Universidade Federal de Itajubá
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Nelson Dallocchio

Modelagem Orientada a Objetos de Estimadores de Estados Aplicados à Supervisão da Segurança Operativa de Redes Elétricas

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do título de Mestre em Ciências em Engenharia Elétrica.

Área de Concentração: Sistema Elétricos de Potência

Orientador: Prof. Dr. Robson Celso Pires

Setembro 2007
Itajubá - MG

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Margareth Ribeiro- CRB_6/1700

D147m

Dallocchio, Nelson

Modelagem orientada a objetos de estimadores de estados aplicados à supervisão da segurança / Nelson Dallocchio. -- Itajubá,(MG) : [s.n.], 2007.

225 p. : il.

Orientador: Prof. Dr. Robson Celso Pires.

Dissertação (Mestrado) – Universidade Federal de Itajubá.

1. Estimação de estados robusta. 2. Modelagem orientada a objetos. 3. Supervisão da rede operativa de redes elétricas. I. Pires, Robson Celso, orient. II. Universidade Federal de Itajubá. III. Título.

CDU 621.3:004.045(043)

Ministério da Educação
UNIVERSIDADE FEDERAL DE ITAJUBÁ
Criada pela Lei nº 10.435, de 24 de abril de 2002

ANEXO II

FOLHA DE JULGAMENTO DA BANCA EXAMINADORA

Título da Dissertação: **"Modelagem Orientada a Objetos de Estimadores de Estados Aplicados à Supervisão da Segurança Operativa de Redes Elétricas"**

Autor: **Nelson Dallocchio**

JULGAMENTO

Examinadores	Conceito A = Aprovado R = Reprovado	Rubrica
1º	A	
2º	A	
3º	A	
4º	A	

Observações:

1) O Trabalho será considerado Aprovado se todos os Examinadores atribuírem conceito A.
2) O Trabalho será considerado Reprovado se forem atribuídos pelos menos 2 conceitos R.
3) O Trabalho será considerado Insuficiente (I) se for atribuído pelo menos um conceito R. Neste caso o candidato deverá apresentar novo trabalho. A banca deve definir como avaliar a nova versão da Dissertação.

Resultado Final: Conceito: A, ou seja, Aprovado

Observações: _____

Itajubá, 28 de setembro de 2007.

Prof. Dr. Antonio José Alves Simões Costa
1º Examinador - UFSC

Prof. Dr. Alessandro Manzoni
2º Examinador – UFRJ

Prof. Dr. Armando Martins Leite da Silva
3º Examinador - UNIFEI

Prof. Dr. Robson Celso Pires
4º Examinador – UNIFEI - (Orientador)

Pró-Reitoria de Pesquisa e Pós-Graduação - Av . BPS, 1303 - Caixa Postal 50 - 37500-903 - ITAJUBÁ/MG – BRASIL
Tel.: (35) 3629-1118 - 3629-1121 - Fax (35) 3629-1120 - posgrad@unifei.edu.br

AGRADECIMENTOS

- *À CAPES pelo apoio recebido em forma de bolsa de estudos, a qual foi indispensável para a realização deste trabalho.*

- *À CEEE pelo apoio financeiro aos equipamentos necessários ao desenvolvimento deste projeto e pelo fornecimento dos softwares e informações necessários.*

- *Ao Prof. Robson Celso Pires pela orientação exemplar deste trabalho, pelo gerenciamento do projeto de pesquisa e desenvolvimento e pelo constante apoio fornecido perante às minhas dificuldades pessoais.*

- *Aos colegas do Grupo de Engenharia de Sistemas da Universidade Federal de Itajubá pelos tempos de convivência durante meus últimos anos em Itajubá.*

- *Ao amigo Deyvid L.S.Marques pela constante ajuda durante meus períodos de afastamento.*

- *À minha família pela compreensão nos momentos de ausência.*

- *Aos profissionais da CEEE pelas informação históricas prestadas.*

- *À Universidade Federal de Itajubá pela formação acadêmica.*

- *Aos colegas de Receita Federal do Brasil pela compreensão dos difíceis momentos vividos durante a escrita deste trabalho. Em especial a Jonas Bernardino Santana e Erinaldo Arruda Silva.*

ÍNDICE GERAL

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A utilização de hardware avançado e computadores digitais em centros de operação de sistemas elétricos potencializam as funções de supervisão da segurança e controle operativo em tempo real. Estas funções permitem operar os sistemas elétricos de forma segura, o que possibilita ao operador tomar decisões mais efetivas. Especificamente, os Estimadores de Estados em Sistemas de Potência têm um importante papel somado aos sistemas de aquisição de dados no sentido de avaliar os estados da rede o mais próximo possível dos estados verdadeiros do sistema supervisionado.

Este trabalho verifica experimentalmente o desempenho de um estimador de estados robusto escrito em linguagem de programação orientada a objetos. O programa faz uso do Método de Mínimos Quadrados com Ponderação Variável (MQPV) e Restrições de Igualdade (RI). A metodologia empregada tem a característica de executar o processamento de erros grosseiros durante o processo iterativo de busca de solução do problema, mesmo quando é estendida ao problema de estimação de estados com restrições de igualdade. Neste trabalho, as restrições de Igualdade podem ser processadas através de duas abordagens: a) Método Clássico dos Pesos (MP); b) Método dos Pesos com Refinamento Iterativo (MPRI). Independentemente das medidas e/ou pseudo-medidas terem sido classificadas como pontos de alavancamento, a metodologia proposta, aqui denominada MQPV/RI, permite processar erros grosseiros e/ou topológicos satisfatoriamente.

A operacionalização dos algoritmos utilizados é explicitamente demonstrada ao longo do processo iterativo de solução de um sistema exemplo de 5 barras. Além disso, a metodologia também é testada no sistema de uma Companhia de Energia Elétrica do Sul do Brasil contendo 126 barras, além de sistemas realísticos de 340, 730 e 1916 barras. Os resultados produzidos são comparados, analisados e comentados detalhadamente no capítulo que descreve o conjunto de simulações realizadas nos sistemas testes.

ABSTRACT

The use of advanced hardware and digital computers in electrical management systems potentially enhance the supervision and control tasks in real time operation mode. These functions allow the safe operation of electrical networks, which effectively improve the operator decision. Specifically, the state estimators applied to power systems have an important role along with the SCADA systems, in order to evaluate the network states as close as possible to the actual states of the monitored system.

This work evaluates a robust state estimator algorithm performance when it is written in object-oriented programming language. The program makes use of the Iterative Weighted Least-squares method (IWLS) with equality constraints (EC). The employed methodology has the feature of processing bad data during the iterative search for the problem solution, even when it is extended to the state estimation problem with equality constraints. In this work, the equality constraints can be modeled through two approaches: a) Classical Weighting Method (WM); b) Weighting Method with Iterative Refinement (WMIR). Independently from the fact that regular and/or pseudo measurements have been classified as leverage points, the employed methodology, named IWLS-EC, allows the satisfactory processing of bad data and/or topology errors.

The algorithm performances are *explicitly* tracked throughout the iterations carried out on a small example system of 5 buses. Moreover, the proposed methodology is also evaluated in an actual system in the south of Brazil with 126 buses, and also in other realistic systems with 340, 730 and 1916 buses. The obtained results are compared, analyzed and commented on in the chapter that describes the set of simulations carried out on test systems.

Capítulo 1

Introdução

1.1 Considerações Gerais

A estimação de estados aplicada a sistemas elétricos de potência (EESP) foi inicialmente proposta com o objetivo de mitigar o risco de operações indevidas geradas por informações errôneas obtidas através do sistema de telemetria. Até sua proposição e utilização, as únicas informações operativas disponíveis para os operadores eram as mensurações de grandezas do sistema e os estados operativos dos equipamentos de manobra. Erros neste conjunto de informações poderiam resultar em atuações indesejadas sobre o sistema, gerando deste o fornecimento de energia fora dos padrões de qualidade estabelecidos até danos permanentes em equipamentos e componentes da rede elétrica ou, até mesmo, queda de todo o sistema elétrico interligado.

A estimação de estados aplicada a sistemas elétricos de potência tornou-se, desde sua proposição inicial no final da década de 60 [92], numa das mais importantes áreas de pesquisas relacionadas à operação sistêmica. Ao longo do tempo, novas técnicas, estudos e métodos foram propostos e desenvolvidos com o intuito de solucionar problemas afetos à EESP. A literatura técnica possui um variado conjunto de pesquisas que formam suas diversas subáreas, dentre as quais destacam-se: a identificação e o tratamento de erros; a análise da capacidade de estimação dos estados do sistema (observabilidade); a utilização de restrições de igualdade na formulação do problema de EESP; e a utilização métodos e técnicas matemáticas aplicados à EESP.

Apesar de existirem diferentes metodologias aplicadas à EESP, o estimador de estados baseado em mínimos quadrados ponderados com identificação de erros grosseiros através de testes de hipóteses se popularizou nos centros de operação de sistemas em todo mundo – tornando-se o estimador-padrão da área. Isto se deve tanto pela qualidade de seus resultados quanto pela comprovação de sua eficácia durante anos de estudos e aplicações. Porém isto não significa que ele seja extremamente eficiente no ponto de vista computacional ou que esteja imune a problemas de detecção e identificação de alguns tipos de erros, mesmo em casos onde

haja redundância de informações operativas da rede elétrica [91]. A existência de múltiplos erros grosseiros, caracterizados como aqueles com magnitude superior a 3 desvios-padrão, resulta num significativo aumento do esforço computacional quando seu método de processamento se restrinja a uma única identificação por vez. Ademais, haverá falha na detecção de tais erros quando estes ocorrerem em medidas formadoras de *pontos de alavancamento* [91]. A literatura técnica descreve métodos e técnicas que permitem identificar ou reduzir o efeito de erros grosseiros situados em tais medidas [55-91].

Novos métodos de estimação de estados utilizando-se estimadores não-quadráticos e índices de projeção estatísticas [91] possibilitaram abordar tanto o problema dos efeitos nocivos da existência de medidas formadoras de *pontos de alavancamento* quanto minimizar os efeitos da presença de erros no conjunto de medidas através do uso ponderações variáveis. A aplicação deste método, denominado de estimação de estados robusta baseada mínimos quadrados com ponderação variável (MQPV), possibilita que o conjunto de medidas contaminadas com erros grosseiros sejam identificadas – independentemente da medida contaminada ser formadora de ponto de alavancamento – e devidamente penalizadas, reduzindo-se drasticamente os seus efeitos sobre os estados estimados para o sistema. Portanto os estados resultantes deste método de estimação podem ser considerados válidos e, como subproduto, todas as medidas contaminadas com erros grosseiros são identificadas ao término do processo iterativo de solução. A robustez do referido método passou a abranger a solução numérica do problema de estimação de estados quando utilizou-se as rotações ortogonais de Givens como método de fatoração aplicado à solução problema [102], a estabilidade numérica deste método de fatoração, proposto inicialmente para EESP em [63], possibilita minimizar o mau condicionamento numérico da matriz de coeficientes do sistema linear que compõem a solução do problema. Contudo formulou-se uma metodologia robusta, mas que ainda carece de uma melhor análise quando aplicada a sistemas elétricos de potência realísticos.

Alguns problemas relativos a supervisão operativa de sistema do transmissão de energia elétrica do Estado do Rio Grande do Sul gerou o interesse da Companhia Estadual de Energia Elétrica do Rio Grande do Sul (CEEE) na pesquisa e no desenvolvimento de estimadores de estados que utilizassem novas metodologias diferentes daquela aplicada ao seu sistema. Logo surgiu a oportunidade de se aplicar o estimador de estados robusto baseado em MQPV formulado pelos trabalho [91] e [102] em sistemas realísticos e estudar seu comportamento. Entretanto, como parte natural do processo de pesquisa e desenvolvimento, o estimador de estados resultante deveria ser compatível com a seqüência lógica e operacional do centro de

operação de sistema da CEEE. Portanto o sistema resultante deveria possuir todos os requisitos necessários a integração com o sistema de aquisição de dados da CEEE, trabalhando de forma automatizada e integrada.

Por anos o desenvolvimento de funções e ferramentas computacionais voltadas a sistemas de energia elétrica (SEE) foram desenvolvidos através de linguagens científicas, mais especificamente a linguagem de programação FORTRAN. Tal uso se deve principalmente aos recursos matemáticos necessários a implementação destas funções e ferramentas que são disponibilizadas por linguagens científicas. Entretanto, ao longo das últimas décadas, novos conceitos de programação foram desenvolvidos e aplicados a diversos tipos de problemas computacionais. A necessidade de uma metodologia de programação que pudesse lidar com a complexidade e a grandiosidade dos novos sistemas e mitigar as incertezas sobre o seu processo de desenvolvimento, as quais colocaram em crise todo o mercado de desenvolvimento de sistemas durante a década de 70, fez surgir um novo conceito em programação conhecido como programação orientada a objetos. Durante anos, os sistemas voltados para SEE ignoram a sua existência tanto por desconhecimento quanto pela idéia de perda de eficiência associada a este método de programação. Novas aplicações em SEE utilizaram a modelagem e a programação orientada a objetos [14-39], onde grande parte destes trabalhos concluíram que a ineficiência computacional esta diretamente relacionada com as práticas computacionais adotadas.

1.2 Objetivos

O presente trabalho tem o objetivo desenvolver e implementar um estimador de estados robusto baseado em mínimos quadrados com ponderação variável utilizando-se a modelagem orientada a objetos (MOO). Para tanto serão desenvolvidos um modelo de sistema elétrico de potência (SEP) para regime permanente, ferramentas matemáticas e um estimador de estados utilizando a modelagem orientada a objetos. Tal modelo deverá ser integrável à seqüência lógica e operacional do centro de operações de sistemas (COS) da CEEE, devendo possuir uma estrutura capaz de relacionar as informações do sistema de aquisição de dados aos componentes da rede elétrica e de efetuar manobras primárias entre os componentes de rede para configurar o modelo de SEP conforme o estado operativo do sistema de transmissão.

As ferramentas matemáticas deverão ser computacionalmente eficientes e possuir classes capazes de lidar adequadamente com grandes vetores e matrizes esparsos. Também deverão

implementar métodos de fatoração baseados em transformações ortogonais, mas não limitando-se a estas, e métodos de ordenação de linhas e colunas.

O estimador de estados deverá ser desenvolvido numa base que proporcione fácil manutenção e evolução, sendo capaz de implementar não só a metodologia a ser desenvolvida, mas também outra metodologia relatada pela literatura técnica. Por fim o estimador de estados deverá ser integrável à seqüência lógica e operacional do centro de operações da CEEE e deverá ter seus resultados registrados e analisados.

1.2 Estrutura de apresentação

No Capítulo 2, os conceitos de engenharia de software serão introduzidos para fins de registros das práticas utilizadas no desenvolvimento do projeto. Aconselha-se a leitura do referido capítulo a pessoas que possuam interesse em métodos de desenvolvimento de sistemas e não conheçam as práticas e as metodologias envolvidas.

No Capitulo 3, a modelagem orientada a objetos será detalhada, apresentando-se os conceitos da modelagem e da programação orientadas a objetos e a linguagem unificada de modelamento (*Unified Modeling Language* - *UML*). Os conceitos abordados neste capítulo serão utilizados durante dos os demais capítulos do trabalho.

No Capitulo 4, os conjuntos de modelos orientados a objetos que formam as diferentes estruturas de classes que compõem o projeto de software deste trabalho serão apresentados. Inicialmente serão apresentadas as revisões bibliográficas e a arquitetura básica do projeto. Posteriormente são detalhados dois diferentes núcleos que compõem o modelo de SEP proposto pelo projeto. Por sua vez, as teorias e classes que formam as ferramentas matemáticas serão introduzidas conjuntamente. Por fim, a teoria e a estrutura de classes dos estimadores de estados desenvolvidos no âmbito deste trabalho serão apresentadas.

No Capítulo 5, os resultados numéricos dos testes realizados com as ferramentas matemáticas serão mostrados e comparados. Posteriormente, o conjunto de testes utilizando-se os estimadores de estados desenvolvidos sobre sistemas testes será realizado terão seus resultados apresentados detalhadamente. Por fim, serão apresentadas as experiências relativas à aplicação dos estimadores de estados desenvolvidos no sistema de transmissão da CEEE.

Finalmente, no Capítulo 6, apresentam-se as principais conclusões do trabalho desenvolvido.

Capítulo 2

A Engenharia de Software

2.1 Considerações Gerais

A evolução do hardware e o desenvolvimento de novas técnicas computacionais permitiram a aplicação da computação a uma extensa área de conhecimento humano. Os sistemas computacionais se tornaram cada vez mais complexos, ganharam, gradativamente, importância e se mostraram imprescindíveis para diversas atividades. Sua crescente utilização resultou, nas últimas décadas, numa significativa melhoria nos procedimentos e métodos aplicados ao processo de desenvolvimento de software, tendo como motivação a dificuldade de manutenção, a falta de confiabilidade, desempenho inferior ao esperado, a falta de uma previsão confiável dos custos e prazos, dentre outros problemas encontrados pela indústria do software na implementação de grandes sistemas.

Um sistema computacional não se limita ao programa de computador, este também abrange toda documentação de seu projeto, toda a informação necessária para sua correta execução e todo fundamento teórico que embasou seu desenvolvimento. Portanto, o software que possui *valor* é aquele que engloba, dentro da possibilidade técnica, todos os aspectos supracitados. Logo, para se obter êxito no desenvolvimento, é aconselhável a utilização de um conjunto de procedimentos que abrange desde a especificação até a manutenção do sistema, conhecido como *processo de software*.

A engenharia de software é a disciplina responsável por toda parte produtiva de um sistema computacional, esta se ocupa em estudar os diferentes processos de software e seus respectivos efeitos no produto final. Não é o objetivo deste documento o detalhamento da engenharia de software, porém um estudo resumido se faz necessário para se obter um entendimento mais conciso dos conceitos aplicados neste trabalho. Para tanto, as próximas seções deste capítulo se focarão nos conceitos básicos de engenharia de software. Estudos mais aprofundados sobre os conceitos de engenharia de software podem ser encontrados em [10].

2.2 Engenharia de Sistemas com Base em Computadores

Existe uma relação direta entre a engenharia de software e a engenharia de sistemas, uma vez que muitos dos requisitos e problemas que o desenvolvimento de um software visa suprir têm origem nas decisões tomadas no projeto do sistema. Um exemplo para tal questão seria o software que faz a aquisição de dados de um sistema de telemetria, os protocolos implementados nos software dependem das características dos diversos equipamentos que constituem o sistema de telemetria. Contudo existe uma estreita relação entre a engenharia de sistemas e a engenharia de software, tendo em vista que grande parte do processo de software possui fundamento na engenharia de sistemas. Assim, muitos dos conceitos utilizados em engenharia de software se originaram na engenharia de sistemas [10]. Especificar, projetar, implementar, validar e manter sistemas como um todo são atividades pertinentes à engenharia de sistemas.

Genericamente, um sistema é um grupo de componentes inter-relacionados, que possuem funções específicas e trabalham conjuntamente para atingir um objetivo. Os componentes podem ser lógicos (software) ou físicos (hardware), as inter-relações incluem as interações entre o sistema, o usuário e o ambiente. Portanto um sistema com base em computadores envolve uma série de atividades que extrapolam as atribuições dadas ao software. Seu processo de desenvolvimento não apenas considera os aspectos relativos às funcionalidades específicas, mas todo conjunto de recursos físicos, humanos e ambientais envolvidos.

Sistemas mais complexos, geralmente, são constituídos através do uso de sistemas independentes. Tais sistemas são conhecidos como *subsistemas* e fornecem uma ou mais funcionalidades (serviços) para os sistemas que o englobam. O sistema de bancos de dados de um sistema de gerenciamento de energia – que integra os sistemas utilizados em toda a empresa de energia elétrica – é um exemplo de subsistema, uma vez que ele pode ser utilizado tanto pelas ferramentas computacionais aplicadas ao planejamento como pelas ferramentas aplicadas à operação. Cada ferramenta tem uma necessidade específica quanto à informação necessária, portanto o sistema de banco de dados é projetado para suprir às informações requisitadas e se adaptar quando um novo tipo de informação tiver que ser armazenado.

2.2.1 Propriedades emergentes

Algumas propriedades dos sistemas são de difícil previsão, ou até mesmo impossíveis de serem previstas, devido às complexas inter-relações de seus componentes. Tais

propriedades somente emergem quando o sistema estiver operacional, uma vez que o sistema não é simplesmente a soma de suas partes. Esse tipo de propriedade é conhecido como propriedades emergentes [10] e são classificadas em dois tipos:

- ✓ *Propriedades emergentes funcionais*: São obtidas somente quando o sistema trabalhar como um todo para atingir seu objetivo. A estimação de estados em sistemas elétricos de potência (SEP), por exemplo, só é efetivamente realizada com o auxilio do sistema de aquisição de dados (ou do sistema de banco de dados) e do configurador de redes.

- ✓ *Propriedades emergentes não funcionais*: São relacionadas com o comportamento do sistema em seu ambiente operacional. A confiabilidade, o desempenho e a segurança dos sistemas são exemplos de propriedades não funcionais. O desempenho, como um todo, dificilmente pode ser previsto pelo projetista do estimador de estados, uma vez que este não possui informações sobre o desempenho do configurador, do sistema de aquisição de dados e do esforço computacional despendido em suas inter-relações.

2.2.2 O ambiente computacional

O ambiente computacional é coleção de sistemas que interagem entre si para atingir objetivos diversos. Muitas vezes o ambiente computacional é considerado um sistema como um todo, porém, dentro de um ambiente computacional, podem existir sistemas que possuem finalidades diversas e não interagem entre si. O funcionamento, a confiabilidade e o desempenho dos diversos sistemas dependem do ambiente computacional a qual estão inseridos. A rede de computadores e sua organização são exemplos de componentes do ambiente computacional que impactam diretamente o funcionamento de um sistema. A quantidade, as restrições de acesso e a posição de servidores de banco de dados podem gerar um sistema mais ou menos robusto, eficiente e confiável.

O ambiente computacional depende diretamente do ambiente organizacional no qual este foi desenvolvido. As questões políticas, econômicas e funcionais afetam o planejamento e a evolução do ambiente computacional, portanto, indiretamente, elas também afetam a projeto de um dado sistema. As mudanças no processo, nas tarefas realizadas e na organização de uma

dada entidade, provocam mudanças no ambiente computacional e nos sistemas que o compõem.

2.2.3 Modelagem de sistemas

Um modelo é representação, geralmente gráfica, dos componentes de um sistema e suas respectivas relações com o objetivo de fornecer as informações necessárias para uma compreensão rápida daquilo que esta sendo projetado. O nível de detalhamento e o tipo de um modelo dependem das informações que se deseja representar, do resultado que se pretende atingir e do nível de informação que se pretende fornecer ao leitor. Os modelos mais simples – como os modelos de arquitetura – visam fornecer uma visão geral do sistema a pessoas que, normalmente, não precisam de um conhecimento mais detalhados (como gerentes e diretores).

Os modelos fazem parte do projeto do sistema e cada tipo de modelo visa fornecer um conjunto específico de informações para um determinado grupo de pessoas envolvidas com o projeto. Existem modelos que detalham a estrutura de um componente, modelos que mostram o comportamento de um ou mais componentes, outros que mostram a interação entre o sistema e seus usuários. Os principais modelos, dentro da engenharia de software, serão comentados nas próximas seções deste capítulo. Por hora, este trabalho limitar-se-á a apresentar o modelo de arquitetura.

O modelo de arquitetura do sistema é uma representação simplificada dos subsistemas e dos principais componentes e suas respectivas relações. Normalmente, o modelo de arquitetura é representado por um diagrama de blocos, onde cada retângulo representa um subsistema/componente e cada seta/flecha que interliga os retângulos representa um tipo de relação. Uma relação pode representar um fluxo de dados, uma utilização ou qualquer outro tipo de dependência. A Figura 1 exemplifica o modelo de arquitetura de sistema com base num sistema computacional para centros de operações de SEP.

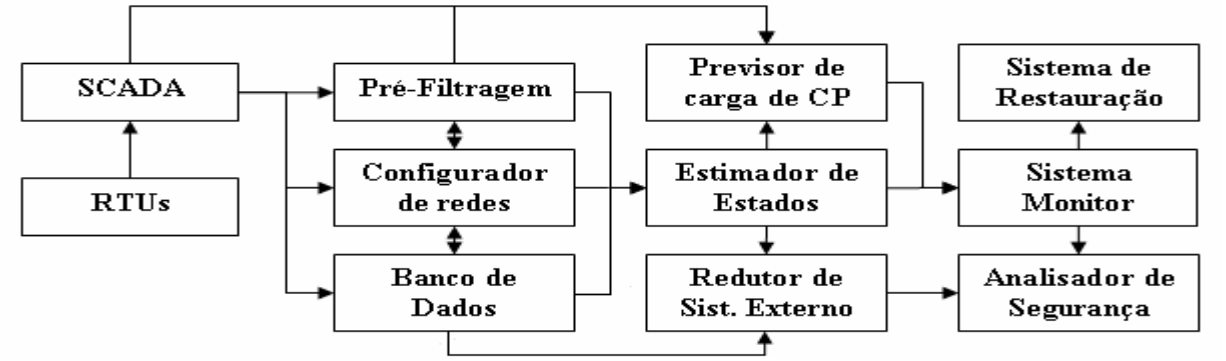


Figura 1 – Arquitetura de um sistema computacional para centros de operações de SEP

2.2.4 O processo de engenharia de sistemas

Todo sistema possui um ciclo de vida, o qual compreende desde a especificação do sistema até a sua desativação (quando for o caso). As fases, que compreendem o processo de engenharia de sistemas, mostradas na Figura 2 Figura 2 , definem a ordem das atividades que devem ser executadas.

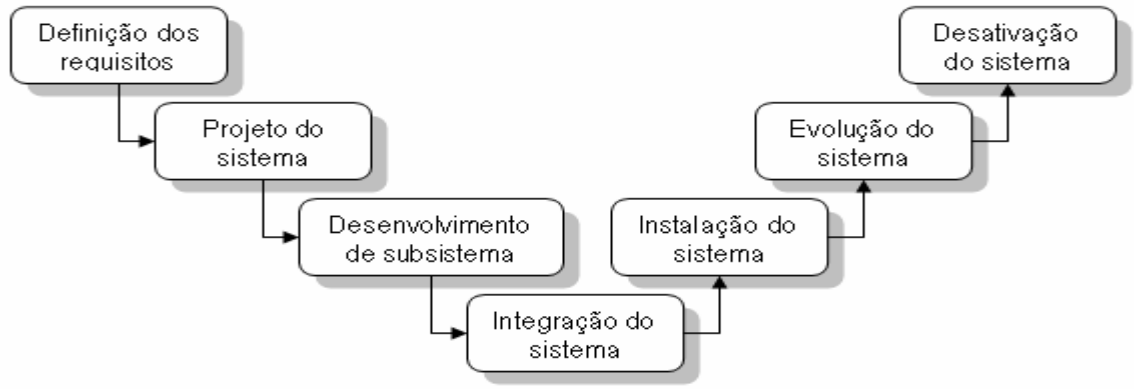


Figura 2 – As fases do processo de engenharia de sistema

O processo de engenharia de sistema influenciou o modelo *em cascata* – que será discutido na próxima seção – do processo de software, porém cabe ressaltar algumas importantes distinções entre o processo de desenvolvimento de software e o processo de engenharia de sistemas:

✓ *Envolvimento Interdisciplinar*: A engenharia de sistemas possui um forte envolvimento interdisciplinar devido às peculiaridades do sistema. Diferentes áreas da engenharia podem estar envolvidas no processo de engenharia, o que não ocorre com a mesma intensidade e relevância dentro dos processos de software.

✓ *Flexibilidade reduzida durante o desenvolvimento*: Depois de especificado, projetado e parcialmente desenvolvido, um sistema possui um custo elevado para atender a modificações em alguns requisitos. No desenvolvimento de software, há uma mitigação deste custo – haja vista a flexibilidade do produto. Devido a essa característica, muitos problemas de mudanças de requisitos dentro da engenharia de sistemas são solucionados com o desenvolvimento de um software.

2.2.4.1 Definição dos requisitos

O processo de levantamento de requisitos consiste em consultas a clientes, usuários finais e consultores especializados, além de levantamentos da literatura técnica e consultas aos projetistas de sistemas pilotos e pesquisadores no caso de sistemas com novas tecnologias originadas de pesquisas. Contudo, o levantamento de requisitos concentra-se na definição do conjunto de requisitos que o sistema deve atender, os quais são, normalmente, classificados em três tipos [10]:

- ✓ *Requisitos funcionais abstratos*: São as funções básicas que o sistema deve fornecer, definidas de forma abstrata, sem a necessidade de um detalhamento mais aprofundado – uma vez que esse detalhamento ocorre na fase de projeto do sistema.
- ✓ *Propriedades de sistemas*: São as propriedades emergentes não funcionais, como a confiabilidade, desempenho e segurança. Estas propriedades são definidas previamente e o projeto deve ser desenvolvido de forma a garantir que as mesmas sejam obtidas dentro de uma margem de segurança.
- ✓ *Características não desejadas*: Tão importante quanto definir aquilo que o sistema deve apresentar é definir aquilo que o sistema não deve apresentar. Obviamente, dentro do senso técnico, devem-se definir as características não desejadas e passíveis de serem encontradas no sistema.

2.2.4.2 Projeto de sistemas

Projeto de sistemas consiste na definição de quais são os componentes/subsistemas que integram o sistema e quais são as funcionalidades oferecidas por cada componente/subsistema. Geralmente, o projeto de sistema fornece um conjunto de modelos e documentos em diferentes níveis de detalhamento, os quais instruirão as equipes de cada área envolvida no projeto durante a etapa de desenvolvimento dos componentes/subsistemas. A Figura 3 mostra as atividades desenvolvidas no projeto de sistemas. Note que as atividades não possuem uma ordem cronológica rígida, uma vez que é possível haver um retorno a atividades pretéritas devido aos resultados de problemas e decisões em etapas futuras.

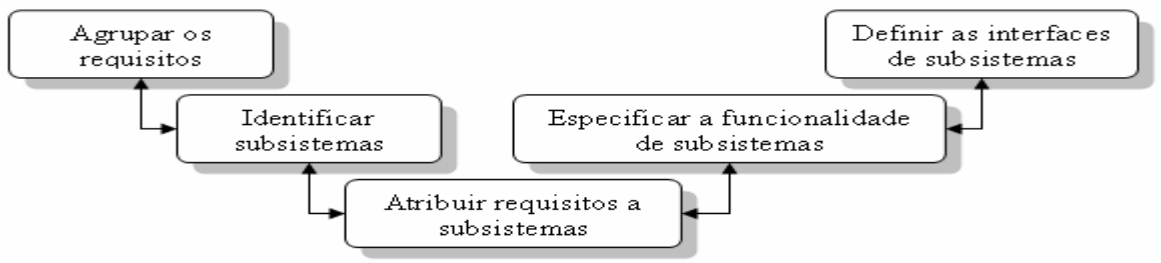


Figura 3 – Atividades envolvidas no projeto de sistemas

- *Agrupar os requisitos*: Os requisitos que possuem características comuns, relações ou interdependência são agrupados após uma análise cuidadosa de cada um. Existem diferentes possibilidades de agrupamento, portanto a análise deve considerar a agrupamento mais conveniente para os objetivos do sistema.

- *Identificar subsistemas*: Esta atividade pode ser fundida com a anterior se houver possibilidade, uma vez que os subsistemas são identificados para atender aos requisitos, trabalhando individual ou coletivamente. Entretanto, um dado subsistema pode ser imposto por outros fatores que não sejam o atendimento aos requisitos, como fatores organizacionais e ambientais. No último caso as atividades devem ser separadas, total ou parcialmente.

- *Atribuir requisitos a subsistemas*: Normalmente esta atividade é feita de forma direta, entretanto existe a possibilidade de subsistemas pré-existentes ou comprados de terceiros integrar o sistema. Neste caso a atribuição é necessária, mesmo porque tais subsistemas podem ser utilizados parcialmente e atender um menor número de requisitos do que poderia.

- *Especificar a funcionalidade de subsistemas*: Nesta atividade as funções de cada subsistema são identificadas e definidas, bem como as relações existentes entre os subsistemas.

- *Definir a interface de subsistemas*: Interfaces são os meios pelos quais duas ou mais entidades se comunicam ou relacionam. Após a definição e especificação da funcionalidade de um subsistema, a sua interface deve ser definida para garantir sua correta utilização e possibilitar o desenvolvimento de subsistemas em paralelo.

2.2.4.3 Desenvolvimento de subsistemas

A fase desenvolvimento compreende o desenvolvimento propriamente dito, ou seja, todas as atividades para a concretização do componente ou subsistema. No último caso, existe a necessidade de se iniciar um novo processo de engenharia para cada subsistema integrante do processo.

Dentre os documentos resultantes da fase de projeto, existe um cronograma que deve orientar os coordenadores quanto à ordem e às atividades desenvolvidas pelas equipes/setores. Muitos subsistemas/componentes são dependentes cronológicos, ou seja, somente serão desenvolvidos após a conclusão de outros subsistemas. Portanto, para que não haja atraso no desenvolvimento e resultante aumento de custos, as atividades devem previstas por especialistas e coordenadas para não causar excesso ou escassez de atividades para um determinado setor/equipe. Como resultado, muitas vezes, existe o desenvolvimento em paralelo de subsistemas, visando otimizar os recursos humanos e materiais.

Contudo, dentro do desenvolvimento, emergem problemas que exorbitam a competência de um subsistema. Nestes casos, o projeto do sistema deve ser modificado a tempo para que não haja a impossibilidade de modificação de outros subsistemas/componentes. Geralmente, depois de concluído, muitos subsistemas possuem um elevado custo de adaptação, o que resulta – em muitos casos – na utilização de softwares para as correções e adaptações necessárias.

2.2.4.4 Integração de sistemas

Depois de desenvolvidos, os subsistemas devem ser integrados para que formem o sistema como um todo. Essa integração pode ser desenvolvida em fases, integrando um subsistema por vez, ou pode ser adotada a abordagem *big bang*, onde todos os subsistemas são integrados concomitantemente.

Dentre as abordagens descritas acima, abordagem por fases é, sem dúvida, a mais utilizada. As razões pela preferência da primeira abordagem estão na organização e nos custos envolvidos com a integração. Na abordagem *big bang*, a programação das equipes e a disponibilidade de tempo e espaço são mais dispendiosas, além da programação realizada no próprio projeto do sistema dificilmente resultar no término simultâneo dos diferentes subsistemas. Outra razão se baseia nos custos e no tempo necessário para se localizar os erros durante a etapa de testes que o sistema será submetido. Os erros são mais facilmente

encontrados quando se integra por partes o sistema, uma vez que a procura se limitará às possibilidades relacionadas com a integração do último subsistema.

2.2.4.5 Instalação de sistemas

Muitos problemas podem ser encontrados quando se instala o sistema no ambiente no qual deverá operar. Aquilo que parece ser um processo simples e rápido pode tornar-se complexo e levar meses e até anos para a sua realização.

Os problemas podem ser:

- *Ambientais*: Problemas relacionados com a falta ou a má de especificação de requisitos ambientais que levam os sistemas a operar indevidamente ou não atender a plenitude dos demais requisitos especificados.

- *Humanos*: Muitos sistemas vêem substituir o trabalho humano, a hostilidade das pessoas envolvidas no processo de instalação pode prejudicar o processo por falta de informação ou cooperação em relação às tarefas desenvolvidas pelos instaladores.

- *Organizacionais*: Muitas organizações são cautelosas, exigindo que o novo sistema coexista com o antigo para validar os resultados e garantir sua operabilidade. Conflitos podem surgir no compartilhamento de algum componente ou recurso.

- *Físicos*: Muitas entidades sofrem modificações ou estão impossibilitadas de efetuarem algum tipo de mudança (como no caso de bancos comerciais instalados em patrimônios históricos). Nestes casos algumas modificações podem ser necessárias, como se especificar uma rede sem fios devido à impossibilidade de passar cabos de rede.

Após a instalação, normalmente, existe o treinamento dos operadores do sistema e as modificações procedimentais das atividades desenvolvidas, caso necessárias.

2.2.4.6 Evolução de sistemas

Ao longo do tempo, novas técnicas, novos requisitos, novos equipamentos, surgem como opções de melhoria ou até mesmo imposições de melhoria. Os sistemas passam a ser reavaliados técnica e economicamente para embasar uma decisão de evolução ou substituição sistêmica. A evolução do sistema deve ser feita de forma cuidadosa, pois uma única alteração

num determinado subsistema pode acarretar uma série de modificações nos demais subsistemas. Problemas quanto à documentação e aos modelos de antigos sistemas podem dificultar simples alterações, bem como a grande complexidade técnica de alguns subsistemas podem tornar seu novo desenvolvimento dispendioso – levando-o a ser mantido na forma que se encontra. Assim, a análise técnica-econômica deve ser feita considerando possíveis complicadores dentro do processo de atualização.

2.2.4.7 Desativação de sistemas

A retirada de operação é uma tarefa relativamente simples quando se trata de sistemas computacionais, porém, em outras áreas, a retirada de operação requer uma série de procedimentos – principalmente quando o sistema trabalha com materiais prejudiciais à saúde e ao meio ambiente. Contudo, para a retirada de operação de um sistema computacional, basta certificar-se da eficácia, da independência e da operabilidade do novo sistema. Muitas vezes, por erros quanto essas certificações, a desativação torna-se uma tarefa que pode levar a não operação da entidade por um razoável tempo e resultar num prejuízo significativo.

2.3 O Processo de Software

O conjunto de atividades coordenadas com o propósito de se obter um produto de software é conhecido como processo de software. Durante décadas, a indústria do software desenvolveu produtos para os mais diversos setores da sociedade. Dessa experiência resultaram diversos processos de softwares aplicados no desenvolvimento de tipos diferentes de produtos, sendo estudados e melhorados constantemente por diferentes entidades de pesquisa, desde a pesquisa com o cunho industrial até a pesquisa com o cunho militar.

Estudos e exigências recentes levaram ao desenvolvimento de modelos para avaliar e melhorar o processo utilizado no desenvolvimento de softwares nas organizações [41-43]. Contudo, apesar do grande empenho de pesquisadores e estudiosos, não existe um processo de software considerado ideal. Tendo em vista que o processo de software é um processo intelectual (que depende de julgamento humano) e que existem de características peculiares das diversas áreas de aplicação, cada empresa desenvolvedora criou processos próprios adaptados a sua realidade.

Embora existam diferentes processos, todos apresentam atividades fundamentais comuns. Portanto, nessa seção, este estudo se focará, de forma resumida, a expor tais atividades.

2.3.1 Modelos de processo de software

Os modelos de processo de software são representações abstratas e arquiteturais de um processo de software. Tais modelos não primam pelos detalhes das atividades inerentes do processo, eles, simplesmente, relacionam e indicam as maneiras de desenvolver tais atividades.

2.3.1.1 Modelo em cascata

Como dito anteriormente, o modelo em cascata teve origem em outros processos de engenharia e foi o primeiro modelo publicado do processo de desenvolvimento de software [44]. Sua denominação advém da estrutura das fases, como mostra a Figura 4, que se assemelham a uma cascata.

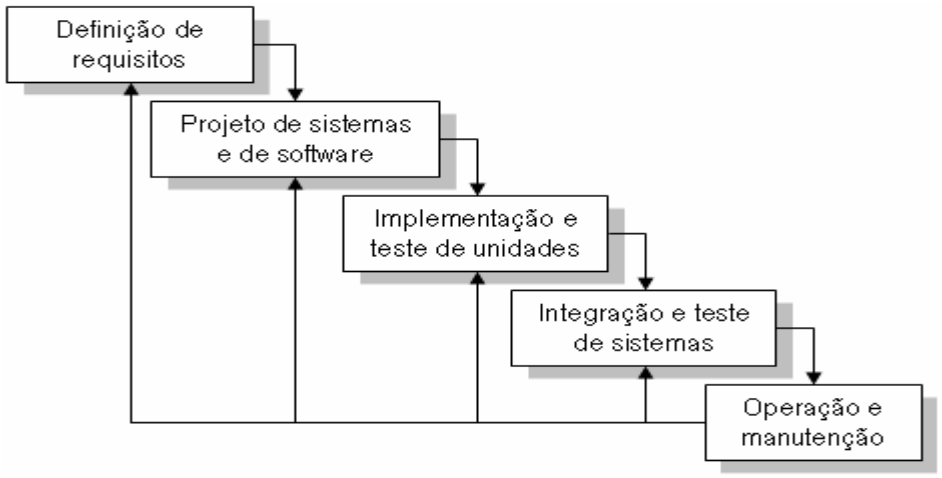


Figura 4 – O modelo de desenvolvimento de software em cascata

Os principais estágios do modelo retratam as atividades de desenvolvimento fundamentais:

- *Definição dos requisitos*: Os objetivos do sistema, contendo suas funcionalidades e restrições operativas, são estabelecidos e detalhados através das consultas a clientes, usuário, consultores técnicos e pesquisadores. Neste estágio, o sistema é especificado.

- *Projeto de sistemas e de software*: São projetos tanto da parte física (hardware) quanto da parte lógica (software) do sistema como um todo, fundamentados na especificação da etapa anterior. Como resultado, a arquitetura do sistema computacional é descrita e documentada.

- *Implementação e teste de unidades*: Consiste na implementação das unidades de programa, ou seja, o desenvolvimento propriamente dito dos componentes/subsistemas que integram o sistema. Tais unidades, depois de implementadas, são avaliadas através de uma seqüência de testes.

- *Integração e teste de sistemas*: As unidades de programa são integradas e testadas para formar o sistema completo. Os testes são mais elaborados e visam avaliar o atendimento aos requisitos especificados no primeiro estágio. Após os testes, o sistema é entregue ao cliente.

- *Operação e manutenção*: Trata-se de se tornar o sistema operativo em seu ambiente computacional. Neste estágio, o sistema é reavaliado para descobrir erros que passaram pelas etapas anteriores, melhorando a implementação das unidades e evoluindo o sistema conforme o surgimento de novos requisitos.

O modelo em cascata não é um modelo linear, este envolve uma série de iterações das atividades de desenvolvimento. Em cada iteração, as fases do modelo são executadas novamente para uma nova parte do sistema. Assim o sistema é montado sequencialmente. Teoricamente, numa iteração de software, um dado estágio do modelo não se inicia sem o término do outro. Na prática, os estágios se sobrepõem e trocam informações entre si.

O modelo em cascata não é muito utilizado devido ao elevado custo da iteração de software, a qual requer que uma parte significativa do trabalho seja reavaliada e 'retrabalhada'. Tal característica induz à suspensão de parte do desenvolvimento para um posterior tratamento, resultando em adaptações de software que comprometem a integridade do sistema.

2.3.1.2 Desenvolvimento evolucionário

O desenvolvimento evolucionário é caracterizado por ter as atividades de especificação, desenvolvimento e validação sendo realizadas rapidamente para gerar uma versão que será avaliada pelo usuário do sistema. Durante o desenvolvimento, muitas versões intermediárias são obtidas e avaliadas até que o sistema final seja alcançado. Isso facilita o processo de especificação do sistema, uma vez que os requisitos são avaliados constantemente e acrescentados à medida que o sistema evolui, sendo, portanto, melhor compreendidos tanto por parte dos desenvolvedores quanto por parte dos usuários. A Figura 5 mostra a estrutura do desenvolvimento evolucionário.

Existem algumas desvantagens técnicas e organizacionais de se utilizar esse modelo de processo de software, são elas:

- *Gerenciamento do processo*: A documentação das versões intermediárias, devido ao grande número de versões, não é realizada de forma a refletir detalhadamente o estágio de desenvolvimento. A medição do progresso do sistema, para avaliar o desenvolvimento, torna-se mais complexa e menos exata.

- *Má estruturação do sistema*: As mudanças e a inserção de novos requisitos podem corromper a estrutura do sistema, uma vez que as adaptações podem ser de difícil realização numa estrutura que não previa determinado requisito.

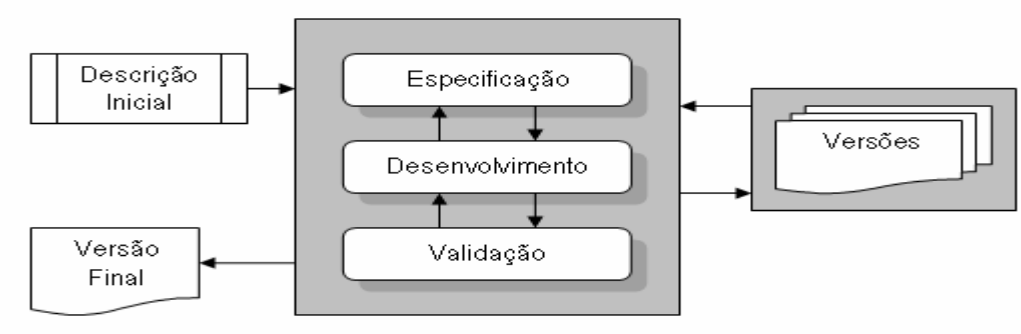


Figura 5 – Desenvolvimento evolucionário

Há dois tipos de desenvolvimento evolucionário:

- *Desenvolvimento exploratório*, que consiste em trabalhar com o cliente, explorando os requisitos e evoluindo constantemente o sistema.

- *Fazer protótipos descartáveis*, que consiste em entender os requisitos do cliente através de programas descartáveis de rápida implementação e posterior implementação.

2.3.1.3 Desenvolvimento formal de sistemas

O desenvolvimento formal de sistemas se aproxima muito do modelo em cascata, discutido anteriormente, com algumas modificações na especificação de requisitos e implementação do sistema. Este modelo de processo de software é caracterizado pela especificação expressa em notação matemática e a sua transformação matemática formal. A Figura 6 ilustra o processo formal.

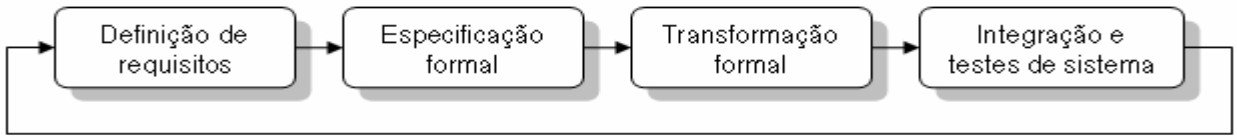


Figura 6 - Desenvolvimento formal de sistemas

No desenvolvimento formal, o sistema é especificado por um conjunto de requisitos definidos através de notação matemática detalhada, onde cada requisito é representado por uma ou mais expressões matemáticas. O projeto, implementação e testes de unidades são substituídos por um processo de desenvolvimento transformacional, onde a especificação formal é refinada por meio de uma série de transformações até ser inserida no sistema (transformada em código).

Esta abordagem é particularmente adequada ao desenvolvimento de sistemas que tenham rigorosas exigências de segurança, confiabilidade e garantia. Projetos utilizando o desenvolvimento formal [45] constataram um menor número de defeitos e um custo competitivo quando comparado com as outras abordagens, porém utilizaram pessoal especializado e foram aplicados a sistemas que requerem certificações de comitês técnicos. Contudo, para a grande maioria dos sistemas, essa abordagem não trás um benefício latente.

2.3.1.4 Desenvolvimento orientado a reuso

O reuso é prática comum dentro do processo de software, haja vista que muitos desenvolvedores aproveitam códigos similares em novas implementações. Para tanto, algumas adaptações são feitas e o resultado do trabalho é obtido de forma mais rápida, sendo tal prática muito utilizada no desenvolvimento evolucionário. Este tipo de procedimento é conhecido como reuso informal, e acontece independentemente da abordagem genérica utilizada.

Nos últimos anos, surgiu a engenharia de software com base em componentes, a qual conta com o reuso no desenvolvimento de sistemas. Tal abordagem, que esta se tornando cada vez mais utilizada, é conhecida como *desenvolvimento orientado a reuso*. Seu aparente sucesso é fruto do desenvolvimento de sistemas cada vez maiores e, portanto, mais dispendiosos de serem desenvolvidos. O reuso diminui a quantidade de trabalho a ser empenhada e, consequentemente, o tempo e custo de desenvolvimento do sistema.

2.3.1.5 Desenvolvimento incremental

Os modelos apresentados anteriormente ou possuem uma inflexibilidade quanto à inserção de novos requisitos – como o modelo em cascata – ou se transformam em sistemas estruturalmente comprometidos e de difícil manutenção conforme novos requisitos são acrescentados – como o desenvolvimento evolucionário. Para solucionar o problema, uma abordagem intermediária foi proposta [46] com o objetivo de reduzir o 'retrabalho' do modelo em cascata e proporcionar aos projetistas e usuários a postergação de uma análise mais detalhada dos requisitos. Esta abordagem é conhecida como desenvolvimento incremental pelo fato de ser base baseada num processo iterativo de desenvolvimento que relaciona, a cada iteração, as funcionalidades a serem implementadas a incrementos. A Figura 7 ilustra as atividades do desenvolvimento incremental.

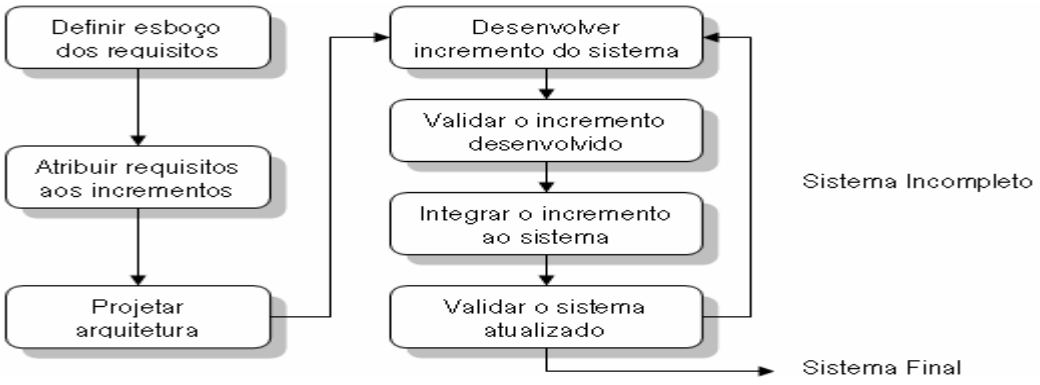


Figura 7 – Desenvolvimento incremental

Inicialmente, um esboço dos requisitos contendo as funcionalidades com suas respectivas prioridades ou importâncias é identificado. Após a identificação das funcionalidades, uma série de estágios de desenvolvimento, ordenada por ordem crescente de importância, é definida de forma a disponibilizar as principais funções nas primeiras versões entregues. Caso haja mudanças ou inserção de requisitos em estágios futuros, suas funcionalidades serão reavaliadas, relacionadas a incrementos e inseridas no conjunto de incrementos a implementar. Para tanto, se faz necessário que o projeto de arquitetura também seja atualizado para refletir as mudanças geradas pelas novas funcionalidades, porém – uma vez que as principais funções já foram identificadas – as mudanças na arquitetura projetada são pequenas.

A principal modificação desta abordagem, em relação ao desenvolvimento evolucionário, está no tratamento dado aos requisitos, que neste caso são avaliados anteriormente e não de forma contínua conforme o sistema evolui. Porém, quando comparado com o modelo em cascata, o desenvolvimento incremental proporciona o acompanhamento pelo cliente/usuário, através de versões intermediárias, do processo de desenvolvimento.

O desenvolvimento incremental também possui restrições. Os incrementos não devem ser grandes, mas cada incremento deve produzir alguma funcionalidade. Isto torna o desenvolvimento mais oneroso devido ao mapeamento de funcionalidades a incrementos de tamanho limitado. Outra questão emergente é a identificação de facilidades comuns utilizadas por diferentes partes do sistema, essa tarefa também fica prejudicada – uma vez que os requisitos não são detalhadamente analisados.

2.3.1.6 Desenvolvimento em espiral

Os modelos apresentados até agora representam o processo de software como uma seqüência de atividades, as quais podem ou não trocar de informações durante a transição de uma para outra. Apesar da boa adaptação destes modelos a muitas aplicações, algumas questões – como a análise do risco e o planejamento das fases – não eram tratadas satisfatoriamente. Muitos imprevistos durante o desenvolvimento comprometiam o tempo e, consequentemente, o custo envolvido no projeto.

Um novo modelo de desenvolvimento, conhecido como modelo de desenvolvimento em espiral [47], permitiu um melhor planejamento e um desenvolvimento mais seguro do sistema. Internamente o modelo em espiral utiliza outros modelos de desenvolvimento, tendo, porém, a

adição de atividades preliminares e posteriores como melhorias. A grande distinção deste modelo é a inclusão da análise de riscos dentro das fases do processo de software.

O modelo supracitado, conforme mostra a Figura 8, consiste numa espiral divida em 4 setores, onde cada setor representa um conjunto de atividades aplicadas à fase do processo de software ou de desenvolvimento. Cada camada da espiral corresponde a uma fase de desenvolvimento, sempre começando de dentro para fora. A primeira camada, ou fase inicial, consiste no estudo de viabilidade técnico-econômica do sistema; A segunda camada, a análise de requisitos; a terceira camada, o projeto do sistema, e assim por diante.

Os setores ou as atividades adicionais são realizados em todas as fases de desenvolvimento, tendo cada um deles as seguintes características:

1. *Definição de objetivos*: Os objetivos, as restrições, bem como os riscos do processo e do produto são identificados e plano de gerenciamento detalhado é elaborado. Dependendo do grau de risco, um plano de possíveis alternativas deve ser proposto.

2. *Avaliação e redução de riscos*: Para cada um dos riscos identificados, é realizada uma análise detalhada e são tomadas as providências para reduzir a probabilidade de ocorrência.

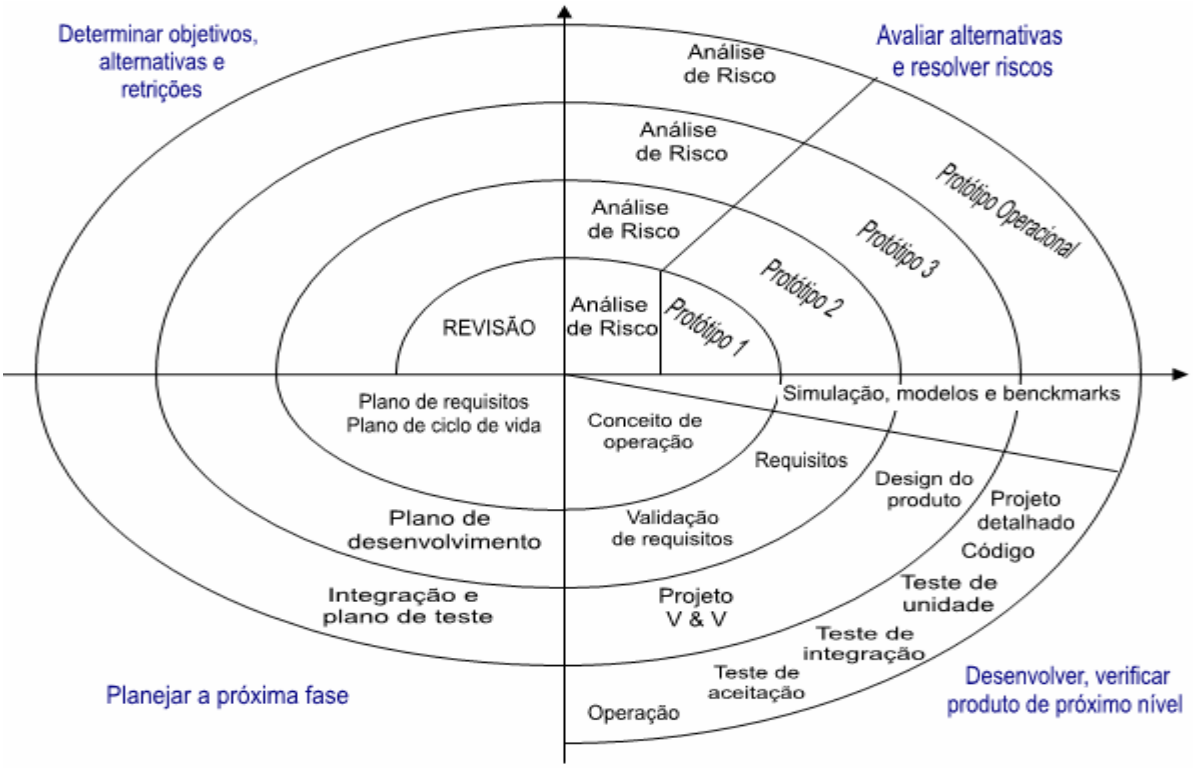


Figura 8 – Desenvolvimento em espiral

3. *Desenvolvimento e validação*: Seleciona-se o modelo de desenvolvimento do sistema que melhor se adaptar às exigências do projeto e os riscos avaliados e efetua-se, após o desenvolvimento da fase, a respectiva validação.

4. *Planejamento*: Realiza-se uma revisão do projeto para decidir sobre sua continuidade ou seu cancelamento. Caso o projeto continue, um plano de metas e realizações é traçado para a próxima fase do processo de software.

2.3.2 Especificação de software

A especificação de requisitos é uma atividade particularmente importante do processo de software, uma vez que os efeitos dos erros cometidos nesta atividade produzem, inevitavelmente, futuros problemas no projeto, implementação, validação e operação do sistema. Muitos estudos se voltaram para a especificação de software nas últimas décadas [47-50], desenvolvendo uma disciplina a parte conhecida como engenharia de requisitos – a qual tem a finalidade básica de definir, com clareza e segurança, as funções do sistema e as restrições operacionais e de desenvolvimento.

O processo de engenharia de requisitos, como mostra a Figura 9, possui quatro fases principais, onde cada uma delas produz um determinado tipo de documento. A consolidação destes documentos ou a documentação dos requisitos é a especificação do sistema propriamente dita. No entanto, durante o processo, pode-se concluir que não existe viabilidade técnico-econômica para o desenvolvimento do sistema, resultando no término das atividades relacionadas ao mesmo. Contudo, caso o sistema seja viável, a documentação deve ser elaborada em dois níveis distintos de detalhamento. O primeiro nível ou alto nível, destinado para os clientes e usuário, apresenta os requisitos de através de uma linguagem de fácil entendimento e os modelos resultantes sem os detalhes técnicos. O documento com nível mais baixo ou baixo nível, destinado à equipe de projeto e desenvolvimento, possui os detalhes necessários para o projeto, desenvolvimento e validação do sistema, bem como utiliza uma linguagem técnica.

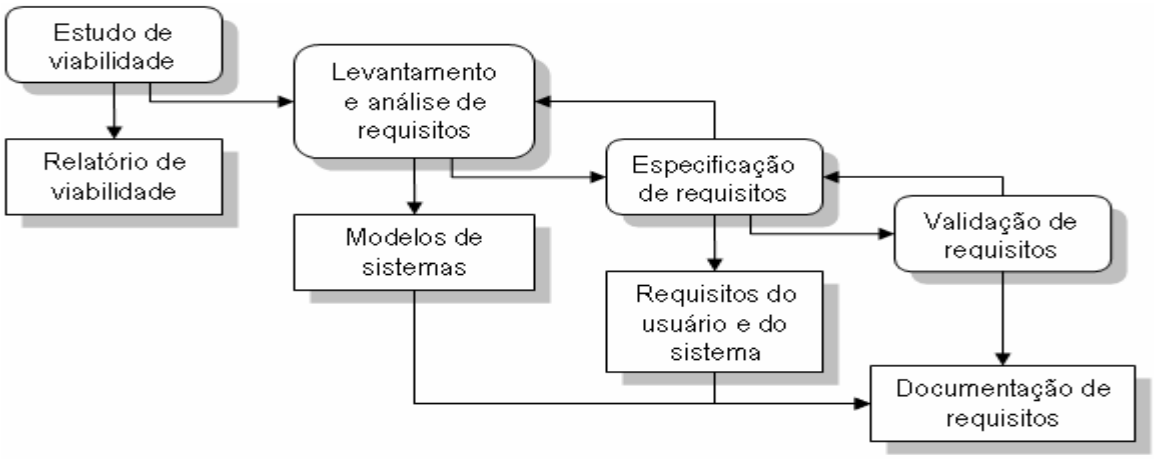


Figura 9 – O processo de engenharia de requisitos

As principais fases do processo de engenharia de requisitos são:

1. Estudo de viabilidade: Os requisitos principais e as finalidades do sistema são analisados técnica e economicamente com base no orçamento disponível e nas tecnologias que podem ser empregadas. O estudo deve ser rápido e barato. Seu resultado é fundamental para continuidade do processo de software.

2. Levantamento e análise de requisitos: São empregadas técnicas de levantamento de requisitos (como a etnografia e a análise de cenários) e análises de documentos e sistemas existentes. O levantamento pode resultar em modelos simplificados ou protótipos de sistemas, visando à compreensão mais detalhada dos requisitos.

3. Especificação de requisitos: Consiste na tradução das informações para um documento que defina o conjunto de requisitos formalmente, seja através do modelo e/ou protótipo ou através da relação de requisitos identificados. Os requisitos definidos no documento podem ser da funcionalidade a ser fornecida (requisitos do sistema) ou da necessidade de clientes e usuários finais (requisitos do usuário).

4. Validação de requisitos: Verificar a existência de conflitos e incompatibilidade entre os requisitos especificados visando corrigir tais problemas. Os requisitos são analisados quanto a sua pertinência, consistência e integralidade.

No processo de engenharia de requisitos, não existe uma seqüência rigorosa a ser seguida, uma vez que a análise é feita de forma contínua – podendo ser descobertos novos requisitos em outras etapas do processo (como na especificação). Logo, as etapas de análise, especificação e validação são intercaladas.

2.3.4 Projeto de software

O projeto, a implementação e a especificação do software são fases distintas do processo de software, porém algumas vezes não há uma distinção nítida destas fases – dependendo do modelo de processo utilizado, como o caso do desenvolvimento evolucionário. Contudo, o projeto e a implementação são atividades de conversão dos requisitos especificados em um software ou sistema executável.

O projeto se ocupa em descrever a estrutura do software a ser implementado em conjunto com a estrutura dos dados, das interfaces entre os componentes/subsistemas e dos algoritmos fundamentais, quando necessários. O desenvolvimento do projeto não é feito de forma direta, trata-se de um processo iterativo de acréscimo de detalhes e informações que geram versões intermediárias. A cada novo acréscimo, a última versão do projeto é novamente analisada, as modificações na antiga estrutura são destacadas e os novos detalhes são inseridos.

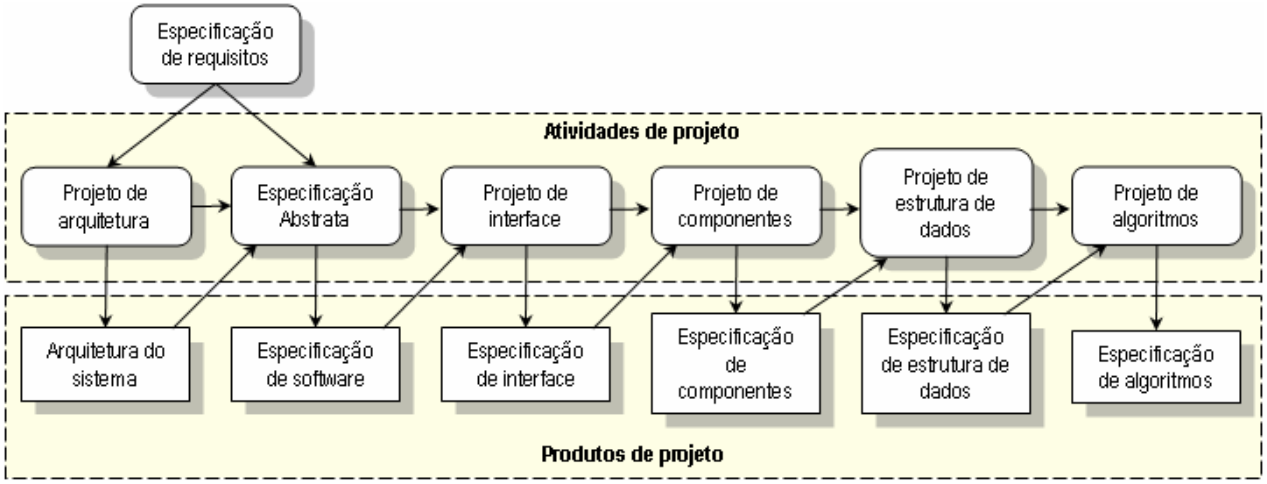


Figura 10 – Modelo genérico do processo de projeto

A Figura 10 mostra um modelo genérico de processo de projeto, o qual – na prática – deve ser adaptado às características do projeto. O processo de projeto é decomposto em

diferentes estágios de desenvolvimento, onde cada estágio é responsável pela adição de um determinado tipo de informação ou detalhe ao projeto. Os primeiros estágios resultam em modelos do sistema com um menor grau de detalhamento, sendo estes utilizados em contatos com clientes e usuários. Os estágios finais proporcionam modelos mais detalhados, os quais serão utilizados na implementação do software. A cada novo estágio, as informações sobre o projeto e, consequentemente, os modelos são reavaliados e, portanto, proporcionam a identificação de erros ou omissões nos estágios anteriores. Apesar do modelo apresentado na Figura 10 ser seqüencial, na prática existe o retorno de um estágio para outro devido aos problemas encontrados na reavaliação.

Abaixo, as atividades específicas do processo de projeto são resumidas:

1. *Projeto de arquitetura*: Consiste na identificação de quais subsistemas compõem o sistema a ser desenvolvido e quais são suas relações;

2. *Especificação Abstrata*: As funcionalidades fornecidas por cada subsistema são identificadas e definidas, bem como suas restrições operacionais e de desenvolvimento;

3. *Projeto de interface*: A interface de cada subsistema é especificada e documentada para viabilizar o desenvolvimento em paralelo de subsistemas. Deve-se atentar para não existir ambigüidades na especificação;

4. *Projeto de componentes*: São identificados os componentes e especificadas suas funções. Para cada componente é projetada uma interface.

5. *Projeto de estrutura de dados*: As diferentes estruturas de dados são projetadas em detalhes e especificadas, considerando-se as características dos componentes e subsistemas, bem como suas relações e finalidades.

6. *Projeto de algoritmo:* Os principais algoritmos, ou até mesmo todos, são elaborados detalhadamente de forma representativa, ou seja, projetados. Como resultado, obtém-se a especificação dos algoritmos que será utilizada na implementação do software.

2.3.4.1 Métodos estruturados de projeto

Os métodos estruturados são procedimentos, notações e diretrizes aplicados ao processo de projeto com o objetivo de orientar a equipe de projeto e padronizar a informação que consta na documentação do projeto. Nas últimas décadas foram propostos diferentes métodos de estruturados, dos quais se destacam o 'Projeto Estruturado' [51], a 'Analise de Sistemas Estruturados' [52], o 'Desenvolvimento de Sistemas' [53] e as diferentes abordagens de projeto orientado a objetos [1-6, 9].

Muitos sistemas de grande porte tiveram sucesso na aplicação de métodos estruturados de projeto, obtendo reduções significativas nos custos. O sucesso de tais métodos foi atribuído à padronização das notações utilizadas no projeto e a velocidade e segurança na produção de sua documentação-padrão. Entretanto não se pode demonstrar que um determinado método é melhor ou pior que outro, uma vez que estes possuem dependências com o domínio de uma aplicação.

Apesar das diferenças encontradas nos diferentes métodos, características comuns são encontradas em todos, como, por exemplo, a contemplação de um modelo de processo de projeto, de notações padronizadas, de formulários de relatórios, de regras e diretrizes de projeto. Além de poderem aceitar alguns ou todos os seguintes modelos de um sistema:

1. *Modelo de fluxo de dados:* Modelagem do sistema com foco nas transformações que são aplicadas sobre os dados pelos diferentes procedimentos e funções;

2. *Modelo de relacionamento de entidades:* Muito utilizado na estruturação de base de dados, tal modelo se caracteriza por exibir as entidades do domínio da aplicação e suas respectivas relações;

3. *Modelo estrutural:* Modelagem dos componentes do sistema e suas interações;

4. *Métodos orientados a objetos*: Modelo de herança, modelo de relacionamento estático e dinâmico, modelo de interação, etc. Tais modelos serão discutidos com maiores detalhes no próximo capítulo, quando do estudo da UML. Tais modelos são conhecidos como diagramas.

Contudo, apesar da abordagem metódica sugerida pelos métodos estruturados de projeto, a aplicação de tais métodos, na prática, não é seguida estritamente [54]. Estes fornecem margens para as decisões e as adaptações necessárias, não sendo rígidos quanto a sua orientação. Tais métodos indicam a padronização e as diretrizes seguidas, indicando a adoção de boas práticas na condução do processo de projeto. Entretanto a criatividade do projetista é essencial para a decomposição do sistema e a garantia de atendimento da especificação, sendo o método aplicado resultante das circunstâncias locais.

2.3.5 Validação de software

A validação, ou verificação e validação (V & V), se concentra na determinação de erros ou omissões existente no sistema quanto a sua especificação. Tais erros podem ter origem no mau funcionamento de um componente, módulo, subsistema ou sistema – mas, em geral, eles estão relacionados ao não entendimento dos requisitos ou a omissão de parte destes na especificação ou projeto do sistema. Durante a validação, todas as etapas anteriores do processo de software são verificadas, desde os testes feitos pelos programadores (teste de unidade e módulo) até os testes feitos por uma equipe independente de testadores (teste de subsistemas, sistema e aceitação). Cada fase do processo de software gera ou um plano de teste para verificar sua adequação ou um conjunto de testes para assegurar sua funcionalidade.

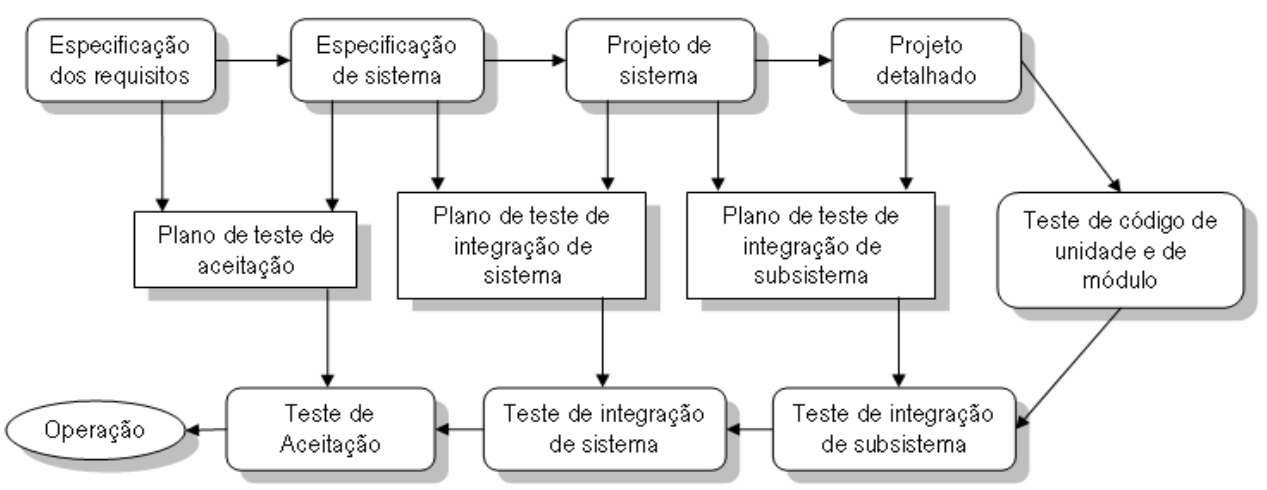


Figura 11 – Fases de teste no processo de software

Os diferentes tipos de testes são:

- ✓ *Teste de unidade:* Verificação da correta operação dos componentes individuais, normalmente, feita pelos próprios programadores;

- ✓ *Teste de Módulo:* Verificação do conjunto dos procedimentos/funções ou das classes de objetos que formam os módulos do sistema. O módulo é composto pelos componentes diretamente relacionados e sua verificação engloba a avaliação destas relações. Normalmente essa tarefa é desempenhada pelos programadores;

- ✓ *Teste de subsistemas:* Verificação da correta definição e operação das interfaces em conjunto com a adequação dos módulos às funcionalidades esperadas do subsistema. Basicamente, o teste de subsistemas verifica a integração dos módulos que formam o subsistema, bem como a validação da interface. Esse teste, normalmente, é feito por uma equipe independente de testadores, devido à amplitude dos testes;

- ✓ *Teste de sistema:* Verificação da integração dos diferentes subsistemas e de suas interfaces, bem como a validação das propriedades emergentes e dos requisitos funcionais e não funcionais;

- ✓ *Teste de aceitação:* É o estágio pré-operacional, no qual o sistema é testado como os dados reais do problema (do cliente) ao invés de dados criados para os testes anteriores. Esse teste pode revelar erros e omissões na definição de requisitos, problemas na especificação do sistema, não atendimento das necessidades dos usuários e desempenho inaceitável. O sistema só entrará em operação se as equipes de teste do cliente e do desenvolvedor aceitarem os resultados obtidos e validarem o sistema perante aquilo que foi contratado.

Apesar da aparente independência das diferentes fases de teste, conforme mostra a Figura 11, existe a possibilidade de testes posteriores – como o teste de integração de sistemas – localizarem erros não verificados em testes anteriores – como o teste de módulos. Portanto, a verificação e validação é um estágio iterativo, uma vez que os erros encontrados devem ser solucionados e os testes devem ser refeitos para assegurar a não existência de erros e omissões causados pela última atualização.

2.3.6 Evolução de software

A evolução ou manutenção do software é a tarefa de atualizar e incorporar funcionalidades às novas exigências dos clientes. As alterações organizacionais, o surgimento de novas tecnologias e novas práticas são alguns exemplos de mudanças que podem acarretar a necessidade da manutenção do software. Contudo, cabe ressaltar, a manutenção é, por vezes, mais dispendiosa que o próprio desenvolvimento inicial do software e, inerentemente, está vinculada ao projeto existente. Portanto uma análise técnica-econômica e organizacional deverá avaliar as mudanças necessárias e decidir sobre a evolução ou a criação de um novo sistema.

Muitos sistemas computacionais estão se tornando cada vez mais complexos e, consequentemente, grandes. Tal característica torna um novo desenvolvimento algo não desejado tanto pelo ponto de vista econômico quanto pelo ponto de vista operacional da organização. A manutenção contínua e o reuso tornaram-se, atualmente, práticas comuns no sentido de reduzir custos e prazos, levando a consideração – pela engenharia de software – de um processo integrado de desenvolvimento e manutenção contínua em face das constantes mudanças nos requisitos dos grandes sistemas computacionais.

Capítulo 3

A Modelagem Orientada a Objetos

3.1 Considerações Gerais

A computação está em constante transformação, seja pela evolução do hardware ou pelo surgimento de novos conceitos e técnicas computacionais. A evolução do hardware está diretamente ligada à limitação da aplicação da computação a áreas que necessitam de alto desempenho ou de componentes especiais. Por sua vez, os novos conceitos computacionais surgem através dos problemas encontrados com a própria aplicação da computação nas áreas fins ou como forma de solução da limitação do hardware disponível.

As mudanças conceituais revolucionaram a maneira com a qual se desenvolvia o software, principalmente quando os novos conceitos e técnicas estavam diretamente ligados às linguagens de programação. Na década de 50 as linguagens de baixo nível impunham aos programadores um profundo conhecimento de códigos e símbolos utilizados na programação, além de um conhecimento do hardware para o qual o programa seria desenvolvido. As linguagens de alto nível, desenvolvidas para serem mais parecidas com a linguagem humana, eliminaram a complexidade de utilização dos símbolos. As linguagens estruturadas facilitaram a análise dos programas pela não utilização de diretivas de direcionamento lógico, pois tais diretivas – como o *goto* – causavam transtornos na análise de grandes sistemas. As linguagens procedimentais eliminaram a necessidade de repetições exaustivas do mesmo bloco de código no programa e criaram uma série de técnicas relacionadas ao uso de procedimentos e funções. As linguagens orientadas a objetos, por fim, transformaram o modo de análise e de desenvolvimento de software, uma vez que a definição dos objetos (dados encapsulados) e de seus relacionamentos transformou-se no foco da programação ao invés das transformações sucessivas que o fluxo de dados sofria – a qual se focava nos procedimentos e nas funções.

Desde o final da década de 60, com o surgimento da linguagem *SIMULA*, uma crescente importância foi dada à programação orientada a objetos (POO) – resultando em técnicas que, atualmente, exorbitam o desenvolvimento de sistemas, chegando a ter aplicabilidade em outras áreas como administração de empresas, engenharia de negócios, gerenciamento e modelagem de dados etc. Durante a década de 90, diferentes estudos focaram-se na análise, na

metodologia e na modelagem de sistemas baseados na orientação a objetos (OO) [1-6], levando ao desenvolvimento da *Unified Modeling Language* [7-8] – ou UML – e do processo de desenvolvimento em UML [9].

Devido à importância que a OO possui neste trabalho, o presente capítulo se destina à apresentação dos conceitos básicos da modelagem orientada a objetos (MOO) e à introdução dos principais conceitos e diagramas da UML. Maiores aprofundamentos teóricos podem ser encontrados em [8], [9] [10], [11] e [12].

3.2 Os Princípios da Programação Orientada a Objetos

Durante décadas, o modelo de fluxo de dados foi o principal modelo utilizado no desenvolvimento de sistemas. O motivo de sua grande importância se baseava na concepção dada para o próprio software, o qual era visto como um conjunto procedimentos e funções que utilizavam e transformavam os dados com o intuito de se obter determinadas saídas. Neste modelo os elementos principais são os procedimentos e as funções que orientam o fluxo de dados de forma coordenada. Grande parte do esforço foca-se na implementação do sistema, sendo a padronização dos dados utilizados entre os principais elementos uma atividade essencial que demanda uma análise e definição pretérita, ou seja, durante o projeto do sistema. Entretanto a maioria dos problemas relativos à utilização destes dados somente é detectada durante a implementação, fase na qual a incompatibilidade ou a necessidade concretizam-se numa situação que requer uma solução imediata – normalmente resolvida pela manipulação da estrutura local de dados.

A inflexibilidade do modelo de fluxo de dados frente às mudanças é o resultado desta definição e implementação precoce dos elementos do sistema, o que limita o campo de opções de projeto durante o desenvolvimento do software e, consequentemente, aumenta seus custos.

A programação orientada a objetos proporcionou uma profunda mudança na filosofia de desenvolvimento de sistemas. Conceitualmente, um objeto é uma representação – através de um conjunto de dados e de métodos – de um conceito ou elemento do domínio da aplicação. A principal atividade de desenvolvimento no modelo orientado a objetos é a definição dos objetos e de suas relações, ou seja, é a descrição do problema através de sua decomposição em um conjunto de elementos reais que interagem entre si. Desta forma, o entendimento das pessoas quanto à essência do projeto se torna mais fácil, bem como seu gerenciamento e sua manutenção – haja vista que a estrutura e os objetos descritos no modelo representam

conceitos reais que quase não sofrem alterações (quando bem definidos). Projetar um sistema utilizando a orientação a objetos é, resumidamente, pensar em termos de *coisas* afetas ao problema em questão.

Muitos são os benefícios obtidos com adoção da modelagem orientada a objetos, dentre os quais se destacam [10]:

- Um modelo de sistema que descreve com maior realismo os conceitos reais daquilo que está sendo projetado;

- Maior facilidade de compreensão e, consequentemente, de manutenção devido ao mapeamento entre os objetos e os conceitos do mundo real;

- Postergação das decisões sobre o detalhamento do projeto para etapas futuras de desenvolvimento, ou seja, maior flexibilidade;

- Maior facilidade de transição entre as etapas de análise, projeto e implementação através da adição contínua de detalhes.

A etapa de análise, que compreende a modelagem abstrata do sistema, é a atividade mais importante do desenvolvimento devido à própria natureza da orientação a objeto. Entretanto o modelo resultante que define os objetos e suas relações não possui um aprofundamento técnico, ou seja, um detalhamento da estrutura dos objetos. Somente na etapa de projeto que os detalhes técnicos serão acrescentados em conjunto com objetos infra-estruturais necessários (interfaces gráficas, comunicação, banco de dados). Esta análise prioritária da essência do problema e o projeto detalhado do sistema revelam eventuais falhas e omissões, os quais são solucionados antes da implementação do sistema. Portanto a correção de erros no desenvolvimento orientado a objetos é, normalmente, menos dispendiosa e mais fácil de ser executada.

3.2.1 Objetos e classes de objetos

O objeto é uma representação, através de um conjunto de dados e de métodos, de um conceito ou elemento do domínio da aplicação, como dito anteriormente. Para o conceito de objeto ser pleno, deve-se entender que os dados representam os atributos do objeto – uma

estrutura que se limita a descrevê-lo conceitualmente – e seus valores determinam seu estado atual. Por sua vez, os métodos são os procedimentos e as funções afetos ao domínio do elemento com o objetivo de garantir que os estados se comportarão conforme o conceito real e independentemente das ações externas. Tais métodos fornecem o controle interno do estado e das informações junto com as funcionalidades que podem ser obtidas através destes. A Figura 12 é uma forma acadêmica de visualizar o conceito de objeto, nela os atributos do objeto são encapsulados por um conjunto de métodos que dão acesso à informação, fornecem funcionalidades e controlam o seu estado interno.

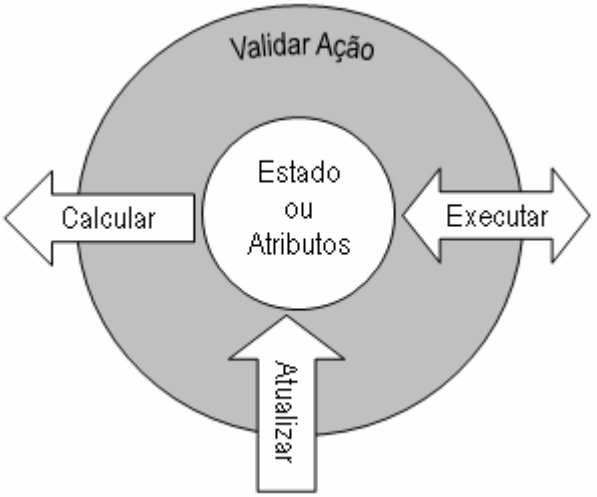


Figura 12 – Dados encapsulados

A descrição abstrata da estrutura do objeto é conhecida como classe de objeto. A classe representa a estrutura ou a forma com a qual os objetos serão criados, ou melhor, instanciados. O corpo humano possui uma estrutura complexa que é própria do ser humano. Porém existe uma nítida diferença entre a pessoa propriamente dita e a estrutura do corpo humano, pois a estrutura é algo que o define como ser humano e a pessoa é a individualização da espécie. A relação entre classe de objetos e objeto é análoga. Quando se define uma classe, determina-se qual a estrutura que os objetos oriundos desta possuirão. Quando se cria um objeto, aloca-se uma quantidade de memória equivalente àquela estrutura e define-se que esta memória será regida pelos métodos implementados na classe. Assim, a cada criação, uma nova quantidade de memória é alocada e ganha um corpo funcional.

O presente trabalho, com intuito de exemplificar os princípios da POO, passa a apresentar um software para desenhos básicos que utiliza formas geométricas simples – muito utilizado na literatura técnica como exemplo. O objetivo da aplicação é criar uma figura livre composta por diferentes elementos de desenho, conforme mostra a Figura 13.

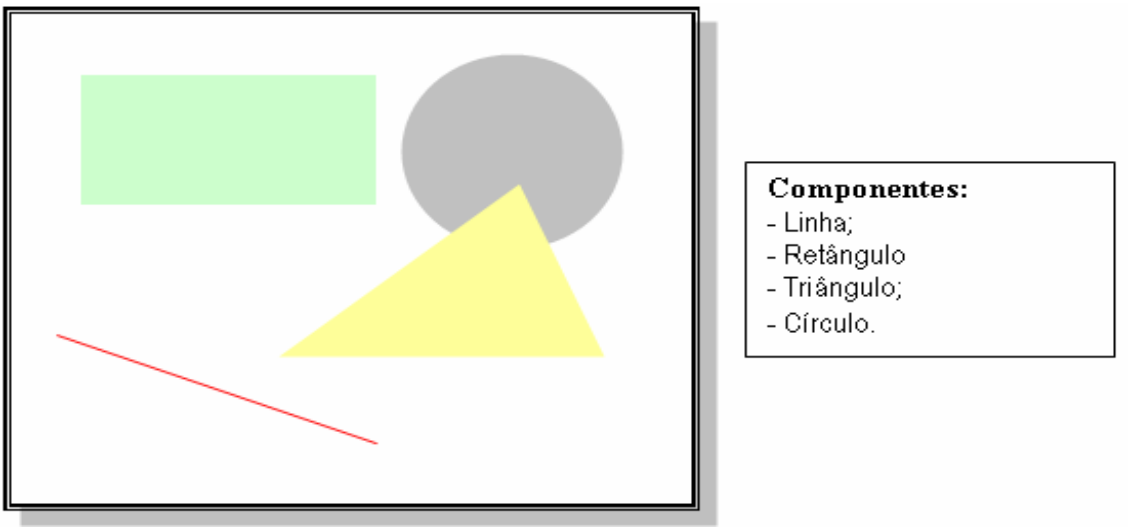


Figura 13 – Software para desenhos básicos.

Parte de um possível modelo para solucionar o problema é apresentada na Figura 14 . Neste modelo parcial os elementos que constituem o domínio da aplicação são claramente identificados e definidos. Cada classe de objeto possui uma estrutura que caracteriza o objeto a ser instanciado, não havendo definições impróprias que extrapolem seu limite conceitual. A MOO purista prima pela não incorporação de características que surjam com a única finalidade de simplificar o trabalho de implementação, prejudicando o modelo real. Entretanto, existem casos que tal incorporação é aceita restritamente para não gerar um trabalho excessivo e soluções inadequadas quanto aos requisitos do software.

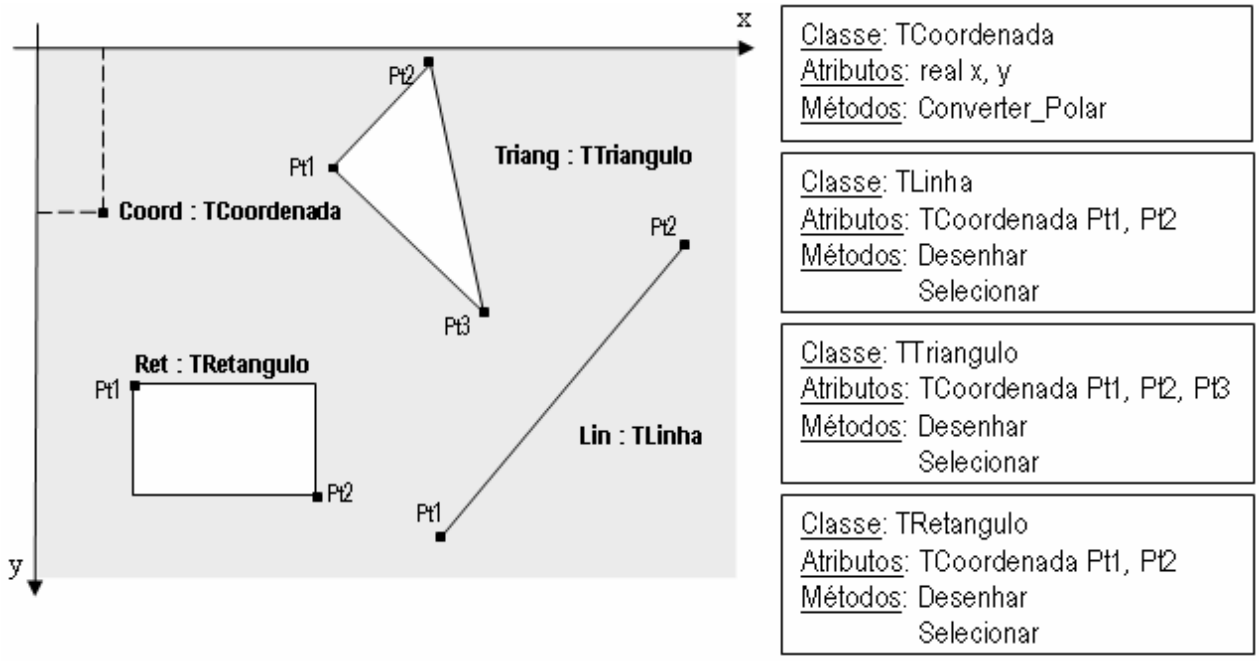


Figura 14 – Modelo parcial para software de desenho

A princípio, a identificação dos objetos e a decomposição do problema parece ser uma tarefa simples, mas, na verdade, esta atividade é mais difícil no desenvolvimento de sistemas complexos. Mesmo em problemas simples, como o apresentado, a modelagem requer o entendimento dos requisitos e depende da criatividade e experiência do projetista.

Os desenhos são, na realidade, realizados dentro de uma área gráfica que depende da linguagem adotada, bem como do sistema operacional. Entretanto, sem a necessidade de defini-la no momento, cabe ressaltar que o método “Desenhar” utiliza esta área como parâmetro, assim o desenho é feito dentro da área externa informada. Por sua vez, o método “Selecionar” informa se uma dada coordenada passada como parâmetro pertence ou não ao domínio geográfico do objeto. Tais métodos serão utilizados por outros objetos na realização de suas tarefas, sem prejuízo de sua utilização interna quando necessário. Porém os métodos e os atributos que outros objetos terão acesso serão aqueles definidos como públicos, uma vez que existem métodos que somente são de uso interno e são restritos ao próprio objeto, tais métodos são conhecidos como privados (possuindo variantes como os métodos protegidos). Os atributos privados ou protegidos somente podem ser alterados externamente através de métodos públicos que controlam sua modificação.

Os objetos se comunicam entre si através de mensagens, as quais desencadeiam a execução de métodos internos ou externos e a interação com outros objetos de seu domínio para a realização de uma dada tarefa. Entretanto tais as mensagens são, na verdade, métodos declarados, as quais – nas primeiras linguagens POO – eram definidas explicitamente. Atualmente existe o sistema de mensagens que é próprio do sistema operacional, o qual vincula um dado tipo de mensagem a uma gama de objeto que declararam a utilização deste tipo de em sua estrutura. Porém, internamente, cada mensagem está vinculada à execução de um método específico e de formatação conhecida. Grande parte da comunicação entre os objetos é feita através de eventos, os quais são ponteiros de funções conhecidas que são executados quando uma dada ação é realizada.

3.2.2 Herança

Uma das grandes inovações geradas pela POO consiste na capacidade de uma classe de objetos de herdar as características de uma outra classe existente. Tal característica, por óbvio, é conhecida como hereditariedade e proporcionou um grande avanço no desenvolvimento de sistemas. Com o aparecimento desta característica, uma dada classe pode herdar a estrutura de uma outra classe e inserir novas características que lhe são próprias. Tal advento foi

benéfico tanto para a manutenção do sistema, pela clareza com a qual as diferentes classes se originam, quanto para o próprio desenvolvimento, pela estruturação e a descrição que se pode obter do problema a ser modelado.

As classes que são originadas pela herança são conhecidas como classes descendentes. Por sua vez, as classes antecessoras são aquelas que herdam suas características para as descendentes. Desta forma, cria-se uma hierarquia construtiva entre as diferentes classes do domínio da aplicação, sendo as classes antecessoras mais genéricas – agrupando características e funcionalidades em comum – e as classes descendentes mais específicas – voltadas para as características do problema e sua solução. Muitas linguagens de programação, como o C++, possibilitam a herança múltipla que permite a junção de duas ou mais classes bases para formar uma classe descendente.

3.2.3 Polimorfismo

Os objetos criados com a utilização de classes descendentes podem assumir os formatos de todas as suas classes antecessoras, ou seja, sempre há a possibilidade de regressão hierárquica. Assim, um objeto criado a partir de uma classe B (descendente da classe A) pode ser tipificado e utilizado como um objeto da classe A, estando limitado à estrutura de A enquanto não sofrer a mudança de tipo para B. Assim pode-se criar uma lista de objetos da classe A formada por instancias de classes descendentes, como B, C e D.

Existem funções especiais, conhecidas como funções virtuais, que possibilitam a modificação de suas implementações numa classe antecessora quando redefinidas numa classe descendente. Portanto, se uma função virtual “Desenha” implementada na classe A for redeclarada para executar uma tarefa diferente numa classe B, esta função será também modificada nas instancias de B que trabalham tipificadas como A. Assim ao executar “Desenha” num objeto originado de B, mas sendo tipificado como A, executará a tarefa implementada na classe B e não a implementada na classe A. Tal característica é conhecida como polimorfismo [12], na qual uma mesma classe pode possuir diferentes implementações que dependem das funções virtuais declaradas/herdadas e de suas modificações nas classes descendentes. Portanto o polimorfismo é uma característica intrinsecamente associada à herança.

3.2.4 Exemplificando a herança e o polimorfismo

O modelo completo do núcleo do problema proposto pode ser visualizado na . A classe *TElemento* é a classe antecessora das demais classes que representam elementos de desenho.

Nela são declaradas as características que os elementos de desenho devem apresentar, sendo que todos devem ter uma função que os desenham numa área indicada e todos devem possuir uma função que indica quando eles foram selecionados. Tais métodos são virtuais, indicando que serão modificados por classes descendentes. Assim cada classe descendente modifica cada um dos métodos para condizer com sua característica própria.

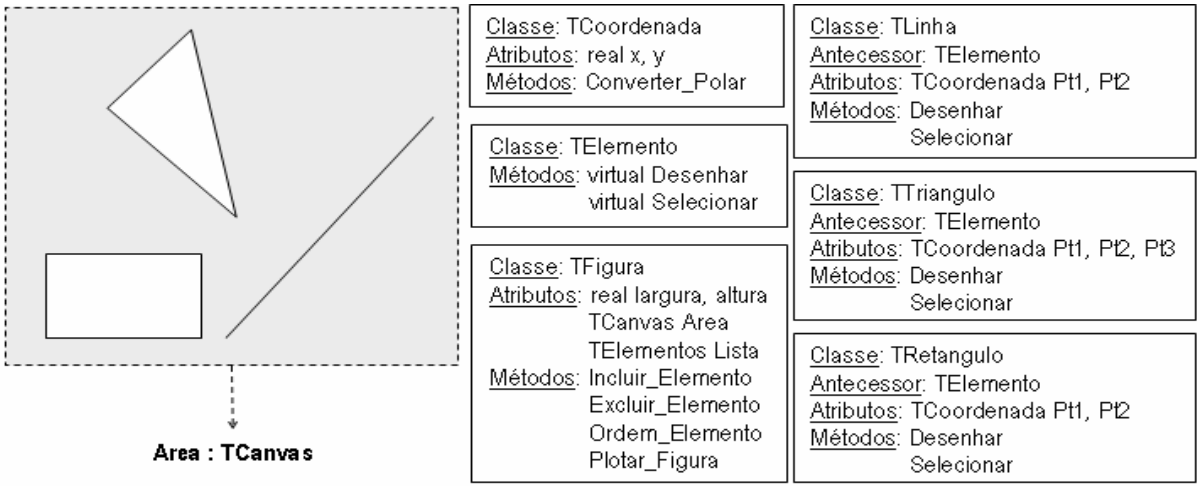


Figura 15 – Modelo do núcleo problema (software de desenho simples)

A classe *TFigura* possui uma lista de instancias ou de objetos da classe *TElemento*, uma vez que a figura é a coleção de elementos de desenho (conceito identificado). Porém, para a classe *TFigura*, não importa qual é o desenho que será feito por cada elemento, esta somente necessita dos métodos da classe *TElemento*. Quando for necessário, a figura irá chamar "Desenha" de cada elemento passando sua área de impressão como parâmetro, formando assim a imagem desejada. Os demais métodos da classe *TFigura* são referentes ao cadastro dos elementos, seleção do elemento e sua ordem na lista de impressão. Os objetos externos voltados para a aplicação (não mostrados no modelo) serão encarregados de criar e de cadastrar cada elemento conforme a solicitação do usuário.

3.3 Métodos de Projeto Orientado a Objeto e a origem da UML

Os modelos exemplificativos apresentados abordam o problema da modelagem de forma rústica, haja vista que pouca informação pode ser retirada da forma que estes se apresentam. Durante as últimas duas décadas houve uma grande preocupação com os métodos de projeto e a modelagem orientada a objetos, direcionando os estudos ao devido tratamento que a tecnologia requeria. A filosofia da orientação a objetos não se limita à aplicação das técnicas de

programação orientada a objetos, esta engloba todos os conceitos e técnicas inerentes ao processo de desenvolvimento de software ou processo de software. Assim diferentes etapas do processo de software necessitavam de mudanças para contemplar de forma eficiente a nova filosofia de desenvolvimento, bem como o desenvolvimento de novos modelos que abordassem de forma satisfatória as características da OO.

Algumas metodologias propostas durante o final da década e 80 e a primeira metade da década de 90 ganharam reconhecimento tanto da comunidade científica quanto da comunidade de desenvolvedores de sistemas, iniciando um grande debate sobre as vantagens e desvantagens de cada uma delas. Na metade da década de 90, com a união dos propositores de três diferentes metodologias, iniciou-se o desenvolvimento de um padrão unificado denominado *Unified Modeling Language* ou UML, a qual teve a contribuição de diferentes segmentos da indústria do software em suas versões posteriores [8]. Atualmente a UML é a linguagem-padrão utilizada no processo de software orientado a objetos, reconhecimento dado pelo OMG (*Object Management Group*) em 1997 – grupo que se assumiu a responsabilidade por sua evolução. Atualmente já existe uma versão melhorada da UML, conhecida como UML 2.

O presente trabalho não possui o objetivo de detalhar as metodologias supracitadas, entretanto se faz necessário apresentar os conceitos básicos da UML para a compreensão dos modelos que serão apresentados nos próximos capítulos. Para tanto os métodos que deram origem à UML serão brevemente comentados antes da introdução aos conceitos da UML.

3.2.1 Booch

A popularidade do método de Booch [5] é devido à amplitude e à diversidade de seus modelos e à sua origem centrada nas linguagens de programação orientadas a objetos. A metodologia empregou técnicas de desenho orientadas a objeto para a descrição das diferentes visões do sistema. Cada visão é descrita por um conjunto de diagramas, sendo esta pertencente ou a sua modelagem lógica ou a sua modelagem física.

O processo de software se inicia com a fase de análise de requisitos, onde são levantados, definidos e analisados os requisitos que o sistema deve apresentar. A fase seguinte compreende a modelagem lógica do sistema, denominada análise do domínio do problema, na qual são identificados, definidos e especificados os objetos e as suas relações – resultando em diagramas estáticos e dinâmicos. Por fim, a última fase do processo de software foi denominada de projeto físico, a qual engloba – dentre outras atividades – o projeto de software, a implementação, a descrição modular do sistema e a alocação de processos.

Para efetuar todo o processo de especificação do sistema em desenvolvimento, utilizam-se os diagramas estáticos de classes, de objetos, de módulos e de processos, e os diagramas dinâmicos de transição de estados e interação. Tais diagramas serão comentados em conjunto com a UML, cabendo salientar que estes sofreram pequenas modificações – inclusive na nomenclatura adotada.

3.2.2 OMT (Object Modeling Technique)

A OMT [2] é um método de projeto orientado a objetos constituído por diferentes modelos que formam uma perspectiva estrutural, dinâmica e funcional do sistema. Segundo os próprios autores, o sistema é dividido em visões ortogonais que podem ser trabalhadas e representadas por uma notação uniforme. As visões são constituídas por modelos que evoluem iterativamente durante o ciclo de desenvolvimento do sistema. A metodologia compõe-se de seguintes fases:

- **Análise:** Preocupa-se com a compreensão e a modelagem da aplicação e do domínio em que atua. É constituída pelo modelo de objetos (diagramas de objetos e dicionário de dados), onde é descrita uma estrutura estática dos objetos, suas classes e relacionamentos; pelo modelo dinâmico (diagramas de estado e diagramas de eventos), onde são representados os aspectos de controle do sistema; e pelo modelo funcional (diagramas de fluxo de dados), onde são descritas as transformações das funções sobre os dados.

- **Projeto do sistema:** Descreve a arquitetura básica do sistema, organizando o sistema em subsistemas – definindo suas funcionalidades, dividindo-os em processos e alocando tais processos aos processadores. Nesta fase são feitas decisões estratégicas de alto nível, como plataforma da aplicação, linguagem de programação e qual sistema de gerenciamento de banco de dados será utilizado.

- **Projeto dos objetos:** Desenvolve, refina e otimiza os modelos da fase de análise para a produção de um projeto base. Esta fase é constituída pelos mesmos modelos da fase de análise, porém mais detalhados – aproximando-os da implementação.

- **Implementação:** Nesta fase é feito o projeto da base de dados, através transcrição de um modelo de objetos para projeto de um banco de dados, e o mapeamento do projeto orientado a objeto para uma linguagem de programação – ou seja – uma codificação.

A OMT apresenta as seguintes características: (1) desloca o esforço de desenvolvimento para a análise; (2) enfatiza a estrutura de dados e não as funções; (3) compõe um processo de desenvolvimento contínuo e iterativo.

3.2.3 OOSE (Object-Oriented Software Engineering)

A OSSE [4] tornou-se popular pela a sua maneira simples e completa de se especificar o sistema utilizando o conceito de *use cases*. Um *use case* é um cenário de uso particular do sistema proposto, o qual descreve a relação entre um ou mais agentes (usuários, outros sistemas, equipamentos, etc.) e o comportamento ou o resultado esperado do sistema frente a alguma ação ou evento externo causado por este agente. O *use case* representa, portanto, um diálogo entre um ator (executor externo de interações) e o sistema – capturando alguma funcionalidade pela representação da ação esperada do sistema. O conjunto final de *use cases* especifica de forma completa o sistema proposto.

A metodologia descreve o sistema através dos seguintes modelos: o modelo de requisitos ou especificações (capta a funcionalidade requerida do sistema sob a perspectiva dos usuários); o modelo de análise (define a estrutura de objetos do sistema e suas relações); o modelo de projeto (refina o modelo de análise em termos do ambiente de implementação); o modelo de implementação (converte os blocos refinados para as características da linguagem escolhida); e o modelo de teste, que permitirá testar o sistema através dos requisitos identificados no primeiro modelo.

3.3 UML – A unificação dos métodos

Uma instabilidade assombrou o universo da orientação a objetos durante o início da década de 90. Tal fato era resultado da diversidade de metodologias de projeto existentes – em torno de 50 no início de 1994 –, as quais fragmentavam o mercado, geravam incompatibilidades e dificultavam o aprendizado [13]. No meio deste iminente risco, a unificação dos métodos obteve apoio imediato da comunidade de desenvolvedores. A UML não propôs algo novo, esta,

simplesmente, visava unir as melhores práticas existentes no plano teórico. O intuito era gerar uma metodologia que pudesse ser aplicada na modelagem de qualquer tipo de sistema e que fosse usável e entendível tanto para o homem quanto para a máquina, sendo ou não o software um elemento deste desenvolvimento. A UML se baseou em conceitos sólidos e amplamente testados, bem como gerou uma documentação [13] bem definida de sua semântica.

A UML, além de possuir um processo de desenvolvimento de sistema, é composta por diferentes partes, as quais, resumidamente, são:

- **Visões:** A visão é um conjunto de diferentes diagramas (modelos) que mostram os aspectos particulares do sistema segundo uma determinada perspectiva.

- **Modelo de Elementos:** São os elementos apresentados graficamente nos diagramas que representam os diferentes conceitos que integram a orientação a objetos, como as classes, os objetos, as associações, dentre outros;

- **Mecanismos Gerais:** São mecanismos que provém informação ou comentários adicionais aos diagramas, bem como proporcionam a extensão da linguagem para casos e modelos particulares que necessitam de adaptações;

- **Diagramas:** São os modelos gráficos propriamente ditos que formam as diferentes visões, onde cada um proporciona um conjunto de informações sobre um aspecto particular do sistema e de seus componentes.

3.3.1 O processo unificado de desenvolvimento em UML

O processo de desenvolvimento de software – chamado de processo de software – consiste nas diferentes atividades, organizadas em fases ou etapas, necessárias para um desenvolvimento organizado e controlado do produto. O apêndice A resume objetivamente diferentes metodologias de processo de software existentes, estando baseado no excelente trabalho apresentado em [10].

O surgimento da UML solucionou a denominada “guerra dos métodos” consolidando uma notação unificada e proporcionando uma documentação e uma modelagem adequadas para o sistema, entretanto as empresas tinham que adaptar a metodologia ao seu próprio processo de software. O processo unificado de desenvolvimento [9], conhecido como *Rational Unified*

Process (RUP), veio preencher essa lacuna. Trata-se de uma evolução do *Rational Objectory Process* (ROP), proposta por Ivan Jacobson na metodologia *Objectory* – uma abordagem mais completa da OOSE.

O RUP é um modelo de processo de desenvolvimento genérico, baseado em componentes e desenvolvido para dar suporte a UML. O processo é dirigido por *use cases* e centrado na arquitetura do sistema – favorecendo o paralelismo de processos dentro do processo principal. Trata-se de um processo incremental que disponibiliza versões intermediárias ao usuário, onde cada versão corresponde a um ciclo de desenvolvimento. Cada ciclo é dividido em 4 fases que priorizam objetivos diferentes. Por sua vez, cada fase é composta por um conjunto de iterações – onde cada interação representa uma seqüência de atividades (*workflow*) que resulta num incremento de sistema (artefato). A Figura 16 ilustra o ciclo de desenvolvimento e a concentração de trabalho de cada atividade nas diferentes fases/ iterações.

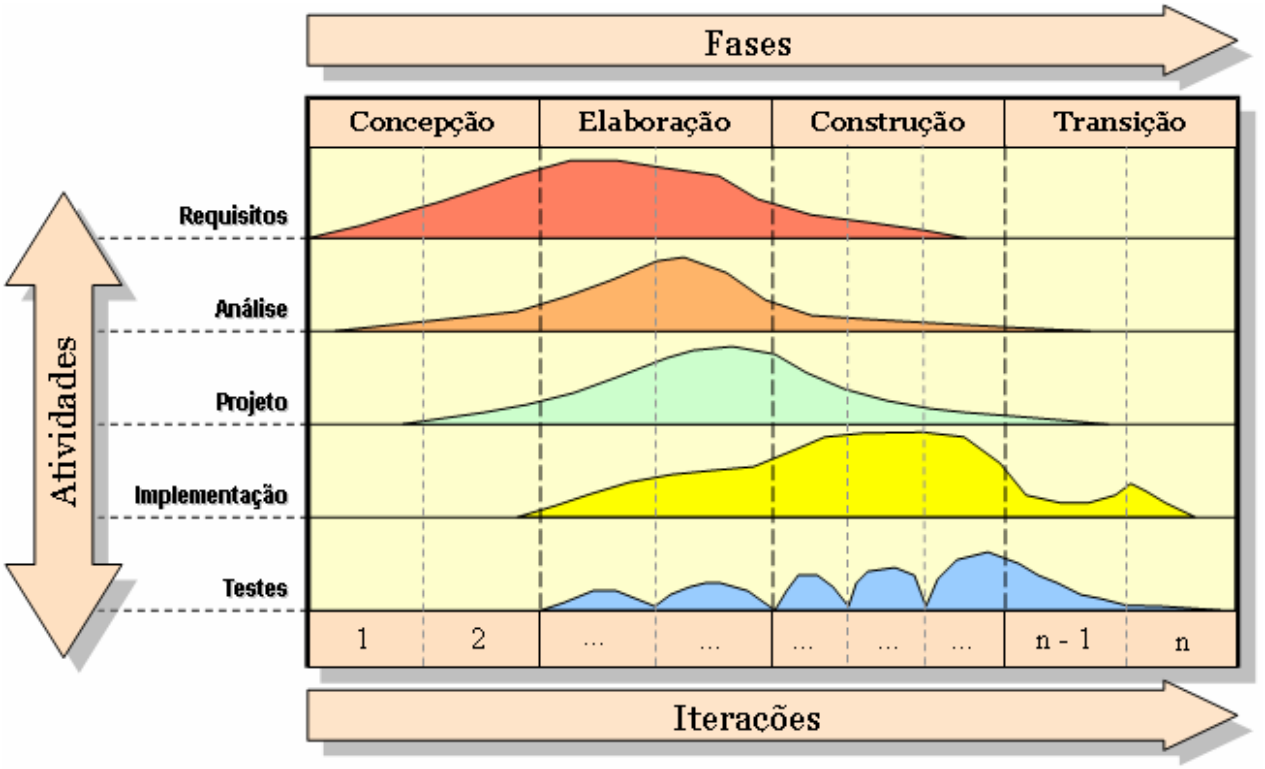


Figura 16 – Fases, iterações e atividades de um ciclo de desenvolvimento do RUP.

As fases orientam o trabalho a ser desenvolvido, sendo concebidas de forma genérica e tendo suas atividades adaptadas ao ciclo de desenvolvimento. Uma fase possui quase todas as

atividades, dando maior ou menor ênfase as atividades relacionadas ao seu objetivo. Os objetivos principais das diferentes fases são apresentados a seguir:

- **Concepção ou Iniciação:** Trata-se da compreensão do problema e delimitação do projeto, verificando a pertinência do ciclo. Estimam-se os custos, o tempo e o retorno do investimento. Identificam-se e mitigam-se os riscos inerentes ao ciclo. Formula-se uma arquitetura geral e cria-se um planejamento inicial baseado nas estimativas;

- **Elaboração:** Enfatiza o levantamento e a análise dos requisitos através dos use cases. Desenvolve-se ou adapta-se a arquitetura do sistema solidamente para guiar as fases posteriores. Monitoram-se os riscos e seus impactos, e desenvolve-se e refina-se o projeto;

- **Construção:** Prioriza o desenvolvimento, pautando-se no projeto do sistema, para gerar ou evoluir a versão executável que será disponibilizada ao cliente. Detalhes técnicos omissos não capturados ou irrelevantes nas atividades anteriores emergem gerando atualizações;

- **Transição:** Adaptações necessárias são levantadas pelos usuários resultando num desenvolvimento adicional para atender ou modificar alguns requisitos solicitados.

As atividades seqüências de cada iteração, por sua vez, possuem objetivos bem definidos, conforme exposto abaixo:

- **Levantamento e análise dos requisitos:** Captura os requisitos do sistema através das *use cases*, identificando os autores externos e as funcionalidades que o sistema deve apresentar. Uma visão geral do sistema pode ser obtida através da análise preliminar dos requisitos e restrições, resultando na descrição abstrata da arquitetura, modelagem abstrata das principais interfaces com o usuário e resultante delimitação do sistema. A técnica auxilia o acordo entre desenvolvedores e clientes com o uso da descrição de cenários e seus documentos servem de base contratual;

- **Análise:** Desenvolve uma especificação mais precisa dos requisitos e restrições. Criam-se os modelos estáticos e dinâmicos voltados ao domínio do problema. Os modelos estáticos descrevem a estruturação do núcleo do problema em classes, objetos, mecanismos e suas relações. Os modelos dinâmicos apresentam o comportamento do sistema, classes e

objetos em diferentes aspectos funcionais. Os modelos de análise são abstratos, ou seja, estão distantes das decisões técnicas de projeto, eles modelam o domínio do problema e especificam o sistema com base nos requisitos levantados – sendo a especificação técnica postergada;

- **Projeto (Design):** Expande o modelo da análise para incorporar a solução técnica. Novas classes são adicionadas para prover a infra-estrutura do sistema (base de dados, comunicação, interface), mesclando-se com as classes que descrevem o domínio do problema. Os modelos passam a especificar detalhes técnicos do sistema para orientar a implementação do sistema;

- **Implementação (Programação):** Converte as especificações detalhadas do sistema em códigos da linguagem de programação escolhida. Trata-se de uma tarefa quase mecanizada, sem muitas complicações, porém a facilidade desta atividade depende da linguagem escolhida e independência do projeto do sistema. O modelo de projeto deve descrever a solução técnica sem considerar ou ser influenciado pela linguagem de implementação, assegurando a descrição real do sistema. Os testes de responsabilidade dos programadores (testes de unidade, classe e módulo) são realizados dentro da atividade de implementação;

- **Testes:** O sistema passa por testes para verificar sua correta integração, o atendimento aos requisitos e comportamento perante situações reais de operação. Os testes de integração verificam o correto acoplamento dos diferentes módulos e subsistemas, bem como verificam o atendimento aos requisitos que só podem ser analisados na operação sistêmica (propriedades emergentes). O teste de aceitação exercita o sistema com dados reais de operação e, por vezes, simula o ambiente operacional.

3.3.2 Os aspectos do sistema representados pelas diferentes visões

Utopicamente, o modelo ideal deve descrever todas as informações e detalhes do sistema numa única representação gráfica livre de ambigüidades e de fácil entendimento. Tal hipótese é impossível de ser concretizada devido à quantidade e à diversidade de informações e de detalhes que um sistema possui. A simplicidade, a completude e o detalhamento só podem

coexistir com o uso da decomposição organizada da informação que envolve o problema, sendo paradoxal o modelo ideal supracitado.

As visões, inseridas na UML, são uma forma organizada de mostrar os diferentes aspectos do sistema. Cada visão prioriza um tipo de informação e é composta por um conjunto de diagramas relacionados a ela. Os diagramas não pertencem a uma dada visão, podendo estes compor uma ou mais visões sempre que necessário. A UML possui as seguintes visões:

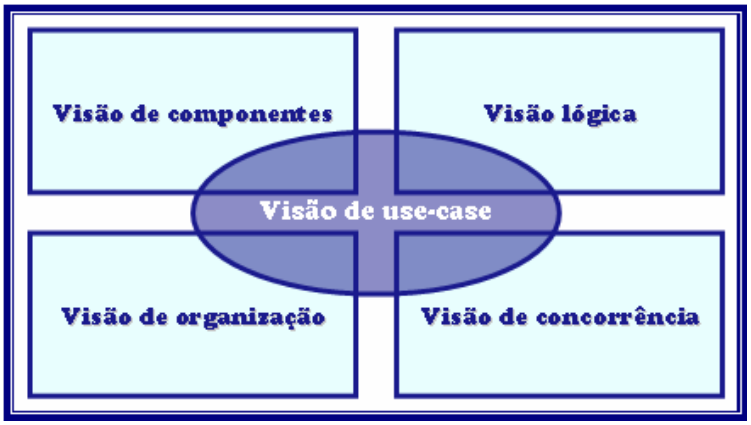


Figura 17 – Visões da UML

- **Visão de "use-case":** Composta pelo diagrama de use-case e, eventualmente, o diagrama de atividades. Descreve as funcionalidades do sistema baseada na interação deste com os autores externos. Trata-se da visão central, uma vez que seu conteúdo é fundamental para o desenvolvimento das demais visões;

- **Visão lógica:** Descreve a estrutura estática e dinâmica do sistema. A estrutura estática inclui a descrição das classes, dos objetos e seus relacionamentos através do diagramas de classes e objetos. A estrutura dinâmica retrata o comportamento dos objetos e suas interações através do diagrama de estados, seqüência, colaboração e atividade. Trata-se de uma visão interna do sistema.

- **Visão dos componentes:** Descreve a implementação mostrando a decomposição do sistema em pacotes de componentes, componentes (módulos ou arquivos do sistema como a DLL ou o EXE), interfaces e as suas dependências;

- **Visão de concorrência:** Descreve os diversos processos do sistema e suas respectivas alocações nos processadores. Mostra a existência de processos paralelos, suas

comunicações e do controle de eventos assíncronos. A visão é composta pelos diagramas de estado, de seqüência, de colaboração, de atividade, de componentes e de execução.

- **Visão de organização:** Descreve a organização física do sistema, ou seja, os computadores, os periféricos, a rede de computadores e suas respectivas conexões. Representado pelo diagrama de execução.

3.3.3 Modelos de elementos

Os modelos de elementos são descrições formais e claras – representados através de desenhos simples – dos diferentes conceitos que tangem a orientação a objetos e a sua modelagem, abrangendo todos os elementos que compõem os diferentes diagramas. Um elemento pode existir em diferentes diagramas, entretanto existem regras que restringem sua utilização. As classes, os objetos, os estados de um objeto, os diferentes tipos de relações, os pacotes e os componentes são alguns exemplos de modelos de elementos. Os principais modelos são apresentados a seguir.

3.3.3.1 Classes e objetos

A classe é representada por um retângulo dividido em três compartimentos, conforme mostra a Figura 18. O primeiro compartimento é obrigatório em qualquer nível de detalhamento, pois este identifica a classe através de seu nome. O segundo compartimento mostra os atributos da classe e, opcionalmente, seus respectivos tipos e valores iniciais. O último compartimento agrupa o conjunto de métodos que manipulam dados, proporcionam a comunicação com outras classes e exteriorizam as funcionalidades da classe. Os métodos podem opcionalmente ter o valor de retorno, a descrição dos parâmetros e classificação de seu tipo.

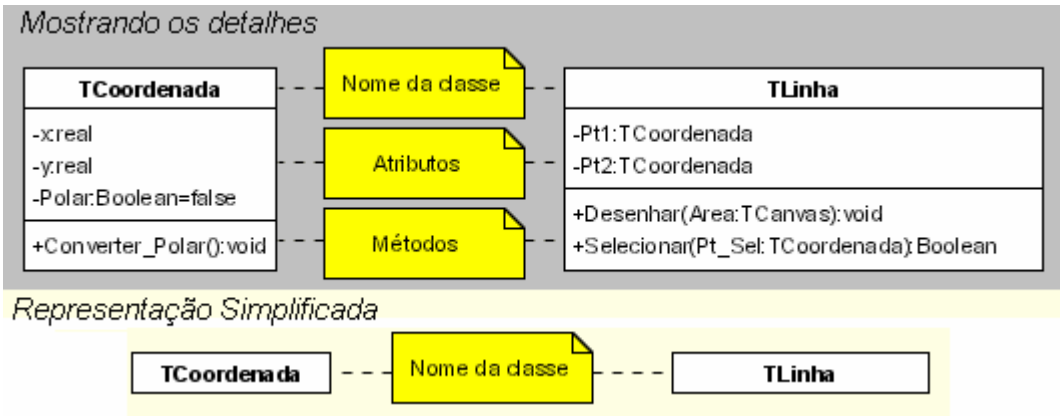


Figura 18 – Representação gráfica detalhada e simplificada das classes em UML

O objeto aproxima-se da representação da classe, entretanto este possui uma identidade e valores para seus atributos que demonstram sua situação num dado instante. Ele é utilizado em diagramas que mostram o perfil de certos momentos da execução do sistema – podendo ser criado, manipulado e destruído. A diferença entre a representação do objeto e da classe é dada pelo nome do objeto (seguido da identificação da classe) que está sublinhado e a exposição dos valores dos seus atributos, quando for o caso, conforme a Figura 19.

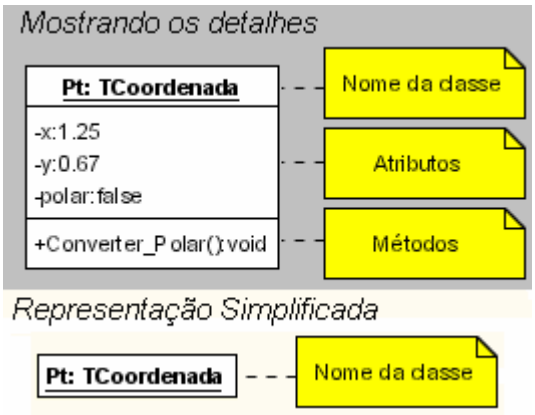


Figura 19 - Representação gráfica detalhada e simplificada dos objetos em UML

3.3.3.2 Estados

Os estados de um objeto refletem os valores de seus atributos e a seqüência de acontecimentos (eventos) ocorridos. O estado é importante para visualizar o comportamento do objeto num dado instante e sua possibilidade de mudança e ação. O modelo de um estado, mostrado na Figura 20, pode possuir três compartimentos: o primeiro mostra o nome do estado e o identifica (obrigatório); O segundo (opcional) agrupa os atributos relevantes e mostra seus

valores, podendo até mostrar variáveis temporárias importantes para um dado estado; O terceiro, opcional, mostra o conjunto de atividades executadas dentro daquele estado, ou seja, lista o conjunto de ações e eventos executados – podendo estar dividido por etapas como as atividades ao entrar no estado, durante a permanência e ao sair do estado.

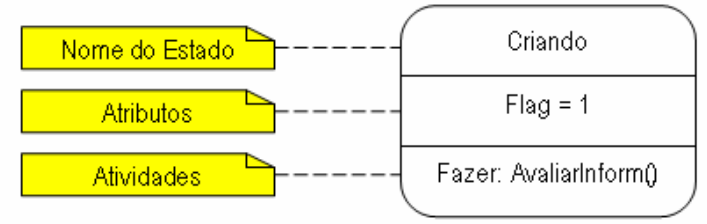


Figura 20 – O estado através da UML

3.3.3.3 Pacotes

Os pacotes são agrupamentos de elementos segundo sua a pertinência funcional e organizacional. Cada elemento é inserido em um único pacote, sendo possível uma importação referenciada. Os pacotes podem agrupar outros pacotes e possuir relações de dependência, refinamento e generalização (herança). A Figura 21 exibe a estrutura do modelo de um pacote.

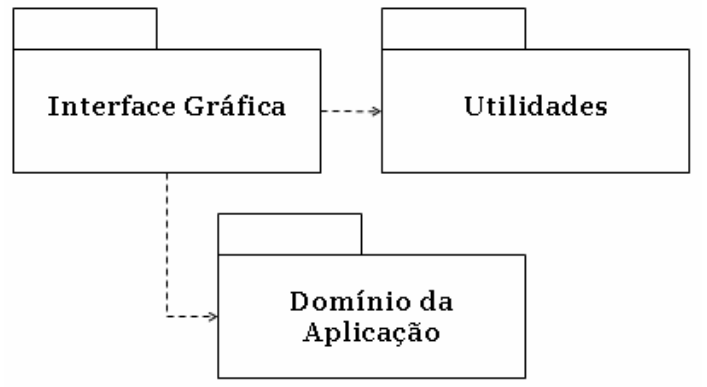


Figura 21 – Representação de pacotes e suas dependências em UML

3.3.3.4 Componentes

O componente é a representação de módulos, bibliotecas e arquivos que compõem, resultam ou são utilizados dentro da implementação do sistema. Qualquer arquivo que tenha algum vínculo com o sistema, seja uma biblioteca de vínculo dinâmico (DLL) ou um arquivo executável (EXE), é considerado um componente. A Figura 22 exemplifica o exposto.

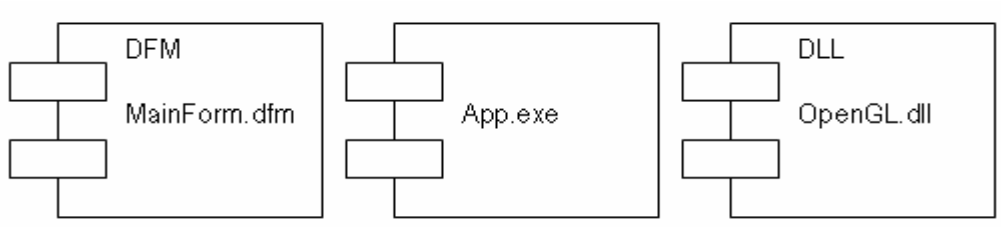


Figura 22 – Exemplos de componentes em UML

3.3.3.5 Relacionamentos

Os relacionamentos são representados através das ligações entre as entidades de um dado diagrama. Estes possuem significados distintos e características próprias que dependem do tipo de relação existente. Os principais relacionamentos são:

- **Associação**: Indica uma conexão direta entre classes, ou seja, um tipo ligação conceitual entre as classes. A associação demonstra que pelo menos uma das classes envolvidas conhece a outra e a possui como atributo, sem prejuízo para a associação bidirecional – quando ambas se conhecem e se utilizam. O fato de possuir a classe como atributo não qualifica o relacionamento como uma espécie de composição, o qual será visto a seguir. A Figura 23 exemplifica alguns tipos de associações existentes – a seta indica que a associação somente pode ser usada pela classe que aponta (associação unidirecional);

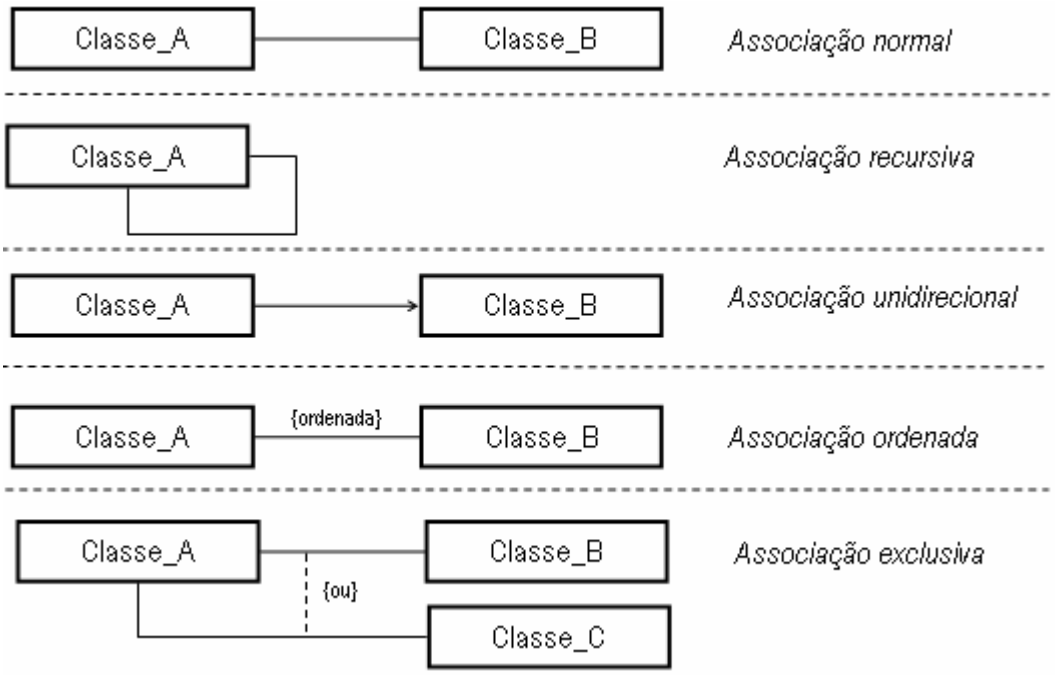


Figura 23 – Diferentes tipos de associações em UML

- **Dependência**: Caracteriza uma utilização propriamente dita, ou seja, quando uma classe utiliza (através de parâmetros, por exemplo) outra classe sem possuí-la como atributo, ou seja, sem possuir uma ligação direta. A dependência mostra que alterações numa classe (independente) afetam a outra classe (dependente). A Figura 24 mostra a relação de dependência.

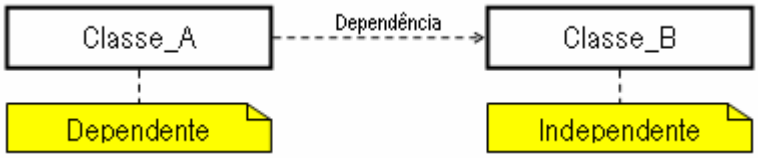


Figura 24 – Relação de dependência entre classes.

- **Agregação**: É um tipo particular de associação, a qual indica que uma classe é composta por uma ou mais classes. Uma agregação indica que uma classe faz parte da composição de outra, sendo possível compor uma classe através de diversas classes diferentes. Na agregação compartilhada, a classe componente pode ser parte de uma ou mais classes compostas - onde a criação, o gerenciamento e a destruição da classe componente possuem controle externo. Na agregação de composição, a classe componente pertence à classe composta – na qual a classe componente é criada e destruída pela classe composta. A Figura 25 ilustra a agregação.

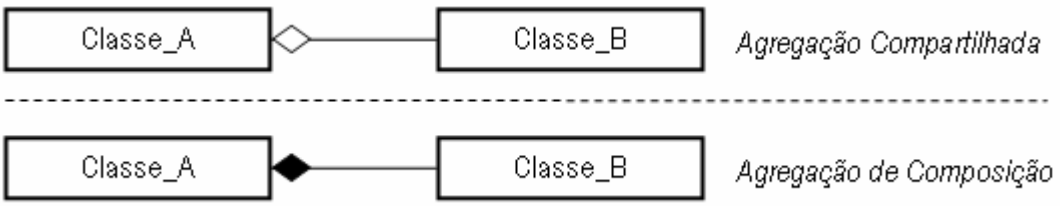


Figura 25 – Agregação de classes em UML

- **Generalização**: Trata-se da especialização de uma classe mais geral resultando numa outra classe mais específica. Também denominada de herança, a generalização permite que uma classe herde às características de outra e acrescente funcionalidades, atributos e características próprias. Como discutido anteriormente, um objeto mais específico pode ser usado como instância de uma classe mais geral (dentro das limitações já comentadas). A Figura 26 mostra os alguns tipos possíveis de generalizações.

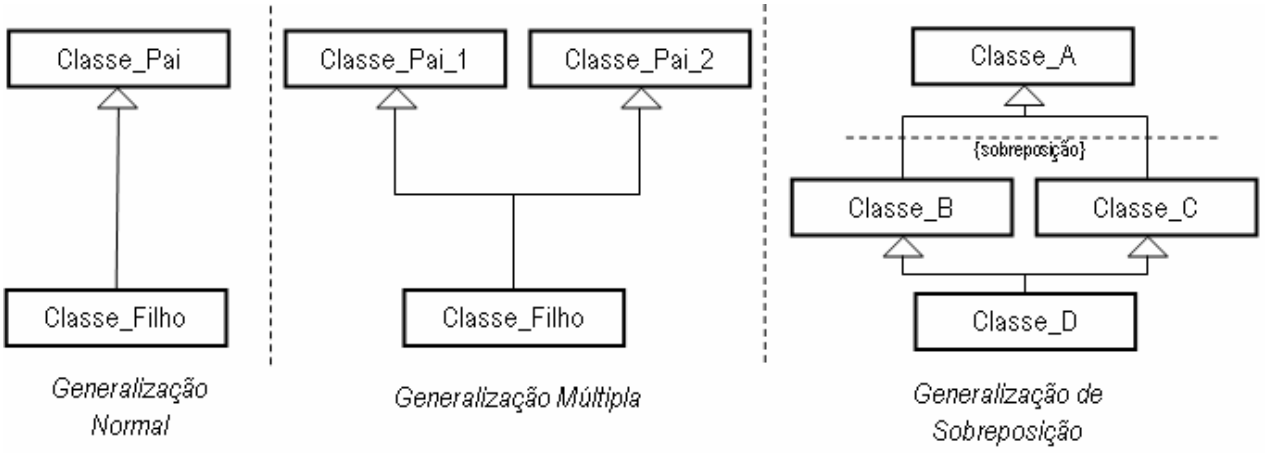


Figura 26 – Diferentes tipos de generalização (herança) representados em UML

3.3.3.6 Classes Templates

Algumas linguagens proporcionam a possibilidade de se especificar o tipo de algumas variáveis internas durante a utilização da classe ou da estrutura. A tipificação da variável somente ocorre durante a compilação do programa, dando versatilidade ao código e ao elemento. Tal técnica é denominada de *template* e proporciona a postergação da definição completa do elemento. Os tipos não especificados são definidos no momento da sua utilização, sendo passados como parâmetro. A Figura 27 exibe a representação de *templates* em UML.

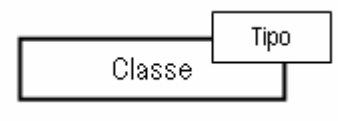


Figura 27 – Templates em UML

3.3.4 Mecanismos gerais

A UML utiliza alguns mecanismos em seus diagramas para tratar informações adicionais.

- **Ornamentos:** Existe uma grande diversidade de ornamentos na UML, exemplos são os ornamentos de especificação de multiplicidade de relacionamentos, onde a multiplicidade é um número ou um intervalo que indica quantas instâncias de um tipo conectado pode estar envolvido na relação.

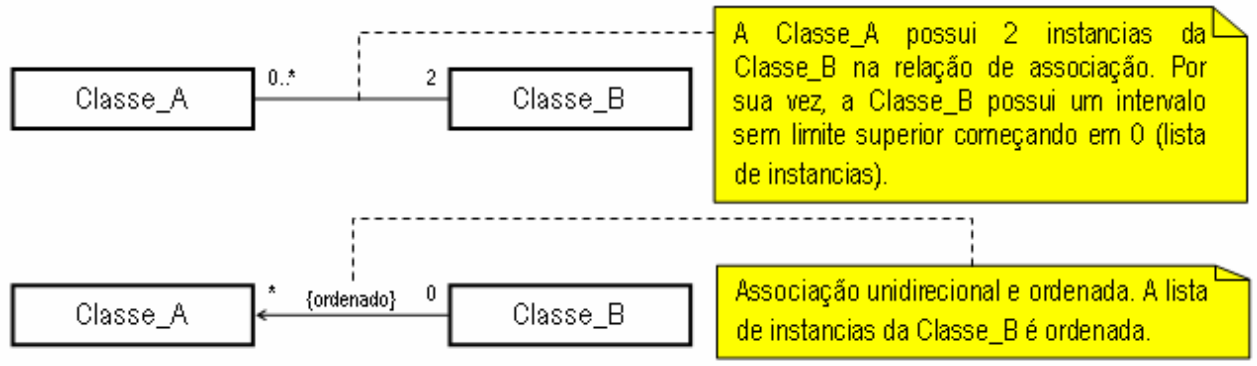


Figura 28 – Multiplicidade e indexação como exemplo de ornamentos

- **Notas:** Nem tudo pode ser definido em uma linguagem de modelagem, independentemente de sua extensão. Para permitir a adição de informações a um modelo, as quais não poderiam ser representadas de outra forma, UML provê a capacidade de adicionar notas. Uma nota pode ser colocada em qualquer lugar em um diagrama e pode conter qualquer tipo de informação. A nota destaca-se dentro do diagrama, possui formato próprio e está ligada àquilo que pretende detalhar ou informar através de uma linha pontilhada, como mostrada na Figura 28.

3.3.5 Diagramas

A UML possui nove tipos diferentes de diagramas que compõem três subgrupos diferentes: os diagramas estáticos mostram a estrutura do sistema e suas relações internas; os diagramas dinâmicos enfatizam o comportamento do sistema e de seus componentes; e os diagramas funcionais. As principais características de cada diagrama são mostradas a seguir.

3.3.5.1 Diagrama de caso de uso ou use-case

O diagrama use-case emprega a técnica de análise de cenários para descrever e definir os requisitos do sistema e suas funcionalidades. Nele são capturadas as funcionalidades esperadas com base na relação entre o sistema e os autores externos (usuários, sistemas externos, dispositivos). Um determinado cenário de utilização mostra quais são as funções (*use-cases*) que um autor externo pode realizar dentro do sistema, bem como a descrição destas funções, através da documentação anexa ao caso, contendo os dados de entrada, o conjunto de ações do sistema, os dados de saída e os comentários. O conjunto de diagramas de caso de uso especifica todas as funcionalidades que o sistema deve apresentar. A Figura 29 mostra um diagrama de caso de uso, sendo que cada *use-case* mostrada deve ser descrita através de um formulário apropriado.

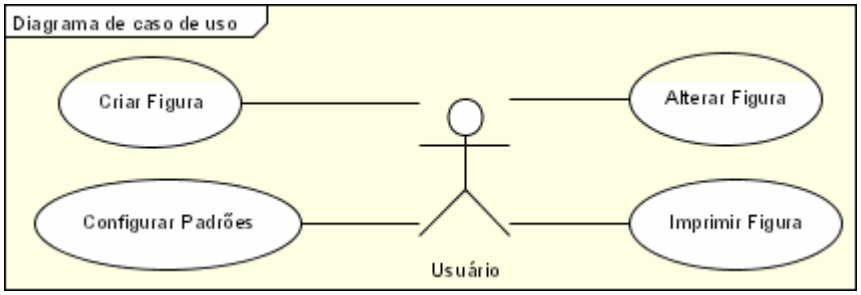


Figura 29 – Diagrama de caso de uso ou *use-case*

3.3.5.2 Diagrama de classes

O diagrama estático de classes mostra o conjunto de classes que compõem o domínio da aplicação, bem como os relacionamentos existentes entre elas. O nível de detalhamento do diagrama depende tanto do estágio que se encontra o processo de software como das pessoas que utilizarão suas informações. Pode haver o agrupamento de classes em determinados

diagramas devido ao tamanho e prioridade da informação mostrada. Tais agrupamentos são, na verdade, pacotes que serão detalhados em outros diagramas de classes. A Figura 30 exemplifica um diagrama.

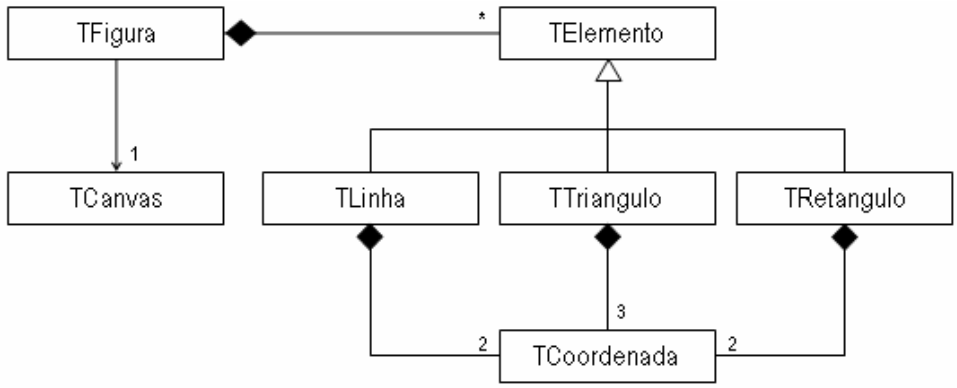


Figura 30 – Diagrama de classes

3.3.5.3 Diagrama de objetos

O diagrama estático de objeto visa fornecer uma visão diferenciada do diagrama de classes. Tal diagrama prima pelo detalhamento da estrutura do sistema em execução, ou seja, trata-se de um perfil executório do sistema nos moldes mostrados no diagrama de classes. Ao invés de classes, o diagrama de objetos mostra os objetos instanciados e suas relações num determinado momento ou ponto de execução. Além do valor dos atributos internos, o diagrama de objetos mostra e detalha todas as relações existentes e não só a multiplicidade da relação – como ocorre no diagrama de classes. Trata-se de uma exemplificação do diagrama de classes que muitas vezes pode ser suprimido dependendo de sua complexidade.

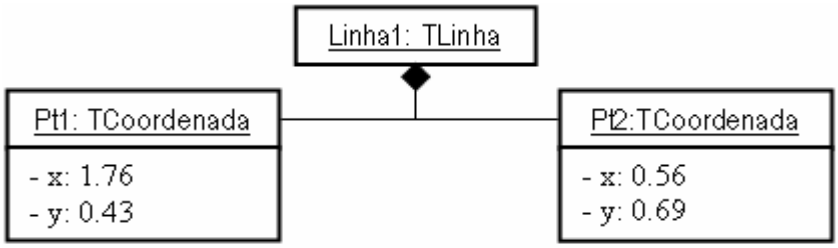


Figura 31 – Exemplo de diagrama de objetos para um caso simplificado

3.3.5.4 Diagrama de estados

O diagrama de estados visa complementar o diagrama de classes dando uma perspectiva dinâmica das classes descritas. Este diagrama tem o objetivo de mostrar os possíveis estados que um objeto pode possuir; as atividades executadas em diferentes etapas do estado (ao entrar, durante e ao sair); e quais são os eventos que geram a transição entre os estados. Tal modelo assemelha-se com o conceito de máquina de estados.

Somente as classes que possuem um número finito de estado e um perfil comportamental compatível podem ser descritas por este diagrama. O ponto inicial do diagrama é representado por um círculo totalmente preenchido e o ponto final (se houver) é representado por um círculo vazado. Os eventos são representados através do texto adjunto às setas de transição. A Figura 32 exemplifica o diagrama.

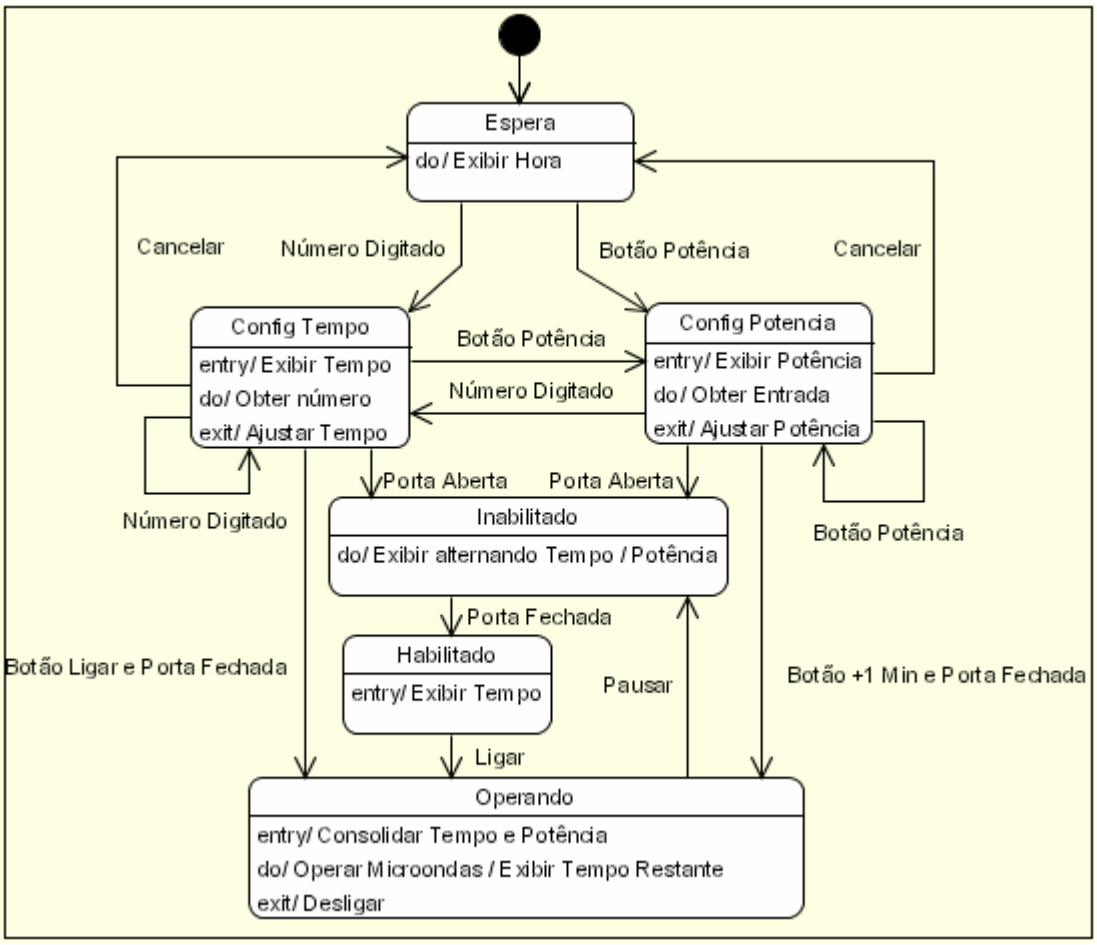


Figura 32 – Diagrama de estados modelando o comportamento de um forno de microondas.

3.3.5.5 Diagrama de seqüência

O diagrama de seqüência visa apresentar o conjunto de colaborações dinâmicas entre os diferentes objetos relacionados a uma tarefa do sistema. As interações entre os objetos são mostradas através das mensagens enviadas, as quais estão dispostas na seqüência temporal de ocorrência. O conjunto de objetos relacionados à tarefa é apresentado no eixo horizontal do diagrama, dispostos por ordem de participação e representados por um retângulo e uma linha vertical pontilhada denominada de *linha de vida*. O controle temporal é apresentado no eixo vertical do diagrama, enfatizando o objeto que possui o controle do processo em determinado momento. A Figura 33 mostra um exemplo de diagrama de seqüência.

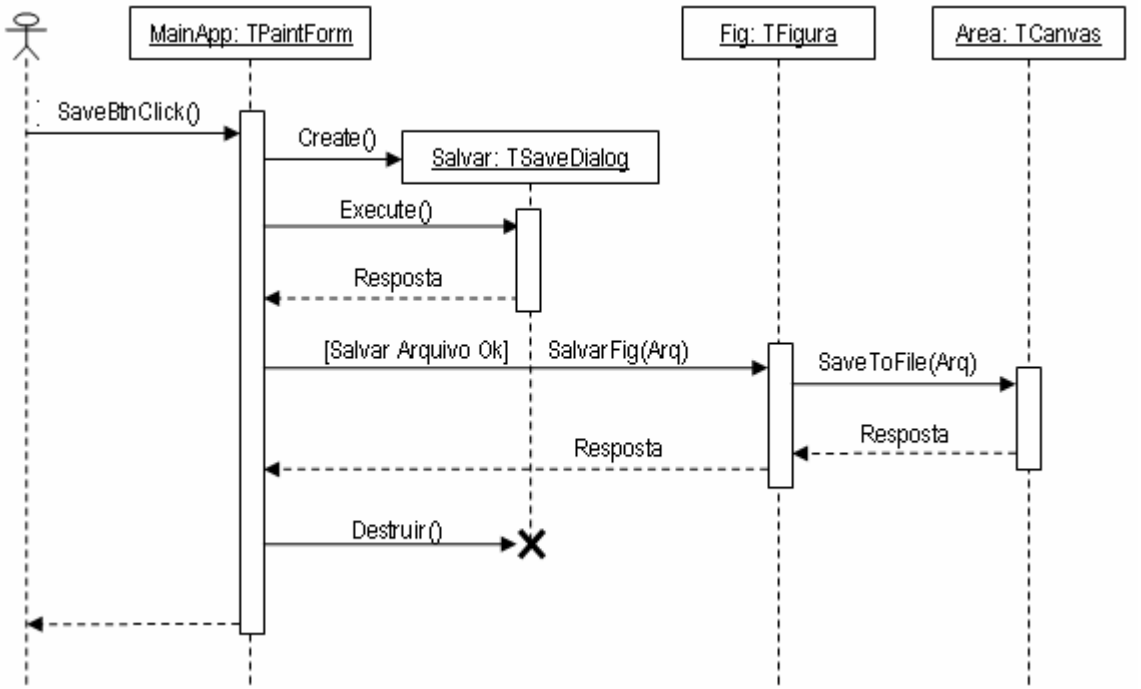


Figura 33 – Exemplo de diagrama de seqüência

3.3.5.6 Diagrama de colaboração

O diagrama de colaboração possui o mesmo objetivo que o diagrama de seqüência, entretanto este diagrama enfatiza a estrutura dos relacionamentos existentes entre os objetos. As mensagens são nomeadas como no diagrama de seqüência e possuem uma identificação numérica que classifica a sua ordem de ocorrência e setas que indicam o sentido de execução. A mensagem é apresentada ao lado do relacionamento existente e não possui controle temporal

do processo. Tanto o diagrama de seqüência quanto o de colaboração podem incluir agentes externos que iniciam a tarefa apresentada. A Figura 34 mostra o diagrama de colaboração para o caso mostrado na Figura 33.

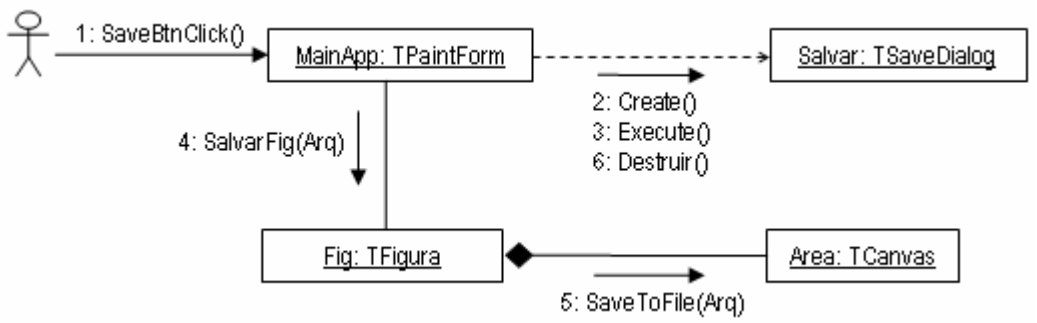


Figura 34 – Diagrama de colaboração para o mesmo caso do diagrama de seqüência mostrado.

3.3.5.7 Diagrama de atividade

O diagrama de atividade se assemelha muito ao tradicional fluxograma, apesar de possuir uma notação gráfica incompatível. Este diagrama apresentada o conjunto de atividades, organizados de forma seqüencial e lógica, que envolve uma determinada tarefa. Suas principais diferenças em relação ao fluxograma é a habilidade de trabalho com objetos, capacidade de divisão e organização da tarefa entre diferentes responsáveis, possibilidade de modelar processos concorrentes e possibilidade de apresentar as mensagens enviadas e recebidas.

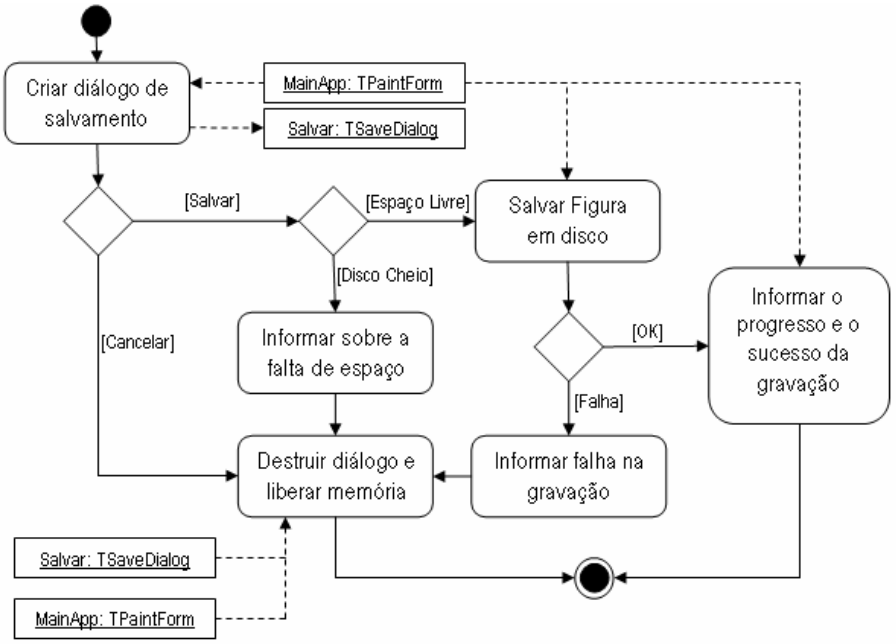


Figura 35 – Diagrama de atividades.

Conforme mostra a Figura 35, as atividades são representadas como um retângulo arredondado; os objetos gerados (indicados) ou utilizados (indicando) em cada atividade são mostrados como retângulos; a divisão ou a união de processos é feita através de uma barra horizontal; o condicionamento lógico é representado por um losango; o início é representado por um círculo cheio; e o término é representado por um círculo com um círculo cheio interno.

3.3.5.8 Diagrama de componentes

O diagrama de componentes descreve a decomposição do sistema em diferentes tipos de módulos e suas respectivas dependências. Trata-se de um diagrama funcional que descreve a organização física do sistema, representando a estrutura do código gerado, ou seja, a coletânea de arquivos que compõem o sistema em seu ambiente de desenvolvimento. Dentre os diferentes tipos de arquivos (módulos), somente os executáveis podem ser instanciados no diagrama de execução (diferentes processos).

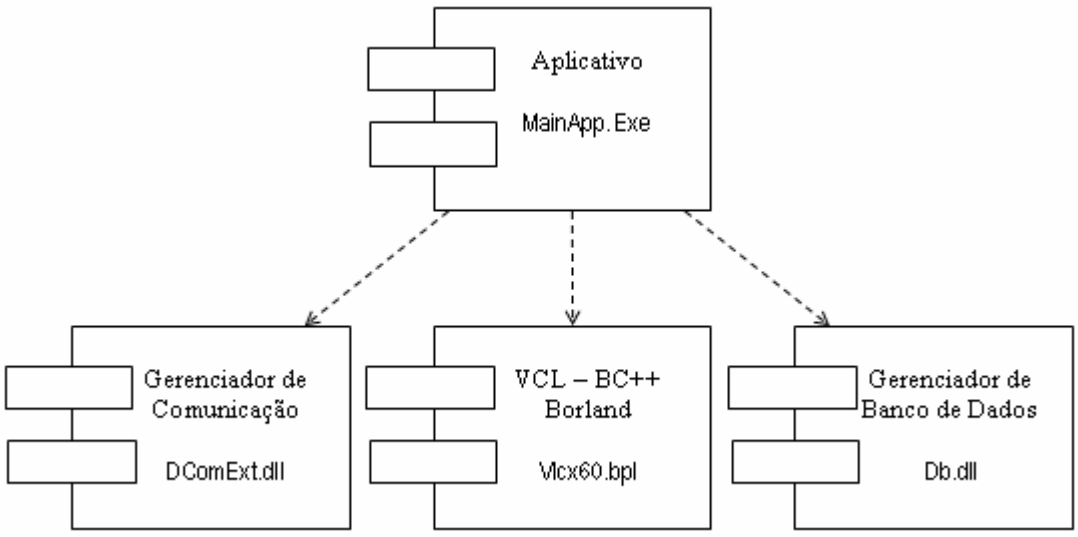


Figura 36 – Diagrama de componentes em UML

3.3.5.9 Diagrama de execução

O diagrama de execução tem o objetivo de descrever o ambiente computacional, enfatizando a disposição dos computadores, periféricos e a suas formas de conexão, bem como a alocação dos diferentes processos e objetos nos diversos processadores ou computadores. O diagrama de execução demonstra a arquitetura *run-time* de processadores, os componentes físicos (*devices*) e de software que rodam no ambiente onde o sistema desenvolvido será

utilizado. É a última descrição física da topologia do sistema, descrevendo a estrutura de hardware e software que executam em cada unidade.

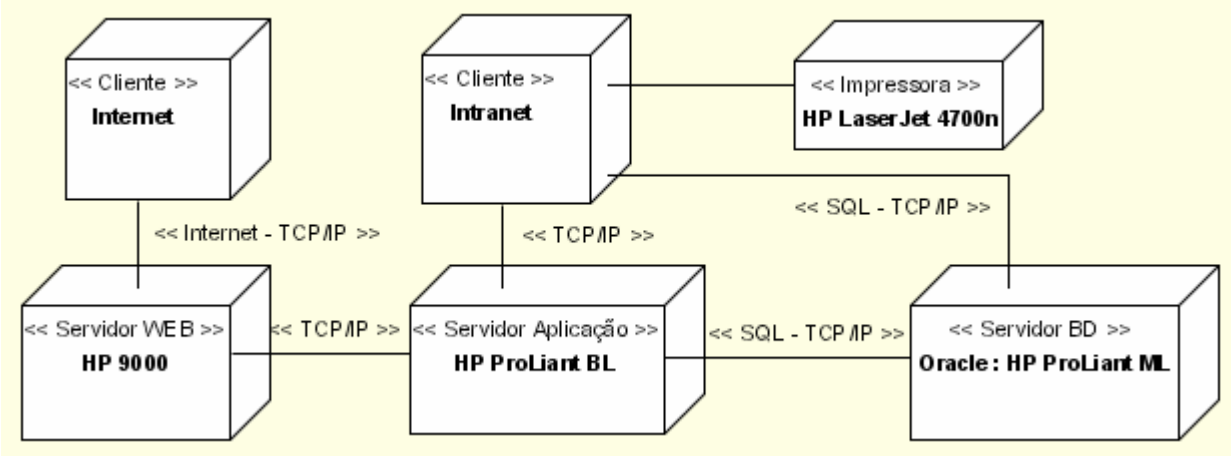


Figura 37 – Exemplo de diagrama de execução

Capítulo 4

O Projeto Orientado a Objetos de Estimadores de Estados para Sistemas Elétricos de Potência.

4.1 Considerações Gerais

O projeto de software necessita de prazo, recursos humanos, recursos materiais e, sobretudo, recursos financeiros adequados para um desenvolvimento satisfatório. Outras limitações, como o contexto de desenvolvimento do projeto, causam impactos nas decisões de projeto e, consequentemente, na modelagem do software proposto. O projeto de software, devido às limitações orçamentárias e cronológicas, sofre as adaptações necessárias para atingir com êxito os objetivos traçados contratualmente. Portanto o resultado final de um projeto não pode ser atribuído somente à metodologia de desenvolvimento, pois existem fatores contratuais que influenciam mais que qualquer prática adotada.

Existem áreas de aplicação nas quais diferentes estudos estão inter-relacionados ao núcleo do problema e, muitas vezes, tais estudos devem ou podem se complementar. Os softwares para sistemas elétricos de potência possuem essa característica, todos dependem em maior ou menor grau do modelo de SEE utilizado e da forma com que este foi computacionalmente modelado. As inflexibilidades no modelo implementado geram grandes custos para a futura evolução do projeto de software, as quais - por vezes - implicam na remodelagem parcial ou total do sistema. Uma modelagem mais aprimorada que vise atender diversos tipos de estudos não pode ser elaborada sem os recursos e as técnicas adequados. Entretanto a quantidade e a qualidade dos recursos necessários advêm da técnica utilizada no processo de software.

As ferramentas computacionais para sistemas de energia elétrica, historicamente, sempre foram especializadas, ou seja, voltadas à solução de um conjunto pequeno de problemas específicos e correlacionados. A maioria destas ferramentas era implementada utilizando linguagens estruturadas – grande parte delas foi desenvolvida em FORTRAN – e não possuía o conceito de reutilização de software. Sempre que uma mudança significativa emergia, tal como um novo modelo de componente do sistema, um grande esforço era despendido na

adaptação do software, o qual muitas vezes precisava ser reimplementado. Tal característica é contrária ao dinamismo do setor elétrico, o qual constantemente sofre mudanças com o surgimento ou a alteração de novos estudos, metodologias ou modelos de equipamentos. Outros fatores importantes são as mudanças no modelo setorial e a eficiência técnico-econômica almejada e fomentada pelo novo mercado de energia elétrica, que impõem mudanças rápidas de cenários e exige uma grande adaptabilidade de seus agentes. Tal contexto implica na mudança de filosofia de desenvolvimento de ferramentas computacionais para atender as exigências das características do setor – tendo a agilidade e a flexibilidade como características essenciais do software, e a integração e a interação de ferramentas como diretrizes de desenvolvimento.

O presente capítulo apresenta o contexto de desenvolvimento do projeto de estimador de estados robusto aplicado ao sistema de transmissão de energia da companhia estadual de energia elétrica do Rio Grande do Sul (CEEE), bem como seu impacto no modelo de sistema elétrico adotado. A arquitetura básica do projeto é introduzida para guiar o detalhamento e as relações entre os modelos que compõem o sistema. Os modelos atuais de SEE são comentados brevemente com o intuito de mostrar o estado da arte, sua evolução e sua tendência. Posteriormente, apresentam-se os motivos e a limitações que levaram a adoção de um modelo direcionado. Por fim são mostradas as particularidades do projeto que tangem à estimação de estados em sistemas elétricos de potência.

4.2 Arquitetura básica

A arquitetura do software de estimação de estados, mostrada na Figura 38, fornece uma visão geral do projeto. Sua introdução tem o objetivo de tornar claros os principais pontos que serão discutidos e detalhados no âmbito deste trabalho. Como se pode notar, os principais conjuntos de classes estão agrupados em pacotes para facilitar o entendimento das relações existentes e para fornecer uma visão simplificada do projeto.

As interfaces gráficas são responsáveis pela interação entre o software e o usuário, fornecendo os mecanismos necessários para uma perfeita articulação entre as ações externas, os procedimentos internos e as respostas esperadas. Trata-se do conjunto de formulários (janelas) e diálogos que compõem a camada externa, ou melhor, a ‘casca’ do software. A interface gráfica orienta a utilização da ferramenta e possibilita a inserção um conjunto de procedimentos para o controle do fluxo lógico. Normalmente, por serem classes intermediárias entre a comunicação homem-máquina, as interfaces são dependentes de um grande conjunto

de classes que formam o núcleo do problema ou compõem a infra-estrutura do software. Boa parte das classes infra-estruturais compõe as próprias interfaces gráficas (como botões ou editores de textos), entretanto a relação de agregação não é claramente mostrada por se considerar inerente ao caso. Entretanto a dependência entre estas classes é explícita – mostrando que, dentre os conjuntos de classes relacionados à linguagem de desenvolvimento, as interfaces gráficas são as mais dependentes. Dentro do escopo do projeto, as interfaces gráficas são dependentes das ferramentas matemáticas, do modelo de SEE, dos filtros e dos aplicativos/funções para SEE. Tal dependência deriva da sua função primária de exibir as informações relacionadas ao domínio da aplicação.

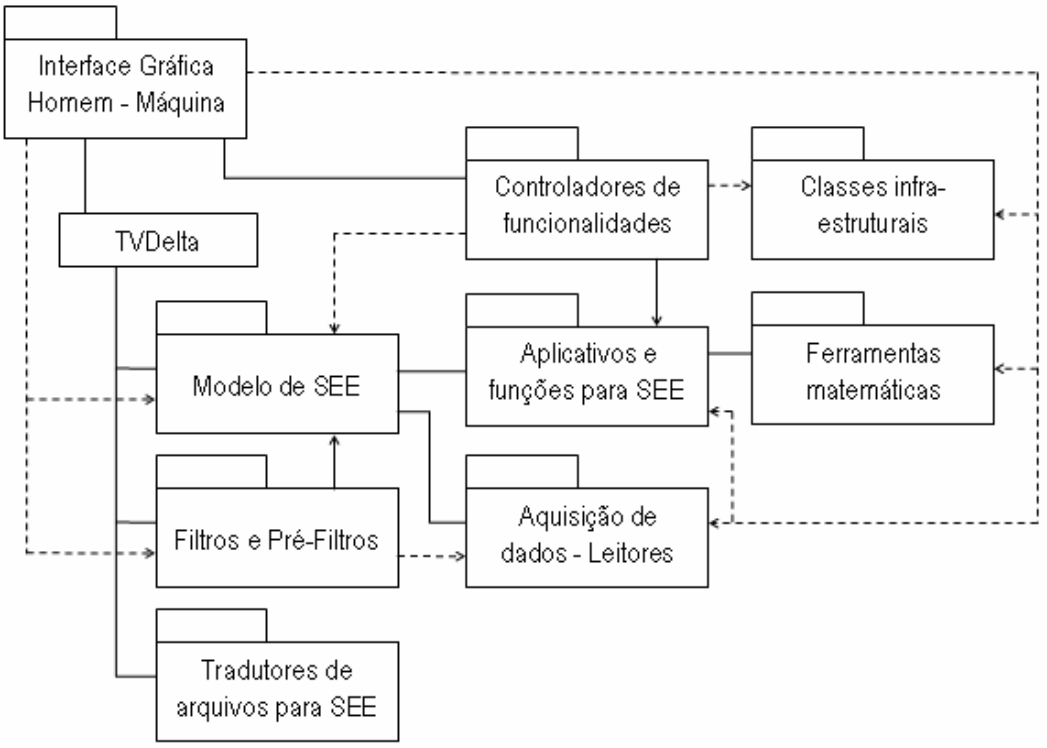


Figura 38 – Arquitetura básica do projeto

O *TVDelta* é única classe exibida neste diagrama que não faz parte de nenhum pacote, pois possui a característica peculiar de ser responsável pelo cadastro dos objetos do modelo do SEE e dos filtros e pré-filtros aplicados nos estudos. Trata-se de uma classe organizacional com o objetivo controlar a criação, a manipulação e a destruição dos objetos, fornecendo acesso àquilo que foi criado através de diferentes tipos de listas. Não se enquadra na característica de nenhum pacote mostrado na arquitetura do EE, apesar de sua proximidade ao modelo de SEE. Não pode ser confundido com o objeto representativo do SEE (*TSistema*), que será mostrado nas próximas seções, pois o TVDelta é voltado ao cadastro dos elementos independente da

condição do objeto (criação completa ou incompleta) e do seu tipo – representando o estudo carregado e possuindo, inclusive, a possibilidade de se cadastrar conjuntos independentes de sistemas elétricos. Para fornecer a possibilidade de manipulação de tipos distintos de arquivos descritores de sistemas, foram desenvolvidas classes específicas de leitura – agrupadas no pacote de tradutores de arquivos para SEE – de utilização interna da classe *TVDelta*.

As classes que descrevem equipamentos, dispositivos ou qualquer elemento físico ou conceitual que tenha relação com o SEE pertencem ao modelo de SEE. Internamente o modelo de SEE pode ser dividido em outros modelos como o modelo do sistema elétrico de potência, do sistema de monitoramento e comando ou do sistema de proteção. Será visto que a literatura técnica possui atualmente bons modelos de SEE que podem ser aplicados a quase todos os tipos de estudos. Entretanto, independentemente do modelo computacional empregado, a sua função é descrever adequadamente o sistema e seu comportamento, subsidiando as informações necessárias direta ou indiretamente. O modelo de SEE desenvolvido neste trabalho contempla o SCADA, entretanto a independência deste com relação às diferentes tecnologias empregadas para aquisição de dados é alcançada pelo uso de leitores. Tais leitores compatibilizam as diferentes tecnologias utilizadas no SCADA com o padrão interno de entrada de dados, possibilitando a aplicação da mesma estrutura de classes em empresas que usam tecnologias diferentes de aquisição de dados.

As funções e os aplicativos para SEE são estudos específicos realizados com as informações obtidas do modelo de SEE, gerando resultados que podem ou não modificar os atributos dos objetos do modelo (como os estados em EE). Cada função tem o objetivo de alcançar um resultado específico de acordo com a parametrização informada. Os controladores de funcionalidades, por sua vez, são interfaces responsáveis tanto pela configuração da função quanto pela realização de estudos que envolvem mais de uma função. São os controladores que mesclam diferentes funções para alcançar um resultado específico ou para gerar seqüências temporais de estudos. Portanto há uma distinção entre as funções (como a estimação de estados ou o fluxo de potência) e sua execução coordenada pelos controladores (como a seqüência operacional: EE, FP, análise de contingências, previsão de carga de CP).

As ferramentas matemáticas são classes independentes direcionadas ao tratamento e a manipulação de matrizes, de vetores e de solução de sistemas lineares. Tais classes podem ser aplicadas a qualquer tipo de projeto que necessite de suporte matemático mais elaborado, não necessariamente aqueles afetos ao setor elétrico. Por óbvio, em seu desenvolvimento, as ferramentas sofreram a influência da estimação de estados e das características intrínsecas do tratamento matemático dentro de SEE. Entretanto não houve prejuízo na sua finalidade, ao

contrário, houve um refinamento e adoção de metodologias eficientes que durante décadas foram discutidas na literatura técnica.

4.3 MOO aplicada em SEE

O objetivo deste trabalho não é apresentar uma modelagem genérica de sistema elétrico de potência e sim projetar um estimador de estados direcionado a sistemas de transmissão realísticos. Entretanto não se pode separar a aplicação do domínio do problema. Portanto, sem demais aprofundamentos, se faz mister comentar os modelos existentes na literatura técnica e o modelo elaborado para este trabalho – explicando os motivos das simplificações e da especialização adotados. Cabe comentar que os atuais modelos orientados a objeto [14–18] são genéricos e bem elaborados, permitindo o desenvolvimento de quase todas as funções e estudos que se baseiem em sistemas elétricos de potência. Apesar da boa qualidade dos atuais modelos elaborados, a orientação a objetos ainda não possui um vasto número de trabalhos apresentados nesta área de aplicação.

O emprego da orientação a objetos como técnica de desenvolvimento de ferramentas computacionais para SEE foi conveniente, devido à característica do problema a ser modelado, porém, diferentemente de outras áreas, ocorreu tardiamente. A demora no emprego desta tecnologia foi motivada pela complexidade do problema, pela necessidade de velocidade, pela cultura de linguagens voltadas à aplicação científica e pela própria evolução dos sistemas operacionais e do hardware – os quais direcionavam a linguagem em razão dos requisitos da ferramenta a ser desenvolvida. A indústria do FORTRAN, assim conhecida pelo extenso conjunto de ferramentas computacionais implementadas em FORTRAN, vigorou durante décadas e enraizou suas características na maioria dos desenvolvedores voltados ao setor elétrico. Entretanto os primeiros estudos mostrando a aplicação da POO [19, 21-22] já abordavam as questões da inflexibilidade, da falta de robustez e do alto custo da manutenção dos projetos implementados com linguagens estruturadas.

4.3.1 Revisão bibliográfica

As técnicas de POO foram aplicadas ao problema de fluxo de potência pela primeira vez por Neyer et alii [19] no início da década de 90. Este trabalho propunha a modelagem do sistema elétrico através de duas diferentes perspectivas: a modelagem física descrevia a

estrutura da rede elétrica através de sua decomposição em elementos físicos (estáticos); a modelagem conceitual descrevia os componentes do problema de fluxo de potência – os quais eram dinâmicos, pois se originavam com base no estado operativo (*status*) dos elementos físicos. Entretanto o desempenho do programa não foi satisfatório frente àqueles desenvolvidos em FORTRAN, ficando até 3 vezes mais lento. O problema relatado era devido ao excesso de mensagens entre os objetos que compunham o sistema que consumia entre 30% e 50% do tempo computacional.

O trabalho de Folley et alii [20] propôs uma interface homem-máquina para SEE utilizando uma estrutura hierárquica de classes. Tal estrutura foi utilizada em Folley e Bose [21] em conjunto com a perspectiva física e conceitual proposta por [19]. Neste trabalho a organização da descrição física do SEE foi feita através de subestações, descrevendo mais fielmente a estrutura real do sistema. A rede operativa era inicialmente processada por subestação – a qual era responsável pela criação dos barramentos dentro de seu domínio físico – e posteriormente processada externamente com o uso das linhas de transmissão. A solução do problema de fluxo de potência foi feita através do método de Gauss-Seidel pela coerência deste à metodologia OO, segundo os autores do trabalho. Entretanto o problema de performance relatado por [19] também foi encontrado em [21].

O problema de desempenho das abordagens OO passou a ser questionado como sendo resultado da prática computacional utilizada nos modelos anteriores, segundo o que foi apresentado por Hakavik e Holen [22]. Eles propuseram um novo conjunto hierárquico de classes voltado para o estudo de fluxo de potência, bem como apresentaram um conjunto hierárquico de aplicativos baseados nesta estrutura. Também foram apresentadas classes para manipulação matricial/vetorial esparsa e respectiva solução de sistemas lineares. A comprovação da eficiência das classes matriciais e vetoriais foi feita utilizando módulos externos implementados em FORTRAN para solução de sistemas lineares. A comparação mostrou que a utilização de boas práticas computacionais para o tratamento matemático do problema possuía desempenho equivalente às implementações em FORTRAN, concluindo que o desempenho da abordagem OO depende das práticas computacionais utilizadas no tratamento matemático.

As experiências da aplicação da OO a sistemas lineares de grande porte não demoraram a aparecer. Em 1995, Zhou [23] apresentou um trabalho voltado ao estudo do fluxo de potência que possuía uma estrutura de classes semelhantes a [22]. Neste trabalho Zhou relata um bom desempenho computacional e mostra manipulações de matrizes e de sistemas lineares de grande porte.

Manzoni, Silva e Decker [24] elaboraram um novo conjunto hierárquico de classes para representação de SEE, bem como apresentaram uma nova estruturação para a classe matricial direcionada para a eficiência computacional e para manipulação de matrizes esparsas de grande porte. Este trabalho voltou-se simulação da dinâmica de SEP e apresentou um bom desempenho computacional quando comparado ao aplicativo de análise de transitórios eletromecânicos (ANATEM) desenvolvido em FORTRAN pelo CEPEL. Os autores atribuíram às boas práticas computacionais o bom desempenho obtido e mostraram que a OO pode ser uma alternativa viável mesmo para sistemas de grande porte e com requisitos de velocidade.

A questão da padronização dos modelos para que pudessem suportar as diferentes funções e estudos do SEE foi levantada por Zhu e Jossman [25] em seu trabalho sobre a utilização de *Padrões de Projeto* na modelagem de SEE. Entretanto, segundo os autores, existe a dificuldade de se alcançar uma estrutura *ótima* devido à diversidade de estudos que a base comum deveria suportar.

O modelo unificado começou a ser discutido com maior ênfase a partir de 2000, quando alguns trabalhos propuseram soluções para a integração de ferramentas e a generalização dos modelos de SEE. Agostini et alii [26] apresentou um conjunto hierárquico de classes genéricas para a modelagem física do sistema elétrico, classificando os elementos de composição do sistema conforme seu tipo de conexão. Como exemplo do trabalho foi implementado, com base na estrutura de classes proposta, um fluxo de potência linearizado e uma simulação da dinâmica do sistema. A evolução deste trabalho foi apresentada em [16] e [17], os quais reformularam parcialmente a estrutura hierárquica de classes e inseriram a classificação explícita de elementos estruturais e elementos de composição. Os elementos estruturais representam os dispositivos físicos do sistema e os elementos de composição agrupam diferentes equipamentos que formam um elemento físico. A adaptabilidade do modelo a um dado tipo de estudo foi possível com o uso do padrão de projeto *Adapter*, o qual proporciona a troca de funcionalidade em tempo de execução através do uso de classes adaptadoras (compatibilizam a interface comum da classe ou objeto a um tipo de aplicação e inserem as características funcionais necessárias). Os trabalhos apresentam uma estrutura hierarquizada de classes para diversas aplicações em SEE, classes matemáticas para manipulação de matrizes e solução de sistemas lineares e classes para gerenciamento de bancos de dados.

A tendência de integração dos diversos estudos e funções dentro dos centros de operação e controle continuou com o trabalho apresentado em Becker et alii [14]. Pautando na criação de um sistema de informações integrado, visando flexibilidade e reutilização de módulos prontos, o CIM (*Common Information Model*) – modelo de informação apresentado no trabalho –

fornece um conjunto de classes para a representação de SEE e impõem uma padronização de interfaces para a integração das diferentes ferramentas computacionais. A CIM utiliza o conceito de *metadados* para viabilizar a comunicação entre as aplicações existentes, considerando também as novas aplicações. O CMI é composto pelos seguintes modelos: *wires mode; SCADA model; load model; energy scheduling model; e generation model*.

Pandit et alii [15] apresenta uma estruturação de classes agrupadas nas seguintes categorias: *Apparatus*, representa a modelagem física do SEE; *Substation*, agrupa os elementos físicos e coordena o processamento topológico; e *Graph*, cria a conectividade entre os elementos físicos e subsidia diferentes aplicações. A rede elétrica é representada pela classe *Network*, que simplesmente agrega elementos físicos através de uma representação matricial (*Ybus*) após o processamento topológico. As diferentes aplicações derivam da classe *Network* e agregam suas características específicas. Um processador de topologia que agrega as diferentes categorias de classes também é proposto e posteriormente detalhado em [27]. O tratamento matemático é realizado por um conjunto de classes matriciais, vetoriais e de solução de sistema lineares voltados para aplicações distintas, as quais são detalhadas em [28] e [29].

O trabalho recente de Manzoni [18] refina os conceitos apresentados nos trabalhos anteriores e complementa a modelagem com a inserção de novas classes voltadas à representação do monitoramento, da proteção, da descrição espacial-topológica, do controle/operação e da propriedade organizacional. Uma estrutura hierárquica e relacional de classes proporciona eficientemente a descrição topológica do sistema e sua subdivisão física e lógica. A descrição física representa o sistema físico e sua organização espacial, descrevendo o arranjo dos equipamentos e dispositivos das subestações interligadas por linhas de transmissão e agrupas em área e subáreas. A descrição lógica reflete o estado operativo do sistema após a configuração de rede, resultando na ilhas operativas que são representadas pelas barras elétricas e o ramos que interconectam – desconsiderando os elementos de manobra. As áreas possuem centros de controle e operação e são formadas pelo conjunto de subestações, linhas de transmissão e subáreas. Os elementos e dispositivos que compõem o sistema possuem um modelo versátil que distingue sua disposição física da modelagem teórica – sendo possível a troca de funcionalidade (modelo) em tempo de execução para atender a aplicação. O trabalho renova o tratamento matemático através de um conjunto de classes que fornecem a capacidade de compor equações e fornecer derivadas automáticas – baseado em blocos matemáticos multifuncionais organizados na forma de diagrama de blocos. Os modelos (conjunto de equações) podem ser facilmente gerados externamente para representar novos comportamentos e dispositivos (dispositivos definidos pelo usuário), bem como facilmente

trocados em tempo de execução. Também são apresentadas classes otimizadas para a manipulação de matrizes e solução de sistemas lineares. Devido à versatilidade, à completude, à flexibilidade e à generalização da modelagem proposta por este trabalho, considera-se esta o padrão a ser seguido e evoluído para atendimento das particularidades das diversas funções.

A orientação a objetos também foi aplicada na modelagem de outras ferramentas e estudos dentro do setor elétricos, tais como as interfaces gráficas [20, 21, 30], a modelagem e gerenciamento de base de dados [14, 31-34], a estimação de estados [29], o planejamento da transmissão [35], o sistema de distribuição [25, 36, 37], a simulação de faltas [38] e o diagnóstico de falhas e a restauração de sistemas [39, 40].

4.3.2 O contexto de desenvolvimento do projeto

O projeto do estimador de estado robusto teve início com o interesse da Companhia Estadual de Energia Elétrica do Rio Grande do Sul (CEEE) em reformular parcialmente as ferramentas computacionais de seu centro de operação de sistema (COS) e testar novas metodologias em áreas que apresentavam alguns problemas técnicos. A CEEE, atendendo a lei n˚ 9.991/2000, identificou e definiu os temas prioritários para a aplicação de recursos destinados a projetos de P&D. Dentre os temas identificados incluía-se a estimação de estados aplicada ao sistema de transmissão de energia elétrica.

A proposta formulada pela companhia previa a implementação do projeto no prazo de 1 ano, visando exclusivamente o desenvolvimento da função de estimação, e se baseando nos procedimentos e no ambiente computacional do COS. O estimador deveria adaptar-se, dentro das possibilidades, a estrutura operativa sem gerar grandes reformulações.

O COS da CEEE possuía, dentre outras ferramentas: um estimador de estados desenvolvido pelo CEPEL; um configurador de redes desenvolvido internamente; um software (SARON) para gerenciar a execução do estimador e gerar o arquivo de entrada para o estudo de fluxo de potência (ANAREDE); uma base de dados do histórico operativo criada pela companhia; e um sistema de aquisição de dados baseado em rotinas desenvolvidas internamente, as quais subsidiavam o estimador em tempo real. A integração das ferramentas, quando existente, era feita por softwares específicos que convertiam arquivos de saída em arquivos de entrada de outras funções.

O arquivo de entrada do estimador de estados descrevia o sistema através de conexões barra-ramo utilizando a técnica de cartões de comando de estrutura semelhante ao arquivo utilizado pelo ANAREDE. A descrição dos medidores era feita em conjunto com os elementos de

rede através de um identificador e de sua respectiva ponderação. A mesma técnica era utilizada para informar os dispositivos de manobra que cada ramo e elemento shunt possuía, possibilitando uma configuração primária de redes. O descritor (arquivo de entrada) possuía cartões para informar medidores fictícios criados por composição e para informar medidores especiais alocados para grupos e sistemas. O formato do descritor é detalhado no apêndice B, junto com o manual do estimador robusto desenvolvido.

O estimador de estados da CEEE operava em duas frentes distintas: estimação em tempo real, onde as informações sobre as medições e sobre o status dos elementos de manobra eram obtidas através de rotinas de leitura baseadas no protocolo IEC 60870-5-101; e a estimação de estados em modo estudo, onde a leitura era feita diretamente do histórico operativo (banco de dados). Ambos possuíam o mesmo núcleo funcional, o qual se baseava na metodologia de mínimos quadrados ponderados (*Weighted Least Squares - WLS*) aplicada ao modelo CA desacoplado, utilizando a decomposição LDU como método de fatoração.

A CEEE propôs paralelamente um projeto de configurador de redes que, segundo os engenheiros da companhia, seria integrado posteriormente às demais ferramentas do COS, inclusive o estimador de estados robusto. Portanto os projetos foram desenvolvidos no mesmo ciclo de P&D, deixando a integração das ferramentas para ser realizada num projeto futuro.

As limitações impostas ao desenvolvimento do EE robusto surgiram tanto por parte da companhia, que necessitava de um software funcional com aplicabilidade imediata, quanto por parte dos requisitos de projeto, que limitavam o tempo de desenvolvimento e direcionavam a modelagem do problema. A solução encontrada foi restringir a abrangência do modelo de SEE, focando-o principalmente nos estudos de regime permanente e nas das características existentes na companhia, sem limitar a possibilidade de evolução da ferramenta em seu campo de aplicação. Como resultado de tais fatores, o modelo de SEE proposto restringiu-se a integração do modelo barra-ramo a dispositivos de manobra e de supervisão que possibilitassem uma configuração primária de redes e uma supervisão operativa baseada nos componentes do sistema.

4.3.3 O modelo de SEE proposto

O modelo de SEE proposto por este trabalho difere significativamente de grande parte dos modelos atuais por não contemplar a descrição físico-topológica do sistema. Apesar de ser uma característica necessária, tal requisito invade o campo de responsabilidade do configurador de redes e adiciona complexidade e tempo na execução do projeto. Entretanto, cabe ressaltar,

que as novas vertentes na EESP prezam pela integração do configurador de redes e do estimador de estados para possibilitar a estimação topológica de maneira eficiente [93].

As restrições funcionais e orçamentárias levaram a objetivação e o direcionamento do modelo computacional, o qual se voltou a representar o sistema através do modelo barra-ramo. Entretanto, visando aplicabilidade, tal modelo deveria ter a capacidade de representar os dispositivos de manobra dos ramos e dos elementos shunt, possibilitando uma configuração primária de redes. Ademais, o modelo de SEE, por óbvio, deveria ser capaz de representar todo o plano de medição e suas demais características específicas relacionadas, como a representação de medidas compostas.

A apresentação do modelo de SEE numa única figura compromete seu entendimento devido a sua quantidade de classes e a dificuldade de dispor estas de tal forma que os relacionamentos não sejam dúbios quando vistos num primeiro momento. Visando a boa organização, o modelo será apresentado inicialmente por núcleos e posteriormente completo.

4.3.3.1 Núcleo abstrato

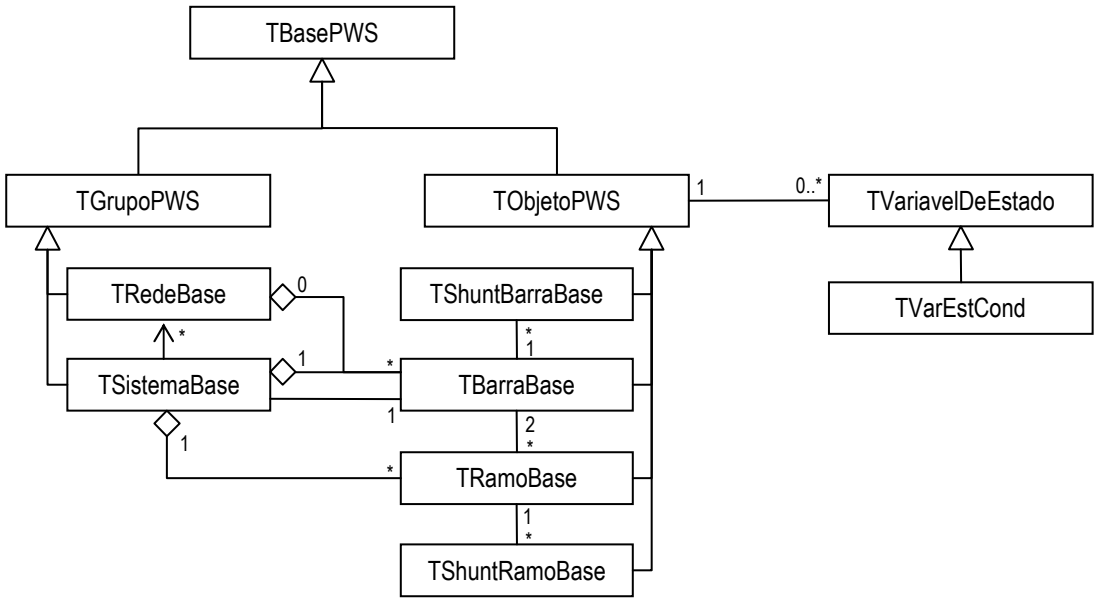


Figura 39 – Modelo de SEE, Parte 1: Classes do núcleo abstrato.

A primeira parte, ilustrada na Figura 39 , mostra uma estrutura hierárquica de classes responsável pela representação básica do SEP. Todas as classes que representam grupos ou elementos do SEP possuem como origem a classe *TBasePWS*. Tal característica foi introduzida para permitir a manipulação de objetos deste domínio independentemente de seu tipo elementar ou grupal, possibilitando a identificação da instância através de funções virtuais classificadoras que retornam o tipo predefinido de cada nível de abstração. O segundo nível de abstração divide

as entidades em grupos ou elementos através das classes *TGrupoPWS* e *TObjetoPWS* respectivamente. As classes derivadas de *TGrupoPWS* agrupam objetos descendentes de *TObjetoPWS* independentemente da finalidade deste conjunto.

As classes derivadas de *TObjetoPWS* representam componentes integrantes do modelo barra-ramo de SEP e impõem aos seus descendentes a posse de uma lista interna de variáveis de estados (representadas por *TVariavelDeEstado*) que pode ser ou não nula. Tal imposição induz ao entendimento que qualquer componente pode possuir variáveis de estados dependendo do estudo ou função a ser executada e que tais variáveis podem integrar o problema. Portanto preza-se pela liberdade na realização do estudo e pela facilidade de atualização da lista de variáveis de estado modeladas.

O modelo barra-ramo foi representado através da utilização de quatro classes bases (terceiro nível de abstração) que posteriormente serão especializadas para atender as particularidades dos elementos que compõem o SEP. O SEP, concebido através do modelo barra-ramo, pode ser idealizado fisicamente como um conjunto de nós interligados por ramos, semelhante a um grafo. Entretanto o SEP possui a particularidade da existência do nó comum (terra) e dos elementos shunts que conectam este aos demais nós.

O terceiro nível de abstração, que representa abstratamente o modelo barra-ramo não manobrável, possui as seguintes classes:

- **TBarraBase:** responsável por representar a barra, possuindo uma lista de objetos *TRamoBase* (Ramos) e uma lista *TShuntBarraBase* (Shunts). Além de propriedades peculiares para identificação (número ID) e estado;

- **TRamoBase:** responsável por representar qualquer elemento que interliga duas barras, possuindo variáveis para apontar as barras terminais e uma lista de objetos *TShuntRamoBase* (Shunts). Além de propriedades para identificação, estado e circuito;

- **TShuntBarraBase:** responsável por representar todo elemento que conecta uma barra à terra, possuindo uma variável para apontar à barra e propriedades de identificação, estado, tipo de controle e variáveis de controle;

- **TShuntRamoBase:** responsável por representar todo elemento que conecta o extremo de um ramo à terra, possuindo uma variável para apontar ao ramo e propriedades para identificação, sentido de conexão e estado. Tal classe foi introduzida com o intuito de

abordar de forma mais eficiente os elementos shunts que são fisicamente dependentes do estado operativo do ramo, ou seja, estão fisicamente dispostos após o disjuntor que conecta o ramo ao barramento.

Os modelos que descrevem fisicamente o SEP – comentados anteriormente – não necessitam da inclusão da classe *TShuntRamoBase*, uma vez que o motivo fundamental para a sua existência torna-se superado com a montagem físico-topológica do sistema.

A estrutura de classes adotada para representar de forma abstrata o modelo barra-ramo evidencia a flexibilidade dada pela OO ao representar tal modelo de forma similar a sua concepção lógica. Torna-se claro a possibilidade de navegação entre os diversos objetos que compõem o modelo com base no conhecimento de um único objeto, independente do tipo da classe. Tal característica é obtida através do uso de apontadores (ou ponteiros), que possibilitam a referência ao objeto informado. Assim, para conhecer todos os objetos que formam o sistema modelado, basta ter informação de um único objeto – como, por exemplo, uma barra (barra de referência).

Não obstante as características supracitadas, o modelo básico proposto contempla a conceito de grupo em sua formação. Ou seja, apesar de existir a independência conceitual, as classes elementares derivadas de *TObjetoPWS* possuem o conhecimento de sua participação em um grupo. Assim o grupo é parte integrante do núcleo do modelo e fornece as informações que lhe são próprias ao conjunto, ou seja, informações globais pertinentes ao grupo de elementos e não ao elemento isolado. Tal característica viola, em tese, as regras da OO purista, uma vez que exorbita o conceito da classe “elemento” adicionando informações de uma classe “grupo”. Entretanto a finalidade do elemento é ser parte integrante do grupo e, sobretudo, interagir com este de forma ativa. Portanto não há violação de regras, simplesmente há uma abordagem conceitual conjunta que se baseia na relação elemento-grupo.

A classe *TSistemaBase*, derivada de *TGrupoPWS*, representa o SEP na visão do modelo barra-ramo e é responsável por gerir as propriedades, ações e eventos que envolvem o conjunto de elementos que constituem o sistema. Para tanto, esta classe possui uma lista de apontadores para os objetos *TBarraBase* e *TRamoBase* interconectados entre si, um apontador para a barra de referência e um conjunto de propriedades próprias, como um número inteiro de identificação e um valor real para representação da potência base. Cada objeto da classe *TBarraBase* e *TRamoBase* possui um apontador para um objeto *TSistemaBase* e uma função de reconhecimento de conectividade. Após informar a barra de referência ao objeto *TSistemaBase*, este chama a função de reconhecimento daquele objeto e informa, através do

parâmetro da função, o seu próprio ponteiro. Inicia-se, então, uma reação em cadeia, na qual a barra passa a executar o mesmo procedimento com aqueles objetos conectados a ela e que ainda não possuem o mesmo ponteiro de sistema como propriedade interna.

As classes que representam sistemas, ilhas ou redes nos modelos de SEP atuais não possuem uma lista de ramos (ou conexões) como a existente na classe *TSistemaBase*. Todas as informações necessárias para a montagem físico-topológica podem ser obtidas diretamente do conhecimento do conjunto de pontos elétricos que formam o sistema físico. A configuração físico-topológica é atribuída à classe que representa a subestação, existindo uma distinção latente entre a barra e o nó ou ponto elétrico. As barras, naqueles modelos, resultam do processamento topológico baseado nos estados dos ramos lógicos existentes na subestação, ou seja, são resultantes da junção dos pontos elétricos interligados por ramos com continuidade lógica (chave fechada). Portanto a barra passa a existir após o processamento topológico, sendo a representação da união de um determinado conjunto de pontos elétricos.

No modelo apresentado neste trabalho, a gerência de toda a ação sobre o sistema é coordenada pela classe *TSistemaBase* – incluindo sua configuração lógica ou operativa. Tal configuração é equivalente à montagem físico-topológica supracitada, porém, neste caso, as barras são pré-existentes e imutáveis (devido à origem do modelo). Entretanto o modelo deve ser capaz de lidar com situações onde somente um dos extremos do ramo se encontra aberto, o que resulta na existência temporária de uma barra fictícia relacionada àquele ramo. Visando solucionar esta particularidade e, ao mesmo tempo, estender a flexibilidade do modelo sem afetar significativamente o desempenho e os recursos computacionais disponíveis, optou-se pela inclusão de uma lista de ramos gerenciada pela classe *TSistemaBase*. Tal relacionamento possibilita a àquela classe monitorar e gerenciar a criação ou destruição de barras fictícias temporárias durante a configuração lógica ou operativa do sistema e estender a flexibilidade do modelo ao disponibilizar uma listagem completa dos ramos existentes.

A configuração lógica ou operativa é o processo pelo qual são formadas as ilhas operativas, as quais são representadas pela classe *TRedeBase*. O procedimento de criação de uma ilha operativa é similar àquele utilizado para formação do sistema, mas há diferenças no tratamento da propagação entre os elementos pertencentes a um mesmo conjunto. As classes elementares não possuem uma ligação direta com a classe *TRedeBase*, mas existe uma propriedade interna que indica o estado operativo de um determinado objeto.

As classes *TBarraBase* e *TRamoBase* possuem uma função de reconhecimento de conectividade lógica que, de forma similar ao procedimento de reconhecimento do sistema, verifica a continuidade lógica do sistema com base nas condições impostas por funções virtuais.

Portanto as classes descendentes de *TBarraBase* e *TRamoBase* podem impor mudanças na configuração lógica do sistema baseando-se tão somente na redefinição destas funções e de suas respectivas condições de continuidade lógica.

Apesar da semelhança entre as classes *TSistemaBase* e *TRedeBase* existem outras informações importantes a saber:

- A classe *TRedeBase* representa, por meio de uma lista de objetos *TBarraBase*, o conjunto de barras que formam uma ilha operativa, a qual é delimitada através das condições implementadas nas classes descendentes de *TBarraBase* e *TRamoBase.* Diferentemente da classe *TSistemaBase*, a classe *TRedeBase* não armazena informações sobre os ramos da ilha operativa;

- Classe *TSistemaBase* pode se encontrar ou no estado operativo, quando disponibiliza uma lista de ilhas operativas, ou no estado não-operativo, quando não houve tal configuração lógica ou ocorreu alguma alteração estrutural no grupo de objetos que constituem o sistema (como uma inclusão, alteração ou exclusão de barra, ramo, ou qualquer elemento shunt);

Cada classe descendente de *TObjetoBase* possui a possibilidade de gerir um conjunto de objetos *TVariavelDeEstado* que representam os estados de um determinado elemento a ser modelado. Normalmente, nos modelos atuais, a gerência e a utilização de variáveis de estados dependem do estudo a ser realizado com o respectivo modelo. Entretanto, devido à vinculação precoce deste modelo ao estudo de sistemas elétricos em regime permanente, a relação entre a classe e seu respectivo conjunto de variáveis de estado foi feita antecipadamente. Isto sugere que qualquer modificação necessária à adaptação deste modelo a estudos que envolvam variáveis de estados distintas, ou a utilização de uma configuração dinâmica de variáveis de estados, acarretará necessidade de inserção de um dispositivo de gerência de estados na classe *TObjetoBase*. Tal dispositivo deverá ser capaz de definir os tipos de variáveis utilizadas em cada estudo e viabilizar a translação entre modelos distintos. Atualmente, os modelos propostos por [16], [17] e [18] possuem a referida adaptabilidade.

A classe *TVariavelDeEstado* possui propriedades para identificar o tipo de estado, armazenar seu respectivo valor, indicar a mutabilidade deste valor, identificar a variável e estabelecer uma relação com seu objeto proprietário, além de possuir procedimentos e funções para alterar ou incrementar o valor do estado. O tipo e o objeto proprietário são inalteráveis e definidos durante a criação do objeto, ou seja, a criação, a gerência e a destruição destes objetos pertencem às classes elementares derivadas de *TObjetoBase*.

Não existe limitação ou validação de valores impostos diretamente aos objetos da classe *TVariavelDeEstado*. Entretanto alguns estados, como a valor da relação de transformação dos

transformadores e LTCs, possuem limites físicos que devem ser modelados adequadamente. Estas variáveis de estados que representam parâmetros da SEE foram contempladas através da classe *TVarEstCond* descendente da classe *TVariavelDeEstado*. Tal classe possui definição de limites para o valor do estado e a definição de valores incrementais máximos. Com esta definição, as variáveis de estados com limitação física tangível podem ser adequadamente trabalhadas.

4.3.3.2 Núcleo de manobra e supervisão

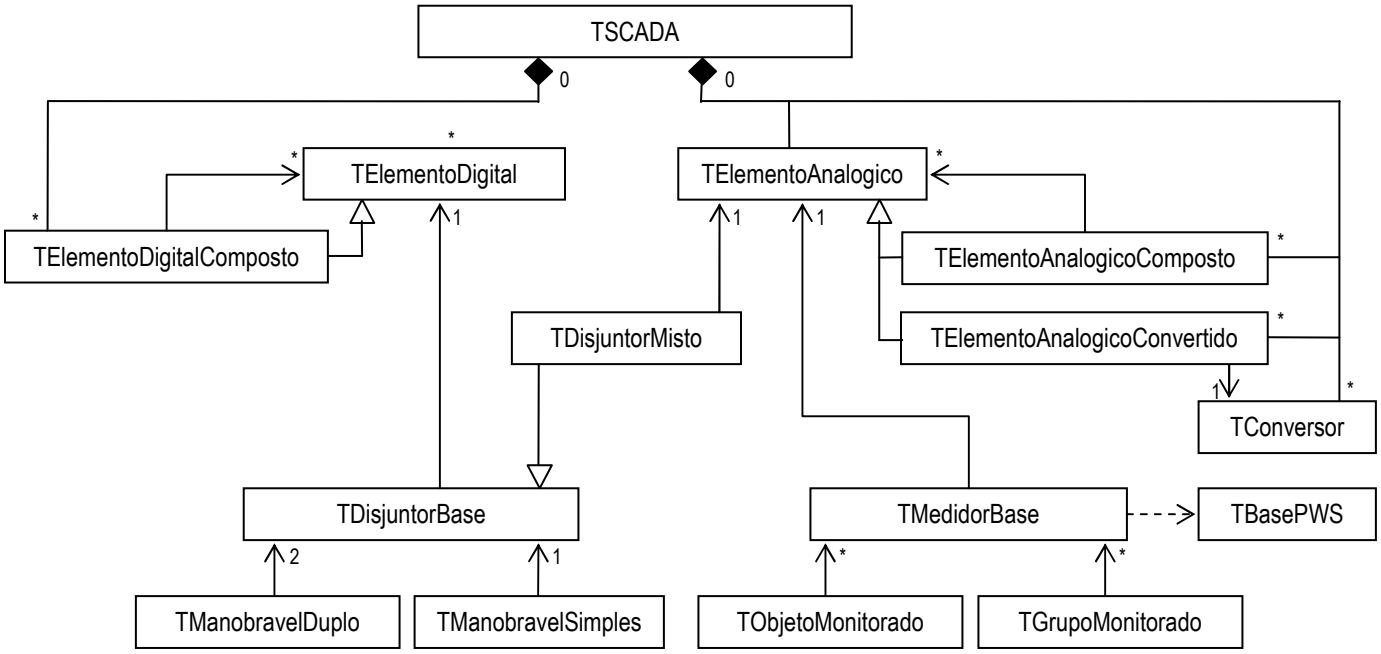


Figura 40 – Modelo de SEE, Parte 2: Classes de manobra e supervisão.

Modelos computacionais são desenvolvidos com o objetivo de simular, supervisionar e controlar os sistemas físicos reais. A simulação visa obter informações comportamentais do sistema frente a uma determinada situação, baseando no seu modelo físico e no conjunto de condições iniciais e eventos físico-temporais conhecidos ou previamente definidos dentro de um conjunto de possibilidades. Portanto toda simulação visa obter informações ou comportamentos do sistema – próprios de situações passadas[1] ou futuras – através do emprego de um determinado modelo. Por sua vez, a supervisão e o controle necessitam representar o sistema monitorado para subsidiar as decisões operativas, ou seja, o modelo computacional adotado deve processar as informações de campo e refletir o estado operativo real, fornecendo a possibilidade de atuação sobre sistema através do uso dos elementos físicos apropriados.

[1] Lembra-se que mesmo a simulação pós-operativa necessita de informações operacionais passadas.

A parte do modelo apresentado na subseção anterior representa a relação entre os elementos e os grupos de qualquer sistema que se comporte de forma semelhante a um grafo. Entretanto tal modelo não se vincula externamente a qualquer tipo de informação sobre o estado operativo do sistema modelado. Para solucionar tal deficiência, foi projetado um conjunto de classes – mostrado na Figura 40 – que fornece adaptabilidade externa ao modelo. Estas classes inserem a possibilidade de manobra e supervisão de grandezas no contexto do modelo de SEE. Todos os elementos do núcleo abstrato que possuem a função de interligar barras devem ter a possibilidade de configurar seu estado operativo (aberto ou fechado) com base em variáveis binárias e/ou analógicas. Por outro lado, todos os elementos daquele núcleo podem ter grandezas físicas susceptíveis de mensuração. Logo o modelo deve ser capaz de trabalhar tanto com variáveis digitais (ou binárias) quanto com variáveis analógicas (ou reais) e fornecer uma estrutura de manobra e supervisão que possa ser agregada ao núcleo abstrato.

As classes do núcleo de manobra e supervisão foram projetadas para serem independentes das classes dos demais núcleos, ou seja, tais classes têm o objetivo de disponibilizar as informações sistêmicas às demais classes do modelo de SEE, sem qualquer relação à sua efetiva utilização.

A classe *TSCADA* tem a função de criar, gerenciar e destruir elementos analógicos e digitais e conversores de grandezas, além de realizar a seleção e repasse de tais elementos para o objeto encarregado pela leitura destas grandezas. A classe TLeitor, que será estuda mais detalhadamente nas seções subseqüentes, é a responsável por realizar a leitura de valores baseando-se nas tecnologias implementadas em cada caso, ou seja, no conjunto de tecnologias de acesso as informações de telemetria que co-existem nos centros de controle. Cada descendente desta classe pode implementar seu próprio método de leitura baseado, por exemplo, em diferentes tecnologias e protocolos de rede, em bases de dados ou em sistemas híbridos que utilizam diversos protocolos. Portanto preza-se pelo reconhecimento da diversidade de tecnologias de comunicação que podem co-existir dentro dos centros de operação de sistemas e pela independência funcional da estrutura de classes de supervisão e manobra.

Independentemente do local ou do equipamento que se origina uma determinada leitura, esta é identificada pelo sistema de aquisição como sendo simplesmente um *elemento de informação*. Apesar da importância indiscutível de todos componentes e tecnologias que integram uma telemetria, para o sistema de supervisão, cada elemento ou registro interno resume-se apenas numa leitura temporal identificada. O sistema de aquisição de dados, por este ponto de vista, resume-se em um conjunto de elementos de informação que estrutura o monitoramento. Os aspectos físicos pertinentes aos equipamentos de medição, transformação

ou transdução não são modelados por tais elementos, apesar de muitas de suas características terem origem naquele conjunto. Ideologicamente, ao se modelar um sistema de controle e aquisição de dados deve-se incluir todo sistema de transformação ou transdução de sinais, suas características técnicas, sua relação com os medidores, os medidores e o sistema de comunicação. Assim, para cada medidor com comunicação efetiva, se criaria um elemento de informação no centro de aquisição. Infelizmente, além de muitas empresas não possuírem todas as informações físicas adequadas, este modelo somente traria vantagem quando efetivamente utilizado sobre o modelo físico-topológico.

As classes *TElementoDigital* e *TElementoAnalogico* representam respectivamente os elementos digitais e analógicos do sistema de aquisição de dados e são semelhantes quanto à implementação e ao modelo conceitual. Ambas as classes possuem um identificador extenso e conversível, *flags* que indicam o tipo de entrada de informação (registro [entrada manual] ou leitura [entrada automática]), a validade da informação e sua a disponibilidade no sistema de aquisição empregado e, por fim, uma variável binária ou real para armazenar as informações. A classe *TElementoAnalogico* possui ainda uma variável que representa a precisão do elemento e um *flag* que indica se o elemento representa uma restrição física, ou seja, livre de erros. A leitura, manual ou automática, é feita por uma função específica que indica o valor da variável e seu respectivo o estado, com ou sem falha. Por vezes a velocidade de varredura do sistema de aquisição é superior ao tempo de atualização ou de envio de uma determinada telemetria, ocorrendo, neste caso, a invalidação parcial da medida ou do estado digital. Por este motivo existe a necessidade de conhecimento do estado da leitura, pois algumas vezes é possível utilizar medidas ou estados considerados *falhados* para salvar a perda de informação naquele determinado ciclo de medição (ou varredura). Neste caso, o estudo deve tratar adequadamente o *valor* sistêmico desta informação.

Existem outras particularidades que não são próprias do sistema de aquisição de dados, mas que este deve possibilitar alguma solução aparente. O uso de disjuntores com circuitos de disparo intertravados e a possibilidade de redução do sistema elétrico em operação são alguns exemplos onde uma aplicação de elementos de informação compostos por outros elementos de informação simplificam e acrescem informação global. No caso da redução do sistema elétrico em operação, os medidores de fluxo em linhas adjacentes que tiveram uma de suas barras removidas podem formar conjuntamente uma injeção de potência ativa ou reativa em barras de passagem. Outro exemplo é o conhecimento da lógica de intertravamento de disjuntores, a qual pode estender a abragência do monitoramento destes equipamentos para outros equipamentos que não possuem supervisão direta. Entretanto deve existir um dispositivo computacional que

possibilite tal aplicação. Para tanto foram desenvolvidas as classes ditas *compostas*, ou seja, as classes *TElementoDigitalComposto* e *TElementoAnalogicoComposto*. Estas classes descendem respectivamente de *TElementoDigital* e *TElementoAnalogico* e possuem a capacidade de compor informação através de um conjunto de objetos de sua respectiva classe antecessora. Assim estas classes possuem a mesma estrutura de suas antecessoras, com exceção do dispositivo de leitura, e podem manipular a informação contida nos elementos de sua lista para gerar uma nova informação. Os elementos da lista podem ser objetos da classe antecessora ou de suas descendentes, o que inclui a possibilidade de manipular informação de diferentes tipos de classe que derivam de *TElementoAnalogicoComposto* ou *TElementoDigitalComposto*.

A conversão de valores é outra prática comum dentro de centros de operação de sistemas. Os motivos para tal prática abrangem desde a economia na transmissão de dados até a tecnologia do sistema de telemetria. A conversão deve ser feita através de funções que trabalham diretamente com a coleta dos dados. Portanto deve haver um dispositivo que converta tais medidas para serem utilizadas pelas diversas funções do COS. Para contemplar tal característica, foi desenvolvida uma classe abstrata de conversão chamada de *TConversor*. Tal classe possui uma função abstrata de conversão que recebe o valor formado e retorna um valor processado pelas condições da função implementada nas classes descendentes. Suas classes descendentes utilizam desde tabelas de conversão D/A até funções matemáticas próprias. Suas propriedades internas dependem somente da existência ou não de parâmetros de conversão. A classe *TElementoAnalogicoConvertido* possui um objeto da classe *TConversor* para efetuar automaticamente a conversão de valores antes de seu armazenamento interno.

Todas as classes comentadas anteriormente somente formam a base para a inserção de informação externa ao modelo de SEE, entretanto não foram modeladas para ter aplicabilidade imediata sobre as classes que compõem o núcleo abstrato. Para ser aplicável, esta estrutura deve ser utilizada por outras classes que estão diretamente relacionadas ao núcleo deste modelo e que são conceitos reais do sistema. As classes *TDisjuntorBase* e *TMedidorBase* representam tais conceitos ao modelarem o disjuntor e o medidor de forma flexível. A classe *TDisjuntorBase* manipula e utiliza a informação de um objeto *TElementoDigital* para indicar seu estado operativo (aberto ou fechado), além de conter propriedades que indicam seu estado operacional normal e seu vínculo a um componente do sistema. Por sua vez a classe *TMedidorBase* utiliza a informação de um objeto *TElementoAnalogico* para indicar a magnitude da grandeza mensurada, além de possuir propriedades que indicam o tipo de grandeza, seu sentido (grandeza invertida ou não) e o componente do sistema que esta sendo monitorado. O relação com o componente do sistema monitorado é feita através da classe TBasePWS, pois

existe a possibilidade de grupo de componentes possuir medidores, como um medidor de temperatura numa subestação, e não só componentes do sistema.

Uma particularidade dos centros de operação de sistemas foi introduzida ao modelo de manobra e supervisão. Trata-se da classe TDisjuntorMisto, derivada da classe TDisjuntorBase, que agregada um elemento de informação analógico (objeto da classe TElementoAnalogico) para subsidiar e compor o estado operativo da classe. Tal procedimento – estranho e sem finalidade quando analisado num primeiro momento – permite cobrir falhas de leitura do sistema de aquisição de dados gerados por fatores diversos. Muitas vezes o COS não recebe, por horas, a informação de um elemento digital por falha na comunicação ou avaria do equipamento de telemetria. Aos responsáveis pela supervisão restam analisar as informações disponíveis para configurar manualmente o estado do elemento. Uma das informações que viabilizam a tomada de decisão é o conhecimento de uma grandeza analógica atrelada diretamente ao local de origem do problema, como o fluxo de potência ativa ou reativa no ramo. Devido a esta prática, alguns softwares utilizados nos COS permitem que o estado do componente de manobra seja definido por uma lógica que utilize elementos de informação digitais e analógicos. Da mesma forma, a classe *TDisjuntorMisto* permite escolher uma lógica de composição com os elementos disponíveis para definir o estado operativo do componente.

Os modelos de SEE que trabalham com a montagem físico-topológica, como os apresentados em [16], [17] e [18], trabalham de forma diferente com os conceitos de disjuntor e medidor, uma vez que os disjuntores são especializações de ramos elétricos e os medidores atrelam-se a um ponto elétrico específico (e não a um componente). Entretanto a inserção de monitoramento amplo e a possibilidade de manobra num modelo barra-ramo necessitam de uma abordagem distinta da realizada por modelos que contemplam configuradores de redes. Tal abordagem de ser voltada aos componentes do sistema e a forma que essa informação deve ser trabalhada para não gerar restrições ou dependências. Portanto são necessárias classes específicas para acoplar as funcionalidades desejadas ao modelo de SEE e aos seus componentes. Isto será feito explorando as funções abstratas dos componentes do modelo de SEE.

Os elementos ou componentes de interligação (*TRamoBase*, *TShuntBarraBase* ou *TShuntRamoBase*) possuem uma ou duas funções abstratas que atuam na configuração lógica do sistema. Suas classes descendentes podem alterar ou sobrescrever tais funções com a lógica que importarem necessárias a este fim. A utilização de herança múltipla pode adicionar tais informações sem necessariamente alterar a essência da classe inicial, bastando criar uma classe descendente que una o elemento de interligação a uma classe responsável pela

manobra. A classe de manobra deve conter somente as novas definições das funções abstratas e as propriedades que estas irão utilizar para condicionar a continuidade lógica do processo de montagem operativa. As classes *TManobravelSimples* e *TManobravelDuplo* foram desenvolvidas com base nessa metodologia. A classe *TManobravelSimples* possui um objeto da classe *TDisjuntorBase* e a redefinição da função abstrata das classes *TShuntBarraBase* e *TShuntRamoBase*. Durante a configuração lógica, a função condicionará a continuidade lógica ao estado operativo do respectivo objeto (disjuntor). A classe *TManobravelDuplo*, por sua vez, possui dois objetos da classe *TDisjuntorBase* – um para cada extremo – e a redefinição das funções abstratas da classe *TRamoBase*. A forma de atuação durante a configuração lógica do sistema é semelhante à realizada pela classe *TManobravelSimples*. Portanto tais classes são os elos entre o núcleo abstrato, o sistema de manobra e a evolução do modelo que será vista a frente.

A mensuração de grandezas, diferentemente do sistema de manobra, acresce informações externas aos componentes do núcleo abstrato sem alterar qualquer característica implementada anteriormente. O modelo de medição deve ser capaz de agregar esta informação sem limitar a quantidade de medidores que cada componente pode possuir ou sem condicionar o vínculo ao tipo de grandeza mensurada ou a qualquer característica própria do sistema de medição. Tal limitação prejudicaria a abordagem de novas tecnologias e dificultaria a inclusão de novas grandezas ao conjunto de possíveis grandezas mensuradas, sem considerar o fato que a forma abstrata das classes componentes impossibilita o condicionamento a uma dada característica. Entretanto deve-se possibilitar que as classes que descendam das classes componentes possam realizar o referido condicionamento, pois pode ser necessário evitar vinculação imprópria ou indevida. Baseando-se nestas diretrizes, a classe *TObjetoMonitorado* foi desenvolvida para tornar possível o uso do modelo de medição pelas classes que descenderem das classes componentes. Para tanto, a classe *TObjetoMonitorado* possui uma lista de objetos *TMedidorBase*, um conjunto de funções e procedimentos para gerenciar e acessar as informações dos medidores e uma função abstrata que condiciona a entrada de um dado medidor na lista. As classes que descenderem da herança múltipla de uma classe componente e a classe *TObjetoMonitorado* possuirão a capacidade de trabalhar com medidores, criando-se vínculos automáticos entre os medidores e o componente após a inclusão deste último a lista de medidores.

Os grupos de componentes, derivados da classe TGrupoPWS, também possuem a possibilidade de serem monitorados. O relacionamento é realizado de forma análoga ao que fora modelado para os componentes, entretanto toda a relação é realizada para vincular o grupo

aos medidores. A classe TGrupoMonitorado é semelhante a classe TObjetoMonitorado, ressalvando sua especialidade. Todas as demais implementações são idênticas, tendo a mesma metodologia e finalidade.

4.3.3.2 Núcleo de acoplamento e especialização

O último núcleo do modelo de SEE proposto tem a finalidade de unir as funcionalidades introduzidas pelas classes do núcleo de manobra e supervisão aos elementos e grupos que formam o núcleo abstrato e de definir os elementos que compõem o modelo barra-ramo que será utilizado nos estudos em regime permanente. O acoplamento das classes de manobra e supervisão às classes abstratas resultará em classes com a capacidade de armazenar informação externa e manipular essa informação no processo de configuração lógica ou operativa do sistema. Tal acoplamento é realizado através de herança múltipla, onde a classe descendente unifica por herança as funcionalidades e as propriedades das classes antecessoras. Para facilitar o entendimento e a exposição do diagrama de classes, o núcleo de acoplamento e especialização será divido e apresentado em partes.

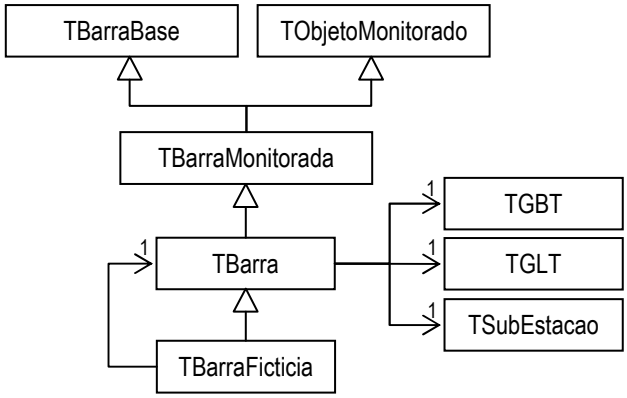


Figura 41 – Acoplamento e especialização da classe *TBarraBase*.

Descendendo das classes *TBarraBase* e *TObjetoMonitorado* por herança múltipla, a classe *TBarraMonitorada* agrega ao elemento abstrato do modelo a capacidade de manipular e vincular informações externas que são oriundas do sistema de medição. Une-se, portanto, funcionalidades distintas, mas não se altera qualquer característica de interligação, configuração física, configuração lógica ou qualquer propriedade advinda da classe *TBarraBase*. Entretanto, por motivos de adaptação à estrutura existente, há a redefinição de algumas funções virtuais das classes antecessoras tanto para limitar os tipos de medidores que podem ser utilizados por esta classe quanto para atualizar funções e procedimentos da classe *TBarraBase* relacionados ao monitoramento de grandezas.

A classe *TBarra* concretiza o conceito de barra dentro do modelo proposto. Tal classe não possui grandes mudanças ou inserções, restringindo-se a inclusão de um identificador alfanumérico (nome), de um ponteiro para o objeto responsável pela definição de sua tensão base (TGBT), de um ponteiro para o objeto responsável pela definição dos limites de tensão (TGLT) e de um ponteiro para o objeto que representa a subestação de situação física da barra. Tais propriedades foram incluídas tanto para preservar as informações presentes nos arquivos de descrição de rede, possibilitando uma identificação mais apropriada das barras e uma apresentação mais completa das informações geradas, quanto para viabilizar a conversão de unidades. A classe TGBT modela o grupo base de tensão, possuindo uma propriedade que indica o valor da tensão base e outra propriedade que associa uma cor característica a esta tensão. Por sua vez, a classe TGLT modela o grupo de limite de tensão, o qual indica os limites operacionais e emergenciais de tensão por unidade (p.u.) que a barra está sujeita, sendo dois limites normais e dois limites emergenciais utilizados na supervisão operacional do sistema. A classe *TSubEstação*, ao contrário dos modelos atuais, restringe-se a identificar um determinado local e associá-lo a uma determinada área, tendo – exclusivamente – o intuito de preservar a informação geográfica e disponibilizá-la para outras funções que possam fazer uso dela.

As funções que retornam as grandezas operacionais da barra – como a injeção líquida de potência ativa ou reativa, a tensão complexa ou polar, as derivadas em relação a qualquer estado, dentre outras – foram implementadas na classe *TBarraBase* por não necessitar de informações que são próprias dos demais componentes do sistema e de seus respectivos refinamentos ou especializações. Portanto, para obter a injeção reativa líquida na barra, basta somar as os fluxos de potência reativa de todos os ramos e as injeções reativas de todos shunts, eliminando a necessidade de qualquer conhecimento adicional sobre a natureza do componente. Todas as demais grandezas operacionais e suas respectivas derivadas são obtidas da mesma maneira. Entretanto essa técnica vincula precocemente a funcionalidade da classe ao tipo da variável estado utilizada, diferentemente dos modelos atuais que utilizam um padrão de conversão – conhecido como padrão de projeto *Adapter* [17] – para vincular dinamicamente a funcionalidade da classe aos tipos estados empregados a um determinado estudo. Tal prática foi utilizada em [16], [17] e [18].

A inserção de dispositivos de manobra dentro de um modelo barra-ramo, onde as barras não possuem a capacidade de agrupamento, gera a necessidade de se criar conceitos ou classes específicas para lidar com algumas possibilidades operativas do sistema. Por vezes, devido à falta de equipamentos apropriados, à situação operativa ou – até mesmo – à economia nas instalações físicas, algumas linhas de transmissão operam com uma de suas extremidades

aberta para explorar seu potencial capacitivo, injetando potência reativa no sistema. Contudo o modelo deve ser capaz de interpretar tal prática, considerando a extremidade aberta de qualquer ramo como uma barra fictícia temporária. Esta barra seria criada durante a configuração lógica ou operativa do sistema e sua existência estaria vinculada a condição operativa ramo. A classe TBarraFicticia foi criada para dar esta adaptabilidade ao modelo, sendo responsável por simular as características próprias de uma barra terminal – como as injeções nulas de potência, a inexistência de monitoramento e a interligação única com ramo de origem – e por manter as propriedades físicas da barra substituída – como a classe de tensão e a subestação de situação. Os modelos que trabalham com a montagem físico-topológica abordam diferentemente a situação supracitada, uma vez que o modelo do sistema é resultante do processamento topológico baseado no estado operativo dos ramos lógicos (chaves e disjuntores) pertencentes ao modelo físico da rede.

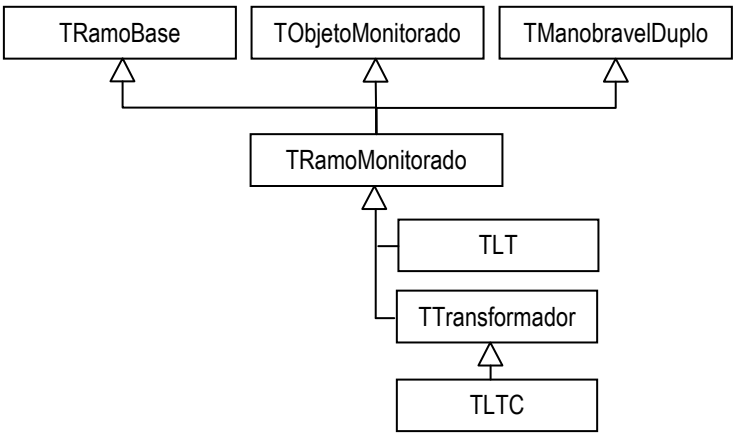


Figura 42 – Acoplamento e especialização da classe *TRamoBase*.

O modelo de SEP proposto tem o objetivo de representar o modelo barra-ramo usado em estudos de regime permanente, acrescentando a este a possibilidade de lidar com informações do sistema de monitoramento e de realizar a configuração primária de rede. Neste tipo de configuração as barras são previamente conhecidas e os ramos – bem como os elementos shunts – podem ser providos de disjuntores que fornecem as informações necessárias para a configuração lógica ou operativa do modelo. Portanto as classes descendentes dos ramos e shunts abstratos devem ser possuir uma estrutura capaz de manipular informação externa e de realizar as manobras operativas necessárias à configuração operativa do modelo.

A especialização das classes do núcleo abstrato que são responsáveis pelas interconexões, além de inserir a capacidade de manipular as informações do sistema de medição, deve proporcionar às classes descendentes a possibilidade de definir seu estado

operativo com base nos dispositivos de manobra. Portanto a especialização destas classes irá agregar à respectiva classe abstrata as características de monitoramento e de manobra desenvolvidas no núcleo de supervisão. A Figura 42 mostra a composição da classe da *TRamoMonitorado* através do uso de herança múltipla. Tal classe preserva todas as características da classe abstrata, agregando as funcionalidades desejadas e alterando somente as funções condicionais de configuração lógica ou operativa. Ressalta-se que classe a *TManobravelDuplo* pode ou não associar disjuntores a cada ponto terminal do ramo. No caso de inexistência de um dos disjuntores terminais, considera-se fechado o respectivo terminal para efeito da citada configuração.

As classes que descendem da classe *TRamoMonitorado* representam os elementos de interligação do modelo barra-ramo, possuindo todas as funcionalidades de supervisão e manobra implementadas. Dentro da modelagem proposta, foram implementados os seguintes componentes:

- ***Classe TLT***: Adiciona as propriedades específicas para a representação de linhas de transmissão através do circuito π equivalente, ou seja, a condutância, a susceptância série e a susceptância shunt, além de incluir os valores limites para capacidade de transmissão de potência em operação normal e emergencial. Todas as funções relativas aos parâmetros elétricos, fluxos de potência e suas derivadas em relação aos estados são formulados com base nas propriedades da linha;

- ***Classe TTransformador***: Adiciona as propriedades específicas para a representação de transformadores com TAP em fase através do circuito π equivalente, ou seja, a condutância, a susceptância série, a posição e o lado que se encontra o TAP, além de incluir os valores limites para capacidade de transmissão de potência em operação normal e emergencial. Todas as funções relativas aos parâmetros elétricos, fluxos de potência e suas derivadas em relação aos estados são formulados com base nas propriedades do transformador;

- ***Classe TLTC***: Herda todas as propriedades da classe *TTransformador* e inclui as propriedades próprias dos transformadores em fase com comutação sob carga, ou seja, os valores limites de variação do TAP, o número de passos entre a posição mínima e máxima e a indicação da barra de tensão controlada.

As classes comentadas acima constituem os conceitos finais que serão utilizados na aplicação do modelo barra-ramo aos estudos de regime permanente, entretanto tal fato não limita ou restringe futuras especializações que utilizem ou trabalhem com outros parâmetros elétricos, como a representação de linhas de transmissão utilizando parâmetros distribuídos ou modelos descritivos para LTs longas, ou que implementem conceitos não abordados no

presente trabalho, como os transformadores defasadores. Ressalta-se a não implementação de classes compostas que envolvem dois ou mais componentes, como os transformadores de três enrolamentos, devido a este tipo de atribuição ser própria dos configuradores de rede. Os modelos de tais componentes devem ser compostos diretamente nos arquivos de descrição de redes.

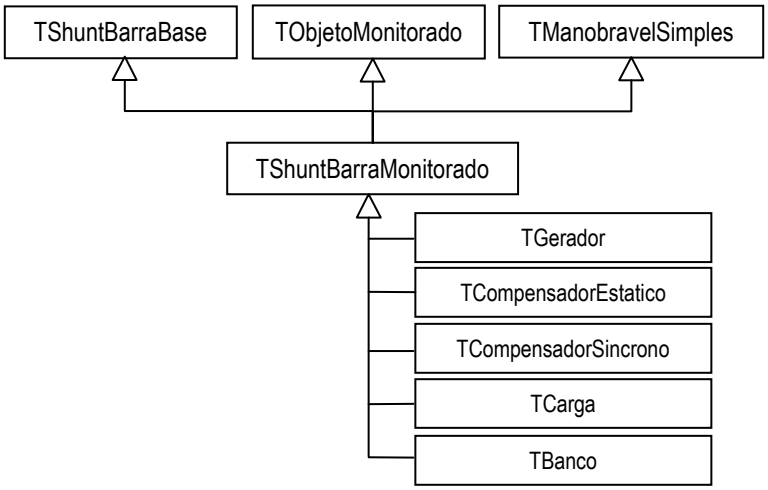


Figura 43 - Acoplamento e especialização da classe *TShuntBarraBase*.

A especialização e acoplamento das classes derivadas de TShuntBarraBase, que representa todos os elementos shunt associados a uma determinada barra, é desenvolvido de forma análoga àquela realizada as classes descendentes de TRamoBase. A classe TShuntBarraBase preserva todas as características de sua classe antecessora do núcleo de abstrato, com exceção da função de configuração lógica.

Todos os componentes do modelo barra-ramo que estão dispostos fisicamente como elementos shunts vinculados a barra, independentemente de seu tipo de controle ou de sua atuação ativa ou passiva em determinado estudo em regime permanente, descendem da classe TShuntBarraMonitorado. Tais componentes, mesmo que não utilizados diretamente, contribuem para a representação do modelo e disponibilizam informações de configuração física do sistema (mais especificamente sobre a barra). O modelo proposto por este trabalho, sem a pretensão de ser exaustivo, abrangeu os seguintes componentes:

- ***TGerador:*** Representa o conceito de grupo gerador associado aos estudo de regime permanente, acrescentando as seguintes propriedades operativas: injeção/absorção de potência ativa, geração mínima de potência ativa, geração máxima de potência ativa, injeção/absorção de potência reativa, geração mínima de potência reativa, geração máxima de potência reativa e

tensão terminal. Todas as funções relativas às condições operativas, as grandezas elétricas e suas derivadas em relação aos estados são formulados com base nas propriedades do gerador;

- ***TCompensadorEstatico:*** Representa o conceito de compensador estático de reativos e adiciona as seguintes propriedades específicas referentes a seu modelo em estudos em regime permanente: injeção/absorção de potência reativa; geração mínima de potência reativa; geração máxima de potência reativa; Valor da tensão, em p.u., da barra de tensão controlada; indicador da barra de controle; coeficiente de inclinação da área linear da curva de controle; e indicar do modo de controle por corrente ou por potência. Todas as funções relativas às condições operativas, as grandezas elétricas e suas derivadas em relação aos estados são formulados com base nas propriedades específicas do componente;

- ***TCompensadorSincrono:*** Representa o conceito de compensador síncrono e adiciona as seguintes propriedades referentes a seu modelo em estudos em regime permanente: injeção/absorção de potência reativa; geração mínima de potência reativa; geração máxima de potência reativa; Valor da tensão, em p.u., da barra de tensão controlada; e indicador da barra de controle. Todas as funções relativas às condições operativas, as grandezas elétricas e suas derivadas em relação aos estados são formulados com base nas propriedades específicas do componente;

- ***TCarga:*** Representa o conceito de carga através do modelo exponencial e polinomial aplicados em estudos em regime permanente, acrescentando as propriedades específicas de cada modelo, ou seja, as potências ativa e reativa de definição, os expoentes ativo e reativo de tensão (exponencial), os coeficientes ativos e reativos do polinômio de segundo grau (polinomial) e o indicador do modelo a ser utilizado. Todas as funções relativas aos parâmetros elétricos, às grandezas elétricas e suas derivadas em relação aos estados são formulados com base nas propriedades do modelo de carga;

- ***TBanco:*** Representa o conceito de banco de capacitores e reatores aplicado a estudos em regime permanente através da definição da susceptância shunt e da formulação das funções relativas aos parâmetros elétricos, às grandezas elétricas e suas e suas derivadas em relação aos estados associado ao modelo do componente;

A distinção dos citados componentes é realizada através das propriedades abstratas que definem o tipo de elemento e de controle (ativo ou passivo) e proporcionam a associação de variáveis controladas. No caso da EESP, somente os elementos shunt de controle passivo são utilizados no estudo – restando aos elementos de controle ativo o subsídio informações relativas configuração do sistema.

Os modelos acima representados não limitam futuras especializações que trabalhem com parâmetros ou modelos elétricos distintos ou que abordem conceitos não implementados nesta proposta.

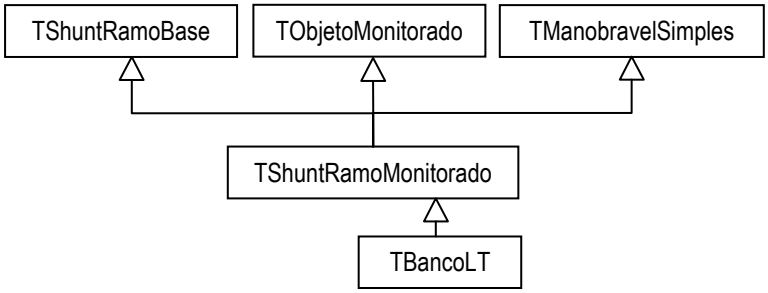


Figura 44 - Acoplamento e especialização da classe *TShuntRamoBase*.

As mesmas considerações feitas para as especializações dos elementos shunt associados à barra, aplicam-se às classes descendentes de TShuntRamoBase. O objetivo da implementação deste tipo de componente, como mencionado anteriormente, é de modelar adequadamente os elementos shunt dispostos após os disjuntores das linhas de transmissão. Apesar de esta configuração ser possível para a maioria dos componentes shunts implementados, visualiza-se sua aplicação somente aos bancos de capacitores e reatores. Todos os aspectos e características próprias deste banco são iguais àqueles implementados para a classe TBanco. A principal distinção entre as classes está na forma de abordagem dos referidos componentes pelas classes proprietárias. No caso da classe TBancoLT, ou qualquer classe derivada de TShuntRamoBase, suas propriedades e grandezas elétricas compõem as propriedades e grandezas da LT (ou qualquer outra classe de interligação). Portanto a potência reativa injetada/consumida pelo banco, como a sua susceptância shunt, são consideradas conjuntamente no fluxo reativo e na susceptância shunt da LT, ou seja, a existência do banco é transparente para as barras terminais da LT. Tais considerações também são aplicáveis, por óbvio, as derivadas das referidas grandezas.

A utilização de medidores associados a grupos de componentes, como subestações ou sistemas, apesar de não proporcionar qualquer benefício direto às funções como EESP, fluxo de potência ou análise de contingências, adicionam informações globais que podem ser úteis aos operadores de sistema e funções que trabalham com correlações de dados, como a previsão de carga de curto prazo. Tais medições podem ser relativas a informações climáticas – como a temperatura média, o índice pluviométrico regional ou umidade relativa do ar – ou a informações sintetizadas – como a potência ativa consumida por uma determinada região. Independentemente do seu uso, o modelo deve ser capaz de permitir a

vinculação de medidores ao sistema – mesmo que não sejam medidores de fato, mas sim indicadores compostos por outras medidas.

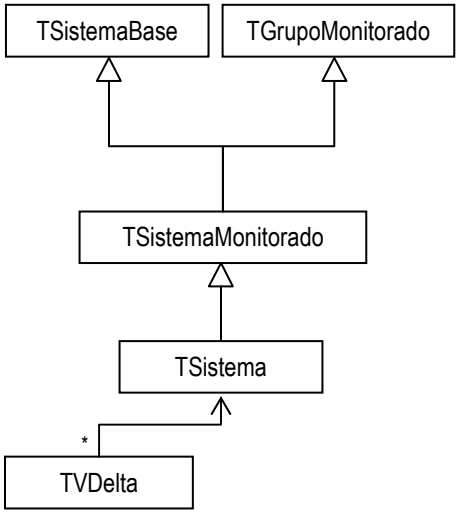


Figura 45 - Acoplamento e especialização da classe *TSistemaBase*.

A Figura 45 mostra as sucessivas especializações que dão origem a classe TSistema. A primeira especialização resulta na classe TSistemaMonitorado, a qual preserva todas as características de sua classe antecessora oriunda do núcleo abstrato e insere a capacidade de manipular informação externa através do uso de medidores globais. A segunda especialização (TSistema) adiciona uma propriedade para identificação alfanumérica, um valor angular para a barra de referência e limites de carga para classificação da condição operativa do sistema, bem como acrescenta algumas funções para manipulação do estado operativo – como a leitura e gravação de um determinado ponto operativo – e funções associadas à geração de arquivos descritores primitivos (os quais independem das classes de leitura e gravação que serão analisadas oportunamente dentro do pacote de tradutores). A classe TSistema representa e exterioriza o conceito proposto de sistema para as demais classes que compõem a aplicação.

A classe TVDelta é responsável pela montagem e configuração do sistema através do uso de classes de tradução de arquivos descritores. Na Figura 45 é mostrada a relação entre esta classe e a classe TSistema, entretanto a classe TVDelta possui a mesma relação com todos os objetos do modelo de proposto. Ressalta-se o cunho organizacional da referida classe, portanto sua relação com as demais classes resulta do processo natural de cadastro e gerência de objetos. Como a representação de todas as relações tornaria o diagrama redundante e excessivo, optou-se por sua representação resumida e explicada.

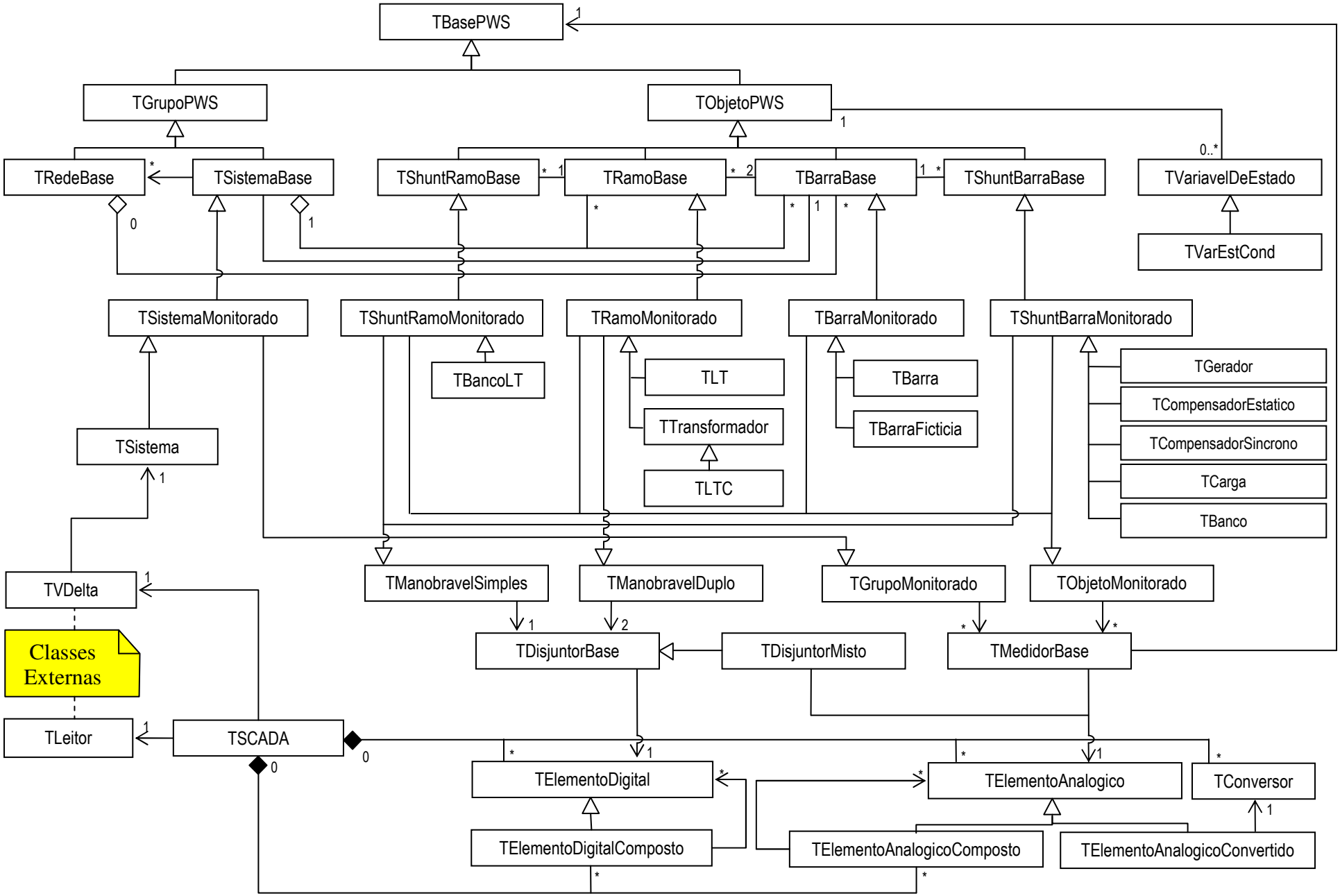


Figura 46 – Diagrama completo do modelo de SEE proposto.

4.3.3.3 Considerações finais sobre o modelo de SEE proposto

O modelo de SEE proposto tem a finalidade de proporcionar a qualquer estudo em regime permanente uma a estrutura de classes que possibilite a representação direta e flexível do modelo barra-ramo e de todas as suas propriedades elétricas inerentes. Ressalta-se que a EESP baseia-se nas informações do modelo proposto, entretanto a utilização deste restringe-se a montagem do problema e a atualização de grandezas. Preza-se pela flexibilidade e pela independência das classes que compõem estrutura do estimador de estados e das ferramentas matemáticas, haja vista a aplicabilidade desta função a outros modelos que possam advir futuramente. Portanto o objetivo deste trabalho é desenvolver e analisar os estimadores de estados aplicados a sistemas realísticos e, conseqüentemente, as ferramentas matemáticas envolvidas na solução do problema.

O diagrama do modelo proposto, abrangendo a maioria das classes apresentadas nesta subseção, é mostrado na Figura 46.

4.3.4 Tradutores de arquivos para SEE

A tecnologia de armazenamento de dados, por se tratar de uma área computacional com ampla aplicabilidade e de fundamental importância para diversos setores da economia que trabalham e lidam com uma quantidade gigantesca de informação, encontra-se em processo contínuo de desenvolvimento. O mercado de sistemas gerenciadores de banco de dados (SGBD) corresponde a um dos setores mais lucrativos e competitivos da economia computacional. Nas últimas décadas foram desenvolvidas diversas ferramentas e inúmeros sistemas que aprimoravam gradativamente técnicas e processos visando incrementar a confiabilidade ou a flexibilidade e reduzir a velocidade de acesso aos dados. Atualmente existem linguagens específicas para a manipulação de bancos de dados – como a Structured Query Language (SQL) –, diversos SGBD – como o Oracle Database 10G – e diferentes metodologias – como os bancos de dados relacionais e os bancos de dados orientados a objeto.

A evolução da tecnologia de armazenamento é diretamente relacionada à evolução das aplicações principais. As ferramentas computacionais voltadas para o setor energia elétrica, por sua própria historicidade, sempre utilizaram o armazenamento em extensos arquivos texto com formatos variados e baseados nos procedimentos próprios da linguagem de desenvolvimento. Apesar de algumas aplicações introduzem uma abordagem diferenciada, como em [34], grande parte das ferramentas computacionais dos centros de controle de sistemas ainda se baseiam nesta metodologia. Tal característica aplica-se ao antigo estimador de estados implementado na

CEEE, o qual possui uma estrutura de entrada de dados semelhante ao ANAREDE e às demais ferramentas desenvolvidas pelo CEPEL.

As ferramentas de cunho acadêmico, que formam a base de muitas ferramentas comerciais, estão mais fortemente vinculadas a formas de armazenamento simplificadas – normalmente baseadas em arquivos textos estruturados – e que são implementadas por rotinas nativas das linguagens de programação. Por razões práticas, as linguagens de programação científicas – como o FORTRAN – enraizaram-se neste meio de desenvolvimento, dando origem a inúmeras ferramentas que utilizam este tipo de entrada de dados. Tais arquivos formam a grande plataforma de testes para as ferramentas em desenvolvimento, impondo a necessidade da implementação de dispositivos de leitura destes padrões ou de ferramentas de conversão de dados.

O presente trabalho prima pela aplicabilidade do estimador de estados à estrutura e à seqüência operativa do COS da CEEE. Logo a mudança ou a imposição de metodologia estranha àquela utilizada no COS geraria dificuldades na implementação e, consequentemente, na aceitação da nova ferramenta. Portanto o estimador deve reconhecer o arquivo da CEEE e aceitar este formato como o padrão de entrada de dados, dando continuidade a cultura operacional da empresa e facilitando a introdução da nova ferramenta. Para suprir essa necessidade e possibilitar os testes comparativos durante a avaliação da ferramenta, foram desenvolvidas classes de leitura e validação de dados. Tais classes reconhecem dois padrões distintos de arquivos descritores de sistemas – ambos detalhados no Anexo B –, cujas descrições são baseadas no modelo barra-ramo. Trata-se do padrão comercial do antigo EE da CEEE e do padrão acadêmico do EE desenvolvido no LABSPOT/CTC/EEL da Universidade Federal de Santa Catarina (UFSC). A Figura 47 mostra o diagrama das referidas classes.

4.3.4.1 – Padrão do EE da LABSPOT/CTC/EEL

O processo de verificação e validação de sistemas ou ferramentas computacionais constitui fase de fundamental importância dentro do processo de desenvolvimento de sistemas, sendo – inclusive – umas das fases de desenvolvimento mais estudadas dentro da engenharia de software. O plano de testes abrange desde testes de códigos e unidades específicas até testes de integração e de aceitação. Tais testes visam validar o conjunto de requisitos que o sistema deve atender – incluindo as propriedades emergentes – e eliminar ou identificar possíveis erros, problemas ou impropriedades.

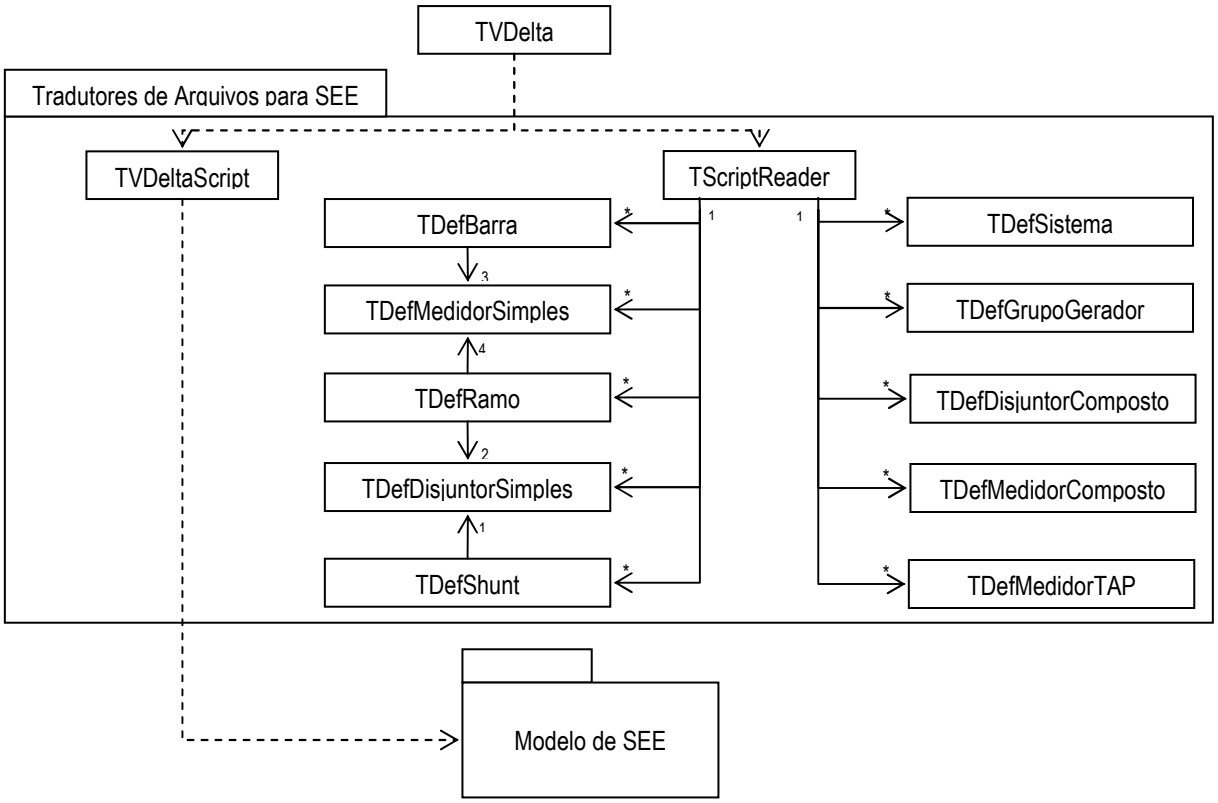


Figura 47 – Classes responsáveis pela leitura e validação de dados.

O uso de sistemas comparativos é comum dentro do âmbito da validação de sistemas, especialmente quando o sistema em desenvolvimento executa uma quantidade exaustiva de operações e/ou possui alta complexidade. Neste contexto, optou-se pela utilização do estimador de estados desenvolvido no LABSPOT/CTC/EEL da Universidade Federal de Santa Catarina (UFSC), utilizado em [55], pois este aborda tanto estimadores de míninos quadrados ponderados (MQP) quanto estimadores de mínimos quadrados com ponderação variável (MQPV).

A classe TVDeltaScript foi desenvolvida para viabilizar a leitura do arquivo padrão EE/LABSPOT e criar diretamente os objetos componentes do modelo barra-ramo proposto. Para tanto, esta classe possui um conjunto de listas estruturadas e um conjunto de procedimentos de leitura, validação e criação. Toda a informação contida no arquivo descritor é armazenada nas listas estruturadas de dados que irão subsidiar o processo de validação de informações. Após a validação, a classe tradutora passa a criar os componentes e seus objetos correlacionados e a montar gradualmente o sistema. Ao término deste processo, a classe disponibilizará um indicador de status, um arquivo de descrição de leitura, as listas de componentes e o objeto que representa o sistema.

A criação direta é possibilitada tanto pela simplicidade estrutural do arquivo descritor quanto pela fácil correlação das informações que compõem o sistema. Ressalta-se que o cunho acadêmico deste padrão é outro fator facilitador, uma vez que os erros de validação terão as suas conseqüências limitadas ao resultado do estudo. O formato deste padrão é detalhado no anexo B deste trabalho.

4.3.4.2 – Padrão do EE da CEPEL/CEEE

O formato padrão do arquivo descritor de sistema utilizado no COS da CEEE é baseado no difundido padrão do software de fluxo de potência do CEPEL, conhecido como ANAREDE. Tal característica é justificada tanto pela época do desenvolvimento da ferramenta, durante o início da década de 90, quanto pelo seu desenvolvedor ser o próprio CEPEL. Trata-se de um formato adaptado para reconhecer o sistema de manobra e supervisão, mas com a mesma estrutura de comandos e de cartões de leitura encontrado nas diversas ferramentas para SEE desenvolvidas por aquele centro de pesquisas.

Diferentemente da estrutura simplificada e direta do arquivo descritor do EE/LABSPOT, o descritor do EE/CEPEL vincula uma quantidade significativa de informações a cada componente do sistema, estruturando a informação parcialmente no cartão de leitura do componente e parcialmente em cartões específicos de detalhamento do objeto correlacionado. Ademais, parte das informações contidas no arquivo descritor não é efetivamente utilizada pelo estimador de estados, mas pelas demais funções que compõem a seqüência operativa do COS. Portanto uma análise mais aprofundada das informações deve ser feita para evitar que erros de interpretação influenciem o resultado da estimação de estados ou o resultado das outras funções que utilizam as informações geradas pelo EE.

A criação direta dos objetos componentes do sistema pela classe responsável pela leitura deste padrão geraria funções de validação demasiadamente longas e complexas, prejudicando futuras adaptações e evoluções na ferramenta. A classificação da informação contida em cada campo do cartão de leitura, que determina qual informação é primordial para o estudo e para a seqüência operativa, por ser dependente das informações contidas em outros campos, também resultaria em funções de leitura e de decomposição de dados extensas. Para limitar a complexidade da classe de tradução principal e simplificar seu código, optou-se pela criação de classes responsáveis pela leitura, validação parcial e classificação de cada componente do sistema. A classe principal seria responsável pela interpretação da seqüência de comandos e pela leitura das informações mais simples. A

criação e validação global do sistema ficaria a cargo da classe TVDelta e teria como base o conjunto de objetos de definição de componentes disponibilizados pela classe principal.

A Figura 47 mostra a classe TScriptReader, responsável por coordenar a leitura e identificar cada comando ou operação no arquivo descritor, e seu conjunto de listas de classes de definição de componentes. Tais classes possuem todas as propriedades do componente representado e seus respectivos relacionamentos diretos. Durante a varredura de um determinado cartão, a linha do arquivo que define o componente é repassada ao objeto responsável por sua leitura. Este objeto irá verificar a existência de erros de formatação, analisar os conflitos ou omissões de informações, gerar as mensagens de leitura e indicar se as informações repassadas são suficientes para a criação do componente.

Os medidores e disjuntores são definidos inicialmente dentro dos cartões dos componentes do sistema – como as barras, ramos e shunts – e podem ser redefinidos dentro de cartões específicos. Neste último caso, a redefinição importará na criação de um medidor ou disjuntor composto.

As demais informações sobre este formato são apresentadas no anexo B deste trabalho. Ressalta-se que esta seção limita-se a apresentar uma visão geral sobre as classes, as estruturas e os procedimentos de entrada de dados.

4.3.5 – Aquisição de dados

A aquisição de dados no âmbito das ferramentas computacionais que compõem as funções do COS ou COD não se confunde com a aquisição de dados propriamente dita realizada pelo SCADA. As ferramentas computacionais buscam ou são motivadas a buscar as informações operativas do sistema diretamente de um banco ou tabela de dados residente em um servidor, seja este um servidor do histórico operativo ou um servidor de consolidação das informações de tempo-real. O servidor que consolida as informações de supervisão do SCADA é o elo entre a seqüência operativa em tempo-real e o complexo conjunto de tecnologias de telemetria existentes no centro de operações. Para esclarecer o exposto, passa-se a comentar – sem demais aprofundamentos – a arquitetura de comunicação e o processo de varredura e armazenamento.

4.3.5.1 – Arquitetura de comunicação

A arquitetura dos sistemas de comunicação e automação no âmbito das subestações e usinas envolve um diverso conjunto de tecnologias que abrange diferentes meios e

protocolos de comunicação. O surgimento de novos dispositivos eletrônicos inteligentes (IED – *Intelligent Electronic Device*) aplicados ao controle, à proteção e à supervisão que exploravam diferentes tecnologias – como relés numéricos, controladores eletrônicos, medidores digitais e PLCs – resultou em complexas redes de comunicação, gerando nas últimas décadas a busca por padrões que promovessem a interoperabilidade e o crescimento sustentável deste mercado [56].

As redes das subestações e usinas são normalmente compostas por microcomputadores, equipamentos de comunicação e diversos tipos de IEDs dispostos em um ou mais segmentos. Dentre os componentes da rede, a unidade terminal remota (RTU – *Remote Terminal Unit*) destaca-se por ser o componente responsável pela comunicação com o centro de operações. A RTU é normalmente é um microcomputador industrial ou um PLC que exerce a função de roteador, controlando e mantendo o link de comunicação IED-COS, e que pode executar algumas funções de controle e supervisão descentralizadas. A RTU pode trabalhar em conjunto com um concentrador de dados e controlar a supervisão nos casos de perda do link de comunicação.

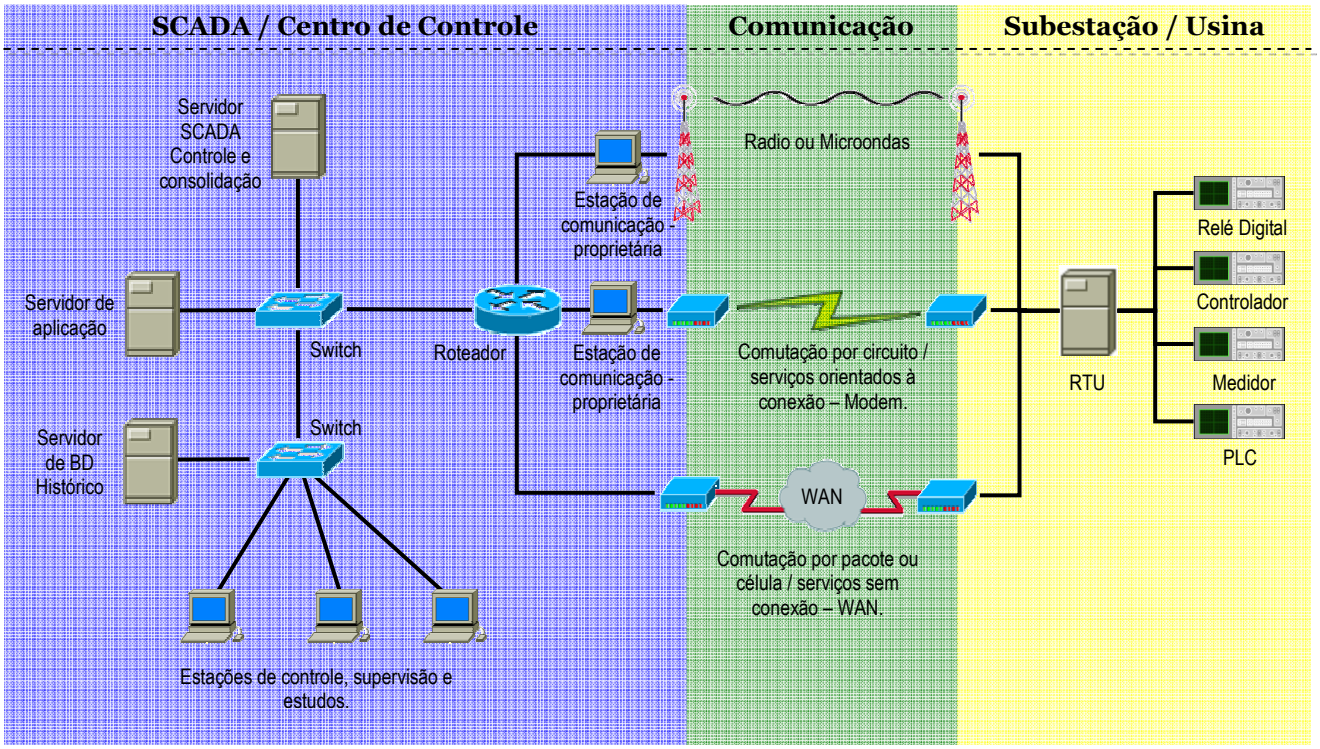


Figura 48 – Arquitetura simplificada da comunicação entre o COS e as subestações e usinas.

A comunicação entre a RTU e o COS é regida por protocolos de transmissão de dados padronizados pelo IEC (*International Electrotechnical Commission*) ou aceitos como padrão pelo mercado. Nesta área encontram-se os protocolos de transmissão de dados que

implementam funções de telemetria e telecomando básicas, como o IEC 60870-5-101 e o DNP 3.0, os protocolos de transmissão para a interface com equipamentos de proteção, IEC 60870-5-103, os protocolos de transmissão que permitem desempenhar as funções básicas de telemetria e telecomando com completo acesso às redes TCP/IP comerciais (4 camadas), como o IEC 60870-5-104 e as novas versões do DNP 3.0, e protocolos de transmissão que permitem desempenhar as funções básicas de telemetria e telecomando através do acesso seguro às redes TCP/IP (implementando sistemas de autenticação e criptografia), como IEC 60870-6-503.

O controle da comunicação dentro do COS normalmente é realizado por computadores ou equipamentos dedicados que interpretam os protocolos de transmissão e se comunicam com o servidor SCADA através da tecnologia da LAN local. Estes computadores também podem executar as tarefas de varredura cíclica do sistema de telemetria, realizando a consolidação parcial e repassando periodicamente estes valores ao servidor. É comum encontrar softwares proprietários de fabricantes de equipamentos de medição e telemetria rodando nestes equipamentos, mas não é raro que esse sistema seja desenvolvido pela própria empresa concessionária de transmissão ou distribuição – como o caso da CEEE. Ressalta que o uso de protocolos baseados em TCP/IP (IEC 60870-5-104 ou IEC 60870-5-503) possibilita a comunicação direta entre o servidor SCADA e o RTU. A Figura 48 ilustra a arquitetura simplificada descrita acima.

4.3.5.2 – O processo de varredura e armazenamento

A varredura do sistema de telemetria é realizada a cada intervalo pré-determinado de tempo, sendo este procedimento controlado diretamente pelo servidor SCADA ou por intermédio do recurso cíclico do protocolo de transmissão. Ao fim da varredura ou do tempo máximo de resposta, uma tabela com as informações operativas é montada e disponibilizada para o servidor de aplicação.

A seqüência operativa pode ser iniciada temporariamente pelo servidor de aplicação, o qual buscará a tabela consolidada no servidor SCADA, ou iniciada pelo próprio servidor SCADA após a conclusão da varredura – através do disparo de um evento (chamada) ao servidor de aplicação.

O armazenamento das informações operativas é efetuado em intervalos maiores de tempo que o ciclo de varredura – normalmente a cada minuto – e realizado por um processo permanente que é executado no servidor SCADA, no servidor de aplicação ou no servidor do

banco de dados operativo. Caso resida no servidor SCADA, o processo efetuará a gravação da tabela com as informações operativas ao término do intervalo de armazenamento. Se residir no servidor de aplicação, o processo gravará uma tabela disponibilizada pela seqüência operativa que normalmente contém as informações de telemetria acrescidas de alguns resultados ou informações do processamento. No último caso, o processo buscará as informações na tabela do servidor SCADA ou na tabela do servidor de aplicação. No caso da CEEE, o processo de armazenamento reside no servidor de aplicação e armazena algumas informações resultantes da estimação de estados.

4.3.5.3 – Classes de leitura

O procedimento de aquisição de dados por parte das funções ou ferramentas operacionais do COS é dependente da metodologia adotada pela concessionária para o tratamento das informações operativas. Tal metodologia pode levar à implementação de funções de leitura de dados e/ou ao desenvolvimento de processos de comunicação dedicados. A leitura de dados se fará nos arquivos temporários disponibilizados pelo sistema SCADA ou nas bases de dados do histórico operativo. Por outro lado, podem ser desenvolvidos processos dedicados de comunicação para a passagem de informações operativas baseando-se na arquitetura cliente-servidor ou na arquitetura de objetos distribuídos.

A dependência a uma metodologia específica resulta na necessidade de se desvincular da classe *TSCADA* o procedimento de leitura das informações operativas e repassá-lo a uma classe abstrata que possa ser posteriormente customizada. Tal posicionamento gera a independência do modelo proposto em relação à cultura do COS e concentra o trabalho de implementação da ferramenta no desenvolvimento de uma classe compatível com a metodologia de trabalho da concessionária. A Figura 49 mostra o diagrama de classe do pacote “Aquisição de dados – Leitores” e introduz o conceito da classe *TLeitor*.

A classe *TLeitor* possui referências às listas de elementos digitais e analógicos da classe *TSCADA,* as quais são repassadas por funções específicas e limitam-se aos elementos de informação simples (por serem os únicos que se vinculam às informações externas). A leitura pode ser feita com base nas informações em tempo real ou através da definição de um momento pretérito. A distinção é feita através de uma propriedade binária definida como ‘TempoReal’ e que indica o modo de leitura a ser utilizado (1 = Tempo Real ou 0 = Histórico) . Caso se opte por uma leitura na base histórica de dados, deve-se informar a data (‘Data’) e a

hora (‘Hora’) da “foto” (conjunto de informações operativas relativas a um determinado momento) a ser carregada. Por fim, a classe TLeitor define uma função virtual destinada à leitura ou a aquisição das telemetrias. Tal função deve ser redefinida em suas classes descendentes, conforme a metodologia de acesso do COS em questão.

A classe *TSCADA* possui uma relação de agregação compartilhada com a classe *TLeitor*, sendo a criação e a destruição das instâncias da classe *TLeitor* ou de suas classes descendentes uma responsabilidade da classe *TVDelta*. Portanto a classe *TSCADA* possui um objeto da classe *TLeitor* em sua estrutura, mas não detém seu controle. Este objeto é, na verdade, uma propriedade de classe que deve ser repassada ou definida externamente. Tal prática desvincula a classe *TSCADA* das futuras especializações da classe *TLeitor*.

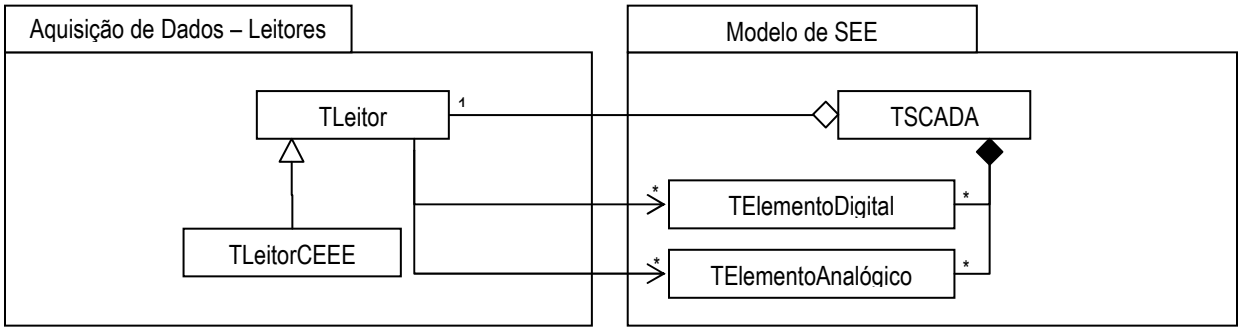


Figura 49 – Digrama de classes: Aquisição de dados

Os procedimentos de aquisição de dados e de acesso ao histórico operativo da CEEE foram implementadas através da classe *TLeitorCEEE*. Tais procedimentos são voltados à leitura de arquivos binários estruturados que são utilizados na operação em tempo real e na formação da base de dados histórica. Estes arquivos são compostos por uma estrutura com informações gerais da “foto” e por um conjunto de estruturas simples que representam os elementos analógicos e digitais do sistema de telemetria. Ressalta-se que o arquivo relativo às informações em tempo-real reside no servidor SCADA e é acessado periodicamente pelo EE da seqüência operativa (tempo-real), enquanto os arquivos que formam a base histórica residem no servidor de banco de dados e são acessados pelo EE em modo estudo. A formatação detalhada destes arquivos não pode ser mencionada neste trabalho por se tratar de uma propriedade da concessionária de transmissão financiadora do projeto.

As estruturas de gravação dos elementos digitais e analógicos são compostas pelo código ou indicador do elemento, pelo estado (binário) ou valor (real) medido e pelo status da leitura (normal ou falhada). Por sua vez, uma “foto” representa o conjunto estruturado de valores digitais e analógicos obtidos do sistema de telemetria num dado momento. Cada

arquivo que forma o histórico operativo contém um total de 1440 “fotos” que correspondem à gravação diária minuto a minuto das informações disponibilizadas pelo sistema SCADA acrescidas de algumas informações resultantes da estimação de estado em tempo real.

4.3.6 – Ferramentas matemáticas

Atualmente não se pode imaginar o desenvolvimento de aplicativos e ferramentas no âmbito da operação e do planejamento de sistemas elétricos de potência sem considerar o adequado suporte matemático-computacional. Qualquer sistema desta área que almeje eficiência e confiabilidade deve tratar prioritariamente a estruturação dos dados matemáticos e a racionalização das funções serem implementadas, tentando – na medida do possível – dar flexibilidade e independência estes elementos. Devido à limitação de memória e de capacidade de processamento, a computação matemática voltada para sistemas elétricos de potência desenvolveu-se com base na economia de memória e na redução das operações matemáticas, construindo uma extensa literatura técnica [59-77] que deve ser observada na implementação de qualquer projeto.

Esta subseção destina-se a apresentar todo o pacote matemático desenvolvido para dar suporte às funções e estudos de SEE. Apesar de ser total independente das demais classes que compõem o projeto – podendo ser aplicada, sem prejuízo, em quaisquer outros estudos ou áreas –, o presente pacote sofreu influência da literatura técnica voltada à estimação de estado. Cabe ressaltar que foram desenvolvidas ferramentas para teste, validação e comparação das classes matemáticas, e que os resultados que serão apresentados foram obtidos com o uso destas ferramentas. Esta subseção esta estruturada da seguinte maneira: vetores e matrizes;

4.3.6.1 – Vetores e matrizes

O pacote matemático abordou de forma abrangente e flexível as principais estruturas de dados utilizadas pelas ferramentas computacionais e funções voltadas a SEP. Foram desenvolvidas classes específicas para o tratamento de vetores e matrizes completos e, principalmente, esparsos. Tais classes podem trabalhar com qualquer tipo de dado completamente definido – compreendido como aquele que possui a definição de todas as operações aritméticas, atributivas e comparativas –, uma vez que todas as referidas classes são do tipo *templates*. Esta característica possibilita que a definição do tipo de dado necessário a determinado estudo seja realizada durante o uso da classe, gerado flexibilidade durante o processo de implementação do sistema. Outra característica relevante destas classes é a

facilidade em se realizar as operações aritméticas básicas envolvendo vetores e matrizes completos ou esparsos e operações atributivas, gerando a capacidade de tratar tais classes – para efeito matemático – como variáveis simples. A Figura 50 apresenta de forma não detalhada o conjunto de classes supracitado.

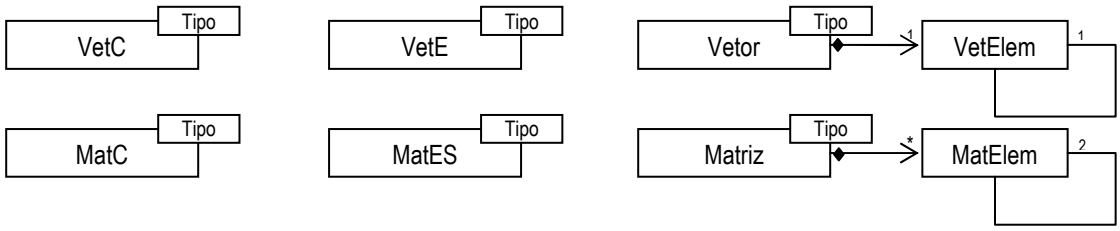


Figura 50 – Diferentes classes que abordam o conceito de vetores e matrizes.

Os conceitos de vetores e matrizes completos foram implementados nas classes *VetC* e *MatC* respectivamente. Trata-se de uma abordagem simplista que utiliza alocação seqüencial de memória conforme o tamanho ou dimensão do vetor ou da matriz. Ambas as classes possuem estruturas que se tornam ineficientes conforme a dimensão do vetor ou da matriz aumenta, resultando – portanto – numa utilização restrita. A Figura 51 mostra as propriedades internas das classes VetC e MatC, onde se pode ressaltar o uso de um vetor do tipo pós-definido alocado conforme a dimensão do respectivo objeto. No caso da matriz completa, optou-se pela implementação com um único vetor ao invés do uso de uma lista de vetores.

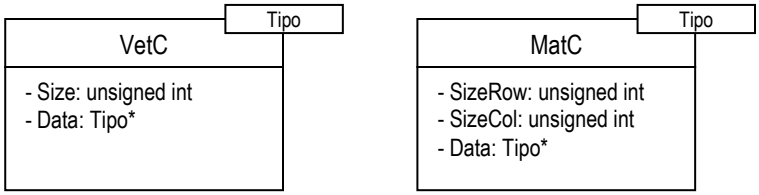


Figura 51 – Propriedades interbas das classes VetC e MatC

As classes *VetE* e *MatES* implementam o conceito de vetores e matrizes esparsos através do uso de uma estrutura interna direcionada ao armazenamento dos elementos não nulos que é baseada no uso vetores de índices. Neste tipo de abordagem, cada elemento não nulo é armazenado conjuntamente com seu respectivo índice, independentemente do tamanho ou dimensão do vetor ou da matriz.

Conforme ilustra a Figura 52, a classe *VetE* possui dois vetores: um para o armazenamento do valor e outro para o armazenamento do índice dos elementos não nulos do vetor. A quantidade de elementos não nulos, representado pela variável interna *FSize*, não corresponde ao tamanho do vetor, representado pela variável interna *FDimensao.*

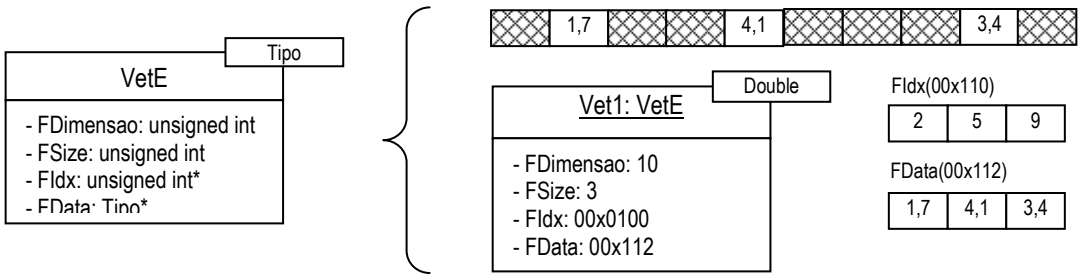


Figura 52 – Propriedades internas e exemplificação da classe VetE

A classe MatES possui variáveis que determinam a dimensão, o número de elementos não nulos, o número de linhas não nulas e vetores que indicam a posição inicial de cada linha, o índice de cada linha e o índice de cada elemento não nulo, além do vetor que armazena o valor de cada elemento não nulo. Observa-se que tal estrutura adiciona uma quantidade significativa de memória para armazenar os elementos não nulos, portanto seu uso deve se restringir às matrizes com elevada esparsidade. Cabe ressaltar que tal estrutura requer uma quantidade maior instruções para realizar qualquer operação que envolva colunas (como a multiplicação ou a fatoração orientada por coluna) e prejudica a geração segura de referências de memória aos dados dos vetores internos, apesar ser eficiente durante operações de cópia ou operações que envolvam linhas (como a soma ou subtração ou fatoração orientada por linha).

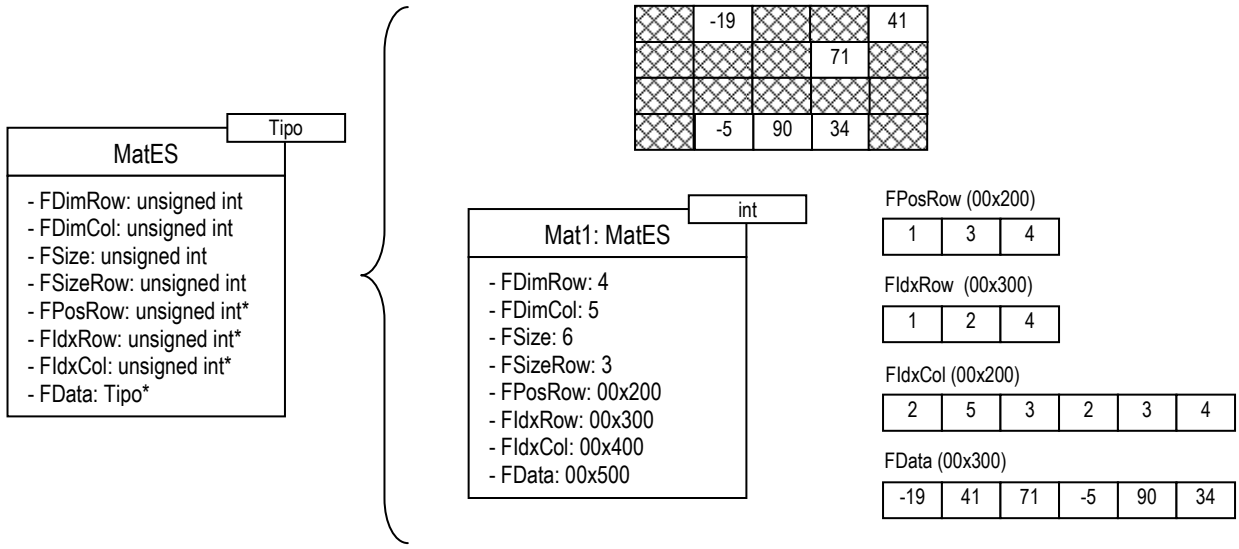


Figura 53 - Propriedades internas e exemplificação da classe MatES

Visando dar mais flexibilidade e aumentar a eficiência nas operações orientadas a coluna, implementou-se – através das classes Vetor e Matriz – o conceito de vetores e matrizes baseados em listas encadeadas. Tais estruturas, utilizadas em [24] e [29], são formadas por elementos vetoriais e matriciais que possuem ponteiros internos que indicam o próximo elemento da linha e da coluna, este último específico para matrizes. Tais elementos, além dos

ponteiros, armazenam os respectivos índices e os valores e são completamente definidos, ou seja, possuem todos os operadores aritméticos, comparativos e atributivos carregados. Tal característica proporciona uma grande simplificação tanto nos procedimentos de busca quanto na manipulação e gerenciamento da memória.

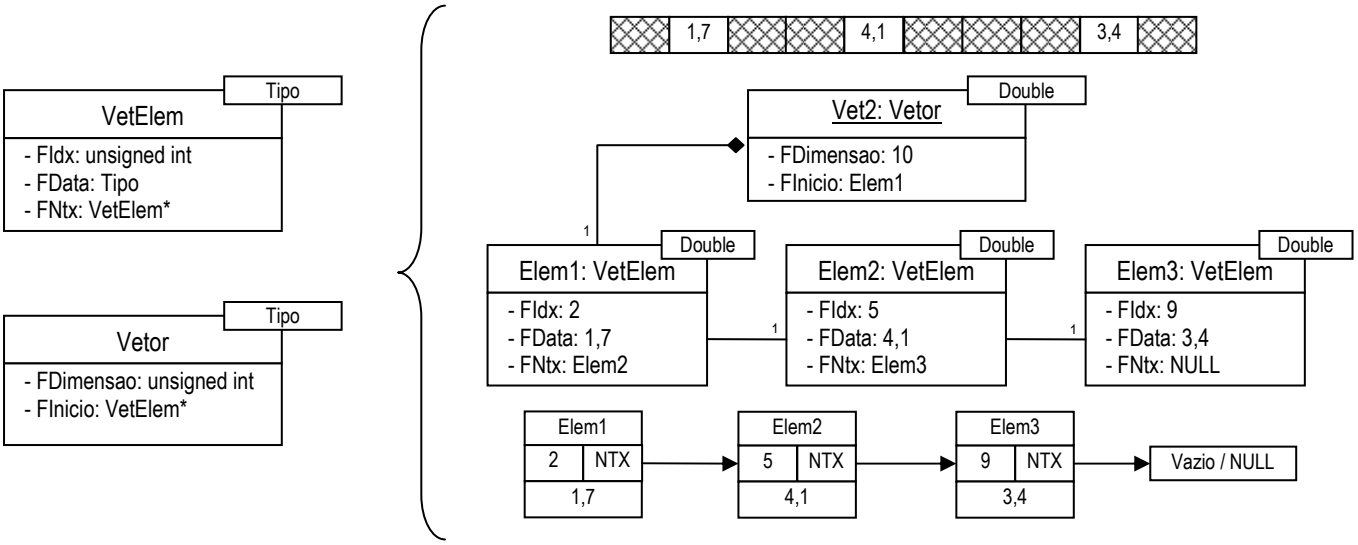


Figura 54 - Propriedades internas e exemplificação das classes *VetElem* e *Vetor*.

Apesar da aparente complexidade mostrada na Figura 54, as classes *Vetor* e *VetElem* são facilmente implementadas e manipuladas. Observa-se que a classe *VetElem* possui associação recursiva e concentra todas as informações referentes ao elemento do vetor. A classe *Vetor* é responsável por criar, associar, gerenciar e destruir as instâncias (objetos) da classe *VetElem*, entretanto possui somente um ponteiro que indica o início da lista encadeada. Ressalta-se a facilidade de inserção e remoção de elementos no vetor, de concatenar ou seccionar vetores e de gerenciar a memória alocada. Outra característica importante é a possibilidade de se realizar referências seguras a variável interna de qualquer objeto da classe *VetElem*, uma vez que alteração na seqüência da lista encadeada não altera a posição memória que destinada à instância da referida classe.

Os elementos matriciais que constituem a matriz baseada em listas encadeadas, implementados através da classe *MatElem*, além de possuírem as variáveis internas que armazenam o índice da coluna, o valor do elemento e o ponteiro para o próximo elemento da linha – como na classe *VetElem* –, possuem também variáveis internas que armazenam o índice da linha e o ponteiro para o próximo elemento da coluna. Desta forma, torna-se possível 'caminhar' tanto horizontal quanto verticalmente pela linha ou coluna do respectivo elemento.

A classe *Matriz*, de forma análoga à classe *Vetor*, é responsável pela criação, associação, gerenciamento e destruição das instâncias da classe MatElem, além de possuir todas as

funções matemáticas matriciais. Tal classe possui dois vetores de instâncias da classe MatElem que armazenam todos os primeiros elementos das linhas e colunas da matriz, possibilitando, assim, a orientação vertical e horizontal individualizada. Diferentemente da estrutura apresentada em [24] e [29], os vetores da classe *Matriz* são cheios, ou seja, possuem seu tamanho definido conforme sua dimensão. Tal procedimento simplifica o acesso a qualquer linha ou coluna, uma vez que a verificação da existência da linha ou da coluna é feita diretamente através da comparação com o elemento nulo (NULL) – apesar do aparente aumento dos recursos alocados. A Figura 55 exemplifica a estruturação das referidas classes.

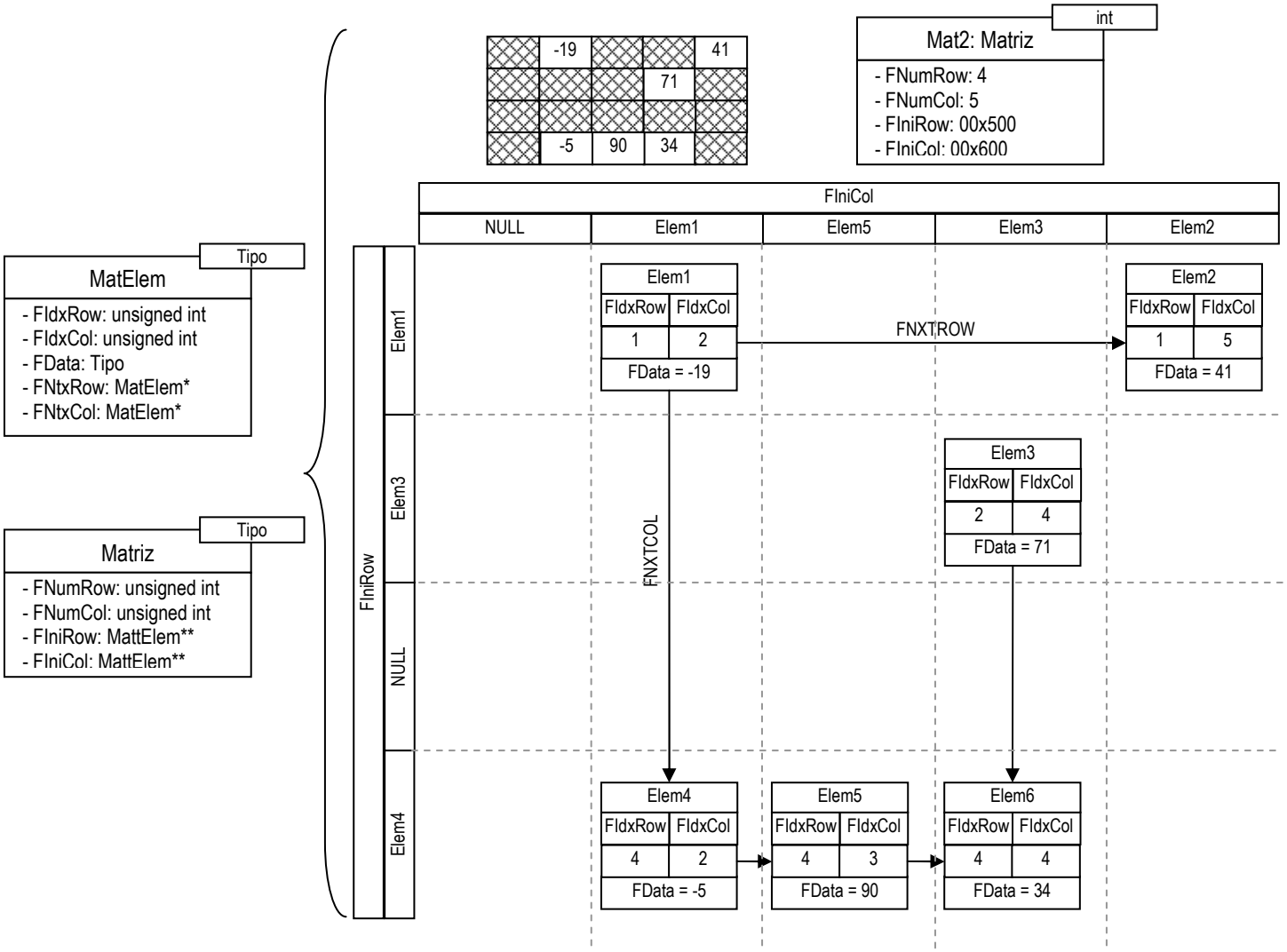


Figura 55 - Propriedades internas e exemplificação das classes *VetElem* e *Vetor*.

4.6.3.3 – Métodos de Fatoração

Apesar das inúmeras metodologias de fatoração relatadas pela literatura técnica [59-61] e da existência de outros métodos que apresentam propriedades numéricas semelhantes [62-63], o presente trabalho procurou implementar métodos de fatoração numericamente estáveis [64-66] que foram estudados e aplicados com sucesso no âmbito da EESP [67-68]. Trata-se de métodos de fatoração baseados na decomposição QR, os quais podem ser aplicados a matrizes esparsas de grande porte de forma seletiva. Tais métodos são variantes rápidas das rotações ortogonais de Givens [69] que possuem dois ou três multiplicadores e podem ser orientados tanto por linha quanto por coluna.

ORTOGONALIDADE [60]

Uma matriz $Q \in \Re^{mxm}$ é dita ortogonal se $Q^T Q = I$. Pode-se, portanto, provar que a norma euclidiana de qualquer transformação ortogonal permanece inalterada, pois se $Q^T Q = I$, então $\|Qx\|_2^2 = x^T Q^T Qx = x^T x = \|x\|_2^2$. Tal propriedade é fundamental na solução de problema de mínimos quadrados ponderados (MQP) envolvendo métodos de fatoração baseados na decomposição QR.

DECOMPOSIÇÃO QR [61]

Seja $A \in \Re^{mxn}, m \geq n$. Então existe uma matriz ortogonal $Q \in \Re^{mxm}$ tal que

$$A = QR, \tag{4.1}$$

onde $R \in \Re^{mxn}$ é uma matriz triangular superior com elementos diagonais não negativos. Esta decomposição é conhecida como *decomposição QR da matriz A*, sendo a matriz R chamada de *fator R* de A.

A prova da decomposição supracitada será realizada após a discussão das transformações ortogonais elementares, mais especificamente das rotações de Givens [69].

ROTAÇÕES DE GIVENS

A utilização de matrizes ortogonais para produzir rotações de planos com o intuito de zerar elementos matriciais foi proposta primeiramente por Jacobi como um método para calcular autovalores de matrizes simétricas. Tais rotações eram conhecidas como *rotações de Jacobi*. A utilização deste método na triangulação de matrizes foi proposta por Wallace Givens [69] e passou, desde então, a ser conhecido como *rotações de Givens*.

Trata-se de uma transformação ortogonal elementar que permite a introdução de zeros em vetores e matrizes de forma seletiva. Tal propriedade a torna vantajosa quando se está trabalhando com matrizes e vetores esparsos.

A matriz de rotação, conforme (4.2), é idêntica à matriz identidade, exceto pela alteração nos elementos que ocupam as posições (i, i), (i, j), (j, i) e (j, j) que são responsáveis pela rotação de θ radianos em sentido horário sobre coordenadas dos planos (i, j).

$$Q(i,k,\theta) = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & s & & & \\ & & -s & c & & & \\ & & & & \ddots & & \\ & & & & & 1 & \\ & & & i & j & & \end{bmatrix} \begin{matrix} \\ \\ i \\ j \\ \\ \\ \end{matrix} \rightarrow \begin{cases} c = \cos(\theta) \\ s = sen(\theta) \end{cases} \tag{4.2}$$

Aplicando a matriz de rotação a um vetor qualquer e impondo que o vetor resultante possua o elemento que ocupa a coordenada j nulo, tem-se:

$$\begin{bmatrix} 1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & & \vdots & & \vdots \\ 0 & \cdots & c & \cdots & s & \cdots & 0 \\ \vdots & & \vdots & \ddots & \vdots & & \vdots \\ 0 & \cdots & -s & \cdots & c & \cdots & 0 \\ \vdots & & \vdots & & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \end{bmatrix} \times \begin{bmatrix} v_1 \\ \vdots \\ v_i \\ \vdots \\ v_j \\ \vdots \\ v_m \end{bmatrix} = \begin{bmatrix} v_1 \\ \vdots \\ cv_i + svj \\ \vdots \\ -svi + cv_j \\ \vdots \\ v_m \end{bmatrix} \rightarrow \begin{cases} (i) \quad c^2 + s^2 = 1 \\ (ii) \quad -svi + cv_j = 0 \\ \therefore \\ c = \dfrac{v_i}{\sqrt{v_i^2 + v_j^2}} \\ s = \dfrac{v_j}{\sqrt{v_i^2 + v_j^2}} \end{cases} \tag{4.3}$$

Da mesma maneira, pode-se aplicar a matriz de rotação a qualquer matriz e impor que o primeiro elemento da linha j seja nulo, conforme (4.4). Esta rotação ortogonal irá modificar única e exclusivamente os elementos das linhas i e j que definem as coordenadas dos planos que sofrem rotações.

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \times \begin{bmatrix} x_1 & \cdots & x_n \\ y_1 & \cdots & x_n \end{bmatrix} = \begin{bmatrix} x_1' & \cdots & x_{n+1}' \\ 0 & \cdots & y_{n+1}' \end{bmatrix} \Leftrightarrow \begin{matrix} x_i' = c \cdot x_i + s \cdot y_i \\ y_i' = -s \cdot x_i + c \cdot y_i \end{matrix}$$

$$\therefore \tag{4.4}$$

$$c = \frac{x_1}{\sqrt{x_1^2 + y_1^2}} \quad s = \frac{y_1}{\sqrt{x_1^2 + y_1^2}}$$

As rotações de Givens convencionais necessitam do cálculo de uma raiz quadrada e de 4 operações de multiplicação para cada par de elementos das linhas envolvidas no processo rotação. Conforme se pode notar, novos elementos ou elementos intermediários são formados a cada rotação de planos.

Uma seqüência de rotações pode ser coordenadamente aplicada sobre uma matriz para transformá-la numa matriz triangular superior. Neste caso, conforme o exemplo abaixo, a matriz ortogonal Q da transformação QR seria a transposta do resultado da multiplicação de todas as matrizes de rotação, ou seja:

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{G1(3,4,\theta)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{G2(2,3,\theta)} \begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{G3(3,4,\theta)}$$

$$\begin{bmatrix} \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{G4(1,2,\theta)} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{G5(2,3,\theta)} \begin{bmatrix} \times & \times & \times \\ 0 & \times & \times \\ 0 & 0 & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{G6(3,4,\theta)} R$$

$$G_6(3,4,\theta)\times G_5(2,3,\theta)\times G_4(1,2,\theta)\times G_3(3,4,\theta)\times G_2(2,3,\theta)\times G_1(3,4,\theta)\times A = R$$
$$se \quad A = QR, \quad então \quad Q^T A = R \quad e \quad Q = (G_6 \times G_5 \times G_4 \times G_3 \times G_2 \times G_1)^T \tag{4.5}$$

ROTAÇÕES DE GIVENS COM 3 MULTIPLICADORES

Visando simplificar o método de rotações através da eliminação do cálculo da raiz quadrada e reduzir o número de multiplicações nos demais elementos das linhas envolvidas no processo de rotação, Gentleman [64] propôs a seguinte formulação:

$$\begin{bmatrix} c & s \\ -s & c \end{bmatrix} \times \begin{bmatrix} \sqrt{d} & 0 \\ 0 & \sqrt{w} \end{bmatrix} \times \begin{bmatrix} u_1 & \dots & u_{n+1} \\ p_1 & \dots & p_{n+1} \end{bmatrix} = \begin{bmatrix} \sqrt{d'} & 0 \\ 0 & \sqrt{w'} \end{bmatrix} \times \begin{bmatrix} u'_1 & \dots & u'_{n+1} \\ 0 & \dots & p'_{n+1} \end{bmatrix}$$

$$onde \rightarrow \begin{matrix} x_i = \sqrt{d} \cdot u_i \\ y_i = \sqrt{w} \cdot p_i \end{matrix} \quad \begin{matrix} x'_i = \sqrt{d'} \cdot u'_i \\ y'_i = \sqrt{w'} \cdot p'_i \end{matrix} \quad e \quad c = \frac{\sqrt{d} \cdot u_1}{\sqrt{d \cdot u_1^2 + w \cdot p_1^2}} \quad s = \frac{\sqrt{w} \cdot p_1}{\sqrt{d \cdot u_1^2 + w \cdot p_1^2}} \tag{4.6}$$

Simplificando, tem-se:

$$u'_i = c \cdot u_i \cdot \sqrt{\frac{d}{d'}} + s \cdot p_i \cdot \sqrt{\frac{w}{d'}} \quad e \quad p'_i = -s \cdot u_i \cdot \sqrt{\frac{d}{w'}} + c \cdot p_i \cdot \sqrt{\frac{w}{w'}} \tag{4.7}$$

Impondo as seguintes considerações a (4.6),

$$(i) \quad \sqrt{d'} = \sqrt{d \cdot u_1^2 + w \cdot p_1^2} \quad (ii) \quad \sqrt{w'} = \frac{u_1 \sqrt{d \cdot w}}{\sqrt{d \cdot u_1^2 + w \cdot p_1^2}} \quad (iii) \quad u_1 = 1 \tag{4.8}$$

Obtém-se:

$$\begin{aligned} u_i^{'} &= c_m \cdot u_i + s_m \cdot p_i \\ p_i^{'} &= -p_1 \cdot u_i + p_i \end{aligned} \quad onde \rightarrow \quad c_m = \frac{d}{d + w \cdot p_1^2} \quad e \quad s_m = \frac{w \cdot p_1}{d + w \cdot p_1^2} \quad \therefore \quad u_1^{'} = 1 \quad (4.9)$$

Uma vantagem desta formulação, além das mencionadas acima, é que a matriz triangular superior resultante do processo de rotações possui a diagonal unitária, gerando economia computacional no processo de substituição inversa.

Esta metodologia de fatoração mostrou-se ser aderente ao problema de EESP [67], possuindo as seguintes vantagens em relação os métodos derivados da fatoração LU:

- Aplicação direta sobre a matriz de observação ponderada ou a matriz de observação aumentada (a qual considera o vetor independente como última coluna da matriz);
- Estabilidade aos problemas de mau condicionamento numérico gerados pela ponderação excessiva empregada na abordagem de EESP com restrições de igualdade;
- Possibilidade de acompanhar a evolução da soma ponderada dos quadrados dos resíduos (SPQR) durante a fatoração da matriz de observação aumentada, uma vez que o elemento adicional do vetor diagonal $\bar{d}$ acumula a contribuição da medida à SPQR após o processamento de todos os elementos da linha;

Mesmo com os avanços gerados por esta formulação, o algoritmo continuava demandando maior esforço computacional e, consequentemente, maior tempo de processamento do que métodos derivados da fatoração LU. Tal fato gerou, a priori, certa resistência por parte da comunidade de desenvolvedores e pesquisadores, entretanto tal problema foi minimizado com utilização de métodos para ordenação de colunas e linhas [68] que serão comentados adiante.

ROTAÇÕES DE GIVENS COM 2 MULTIPLICADORES

A primeira proposta das rotações de Givens sem raízes quadradas e com apenas dois multiplicadores [64] mostrou-se ser numericamente instável [65]. Três novas formulações foram propostas [65] para dar estabilidade ao método, entretanto apresentavam possibilidades de *underflow* nos fatores de escala *d* e *w* [68]. Um novo esquema com controle de *underflow* e *overflow* baseado supervisão dos fatores de escala e na escolha da matriz ortogonal a ser empregada na rotação foi proposto por [70], entretanto uma operação de comparação a cada rotação foi adicionada. Por fim, Vempati et ali [68] propôs uma variante deste esquema que simplificava o processo de escolha da matriz ortogonal e a forma de controle de *underflow* e *overflow* – conforme a formulação abaixo.

Considerando que $\sqrt{d'} = c \cdot \sqrt{d}$ e $\sqrt{w'} = c \cdot \sqrt{w}$ e reescrevendo (4.7), obtém-se a primeira opção do método conhecida como 'opção-c':

$$u_i' = u_i + \beta_c \cdot v_i \quad e \quad v_i' = -\alpha_c \cdot u_i + v_i \tag{4.10}$$

onde $\alpha_c = \frac{v_1}{u_1} \quad \beta_c = \alpha_c \cdot \frac{w}{d} \quad \gamma_c = \alpha_c \cdot \beta_c$

e $u_1' = u_1(1+\gamma_c) \quad d' = \frac{d}{1+\gamma_c} \quad w' = \frac{w}{1+\gamma_c}$

De forma análoga, considerando que $\sqrt{d'} = s \cdot \sqrt{w}$ e $\sqrt{w'} = s \cdot \sqrt{d}$ e reescrevendo (4.7), obtém-se a segunda opção do método conhecida como 'opção-s':

$$u_i' = \beta_s \cdot u_i + v_i \quad e \quad v_i' = -u_i + \alpha_s \cdot v_i \tag{4.11}$$

onde $\alpha_s = \frac{u_1}{v_1} \quad \beta_s = \alpha_s \cdot \frac{d}{w} \quad \gamma_s = \alpha_s \cdot \beta_s$

e $u_1' = v_1(1+\gamma_s) \quad d' = \frac{w}{1+\gamma_s} \quad w' = \frac{d}{1+\gamma_s}$

Durante o processo de rotações, a opção a ser aplicada será aquela que possuir o menor valor de γ. Ou seja, se $d \cdot u_1^2 \geq w \cdot v_1^2$, escolhe-se a 'opção-c', caso contrário, escolhe-se a 'opção-s'. Desta forma limita-se o crescimento dos fatores de escala *d* e *w*, uma vez que:

$$0 \leq \gamma = \min(\gamma_c, \gamma_s) \leq 1 \quad \therefore \quad 0{,}5 \leq \frac{1}{1+\gamma} \leq 1$$

Segundo os resultados obtidos em [68], a rotações de Givens com 2 multiplicadores possui velocidade de processamento superior quando comparada com a versão com 3 multiplicadores [64]. Entretanto, cabe ressaltar, que a matriz triangular superior resultante do processo de rotações não possui diagonal unitária e que o acompanhamento da SPQR durante a fatoração da matriz aumentada não é feito de forma direta.

PROCESSO DE FATORAÇÃO ORIENTADO POR COLUNA

O processo de fatoração orientado por coluna consiste na aplicação seqüencial e coordenada das rotações de Givens sobre uma determinada matriz $A \in \Re^{mxn}$, eliminado a cada rotação os elementos não nulos existentes abaixo do pivô escolhido da coluna sobre processamento. Portanto processa-se sequencialmente todas as colunas da matriz A até transformá-la no fator R da decomposição QR.

$$A = \begin{bmatrix} * & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ \times & \times & \times \end{bmatrix} \xrightarrow{Q1} A_1 = \begin{bmatrix} * & \times & \times \\ \times & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{Q2} A_2 = \begin{bmatrix} * & \times & \times \\ \times & \times & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{Q3} \quad A_3$$

$$A_3 = \begin{bmatrix} * & \times & \times \\ 0 & * & \times \\ 0 & \times & \times \\ 0 & \times & \times \end{bmatrix} \xrightarrow{Q4} A_4 = \begin{bmatrix} * & \times & \times \\ 0 & * & \times \\ 0 & \times & \times \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{Q5} A_5 = \begin{bmatrix} * & \times & \times \\ 0 & * & \times \\ 0 & 0 & * \\ 0 & 0 & \times \end{bmatrix} \xrightarrow{Q6} \quad A_6 = R$$

$$se \quad A \cdot x = b, \quad A = QR \quad e \quad Q = (Q_6 \times Q_5 \times Q_4 \times Q_3 \times Q_2 \times Q_1)^T, \quad tem-se$$
$$Q^T A \cdot x = Q^T b \quad \rightarrow \quad R \cdot x = Q^T b \tag{4.12}$$

Este processo necessita de um esforço adicional para escolher o elemento não nulo que será o pivô da coluna em processamento e para controlar as linhas já processadas. Em alguns casos é necessária a realização de permutas entre linhas e/ou colunas para garantir a seleção do pivô.

Um novo estudo comparativo [28] mostrou que a fatoração orientada por coluna utilizando rotações de Givens com 2 multiplicadores e métodos de ordenação de linhas (VPAIR) e de colunas (MDA) produzia uma quantidade de elementos intermediários inferior ao processo de fatoração orientado por linha. Tal fato indica, aparentemente, uma redução no tempo de processamento, desde que o algoritmo seja computacionalmente eficiente.

Considerando o sistema linear:

$$R \cdot A \cdot x = R \cdot b, \quad onde \quad \begin{cases} A \in \Re^{mxn}, & m \geq n \\ R \in \Re^{mxm}, & Ponderação \\ b \in \Re^{m}, & x \in \Re^{n} \end{cases} \tag{4.13}$$

O seguinte algoritmo pode ser aplicado para obter a fatoração da matriz aumentada:

1 Definir o vetor $w = \left\{ \sqrt{r_{11}}, \quad \sqrt{r_{22}}, \quad \ldots \quad \sqrt{r_{mm}} \right\}$.

2 Definir a matriz aumentada $U = \begin{bmatrix} A & b \end{bmatrix} \in \Re^{mxn+1}$.

3 Seja k = 1, faça:

3.1 Escolha o pivô da coluna k, realizando permutações caso seja necessário.

3.2 Seja l = índice de linha do pivô, faça:

3.2.1 Seja i = índice de linha do próximo elemento não nulo abaixo de l na coluna k.

3.2.2 Se $w_i = 0$, então:

3.2.2.1 Excluir toda a linha i.

3.2.3 Senão:

3.2.3.1 Se $w_l \cdot u_{lk}^2 \geq w_i \cdot u_{ik}^2$, então:

3.2.3.1.1 $\alpha = \frac{u_{ik}}{u_{lk}}, \quad \beta = \alpha \cdot \frac{w_i}{w_l}, \quad \gamma = \alpha \cdot \beta$

3.2.3.1.2 Seja j = k+1, faça:

3.2.3.1.2.1 Seja $aux = u_{lj}$.

3.2.3.1.2.2 $u_{lj} = u_{lj} + \beta \cdot u_{ij} \quad e \quad u_{ij} = -\alpha \cdot aux + u_{ij}$

3.2.3.1.2.3 Se j <= n+1 volte a (3.2.3.1.2.1), senão vá a (3.2.3.1.3).

3.2.3.1.3 $u_{lk} = u_{lk} \cdot (1+\gamma), \quad e \quad u_{ik} = 0$

3.2.3.1.4 $w_l = \frac{w_l}{(1+\gamma)} \quad e \quad w_i = \frac{w_i}{(1+\gamma)}$

3.2.3.2 Senão:

3.2.3.2.1 $\alpha = \frac{u_{lk}}{u_{ik}}, \quad \beta = \alpha \cdot \frac{w_l}{w_i}, \quad \gamma = \alpha \cdot \beta$

3.2.3.2.2 Seja j = k+1, faça:

3.2.3.2.2.1 Seja $aux = u_{lj}$.

3.2.3.2.2.2 $u_{lj} = \beta \cdot u_{lj} + u_{ij} \quad e \quad u_{ij} = -aux + \alpha \cdot u_{ij}$

3.2.3.2.2.3 Se j <= n+1 volte a (3.2.3.2.2), senão vá a (3.2.3.2.3).

3.2.3.2.3 $u_{lk} = u_{ik} \cdot (1+\gamma), \quad u_{ik} = 0, \quad e \quad aux = w_l$

3.2.3.2.4 $w_l = \frac{w_i}{(1+\gamma)} \quad e \quad w_i = \frac{aux}{(1+\gamma)}$

3.2.4 Escolher novo elemento não nulo abaixo de l na coluna k, se existir volte a (3.2.1), senão vá a (3.3).

3.3 k=k+1.

3.4 Se k < n+1 volte a (3.1), senão vá a (4).

4 Retornar U e w.

5 Fim.

PROCESSO DE FATORAÇÃO ORIENTADO POR LINHA

O processo de fatoração orientado por linha introduz uma modificação na estrutura da matriz $A \in \Re^{mxn}$ para possibilitar o processamento seqüencial de todos os elementos não nulos de suas linhas. Considere o seguinte sistema linear:

$$A \cdot x = b, \quad onde \quad A \in \Re^{mxn}, \quad x \in \Re^{n} \quad e \quad b \in \Re^{m}$$

A inserção de uma matriz triangular superior nula $U \in \Re^{nxn}$ e do vetor nulo $v \in \Re^{n}$ não acarretaria nenhuma mudança na solução do problema, assim pode-se escrever:

$$\begin{bmatrix} U \\ A \end{bmatrix} \cdot x = \begin{bmatrix} v \\ b \end{bmatrix} \quad ou \quad \begin{bmatrix} \underline{0} \\ A \end{bmatrix} \cdot x = \begin{bmatrix} \underline{0} \\ b \end{bmatrix} \tag{4.14}$$

Torna-se evidente que todos os elementos não nulos de A deverão sofrer rotações com as respectivas linhas de U. Entretanto tal processo pode ser feito linha por linha, eliminando-se sequencialmente todos os elementos não nulos de cada linha de A.

O benefício direto desta forma de processamento é a eliminação do esforço adicional gerado pela seleção e pelo controle do pivô, uma vez que a seleção da linha de U depende do índice do elemento não nulo da linha de A que está sendo rotacionada – conforme a Figura 56. O benefício indireto é a possibilidade de um acompanhamento mais apurado em problemas de mínimos quadrados ponderados ou qualquer problema estruturado por linha, uma vez que a contribuição de cada linha pode ser feita de forma incremental ao término de sua completa rotação.

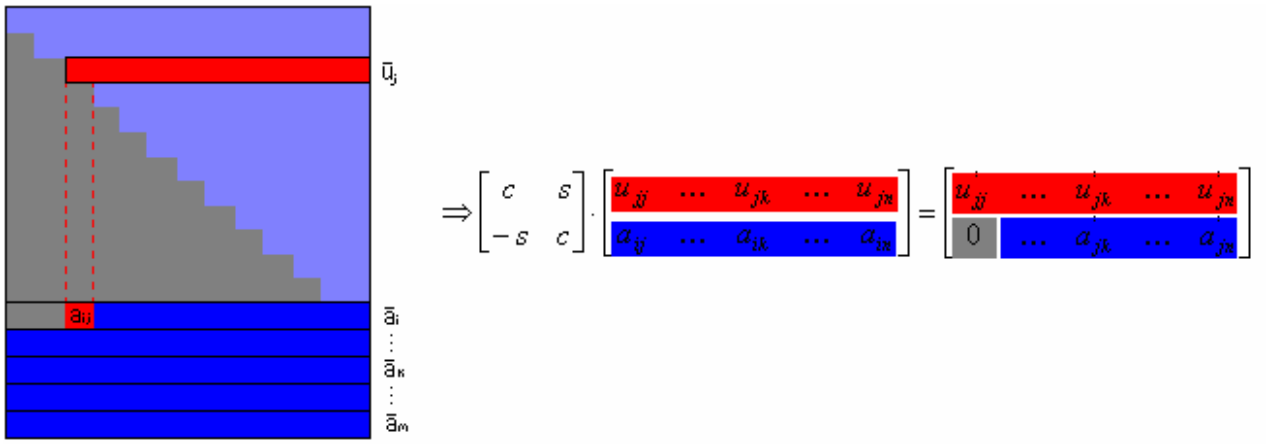


Figura 56 – Rotações orientadas por linha.

Visando atender as condições necessárias à aplicação das rotações de Givens com 3 multiplicadores e tornar a estrutura da matriz U mais aderente ao algoritmo de fatoração, pode-se reescrever (4.13) da seguinte maneira:

$$\begin{bmatrix} U \\ A \end{bmatrix} \cdot x = \begin{bmatrix} v \\ b \end{bmatrix} \quad ou \quad \begin{bmatrix} D^{1/2} & 0 \\ 0 & W^{1/2} \end{bmatrix} \begin{bmatrix} I \\ A_m \end{bmatrix} \cdot x = \begin{bmatrix} D^{1/2} & 0 \\ 0 & W^{1/2} \end{bmatrix} \begin{bmatrix} \underline{0} \\ b_m \end{bmatrix}$$

$$onde \quad D = \begin{bmatrix} d_1 & & \\ & \ddots & \\ & & d_n \end{bmatrix} \quad e \quad W = \begin{bmatrix} w_1 & & \\ & \ddots & \\ & & w_m \end{bmatrix} \tag{4.15}$$

Os elementos da matriz diagonal D representam o quadrado da ponderação das linhas da matriz U, os quais são inicialmente nulos, e os elementos da matriz W representam o quadrado da ponderação das linhas do sistema linear a ser solucionado. Desta forma o primeiro elemento da matriz triangular superior que representa o fator R de A será sempre unitário, conforme as considerações impostas em (4.8).

Considerando o sistema linear apresentado em (4.13), o seguinte algoritmo pode ser aplicado para realizar a fatoração da matriz aumentada:

1 Definir a matriz aumentada $V = \begin{bmatrix} A & b \end{bmatrix} \in \Re^{mxn+1}$

2 Definir a matriz triangular superior unitária $U = I \in \Re^{(n+1)x(n+1)}$

3 Definir o vetor de ponderação de V como $w = \left\{ \sqrt{r_{11}}, \quad \sqrt{r_{22}}, \quad \ldots \quad \sqrt{r_{mm}} \right\}$

4 Definir o vetor de ponderação de U como $d = \{0, \quad 0, \quad \ldots \quad 0\} \in \Re^n$

5 Seja k = 1. Faça:

5.1 Se $w_k > 0$, então:

5.1.1 Seja i = índice do primeiro elemento não nulo de v_k.

5.1.2 Seja $aux1 = d_i + w_k \cdot v_{ki}$.

5.1.3 Seja $c = \dfrac{d_i}{aux1} \quad e \quad s = w_k \cdot \dfrac{v_{ki}}{aux1}$.

5.1.4 Seja j = i+1. Faça:

5.1.4.1 Seja $aux2 = u_{ij}$.

5.1.4.2 $u_{ij} = c \cdot u_{ij} + s \cdot v_{kj} \quad e \quad v_{kj} = v_{kj} - v_{ki} \cdot aux2$.

5.1.4.3 j=j+1.

5.1.4.4 Enquanto $j \leq n+1$ volte a (5.1.4.1), senão vá a (5.1.5).

5.1.5 $v_{ki} = 0, \quad w_k = \dfrac{d_i}{aux1} \quad e \quad d_i = aux1$

5.1.6 Caso ainda exista algum elemento não-nulo em v_k, faça i = índice do próximo elemento não nulo e retorne a (5.1).

5.2 k = k +1

5.3 Se k > m vá a (6), senão retorne a (5.1).

6 Retorne U e d.

7 Fim.

FATORAÇÃO LDL^T

A fatoração LDL^T foi implementada dentro do escopo do pacote de ferramentas matemáticas por ser amplamente utilizadas por outras funções de SEE, tais como o fluxo de potência e a análise de contingências. Outro fator importante para a sua contemplação é a necessidade de se possuir uma referência temporal para os algoritmos de fatoração baseados na decomposição QR. Cabe ressaltar que o desenvolvimento de um algoritmo extremamente eficiente não era o objetivo do projeto. Caso haja interesse futuro em algoritmos com maior desempenho computacional, deve-se ter como referência os trabalhos [24] e [71].

A fatoração LU é uma variante do método de eliminação de Gauss que permite decompor uma matriz quadrada genérica em uma matriz triangular inferior e uma matriz triangular superior unitária. O diferencial da fatoração LU em relação à eliminação de Gauss é capacidade de se obter novas soluções para o mesmo sistema linear caso este sofra alterações em seu vetor independente.

Seja $A \in \Re^{nxn}$ uma matriz que se deseja triangulariar. As seguintes eliminações devem ser feitas para eliminar o elemento não-nulo a_{ik}:

$$L_{ik} \cdot D_{kk} \cdot \overline{A} = \tilde{A} \tag{4.16}$$

onde

$$L_{ik} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & \vdots & \ddots & & \\ & & -\overline{a}_{ik} & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix} \qquad D_{kk} = \begin{bmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & \frac{1}{\overline{a}_{kk}} & & & \\ & & & \ddots & & \\ & & & & 1 & \\ & & & & & \ddots \\ & & & & & & 1 \end{bmatrix}$$

$$\overline{A} = \begin{bmatrix} 1 & \times & \times & \times & \times & \times & \times & \times \\ & \ddots & \times & \times & \times & \times & \times & \times \\ & & \overline{a}_{kk} & \times & \times & \times & \times & \times \\ & & 0 & \ddots & \times & \times & \times & \times \\ & & \vdots & \times & \ddots & \times & \times & \times \\ & & \overline{a}_{ik} & \times & \times & \overline{a}_{ii} & \times & \times \\ & & \vdots & \vdots & \vdots & \vdots & \ddots & \times \\ & & \times & \times & \times & \times & \times & \overline{a}_{nn} \end{bmatrix} \qquad \tilde{A} = \begin{bmatrix} 1 & \times & \times & \times & \times & \times & \times & \times \\ & \ddots & \times & \times & \times & \times & \times & \times \\ & & 1 & \times & \times & \times & \times & \times \\ & & 0 & \ddots & \times & \times & \times & \times \\ & & \vdots & \times & \ddots & \times & \times & \times \\ & & 0 & \times & \times & \overline{a}_{ii} & \times & \times \\ & & \vdots & \vdots & \vdots & \vdots & \ddots & \times \\ & & \times & \times & \times & \times & \times & \overline{a}_{nn} \end{bmatrix}$$

A eliminação de todos os elementos nulos abaixo da diagonal da coluna k pode ser obtida através de eliminações sucessivas, ou seja, aplicando coordenadamente as transformações mostradas acima. Desta forma, a matriz resultante da referida eliminação – chamada de matriz de eliminação – será:

$$\left[L_{nk} \cdot L_{(n-1)k} \ldots L_{ik} \ldots L_{(k+1)k} \cdot D_{nn}\right] = \left[L_K\right] \tag{4.17}$$

onde

$$L_K = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \frac{1}{\overline{a}_{kk}} & & & & \\ & & \vdots & \ddots & & & \\ & & -\frac{\overline{a}_{ik}}{\overline{a}_{kk}} & & 1 & & \\ & & \vdots & & & \ddots & \\ & & -\frac{\overline{a}_{nk}}{\overline{a}_{kk}} & & & & 1 \end{bmatrix}$$

As demais colunas da matriz A podem ser eliminadas de forma análoga, resultando numa matriz triangular superior unitária. Assim:

$$\begin{gathered} L_N \cdot L_{N-1} \ldots L_I \ldots L_2 \cdot L_1 \cdot A = U, \quad \therefore \quad A = (L_1^{-1} \cdot L_2^{-1} \ldots L_I^{-1} \ldots L_{N-1}^{-1} \cdot L_N^{-1}) \cdot U \\ ou \quad A = L \cdot U \end{gathered} \tag{4.18}$$

Sendo a inversa de uma matriz triangular inferior também uma matriz triangular inferior e o produto de duas matrizes triangulares inferiores uma matriz triangular inferior, pode-se concluir

que a matriz $L = (L_1^{-1} \cdot L_2^{-1} \ldots L_I^{-1} \ldots L_{N-1}^{-1} \cdot L_N^{-1})$ também é triangular inferior. Também pode ser verificado que:

$$L_K^{-1} = \begin{bmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & \overline{a}_{kk} & & & & \\ & & \vdots & \ddots & & & \\ & & \overline{a}_{ik} & & 1 & & \\ & & \vdots & & & \ddots & \\ & & \overline{a}_{nk} & & & & 1 \end{bmatrix} \quad e \quad L = \begin{bmatrix} \overline{a}_{11}^{(1)} & & & & & & \\ \vdots & \ddots & & & & & \\ \overline{a}_{k1}^{(1)} & & \overline{a}_{kk}^{(k)} & & & & \\ \vdots & & \vdots & \ddots & & & \\ \overline{a}_{i1}^{(1)} & & \overline{a}_{nk}^{(k)} & & \overline{a}_{ii}^{(i)} & & \\ \vdots & & \vdots & & \vdots & \ddots & \\ \overline{a}_{n1}^{(1)} & & \overline{a}_{nk}^{(k)} & & \overline{a}_{ni}^{(i)} & & \overline{a}_{nn}^{(n)} \end{bmatrix}$$

A equação (4.18) também pode ser escrita da seguinte forma:

$$A = LDU \tag{4.19}$$

onde

$$L = \begin{bmatrix} 1 & & & & & \\ \vdots & \ddots & & & & \\ \dfrac{\overline{a}_{k1}^{(1)}}{\overline{a}_{11}^{(1)}} & & 1 & & & \\ \vdots & & \vdots & \ddots & & \\ \dfrac{\overline{a}_{i1}^{(1)}}{\overline{a}_{11}^{(1)}} & & \dfrac{\overline{a}_{nk}^{(k)}}{\overline{a}_{kk}^{(k)}} & & 1 & \\ \vdots & & \vdots & & \vdots & \ddots \\ \dfrac{\overline{a}_{n1}^{(1)}}{\overline{a}_{11}^{(1)}} & & \dfrac{\overline{a}_{nk}^{(k)}}{\overline{a}_{kk}^{(k)}} & & \dfrac{\overline{a}_{ni}^{(i)}}{\overline{a}_{ii}^{(i)}} & 1 \end{bmatrix} \quad e \quad D = \begin{bmatrix} \overline{a}_{11}^{(1)} & & & & & \\ & \ddots & & & & \\ & & \overline{a}_{kk}^{(k)} & & & \\ & & & \ddots & & \\ & & & & \overline{a}_{ii}^{(i)} & \\ & & & & & \ddots \\ & & & & & \overline{a}_{nn}^{(n)} \end{bmatrix}$$

A solução de um sistema linear após a decomposição LDU pode ser dada com uma substituição direta e outra substituição inversa, ou seja:

$$(i) \quad A \cdot x = b \rightarrow L \cdot D \cdot U \cdot x = b$$
$$(ii) \quad L \cdot y = b \quad \rightarrow \quad Substituição\ Direta$$
$$(iii) \quad D \cdot U \cdot x = y \quad \rightarrow \quad Substituição\ Inversa$$

Se a matriz A for simétrica e não singular, a matriz U será a transposta da matriz L [60]. Portanto a fatoração LDU se transformará em LDL^T, sendo aplicável a todos os sistemas lineares solucionados com a aplicação do método da equação normal de Gauss.

4.6.3.3 – Ordenação matricial

A produção de elementos intermediários é inerente ao processo de fatoração e independente da metodologia utilizada. Quanto maior for o número de elementos intermediários produzidos, maior será o esforço computacional necessário à completude do processo e, consequentemente, maior será o tempo de processamento. Portanto a minimização dos elementos intermediários criados durante o processo de fatoração é uma das principais características que uma ferramenta computacional matemática que aborde a solução de sistemas lineares deve possuir.

Uma das maneiras mais eficientes para diminuir a quantidade de elementos criados durante as transformações impostas sobre a matriz durante o processo de fatoração é a utilização de métodos de ordenação de linhas e colunas. Tais métodos tentam dispor as linhas e as colunas de forma a possibilitar que as projeções, transformações ou rotações do processo de fatoração envolvam, na medida do possível, somente os elementos já existentes nas linhas/colunas em cada operação elementar. No caso da fatoração utilizando rotações de Givens, deseja-se que as linhas envolvidas na rotação elementar possuam estruturas semelhantes e que a inevitável criação dos elementos intermediários seja postergada ao máximo para evitar a propagação destes elementos em futuras rotações. Tais características também são desejadas no processo de eliminação da fatoração LU e de suas variantes.

O presente trabalho visou implementar os métodos de ordenação que apresentaram os melhores resultados no estudo comparativo realizado por Vempati et alii [68], mas não se limitando somente àqueles. O método de ordenação de colunas A1 [73], conhecido como método de Gomes e Franquelo, uma variante do Minimum Degree Algorithm (MDA) [72], também foi abordado. Os métodos implementados no presente trabalho foram os:

- Métodos de ordenação de colunas: MDA [72] e A1 [73];
- Métodos de ordenação de linhas: R1 [74], R2, R3 [75], R4 e R5 [68].

MINIMUM DEGREE ALGORITHM – MDA

O método MDA de ordenação de colunas foi proposto inicialmente proposto por Tinney e Walker [72], denominado de algoritmo S2 e também conhecido como *Tinney II*. Rose [76] estudou o algoritmo S2 sob a ótica da teoria de grafos e desenvolveu um modelo teórico para este algoritmo, nomeando-o de Minimum Degree Algorithm.

Trata-se de um modelo simples que representa a estrutura de matriz através de um grafo equivalente, sendo cada coluna da matriz representada por um nó. As arestas do grafo são formadas através da combinação dos elementos de cada linha da matriz, onde cada par de

elementos não nulos representa a aresta que interliga os respectivos nós (colunas). Ressalta-se que, independentemente do número de pares existentes nas linhas da matriz, a interligação de dois nós deverá ser realizada através de uma única aresta. A Figura 57 abaixo exemplifica o referido grafo:

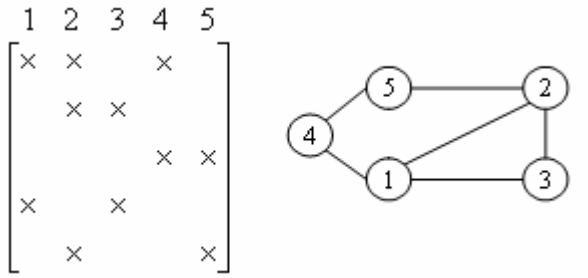


Figura 57 – Representação de uma matriz através do modelo de grafo do MDA

O grau de determinado nó é igual ao número de nós adjacentes ou ao seu número de arestas. O MDA objetiva-se em classificar os nós pela ordem crescente de seus respectivos graus, aplicando – para tanto – o seguinte algoritmo:

1. Dada uma matriz A, crie seu grafo conforme a ordem natural das colunas;
2. Crie um vetor vazio para gravar a ordem de seleção dos nós;
3. Selecione o nó que possui o menor grau dentro do grafo. Em caso de empate, escolha aquele que possua o menor índice;
4. Adicione o índice do nó selecionado ao vetor de ordenação;
5. Elimine o nó selecionado do grafo, criando novas arestas entre seus nós adjacentes;
6. Caso exista algum nó no grafo, volte ao passo 3. Senão finalize o processo.

Para a matriz exemplificativa mostrada na Figura 57, o algoritmo acima produziria o seguinte resultado:

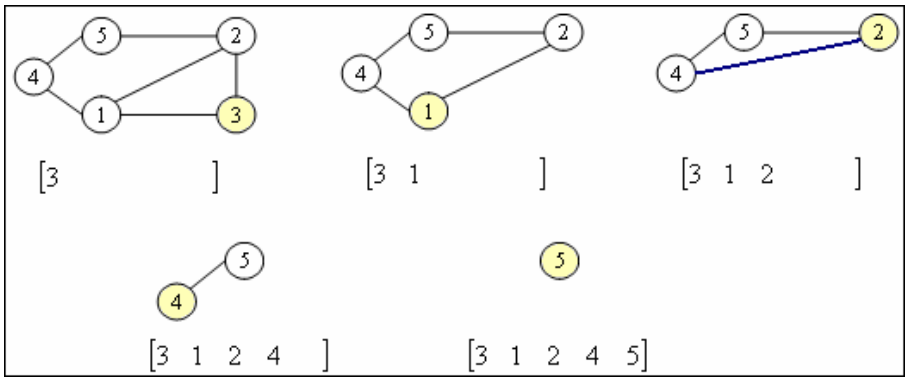


Figura 58 – Ilustração da seqüência de eliminação de nós do MDA

MÉTODO DE GOMES E FRANQUELO – A1

Gomes e Franquelo [73] propuseram uma pequena alteração no MDA para garantir que os nós que possuíssem o menor número de nós adjacentes eliminados tivessem maior prioridade durante o processo de seleção. Desta forma, aumentava-se a probabilidade de seleção de nós que possuíssem menor grau inicial. Para tanto, deveria ser adicionado um contador a cada nó para registrar o número de nós adjacentes que foram eliminados durante o processo.

O algoritmo proposto continuava igual ao MDA, exceto pela adição do critério de desempate pelo menor valor de nós adjacentes eliminados e pela adição das variáveis necessárias ao controle deste registro – conforme mostra o algoritmo abaixo:

1. Dada uma matriz A, crie seu grafo conforme a ordem natural das colunas;
2. Crie um vetor vazio para gravar a ordem de seleção dos nós;
3. Crie um vetor nulo para registrar o número de nós adjacentes eliminados de cada nó;
4. Selecione o nó que possui o menor grau dentro do grafo. Em caso de empate, escolha aquele que possua número de nós adjacentes eliminados. Mantendo-se o empate, escolha aquele que possua o menor índice;
5. Adicione o índice do nó selecionado ao vetor de ordenação;
6. Atualize o vetor que registra o número de nós eliminados para todos os nós adjacentes ao nó selecionado.
7. Elimine o nó selecionado do grafo, criando novas arestas entre seus nós adjacentes;
8. Caso exista algum nó no grafo, volte ao passo 4. Senão finalize o processo.

Para a matriz exemplificativa mostrada na Figura 57, o algoritmo acima produziria o seguinte resultado:

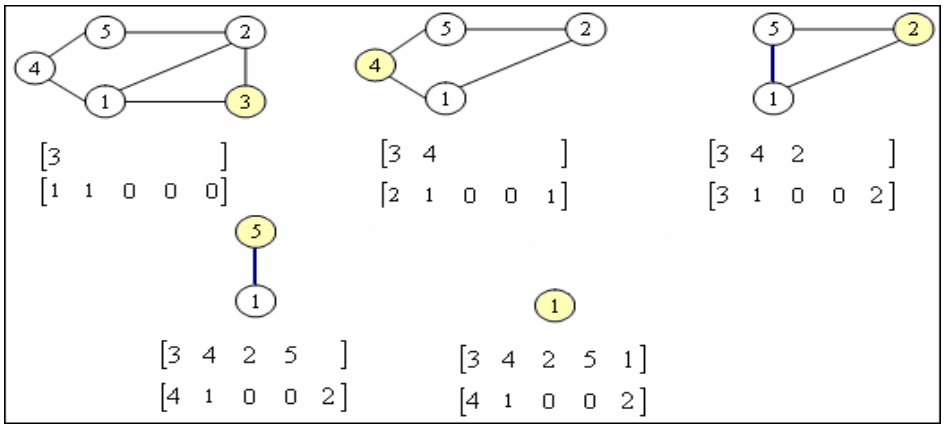


Figura 59 - Ilustração da seqüência de eliminação de nós do método de Gomes e Franquelo (A1)

MÉTODOS DE ORDENAÇÃO DE LINHAS

Os métodos de ordenação de linhas são aplicados prioritariamente aos sistemas lineares cujo processo de solução utilize métodos de fatoração direta, dentre os quais se destacam as rotações de Givens. Dentre os métodos de ordenação de linhas existentes na literatura técnica, optou-se pela implementação daqueles descritos e estudados por Vempati et alii [68]. Tais métodos são simples, rápidos e classificam as linhas da matriz através das seguintes heurísticas:

- **R1:** ordem crescente da quantidade de elementos não nulos;
- **R2:** ordem crescente da soma dos índices de coluna dos elementos não nulos;
- **R3:** ordem crescente do maior índice de coluna dos elementos não nulos;
- **R4:** ordem crescente do maior índice de coluna dos elementos não nulos com desempate através da menor quantidade de elementos não nulos;
- **R5:** ordem crescente do maior índice de coluna dos elementos não nulos com desempate através da menor soma dos índices de coluna dos elementos não nulos.

Cabe ressaltar que, independente do método utilizado para a ordenação de linhas, as medidas de tensão serão processadas antes que as demais medidas relacionadas a uma determinada barra do sistema, sejam estas injeções ou fluxos.

CONSIDERAÇÕES FINAIS SOBRE OS MÉTODOS DE ORDENAÇÃO

Além de reduzir o número de elementos intermediários criados durante o processo de fatoração e, consequentemente, manter a esparsidade da matriz triangular resultante, os métodos de ordenação devem ser rápidos e simples para não onerar demasiadamente o processo de solução. Cabe ressaltar que o principal objetivo destes métodos é reduzir o esforço computacional necessário à fatoração matricial e, por óbvio, o tempo de processamento.

Os métodos de ordenação de colunas visam tornar esparsa a matriz triangular resultante do processo de fatoração. Por sua vez, os métodos de ordenação de linhas visam evitar a criação de elementos intermediários na matriz sob fatoração, contribuindo significativamente para a redução do esforço computacional necessário a completude da fatoração.

Robey e Sulsky [77] propuseram um novo método de ordenação de linhas, conhecido como VPAIR, que se baseia na seleção das linhas que irão produzir o menor número de elementos intermediários a cada passo do processo da fatoração QR. Tal método pode ser aplicado antes ou durante o processo de rotação, sendo que sua versão “on-line” obteve os melhores resultados.

Um estudo recente [28] mostrou que a fatoração orientada por coluna – utilizando Givens com 2 multiplicadores – conjugada com *MDA* e o *VPAIR on-line* produzia uma quantidade menor de elementos intermediários quando comparado ao melhor método descrito em [68]. Os resultados também mostram uma significativa redução no tempo de execução do EE que utilizava a metodologia proposta quando comparado ao tempo de execução do EE que utilizava a melhor metodologia de [68].

A análise e algumas considerações nos resultados alcançados em [28] e reescritos em [29] serão feitas durante a apresentação dos resultados numéricos dos testes realizados na ferramenta computacional. Por hora, cabe considerar que a técnica apresentada em [77] não foi considerada computacionalmente viável durante os testes realizados em protótipos do pacote matemático desenvolvido neste trabalho.

4.6.3.4 – Métodos para a inversão matricial

A inversão matricial é muito utilizada durante a formulação e o desenvolvimento matemáticos de teorias e de métodos, entretanto – na prática – sua utilização é limitada e restrita a algumas funções e metodologias que inevitavelmente necessitam de seu cálculo completo ou parcial. O motivo deste repúdio é a exorbitante quantidade de operações matemáticas que envolvem o processo de inversão, bem como a necessidade de grandes quantidades de memória para armazenar os resultados obtidos.

Os métodos clássicos de detecção de erros grosseiros para EESP [78-84] necessitam do cálculo da matriz de sensilibidade como parte do processo de obtenção do vetor de resíduos normalizados – os quais serão apresentados nos próximos capítulos deste trabalho. Tal cálculo, além das operações matriciais básicas de multiplicação e subtração, necessita de uma operação de inversão matricial. Portanto, apesar de seu uso restrito, o pacote matemático apresentado neste trabalho implementou duas técnicas de inversão matricial para dar suporte à EESP. A primeira realiza a inversão matricial completa e a segunda, formulada por Broussolle [85], realiza uma inversão matricial esparsa voltada à detecção de erros grosseiros em EESP.

INVERSÃO MATRICIAL COMPLETA

Considere a seguinte matriz quadrada que se deseja inverter:

$$G = A^T A, \quad onde \quad \begin{cases} A = W^{-1/2} \cdot H \\ W \rightarrow Matriz\ Diagonal \\ G \in \Re^{nxn}, W \in \Re^{mxm}, A \in \Re^{nxm}\ e\ m \geq n \end{cases} \tag{4.19}$$

Sabendo que $G \cdot G^{-1} = I$ e considerando que $G^{-1} = \{x_1, x_2, \ldots, x_n\}$ e $I = \{e_1, e_2, \ldots, e_n\}$, onde x_i e e_i correspondem às inésimas colunas das matrizes G^{-1} *e* I respectivamente. Pode-se obter G^{-1} através da solução seqüencial dos seguintes sistemas lineares:

$$G \cdot x_i = e_i, \ onde \ \ i = 1..n \tag{4.20}$$

Para solucionar os sistemas lineares acima pode ser utilizada tanto a fatoração LDL^T de G quanto a fatoração QR de A, desde a matriz A tenha *rank* igual a *n.* Para mostrar esta possibilidade, pode-se formular o seguinte teorema através dos teoremas apresentados em [60] e [61]:

TEOREMA 4.1: *Se* $A \in \Re^{nxm}$ *possui rank igual a n e se o fator R de sua decomposição QR possuir elementos diagonais positivos, então o fator R da decomposição QR de A será igual tanto ao fator de Cholesky de* $A^T A$ *quanto a matriz* $D^{1/2} L^T$ *da fatoração LDL^T.*

PROVA: *Se rank(A) = n, então tanto o fator de Cholesky quanto a matriz L da fatoração LDL^T serão únicos. Logo, tem-se:*

$$G = A^T \cdot A = L \cdot D \cdot L^T = (L \cdot D^{1/2}) \cdot (D^{1/2} \cdot L^T) \rightarrow Fatoração \ LDL^T$$
$$G = A^T \cdot A = R^T \cdot R \rightarrow onde \ R \ é \ o \ \ fator \ de \ Cholesky \ de \ G$$
$$G = A^T \cdot A = (R^T \quad 0) \cdot Q^T \cdot Q \cdot \begin{pmatrix} R \\ 0 \end{pmatrix} = R^T \cdot R \rightarrow onde \ R \ é \ o \ \ fator \ da \ decomposição \ QR$$

Desta forma as colunas da matriz G^{-1} podem ser obtidas sequencialmente através de uma fatoração inicial de *G* ou *A* e do uso de substituições diretas e inversas, conforme o processo de solução através da fatoração LU.

INVERSÃO MATRICIAL ESPARSA DE BROUSSOLLE

O mérito da formulação da inversão matricial esparsa ora apresentada deve ser atribuído aos autores Takahashi, Fagan e Chen [86] que a aplicaram inicialmente ao cálculo de correntes de curto circuito. Broussole [85] adaptou tal técnica à EESP, mais especificamente ao cálculo do vetor de resíduos normalizados.

Considere a matriz apresentada em (4.19). Pode-se reescrevê-la da seguinte maneira:

$$\begin{cases} G = LDL^T \\ G \cdot G^{-1} = I \end{cases} \rightarrow \quad LDL^T \cdot G^{-1} = I \quad ou \quad L^T \cdot G^{-1} = D^{-1} L^{-1} \tag{4.21}$$

Sendo $Z = G^{-1}$ e $T = I - G^{-1} = I - Z$, tem-se:

$$Z = D^{-1} L^{-1} + T \cdot Z \tag{4.22}$$

As seguintes observações devem ser feitas:

- A matriz *L* é triangular inferior unitária, portanto sua inversa possui as mesmas características. Logo a diagonal de $D^{-1}L^{-1}$ é igual a D^{-1};
- A matriz *Z* é simétrica uma vez que *G* é simétrica. Assim somente os elementos da parte diagonal superior de *Z* precisam ser calculados, ou seja, os elementos z_{ij} tais que $j \geq i$.

Tais considerações resultam no não uso da matriz L^{-1} para fins do cálculo da parte triangular superior da matriz *Z,* denominada abaixo como Z_S. Assim, tem-se:

$$Z_S = D^{-1} + T \cdot Z \tag{4.23}$$

O calculo dos elementos não nulos de determinada linha k de Z_S dependem somente dos elementos não nulos das linhas subseqüentes, devido à forma peculiar de T. Assim Z_S deve ser calculado a partir de sua última linha, conforme as equações abaixo.

$$z_{ji} = \sum_{k=i+1}^{n} t_{ki} \cdot z_{jk} \quad e \quad z_{ii} = d_{ii} + \sum_{k=i+1}^{n} t_{ki} \cdot z_{jk} \tag{4.24}$$

A matriz *Z* resultante do cálculo acima será esparsa em virtude da esparsidade da matriz T e possuirá somente os elementos básicos da matriz inversa completa. Cabe ressaltar que a matriz Z terá todos os elementos não nulos ocupando as mesmas posições que os elementos não nulos de G, ou seja, as matrizes possuem estruturas iguais.

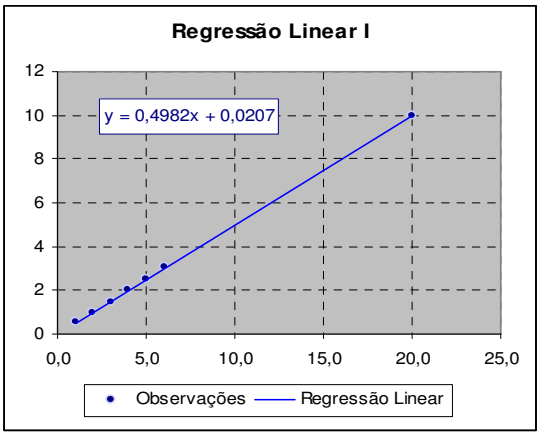
4.6.3.4 – Pontos de alavancamento

Os pontos de alavancamento ou, em inglês *leverage points*, foram assim designados por exercerem enorme influência nos resultados de regressões ou quaisquer outros problemas de minimização de resíduos, agindo de forma semelhante a uma alavanca. Tal característica se deve ao posicionamento discrepante destes pontos quando comparados aos demais pontos do espaço do problema. Tais pontos foram inicialmente tratados em análise de estatística robusta de regressão [87-89] e posteriormente foram introduzidos em EESP por Milli et alli [90-91].

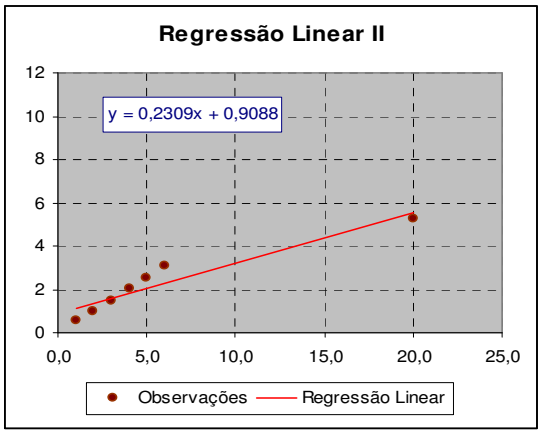
A regressão linear é um exemplo simples e didático para introduzir o conceito de pontos de alavancamento e mostrar seus efeitos. Para tanto considere as séries de observações mostrados na Tabela 1. Note que a 7ª observação possui um distanciamento horizontal elevado quando comparado com os demais pontos.

Tabela 1: Dados e resultados das regressões lineares exemplificativas.

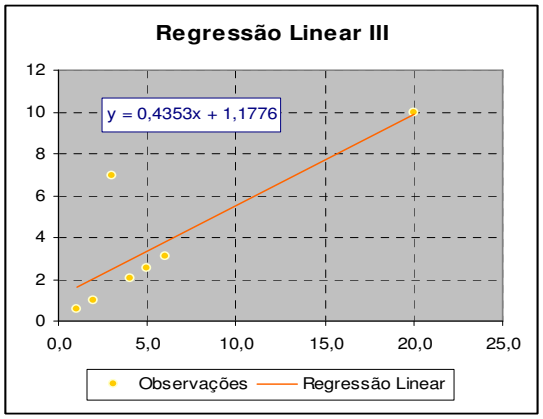
	Regressão I			Regressão II			Regressão III		
	Observações		Estimativas	Observações		Estimativas	Observações		Estimativas
No.	x	y	ÿ	x	y	ÿ	x	Y	ÿ
1ª	1,0	0,54	0,52	1,0	0,54	1,14	1,0	0,54	1,61
2ª	2,0	0,98	1,02	2,0	0,98	1,37	2,0	0,98	2,05
3ª	3,0	1,45	1,52	3,0	1,45	1,60	3,0	6,97	2,48
4ª	4,0	2,03	2,01	4,0	2,03	1,83	4,0	2,03	2,92
5ª	5,0	2,52	2,51	5,0	2,52	2,06	5,0	2,52	3,35
6ª	6,0	3,08	3,01	6,0	3,08	2,29	6,0	3,08	3,79
7ª	20,0	9,97	9,98	20,0	5,23	5,53	20,0	9,97	9,88



(a) Série com pequenos erros aleatórios



(b) Erro grosseiro na última observação



(c) Erro grosseiro na terceira observação

Figura 60 - Resultados das regressões lineares exemplificativas.

As observações da primeira série possuem pequenos erros aleatórios e o resultado de sua regressão pode ser visto no item (a) da Figura 60. A segunda série, por sua vez, além dos pequenos erros aleatórios, possui um erro grosseiro na 7ª observação. O item (b) da Figura 60 mostra o impacto deste erro no resultado da regressão. Por fim, a terceira série possui um erro grosseiro na 3ª observação, mantendo os pequenos erros aleatórios nas demais observações.

O item (c) da Figura 60 mostra que, apesar da magnitude do erro ser compatível com àquela imposta na 7ª observação da segunda série, o resultado não sofreu impactos maiores.

Pequenas variações nos estados da função que se está minimizando podem acarretar grandes desvios (resíduos) nas proximidades do ponto de alavancamento. Esta propriedade torna o ponto de alavancamento mais influente que os demais pontos dentro do processo de minimização dos resíduos, tal influência pode ser tanto benéfica quanto maléfica para a solução do problema. Caso não existam erros grosseiros no conjunto de medidas ou observações consideradas como pontos de alavancamento, a solução do problema tenderá ao seu valor real – mesmo que existam erros de magnitude significativa nas outras medidas/observações. Isto se deve ao fato que as mudanças nos estados geradas por estes erros produzirão – junto aos pontos de alavancamento – resíduos maiores que o próprio resíduo das medidas contaminadas com erro grosseiro. Por outro lado, erros nas medidas consideradas como pontos de alavancamento tenderam a ser não detectáveis e produzirão grandes distorções no resultado do problema. Portanto os problemas que envolvem minimização de resíduos devem ser capazes de identificar e tratar adequadamente tais pontos.

Para identificar os pontos discrepantes dentro da *nuvem* de pontos que formam o espaço do problema, é necessário determinar as distâncias de cada ponto em relação aos demais pontos da *nuvem*, padronizar estas distâncias em relação a melhor distância representativa do conjunto (distância média ou mediana) e comparar os valores resultantes a um índice estatístico de dispersão.

A literatura propõe algumas técnicas de identificação e tratamento de pontos de alavancamento [87-91], entretanto este trabalho irá se restringir ao método proposto por Milli et alli [91] e adotado no trabalho de Pires [55] por ser computacional e estaticamente eficiente e apresentar resultados satisfatórios. Em [91] alguns métodos de identificação baseados em estatística não robusta foram descartados por não serem capazes de lidar adequadamente com a existência de múltiplos pontos de alavancamento. Métodos baseados em estatística de projeção foram, então, propostos para lidar com o referido problema e adaptados para as condições peculiares da EESP.

DEFINIÇÃO DO ESPAÇO FATOR E DE SEUS PONTOS

A nuvem de pontos do espaço fator (espaço do problema) é definida pelas linhas do sistema linear resultante da formulação matemática do problema de minimização, ou seja, cada linha deste sistema representa um ponto no espaço fator de n dimensões (onde n é o número de estados do problema). No caso específico de EESP, os pontos são definidos pelas linhas da matriz Jacobiana ponderada pelos desvios-padrão das respectivas medidas ($R^{-1/2} \cdot H$).

ÍNDICE DE PROJEÇÃO ESTATÍSTICA

A identificação do ponto de alavancamento é realizada através de seu índice de projeção estatística. Defini-se tal índice como sendo a maior distância resultante da projeção do segmento de reta que une o i-nésimo ponto à mediana do conjunto de pontos sobre todos os segmentos de reta igualmente obtidos para os outros pontos [55]. Algumas peculiaridades dos sistemas elétricos de potência impedem a aplicação do algoritmo de projeção – dos quais resultaram os índices definidos acima – da forma como foi formulado, resultando em algumas modificações simplificadoras [91]. Não obstante a definição acima, o índice de projeção estatística que considera as condições impostas pelo modelo de regressão em sistemas de potência é dado pela seguinte equação:

$$PS_i = \max_{\underline{v}} \frac{\left|\underline{h}_i^T \cdot \underline{v}\right|}{Es_{\underline{v}}} = \max_{j=1..m} \frac{\left|\underline{h}_i \cdot \underline{h}_j^T\right|}{Es_{hj}} \tag{4.25}$$

onde $\underline{h}$ é o vetor linha da matriz Jacobiana $R^{-1/2} \cdot H$, $\underline{v} = \underline{h}_1 \ldots \underline{h}_m$ e Es_{hj} é o estimador de escala MAD (*Median Absolute Deviation*) da j-nésima linha (ponto ou medida). O conceito de estimador de escala será apresentado a seguir. Cabe ressaltar, anteriormente, que o esforço computacional para obtenção de $R^{-1/2} \cdot H \cdot H^T \cdot R^{-1/2}$ é relativamente baixo devido à esparsidade da matriz jacobiana. Por fim, pode-se notar que o índice de projeção estatística do i-nésimo ponto (linha) é dado pela máxima distância padronizada resultante da projeção do segmento de reta que une a origem do espaço fator ao i-nésimo ponto (linha) sobre os demais segmentos igualmente obtidos para os outros pontos (linhas).

ESTIMADORES DE ESCALA

Os estimadores de escalas são utilizados para possibilitar que um determinado grupo de pontos dentro do espaço fator possua uma mesma base de comparação, permitindo – desta forma – determinar quais pontos são realmente discrepantes em relação aos demais. Portanto as distâncias resultantes da projeção estatística comentada acima devem ser padronizadas pelo estimador de escala do respectivo grupo de pontos. O estimador de escala mais conhecido e utilizado é a variância (σ^2), entretanto seu desvio frente à presença de valores discrepantes torna desaconselhável sua aplicação neste caso. Análises anteriores [55] [91] indicam que é suficiente utilizar o estimador de escala MAD (*Median Absolute Deviation* ou Desvio Absoluto da Mediana) devido a sua remota possibilidade de sofrer influência de medidas discrepantes. O MAD de um determinado ponto (linha ou medida) j é dado por [55]:

$$MAD_j = b_k \cdot 1.4826 \cdot mediana\left|\underline{h}_j \cdot \underline{h}_i^T\right|,\ onde\ \ 1 \le i \le m \qquad (4.26)$$

onde $\underline{h}$ é o vetor linha da matriz Jacobiana $R^{-1/2} \cdot H$ e b_K é um fator de correção para pequenos conjuntos de projeções válidas – definidas para EESP como todas aquelas que $\left|\underline{h}_j \cdot \underline{h}_i^T\right|$ > 0. Os valores deste fator podem ser obtidos da Tabela 2 abaixo:

Tabela 2 – Fatores de correção do estimador de escala MAD.

K	2	3	4	5	6	7	8	9	> 9
b_k	1,196	1,495	1,363	1,206	1,200	1,140	1,129	1,107	$\frac{k}{k-0,8}$

IDENTIFICAÇÃO E TRATAMENTO DOS PONTOS DE ALAVANCAMENTO [91][90][55]

A identificação de medidas caracterizadas como pontos de alavancamento é realizada através da comparação do valor do índice de estatística de projeção de determinada linha (ponto ou medida) com seu respectivo valor de limiar. O valor de limiar de cada linha (ponto ou medida) é obtido através da distribuição *Qui-quadrada* ($\chi^2_{v,97,5\%}$) com percentil de 97.5% [91], onde o número de graus de liberdade (*v*) da referida distribuição é igual ao número de elementos não-nulos da respectiva linha. Analiticamente, uma linha (ponto ou medida) é considerada um ponto de alavancamento quando satisfizer a seguinte condição:

$$PS_i > b_i = \chi^2_{v,97,5\%} \qquad (4.27)$$

A incorporação de medidas caracterizadas como pontos de alavancamento no problema de estimação de estados em sistemas de potência é feita atribuindo-se um peso adicional às medidas assim classificadas, além daqueles referentes aos desvios-padrão das medidas. O cálculo dos fatores de pesos adicionais para levar em conta possíveis pontos de alavancamento é feito a partir da seguinte expressão [91]:

$$w_i^{PA} = \min\left\{1, \left(\frac{b_i}{PS_i}\right)^2\right\},\ onde\ i = 1..m \qquad (4.28)$$

4.6.3.4 – Desenvolvimento e implementação do pacote computacional

Durante o desenvolvimento dos protótipos das classes vetoriais e matriciais esparsas e a realização dos testes e da validação dos métodos e das teorias matemáticos descritos nas subseções anteriores, todas as funções foram inicialmente implementadas dentro da própria estrutura das respectivas classes. Apesar de muitas destas funções serem inerentes ao conceito

da matriz, tais como a fatoração ou a inversão, havia a preocupação de se agrupar as funções em classes específicas e voltadas a um determinado grupo de competências. Suas estruturas estavam ficando cada vez mais complexas e muitas das funções implementadas necessitavam de controles externos que exorbitavam o limite conceitual da classe.

Os protótipos desenvolvidos também tinham como objetivo possibilitar uma comparação das classes implementadas, criando situações realistas das dificuldades que seriam encontradas durante o uso da ferramenta. Assim a ênfase no desenvolvimento seria dada à classe que apresentasse a melhor relação entre a eficiência, aplicabilidade e flexibilidade. Desta forma, as classes específicas e voltadas a um determinado grupo de competências seriam desenvolvidas prioritariamente para a classe matricial escolhida e, em segundo plano, para algumas funções da outra classe matricial. Portanto parte das funções que serão apresentadas a seguir ainda se encontra na estrutura da classe matricial secundária, a qual não será objeto de detalhamento deste trabalho.

A ESCOLHA DA CLASSE MATRICIAL PRINCIPAL

Apesar da classe *MatES* apresentar um desempenho 10% superior no processo de fatoração utilizando as rotação de Givens orientadas por linha, um desempenho 5% superior no processo de ordenação tanto por linhas quanto por colunas e maior facilidade nos procedimentos de cópia, a classe *Matriz* foi escolhida pelos seguintes motivos:

- Tanto o desempenho no processo de fatoração utilizando rotações de Givens orientadas por linha quanto o desempenho no processo de ordenação foram considerados satisfatórios, sendo a mora adicional atribuída ao procedimento de ligação dos ponteiros das colunas nas duas funções. As funções implementadas em ambas as classes são semelhantes e o uso procedimentos recursivos na classe *Matriz* se mostrou extremamente eficiente;

- As funções que necessitam de orientação por coluna, apesar de não ser o escopo principal da ferramenta, são mais facilmente implementadas na classe *Matriz*, apresentando uma eficiência muito superior. Cabe ressaltar que implementações futuras podem necessitar desta facilidade e que a evolução da ferramenta é um fator a ser considerado;

- A estrutura de armazenamento classe *MatES* baseada em listas segmentadas torna o acesso aos elementos da matriz através da passagem de referências diretas à memória dependente da estrutura interna e susceptível a erros frente a alterações nos dados, praticamente impossibilitando esta prática. Ressalta-se que essa técnica pode gerar grande economia computacional em problemas de solução iterativa que necessitam de alterações nos valores dos elementos da matriz dos coeficientes a cada iteração, tais como alguns métodos de EESP.

Os motivos listados acima não são exaustivos, mas foram relevantes e decisivos para a escolha da classe Matriz.

SISTEMAS LINEARES

A classe *SistemaLinear* foi desenvolvida com o intuito de agrupar as funções relacionadas à solução de sistemas lineares, possibilitando o emprego de diversas técnicas de fatoração, a utilização de diferentes métodos de solução, o armazenamento e a análise das informações resultantes do processo de solução e, por fim, a preservação da matriz e dos vetores que formam o sistema. A figura abaixo mostra o diagrama da referida classe.

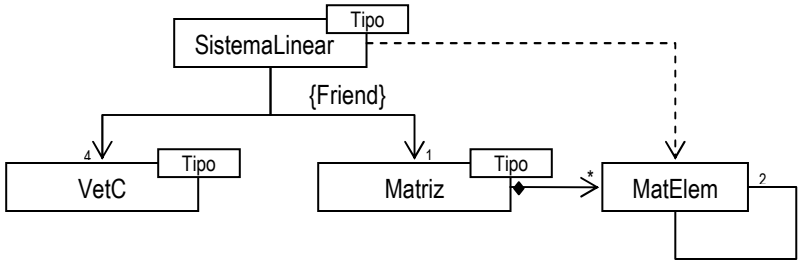


Figura 61 - Relacionamentos da classe *SistemaLinear*.

Internamente a classe *SistemaLinear* possui propriedades para armazenar a matriz triangular resultante do processo de fatoração (FMatFat); o vetor independente após as rotações (FVetBFat); o vetor de estados (FVetX); o vetor responsável pela diagonalização da matriz fatorada (FVetDiag); um vetor que indica os estados com pivôs nulos (FVetZeroDiag); uma lista para armazenar, caso solicitado, as rotações do processo de fatoração (FRot); variáveis que indicam qual o método de fatoração e solução a ser aplicado (FFatMetodo e FMetodo); uma variável para armazenar a SPQR quando possível e solicitado (FSPQR); além de flags de controle. A Figura 62 mostra a estrutura interna dos dados.

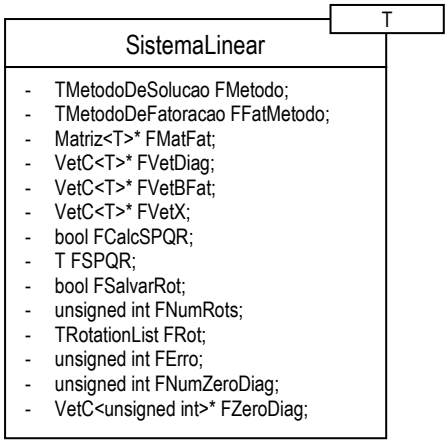


Figura 62 – Propriedades internas da classe SistemaLinear.

O sistema linear pode ser solucionado diretamente (SOL_JACOB), através da fatoração QR, ou pelo método da equação normal de Gauss (SOL_GANHO). Para tanto basta informar o método de solução a ser empregado através da função *SetMetSolucao(Metodo)*. Cabe ressaltar que o processo de solução utilizará a fatoração indicada em FFatMetodo, caso possível e independentemente de ser a forma mais eficiente. Os seguintes métodos podem ser empregados na fatoração e, consequentemente, na solução do sistema linear:

- Rotações de Givens com 3 multiplicadores e orientada por linha (FAT_GIVENS_3M);
- Rotações de Givens com 2 multiplicadores e orientada por linha (FAT_GIVENS_2M_ROP);
- Rotações de Givens com 2 multiplicadores e orientada por coluna (FAT_GIVENS_2M_COP);
- Fatoração LDL^T (FAT_LDU);

A fatoração pode ser feita através da função pública *Fatorar()*, passando-se como parâmetro a matriz com ou sem o vetor de ponderação de linhas. Esta função irá analisar a possibilidade de fatoração pelo método indicado em FFatMetodo e irá chamar a correspondente função interna. A Figura 63 mostra as funções envolvidas neste processo.

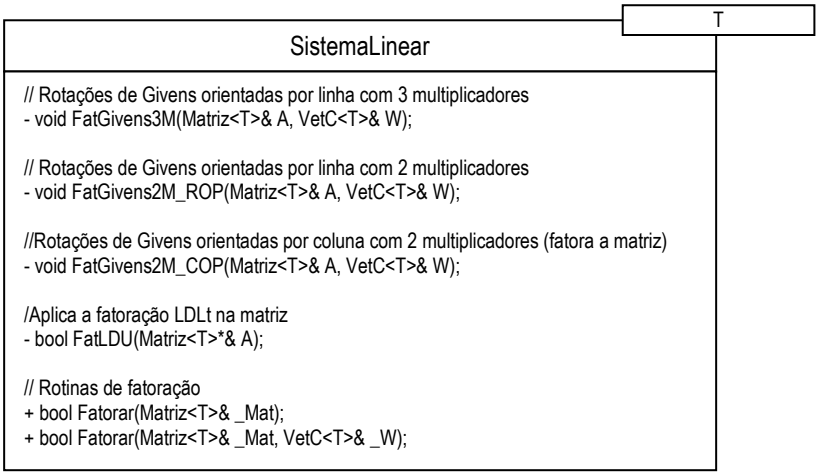


Figura 63 – Função envolvidas no processo de fatoração

A solução do sistema linear, utilizando os métodos de fatoração descritos acima, é realizada através da função pública *Solucionar()*, passando-se como parâmetro a matriz de coeficientes, o vetor de ponderação das linhas e o vetor independente. Salienta-se que nenhuma alteração ocorrerá nos parâmetros, ou seja, o sistema linear informado continuará o mesmo após o processamento. A solução será dada através dos seguintes métodos:

- Aplicação direta da fatoração QR (SOL_JACOB) sobre sistema linear $W^{-1/2} \cdot A \cdot x = W^{-1/2} \cdot b$, sendo válido para todos os métodos de fatoração, exceto a fatoração LDL^T;

- Aplicação da Equação Normal de Gauss (SOL_GANHO) sobre o sistema de equações, solucionando $A^T \cdot W \cdot A \cdot x = A^T \cdot W \cdot b$. Sendo válido para todos os métodos de fatoração.

A classe também possibilita, através da função *AtualizarSolucao()*, a resolução do sistema sem a necessidade de nova fatoração. Será verificada tanto a existência de uma solução pretérita quando o armazenamento das rotações nos casos de utilização de métodos de fatoração QR. Caso não seja possível solucionar aplicando o conjunto de rotações sobre o novo vetor independente, a nova solução – independente do método de fatoração utilizado inicialmente – será dada através do processo de substituição direta e inversa, conforme o processo usual da fatoração LDL^T. Para tanto a função necessitará do mesmo conjunto de parâmetros utilizados na solução inicial (matriz de coeficientes, vetor de ponderação e vetor independente). A Figura 64 mostra o conjunto de procedimentos e funções envolvidos neste processo.

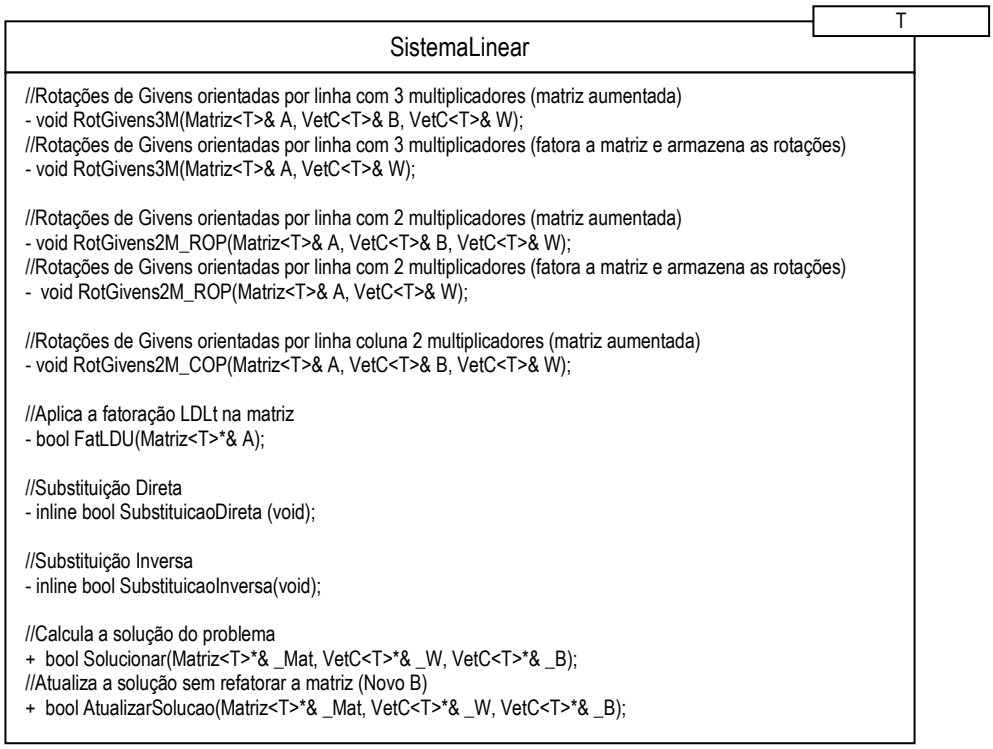


Figura 64 – Funções envolvidas no processo de solução e atualização da solução da classe *SistemaLinear*

Os métodos para inversão matricial também foram inseridos no contexto da classe *SistemaLinear* por necessitarem das funções de fatoração e/ou de substituição (direta e inversa) implementadas nesta classe. Diferentemente dos procedimentos e das funções de fatoração e solução de sistemas lineares, as funções de inversão retornam seu resultado através da matriz passada como parâmetro. Cabe esclarecer que a matriz passada como parâmetro forma a

matriz a ser invertida de tal forma que, sendo *A* o parâmetro da função, a matriz a ser invertida será $G = A^T A$. Caso o valor deste parâmetro seja nulo e exista o resultado de uma fatoração armazenada em FMatFat, as funções de inversão entenderão que o resultado da fatoração da matriz a ser invertida se encontra disponível e continuarão o processo de inversão a partir deste ponto. Para inverter a matriz utilizando a método esparso, deve-se utilizar a função *InverterGanho()*. Caso se deseje inverter pelo método do vetor unitário (inversão completa), deve-se utilizar a função *InverterGanho2()*, conforme a Figura 65.

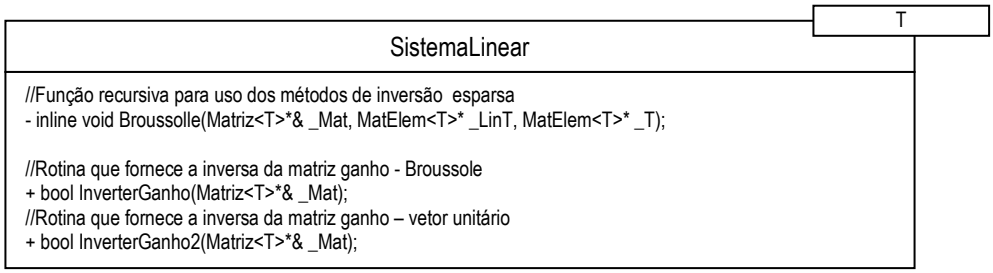


Figura 65 – Funções da classe *SistemaLinear* envolvidas no processo de inversão matricial.

ORDENAÇÃO MATRICIAL

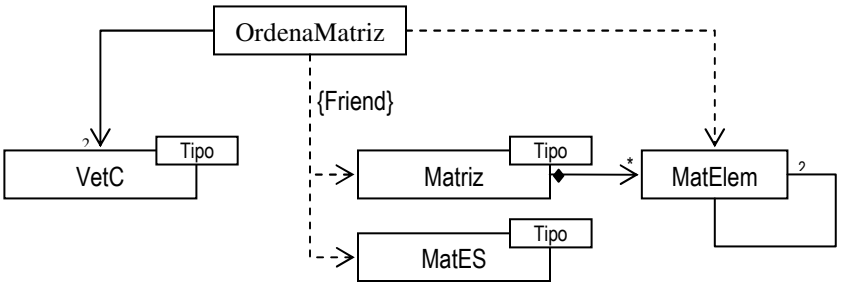


Figura 66 – Relacionamentos da classe *OrdenaMatriz*.

A diversidade de métodos de ordenação de linhas e de colunas abordados neste trabalho e suas diferentes formas de aplicação resultaram no desenvolvimento da classe *OrdenaMatriz*. Tal classe tem o objetivo de agrupar os métodos de ordenação existentes, possibilitando diferentes formas de aplicação da ordenação matricial e o armazenamento adequado dos resultados do processamento.

A classe *OrdenaMatriz* possui dois vetores para armazenar o resultado da ordenação tanto das linhas quanto das colunas (FVetOrdLin e FVetOrdCol); duas variáveis de tipo enumerado para indicar os métodos a serem aplicados (FMetLin e FMetCol); e propriedades lógicas para informar se a ordenação ocorrerá numa matriz genéricas ou numa matriz simétrica (FJacob), se os métodos indicados deverão ser aplicados nas linhas e/ou nas colunas (FOrdenarLin e

FOrdenarCol) e se após o processamento a ordenação resultante deverá ser aplica à matriz informada como parâmetro. A Figura 67 mostra todas as propriedades internas desta classe.

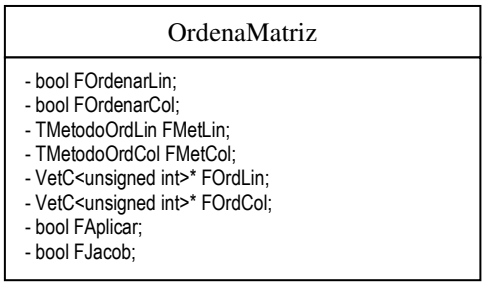


Figura 67 – Propriedades internas da classe *OrdenaMatriz*.

A classe *OrdenaMatriz* pode lidar com os todos os métodos de ordenação descritos na subseção 4.6.3.3, sendo tais métodos identificados pelos seguintes códigos:

- **MD:** Método de ordenação de colunas MDA (Minimum Degree Algorithm) [76];
- **A1:** Método de ordenação de colunas proposto Gomes e Franquelo A-1 [73];
- **R1, R2, R3, R4, R5:** Métodos de ordenação de linhas descritos por Vempati et alli [68].

O processo de ordenação de colunas faz uso de estruturas de dados que trabalham com uma lista de relacionamento dos nós ao invés do modelo propriamente dito do grafo apresentado por Rose [76]. Tais estruturas proporcionam tanto a implementação do MDA quanto do método A1. Apesar de algumas implementações [29] criarem classes específicas para representar tal grafo, os resultados durante as fases de teste dos protótipos indicaram que o uso desta estrutura era produzia resultados mais satisfatórios.

Caso a matriz a ser ordenada for indicada – através da função *Jacob()* – como sendo simétrica, a ordenação das linhas através dos métodos descritos acima será vedada. Neste caso a ordem obtida na ordenação das colunas será aplicada também nas linhas para manter a simetria da matriz.

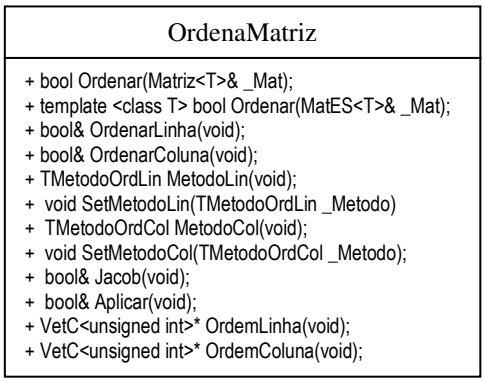


Figura 68 – Funções públicas da classe OrdenaMatriz.

A ordenação pode ser aplicada tanto na classe *Matriz* quanto na classe *MatES* através da função pública Ordenar(), passando como parâmetro a matriz que se deseja ordenar. Caso o resultado da ordenação deva ser aplicado na matriz passada como parâmetro, a função *Aplicar()* dever definida como verdadeira durante a configuração das propriedades da classe. As demais funções públicas da classe são mostradas na Figura 68.

PONTOS DE ALAVANCAMENTO

Em virtude dos pontos de alavancamento estarem relacionados diretamente à matriz de coeficiente do sistema linear resultante do processo de minimização dos resíduos e sua identificação e tratamento se limitarem ao conhecimento desta matriz, entende-se ser correto abordá-los dentro do âmbito das ferramentas matemáticas. A classe *PesoPA* foi desenvolvida para tratar deste assunto, uma vez que a especificidade dos pontos de alavancamento extrapola os limites conceituais da classe SistemaLinear.

Como as demais classes apresentadas acima, a classe *PesoPA* utiliza a matriz de coeficientes do sistema linear somente para obter as informações necessárias aos cálculos – não alterando seus valores. Trata-se de uma classe simples e direcionada que utiliza os recursos da classe *Matriz* para realizar quase todos os cálculos necessários. Seu principal objetivo é determinar o peso associado a cada linha da matriz de coeficientes e, consequentemente, indicar a presença de pontos discrepantes.

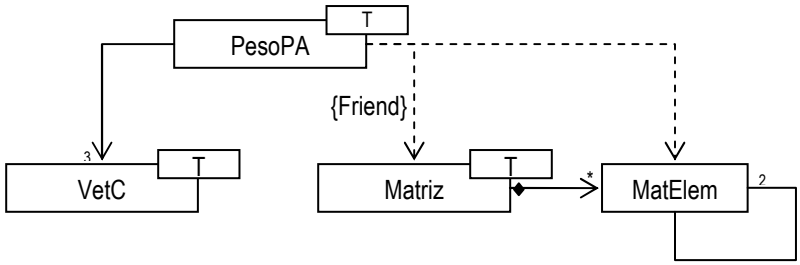


Figura 69 – Relacionamentos da classe *PesoPA*

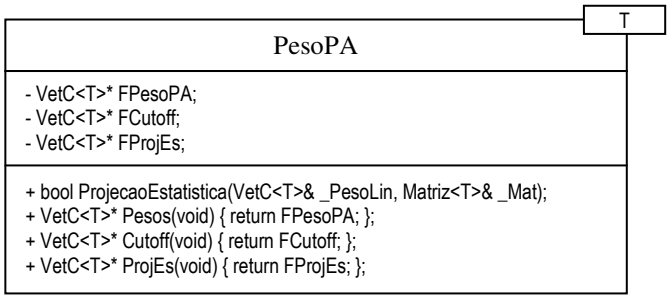


Figura 70 – Estrutura interna da classe *PesoPA*.

A classe PesoPA implementa os procedimentos de identificação e tratamento apresentados em 4.6.3.4 armazenando o valor de limiar (FCutoff), o valor do índice de projeção estatística (FProjEs) e a ponderação (FPesoPA) de cada linha da matriz informada. Para tanto é necessário informar a matriz de coeficientes do sistema linear com o respectivo vetor de ponderação através da função *ProjecaoEstatistica()*. A Figura 70 mostra a estrutura interna da classe.

CONSIDERAÇÕES FINAIS SOBRE AS FERRAMENTAS MATEMÁTICAS

O presente documento se limitou a comentar as principais funcionalidades da ferramenta matemática desenvolvida, não significando que as classes vetoriais e matriciais apresentadas não possuam outras funções. Entendeu-se ser extenso, cansativo e desnecessário o detalhamento pormenorizado de todas as classes.

Os resultados comparativos e o desempenho das classes e de suas respectivas funções serão mostrados no capítulo destinado a resultados numéricos. Por ora, cabe ressaltar que muitas outras funcionalidades, como o cálculo de autovetores e autovalores, ainda necessitam ser desenvolvidas para possibilitar a aplicação desta ferramenta de forma integral a outras funções de SEE.

4.3.6 – Estimação de estados em sistema elétricos de potência

Apesar da arquitetura básica do projeto indicar de forma genérica um pacote para aplicativos e funções de SEE, conforme Figura 38, o presente trabalho seguiu as diretivas iniciais de projeto e focou-se no desenvolvimento de estimadores de estados para sistemas elétricos de potência. Novas funções baseadas no modelo proposto neste trabalho podem, futuramente, ser desenvolvidas e integrar o referido pacote de classes. Esta integração deve ser incentivada por possibilitar que funções aparentemente distintas e baseadas num único modelo possam compartilhar seus recursos, gerando um ambiente propício para o desenvolvimento de novas pesquisas e estudos.

Esta seção tem o objetivo de detalhar a estrutura e o relacionamento das classes que compõem os estimadores de estados implementados no projeto, mostrando seus recursos, a seqüência de execução e as dificuldades encontradas no tratamento de alguns componentes. Inicialmente apresentar-se-á a formulação teórica dos estimadores de mínimos quadrados ponderados (MQP) e dos estimadores robustos de mínimos quadrados com ponderação variável (MQPVI) e suas variantes. Posteriormente será introduzida a estrutura de classes proposta para

abordar esta função no âmbito deste projeto, desde as classes gerais abstratas até o detalhamento das classes destinadas a cada um dos estimadores de estado desenvolvidos.

4.3.6.1 – Considerações iniciais

A utilização da estimação de estados em sistemas elétricos de potência é intimamente ligada ao conceito de segurança operativa. Os operadores do sistema necessitam de informações confiáveis que subsidiem o processo decisório frente à ocorrência de eventos comuns ou excepcionais. Erros operativos podem gerar danos em equipamentos, corte parcial de carga e – em casos extremos – queda total do sistema elétrico interligado, resultando invariavelmente em prejuízos tanto dos agentes do sistema quanto dos consumidores. Portanto, além de prover treinamento adequado e contínuo de operadores e estabelecer procedimentos de segurança, as empresas operadoras devem dispor de ferramentas computacionais para minimizar a probabilidade de incidentes operativos. Tais ferramentas compõem o sistema de supervisão operativa que deve ser capaz de lidar com problemas provenientes do sistema de telemetria, de informar as conseqüências de possíveis contingências, de indicar o nível de segura e as possíveis ações de restabelecimento e, por fim, de prever o estado sistêmico no curto prazo.

A estimação de estados para sistemas elétricos de potência é uma das funções que compõem a seqüência lógica e operacional dos centros de operação de sistemas. Ela foi proposta inicialmente por F.C. Schweppe, J. Wildes e D. Row [92] como uma forma de minimizar os problemas gerados pela existência de medidas errôneas ou pela perda de canais de comunicação, tendo como principais objetivos a filtragem de medidas redundantes, a eliminação de medidas errôneas, a obtenção de estados confiáveis e a produção de algumas informações em partes não monitoradas do sistema – tais como fluxos ou injeções líquidas de potência. Atualmente seu uso foi estendido à análise de erros em configuração de redes [93], entretanto tal abordagem não será tratada neste trabalho.

A formulação matemática do problema de estimação de estados em sistemas elétricos de potência baseia-se na adoção de um modelo não-linear de medição com considerações peculiares e na minimização de resíduos com solução dada através do método de Gauss-Newton ou Newton-Raphson. As subseções a seguir irão detalhar a formulação matemática da EESP e seus respectivos métodos de detecção e tratamento de medidas consideradas grosseiras.

4.3.6.1 – O modelo de medição não-linear

O sistema elétrico de potência pode ser representado através de um modelo físico-matemático que descreve o comportamento de suas grandezas baseando-se num conjunto de variáveis, as quais são denominadas de *estados do sistema*. Portanto todas as grandezas do sistema, ou melhor, o estado operativo do sistema pode ser determinado conhecendo-se seu modelo e o valor de seus estados. Se o SEP for representado através de um modelo DC, os estados do sistema serão os ângulos de tensão de todas as suas barras. Por outro lado, se o SEP for representado através de seu modelo AC, os estados do sistema serão todos os módulos e ângulos de tensão de suas barras. Caso se deseje determinar o estado operativo do sistema conhecendo-se o valor do ângulo da barra de referência, devem ser encontrados todos os demais estados do sistema, ou seja, deverão ser calculados N-1 estados para o modelo DC e 2N-1 estados para o modelo AC – onde N representa o número de barras do sistema. Neste caso, as variáveis que representarão os estados não conhecidos junto ao conjunto de equações que modela o problema de estimação são denominadas de *variáveis de estado*.

O centro de operação de sistema dispõe de um conjunto de medidas oriundas dos sistemas de telemetria. Tais medidas são grandezas relacionadas a um determinado componente do sistema que podem ser descritas através de equações cujas variáveis são os estados do sistema. Considerando que todas as medidas possuem erros que são inerentes ao processo de medição, o conjunto de medidas pode ser representado pelo seguinte modelo:

$$\underline{z} = h(\underline{x}) + \underline{e} \tag{4.29}$$

onde $\underline{z}$ é o vetor de grandezas medidas, $h(.)$ é o função-vetor não-linear relacionando os estados às grandezas medidas, $\underline{x}$ é o vetor de estados e $\underline{e}$ é vetor de erros de medição gerados pela imprecisão dos medidores, distorções causadas pelos transformadores de instrumentos, etc.

Pode-se supor que os erros de medição sejam pequenos e regidos através da distribuição normal de Gauss, tenham esperança matemática nula e sejam não-correlacionados. Desta forma, tem-se:

$$E(\underline{e}) = 0 \quad E\left(\underline{e} \cdot \underline{e}^T\right) = R_Z \tag{4.30}$$

onde R_Z é a matriz de covariância dos erros de medição.

As equações (4.29) e (4.30) constituem o modelo de medição adotado em EESP [92-93], o qual relaciona os estados do sistema ao conjunto de medidas e proporciona a formulação do problema de minimização dos resíduos que será apresentado nas subseções a seguir. Caso o

SEP seja representado através do modelo AC o modelo de medição será não-linear. Caso o SEP seja representado através do modelo DC o modelo de medição será linear.

4.3.6.2. Redundância no conjunto de medidas

A capacidade de se estimar estados depende tanto da quantidade de medidas disponíveis quanto da forma com que as medidas estão distribuídas através do sistema. Caso existam medidas suficientes com uma distribuição adequada, a estimação de estados poderá ser realizada e, neste caso, a rede é considerada observável. Quanto maior for o número de medidas disponíveis, melhor será as propriedades estatísticas do estimador e mais próximo do verdadeiro estado operativo estará seu resultado. Para melhorar o resultado do estimador é comum impor a formulação de seu problema um conjunto de restrições extraídas das condições físicas do sistema elétrico de potência, como as injeções de potência nulas em barras de passagem. Tais restrições melhoraram as propriedades estatísticas do estimador de estados, uma vez que acrescentam informações livre de erros ou limitam a possibilidade de ocorrência de alguns erros. Dependendo da condição física imposta, a restrição pode ser considerada ou representada como uma medida fictícia ou *psedomedida*.

Uma das formas de se analisar a qualidade do plano de medição é através da utilização de índices associados à redundância local de medidas [94]. O conceito de redundância local de medidas baseia-se nas definições de *conjunto fundamental* e *máxima fração de contaminação* ($f_{j,\max}$) de medidas associadas a cada estado.

Define-se conjunto fundamental, $Z_j = \{z_i\}$, como sendo o grupo de medidas, z_i, associadas a cada variável de estado, x_j. Portanto, o conjunto de medidas que figuram em cada coluna da matriz Jacobiana é um conjunto fundamental.

O número máximo de medidas com erros grosseiros associadas a cada estado que pode ser processado por um estimador deve obedecer a seguinte condição [55] [94]:

$$f_{j,\max} \leq \left\lceil \frac{m_j - 1}{2} \right\rceil \tag{4.31}$$

onde m_j é o tamanho do j-nésimo conjunto fundamental, e a operação $\lceil \cdot \rceil$ representa a parte inteira do argumento.

A condição expressa na (4.31), quando respeitada, permite aos estimadores de estados obterem estimativas válidas na hipótese de que medidas com erros grosseiros tenham sido corretamente identificadas.

4.3.6.3 O problema de EESP formulado através de métodos de MQP

O problema de estimação de estado pode ser formulado através do método de mínimos quadrados ponderados cuja função-custo é:

$$J(\underline{x}) = \left[\underline{z} - h(\underline{x})\right]^T \cdot R_z^{-1} \cdot \left[\underline{z} - h(\underline{x})\right] = \underline{r}^T \cdot R_z^{-1} \cdot \underline{r} = \sum_{i=1}^{m} \left(\frac{z_i - h_i(\underline{x})}{\sigma_i} \right)^2 \qquad (4.32)$$

onde $\underline{r} = \underline{z} - h(\underline{x})$ é o vetor de resíduos, σ_i é o desvio padrão da i-nésima medida e *m* é o número total de medidas.

Os estados que melhor representam o ponto operativo do sistema serão obtidos quando a função-custo mostrada acima for mínima. Entretanto o problema de minimização deve considerar a precisão de cada medida, ou seja, quanto menor o desvio padrão de uma determinada medida, maior deverá ser a influência de seu resíduo. Portanto a utilização do método de mínimos quadrados ponderados é aderente ao problema de minimização nas condições descritas.

O vetor de estados, conforme será visto adiante, pode ser obtido através do seguinte processo iterativo:

$$\begin{aligned} G\left(\underline{x}^k\right) \cdot \Delta \underline{x}^k &= -g\left(\underline{x}^k\right) \\ \underline{x}^{k+1} &= \underline{x}^k + \Delta \underline{x}^k \end{aligned} \qquad (4.33)$$

onde $g(\underline{x})$ é o gradiente de $J(\underline{x})$ e $G(\underline{x})$ é a matriz ganho que pode ser obtida através do método de solução do problema de minimização, podendo ser tanto o método Gauss-Newton quanto o método Newton-Raphson. Para que $G(\underline{x})$ obtido através do método de Gauss-Newton seja igual a $G(\underline{x})$ obtido através do método de Newton-Raphson, as segundas derivadas da matriz Hessiana de $J(\underline{x})$ devem ser ignoradas [93]. A convergência do processo é obtida quando o máximo valor incremental dos estados for menor ou igual a uma tolerância pré-estabelecida ($\max|\Delta x_i| \leq \varepsilon$). Costuma-se adotar, normalmente, valores na faixa de 1,0x10^{-3}.

SOLUÇÃO DO PROBLEMA PELO MÉTODO DE GAUSS-NEWTON

A função-vetor não-linear $h(\underline{x})$ pode ser linearizada em torno do ponto $\underline{x}^k$ e ao longo da direção $\Delta \underline{x} = \underline{x} - \underline{x}^k$ através da série de Taylor até o termo de primeira ordem, resultando:

$$h(\underline{x}) \approx h(\underline{x}^k) + H(\underline{x}^k) \cdot \Delta \underline{x} \qquad (4.34)$$

onde $H(\underline{x}) = \dfrac{\partial h(\underline{x})}{\partial \underline{x}}$ é a matriz Jacobinada.

A substituição de (4.34) em (4.29) resulta na linearização do plano de medição, $\Delta \underline{z} = H(\underline{x}^k) \cdot \Delta \underline{x} + \underline{e}$, o qual possui as mesmas propriedades do que plano de medição não linear – conforme (4.30). Logo $J(\underline{x})$ pode ser reescrita da seguinte forma:

$$J(\Delta \underline{x}) = \left[\Delta \underline{z} - H(\underline{x}^k) \cdot \Delta \underline{x}\right]^T \cdot R_z^{-1} \cdot \left[\Delta \underline{z} - H(\underline{x}^k) \cdot \Delta \underline{x}\right] \quad (4.35)$$

onde $\Delta \underline{z} = \underline{z} - h(\underline{x}^k)$. O mínimo da função-custo será obtido quando:

$$\frac{\partial J(\Delta \underline{x})}{\partial \Delta \underline{x}} = 0 \rightarrow -H^T(\underline{x}^k) \cdot R_z^{-1} \cdot (\Delta \underline{z} - H(\underline{x}^k) \cdot \Delta \underline{x}) = 0 \rightarrow H^T(\underline{x}^k) \cdot R_z^{-1} \cdot H(\underline{x}^k) \cdot \Delta \underline{x} = H^T(\underline{x}^k) \cdot R_z^{-1} \cdot \Delta \underline{z}$$

$$\therefore$$

$$G(\underline{x}^k) = H^T(\underline{x}^k) \cdot R_z^{-1} \cdot H(\underline{x}^k) \quad e \quad g(\underline{x}^k) = -H^T(\underline{x}^k) \cdot R_z^{-1} \cdot \Delta \underline{z}$$

O incremento do vetor de estados ser obtido através da seguinte equação:

$$\Delta \underline{x} = \left[H^T(\underline{x}^k) \cdot R_z^{-1} \cdot H(\underline{x}^k)\right]^{-1} \cdot H^T(\underline{x}^k) \cdot R_z^{-1} \cdot \Delta \underline{z} \quad (4.36)$$

4.3.6.4 Tratamento de erros grosseiros na EESP clássica

Conforme as suposições do modelo de medição, os erros associados às medidas apresentam distribuição gaussiana. Tais erros são considerados normais caso estejam dentro da faixa de ±3σ e, neste caso, serão adequadamente filtrados pelo EE caso haja redundância de medidas. Erros que extrapolam essa faixa são considerados grosseiros e devem ser adequadamente tratados por comprometerem o resultado da estimação de estados. Quando a magnitude destes erros ultrapassa a faixa de ±20σ, sua detecção tende a ser realizada por algoritmos de pré-filtragem. Entretanto, se existir os erros de magnitude entre ±3σ e ±20σ, o EE deve detectá-los, identificá-los e excluí-los ou recuperá-los. Apesar das diversas técnicas apresentadas pela literatura [92] [81-82] [95], este trabalho se limitará a apresentar o algoritmo baseado em *testes de hipótese* [83], o qual foi implementado neste projeto.

DETECÇÃO DE ERROS GROSSEIROS ATRAVÉS DE TESTE DE HIPÓTESES

Caso o conjunto $X = \{x_1, ...x_m\}$ seja composto por variáveis aleatórias independentes normalmente distribuídas que possuam esperança matemática zero e variância unitária, ou seja, distribuição $N(0,1)$, então a variável aleatória $y = \sum_{i=1}^{m} x_i^2$ terá distribuição qui-quadrada com ***m*** graus de liberdade (χ_m^2) [93]. Pode ser mostrado, então, que a distribuição qui-quadrada de *y* terá média *m* e variância 2*m* [83]. Caso o conjunto $X = \{x_1, ...x_m\}$ seja definido por um conjunto

de ***m*** equações das quais ***n*** sejam linearmente independentes ($m \geq n$), então y terá distribuição qui-quadrada com ***m-n*** graus de liberdade (χ^2_{m-n}).

Assumindo as suposições feitas na formulação do modelo de medição, ou seja, que erros de medição $\underline{e} = \{e_1, \ldots, e_m\}$ sejam não-correlacionados e regidos através da distribuição normal de Gauss e tenham esperança matemática nula e variância σ_i^2, ou seja, distribuição $N(0, \sigma_i^2)$, a função-custo será definida por:

$$J(\underline{x}) = \sum_{i=1}^{m} \left(\frac{z_i - h_i(\underline{x}^k)}{\sigma_i} \right)^2 = \sum_{i=1}^{m} \left(\frac{z_i - \hat{z}_i}{\sigma_i} \right)^2 ,$$

podendo considerá-la como tendo distribuição Qui-quadrada com ***m-n*** graus de liberdade (χ^2_{m-n}), onde *m* é o número de medidas e *n* é o número de variáveis de estado. Ressalta-se que $\hat{z}_i = h_i(\underline{x}^k)$ deve ser calculado num ponto próximo ao ponto de solução. Portanto pode-se esperar que:

$$E[J(\underline{x})] = m - n \quad e \quad E\left\{[J(\underline{x}) - (m-n)]^2\right\} = 2 \cdot (m-n) \tag{4.37}$$

A formulação acima permite definir um teste de hipótese que verifique a condição descrita pela equação (4.37), onde H_0, $E[J(\underline{x})] = m - n$; e H_1, $E[J(\underline{x})] > m - n$. Através da hipótese alternativa (H1) pode determinar que:

- Se $J(\underline{x}) > C$ deve-se rejeitar H_0 e concluir que existe pelo menos um erro grosseiro no conjunto de medidas;

- Se $J(\underline{x}) \leq C$ deve-se aceitar H_0 e concluir que os erros das medidas situam-se dentro da faixa esperada e foram filtrados devidamente filtrados.

Neste caso C é definido com base na probabilidade de falso alarme (α) ou de se cometer um erro do tipo 1, ou seja, de se rejeitar H_0 sendo esta hipótese realmente verdadeira. Assim:

$$C = \chi^2_{m-n,1-\alpha} \tag{4.38}$$

O teste de hipóteses acima deve ser feito num ponto próximo ao da solução habitual do sistema. Portanto deve-se iniciar o procedimento de detecção de erros grosseiros numa iteração relativamente próxima à iteração habitual do estimador de estados para um dado sistema. Caso a hipótese nula (H_0) seja rejeitada, deve-se identificar o erro grosseiro. Caso contrário, o processo de estimação de estados deve ser continuado.

IDENTIFICAÇÃO DE ERROS GROSSEIROS

Caso seja constatado que o conjunto de medidas possui uma ou mais medidas contaminadas com erros do tipo grosseiro ($\pm 3\sigma < e_i < \pm 20\sigma$), tais medidas devem ser identificadas para posterior tratamento. O processo de identificação baseia-se na análise dos resíduos de estimação, entretanto o exame direto da magnitude dos resíduos não é uma forma confiável de se identificar as medidas com erros grosseiros. Sabe-se que as medidas possuem diferentes valores de desvio-padrão, portanto um erro que pode ser considerado normal para determinadas medidas pode ser grosseiro para outras. Logo os resíduos devem ser comparados sobre uma base comum, ou seja, os resíduos devem ser divididos pelo valor de seus respectivos desvios-padrão. Tal processo é conhecido como *normalização de resíduos*. Para tanto é necessário calcular a matriz de covariância dos resíduos de estimação.

Todos os vetores e matrizes que são funções do vetor de estados do sistema serão apresentados, deste ponto em diante, de forma simplificada. Assim $G(\underline{x})$ e $H(\underline{x})$ serão apresentados como $G \; e \; H$, respectivamente.

Através da comparação de (4.32) e (4.35), conclui-se que:

$$\underline{\hat{r}} = \Delta \underline{z} - H \cdot \Delta \underline{x} \tag{4.39}$$

Aplicando substituindo (4.36) em (4.39), obtém-se:

$$\underline{\hat{r}} = \left[I - H \cdot \left(H^T \cdot R_z^{-1} \cdot H \right)^{-1} \cdot H^T \cdot R_z^{-1} \right] \cdot \Delta \underline{z} \tag{4.40}$$

Se $y \in \Re^m$ é um vetor de variáveis aleatórias cuja esperança matemática é $E(y) = \bar{y}$ e $v \in \Re^n$ é um vetor de variáveis aleatórias cuja especa matemática é $E(v) = \bar{v}$. Assumindo que exista uma matriz $A \in \Re^{mxn}$ tal que:

$$y = A \cdot v \tag{4.41}$$

Então a matriz de covariância R_y pode ser definida como:

$$R_y = E\left[\left(y - \bar{y} \right) \cdot \left(y - \bar{y} \right)^T \right] = E\left[A \cdot \left(v - \bar{v} \right) \cdot \left(v - \bar{v} \right)^T \cdot A^T \right] = A \cdot R_v \cdot A^T \tag{4.42}$$

Aplicando a teoria formulada em (4.41) e (4.42) em (4.40), obtém-se:

$$R_{\hat{r}} = \left[I - H \cdot \left(H^T \cdot R_z^{-1} \cdot H \right)^{-1} \cdot H^T \cdot R_z^{-1} \right] \cdot R_z \cdot \left[I - H \cdot \left(H^T \cdot R_z^{-1} \cdot H \right)^{-1} \cdot H^T \cdot R_z^{-1} \right]^T \tag{4.43}$$

Desenvolvendo (4.43), tem-se:

$$R_{\hat{r}} = R_z - H \cdot G^{-1} \cdot H^T = \left[I - H \cdot \left(H^T \cdot R_z^{-1} \cdot H \right)^{-1} \cdot H^T \cdot R_z^{-1} \right] \cdot R_z \tag{4.44}$$

Da mesma forma, aplicando a teoria formulada em (4.41) e (4.42) em (4.36), obtém-se:

$$R_{\hat{x}} = \left[G^{-1} \cdot H^T \cdot R_z^{-1} \right] \cdot R_z \cdot \left[G^{-1} \cdot H^T \cdot R_z^{-1} \right]^T = G^{-1} \tag{4.45}$$

Agora aplicando a teoria formulada em (4.41) e (4.42) em (4.34), obtém-se:

$$R_{\hat{z}} = H \cdot R_{\hat{x}} \cdot H^T = H \cdot G^{-1} \cdot H^T \tag{4.46}$$

Reformulando (4.44) conhecendo-se (4.46), concluí-se:

$$R_{\hat{r}} = R_z - H \cdot G^{-1} \cdot H^T = R_z - R_{\hat{z}} \tag{4.47}$$

Definindo a matriz de sensibilidade dos resíduos e reescrevendo (4.40), tem-se:

$$S = I - H \cdot G^{-1} \cdot H^T \cdot R_z^{-1} = R_{\hat{r}} \cdot R_z^{-1} \rightarrow \hat{r} = S \cdot z \tag{4.48}$$

A equação (4.48) torna claro que medidas contaminadas com erros considerados grosseiros afetam as demais medidas do plano de medição, umas mais outras menos, conforme o grau de sensibilidade definido através da matriz de sensibilidade dos resíduos.

Com a determinação da matriz de covariância dos resíduos de estimação (4.47), pode-se definir o vetor de resíduos normalizados conforme a equação abaixo:

$$r_i^n = \frac{r_i}{\sqrt{w_{ii}}} \tag{4.49}$$

onde w_{ii} é a diagonal da matriz $R_{\hat{r}}$.

A medida possuir o maior resíduo normalizado absoluto será identificada como possuidora de erro grosseiro. Apesar de existirem métodos que permitem a identificação múltipla de medidas com erros grosseiros [93], o presente método só pode ser utilizado para identificar uma única medida por vez. Isto se deve ao espalhamento do erro mencionado acima.

Após a identificação do erro grosseiro, a medida pode ser eliminada do processo de estimação ou recuperada através da estimação da magnitude de seu erro. Este último caso é conhecido como recuperação de medidas afetadas por erros grosseiros.

RECUPERAÇÃO DE MEDIDAS COM ERROS GROSSEIROS

A relação entre as medidas e os resíduos estimados é dada pela seguinte (4.48). Supondo que z_j é a medida identificada como possuidora de erro grosseiro, então j-nésima equação de (4.48) pode ser escrita da seguinte forma:

$$\hat{r}_j = S_j \cdot z \,,$$

onde $\hat{r}_j$ é o resíduo estimado da j-nésima medida e S_j é a j-nésima linha da matriz de sensibilidade S.

Caso c_j seja a correção estimada para retirar o efeito do erro grosseiro da respectiva medida, a medida resultante seria escrita da seguinte forma:

$$z^{corr} = z + c_j . u_j \tag{4.50}$$

onde z^{corr} é o vetor de medidas corrigido e u_j é um vetor coluna nulo, exceto pelo valor unitário do elemento de posição *j*.

Sendo c_j estimado de tal forma que o resíduo resultante de sua adição seja nulo, tem-se:

$$r_j^{corr} = S_j \cdot (z + c_j u_j) = \hat{r}_j + c_j \cdot S_{jj} = 0 \Rightarrow c_j = -\hat{r}_j / S_{jj} \qquad (4.51)$$

onde S_{jj} é o j-nésimo elemento diagonal da matriz de sensibilidade dos resíduos.

Portanto, caso se opte pela preservação da redundância, pode-se corrigir a medida através das equações (4.50) e (4.51) ao invés de eliminá-la do vetor de medidas.

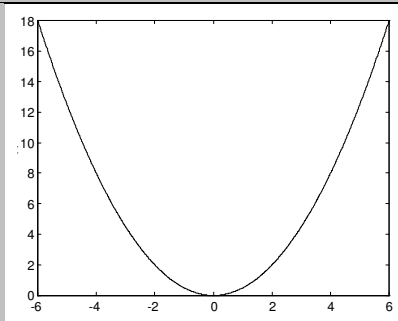
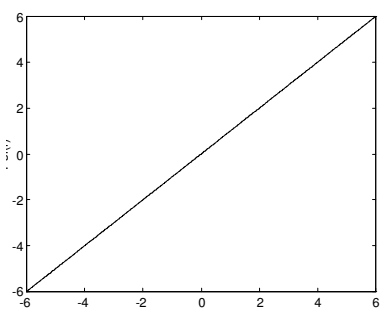
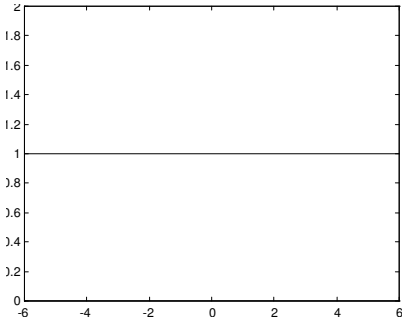
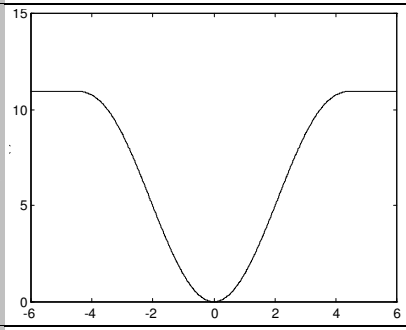
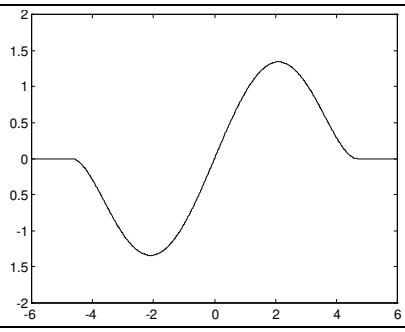
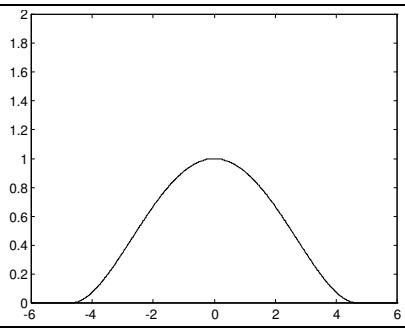
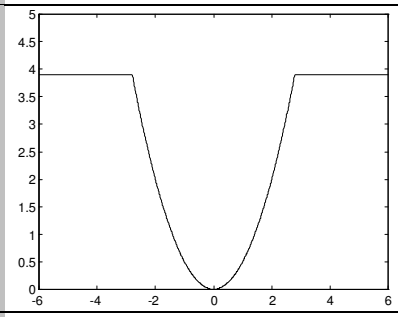
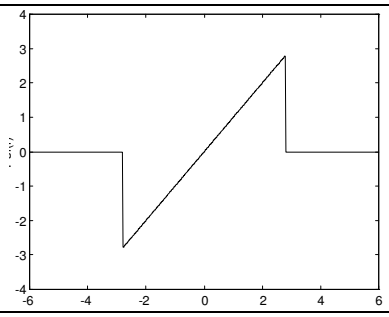
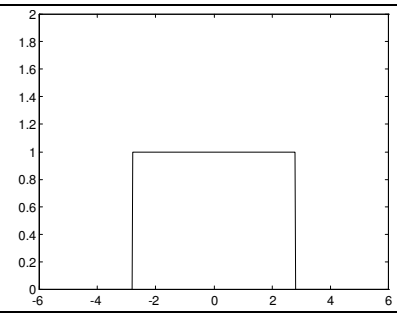
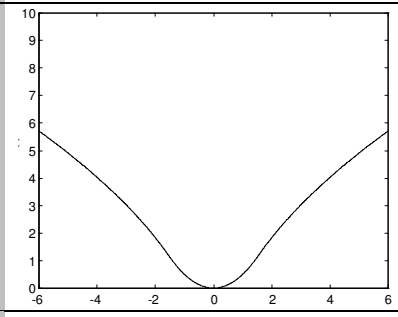
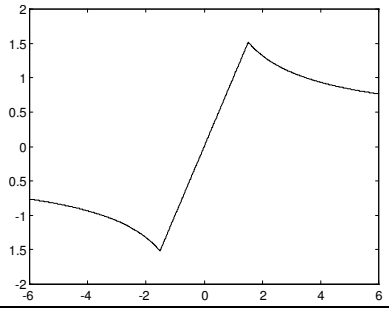
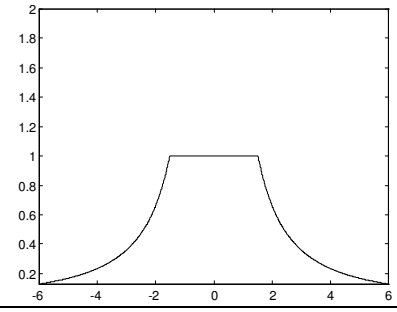
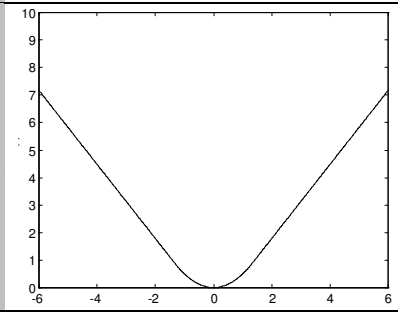
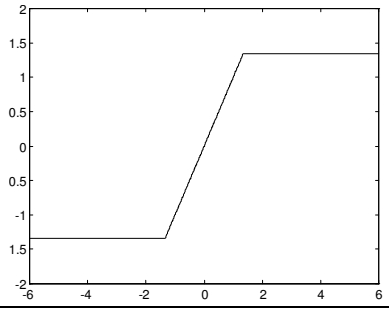
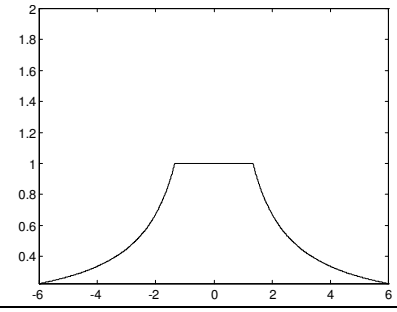
4.3.6.5 Estimadores baseados em critérios não-quadráticos

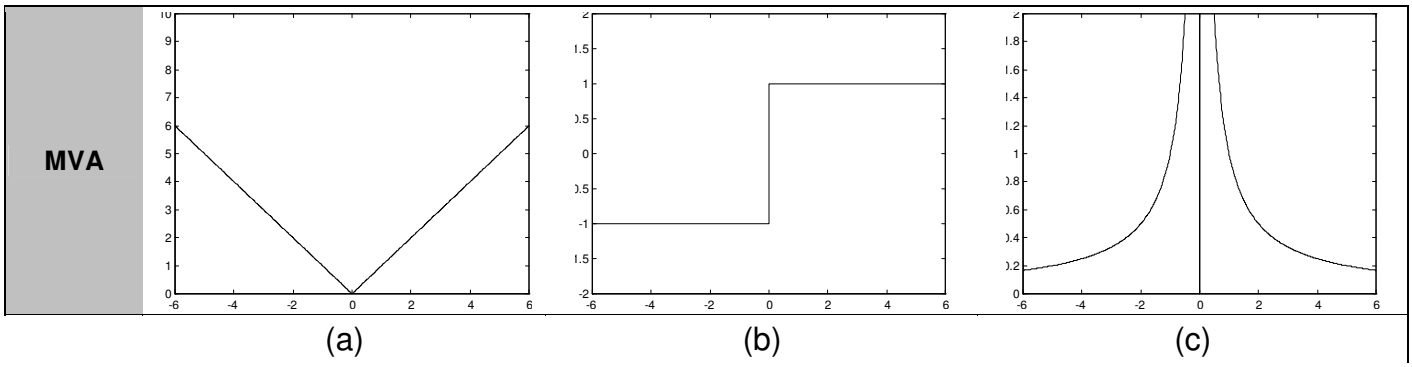
Os estimadores baseados em critérios não-quadráticos foram formulados com objetivo de reduzir a influência de medidas errôneas dentro do processo de estimação de estados, utilizando uma ponderação dinâmica ou variável associada à magnitude dos resíduos de estimação. As medidas que apresentarem resíduos superiores a um determinado limite são subponderadas e, consequentemente, passam ter menor influência que as demais medidas. Tanto a forma de ponderação quanto os limites entre o segmento quadrático e o não quadrático – conhecido como ponto de transição – variam conforme o estimador utilizado e sua eficiência estatística [55], entretanto a maioria destes estimadores *penaliza* a medida de forma proporcional à inversa da magnitude do resíduo, ou seja, quanto maior o resíduo maior a penalidade.

Tabela 3 – Características dos estimadores

Estim.	*Domínio*	$\rho(r)$	$\psi(r) = \dot{\rho}(r)$	β
MQP	$\Re$	$r^2/2$	r	∞
QCA	$\lvert r \rvert \leqslant \beta$ $\lvert r \rvert > \beta$	$(\beta^2/2)[1-[1-(r/\beta)^2]^3]$ $\beta^2/2$	$3r[1-(r/\beta)^2]^2$ 0	4.685
QC	$\lvert r \rvert \leqslant \beta$ $\lvert r \rvert > \beta$	$r^2/2$ $\beta^2/2$	r 0	2.795
QRQ	$\lvert r \rvert \leqslant \beta$ $\lvert r \rvert > \beta$	$r^2/2$ $2\beta^{3/2}\sqrt{\lvert r \rvert} - \frac{3}{2}\beta^2$	r $\beta^{3/2}\frac{sign(r)}{\sqrt{\lvert r \rvert}}$	1.264
QT	$\lvert r \rvert \leqslant \beta$ $\lvert r \rvert > \beta$	$r^2/2$ $\beta \lvert r \rvert - \beta^2/2$	r $sign(r).\beta$	1.345
MVA	$\Re$	$\lvert r \rvert$	$sign(r)$	0

Tabela 4 – Propriedades dos estimadores

Estimador	$\rho(\cdot)$	$\Psi(\cdot)$	$q(\cdot)$
MQP			
QCA			
QC			
QRQ			
QT			



Os seguintes estimadores que são detalhados nas Tabela 3 e Tabela 4: *Mínimos quadrados ponderados* (MQP, Gauss/Legendre – 1795); *Quadrático-constante analítico* (QCA, Beaton/Tukey – 1974); *Quadrático-constante* (QC, Huber – 1964; Hinich/Talwar – 1975); *Quadrático-raiz quadrada* (QRQ, Merril/Schweppe – 1971); *Quadrático-tangente* (QT, Huber – 1964); *Mínimo valor absoluto* (MVA).

Na Tabela 3, as variáveis r e β, representam os resíduos de estimação e o valor do ponto de transição que define cada estimador, respectivamente. Os valores de β utilizados neste trabalho foram definidos a partir de uma eficiência estatística de 95 %. As curvas mostradas na Tabela 4 são da função-custo $\rho()$, da derivada da função-custo em relação aos resíduos $\psi()$ e da função peso $q()$. Por fim, cabe comentar que dentre os estimadores detalhados acima, os únicos a apresentam a função $\rho()$ convexa em todo o seu domínio são os estimadores QT e MVA. Trata-se de uma propriedade importante, uma vez que garante a inexistência de mínimos locais [96].

Os estimadores baseados em critérios não-quadráticos foram inicialmente utilizados em EESP por Merrill e Schweppe [97]. Outras propostas neste sentido podem ser encontradas em [98-101]. Apesar do relativo sucesso conseguido, a utilização de pontos de transição que definem a função-custo $\rho()$ e a primeira derivada $\psi()$ em bases heurísticas podem comprometer a eficiência e/ou robustez dos estimadores.

O estimador de estados quadrático-tangente foi utilizado no tratamento de pontos de alavancamento por Mili et alli [91], obtendo bons resultados tanto na detecção e tratamento dos pontos de alavancamento quanto na obtenção da solução do problema de EESP. Posteriormente, Pires et alli [102] aumentaram a robustez numérica deste estimador incorporando as rotações de Givens baseadas em 3 multiplicadores como método de fatoração, conforme proposta de Simões-Costa e Quintana [63]. Por fim, Pires [55] comparou, através desta formulação, outros estimadores baseados em critérios não quadráticos – conforme a relação mostrada nas tabelas 3 e 4 (com exceção do estimador MQP) –, concluindo que o estimador quadrático-tangente proposto por Huber era o mais robusto por não estar sujeito ao

problema de solução de mínimo local. O estimador resultante destas pesquisas, conhecido por estimador de estados robusto de mínimos quadrados com ponderação variável (ou IRLS – *Iteratively Reweightes Least-Square*), por possuir robustez tanto estatística quanto numérica foi escolhido como o estimador a ser prioritariamente implementado neste projeto. A seguir apresenta-se sua formulação matemática.

4.3.6.6 EE robusto de mínimos quadrados com ponderação variável [91]

O problema de estimação de estados formulado através de critérios não-quadráticos e considerando os pontos de alavancamento pode ser apresentado como:

$$\min J(\underline{\hat{x}}) = \underline{w}^T \cdot \rho(\underline{\tilde{r}}_S) \cdot \underline{w} \tag{4.52}$$

onde $\rho(\underline{\tilde{r}}_S)$ representa o estimador não-quadrático utilizado, $\underline{w}$ representa a ponderação relativa aos pontos de alavancamento e $\underline{\tilde{r}}_S$ o vetor de resíduos padronizados e inversamente ponderados pelos pesos relativos aos pontos de alavancamento (conforme 4.6.3.4). Sendo:

$$\rho(\underline{\tilde{r}}_S) = \begin{bmatrix} \rho(\tilde{r}_{S1}) & & \\ & \ddots & \\ & & \rho(\tilde{r}_{Sm}) \end{bmatrix} \quad e \quad \underline{w} = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

O vetor de resíduos $\underline{\tilde{r}}_S$ é definido da seguinte maneira [91]:

$$\underline{\tilde{r}}_{Si} = \xi \cdot \underline{r}_{Si} = \frac{r_i}{w_i \cdot \sigma_i} \quad onde \quad \xi = \frac{1}{w_i} \tag{4.53}$$

Localizando o ponto de mínimo de (4.52), ou seja, $\nabla J(\underline{\hat{x}}) = 0$, tem-se:

$$\left(\frac{\partial \underline{\tilde{r}}_S}{\partial \underline{\hat{x}}}\right)^T \cdot \left(\frac{\partial \rho}{\partial \underline{\tilde{r}}_S}\right) \cdot \hat{w}^2 \cdot e = 0 \tag{4.54}$$

onde:

$$\hat{w} = \begin{bmatrix} w_1 & & \\ & \ddots & \\ & & w_m \end{bmatrix} \quad e \quad \hat{w}^2 \cdot e = \begin{bmatrix} w_1 \\ \vdots \\ w_m \end{bmatrix}$$

Reescrevendo (4.53), tem-se:

$$\underline{\tilde{r}}_S = R^{-1/2} \cdot \hat{w}^{-1} \cdot \underline{r} = R^{-1/2} \cdot \hat{w}^{-1} \cdot [\underline{z} - h(\underline{\hat{x}})] \tag{4.55}$$

onde:

$$R^{-1/2} = diag\left\{ 1/\sigma_1, \ldots, 1/\sigma_m \right\}$$

Reescrevendo (4.54) com base em (4.55), obtêm-se:

$$-H^T(\underline{\hat{x}})\cdot\hat{w}^{-1}\cdot R^{-1/2}\cdot\Psi(\underline{\tilde{r}}_S)\cdot\hat{w}^2\cdot e=\underline{0} \tag{4.56}$$

onde:

$$H=\left.\frac{\partial h(\underline{\hat{x}})}{\partial\underline{\hat{x}}}\right|_{\underline{\hat{x}}=\underline{\hat{x}}^K}\quad e\quad \Psi(\underline{\tilde{r}}_S)\equiv\frac{\partial\rho(\underline{\tilde{r}}_S)}{\partial\underline{\tilde{r}}_S} \tag{4.57}$$

Utilizando a propriedade de comutação de matrizes diagonais em (4.56), obtêm-se:

$$-H^T(\underline{\hat{x}})\cdot\hat{w}\cdot R^{-1/2}\cdot\Psi(\underline{\tilde{r}}_S)\cdot e=\underline{0} \tag{4.58}$$

Sendo $\tilde{R}_S=diag\{\tilde{r}_{S1},\ldots,\tilde{r}_{Sm}\}$, pode-se definir a matriz de ponderação iterativa dos resíduos:

$$Q(\underline{\tilde{r}}_S)=\Psi(\underline{\tilde{r}}_S)\cdot\tilde{R}_S^{-1} \tag{4.59}$$

Logo (4.58) pode ser reescrita da seguinte forma:

$$-H^T(\underline{\hat{x}})\cdot\hat{w}\cdot R^{-1/2}\cdot\Psi(\underline{\tilde{r}}_S)\cdot\tilde{R}_S^{-1}\cdot\tilde{R}_S\cdot e=-H^T(\underline{\hat{x}})\cdot R^{-1}\cdot Q(\underline{\tilde{r}}_S)\cdot\underline{r}=\underline{0} \tag{4.60}$$

sendo:

$$\left.\begin{array}{l}\hat{w}\cdot\tilde{R}_S=R^{-1/2}\cdot R_{ES}\\ \underline{r}=R_{ES}\cdot e\end{array}\right\}\quad\rightarrow\quad R_{ES}=\begin{bmatrix}r_1 & & \\ & \ddots & \\ & & r_m\end{bmatrix}$$

Por fim, pode-se novamente reescreve (4.60) como:

$$-H^T(\underline{\hat{x}})\cdot R^{-1}\cdot Q(\underline{\tilde{r}}_S)\cdot[\underline{z}-h(\underline{\hat{x}})]=\underline{0} \tag{4.61}$$

Expandido a função $h(\underline{\hat{x}})$ em torno de $\underline{x}^k$ através da série de Taylor e até termo de primeira ordem, tem-se:

$$h(\underline{\hat{x}})=h(\underline{\hat{x}}^k)+H(\underline{\hat{x}}^k)\cdot\Delta\underline{x}^k \tag{4.62}$$

Substituindo (4.62) em (4.61), resulta em:

$$\Delta\underline{x}^k=\left(H^T(\underline{\hat{x}}^k)\cdot R^{-1}\cdot Q(\underline{\tilde{r}}_S^k)\cdot H^T(\underline{\hat{x}}^k)\right)^{-1}\cdot H^T(\underline{\hat{x}}^k)\cdot R^{-1}\cdot Q(\underline{\tilde{r}}_S^k)\cdot\Delta\underline{z}^k \tag{4.63}$$

ou

$$\begin{array}{l}G_Q(\underline{\hat{x}}^k)\cdot\Delta\underline{x}^k=\cdot H^T(\underline{\hat{x}}^k)\cdot R^{-1}\cdot Q(\underline{\tilde{r}}_S^k)\cdot\Delta\underline{z}^k\\ onde\ \ G_Q(\underline{\hat{x}}^k)=H^T(\underline{\hat{x}}^k)\cdot R^{-1}\cdot Q(\underline{\tilde{r}}_S^k)\cdot H^T(\underline{\hat{x}}^k)\end{array} \tag{4.64}$$

A solução do problema será obtida através do processo iterativo mostrado em (4.33). Deve-se preferencialmente utilizar métodos de fatoração numericamente estáveis, tais como as rotações de Givens baseadas em 2 ou 3 multiplicadores, uma vez que a ponderação utilizada para tratar os pontos de alavancamento pode resultar em mal-condicionamento numérico.

4.3.6.7 Desacoplamento de planos do MDRDI [55]

O modelo de medição não linear (4.29) pode ser reescrito da seguinte maneira:

$$\begin{bmatrix} z_\delta \\ z_v \end{bmatrix} = \begin{bmatrix} h_\delta(\underline{x}) \\ h_v(\underline{x}) \end{bmatrix} + \begin{bmatrix} \underline{e}_\delta \\ \underline{e}_v \end{bmatrix} \quad \rightarrow \quad E\left\{\begin{bmatrix} \underline{e}_\delta \\ \underline{e}_v \end{bmatrix}\right\} = 0 \quad e \quad E\left\{\begin{bmatrix} \underline{e}_\delta \\ \underline{e}_v \end{bmatrix} \cdot \begin{bmatrix} \underline{e}_\delta \\ \underline{e}_v \end{bmatrix}^T\right\} = \begin{bmatrix} R_\delta & \\ & R_v \end{bmatrix}$$

onde: $z_\delta \; e \; z_v$ são os vetores de medidas relativos aos planos $P\delta \; e \; QV$, respectivamente; $h_\delta(\underline{x}) \; e \; h_v(\underline{x})$ são as funções lineares que relacionam os estados às medidas relativas aos planos $P\delta \; e \; QV$, respectivamente; $\underline{e}_\delta \; e \; \underline{e}_v$ são os erros de medição relativos às medidas pertencentes aos planos $P\delta \; e \; QV$, respectivamente; e $R_\delta \; e \; R_v$ são as matriz de covariância dos erros de medição relativas às medidas pertencentes aos planos $P\delta \; e \; QV$, respectivamente.

De forma análoga ao modelo de medição não-linear, a matriz Jacobiana também pode ser particionada. Assim:

$$H(\underline{x}) = \begin{bmatrix} H_{P\delta}(\underline{x}) & H_{PV}(\underline{x}) \\ H_{Q\delta}(\underline{x}) & H_{QV}(\underline{x}) \end{bmatrix} \rightarrow \begin{cases} H_{P\delta}(\underline{x}) = \dfrac{\partial h_\delta(\underline{x})}{\partial \underline{x}_\delta} & H_{PV}(\underline{x}) = \dfrac{\partial h_\delta(\underline{x})}{\partial \underline{x}_V} \\ H_{P\delta}(\underline{x}) = \dfrac{\partial h_V(\underline{x})}{\partial \underline{x}_\delta} & H_{QV}(\underline{x}) = \dfrac{\partial h_V(\underline{x})}{\partial \underline{x}_V} \end{cases}$$

Logo a matriz ganho $G(\underline{x})$ pode ser escrita da seguinte forma:

$$G(\underline{x}) = \begin{bmatrix} H_{P\delta}(\underline{x}) & H_{PV}(\underline{x}) \\ H_{Q\delta}(\underline{x}) & H_{QV}(\underline{x}) \end{bmatrix}^T \cdot \begin{bmatrix} R_\delta^{-1} & \\ & R_V^{-1} \end{bmatrix} \cdot \begin{bmatrix} Q_\delta(\tilde{\underline{r}}_{S\delta}) & \\ & Q_V(\tilde{\underline{r}}_{SV}) \end{bmatrix} \cdot \begin{bmatrix} H_{P\delta}(\underline{x}) & H_{PV}(\underline{x}) \\ H_{Q\delta}(\underline{x}) & H_{QV}(\underline{x}) \end{bmatrix}$$

onde $Q_\delta(\tilde{\underline{r}}_{S\delta})$ e $Q_V(\tilde{\underline{r}}_{SV})$ são os pesos iterativos relativos aos resíduos das medidas pertencentes aos planos $P\delta \; e \; QV$, respectivamente.

Desprezando-se as matrizes $H_{PV}(\underline{x})$ e $H_{Q\delta}(\underline{x})$, conforme o desacoplamento de planos realizado sobre o modelo, a matriz Jacobiana e a matriz ganho podem ser escritas da seguinte forma:

$$H(\underline{x}) = \begin{bmatrix} H_{P\delta}(\underline{x}) & \\ & H_{QV}(\underline{x}) \end{bmatrix} \quad e \quad G((\underline{x}) = \begin{bmatrix} G_{P\delta}(\underline{x}) & \\ & G_{QV}(\underline{x}) \end{bmatrix}$$

onde

$$G_{P\delta}(\underline{x}) = H_{P\delta}^T(\underline{x}) \cdot R_\delta^{-1} \cdot Q_\delta(\tilde{\underline{r}}_{S\delta}) \cdot H_{P\delta}(\underline{x}) \quad e \quad G_{QV}(\underline{x}) = H_{QV}^T(\underline{x}) \cdot R_V^{-1} \cdot Q_V(\tilde{\underline{r}}_{QV}) \cdot H_{QV}(\underline{x})$$

Esta consideração divide o problema de estimação de estados em dois subproblemas independentes relativos aos $P\delta \; e \; QV$, respectivamente. As variáveis de estado subproblema

$P\delta$ ($\underline{x}_\delta$) serão os ângulos de tensão das barras e o defasamento angular dos transformadores defasadores. Por outro lado, As variáveis de estado subproblema QV ($\underline{x}_V$) serão as magnitudes de tensão das barras e a posição do TAP dos transformadores com comutação sobre carga (LTC).

Cada iteração do problema completo corresponderá a duas iterações para o problema formulado com desacoplamento de planos, ou seja, uma iteração para cada plano. Neste caso, cada iteração é conhecida como ½ iteração.

A ½ iteração $P\delta$:

$$G_{P\delta}(\underline{x}_\delta^k)\cdot\Delta\underline{x}_\delta^k = H_{P\delta}^T(\underline{x}_\delta^k)\cdot R_\delta^{-1}\cdot Q_\delta(\underline{\tilde{r}}_{S\delta})\cdot\Delta z_\delta$$
$$\underline{x}_\delta^{k+1} = \underline{x}_\delta^k + \Delta\underline{x}_\delta^k$$

A ½ iteração QV:

$$G_{QV}(\underline{x}_V^k)\cdot\Delta\underline{x}_V^k = H_{QV}^T(\underline{x}_V^k)\cdot R_V^{-1}\cdot Q_V(\underline{\tilde{r}}_{SV})\cdot\Delta z_V$$
$$\underline{x}_V^{k+1} = \underline{x}_V^k + \Delta\underline{x}_V^k$$

Apesar das demais simplificações sugeridas para o desacoplamento no modelo, relativas às resistências das linhas de transmissão e a forma simplificada de cálculo da matriz Jacobiana, a implementação deste estimador somente desprezou as matrizes $H_{PV}(\underline{x})$ e $H_{Q\delta}(\underline{x})$.

4.3.6.7 EESP com restrições de igualdade

Algumas características físicas dos sistemas elétricos de potência podem ser consideradas na formulação do problema de EESP tanto para aumentar a redundância de informação quanto para impor as condições operativas que devem ser invariavelmente ser respeitadas. Estas características físico-operativas impõem ao problema de EESP um conjunto de restrições que devem ser respeitadas durante o processo de busca da solução. Apesar de serem muitas as possíveis restrições, aquelas que trazem benefícios diretos à EESP por serem incorporadas diretamente ao sistema de equações são as restrições de igualdade. Neste caso, a formulação do problema de EESP seria dada da seguinte maneira:

$$\min J(\underline{\hat{x}}) = \underline{r}^T\cdot R_z^{-1}\cdot\underline{r} \quad ou \quad \min J(\underline{\hat{x}}) = \underline{w}^T\cdot\rho(\underline{\tilde{r}}_S)\cdot\underline{w}, \quad onde \quad c(\underline{\hat{x}}) = \underline{b} \qquad (4.65)$$

onde $c(\underline{\hat{x}})$ é o vetor-função não-linear que relaciona os estados $\hat{x}$ ao vetor $\underline{b}$ que contém as informações determinísticas do sistema elétrico de potência, sendo $c(\underline{\hat{x}}) = \underline{b}$ o conjunto de equações que definem as restrições impostas.

Uma das formas de solução dos problemas formulados acima é considerar o conjunto de equações que definem as restrições como parte do problema de minimização, sendo que o peso atribuído a seus resíduos seria elevado quando comparado aos pesos atribuídos aos demais resíduos. Logo:

$$c_p(\underline{\hat{x}}) = \mu \cdot c(\underline{\hat{x}}) \quad e \quad \underline{b}_p = \mu \cdot \underline{b} \quad \rightarrow \quad c_p(\underline{\hat{x}}) = \underline{b}_p \quad onde \begin{cases} c_p(\underline{\hat{x}}) \in \Re^r \\ \underline{b}_p \in \Re^r \end{cases}$$

$$h_{aug}(\underline{\hat{x}}) = \begin{bmatrix} h(\underline{\hat{x}}) \\ c_p(\underline{\hat{x}}) \end{bmatrix}, \quad \underline{z}_{aug} = \begin{bmatrix} \underline{z} \\ \underline{b}_p \end{bmatrix} \quad e \quad R_{aug}^{-1} = \begin{bmatrix} \sigma_1^{-2} & & & & & \\ & \ddots & & & & \\ & & \sigma_m^{-2} & & & \\ & & & 1 & & \\ & & & & \ddots & \\ & & & & & 1 \end{bmatrix} \quad onde \begin{cases} h_{aug}(\underline{\hat{x}}) \in \Re^{m+r} \\ \underline{z}_{aug} \in \Re^{m+r} \\ R_{aug}^{-1} \in \Re^{(m+r)x(m+r)} \end{cases}$$

Tal método é conhecido como métodos dos pesos [60] e sua aplicação ao problema EESP formulado pelo MQP resulta em:

$$\min J(\underline{\hat{x}}) = \underline{r}_{aug}^T \cdot R_{aug}^{-1} \cdot \underline{r}_{aug} \quad \rightarrow \quad \begin{bmatrix} G \\ G_C \end{bmatrix} \cdot \Delta\underline{\hat{x}} = \begin{bmatrix} H^T \cdot R^{-1} \cdot \Delta\underline{z} \\ C^T \cdot \Delta\underline{b}_p \end{bmatrix} \tag{4.66}$$

onde

$$Gc = C^T \cdot C \quad e \quad C = \frac{\partial c_p(\underline{\hat{x}})}{\partial \underline{\hat{x}}} \tag{4.67}$$

Este mesmo desenvolvimento pode ser aplicado ao MQPV, entretanto deve-se desconsiderar a ponderação iterativa e o peso relacionado aos pontos de alavancamento relativos às equações que definem as restrições. Da seguinte forma:

$$\underline{w}_{aug}^T = \begin{bmatrix} w_1 & \dots & w_m & 1 & \dots & 1 \end{bmatrix} \quad e \quad \rho_{aug}^T(\underline{\tilde{r}}_S) = \begin{bmatrix} \rho(\tilde{r}_{S1}) & \dots & \rho(\tilde{r}_{Sm}) & 1 & \dots & 1 \end{bmatrix}$$

Logo sua aplicação ao problema EESP formulado pelo MQPV resulta em:

$$\min J(\underline{\hat{x}}) = \underline{w}_{aug}^T \cdot \rho_{aug}(\underline{\tilde{r}}_S) \cdot \underline{w}_{aug} \quad \rightarrow \quad \begin{bmatrix} G_Q \\ G_C \end{bmatrix} \cdot \Delta\underline{\hat{x}} = \begin{bmatrix} H^T \cdot R^{-1} \cdot Q(\underline{\tilde{r}}_S) \cdot \Delta\underline{z} \\ C^T \cdot \Delta\underline{b}_p \end{bmatrix} \tag{4.68}$$

A grande vantagem do método dos pesos é a sua simplicidade de implementação e seu resultado satisfatório para grande parte dos problemas. Contudo a ponderação excessiva causa mal-condicionamento numérico da matriz a ser fatorada na maioria das situações, impondo a necessidade de se utilizar de métodos de fatoração numericamente estáveis.

O método utilizado para tratar as restrições de igualdade neste projeto foi proposto por Gouvêa et alli [104] e consiste na aplicação do método dos pesos com refinamento iterativo proposto por Van Loan [103]. A grande vantagem deste método é a utilização de pesos compatíveis entre os resíduos das equações que definem as restrições e os demais resíduos na formulação do problema de EESP. O método consiste em ponderar moderadamente os resíduos das equações que definem restrições e realizar um processo iterativo de refinamento das restrições a cada iteração do método de Newton. Para tanto, após a k-nésima iteração do laço externo (iteração do método de Newton do estimador de estados) e da definição do peso relativo às restrições (μ por volta de 1x10^6 a 1x10^9), deve solucionar o seguinte problema para o estimador formulado através do MQP:

$$\begin{bmatrix} G \\ G_C \end{bmatrix} \cdot \Delta\underline{\hat{x}} = \begin{bmatrix} \underline{0} \\ C^T \cdot \Delta\underline{b}_p^k \end{bmatrix} \tag{4.69}$$

ou o seguinte para o estimador de estados formulado pelo MQPV:

$$\begin{bmatrix} G_Q \\ G_C \end{bmatrix} \cdot \Delta\underline{\hat{x}} = \begin{bmatrix} \underline{0} \\ C^T \cdot \Delta\underline{b}_p^k \end{bmatrix} \tag{4.70}$$

onde

$$\Delta\underline{b}_p^k = \underline{b}_p^k - C \cdot \underline{\hat{x}}_{\text{int}}^k \tag{4.71}$$

Para tanto se deve iniciar o laço interno (processo iterativo de refinamento) com $\underline{x}_{\text{int}}^k = \underline{x}^k$ e a cada iteração atualizar o vetor de estados interno ao processo $\underline{x}_{\text{int}}^{k+1} = \underline{x}_{\text{int}}^k + \Delta\underline{x}$. A convergência se dará quando:

$$\left\| \underline{b}_p^k - C \cdot \underline{\hat{x}}_{\text{int}}^k \right\| \leq \xi \cdot \|C\|_\infty \cdot \left\| \underline{\hat{x}}_{\text{int}}^k \right\| \tag{4.72}$$

onde ξ é a tolerância do laço interno, podendo-se adotar 1x10^{-2} ou 1x10^{-3}.

Trata-se de um método eficiente e que soluciona o problema do mal-condicionamento numérico da matriz a ser fatorada. Ressalta-se que a matriz fatorada no laço externo é utilizada no laço interno, bastando atualizar o lado direito do sistema de equações. Neste caso, podem ser utilizados as rotações armazenadas ou o procedimento de substituição direta e inversa da fatoração LU (mesmo não sendo este o método de fatoração, ver 4.6.3.4).

4.3.6.7 A estimação paramétrica com EESP baseada em MQPV

A inclusão da posição do TAP de transformadores com comutação sobre carga (LTC) como variável de estado na EESP baseada em MQPV mostrou gerar diversos problemas de

convergência durante a fase de validação das classes e do algoritmo por elas implementado. A estimação de parâmetros considerados sensíveis através de estimadores baseados em MQPV, quando tratados de forma análoga à EESP baseada em MQP [93], tornava o método instável e praticamente inviável por não alcançar a convergência mesmo em sistema com elevada redundância. Os estudos constataram que a estimação de parâmetros, quando os parâmetros são considerados sensíveis, ou seja, quando produzirem consideráveis variações nas estimativas de grandezas diretamente relacionadas frente a pequenas variações dos parâmetros, produzia elevados resíduos nas medidas dependentes destes parâmetros durante as primeiras iterações. Tal fato gerava a exclusão da maioria das medidas dependentes destes parâmetros devido aos baixos pesos resultante da penalidade imposta pelo método de MQPV.

Nos estimadores baseados em MQP, a estimação paramétrica é feita de forma direta, ou seja, consideram-se os parâmetros a serem estimados como variáveis de estados e calcula-se sua contribuição no desenvolvimento matemático do problema. A única modificação gerada é a inclusão destes parâmetros no vetor de estados ($\underline{\hat{x}}$), sendo as demais alterações – como o cálculo da matriz Jacobiana ($H(\underline{\hat{x}}) = \partial h(\underline{\hat{x}}) / \partial \underline{\hat{x}}$) – decorrências diretas desta inclusão. A única ressalva feita recai sobre a redundância do plano de medição, uma vez que a estimação de novos estados depende da disponibilidade local de medidas. Por prudência, pode-se adicionar *pseudomedidas* associadas aos parâmetros a serem estimados, contudo sem qualquer contribuição qualitativa à estimação.

A escassez de relatos na literatura técnica para o problema de estimação paramétrica na EESP baseada em critérios não-quadráticos resultou na necessidade de se desenvolver uma metodologia eficiente para tratar adequadamente o problema. Como a estimação de parâmetros sensíveis neste trabalho se limita à posição do TAP dos LTC, todos os esforços se voltaram a este problema. Com base nas informações obtidas dos resultados dos testes realizados em diversos sistemas, observou-se que:

- Os resíduos associados às medidas dependentes da posição de TAP a ser estimada eram elevadas nas primeiras iterações devido ao valor inicial do TAP ser normalmente informado nas proximidades de sua posição real. Nas primeiras iterações, estando os demais estados relacionados às medidas (magnitude e ângulo das tensões nas barras adjacentes) relativamente distantes dos valores reais, os valores estimados das grandezas relacionadas ao TAP eram discrepantes quando comparados com os valores medidos. Uma forma de solucionar este problema é considerar a posição central, normalmente a nominal, como valor inicial do TAP no processo de estimação, reduzindo – desta forma – consideravelmente os resíduos iniciais;

- A modelagem da posição do TAP como variável de estado resultava na formação de *conjuntos críticos* de medidas nos locais do sistema com menor redundância de informação. Tais conjuntos associados com a sensibilidade de suas medidas ao valor do TAP geravam instabilidade no processo de convergência. Uma forma de solucionar este problema – considerando que o valor inicial do TAP deve ser central – é adicionar uma *pseudomedida* de TAP com valor igual ao valor inicial adotado para a variável de estado que representa o TAP;

- Mesmo com a redução significativa dos resíduos iniciais e a minimização do problema de formação de *conjuntos críticos*, os incrementos relativos ao TAP em cada iteração do processo de solução eram muito pequenos. Desta forma gerava-se uma resistência à convergência, transferindo o problema que inicialmente era restrito às iterações iniciais para as iterações subseqüentes. Alguns testes mostraram que a resistência a mudanças era gerada pelo desvio-padrão relativamente pequeno adotado para a *pseudomedida* de TAP que solucionava o problema de formação de *conjuntos críticos*. Sendo os resíduos da *pseudomedida* de TAP normalmente muito superiores aos resíduos das demais medidas relacionadas (fluxo de potência reativa e injeção de potência reativa em p.u.), sua influência no problema de minimização será sabidamente superior e o estimador de estados tenderá a manter a posição inicial do TAP para reduzir a influência de seu resíduo. Para solucionar este problema, deve-se adotar como desvio-padrão da *pseudomedida* de TAP um valor igual à maior variação entre as possíveis posições de TAP, ou seja, a diferença entre o TAP máximo e o TAP mínimo. Desta forma a ponderação de seu resíduo será minimizada, impondo que as informações associadas às demais medidas relacionadas à posição de TAP possuam maior influência.

Os resultados destas análises comportamentais do estimador de estado baseado em MQPV compõem um método eficiente para tratar a questão da estimação da posição de TAP, produzindo bons resultados para todos os sistemas que possuam redundância adequada de informações. Para tanto, a seguinte heurística deve ser adotada:

1. Considerar a posição central como valor inicial da variável de estado que representa a posição do TAP do LTC, independente de existir uma medida ou estimativa para a posição de TAP em questão;
2. Incluir uma *pseudomedida* relativa à posição de TAP a ser estimada, sendo seu valor igual ao valor inicial da variável de estado que a representa;
3. Impor que o desvio-padrão da pseudomedida criada seja igual à maior variação entre as possíveis posições de TAP, ou seja, a diferença entre o TAP máximo e o TAP mínimo.

Os resultados desta heurística serão mostrados no capítulo de resultados numéricos.

4.3.7 – A modelagem orientada a objetos de estimadores de estados para SEP

A elaboração de um estimador de estados que possibilite a aplicação dos métodos descritos na seção anterior ao modelo de SEP proposto neste trabalho de forma independente, flexível e que possibilite a rápida implementação de novos métodos deve ser baseada numa análise substancial dos requisitos e elementos que formam o problema de EESP.

A estimação de estados recai num problema de minimização de resíduos, onde cada resíduo está associado a uma medida e a um valor estimado para esta grandeza. Portanto deve haver compatibilidade entre as medidas tratadas pelo estimador de estados e as medidas que os componentes do modelo de SEP podem fornecer. Portanto definiu-se um tipo enumerado para representar o conjunto de medidas que podem ser trabalhadas pelo estimador de estados e que devem estar implementadas no modelo de SEP. Tais medidas devem ser consideradas por todos os componentes do modelo, principalmente pela classe TSCADA. Este tipo enumerado, definido como *TipoDeGrandeza*, foi assim definido:

TipoDeGrandeza {MV, MP, MQ, Mtd, Mtp, Mud, Mup, Mcd, Mcp, Mtap, Mtemp}

onde:

- **MV:** Medida de magnitude de tensão;
- **MP:** Medida de injeção de potência ativa;
- **MQ:** Medida de injeção de potência reativa;
- **Mtd:** Medida de fluxo de potência ativa da barra DE para barra PARA;
- **Mtp:** Medida de fluxo de potência ativa da barra PARA para a barra DE;
- **Mud:** Medida de fluxo de potência reativa da barra DE para a barra PARA;
- **Mup:** Medida de fluxo de potência reativa da barra PARA para a barra DE;
- **Mtap:** Medida de posição de TAP;
- **Mtemp:** Medida de temperatura.

O estimador de estados, por óbvio, deve conhecer quais os possíveis estados a serem estimados, ou seja, quais os estados que definem o modelo de SEP. Logo, faz-se necessária outra definição global. O tipo enumerado abaixo, denominado de *Estados*, define quais são os estados que podem ser estimados e que definem o modelo de SEP:

Estados {V, THETA, TAP};

onde V é a magnitude de tensão das barras; THETA é o ângulo de tensão das barras e TAP é a posição do TAP de transformadores com comutação sobre carga.

4.3.7.1 O uso da abstração na obtenção de informações sobre o sistema

O estimador de estados necessita da obtenção de informações próprias dos componentes do sistema tais como quais são os estados relacionados a uma determinada grandeza medida, qual é o valor estimado de uma determinada grandeza ou qual é a derivada de uma determinada grandeza em relação a um determinado estado. É comum que o estimador de estados trabalhe diretamente com os modelos de componentes do SEP e que conheça tanto os estados quanto as medidas relacionadas a cada componente. Tal prática gera enorme dificuldade de evolução do estimador, além de tornar o código dependente do modelo e muito mais complexo que o necessário.

O ideal é que o estimador se relacione exclusivamente com classes abstratas que possam fornecer tais informações independentemente do modelo do componente ou do conhecimento de quais são as medidas ou estados que envolvem este componente. Para tanto se faz uso das classes abstratas mostradas na Figura 39, mais especificamente da classe *TObjetoPWS*. Apesar de não mencionado anteriormente, tal classe possui funções virtuais abstratas que – através da definição de medidas e estados realizada pelos tipos enumerados mostrados acima – permitem fornecer as informações que o estimador de estados necessita. Tais funções são redefinidas nas classes descendentes e, portanto, sobrescrevem as funções definidas nesta classe. A Figura 71 mostra as referidas funções.

TObjetoPWS
//Informa sobre os estados do componente unsigned int NumEstados(void) { return FNumEstados; }; TVariavelDeEstado* Estado(unsigned int index) { return FEstado[index]; }; *//Informa sobre as grandezas* virtual double ValorDaGrandeza(TipoDeGrandeza _Grand) = 0; *//Informa sobre o valor base de uma grandeza* virtual double BaseDaGrandeza(TipoDeGrandeza _Grand) = 0; *//Informa sobre a relação grandezas X estados* virtual void EstadosRelacionados(TipoDeGrandeza _Grand, unsigned int& _Num, TVariavelDeEstado**& _Lista) = 0; *//Informa sobre a derivada de uma grandeza em relação a um estado* virtual double Derivada(TipoDeGrandeza Grand, TVariavelDeEstado* Estado) = 0;

Figura 71 - Funções da classe *TObjetoPWS* utilizadas pelo estimador de estados.

As funções *NumEstados()* e *Estado()* indicam a quantidade de variáveis de estados do componente do sistema e quais são os estados do componente. Ressalta que os estados são representados pela classe *TVariavelDeEstado*, a qual internamento utiliza as informações definidas pelo tipo *Estados*. A função ValorDaGrandeza() retorna o valor estimado da grandeza informada por parâmetro, para tanto utiliza os valores atuais dos estados dos componentes do

sistema. A função *EstadosRelacionados()* retorna, através de parâmetro, quais são os estados relacionados a uma determinada medida ou grandeza. Por fim, a função Derivada() retorna o valor da derivada da grandeza indicada por parâmetro em relação ao estado também indicado por parâmetro.

Com base nas informações acima, pode-se notar que para incluir um novo estado ou um novo tipo de medida no problema de estimação de estados, basta que o estado ou a medida seja adicionada aos respectivos tipos definidos e que o modelo o modelo do componente do componente seja atualizado. Nenhuma alteração será necessária no código de estimação de estados ou em suas classes.

Por fim, cabe ressaltar que esta técnica de abstração pode ser utilizada em modelos de SEP que utilizam o padrão de projeto *Adapter,* tais como [16-18], uma vez que o objeto que define a estrutura da classe em relação aos estados de determinado estudo pode ser hierarquizado da forma apresentada neste trabalho para fins de estimação de estados.

4.3.7.2 O conceito de Elemento de Estimação

O estimador de estados baseia-se nas medidas e nas restrições de igualdade impostas ao problema de estimação, as quais se relacionam a um conjunto de outras propriedades – tais como desvios-padrões ou ponderações, o componente ou a condição do sistema que originou a informação, o tipo de informação, dentre outras características próprias. Entretanto, independente de se tratar de uma medida ou restrição, este conjunto de informações relacionadas possuem características em comum que, quando analisadas conjuntamente, podem ser consideradas como os elementos do estimador de estados.

Para tornar a idéia mais compreensível, considere a exclusivamente as medidas dentro do problema de estimação de estados. A partir do conhecimento de determinada medida e seu respectivo valor estimado, pode-se determinar o resíduo. Cada resíduo está associado a uma ponderação, seja ela oriunda das características do medidor ou oriunda de uma função estatística. A estimação do valor desta medida está relacionada a um conjunto de estados, os quais – por sua vez – dão informações sobre quais derivadas devem ser obtidas para se determinar a linha da matriz Jacobiana associada à solução do problema. A construção desta linha da matriz Jacobiana depende do conhecimento do componente do sistema (TObjetoPWS) e será feita com o uso de um objeto da classe *Matriz*. Portanto o conjunto de informações relacionadas a uma determinada medida capaz de definir todas suas características próprias para o problema de estimação de estado é definido como elemento de estimação. Tais

considerações também podem ser feitas para o caso de restrições de igualdade. Portanto a classe *TEE* foi desenvolvida para implementar esse conceito de forma abstrata, ou seja, não estando relacionada a nenhum método de estimação de estado. A Figura 72 mostra o seu diagrama de classes.

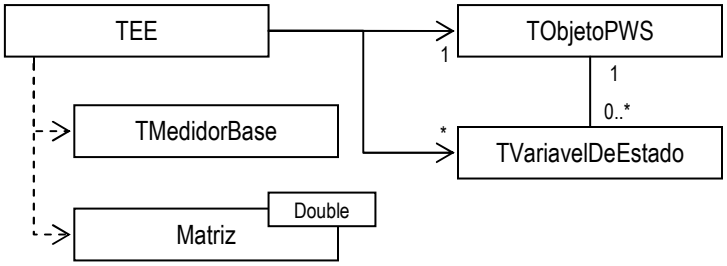


Figura 72 – Diagrama de classes de *TEE*

A classe TEE é responsável por implementar todas as funcionalidades relativas às medidas e às restrições de igualdade; armazenar o valor medido ou o valor da condição restritiva e todas as suas propriedades relacionadas; montar a linha da matriz Jacobiana, armazenando os ponteiros para acesso direto aos respectivos elementos internos da matriz; e por realizar a separação de planos através de funções internas definidas através de conjuntos de tipos enumerados globais. Para tanto ela possui as seguintes propriedades mostradas na Figura 73.

TEE	
// Número de estados associados à grandeza ou restrição	
- unsigned int FNumEstados;	
//Os estados associados à grandeza ou restrição	
- TVariavelDeEstado** FEstado;	
///O objeto (componente de rede) de origem	
- TObjetoPWS* FObjeto;	
//A lista de ponteiros que associam ao jacobiano	
- double** FJacob;	
//Informações Gerais	
- TipoDeGrandeza FGrandeza;	//Grandeza a qual se refere o elemento
- double FMedida;	//Medida ou estimativa do elemento (PU)
- double FPrecisao;	//Precisão da medida ou estimativa (PU)
- unsigned int FLinha;	//A sua posição no jacobiano
- bool FRestrito;	//Se o elemento refere-se a uma restrição
- bool FEG;	//Se este foi considerado como Erro Grosseiro
- bool FFiltrado;	//Se este foi alterado pelo algoritmo de pré-filtragem
- bool FFalha;	//Se a medida obtida era considerada “falhada”

Figura 73 – Propriedades internas da classe *TEE*

As propriedades *FNumEstados* e *FEstado* representam a quantidade e os estados relacionados a uma determinada medida ou restrição de igualdade, sendo estas informações obtidas diretamente do componente do sistema (ou componente de rede) a qual a medida ou a restrição de igualdade esta associada. Para tanto, a classe *TEE* faz uso das funções abstratas da classe *TObjetoPWS*. O componente que a medida ou restrição pertence é definido durante a criação da instância da classe e armazenado na propriedade *FObjeto.*

A classe *TEE* utiliza as informações sobre o conjunto de estados e as funções abstratas do componente do sistema para criar a linha da matriz Jacobiana relacionada à medida ou à restrição. Durante a criação, ela armazena os ponteiros de memória na lista *FJacob*, possibilitando o acesso direto aos elementos internos matriz. Tal prática possibilita atualizar a matriz Jacobiana independentemente da ordem resultante dos métodos de ordenação, bem como gera grande economia computacional por não utilizar os procedimentos de busca da classe *Matriz*. Ressalta-se que essa prática só é possível devido à estrutura de dados da classe matricial utilizada.

Apesar de depender da classe *TMedidorBase*, a classe *TEE* não utiliza as informações que poderiam ser obtidas através do acesso direto aos objetos que definem os medidores. Por estarem associados à classe *TSCADA* e, portanto, serem atualizados periodicamente, tais objetos poderiam mudar de valores durante o processo de estimação. Logo a classe *TEE* armazena as informações dos medidores em propriedades internas. Assim o tipo de medida, a precisão do medidor, o valor medido e o indicativo de falha são armazenados em *FGrandeza*, *FPrecisao*, *FMedida* e *FFalha*, respectivamente. Este último indica que a medida foi considerada "falhada" no último ciclo de medição, entretanto optou-se por sua utilização no processo de estimação com o uso de uma ponderação específica.

A propriedade *FLinha* indica qual a posição da linha da matriz Jacobiana que estão as informações geradas pelo elemento de estimação, sendo esta informação utilizada para fins de refinamento da solução. A propriedade *FRestrito* indica se o elemento de estimação é uma medida ou uma restrição de igualdade. A propriedade *FEG* indica se a medida processada esta contaminada com erros considerados grosseiros. Por fim, a propriedade *FFiltrado* indica se a medida a ser processada foi alterada pela funções de pré-filtragem (as quais não serão comentadas neste trabalho). Estas últimas propriedades são definidas pelo objeto que faz a estimação de estados, o qual será detalhado posteriormente.

Conforme mostra a Figura 74, os procedimentos e as funções da classe TEE são variados e relativos às funcionalidades que este tipo de elemento deve possuir. Dentre o grupo que corresponde a cálculos relacionados ao processo de estimação encontram-se:

1. *GetResiduo()*: Retorna o resíduo do elemento de estimação;
2. *GetPesoEstimacao()*: Retorna o peso utilizado durante a estimação de estados e corresponde o inverso do quadrado do valor calculado de desvio-padrão. Tal valor é obtido através da função *GetSigma()*;
3. *GetSigma()*: Retorna o valor calculado para o desvio padrão através da seguinte equação: $\sigma = \sqrt{(Fmedida \times Fprecisao)^2 + Fprecisao^2}$;

4. *GetResiduoPadronizado()*: Retorna o valor do resíduo padronizado;
5. *GetBase()*: Retorna o valor base da grandeza representada pelo elemento de estimação;

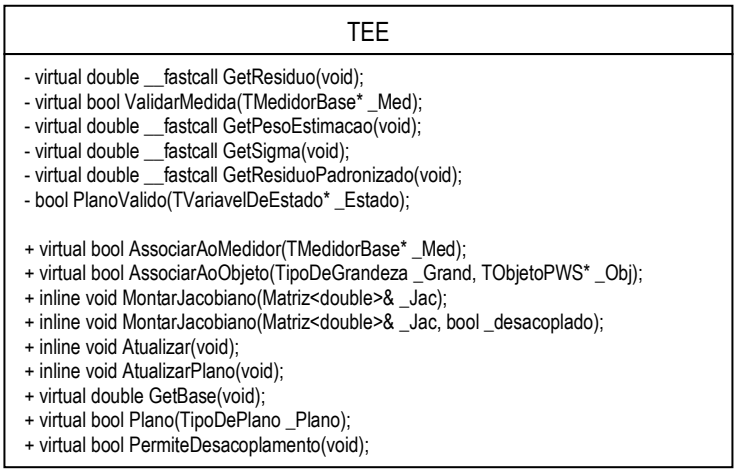


Figura 74 – Funções e procedimentos privados e públicos da classe *TEE*

Dentre as funções que pertencem de validação, têm-se:

1. *ValidarMedida()*: Indica se a medida informada pode ou não ser utilizada pelo elemento de estimação, sendo utilizada pela função *AssociarAoMedidor()*;
2. *PlanoValido()*: Indica se determinado estado pertence ao plano da grandeza representada pelo elemento de estimação;
3. *Plano()*: Indica se determinado plano corresponde ao plano da medida ou restrição. A verificação dos planos é feita através de funções globais que utilizam estruturas e tipo enumerados que relacionam as medidas e os estados aos planos;
4. *PermiteDesacoplamento()*: Informa se o elemento de estimação pode ou não ser desacoplado, ou seja, se ele possui grandezas pertencentes a mais de um plano;

Dentre as funções de associação estão:

1. *AssociarAoMedidor()*:Cria o vínculo do elemento de estimação com as informações de determinado medidor. Para tanto o medidor deve pertencer ao componente de rede informado através de *AssociarAoObjeto()*;
2. *AssociarAoObjeto():* Cria o vínculo entre o componente de rede e a grandeza que será representada com o elemento de estimação;

Dentre as funções relativas à montagem da matriz Jacobiana, estão:

1. *MontarJacobiano()*: Cria a linha da matriz jacobiana e insere-a na matriz informado como parâmetro conforme a posição indicada em *FLinha*. Caso se indique que o desacoplamento de planos, a linha criada corresponderá ao plano da grandeza medida;

2. *Atualiza()*: Atualiza os valores da linha da matriz jacobiana através de acesso direto aos elementos criados anteriormente;
3. *AtualizarPlano()*: Verifica e atualiza o plano do elemento em caso de alterações na estrutura global que relaciona os estados e grandezas aos planos;

A classe *TEE* não pode ser aplicada diretamente à estimação de estados. Trata-se de uma classe abstrata que define o conjunto comum de propriedades dos diversos tipos de estimadores. Caso existam divergências entre sua estrutura e a estrutura final que o elemento de estimação de um novo tipo de estimador de estados deva apresentar, o TEE poderá ser redefinido – uma vez que a maioria de suas funções são virtuais.

4.3.7.3 A classe abstrata para a estimação de estados

A classe abstrata *TPSSE* tem o objetivo de estruturar o problema de estimação de estados e organizar seu processo de solução, sendo responsável por:

- Obter o conjunto de elementos de estimação e o conjunto de variáveis de estado através da análise dos componentes e das condições da ilha operativa (*TRedeBase*);
- Realizar operações que envolvam o conjunto de elementos de estimação, tais como a montagem da matriz Jacobiana, a ordenação matricial e a separação de planos;
- Definir o conjunto comum de propriedades de controle e de configuração do processo de estimação, tais como o número máximo de iterações, a tolerância mínima para a convergência, a forma de iniciação das variáveis de estado, se a haverá desacoplamento de planos, se haverá processamento de restrições de igualdade, se a solução deverá ser refinada, dentre outras.
- Definir o conjunto de métodos a serem aplicados na solução do problema, ou seja, o método de ordenação de linhas, o método de ordenação de colunas, a método de solução do sistema de linear e o método de fatoração;
- Organizar as estruturas internas de dados conforme o tipo de estimador a ser utilizado, ou seja, acoplado ou desacoplado;
- Verificar se o problema de estimação de estados é solucionável, indicando as possíveis variáveis de estados não-observáveis.

As classes descendente de *TPSSE*, conforme será visto nas subseções adiante, utilizarão a estrutura herdada para implementar o algoritmo de estimação de estados, restringindo-se a adicionar as propriedades específicas do estimador de estados a ser desenvolvido. Portanto a classe *TPSSE* pode ser concebida como uma classe organizacional que trabalha com o modelo

de SEP proposto para disponibilizar um ambiente onde futuras especializações (classes descendentes) se dediquem somente ao algoritmo de estimação de estados.

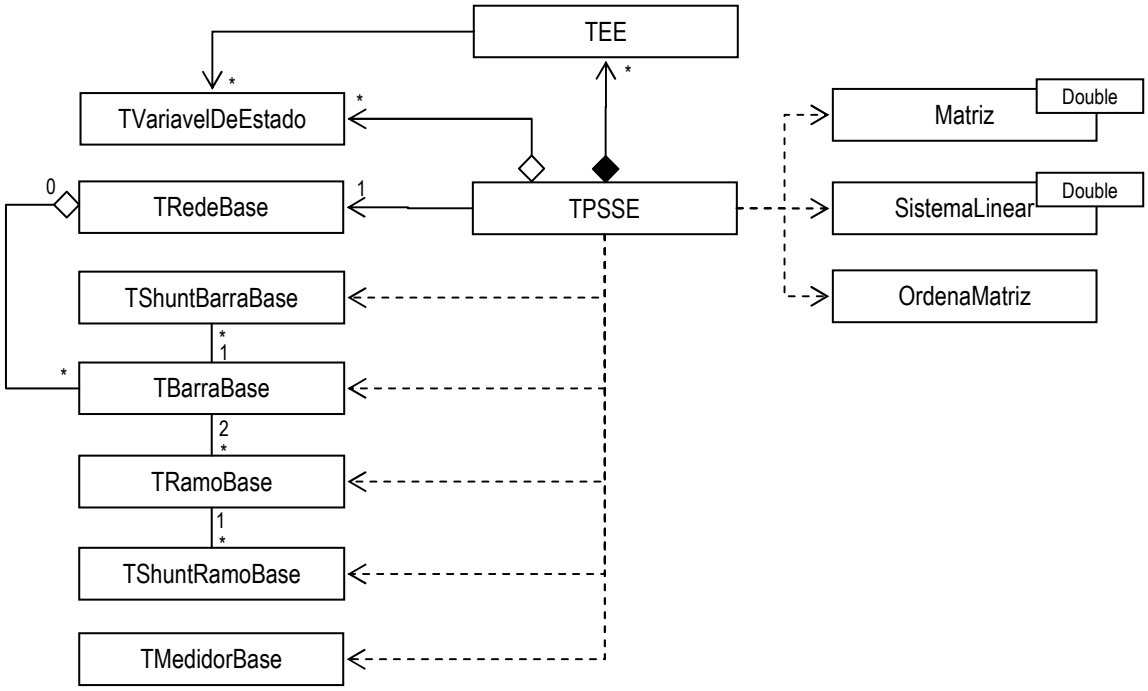


Figura 75 – Diagrama de classes de *TPSSE*

Conforme mostra a Figura 75, a classe *TPSSE* é composta por um grupo de elementos de estimação e de variáveis de estados. Todas as informações sobre o sistema elétrico são obtidas através da ilha operativa, ou seja, o conjunto de variáveis de estado e de elementos de estimações são formados através de um processo de varredura nos elementos que compõem esta ilha (*TRedeBase*). Nota-se que não há a necessidade do EE conhecer as classes mais específicas do modelo de SEP, bastando conhecer as classes que formam seu núcleo abstrato. Por sua vez, os relacionamentos com as classes oriundas do pacote de ferramentas matemáticas se limitam ao desenvolvimento das rotinas de ordenação, montagem da matriz jacobiana e análise da possibilidade de solução do sistema linear gerado pelo processo de solução.

Conforme mostra a Figura 76, a classe *TPSSE* possui um extenso conjunto de propriedades internas que pode ser agrupadas por finalidade. As propriedades que representam as informações relativas à ilha operativa são:

1. *FRede*: Objeto que representa a ilha operativa;
2. *FNumEEs e FEE*: Número e lista de elementos de estimação para o estimador acoplado;
3. *FNumEstados e FVarEstado*: Número de lista de variáveis de estados para o estimador acoplado;

4. *FNumEEs_PD e FEE_PD*: Número e lista de elementos de estimação do plano P-δ;
5. *FNumEEs_QV e FEE_QV*: Número e lista de elementos de estimação do plano Q-V;
6. *FNumEstados_PD e FEstado_PD*: Número e lista de variáveis de estado do plano P-δ;
7. *FNumEstados_QV e FEstado_QV*: Número e lista de variáveis de estado do plano P-δ;

Apesar de haver duplicidade de listas, as informações sobre o conjunto de elementos de estimação e variáveis de estado ou são armazenadas em lista única ou são armazenadas em listas separadas, dependendo do tipo de estimador utilizado.

TPSSE

```
- TRedeBase* FRede;

- unsigned int FNumEEs;
- TEE** FEE;
- unsigned int FNumEstados;
- TVariavelDeEstado **FVarEstado;

- unsigned int FNumEEs_PD;
- TEE** FEE_PD;
- unsigned int FNumEEs_QV;
- TEE** FEE_QV;
- unsigned int FNumEstados_PD;
- TVariavelDeEstado** FEstado_PD;
- unsigned int FNumEstados_QV;
- TVariavelDeEstado** FEstado_QV;

- unsigned int FNumIteracoes;          //Número máximo de Iterações
- unsigned int FIteracaoJacCTE;        //Iteração a partir da qual o a matriz Jacobiana será constante
- unsigned int FNumMaxIteracoes;       //Número máximo de iterações
- double FMaxErroAdmitido;             //Tolerância do laço externo
- double* FSPQR;                       //Valores da soma ponderada dos resíduos por iteração
- bool FConvergido;                    //Indica a convergência do processo
- bool FFlatStart;                     //Indica se as variáveis de estados serão iniciadas com perfil plano de tensão
- bool FAdmiteDesacoplamento;          //Indica se há possibilidade de se desacoplar os planos
- bool FDesacoplado;                   //Indica se o estimador será desacoplado ou acoplado
- bool FRestringir;                    //Indica se as restrições de igualdade devem ser processadas

- bool FMeiaIterPDelta;                //Indica se antes de se iniciar o processo iterativo deve realizar uma meia iteração Pδ
- bool FRefinar;                       //Indica se a solução do estimador deverá se refinada

- TMetodoOrdLin FMetOrdLin;            //Método de ordenação de linhas a ser aplicado
- TMetodoOrdCol FMetOrdCol;            //Método de ordenação de coluna a ser aplicado
- TMetodoDeSolucao FSolMet;            //Método de solução do sistema linear – Equação normal ou Fatoração direta
- TMetodoDeFatoracao FFatMet;          //Método de fatoração a ser aplicado

- double FP_Inj, FP_Cons;              //Balanço de potência ativa – Quantidade injetada x Quantidade consumida
- double FQ_Inj, FQ_Cons;              //Balanço de potência reativa – Quantidade injetada x Quantidade consumida

- unsigned int FNumMedidasErroneas;    //Número de medidas contaminadas com erro grosseiro
- TEE **FEEMedErronea;                 //Lista de medidas (elemento de estimação) contaminadas com erro grosseiro
```

Figura 76 – Propriedades internas da classe PSSE

A finalidade da demais propriedades internas pode ser obtida diretamente da Figura 76, não necessitando de uma explicação mais detalhada.

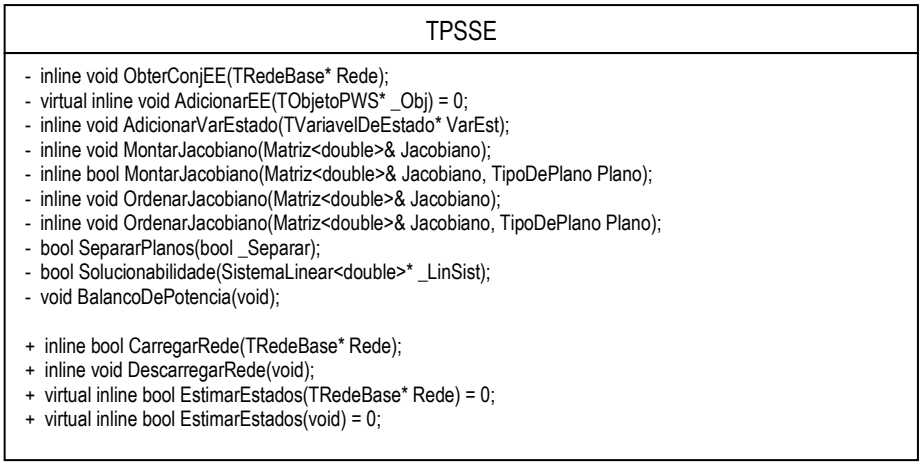


Figura 77 – Principais funções e procedimentos privados e públicos da classe *TPSSE*

A Figura 77 mostra as principais funções implementadas na classe *TPSSE*. As funções relacionadas ao processo de montagem da estrutura do problema de estimação são:

1. *CarregarRede():* Libera a informação existente nas estruturas de dados e inicia o processo de varredura das informação da ilha operativa;
2. *DescarregarRede():* Libera a memória alocada nas estruturas de dados;
3. *ObterConjEE():* Realiza uma varredura na ilha operativa para obter o conjunto de medidores associados a cada componente. Posteriormente analisa as condições físico-operativas para determinar as injeções nulas de potência e as demais restrições de igualdade;
4. *AdicionarEE():* Trata-se de uma função virtual-abstrata que deve ser implementada nas classes descendentes com o objetivo de criar o elemento de estimação relativo ao método empregado.
5. *AdicionarVarEstado()*: Adiciona a variável de estado à lista interna, caso necessário;

As funções e procedimento relativos ao sistema linear são:

1. *MontarJacobiano():* Monta a matriz Jacobiana através da varredura da lista de elementos de estimação. Cada elemento de estimação adicionará uma linha à matriz passada como parâmetro. Caso necessário, pode-se montar a matriz relativa a um determinado plano, bastando informar o plano desejado;
2. *OrdenarJacobiano():* Aplica os métodos de ordenação de linhas e colunas na matriz jacobiana, ordenando posteriormente as listas de elementos de estimação e variáveis de estado conforme o resultado obtido;

3. *Solucionabilidade():* Analisa o sistema linear que forma a solução do problema de estimação de estados e verifica se este possui solução única, indicando as possíveis variáveis de estado não observáveis;

As demais funções são:

1. *SepararPlanos()*: Separa ou acopla os planos P-δ ou QV conforme solicitado, realizando a transição dos elementos de estimação e das variáveis de estado da lista completa para as listas relativas aos planos, ou vice-versa;
2. *BalancoDePotencia()*: Realiza o balanço de potência da ilha operativa através da soma das potências ativas e reativas de cada barra injetadas ou consumidas, conforme o caso.
3. *EstimarEstados()*: Trata-se da função abstrata e virtual que deverá ser sobrescrita nas classes dependentes para implementar o algoritmo do método de estimação a qual a classe se refere;

Cabe ressaltar que o conjunto de medidores obtidos através da varredura da ilha operativa pode, sem nenhum prejuízo e com pouca alteração no código, ser repassado diretamente para a classe *TPSSE*. Neste caso os medidores deverão indicar a qual componente de rede pertence a grandeza mensurada. Entretanto deve-se informar a ilha operativa para que a classe *TPSSE* possa obter o conjunto de restrições de igualdade.

4.3.7.4 A especialização das classes de estimação de estados

As subseções anteriores apresentaram as classes abstratas que formam as bases necessárias à implementação de diferentes métodos de estimação de estados. A partir das classes TEE e TPSSE serão especializadas classes voltadas aos métodos de estimação de estados baseados em MQP e MQPV já comentados neste trabalho, conforme mostra a Figura 78. Cabe ressalta que as classes ditas “clássicas” implementam o algoritmo do estimador de estados baseados em MQP. Por sua vez, as classes ditas “robustas” referem-se ao estimador de estados baseado em MQPV. O relacionamento entre os pares de classes derivadas de TEE e TPSSE, os quais formam o conjunto do método a ser implementado, limita-se à utilização – uma vez que a classes derivada TPSSE somente necessita conhecer a correspondente classe TEE para a redefinição da função de adição (*AdicionarEE)* e para utilização interna na rotina de estimação.

Inicialmente serão apresentados os detalhes das classes TEEClassico e TPSSEClassico, incluindo fragmentos de código. Posteriormente serão apresentados os detalhes das classes TEERobusto e TPSSERobusto utilizando-se o mesmo padrão.

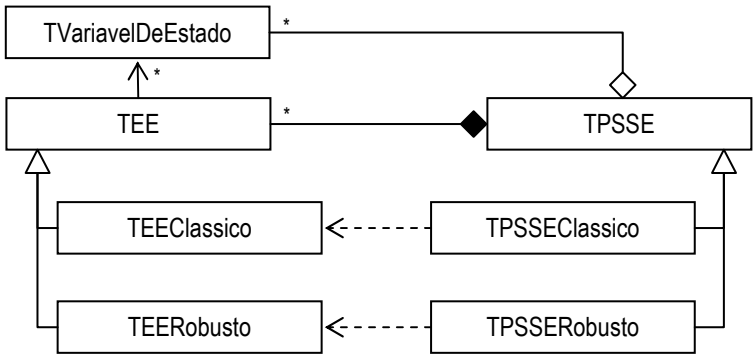


Figura 78 – Especialização das classes de estimação de estados.

4.3.7.5 Implementação do estimador de estados baseado em MQP

As classes abstratas praticamente disponibilizam todos os requisitos necessários à implementação do estimador de estados baseado em MQP, restando somente aqueles relativos à identificação e tratamento de erros grosseiros e à inclusão das restrições de igualdade.

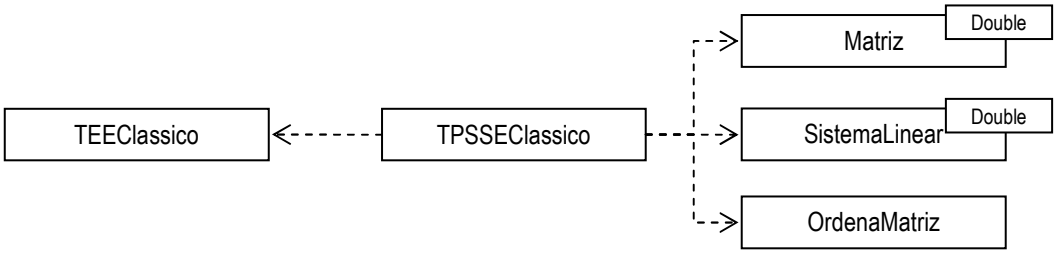


Figura 79 – Diagrama de classes do estimador de estados baseado em MQP

O tratamento de erros grosseiros deve possibilitar que a medida contaminada seja ou excluída do problema ou recuperada através da estimativa da magnitude de seu erro. A identificação de medidas com erros grosseiros requer que a diagonal da matriz de covariância dos resíduos de estimação seja conhecida. Por sua vez, a estimativa da magnitude do erro grosseiro requer que o elemento da diagonal principal da matriz de sensibilidade relativo à medida identificada seja também conhecido. Baseando-se em (4.44) e (4.48), o conhecimento dos elementos diagonais da matriz sensibilidade seria suficiente – uma vez que a covariância dos erros de medição já se encontra disponível na classe *TEE*. Uma informação importante e útil para o processo de estimação se torna produto desta escolha: a identificação de medidas críticas. Caso o valor do elemento da diagonal principal da matriz de sensibilidade relativo à determinada medida seja nulo, então a medida é considerada crítica e seu resíduo é nulo [93].

Com base nas informações comentadas acima, o elemento de estimação da abordagem clássica deve conhecer o valor do elemento da diagonal principal da matriz de sensibilidade,

possibilitando o cálculo do resíduo normalizado e da magnitude do erro incorporado à medida e sua identificação como medida crítica.

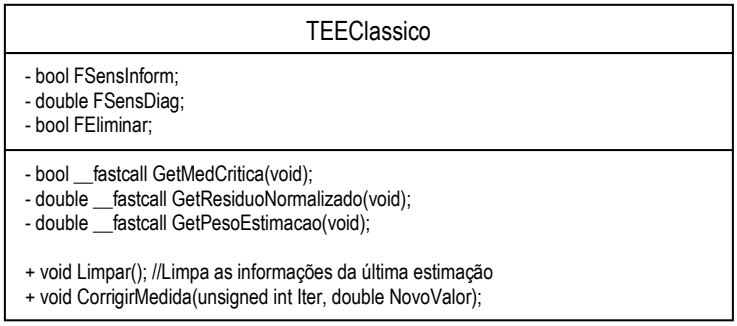


Figura 80 – Propriedades, funções e procedimentos adicionados na classe *TEEClassico*

A mostra o detalhamento da classe TEEClassico. As propriedades e funções incluídas ou alteradas foram:

1. *FSensInform*: Indica se o elemento da diagonal de S foi informado;
2. *FSensDiag*: Valor do elemento da diagonal principal da matriz de sensibilidade relativo à medida ou restrição representada pelo elemento de estimação;
3. *FEliminar*: Indica se haverá correção ou eliminação da medida caso esta seja identificada sua contaminação por erro grosseiro;
4. *GetMedCritica()*: Informa se a medida ou restrição de igualdade que esta sendo representada pelo elemento de estimação é considerada crítica;
5. *GetResiduoNormalizado()*: Calcula o valor do resíduo normalizado, conforme (4.49);
6. *GetPesoEstimacao()*: Sobrescreve a função relativa à ponderação dos resíduos da classe TEE, possibilitando a eliminação da medida através da atribuição de um peso próximo do *flag numérico*[2]*,* caso a medida seja identificada como errônea e se opte por eliminá-la do processo de estimação;
7. *Limpar()*: Atribui os valores padrões às variáveis internas;
8. *CorrigirMedida()*: Corrige o valor medido conforme (4.50) e (4.51), caso seja necessário;

A classe TPSSEClassico, por sua vez, deve implementar os algoritmos de tratamento de erros grosseiros e de restrições de igualdade, além do próprio algoritmo do estimador de estados baseados em MQP.

No caso de tratamento de erros grosseiros, o algoritmo deverá calcular a matriz de sensibilidade e a selecionar do maior resíduo normalizado. Apesar do cálculo da referida matriz ser computacionalmente oneroso, este não precisa ser refeito a cada nova iteração – uma vez

[2] *Flag Numérico* é o limite para que um determinado valor seja considerado zero. Neste projeto todos os valores inferiores a $1X10^{-12}$ são considerados nulos.

que os valores da diagonal principal pouco mudam durante o processo de solução do problema, mesmo com a eliminação e recuperação de medidas. Assim, após a iteração escolhida para início da detecção de erros grosseiros, a matriz de sensibilidade é calcula e seus valores diagonais serão repassados aos respectivos elementos de estimação.

Apesar da aparente dificuldade, o processamento das restrições de igualdade através do método dos pesos com refinamento iterativo simplifica-se com o uso dos recursos da classe *SistemaLinear*. Tal classe possibilita que, a partir de uma solução já encontrada para determinado sistema linear, novas soluções sejam obtidas para diferentes vetores independentes. Ressalta-se, mais uma vez, que este processo independe do método de fatoração escolhido, desde que se tenha uma solução pretérita. Assim o método apresentado em 4.3.6.7 resume-se em determinar o lado direito do sistema de equações e testar a convergência do processo, uma vez que as informações contidas nos elementos de estimação e as funcionalidades das classes matemáticas simplificam demasiadamente o processo.

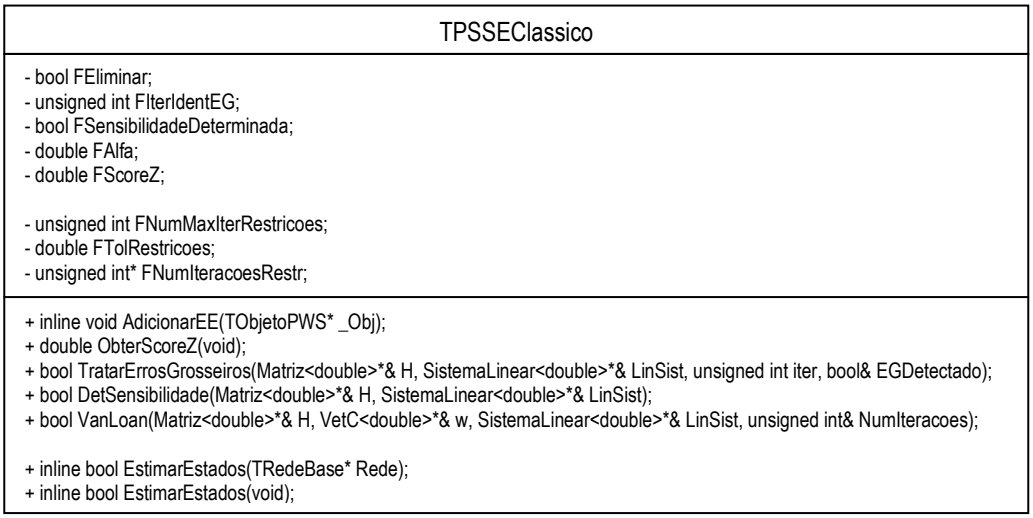


Figura 81 – Propriedades, funções e procedimentos adicionados ou sobrescritos na classe *TPSSEClassico*

As propriedades adicionadas na classe TPSSEClassico, conforme Figura 81, são:

1. *FEliminar*: Indica se a medida com erro grosseiro será eliminada ou recuperada;
2. *IterIdentEG*: Iteração a partir da qual se inicia o processo de detecção de erros grosseiros;
3. *FSensibilidadeDeterminanda*: Indica se a matriz de sensibilidade foi calculada;
4. *FAlfa*: Probabilidade de falso alarme ou de se rejeitar indicar erros grosseiros quando de fato eles não existem;
5. *FScoreZ*: Valor da distribuição Qui-Quadrado com grau de liberdade igual à diferença entre o número de elementos de estimação válidos e o número de estados, sendo a probabilidade de falso alarme igual a FAlfa;

```
inline bool TPSSEClassico::EstimarEstados(TRedeBase* Rede)
{ ObterConjEE(Rede);          //Obtendo o conjunto de EE e de Variáveis de Estado da Rede

  //Variáveis do processo de estimação
  SistemaLinear<double> LinSist();          //Sistema Linear
  Matriz<double> H(0,0);                    //Matriz Jacobiana
  VetC<double> r(FNumEEs);                  //Vetor de Resíduos
  VetC<double> w(FNumEEs);                  //Vetor de Pesos

  unsigned int Iteracao = 0;          //Número da interação
  double EMax;                        //Máximo erro (variação dos estados)
  bool ComEG;                         //Flag indicativo de Erro Grosseiro
  FConvergido = false;                //Flag de convergência

  LinSist.SetMetSolucao(FSolMet);           //Configurando o método de solução
  LinSist.SetMetFatoracao(FFatMet);         //Configurando o método de Fatoração

  MontarJacobiano(H);                 //Montando a matriz Jacobiana
  OrdenarJacobiano(H);                //Ordenando linhas e colunas

  do
  { Iteracao++;
    //Verificando a necessidade de fatorar o Jacobiano
    if (Iteracao < FIteracaoJacCTE)
    { for (int i = 0; i < FNumEEs; i++)
      {  r(i) = FEE[i]->Residuo;            //Vetor de resíduos
         w(i) = FEE[i]->PesoEstimacao;      //Vetor de ponderação
         FEE[i]->Atualizar();               //Atualizando linhas do Jacobiano
      }
    }
    else
    { for (int i = 0; i < FNumEEs; i++)
        r(i) = FEE[i]->Residuo;             //Vetor de resíduos
    }

    //Solucionando o problema, fatorando caso o jacobiano tenha sido atualizado
    if (LinSist.Solucionar(&H, &w, &r, (Iteracao < FIteracaoJacCTE)))
    { //Restrições de igualdade
      VanLoan(&H, &w, &LinSist, FNumIteracoesRestr[Iteração-1]);
      //Obtendo a SPQR
      FSPQR[Iteracao - 1] = LinSist.SPQR();
      //Determinando o maior incremento e atualizando os estados
      EMax = 0.00;
      for (unsigned int i = 0; i < FNumEstados; i++)
      { FVarEstado[i]->Incrementar(LinSist.Solucao()->GetData(i));

        if (fabs(LinSist.Solucao()->GetData(i)) > EMax)
           EMax = fabs(LinSist.Solucao()->GetData(i));
      }

      //Efetuando o tratamento de EG
      if (FIterIdentEG <= Iteracao)
          TratarErrosGrosseiros(&H, &LinSist, Iteracao, ComEG);

      //Testando a convergência do processo
      FConvergido = ((EMax < FMaxErroAdmitido)&&(!ComEG));
    }
    else
    { //Verificando as possíveis variáveis de estado não observáveis
      Solucionabilidade(LinSist, FVarEstado, Recompor);
      return false;
    }
  } while ((Iteracao < FNumMaxIteracoes)&&(!FConvergido));

  BalancoDePotencia();
  return true;
};
```

Figura 82 – Rotina de estimação de estados da classe *TPSSEClassico*.

6. *FNumMaxIterRestricoes*: Número máximo de iterações do laço interno;
7. *FTolRestricoes*: Fator de tolerância do critério de convergência do método dos pesos com refinamento iterativo, conforme 4.3.6.7;
8. *FNumIteracoesRestr*: Vetor que contém o número de iterações do método dos pesos com refinamento iterativo para cada iteração do processo de solução estimador de estados;

Por sua vez, as funções e procedimentos adicionados ou sobrescritos são:

1. *AdicionarEE()*: Sobrescreve a função virtual de TPSSE para criar elementos de estimação compatíveis com a classe TPSSEClassico, ou seja, objeto da classe TEEClassico;
2. *ObterScoreZ()*: Calcula *FScoreZ* a partir das informações da ilha operativa e da probabilidade de falso alarme;
3. *TrataErrosGrosseiros()*: Realiza a detecção, identificação e eliminação ou recuperação de medidas com erros grosseiros;
4. *DetSensibilidade()*: Calcula a diagonal da matriz de sensibilidade;
5. *VanLoan()*: Executa o procedimento descrito em 4.3.6.7;
6. *EstimarEstados()*: Sobrescreve a função abstrata e virtual da classe TPSSE inserindo o algoritmo de estimação de estado baseado em MQP.

A Figura 82 detalha a rotina de estimação de estados da classe TPSSEClassico. As propriedades referenciadas pelas classes mostradas no fragmento de código utilizam o nome publicado[3] e, portanto, divergem daquelas mostradas anteriormente.

Os resultados dos testes desta classe serão mostrados no próximo capítulo.

4.3.7.6 Implementação do estimador de estados baseado em MQPV

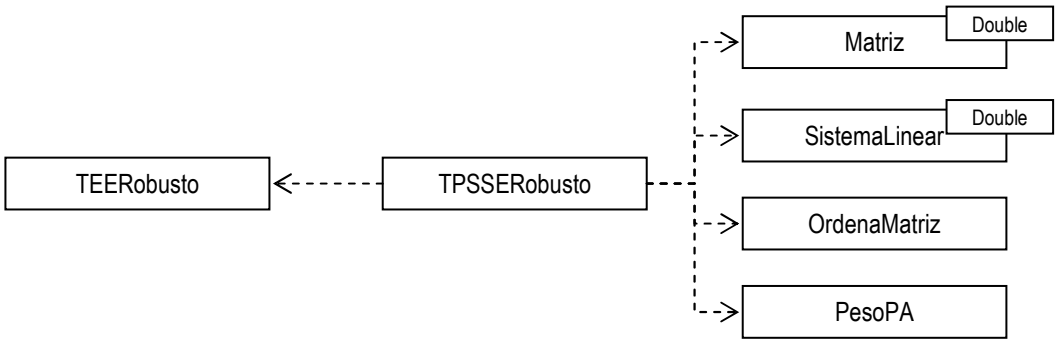


Figura 83 – Diagrama de classes do estimador de estados robusto baseado em MQPV

[3] Nome publicado ou propriedade externa é um artifício da linguagem Borland C++ Builder para possibilitar um mesmo acesso tanto para a leitura quanto para a função que modifica o valor da propriedade interna, as quais não foram mostradas por simplificação.

O estimador de estados robusto baseado em MQPV, implementado através das classes TEERobusto e TPSSERobusto, acrescenta às estruturas das classes antecessoras TEE e TPSEE, respectivamente, as propriedades referentes à ponderação variável dos resíduos, ao peso dos pontos de alavancamento e às restrições de igualdade.

O cálculo dos pesos dos pontos de alavancamento depende do conhecimento da matriz Jacobiana, logo deve ser realizado pela classe TPSSERobusto. Por sua vez, a determinação da ponderação variável ou da penalidade imposta às medidas com resíduos superiores ao limite de transição depende somente do estimador não-quadrático utilizado e do resíduo padronizado, sendo – portanto – realizado pela classe TEERobusto por ser uma característica própria do elemento de estimação.

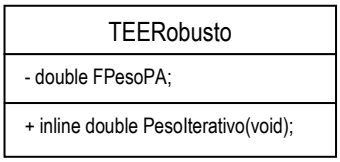


Figura 84 – Acréscimos necessários à classe *TEERobusto*

Conforme a Figura 84, a classe TEERobusto somente acrescenta à estrutura da classe antecessora uma propriedade para armazenar o valor do peso relativo ao ponto de alavancamento (*FPesoPA*) e uma função que determina o valor da penalidade do elemento ou seu peso variável (*PesoIterativo()*). Ressalta-se que este trabalho utilizou o estimador quadrático-tangente proposto por Huber, conforme mostrado e mencionado em 4.3.6.5.

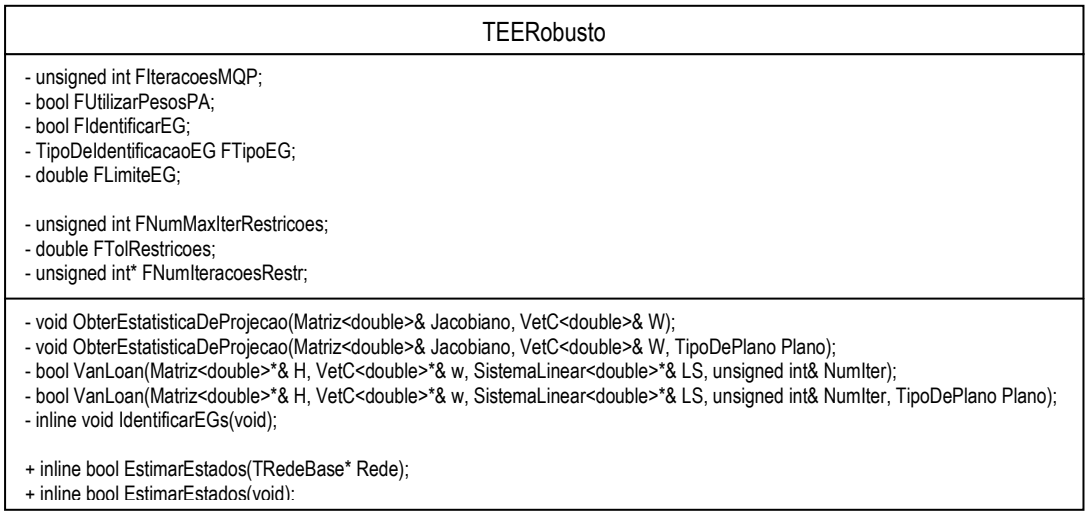


Figura 85 - Propriedades, funções e procedimentos adicionados ou sobrescritos na classe *TPSSEClassico*

A classe TPSSERobusto implementa tanto o estimador de estados comum quanto o estimador de estados com desacoplamento de planos, utilizando – para tanto – os recursos disponíveis na classe TPSSE.

O método de MQPV, se aplicado a partir da primeira iteração, poderá ocasionar a exclusão de uma quantidade significativa de medidas, dependendo dos valores iniciais dos estados da ilha operativa. Os resíduos da primeira iteração – quando o estimador inicia a ilha com o perfil plano de tensões – tendem a serem grandes e, consequentemente, a peso variável destas medidas será pequeno, levando a provável divergência do caso. Para mitigar este problema, realiza-se um número pré-definido de iterações sem considerar a ponderação variável das medidas, resultando na imposição o método MQP.

Uma particularidade do estimador de estados baseado em MQPV é a identificação de erros grosseiros, uma vez que nem todas as medidas penalizadas durante o processo iterativo estão contaminadas com esse tipo de erro. Uma das formas de se identificar a presença de erros grosseiros é através da análise dos valores dos resíduos padronizados após a última iteração. Caso tais resíduos possuam valor superior a determinado limite, normalmente 3 ou 4, o erro pode ser considerado grosseiro. Entretanto pode-se também analisar o valor do peso iterativo da última iteração. Neste caso deve ser determinado um limite que corresponda ao número máximo de desvios-padrões aceitáveis. No caso do estimador quadrático-tangente, valores de ponderação variável inferiores a 0,25 correspondem a resíduos com magnitude superior a 5 desvios-padrões. Para mais flexível o processo de identificação, ambas as técnicas foram desenvolvidas para classe TPSSERobusto – possibilitando, inclusive, a configuração de seus limites.

Os pontos de alavancamento em EESP normalmente são gerados por medidas pertencentes a componentes que apresentam parâmetros discrepantes em relação aos demais, tais como curtas linhas de transmissão, ou por medidas pertencentes ou adjacentes a barras que possuem um elevado número de ramos. Alguns sistemas, como o caso da CEEE, acabam formando uma grande quantidade de pontos de alavancamento pela forma com que são modelados. Considerar no modelo do sistema todos os geradores que formam um grupo de geração, ao invés de um único gerador equivalente, propicia a formação de pontos de alavancamento. Portanto, em determinados modelos, torna-se necessário que a ponderação de tais pontos seja inibida. Para tanto a classe TPSSERobusto proporcionada essa opção.

Conforme mostra a Figura 85, as propriedades adicionadas à estrutura da classe TPSSE necessárias ao desenvolvimento do estimador de estados robusto são:

1. *FIteracoesMQP*: Número de iterações iniciais sem se penalizar as medidas que apresentarem resíduos acima do limite de transição;
2. *FUtilizarPesosPA*: Indica se haverá ou não a identificação e o tratamento das medidas consideradas como pontos de alavancamento.

3. *FIdentificarEG*: Indica se haverá ou não a identificação de medidas com erros grosseiros;
4. *FTipoEG*: Indica a técnica de identificação a ser utilizada (RESIDUO_PADRONIZADO para identificação através do resíduo ou PESO para a identificação através da ponderação);
5. *FLimiteEG*: O valor limite relacionado a técnica de identificação de erros grosseiros;

As demais propriedades referem-se ao método dos pesos com refinamento iterativo e são iguais àquelas comentadas na classe TPSSEClassico. Por sua vez, as funções e os procedimentos adicionados e sobrescritos são:

1. *AdicionarEE()*: Sobrescreve a função virtual de TPSSE para criar elementos de estimação compatíveis com a classe TPSSERobusto, ou seja, objeto da classe TEERobusto;
2. *ObterEstatísticaDeProjeção()*: Identifica e calcula o peso dos pontos de alavancamento para a matriz Jacobiana ponderada, tanto para a o estimador completo quanto para o estimador desacoplado. Ressalta-se que a ponderação que deve ser repassa a esta função é somente o inverso da variância de cada medida.
3. *VanLoan()*: Executa o procedimento descrito em 4.3.6.7;
4. *IdentificarEG()*: Identifica os erros grosseiro ao término do processo de estimação, utilizando a técnica escolhida com o respectivo valor limiar;
5. *EstimarEstados()*: Sobrescreve a função abstrata e virtual da classe TPSSE inserindo o algoritmo de estimação de estado baseado em MQPV.

A Figura 86 exibe um fragmento de código da rotina de estimação relativo ao estimador de estados completo ou acoplado para melhor entendimento das propriedades, funções e procedimentos comentados nesta subseção. Os resultados utilizando ambos estimadores serão apresentados no próximo capítulo.

4.4 Considerações Finais

O presente capítulo procurou abordar o modelo de SEP, as ferramentas matemáticas, o modelo de estimador de estados e as demais classes que compõem o projeto de estimador robusto de estados desenvolvido para CEEE. Apesar de não apresentar todos os pacotes pertencentes à arquitetura básica do sistema, os principais pacotes foram comentados e, dentro do bom senso, foram detalhados. Procurou-se sempre apresentar as classes em conjunto com as respectivas teorias, mas sem a pretensão de ser exaustivo.

O próximo capítulo mostrará os resultados numéricos das ferramentas matemáticas e dos estimadores de estados desenvolvidos, incluindo os resultados obtidos no sistema de transmissão da CEEE.

```
inline bool TPSSERobusto::EstimarEstados(TRedeBase* Rede)
{ ObterConjEE(Rede);          //Obtendo o conjunto de EE e de Variáveis de Estado da Rede

  SistemaLinear<double> LinSist();          //Sistema Linear
  Matriz<double> H(0,0);                    //Matriz Jacobiana
  VetC<double> r(FNumEEs);                  //Vetor de Resíduos
  VetC<double> w(FNumEEs);                  //Vetor de Pesos
  unsigned int Iteracao = 0;                //Número da interação
  double EMax;                              //Máximo erro (variação dos estados)
  FConvergido = false;                      //Flag de convergência

  LinSist.SetMetSolucao(FSolMet);           //Configurando o método de solução
  LinSist.SetMetFatoracao(FFatMet);         //Configurando o método de Fatoração

  MontarJacobiano(H);                       //Montando a matriz Jacobiana
  OrdenarJacobiano(H);                      //Ordenando linhas e colunas

  if (FUtilizarPesosPA)                     //Calculando os pesos dos PAs
  { //Montando a matriz Jacobiana e o vetor de ponderação
    for (unsigned int i = 0; i < FNumEEs; i++)
    { w(i) = FEE[i]->PesoEstimacao;
      FEE[i]->AtualizarPlano();
    }
    ObterEstatisticaDeProjecao(H, w);       //Determinando os pesos PAs
  }

  do                                        //Início do processo iterativo
  { Iteracao++;

    if (Iteracao < FIteracaoJacCTE)         //Atualizando a Matriz Jacobiana
      for (unsigned int i = 0; i < FNumEEs; i++)
        FEE[i]->Atualizar();

    if (Iteracao > FIteracoesMQP)           //Vetor de resíduos e ponderação
      for (int i = 0; i < FNumEEs; i++)
      {  _TEERobusto *_EE = dynamic_cast<TEERobusto *>(FEE[i]);   //Cast
         r(i) = _EE->Residuo;                                     //Vetor de resíduos
         w(i) = _EE->PesoEstimacao*_EE->PesoIterativo();          //Ponderação MQPV
      }
    else
      for (int i = 0; i < FNumEEs; i++)
      {  r(i) = FEE[i]->Residuo;                                  //Vetor de resíduos
         w(i) = FEE[i]->PesoEstimacao;                            //Ponderação MQP
      }

    if (LinSist.Solucionar(&H, &w, &r))     //Solucionando o problema
    { //Restrições de igualdade
      VanLoan(&H, &w, &LinSist, FNumIteracoesRestr[Iteração-1]);
      //Obtendo a SPQR
      FSPQR[Iteracao - 1] = LinSist.SPQR();
      //Determinando o maior incremento e atualizando os estados
      EMax = 0.00;
      for (unsigned int i = 0; i < FNumEstados; i++)
      { FVarEstado[i]->Incrementar(LinSist.Solucao()->GetData(i));

        if (fabs(LinSist.Solucao()->GetData(i)) > EMax)
           EMax = fabs(LinSist.Solucao()->GetData(i));
      }
      FConvergido = ((EMax < FMaxErroAdmitido)&&(!ComEG)); //Testando a convergência
    }
    else //Verificando as possíveis variáveis de estado não observáveis
    { Solucionabilidade(LOC_ESTIMACAO, LinSist, FVarEstado, Recompor);
      return false;
    }
  } while ((Iteracao < FNumMaxIteracoes)&&(!FConvergido));

  if (FIdentificarEG) IdentificarEGs();     //Identificando Erros Grosseiros
  BalancoDePotencia();                      //Balanço de potência
  return true;
};
```

Figura 86 – Rotina de estimação da classe *TPSSERobusto*: Fragmento de código do estimador acoplado.

Capítulo 5

Resultados Numéricos

5.1 Introdução

O presente capítulo tem o objetivo de apresentar os resultados numéricos obtidos durante a fase de teste e de validação das ferramentas matemáticas e dos diferentes estimadores de estados desenvolvidos com o suporte do modelo de SEP proposto neste trabalho. Inicialmente serão mostrados os desempenhos e a análise comparativa das classes e dos métodos implementados no pacote de ferramentas matemáticas, fazendo algumas comparações e críticas em relação a algumas publicações recentes. Posteriormente serão apresentados os resultados completos dos estimadores de estados aplicados a sistemas teste de pequeno porte. Em seguida, o resumo dos resultados obtidos em sistemas teste maiores com o objetivo de mostrar e registrar o desempenho dos estimadores. Por fim, os resultados da aplicação do estimador de estados robusto baseado em MQPV sobre o sistema de transmissão da CEEE serão apresentados em conjunto com os resultados do estimador baseado em MQP do CEPEL. As dificuldades e problemas encontrados na modelagem do sistema de transmissão e do plano de medição da CEEE também serão abordados.

As ferramentas matemáticas serão testadas com o uso da matriz jacobiana obtida do sistema sul-sudeste reduzido de 1916 barras, a qual possui 9305 linhas, 3831 colunas e 39956 elementos não-nulos. O uso de matrizes menores prejudicaria a qualidade da tomada de tempo dos testes realizados, uma vez que o tempo de processamento da maioria dos testes com matrizes menores é próximo da precisão das funções utilizadas para registrar o tempo.

Os estimadores de estados serão testados e validados com o uso dos sistemas de teste do IEEE com 14 e 118 e de redes equivalentes do sistema sul-sudeste brasileiro com 340, 730 e 1916 barras. A Tabela 5 apresenta o resumo das características de tais sistemas.

Tabela 5 – Características principais dos sistemas de teste utilizados.

Sistema Teste	B	C	D	E	F
Número de Barras	14	118	340	730	1916
Número de Ramos	20	179	494	913	2357
Qtd de medidas de fluxo	40	242	1160	2524	6116
Qtd de medidas de injeções	15	86	485	1088	2126
Qtd de medidas de tensão	7	30	159	402	1063
Qtd total de medidas	60	358	1804	4014	9305

Os arquivos que descrevem todos os parâmetros dos sistemas supracitados e o arquivo que possui os dados da matriz utilizada nos testes das ferramentas matemáticas estão gravados no diretório "\\DADOS\SISTEMAS" e "\\DADOS\MATRIZES" do CD em anexo, bem como a versão acadêmica do projeto desenvolvido.

Todos os testes deste trabalho serão apresentados, quando cabível, com os respectivos desvios-padrões das medidas de tempo de processamento. Sendo o sistema operacional multitarefa, não se pode prever o comportamento de processos concorrentes. Logo, para mitigar possíveis erros na mensuração do tempo de processamento, todos os testes foram realizados 10 vezes em momentos e seqüências diferentes. O computador utilizado foi um PENTIUM 4 de 2.66 GHz, 512 MB de RAM, 10GB de espaço livre no disco rígido e utilizando o sistema operacional Windows 2000 com *service pack* 4.

A seção 5.2 apresentará todos os testes realizados com as classes que compõem o pacote de ferramentas matemáticas e seus respectivos resultados. A seção 5.3 apresentará os testes completos com o sistema de 14 barras do IEEE. A seção 5.4 apresentará o resumo, os comentários e as observações dos testes com os sistemas de 118, 340, 730 e 1916 barras. A seção 5.5 apresentará as características do sistema de transmissão da CEEE e as principais dificuldades encontradas durante a aplicação do estimador de estados robusto, fornecendo algumas as comparações e tecendo os comentários cabíveis. Por fim, a seção 5.6 apresentará as conclusões obtidas a partir do conjunto de resultados.

5.2 Ferramentas matemáticas

5.2.1 Dos testes de desempenho realizados

Os testes de desempenho das ferramentas matemáticas estão agrupados por tipo de operação realizada. A maioria das funcionalidades descritas no capítulo anterior serão objetos destes testes e sempre que possível serão exibidos os testes realizados tanto na classe *MatES* quanto na classe *Matriz*. Cabe ressaltar que a validação dos algoritmos e operações foi realizada comparando-se os resultados obtidos com aqueles gerados através do software matemático Matlab.

Os grupos e seus correspondentes testes serão:

1. *Operações básicas:* Serão analisados os desempenhos das classes frente a operações de adições, subtração, multiplicação e atribuição;
2. *Ordenação:* Serão verificados os esforços computacionais para a aplicação de cada possível combinação de métodos de ordenação;

3. *Fatoração:* Serão comparados os desempenhos de todos os métodos de fatoração desenvolvidos para as classes *Matriz* e *MatES* conforme as possíveis combinações dos métodos de ordenação de linhas e colunas;
4. *Demais testes:* Os desempenhos da inversão esparsa de Broussolle para diferentes métodos de ordenação e registro do esforço computacional relativo ao cálculo dos pontos de alavancamento;

5.2.2 Testando operações matriciais básicas

5.2.2.1 Procedimentos aplicados a cada teste

As variáveis MAT1 e MAT2 mencionadas nestes testes são objetos das classes Matriz ou MatES, conforme a classe sob análise. Os testes relativos às operações básicas foram:

- **OP1 – Atribuição:** O teste de atribuição consiste em atribuir a MAT2 a matriz de teste carregada em MAT1, repetidamente por 500 vezes.
- **OP2 – Soma e subtração:** O teste de soma e subtração consiste e se atribuir a MAT2 e a MAT1 a matriz de testes e alterar soma e subtração por 100 vezes;
- **OP3 – Multiplicação:** O teste de multiplicação consiste em se atribuir a MAT1 e MAT2 a matriz de testes, transpor MAT2 e calcular o produto de MAT2 por MAT1.

5.2.2.2 Resultados obtidos

Tabela 6 – Resultados do teste de operações matriciais básicas

Teste	Classe *MatES*		Classe *Matriz*	
	tempo	±σ	tempo	±σ
OP1	0,563	0,011	7,468	0,331
OP2	0,398	0,013	7,732	0,822
OP3	78,400	5,687	0,11	0,004

5.2.2.3 Comentários

Há uma nítida vantagem da classe MatES em procedimentos que envolvem atribuições repetitivas. Isto mostra que a o uso de listas formadas por blocos seqüenciais de memória aliada às funções primárias de movimentação e alocação são superiores, para este caso, ao uso de objetos que alocam memória de forma segmentada.

O teste de multiplicação para a classe MatES pode ser considerado inválido, uma vez que o procedimento utilizado no cálculo é rudimentar e extremamente lento. Há algoritmos para este tipo de estrutura de dados que possibilitam a eficiência alcançada com a classe *Matriz*.

5.2.3 Comparando o esforço computacional dos métodos de ordenação matricial

5.2.3.1 Descrição do teste

A matriz de teste será carregada em MAT1, objeto da classe MatES ou Matriz, e posteriormente serão aplicadas a ela todas as possíveis combinações entre métodos de ordenação de linhas e de ordenação de colunas. A aplicação isolada de métodos de ordenação de linhas será excluída do teste.

5.2.3.2 Resultados obtidos

Tabela 7 – Resultados dos testes de ordenação

Teste		**Classe *MatES***		**Classe *Matriz***	
Método - Coluna	**Método - Linha**	**tempo**	**±σ**	**tempo**	**±σ**
MDA	**Isolado**	0,40340	0,03755	0,40790	0,04698
	R1	0,54660	0,04934	0,53280	0,04703
	R2	0,53130	0,03102	0,53910	0,05976
	R3	0,53770	0,03302	0,54360	0,04325
	R4	0,53580	0,04929	0,54680	0,04965
	R5	0,52810	0,03729	0,54510	0,04664
A1	**Isolado**	0,40010	0,05100	0,40770	0,04087
	R1	0,55010	0,05281	0,53620	0,05466
	R2	0,52810	0,02965	0,54060	0,04535
	R3	0,53890	0,04511	0,55350	0,05595
	R4	0,54370	0,04856	0,54380	0,04340
	R5	0,54230	0,06255	0,55150	0,05478

5.2.3.3 Comentários

O objetivo do teste de ordenação restringe-se ao registro do esforço computacional necessário à ordenação de cada classe. A análise destes valores sem considerar a respectiva redução do esforço computacional relativo à fatoração matricial não tem significado.

5.2.4 Análise comparativa dos métodos de fatoração

5.2.4.1 Descrição do teste

Todos os métodos de fatoração implementados para ambas as classes matriciais terão seu respectivo esforço computacional registrado para cada combinação dos métodos de ordenação de linhas e colunas. Para registro do impacto dos métodos de ordenação na redução do esforço

computacional necessário a fatoração, será exibido o tempo necessário a fatoração da matriz de testes sem empregar qualquer método de ordenação.

5.2.4.2 Resultados

Tabela 8 – Desempenho dos diversos métodos de fatoração da classe *Matriz* frente às possíveis ordenações

Classe Matriz									
Fatoração		Givens 2M COP		Givens 2M ROP		Givens 3M ROP		LDL^T	
Método Coluna	Método Linha	tempo	±σ	tempo	±σ	tempo	±σ	tempo	±σ
Nenhum	Nenhum	16,38567	0,11336	2,43767	0,03838	2,56233	0,02192	1,19267	0,08355
MDA	Nenhum	5,01017	0,05440	0,51958	0,02071	0,52350	0,02666	0,24092	0,01181
	R1	4,95183	0,06950	0,48433	0,02577	0,47925	0,01950	0,23958	0,01192
	R2	2,41417	0,02992	0,36350	0,02073	0,36859	0,01795	0,24725	0,01832
	R3	0,19925	0,01940	0,16158	0,01771	0,15375	0,01824	0,25658	0,02536
	R4	0,19542	0,01517	0,15500	0,00991	0,15883	0,02352	0,25783	0,02030
	R5	0,19792	0,02536	0,14983	0,01423	0,14958	0,01802	0,25250	0,03332
A1	Nenhum	5,15617	0,07863	0,54167	0,03199	0,54442	0,02452	0,25017	0,00970
	R1	5,10950	0,05676	0,50133	0,01276	0,50150	0,01177	0,24733	0,01192
	R2	2,49459	0,06015	0,39575	0,02656	0,39575	0,01842	0,24992	0,02596
	R3	0,19550	0,01287	0,15517	0,01496	0,14459	0,01572	0,25650	0,02562
	R4	0,20042	0,02161	0,14842	0,00939	0,15358	0,01845	0,25900	0,02150
	R5	0,19667	0,01805	0,14709	0,00875	0,13934	0,01822	0,25383	0,01960

Tabela 9 - Desempenho dos diversos métodos de fatoração da classe *MatES* frente às possíveis ordenações

Classe MatES					
Fatoração		Givens 2M ROP		Givens 3M ROP	
Método Coluna	Método Linha	tempo	±σ	tempo	±σ
Nenhum	Nenhum	2,52956	0,03095	2,29000	0,02001
MDA	Nenhum	0,64411	0,02564	0,63889	0,02419
	R1	0,57300	0,02013	0,57800	0,02731
	R2	0,42711	0,01616	0,43233	0,01266
	R3	0,13689	0,00844	0,14222	0,01186
	R4	0,14411	0,01616	0,13922	0,00871
	R5	0,14393	0,01306	0,14483	0,01730
A1	Nenhum	0,62489	0,01625	0,63356	0,01566
	R1	0,55933	0,01279	0,56944	0,02967
	R2	0,42011	0,01703	0,42878	0,01306
	R3	0,13733	0,01279	0,13889	0,01254
	R4	0,13711	0,02372	0,13178	0,00831
	R5	0,12767	0,01306	0,13784	0,02613

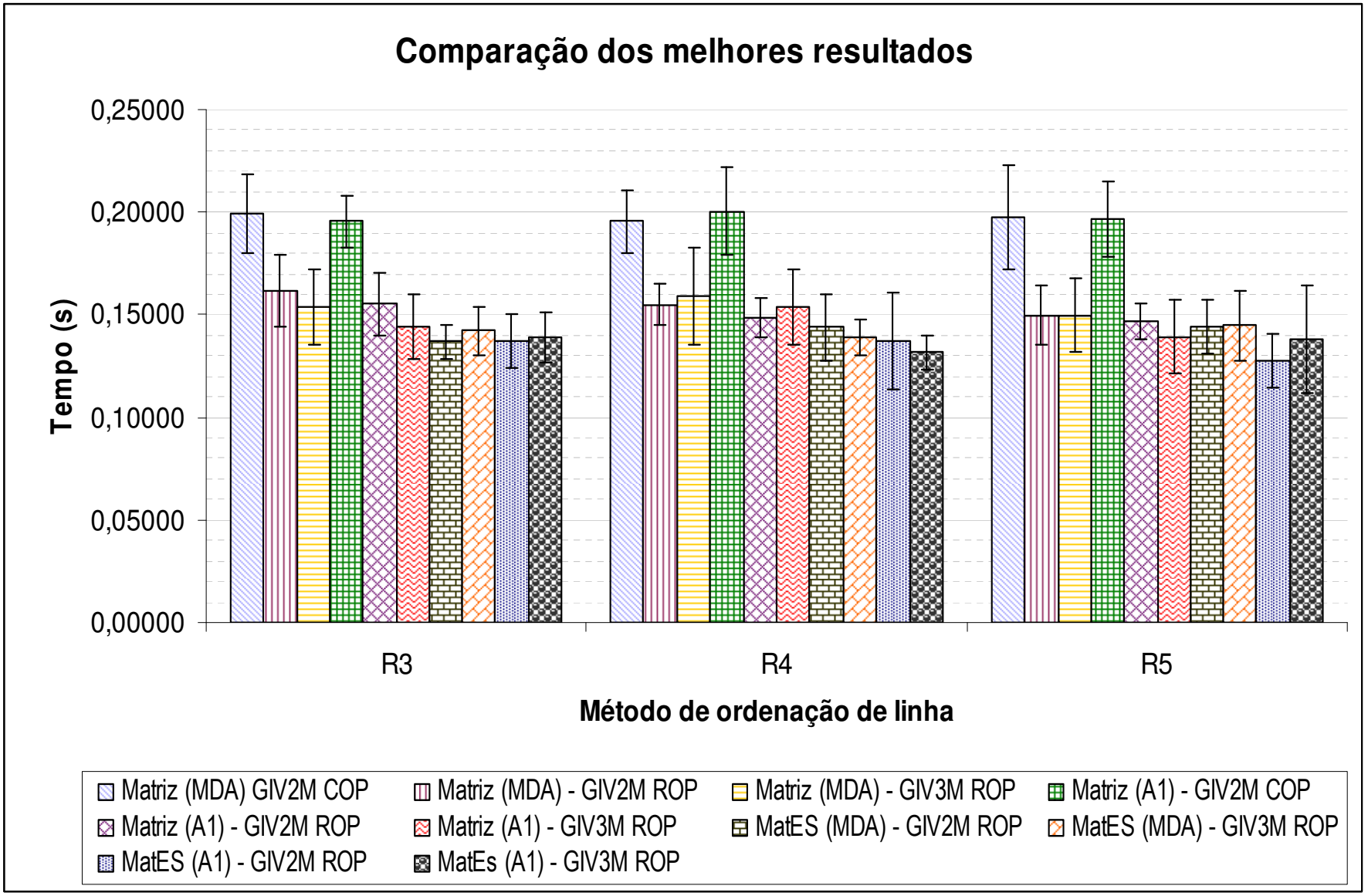


Figura 87 – Análise comparativa das melhores combinações entre métodos de fatoração e ordenação, conforme tabelas 8 e 9.

5.2.4.3 Comentários e conclusões

As seguintes considerações devem ser feitas antes de qualquer análise:

- Os resultados relativos à fatoração LDL^T incluem o tempo necessário para se aplicar a equação normal de Gauss ao sistema linear formado pela matriz de testes. Sendo a operação $MAT1^T*MAT1$ – onde *MAT1* é a matriz de teste – parte do processo de solução, o respectivo esforço computacional deve ser considerado;
- O algoritmo desenvolvido para a fatoração LDL^T não utiliza recursos especiais para se obter um melhor desempenho, como em [24] e [71]. Entretanto isto não significa que existam operações lógicas ou aritméticas desnecessárias ao código;
- As grandes variações são resultantes da redução da velocidade de processamento em virtude do aquecimento natural do processador perante a um período relativamente longo de operações consecutivas e ininterruptas;
- Todas as possíveis conclusões se limitaram aos métodos aplicados neste trabalho e sob as condições impostas por um sistema operacional multitarefa.

Com base nestas observações e no conjunto de resultados obtidos, pode-se afirmar que sistemas lineares semelhantes àqueles obtidos a partir do problema de EESP serão solucionados com menor esforço computacional quando:

I. A matriz de coeficientes for previamente ordenada utilizando-se qualquer combinação dos métodos de ordenação de colunas MDA e A1 e os métodos de ordenação de linhas de R3, R4 e R5; e
II. Os métodos de fatoração baseados em rotações ortogonais de Givens com 2 ou 3 multiplicados forem aplicados diretamente à matriz utilizando-se processamento orientado por linha.

Não se pode se afirmar mais nada a partir dos resultados obtidos, apesar do desempenho da classe *MatES* ser aparentemente melhor. Entretanto deve-se comentar que o os métodos de fatoração orientados por linha obtiveram tais resultados para a classe Matriz devido à completa desconsideração da lista de encadeamento de colunas. Caso o algoritmo mantivesse a cada rotação a estrutura de ligação das colunas matriciais, dificilmente os resultados obtidos seriam compatíveis com aqueles obtidos para a classe MatES. Este é um detalhe importante, pois evita procuras desnecessárias para manter esquemas de ligações supérfluos ao objetivo do algoritmo. Após o término do processo, esquema de ligações de colunas é efeito sobre a matriz R resultante.

Não era objetivo deste trabalho a implementação da fatoração utilizando-se rotações de Givens com 2 multiplicadores com orientação por coluna. Entretanto os resultados obtidos em [105] e [29] indicaram que este método aplicado a determinadas estruturas de dados produzia resultados superiores àqueles mostrados em [68]. Não houve menção direta sobre qual seria tal estrutura em [105], mas em [29] indica-se uma estrutura de dados semelhante à classe *Matriz*. Entretanto algumas considerações devem ser feitas sobre os resultados obtidos em [105] e indicados em [29], inclusive sobre os resultados utilizando-se o método de ordenação dinâmica de linhas VPAIR:

- Os métodos de ordenação estudados em [68] são aplicados antes do processo de fatoração e, sobretudo, são simples e relativamente rápidos;

- Os métodos de ordenação de linhas que comparem a disposição dos elementos não nulos antes da rotação ortogonal elementar e selecionem as linhas que irão produzir o menor número de elementos intermediários produzirão provavelmente uma quantidade menor de elementos intermediários quando comparado a qualquer outro método de ordenação de linhas que seja aplicado à estrutura da matriz de forma pretérita;

- A redução da quantidade de elementos intermediários, como indicado em [29] e [105] para a combinação do MDA e VPAIR, não significa que o método irá realizar a fatoração em um intervalo de tempo menor. Trata-se de um indicativo valioso que associado a boas técnicas computacionais pode resultar em redução do tempo de processamento;

- As estruturas de dados (matrizes e vetores) utilizadas na fatoração não foram detalhadas em [105], bem como se omitiu detalhes técnicos sobre a prática computacional adotada. Em [29] há o detalhamento da estrutura das matrizes e vetores, sem se relacionar explicitamente aos resultados obtidos em [105]. Entretanto há uma pequena menção ao uso de tais estruturas em [29] e uma menção a esse tipo de estrutura em [105]. Neste caso, se entende que a matriz utilizada é estruturada através de listas encadeadas, com estrutura semelhante às estruturas propostas neste trabalho e em [24];

- Os resultados de [105] mostram uma significativa diferença entre o tempo computacional do estimador utilizando ROP (Ordenação MDA e R5 com fatoração através de rotações de Givens com 2 multiplicadores com orientação por linha) e o estimador utilizando COP1 (Ordenação MDA e R5 com fatoração através de rotações de Givens com 2 multiplicadores com orientação por coluna). O estimador utilizando COP1 era 25% mais rápido que o estimador utilizando ROP.

Durante o desenvolvimento dos algoritmos da classe Matriz, alguns resultados semelhantes foram obtidos para os supracitados esquemas de fatoração ROP e COP1. Entretanto o

algoritmo ora desenvolvido mantinha a estrutura de ligação das colunas em ROP, gerando um processamento adicional nada desprezível. Após a otimização deste algoritmo, o tempo necessário ao término da fatoração foi reduzido em mais de 45%. Isto indica que os autores de [105] possivelmente não otimizaram o código que deu origem às informações sobre este esquema de fatoração.

Futuros trabalhos devem ser desenvolvidos para propiciar uma comparação em entre os métodos de ordenação de linhas discutidos em [68] e aplicados neste trabalho e o método de ordenação dinâmico de linhas VPAIR.

5.2.5 Demais testes

5.2.5.1 Desempenho da inversão esparsa de Broussole

Tabela 10 – Desempenho do algoritmo de inversão matricial esparsa proposto por Broussolle

Classe Matriz									
Fatoração		Givens 2M ROP		Givens 3M ROP		Givens 2M COP		LDL^T	
Método - Coluna	Método - Linha	tempo	±σ	tempo	±σ	tempo	±σ	tempo	±σ
A1	Nenhum	0,59400	0,03797	0,61433	0,03318	5,11933	0,07112	0,34400	0,02263
	R1	0,55733	0,02531	0,57867	0,02192	5,19800	0,01225	0,34900	0,04654
	R2	0,45267	0,04455	0,45333	0,03838	2,52100	0,08941	0,35400	0,03379
	R3	0,25500	0,05620	0,23967	0,04631	0,28667	0,02531	0,34367	0,04385
	R4	0,23467	0,02192	0,23967	0,03395	0,29700	0,00000	0,35967	0,07675
	R5	0,21300	0,03379	0,21833	0,03838	0,28133	0,05826	0,32800	0,08910

Conforme demonstra os resultados da Tabela 10, o desempenho da inversão esparsa proposta por Broussolle está diretamente relacionado ao esquema de fatoração adotado no problema. Estando disponível a matriz triangular resultante do processo de fatoração, o algoritmo requer 0,081 ± 0,021s, conforme as diferenças obtidas entre os testes de desempenho mostrados nas tabelas 9 e 10.

5.2.5.2 Pontos de alavancamento

A identificação e o cálculo dos pesos relativos aos pontos de alavancamento dependem somente do espaço fator da gerado pela matriz de coeficientes do sistema linear sob análise. Para o caso da matriz de teste utilização em toda esta subseção, os testes de desempenho mostraram que o calculo dos pontos de alavancamento adiciona 0,375 ± 0,073s ao processo de estimação de estados.

5.3 Validação dos estimadores de estados: Sistema IEEE de 14 barras

5.3.1 Introdução

A validação dos estimadores de estados desenvolvidos neste trabalho será feita através dos sistemas de teste do IEEE de 14 barras, pois permite uma análise mais detalhada do comportamento e dos resultados dos estimadores. O sistema do IEEE de 14 barras foi escolhido por ser relativamente pequeno e possuir quase todos os componentes de rede desenvolvidos, além de possibilitar todos os testes necessários aos estimadores desenvolvidos.

Inicialmente serão apresentados os parâmetros do sistema, o ponto operativo de testes, o plano de medição, as configurações iniciais dos estimadores e a descrição dos testes a serem realizados. Posteriormente serão apresentados os resultados detalhados de cada teste e serão feitos os comentários que se fizerem necessários.

5.3.2 O sistema IEEE de 14 barras

Tabela 11 – Parâmetros do sistema IEEE de 14 barras

Ramo			Série		Shunt		TAP
No	DE	PARA	Gij [pu]	Bij [pu]	B/2 [pu]	B/2 [MVA]	
1	1	2	9,9558	-30,5380	0,0528	0,0528	
2	1	5	1,0262	-4,2357	0,0246	0,0246	
3	2	3	1,1347	-4,7812	0,0219	0,0219	
4	2	4	1,6864	-5,1163	0,0187	0,0187	
5	2	5	1,7008	-5,1934	0,0170	0,0170	
6	3	4	1,9866	-5,0695	0,0173	0,0173	
7	4	5	6,8439	-21,5830	0,0064	0,0064	
8	4	7		-4,7824			0,978
9	4	9		-1,7979			0,969
10	5	6		-7,9365			0,932
11	6	11	1,9550	-4,0941			
12	6	12	1,5261	-3,1760			
13	6	13	3,0978	-6,1019			
14	7	8		-5,6786			
15	7	9		-9,0909			
16	9	10	3,9020	-10,3650			
17	9	14	1,4238	-3,0289			
18	10	11	1,8804	-4,4025			
19	12	13	2,4888	-2,2520			
20	13	14	1,1371	-2,3150			
Shunts							
Barra			Bii [pu]	Bii [MVA]	Barra	Bii [pu]	Bii [MVA]
1			-0,2300	-0,2300	12	0,1000	0,1000

O sistema IEEE de 14 barras possui 17 são linhas de transmissão, 3 transformadores, 1 banco de capacitores e 1 banco de reatores, totalizando 20 ramos e 2 bancos. Os grupos geradores situam-se nas barras de número 1, 2, 3, 6 e 8. As tabelas 11 e 12 mostram, respectivamente, os parâmetros e o ponto operativo escolhido para os testes.

Tabela 12 – Ponto operativo do conjunto de testes

Barra	Tensão		Geração		Carga	
	V	Φ	Pg	Qg	Pc	Qc
1	1,0600	0,000	1,4480	0,4948		
2	1,0450	-1,500	0,4000	0,0348	0,2170	0,1270
3	1,0100	-6,280	0,4000	0,1056	0,9420	0,1900
4	1,0150	-5,430			0,4780	0,0390
5	1,0180	-4,390			0,0760	0,0160
6	1,0700	-5,630	0,4000	-0,0526	0,1120	0,0750
7	1,0490	-7,440				
8	1,0900	-7,440	0,0000	0,2567		
9	1,0290	-8,510			0,2950	0,1660
10	1,0280	-8,280			0,0900	0,0580
11	1,0440	-7,080			0,0350	0,0180
12	1,0690	-7,170			0,0610	0,0160
13	1,0510	-6,950			0,1350	0,0580
14	1,0200	-8,850			0,1490	0,0500

O plano de medição escolhido, mostrado na Figura 88, possui um total de 60 medidas, das quais 7 são de magnitude de tensão, 7 de injeção de potência ativa, 6 de injeção de potência reativa e 40 de fluxo de potência ativa e reativa. Com exceção da barra número 1, a qual possui somente mensuração de potência ativa, todas as demais medidas de injeção e fluxo são tanto de potência ativa quanto reativa.

Os testes serão realizados de forma seqüencial, partindo-se do caso mais simples. O objetivo de cada teste é validar modelos ou analisar funcionalidades e possíveis problemas. A seqüência de testes para ambos os estimadores serão as seguintes:

- **T1:** Todas as medidas perfeitas (sem ruídos);
- **T2:** Adição da estimação da posição dos TAPs dos transformadores
- **T3:** Adição de ruídos e erros grosseiros;
- **T4:** Adição de erros grosseiros próximos a transformadores;
- **T5:** Substituição dos erros grosseiros próximos a transformadores por erros grosseiros em medidas identificadas como pontos de alavancamento;
- **T6:** Substituição dos erros grosseiros em medidas identificadas como pontos de alavancamento por erros grosseiros próximos a barras de passagem ou transição, incluindo-se o processamento das restrições de igualdade;

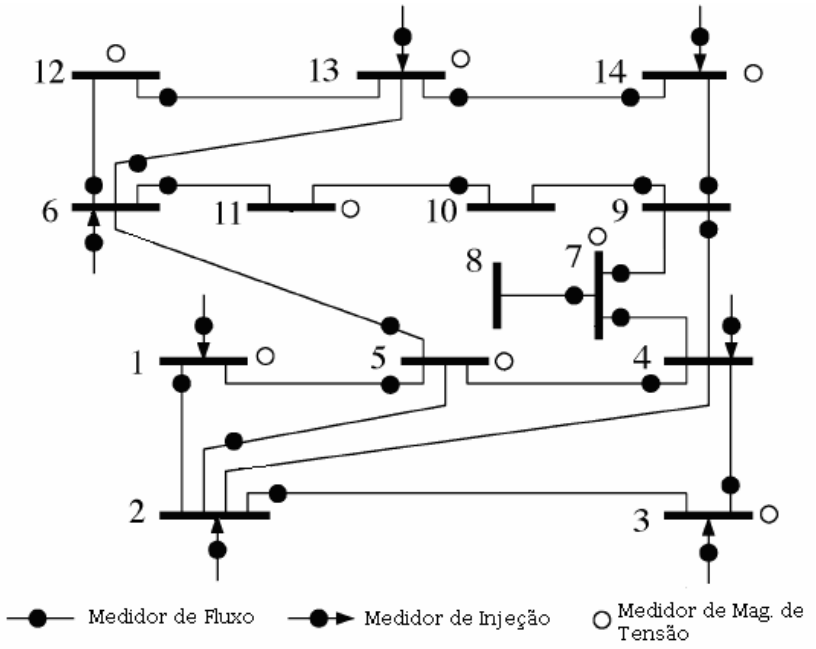


Figura 88 – Plano de medição do sistema IEEE de 14 barras

As configurações dos estimadores de estados e as precisões dos medidores para os testes realizados são mostradas nas tabelas 13 e 14, respectivamente.

Tabela 13 – Configurações dos estimadores MQP, MQPV e MQPV-Descoplado para os teste de validação

Configurações comuns	
Número máximo de iterações	50
Tolerância para convergência	0,001
Matriz Jacobiana constante	Não
Utilizar perfil plano de tensões	Sim
Incluir as restrições de igualdade	Somente no teste T6
Tolerância aplicada às restrições	0,001
MQP	
Início do tratamento de EG	3ª iteração
Probabilidade de falso alarme	0,025
MPQV e MPQV-Desacoplado	
Número de iterações MQP	1
Método de identificação de EG	Peso ou Ponderação
Valor limite da identificação de EG	0,448 (aprox. 3σ)
Considerar pontos de alavancamento	Somente em T5, T6 e T7.

Tabela 14 – Precisão adotada para os medidores durante os testes de validação

Precisões dos medidores				
V	P	Q	t	u
0,10%	1,00%	1,00%	1,00%	1,00%

Tabela 15 – Valores medidos e estimados do teste T1.

#	Conjunto de Medidas				MQPV		MQPV - Desacoplado		MQP	
	Medida			Valor medido	3 Iterações		6 Iterações		3 Iterações	
	Tipo	ID			Valor Estimado	Resíduo	Valor Estimado	Resíduo	Valor Estimado	Resíduo
1	V	1		1,060E+00	1,060E+00	-4,561E-06	1,060E+00	1,124E-04	1,060E+00	-4,561E-06
2	V	3		1,010E+00	1,010E+00	-3,629E-06	1,010E+00	-3,171E-05	1,010E+00	-3,629E-06
3	V	5		1,018E+00	1,018E+00	3,810E-06	1,018E+00	7,027E-05	1,018E+00	3,810E-06
4	V	11		1,045E+00	1,045E+00	-3,784E-06	1,045E+00	-9,252E-06	1,045E+00	-3,784E-06
5	V	12		1,069E+00	1,069E+00	3,940E-06	1,069E+00	-5,174E-05	1,069E+00	3,940E-06
6	V	13		1,051E+00	1,051E+00	3,204E-05	1,051E+00	1,923E-06	1,051E+00	3,204E-05
7	V	14		1,020E+00	1,020E+00	-2,724E-05	1,020E+00	-8,127E-05	1,020E+00	-2,724E-05
8	P	1		1,448E+00	1,448E+00	1,446E-05	1,448E+00	-2,651E-06	1,448E+00	1,446E-05
9	P	2		1,830E-01	1,830E-01	8,672E-06	1,830E-01	1,128E-05	1,830E-01	8,675E-06
10	P	3		-5,420E-01	-5,420E-01	2,567E-06	-5,420E-01	-3,355E-06	-5,420E-01	2,568E-06
11	P	4		-4,780E-01	-4,780E-01	-2,645E-06	-4,780E-01	-5,099E-06	-4,780E-01	-2,645E-06
12	P	6		2,880E-01	2,880E-01	-4,335E-05	2,881E-01	-4,999E-05	2,880E-01	-4,335E-05
13	P	13		-1,350E-01	-1,349E-01	-6,483E-05	-1,350E-01	-1,742E-05	-1,349E-01	-6,483E-05
14	P	14		-1,490E-01	-1,491E-01	5,897E-05	-1,490E-01	2,000E-05	-1,491E-01	5,897E-05
15	Q	2		-9,222E-02	-9,222E-02	-4,194E-06	-8,823E-02	-3,995E-03	-9,222E-02	-4,197E-06
16	Q	3		-8,444E-02	-8,444E-02	-4,286E-06	-8,424E-02	-2,001E-04	-8,444E-02	-4,286E-06
17	Q	4		-3,900E-02	-3,896E-02	-3,850E-05	-3,622E-02	-2,782E-03	-3,896E-02	-3,850E-05
18	Q	6		-1,276E-01	-1,275E-01	-1,577E-05	-1,317E-01	4,166E-03	-1,275E-01	-1,577E-05
19	Q	13		-5,800E-02	-5,791E-02	-9,031E-05	-5,650E-02	-1,497E-03	-5,791E-02	-9,031E-05
20	Q	14		-5,000E-02	-5,007E-02	7,368E-05	-4,964E-02	-3,611E-04	-5,007E-02	7,368E-05
21	t	1	2	1,050E+00	1,050E+00	1,247E-05	1,050E+00	-2,026E-05	1,050E+00	1,247E-05
22	t	2	3	4,658E-01	4,658E-01	-6,907E-06	4,658E-01	-2,423E-05	4,658E-01	-6,907E-06
23	t	2	5	3,277E-01	3,277E-01	-2,463E-06	3,277E-01	-4,157E-07	3,277E-01	-2,463E-06
24	t	3	4	-8,588E-02	-8,588E-02	6,645E-06	-8,586E-02	-1,722E-05	-8,588E-02	6,645E-06
25	t	4	5	-4,315E-01	-4,315E-01	1,261E-05	-4,315E-01	1,367E-05	-4,315E-01	1,261E-05
26	t	5	1	-3,900E-01	-3,900E-01	8,900E-06	-3,900E-01	-5,147E-06	-3,900E-01	8,900E-06
27	t	5	6	2,020E-01	2,021E-01	-4,343E-05	2,020E-01	-1,571E-05	2,021E-01	-4,343E-05
28	t	6	11	1,694E-01	1,693E-01	9,915E-06	1,693E-01	6,633E-05	1,693E-01	9,915E-06
29	t	6	12	9,989E-02	9,984E-02	4,963E-05	9,990E-02	-4,274E-06	9,984E-02	4,963E-05
30	t	6	13	2,209E-01	2,209E-01	2,667E-05	2,209E-01	4,524E-05	2,209E-01	2,667E-05
31	t	7	4	-1,826E-01	-1,826E-01	-5,188E-06	-1,826E-01	3,016E-05	-1,826E-01	-5,188E-06
32	t	7	8	3,331E-16	3,603E-16	-2,720E-17	3,602E-16	-2,717E-17	3,603E-16	-2,720E-17
33	t	7	9	1,826E-01	1,826E-01	1,489E-06	1,826E-01	-1,833E-05	1,826E-01	1,489E-06
34	t	9	4	-1,040E-01	-1,040E-01	-4,340E-06	-1,040E-01	1,391E-05	-1,040E-01	-4,340E-06
35	t	9	10	-4,029E-02	-4,029E-02	5,801E-06	-4,028E-02	-1,264E-05	-4,029E-02	5,801E-06
36	t	9	14	3,174E-02	3,174E-02	-1,664E-06	3,174E-02	-1,792E-06	3,174E-02	-1,664E-06
37	t	10	11	-1,304E-01	-1,304E-01	1,534E-05	-1,304E-01	-2,995E-05	-1,304E-01	1,534E-05
38	t	12	13	3,766E-02	3,772E-02	-6,357E-05	3,774E-02	-8,115E-05	3,772E-02	-6,357E-05
39	t	13	14	1,199E-01	1,199E-01	-1,061E-05	1,199E-01	3,256E-05	1,199E-01	-1,061E-05
40	t	14	13	-1,175E-01	-1,175E-01	1,222E-05	-1,174E-01	-2,866E-05	-1,175E-01	1,222E-05
41	u	1	2	1,485E-01	1,485E-01	2,090E-06	1,445E-01	4,005E-03	1,485E-01	2,092E-06
42	u	2	3	6,877E-02	6,877E-02	-2,684E-06	6,859E-02	1,759E-04	6,877E-02	-2,684E-06
43	u	2	5	4,179E-02	4,178E-02	9,164E-06	4,214E-02	-3,504E-04	4,178E-02	9,164E-06
44	u	3	4	-1,032E-02	-1,032E-02	-6,289E-06	-1,029E-02	-2,709E-05	-1,032E-02	-6,290E-06
45	u	4	5	4,221E-02	4,215E-02	6,734E-05	4,430E-02	-2,091E-03	4,215E-02	6,734E-05
46	u	5	1	-1,069E-01	-1,069E-01	1,776E-06	-1,066E-01	-2,470E-04	-1,069E-01	1,776E-06
47	u	5	6	1,990E-01	1,990E-01	4,251E-05	1,998E-01	-7,504E-04	1,990E-01	4,251E-05
48	u	6	11	5,810E-02	5,814E-02	-4,349E-05	5,725E-02	8,538E-04	5,814E-02	-4,349E-05
49	u	6	12	-4,162E-02	-4,170E-02	8,138E-05	-4,264E-02	1,014E-03	-4,170E-02	8,138E-05
50	u	6	13	4,648E-02	4,649E-02	-8,644E-06	4,490E-02	1,583E-03	4,649E-02	-8,644E-06
51	u	7	4	5,901E-02	5,900E-02	9,564E-06	5,870E-02	3,109E-04	5,900E-02	9,564E-06
52	u	7	8	-2,470E-01	-2,470E-01	1,469E-11	-2,470E-01	1,172E-08	-2,470E-01	2,182E-11
53	u	7	9	1,881E-01	1,882E-01	-2,555E-05	1,883E-01	-1,394E-04	1,882E-01	-2,555E-05
54	u	9	4	-3,063E-02	-3,064E-02	8,098E-06	-3,077E-02	1,400E-04	-3,064E-02	8,098E-06
55	u	9	10	2,684E-02	2,683E-02	1,067E-05	2,668E-02	1,598E-04	2,683E-02	1,067E-05
56	u	9	14	1,909E-02	1,913E-02	-4,657E-05	1,874E-02	3,475E-04	1,913E-02	-4,657E-05
57	u	10	11	-3,132E-02	-3,135E-02	2,445E-05	-3,160E-02	2,751E-04	-3,135E-02	2,445E-05
58	u	12	13	5,416E-02	5,422E-02	-6,222E-05	5,434E-02	-1,849E-04	5,422E-02	-6,222E-05
59	u	13	14	3,617E-02	3,624E-02	-6,984E-05	3,619E-02	-2,074E-05	3,624E-02	-6,984E-05
60	u	14	13	-3,122E-02	-3,129E-02	6,793E-05	-3,125E-02	2,350E-05	-3,129E-02	6,793E-05
Graus de liberdade				33	SPQR	1,556E-03	SPQR	6,984E-01	SPQR	5,698E-03

5.3.2.1 Teste T1: Validação dos modelos e dos algoritmos

O teste T1 limita-se a analisar e validar os modelos dos componentes de rede e as rotinas dos estimadores de estados desenvolvidos. Entretanto não se pode afirmar que os métodos de detecção e tratamento de EG estejam corretos. Por não contemplar ruídos ou erros grosseiros no conjunto de medidas, os erros percentuais entre os *valores reais* e as medidas estimadas pelos estimadores *acoplados* devem ser menores que os respectivos desvios-padrão. Para o estimador desacoplado no modelo, devido às considerações feitas durante a sua formulação, tais erros podem ser superiores aos respectivos desvios-padrão.

Os resultados mostrados nas tabelas 15 e 16 mostram o resultado que os modelos e parte dos algoritmos dos estimadores estão corretos.

Tabela 16 – Valores dos estados estimados no teste T1

Barra	Estados reais		MQPV		MQPV-Desacoplado		MQP	
	V	φ [rad]	V	φ [rad]	V	φ [rad]	V	φ [rad]
1	1,0600	0,000	1,0600	0,0000	1,0599	0,0000	1,0600	0,0000
2	1,0450	-0,026	1,0450	-0,0262	1,0450	-0,0263	1,0450	-0,0262
3	1,0100	-0,110	1,0100	-0,1096	1,0100	-0,1097	1,0100	-0,1096
4	1,0150	-0,095	1,0146	-0,0948	1,0146	-0,0948	1,0146	-0,0948
5	1,0180	-0,077	1,0184	-0,0766	1,0183	-0,0766	1,0184	-0,0766
6	1,0700	-0,098	1,0700	-0,0984	1,0698	-0,0984	1,0700	-0,0984
7	1,0490	-0,130	1,0485	-0,1299	1,0485	-0,1300	1,0485	-0,1299
8	1,0900	-0,130	1,0900	-0,1299	1,0900	-0,1300	1,0900	-0,1299
9	1,0290	-0,149	1,0290	-0,1485	1,0289	-0,1486	1,0290	-0,1485
10	1,0280	-0,145	1,0280	-0,1445	1,0280	-0,1446	1,0280	-0,1445
11	1,0440	-0,124	1,0445	-0,1236	1,0445	-0,1236	1,0445	-0,1236
12	1,0690	-0,125	1,0689	-0,1252	1,0690	-0,1253	1,0689	-0,1252
13	1,0510	-0,121	1,0510	-0,1212	1,0510	-0,1213	1,0510	-0,1212
14	1,0200	-0,154	1,0200	-0,1544	1,0201	-0,1545	1,0200	-0,1544

5.3.2.2 Teste T2: Validação da heurística adotada para estimação dos TAPs

A heurística adotada para solucionar os problemas comentados em 4.3.6.7 e obter a estimação da posição do TAP de transformadores com o uso de estimadores baseados em MQPV pode ser inicialmente avaliada através da inclusão dos TAPs dos transformadores representados pelo ramos 8, 9 e 10.

Conforme os resultados mostrados na Tabela 18, todas as variáveis de estados foram adequadamente estimadas independente do estimador utilizado. Entretanto isto não é suficiente para afirmar que a heurística relativa à estimação de TAPs e adotada para os estimadores MQPV pode ser utilizada sem ressalvas. Deve-se, para tanto, verificar o comportamento destes estimadores frente à presença de ruídos e, principalmente, de erros grosseiros.

Tabela 17 – Valores medidos e estimados – Teste T2

#	Conjunto de Medidas				MQPV		MQPV - Desacoplado		MQP	
	Medida			Valor medido	4 Iterações		6 Iterações		3 Iterações	
	Tipo	ID			Valor Estimado	Resíduo	Valor Estimado	Resíduo	Valor Estimado	Resíduo
1	V	1		1,060E+00	1,060E+00	-3,519E-06	1,060E+00	-1,344E-05	1,060E+00	-7,665E-06
2	V	3		1,010E+00	1,010E+00	-2,904E-06	1,010E+00	3,209E-06	1,010E+00	-7,407E-06
3	V	5		1,018E+00	1,018E+00	5,016E-06	1,018E+00	8,325E-06	1,018E+00	-2,725E-06
4	V	11		1,045E+00	1,045E+00	-3,297E-06	1,045E+00	-5,849E-05	1,045E+00	1,405E-06
5	V	12		1,069E+00	1,069E+00	2,017E-06	1,069E+00	-1,291E-04	1,069E+00	3,939E-06
6	V	13		1,051E+00	1,051E+00	2,905E-05	1,051E+00	2,207E-04	1,051E+00	3,348E-05
7	V	14		1,020E+00	1,020E+00	-2,501E-05	1,020E+00	-3,646E-05	1,020E+00	-2,056E-05
8	P	1		1,448E+00	1,448E+00	1,196E-05	1,450E+00	-1,634E-03	1,448E+00	1,522E-05
9	P	2		1,830E-01	1,830E-01	7,763E-06	1,815E-01	1,540E-03	1,830E-01	9,498E-06
10	P	3		-5,420E-01	-5,420E-01	1,880E-06	-5,417E-01	-2,674E-04	-5,420E-01	9,400E-06
11	P	4		-4,780E-01	-4,780E-01	-1,476E-08	-4,786E-01	6,266E-04	-4,780E-01	-1,886E-06
12	P	6		2,880E-01	2,881E-01	-9,943E-05	2,944E-01	-6,424E-03	2,881E-01	-9,281E-05
13	P	13		-1,350E-01	-1,349E-01	-6,642E-05	-1,384E-01	3,398E-03	-1,349E-01	-6,972E-05
14	P	14		-1,490E-01	-1,491E-01	5,214E-05	-1,497E-01	7,384E-04	-1,490E-01	4,071E-05
15	Q	2		-9,222E-02	-9,222E-02	6,398E-07	-9,278E-02	5,564E-04	-9,224E-02	1,809E-05
16	Q	3		-8,444E-02	-8,444E-02	1,194E-06	-8,439E-02	-5,403E-05	-8,447E-02	2,738E-05
17	Q	4		-3,900E-02	-3,903E-02	3,120E-05	-3,885E-02	-1,500E-04	-3,905E-02	5,021E-05
18	Q	6		-1,276E-01	-1,254E-01	-2,214E-03	-1,259E-01	-1,656E-03	-1,254E-01	-2,200E-03
19	Q	13		-5,800E-02	-5,791E-02	-8,897E-05	-6,230E-02	4,296E-03	-5,790E-02	-9,883E-05
20	Q	14		-5,000E-02	-5,006E-02	6,115E-05	-4,997E-02	-2,877E-05	-5,004E-02	3,767E-05
21	t	1	2	1,050E+00	1,050E+00	1,155E-05	1,051E+00	-1,496E-03	1,050E+00	1,390E-05
22	t	2	3	4,658E-01	4,658E-01	-6,378E-06	4,658E-01	6,983E-05	4,658E-01	-7,239E-06
23	t	2	5	3,277E-01	3,277E-01	-4,342E-06	3,276E-01	8,223E-05	3,277E-01	-1,854E-06
24	t	3	4	-8,588E-02	-8,588E-02	6,473E-06	-8,569E-02	-1,897E-04	-8,589E-02	1,309E-05
25	t	4	5	-4,315E-01	-4,315E-01	5,336E-06	-4,323E-01	8,099E-04	-4,315E-01	6,465E-06
26	t	5	1	-3,900E-01	-3,900E-01	1,040E-05	-3,902E-01	1,429E-04	-3,900E-01	9,768E-06
27	t	5	6	2,020E-01	2,020E-01	-4,721E-06	2,022E-01	-1,933E-04	2,020E-01	-4,884E-08
28	t	6	11	1,694E-01	1,694E-01	3,375E-07	1,710E-01	-1,611E-03	1,694E-01	2,078E-06
29	t	6	12	9,989E-02	9,985E-02	4,815E-05	1,017E-01	-1,760E-03	9,984E-02	5,110E-05
30	t	6	13	2,209E-01	2,209E-01	2,036E-05	2,240E-01	-3,073E-03	2,209E-01	2,696E-05
31	t	7	4	-1,826E-01	-1,826E-01	-9,697E-06	-1,829E-01	2,992E-04	-1,826E-01	-1,084E-05
32	t	7	8	3,331E-16	3,603E-16	-2,719E-17	3,604E-16	-2,734E-17	3,603E-16	-2,718E-17
33	t	7	9	1,826E-01	1,826E-01	2,909E-06	1,829E-01	-2,634E-04	1,826E-01	2,700E-06
34	t	9	4	-1,040E-01	-1,040E-01	-9,100E-06	-1,042E-01	1,624E-04	-1,040E-01	-1,356E-05
35	t	9	10	-4,029E-02	-4,030E-02	1,288E-05	-4,001E-02	-2,762E-04	-4,031E-02	1,698E-05
36	t	9	14	3,174E-02	3,173E-02	1,211E-05	3,255E-02	-8,091E-04	3,172E-02	2,427E-05
37	t	10	11	-1,304E-01	-1,304E-01	3,447E-05	-1,297E-01	-7,190E-04	-1,304E-01	4,596E-05
38	t	12	13	3,766E-02	3,772E-02	-6,291E-05	3,808E-02	-4,237E-04	3,772E-02	-6,680E-05
39	t	13	14	1,199E-01	1,199E-01	-1,772E-05	1,198E-01	1,193E-04	1,199E-01	-1,830E-05
40	t	14	13	-1,175E-01	-1,175E-01	1,894E-05	-1,174E-01	-1,076E-04	-1,175E-01	1,948E-05
41	u	1	2	1,485E-01	1,485E-01	5,546E-06	1,490E-01	-5,266E-04	1,485E-01	2,676E-05
42	u	2	3	6,877E-02	6,877E-02	-1,438E-06	6,872E-02	4,822E-05	6,877E-02	-3,318E-06
43	u	2	5	4,179E-02	4,178E-02	8,335E-06	4,172E-02	7,094E-05	4,176E-02	2,455E-05
44	u	3	4	-1,032E-02	-1,032E-02	4,544E-07	-1,030E-02	-1,723E-05	-1,035E-02	2,418E-05
45	u	4	5	4,221E-02	4,218E-02	3,077E-05	4,199E-02	2,229E-04	4,221E-02	3,161E-06
46	u	5	1	-1,069E-01	-1,069E-01	1,730E-06	-1,069E-01	1,950E-05	-1,068E-01	-1,277E-05
47	u	5	6	1,990E-01	1,967E-01	2,268E-03	2,008E-01	-1,826E-03	1,967E-01	2,309E-03
48	u	6	11	5,810E-02	5,817E-02	-7,241E-05	5,889E-02	-7,852E-04	5,816E-02	-5,946E-05
49	u	6	12	-4,162E-02	-4,169E-02	6,674E-05	-4,173E-02	1,068E-04	-4,171E-02	8,689E-05
50	u	6	13	4,648E-02	4,651E-02	-2,778E-05	4,920E-02	-2,724E-03	4,649E-02	-8,135E-06
51	u	7	4	5,901E-02	5,905E-02	-3,808E-05	5,882E-02	1,901E-04	5,906E-02	-4,480E-05
52	u	7	8	-2,470E-01	-2,470E-01	-5,634E-15	-2,470E-01	2,759E-07	-2,470E-01	-4,857E-15
53	u	7	9	1,881E-01	1,881E-01	7,404E-07	1,881E-01	5,773E-06	1,881E-01	1,530E-05
54	u	9	4	-3,063E-02	-3,056E-02	-6,366E-05	-3,068E-02	5,207E-05	-3,048E-02	-1,455E-04
55	u	9	10	2,684E-02	2,681E-02	2,628E-05	2,704E-02	-2,070E-04	2,680E-02	3,810E-05
56	u	9	14	1,909E-02	1,911E-02	-2,178E-05	1,957E-02	-4,868E-04	1,909E-02	1,471E-06
57	u	10	11	-3,132E-02	-3,138E-02	6,026E-05	-3,094E-02	-3,856E-04	-3,141E-02	8,743E-05
58	u	12	13	5,416E-02	5,422E-02	-5,738E-05	5,556E-02	-1,401E-03	5,423E-02	-6,635E-05
59	u	13	14	3,617E-02	3,625E-02	-8,243E-05	3,569E-02	4,758E-04	3,625E-02	-8,186E-05
60	u	14	13	-3,122E-02	-3,130E-02	7,973E-05	-3,077E-02	-4,573E-04	-3,130E-02	7,908E-05
Graus de liberdade				30	SPQR	9,921E-02	SPQR	1,100E+00	SPQR	5,698E-03

Tabela 18 – Valores dos estados estimados no teste T2.

Barra	Estados reais		MQPV		MQPV-Desacoplado		MQP	
	V	φ [rad]	V	φ [rad]	V	φ [rad]	V	φ [rad]
1	1,0600	0,0000	1,0600	0,0000	1,0600	0,0000	1,0600	0,0000
2	1,0450	-0,0262	1,0450	-0,0262	1,0450	-0,0263	1,0450	-0,0262
3	1,0100	-0,1096	1,0100	-0,1096	1,0100	-0,1096	1,0100	-0,1096
4	1,0150	-0,0948	1,0146	-0,0948	1,0145	-0,0948	1,0146	-0,0948
5	1,0180	-0,0766	1,0184	-0,0766	1,0184	-0,0766	1,0184	-0,0766
6	1,0700	-0,0983	1,0700	-0,0984	1,0703	-0,0984	1,0700	-0,0984
7	1,0490	-0,1299	1,0485	-0,1299	1,0487	-0,1300	1,0485	-0,1299
8	1,0900	-0,1299	1,0900	-0,1299	1,0902	-0,1300	1,0900	-0,1299
9	1,0290	-0,1485	1,0289	-0,1485	1,0292	-0,1486	1,0289	-0,1485
10	1,0280	-0,1445	1,0280	-0,1445	1,0282	-0,1446	1,0280	-0,1445
11	1,0440	-0,1236	1,0445	-0,1236	1,0446	-0,1238	1,0445	-0,1236
12	1,0690	-0,1251	1,0689	-0,1252	1,0690	-0,1256	1,0689	-0,1252
13	1,0510	-0,1213	1,0510	-0,1212	1,0508	-0,1214	1,0510	-0,1212
14	1,0200	-0,1545	1,0200	-0,1544	1,0200	-0,1546	1,0200	-0,1544
TAP	TAP 4-7	0,978	TAP 4-7	0,978	TAP 4-7	0,978	TAP 4-7	0,978
TAP	TAP 4-9	0,969	TAP 4-9	0,969	TAP 4-9	0,969	TAP 4-9	0,969
TAP	TAP 5-6	0,932	TAP 5-6	0,932	TAP 5-6	0,932	TAP 5-6	0,932

5.3.2.3 Teste T3: Análise da capacidade de detecção e tratamento de erros grosseiros

Após comprovar a conformidade do modelo e de parte das rotinas dos estimadores de estados, se faz necessário analisar possíveis inconformidades nas rotinas de processamento de erros grosseiros. Para tanto foram adicionados ruídos com magnitude de até 3σ e erros grosseiros com magnitude de 8σ a 20σ, conforme as linhas destacadas de laranja na Tabela 19.

Os resultados obtidos, mostrados nas tabelas 19 e 20, indicam que todos os estimadores de estados conseguiram identificar e tratar todas as medidas contaminadas com erros grosseiros. A Tabela 19 relaciona para cada medida o valor real da grandeza extraído do caso de fluxo de potência, seu valor simulado, o valor estimado e seu respectivo desvio percentual em relação ao valor real. As linhas destacadas de laranja indicam quais são as medidas contaminadas com erros grosseiros e as linhas destacadas de cinza indicam o espalhamento destes erros. Pode-se notar que os estimadores MQPV permitem que a influência das medidas com erros grosseiros cause grandes desvios nas demais medidas fortemente correlacionadas. Logo se conclui que os estados resultantes destes estimadores não podem ser considerados sempre válidos, sob pena da aceitação de grandes desvios em outras medidas quando erros grosseiros estão presentes no conjunto. Entretanto tal efeito é drasticamente reduzido quando os erros grosseiros identificados são eliminados e o resultado final é refinado através de uma última iteração MQP.

Tabela 19 – Valores medidos e estimados – Teste T3

#	Conjunto de Medidas					MQPV		MQPV - Desacoplado		MQP	
	Medida			Valor real	Valor medido	6 Iterações		7 Iterações		8 Iterações	
	Tipo	ID				Valor Estimado	Resíduo (%)	Valor Estimado	Resíduo	Valor Estimado	Resíduo
1	V	1		1,060E+00	1,059E+00	1,060E+00	-0,047%	1,060E+00	-0,047%	1,059E+00	-0,066%
2	V	3		1,010E+00	1,008E+00	1,010E+00	-0,050%	1,010E+00	-0,050%	1,009E+00	-0,069%
3	V	5		1,018E+00	1,020E+00	1,018E+00	-0,029%	1,018E+00	-0,029%	1,019E+00	0,010%
4	V	11		1,045E+00	1,066E+00	1,047E+00	0,249%	1,048E+00	0,287%	1,043E+00	-0,153%
5	V	12		1,069E+00	1,067E+00	1,068E+00	-0,065%	1,068E+00	-0,065%	1,068E+00	-0,131%
6	V	13		1,051E+00	1,052E+00	1,051E+00	0,000%	1,051E+00	-0,019%	1,050E+00	-0,076%
7	V	14		1,020E+00	1,019E+00	1,020E+00	-0,039%	1,020E+00	-0,010%	1,019E+00	-0,118%
8	P	1		1,448E+00	1,386E+00	1,426E+00	-1,540%	1,427E+00	-1,450%	1,417E+00	-2,148%
9	P	2		1,830E-01	1,843E-01	1,863E-01	1,792%	1,865E-01	1,923%	1,865E-01	1,885%
10	P	3		-5,420E-01	-5,425E-01	-5,388E-01	-0,594%	-5,371E-01	-0,911%	-5,453E-01	0,611%
11	P	4		-4,780E-01	-4,802E-01	-4,810E-01	0,621%	-4,810E-01	0,626%	-4,781E-01	0,025%
12	P	6		2,880E-01	2,872E-01	2,871E-01	-0,299%	2,889E-01	0,299%	2,888E-01	0,267%
13	P	13		-1,350E-01	-1,328E-01	-1,360E-01	0,741%	-1,392E-01	3,111%	-1,313E-01	-2,756%
14	P	14		-1,490E-01	-1,454E-01	-1,507E-01	1,107%	-1,487E-01	-0,188%	-1,476E-01	-0,933%
15	Q	2		-9,222E-02	-9,212E-02	-9,073E-02	-1,611%	-9,092E-02	-1,411%	-9,358E-02	1,474%
16	Q	3		-8,444E-02	-8,379E-02	-8,671E-02	2,683%	-8,751E-02	3,632%	-8,727E-02	3,354%
17	Q	4		-3,900E-02	-3,918E-02	-3,937E-02	0,954%	-3,901E-02	0,031%	-3,887E-02	-0,333%
18	Q	6		-1,276E-01	-1,276E-01	-1,276E-01	0,055%	-1,256E-01	-1,560%	-1,266E-01	-0,792%
19	Q	13		-5,800E-02	-5,804E-02	-5,756E-02	-0,755%	-5,880E-02	1,381%	-5,501E-02	-5,160%
20	Q	14		-5,000E-02	-4,985E-02	-5,506E-02	10,128%	-5,633E-02	12,654%	-5,049E-02	0,972%
21	t	1	2	1,050E+00	1,048E+00	1,034E+00	-1,505%	1,035E+00	-1,448%	1,029E+00	-2,020%
22	t	2	3	4,658E-01	4,706E-01	4,625E-01	-0,719%	4,619E-01	-0,846%	4,641E-01	-0,374%
23	t	2	5	3,277E-01	3,233E-01	3,223E-01	-1,636%	3,231E-01	-1,404%	3,187E-01	-2,732%
24	t	3	4	-8,588E-02	1,449E-02	-8,590E-02	0,030%	-8,475E-02	-1,314%	-9,088E-02	5,832%
25	t	4	5	-4,315E-01	-4,373E-01	-4,388E-01	1,694%	-4,383E-01	1,580%	-4,407E-01	2,120%
26	t	5	1	-3,900E-01	-3,909E-01	-3,837E-01	-1,613%	-3,845E-01	-1,426%	-3,805E-01	-2,443%
27	t	5	6	2,020E-01	2,027E-01	2,023E-01	0,124%	2,012E-01	-0,386%	2,035E-01	0,728%
28	t	6	11	1,694E-01	1,734E-01	1,687E-01	-0,413%	1,750E-01	3,330%	1,723E-01	1,718%
29	t	6	12	9,989E-02	9,921E-02	9,830E-02	-1,596%	9,341E-02	-6,487%	1,007E-01	0,758%
30	t	6	13	2,209E-01	2,223E-01	2,225E-01	0,683%	2,217E-01	0,339%	2,193E-01	-0,729%
31	t	7	4	-1,826E-01	-1,814E-01	-1,831E-01	0,301%	-1,840E-01	0,772%	-1,807E-01	-1,030%
32	t	7	8	3,331E-16	3,344E-16	3,608E-16	8,314%	1,806E-16	-45,783%	3,592E-16	7,845%
33	t	7	9	1,826E-01	1,832E-01	1,838E-01	0,652%	1,842E-01	0,898%	1,838E-01	0,646%
34	t	9	4	-1,040E-01	-1,047E-01	-1,044E-01	0,375%	-1,048E-01	0,750%	-1,035E-01	-0,519%
35	t	9	10	-4,029E-02	-4,004E-02	-4,123E-02	2,343%	-3,984E-02	-1,114%	-3,967E-02	-1,524%
36	t	9	14	3,174E-02	3,197E-02	3,248E-02	2,319%	3,119E-02	-1,727%	3,024E-02	-4,713%
37	t	10	11	-1,304E-01	-1,295E-01	-1,328E-01	1,818%	-1,273E-01	-2,332%	-1,286E-01	-1,358%
38	t	12	13	3,766E-02	1,177E-01	3,817E-02	1,354%	4,144E-02	10,038%	3,561E-02	-5,447%
39	t	13	14	1,199E-01	1,199E-01	1,208E-01	0,784%	1,202E-01	0,217%	1,200E-01	0,050%
40	t	14	13	-1,175E-01	-1,190E-01	-1,184E-01	0,766%	-1,177E-01	0,213%	-1,175E-01	0,034%
41	u	1	2	1,485E-01	1,492E-01	1,497E-01	0,808%	1,489E-01	0,256%	1,475E-01	-0,694%
42	u	2	3	6,877E-02	6,881E-02	7,002E-02	1,824%	7,005E-02	1,869%	6,984E-02	1,566%
43	u	2	5	4,179E-02	2,420E-01	4,281E-02	2,451%	4,216E-02	0,905%	4,030E-02	-3,554%
44	u	3	4	-1,032E-02	-1,038E-02	-1,088E-02	5,426%	-1,156E-02	11,966%	-1,192E-02	15,464%
45	u	4	5	4,221E-02	4,251E-02	4,188E-02	-0,779%	4,103E-02	-2,793%	4,046E-02	-4,160%
46	u	5	1	-1,069E-01	-1,084E-01	-1,081E-01	1,142%	-1,074E-01	0,533%	-1,060E-01	-0,796%
47	u	5	6	1,990E-01	2,004E-01	1,998E-01	0,382%	2,000E-01	0,518%	2,010E-01	1,000%
48	u	6	11	5,810E-02	5,790E-02	4,802E-02	-17,343%	4,412E-02	-24,064%	6,044E-02	4,031%
49	u	6	12	-4,162E-02	-4,250E-02	-3,553E-02	-14,637%	-3,247E-02	-21,979%	-4,031E-02	-3,145%
50	u	6	13	4,648E-02	4,680E-02	5,112E-02	9,996%	5,434E-02	16,918%	4,567E-02	-1,738%
51	u	7	4	5,901E-02	5,803E-02	5,813E-02	-1,496%	5,804E-02	-1,652%	5,841E-02	-1,025%
52	u	7	8	-2,470E-01	-2,434E-01	-2,434E-01	-1,458%	-2,434E-01	-1,458%	-2,434E-01	-1,458%
53	u	7	9	1,881E-01	1,904E-01	1,904E-01	1,175%	1,904E-01	1,196%	1,904E-01	1,175%
54	u	9	4	-3,063E-02	-3,049E-02	-3,052E-02	-0,336%	-3,042E-02	-0,692%	-3,010E-02	-1,714%
55	u	9	10	2,684E-02	2,679E-02	2,362E-02	-12,003%	2,295E-02	-14,469%	2,657E-02	-0,995%
56	u	9	14	1,909E-02	1,908E-02	2,352E-02	23,215%	2,561E-02	34,191%	1,835E-02	-3,882%
57	u	10	11	-3,132E-02	-3,114E-02	-3,844E-02	22,732%	-3,962E-02	26,505%	-3,167E-02	1,095%
58	u	12	13	5,416E-02	5,364E-02	5,004E-02	-7,604%	4,716E-02	-12,919%	5,322E-02	-1,730%
59	u	13	14	3,617E-02	3,665E-02	3,699E-02	2,276%	3,609E-02	-0,205%	3,744E-02	3,537%
60	u	14	13	-3,122E-02	-3,074E-02	-3,196E-02	2,348%	-3,113E-02	-0,301%	-3,246E-02	3,965%
Graus de liberdade					30	SPQR	8,111E+01	SPQR	8,391E+01	SPQR	1,033E+01

Tabela 20 - Valores dos estados estimados no teste T3 sem refinamento do resultado.

Barra	Estados reais		MQPV		MQPV-Desacoplado		MQP	
	V	φ [rad]	V	φ [rad]	V	φ [rad]	V	φ [rad]
1	1,0600	0,0000	1,0595	0,0000	1,0595	0,0000	1,0593	0,0000
2	1,0450	-0,0262	1,0446	-0,0258	1,0446	-0,0259	1,0445	-0,0257
3	1,0100	-0,1096	1,0095	-0,1086	1,0095	-0,1085	1,0093	-0,1088
4	1,0150	-0,0948	1,0141	-0,0938	1,0142	-0,0939	1,0145	-0,0932
5	1,0180	-0,0766	1,0181	-0,0753	1,0181	-0,0754	1,0185	-0,0747
6	1,0700	-0,0983	1,0707	-0,0970	1,0708	-0,0971	1,0691	-0,0966
7	1,0490	-0,1299	1,0496	-0,1289	1,0502	-0,1292	1,0472	-0,1281
8	1,0900	-0,1299	1,0904	-0,1289	1,0910	-0,1292	1,0881	-0,1281
9	1,0290	-0,1485	1,0298	-0,1476	1,0304	-0,1479	1,0274	-0,1469
10	1,0280	-0,1445	1,0291	-0,1436	1,0298	-0,1441	1,0264	-0,1429
11	1,0440	-0,1236	1,0471	-0,1229	1,0475	-0,1244	1,0429	-0,1222
12	1,0690	-0,1251	1,0682	-0,1228	1,0682	-0,1215	1,0675	-0,1236
13	1,0510	-0,1213	1,0510	-0,1198	1,0508	-0,1196	1,0502	-0,1194
14	1,0200	-0,1545	1,0196	-0,1531	1,0199	-0,1529	1,0188	-0,1525
TAP	TAP 4-7	0,978	TAP 4-7	0,976	TAP 4-7	0,976	TAP 4-7	0,979
TAP	TAP 4-9	0,969	TAP 4-9	0,967	TAP 4-9	0,967	TAP 4-9	0,971
TAP	TAP 5-6	0,932	TAP 5-6	0,931	TAP 5-6	0,931	TAP 5-6	0,933

Tabela 21 – Estimativas das grandezas destacadas na tabela 19 após refinamento do resultado

#	Medidas		MQPV	
			6 Iterações	
	Valor real	Valor medido	Valor Estimado	Resíduo (%)
48	5,810E-02	5,790E-02	5,877E-02	1,146%
49	-4,162E-02	-4,250E-02	-3,960E-02	-4,846%
55	2,684E-02	2,679E-02	2,612E-02	-2,668%
56	1,909E-02	1,908E-02	1,905E-02	-0,210%
57	-3,132E-02	-3,114E-02	-3,270E-02	4,390%
58	5,416E-02	5,364E-02	5,295E-02	-2,227%
			SPQR	1,065E+01

5.3.2.4 Teste T4: Estimação paramétrica na presença de erros grosseiros

A inserção de um erro grosseiro próximo a uma barra que possua transformadores com comutação sobre carga possibilitar analisar se os estimadores implementados são capazes de identificar o erro, eliminá-lo e estimar os estados da rede de forma a minimizar a propagação ou o espalhamento deste erro. Sendo a barra n˚ 4 adjacente a dois LTC, a inserção de um erro grosseiro com magnitude de 8σ na medida de fluxo reativo entre a barra n˚ 4 e a barra n˚ 5 é suficiente para possibilitar tal análise. Os resultados são mostrados nas tabelas a seguir.

Tabela 22 – Valores medidos e estimados – Teste T4

#	Conjunto de Medidas					MQPV		MQPV - Desacoplado		MQP	
	Medida			Valor real	Valor medido	5 Iterações		7 Iterações		9 Iterações	
	Tipo	ID				Valor Estimado	Resíduo (%)	Valor Estimado	Resíduo	Valor Estimado	Resíduo
1	V	1		1,060E+00	1,059E+00	1,058E+00	-0,1604%	1,059E+00	-0,0849%	1,059E+00	-0,057%
2	V	3		1,010E+00	1,008E+00	1,009E+00	-0,1188%	1,009E+00	-0,0891%	1,009E+00	-0,069%
3	V	5		1,018E+00	1,020E+00	1,014E+00	-0,3928%	1,019E+00	0,0393%	1,018E+00	-0,010%
4	V	11		1,045E+00	1,066E+00	1,044E+00	-0,0957%	1,043E+00	-0,1628%	1,043E+00	-0,153%
5	V	12		1,069E+00	1,067E+00	1,068E+00	-0,1216%	1,068E+00	-0,1216%	1,068E+00	-0,131%
6	V	13		1,051E+00	1,052E+00	1,050E+00	-0,0666%	1,050E+00	-0,0666%	1,050E+00	-0,076%
7	V	14		1,020E+00	1,019E+00	1,019E+00	-0,0980%	1,019E+00	-0,0980%	1,019E+00	-0,118%
8	P	1		1,448E+00	1,386E+00	1,416E+00	-2,1892%	1,413E+00	-2,4171%	1,416E+00	-2,182%
9	P	2		1,830E-01	1,843E-01	1,864E-01	1,8798%	1,853E-01	1,2295%	1,864E-01	1,847%
10	P	3		-5,420E-01	-5,425E-01	-5,444E-01	0,4373%	-5,456E-01	0,6716%	-5,456E-01	0,670%
11	P	4		-4,780E-01	-4,802E-01	-4,795E-01	0,3096%	-4,904E-01	2,5879%	-4,779E-01	-0,013%
12	P	6		2,880E-01	2,872E-01	2,888E-01	0,2743%	2,870E-01	-0,3368%	2,887E-01	0,250%
13	P	13		-1,350E-01	-1,328E-01	-1,314E-01	-2,6889%	-1,281E-01	-5,1481%	-1,313E-01	-2,741%
14	P	14		-1,490E-01	-1,454E-01	-1,479E-01	-0,7584%	-1,448E-01	-2,7987%	-1,476E-01	-0,926%
15	Q	2		-9,222E-02	-9,212E-02	-8,511E-02	-7,7066%	-9,434E-02	2,2967%	-9,130E-02	-0,993%
16	Q	3		-8,444E-02	-8,379E-02	-7,715E-02	-8,6308%	-8,800E-02	4,2112%	-8,689E-02	2,895%
17	Q	4		-3,900E-02	-3,918E-02	-2,280E-02	-41,5513%	-4,027E-02	3,2590%	-4,005E-02	2,692%
18	Q	6		-1,276E-01	-1,276E-01	-1,276E-01	0,0235%	-1,153E-01	-9,6268%	-1,281E-01	0,431%
19	Q	13		-5,800E-02	-5,804E-02	-5,543E-02	-4,4328%	-5,435E-02	-6,2914%	-5,501E-02	-5,155%
20	Q	14		-5,000E-02	-4,985E-02	-5,131E-02	2,6160%	-5,152E-02	3,0420%	-5,049E-02	0,978%
21	t	1	2	1,050E+00	1,048E+00	1,029E+00	-1,9529%	1,027E+00	-2,1625%	1,028E+00	-2,029%
22	t	2	3	4,658E-01	4,706E-01	4,643E-01	-0,3199%	4,642E-01	-0,3435%	4,642E-01	-0,343%
23	t	2	5	3,277E-01	3,233E-01	3,188E-01	-2,6980%	3,159E-01	-3,5984%	3,185E-01	-2,787%
24	t	3	4	-8,588E-02	1,449E-02	-8,970E-02	4,4483%	-9,108E-02	6,0646%	-9,107E-02	6,049%
25	t	4	5	-4,315E-01	-4,373E-01	-4,377E-01	1,4321%	-4,526E-01	4,8804%	-4,407E-01	2,130%
26	t	5	1	-3,900E-01	-3,909E-01	-3,791E-01	-2,8101%	-3,782E-01	-3,0204%	-3,802E-01	-2,528%
27	t	5	6	2,020E-01	2,027E-01	2,044E-01	1,1930%	2,022E-01	0,1139%	2,035E-01	0,723%
28	t	6	11	1,694E-01	1,734E-01	1,723E-01	1,7656%	1,774E-01	4,7298%	1,722E-01	1,707%
29	t	6	12	9,989E-02	9,921E-02	1,009E-01	0,9881%	9,682E-02	-3,0723%	1,006E-01	0,738%
30	t	6	13	2,209E-01	2,223E-01	2,200E-01	-0,4300%	2,151E-01	-2,6523%	2,193E-01	-0,729%
31	t	7	4	-1,826E-01	-1,814E-01	-1,792E-01	-1,8349%	-1,804E-01	-1,1886%	-1,806E-01	-1,085%
32	t	7	8	3,331E-16	3,344E-16	3,595E-16	7,9233%	3,592E-16	7,8572%	3,592E-16	7,845%
33	t	7	9	1,826E-01	1,832E-01	1,831E-01	0,2574%	1,826E-01	0,0219%	1,838E-01	0,652%
34	t	9	4	-1,040E-01	-1,047E-01	-1,020E-01	-1,8940%	-1,033E-01	-0,7307%	-1,035E-01	-0,529%
35	t	9	10	-4,029E-02	-4,004E-02	-4,025E-02	-0,1067%	-3,834E-02	-4,8327%	-3,967E-02	-1,544%
36	t	9	14	3,174E-02	3,197E-02	3,005E-02	-5,3214%	2,741E-02	-13,6295%	3,026E-02	-4,653%
37	t	10	11	-1,304E-01	-1,295E-01	-1,300E-01	-0,2915%	-1,254E-01	-3,8349%	-1,286E-01	-1,373%
38	t	12	13	3,766E-02	1,177E-01	3,552E-02	-5,6669%	3,655E-02	-2,9530%	3,563E-02	-5,391%
39	t	13	14	1,199E-01	1,199E-01	1,204E-01	0,4420%	1,200E-01	0,0834%	1,200E-01	0,042%
40	t	14	13	-1,175E-01	-1,190E-01	-1,180E-01	0,4256%	-1,176E-01	0,0766%	-1,175E-01	0,034%
41	u	1	2	1,485E-01	1,492E-01	1,554E-01	4,6058%	1,470E-01	-1,0033%	1,493E-01	0,539%
42	u	2	3	6,877E-02	6,881E-02	6,609E-02	-3,8885%	7,024E-02	2,1377%	7,049E-02	2,513%
43	u	2	5	4,179E-02	2,420E-01	5,761E-02	37,8715%	3,876E-02	-7,2345%	4,209E-02	0,725%
44	u	3	4	-1,032E-02	-1,038E-02	-5,625E-03	-45,4975%	-1,230E-02	19,2133%	-1,091E-02	5,687%
45	u	4	5	4,221E-02	1,223E-01	1,004E-01	137,9181%	3,502E-02	-17,0283%	4,067E-02	-3,655%
46	u	5	1	-1,069E-01	-1,084E-01	-1,202E-01	12,4848%	-1,049E-01	-1,8531%	-1,076E-01	0,739%
47	u	5	6	1,990E-01	2,004E-01	2,015E-01	1,2362%	1,892E-01	-4,9045%	2,026E-01	1,819%
48	u	6	11	5,810E-02	5,790E-02	5,839E-02	0,5009%	5,838E-02	0,4750%	6,043E-02	4,007%
49	u	6	12	-4,162E-02	-4,250E-02	-3,954E-02	-4,9927%	-3,888E-02	-6,5832%	-4,031E-02	-3,159%
50	u	6	13	4,648E-02	4,680E-02	4,633E-02	-0,3249%	4,635E-02	-0,2862%	4,569E-02	-1,706%
51	u	7	4	5,901E-02	5,803E-02	7,577E-02	28,4003%	5,723E-02	-3,0097%	5,773E-02	-2,164%
52	u	7	8	-2,470E-01	-2,434E-01	-2,434E-01	-1,4576%	-2,434E-01	-1,4576%	-2,434E-01	-1,458%
53	u	7	9	1,881E-01	1,904E-01	1,904E-01	1,2172%	1,904E-01	1,2012%	1,904E-01	1,175%
54	u	9	4	-3,063E-02	-3,049E-02	-1,215E-02	-60,3454%	-3,252E-02	6,1776%	-3,073E-02	0,333%
55	u	9	10	2,684E-02	2,679E-02	2,634E-02	-1,8594%	2,623E-02	-2,2693%	2,656E-02	-1,040%
56	u	9	14	1,909E-02	1,908E-02	1,941E-02	1,7132%	1,937E-02	1,4774%	1,834E-02	-3,893%
57	u	10	11	-3,132E-02	-3,114E-02	-3,218E-02	2,7521%	-3,248E-02	3,6843%	-3,169E-02	1,178%
58	u	12	13	5,416E-02	5,364E-02	5,281E-02	-2,5000%	5,165E-02	-4,6290%	5,321E-02	-1,745%
59	u	13	14	3,617E-02	3,665E-02	3,723E-02	2,9559%	3,743E-02	3,4840%	3,745E-02	3,548%
60	u	14	13	-3,122E-02	-3,074E-02	-3,222E-02	3,1964%	-3,244E-02	3,8978%	-3,247E-02	3,978%
Graus de liberdade					30	SPQR	2,526E+01	SPQR	1,241E+01	SPQR	1,227E+01

Tabela 23 - Valores dos estados estimados no teste T4 com refinamento do resultado.

Barra	Estados reais		MQPV		MQPV-Desacoplado		MQP	
	V	φ [rad]	V	φ [rad]	V	φ [rad]	V	φ [rad]
1	1,0600	0,0000	1,0600	0,0000	1,0591	0,0000	1,0593	0,0000
2	1,0450	-0,0262	1,0400	-0,0257	1,0444	-0,0257	1,0444	-0,0257
3	1,0100	-0,1096	1,0100	-0,1090	1,0091	-0,1088	1,0092	-0,1088
4	1,0150	-0,0948	1,0100	-0,0934	1,0144	-0,0932	1,0142	-0,0932
5	1,0180	-0,0766	1,0100	-0,0740	1,0188	-0,0743	1,0183	-0,0746
6	1,0700	-0,0983	1,0700	-0,0961	1,0690	-0,0961	1,0695	-0,0966
7	1,0490	-0,1299	1,0500	-0,1280	1,0473	-0,1280	1,0478	-0,1280
8	1,0900	-0,1299	1,0900	-0,1280	1,0882	-0,1280	1,0887	-0,1280
9	1,0290	-0,1485	1,0300	-0,1470	1,0275	-0,1467	1,0279	-0,1467
10	1,0280	-0,1445	1,0300	-0,1430	1,0265	-0,1428	1,0270	-0,1427
11	1,0440	-0,1236	1,0400	-0,1220	1,0428	-0,1228	1,0436	-0,1221
12	1,0690	-0,1251	1,0700	-0,1230	1,0676	-0,1220	1,0676	-0,1234
13	1,0510	-0,1213	1,0500	-0,1190	1,0503	-0,1184	1,0505	-0,1193
14	1,0200	-0,1545	1,0200	-0,1520	1,0190	-0,1514	1,0190	-0,1523
TAP	TAP 4-7	0,978	TAP 4-7	0,980	TAP 4-7	0,979	TAP 4-7	0,979
TAP	TAP 4-9	0,969	TAP 4-9	0,978	TAP 4-9	0,969	TAP 4-9	0,970
TAP	TAP 5-6	0,932	TAP 5-6	0,929	TAP 5-6	0,934	TAP 5-6	0,932

Os resultados das tabelas 22 e 23 merecem alguns comentários adicionais. A SPQR dos métodos MQPV considera a ponderação variável ou a penalidade da medida, portanto a SPQR relativa ao estimador MQPV é próxima à SPQR do estimador MQP para este caso. Isto não significa que os resultados foram considerados satisfatórios, pelo contrário, os resultados obtidos do estimador MQPV mostram nítida dificuldade para método identificar e corrigir o erro grosseiro inserido na medida de fluxo de potência reativa entre as barras n˚ 4 e n˚ 5. Além de não identificar tal erro, o estimador MQPV considerou errônea a medida de magnitude de tensão da barra n˚ 5. Apesar da convergência, a maioria dos estados estimados não corresponde aos estados reais, ou seja, as estimativas são consideradas impróprias. Contudo os demais estimadores identificaram todas as medidas com erros grosseiros e obtiveram estados considerados razoáveis para o caso estudado. O resultado do estimador MQP foi mais aderente aos valores reais, como se pode notar através da comparação entre os desvios percentuais. Por sua vez, o estimador MQPV desacoplado – mesmo refinando o resultado – não foi capaz de evitar satisfatoriamente a propagação dos erros, uma vez que algumas medidas estimadas apresentam consideráveis desvios em relação aos valores reais.

Cabe ressaltar que a quantidade e a distribuição dos erros grosseiros para este sistema tornam-se fatores relevantes a serem considerados no estudo, uma vez que 8,33% das medidas

estão contaminadas com erros grosseiros e há uma grande concentração de tais erros próximos às barras 4 e 5.

Concluí-se que o estimador MQP proporciona melhores resultados quando erros de grande magnitude situam-se em medidas próximas aos LTCs que estão tendo suas respectivas posições de TAP estimadas. Isto se deve à correção ou à eliminação destas medidas durante o processo de estimação de estados. No caso do estimador MQPV desacoplado, os grandes desvios gerados pela existência de erros grosseiros não são perfeitamente minimizados pelo processo de refinamento do resultado, indicando a necessidade de aplicação de técnicas mais elaboradas. Por fim, o estimador MQPV carece de uma metodologia mais eficiente para a estimação de posições de TAPs de transformadores.

5.3.2.5 Teste T5: Erros grosseiros em medidas formadoras de pontos de alavancamento

A identificação e o cálculo dos pesos relativos aos pontos de alavancamento foram confirmados através do programa desenvolvidos por Pires [55] em seu estudo comparativo de estimadores robustos. Para avaliar o impacto de erros grosseiros sobre medidas formadoras de pontos de alavancamento, a medida de injeção de potência ativa da barra n˚ 2 foi contaminada por um erro de magnitude igual a 9σ. Entretanto a manutenção do erro na medida de fluxo de potência ativa entre as barras n˚ 2 e n˚ 5 tornaria os resultados questionáveis, uma vez que haveria uma concentração de erros grosseiros sobre as medidas da barra n˚ 2. Portanto excluiu-se a referida medida para analisar exclusivamente os efeitos do erro grosseiro sobre a referida medida formadora de ponto de alavancamento. Cabe salientar que os resultados foram obtidos mantendo-se as estimações das posições dos TAPs dos LTCs.

A Tabela 24 mostra parte dos resultados obtidos com a formatação utilizada nos estudos anteriores, entretanto foi acrescentada uma coluna referente aos pesos dos pontos de alavancamento. As tabelas 24 e 25 mostram os resultados obtidos neste estudo.

O estimador MPQ, como era esperado, não consegui identificar o erro na medida de injeção de potência ativa na barra n˚ 2, mas identificou as demais medidas com erros grosseiros. Ressalvando os resultados obtidos nos trabalhos [55] e [91], os quais não estimaram posições de TAPs, os resultados obtidos neste estudo mostram que os estimadores MQPV acoplado e desacoplado não só identificaram como errônea a referida medida de injeção formadora de ponto de alavancamento como também um conjunto de outras medidas não portadoras de erros grosseiros que eram formadoras de pontos de alavancamento. Tais medidas possuem pesos relativamente baixos, contribuindo significativamente para a equivocada identificação – mas a

maior contribuição pode ser atribuída à estimação dos TAPs dos LTCs. A inclusão de tais parâmetros como variáveis de estados reduz a redundância local de informações, contribui para o aumento do número de iterações e muda a dinâmica de convergência do processo. Durante o processo de estimação de estados, os resíduos das medidas relativamente próximas às barras terminais dos LTCs tendem a serem maiores devido à oscilação da posição de TAP em torno do valor real ou esperado. Caso tais medidas sejam formadoras de pontos de alavancamento e seus respectivos pesos sejam relativamente baixos, os estimadores baseados em MQPV irão ampliar os valores residuais e passarão a penalizar drasticamente tais medidas. Durante as iterações subseqüentes, estas medidas poderão ou não voltar ao segmento quadrático do estimador utilizado – dependendo da influência e do impacto da penalidade de tais medidas sobre os estados e as outras medidas. Caso se iniba a estimação dos TAPs dos LTCs, os estimadores baseados em MQPV identificarão única e exclusivamente os erros grosseiros simulados no estudo – corroborando os resultados obtidos em [55] e [91].

Com base nos resultados e nas observações supracitadas, pode-se concluir que o tratamento de medidas formadoras de pontos de alavancamento deve ser revista e adaptada a problemas cuja estimação de parâmetros se assemelhe a realizada neste teste. Os estimadores MQP ainda carecem de métodos confiáveis para a identificação e tratamento de erros grosseiros em medidas formadoras de pontos de alavancamento. Uma possível solução para ambos para os estimadores estudados seria o desenvolvimento de um método que analisasse os erros em medidas adjacentes àquelas identificadas como pontos de alavancamento, uma vez que todas as medidas adjacentes tendem a apresentar erros significativos.

Tabela 24 – Valores dos estados estimados no teste T5 com refinamento do resultado.

Barra	Estados reais		MQPV		MQPV-Desacoplado		MQP	
	V	φ [rad]	V	φ [rad]	V	φ [rad]	V	φ [rad]
1	1,0600	0,0000	1,0595	0,0000	1,0595	0,0000	1,0599	0,0000
2	1,0450	-0,0262	1,0445	-0,0265	1,0445	-0,0265	1,0451	-0,0257
3	1,0100	-0,1096	1,0092	-0,1102	1,0092	-0,1102	1,0092	-0,1119
4	1,0150	-0,0948	1,0143	-0,0950	1,0144	-0,0951	1,0140	-0,0975
5	1,0180	-0,0766	1,0183	-0,0765	1,0183	-0,0767	1,0180	-0,0793
6	1,0700	-0,0983	1,0802	-0,0981	1,0076	-0,0980	1,0691	-0,1015
7	1,0490	-0,1299	1,0590	-0,1293	0,9883	-0,1292	1,0472	-0,1339
8	1,0900	-0,1299	1,0994	-0,1293	1,0315	-0,1292	1,0882	-0,1339
9	1,0290	-0,1485	1,0394	-0,1476	0,9674	-0,1473	1,0274	-0,1526
10	1,0280	-0,1445	1,0384	-0,1437	0,9662	-0,1435	1,0265	-0,1486
11	1,0440	-0,1236	1,0546	-0,1233	0,9819	-0,1237	1,0429	-0,1275
12	1,0690	-0,1251	1,0791	-0,1242	1,0084	-0,1236	1,0675	-0,1284
13	1,0510	-0,1213	1,0614	-0,1204	0,9893	-0,1197	1,0502	-0,1244
14	1,0200	-0,1545	1,0307	-0,1529	0,9588	-0,1520	1,0188	-0,1578
TAP	TAP 4-7	0,978	TAP 4-7	0,969	TAP 4-7	1,053	TAP 4-7	0,979
TAP	TAP 4-9	0,969	TAP 4-9	0,961	TAP 4-9	1,045	TAP 4-9	0,971
TAP	TAP 5-6	0,932	TAP 5-6	0,926	TAP 5-6	1,017	TAP 5-6	0,932

Tabela 25 – Teste T5: Índices do estudo e os valores reais, medidos e estimados.

#	Conjunto de Medidas						MQPV		MQPV - Desacoplado		MQP	
	Medida			Valor real	Valor medido	Peso PA	11 Iterações		13 Iterações		7 Iterações	
	Tipo	ID					Estimativa	Resíduo (%)	Estimativa	Resíduo (%)	Estimativa	Resíduo (%)
1	V	1		1,060E+00	1,059E+00	8,129E-01	1,060E+00	-0,0472%	1,060E+00	-0,0472%	1,060E+00	-0,0094%
2	V	3		1,010E+00	1,008E+00	8,395E-01	1,009E+00	-0,0792%	1,009E+00	-0,0792%	1,009E+00	-0,0792%
3	V	5		1,018E+00	1,020E+00	7,247E-01	1,018E+00	-0,0098%	1,018E+00	-0,0098%	1,018E+00	-0,0393%
4	V	11		1,045E+00	1,066E+00	9,290E-01	1,055E+00	0,9670%	9,819E-01	-5,9895%	1,043E+00	-0,1532%
5	V	12		1,069E+00	1,067E+00	3,556E-01	1,079E+00	0,9543%	1,008E+00	-5,6600%	1,068E+00	-0,1310%
6	V	13		1,051E+00	1,052E+00	1,908E-01	1,061E+00	0,9895%	9,893E-01	-5,8754%	1,050E+00	-0,0761%
7	V	14		1,020E+00	1,019E+00	1,663E-01	1,031E+00	1,0490%	9,588E-01	-5,9961%	1,019E+00	-0,1176%
8	P	1		1,448E+00	1,465E+00	1,000E+00	1,454E+00	0,3867%	1,455E+00	0,4558%	1,440E+00	-0,5456%
9	P	2		1,830E-01	2,745E-01	1,364E-01	1,751E-01	-4,3443%	1,770E-01	-3,2568%	2,567E-01	40,2842%
10	P	3		-5,420E-01	-5,425E-01	1,000E+00	-5,465E-01	0,8229%	-5,459E-01	0,7196%	-5,545E-01	2,3044%
11	P	4		-4,780E-01	-4,802E-01	1,000E+00	-4,810E-01	0,6276%	-5,245E-01	9,7280%	-4,893E-01	2,3556%
12	P	6		2,880E-01	2,872E-01	1,000E+00	2,885E-01	0,1632%	2,560E-01	-11,0972%	2,889E-01	0,3264%
13	P	13		-1,350E-01	-1,328E-01	1,000E+00	-1,339E-01	-0,8148%	-1,201E-01	-11,0741%	-1,312E-01	-2,8296%
14	P	14		-1,490E-01	-1,454E-01	1,000E+00	-1,475E-01	-1,0000%	-1,286E-01	-13,7181%	-1,474E-01	-1,0872%
15	Q	2		-9,222E-02	-9,212E-02	1,000E+00	-9,341E-02	1,2882%	-9,361E-02	1,5029%	-9,348E-02	1,3641%
16	Q	3		-8,444E-02	-8,379E-02	1,000E+00	-8,685E-02	2,8552%	-8,734E-02	3,4284%	-8,588E-02	1,6994%
17	Q	4		-3,900E-02	-3,918E-02	1,000E+00	-4,753E-02	21,8641%	-1,249E-01	220,2308%	-3,765E-02	-3,4667%
18	Q	6		-1,276E-01	-1,276E-01	1,000E+00	-1,061E-01	-16,8391%	1,129E-01	-188,4682%	-1,271E-01	-0,3763%
19	Q	13		-5,800E-02	-5,804E-02	1,000E+00	-5,814E-02	0,2397%	-5,867E-02	1,1603%	-5,510E-02	-5,0034%
20	Q	14		-5,000E-02	-4,985E-02	1,000E+00	-4,997E-02	-0,0540%	-5,006E-02	0,1100%	-5,063E-02	1,2680%
21	t	1	2	1,050E+00	1,048E+00	2,427E-01	1,056E+00	0,6097%	1,056E+00	0,6192%	1,029E+00	-1,9815%
22	t	2	3	4,658E-01	4,706E-01	1,000E+00	4,674E-01	0,3456%	4,675E-01	0,3692%	4,809E-01	3,2459%
23	t	2	5	3,277E-01	3,233E-01	1,000E+00	3,253E-01	-0,7081%	3,264E-01	-0,3815%	3,469E-01	5,8721%
24	t	3	4	-8,588E-02	1,449E-02	1,000E+00	-8,882E-02	3,4259%	-8,815E-02	2,6503%	-8,389E-02	-2,3080%
25	t	4	5	-4,315E-01	-4,373E-01	7,295E-01	-4,375E-01	1,3881%	-4,367E-01	1,2004%	-4,314E-01	-0,0348%
26	t	5	1	-3,900E-01	-3,909E-01	1,000E+00	-3,893E-01	-0,1949%	-3,901E-01	0,0282%	-4,024E-01	3,1742%
27	t	5	6	2,020E-01	2,027E-01	1,000E+00	2,031E-01	0,5148%	1,706E-01	-15,5438%	2,056E-01	1,7771%
28	t	6	11	1,694E-01	1,734E-01	1,000E+00	1,725E-01	1,8778%	1,552E-01	-8,3673%	1,741E-01	2,8048%
29	t	6	12	9,989E-02	9,921E-02	1,000E+00	9,910E-02	-0,7969%	8,170E-02	-18,2135%	1,005E-01	0,5976%
30	t	6	13	2,209E-01	2,223E-01	1,000E+00	2,199E-01	-0,4707%	1,898E-01	-14,1079%	2,199E-01	-0,4526%
31	t	7	4	-1,826E-01	-1,814E-01	1,000E+00	-1,819E-01	-0,3451%	-1,550E-01	-15,1230%	-1,889E-01	3,4507%
32	t	7	8	3,331E-16	3,344E-16	1,000E+00	1,835E-16	-44,9065%	3,213E-16	-3,5218%	1,796E-16	-46,0744%
33	t	7	9	1,826E-01	1,832E-01	1,000E+00	1,834E-01	0,4436%	1,574E-01	-13,7897%	1,832E-01	0,3341%
34	t	9	4	-1,040E-01	-1,047E-01	1,000E+00	-1,038E-01	-0,2115%	-8,795E-02	-15,4437%	-1,064E-01	2,2594%
35	t	9	10	-4,029E-02	-4,004E-02	1,000E+00	-4,005E-02	-0,5808%	-3,222E-02	-20,0333%	-4,054E-02	0,6131%
36	t	9	14	3,174E-02	3,197E-02	1,000E+00	2,997E-02	-5,5860%	2,516E-02	-20,7246%	2,900E-02	-8,6326%
37	t	10	11	-1,304E-01	-1,295E-01	1,000E+00	-1,295E-01	-0,6903%	-1,108E-01	-15,0560%	-1,306E-01	0,1457%
38	t	12	13	3,766E-02	1,177E-01	1,000E+00	3,777E-02	0,3080%	3,943E-02	4,7189%	3,596E-02	-4,5118%
39	t	13	14	1,199E-01	1,199E-01	1,000E+00	1,201E-01	0,1501%	1,057E-01	-11,8349%	1,210E-01	0,9174%
40	t	14	13	-1,175E-01	-1,190E-01	1,000E+00	-1,177E-01	0,1873%	-1,035E-01	-11,8669%	-1,185E-01	0,8938%
41	u	1	2	1,485E-01	1,492E-01	9,570E-01	1,474E-01	-0,7205%	1,472E-01	-0,8754%	1,471E-01	-0,9360%
42	u	2	3	6,877E-02	6,881E-02	1,000E+00	6,970E-02	1,3640%	6,978E-02	1,4760%	7,084E-02	3,0204%
43	u	2	5	4,179E-02	4,195E-02	1,000E+00	3,961E-02	-5,2099%	3,938E-02	-5,7627%	3,941E-02	-5,6933%
44	u	3	4	-1,032E-02	-1,038E-02	1,000E+00	-1,220E-02	18,1668%	-1,262E-02	22,3137%	-1,237E-02	19,8333%
45	u	4	5	4,221E-02	4,251E-02	1,000E+00	3,999E-02	-5,2710%	4,017E-02	-4,8493%	3,883E-02	-8,0072%
46	u	5	1	-1,069E-01	-1,084E-01	1,000E+00	-1,051E-01	-1,6846%	-1,048E-01	-1,9279%	-1,040E-01	-2,7141%
47	u	5	6	1,990E-01	2,004E-01	1,000E+00	1,773E-01	-10,9296%	-4,581E-02	-123,0206%	2,009E-01	0,9397%
48	u	6	11	5,810E-02	5,790E-02	1,000E+00	5,827E-02	0,2960%	5,738E-02	-1,2478%	5,968E-02	2,7229%
49	u	6	12	-4,162E-02	-4,250E-02	1,000E+00	-4,135E-02	-0,6607%	-4,138E-02	-0,5766%	-4,020E-02	-3,4045%
50	u	6	13	4,648E-02	4,680E-02	1,000E+00	4,668E-02	0,4303%	4,712E-02	1,3878%	4,558E-02	-1,9299%
51	u	7	4	5,901E-02	5,803E-02	1,000E+00	6,492E-02	10,0085%	1,188E-01	101,3557%	5,956E-02	0,9388%
52	u	7	8	-2,470E-01	-2,434E-01	1,000E+00	-2,434E-01	-1,4617%	-2,421E-01	-1,9718%	-2,434E-01	-1,4576%
53	u	7	9	1,881E-01	1,904E-01	1,000E+00	1,904E-01	1,1853%	1,894E-01	0,6644%	1,904E-01	1,2012%
54	u	9	4	-3,063E-02	-3,049E-02	1,000E+00	-2,701E-02	-11,8033%	-2,697E-03	-91,1954%	-2,918E-02	-4,7279%
55	u	9	10	2,684E-02	2,679E-02	1,000E+00	2,676E-02	-0,2944%	2,656E-02	-1,0471%	2,701E-02	0,6484%
56	u	9	14	1,909E-02	1,908E-02	1,000E+00	1,907E-02	-0,1100%	1,883E-02	-1,3517%	1,907E-02	-0,1153%
57	u	10	11	-3,132E-02	-3,114E-02	1,000E+00	-3,121E-02	-0,3448%	-3,118E-02	-0,4629%	-3,062E-02	-2,2476%
58	u	12	13	5,416E-02	5,364E-02	1,000E+00	5,393E-02	-0,4247%	5,327E-02	-1,6507%	5,291E-02	-2,2988%
59	u	13	14	3,617E-02	3,665E-02	1,000E+00	3,608E-02	-0,2350%	3,595E-02	-0,6083%	3,693E-02	2,1098%
60	u	14	13	-3,122E-02	-3,074E-02	1,000E+00	-3,122E-02	0,0000%	-3,151E-02	0,9256%	-3,188E-02	2,0978%
Graus de liberdade						30	SPQR	1,460E+01	SPQR	1,273E+03	SPQR	2,139E+01

5.3.2.5 Teste T6: Restrições de igualdade

O processamento das restrições de igualdade através do método dos pesos com refinamento iterativo aplicado a EESP proposto por Gouvêa et alli [104] foi aplicado a este trabalho utilizando uma abordagem ligeiramente diferente. Ao invés de se ponderar as restrições de igualdade diretamente através de um peso previamente definido, optou-se por uma ponderação através da adoção de uma precisão fictícia menor que as precisões das demais medidas. Desta forma, calcula-se a ponderação das restrições de forma indireta, mas comum para todos os elementos de estimação que compõem o problema. Assim a precisão adotada para as restrições de igualdade foi igual a $1x10^{-4}$, resultando numa ponderação de $5x10^{7}$. Pode-se observar que tal ponderação não é excessivamente alta, sendo uma das grandes vantagens de adoção deste método.

O sistema de 14 barras do IEEE gera três restrições de igualdade relativas a injeções nulas de potência, as quais são as injeções da barra n° 7 e a injeção de potência ativa da barra n° 8. A inclusão de tais restrições permite simular um erro grosseiro na medida de fluxo de potência ativa entre as barras n° 7 e n° 8, visando dificultar as condições de aplicação do método restritivo supracitado.

Os resultados obtidos deste estudo são mostrados nas tabelas 26 e 27.

Tabela 26 – Valores dos estados estimados no teste T5 com refinamento do resultado.

	Estados reais		MQPV		MQPV-Desacoplado		MQP	
Barra	V	φ [rad]	V	φ [rad]	V	φ [rad]	V	φ [rad]
1	1,0600	0,0000	1,0595	0,0000	1,0592	0,0000	1,0593	0,0000
2	1,0450	-0,0262	1,0445	-0,0262	1,0445	-0,0257	1,0445	-0,0257
3	1,0100	-0,1096	1,0093	-0,1100	1,0093	-0,1087	1,0093	-0,1088
4	1,0150	-0,0948	1,0143	-0,0948	1,0145	-0,0931	1,0144	-0,0934
5	1,0180	-0,0766	1,0183	-0,0763	1,0185	-0,0745	1,0184	-0,0749
6	1,0700	-0,0983	1,0693	-0,0983	1,0690	-0,0963	1,0691	-0,0969
7	1,0490	-0,1299	1,0474	-0,1300	1,0471	-0,1281	1,0470	-0,1285
8	1,0900	-0,1299	1,0886	-0,1300	1,0884	-0,1281	1,0883	-0,1285
9	1,0290	-0,1485	1,0278	-0,1487	1,0275	-0,1466	1,0274	-0,1471
10	1,0280	-0,1445	1,0268	-0,1447	1,0266	-0,1428	1,0264	-0,1432
11	1,0440	-0,1236	1,0435	-0,1240	1,0428	-0,1229	1,0429	-0,1225
12	1,0690	-0,1251	1,0676	-0,1252	1,0676	-0,1223	1,0675	-0,1238
13	1,0510	-0,1213	1,0503	-0,1211	1,0503	-0,1185	1,0502	-0,1197
14	1,0200	-0,1545	1,0190	-0,1542	1,0190	-0,1514	1,0188	-0,1527
TAP	TAP 4-7	0,978	TAP 4-7	0,979	TAP 4-7	0,979	TAP 4-7	0,979
TAP	TAP 4-9	0,969	TAP 4-9	0,971	TAP 4-9	0,971	TAP 4-9	0,971
TAP	TAP 5-6	0,932	TAP 5-6	0,933	TAP 5-6	0,933	TAP 5-6	0,933

Tabela 27 – Teste T6: Índices do estudo e os valores reais, medidos e estimados.

#	Conjunto de Medidas					MQPV		MQPV - Desacoplado		MQP	
	Medida			Valor real	Valor medido	5 Iterações		7 Iterações		9 Iterações	
	Tipo	ID				Valor Estimado	Resíduo (%)	Valor Estimado	Resíduo	Valor Estimado	Resíduo
1	V	1		1,060E+00	1,059E+00	1,060E+00	-0,047%	1,059E+00	-0,075%	1,059E+00	-0,066%
2	V	3		1,010E+00	1,008E+00	1,009E+00	-0,069%	1,009E+00	-0,069%	1,009E+00	-0,069%
3	V	5		1,018E+00	1,020E+00	1,018E+00	-0,010%	1,019E+00	0,010%	1,018E+00	0,000%
4	V	11		1,045E+00	1,066E+00	1,044E+00	-0,096%	1,043E+00	-0,163%	1,043E+00	-0,153%
5	V	12		1,069E+00	1,067E+00	1,068E+00	-0,122%	1,068E+00	-0,122%	1,068E+00	-0,131%
6	V	13		1,051E+00	1,052E+00	1,050E+00	-0,067%	1,050E+00	-0,067%	1,050E+00	-0,076%
7	V	14		1,020E+00	1,019E+00	1,019E+00	-0,098%	1,019E+00	-0,098%	1,019E+00	-0,118%
8	P	1		1,448E+00	1,386E+00	1,444E+00	-0,283%	1,415E+00	-2,300%	1,419E+00	-2,010%
9	P	2		1,830E-01	1,843E-01	1,842E-01	0,628%	1,863E-01	1,792%	1,879E-01	2,694%
10	P	3		-5,420E-01	-5,425E-01	-5,464E-01	0,808%	-5,448E-01	0,507%	-5,444E-01	0,450%
11	P	4		-4,780E-01	-4,802E-01	-4,805E-01	0,531%	-4,785E-01	0,100%	-4,781E-01	0,010%
12	P	6		2,880E-01	2,872E-01	2,888E-01	0,292%	2,867E-01	-0,448%	2,888E-01	0,267%
13	P	7		0,000E+00	0,000E+00	4,012E-05	----------	4,085E-05	----------	4,879E-07	----------
14	P	8		0,000E+00	0,000E+00	6,321E-08	----------	1,586E-07	----------	3,404E-11	----------
15	P	13		-1,350E-01	-1,328E-01	-1,314E-01	-2,674%	-1,281E-01	-5,119%	-1,313E-01	-2,763%
16	P	14		-1,490E-01	-1,454E-01	-1,479E-01	-0,732%	-1,449E-01	-2,725%	-1,476E-01	-0,940%
17	Q	2		-9,222E-02	-9,212E-02	-9,328E-02	1,151%	-9,349E-02	1,381%	-9,332E-02	1,194%
18	Q	3		-8,444E-02	-8,379E-02	-8,673E-02	2,710%	-8,758E-02	3,717%	-8,717E-02	3,228%
19	Q	4		-3,900E-02	-3,918E-02	-3,982E-02	2,095%	-4,077E-02	4,546%	-3,847E-02	-1,356%
20	Q	6		-1,276E-01	-1,276E-01	-1,231E-01	-3,504%	-1,212E-01	-5,002%	-1,267E-01	-0,659%
21	Q	7		0,000E+00	0,000E+00	1,136E-03	----------	1,572E-03	----------	1,390E-05	----------
22	Q	13		-5,800E-02	-5,804E-02	-5,540E-02	-4,486%	-5,438E-02	-6,243%	-5,501E-02	-5,153%
23	Q	14		-5,000E-02	-4,985E-02	-5,125E-02	2,490%	-5,162E-02	3,246%	-5,049E-02	0,988%
24	t	1	2	1,050E+00	1,048E+00	1,048E+00	-0,210%	1,027E+00	-2,163%	1,030E+00	-1,924%
25	t	2	3	4,658E-01	4,706E-01	4,675E-01	0,369%	4,636E-01	-0,481%	4,642E-01	-0,350%
26	t	2	5	3,277E-01	4,538E-01	3,255E-01	-0,653%	3,180E-01	-2,933%	3,199E-01	-2,353%
27	t	3	4	-8,588E-02	1,449E-02	-8,863E-02	3,210%	-9,081E-02	5,743%	-8,992E-02	4,707%
28	t	4	5	-4,315E-01	-4,373E-01	-4,379E-01	1,481%	-4,415E-01	2,322%	-4,401E-01	1,977%
29	t	5	1	-3,900E-01	-3,909E-01	-3,883E-01	-0,451%	-3,798E-01	-2,623%	-3,816E-01	-2,169%
30	t	5	6	2,020E-01	2,027E-01	2,038E-01	0,896%	2,017E-01	-0,134%	2,036E-01	0,772%
31	t	6	11	1,694E-01	1,734E-01	1,719E-01	1,512%	1,766E-01	4,299%	1,723E-01	1,766%
32	t	6	12	9,989E-02	9,921E-02	1,009E-01	1,018%	9,693E-02	-2,966%	1,006E-01	0,748%
33	t	6	13	2,209E-01	2,223E-01	2,198E-01	-0,498%	2,149E-01	-2,738%	2,194E-01	-0,715%
34	t	7	4	-1,826E-01	-1,814E-01	-1,829E-01	0,197%	-1,812E-01	-0,750%	-1,820E-01	-0,307%
35	t	7	8	3,331E-16	2,000E-01	-6,321E-08	----------	-1,586E-07	----------	-3,404E-11	----------
36	t	7	9	1,826E-01	1,832E-01	1,830E-01	0,203%	1,812E-01	-0,745%	1,820E-01	-0,323%
37	t	9	4	-1,040E-01	-1,047E-01	-1,041E-01	0,087%	-1,031E-01	-0,846%	-1,036E-01	-0,394%
38	t	9	10	-4,029E-02	-4,004E-02	-4,004E-02	-0,608%	-3,807E-02	-5,498%	-3,971E-02	-1,430%
39	t	9	14	3,174E-02	3,197E-02	3,033E-02	-4,433%	2,790E-02	-12,105%	3,019E-02	-4,887%
40	t	10	11	-1,304E-01	-1,295E-01	-1,295E-01	-0,644%	-1,247E-01	-4,349%	-1,287E-01	-1,289%
41	t	12	13	3,766E-02	1,177E-01	3,544E-02	-5,877%	3,639E-02	-3,365%	3,562E-02	-5,404%
42	t	13	14	1,199E-01	1,199E-01	1,202E-01	0,242%	1,196E-01	-0,242%	1,200E-01	0,092%
43	t	14	13	-1,175E-01	-1,190E-01	-1,177E-01	0,221%	-1,172E-01	-0,247%	-1,176E-01	0,077%
44	u	1	2	1,485E-01	1,492E-01	1,476E-01	-0,633%	1,476E-01	-0,646%	1,477E-01	-0,559%
45	u	2	3	6,877E-02	6,881E-02	6,982E-02	1,533%	6,998E-02	1,765%	6,986E-02	1,592%
46	u	2	5	4,179E-02	4,195E-02	3,986E-02	-4,600%	4,027E-02	-3,623%	4,044E-02	-3,226%
47	u	3	4	-1,032E-02	-1,038E-02	-1,197E-02	15,987%	-1,201E-02	16,345%	-1,181E-02	14,388%
48	u	4	5	4,221E-02	4,251E-02	3,953E-02	-6,365%	3,985E-02	-5,596%	3,998E-02	-5,290%
49	u	5	1	-1,069E-01	-1,084E-01	-1,053E-01	-1,460%	-1,060E-01	-0,777%	-1,061E-01	-0,730%
50	u	5	6	1,990E-01	2,004E-01	1,969E-01	-1,035%	1,954E-01	-1,824%	2,011E-01	1,075%
51	u	6	11	5,810E-02	5,790E-02	5,860E-02	0,855%	5,846E-02	0,625%	6,041E-02	3,971%
52	u	6	12	-4,162E-02	-4,250E-02	-3,958E-02	-4,904%	-3,895E-02	-6,413%	-4,031E-02	-3,157%
53	u	6	13	4,648E-02	4,680E-02	4,634E-02	-0,295%	4,633E-02	-0,321%	4,567E-02	-1,745%
54	u	7	4	5,901E-02	5,803E-02	5,828E-02	-1,239%	5,845E-02	-0,946%	5,693E-02	-3,532%
55	u	7	8	-2,470E-01	-2,434E-01	-2,455E-01	-0,611%	-2,454E-01	-0,660%	-2,454E-01	-0,660%
56	u	7	9	1,881E-01	1,904E-01	1,883E-01	0,096%	1,885E-01	0,175%	1,884E-01	0,159%
57	u	9	4	-3,063E-02	-3,049E-02	-2,882E-02	-5,916%	-2,898E-02	-5,368%	-2,968E-02	-3,089%
58	u	9	10	2,684E-02	2,679E-02	2,622E-02	-2,303%	2,620E-02	-2,381%	2,659E-02	-0,924%
59	u	9	14	1,909E-02	1,908E-02	1,921E-02	0,629%	1,936E-02	1,404%	1,838E-02	-3,715%
60	u	10	11	-3,132E-02	-3,114E-02	-3,246E-02	2,139%	-3,255E-02	3,661%	-3,162E-02	0,218%

61	u	12	13	5,416E-02	5,364E-02	5,288E-02	2,139%	5,178E-02	3,661%	5,321E-02	0,218%
62	u	13	14	3,617E-02	3,665E-02	3,736E-02	2,139%	3,752E-02	3,661%	3,742E-02	0,218%
63	u	14	13	-3,122E-02	-3,074E-02	-3,237E-02	3,667%	-3,256E-02	4,292%	-3,244E-02	3,882%
Graus de liberdade					33	SPQR	1,330E+02	SPQR	2,581E+02	SPQR	1,430E+01

A análise dos dados relacionados nas tabelas 26 e 27 mostram que, mesmo com a existência um erro grosseiro entre as barras com injeção de potência nula, a aplicação do método dos pesos com refinamento iterativo possibilitou manter os valores destas injeções próximos de zero. Os resultados obtidos para o estimador MQP foram nitidamente mais baixos, entretanto àqueles obtidos pelos estimadores MQPV também podem ser considerados satisfatórios. Outra informação importante é a contribuição deste método à SPQR, uma vez que os resíduos associados às restrições compõem este índice e, mesmo pequenos desvios, agregam significativas contribuições devido à ponderação utilizada.

5.3.3 Conclusões obtidas através dos testes com o sistema IEEE de 14 barras

Baseando nos resultados obtidos a partir da seqüência de teste com o sistema de 14 barras do IEEE, pode-se concluir que:

- Não existem erros, impropriedades ou inconformidades relacionadas aos modelos de componentes do sistema e às rotinas dos estimadores implementados neste trabalho;
- Os estimadores sob análise são capazes de identificar e corrigir múltiplos erros grosseiros. Entretanto os estimadores MQPV necessitam da aplicação adicional de métodos de refinamento de resultados por possibilitar que a correlação entre os resíduos e os erros de medição cause o espalhamento de erros grosseiros, gerando penalidades impróprias em outras medidas. Ou seja, sendo:

  $\underline{\hat{r}} = S \cdot \underline{e}$, onde $\underline{e}$ é o vetor contendo os erros de medição.

  A existência de um erro do tipo grosseiro e_i irá contribuir com o resíduo $\hat{r}_j$ na proporção $S_{ji} \cdot e_i$, possibilitando a propagação do erro e, consequentemente, a aplicação de possíveis penalidades a outras medidas. Entretanto deve-se ressaltar que a propagação do erro é facilitada pela convergência oscilatória das variáveis de estados que representam ou estão diretamente relacionadas às posições de TAPs dos LTCs.
- A estimação da posição dos TAPs dos transformadores com comutação sobre carga pode ser realizada através do emprego dos estimadores MQP e MQPV desacoplado. Em relação ao estimador MQPV desacoplado, deve-se utilizar um método de refino de resultados mais aprimorado que o atualmente implementado. Sobretudo deve-se evitar o

uso do estimador MQPV, sob pena de aceitação de erros grosseiros em medidas adjacentes as barras terminais do LTC cuja posição de TAP se esteja estimando.

- O tratamento de erros grosseiros em medidas formadoras de ponto de alavancamento deve ser realizado com ressalvas para os estimadores baseados em MQPV quando as posições de TAPs estão sendo estimadas, uma vez que a convergência oscilatória destas variáveis de estado pode resultar na identificação equivocada de erros grosseiros em medidas formadoras de ponto de alavancamento. Em relação aos estimadores baseados em MQP, devem ser formulados métodos e técnicas capazes de lidar de forma abrangete com este problema;
- O uso do método dos pesos com refinamento iterativo é aderente ao problema de EESP, independente do método de estimação adotado.

5.4 Analisando o desempenho em sistemas de médio e grande porte

O objetivo da presente subseção é analisar o desempenho dos estimadores de estados implementados utilizando-se o esquema de fatoração formado pela rotações de Givens com 3 multiplicadores e ordenação com MDA e R5. Devido a diferença entre os melhores esquemas ser pequena e não ser o intuto desta subseção refazer os testes já mostrados em 5.1, não há motivos de se fazer sequencias exaustivas de testes. Os sistemas utilizados serão aqueles relacionados na Tabela 5, sendo que todos os sistemas foram obtidos de estudos de fluxo de potência. A partir dos resultados destes estudos, ruídos e erros grosseiros foram inseridos através de um simulador de medidas. Entretanto alguns problemas na geração de medidas de injeção de fluxo de potência reativa em barras com bancos de capacitores ou reatores prejudicaram os resultados. Para o caso de 118, 340 e 730 barras, os estimadores conseguem filtrar o conjunto de erros,mas para o sistema de 2000 barras o processo é divergente. A configuração dos estimadores é semelhante é mostrada Tabela 13, desconsiderando o cálculo dos pontos de alavancamento e o processamento das restrições de. A Tabela 28 relacionada os resultados obtidos.

Tabela 28 – Desempenho dos estimadores em sistemas de médio e grande porte

	Estimador MQPV				Estimador MQPV – DR				Estimador MQP			
Sistema	118	340	730	2000	118	340	730	2000	118	340	730	2000
Iterações	4	4	26	50	6	19	38	50	3	6	8	18
Tempo (s)	0,079	0,329	4,329	12,73	0,048	0,25	2,864	9,485	0,109	0,726	3,302	8,031

O estimador MQP, devido a estratégia de se processar sequencialmente os erros grosseiro através de iterações sucessivas sem reinício das variáveis de estado, acaba obtendo

convergências inválidas para os sistemas de maior porte. Neste caso deve-se implementar métodos para a detecção e o tratamento de múltiplos erros grosseiros [93]. Contudo o desempenho computacional dos estimadores pode ser considerado satisfatório. Nota-se que o estimador MQP é relativamente mais lento que os demais, mas considerando que a matriz jacobiana foi recalculada e fatorada a cada iteração. O estimador MQP é o único que não necessita fatorar a matriz Jacobiana quando ela passa a ser constante, portanto seu desempenho pode ser significativamente melhorado.

5.5 Aplicação de estimadores robustos a sistemas realísticos: O caso da CEEE

Esta seção pretende registrar a experiência da aplicação de estimadores robustos baseados em MQPV em sistemas de transmissão realísticos, mais especificamente no sistema de transmissão da Companhia Estadual de Energia Elétrica do Estado do Rio Grande do Sul – CEEE. O principal objetivo é mostrar em linhas gerais quais foram as dificuldades encontradas durante o desenvolvimento e a implantação do estimador robusto no Centro de Operação de Sistemas da CEEE, mostrando os principais detalhes técnicos, as características do modelo utilizado pela CEEE e os diversos fatores que tornam possível a aplicação deste estimador. Não é o intuito deste trabalho mostrar uma quantidade exaustiva de resultados, mas tão somente apontar para os problemas encontrados através de resultados parciais e relatos sobre as situações encontradas – repassando para mundo acadêmico algumas informações que devem balizar futuras pesquisa e teorias.

Inicialmente, na subseção 5.5.1, o sistema de transmissão da CEEE será introduzido para possibilitar uma visão geral sobre o objeto desta análise. Conjuntamente com a apresentação do sistema da CEEE, será comentado o conjunto de ferramentas que formava a seqüência lógica e operativa da Companhia quando se iniciou o desenvolvimento dos estimadores no âmbito do projeto de P&D da CEEE do ciclo de 2002/2003. Posteriormente, na subseção 5.5.2, as características não comentadas do modelo barra-ramo do sistema de transmissão serão detalhadas em conjunto com os resumos dos resultados obtidos das aplicações dos estimadores desenvolvidos sobre a base histórica da CEEE. A seção 5.5.3, por sua vez, irá relatar um conjunto de observações e de críticas baseadas situações constatadas na CEEE e nas práticas recomendadas pela literatura técnica.

5.5.1 O sistema de transmissão de energia elétrica da CEEE

O sistema de transmissão apresentado neste trabalho refere-se à configuração utilizada no COS da CEEE durante o período de agosto a outubro de 2003. Apesar de estar relativamente defasado em relação ao sistema atual, acumulando expansões de quase 4 anos, as principais características abordadas neste trabalho continuam presentes na versão de março de 2007. Optou-se pela apresentação deste modelo por ser aquele que foi utilizado mais frequentemente durante a fase de implementação dos estimadores. A Figura 89 mostra modelo barra-ramo completo do referido sistema de transmissão da CEEE.

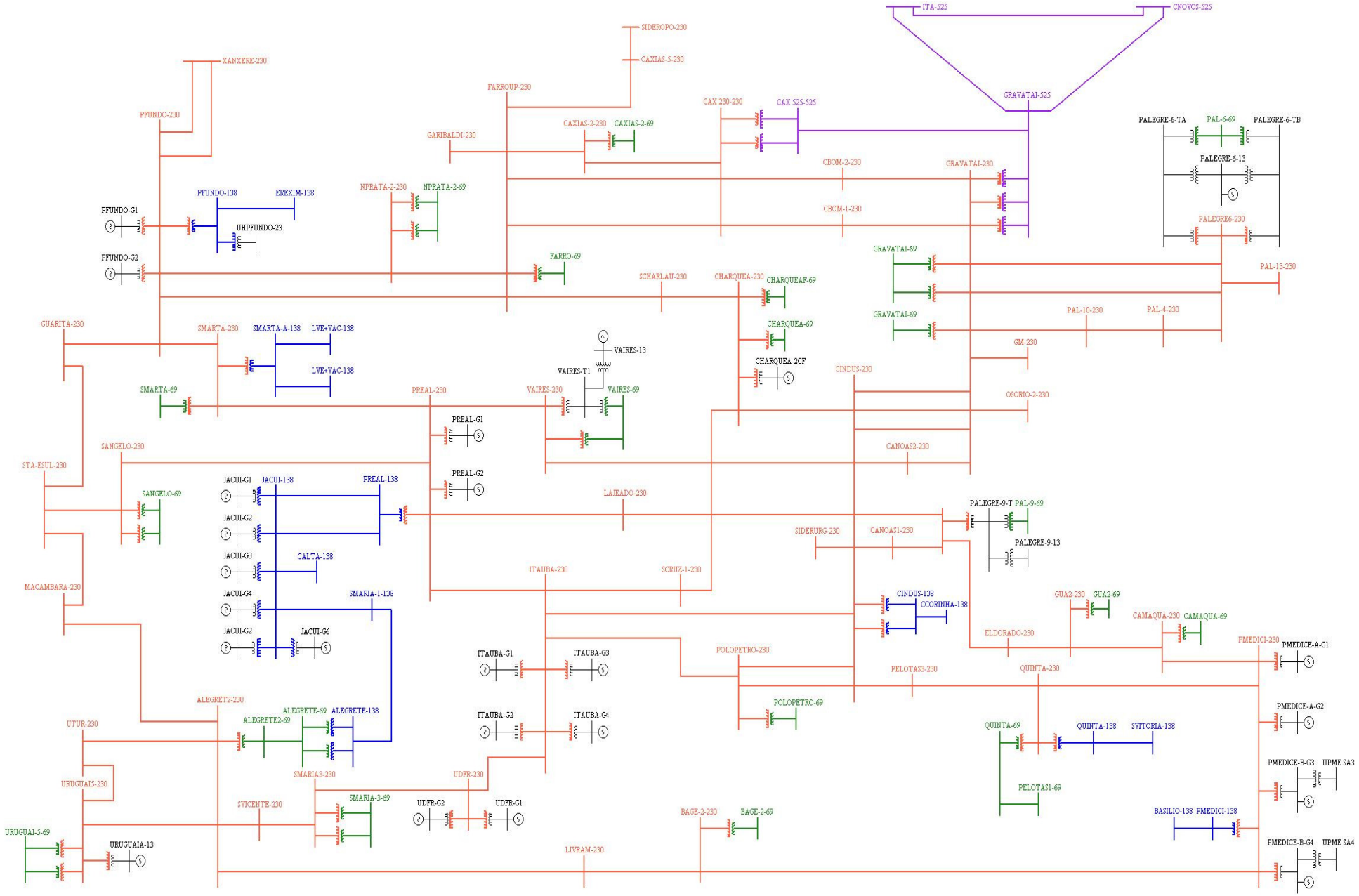


Figura 89 – Modelo barra-ramo manobrável relativo ao sistema de transmissão da CEEE durante o período de agosto a outubro de 2003

O sistema de transmissão da CEEE é relativamente pequeno, possuindo um total de 126 barras e 167 ramos. O modelo barra-ramo é apresentado de forma completa, entretanto isso não significa que todos seus elementos estarão operando ao mesmo tempo. Trata-se de um modelo barra-ramo manobrável, onde estão inseridos disjuntores equivalentes baseados na combinação de estados dos disjuntores reais do sistema. Tais disjuntores equivalentes eram formados pelo antigo configurador de redes da CEEE e possibilitavam que o modelo barra-ramo relativo ao sistema em operação fosse montado pelo estimador de estados desenvolvido pelo CEPEL. Os detalhes mais aprofundados sobre os aspectos computacionais e operacionais serão discutidos adiante, após alguns comentários que se fazem necessários sobre o modelo do sistema de transmissão completo.

Conforme mostra a Figura 89, os componentes do modelo são diferenciados por cores. Pode-se notar que existem 4 barras de 525kV destacadas de roxo que formam o anel principal de interconexão com o sistema elétrico nacional. Dentre tais barras, destacam-se a barra de Gravataí e Caxias por situarem-se nas principais subestações transformadoras do sistema de transmissão da CEEE. A barra de Campos Novos (CNOVOS-525) também merece destaque por ser a barra de referência do modelo.

A maior parte das barras mostradas no modelo forma o sistema de transmissão secundário de 230kV, destacado de vermelho, cujos pontos de maior importância são as interconexões com o anel de 525kV através dos LTCs das subestações de Caxias e Gravataí, as interconexões com os sistemas elétricos dos paises vizinhos através das barras de Uruguaina e Livravamento (situadas o parte inferior esquerda da Figura 89) e as interconexões com os diversos grupos geradores abrangidos pelo sistema de transmissão da CEEE. O sistema secundário de 230kV é composto por 51 barras e 127 ramos, dos quais 69 são linhas de transmissão, 19 são transformadores e 33 são LTCs.

A interconexão com os sistemas de distribuição é realizada, em sua maioria, através de segmentos em 138kV (azul) e 69kV (verde), com exceção parcial da região metropolitana de Porta Alegre e da cidade de Caixas. Os grupos geradores fornecem energia diretamente ao sistema de 230 kV, com exceção do grupo gerador de Jacuí. O sistema de 138kV pertencente ao sistema de transmissão da CEEE é composto por 16 barras e 30 ramos, dos quais 13 são linhas de transmissão, 7 são transformadores e 10 são LTCs. Por sua vez, o sistema de 69kV é composto por 22 barras e 30 ramos.

As demais barras do sistema ou são barras terminais de 13.8kV dos grupos geradores ou são barras fictícias originadas através do modelo em Y aplicado a alguns transformadores de 3 enrolamentos.

O consumo mínimo do sistema ocorre por volta das 4 horas dos domingos e gira em torno de 1700MW. Por sua vez, a consumo médio durante o horário de pico é de 3500 MW e ocorre por volta

das 19 horas e 30 minutos dos dias úteis. O consumo médio durante os dias úteis é aproximadamente 2700 MW.

5.5.1.1 – O sistema de telemetria da CEEE

O sistema de telemetria da CEEE é composto, em sua maioria, por canais de comunicação via satélite (microondas), mas também existem outros tipos de canais de comunicação, como – por exemplo – canais de radiodifusão. A confiabilidade destes meios de comunicação é considerada satisfatória, entretanto existem problemas relacionados direta e indiretamente com o sistema de telemetria – os quais foram classificados da seguinte maneira:

- *Perda de canal de comunicação*: Ocorre quando um dos canais de comunicação entre as subestações e o centro de operação do sistema é perdido por algum motivo técnico ou natural. Ocorre com baixa freqüência e, na maioria das vezes, afeta os canais de comunicação via satélite. Tais perdas são temporárias e estão relacionadas a grandes tempestades;
- *Medidas intermitentes*: Algumas medidas apresentam ocasionalmente erros grosseiros ou não são relacionadas no quadro de varredura do sistema de aquisição de dados de tal forma que, durante um intervalo de tempo, a falha ocorre ora sim ora não em intervalos menores e aproximadamente iguais;
- *Erros permanentes*: Algumas medidas apresentam erros grosseiros de forma continuada.

O conjunto de elementos digitais e analógicos oriundos do sistema de telemetria é processado diretamente por processos dedicados que implementam os protocolos de comunicação da família *IEC 60870-5* e *IEC 60870-6*. Tais processos são voltados à operação em tempo real, sendo responsáveis por gerar periodicamente, a cada 2 segundos, os arquivos com a relação de todos os elementos analógicos e digitais existentes. A omissão de determinado elemento digital ou analógico durante um número pré-determinado de períodos subseqüentes é considerada uma falha, mantendo – para fins de registro – o último valor obtido do estado digital ou da medida analógica e alterando seu status para *falhado*.

A gravação das informações do sistema de telemetria na base de dados histórica do sistema ocorre a cada minuto, sendo os dados gravados segundo uma estrutura definida diariamente para o conjunto de informações disponíveis. Cada gravação corresponde a uma *foto* do sistema de telemetria, sendo que para o mesmo dia todas as fotos devem obrigatoriamente possuir a mesma estruturação. A base história do sistema de telemetria é formada por um conjunto de arquivos diários nomeados conforme o dia a qual se referem e gravados numa estrutura de diretórios nomeada conforme o mês e o ano do respectivo arquivo.

5.5.1.2 – Comentários sobre o plano de medição da CEEE

O plano de medição do sistema de transmissão da CEEE é composto por medidas de injeção de potência ativa e reativa, medidas de fluxo de potência ativa e reativa e medidas de magnitude de tensão, conforme a quantificação mostrada na Tabela 29. Cabe ressaltar que parte destes medidores é obtida através da composição de elementos analógicos do sistema de telemetria, onde alguns elementos compositores formam outras medidas. Neste caso há o correlacionamento de medidores, gerando-se fonte de problemas relacionados à identificação de erros grosseiros.

Tabela 29 – Quantidade e tipos de medidores alocados no sistema de transmissão da CEEE

Sistema de transmissão da CEEE	
Medidor	Qtd
Magnitude de tensão (V)	73
Injeção de potência Ativa (P)	43
Injeção de potência Ativa (Q)	45
Fluxo de potência Ativa (t)	191
Fluxo de potência Reativa (u)	191

Apesar da quantidade de medidores ser considerada satisfatória, sua distribuição não favorece a observação de todas as barras do sistema. Muitos medidores estão concentrados junto a LTCs graças à imposição do Operador Nacional do Sistema Elétrico (ONS), o qual exigiu medidas de fluxo ativo e reativo em todos os LTCs do sistema interligado brasileiro. Entretanto algumas partes do sistema de transmissão da CEEE necessitam de uma quantidade de informação local mais adequada, conforme o levantamento de medidas críticas realizado em fotos de setembro de 2003. A Tabela 30 mostra a quantidade de medidas críticas no caso que será estudado nas subseções subseqüentes.

Tabela 30 – Medidas críticas do retiradas do sistema em operação no dia 04/09/2003 às 01:01:00hs

Local		**Medida**	**Local**			**Medida**
BARRA	1176	P	BARRA	1247		V
BARRA	1192	P	BARRA	1251		V
BARRA	1179	Q	BARRA	1257		V
BARRA	1181	Q	BARRA	1275		V
BARRA	1192	Q	BARRA	1279		V
BARRA	1042	V	BARRA	1284		V
BARRA	1189	V	BARRA	1295		V
BARRA	1195	V	BARRA	1298		V
BARRA	1197	V	BARRA	2077		V
BARRA	1201	V	BARRA	2087		V
BARRA	1202	V	LT	964	955	td
BARRA	1203	V	LT	1238	2057	td
BARRA	1207	V	LT	964	955	ud
BARRA	1209	V	LT	1238	2057	ud
BARRA	1226	V	TRAFO	1296	1068	td
BARRA	1238	V	TRAFO	1296	1068	ud
BARRA	1245	V				

Outro fator de extrema importância em sistemas realísticos é a dinâmica do plano de medição. Conforme a configuração operativa do sistema elétrico e o conjunto de falhas relacionadas ao sistema de telemetria, há a variações na quantidade de informação disponíveis. Logo a análise do plano de medição não pode ser baseada única e exclusivamente no plano de medição completo do sistema de transmissão. As medidas e os conjuntos críticos levantados através do estudo de observabilidade do plano completo de medição continuarão válidos para qualquer modificação ulterior. Entretanto nada poderá ser afirmado sobre a formação de medidas e conjuntos críticos durante a configuração do sistema em operação.

As restrições de igualdade do sistema de transmissão da CEEE, obtidas do conhecimento das barras de transição ou passagem, ajudam a aumentar a redundância de informações sobre o sistema. Ao todo são 33 restrições relativas a injeções nulas de potência ativa e 30 restrições relativas a injeções nulas de potência reativa para o sistema estudo neste capítulo. Entretanto, como serão comentados posteriormente, alguns modelos de componentes de rede ou erros de configuração do sistema dificultam demasiadamente o processo restritivo.

O fator mais relevante do plano de medição da CEEE que deve ser registrado neste trabalho recai sobre a precisão dos medidores que é utilizada na estimação dos estados do sistema de transmissão. Relatos dos profissionais e consultores da CEEE indicam que antes da desregulamentação do mercado de energia elétrica e da divisão da empresa, a aferição dos medidores do sistema era feita de global para todo o sistema de transmissão do Estado do Rio Grande do Sul. Durante este período os pesos associados às medidas baseavam-se nas precisões obtidas em campo. Entretanto, depois da desregulamentação do mercado, da conseqüente redução do quadro de funcionários e da fragmentação da CEEE, o plano de manutenção e aferição de medidores passou a sofrer alguns problemas de ordem administrativa e gerencial por parte das empresas com compõem a antiga estrutura da CEEE. Como conseqüência houve uma notável perda de qualidade do serviço, refletindo na operação do sistema de transmissão com o passar do tempo e com a expansão da rede elétrica. Atualmente a CEEE utiliza um conjunto de pesos obtidos de forma empírica, os quais contribuem significativamente e de forma direta para os problemas enfrentados pelo estimador de estados da companhia. Apesar das tentativas de se estimar os estados do sistema com o estimador MQP desenvolvido, a falta de informações confiáveis – as quais formam a base do método matemático aplicado à estimação de estados – impossibilitou qualquer convergência que pudesse ser considerada válida. Será visto nas seções subseqüentes que o estimador baseado em MQPV, apesar de todos os problemas levantados durante a fase de testes e validação, pode ser aplicado a este tipo de sistema, ou melhor, sua aplicação deixa de ser questionável e passa a ser aconselhável. Por óbvio não se pretende obter um conjunto de estados que reflitam o ponto operativo real do sistema, mas podem ser obtidos valores aproximados.

Maiores informações sobre o plano de medição do sistema da CEEE estudado neste capítulo podem ser obtidas diretamente de seu arquivo de descrição de redes disponível no CD que acompanha este trabalho. O manual do sistema que descreve a formatação deste também se encontra neste CD. Consulte o Anexo único deste trabalho para localizar os referidos arquivos.

5.5.1.3 – Estimador de estados do CEPEL

Durante a fase de implementação dos estimadores de estados desenvolvidos no âmbito do projeto de P&D, uma das ferramentas que melhor auxiliaram a análise dos resultados foi o estimador de estados desenvolvido pelo CEPEL. Tal estimador é baseado em mínimos quadrados ponderados (MQP) e utiliza testes de hipóteses baseados na SQPR para identificação de medidas de medidas contaminadas com erros grosseiros. As medidas identificadas são recuperadas através da estimação da magnitude do erro grosseiro utilizando-se o método comentado em 4.3.6.4. As restrições de igualdade são processadas através do problema de estimação de estados irrestrito utilizando-se o método dos pesos. Entretanto, segundo Valmir Zampieri, o consultor da CEEE, que acompanhou a implementação deste estimador, o método utiliza ponderação variável para as restrições de igualdade com intuito de evitar mal-condicionamento numérico da matriz Jacobiana.

Alguns softwares desenvolvidos pela própria CEEE possibilitam a inserção deste estimador dentro da seqüência lógica e operacional do COS. Os resultados do estimador possuem formato incompatível com àquele utilizado pelo sistema que realiza a análise de contingências. Portanto os softwares desenvolvidos pela CEEE, conhecidos como *SARON em tempo real* e *SARON modo de estudo*, além de configurar a foto utilizada pelo estimador através do arquivo em tempo real ou do histórico operativo, cria os arquivos de entrada para os estudos de análise de contingências e fluxo de potência. A seqüência de operação da CEEE pode ser vista na Figura 90.

Para fins de análise, o resumo do resultados obtidos pelo estimador de estados do CEPEL para caso de estudo escolhido será mostrado. Cabe comentar, anteriormente, que tal caso foi selecionado por agregar alguns problemas comuns à estimação de estado no sistema de transmissão da CEEE. A tolerância de convergência do estimador da CEPEL é de $1x10^{-3}$. Conforme mostra os resultados da Figura 91, os estimador de estados da CEPEL alcançou a convergência com a SPQR muito superior ao limite do limiar da distribuição Qui-quadrada. Dentre o conjunto de medidas recuperadas, encontra-se a restrição de igualdade (ZERO) que modela a injeção nula de potência reativa da barra de 525kV de Gravataí. Nitidamente, pode ser notada que as medidas adjacente a barra de Gravataí foram as que apresentaram os maiores resíduos normalizados, sendo consecutivamente corrigidas. Tal fato indica duas prováveis situações: A existência de um ou mais erros grosseiros neste conjunto de medidas com provável estimação errônea da magnitude dos erros em virtude da ponderação utilizada; ou a existência de um erro no modelo que próximo às barras de 525kV.

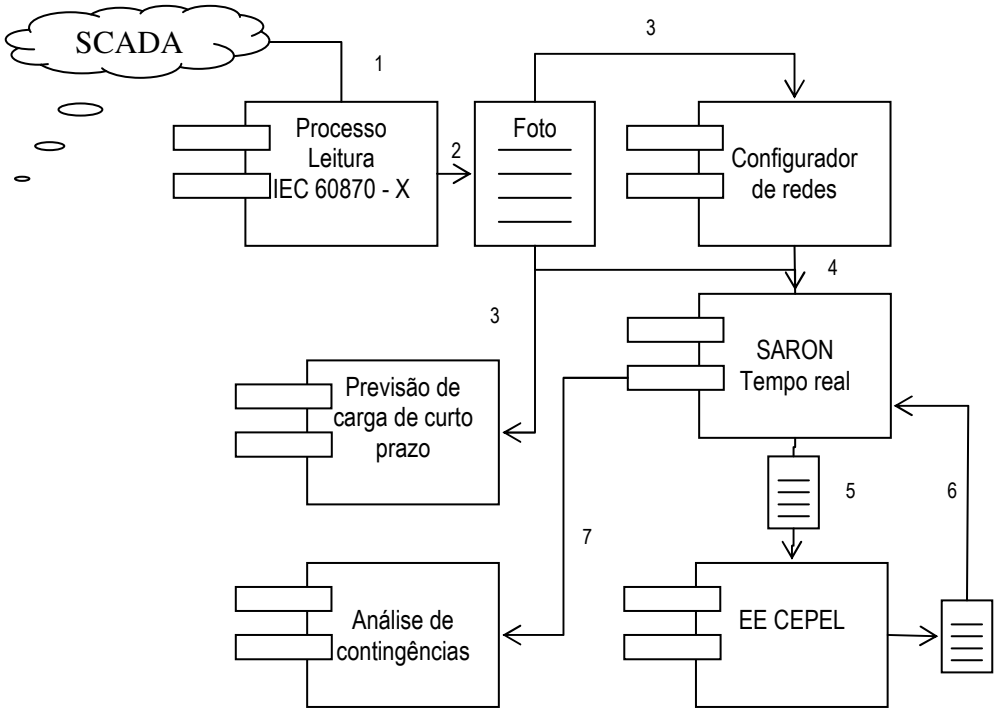


Figura 90 – Seqüência lógica e operacional simplificada do COS-CEEE

```
+-------------------------------------------------------------------------------+
|              SUMARIO DAS REESTIMACOES :  4/09/2003 -  1: 1: 1                 |
+-------------------------------------------------------------------------------+
| Jx Lim, Esp, Calc :      393.66     318.00    3249.03  Nos: 124 124           |
+---+---+---------+---------------------+---------+-------------+-----------+
| P | Q |  Jx Tot | Estac FR   Estac TO | Medida  | Val Medido  | Val Rec.  |
+---+---+---------+---------------------+---------+-------------+-----------+
|13 |13 |13787.05 | GRAVATAI -          | 5 - ZERO |      .00    | -487.00   |
|21 |21 |27863.24 | GRAVATAI -          | 5 - ZERO |      .00    |  201.98   |
|24 |24 |21341.20 | CAX 525- - GRAVATAI | 2 -16126 |   136.30    |  -82.81   |
|17 |17 |12845.98 | GRAVATAI -          | 5 - ZERO |      .00    |  590.47   |
|23 |23 |42157.79 | GRAVATAI - CAX 525- | 2 - 4203 |   -87.00    |  183.94   |
| 4 | 4 |28751.35 | CAX 525- - GRAVATAI | 2 -16126 |   136.30    | -356.11   |
|20 |20 |16259.58 | GRAVATAI - ITA----- | 2 - 4205 |  -154.00    |   33.60   |
| 3 | 3 |10015.70 | GRAVATAI - CAX 525- | 2 - 4203 |   -87.00    |  339.87   |
| 3 | 3 | 6087.22 | CAX 525- - GRAVATAI | 2 -16126 |   136.30    | -459.64   |
|13 |13 | 4029.01 | CAX 525- - ITA----- | 2 -16132 |   -10.00    |   71.39   |
+-------------------------------------------------------------------------------+

Carga CEEE (MW)=  2054.0    Perdas CEEE(MW)=    26.7  Temperatura    =   15.0
Estudo P   (MW)=  1983.9    Estudo Q (MVAr)=  560.8   Fator Potencia =  .962
```

Figura 91 – Resumo dos resultados do estimador do CEPEL para o caso em estudo.

Analisando-se outro caso, durante o período da manhã, encontra-se um resultado muito diferente para o mesmo sistema. Entretanto nota-se que a barra de Gravataí não possui o conjunto de banco de capacitores ligados. Voltando ao caso em questão e substituindo os bancos de capacitores por banco de reatores com mesma potência (invertendo-se o sinal), baseado na existência de um conjunto banco de capacitores na barra de 230kV de Gravataí, pode solucionar parte do problema do caso anterior, conforme pode ser visto na Figura 92.

```
+-------------------------------------------------------------------------+
|              SUMARIO DAS REESTIMACOES :  4/09/2003 -  1: 1: 1           |
+-------------------------------------------------------------------------+
| Jx Lim, Esp, Calc :      393.66     318.00    2524.81  Nos: 124 124      |
+---+---+----------+--------------------+----------+-------------+----------+
| P | Q |  Jx Tot  | Estac FR   Estac TO | Medida   | Val Medido  | Val Rec. |
+---+---+----------+--------------------+----------+-------------+----------+
|14 |14 |11140.18 | GRAVATAI -          | 5 - ZERO |      .00    |    -.38  |
| 1 | 1 | 9536.01 | GRAVATAI -          | 5 - ZERO |      .00    |    -.71  |
| 1 | 1 | 9580.75 | GRAVATAI -          | 5 - ZERO |      .00    |   -1.03  |
| 1 | 1 | 7798.64 | GRAVATAI - CAX 525- | 2 - 4203 |   -87.00    | -225.24  |
| 3 | 3 | 4901.45 | GRAVATAI -          | 5 - ZERO |      .00    |    -.76  |
| 1 | 1 | 4007.07 | GRAVATAI -          | 5 - ZERO |      .00    |    -.54  |
| 1 | 1 | 4003.00 | GRAVATAI -          | 5 - ZERO |      .00    |    -.31  |
| 1 | 1 | 3487.29 | GRAVATAI -          | 5 - ZERO |      .00    |    -.11  |
| 1 | 1 | 3003.71 | GRAVATAI -          | 5 - ZERO |      .00    |     .05  |
| 1 | 1 | 2784.68 | GRAVATAI -          | 5 - ZERO |      .00    |     .21  |
+-------------------------------------------------------------------------+

Carga CEEE (MW)=  2054.0    Perdas CEEE(MW)=    26.7   Temperatura    =   15.0
 Estudo P   (MW)= 1977.7    Estudo Q (MVAr)=  546.7    Fator Potencia =  .964
```

Figura 92 - – Resumo dos resultados do estimador do CEPEL com alterações na modelagem de componentes.

Logo pode-se concluir que um dos problemas que afetam significativamente os resultados da estimação de estados no sistema de transmissão da CEEE são pequenos equívocos que geram grandes impactos e erros do tipo 100%, ou seja, erros de modelagem de sistemas. Entretanto um resultado que se sobrepôs às demais impropriedades foram os modelos adotados para os LTCs entre as barras de 525kV e 230kV de Gravataí. Em todos os estudos realizados, existe uma significativa diferença entre os fluxo mensurados e os fluxo estimados. Sabe-se, por intermédio dos profissionais e consultores da CEEE, que o modelo dos transformadores de 3 enrolamentos de Gravataí não é completo, entretanto tal modelo deve ser corrigido devido a sua peculiar importância para o restante do sistema. As informações destacadas de verde na representam os fluxo ativos estimados e mensurados e as informações destacadas de amarelo representam o fluxo reativo estimados e mensurados.

```
976         536.0  539.8      .0     .0 ( ZERO)
GRAVATAI-525 40.8 ( 4168)    -.1     .0 ( ZERO)
                    SHUNT                                          -104.2  -100.0
                    964*CAX 525--525  -331.6  -334.0 ( 4202)  -225.2   -87.0 ( 4203)
                    995*ITA------525  -315.0  -316.0 ( 4204)  -143.0  -154.0 ( 4205)
                    1210-GRAVATAI-230   214.0   220.0 ( 4166)    75.3    67.0 ( 4167)
                    1210*GRAVATAI-230   218.6   237.0 ( 4119)   107.1   101.0 ( 4120)
                    1210*GRAVATAI-230   214.0   198.0 ( 4121)    81.4    75.0 ( 4122)
```

Figura 93 - Detalhamento dos fluxos ativos e reativos nos LTCs entre as barras de 525kV e 230kK de Gravataí

Por fim, cabe ressaltar que não há informações relativas às posições de TAP de alguns transformadores. Tal fato gerou o hábito se estimar tais posição sem observar se existem redundância local de informação que possibilite tais estimativas. Como resultado, há a formação de conjuntos críticos devido a perda local de informação resultante da inserção de uma nova variável de estado, contribuindo significativamente para degradação da qualidade das estimativas obtidas.

5.5.2 A aplicação dos estimadores de estados desenvolvidos no sistema de transmissão de energia elétrica da CEEE

A adequada aplicação dos estimadores desenvolvidos no âmbito deste trabalho ao sistema de transmissão da CEEE, como comentado anteriormente, limitou-se aos estimadores de estados robustos baseados em MQPV. Tal fato foi motivado pela inexistência das precisões do conjunto de medidores que compõem o sistema de monitoramento da CEEE e acabou invalidado a utilização do estimador de estados baseado em MQP. Apesar das tentativas de se converter as ponderações utilizadas no estimador de estados do CEPEL, concluíu-se que o mesmo possui uma heurística diferenciada no trato de tais pesos. As precisões obtidas durante as tentativas de conversão eram tão inadequadas quanto a adoção das precisões sugeridas pela literatura técnica [55]. Entretanto os estimadores de estados baseados em MQPV mostraram-se menos sensíveis em relação à utilização das reais precisões. Salienta-se que isto influi no resultado obtido por modificar o limite de transição entre o segmento quadrático e não-quadrático do estimador utilizado, mas – sobretudo – não invalida a sua aplicação. As precisões adotadas para os medidores na estimação de estados do sistema CEEE podem ser vistas na , conforme o tipo de medidor. Por sua vez, as restrições de igualdade foram ponderadas através da utilização de uma precisão de $1x10^{-4}$.

Alguns detalhes do modelo barra-ramo do sistema de transmissão da CEEE relacionados à formação de pontos de alavancamento devem ser comentados, apesar de não ser aconselhado o tratamento de tais pesos conjuntamente com a estimação de TAPs de transformadores. Os grupos geradores da CEEE são modelados de forma distribuída, ou seja, não utiliza-se um gerador equivalente para representá-los. O modelamento distribuído é utilizado para simplificar a formação dos modelos das funções subseqüentes. Entretanto seu uso para estimação de estados facilita a formação de pontos de alavancamento devido a quantidade de ramos que incidem sobre o barramento principal do grupo gerador. Aconselha-se a utilização de um modelo equivalente e a posterior determinação das contribuições de cada gerador para a montagem dos modelos das funções subseqüentes. Tal prática evita que erros que podem ser prejudiciais a toda a seqüência operativa não sejam devidamente tratados pelo estimador de estados.

Ressalvando todas as características supracitadas do sistema de transmissão da CEEE, a utilização de estimadores robustos baseados em MQPV pode ser realizada mesmo desconhecendo-se as precisões dos medidores. Por óbvio, o resultado não corresponderá ao esperado. Entretanto se a precisão adotada for inferior a precisão real, haverá uma redução da influência desta medida devido à penalidade gerada relativa aos resíduos padronizados maiores.

Os estimadores de estados foram aplicados ao sistema de transmissão da CEEE conforme a configuração mostrada na Tabela 31 . Destaca-se que o refinamento de resultados foi inibido devido às condições do sistema, pois a eliminação das medidas consideradas errôneas – conforme o método de identificação utilizado durante os testes – resultava num sistema não solucionável, ou

seja, com elementos nulos na diagonal da matriz triangular resultante do processo de fatoração. Isto se deve à presença de conjuntos críticos que possuem medidas com erros grosseiros. Consequentemente, explicam-se os valores elevados da SPQR para todas as estimações mostradas na Tabela 32.

Tabela 31 – Configurações dos estimadores aplicados ao sistema de transmissão da CEEE

Configurações comuns	
Número máximo de iterações	50
Tolerância para convergência	0,001
Matriz Jacobiana constante	Não
Utilizar perfil plano de tensões	Sim
Incluir as restrições de igualdade	Sim
Tolerância aplicada às restrições	0,001
Número de iterações MQP	1
Método de identificação de EG	Peso ou Ponderação
Valor limite da identificação de EG	0,448 (aprox. 3σ) e 0,100 (aprox. 10σ)
Considerar pontos de alavancamento	Não
Refinar resultados	Não

Tabela 32 – Resumo dos resultados obtidos com as estimações

	EE - CEPEL	**MPQV**	**MPQV-DR**
SPQR	2524,81	1360,4	1784
Quantidade de EG	2	77 (3σ) e 13 (10σ)	99 (3σ) e 16 (10σ)
Número de iterações	10	6	9
Tempo	*****	125 ms	78 ms
Graus de liberdade	318	311	311
Qui-Quadrada (97,5%)	393,66	359,88	359,88

Tabela 33 – Comparação entre os estados estimados do sistema da CEEE para os diferente estimadores

Barra		EE - CEPEL		MQPV		MQPV-DR		Barra		EE - CEPEL		MQPV		MQPV-DR	
ID	Nome	V	φ [rad]	V	φ [rad]	V	φ [rad]	ID	Nome	V	φ [rad]	V	φ [rad]	V	φ [rad]
901	CHARQUEA-2CF	0,966	0,750	0,970	0,750	0,970	0,750	1218	LAJEADO--230	1,028	0,698	1,031	0,697	1,032	0,697
917	PFUNDO --G1	0,961	0,785	0,964	0,792	0,965	0,792	1221	LVE□-1383 99	0,996	0,576	0,979	0,580	0,979	0,580
918	PFUNDO---G2	0,961	0,716	0,964	0,719	0,964	0,720	1222	MARAU----138	1,012	0,646	1,007	0,647	1,007	0,648
930	ALEGRETE-138	1,036	0,820	1,041	0,830	1,042	0,830	1223	UDFR-----230	1,020	0,803	1,023	0,796	1,023	0,796
951	CHARQUEA-230	1,031	0,681	1,034	0,689	1,035	0,689	1225	MACAMBARA230	1,057	0,838	1,056	0,839	1,057	0,840
955	CNOVOS---525	1,023	0,820	1,028	0,815	1,028	0,815	1226	NPRATA-2--69	1,026	0,646	1,027	0,652	1,027	0,653
962	FARROUP--230	1,029	0,698	1,033	0,708	1,034	0,708	1228	NPRATA-2-230	1,037	0,698	1,040	0,695	1,040	0,695
964	CAX 525--525	1,038	0,733	1,044	0,740	1,043	0,740	1230	LIVRAM---230	1,049	0,750	1,052	0,759	1,052	0,759
965	CAX 230--230	1,030	0,716	1,033	0,722	1,034	0,722	1231	OSORIO-2-230	1,042	0,646	1,045	0,648	1,046	0,647
976	GRAVATAI-525	1,020	0,716	1,028	0,713	1,028	0,713	1233	PELOTAS1--69	1,026	0,489	1,028	0,492	1,027	0,491
995	ITA------525	0,993	0,838	1,000	0,834	0,999	0,833	1236	PELOTAS3-230	1,020	0,541	1,026	0,550	1,026	0,548
1041	PFUNDO---230	1,025	0,716	1,028	0,724	1,028	0,724	1238	PMEDICI--138	1,021	0,576	1,021	0,582	1,021	0,582
1042	PFUNDO---138	1,002	0,663	1,003	0,671	1,003	0,672	1239	PMEDICI--230	1,033	0,611	1,038	0,610	1,038	0,609
1046	STA-ESUL-230	1,028	0,803	1,031	0,801	1,031	0,801	1242	PREAL----138	1,016	0,820	1,013	0,829	1,012	0,829
1057	SIDEROPO-230	1,013	0,750	1,018	0,756	1,014	0,757	1243	PREAL----230	1,022	0,785	1,025	0,784	1,025	0,784
1068	URUGUAIA- 13	1,031	0,942	1,031	0,951	1,031	0,952	1245	QUINTA----69	1,045	0,489	1,046	0,488	1,046	0,487
1069	XANXERE--230	1,026	0,716	1,029	0,727	1,029	0,728	1246	QUINTA---230	1,010	0,541	1,016	0,547	1,016	0,545
1155	ITAUBA----G1	0,964	0,785	0,966	0,786	0,966	0,786	1247	QUINTA---138	0,947	0,541	0,948	0,544	0,948	0,542
1156	ITAUBA----G2	0,937	0,785	0,940	0,786	0,941	0,786	1248	CANOAS2--230	1,030	0,681	1,034	0,690	1,034	0,690
1157	ITAUBA----G3	0,944	0,890	0,947	0,883	0,947	0,883	1249	TAQUARA--230	1,044	0,646	1,047	0,644	1,048	0,644
1158	ITAUBA----G4	0,938	0,873	0,942	0,870	0,943	0,870	1250	SANGELO--230	1,028	0,803	1,031	0,800	1,031	0,800
1162	JACUI-----G1	0,941	1,012	0,943	1,007	0,943	1,008	1251	SANGELO---69	1,017	0,785	1,017	0,778	1,017	0,778
1163	JACUI-----G2	0,924	1,012	0,922	1,017	0,919	1,017	1256	SCRUZ-1--230	1,033	0,716	1,036	0,716	1,037	0,716
1164	JACUI-----G3	0,927	1,012	0,924	1,018	0,922	1,019	1257	CINDUS---138	1,010	0,646	1,012	0,652	1,012	0,652
1165	JACUI-----G4	0,940	1,012	0,942	1,006	0,942	1,008	1258	CINDUS---230	1,029	0,681	1,033	0,680	1,033	0,680
1167	JACUI-----G6	0,925	1,012	0,923	1,013	0,920	1,013	1259	PAL-4----230	1,029	0,681	1,033	0,674	1,034	0,674
1174	PREAL-----G2	0,939	0,838	0,941	0,847	0,941	0,847	1260	PALEGRE-6-TA	1,006	0,611	1,012	0,621	1,012	0,621
1175	PREAL-----G1	0,936	0,855	0,940	0,849	0,941	0,849	1261	PALEGRE-6-TB	1,006	0,611	1,012	0,622	1,012	0,622
1176	UDFR-----G2	0,983	0,785	0,986	0,795	0,987	0,795	1262	PAL-6-----69	1,006	0,628	1,012	0,624	1,013	0,624
1177	UDFR-----G1	0,974	0,873	0,977	0,865	0,977	0,865	1263	PALEGRE6-230	1,030	0,681	1,034	0,678	1,034	0,678
1179	PALEGRE-6-13	0,998	0,611	1,004	0,622	1,005	0,621	1265	PALEGRE-9--T	1,020	0,628	1,023	0,635	1,024	0,635
1180	PALEGRE-9-13	1,018	0,628	1,000	0,000	0,938	0,664	1266	PAL-9-----69	1,020	0,646	1,024	0,636	1,024	0,636
1181	VAIRES----13	0,937	0,663	0,938	0,664	1,038	0,841	1267	PAL-9----230	1,028	0,663	1,031	0,671	1,032	0,671
1182	ALEGRETE--69	1,031	0,838	1,038	0,840	1,036	0,848	1268	PAL-10---230	1,029	0,681	1,033	0,674	1,034	0,674
1185	ALEGRETE2-69	1,028	0,838	1,036	0,848	1,042	0,872	1270	PAL-13---230	1,029	0,681	1,033	0,676	1,033	0,676
1186	ALEGRET2-230	1,042	0,873	1,042	0,871	1,000	0,000	1273	SIDERURG-230	1,023	0,663	1,026	0,673	1,027	0,673
1188	BAGE-2---230	1,038	0,646	1,043	0,641	1,043	0,640	1274	SVITORIA-138	0,993	0,489	0,986	0,496	0,988	0,488
1189	BAGE-2----69	1,041	0,593	1,041	0,593	1,042	0,593	1275	SMARTA-A-138	1,010	0,663	1,011	0,667	1,011	0,667
1190	CANOAS1--230	1,028	0,663	1,032	0,673	1,032	0,673	1276	SMARIA-1-138	1,013	0,803	1,008	0,811	1,011	0,807
1192	CAXIAS-5-230	1,028	0,716	1,032	0,709	1,032	0,709	1278	SMARIA3--230	1,033	0,785	1,035	0,793	1,035	0,793
1194	CAMAQUA--230	1,041	0,611	1,045	0,618	1,046	0,618	1279	SMARTA----69	0,999	0,646	1,000	0,651	1,000	0,652
1195	CAMAQUA---69	1,042	0,559	1,042	0,565	1,042	0,565	1281	SMARTA---230	1,023	0,698	1,020	0,705	1,020	0,705
1196	CBOM-2---230	1,034	0,698	1,038	0,693	1,038	0,693	1284	SMARTA----46	0,938	0,646	0,939	0,650	0,939	0,650
1197	GUA2------69	1,043	0,593	1,044	0,592	1,044	0,592	1286	SVICENTE-230	1,042	0,820	1,043	0,820	1,043	0,820
1198	CBOM-1---230	1,034	0,698	1,037	0,693	1,038	0,693	1294	UTUR-----230	1,042	0,960	1,041	0,954	1,041	0,955
1199	GUA2----_230	1,030	0,646	1,034	0,647	1,034	0,647	1295	URUGUAI-5-69	1,053	0,942	1,053	0,943	1,053	0,944
1200	CAXIAS-2-230	1,027	0,698	1,031	0,706	1,031	0,706	1296	URUGUAI5-230	1,042	0,942	1,041	0,951	1,041	0,952
1201	CAXIAS-2--69	0,966	0,593	0,967	0,595	0,967	0,595	1297	VAIRES----T1	0,945	0,663	0,945	0,664	0,945	0,664
1202	CHARQUEAF-69	1,037	0,646	1,037	0,645	1,037	0,645	1298	VAIRES----69	0,992	0,663	0,993	0,666	0,993	0,666
1203	CHARQUEA--69	1,037	0,681	1,038	0,689	1,038	0,689	1299	VAIRES---230	1,021	0,698	1,024	0,703	1,025	0,703
1204	POLOPETRO230	1,016	0,663	1,019	0,669	1,020	0,669	2057	BASILIO--138	1,037	0,559	1,038	0,553	1,038	0,553
1206	GARIBALDI230	1,023	0,698	1,028	0,702	1,029	0,702	2061	CALTA----138	0,998	0,803	0,995	0,798	0,994	0,798
1207	FARRO-----69	1,018	0,681	1,019	0,673	1,019	0,673	2068	CCORINHA-138	1,005	0,646	1,006	0,642	1,006	0,642
1209	GRAVATAI--69	1,031	0,593	1,032	0,603	1,032	0,603	2069	EREXIM---138	0,987	0,628	0,988	0,629	0,988	0,629
1210	GRAVATAI-230	1,036	0,681	1,039	0,688	1,039	0,688	2077	POLOPETRO-69	1,043	0,593	1,044	0,594	1,044	0,594
1212	ELDORADO-230	1,030	0,646	1,033	0,655	1,033	0,655	2082	GM ------230	1,034	0,681	1,038	0,686	1,038	0,686
1213	GUARITA--230	1,035	0,716	1,038	0,725	1,039	0,726	2086	SCHARLAU-230	1,034	0,698	1,037	0,693	1,037	0,692
1215	ITAUBA---230	1,019	0,785	1,022	0,788	1,022	0,788	2087	SMARIA-3--69	1,031	0,768	1,031	0,766	1,031	0,766
1216	JACUI----138	1,016	0,838	1,014	0,840	1,013	0,840	2095	UHPFUNDO--23	0,986	0,646	0,989	0,645	0,989	0,645

5.5.3 Comentários finais sobre a estimação de estados no sistema de transmissão de energia elétrica da CEEE

Os resultados obtidos foram aquém daqueles esperados no ínicio do projeto, mas isto não se deve aos modelos desenvolvidos ou os métodos utilizados. Ao contrário, apesar das limitações impostas ao estimador MQPV acoplado, o conjunto de modelos e ferramentas que cominaram nos estimadores de estados implementados obtiveram os resultados esperados e mostraram ser aplicáveis a sistemas realísticos considerando-se todas as suas peculiaridades. Também não se pode mencionar sobre qualquer problema relativo ao estimador desenvolvido pelo CEPEL, uma vez que nada pode ser extraído de seus resultados sob as condições impostas pelo conjunto de impropriedades comentadas ao longo desta subseção. Os registros que devem ser feitos recaem sobre o próprio sistema da CEEE, os quais são:

- O uso de ponderações obtidas de forma empírica prejudicam e desviam a finalidade da estimação de estados. Esta função baseia-se na minimização ponderada dos resíduos, onde a confiança em determinada medida é obtida através de sua respectiva precisão. Portanto há um direcionamento do resultado quando se determina pesos para solucionar certo problema vivenciado;
- A redundância de informações sobre o sistema é fundamental para que se possa identificar e filtrar medidas errôneas. Portanto as expansões do sistema e a estimação paramétrica devem ser feitas acompanhadas de um estudo de observabilidade, bem como deve-se analisar o plano de medição para identificar os pontos críticos da supervisão operativa;
- Erros no modelo de componentes, na situação verídica de medidores e na configuração da rede operativa invariavelmente ocasionará ou a divergência do estudo ou convergência para um ponto operativo não condizente com a realidade;
- A modelo de sistema deve ser o mais adequado possível à função ou ao estudo em questão, independentemente do formato dos resultados obtidos. Caso seja necessário ou possível, deve-se optar pelo melhor modelo e convertê-lo após a finalização do processo.

Capítulo 6

Conclusões

Nos capítulos anteriores foram apresentados um conjunto de modelos orientados a objeto com o intuito de dar suporte à implementação de estimadores de estados robustos na Companhia Estadual de Energia elétrica do Rio Grande do Sul. O modelo de sistema elétrico proposto neste trabalho não tem o objetivo de possibilitar que toda ou qualquer função relacionada a sistemas de energia elétrica seja desenvolvida. A modelagem do sistema limitou-se – por restrições técnicas e orçamentárias – ao desenvolvimento de um conjunto de classes pudesse representar o modelo barra-ramo de regime permanente com as peculiaridades necessárias à contemplação de um sistema de monitoramento e manobra segundo as características encontradas nos centros de operação de sistema. Conforme mostrado no capítulo 4, todas as classes representam componentes ou grupos do sistema descendem de um núcleo abstrato e comum, sendo posteriormente unidas às facilidades de outras classes desenvolvidas especificamente para a manobra e o monitoramento de grandezas. Novos modelos de componentes podem ser criados a partir destas classes ou especializados diretamente das classes que compõem o núcleo especializado, possibilitando uma manutenção mais rápida e eficiente do sistema e uma evolução facilitada. A modelamento conceitual utilizado na MOO permite uma fácil compreensão do sistema por pessoas que não estavam envolvidas em seu desenvolvimento. Portanto a MOO possibilita a reciclagem de recursos humanos em empresas e centros de pesquisas, solucionando problemas em equipes com alta rotatividade de pessoas – como os centros de pesquisa universitários.

O pacote de ferramentas matemáticas proposto neste trabalho possibilita o desenvolvimento rápido e direto de sistemas que trabalham com matrizes e vetores esparsos e necessitam de métodos eficientes de solução de sistemas lineares. Conforme mostrado no capítulo 4, o pacote possui classes específicas para ordenação matricial e solução de sistemas lineares, as quais utilizam um conjunto dos melhores métodos disponíveis na literatura técnica. A eficiência das ferramentas matemáticas foi comprovada através de exaustivos de testes, os quais indicaram os métodos que proporcionam os melhores resultados. Os resultados dos testes realizados neste trabalho se contrapõem aos resultados obtidos por Soman et alli [29] e Pandit et alli [105], corroborando parcialmente os resultados obtidos por Vempati et alli [68]. Dentre os esquemas mais eficiente de fatoração matricial utilizando-se métodos ortogonais, encontram-se as rotações de Givens com 2 ou

3 multiplicadores antecedidas por combinações dos métodos de ordenação de linhas MDA[69] e A1[73] e os métodos de ordenação de colunas R3, R4 e R5 [68]. O pacote de ferramentas matemáticas ainda possui métodos para a inversão matricial completa ou esparsa e para o cálculo de pesos de pontos de alavancamento de sistemas lineares. As classes matriciais implementadas possuem um conjunto de operadores que possibilitam que as operações básicas sejam realizadas de forma direta e um conjunto de funções variadas que possibilitam, dentre outras funcionalidades, o cálculo de normas euclidianas e infinitas.

Os estimadores de estados foram implementados utilizando o conceito de elemento de estimação, o qual concentra todos os recursos e requisitos associados às medidas ou restrições de igualdade que compõem o problema de estimação de estados. Tal conceituação permite abordar as estimação de estados de forma diferenciada, atribuindo ao estimador de estados somente os procedimentos globais e de coordenação que envolvam o conjunto de medidas, resíduos ou solução do sistema linear. Todos as operações que possam ser realizadas com as informações da medida ou restrição e do componente de rede que a originou passa a ser de responsabilidade do elemento de estimação. Foi proposto neste trabalho duas classes abstratas para estimação de estados, as quais implementam um conjunto comum de propriedades, procedimentos e funções – possibilitando que as classes descendentes voltem-se para o desenvolvimento do algoritmo do estimador que se esteja implementando. A partir destas classes abstratas foram criadas as classes que compõem o estimador baseado em mínimos quadrados ponderados (MQP) com identificação de erros grosseiros através de testes de hipóteses e as classes que compõem o estimador de estados baseado mínimos quadrados com ponderação variável (MQPV). A implementação de ambos estimadores requereu um conjunto mínimo de propriedades e funções adicionais, sendo voltado para os algoritmos de estimação. Portanto propõem-se a utilização deste modelo no desenvolvimento de novos estimadores.

Os testes com os estimadores implementados mostraram que:

- É necessário se aplicar métodos de refinamento de resultados quando se estima os estados do sistema através dos métodos baseados em MQPV devido a possibilidade de erros grosseiros influenciarem outras medidas, principalmente quando está se estimando parâmetros do sistema;
- Erros grosseiros próximos a LTCs cujas posições de TAPs estão sendo estimadas possibilita que o estimador MQPV penalize medidas adjacentes às barras terminais do LTC, podendo acarretar na sua equivocada identificação como portadora de erro grosseiro e na conseqüente convergência para um ponto operativo que não corresponde com o real estado do sistema;
- O tratamento de erros grosseiros em medidas formadoras de ponto de alavancamento deve ser realizado com ressalvas para os estimadores baseados em MQPV quando as

posições de TAPs estão sendo estimadas, uma vez que a convergência oscilatória destas variáveis de estado pode resultar na identificação equivocada de erros grosseiros em medidas formadoras de ponto de alavancamento.

O uso dos estimadores de estados no sistema de transmissão da CEEE foi limitado e obteve resultados aquém do esperado devido a um conjunto de fatores próprios do modelo e das práticas utilizadas pela referida companhia em relação a questões relacionadas diretamente com a estimação de estados. Dentre os impropriedades e inconformidades destacadas neste trabalho, encontram-se:

- O uso de ponderações obtidas de forma empírica ao invés de precisões obtidas através de programas de manutenção do parque de medidores, gerando significativo prejuízo nos resultados obtidos e, até mesmo, a limitação no uso de estimadores mais dependente deste parâmetros – como o estimador baseado em MQP;
- O desconhecimento ou a não realização de análises de observabilidade conjugada com a expansão da rede elétrica e com o hábito de ser estimar os TAPs de transformadores aumentam as incertezas associadas à estimação de estados, criam medidas e conjuntos críticos e reduzem a capacidade de filtragem de erros no conjunto de medidas que formam o plano de medição. Existe uma quantidade significativa de medidas e conjuntos críticos no plano de medição da CEEE, indicando problemas de identificação de erros de medição e provável obtenção de estados que não correspondem aos verdadeiros estados do sistema;
- Erros no modelo de componentes, na situação verídica de medidores e na configuração da rede operativa ocasionando ou a divergência do estudo ou convergência para um ponto operativo não condizente com a realidade;

Considerando todos os fatores envolvidos no desenvolvimento e na aplicação de estimadores de estados ao sistema de transmissão da CEEE, pode-se concluir que o trabalho atingiu seus objetivos iniciais, apesar dos resultados divergirem daqueles desejados. O conjunto de classes que formam os modelos propostos por este trabalho mostraram ser compatíveis com os requisitos necessários à estimação de estados em sistemas reais considerando todas as suas particularidades.

Anexo

Diferentemente das modelos tradicionais de dissertação, optou-se por apresentar dos dados e o manual do sistema desenvolvido em formato digital, através de um CD de dados. Abaixo relaciona-se o conjunto de informações que constituem o CD anexado.

Manual do sistema

O manual do sistema encontra-se no diretório "\\Manual" e é formado por um único arquivo "\\Manual\VDTAP.pdf".

Programa VDTAP

A versão acadêmica do programa VDTAP, a qual possibilita exclusivamente a abertura de arquivos com formato VDELTA-LABSPOT, encontra-se no diretório "\\DVTAP", sendo o arquivo do programa "\\VDTAP\VDTAP.exe".

Diretório de dados

Os arquivos de dados utilizados neste trabalho encontram-se no diretório "\\Dados", entretanto este diretório possui outros sistema e matrizes de testes. Os sistemas e a matriz de teste utilizados no trabalho são:

1. Sistema IEEE 14 barras: "\\Dados\IEEE14.dad";
2. Sistema IEEE 118 barras: "\\Dados\IEEE118.dad";
3. Sistema Sul-Sudeste de 340 barras: "\\Dados\Sist340.dad";
4. Sistema Sul-Sudeste de 730 barras: "\\Dados\Sist730.dad";
5. Sistema Sul-Sudeste de 1916 barras: "\\Dados\Sist2000.dad";
6. Sistema de transmissão da CEEE (Ago-Out 2003): "\\Dados\CEEE.ee";
7. Matriz de teste das ferramentas matemáticas: "\\Dados\H2000.mat".

Referências Bibliográficas

[1] P. COAD, E. YOURDON, *Object-oriented analysis*, Ed. Prentice Hall, 1990.

[2] J. RUMBAUGH, M. BLAHA ET AL., *Object-oriented modeling and desing*, Ed. Prentice Hall, 1991.

[3] P.J. ROBINSON, *Hierarchical object-oriented design*, Ed. Prentice Hall, 1992.

[4] I. JACOBSON, M. CHRISTERSON ET AL, *Object-oriented software engineering,* Ed. Addison Wesley, 1993.

[5] G. BOOCH, *Object-oriented analysis and design with applications*, Ed. Addison Wesley, 1994.

[6] I. GRAHAM, *Object-oriented methods,* Ed. Addison Wesley, 1994.

[7] J. RUMBAUGH, I. JACOBSON ET AL., *The unified modeling language reference manual*, Ed. Addison Wesley, 1999.

[8] G. BOOCH, J. RUMBAUGH, I. JACOBSON, *The unified modeling language - user guide,* Ed. Addison Wesley, 1999.

[9] J. RUMBAUGH, I. JACOBSON ET AL., *The Unified Software Development Process*, Ed. Addison Wesley, 1999.

[10] I. SOMMERVILLE, *Engenharia de software*, Ed. Addison Wesley, 2003.

[11] J. D. FURLAN, *Modelagem de Objetos através da UML – Análise e Desenho Orientados a Objeto,* Ed. Makron Books, 1998.

[12] L. KLANDER, K. JAMSA, *Programando em C/C++: A Bíblia*, Ed. Makron Books, 1999.

[13] DOCUMENTAÇÃO OFICIAL DA UML, "http://www-306.ibm.com/software/rational/uml/", obtida em fevereiro de 2006.

[14] D. BECKER, H. FALK, J. GILLERMAN, S. MAUSER, R. PODMORE, L. SCHNEBERGER, "Standards-Based Approach Integrates Utility Applications", IEEE Computer Applications in Power, pp 13-20, October 2000.

[15] S. PANDIT, S.A. SOMAN, S.A. KHAPARDE, "Object-Oriented Design for Power System Applications", IEEE Computer Applications in Power, pp 43-47, October 2000.

[16] M.N. AGOSTINI, I.C. DECKER, A.S. SILVA, "Nova Filosofia para o Projeto de Software na Área de Sistemas de Energia Elétrica utilizando Modelagem Orientada a Objetos", Anais do XIV Congresso Brasileiro de Automática, Natal – RN – Brasil, pp 1555-1561, Setembro 2002.

[17] M.N. AGOSTINI, *Nova Filosofia para o Projeto de Software para Sistemas de Energia Elétrica usando Modelagem Orientada a Objetos*, Tese de Doutorado, UFSC, 2002.

[18] A. MANZONI, *Desenvolvimento de um Sistema Computacional Orientado a Objetos para Sistemas Elétricos de Potência: Aplicação a Simulação Rápida e Análise da Estabilidade de Tensão*, Tese de Doutorado, UFRJ, 2005.

[19] A. F. NEYER, F. F. WU, K. IMHOF, "Object-Oriented Programming for Flexible Software: Example of a Load Flow", IEEE Transactions on Power Systems, Volume 5, No. 3, pp 689-696, August 1990.

[20] M. FOLEY, A. BOSE, W. MITCHEL, A. FAUSTINI, "An Object Based Graphical User Interface for Power Systems", IEEE Transactions on Power Systems, Volume 8, No. 1, pp 97-104, November 1993.

[21] M. FOLEY, A. BOSE, "Object-Oriented on-line Network Analysis", IEEE Transactions on Power Systems, Volume 10, No. 1, February 1995.

[22] B. HAKAVIK, A. T. HOLEN, "Power System Modelling and Sparse Matrix Operations Using Object-Oriented Programming", IEEE Transactions on Power Systems, Volume 9, No. 2, pp 1045-1051, May 1994.

[23] E. Z. ZHOU, "Object-Oriented Programming, C++ and Power System Simulation", IEEE/PES Winter Meeting, paper 95 SW 219-5 PWRS, January/February 1995.

[24] A. MANZONI, A. S. SILVA, I. C. DECKER, "Power Systems Dynamics Simulation Using Object-Oriented Programming", IEEE Transactions on Power Systems, Volume 14, No. 1, pp 249-255, February 1999.

[25] J. ZHU, P. JOSSMAN, "Application of Design Patterns for Object-Oriented Modeling of Power Systems", IEEE Transactions on Power Systems, Volume 14, No. 2, pp 532-537, 1999.

[26] M.N. AGOSTINI, I.C. DECKER, A.S. SILVA, "Desenvolvimento e Implementação de uma Base Computacional Orientada a Objetos para Aplicações em Sistemas de Energia Elétrica", Anais do XIII Congresso Brasileiro de Automática - CBA, Florianópolis – SC – Brasil, pp 1850-1856, Setembro 2000.

[27] S. PANDIT, S.A. SOMAN, S.A. KHAPARDE, "Object-Oriented Network Topology Processor", IEEE Computer Applications in Power, pp. 42-46, April 2001.

[28] S. PANDIT, S.A. SOMAN, S.A. KHAPARDE, "Design of Generic Direct Sparse Linear System Solver in C++ for Power System Analysis", IEEE Transactions on Power Systems, Volume 16, No. 4, pp 647-652, November 2001.

[29] S.A. SOMAN, S.A. KHAPARDE, S. PANDIT, *Computational Methods for Large Sparse Power Systems Analysis: An Object Oriented Approach*, Ed. Kluwer Academic Publisher, 2001.

[30] S. LI, S. M. SHAHIDEHPOUR, "An Object Oriented Power System Graphics Package for Personal Computer Enviroment", IEEE Transactions on Power Systems, Volume 8, No. 3, pp 1054-1060, August 1993.

[31] D. RUMPEL, S.S. VENKATA, K. PANDJI, C.C. LIU, N.M. NAGDY, "Real-Time Database for Power Systems using Language Oriented Data Structure", IEEE Transactions on Power Systems, Volume 5, No. 3, pp 993-1000, August 1990.

[32] D.G. FLINN, R.C. DUGAN, “A Database for Diverse Power System Simulation Applications”, IEEE Transactions on Power Systems, Volume 7, No. 2, pp 784-790, May 1992.

[33] S. N. IROMONGER, M.J. BUSHNELL, R. PATEL, M.E. BRADLEY, B.W. VAUGHAN, “An Object-Oriented Power System Model and Graphical Information Display System for Control Engineers”, Power System Control and Management, Conference Publication No. 421, IEE, pp 120-124, April 1996.

[34] N.B.P. PHILLPS, J.O. GANN, M.R. IRVING, “The Simian Architecture - An Object Orientated Framework for Integrated Power System Modelling Analysis and Control”, Power System Control and Management, Conference Publication No. 421, IEE, pp 148-153, April 1996.

[35] E. HANDSCHIN, M. HEINE, D. KÖNIG, T. NIKODEM, T. SEIBT, R. PALMA, “Object-Oriented Software Engineering for Transmission Planning in Open Access Schemes”, IEEE Transactions on Power Systems, Volume 13, No. 1, pp 94-100, February 1998.

[36] J. ZHU, D. LUBKEMAN, “Object-Oriented Development of Software Systems for Power System Simulations”, IEEE Transactions on Power Systems, Volume 12, No. 2, pp 1002-1007, May 1997.

[37] L.R. ARAUJO, P.A.N. GARCIA, J.L.R. PEREIRA, “Modelagem Orientada a Objetos Aplicada na Solução de Programas de Distribuição”, Anais do XIV Congresso Brasileiro de Automática - CBA, Natal – RN – Brasil, pp 844-848, Setembro de 2002.

[38] M.E. BRADLEY, M.J. BUSHNELL, S.I. MACLEAN, “Object-Oriented Creation of Fault Sequences for On-Line Transient Stability Analysis”, 13th PSCC – Power Systems Computation Conference, Trondheim, pp 654-660, June/July 1999.

[39] Z. L. GAING, C. N. LU, B. S. CHANG ET AL., “An Object-Oriented Approach for Implementing Power System Restoration Package”, IEEE Transactions on Power Systems, Volume 11, No. 1, pp 483-489, February 1996.

[40] H. MIAO, M. SFORNA, C. C. LIU, “A New Logic-Based Alarm Analyzer for Online Operational Environment”, IEEE Transactions on Power Systems, Volume 11, No. 3, pp 1600-1606, August 1996.

[41] M.C. PAULK, B. CURTIS, M.B. CHRISSIS, C.V. WEBER, “Capability Maturity Model, version 1.1”, IEEE Software, Volume 10, Issue 4, pp 18-27, July 1993.

[42] V. HAASE, R. MESSNARZ; G. KOCH, H.J. KUGLER, P. DECRINIS, “Bootstrap: fine-tuning process assessment”, IEEE Software, Volume 11, Issue 4, pp 25-35, July 1994.
.
[43] P. KOCH, S. KUVAJA, L. MILA, A. KRZANIK, S. BICEGO, G. SAUKKONEN, *Software Process Assessment and Improvement: The BOOTSTRAP Approach*, Ed. Blackwell Publishers, 1994.

[44] W.W. ROYCE, “Managing the development of large software systems: concepts and techniques”, Proceedings of IEEE WESTCON, 1970.

[45] R.S. OSHANA, R. C. LINGER, “Capability Maturity Model software development using Cleanroom software engineering principles-results of an industry project”, Volume 7, pp 10, HICSS-32, Proceedings of the 32nd Annual Hawaii International Conference on System Sciences, 1999.

[46] H.D. MILLS ET AL, “The management of software engineering”, Volume 24, issue 5, pp 414-477, IBM Systems Journal, 1980.

[47] M. W. ALFORD, "A Requirements Engineering Methodology for Real-Time Processing Requirements", IEEE Trans. on Software Engineering, vol. SE-3, pp 60-69, January 1977.

[48] K. HENINGER, "Specifying Software Requirements for Complex Systems: New Techniques and Their Application", IEEE Trans. on Software Engineering, vol. SE-6, pp 02-13, January 1980.

[49] A. M. DAVIS, *Software Requirements: Objects, Functions and States*, Ed. Prentice Hall, 1993.

[50] R.H. THAYER, M. DORFMAN, *IEEE Recommended Practice for Software Requirements Specifications*, IEEE Computer Society, 1993.

[51] L.L. CONSTANTINE, E. YOURDON, *Structured Design*, Ed. Prentice Hall, 1979.

[52] C. GANE, T SARSON, *Structured Systems Analysis*, Ed. Prentice Hall, 1979.

[53] M.A JACKSON, *System Development*, Ed. Prentice Hall, 1983.

[54] J. BANSLER, K. BØDKER, "A Reappraisal of Structured Analysis: Design in an Organizational Context", ACM Transactions on Information Systems, Volume 11, Issue 2, pp 165-193, 1993.

[55] R. C. PIRES, *Estimadores Definidos com Critérios Estatísticos Numéricos Aplicados à Estimação Robusta de Estados em Sistemas de Potência*, Tese de Doutorado, UFSC, 1998.

[56] R. MACKIEWICZ, SISCO INC., "Technical Overview and Benefits of the IEC 8150 Standard for Substation Automation". Disponível em <http://www.sisconet.com/techinfo.htm>. Acesso em 10 de maio de 2007.

[57] E. BERTAZINI, *Análise de funções de um conversor de protocolos de comunicação para a automação elétrica baseado na utilização da linguagem de modelagem unificada*, Dissertação de Mestrado, USP, 2006.

[58] IEC 60870-5-3, *Telecontrol Equipment Systems – Part 5: Transmission Protocols – Section 3: General Structure of Application Data*, IEC, 1992.

[59] I. S. DUFF, A. M. ERISMAN E J. K. REID, *Direct Methods for Sparse Matrices,* Clarendon Press, Oxford, 1986.

[60] G. H. GOLUB E C. F. VAN LOAN, *Matrix Computation (second edition),* The Johns Hopkins University Press, 1989.

[61] Å. BJÖRCK, *Matrix Numerical Methods for Least Squares Problems,* Society for Industrial and Applied Mathematics – SIAM, 1996.

[62] A. GJELSVIK, S. AAM E L. HOLTEN, "Hachtel's Augmented Matrix Method – a Rapid Method Improving Numerical Stability in Power System Static State Estimation", IEEE - Transactions on Apparatus and Systems, Vol. PAS 104, pp 2987-2993, November 1985.

[63] A. SIMÕES-COSTA E V. H. QUINTANA, "A Robust Numerical Technique for Power System State Estimation", IEEE - Transactions on Apparatus and Systems, Vol. PAS 100, pp 691-698, February 1981.

[64] W. M. GENTLEMAN, "Least Squares Computations by Givens Transformations without Square Roots", J. Inst. Math. Applic., 12:329-336, 1973.

[65] S. HAMMARLING, "A Note on the Modifications to the Givens Plane Rotation", J. Inst. Math. Applic., 13:215-218, 1974.

[66] A. GEOGE E M. T. HEATH, "Solution of Sparse Linear Least Squares Problems Using Givens Rotations", Linear Algebra and its Applications, Number 34, pp. 69-83, 1980.

[67] A. SIMÕES-COSTA E V. H. QUINTANA, "An Orthogonal Row Processing Algorithm for Power System Sequential State Estimation", IEEE - Transactions on Apparatus and Systems, Vol. PAS 100, pp. 3791-3800, August 1981.

[68] N. VEMPATI, I. W. SLUTSKER E W. F. TINNEY, "Enhancements to Givens Rotations for Power System State Estimation", IEEE - Transactions on Power Systems, Vol. 6, Number 2, pp. 842-849, May 1991.

[69] W. Givens, "Computation of the Plane Unitary Rotations Transforming a General Matrix to Triangular Form", J. Soc. Ind. Appl. Math., Vol. 6, pp. 26-50.

[70] G. H. GOLUB E C. F. VAN LOAN, "The Orthogonal Factorization of a Large Sparse Matrix", *Matrix Computation,* The Johns Hopkins University Press, pp. 156 - 162, 1983.

[71] M.N. AGOSTINI, J.M.F FERREIRA, I.C. DECKER E A.S. SILVA, "Object Oriented Matrix Structure for the Development of Computing Tools in Eletric Power Systems", VIII Simpósio de Especialista em Planejamento da Operação e Expansão Elétrica – SEPOPE, IP-108, Brasília, Maio de 2002.

[72] W.F. TINNEY E J.W. WALKER, "Direct Solutions of Sparse Network Equations by Optimally Ordered Triangular Factorization", Proceedings of the IEEE, Vol 55, pp. 1801-1809, 1967.

[73] A. GOMES E L.G. FRANQUELO, "Node Ordering Algorithms for Sparse Vector Method Improvement", IEEE Trans. on Power Systems, Vol. 3, n. 1, pp. 73-79, February 1988.

[74] I.S. DUFF, "Pivot Selection and Row Ordering in Givens Reduction on Sparse Matrices", Computing No. 13, pp. 239-248, 1974.

[75] A. GEORGE AND M.T. HEATH, "Solution of Sparse Linear Least Squares Problems using Givens Rotations", Linear Algebra and its Applications, No. 34, pp 69-83, 1980.

[76] D. J. ROSE, "A graph-theoretic study of the numerical solution of sparse positive definite systems of linear equations", in Graph Theory and Computing, R. C. Read, pp. 183–217, New York: Academic Press, 1973.

[77] T.H. ROBEY E D.L. SULSKY, "Row ordering for sparse QR decomposition", SIAM Journal of Matrix Analysis and Applications, 15(4), pp 1208-1255, 1994.

[78] F. ABOYTES E B.J. CORY, "Identification of measurement and configuration errors in static estimation", Proc. 9th Power Industry Computer Application Conference, pp. 298-302, New Orleans, June 1975.

[79] H. DURAN, "Surrogate measurements make faster estimation optimal and general", Paper A 77 598-6, IEEE PES Summer Meeting, Mexico City, 1977.

[80] A. GARCIA, A. MONTICELLI E P. ABREU, "Fast decoupled state estimation and bad data processing", IEEE Trans. Power App. and Syst., Vol. 98, No. 5, pp. 1645-1652, September 1979.

[81] N. XIANG, S. WANG E. YU, "Estimation and identification of multiple bad data in power system state estimation", Proc. 7th Power System Computation Conference, PSCC, Lausanne, July 1981.

[82] N. XIANG, S. WANG E E. YU, "A new approach for detection and identification of multiple bad data in power system state estimation", IEEE Trans. Power App. And Syst., Vol. 101, No. 2, pp. 454-462, February 1982.

[83] L. MILI, T. VAN CUTSEM E M. RIBBENS-PAVELLA, "Hypothesis testing identification: A new method for bad data analysis in power system state estimation", IEEE Trans. Power App. and Syst., Vol. 103, No. 11, pp. 3239-3252, November 1984.

[84] F.F WU, E.H.E LIU E S.M. LUN, "Observability analysis and bad data-processing for state estimation with equality constraints", IEEE Trans. Power Syst., Vol. 3, pp. 541-578, May 1998.

[85] F. BROUSSOLLE, "State estimation in power systems: Detecting bad data through the sparse inverse matrix method", IEEE Trans. Power App. and Syst., Vol. 97, No. 3, pp. 678-682, May 1978.

[86] TAKAHASHI, FAGAN E CHEN, "Formation of sparse bus impedance matrix", 8th PICA Conference, Minneapolis, June 1973.

[87] P.J. HUBER, "Robust estimation of a location parameter", Annals of Mathematical Statistics, 35:73-101, 1964.

[88] F.R. HAMPEL, E.M. RONCHETTI, P.J. ROUSSEEUW E W.A. STAHEL, *Robust Statistics: The approach based influence functions,* 1986.

[89] P.J. ROUSSEEUW E A.M. LEROY, *Robust Regression and Outlier Detection,* Series in Probability and Mathematical Statistics, John Wiley & Sons, 1987.

[90] L. MILLI, V. PHANIRAJ E P.J. ROUSSEEUW, "Least Median of Squares Estimation in Power Systems", IEEE Transactions on Power System, 6(2):511-523, May 1991.

[91] L. MILLI, M.G. CHENIAE, N. S. VICHARE E P.J. ROUSSEEUW, "Robust State Estimation Based on Projection Statistics", IEEE Transactions on Power System, 11(2):1118-1127, May 1996.

[92] F.C. SCHWEPPE, J. WILDES E D.B ROM, "Power System Static State Estimation: Part I, II, II", IEEE-Transaction on Power App. And Syst., PAS-89(1):120-135, January 1970.

[93] A. MONTICELLI, *State Estimation in Electric Power Systems: A Generalized Approach, Power Eletronics and Power Systems Series,* Serie Editor: M.A. Pai, Klumer Academic Publishers, 1999.

[94] L. MILLI, V. PHANIRAJ E P.J. ROUSSEEUW, "High Breakdown Point Estimation in Power Systems", IEEE -Transactions on Power Systems, 6(2):511-523, May 1991.

[95] A. MONTECELLI E A. GARCIA, "Realible Bad Data Processing for Real-Time State Estimation", IEEE-Transaction on Power App. And Syst., PAS-102(5):1126-1139, May 1983.

[96] L. Mili, V. Cutsem e M. Ribbens-Pavella, "Bad Data Identification Methods in Power System State Estimation – a Comparative Study", IEEE-Transaction on Power App. And Syst., PAS-104:3037-3049, November 1985.

[97] H.M. MERRIL E F.C. SCHWEPPE, "Bad Data Suppression in Power System Static State Estimation", IEEE-Transaction on Power App. And Syst., PAS-90:2718-2725, November 1971.

[98] D.M. FALCÃO, P.A. COOKE E A. BRAMELLER, "Nonquadratic State Estimation: A Comparison of Methods", Proceedings of the 7th PSCC Conference, 1002:1006, 1981.

[99] F. ZHUANG E K. BALASUBRAMANIAN, "BAD DATA SUPPRESSION IN POWER SYSTEM STATE ESTIMATION WITH A VARIABLE QUADRATIC-CONSTANT CRITERION", IEEE-Transaction on Power App. And Syst., PAS-104:857-863, April 1985.

[100] A.J.A SIMÕES-COSTA E J.G. ROLIM, "Iterative Bad Data Suppression Applied to State Estimators Based on Augmented Matrix Method", Electric Power System Research, (20):205-213, 1991.

[101] R. BALDICK, K.A. CLEMENTS, Z. PINJO-DZIGAL E P.W. DAVIS, "Implementing Nonquadratic Objective Functions for State Estimation and Bad Data Rejection", IEEE-Transactions on Power Systems, 12(1):376:382, February 1997.

[102] R.C. PIRES, A.J.A. SIMÕES-COSTA E L. MILI, "Iteratively Reweighted Least-Squares State Estimation Through Givens Rotations", IEEE-Transactions on Power Systems, (4):1499-1507, November 1999.

[103] V. LOAN, "On the Method of Weightings for Equality Constrained Least-Squares Problems", SIAM J. Numer. Anal., 22(5):851-864, October 1985.

[104] J.P.S GOUVÊA E A.J.A. SIMÕES-COSTA, "Estimador de Estados Ortogonal com Restrições de Igualdade", Revista da Sociedade Brasileira de Automática, pp: 1-8, 1997.

[105] A. PADIAN, K. PATHASARATHY E S.A. SOMAN, "Towards Faster Givens Rotations Based Power System State Estimator", IEEE-Trans. on Power Systems, 14(3):837-843, 1999.