



Universidade Federal
do Rio de Janeiro

Escola Politécnica

DIAGNOSE DE FALHAS DE UMA UNIDADE DE SEPARAÇÃO TRIFÁSICA
USANDO MODELOS A EVENTOS DISCRETOS.

Ana Elisa Araújo Martins

Projeto de Graduação apresentado ao Curso de Engenharia de Controle e Automação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheira.

Orientadores: João Carlos dos Santos Basílio
Gustavo da Silva Viana

Rio de Janeiro
Outubro de 2018

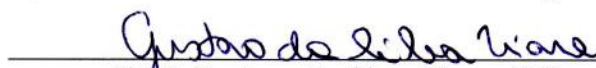
DIAGNOSE DE FALHAS DE UMA UNIDADE DE SEPARAÇÃO TRIFÁSICA
USANDO MODELOS A EVENTOS DISCRETOS.

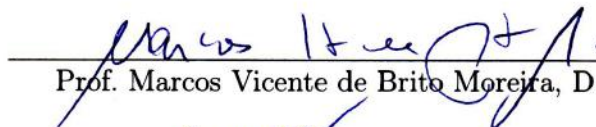
Ana Elisa Araújo Martins

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO
CURSO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO DA ESCOLA
POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO
PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU
DE ENGENHEIRA DE CONTROLE E AUTOMAÇÃO.

Examinado por:


Prof. João Carlos dos Santos Basílio, Ph.D.


Prof. Gustavo da Silva Viana, D.Sc.


Prof. Marcos Vicente de Brito Moreira, D.Sc.


Profa. Ofélia de Queiroz Fernandes Araújo, Ph.D.

RIO DE JANEIRO, RJ – BRASIL
OUTUBRO DE 2018

Martins, Ana Elisa Araújo

Diagnose de falhas de uma unidade de separação trifásica usando modelos a eventos discretos./Ana Elisa Araújo Martins. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2018.

XIII, 106 p.: il.; 29, 7cm.

Orientadores: João Carlos dos Santos Basílio

Gustavo da Silva Viana

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Controle e Automação, 2018.

Referências Bibliográficas: p. 91 – 93.

1. Sistemas a eventos discretos. 2. Autômatos. 3. Diagnose de falha. 4. Mapa de sensores. 5. Extração e recuperação de petróleo. I. Basílio, João Carlos dos Santos *et al.* II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Controle e Automação. III. Título.

*Dedico à minha avó Rosarita,
por sua força e sabedoria,
e ao meu avô Luiz Carlos,
por sua paixão pela matemática.*

Agradecimentos

Gostaria de agradecer ao orientador João Carlos Basilio pela oportunidade de participar do Laboratório de Controle e Automação (LCA/UFRJ) e por fazer deste trabalho um processo de crescimento profissional e pessoal.

Ao orientador Gustavo da Silva Viana, o meu grande agradecimento pela orientação sensível e pragmática que me propiciou seguir em frente e concluir este projeto e pela dedicação nas revisões e no treinamento para a apresentação.

Agradeço à professora Ofélia de Queiroz Fernandes Araújo por participar da banca e pela expressiva contribuição com as possibilidades de aplicações práticas deste trabalho, e ao professor Marcos Vicente Moreira pela participação na banca e pelas contribuições na teoria de Sistemas a Eventos Discretos.

Sou muito grata ao Carlos Eduardo Nunes pelas primeiras orientações e discussões acerca da diagnose de falhas, ao Victor Hugo Cruz pelas discussões sobre *sensor mapping* e *reset event*, ao Leonardo Bermeo pela solicitude em responder às questões sobre o *DESLAB*, ao Marcos Vinícius Alves e ao Felipe Cabral pelos comentários sobre a teoria de Sistemas a Eventos Discretos, à Angela Arana pelo auxílio inicial com a simulação e ao Félix Estrella pela ajuda com a bibliografia.

Pela atenção e delicadeza em revisar a apresentação deste trabalho, agradeço aos colegas Antonio Galiza, Thiago Tuxi e Juliano Ramaldes. Estendo meus agradecimentos aos colegas do LCA que tive a chance de conviver e que, não apenas contribuíram diretamente para este trabalho, como também proporcionaram um ambiente colaborativo, além de criativo em diversas áreas. Assim como agradeço aos colegas do Laboratório de Aplicações de Supercondutores que, mais do que a copa, compartilharam o apoio durante almoços e cafés.

Agradeço aos aprendizados durante o estágio na gerência de Automação, Equipamentos Dinâmicos e Confiabilidade do Centro de Pesquisa da Petrobras, uma experiência valiosa para a realização deste trabalho.

Às mulheres do Coletivo ComCiência Feminina UFRJ, em especial Thaís Oliveira, Leticia Ramos, Lorrane Morena e Thaís Rachel, agradeço pela sororidade com as questões de gênero na Engenharia e pela perspectiva de que é possível construir uma universidade solidária, social e diversa.

Gostaria de expressar minha gratidão aos irmãos Igor, Érika e Marta, pelo apoio

afetivo e pelo incentivo ao desenvolvimento de meu potencial e à amiga Roberta Lemgruber pelo entusiasmo por este trabalho, pela empatia e pela motivação.

Sou profundamente agradecida ao meu companheiro Jordan, pela força e apoio contínuos, pelos elementos fundamentais que contribuíram indiretamente para a conclusão deste projeto, bem como pelos elementos que auxiliaram diretamente, como os importantes comentários sobre a escrita e sobre a apresentação deste trabalho

Sobretudo gostaria de agradecer à minha mãe, Luiza, e meu pai, Antônio, por sempre me estimularem à criatividade, à curiosidade, aos questionamentos e aos estudos, pelo incentivo para desenvolver meus talentos e pelos ensinamentos para enfrentar os desafios de cada etapa como oportunidades para alcançar o potencial como ser humana.

“Sempre há os quatro lados da montanha.”

(Willy Chen, durante excursão em Salinas, pelo clube de montanhismo UNICERJ.)

Resumo do Projeto de Graduação apresentado à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenharia de Controle e Automação.

Diagnose de falhas de uma unidade de separação trifásica usando modelos a eventos discretos.

Ana Elisa Araújo Martins

Outubro/2018

Orientadores: João Carlos dos Santos Basílio

Gustavo da Silva Viana

Curso: Engenharia de Controle e Automação

O presente trabalho desenvolve o projeto de um sistema de diagnose de falhas para uma unidade de separação trifásica água-óleo-gás através de modelos a eventos discretos, usando a teoria dos autômatos como o formalismo para a modelagem da planta. O sistema de diagnose de falha proposto tem como objetivo ser parte de um sistema inteligente que fornece suporte operacional ao processamento primário de processos de plataforma de petróleo *offshore*. O modelo de processo adotado é baseado em um sistema real de plataforma de produção da Petrobras que opera no Campo de Marlim na Bacia de Campos. Para os cálculos com autômatos, é empregada a ferramenta computacional *DESLAB*. A plataforma *Matlab/Simulink* é utilizada para simulação dinâmica do processo, enquanto a ferramenta *Stateflow* é aplicada para validar o sistema diagnosticador de falhas acoplado ao processo simulado.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Engineer.

DISCRETE EVENT MODEL FOR FAULT DIAGNOSIS IMPLEMENTATION
ON THREE-PHASE SEPARATION SYSTEM BENCHMARK

Ana Elisa Araújo Martins

October/2018

Advisors: João Carlos dos Santos Basílio
Gustavo da Silva Viana

Course: Automation and Control Engineering

This project develops a fault diagnosis system for a three-phase separator unit through discrete event models, being automata theory the formal basis for plant modelling. The designed diagnosis system intends to be part of an intelligent system for operational support of primary treatment processes in offshore oil and gas production platforms. The adopted process model is based on a real offshore platform operating in the Marlim field, located in the northeastern part of Campos Basin. The automata calculations are performed with *DESLAB* computational tool, and the simulation platform *Simulink* is used with the *Stateflow* tool for validation of the fault diagnosis system linked to the real process simulation.

Sumário

Lista de Figuras	xi
Lista de Tabelas	xiii
1 Introdução	1
2 Sistema a eventos discretos	4
2.1 Conceitos fundamentais	4
2.2 Linguagem	6
2.3 Autômato	9
2.3.1 Operações com Autômatos	15
2.3.2 Sistemas parcialmente observáveis	18
2.3.3 Mapa de sensores	20
2.3.4 Autômato diagnosticador	21
2.3.5 Diagnosticador <i>Viana-Basilio</i>	26
2.4 Sistema de diagnose de falhas	26
2.5 Ferramenta computacional para eventos discretos	30
2.6 Conclusão do capítulo	30
3 Modelagem e simulação do processo contínuo	31
3.1 Modelo da unidade separação trifásica no processamento primário de petróleo	31
3.1.1 Modelo do separador	33
3.1.2 Equações de estado do separador	34
3.1.3 Equações algébricas de processo	36
3.1.4 Equações de saída	37
3.1.5 Modelo hidrodinâmico do hidrociclone para águas oleosas	39
3.1.6 Modelo integrado	44
3.2 Controle da unidade de separação trifásica	44
3.2.1 Estratégia de controle	44
3.2.2 Modelo do sistema de controle da unidade de separação	50
3.3 Simulação do processo	52

3.4	Conclusão do capítulo	58
4	Modelo a eventos discretos	59
4.1	Modelo do sistema a eventos discretos	59
4.1.1	Modelagem da válvula de controle	60
4.1.2	Modelagem do controlador	61
4.1.3	Modelagem da planta	63
4.2	Modelo do diagnosticador de falha	63
4.2.1	Mapa de sensores	67
4.3	Simulação com diagnosticador de falhas	70
4.3.1	Implementação com Simulink	70
4.3.2	Resultados	71
4.3.3	Falhas na válvula de óleo	74
4.3.4	Falhas na válvula de água	74
4.3.5	Falhas na válvula de óleo dos hidrocilones BOW e PDC	79
4.3.6	Falhas na válvula de óleo do hidrocilone DC	79
4.3.7	Falhas na válvula de gás	86
4.3.8	Avaliação dos diagnosticadores de falha	89
4.4	Conclusão do capítulo	89
5	Conclusões e trabalhos futuros	90
	Referências Bibliográficas	91
A	Código Fonte	94
A.1	Código Python com <i>DESLAB</i>	94
A.1.1	<i>Toolbox</i> para autômato diagnosticador	94
A.1.2	Composição paralela para o modelo da Planta	96
A.1.3	Autômato diagnosticador da Planta através da busca por com- ponentes fortemente conexos	97
A.1.4	Mapa de sensores	98
A.1.5	Diagnosticador do modelo após mapa de sensores	99
A.1.6	<i>Toolbox</i> para exportar autômatos do <i>DESLAB</i> ao <i>Stateflow</i>	99
A.2	Arquivo <i>Matlab</i> que gera <i>Stateflow</i> automaticamente - exemplo criado com a <i>toolbox</i> da seção A.1.6	102

Lista de Figuras

2.1	Diagrama de transição de estados representado como autômato.	10
2.2	Exemplo de autômato.	12
2.3	Autômatos com saídas <i>Moore</i> e <i>Mealy</i> equivalentes [1].	14
2.4	Composição produto entre os autômatos das figuras 2.1 e 2.2.	18
2.5	Composição paralela entre os autômatos das figuras 2.1 e 2.2.	18
2.6	Observador equivalente através do algoritmo 2.1.	20
2.7	Diagnosticador para o autômato parcialmente observável da figura 2.6a.	23
2.8	Autômato rotulador.	24
3.1	Diagrama de uma unidade de processamento primário de petróleo [2].	32
3.2	Vaso separador trifásico.	34
3.3	Tipos de hidrociclones.	39
3.4	Modelo do hidrociclone.	41
3.5	Modelo da bateria de três hidrociclones.	43
3.6	Fluxograma do separador trifásico integrado com bateria de hidrociclones e malhas de controle.	45
3.7	Resposta do sistema de separação em malha fechada após mudança nos <i>setpoints</i> de controle.	55
3.8	Resposta do sistema de separação em malha fechada após degrau nas variáveis de entrada.	56
3.9	Balanco de massa de líquidos do sistema de separação simulado.	57
4.1	Autômato da válvula de controle.	61
4.2	Autômato do controlador.	63
4.3	Ciclo em uma componente fortemente conexa que viola a condição de diagnosticabilidade no autômato G_{scc}	65
4.4	Modelo em <i>Stateflow</i> para o diagnosticador da falha <i>so</i>	70
4.5	Gerador de eventos a partir de sensores reais e virtuais.	71
4.6	Modelo em <i>Simulink</i> dos diagnosticadores de falha.	72
4.7	Modelo em <i>Simulink</i> do controlador e modos de falhas.	72
4.8	Modelo em <i>Simulink</i> do hidrociclone e modos de falha.	73

4.9	Variáveis controladas após falha aberta na válvula de óleo S_o .	75
4.10	Variáveis manipuladas após falha aberta na válvula de óleo S_o .	75
4.11	Variáveis controladas após falha fechada na válvula de óleo S_o .	76
4.12	Variáveis manipuladas após falha fechada na válvula de óleo S_o .	76
4.13	Variáveis controladas após falha aberta na válvula de água S_{w3} .	77
4.14	Variáveis manipuladas após falha aberta na válvula de água S_{w3} .	78
4.15	Variáveis controladas após falha fechada na válvula de água S_{w3} .	78
4.16	Variáveis manipuladas após falha fechada na válvula de água S_{w3} .	79
4.17	Variáveis controladas após falha aberta na válvula de óleo BOW S_{o1} .	80
4.18	Variáveis manipuladas após falha aberta na válvula de óleo BOW S_{o1} .	80
4.19	Variáveis controladas após falha fechada na válvula de óleo BOW S_{o1} .	81
4.20	Variáveis manipuladas após falha fechada na válvula de óleo BOW S_{o1} .	81
4.21	Variáveis controladas após falha aberta na válvula de óleo PDC S_{o2} .	82
4.22	Variáveis manipuladas após falha aberta na válvula de óleo PDC S_{o2} .	82
4.23	Variáveis controladas após falha fechada na válvula de óleo PDC S_{o2} .	83
4.24	Variáveis manipuladas após falha fechada na válvula de óleo PDC S_{o2} .	83
4.25	Variáveis controladas após falha aberta na válvula de óleo DC S_{o3} .	84
4.26	Variáveis manipuladas após falha aberta na válvula de óleo DC S_{o3} .	85
4.27	Variáveis controladas após falha fechada na válvula de óleo DC S_{o3} .	85
4.28	Variáveis manipuladas após falha fechada na válvula de óleo DC S_{o3} .	86
4.29	Variáveis controladas após falha aberta na válvula de gás S_g .	87
4.30	Variáveis manipuladas após falha aberta na válvula de "gás S_g .	87
4.31	Variáveis controladas após falha fechada na válvula de gás S_g .	88
4.32	Variáveis manipuladas após falha fechada na válvula de gás S_g .	88
4.33	Diagnose de falha realizada através da simulação com <i>Stateflow</i> .	89

Lista de Tabelas

3.1	Variáveis de estado do separador.	36
3.2	Variáveis de entrada e de saída do separador.	36
3.3	Parâmetros de modelagem do separador trifásico.	36
3.4	Variáveis de processo.	37
3.5	Parâmetros de modelagem das variáveis de processo.	37
3.6	Variáveis de controle.	38
3.7	Parâmetros para modelagem das vazões de saída do separador.	38
3.8	Variáveis da modelagem do hidrociclone.	41
3.9	Parâmetros do hidrociclone.	41
3.10	Parâmetros da série de hidrociclones BOW, PDC e DC.	44
3.11	Valores de <i>setpoint</i> das malhas de controle.	52
3.12	Variáveis manipuladas do sistema de controle do separador.	52
3.13	Parâmetros dos controladores PI.	52
3.14	Mudança de <i>setpoint</i> das malhas de controle.	54
3.15	Mudança nas variáveis de entrada.	55
4.1	Tabela de transição de estados da válvula	61
4.2	Tabela de transição de estados do controlador	62
4.3	Tabela de transição de estados da Planta	64
4.4	Tabela de transição de estados do Diagnosticador para a falha sc	68

Capítulo 1

Introdução

Os sistemas de diagnose de falhas são cruciais em processos de engenharia de sistemas complexos, tanto sob os aspectos de desempenho, confiabilidade e qualidade dos processos, quanto sob os aspectos de riscos e de prevenção de acidentes. Os sistemas FDD são ponto central do gerenciamento de eventos anormais (AEM, do inglês *abnormal event management*) e apontam, diagnosticam e classificam possíveis falhas [3].

A indústria de produção *offshore* de petróleo e gás, em que os equipamentos operam continuamente 24/7, 365 dias ao ano, e estão sujeitos à mudança abrupta das condições operacionais, pode ser amplamente beneficiada pelo uso de técnicas analíticas e preditivas para identificar falhas na maquinaria e problemas em estágios iniciais, não apenas para minimizar as perdas de produção indesejada, como também para evitar a subsequência de eventos catastróficos e, sobretudo, para reduzir o risco potencial de materiais inflamáveis serem liberados ao ambiente ocasionando fogo e explosão.

A análise, o monitoramento e o controle dos equipamentos industriais é ainda mais crítico no setor de produção petróleo e gás *offshore* no Brasil, que possui o desafio trazido pelas lâminas d'água profundas e pelo óleo pesado, o que também proporciona o aumento na quantidade de água produzida nos campos marítimos [2]. Desta forma, além da separação entre o óleo e o gás como uma das primeiras operações após o óleo cru ser extraído dos poços petrolíferos, é necessário o tra-

tamento primário das águas oleosas pelas unidades de processamento primário de petróleo.

A unidade de separação é responsável por separar as fases oleosa, aquosa e gasosa e impurezas sólidas da produção de petróleo, e se baseia no princípio da gravidade para permitir que o fluido de maior densidade, ou seja, a água, se deposite no fundo do vaso separador, enquanto os de menor densidade, o óleo e o gás, fluam para o topo do vaso. Além disso, podem ser empregados hidrociclones a jusante da água separada para reduzir os teores de óleo nas águas descartadas ou reinjetadas.

O dispositivo mais usado para o controle de nível e de pressão em uma unidade de separação é a válvula de controle [4], que envia um sinal de controle ao atuador da válvula e utiliza algoritmos de controle como o proporcional-integrativo. Segundo [4], 20% das falhas em vasos separadores advém das válvulas de controle, portanto, é um aspecto relevante para o enfoque da diagnose de falhas. Dentre as causas possíveis de falhas na válvula, estão o agarramento, a folga mecânica, o suprimento de pressão e falhas por falta de energia.

Neste trabalho, é desenvolvido um modelo de diagnose de falhas na válvula de controle por agarramento nas posições totalmente aberta ou totalmente fechada. A opção de se realizar a diagnose de falhas através de sistemas a eventos discretos (SED) é por ser uma abordagem simples frente a complexidade da planta, além de ser uma recente área de pesquisa [5].

Os SEDs podem ser aplicados aos processos de natureza discreta, como manufatura, transporte, comunicação, redes e engenharia de software. No entanto, é um tema de pesquisa ativa a utilização de modelos com estados discretos e dirigido por evento aplicado à diagnose de falhas em sistemas de natureza contínua, como em controle de processos industriais [5]. Para a modelagem de um sistema com um comportamento físico, isto é, descrito por modelos de equações diferenciais, um desafio é capturar seu comportamento lógico e sequencial para aplicar o arcabouço da teoria de SEDs e detectar as falhas na planta.

Neste trabalho é proposto um modelo de diagnosticador de falhas por eventos

discretos para uma unidade de separação trifásica a partir da modelagem contínua realizada para o caso real da plataforma *offshore* da empresa Petrobras localizada no Campo de Marlim. O modelo contínuo do processo é simulado e validado em plataforma *Simulink* e um modelo discreto para a diagnose baseado no comportamento discreto da planta será calculado e verificado com auxílio da ferramenta computacional *DESLAB*. Por fim, com a ferramenta *Stateflow*, será analisado o comportamento do diagnosticador junto ao processo contínuo.

Este trabalho está estruturado da seguinte forma: o capítulo 2 aborda os conceitos fundamentais da teoria de sistemas a eventos discretos e as operações com linguagens e com autômatos, além de tratar da diagnose de falhas e dos algoritmos executados para diagnosticadores e para verificadores. Em seguida, o capítulo 3 descreve o modelo contínuo do processo e do controle do separador trifásico com hidrociclones, bem como sua simulação dinâmica. No capítulo 4 são apresentados os modelos a eventos discretos da planta e do diagnosticador de falha, assim como é mostrada a simulação da diagnose de falhas com a ferramenta *Stateflow* aplicada à simulação contínua do processo. Por fim, o capítulo 5 trata da conclusão e de trabalhos futuros.

Capítulo 2

Sistema a eventos discretos

Neste capítulo, é abordada a base da teoria de sistemas a eventos discretos (SED), utilizada na realização deste trabalho. A seção 2.1 descreve os fundamentos de SEDs, enquanto a seção 2.2 elucida a representação de SEDs através de linguagens e algumas operações possíveis. A seção 2.3 ilustra a representação das linguagens por autômatos e o formalismo desta modelagem, além de descrever os autômatos observador e diagnosticador, usados como técnica para resolver e analisar problemas de sistemas com eventos parcialmente observáveis e para o tratamento da diagnose de eventos. A abordagem de diagnose através do mapa de sensores é vista na seção 2.3.3. Na seção 2.3.4, é descrito um algoritmo de diagnosticador apresentado em [6] que busca por ciclos, enquanto a seção 2.3.5 descreve uma alternativa ao algoritmo do diagnosticador através de algoritmo que busca componentes fortemente conexos. A seção 2.4 detalha os conceitos do sistema diagnosticador de falhas. Por fim, a seção 2.5 retrata a ferramenta computacional utilizada neste trabalho para análise e síntese de SEDs.

2.1 Conceitos fundamentais

Neste trabalho, são considerados os sistemas dinâmicos classificados como sistemas a eventos discretos (SEDs) [1],[7]. Um *sistema* é definido como a combinação de elementos que realizam em conjunto uma tarefa que nenhum dos elementos é

capaz de realizar individualmente. Para que o sistema se mantenha com o comportamento desejado, projeta-se um sistema de controle. Um sistema é chamado *dinâmico* quando a saída atual depende dos valores passados da entrada, além de poder depender também de outras saídas do sistema. O estado de um sistema é a informação necessária para determinar, de forma única, a saída a partir da entrada em determinado instante de tempo. Desta forma, as variáveis de estado descrevem a relação entre entrada e saída do sistema.

Um sistema possui estados discretos quando o conjunto formado por todos os valores possíveis das variáveis de estado, isto é, o espaço de estados, é um conjunto discreto. Diferentemente do caso contínuo, o comportamento das variáveis de um sistema discreto não pode ser descrito por equações diferenciais. Um exemplo é o sistema de depósito de uma fábrica, que pode ser modelado a partir do número de produtos chegando ao depósito como a entrada, do número de produtos saindo como a saída e, como variável de estado, o número de caixas no depósito, que é uma variável discreta.

Já quando a entrada e a saída são definidas somente em instantes de tempo discreto, tem-se o chamado sistema discreto no tempo. O fato da variável de estado ser discreta no tempo não implica em um espaço de estados discreto.

A classe de *sistemas a eventos discretos* (SED), ocorre quando os estados evoluem a partir de eventos, isto é, a mudança de estados de um sistema é promovida a partir de estímulos externos ou internos em intervalos de tempo não necessariamente conhecidos ou regulares. Portanto, somente a passagem do tempo não é suficiente para que o sistema evolua, ou seja, a transição de estados é dirigida por eventos.

Além disto, nos SEDs, todos os valores possíveis para o estado são descritos por um conjunto discreto de valores e todas as transições de estado são observadas somente em pontos discretos no tempo. É considerado que o evento ocorre instantaneamente e ocasionam a transição de um valor de estado para outro.

Os sistemas modelados por variáveis contínuas são relacionados a fenômenos naturais que lidam com quantidades como posição, velocidade e aceleração de corpos

rígidos, assim como a fenômenos decorrentes de transferências de massa, energia e momento, com variáveis como pressão, temperatura e vazão de fluidos e gases, ou até mesmo descrevem sistemas econômicos e populacionais. Já os SEDs tratam de quantidades que envolvam a contagem de números inteiros, como a chegada de um cliente numa fila, ou como o número de aviões na pista de pouso e decolagem, ou como as variáveis discretas que se encontram nos sistemas atuais dependentes de computador. Estes sistemas também dependem de eventos para a transição de seus estados, como o apertar de um botão ou como a mudança da luz de um semáforo. Desta forma, como resposta à ocorrência do evento, há uma mudança no sistema, que pode ou não ser observada externamente, e permanece nesse estado até o próximo evento. Exemplos de SEDs ocorrem nos sistemas com procedimentos e receitas de operação, tais como sistemas que implicam em acionamento e desligamento de bombas e compressores.

Como não é possível modelar um SED através de equações diferenciais ou à diferenças, para descrever o comportamento de um SED, é necessário conhecer a sequência dos estados percorridos, bem como os eventos que provocaram tais transições de estado. Assim, considera-se o conjunto de eventos de um SED como o *alfabeto* do sistema e a *palavra* do sistema cada sequência de eventos de tamanho finito. Chama-se *linguagem* o conjunto de todas as sequências de comprimento finito geradas por um sistema e será formalmente definida na seção 2.2.

2.2 Linguagem

A teoria de linguagens é o formalismo usado para desenvolver modelos apropriados aos SEDs que descrevam adequadamente o comportamento desses sistemas. Ela fornece uma estrutura com técnicas analíticas para alcançar os objetivos de projeto, controle e avaliação de desempenho.

O comportamento de um SED é descrito pelo conhecimento da sequência de estados visitados e dos eventos associados às transições de estados. Isto é, o comportamento é dado em termos da sequência de eventos na forma $e_1e_2\dots e_n$, de modo

que não é considerado os instantes de tempo que ocorrem tais eventos, apenas a ordem temporal. Esse nível de abstração é chamado de lógico ou atemporal, e é modelado pela linguagem. O conjunto de eventos E associado ao sistema é o *alfabeto* da linguagem. Já a *palavra* s é uma sequência finita de eventos e o comprimento $|s|$ de s é dado pelo número de eventos na palavra. Uma palavra sem eventos possui comprimento zero e é chamada de ϵ .

Assim, define-se formalmente uma linguagem como:

Definição 2.1 (*Linguagem*)

Uma linguagem L definida sobre um conjunto de eventos E é um conjunto de palavras de comprimento finito formada pelos eventos em E .

A *concatenação* é a operação principal para linguagens. A concatenação s_1s_2 de duas palavras, s_1 e s_2 , é a nova palavra formada pela sequência de eventos s_1 imediatamente seguida pela sequência de eventos s_2 . A concatenação pode ser definida da seguinte forma:

Definição 2.2 (*Concatenação*)

Sejam $L_1, L_2 \subseteq E^*$, então

$$L_1L_2 := \{s \in E^* | (s = s_1s_2) \wedge (s_1 \in L_1) \wedge (s_2 \in L_2)\}$$

A palavra vazia ϵ é o elemento neutro da concatenação, isto é, para uma palavra s , tem-se que $s\epsilon = \epsilon s = s$.

Outra operação é o *Fecho de Kleene*, denotado por E^* , que é o conjunto com *todas* as palavras finitas compostas de elementos de E . Tal conjunto E^* é infinito, já que possui todas as palavras de tamanho arbitrariamente longo. Logo, uma linguagem para um conjunto de eventos E é um subconjunto de E^* . Como casos particulares de linguagens, temos os conjuntos \emptyset , E , E^* . De forma análoga, ao estender a definição acima para o caso de palavras de tamanho maior do que 1, tem-se a operação de fecho de Kleene para linguagens L^* .

Definição 2.3 (*Fecho de Kleene*)

Seja $L \subseteq E^*$, então

$$L^* := \{\epsilon\} \cup L \cup LL \cup LLL \dots$$

Um elemento de L^* é portanto formado pela concatenação de um número finito, porém, arbitrariamente longo de elementos de L . Uma vez que, pela definição, $(L^*)^* = L^*$, a operação fecho de Kleene é idempotente.

Seja uma palavra $s = abc$ com $a, b, c \in E^*$, então a é chamado de *prefixo* de s , b é chamada de *subpalavra* e c o *sufixo* de s . O sufixo de s após o prefixo a de s é definido por s/a . Note que ambos ϵ e s podem ser considerados prefixos, subpalavras ou sufixos de s . Desta forma, são definidas também a operação de *fecho de prefixo* \bar{L} , que consiste no conjunto de todos os prefixos de todas as palavras em L , e a operação de *pós-linguagem* de L após s , ou L/s ,

Definição 2.4 (*Fecho de Prefixo*)

Seja $L \subseteq E^*$, então

$$\bar{L} := \{s \in E^* \mid (\exists a \in E^*)(sa \in L)\}.$$

Se qualquer prefixo de qualquer palavra de uma linguagem L for também um elemento de L , então $L = \bar{L}$ e a linguagem L é chamada de *prefixo fechada*.

Definição 2.5 (*Pós-Linguagem*)

Seja $L \subseteq E^*$ e $s \in L$, então

$$L/s := \{t \in E^* \mid st \in L\}.$$

Por definição, se $s \notin \bar{L}$, então $L/s = \emptyset$.

Outra operação importante é a *projeção* P de palavras e linguagens, também chamada de *projeção natural*, a partir de um conjunto de eventos E_a para um conjunto menor de eventos E_b , isto é, $E_b \subset E_a$. Esta operação remove os eventos que não pertencem ao conjunto menor de eventos E_b das palavras formadas a partir de um conjunto maior de eventos E_a .

Definição 2.6 (*Projeção*)

Seja $P : E_a^* \rightarrow E_b^*$, a projeção P é definida como:

$$\begin{aligned} P(\epsilon) &:= \epsilon \\ P(e) &:= \begin{cases} e, & \text{se } e \in E_b \\ \epsilon, & \text{se } e \in E_a \setminus E_b \end{cases} \\ P(se) &:= P(s)P(e), \text{ para } s \in E_a^*, e \in E_a. \end{aligned}$$

A ideia do mapeamento inverso da projeção P^{-1} é, que dada uma sequência s , obtém-se todas as sequências s' tais que $P(s') = s$. A operação é definida como:

Definição 2.7 (*Projeção Inversa*)

Seja $P^{-1} : E_b^* \rightarrow 2^{E_a^*}$, então

$$P^{-1}(t) := \{s \in E_a^* | P(s) = t\}.$$

Os conceitos de projeção e projeção inversa podem ser estendidos para linguagens da seguinte forma.

- Seja $L \subseteq E_a^*$, então $P(L) := \{t \in E_b^* | (\exists s \in L)(P(s) = t)\}$.
- Para $L_b \subseteq E_b^*$, $P^{-1}(L_b) := \{s \in E_a^* | (\exists t \in L_b)(P(s) = t)\}$.

Estende-se, portanto, as definições para todas as palavras na linguagem.

Após esta seção apresentar as definições das operações para linguagens, é apresentado um formalismo usado para representar linguagens, que fornece uma estrutura com operações bem definidas e fácil de manipular. Este formalismo, para a modelagem atemporal dos SEDs, é chamado de *autômato*.

2.3 Autômato

O *autômato* é uma ferramenta matemática para expressar linguagens com regras bem definidas. Para apresentá-lo, é usado a representação através de um *diagrama de transição de estados*, ou seja, um grafo direcionado, em que os nós representam os estados do sistemas, enquanto as arestas representam as transições entre os estados. O exemplo da figura 2.1 ilustra uma representação do autômato.

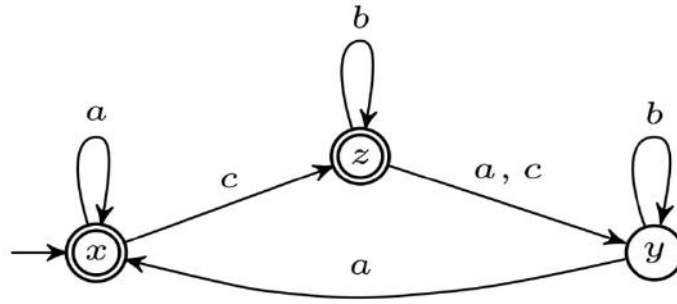


Figura 2.1: Diagrama de transição de estados representado como autômato.

A definição formal do autômato é dada através da *máquina de estados*, ou *gerador* G , a seguir.

Definição 2.8 (*Autômato Determinístico*)

Um autômato determinístico G é a sêxtupla

$$G = (X, E, f, \Gamma, x_0, X_m)$$

tal que

- X é o conjunto de estados,
- E é o conjunto finito de eventos associado com G ,
- $f : X \times E \rightarrow X$ é a função de transição,
- $\Gamma : X \rightarrow 2^E$ é a função de eventos ativos, isto é, o conjunto de todos os eventos para os quais a função de transição é definida,
- x_0 é o estado inicial,
- $X_m \subseteq X$ é o conjunto de estados marcados.

Note que, a partir da definição 2.8, um evento pode ocorrer sem que o sistema mude de estado, como em $f(x, a) = x$; tal que $x \in X$ e $a \in E$. Além disso, dois eventos diferentes podem ocorrer em um dado estado e causar a transição para um mesmo estado. Como não é necessário haver para cada estado em X uma transição definida para cada evento em E , a função f é chamada de parcial no domínio $X \times E$. Uma vez que a função de transição só determina um único estado evoluído a partir de um mesmo evento, o autômato é considerado determinístico. Além disso, é necessário definir o estado inicial x_0 e X_m , que é um subconjunto de X

que define estados marcados correspondentes ao término de uma tarefa ou operação pelo sistema.

Para o exemplo da figura 2.1, é visto que:

$$\begin{aligned}
 X &= x, y, z \\
 E &= a, b, g \\
 f(x, a) &= f(y, a) = x \\
 f(x, c) &= f(z, b) = z \\
 f(z, a) &= f(z, c) = f(y, b) = y
 \end{aligned} \tag{2.1}$$

A dinâmica de um sistema, modelado por um autômato, ocorre da seguinte forma. O sistema começa no estado inicial x_0 e, após a ocorrência de um evento $e \in \Gamma(x_0) \subseteq E$, haverá a transição para um estado $f(x_0, e) \in X$. O processo continua de acordo com as transições em que a função f é definida. Por uma questão de conveniência, a função f é sempre estendida do domínio $X \times E$ para o domínio $X \times E^*$ recursivamente da seguinte forma:

$$\begin{aligned}
 f(x, \epsilon) &:= x \\
 f(x, se) &:= f(f(x, s), e) \text{ para } s \in E^*, e \in E
 \end{aligned}$$

Esta é, então, chamada de função de transição estendida. A partir desta, define-se a *linguagem gerada* por um autômato G , $\mathcal{L}(G)$, que fornece todas as sequências de eventos possíveis de serem percorridas em um diagrama de transição de estados a partir do estado inicial. É formalmente descrita a seguir.

Definição 2.9 (*Linguagem gerada por um autômato*)

$$\mathcal{L}(G) := \{s \in E^* \mid f(x_0, s)!\},$$

tal que a notação “!” significa “é definida”. Neste caso, trabalha-se com a função de transição estendida $f : X \times E^* \rightarrow X$.

Note que $\mathcal{L}(G)$ é prefixo fechada, já que um caminho só é percorrido se todos os seus prefixos forem também percorridos. Por consequência da definição acima, para um autômato G com conjunto de estados X não nulo, $\epsilon \in \mathcal{L}(G)$. Além disso, uma palavra s pertence à linguagem gerada $\mathcal{L}(G)$ se, e somente se, f é definida em (x_0, s) , isto é, se essa sequência de eventos corresponde a um caminho admissível no diagrama de transição de estados. Se f é uma função total no domínio, então $\mathcal{L}(G) = E^*$. Um autômato é chamado vazio se gera o conjunto vazio. Dois autômatos, G_a e G_b , são chamados de *equivalentes* quando $\mathcal{L}(G_a) = \mathcal{L}(G_b)$.

A figura 2.2 mostra um exemplo de autômato, cuja linguagem gerada é $\mathcal{L}(G) = \{\epsilon, a, aa, ab, aba, aaa, aab, \dots\}$.

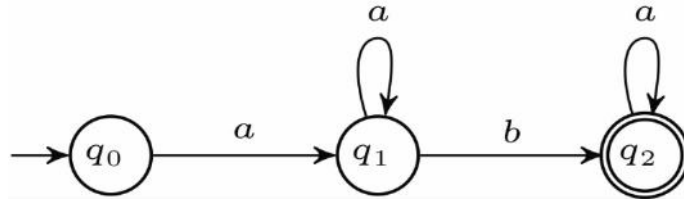


Figura 2.2: Exemplo de autômato.

O subconjunto de $\mathcal{L}(G)$ composto somente pelas palavras que levam do estado inicial ao estado marcado é chamado de linguagem marcada por um autômato, ou linguagem reconhecida por um autômato, e é definida formalmente a seguir.

Definição 2.10 (*Linguagem marcada por um autômato*)

$$\mathcal{L}_m(G) := \{s \in \mathcal{L}(G) \mid f(x_0, s) \in X_m\},$$

A linguagem marcada por um autômato não é necessariamente fechada em prefixo, o que ocorre somente quando todos os estados forem marcados. Assim, para um autômato G , se X_m é um subconjunto estritamente contido em X , então $\mathcal{L}_m(G) \subset \overline{\mathcal{L}_m(G)}$. Analogamente, $X_m = X \iff \mathcal{L}_m(G) = \overline{\mathcal{L}_m(G)}$.

Um autômato possui estados de bloqueio quando G alcança um estado x , tal que $x \notin X_m$ e $\Gamma(x) = \emptyset$. Neste caso, como nenhum evento é associado ao estado x , ele é chamado de *deadlock*.

Definição 2.11 (*Bloqueio*)

Um autômato apresenta estados de bloqueio se $\bar{\mathcal{L}}_m(G) \subset \mathcal{L}(G)$.

E não bloqueio quando $\bar{\mathcal{L}}_m(G) = \mathcal{L}(G)$.

Um autômato não determinístico ocorre quanto for necessário modificar as seguintes condições de um autômato determinístico:

- Todas as transições possuem eventos $e \in E$.
- O estado inicial é um conjunto unitário.
- Um evento $e \in \Gamma(x)$ causa uma transição de um estado x para um único estado $y = f(x, e)$.

Em geral, isto é necessário para análise ou para modelagem de um sistema. Um exemplo é o caso em que autômatos precisam ser combinados e múltiplas transições ocorrem com o mesmo rótulo de evento, assim $f(x, e)$ representaria um conjunto de estados. Além disso, a palavra nula ϵ , ou seja, sem eventos, pode estar presente no diagrama de transição de estados de um autômato, isto é, as transições entre distintos estados teriam ϵ como rótulo. Isso é relevante na modelagem de SEDs em que não é possível observar o evento que representa uma mudança no estado interno de um sistema. Um exemplo é quando não há um sensor para uma determinada transição de estado ou modelo de inferência. Assim, o evento é denominado *não-observável* e é usada a transição ϵ para representar que a transição entre dois estados pode ocorrer. Além disso, na análise do comportamento do sistema, algumas transições podem necessitar ter seus rótulos apagados, usando, portanto, ϵ como rótulo. Como ϵ é o elemento identidade para concatenação de palavras, o rótulo ϵ não é visto quando ocorre uma transição. A definição 2.12 descreve formalmente um autômato não-determinístico.

Definição 2.12 (*Autômato Não Determinístico*)

Um autômato não determinístico G_{nd} é a sêxtupla

$$G_{nd} = (X, E \cup \{\epsilon\}, f_{nd}, \Gamma, x_0, X_m)$$

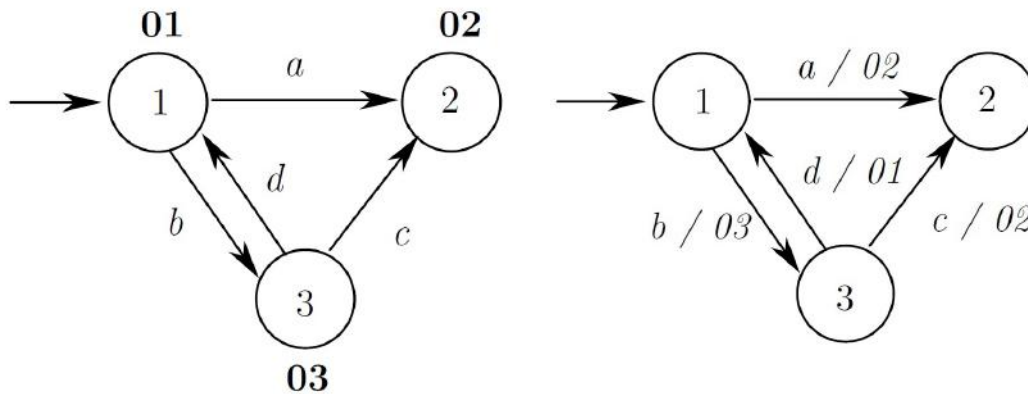
em que os parâmetros possuem o mesmo significado da definição 2.8 com exceção de:

- $f_{nd} : X \times E \cup \{\epsilon\} \rightarrow 2^X$, ou seja, $f_{nd}(x, e) \subseteq X$ onde f_{nd} estiver definida.
- $x_0 \subseteq X$, ou seja, o estado inicial pode ser um conjunto de estados.

Para caracterizar as palavras marcadas e geradas de autômatos não determinísticos com transições ϵ , é necessário usar a função f_{nd}^{exp} , que é a extensão de f_{nd} ao domínio $X \times E^*$. Para definir $f_{nd}^{exp}(x, \epsilon)$, é preciso inicialmente definir $\epsilon R(x)$, isto é, o conjunto de todos os estados que são alcançáveis a partir de x ao percorrer as transições rotuladas com ϵ no diagrama de transição de estados. Por convenção, $x \in \epsilon R(x)$. A construção de f_{nd} é feita recursivamente, primeiro define-se $f_{nd}^{exp}(x, \epsilon)$, em seguida, para $u \in E^*$ e $e \in E$, tem-se $f_{nd}^{exp}(x, ue) := \epsilon R[z | z \in f_{nd}(y, e)]$ para algum estado $y \in f_{nd}^{ext}(x, u)$.

Para modelagem de sistemas, é útil a variação de autômatos com *entradas e saídas*, como *Moore* e *Mealy* [1].

Os autômatos *Moore* são autômatos em que há uma função que atribui uma saída para cada estado. Um exemplo é mostrado na figura 2.3a. Neste caso, há uma função de saída que atribui uma saída para cada estado, que é emitida pelo autômato quando alcança o estado correspondente. É como a generalização do autômato padrão que possui saídas binárias para cada estado: marcado e não marcado.



(a) Autômato *Moore*.

(b) Autômato *Mealy*.

Figura 2.3: Autômatos com saídas *Moore* e *Mealy* equivalentes [1].

Por sua vez, os autômatos *Mealy* são autômatos com entradas e saídas. A figura 2.3b exemplifica como as transições são rotuladas: por um par de eventos na forma

(evento de entrada)/(evento de saída). O conjunto de eventos de saída não é necessariamente o mesmo conjunto de eventos de entrada. Assim, para uma transição e_i/e_o de um estado x a um estado y , se o autômato recebe o evento de entrada e_i quando está em um estado x , ele fará a transição a um estado y em um processo que emite o evento de saída e_o . Um autômato *Mealy* pode ser interpretado como um autômato padrão em que o conjunto de eventos E (veja definição 2.8) seria o conjunto de todos os eventos rotulados como *entrada/saída*, e a linguagem gerada seria todas as palavras da forma *entrada/saída* que podem ser geradas pelo autômato *Mealy*.

Para os autômatos *Moore* serem vistos como um autômato padrão, primeiro é necessário interpretá-lo como um autômato *Mealy*, ao considerar a saída de um estado associada a todos os eventos que entram no estado, para então repetir o processo da conversão de *Mealy* para um autômato padrão. A conversão entre *Moore* e *Mealy* é ilustrada na figura 2.3.

As noções de *estado de saída* e de *evento de saída* são úteis para a modelagem a eventos discretos de sistemas eletromecânicos e de outras aplicações com sensores do estado físico do sistema, e será usada para modelar, na seção 4.1, o sistema industrial descrito no capítulo 3. As operações realizadas com autômatos serão descritas na próxima subseção.

2.3.1 Operações com Autômatos

As operações com autômatos permitem combinar dois autômatos ou mais, assim, modelos de sistemas completos podem ser construídos a partir de modelos de componentes individuais do sistema. A operação *parte acessível* de um autômato G , $Ac(G)$, ocorre ao remover os estados que, a partir do estado inicial, não são alcançáveis por nenhuma palavra da linguagem gerada. Ao remover o estado, também são removidas todas as transições associadas a tal estado. Portanto, as linguagens geradas e marcadas por G e por $Ac(G)$ são as mesmas. Um autômato é chamado de acessível quando $G = Ac(G)$. Formalmente, a operação de tomar a parte acessível de um

autômato é definida a seguir.

Definição 2.13 (*Parte Acessível*)

$$Ac(G) := (X_{ac}, E, f_{ac}, \Gamma_{ac}, x_0, X_{ac,m}),$$

tal que

- $X_{ac} = \{x \in X \mid (\exists s \in E^*)(f(x_0, s) = x)\}$,
- $X_{ac,m} = X_m \cap X_{ac}$,
- $f_{ac} = f|_{X_{ac} \times E} : X_{ac} \times E \rightarrow X_{ac}$.

Um estado x de um autômato G é coacessível ao conjunto de estados marcados se existe um caminho no diagrama de transição de estados do autômato que leve x a um estado marcado. A operação de deletar todos os estados de um autômato G que não são coacessíveis é chamada de $CoAc(G)$ e é definida a seguir.

Definição 2.14 (*Parte Coacessível*)

$$CoAc(G) := (X_{coac}, E, f_{coac}, \Gamma_{coac}, x_{0,coac}, X_m)$$

tal que

- $X_{coac} = \{x \in X \mid (\exists s \in E^*)(f(x, s) = x)\}$,
- $f_{coac} = f|_{X_{coac} \times E} : X_{coac} \times E \rightarrow X_{coac}$,
- $f_{coac} = \begin{cases} x_0, & \text{se } x_0 \in X_{coac} \\ \text{indefinido}, & \text{se } x_0 \notin X_{coac} \end{cases}$

A operação *trim* ocorre quando um autômato é tanto acessível quanto coacessível, isto é,

$$Trim(G) := CoAc[Ac(G)] = Ac[CoAc(G)].$$

A operação de *projeção* de um autômato G em um conjunto de eventos E_s contido no conjunto E de eventos de G ocorre ao substituir por ϵ todos os rótulos das transições no conjunto $E \setminus E_s$.

Para a operação de *projeção inversa*, considere a linguagem $K_s = \mathcal{L}(G) \subseteq E_s^*$ e seja um conjunto de eventos $E_l \supset E_s$. Seja P_s a projeção a partir de E_l^* para

E_s^* . Assim, um autômato que gera $P_s^{-1}(K_s)$ pode ser obtido ao adicionar laços para todos os eventos em $E_l \setminus E_s$ em todos os estados de G .

Para unir o comportamento de um conjunto de autômatos que operam concorrentemente, é necessário utilizar as operações de composição entre autômatos, seja a *composição produto* \times ou a *composição paralela* \parallel .

Sejam os autômatos acessíveis $G_1 = (X_1, E_1, f_1, \Gamma_1, x_{01}, X_{m1})$ e $G_2 = (X_2, E_2, f_2, \Gamma_2, x_{02}, X_{m2})$.

Definição 2.15 (*Composição Produto*)

$$G_1 \times G_2 := Ac(X_1 \times X_2, E_1 \cup E_2, f, \Gamma_{1 \times 2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}),$$

tal que

- $f((x_1, x_2), e) := \begin{cases} (f_1(x_1, e), f_2(x_2, e)), & \text{se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ \text{indefinido}, & \text{se } e \notin \Gamma_1(x_1) \cap \Gamma_2(x_2) \end{cases}$
- $\Gamma_{1 \times 2}(x_1, x_2) = \Gamma_1(x_1) \cap \Gamma_2(x_2)$

Definição 2.16 (*Composição Paralela*)

$$G_1 \parallel G_2 := Ac(X_1 \times X_2, E_1 \cup E_2, f, \Gamma_{1 \parallel 2}, (x_{01}, x_{02}), X_{m1} \times X_{m2}),$$

tal que

- $f((x_1, x_2), e) := \begin{cases} (f_1(x_1, e), f_2(x_2, e)), & \text{se } e \in \Gamma_1(x_1) \cap \Gamma_2(x_2) \\ (f_1(x_1, e), x_2), & \text{se } e \in \Gamma_1(x_1) \setminus E_2 \\ (x_1, f_2(x_2, e)), & \text{se } e \in \Gamma_2(x_2) \setminus E_1 \\ \text{indefinido}, & \text{casos restantes} \end{cases}$
- $\Gamma_{1 \parallel 2}(x_1, x_2) = [\Gamma_1(x_1) \cap \Gamma_2(x_2)] \cup [\Gamma_1(x_1) \setminus E_2] \cup [\Gamma_2(x_2) \setminus E_1]$

Portanto, as operações de composição produto e paralela são formas de conectar os componentes de dois sistemas representados pelos autômatos G_1 e G_2 . A primeira é denominada composição completamente síncrona, já que as transições entre os autômatos deve sempre ser sincronizada com um evento em comum. A composição produto entre os autômatos das figuras 2.1 e 2.2 é vista na figura 2.4.

Já a composição paralela, ou composição síncrona, é menos restritiva, pois inclui o conjunto de eventos *privados* de cada autômato que pertencem ao seu comportamento interno, além dos *eventos em comum* que são compartilhados entre ambos

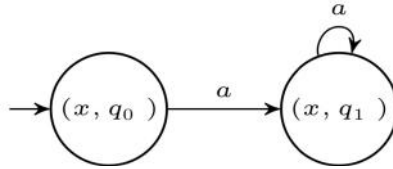


Figura 2.4: Composição produto entre os autômatos das figuras 2.1 e 2.2.

autômatos e representam o acoplamento entre os respectivos sistemas. Desta forma, esta operação é usada para modelar sistemas compostos de componentes que interagem entre si, e será usada no capítulo 4 para a modelagem do sistema. Um exemplo de composição paralela entre os autômatos das figuras 2.1 e 2.2 é vista na figura 2.5.

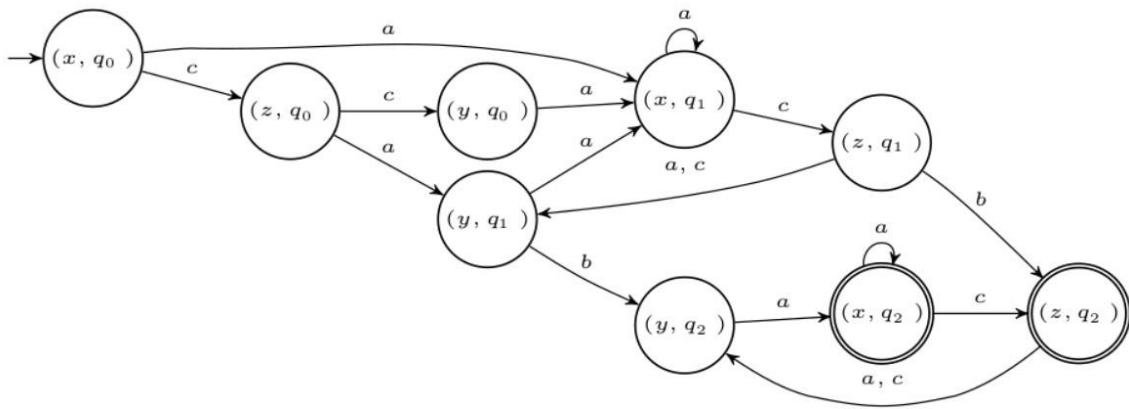


Figura 2.5: Composição paralela entre os autômatos das figuras 2.1 e 2.2.

2.3.2 Sistemas parcialmente observáveis

Um sistema a eventos discretos parcialmente observável é composto por eventos que não podem ser vistos por observadores externos ou eventos rotulados por transições ϵ . Podem ocorrer devido à ausência de sensores que gravem a ocorrência do evento, ou por eventos que representam falhas. Os sistemas parcialmente observáveis devem, portanto, ser modelados como autômatos não determinísticos G_{nd} (definição 2.12). Nesses casos, para analisar o sistema, é necessário realizar uma estimação de estados. O conjunto de eventos E pode ser particionado como $E = E_o \dot{\cup} E_{uo}$, em que E_o é o conjunto dos eventos observáveis e E_{uo} é o conjunto dos eventos não observáveis e, portanto, $E_o \cap E_{uo} = \emptyset$. Os eventos observáveis são aqueles cuja ocorrência pode ser registrada por sensores.

Um autômato não determinístico G_{nd} pode ser transformado em um autômato determinístico. O autômato determinístico equivalente, denominado observador (G_{obs}), traça a estimativa de estado do autômato não determinístico através de transições rotuladas por eventos em E . O espaço de estados do autômato observador será um subconjunto do conjunto potência 2^X de G_{nd} .

A construção do observador usa a extensão da definição de alcance não observável de um estado $x \in X$, denotado $UR(x)$, tal que $UR(x) = \{y \in X | (\forall t \in E_{uo}^*)(f(x, t) = y)\}$, de forma que o alcance não observável de um conjunto de estados $B \in 2^X$ é definido como

$$UR(B) = \bigcup_{x \in B} UR(x)$$

O algoritmo 2.1 mostra os passos para a transformação $Obs(G_{nd})$ de um autômato não determinístico G_{nd} para um determinístico G_{obs} .

Algoritmo 2.1 (*Construção do Observador*)

Seja $G_{nd} = (X, E \cup \{\epsilon\}, f_{nd}, x_0, X_m)$ um autômato não determinístico.

Para construir $Obs(G_{nd} = (X_{obs}, E, f_{obs}, \Gamma_{obs}, x_{0,obs}, X_{m,obs}))$, faça:

Passo 1: Defina $x_{0,obs} := UR(x_0)$ e faça $X_{obs} = \{x_{0,obs}\}$ e $\tilde{X}_{obs} = X_{obs}$

Passo 2: Faça $\hat{X}_{obs} \leftarrow \tilde{X}_{obs}$ e $\tilde{X}_{obs} \leftarrow \emptyset$

Passo 3: Para cada estado $B \in \hat{X}_{obs}$, faça $\Gamma_{obs}(B) = (\bigcup_{x \in B} \Gamma(x)) \cap E_o$ e para cada evento $e \in \Gamma_{obs}(B)$, defina $f_{obs}(B, e) := UR(\{x \in X | (\forall y \in B)(x = f_{nd}(y, e))\})$ e faça $\tilde{X}_{obs} \leftarrow \tilde{X}_{obs} \cup f_{obs}(B, e)$

Passo 4: $X_{obs} \leftarrow X_{obs} \cup \tilde{X}_{obs}$

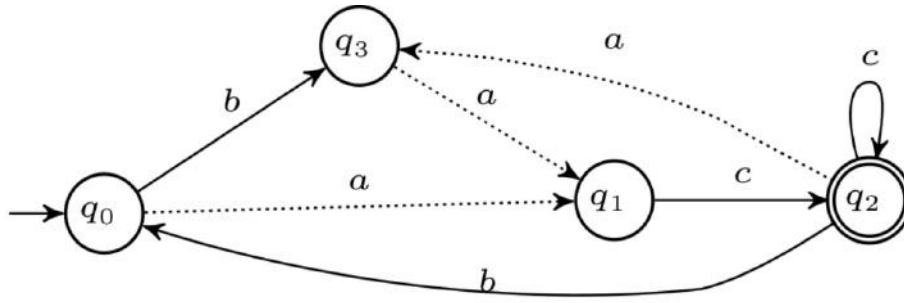
Passo 5: Repita os passos 2 a 4 até que seja construída toda $Ac(Obs(G))$

Passo 6: $X_{m,obs} = \{B \in X_{obs} | (B \cap X_m) \neq \emptyset\}$

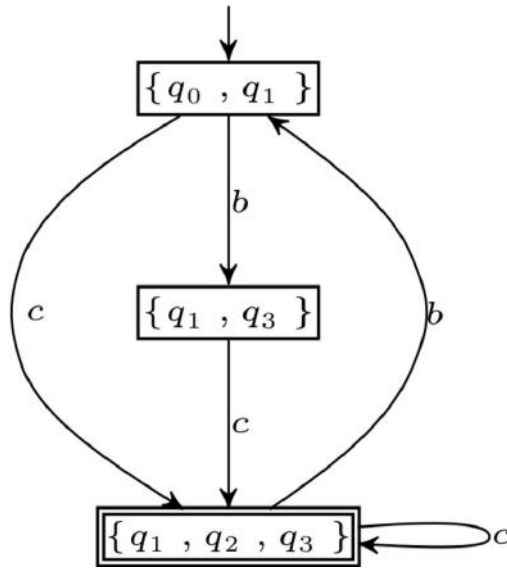
A linguagem gerada por $Obs(G_{nd})$ é a projeção da linguagem de G_{nd} sobre o conjunto de eventos observáveis, ou seja, $\mathcal{L}(Obs(G_{nd})) = P[\mathcal{L}(G_{nd})]$, com $P : E^* \rightarrow E_o^*$.

A figura 2.6b ilustra a obtenção do observador através do algoritmo 2.1 para o autômato parcialmente observável da figura 2.6a, tal que, as linhas tracejadas

representam transições com eventos não observáveis.



(a) Autômato parcialmente observável.



(b) Autômato observador equivalente.

Figura 2.6: Observador equivalente através do algoritmo 2.1.

2.3.3 Mapa de sensores

O *mapa de sensores* é um método sistemático e sofisticado para a diagnose de eventos não observáveis (falhas), proposto por [6] e se baseia na abordagem de sistemas a eventos discretos tanto para sistemas de natureza discreta por eventos, quanto para sistemas dinâmicos de variável contínua que possam ser considerados SED com adequada abstração.

Este método se baseia na leitura dos sensores serem modeladas no sistema como a saída de um estado através do autômato *Moore* (como foi feito na figura 2.3a).

Para a conversão de um autômato *Moore* em um autômato padrão, a informação sobre a saída de um estado é fornecida pelo o nome dos eventos entrando nele, desta

forma, é necessário renomear a transição do estado x para o estado y devido a um evento a com um novo evento chamado (a / saída do estado y). Se o evento a for observável, a interpretação do novo evento é: “evento a ocorre e é imediatamente seguido por uma saída igual à saída do estado y ”. Entretanto, esta conversão é inadequada se a for um evento não observável, já que, embora o rótulo do evento a seja não observável, por definição de saída do estado, a saída do estado y é observável. Assim, para modificar a conversão nesta situação, deve-se primeiro analisar se a saída do estado y posterior ao evento não observável a é diferente da saída do estado x que parte o evento. Neste caso, é criado um estado intermediário x_{xay} e define-se duas novas transições, uma primeira correspondente ao evento não observável a que parte do estado x ao x_{xay} , e uma transição observável do estado x_{xay} ao estado y rotulada como a variação nas saídas dos estados x e y na forma *saída do estado x , saída do estado y* . Desta forma, a mudança de estado do sistema seria capturada pelo modelo na forma de um evento observável.

O algoritmo para o mapa de sensores é explicado formalmente em [6].

2.3.4 Autômato diagnosticador

Lidar com incertezas na análise de sistemas significa que há alguma forma de não determinismo no modelo, capturadas na modelagem através do uso de eventos não observáveis. Para determinar se certos eventos não observáveis ocorreram em uma palavra de eventos executados pelo sistema, realiza-se a *diagnose de eventos*. Se os eventos não observáveis modelam a falha de componentes do sistema, então o conhecimento que um desses eventos já ocorreu é muito importante para monitorar o desempenho do sistema. Assim, ao realizar observações do comportamento do sistema, é possível reduzir a incerteza sobre o prefixo da palavra de eventos executada pelo sistema.

Como exemplo, pelo autômato da figura 2.6a, após observar a palavra $p = b$, não é possível definir se o sistema já executou o evento não observável a , enquanto após a observação da palavra $q = pc$, sabe-se com certeza que a já ocorreu, pois todas as

palavras de $\mathcal{L}(G)$ que são projetadas em bc contém a , ou seja, foi diagnosticado a ocorrência do evento não observável a após observar q .

O *autômato diagnosticador* é uma adaptação do autômato observador na forma de um procedimento de construção automática para a inferência de forma explícita dos eventos não observáveis a partir de sequências de eventos passados. Ou seja, o autômato diagnosticador G_{diag} , também denotado $Diag(G)$, rastreia o comportamento de um autômato G , e diagnostica, quando possível, as ocorrências prévias de determinados eventos não observáveis.

Apesar de similares, G_{diag} se diferencia de G_{obs} por adicionar rótulos aos *estados* de G , na forma N , se um evento não observável $e_d \in E_{uo}$ ainda não ocorreu, e Y , se já ocorreu. O rótulo será anexado ao estado $x \in X$ com a respectiva notação xN e xY , ou (x, N) e (x, Y) . Para múltiplos eventos não observáveis, sem perda de generalidade, podem ser construídos diagnosticadores para cada evento a ser diagnosticado, ou então construir um único diagnosticador que rastreia todos os eventos de interesse, comum em sistemas de diagnose com arquitetura centralizada.

O algoritmo de construção de G_{diag} é similar ao algoritmo 2.1 com as seguintes modificações:

- M1:** Na construção do alcance não observável do estado inicial x_0 :
- (i) Colocar o rótulo N nos estados que possam ser alcançados a partir de x_0 por palavras não observáveis em $[E_{uo} \setminus \{e_d\}]^*$;
 - (ii) Colocar o rótulo Y nos estados que possam ser alcançados a partir de x_0 por palavras não observáveis que contenham ao menos uma ocorrência de e_d ;
 - (iii) Se um estado z pode ser alcançado tanto pela execução do evento e_d quanto sem a execução, então criar as entradas zN e zY ao conjunto inicial de estados de G_{diag} .
- M2:** Na construção dos estados subsequentes de G_{diag} :
- (i) Seguir as regras para a função de transição de $Obs(G)$, com as modificações $M1$ para construção dos alcances não observáveis com estados rotulados;
 - (ii) Propagar o rótulo Y por todos os estados alcançáveis a partir do estado zY , de forma a indicar que o evento e_d ocorreu no processo de alcançar o estado z e, portanto, no processo de alcançar o novo estado a partir de z .
- M3:** Não será definido o conjunto de estados marcados para G_{diag}

Em suma, G_{diag} possui E_o como conjunto de eventos, isto é, é um autômato

determinístico e gera a linguagem $\mathcal{L}(G_{diag}) = P[\mathcal{L}(G)]$. Cada estado de G_{diag} é um subconjunto de $X \times \{N, Y\}$. E como consequência da modificação $M1(iii)$, não há necessariamente um mapeamento biunívoco entre os estados de G_{diag} e os estados de $Obs(G)$.

A figura 2.7 ilustra a construção do autômato diagnosticador para o autômato da figura 2.6a.

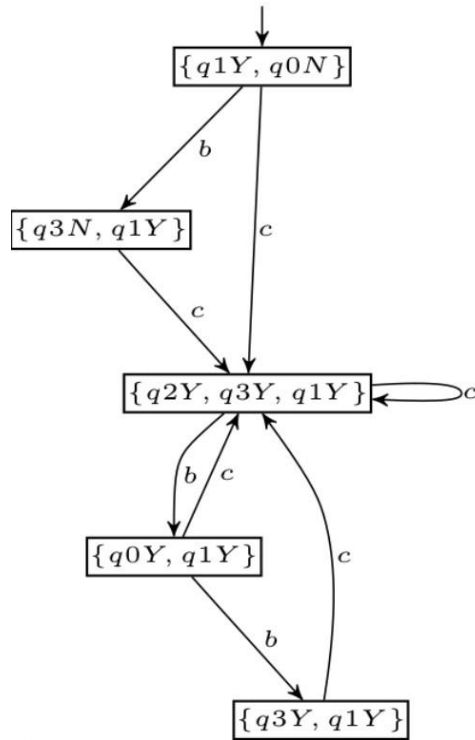


Figura 2.7: Diagnosticador para o autômato parcialmente observável da figura 2.6a.

Uma alternativa para a construção de G_{diag} explora as regras de propagação dos rótulos dos estados. O rótulo do estado inicial x_0 é sempre N . Uma vez que um rótulo N de um estado de G seja modificado para Y por uma subpalavra contendo e_d , o rótulo permanecerá Y para todos os futuros estados alcançáveis de G , ou seja, o evento e_d causa uma transição do rótulo de N para Y e, após isto, não há mais mudança no rótulo. Essa regra de propagação pode ser vista como um autômato rotulador, chamado de A_l para a diagnose do evento e_d conforme mostra a figura 2.8. Então, para obter G_{diag} , primeiro é feita a composição paralela entre G e A_l ,

para depois o autômato diagnosticador, na forma a seguir.

$$G_{diag} = Obs(G \parallel A_l) \quad (2.2)$$

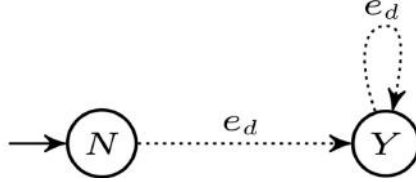


Figura 2.8: Autômato rotulador.

A composição paralela entre G e A_l não afeta a linguagem gerada por G , mas irá alterar o nome dos estados de G na forma $(x, label)$, tal que $label \in \{N, Y\}$ representa o rótulo, que será N ou Y dependendo, respectivamente, da ausência ou da presença do evento e_d nas palavras que alcançam x a partir de x_0 . A obtenção de $G \parallel A_l$ é suficiente para obter o observador e computar o diagnosticador.

A diagnose do evento não observável e_d é dada, portanto, da seguinte forma: se todos os estados de G no estado atual do diagnosticador contiverem o rótulo N , é o caso do *estado negativo* e é garantido que o evento não observável e_d não ocorreu. De forma análoga, se todos os estados de G no estado atual de G_{diag} contiverem o rótulo Y , é chamado de *estado positivo* e garantido que ocorreu e_d em algum ponto do passado. Por fim, se o estado atual de G_{diag} contiver ao menos um estado de G com rótulo Y e ao menos um com N , então é chamado de *estado incerto* e o evento e_d pode ou não já ter ocorrido no passado.

A diagnosticabilidade de um evento não observável e_d irá depender do sufixo da palavra após a ocorrência de e_d não percorrer um ciclo de estados incertos em G_{diag} . A definição formal de diagnosticabilidade para linguagens vivas é dada a seguir.

Definição 2.17 (*Diagnosticabilidade*)

Para a linguagem viva $\mathcal{L}(G)$, isto é, $\mathcal{L}(G) := (\forall s \in \mathcal{L}(G)(\exists e|se \in \mathcal{L}(G))$, um evento não observável e_d é dito não diagnosticável em $\mathcal{L}(G)$, se existem duas palavras s_N e s_Y na linguagem sendo que s_Y contém e_d então s_N não contém e_d com s_Y arbitrariamente longa após e_d tal que a projeção $P(s_N) = P(s_Y)$. Quando não existem palavras s_N e s_Y que satisfaçam as condições acima, então e_d é dito ser diagnosticável em $\mathcal{L}(G)$.

Portanto, a diagnosticabilidade é a propriedade que permite detectar e localizar a ocorrência de um evento de falha não observável após um número finito de eventos.

O diagnosticador G_{diag} obtido pode ser usado tanto para a *verificação* de diagnosticabilidade quanto para diagnosticar a falha de maneira *online*.

Para a verificação da diagnosticabilidade usando G_{diag} é necessária a definição de *ciclo indeterminado*. As definições para ciclos em um autômato e ciclo indeterminado de um conjunto de estados incertos são dadas a seguir [8].

Definição 2.18 (*Ciclo em um autômato*)

Seja $\mathcal{L}(G, x_1) = \{v \in E^* | (\exists uv \in \mathcal{L})(f(x_0, u) = x_1 \wedge uv \in \mathcal{L})\}$.

Um conjunto de estados $\{x_1, x_2, \dots, x_n\} \subseteq X$ forma um ciclo em um autômato G , se existir uma sequência $s = e_1 e_2 \dots e_n \in \mathcal{L}(G, x_1)$ tais que $f(x_l, e_l) = x_{l+1}$, $l = 1, \dots, n-1$, $f(x_n, e_n) = x_1$.

Definição 2.19 (*Ciclo indeterminado*)

Um conjunto de estados incertos $\{x_{d_1}, x_{d_2}, \dots, x_{d_p}\} \subset X_d$ forma um ciclo indeterminado se as seguintes condições forem satisfeitas:

1. $x_{d_1}, x_{d_2}, \dots, x_{d_p}$ forma um ciclo em G_d .
2. $\exists (x_l^{k_l}, Y), (\tilde{x}_l^{r_l}, N) \in x_{d_l}, x_l^{k_l}$ não necessariamente distinto de $\tilde{x}_l^{r_l}$, $l = 1, 2, \dots, p$, $k_l = 1, 2, \dots, m_l$, $r_l = 1, 2, \dots, \tilde{m}_l$ de tal sorte que as sequências de estados $\{x_l^{k_l}\}$, $l = 1, 2, \dots, p$, $k_l = 1, 2, \dots, m_l$ e $\{\tilde{x}_l^{r_l}\}$, $l = 1, 2, \dots, p$, $r_l = 1, 2, \dots, \tilde{m}_l$ podem ser rearranjadas para formar ciclos em G , cujas sequências correspondentes s e \tilde{s} , formadas com os eventos que definem a evolução dos ciclos, têm como projeção e_1, e_2, \dots, e_p , definida de acordo com a definição 2.18 para ciclos em autômatos.

Além disso, para determinar a natureza de um ciclo oculto, ou escondido, [8] em um autômato G com eventos não observáveis, é necessário buscar por ciclos de estados conectados com eventos não observáveis e o problema da busca de ciclos é possui complexidade pior do que exponencial no número de estados do autômato.

A definição de ciclos escondidos é dada a seguir.

Definição 2.20 (*Ciclos escondidos e ciclos escondidos indeterminados*).

Seja $x_d = \{x_1 l_1, x_2 l_2, \dots, x_n l_n\}$ um estado G_d . Então, existirá um ciclo escondido em x_d para algum $\{i_1, i_2, \dots, i_k\} \subseteq \{1, 2, \dots, n\}$, se as seguintes condições forem verdadeiras:

C1 $x_{i_1}, x_{i_2}, \dots, x_{i_k}$ formam um ciclo em G .

C2 $e_{i_1}, e_{i_2}, \dots, e_{i_k} \subseteq E_{uo}$,
tal que $f(x_{i_j}, e_{i_j}) = x_{i_{j+1}}$, $j = 1, 2, \dots, k-1$, $f(x_{i_k}, e_{i_k}) = x_{i_1}$.

C3 $l_{i_j} = Y$, $j = 1, 2, \dots, k$, então x_d tem um ciclo escondido indeterminado.

A partir dessas definições, é possível enunciar a seguinte condição necessária e suficiente para a diagnosticabilidade de uma linguagem.

Teorema 2.1 *Uma linguagem \mathcal{L} gerada por autômato G será diagnosticável em relação a uma projeção P e a um conjunto $E_f = \{e_d\}$, tal que $E_f \subseteq E_{uo}$, se e somente se, o seu diagnosticador G_{diag} não tiver ciclos indeterminados (observados e escondidos).*

Contudo, como a verificação de diagnosticabilidade de uma linguagem com G_{diag} necessita da busca por ciclos que, no pior caso, é de complexidade pior do que exponencial [9], será utilizado o autômato diagnosticador G_{scc} que executa um algoritmo de verificação através da busca de componentes fortemente conexas, e será descrito na seção a seguir.

2.3.5 Diagnosticador *Viana-Basilio*

Uma nova condição necessária e suficiente para a verificação da diagnosticabilidade foi realizada em [8, 9], definida pelo teorema 2.2 a seguir.

Teorema 2.2 [8, 9] *Seja $G_{scc} = G_{diag} \parallel G_l$, uma linguagem L , gerada por um autômato G , será diagnosticável em relação à projeção natural P e a um conjunto $E_f = \{e_d\}$, tal que $E_f \subseteq E_{uo}$, se, e somente se, não existirem componentes fortemente conexas formadas por estados $(x_d; x_l)$, tais que x_d seja incerto e x_l seja certo.*

Mais detalhes podem ser vistos em [9].

2.4 Sistema de diagnose de falhas

Uma falha pode ser definida como um desvio não desejado do comportamento normal ou pretendido de um sistema ou de seus componentes. Tal desvio pode ser considerado tolerável ou, caso ocasione avaria ou pane, denominado crítico.

O objetivo da diagnose de falhas consiste em *detectar, isolar e identificar* a falha. Enquanto a detecção decide se a falha já ocorreu ou se o sistema está em condições normais, o isolamento localiza os componentes do sistema causadores da falha e a identificação revela os aspectos da natureza da falha, como tamanho, importância e criticidade. Para os três aspectos, é necessário o modelo do sistema em seu comportamento desejável, já para o isolamento e para a identificação das falhas, também deve ser modelado como o sistema responde a determinadas falhas. Neste trabalho, os três aspectos serão agrupados para a construção do *diagnosticador de falha*.

A diagnose de falha em através de sistemas a eventos discretos é fundamentada pelas premissas de que as falhas a serem diagnosticadas são eventos não observáveis, ou seja, não são registradas por sensores, e a ocorrência de falhas altera o comportamento do sistema, sem necessariamente levar o sistema a uma parada [10].

As falhas são consideradas como uma entrada adicional para a modelagem do sistema, e divididas conforme os modos de falha correspondentes aos tipos de falha que provocam o mesmo efeito em um componente do sistema.

Nos modelos em SEDs, as falhas podem ser consideradas permanentes, graduais ou intermitentes. Além disso, podem ser classificadas com respeito ao elemento do sistema que a originou, isto é, falhas causada por *sensores*, como na discrepâncias entre valor real e medido por um sensor, falhas por *atuadores*, como quando um atuador é preso e seus comandos se diferenciam da sua saída real, falhas na *planta*, como por obstruções da tubulação ou vazamentos, que alteram a dinâmica no sistema, e falhas de execução e implementação do *controlador*, devido a problemas de hardware ou software.

As quatro classes acima descritas podem ser apropriadamente modeladas através dos autômatos, uma vez que, com a linguagem gerada pelo autômato, é possível representar todas as execuções ou sequência de eventos do sistema no estado de operação normal e de falha. A falha é então modelada como um evento não-observável associado ao comportamento do sistema dirigido por eventos. Para o

propósito deste trabalho, é tratada a classe de falhas por atuador, e, para o sistema descrito no capítulo 3, será modelada a falha na válvula de controle, considerada permanente e de tipos *falha fechada* e *falha aberta*.

As definições e algoritmos tratados neste capítulo constituem a base formal para a diagnose de falhas e para a análise da diagnosticabilidade de sistemas a eventos discretos modelados por autômatos. Os métodos para a diagnose de falhas usados neste trabalho lidam com a estimação de estado e a identificação da falha com modelos *comportamentais* baseado em *eventos*.

O método *mapa de sensores* decide se uma falha ocorreu somente pela observação da sequência de eventos, e é classificada como diagnose baseado em eventos. Esse método poderia ser aplicado também ao caso de falhas intermitentes, já que considera a falha como a ocorrência de um evento não observável. Essa abordagem requer a inicialização de ambos diagnosticador e planta ao mesmo tempo, pois o diagnosticador toma decisão com base na sequência de eventos observada. Tal inicialização nem sempre é fácil de obter em sistemas reais e pode necessitar a adição de eventos na fase de modelagem.

Na modelagem do sistema tratada no capítulo 4, não será necessário atribuir um conjunto de estados marcados, pois é considerado o comportamento completo do sistema, isto é, com a linguagem prefixo-fechada.

O problema da diagnose de falhas é determinar quais eventos não observáveis, no caso, as falhas, explicariam uma determinada sequência de eventos observada. Para isto, pode ser construído o autômato determinístico *observador*, cujas as transições são devidas aos eventos observáveis do sistema e os estados estimativas do estados reais do sistema.

Ao associar rótulos aos estados estimados pelo autômato observador que possam ser alcançados, ou não, por um traço contendo o evento de falha, é construído, portanto, o autômato *diagnosticador*, que possibilita determinar a diagnosticabilidade de um sistema. Como a busca por componentes fortemente conexas é mais eficiente que por ciclos, será usado o método proposto pelo teorema 2.2 para construção do

autômato diagnosticador.

O diagnosticador abordado na seção 2.3.4 e na equação 2.2 pode ser usado tanto na diagnose *online*, quanto para a *verificação* da diagnosticabilidade. Entretanto, a verificação com esse autômato possui complexidade pior do que exponencial, já que realiza a busca por ciclos indeterminados no autômato. Desta forma, será usado o diagnosticador *Viana-Basilio* para a diagnose *offline*, uma vez que realiza a busca por componentes fortemente conexos, o que leva a possuir complexidade linear [11].

O autômato diagnosticador é baseado na compilação *offline* das trajetórias observadas, mas pode ser usado conectado ao sistema para diagnose *online* após a ocorrência dos eventos observáveis. De acordo com o rótulo do estado, será diagnosticado de forma *online* se o sistema está em um estado normal, em um estado de falha ou em um estado incerto.

Caso os modelos não satisfaçam as condições para diagnosticabilidade, poderão ser alterados com as seguintes soluções: adicionar ao sistema mais sensores, que podem ser irrelevantes à operação normal do sistema; introduzir sensores virtuais para analiticamente aumentar a quantidade de informação dada pelos sensores reais; ou uma solução ativa ao usar ações de controle apropriadas para garantir que um sistema sempre permaneça diagnosticável. A última solução, entretanto, pode limitar o comportamento aceitável do sistema sob o ponto de vista do controle do processo.

Os sistemas serão modelados através da composição paralela de autômatos que representam subsistemas do sistema global. Dois tipos distintos de falhas serão considerados, e um diagnosticador será construído para cada tipo.

Os métodos formais da teoria de sistemas a eventos discretos, portanto, serão aplicados no capítulo 4 para discretizar e analisar o sistema de natureza contínua da unidade de separação trifásica de processamento primário de petróleo descrita no capítulo 3.

2.5 Ferramenta computacional para eventos discretos

A ferramenta computacional utilizada neste trabalho para análise e síntese de algoritmos para SEDs é o programa de computação científica *DESLAB* [12]. Escrito em linguagem *Python*, o *DESLAB* fornece uma ferramenta unificada que integra o formalismo de autômatos, os algoritmos para grafos e os cálculos numéricos. Ela permite a definição de variáveis simbólicas do tipo *autômato*, com sintaxe facilitada para a abstração por teoria de eventos discretos, capazes de incorporar instruções para manipular, operar e visualizar os autômatos. As operações e as imagens dos autômatos deste trabalho foram realizadas com auxílio do software *DESLAB*. Em [13, 14] é fornecido detalhamento sobre a ferramenta *DESLAB*.

2.6 Conclusão do capítulo

Este capítulo ilustrou as ferramentas e o embasamento teórico para a construção de modelos e para a análise de sistemas na classe de eventos discretos. Foram apresentados os conceitos fundamentais da teoria, com as definições para linguagem e autômato, assim como as operações realizadas com autômatos, até concluir sobre a verificação da diagnosticabilidade de um modelo com eventos não observáveis e apresentar o sistema diagnosticador de falhas através de SED. O capítulo 3 irá mostrar o modelo de um sistema contínuo no tempo, cujo comportamento será modelado e analisado por eventos discretos no capítulo 4.

Capítulo 3

Modelagem e simulação do processo contínuo

Este capítulo objetiva detalhar o modelo e a simulação do processo que será usado como base para a análise do diagnosticador de falhas. O sistema de controle da unidade de separação trifásica do processamento primário de petróleo é a aplicação industrial escolhida para validar a eficácia da teoria de sistemas a eventos discretos (SED) para diagnose de falhas das válvulas de controle.

Este processo será descrito na seção 3.1 junto com as equações utilizadas para a modelagem. A seção 3.2 revela a estratégia de controle utilizada, suas equações e parâmetros de sintonia. Já na seção 3.3 é detalhada a simulação da planta de separação e das estratégias de controle baseadas neste modelo.

3.1 Modelo da unidade separação trifásica no processamento primário de petróleo

O processamento primário de petróleo nas plataformas *offshore* consiste na separação dos fluidos do poço. O produto de extração dos reservatórios é uma mistura de petróleo, gás e água do mar que precisa ser separado. Efetivamente, além de não possuir valor econômico, a água do mar contém impurezas sólidas, sais solúveis e

contaminantes que podem gerar corrosão ou formar depósitos inorgânicos nas tubulações de produção, de transporte e de refino do petróleo.

Chama-se de separador o vaso ou a série de vasos de pressão utilizados para separar esses fluidos, nomeado de separador bifásico quando são separados duas fases, como os componentes líquidos dos gasosos, ou trifásico quando são decompostas as fases da água, do óleo (ou outros hidrocarbonetos líquidos) e do gás. O separador funciona com o princípio da coalescência e da diferença de densidade das fases, o que permite estratificá-las em gás no topo, água no fundo e óleo no meio do vaso. Com um sistema de controle adequado, os separadores podem também funcionar como vaso pulmão e absorver flutuações da corrente de entrada.

No processo de separação trifásica, o gás separado passa por outros processos primários até ser comprimido e enviado ao gasoduto, o óleo é bombeado para um navio ou enviado ao oleoduto, enquanto a água é injetada no poço, armazenada em tanque e descarregada periodicamente por navios aliviadores ou descartada no mar. Por questões ambientais, o descarte da água no mar deve ter baixos teores de óleo [15]. Com o aumento na quantidade de água produzida em poços de águas profundas, além de óleos crescentemente pesados, equipamentos como hidrociclones podem ser adotados nesta unidade para remover óleo residual da fase aquosa obtida com o separador. O esquema de uma unidade de processamento primário de uma plataforma de petróleo é visto na figura 3.1.

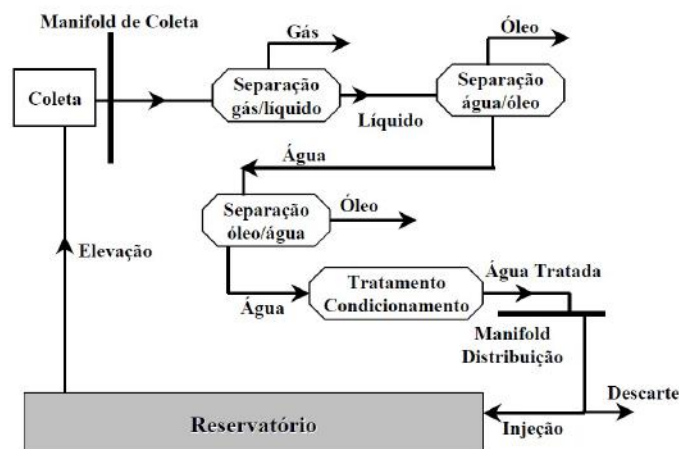


Figura 3.1: Diagrama de uma unidade de processamento primário de petróleo [2].

Uma modelagem matemática dinâmica de uma unidade de separação trifásica foi realizada por [16, 17] e adaptada por [18, 19], baseado nos dados de uma planta real da plataforma P-XX localizada no campo de Marlim na Bacia de Campos, pertencente à empresa PETROBRAS. O modelo fluidodinâmico do hidrociclone proposto por [20] também foi utilizado na modelagem. Já os controles propostos por [18, 19, 21] foram usados para a modelagem e simulação do modelo integrado do processo contínuo.

Nas seguintes subseções, são mostrados as equações da modelagem do separador e do hidrociclone que serão utilizadas de forma acoplada para a simulação deste trabalho.

3.1.1 Modelo do separador

Um esquema de um separador trifásico em plantas de processamento primário de petróleo é mostrado na figura 3.2. Apenas um vaso é considerado para o processo de separação das três fases com geometria cilíndrica horizontal, que é projetado para trabalhar a temperaturas em torno de 90°C e pressões próximas a $10\text{kgf}/\text{cm}^2$ e promove um padrão de fluxo adequado à segregação gravitacional [18]. Na entrada do vaso, uma placa defletora promove a coalescência para separação das fases líquida e gasosa e favorece a liberação do gás. Então, a fase líquida acumula-se na câmara de separação para separar por gravidade a fase aquosa da oleosa. O controle do nível h_w da interface água/óleo na câmara de separação atua sobre a vazão de saída da água W_{out} . O vertedouro favorece que o óleo migre para a câmara de separação para a câmara de óleo. O controle do nível h_o de óleo é realizado usando a vazão de saída de óleo O_{out} como variável manipulada. Já o controle de pressão do vaso atua sobre a vazão de saída do gás G_{out} . A figura 3.2 também mostra as variáveis de vazão de entrada de óleo, gás e água, respectivamente, O_{in} , G_{in} e W_{in} , além do nível total h_t da mistura líquida na câmara de separação.

Dispositivos internos para melhorar a eficiência da separação podem ser usados, como placas coalescedoras, agente antiespumante e *demisters*, mas, para efeitos da

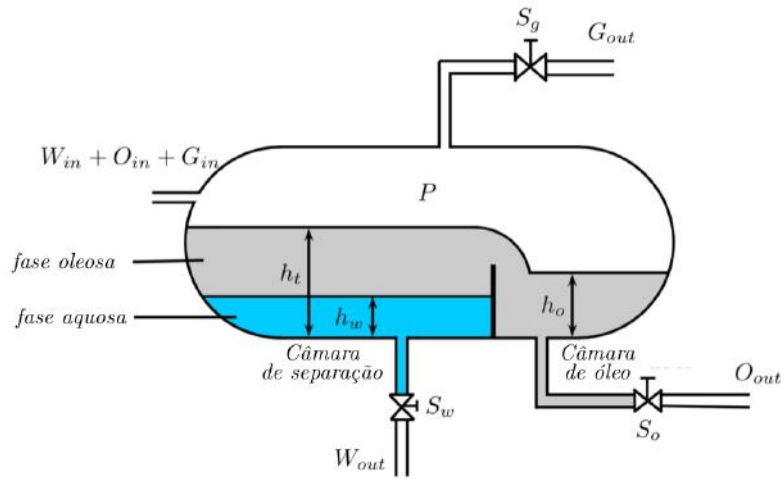


Figura 3.2: Vaso separador trifásico.

modelagem, não foram considerados os efeitos de agentes emulsificantes, nem a geração de espuma e nem o arraste de líquidos pela fase gasosa. Já o uso das placas paralelas horizontais, usadas para reduzir o espaço de escoamento das gotas de água e de óleo dispersas respectivamente nas fases oleosas e aquosas, foi modelado para determinar a eficiência de remoção da água da fase oleosa E_{wso} e do óleo da fase aquosa E_{osw} . Foi considerado um escoamento laminar nas placas e um perfil de velocidades parabólico.

A geometria do vaso horizontal cilíndrico de diâmetro D e comprimento C ocasiona uma relação não linear entre a altura do nível h e o volume acumulado de líquido V na forma $\frac{dV}{dt} = 2C\sqrt{(D-h)h}\frac{dh}{dt}$ que será usada a seguir para determinar as equações de estado do separador.

3.1.2 Equações de estado do separador

As equações de estado do separador, modeladas por [16, 17], para o nível e o volume foram obtidas a partir dos balanços de massa das fases dos fluidos no separador, enquanto a equação para a pressão é derivada a partir do balanço de energia do sistema. A descrição das variáveis de estado e as respectivas condições iniciais são dadas pela tabela 3.1. Já a descrição das variáveis de entrada e de saída do separador é mostrada na tabela 3.2. As seguintes equações podem ser definidas.

Varição do nível total da mistura líquida na câmara de separação h_t :

$$\frac{dh_t(t)}{dt} = \frac{W_{in}(t) + O_{in}(t) - O_{weir}(t) - W_{out}(t)}{2C_{sc}\sqrt{h_t(t)}(D - h_t(t))} \quad (3.1)$$

Varição do nível da fase oleosa na câmara de óleo h_o :

$$\frac{dh_o(t)}{dt} = \frac{O_{weir}(t) - O_{out}(t)}{2C_{oc}\sqrt{h_o(t)}(D - h_o(t))} \quad (3.2)$$

Varição do nível da fase aquosa na câmara de separação h_w :

$$\frac{dh_w(t)}{dt} = \frac{(1 - T_{og}E_{osw})W_{in}(t) - W_{out}(t) + B_{sw}E_{wso}O_{in}(t)}{2C_{sc}\sqrt{h_w(t)}(D - h_w(t))} \quad (3.3)$$

Varição do volume de água dispersa na fase oleosa na câmara de separação V_{wsc} :

$$\frac{dV_{wsc}(t)}{dt} = O_{in}(t)B_{sw}(1 - E_{osw}) - \frac{O_{weir}(t)V_{wsc}(t)}{V_{sc}(t) - V_{wps}(t)} \quad (3.4)$$

Varição do volume de óleo disperso na fase aquosa da câmara de separação V_{osc} :

$$\frac{dV_{osc}(t)}{dt} = W_{in}(t)T_{og}(1 - E_{wso}) - \frac{W_{out}(t)V_{osc}(t)}{V_{wps}(t)} \quad (3.5)$$

Varição do volume de água dispersa na fase oleosa na câmara de óleo V_{woc} :

$$\frac{dV_{woc}(t)}{dt} = \frac{O_{weir}(t)V_{wsc}(t)}{V_{sc}(t) - V_{wps}(t)} - \frac{O_{out}(t)V_{woc}(t)}{V_{oc}(t)} \quad (3.6)$$

Varição da pressão da fase gasosa p :

$$\frac{dp(t)}{dt} = \frac{W_{in}(t) + O_{in}(t) + G_{in}(t) - W_{out}(t) + O_{out}(t) + G_{out}(t)}{V_t - V_{sc}(t) - V_{oc}(t)} p(t) \quad (3.7)$$

Os valores e o significado dos parâmetros das equações 3.1 a 3.7 são dados pela tabela 3.3. As variáveis de processo V_{sc} , V_{oc} , V_{wps} , O_{weir} utilizadas nas equações de estado são descritas na tabela 3.4 e suas equações são explicitadas na seguinte subseção.

Tabela 3.1: Variáveis de estado do separador.

Variável	Descrição	Condição inicial
$h_t(t)$	Nível total da mistura líquida câmara de separação	0,9143 m
$h_o(t)$	Nível da fase oleosa na câmara de óleo	0,4976 m
$h_w(t)$	Nível da fase aquosa na câmara de separação	0,4931 m
$p(t)$	Pressão da fase gasosa no separador	9,480 kgf/cm ²
$V_{wsc}(t)$	Volume de água na fase oleosa na câmara de separação	0,0228 m ³
$V_{osc}(t)$	Volume de óleo na fase aquosa na câmara de separação	0,0002 m ³
$V_{woc}(t)$	Volume de água na fase oleosa na câmara de óleo	0,0040 m ³

Tabela 3.2: Variáveis de entrada e de saída do separador.

Variável	Descrição	Condição inicial
$W_{in}(t)$	Vazão de entrada de água	0,0133 m ³ /s
$O_{in}(t)$	Vazão de entrada de óleo	0,0167 m ³ /s
$G_{in}(t)$	Vazão de entrada de gás	0,1300 m ³ /s
$W_{out}(t)$	Vazão de saída de água da câmara de separação	0,0133 m ³ /s
$O_{out}(t)$	Vazão de saída de óleo da câmara de óleo	0,0167 m ³ /s
$G_{out}(t)$	Vazão de saída de gás do separador	0,1300 m ³ /s

Tabela 3.3: Parâmetros de modelagem do separador trifásico.

Parâmetros	Descrição	Valor modelado
C_{sc}	Comprimento da câmara de separação	4,4 m
C_{oc}	Comprimento da câmara de óleo	1,0 m
D	Diâmetro do vaso separador	1,8 m
V_t	Volume total do vaso separador	16,0315 m ³
B_{sw}	Teor de água e sedimentos na fase oleosa	0,100
T_{og}	Teor de óleo e graxa na fase aquosa	0,1300
E_{osw}	Eficiência de separação do óleo da fase aquosa	0,9994
E_{wso}	Eficiência de separação da água da fase oleosa	0,9292

3.1.3 Equações algébricas de processo

As equações algébricas de processo foram modeladas por [16, 17] de acordo com a geometria do vaso conforme as fórmulas abaixo.

$$V_{sc}(t) = \frac{C_{sc}D^2}{4} \left\{ \arccos\left(1 - \frac{2h_t(t)}{D}\right) - \sin\left[\arccos\left(1 - 2\frac{h_t(t)}{D}\right)\right] \cos\left[\arccos\left(1 - 2\frac{h_t(t)}{D}\right)\right] \right\} \quad (3.8)$$

$$V_{oc}(t) = \frac{C_{oc}D^2}{4} \left\{ \arccos\left(1 - 2\frac{h_o(t)}{D}\right) - \sin\left[\arccos\left(1 - 2\frac{h_o(t)}{D}\right)\right] \cos\left[\arccos\left(1 - 2\frac{h_o(t)}{D}\right)\right] \right\} \quad (3.9)$$

$$V_{wps}(t) = \frac{C_{sc}D^2}{4} \left\{ \arccos\left(1 - 2\frac{h_w(t)}{D}\right) - \sin\left[\arccos\left(1 - 2\frac{h_w(t)}{D}\right)\right] \cos\left[\arccos\left(1 - 2\frac{h_w(t)}{D}\right)\right] \right\} \quad (3.10)$$

$$O_{weir}(t) = \frac{24,88 \cdot \sqrt{2g}}{60} [C_{weir} - 0.2(h_t(t) - h_{weir})] \cdot (h_t(t) - h_{weir})^{1.5} \quad (3.11)$$

$$\rho_{op}(t) = \rho_o \left(1 - \frac{V_{wsc}(t)}{V_{sc}(t) - V_{wps}(t)} \right) + \rho_w \frac{V_{wsc}(t)}{V_{sc}(t) - V_{wps}(t)} \quad (3.12)$$

$$\rho_{wp}(t) = \rho_w \left(1 - \frac{V_{osc}(t)}{V_{wps}(t)} \right) + \rho_o \frac{V_{osc}(t)}{V_{wps}(t)} \quad (3.13)$$

O significado das variáveis de processo estão na tabela 3.4 e os parâmetros utilizados nas equações são descritos nas tabelas 3.3 e 3.5 junto aos seus valores modelados.

Tabela 3.4: Variáveis de processo.

Variável	Descrição
$V_{sc}(t)$	Volume da mistura líquida na câmara de separação
$V_{oc}(t)$	Volume da fase oleosa na câmara de óleo
$V_g(t)$	Volume de gás no separador
$V_{wps}(t)$	Volume da fase aquosa na câmara de separação
$\rho_{op}(t)$	Massa específica da fase oleosa
$\rho_{wp}(t)$	Massa específica da fase aquosa
$O_{weir}(t)$	Vazão de óleo sobre o vertedouro

Tabela 3.5: Parâmetros de modelagem das variáveis de processo.

Parâmetros	Descrição	Valor
h_{weir}	Altura da chicana	0,867 m
C_{weir}	Comprimento do vertedouro	0,9 m
ρ_w	Massa específica da fase aquosa	965 kg/m ³
ρ_o	Massa específica da fase oleosa	855 kg/m ³
g	Aceleração da gravidade	9,80665 m/s ²

3.1.4 Equações de saída

As variáveis de saída, conforme descritas na tabela 3.2 obedecem as seguintes equações de vazão calculadas por [16] através de [22].

$$W_{out}(t) = C_{wm}S_w(t) \frac{\sqrt{d_w(p(t) - p_{ds}) + 10^{-4} \cdot \gamma_w h_w(t) + 10^{-4} \cdot \gamma_o(h_t(t) - h_w(t))}}{0,0693 \cdot 60 \cdot \rho_{op}(t)} \quad (3.14)$$

$$O_{out}(t) = C_{om}S_o(t) \frac{\sqrt{d_o(p(t) - p_{ds}) + 10^{-4} \cdot \gamma_o h_o(t)}}{0,0693 \cdot 60 \cdot \rho_{op}(t)} \quad (3.15)$$

$$G_{out}(t) = C_{gm}S_g(t) \cdot \frac{R \cdot T}{2,832 \cdot 60 \cdot p(t) \cdot M_g} \cdot \sqrt{d_g(p(t) - p_{agv})(p(t) + p_{agv})} \quad (3.16)$$

As variáveis são descritas nas tabelas 3.1, 3.2 e 3.6 e os parâmetros utilizados possuem valores especificados nas tabelas 3.3 e 3.7.

Tabela 3.6: Variáveis de controle.

Variável	Descrição
$S_w(t)$	Abertura da válvula de água
$S_o(t)$	Abertura da válvula de óleo
$S_g(t)$	Abertura da válvula de gás

Tabela 3.7: Parâmetros para modelagem das vazões de saída do separador.

Parâmetros	Descrição	Valor
γ_w	Peso específico da água	965 kgf/m^3
γ_o	Peso específico do óleo	855 kgf/m^3
d_w	Densidade específica do água	0,965
d_o	Densidade específica do óleo	0,855
d_g	Densidade específica do gás	0,565
p_{ds}	Pressão a jusante das válvulas de óleo e de água	1 kgf/cm^2
p_{agv}	Pressão da unidade de compressão a jusante da válvula de gás	8,5 kgf/cm^2
M_g	Peso molecular do gás	16,48 g/mol
R	Constante universal dos gases	0,082 $\frac{atm \cdot l}{mol \cdot K}$
T	Temperatura da carga	360 K
C_{wm}	Coefficiente de vazão máximo da válvula de água	36,70
C_{om}	Coefficiente de vazão máximo da válvula de óleo	44,43
C_{gm}	Coefficiente de vazão máximo da válvula de gás	74,05

3.1.5 Modelo hidrodinâmico do hidrociclone para águas oleosas

O hidrociclone é um equipamento para recuperar hidrocarbonetos líquidos da corrente de água oleosa ao usar a queda de pressão do sistema para promover energia ou força motriz de forma a ocasionar a separação entre água e óleo. Uma bomba também pode ser usada para aumentar a pressão de alimentação.

Um exemplo de um hidrociclone estilo *liner* é mostrado na figura 3.3a e o esquema típico de um vaso de hidrociclone é mostrado na figura 3.3b.

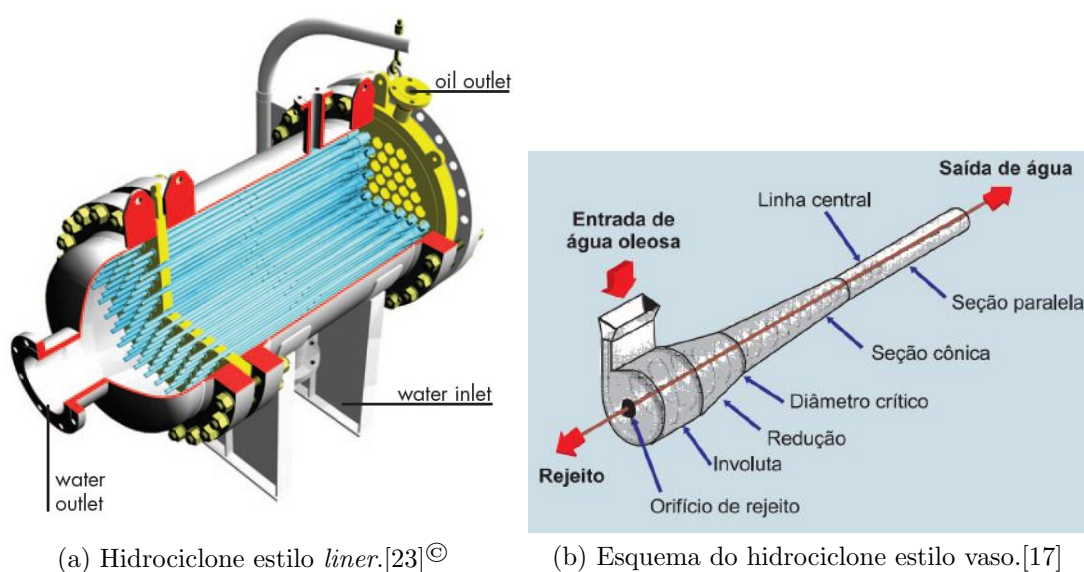


Figura 3.3: Tipos de hidrociclones.

Para atender as especificações de vazão, é possível utilizar o hidrociclone no estilo *liner*, quando vários vasos de hidrociclones são instalados de forma paralela no interior de um vaso de pressão, em número que varia de 1 a 400. Cada um dos vasos de hidrociclone possui uma alimentação tangencial de água, que é forçada a girar rapidamente, gerando altas forças centrífugas. Tais forças combinadas à forma cônica do perfil interno, ocasionam a aceleração radial. Isto força efetivamente a água para fora do eixo central em direção às paredes externas e força o óleo, de menor densidade, em direção ao núcleo central que se forma ao longo do hidrociclone. A água realiza um movimento em espiral pela seção cônica do ciclone e sai através do bocal de saída de água, enquanto o óleo no núcleo central é forçado na direção

reversa pela contrapressão da saída de água, e sai por um pequeno orifício na saída de rejeitos oleosos. A vazão do rejeito de óleo é controlada por este orifício e pode ser regulada por uma válvula de controle na saída que é configurada tipicamente para uma vazão entre 2 e 4% da vazão de entrada. [23] Como as forças gravitacionais geradas nos hidrociclones são muito elevadas, eles são adequados para serem instalados em estruturas móveis como plataformas de petróleo.

Dados operacionais obtidos por [17] aplicados ao modelo fenomenológico proposto por [20], sob as condições de modelagem monofásica do escoamento não viscoso, ausência de coalescência na fase dispersa e constantes de tempo do hidrociclone pequenas em relação às do vaso separador, permitiram a modelagem empírica do modelo estático do hidrociclone, com a obtenção da seguinte estimativa de eficiência do hidrociclone para águas oleosas:

$$\text{eficiência} = \frac{1}{a \cdot \exp^{b \cdot \text{split}}} \quad (3.17)$$

Onde:

$$a = 3619,6 - 1775039,5 \cdot W_{out} ,$$

$$b = 208,9 ,$$

W_{out} é a vazão de entrada no hidrociclone,

e *split* é a razão entre as vazões no *overflow* (topo) e *underflow* (fundo).

A equação 3.17 foi usada para obter o modelo dinâmico não linear para o hidrociclone e permitiu considerar a não dependência entre a pressão interna do equipamento e a eficiência [20]. Entretanto, o conhecimento das perdas de carga entre a linha de topo e de fundo do equipamento são fundamentais para o controle de vazão, conforme as equações 3.18 a 3.21.

$$\Delta P_o(t) = \alpha_1 W_o(t), \quad (3.18)$$

$$\Delta P_w(t) = \alpha_2 W_w(t) \quad (3.19)$$

$$W_w(t) = \frac{Cv_{m_w} S_w(t)}{\rho_{wp}(t) 4,158} \sqrt{d_w (P_{w_0} - \Delta P_w(t) - P_w)} \quad (3.20)$$

$$W_o(t) = \frac{Cv_{m_o} S_o(t)}{\rho_{op}(t) 4,158} \sqrt{d_o (P_{w_0} - \Delta P_o(t) - P_o)}, \quad (3.21)$$

As tabelas 3.6 e 3.8 descrevem as variáveis usadas, enquanto a tabela 3.9 apresenta o significado dos parâmetros utilizados. Já a figura 3.4 mostra um esquema de um módulo do hidrociclone.

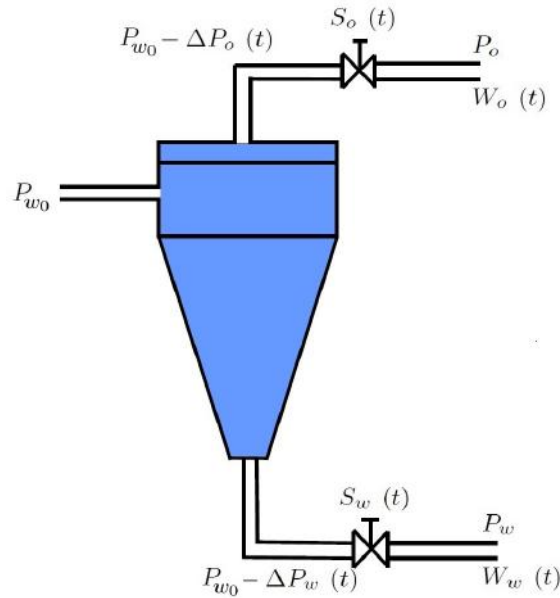


Figura 3.4: Modelo do hidrociclone.

Tabela 3.8: Variáveis da modelagem do hidrociclone.

Variável	Descrição
$W_o(t)$	Vazão de óleo na linha de topo do hidrociclone em $[m^3/s]$
$W_w(t)$	Vazão de água na linha de fundo do hidrociclone em $[m^3/s]$
$\Delta P_o(t)$	Diferencial de pressão entre alimentação e descarga no topo em $[kgf/cm^2]$
$\Delta P_w(t)$	Diferencial de pressão entre alimentação e descarga no fundo em $[kgf/cm^2]$

Tabela 3.9: Parâmetros do hidrociclone.

Parâmetro	Descrição
Cv_{m_o}	Coefficiente de descarga máximo da válvula de topo (adimensional)
Cv_{m_w}	Coefficiente de descarga máximo da válvula de fundo (adimensional)
α_1	Parâmetro de ajuste do modelo para perda de carga (adimensional)
α_2	Parâmetro de ajuste do modelo para perda de carga (adimensional)
P_{w_0}	Pressão na alimentação do hidrociclone em $[kgf/cm^2]$
P_o	Pressão na descarga da linha de topo em $[kgf/cm^2]$
P_w	Pressão na descarga da linha de fundo em $[kgf/cm^2]$

Os hidrociclones mais utilizados são do tipo DC (*De-oiler Cyclone*), que são projetados para tratar misturas com teores de óleo menores que 2000ppm. Uma evolução no tratamento de águas oleosas de poços petrolíferos foi introduzida na planta estudada pela empresa norueguesa *Kvaerner*. Consiste em novos tipos de hidrociclones usados em sequência. [21] Desta forma, a mistura líquida separada do gás passa inicialmente por um hidrociclone do tipo BOW (*Bulk Oil-Water Cyclone*), que diminui os teores de óleo de uma corrente de até 50% para no máximo 15%. O fluido passar então por um outro hidrociclone do tipo PDC (*Pre De-oiler Cyclone*), que trata correntes com teor de óleo de no máximo de 15% para a sair com teores da ordem de 1500ppm. Esta corrente poderia então passar pelo hidrociclone DC para se obter uma saída de água com teor de 200ppm de óleo. [17]. Além disso, essa tecnologia permite reduzir o tamanho das plataformas de exploração de petróleo.

O esquema da bateria de hidrociclones é mostrado na figura 3.5. A partir das equações para um único hidrociclone, obtém-se, através do balanço de massa e de energia, os cálculos para cada um dos hidrociclones da bateria conforme equações a seguir.

As variáveis S_{o_i} , S_{w_i} , W_{o_i} , W_{w_i} , ΔP_{o_i} , ΔP_{w_i} possuem o mesmo significado das tabelas 3.6 e 3.8, com $i = 1$ correspondendo ao hidrociclone BOW, $i = 2$ correspondendo ao PDC e $i = 3$ ao DC. Os valores modelados dos parâmetros e seus significados são dados pela tabela 3.10.

$$\begin{aligned} \Delta P_{o_i}(t) &= \alpha_1 W_{o_i}(t), \\ \text{para } i &= 1, 2, 3 \end{aligned} \quad (3.22)$$

$$\Delta P_{w_3}(t) = \alpha_2 W_{w_3}(t) \quad (3.23)$$

$$\begin{aligned} W_{o_i}(t) &= \frac{Cv_{m_o} S_{o_i}(t)}{\rho_{op}(t) 4,158} \sqrt{d_o(P_{w_{i-1}} - \Delta P_{o_i}(t) - P_{o_i})}, \\ \text{para } i &= 1, 2, 3 \end{aligned} \quad (3.24)$$

$$W_{w_1}(t) = W_{o_2}(t) + W_{w_2}(t) \quad (3.25)$$

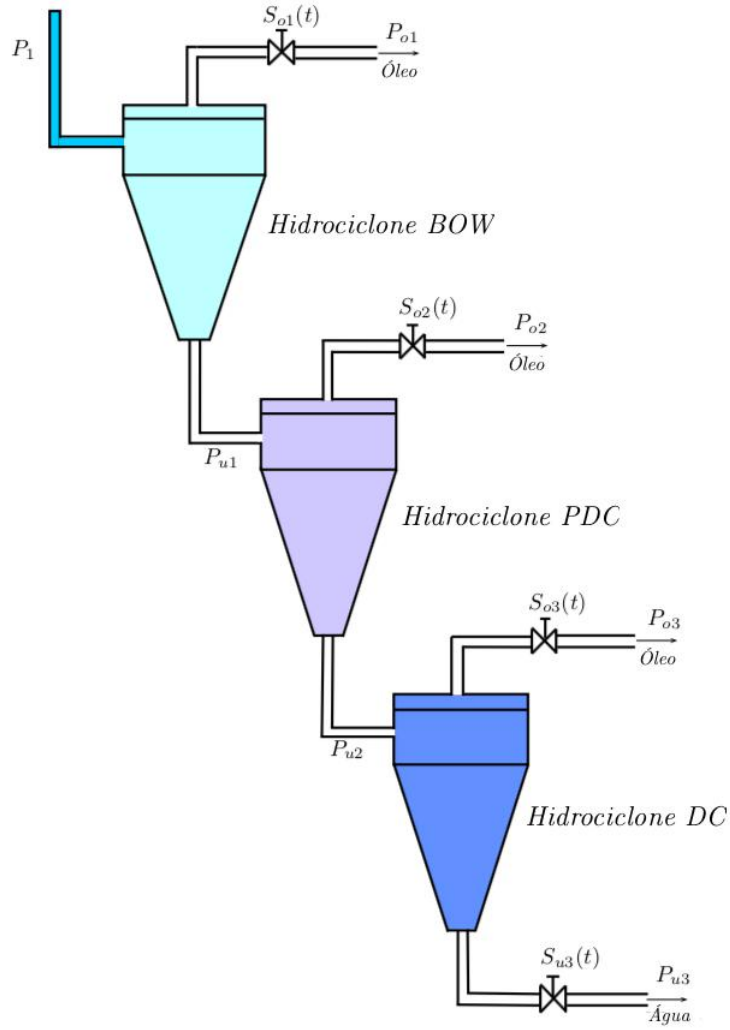


Figura 3.5: Modelo da bateria de três hidrociclones.

$$W_{w2}(t) = W_{o3}(t) + W_{w3}(t) \quad (3.26)$$

$$W_{w3}(t) = \frac{Cv_{m_w} S_{w3}(t)}{\rho_{wp}(t) 4,158} \sqrt{d_w (P_{w2} - \Delta P_{w3}(t) - P_{w3})} \quad (3.27)$$

$$P_{dr}(t) = \frac{\Delta P_{o3}(t)}{\Delta P_{w3}(t)} \quad (3.28)$$

A equação explícita para $P_{dr}(t)$ será relevante para o controle do nível de água na câmara de separação h_w no modelo integrado e será explicado na seção 3.2.

Tabela 3.10: Parâmetros da série de hidrociclones BOW, PDC e DC.

Parâmetro	Descrição	Valor
Cv_{m_o}	Coefficiente de descarga máximo da válvula de topo	1.85
Cv_{m_w}	Coefficiente de descarga máximo da válvula de fundo	45
α_1	Parâmetro de ajuste do modelo para perda de carga	7000
α_2	Parâmetro de ajuste do modelo para perda de carga	150
P_{w_0}	Pressão na alimentação do hidrociclone BOW	10 kgf/cm^2
P_{o_1}	Pressão na descarga da linha de óleo do BOW	7 kgf/cm^2
P_{w_1}	Pressão na descarga da linha de água do BOW	7 kgf/cm^2
P_{o_2}	Pressão na descarga da linha de óleo do PDC	5 kgf/cm^2
P_{w_2}	Pressão na descarga da linha de água do PDC	5 kgf/cm^2
P_{o_3}	Pressão na descarga da linha de óleo do DC	1 kgf/cm^2
P_{w_3}	Pressão na descarga da linha de água do DC	1 kgf/cm^2

3.1.6 Modelo integrado

O modelo integrado do sistema de separação trifásica consiste no sistema composto pelo separador integrado à bateria de hidrociclones, como é representado através da figura 3.6.

Para o modelo integrado, foram considerados os valores da tabela 3.2 correspondendo ao estado estacionário das variáveis de entrada e de saída quando as válvulas de controle estão abertas entre 60% a 70%, na região linear. Esse requisito equivale a presumir que o sistema de diagnose de falhas, na prática, somente começa a funcionar após o sistema estar em estado estacionário.

A vazão de saída de água da câmara de separação é dada pela soma das vazões de saída dos três hidrociclones, de acordo com a equação a seguir.

$$W_{out}(t) = W_{o_1}(t) + W_{o_2}(t) + W_{o_3}(t) + W_{w_3}(t). \quad (3.29)$$

3.2 Controle da unidade de separação trifásica

3.2.1 Estratégia de controle

O controle em malha fechada de processos industriais é um sistema que mantém de forma automática determinadas variáveis da planta em valores especificados de operação e, com isso, permite melhorias de produtividade e da qualidade da

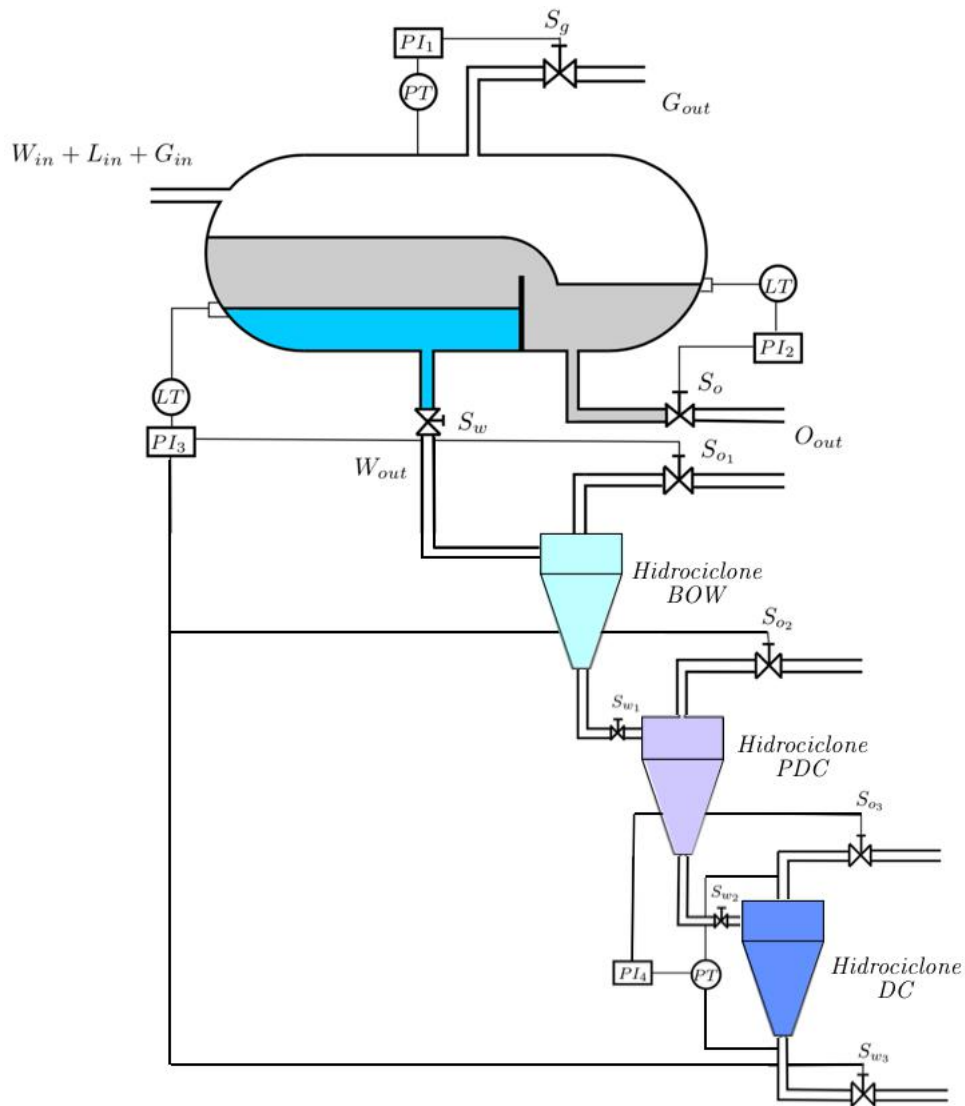


Figura 3.6: Fluxograma do separador trifásico integrado com bateria de hidrociclones e malhas de controle.

produção, aumento da confiabilidade dos sistemas e elevação do nível de segurança da unidade. O controle realimentado, ou *feedback*, é um controle em malha fechada que determina o desvio entre o valor desejado, ou *setpoint*, de uma variável com o seu valor medido no processo e atua manipulando a saída de forma a eliminar esse erro, ainda que hajam perturbações externas ou não linearidades no sistema. Com uma sintonia adequada, é possível garantir a estabilidade do sistema realimentado.

Para controles de vazão, de nível e de pressão é tipicamente aplicado na indústria o algoritmo do controlador proporcional e integral (PI) [24]. Este algoritmo apresenta simplicidade no ajuste dos parâmetros para um bom desempenho, além de

estar disponível em quase todos os equipamentos industriais de controle. A saída do PI é um sinal gerado ao calcular os termos proporcional e integral do erro entre a variável controlada do processo e o seu valor desejado. A fórmula para o algoritmo PI paralelo clássico é mostrada a seguir.

$$u(t) = K_P \cdot e(t) + K_P \cdot \frac{1}{\tau_I} \cdot \int e(t)dt + u_0 , \quad (3.30)$$

em que K_P é a constante proporcional, τ_I é a constante integral, $e(t)$ é o erro de rastreamento entre a variável controlada e o *setpoint*, o ganho integral do controlador é dado por $\frac{1}{\tau_I}$ e u_0 é a saída de estado estacionário (bias), ou seja, a posição da saída do controlador no instante que foi colocado em automático.

Ao utilizar controladores PI, pode ser necessário empregar mecanismos de *anti-reset windup*, isto é, para eliminar a saturação do termo integral. Uma forma é limitar a saída do controlador entre a faixa de operação de 0 a 100% para evitar a saturação da saída, além de ser necessário anular a contribuição integral. Um exemplo disto em controladores de posição é parar a execução do termo integral sempre que a saída alcançar os limites de 0 e 100%.

Com o uso de controladores digitais, é utilizado normalmente o controle PI com a implementação do algoritmo de velocidade, também chamado de algoritmo incremental. Ele é vantajoso pois não necessita inicializar, ou seja, ao passar o controle de manual para automático (*mode switch*), não requer o valor inicial u_0 da variável manipulada. Além disso, ele permite eliminar de forma simples o *reset windup* ao somar, no mesmo ciclo de execução, a variação do controlador com a posição atual da válvula, limitando a saída entre os extremos de 0 a 100%. Como a maioria dos cálculos são feitos somente com incrementos, o algoritmo de velocidade apresenta outro benefício de frequentemente usar palavras de tamanho curto para os cálculos computacionais.

Para implementar uma lei de controle PI de tempo contínuo em um computador digital, é necessário aproximar o fator integral do tempo contínuo. Existem diferentes métodos para a aproximação digital, dentre eles pela aproximação de Euler

(ou *forward differences*), por *backward differences* ou pela aproximação de Tustin (ou método trapezoidal) [25]. Assim, uma forma simples para derivar a fórmula no controle digital do algoritmo em velocidade do PI paralelo se dá ao substituir o termo integral por diferenças finitas na forma a seguir [26].

$$\int_0^t e(t^*)dt^* \approx \sum_{j=1}^k e_j \Delta t, \quad (3.31)$$

em que Δt é o período de amostragem, isto é, o tempo entre medições sucessivas da variável controlada, e e_k é o erro no k -ésimo tempo de amostragem para $k = 1, 2, \dots$

Substituindo a equação 3.31 na equação 3.30, tem-se a forma posicional do PI digital:

$$u_k = u_0 + K_P \left[e_k + \frac{\Delta t}{\tau_I} \sum_{j=1}^k e_j \right] \quad (3.32)$$

onde u_k é o controlador no k -ésimo tempo de amostragem.

Enquanto no algoritmo posicional é calculado o valor real da saída do controlador, no algoritmo de velocidade é a mudança na saída do controlador que é calculada. Sua fórmula é derivada ao escrever a equação 3.32 para o instante de amostragem $k - 1$ e obtendo a diferença com o instante de amostragem k .

$$u_{k-1} = u_0 + K_P \left[e_{k-1} + \frac{\Delta t}{\tau_I} \sum_{j=1}^{k-1} e_j \right] \quad (3.33)$$

O somatório acima começa em $j=1$, pois é suposto que o processo esteja no estado estacionário desejado para $j \leq 0$, logo, $e_j = 0$ para $j \leq 0$.

A implementação do algoritmo PI paralelo clássico na forma de velocidade é portanto dada pela equação a seguir.

$$\Delta u_k = u_k - u_{k-1} = K_P \left[e_k - e_{k-1} + \frac{\Delta t}{\tau_I} e_k \right] \quad (3.34)$$

Para obter a equação explícita para a saída u_k do controlador, basta rearranjar a equação 3.34:

$$u_k = u_{k-1} + K_P \left[e_k - e_{k-1} + \frac{\Delta t}{\tau_I} e_k \right] \quad (3.35)$$

A unidade de separação trifásica é um sistema multivariável, ou MIMO (*Multi-Input, Multi-Output*). O vaso separador apresenta três malhas de controle: uma malha para a pressão do vaso, uma malha para o nível da fase oleosa na câmara de óleo e uma para o nível da fase aquosa na câmara de separação. Através da modelagem da dinâmica e dos tempos de resposta do sistema do vaso separador sem a bateria de hidrociclones, foi detectado que as malhas do vaso separador possuem interação fraca e, portanto, a configuração de controle pode ser realizada de forma desacoplada, ou multimalha (*multi-loop*) [18]. Assim, foi determinada, para as três malhas do vaso separador, a estratégia de controle realimentado descentralizado com PI. No entanto, ao acrescentar a bateria de hidrociclones no modelo integrado, não convém usar *feedback* simples para a malha de controle do nível da água na câmara de separação, uma vez que o hidrociclone possui um tempo de residência pequeno quando sujeito às oscilações da carga [19]. Neste caso, em vez de se realizar o controle do nível da água, isto é, do acúmulo de massa de líquido, se realiza o controle da vazão, ou seja, da quantidade de massa por unidade de tempo, da corrente de saída de água da câmara de separação dada pela equação 3.29.

No controle de vazão, a constante de tempo do processo é de ordem de segundos e o tempo morto normalmente é desprezível. Neste caso, a resposta do sistema depende do tempo de resposta dos sensores e dos elementos finais de controle, do tamanho das tubulações, da amostragem do controlador e da transmissão de sinal. A estratégia de controle de razão é comumente usada na indústria para o controle de vazão e consiste no controle da razão entre duas vazões ao manipular uma delas. É possível implementar o controle de razão com uma *estação de razão*, isto é, utilizando dois PID, um para controlar a vazão de uma corrente e enviar um sinal para o segundo PID que controla a razão através da manipulação da segunda corrente. Outra forma é manipular a segunda corrente utilizando somente um PID, cuja variável controlada é dada pelo cálculo da divisão do valor das duas correntes

(razão). Embora esta implementação não seja tão flexível quanto a anterior e o cálculo da razão possa introduzir problemas de ruído e de sintonia, ela apresentou um bom desempenho nas simulações feitas por [21] para o modelo integrado.

A partir dessas considerações e das equações obtidas com o modelo hidrodinâmico do hidrociclone para águas oleosas, foi estabelecida uma estratégia de controle da vazão W_{out} através de uma malha controle *feedback* do nível h_w com algoritmo PI atuando nas válvulas S_{o1} , S_{o2} e S_{w3} em cascata com uma malha de controle *feedforward* que efetua o controle de razão com algoritmo PI manipulando a válvula S_{o3} de modo a manter no *setpoint* a variável P_{d3} , que representa a divisão entre as medições dos diferenciais de pressão correspondentes as vazões do topo e do fundo do último hidrociclone da bateria (*De-oiler Cyclone*).

Na figura 3.6 é possível ver o esquema da implementação das estratégias de controle para o modelo de separação trifásica, que será explicado a seguir.

Para a planta de separação trifásica, tem-se que a manipulação da abertura da válvula S_g na saída de gás controla a pressão no vaso separador. O controle do nível de óleo na câmara de óleo ocorre ao manipular a abertura da válvula de óleo S_o . Já o controle do nível de água da câmara de separação é realizado ao se manipular as quatro válvulas de controle S_{o1} , S_{o2} , S_{o3} , S_{w3} correspondentes às linhas de saída da bateria de hidrociclone, cuja soma das vazões corresponde à vazão de saída da água oleosa que sai da câmara de separação W_{out} (veja equação 3.29). Ou seja, S_g , S_o , S_{o1} , S_{o2} , S_{o3} , S_{w3} são as variáveis manipuladas e p , h_o , h_w são as variáveis controladas.

O nível da mistura na câmara de separação afeta diretamente a qualidade da separação trifásica. Um aumento do nível pode ocasionar arraste do óleo para a linha de gás, assim como arraste de água para a linha de óleo, devido ao aumento da pressurização do sistema, o que pode levar até ao alcance da pressão máxima de trabalho admissível (PMTA). Já a diminuição do nível poderia arrastar gás para a linha de óleo, ou óleo para linha de água. Com isso, é notável a importância dessa malha de controle.

Duas principais perturbações do sistema decorrem da queda de uma bomba de exportação e da variação brusca da vazão de produção. Caso uma bomba de exportação caia, o sistema pode perder a capacidade de escoar o óleo produzido. Para evitar uma parada geral da produção por nível alto no separador, pode-se diminuir manualmente a vazão de produção, ou acrescentar outro controlador PID de nível alto que atue nas válvulas *choke* cortando automaticamente a produção.

As golfadas no escoamento da produção é um fator relevante para ocasionar uma variação brusca na produção de óleo, o que afeta a qualidade da separação e pode ocasionar a parada emergencial da plataforma por nível muito alto no vaso separador. A não uniformidade da mistura multifásica escoada nas linhas de produção até a plataforma pode desencadear condições para um fluxo intermitente com bolsões de líquido seguido de ondas de produção de gás. As condições para a formação das golfadas podem ser devidas à diferença de velocidade entre as fases, pela geometria do terreno ou por cotovelos nos risers. Esses fatores levam ao acúmulo de líquido na base do *riser* que bloqueia o gás, de forma que a pressão na base aumenta e o líquido se acumula no *riser*. Quando a pressão na base é grande o suficiente para deslocar o líquido acumulado, este volume líquido é então bruscamente enviado à plataforma e o processo se repete ciclicamente.

Além disso, a variação na composição do petróleo e na vazão de produção dos poços devido à maturação do reservatório pode tornar inadequadas as dimensões do vaso e os valores de *setpoint* que garantem uma adequada separação da mistura por toda a vida útil da plataforma. Em face disto, para garantir a qualidade da separação, faz-se necessária uma boa estratégia de controle do processo e uma análise de fatores como a composição do petróleo do reservatório e o dimensionamento de equipamentos, como vasos e válvulas.

3.2.2 Modelo do sistema de controle da unidade de separação

A partir das considerações anteriores sobre o modelo do processo de separação trifásica com bateria de hidrociclones, foi escolhida uma estratégia de controle des-

centralizado com 4 malhas, sendo três malhas de controle realimentado e uma malha de controle de razão, todas com a utilização do algoritmo de controle digital PI na forma de velocidade. Ao discretizar a equação 3.30 para o PI paralelo clássico com tempo de amostragem T_s e com a aproximação discreta do termo integral feita através do método de Euler (também chamado de *forward differences*) [25], isto é,

$$I(t) = \frac{K_P}{\tau_I} \int_0^t e(t^*) dt^* \therefore I(k+1) = I(k) + \frac{K_P T_s}{\tau_I} e(k) \quad (3.36)$$

é determinada a equação para a saída dos controladores para cada instante de amostragem k como:

$$u_i(k) = u_i(k-1) + K_{c_i} \cdot e_i(k) + K_{c_i} \left(\frac{T_s}{\tau_i} - 1 \right) \cdot e_i(k-1) \quad (3.37)$$

para $i = 1, 2, 3, 4$

Onde K_{c_i} é o ganho proporcional para malha i , τ_i é a constante de tempo integral da malha i e u_i e e_i são, respectivamente, a saída do controlador e o sinal de erro da malha i dados pelas equações abaixo.

$$\begin{aligned} u_1(t) &= S_g(t), \\ u_2(t) &= S_o(t), \\ u_3(t) &= S_{o_1}(t) = S_{o_2}(t) = S_{u_3}(t), \\ u_4(t) &= S_{o_3}(t) \end{aligned} \quad (3.38)$$

$$\begin{aligned} e_1(t) &= p_{sp} - p(t) \\ e_2(t) &= h_{o,sp} - h_o(t) \\ e_3(t) &= h_{w,sp} - h_w(t) \\ e_4(t) &= P_{dr,sp} - P_{dr}(t) \end{aligned}$$

Os parâmetros p_{sp} , $h_{o,sp}$, $h_{w,sp}$ correspondem, respectivamente, aos valores de *setpoint* das variáveis p , h_o , h_w , descritas na tabela 3.1. O parâmetro $P_{dr,sp}$ corresponde ao valor desejado da razão de diferenciais de pressão no hidrociclone DC $P_{dr}(t)$, explicitada na equação 3.28. Os valores de *setpoint* obtidos na modelagem

se encontram na tabela 3.11 e as condições iniciais das variáveis manipuladas se encontram na tabela 3.12. As válvulas não controladas das correntes de água dos hidrociclones intermediários BOW e PDC serão simuladas totalmente abertas.

Tabela 3.11: Valores de *setpoint* das malhas de controle.

Parâmetro	Setpoint
$h_{o,sp}$	0,4976 m
$h_{w,sp}$	0,4931 m
p_{sp}	9,4806 kgf/cm ²
$P_{dr,sp}$	1,25

Tabela 3.12: Variáveis manipuladas do sistema de controle do separador.

Variável	Descrição	Condição Inicial ([0,1])
$S_o(t)$	Abertura da válvula de óleo	0,5
$S_{o_1}(t)$	Abertura da válvula de óleo do BOW	0,5
$S_{o_2}(t)$	Abertura da válvula de óleo do PDC	0,5
$S_{o_3}(t)$	Abertura da válvula de óleo do DC	0,5
$S_{w_3}(t)$	Abertura da válvula de água do DC	0,5
$S_g(t)$	Abertura da válvula de gás	0,5

A partir da curva de reação da modelagem do processo e ajuste através dos parâmetros pelo método de sintonia do *lugar das raízes* e pela análise do comportamento do sistema simulado [18, 19], os parâmetros dos quatro controladores são dados pela tabela 3.13.

Tabela 3.13: Parâmetros dos controladores PI.

Controlador	k_{c_i}	τ_i
PI_1	-0,096	14,6667
PI_2	-0,5156	366,228
PI_3	-2,7749	366,228
PI_4	0,1	8

3.3 Simulação do processo

Para a simulação do processo, foram usados os parâmetros que [16, 18] obtiveram dos dados da planta da plataforma P-XX da empresa Petrobras. Os valores dos parâmetros das dimensões do separador (C_{sc} , C_{oc} , D , V_t , h_{weir} , C_{weir}) são dados

pela tabelas 3.3 e 3.5. Os valores para a eficiência de separação do óleo da fase aquosa E_{osw} e da água da fase oleosa E_{wso} conforme a tabela 3.5 foram obtidos a partir da geometria das placas paralelas, cujo comprimento é $4,2m$, a largura vale $1,4m$ e a distância entre as placas é de $1,7mm$. Os parâmetros físico-químicos ($\rho_w, \rho_o, g, \gamma_w, \gamma_o, d_w, d_o, d_g, p_{ds}, p_{agv}, M_g, R, T$) utiliza os dados das tabelas 3.5 e 3.7.

Já os coeficientes de vazão (C_{wm}, C_{om}, C_{gm}) das válvulas de controle usados na simulação são dados pela tabela 3.7 e a abertura nessas válvulas ($S_l(t), S_w(t), S_g(t)$) foram consideradas inicialmente em 50%.

As condições de alimentação (B_{sw}, T_{og}) foram simuladas conforme os valores das tabelas 3.3, as condições iniciais das variáveis de entrada ($W_{in}(t), O_{in}(t), G_{in}(t)$) são dadas conforme tabela 3.2 e todas as condições iniciais são usadas para $t = 0$. O ambiente de simulação dinâmica escolhido foi o software *Simulink*. Um arquivo “parameters.m” com os dados dos parâmetros do modelo, das condições iniciais e dos valores de *setpoint* é usado pelo programa *Simulink* na simulação.

Foram criados blocos para a simulação correspondente às equações dinâmicas do processo de separação trifásica com hidrociclones de acordo com as seções 3.1 e 3.2 e as seguintes considerações foram feitas. Para as equações de estado, foi necessário limitar o valor do sinal $h_t(t)$ para positivo ($0,000001$) e menor ou igual ao diâmetro D do vaso separador de forma a manter o significado físico. Para os sinais $h_w(t)$ e $h_o(t)$ também foi necessário um bloco saturador que limita a saída entre menor ou igual à altura da chicana h_{weir} e maior do que zero. Os parâmetros com as condições iniciais $h_t(0), h_w(0), h_o(0)$ foram definidos no bloco integrador.

O sinal da pressão $p(t)$ também foi simulado para operar dentro dos limites inferior maior do que zero e limite superior dado pelo valor da PMTA, atribuído como $15kgf/cm^2$.

Os sinais $V_{wsc}(t), V_{osc}(t), V_{woc}(t)$ também contaram com um bloco de saturação de modo a manter o limite inferior acima de zero e manter o significado físico e foram usadas as condições inicial $V_{wsc}(0), V_{osc}(0), V_{woc}(0)$ no bloco integrador.

Para a equação de O_{weir} foi colocada uma restrição de saída nula caso $h_t < h_{weir}$.

Para as simulação das vazões do hidrociclone, foi necessário achar a forma explícita para W_{o_i} , $i = 1, 2, 3$, a partir das equações 3.22 e 3.24, e para W_{w_3} a partir das equações 3.23 e 3.27, tomando, em cada caso, a raiz quadrada positiva, de forma que o modelo seja fisicamente coerente. Desta forma:

Seja $A(t) = \frac{Cv_{m_o}S_{o_i}(t)}{\rho_{op}(t)4,158}$, $B(t) = \alpha_1 d_o(A(t))^2$ e $C(t) = (P_{w_{i-1}} - P_{o_i})d_0(A(t))^2$ então:

$$W_{o_i}(t) = \frac{-B(t) + \sqrt{(B(t))^2 + 4C(t)}}{2}, \text{ para } i = 1, 2, 3 \quad (3.39)$$

Similarmente, tem-se que:

$$W_{w_3}(t) = \frac{-B(t) + \sqrt{(B(t))^2 + 4C(t)}}{2}, \quad (3.40)$$

Onde $A(t) = \frac{Cv_{m_w}S_{w_3}(t)}{\rho_{wp}(t)4,158}$, $B(t) = \alpha_2 d_w(A(t))^2$ e $C(t) = (P_{w_2} - P_{w_3})d_0(A(t))^2$.

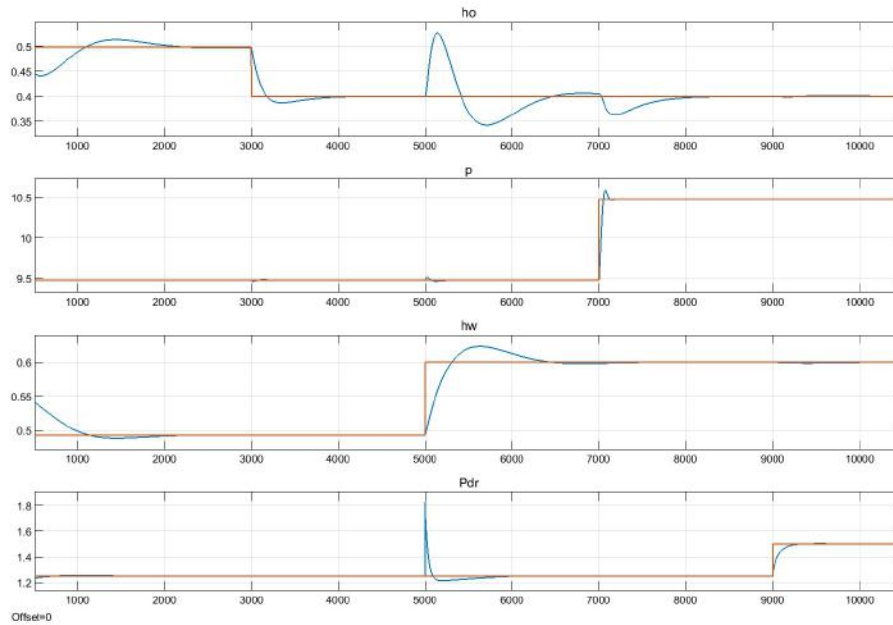
Para avaliar o modelo implementado no simulink, foram realizadas mudanças de *setpoint*(SP) nas variáveis controladas da forma conforme a tabela 3.14. O comportamento do sistema é mostrado na figura 3.7.

Tabela 3.14: Mudança de *setpoint* das malhas de controle.

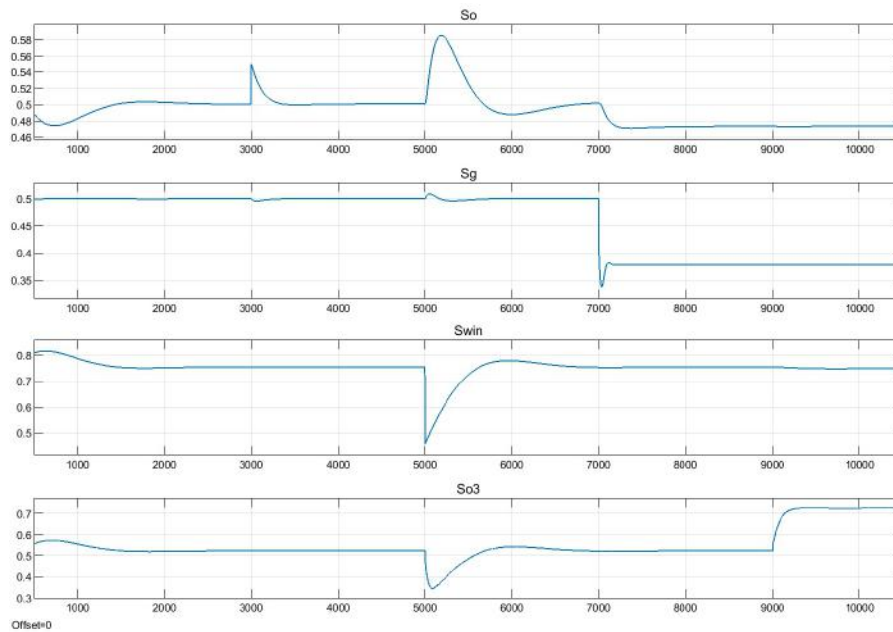
Parâmetro	Novo SP	Instante da mudança
$h_{o,sp}$	0,4 m	3000s
$h_{w,sp}$	0,6 m	5000s
p_{sp}	10,48 kgf/cm ²	7000s
$P_{dr,sp}$	1.5	9000s

O comportamento do sistema após a mudança é mostrado nas figuras 3.7a e 3.7b.

A figura 3.7a mostra que as malhas de controle modeladas foram capazes de levar as variáveis controladas do sistema aos novos valores de *setpoint*. Pela figura, também é possível observar a característica integradora das variáveis de nível de óleo da câmara de óleo, e de nível da água da câmara de separação. O desempenho do sistema também foi analisado após a mudança na carga de alimentação. O teste nas variáveis de entrada se deu conforme a tabela 3.15



(a) Variáveis controladas e *setpoint*

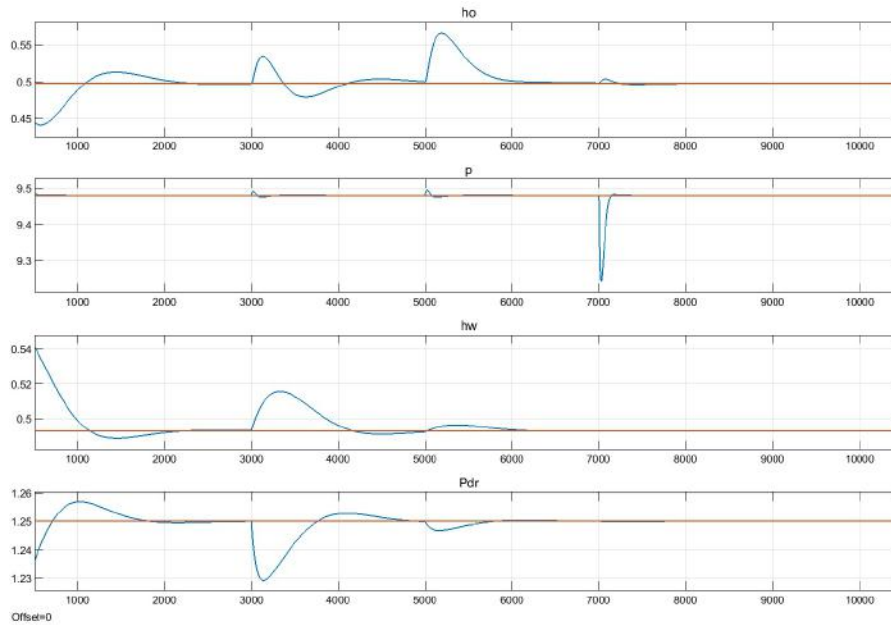


(b) Variáveis manipuladas

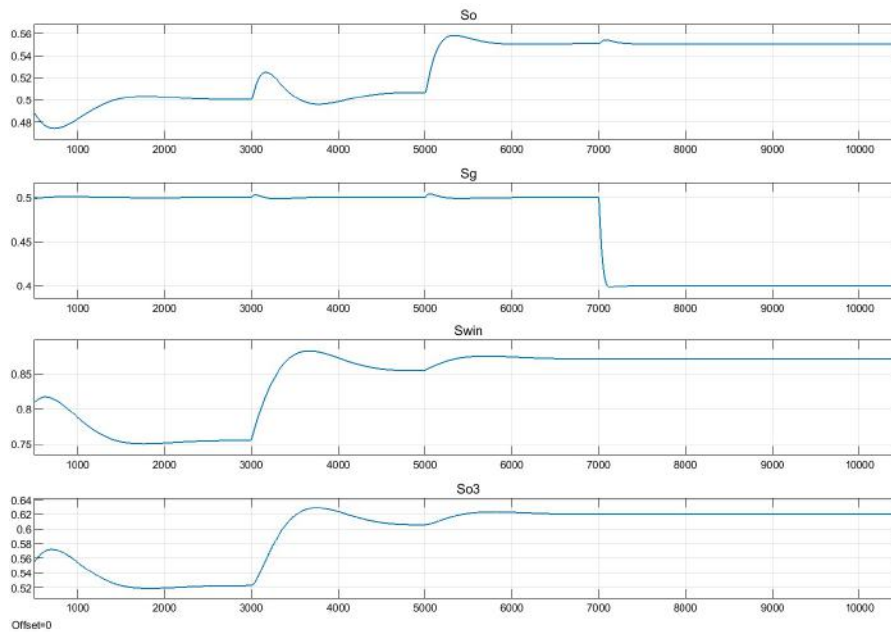
Figura 3.7: Resposta do sistema de separação em malha fechada após mudança nos *setpoints* de controle.

Tabela 3.15: Mudança nas variáveis de entrada.

Parâmetro	Porcentagem variada	Instante da mudança
W_{in}	aumento de 10%	3000s
O_{in}	aumento de 10%	5000s
G_{in}	diminuição de 20%	7000s



(a) Variáveis controladas e *setpoint*.

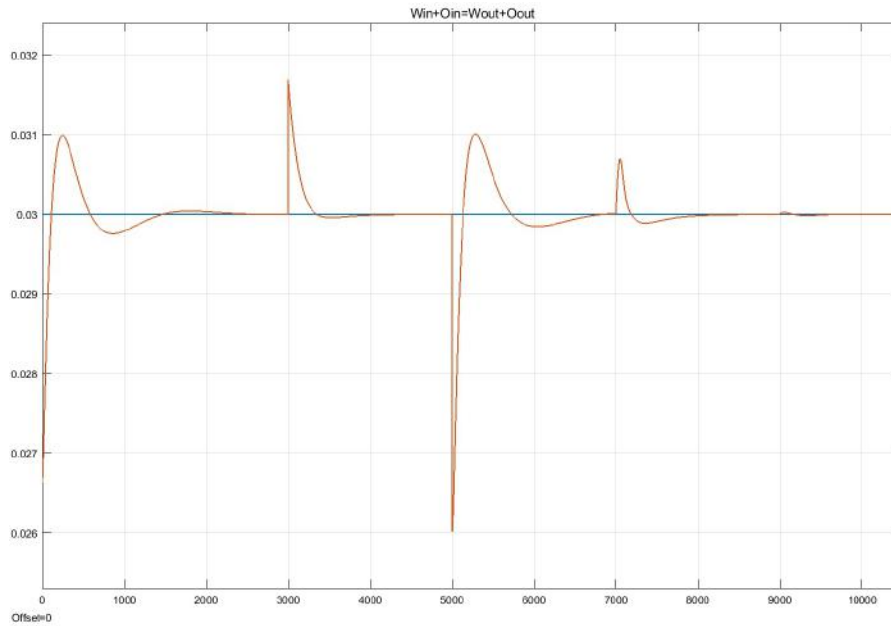


(b) Variáveis manipuladas.

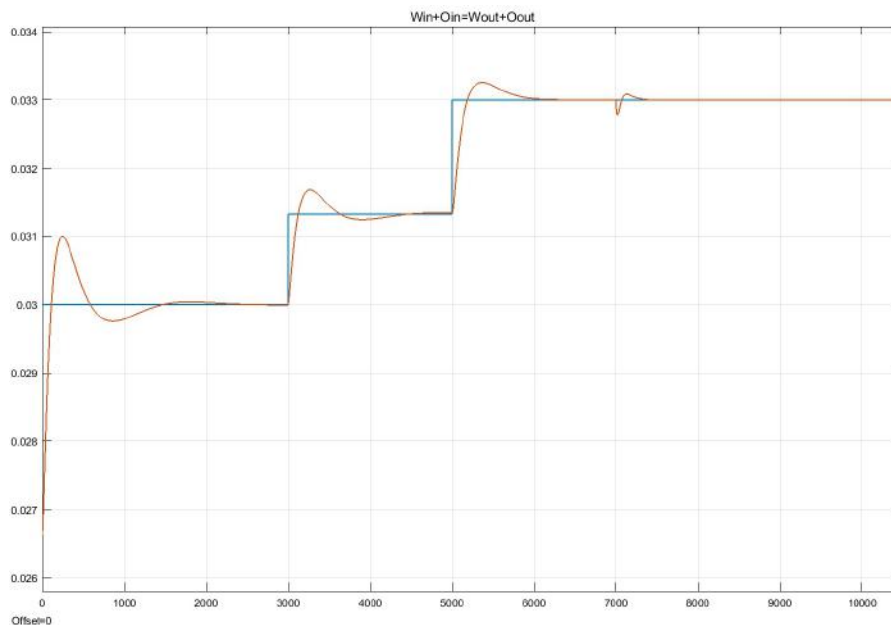
Figura 3.8: Resposta do sistema de separação em malha fechada após degrau nas variáveis de entrada.

A partir da figura 3.8a, é possível notar que, com o aumento na vazão de entrada de água, ambos os nível de água e de óleo tendem a aumentar, DC até que a ação de controle os leva de volta ao *setpoint*. Já o aumento na vazão de entrada de óleo, leva a um aumento discreto ao nível de água da câmara de separação. A diminuição na pressão de gás do separador é rapidamente corrigida pela ação controle e afeta

levemente o nível de óleo da câmara de óleo. Um fator para analisar se o modelo e a simulação foram adequadamente implementada é analisar o balanço de massa dos líquidos. A figura 3.9 mostra o balanço de massa para as duas perturbações tratadas acima. Pode-se notar que o sistema é bem equilibrado.



(a) Balanço de massa após mudança de *setpoint*.



(b) Balanço de massa após mudança nas correntes de entrada.

Figura 3.9: Balanço de massa de líquidos do sistema de separação simulado.

3.4 Conclusão do capítulo

Neste capítulo foi validado, através das referências bibliográficas e do comportamento na simulação, o modelo para o processo de separação trifásica de uma unidade de processamento primário de petróleo, que, apesar de ser um sistema contínuo na natureza, será discretizado e analisado como um SED no próximo capítulo.

Capítulo 4

Modelo a eventos discretos

Neste capítulo, o modelo do processo contínuo é avaliado sob a ótica de um sistema a eventos discretos (SED) na seção 4.1. Um modelo de diagnosticador de falha será proposto na seção 4.2 e sua simulação realizada e avaliada na seção 4.3.

4.1 Modelo do sistema a eventos discretos

Para realizar o diagnosticador de falhas, foi necessário modelar o processo contínuo como um SED. Desta forma, o diagnóstico é classificado como fundamentado pela construção do modelo comportamental, que deve conter informação suficiente do comportamento do sistema e de sua instrumentação.

Para a modelagem da planta com enfoque em diagnosticar a falha nas válvulas de controle, foi realizado um modelo a eventos discretos através de autômatos para os elementos do sistema baseado no trabalho de [19]. As operações com autômatos possibilitam que modelos completos sejam feitos partindo de componentes individuais. Desta forma, o modelo completo da planta é construído através da composição paralela de autômatos, conforme visto na definição 2.16.

Os componentes considerados para o modelo são: as válvulas de controle, o controlador, os sensores de pressão do vaso separador, de nível de água e de óleo e o sensor de diferencial de pressão do hidrociclone DC. Será então construído um autômato para a válvula e um para o controlador e os sensores serão modelados

intrinsecamente na forma de eventos.

Nas seções a seguir, é mostrado o modelo do componente da planta que responde a válvula de controle e do componente correspondente ao controlador de nível. O comportamento é adequado às válvulas $S_o, S_{o1}, S_{o2}, S_{o3}, S_{w3}$, cujo significado corresponde à descrição dada pela tabela 3.12. Para a válvula de pressão S_g , a ação de controle é similar, isto é, o comando de abrir a válvula ocorre quanto o valor da variável está acima do *setpoint*, e *vice versa*. Portanto, poderá ser aplicado o mesmo modelo das válvulas para correntes líquida e para a gasosa e, sem perda de generalidade, será descrito o modelo para uma válvula de controle de nível. Na implementação da simulação, todas as válvulas conterão um modelo discreto.

4.1.1 Modelagem da válvula de controle

Para a modelagem da válvula, foram considerados três estados para o comportamento normal do sistema: um para a válvula totalmente fechada (V_C), outro estado para a válvula totalmente aberta (V_O) e outro para a válvula parcialmente aberta (V_{POC}), dois estados para o comportamento de falha: um estado para a válvula travada aberta (V_{SO}) outro para a válvula travada fechada (V_{SC}) e, por fim, um estado inicial (V_I) para que o sistema seja capaz de alcançar quaisquer outros estados no caso de reinicialização.

Os eventos considerados foram os eventos observáveis correspondentes aos comandos de abrir ou de fechar a válvula em uma posição intermediária (*ocp*), de abrir a válvula totalmente (*ot*), isto é, correspondente a abertura de 100% e de fechar a válvula totalmente (*ct*), ou seja, 0% de abertura; os eventos não observáveis que correspondem à falha na válvula e ocasionam a transição da válvula para a posição de travamento aberta (*so*) ou de travamento fechada (*sc*) e, por fim, um evento observável *reset event* (*re*) capaz de levar o sistema ao estado inicial, para o caso de uma reinicialização.

As transições de estados para o modelo por autômato da válvula foram associadas de acordo com a tabela 4.1, que correspondem ao autômato da figura 4.1.

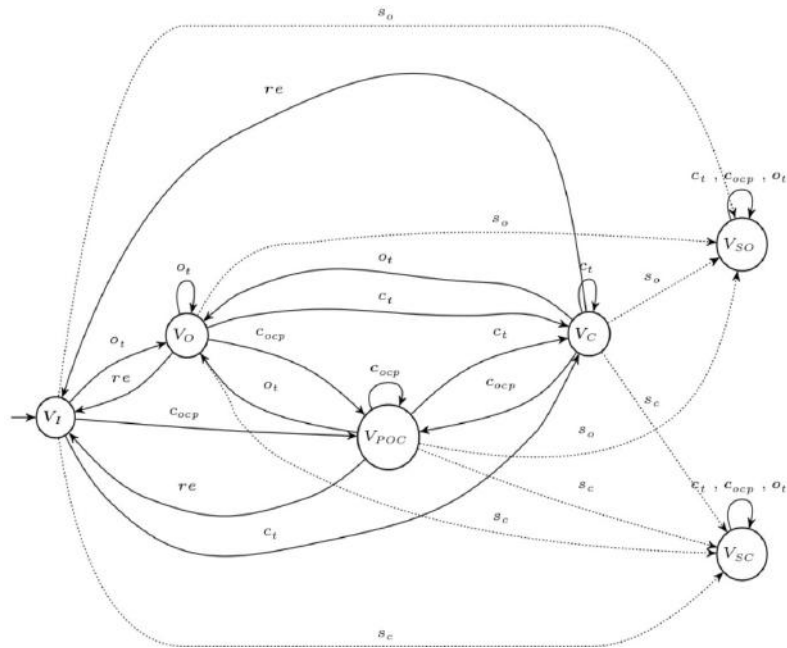


Figura 4.1: Autômato da válvula de controle.

Tabela 4.1: Tabela de transição de estados da válvula

		Eventos					
		ct	ocp	ot	re	sc	so
Estados	VI	VC	VPOC	VO	-	VSC	VSO
	VC	VC	VPOC	VO	VI	VSC	VSO
	VPOC	VC	VPOC	VO	VI	VSC	VSO
	VO	VC	VPOC	VO	VI	VSC	VSO
	VSC	VSC	VSC	VSC	-	-	-
	VSO	VSO	VSO	VSO	-	-	-

4.1.2 Modelagem do controlador

Para o modelo do controlador PI utilizado no processo, foi necessário avaliar a resposta do sistema à ação de controle. Através da simulação de perturbações no sistema, mostradas nas figuras 3.7 e 3.8, constatou-se a necessidade da introdução de sensores virtuais para uma modelagem adequada do autômato correspondente ao controlador.

Um sensor virtual é uma estimativa em tempo real de uma variável desejada a partir da medição de dados da planta. Desta forma, além dos eventos correspon-

dentes ao sensor de nível discretizados como nível acima do *setpoint*, nível abaixo do *setpoint* e nível no *setpoint*, foram construídos sensores virtuais que informam, em tempo real, a variação do nível, isto é, se está subindo, descendo, ou estável com relação à medição em um tempo de amostragem anterior.

O modelo discreto do controlador conta então com os estados: C_{SP} para o nível no *setpoint*, C_{DD} , para nível abaixo do *setpoint* e descendo, C_{DR} para o nível abaixo do *setpoint* e subindo, C_{UR} , para o nível acima do *setpoint* e subindo, C_{UD} para o nível acima do *setpoint* e descendo, além de C_I correspondendo ao estado inicial. O modelo para os controladores PI de pressão e de razão seguirão o mesmo modelo, para o valor da variável de pressão, ou da razão de pressão, acima, abaixo ou no *setpoint* e com os sensores virtuais indicando a variação ascendente ou descendente do valor da variável.

Os eventos para esse modelo são: *nsp* correspondente ao sensor de nível no *setpoint*, *ndd*, correspondente ao sensor de nível abaixo do *setpoint* e descendo, *ndr* para o nível abaixo do *setpoint* e subindo, *nur*, para o nível acima do *setpoint* e subindo, *nud* para o nível acima do *setpoint* e descendo, além de *re* correspondendo ao evento *reset event*, que modela a possibilidade de voltar ao estado inicial em caso de reinicialização do sistema.

As transições de estado para o autômato do controlador foram associadas de acordo com a tabela 4.2, que correspondem ao autômato da figura 4.2.

Tabela 4.2: Tabela de transição de estados do controlador

		Eventos								
		ct	ndd	ndr	nsp	nud	nur	ocp	ot	re
Estados	CI	-	CDD	CDR	CSP	CUD	CUR	-	-	-
	CDD	CDD	-	CDR	CSP	-	-	CDD	-	CI
	CUR	-	-	-	CSP	CUD	-	CUR	CUR	CI
	CUD	-	CDD	-	CSP	-	CUR	CUD	CUD	CI
	CDR	CDR	CDD	-	CSP	-	CUR	CDR	-	CI
	CSP	-	CDD	-	CSP	-	CUR	-	-	CI

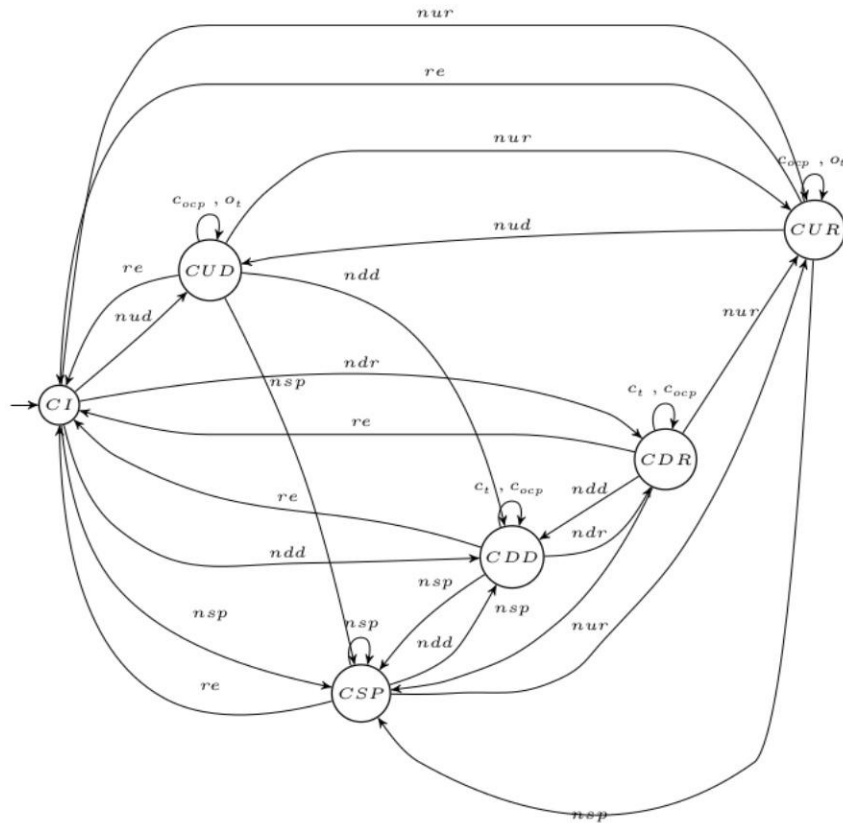


Figura 4.2: Autômato do controlador.

4.1.3 Modelagem da planta

O modelo discreto da planta é dado pela composição paralela dos elementos do sistema, no caso, do autômato correspondente à válvula e autômato correspondente ao controlador. Com auxílio do software *DESLAB*, foi realizada a composição paralela entre os autômatos. O código se encontra na seção A.1.2. Uma *toolbox* no programa foi criada para exportar os dados automaticamente na forma da tabela 4.3. O autômato G_p da planta é, portanto, composto de 33 estados, 11 eventos e 198 transições.

4.2 Modelo do diagnosticador de falha

A construção do diagnosticador de teste, isto é, do autômato usado para fazer a verificação da diagnosticabilidade do sistema, por meio do algoritmo descrito pelo teorema 2.2 possui a informação necessária à verificação da diagnosticabilidade da

Tabela 4.3: Tabela de transição de estados da Planta

		Eventos										
		ct	ndd	ndr	nsp	nud	nur	ocp	ot	re	sc	so
Estados	VICI	-	VICDD	VICDR	VICSP	VICUD	VICUR	-	-	-	VSCCI	VSOCI
	VOCDR	VCCDD	-	VOCDR	VOCS	-	-	VPOCCDD	-	VICI	VSCDD	VSOCDD
	VSCCUR	-	-	-	VSCCSP	VSCCUD	-	VSCCUR	VSCCUR	-	-	-
	VICSP	-	VICDD	-	VICSP	-	VICUR	-	-	-	VSCCSP	VSOCSP
	VSCCUDR	VSCCUDR	VSCCDD	-	VSCCSP	-	VSCCUR	VSCCUDR	-	-	-	-
	VSCCDD	VSCCDD	-	VSCCUDR	VSCCSP	-	-	VSCCDD	-	-	-	-
	VOCUD	-	VOCDD	-	VOCSP	-	VOCUR	VPOCCUD	VOCUD	VICI	VSCCUD	VSOCUD
	VOCUR	-	-	-	VOCSP	VOCUD	-	VPOCCUR	VOCUR	VICI	VSCCUR	VSOCUR
	VPOCCDR	VCCDR	VPOCCDD	-	VPOCCSP	-	VPOCCUR	VPOCCDR	-	VICI	VSCCUDR	VSOCDR
	VOCDR	VCCDR	VOCDR	-	VOCS	-	VOCUR	VPOCCDR	-	VICI	VSCCUDR	VSOCDR
	VSCCUD	-	VSCCDD	-	VSCCSP	-	VSCCUR	VSCCUD	VSCCUD	-	-	-
	VPOCCDD	VCCDD	-	VPOCCDR	VPOCCSP	-	-	VPOCCDD	-	VICI	VSCCDD	VSOCDD
	VSOCUD	-	VSOCDD	-	VSOCSP	-	VSOCUR	VSOCUD	VSOCUD	-	-	-
	VICDR	VCCDR	VICDD	-	VICSP	-	VICUR	VPOCCDR	-	-	VSCCUDR	VSOCDR
	VSOCDR	VSCCUDR	VSOCDD	-	VSOCSP	-	VSOCUR	VSOCDR	-	-	-	-
	VPOCCSP	-	VPOCCDD	-	VPOCCSP	-	VPOCCUR	-	-	VICI	VSCCSP	VSOCSP
	VICDD	VCCDD	-	VICDR	VICSP	-	-	VPOCCDD	-	-	VSCCDD	VSOCDD
	VCCDR	VCCDR	VCCDD	-	VCCSP	-	VCCUR	VPOCCDR	-	VICI	VSCCUDR	VSOCDR
	VSCCSP	-	VSCCDD	-	VSCCSP	-	VSCCUR	-	-	-	-	-
	VSOCUR	-	-	-	VSOCSP	VSOCUD	-	VSOCUR	VSOCUR	-	-	-
	VPOCCUD	-	VPOCCDD	-	VPOCCSP	-	VPOCCUR	VPOCCUD	VOCUD	VICI	VSCCUD	VSOCUD
	VOCSP	-	VOCDR	-	VOCS	-	VOCUR	-	-	VICI	VSCCSP	VSOCSP
	VCCUD	-	VCCDD	-	VCCSP	-	VCCUR	VPOCCUD	VOCUD	VICI	VSCCUD	VSOCUD
	VCCUR	-	-	-	VCCSP	VCCUD	-	VPOCCUR	VOCUR	VICI	VSCCUR	VSOCUR
	VSOCSP	-	VSOCDD	-	VSOCSP	-	VSOCUR	-	-	-	-	-
	VSOCI	-	VSOCDD	VSOCDR	VSOCSP	VSOCUD	VSOCUR	-	-	-	-	-
	VPOCCUR	-	-	-	VPOCCSP	VPOCCUD	-	VPOCCUR	VOCUR	VICI	VSCCUR	VSOCUR
	VCCDD	VCCDD	-	VCCDR	VCCSP	-	-	VPOCCDD	-	VICI	VSCCDD	VSOCDD
	VSCCI	-	VSCCDD	VSCCUDR	VSCCSP	VSCCUD	VSCCUR	-	-	-	-	-
	VCCSP	-	VCCDD	-	VCCSP	-	VCCUR	-	-	VICI	VSCCSP	VSOCSP
	VICUR	-	-	-	VICSP	VICUD	-	VPOCCUR	VOCUR	-	VSCCUD	VSOCUR
	VICUD	-	VICDD	-	VICSP	-	VICUR	VPOCCUD	VOCUD	-	VSCCUD	VSOCUD
VSOCDD	VSOCDD	-	VSOCDR	VSOCSP	-	-	VSOCDD	-	-	-	-	

linguagem que modela o comportamento da planta e será usada para a análise da diagnosticabilidade do modelo da planta obtido na seção 4.1.3 para cada uma das falhas.

O algoritmo para a análise da diagnosticabilidade através da busca por elementos fortemente conexos foi implementado com o código descrito na seção A.1.3 do apêndice A para a falha *sc*. O mesmo código foi usado de forma análoga para a falha *so*. Foi constatado que o sistema não é diagnosticável para as falhas *so* e *sc* correspondente aos eventos não observáveis para a válvula travada na posição aberta e na posição fechada, respectivamente.

O programa *DESLAB* fornece uma representação gráfica do autômato diagnosticador. Um exemplo de um ciclo dentro de um componente fortemente conexo é mostrado na figura 4.3. Isso é suficiente para a condição de não diagnosticabilidade do modelo, de acordo com o teorema 2.2. Note que os estados pertencentes a esse ciclo são da forma possuem (x_d, x_l) , tal que x_d é incerto e x_l contém o rótulo Y.

Foram encontradas 7 componentes fortemente conexas no autômato. Destas, 3 contém somente estados de falha. A seguir, as componentes fortemente conexas encontradas são mostradas para o caso da falha totalmente fechada *sc*:

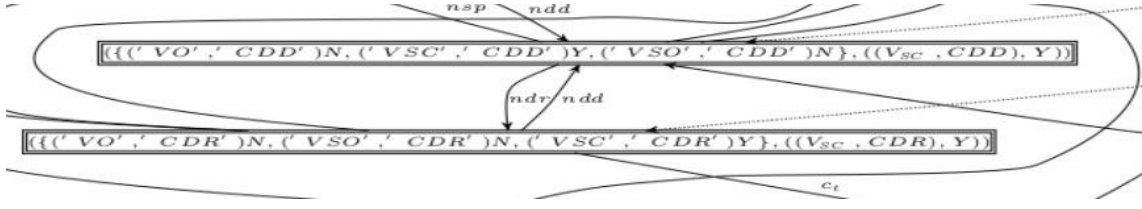


Figura 4.3: Ciclo em uma componente fortemente conexa que viola a condição de diagnosticabilidade no autômato G_{scc} .

1. (((('VO', 'CDD'), 'N'), (('VSC', 'CDD'), 'Y'), (('VSO', 'CDD'), 'N'))), (('VSC', 'CDD'), 'Y')
 (((('VO', 'CDR'), 'N'), (('VSO', 'CDR'), 'N'), (('VSC', 'CDR'), 'Y'))), (('VSC', 'CDR'), 'Y')
 (((('VSO', 'CDR'), 'N'), (('VPOC', 'CDR'), 'N'), (('VSC', 'CDR'), 'Y'))), (('VSC', 'CDR'), 'Y')
 (((('VSO', 'CSP'), 'N'), (('VSC', 'CSP'), 'Y'), (('VPOC', 'CSP'), 'N'))), (('VSC', 'CSP'), 'Y')
 (((('VSO', 'CDD'), 'N'), (('VSC', 'CDD'), 'Y'), (('VPOC', 'CDD'), 'N'))), (('VSC', 'CDD'), 'Y')
 (((('VSO', 'CDD'), 'N'), (('VSC', 'CDD'), 'Y'), (('VC', 'CDD'), 'N'))), (('VSC', 'CDD'), 'Y')
 (((('VSO', 'CDR'), 'N'), (('VC', 'CDR'), 'N'), (('VSC', 'CDR'), 'Y'))), (('VSC', 'CDR'), 'Y')
 (((('VSO', 'CSP'), 'N'), (('VSC', 'CSP'), 'Y'), (('VC', 'CSP'), 'N'))), (('VSC', 'CSP'), 'Y')
 (((('VSO', 'CUR'), 'N'), (('VSC', 'CUR'), 'Y'), (('VC', 'CUR'), 'N'))), (('VSC', 'CUR'), 'Y')
 (((('VC', 'CUD'), 'N'), (('VSC', 'CUD'), 'Y'), (('VSO', 'CUD'), 'N'))), (('VSC', 'CUD'), 'Y')
 (((('VPOC', 'CUD'), 'N'), (('VSC', 'CUD'), 'Y'), (('VSO', 'CUD'), 'N'))), (('VSC', 'CUD'), 'Y')
 (((('VSO', 'CUR'), 'N'), (('VSC', 'CUR'), 'Y'), (('VPOC', 'CUR'), 'N'))), (('VSC', 'CUR'), 'Y')
 (((('VSO', 'CUR'), 'N'), (('VSC', 'CUR'), 'Y'), (('VO', 'CUR'), 'N'))), (('VSC', 'CUR'), 'Y')
 (((('VSO', 'CUD'), 'N'), (('VSC', 'CUD'), 'Y'), (('VO', 'CUD'), 'N'))), (('VSC', 'CUD'), 'Y')
 (((('VSO', 'CSP'), 'N'), (('VSC', 'CSP'), 'Y'), (('VO', 'CSP'), 'N'))), (('VSC', 'CSP'), 'Y')
2. (((('VSO', 'CSP'), 'N'), (('VSC', 'CSP'), 'Y'), (('VI', 'CSP'), 'N'))), (('VSC', 'CSP'), 'Y')
 (((('VI', 'CDD'), 'N'), (('VSO', 'CDD'), 'N'), (('VSC', 'CDD'), 'Y'))), (('VSC', 'CDD'), 'Y')
 (((('VSO', 'CDR'), 'N'), (('VI', 'CDR'), 'N'), (('VSC', 'CDR'), 'Y'))), (('VSC', 'CDR'), 'Y')
 (((('VSO', 'CUR'), 'N'), (('VSC', 'CUR'), 'Y'), (('VI', 'CUR'), 'N'))), (('VSC', 'CUR'), 'Y')
 (((('VI', 'CUD'), 'N'), (('VSC', 'CUD'), 'Y'), (('VSO', 'CUD'), 'N'))), (('VSC', 'CUD'), 'Y')
3. (((('VSC', 'CI'), 'Y'), (('VSO', 'CI'), 'N'), (('VI', 'CI'), 'N'))), (('VSC', 'CI'), 'Y')

A tabela 4.4 mostra o diagrama de transição de estados do diagnosticador *Viana-Basilio* com relação à falha *sc*. Para a falha *so*, obtém-se de forma análoga à falha *sc*. A tabela foi obtida de forma automática através de uma *toolbox* criada para o *DESLAB* que converte um autômato em uma tabela *LaTex*. Com objetivo de se obter uma tabela menor, foram removidos os caracteres ' e os espaços dos nomes dos estados. Entretanto, como ainda seria uma tabela muito grande para apresentar, os estados foram renomeados da forma a seguir.

- q1: [((VSC,CI),Y),((VSO,CI),N),((VI,CI),N)],((VI,CI),N)
 q2: [((VSO,CUR),N),((VSC,CUR),Y),((VI,CUR),N)],((VSO,CUR),N)
 q3: [((VC,CUD),N),((VSC,CUD),Y),((VSO,CUD),N)],((VSC,CUD),Y)

q4: [((VSO,CUR),N),((VSC,CUR),Y),((VI,CUR),N)],((VI,CUR),N)
q5: [((VC,CUD),N),((VSC,CUD),Y),((VSO,CUD),N)],((VC,CUD),N)
q6: [((VSO,CUR),N),((VSC,CUR),Y),((VPOC,CUR),N)],((VSC,CUR),Y)
q7: [((VSO,CDR),N),((VI,CDR),N),((VSC,CDR),Y)],((VI,CDR),N)
q8: [((VSO,CSP),N),((VSC,CSP),Y),((VC,CSP),N)],((VSO,CSP),N)
q9: [((VSO,CDD),N),((VSC,CDD),Y),((VPOC,CDD),N)],((VSC,CDD),Y)
q10: [((VSO,CDR),N),((VI,CDR),N),((VSC,CDR),Y)],((VSC,CDR),Y)
q11: [((VSO,CSP),N),((VSC,CSP),Y),((VO,CSP),N)],((VO,CSP),N)
q12: [((VSO,CDR),N),((VC,CDR),N),((VSC,CDR),Y)],((VSC,CDR),Y)
q13: [((VSO,CUR),N),((VSC,CUR),Y),((VC,CUR),N)],((VSC,CUR),Y)
q14: [((VC,CUD),N),((VSC,CUD),Y),((VSO,CUD),N)],((VSO,CUD),N)
q15: [((VSO,CUD),N),((VSC,CUD),Y),((VO,CUD),N)],((VO,CUD),N)
q16: [((VSO,CUR),N),((VSC,CUR),Y),((VO,CUR),N)],((VSC,CUR),Y)
q17: [((VSO,CDD),N),((VSC,CDD),Y),((VPOC,CDD),N)],((VPOC,CDD),N)
q18: [((VO,CDR),N),((VSO,CDR),N),((VSC,CDR),Y)],((VO,CDR),N)
q19: [((VO,CDD),N),((VSC,CDD),Y),((VSO,CDD),N)],((VSO,CDD),N)
q20: [((VSO,CUR),N),((VSC,CUR),Y),((VO,CUR),N)],((VSO,CUR),N)
q21: [((VSO,CDD),N),((VSC,CDD),Y),((VC,CDD),N)],((VC,CDD),N),
q22: [((VSO,CDR),N),((VPOC,CDR),N),((VSC,CDR),Y)],((VSO,CDR),N)
q23: [((VSO,CUR),N),((VSC,CUR),Y),((VC,CUR),N)],((VC,CUR),N)
q24: [((VO,CDD),N),((VSC,CDD),Y),((VSO,CDD),N)],((VSC,CDD),Y)
q25: [((VSO,CUR),N),((VSC,CUR),Y),((VI,CUR),N)],((VSC,CUR),Y)
q26: [((VI,CUD),N),((VSC,CUD),Y),((VSO,CUD),N)],((VSO,CUD),N)
q27: [((VI,CUD),N),((VSC,CUD),Y),((VSO,CUD),N)],((VI,CUD),N)
q28: [((VPOC,CUD),N),((VSC,CUD),Y),((VSO,CUD),N)],((VSC,CUD),Y)
q29: [((VSO,CUR),N),((VSC,CUR),Y),((VO,CUR),N)],((VO,CUR),N)
q30: [((VI,CDD),N),((VSO,CDD),N),((VSC,CDD),Y)],((VI,CDD),N)
q31: [((VSO,CSP),N),((VSC,CSP),Y),((VC,CSP),N)],((VC,CSP),N)
q32: [((VSO,CDR),N),((VC,CDR),N),((VSC,CDR),Y)],((VC,CDR),N)
q33: [((VSO,CSP),N),((VSC,CSP),Y),((VPOC,CSP),N)],((VSO,CSP),N)
q34: [((VO,CSP),N),((VSC,CSP),Y),((VSO,CSP),N)],((VSO,CSP),N)
q35: [((VSO,CDR),N),((VI,CDR),N),((VSC,CDR),Y)],((VSO,CDR),N)
q36: [((VPOC,CUD),N),((VSC,CUD),Y),((VSO,CUD),N)],((VPOC,CUD),N)
q37: [((VO,CDD),N),((VSC,CDD),Y),((VSO,CDD),N)],((VO,CDD),N)
q38: [((VSO,CSP),N),((VSC,CSP),Y),((VI,CSP),N)],((VSO,CSP),N)
q39: [((VSC,CI),Y),((VSO,CI),N),((VI,CI),N)],((VSC,CI),Y)
q40: [((VSO,CSP),N),((VSC,CSP),Y),((VC,CSP),N)],((VSC,CSP),Y)

q41: [((VO,CDR),N),((VSO,CDR),N),((VSC,CDR),Y)],((VSO,CDR),N)
q42: [((VO,CDR),N),((VSO,CDR),N),((VSC,CDR),Y)],((VSC,CDR),Y)
q43: [((VSO,CDD),N),((VSC,CDD),Y),((VC,CDD),N)],((VSO,CDD),N)
q44: [((VI,CDD),N),((VSO,CDD),N),((VSC,CDD),Y)],((VSC,CDD),Y)
q45: [((VSO,CDR),N),((VPOC,CDR),N),((VSC,CDR),Y)],((VSC,CDR),Y)
q46: [((VI,CUD),N),((VSC,CUD),Y),((VSO,CUD),N)],((VSC,CUD),Y)
q47: [((VSC,CI),Y),((VSO,CI),N),((VI,CI),N)],((VSO,CI),N)
q48: [((VPOC,CUD),N),((VSC,CUD),Y),((VSO,CUD),N)],((VSO,CUD),N)
q49: [((VSO,CSP),N),((VSC,CSP),Y),((VPOC,CSP),N)],((VPOC,CSP),N)
q50: [((VSO,CSP),N),((VSC,CSP),Y),((VI,CSP),N)],((VSC,CSP),Y)
q51: [((VSO,CUR),N),((VSC,CUR),Y),((VC,CUR),N)],((VSO,CUR),N)
q52: [((VSO,CSP),N),((VSC,CSP),Y),((VI,CSP),N)],((VI,CSP),N)
q53: [((VSO,CSP),N),((VSC,CSP),Y),((VPOC,CSP),N)],((VSC,CSP),Y)
q54: [((VSO,CDR),N),((VC,CDR),N),((VSC,CDR),Y)],((VSO,CDR),N)
q55: [((VSO,CDD),N),((VSC,CDD),Y),((VC,CDD),N)],((VSC,CDD),Y)
q56: [((VSO,CUR),N),((VSC,CUR),Y),((VPOC,CUR),N)],((VSO,CUR),N)
q57: [((VSO,CDD),N),((VSC,CDD),Y),((VPOC,CDD),N)],((VSO,CDD),N)
q58: [((VSO,CUR),N),((VSC,CUR),Y),((VPOC,CUR),N)],((VPOC,CUR),N)
q59: [((VI,CDD),N),((VSO,CDD),N),((VSC,CDD),Y)],((VSO,CDD),N)
q60: [((VSO,CUD),N),((VSC,CUD),Y),((VO,CUD),N)],((VSO,CUD),N)
q61: [((VO,CSP),N),((VSC,CSP),Y),((VSO,CSP),N)],((VSC,CSP),Y)
q62: [((VSO,CDR),N),((VPOC,CDR),N),((VSC,CDR),Y)],((VPOC,CDR),N)
q63: [((VSO,CUD),N),((VSC,CUD),Y),((VO,CUD),N)],((VSC,CUD),Y)

4.2.1 Mapa de sensores

A abordagem por mapa de sensores foi realizada conforme a seção 2.3.3 . Foi considerado em estado intermediário entre os estados de origem e chegada das falhas e, baseado no modelo comportamental do sistema, adicionou-se uma transição observável correspondente ao efeito da falha-aberta (*so*) e da falha-fechada (*sc*). A transição no sistema devido à falha aberta corresponde, portanto, ao sensor virtual de nível nível abaixo do *setpoint* e descendo (*ndd*) e o comando de fechar totalmente a válvula *ct*, isto é, um sensor denominado *ctnndd*, enquanto a falha fechada promove a transição de estado para o nível acima do *setpoint* e subindo *nur* e o comando

Tabela 4.4: Tabela de transição de estados do Diagnosticador para a falha sc

		Eventos										
		ct	ndd	ndr	nsp	nud	nur	ocp	ot	re	sc	so
Estados	q1	-	q30	q7	q52	q27	q4	-	-	-	q39	q47
	q2	-	-	-	q38	q26	-	q56	q20	-	-	-
	q3	-	q55	-	q40	-	q13	q28	q63	-	-	-
	q4	-	-	-	q52	q27	-	q58	q29	-	q25	q2
	q5	-	q21	-	q31	-	q23	q36	q15	q1	q3	q14
	q6	-	-	-	q53	q28	-	q6	q16	-	-	-
	q7	q32	q30	-	q52	-	q4	q62	-	-	q10	q35
	q8	-	q43	-	q8	-	q51	-	-	-	-	-
	q9	q55	-	q45	q53	-	-	q9	-	-	-	-
	q10	q12	q44	-	q50	-	q25	q45	-	-	-	-
	q11	-	q37	-	q11	-	q29	-	-	q1	q61	q34
	q12	q12	q55	-	q40	-	q13	q45	-	-	-	-
	q13	-	-	-	q40	q3	-	q6	q16	-	-	-
	q14	-	q43	-	q8	-	q51	q48	q60	-	-	-
	q15	-	q37	-	q11	-	q29	q36	q15	q1	q63	q60
	q16	-	-	-	q61	q63	-	q6	q16	-	-	-
	q17	q21	-	q62	q49	-	-	q17	-	q1	q9	q57
	q18	q32	q37	-	q11	-	q29	q62	-	q1	q42	q41
	q19	q43	-	q41	q34	-	-	q57	-	-	-	-
	q20	-	-	-	q34	q60	-	q56	q20	-	-	-
	q21	q21	-	q32	q31	-	-	q17	-	q1	q55	q43
	q22	q54	q57	-	q33	-	q56	q22	-	-	-	-
	q23	-	-	-	q31	q5	-	q58	q29	q1	q13	q51
	q24	q55	-	q42	q61	-	-	q9	-	-	-	-
	q25	-	-	-	q50	q46	-	q6	q16	-	-	-
	q26	-	q59	-	q38	-	q2	q48	q60	-	-	-
	q27	-	q30	-	q52	-	q4	q36	q15	-	q46	q26
	q28	-	q9	-	q53	-	q6	q28	q63	-	-	-
	q29	-	-	-	q11	q15	-	q58	q29	q1	q16	q20
	q30	q21	-	q7	q52	-	-	q17	-	-	q44	q59
	q31	-	q21	-	q31	-	q23	-	-	q1	q40	q8
	q32	q32	q21	-	q31	-	q23	q62	-	q1	q12	q54
	q33	-	q57	-	q33	-	q56	-	-	-	-	-
	q34	-	q19	-	q34	-	q20	-	-	-	-	-
	q35	q54	q59	-	q38	-	q2	q22	-	-	-	-
	q36	-	q17	-	q49	-	q58	q36	q15	q1	q28	q48
	q37	q21	-	q18	q11	-	-	q17	-	q1	q24	q19
	q38	-	q59	-	q38	-	q2	-	-	-	-	-
	q39	-	q44	q10	q50	q46	q25	-	-	-	-	-
	q40	-	q55	-	q40	-	q13	-	-	-	-	-
	q41	q54	q19	-	q34	-	q20	q22	-	-	-	-
	q42	q12	q24	-	q61	-	q16	q45	-	-	-	-
	q43	q43	-	q54	q8	-	-	q57	-	-	-	-
	q44	q55	-	q10	q50	-	-	q9	-	-	-	-
	q45	q12	q9	-	q53	-	q6	q45	-	-	-	-
	q46	-	q44	-	q50	-	q25	q28	q63	-	-	-
	q47	-	q59	q35	q38	q26	q2	-	-	-	-	-
	q48	-	q57	-	q33	-	q56	q48	q60	-	-	-
	q49	-	q17	-	q49	-	q58	-	-	q1	q53	q33
	q50	-	q44	-	q50	-	q25	-	-	-	-	-
	q51	-	-	-	q8	q14	-	q56	q20	-	-	-
	q52	-	q30	-	q52	-	q4	-	-	-	q50	q38
	q53	-	q9	-	q53	-	q6	-	-	-	-	-
	q54	q54	q43	-	q8	-	q51	q22	-	-	-	-
	q55	q55	-	q12	q40	-	-	q9	-	-	-	-
	q56	-	-	-	q33	q48	-	q56	q20	-	-	-
	q57	q43	-	q22	q33	-	-	q57	-	-	-	-
	q58	-	-	-	q49	q36	-	q58	q29	q1	q6	q56
	q59	q43	-	q35	q38	-	-	q57	-	-	-	-
	q60	-	q19	-	q34	-	q20	q48	q60	-	-	-
	q61	-	q24	-	q61	-	q16	-	-	-	-	-
	q62	q32	q17	-	q49	-	q58	q62	-	q1	q45	q22
	q63	-	q24	-	q61	-	q16	q28	q63	-	-	-

de abrir válvula completamente (*ot*), ou seja, correspondente à leitura de um sensor chamado *otnur*.

Novamente os algoritmos dos diagnosticadores foram aplicados e, dessa vez, foi possível detectar e diagnosticar ambas as falhas. Os códigos utilizados se encontram no apêndice A nas seções A.1.4 e A.1.5. O novo autômato da planta possui então 75 estados, pois foram adicionados aos 33 estados originais 21 novos estados intermediários correspondentes às transições compostas do evento de falha *sc* e 21 novos estados adicionados correspondentes à falha *so*. O número de eventos continuou 11, que consiste nos 9 eventos observáveis do modelo da planta sem mapa de sensores, e dois novos eventos correspondentes à leitura dos sensores *ctnndd* e *otnur*. O número de transições do novo autômato é de 156, já que foram excluídas as 42 transições dadas pelos eventos não observáveis *sc* e *so* do modelo composto da planta das 198 transições originais. O diagrama do autômato, assim como a tabela de transição de estados, é excessivamente grande para ser apresentada, entretanto, pode ser facilmente obtida com os códigos do apêndice A.

Pelo algoritmo do diagnosticador *offline* *Viana-Basilio*, através do código explicitado na seção A.1.3 do apêndice A, foi verificado que o modelo da planta realizado com auxílio do método mapa de sensores é diagnosticável. A partir deste modelo da planta, foi obtido o diagnosticador que será usado *online* através do algoritmo descrito na seção 2.3.4, cujo código se encontra na seção A.1.1. Este autômato diagnosticador *online* será usado para modelar a diagnose de falha em tempo real através da ferramenta *Stateflow* do software de simulação dinâmica *Simulink*. Com a sintaxe para a API do *Simulink* [27], uma *toolbok* para o *DESLAB* foi criada para exportar automaticamente o autômato obtido com o *DESLAB*, para o *Stateflow*. O código está transcrito na seção A.1.6 do apêndice A.

4.3 Simulação com diagnosticador de falhas

4.3.1 Implementação com Simulink

Para implementar o autômato gerado com o *DESLAB*, foi criada uma *toolbox* que exporta o autômato para a ferramenta *Stateflow* do *Simulink*, cujo código está transcrito na seção A.1.6 do apêndice A. A *toolbox* utilizou os comandos de API do *Simulink* para gerar automaticamente os estados, eventos e transições correspondentes. Como o arquivo *matlab* gerado automaticamente pela *toolbox* possui 1325 linhas para gerar o *Stateflow* correspondente às falhas *sc* e *so*, foi adicionado ao apêndice A, na seção A.2, um exemplo de arquivo *matlab* menor obtido automaticamente para o autômato da válvula, que cria um modelo em *Stateflow* ao ser executado.

O modelo em *Stateflow* gerado possui gráfico de acordo com a figura 4.4 para o tipo de falha *so*. O modelo *Stateflow* para autômato *sc* é análogo. Como o autômato possui grande número de estados e objetivo é analisar a saída do *Stateflow*, não foi necessário automatizar as transições do gráfico para aparecer de forma mais clara.

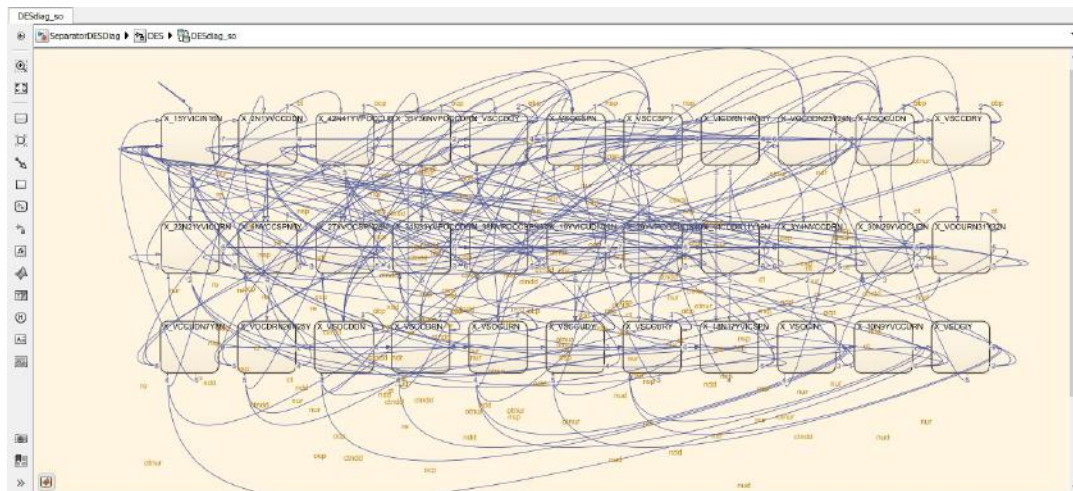


Figura 4.4: Modelo em *Stateflow* para o diagnosticador da falha *so*.

Além disso, para a simulação, foi necessário criar o gerador de eventos para fornecer a informação correspondente aos sensores discretizados e aos sensores virtuais. Cada diagnosticador possui o seu gerador de eventos. Um exemplo da implementação para os sensores virtuais do nível de óleo é visto na figura 4.5.

O modelo do *Simulink* criado para a análise dos diagnosticadores foi conforme

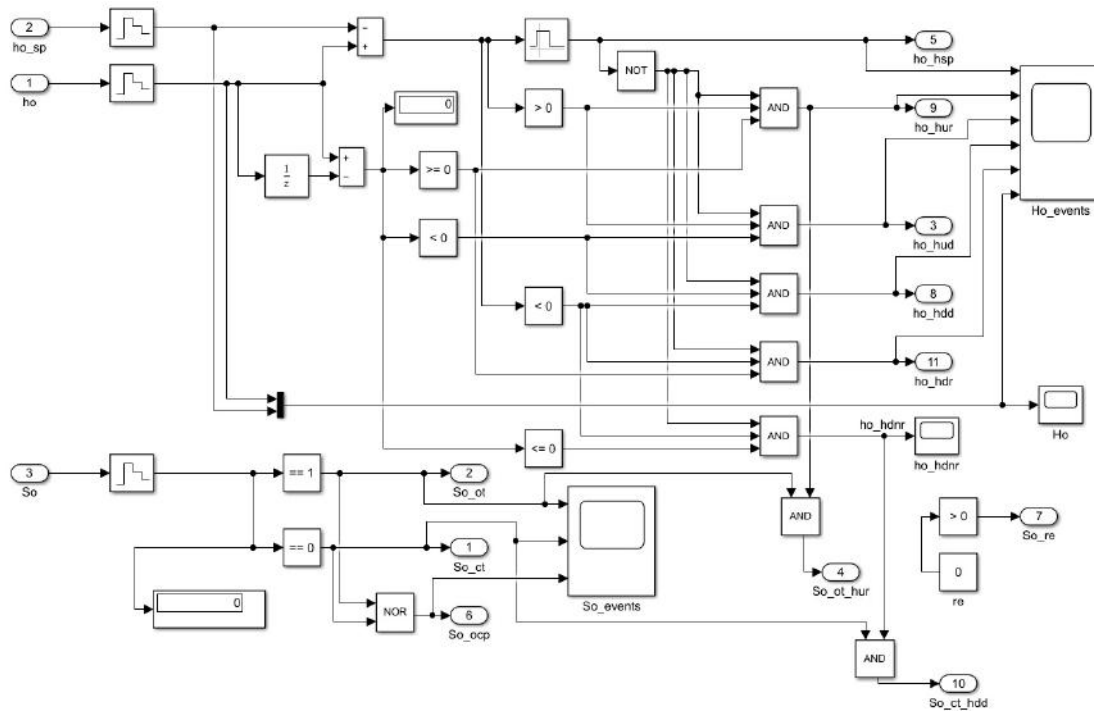


Figura 4.5: Gerador de eventos a partir de sensores reais e virtuais.

a figura 4.6. Como as falhas nas válvulas S_{o1} e S_{o2} não provocam alteração nas variáveis manipuladas, de acordo com a seção 4.3.2 a seguir, apenas foi necessário simular para as válvulas S_g , S_o , S_{o3} e S_{w3} . Para constatar que o sistema foi capaz de diagnosticar a falha, os estados que só contenham o rótulo Y foram unidos e visualizados em um bloco de visualização *scope*, em que a saída corresponde ao valor 1 caso esteja em determinado estado, e 0 caso não esteja. A figura 4.6 ilustra em detalhe o caso das falhas *so sc* para a válvula de óleo da câmara de óleo.

Os resultados do comportamento do sistema e do diagnosticador frente às falhas nas válvulas são apresentados na próxima seção.

4.3.2 Resultados

Nesta subseção, é feita inicialmente a análise do comportamento de falha para cada válvula e a seguir é avaliado se o diagnosticador proposto foi capaz de detectar a falha.

Três parâmetros foram determinados para cada falha, um determina se é um estado normal ou de falha, outro determina se a falha é aberta ou é fechada e o

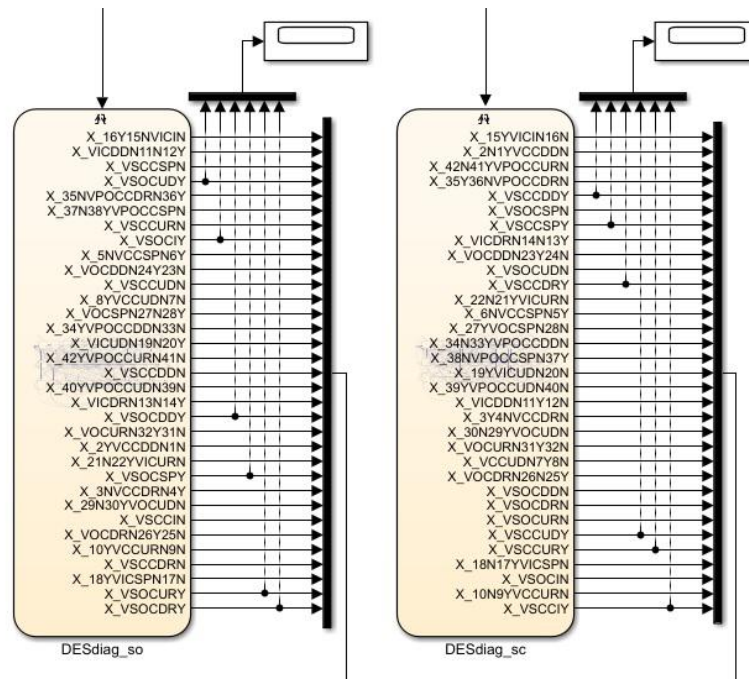


Figura 4.6: Modelo em *Simulink* dos diagnosticadores de falha.

terceiro indica o tempo da ocorrência da falha modelados através do bloco *switch* no *Simulink*, como é visto nas figuras 4.7 e 4.8.

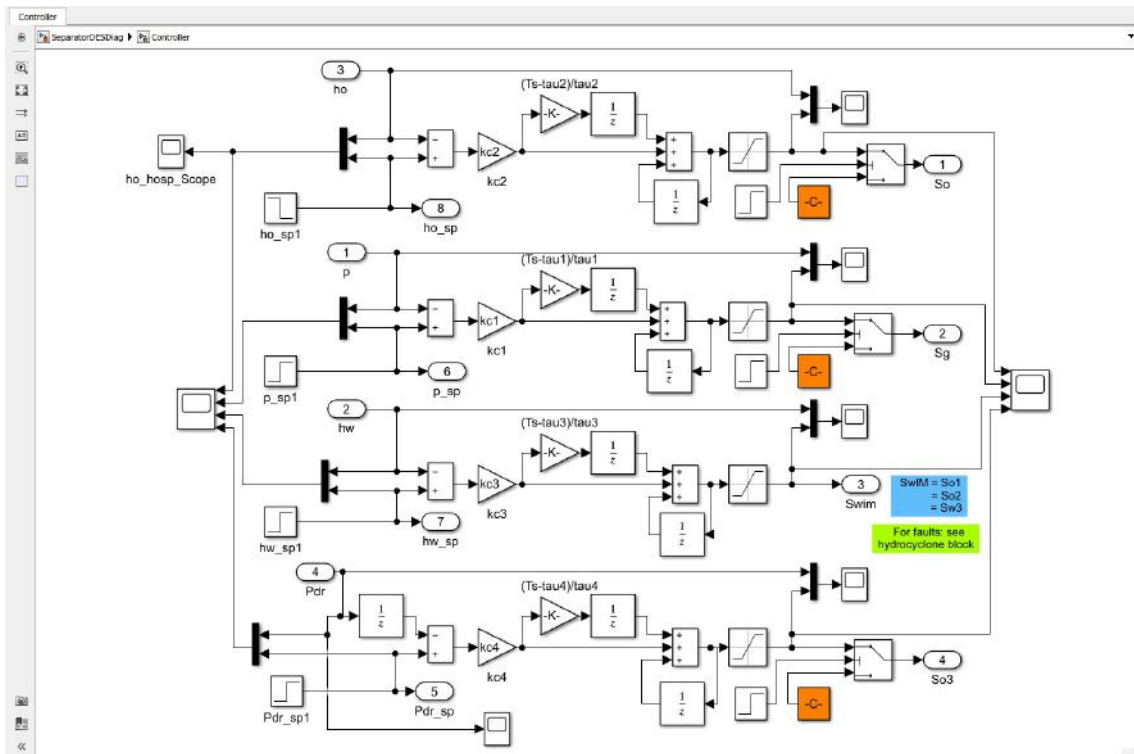


Figura 4.7: Modelo em *Simulink* do controlador e modos de falhas.

Foram simuladas individualmente as falhas abertas e as falhas fechadas nas

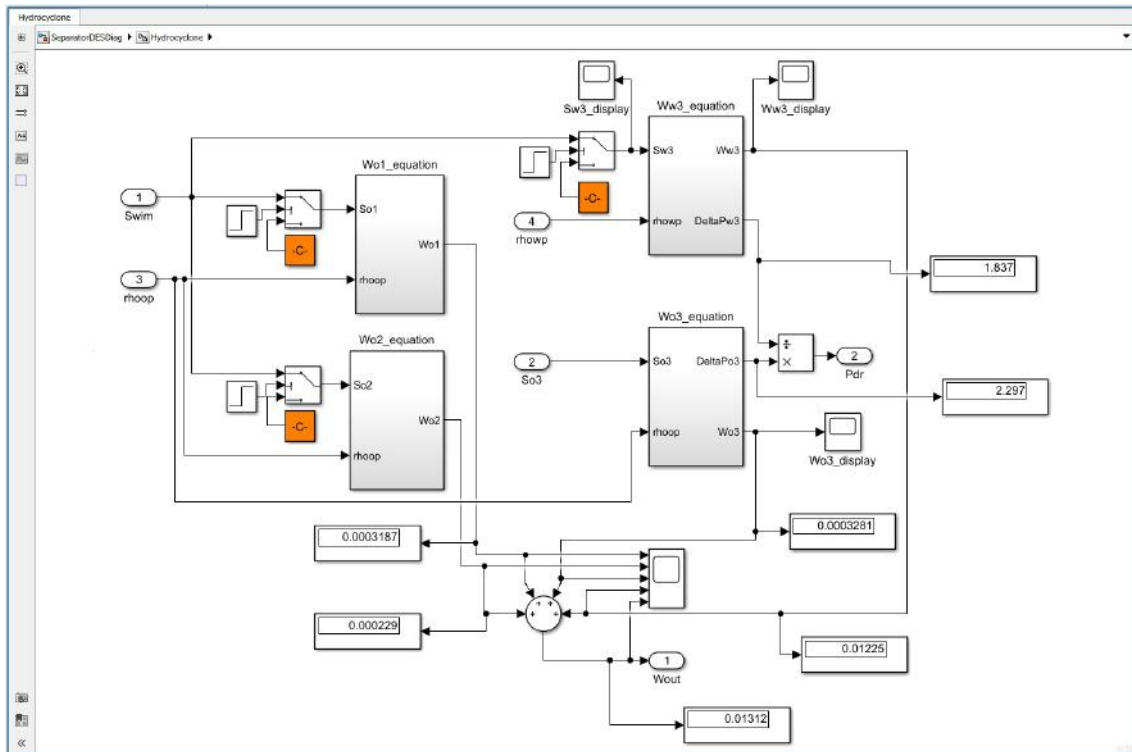


Figura 4.8: Modelo em *Simulink* do hidrociclone e modos de falha.

válvulas de gás e de óleo do vaso separador, nas válvulas de óleo dos hidrociclones BOW, PDC, DC e na válvula de água do hidrociclone DC, para, então, analisar as saídas do controlador, isto é, as variáveis controladas, respectivamente Sg , So , $So1$, $So2$, $So3$, $Sw3$, descritas pela tabela 3.12, de forma que, pela modelagem, tem-se $So1 = So2 = Sw3 = Swim$, e também o comportamento das variáveis manipuladas para cada falha, isto é h_o , p , h_w , P_{dr} são as variáveis controladas. Como é pressuposto que o sistema está em estado estacionário quando há a falha, foram analisadas durante um tempo de $10000s$ as falhas ocorrendo no instante igual a $5000s$. Foram observados portanto os seguintes comportamentos de acordo com as figuras de 4.9 a 4.32 a seguir. Observe que, para a falha aberta da válvula de água, foi necessário parar a simulação quando o nível de água chega ao valor 0, uma vez que corresponde a uma parada geral da planta, já que, neste caso, é crítico o caso de entrar óleo na saída de água. Para o caso da falha fechada da água, também pode ocorrer a parada do sistema, quando o nível de água for suficientemente alto de forma que a água seja levada à câmara de óleo.

4.3.3 Falhas na válvula de óleo

Para avaliar a diagnosticabilidade das falhas na válvula de óleo, foi simulado individualmente cada situação de falha, correspondentes ao parâmetro $so_stuck = 1$ para o caso de falha aberta, em que se considera o agarramento na válvula correspondendo a 100% de abertura na válvula, e ao $so_stuck = 0$ para o caso de travamento com a válvula travada completamente fechada. O tempo determinado pela simulação para a ocorrência da falha foi em $5000s$. As figuras de 4.9 a 4.12 mostram o comportamento das variáveis do sistema após a ocorrência das respectivas falhas.

Pela figura 4.10, é possível notar que, após o evento da válvula falhar aberta, há uma queda abrupta no nível de óleo até saturar no nível nulo. A ação de controle pela válvula de óleo, observada pela figura 4.9, indica uma tentativa de compensar a queda do nível ao enviar o sinal de controle para 0% de abertura, ação que não é capaz de alterar o comportamento do sistema, visto que a válvula está travada. Pelas figuras também é possível notar que a mudança no nível da água acarreta em alteração da pressão do sistema, que é rapidamente controlada pelo controle de pressão.

Já as figuras 4.11 e 4.12 mostram o controlador da válvula de óleo, cujo o sinal era mantido estável próximo à 50% de abertura, passar a aumentar o sinal, a partir do instante $5000s$, até se manter saturado com o sinal de abrir totalmente (100%) sem que isto seja capaz de alterar o nível de óleo, saturado no valor máximo admissível pela modelagem, com $0,867m$.

4.3.4 Falhas na válvula de água

Para simular as falhas na válvula de água na saída no hidrociclone DC, capaz de controlar o nível de água da câmara de separação, foi atribuído ao parâmetro $sw3_stuck$ os valores 1 e 0 correspondentes ao travamento da válvula nas posições 100% e 0% de abertura no tempo de simulação $5000s$.

Pelas figuras 4.13 e 4.14, foi observado o comportamento das variáveis controladas e manipuladas desta malha de controle quando a válvula falha aberta e observou-se

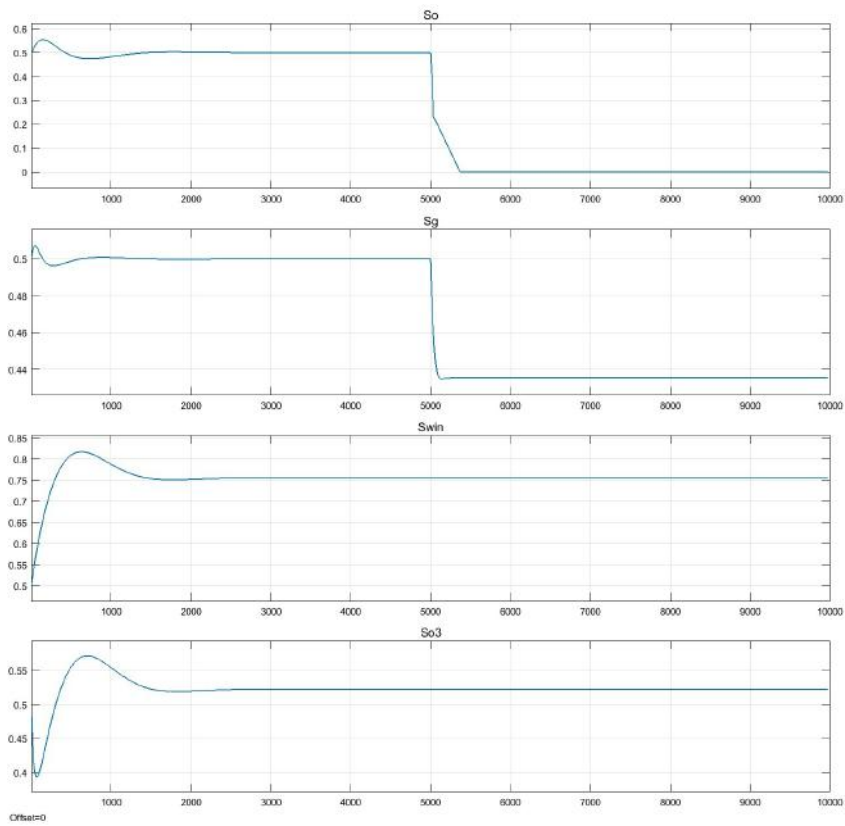


Figura 4.9: Variáveis controladas após falha aberta na válvula de óleo S_o .

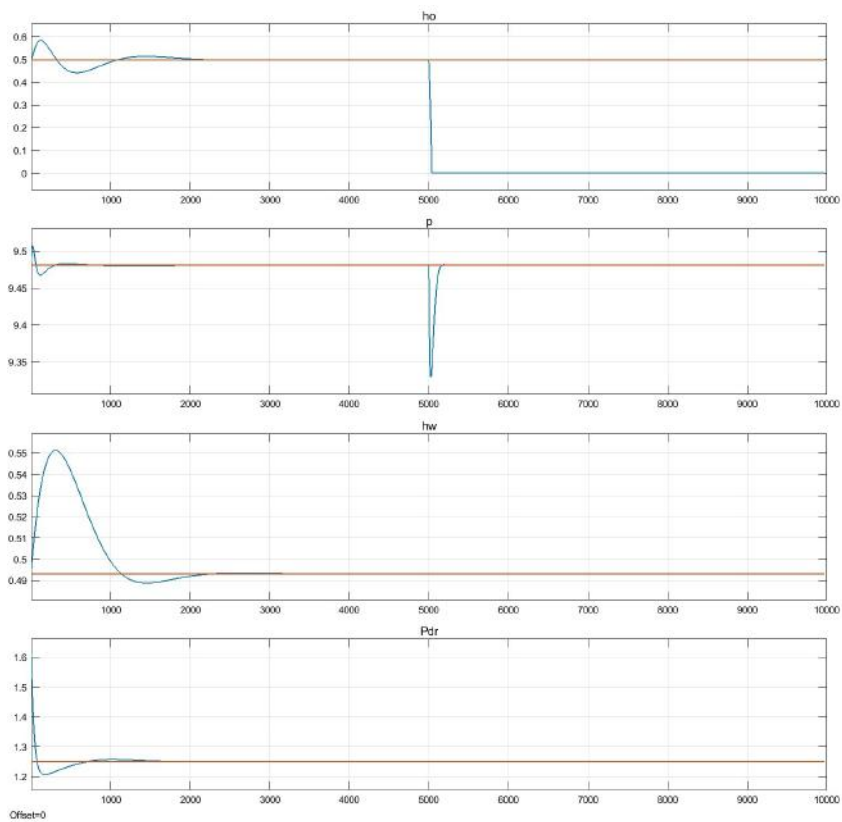


Figura 4.10: Variáveis manipuladas após falha aberta na válvula de óleo S_o .

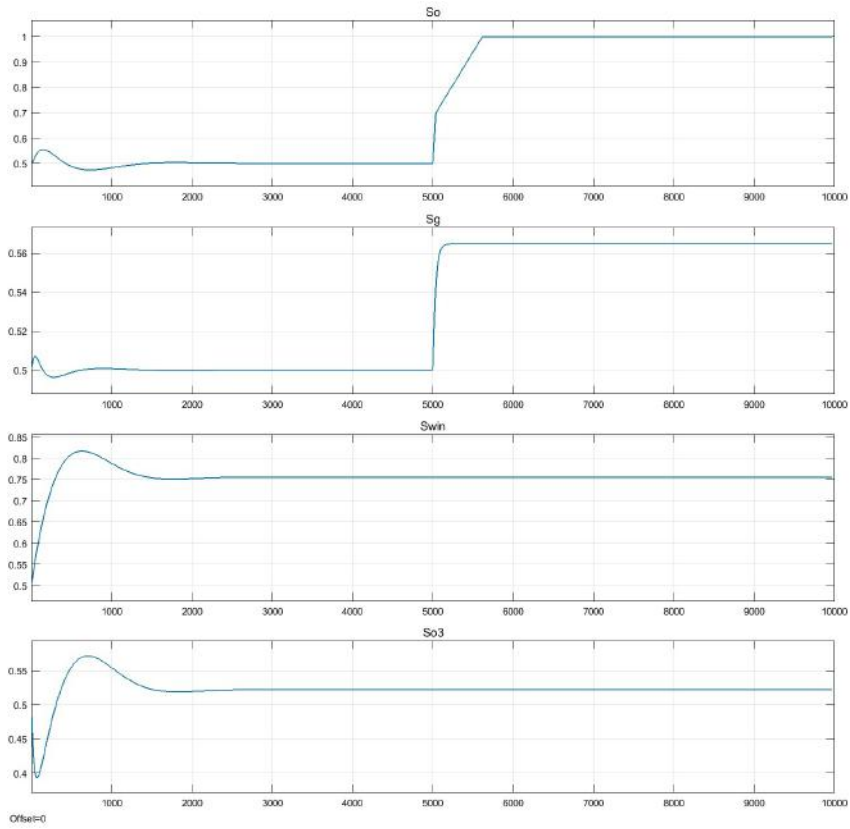


Figura 4.11: Variáveis controladas após falha fechada na válvula de óleo S_o .

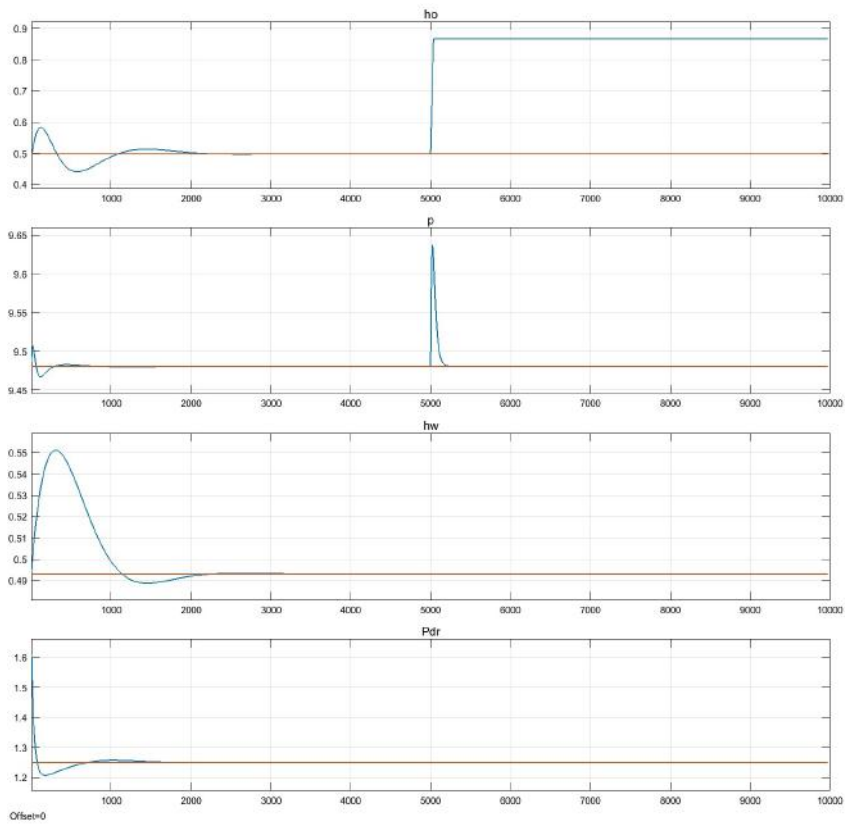


Figura 4.12: Variáveis manipuladas após falha fechada na válvula de óleo S_o .

que o nível de água hw passa a cair quase linearmente a partir do instante 5000s, mesmo com a saída do controlador S_{win} indicando sinal para fechar a válvula e chegando a saturar no valor nulo correspondente ao fechamento completo da válvula. Pouco depois do tempo 6000s a simulação é interrompida pois o nível da água alcança o valor zero, o que corresponde ao estado crítico de parada da planta.

Já as figuras 4.15 e 4.16 mostram o comportamento para o caso de falha fechada na válvula de água, o que é observado com o nível de água crescente a partir do tempo 5000s até alcançar rapidamente a saturação no valor máximo permitido pela modelagem do processo, isto é, no valor da altura da chicana. O nível não volta a diminuir, ainda que o sinal de controle indique o valor de 100% para a válvula abrir completamente.

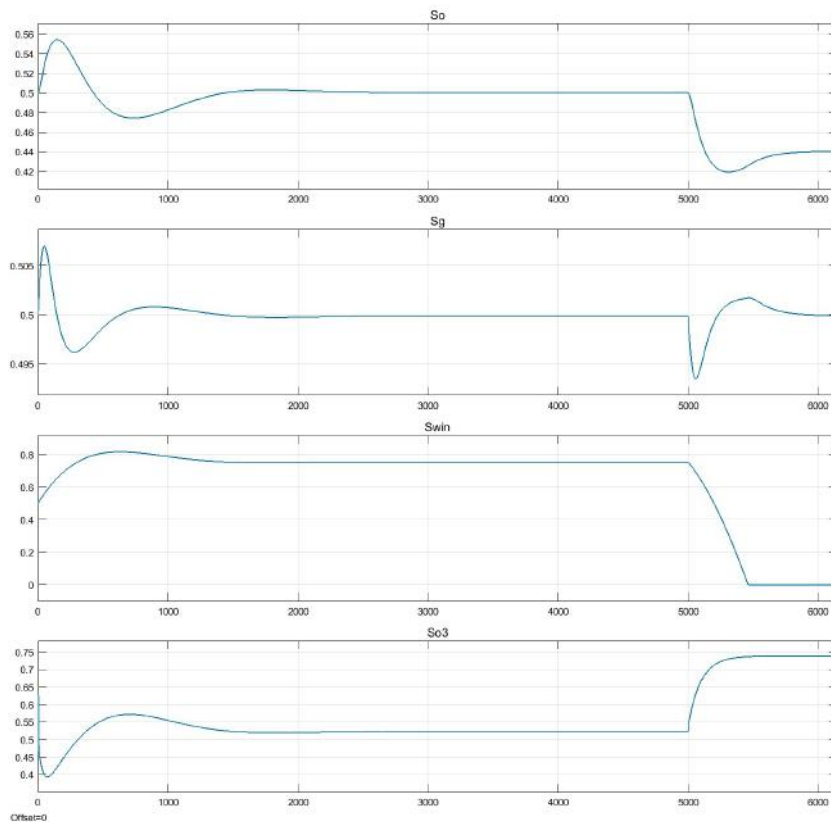


Figura 4.13: Variáveis controladas após falha aberta na válvula de água S_{w3} .

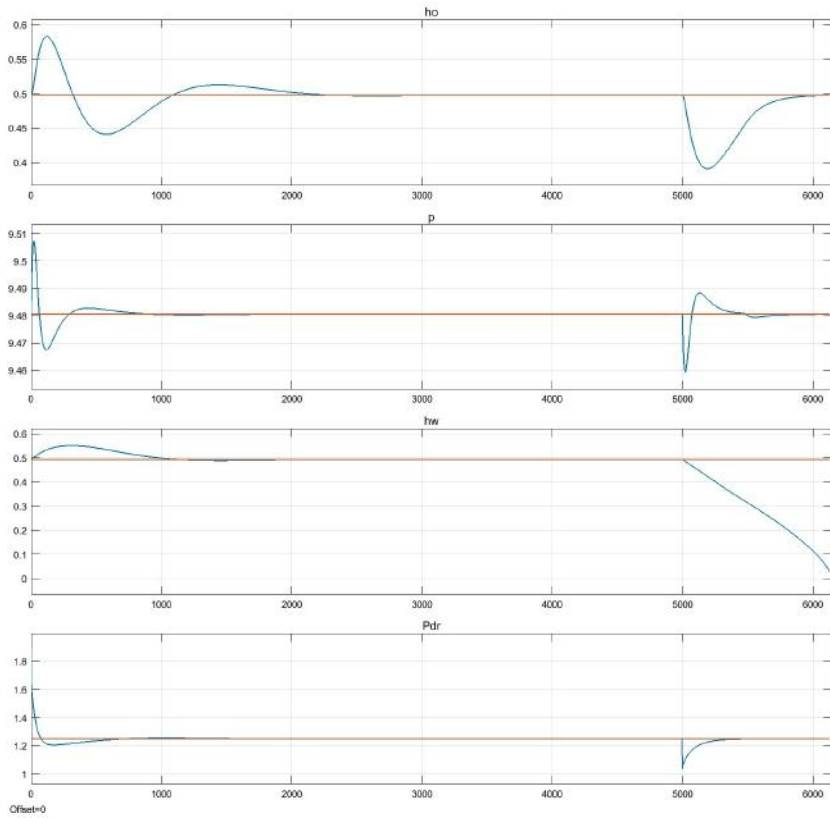


Figura 4.14: Variáveis manipuladas após falha aberta na válvula de água S_{w3} .

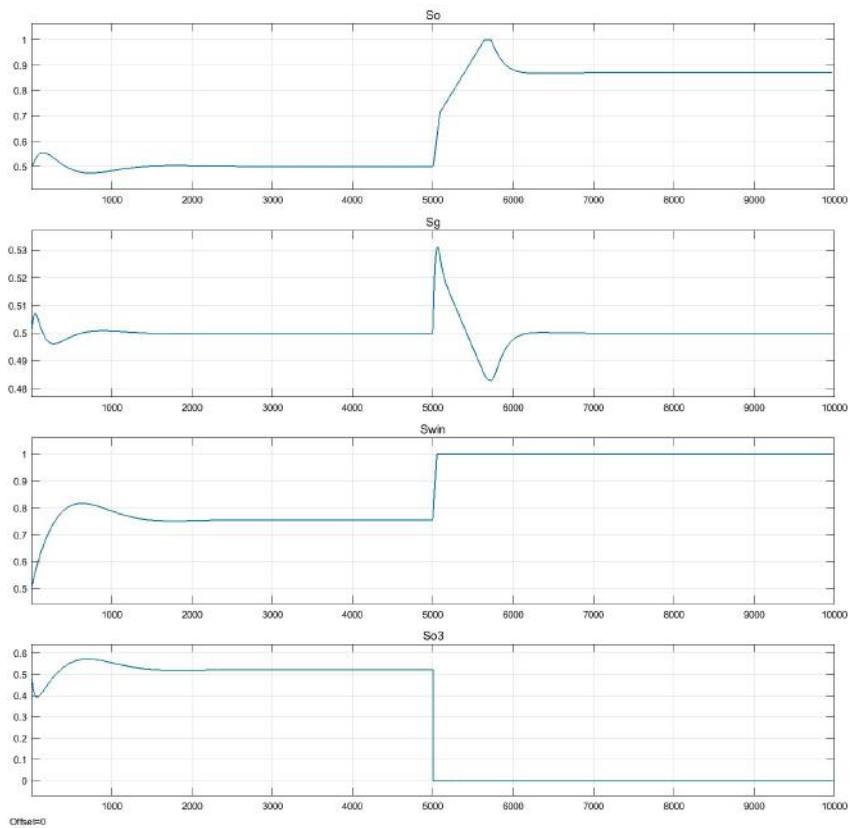


Figura 4.15: Variáveis controladas após falha fechada na válvula de água S_{w3} .

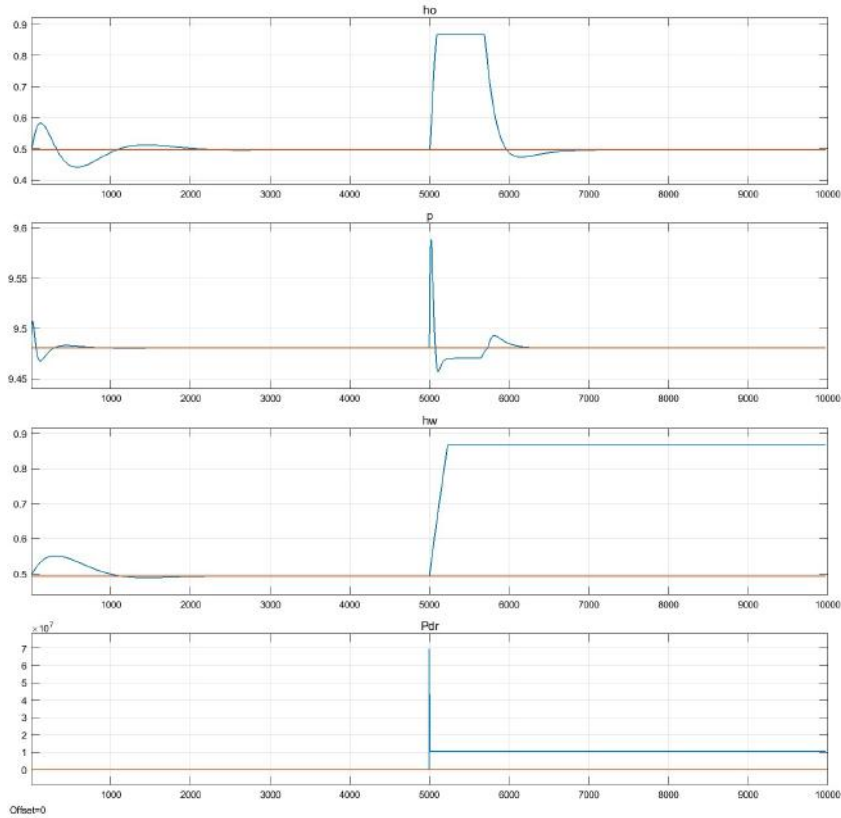


Figura 4.16: Variáveis manipuladas após falha fechada na válvula de água S_{w3} .

4.3.5 Falhas na válvula de óleo dos hidrocilones BOW e PDC

As figuras de 4.17 a 4.24 mostram os efeitos das falhas aberta e fechada das válvulas de óleo dos hidrocilones BOW e PDC. É possível observar que o comportamento do sistema se manteve praticamente inalterado apesar das falhas, e, portanto, o controle para a variação do nível de água h_w , que age nas válvulas S_{o1} , S_{o2} e S_{w3} apenas será afetado quando houver a falha na válvula de água do hidrociclone DC, correspondente à S_{w3} . Desta forma, apenas foi implementado o *stateflow* para diagnosticar a falha nesta última válvula, de acordo com os sensores das variações do nível de água da câmara de separação h_w .

4.3.6 Falhas na válvula de óleo do hidrociclone DC

As figuras de 4.25 a 4.28 correspondem aos efeitos das falhas completamente aberta e completamente fechada da válvula de óleo do hidrociclone DC.

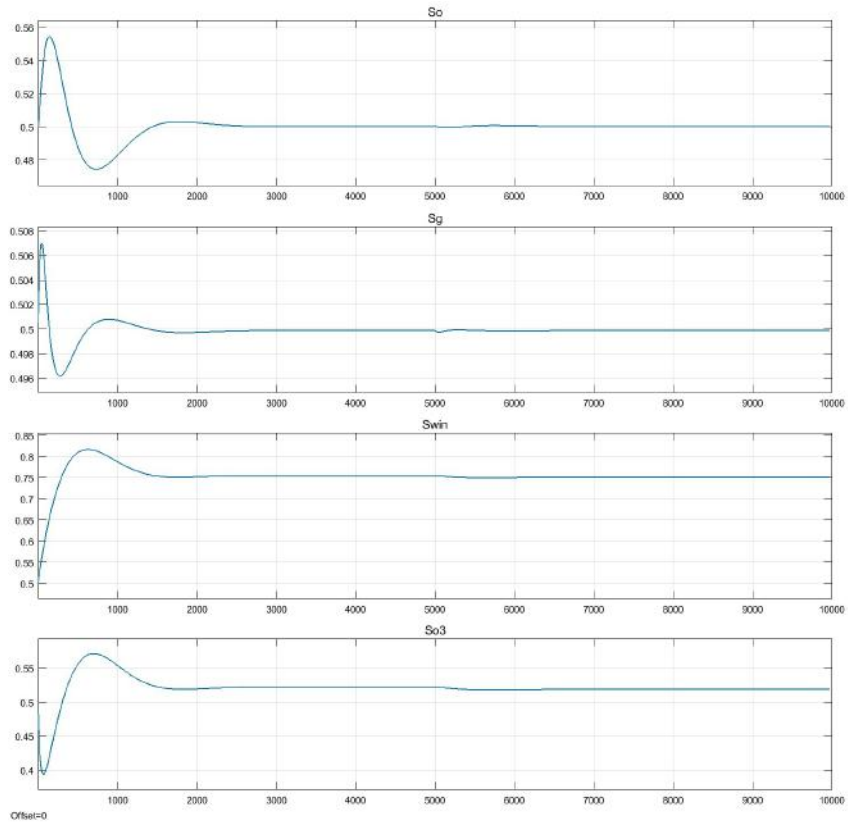


Figura 4.17: Variáveis controladas após falha aberta na válvula de óleo BOW S_{o1} .

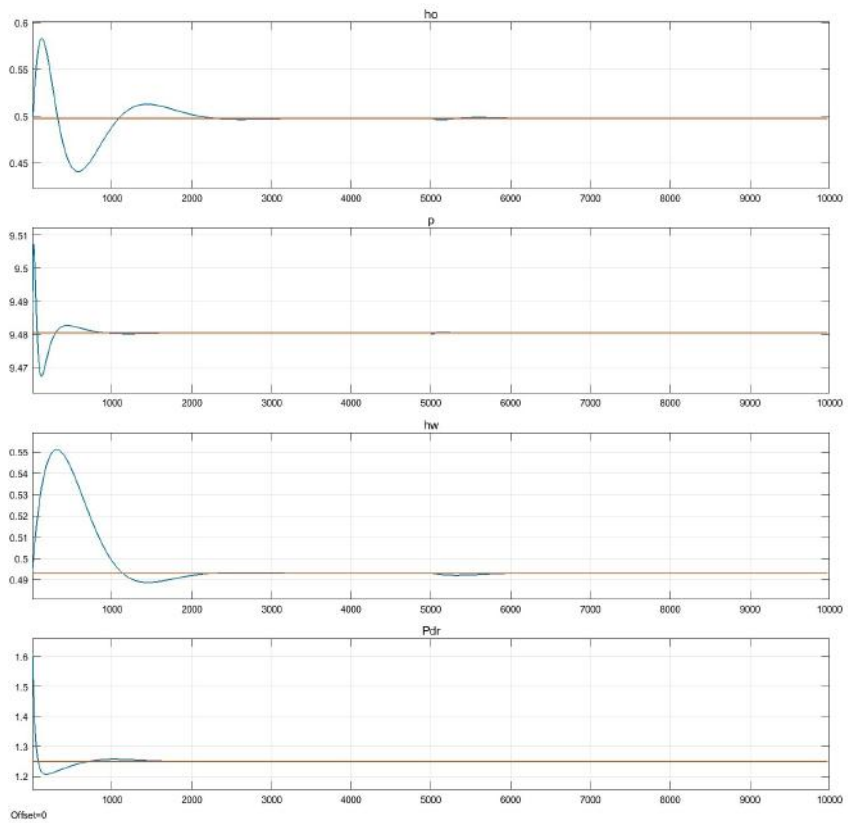


Figura 4.18: Variáveis manipuladas após falha aberta na válvula de óleo BOW S_{o1} .

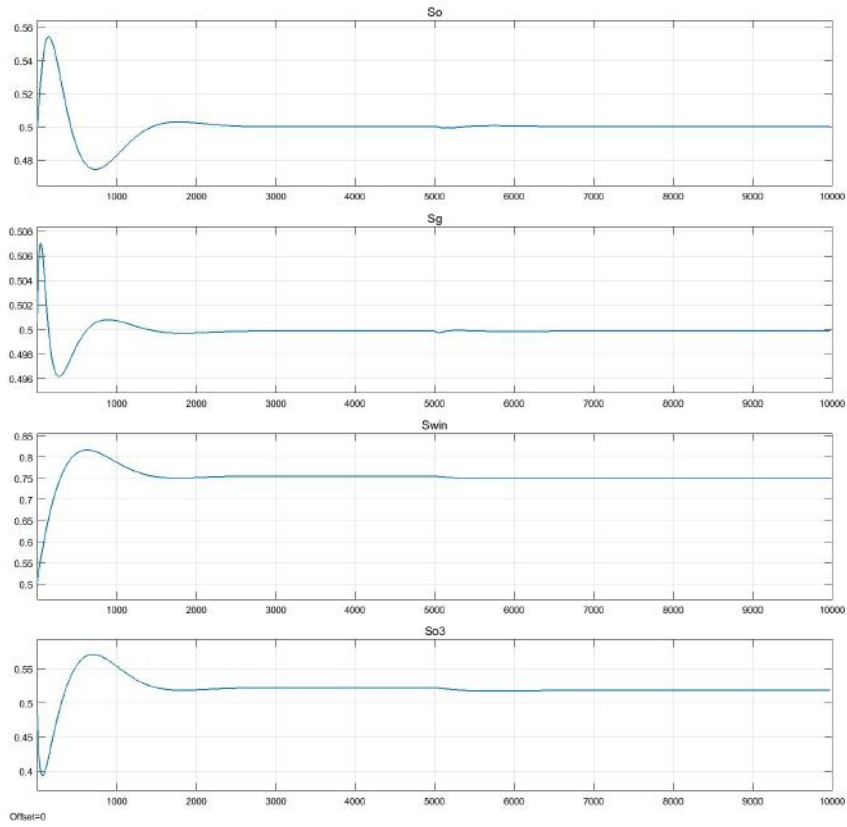


Figura 4.19: Variáveis controladas após falha fechada na válvula de óleo BOW S_{o1} .

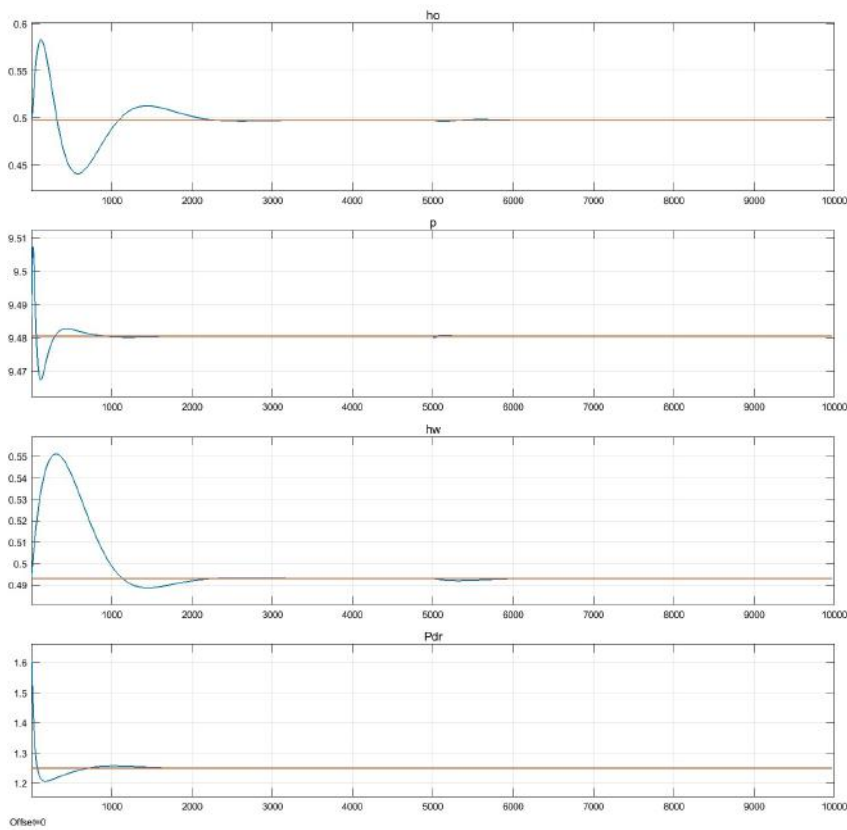


Figura 4.20: Variáveis manipuladas após falha fechada na válvula de óleo BOW S_{o1} .

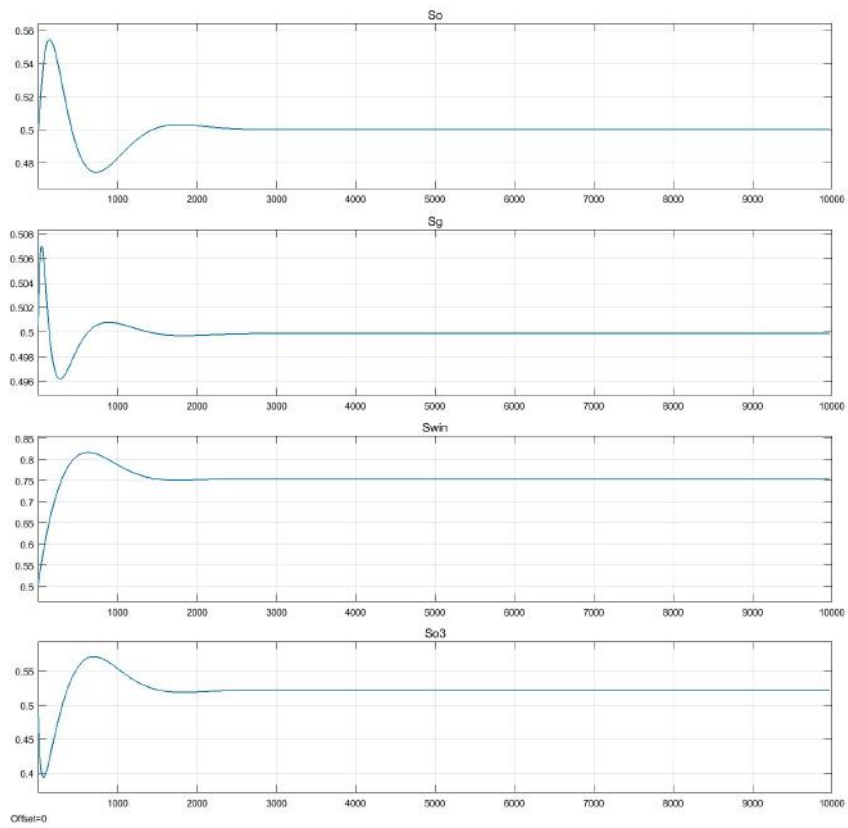


Figura 4.21: Variáveis controladas após falha aberta na válvula de óleo PDC S_{o2} .

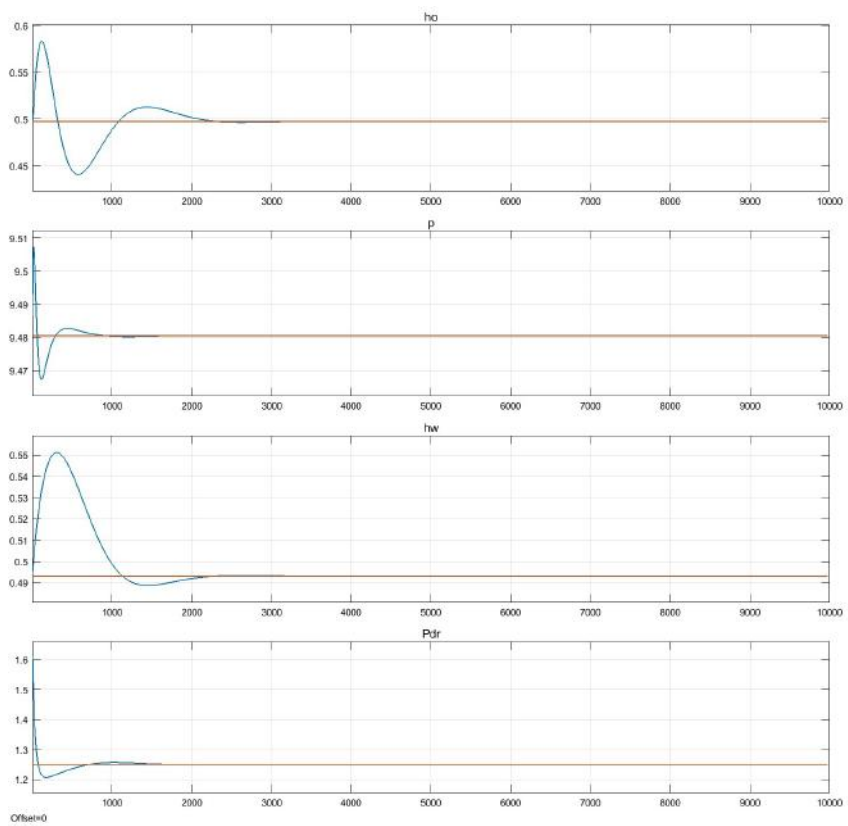


Figura 4.22: Variáveis manipuladas após falha aberta na válvula de óleo PDC S_{o2} .

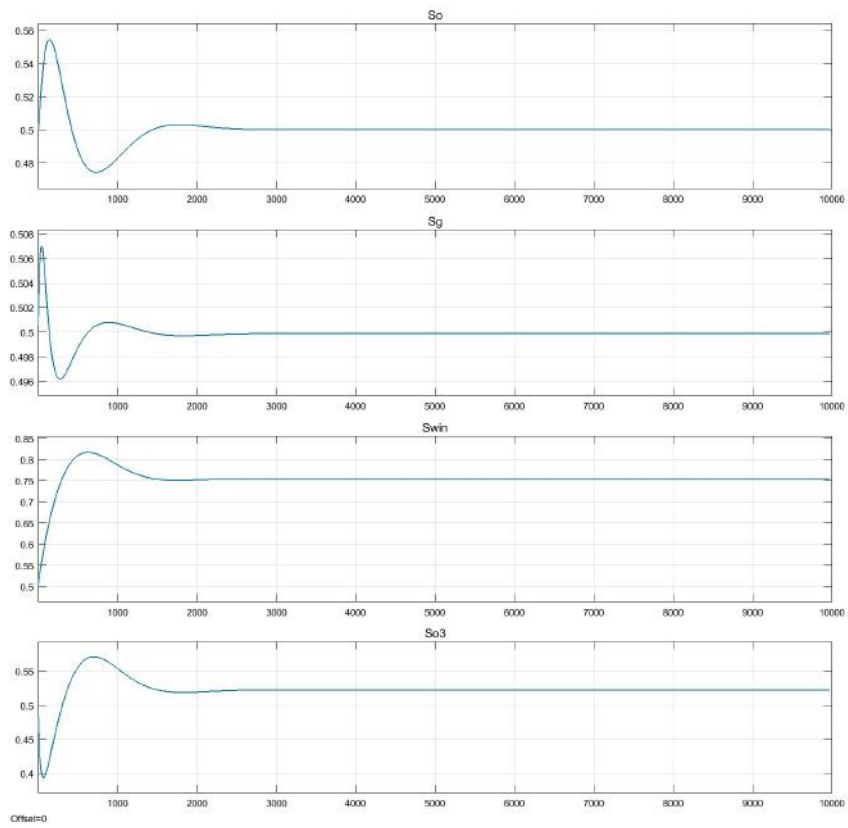


Figura 4.23: Variáveis controladas após falha fechada na válvula de óleo PDC S_{o2} .

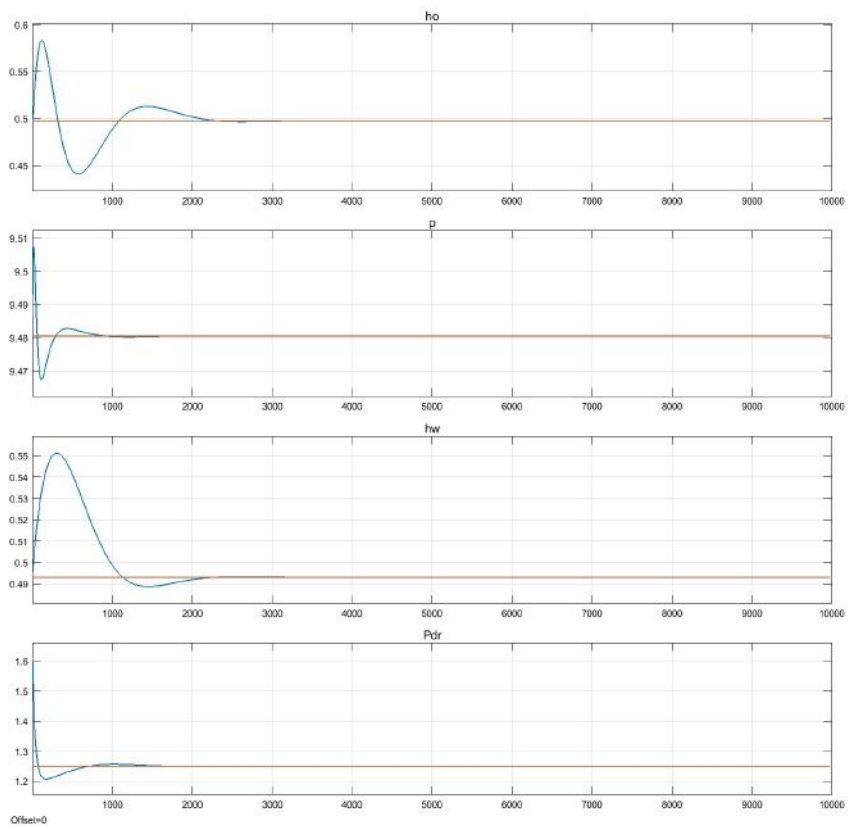


Figura 4.24: Variáveis manipuladas após falha fechada na válvula de óleo PDC S_{o2} .

Caso a válvula trave aberta, é possível notar na figura 4.26 que a razão Pdr , que representa a divisão entre as medições dos diferenciais de pressão entre as vazões do topo e do fundo do último hidrociclone da bateria (DC), satura no valor de 1,722, o que é devido à saturação da vazão W_{o3} no valor de $0,0004477m^3/s$, valor compatível com a equação 3.24 com valor de $S_{o3} = 1$. Pela na figura 4.25, observa-se que a saída de controle para a válvula de óleo neste hidrociclone tenta controlar o sistema com a saturação no valor zero pouco depois do tempo 5000s, ação de controle que não surte efeito no sistema.

Já a figura 4.27 mostra o controlador saturando no valor de abertura máxima da válvula, na tentativa de controlar o valor da razão que cai à zero, já que a vazão de óleo no hidrociclone DC se torna nula devido à falha completamente fechada em sua válvula de óleo, de acordo com a simulação e com a modelagem descrita pela equação 3.24.

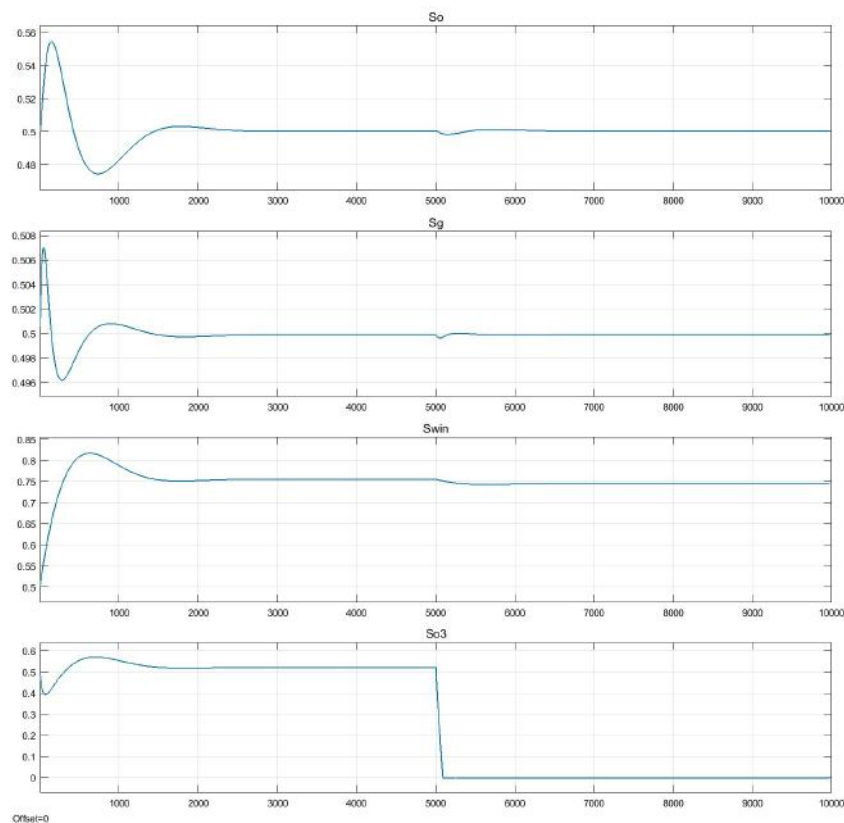


Figura 4.25: Variáveis controladas após falha aberta na válvula de óleo DC S_{o3} .

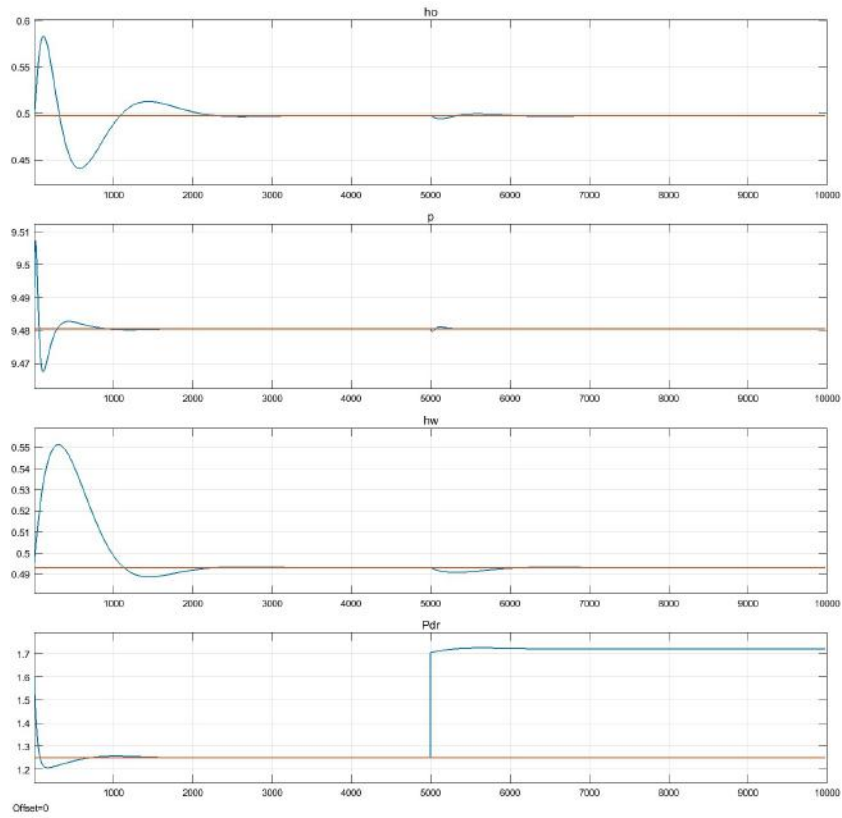


Figura 4.26: Variáveis manipuladas após falha aberta na válvula de óleo DC S_{o3} .

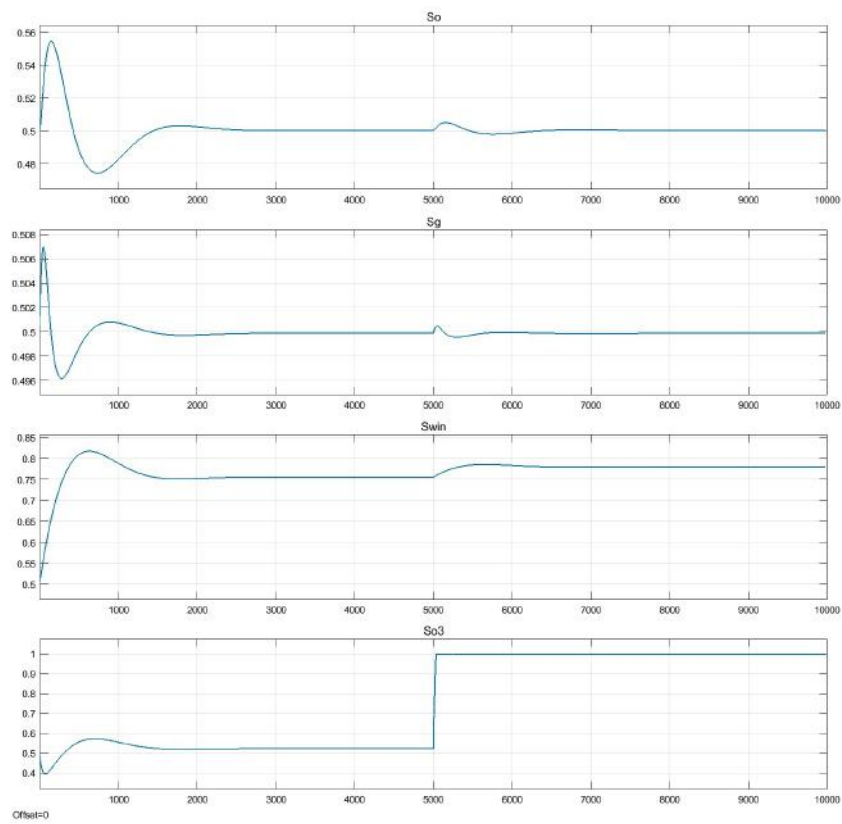


Figura 4.27: Variáveis controladas após falha fechada na válvula de óleo DC S_{o3} .

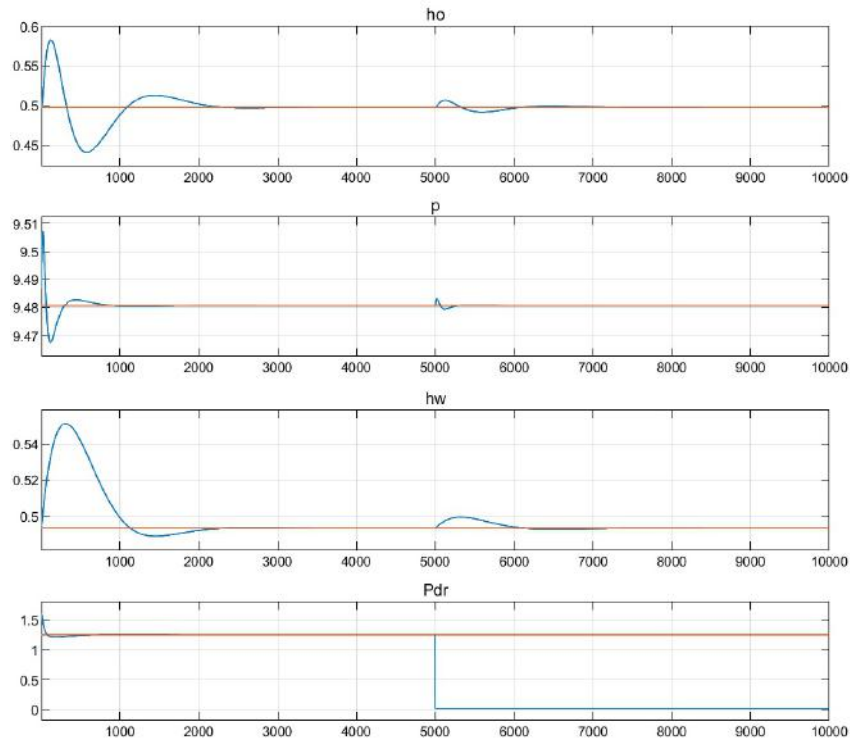


Figura 4.28: Variáveis manipuladas após falha fechada na válvula de óleo DC S_{o3} .

4.3.7 Falhas na válvula de gás

Por fim, as figuras de 4.29 a 4.32 tratam do comportamento do sistema quando ocorrem as falhas por agarramento nas posições totalmente aberta e totalmente fechada na válvula de gás do vaso separador.

Quando a simulação indica a falha completamente aberta na válvula de gás no tempo de 5000s, nota-se, pela figura 4.30 uma queda rápida da pressão p do sistema e saturação no valor zero. Pela figura 4.29, observa-se que a ação de controle nesta válvula S_g tenta rapidamente controlar o sistema ao enviar o sinal para fechar a válvula, o que não promove mudança no comportamento do sistema.

Enquanto pelas figuras 4.31 e 4.32, é notado o aumento abrupto da pressão do sistema até o valor máximo admissível, sem que o controlador seja capaz de controlar, mesmo com o valor da ação de controle correspondente à abertura máxima da válvula.

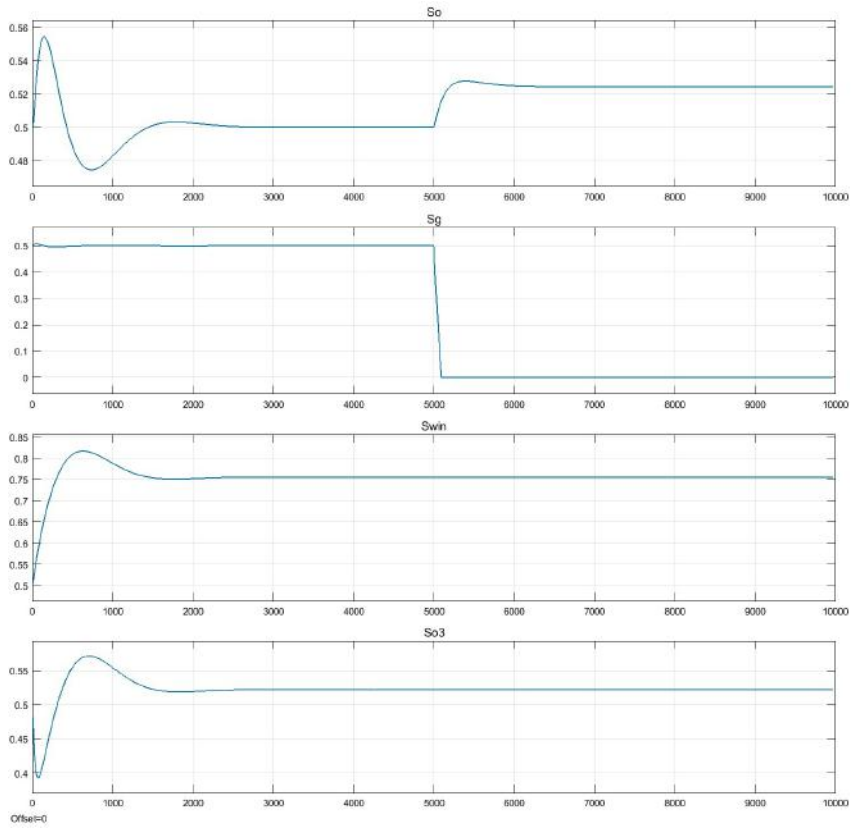


Figura 4.29: Variáveis controladas após falha aberta na válvula de gás S_g .

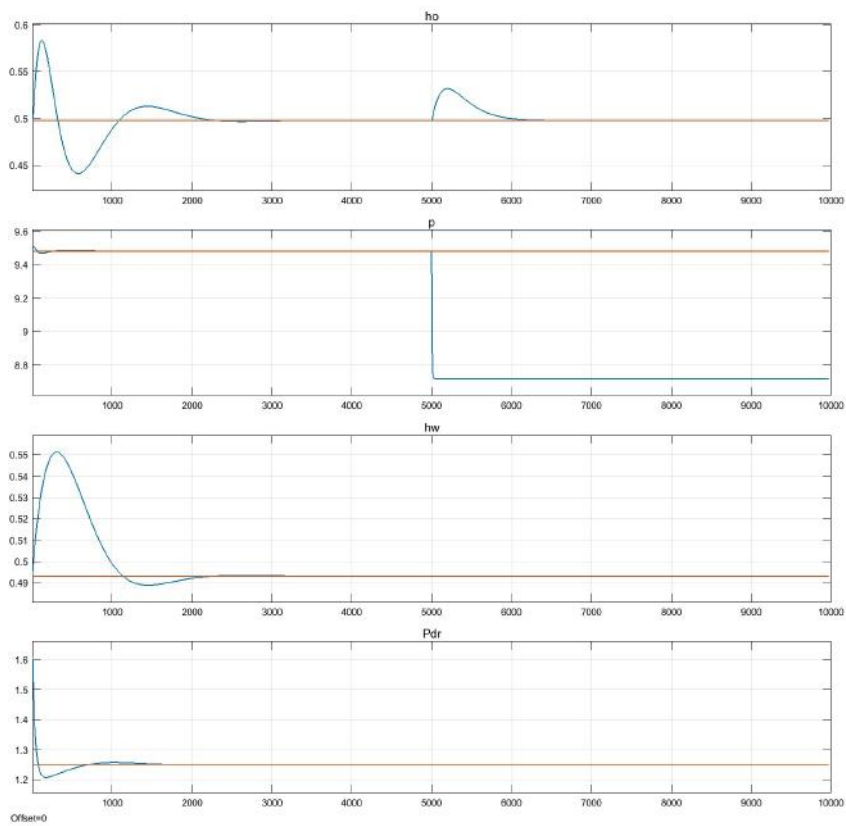


Figura 4.30: Variáveis manipuladas após falha aberta na válvula de " gás S_g .

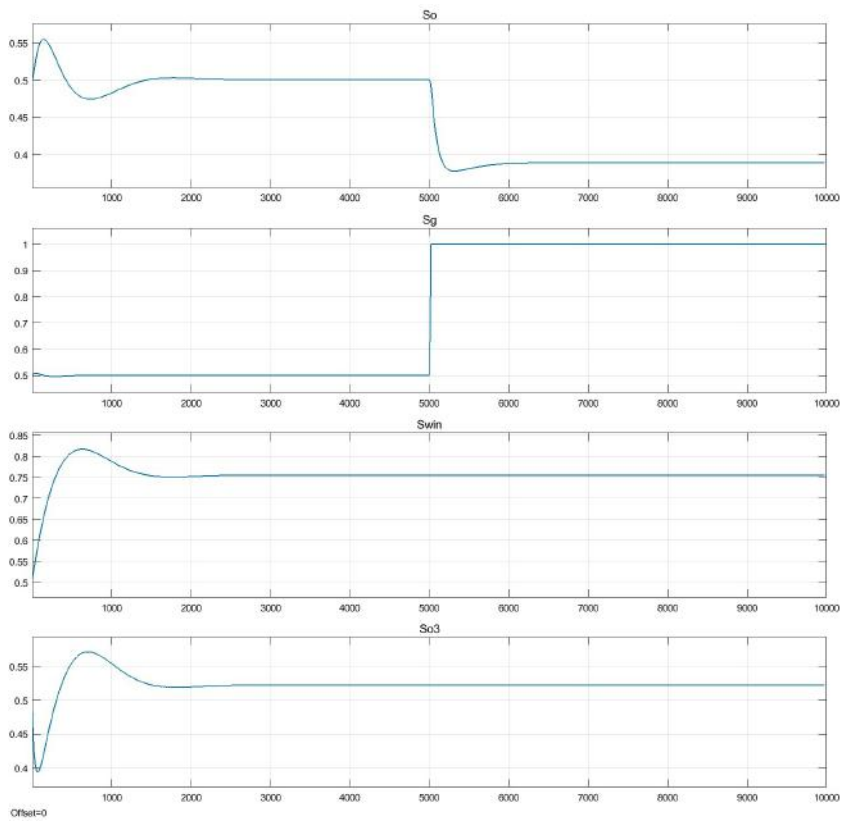


Figura 4.31: Variáveis controladas após falha fechada na válvula de gás S_g .

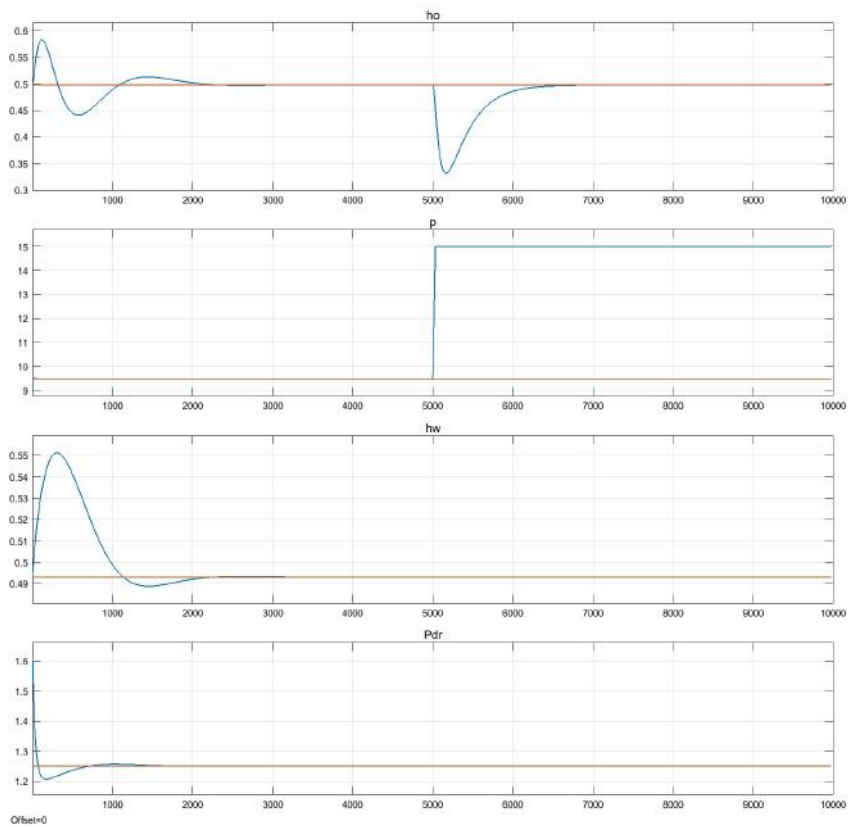


Figura 4.32: Variáveis manipuladas após falha fechada na válvula de gás S_g .

4.3.8 Avaliação dos diagnosticadores de falha

A avaliação dos diagnosticadores foi feita por meio de *scopes* para as saídas dos estados com rótulos Y dos diagnosticadores mostrados na figura 4.6. Desta forma, o sistema responde com o valor 1 assim que alcança um estado Y . Na figura 4.33, é mostrada a saída para o caso do diagnosticador de falha travada fechada na válvula de água (S_{w3}). É possível observar que o sistema foi capaz de diagnosticar a falha após o tempo de 5000s do início da simulação. Os outros diagnosticadores obtiveram resposta análoga.

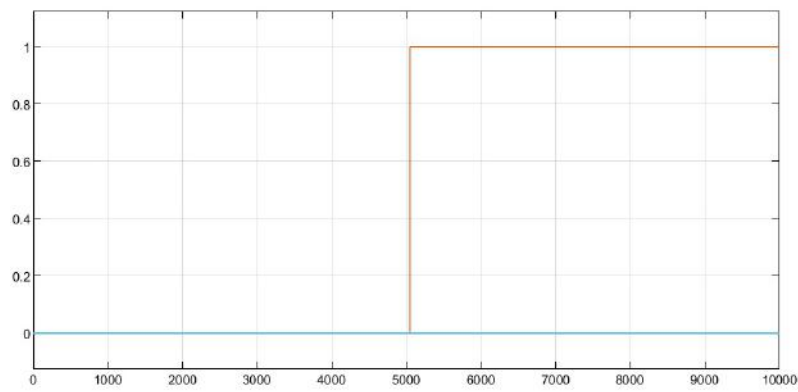


Figura 4.33: Diagnóstico de falha realizada através da simulação com *Stateflow*.

4.4 Conclusão do capítulo

Como conclusão, foi possível obter um modelo a eventos discretos para o diagnóstico das falhas capaz de diagnosticar as falhas nas válvulas. O modelo foi testado com a simulação e analisado na seção 4.3.

Capítulo 5

Conclusões e trabalhos futuros

A relevância do presente trabalho se dá por acrescentar um estudo de caso sobre a validação de um diagnosticador de falhas através da teoria de sistemas a eventos discretos aplicada a um modelo de processo industrial real com característica contínua no tempo.

O sistema de diagnose de falha proposto teve como objetivo ser parte de um sistema inteligente que fornece suporte operacional ao tratamento primário de processos de plataforma de petróleo *offshore*. O modelo de processo adotado foi baseado em um sistema real de plataforma de produção da Petrobras que opera no Campo de Marlim na Bacia de Campos. A partir do modelo contínuo foi possível obter um modelo a eventos discretos capaz de diagnosticar as falhas nas válvulas do sistema de controle. As ideias para trabalhos futuros são:

1. Analisar a necessidade de usar *sensor mapping* para assegurar a diagnosticabilidade.
2. Propor alternativas ao uso do mapeamento de sensores.
3. Considerar as falhas como intermitentes introduzindo-se um evento não-observável r_e que levaria a válvula ao estado inicial.
4. Implementação de um sistema de gerenciamento de alarmes, com prioridades de acordo com a falha, de forma automática, de forma a auxiliar o operador.

Referências Bibliográficas

- [1] CASSANDRAS, C. G., LAFORTUNE, S., *Introduction to Discrete Event System*. Springer-Verlag New York, Inc.: Secaucus, NJ, 2008.
- [2] PETROBRAS, “Processamento Primário de Petróleo”, Escola de Ciências e Tecnologia E&P, Universidade Petrobras.
- [3] VENKATASUBRAMANIAN, V., RENGASWAMY, R., YIN, K., et al., “A review of process fault detection and diagnosis Part I: Quantitative model-based methods”, *Computers and Chemical Engineering*, v. 27, pp. 293–311, 2003).
- [4] SINTEF Industrial Management, *Offshore Reliability Data Handbook (OREDA)*, 4th ed., 2002.
- [5] ZAYTOON, J., LAFORTUNE, S., “Overview of fault diagnosis methods for Discrete Event Systems”, *Annual Reviews in Control*, v. 37, pp. 308–320, 2013.
- [6] SAMPATH, M., SENGUPTA, R., LAFORTUNE, S., et al., “Failure Diagnosis using Discrete-Event Models”, *IEEE Transactions on Automatic Control*, v. 4, pp. 105–124, 1996.
- [7] RAMADGE, P. J. G., WONHAM, W. M., “The Control of Discrete Event Systems”, *Proceedings of the IEEE*, v. 77, pp. 81–98, 1989.
- [8] VIANA, G. D. S., *Diagnose de Falhas de Sistemas a Eventos Discretos com Transições Ponderadas*, Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, 2014.
- [9] VIANA, G. S., BASILIO, J. C., MOREIRA, M. V., “Computation of the maximum time for failure diagnosis of discrete-event systems”, *American Control Conference*, v. 1, 2015).
- [10] BASILIO, J. C., CARVALHO, L. K., MOREIRA, M. V., “Diagnose de falhas em sistemas a eventos discretos modelados por autômatos finitos”, *Revista Controle & Automação*, v. 21, pp. 510–533, 9 2010.

- [11] CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L., et al., *Introduction to Algorithms*. 3rd ed. The MIT Press, 2011.
- [12] CLAVIJO, L. B., BASILIO, J. C., CARVALHO, L. K., “DESLAB: A scientific computing program for analysis and synthesis of discrete-event systems”, *Workshop Series on Discrete Event Systems*, v. 77, pp. 349–355, 2012.
- [13] CLAVIJO, L. E. B., *Aspectos computacionais associados à implementação de algoritmos para sistemas a eventos discretos*, Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, 2014.
- [14] COUTINHO, L. E. A. A., “Um Tutorial para o Programa Científico Computacional DESLab.” Projeto de Graduação, Universidade Federal do Rio de Janeiro, 2014.
- [15] CONSELHO NACIONAL DO MEIO AMBIENTE, “Resolução nº 393/07: Dispõe sobre o descarte contínuo de água de processo ou de produção em plataformas marítimas de petróleo e gás natural, e dá outras providências.” Publicada no Diário Oficial da União em 09/08/2007, 2007.
- [16] NUNES, G. C., *Modelagem e simulação dinâmica de separador trifásico água-óleo-gás*, Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, 1994.
- [17] NUNES, G. C., “Modelagem dinâmica do processamento primário de petróleo: conceitos fundamentais e aplicação em controle de processos”, *Boletim técnico da Produção de Petróleo*, v. 2, pp. 29–47, 2007.
- [18] FILGUEIRAS, N. G. T., *Modelagem, Análise e Controle de um Processo de Separação Óleo/Água*, Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, 2005.
- [19] NUNES, C. E. V., *Sistema Inteligente de Suporte Operacional em Processos de Tratamento Primário de Petróleo*, Dissertação de Mestrado, Universidade Federal de Sergipe, 2012.
- [20] MORAES, C. A. C., *Modelo fluidodinâmico para a estimativa de eficiência em hidrociclone para águas oleosas*, Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, 1994.
- [21] SILVEIRA, M. A. C. R., *Controle de um processo de tratamento primário de petróleo*, Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, 2006.

- [22] Masoneilan International, Inc., *Masoneilan Handbook for Control Valve Sizing*, 5th ed., 1975.
- [23] Process Group Pty Ltd, *Hydrocyclones Deoiling*, 2012, Brochure.
- [24] CAMPOS, M. C. M. M., TEIXEIRA, H. C. G., *Controles típicos de equipamentos e processos industriais*. Edgar Blücher: São Paulo, 2006.
- [25] ÅSTRÖM, K. J., HÄGGLUNG, T., *PID controllers: theory, design, and tuning*. 2nd ed. Instrument Society of America: North Carolina, 1994.
- [26] SEBORG, D. E., MELLICHAMP, D. A., EDGAR, T. F., et al., *Process Dynamics and Control*. 3rd ed. John Wiley & Sons, Inc, 2011.
- [27] The MathWorks, Inc, *Stateflow API*, 2015, R2015a.
- [28] OGATA, K., *Discrete-time control systems*. 2nd ed. Prentice Hall: New Jersey, 1994.
- [29] HUANG, Z., CHANDRA, V., JIANG, S., et al., “Modeling Discrete Event Systems with Faults using a Rules based Modeling Formalism”, *Mathematical Modelling of Systems*, v. 1, pp. 396–401, 1996).

Apêndice A

Código Fonte

Códigos fontes utilizados no software deslab para verificação e diagnosticabilidade do modelo em SED.

A.1 Código Python com *DESLAB*

A subseção A.1.1 contém trechos do código da *toolbox* “*diagnosis.py*” com o algoritmo para a construção do diagnosticador. As subseções A.1.2 e A.1.3, contêm o código do arquivo criado com o modelo e as operações dos autômatos usados no capítulo 4.

A.1.1 *Toolbox* para autômato diagnosticador

```
12 def labelfailaut(sf):
13     """
14     This function creates a label failure automaton in
15     the context of failure diagnosis
16     """
17     # first we check if input parameters were correctly passed
18     if isinstance(sf, list):
19         sf = sf[0]
20     elif isinstance(sf, set):
21         sf = list(sf)[0]
22     elif isinstance(sf, frozenset):
23         sf = list(sf)[0]
24     elif isinstance(sf, str):
25         pass
26     else:
27         raise invalidArgument, 'sf must be string, list or set'
28
29     Y,N = syms('Y N')
30     sigmaA = [sf]
31     XA = [Y,N]
32     XOA = [N]
33     XmA = []
34     T = [(N,sf,Y),(Y,sf,Y)]
35     A1=fsa(XA,sigmaA,T,XOA,XmA, Sigobs=EMPTYSET, name='A1')
36     return A1
```

```

91 def diagnoser(G, sf, Sigma_o=UNDEFINED):
92     """Function that builds and renders an event Diagnoser
93     based on theoretical concepts of Sampath and Lafortune"""
94     # first we check if input parameters were correctly passed
95     if isinstance(sf, list):
96         sf = sf[0]
97     elif isinstance(sf, set):
98         sf = list(sf)[0]
99     elif isinstance(sf, frozenset):
100         sf = list(sf)[0]
101     elif isinstance(sf, str):
102         pass
103     else:
104         raise invalidArgument, 'sf must be string, list or set'
105     # we can use the inner observable events for diagnoser computation
106     # or the set of observable events passed by the user
107     if (Sigma_o == UNDEFINED) : # Sigma_o was not defined
108         Sigma_obs = G.Sigobs
109     elif isinstance(Sigma_o, list) | isinstance(Sigma_o, set):
110         Sigma_obs = frozenset(Sigma_o)
111     elif isinstance(Sigma_o, frozenset):
112         Sigma_obs = Sigma_o
113     else:
114         raise invalidArgument, 'Observed set must be of type set or list'
115     # we render the automaton using accepted notation
116     def latex(D):
117         tex_tab = []
118         for xd in D:
119             xd_tex = []
120             for x,l in xd:
121                 xd_tex.append(str(x)+l)
122             xd_tex = reduce(lambda i,j: i+ ','+j, (t for t in xd_tex))
123             xd_tex = r'\{' + xd_tex + r'\}'
124             tex_tab.append((xd, xd_tex))
125         return dict(tex_tab)
126     #label failure automaton
127     A = labelfailaut(sf)
128     D = observer(G//A, Sigma_obs)
129     D = D.setpar(name = 'Diagnoser', table = latex(D))
130     D.setgraphic('diagnoser')
131     return D

```

```

135 def is_diagnosable_diag(G, sf, Sigma_o= UNDEFINED ):
136     """This function looks for indeterminate states, i. e.
137     states that belong to indeterminate SCCs
138     """
139     if isinstance(sf, list):
140         sf = sf[0]
141     elif isinstance(sf, set):
142         sf = list(sf)[0]
143     elif isinstance(sf, frozenset):
144         sf = list(sf)[0]
145     elif isinstance(sf, str):
146         pass
147     else:

```

```

148         raise invalidArgument, 'sf must be string, list or set'
149     # we can use the inner observable events for diagnoser computation
150     # or the set of observable events passed by the user
151     if (Sigma_o == UNDEFINED) : # Sigma_o was not defined
152         Sigma_obs = G.Sigobs
153     elif isinstance(Sigma_o,list) | isinstance(Sigma_o,set):
154         Sigma_obs = frozenset(Sigma_o)
155     elif isinstance(Sigma_o, frozenset):
156         Sigma_obs = Sigma_o
157     else:
158         raise invalidArgument, 'Observed set must be of type set or list'
159     # Calculating the labeling plant
160     A=labelfailaut(sf)
161     G1 = G//A # G_l
162     # Calculating diagnoser
163     Gd = diagnoser(G, sf, Sigma_obs)
164     Gt= Gd//G1
165     SCC = strconncomps(Gt)
166     SCC_selfloop = [[1] for l in selfloopnodes(Gt)]
167     SCC_multiple = [C for C in SCC if (len(C)>1)]
168     SCC_nontrivial = SCC_selfloop + SCC_multiple
169     diagnosable = True
170     for C in SCC_nontrivial:
171         if any(l=='Y' and YN_type(xd)=='YN' for xd, (x,l) in C):
172             diagnosable = False
173             break
174     return diagnosable, Gt

```

A.1.2 Composição paralela para o modelo da Planta

```

1  # -*- coding: utf-8 -*-
2  # DES model of Controller and Valve from Three-phase Separator process
3  from deslab import *
4
5  syms('VI VPOC VC VO VSC VSO ocp ct ot so sc CI CSP CUR CUD CDD CDR nsp nur nud ndd ndr ctndd
6  ↔ otnur re')
7  table = [(VI, 'V_I'), (VPOC, 'V_{POC}'), (VC, 'V_C'), (VO, 'V_O'), (VSC, 'V_{SC}'), (VSO, 'V_{SO}'), (ocp, 'c_
8  ↔ {ocp}'), (ct, 'c_t'), (ot, 'o_t'), (so, 's_o'), (sc, 's_c')]
9
10 ## Value Automaton
11 Xv = [VI, VPOC, VC, VO, VSC, VSO]
12 Sigmav = [ocp, ct, ot, so, sc, re]
13 Sigma_ov = [ocp, ot, ct, re]
14 XOv = [VI]
15 Xmv = []
16 Tv = [(VI, ocp, VPOC), (VI, ot, VO), (VI, ct, VC), (VI, sc, VSC), (VI, so, VSO), (VPOC, ocp, VPOC), (VPOC, ot, VO), (VP
17 ↔ OC, ct, VC), (VPOC, so, VSO), (VPOC, sc, VSC), (VO, ot, VO), (VO, ocp, VPOC), (VO, ct, VC), (VO, so, VSO), (VO, sc,
18 ↔ VSC), (VC, ct, VC), (VC, ocp, VPOC), (VC, ot, VO), (VC, sc, VSC), (VC, so, VSO), (VSO, ct, VSO), (VSO, ot, VSO), (V
19 ↔ SO, ocp, VSO), (VSC, ct, VSC), (VSC, ot, VSC), (VSC, ocp, VSC), (VPOC, re, VI), (VC, re, VI), (VO, re, VI)]
20 V = fsa(Xv, Sigmav, Tv, XOv, Xmv, table, name='Valve', Sigobs = Sigma_ov )
21
22 ## Controller Automaton
23 Xc = [CI, CSP, CUR, CUD, CDD, CDR]
24 Sigmac = [ocp, ct, ot, nsp, nur, nud, ndd, ndr, re]
25 XOc = [CI]

```

```

21 Xmc = []
22 Tc = [(CI,nsp,CSP), (CI,nur,CUR), (CI,nud,CUD), (CI,ndd,CDD), (CI,ndr,CDR), (CSP,nsp,CSP), (CSP,nur,CUR) ]
↪ , (CSP,ndd,CDD), (CSP,re,CI), (CUR,nsp,CSP), (CUR,nud,CUD), (CUR,ocp,CUR), (CUR,ot,CUR), (CUR,re,CI) ]
↪ , (CUD,nsp,CSP), (CUD,nur,CUR), (CUD,ndd,CDD), (CUD,ocp,CUD), (CUD,ot,CUD), (CUD,re,CI), (CDD,nsp,CS
↪ P), (CDD,ndr,CDR), (CDD,ocp,CDD), (CDD,ct,CDD), (CDD,re,CI), (CDR,nsp,CSP), (CDR,nur,CUR), (CDR,ndd,
↪ CDD), (CDR,ocp,CDR), (CDR,ct,CDR), (CDR,re,CI)]
23 Gc = fsa(Xc,Sigmac,Tc,X0c,Xmc,table,name='Controller')
24 draw(V,Gc,'figure')
25
26 ## Plant Automaton
27 Gp = V//Gc
28 Gp.name = 'Plant'
29 print(Gp)

```

A.1.3 Autômato diagnosticador da Planta através da busca por componentes fortemente conexos

```

30 def marc_y_gl(Gl):
31     Gly = Gl.copy()
32     marc=frozenset([])
33     for x in Gly.X:
34         if 'Y' == x[1]:
35             xmnew= x
36             marc=marc|frozenset([xmnew])
37     Gly.Xm = marc
38     return Gly
39
40 def marc_n_gd(Gd):
41     Gdy = Gd.copy()
42     marc=frozenset([])
43     for x in Gdy.X:
44         for el in x:
45             if 'N' == el[1]:
46                 marc=marc|frozenset([x])
47     Gdy.Xm = marc
48     return Gdy
49
50 A1 = labelfailaut(sc)
51 G11 = Gp // A1
52 G11m = marc_y_gl(G11)
53 Gd1 = diagnoser(Gp,sc,Gp.Sigobs)
54
55 Gd1m= marc_n_gd(Gd1)
56 Gt1 = Gd1//G11
57 Gt1m = Gd1m//G11m
58 G11.setgraphic(style='observer',direction = 'UD')
59 Gd1.setgraphic(style='observer',direction = 'UD')
60 Gt1m.setgraphic(style='observer',direction = 'UD')
61
62 SCC = strconncomps(Gt1m.Graph)
63 C_1 = selfloopnodes(Gt1m.Graph)
64 SCC_NT = [C for C in SCC if (len(C)>1) | (C in C_1)]
65 diagnosable = True
66 estadosSCC_NT = frozenset([])

```

```

67 for x in SCC_NT:
68     for estado in x:
69         estadosSCC_NT=estadosSCC_NT|frozenset([estado])
70 if (estadosSCC_NT & Gt1m.Xm) <> frozenset([]):
71     diagnosable = False
72 print 'DIAGNOSTICABILIDADE= '
73 print diagnosable
74
75 Gt1mfinal = Gt1m.setpar(Xm = estadosSCC_NT & Gt1m.Xm)
76 Gt1mfinal.setgraphic(style='observer',direction = 'UD')
77
78 SCCfinal = strconncomps(Gt1mfinal.Graph)
79 print(SCCfinal)

```

A.1.4 Mapa de sensores

```

80 Gpuo = sorted(Gp.Sigma-Gp.Sigobs)
81 Gpx=(sorted(Gp.X))
82 len(Gpx)
83 for i in range(0,len(Gpx)):
84     print(Gpx[i],i)      #See Vsc and Vso non-obervable states range (from 21 to 32)
85
86 #List all transitions and count
87 Gps = sorted(Gp.Sigma)
88 t=0
89 for i in range(0,len(Gpx)):    #All states
90     for j in range(0,len(Gps)): #All events
91         if Gp.delta(Gpx[i],Gps[j]) <> None:
92             t+=1
93         print(Gpx[i],(i+1), Gps[j], Gp.delta(Gpx[i],Gps[j]),t)
94
95 ## Sensor Map
96 #Deleting transitions from unobservable events
97 DT = list()    #Create list of Gp deleted transitions
98 DTi = list()  #Create list of inicial states from Gp deleted transitions
99 DTe = list()  #Create list of events from Gp deleted transitions
100 DTf = list()  #Create list of final states from Gp deleted transitions
101 a=0
102 for i in range(0,len(Gpx)):
103     for j in range(0,len(Gpuo)):    #All unobservable events
104         for k in range(21,len(Gpx)): #All states with Vso and Vsc
105             if Gp.delta(Gpx[i],Gpuo[j])==Gpx[k]:
106                 T = (Gpx[i],Gpuo[j],Gpx[k])
107                 DT.insert(a,T)
108                 DTi.insert(a,Gpx[i])
109                 DTe.insert(a,Gpuo[j])
110                 DTf.insert(a,Gpx[k])
111                 a+=1
112                 print(T,a)
113                 Gp = Gp.deletetransition(T)
114                 len(DT)
115                 print(Gp) #number of transitions 198-42= 156? len(DT) = 42
116
117 #New states
118 Xsm = list()

```

```

119 for i in range(0,len(DTf)):
120     #Xsm.insert(i,(DTf[i],i))
121     Xsm.insert(i,(i+1))
122     print(Xsm[i],(i+1),DTf[i])
123
124     #Add unobserved events transitions to new states
125     for i in range(0,len(Xsm)):
126         Tuo = (DTi[i],DTe[i],Xsm[i])
127         print(Tuo,i)
128         Gp = Gp.addtransition(Tuo) #addtransition also adds new states
129         print(Gp) #number of transitions 156+42 = 198? number of states 33+42 = 75?
130
131     #Sensor map from new states to fault states
132     for i in range(0,len(Xsm)):
133         if DTe[i] == 'so':
134             Tso = (Xsm[i],'ctnadd',DTf[i])
135             print(Tso,'Tso',i)
136             Gp = Gp.addtransition(Tso) #addtransition also adds new events
137         else: #if DTe[i] == 'sc':
138             Tsc = (Xsm[i],'otnur',DTf[i])
139             print(Tsc,'Tsc',i)
140             Gp = Gp.addtransition(Tsc)
141
142     print(Gp) #number of transitions 198+42 = 240? number of events 11+2 = 13?
143     print(sorted(Gp.Sigma))
144     print(sorted(Gp.Sigobs))

```

A.1.5 Diagnosticador do modelo após mapa de sensores

```

145 ## Is Gp with Sensor Map diagnosable? (See file ~/deslab/toolboxes/diagnosis.py)
146 DiagSCC_so = is_diagnosable_diag(Gp, so)
147 print(Verscc_so[0]) #True or False
148 print(Verscc_so[1])
149 Diag_so = diagnoser(Gp, so)
150
151 Verscc_sc = is_diagnosable_diag(Gp, sc)
152 print(Verscc_sc[0]) #True or False
153 print(Verscc_sc[1])
154 Diag_sc = diagnoser(Gp, sc)
155
156 sizeso = autom2sf(Diag_so)
157 sizesc = autom2sf(Diag_sc)

```

A.1.6 *Toolbox* para exportar autômatos do *DESLAB* ao *Stateflow*

```

1 # coding=utf-8
2 from deslab import *
3 import os
4 from sets import Set
5

```



```

6  """This is the toolbox for creating Stateflow chart to an automaton. It creates .m file at
↪  "/home/controlle/Documentos/sf_API".
7  """
8
9  #see file automatadefs.py
10 def autom2sf(G):
11     """Function that builds Stateflow chart from an automaton"""
12     # Create Stateflow API .m file
13     if not os.path.isdir("./home/controlle/Documentos/sf_API"):
14         os.mkdir("./home/controlle/Documentos/sf_API")
15         os.chdir("./home/controlle/Documentos/sf_API")
16         os.getcwd()
17         sfo = open("autom2sf_"+G.name+".m",'w+')
18
19     # Writing API code to create new simulink model and create sf chart
20     sfo.write('%\t<!> Close all simulink models <!>\r\nsfnew;\r\nrt = sfroot;\r\nnm =
↪  rt.find(\'-isa\',\'Stateflow.Machine\');\r\nchart =
↪  m.find(\'-isa\',\'Stateflow.Chart\');\r\nchart.set(\'Name\',\'DESdiag\');\t% Set name of the
↪  Stateflow block\r\n\r\n%\tCreate states\r\n')
21
22     # Create Stateflow states
23     # Create a set of states that are reached from observable events
24     GXob = Set(G.XO) #Insert initial states (they may never be reached)
25     for x in G.X:
26         for eo in G.Sigobs:
27             if G.delta(x,eo): #Condition for delta function is not None
28                 if x not in G.delta(x,eo): #Avoid unreachable self-loop state
29                     if len(G.delta(x,eo))> 1: #Is it not a frozenset? print '>'
30                         GXob.add(G.delta(x,eo))
31                 else: #It is a frozenset! print iter(G.delta(x,eo)).next()
32                     GXob.add(next(iter(G.delta(x,eo)))) #iter(G.delta(x,eo)).next()
33
34     GXo = list(GXob) #Create unsorted list of observable states
35
36     #Sort the initial states to list beginning
37     a = 0
38     for i in range(0,len(GXo)):
39         if GXo[i] in G.XO:
40             GXo[a],GXo[i] = GXo[i],GXo[a]
41             a+=1
42
43     #Write API code to create sf states
44     for i in range(0,len(GXo)):
45         p = 80+i*120
46         sfo.write(''.join(GXo[i])+'.Name =
↪  \''.join(GXo[i])+\';\r\n'+'.Position = [%s 80 90 80];\r\n'%p)
47         sfo.write(''.join(GXo[i])+'.HasOutputData = true;\r\n')
48
49     for i in range(0,len(G.XO)): #Initial states description:
50         sfo.write(''.join(GXo[i])+'.Description = \'Initial state\';\r\n')
51
52     #Marked States description and output
53     for xm in G.Xm:
54         if xm in G.XO:
55             sfo.write(''.join(xm)+'.Description = \'Initial and Marked state\';\r\n')
56         elif xm in GXob:
57             sfo.write(''.join(xm)+'.Description = \'Marked state\';\r\n')
58

```

```

59 #Create transitions and label it to listen for events
60 sfo.write('\r\n%\tCreate transitions and label it to listen for events\r\n')
61 for xo in GXob:
62 for eo in G.Sigobs:
63 if G.delta(xo, eo):
64 if len(G.delta(xo, eo)) > 1: #Not a frozenset!
65 xod = G.delta(xo, eo)
66 else: #It is a frozenset!
67 xod = next(iter(G.delta(xo, eo))) #or iter(G.delta(xo, eo)).next()
68 sfo.write(''.join(xo)+''.join(eo)+' =
↳ Stateflow.Transition(chart);\r\n'+'.join(xo)+''.join(eo)+' .Source =
↳ '+''.join(xo)+';\r\n'+'.join(xo)+''.join(eo)+' .Destination =
↳ '+''.join(xod)+';\r\n'+'.join(xo)+''.join(eo)+' .LabelString = \''+''.join(eo)+'\';\r\n')
69 if xo == xod: #Self-loop
70 sfo.write(''.join(xo)+''.join(eo)+' .SourceOClock =
↳ 1;\r\n'+'.join(xo)+''.join(eo)+' .DestinationOClock =
↳ 2;\r\n'+'.join(xo)+''.join(eo)+' .MidPoint =
↳ ['+''.join(xo)+' .Position(1)+'+''.join(xo)+' .Position(3)
↳ '+''.join(xo)+' .Position(2)-10];\r\n'+'.join(xo)+''.join(eo)+' .LabelPosition=
↳ ['+''.join(xo)+' .Position(1)+'+''.join(xo)+' .Position(3) '+''.join(xo)+' .Position(2)-25 0 0
↳ ];\r\n')
71 else:
72 sfo.write(''.join(xo)+''.join(eo)+' .SourceOClock = 3;\r\n')
73
74 #Create events
75 sfo.write('\r\n%\tCreate events\r\n')
76 b=1
77 for eo in G.Sigobs:
78 sfo.write(''.join(eo)+' = Stateflow.Event(chart);\r\n'+'.join(eo)+' .Name =
↳ \''+''.join(eo)+'\';\r\n'+'.join(eo)+' .Scope = \'Input\';\r\n'+'.join(eo)+' .Port =
↳ %s;\r\n'%b)
79 b+=1
80
81 #Inicial transitions
82 sfo.write('\r\n%\tInicial transition\r\n')
83 for i in range(0, len(G.XO)):
84 sfo.write('dt%s = Stateflow.Transition(chart);\r\nndt%s.Destination = '%(i, i))
85 sfo.write(''.join(GXo[i])+';\r\nndt%s.DestinationOClock = 0;\r\nxsource = '%i)
86 sfo.write(''.join(GXo[i])+' .Position(1)+'+''.join(GXo[i])+' .Position(3)/2-10;\r\nysource =
↳ '+''.join(GXo[i])+' .Position(2)-20;\r\nndt%s.SourceEndPoint = [xsource ysource];\r\n%i)
87
88 sfo.write('sfsave(m.Id, \'DESmodel\');\r\n\r\n% Script to add Input Block for Stateflow
↳ simulation\r\nsimulink;\t% Open Simulink Library\r\nadd_block(\'simulink/Sinks/To
↳ Workspace\', \'DESmodel/output\');\r\nset_param(\'DESmodel/output\', \'Position\', [170 30 240
↳ 55]);\t% Set Position\r\n% Add Routing Mux\r\nadd_block(\'simulink/Signal
↳ Routing/Mux\', \'DESmodel/route\');\r\nset_param(\'DESmodel/route\', \'Position\', [140, 26,
↳ 145, 64]);\t%Set Position\r\n')
89
90 #Outputs
91 sfo.write('set_param(\'DESmodel/route\', \'Inputs\', \'\')
92 sfo.write(str(len(GXo)))
93 sfo.write('\');\t% Number of input ports\r\n% Add lines between blocks\r\n% Add lines from
↳ Stateflow to routing Mux\r\n')
94 for i in range(1, len(GXo)+1):
95 sfo.write('add_line(\'DESmodel\', \'DESdiag/%s\', \'route/%s\');\r\n' %(i, i))
96 sfo.write('% Add line from route to output\r\nadd_line(\'DESmodel\', \'route/1\', \'output/1\');\r\n')
↳ nsfsave(\'DESmodel\', \'DESmodel\');')
97

```

```

98 position = sfo.tell()
99 sfo.close()
100
101 return position

```

A.2 Arquivo *Matlab* que gera *Stateflow* automaticamente - exemplo criado com a *toolbox* da seção A.1.6

```

1  %      <!-- Close all simulink models -->
2  sfnew;
3  rt = sfroot;
4  m = rt.find('-isa','Stateflow.Machine');
5  chart = m.find('-isa','Stateflow.Chart');
6  chart.set('Name','DESdiag');      % Set name of the Stateflow block
7
8  %      Create states
9  X_VI = Stateflow.State(chart);
10 X_VI.Name = 'X_VI';
11 X_VI.Position = [80 80 90 80];
12 X_VI.HasOutputData = true;
13 X_VC = Stateflow.State(chart);
14 X_VC.Name = 'X_VC';
15 X_VC.Position = [200 80 90 80];
16 X_VC.HasOutputData = true;
17 X_VPOC = Stateflow.State(chart);
18 X_VPOC.Name = 'X_VPOC';
19 X_VPOC.Position = [320 80 90 80];
20 X_VPOC.HasOutputData = true;
21 X_VO = Stateflow.State(chart);
22 X_VO.Name = 'X_VO';
23 X_VO.Position = [440 80 90 80];
24 X_VO.HasOutputData = true;
25 X_VSC = Stateflow.State(chart);
26 X_VSC.Name = 'X_VSC';
27 X_VSC.Position = [560 80 90 80];
28 X_VSC.HasOutputData = true;
29 X_VSO = Stateflow.State(chart);
30 X_VSO.Name = 'X_VSO';
31 X_VSO.Position = [680 80 90 80];
32 X_VSO.HasOutputData = true;
33 X_VI.Description = 'Initial state';
34
35 %      Create transitions and label it to listen for events
36 X_VIso = Stateflow.Transition(chart);
37 X_VIso.Source = X_VI;
38 X_VIso.Destination = X_VSO;
39 X_VIso.LabelString = 'so';
40 X_VIso.SourceOClock = 3;
41 X_VIsc = Stateflow.Transition(chart);
42 X_VIsc.Source = X_VI;
43 X_VIsc.Destination = X_VSC;

```

```

44 X_VIsc.LabelString = 'sc';
45 X_VIsc.SourceOClock = 3;
46 X_VIocp = Stateflow.Transition(chart);
47 X_VIocp.Source = X_VI;
48 X_VIocp.Destination = X_VPOC;
49 X_VIocp.LabelString = 'ocp';
50 X_VIocp.SourceOClock = 3;
51 X_VIot = Stateflow.Transition(chart);
52 X_VIot.Source = X_VI;
53 X_VIot.Destination = X_VO;
54 X_VIot.LabelString = 'ot';
55 X_VIot.SourceOClock = 3;
56 X_VIct = Stateflow.Transition(chart);
57 X_VIct.Source = X_VI;
58 X_VIct.Destination = X_VC;
59 X_VIct.LabelString = 'ct';
60 X_VIct.SourceOClock = 3;
61 X_VCre = Stateflow.Transition(chart);
62 X_VCre.Source = X_VC;
63 X_VCre.Destination = X_VI;
64 X_VCre.LabelString = 're';
65 X_VCre.SourceOClock = 3;
66 X_VCso = Stateflow.Transition(chart);
67 X_VCso.Source = X_VC;
68 X_VCso.Destination = X_VSO;
69 X_VCso.LabelString = 'so';
70 X_VCso.SourceOClock = 3;
71 X_VCsc = Stateflow.Transition(chart);
72 X_VCsc.Source = X_VC;
73 X_VCsc.Destination = X_VSC;
74 X_VCsc.LabelString = 'sc';
75 X_VCsc.SourceOClock = 3;
76 X_VCocp = Stateflow.Transition(chart);
77 X_VCocp.Source = X_VC;
78 X_VCocp.Destination = X_VPOC;
79 X_VCocp.LabelString = 'ocp';
80 X_VCocp.SourceOClock = 3;
81 X_VCot = Stateflow.Transition(chart);
82 X_VCot.Source = X_VC;
83 X_VCot.Destination = X_VO;
84 X_VCot.LabelString = 'ot';
85 X_VCot.SourceOClock = 3;
86 X_VCct = Stateflow.Transition(chart);
87 X_VCct.Source = X_VC;
88 X_VCct.Destination = X_VC;
89 X_VCct.LabelString = 'ct';
90 X_VCct.SourceOClock = 1;
91 X_VCct.DestinationOClock = 2;
92 X_VCct.MidPoint = [X_VC.Position(1)+X_VC.Position(3) X_VC.Position(2)-10];
93 X_VCct.LabelPosition= [X_VC.Position(1)+X_VC.Position(3) X_VC.Position(2)-25 0 0 ];
94 X_VPOCcre = Stateflow.Transition(chart);
95 X_VPOCcre.Source = X_VPOC;
96 X_VPOCcre.Destination = X_VI;
97 X_VPOCcre.LabelString = 're';
98 X_VPOCcre.SourceOClock = 3;
99 X_VPOCso = Stateflow.Transition(chart);
100 X_VPOCso.Source = X_VPOC;
101 X_VPOCso.Destination = X_VSO;

```

```

102 X_VPOCso.LabelString = 'so';
103 X_VPOCso.SourceOClock = 3;
104 X_VPOCsc = Stateflow.Transition(chart);
105 X_VPOCsc.Source = X_VPOC;
106 X_VPOCsc.Destination = X_VSC;
107 X_VPOCsc.LabelString = 'sc';
108 X_VPOCsc.SourceOClock = 3;
109 X_VPOCocp = Stateflow.Transition(chart);
110 X_VPOCocp.Source = X_VPOC;
111 X_VPOCocp.Destination = X_VPOC;
112 X_VPOCocp.LabelString = 'ocp';
113 X_VPOCocp.SourceOClock = 1;
114 X_VPOCocp.DestinationOClock = 2;
115 X_VPOCocp.MidPoint = [X_VPOC.Position(1)+X_VPOC.Position(3) X_VPOC.Position(2)-10];
116 X_VPOCocp.LabelPosition= [X_VPOC.Position(1)+X_VPOC.Position(3) X_VPOC.Position(2)-25 0 0 ];
117 X_VPOCot = Stateflow.Transition(chart);
118 X_VPOCot.Source = X_VPOC;
119 X_VPOCot.Destination = X_VO;
120 X_VPOCot.LabelString = 'ot';
121 X_VPOCot.SourceOClock = 3;
122 X_VPOCct = Stateflow.Transition(chart);
123 X_VPOCct.Source = X_VPOC;
124 X_VPOCct.Destination = X_VC;
125 X_VPOCct.LabelString = 'ct';
126 X_VPOCct.SourceOClock = 3;
127 X_VOre = Stateflow.Transition(chart);
128 X_VOre.Source = X_VO;
129 X_VOre.Destination = X_VI;
130 X_VOre.LabelString = 're';
131 X_VOre.SourceOClock = 3;
132 X_VOso = Stateflow.Transition(chart);
133 X_VOso.Source = X_VO;
134 X_VOso.Destination = X_VSO;
135 X_VOso.LabelString = 'so';
136 X_VOso.SourceOClock = 3;
137 X_VOsc = Stateflow.Transition(chart);
138 X_VOsc.Source = X_VO;
139 X_VOsc.Destination = X_VSC;
140 X_VOsc.LabelString = 'sc';
141 X_VOsc.SourceOClock = 3;
142 X_VOocp = Stateflow.Transition(chart);
143 X_VOocp.Source = X_VO;
144 X_VOocp.Destination = X_VPOC;
145 X_VOocp.LabelString = 'ocp';
146 X_VOocp.SourceOClock = 3;
147 X_VOot = Stateflow.Transition(chart);
148 X_VOot.Source = X_VO;
149 X_VOot.Destination = X_VO;
150 X_VOot.LabelString = 'ot';
151 X_VOot.SourceOClock = 1;
152 X_VOot.DestinationOClock = 2;
153 X_VOot.MidPoint = [X_VO.Position(1)+X_VO.Position(3) X_VO.Position(2)-10];
154 X_VOot.LabelPosition= [X_VO.Position(1)+X_VO.Position(3) X_VO.Position(2)-25 0 0 ];
155 X_VOct = Stateflow.Transition(chart);
156 X_VOct.Source = X_VO;
157 X_VOct.Destination = X_VC;
158 X_VOct.LabelString = 'ct';
159 X_VOct.SourceOClock = 3;

```

```

160 X_VSCocp = Stateflow.Transition(chart);
161 X_VSCocp.Source = X_VSC;
162 X_VSCocp.Destination = X_VSC;
163 X_VSCocp.LabelString = 'ocp';
164 X_VSCocp.SourceOClock = 1;
165 X_VSCocp.DestinationOClock = 2;
166 X_VSCocp.MidPoint = [X_VSC.Position(1)+X_VSC.Position(3) X_VSC.Position(2)-10];
167 X_VSCocp.LabelPosition= [X_VSC.Position(1)+X_VSC.Position(3) X_VSC.Position(2)-25 0 0 ];
168 X_VSCot = Stateflow.Transition(chart);
169 X_VSCot.Source = X_VSC;
170 X_VSCot.Destination = X_VSC;
171 X_VSCot.LabelString = 'ot';
172 X_VSCot.SourceOClock = 1;
173 X_VSCot.DestinationOClock = 2;
174 X_VSCot.MidPoint = [X_VSC.Position(1)+X_VSC.Position(3) X_VSC.Position(2)-10];
175 X_VSCot.LabelPosition= [X_VSC.Position(1)+X_VSC.Position(3) X_VSC.Position(2)-25 0 0 ];
176 X_VSCct = Stateflow.Transition(chart);
177 X_VSCct.Source = X_VSC;
178 X_VSCct.Destination = X_VSC;
179 X_VSCct.LabelString = 'ct';
180 X_VSCct.SourceOClock = 1;
181 X_VSCct.DestinationOClock = 2;
182 X_VSCct.MidPoint = [X_VSC.Position(1)+X_VSC.Position(3) X_VSC.Position(2)-10];
183 X_VSCct.LabelPosition= [X_VSC.Position(1)+X_VSC.Position(3) X_VSC.Position(2)-25 0 0 ];
184 X_VSOocp = Stateflow.Transition(chart);
185 X_VSOocp.Source = X_VSO;
186 X_VSOocp.Destination = X_VSO;
187 X_VSOocp.LabelString = 'ocp';
188 X_VSOocp.SourceOClock = 1;
189 X_VSOocp.DestinationOClock = 2;
190 X_VSOocp.MidPoint = [X_VSO.Position(1)+X_VSO.Position(3) X_VSO.Position(2)-10];
191 X_VSOocp.LabelPosition= [X_VSO.Position(1)+X_VSO.Position(3) X_VSO.Position(2)-25 0 0 ];
192 X_VSOot = Stateflow.Transition(chart);
193 X_VSOot.Source = X_VSO;
194 X_VSOot.Destination = X_VSO;
195 X_VSOot.LabelString = 'ot';
196 X_VSOot.SourceOClock = 1;
197 X_VSOot.DestinationOClock = 2;
198 X_VSOot.MidPoint = [X_VSO.Position(1)+X_VSO.Position(3) X_VSO.Position(2)-10];
199 X_VSOot.LabelPosition= [X_VSO.Position(1)+X_VSO.Position(3) X_VSO.Position(2)-25 0 0 ];
200 X_VSOct = Stateflow.Transition(chart);
201 X_VSOct.Source = X_VSO;
202 X_VSOct.Destination = X_VSO;
203 X_VSOct.LabelString = 'ct';
204 X_VSOct.SourceOClock = 1;
205 X_VSOct.DestinationOClock = 2;
206 X_VSOct.MidPoint = [X_VSO.Position(1)+X_VSO.Position(3) X_VSO.Position(2)-10];
207 X_VSOct.LabelPosition= [X_VSO.Position(1)+X_VSO.Position(3) X_VSO.Position(2)-25 0 0 ];
208
209 % Create events
210 re = Stateflow.Event(chart);
211 re.Name = 're';
212 re.Scope = 'Input';
213 re.Port = 1;
214 so = Stateflow.Event(chart);
215 so.Name = 'so';
216 so.Scope = 'Input';
217 so.Port = 2;

```

```

218 sc = Stateflow.Event(chart);
219 sc.Name = 'sc';
220 sc.Scope = 'Input';
221 sc.Port = 3;
222 ocp = Stateflow.Event(chart);
223 ocp.Name = 'ocp';
224 ocp.Scope = 'Input';
225 ocp.Port = 4;
226 ot = Stateflow.Event(chart);
227 ot.Name = 'ot';
228 ot.Scope = 'Input';
229 ot.Port = 5;
230 ct = Stateflow.Event(chart);
231 ct.Name = 'ct';
232 ct.Scope = 'Input';
233 ct.Port = 6;
234
235 % Inicial transition
236 dt0 = Stateflow.Transition(chart);
237 dt0.Destination = X_VI;
238 dt0.DestinationOClock = 0;
239 xsource = X_VI.Position(1)+ X_VI.Position(3)/2-10;
240 ysource = X_VI.Position(2)-20;
241 dt0.SourceEndPoint = [xsource ysource];
242 sfsave(m.Id,'DESmodel');
243
244 % Script to add Input Block for Stateflow simulation
245 simulink; % Open Simulink Library
246 add_block('simulink/Sinks/To Workspace','DESmodel/output');
247 set_param('DESmodel/output','Position',[170 30 240 55]); % Set Position
248 % Add Routing Mux
249 add_block('simulink/Signal Routing/Mux','DESmodel/route');
250 set_param('DESmodel/route','Position',[140, 26, 145, 64]); %Set Position
251 set_param('DESmodel/route','Inputs','6'); % Number of input ports
252 % Add lines between blocks
253 % Add lines from Stateflow to routing Mux
254 add_line('DESmodel','DESdiag/1','route/1');
255 add_line('DESmodel','DESdiag/2','route/2');
256 add_line('DESmodel','DESdiag/3','route/3');
257 add_line('DESmodel','DESdiag/4','route/4');
258 add_line('DESmodel','DESdiag/5','route/5');
259 add_line('DESmodel','DESdiag/6','route/6');
260 % Add line from route to output
261 add_line('DESmodel','route/1','output/1');
262 sfsave('DESmodel','DESmodel');

```