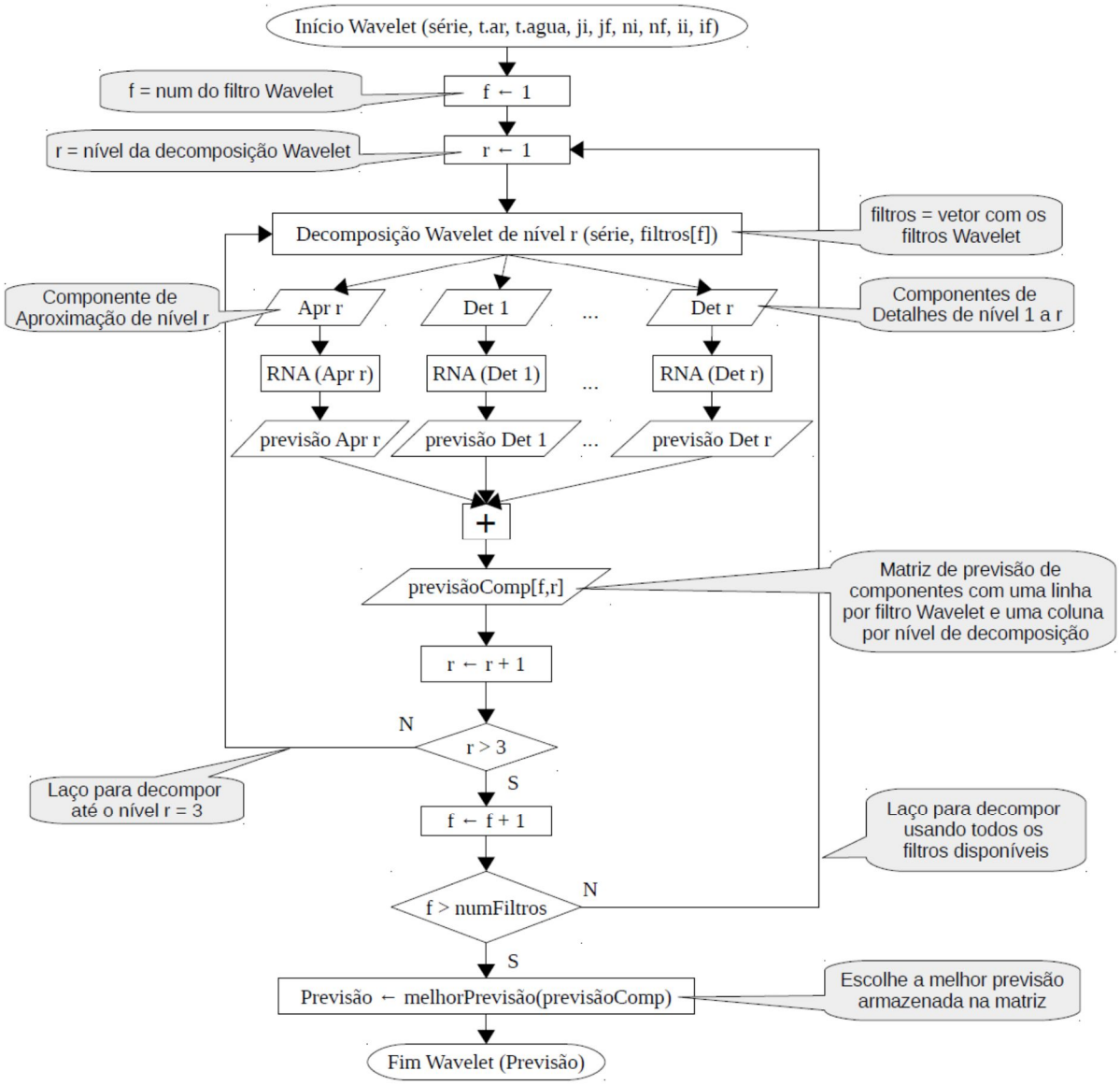


FIGURA 13 – PROCEDIMENTO DE DECOMPOSIÇÃO WAVELET.

FONTE: O autor

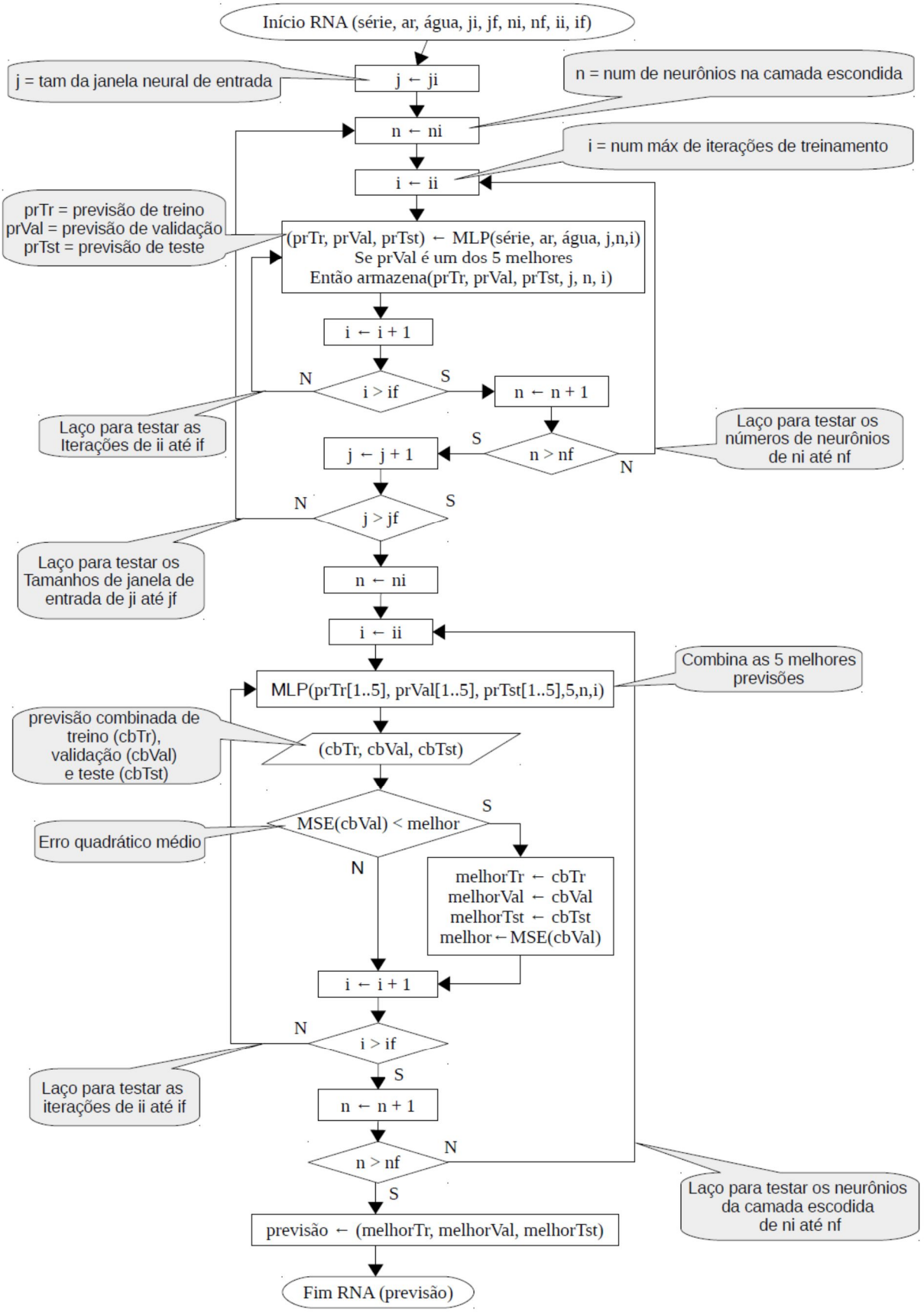


FIGURA 14 – PROCEDIMENTO DE PREVISÃO RNAX.

FONTE: O autor

A Figura 14 exibe a estrutura do procedimento RNAX, o qual recebe os seguintes parâmetros: *série* (a CW a ser prevista), *ar* (a série temporal da temperatura do ar), *ag* (a série temporal da temperatura da água), *ji* e *jf* (janela inicial e final), *ni* e *nf* (número inicial e final de neurônios na camada escondida) e *ii* e *if* (número inicial e final de iterações para treinamento da rede neural).

Internamente, as variáveis usadas são *j*, *n* e *i* (índices para janela, número de neurônios e iterações), *prTr*, *prVal* e *prTst* (previsões de treinamento, validação e teste), *prTr[1..5]*, *prVal[1..5]* e *prTst[1..5]* (as 5 melhores previsões de treinamento, validação e teste), *cbTr*, *cbVal* e *cbTst* (o resultado das combinações das melhores previsões de treinamento, validação e teste), *melhor* (o menor erro quadrático médio – MSE – para as previsões de validação combinadas), *melhorTr*, *melhorVal* e *melhorTst* (melhores previsões combinadas das amostras de treinamento, validação e teste, respectivamente), e, finalmente, *previsão* (a melhor previsão encontrada, incluindo treinamento, validação e teste).

O procedimento MLP cria uma rede neural Multilayer Perceptron com os parâmetros informados, e retorna um vetor contendo o ajuste para a amostra (treino e validação) e a previsão para fora da amostra (teste). O procedimento MSE(cbVal) calcula o erro quadrático médio da amostra de validação obtida pelo modelo, em relação à amostra de validação da série original.

Basicamente, o algoritmo se inicia pela filtragem SSA com o objetivo de redução de ruídos, ilustrado na Figura 12. Para cada um dos valores de tamanho de janela *l*, a série temporal de subpressão é decomposta em *l* componentes, que depois são agrupadas em 4 grupos por clusterização hierárquica, em ordem decrescente de contribuição para a composição da série.

Em seguida, os três primeiros grupos são usados para a reconstrução da série, com componente estocástica com menor magnitude. Cada série reconstruída é modelada pelos procedimentos descritos nas Figuras 13 e 14. O procedimento (a) faz decomposições Wavelet para todas as combinações possíveis dos filtros disponíveis e níveis de decomposição de 1 a 3, usando dois laços aninhados. Para cada CW ele chama o procedimento RNA ilustrado na Figura 14. Este procedimento RNAX realiza o teste das combinações possíveis de tamanho da janela, número de neurônios na camada escondida e número de iterações, fazendo previsões RNA considerando as variáveis causais. Em seguida, ele seleciona os cinco melhores e

combinando-os usando outras RNAs, variando os parâmetros de tamanho da janela e número de iterações em mais dois laços aninhados, e finalmente seleciona a melhor combinação com base no resultado do MSE da amostra de validação.

Esta combinação é então devolvida como a previsão CW. Todas as previsões CW são somadas para formar a previsão de subpressão por um filtro e um nível de decomposição. Então, a melhor previsão de subpressão é selecionada como a previsão Wavelet. Finalmente, é escolhida a melhor previsão Wavelet entre todas as séries filtradas com diferentes tamanhos de janela SSA, e é retornada como a previsão não linear final $\widehat{\mathrm{N}}_{\mathrm{t}}$.

Passo 4: As previsões lineares e não lineares geradas – representadas, respectivamente por $\hat{\mathrm{L}}_{\mathrm{t}}$ e $\widehat{\mathrm{N}}_{\mathrm{t}}$ $(\mathrm{t} = \mathrm{t}'' + 1, \dots, \mathrm{T} + \mathrm{h})$ – são combinadas, de forma linear -, em (3.1), gerando-se previsões híbridas, denotadas por $\hat{\mathrm{y}}_{\mathrm{t}}$ $(\mathrm{t} = \mathrm{t}'' + 1, \dots, \mathrm{T} + \mathrm{h})$, agregadoras de informações da estrutura de autodependência linear e não linear da série temporal de subpressão subjacente.

$$\hat{\mathrm{y}}_{\mathrm{t}} = \frac{\hat{\mathrm{L}}_{\mathrm{t}} + \widehat{\mathrm{N}}_{\mathrm{t}}}{2} \tag{3.1}$$

Passo 5: Para se gerar os intervalos de previsão, é realizado o procedimento de Bootstrap associado a cada previsor – isto é, ARIMAX e CNSSAW iterativa –, conforme descrito na seção 2.2.6. Cada i-ésimo cenário, em cada instante é combinado conforme a equação (3.2). Isto é,

$$\hat{\mathrm{y}}^{(\mathrm{i})}{}_{\mathrm{t}} = \frac{\hat{\mathrm{L}}^{(\mathrm{i})}{}_{\mathrm{t}} + \widehat{\mathrm{N}}^{(\mathrm{i})}{}_{\mathrm{t}}}{2} \tag{3.2}$$

onde: $\hat{\mathrm{L}}^{(\mathrm{i})}{}_{\mathrm{t}}$ e $\widehat{\mathrm{N}}^{(\mathrm{i})}{}_{\mathrm{t}}$, são, no instante t, os i-ésimos cenários gerados da integração ARIMAX-Boostrap e CNSSAW-Bootstrap, respectivamente, seguindo a abordagem de amostragem descrita na seção 2.2.6.

4 RESULTADOS E DISCUSSÕES

Neste capítulo são apresentados os principais resultados da aplicação do método proposto (descrito no capítulo 3) para a previsão de séries temporais de subpressão. Para tanto, se utilizaram dados históricos de três piezômetros na feição contato concreto-rocha, no Trecho F, da usina hidrelétrica de Itaipu. Foram consideradas as médias mensais dos instrumentos aferidos semanalmente por técnicos de campo. Como condições de contorno, foram consideradas as médias mensais dos dados de temperatura ambiente, temperatura da água e nível do reservatório.

4.1 Modelagem das três séries temporais mensais de piezômetros de Itaipu

Nas Figuras 15, 16 e 17, a seguir, pode-se observar as previsões (ou ajustes) do método híbrido proposto, na amostra de treinamento (*in-sample*), cujo gráfico no plano cartesiano é dado pela linha vermelha pontilhada, para as leituras observadas (curva azul contínua) aferidas, respectivamente nos piezômetros PS-F-73, PS-F-74 e PS-F-8.

Nota-se que, nos três casos, as previsões pontuais mensais exibem forte correlação com os dados de subpressão históricos, uma vez que, visualmente, as curvas vermelha e azul se encontram muito próximas, em praticamente todos os instantes. Isso mostra que, de fato, as previsões híbridas foram altamente precisas e que o método híbrido proposto pôde explicar a maior parte da variabilidade exibida pelos dados históricos mensais de subpressão. Especificamente, as correlações foram: 95,15% para o PS-F-74; 97,07% para o PS-F-73; e 95,98% para o PS-F-8.

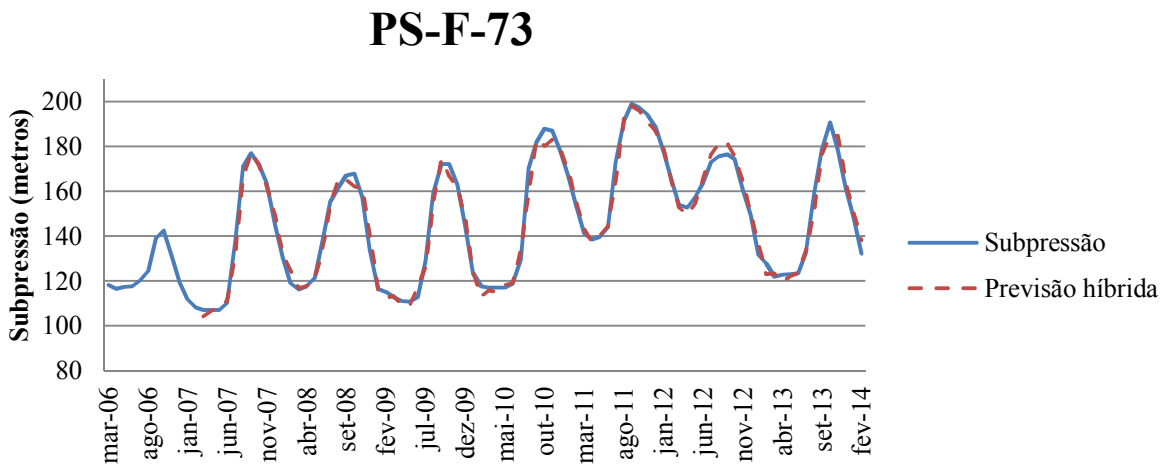


FIGURA 15 – AJUSTE DO MÉTODO PROPOSTO, NA AMOSTRA DE TREINO, ÀS LEITURAS DO PIEZÔMETRO PS-F-73

FONTE: O autor.

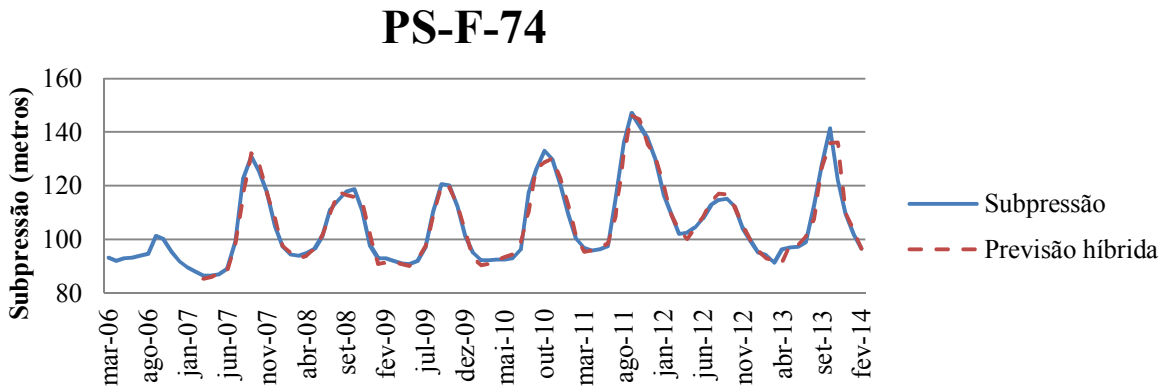


FIGURA 16 – AJUSTE DO MÉTODO PROPOSTO, NA AMOSTRA DE TREINO, ÀS LEITURAS DO PIEZÔMETRO PS-F-74

FONTE: O autor.

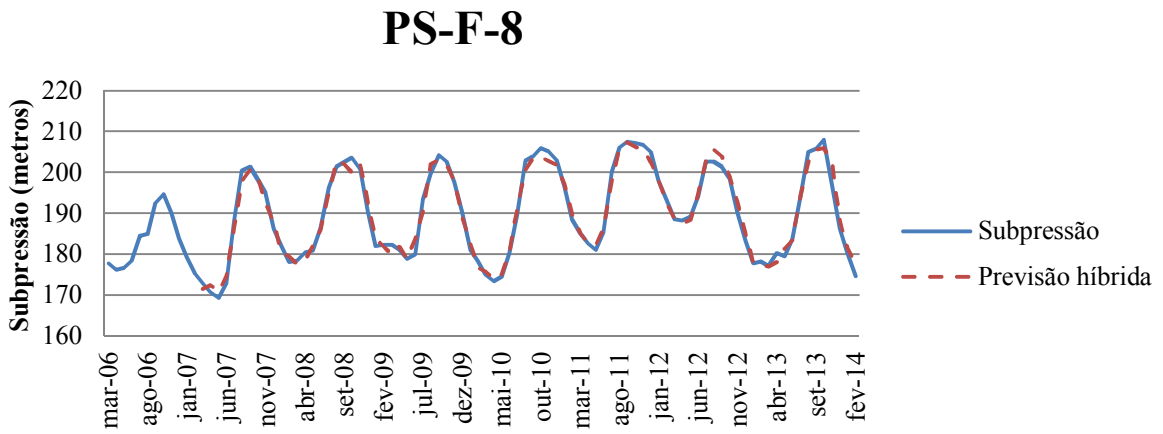


FIGURA 17 – AJUSTE DO MÉTODO PROPOSTO, NA AMOSTRA DE TREINO, ÀS LEITURAS DO PIEZÔMETRO PS-F-8.

FONTE: O autor.

Nas Figuras 18, 19 e 20, pode-se observar as previsões pontuais, bem como os intervalos de previsão *Bootstrap* a elas associadas, na amostra de teste (*out-of-sample*) dos piezômetros PS-F-73, PS-F-74 e PS-F-8, respectivamente, em um horizonte de previsão igual a 12 meses.

No que concerne às previsões intervalares associadas às previsões híbridas na amostra de teste, optou-se pelo usual nível de confiança de 95%. Pode-se notar que, nos piezômetros PS-F-73 e PS-F-74, não ocorreu violação alguma nos intervalos preditivos; enquanto que, no piezômetro PS-F-8, houve apenas uma violação (ou seja, um alerta amarelo). Do total de 36 previsões, sendo 12 para cada instrumento, houve apenas 1 leitura fora do intervalo preditivo, o que está coerente com o nível de confiança de 95%. Além disso, é importante salientar que as amplitudes dos intervalos de previsão não tomaram valores exorbitantes, o que significa que a incerteza acerca do futuro estava dentro do aceitável, do ponto de vista prático.

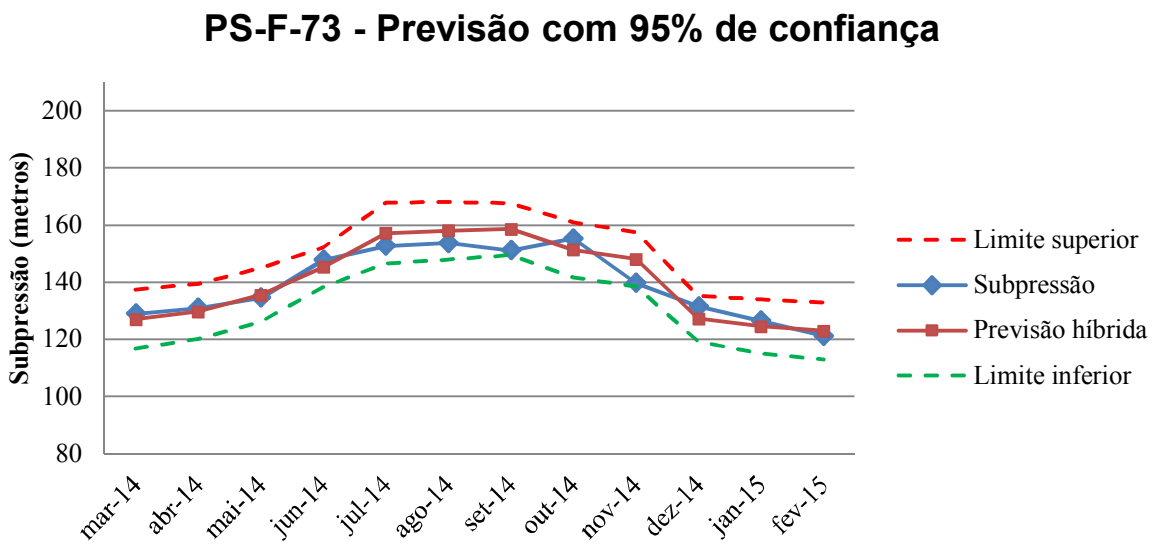


FIGURA 18 – PREDIÇÕES MENSAIS PARA AS LEITURAS FUTURAS DO PIEZÔMETRO PS-F-73
FONTE: O autor.

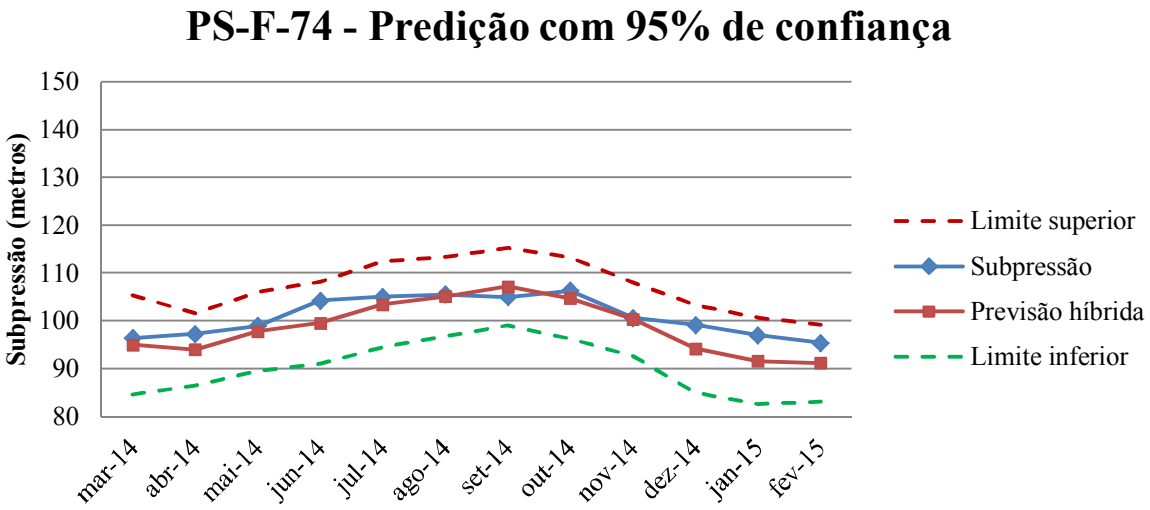


FIGURA 19 – PREDIÇÕES MENSAIS PARA AS LEITURAS FUTURAS DO PIEZÔMETRO PS-F-74
FONTE: O autor.

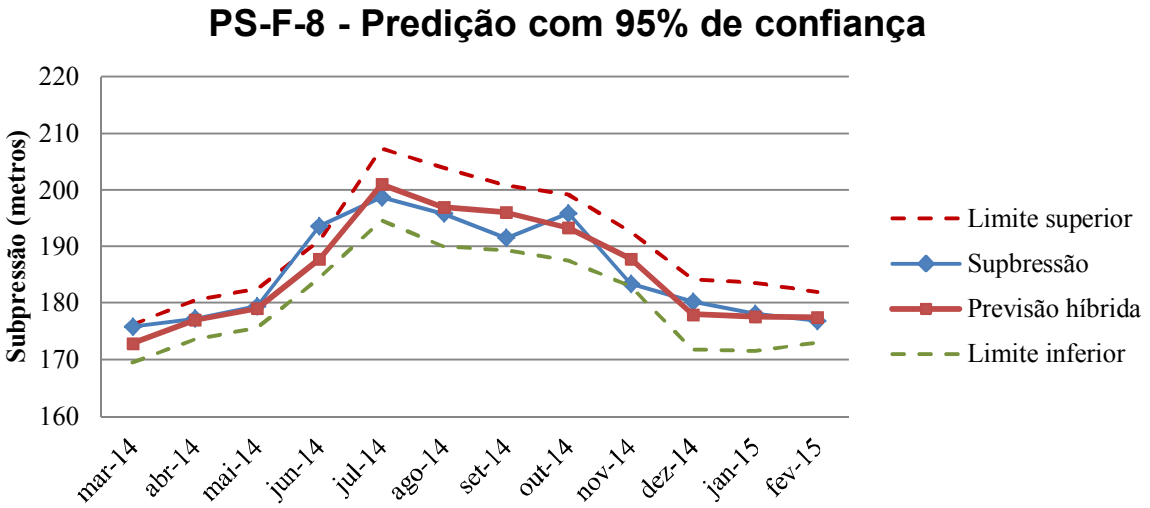


FIGURA 20 – PREDIÇÕES MENSAIS PARA AS LEITURAS FUTURAS DO PIEZÔMETRO PS-F-8
FONTE: O autor.

As Tabelas 1, 2 e 3 apresentam os dados referentes aos gráficos das Figuras 18, 19 e 20, respectivamente, incluindo as leituras, a previsão pontual e os limites inferior e superior do intervalo de previsão, associados aos piezômetros PS-F-73, PS-F-74 e PS-F-8.

TABELA 1 – PREDIÇÕES MENSAIS PARA AS LEITURAS FUTURAS DO PIEZÔMETRO PS-F-73

Mês	Leituras	Previsão	Lim. Inferior	Lim. Superior
Mar-2014	129,00	127,11	116,78	137,45
Abr-2014	131,00	129,84	120,13	139,55
Mai-2014	134,67	135,67	126,23	145,11
Jun-2014	147,80	145,32	138,32	152,33
Jul-2014	152,75	157,28	146,57	167,98
Ago-2014	153,75	158,16	148,12	168,19
Set-2014	151,25	158,68	149,51	167,84
Out-2014	155,40	151,42	141,74	161,09
Nov-2014	139,75	148,20	138,70	157,69
Dez-2014	131,60	127,29	119,33	135,25
Jan-2015	126,50	124,64	115,10	134,18
Fev-2015	121,33	123,10	113,13	133,06

FONTE: O autor.

TABELA 2 – PREDIÇÕES MENSAIS PARA AS LEITURAS FUTURAS DO PIEZÔMETRO PS-F-74

Mês	Leituras	Previsão	Lim. Inferior	Lim. Superior
Mar-2014	96,40	95,01	84,65	105,38
Abr-2014	97,27	94,04	86,42	101,66
Mai-2014	99,00	97,83	89,61	106,04
Jun-2014	104,27	99,60	91,05	108,15
Jul-2014	105,10	103,49	94,47	112,51
Ago-2014	105,57	105,13	96,85	113,41
Set-2014	105,02	107,19	99,08	115,29
Out-2014	106,37	104,74	96,20	113,28
Nov-2014	100,67	100,37	92,68	108,06
Dez-2014	99,17	94,18	84,98	103,39
Jan-2015	97,04	91,63	82,56	100,69
Fev-2015	95,40	91,17	83,16	99,18

FONTE: O autor.

TABELA 3 – PREDIÇÕES MENSAIS PARA AS LEITURAS FUTURAS DO PIEZÔMETRO PS-F-8

Mês	Leituras	Previsão	Lim. Inferior	Lim. Superior
Mar-2014	175,81	172,87	169,57	176,16
Abr-2014	177,28	177,03	173,59	180,47
Mai-2014	179,45	179,07	175,60	182,55
Jun-2014	**193,60**	187,79	184,49	191,09
Jul-2014	198,73	201,04	194,69	207,38
Ago-2014	195,85	196,98	190,03	203,93
Set-2014	191,50	196,05	189,30	200,81
Out-2014	195,88	193,33	187,48	199,18
Nov-2014	183,40	187,86	183,05	192,67
Dez-2014	180,24	178,01	171,80	184,22
Jan-2015	178,10	177,57	171,59	183,54
Fev-2015	176,85	177,46	172,98	181,95

FONTE: O autor.

Na Tabela 4, podem ser observados os valores das amplitudes de cada instante na amostra de teste, para os três casos. Em negrito, observa-se o melhor caso e o pior caso - a saber, 6,590m (no PS-F-8) e 21,402m (no PS-F-73),

respectivamente - de amplitudes dos intervalos de previsão aqui considerados. Em outras palavras, o erro de estimação, no melhor caso, foi de 3,295m (ou seja, igual a 1,88% do valor da previsão); enquanto que, no pior caso, foi de 10,701m (isto é, igual a 7% do valor da previsão).

Dado que as séries históricas mensais de subpressão de Itaipu apresentam, em regra, mudanças de regime em sua flutuação estocástica, um erro percentual máximo igual a 7%, em relação a um valor da predição, pode ser classificado como baixo. Isso corrobora a afirmativa de que os intervalos de previsão tomaram valores de amplitudes dentro de uma faixa considerada aceitável, do ponto de vista prático-operacional, de maneira a ser significativamente informativos.

TABELA 4 – AMPLITUDES DOS INTERVALOS DE PREVISÃO E AS RESPECTIVAS LEITURAS FUTURAS DOS TRÊS PIEZÔMETROS, NA AMOSTRA DE TESTE.

	Amplitudes			**Leituras**		
Mês	PS-F-8	PS-F-73	PS-F-74	PS-F-8	PS-F-73	PS-F-74
Mar-2014	**6,590**	20,662	20,736	175,81	129,00	96,40
Abr-2014	6,881	19,414	15,241	177,28	131,00	97,27
Mai-2014	6,958	18,874	16,425	179,45	134,67	99,00
Jun-2014	6,598	14,018	17,100	**193,60**	147,80	104,27
Jul-2014	12,685	**21,402**	18,035	198,73	152,75	105,10
Ago-2014	13,903	20,076	16,558	195,85	153,75	105,57
Set-2014	11,503	18,335	16,217	191,50	151,25	105,02
Out-2014	11,703	19,349	17,083	195,88	155,40	106,37
Nov-2014	9,618	18,990	15,376	183,40	139,75	100,67
Dez-2014	12,420	15,929	18,408	180,24	131,60	99,17
Jan-2015	11,953	19,077	18,129	178,10	126,50	97,04
Fev-2015	8,976	19,935	16,012	176,85	121,33	95,40

FONTE: O autor.

Outro ponto, como mencionado anteriormente, é o fato de que somente a leitura no mês de jun-2014, na amostra de teste, do PS-F-8 assumiu valor fora do intervalo de previsão, a um nível 95% de confiança, em que seus limites inferior e superior foram, respectivamente, iguais a 184,49m e 191,09m. Percebe-se que a diferença entre o limite superior (191,09m) e o valor aferido (193,60m) foi de 2,51m; ou seja, 1,31% acima do limite superior, de modo que tal violação poderia, do ponto de vista estatístico-numérico, ser classificada como “desprezível”.

No que se refere às previsões (pontuais) *out-of-sample*, alguns pontos merecem ser destacados, a fim de se evidenciar, com um maior refinamento, o alto nível de acurácia e de poder de generalização que o método híbrido proposto alcançou nos três casos supracitados. Nesta perspectiva, tem-se que, em um processo de previsão, um dos aspectos desejáveis é que os erros *out-of-sample*

tomem valores dentro de uma faixa de regularidade; ou seja, que não oscilem tanto a ponto de tomarem valores aberrantes, comprometendo o processo de tomada de decisão. Para este fim, a estatística APE (*Absolute Percentage Error*) foi utilizada e os resultados obtidos, para os três piezômetros em questão, podem ser verificados na Tabela 5.

TABELA 5 – VALORES DE APE RELATIVAS AOS TRÊS PIEZÔMETROS, NA AMOSTRA DE TESTE

	APE		
MÊS	*PS-F-8*	*PS-F-73*	*PS-F-74*
Mar-2014	1,67%	1,46%	1,44%
Abr-2014	**0,14%**	0,89%	3,32%
Mai-2014	0,21%	0,74%	1,19%
Jun-2014	3,00%	1,67%	4,48%
Jul-2014	1,16%	2,96%	1,53%
Ago-2014	0,58%	2,87%	0,41%
Set-2014	2,38%	4,91%	2,06%
Out-2014	1,30%	2,56%	1,53%
Nov-2014	2,43%	**6,04%**	0,30%
Dez-2014	1,24%	3,28%	5,03%
Jan-2015	0,30%	1,47%	5,58%
Fev-2015	0,35%	1,46%	4,43%

FONTE: O autor.

Pode-se notar que o pior e o melhor caso são iguais, respectivamente, a 6,04% (no PS-F-73) e a 0,14% (no PS-F-8). Quanto ao desvio-padrão - que consiste em uma medida de espalhamento “padrão” em torno do valor da estatística MAPE (*Mean Abosolute Percentage Error*), apresentada na Figura 21 -, os valores obtidos foram: 0,97% (no PS-F-8); 1,62% (no PS-F-73); e 1,86% (no PS-F-74). Ou seja, um afastamento padrão máximo, em torno do MAPE, igual a 1,86%, no caso do PS-F-74.

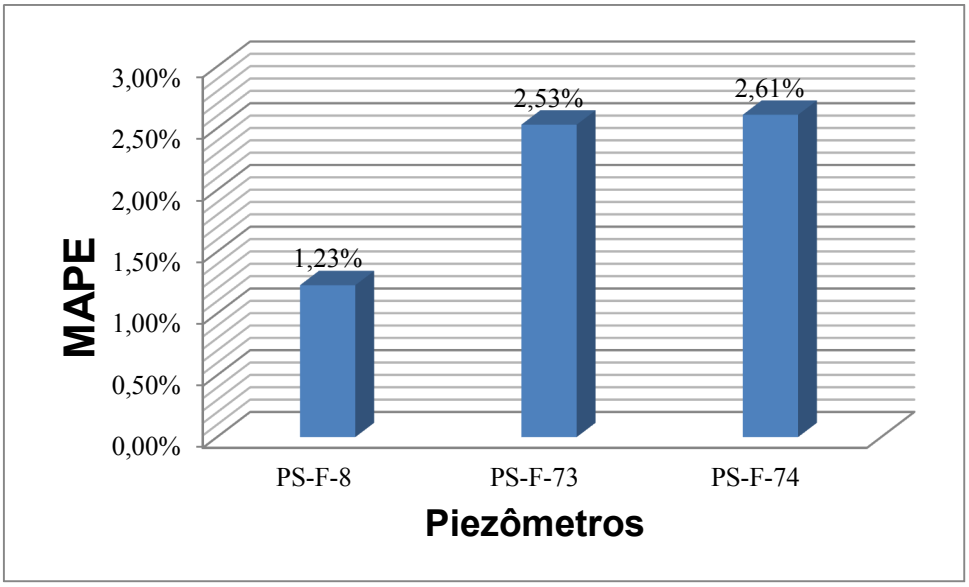


FIGURA 21 – VALORES DE MAPE DOS TRÊS PIEZÔMETROS, NA AMOSTRA DE TESTE
FONTE: O autor.

Em última análise, para efeito de comparação, os três piezômetros supracitados foram também modelados pelos seguintes previsores tradicionais: ARIMA (automático), ARIMAX e Rede Neural Artificial (RNA). Para tal, foi considerada a estatística RMSE (*Root Mean Square Error*), nas amostras de treinamento e de teste, apresentada na Tabela 6. O objetivo aqui foi o de verificar a capacidade de mapeamento de estruturas de autodependência nos dados históricos (na amostra de treino) e de generalização (na amostra de teste).

Pode-se observar que o método híbrido aqui proposto alcançou um melhor resultado (isto é, RMSE menor) significativamente superior a todos os quatro métodos preditivos usados como benchmark, na modelagem dos três piezômetros. Um melhor detalhamento, em termos de *ganhos relativos* (que são calculados pela diferença entre os valores de RMSE dos métodos proposto e benchmark dividido pelo valor de RMSE do método proposto), pode ser verificado na Tabela 7.

TABELA 6 – ESTATÍSTICA RMSE PARA AS PREVISÕES RELATIVAS AOS PIEZÔMETROS PS-F-8, PS-F-73 E PS-F-74, NAS AMOSTRAS DE TREINO E DE TESTE

	RMSE					
	PS-F-8		**PS-F-73**		**PS-F-74**	
Métodos	*Treino*	*Teste*	*Treino*	*Teste*	*Treino*	*Teste*
ARIMAX	3,0512	5,4764	6,3803	7,0471	4,4697	3,4860
ARIMA	4,5687	6,9867	8,5101	10,5390	6,5801	7,6699
RNA	5,2652	7,8732	6,5193	10,1781	6,6417	8,9728
MÉTODO PROPOSTO	**1,6988**	**2,9151**	**3,6098**	**4,2765**	**2,7415**	**3,1341**

FONTE: O autor.

TABELA 7 – GANHOS RELATIVOS EM RELAÇÃO À ESTATÍSTICA RMSE, NAS AMOSTRAS DE TREINO E DE TESTE, RELATIVAS AOS PIEZÔMETROS PS-F-8, PS-F-73 E PS-F-74

	PS-F-8		**PS-F-73**		**PS-F-74**	
Métodos	*Treino*	*Teste*	*Treino*	*Teste*	*Treino*	*Teste*
ARIMAX	79,60%	87,86%	76,74%	64,79%	**63,04%**	**11,23%**
ARIMA	168,94%	139,67%	135,75%	146,44%	140,02%	144,72%
RNA	**209,94%**	170,08%	80,60%	138,00%	142,27%	**186,30%**

FONTE: O autor.

Nota-se, na Tabela 7, que para a amostra de treino, o modelo ARIMAX, que é o melhor resultado entre os métodos benchmarking, na modelagem do piezômetro PS-F-74, obteve um valor de RMSE 63,04% superior ao RMSE do método híbrido proposto; por outro lado, o previsor RNA alcançou, no mapeamento do piezômetro PS-F-8, um valor de RMSE 209,94% superior ao tomado pelo previsor proposto.

Por sua vez, quanto à amostra de teste, o previsor ARIMAX obteve melhor resultado (a saber, RMSE 11,23% superior ao do método proposto), na modelagem do piezômetro PS-F-74; enquanto a RNA alcançou um valor de RMSE pior, também na predição do piezômetro PS-F-74, igual a 186,30% superior ao método proposto.

5 CONCLUSÃO

Nesta tese de doutorado, foi desenvolvido um método de previsão híbrido cujo objetivo é a produção de predições pontuais e intervalares (aqui interpretados como alertas amarelos) de leituras futuras aferidas em piezômetros em fundações de barragens de concreto. O referido método utiliza, de forma integrada, algumas das principais técnicas numéricas exibidas na literatura, a saber: método SSA; decomposição Wavelet; modelo ARIMAX, rede neural artificial; combinação de previsões; e amostrador *Bootstrap*. A fim de mostrá-la e ilustrá-la numericamente, assim como detalhá-la em um caso real de aplicação, foram produzidas previsões e intervalos de previsão, com 95% de certeza, a eles associados, para leituras futuras de três piezômetros da Usina Hidrelétrica de Itaipu, referidos aqui por PS-F-8, PS-F-73 e PS-F-74. Para efeito de comparação, os dados históricos supracitados foram também modelados através de três outros previsores consagrados, a saber: ARIMAX, ARIMA e RNA. Para a produção dos intervalos de previsão, com 95% de confiança, associados a estes, utilizou-se a técnica *Bootstrap*.

No passo 1, as condições de contorno (a saber, a temperatura ambiente e a temperatura da água no fundo do lago) foram modeladas, satisfatoriamente, do ponto de vista de acurácia, por meio de modelos ARIMA automáticos. No passo 2, por sua vez, o modelo ARIMAX automático também exibiu resultados satisfatórios, na maneira em que a complexidade inerente aos dados de subpressão era altíssima; desse modo, as estruturas de autodependência linear exibidas por estes foram corretamente filtradas.

No que tange ao método preditivo não linear CNSSAW do Passo 3, tem-se que, previamente à predição dos dados subjacentes, a abordagem SSA foi adotada no processo de filtragem dos três dados de supressão de Itaipu, gerando suas versões suavizadas (ou menos ruidosas). Como consequência, as componentes ortogonais Wavelet geradas a partir deles exibiram comportamento menos ruidoso. Em outras palavras, a combinação dos métodos SSA e de decomposição Wavelet permitiu eliminar ruídos, que tendem a prejudicar as previsões, e, ao mesmo tempo, capturar informações da frequência espectral, presente em sinais temporais. Os parâmetros ótimos das técnicas de pré-processamento SSA e Wavelet são calculados internamente dentro de um processo iterativo. O espectro de parâmetros

possui milhares (ou, dependendo da configuração, milhões) de possibilidades de solução.

No Passo 4, foram geradas as previsões híbridas do método proposto que integram informações lineares (do modelos ARIMA e ARIMAX, dos passos 1 e 2), e não lineares e da frequência espectral (do previsor CNSSAW, do passo 3). Por fim, no Passo 5, os valores de quantis simulados por *Bootstrap* foram construídos dotados de informações lineares e não lineares e, com efeito, os intervalos de previsão, construídos com 95% de credibilidade, se mostraram eficientes, visto que apresentaram apenas uma violação (Figura 20, a saber).

Dados os resultados nas figuras e nas tabelas no capítulo 4, verifica-se claramente que o método híbrido proposto alcançou desempenho preditivo substancialmente superior às técnicas benchmarking, em todas as análises. Isso leva a concluir que se trata de uma alternativa pertinente e eficiente à predição de dados de subpressão em barragens de concreto.

Entre as inovações trazidas por esta tese, cabe destacar:

(i) a aplicação de métodos estatísticos e de aprendizado de máquina para o contexto de previsão de séries temporais de subpressão em fundações de barragens de concreto;

(ii) a busca automática dos melhores parâmetros para a modelagem, tanto de SSA, quanto de Wavelets e Redes Neurais Artificiais;

(iii) a inclusão de variáveis causais na previsão de séries temporais por Redes Neurais Artificiais, aqui referenciadas por RNAX;

(iv) a aplicação do teste estatístico BDS para selecionar somente os modelos de previsão de redes neurais RNAX que apresentem como resíduo apenas ruído branco;

(v) a aplicação de um conjunto de Redes Neurais Artificiais para realizar a combinação não linear das melhores previsões obtidas, auferindo um ganho preditivo; e

(vi) a aplicação da técnica de agrupamento hierárquico automático na aplicação da técnica de SSA para a redução da magnitude da componente estocástica da série temporal a ser prevista.

TRABALHOS FUTUROS

Dados os resultados e conclusões destacadas ao longo desta tese, tem-se que é factível a execução de outras pesquisas referentes ao desenvolvimento de métodos de previsão, decorrentes ou similares ao desta tese, cuja finalidade é, cada vez mais, gerar eficiência preditiva, na modelagem de séries temporais de piezômetros, bem como de outros instrumentos em instalados barragens, em geral (não só as de estruturas de concreto). Nesta da gama de alternativas, eis alguns exemplos.

É possível se combinar, de forma não linear e linear, predições para as componentes Wavelet suavizadas via RNA' s recorrentes, a exemplo das redes de Ellman (HAYKIN, 2001). É factível o teste de diferentes padrões de entrada, considerando outros valores paramétricos de janela neural e algoritmos de treinamento;

Podem-se testar diferentes combinações de previsões, em que os pesos adaptativos podem ser ajustados com técnicas numéricas multiobjetivo. Como aperfeiçoamento no Passo 3, do ponto de vista de tempo de processamento computacional, pode-se usar algoritmos genéticos e PSO para a obtenção de parâmetros neuronais ótimos para as RNA' s.

Em séries históricas de instrumento que apresentem volatilidade (variância condicional) - que é uma estrutura de autocorrelação simples e parcial quadrática nos dados -, as predições da média e da variância condicionais das leituras futuras dos instrumentos podem se dar por meio de uma RNA-SSA-Wavelet recorrente. Uma vez que volatilidade não é observável, existe incerteza quanto aos modelos individuais a serem utilizados em sua modelagem. Assim sendo, pode-se combinar linearmente e não linearmente os mais prováveis.

Modelos Bayesianas talvez possam ser mais eficientes, do ponto de vista de ajuste e de tempo de processamento, no processo de estimação paramétrica do método proposto nesta tese.

REFERÊNCIAS

ABELÉM, A. J. G. **Redes neurais artificiais na previsão de séries temporais**. PUC-Rio. Rio de Janeiro. 1994.

ABRAMOVICH, F.; BAILEY, T. C.; SAPATINAS, T. Wavelet analysis and its statistical applications. **The Statistician**, v. 49, n. 1, p. 1-29, 2000.

ANDRADE, R. M. D. **Hidrogeotecnia nas Barragens:** Método de Análise. Rio de Janeiro: Engevix, 1984.

ANDRADE, R. M. D. **Mecânica do Escoamento em Maciços Fraturados Aplicada a Barragens**. Rio de Janeiro: Engevix, 1988.

BARTKIEWICZ, W. Prediction Intervals for Short-Term Load Forecasting Neuro-Fuzzy Models. **Przeglad Elektrotechniczny**, p. 284-287, 2012.

BERGMEIR, C.; BENITEZ, J. M. Neural Networks in R Using the Stuttgart Neural Network Simulator: RSNNS. **Journal of Statistical Software**, v. 46, n. 7, p. 1-26, 2012.

BOX, G. E. P.; JENKINS, G. M. **Time series analysis:** forecasting and control. San Francisco: Holden-Day, 1970.

BOX, G. E. P.; JENKINS, G. M.; REINSEL, G. C. **Time Series Analysis:** Forecasting and Control. 4ª. ed. New Jersey: Wiley, 2008.

BOX, G. E. P.; TIAO, G. C. Intervention Analysis with Applications to Economic and Environmental Problems, v. 70, n. 349, p. 70-79, 1975.

BRASIL. Lei Nº 12.334 de 20 de setembro de 2010. **Presidência da República**, 2010. Disponivel em: <http://www.planalto.gov.br/ccivil_03/_Ato2007-2010/2010/Lei/L12334.htm>. Acesso em: 5 maio 2014.

BROCK, W.; DECHERT, W. D.; SCHEINKMAN, J. A test for independence based on the correlation dimention. **Economic Reviews**, v. 15, n. 3, p. 197-235, 1987.

CAMPOS, L. C. D. **Modelo estocástico periódico baseado em redes neurais**. PUC-Rio. Rio de Janeiro. 2010.

CARNEY, J. G.; CUNNINGHAM, P.; BHAGWAN, U. **Confidence and prediction intervals for neural network ensembles**. International Join Conference on Neural Networks. Washington, DC, USA: [s.n.]. 1999.

CASSIANO, K. M.; MENEZES, M. L. D.; PESSANHA, J. F. M. **Algumas Abordagens para Identificação de Ruídos na Análise Espectral Singular de Séries Temporais**. XLVI Simpósio Brasileiro de Pesquisa Operacional. Salvador: [s.n.]. 2014. p. 1276-1286.

CAVELERI, R.; RIBEIRO, E. P. Combinação de previsões de volatilidade: um estudo. **Economia**, Brasília, v. 12, n. 2, p. 239-261, mai/ago 2011.

CDBD. **Guia Básico de Segurança de Barragens - Comitê Brasileiro de Barragens**. XXIII Seminário Nacional de Grandes Barragens. Belo Horizonte: [s.n.]. Março 1999.

CLEMEN, R. T. Combining forecasts: a review and annotated bibliography. **International Journal of Forecasting**, North-Holland, v. 5, p. 559-583, 1989.

CNRH. Conselho Nacional de Recursos Hídricos - Resolução Nº 144, de 10 de julho de 2012. **Diário Oficial da União**, 2012. Disponivel em: <http://www.cnrh.gov.br/index.php?option=com_docman&task=doc_download&gid=1636>. Acesso em: 12 Dez 2015.

CYBENKO, G. Approximation by Superposition of a sigmoidal function. **Mathematics of Control, Signals and Systems**, v. 2, n. 4, p. 303-314, Dez 1989.

DAUBECHIES, I. Orthonormal Bases of Compactly Supported Wavelets. **Communications on Pure and Applied Mathematics**, v. 41, p. 909-996, 1988.

DAUBECHIES, I. **Ten Lectures on Wavelets**. [S.l.]: CBMS-NSF Regional Conference Series In Applied Mathematics: SIAM, 1992.

DONOHO, D. L.; JOHNSTONE, I. M. Ideal Spatial Adaptation by Wavelet Shrinkage. **Biometrika**, v. 81, n. 3, p. 425-455, 1994.

DONOHO, D. L.; JOHNSTONE, I. M. Minimax estimation via wavelet shrinkage. **The Annals of Statistics**, v. 26, n. 3, p. 879-971, Jun 1998.

DONOHO, D. L.; JOHNSTONE, I. M. Adapting to unknow smoothness via wavelet shrinkage. **Journal of the American Statistical Association**, v. 90, n. 432, p. 1-27, Dez 1999.

EFRON, B. Bootstrap methods: another look at the Jacknife. **The Annals of Statistics**, v. 7, n. 1, p. 1-26, 1979.

EFRON, B.; TIBSHIRANI, R. Bootstrap Methods for Standard Errors, Confidence Intervals, and Other Measures of Statistical Accuracy. **Statistical Science**, v. 1, n. 1, p. 54-75, Fev 1986.

ELSNER, J. B.; TSONIS, A. A. **Singular Spectrum Analysis:** a new tool in time series analysis. [S.l.]: Springer, 1996.

FARIA, Á.; MUBWANDARIKWA, E. Multimodality on the geometric combination of Bayesian forecasting models. **International Journal of Statistics and Management Systems**, v. 3, p. 1-25, 2008.

FIORINI, A. S. et al. **Barragem de Itaipu – Comparação de Comportamento dos Diferentes Tipos de Barragem de Concreto**. XXIV Seminário Nacional de Grandes Barragens. Fortaleza: [s.n.]. 2001. p. 599-614.

FIRMINO, P. R. A.; MATTOS NETO, P. S. G.; FERREIRA, T. A. E. Correcting and combining time series forecasters. **Neural Networks**, v. 50, p. 1-11, Fev 2014.

FLORES, B. E.; WHITE, E. M. A Framework for the combination of forecasts. **Journal of the Academy of Marketing Science**, v. 16, n. 3, p. 95-103, Set 1988.

FREEDMAN, D. A.; PETERS, S. C. Bootstrapping a Regression Equation: Some Empirical Results. **Journal of the American Statistical Association**, v. 79, n. 385, p. 97-106, Mar 1984.

GOLYANDINA, N.; KOROBEYNIKOV, A. Basic Singular Spectrun Analysis and Forecasting with R. **Computational Statistics and Data Analysis**, v. 71, p. 934-954, 2014.

GOLYANDINA, N.; NEKRUTKIN, V.; ZHIGLJAVSKY, A. A. **Analysis of time series structure:** SSA and Related Techniques. [S.l.]: Chapman & Hall, 2001.

GOLYANDINA, N.; ZHIGLJAVSKY, A. **Singular Spectrum Analysis for Time Series**. [S.l.]: Springer, 2013.

GUPTA, S.; WILTON, P. C. Combination of Forecasts: An Extension. **Management Science**, v. 33, n. 3, p. 356-372, 1987.

HAAR, A. Zur Theorie der orthogonalen Funktionensysteme. **Mathematische Annalen**, v. 69, n. 3, p. 331-371, Set 1910.

HAMILTON, J. D. **Time Series Analysis**. [S.l.]: Princeton Unioversity Press, 1994.

HASSANI, H. Singular Spectrum Analysis: Methodology and Comparison. **Journal of Data Science**, Munich, v. 5, p. 239-257, 1 Apr 2007.

HAYKIN, S. **Redes Neurais:** Princípios e Prática. 2. ed. Porto Alegre: Bookman, 2001.

HERRERA, F.; LOZANO, M. Tackling real-coded genetic algoritms: Operators and tools for behavioral analysis. **Artificial Intelligence Review**, v. 12, n. 4, p. 265-319, 1998.

HESKES, T. Pratical confidence and predction intervals. **Advances in Neural Information Processing Systems**, Cambridge, v. 9, p. 176-182, 1997.

HYNDMAN, R. J. forecast: Forecasting functions for time series and linear models. R package version 6.1, 2015. Disponivel em: <http://github.com/robjhyndman/forecast>. Acesso em: 03 mar. 2015.

HYNDMAN, R. J.; KHANDAKAR, Y. Automatic Time Series Forecasting: The forecast Package for R. **Journal of Statistical Software**, v. 27, n. 3, p. 1-22, july 2008.

ITAIPU. **Usina Hidrelétrica de Itaipu:** Aspectos Técnicos das Estruturas Civis. Foz do Iguaçu: Itaipu Binacional, 2008.

ITAIPU. **Itaipu - Usina Hidrelétrica - Aspectos de Engenharia**. Foz do Iguaçu: Itaipu Binacional, 2009.

ITAIPU BINACIONAL. Características da Barragem. **Itaipu Binacional**. Disponivel em: <https://www.itaipu.gov.br/energia/caracteristicas-da-barragem>. Acesso em: 10 fev. 2016.

KARTHIKEYAN, L.; KUMAR, D. N. Predictability of nonstationary time series using wavelet and EMD based ARMA models. **Journal of Hydrology**, v. 502, p. 103-119, Mai 2013.

KHAN, M. A. R.; POSKITT, D. S. A Note on Window Length Selection in Singular Spectrum Analysis. **Australian & New Zealand Journal of Statistic**, v. 55, n. 2, p. 87-108, jun 2013.

KHOSRAVI, A. et al. Constructing Optimal Prediction Intervals by Using Neural Networks and Bootstrap Method. **IEEE Transaction on Neural Networks and Learning Systems**, v. 26, n. 8, p. 1810-1815, Set 2014.

KISI, O. Wavelet regression model for short-term streamflow forecasting. **Journal of Hydrology**, v. 389, p. 344-353, 2010.

KISI, O.; CIMEN, M. A wavelet-support vector machine conjunction model for monthly streamflow forecasting. **Journal of Hydrology**, n. 339, p. 132-140, 2011.

KOROBEYNIKOV, A. Computation- and space-efficient implementation of SSA. **Statistic and Its Interface**, v. 3, n. 3, p. 257-368, 2010.

KOVACS, Z. L. **Redes Neurais Artificiais - Fundamentos e Aplicações**. [S.l.]: Livraria da Física, 2002.

KUBRUSLY, C. S. **Elements of Operator Theory**. Boston: Birkhäuser, 2001.

KUBRUSLY, C. S.; LEVAN, N. Abstract wavelet generated by Hilbert space shift operators. **Advances in Mathematical Sciences and Applications**, Tokyo, v. 16, n. 2, p. 643-660, Dez 2006.

KUPERMAN, S. C. et al. **Critérios para fixação de valores limites da instrumentação civil de barragens de concreto e de terra**. XXV Seminário Nacional de Grandes Barragens. Salvador: [s.n.]. 2003. p. 1-16.

LEVAN, N.; KUBRUSLY, C. S. A wavelet "time-shift-detail" decomposition. **Mathematics and Computers in Simulation**, Amsterdam, v. 63, n. 2, p. 73-78, Jun 2003.

LIMA, F. G. **Modelos de previsões de séries temporais financeiras com combinação de filtros de kalman e wavelets**. USP - Universidade de São Paulo. Ribeirão Preto. 2011.

LIMA, P. C. D. Wavelets: uma introdução. **Matemática Universitária**, n. 33, p. 13-44, Dez 2002.

LÜTKERPOHL, H. **New Introduction to Multple Time Series Analysis**. [S.l.]: Springer, 2007.

MALLAT, S. **A Wavelet Tour of Signal Processing**. 3. ed. Burlington: Academic Press, 2009.

MARQUARDT, D. W. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. **Journal of the Society for Industrial and Applied Mathematics**, v. 11, n. 2, Jun 1963.

MARTINS, V. L. M. **Comparação de combinação de previsões correlacionadas e não correlacionadas com as suas previsões individuais : um estudo com séries industriais**. UFRGS - Universidade Federal do Rio Grande do Sul. Porto Alegre. 2011.

MASCARENHAS, A. T. **Qualidade Hidráulica e Deteriorações em Fundações de Barragens de Betão**. Lisboa: Laboratório Nacional de Engenharia Civil, 2005.

MASCARENHAS, A. T. **Engenharia de Barragens de Betão:** Estudo Hidrogeotécnico das Fundações. Lisboa: Laboratório Nacional de Engenharia Civil, 2009.

MENEZES, L. D.; SOUZA, R. C.; PESSANHA, J. F. M. Electricity consumption forecasting using singular spectrum analysis. **Dyna**, Medellín, v. 82, p. 138-146, maio 2015.

MONTGOMERY, D. ; RUNGER, G. C. **Estatística aplicada e probabilidade para engenheiros**. 5. ed. [S.l.]: LTC, 2012.

MORETTIN, P. A. Wavelets in Statistics. **Journal of Mathematical Sciences**, São Paulo, v. 3, p. 211-272, 1997.

MORETTIN, P. A.; TOLOI, C. C. M. **Análise de Séries Temporais**. 2. ed. [S.l.]: Blucher, 2006.

MORRIS, J. M.; PERAVALI, R. Minimum-bandwidth discrete-time wavelets. **Signal Processing**, v. 76, n. 2, p. 181-193, julho 1999.

NALLEY, D.; ADAMOWSKI, J.; KHALIL, B. Using discrete wavelet transforms to analyze trends in streamflow and precipitation in Quebec and Ontario (1954–2008). **Journal of Hydrology**, n. 475, p. 204-228, 2012.

NIX, D. A.; WEIGEND, A. S. **Learning local error bars for nonlinear regression**. Advances in Neural Information Processing Systems 7 (NIPS*94). Cambridge, MA: MIT Press. 1995. p. 489-496.

OSAKO, C. I. **A Manutenção dos Drenos nas Fundações de Barragens: O caso da Usina Hidrelétrica de Itaipu**. UFPR. Curitiba. 2002.

PAIVA, A. B. D. Dia a Dia Educação. **Secretaria da Educação - Governo do Estado do Paraná**, 2015. Disponivel em: <http://www.biologia.seed.pr.gov.br/modules/galeria/detalhe.php?foto=256&evento=3>. Acesso em: 05 Jan 2016.

PALIT, A. K.; POPOVIC, D. **Computational intelligence in time series forecasting**. London: Springer, 2005.

PANKRATZ, A. **Forecasting with Dynamic Regression Models**. New York: John Wiley & Sons, 1991.

PESSANHA, J. F. M. et al. **Usando a Análise Espectral Singular na Previsão da Produção Mensal de um Parque Eólico**. Simpósio de Pesquisa Operacional e Pesquisa da Marinha. Rio de Janeiro: Blucher. 2014. p. 201-212.

R CORE TEAM. A language and environment for statistical computing. **R Foundation for Statistical Computing**, 2015. Disponivel em: <http://www.R-project.org>. Acesso em: 20 jan. 2015.

RAGSDALE, C. T. **Spreadsheet Modeling & Decision Analysis**. 6. ed. [S.l.]: South-Western College Pub, 2010.

ROYER, J. C. et al. Short-term solar radiation forecasting by using an iterative combination of wavelet artificial neural networks. **Independent Journal of Management & Production (IJM&P)**, v. 7, n. 1, p. 271-288, 2016.

RUSSEL, S.; NORVIG, P. **Artificial Intelligence:** A Modern Approach. New Jersey: Prentice Hall, 1995.

SOUZA, R. C.; NETO, A. C. A Bootstrap Simulation Study in ARMA(p,q) Structures. **Journal of Forecasting**, v. 15, p. 343-353, 1996.

TAFNER, M. A.; XEREZ, M. D.; RODRIGUES FILHO, I. W. **Redes Neurais Artificiais:** introdução e princípios de neurocomputação. Blumenau: EKO, 1996.

TEIXEIRA JR, A. et al. **Redes neurais artificiais e decomposição wavelet na previsão da radiação solar direta**. XLIV Simpósio Brasileiro de Pesquisa Operacional. Rio de Janeiro: [s.n.]. 2012. p. 1401-1412.

TEIXEIRA JR, A. et al. Residential electricity consumption forecasting using a geometric combination approach. **International Journal of Energy and Statistic**, v. 1, n. 2, p. 1-13, Jun 2013.

TEIXEIRA JR, L. A. **Combinação SSA-Wavelet de Métodos Preditivos com Ajuste Numérico MINIMAX, na Geração de Previsões e de Cenários**. PUC-Rio. Rio de Janeiro, p. 1-114. 2013.

TIBSHIRANI, R. A comparison of some error estimates for neural network models. **Neural Computation**, Cambridge, v. 8, n. 1, p. 152-163, Jan 1996.

TIWARI, M. K.; CHATTERJEE, C. Development of an accurate and reliable hourly flood forecasting model using wavelet–bootstrap–ANN (WBANN) hybrid approach. **Journal of Hydrology**, v. 394, p. 458-470, 2010.

TOURISTLINK. Contraforte barragem. **Touristlink.com**, 2015. Disponivel em: <http://www.touristlink.com.br/Fran%C3%A7a/contraforte-barragem/overview.html>. Acesso em: 10 Fev 2016.

WALLIS, K. F. Combining forecasts: forty years later. **Applied Financial Economics**, v. 21, p. 33-41, 2011.

WERNER, L.; RIBEIRO, J. L. D. Modelo composto para prever demanda através da integração de previsões. **Produção**, v. 16, n. 3, p. 493-509, Dez 2006.

WHITCHER, B. waveslim: Basic wavelet routines for one-, two- and three-dimentional signal processing. R package version 1.7.5. **CRAN R project**, 2015. Disponivel em: <http://CRAN.R-project.org/package=waveslim>. Acesso em: 15 abr. 2015.

WIKIMEDIA COMMONS. File:Img barrage poids.jpg. **Wikimedia Commons**, 2014. Disponivel em: <https://commons.wikimedia.org/wiki/File:Img_barrage_poids.jpg>. Acesso em: 10 Fev 2016.

WIKIPEDIA. Barragem em arco. **Wikipedia**, 2014. Disponivel em: <https://pt.wikipedia.org/wiki/Barragem_em_arco>. Acesso em: 10 Fev 2016.

ZHANG, G. P. Time series forecasting using a hybrid ARIMA and neural network model. **Neurocomputing**, v. 50, p. 159-175, 2003.

ZOU, H.; YANG, Y. Combining time series models for forecasting. **International Journal of Forecasting**, v. 20, n. 1, p. 69-84, 2004.

APÊNDICE

CÓDIGOS-FONTE DESENVOLVIDOS, EM ORDEM ALFABÉTICA.

```
###
# Função decompoe.serie
# Decompõe uma série temporal fazendo a modelagem e previsão da série
# Previsão arimax em paralelo com SSA + wavelet + redes neurais
# Separando as amostras de treino e validação, fazendo decomposições
#     independentes.
# Escolha do melhor modelo pelo MSE na amostra de validação
# SSA para encolhimento de ruídos
# Incluindo na lista de saída os modelos retreinados incluindo os dados
#      de validação
# Considerando variáveis explicativas
# Parâmetros de entrada:
#       serie: série temporal de subpressão
#       t.ar: matriz com a série temporal de temperatura ambiente.
#             Cada coluna é a série com uma defasagem de tempo
#       t.agua: st da temperatura da água que passa no piezômetro
#       rn.minjan: valor mínimo para a janela da rede neural
#       rn.maxjan: valor máximo para a janela da rede neural
#       rn.minneu: número mínimo de neurônios na camada escondida
#       rn.maxneu: número máximo de neurônios na camada escondida
#       rn.minit: número minimo de iterações no treinamento neural
#       rn.maxit: número maximo de iterações no treinamento neural
#       rn.tam.teste: tamanho das amostras de validação e teste
###
decompoe.serie <- function(serie, t.ar, t.agua, rn.minjan, rn.maxjan,
                           rn.minneu, rn.maxneu, rn.minit, rn.maxit,
                           rn.tam.teste)
{
   # ARIMAX
   # n = tamanho da série
   n<-length(serie)
   # n.treino = tamanho da série até o final do treino
   n.treino <- n - (2*rn.tam.teste)
   input.series=cbind(t.ar,t.agua)
   colnames(input.series) <- c("Ar2", "Ar3","Ar4","Agua")
   linear <- auto.arima(serie,xreg=input.series)
   prev.arimax <- fitted(linear,xreg=input.series)
   # Graus de liberdade perdidos. Verificar as perdas por arima.
   tll <- rn.maxjan
   tam.prev <- n - tll
   tam.prev.treino <- tam.prev - (2*rn.tam.teste)
   prev.arimax.tll <- array(0,dim=n-tll)
   serie.tll <- array(0,dim=n-tll)
   for (i in 1:tam.prev) {
      prev.arimax.tll[i] <- prev.arimax[tll+i]
      serie.tll[i] <- serie[tll+i]
   }
   # Separa os trechos até o final do treino e até o final da validação
   # para poder fazer a decomposição wavelet sem considerar dados futuros
   serie.treino <- serie[1:n.treino]
   serie.tll.treino <- serie.tll[1:tam.prev.treino]
```

```
# SSA
# Limites do l ótimo, inteiro: lmin e lmax
lmin <- floor((log(n))^1.5)
lmax <- ceiling((log(n))^2)
if (lmax >= n/2) { lmax <- floor(n/2) }
# Número de tentativas SSA, para definir a quantidade de vetores
# (colunas da matriz)
num.ssa <- lmax - lmin + 1
tend.ssa <- array(0, dim=c(num.ssa,n))
tend.ssa.treino <- array(0, dim=c(num.ssa,n.treino))
erro.ssa <- array(0, dim=c(num.ssa,n))
erro.ssa.treino <- array(0, dim=c(num.ssa,n.treino))

# Especificação de filtros wavelets
filtros=c("haar","d4","d8","d16","mb4","mb8","mb16","mb24","bs3.1",
          "fk4","fk6","fk8","fk14","fk22","la8","la16","la20",
          "bl14","bl20")
nfiltros=length(filtros)
# Matriz para armazenar as decomposições wavelets
# Primeiro índice = Número da tentativa SSA
# Segundo índice = Número do filtro
# Terceiro índice = Número de componentes Wavelet
#                   (6 tendencia (3 aprox + 3 detalhe) + 6 erro SSA)
# Quarto índice = leitura (1 a n)

rn <- array(0, dim=c(num.ssa,nfiltros,6,tam.prev))
rn.treino <- array(0, dim=c(num.ssa,nfiltros,6,tam.prev.treino))

# Matriz para armazenar as previsões e erros de rn
# Primeiro índice = Número da tentativa SSA
# Segundo índice = Número do filtro
# Terceiro índice = Nível da decomposição Wavelet da tendência SSA
# Quarto índice = Valores (1 a n)
prev <- array(0, dim=c(num.ssa,nfiltros,3,tam.prev))
erro <- array(0, dim=c(num.ssa,nfiltros,3,tam.prev))
prev.treino <- array(0, dim=c(num.ssa,nfiltros,3,tam.prev.treino))
erro.treino <- array(0, dim=c(num.ssa,nfiltros,3,tam.prev.treino))

melhor_prev <- array(0, n)
melhor_erro <- array(0, n)
melhor_mse <- 0

# Parametros de melhor resultado
melhor_r <- 1 # Nível de decomposição Wavelet da tendência ssa
melhor_filtro <- 1
melhor_rn_mod <- vector("list",6)
i <- 1
for (l in lmax:lmax) { # Laço para encontrar L ótimo
   i <- l - lmin + 1

   # Serie incluindo a validação
   dec.ssa <- ssa(serie, L=l)
   grupos <- grouping.auto(dec.ssa, grouping.method = "wcor",
                           groups=1:l, nclus=4)
   tendencia <- reconstruct(dec.ssa,groups=list(c(grupos$'1',
                            grupos$'2', grupos$'3')))
   tend.ssa[i,] <- tendencia$F1
   # Serie somente de treino
   dec.ssa <- ssa(serie.treino, L=l)
   grupos <- grouping.auto(dec.ssa, grouping.method = "wcor",
                           groups=1:l, nclus=4)
```

```
tendencia <- reconstruct(dec.ssa,groups=list(c(grupos$'1',
                         grupos$'2', grupos$'3')))
tend.ssa.treino[i,] <- tendencia$F1
# Wavelets
comp <- array(0,dim=c(n,6))
comp.treino <- array(0,dim=c(n.treino,6))
for (f in 1:nfiltros) {
   for (r in 1:3) {
      # Decomposição da serie até o final da validação
      dec.serie<-mra(tend.ssa[i,], wf=filtros[f], J=r)
      # Aproximação Wavelet de nível r
      comp[,2*r-1] <- unlist(dec.serie[r+1])
      # Detalhe Wavelet de nível r
      comp[,2*r] <- unlist(dec.serie[r])

      # Decomposição da serie até o final do treino
      dec.serie<-mra(tend.ssa.treino[i,], wf=filtros[f], J=r)
      # Aproximação Wavelet de nível r
      comp.treino[,2*r-1] <- unlist(dec.serie[r+1])
      # Detalhe Wavelet de nível r
      comp.treino[,2*r] <- unlist(dec.serie[r])
   }
   rn_mod <- list()
   rn_mod2 <- list()
   rn_par <- list()
   for (h in 1:6) {
      co <- comp[,h]
      co.treino <- comp.treino[,h]
      p_rna <- prev_rna11_comp(co.treino, co, t.ar, t.agua,
                               rn.minjan, rn.maxjan, rn.minneu,
                               rn.maxneu, rn.minit, rn.maxit,
                               rn.tam.teste)
      rn[i,f,h,] = c(p_rna$treino.comb, p_rna$valid.comb,
                     p_rna$test.comb)

      mod_comp <- list()
      mod_comp <- append(mod_comp, list(mod1=p_rna$modelo1))
      mod_comp <- append(mod_comp, list(mod2=p_rna$modelo2))
      mod_comp <- append(mod_comp, list(mod3=p_rna$modelo3))
      mod_comp <- append(mod_comp, list(mod4=p_rna$modelo4))
      mod_comp <- append(mod_comp, list(mod5=p_rna$modelo5))
      mod_comp <- append(mod_comp, list(mod.comb=p_rna$modelo))
      mod_comp <- append(mod_comp, list(norm=p_rna$norm,
                                        janela=p_rna$janela,
                                        neuronios=p_rna$neuronios,
                                        iteracoes=p_rna$iteracoes,
                                        norm.comb=p_rna$norm.comb,
                                 neuronios.comb=p_rna$neuronios.comb,
                                        serie.treino=co.treino))
      rn_mod <- append(rn_mod, mod_comp)

      # Modelo2: inclui os dados de validação
      mod2_comp <- list()
      mod2_comp <- append(mod2_comp, list(mod1=p_rna$modelo2.1))
      mod2_comp <- append(mod2_comp, list(mod2=p_rna$modelo2.2))
      mod2_comp <- append(mod2_comp, list(mod3=p_rna$modelo2.3))
      mod2_comp <- append(mod2_comp, list(mod4=p_rna$modelo2.4))
      mod2_comp <- append(mod2_comp, list(mod5=p_rna$modelo2.5))
      mod2_comp <- append(mod2_comp,
                          list(mod.comb=p_rna$modelo2.comb))
      rn_mod2 <- append(rn_mod2, mod2_comp)
```

```
            rn_par <- append(rn_par, list(norm=p_rna$norm,
                                          janela=p_rna$janela,
                                          neuronios=p_rna$neuronios,
                                          iteracoes=p_rna$iteracoes,
                                          norm.comb=p_rna$norm.comb,
                                    neuronios.comb=p_rna$neuronios.comb,
                                          serie.treino=co.treino))

         }

         # A previsão é a soma das componentes de aproximação e de
         #   detalhes de nível r
         # O erro de previsão é a diferença entre a previsão e a serie
         #   original

         for (r in 1:3) { # Tendência SSA (primeiras 6 componentes)
            # Aproximação
            prev[i,f,r,] <- prev[i,f,r,] + rn[i,f,2*r-1,]

            # Detalhes
            for (k in 1:r) {
                prev[i,f,r,] <- prev[i,f,r,] + rn[i,f,2*k,]
            }

            # Previsão final é a média da previsao arimax com RN
            for (k in 1:tam.prev) {
                prev[i,f,r,k] <- (prev[i,f,r,k] + prev.arimax.tll[k])/2
            }

            erro[i,f,r,] <- serie.tll - prev[i,f,r,]
            if (ruidoBranco(erro[i,f,r,]) == 'Y') {
               mse <- mse_comp(prev[i,f,r,],serie.tll)

               if ((melhor_mse == 0) || (mse < melhor_mse)) {
                  melhor_mse <- mse
                  melhor_aic <- 0
                  melhor_erro <- erro[i,f,r,]
                  melhor_prev <- prev[i,f,r,]
                  melhor_l <- l # Número de componentes SSA
                  melhor_i <- i # Posição do numero de componentes SSA
                  melhor_r <- r # Nível de dec Wavelet da tendência ssa
                  melhor_r2 <- 0 # Nível de dec Wavelet do erro ssa
                  melhor_filtro <- f # Filtro Wavelet
                  melhor_rn_mod <- rn_mod
                  melhor_rn_mod2 <- rn_mod2
               }
            }
         }
      }
   }
   lista <- list(prev = prev, erro = erro, rn = rn,
                 mse = melhor_mse, aic = melhor_aic, l = melhor_l,
                 i = melhor_i, f = melhor_filtro, r = melhor_r,
                 r2 = melhor_r2, modelo_rn = melhor_rn_mod,
                 modelo2_rn = melhor_rn_mod2, parametros <- rn_par)
   decompoe.serie <- lista
}

decompoe.serie_comp <- cmpfun(decompoe.serie)
```

```
###
# Função denormaliza
# Aplica a transformação inversa na serie informada.
# Se norm = 1, a normalização é de -1 a 1
#    norm = 2, a normalização é de 0 a 1
#    norm = 3, aplica a normalização com média e desvio padrão
###
denormaliza <- function(serie, norm, serie.media, serie.mult, serie.min,
                        serie.max, serie.sd) {
   if (norm == 1) {
       denormaliza <- denormaliza1_1_comp(serie, serie.media, serie.mult)
   } else if (norm == 2) {
       denormaliza <- denormaliza0_1_comp(serie, serie.min, serie.max)
   } else {
       denormaliza <- denormaliza1_1_comp(serie, serie.media, serie.sd)
   }
}

# Compila a função
denormaliza_comp = cmpfun(denormaliza)


###
# Função denormaliza0_1
# Aplica a transformação inversa na série informada
# serie = serie a ser transformada, entre 0 e 1
# min.serie = mínimo da série original
# max.serie = máximo da série original
###
denormaliza0_1 <- function(serie, min.serie, max.serie) {
   n<-length(serie)
   serie2 <- array(0,dim=c(n))
   range.serie<-max.serie-min.serie
   for(i in 1:n) {
      serie2[i]<-(serie[i]*range.serie)+min.serie
   }
   denormaliza0_1 <- serie2
}

# Compila a função
denormaliza0_1_comp = cmpfun(denormaliza0_1)


###
# Função denormaliza1_1
# Aplica a transformação inversa na série informada
# serie = serie a ser transformada, entre -1 e 1
# min.serie = mínimo da série original
# max.serie = máximo da série original
###
denormaliza1_1 <- function(serie, media, multiplicador) {
   n<-length(serie)
   serie2 <- array(0,dim=c(n))
   for(i in 1:n) {
      serie2[i]<-(serie[i]*multiplicador)+media
   }
   denormaliza1_1 <- serie2
}

# Compila a função
denormaliza1_1_comp = cmpfun(denormaliza1_1)
```

```
###
# Função que prepara uma série de entrada para a previsão de redes neurais
# Parâmetros:
#    serie = série original
#    inicio = posição do início da série original
#    fim = posição do final da série original
#    janela = número de colunas a serem montadas
###
entrada <- function(serie,inicio,fim,janela) {
   n<-length(serie)
   tam.serie <- fim-inicio+1
   if (fim + janela -1 > n) {
      tam.serie <- n - janela - inicio + 1
   }
   serie2 <- array(0,dim=c(tam.serie,janela))
   for(i in inicio:fim) {
      for (j in 1:janela) {
         serie2[i-inicio+1,j]<-serie[i+j-1]
      }
   }
   entrada <- serie2
}
# Compila a função
entrada_comp = cmpfun(entrada)



###
# Função montaAr234Agua
# Compõe uma matriz com quatro colunas:
#        Temperatura do ar de 2, 3 e 4 meses anteriores
#        Temperatura da água atual
# Entradas: t.ar = Temperatura média mensal do ar
#           t.agua = Temperatura média mensal da água
#           regs = Número de registros a compor (0 = todos)
# Saída: Matriz com quatro colunas:
###
montaAr234Agua <- function(t.ar,t.agua,regs)
{
   # Linhas dos registros de entrada
   n <- min(length(t.ar),length(t.agua))
   k <- min(n,regs)  # Linhas desejadas na saída
   ar.234.agua <- array(0,c(k,4))
   for (i in 1:k) {
      lin.ent <- n-i+1
      lin.sai <- k-i+1
      ar.234.agua[lin.sai,1] <- t.ar[lin.ent-2]
      ar.234.agua[lin.sai,2] <- t.ar[lin.ent-3]
      ar.234.agua[lin.sai,3] <- t.ar[lin.ent-4]
      ar.234.agua[lin.sai,4] <- t.agua[lin.ent]
   }
   montaAr234Agua <- ar.234.agua
}
```

```
###
# Função mse
# Retorna o erro quadrático médio (MSE) da previsão.
# Parâmetros: serie = série original
#             prev = série prevista
###
mse <- function(serie,prev) {
   n<-length(serie)
   soma <- 0
   for(i in 1:n) {
      soma <- soma + ((serie[i] - prev[i]) * (serie[i] - prev[i]))
   }
   mse <- soma / n
}

# Compila a função
mse_comp <- cmpfun(mse)


###
# Função normaliza
# Aplica a transformação na serie informada.
# Se norm = 1, a normalização é de -1 a 1
#    norm = 2, a normalização é de 0 a 1
#    norm = 3, aplica a normalização com média e desvio padrão
###
normaliza <- function(serie,norm, serie.media, serie.mult, serie.max,
                      serie.min, serie.sd)
{
   if (norm == 1) {
       normaliza <- normaliza1_1_comp(serie, serie.media, serie.mult)
   } else if (norm == 2) {
       normaliza <- normaliza0_1_comp(serie, serie.min, serie.max)
   } else {
       normaliza <- normaliza1_1_comp(serie, serie.media, serie.sd)
   }
}

# Compila a função
normaliza_comp = cmpfun(normaliza)


###
# Função normaliza0_1
# Aplica uma transformação linear na serie para deixá-la entre 0 e 1
# Parâmetros: serie = série informada
#             min.serie = valor mínimo que a série assume
#             max.serie = valor máximo que a série assume
# Retorno: A série transformada
###
normaliza0_1 <- function(serie, min.serie, max.serie) {
   n<-length(serie)
   serie2 <- array(0,dim=c(n))
   range.serie<-abs(max.serie-min.serie)
   for(i in 1:n) {
      serie2[i]<-(serie[i]-min.serie)/range.serie
   }
   normaliza0_1 <- serie2
}
# Compila a função
normaliza0_1_comp = cmpfun(normaliza0_1)
```

```
###
# Função normaliza1_1
# Aplica uma transformação linear na série informada
# serie = serie a ser transformada
# media.serie = média da série original
# max.serie = valor máximo absoluto da série original.
#             Se max.serie = desvio padrao, funciona como zscore
###
normaliza1_1 <- function(serie,media,max.serie) {
   n<-length(serie)
   serie2 <- array(0,dim=c(n))
   for(i in 1:n) {
      serie2[i]<-(serie[i]-media)/max.serie
   }
   normaliza1_1 <- serie2
}

# Compila a função
normaliza1_1_comp = cmpfun(normaliza1_1)



###
# Rotina para incluir os pacotes necessários ao processamento
###
require(compiler)
require(forecast)
require(waveslim)
require(Rssa)
require(tseries)
require(RSNNS)
mlp_comp <- cmpfun(mlp)
predict_comp <- cmpfun(predict)
```

```
###
# Função preve_serie_boot
# É chamada pelo bootstrap para fazer a previsão de uma série por um
# modelo neural informado
# Parâmetros:
#    serie: série temporal original
#    ar.234.agua.norm: série temporal de variáveis exógenas normalizadas
#                      temp. ambiente com defasagem de 2, 3 e 4 meses
#                      temperatura da água
#    norma: normalização utilizada (0, 1 ou 2)
#    modelo: modelo neural treinado
#    serie.treino: amostra de treino da série original, para usar a
#                  mesma normalização usada no treinamento
###
preve_serie_boot <- function(serie, ar.234.agua.norm, norma, modelo,
                             serie.treino)
{
   tam.serie <- length(serie)     # Tamanho da série
   janela <- modelo$nInputs - 4   # Tamanho da janela

   # Normalização
   se <- serie
   serie.media <- mean(se)
   serie.max <- max(se)
   serie.min <- min(se)
   serie.mult <- max(abs(serie.min-serie.media),serie.max-serie.media)
   serie.sd <- sd(se)
   serie.norm <- normaliza(serie, norma, serie.media, serie.mult,
                           serie.max, serie.min, serie.sd)

   # Montagem do padrão de entrada para a previsão
   inputSerie <- entrada_comp(serie.norm, tam.serie-janela+1,
                              tam.serie-janela+1, janela)
   padraoEntrada <- array(0,dim=c(1,janela+4))
   padraoEntrada[1,] <- c(inputSerie,ar.234.agua.norm)
   previsao <- predict(modelo, padraoEntrada)

   # Desnormaliza a previsão
   preve_serie_boot <- denormaliza(previsao, norma, serie.media,
                                   serie.mult, serie.max, serie.min,
                                   serie.sd)
}

preve_serie_boot_comp <- cmpfun(preve_serie_boot)
```

```
###
# Função preve_serie
# Faz a previsão de uma série por um modelo neural informado
# Parâmetros:
#     serie: série temporal original
#     ar.234.agua.norm: série temporal de variáveis exógenas normalizadas
#                       temp. ambiente com defasagem de 2, 3 e 4 meses
#                       temperatura da água
#     norma: normalização utilizada (0, 1 ou 2)
#     modelo: modelo neural treinado
#     serie.treino: amostra de treino da série original, para usar a
#                   mesma normalização usada no treinamento
###
preve_serie <- function(serie, ar.234.agua.norm, norma, modelo,
                        serie.treino)
{
   tam.serie <- length(serie)     # Tamanho da série
   janela <- modelo$nInputs - 4   # Tamanho da janela

   # Normalização
   se <- serie.treino
   serie.media <- mean(se)
   serie.max <- max(se)
   serie.min <- min(se)
   serie.mult <- max(abs(serie.min-serie.media),serie.max-serie.media)
   serie.sd <- sd(se)
   serie.norm <- normaliza(serie, norma, serie.media, serie.mult,
                           serie.max, serie.min, serie.sd)

   # Montagem do padrão de entrada para a previsão
   inputSerie <- entrada_comp(serie.norm, tam.serie-janela+1,
                              tam.serie-janela+1,j anela)
   padraoEntrada <- array(0,dim=c(1,janela+4))
   padraoEntrada[1,] <- c(inputSerie,ar.234.agua.norm)
   previsao <- predict(modelo, padraoEntrada)

   # Desnormaliza a previsão
   preve_serie <- denormaliza(previsao, norma, serie.media, serie.mult,
                              serie.max, serie.min, serie.sd)
}

preve_serie_comp <- cmpfun(preve_serie)
```

```
###
# Função previsao1Passo
# Realiza a previsão hibrida um passo adiante de uma série temporal
# Previsao paralela Arima e SSA+Wavelet+RN
# SSA só como redução de ruídos
# RN com modelo retreinado considerando validação e variáveis explicativas
# Parâmetros de entrada:
#    serie = série original de entrada
#    mod_arimax = modelo arimax usado para a previsão
#    ar.234.agua.norm = as séries temporais das variáveis exógenas
#                       (causais) normalizadas
#    l = Tamanho da janela SSA
#    f = Indice que identifica o filtro wavelet usado na decomposiçao
#    r = nível de decomposição wavelet da tendência SSA
#    r2 = nível de decomposição wavelet do erro SSA
#    mod_rna = modelos de redes neurais treinados para as previsões
#              das componentes e combinação das mesmas, incluindo
#              as formas de normalização
#    mod2_rna = modelos de redes neurais treinados considerando a
#               amostra de validação
#    mod_rna é uma lista com os seguintes grupos:
#        Apr1 da tendencia SSA
#        Det1 da tendencia SSA
#        Apr2 da tendencia SSA
#        Det2 da tendencia SSA
#        Apr3 da tendencia SSA
#        Det3 da tendencia SSA
#
#    Cada grupo é composto por 13 posições:
#        os 5 melhores modelos de previsão das componentes
#        o modelo da combinação das melhores componentes
#        um vetor com as 5 melhores normas
#        um vetor com as 5 melhores janelas
#        um vetor com os 5 melhores numeros de neurônios
#        um vetor com as 5 melhores iterações
#        a norma usada na combinação
#        a quantidade de neurônios usadas na combinação
#        a quantidade de iterações usadas na combinação
###

previsao1Passo <- function (serie, mod_arimax, ar.234.agua.norm, l, f, r,
                            r2, mod_rna, mod2_rna)
{
      offset_norma <- 7
      offset_janela <- 8
      offset_neuronios <- 9
      offset_iteracoes <- 10
      offset_norma_comb <- 11
      offset_neuronios_comb <- 12
    offset_serie_treino <- 13

    ### Passo 1: Decompor e reagrupar com SSA com l componentes a serie
    # em tendencia e erro SSA
    dec.ssa <- ssa(serie, L=l)
    grupos <- grouping.auto(dec.ssa, grouping.method = "wcor", groups=1:l,
                            nclus=4)
    tendencia <- reconstruct(dec.ssa,groups=list(c(grupos$'1', grupos$'2',
                             grupos$'3')))
    tend.ssa <- tendencia$F1

    ### Passo 2: Decompor a tendência SSA com wavelets em nível r,
```

```
# com filtro f
filtros=c("haar","d4","d8","d16","mb4","mb8","mb16","mb24","bs3.1",
          "fk4","fk6","fk8","fk14","fk22","la8","la16","la20",
          "bl14","bl20")
# Reserva espaço para as componentes wavelet (1 apr. + r detalhes)
n <- length(tend.ssa)
comp <- array(0,dim=c(n,r+1))
tend<-mra(tend.ssa, wf=filtros[f], J=r)
# Tendencia SSA, Aproximação Wavelet de nível r
comp[,1] <- unlist(tend[r+1])
for (i in 1:r) {
    # Tendencia SSA, Detalhe Wavelet de nível até r
    comp[,i+1] <- unlist(tend[i])
}

### Passo 3: Prever cada uma das componentes da tendencia SSA
# Reserva espaço para as previsões um passo adiante
prev_comp.norm <- array(0,dim=r+1)
res <- array(0,dim=5)
# Indice dos modelos da aproximação de nível r
apr.r <- (r-1)*26 # 13 para aproximação e 13 para detalhes
ind.mod2 <- (r-1)*12
# Separa as normas dos melhores modelos e a norma combinada do modelo
norma <- mod_rna[[offset_norma+apr.r]]
norma.comb <- mod_rna[[offset_norma_comb+apr.r]]
serie.treino <- comp[,1]

# Preve a componente de aproximação de nível r
# Obtem as 5 melhores previsões
for (i in 1:5) {
   res[i] <- preve_serie_comp(comp[,1], ar.234.agua.norm, norma[i],
                              mod2_rna[[i+ind.mod2]], serie.treino)
}
# Combina as 5 melhores previões
res.norm <- array(0,dim=c(1,5))
co <- comp[,1]
comp.media <- mean(co)
comp.max <- max(co)
comp.min <- min(co)
comp.mult <- max(abs(comp.media-comp.min),comp.max-comp.media)
comp.sd <- sd(co)
res.norm[1,] <- normaliza_comp(res, norma.comb, comp.media, comp.mult,
                               comp.max, comp.min, comp.sd)
prev_comp.norm[1] <- predict(mod2_rna[[6+ind.mod2]],res.norm)
prev_comp[1] <- denormaliza_comp(prev_comp.norm[1], norma.comb,
                                 comp.media, comp.mult, comp.max,
                                 comp.min, comp.sd)

# Preve as r componentes de detalhe
for (k in 1:r) {
   # Separa as normas dos melhores modelos e a norma combinada
   det.r <- 13+(k-1)*26
   ind.mod2 <- 6+(k-1)*12
   norma <- mod_rna[[offset_norma+det.r]]
   norma.comb <- mod_rna[[offset_norma_comb+det.r]]
   serie.treino <- comp[,k+1]
   co <- comp[,k+1]
   comp.media <- mean(co)
   comp.max <- max(co)
   comp.min <- min(co)
   comp.mult <- max(abs(comp.media-comp.min),comp.max-comp.media)
```

```
        comp.sd <- sd(co)
        # Obtem as 5 melhores previsões
        for (i in 1:5) {
           res[i] <- preve_serie_comp(comp[,k+1],ar.234.agua.norm,norma[i],
                                      mod2_rna[[i+ind.mod2]],serie.treino)
        }
        # Combina as 5 melhores previões
        res.norm[1,] <- normaliza_comp(res, norma.comb, comp.media,
                                       comp.mult, comp.max, comp.min,
                                       comp.sd)
        prev_comp.norm[k+1] <- predict(mod2_rna[[6+ind.mod2]],res.norm)
        prev_comp[k+1] <- denormaliza_comp(prev_comp.norm[k+1], norma.comb,
                                           comp.media, comp.mult, comp.max,
                                           comp.min, comp.sd)
    }

    ### Passo 4: Somar as previsões de todas as componentes para obter a
    ###          previsão em T+1
    soma <- 0
    for (i in 1:(r+1)) {
       soma <- soma + prev_comp[i]
    }
    previsao1Passo <- soma

}

previsao1Passo_comp <- cmpfun(previsao1Passo)
```

```
###
# Função previsao1Passo_boot
# Realiza a previsão hibrida um passo adiante para bootstrap
# Previsao paralela Arima e SSA+Wavelet+RN
# SSA só como redução de ruídos
# RN com modelo retreinado considerando validação e variáveis explicativas
# Parâmetros de entrada:
#    serie = série original de entrada
#    mod_arimax = modelo arimax usado para a previsão
#    ar.234.agua.norm = as séries temporais das variáveis exógenas
#                       (causais) normalizadas
#    l = Tamanho da janela SSA
#    f = Indice que identifica o filtro wavelet usado na decomposiçao
#    r = nível de decomposição wavelet da tendência SSA
#    r2 = nível de decomposição wavelet do erro SSA
#    mod2_rna = modelos de redes neurais treinados considerando a
#               amostra de validação
#        é uma lista com os seguintes grupos:
#        Apr1 da tendencia SSA
#        Det1 da tendencia SSA
#        Apr2 da tendencia SSA
#        Det2 da tendencia SSA
#        Apr3 da tendencia SSA
#        Det3 da tendencia SSA
#
#    Cada grupo é composto por 6 posições:
#        os 5 melhores modelos de previsão das componentes
#        o modelo da combinação das melhores componentes
#
#    param_rna é uma lista com os seguintes grupos:
#        Apr1 da tendencia SSA
#        Det1 da tendencia SSA
#        Apr2 da tendencia SSA
#        Det2 da tendencia SSA
#        Apr3 da tendencia SSA
#        Det3 da tendencia SSA
#    Cada grupo contém 7 componentes:
#        um vetor com as 5 melhores normas
#        um vetor com as 5 melhores janelas
#        um vetor com os 5 melhores numeros de neurônios
#        um vetor com as 5 melhores iterações
#        a norma usada na combinação
#        a quantidade de neurônios usadas na combinação
#        a série usada no treinamento, para normalização vetor de treino
###

previsao1Passo_boot <- function (serie, ar.234.agua.norm, l, f, r, r2,
                                 param_rna, mod2_rna)
{

      offset_norma <- 1
      offset_janela <- 2
      offset_neuronios <- 3
      offset_iteracoes <- 4
      offset_norma_comb <- 5
      offset_neuronios_comb <- 6
    offset_serie_treino <- 7

    ### Passo 1: Decompor e reagrupar com SSA com l componentes a serie
    # em tendencia e erro SSA
    dec.ssa <- ssa(serie, L=l)
```

```
grupos <- grouping.auto(dec.ssa, grouping.method = "wcor", groups=1:l,
                        nclus=4)
tendencia <- reconstruct(dec.ssa,groups=list(c(grupos$'1',grupos$'2',
                         grupos$'3')))
tend.ssa <- tendencia$F1

### Passo 2: Decompor a tendência com wavelets em nível r,
# com filtro f
filtros=c("haar","d4","d8","d16","mb4","mb8","mb16","mb24","bs3.1",
          "fk4","fk6","fk8","fk14","fk22","la8","la16","la20",
          "bl14","bl20")
# Reserva espaço para as componentes wavelet (1 apr. + r detalhes)
n <- length(tend.ssa)
comp <- array(0,dim=c(n,r+1))
tend<-mra(tend.ssa, wf=filtros[f], J=r)
# Tendencia SSA, Aproximação Wavelet de nível r
comp[,1] <- unlist(tend[r+1])
for (i in 1:r) {
    # Tendencia SSA, Detalhe Wavelet de nível até r
    comp[,i+1] <- unlist(tend[i])
}

### Passo 3: Prever cada uma das componentes da tendencia SSA
# Reserva espaço para as previsões um passo adiante
prev_comp.norm <- array(0,dim=r+1)
res <- array(0,dim=5)
# Indice dos modelos da aproximação de nível r
par.r <- (r-1)*7
ind.mod2 <- (r-1)*12
# Separa as normas dos melhores modelos e a norma combinada do modelo
norma <- param_rna[[offset_norma+par.r]]
norma.comb <- param_rna[[offset_norma_comb+par.r]]
serie.treino <- param_rna[[offset_serie_treino+par.r]]

# Preve a componente de aproximação de nível r
# Obtem as 5 melhores previsões
for (i in 1:5) {
   res[i] <- preve_serie_boot_comp(comp[,1], ar.234.agua.norm,
                                   norma[i], mod2_rna[[i+ind.mod2]],
                                   comp[,1])
}
# Combina as 5 melhores previões
res.norm <- array(0,dim=c(1,5))
co <- comp[,1]
comp.media <- mean(co)
comp.max <- max(co)
comp.min <- min(co)
comp.mult <- max(abs(comp.media-comp.min),comp.max-comp.media)
comp.sd <- sd(co)
res.norm[1,] <- normaliza_comp(res, norma.comb, comp.media, comp.mult,
                               comp.max, comp.min, comp.sd)
prev_comp.norm[1] <- predict(mod2_rna[[6+ind.mod2]],res.norm)
prev_comp[1] <- denormaliza_comp(prev_comp.norm[1], norma.comb,
                                 comp.media, comp.mult, comp.max,
                                 comp.min, comp.sd)

# Preve as r componentes de detalhe
for (k in 1:r) {
   # Separa as normas dos melhores modelos e a norma combinada
   det.r <- 13+(k-1)*26
   par.r <- (k-1)*7
```

```
        ind.mod2 <- 6+(k-1)*12
        norma <- param_rna[[offset_norma+par.r]]
        norma.comb <- param_rna[[offset_norma_comb+par.r]]
        serie.treino <- param_rna[[offset_serie_treino+par.r]]
        co <- comp[1:(n-12),k+1]
        comp.media <- mean(co)
        comp.max <- max(co)
        comp.min <- min(co)
        comp.mult <- max(abs(comp.media-comp.min),comp.max-comp.media)
        comp.sd <- sd(co)
        # Obtem as 5 melhores previsões
        for (i in 1:5) {
           res[i] <- preve_serie_boot_comp(comp[,k+1], ar.234.agua.norm,
                                           norma[i], mod2_rna[[i+ind.mod2]],
                                           serie.treino)
        }
        # Combina as 5 melhores previões
        res.norm[1,] <- normaliza_comp(res, norma.comb, comp.media,
                                       comp.mult, comp.max, comp.min,
                                       comp.sd)
        prev_comp.norm[k+1] <- predict(mod2_rna[[6+ind.mod2]],res.norm)
        prev_comp[k+1] <- denormaliza_comp(prev_comp.norm[k+1], norma.comb,
                                           comp.media, comp.mult, comp.max,
                                           comp.min, comp.sd)
    }

    ### Passo 4: Somar as previsões de todas as componentes para obter
    ###          a previsão em T+1
    soma <- 0
    for (i in 1:(r+1)) {
       soma <- soma + prev_comp[i]
    }
    previsao1Passo_boot <- soma

}

previsao1Passo_boot_comp <- cmpfun(previsao1Passo_boot)
```

```
###
# Função previsaoHpassos
# Realiza a previsão hibrida H passos adiante de uma série temporal
# SSA como redução de ruídos + Wavelet + RNAX
# Recebe modelo treinado incluindo dados de validação
# Parâmetros de entrada:
#    serie = série original de entrada
#    leituras = próximas h leituras, para fins de avaliação e avanço one
#               step
#    mod_arimax = modelo arimax usado para a previsão
#    t.ar = temperatura ambiente
#    t.agua = temperatura da água
#    l = Tamanho da janela SSA
#    f = Indice que identifica o filtro wavelet usado na decomposiçao
#    r = nível de decomposição wavelet da tendência SSA
#    r2 = nível de decomposição wavelet do erro SSA
#    mod_rna = modelos de redes neurais treinados para as previsões
#              das componentes e combinação das mesmas, incluindo
#              as formas de normalização
#    mod_rna é uma lista com os seguintes grupos:
#        Apr1 da tendencia SSA
#        Det1 da tendencia SSA
#        Apr2 da tendencia SSA
#        Det2 da tendencia SSA
#        Apr3 da tendencia SSA
#        Det3 da tendencia SSA
#
#    Cada grupo é composto por 8 posições:
#        os 5 melhores modelos de previsão das componentes
#        o modelo da combinação das melhores componentes
#        um vetor com as 5 melhores normas
#        a norma usada na combinação
###
previsaoHpassos <- function (serie, leituras, mod_arimax, t.ar, t.agua,
                             l, f, r, r2, mod_rna, mod2_rna)
{
    tam.serie <- length(serie)
    horizonte <- length(leituras)
    tam.ar <- length(t.ar)
    tam.agua <- length(t.agua)
    # Prevê comportamento das séries explicativas, fora da amostra, one step
    # Temperatura do ar
    t.ar.serie <- t.ar[1:tam.serie]
    t.ar.teste <- t.ar[tam.serie+1:tam.ar]
    arima.ar <- auto.arima(t.ar.serie)
    arima2.ar <- Arima(t.ar.teste,model=arima.ar)
    p <- fitted(arima2.ar)
    prev.ar <- p[1:horizonte]
    # Temperatura da água
    t.agua.serie <- t.agua[1:tam.serie]
    t.agua.teste <- t.agua[tam.serie+1:tam.agua]
    arima.agua <- auto.arima(t.agua.serie)
    arima2.agua <- Arima(t.agua.teste,model=arima.agua)
    p <- fitted(arima2.agua)
    prev.agua <- p[1:horizonte]

    # Previsão One-Step Arimax da subpressão
    t.ar.prev <- c(t.ar.serie,prev.ar)
    t.agua.prev <- c(t.agua.serie,prev.agua)
    ar.234.agua.prev <- montaAr234Agua(t.ar.prev, t.agua.prev, horizonte)
    arimax2 <- Arima(leituras, model=mod_arimax, xreg=ar.234.agua.prev)
```

```
    prev.arimax <- fitted(arimax2)

    p <- array(0,dim=horizonte)
    previsao <- array(0,dim=horizonte)
    ini <- 1
    for (i in 1:horizonte) {
        p[i] <- previsao1Passo_comp(serie, mod_arimax, t.ar.serie,
                                    t.agua.serie, l, f, r, r2, mod_rna,
                                    mod2_rna)
        previsao[i] <- (p[i] + prev.arimax[i])/2
        serie <- c(serie[2:tam.serie],leituras[i])
        ini <- ini+1
        fim <- tam.serie+i
        t.ar.serie <- t.ar[ini:fim]
        t.agua.serie <- t.agua[ini:fim]
    }
    previsaoHpassos <- previsao
}
previsaoHpassos_comp <- cmpfun(previsaoHpassos)


###
# Função previsaoHpassos_boot
# Realiza a previsão hibrida H passos adiante de uma série temporal
# SSA como redução de ruídos + Wavelet + RNAX
# Parâmetros de entrada:
#    serie = série original de entrada
#    leituras = próximas h leituras, para fins de avaliação e avanço
#               one step
#    mod_arimax = modelo arimax usado para a previsão
#    var_causais = as séries temporais das variáveis exógenas (causais)
#    l = Tamanho da janela SSA
#    f = Indice que identifica o filtro wavelet usado na decomposiçao
#    r = nível de decomposição wavelet da tendência SSA
#    r2 = nível de decomposição wavelet do erro SSA
#    mod_rna = modelos de redes neurais treinados para as previsões
#              das componentes e combinação das mesmas, incluindo
#              as formas de normalização
#    mod_rna é uma lista com os seguintes grupos:
#        Apr1 da tendencia SSA
#        Det1 da tendencia SSA
#        Apr2 da tendencia SSA
#        Det2 da tendencia SSA
#        Apr3 da tendencia SSA
#        Det3 da tendencia SSA
#    Cada grupo é composto por 8 posições:
#        os 5 melhores modelos de previsão das componentes
#        o modelo da combinação das melhores componentes
#        um vetor com as 5 melhores normas
#        a norma usada na combinação
###

previsaoHpassos_boot <- function (serie, leituras, mod_arimax, t.ar,
                                  t.agua, l, f, r, r2, param_rna, mod2_rna)
{
    tam.serie <- length(serie)
    horizonte <- length(leituras)
    tam.ar <- length(t.ar)
    tam.agua <- length(t.agua)
    # Prevê o comportamento das séries explicativas, para fora da amostra,
    #       one step
```

```
    # Temperatura do ar
    t.ar.serie <- t.ar[1:tam.serie]
    t.ar.teste <- t.ar[tam.serie+1:tam.ar]
    arima.ar <- auto.arima(t.ar.serie)
    arima2.ar <- Arima(t.ar.teste,model=arima.ar)
    p <- fitted(arima2.ar)
    prev.ar <- p[1:horizonte]
    # Temperatura da água
    t.agua.serie <- t.agua[1:tam.serie]
    t.agua.teste <- t.agua[tam.serie+1:tam.agua]
    arima.agua <- auto.arima(t.agua.serie)
    arima2.agua <- Arima(t.agua.teste,model=arima.agua)
    p <- fitted(arima2.agua)
    prev.agua <- p[1:horizonte]

    # Previsão One-Step Arimax da subpressão
    t.ar.prev <- c(t.ar.serie,prev.ar)
    t.agua.prev <- c(t.agua.serie,prev.agua)
    ar.234.agua.prev <- montaAr234Agua(t.ar.prev, t.agua.prev, horizonte)
    arimax2 <- Arima(leituras, model=mod_arimax, xreg=ar.234.agua.prev)
    prev.arimax <- fitted(arimax2)

    # Previsão One-Step por redes neurais da subpressão
    # Normalizar as variáveis exógenas pela amostra de treino
    ar.media <- mean(t.ar)
    ar.max <- max(t.ar)
    ar.min <- min(t.ar)
    ar.mult <- max(abs(ar.media-ar.min),ar.max-ar.media)
    ar.sd <- sd(t.ar)
    agua.media <- mean(t.agua)
    agua.max <- max(t.agua)
    agua.min <- min(t.agua)
    agua.mult <- max(abs(agua.media-agua.min),agua.max-agua.media)
    agua.sd <- sd(t.agua)
    norm <- 1
    t.ar.norm <- normaliza_comp(t.ar.prev, norm, ar.media, ar.mult,
                                ar.max, ar.min, ar.sd)
    t.agua.norm <- normaliza_comp(t.agua.prev, norm, agua.media,
                                  agua.mult, agua.max, agua.min, agua.sd)
    ar.234.agua.norm <- montaAr234Agua(t.ar.norm, t.agua.norm, horizonte)

    previsao <- array(0,dim=h)
    prev.rn <- array(0,dim=h)
    ini <- 1
    t.ar.i <- t.ar.serie
    t.agua.i <- t.agua.serie
    for (i in 1:horizonte) {
        t.ar.i <- c(t.ar.i[2:tam.serie],prev.ar[i])
        t.agua.i <- c(t.agua.i[2:tam.serie],prev.agua[i])
        prev.rn[i] <- previsao1Passo_boot11_comp(serie,
                                                 ar.234.agua.norm[i,],
                                                 l, f, r, r2, param_rna,
                                                 mod2_rna)
        serie <- c(serie[2:tam.serie],leituras[i])
        ini <- ini+1
        fim <- tam.serie+i
        previsao[i] <- (prev.rn[i] + prev.arimax[i])/2
    }
    previsaoHpassos_boot <- previsao
}
previsaoHpassos_boot_comp <- cmpfun(previsaoHpassos_boot)
```

```
###
# Função prev_rna
# Faz a previsão RNA com variáveis explicativas (RNAX)
# Decompondo separadamente as amostras de treino, validação e teste
# Retreinando a série incluindo dados de validação
# Escolhendo pela média entre o MSE da validação e o MSE do teste
# Parâmetros:
#     serie.treino: amostra de treino da série original
#     serie: série original completa
#     t.ar: série temporal da temperatura do ar,
#           com defasagens de 2, 3 e 4 meses
#     t.agua: série temporal da temperatura da água
#     janela.min: valor mínimo para o tamanho da janela neural
#     janela.max: valor máximo para o tamanho da janela neural
#     neuro.ce.min: número mínimo de neurônios na camada escondida
#     neuro.ce.max: número máximo de neurônios na camada escondida
#     iter.min: número mínimo de iterações de treinamento
#     iter.max: número máximo de iterações de treinamento
#     tam.teste: tamanho das amostras de validação e teste
###

prev_rna <- function(serie.treino, serie, t.ar, t.agua, janela.min,
                     janela.max, neuro.ce.min, neuro.ce.max, iter.min,
                     iter.max, tam.teste)
{
   num_melhores <- 5
   tam.serie.treino <- length(serie.treino)
   tam.serie <- length(serie)
   tam.serie.valid <- tam.serie - tam.teste
   tam.janela <- janela.max - janela.min + 1
   tam.neuro <- neuro.ce.max - neuro.ce.min + 1
   tam.iter <- iter.max - iter.min + 1
   tam.treino <- tam.serie.treino - janela.max
   ini.treino <- janela.max + 1
   fim.treino <- ini.treino + tam.treino - 1
   ini.valid  <- fim.treino + 1
   fim.valid  <- ini.valid + tam.teste - 1
   ini.test   <- fim.valid + 1
   fim.test  <- ini.test + tam.teste - 1
   serie.valid <- serie[1:tam.serie.valid]
   serie.media <- mean(serie.valid)
   serie.max <- max(serie.valid)
   serie.min <- min(serie.valid)
   serie.mult <- max(abs(serie.min-serie.media),serie.max-serie.media)
   serie.sd <- sd(serie.valid)
   serie.media.treino <- mean(serie.treino)
   serie.max.treino <- max(serie.treino)
   serie.min.treino <- min(serie.treino)
   serie.mult.treino <- max(abs(serie.min.treino - serie.media.treino),
                            serie.max.treino - serie.media.treino)
   serie.sd.treino <- sd(serie.treino)

   ar.media <- mean(t.ar[,1])
   ar.max <- max(t.ar[,1])
   ar.min <- min(t.ar[,1])
   ar.mult <- max(abs(ar.min-ar.media),ar.max-ar.media)
   ar.sd <- sd(t.ar[,1])
   agua.media <- mean(t.agua)
   agua.max <- max(t.agua)
   agua.min <- min(t.agua)
   agua.mult <- max(abs(agua.media-agua.min),agua.max-agua.media)
```

```
agua.sd <- sd(t.agua)

num_f <- 1
num_norm <- 2
prevTrain <- array(0,dim=c(tam.janela, tam.neuro, tam.iter, num_f,
                   num_norm, tam.treino)) # Previsoes de Treinamento
prevValid <- array(0,dim=c(tam.janela, tam.neuro, tam.iter, num_f,
                   num_norm, tam.teste)) # Previsoes de Validação
prevTest  <- array(0,dim=c(tam.janela, tam.neuro, tam.iter, num_f,
                   num_norm, tam.teste)) # Previsoes de Teste

melhor_mse  <- array(0,dim=num_melhores)
melhor_j    <- array(0,dim=num_melhores)
melhor_ce   <- array(0,dim=num_melhores)
melhor_iter <- array(0,dim=num_melhores)
melhor_f_a  <- array(0,dim=num_melhores)
melhor_norm <- array(0,dim=num_melhores)
melhor_prevValid <- array(0,dim=c(num_melhores,tam.teste))
melhor_prevTest  <- array(0,dim=c(num_melhores,tam.teste))
melhor_prevTrain <- array(0,dim=c(num_melhores,tam.treino))
melhor_prev2     <- array(0,dim=c(num_melhores,
                                  (tam.serie.valid - janela.max)))
melhor_prevValid_comb <- array(0,dim=tam.teste)
melhor_prevTrain_comb <- array(0,dim=tam.treino)
melhor_prevTest_comb <- array(0,dim=tam.teste)
melhor_mse_comb <- 0
melhor_modelo1 <- 0
melhor_modelo2 <- 0
melhor_modelo3 <- 0
melhor_modelo4 <- 0
melhor_modelo5 <- 0
melhor_m1 <- 0
melhor_m2 <- 0
melhor_m3 <- 0
melhor_m4 <- 0
melhor_m5 <- 0
prevValid.final  <- array(0,dim=c(num_melhores,tam.teste))
prevTest.final  <- array(0,dim=c(num_melhores,tam.teste))
prevTrain.final  <- array(0,dim=c(num_melhores,tam.treino))
func_ativ = c("Act_TanH","Act_Signum","Act_Logistic")

# Normalização
norm <- 1
serie.norm <- normaliza_comp(serie, norm, serie.media.treino,
                             serie.mult.treino, serie.max.treino,
                             serie.min.treino, serie.sd.treino)
serie.norm.treino <- normaliza_comp(serie.treino, norm,
                                    serie.media.treino,
                                    serie.mult.treino,
                                    serie.max.treino,
                                    serie.min.treino,
                                    serie.sd.treino)
t.ar.2.norm <- normaliza_comp(t.ar[,1], norm, ar.media, ar.mult,
                              ar.max, ar.min, ar.sd)
t.ar.3.norm <- normaliza_comp(t.ar[,2], norm, ar.media, ar.mult,
                              ar.max, ar.min, ar.sd)
t.ar.4.norm <- normaliza_comp(t.ar[,3], norm, ar.media, ar.mult,
                              ar.max, ar.min, ar.sd)
t.agua.norm <- normaliza_comp(t.agua, norm, agua.media, agua.mult,
                              agua.max, agua.min, agua.sd)
```

```
# Tamanho da janela
for (j in janela.min:janela.max) {
  i.treino <- janela.max-j+1
  f.treino <- janela.max-j+tam.treino
  iiT <- 1
  fiT <- tam.serie-j-tam.teste
  iiV <- tam.serie-j-tam.teste+1
  fiV <- tam.serie-j
  itT <- j+1
  ftT <- tam.serie-tam.teste
  itV <- tam.serie-tam.teste+1
  ftV <- tam.serie
  inputTrain <- entrada_comp(serie.norm.treino, 1, tam.serie.treino-j,
                             j)
  linhas <- length(inputTrain[,1])
  targetTrain <- entrada_comp(serie.norm.treino, j+1,
                              tam.serie.treino, 1)
  lj <- linhas+j
  dadosTrain <- cbind(inputTrain, t.ar.2.norm[(j+1):lj],
                      t.ar.3.norm[(j+1):lj], t.ar.4.norm[(j+1):lj],
                      t.agua.norm[(j+1):lj])

  inputValid <- entrada_comp(serie.norm,
                             tam.serie.valid-j-tam.teste+1,
                             tam.serie.valid-j, j)
  targetValid <- entrada_comp(serie.norm, tam.serie.valid-tam.teste+1,
                              tam.serie.valid, 1)
  dadosValid <- cbind(inputValid,
         t.ar.2.norm[(tam.serie.valid-tam.teste+1):tam.serie.valid],
          t.ar.3.norm[(tam.serie.valid-tam.teste+1):tam.serie.valid],
          t.ar.4.norm[(tam.serie.valid-tam.teste+1):tam.serie.valid],
          t.agua.norm[(tam.serie.valid-tam.teste+1):tam.serie.valid])

  inputTest <- entrada_comp(serie.norm, tam.serie-j-tam.teste+1,
                            tam.serie-j, j)
  targetTest <- entrada_comp(serie.norm, tam.serie-tam.teste+1,
                             tam.serie, 1)
  dadosTest <- cbind(inputTest,
                      t.ar.2.norm[(tam.serie-tam.teste+1):tam.serie],
                      t.ar.3.norm[(tam.serie-tam.teste+1):tam.serie],
                      t.ar.4.norm[(tam.serie-tam.teste+1):tam.serie],
                      t.agua.norm[(tam.serie-tam.teste+1):tam.serie])

  # Prepara os dados para o treino do modelo incluindo os dados de
  # validação
  input2 <- entrada_comp(serie.norm, 1, tam.serie.valid-j, j)
  linhas2 <- length(input2[,1])
  target2 <- entrada_comp(serie.norm, j+1, tam.serie.valid, 1)
  lj2 <- linhas2+j
  dados2 <- cbind(input2, t.ar.2.norm[(j+1):lj2],
                  t.ar.3.norm[(j+1):lj2], t.ar.4.norm[(j+1):lj2],
                  t.agua.norm[(j+1):lj2])
  # Número de neuronios na camada escondida
  for (neuro.ce in neuro.ce.min:neuro.ce.max) {
    # Número máximo de iterações
    for (iter in iter.min:iter.max) {
      # Função de ativação
      for (f in 1:num_f) {
```

```
   modelo <- mlp_comp(dadosTrain, targetTrain, size=neuro.ce,
                      maxit=iter, initFunc="Randomize_Weights",
                      initFuncParams=c(-0.3,0.3),
                      learnFunc="SCG",
                      learnFuncParams=c(0,0,0,0),
                      updateFunc="Topological_Order",
                      updateFuncParams=0.0,
                      hiddenActFunc=func_ativ[f],
                      shufflePatterns=F,linOut=T)

# obtem os valore previstos de treinamento do modelo
   # Pega somente os últimos "tam.treino" valores,
   # desprezando a janela inicial
prevTrain[j-janela.min+1, neuro.ce-neuro.ce.min+1,
             iter-iter.min+1,f,norm,] <- modelo$fitted.values
                                          [i.treino:f.treino]
# Faz a previsão na amostra de validação
prevValid[j-janela.min+1, neuro.ce-neuro.ce.min+1,
             iter-iter.min+1,f,norm,] <- predict_comp(modelo,
                                                      dadosValid)
mse.valid <- mse_comp(prevValid[j-janela.min+1,
                                neuro.ce-neuro.ce.min+1,
                                iter-iter.min+1,f,norm,],
                      targetValid)
# Treina o modelo incluindo a amostra de validação,
# e prevê a amostra de teste
modelo2 <- mlp_comp(dados2, target2, size=neuro.ce,
                    maxit=iter, initFunc="Randomize_Weights",
                    initFuncParams=c(-0.3,0.3),
                    learnFunc="SCG",
                    learnFuncParams=c(0,0,0,0),
                    updateFunc="Topological_Order",
                    updateFuncParams=0.0,
                    hiddenActFunc=func_ativ[f],
                    shufflePatterns=F,linOut=T)
   prevTest[j-janela.min+1, neuro.ce-neuro.ce.min+1,
            iter-iter.min+1,f,norm,] <- predict_comp(modelo2,
                                                     dadosTest)
   mse.teste <- mse_comp(prevTest[j-janela.min+1,
                                  neuro.ce-neuro.ce.min+1,
                                  iter-iter.min+1,f,norm,],
                         targetTest)

# Testa cada previsão contra o vetor de melhores
for (ll in 1:num_melhores) {
  n <- num_melhores - ll + 1
  if (melhor_mse[n] == 0 || mse.valid < melhor_mse[n]) {
    prev2 <- modelo2$fitted.values
    # Abre espaço no vetor para o novo elemento
    if (ll > 1) {
      melhor_mse[n+1]    <- melhor_mse[n]
      melhor_j[n+1]      <- melhor_j[n]
      melhor_ce[n+1]     <- melhor_ce[n]
      melhor_iter[n+1]   <- melhor_iter[n]
      melhor_f_a[n+1]    <- melhor_f_a[n]
      melhor_norm[n+1]   <- melhor_norm[n]
      melhor_prevValid[n+1,] <- melhor_prevValid[n,]
      melhor_prevTrain[n+1,] <- melhor_prevTrain[n,]
      melhor_prevTest[n+1,]  <- melhor_prevTest[n,]
      melhor_prev2[n+1,]     <- melhor_prev2[n,]
    }
```

```
      # Insere novo elemento no vetor
      melhor_mse[n]    <- mse.valid
      melhor_j[n]      <- j
      melhor_ce[n]     <- neuro.ce
      melhor_iter[n]   <- iter
      melhor_f_a[n]    <- f
      melhor_norm[n]   <- norm
      melhor_prevValid[n,] <- prevValid[j-janela.min+1,
                                        neuro.ce-neuro.ce.min+1,
                                        iter-iter.min+1,f,norm,]
      melhor_prevTest[n,]  <- prevTest[j-janela.min+1,
                                       neuro.ce-neuro.ce.min+1,
                                       iter-iter.min+1,f,norm,]
      melhor_prevTrain[n,] <- prevTrain[j-janela.min+1,
                                        neuro.ce-neuro.ce.min+1,
                                        iter-iter.min+1,f,norm,]
      x <- prev2[((janela.max-j)+1):(tam.serie.valid-j)]
      melhor_prev2[n,] <- prev2[((janela.max-j)+1):
                                (tam.serie.valid-j)]
      if (ll == 1) {
        melhor_modelo5 <- modelo
        melhor_m5 <- modelo2
      }
      else if (ll == 2) {
        melhor_modelo5 <- melhor_modelo4
        melhor_modelo4 <- modelo
        melhor_m5 <- melhor_m4
        melhor_m4 <- modelo2
      }
      else if (ll == 3) {
        melhor_modelo5 <- melhor_modelo4
        melhor_modelo4 <- melhor_modelo3
        melhor_modelo3 <- modelo
        melhor_m5 <- melhor_m4
        melhor_m4 <- melhor_m3
        melhor_m3 <- modelo2
      }
      else if (ll == 4) {
        melhor_modelo5 <- melhor_modelo4
        melhor_modelo4 <- melhor_modelo3
        melhor_modelo3 <- melhor_modelo2
        melhor_modelo2 <- modelo
        melhor_m5 <- melhor_m4
        melhor_m4 <- melhor_m3
        melhor_m3 <- melhor_m2
        melhor_m2 <- modelo2
      }
      else if (ll == 5) {
        melhor_modelo5 <- melhor_modelo4
        melhor_modelo4 <- melhor_modelo3
        melhor_modelo3 <- melhor_modelo2
        melhor_modelo2 <- melhor_modelo1
        melhor_modelo1 <- modelo
        melhor_m5 <- melhor_m4
        melhor_m4 <- melhor_m3
        melhor_m3 <- melhor_m2
        melhor_m2 <- melhor_m1
        melhor_m1 <- modelo2
      }
    }
  }
```

```
      }
    }
  }
}
# Desnormalizando as melhores previsões
for (i in 1:num_melhores) {
   prevValid.final[i,] <- denormaliza_comp(melhor_prevValid[i,],
                                           melhor_norm[i],
                                           serie.media.treino,
                                           serie.mult.treino,
                                           serie.max.treino,
                                           serie.min.treino,
                                           serie.sd.treino)
   prevTest.final[i,]  <- denormaliza_comp(melhor_prevTest[i,],
                                           melhor_norm[i],
                                           serie.media.treino,
                                           serie.mult.treino,
                                           serie.max.treino,
                                           serie.min.treino,
                                           serie.sd.treino)
   prevTrain.final[i,] <- denormaliza_comp(melhor_prevTrain[i,],
                                           melhor_norm[i],
                                           serie.media.treino,
                                           serie.mult.treino,
                                           serie.max.treino,
                                           serie.min.treino,
                                           serie.sd.treino)
}

# Combinando as melhores previsões
norm <- 1
# Tamanho da janela para a combinacao = num_melhores
# Preparando os dados de entrada
inputTrain  <- array(0,dim=c(tam.treino,num_melhores))
targetTrain <- array(0,dim=c(tam.treino,1))
inputValid  <- array(0,dim=c(tam.teste,num_melhores))
targetValid <- array(0,dim=c(tam.teste,1))
inputTest   <- array(0,dim=c(tam.teste,num_melhores))
targetTest  <- array(0,dim=c(tam.teste,1))
input2 <- array(0,dim=c((tam.serie.valid-janela.max),num_melhores))
target2 <- array(0,dim=c((tam.serie.valid-janela.max),1))

for (i in 1:num_melhores) {
  # Normalizando as previsões de treino
  melhor_prevValid[i,] <- normaliza_comp(prevValid.final[i,], norm,
                                         serie.media.treino,
                                         serie.mult.treino,
                                         serie.max.treino,
                                         serie.min.treino,
                                         serie.sd.treino)
  melhor_prevTrain[i,] <- normaliza_comp(prevTrain.final[i,], norm,
                                         serie.media.treino,
                                         serie.mult.treino,
                                         serie.max.treino,
                                         serie.min.treino,
                                         serie.sd.treino)
  melhor_prevTest[i,]  <- normaliza_comp(prevTest.final[i,], norm,
                                         serie.media.treino,
                                         serie.mult.treino,
                                         serie.max.treino,
                                         serie.min.treino,
```

```
                                          serie.sd.treino)
}

# As entradas são as transpostas das matrizes de melhores previsões
inputTrain <- t(melhor_prevTrain)
inputValid <- t(melhor_prevValid)
inputTest  <- t(melhor_prevTest)
input2     <- t(melhor_prev2)

for (j in 1:tam.treino) {
  targetTrain[j] <- serie.norm.treino[janela.max+j]
}
for (j in 1:tam.teste) {
  targetValid[j] <- serie.norm[ini.valid+j-1]
  targetTest[j] <- serie.norm[ini.test+j-1]
}
# Target para o treinamento do modelo 2 (inclui a validação)
for (j in 1:(tam.serie.valid - janela.max)) {
  target2[j] <- serie.norm[janela.max+j]
}

# Número de neuronios na camada escondida
for (neuro.ce in neuro.ce.min:neuro.ce.max) {
  # Número máximo de iterações
  for (iter in iter.min:iter.max) {
    # Função de ativação
    for (f in 1:num_f) {
      # Criando o modelo de rede neural para combinar as previsões
      modelo <- mlp_comp(inputTrain, targetTrain, size=neuro.ce,
                         maxit=iter, initFunc="Randomize_Weights",
                         initFuncParams=c(-0.3,0.3), learnFunc="SCG",
                         learnFuncParams=c(0,0,0,0),
                         updateFunc="Topological_Order",
                         updateFuncParams=0.0,
                         hiddenActFunc=func_ativ[f],
                         shufflePatterns=F,linOut=T)
      prevValid.comb <- predict(modelo, inputValid)
      mseValid.comb <- mse_comp(prevValid.comb, targetValid)
      modelo2 <- mlp_comp(input2, target2, size=neuro.ce, maxit=iter,
                          initFunc="Randomize_Weights",
                          initFuncParams=c(-0.3,0.3), learnFunc="SCG",
                          learnFuncParams=c(0,0,0,0),
                          updateFunc="Topological_Order",
                          updateFuncParams=0.0,
                          hiddenActFunc=func_ativ[f],
                          shufflePatterns=F,linOut=T)
      prevTest.comb <- predict(modelo, inputTest)
      mseTest.comb <- mse_comp(prevTest.comb, targetTest)
      mse.comb <- mseValid.comb

      if ((melhor_mse_comb == 0) || (mse.comb < melhor_mse_comb)) {
         melhor_mse_comb <- mse.comb
         melhor_prevValid_comb <- prevValid.comb
         melhor_prevTrain_comb <- modelo$fitted.values
         melhor_prevTest_comb  <- prevTest.comb
         melhor_norm_comb <- norm
         melhor_f_a_comb <- f
         melhor_neuro_comb <- neuro.ce
         melhor_iter_comb <- iter
         melhor_modelo_comb <- modelo
         melhor_m_comb <- modelo2
```

```
        }
      }
    }
  }

  # Denormaliza as previsões combinadas
  prevTrain.comb.final <- denormaliza_comp(melhor_prevTrain_comb,
                                           melhor_norm_comb,
                                           serie.media.treino,
                                           serie.mult.treino,
                                           serie.max.treino,
                                           serie.min.treino,
                                           serie.sd.treino)
  prevValid.comb.final <- denormaliza_comp(melhor_prevValid_comb,
                                           melhor_norm_comb,
                                           serie.media.treino,
                                           serie.mult.treino,
                                           serie.max.treino,
                                           serie.min.treino,
                                           serie.sd.treino)
  prevTest.comb.final  <- denormaliza_comp(melhor_prevTest_comb,
                                           melhor_norm_comb,
                                           serie.media.treino,
                                           serie.mult.treino,
                                           serie.max.treino,
                                           serie.min.treino,
                                           serie.sd.treino)
  lista <- list(validacao=prevValid.final, janela=melhor_j,
                neuronios=melhor_ce, iteracoes=melhor_iter, f
                unc_ativ=melhor_f_a, norm=melhor_norm,
                treino=prevTrain.final, valid.comb=prevValid.comb.final,
                treino.comb=prevTrain.comb.final,
                test.comb=prevTest.comb.final, mse.comb=melhor_mse_comb,
                norm.comb=melhor_norm_comb, func_ativ.comb=melhor_f_a_comb,
                neuronios.comb=melhor_neuro_comb,
                iter.comb=melhor_iter_comb, modelo=melhor_modelo_comb,
                modelo1=melhor_modelo1, modelo2=melhor_modelo2,
                modelo3=melhor_modelo3, modelo4=melhor_modelo4,
                modelo5=melhor_modelo5, modelo2.1=melhor_m1,
                modelo2.2=melhor_m2, modelo2.3=melhor_m3,
                modelo2.4=melhor_m4, modelo2.5=melhor_m5,
                modelo2.comb=melhor_m_comb)
  prev_rna <- lista
}

# Compila função
prev_rna_comp <- cmpfun(prev_rna)
```

```
###
# Função prev_rna_boot
# Faz a previsão RNA com variáveis explicativas para bootstrap
# Decompondo separadamente as amostras de treino, validação e teste
# Retreinando a série incluindo dados de validação
# Escolhendo pela média entre o MSE da validação e o MSE do teste
# Parâmetros:
#     serie: série original completa
#     t.ar: série temporal da temperatura do ar,
#           com defasagens de 2, 3 e 4 meses
#     t.agua: série temporal da temperatura da água
#     janela: vetor com os tamanhos das janelas dos 5 melhores modelos
#     neuronios: vetor com a quantidade de neurônios na camada escondida
#                dos 5 melhores modelos
#     iteracoes: vetor com o número de iterações de treinamento dos 5
#                melhores modelos
#     neuronios.comb: número de neurônios na camada escondida do modelo
#                     de combinação de previsões
#     iterações.comb: número de iterações de treinamento do modelo de
#                     combinação de previsões
#     nr.maxjan: tamanho máximo de janelas testado (para cálculo dos
#                graus de liberdade perdidos)
#     tam.teste: tamanho das amostras de validação e teste
###
prev_rna_boot <- function(serie, t.ar, t.agua, janelas, neuronios,
                          iteracoes, neuronios.comb, iteracoes.comb,
                          rn.maxjan, tam.teste) {

   num_melhores <- 5
   tam.serie <- length(serie)
   tam.janela <- 1
   tam.neuro <- 1
   tam.iter <- 1
   # tam.teste <- round(tam.serie/10)
   janela.max <- rn.maxjan
   tam.treino <- tam.serie - janela.max
   ini.treino <- janela.max + 1
   fim.treino <- ini.treino + tam.treino - 1
   serie.media <- mean(serie)
   serie.max <- max(serie)
   serie.min <- min(serie)
   serie.mult <- max(abs(serie.min-serie.media),serie.max-serie.media)
   serie.sd <- sd(serie)
   num_f <- 1
   num_norm <- 1
   norm <- 1
   normas <- array(1,dim=5)
   prevTrain <- array(0,dim=c(5,tam.treino)) # Previsao de Treinamento
   func_ativ = c("Act_TanH","Act_Signum","Act_Logistic")

   # Normalização
   ar.media <- mean(t.ar[,1])
   ar.max <- max(t.ar[,1])
   ar.min <- min(t.ar[,1])
   ar.mult <- max(abs(ar.min-ar.media),ar.max-ar.media)
   ar.sd <- sd(t.ar[,1])
   agua.media <- mean(t.agua)
   agua.max <- max(t.agua)
   agua.min <- min(t.agua)
   agua.mult <- max(abs(agua.media-agua.min),agua.max-agua.media)
   agua.sd <- sd(t.agua)
```

```
t.ar.2.norm <- normaliza_comp(t.ar[,1], norm, ar.media, ar.mult,
                              ar.max, ar.min, ar.sd)
t.ar.3.norm <- normaliza_comp(t.ar[,2], norm, ar.media, ar.mult,
                              ar.max, ar.min, ar.sd)
t.ar.4.norm <- normaliza_comp(t.ar[,3], norm, ar.media, ar.mult,
                              ar.max, ar.min, ar.sd)
t.agua.norm <- normaliza_comp(t.agua, norm, agua.media, agua.mult,
                              agua.max, agua.min, agua.sd)

serie.norm <- normaliza(serie, 1, serie.media, serie.mult, serie.max,
                        serie.min, serie.sd)

for (i in 1:5) { # Treina os 5 melhores modelos
   j <- janelas[i]
   i.treino <- janela.max-j+1
   f.treino <- janela.max-j+tam.treino
   iiT <- 1
   fiT <- tam.serie-j
   itT <- j+1
   ftT <- tam.serie
   inputTrain<-entrada_comp(serie.norm,iiT,fiT,j)
   targetTrain<-entrada_comp(serie.norm,itT,ftT,1)
   linhas <- length(inputTrain[,1])
   targetTrain <- entrada_comp(serie.norm, j+1, tam.serie, 1)
   lj <- linhas+j
   dadosTrain <- cbind(inputTrain, t.ar.2.norm[j+1:lj],
                       t.ar.3.norm[j+1:lj], t.ar.4.norm[j+1:lj],
                       t.agua.norm[j+1:lj])

   modelo <- mlp_comp(dadosTrain, targetTrain, size=neuronios[i],
                      maxit=iteracoes[i], initFunc="Randomize_Weights",
                      initFuncParams=c(-0.3,0.3), learnFunc="SCG",
                      learnFuncParams=c(0,0,0,0),
                      updateFunc="Topological_Order",
                      updateFuncParams=0.0, hiddenActFunc=func_ativ[1],
                      shufflePatterns=F,linOut=T)
   # obtem os valore previstos de treinamento do modelo
   # Pega somente os últimos "tam.treino" valores, desprezando a janela
   #   inicial
   prevTrain[i,] <- modelo$fitted.values[i.treino:f.treino]
   # Faz a previsão na amostra de validação
   if (i == 1) { modelo1 <- modelo }
   else if (i == 2) { modelo2 <- modelo }
   else if (i == 3) { modelo3 <- modelo }
   else if (i == 4) { modelo4 <- modelo }
   else if (i == 5) { modelo5 <- modelo }
}

# Tamanho da janela para a combinacao = num_melhores
# Preparando os dados de entrada
targetTrain <- array(0,dim=tam.treino)

# As entradas são as transpostas das matrizes de melhores previsões
inputTrain <- t(prevTrain)
previsoes <- inputTrain
for (i in 1:5) {
   previsoes[,i] <- denormaliza(inputTrain[,i], 1, serie.media,
                                serie.mult, serie.max, serie.min,
                                serie.sd)
}
```

```
   for (j in 1:tam.treino) {
      targetTrain[j] <- serie.norm[janela.max+j]
   }
   # Criando o modelo de rede neural para combinar as previsões
   modelo <- mlp_comp(inputTrain, targetTrain, size=neuronios.comb,
                      maxit=iteracoes.comb, initFunc="Randomize_Weights",
                      initFuncParams=c(-0.3,0.3), learnFunc="SCG",
                      learnFuncParams=c(0,0,0,0),
                      updateFunc="Topological_Order",
                      updateFuncParams=0.0, hiddenActFunc=func_ativ[1],
                      shufflePatterns=F,linOut=T)
   prevTrain_comb <- modelo$fitted.values
   modelo.comb <- modelo

   # Denormaliza as previsões combinadas
   prevTrain.comb.final <- denormaliza(prevTrain_comb, 1, serie.media,
                                       serie.mult, serie.max, serie.min,
                                       serie.sd)
   lista <- list(treino.comb=prevTrain.comb.final,
                 janela=janelas, norm=normas, neuronios=neuronios,
                 iteracoes=iteracoes, neuronios.comb=neuronios.comb,
                 iter.comb=iteracoes.comb, norm.comb=1,
                 modelo1=modelo1, modelo2=modelo2, modelo3=modelo3,
                 modelo4=modelo4, modelo5=modelo5, modelo.comb=modelo.comb)
   prev_rna_boot <- lista
}

# Compila função
prev_rna_boot_comp <- cmpfun(prev_rna_boot)
```

```
###
# Função r2.ajustado
# Obtem o valor de r^2 ajustado para os resíduos do modelo informado
###
r2.ajustado <- function(modelo) {
   n<-length(modelo$residuals)
   npar<-length(modelo$coef)
   media.serie<-sum(modelo$x)/n
   soma.res<-0
   soma.total<-0
   for(i in 1:n) {
      soma.res<-soma.res+(modelo$residuals[i]*modelo$residuals[i])
      soma.total<-soma.total + ( (modelo$x[i] - media.serie)
                                 * (modelo$x[i] - media.serie) )
   }
   r2<-1-(soma.res/soma.total)
   r2.ajustado<-1-((n-1)/(n-npar-1))*(1-r2)
}
```

```
###
# Função rmse2
# Obtem o RMSE da serie de residuos informada
###
rmse2 <- function(residuos) {
   n<-length(residuos)
   soma.res <- 0
   for(i in 1:n) {
      soma.res <- soma.res + (residuos[i]*residuos[i])
   }
   media.res <- soma.res/n
   rmse2 <- sqrt(media.res)
}

# Compila a função
rmse2_comp <- cmpfun(rmse2)


###
# Função ruidoBranco
# Testa se um vetor de erros informado como parâmetros é ruído branco
# Se o teste bds aceita H0 (ruido branco) retorna 'Y'
# Caso contrário retonra 'N'
###
ruidoBranco <- function(erro) {

   aceitaBDS <- 'N'
   # Se passou pela FAC e FACP, testa o bds
   b <- bds.test(erro)
   valor.p <- b$p.value
   for (l in 1:2) {
        for (c in 1:4) {
           if (valor.p[l,c] > 0.05) {
                # Encontrou valor que aceita ruído branco (aceita H0)
                aceitaBDS <- 'Y'
           }
        }
   }
   ruidoBranco <- aceitaBDS
}
ruidoBranco_comp <- cmpfun(ruidoBranco)
```

```
###
# Função treina.serie
# Faz o treinamento das redes neurais definidas nos modelos informados,
# para bootstrap
# Parâmetros:
#    serie: série perturbada usada para o treinamento dos modelos
#    t.ar: série com a temperatura do ar
#    t.agua: série com a temperatura da água
#    l_ssa: número de componentes SSA
#    f_filtro: número do filtro wavelet
#    r_tend: nível de decomposição wavelet da tendencia ssa
#    modelos_rna: lista com os modelos rna escolhidos
#    parametros: uma lista com os parâmetros (janela, neuronios, iterações)
#                para cada modelo
#    rn.maxjan: tamanho máximo de janela testado (graus de liberdade
#               perdidos)
#    rn.tam.teste: tamanho das amostras de validação e de teste
#    mod_rna é uma lista com os seguintes grupos:
#        Apr1 da tendencia SSA
#        Det1 da tendencia SSA
#        Apr2 da tendencia SSA
#        Det2 da tendencia SSA
#        Apr3 da tendencia SSA
#        Det3 da tendencia SSA
#
#    Cada grupo é composto por 13 posições:
#        os 5 melhores modelos de previsão das componentes
#        o modelo da combinação das melhores componentes
#        um vetor com as 5 melhores normas
#        um vetor com as 5 melhores janelas
#        um vetor com os 5 melhores numeros de neurônios
#        um vetor com as 5 melhores iterações
#        a norma usada na combinação
#        a quantidade de neurônios usadas na combinação
#        a quantidade de iterações usadas na combinação
###

treina.serie <- function(serie, t.ar, t.agua, l_ssa, f_filtro, r_tend,
                         modelos_rna, parametros, rn.maxjan, rn.tam.teste)
{
   offset_modelo1 <- 1
   offset_modelo2 <- 2
   offset_modelo3 <- 3
   offset_modelo4 <- 4
   offset_modelo5 <- 5
   offset_modelo_comb <- 6
   offset_norma <- 7
   offset_janela <- 2
   offset_neuronios <- 3
   offset_iteracoes <- 4
   offset_norma_comb <- 5
   offset_neuronios_comb <- 6

   # ARIMAX
   n<-length(serie)
   tll <- rn.maxjan # Graus de liberdade perdidos.
   serie.tll <- array(0,dim=n-tll)
   for (i in 1:(n-tll)) {
      serie.tll[i] <- serie[tll+i]
   }
   tam.prev <- n-tll
```

```
# SSA: l ótimo, inteiro: l_ssa

# Número de tentativas SSA, para definir a quantidade de vetores
# (colunas da matriz)
num.ssa <- 1
tend.ssa <- array(0, dim=n)
# Especificação de filtros wavelets
filtros=c("haar","d4","d8","d16","mb4","mb8","mb16","mb24","bs3.1",
          "fk4","fk6","fk8","fk14","fk22","la8","la16","la20",
          "bl14","bl20")
nfiltros=1
# Matriz bidimensional para armazenar as decomposições wavelets
# Primeiro índice = Número de componentes Wavelet
# (6 tendencia (3 aprox + 3 detalhe))
# Segundo índice = leitura (1 a n)
rn <- array(0, dim=c(6,tam.prev))
# Matriz tridimensional para armazenar as previsões e erros de rn
# Primeiro índice = Nível da decomposição Wavelet da tendência SSA
#                   (1 a 6)
# Terceiro índice = Valores (1 a n)
prev <- array(0, dim=tam.prev)
erro <- array(0, dim=tam.prev)
melhor_prev <- array(0, n)
melhor_erro <- array(0, n)
# Parametros de melhor resultado
melhor_rn_mod <- vector("list",12)

# Filtragem SSA
dec.ssa <- ssa(serie, L=l_ssa)
grupos <- grouping.auto(dec.ssa, grouping.method = "wcor",
                        groups=1:l_ssa, nclus=4)
tendencia <- reconstruct(dec.ssa,groups=list(c(grupos$'1', grupos$'2',
                                               grupos$'3')))
tend.ssa <- tendencia$F1

# Decomposição Wavelet
# comp = matriz de componentes Wavelet
comp <- array(0,dim=c(n,6))
for (r in 1:r_tend) {
   tend<-mra(tend.ssa, wf=filtros[f_filtro], J=r)
   # Tendencia SSA, Aproximação Wavelet de nível r
   comp[,2*r-1] <- unlist(tend[r+1])
   # Tendencia SSA, Detalhe Wavelet de nível r
   comp[,2*r] <- unlist(tend[r])
}
rn_mod <- list()
rn_mod2 <- list()
for (h in 1:6) {
   componente <- comp[,h]
   # Indice dos modelos da aproximação de nível r
   componente_h <- (h-1)*6
   componente_par <- (h-1)*7
   # Separa as normas dos melhores modelos e a norma combinada do modelo
   modelo1        <- modelos_rna[[offset_modelo1+componente_h]]
   modelo2        <- modelos_rna[[offset_modelo2+componente_h]]
   modelo3        <- modelos_rna[[offset_modelo3+componente_h]]
   modelo4        <- modelos_rna[[offset_modelo4+componente_h]]
   modelo5        <- modelos_rna[[offset_modelo5+componente_h]]
   modelo.comb    <- modelos_rna[[offset_modelo_comb+componente_h]]
   janelas        <- parametros[[offset_janela+componente_par]]
```

```
      neuronios      <- parametros[[offset_neuronios+componente_par]]
      neuronios.comb <- parametros[[offset_neuronios_comb+componente_par]]
      iteracoes      <- parametros[[offset_iteracoes+componente_par]]
      iteracoes.comb <- 15

      p_rna <- prev_rna_boot11_comp(componente, t.ar, t.agua, janelas,
                                    neuronios, iteracoes, neuronios.comb,
                                    iteracoes.comb, rn.maxjan, rn.tam.teste)

      rn[h,] = p_rna$treino.comb
      mod_comp <- list()
      mod_comp <- append(mod_comp, list(mod1=p_rna$modelo1))
      mod_comp <- append(mod_comp, list(mod2=p_rna$modelo2))
      mod_comp <- append(mod_comp, list(mod3=p_rna$modelo3))
      mod_comp <- append(mod_comp, list(mod4=p_rna$modelo4))
      mod_comp <- append(mod_comp, list(mod5=p_rna$modelo5))
      mod_comp <- append(mod_comp, list(mod.comb=p_rna$modelo.comb))
      mod_comp <- append(mod_comp, list(norm=p_rna$norm,
                                        janela=p_rna$janela,
                                        neuronios=p_rna$neuronios,
                                        iteracoes=p_rna$iteracoes,
                                        norm.comb=p_rna$norm.comb,
                                        neuronios.comb=p_rna$neuronios.comb,
                                        serie.treino=serie))
      rn_mod <- append(rn_mod, mod_comp)

      # Modelo2: inclui os dados de validação
      mod2_comp <- list()
      mod2_comp <- append(mod2_comp, list(mod1=p_rna$modelo1))
      mod2_comp <- append(mod2_comp, list(mod2=p_rna$modelo2))
      mod2_comp <- append(mod2_comp, list(mod3=p_rna$modelo3))
      mod2_comp <- append(mod2_comp, list(mod4=p_rna$modelo4))
      mod2_comp <- append(mod2_comp, list(mod5=p_rna$modelo5))
      mod2_comp <- append(mod2_comp, list(mod.comb=p_rna$modelo.comb))
      rn_mod2 <- append(rn_mod2, mod2_comp)
   }

   # Inserir as previsões através de Redes Neurais
   # A previsão é a soma das componentes de aproximação
   # e de detalhes de nível r (tendencia)
   # Soma a componente de aproximação
   prev <- rn[(2*r_tend)-1,]
   for (k in 1:r_tend) { # Soma as componentes de detalhes
      prev <- prev + rn[(2*k),]
   }
   # O erro de previsão é a diferença entre a previsão e a serie original
   erro <- serie.tll - prev

   lista <- list(prev = prev, erro = erro, rn = rn, modelo_rn=rn_mod,
                 modelo2_rn=rn_mod2)
   treina.serie <- lista
}

treina.serie_comp <- cmpfun(treina.serie)
```