Erick Wakamoto Takarabe

Sistemas de Controle Distribuídos em Redes de Comunicação

São Paulo
Agosto de 2009

Erick Wakamoto Takarabe

Sistemas de Controle Distribuídos em Redes de Comunicação

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Mecânica da Escola Politécnica da Universidade de São Paulo para obtenção do Título de Mestre em Engenharia Mecânica.

Área de concentração:
Engenharia de Controle e Automação Mecânica

Orientador:
Prof. Dr. Newton Maruyama

São Paulo
Agosto de 2009

Este exemplar foi revisado e alterado em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, 20 de outubro de 2009.

Assinatura do autor _____

Assinatura do orientador_____

FICHA CATALOGRÁFICA

Takarabe, Erick Wakamoto
Sistemas de controle distribuídos em redes de comunicação / E.W. Takarabe. -- ed.rev. -- São Paulo, 2009.
110 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos.

1. Sistemas de controle I. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos II. t.

À minha família: berço e razão de tudo.

Agradecimentos

Primeiramente gostaria de agradecer aos meus pais por todo amor e carinho que foram fundamentais para chegar até este presente momento, principalmente nas horas difíceis (que não foram poucas!). E à Aline, Seu Tada, Dona Cris, Dri, Sato, Mirianzinha (Mariazinha), Ró e Clau por me acolherem como um membro desta linda família.

Não somente aos amigos que me acompanharam nesta etapa de minha vida, mas também aos que estiveram nas anteriores, meu sincero muitíssimo obrigado! Me sinto muito afortunado em dizer que o motivo de não citar seus nomes é que não haveria espaço suficiente para todos eles.

Para os membros da banca de qualificação, Prof. Dr. Paulo Eigi Miyagi e Prof. Dr. Fábio Gagliardi Cozman, fica meu agradecimento pelas críticas construtivas e minha admiração pelas respectivas contribuições ao mundo da pesquisa. Sou grato também ao meu orientador, Prof. Dr. Newton Maruyama, cujos ensinamentos foram muito além do tema desta pesquisa.

Gostaria por fim de agradecer ao apoio da FAPESP através de minha bolsa de estudos.

"O homem que aos 50 anos vê o mundo do mesmo modo que via aos 20, perdeu 30 anos de sua vida."(Muhammad Ali)

Resumo

Sistemas de controle distribuídos cujas malhas são fechadas através de uma rede de comunicação são chamados de sistemas de controle distribuídos em redes de comunicação (NCS - *Networked Control System*). Este tipo de arquitetura permite a divisão do sistema de controle em módulos interconectados através da rede de comunicação, proporcionando a divisão do processamento, a redução de custo e de peso, além de facilitar o diagnóstico e manutenção do sistema e de aumentar a sua flexibilidade e agilidade; e por isso seu emprego na indústria está se tornando comum (e.g., *fly-by-wire* e *drive-by-wire*). Porém, a distribuição do processamento e a inserção de uma rede de comunicação aumenta a complexidade da análise e do projeto deste tipo de sistema. Um dos fatores que contribui para esse aumento da complexidade é a presença de atrasos aleatórios nos sinais de controle, causados pela dinâmica do sistema computacional (conjunto de *hardware* e *software*) que serve como plataforma para implementação do sistema de controle digital.

Este trabalho faz um estudo sobre este tipo de sistema sob a perspectiva destes sinais com atrasos. Para isso, faz-se uso dos *toolboxes* para MATLAB: TrueTime e Jitterbug. Através destas ferramentas, mostra-se a existência de uma relação de compromisso entre o desempenho do controle e o desempenho do sistema computacional.

Através deste estudo, é proposto uma solução de um sistema de controle do tipo NCS para um ROV (do inglês *Remotely Operated Vehicle*), modelado através de 6 equações diferenciais desacopladas não-lineares. Este tipo de veículo tem uma relevância econômica significativa para o Brasil, visto que é utilizado em operações de manutenção e instalação de plataformas de extração do petróleo que está depositado em profundidades que variam de mil a 2 mil metros. Para este NCS proposto, são utilizados controladores do tipo PI com estrutura *feedback-feedfoward* cujos parâmetros de projeto são obtidos em função dos atrasos inseridos pelo sistema computacional.

Palavras-chave: NCS, rede de comunicação, sistemas de controle distribuídos, TrueTime, Jitterbug, ROV.

Abstract

Distributed control systems wherein the control loops are closed through a communication network are called Networked Control Systems (NCSs). This type of architecture allows the control systems division into modules interconnected through the communication network, providing the processing division, reduction of cost and weight, and facilitates the systems diagnosis and maintenance, and increases their flexibility and agility. Therefore its use in industry is becoming common (eg, fly-by-wire and drive-by-wire). However, the processing distribution and the communication network insertion increase the system analysis and design complexity. One of the factors that contributes to this increased complexity is the presence of random time delays, caused by the dynamics of the computer system (set of hardware and software) used as a platform for digital control system implementation.

This work deals with the networked control systems under these random time delays view. For this, it is used two MATLAB toolboxes: Jitterbug and TrueTime. With these tools, it is shown the existence of a relationship between the performance of control and performance of computer system.

With this study, it proposed a solution of a NCS for a ROV (Vehicle Operated Remotely), modeled by 6 differential nonlinear decoupled equations. This type of vehicle has a significant economic relevance for Brazil, as it is used in maintenance and installation of platforms for oil extraction deposited at depths ranging from thousand to 2 thousand meters. For this proposed NCS are adopted PI controllers with feedfoward-feedback structure whose parameters design are given in terms of delay inserted by the computer system.

Keywords: ROV, NCS, network, distributed control, TrueTime, Jitterbug.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas

ROV *Remotely Operated Underwater Vehicle*

NCS *Networked Control System*

A/D Analógico-Digital

D/A Digital-Analógico

LQG/LTR *Linear Quadratic Gaussian / Loop Transfer Recovery*

CPU *Central Processing Unit*

STR Sistema de Tempo Real

RTOS *Real-Time Operating System*

FP *Fixed Priority*

RM *Rate Monotonic*

EDF *Earliest Deadline First*

FBW *Fly by Wire*

DBW *Drive by Wire*

MAC *Media Access Cotnrol*

CSMA/CD *Carrier Sense Multiple Access with Collision Detection*

CSMA/AMP *Carrier Sense Multiple Access with Arbitration on Message Priority*

FDMA *Frequency Division Multiple Access*

TDMA *Time Division Multiple Access*

WLAN *Wireless Local Area Network*

CSMA/CA *Carrier Sense Multiple Access with Collision Avoidance*

ZOH *Zero-Order-Hold*

MJLS *Markov Jump Linear Systems*

MSS *Mean Square Stability*

WCET *Worst Case Execution Time*

CAN *Controller Area Network*

FIFO *First in, First out*

VSOR Veículo Submarino Operado Remotamente

GL Grau de Liberdade

Lista de Símbolos

Os seguintes símbolos serão utilizados:

símbolo	descrição
h	Período de amostragem nominal
h_k	Intervalo de amostragem
L_{io}^k	Latência de entrada-saída
L_s^k	Latência de amostragem
$\mathcal{L}$	Transformada de Laplace
$H(t)$	Função de *Heaviside*
$L(s)$	Função de malha aberta
$S(s)$	Função de sensibilidade
$\lambda_i(A)$	Autovalores de A
J	Custo total do sistema obtido através do Jitterbug
$P_\tau(k)$	Probabilidade de um atraso de $k\delta$ segundos
$P(k\delta)$	Covariância do estado no instante $k\delta$
$\phi_y(\omega)$	Densidade espectral de uma saída y
$r_u(k)$	Função de covariância de y
L_S^k	Latência do sensor
L_C^k	Latência do controlador
τ_i	Força/torque correspondente ao GL i
m_i	Inércia/massa virtual correspondente ao GL i
d_{L_i}	Coeficiente de arrasto linear correspondente ao GL i
d_{Q_i}	Coeficiente de arrasto quadrático correspondente ao GL i
b_i	Modelo de distúrbio correspondente ao GL i
η	Vetor de posicão/orientação do veículo no referencial inercial
ν	Vetor de velocidades do veículo no referencial móvel
τ^{GL}	Vetor de forças/torques no veículo no referencial móvel

Conteúdo

1 Introdução

Sistemas de controle distribuído em redes de comunicação, ou conhecidos também como *Networked Control Systems* (NCSs), estão atualmente em evidência, visto que proporcionam aos sistemas de controle características como distribuição do processamento, redução de custo e de peso, além de facilitar o diagnóstico e manutenção do sistema e de aumentar a sua flexibilidade e agilidade (com relação a possíveis adaptações e modificações) (WALSH et al., 2002). Estas características se tornam mais importantes à medida que a complexidade dos sistemas de controle aumenta.

Um exemplo de um sistema de controle complexo é um ROV: veículo submergível não-tripulado ligado a um navio na superfície através de um cordão umbilical tanto para alimentação (energia) quanto para transmissão de dados. Fatores que contribuem para o aumento desta complexidade são as suas restrições físicas e operacionais. Devido a estas restrições, o controle deste tipo de veículo é realizado através de sistemas embarcados[1] de forma a minimizar o espaço ocupado, o peso e o consumo de energia.

Neste trabalho será estudado NCS com o intuito de propor soluções deste tipo para o controle de um ROV. Para isso, o capítulo inicia com uma contextualização de NCSs na Seção 1.1, apresentando seus componentes, seu histórico e os problemas envolvidos do ponto de vista de controle. As Seções 1.2 e 1.3 apresentam a visão geral sobre dois importantes componentes do NCS: o sistema operacional de tempo real e a rede de comunicação.

[1] Um sistema embarcado é um sistema computacional dedicado projetado para uma ou algumas funções específicas, geralmente em *real-time* (CATSOULIS, 2005).

1.1 Contextualização: NCS

Sistemas de controle distribuído cuja malha é fechada através de uma rede de comunicação são chamados de NCSs, como visto na Figura 1.1. Este tipo de sistema apresenta uma série de vantagens já citadas anteriormente. Porém, a distribuição do processamento e a inserção de uma rede de comunicação aumentam a complexidade da análise e do seu projeto. Além de envolver todas as dificuldades encontradas em uma implementação de controle discreto (e.g., *aliasing*; problemas de aproximações; imprecisões numéricas), há também o indeterminismo temporal (e.g. atrasos nos sinais) causados pela dinâmica do sistema computacional[2]; além da possibilidade de perdas de dados, especialmente durante a transmissão de mensagens pela rede de comunicação.

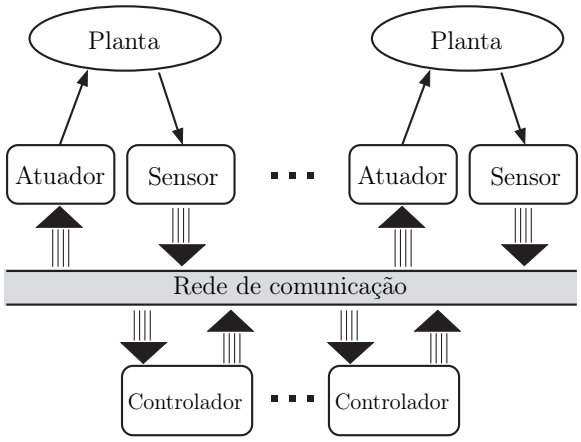


Figura 1.1: Modelo de um NCS.

O histórico de sistemas de controle distribuídos tem início em meados dos anos 70, quando sistemas de controle computadorizado[3] (Figura 1.2) foram divididos em módulos de controle conectados entre si. Porém, estes módulos eram pouco dependentes uns dos outros, sendo as tarefas relativas ao controle realizadas individualmente em cada módulo, onde apenas alguns sinais de monitoramento eram compartilhados.

Com o desenvolvimento e a popularização das redes de comunicação como a Ethernet, e redes de caráter industrial como Fieldbus, CAN-Bus, DeviceNet. etc., além das redes wireless como Bluetooth e ZigBee, foi possível a descentralização dos sensores, processadores e, conseqüentemente, o surgimento dos NCSs. Tais sistemas de controle possuem

[2]Neste trabalho, sistema computacional se refere ao conjunto de *software* e *hardware* sobre o qual o sistema de controle é implantado.

[3]Sistema de controle computadorizado é composto basicamente por: um sistema de conversão Analógico-Digital (A/D); um sistema computacional onde é implementado o controlador em sua forma discreta; e um sistema de conversão Digital-Analógico (D/A).

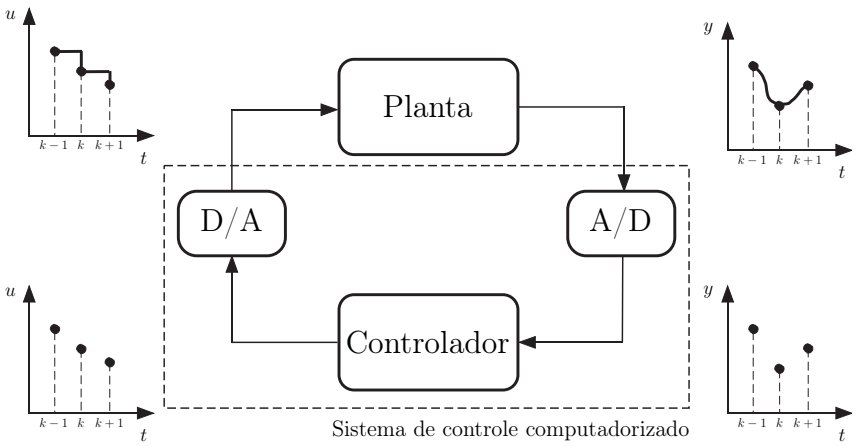


Figura 1.2: Sistema de controle computadorizado composto por: um sistema de conversão Analógico-Digital (A/D); um sistema computacional (*hardware* e *software*) onde é implementado o controlador em sua forma discreta; e um sistema de conversão Digital-Analógico (D/A).

as seguintes características:

- processamento distribuído,
- sensores distribuídos,
- atuadores distribuídos,
- redes de comunicação,
- sistemas operacionais.

Atualmente, para a maior parte da comunidade de controle, o ciclo de controle (composto basicamente pela amostragem do sinal de saída, cálculo do sinal de controle e atuação sobre a planta) é realizado instantaneamente e nos instantes de amostragem kh. Entretanto, esta simplificação pode não caracterizar bem a realidade do sistema de controle computadorizado, pois é necessário um certo tempo para que este ciclo seja executado. E este tempo pode ainda variar de acordo com os atrasos introduzidos pelo sistema computacional. O trabalho de Chow e Tipsuwan (2003) mostra experimentos onde os atrasos deterioram o desempenho do sistema de controle computadorizado, conduzindo o sistema para a região de instabilidade.

Portanto, novas abordagens são necessárias para fazer a análise e o projeto de NCSs. Uma delas pode ser observada em Cervin (2003), onde é apresentada a metodologia de

co-design, na qual o projeto do sistema computacional e do sistema de controle são tratados em conjunto. Este método vem apresentando bons resultados no que diz respeito à melhora do desempenho de NCS, porém, sua implementação não é trivial, visto que exige soluções no nível do *kernel* do sistema operacional.

Dentro do contexto clássico da teoria de controle, o objetivo principal do projeto é obter um controlador K para gerar o esforço de controle $u(t)$ de forma a fazer com que o sistema dinâmico tenha um comportamento desejado. Numerosas técnicas desenvolvidas nas últimas décadas podem ser utilizadas para este fim, como por exemplo: LQG/LTR (CRUZ, 1996), controle preditivo (CLARKE; MOHTADI; TUFFS, 1987), H_∞ (POSTLETHWAITE, 1996), etc. Em qualquer uma dessas técnicas, obtém-se o controlador K descrito através de uma equação diferencial ou ainda através de um sistema de equações diferenciais descrevendo um espaço de estados (eventualmente controladores se reduzem a constantes).

Para obter um sistema de controle computadorizado é necessário transformar as equações diferenciais do controlador K em equações de diferença. Para isto, utiliza-se a teoria clássica de controle de tempo discreto: a Transformada Z (ÅSTRöM; WITTENMARK, 1997). Uma descrição do controlador em espaço de estados em tempo discreto é representada por

$$x_{k+1} = \Phi x_k + \Gamma e_k, \tag{1.1}$$

$$u_k = -K x_k, \tag{1.2}$$

onde x_k é o vetor de estados do controlador; $e(k)$ é o vetor de erro do sistema; e $u(k)$ é o vetor do esforço de controle, todos tomados no instante kh com $k = 0, 1, 2, \ldots, n$. Portanto para o correto funcionamento do sistema de controle computadorizado, todos os sub-sistemas devem cumprir a meta estabelecida pelo período de amostragem nominal h, ou seja, a cada período h devem ser realizados uma conversão A/D; um cálculo do esforço de controle $u(k)$; e uma conversão D/A.

Inicialmente esta meta não gerava tantos problemas, visto que os sistemas de controle possuíam um caráter monolítico, com *hardware* e *software* implementados de forma customizada. Posteriormente, a possibilidade de miniaturização de componentes levou ao aparecimento de microcontroladores de baixo custo e com memória suficiente para rodar um sistema operacional multitarefas (*multitasking*) onde os recursos computacionais são compartilhados entre as tarefas (*tasks*) em execução (vide Figura 1.3). A partir de então

houve a possibilidade da implementação de várias malhas de controle simultâneas, graças a um gerenciamento dos recursos computacionais limitados (através de algoritmos de escalonamento, por exemplo) de uma maneira mais eficiente.

A despeito dessa mudança, a teoria de controle discreto continua sendo utilizada da mesma forma, e os usuários são obrigados a tentar fazer o *software* de maneira a cumprir a meta para cada controlador, aumentando consideravelmente a sua complexidade. Quando se aumenta o compartilhamento de recursos computacionais, aumenta-se a probabilidade de indeterminismo temporal.

Nota-se então que vários fatores podem influenciar no desempenho do sistema de controle multitarefas, como o poder de processamento e a carga da CPU, o período de amostragem nominal requerido para cada controlador, o algoritmo de escalonamento de processos, etc. Em geral, se os requisitos não são cumpridos, considera-se que os recursos computacionais devem ser aumentados.

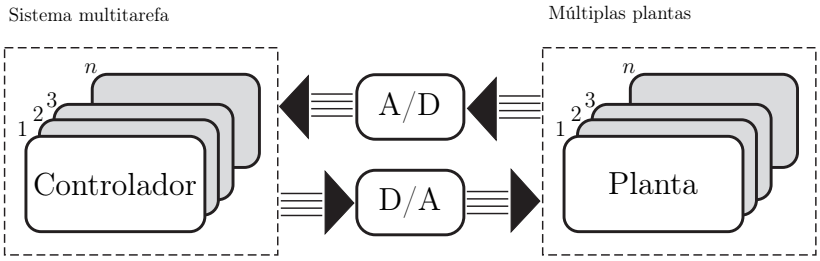


Figura 1.3: Múltiplas plantas sendo controladas por múltiplos controladores em um sistema de processamento multitarefa.

O intervalo entre a leitura das variáveis de entrada e a atuação do esforço de controle é crucial para o desempenho do controle. As variações do tempo de execução do ciclo de controle causadas pela dinâmica do sistema computacional podem levar a deterioração substancial do desempenho do sistema de controle, como visto nos trabalhos de Nilsson (1998), Cervin et al. (2003), Cervin (2003), Lincoln (2003), Årzèn e Cervin (2005), Henriksson et al. (2006).

Os efeitos do indeterminismo temporal (exemplificado na Figura 1.4) no desempenho do sistema de controle é de difícil avaliação. O tempo entre o início e o fim do ciclo de controle pode ser modelado como um atraso na entrada do processo. O tempo entre o instante de amostragem k e o início do ciclo de controle também tem um efeito negativo no desempenho do sistema de controle que é mais acentuado em sistemas com freqüência de amostragem menor e sistemas com menor margem de fase (HENRIKSSON, 2006). Nesse

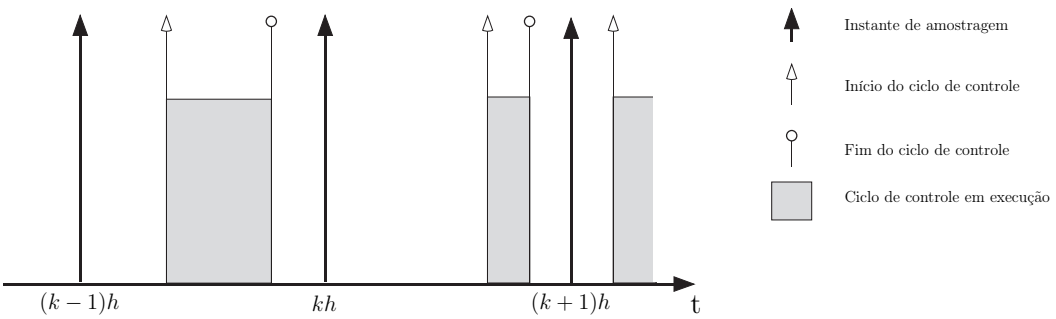


Figura 1.4: Diagrama de tempo dos ciclos de controle.

caso, mesmo pequenas variações no intervalo de amostragem podem causar instabilidade. Uma análise dos efeitos de atrasos aleatórios e intervalos de amostragens em controle ótimo foi apresentada por Davidson (1973).

E quando se trata de NCSs, deve-se considerar também o indeterminismo causado por atrasos variáveis e possíveis perdas de dados durante a transmissão via rede, que dependem basicamente do tipo de protocolo de comunicação e das condições do tráfego presente na rede. Dentro deste contexto, a comunidade de sistemas de controle vem abordando estes problemas, como nos trabalhos de Siljak (1991), Brockett (1995), Ooi et al. (1997), Nilsson (1998), Wong e Brockett (1999). Tais trabalhos modelam as dinâmicas dos canais de comunicação ou digitalização através de modelos determinísticos ou estocásticos.

Nos trabalhos de Montestruque e Antsaklis (2005a, 2005b, 2006) são abordados NCSs utilizando modelos explícitos. Nesta abordagem, é utilizado um observador de estados com o objetivo de reduzir o volume de informações no canal de comunicação. O sistema de controle utiliza as estimativas provenientes do observador enquanto não recebe dados através da rede. Condições de estabilidade do sistema são derivadas considerando diferentes cenários como compensação de atraso, intervalos de atualização constantes e variáveis, plantas não-lineares, erros de quantização, etc. A avaliação de desempenho é realizada utilizando normas modificadas H_2 e a teoria de *lifting* (YAMAMOTO; KHARGONEKAR, 1996), onde o sistema em tempo discreto é analisado como se fosse um sistema em tempo contínuo.

Para análise e projeto de NCSs, estão em desenvolvimento no *Department of Automatic Control of Lund University* dois *toolboxes* para MATLAB:

- Jitterburg (CERVIN; LINCOLN, 2006): torna possível a análise estatística do impacto de latências, *jitters*, amostras perdidas, computações abortadas, etc. sobre o desempenho do sistema de controle,

- TrueTime (ANDERSSON; HENRIKSSON; CERVIN, 2007): simula o comportamento de um sistema operacional de tempo real com redes de comunicação, utilizando MATLAB / SIMULINK.

1.2 Sistema de tempo real (STR)

Uma definição de STR é dada por Gambier (2004):

> "STR é um sistema computacional cujo correto funcionamento não depende somente do resultado lógico da computação, mas também do tempo em que os resultados são produzidos".

Ou seja, o sistema deve entregar o resultado correto dentro de um prazo específico, podendo ser de alguns milisegundos a até segundos. Portanto conclui-se que um sistema de controle computadorizado (inclusive um NCS) também é um STR.

Para cumprir a sua meta, o sistema deve ser capaz alocar os seus recursos computacionais de forma que seu desempenho seja maximizado. Esta alocação de recursos é realizada pelo Sistema Operacional de Tempo Real (RTOS -*Real-Time Operating System*), através de uma operação denominada escalonamento de processos (FARINES; PRAGA; OLIVEIRA, 2000).

Para um STR, mais importante que a rapidez de cálculo é o conceito de previsibilidade. A previsibilidade é o requisito básico no desenvolvimento de um STR. Um STR é dito previsível quando, independentemente de variações que ocorrem em nível de *hardware* e *software*, seu comportamento pode ser antecipado antes de sua execução. A garantia de previsibilidade envolve análises complexas, visto que depende de conhecimentos da carga computacional, da probabilidade de ocorrência de falhas (e.g., perda de dados), dos tempos de computação[4], do *hardware* envolvido, etc., que em geral são de caráter não-determinísticos. Alguns autores analisam a previsibilidade de STR do ponto de vista estocástico.

Os STRs podem ser classificados em dois tipos, do ponto de vista da segurança (*safety*):

[4]Tempo de computação (WCET - *Worst Case Execution Time*) é o tempo gasto, no pior caso, na execução de códigos da aplicação.

- *Soft real-time* ou sistemas não-críticos: quando as conseqüências de uma falha devido à perda de um *deadline* não é crítico para o funcionamento do sistema (e.g., perda de um *frame* durante a execução de um vídeo);

- *Hard real-time* ou sistemas críticos: quando as conseqüências de uma falha devido à perda de um *deadline* é crítico para o funcionamento do sistema (e.g., sistemas de controle de vôo).

Um outro conceito importante em relação ao STR é o conceito de processo (também chamado de *task*). Processo pode ser definido como um programa em execução (DEITEL; DEITEL; CHOFFNES, 2005). Como dito anteriormente, o RTOS deve assegurar que cada processo receba uma quantidade suficiente de tempo de processamento. Porém, em qualquer sistema, o número de processos verdadeiramente executados em concorrência (em paralelo) é obrigatoriamente igual ao número de processadores em um sistema. Portanto, a qualquer dado instante, alguns processos podem ser executados, outros não. Durante o seu tempo de vida um processo passa por uma série de estados distintos, que de forma simplificada pode ser resumido em três estados:

- estado de execução, quando o processo é executado em um processador;

- estado de pronto, quando o processo poderia ser executado se houvesse um processador disponível;

- estado bloqueado, quando o processo está aguardando a ocorrência de um evento (como a conclusão de entrada/saída, por exemplo) para prosseguir sua execução.

O RTOS mantém uma lista de prontos e uma lista de bloqueados. A transição de um processo de uma lista para outra é representada na Figura 1.5. Quando um usuário executa um programa, processos são criados e inseridos na lista de prontos. A lista de prontos é organizada por prioridades, de modo que o próximo processo a receber o processador será o primeiro da lista (o processo de maior prioridade). Quando o processo chega ao topo da lista e há processador disponível, aquele processo é designado ao processador. Esta transição é chamada de despacho do processo. Para evitar que qualquer um dos processos monopolize o sistema, o sistema operacional estabelece um relógio de interrupção em *hardware* (temporizador); se o processo não devolver o processador voluntariamente, a interrupção gerada pelo temporizador faz com que este processo volte para lista de prontos. Se um processo em execução iniciar uma operação de entrada/saída antes do temporizador

atuar, este processo vai para a lista de bloqueados, liberando o processador. Esta transição é chamada de bloqueio do processo. Quando o evento pelo qual o processo espera é concluído, este processo vai para a lista de pronto. Esta transição é chamada de despertar do processo.

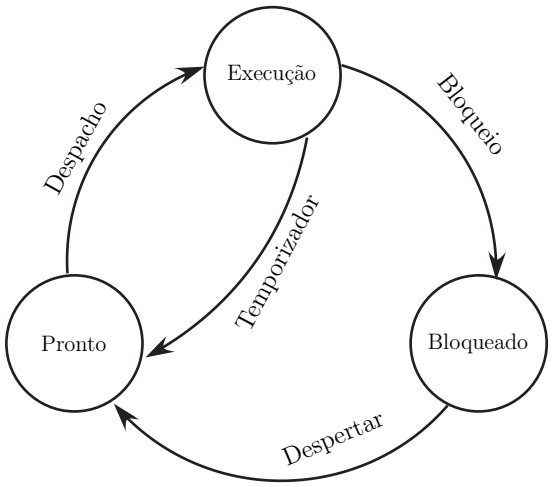


Figura 1.5: Transição de estado de processos.

A estratégia utilizada pelo sistema operacional para escolher o processo que executa é chamada de política de escalonamento. Através deste escalonamento de processos, o sistema pode alocar mais eficientemente os recursos computacionais para satisfazer alguns critérios de desempenho. Para o caso do RTOS, o sistema tem como objetivo garantir a conclusão de cada processo dentro do seu prazo estabelecido. As políticas de escalonamento podem ser preemptivas ou não-preemptivas. No caso de um escalonamento não-preemptivo, uma vez que um processo obtém um processador, o sistema não pode retirar este processador deste processo. No caso de escalonamento preemptivo, o sistema pode retirar o processador, realizando o chamado chaveamento de contexto[5]. O escalonamento preemptivo é útil em sistemas nos quais processos de alta prioridade exigem resposta rápida, como STRs (DEITEL; DEITEL; CHOFFNES, 2005). Algoritmos de escalonamento muitas vezes utilizam prioridades para determinar os processos que devem ser executados. Quando cada processo possui uma prioridade estática, o escalonamento é chamado de escalonamento por prioridade fixa (FP - *fixed priority*).

As políticas de escalonamentos utilizadas em STRs são chamadas de escalonamento de tempo real. Estas políticas tem como objetivo fazer com que os processos gerem as saídas corretas num determinado tempo. Os algoritmos de escalonamento de tempo real

[5]O sistema operacional realiza um chaveamento de contexto para interromper um processo em execução e começa a executar o processo que está no topo da lista de prontos.

mais utilizados são:

- escalonamento por taxa monotônica (RM - *rate monotonic*) - algoritmo de alternância circular, preemptivo, por prioridade, que eleva a prioridade de um processo linearmente com a freqüência com a qual ele deve executar;
- escalonamento por prazo-mais-curto-primeiro (EDF - *earliest deadline first*) - algoritmo de escalonamento preemptivo que despacha primeiro o processo com o prazo mais curto.

Sob o contexto de NCS, a execução dos processos relacionados ao controle introduz um atraso, mesmo que mínimo, ao ciclo de controle; sendo que em alguns casos este atraso é de caráter estocástico (CERVIN, 2003). Portanto a dinâmica do STR (ou mais precisamente do RTOS) deve ser levada em consideração na análise do desempenho do NCS.

1.3 Redes de comunicação

As redes de comunicação foram introduzidas em sistemas de controle em meados da década de 70 na indústria automobilística para promover a redução de custos com cabeamento e a modularização de sistemas. Atualmente é crescente a incorporação de redes de comunicação em sistemas de controle, como em aviões (o chamado *fly by wire*[6] - FBW) e em carros (o chamado *drive by wire*[7] - DBW), por exemplo.

A maioria das redes de comunicação são organizadas como uma pilha de camadas ou níveis, colocadas umas sobre as outras. O número de camadas, o nome, o conteúdo e a função de cada camada diferem de uma rede para outra. No entanto, em todas as redes o objetivo de cada camada é oferecer determinados serviços às camadas superiores, isolando essas camadas dos detalhes de implementação desses recursos. Em certo sentido, cada camada é uma espécie de máquina virtual, oferecendo determinados serviços à camada situada acima dela.

A camada n de um nó da rede (conhecido também como *host*[8]) se comunica com a camada n de outro nó. Coletivamente, as regras e convenções utilizadas nesse diálogo são

[6] *fly by wire* se refere ao uso de dispositivos eletrônicos em vez de mecânicos para interligar os comandos do piloto aos atuadores de controle de vôo (TISCHLER, 1996).

[7] *drive by wire* é o dual do *fly by wire* para veículos terrestres.

[8] *Host* um dispositivo conectado a rede capaz de enviar e receber dados pelo canal de comunicação.

conhecidas como o protocolo da camada n. Basicamente, um protocolo é um acordo entre as partes que se comunicam, estabelecendo como se dará a comunicação. As entidades que ocupam as camadas correspondentes em diferentes nós são chamadas pares (*peers*). Os pares podem ser processos, dispositivos de *hardware* ou mesmo seres humanos. Em outras palavras, são os pares que se comunicam utilizando o protocolo.

Na realidade, os dados não são transferidos diretamente da camada n de um nó para a camada n de outro nó. Em vez disso, cada camada transfere os dados e as informações de controle para a camada imediatamente abaixo dela, até ser alcançada a camada mais baixa. Abaixo da camada 1 encontra-se o meio físico através do qual se dá a transmissão física dos sinais de comunicação. Na Figura 1.6, a comunicação virtual é mostrada por linhas tracejadas e a comunicação física por linhas contínuas.

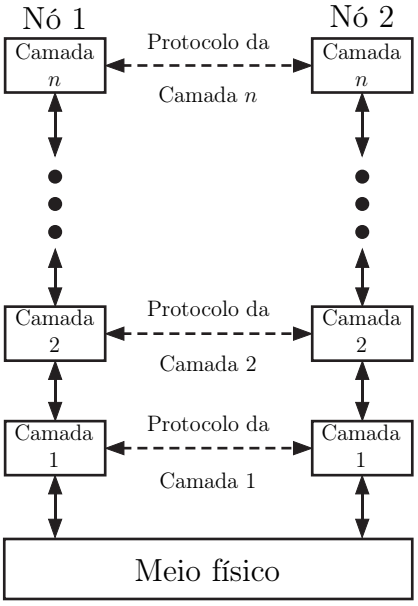


Figura 1.6: Estrutura de rede em camadas.

Quando apenas um único canal está disponível, a determinação do próximo nó que deve acessar o meio é mais complexa. Existem vários protocolos destinados a solucionar o problema. Estes protocolos pertencem a subcamada chamada MAC (*media access control*). Dentre estes protocolos, pode-se destacar:

- protocolo CSMA/CD - *Carrier Sense Multiple Access with Collision Detection* O sistema identifica quando o meio físico está disponível e inicia a transmissão. Caso haja uma colisão (dois nós tentando acessar o meio físico ao mesmo tempo), o sistema a detecta e todas as transmissões são canceladas; e um sinal de colisão é

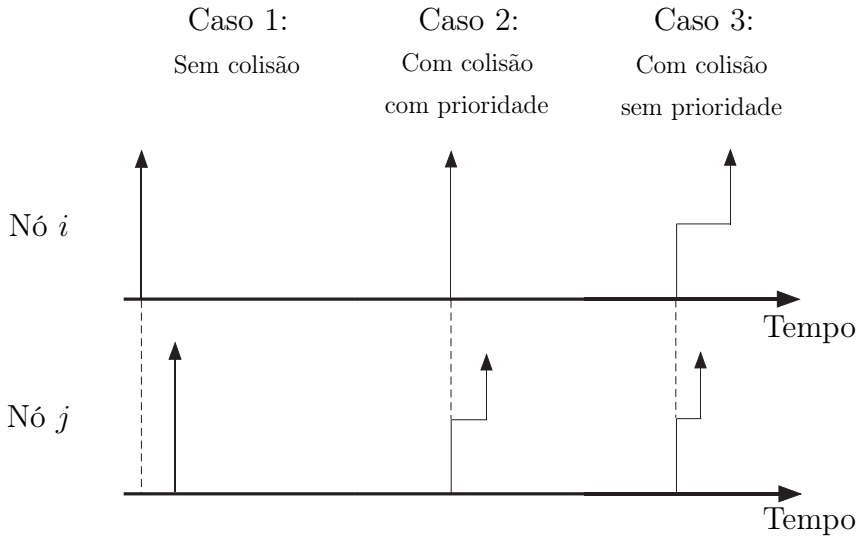


Figura 1.7: Diagrama de tempo para dois nós e um acesso aleatório à rede, onde: o Caso 1 trata de uma situação sem colisão; o Caso 2 trata de uma situação com colisão, onde o nó i tem maior prioridade; e o Caso 3 trata de uma situação com colisão e sem prioridade, onde os dois nós retransmitem a mensagem depois de um tempo aleatório.

emitido. Os nós esperam um tempo aleatório para retransmissão das mensagens (ou pacotes). Isto pode ser visualizado através do Caso 3 da Figura 1.7.

- protocolo CSMA/AMP - *Carrier Sense Multiple Access with Arbitration on Message Priority* O sistema identifica quando o meio está disponível e inicia a transmissão. Caso haja uma colisão, o sistema a detecta e permite que a mensagem de maior prioridade continue sua transmissão. Se duas ou mais mensagens de maior prioridade colidirem, é feita uma escolha aleatória da mensagem que completará a transmissão. Isto pode ser visualizado através do Caso 2 da Figura 1.7.

- FDMA - *Frequency Division Multiple Access* A transmissão de cada nó é feita em uma determinada banda, sendo independente uma da outra. Neste caso não há colisões.

- TDMA - *Time Division Multiple Access* Este protocolo funciona de modo semelhante ao FDMA, porém, cada nó tem 100% da banda disponível somente em determinados períodos de tempo. Se a transmissão não for completada durante um período, ele deve aguardar o próximo disponível.

- WLAN - *Wireless Local Area Network* Este protocolo é definido pela família de especificações IEEE 802.11, baseado no protocolo CSMA/CA (*Carrier Sense Multiple Access with Collision Avoidance*). Cada nó avisa sobre a sua transmissão e o tempo

previsto para realizar a tarefa. Assim, os outros nós que pretendem enviar mensagens esperam decorrer este tempo para então iniciar suas transmissões. Nota-se que este método não impede que as colisões ocorram, mas as evita.

Nota-se que a rede de comunicação é uma potencial fonte de atrasos, e de maioria estocásticos (NILSSON, 1998). Portanto informações sobre a rede de comunicação são relevantes na avaliação do desempenho do NCS.

1.4 Objetivos e apresentação do trabalho

Neste trabalho, o objetivo é analisar sistemas de controle distribuídos em redes de comunicação, mais precisamente a integração das dinâmicas do sistema computacional e do sistema de controle, sob o ponto de vista de atrasos e do desempenho e estabilidade do controle, com o intuito de obter uma solução de controle do tipo NCS para um ROV. Arquiteturas semelhantes à esta são encontradas em ROVs comerciais, como por exemplo o chamado *Distributed Intelligence Control System*, implementado no ROV Seaeye Falcon, da Saab Technologies (SAAB, 2009).

No Capítulo 2 são apresentados dois métodos para análise do impacto dos atrasos em sistemas de controle:

- utilizando um modelo de sistema com atraso e em tempo discreto, que permite fazer a análise em freqüência do sistema para o pior caso de atraso (atraso constante);
- utilizando as ferramentas TrueTime e Jitterbug, que, em conjunto, analisam o impacto de atrasos aleatórios sobre o sistema de controle através da teoria de Sistemas Lineares com Saltos Markovianos (MJLS) e de uma função de custo que relaciona energia na entrada e na saída.

No Capítulo 3 são apresentadas três candidatas para redes de comunicação para NCSs (chamadas também de redes de controle), a saber: Ethernet, ControlNet e DeviceNet. Cada qual com vantagens e desvantagens para sistemas de controle. Um resultado prático apresentado é o uso da rede do tipo Ethernet em NCSs, apesar desta ser considerada (teoricamente) não-determinística (característica necessária para redes de controle). E foram comparados parâmetros relevantes para NCS das candidatas. Desta forma mostra-se que a implementação de cada uma depende do contexto em que elas serão inseridas.

No Capítulo 4 é visto um caso semelhante ao apresentado no trabalho de Chow e Tipsuwan (2003). É possível observar a deterioração do desempenho do NCS conforme o aumento dos atrasos causados pelo sistema computacional. Ao final confirma-se que existe uma relação de compromisso entre o desempenho do sistema computacional e o desempenho do sistema de controle (LIAN; MOYNE; TILBURY, 2002), ou seja, conclui-se que utilizar as informações do sistema computacional na avaliação do desempenho de NCSs contribui para a obtenção de um equilíbrio entre o desempenho do sistema computacional e o desempenho do controle.

No Capítulo 5 é apresentado o modelo matemático do ROV que será utilizado. Este modelo é baseado no trabalho de Caccia, Indiveri e Veruggio (2000) e de Avila, Maruyama e Adamowski (2008), onde os 6 graus de liberdade são desacoplados, sob a hipótese de operações com baixas velocidades. É apresentada a estrutura de controle hierárquica (CACCIA; VERUGGIO, 2000; SOUZA, 2003), onde níveis de controle superiores (e.g., guiagem) são responsáveis pela especificação de valores de referências para níveis de controle inferiores (e.g., estratégia de controle). E por fim mostra-se a estratégia de controle utilizando um controlador do tipo PI com estrutura *feedback-feedfoward*, cujos parâmetros de projeto são dados por ζ_i e ω_{n_i}.

No Capítulo 6 mostra-se como são obtidos os parâmetros ζ_i e ω_{n_i} ótimos para alguns casos de atraso através do TrueTime e do Jitterbug. Estes parâmetros são utilizados em simulações com um modelo do ROV em realidade virtual. Nestas simulações foram realizadas simples tarefas de posicionamento do veículo. Como resultado observa-se que, dado que o sistema é tipicamente estável, os efeitos do indeterminismo temporal pouco afetam o desempenho do sistema de controle para esta determinada tarefa.

No Capítulo 7 constam as conclusões deste trabalho, bem como as sugestões para trabalhos futuros.

2 Métodos de Análise

Na teoria clássica de controle discreto, o ciclo de controle (amostragem do sinal de saída, cálculo do sinal de controle e atuação sobre a planta) é considerado ideal, ou seja, sua execução ocorre instantaneamente nos instantes de amostragem kh, com $k = 1, 2, \ldots$. Porém na prática, em um sistema de controle computadorizado, este ciclo consome um certo tempo para ser executado devido à dinâmica do sistema computacional. O objetivo deste capítulo é apresentar os métodos utilizados para analisar como a interação da dinâmica do sistema computacional e da dinâmica sistema de controle impactam no desempenho do NCS.

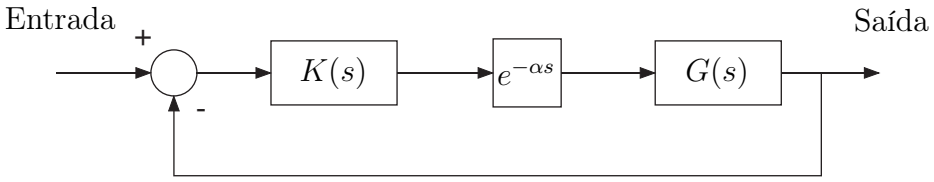


Figura 2.1: Modelo de um sistema com atraso em tempo contínuo, onde $K(s)$ e $G(s)$ representam respectivamente o controlador e a planta.

Em uma primeira aproximação, o tempo de execução do ciclo de controle pode ser representado por um atraso constante α na entrada da planta (Figura 2.1). Assim, dado um sistema SISO cuja resposta em malha aberta no tempo (sem atraso) seja representada por uma função $f(t)$, cuja Transformada de Laplace ($\mathcal{L}[f(t)]$) é $F(s)$, sua a resposta no tempo com atraso é $f(t-\alpha)H(t-\alpha)$, onde $H(t)$ é a função degrau de Heaviside. Aplicando novamente a Transformada de Laplace, a função de malha aberta do sistema com atraso é dada por

$$L(s) = \mathcal{L}[f(t-\alpha)H(t-\alpha)] = e^{-\alpha s}F(s) \tag{2.1}$$

(WOLOVICH, 1994). Sabendo que $e^{-j\omega}$ pode ser escrito como

$$e^{-j\alpha\omega} = \cos\alpha\omega - j\sin\alpha\omega, \tag{2.2}$$

onde

$$\begin{aligned} |e^{-j\alpha\omega}| &= 1, \\ \angle\{e^{-j\alpha\omega}\} &= -\alpha\omega. \end{aligned}$$

Desta forma a magnitude e a fase do sistema em malha aberta com atraso são dadas por

$$20\log|L(j\omega)| = 20\log|F(j\omega)| \tag{2.3}$$

$$\angle\{L(j\omega)\} = -\alpha\omega + \angle\{F(j\omega)\}. \tag{2.4}$$

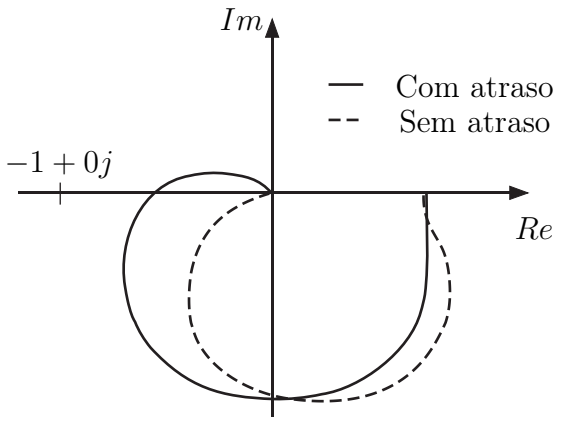


Figura 2.2: Exemplo de um diagrama de Nyquist de um sistema de segunda ordem com e sem atraso (WOLOVICH, 1994).

Portanto para este caso, o ganho da função de malha aberta não sofre alterações, porém sua fase sofre um atraso proporcional à freqüência, piorando o desempenho do controle em malha fechada principalmente em altas freqüências. O efeito deste atraso pode ser visualizado no Diagrama de Nyquist da Figura 2.2, onde a curva da função de malha aberta do sistema com atraso está mais próxima do ponto crítico $(-1+0j)$ do que a do sistema sem atraso.

Uma possível solução para o problema deste atraso é considerá-lo no projeto do controlador através da inclusão de um erro de modelagem da planta do sistema. O diagrama de blocos do novo modelo da planta $G_R(s)$ (chamado de aqui de "modelo real") é mostrado na Figura 2.3, onde há também o erro de modelagem $\varepsilon_M(s)$ (chamado aqui de erro multiplicativo) e o modelo nominal da planta $G_N(s)$. Desta forma, tem-se que

$$G_R(s) = [1 + \varepsilon_M(s)]G_N(s). \tag{2.5}$$

Admite-se que este erro multiplicativo possui um limite superior, de forma que

$$|\varepsilon_M(j\omega)| \leq e_M(\omega) \ \forall \omega \in \mathbb{R}. \tag{2.6}$$

O comportamento típico de $e_M(\omega)$ é ilustrado qualitativamente na Figura 2.4. Este comportamento sugere que os modelos nominais são fiéis em baixas freqüências e grosseiros em altas (CRUZ, 1996). A Figura 2.5 mostra a relação de $G_N(j\omega)$, $G_R(j\omega)$ e $e_M(\omega)$ através do Diagrama de Nyquist, onde os círculos tracejados demarcam a área das trajetórias possíveis da curva de $G_R(j\omega)$. O modelo matemático que deriva esta gráfico é detalahdo no Anexo A.

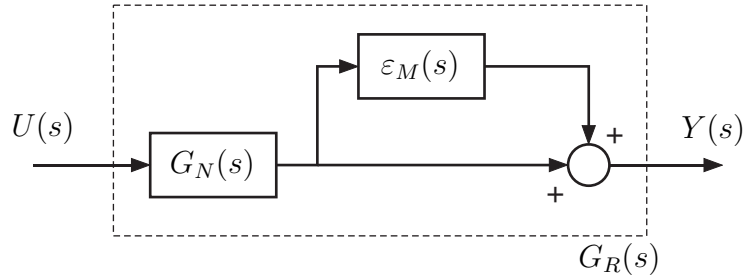


Figura 2.3: Diagrama de blocos do modelo real $G_R(s)$ da planta do sistema, composto pelo erro multiplicativo $\varepsilon_M(s)$ e pelo modelo nominal $G_N(s)$.

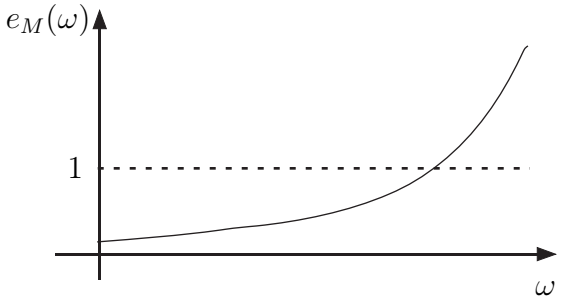


Figura 2.4: Ilustração qualitativa do comportamento típico de $e_M(\omega)$.

Mesmo considerando o atraso proveniente do sistema computacional de forma aproximada no projeto do controlador robusto, é necessário analisar com mais detalhes as interações entre o sistema computacional e o sistema de controle de forma a aumentar a confiabilidade do projeto do NCS. Para tanto, neste capítulo é apresentado na Seção 2.1 o método de análise que utiliza um modelo de sistema com atraso baseado na teoria clássica de controle que permite fazer a análise em freqüência do sistema considerando o pior caso de tempo para execução do ciclo de controle (maior atraso possível). Outro possível método de análise é através dos *toolboxes* TrueTime (ANDERSSON; HENRIKSSON; CERVIN, 2007) e Jitterbug (CERVIN; LINCOLN, 2006), apresentado na Seção 2.2. Na Seção 2.3

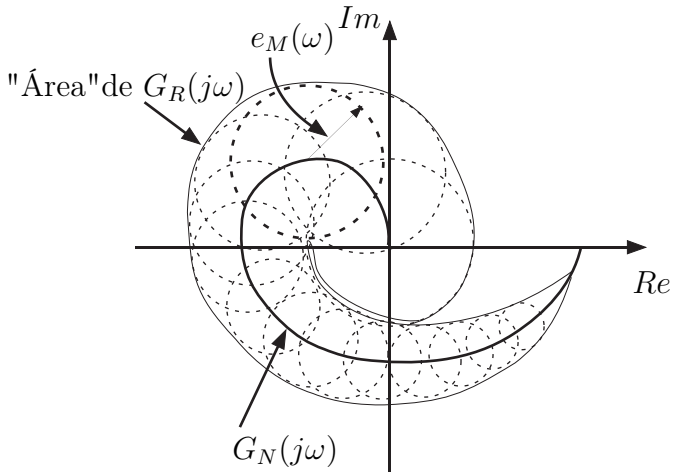


Figura 2.5: Diagrama de Nyquist mostrando a relação entre $G_N(j\omega)$, $G_R(j\omega)$ e $e_M(\omega)$. Os círculos tracejados demarcam a área das trajetórias possíveis da curva de $G_R(j\omega)$.

é apresentado um exemplo que permite mostrar e comparar as análises de um sistema através dos dois métodos.

2.1 Método com o Modelo do Sistema com Atraso

Como visto na seção anterior, o tempo de execução do ciclo de controle pode ser aproximado por um atraso de transporte do sinal de controle na entrada da planta através da inserção de um termo exponencial para sistemas em tempo contínuo. Porém, para sistemas amostrados que utilizam um reconstrutor de ordem zero (ZOH - *zero-order-hold*), cujo sinal de controle é constante entre os instantes de amostragem, existe outra forma para representar este atraso (ÅSTRöM; WITTENMARK, 1997), como será mostrado ao longo desta seção.

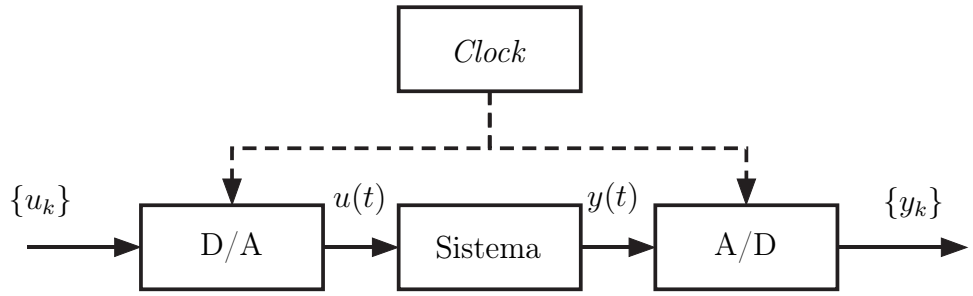


Figura 2.6: Diagrama de blocos com um sistema em tempo contínuo conectado aos conversores D/A e A/D, onde $\{u_k\}$ e $\{y_k\}$ são seqüências do sinal de controle e do sinal de saída, respectivamente, tomados nos instantes de amostragem kh; e *Clock* é sincronizador do do sistema computacional.

Seja o sistema em tempo contínuo da Figura 2.6, cuja representação em espaço de estados é

$$\begin{aligned} \dot{x}(t) &= Ax(t) + u(t), \\ y(t) &= Cx(t) + Du(t). \end{aligned} \tag{2.7}$$

Seu estado no instante t, onde $t_k \leq t \leq t_{k+1}$, é dado por

$$x(t) = e^{A(t-t_k)}x_k + \int_{t_k}^{t} e^{A(t-s)}Bu(s)ds, \tag{2.8}$$

onde $x_k = x(t_k)$. Como o sinal de controle é constante entre os instantes de amostragem, tem-se que

$$\begin{aligned} x(t) &= e^{A(t-t_k)}x_k + \int_0^{t-t_k} e^{As}ds\ Bu_k \\ &= \Phi(t,t_k)x_k + \Gamma(t,t_k)u_k \end{aligned}, \tag{2.9}$$

Se os conversores D/A e A/D estiverem perfeitamente sincronizados; os tempos das conversões forem desprezíveis; e o período de amostragem for constante, o modelo se torna

invariante no tempo e é descrito por

$$\begin{array}{rcl} x_{k+1} & = & \Phi x_k + \Gamma u_k \\ y_k & = & C x_k + D u_k \end{array}, \tag{2.10}$$

onde

$$\begin{array}{rcl} \Phi & = & e^{Ah} \\ \Gamma & = & \displaystyle\int_0^h e^{As} ds \; B. \end{array}$$

Para inserir um atraso constante τ neste modelo, parte-se da seguinte equação

$$\dot{x}(t) = Ax(t) + Bu(t-\tau).$$

Nesta representação, percebe-se que a atuação do controle no instante t é defasado de um atraso τ. Assumindo que o atraso τ é menor que o período de amostragem nominal h, a integração sobre um período de amostragem é dado por

$$x_{k+1} = e^{Ah} x_k + \int_{kh}^{(k+1)h} e^{A((k+1)h-s')} B u(s'-\tau) ds'.$$

Esta integral pode ser dividida em duas integrais, de forma que

$$x_{k+1} = \Phi x_k + \Gamma_0 u_k + \Gamma_1 u_{k-1}, \tag{2.11}$$

onde

$$\begin{array}{rcl} \Phi & = & e^{Ah}, \\ \Gamma_0 & = & \int_0^{h-\tau} e^{As} ds \; B, \\ \Gamma_1 & = & e^{A(h-\tau)} \int_0^{\tau} e^{As} ds \; B. \end{array} \tag{2.12}$$

Fisicamente isto pode ser interpretado como a soma do efeito do sinal de controle u_{k-1} atuando por um período τ, e o efeito do sinal de controle u_k atuando por um período $h-\tau$ (Figura 2.7), entre dois instantes de amostragem. Adotando como vetor de estados

$$X_k = \begin{bmatrix} x_k \\ u_{k-1} \end{bmatrix},$$

obtém-se o espaço de estados em tempo discreto dado por

$$\begin{array}{rcl} X_{k+1} & = & \begin{bmatrix} \Phi & \Gamma_1 \\ \mathbf{0} & \mathbf{0} \end{bmatrix} X_k + \begin{bmatrix} \Gamma_0 \\ \mathbf{I} \end{bmatrix} u_k, \\ y_k & = & \begin{bmatrix} C & 0 \end{bmatrix} X_k + D u_k. \end{array} \tag{2.13}$$

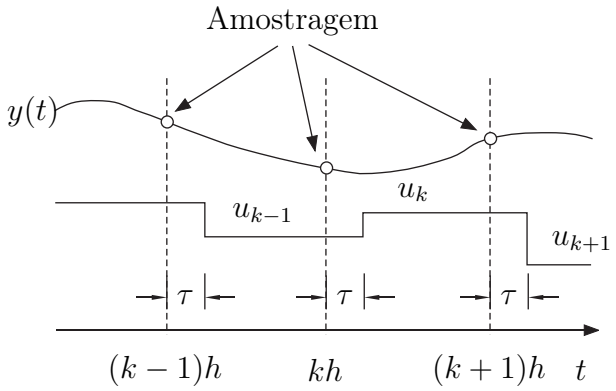


Figura 2.7: Diagrama de tempo do modelo do sistema com atraso, onde o sinal de controle u_{k-1} atua por um período τ, e o sinal de controle u_k atua por um período $h-\tau$.

Para os casos onde o atraso τ é maior do que h, do tipo

$$\tau = (d-1)h + \tau' \quad 0 < \tau' \leq h, \tag{2.14}$$

onde $d = \lceil \tau/h \rceil$, a Equação (2.11) pode ser escrita como

$$x_{k+1} = \Phi x_k + \Gamma_0 u_{k-(d-1)} + \Gamma_1 u_{k-d}, \tag{2.15}$$

onde Γ_0 e Γ_1 são dados pela Equação (2.12), substituindo-se τ por τ'. Assim, o novo sistema é dado por:

$$\begin{bmatrix} x_{k+1} \\ u_{k-(d-1)} \\ \vdots \\ u_{k-1} \\ u_k \end{bmatrix} = \begin{bmatrix} \Phi & \Gamma_1 & \Gamma_0 & \cdots & 0 \\ 0 & 0 & I & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & I \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} x_k \\ u_{k-d} \\ \vdots \\ u_{k-2} \\ u_{k-1} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ I \end{bmatrix} u_k. \tag{2.16}$$

Através desta representação é possível fazer a análise em freqüência do sistema considerando o pior caso do tempo de execução do ciclo de controle. Para isso, escreve-se o sistema em espaço de estados da Equação (2.10) na forma de uma função de transferência em tempo discreto

$$H(z) = \frac{U(z)}{Y(z)} = C[zI - \Phi]^{-1}\Gamma + D. \tag{2.17}$$

Os pólos desta função de transferência são os autovalores de Φ. A relação entre os pólos de tempo discreto e de tempo contínuo é dado por

$$\lambda_i(\Phi) = e^{\lambda_i(A)h}, \tag{2.18}$$

onde $\lambda_i(A)$ são autovalores de A e $\lambda_i(\Phi)$ são os autovalores de Φ. Esta mudança de representação é chamada de mapeamento do plano-s no plano-z, quando $z = \exp(sh)$. Portanto a análise em freqüência é obtida através da substituição de $z = e^{j\omega h}$, onde ω é a freqüência do sinal de entrada.

2.2 Método com os *Toolboxes* Jitterbug e TrueTime

Nesta seção será apresentado o método de análise que utiliza os *toolboxes* para MATLAB que estão em desenvolvimento no *Department of Automatic Control of Lund University*: Jitterbug e TrueTime (CERVIN; LINCOLN, 2006; ANDERSSON; HENRIKSSON; CERVIN, 2007).

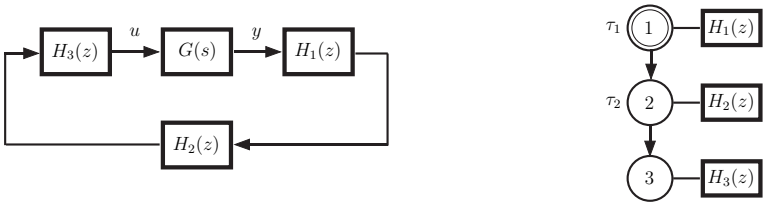
2.2.1 Jitterbug

No Jitterbug, o modelo do sistema de controle (chamado aqui de modelo Jitterbug) é descrito por dois modelos paralelos: um modelo de sinal (Figura 2.8a) e um modelo de tempo (Figura 2.8b). O modelo de sinal é dado por um conjunto de sistemas lineares em tempo contínuo e em tempo discreto, conectados entre si. O modelo de tempo consiste em um conjunto de "nós de tempo"que descreve quando diferentes sistemas de tempo discreto devem ser atualizados durante o ciclo de controle.

Um exemplo de um modelo Jitterbug é mostrado na Figura 2.8, onde um sistema de controle computadorizado é modelado por quatro blocos. A planta é descrita por um sistema em tempo contínuo $G(s)$, e o controlador é descrito por três sistemas em tempo discreto $H_1(z)$, $H_2(z)$ e $H_3(z)$. O sistema $H_1(z)$ pode representar a amostragem periódica, $H_2(z)$ pode representar o cálculo do sinal de controle, e $H_3(z)$ pode representar a atuação. O respectivo modelo de tempo mostra que no início de cada ciclo de controle, $H_1(z)$ deve ser executado primeiro. Então existe um atraso aleatório τ_1, antes que $H_2(z)$ seja executado, e um outro atraso aleatório τ_2 antes que $H_3(z)$ seja executado. Os atrasos consideram o tempo de cálculo, atrasos de escalonamento ou atrasos introduzidos pela rede de comunicação.

Para o modelo de sinal, o sistema em tempo contínuo é descrito por

$$\begin{aligned} \dot{x_c}(t) &= Ax_c(t) + Bu(t) + v_c(t) \\ y(t) &= Cx_c(t), \end{aligned}$$



(a) Modelo de sinal. **(b)** Modelo de tempo.

Figura 2.8: Exemplo de um modelo Jitterbug de um sistema controlado por computador.

onde A, B e C são matrizes constantes, v_c é um processo de ruído branco em tempo contínuo com covariância R_{1c}. O custo deste sistema é especificado como

$$J_c = \lim_{T\to\infty} \frac{1}{T} \int_0^T \begin{pmatrix} x_c(t) \\ u(t) \end{pmatrix}^T Q_c \begin{pmatrix} x_c(t) \\ u(t) \end{pmatrix} dt, \tag{2.19}$$

onde Q_c é uma matriz semidefinida positiva. E o modelo em tempo discreto é descrito por

$$\begin{aligned} x_d(t_{k+1}) &= \Phi x_d(t_k) + \Gamma u(t_k) + v_d(t_k) \\ y(t_k) &= C x_d(t_k) + D u(t_k) + e_d(t_k), \end{aligned}$$

onde Φ, Γ, C e D são matrizes não necessariamente constantes. A covariância dos processos de ruído branco em tempo discreto v_d e e_d é dado por

$$R_d = \mathbf{E}\left\{ \begin{pmatrix} v_d(t_k) \\ e_d(t_k) \end{pmatrix} \begin{pmatrix} v_d(t_k) \\ e_d(t_k) \end{pmatrix}^T \right\}. \tag{2.20}$$

O sinal de entrada u é amostrado quando o sistema é atualizado, e o estado x_d e o sinal de saída y são mantidos constantes entre as atualizações. O custo deste sistema é especificado como

$$J_d = \lim_{T\to\infty} \frac{1}{T} \int_0^T \begin{pmatrix} x_d(t) \\ u(t) \end{pmatrix}^T Q_d \begin{pmatrix} x_d(t) \\ u(t) \end{pmatrix} dt, \tag{2.21}$$

onde Q_d é uma matriz semidefinida positiva.

O sistema completo é obtido através das conexões entre as entradas e saídas dos sistemas em tempo contínuo e em tempo discreto. Formulações MIMO (*multi-input* e

multi-output) são permitidas. O custo total é a soma dos custos de todos os subsistemas

$$J = \sum J_c + \sum J_d. \tag{2.22}$$

Percebe-se que tanto J_c quanto J_d refletem uma relação de energia do sistema, ou seja, um valor de J elevado indica uma melhor relação de energia na entrada e na saída do sistema. Em outros termos, um custo elevado indica que o sistema se encontra próximo da região de instabilidade, onde $J \to \infty$.

No modelo de tempo, cada nó é associado com um ou mais sistemas em tempo discreto que devem ser atualizados quando o nó se torna ativo. No instante zero, o primeiro nó é ativo. O primeiro nó também pode ser declarado periódico, ou seja, a execução será reiniciada neste nó a cada h segundos. Isto é útil para modelar controladores periódicos e simplifica os cálculos dos custos. Cada nó é associado a um atraso τ, que deve ser transcorrido antes que o próximo nó se torne ativo. Um atraso é descrito por uma função densidade probabilidade discreta

$$P_\tau = [\ P_\tau(1) \quad P_\tau(2) \quad \dots\ ], \tag{2.23}$$

onde $P_\tau(k)$ representa a probabilidade de ocorrência de um atraso de $k\delta$ segundos. O passo da discretização do tempo δ é uma constante especificada para todo o modelo.

O mesmo sistema discreto pode ser atualizado em vários nós de tempo. É possível especificar diferentes equações de diferenças (i.e., diferentes matrizes Φ, Γ, C e D) nos vários casos. Isto pode representar controladores com compensação de *jitter*.

O cálculo do custo total é realizado em três passos:

1. as funções de custo, o ruído em tempo contínuo e os sistemas em tempo contínuo são discretizados utilizando um passo δ;

2. o sistema em malha fechada é então formulado como um Sistema Linear com Saltos Markovianos (*Markov Jump Linear Systems* – MJLS), onde são usados nós de Markov para representar o período entre os nós de execução;

3. a variância estacionária de todos os estados do sistema é calculado.

Inicialmente, os estados e os custos são considerados no tempo contínuo. Os estados em tempo discreto são tratados como estados em tempo contínuo com dinâmica nula. Isto

significa que o sistema total pode ser escrito como

$$\dot{x}(t) = Ax(t) + v_c(t), \tag{2.24}$$

onde x representa todos os estados do sistema e v_c é um processo de ruído branco em tempo contínuo com covariância R_c. Para modelar as mudanças em tempo discreto de alguns estados quando um nó de tempo n é ativado, o estado é transformado instantaneamente por

$$x(t^+) = E_n x(t) + e_n(t), \tag{2.25}$$

onde $e_n(t)$ é um processo de ruído branco em tempo contínuo com covariância W_n.

Para o primeiro passo, o sistema da Equação (2.24) é amostrado com um período δ, obtendo-se

$$x(k\delta + \delta) = \Phi x(k\delta) + v(k\delta), \tag{2.26}$$

onde a covariância de v é R. A função de custo para todo o sistema é então definida por

$$J = \lim_{N\to\infty} \frac{1}{N\delta} \sum_{k=0}^{N-1} (x^T(k\delta)Qx(k\delta) + q), \tag{2.27}$$

onde

$$\Phi = e^{A\delta} \tag{2.28}$$

$$R = \int_0^\delta e^{A(\delta-\tau)} R_c e^{A^T(\delta-\tau)}\, d\tau \tag{2.29}$$

$$Q = \int_0^\delta e^{A^T t} Q_c e^{At}\, dt \tag{2.30}$$

$$q = tr\left(Q_c \int_0^\delta \int_0^\delta e^{A(t-\tau)} e^{A^T(t-\tau)}\, d\tau\, dt\right). \tag{2.31}$$

Para o segundo passo, o sistema é modelado como um MJLS (Figura 2.9), onde cada estado de Markov representa o estado do sistema para cada atraso. Cada nó de tempo é representado por um nó de Markov. Nós adicionais podem ser inseridos para representar atrasos entre os nós de tempo. A covariância do estado $P(k\delta) = \mathbf{E}\{x(k\delta)x^T(k\delta)\}$ pode ser escrita na forma recursiva como

$$P(k\delta + \delta) = \Phi P(k\delta)\Phi^T + R. \tag{2.32}$$

Quando um novo nó de tempo é alcançado, os estados mudam de acordo com a Equa-

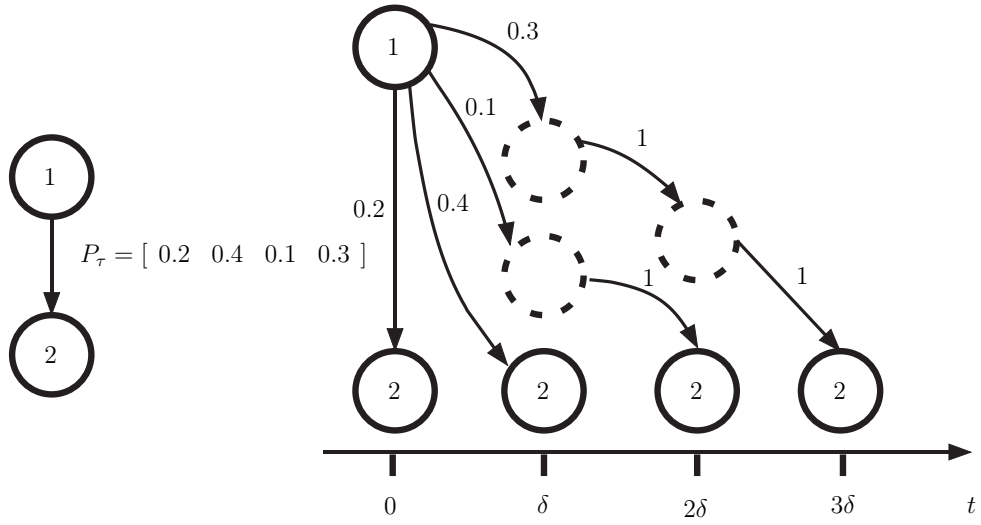


Figura 2.9: Exemplo de um modelo de tempo e a sua correspondente cadeia de Markov.

ção (2.25). E a covariância do estado é descrita por

$$P(k\delta^+) = E_n P(k\delta) E_n^T + W_n, \tag{2.33}$$

onde W_n é a covariância do ruído $e_n(k\delta)$ no nó n. Desta forma, define-se:

$$\Phi_n = \begin{cases} \Phi & \text{se } n \text{ é um nó intermediário,} \\ E_n \Phi & \text{se } n \text{ é um nó de tempo.} \end{cases} \tag{2.34}$$

Analogamente:

$$R_n = \begin{cases} R & \text{se } n \text{ é um nó intermediário,} \\ E_n R E_n^T W_n & \text{se } n \text{ é um nó de tempo.} \end{cases} \tag{2.35}$$

Para o terceiro passo, toma-se $\pi_n(k\delta)$ como a probabilidade do sistema estar no estado de Markov n no instante $k\delta$, e $P_n(k\delta)$ a covariância do estado se o sistema está no estado de Markov n no instante $k\delta$. Então, seja σ a matriz de transição da cadeia de Markov, tal que

$$\pi(k\delta + \delta) = \sigma \pi(k\delta). \tag{2.36}$$

Portanto a covariância do estado é dada por

$$P_n(k\delta + \delta) = \sum_i \sigma_{ni} \pi_i(k\delta)(\Phi_n P_i(k\delta) \Phi_i^T + R_n), \tag{2.37}$$

e o custo no passo k é dado por

$$\frac{1}{\delta}\sum_{n}\pi_n(k\delta)(tr(P_n(k\delta)Q)+q). \tag{2.38}$$

O custo total é calculado sobre a discretização de um período, para sistemas periódicos.

Para sistemas periódicos, a ferramenta pode também calcular a densidade espectral de todas as saídas. Isto pode ser usado para análise em freqüência do sistema em malha fechada. A densidade espectral de uma saída y é definida como

$$\phi_y(\omega)=\frac{1}{2\pi}\sum_{k=-\infty}^{\infty} r_y(k)e^{-j\omega k}, \tag{2.39}$$

onde $r_y(k)$ é a função de covariância de y. Esta função é computada como

$$\begin{aligned} r_y(k) &= \mathbf{E}\{y(t)y^T(t)(t+kh)\} \\ &= \mathbf{E}\{Cx(t)x^T(t+kh)C^T\}\ , \\ &= C\overline{\Phi}^{|k|}P_1(\infty)C^T \end{aligned} \tag{2.40}$$

onde $\overline{\Phi}$ é a matrix de transição média sobre um período e $P_1(\infty)$ é a covariância estacionária em um nó periódico. A densidade espectral retorna como um sistema linear $F(z)$ tal que $\phi_y(\omega)=F(e^{j\omega})$.

A análise feita com o modelo com atraso apresentado na seção anterior é feita considerando somente o pior caso de atraso. Já com o Jitterbug a análise é feita para casos específicos de atrasos (ou mais precisamente, de distribuições de atrasos). O valor da função de custo é utilizado como parâmetro de comparação de desempenho e estabilidade do sistema de controle entre os diferentes casos analisados. A estabilidade aqui citada é a estabilidade no sentido quadrático médio (*Mean Square Stability* – MSS).

Uma forma simples de interpretar a análise com o Jitterbug é que o desempenho e a estabilidade do sistema de controle depende tanto do valor do atraso como também da freqüência que ele ocorre, i.e., podem existir atrasos que deixam o sistema momentaneamente instável, porém ele se mantém estável na média. Portanto, este método é menos conservador que o método de análise com o modelo com atraso.

Para cada análise com o Jitterbug, é necessário fornecer como dado de entrada as distribuições dos atrasos para cada nó de tempo. Estas distribuições podem ser obtidas através de experimentos ou simulações, como por exemplo através do TrueTime, *toolbox* que será apresentado na próxima seção.

2.2.2 TrueTime

O TrueTime é um *toolbox* para MATLAB/SIMULINK que permite a simulação do comportamento temporal de um *kernel* multi-tarefa de tempo real executando *tasks* de controle, sendo possível acrescentar alguns modelos de redes de comunicação.

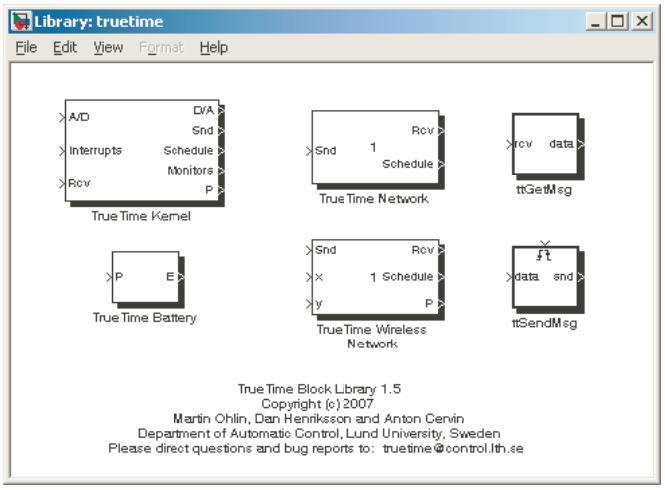


Figura 2.10: Biblioteca de blocos do TrueTime.

Esta ferramenta é composta por uma biblioteca de blocos TrueTime (Figura 2.10) para SIMULINK, que se conectam a outros blocos do SIMULINK para formar os modelos TrueTime, mostrado na Figura 2.11. O bloco TT-*Kernel* é dirigido a eventos e executa códigos que modelam *tasks*. Os modelos de *tasks* são descritos por arquivos do tipo `"arquivo.m"`(como por exemplo o Código 2.1), chamados de *code functions*. O *code function* é dividido em segmentos, e cada segmento é composto basicamente por pequenas tarefas, como por exemplo a leitura de sinais através do conversor A/D através da função `ttAnalogIn()`. Uma das características dos *code functions* (ou das *tasks*) é o seu pior caso de tempo de execução (`exectime`), ou simplesmente WCET[1] (*Worst Case Execution Time*). Além do WCET, as *tasks* são caracterizadas pelo seu período nominal de amostragem (para *tasks* periódicas), pelo seu *deadline* relativo e pela sua prioridade de execução (para o caso de escalonamento por prioridades). Estes atributos são definidos pelo usuário através do *script* de inicialização do TT-*Kernel*, chamado de *initialization script* (como por exemplo o Código 2.2). Nele também é definido a política de escalonamento de *tasks* utilizada por este *kernel*. Desta forma, as informações do sistema computacional são inseridas na avaliação de desempenho do NCS.

[1]O WCET é o tempo máximo que uma *task* demora para ser executada em determinado sistema computacional.

Dos atributos listados, apenas o WCET não é definido no projeto do sistema computacional. A estimativa do WCET é obtida a partir de medidas do tempo decorrido para execução da *task* em questão. Outra maneira de obter esta estimativa é através de ferramentas de análises de WCET (SANDELL, 2004), como aiT, Bound-T, SWEET e Chronos; que estimam o número de ciclos de *clock* necessário para a execução da *task*, através do seu código Assembler.

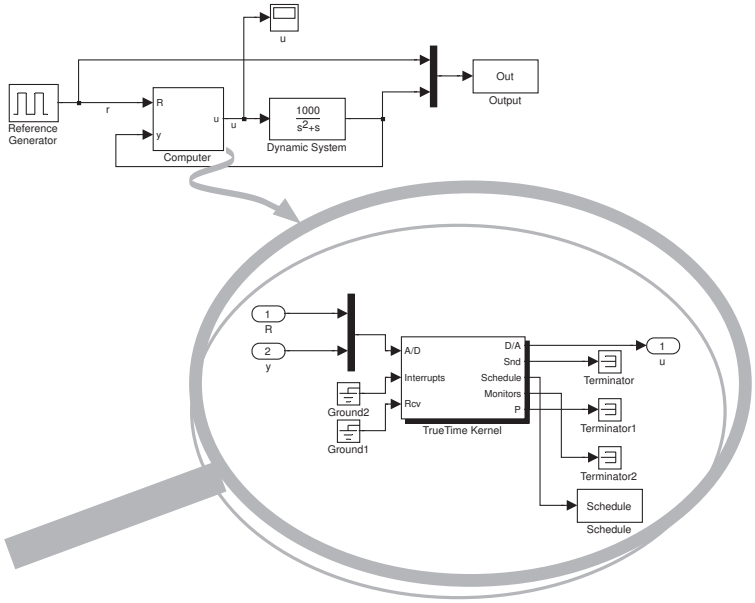


Figura 2.11: Modelo *TrueTime* do sistema de controle de tempo real de um sistema dinâmico.

Código 2.1: Exemplo de *code function* (`Pcontroller.m`) para um controlador proporcional.

```
1  function [exectime, data] = Pcontroller(segment, data)
2  switch segment,
3      case 1,
4          r = ttAnalogIn(1);              % Leitura do sinal de referência
5          y = ttAnalogIn(2);              % Leitura do sinal de saída
6          data.u = data.K *(r-y);         % Cálculo do sinal de controle
7          exectime = 0.002;               % Tempo de execução desta task
8      case 2,
9          ttAnalogOut(1, data.u);         % Atuação do sinal de controle
10         exectime = -1;                  % Fim da task
11 end
```

Código 2.2: Exemplo de *initialization script* (`exampleInit.m`)

```
1  function exampleInit(argument)
2  ttInitKernel(2,1,'prioFP');            % Iniciação do Kernel com
3                                         % prioridade fixa
4  data.u = 0;                            % Valores iniciais de
5  data.K = 2;                            % variáveis
6  period = 0.1;                          % Período da task
7  prio = 5;                              % Prioridade da task
8  ttCreatePeriodicTask('ctrl',offset,period,prio,
9      'Pcontroller',data);                % Cria a task periódica
```

Para a análise do sistema com rede de comunicação, utilizam-se os blocos de rede TrueTime. Estes blocos simulam o acesso ao meio e a transmissão de pacotes em uma rede local (LAN). Os protocolos de rede suportados são:

- CSMA/CD, por exemplo *Ethernet*;
- CSMA/AMP, por exemplo *CAN*;
- *Round Robin*, por exemplo *Token Bus*;
- FDMA;
- TDMA;
- Ethernet comutada;
- IEEE 802.11b/g (WLAN);

- IEEE 802.15.4 (ZigBee).

É possível definir a taxa de transmissão, o tamanho mínimo da mensagem, os WCET's e a probabilidade de perda da mensagem durante a transmissão. Para o caso com rede *wireless*, é possível estipular a potência de transmissão, o *threshold* do sinal receptor e o limite de retransmissão.

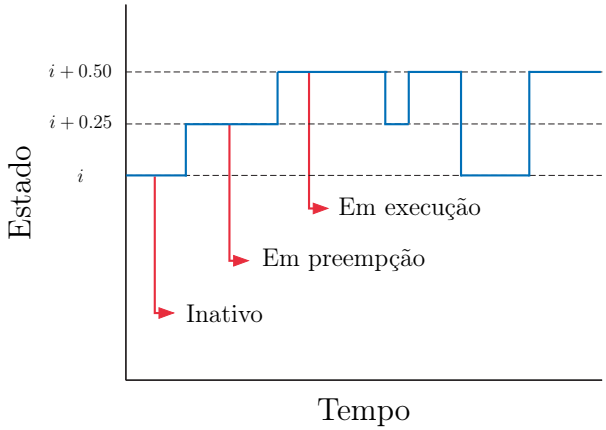


Figura 2.12: Exemplo de um diagrama de escalonamento, que mostra os 3 estados das tasks (escalonamento de *tasks*) ou dos nós da rede (escalonamento da rede), através dos nível de amplitude i (inativo), $i + 0.25$ (em preempção) e $i + 0.50$ (em execução), com $i = 1, \ldots, N$, onde N é o números de *tasks* ou de nós da rede.

O TrueTime fornece diagramas de escalonamento como exemplificado na Figura 2.12, nos quais são observados três estados das *tasks* (ou processos) e/ou dos nós da rede, em função do tempo. Os estados inativo, em preempção e em execução são representados, respectivamente, por níveis de amplitude i , $i + 0.25$ e $i + 0.50$, com $i = 1, \ldots, N$, onde N é o número de *tasks* ou de nós da rede. No estado inativo, a *task* ou o nó não requerem recursos computacionais (memória, CPU, meio físico da rede); no estado de preempção, eles aguardam a disponibilidade dos recursos; e no estado de execução, eles utilizam os recursos. Através destes diagramas, pode-se avaliar o uso dos recursos computacionais, detectando, por exemplo, o fenômeno de *starvation*[2].

Além dos diagramas, o TrueTime fornece logs[3] que fornecem informações sobre os atrasos do sistema, que são utilizadas na análise com o Jitterbug (CERVIN et al., 2006). Desta forma, a análise de desempenho do NCS não fica restrita ao caso de atraso constante, ou mesmo à análise do pior caso.

[2] *Starvation* ocorre quando um processo nunca é executado ("morre de fome"), pois processos de prioridade maior sempre o impedem de ser executado (TANENBAUM, 2003).

[3] Logs são registros de atividades das *tasks*.

2.3 Exemplo Introdutório

Para exemplificar e comparar os métodos apresentados durante o capítulo, toma-se um sistema com uma planta

$$G(s) = \frac{4}{s^2 - 4}$$

e um controlador

$$K(s) = \frac{0.894\ 10^4 s^2 + 4.21\ 10^4 s + 3.586\ 10^4}{s^3 + 88.2s^2 + 3896.0s - 137.9},$$

cuja forma em tempo discreto é

$$K_d(z) = \frac{59.71z^{-1} - 108.90z^{-2} + 49.45z^{-3}}{1.00 - 1.13z^{-1} + 0.20z^{-2} - 0.07z^{-3}},$$

com período de amostragem $h = 30ms$.

Utilizando o modelo do sistema com atraso, a planta é representada por

$$G'_d(z) = \begin{bmatrix} C & 0 \end{bmatrix} \left(zI - \begin{bmatrix} \Phi & \Gamma_1 \\ 0 & 0 \end{bmatrix} \right)^{-1} \begin{bmatrix} \Gamma_0 \\ 0 \end{bmatrix} + D$$

onde

$$\begin{aligned} \Phi &= \begin{bmatrix} 1.002 & 0.120 \\ 0.030 & 1.002 \end{bmatrix} \\ \Gamma_0 &= \begin{bmatrix} 0.0090 \\ 0.0000 \end{bmatrix} \\ \Gamma_1 &= \begin{bmatrix} 0.0210 \\ 0.0004 \end{bmatrix}. \end{aligned}$$

O diagrama de Bode do sistema em malha fechada é visto na Figura 2.13. O comportamento do sistema com atraso tem maior amplitude de oscilação, com freqüência de ressonância $\omega_n \simeq 8\ rad/s$. Utilizando a ferramenta TrueTime, é possível simular o sistema composto pelo controlador discreto ($K_d(z)$) e pela planta contínua ($G(s)$) para o caso com e sem atraso. O resultado é visto na Figura 2.14, que está de acordo com os resultados obtidos no diagrama de Bode: movimento oscilatório de freqüência ω_n.

Utilizando o *toolbox* Jitterbug e considerando os casos do sistema sem atraso e com atraso constante, é possível obter os espectros de freqüência dos sinais de saída como observado na Figura 2.15. Nota-se que este resultado apresenta certa semelhança com o

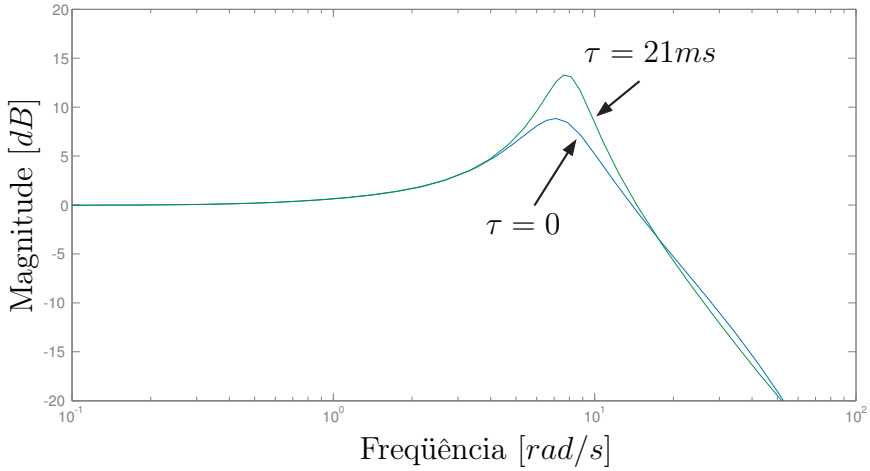


Figura 2.13: Diagrama de bode do sistema com $\tau = 0$ e $\tau = 21ms$.

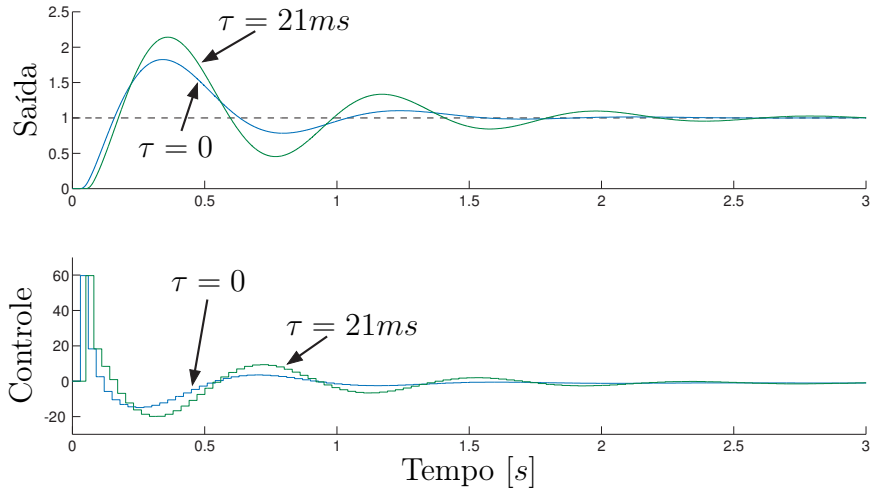


Figura 2.14: Resposta no tempo do sistema com e sem atraso $\tau = 21ms$, com a planta contínua ($G(s)$) e o controlador discreto ($K_d(z)$).

apresentado na Figura 2.13, indicando a concordância entre estes dois métodos. Para o caso sem atraso a função de custo tem valor $J = 7.1687$; e para o caso com $\tau = 21ms$, $J = 9.9541$, indicando que ambos os sistemas são estáveis no sentido quadrático médio (J finito), porém o desempenho do sistema com atraso é pior. Neste simples caso, não foi necessário utilizar o TrueTime para obter a distribuição do atraso, visto que foi adotado como constante para permitir a comparação entre os métodos. Porém como explicado anteriormente, o Jitterbug não se limita a fazer análise somente para casos com atraso constante.

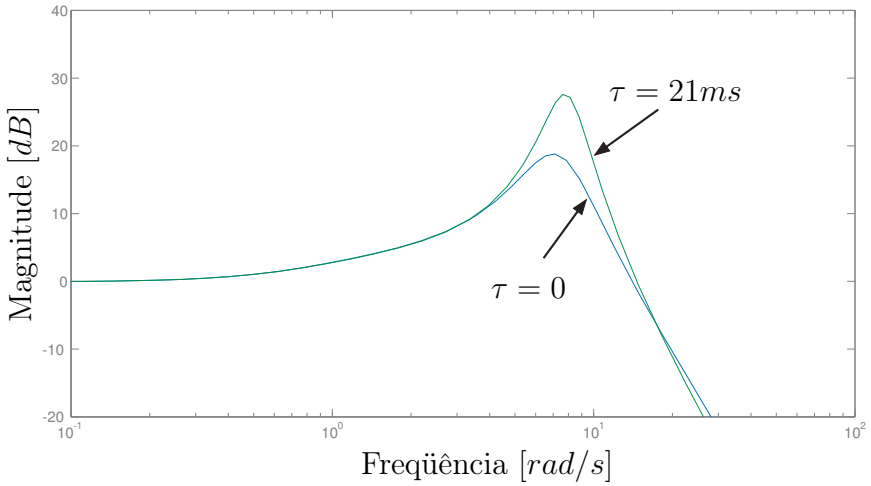


Figura 2.15: Densidade espectral do sinal de saída do sistema com e sem atraso $\tau = 21ms$, obtido através do Jitterbug.

2.4 Conclusão

Neste capítulo foram apresentados os métodos que serão utilizados neste trabalho para analisar o desempenho de sistemas de controle quando estes sofrem efeitos de atrasos aleatórios. O primeiro método envolve o uso de um modelo de um sistema com atraso, que permite fazer a análise através das teorias clássicas de controle. Embora este método seja mais simples, a análise é feita sempre considerando o pior caso de atraso, podendo assim ser um tanto quanto conservador. Já o outro método envolve o uso de dois *toolboxes* (TrueTime e Jitterbug) que, através das informações do sistema computacional fornecem as distribuições dos atrasos aleatórios. Estas distribuições são utilizadas na análise estocástica de desempenho e estabilidade do sistema com atrasos aleatórios através da teoria de MJLS. Através de um exemplo foi possível verificar a eficácia e a concordância entre os métodos.

3 Redes de Comunicação para Sistemas de Controle

A arquitetura tradicional de redes de comunicação para sistemas de controle que vem sendo implantada durante décadas é a do tipo ponto-a-ponto (*peer-to-peer*), ou seja, um cabo conecta o computador do controle central a cada sensor e atuador. Entretanto o aumento da complexidade destes sistemas está empurrando este tipo de arquitetura para o seu limite (LIAN, 2001). Assim, o sistema de controle ponto-a-ponto centralizado vem sendo substituído por sistemas descentralizados através de uma rede de comunicação comum à todos os seus componentes.

A introdução deste tipo de arquitetura pode aumentar a eficiência, a flexibilidade e a confiabilidade destas aplicações integradas e reduzir os custos e o tempo de instalação, de reconfiguração e manutenção. Entretanto, esta mudança introduz atrasos e perdas nas transmissões dos sinais de controle.

Os sinais de controle em NCSs são transmitidos por pequenas porém freqüentes mensagens que trafegam entre inúmeros nós da rede. Em princípio, estas mensagens devem respeitar os requisitos de tempo de STRs. Esta é a principal característica de redes de comunicação utilizadas em sistemas de controle. Em outras aplicações de redes de comunicação (e.g., Internet), o tamanho, a periodicidade das mensagens e o tempo de envio/recebimento não são fatores críticos para a correta operação do sistema.

A adoção de redes de comunicação em sistemas de controle possibilita a distribuição das funções de processamento e carga computacional entre algumas pequenas unidades de processamento. Esta distribuição do processamento pode tornar o sistema mais robusto visto que um sistema centralizado está sujeito à um único ponto de falha. Além disso, a rede permite que o sinal proveniente de um sensor seja disponibilizado para vários dispositivos ao mesmo tempo sem a necessidade de duplicação de sensores.

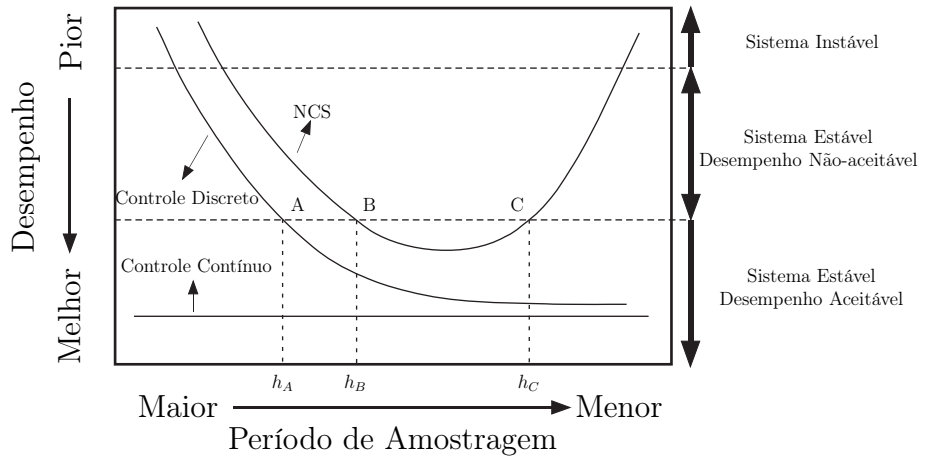


Figura 3.1: Comparação de desempenho do controle para os casos de controle contínuo, controle discreto e NCS em função do período de amostragem adotado.

Como já visto, o objetivo principal do projeto de NCSs é garantir que, mesmo com perdas e atrasos dos sinais de controle, o sistema atenda às suas especificações de desempenho, que incluem margens de fase e de ganho, máximo sobressinal, erro em regime, entre outros. E uma forma para atingir (parte) deste objetivo é analisar o efeito de atrasos em NCS, como observado no Capítulo 2.

Uma forma de apresentar os resultados das análises é através de um gráfico semelhante ao da Figura 3.1. Neste tipo de gráfico os desempenhos do sistemas de controle são dados em função do período de amostragem adotado. Neste gráfico, especificamente, são comparados três casos: sistema de controle em tempo contínuo, em tempo discreto e NCS. No caso contínuo, o desempenho do sistema de controle é independente do período de amostragem. Já para o caso discreto, percebe-se que um maior período de amostragem degrada o desempenho do sistema de controle, podendo levar o sistema para região de instabilidade. No caso de NCS, um maior período de período de amostragem também degrada o desempenho do sistema de controle. Porém, com um menor período de amostragem aumenta-se a carga na rede, aumentando assim a probabilidade de perdas e de atrasos dos sinais de controle. Desta forma, um menor período de amostragem também degrada o desempenho do sistema de controle.

Para realizar este tipo de análise, é necessário conhecer os fatores que impactam o desempenho do sistema de controle e como variam em função da rede de comunicação adotada. Este é o objetivo de trabalhos como Montestruque e Antsaklis (2005a), Lian (2001) e Nilsson (1998) que analisam este comportamento através de modelos de redes de comunicação. Os fatores que impactam no desempenho de sistemas de controle incluem:

- o atraso para acesso ao meio – tempo necessário para que o dispositivo tenha acesso ao meio físico da rede;
- o tempo de transmissão – tempo necessário para que a mensagem percorra o meio físico, que depende da velocidade de transmissão e do tamanho das mensagens;
- porcentagem de colisões – relação entre o número de mensagens que acessam o meio físico ao mesmo tempo e o número total de mensagens;
- porcentagem de perda de mensagens – relação entre o número de mensagens que não são transmitidas com sucesso e o número total de mensagens.

Para sistemas de controle, as candidatas à rede de comunicação precisam basicamente atender a dois critérios principais: atrasos limitados e garantia de transmissão. Ou seja, as mensagens devem ser transmitidas com sucesso em um tempo limitado. A perda de mensagens e/ou atrasos na transmissão deterioram o desempenho de NCSs e até podem torná-los instáveis (WALSH et al., 2002).

Assim, as redes de comunicação podem ser classificadas como: redes de controle e redes de dados. A diferença entre as duas é que a rede de controle tem requisitos de tempo real (o tempo de resposta pré-determinado deve ser obedecido) (LIAN, 2001). Além disso, em redes de controle as mensagens são, geralemente, pequenas (em número de bits) e periódicas; diferentemente de redes de dados, cujas mensagens são, geralmente, grandes (em número de bits) e aperiódicas. O objetivo deste capítulo é apresentar as principais candidatas a rede de controle (Ethernet, ControlNet e DeviceNet) mostrando as vantagens e as desvantagens de sua implementação em NCSs, na Seção 3.1. Na Seção 3.2, são comparados parâmetros relevantes para NCSs destas possíveis candidatas.

3.1 Candidatas à rede de controle

As principais candidatas à rede de comunicação para NCSs consideradas neste trabalho são aquelas cuja implementação em sistemas de controle tem sido bem sucedida nos últimos anos, a saber: Ethernet, ControlNet e DeviceNet. É importante salientar que não estão sendo desconsiderados outros tipos de rede para uso em sistemas de controle, apenas constata-se que os três tipos citados são os mais aplicados.

3.1.1 Ethernet

A Ethernet é tida como a solução mais prática para NCSs devido ao seu baixo custo, a sua flexibilidade (e.g., facilidade de inclusão nós na rede) e a suas altas taxas de transmissão. Em contrapartida, a natureza não-determinística do seu mecanismo de detecção de colisões (CSMA/CD) é sua maior desvantagem para este tipo de implementação, ou seja, não há garantias (teoricamente) com relação ao tempo de resposta.

Portanto, se os atrasos são não-determinísticos, por que considera-se a Ethernet como uma candidata a rede de controle? Apesar de teoricamente os atrasos serem não-determinísticos, na prática observa-se que, para redes Ethernet com poucos nós, as altas taxas de transmissão proporcionam ao sistema atrasos relativamente pequenos (OTANEZ et al., 2002).

Dado que neste tipo de rede, para cada colisão os nós esperam um tempo aleatório para reenviar as mensagens, é natural relacionar o aumento do número de colisões (decorrente de um aumento do tráfego) com maiores atrasos e, conseqüentemente, com pior desempenho do sistema de controle. O trabalho de (OTANEZ et al., 2002) mostra através de relações empíricas que os atrasos são diretamente proporcionais a taxa de colisão na rede. E da mesma forma, este mesmo trabalho também mostra que, para pequenas cargas na rede, o desempenho do sistema de controle (medido pelo critério IAE) é diretamente proporcional ao uso da rede (medido em relação à sua largura de banda).

Assim conclui-se que o uso da Ethernet em NCSs é possível, mesmo que os atrasos envolvidos sejam de caráter não-determinísticos. Para tanto, é necessário que os atrasos sejam minimizados através do uso mais eficiente da banda da rede e da diminuição das colisões das mensagens no meio. Para isto, recomenda-se que o uso da rede não ultrapasse 25% da sua largura de banda (OTANEZ et al., 2002), em sistemas de controle. Outra possibilidade é utilizar a chamada Ethernet Comutada (*Switched* Ethernet) que permite uma conexão direta entre o nó emissor e o nó receptor, eliminando o problema de colisões (KAPLAN, 2001).

3.1.2 ControlNet

A rede do tipo ControlNet é tida como determinística pois o tempo de espera de um nó para envio da mensagem tem um limite máximo. Este tipo de rede utiliza o mecanismo de detecção de colisão *token bus*, onde o nó com o *token* transmite a mensagem até que

esta acabe ou até que o seu período de transmissão acabe. Então o nó passa o *token* ao seu sucessor lógico (não necessariamente seu "vizinho"físico). Se o nó não tem nenhuma mensagem para transmitir, ele apenas passa o *token* para seu sucessor lógico. Neste tipo de rede não ocorrem colisões, visto que só um nó pode transmitir de cada vez.

Portanto neste caso os atrasos são limitados, tornando a rede determinística. E mesmo que o protocolo *token bus* seja eficiente e determinístico para altas cargas na rede, para baixas cargas o desempenho da rede é menor quando comparado ao desempenho de redes com outros protocolos. Em geral, quando há muitos nós na rede, uma porcentagem alta do tempo de rede (tempo do processo de transmissão de mensagens) é consumida no processo de transição do *token* entre os nós, para os casos de baixas cargas (KOUBIAS; PAPADOPOULOS, 1995). Outro problema é a transmissão de mensagens grandes, visto que estas para serem totalmente transmitidas precisam que o *token* passe algumas vezes pelo nó emissor, aumentando consideravelmente o tempo de rede.

3.1.3 DeviceNet (CAN)

CAN é um protocolo de comunicação serial desenvolvido principalmente para aplicações na indústria automotiva porém oferece bom desempenho em aplicações industriais de tempo-crítico (LIAN; MOYNE; TILBURY, 2002). Este protocolo é otimizado para pequenas mensagens e usa o mecanismo CSMA/AMP para controle de acesso ao meio. Neste caso, o protocolo é orientado a mensagem, e cada mensagem tem uma prioridade específica que é usada para ordenar o acesso ao meio em caso de colisões. No caso de dois nós iniciarem a transmissão de suas mensagens ao mesmo tempo, aquele que tiver maior prioridade vai dar seguimento no processo de envio. Com isso a transmissão não é interrompida.

A rede do tipo DeviceNet é de custo relativamente baixo e tem sido bem aceita em aplicações industriais (LIAN; MOYNE; TILBURY, 2002). Além disso, a vantagem principal deste tipo de rede para NCSs é o seu determinismo, visto que para mensagens de alta prioridade, o atraso na transmissão pode ser garantido. Porém, a maior desvantagem da rede CAN é sua baixa taxa de transmissão (máximo $500Kbps$), que é menor do que outros tipos de rede. Outro fator limitante para a sua aplicação é a distância física máxima permitida entre os nós (menor do que em outros tipos de rede). E este tipo de rede não é recomendada para transmissão de grandes mensagens.

3.2 Comparação entre as Candidatas

Alguns parâmetros das candidatas são apresentados na Tabela 3.1. Segundo o trabalho de Lian (2001), para pequenas mensagens (menor que $10bytes$) a rede Ethernet precisa de mais *overhead*[1] do que ControlNet e DeviceNet. Entretanto, para mensagens longas, DeviceNet requer mais *bits* para transmitir a mesma informação.

	Ethernet	ControlNet	DeviceNet
Tx. de transmissão $[Mbps]$	10	5	0.5
Máx. distância $[m]$	2500	1000	100
Máx. tamanho $[byte]$	1500	504	8
Máx. número de nós	> 1000	99	64

Tabela 3.1: Parâmetros das redes de controle.

O referido trabalho também apresenta o tempo de rede em função do tamanho das mensagens. Conclui-se que o tempo de rede para DeviceNet é maior do que nas outras redes, independentemente do tamanho das mensagens. Isto pode ser explicado pela sua baixa taxa de transmissão. Para pequenas mensagens, o tempo de rede para ControlNet é menor que para Ethernet. E esta situação se inverte para mensagens maiores. Isto é justificado pelo tamanho dos respectivos *overheads*.

3.3 Conclusão

O aumento da complexidade de sistemas de controle distribuídos cria necessidades para a implantação de novas soluções de rede de comunicação (diferentes de P2P) que aumentam a eficiência, a flexibilidade, a confiabilidade, a robustez (contra falhas de *hardware*), etc dos sistemas de controle. Porém, como visto ao longo deste capítulo, para implementação em NCSs, as redes de comunicação (redes de controle) devem atender à requisitos de tempo real, ou seja, o tempo de resposta pré-determinado deve ser obedecido.

Neste capítulo foram descritos três possíveis tipos de rede para NCSs: Ethernet, ControlNet e DeviceNet. Cada qual com suas vantagens e desvantagens. Um resultado prático visto foi o uso da Ethernet em NCSs, mesmo não sendo considerada (teoricamente) uma rede de comunicação determinística. E comparou-se parâmetros relevantes para NCS

[1] A informação que acompanhada uma mensagem na rede para garantir sua transferência sem erros para destino desejado.

das três candidatas: taxa de transmissão, máxima distância entre nós, tamanho máximo dos pacotes e número máximo de nós. Portanto, com estes dados pode-se avaliar qual tipo de rede é melhor para cada situação.

4 Estudo de Caso

Normalmente, o controlador é projetado e analisado em tempo contínuo. Atendendo aos requisitos, o controlador é discretizado e analisado através de simulações. Nestas simulações o sistema de controle discreto é considerado ideal, i.e., o ciclo de controle (amostragem, cálculo do sinal de controle e atuação) é executado instantaneamente no momento que a amostragem é efetuada. Esta pode ser uma aproximação válida para sistemas onde o tempo de execução do ciclo é pequeno quando comparado ao seu período de amostragem, de forma que não haja risco de perda de desempenho do sistema de controle.

Porém, como observado no Capítulo 2, existem casos em que é necessário considerar este tempo de execução no projeto do sistema de controle. O trabalho de Lian, Moyne e Tilbury (2002) mostra a existência de uma relação de compromisso entre o desempenho do controlador e o desempenho do sistema computacional, ambas em função do período de amostragem nominal h. Um maior h tende a diminuir a carga computacional, promovendo um melhor desempenho do sistema computacional. E um aumento no valor de h tende a degradar o desempenho do controlador discreto. Portanto no desenvolvimento do sistema de controle computadorizado, esta relação de compromisso deve ser levada em consideração na escolha do período de amostragem nominal.

O objetivo deste capítulo é, através de um exemplo, mostrar esta relação de compromisso através dos métodos apresentados no Capítulo 2. Para isso, na Seção 4.1 é apresentado o modelo da planta e o controlador PID, assim como sua representação em equações de diferenças. Na Seção 4.2 é apresentada a análise com modelo com atraso. Nas Seções 4.3 e 4.4, o sistema é analisado através dos *toolboxes* Jitterbug e TrueTime. Na Seção 4.5 são apresentadas as conclusões obtidas com as análises realizadas ao longo do capítulo.

4.1 O Sistema

Neste estudo de caso será analisado o controle de velocidade de rotação de um motor de corrente contínua representado por

$$G(s) = \frac{1.5}{s^2 + 0.6s + 4.55} = \frac{5}{(s - (-0.3 + i2.11))(s - (-0.3 - i2.11))}, \tag{4.1}$$

cuja resposta no tempo para uma entrada degrau é dada na Figura 4.1.

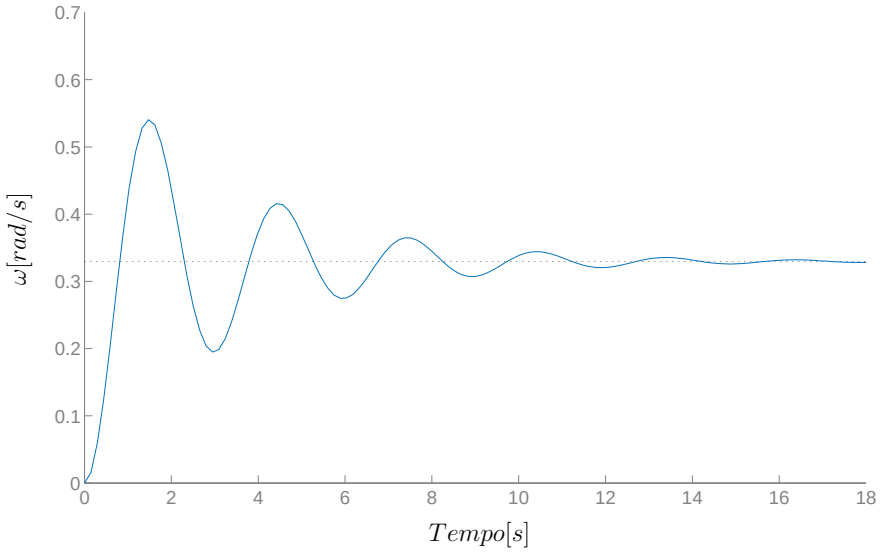


Figura 4.1: Resposta do motor CC para uma entrada degrau unitário.

É adotado para este sistema um controlador do tipo PID, cujo diagrama de blocos é apresentado na Figura 4.2.

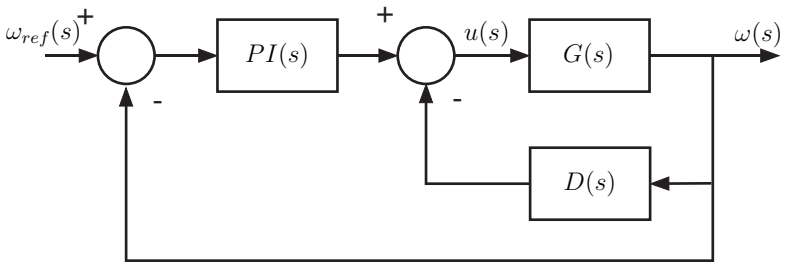


Figura 4.2: Diagrama de implementação do controlador PID adotado, onde $\omega_{ref}(s)$ é o sinal de referência, $u(s)$ é o sinal de controle, $\omega(s)$ é o sinal de saída, $PI(s)$ é o termo proporcional e integral do controlador e $D(s)$ e o termo derivativo do controlador.

O sistema portanto é descrito por

$$\frac{\omega(s)}{\omega_{ref}(s)} = \frac{G(s)PI(s)}{1 + G(s)[PI(s) + D(s)]}, \tag{4.2}$$

onde $PI(s)$ refere-se ao termo proporcional e integral do controlador dado por

$$PI(s) = K\left(1 + \frac{1}{T_i s}\right), \tag{4.3}$$

D(s) refere-se ao termo derivativo dado por

$$D(s) = \frac{KT_d s}{1 + \frac{T_d s}{N}}, \tag{4.4}$$

enquanto que a função de malha aberta é dada por

$$L(s) = \frac{G(s)PI(s)}{1 + G(s)D(s)}. \tag{4.5}$$

Para obter um sistema com amortecimento supercrítico e com uma resposta relativamente rápida, adota-se:

- $K = 72$,
- $T_i = 4$,
- $T_d = 0.16$,
- $N = 32$.

Desta forma, o sistema apresenta margens de ganho $K_g = \infty$ e de fase $\gamma = 68.5°$. A resposta do sistema para entrada degrau no domínio do tempo (Figura 4.3) mostra que o sistema atende aos requisitos, apresentando um tempo de subida[1] $T_r \approx 0.26s$.

A representação do controlador PID em tempo discreto através através de equações de diferenças (ÅSTRöM; HäGGLUND, 1995) é dado por

$$\begin{aligned} P_k &= Ke_k \\ I_{k+1} &= I_k + b_{i1}e_{k+1} + b_{i2}e_k \\ D_k &= a_d D_{k-1} - b_d(y_k - y_{k-1}) \\ u_k &= P_k + I_k + D_k. \end{aligned} \tag{4.6}$$

Nestas equações, P_k, I_k e D_k representam as ações proporcional, integral e derivativa, respectivamente, do controlador no instante k. Os coeficientes b_{i1}, b_{i2}, a_d e b_d são dados em função do período de amostragem h; dos parâmetros K, T_i, T_d e N; e do método

[1]Geralmente, para sistemas superamortecidos, o tempo de subida é definido como o tempo requerido para que a resposta passe de 10% a 90% do seu valor final.

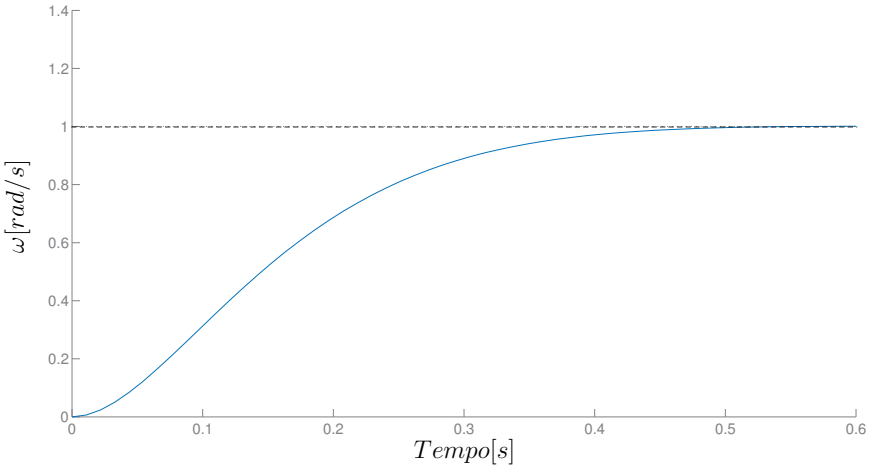


Figura 4.3: Resposta do sistema em malha fechada dado pela Equação (4.2) para uma entrada degrau.

	Diferenças para frente	Diferenças para trás	Tustin
b_{i1}	0	$\frac{Kh}{T_i}$	$\frac{Kh}{2T_i}$
b_{i2}	$\frac{Kh}{T_i}$	0	$\frac{Kh}{2T_i}$
a_d	$1-\frac{Nh}{T_d}$	$\frac{T_d}{T_d+Nh}$	$\frac{2T_d-Nh}{2T_d+Nh}$
b_d	KN	$\frac{KT_dN}{T_d+Nh}$	$\frac{2KT_dN}{2T_d+Nh}$

Tabela 4.1: Coeficientes do controlador PID em tempo discreto, dados em função do método de aproximação.

de aproximação utilizado (diferenças para trás, diferenças para frente, Tustin, etc.), como observado na Tabela 4.1. O sinal de referência é r_k e o sinal de erro é dado por $e_k = r_k - y_k$.

O próximo passo para a implementação de controlador em um sistema digital é a escolha do período de amostragem h para o ciclo de controle. Existem diversas regras práticas para esta escolha. Uma delas apresentada por Åström e Hägglund (1995), estabelece que o período de amostragem h é tal que

$$h = \frac{T_r}{N_r}, \tag{4.7}$$

onde N_r é o número de ciclos de controle por tempo de subida, e pertence ao intervalo de 4 a 10 ciclos. Para este caso, esta regra prática fornece o intervalo $[0.026, 0.065]s$ para a escolha do período de amostragem. Nesta escolha nota-se que existe um limite máximo e um limite mínimo para h. Isto porque um período de amostragem longo prejudica a reconstrução do sinal amostrado deteriorando o desempenho do controlador; e um período

de amostragem muito curto aumenta a carga de utilização do sistema computacional (ÅSTRöM; WITTENMARK, 1997), aumentando a probabilidade de atrasos na realização de suas tarefas (WITTENMARK; NILSSON; TORNGREN, 1995). Deste fato, conclui-se que há uma relação de compromisso entre desempenho do controlador e o desempenho do sistema computacional. Porém, este limite mínimo dado pela regra prática não considera diretamente as informações do sistema computacional sobre o qual o controlador será implementado, podendo ser uma escolha conservadora ou inadequada. Um exemplo deste fato pode ser observado no trabalho de Cervin et al. (2006), onde para alguns sistemas, o período de amostragem mínimo dado pela regra prática torna o sistema instável.

Considerando que este sistema de controle discreto seja ideal, com período de amostragem $h = 0.05s$ e sem a presença de atrasos, a sua simulação através do SIMULINK oferece a saída mostrada na Figura 4.4. Observa-se que o sistema se comporta da maneira especificada no projeto. Portanto, neste caso o projeto do sistema de controle poderia ser considerado satisfatório. Porém, como se mostrará nas próximas seções, a aproximação por um sistema ideal pode não ser suficiente para garantir um bom desempenho do sistema de controle computadorizado.

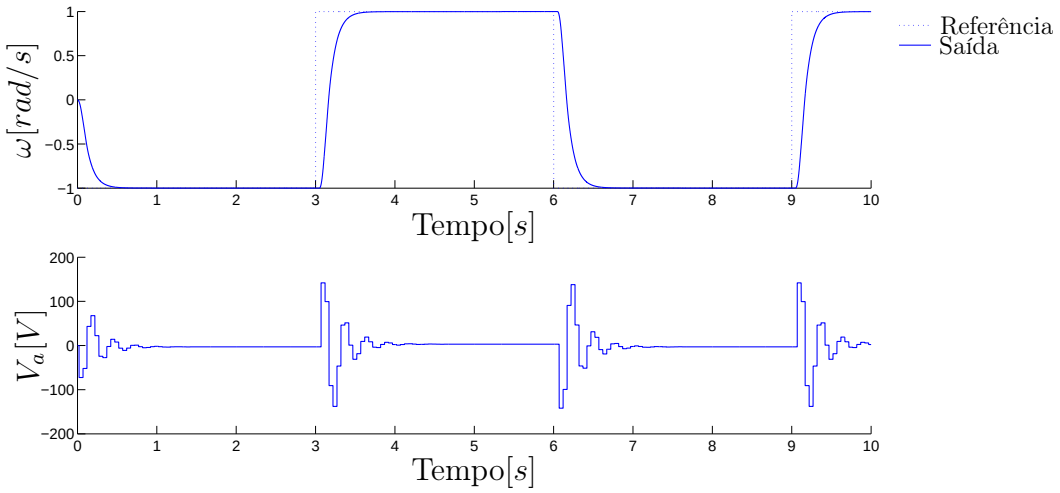


Figura 4.4: Saída ω e sinal de controle V_a para o sistema de controle discreto ideal.

4.2 Análise com Modelo do Sistema com Atraso

Assim como o modelo ideal, o modelo do sistema com atraso é uma aproximação do sistema real pois ele também considera que a amostragem ocorre em um tempo infinitesimal e exatamente nos instantes de amostragem kh. A diferença entre os modelos está no

atraso introduzido para representar o tempo decorrido entre o cálculo do sinal de controle e sua atuação no sistema, como mostrado na Figura 2.7.

A partir da Equação (4.2) e do modelo da planta com atraso

$$G_d(z) = \begin{bmatrix} C & 0 \end{bmatrix} \left[zI - \begin{bmatrix} \Phi & \Gamma_0 \\ 0 & 0 \end{bmatrix} \right]^{-1} \begin{bmatrix} \Gamma_0 \\ I \end{bmatrix}, \tag{4.8}$$

tem-se a função de transferência do sistema em malha fechada em tempo discreto com atraso

$$G_{dcl}(z) = \frac{G_d(z)PI(z)}{1 + G_d(z)[PI(z) + D(z)]}, \tag{4.9}$$

onde $PI_d(z)$ e $D_d(z)$ são as funções de transferência que compõem o controlador PID em tempo discreto. Com esta função de transferência, são obtidos os diagramas de Bode em tempo discreto, ilustrados na Figura 4.5. Foram considerados três casos:

- $h = 0.05s$ sem atraso;
- $h = 0.05s$ com atraso $d = 0.01s$ $(d = 0.20h)$;
- $h = 0.05s$ com atraso $d = 0.03s$ $(d = 0.60h)$.

Nestes diagramas observa-se que conforme o atraso aumenta o sistema em malha fechada adquire um comportamento mais oscilatório; e para o caso com $d = 0.03s$ o sistema apresenta uma freqüência de ressonância $\omega_r \approx 21.0\ rad/s$. Utilizando o modelo SIMULINK da Figura 4.6, onde é introduzido um atraso de transporte na entrada da planta, estes casos são simulados e os resultados são mostrados na Figura 4.7. Nota-se que, assim como esperado, para um tempo de execução do ciclo de controle (ou atraso) pequeno $(d = 0.01s)$ em relação ao seu período de amostragem $(h = 0.05s)$ o sistema se comporta de forma semelhante ao sistema dito ideal (sem atraso); e para o caso com $d = 0.03s$, o sistema oscila na freqüência natural amortecida.

Para fazer a análise em freqüência do sistema para diferentes períodos de amostragem com alguns casos de atraso, parte-se das Equações (2.11), (2.12), resultando na função de malha aberta em tempo discreto

$$L_d(z) = \frac{G_d(z)PI(z)}{1 + G_d(z)D(z)}.$$

Com esta função de malha aberta, são obtidos os diagramas de Nyquist em tempo discreto, ilustrados na Figura 4.8. Nele são apresentados os resultados para os casos:

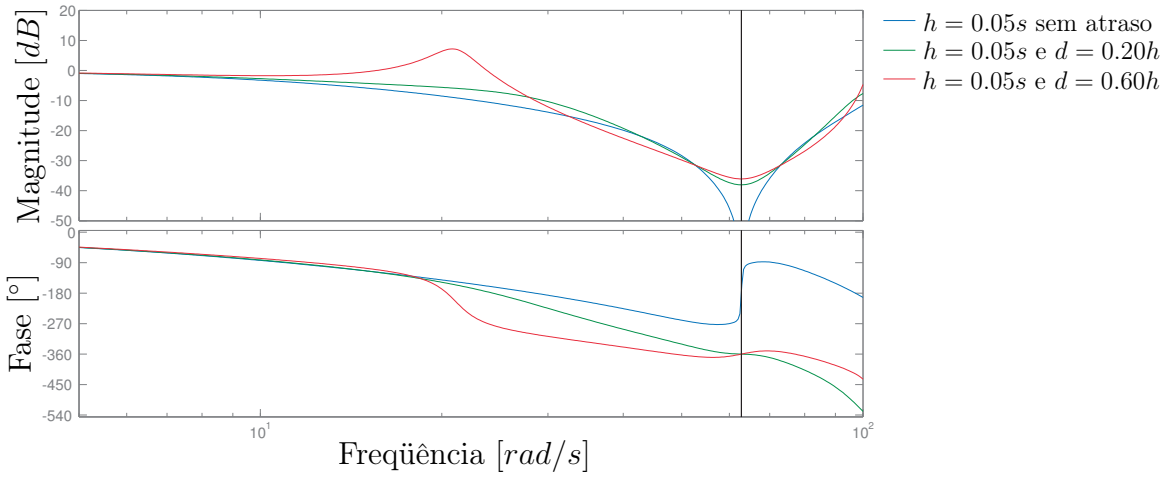


Figura 4.5: Diagramas de Bode obtido através da função de transferência $G_{dcl}(z)$ com o $h = 0.05s$ e alguns casos de atrasos. Observa-se também no diagrama a freqüência de Nyquist (metade da freqüência de amostragem).

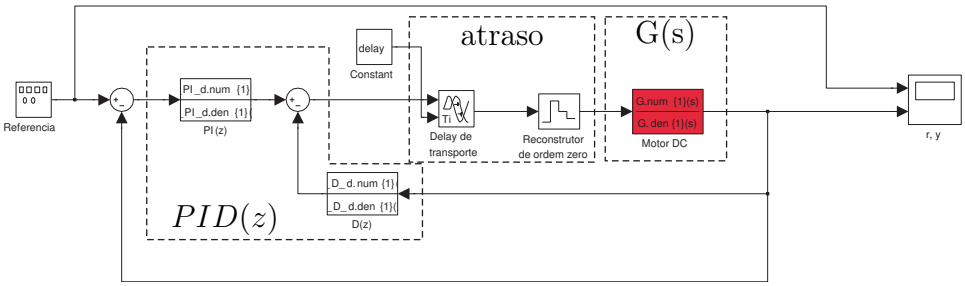


Figura 4.6: Modelo SIMULINK do sistema com atraso. O atraso é representado por um atraso de transporte na entrada da planta com um reconstrutor de ordem zero.

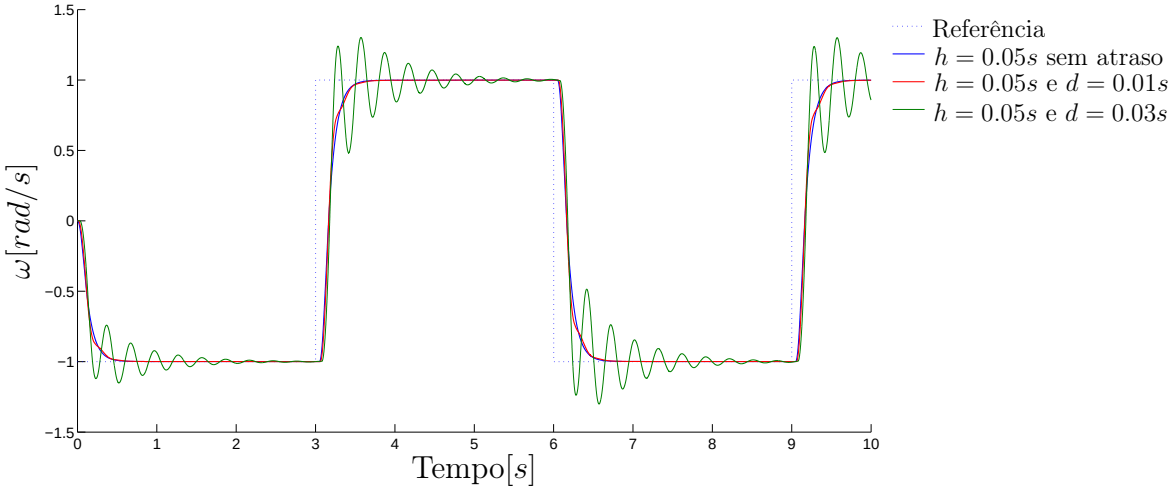


Figura 4.7: Resultado da simulação com o modelo SIMULINK da Figura 4.6 para os casos com $h = 0.05s$ e atrasos $d = 0$, $0.20h$ e $0.60h$.

- $h = 0.026s$ sem atraso;
- $h = 0.026s$ com atraso $d = 0.01s$;
- $h = 0.026s$ com atraso $d = 0.03s$;
- $h = 0.05s$ sem atraso;
- $h = 0.05s$ com atraso $d = 0.01s$;
- $h = 0.05s$ com atraso $d = 0.03s$.

Os valores aqui adotados para o período de amostragem foram obtidos pela regra prática dada pela Equação (4.7). A partir destes diagramas pode-se afirmar que o caso com $h = 0.05s$ e $d = 0.01s$ apresenta um melhor desempenho do que o caso com $h = 0.026s$ e $d = 0.03s$. A resposta no tempo para estes dois casos são observados na Figura 4.9, corroborando com a análise em freqüência. Ou seja, mostra-se que um sistema de controle que apresenta um período de amostragem maior (no caso, $h = 0.05s$) pode ter um desempenho melhor quando comparado ao caso de com um período de amostragem menor (no caso, $h = 0.026s$), dependendo do atraso presente neste sistema.

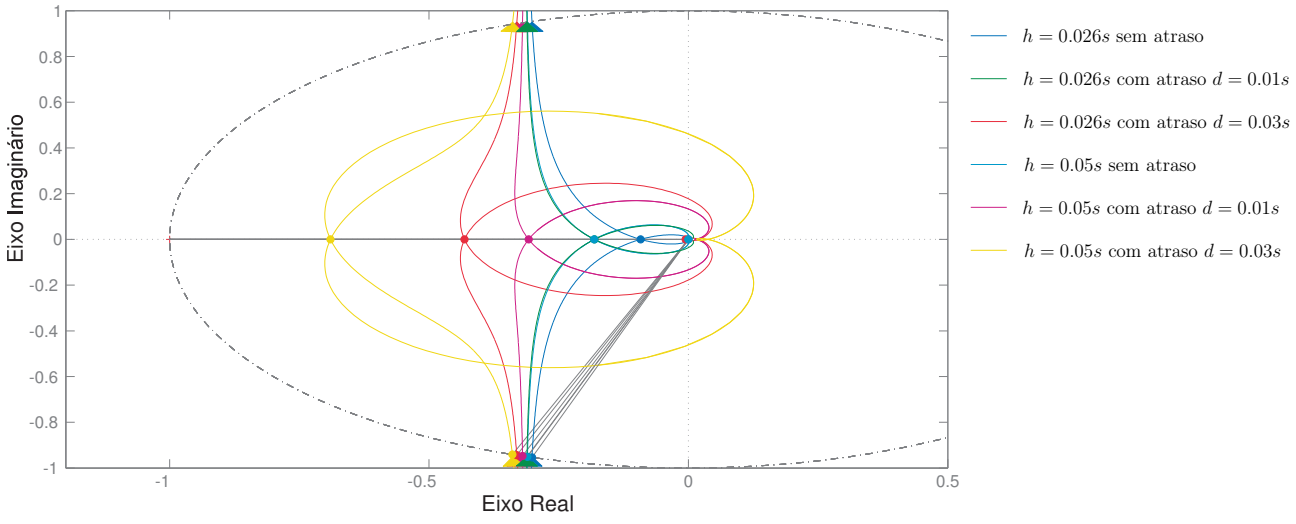


Figura 4.8: Diagramas de Nyquist da função de malha aberta $L_d(z)$ para 6 situações distintas.

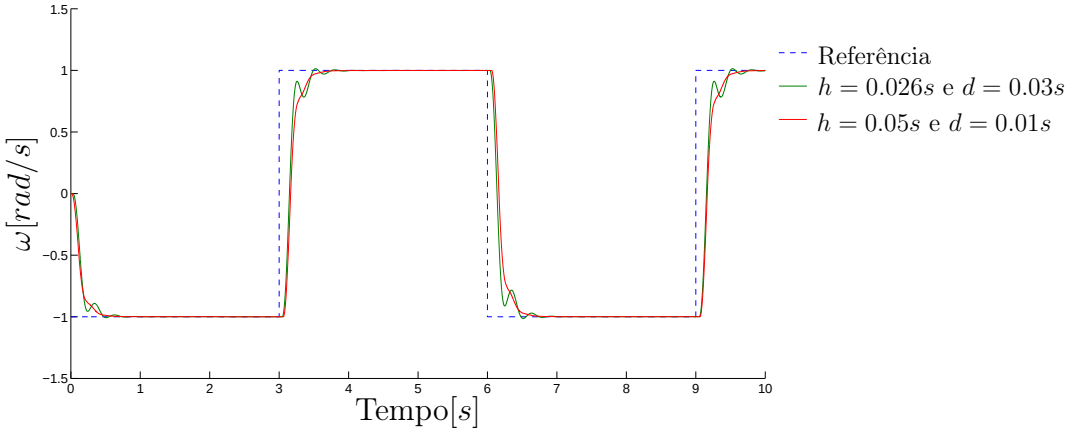


Figura 4.9: Saídas do sistema para $h = 0.05s$ e $d = 0.01s$ e para $h = 0.026s$ e $d = 0.03s$.

4.3 Simulação com o TrueTime

Na seção anterior foi utilizado um modelo de sistema com atraso na análise em freqüência de um sistema de controle computadorizado. Adotou-se como hipótese que o tempo entre o início e o fim do ciclo de controle é definido como uma atraso constante. Porém, sob o contexto de NCS esta hipótese simplificadora podem não ser suficiente (NILSSON, 1998). Nesta seção mostrar-se-á como a presença de atrasos aleatórios influencia no desempenho do NCS, concluindo que existe uma relação de compromisso entre o desempenho do sistema de controle e o desempenho sistema computacional. Para tanto utiliza-se a ferramenta TrueTime e o modelo de NCS observado na Figura 4.10.

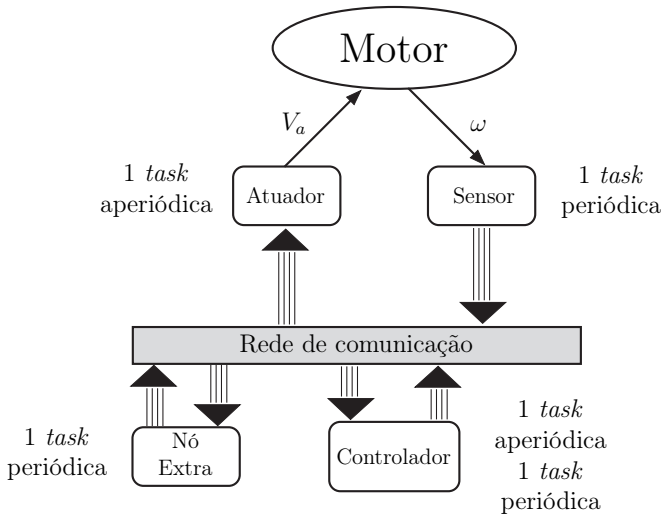


Figura 4.10: Modelo de NCS utilizado nesta seção.

Nesta seção, o ciclo de controle é composto por uma seqüência de três etapas:

- amostragem (ou sensoriamento) e transmissão pela rede de comunicação;
- cálculo do sinal de controle e transmissão pela rede de comunicação;
- atuação do sinal de controle sobre a planta.

O modelo TrueTime (Figura 4.11) deste caso é composto por um NCS para controlar a velocidade do motor CC através de uma rede de comunicação CAN. Nesta rede estão conectados os nós do controlador, do atuador, do sensor e um nó extra que representa outros nós ligados à rede e que não participam diretamente do ciclo de controle. Cada nó é composto por um *kernel*-RT com política de escalonamento de *tasks* por prioridade fixa.

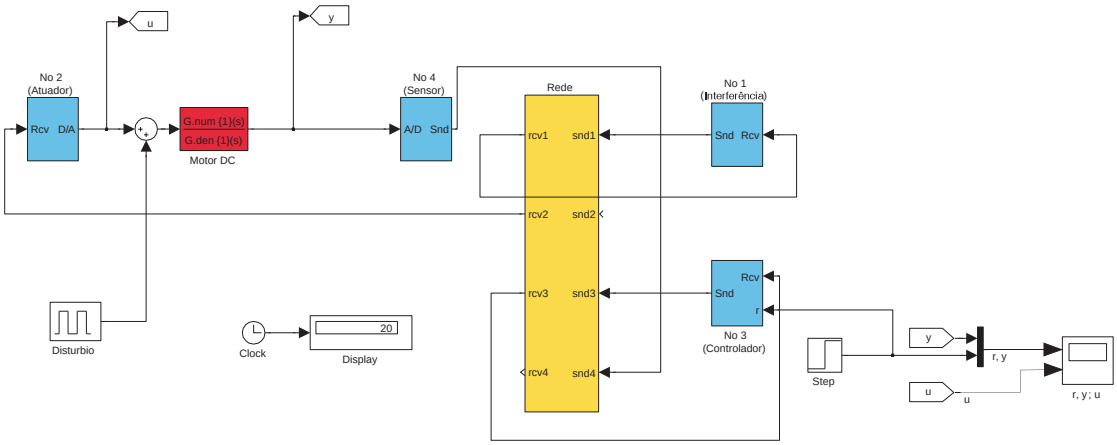


Figura 4.11: Modelo TrueTime do sistema de controle do motor CC.

Para este caso, adota-se um sistema de controle *mono-rate*, ou seja, com somente um período de amostragem nominal h para o ciclo de controle. O ciclo de controle inicia no sensor, onde uma *task* periódica faz a leitura do sinal de saída ω através do conversor A/D e envia esta informação pela rede para o nó do controlador. Quando a mensagem chega no controlador uma interrupção inicia uma *task* aperiódica que contém as equações de diferenças do controlador PID. O sinal de controle então é calculado e enviado para o nó do atuador. Quando a mensagem chega no atuador uma interrupção inicia a *task* aperiódica que transforma a informação em sinal de controle V_a através do conversor D/A de ordem zero. Como o nó extra da rede, é inserida uma *task* extra no *kernel* do controlador representando outras *tasks* que não participam diretamente do ciclo de controle. Os atributos destas *tasks* são dados na Tabela 4.2.

De acordo com o trabalho realizado por Nilsson (1998), numa rede CAN, as mensagens do sensor para o controlador e as mensagens do controlador para o atuador são formadas

Kernel	*Task*	Prioridade	WCET	Período	*Deadline*
controlador	periódica	1	$1s$	$4s$	–
	aperiódica	2	$0.005s$	–	$100s$
sensor	periódica	2	$0.003s$	h	–
atuador	aperiódica	2	$0.003s$	–	$100s$
extra	periódica	1	–	$0.001s$	–

Tabela 4.2: Características das *tasks* do sistema.

por palavras que variam entre 64 a 128 bits, incluindo os bits de endereço e os bits de controle. O trabalho de Chow e Tipsuwan (2003) indica que as mensagens do sensor para o controlador são menores (em números de bits) do que as mensagens do controlador para o atuador. Portanto, para este caso estipula-se:

- mensagens com prioridade 2 e 80 bits, do sensor para o controlador;
- mensagens com prioridade 3 e 120 bits, do controlador para o atuador;
- mensagens com prioridade 1 e 80 bits, do nó extra para ele mesmo.

As mensagens do nó extra são colocadas na rede de forma a ocupar cerca de 80% da banda de forma aleatória, sendo definida para esta rede uma taxa de transferência de 80 $Kbps$, com probabilidade nula de perda de dados.

As grandezas analisadas neste caso são definidas como:

- L_S^k - latência do sensor (tempo decorrido entre o início da amostragem e o início do cálculo);
- L_C^k - latência do controlador (tempo decorrido entre o início do cálculo e o início da atuação);
- h_k - intervalo de amostragem (tempo decorrido entre o final de dois ciclos de controle consecutivos).

Estas grandezas estão ilustradas na Figura 4.12.

São analisados cinco casos: $h = 0.01s$, $0.02s$, $0.04s$, $0.06s$ e $0.07s$. Como no trabalho de Lian, Moyne e Tilbury (2002), para avaliar o desempenho do NCS em cada caso, utiliza-se o critério ITAE dado por

$$ITAE = \int_{t_0}^{t_f} t|e|dt, \tag{4.10}$$

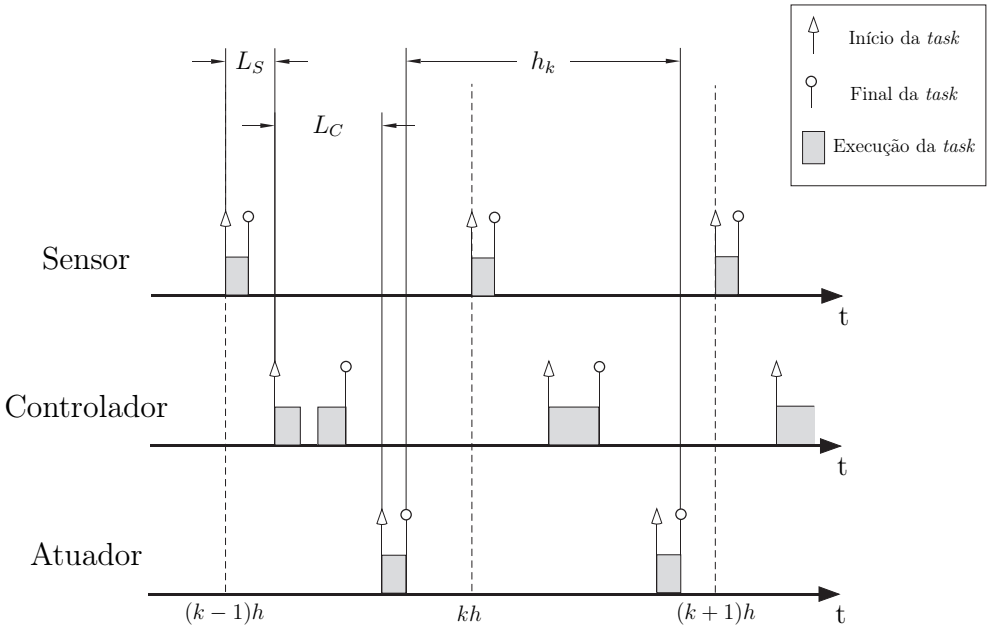


Figura 4.12: Ilustração das grandezas analisadas neste estudo de caso e das *tasks* envolvidas no ciclo de controle. Nota-se que a *task* no controlador pode apresentar um intervalo durante sua execução devido à preempção.

onde t_0 e t_f são os tempos inicial e final do período avaliado e e é a diferença entre o sinal de referência e o sinal de saída. O critério ITAE é uma medida relativa entre desempenho de sistemas de controle, visto que um valor ITAE maior (em relação a um valor ITAE menor) indica que o sistema é relativamente menos estável que outro, de forma que valores muito elevados indicam que o sistema tende a ser instável.

Considerando que os WCETs das *tasks* são independentes do período de amostragem nominal, tem-se:

- o tempo de execução da *task* periódica no sensor $ET_S = 0.004s$;
- o tempo de execução da *task* aperiódica no controlador $ET_C = 0.0065s$.

Atribui-se valores relativamente altos para os *deadlines* das *tasks* aperiódicas para garantir que sejam sempre executadas, respeitando assim a hipótese de não haver perda de dados.

Com estas definições, os casos são simulados. A partir destas simulações são obtidos os diagramas de escalonamento das *tasks* do controlador e da rede, mostrados nas Figuras 4.13 e 4.14, respectivamente. No diagrama de escalonamento das *tasks*, é observada uma situação de preempção das *tasks* de controle. As *tasks* de controle em preempção formam uma fila (FIFO-*First In, First Out*) e são executadas conforme disponibilidade

de recursos (ANDERSSON; HENRIKSSON; CERVIN, 2007). Quanto mais *tasks* entram na fila de preempção, mais tempo é necessário para sua execução (destaque na Figura 4.13), o que implica em maiores latências. E, juntamente com a execução desta fila, ocorre um aumento do tráfego na rede (destaque na Figura 4.14), que tende a aumentar os atrasos entre o envio e recebimento de mensagens. Assim fica evidente duas fontes de aumento das latências: a preempção das tasks de controle e o aumento do tráfego da rede. Portanto conclui-se que conforme aumenta o período de amostragem h:

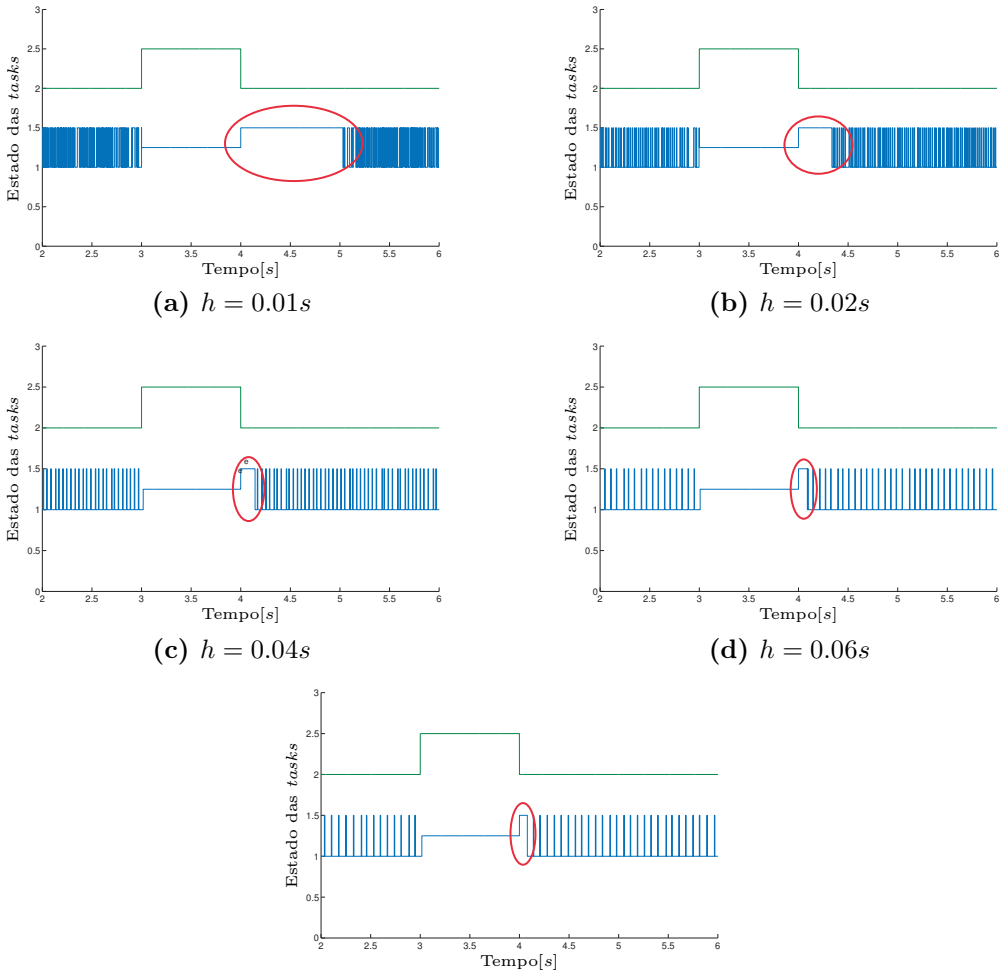
- as latências L_S^k tendem a diminuir, aproximando-se de $ET_S = 0.004s$;
- as latências L_C^k tendem a diminuir, aproximando-se de $ET_C = 0.0065s$);
- os intervalos h_k tendem a se aproximar do período de amostragem nominal h,

como observado nos histogramas nas Figuras 4.15, 4.16 e 4.17.

Portanto para maiores períodos de amostragem nominal h tem-se menores latências, resultando em um melhor o desempenho do sistema computacional. Porém isto dificulta a reconstrução do sinal, prejudicando o desempenho do sistema de controle. Além disso, durante o período entre duas amostragens consecutivas (ou *intersample*), o sistema de controle discreto opera em malha aberta, sendo sensível à distúrbios (ÅSTRöM; WITTENMARK, 1997). Nas simulações são introduzidos distúrbios para observar esta sensibilidade nos *intersamples*. As saídas obtidas nestas simulações são mostradas na Figura 4.18; e sobre elas é aplicado o critério ITAE, resultando nos valores contidos na Tabela 4.3. Percebe-se que o período de amostragem nominal muito curto ou muito longo resulta na piora do desempenho do NCS. Isto porque o desempenho deste sistema depende tanto do desempenho do sistema computacional (melhora com o aumento de h) quanto do desempenho do sistema de controle (piora com o aumento de h). Esta conclusão também é apresentada no trabalho de Lian, Moyne e Tilbury (2002), onde esta relação de compromisso é colocada em evidência.

h	ITAE	Sistema
0.01	$8.997\ 10^{13}$	Instável
0.02	$6.863\ 10^{5}$	Estável
0.04	$7.414\ 10^{4}$	Estável
0.06	$9.700\ 10^{5}$	Estável
0.07	$1.223\ 10^{9}$	Instável

Tabela 4.3: Critério ITAE aplicados aos resultados obtidos através do TrueTime.



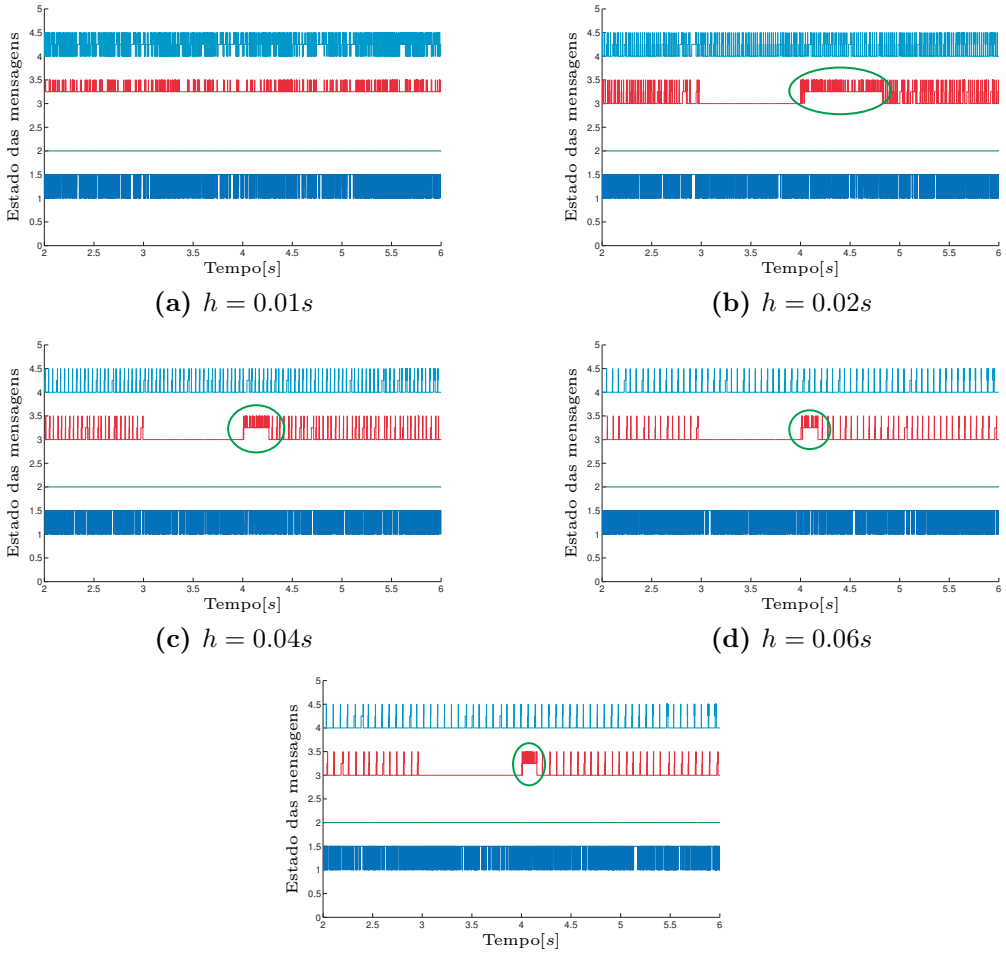
(a) $h = 0.01s$ **(b)** $h = 0.02s$

(c) $h = 0.04s$ **(d)** $h = 0.06s$

(e) $h = 0.07s$

Figura 4.13: Diagramas de escalonamentos das *tasks* do controlador, destacando o intervalo em que as *tasks* de controle em preempção são executadas.

Portanto, para um período de amostragem nominal h dentro do intervalo $[0.02, 0.06]s$ o NCS é estável, segundo o critério ITAE. E segundo este mesmo critério, a escolha ótima é $h = 0.04s$. As distribuições das latências obtidas serão utilizadas na análise com o Jitterbug.

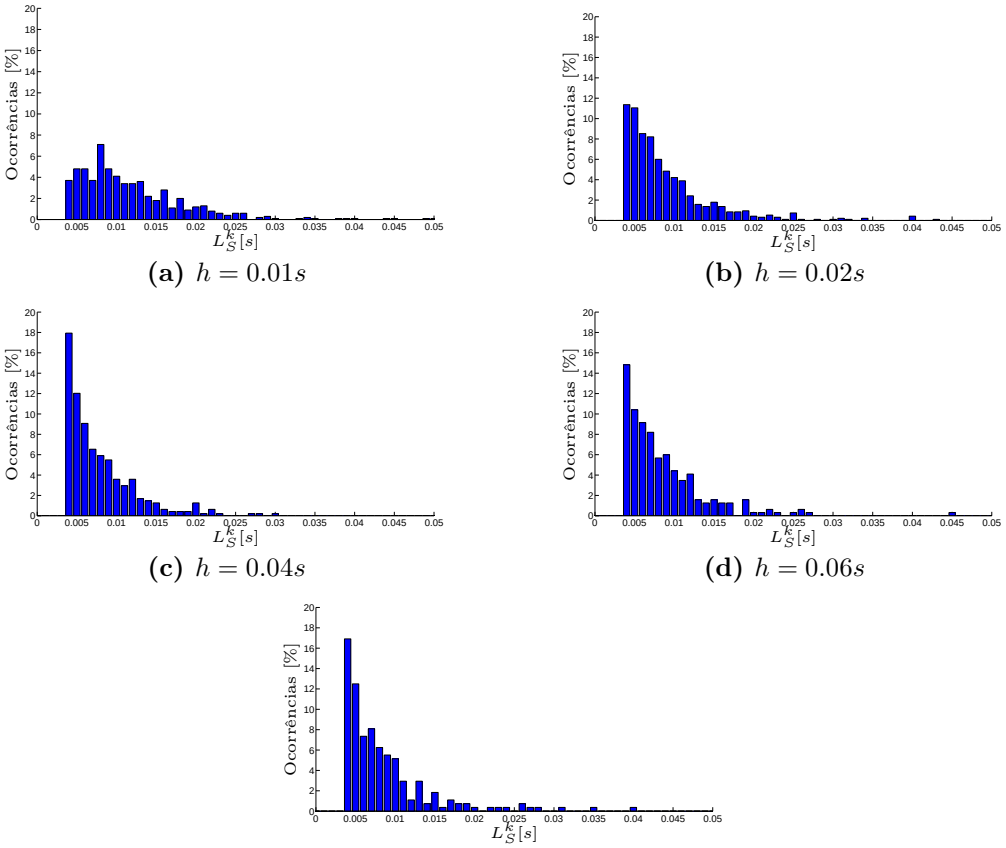


(a) $h = 0.01s$ **(b)** $h = 0.02s$

(c) $h = 0.04s$ **(d)** $h = 0.06s$

(e) $h = 0.07s$

Figura 4.14: Diagrama de escalonamento da rede, destacando o intervalo em que ocorre um aumento no tráfego da rede.

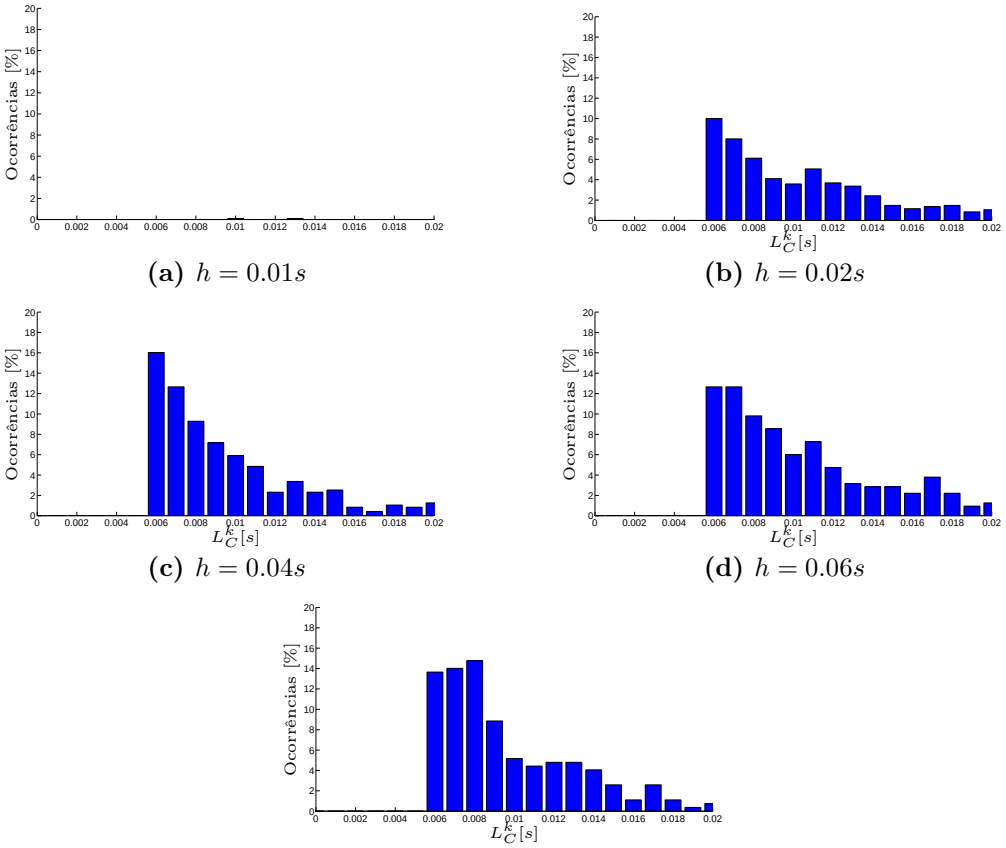


(a) $h = 0.01s$ **(b)** $h = 0.02s$

(c) $h = 0.04s$ **(d)** $h = 0.06s$

(e) $h = 0.07s$

Figura 4.15: Distribuição das latências L_S^k.

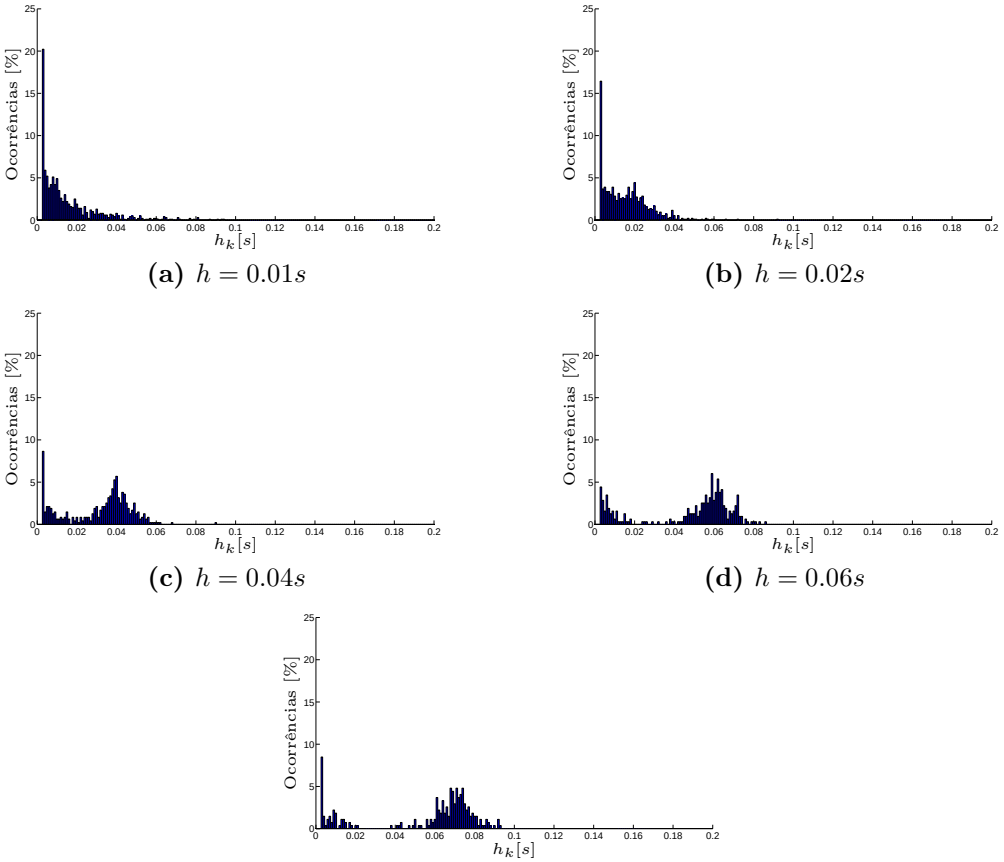


(a) $h = 0.01s$ **(b)** $h = 0.02s$

(c) $h = 0.04s$ **(d)** $h = 0.06s$

(e) $h = 0.07s$

Figura 4.16: Distribuição das latências L_C^k.



(a) $h = 0.01s$ **(b)** $h = 0.02s$

(c) $h = 0.04s$ **(d)** $h = 0.06s$

(e) $h = 0.07s$

Figura 4.17: Distribuição dos intervalos de amostragem h_k.

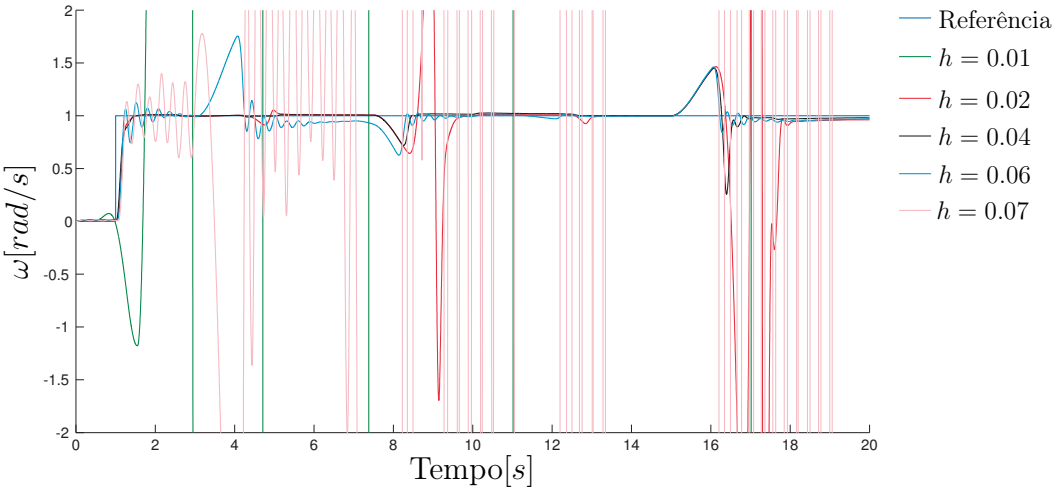


Figura 4.18: Saídas do sistema para h: $0.01s$, $0.02s$, $0.04s$, $0.06s$ e $0.07s$, obtidas com TrueTime.

4.4 Análise com o Jitterbug

O modelo Jitterbug para este caso é semelhante ao modelo da Figura 2.8, onde $H_1(z) = 1$, $H_2(z) = PID(z)$ e $H_3(z) = 1$ representam o nó do sensor, do controlador e do atuador, respectivamente; e $G(s)$ é o modelo do motor. O modelo de tempo é dado por τ_1 e τ_2, que são L_S e L_C, respectivamente, obtidos na Seção 4.3. É inserido na saída da planta um processo de ruído branco em tempo discreto com média nula e covariância $R = 2\pi$. Este modelo é implementado através do Código 4.1 para o cálculo da função de custo do desempenho do NCS.

Código 4.1: *Script* em MATLAB para o cálculo da função de custo do NCS através do Jitterbug.

```
1  Tau_1 = load('L_S');                % Distribuição das latências de
       amostragem
2  Tau_2 = load('L_C');                % Distribuição das latências de cálculo
3  G = load('Motor_CC');               % Planta
4  H1 = 1;                             % Amostrador
5  H2 = load('Controlador');           % Controlador PID discreto
6  H3 = 1;                             % Atuador
7  R = diag([0 2*pi]);                 % Covariância do ruído na saída da
       planta
8  dt = 0.001;                         % Passo de discretização do tempo
9
10 N = initjitterbug(dt,h);            % Inicialização do Jitterbug
11
12 N = addtimingnode(N,1,Tau_1,2);     % Primeiro nó de tempo
13 N = addtimingnode(N,2,Tau_2,3);     % Segundo nó de tempo
14 N = addtimingnode(N,3);             % Terceiro nó de tempo
15
16 N = addcontsys(N,1,G,4);            % Inserção da planta
17 N = adddiscsys(N,2,H1,1,1,[],R);    % Conexão do amostrador ao nó 1
18 N = adddiscsys(N,3,H2,2,2,[],R);    % Conexão do controlador ao nó 2
19 N = adddiscsys(N,4,H3,3,3);         % Conexão do atuador ao nó 3
20
21 N = calcdynamics(N);                % Cálculo das dinâmicas internas
22 J = calccost(N);                    % Cáculo do custo
```

Os resultados obtidos com o Jitterbug (Tabela 4.4) estão de acordo com os resultados obtidos na seção anterior. Verifica-se que existe um intervalo para o período de amostragem nominal do ciclo de controle ($[0.02, 0.06]s$) para o qual o NCS é estável; e existe um

ponto ótimo de operação ($h = 0.04s$).

h	Custo J	Sistema
$0.01s$	∞	Instável
$0.02s$	88.6	Estável
$0.04s$	71.0	Estável
$0.06s$	88.9	Estável
$0.07s$	∞	Instável

Tabela 4.4: Resultados das simulações com o Jitterbug para o caso estudado, com h: $0.01s$, $0.02s$, $0.04s$, $0.06s$ e $0.07s$.

4.5 Conclusão

Neste capítulo estudou-se o caso de um NCS utilizando os métodos apresentados no Capítulo 2. Os resultados obtidos com o modelo com atraso mostram que um sistema de controle que apresenta um período de amostragem maior pode ter um desempenho melhor quando comparado ao caso de com um período de amostragem menor, dependendo do atraso presente neste sistema. Com os resultados obtidos com o TrueTime e o Jitterbug é possível estender esta afirmação, chegando à conclusão que existe uma relação de compromisso entre o desempenho do sistema de controle e o desempenho do sistema computacional com relação à escolha do período de amostragem nominal h, como também mostra o trabalho de Lian, Moyne e Tilbury (2002). Foi apresentado um intervalo para o período de amostragem nominal h ($0.02s$ a $0.06s$) no qual o NCS é estável, bem como um ponto ótimo de operação ($h = 0.04s$).

5 O ROV

O objetivo deste capítulo é apresentar o modelo do ROV considerado neste trabalho (Figura 5.1), além da arquitetura de controle e a estrutura de controle. A descrição detalhada deste veículo (conhecido por VSOR – Veículo Submarino Operado Remotamente) está contida no trabalho de Avila (2008).

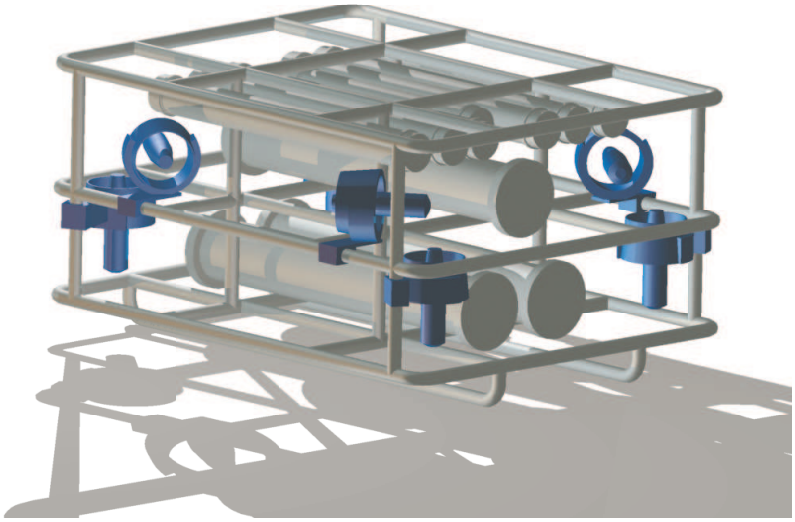


Figura 5.1: Modelo CAD de um ROV.

Sob a hipótese de operações com baixas velocidades, o veículo pode ser modelado através de 6 equações diferenciais desacopladas referentes aos seus 6 graus de liberdade (GLs), como visto na Seção 5.1. Com relação ao controle do veículo, existem níveis hierárquicos de controle, onde o nível superior gera sinais de referência para o nível imediatamente inferior, como visto na Seção 5.2. Particularmete, este trabalho está focado no controle de velocidade do veículo através controladores do tipo proporcional-integral (PI), como visto na Seção 5.3.

5.1 Modelo do VSOR

Assim como em Fossen (1994), utiliza-se dois sistemas de coordenadas (o referencial inercial e o referencial móvel) observados na Figura 5.2 para descrever os movimentos dos veículos marinhos nos 6 graus de liberdade. O sistema de coordenadas fixo ao veículo ($O_mX_mY_mZ_m$) é o referencial móvel, e a origem O_m é usualmente escolhida coincidente ao centro de gravidade (CG) do veículo. O referencial inercial ($O_0X_0Y_0Z_0$) é tomado em um ponto fixo na superfície da Terra. Usualmente, as variáveis de posição e orientação[1] são descritas em relação ao referencial inercial; e as variáveis de velocidade (linear e angular) e aceleração (linear e angular) em relação ao referencial móvel. Estas variáveis e sua representação estão listadas na Tabela 5.1.

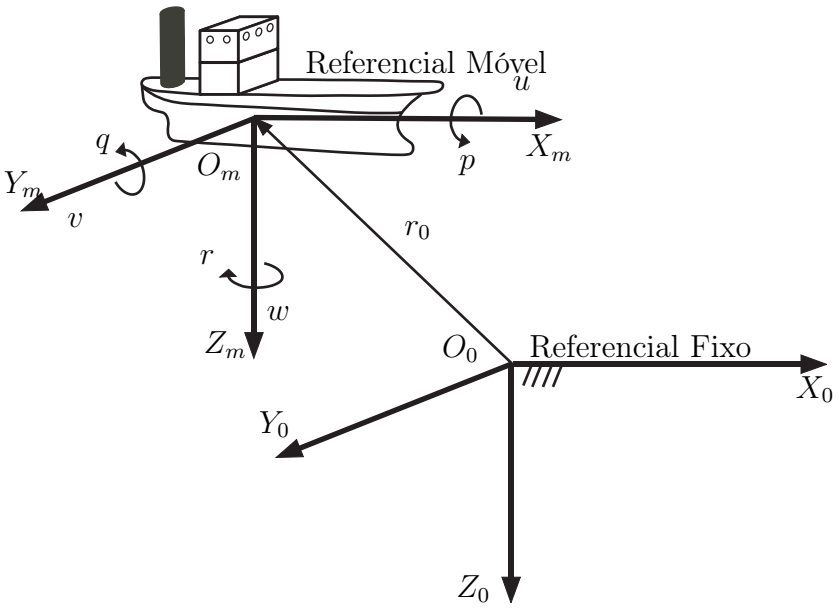


Figura 5.2: Sistema de coordenadas para veículos marinhos, com 6 graus de liberdade (FOSSEN, 1994).

Para definir a relação das variáveis no referencial móvel no referencial inercial, adotam-se os vetores:

$$\begin{array}{rclrclrcl} \eta & = & [\eta_1^T, \eta_2^T]^T; & \eta_1 & = & [x, y, z]^T; & \eta_2 & = & [\phi, \theta, \psi]^T. \\ \nu & = & [\nu_1^T, \nu_2^T]^T; & \nu_1 & = & [u, v, w]^T; & \nu_2 & = & [p, q, r]^T. \\ \tau^{GL} & = & [\tau_L^{GL^T}, \tau_A^{GL^T}]^T; & \tau_L^{GL} & = & [X, Y, Z]^T; & \tau_A^{GL} & = & [K, M, N]^T. \end{array}$$

Estas notações serão utilizadas ao longo deste trabalho e estão de acordo com as encon-

[1]Em engenharia naval, a orientação é usualmente representada pelos Ângulos de Euler (FOSSEN, 1994).

GL	Movimento	Forças e Momentos	Velocidades (linear e angular)	Posições e Ângulos de Euler
1	*Surge* - translação em x	X	u	x
2	*Sway* - translação em y	Y	v	y
3	*Heave* - translação em z	Z	w	z
4	*Roll* - rotação em x	K	p	ϕ
5	*Pitch* - rotação em y	M	q	θ
6	*Yaw* - rotação em z	N	r	ψ

Tabela 5.1: Notações definidas pela SNAME (*The Society of Naval Architects and Marine Engineers*) utilizadas para veículos marinhos.

tradas em Fossen (1994). A velocidade de veículo no referencial inercial é dada por

$$\dot{\eta}_1 = J_1(\eta_2)\nu_1, \tag{5.1}$$

onde $J_1(\eta_2)$ é composição de três rotações consecutivas dada por

$$J_1(\eta_2) = \begin{bmatrix} \cos\psi\cos\theta & -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \sin\psi\sin\phi + \cos\psi\cos\phi\sin\theta \\ \sin\psi\cos\theta & \cos\psi\cos\phi + \sin\phi\sin\theta\sin\psi & -\cos\psi\sin\phi + \sin\theta\sin\psi\cos\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{5.2}$$

Sua inversa é dada pela sua transposta ($J_1^{-1}(\eta_2) = J_1^T(\eta_2)$).

Analogamente, tem-se que

$$\dot{\eta}_2 = J_2(\eta_2)\nu_2, \tag{5.3}$$

onde

$$J_2(\eta_2) = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & -\frac{\sin\phi}{\cos\theta} & \frac{\cos\phi}{\cos\theta} \end{bmatrix}. \tag{5.4}$$

Porém, $J_2^{-1}(\eta_2) \neq J_2^T(\eta_2)$. A matriz inversa de $J_2(\eta_2)$ é dada por

$$J_2^{-1}(\eta_2) = \begin{bmatrix} 1 & 0 & -\sin\theta \\ 0 & \cos\phi & \cos\theta\sin\phi \\ 0 & -\sin\phi & \cos\theta\cos\phi \end{bmatrix}. \tag{5.5}$$

O modelo do VSOR é baseado nos modelos utilizados em Caccia, Indiveri e Veruggio (2000) e em Avila (2008), onde os 6 GLs são representados por 6 equações diferenciais desacopladas sob a hipótese de operações com baixas velocidades. Uma modelagem mais completa, considerando o sistema acoplado, pode ser encontrado em Fossen (1994) e em

Souza (2003), por exemplo.

Para os GLs *Surge*, *Sway*, *Heave* e *Yaw*, a equação é dada por

$$\tau_i(t) = m_i\dot{x}_i(t) + d_{L_i}x_i(t) + d_{Q_i}x_i(t)|x_i(t)| + b_i, \tag{5.6}$$

onde x_i é a velocidade do veículo no GL i dado em relação ao referencial móvel; τ_i é a força/torque aplicada ao veículo no GL i; m_i é a inércia/massa virtual[2] referente ao GL i; d_{L_i} é o coeficiente de arrasto linear e d_{Q_i} é o coeficiente de arrasto quadrático referentes ao GL i e b_i é o modelo de distúrbio para o GL i.

Para veículos com geometria indefinida, do tipo *open frame*, os GLs *Roll* e *Pitch* são basicamente estáveis, sendo estes modelados por

$$\tau_i(t) = m_i\dot{x}_i(t) + d_{Q_i}x_i(t)|x_i(t)| + Bz_B\sin\alpha + b_i, \tag{5.7}$$

onde x_i é a velocidade do veículo no GL i dado em relação ao referencial móvel; τ_i é a força/torque aplicada ao veículo no GL i; m_i é a inércia/massa virtual referente ao GL i; e o termo B é a resultante do empuxo e da força peso, z_B é a altura referente ao centro de flutuação e $\alpha = \phi, \theta$.

Os valores das constantes citadas nesta seção estão contidos no Apêndice A.

5.2 Arquitetura de Controle

A arquitetura do sistema de controle do ROV utilizada atualmente é mostrada na Figura 5.3, também conhecida como arquitetura centralizada. Ela é composta basicamente por dois módulos: o sistema computacional de superfície e o sistema computacional embarcado. O sistema computacional de superfície consiste de um *laptop* executando o sistema operacional Windows XP, sendo responsável pelas tarefas de alto-nível. Ele se comunica com o sistema computacional embarcado através da rede com protocolo Ethernet TCP/IP, usando um cabo 4-26 AWG. O sistema embarcado consiste numa placa padrão PC-104 que executa o RTOS VxWorks da Windriver. Esta placa é composta por um processador NS Geode de $300MHz$, com baixo consumo de energia; portas A/Ds; portas D/As; e portas seriais. Os programas são desenvolvidos e compilados no computador de superfície através do sistema de desenvolvimento Windriver Tornado e enviados atra-

[2]Inércia virtual se refere à soma da inércia do veículo com a inércia resultante do fenômeno de massa adicionada.

Variável	Sensor (fabricante)	Precisão	Taxa de aquisição	Saída
Yaw	Compasso TCM2 (PNI)	$\pm 1°$	$13Hz$	Digital
Roll e *pitch*	Tilt Series 757 (Applied Geomechanics)	$\pm 2°$	$20Hz$	Analógica
Profundidade	Sensor de pressão, MPX5100DP (Motorola)	$3.5cm$	$20Hz$	Analógica
Taxa de *yaw*	Fiber Optic Gyro, E-Core 2000 (KVH)	$Bias < 2°/h$	$10Hz$	Digital
Altura	Altímetro, PA200 (Tritech)	$1mm$	$10Hz$	Digital
Aceleração linear	Central inercial, VG700A (Crossbow)	$Bias < 12mg$	$100Hz$	Digital
Velocidade angular	Central inercial, VG700A (Crossbow)	$Bias < 20°/h$	$100Hz$	Digital
Velocidade linear	Sensor Doppler (NavQuest 600 Micro)	$1mm/s$	$5Hz$	Digital

Tabela 5.2: Sistema sensorial do ROV (AVILA; MARUYAMA; ADAMOWSKI, 2008).

vés da rede Ethernet para o sistema embarcado. As informações sobre o movimento do veículo são obtidas através de sensores listados na Tabela 5.2. Estas informações são processadas e utilizadas no cálculo do sinal de controle. Este sinal de controle age sobre o sistema através do sistema de propulsão, composto por oito propulsores modelo 1021 da Tecnadyne Inc e seus respectivos *drivers* de acionamento.

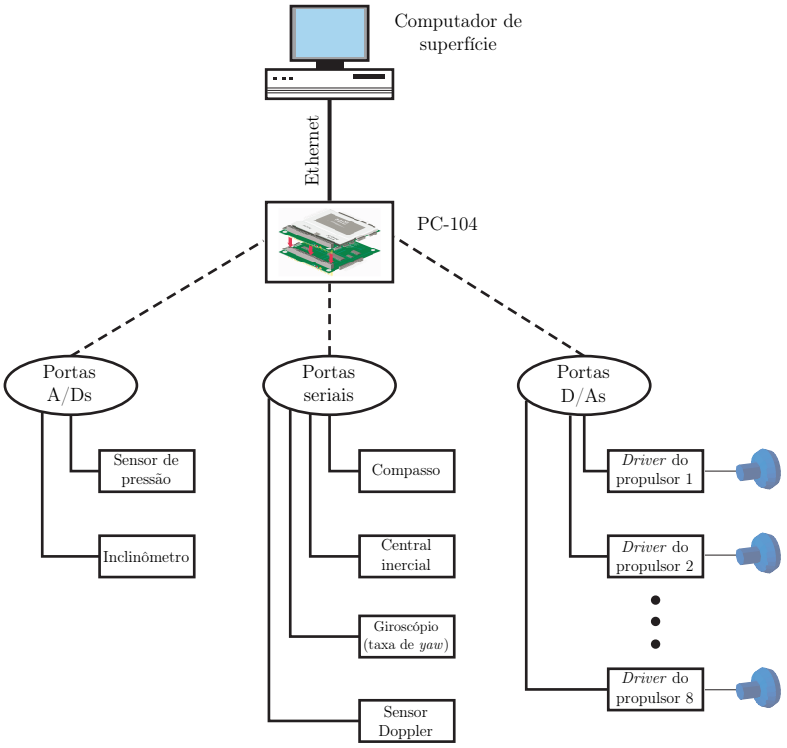


Figura 5.3: Diagrama esquemático da arquitetura atual do sistema de controle do ROV.

O controle do veículo é estruturado hierarquicamente (CACCIA; VERUGGIO, 2000; SOUZA, 2003), onde níveis de controle superiores são responsáveis pela especificação de valores de referências para níveis de controle inferiores. Por exemplo, no caso de uma tarefa de posicionamento, o usuário (o operador humano ou um sistema autônomo) fornece a posição/orientação desejada para o veículo ao nível de controle mais alto, chamado de

"guiagem". A guiagem é responsável por controlar a trajetória do veículo que resultará na desejada posição/orientação através da especificação de valores de referência de velocidades para o próximo nível de controle, chamado de "estratégia de controle". A estratégia de controle é responsável por controlar as velocidades do veículo através de especificação de valores de referência de força/torque aplicados nos 6 graus de liberdade. O sistema de propulsão é responsável por fornecer torque/força nos 6 graus de liberdade através do controle dos propulsores. Esta hierarquia de controle pode ser vista na Figura 5.4. No caso da arquitetura de controle atual, todos os níveis da estrutura hierárquica de controle são implementados no sistema embarcado.

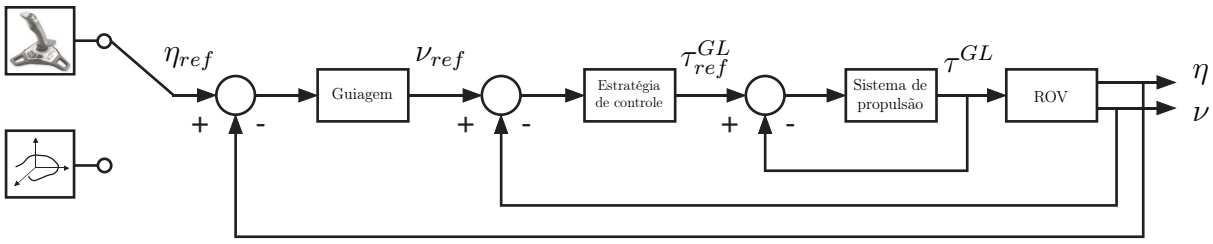
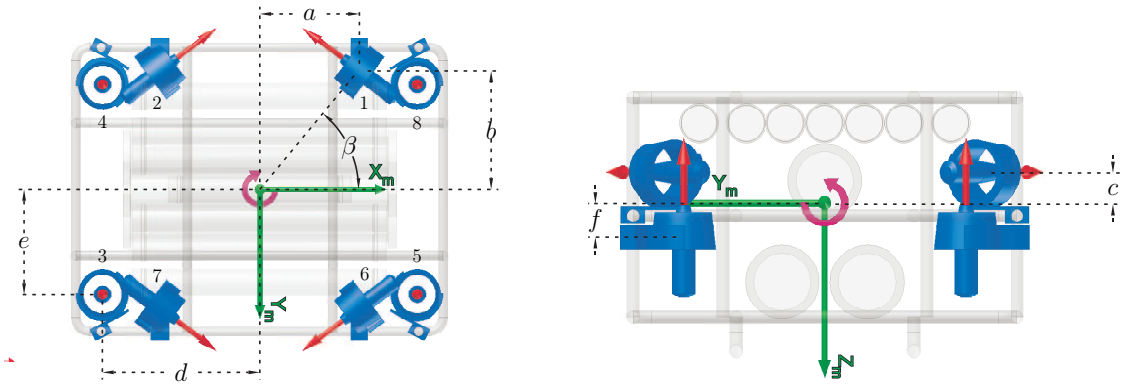


Figura 5.4: Estrutura hierárquica do controle do veículo. As variáveis com índice ref se referem aos valores de referência.

A guiagem e a estratégia de controle calculam o(a) torque/força em cada grau de liberdade para realizar os movimentos definidos pelo usuário. Estes torques/forças dos graus de liberdade precisam ser mapeados nas forças axiais dos propulsores. Isto é feito através de uma matriz de alocação de empuxo, obtida em função do posicionamento dos propulsores em relação ao referencial móvel (Figura 5.5).



(a) Vista superior do veículo. **(b)** Vista frontal do veículo.

Figura 5.5: Posicionamento dos propulsores em relação ao referencial móvel (sistema de coordenadas verde), considerando que o CG do ROV é o ponto de intersecção dos planos de simetria. As forças axiais produzidas pelos propulsores são representadas pelas flechas em vermelho. Os propulsores são identificados por números de 1 a 8.

Seja T_i a força axial gerada pelo propulsor i, e considerando a simetria do veículo e o posicionamento dos propulsores, as forças nos graus de liberdade são dadas por

$$\begin{aligned} X &= (-T_1 + T_2 - T_6 + T_7)\sin(\beta); && (5.8)\\ Y &= (-T_1 - T_2 + T_6 + T_7)\cos(\beta); && (5.9)\\ Z &= -T_3 - T_4 - T_5 - T_8; && (5.10)\\ K &= (T_1 + T_2 - T_6 - T_7)\cos(\beta)c + (-T_3 + T_4 - T_5 + T_8)e; && (5.11)\\ M &= (T_1 - T_2 + T_6 - T_7)\sin(\beta)c + (-T_3 + T_4 - T_5 + T_8)d; && (5.12)\\ N &= (-T_1 + T_2 + T_6 - T_7)\sqrt{a^2 + b^2}. && (5.13) \end{aligned}$$

Na forma matricial, tem-se que

$$\tau^{GL} = B\ T, \tag{5.14}$$

onde B é a matriz de controle e $T = \begin{bmatrix} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 \end{bmatrix}^T$ é o vetor de propulsão. A matriz de controle é, portanto

$$B = \begin{bmatrix} -\sin(\beta) & \sin(\beta) & 0 & 0 & 0 & -\sin(\beta) & \sin(\beta) & 0 \\ -\cos(\beta) & -\cos(\beta) & 0 & 0 & 0 & \cos(\beta) & \cos(\beta) & 0 \\ 0 & 0 & -1 & -1 & -1 & 0 & 0 & -1 \\ \cos(\beta)c & \cos(\beta)c & -e & e & -e & -\cos(\beta)c & -\cos(\beta)c & e \\ \sin(\beta)c & -\sin(\beta)c & -d & -d & d & \sin(\beta)c & -\sin(\beta)c & d \\ -\sqrt{a^2+b^2} & \sqrt{a^2+b^2} & 0 & 0 & 0 & \sqrt{a^2+b^2} & -\sqrt{a^2+b^2} & 0 \end{bmatrix}. \tag{5.15}$$

Percebe-se que a matriz de controle não é quadrada, visto que ela apresenta mais entradas de controle (8) do que graus de liberdade (6), ou seja, há mais de uma forma de se obter determinados(as) torques/forças nos GLs. Uma solução possível para fazer a alocação de empuxo é utilizar a matriz pseudo-inversa de Moore-Penrose (FOSSEN, 1994), que minimiza a energia inserida no sistema. Esta matriz, denominada matriz de alocação de empuxo, é dada por

$$B^\dagger = B^T(B\ B^T)^{-1}. \tag{5.16}$$

Assim, o mapeamento dos(as) torques/forças dos GLs nas forças axiais dos propulsores é dado por

$$T = B^\dagger\ \tau^{GL}. \tag{5.17}$$

Uma possível arquitetura do tipo NCS para o ROV é mostrada na Figura 5.6. Nota-se que o sistema embarcado é dividido em três módulos interligados através de rede

de comunicação: o controle central, a fusão sensorial e o sistema de propulsão. Cada módulo possui um núcleo de processamento. O controle central é responsável pela guiagem e pela estratégia de controle. O sistema de propulsão é responsável pelo controle dos propulsores. A fusão sensorial é responsável por adquirir e processar os sinais provenientes dos sensores, disponibilizando para o sistema a estimação das variáveis de estado do veículo (posição/orientação, velocidades e acelerações).

Arquiteturas deste tipo já são empregadas em ROVs comerciais, como o chamado *Distributed Intelligence Control System*, implementado no ROV Seaeye Falcon, da Saab Technologies (SAAB, 2009). Além disso, este tipo de arquitetura proporciona a distribuição do processamento, a redução de custo e de peso, além de facilitar o diagnóstico e manutenção do sistema e de aumentar a sua flexibilidade e agilidade (WALSH et al., 2002). Razões estas para motivar o desenvolvimento deste tipo de arquitetura para o VSOR.

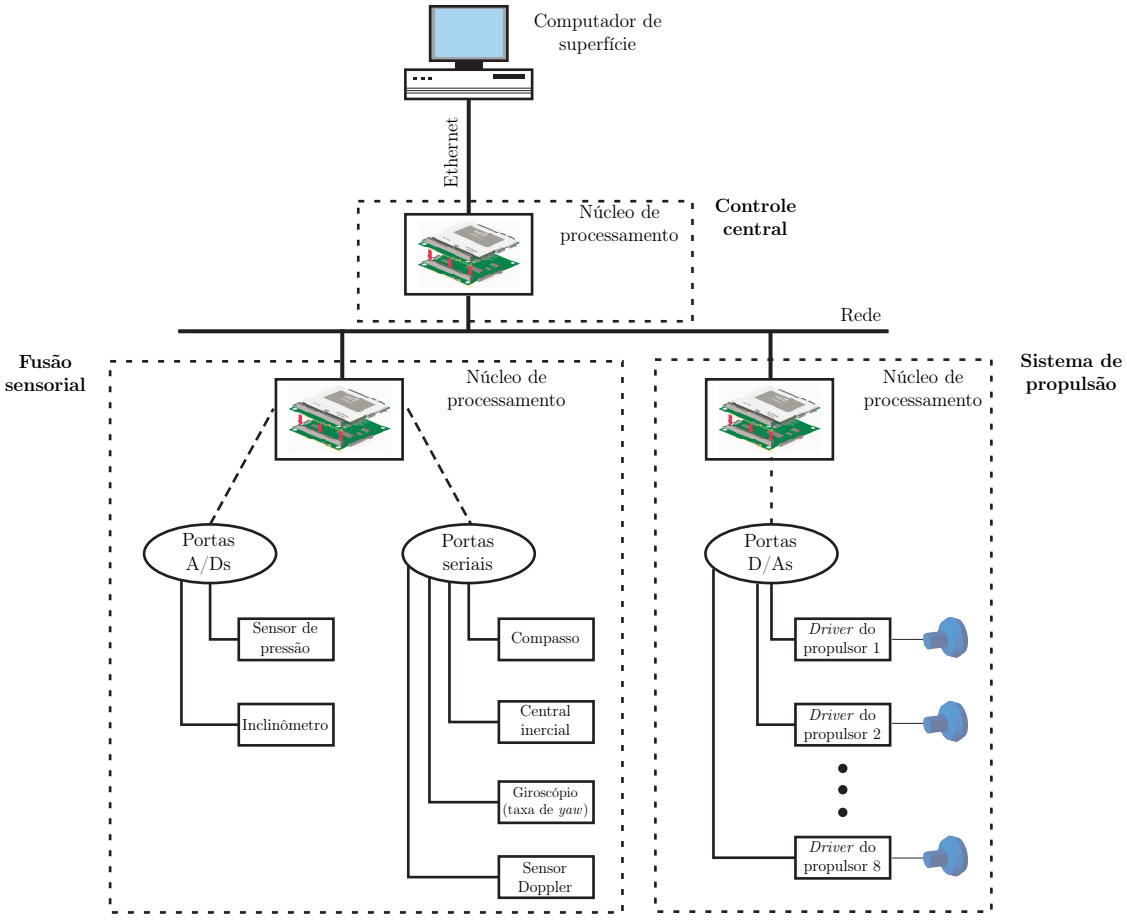


Figura 5.6: Diagrama esquemático da possível arquitetura de controle do tipo NCS para o VSOR.

5.3 Estratégia de Controle

O controle de velocidade (ou estratégia de controle) corresponde à malha de controle na Figura 5.4 que recebe com entrada $\nu_{ref} - \nu$ e fornece como saída τ_{ref}. Neste trabalho, o controle de velocidade do GL i é realizado através de um controlador PI com estrutura *feedback-feedfoward* (CACCIA; VERUGGIO, 2000; SOUZA, 2003). Devido a não-linearidade da equação que representa a dinâmica do GL i, os ganhos do controlador variam de acordo com o ponto de operação.

Reorganizando os termos da Equação (5.6), tem-se a equação diferencial

$$\dot{x}_i = -\frac{d_{L_i}}{m_i}x_i - \frac{d_{Q_i}}{m_i}x_i|x_i| - \frac{b_i}{m_i} + \frac{1}{m_i}\tau_i, \tag{5.18}$$

cuja linearização em torno do ponto de operação x_i^* fornece

$$\dot{x}_{i\delta} = a(x_i^*)x_{i\delta} + \frac{1}{m_i}\tau_{i\delta}, \tag{5.19}$$

onde

$$a(x_i^*) = -\frac{d_{L_i}}{m_i} - 2\frac{d_{Q_i}}{m_i}|x_i^*|, \tag{5.20}$$

$$\tau_i = \tau_{i\delta} + \overline{\tau}_i, \tag{5.21}$$

$$\overline{\tau}_i = d_{L_i}x_i^* + d_{Q_i}x_i^*|x_i^*| + b_i. \tag{5.22}$$

Como pode ser observado, o sinal de controle τ_i é composto por dois termos: o termo *feedback* $\tau_{i\delta}$ e o termo *feedfoward* $\overline{\tau}_i$.

Como citado anteriormente, os GLs *Roll* e *Pitch* são em geral estáveis. Portanto, para a linearização da Equação (5.7) considera-se que os ângulos ϕ e θ são próximos de zero. Assim, o modelo linearizado para estes 2 GLs é dado também pela Equação (5.19), porém excluindo-se o termo referente ao arrasto linear.

Desta forma, para cada ponto de operação desejado há uma função de transferência de primeira ordem para cada GL, do tipo

$$G_i(s) = \frac{\frac{1}{m_i}}{s - a(x_i^*)}. \tag{5.23}$$

Seja o controlador PI com estrutura

$$PI_i(s) = K_{P_i} + \frac{K_{I_i}}{s}, \tag{5.24}$$

onde K_{P_i} refere-se ao ganho proporcional e K_{I_i} refere-se ao ganho integral para o GL i. Desta forma, a equação característica do sistema controlado em malha fechada é dada por

$$s^2 + \left(\frac{1}{m_i}K_{P_i} - a(x_i^*)\right)s + \frac{1}{m_i}K_{I_i}. \tag{5.25}$$

Seja a equação característica desejada para o sistema linearizado em malha fechada dada por

$$s^2 + 2\zeta_i\omega_{n_i} + \omega_{n_i}^2,$$

onde ζ_i é o coeficiente de amortecimento para o GL i e ω_{n_i} a freqüência natural para o GL i, ambos parâmetros de projeto. Por comparação, os ganhos do controlador são dados por

$$K_{P_i} = \frac{2\zeta_i\omega_{n_i} + a(x_i^*)}{\frac{1}{m_i}}, \tag{5.26}$$

$$K_{I_i} = \frac{\omega_{n_i}^2}{\frac{1}{m_i}}. \tag{5.27}$$

Na forma discreta, este controlador PI pode ser representado por

$$\begin{aligned} P_k &= K_{P_i}e_k \\ I_{k+1} &= I_k + b_{i1}e_{k+1} + b_{i2}e_k \\ b_{i1} &= K_{I_i}\tfrac{h}{2} \\ b_{i2} &= K_{I_i}\tfrac{h}{2} \\ u_k &= P_k + I_k. \end{aligned} \tag{5.28}$$

Percebe-se que o termo integral é composto por uma integral numérica do sinal de erro, considerando o passo constante (h constante). Ou seja, em uma situação real em que este passo varia, o valor do sinal de controle pode não ser o valor esperado para um determinado instante.

5.4 Conclusão

Neste capítulo foi descrito o ROV utilizado neste trabalho. Foi mostrada a modelagem da dinâmica do veículo através de 6 equações diferenciais não-acopladas, considerando operações de baixas velocidades. O modelo cinemático relaciona as velocidades e posições/orientações nos sistemas de coordenada inercial e móvel. Foram descritas a arquitetura de

controle atual e uma possível arquitetura do tipo NCS, além da matriz de alocação de empuxo utilizada. E por fim mostrou-se a estratégia de controle utilizando um controlador do tipo PI com estrutura *feedback-feedfoward*, cujos parâmetros de projeto são dados por ζ_i e ω_{n_i}.

6 Resultados

O objetivo deste capítulo é utilizar o TrueTime e o Jitterbug no projeto dos controladores apresentados no Capítulo 5; e, implementar estes controladores no modelo do ROV descrito no Capítulo 5 para observar o seu comportamento no tempo.

Como visto no capítulo anterior, as constantes dos controladores (K_{P_i} e K_{I_i}) foram obtidas em função de dois parâmetros de projeto: ζ_i e ω_{n_i}. Estes parâmetros podem ser obtidos a partir de requisitos de projeto, como tempo de subida, máximo sobre-sinal, tempo de assentamento, etc. O que se mostrará ao longo do capítulo é que a escolha dos valores destes parâmetros também dependem das condições do sistema computacional utilizado. Por exemplo, supondo que para um dado sistema seja desejável uma resposta rápida (menor constante de tempo), o que no caso corresponderia, para um dado valor de ζ, um valor de ω_n maior. Porém, como visto no Capítulo 4, quanto menor a constante de tempo do sistema, menor deve ser o período de amostragem nominal h para o ciclo de controle. E também foi observado neste capítulo que, quanto menor o período de amostragem, mais recursos computacionais são requeridos. Portanto, para uma situação onde os recursos computacionais são consideravelmente limitados, a escolha de parâmetros poderia ser inadequada para o caso de respostas rápidas.

Primeiramente são realizadas simulações com o TrueTime (Seção 6.1), sob algumas hipóteses simplificadoras, para estimação das funções densidades probabilidade (FDPs) dos atrasos inseridos pelo sistema computacional. Estas FDPs são utilizadas para criar modelos de Sistemas Lineares de Saltos Markovianos. No Jitterbug, estes modelos são utilizados no cálculo da função de custo, que por sua vez será utilizada para avaliar qualitativamente o desempenho e a estabilidade (no sentido quadrático médio) para um dado GL (Seção 6.2). Na Seção 6.3 é feita a análise temporal do comportamento do ROV utilizando um modelo em realidade virtual do veículo.

6.1 Análise com o TrueTime

O modelo TrueTime do ROV é semelhante ao utilizado no Capítulo 4, porém existem agora 6 plantas a serem controladas, ou seja, existem 6 *tasks* de controle no controle central.

Para as simulações são adotadas algumas hipóteses de forma a simplificar as análises:

- não há perda de dados,
- as amostragens ocorrem sempre nos instantes de amostragem kh;
- serão considerados apenas os atrasos referentes ao transporte dos dados pela rede e ao escalonamento das *tasks* no controle central;
- as plantas são consideradas idênticas na forma de uma função de transferência de primeira ordem.

Uma forma de escolha de período de amostragem nominal do ciclo de controle é através da regra prática apresentada em Åström e Wittenmark (1997), na qual o período de amostragem deve estar contido no intervalo

$$0.2 \leq \omega_n h \leq 0.6. \tag{6.1}$$

Para estas simulações, o período de amostragem é parametrizado por

$$h \approx \frac{0.4}{\omega_n}. \tag{6.2}$$

Percebe-se nesta parametrização que o período de amostragem aumenta com o inverso da freqüência natural do sistema, o que reflete a relação entre o período de amostragem e o tempo de resposta do sistema citada anteriormente.

Serão analisados 4 casos de operação:

1. sistema somente com as *tasks* de controle;
2. sistema com as *tasks* de controle e com um nó extra na rede para que 80% da banda da rede seja utilizada de forma aleatória, representando uma situação de maior tráfego na rede;

3. sistema com as *tasks* de controle e com uma *task* periódica extra no controle central com WCET uniformemente variável no intervalo $[0, 500]ms$, com período $1000ms$ e prioridade 1, representando outras possíveis *tasks* sendo executadas no sistema;

4. sistema com as *tasks* de controle, o nó extra na rede e a *task* extra especificados anteriormente.

O primeiro caso servirá como referência para as comparações com outros casos, visto que este é o caso mais próximo do funcionamento ideal. O objetivo do caso 2 é analisar a influência dos atrasos devido à concorrência pelo recurso da rede de comunicação. Analogamente, o caso 3 tem por objetivo analisar a influência dos atrasos devido à concorrência das *tasks* no controle central. E por fim, o caso 4 refere-se à análise da influência de todos os atrasos citados anteriormente.

Cada caso pode ser subdividido em três conjuntos de simulações, onde para cada uma varia-se o parâmetro $\omega_{n_i} = \{1, 2, 3\}rad/s$. Para estes parâmetros, escolhe-se $h = \{400, 200, 130\}ms$, respectivamente. Para todos os casos as *tasks* de controle possuem o mesmo WCET ($10ms$) e a mesma prioridade (2). Os nós do controlador, do sensor e do atuador também são interligados por uma rede CAN. As mensagens possuem a mesma prioridade e o mesmo número de bits (80 bits), por simplificação.

As grandezas avaliadas são as mesmas descritas no Capítulo 4:

- L_S^k - latência do sensor;
- L_C^k - latência do controlador;
- h_k - intervalo de amostragem.

6.1.1 Caso 1

Este pode ser considerado o caso ideal, onde os atrasos são constantes. O pior caso da latência do sensor é

$$L_S^k = \frac{80bits}{80\frac{bits}{ms}} + 5 \times 10 = 51ms,$$

referente ao tempo de transporte da mensagem pela rede somado ao tempo em que a *task* demora para iniciar o cálculo do sinal de controle (ou seja, o tempo em que a *task* espera para que as outras cinco *tasks* sejam executadas). Desta análise prévia, conclui-se

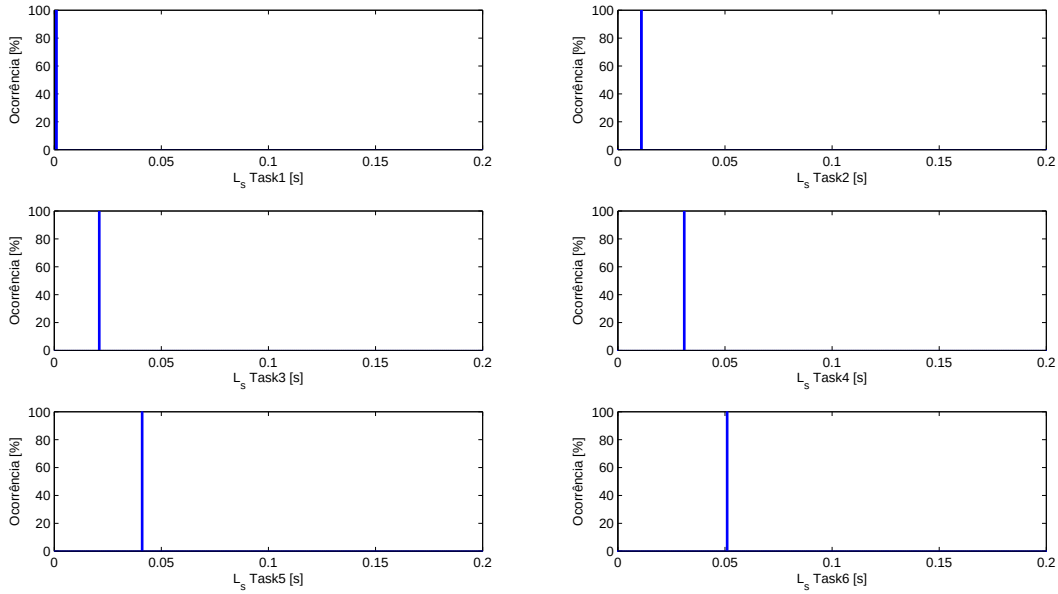


Figura 6.1: Histogramas das latências do sensor das *tasks* de controle para o caso com $h = 400ms$.

que o atraso devido à rede, neste caso, é pequeno (2%) em relação ao atraso gerado pela preempção das *tasks*. Os histogramas das latências do sensor para cada *task* de controle obtidas através das simulações com $h = 400ms$ são apresentados na Figura 6.1. Percebe-se que os histogramas refletem o resultado analítico do pior caso de latência do sensor.

A latência do controlador é idêntica para todos os casos, sendo dada por

$$L_C^k = 10 + \frac{80bits}{80\frac{bits}{ms}} = 11ms,$$

referente ao tempo de execução da *task* somado ao tempo de transporte da mensagem pela rede. Novamente, o atraso referente à rede, neste caso, é pequeno (10%) em relação ao atraso devido ao tempo de execução da *task*. Os histogramas das latências do controlador para cada *task* de controle obtidas através das simulações com $h = 400ms$ são apresentados na Figura 6.2. Percebe-se que os histogramas refletem o resultado analítico do pior caso de latência do controlador.

Para o pior caso (a *task* que precisa esperar as outras cinco serem executadas), o tempo de execução do ciclo completo ($51 + 11 = 62ms$) não ultrapassa nenhum dos valores dos períodos de amostragem nominal adotados. Assim, espera-se que os intervalos de amostragem sejam constantes e igual ao período de amostragem nominal ($h = h_k$).

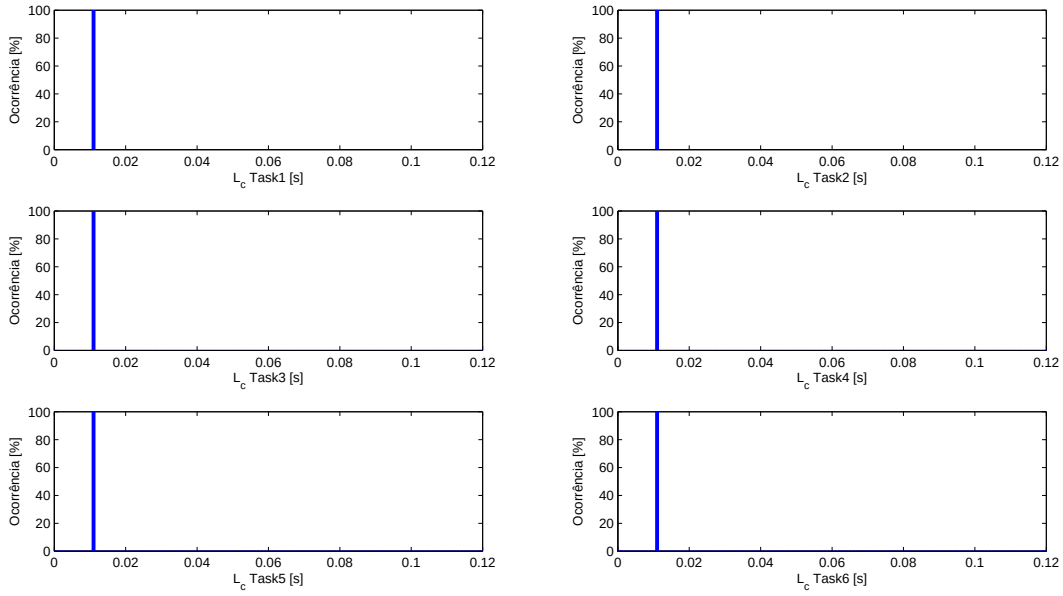


Figura 6.2: Histogramas das latências do controlador das *tasks* de controle para $h = 400ms$, para o Caso 1.

Este resultado analítico corrobora com os histogramas dos intervalos de amostragem para cada *task* de controle obtidos através das simulações com $h = 400ms$, apresentados na Figura 6.3.

As distribuições das latências para as simulações com $h = \{200, 130\}ms$ são idênticas às encontradas para o caso em que $h = 400ms$. A diferença está na carga[1] na CPU, como pode ser observado nos gráficos da Figura 6.4. Nota-se que quanto menor o período de amostragem, menos tempo o processador ficará ocioso.

[1]Porcentagem referente ao tempo médio no qual o processador está sendo utilizado.

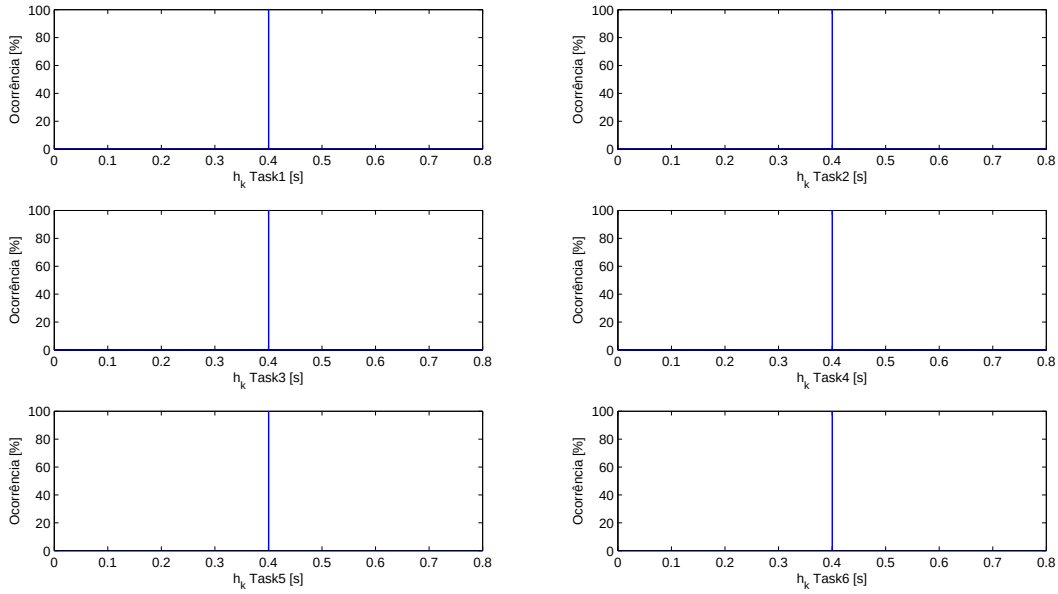
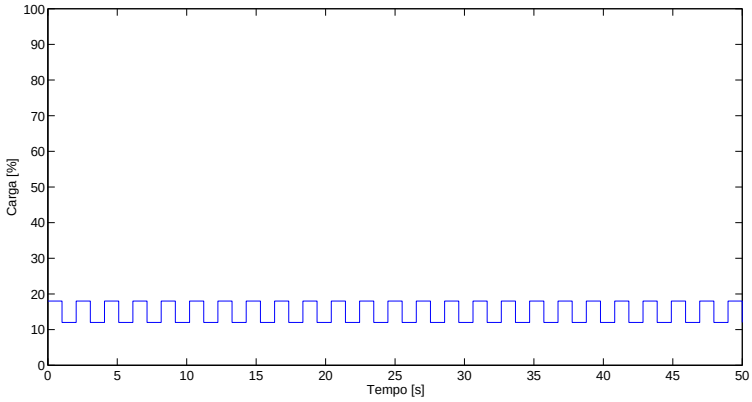
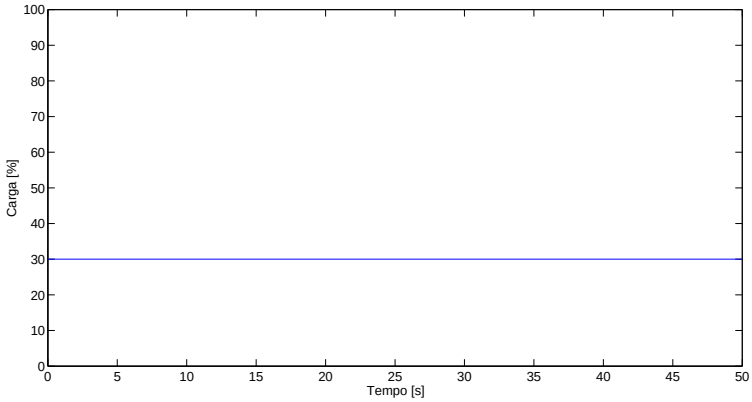


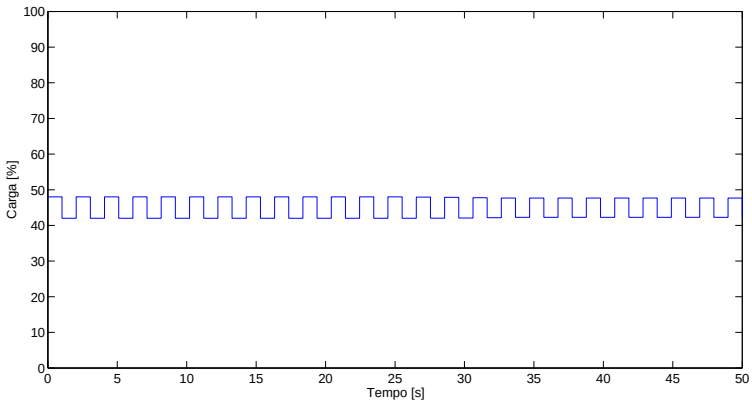
Figura 6.3: Histogramas dos intervalos de amostragem das *tasks* de controle para $h = 400ms$, para o Caso 1.



(a) $h = 400ms$



(b) $h = 200ms$



(c) $h = 130ms$

Figura 6.4: Carga na CPU em função do tempo para $h = \{400, 200, 130\}ms$, para o Caso 1.

6.1.2 Caso 2

O intuito agora é verificar a influência do aumento de carga[2] na rede. Para isso acrescenta-se um nó extra na rede de forma que 80% de sua banda seja utilizada aleatoriamente. Como as latências agora possuem um termo de caráter aleatório (atraso no envio e recebimento de mensagens pela rede), descrições formais não são feitas. E por isso não é possível predizer o comportamento dos intervalos de amostragem sem recorrer aos resultados das simulações.

Para este caso, as latências apresentadas correspondem à simulação que adota $h = 130ms$. Este será considerado o pior caso visto que, pressupõe-se que os atrasos serão maiores quanto mais escassos forem os recursos computacionais. Entretanto, são tomadas também as estatísticas das latências para $h = \{400, 200\}ms$ na análise com o Jitterbug na próxima seção.

O resultado da simulação para a latência do sensor é mostrado na Figura 6.5. É possível observar que neste caso, ainda existem os atrasos constantes devido ao escalonamento das *tasks* de controle. Percebe-se que a rede de comunicação deu à estas latências um caráter aleatório, cujas ocorrências se concentram próximo aos valores obtidos para o Caso 1. E ainda, quanto maior o atraso referente ao escalonamento, mais distribuídas são as latências do sensor.

Resultados semelhantes podem ser observados na Figura 6.6, onde as distribuições das latências L_C^k também apresentam um caráter aleatório e concentram-se próximo aos valores obtidos para o Caso 1. E ainda, quanto maior o atraso referente ao escalonamento, mais distribuídas são as latências do controlador.

Estas variações das latências resultam na variação dos intervalos de amostragem, visto na Figura 6.7. Percebe-se que estes intervalos de amostragem são normalmente distribuídos em torno do período de amostragem nominal $h = 130ms$, e quanto maior o atraso referente ao escalonamento, mais disperso é a distribuição, como verificado na Tabela 6.1.

Novamente, para cada h existe uma diferente carga na CPU, como pode ser observado nos gráficos da Figura 6.8. A variação das latências causa uma clara variação na carga somente para o pior caso, com $h = 130ms$, cujos recursos computacionais são mais escassos.

[2]Porcentagem referente ao tempo médio no qual a rede está sendo utilizada

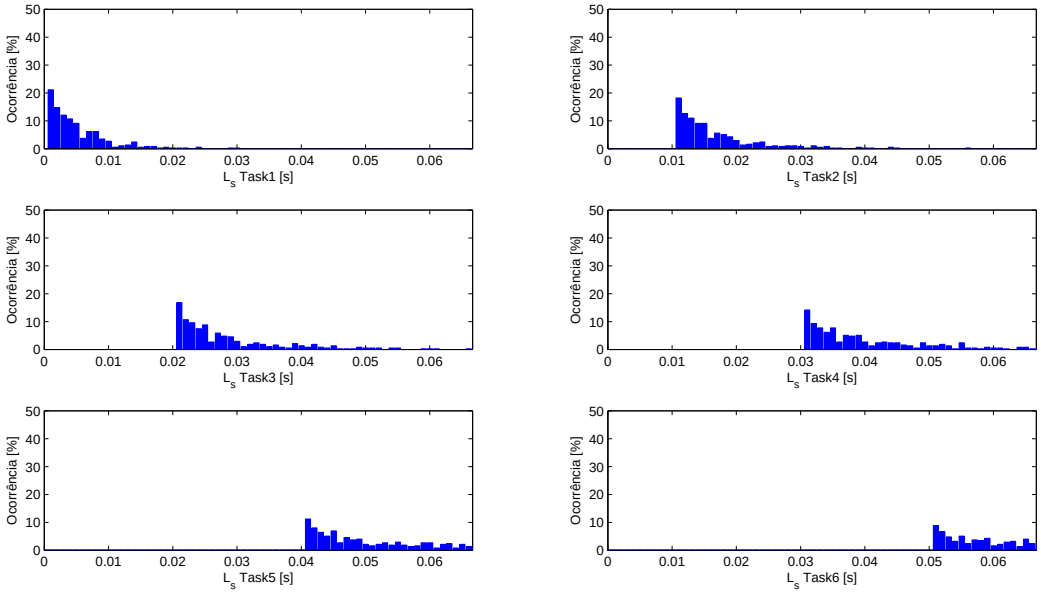


Figura 6.5: Histogramas das latências do sensor das *tasks* de controle para $h = 130ms$, para o Caso 2.

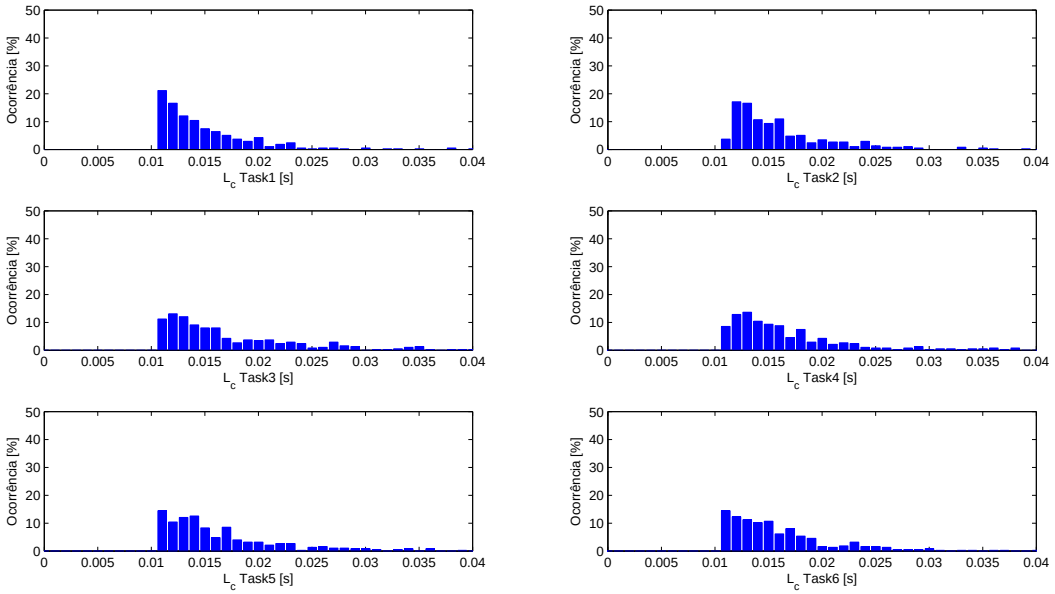


Figura 6.6: Histogramas das latências do controlador das *tasks* de controle para $h = 130ms$, para o Caso 2.

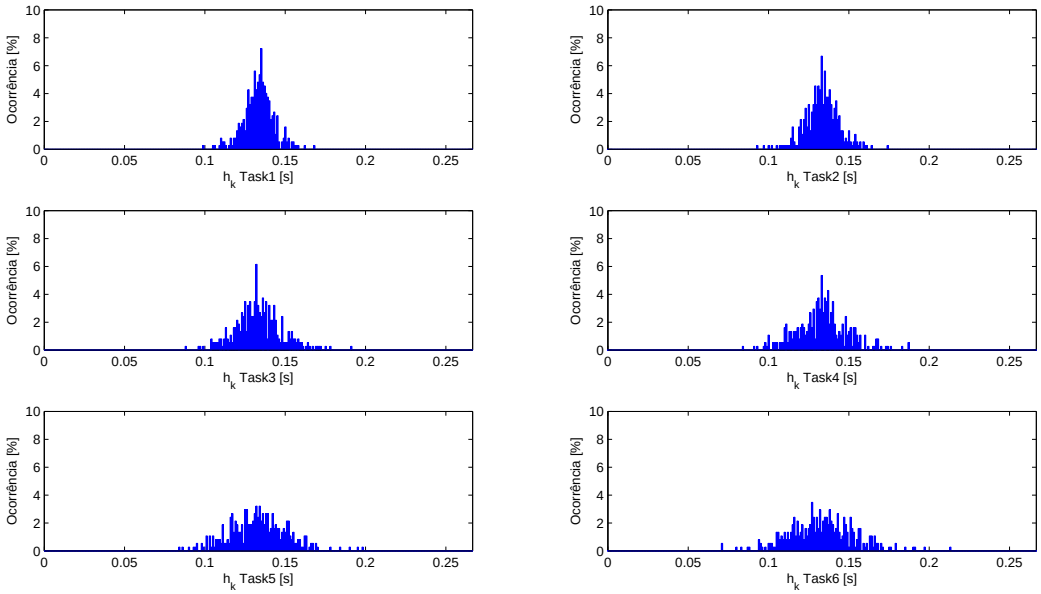
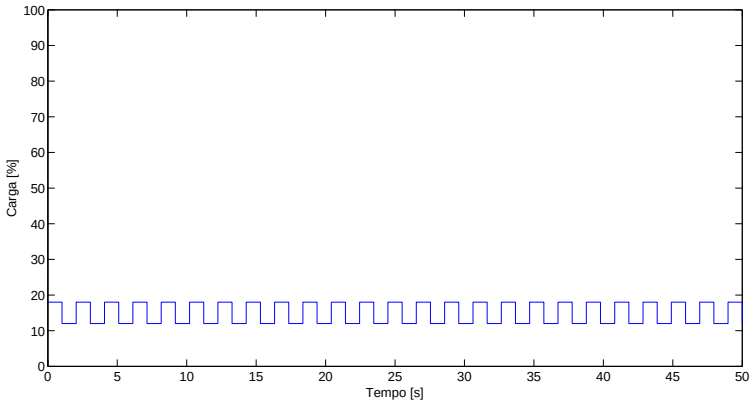


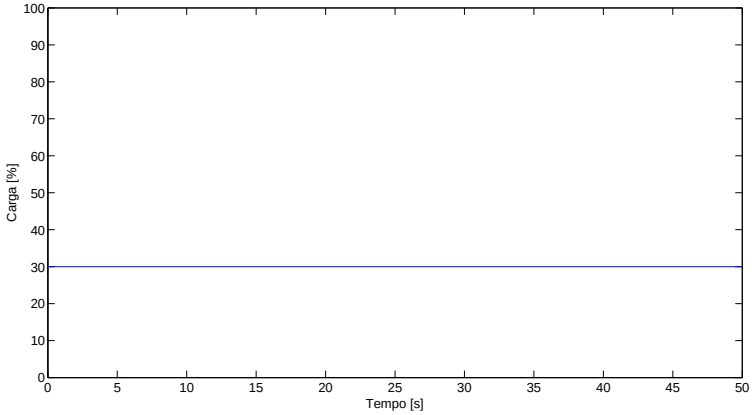
Figura 6.7: Histogramas dos intervalos de amostragem das *tasks* de controle para $h = 130ms$, para o Caso 2.

Task	1	2	3	4	5	6
Média$[ms]$	130	130	130	130	130	130
Desvio padrão$[ms]$	9.4	10.7	14.4	16.1	17.7	20.0

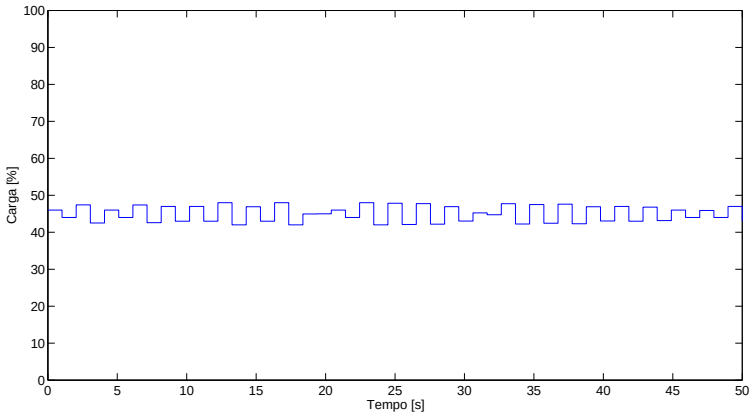
Tabela 6.1: Médias e desvios padrão dos intervalos de amostragem das *tasks* de controle para $h = 130ms$, para o Caso 2.



(a) $h = 400ms$



(b) $h = 200ms$



(c) $h = 130ms$

Figura 6.8: Carga na CPU em função do tempo para $h = \{400, 200, 130\}ms$, para o Caso 2.

6.1.3 Caso 3

O intuito agora é verificar a influência do aumento de carga na CPU. Para isso acrescenta-se uma *task* periódica extra no controle central com WCET uniformemente variável no intervalo $[0, 500]ms$, com período $1000ms$ e prioridade 1. Nota-se que as latências possuem um termo de caráter aleatório (atraso no escalonamento das *tasks*). E novamente, as latências apresentadas correspondem à simulação que adota $h = 130ms$ pela mesma hipótese anterior. Entretanto, são tomadas também as estatísticas das latências para $h = \{400, 200\}ms$ na análise com o Jitterbug na próxima seção.

O escalonamento das *tasks* no controle central está representado no diagrama de escalonamento da Figura 6.9. Nota-se que a *task* extra nunca fica no estado de preempção, devido à sua maior prioridade.

O resultado da simulação para a latência do sensor é mostrado na Figura 6.10. Percebe-se que aproximadamente metade das ocorrências se concentram perto dos valores obtidos no Caso1, porém, outros 50% estão mais dispersos que no Caso 2.

O resultado da simulação para a latência do controlador é mostrado na Figura 6.11. Diferentemente do Caso 2, as distribuições, neste caso, não experimentam muita variação, se concentrando próximo aos valores obtidos no Caso 1.

Pela Figura 6.12 nota-se que os intervalos de amostragem ficam concentrados principalmente em dois valores, sendo um deles equivalente ao período de amostragem nominal.

A carga na CPU para cada caso é dada na Figura 6.13. Percebe-se que a carga computacional é variável. As médias e os desvios padrão são dados na Tabela 6.2. Nota-se que a diminuição dos períodos de amostragem causam um aumento na média da carga, mas não altera o desvio padrão.

h	$h = 400ms$	$h = 200ms$	$h = 130ms$
Médias [%]	43.1	55.6	71.7
Desvios padrão [%]	14.9	14.6	14.9

Tabela 6.2: Médias e desvios padrão da carga na CPU, para $h = \{400, 200, 130\}ms$, para o Caso 3.

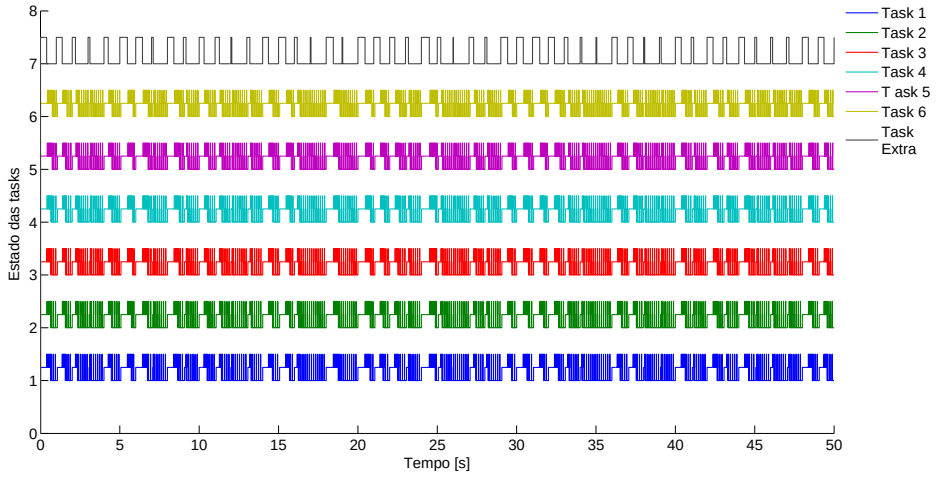


Figura 6.9: Diagrama de escalonamento das *tasks* do controle central para $h = 130ms$, para o Caso 3.

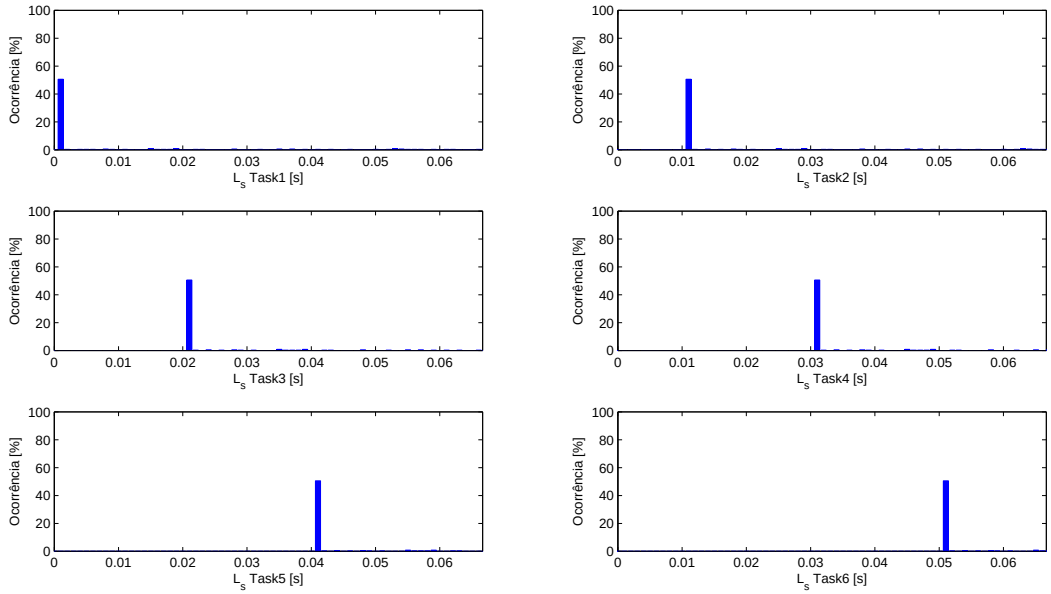


Figura 6.10: Histogramas das latências do sensor das *tasks* de controle para $h = 130ms$, para o Caso 3.

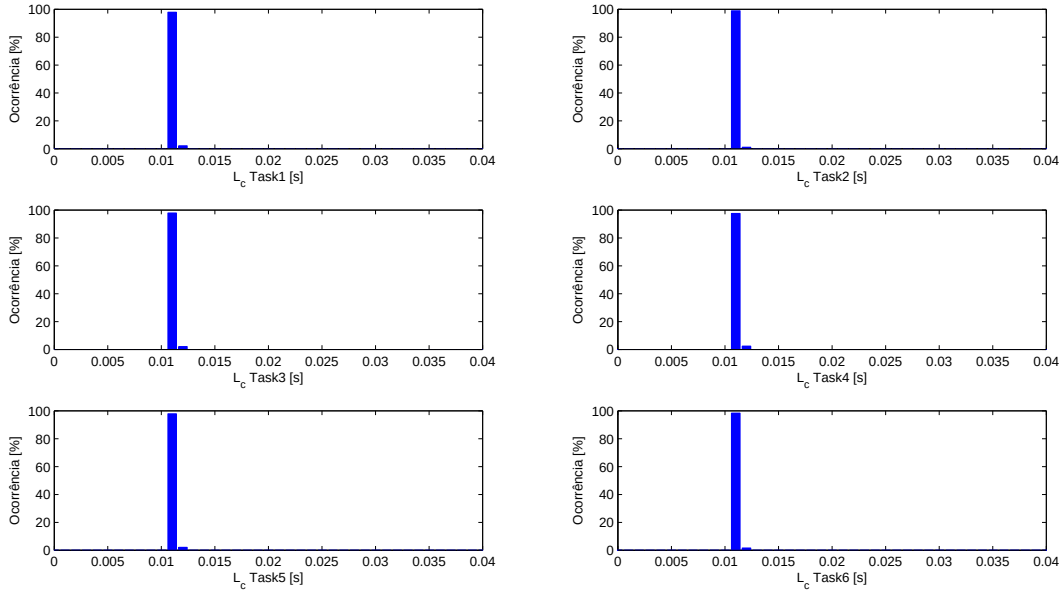


Figura 6.11: Histogramas das latências do controlador das *tasks* de controle para $h = 130ms$, para o Caso 3.

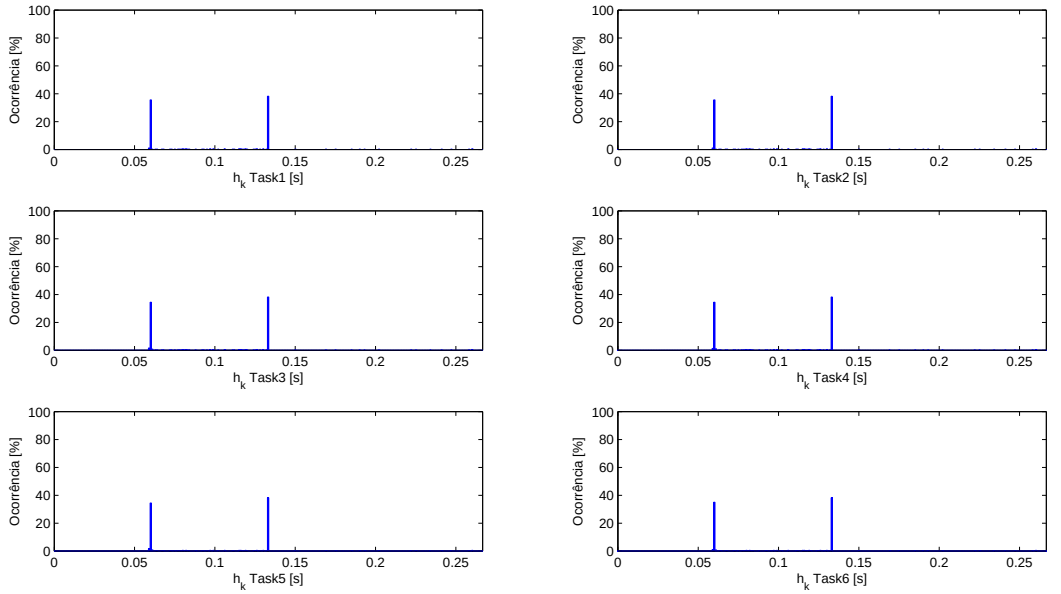
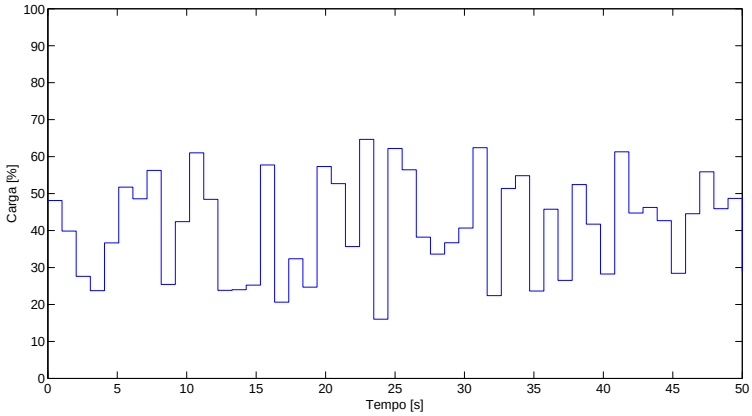
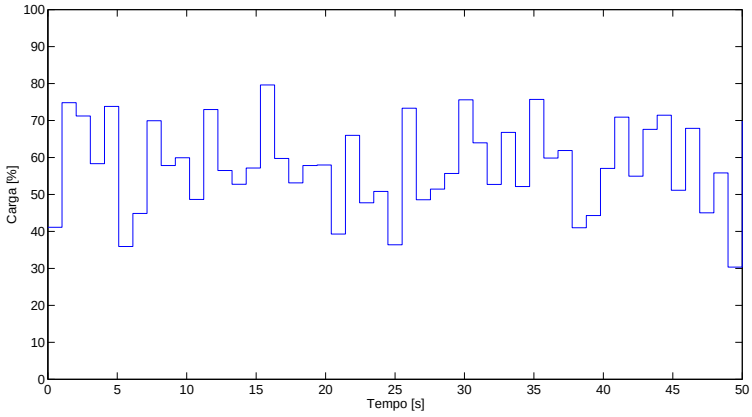


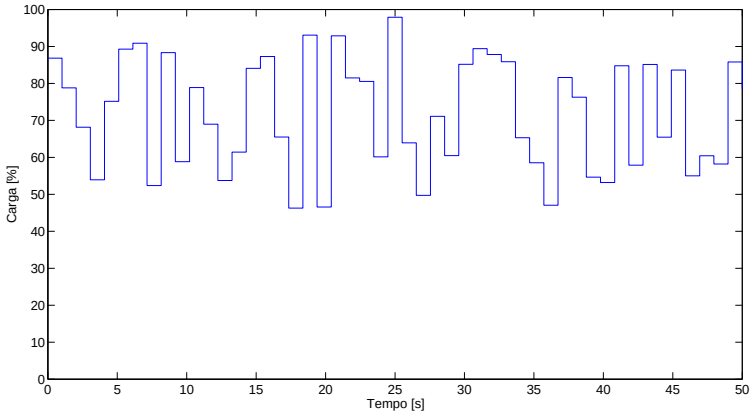
Figura 6.12: Histogramas dos intervalos de amostragem das *tasks* de controle para $h = 130ms$, para o Caso 3.



(a) $h = 400ms$



(b) $h = 200ms$



(c) $h = 130ms$

Figura 6.13: Carga na CPU em função do tempo para $h = \{400, 200, 130\}ms$, para o Caso 3.

6.1.4 Caso 4

Nos Casos 2 e 3, o problema dos atrasos devido ao tráfego na rede de comunicação e ao escalonamento de *tasks* foi tratado separadamente. Neste Caso 4, estas duas fontes de atraso

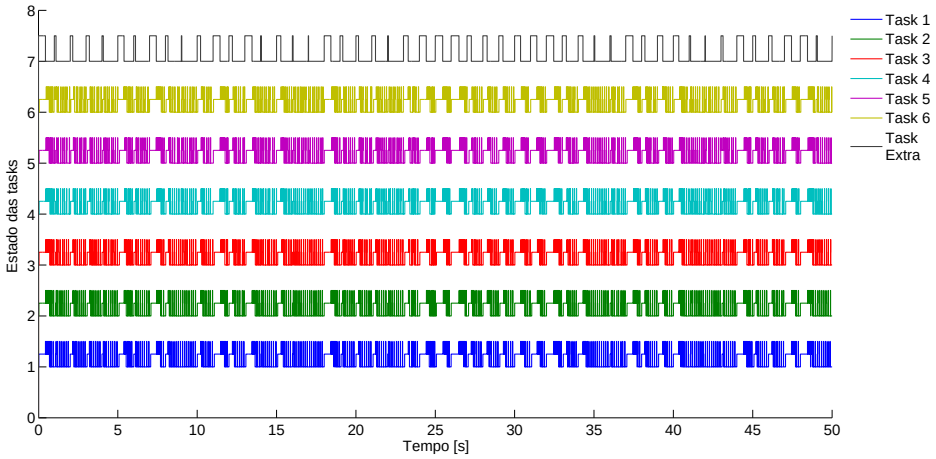


Figura 6.14: Diagrama de escalonamento das *tasks* do controle central para $h = 130ms$, para o Caso 4.

agem em conjunto. Novamente, as latências apresentadas correspondem à simulação que adota $h = 130ms$.

Percebe-se através das Figuras 6.14, 6.15, 6.16 e 6.17 que tanto os efeitos do Caso 2 como os efeitos do Caso 3 estão contidos neste Caso 4. Estes efeitos agindo em conjunto, como visto na Figura 6.18, aumenta a dispersão da distribuição intervalo de amostragem de forma que esta se aproxima mais de uma distribuição uniforme (com intervalo de aproximadamente $[50, 150]ms$) do que uma distribuição normal. O impacto maior desta distribuição uniforme, do ponto de vista de controle, é no termo integral do controlador discreto, que, como visto no Capítulo 5, considera o intervalo de integração (h) constante. Na análise com o Jitterbug será possível analisar esse impacto no que se refere à estabilidade (no sentido quadrático médio) e ao desempenho do sistema de controle.

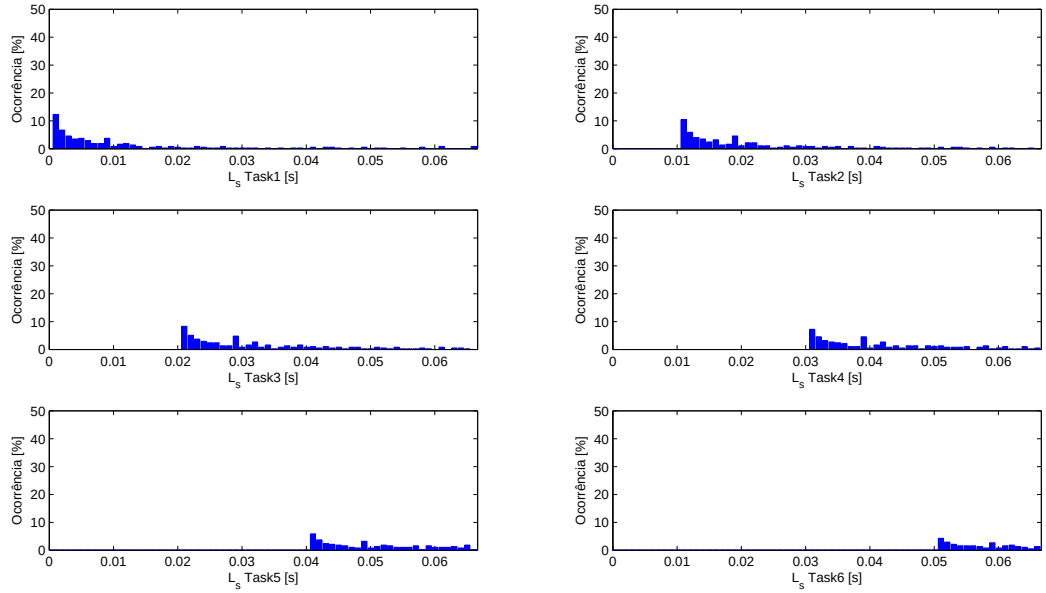


Figura 6.15: Histogramas das latências do sensor das *tasks* de controle para $h = 130ms$, para o Caso 4.

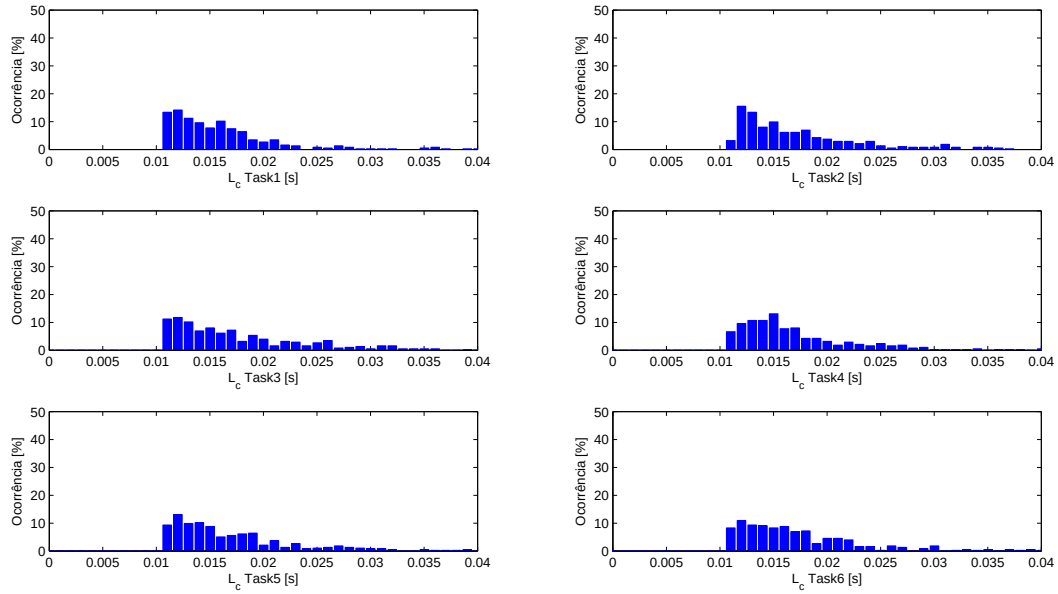
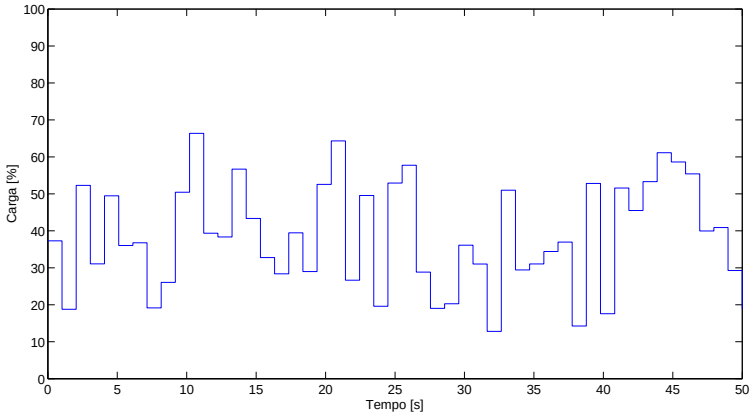
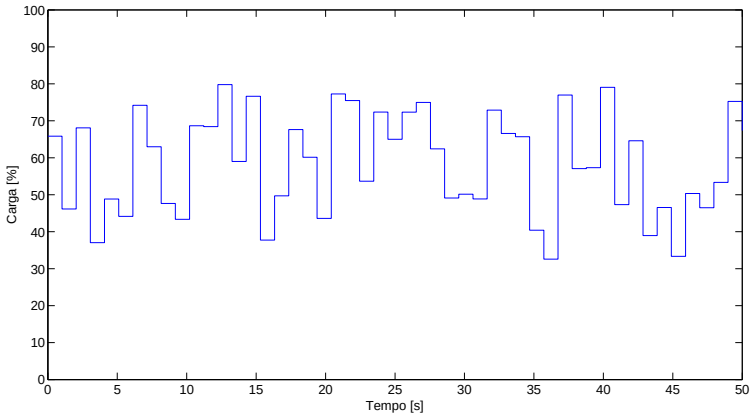


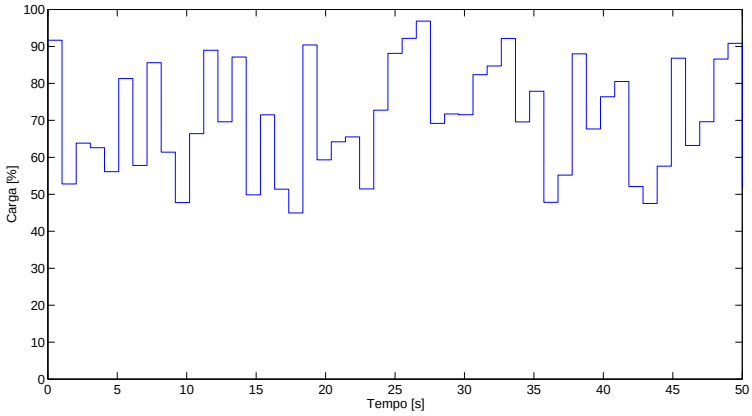
Figura 6.16: Histogramas das latências do controlador das *tasks* de controle para $h = 130ms$, para o Caso 4.



(a) $h = 400ms$



(b) $h = 200ms$



(c) $h = 130ms$

Figura 6.17: Carga na CPU em função do tempo para $h = \{400, 200, 130\}ms$, para o Caso 4.

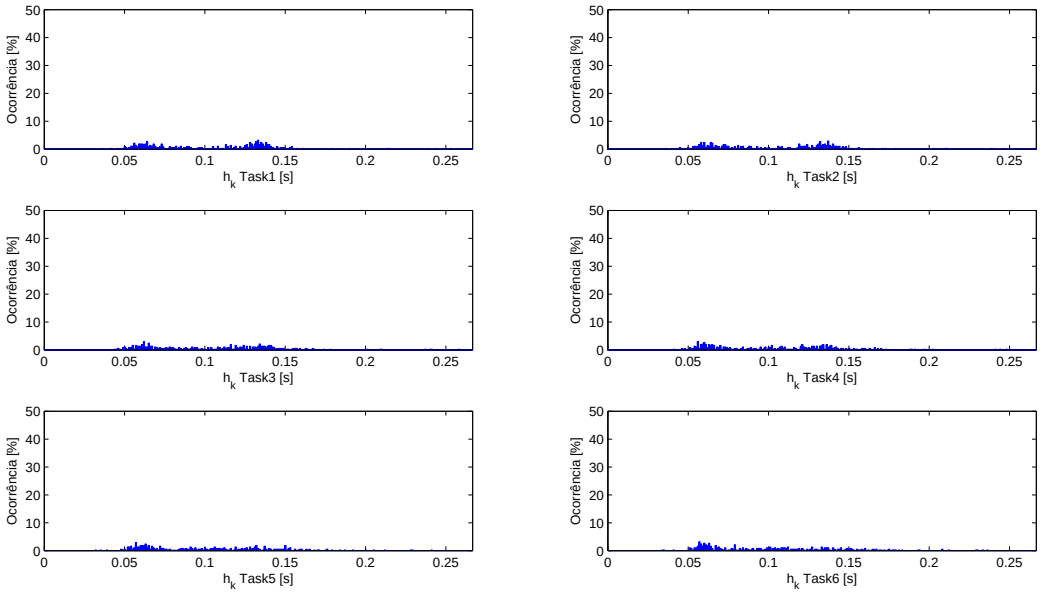


Figura 6.18: Histogramas dos intervalos de amostragem das *tasks* de controle para $h = 130ms$, para o Caso 4.

6.2 Análise com o Jitterbug

O modelo Jitterbug é o mesmo adotado no Capítulo 4, onde $H_1(z) = 1$, $H_2(z) = PI(z)$ e $H_3(z) = 1$ representam o nó do sensor, do controlador e do atuador, respectivamente; e $G(s)$ é o modelo de um GL i (*surge*,*sway*,*heave* e *yaw*) representado por uma função de transferência de primeira ordem, do tipo

$$G(s) = \frac{b}{s+a},$$

onde

$$\begin{aligned} a(x_i^*) &= -\frac{d_{L_i}}{m_i} - 2\frac{d_{Q_i}}{m_i}|x_i^*|, \\ b &= \frac{1}{m_i}. \end{aligned}$$

Portanto, para cada ponto de operação existe uma função de transferência. No caso de *roll* e *pitch* há o controle passivo (ou seja, a resultante do empuxo) que mantém estes dois GLs estáveis (SOUZA, 2003), por isso não foram considerados nesta análise.

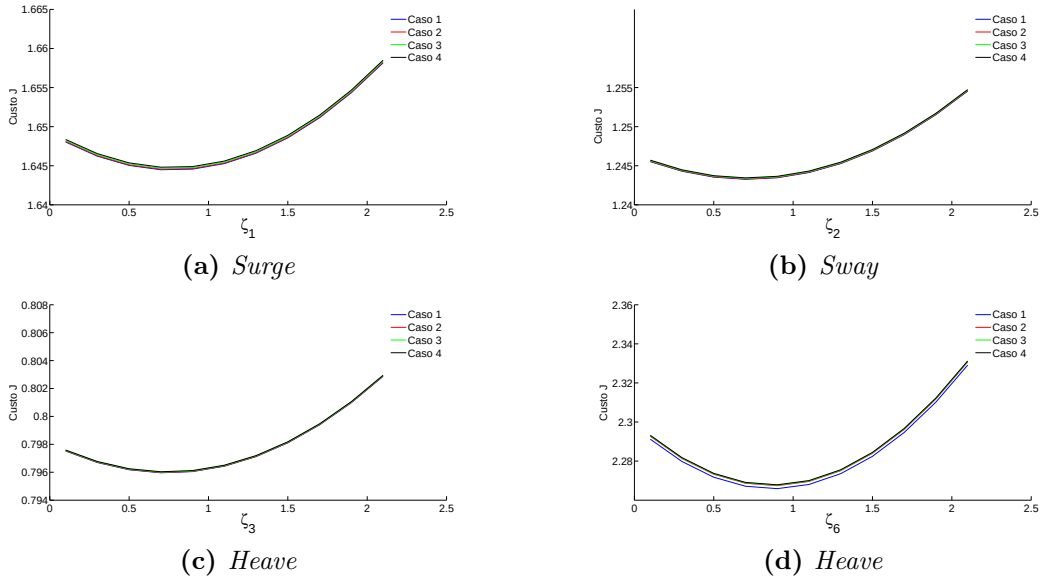
O cálculo da função de custo é feito fixando-se w_{n_i} (o que é equivalente à fixar h pela parametrização adotada); adotando os pontos de operação $x_1^* = 0.5m/s$, $x_2^* = 0.5m/s$, $x_3^* = 0.5m/s$ e $x_6^* = 1.4rad/s$; e variando-se o coeficiente de amortecimento ζ_i. Desta forma, é possível obter o conjunto de parâmetros de projeto ótimos, que proporciona o menor custo J. Faz-se isso para cada caso visto na seção anterior, através das distribuições das latências obtidas.

Sabendo que para um dado coeficiente de amortecimento, o sistema responde mais rapidamente para uma freqüência natural maior, espera-se que quanto mais rápida a resposta, maior será o impacto das latências sobre desempenho e estabilidade (no sentido quadrático médio) do sistema. Isto pode ser verificado nas Figuras 6.19, 6.20 e 6.21, onde as curvas dos custos referentes aos Casos 2, 3 e 4 estão relativamente mais distantes da curva do custo referente ao Caso 1 conforme aumenta-se ω_{n_i}.

Pelos gráficos também é possível notar que existem valores mínimos para as funções de custo, que, segundo o critério de avaliação utilizado, correspondem aos pares (ζ_i,ω_{n_i}) ótimos para cada caso e situação de atraso.

Nota-se também que mesmo para os piores casos de atraso, existem intervalos relativamente grandes para a escolha dos parâmetros (ζ_i,ω_{n_i}) que resultam num custo limitado (J

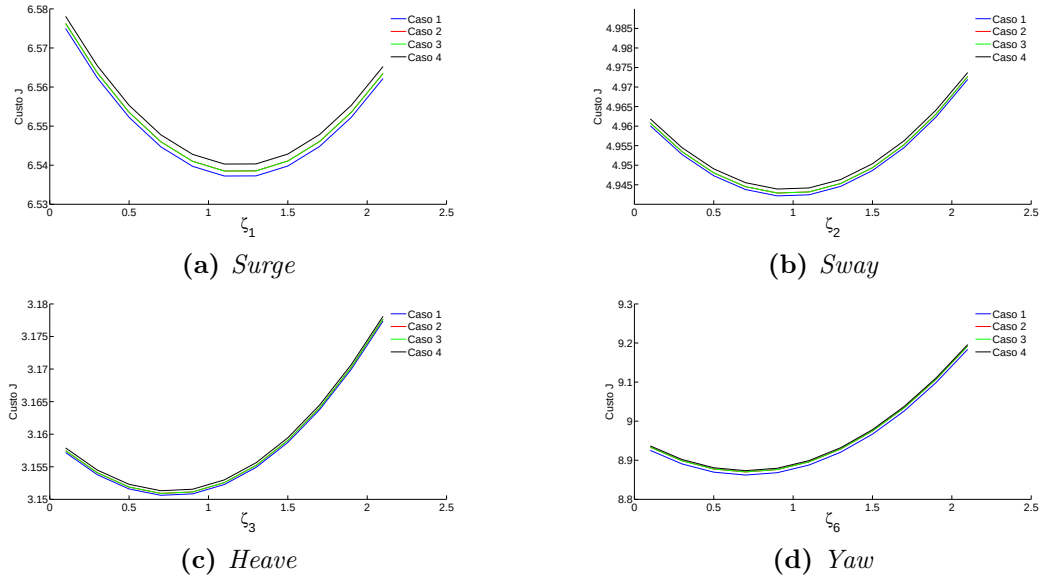
finito), ou seja, o sistema é considerado estável no sentido quadrático médio. Isto porque os sistemas adotados ($G(s)$) são sistemas de primeira ordem com pólos no SPE (semiplano esquerdo) do plano-s, ou seja, sistemas tipicamente estáveis. Se fossem adotados sistemas tipicamente instáveis (como visto no trabalho de (CERVIN et al., 2006)), provavelmente os intervalos para escolha de (ζ_i,ω_{n_i}) seriam relativamente menores.



(a) *Surge* **(b)** *Sway*

(c) *Heave* **(d)** *Heave*

Figura 6.19: Custo J em função de ζ_i, para os Casos $\{1, 2, 3, 4\}$, considerando $\omega_{n_i} = 1rad/s$, o que corresponde a $h = 400ms$.

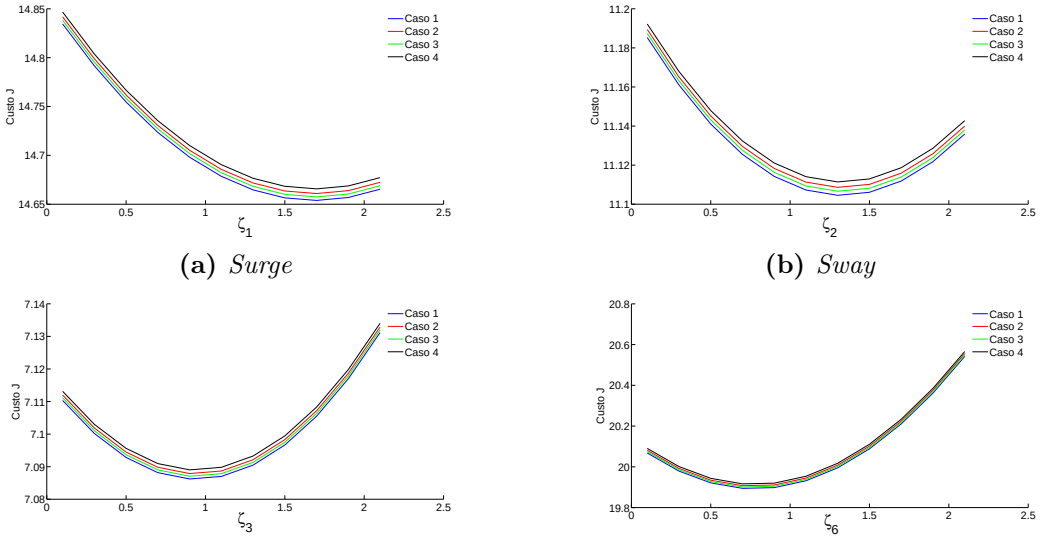


(a) *Surge* **(b)** *Sway*

(c) *Heave* **(d)** *Yaw*

Figura 6.20: Custo J em função de ζ_i, para os Casos $\{1, 2, 3, 4\}$, considerando $\omega_{n_i} = 2rad/s$, o que corresponde a $h = 200ms$.

No entanto, é importante salientar que a função de custo J reflete a relação de energia



(a) *Surge* **(b)** *Sway*

(c) *Heave* **(d)** *Yaw*

Figura 6.21: Custo J em função de ζ_i, para os Casos $\{1, 2, 3, 4\}$, considerando $\omega_{n_i} = 3rad/s$, o que corresponde a $h = 130ms$.

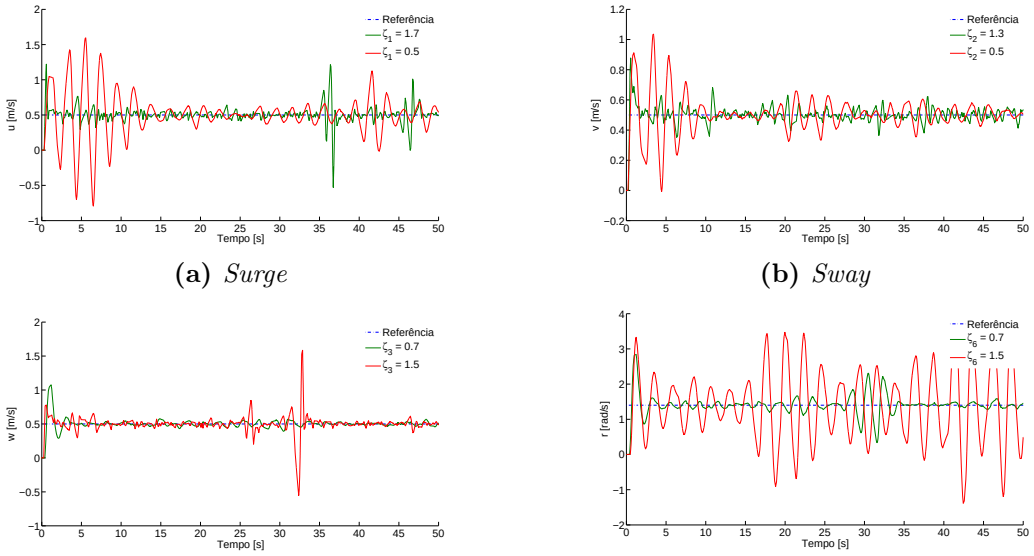
do sistema (Equações (2.19) e (2.21)). Portanto, o desempenho ótimo se refere à melhor relação de energia para determinados parâmetros (ω_{n_i}, ζ_i) e casos de atrasos.

6.3 Simulações com o Modelo do ROV

Através das simulações que envolvem o modelo dinâmico do ROV descrito no Capítulo 5 em conjunto com o TrueTime, verifica-se o comportamento do sistema no tempo, comparando com os resultados da seção anterior. Foram acrescentados nestas simulações ruído branco gaussiano nos sinais adquiridos pelos sensores e sinais de distúrbio na entrada das plantas a fim de aproximar a simulação de situações encontradas na prática.

A Figura 6.22 mostra a resposta do sistema no tempo nos GLs *surge*, *sway*, *heave* e *yaw* considerando $w_{n_1} = w_{n_2} = w_{n_3} = w_{n_6} = 3[rad/s]$ para o Caso 4. Foram considerados dois sistemas: um com parâmetros (ζ_i,ω_{n_i}) ótimos referentes aos menores valores da função de custo J, de acordo com a Figura 6.21; e outro diferente deste. Visualmente é possível notar que, embora ambos os sistemas sejam estáveis, aquele com os coeficientes de amortecimento ditos ótimos para este caso apresenta menores amplitudes de oscilação.

A Figura 6.23 mostra a resposta no tempo do sistema no GL *surge* para $(\omega_{n_1}, \zeta_1) = \{(1, 0.7), (2, 1.2), (3, 1.7)\}$ (referentes aos menores valores de J) para o Caso 4. Nota-se que, conforme ω_{n_1} aumenta, o sistema responde mais rapidamente, porém, as amplitudes



(a) *Surge* **(b)** *Sway*

(c) *Heave* **(d)** *Yaw*

Figura 6.22: Resposta do sistema no tempo nos GLs *surge*, *sway*, *heave* e *yaw* considerando $w_{n_1} = w_{n_2} = w_{n_3} = w_{n_6} = 3[rad/s]$ para o Caso 4.

de oscilação são maiores.

Para visualizar a influência do comportamento temporal de cada GL na realização de tarefas, como por exemplo o deslocamento do veículo sobre uma trajetória pré-definida, criou-se um modelo do VSOR em realidade virtual (Figura 6.24) através do SIMULINK. Com este modelo simulou-se três situações envolvendo o deslocamento do móvel no plano XY (correspondente aos GLs *surge*, *sway* e *yaw*), cujos resultados são observados nas Figuras 6.25, 6.26 e 6.27.

Nestes gráficos são comparados os Casos 1 e 4, para os parâmetros ótimos obtidos na seção anterior. Nota-se que o efeito dos atrasos não comprometeram o desempenho do sistema de controle na realização desta tarefa específica, para os parâmetros aqui avaliados.

O sistema com menor tempo de resposta (maior ω_{n_i} e, conseqüentemente menor h) conseguiu se manter mais próximo da trajetória de referência do que os outros sistemas. Do ponto de vista de controle, era esperado este comportamento. Porém, como visto no Capítulo 4, existe uma relação de compromisso entre o desempenho do sistema de controle e o desempenho do sistema computacional. Portanto, é provável que para valores de ω_{n_i} maiores que $3rad/s$ (ou seja, $h > 130ms$) os efeitos dos atrasos inseridos pelo sistema computacional sejam mais relevantes no desempenho do sistema de controle para realização desta tarefa.

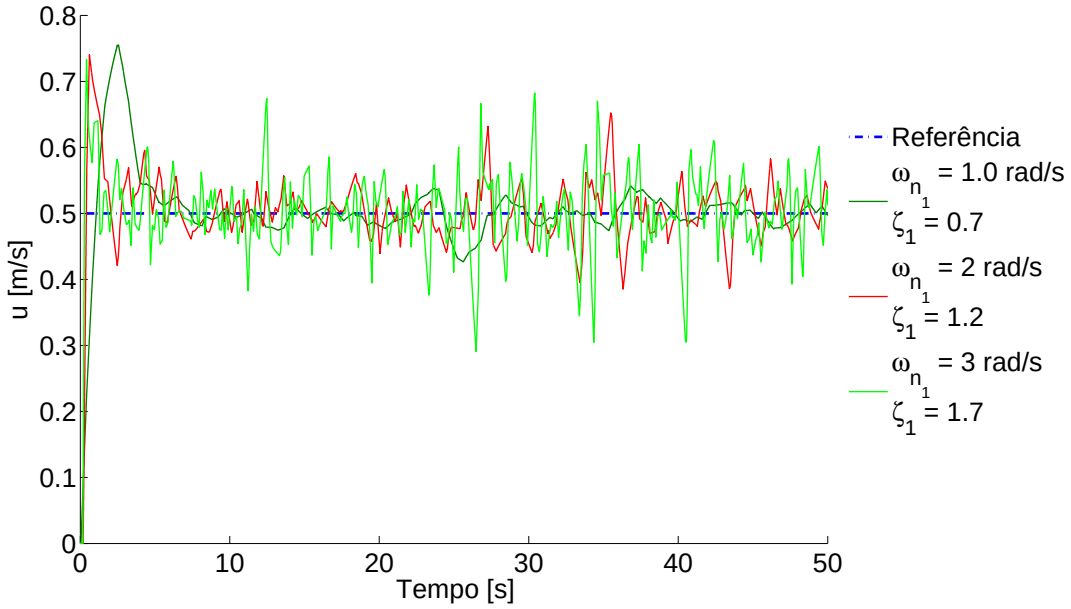


Figura 6.23: Resposta do sistema no GL *surge* para $(\omega_{n_1}, \zeta_1) = \{(1, 0.7), (2, 1.2), (3, 1.7)\}$ para o Caso 4.

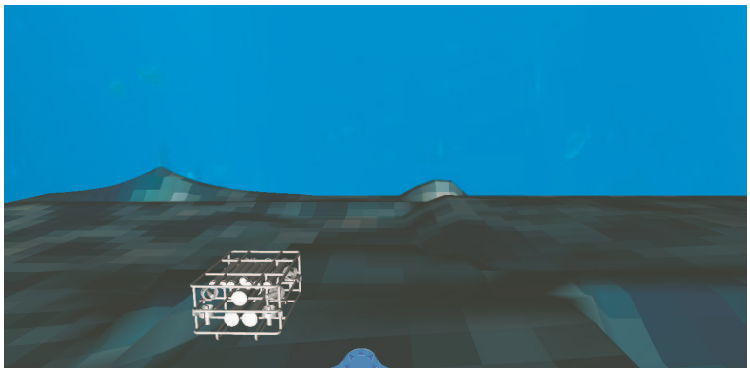


Figura 6.24: Modelo do VSOR em realidade virtual.

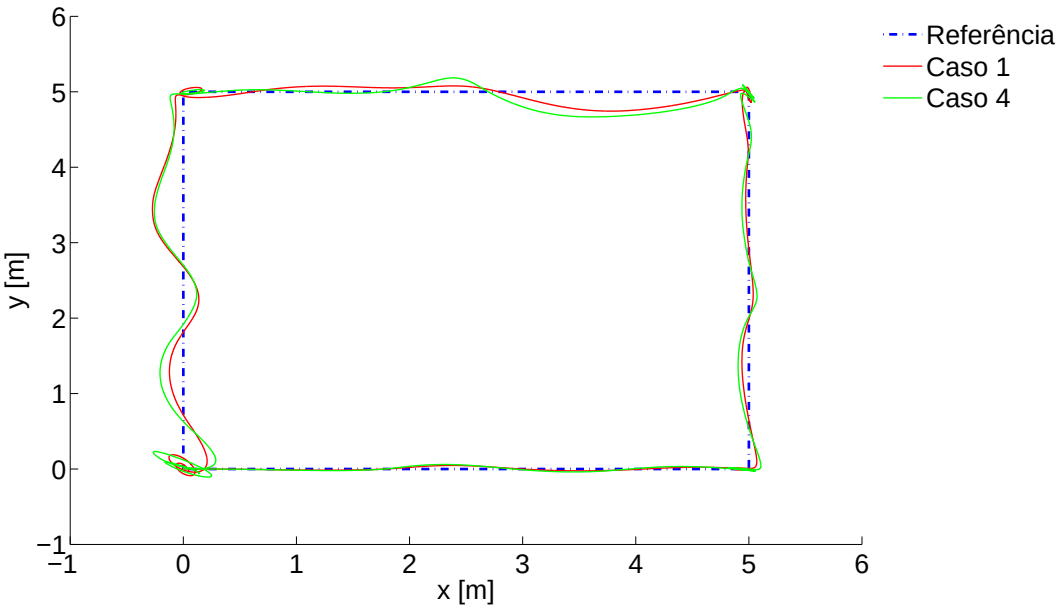


Figura 6.25: Deslocamento no plano XY do veículo para $(\omega_{n_1}, \zeta_1) = (1, 0.7)$, $(\omega_{n_2}, \zeta_2) = (1, 0.8)$, $(\omega_{n_3}, \zeta_3) = (1, 0.8)$ e $(\omega_{n_6}, \zeta_6) = (1, 0.9)$, para o Caso 1 e para o Caso 4.

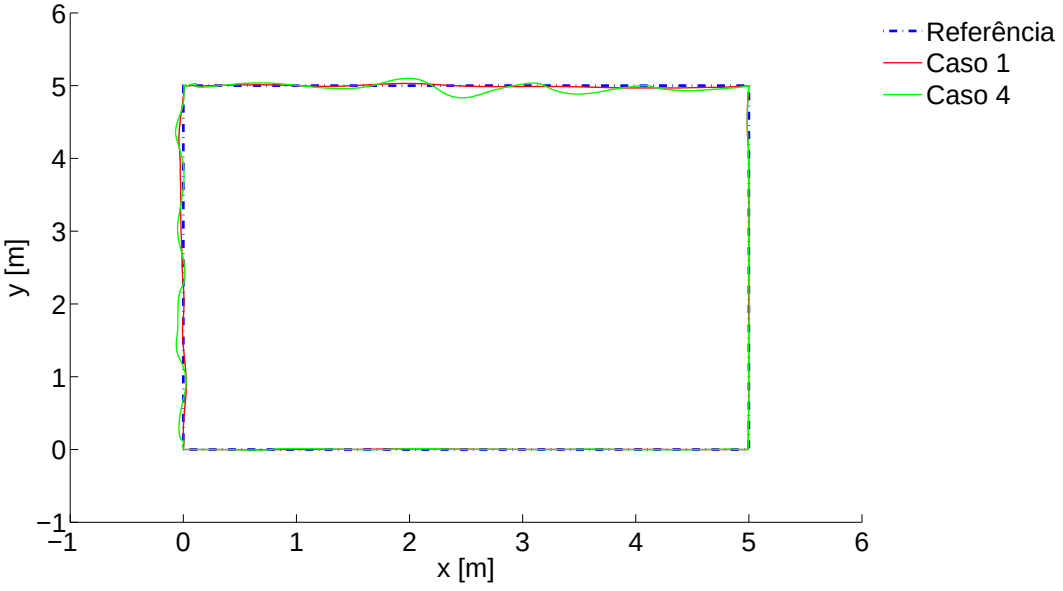


Figura 6.26: Deslocamento no plano XY do veículo para $(\omega_{n_1}, \zeta_1) = (2, 1.2)$, $(\omega_{n_2}, \zeta_2) = (2, 1.0)$, $(\omega_{n_3}, \zeta_3) = (2, 0.8)$ e $(\omega_{n_6}, \zeta_6) = (2, 0.8)$, para o Caso 1 e para o Caso 4.

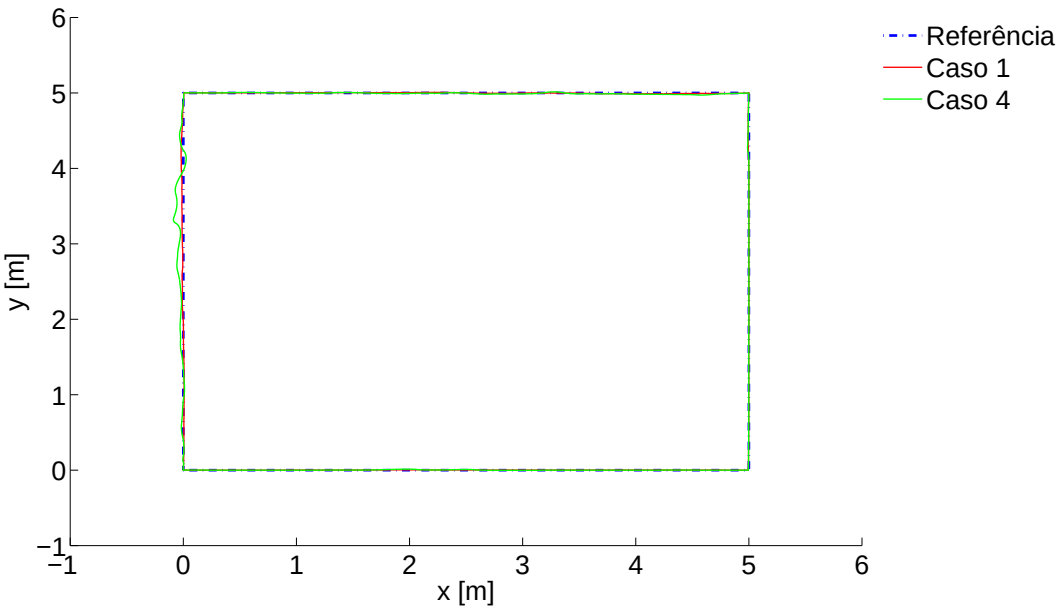


Figura 6.27: Deslocamento no plano XY do veículo para $(\omega_{n_1}, \zeta_1) = (3, 1.7)$, $(\omega_{n_2}, \zeta_2) = (3, 1.3)$, $(\omega_{n_3}, \zeta_3) = (2, 1.0)$ e $(\omega_{n_6}, \zeta_6) = (2, 0.8)$, para o Caso 1 e para o Caso 4.

6.4 Conclusões

Os resultados obtidos foram parametrizados em função dos parâmetros de projeto do controlador, de forma que estes podem ser escolhidos dentre um conjunto (ω_{n_i}, ζ_i) tal que o sistema seja estável (no sentido quadrático médio) e apresente um comportamento desejado (desempenho), mesmo em presença de atrasos devido ao sistema computacional.

Os resultados obtidos com o TrueTime permitem avaliar como as latências se comportam em função da carga na rede, da carga na CPU e da freqüência natural (ω_{n_i}) desejada para o sistema, ou mais precisamente do período de amostragem nominal h do ciclo de controle. O comportamento das latências são descritas por suas respectivas FDPs.

Estas FDPs foram utilizadas na formulação de Modelos Lineares de Saltos Markovianos para os GLs *surge*, *sway*, *heave* e *yaw*. E estes modelos foram utilizados para o cálculo da função de custo J através do Jitterbug para cada caso de atraso e para cada par (ω_{n_i}, ζ_i), com $\omega_{n_i} = \{1, 2, 3\} rad/s$ e ζ_i variando no intervalo [0.1,2.1]. Os valores mínimos de J indicam os parâmetros (ω_{n_i}, ζ_i) ótimos (pontos de melhor relação de energia do sistema) para cada caso.

Com os parâmetros ótimos obtidos com o Jitterbug, implementou-se os controladores

no modelo do VSOR em realidade virtual para visualização do comportamento temporal do sistema em presença de atrasos. Foi possível constatar que, para os valores de ω_{n_i} e para a tarefa adotados, os atrasos devido ao sistema computacional não são os determinantes para o bom desempenho do sistema no tempo, ou seja, a sensibilidade do sistema aos atrasos é baixa. O melhor desempenho no tempo para a realização da tarefa foi obtido para o sistema com menor tempo de resposta. Uma possível explicação é que o modelo do VSOR envolve plantas tipicamente estáveis (funções de transferência de primeira ordem com pólos no SPE), diferentemente das adotadas no trabalho de Cervin et al. (2006) onde a sensibilidade do sistema aos atrasos é alta.

O modelo do sistema com atraso em tempo discreto mostrado no Capítulo 2, embora não utilizado neste presente capítulo, poderia ser útil na análise me freqüência da sensibilidade do sistema frente aos atrasos (utilizando o pior caso de atraso obtido através do TrueTime). Com isso seria possível criar barreiras de desempenho e estabilidade em freqüência para uso em técnicas de projeto de controladores robustos.

7 Conclusões

Este trabalho teve como foco o estudo de uma nova classe de sistemas de controle (sistemas de controle distribuídos em redes de comunicação – NCS), tendo como objetivo o desenvolvimento de uma solução de controle para um ROV (VSOR), utilizando a referida arquitetura. Neste trabalho, foi abordado o problema da presença de atrasos inseridos pelo sistema computacional, mais especificamente atrasos devido ao escalonamento de processos e ao tráfego de mensagens na rede. Discutiu-se também possíveis candidatas para rede de controle para NCSs.

Para análise do impacto destes atrasos em sistemas de controle, foram apresentados dois métodos:

- utilizando um modelo de sistema com atraso e em tempo discreto, que permite fazer a análise em freqüência do sistema para o pior caso (atraso constante);
- utilizando as ferramentas TrueTime e Jitterbug, que, em conjunto, analisam o impacto de atrasos aleatórios sobre o sistema de controle através da teoria de Sistemas Lineares com Saltos Markovianos (MJLS).

E aplicando estes métodos em um caso de estudo, foi possível constatar que existe uma relação de compromisso entre o desempenho do sistema de controle e o desempenho do sistema computacional: um período de amostragem longo para o ciclo de controle prejudica o sistema de controle e ao mesmo tempo beneficia o sistema computacional, e vice-versa.

Foi apresentado também o modelo do ROV em questão, onde o sistema é descrito por 6 equações diferenciais desacompladas não-lineares referentes aos GLs do veículo, sob a hipótese de operações de baixas velocidades. O modelo inclui ainda uma matriz de alocação de empuxo dada em função da geometria do veículo e controladores do tipo PI com estrutura *feedback-feedfoward*, parametrizados em função de duas variáveis de projeto: ω_{n_i} e ζ_i.

O desempenho e estabilidade do sistema para alguns valores de (ω_{n_i}, ζ_i) foram analisados através do TreuTime e Jitterbug, sob alguns casos de atrasos. Para cada um destes casos foram obtidos pontos ótimos (considerados pontos de melhor relação de energia na entrada e na saída do sistema), que foram testados no modelo do ROV em realidade virtual. Os resultados mostram que para os intervalos de (ω_{n_i}, ζ_i) analisados, os atrasos devido ao sistema computacional não influem muito no desempenho e estabilidade do sistema de controle na realização de uma determinada tarefa. Atribui-se isto ao fato do modelo do veículo envolver plantas tipicamente estáveis (funções de transferência de primeira ordem com pólos no SPE).

Mostrou-se que, do ponto de vista de desempenho e estabilidade de sistemas com atrasos, é viável a implantação do NCS proposto sob as condições de operação citadas para o controle do VSOR, proporcionando ao sistema a redução de custo e do peso; a facilidade de diagnóstico e manutenção do sistema; e maior agilidade e flexibilidade.

7.1 Sugestões para Trabalhos Futuros

Sugere-se para trabalhos futuros o uso de dados experimentais para simulações com o TrueTime de forma que a simulação se aproxime mais das situações encontradas na prática. Como por exemplo a medição do WCET através de ferramentas específicas. A análise neste trabalho considerou que todas as *tasks* de controle possuem o mesmo período de amostragem. Sugere-se que combinações de diferentes períodos de amostragem sejam estudados, com especial atenção no problema da alocação de empuxo. Outros parâmetros podem ser alterados, como o algoritmo de escalonamento de processos e o tipo de rede.

Neste trabalho tratou-se apenas do problema dos atrasos inseridos pelo sistema computacional. Porém outro problema que pode ser encontrado, especialmente em se tratando de redes de comunicação, é a perda de dados. Sugere-se que sejam estudados os efeitos da perda de dados durante o ciclo de controle.

Pode-se também aplicar outras técnicas de projeto de controladores, como por exemplo técnicas de controle robusto. E também estudar a aplicação de técnicas para diminuir o impacto de atrasos e perdas de dados, como visto em Montestruque e Antsaklis (2005a), através da técnica de *model-based*.

Anexo A – Robustez de Sistemas SISO

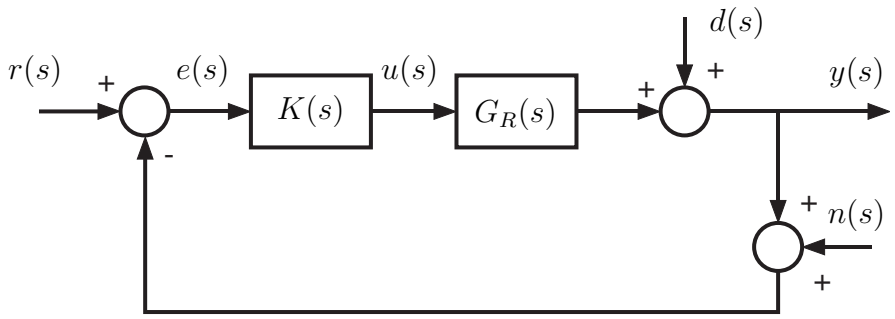


Figura A.1: Diagrama do sistema em malha fechada e em tempo contínuo.

O modelo real de um sistema de controle típico é representado pelo diagrama de blocos na Figura A.1. Nele são apresentados em freqüência:

- o sinal de referência $r(s)$,
- o sinal de erro $e(s)$,
- o controlador $K(s)$,
- o sinal de controle $u(s)$,
- o modelo real da planta $G_R(s)$,
- o sinal de distúrbio $d(s)$,
- o sinal de ruído $n(s)$,
- o sinal de saída $y(s)$.

Para este sistema, o sinal de saída é representado por

$$y(s) = S(s)L(s)r(s) + S(s)d(s) - S(s)L(s)n(s), \tag{A.1}$$

e o sinal de erro é representado por

$$e(s) = S(s)r(s) - S(s)d(s) - S(s)n(s), \tag{A.2}$$

onde $S(s)$ é a chamada função de sensibilidade do sistema definida por

$$S(s) = \frac{1}{1 + L(s)}, \tag{A.3}$$

e $L(s)$ é a chamada função de malha aberta do sistema definida por

$$L(s) = G_R(s)K(s). \tag{A.4}$$

Sobre este modelo real aplica-se a técnica de controle robusto[1] (CRUZ, 1996). O objetivo neste caso é garantir a estabilidade e o desempenho do sistema de controle mesmo sob o efeito das variações provocadas pela dinâmica do sistema computacional. Para garantir a estabilidade robusta deste sistema, utiliza-se o Critério de Nyquist que permite afirmar que a malha fechada é dada por

$$C_R(s) = \frac{G_R(s)K(s)}{1 + G_R(s)K(s)} \tag{A.5}$$

é estável se e somente se o número de envolvimentos no sentido anti-horário do Diagrama de Nyquist de $L(j\omega)$ em torno do ponto $-1 + j0$ é igual ao número de pólos instáveis de $L(s)$. Esta condição pode ser expressa por

$$[1 + \theta\varepsilon_M(j\omega)]G_N(j\omega)K(j\omega) \neq -1 + j0, \text{ para } \theta \in [0, 1]. \tag{A.6}$$

Ou seja, a área demarcada pelos círculos tracejados na Figura 2.5 não deve conter o ponto $-1 + j0$. Por sua vez, esta condição pode ser expressa por

$$\left| \frac{G_N(j\omega)K(j\omega)}{1 + G_N(j\omega)K(j\omega)} \right| < \frac{1}{e_M(\omega)}, \tag{A.7}$$

chamada por Cruz (1996) de Condição de Robustez de Estabilidade. Adotando novamente a hipótese de que os efeitos deste erro de modelagem sejam mais perceptíveis em altas freqüências (ou seja, $e_M(\omega) \gg 1$), a condição pode ser aproximada para

$$|G_N(j\omega)K(j\omega)| < \frac{1}{e_M(\omega)}, \text{ para } \{\omega \mid e_M(\omega) \gg 1\}. \tag{A.8}$$

[1]Robustez é a característica que permite ao sistema de controle obter um desempenho satisfatório, mesmo em condições de operação não explicitadas no projeto do controlador.

Desta forma o ganho da função de malha aberta nominal deve ser "pequeno"(ou seja, $|G_N(j\omega)K(j\omega)| \ll 1$) em altas freqüências para garantir a estabilidade do sistema.

Referências

ANDERSSON, M.; HENRIKSSON, D.; CERVIN, A. *TRUETIME 1.5 - Reference Manual*. [S.l.], June 2007.

AVILA, J. P. J. *Modelagem e identificação de parâmetros hidrodinâmicos de um veículo robótico submarino*. Tese (Doutorado) — Escola Politécnica da Universidade de São Paulo, 2008.

AVILA, J. P. J.; MARUYAMA, N.; ADAMOWSKI, J. C. Hydrodynamic parameter estimation of an open frame unmanned underwater vehicle. In: *Proceedings of the 17th World Congress. The International Federation of Automatic Control, Seoul, Korea.* [S.l.: s.n.], 2008.

BROCKETT, R. W. Stabilization of motor networks. In: *Proc. 34th IEEE Conf. on Dec. and Control*. [S.l.: s.n.], 1995. p. 1484–1488.

CACCIA, M.; INDIVERI, G.; VERUGGIO, G. Modeling and identification of open-frame variable configuration unmanned underwater vehicles. *Oceanic Engineering, IEEE Journal of*, v. 25, n. 2, p. 227–240, 2000. ISSN 0364-9059.

CACCIA, M.; VERUGGIO, G. Guidance and control of a reconfigurable unmanned underwater vehicle. *Control Engineering Practice*, v. 8, n. 1, p. 21–37, jan. 2000. Disponível em: <http://www.sciencedirect.com/science/article/B6V2H-3Y51H01-4/2/fd60b2e613260d7e5c02eddca88abe8f>.

CATSOULIS, J. *Designing Embedded Hardware*. [S.l.]: O'Reilly, 2005.

CERVIN, A. *Integrated Control and Real-Time Scheduling*. Tese (Doutorado) — Department of Automatic Control, Lund University, Sweden, abr. 2003.

CERVIN, A.; ARZEN, K.-E.; HENRIKSSON, D.; LLUESMA, M.; BALBASTRE, P.; RIPOLL, I.; CRESPO, A. Control loop timing analysis using truetime and jitterbug. In: *Computer-Aided Control Systems Design, 2006 IEEE International Symposium on*. [S.l.: s.n.], 2006. p. 1194–1199.

CERVIN, A.; HENRIKSSON, D.; LINCOLN, B.; EKER, J.; ARZEN, K.-E. How does control timing affect performance? analysis and simulation of timing using jitterbug and truetime. *Control Systems Magazine, IEEE*, v. 23, n. 3, p. 16–30, 2003. ISSN 0272-1708.

CERVIN, A.; LINCOLN, B. *Jitterbug 1.21 Reference Manual*. Box 118, SE 221 00 Lund, Sweden, February 2006.

CHOW, M.-Y.; TIPSUWAN, Y. Gain adaptation of networked dc motor controllers based on qos variations. *Industrial Electronics, IEEE Transactions on*, v. 50, n. 5, p. 936–943, 2003. ISSN 0278-0046.

CLARKE, D. W.; MOHTADI, C.; TUFFS, P. S. Generalized predictive control - part i. the basic algorithm. *Automatica*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 23, n. 2, p. 137–148, 1987. ISSN 0005-1098.

CRUZ, J. J. *Controle Robusto Multivariável*. [S.l.]: Edusp, 1996.

DAVIDSON, C. *Random sampling and random delays in optimal control systems*. Tese (Doutorado) — Royal Institute of Technology, 1973.

DEITEL, H. M.; DEITEL, P. J.; CHOFFNES, D. R. *Sistemas Operacionais*. [S.l.]: Prentice Hall, 2005.

FARINES, J.-M.; PRAGA, J. da S.; OLIVEIRA, R. da Silva de. *Sistemas de Tempo Real*. [S.l.]: IME-USP, 2000.

FOSSEN, T. I. *Guidance and control of ocean vehicles*. [S.l.]: John Wiley & Sons, 1994.

GAMBIER, A. Real-time control systems: a tutorial. In: *Control Conference, 2004. 5th Asian*. [S.l.: s.n.], 2004. v. 2, p. 1024–1031 Vol.2.

HENRIKSSON, D. *Resource-Constrained Embedded Control and Computing Systems*. Tese (Doutorado) — Department of Automatic Control, Lund Institute of Technology, Sweden, jan. 2006.

HENRIKSSON, D.; CERVIN, A.; ANDERSSON, M.; ÅRZèN, K.-E. Truetime: Simulation of networked computer control systems. In: *Proceedings of the 2nd IFAC Conference on Analysis and Design of Hybrid Systems*. Alghero, Italy: [s.n.], 2006. Invited talk.

KAPLAN, G. Ethernet's winning ways. *Spectrum, IEEE*, v. 38, n. 1, p. 113–115, 2001. ISSN 0018-9235.

KOUBIAS, S. A.; PAPADOPOULOS, G. D. Modern fieldbus communication architectures for real-time industrial applications. *Computers in Industry*, v. 26, n. 3, p. 243–252, ago. 1995. ISSN 0166-3615. Disponível em: <http://www.sciencedirect.com/science/article/B6V2D-3YF4GSB-5/2/e84f92c5abb6b6ef05dce1fb90aa4738>.

LIAN, F.-L. *Analysis, Design, Modeling, and Control of Networked Control Systems*. Tese (Doutorado) — University of Michigan, 2001.

LIAN, F.-L.; MOYNE, J.; TILBURY, D. Network design consideration for distributed control systems. *Control Systems Technology, IEEE Transactions on*, v. 10, n. 2, p. 297–307, 2002. ISSN 1558-0865.

LINCOLN, B. *Dynamic programming and time-varying delay systems*. Tese (Doutorado) — Lund Institute of Technology, 2003.

MONTESTRUQUE, L.; ANTSAKLIS, P. Networked control systems: a model-based approach. In: *IAESTED International Conference on Networks and Communication Systems, Thailand*. [S.l.: s.n.], 2005.

MONTESTRUQUE, L.; ANTSAKLIS, P. Quantization in model based networked control systems. In: *Proceedings of the 16th IFAC World Congress, Prague*. [S.l.: s.n.], 2005.

MONTESTRUQUE, L.; ANTSAKLIS, P. *Performance Evaluation for Model-Based Networked Control Systems*. 2006. 231–249 p.

NILSSON, J. *Real-time control systems with delays*. Tese (Doutorado) — Lund Institute of Technology, 1998.

OOI, J.; VERBOUT, S.; LUDWIG, J.; WORNELL, G. A separation theorem for periodic sharing information patterns in decentralized control. *IEEE Trans. Autom. Control*, Vol 42, p. pp. 1546–1550., 1997.

OTANEZ, P. G.; PARROTT, J. T.; MOYNE, J. R.; TILBURY, D. M. The implications of ethernet as a control network. In: *Global Powertrain Conference*. [S.l.: s.n.], 2002.

POSTLETHWAITE, S. *Multivariable Feedback Control*. [S.l.]: Wiley, 1996.

ÅRZèN, K.-E.; CERVIN, A. Control and embedded computing: Survey of research directions. In: *Proc. 16th IFAC World Congress*. Prague, Czech Republic: [s.n.], 2005.

SAAB, T. Site acessado em Jan 2009. Http://www.seaeye.com/.

SANDELL, D. *Evaluating Static Worst-Case Execution-Time Analysis for a Commercial Real-Time Operating System*. Dissertação (Mestrado) — Dept. of Computer Science - Mälardalen University, 2004.

SILJAK, D. *Decentralized control of complex systems*. [S.l.]: Academic Press, 1991.

SOUZA, E. C. *Modelagem e controle de veículos submarinos não tripulados*. Dissertação (Mestrado) — Escola Politécnica da Universidade de São Paulo, 2003.

ÅSTRöM, K. J.; HäGGLUND, T. *PID Controllers: Theory, Design, and Tuning*. [S.l.]: International Society for Measurement and Con, 1995.

ÅSTRöM, K. J.; WITTENMARK, B. *Computer-controlled systems: theory and design*. [S.l.]: Prentice Hall, 1997.

TANENBAUM, A. S. *Sistemas Operacionais Modernos*. [S.l.]: Prentice Hall, 2003.

TISCHLER, M. B. *Advances In Aircraft Flight Control*. [S.l.]: Taylor&Francis, 1996.

WALSH, G.; WALSH, G.; YE, H.; BUSHNELL, L. Stability analysis of networked control systems. *Control Systems Technology, IEEE Transactions on*, v. 10, n. 3, p. 438–446, 2002. ISSN 1063-6536.

WITTENMARK, B.; NILSSON, J.; TORNGREN, M. Timing problems in real-time control systems. In: *American Control Conference, 1995. Proceedings of the*. [S.l.: s.n.], 1995. v. 3, p. 2000–2004 vol.3.

WOLOVICH, W. A. *Automatic control systems. Basic analysis and design.* [S.l.]: Harcourt Brace College Publishers, 1994.

WONG, W.; BROCKETT, R. Systems with finite comunnication bandwidth constraints ii: Stabilization with limited information feedback. *IEEE Trans. Autom. Control*, Vol 44, p. pp. 1049–1053, 1999.

YAMAMOTO, Y.; KHARGONEKAR, P. Frequency response of sampled-data systems. *Automatic Control, IEEE Transactions on*, v. 41, n. 2, p. 166–176, 1996. ISSN 0018-9286.

Apêndice A – Parâmetros do VSOR

Os valores das variáveis citadas no Capítulo 5 do VSOR são apresentados na Tabela A.1.

GL	i	m_i	d_{L_i}	d_{Q_i}	b_i
Surge	(1)	$841.93Kg$	$82.3Kg/s$	$309.07Kg/m$	$-0.05N$
Sway	(2)	$841.93Kg$	$8.5Kg/s$	$505.45Kg/m$	$-0.14N$
Heave	(3)	$839.37Kg$	$50.4Kg/s$	$759.73Kg/m$	$-0.85N$
Roll	(4)	$70.51Kg.m^2/rad$	$0Kg.m^2/(rad.s)$	$174.82Kg.m^2$	$0.16N.m$
Pitch	(5)	$70.51Kg.m^2/rad$	$0Kg.m^2/(rad.s)$	$174.82Kg.m^2$	$0.16N.m$
Yaw	(6)	$206.74Kg.m^2/rad$	$18.21Kg.m^2/(rad.s)$	$94.72Kg.m^2$	$-2.53N.m$

Tabela A.1: Dados do VSOR referentes aos GLs (AVILA, 2008).

Para o cálculo das forças restaurativas , adotou-se:

- gravidade $g = 10m/s^2$,
- densidade do meio (água) $\rho = 1000Kg/m^3$,
- volume deslocado $\forall = 0.19265m^3$ (AVILA, 2008),
- altura do centro de flutuação $z_B = 0.09m$ (AVILA, 2008),
- massa do veículo $m = 188.6Kg$ (AVILA, 2008).

Portanto, a força de flutuação é dada por

$$B = mg - \rho g \forall = 40.5N.$$

Para a alocação de empuxo, adotou-se:

- $a = 0.40m$,

- $b = 0.50m$,
- $c = 0.15m$,
- $d = 0.65m$,
- $e = 0.45m$,
- $\beta = \pi/4$.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo