DISSERTAÇÃO DE MESTRADO

DESENVOLVIMENTO DE CONTROLADOR PREDITIVO PARA CONTROLE DE ATITUDE DE SATÉLITES E VALIDAÇÃO EM HIL

REURISON SILVA RODRIGUES

Brasília, Junho de 2018

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

DESENVOLVIMENTO DE CONTROLADOR PREDITIVO PARA CONTROLE DE ATITUDE DE SATÉLITES E VALIDAÇÃO EM HIL

REURISON SILVA RODRIGUES

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA MECÂNICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM SISTEMAS MECATRÔNICOS

APROVADA POR:

Prof. Dr. André Murilo de Almeida, Gama/UnB
Orientador

Prof. Dr. Guilherme Caribé de Carvalho, ENM/UnB
Examinador interno ao PPMEC

Prof. Dr. Luiz Carlos Gadelha de Souza, CECS/U-FABC
Examinador externo ao PPMEC

BRASÍLIA/DF, 15 JUNHO DE 2018

Silva Rodrigues, Reurison
Desenvolvimento de controlador preditivo para controle de atitude de satélites e validação em HIL. / REURISON SILVA RODRIGUES. –Brasil, 2018.
100 p.

Orientador: André Murilo de Almeida Pinto
Dissertação (Mestrado) – Universidade de Brasília – UnB
Faculdade de Tecnologia – FT
Programa de Pós-Graduação em Sistemas Mecatrônicos – PPMEC, 2018.

1. Controle preditivo. 2. Satélites artificiais. 3. *Hardware-in-the-loop*. 4. *Beaglebone*. 5. Controle de atitude de satélites. I. André Murilo de Almeida Pinto, orientador. II. Universidade de Brasília. III. Faculdade de Tecnologia.

Agradecimentos

Primeiramente aos meus pais Francisco Rodrigues e Maria das Graças pelo grande esforço que sempre fizeram para que eu fosse atrás dos meu sonhos. Obrigado por todo seu apoio e compreensão que sem dúvida me ajudaram a chegar onde hoje estou.

Gostaria de agradecer ao meu orientador André Murilo pela sua dedicação a este trabalho, os ensinamentos passados, sua orientação e disponibilidade em sempre me ajudar nos momentos difíceis.

Ao meu coorientador Renato Vilela Lopes pelas orientações acadêmicas, troca de ideias e suporte durante o meu mestrado.

Ao professor Renato Coral pelas contribuições para o desenvolvimento deste trabalho.

Aos amigos que fiz durante o mestrado: Rafael Rodrigues, Pedro Jorge, Alceu Castanheira e Fabián Barrera. Por todas as discussões técnicas, conselhos e os inúmeros momentos de descontração que tornaram essa jornada mais fácil.

Agradeço ao Programa de Pós-Graduação em Sistemas Mecatrônicos (PPMEC) da UNB pelo suporte no desenvolvimento do trabalho.

Agradeço à CAPES e ao CNPQ pelo fomento, incentivo e auxílio financeiro na realização deste trabalho.

E por último, mas não menos importante, a minha esposa Larissa Évelin por todo carinho, paciência e companheirismo durante todo esse trajeto.

REURISON SILVA RODRIGUES

RESUMO

Os satélites artificiais são sistemas robóticos extremamente complexos e caros, empregados em aplicações científicas, militares e de comunicação. O controle de atitude de satélites é responsável por assegurar que o objeto espacial se encontre na posição, velocidade e trajetória corretas, estabilizando o veículo espacial e o orientando nas direções desejadas durante a missão, independente de perturbações externas. É importante que o projeto do controlador em desenvolvimento seja capaz de respeitar requisitos de segurança e funcionamento do satélite, lidando com restrições próprias do sistema e ao ambiente em que ele está inserido. Caso estes requisitos não sejam respeitados, falhas podem ocorrer, ocasionado em funcionamento não adequado do satélite e possível falha da missão. Para evitar estes erros, o projeto dos controladores deve seguir uma metodologia de testes e validação antes de serem postos em operação. Um das dificuldades relacionadas a esta etapa é a não possibilidade de usar o satélite real para testar/validar as leis de controle desenvolvidas. Levando-se em consideração a dificuldade de validação experimental dos controladores e que considerem na sua formulação as restrições físicas inerentes ao satélite, este trabalho tem como objetivo o desenvolvimento de um controlador preditivo não linear (NMPC) embarcado em hardware, para o controle de atitude de uma plataforma de testes de satélites com validação do controlador em uma arquitetura do tipo *Hardware-in-the-loop* (HIL). O controle preditivo é uma estratégia que faz uso do modelo do processo e que prediz o comportamento futuro do mesmo sob um horizonte de predição definido. A estratégia leva em consideração restrições inerentes ao sistema (de estado, comando e variação do comando), e através da otimização *online* de uma função custo, definida pelo projetista, gera uma sequência ótima de comandos. Um dos pontos críticos da utilização do MPC embarcado em hardware para situações em tempo real é o alto custo computacional existente na etapa de otimização. Quanto mais rápida for a dinâmica do sistema, exigindo menores períodos de amostragem, mais crítico torna-se este problema. Neste trabalho visando reduzir o custo computacional e assegurar que o controlador possa ser embarcado em um hardware real, é então utilizada a técnica de parametrização exponencial, que reduz o número graus de liberdade existentes no problema de otimização não-parametrizado, reduzindo diretamente o tempo gasto na etapa de otimização. Para validação em tempo real do controlador desenvolvido, simulações em HIL são feitas em uma plataforma de baixo custo desenvolvida na Universidade de Brasília. Os resultados obtidos mostram a possibilidade de se embarcar o controlador preditivo não-linear parametrizado em um hardware de baixa potência e que o sistema de controle de atitude é capaz de lidar com as restrições impostas pelo satélite.

ABSTRACT

Artificial satellites are extremely complex and expensive robotic systems used in scientific, military and communication applications. The attitude control of satellites is responsible for ensuring that the space object is in the correct position, velocity and trajectory, stabilizing the spacecraft and guiding it in the desired directions during the mission, regardless of external disturbances. It is important that the design of the developing controller be able to meet security requirements of the satellite, dealing with constraints of the system itself and the environment in which it is embedded. If these requirements are not met, failures may occur, causing the satellite to malfunction and possible mission failure. In order to avoid these errors, the design of controllers must follow a methodology of testing and validation before they are put into operation. One of the difficulties related to this step is the non-possibility of using the real satellite to test/validate the developed control laws. Taking into account the difficulty of experimental validation of the controllers that consider in their formulation the physical restrictions inherent to the satellite, this work aims at the developing of a non-linear model predictive control (NMPC) embedded in hardware, for the attitude control of a platform of satellite tests with validation in a Hardware-in-the-loop architecture. The model predictive control is a strategy that makes use of the nonlinear model of the process and that predicts the future behavior of the process under a defined prediction horizon. The strategy takes into account constraints inherent to the system (state, command and command variation) and through online optimization of a cost function, defined by the designer, generates an optimal sequence of commands. One of the critical points of NMPC embedded in hardware for real-time situations is the high computational cost in the optimization stage. The faster the system dynamics, requiring shorter sampling time, the more critical this problem becomes. In this work, in order to reduce the computational cost and ensure that the controller can ben embedded in real hardware, the exponential parametrization technique is used, which reduces the number of degrees of freedom in the non-parametrized optimization problem, directly reducing the time spent in the optimization step. For real-time validation of the developed NMPC controller, HIL simulations are carried out on a low cost platform developed at University of Brasilia (UnB). The results obtained show the possibility of embedding the parametrized nonlinear predictive control in a low power hardware and that the control system is able to handle the constraints imposed by the satellite.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E ACROGRAMAS

ACS	Attitude Control System
ADS	Attitude Determination System
AEB	Agência Espacial Brasileira
ADCS	Attitude Determination Control System
BB	BeagleBone Black
CBERS	China-Brazil Earth-Resources Satellite
DAC	Digital-to-Analog Converter
DMC	Dynamic Matrix Control
DMPC	Distributed Model Predictive Control
FPGA	Field Programmable Gate Array
GPS	Global Positioning System
GPIO	General Purpose Input/Output
HIL	Hardware-in-the-loop
ISS	International Space Station
IDCOM	Identification and Command
INPE	Instituto Nacional de Pesquisas Espaciais
LQR	Linear-Quadratic Regulator
LTI	Linear Time-Invariant
MATLAB	Matrix Laboratory
MDB	Model Based Design
MIMO	Multiple-input and multiple-output
MIL	Model-in-the-loop

MMPC	Multiplexed Model Predictive Control
MPC	Model Predictive Control
MPHC	Model Predictive Heuristic Control
NMPC	Nonlinear Model Predictive Control
PID	Proporcional, Integral e Derivativo
PIL	Processor-in-the-loop
PINOCCHIO	Platform Integrating Navigation and Orbital Control Capabilities Hostint Intelligence Onboard
QDMC	Quadratic Dynamic Matrix Control
qpOASES	Quadratic Programming Online Active Set Strategy
RTOS	Real Time Operating System
SCD	Satélite de Coleta de Dados
SDRE	State-Dependent Riccation Equation
SHP	Laboratório de Sistemas Hidráulicos e Pneumáticos
SIL	Software-in-the-loop
SLRT	Simulink Real-Time
SQP	Sequential Quadratic Programming
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol

LISTA DE SÍMBOLOS

θ	Ângulo da plataforma de satélites	$[rad]$
ω	Velocidade angular da plataforma de satélites	$[\frac{rad}{s}]$
Ω	Velocidade angular da roda de reação	$[\frac{rad}{s}]$
$\dot{\Omega}$	Aceleração angular da roda de reação	$[\frac{rad}{s^2}]$
I_{11}, I_{22}, I_{33}	Momento de inércia da plataforma de satélites nos eixos i_1, i_2, i_3	$[\frac{kg}{m^2}]$
I_w	Momento de inércia da roda de reação	$[\frac{kg}{m^2}]$
$A(.)$	Matriz de estados do sistema	
$B(.)$	Matriz de entrada	
$C(.)$	Matriz de saída	
$D(.)$	Matriz de realimentação	
$x(.)$	Vetor de estados	
$u(.)$	Vetor de controle	
$y(.)$	Vetor de saída	
τ	Tempo de amostragem do processo	
k	Instante de amostragem	
N	Horizonte de predição	
$x^u(k)$	Trajetória de estados preditos	
$\hat{u}(x)$	Sequência de controle ótima	
u_{min}, u_{max}	Restrição inferior e superior de comando	$[\frac{rad}{s^2}]$
$\delta_{min}, \delta_{max}$	Restrição inferior e superior da variação de comando	$[\frac{rad}{s^2}]$
n_u	Número de atuadores do sistema	
n_p	Número de graus de liberdade do controle parametrizado	
n_e, α, t_r	Coeficientes da parametrização exponencial MPC	
Q_y	Matriz de ponderação de estados	
Q_u	Matriz de ponderação de comandos	
$_x$	Ponderação de estado final	
x_v	Variável de estado em unidade de tensão na plataforma HIL	
$x_{i(min,max)}$	Limite inferior e superior do estado i em unidade de tensão na plataforma HIL	
q, λ	Coeficientes da parametrização exponencial NMPC	
u_n^*	Comando em estado estacionário para o NMPC	
p_1, p_2, p_3	Parâmetro relacionado ao regime transiente de comando	
p_4, p_5, p_6	Parâmetro relacionado ao regime estacionário de comando	

Capítulo 1

Introdução

Os satélites desde seu surgimento sempre foram de grande utilidade para a humanidade. Eles estão presentes no dia a dia auxiliando na comunicação entre grades distâncias, envio de sinal de internet, televisão e telefone, informações de posicionamento GPS, monitoramento de queimadas e desmatamento, informações meteorológicas e sensoreamento remoto. O Brasil, por meio da Agência Espacial Brasileira (AEB), conta com alguns satélites nacionais. Um dos primeiros satélites brasileiros foi o Satélite de Coleta de Dados (SCD) 1, lançado em 1993, e construído no Instituto Nacional de Pesquisas Espaciais (INPE), com o objetivo de coletar dados ambientais. Posteriormente, em 1998, foi lançado o SCD-2 com os mesmo objetivos do seu sucessor. Em 1999 foi lançado o Satélite Sino-Brasileiro de Recursos Terrestres (CBERS-1) em parceria com a China, com a missão de coletar dados ambientais, monitorar áreas de desmatamento, áreas agrícolas, recursos hídricos e crescimento urbano. Em 2003 foi a vez do CBERS-2 substituir o CBERS-1. O último satélite dessa série foi o CBERS-4 lançado em 2014. A série CBERS tornou o Brasil o maior distribuidor de imagens de satélite do mundo (AEB, 2011). Ainda em 2014 foi o ano do lançamento do *NanosatC-Br1*, primeiro nano satélite brasileiro.

Apesar dos satélites citados anteriormente realizarem diferentes missões, um fator comum entre todos eles e que faz com que desempenhem sua função corretamente é a existência de uma unidade de controle de atitude. A atitude por sua vez é definida como a orientação de uma espaçonave em relação a um sistema de referência conhecido. Podendo ser dividido em: (a) estabilização, processo que mantém o satélite na orientação existente e (b) controle de manobra, que é o procedimento de reorientar a aeronave em outra direção (WERTZ, 1978). Os assim chamados Sistemas de Controle e Determinação de Atitude (ADCS) consistem em um conjunto de sensores e atuadores controlados por algoritmos embarcados em microcontroladores (LIN; JUANG; VINA, 2014). Ele pode ser dividido entre Sistema de Determinação de Atitude *(Attitude Determination System - ADS)* e Sistema de Controle de Atitude *(Attitude Control System - ACS)*, o primeiro sendo responsável por estimar a atitude atual do satélite por meio de sensores e algoritmos de estimação e o segundo responsável por orientar a espaçonave em uma direção específica pelo uso dos atuadores presentes no satélite (JENSEN; VINTHER, 2010). O ADCS é fundamental pois fornece informações de atitude do satélite e procura manter ou posicionar o satélite numa direção específica no espaço. São tarefas da unidade de controle de atitude, por exemplo: posicionar o satélite para que seus painéis solares fiquem voltados para o sol, para geração de energia, posicionamento das antenas de comunicação em direção a terra, além disso, os satélites que estão executando missões de sensoriamento

remoto dependem mais ainda do controle de atitude por que a câmera necessita apontar precisamente para o alvo e o corpo do satélite deve manter-se estável (Hannay, 2009). O controle também é requerido para evitar danos solares ou atmosféricos a componentes sensíveis e para controlar dissipação de calor (WERTZ, 1978).

O objetivo específico do ACS é agir sobre o erro de atitude do satélite, este erro é definido como a diferença entre a posição desejada e o valor de atitude atual, fornecido pelo ADS. O ACS gera, através de atuadores instalados no próprio satélite, os torques necessários para zerar este erro. Esse procedimento ocorre indefinidamente devido à existência de pertubações externas, e imperfeições nas medidas e ações de controle (SNIDER, 2010). O ADCS está presente em quase todos os satélites determinando diretamente o resultado das missões (CHEN, 2011) e sua sobrevivência em órbita (SILANI; LOVERA, 2005).

Os satélites, por serem estruturas tecnologicamente muito complexas que demandam anos de desenvolvimento e altos investimentos, devem ter reduzidas ao máximo qualquer possibilidade de erro durante sua fase de projeto. Falhas posteriores ao lançamento podem ou não ser corrigidas. Um dos exemplos é o caso do telescópio espacial *Hubble*, em que, após o seu lançamento em 1990, foi detectada uma falha no sistema de câmera (ALLEN et al., 1990) para então ser reparada em órbita em 1993. Do ponto de vista dos algoritmos de controle presentes no ACS, estes devem obedecer a requisitos operacionais e de segurança bem definidos. Problemas nesta fase podem levar situações como o ocorrido com o satélite japonês *Hitomi* que ficou rotacionando fora de controle, ultrapassou restrições estruturais e desintegrou-se no espaço, pondo fim a missão (NOW, 2017).

Considerando as restrições inerentes aos sistemas físicos como aspecto fundamental que qualquer lei de controle deve respeitar, é interessante considerar o uso do controlador preditivo baseado em modelos *(Model Predictive Control - MPC)*. Ela tem se tornado bastante popular devido a sua facilidade de implementação e capacidade de lidar com restrições explicitamente (MAYNE et al., 2000). Atualmente é citada como uma das técnicas mais promissoras aplicadas a sistemas aeroespaciais (EREN et al., 2017). O controle preditivo é uma técnica avançada que consiste em resolver um problema de otimização de horizonte deslizante, onde a solução é reiterada a cada período de amostragem baseado em informações de feedback dos sensores (CAMACHO; ALBA, 2007).

Um dos principais problemas associados à implementação do MPC em tempo real é a alta demanda computacional associada a etapa de otimização *online*. Este aspecto deve ser levado em consideração para realização de testes embarcados em *hardware*. Caso o *hardware* não seja capaz de realizar a otimização em tempo hábil o processo que está sendo controlado poderá se tornar instável. Para diminuir a carga computacional da otimização, podem-se utilizar técnicas de parametrização da variável de comando, entre as quais a exponencial é uma delas. A parametrização permite reduzir o número de variáveis presentes no problema de otimização impactando diretamente o tempo gasto no cálculo da solução ótima de controle, sendo capaz de viabilizar a aplicação do MPC em tempo real (ALAMIR, 2013).

Antes de serem embarcadas nos satélites, as leis de controle devem ser testadas e validadas experimentalmente em terra, sendo esta verificação uma das tarefas mais críticas no desenvolvimento do ACS (SILVA et al., 2014). Porém, estando em ambiente terrestre, simular ambientes com condições físicas semelhantes à aquelas encontradas no espaço como ambiente de baixo atrito, influência de campos magnéticos, gravidade zero e livres de torque não é fácil (KIM et al., 2003). Outro fator ainda mais crítico é a

impossibilidade de se utilizar o satélite real para execução de testes, fazendo-se necessário utilizar alguma metodologia que possibilite a validação destes controladores em ambiente terrestre. Uma das metodologias de desenvolvimento bastante utilizada na área aeroespacial, software e indústria automotiva (BERGSTRÖM; GÖRANSSON, 2016) para validação de *software* embarcado é o projeto baseado em modelos: *Model-Based Design (MDB)*. Esta metodologia está centrada no modelo do processo e fornece uma abordagem matemática e visual para o desenvolvimento de sistemas complexos. Ela tem sido descrita como uma abordagem capaz de reduzir o tempo e o custo de desenvolvimento de produtos (TOMAN; KERLÍN; SINGULE, 2011).

Conjuntamente com o MDB deve-se utilizar um fluxo de trabalho. Por isso torna-se interessante o uso do desenvolvimento em V *(V-Cycle)*. A união destes dois tipos de metodologias já foi aplicada com sucesso na verificação do ACS em (SILVA et al., 2014). A vantagem destes procedimentos mostra-se útil pois permite fazer uma análise minuciosa dos blocos constituintes do problema como um todo. Isto permite identificar erros em estágios iniciais de desenvolvimento com mais facilidade.

Umas das principais etapas envolvidas na validação dos controladores é a etapa de *Hardware-in-the-loop (HIL)*. Esta etapa é caracterizada pela operação de componentes reais em conexão com componentes simulados via *software* em tempo real (ISERMANN; SCHAFFNIT; SINSEL, 1999). Uma de suas grandes vantagens é a não necessidade da utilização do modelo físico do sistema sob testes, se tornando um bom candidato para validação de *software* embarcado no domínio dos satélites.

1.1 Contribuições do Manuscrito

O presente trabalho se propõe a utilizar o projeto baseado em modelos pelo método de desenvolvimento em V *(V-cycle)* no desenvolvimento de dois controladores preditivos, um linear e o outro não linear, ambos parametrizados, para o controle de atitude do modelo da plataforma de testes de satélites descrito por Gonzales (2009). Os controladores foram projetados com o intuito de levar em consideração as restrições de estados, comando e variação de comando do modelo. O uso da parametrização foi necessário para reduzir o tempo de cálculo das leis de controle para que fossem embarcadas em *hardware*. Por fim os controladores embarcados serão validados em condições de tempo real através de uma plataforma HIL de baixo custo desenvolvida na Universidade de Brasília (UnB).

1.2 Objetivos da Dissertação

1.2.1 Objetivo Geral

O objetivo geral deste trabalho é o desenvolvimento de um controlador preditivo linear (MPC) e outro não-linear (NMPC) para o controle de atitude de uma plataforma de testes de satélites e validação do mesmo em uma arquitetura HIL desenvolvida na Universidade de Brasília.

1.2.2 Objetivos Específicos

Para atingir o objetivo geral no presente trabalho, foram estabelecidos os seguintes objetivos específicos:

- Desenvolver o algoritmo de controle preditivo linear no *Matlab/Simulink*.
- Embarcar o código do controlador preditivo linear na *Beaglebone Black*.
- Desenvolver o algoritmo de controle não linear no *Matlab/Simulink*.
- Embarcar o código do controlador preditivo não linear na *BeagleBone Black*.
- Desenvolver uma plataforma PIL para validação dos códigos dos controladores desenvolvidos anteriormente.
- Desenvolver uma plataforma HIL para teste dos controladores.
- Validar controladores linear e não linear na plataforma HIL.

1.3 Apresentação do Documento

O presente trabalho é composto de 6 capítulos, divididos da seguinte maneira:

No capítulo 2 é feita uma revisão sobre os tipos de plataformas de testes de satélites, discutindo quais são suas vantagens no projeto de desenvolvimento de satélites, como funcionam e onde são aplicadas. Outro tópico discutido é a validação de controladores embarcados através da união da metodologia *Model-Based Design (MDB)* com o *V-cycle*. Esta metodologia é composta por uma série de etapas que facilita a validação do controlador desenvolvido. Por fim é feita uma revisão dos tipos de controladores utilizados para controle de atitude e por que o controle preditivo é uma opção a ser aplicado neste domínio.

No capítulo 3 apresenta-se o controle preditivo e sua formulação geral. Aqui são feitas as definições acerca deste controlador, seu desenvolvimento histórico e sua estrutura interna. Ainda neste capítulo realiza-se uma discussão das suas vantagens e desvantagens, principalmente no que diz respeito a sua aplicação em situações de tempo real quando embarcado em *hardware*, devido ao alto custo computacional nas etapas de otimização.

O capítulo 4 apresenta a metodologia de trabalho utilizada neste trabalho. Primeiramente discute-se o modelo da plataforma de testes de satélites utilizada para desenvolvimento dos controladores. Em seguida faz-se o desenvolvimento das equações do controlador preditivo linear (MPC) e não linear (NMPC)

No capítulo 5 apresentam-se os resultados obtidos na validação dos controladores NMPC e MPC em tempo real na plataforma HIL e discussões acerca de trabalhos futuros.

Por fim no capítulo 6 apresentam-se as conclusões e as sugestões de trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

Neste capítulo, serão abordadas as plataformas de testes de satélites, sua classificação e aplicações. Estas plataformas são úteis no projeto de desenvolvimento de satélites, pois fornecem um meio de se testar em terra condições próximas daquelas encontradas no espaço, outro motivo que as torna úteis é que o satélite real não pode ser utilizado para testes.

2.1 Simuladores de Satélites

Mesas de mancal aerostático tem sido utilizadas para testes e verificação de software e eletrônicos embarcados desde o início da corrida espacial. Elas preenchem a necessidade da realização de testes relacionados a satélites, dada a impossibilidade de usá-los diretamente. Por fornecerem um ambiente quase livre de torques externos, talvez o mais próximo possível do ambiente espacial, é uma das tecnologias preferidas em dinâmica e controle aeroespacial (LI; GAO, 2010). Estes equipamentos são essenciais pois auxiliam os projetistas seja na detecção de erros em fases iniciais de projeto, verificação de *software/hardware* embarcado e/ou aprimoramento dos mesmos, prevenindo e evitando falhas durante a missão. Além de que é possível testar uma grande quantidade de cenários possíveis sem danificar o satélite real.

O objetivo destas mesas é fornecer movimento de translação e/ou movimento angular em um ambiente de baixo atrito (SCHWARTZ; PECK; HALL, 2003). Este ambiente de baixo atrito deve ser próximo a aqueles que seriam encontrados no espaço por um satélite real, sendo este um ponto principal na validade das plataformas (OLIVEIRA; KUGA; CARRARA, 2015). Porém, fatores existentes nas simulações em terra como ambiente de gravidade 1g, arrasto aerodinâmico e os próprios atritos existentes no simulador devem ser levados em consideração diminuindo as similaridades com o ambiente espacial. As plataformas podem ser utilizadas para testes de: controle de atitude, operações de *docking* (acoplamento entre veículos espaciais) e controle orbital. Segundo Schwartz, Peck e Hall (2003) as plataformas podem ser divididas em três categorias de sistemas: planares, rotacionais e combinados.

2.1.1 Sistemas Planares

Estas plataformas utilizam mancais aerostáticos planos e permitem movimento planar e (não necessariamente) movimento de rotação vertical, sobre uma superfícies plana, geralmente de epóxi ou mármore. Este tipo de plataforma é utilizada para simular operações de *rendezvous*, ou seja, aproximação entre espaçonaves no espaço, formação de voo e *docking*. Estes sistemas carregam seus próprios tanques de ar, responsáveis pela criação do colchão de ar sob si próprio, criando uma superfície de baixo atrito. Jatos de gás podem ser utilizados para movimentos de translação, quando da existência de rotação vertical ela pode ser efetuada através de rodas de reação.

Em Romano, Friedman e Shay (2007) um sistema planar é utilizado para ensaios de aproximação autônoma e acoplamento no espaço, como pode ser visto na figura 2.1. Nesta configuração a plataforma pode flutuar sobre um superfície lisa de epóxi. O movimento de translação é obtido através do uso de jatos de gás e o movimento de rotação é conseguido por *thruster couples* e rodas de reação.

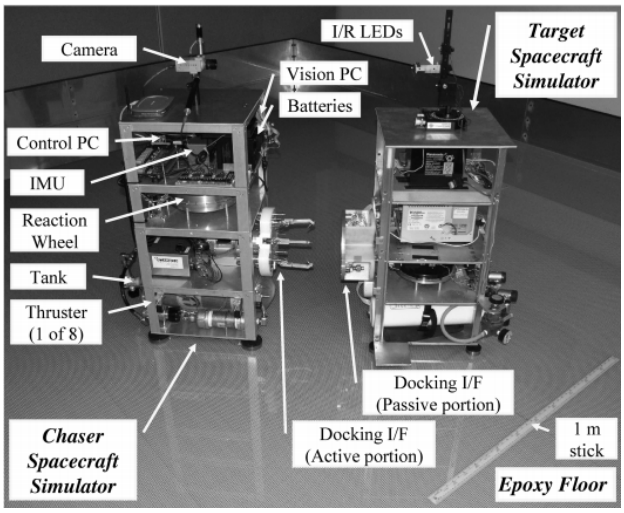


Figura 2.1: Sistema utilizado para simulações de acoplamento.
Fonte: Romano, Friedman e Shay (2007)

Outro tipo de sistema planar é braço robótico da Universidade de *Stanford*, figura 2.2. O braço robótico é utilizado para testar operações onde robôs são utilizados para construção e montagem em órbita, como o *CANADARM*, manipulador presente na estação espacial internacional *International Space Station (ISS)*, capaz de manobrar e capturar cargas no espaço (AIKENHEAD; DANIELL; DAVIS, 1983).

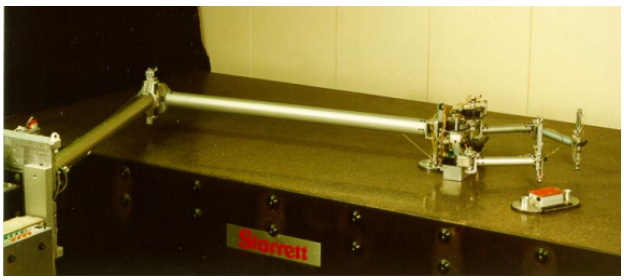


Figura 2.2: Braço robótico de dois elos de *Stanford*.
Fonte: (SCHUBERT; HOW, 1997)

Em Yao et al. (2016) uma plataforma com 3 graus de liberdade foi desenvolvida. O simulador, figura 2.3, pode flutuar em uma mesa de mármore sobre um colchão de ar gerado pela própria plataforma. Os atuadores utilizados para manobrar o simulador são rodas de reação e jatos de gás. O objetivo é validar estratégias de navegação, formação de voo, *rendezvous*, rastreamento de trajetória e desenvolver e testar estratégias de controle. Neste trabalho um controlador Proporcional-Derivativo foi desenvolvido para controlar a atitude e posicionamento do simulador.

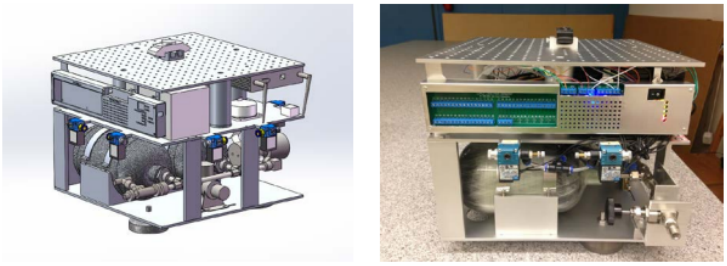


Figura 2.3: Sistema planar.
Fonte: (YAO et al., 2016)

Uma plataforma de baixo custo foi desenvolvida pela Universidade de Roma, chamada de *PINOCCHIO : Platform Integrating Navigation and Orbital Control Capabilities Hosting Intelligence Onboard* (SABATINI; FARNOCCHIA; PALMERINI, 2012), figura 2.4. Esta plataforma é utilizada para testes e verificação de estratégias de navegação em 2D, teste e validação de leis de controle, sensores de navegação e softwares de estimação. A plataforma pode flutuar livremente sobre uma superfície de baixo atrito e possui 3 graus de liberdade, sendo dois de translação e um de rotação. Sua movimentação é feita através da utilização de jatos de gás e rodas de reação. Outro exemplo de plataforma planar pode ser encontrado em Boning et al. (2008).

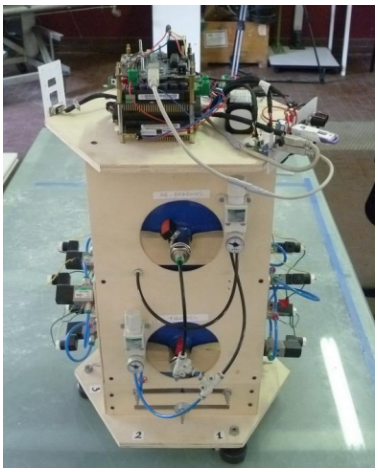


Figura 2.4: Plataforma PINOCCHIO.
Fonte: Sabatini, Farnocchia e Palmerini (2012)

2.1.2 Sistemas Rotacionais

Este tipo de simulador utiliza mancais esféricos, sendo uma das plataformas mais utilizadas na pesquisa de dinâmica e controle de atitude (SCHWARTZ; PECK; HALL, 2003), pois podem prover movi-

mento rotacional sem restrições em um dos seus eixos. Esta classe de simulador é utilizada onde o controle em três eixos é necessário, como o controle de atitude. O simulador esférico ideal permitiria rotação de 360° em todos os seus 3 eixos, ou seja, arfagem, rolagem e guinada. Por limitações devido à montagem destes simuladores eles possuem restrições físicas em um ou todos os eixos de rotação. Restrições estas devido a interferência do pedestal sobre o qual a plataforma está montada, como pode ser visto na figura 2.5.

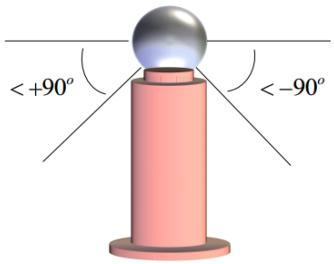


Figura 2.5: Restrições de uma plataforma de testes.
Fonte: adaptado de Snider (2010)

Ainda dentro da classe de sistemas rotacionais, existe um subconjunto que pode ser classificado em: mesa giratória (*tabletop*), guarda chuva (*umbrella*) ou halteres (*dumbbell*).

2.1.2.1 Mesa Giratória

As plataformas do tipo mesa giratória são montadas diretamente na face plana de um rolamento semiesférico e todos os outros componentes como, sensores, atuadores, computadores de bordo, são montados sobre esta placa. Este tipo de plataforma permite movimento de rotação de 360° em torno do eixo de guinada e restrições nos eixos de arfagem e rolagem em ângulos menores que +-90°. A figura 2.6 exemplifica a montagem deste tipo de plataforma.

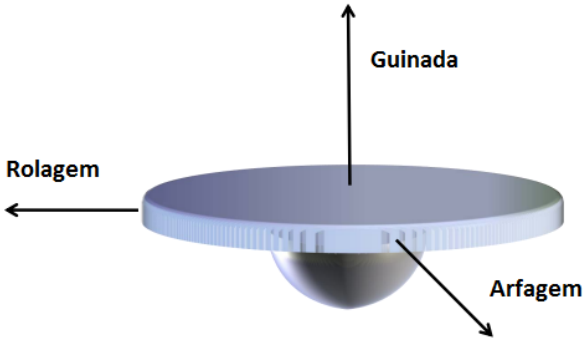


Figura 2.6: Plataforma do tipo mesa giratória.
Fonte: adaptado de (SNIDER, 2010)

Os autores Prado et al. (2005) da Universidade Nacional do México (Unam) mostram o processo do desenvolvimento de uma plataforma do tipo mesa giratória, figura 2.7. Esta plataforma possui como atuadores rodas de reação e barras magnéticas para propósitos de dessaturação das rodas de reação, problema que ocorre quando a roda de reação atinge o limite de rotação e não consegue mais atuar. O objetivo final deste estudo é testar sensores de atitude, atuadores e algoritmos de controle de maneira experimental.

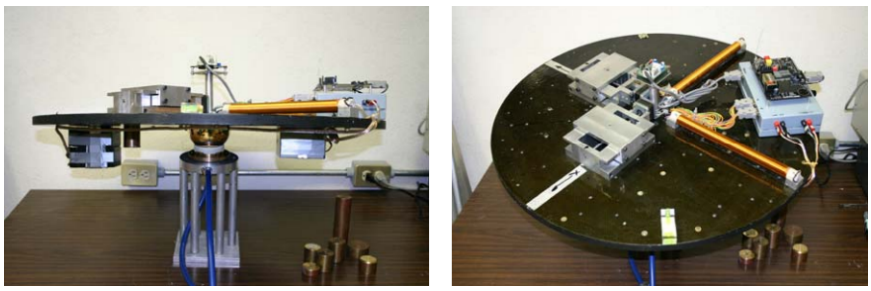


Figura 2.7: Plataforma Unam.
Fonte: Prado et al. (2005)

No Instituto Nacional de Pesquisas Espaciais (INPE) uma plataforma deste tipo foi desenvolvida a fim de simular o controle de atitude de satélites artificiais empregando jatos de gás como atuadores. A plataforma é mostrada na figura 2.8. O objetivo inicial da plataforma era testar os amortecedores de nutação do satélite brasileiro SCD2.

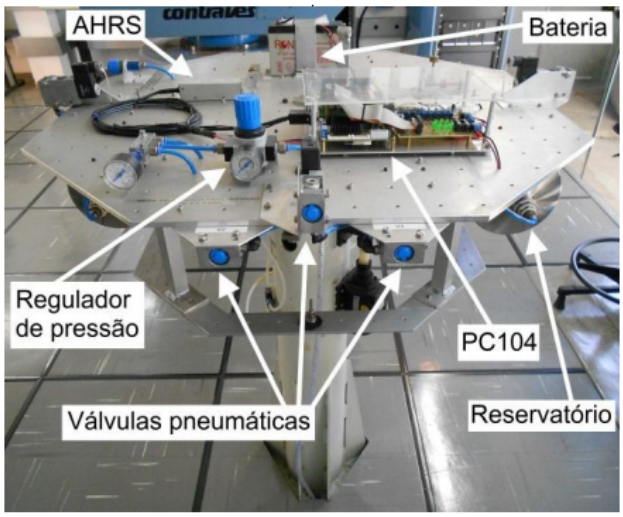


Figura 2.8: Plataforma desenvolvida pelo INPE.
Fonte: (CAMPESATO et al., 2013)

Na figura 2.9 está a plataforma *SimSat II* utilizada por Snider (2010) para testes de controle de atitude em 3 eixos, utilizando um controlador PID. Esta plataforma utiliza rodas de reação como atuadores.

2.1.2.2 Modelo Guarda chuva

As plataformas que seguem esta configuração possuem uma haste de extensão que se sobressai do topo de um rolamento completamente esférico. Sobre esta haste está montada a mesa juntamente com todos os equipamentos necessários para realização de experimento. Esta plataforma reduz drasticamente as interferências no espaço de rotação da plataforma. De maneira semelhante as plataformas do tipo mesa giratória, esta possui movimento de rotação de 360° em torno do eixo de guinada, porém uma amplitude de rotação maior em relação aos eixos de arfagem e rolagem. A figura 2.10 demonstra sua configuração física.

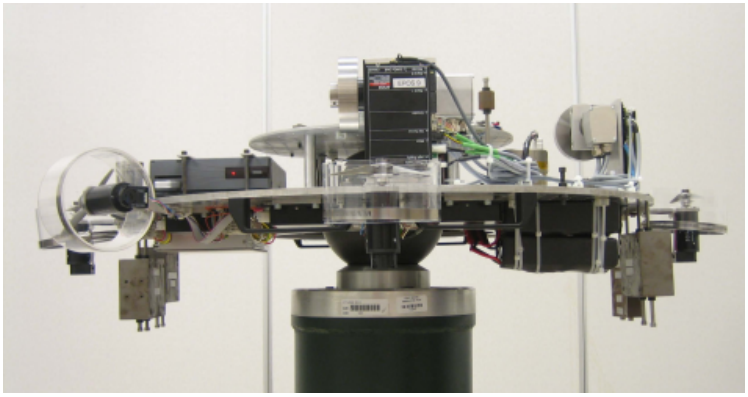


Figura 2.9: Plataforma SimSat II.
Fonte: (SNIDER, 2010)

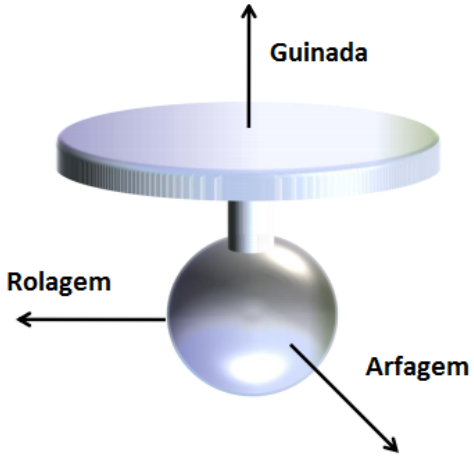


Figura 2.10: Plataforma do tipo guarda chuva.
Fonte: adaptado de (SNIDER, 2010)

Uma das primeiras plataformas deste tipo foi desenvolvida pela Força Aérea dos Estados Unidos e se chamava ASTREX *(Advanced Space Structure Technology Research Experiments)*. O objetivo desta plataforma era testes de integração de componentes de satélite e validação de leis de controle. Esta plataforma era suportada por um mancal esférico de 48cm de diâmetro podendo suportar cargas de até 6600 kg. Utilizava jatos de gás para movimentação em três eixos e rodas de reação para supressão de vibração e pontaria de precisão (DAS et al., 1990) . Outra plataforma que segue esta configuração é mostrada na figura 2.11. Foi desenvolvida pela *Honeywell* e é chamada de MCS/LOS (*Momentum Control System and Line of Sight*). Seu objetivo é o teste de leis de controle e as interações entre o uso de atuadores e a resposta dinâmica da estrutura do satélite.

2.1.2.3 Modelo halteres

Esta plataforma segue a configuração de duas hastes que saem de um rolamento completamente esférico. Nesta plataforma é possível ter movimentos de 360° em torno do eixo de guinada e de rolagem, como pode ser visto na figura 2.12.

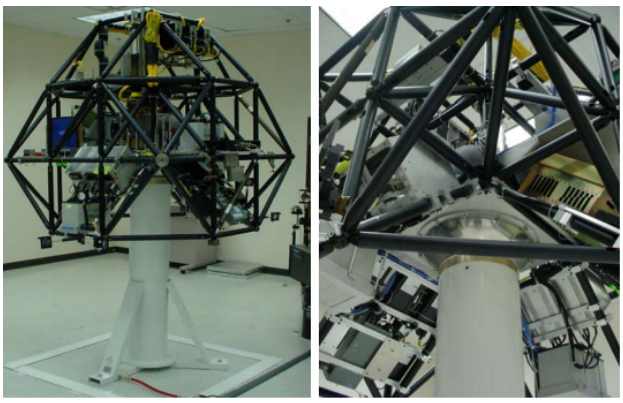


Figura 2.11: Plataforma MCS/LOS.
Fonte: (PECK et al., 2003)

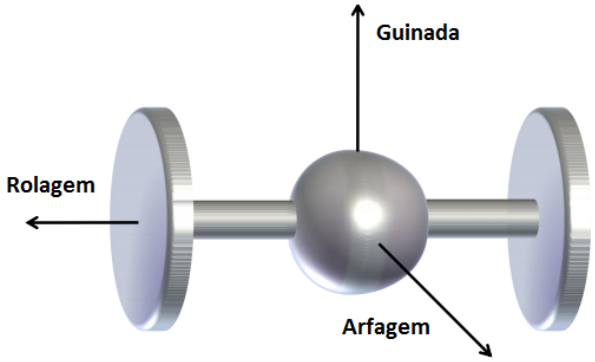


Figura 2.12: Plataforma do tipo halteres.
Fonte: adaptado de Snider (2010)

Na figura 2.13, é mostrada a plataforma *SIMSAT I*, desenvolvida pelo Instituto de Tecnologia da Força Aérea dos Estados Unidos. Esta plataforma é livre pra girar 360° em torno dos eixos de guinada e rolagem e 30° no eixo de arfagem. Ela conta com todos os sistemas de determinação e controle de atitude, sendo principalmente utilizada para práticas de ensino.

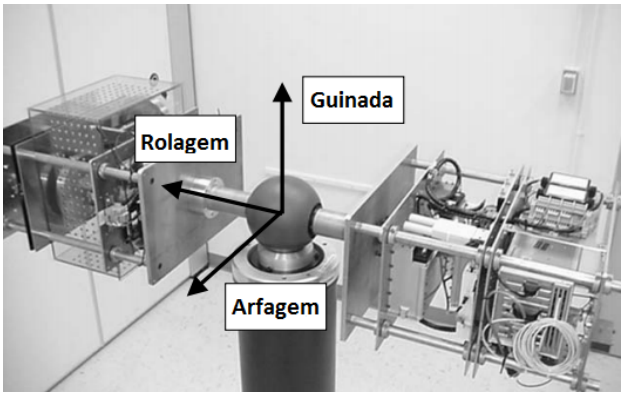


Figura 2.13: Plataforma SIMSAT I
Fonte: (TRAGESSER; AGNES; FULTON, 2002)

Outra plataforma de mesma configuração está em funcionamento no INPE, figura 2.14. Esta plataforma conta com três rodas de reação, três bobinas magnéticas, um GPS e um magnetômetro de três

eixos.

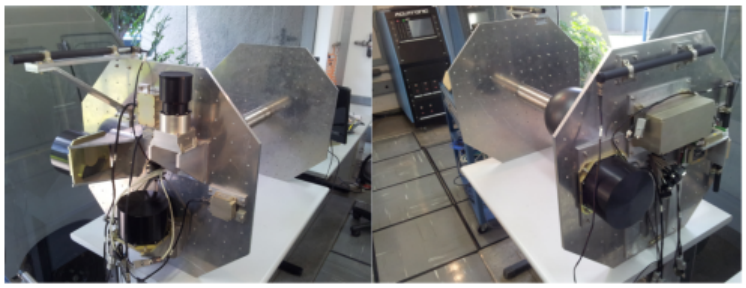


Figura 2.14: Plataforma INPE halteres.
Fonte: Oliveira, Kuga e Carrara (2015)

2.1.3 Sistemas Combinados

A última classe de sistemas rotacionais são os sistemas combinados. Este tipo de simulador é o mais complexo por que unem características dos simuladores planares e rotacionais. Para isto, possuem mancais aerostáticos planares e esféricos numa mesma plataforma. Podem ser úteis para testes de operações de controle de atitude, *docking* e *rendezvous*. Estes sistemas podem ter de 5 a 6 graus de liberdade. Em Gallardo e Bevilacqua (2011) uma plataforma deste tipo é apresentada. Este sistema se destaca dos demais por possuir 6 graus de liberdade. Pela figura 2.15, percebe-se que a plataforma é dividida em estágio superior e inferior. O estágio inferior é responsável por movimento de translação através de um mancal aerostático plano e o estágio superior possui um mancal aerostático esférico, para simulação de atitude, da mesma maneira que uma plataforma do tipo mesa giratória.

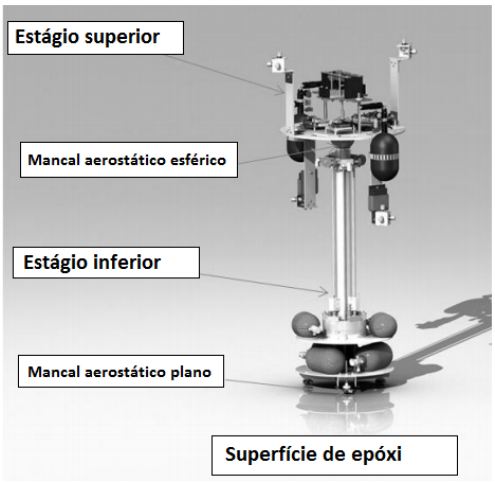


Figura 2.15: Plataforma com 6 graus de liberdade.
Fonte: adaptado de Gallardo e Bevilacqua (2011)

Em Jian et al. (2009) foi desenvolvida uma plataforma com 5 graus de liberdade com o intuito de simular formação de voo autônoma e manobra orbital. Três graus de liberdade rotacional são fornecidos pelo mancal esférico a ar e dois graus de liberdade translacional pelos mancais lineares na base da plataforma 2.16. Esta plataforma utiliza jatos de gás frio como atuadores.

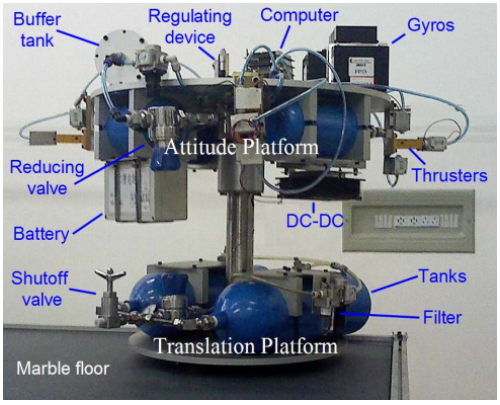


Figura 2.16: Plataforma com 5 graus de liberdade.
Fonte: Jian et al. (2009)

2.2 Metodologia de validação de controladores embarcados

A importância da validação de controladores embarcados para ACS diz respeito à possibilidade de detectar e reduzir possíveis erros de projeto que possam impactar diretamente na missão do satélite. Outro fator é que um dos principais aspectos que tem faltado na área de controle de atitude de satélites é a validação experimental dos resultados teóricos (KIM et al., 2003). As dificuldades em testar e validar o ACS em um ambiente em tempo real e em condições próximas daquelas encontradas no espaço, devem-se a impossibilidade de se utilizar o satélite real ou recriar condições do espaço em laboratório. Logo, torna-se necessário utilizar alguma metodologia que viabilize a execução destes testes em terra. Uma das metodologias de desenvolvimento bastante utilizada na área aeroespacial, de software e indústria automotiva (KELEMENOVÁ et al., 2013) para validação de *software* embarcado é o projeto baseado em modelos *Model-Based Design (MDB)*. Esta metodologia está centrada no modelo do processo e fornece uma abordagem matemática e visual para o desenvolvimento de sistemas complexos (MATHWORKS, 2018b). O *V-cycle* por sua vez é um fluxo de trabalho utilizado para comunicação entre as diferentes partes do MDB, já tendo sido aplicada com sucesso na indústria aeroespacial (TOMAN; KERLÍN; SINGULE, 2011), (SILVA et al., 2014), para validação de *software* embarcado. O ciclo de trabalho em V conjuntamente com a metodologia MDB é mostrado na figura 2.17.

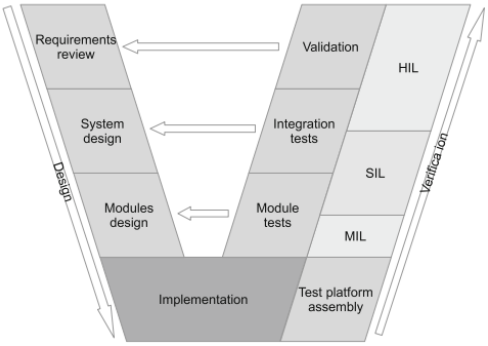


Figura 2.17: Modelo de desenvolvimento em V.
Fonte: Silva et al. (2014)

O ciclo é dividido em duas partes. O lado esquerdo corresponde às definições de projeto e o lado

direito, às etapas de validação. Para cada etapa de *design* existe uma etapa correspondente de validação, característica essa que ajuda na identificação e isolamento de erros encontrados, facilitando a correção dos mesmos. Na etapa de requerimentos, as principais características que o sistema deve possuir são identificadas e documentadas. A etapa de projeto de sistema é onde o modelo abstrato do processo será construído para em seguida ser dividido em módulos, cada um com suas características de entrada e saída. A etapa de projeto de módulos corresponde ao desenvolvimento dos módulos da etapa anterior. As ferramentas mais utilizadas para este propósito são *LabVIEW* e *Matlab/Simulink*. Por praticidade e disponibilidade este trabalho optou por utilizar o *Matlab/Simulink* durante todo o processo de desenvolvimento dos controladores.

Terminada a etapa de *design* passa-se a etapa de implementação e montagem das plataformas de testes necessárias para as etapas de verificação. As etapas são: *Model-in-the-loop (MIL)*, *Software-in-the-loop (SIL)*, *Processor-in-the-loop (PIL)* e *Hardware-in-the-loop (HIL)*. As próximas subseções trarão uma descrição mais detalha de cada um destes testes de validação.

2.2.1 *Model-in-the-loop (MIL)*

A primeira etapa de verificação consiste em simular o controlador desenvolvido, conjuntamente com o modelo do processo no mesmo ambiente de simulação (MATINNEJAD et al., 2015). Neste ponto, tanto o modelo do processo quanto o controlador são descritos utilizando-se a mesma linguagem de blocos no *Simulink*. A figura 2.18 mostra que toda a malha de controle é desenvolvida dentro do *Matlab/Simulink*.

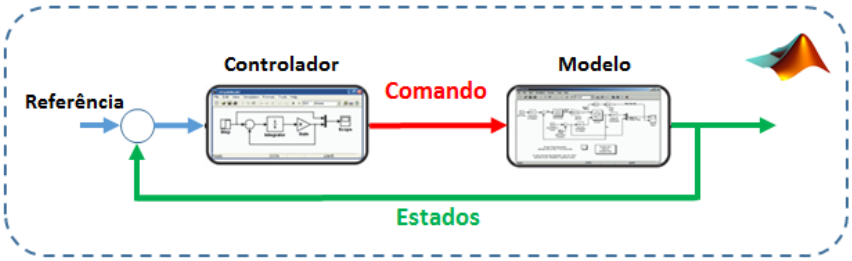


Figura 2.18: Arquitetura de verificação: Model-in-the-loop (MIL).

Aqui vários cenários diferentes podem ser testados, variando-se os parâmetros do modelo e/ou do controlador até que se chegue num conjunto de valores que satisfaçam os requisitos de projeto. Logo o objetivo desta simulação é garantir que a dinâmica do conjunto controlador e modelo do processo esteja de acordo com os requerimentos criados na fase de projeto do *V-cycle*.

2.2.2 *Software-in-the-loop (SIL)*

É na etapa SIL onde ocorre a geração do código do controlador que foi testado e validado anteriormente na etapa de MIL (MATINNEJAD et al., 2015). O desenvolvimento do mesmo pode ser feito através de dois métodos. O primeiro é pela geração automática a partir da ferramenta *Code Generator* presente no *Matlab/Simulink*. Esta ferramenta converte o controlador que foi desenvolvido em linguagem de *script* ou por meio de blocos do *Simulink*, para uma linguagem e um *hardware* específicos, escolhidos pelo projetista. Pode-se citar como vantagens deste processo: redução e/ou eliminação de erros que poderiam

ser introduzidos por um programador e redução no tempo de desenvolvimento. Porém, o projetista perde autonomia sobre o código gerado, não possuindo a liberdade de alterar o código fonte ou permanecendo restrito ao uso de bibliotecas proprietárias. Outro fator importante é que a rastreabilidade e controle do código é extremamente reduzida (SILVA, 2017). O segundo método é o desenvolvimento manual. Este é um método mais lento e exige conhecimento das especificidades do *hardware* em que o controlador será embarcado. Porém, o projetista não se restringe ao uso de bibliotecas proprietárias, podendo utilizar bibliotecas *open-source* e pode inserir modificações e/ou atualizar o código conforme surjam novas necessidades de projeto. Este é o método utilizado neste trabalho.

As simulações em SIL ocorrem com o código do controlador e modelo do processo ainda no *Matlab*. O objetivo é avaliar se o código desenvolvido funciona corretamente, i.e., fornece os mesmos resultados das simulações realizadas em MIL.

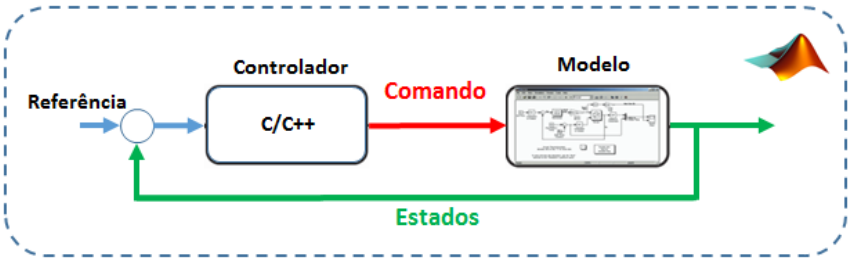


Figura 2.19: Arquitetura de verificação: Software-in-the-loop (SIL).

Na figura 2.19 está indicado que o código do controlador está em C/C++. Apesar destas serem as linguagens mais utilizadas para programação de *software* embarcado (DUBOIS, 2018), existem outras possibilidades como as linguagens de descrição de *hardware Verilog* ou *VHDL*. O tipo de teste feito na etapa SIL é a maneira mais barata de se testar o código produzido sem nenhum *hardware* envolvido (TULPULE et al., 2017). Caso as simulações em SIL apresente um resultado satisfatório, prossegue-se para a etapa de PIL.

2.2.3 *Processor-in-the-loop (PIL)*

A simulação em PIL é um teste onde o código desenvolvido na etapa SIL é compilado e embarcado em um processado dedicado, por exemplo, *Field-Programmable gate array (FPGA)*, *Digital Signal Processor (DSP)* ou microprocessadores, sendo esta a primeira etapa para testar o controlador de maneira embarcada (JIANG et al., 2004). Apesar da simulação PIL ser utilizada para avaliação de controladores, ela também pode ser útil para testar partes complexas de um sistema, onde essas partes podem ser embarcadas em plataformas digitais (MINA et al., 2016). A figura 2.20 mostra a arquitetura de testes em PIL. Percebe-se que o modelo do processo está no ambiente de simulação do *Matlab* e o controlador embarcado em um *hardware* específico. Estas partes se comunicam geralmente por um *link* de comunicação serial, como *ethernet* (TULPULE et al., 2017), para troca de informações de estado e sinal de controle.

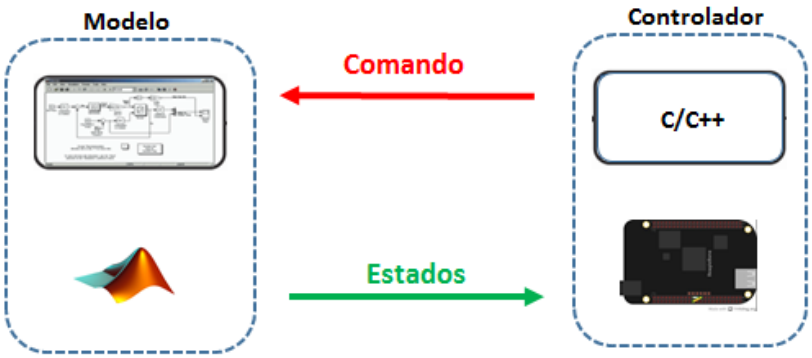


Figura 2.20: Arquitetura de verificação: Processor-in-the-loop (PIL).

Neste tipo de simulação é possível verificar erros de compatibilidade de código com o *hardware* escolhido, detectar erros de alocação de memória e capacidade de processamento. Um fato importante da simulação em PIL é que ela não ocorre em tempo real. Por fim, os resultados desta etapa serão comparados com aqueles obtidos na etapa de SIL. Caso haja concordância entre as simulações o controlador está pronto para ser levado a etapa de HIL.

2.2.4 *Hardware-in-the-loop (HIL)*

Esta etapa consiste em verificar a integração de *hardware* e *software* em um ambiente mais realístico (MATINNEJAD et al., 2015). O modelo do processo é simulado através de *hardware* ou virtualmente através de um computador rodando um sistema operacional em tempo real (*Real time Operational System - (RTOS)*), existindo ainda a possibilidade de simular sensores e atuadores. O controlador por sua vez continua embarcado no *hardware* especificado na etapa PIL.

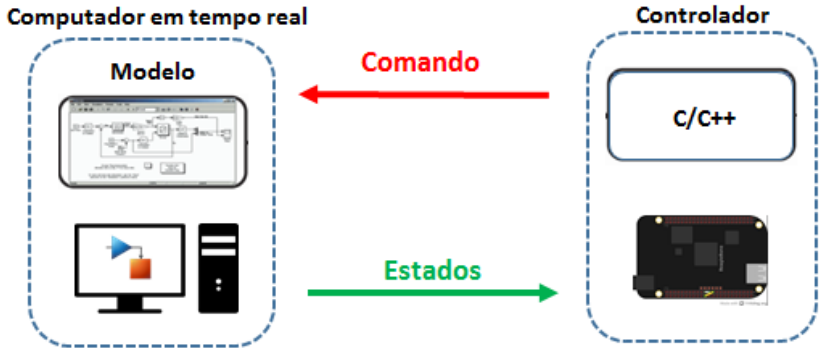


Figura 2.21: Arquitetura de verificação: Hardware-in-the-loop (HIL).

Podem-se citar exemplos de empresas especialistas no fornecimento e desenvolvimento de soluções para testes HIL como: *dSPACE*, *speedgoat* e *National Instruments (NI)*.

Apesar de ser uma das várias etapas de validação da metodologia MDB, ela merece um detalhamento maior pois é a etapa mais próxima da realidade e também a mais cara (MATINNEJAD et al., 2015), devido aos equipamentos envolvidos. Quando não se pode utilizar o modelo físico do processo, o HIL traz a possibilidade de testes entre componentes reais e componentes simulados. O princípio geral de uma

arquitetura HIL é aquela mostrada na figura 2.22.

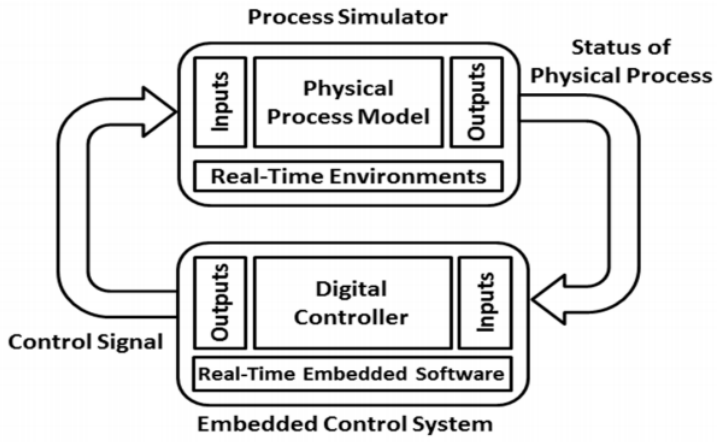


Figura 2.22: Princípio geral de uma arquitetura HIL.
Fonte: adaptado de Short e Abugchem (2017)

A arquitetura faz a separação entre o simulador do modelo físico e o sistema de controle embarcado. O *hardware* responsável pela simulação do modelo físico do processo são chamados de *Real-time target machines*, máquinas capazes de realizar simulações em tempo real e contam com módulos de entrada e saída para comunicação com periféricos externos. O controlador por sua vez pode ser embarcado em *hardware* com poder de processamento em tempo real. No tocante aos testes em HIL aplicados a satélites Bayat (2015) descreve a existência de três tipos de arquiteturas para validação do ACS, elas são:

- Tipo 1: Realiza apenas simulações numéricas, onde todos os elementos de sensoriamento, atuação e estimação são simulados. Geralmente são dois computadores, um responsável por simular o modelo do satélite e o outro responsável pelo controle de atitude se comunicando em tempo real através de uma rede.
- Tipo 2: Esta arquitetura utiliza as plataformas de mancal aerostático apresentadas na seção 2.1. Sendo dividida em dois tipos. O primeiro é somente a plataforma com todos os sensores, atuadores e o próprio controlador sobre a plataforma. No segundo tipo a plataforma conta apenas com sensores, sendo que os atuadores e sistema de controle estão funcionando em um computador a parte que é responsável por movimentar a plataforma.
- Tipo 3: Esta arquitetura conta com três elementos. Um computador responsável por simular o movimento de atitude do satélite, um *hardware* específico para implementação do controlador e por fim uma interface de comunicação entre as partes envolvidas.

Historicamente a técnica HIL foi desenvolvida inicialmente para aplicações militares, indústria espacial (ĆOSIĆ et al., 1999), (ISERMANN; SCHAFFNIT; SINSEL, 1999) e controle de reatores nucleares (LEDIN, 1999). Sua popularidade cresceu e já foi aplicado a validação de ACS em satélites (RODRIGUES et al., 2013), (SLAFER, 1993). Para este tipo de aplicação em específico, o HIL é útil pois reduz o tempo de desenvolvimento dos satélites (CORPINO; STESINA, 2014), permite otimizar o desempenho do sistema (VILATHGAMUWA; YUE; TSENG, 2004) e possibilita o teste de diferentes tipos de cenários, inclusive aqueles considerados extremos, que não seriam possíveis executar no modelo físico real (NABI

et al., 2004). Por não envolver o uso do modelo real do processo também garante a segurança das pessoas que estão envolvidas na execução dos testes. O uso de simulações HIL aplicados ao domínio de satélites tem sido extensivamente descrito na literatura.

Em Ptak e Foundy (1998) é discutido que o uso de simulações do tipo HIL ajudam a aumentar a confiabilidade dos ACS. Neste trabalho os autores desenvolveram um ambiente onde é possível projetar, desenvolver, verificar e validar os sistemas de controle de atitude, sensores e atuadores para missões da Agência Espacial Canadense. Eles trabalham mais especificamente com a validação do hardware de voo do satélite *Odin*. Os testes ocorrem pela integração do modelo dinâmico do satélite simulado virtualmente por um computador específico e o uso de sensores reais. O sistema de controle por sua vez está embarcado em um computador que roda o sistema operacional em tempo real *RT-Linux*.

Em Ure, Kaya e Inalhan (2011) as propriedades operacionais relacionadas ao ADCS do nano satélite *ITU PSAT II* são testadas através das abordagens de SIL e HIL. O trabalho utilizou uma mesa de mancal aerostático para simular a dinâmica do satélite e uma bobina de *Helmholtz* capaz de simular o campo terrestre magnético. Em Corpino e Stesina (2014) é discutido o desenvolvimento de um *framework* para validação de controladores para satélites. Neste trabalho simulações do tipo HIL foram utilizadas para testar e validar os diferentes subsistemas presentes no *Educationl SaTellite @ politecnico di toRino (e-st@ar) CubeSat*, um satélite com propósitos educacionais. Outro trabalho relacionado a HIL e *CubeSat* pode ser encontrado em Tapsawat, Sangpet e Kuntanapreeda (2018).

Um outro simulador HIL para formação de voo autônomo foi desenvolvido por Park et al. (2013). O sistema consiste em um simulador de órbita e outro de atitude. O primeiro é utilizado para propagação de órbita e o segundo para determinação de atitude e controle, para isto utiliza-se uma mesa de mancal aerostático do tipo mesa giratória, que se movimenta através da ação de rodas de reação. Para controle de órbita é utilizado um controlador SDRE e para controle de atitude é utilizado um controlado PD.

Uma arquitetura HIL de baixo custo é desenvolvida pela Universidade de *Stuttgart*. O objetivo desta plataforma é testar o software do computador de bordo para pequenos satélites. A motivação deste trabalho foi diminuir os custos dos equipamentos envolvidos na plataforma através do uso de ferramentas *open-source*, simulação de sensores e atuadores, buscando manter os padrões exigidos pela indústria (FRITZ et al., 2015). As plataformas do tipo HIL também tem se mostrado úteis na validação de atuadores utilizados em satélites. Em Izadi, Abedi e Bolandi (2016) é desenvolvida uma arquitetura de testes HIL para comparação entre o modelo teóricos de um roda de reação e o equipamento real. Os autores Min, Guoqiang e Yudong (2012) analisaram o controle de atitude de um micro satélite através do uso de *thrusters*, utilizando uma plataforma do tipo mesa giratória.

Os parágrafos anteriores deixam claro a utilidade das simulações HIL como meio de validação de controladores embarcados, seja para satélites ou equipamentos utilizados pelo mesmo.

2.3 Controladores para controle de atitude

A área relacionada à pesquisa de sistemas de controle de atitude de satélite tem se tornado o foco (LI et al., 2014) de pesquisas recentes. Abordagens clássicas foram desenvolvidas como PID (AURET, 2012)

e o Regulador Quadrático Linear (LQR) (JR; SANTANA; MARTINS-FILHO, 2007) para estabilizar um satélite multi missões de três eixos. A abordagem através do LQR faz uso do modelo linear do satélite, o que restringe a gama de aplicações deste controlador.

No que diz respeito ao uso em modelo não linear, existem técnicas como a *State-Dependent Riccation Equation (SDRE)* (SOUZA; GONZALES, 2012), (FRENCH, 2003) aplicada a plataformas de testes de satélites para controle de atitude. O SDRE é extensão da técnica LQR para sistemas não lineares (KUMAR; JEROME; RAAJA, 2014). O LQR utiliza o modelo linear do sistema para cálculo do ganho ótimo e o mantém constante durante toda simulação. O SDRE por sua vez utiliza as informações dos estados do modelo e faz o cálculo do ganho ótimo para aquele instante específico recalculando esse ganho à medida que a simulação continua.

Já pra satélites com estruturas flexíveis, por exemplo, painéis solares e antenas de comunicação, algoritmos do tipo H_∞ (CUBILLOS; SOUZA, 2009), controle robusto (LI et al., 2014) e controle *fuzzy* adaptativo (GUAN; LIU; LIU, 2005), (XIAO; HU; MA, 2011) foram desenvolvidas. É relevante também citar estratégias no domínio da frequência (NUDEHI et al., 2008) para controle de um satélite considerando distúrbios e incertezas de um modelo não linear *Multiple-input and multiple-output (MIMO)*.

Os satélites como qualquer sistema físico real, possuem restrições, sejam elas estruturais, de sensores ou atuadores. As estratégias de controle descritas anteriormente não possuem na sua estrutura nenhum tipo de característica que considere explicitamente estas restrições. Estas são obtidas por meios indiretos. Como por exemplo no caso do LQR através das matrizes de ponderação para cálculo do ganho ótimo e no PID através da sintonia dos parâmetros proporcional(K_p), integral(K_i) e derivativo(K_d). É possível afirmar que uma boa estratégia de controle deve levar em consideração as restrições do sistema sobre o controle, sendo este um fato ainda mais problemático em sistemas aeroespaciais devido aos altos custo envolvidos e a não possibilidade de reparação do erro quando o satélite é enviado ao espaço.

2.4 Controle preditivo aplicado ao controle de atitude de satélites

Como explicado anteriormente, uma das grandes vantagens do controle preditivo baseado em modelos é sua capacidade de gerenciar restrições de estado, comando e variação de comando. Esta estratégia de controle já foi considerada em aplicações relacionadas ao controle de atitude de satélites.

Uma das primeiras contribuições nesta área foi feita por Hegrenæs, Gravdahl e Tøndel (2005), onde foi desenvolvido um MPC explícito, ou seja, a solução do problema de otimização é calculada *offline*, para o modelo linear do *European Student Earth (ESEO)* micro satélite. Este satélite utiliza como atuadores *thrustes* e rodas de reação.

Em Chen e Wu (2010) é desenvolvida uma formulação do MPC através de funções de *Laguerre*, para o problema de controle de atitude de um *CubeSat* que utiliza barras de torque magnético como atuadores. A estratégia leva em consideração os limites de saturação do atuador e restrições relacionadas ao campo magnético terrestre.

Outra estratégia foi a *Distributed Model Predictive Control (DMPC)* para o controle de atitude de um nano satélite proposta (XING; LOW; PHAM, 2012). O satélite utilizava barras magnéticas e rodas de

reação como atuadores. Esta estratégia de controle descentralizada tem como objetivo reduzir o tempo de cálculo da estratégia MPC. Pirouzmand e Ghahramani (2013) desenvolveram um controlador MPC robusto para o controle de atitude de um satélite com três graus de liberdade. O objetivo do controle robusto neste caso é lidar com as incertezas na matriz de inércia do satélite, distúrbios externos e restrições na variável de comando do atuador.

Outra estratégia robusta é desenvolvida por Li et al. (2014), desta vez para um satélite com estrutura flexível. O MPC também foi utilizado para controle de atitude de um satélite geoestacionário com estruturas flexíveis e movimento em apenas um eixo de rotação (TAYYEBTAHER; ESMAEILZADEH, 2017), recuperação de falhas de um *CubeSat* (FRANCHI et al., 2018) e operações de *rendezvous* autônomo (LI et al., 2017).

Apesar da abrangência das aplicações anteriores com o MPC, os autores não fizeram nenhuma avaliação do algoritmo de controle embarcado em *hardware*, todos os resultados foram obtidos através de simulações no *Matlab/Simulink*. Uma vez que os sistemas *ADCS* são sistemas embarcados críticos em tempo real (BAYAT, 2015) é importante testar e validar o controlador em condições mais próximas possíveis da realidade de voo de um satélite. Dentro deste contexto, faz se interessante considerar a validação do *software* embarcado utilizado o *Model-Based Design* e o *V-Cycle*.

Um dos aspectos que tem faltado na literatura são testes HIL e verificação de algoritmos de controle preditivo embarcado em hardware para o controle de atitude de satélites. Resultados recentes validaram a estratégia MPC em hardware mas para casos de *rendezvous* e operações de aproximação entre espaçonaves (VIRGILI-LLOP et al., 2016), formação de voo (VALMORBIDA, 2014) e MPC embarcado em FPGA com validação em HIL para operações de *rendezvous* (GILZ et al., 2017). Em Goodyear et al. (2015) o MPC foi embarcado no computador de baixo custo *RaspberryPi*, com o objetivo de auxiliar no controle de desvio de obstáculos de uma espaçonave, mostrando a possibilidade de se utilizar o MPC em computadores com limitada capacidade computacional.

Capítulo 3

Controle Preditivo Baseado em Modelo

Nesta seção será apresentado um breve histórico do controle preditivo, quais foram suas origens e como ele evoluiu com o passar do tempo. Será também abordado como a estratégia de controle funciona, sua formulação teórica, aplicações, vantagens e desvantagens.

3.1 Introdução

O termo preditivo da sigla *(MPC)* advém do fato de que este controlador utiliza o modelo do processo a ser controlado a fim de predizer a resposta futura do mesmo sob o chamado horizonte de predição, um conjunto finito de amostras futuras (CURRIE, 2014). Esta é uma técnica avançada de controle que calcula, a cada período de amostragem, uma sequência ótima de comando através da minimização de uma determinada função custo que expressa os objetivos de controle (MAYNE et al., 2000).

A primeira descrição satisfatória do MPC foi apresentada por Richalet et al. (1976), em um algoritmo baseado em métodos heurísticos, que foi chamado posteriormente de *Model Predictive Heuristic Control (MPHC)* (RICHALET et al., 1978). O software comercializado que implementava este algoritmo chamava-se *Identification and Command (IDCOM)*. Posteriormente dois pesquisadores, *Cutler* e *Ramaker*, que trabalhavam na companhia petrolífera *Shell Oil* desenvolveram uma outra abordagem do tipo MPC, independente daquela desenvolvida por *Richalet*. Essa nova abordagem foi chamada de *Dynamic Matrix Control (DMC)* (CUTLER; RAMAKER, 1980). Estes algoritmos ainda não eram capazes de lidar com restrições e utilizavam o método dos mínimos quadrados para calcular o próximo sinal de controle (CURRIE, 2014). Com o intuito de incluir o tratamento das restrições inerentes ao processo, outro algoritmo foi desenvolvido por Cutler, Morshedi e Haydel (1983), chamado de *Quadratic DMC (QDMC)*. Este, por sua vez, introduziu o conceito de programação quadrática e é visto como o predecessor dos algoritmos modernos MPC (CURRIE, 2014).

A existência de uma problema de otimização dentro da formulação do MPC fez com que, à época do seu surgimento, ele fosse principalmente aplicado a processos de dinâmica lenta, como refinarias e processos químicos. Este processos permitiam que os computadores da época, com limitações de quantidade de memória e velocidade de processamento, realizassem os cálculos da lei de controle dentro dos tempos exigidos. Com a melhoria na velocidade dos processadores ao longo dos anos, foi possível au-

mentar a gama de processos onde o MPC poderia ser utilizado. A título de comparação em 1978, ano de lançamento do chip 8086 de 16 *bits* da *Intel*, este tinha um *clock speed* máximo de *10MHz* (AUTHOR, 2018). Atualmente é possível encontrar microprocessadores de baixo custo do tipo *AM335x* existentes na *BeagleboneBlack* que podem alcançar *1 GHz* (DAIGNEAU; ADVENA, 2018). A figura 3.1 mostra onde se concentram as aplicações do MPC nos dias atuais.

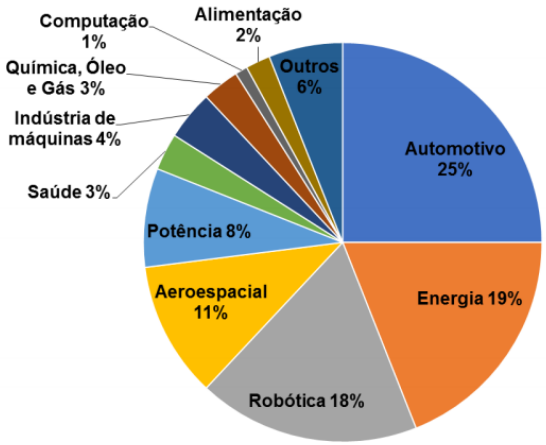


Figura 3.1: Aplicações MPC por segmentos da indústria.
Fonte: Adaptado de Ferreau et al. (2016)

Hoje o controle preditivo baseado em modelos é uma das técnicas avançadas mais promissoras para lidar com sistemas multivariáveis com restrições (SIMON, 2014). O MPC tornou-se bastante popular principalmente em aplicações industriais devido a sua flexibilidade, relativa facilidade de implementação (MAYNE et al., 2000), lidando com modelos multivariáveis e não lineares (ALAMIR, 2013). A maneira como essa estratégia funciona está ilustrada na figura 3.2.

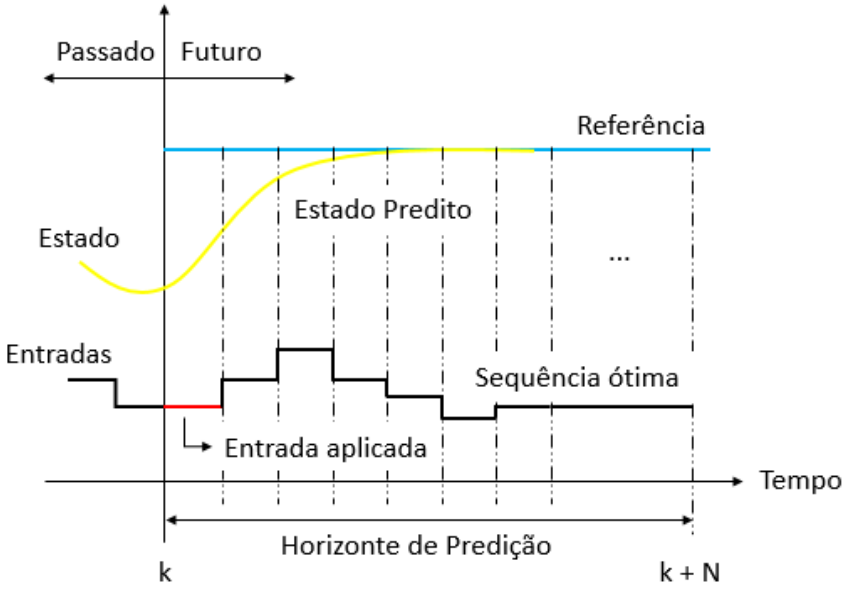


Figura 3.2: Estratégia MPC.
Fonte: Adaptado de Murilo (2009).

O algoritmo utiliza o modelo do sistema para predizer o comportamento futuro do mesmo a partir das informações dos estados do sistema, fornecidas por meio de sensores ou estimação, no instante k, e da referência passada até o instante $k+N$. O valor N é chamado de horizonte de predição. A função custo por sua vez leva em consideração as restrições de estado, comando e variação de comando existentes no processo. A solução, ou minimização dessa função custo, gera uma sequência ótima de comandos sob o horizonte de predição N, porém apenas o primeiro valor dessa sequência é aplicado, indicado em vermelho na figura 3.2. O objetivo de se aplicar apenas o primeiro sinal de controle da sequência gerada é melhorar a capacidade de rejeitar as pertubações (LING et al., 2010), dando mais robustez ao sistema. O processo de otimização por sua vez é então repetido em cada instante de amostragem subsequente e o horizonte de predição deslocado para frente.

3.1.1 Estrutura do controlador MPC

A arquitetura do controlador MPC é mostrada na figura 3.3. Os principais componentes desta arquitetura são o modelo e o otimizador. Nesta arquitetura, o modelo é utilizado para predizer a resposta futura do processo a partir das informações de entrada e estados atuais. O modelo utilizado para predição pode ser linear ou não-linear.

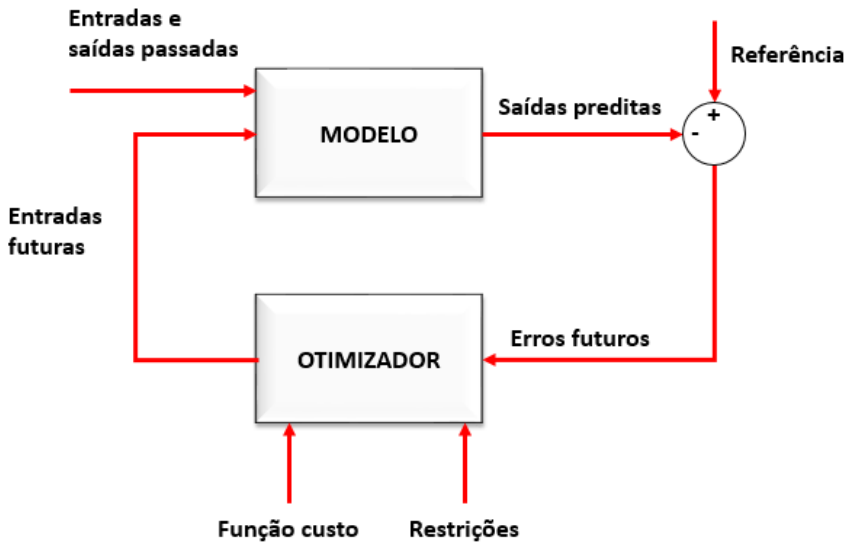


Figura 3.3: Estrutura do controlador MPC.
Fonte: Adaptado de Camacho e Alba (2007)

Segundo Camacho e Alba (2007) e Wang (2009) as representação dos modelos utilizados para o desenvolvimento da estratégia de controle preditivo podem ser de três tipos:

1. Resposta Truncada ao Impulso (TIR) : A obtenção deste tipo de modelo é mais simples, uma vez que é necessário somente as medições da saída do processo quando este é excitado com uma entrada impulso, porém somente sistemas estáveis em malha aberta podem ser descritos dessa maneira.
2. Função de transferência : Utilizada principalmente no meio acadêmico e a derivação do controlador preditivo é mais difícil.

3. Modelo em espaço de estados : A obtenção das equações do MPC é mais fácil para sistemas MIMO.

Neste trabalho a abordagem para desenvolvimento das equações do MPC será aquela proposta por Alamir (2013) baseada em modelo em espaço de estados. A motivação por trás desta escolha deve-se ao fato de que esta representação é a mais utilizada para sistemas multivariáveis (WANG, 2009) e devido à facilidade de se derivar as equações finais do controlador (CAMACHO; ALBA, 2007).

No que se refere à etapa de otimização, é importante destacar que esta é uma das desvantagens do controle preditivo. O motivo deve-se ao fato de que a solução de um problema de otimização deve ser encontrado, em um determinado período de amostragem. Isto restringe a aplicabilidade do MPC para sistemas pequenos e/ou lentos (BEMPORAD et al., 2000), onde as constantes de tempo do sistema variam de alguns milissegundos para alguns segundos. Para sistemas rápidos onde as contantes de tempo são menores este problema é mais desafiador (RAZIEI; JIANG, 2016). O tamanho e a complexidade do problema de otimização por sua vez está diretamente relacionado a quantidade de variáveis do modelo e horizonte de predição utilizado (CAMACHO; ALBA, 2007). Por esses motivos a escolha do otimizador tem um papel crucial na viabilidade da arquitetura de controle quando implementada em hardware.

O tipo de algoritmo de otimização a ser utilizado depende do modelo que se está tentando controlar, isto é, linear ou não linear. Para sistemas do tipo *Linear Time-Independent (LTI)*, é possível reduzir a função custo a uma função convexa quadrática. Uma função $f(x)$ é dita convexa se e somente se (SIMON, 2014):

$$f(\gamma x_1 + (1-\gamma)x_2) \leq \gamma f(x_1) + (1-\gamma)f(x_2), \quad \forall \ x_1, x_2 \ , 0 \leq \gamma \leq 1 \tag{3.1}$$

Em outros termos a convexidade de uma função $f(x)$ pode ser descrita da seguinte maneira: dada uma linha reta que intercepte a função $f(x)$ em dois pontos distintos se esta linha permanecer acima da função, esta será convexa (BOYD; VANDENBERGHE, 2004), como mostrado na figura 3.4. Este tipo de função possui apenas um mínimo global, tornando o problema de otimização mais fácil de ser resolvido (ALAMIR, 2013). Além disso, já existe uma grande disponibilidade de *solvers* capazes de resolver problemas de otimização convexa eficientemente (BOYD; VANDENBERGHE, 2004) e em tempo real (EREN et al., 2017).

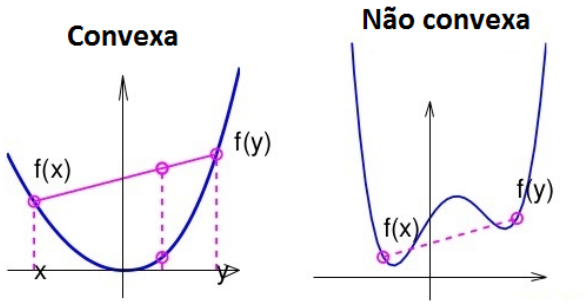


Figura 3.4: Função convexa e não convexa.
Fonte: Adaptado de Tomioka (2012).

Para o caso de sistemas não lineares, não é possível obter uma função custo convexa. Funções não convexas podem ter múltiplos mínimos locais (JOHANSEN, 2011). Isto leva a um problema de otimização

não linear, que na literatura é reconhecido como uma das grandes barreiras na implementação prática do NMPC tem tempo real (OHTSUKA; OZAKI, 2009), (LEE, 2011), (EREN et al., 2017). O motivo é que não há métodos eficazes para resolver o problema geral de programação não-linear (BOYD; VANDENBERGHE, 2004). O problema de otimização pode ser resolvido implicitamente ou explicitamente ou por uma combinação dos dois métodos. A diferença é que o método explícito resolve o problema de otimização *offline* para uma determinada região de operação do modelo e, uma vez que controlador está *online*, ele utiliza *look-up tables* para determinar o valor da função custo naquele instante. Os método implícitos por sua vez resolvem o problema de otimização *online*, durante a operação do controlador. Os trabalhos de Diehl, Ferreau e Haverbeke (2009), Currie (2014) e Eren et al. (2017) fazem uma pesquisa mais abrangente dos tipos de algoritmos de otimização aplicados ao controle preditivo, tanto para o caso linear quanto o não linear.

3.1.2 Formulação geral do controle preditivo

Como exposto na seções anteriores, o controle preditivo prediz o comportamento futuro de um dado processo quando este é submetido a uma sequência de sinais de controle sobre um horizonte de predição N. Esta predição é feita baseada no modelo dinâmico do sistema que se deseja estudar. Pode-se representar o modelo genérico de um sistema dinâmico pela equação 3.2.

$$x(k+1) = f(x(k), u(k)) \tag{3.2}$$

Em que a função $f(\cdot)$ determina a passagem do sistema do estado $x(k)$, no instante de amostragem $k \in \mathbb{N}$, ao estado subsequente $x(k+1)$ quando submetido a um sinal de controle $u(k)$. A formulação geral do MPC apresentada nesta seção tem como referência o trabalho de Murilo (2009) e Mayne et al. (2000).

Analisando a equação 3.2, o valor de x $\in \mathbb{R}^n$ é o vetor de estados, onde n é o número de estados, $\mathbf{u} \in \mathbb{R}^m$ é o vetor de controle, e m é o número de atuadores do sistema. Estas duas grandezas estão definidas no instante k onde $x(k+1)$ é o estado no instante $k+1$ e $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$. Assume-se que o objetivo de controle é estabilizar os estados na origem $x = 0$ e que o par $x = 0$, $u = 0$ é estacionário para a dinâmica da equação 3.2. Além disso, o sinal de controle $\mathbf{u}$ está restrito, de acordo com:

$$u(k) \in \mathbb{U} \subset \mathbb{R}^m \tag{3.3}$$

Onde $\mathbf{U} \subset \mathbb{R}^m$ representa um subconjunto compacto e convexo de valores possíveis para $u(k)$. A sequência geral de comando $\mathbf{u}$ definida em um horizonte de predição N pode ser definida:

$$\mathbf{u} := (u(k) \quad u(k+1) \dots u(k+N-1)) \in \mathbb{U}^N \subset \mathbb{R}^{N.m} \tag{3.4}$$

A equação 3.4 pode ser reescrita em termos do período de amostragem τ e do instante de simulação i.

$$\mathbf{u}(i\tau + t) := u(i) \;\; para \;\; t \in [0, \tau), i \in \{\, k, \ldots, k + N - 1 \,\} \tag{3.5}$$

Considera-se que o subconjunto admissível $\mathbb{U}$ deve conter a origem, ou seja, $0 \in int(\mathbb{U})$, o que significa que o conjunto $\mathbb{U}$ contém a lei de controle assintótica desejada, $u = 0$. O objetivo do controle é encontrar uma lei de controle estabilizadora que respeite a restrição 3.3 e leve a trajetória dos estados para 0, respeitando as seguintes restrições de estado:

$$x(k) \in \mathbb{X} \subset \mathbb{R}^n \tag{3.6}$$

onde $\mathbb{X}$ é um conjunto convexo fechado. Uma restrição de estado final pode ser acrescentada para penalizar o termo $x(k + N)$, para um subconjunto compacto X_f de X:

$$x(k + N) \in \mathbb{X}_f \subset \mathbb{R}^n \tag{3.7}$$

O objetivo do MPC é estabilizar o sistema na origem, para isso deve minimizar uma função custo que reflita os objetivos de controle e respeite as restrições descritas por 3.3 e 3.6. A trajetória dos estados ($x^u(k)$) pode ser obtida através da aplicação de um conjunto de entradas $\mathbf{u}$ definida sob um horizonte de predição N:

$$x^u(k) := (x(k) \;\; x(k+1) \ldots x(k+N)) \in \mathbb{R}^{N.n} \tag{3.8}$$

Pode-se então definir uma função custo genérica em função da sequência de sinais de controle $\mathbf{u}$:

$$J_N(x, \mathbf{u}) = F(x(N)) + \sum_{i=0}^{N-1} L(x(i), u(i)) \tag{3.9}$$

Onde $F(x(N))$ é o custo terminal e o termo $L(x(i), u(i))$ é a ponderação que expressa o desempenho desejado em malha fechada. O N é o horizonte de predição e o termo de ponderação L é escolhido de maneira a respeitar a seguinte expressão:

$$L(x, u) \geq c. \left\| \begin{pmatrix} x \\ u \end{pmatrix} \right\|^2 \;\; para \;\; c > 0 \tag{3.10}$$

Uma vez definido o problema de otimização em 3.9 a estratégia MPC tem como objetivo obter uma sequência de controle ótima através da minimização dessa função custo, respeitando as restrições de comando (3.3), trajetória (3.6) e estado final (3.7). A solução para o problema de otimização pode ser obtida a partir de:

$$\begin{aligned} P_N(k) := \arg\min_u \left[J_N(x, \mathbf{u}) \right] \text{ sujeito a:} \\ \mathbf{u} \in \mathbb{U}^N \;\; , \;\; x(i) \in \mathbb{X} \;\; , \;\; x(N) \in X_f \end{aligned} \tag{3.11}$$

Resolvendo-se o problema de otimização 3.11 obtém-se a seguinte sequência de controle ótima:

$$\hat{\mathbf{u}}(x) = (\hat{u}^{(0)}(x), \ldots, \hat{u}^{(N-1)}(x)) \ , \ \hat{u}^{(i)}(x) \in \mathbb{R}^m. \tag{3.12}$$

Da sequência obtida 3.12, apenas o primeiro valor é aplicado ao sistema, $\hat{u}^{(0)}(x)$, durante o intervalo $[k, k+1[$, resultando na seguinte realimentação discreta de estados:

$$k_N(x) = \hat{u}^{(0)}(x) \tag{3.13}$$

Após a aplicação do valor de comando dado por 3.13, o horizonte de predição é deslocado para frente e todo o processo é repetido.

No que diz respeito às condições de estabilidade, Mayne et al. (2000) emprega a equação 3.9 como função de *Lyapunov* para a análise de estabilidade do controle preditivo, quando uma restrição terminal é utilizada. Os principais pontos da análise empregada pelo autor referem-se ao valor custo terminal $F(.)$ e o conjunto de restrições finais X_f.

Uma vez definido o problema de otimização (3.11), Mayne et al. (2000) indicam que o custo terminal $F(.)$ e a restrição terminal $x(N) \in X_f$ deve satisfazer $F(x) = 0$ e $X_f = 0$ para garantir estabilidade do controlador. A estabilidade requer alta penalização na parte final da trajetória. Isto pode ser obtido através de um horizonte infinito por meio de uma igualdade final ou inclusão de uma restrição no valor do estado final (ALAMIR, 2013).

Capítulo 4

Metodologia

Neste capítulo será introduzido o modelo da plataforma de satélites que foi utilizado para o desenvolvimento das estratégias de controle. Será abordado o desenvolvimento do controlador preditivo linear (MPC) e não linear (NMPC). Para cada uma destas, foi desenvolvida a técnica de parametrização com o objetivo de reduzir o custo computacional associado à estratégia de controle preditivo, como já discutido no capítulo 3. Para o desenvolvimento destes controladores serão seguidas todas as etapas do *V-cycle*: MIL, SIL, PIL e HIL. Para cada etapa será feita uma exposição do tipo de *hardware* e *software* envolvidos nos testes.

4.1 Validação do modelo da plataforma de testes de satélites

Esta seção busca explicar o funcionamento da plataforma de testes de satélites utilizada neste trabalho. Aqui serão feitas as explicações sobre seu funcionamento e obtenção das equações cinemáticas e dinâmicas. A modelagem aqui apresentada foi obtida do trabalho de Gonzales (2009), não havendo nenhuma modificação nas equações.

4.1.1 Modelo da plataforma de testes de satélites

O modelo da plataforma de testes de satélites utilizada nesta dissertação é apresentada na figura 4.1. Nesta plataforma os elementos responsáveis por manter a orientação desejada são as rodas de reação. A roda de reação é um motor ligado a um volante de alta inércia que é livre para girar em torno de um eixo fixo da espaçonave (VOTEL; SINCLAIR, 2012). Este atuador baseia-se no princípio da conservação do momento angular, que afirma que um sistema sem a ação de torques externos, a quantidade de momento angular é conservada. Através da rotação do volante, a plataforma tende a rotacionar no sentido contrário tentando anular o momento angular gerado pelo volante.

Para propósitos de controle devem-se identificar os principais sistemas de referência existentes na plataforma, que são dois. O primeiro é o sistema de referência inercial $F_i(i_1, i_2, i_3)$, localizado no centro do mancal esférico, considerado o centro de rotação do simulador. O segundo sistema de referência é aquele relacionado à parte móvel da plataforma, F_b (não visível na figura). A parte móvel corresponde à

Figura 4.1: Plataforma de testes de satélites.
Fonte: (GONZALES, 2009)

mesa que está sobre o colchão de ar gerado pela plataforma. Este sistema possui o mesmo centro de F_i, variando-se apenas a orientação do mesmo com respeito ao sistema inercial. Considera-se também que o sistema de referência do corpo está orientado conforme os eixos principais de inércia da mesa giratória.

Denomina-se atitude do simulador a orientação relativa entre o sistema de referência inercial F_i e o sistema de referência do corpo F_b fixo à mesa giratória. Para descrever a orientação de F_b com relação a Fi, utilizam-se os ângulos de *euler* na sequência de rotação 3-2-1, ou seja, partindo-se de F_i para se chegar a F_b, o eixo 3 deverá ser rotacionado de um ângulo θ_1, o eixo 2 deverá ser rotacionado de um ângulo θ_2, e o eixo 1 deverá ser rotacionado de um ângulo θ_3. O modelo matemático da plataforma é não linear e pode ser descrito pelas seguintes equações, para um desenvolvimento completo do modelo, verificar (GONZALES, 2009):

$$\dot{\theta}_1 = \omega_2.\frac{sin(\theta_3)}{cos(\theta_2)} + \omega_3.\frac{cos(\theta_3)}{cos(\theta_2)} \tag{4.1}$$

$$\dot{\theta}_2 = \omega_2.cos(\theta_3) - \omega_3.sin(\theta_3) \tag{4.2}$$

$$\dot{\theta}_3 = \omega_1 + \omega_2.\frac{sin(\theta_3).sin(\theta_2)}{cos(\theta_2)} + \omega_3.\frac{cos(\theta_3).sin(\theta_2)}{cos(\theta_2)} \tag{4.3}$$

$$\dot{\omega}_1 = \omega_2.\frac{(I_{22}\omega_3 - I_\omega\Omega_3)}{(I_{11}+I_\omega)} + \omega_3.\frac{(-I_{33}\omega_2 + I_\omega\Omega_2)}{(I_{11}+I_\omega)} - \dot{\Omega}_1.\frac{I_\omega}{(I_{11}+I_\omega)} \tag{4.4}$$

$$\dot{\omega}_2 = \omega_1.\frac{(-I_{11}\omega_3 + I_\omega\Omega_3)}{(I_{22}+I_\omega)} + \omega_3.\frac{(I_{33}\omega_1 - I_\omega\Omega_1)}{(I_{22}+I_\omega)} - \dot{\Omega}_2.\frac{I_\omega}{(I_{22}+I_\omega)} \tag{4.5}$$

$$\dot{\omega}_3 = \omega_1.\frac{(I_{11}\omega_2 - I_\omega\Omega_2)}{(I_{33}+I_\omega)} + \omega_2.\frac{(-I_{22}\omega_1 + I_\omega\Omega_1)}{(I_{33}+I_\omega)} - \dot{\Omega}_3.\frac{I_\omega}{(I_{33}+I_\omega)} \tag{4.6}$$

As variáveis θ_1, θ_2 e θ_3 são os ângulos de *Euler* que descrevem a atitude do simulador como a orientação relativa entre o sistema de referência inercial F_i e o sistema fixo no corpo F_b. Estes são os ângulos medidos pela rotação da plataforma em relação aos eixos i_1, i_2 e i_3. Por sua vez as variáveis ω_1, ω_2 e ω_3 são as velocidades angulares da plataforma do eixo de referência F_b em relação a F_i.

As variáveis de comando, responsáveis pela movimentação da plataforma, são as acelerações das rodas de reação, ou seja, $\dot{\Omega}_1$, $\dot{\Omega}_2$ e $\dot{\Omega}_3$. Os termos I_{11}, I_{22}, I_{33} referem-se ao momentos de inércia da plataforma em relação aos eixos i_1, i_2 e i_3 respectivamente e I_w é o momento de inércia das rodas de reação, igual para as três presentes no sistema. Por fim, as variáveis Ω_1, Ω_2, Ω_3 são as velocidades angulares das rodas de reação.

No trabalho desenvolvido por Gonzales (2009), foi utilizado um estimador de *Kalman* para estimar as velocidades angulares da plataforma, considerando que os ângulos são medidos diretamente. No presente trabalho esta abordagem não foi utilizada. Aqui consideram-se os ângulos de *euler* e as velocidades angulares da plataforma são diretamente medidos. As equações apresentadas (4.1 - 4.6) mostram que o modelo da plataforma é um sistema com múltiplas entradas e múltiplas saídas (MIMO), sendo possível reescrever este conjunto de equações no seguinte formato em espaço de estados:

$$\begin{cases} \dot{x} = A(x,u)x + Bu \\ y = Cx + Du \end{cases} \tag{4.7}$$

Onde $x \in \mathbb{R}^n$ é o vetor de estados e *n* a ordem do sistema. As matrizes $\mathbf{A}(x,u) \in \mathbb{R}^{n\times n}$ e $\mathbf{B} \in \mathbb{R}^{n\times n_u}$ descrevem a dinâmica do sistema. A explicação do fato de a matriz A não ser constante será vista mais adiante. O vetor $u \in \mathbb{R}^{n_u}$ por sua vez corresponde às entradas e n_u o número de atuadores no sistema. Denomina-se C a matriz de saída, indicando quais estados estão sendo medidos e por fim a matriz de transmissão direta *D*.

Para o modelo da plataforma, o vetor de estados é definido como $x = (\theta_1 \quad \theta_2 \quad \theta_3 \quad \omega_1 \quad \omega_2 \quad \omega_3)^T$ e o vetor de controle é definido como $u = (\dot{\Omega}_1 \quad \dot{\Omega}_2 \quad \dot{\Omega}_3)^T$. Como indicado em Gonzales (2009), assume-se que todos os estados do sistema são observáveis, logo a matriz de saída pode ser escrita como a matriz identidade $C = \mathbb{I}_{6\times6}$ e a matriz de transmissão direta é $D = 0_{6\times3}$ visto que o sistema não possui alimentação direta de entrada. A matriz *A(x,u)* é definida da seguinte maneira:

$$A(x,u) = \begin{bmatrix} 0 & 0 & 0 & 0 & \frac{\sin(\theta_3)}{\cos(\theta_2)} & \frac{\cos(\theta_3)}{\cos(\theta_2)} \\ 0 & 0 & 0 & 0 & \cos(\theta_3) & -\sin(\theta_3) \\ 0 & 0 & 0 & 1 & \frac{\sin(\theta_3)\sin(\theta_2)}{\cos(\theta_2)} & \frac{\cos(\theta_3)\sin(\theta_2)}{\cos(\theta 2)} \\ 0 & 0 & 0 & 0 & \frac{(I_{22}\omega_3 - I_\omega\Omega_3)}{(I_{11}+I_\omega)} & \frac{(-I_{33}\omega_2 + I_\omega\Omega_2)}{(I_{11}+I_\omega)} \\ 0 & 0 & 0 & \frac{(-I_{11}\omega_3 + I_\omega\Omega_3)}{(I_{22}+I_\omega)} & 0 & \frac{(I_{33}\omega_1 - I_\omega\Omega 1)}{(I_{22}+I_\omega)} \\ 0 & 0 & 0 & \frac{(I_{11}\omega_2 - I_\omega\Omega_2)}{(I_{33}+I_\omega)} & \frac{(-I_{22}\omega_1 + I_\omega\Omega_1)}{(I_{33}+I_\omega)} & 0 \end{bmatrix} \tag{4.8}$$

Nota-se que a matriz acima não é constante pois depende das velocidades angulares das três rodas de reação Ω_1, Ω_2, Ω_3. A matriz B é constante e pode ser definida como:

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{-I_\omega}{I_{11}+I_\omega} & 0 & 0 \\ 0 & \frac{-I_\omega}{I_{22}+I_\omega} & 0 \\ 0 & 0 & \frac{-I_\omega}{I_{33}+I_\omega} \end{bmatrix} \tag{4.9}$$

O desenvolvimento teórico do controle preditivo linear está baseado no uso de modelos lineares e invariantes no tempo (LTI). Por este motivo é necessário obter a linearização da matriz 4.8 em uma determinada região de operação. A linearização da matriz 4.8 é obtida através da aplicação do Jacobiano calculado em um determinado ponto de operação (x_0, u_0). O jacobiano é definido como:

$$J = \frac{\partial f_i}{\partial x_j} = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_6} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_6} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_6}{\partial x_1} & \frac{\partial f_6}{\partial x_2} & \cdots & \frac{\partial f_6}{\partial x_6} \end{pmatrix} \tag{4.10}$$

O ponto operacional escolhido para linearização é aquele para pequenas excursões em torno da origem, ou seja, pequenas variações de ângulo, velocidade angular e aceleração. Calculando-se o jacobiano no ponto de operação $x_0 = (\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3) = (0, 0, 0, 0, 0, 0)$ e $u_0 = (\dot{\Omega}_1, \dot{\Omega}_2, \dot{\Omega}_3) = (0, 0, 0)$, obtêm-se a matriz linearizada:

$$A_l = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{4.11}$$

Logo as matrizes $A_l(4.11)$ e $B(4.9)$ serão utilizadas para o desenvolvimento do controlador preditivo linear. Para determinar a controlabilidade do sistema, deve-se verificar se o *rank* da matriz de controlabilidade é igual a *n*, onde $n = 6$ é a dimensão da matriz A. A matriz de controlabilidade (Co) é definida:

$$Co = \begin{bmatrix} B \\ AB \\ A^2B \\ \vdots \\ A^{n-1}B \end{bmatrix} \tag{4.12}$$

Realizando os cálculos percebe-se que *rank(Co) = 6*. Logo, o sistema dinâmico linear descrito pela matrizes A_l, B, C, D é controlável.

4.2 Formulação do controlador preditivo linear - MPC.

A partir da obtenção do modelo linear da plataforma de testes de satélites, esta seção irá desenvolver todas as equações necessárias para o controlador preditivo linear. Para se definirem adequadamente as expressões do controlador MPC para sistemas LTI, os seguintes elementos são necessários: o modelo linear do sistema, a função custo e as restrições. Nesta seção, será utilizado o desenvolvimento teórico feito por Alamir (2013). Inicialmente deve-se considerar o modelo discreto em espaço de estados do sistema a ser estudado, visto na equação 4.13.

$$x(k+1) = \mathbf{A}x(k) + \mathbf{B}u(k) \tag{4.13}$$

onde $x(k) \in \mathbb{R}^n$ é o vetor de estados e *n* a ordem do sistema. As matrizes $\mathbf{A} \in \mathbb{R}^{n \times n}$ e $\mathbf{B} \in \mathbb{R}^{n \times n_u}$ são constantes no tempo e descrevem a dinâmica do sistema. O vetor $u(k) \in \mathbb{R}^{n_u}$, por sua vez, corresponde às entradas de controle e n_u, o número de entradas do sistema. O mapa de predição com horizonte *N* pode ser obtido calculando-se a sequência *N* de ações futuras, onde *N* é chamado de horizonte de predição. A sequência de ações futuras é dada por:

$$\tilde{\mathbf{u}}(k) := (u(k) \quad u(k+1) \dots u(k+N-1))^T \in \mathbb{R}^{N.n_u} \tag{4.14}$$

O mapa de predição com horizonte *N* permite predizer para cada sequência $\tilde{\mathbf{u}}(k)$ de ações futuras, a trajetória dos estados futuros $\tilde{x}(k|\tilde{u}(k))$ a partir dos estados $x(k)$, onde:

$$\tilde{x}(k|\tilde{u}(k)) := (x(k+1) \quad x(k+2) \dots x(k+N))^T \in \mathbb{R}^{N.n} \tag{4.15}$$

Para sistemas LTI é possível obter uma expressão geral para $\tilde{x}(k|\tilde{u}(k))$ em função das matrizes $\mathbf{A}$ e $\mathbf{B}$ da equação 4.13 e da sequência de controle $\tilde{\mathbf{u}}(k)$. Esta relação é expressa pela equação 4.16.

$$x(k+i) = \Phi_i x(k) + \Psi_i \tilde{u}(k) \tag{4.16}$$

As matrizes Φ_i e Ψ_i para $i \in \{1, ..., N\}$ são definidas por sua vez como:

$$\Phi_i := A^i \tag{4.17}$$

$$\Psi_i := [A^{i-1}B, ..., AB, B] \left(\Pi_1^{(n,N)} \dots \Pi_{i-1}^{(n,N)} \quad \Pi_i^{(n,N)} \right)^T \tag{4.18}$$

Onde $\Pi_i^{(n,N)}$ é chamada de matriz de seleção. Seu objetivo é selecionar o i-ésimo vetor de dimensão n de um vetor composto pela concatenação de N desses vetores. Ela é definida da seguinte maneira:

$$\Pi_i^{(n,N)} := (\underbrace{\mathbb{O}_{nxn}, ..., \mathbb{O}_{nxn}}_{\text{(i-1) termos}} \quad \mathbb{I}_{nxn} \quad \underbrace{\mathbb{O}_{nxn}, ..., \mathbb{O}_{nxn}}_{\text{(N-i) termos}}) \in \mathbb{R}^{n \times (Nn)} \tag{4.19}$$

Para definição da função custo, deve-se primeiramente indicar o vetor de saída do sistema y_r, ou seja, os estados que estão sendo regulados:

$$y_r = C_r x(k) \in \mathbb{R}^{n_r} \tag{4.20}$$

Em que y_r é o vetor que contém n_r combinações lineares do vetor de estado $x(k)$. A matriz $C_r \in \mathbb{R}^{n_r \times n}$ indica efetivamente quais estados serão regulados. Caso todos os estados sejam regulados, então $C_r = \mathbb{I}_{nxn}$. A função custo a ser minimizada correspondente a uma sequência $\tilde{\mathbf{u}}$ de ações futuras, dado o estado atual $x(k)$ e a trajetória de saída desejada $\tilde{y}_r^d$, definida sob o horizonte de predição $[k, k+N]$ é definida:

$$J(\tilde{\mathbf{u}}|x(k), \tilde{y}_r^d(k), \mathbf{u}^d) := \sum_{i=1}^{N} ||y_r(k+i) - y_r^d(k+i)||_{Q_y}^2 + \sum_{i=1}^{N} ||\Pi_i^{(n_u,N)} \tilde{\mathbf{u}} - \mathbf{u}^d||_{Q_u}^2 \tag{4.21}$$

A determinação dessa função custo foi obtida a partir dos trabalho de Alamir (2013). O primeiro termo da equação 4.21, chamado de erro de trajetória, penaliza a diferença entre a referência desejada y_r^d passada a cada um dos estados definidos por y_r. O segundo termo representa a penalização do vetor de controle $\tilde{\mathbf{u}}$. O valor $\mathbf{u}^d$ é o valor de comando desejado em regime permanente. A matriz $Q_y \in \mathbb{R}^{n_r \times n_r}$ é a matriz de ponderação simétrica quadrada utilizada para penalizar o erro de trajetória e n_r representa o número de estados regulados. A norma quadrática de uma matriz *X* ponderada por uma matriz *Q*, ou seja, $||X||^2{}_Q$ é definida da seguinte maneira:

$$||X||_Q^2 = X^T Q X \tag{4.22}$$

A matriz $Q_u \in \mathbb{R}^{n_u \times n_u}$, por sua vez, é a matriz definida positiva de penalização do comando $\tilde{\mathbf{u}}$. Esta matriz determina qual o grau de variação de comando. Se a penalização de comando for alta o controle tem menos liberdade de variação e ser for baixa o comando pode variar com mais rapidez. A equação (4.21) pode ser colocada na forma de uma função quadrática padrão na variável de decisão $\tilde{\mathbf{u}}$, juntamente com as seguintes definições propostas por Alamir (2013):

$$J(\tilde{\mathbf{u}}|x(k), \tilde{y}_r^d(k), \mathbf{u}^d) := \frac{1}{2}\tilde{u}^T H \tilde{u} + [F_1 x(k) + F_2 \tilde{y}_r^d + F_3 u^d]^T \tilde{u} + \textit{Cte} \tag{4.23}$$

$$H := 2\sum_{i=1}^{N}[\Psi_i^T C_r^T Q_y C_r \Psi_i + (\Pi_i^{(n_u,N)})^T Q_u (\Pi_i^{(n_u,N)})] \tag{4.24}$$

$$F_1 := 2\sum_{i=1}^{N}[\Psi_i^T C_r^T Q_y C_r \Phi_i] \tag{4.25}$$

$$F_2 := -2\sum_{i=1}^{N}[\Psi_i^T C_r^T Q_y \Pi_i^{(n_r,N)}] \tag{4.26}$$

$$F_3 := 2\sum_{i=1}^{N}[(\Pi_i^{(n_u,N)})^T Q_u] \tag{4.27}$$

O valor *Cte* na equação 4.23 não envolve nenhum termo que possa ser influenciado pela escolha da sequência $\tilde{\mathbf{u}}$. Este termo apenas faz a função custo transladar verticalmente sem modificar o valor do argumento que minimiza a função custo. Logo, ele pode ser descartado para fins de cálculo.

Depois de definida a função custo, faz-se necessário definir e incluir, no equacionamento do MPC, quais são as restrições impostas pelo modelo. Chamam-se saídas restritas as variáveis que possuem restrições a serem respeitadas. As restrições podem ser de três tipos: de estados, de comando e variação de comando. Para sistemas LTI as restrições associadas aos estados são definidas através da matriz de restrições C_c, mais precisamente:

$$y_c := C_c x + D_c u \tag{4.28}$$

A equação acima deve obedecer à seguinte inequalidade $y_c^{min} \leq y_c \leq y_c^{max}$. Onde $y_c^{min}, y_c^{max} \in \mathbb{R}^{n_c}$ são os limites mínimo e máximo, respectivamente, dos estados do sistema. Nota-se que, devido ao horizonte de predição, a inequalidade deve ser calculada para todos os instantes $k+i$ onde $i \in \{1, ..., N\}$. Isto é:

$$y_c^{min} \leq y_c(k+i) = C_c x(k+i) + D_c u(k+i-1) \leq y_c^{max} \tag{4.29}$$

Substituindo as matrizes de seleção $\Pi_i^{(n,N)}$ e $\Pi_i^{(n_u,N)}$ (equação 4.19) e o mapa de N passos a frente (equação 4.16) na equação 4.29, tem se:

$$\underbrace{\begin{pmatrix} +C_c\Psi_1 + D_c\Pi_1^{(nu,N)} \\ \vdots \\ +C_c\Psi_N + D_c\Pi_N^{(nu,N)} \\ -C_c\Psi_1 - D_c\Pi_1^{(nu,N)} \\ \vdots \\ -C_c\Psi_N - D_c\Pi_N^{(nu,N)} \end{pmatrix}}_{A_{ineq}(1)} \tilde{u} \leq \underbrace{\begin{pmatrix} -C_c\Phi_1 \\ \vdots \\ -C_c\Phi_N \\ +C_c\Phi_1 \\ \vdots \\ +C_c\Phi_N \end{pmatrix}}_{G1^{(1)}} x(k) + \underbrace{\begin{pmatrix} +y_c^{max} \\ \vdots \\ +y_c^{max} \\ -y_c^{min} \\ \vdots \\ -y_c^{min} \end{pmatrix}}_{G3^{(1)}}$$

A restrição associada a variação de controle máxima (δ^{max}) e mínima (δ^{min}) é expressa da seguinte maneira:

$$\forall i \in \{1, ..., N\} \quad \delta^{min} \leq u(k+i) - u(k+i-1) \leq \delta^{max} \tag{4.30}$$

Escrevendo-se a equação anterior matricialmente:

$$\underbrace{\begin{pmatrix} +\mathbb{I} & \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{0} \\ -\mathbb{I} & +\mathbb{I} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ +\mathbb{0} & \mathbb{0} & \mathbb{0} & \cdots & -\mathbb{I} & +\mathbb{I} \\ -\mathbb{I} & \mathbb{0} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{0} \\ +\mathbb{I} & -\mathbb{I} & \mathbb{0} & \cdots & \mathbb{0} & \mathbb{0} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbb{0} & \mathbb{0} & \mathbb{0} & \cdots & +\mathbb{I} & -\mathbb{I} \end{pmatrix}}_{A_{ineq}(2)} \tilde{u} \leq \underbrace{\begin{pmatrix} +\mathbb{I} \\ \mathbb{0} \\ \vdots \\ \mathbb{0} \\ -\mathbb{I} \\ \mathbb{0} \\ \vdots \\ \mathbb{0} \end{pmatrix}}_{G2^{(2)}} u(k-1) + \underbrace{\begin{pmatrix} +\delta^{max} \\ +\delta^{max} \\ \vdots \\ +\delta^{max} \\ -\delta^{min} \\ -\delta^{min} \\ \vdots \\ -\delta^{min} \end{pmatrix}}_{G3^{(2)}}$$

Considerando as restrições máximas (u^{max}) e mínimas (u^{min}) de comando, o comando deve obedecer a seguinte inequação:

$$\forall i \in \{1, ..., N\} \quad u^{min} \leq u(k+i+1) \leq u^{max} \tag{4.31}$$

Onde os vetores $u^{min}, u^{max} \in \mathbb{R}^{n_u}$ são limites que dependem do atuador. Pode-se reescrever a equação (4.31) no formato matricial:

$$\begin{pmatrix} u^{min} \\ \vdots \\ u^{min} \end{pmatrix} \leq \tilde{u} \leq \begin{pmatrix} u^{max} \\ \vdots \\ u^{max} \end{pmatrix} \in \mathbb{R}^{Nn_u} \tag{4.32}$$

Por fim, depois de definida a função custo (4.23), as restrições de estado, comando e variação

de comando, a sequência ótima de comandos $\tilde{\mathbf{u}}^{opt}(x(k))$ é obtida da solução do seguinte problema de otimização:

$$\begin{gathered}\tilde{\mathbf{u}}^{opt}(k) := \arg\min_{u}\left[\frac{1}{2}\tilde{\mathbf{u}}^{T}H\tilde{\mathbf{u}} + F^{T}(K)\tilde{\mathbf{u}}\right] \text{ sujeito a:} \\ A_{ineq}\tilde{\mathbf{u}} \leq B_{ineq}(k) \ , \ \tilde{\mathbf{u}}^{min} \leq \tilde{\mathbf{u}} \leq \tilde{\mathbf{u}}^{max}\end{gathered} \tag{4.33}$$

Onde $F(k)$ e $B_{ineq}(k)$ são definidos:

$$F(k) = F_1 x(k) + F_2 \tilde{y_r}^{d}(k) + F_3 \mathbf{u}^{d} \tag{4.34}$$

$$B_{ineq}(k) = G_1 x(k) + G_2 \mathbf{u}(k-1) + G_3 \tag{4.35}$$

Como apenas o primeiro valor da sequência $\tilde{\mathbf{u}}^{opt}(x(k))$ (equação 4.33) é utilizado, então o ganho ótimo do MPC a cada instante de tempo *k* é:

$$K_{MPC}(x(k)) = \Pi_1^{(nu,N)}.\tilde{\mathbf{u}}^{opt}(x(k)) \tag{4.36}$$

Valor que é aplicado ao sistema durante o intervalo de tempo $[k, k+1[$.

4.2.1 Parametrização dos controladores MPC

Da seção anterior é possível concluir que o problema de otimização pode tornar-se um impeditivo na implementação da estratégia preditiva. Uma vez que este problema deve ser resolvido *online* ele pode gerar problemas de implementação em tempo real. Algumas técnicas foram desenvolvidas com o intuito de reduzir o esforço computacional na etapa de otimização do MPC. O trabalho apresentado por Ling et al. (2010) utiliza uma variante do MPC chamada de *Multiplexed MPC (MMPC)*, aplicada ao processo de produção de semicondutores. O *MMPC* subdivide o problema de otimização inicial em várias partes menores, resolvendo-as sequencialmente e atualizando um sinal de controle por vez (LING et al., 2012), ao invés de todos os sinais de controle ao mesmo tempo como a estratégia MPC clássica faz. Esta técnica permite reduzir a carga computacional em tempo-real, possibilitando utilizar menores períodos de amostragem em sistemas multivariáveis.

Em Xing, Low e Pham (2012) é utilizada a abordagem de *Distributed MPC (DMPC)* para controle de atitude de um satélite com movimento em três eixos. O DMC é uma estratégia de controle distribuída onde, partindo de um sistema completo, criam-se vários subsistemas e o controle de cada um desses subsistemas é feito por uma estratégia MPC de menor complexidade. Logo, ao invés de se ter apenas uma função custo total do sistema, existem várias funções custo locais. Os controladores funcionam em processadores separados e se comunicam entre si para alcançar o objetivo final de controle (CHRISTOFIDES et al., 2013).

Por sua vez, Chen e Wu (2010) constroem a formulação do MPC através de funções de *Laguerre*, aplicado ao controle de atitude de um *CubeSat*. O uso destas funções busca simplificar o problema de otimização, na justificativa de que o computador a bordo do satélite é bastante limitado em termos computacionais e incapaz de lidar com a formulação MPC clássica. Outra abordagem possível é a utilização de técnicas de parametrização do sinal de controle. O objetivo da parametrização é reduzir a dimensão do problema de otimização, consequentemente diminuindo sua complexidade computacional (ALAMIR, 2006). Assumindo que o vetor de controle $\tilde{\mathbf{u}}$ contém n_u componentes, ou seja, número de atuadores presentes no sistema, o problema de otimização do MPC clássico, sem considerar a parametrização, tem como número de graus de liberdade $n_p = N.n_u$. Logo o vetor de controle ótimo a ser encontrado pela otimização da função custo é:

$$\tilde{u}(k) = \begin{pmatrix} \text{u(k)} \ \in \mathbb{R}^{n_u} \\ \vdots \\ \text{u(k + N - 1)} \ \in \mathbb{R}^{n_u} \end{pmatrix} \tag{4.37}$$

A parametrização tem como objetivo diminuir o número de graus de liberdade n_p do problema de otimização. A figura 4.2 mostra o caso da parametrização quando o a sequência de comandos $\tilde{\mathbf{u}}$ é dividida em $n_r = 3$ regiões.

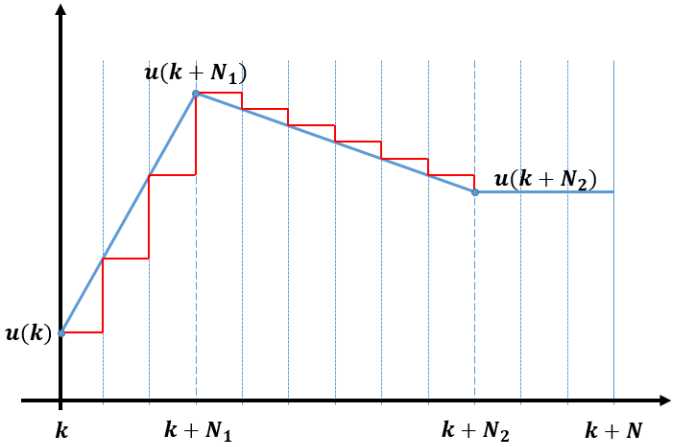


Figura 4.2: Parametrização para $n_r = 3$.
Fonte: Adaptado de Alamir (2013).

Logo a nova sequência de comandos nos instantes intermediários será:

$$\mathbf{p} := \begin{pmatrix} u(k) \\ u(k+N_1) \\ \vdots \\ u(k+N_{n_r-1}) \end{pmatrix} \tag{4.38}$$

Percebe-se que o número de graus de liberdade n_p foi reduzido para $n_p = n_r.n_u$. O problema de otimização a partir de agora não será mais função do horizonte de predição N. Para se encontrar o

valor intermediário de $\mathbf{u}(k+i)$ no instante $i \in [N_{ji}, N_{ji+1}]$, deve-se encontrar a equação da reta do tipo $y(x) = ax + b$ que passa pelos pontos consecutivos $(k+N_1, u(k+N_1))$ e $(k+N_2, u(k+N_2))$ da figura 4.2, logo:

$$u(k+i) := \frac{u(k+N_{j_{i+1}}) - u(k+N_{j_i})}{N_{j_{i+1}} - N_{j_i}}.(i - N_{j_i}) + u(k+N_{j_i}) \tag{4.39}$$

Diferentemente da interpolação utilizando uma reta na forma $y(x) = ax + b$, é a utilização da parametrização exponencial que usa funções do tipo $e^{(x)}$ para obter o perfil de comando $\mathbf{u}$. Outra possibilidade é a utilização de funções *B-splines* (ILKIV; RUSKO, 2005) para parametrização de comando. Para a parametrização exponencial a sequência de comandos $\mathbf{u}$ pode ser escrita da seguinte maneira:

$$u_j(k+i) := \sum_{l=1}^{n_e^{(j)}} \underbrace{\left[e^{\frac{-\lambda_j(i\tau)}{(l-1)\alpha+1}} \right]}_{m_{j,l}(i)} \cdot p_l^{(j)}; \alpha > 1 \tag{4.40}$$

Onde *j* corresponde ao j-ésimo atuador do sistema. O termo $n_e^{(j)}$ indica o número de exponenciais escolhidas para o atuador *j*, τ o período de amostragem, λ_j e $\alpha > 1$ são parâmetros do controlador, definidos pelo projetista. Uma vez que o parâmetro α é definido, o perfil de controle pode ser descrito da seguinte forma:

$$\mathbf{p} := \begin{pmatrix} p^{(1)} \in \mathbb{R}^{n_e^{(1)}} \\ \vdots \\ p^{(n_u)} \in \mathbb{R}^{n_e^{(n_u)}} \end{pmatrix} \in \mathbb{R}^{n_p} \tag{4.41}$$

A expressão 4.40 pode ser escrita de maneira mais compacta:

$$u_j(k+i) := [M_j(i)] \cdot p^{(j)}; p^{(j)} \in \mathbb{R}^{n_e^{(j)}} \tag{4.42}$$

Onde $M_j(i) \in \mathbb{R}^{1 \times n_e^{(j)}}$ é definido:

$$M_j(i) := (m_{j,1}(i) \cdots m_{j,n_e^{(j)}}(i)) \tag{4.43}$$

Reescrevendo a equação 4.42 para cada um dos atuadores existentes no problema, ou seja, $j = 1, ..., n_u$:

$$u(k+i) = \underbrace{BlockDiag\left(M_j(i)_{j=1}^{n_u}\right)}_{M(i)} \begin{pmatrix} p^{(1)} \\ \vdots \\ p^{(n_u)} \end{pmatrix}$$

$$= [M(i)] \cdot p \tag{4.44}$$

Recalculando 4.44 em cada um dos instantes de simulação $i = 0, ..., N-1$, tem se a expressão final da parametrização exponencial Π_e:

$$\tilde{u} = \Pi_e \cdot p \tag{4.45}$$

Onde:

$$\Pi_e := \begin{pmatrix} M(0) \\ \vdots \\ M(N-1) \end{pmatrix} \tag{4.46}$$

Corresponde ao valor da parametrização no instante i que será aplicado ao sistema.

4.3 Formulação do controlador preditivo não linear - NMPC

A importância do estudo dos modelos não lineares advém do fato que a maioria dos processos existentes na natureza são deste tipo (LAILA, 2003). Apesar do modelo não linear de um sistema representar seu comportamento real com mais fidelidade, ele por vezes é de difícil tratamento, seja do ponto de vista da complexidade computacional de simulação ou para o desenvolvimento de leis de controle. Geralmente para fins de controle é obtido a linearização do modelo não linear do sistema. Um dos problemas desta abordagem é que o modelo obtido é válido apenas em uma determinada região de operação. Caso o modelo seja levado para longe do ponto de linearização, a estratégia de controle desenvolvida deixa de funcionar. Este fato motiva o estudo de leis de controle não lineares para lidar com tais sistemas.

4.3.1 Parametrização do NMPC

A implementação da estratégia de controle preditivo é mais complexa para o caso do NMPC (OHTSUKA; OZAKI, 2009), onde a função custo é não convexa, podendo existir muitos mínimos locais. Isto por si só exige algoritmos de otimização mais elaborados e consequentemente mais poder computacional que o caso linear (GROS et al., 2016).

Da mesma maneira que foi desenvolvido para o caso linear, é possível aplicar a técnica de parametrização exponencial para o NMPC partindo da formulação geral apresentada na subseção 3.1.2. Neste trabalho será utilizada a técnica de parametrização proposta por Murilo, Alamir e Alberer (2014). A parametrização é importante por que pode viabilizar a aplicação do NMPC em *hardware*. O perfil geral do comando parametrizado utilizando-se duas exponencias para cada comando é dado por:

$$u_n(i\tau + t) = Sat_{u_{min}}^{u_{max}}(u_n^* + \alpha_1^{u_n}.e^{-\lambda i\tau} + \alpha_2^{u_n}.e^{-q\lambda i\tau}) \text{ para t} \in \mathbb{R}\ [(k-1)\tau, k\tau[\tag{4.47}$$

Onde $n \in \{1, ..., n_u\}$ é o número de atuadores do sistema, τ é o período de amostragem, os valores

$\lambda > 0$ e $q \in \mathbb{N}$ são parâmetros de sintonia. Os termos $\alpha_1^{u_n}$ e $\alpha_2^{u_n} \in \mathbb{R}^m$ são os coeficientes das exponenciais a serem determinados para cada um dos atuadores. O mapa de saturação $Sat_{u_{min}}^{u_{max}}$ é definido da seguinte maneira:

$$Sat_{u_{min}^n}^{u_{max}^n}(u_n) = \begin{cases} u_{min}^n, & \text{se } u_n \leq u_{min}^n \\ u_{max}^n, & \text{se } u_n \geq u_{max}^n \\ u, & \text{Demais casos} \end{cases} \tag{4.48}$$

Retirando-se $Sat_{u_{min}}^{u_{max}}$ da equação 4.47 para simplificar os desenvolvimentos posteriores, ela pode ser reescrita para cada um dos três atuadores do modelo:

$$\begin{aligned} u_1(i\tau + t) &= (u_1^* + \alpha_1^{u_1}.e^{-\lambda i\tau} + \alpha_2^{u_1}.e^{-q\lambda i\tau}) \\ u_2(i\tau + t) &= (u_2^* + \alpha_1^{u_2}.e^{-\lambda i\tau} + \alpha_2^{u_2}.e^{-q\lambda i\tau}) \\ u_3(i\tau + t) &= (u_3^* + \alpha_1^{u_3}.e^{-\lambda i\tau} + \alpha_2^{u_3}.e^{-q\lambda i\tau}) \end{aligned} \tag{4.49}$$

Os valores u_n^* correspondem ao valor em estado estacionário para cada variável de controle *u*. Este valor é dependente do modelo que se está controlando. Para o caso do satélite espera-se que o comando em estado estacionário seja zero, pois uma vez que o satélite se encontra na atitude desejada, seus atuadores devem parar. Esta é uma consideração forte, pois sabe-se que no espaço existem pertubações que evitam que o satélite permaneça livre de torques externos, porém estes não foram levados em consideração durante os testes do controlador. Para se encontrar uma relação entre os valores das exponenciais $\alpha_1^{u_n}$ e $\alpha_2^{u_n}$ com os valores constante λ e q, deve-se calcular o comando (equação 4.47) em dois instantes $i = 0$ e $i = 1$. Para $i = 0$ tem se:

$$\begin{aligned} u_1(k-1) &= u_1^* + \alpha_1^{u_1} + \alpha_2^{u_1} \\ u_2(k-1) &= u_2^* + \alpha_1^{u_2} + \alpha_2^{u_2} \\ u_3(k-1) &= u_3^* + \alpha_1^{u_3} + \alpha_2^{u_3} \end{aligned} \tag{4.50}$$

Para i = 1 e definido-se $K_1 = e^{-\lambda\tau}$ e $K_2 = e^{-q\lambda\tau}$, tem-se:

$$\begin{aligned} u_1(k) &= u_1^* + \alpha_1^{u_1}.K_1 + \alpha_2^{u_1}.K_2 \\ u_2(k) &= u_2^* + \alpha_1^{u_2}.K_1 + \alpha_2^{u_2}.K_2 \\ u_3(k) &= u_3^* + \alpha_1^{u_3}.K_1 + \alpha_2^{u_3}.K_2 \end{aligned} \tag{4.51}$$

Para que as restrições da variação de comando sejam respeitadas, a diferença entre $u_n(k)$ e $u_n(k-1)$ deve ser igual a δ_{max}^n. Logo para os três atuadores é possível escrever:

$$\begin{aligned} u_1(k) - u_1(k-1) &= \alpha_1^{u_1}.(K_1 - 1) + \alpha_2^{u_1}.(K_2 - 1) = \delta_{max}^1 \\ u_2(k) - u_2(k-1) &= \alpha_1^{u_2}.(K_1 - 1) + \alpha_2^{u_2}.(K_2 - 1) = \delta_{max}^2 \\ u_3(k) - u_3(k-1) &= \alpha_1^{u_3}.(K_1 - 1) + \alpha_2^{u_3}.(K_2 - 1) = \delta_{max}^3 \end{aligned} \tag{4.52}$$

Como a restrição de comando possui um limite inferior (δ_{min}^n), pode-se reescrever as equações

anteriores em função de um parâmetro p tal que $p_j \in [-1,+1]^2$ para $j \in \{1,..,n_u\}$.

$$\begin{array}{rcl} p_1.\delta^1_{max} & = & \alpha_1^{u_1}.(K_1-1)+\alpha_2^{u_1}.(K_2-1) \\ p_2.\delta^2_{max} & = & \alpha_1^{u_2}.(K_1-1)+\alpha_2^{u_2}.(K_2-1) \\ p_3.\delta^3_{max} & = & \alpha_1^{u_3}.(K_1-1)+\alpha_2^{u_3}.(K_2-1) \end{array} \tag{4.53}$$

Considerando o valor do comando em estado estacionário u^*_n como outra variável a ser encontrada, pode-se definir $u^*_1 = p_4$, $u^*_2 = p_5$ e $u^*_3 = p_6$, logo a equação 4.50 torna-se:

$$\begin{array}{rcl} u_1(k-1) & = & p_4+\alpha_1^{u_1}+\alpha_2^{u_1} \\ u_2(k-1) & = & p_5+\alpha_1^{u_2}+\alpha_2^{u_2} \\ u_3(k-1) & = & p_6+\alpha_1^{u_3}+\alpha_2^{u_3} \end{array} \tag{4.54}$$

Das equações 4.53 e 4.54 é possível montar um sistema de equações do tipo $Mx = b$, logo:

$$\begin{array}{rcl} u_1(k-1)-p_4 & = & \alpha_1^{u_1}+\alpha_2^{u_1} \\ p_1.\delta^1_{max} & = & \alpha_1^{u_1}.K_1+\alpha_2^{u_1}.K_2 \\ u_2(k-1)-p_5 & = & \alpha_1^{u_2}+\alpha_2^{u_2} \\ p_2.\delta^2_{max} & = & \alpha_1^{u_2}.K_1+\alpha_2^{u_2}.K_2 \\ u_3(k-1)-p_6 & = & \alpha_1^{u_3}+\alpha_2^{u_3} \\ p_3.\delta^3_{max} & = & \alpha_1^{u_3}.K_1+\alpha_2^{u_3}.K_2 \end{array} \tag{4.55}$$

Do sistema de equações 4.55, o vetor das variáveis a serem encontradas é $x = (\alpha_1^{u_1}\alpha_2^{u_1}\alpha_1^{u_2}\alpha_2^{u_2}\alpha_1^{u_3}\alpha_2^{u_3})$ e a matriz M:

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ K_1 & K_2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & K_1 & K_2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & K_1 & K_2 \end{pmatrix} \tag{4.56}$$

A matriz 4.56 é constante pois depende apenas dos parâmetros K_1 e K_2. A matriz *b* é definida:

$$\begin{pmatrix} u_1(k-1)-p_4 \\ p_1\delta^1_{max} \\ u_2(k-1)-p_5 \\ p_2\delta^2_{max} \\ u_3(k-1)-p_6 \\ p_3\delta^3_{max} \end{pmatrix} \tag{4.57}$$

A matriz B por sua vez é dependente dos valores dos parâmetros *p* a serem determinados. Logo o sistema final pode ser reescrito em função dos parâmetros *p* da seguinte maneira $Mx = b(p)$. A solução

desse sistema é dada por:

$$x(p) = M(K_1, K_2)^{-1}.b(p) \tag{4.58}$$

Da equação 4.58 percebe-se que a obtenção dos parâmetros $\alpha_n^{u_n}$, ou seja, $x(p)$, depende única e exclusivamente do valor assumido pela matriz *b*, que por sua vez é dependente dos parâmetros p_i. A matriz 4.56 é constante durante todo o processo de simulação, logo sua inversa pode ser calculada apenas uma vez de maneira *offline* evitando o seu cálculo a cada instante de tempo no código, o que aumentaria a carga computacional do algoritmo de parametrização. Além disso, o cálculo da solução do sistema 4.58 será tão mais complexa quantos mais atuadores existirem no sistema e o número de exponenciais escolhidas para parametrização do comando. Uma vez determinado o valor dos parâmetros $\alpha_n^{u_n}$ a solução do comando parametrizado é dada por:

$$\begin{aligned} u_1(i\tau + t) &= Sat_{u_{min}}^{u_{max}}(p_4 + \alpha_1^{u_1}(p).e^{-\lambda i\tau} + \alpha_2^{u_1}(p).e^{-q\lambda i\tau}) \\ u_2(i\tau + t) &= Sat_{u_{min}}^{u_{max}}(p_5 + \alpha_1^{u_2}(p).e^{-\lambda i\tau} + \alpha_2^{u_2}(p).e^{-q\lambda i\tau}) \\ u_3(i\tau + t) &= Sat_{u_{min}}^{u_{max}}(p_6 + \alpha_1^{u_3}(p).e^{-\lambda i\tau} + \alpha_2^{u_3}(p).e^{-q\lambda i\tau}) \end{aligned} \tag{4.59}$$

Os parâmetros p_i da equação 4.59 são obtidos através da otimização de uma determinada função custo. A função custo descrita aqui é constituída pelo somatório do erro de trajetória, ou seja, a diferença entre a referência passada ao sistema e estados preditos e a penalização do estado final, sendo escrita da seguinte maneira:

$$\hat{p} := argmin[\rho_x. \|X_f\| + \sum_{i=0}^{N} \|Y_{ref} - Y_{pred}\|_{Q_y}^2] \tag{4.60}$$

O primeiro termo da função custo 4.60 refere-se a penalização final de estados ponderada por um escalar $\rho_x > 0$ necessário para reforçar a restrição de estado final. O valor X_f é a diferença entre o estado final predito sobre o horizonte N e o estado final desejado em estado estacionário. O segundo termo é a penalização do erro de trajetória dos estados, onde Y_{ref} é a referência passada para o sistema e Y_{pred} é a saída predita dos estados. A diferença entre estas duas grandezas é ponderada por uma matriz Q_y. Depois de obtida a solução do problema 4.60, o primeiro sinal de controle é aplicado ao sistema de controle durante o período $[k, k+1]$, ou seja:

$$K := u^{(1)}(\hat{p}(.), .) \tag{4.61}$$

4.4 Validação dos controladores

Uma vez obtido todo o desenvolvimento teórico da formulação linear e não linear do controlador preditivo é necessário validar os mesmos tanto em simulações em *software* quanto *hardware*. Logo, nas seções subsequentes será feita uma descrição pormenorizada de como cada etapa do *V-cycle* foi implementada para os testes dos controladores.

4.4.1 Etapa MIL

A primeira etapa de testes tem como objetivo avaliar o comportamento do controlador em vários tipos de cenários, seja passando diferentes referências ao modelo ou modificando parâmetros de sintonia do controlador. Esta etapa fornece a facilidade e rapidez com que estes cenários podem ser simulados até que se encontre um conjunto de valores que forneça um desempenho desejado pelo projetista. As simulações em MIL foram feitas em computador com o *Windows 7*® através do $Matlab^{®}R2015a$.

Para os testes realizados nesta etapa, o código do controlador preditivo foi escrito em linguagem de *script* do próprio *Matlab*. Para o controlador linear foi utilizada a função *quadprog*, ferramenta própria do *Matlab*, para otimização de funções quadráticas sujeitas a restrições. A justificativa para sua escolha decorre da facilidade de uso e por fornecer um meio rápido para validar da estratégia de controle desenvolvida.

Para os testes com o controlador preditivo não linear dois *solvers* foram utilizados. O primeiro foi o algoritmo *interior-point-convex* através da função *fmincon*, outra ferramenta nativa do *Matlab* para otimização de funções custo não convexas e sujeitas a restrições. Da mesma maneira que a função *quadprog*, ela foi escolhida pela facilidade de uso e por serem funções prontas não havendo a necessidade de desenvolvimento de outros algoritmos de otimização, para testes na etapa MIL. O segundo algoritmo utilizado foi o *Sequential Quadratic Programming (SQP)* apresentado por Alamir (2013). Este é um algoritmo iterativo utilizado para otimização de funções não lineares com restrições, baseado em aproximações quadráticas locais da função custo não linear. A utilização do *solver* SQP foi devido ao fato que a função *fmincon* é uma solução fechada e não pode ser utilizada fora do *Matlab*. Logo foi necessário utilizar outro algoritmo capaz de ser embarcado em *hardware*.

O modelo do satélite também foi implementado em linguagem de *script*, ao invés da implementação direta no *Simulink*. Uma vez testados vários cenários e encontrado um conjunto de parâmetros que reflita a necessidade do projeto, prossegue-se para os testes em SIL.

4.4.2 Etapa SIL

Para desenvolvimento do código dos controladores optou-se por escolher a linguagem de programação C/C++, com a utilização da biblioteca *open-source Eigen*. Esta biblioteca implementa uma série de funções para realização da cálculos matriciais (EIGEN, 2018). O seu uso simplifica o desenvolvimento do código pois, evita que o desenvolvedor tenha que reescrever funções auxiliares que não existem nas bibliotecas padrões da linguagem escolhida.

Para o desenvolvimento dos algoritmos de otimização na etapa SIL, para o caso linear, foi utilizado o pacote *qpOASES- quadratic programming online active set strategy*, ferramenta *open-source* desenvolvida em C++ e que, devido a características próprias, é particularmente adequada para aplicações em controle preditivo (FERREAU; POTSCHKA; KIRCHES, 2007–2017). Não foi possível utilizar o *quadprog* pois essa é uma solução proprietária do *Matlab*. Para o caso do controlador preditivo não linear, continuou-se com algoritmo de SQP. Este algoritmo encontra-se desenvolvido em linguagem de *script* do *Matlab* no livro de Alamir (2013). Logo, na etapa SIL ele foi convertido para C/C++ com o intuito de ser

embarcado em *hardware*.

Neste trabalho os códigos em C/C++ não foram validados dentro do ambiente do *Matlab*. O motivo é que os mesmos teriam que passar por grandes alterações para serem colocados no formato *MEX-file*, o único aceito pelo *Matlab*, para poderem ser compilados. Além disso, o *Matlab* não oferece suporte a biblioteca *Eigen*. Então na etapa SIL somente os códigos dos controladores foram desenvolvidos e sua validação foi feita posteriormente na etapa PIL.

4.4.3 Etapa PIL

Após o desenvolvimento dos códigos em C/C++ na etapa anterior faz-se necessário verificar se os mesmos estão funcionando corretamente, ou seja, se apresentam características semelhantes àquelas da etapa MIL. Para realização dos testes em PIL, uma pequena plataforma composta por dois componentes foi desenvolvida. Ela consiste em: um computador com o sistema operacional *Windows 7*® onde está instalado o $Matlab^{®}R2015a$, e um *hardware* onde os códigos desenvolvidos na etapa SIL podem ser embarcados. Um esquemático desta plataforma pode ser vista na figura 4.3. O computador tem o papel de simular o modelo do satélite e o *hardware* é responsável por executar a estratégia de controle desenvolvida. Esta arquitetura é genérica o suficiente para que sejam testados vários tipos de controladores e modelos diferentes. O *hardware* aqui utilizado é o microcontrolador *BeagleBone Black*.

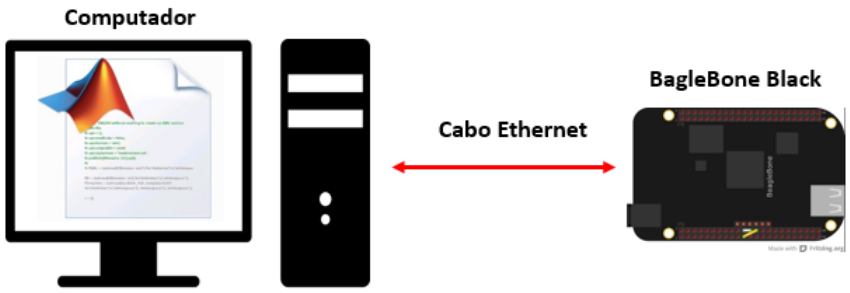


Figura 4.3: Plataforma PIL para validação do código C/C++ desenvolvido na etapa SIL.

O computador e o microcontrolador estão conectados fisicamente por meio de um cabo RJ-45. Este meio físico de comunicação foi escolhido por não necessitar de nenhum outro *hardware* adicional, apenas uma porta do tipo *LAN* em cada um dos equipamentos, que já vem de fábrica na maioria dos equipamentos modernos. O protocolo utilizado para troca de dados é o *Transmission Control Protocol/Internet Protocol* (TCP/IP) (FALL; STEVENS, 2011). Este protocolo tem como vantagens: a verificação de erros durante a comunicação, garantia da entrega e recebimento dos pacotes de dados entre os elementos da plataforma; e mais ainda seu poder de interoperabilidade, traduzido como a capacidade de manter a comunicação entre diferentes equipamentos e computadores com diferentes sistemas operacionais.

A troca de dados é feita através de um *socket*, programa que permite a comunicação entre processos diferentes em máquinas diferentes. Do lado do microcontrolador foi desenvolvido um *socket* em C/C++ e, para o computador, foi criado um *socket* dentro do ambiente do *Matlab* em linguagem de *script*. Uma das variáveis dentro de cada um destes programas é o endereço da máquina de envio/recebimento. Os dois

equipamentos devem possuir a mesma máscara sub-rede e IP's distintos. Dessa maneira os equipamentos podem trocar dados conhecendo apenas o endereço IP um do outro. A figura 4.4 mostra um esquema mais detalhado de como ocorre a simulação e a troca de dados entre o computador e a *BeagleBone Black*.

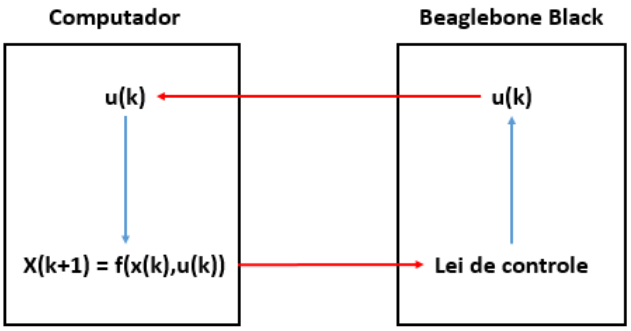


Figura 4.4: Detalhe da simulação e troca de dados entre o computador e a *Beaglebone* utilizando o protocolo TCP/IP.

Da figura anterior percebe-se que o computador recebe o comando $u(k)$ calculado pela *BeagleBone*. Este comando será aplicado no modelo do satélite, representado pela função $x(k+1) = f(x(k), u(k))$, que determina a passagem do estado $x(k)$, para o estado $x(k+1)$ quando submetido a entrada $u(k)$. Este modelo é escrito em linguagem de *script* do *Matlab*. Uma vez calculado o próximo estado $x(k+1)$ ele é enviado de volta a *BeagleBone*, onde é utilizado pela lei de controle para o cálculo do próximo comando $u(k)$. Este ciclo é então repetido até o fim da simulação.

Apesar da figura 4.4 representar duas linhas em vermelho indicando o envio e recebimento de estados e comando, estes dados são trocados através do mesmo canal físico (cabo RJ-45). Isto é possível por uma sincronia entre o computador e a *BeagleBone*. Este é um aspecto criado pelo protocolo TCP/IP. Podendo ser explicado da seguinte maneira: Uma vez que a *BeagleBone* envia o comando ele ficará esperando o recebimento dos estados do sistema. Da mesma maneira que, uma vez que o computador enviar os estados para a *BeagleBone* ele ficará esperando o recebimento do sinal de controle. Isto faz com que não exista perda de dados entre os elementos da arquitetura. Outro detalhe é que esta simulação não ocorre em tempo real.

Ao fim da simulação os dados gerados são comparados com os resultados da etapa MIL. Caso haja concordância entre as simulações, o controlador está pronto para ser testado na plataforma HIL. É importante salientar que as simulações não serão exatamente iguais. Um fator é a diferença entre o tipo de *solver* utilizado nas simulações em MIL e PIL. Ainda que fosse utilizado o mesmo *solver* os equipamentos constituintes da plataforma possuem arquiteturas diferentes gerando divergências quanto a representação numérica utilizada por cada equipamento. O computador por exemplo tem uma precisão numérica de 64 *bits* enquanto a *BeagleBone*, de 32 *bits*.

Outro objetivo das simulações em PIL é determinar quais são os limites máximos e mínimos alcançados pelos estados do sistema. Isto é necessário pois na etapa HIL os canais de leitura analógica presentes na *BeagleBone* precisam ser calibrados corretamente e esta calibração depende destes valores.

4.4.4 Etapa HIL

A última etapa de validação corresponde aos testes na plataforma HIL. Primeiramente, para um melhor entendimento de como a plataforma funciona pode-se observar uma abstração da mesma por meio do diagrama de blocos na figura abaixo.

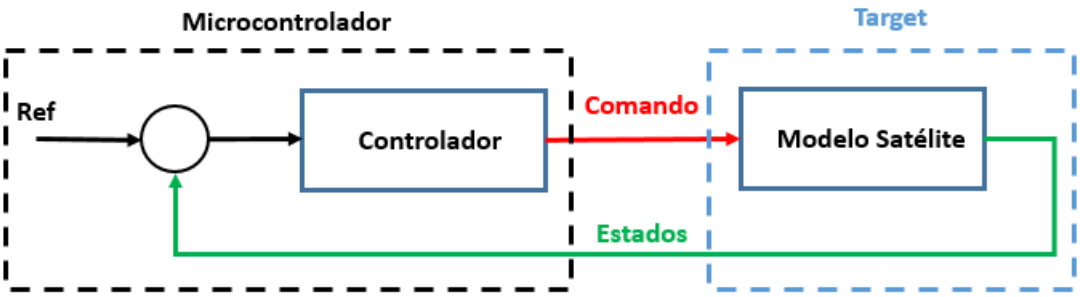


Figura 4.5: Diagrama de blocos da arquitetura HIL.

De maneira semelhante a etapa PIL, a figura 4.5 mostra que a arquitetura HIL está dividida entre dois componentes: o microcontrolador e um computador chamado de *target*.

Dentro do microcontrolador está a lei de controle preditivo e também a definição da referência que será passada ao sistema que está sendo controlado. A lei de controle aqui embarcada calcula o sinal de comando que é o valor das acelerações angulares da roda de reação, i.e., $\dot{\Omega}_1$, $\dot{\Omega}_2$ e $\dot{\Omega}_3$. A máquina *target* é um computador responsável por simular o modelo da plataforma em tempo real. O *target* recebe o sinal de comando calculado pelo microcontrolador e retorna os estados do sistema, ou seja, os ângulos de *euler* θ_1, θ_2 e θ_3 e as velocidades angulares da plataforma ω_1, ω_2 e ω_3. O que distingue a simulação em PIL da simulação HIL é que nesta última o modelo da plataforma é simulado em tempo real e a troca de comando e estados são feitos por canais de comunicação distintos.

A versão final da plataforma utilizada neste trabalho e com uma descrição pormenorizada dos equipamentos utilizados é mostrada na figura 4.6. Vale ressaltar que a motivação para construção de uma plataforma deste tipo advém do fato de que as opções comerciais, como por exemplo equipamentos da *dSPACE* ou *National Instruments* presentes no mercado são caras (MINA et al., 2016), seja pelo *hardware* responsável por executar o controlador ou pelo uso de plataformas de mancal aerostático.

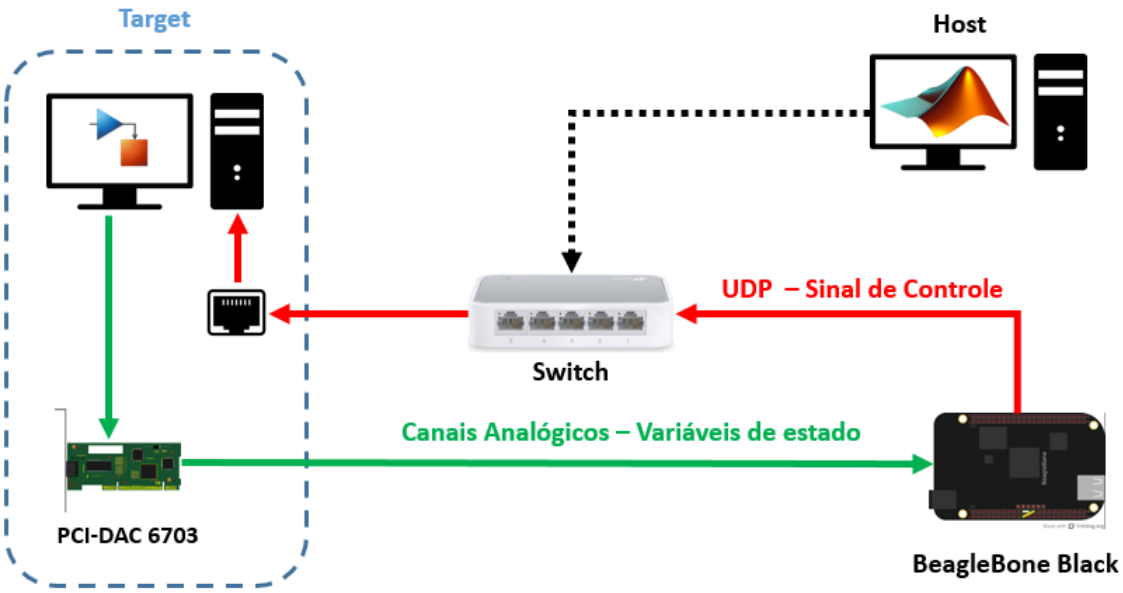


Figura 4.6: Plataforma HIL para validação de controladores.

Os equipamentos constituintes desta plataforma são os computadores *host* e *target*; o *switch*, a placa de aquisição de dados PCI-DAC 6703 e o microcontrolador. Nas próximas subseções será feita uma descrição mais detalhada de cada destes equipamentos.

4.4.4.1 Máquinas *Host* e *Target*

O *host* é um computador que executa o sistema operacional *Windows 7*® onde está instalado o $Matlab^{®}R2015a$. O *host* é responsável pela execução de uma *toolbox* do próprio *Matlab* chamada *Simulink Real-Time (SLRT)*. Esta ferramenta permite criar aplicações em tempo real a partir do *Simulink* (MATHWORKS, 2018a). A nomenclatura utilizada para nomeação das máquinas *host* e *target* é a mesma utilizada pela *MathWorks* para explicar o funcionamento do SLRT. Estas máquinas estão conectadas fisicamente através do *switch* com o uso de cabos RJ-45.

No *host* é feita a criação do modelo da plataforma de satélites por meio de blocos existentes no *Simulink*. A *toolbox* do SLRT apresenta uma séries de blocos dotados de características de execução em tempo real. Depois da etapa de criação do modelo, passa-se a etapa de envio ao *target*. Para ser enviado à máquina *target*, o modelo construído no *Simulink* precisa ser convertido para um formato que seja aceito pela máquina *target*. Essa conversão é feita por uma funcionalidade do *Simulink* chamada de *Build Model*. Ela converte o modelo criado em *Simulink* no formato *.slx* para código compilado em C/C++. Caso não haja nenhum erro nesta etapa o modelo é enviado automaticamente para a máquina *target*. O procedimento desde a criação do modelo até envio do mesmo segue os seguintes passos:

1. Inicialização da máquina *host*.
2. Geração de um disco de *boot* contendo o *kernel* do SLRT.
3. Inicialização da máquina *target* a partir do disco de *boot* da etapa anterior.

4. Abertura de uma conexão TCP/IP entre as duas máquinas, realizado a partir do *host*.
5. Criação do modelo na máquina *host* por meio de diagrama de blocos no *Simulink*.
6. Build Model - Conversão do modelo em .slx para código compilado C/C++.
7. Envio do modelo para o *target*.

Após a etapa de envio do modelo, a máquina *host* passa a atuar no monitoramento da simulação entre o *target* e o microcontrolador. Podendo iniciar ou pausar a simulação e recuperando os dados da mesma.

O *target* por sua vez é computador dedicado, responsável por simular o modelo do satélite em tempo real. Para isto a ele deve executar um sistema operacional em tempo real (RTOS). Uma das funcionalidades presentes no SLRT é que ele vem com um *kernel* em tempo real para ser utilizado na máquina *target*. O *kernel* é um programa de computador responsável por intermediar a comunicação entre os dispositivos de *hardware* e as aplicações ativas no computador durante seu funcionamento. A função de gerenciamento do *kernel* é mostrada na figura abaixo:

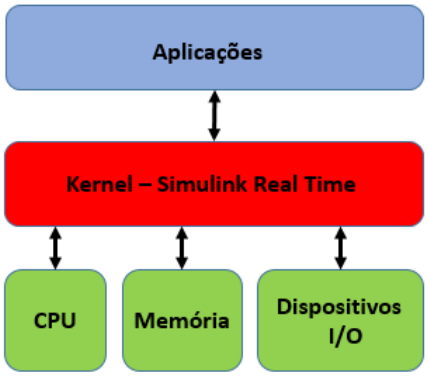


Figura 4.7: Função do *kernel* como gerenciador de recursos entre as aplicações de *software* e *hardware* em um computador.

Por ter controle geral do computador é responsabilidade do *kernel* gerenciar as tarefas para que ocorram em determinados períodos de tempo bem definidos. A denominação de sistema operacional em tempo real não implica necessariamente em rapidez, mas apenas que o sistema é capaz de responder a eventos em um tempo bem definido. Abbott (2011) define que um sistema em tempo real é aquele que em que a correção dos cálculos não depende apenas da correção lógica do cálculo, mas também do momento em que o resultado é produzido. Se as restrições de tempo do sistema não forem atendidas, ocorrerá falha no sistema.

A execução em tempo real é obtida através da minimização da carga computacional que geralmente existe em um sistema operacional não dedicado (WAHBA et al., 2016), como por exemplo no caso do sistema operacional *Windows*. O *kernel* fornecido pelo SLRT é uma versão de um RTOS proprietário conhecido como *On Time RTOS-32*.

Este *kernel* é utilizado para inicializar a máquina *target* que após esta etapa estará pronta para receber os modelos desenvolvidos no *host*. Conectada à máquina *target* existem dois periféricos que são responsáveis pela troca de dados do *target* com os outros equipamentos existentes na plataforma. O pri-

meiro periférico é a placa de aquisição de dados PCI-DAC 6703, responsável pelo envio dos estados do sistema que está sendo simulado para a *BeagleBone Black*. O segundo periférico é uma placa de rede (*intelbras PEG 232 PCI Express*) dedicada, que recebe o valor da variável de controle, calculada pela *BeagleBone Black*.

Um dos importantes parâmetros de configuração da máquina *target* é o seu período de amostragem. Como o seu objetivo é simular o modelo dinâmico de um sistema para que ele seja o mais próximo possível do modelo contínuo, este período de amostragem deve ser suficientemente pequeno. O próprio SLRT impõe um limite mínimo e máximo, 8μs e 10s, respectivamente, para a escolha do período de amostragem do *target*. Para o modelo da plataforma de testes de satélites o menor valor possível alcançado foi de 200μs. Valores abaixo de 200μs causam o efeito de *CPU overload* no *target*, indicando que o computador foi incapaz de processar as operações requeridas durante o período de amostragem escolhido.

4.4.4.2 *Switch*

Um dos elementos centrais desta plataforma é o *switch*. O seu objetivo é criar uma rede local, estabelecendo a comunicação entre o *target*, o *host* e o microcontrolador. O modelo aqui utilizado é o *D-Link 8 Port 10/100 switch*, figura 4.8.

Figura 4.8: Switch 8 portas.

Como explicado anteriormente para ser colocada em completo funcionamento a plataforma passa por dois estágios, um de preparação do modelo a ser enviado ao *target* e o segundo, a simulação da malha de controle propriamente dita. O *switch* atua diretamente nos dois estágios. Em um primeiro momento serve como ponte de comunicação para que o modelo seja enviado do *host* para a máquina *target*. Terminada a passagem do modelo, e inicializada a simulação, o *switch* irá intermediar a comunicação entre o *target* e o microcontrolador. Neste estágio sua função é receber o sinal de controle calculado pelo microcontrolador e enviá-lo ao *target*.

4.4.4.3 Microcontrolador

O microcontrolador é o elemento responsável pela execução da estratégia de controle preditivo. O *hardware* aqui utilizado é a *BeagleBone Black (Rev C)* (BB). A BB é uma placa de desenvolvimento *open-source* de baixo custo. Ela executa uma versão *GNU/Linux* e possui várias funcionalidades que um

computador comum possui, além de GPIO's que possibilitam a comunicação da placa com periféricos externos, como sensores. A BB pode ser vista na figura 4.9.

Figura 4.9: *BeagleBone (Rev C).*

Ela foi escolhida por apresentar baixo custo e por contar com uma série de elementos capazes de serem interfaceados com periféricos externos. Existem outras opções no mercado semelhantes a BB e que podem ser utilizadas para o mesmo propósito, algumas delas são: *Raspberry Pi*, *ODROID*, *Intel Galileo* e *ASUS Tinker Board*. As especificações da BB são mostradas na tabela 4.1.

Tabela 4.1: Especificações *BeagleBone Black (Rev C)*

	BeagleBone Black (Rev C)
Processador	AM3358 ARM Cortex-A8
Velocidade processador	1 Ghz
Qtd. de pinos analógicos	7
Qtd. de pinos digitais	65 (3.3V)
Memória	512MB DDR3 (800 MHz x 16)
Interfaces suportadas	4x UART, 8x PWM, LCD, GPMC, MMC1, 2x SPI, 2x I2C, Conversor A/D, 2xCAN BUS, 4 Timers

Uma das características úteis da BB é a existência de sete entradas analógicas que podem ser interfaceadas com a placa de aquisição de dados PCI-DAC 6703 para leitura de estados do sistema. Caso o modelo a ser simulado na plataforma tenha um número de estados superior a sete podem-se utilizar técnicas de estimação para sobrepassar esta limitação física da BB.

A BB executa duas funções na plataforma HIL. A primeira é de fazer a leitura dos estados do modelo, provenientes da máquina *target*, através de canais analógicos, indicados em verde na figura 4.6. A segunda é executar a estratégia de controle preditivo embarcada e enviar o valor da variável de comando para a máquina *target*, indicado em vermelho na figura 4.6, através de um cabo RJ-45. O envio de comando é feito através da utilização do protocolo de comunicação *User Datagram Protocol* (UDP).

O protocolo UDP não é orientado a conexão. Isto significa que o pacote pode ser enviado a qualquer momento, sem aviso prévio, negociação ou preparação (MATASARU, 2013). Ele não adota nenhum

mecanismo interno para garantir que o pacote chegará ao destino ou chegará na ordem correta de envio, diferentemente do TCP/IP. Esta aparente desvantagem torna o UDP mais leve e o faz um bom candidato para aplicações que exijam operações em tempo real (BERGSTRÖM; GÖRANSSON, 2016). Uma de suas grandes vantagens é a velocidade de comunicação e de comunicação em *broadcast*, i.e, ele é capaz de enviar o mesmo pacote de dados para todos os equipamentos que estão na mesma rede.

É possível encontrar outros trabalhos que utilizam o protocolo UDP para comunicação entre elementos de uma plataforma HIL, como em (SHANGGUAN et al., 2016), (MATASARU, 2013), (BERGSTRÖM; GÖRANSSON, 2016) e (WAHBA et al., 2016).

4.4.4.4 PCI-DAC 6703

Conectada à máquina *target* está a placa de aquisição de dados PCI-DAC 6703, do fabricante *Measurement Computing*, visto na figura 4.10. A função desta placa é receber os estados do modelo que está sendo simulado no *target* e enviá-los a BB.

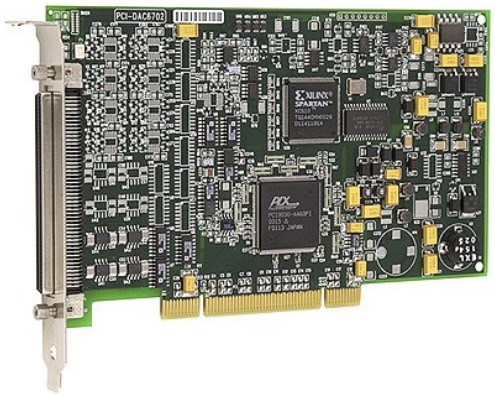


Figura 4.10: Placa de aquisição de dados PCI-DAC 6703.

Esta placa possui 16 saídas analógicas e conversores DAC de 16 *bits*. As saídas analógicas operam numa faixa de tensão de [-10,+10]V. O *Simulink* disponibiliza uma série de blocos de suporte a esta placa. Através destes blocos é possível construir modelos no *Simulink* que enviam e recebem dados de periféricos externos, através da PCI-DAC 6703. O bloco de saída analógica responsável por enviar os estados do modelo que está sendo simulado no *target* para serem lidos pela BB é mostrado na figura abaixo:

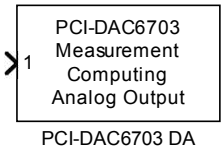


Figura 4.11: Bloco de saída analógica PCI-DAC 6703.

Este bloco recebe os estados do modelo que está sendo simulado no *Simulink* e os reenvia diretamente para a BB. Fisicamente os canais de saída analógica da PCI-DAC 6703 estão conectados diretamente as GPIO's de entrada analógica da BB, que operam na faixa de tensão entre [0,+1.8]V.

4.4.4.5 Conversão entre unidades de engenharia e tensão

A diferença na faixa de operação de tensão entre a BB e PCI-DAC6703 deve ser levada em consideração quando se projeta uma simulação na plataforma HIL. Caso a conversão não seja feita corretamente a estratégia de controle irá funcionar de maneira inesperada. As chamadas unidades de engenharia, são os valores nominais das variáveis do modelo que se está trabalhando, como por exemplo, ângulo (rad), velocidade angular ($\frac{rad}{s}$) e aceleração angular ($\frac{rad}{s^2}$). Estes valores precisam ser convertidos para um valor correspondente de tensão em volts (V), antes de serem enviados ao bloco de saída analógica (figura 4.11). A operação de conversão é feita no *Simulink* pelo seguinte conjunto de blocos:

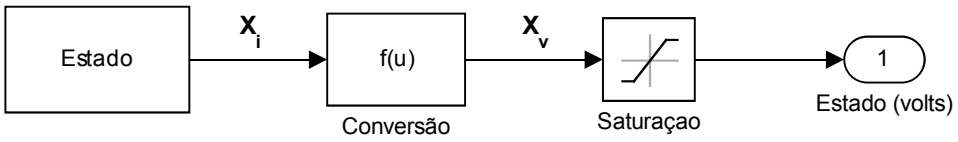


Figura 4.12: Conversão de unidade de engenharia para tensão.

O bloco de conversão recebe o valor do estado em unidades de engenharia, indicado por x_i e os converte para tensão em *volts*, indicado por x_v, utilizando a seguinte relação:

$$x_v = (1.8) * \left(\frac{x_i - x_{i(min)}}{x_{i(max)} - x_{i(min)}} \right) \tag{4.62}$$

O valor de saída do bloco x_v corresponde ao valor do estado i em volts. Os valores $x_{i(min)}$ e $x_{i(max)}$ são os limites mínimo e máximo, respectivamente do estado i, em unidades de engenharia. Este valores, como explicado anteriormente, são obtidos através das simulações em PIL. Uma consideração acerca destes valores é que quando forem ser utilizados na etapa HIL sugere-se acrescentar um fator de correção de 10% tanto no valor máximo quanto no mínimo. Isto deve-se ao fato de que nas simulações em HIL estão presentes variações na leitura dos estados feitas pela BB o que não acontece na etapa PIL.

Após o bloco de conversão, por medida de segurança, é utilizado um bloco de saturação com limites entre 0 e 1.8V, impedindo o envio de uma tensão que esteja fora dos limites operacionais da BB. Uma vez que estes valores são recebidos pela BB, os valores são reconvertidos para unidades de engenharia para que o cálculo da lei de controle funcione corretamente.

4.5 Implementação do controlador e do modelo na plataforma HIL

Definidos todos os elementos da plataforma e os elementos que devem existir na simulação em HIL as próximas subseções mostrarão como o modelo da plataforma foi embarcado na máquina *target* e de que maneira o controlador implementado na *BeagleBone* se comunica com este modelo.

4.5.1 Implementação do modelo da plataforma de satélites no Simulink Real-Time

O modelo a ser simulado em tempo real na máquina *target* é o modelo não linear da plataforma. Este deve ser utilizado por ser o mais próximo da realidade para que todas as dinâmicas do satélite estejam presentes na simulação. O conjunto modelo da plataforma com as interfaces de comunicação com a *BeagleBone* e a placa de aquisição de dados PCI-DAC 6703 é mostrado na figura abaixo:

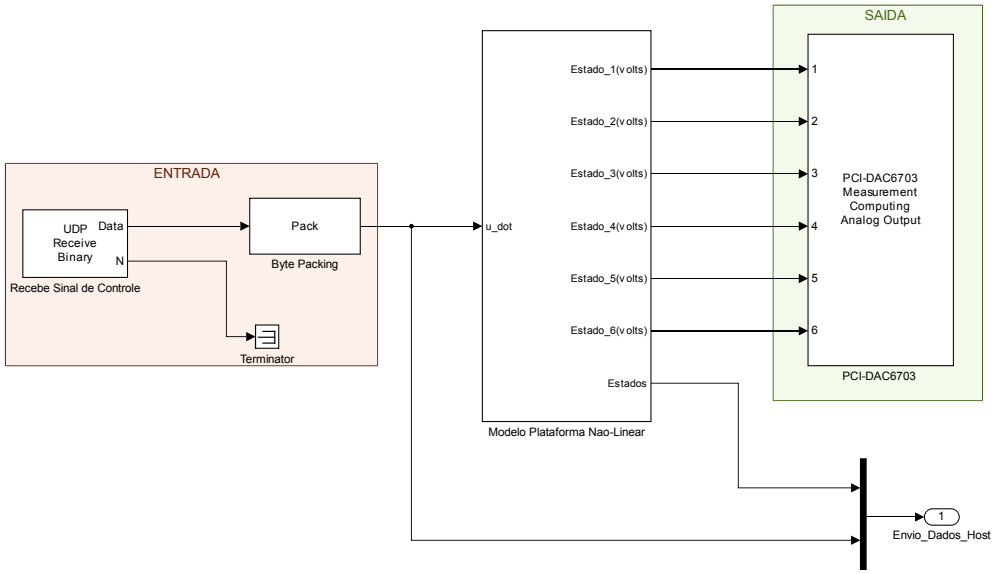


Figura 4.13: Blocos utilizados no modelo rodando no target.

No modelo da figura 4.13 estão destacadas duas áreas: uma de entrada (vermelha) e outra de saída (verde). O subsistema em vermelho indica os blocos utilizados para o recebimento da variável de controle pela *BeagleBone* e a verde indica a passagem dos estados da plataforma de satélite para o bloco de saída analógica da PCI-DAC 6703. Entre as duas áreas está o modelo não linear da plataforma de satélites. O elemento responsável pelo recebimentos de pacotes UDP é o bloco *UDP Receive Binary*. Nele é possível especificar o tamanho do vetor de dados recebidos e o endereço IP da origem dos dados. Este bloco possui características de funcionamento em tempo real.

O modelo da plataforma foi implementado através de uma *S-function*, que é uma linguagem de programação de descrição de blocos no *Simulink*. Eles podem ser escritos em C, C++ ou *Fortran*. A implementação do modelo da plataforma pode ser visto na figura abaixo.

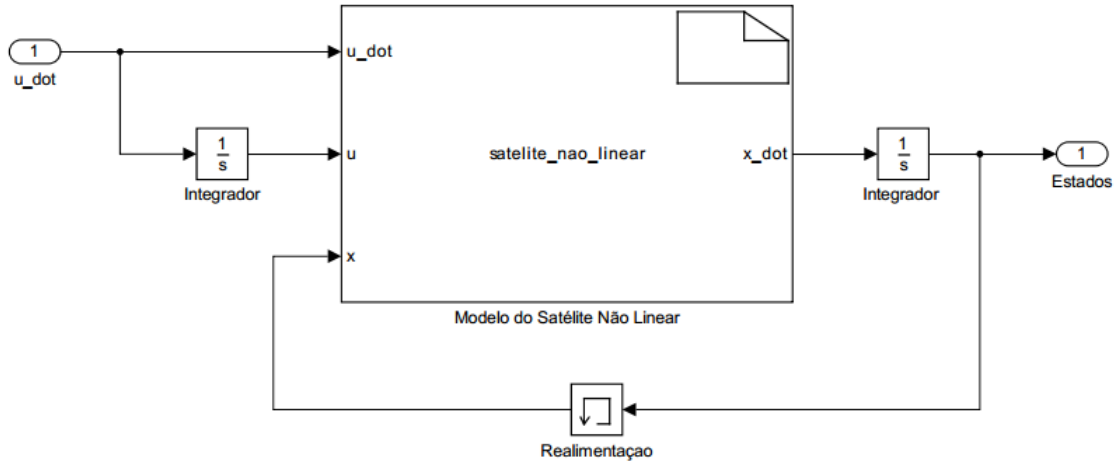


Figura 4.14: Modelo não linear da plataforma de satélite em S-Function.

O conjunto de equações que está no domínio contínuo foi discretizada através do método de *Runge-Kutta* de 4 ordem. O bloco que representa o modelo do satélite (figura 4.14) tem como entradas: o vetor de controle das acelerações da roda de reação, i.e., $\dot{\Omega}_1$, $\dot{\Omega}_2$ e $\dot{\Omega}_3$. Estes valores são provenientes do controlador preditivo; e as velocidades angulares, que são obtidas através da integração da aceleração das rodas de reação, i.e., $u = (\Omega_1, \Omega_2, \Omega_3)$. Por fim existe a realimentação completa de estados, ou seja, considera-se que todos os estados são medidos, onde $x = (\theta_1, \theta_2, \theta_3, \omega_1, \omega_2, \omega_3)$.

Ao modelo desenvolvido da figura 4.14 deve-se acrescentar os blocos de conversão de unidades. A representação final do modelo pode ser visto na figura 4.15.

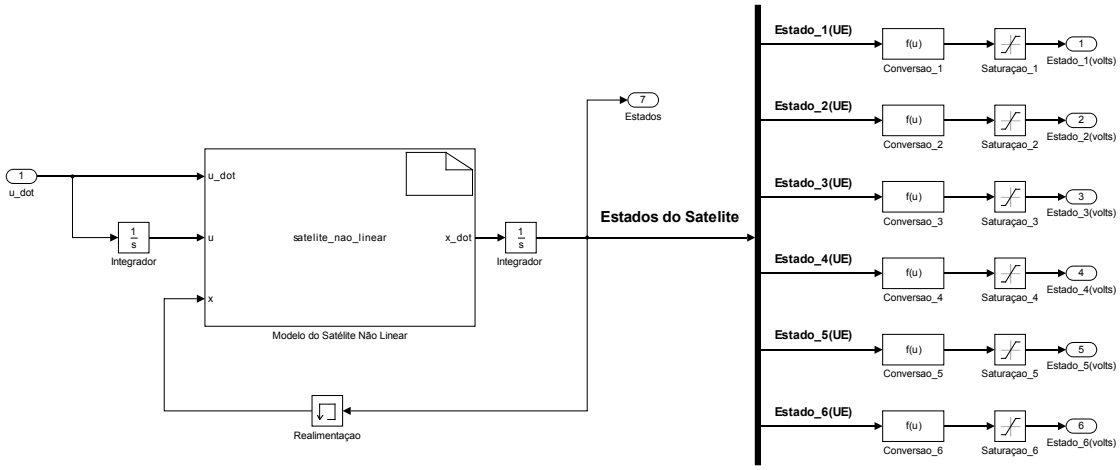


Figura 4.15: Modelo do satélite com blocos de conversão analógica

Os estados do sistema são passados a uma estrutura chamada de *demux* responsável por separá-los e enviar cada um para seu respectivos blocos de conversão entre unidades de engenharia e tensão. Destaca-se que o diagrama de blocos representado pelas figuras 4.13 e 4.15 podem ser reutilizados em outras simulações. Devendo somente se preocupar em modificar o modelo e determinar os valores máximos e mínimos dos estados do sistema.

4.5.2 Algoritmo de controle embarcado na *BeagleBone*

Esta subseção tem como por objetivo explicar a estrutura do algoritmo do controlador preditivo dentro da *BeagleBone*. A estrutura final do controlador pode ser visto, na figura 4.16. Optou-se por não representar os outros elementos da plataforma para simplificar a explicação das funções desempenhadas pelo algoritmo.

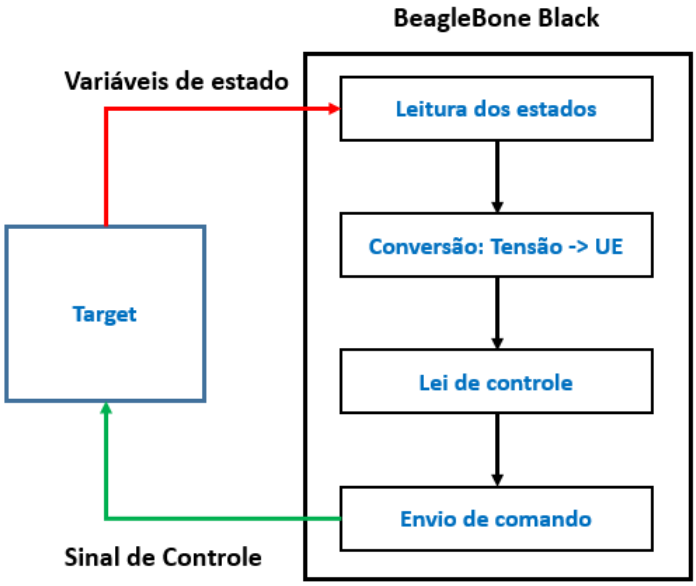


Figura 4.16: Etapas do algoritmo de controle embarcado na BB

O algoritmo de controle da figura 4.16 está baseado em 4 funções principais. A primeira função corresponde a leitura dos estados provenientes da máquina *target*. Estes valores estão em unidades de *volts*. Os valores dos estados são passados então para a função de conversão. Aqui é feita a reconversão das unidades em *volts* para unidades de engenharia (UE), baseada nos limites máximos e mínimos dos estados, obtidos nas simulações em PIL. Depois da conversão os estados são passados para o bloco responsável por executar a lei de controle escolhida, seja MPC ou NMPC. A saída deste bloco corresponde aos sinais de comando. Por fim estes valores são passados a função responsável por enviar os comandos para o *target* através do protocolo UDP. Este ciclo é então repetido até que a simulação chegue ao fim.

Capítulo 5

Resultados

Neste capítulo serão apresentados os resultados da simulações utilizando-se o controlador preditivo, linear (MPC) e não linear (NMPC). A primeira análise deste capítulo refere-se a uma comparação entre as entradas degrau e degrau filtrado. O objetivo é mostrar como o desempenho do sistema pode ser alterada através da modificação do tipo de entrada fornecida ao sistema. Para isto será utilizado apenas o modelo linear da plataforma pois o objetivo é demonstrar apenas a influência da referência nas simulações.

Outra análise é determinar como a resposta do controlador preditivo é modificada à medida que o horizonte de predição (N) é variado. Partindo-se dos resultados anteriores será feita uma discussão de como a parametrização pode ser útil para redução no tempo de cálculo do problema de otimização, mostrando as diferenças entre o controlador parametrizado e o não parametrizado.

Ainda acerca da parametrização será discutido como os parâmetros de sintonia do controlador parametrizado (t_r, α e número de exponenciais (n_e)) modificam a resposta do sistema. Dos testes e análises realizadas anteriormente será escolhido um conjunto de parâmetros e partir dele será feita a validação do controlador passando por todas as etapas do *V-cycle*.

Para justificar a escolha dos controladores preditivos linear e não linear eles serão comparados com os controladores LQR e SDRE respectivamente. E por fim serão mostrados os resultados do tempo necessário para execução dos algoritmos MPC e NMPC embarcados em *hardware*.

5.1 Descrição da Plataforma

O computador utilizado para realização das simulações nas etapas MIL, PIL e HIL possui as seguintes especificações:

O *software* de simulação utilizado foi o *Matlab2015a*. O compilador utilizado no *Matlab2015a* para gerar o código executável do modelo descrito em *Simulink* para envio ao *target* é o *Microsoft Windows SDK 7.1*. Os códigos dos controladores em C/C++ foram compilados na *BeagleBone* utilizando o compilador *open-source GNU Compiler Collection (GCC)*. Por sua vez a *BeagleBone* está executando o sistema operacional *Debian GNU/Linux 7 (wheezy)*.

Tabela 5.1: Especificações do computador utilizado para simulações em *Matlab/Simulink*.

Sistema Operacional	Windows 7 Professional (64 bits)
Processador	Intel(R) Core(Tm) i3 CPU-M-350
Velocidade do Processador	2.27GHz
Número de Núcleos	2
Memória Instalada	3 GB

5.2 Parâmetros físicos do modelo

Os momentos de inércia da plataforma e das rodas de reação obtidos de Gonzales (2009) estão indicados na tabela abaixo:

Tabela 5.2: Momentos de Inércia da Plataforma de Testes de Satélites.

Parâmetro	**Valor**	**Unidade**
I_{11}	1.17	$kg.m^2$
I_{22}	1.17	$kg.m^2$
I_{33}	2.13	$kg.m^2$
I_w	1.8×10^{-3}	$kg.m^2$

O modelo da plataforma de satélites possui três atuadores logo $Q_u \in \mathbb{R}^{n_u \times n_u}$, onde $n_u = 3$. O sistema possui seis estados e todos eles são penalizados logo, $Q_y \in \mathbb{R}^{n \times n}$, onde $n = 6$. O período de amostragem utilizado para discretização das matrizes lineares A e B do sistema contínuo da plataforma foi de $100ms$. As restrições para o modelo da plataforma de testes de satélites e que serão utilizadas nas simulações estão descritas na tabela 5.3.

Tabela 5.3: Restrições de comando e variação de comando.

Parâmetro	**Mínimo**	**Máximo**	**Unidade**
u	-1.5	$+1.5$	$\frac{rad}{s^2}$
δ	-1.0	$+1.0$	$\frac{rad}{s^2}$

As restrições dos atuadores u^{min} e u^{max} delimitam o intervalo de acelerações as quais as rodas de reação podem fornecer. Estes valores foram escolhidos para estarem no limite de torque máximo fornecido pela roda de reação. Apesar de não terem sido obtidos diretamente de uma roda de reação real, são os valores mais próximos daqueles empregados por Gonzales (2009). Os parâmetros δ^{min} e δ^{max} limitam a variação da variável de comando *u* em relação ao tempo, estes parâmetros também são dependentes do atuador. Tanto a restrição de comando quanto sua variação são iguais para os três atuadores.

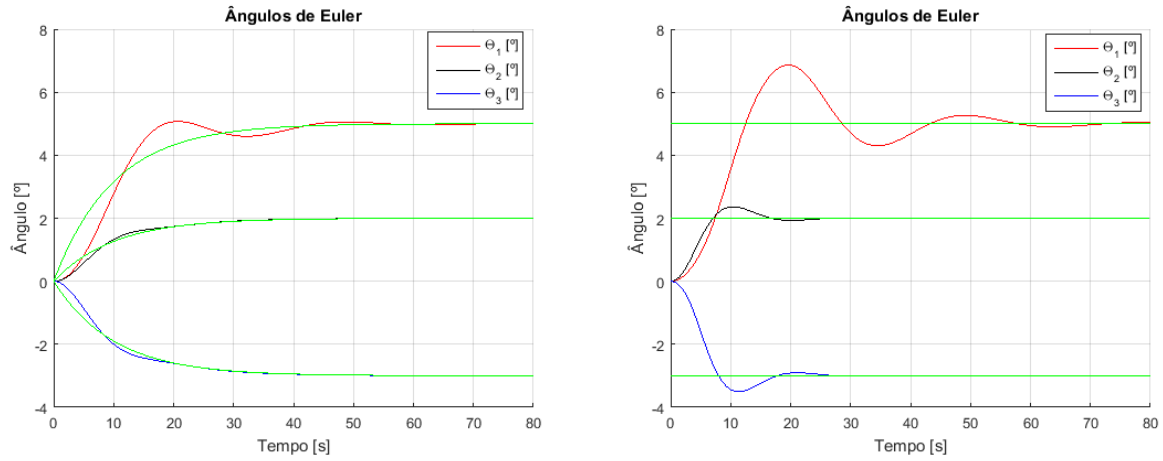
5.3 Comparação Degrau e Degrau Filtrado

Um dos primeiros pontos a ser discutido e que é comum a todas as simulações que serão feitas para o MPC e NMPC diz respeito ao tipo de referência que será utilizada. Esta seção se propõe a analisar o comportamento da dinâmica do sistema quando submetido a um tipo de referência chamada de degrau

filtrado. Diferentemente do degrau não filtrado, este tipo de degrau é suavizado com o objetivo de reduzir possíveis *overshoot's* por não exigir uma mudança brusca no início das simulações. Este tipo de referência pode ser modelado da seguinte maneira:

$$y = A(1 - e^{\frac{-3\tau.i}{t_{ref}}}) \tag{5.1}$$

Onde A é a amplitude do degrau, τ o período de amostragem e t_{ref} é o tempo de resposta desejado para o sistema atingir 95% da amplitude A do degrau. Para determinar o efeito deste tipo de referência na resposta dinâmica do sistema, duas simulações foram realizadas em *Matlab*. A primeira simulação corresponde ao teste feito com a referência filtrada e o segundo cenário com a referência não filtrada. As referências são passadas apenas para os ângulos de *euler* θ_1, θ_2 e θ_3. Para as duas simulações utilizou-se o modelo linear da plataforma de satélites e um controlador preditivo linear com os seguintes parâmetros de sintonia, $N = 40$, $Qy = diag(1e8, 1e8, 1e8, 1, 1, 1)$ e $Qu = diag(1e4, 1e4, 1e4)$. Estes parâmetros foram mantidos contantes nas duas simulações, alterando-se apenas o tipo de referência passada ao sistema. As referências filtradas e não filtradas possuem as mesmas amplitudes de $\theta_1 = 5°$, $\theta_2 = 2°$ e $\theta_3 = -3°$ (todas mostradas em verde nas simulações) e para o caso filtrado a constante $t_{ref} = 30$ foi utilizada. A figura abaixo mostra o resultado comparativo entre os ângulos de *euler* para os dois tipos de entradas.

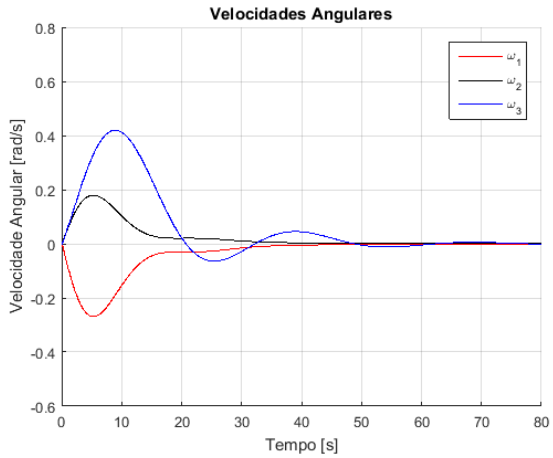


(a) Ângulos de euler - Degrau filtrado.

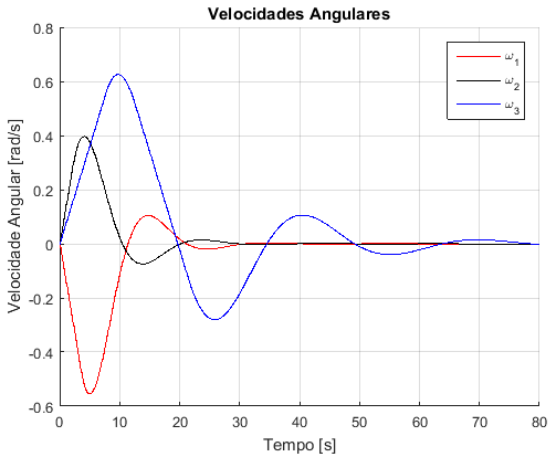
(b) Ângulos de euler - Degrau não filtrado.

Figura 5.1: Comparação entre as respostas dos ângulos de *euler* para degrau filtrado $vs.$ não filtrado.

Da figura 5.1 é possível determinar que o ângulo θ_1 teve uma redução na amplitude do *overshoot* no tempo de simulação $t = 20s$ para a entrada filtrada. Para os ângulos θ_2 e θ_3 houve também redução do *overshoot*. A próxima figura mostra os resultados para as velocidades angulares da plataforma, i.e, ω_1, ω_2 e ω_3.



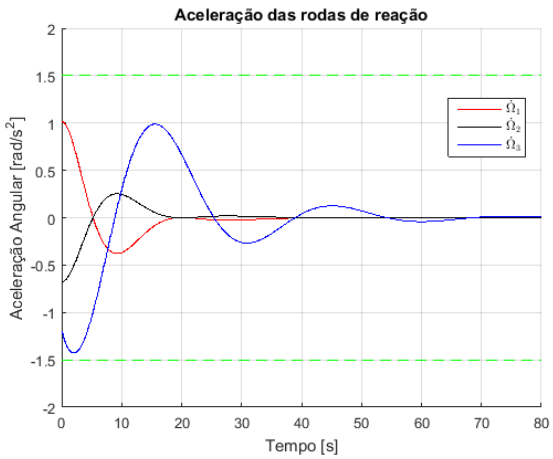
(a) Velocidades angulares: Degrau filtrado.



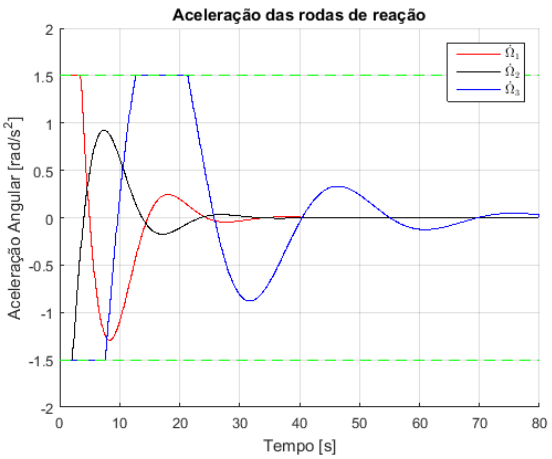
(b) Velocidades angulares: Degrau não filtrado.

Figura 5.2: Comparação entre velocidades angulares para caso filtrado vs. caso não filtrado.

Da figura 5.2 percebe-se uma diminuição na amplitude das velocidades angulares da plataforma e também uma redução nas oscilações das mesmas. A próxima figura mostra o valor do sinal de controle gerado pelo controlador preditivo para as diferentes referências.



(a) Sinal de Controle - Degrau filtrado.



(b) Sinal de Controle - Degrau não filtrado.

Figura 5.3: Comparação entre resposta do controlador para o caso filtrado vs. o caso não filtrado.

Analisando o valor do sinal de controle gerado pelo controlador, destaca-se que a entrada degrau não filtrada faz com que as acelerações atinjam as restrições impostas pelo controlador preditivo, exigindo mais do atuador. Já para o caso filtrado o valor das acelerações se mantém abaixo das restrições. Logo, pode-se concluir que a utilização de referências do tipo filtrada é útil para melhorar o desempenho de um sistema, mantendo todos os outros parâmetros do controlador constantes. Por isso, todas as simulações feitas para os testes de controladores utilizarão entradas deste tipo.

Apesar de os resultados aqui obtidos terem utilizado o controlador preditivo, eles poderiam ter sido feitos utilizando-se outro tipo de controlador como o LQR ou PID. O objetivo desta seção é apenas ressaltar

diferenças na resposta de um mesmo controlador quando submetido a uma referência filtrada.

5.4 Variação do horizonte de predição (N)

A discussão acerca da escolha do horizonte de predição deve ser iniciada pelos seguintes casos: pequenos valores de N e grandes valores N. O primeiro caso discutido é que o horizonte de predição não pode ser extremamente pequeno, pois, como indicado por Alamir (2013) o horizonte de predição está diretamente ligado a questões de estabilidade do controlador preditivo. Existem outras considerações acerca da estabilidade do controlador, mas serão deixadas de lado pois esta análise não é o foco do presente trabalho. O segundo caso discutido é que o horizonte de predição também não pode ser tão grande quanto se queira, por motivos que serão explicados a seguir.

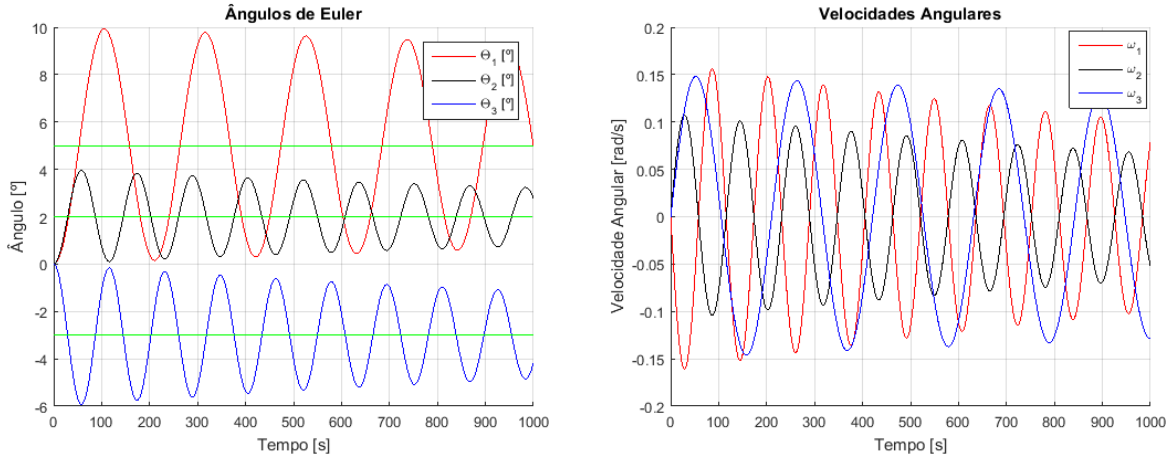
Um dos principais efeitos de um N pequeno é a redução na ordem de grandeza das matrizes envolvidas na formulação do controlador preditivo (vide seção 4.2). Isto resulta em um problema de otimização de baixa dimensão, reduzindo o tempo gasto pelo algoritmo de otimização para encontrar a sequência ótima de controle. Do ponto de vista de resposta dinâmica, um N pequeno torna a resposta do sistema mais suscetível a grandes *overshoot's* e um maior tempo de assentamento. Isto explica-se pelo fato de que, para pequenos valores de N, o controlador não consegue prever o comportamento do sistema a longo prazo, sendo incapaz de gerar um sinal de controle que evite o comportamento indesejado. De maneira oposta, o aumento de N aumenta a ordem de grandeza das matrizes da formulação do controlador preditivo, aumentando a dimensão do problema de otimização, logo aumentando o tempo gasto pelo algoritmo de otimização. Em contrapartida, reduz o tempo de assentamento e reduz o nível de *overshoot* produzido pela resposta do sistema. Pelo mesmo raciocínio utilizado anteriormente, o uso de um N grande faz com que o controlador seja capaz de prever o comportamento do sistema a longo prazo, produzindo um sinal de controle capaz de melhorar o desempenho do sistema.

Das informações do parágrafo anterior pode-se querer inferir que a solução para um melhor desempenho do controlador preditivo seja apenas o aumento progressivo do horizonte de predição. Porém, dois aspectos devem ser levados em consideração quando se deseja aumentar o valor de N: um de cunho teórico e outro prático. Do ponto de vista teórico pode-se imaginar que uma vez em posse de um *hardware* com capacidade ilimitada de memória e poder de processamento, pode-se aumentar o N o quanto se queira. O problema dessa abordagem acontece exatamente devido ao processo de predição em que a estratégia preditiva está fundamentada. A predição é feita a partir de um modelo teórico do sistema. A modelagem por sua vez não é capaz de capturar todas as características presentes em um modelo real, bem como os efeitos externos que agem sobre o mesmo. Logo, o desempenho do controlador preditivo é altamente dependente da acurácia do modelo de predição (EREN et al., 2017), podendo inviabilizar o funcionamento do controlador.

Do ponto de vista prático sabe-se que os *hardwares* são limitados em termos de quantidade de memória e poder de processamento. Como o aumento de N afeta diretamente o tamanho do problema de otimização, necessita-se de um maior poder de processamento para realizar os cálculos no tempo exigido. Caso o *hardware* escolhido não seja capaz de realizar a otimização da função custo dentro do período de amostragem escolhido para o controlador, ele pode gerar uma resposta sub-ótima piorando o desempenho

do controlador, ou até mesmo inviabilizar a lei de controle. Logo, o próprio equipamento utilizado impõe uma barreira ao valor máximo de N a ser utilizado.

Feitas essas considerações, esta seção busca simular vários cenários para diferentes valores de horizonte de predição. O objetivo é determinar como este parâmetro afeta a resposta dinâmica do sistema. Para todas as simulações os parâmetros de sintonia do controlador permaneceram constantes alterando-se apenas o horizonte de predição N. Para todos os casos as seguintes referências para os ângulos de *euler* foram passadas, $\theta_1 = 5°$, $\theta_2 = 2°$ e $\theta_3 = -3°$, os parâmetros de sintonia do controlador são $Qy = diag([1e8, 1e8, 1e8, 1, 1, 1])$ e $Qu = diag([1e4, 1e4, 1e4])$. As simulações serão feitas para os seguintes horizontes de predição N = $[5, 20, 40, 60, 100]$. A figura abaixo corresponde ao horizonte de predição N = 5.

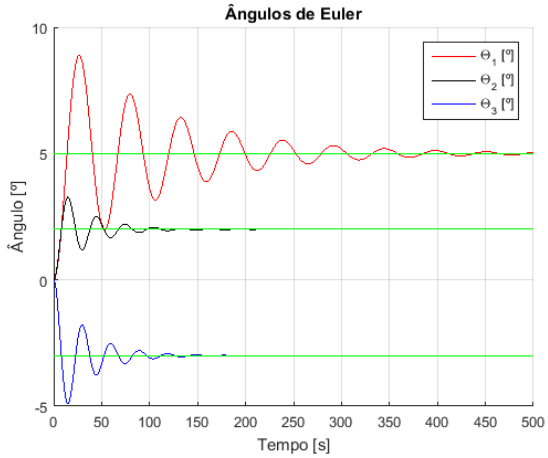


(a) Ângulos de Euler para N = 5.

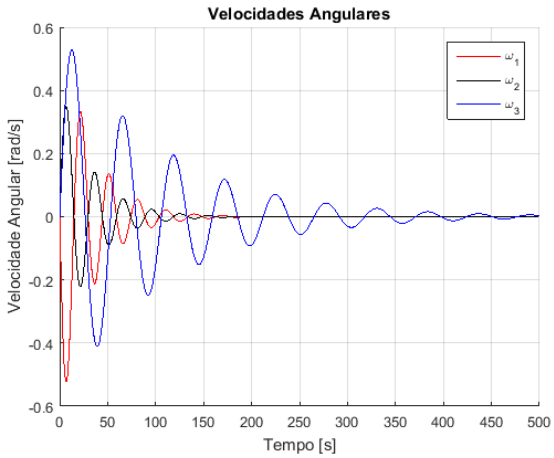
(b) Velocidades angulares para N = 5.

Figura 5.4: Resposta do controlador MPC para N = 5.

A figura 5.4 mostra que a resposta dos estados da plataforma são bastante oscilantes para um pequeno valor do horizonte de predição. Percebe-se que a resposta dos sistema apresenta um pequeno decaimento ao longo do tempo, porém, a simulação teria que ser executada durante um longo tempo para que o sistema estabilizasse. O próximo cenário a ser testado corresponde a um horizonte de predição de N = 20.



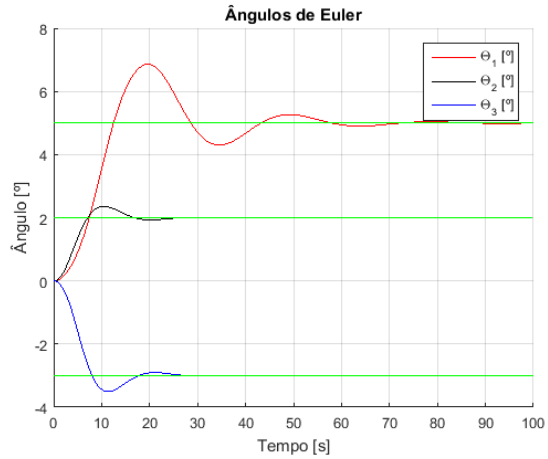
(a) Ângulos de Euler para N = 20.



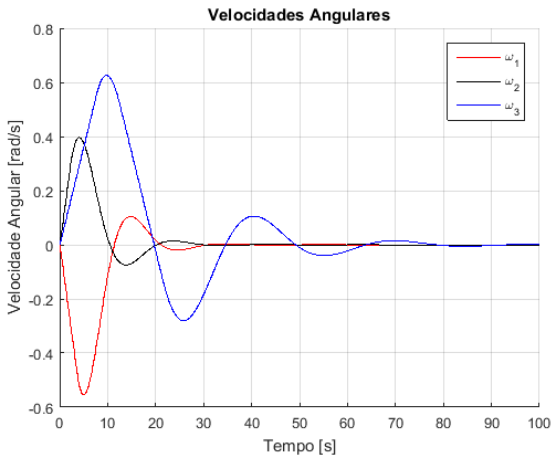
(b) Velocidades angulares para N = 20.

Figura 5.5: Resposta do controlador MPC para N = 20.

Para este horizonte de predição percebe-se uma melhora na resposta transitória comparado com o cenário anterior onde $N = 5$. A resposta do sistema ainda apresenta uma quantidade significativa de oscilação para o ângulo de θ_1, bem como um alto valor de *overshoot*, demorando mais para chegar a estabilidade do que os ângulos θ_2 e θ_3. Porém ainda exige um tempo superior a $t > 500s$ para que sistema alcance a estabilidade por completo. O próximo cenário mostra os resultados para o horizonte de predição N = 40.



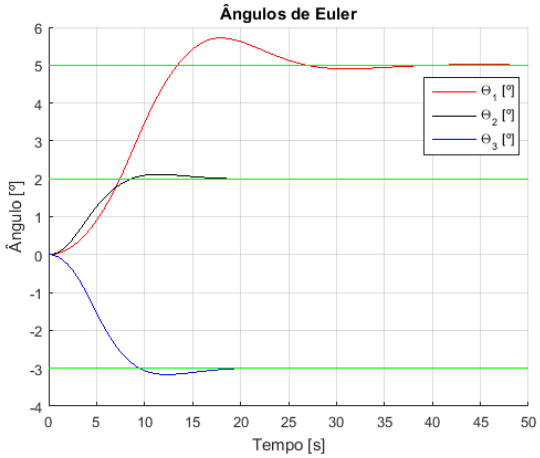
(a) Ângulos de Euler para N = 40.



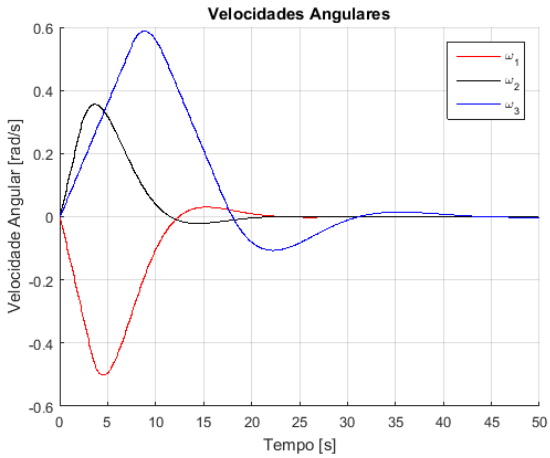
(b) Velocidades angulares para N = 40.

Figura 5.6: Resposta do controlador MPC para N = 40.

Os resultados da figura 5.6 demonstram uma melhora significativa na resposta do controlador. As oscilações foram reduzidas para todos os ângulos de *euler* e também para as velocidades angulares. Apesar da redução em relação ao cenário com horizonte de predição $N = 20$, ainda se nota um valor alto para o *overshoot* apresentado pelo ângulo θ_1. As oscilações das velocidades angulares foram bastante reduzidas e o sistema precisou de um tempo $t < 100s$ para estabilizar. Outro cenário testado corresponde ao horizonte de predição N = 60.



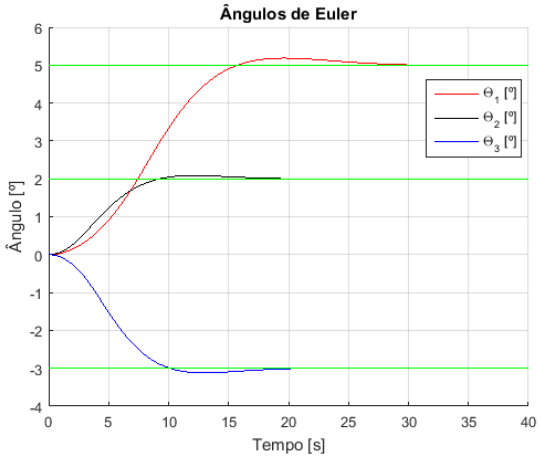
(a) Ângulos de Euler para N = 60.



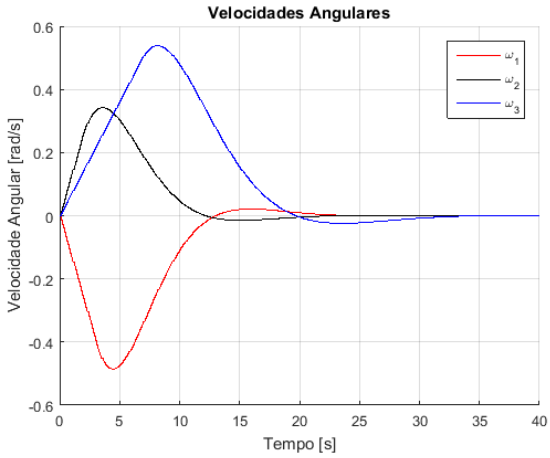
(b) Velocidades angulares para N = 60.

Figura 5.7: Resposta do controlador MPC para N = 60.

Para este cenário o sistema atinge a estabilidade em um tempo de simulação $t < 50s$ e uma diminuição no *overshoot* do ângulos θ_1, θ_2 e θ_3. As oscilações das velocidades angulares também foram reduzidas. O último teste a ser realizado é para um horizonte de predição $N = 100$.



(a) Ângulos de Euler para N = 100.



(b) Velocidades angulares para N = 100.

Figura 5.8: Resposta do controlador MPC para N = 100.

Da figura 5.8 é visto que *overshoot* para todos os ângulos de *euler* foram reduzidos e o sistema foi capaz de atingir a estabilidade em um tempo $t < 35s$. As velocidades angulares ω_1 e ω_2 foram pouco modificadas, apenas a velocidade ω_3 teve sua amplitude no início da simulação alterada e a redução do *undershoot* em $25 < t < 20$.

A análise dos resultados anteriores revela que com a redução no valor de N a resposta do sistema é extremamente oscilatória e o tempo necessário para estabilização do sistema aumenta. Por sua vez o aumento do N melhora a resposta do sistema, reduzindo os *overshoot's* e o tempo de assentamento dos estados do sistema. Um inconveniente é o aumento do tempo necessário para realização dos cálculos. As

informações acerca do *overshoot* e tempo de assentamento relacionado aos ângulos de *euler* para todas as simulações anteriores estão reunidos na tabela 5.4.

Tabela 5.4: Variação de *overshoot* e tempo de assentamento para um N variável.

	θ_1,θ_2,θ_3	
N	**Overshoot (%)**	**Tempo de Assentamento (s)**
5	(100,100,100)	(>1000,>1000,>1000)
20	(54,34,67)	(>500,210,160)
40	(36,18,16)	(97,26,26)
60	(15,5.5,5.5)	(47,20,20)
100	(5,4,4)	(30,20,20)

Esta seção buscou analisar como o desempenho do controlador preditivo varia conforme o horizonte de predição é modificado. Na próxima seção será feita uma análise mais detalhada de como o aumento ou diminuição do horizonte de predição N está relacionado aos tempos gastos na etapa de otimização e como estes são modificados pela técnica de parametrização.

5.5 Controlador Parametrizado Vs. Controlador Não Parametrizado

Como mencionado anteriormente o problema de otimização relacionado ao controle preditivo será tão mais complexo quanto maior for o tamanho do modelo e o valor do horizonte de predição utilizado. A técnica de parametrização é aplicada com o objetivo de reduzir número de graus de liberdade do problema de otimização (ALAMIR, 2013), reduzindo o tempo gasto pelo algoritmo de otimização na determinação do mínimo da função custo. A análise do tempo gasto nesta etapa é um aspecto importante a ser discutido pois é a partir dele que se determina a viabilidade do controlador preditivo para ser embarcado em *hardware*. Nesta seção, para verificar de que maneira a parametrização impacta no desempenho do controle preditivo, busca-se responder a seguinte pergunta: Como o tempo gasto na etapa de otimização varia conforme se aumenta o valor de N para o controlador parametrizado e para o controlador não parametrizado ?

Para isto, dois cenários serão testados em *Matlab*. O primeiro corresponde ao controlador preditivo não parametrizado e o segundo o controlador preditivo parametrizado exponencialmente. Este utiliza duas exponenciais e os seguintes parâmetros de sintonia: $tr = 1.5$ e $\alpha = 20$. Para os dois casos as matrizes de ponderação foram $Q_y = diag(1 \times 10^8, 1 \times 10^8, 1 \times 10^8, 1, 1, 1)$ e $Q_u = diag(1 \times 10^4, 1 \times 10^4, 1 \times 10^4)$, onde o valor das referências passadas foi o mesmo, alterando-se apenas o horizonte de predição. Para otimização da função custo foi utilizado o comando *quadprog* com a escolha do algoritmo *interior-point-convex*, que são disponibilizados pelo *Matlab*. A medição do tempo gasto pelo comando *quadprog* foi feito através dos comandos *tic* e *toc*.

É importante destacar que como essa medição de tempo é realizada pelo *Matlab* que está sendo executado em um computador de propósito geral, essas medições podem variar conforme o número de processos que estão sendo executados em segundo plano no computador durante a execução das simulações. Para isto cada simulação foi repetida várias vezes com o intuito de se obter uma melhor representação do tempo gasto na etapa de otimização.

O controlador preditivo parametrizado e o não parametrizado foram testados para os seguintes valores de horizonte de predição $N = [5, 10, 15, 20, 30, 40, 60, 80, 100]$. O tempo de simulação para cada valor de N foi de $60s$. Durante o *loop* de simulação e nas várias vezes que o comando *quadprog* é chamado, o tempo utilizado para sua execução foi medido e armazenado em um vetor. Ao fim de cada simulação foi feita uma média aritmética simples do tempo total gasto pelo *quadprog* dividido pelo número de chamadas desta função no programa principal. Este valor representa o tempo médio gasto pelo otimizador por simulação. Com o intuito de obter um valor mais representativo, cada simulação para um determinado N foi repetida 10 vezes. Das mesma maneira é realizada uma média aritmética simples entre os valores obtidos nesta 10 simulações. Este por fim é escolhido como o valor do tempo gasto pelo otimizador para um dado valor de horizonte de predição N, onde é finalmente plotado na figura 5.9.

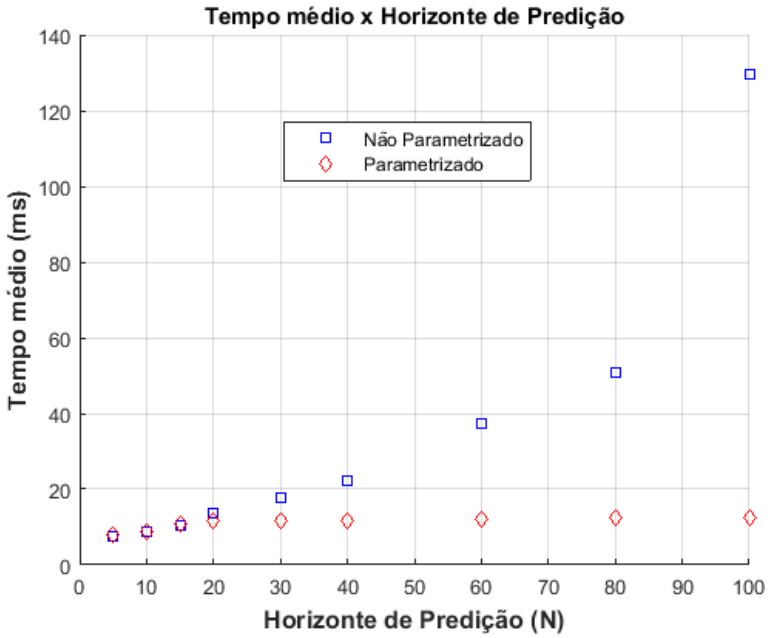


Figura 5.9: Comparação entre o tempo médio gasto pelo *quadprog* em relação ao aumento de N.

Os resultados das simulações demonstram que para valores de $N <= 15$ o tempo gasto pelo otimizador para o caso parametrizado e não parametrizado é praticamente o mesmo. Para valores de $N > 15$ percebe-se que o tempo gasto pelo controlador não parametrizado começa a aumentar rapidamente. Já o tempo gasto no controlador preditivo parametrizado também aumenta mas permanece com valores abaixo dos $20ms$. Para o caso extremo de $N = 100$ o tempo gasto pelo controlador não parametrizado é 10 vezes maior que o tempo gasto pelo controlador parametrizado.

Aliada a discussão da seção anterior, os resultados obtidos com a parametrização demonstram a possibilidade de se projetar um controlador preditivo utilizando grandes valores de N e ao mesmo tempo reduzir a carga computacional gerada pelo problema de otimização, o que pode ser mais crítico quando se planeja embarcar o controlador em *hardware*. Um detalhe importante dos resultados da parametrização é que os dados das simulações foram obtidos através de simulações em um *desktop* e que este não representa as mesmas características de processamento do *hardware* final onde o controlador será embarcado. Porém, ainda é possível obter um bom entendimento de que como a parametrização pode contribuir no

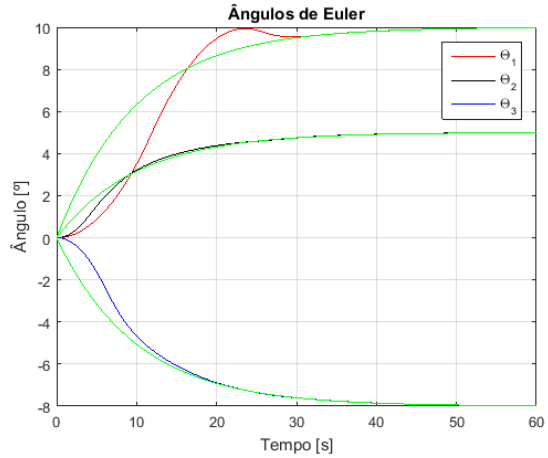
desenvolvimento de controladores preditivos embarcados.

5.6 Parametrização exponencial

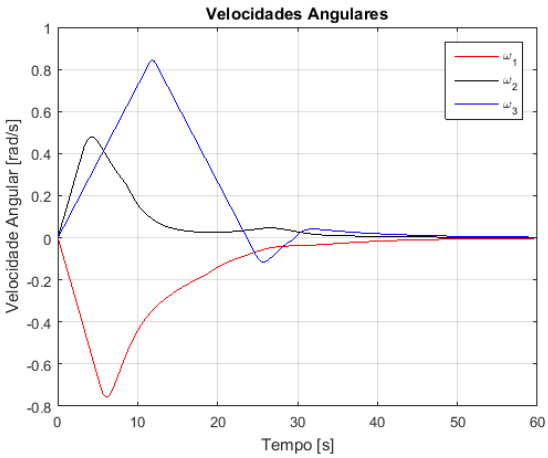
Esta seção irá analisar de que maneira as variáveis da parametrização (4.40) modificam a resposta do controlador. As variáveis são: o número de exponenciais utilizadas (n_e) por atuador, o tempo de resposta do atuador (tr) e a variável α. Para uma análise de como essas variáveis alteram a reposta do controlador, três cenários serão testados. No primeiro cenário α será variado e os outros dois parâmetros serão mantidos constantes. No segundo os valores de t_r e α serão mantidos constantes e o valor de n_e será modificado. O terceiro cenário t_r será variado enquanto n_e e α serão mantidos constantes. Todos os cenários foram simulados com os seguintes parâmetros constantes $Q_y = diag(1 \times 10^8, 1 \times 10^8, 1 \times 10^8, 1, 1, 1)$, $Q_u = diag(0.1, 0.1, 0.1)$ e N = 50.

5.6.1 Cenário 1 - Ajuste de α.

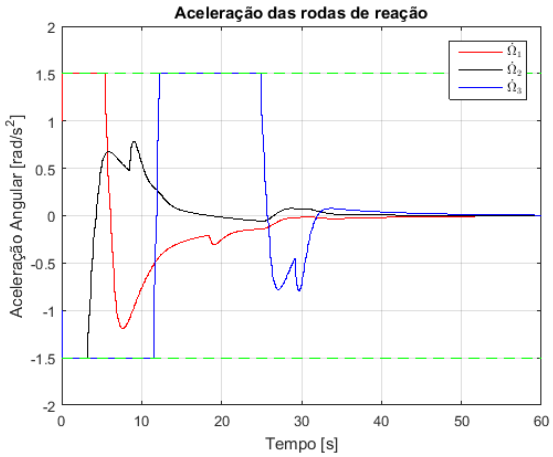
Este primeiro cenário será feito com os seguintes parâmetros constantes $t_r = 1.5$ e $n_e = 2$. O primeiro resultado é mostrado na figura abaixo para o caso onde $\alpha = 1$.



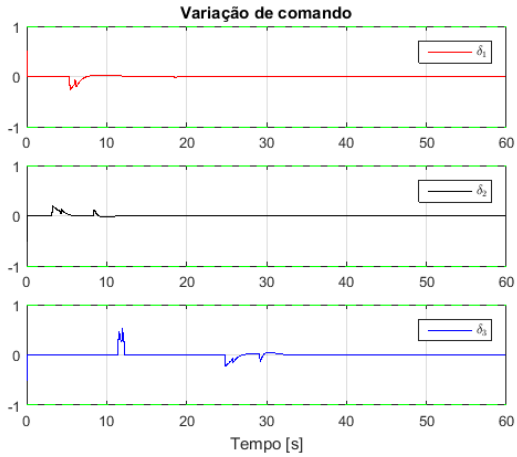
(a) Ângulos de Euler.



(b) Velocidade Angular.



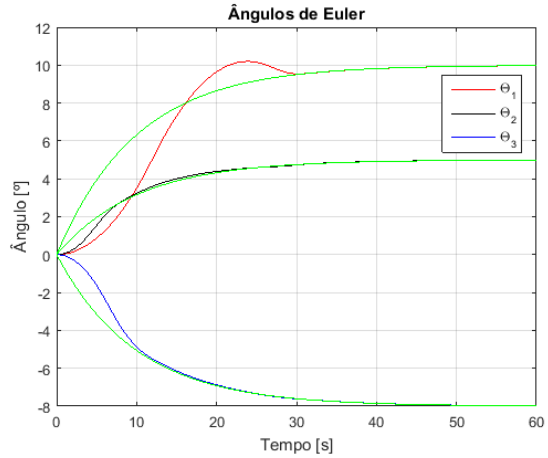
(c) Comando



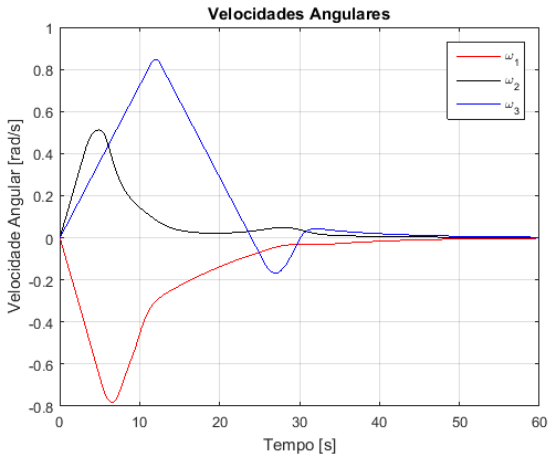
(d) Variação de comando.

Figura 5.10: Controlador parametrizado: ne = 2, tr = 1.5, α = 1.

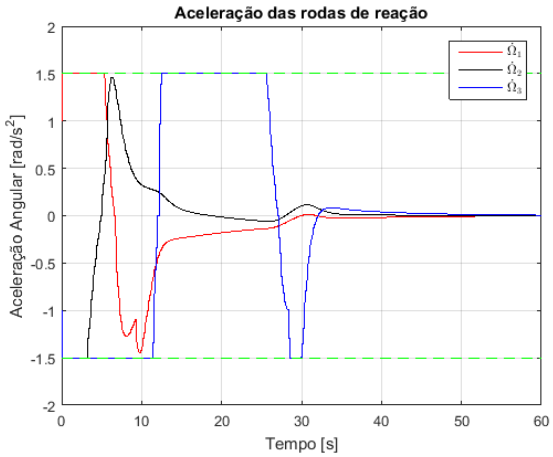
O segundo teste realizado é feito com $\alpha = 10$. Os resultados são mostrados na figura abaixo.



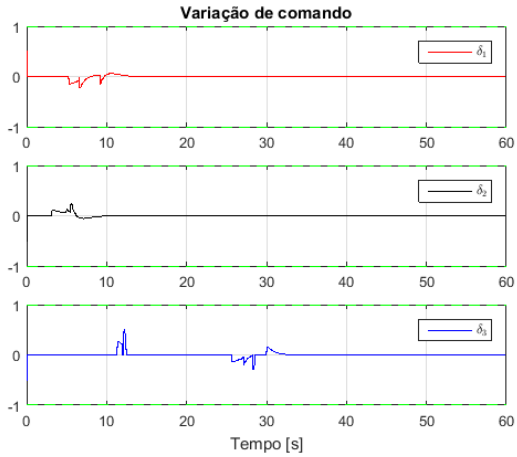
(a) Ângulos de Euler.



(b) Velocidade Angular.



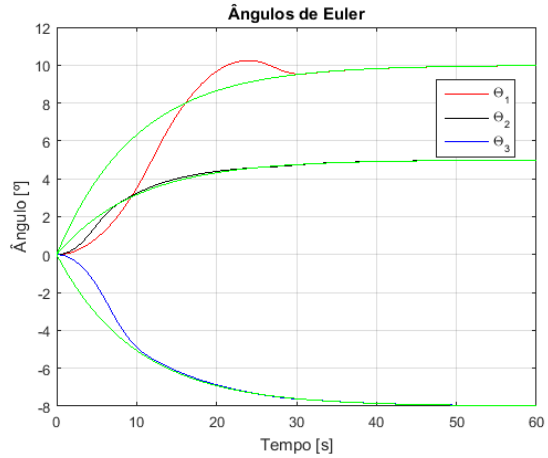
(c) Comando



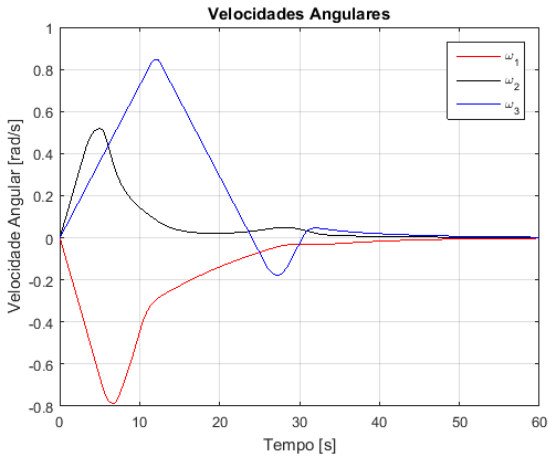
(d) Variação de comando.

Figura 5.11: Controlador parametrizado: ne = 2, tr = 1.5, α = 10.

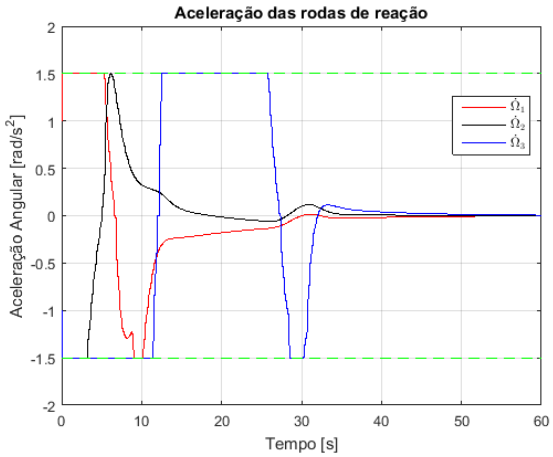
O terceiro teste realizado é feito com α = 20, onde os resultados são mostrados na figura abaixo.



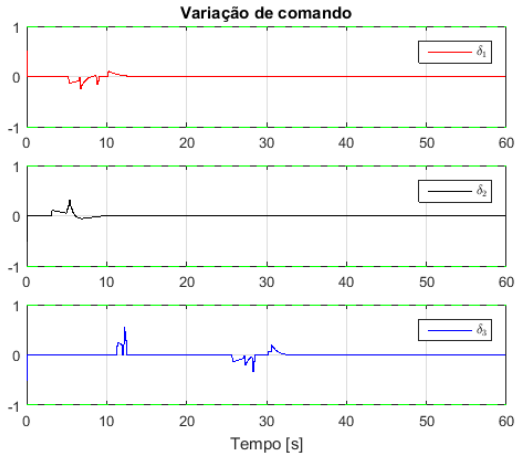
(a) Ângulos de Euler.



(b) Velocidade Angular.



(c) Comando



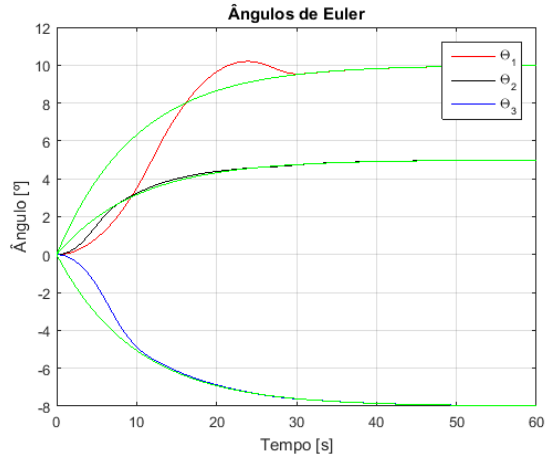
(d) Variação de comando.

Figura 5.12: Controlador parametrizado: ne = 2, tr = 1.5, α = 20.

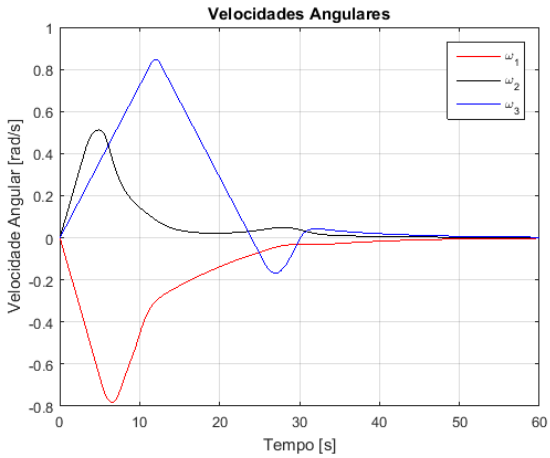
Analisado os gráficos anteriores fica claro que o aumento do valor de α altera a variação do sinal de controle e a variação do mesmo. A medida que ele aumenta mais variável fica o sinal de controle, aumentando a amplitude do mesmo até atingir as restrições impostas, como pode ser concluído pela comparação entre os cenários onde $\alpha = 1$ e $\alpha = 20$.

5.6.2 Cenário 2 - Ajuste de n_e.

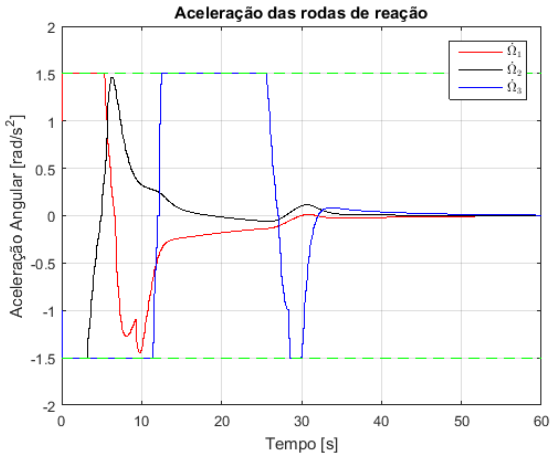
O primeiro teste realizado será feito com os valores de α = 10 e t_r = 1.5 constantes, variando-se o número de exponenciais utilizadas n_e. O primeiro resultado para n_e = 2 é mostrado na figura 5.13.



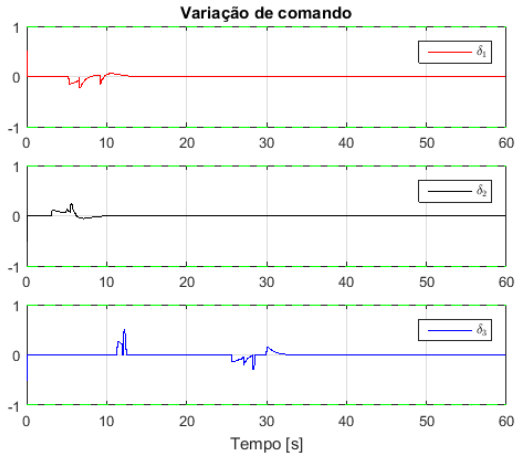
(a) Ângulos de Euler.



(b) Velocidade Angular.



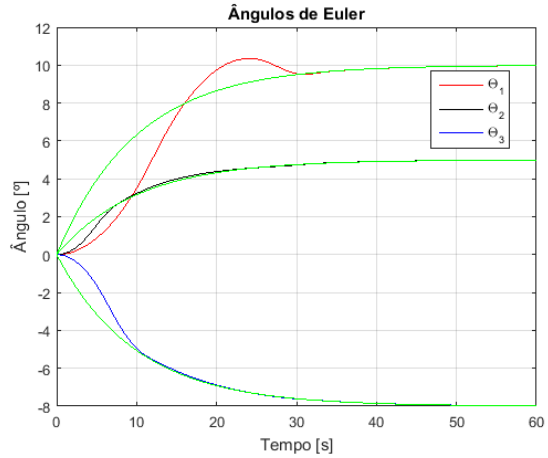
(c) Comando.



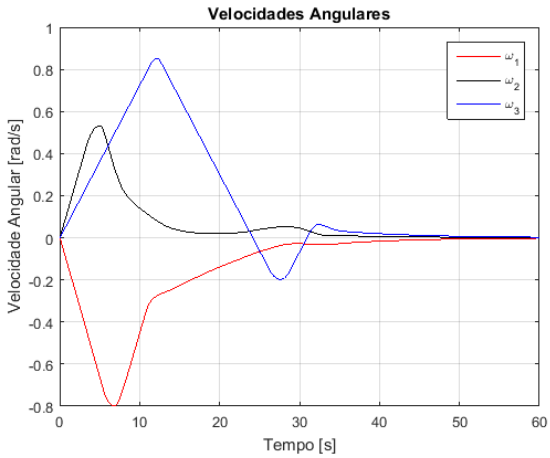
(d) Variação de comando.

Figura 5.13: Controlador parametrizado: ne = 2, tr = 1.5, α = 10

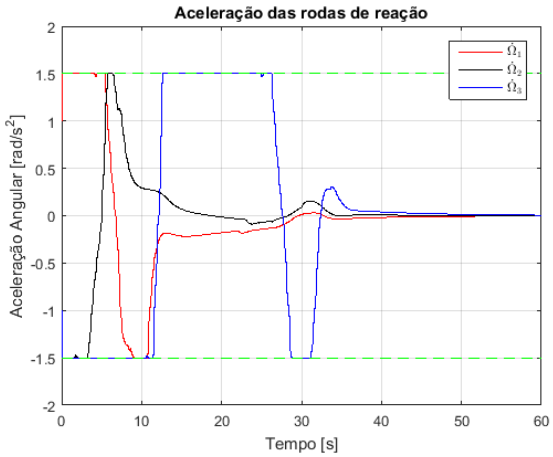
O segundo teste corresponde ao valor de n_e = 4.



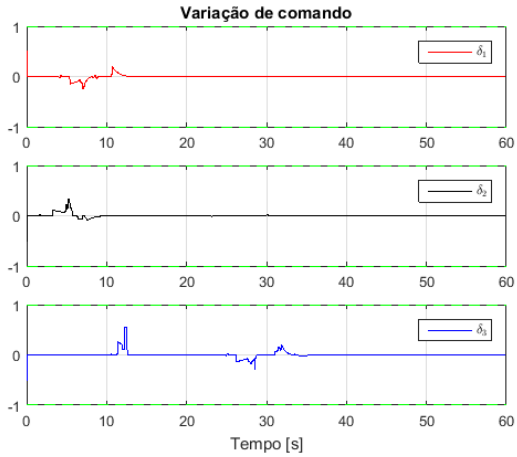
(a) Ângulos de Euler.



(b) Velocidade Angular.



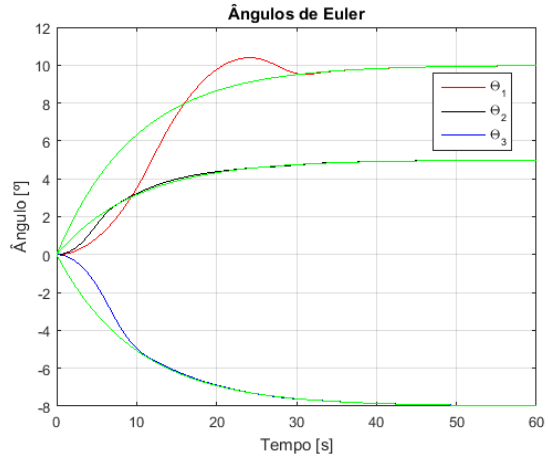
(c) Comando.



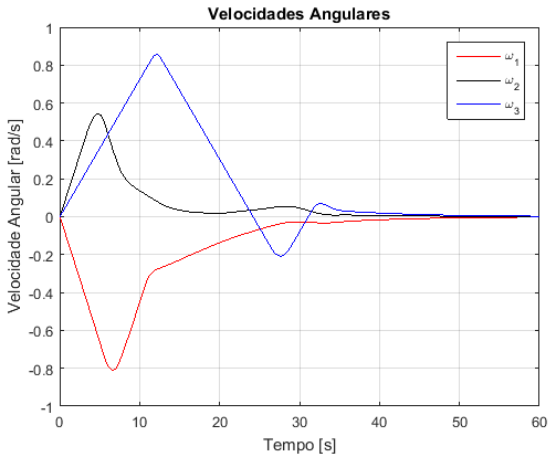
(d) Variação de comando.

Figura 5.14: Controlador parametrizado: ne = 4, tr = 1.5, α = 10.

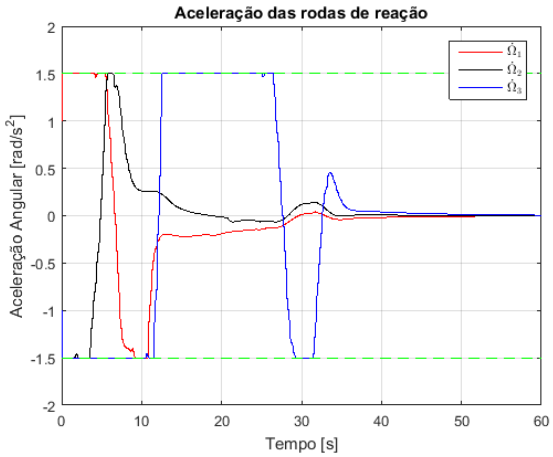
O terceiro teste realizado para n_e = 8.



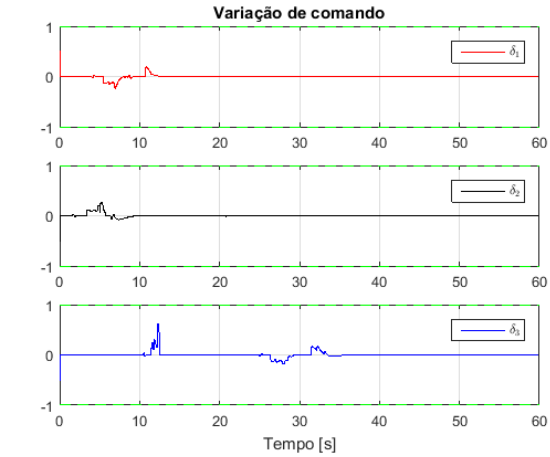
(a) Ângulos de Euler.



(b) Velocidade Angular.



(c) Comando.



(d) Variação de comando.

Figura 5.15: Controlador parametrizado: ne = 8, tr = 1.5, α = 10.

Percebe-se que o aumento do número de exponenciais não altera substancialmente a dinâmica dos estados do sistema. A alteração perceptível é aquela que ocorre no sinal de controle e na variação do mesmo. Considera-se que quanto maior o número de exponenciais melhor a representação do sinal de comando. Porém, quanto maior o número de exponenciais utilizadas, maior a dimensão das matrizes parametrizadas A_{ineq}, H e F do problema de otimização 4.33. Isto ocasiona um aumento no tempo gasto na otimização da função custo, como pode ser visto na tabela 5.5.

Tabela 5.5: Tempo médio gasto no problema de otimização com a variação de n_e

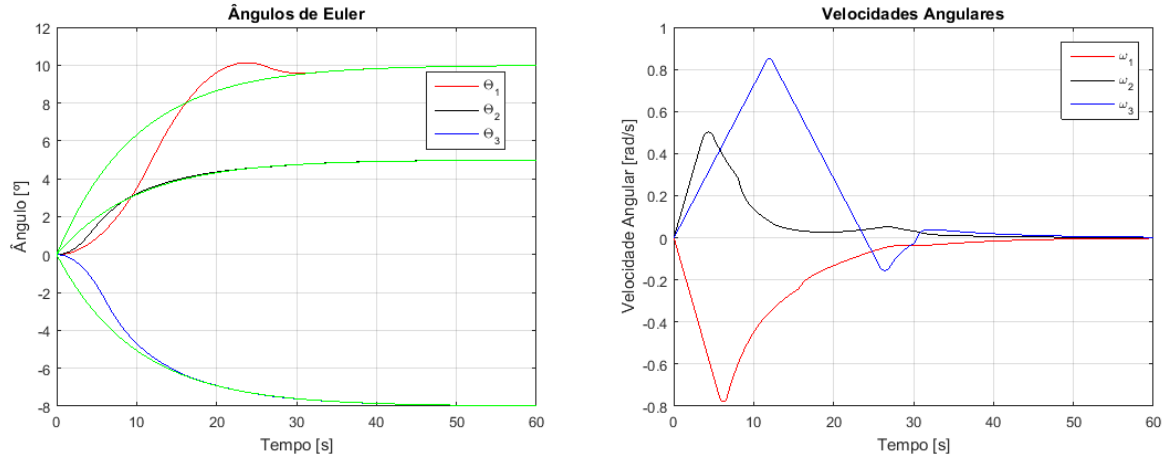
n_e	2	4	8	10	20
Tempo médio (ms)	11,66	28,10	43,22	49,76	81,74

Esta tabela relaciona o tempo gasto pelo otimizador (em milissegundos) a medida que o número de exponenciais aumenta. Aqui utilizou-se a mesma metodologia aplicada na seção 5.5 para o cálculo do tempo médio gasto na etapa de otimização da função custo. Logo, apesar da escolha de um alto n_e ser preferível para obter uma melhor representação do sinal de controle a escolha deste valor deve ser feita

considerando o aumento do custo computacional.

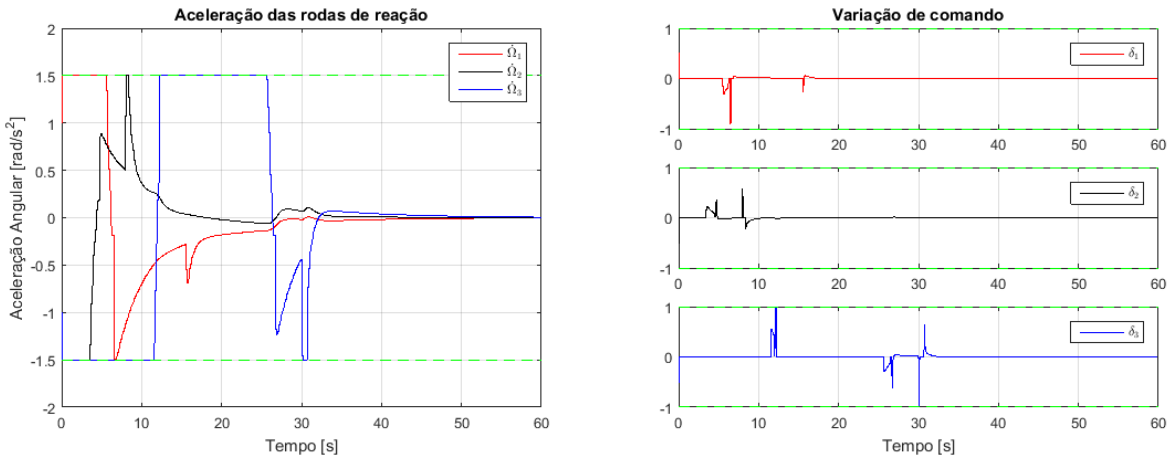
5.6.3 Cenário 3 - Ajuste de t_r.

Este primeiro cenário será feito com os seguintes parâmetros constantes $\alpha = 20$ e $n_e = 2$. O primeiro resultado é mostrado na figura abaixo para o caso onde o tempo de resposta do atuador é $t_r = 0.1$.



(a) Ângulos de Euler.

(b) Velocidade Angular.

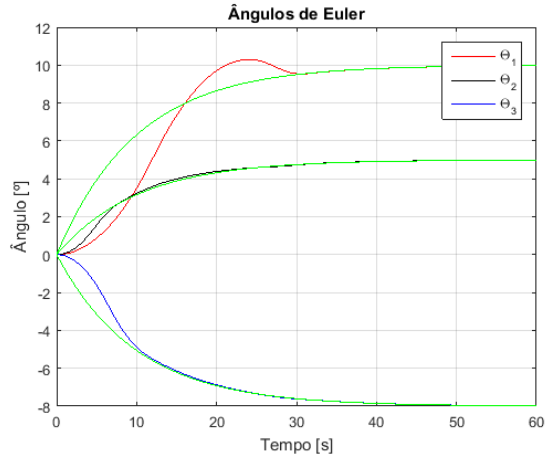


(c) Comando.

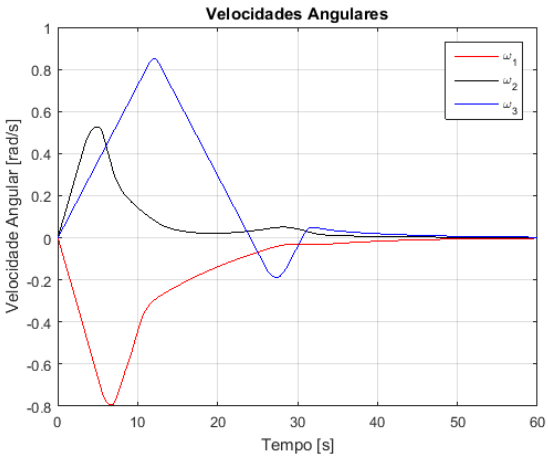
(d) Variação de comando.

Figura 5.16: Controlador parametrizado: ne = 2, tr = 0.1, α = 20.

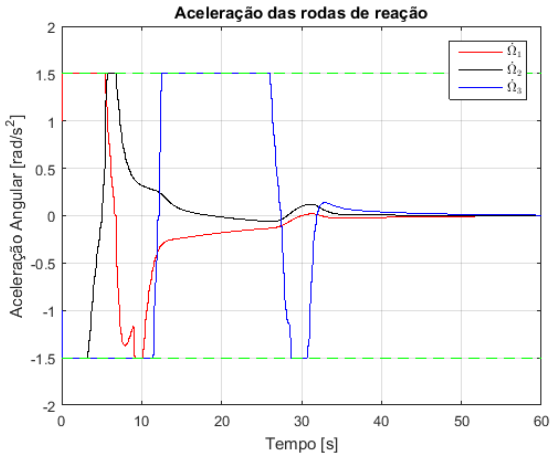
O segundo teste corresponde as simulações para $t_r = 1$.



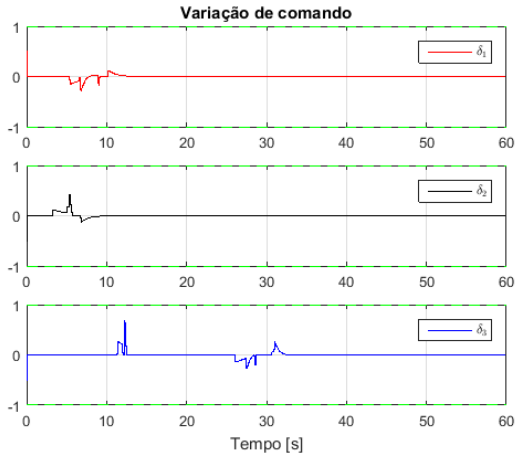
(a) Ângulos de Euler.



(b) Velocidade Angular.



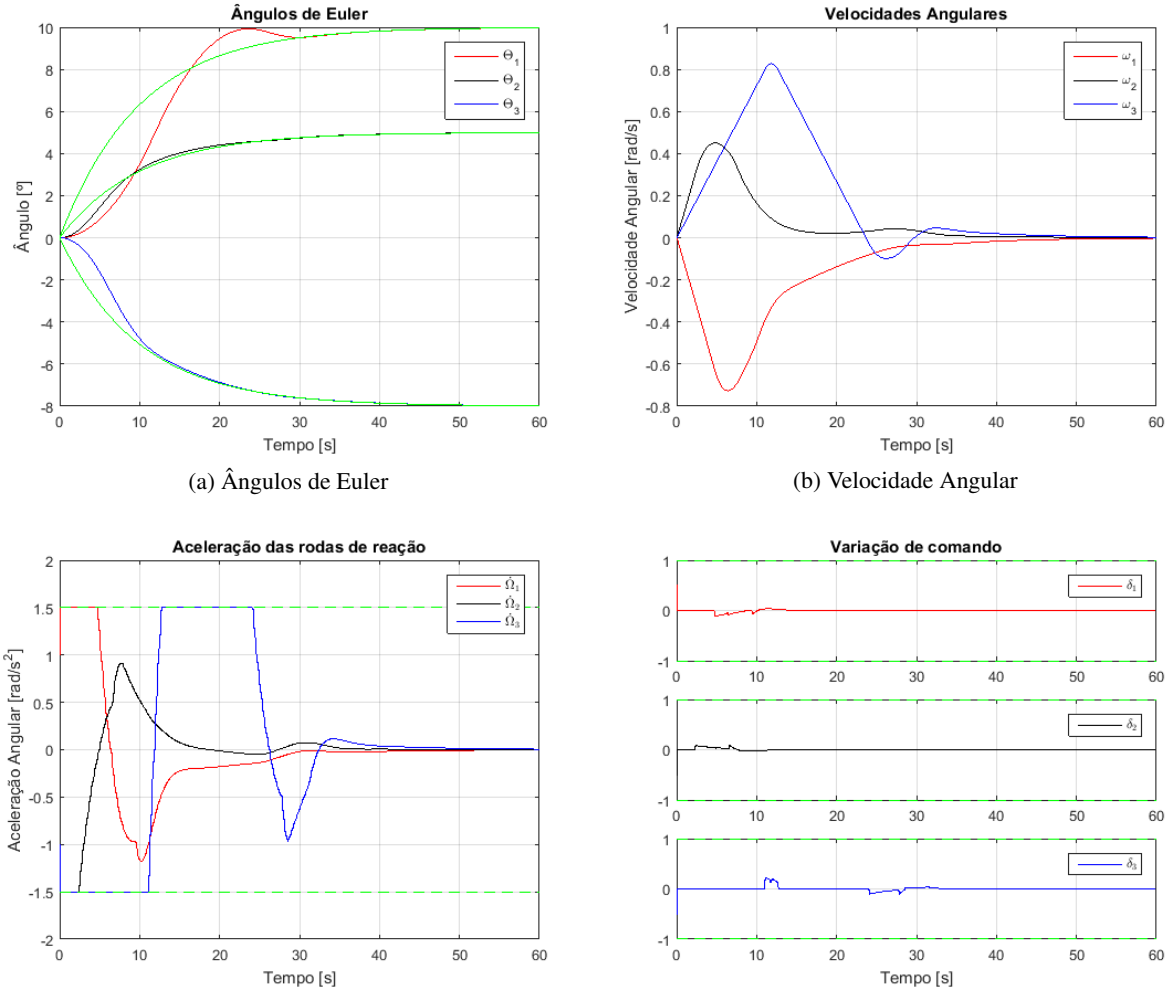
(c) Comando.



(d) Variação de comando.

Figura 5.17: Controlador parametrizado: ne = 2, tr = 1, α = 20.

O terceiro e último testes corresponde as simulações para t_r = 10.



(a) Ângulos de Euler

(b) Velocidade Angular

(c) Comando

(d) Variação de comando

Figura 5.18: Controlador parametrizado: ne = 2, tr = 10, α = 20.

Percebe-se um decréscimo na amplitude de variação de comando a medida que o valor de t_r aumenta. O sinal de comando também é menos variável. Isto explica-se pelo fato de que como o valor de t_r está ligado diretamente ao tempo de resposta do atuador do sistema, quanto maior valor de t_r mais lenta a resposta do atuador e consequentemente uma menor variação em relação ao tempo.

Baseado nas análise dos três cenários anteriores o controlador MPC utilizará um número de exponenciais $n_e = 3$ que fornece um perfil de comando superior a $n_e = 2$ sem aumentar demasiadamente o custo computacional associado a solução do problema de otimização. Além disso, para o sistema em estudo, percebeu-se que para valores de $n_e > 3$ não existiam grandes diferenças no perfil de comando, somente um aumento no tempo associado a solução do problema de otimização, o que viria a ocasionar uma redução na velocidade do controlador quando embarcado em *hardware*.

O tempo de resposta escolhido será $t_r = 8.0$ e $\alpha = 10$. Esta decisão é baseada no fato que nas simulações em HIL, devido aos ruídos existentes no processo de medição dos estados, a escolha de pequenos valores de t_r e α faz com que haja um aumento demasiado da variação de comando, inclusive em alguns cenários ultrapassando as restrições impostas pelo projeto (tabela 5.3). Apesar das contribuições

das variáveis da parametrização t_r e α não serem tão visíveis nas simulações em MIL estas se mostraram decisivas nas simulações em HIL principalmente em controlar a amplitude da variação do sinal de controle.

5.7 Validação do controlador MPC

Nesta seção busca-se validar o controlador preditivo linear parametrizado exponencialmente. Um dos primeiros testes a serem executados corresponde a comparação entre o MPC e o LQR em ambiente MIL. O objetivo desta simulação é mostrar as vantagens do MPC sobre o LQR quanto ao respeito das restrições impostas e validar o MPC em MIL. Uma vez feita esta comparação, o controlador MPC será avaliado nas etapas subsequentes do *V-Cycle*:

- Controlador preditivo linear na *BB* em C/C++ e modelo não linear no *Matlab* - PIL.
- Controlador preditivo linear na *BB* em C/C++ e modelo não linear no *target* - HIL.

Para desenvolvimento do controlador preditivo linear e do controlador LQR utiliza-se o modelo linear da plataforma de testes de satélites, ou seja, as matrizes 4.11 e 4.9, que foram linearizadas em torno da origem. Para a comparação entre o controlador LQR e MPC escolheu-se uma referência de baixa amplitude ($\theta_1 = +11°$, $\theta_2 = +5°$, $\theta_3 = -11°$) filtrada com $t_{ref} = 30$, pois está na região de linearização dos controladores.

Depois de realizada esta comparação será escolhida uma referência de maior amplitude ($\theta_1 = +50°$, $\theta_2 = -30°$, $\theta_3 = 60°$) e $t_{ref} = 30$. O uso desta referência demonstrará que o MPC não consegue controlar o sistema para referências de grande amplitude, ou seja, distante da região de linearização do controlador. Este resultado então justificará o desenvolvimento do controlador preditivo não linear. Os parâmetros escolhidos para validação do controlador preditivo linear estão na tabela abaixo.

Tabela 5.6: Parâmetros escolhidos para validação do MPC

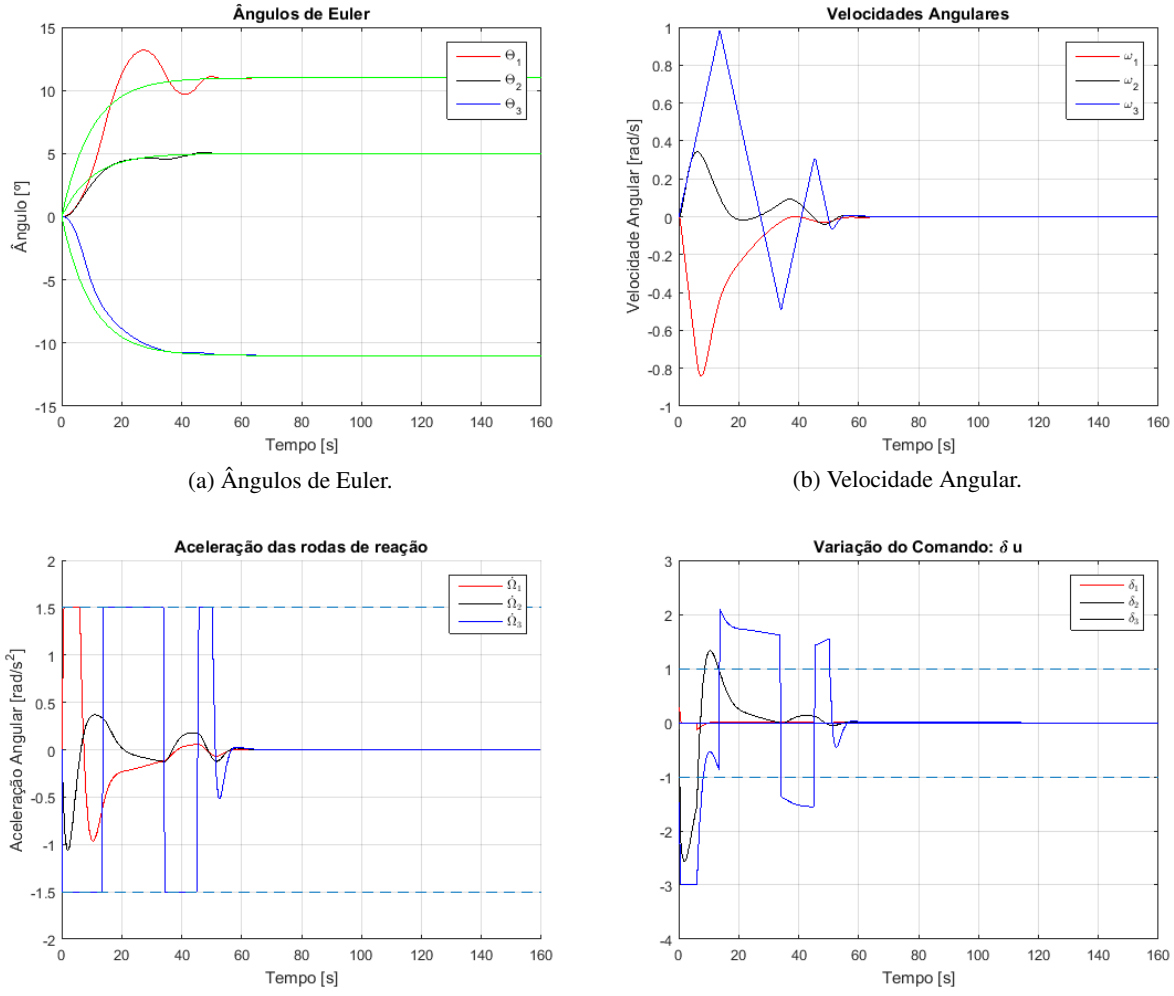
N	55
Q_y	$diag(0.3 \times 10^{10}, 1 \times 10^{8}, 1 \times 10^{8}, 1 \times 10^{3}, 1 \times 10^{3}, 1 \times 10^{3})$
Q_u	$diag(0.5 \times 10^{4}, 0.5 \times 10^{4}, 0.5 \times 10^{4})$
n_e	3
t_r	8.0
α	10

É importante destacar que a escolha final destes parâmetros é feita depois de testes sucessivos nas etapas MIL, PIL e HIL. O motivo é que há a possibilidade de se obter um conjunto de parâmetros que funcionem em MIL mas que não funcionem corretamente em PIL ou HIL, seja por especificidades do *hardware*, do algoritmo de otimização ou ruídos na plataforma HIL. Havendo falha em uma destas etapas, deve-se reiniciar os testes na etapa MIL até que se encontre um conjunto de parâmetros que funcione para todos os casos.

5.7.1 Comparação entre o MPC parametrizado e o controlador LQR no *Matlab* - MIL

O objetivo desta seção é comparar o desempenho do controlador linear LQR com saturação na variável de comando e o controlador preditivo linear. O controlador LQR utiliza as mesmas matrizes de penalização do MPC (tabela 5.6) e o mesmo nível de saturação na variável de comando (tabela 5.3).

Nestes testes o MPC e LQR estão descritos na mesma linguagem de *script* do *Matlab*. Para otimização da função custo quadrática do MPC foi utilizada a função *quadprog* do *Matlab*, através do algoritmo *interior-point-convex*. Aqui busca-se validar o MPC em MIL, verificar se os parâmetros de sintonia fornecem resultados satisfatórios e demonstrar as diferenças com o controlador LQR. A figura 5.20 mostra o resultados da simulação MIL para o controlador LQR para a referência $\theta_1 = +11°, \theta_2 = +5°, \theta_3 = -11°$.



(a) Ângulos de Euler.

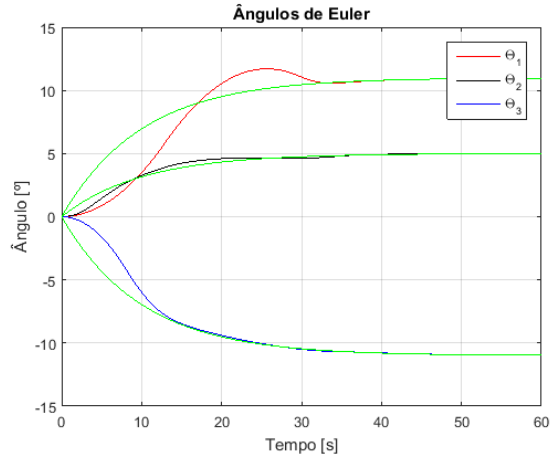
(b) Velocidade Angular.

(c) Comando.

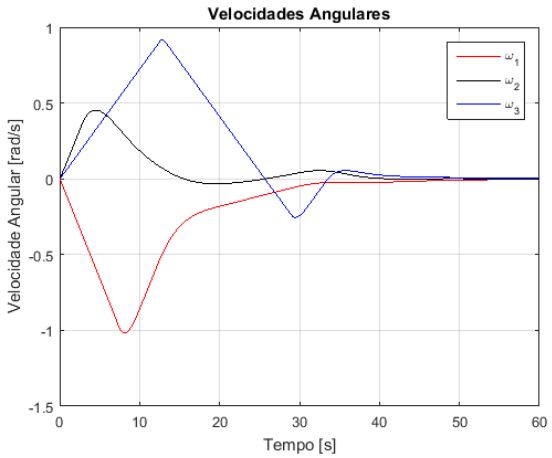
(d) Variação de comando.

Figura 5.19: Controlador LQR saturado em MIL - Referência: $11°, 5°, -11°$.

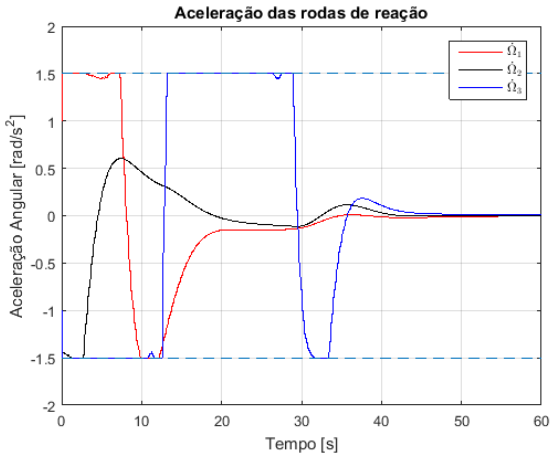
A figura 5.20 mostra o resultado do MPC para a mesma referência.



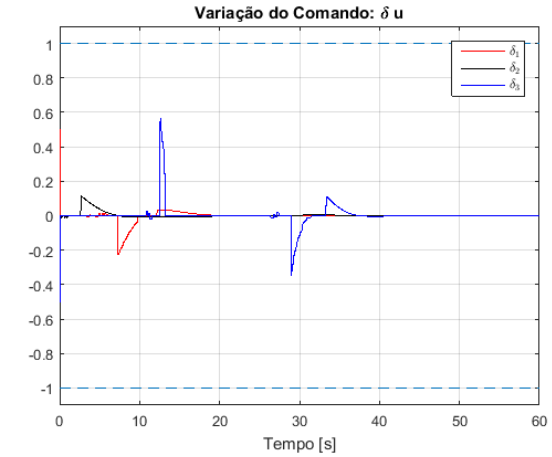
(a) Ângulos de Euler.



(b) Velocidade Angular.



(c) Comando.



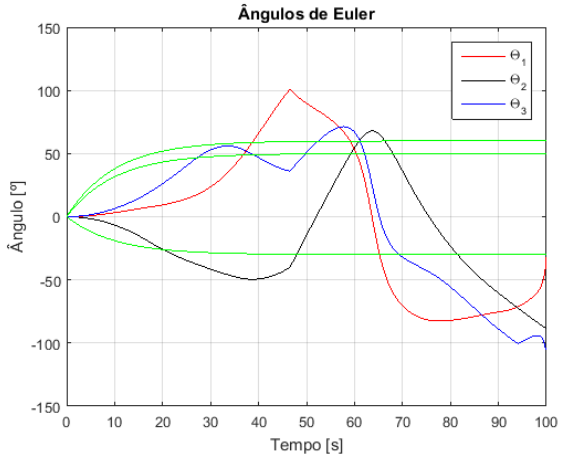
(d) Variação de comando.

Figura 5.20: Validação do MPC em MIL - Referência: $11°, 5°, -11°$.

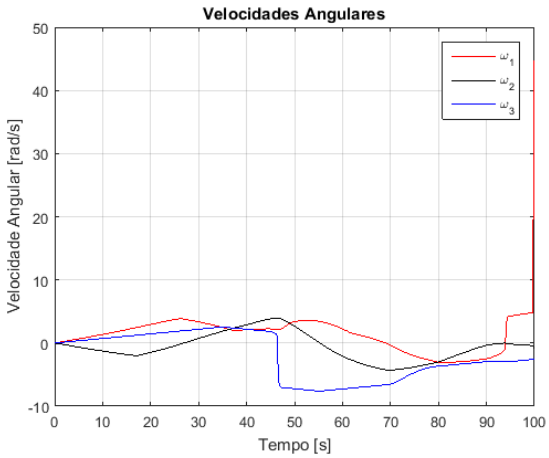
Comparando-se os resultados do MPC com o LQR percebe-se que o controlador MPC é capaz de controlar o sistema, respeitando as restrições impostas. O ângulo de *euler* θ_1, em vermelho na figura 5.20a, apresenta um *overshoot* de aproximadamente 11%, levando em torno de $40s$ para chegar a estabilidade. E para θ_2 e θ_3 este tempo é de aproximadamente $45s$.

Os resultados do controlador LQR também demonstram que apesar de conseguir controlar o sistema, ele apresenta um *overshoot* de 20% para o ângulo θ_1 e também grandes oscilações nos demais estados. O sistema possui tempo de acomodação superior a $78s$. Outro ponto a ser destacado é que o LQR não é capaz de respeitar as restrições impostas para variação de comando, i.e., $\delta_u = \pm 1.0 \frac{rad}{s^2}$ (figura 5.19d). Estes resultados demonstram que o MPC é um bom candidato ao problema em estudo tanto no que diz respeito a restrições do modelo e o desempenho associado com os estados do sistema.

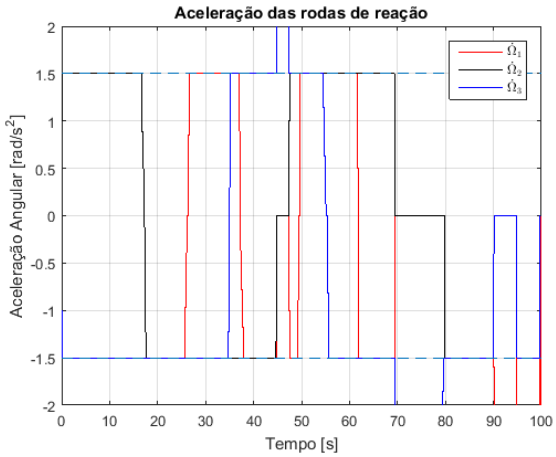
Uma vez terminada esta comparação, o controlador MPC será testado com uma referência de grande amplitude. O objetivo é demonstrar que o controlador linear não é capaz de controlar o sistema quando referências longe do ponto de linearização do controlador são aplicadas ao sistema. A referência utilizada é: $\theta_1 = +50°$, $\theta_2 = -30°$, $\theta_3 = +60°$, a figura 5.21 mostra os resultados desta simulação.



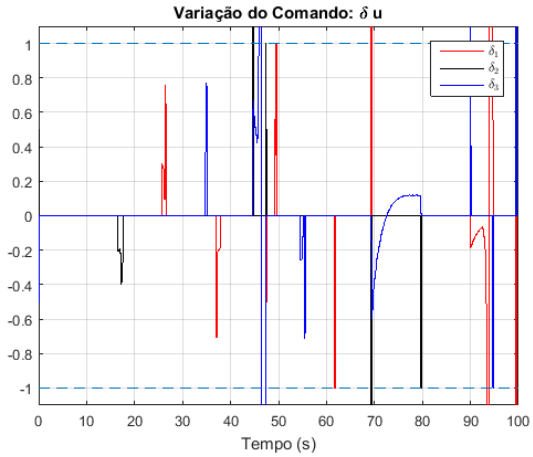
(a) Ângulos de Euler.



(b) Velocidade Angular.



(c) Comando.



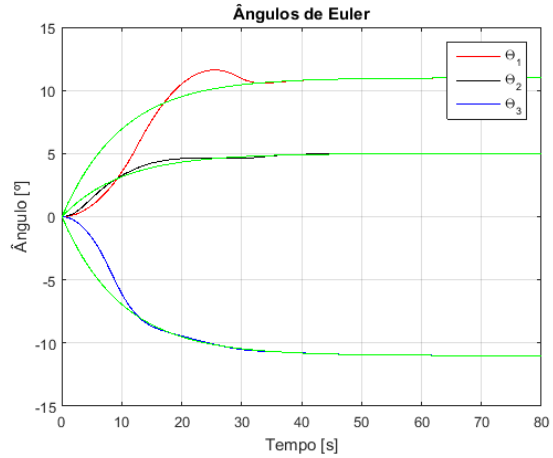
(d) Variação de comando.

Figura 5.21: Validação do MPC em MIL - Referência: $50°, -30°, 60°$.

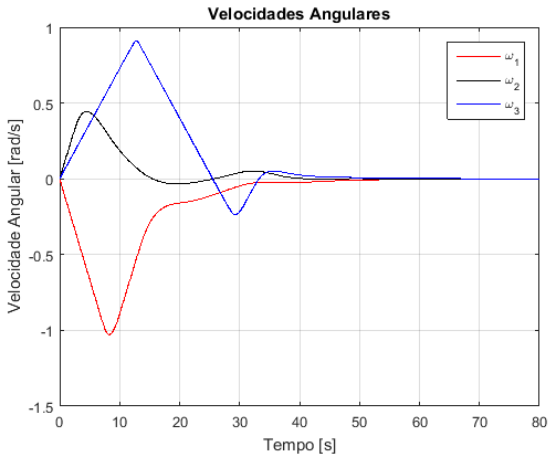
Percebe-se que para referências de grande amplitude o MPC não é capaz de controlar o sistema. O sistema diverge e as restrições de comando e variação de comando não são respeitadas. Isto motiva a utilização do controlador preditivo não linear para atuar nas regiões longe do ponto de linearização. Como neste cenário o controlador falhou, não será necessário prosseguir os testes em PIL e HIL com a utilização desta referência. Prossegue-se com os testes utilizando-se a referência $\theta_1 = +11°$, $\theta_2 = +5°$, $\theta_3 = -11°$ para demonstrar que apesar das limitações do MPC ele ainda pode ser validado em PIL e HIL para referências angulares de baixa amplitude.

5.7.2 Controlador preditivo linear na *BB* em C/C++ e Modelo não linear no *Matlab* - PIL

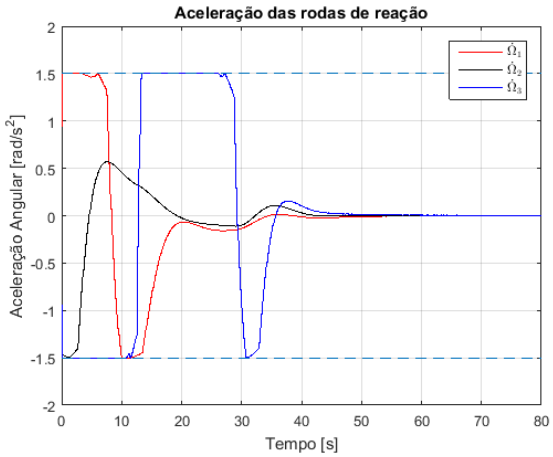
Esta etapa corresponde aos testes onde o controlador preditivo linear está embarcado na *BeagleBone*. O objetivo destas simulações é validar o controlador desenvolvido na etapa MIL, para referências de pequena amplitude. Nesta etapa utiliza-se o *solver qpOASES* para minimização da função custo. O *quadprog* não pode ser utilizado por se tratar de uma solução proprietária não sendo possível sua utilização fora do ambiente do *Matlab*. As figuras abaixo mostram os resultados para o MPC embarcado na *BeagleBone*.



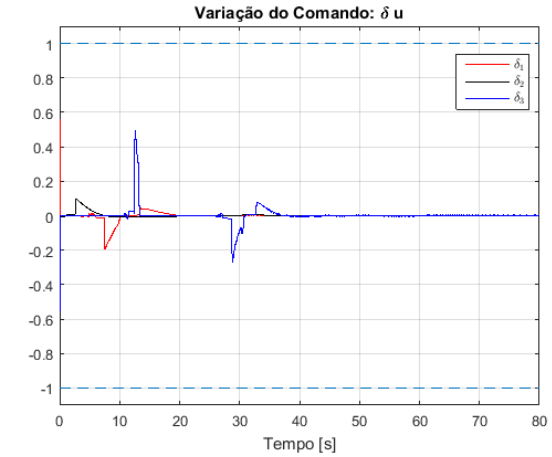
(a) Ângulos de Euler.



(b) Velocidade Angular.



(c) Comando.



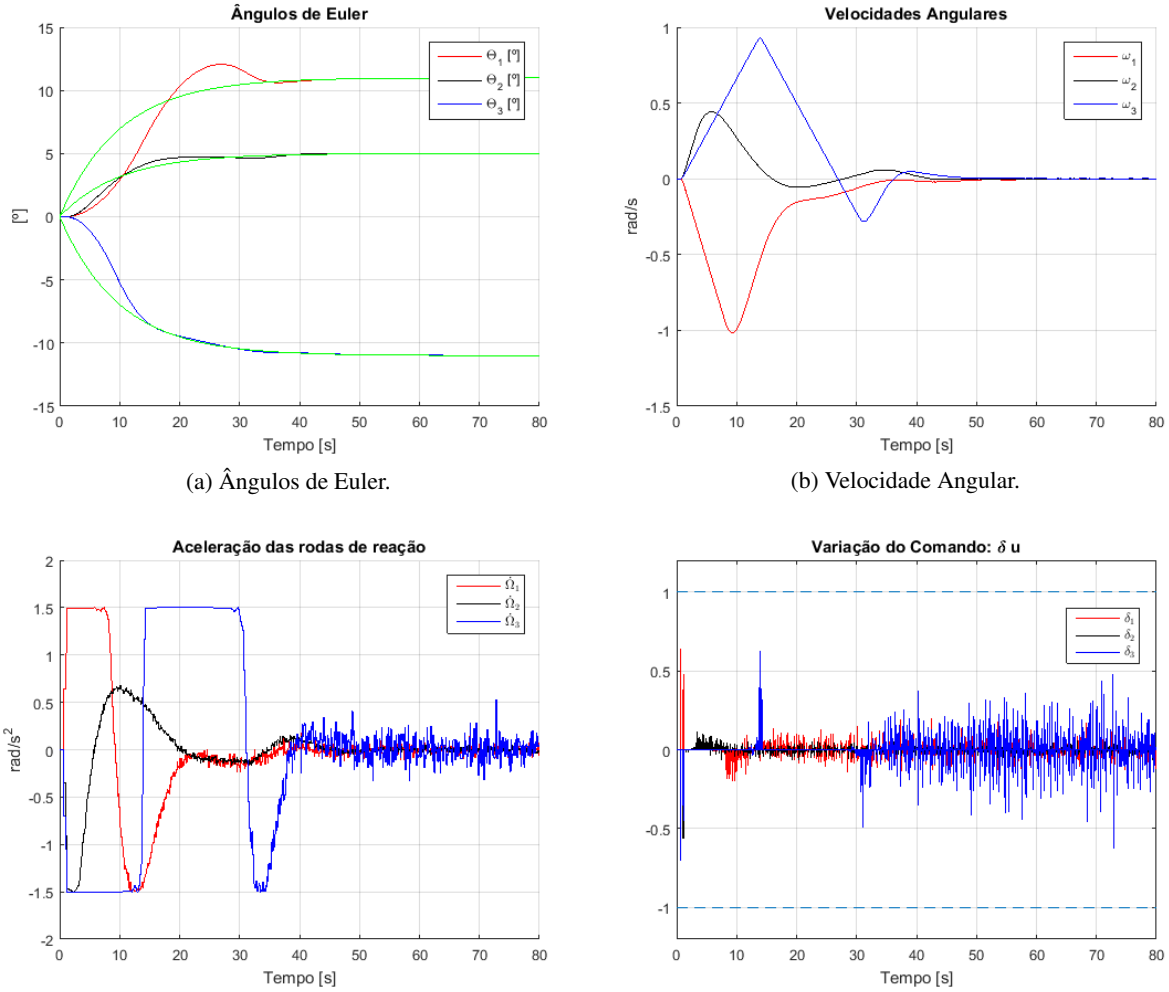
(d) Variação de comando.

Figura 5.22: Validação do MPC em PIL - Referência: $11°, 5°, -11°$.

Pelos resultados obtidos percebe-se que o controlador embarcado apresenta resultados semelhantes ao controlador utilizado na etapa MIL. Porém, nota-se uma diferença no comando gerado pelo MPC, ou seja, na aceleração das rodas de reação, o que impacta consequentemente na variação de comando. As fontes dessas discrepâncias já foram discutidas anteriormente e devem-se ao tratamento numérico entre as arquiteturas de *hardware* onde os testes de MIL e PIL foram realizados. Outro fator importante são as diferenças entre os *solvers quadprog* e *qpOASES*. Apesar das diferenças é possível determinar que o código do controlador desenvolvido em C/C++ foi corretamente projetado pois foi capaz de respeitar as restrições impostas e estabilizou o sistema na referência escolhida. A amplitude dos sinais correspondentes aos ângulos de *euler* e as velocidades angulares apresentaram comportamento semelhantes em MIL e PIL. Uma vez que o controlador foi validado nesta etapa prossegue-se para validação do mesmo na plataforma HIL em tempo real.

5.7.3 Controlador preditivo linear na *BB* em C/C++ e Modelo não linear no *target* - HIL

O último cenário para validação do MPC é o teste realizado na plataforma HIL. O código do controlador continua embarcado na *BeagleBone* como na etapa PIL, acrescido somente de funções de leitura analógica dos estados do *target* e envio de comando via UDP. Apenas o modelo não linear da plataforma é colocado na máquina *target* para simulação em tempo real. A figura abaixo mostra os resultados obtidos através das simulações na plataforma HIL.



(a) Ângulos de Euler.

(b) Velocidade Angular.

(c) Comando.

(d) Variação de comando.

Figura 5.23: Validação do MPC em HIL - Referência: $11°, 5°, -11°$.

Comparando-se os cenários HIL e PIL determina-se que o controlador foi capaz de controlar o sistema simulado em tempo real. Do resultado apresentado pela figura 5.23 percebe-se que o sinal de controle continua oscilando em torno do zero, mesmo quando o estados já atingiram a referência. Estes ruídos presentes no sinal de controle advém de duas fontes. A primeira é da própria passagem dos valor dos estados do modelo simulado no *target* para a placa de aquisição de dados, pelo efeito da conversão digital analógica. A segunda vem da leitura dos estados feita pela *BeagleBone*. A função de conversão entre unidade de tensão e unidades de engenharia, implementada na *BeagleBone*, introduz erros de arredondamento numérico devido a especificidades do próprio *hardware*. Note-se que apesar da existência

de ruído de processo o controlador mostrou-se robusto suficiente para controlar o sistema, respeitando as restrições de comando e da variação de comando.

5.8 Validação do controlador NMPC

Nesta seção prossegue-se com as simulações do controlador preditivo não linear. Da mesma maneira que para o controlador MPC, aqui será feita uma comparação inicial entre o SDRE e o NMPC em ambiente de simulação MIL. O objetivo é avaliar as diferenças entre os controladores não lineares no que diz respeito às restrições impostas pelo modelo. O desenvolvimento do NMPC é necessário para lidar com regiões onde o MPC não foi capaz de controlar o modelo do sistema, ou seja, para referências de grande amplitude, como demonstrado nos testes da etapa MIL do MPC. Finalizada a comparação entre os controladores não lineares na etapa MIL, o controlador NMPC será avaliado nas seguintes etapas do *V-Cycle*:

- Controlador preditivo não linear na *BB* em C/C++ e modelo não linear no *Matlab* - PIL.
- Controlador preditivo não linear na *BB* em C/C++ e modelo não linear no *target* - HIL.

Os controladores NMPC e SDRE serão comparados para a seguinte referência filtrada $\theta_1 = +50°$, $\theta_2 = -30°$, $\theta_3 = +60°$ com $t_{ref} = 30$. Depois da comparação realizada os testes de validação seguirão apenas com o controlador NMPC. Para isto serão feitas duas simulações tanto na etapa PIL quanto HIL. O primeiro teste será para referência angular de baixa amplitude $\theta_1 = +11°$, $\theta_2 = +5°$, $\theta_3 = -11°$ e uma para referências angulares de grande amplitude $\theta_1 = +50°$, $\theta_2 = -30°$, $\theta_3 = +60°$. As referências são filtradas com $t_{ref} = 30$. O objetivo destas duas simulações é demonstrar que o NMPC funciona para referências de pequenas e grandes amplitudes.

O algoritmo de otimização utilizado será o *Sequential Quadratic Programming (SQP)* baseado no trabalho de Alamir (2013). Este algoritmo contém vários parâmetros de sintonia e que não serão detalhados aqui por fugir do escopo do trabalho. Porém, dentre estes parâmetros um destaca-se pela sua importância no uso do controlador embarcado. Esta é a variável N_{ev}. Esse valor representa o número de vezes em que a função custo é avaliada pelo otimizador em um ciclo de simulação, o que representa a tarefa mais pesada computacionalmente no contexto do NMPC (ALAMIR, 2013). O N_{ev} também está associado com a qualidade da solução ótima encontrada pelo otimizador. Caso uma determinada função custo necessite de um número mínimo de N_{ev} e este valor esteja abaixo desse patamar, soluções sub-ótimas serão encontradas.

O valor de N_{ev} conjuntamente com o valor do horizonte de predição N tem efeitos diretos no aumento ou diminuição do custo computacional do controlador, impactando diretamente no tempo gasto pelo NMPC para executar um ciclo de controle. Logo, estes parâmetros devem ser escolhidos de acordo com a limitação computacional do *hardware* utilizado para embarcar o controlador.

Apesar da função custo do NMPC ter sido apresentada em um formato geral (equação 4.60) percebeu-se que nas simulações em MIL que a penalização de comando não apresentava bons resultados. O sistema ou ficava instável ou os estados apresentavam grandes oscilações em torno das referências escolhidas levando muito tempo para convergirem. Por este motivo escolheu-se não penalizar o comando, logo a função custo a ser utilizada pelo NMPC terá o seguinte formato:

$$\hat{p} := argmin\big[\rho_x . \|X_f\| + \sum_{i=0}^{N} \|Y_{ref} - Y_{pred}\|^2_{Q_y}\big] \tag{5.2}$$

Dada uma função custo não-linear (não convexa) com mínimos locais e globais o algoritmo SQP utiliza funções quadráticas para aproximar a função custo em uma determinada região. Através de um método iterativo os parâmetros da função quadrática são variados até que o mínimo da função custo seja encontrado. Uma descrição de como o algoritmo funciona é mostrado na figura abaixo.

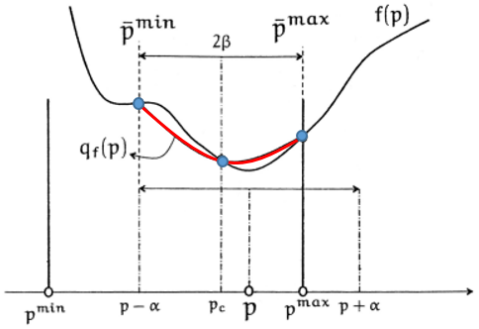


Figura 5.24: Estratégia do algoritmo SQP para otimização de uma função custo f(p).
Fonte: Adaptado de Alamir (2013).

A figura mostra um problema de otimização na variável $p \in [p_{min}, p_{max}]$, em que $p_{min} \neq p_{max}$. Neste trabalho estes parâmetros são constantes e valem $p_{min} = -1$ e $p_{max} = 1$. A busca do valor ótimo da função custo $f(p)$ será feita dentro de um intervalo de confiança definido inicialmente $[\alpha - p(i), \alpha + p(i)]$ onde $p(i)$ é o valor candidato a ótimo global de $f(p)$ no instante i. Esta busca é feita pela função de aproximação quadrática $q_f(p)$, em vermelho na figura 5.24. Comparando-se o mínimo da função quadrática com o mínimo da função custo $f(p)$ no instante i, essa parábola terá seus parâmetros ajustados afim de se aproximar deste mínimo.

O intervalo de confiança inicial para o NMPC foi escolhido como $\alpha = 0.2$. Conforme o algoritmo é executado, este intervalo de confiança pode ser aumentado ou diminuindo segundo regras próprias do algoritmo. Os valores responsáveis pela alteração deste intervalo são β^+ e β^-, que neste trabalho são 2.5 e 0.01, respectivamente. Maiores detalhes acerca do algoritmo SQP podem ser encontrados em Alamir (2013). Os parâmetros utilizados para validação do controlador NMPC podem ser vistos na tabela 5.7.

Tabela 5.7: Parâmetros escolhidos para validação do NMPC

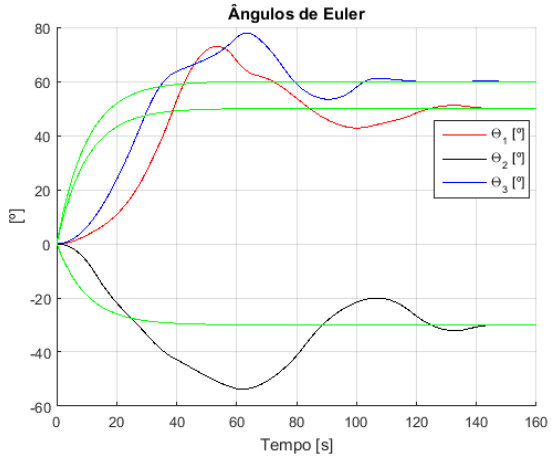
$\mathbf{N}$	75
$\mathbf{Q_y}$	$diag(1 \times 10^2, 1 \times 10^2, 1 \times 10^2, 1, 1, 1)$
$\rho_\mathbf{x}$	1
$\mathbf{n_e}$	2
α	10
λ	0.5
$\mathbf{q}$	8.0
$\mathbf{N_{ev}}$	25

A mesma observação feita para a escolha dos parâmetros de validação do controlador MPC é válida

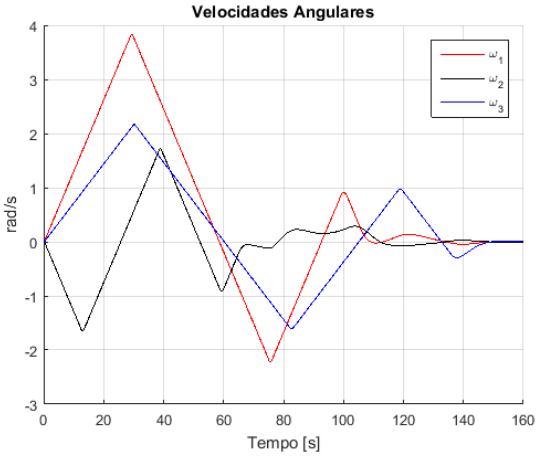
para o NMPC. Eles devem ser testados e validados em todas as etapas, i.e, MIL, PIL e HIL para serem finalmente utilizados. Em comparação com o MPC a escolha destes parâmetros para o NMPC demanda mais tempo. O motivo é o número de parâmetros não somente do controlador mas também do algoritmo de otimização utilizado.

5.8.1 Comparação entre o NMPC e o controlador SDRE no *Matlab* - MIL

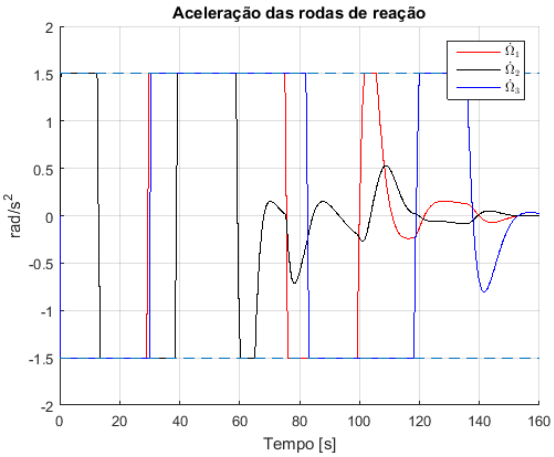
O objetivo desta simulação é comparar o desempenho do SDRE com saturação na variável de comando e o NMPC. O controlador SDRE utiliza a mesma penalização de estados que o NMPC (tabela 5.7) e o mesmo nível de saturação na variável de comando (tabela 5.3). O resultado destas simulações está na figura 5.25. Para este teste será utilizada a referência filtrada $\theta_1 = +50°$, $\theta_2 = -30°$, $\theta_3 = +60°$ com $t_{ref} = 30$. Esta referência foi escolhida por ser a mesma onde o MPC não foi capaz de controlar o sistema. ă ă ă



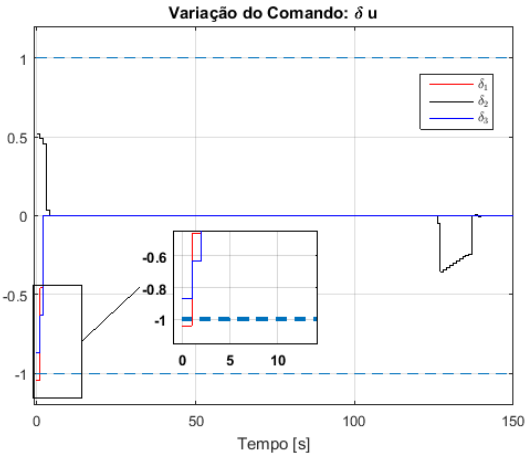
(a) Ângulos de Euler.



(b) Velocidade Angular.



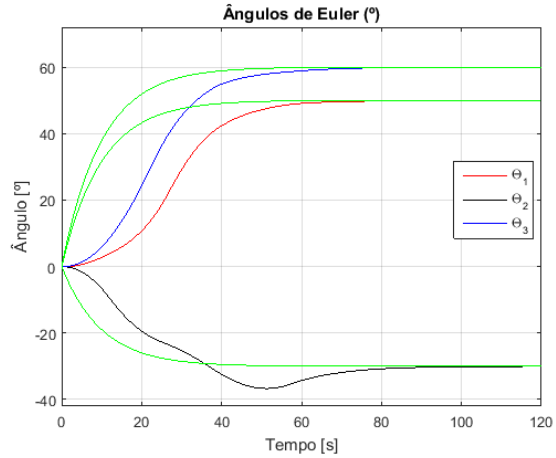
(c) Comando.



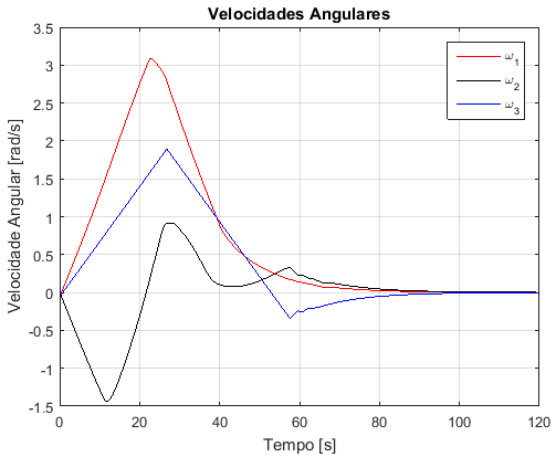
(d) Variação de comando.

Figura 5.25: Validação do SDRE saturado em MIL - Referência: $50°, -30°, 60°$.

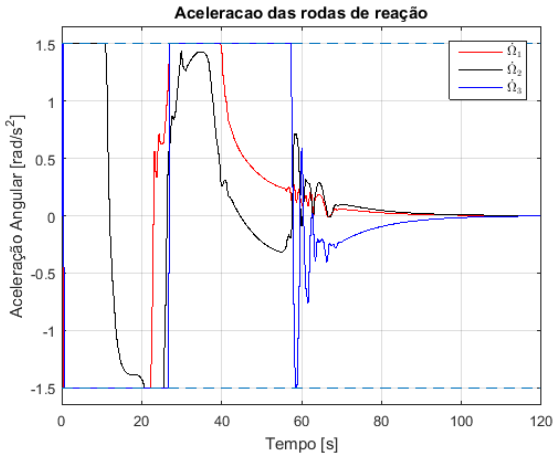
A figura abaixo mostra o resultado para o NMPC quando submetido a mesma referência.



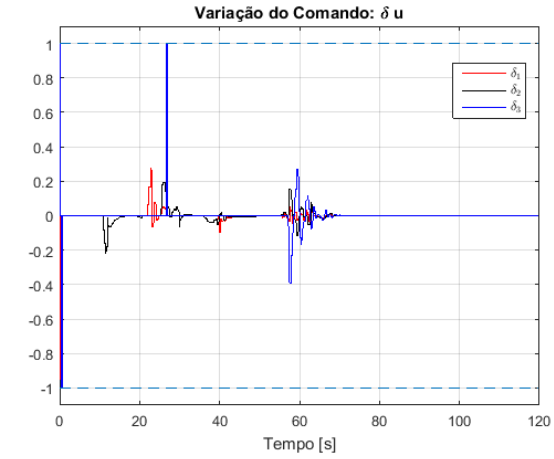
(a) Ângulos de Euler.



(b) Velocidade Angular.



(c) Comando.



(d) Variação de comando.

Figura 5.26: Validação do NMPC em MIL - Referência: $50°, -30°, 60°$.

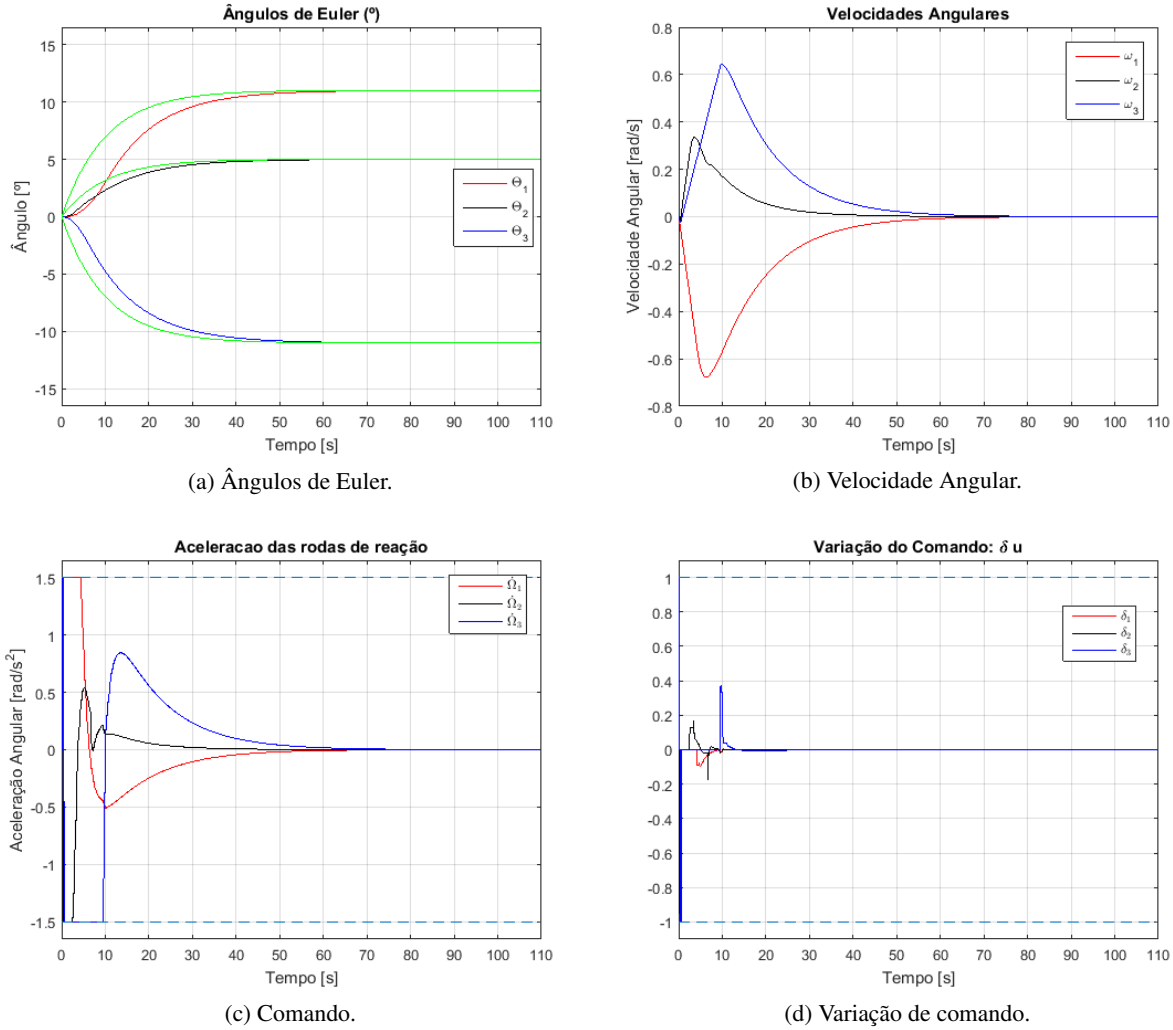
Os resultados demonstram que o NMPC foi capaz de controlar o modelo do sistema para referências de grande amplitude. Tanto o sinal de comando como a sua variação permaneceram dentro das restrições impostas. O ângulo θ_2 apresentou um tempo de assentamento igual a $100s$. Já os ângulos θ_1 e θ_3 possuem tempos de assentamento aproximadamente iguais a $80s$.

O resultado obtido com o controlador SDRE mostra que ele também consegue levar o sistema a estabilidade quando seu sinal de controle é saturado. Porém, os ângulos de *euler* apresentam grandes *overshoot's* e *undershoot's*. Para atingir a completa estabilidade o sistema necessita de um tempo superior a $140s$. Apesar da variação de comando do sinal de controle ter amplitudes menores do que aquelas apresentadas pelo NMPC, logo no início da simulação a variável δ_1 ultrapassa a restrição inferior, mostrado em *zoom* na figura 5.25d.

Da comparação proposta pode-se concluir que o NMPC teve um desempenho superior ao SDRE. Mostrando-se apto a lidar com o problema em estudo. Estabelecida a validade do NMPC na etapa MIL a sua validação será continuada nas etapas PIL e HIL. Antes disso o NMPC será simulado para uma referência de pequena amplitude.

5.8.1.1 Referência $\theta_1 = +11°$, $\theta_2 = +5°$, $\theta_3 = -11°$

O segundo cenário corresponde a mesma referência passada ao controlador MPC e serve de comparação entre o desempenho do MPC e NMPC para um mesmo valor de referência. O resultado é mostrado na figura abaixo.



(a) Ângulos de Euler.

(b) Velocidade Angular.

(c) Comando.

(d) Variação de comando.

Figura 5.27: Validação do NMPC em MIL - Referência: $11°, 5°, -11°$.

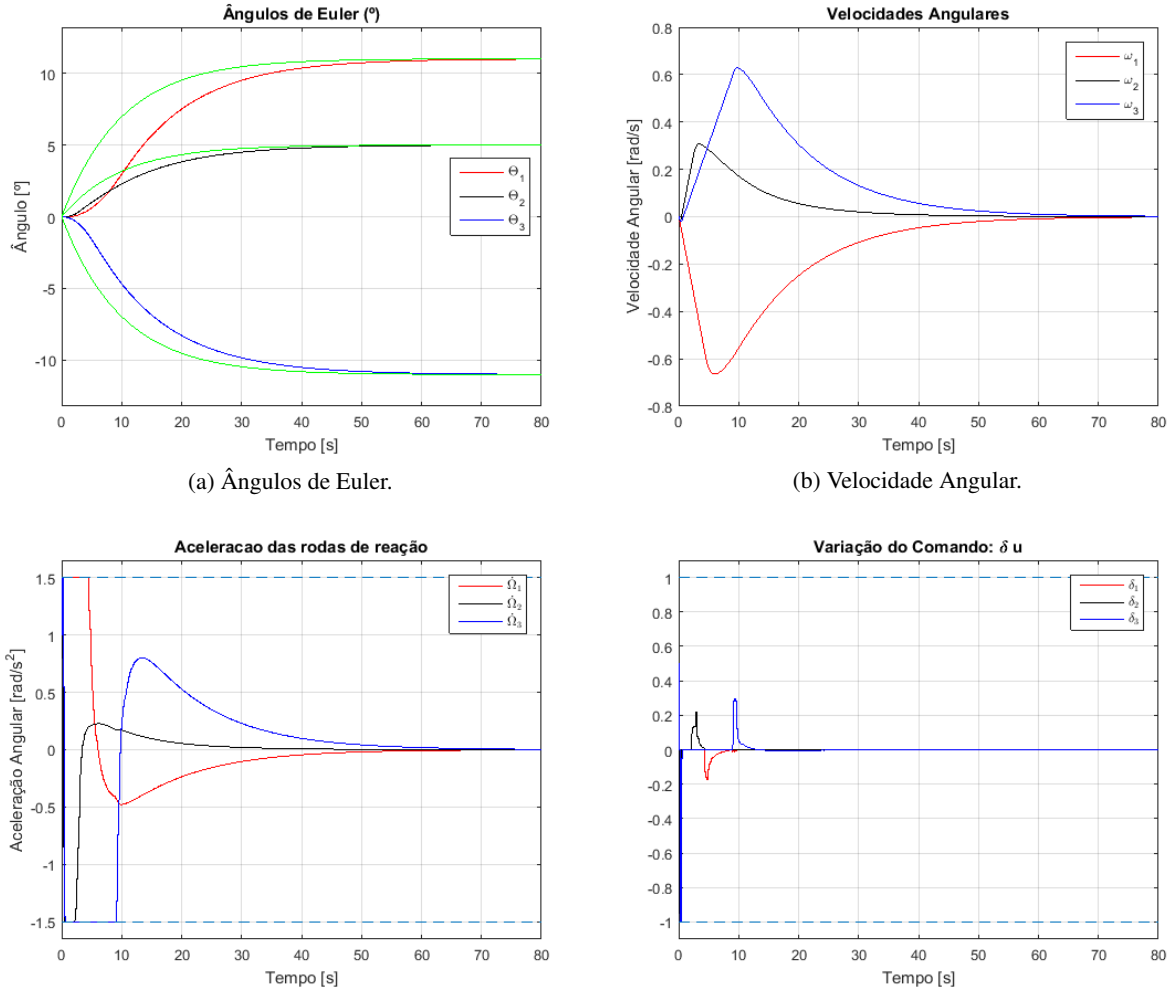
Os resultados mostraram uma resposta melhor do que aquela apresentada pelo controlador MPC (figura 5.20). O *overshoot* do ângulo θ_1 foi reduzido a zero. A amplitude das velocidades angulares da plataforma diminuiu e o sinal de controle chegou ao valor zero antes de $70s$ quando em comparação com o MPC que chegou em $80s$. Outro fator é a diminuição das oscilações da variação de comando para o controlador NMPC.

5.8.2 Controlador preditivo não linear na *BB* em C/C++ e modelo não linear no *Matlab* - PIL

Os testes desta seção tem como objetivo validar o código do controlador NMPC em C/C++ embarcado na *BeagleBone*. O algoritmo de otimização não linear utilizado foi o *SQP*, que foi o mesmo utilizado na etapa MIL, descrito em linguagem de *script* do *Matlab*. Para poder ser embarcado ele foi convertido para C/C++.

5.8.2.1 Referência $\theta_1 = +11°$, $\theta_2 = +5°$, $\theta_3 = -11°$

O cenário PIL com as mesmas referências que foram utilizadas para validar o controlador MPC são mostradas na figura 5.28.



(a) Ângulos de Euler.

(b) Velocidade Angular.

(c) Comando.

(d) Variação de comando.

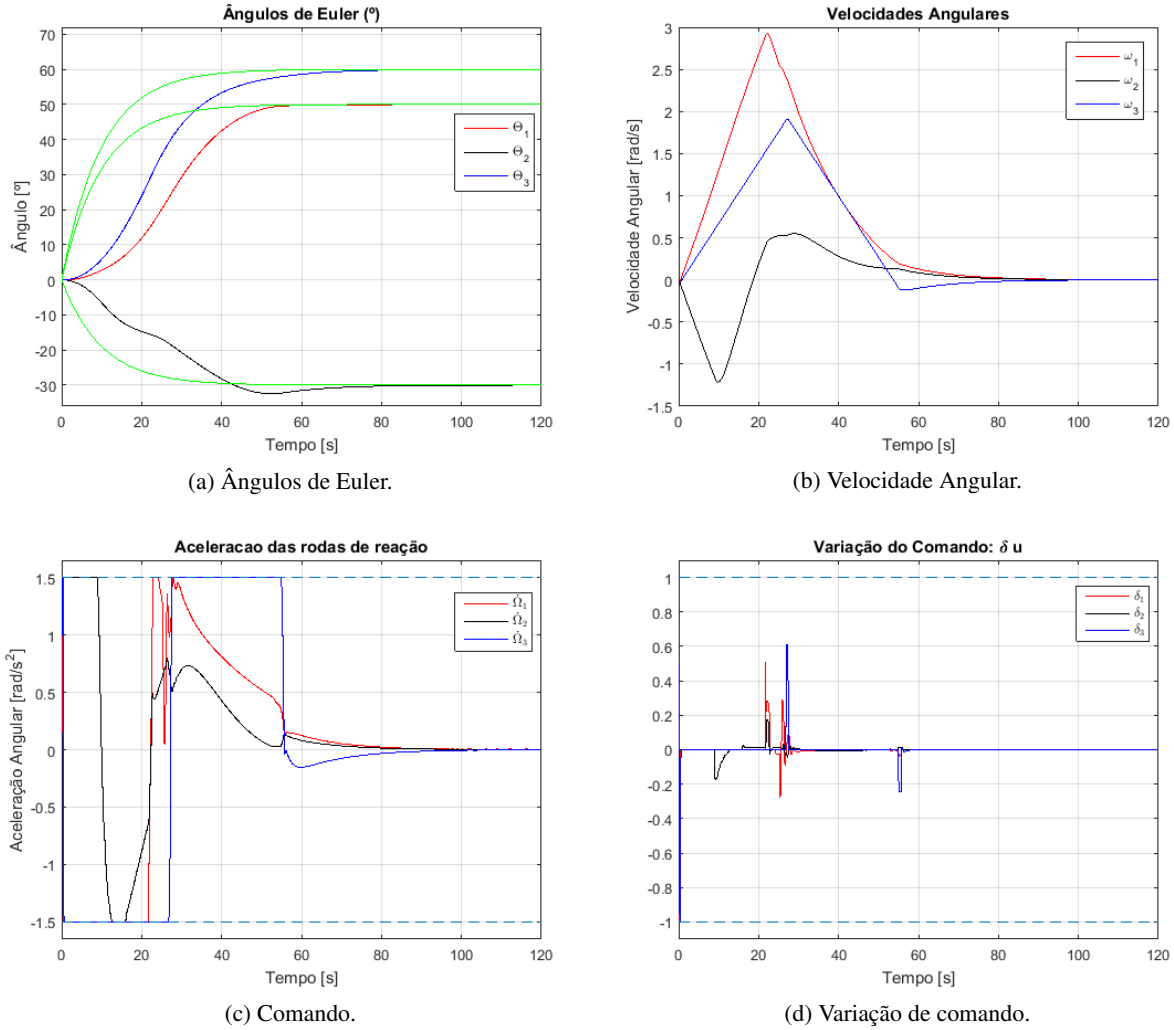
Figura 5.28: Validação do NMPC em PIL - Referência: $11°, 5°, -11°$.

Comparando o resultado anterior com a etapa PIL percebe-se que o comando $\dot{\Omega}_2$ possui uma menor variação em $t = 5s$. Como as acelerações estão praticamente iguais a variação de comando para o caso MIL e PIL também permancem próximos. Os ângulos de *euler* estabilizaram no mesmo tempo da etapa

MIL. As velocidades angulares também apresentaram resultado semelhante a etapa MIL.

5.8.2.2 Referência $\theta_1 = +50°$, $\theta_2 = -30°$, $\theta_3 = 60°$

O resultado em PIL para o cenário de grandes amplitudes angulares é mostrado na figura 5.29.



(a) Ângulos de Euler.

(b) Velocidade Angular.

(c) Comando.

(d) Variação de comando.

Figura 5.29: Validação do NMPC em PIL - Referência: $50°, -30°, 60°$.

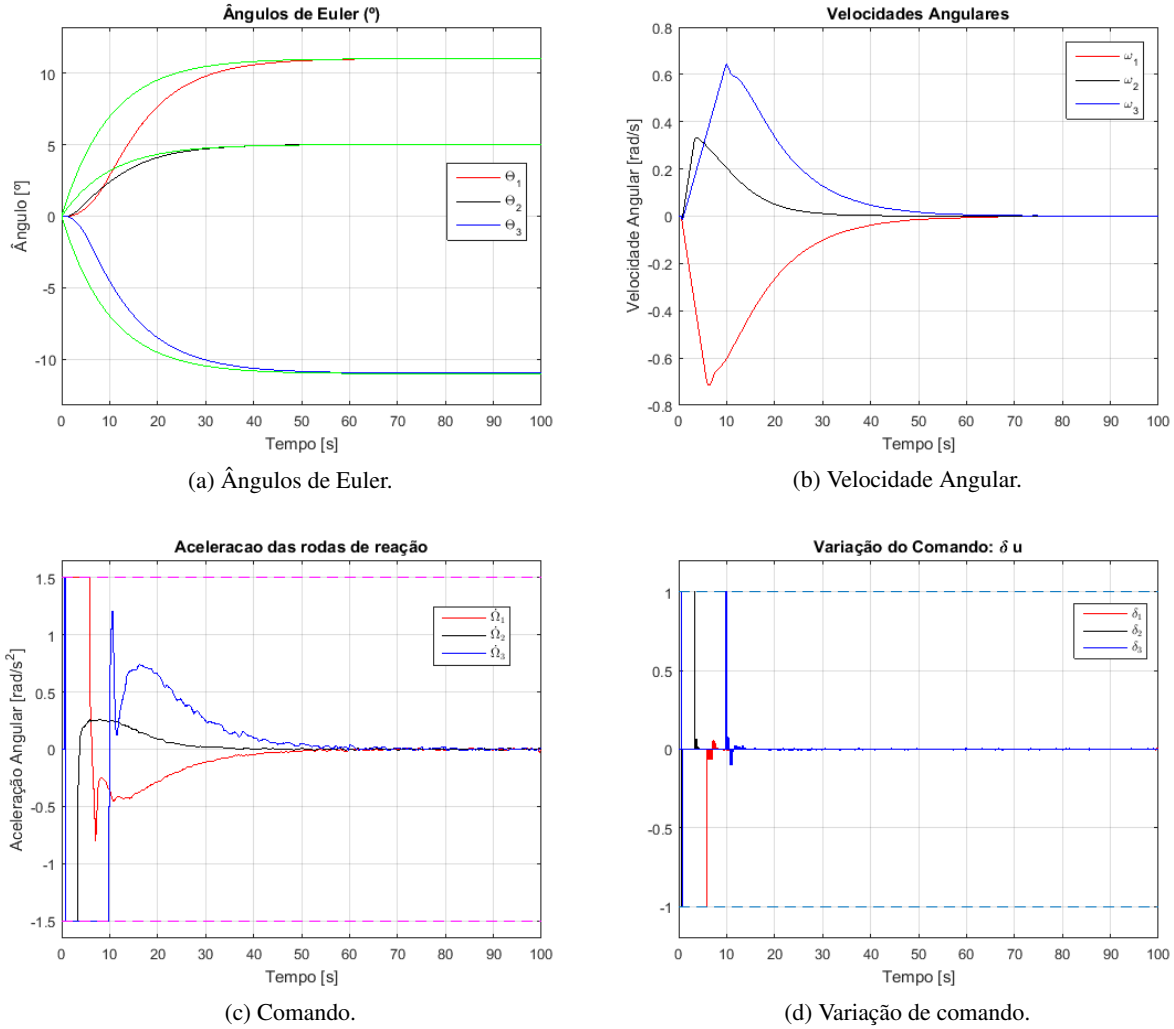
Comparando-se com a simulação em MIL os ângulos de *euler* estabilizaram no mesmo instante de tempo e sem *overshoot*. Quanto as velocidades angulares a única alteração perceptível é um leve acrescimento na amplitude da velocidade angular de ω_2. As diferenças mais acentuadas dizem respeito ao sinal de controle e a sua variação. Nos testes em PIL o comando $\dot{\Omega}_3$ e $\dot{\Omega}_2$ não apresentam oscilações no tempo $t = 60s$ que aparecem na etapa MIL, com amplitudes maiores para $\dot{\Omega}_3$. Quanto a variação de comando as simulações em PIL apresentam uma maior oscilação do que aqueles apresentados na simulação em MIL.

5.8.3 Controlador preditivo não linear na *BB* em C/C++ e modelo não linear no *target* - HIL

O último cenário corresponde aos testes do NMPC embarcado em *hardware* funcionando na plataforma HIL.

5.8.3.1 Referência $\theta_1 = +11°$, $\theta_2 = +5°$, $\theta_3 = -11°$

Os testes em HIL ainda para um cenário de amplitudes angulares reduzidas é mostrado na figura 5.30.



(a) Ângulos de Euler.

(b) Velocidade Angular.

(c) Comando.

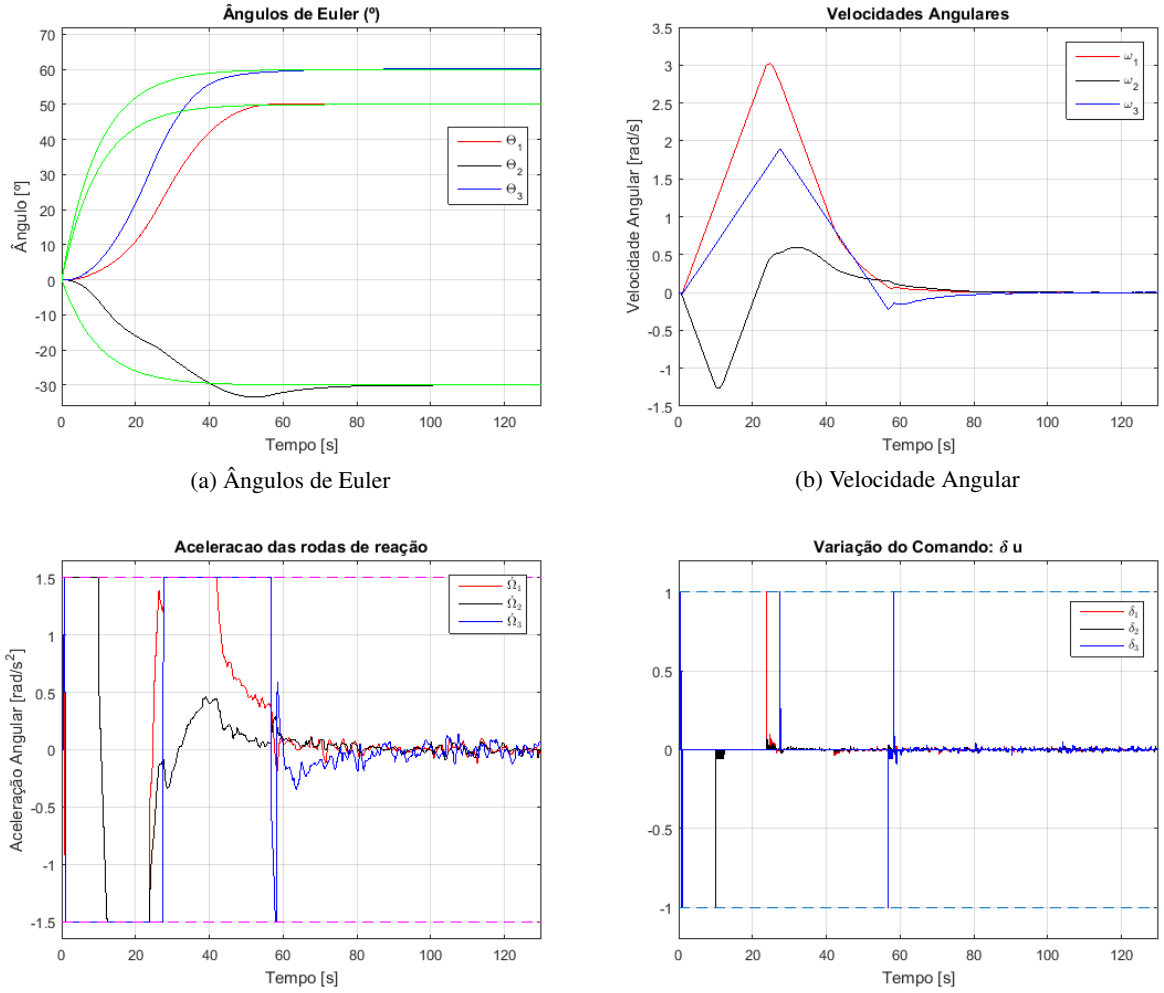
(d) Variação de comando.

Figura 5.30: Validação do NMPC em HIL - Referência $11°, 5°, -11°$.

Do ponto de vista dos estados, estes apresentam comportamento semelhante a aqueles dos testes em PIL. A diferença notável nos testes em HIL é que a variação de comando chega a atingir as restrições impostas pelo NMPC. Outro ponto é o pico do sinal de controle $\dot{\Omega}_1$ no tempo $t = 11s$ que não está presente na etapa PIL.

5.8.3.2 Referência $\theta_1 = +50°$, $\theta_2 = +30°$, $\theta_3 = -60°$

O último cenário para avaliar o controlador NMPC em tempo real é sua aplicação para amplitude angulares elevadas. O resultado das simulações é mostrado na figura 5.31.



(a) Ângulos de Euler

(b) Velocidade Angular

(c) Comando

(d) Variação de comando

Figura 5.31: Validação do NMPC em HIL - Referência $50°, -30°, 60°$

Comparando-se o resultado anterior com o resultado da etapa PIL é perceptível que os ângulos de *euler* e as velocidades angulares se comportaram de maneira igual. A maior diferença reside no sinal de comando que é muito mais variável para as simulações HIL, inclusive as amplitudes destes em regime permanente. As variações de comando também são mais acentuadas pois chegam a alcançar as restrições impostas em vários instantes de simulação, o que não ocorre no caso das simulações em PIL. Nota-se que mesmo em face dos ruídos do processo o controlador NMPC conseguiu controlar o sistema, levando os estados para a referência desejada e respeitando todas as restrições impostas pelo projeto. Conclui-se então que o controlador NMPC funciona corretamente tanto para os casos de referências angulares de pequena e grande amplitude.

5.9 Tempo de cálculo dos controladores embarcados

O tempo de cálculo associado a cada um dos controladores preditivos embarcados na *BeagleBone* são mostrados na tabela 5.8.

Tabela 5.8: Tempo de cálculo dos controladores

Controlador	**Tempo de cálculo (ms)**
MPC	48
NMPC	61

Os resultados mostram que tanto o tempo de cálculo do MPC quanto do NMCP foram inferiores ao período de amostragem escolhido de 100ms. O tempo de cálculo para o NMPC foi superior ao MPC devido aos aspectos discutidos anteriormente, como por exemplo, a existência de um problema de otimização de uma função custo não convexa.

Os tempo obtidos, dentro do período de amostragem escolhido, só foram possíveis através do uso das técnicas de parametrização de comando. Este é um importante resultado pois demonstra a possibilidade da utilização de controladores preditivos embarcados em um *hardware* de baixo custo e limitado poder de processamento.

Capítulo 6

Conclusões

O presente trabalho teve como o objetivo o desenvolvimento de um controlador preditivo para o controle de atitude de satélites, através da combinação do *Model-Based Design* e do *V-cycle*. As etapas MIL, SIL, PIL e HIL desta metodologia de trabalho mostraram-se extremamente úteis na validação dos controladores. Sua grande vantagem surge da possibilidade de desacoplamento das diferentes partes de desenvolvimento do projeto, facilitando a detecção e correção de erros.

A etapa MIL forneceu um meio de testar diferentes tipos de cenários com variados parâmetros de sintonia do controlador, fornecendo um meio mais fácil de encontrar parâmetros de sintonia segundo as necessidades do projeto. A etapa SIL fornece uma metodologia da criação do código do controlador, seja ele por desenvolvimento manual ou geração automática. Por sua vez a etapa PIL atua como uma ponte entre as simulações em *software* e a aplicação em tempo real, através dos testes de código embarcado em *hardware*. Este teste permite verificar a viabilidade do *software* embarcado, em termos de tempo de cálculo e uso de memória. Por fim a plataforma HIL permite o teste final do controlador em um ambiente de tempo real.

Para os controladores preditivos desenvolvidos, foi utilizada a parametrização exponencial, que mostrou-se extremamente útil no momento de embarcar os controladores em *hardware*. Isto possibilitou o uso de horizontes de predição relativamente altos, mesmo em um *hardware* de baixo poder computacional.

A metodologia de desenvolvimento de controladores mostrou-se eficaz na validação dos controladores. Pode-se concluir que os objetivos da dissertação foram atingidos, pois os resultados obtidos com os controladores foram satisfatórios, seja nas simulações em ambiente de *hardware* ou *software*. Tanto o MPC quanto o NMPC foram capazes de lidar com as restrições impostas pelo modelo do sistema em estudo em situações de tempo real. Os controladores também mostraram-se robustos pelos testes efetuados na plataforma HIL, devido aos ruídos introduzidos na malha de controle pela leitura do estados do modelo simulado no *target*.

Pode-se destacar que a contribuição deste trabalho foi embarcar as estratégias de controle preditivo linear e não linear parametrizadas em um *hardware* de baixo custo e baixo poder de processamento para o controle de atitude de um modelo de plataforma de testes de satélites. Essa realização demonstra que apesar do alto custo computacional inerente a estratégia escolhida ela pode ser utilizada em situações práticas, o que ainda é visto na literatura como um problema complexo, principalmente no caso do NMPC.

Outra contribuição foi a criação de uma plataforma de baixo custo para avaliação de controladores em situações e tempo real. A plataforma é versátil o bastante para ser utilizada na avaliação de outros modelos e estratégias de controle.

As contribuições futuras que podem ser feitas a este trabalho são as seguintes:

- Acrescentar o modelo eletromecânico da roda de reação utilizando parâmetros mais próximos da realidade.
- Utilizar um estimador do tipo filtro de *Kalman* para estimar as velocidades angulares do modelo do satélite.
- Incluir nas simulações o efeito das pertubações existentes no espaço como pressão solar, campo magnético e arrasto aerodinâmico.
- Utilizar um sistema operacional em tempo real (RTOS) na *BeagleBone*.
- Modificar a comunicação entre a *Beaglebone Black* e a máquina *target*, substituindo o envio do sinal de controle via UDP por uma comunicação puramente analógica.
- Utilizar outros algoritmo de otimização para o NMPC, como por exemplo o *Nlopt* ou algoritmos bio-inspirados.
- Implementar os algoritmos de controle preditivo em outras plataformas e arquiteturas, como FPGA. Visando melhorar o desempenho dos controladores.

REFERÊNCIAS BIBLIOGRÁFICAS

ABBOTT, D. *Linux for embedded and real-time applications*. [S.l.]: Elsevier, 2011.

AEB. 2011. <http://www.aeb.gov.br/pais-e-maior-distribuidor-de-imagens-por-satelite-do-mundo/>. Acessado: 10-04-2016.

AIKENHEAD, B. A.; DANIELL, R. G.; DAVIS, F. M. Canadarm and the space shuttle. *Journal of Vacuum Science & Technology A: Vacuum, Surfaces, and Films*, AVS, v. 1, n. 2, p. 126–132, 1983.

ALAMIR, M. *Stabilization of nonlinear systems using receding-horizon control schemes: a parametrized approach for fast systems*. [S.l.]: Springer, 2006. v. 339.

ALAMIR, M. *A Pragmatic Story of Model Predictive Control: Self-Contained Algorithms and Case-Studies*. [S.l.]: CreateSpace Independent Publishing Platform, 2013.

ALLEN, L. et al. The hubble space telescope optical systems failure report. *NASA Report*, 1990. Disponível em: <https://www.ssl.berkeley.edu/~mlampton/AllenReportHST.pdf>.

AURET, J. *Design of an aerodynamic attitude control system for a CubeSat*. Tese (Doutorado) — Stellenbosch: Stellenbosch University, 2012.

AUTHOR, T. T. *Evolution of Microprocessors - From 4004 to Core i7*. 2018. Disponível em: <http://www.thetechauthor.com/evolution-of-microprocessors-from-4004-to-core-i7/>.

BAYAT, F. Conceptual design of a low-cost real-time hardware-in-the-loop simulator for satellite attitude control system. *Turkish Journal of Electrical Engineering and Computer Science*, v. 23, n. 3, p. 789, 2015.

BEMPORAD, A. et al. The explicit solution of model predictive control via multiparametric quadratic programming. In: *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No.00CH36334)*. [S.l.: s.n.], 2000. v. 2, p. 872–876 vol.2. ISSN 0743-1619.

BERGSTRÖM, D.; GÖRANSSON, R. Model-and hardware-in-the-loop testing in a model-based design workflow. *MSc Theses*, 2016.

BONING, P. et al. An experimental study of the control of space robot teams assembling large flexible space structures. In: *Proc. of the 9th international symposium on artificial intelligence, robotics and automation in space*. [S.l.: s.n.], 2008.

BOYD, S.; VANDENBERGHE, L. *Convex optimization*. [S.l.]: Cambridge university press, 2004.

CAMACHO, E. F.; ALBA, C. B. *Model predictive control*. [S.l.]: Springer Science & Business Media, 2007.

CAMPESATO, W. L. et al. Projeto, implementação e controle de uma plataforma aerostática com jatos de ar. *Proceeding Series of the Brazilian Society of Computational and Applied Mathematics*, v. 1, n. 1, 2013.

CHEN, X. *Design and Implementation of Model Predictive Control Algorithms for Small Satellite Three-Axis Stabilization*. Tese (Doutorado) — The University of Sidney, 3 2011.

CHEN, X.; WU, X. Model predictive control of cube satellite with magneto-torquers. In: *The 2010 IEEE International Conference on Information and Automation*. [S.l.: s.n.], 2010. p. 997–1002.

CHRISTOFIDES, P. D. et al. Distributed model predictive control: A tutorial review and future research directions. *Computers & Chemical Engineering*, v. 51, p. 21 – 41, 2013. ISSN 0098-1354.

CORPINO, S.; STESINA, F. Verification of a cubesat via hardware-in-the-loop simulation. *IEEE Transactions on Aerospace and Electronic Systems*, v. 50, n. 4, p. 2807–2818, October 2014. ISSN 0018-9251.

ĆOSIĆ, K. et al. Design and implementation of a hardware-in-the-loop simulator for a semi-automatic guided missile system. *Simulation Practice and Theory*, Elsevier, v. 7, n. 2, p. 107–123, 1999. ISSN 0928-4869.

CUBILLOS, X. C. M.; SOUZA, L. C. G. de. Using of h-infinity control method in attitude control system of rigid-flexible satellite. *Mathematical Problems in Engineering*, Hindawi, v. 2009, 2009.

CURRIE, J. *Practical applications of industrial optimization: from high-speed embedded controllers to large discrete utility systems*. Tese (Doutorado) — Auckland University of Technology, 2014.

CUTLER, C.; MORSHEDI, A.; HAYDEL, J. An industrial perspective on advanced control. In: *AICHE annual meeting*. [S.l.: s.n.], 1983.

CUTLER, C. R.; RAMAKER, B. L. Dynamic matrix control: A computer control algorithm. In: *joint automatic control conference*. [S.l.: s.n.], 1980. p. 72.

DAIGNEAU, M.; ADVENA, M. *Beaglebone Black Architecture*. 2018. Disponível em: <http://http://meseec.ce.rit.edu/551-projects/spring2015/2-3.pdf>.

DAS, A. et al. Astrex-a unique test bed for csi research. In: IEEE. *Decision and Control, 1990., Proceedings of the 29th IEEE Conference on*. [S.l.], 1990. p. 2018–2023.

DIEHL, M.; FERREAU, H. J.; HAVERBEKE, N. Efficient numerical methods for nonlinear mpc and moving horizon estimation. In: *Nonlinear model predictive control*. [S.l.]: Springer, 2009. p. 391–417.

DUBOIS, C. *Programming Languages for Embedded Systems 101: Background and Resources*. 2018. Disponível em: <https://www.allaboutcircuits.com/news/programming-languages-for-embedded-systems-101-background-and-resources/>.

EIGEN. 2018. Disponível em: <http://eigen.tuxfamily.org/index.php?title=Main_Page>.

EREN, U. et al. Model predictive control in aerospace systems: Current state and opportunities. *Journal of Guidance, Control, and Dynamics*, American Institute of Aeronautics and Astronautics, 2017.

FALL, K. R.; STEVENS, W. R. *TCP/IP illustrated, volume 1: The protocols*. [S.l.]: addison-Wesley, 2011.

FERREAU, H.; POTSCHKA, A.; KIRCHES, C. *qpOASES webpage*. 2007–2017. Http://www.qpOASES.org/.

FERREAU, H. J. et al. Survey of industrial applications of embedded model predictive control. In: *2016 European Control Conference (ECC)*. [S.l.: s.n.], 2016. p. 601–601.

FRANCHI, L. et al. Model predictive and reallocation problem for cubesat fault recovery and attitude control. *Mechanical Systems and Signal Processing*, v. 98, n. Supplement C, p. 1034 – 1055, 2018. ISSN 0888-3270.

FRENCH, D. B. *Hybrid control strategies for rapid, large angle satellite slew maneuvers*. [S.l.], 2003.

FRITZ, M. et al. Hardware-in-the-loop environment for verification of a small satellite's on-board software. *Aerospace Science and Technology*, v. 47, p. 388 – 395, 2015. ISSN 1270-9638. Disponível em:

<http://www.sciencedirect.com/science/article/pii/S1270963815002783>.

GALLARDO, D.; BEVILACQUA, R. Six degrees of freedom experimental platform for testing autonomous satellites operations. In: *8th International ESA Conference on Guidance, Navigation and Control Systems*. [S.l.: s.n.], 2011.

GILZ, P. R. A. et al. Model predictive control for rendezvous hovering phases based on a novel description of constrained trajectories. *IFAC-PapersOnLine*, v. 50, n. 1, p. 7229 – 7234, 2017. ISSN 2405-8963. 20th IFAC World Congress.

GONZALES, R. G. *Utilização dos métodos SDRE e Filtro de Kalman para o controle de atitude de simuladores de satélites*. Dissertação (Mestrado) — Dissertação de Mestrado do Curso de Pós-Graduação em Engenharia e Tecnologia Espacial, DEM/INPE, São Jose dos Campos, SP, 2009.

GOODYEAR, A. et al. Hardware implementation of model predictive control for relative motion maneuvering. In: *2015 American Control Conference (ACC)*. [S.l.: s.n.], 2015. p. 2311–2316. ISSN 0743-1619.

GROS, S. et al. From linear to nonlinear mpc: bridging the gap via the real-time iteration. *International Journal of Control*, Taylor & Francis, v. 0, n. 0, p. 1–19, 2016.

GUAN, P.; LIU, X.-J.; LIU, J.-Z. Adaptive fuzzy sliding mode control for flexible satellite. *Engineering Applications of Artificial Intelligence*, v. 18, n. 4, p. 451 – 459, 2005. ISSN 0952-1976.

HEGRENÆS Øyvind; GRAVDAHL, J. T.; TøNDEL, P. Spacecraft attitude control using explicit model predictive control. *Automatica*, v. 41, n. 12, p. 2107 – 2114, 2005. ISSN 0005-1098.

ILKIV, B. R.; RUSKO, M. Experimental verification of stabilizing spline-based continuous-time model predictive control scheme with adaptation of terminal set. *IFAC Proceedings Volumes*, Elsevier, v. 38, n. 1, p. 385–390, 2005.

ISERMANN, R.; SCHAFFNIT, J.; SINSEL, S. Hardware-in-the-loop simulation for the design and testing of engine-control systems. *Control Engineering Practice*, v. 7, n. 5, p. 643 – 653, 1999. ISSN 0967-0661.

IZADI, V.; ABEDI, M.; BOLANDI, H. Verification of reaction wheel functional model in hil test-bed. In: IEEE. *Control, Instrumentation, and Automation (ICCIA), 2016 4th International Conference on*. [S.l.], 2016. p. 155–160.

JENSEN, K. F.; VINTHER, K. Attitude determination and control system for aausat3. *Master's Thesis, Aalborg University*, 2010.

JIAN, X. et al. Design and development of a 5-dof air-bearing spacecraft simulator. In: *2009 International Asia Conference on Informatics in Control, Automation and Robotics*. [S.l.: s.n.], 2009. p. 126–130. ISSN 1948-3414.

JIANG, Z. et al. Processor-in-the-loop simulation, real-time hardware-in-the-loop testing, and hardware validation of a digitally-controlled, fuel-cell powered battery-charging station. In: IEEE. *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*. [S.l.], 2004. v. 3, p. 2251–2257.

JOHANSEN, T. A. Introduction to nonlinear model predictive control and moving horizon estimation. *Selected Topics on Constrained and Nonlinear Control*, v. 1, p. 1–53, 2011.

JR, G. A.; SANTANA, A. C.; MARTINS-FILHO, L. S. Dynamics and control of three-axis satellites by thruster actuators using a linear quadratic regulator. In: *Proc. of International Congress of Mechanical ScienceâĂİ, Brasilia*. [S.l.: s.n.], 2007. p. 1–9.

KELEMENOVÁ, T. et al. Model based design and hil simulations. *American Journal of Mechanical Engineering*, v. 1, n. 7, p. 276–281, 2013.

KIM, B. et al. Designing a low-cost spacecraft simulator. *IEEE Control Systems*, v. 23, n. 4, p. 26–37, Aug 2003.

KUMAR, E. V.; JEROME, J.; RAAJA, G. State dependent riccati equation based nonlinear controller design for ball and beam system. *Procedia Engineering*, Elsevier, v. 97, p. 1896–1905, 2014.

LAILA, D. S. *Design and analysis of nonlinear sampled data control systems*. [S.l.]: University of Melbourne, Department of Electrical and Electronic Engineering, 2003.

LEDIN, J. A. Hardware-in-the-loop simulation. *Embedded Systems Programming*, MILLER FREEMAN INC., v. 12, p. 42–62, 1999.

LEE, J. H. Model predictive control: Review of the three decades of development. *International Journal of Control, Automation and Systems*, Springer, v. 9, n. 3, p. 415, 2011.

LI, L. et al. Robust attitude control for flexible satellite during orbit maneuver. In: *Proceedings of 2014 IEEE Chinese Guidance, Navigation and Control Conference*. [S.l.: s.n.], 2014. p. 2531–2536.

LI, Q. et al. Model predictive control for autonomous rendezvous and docking with a tumbling target. *Aerospace Science and Technology*, v. 69, n. Supplement C, p. 700 – 711, 2017. ISSN 1270-9638.

LI, Y.; GAO, Y. Compare tabletop style platform with umbrella style platform of air bearings spacecraft simulator. In: *2010 3rd International Symposium on Systems and Control in Aeronautics and Astronautics*. [S.l.: s.n.], 2010. p. 252–255.

LIN, T. Y.; JUANG, J. C.; VINA. Design and verification of the operating procedure of attitude determination and control subsystem of a nanosatellite. In: *2014 CACS International Automatic Control Conference (CACS 2014)*. [S.l.: s.n.], 2014. p. 51–56.

LING, K. V. et al. Multiplexed mpc for multizone thermal processing in semiconductor manufacturing. *IEEE Transactions on Control Systems Technology*, v. 18, n. 6, p. 1371–1380, Nov 2010. ISSN 1063-6536.

LING, K. V. et al. Multiplexed model predictive control. *Automatica*, v. 48, n. 2, p. 396 – 401, 2012. ISSN 0005-1098. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0005109811005280>.

MATASARU, M. Synchronisation of two dc servo srv-02et motors using xpc target. In: IEEE. *System Theory, Control and Computing (ICSTCC), 2013 17th International Conference*. [S.l.], 2013. p. 337–342.

MATHWORKS. *Build, run, and test real-time applications*. 2018. Disponível em: <https://www.mathworks.com/products/simulink-real-time.html>.

MATHWORKS. *Why Adopt Model-Based Design for Embedded Control Software Development?* 2018. Disponível em: <https://www.mathworks.com/content/dam/mathworks/mathworks-dot-com/campaigns/portals/files/adopting-model-based-design/why-adopt-model-based-design-for-embedded-control-software-development.pdf>.

MATINNEJAD, R. et al. Search-based automated testing of continuous controllers: Framework, tool support, and case studies. *Information and Software Technology*, Elsevier, v. 57, p. 705–722, 2015.

MAYNE, D. et al. Constrained model predictive control: Stability and optimality. *Automatica*, v. 36, n. 6, p. 789 – 814, 2000. ISSN 0005-1098.

MIN, H.; GUOQIANG, Z.; YUDONG, G. Thruster control for micro-satellite attitude and hardware-in-the-loop demonstration. In: IEEE. *Industrial Control and Electronics Engineering (ICICEE), 2012 International Conference on*. [S.l.], 2012. p. 588–591.

MINA, J. et al. Processor-in-the-loop and hardware-in-the-loop simulation of electric systems based in fpga. In: *2016 13th International Conference on Power Electronics (CIEP)*. [S.l.: s.n.], 2016. p. 172–177.

MURILO, A. *Contributions on Nonlinear Model Predictive Control for Fast Systems*. Tese (Theses) —

Institut National Polytechnique de Grenoble - INPG, dez. 2009.

MURILO, A.; ALAMIR, M.; ALBERER, D. A general nmpc framework for a diesel engine air path. *International Journal of Control*, Taylor & Francis, v. 87, n. 10, p. 2194–2207, 2014.

NABI, S. et al. *An overview of hardware-in-the-loop testing systems at Visteon*. [S.l.], 2004.

NOW, S. *Attitude control failures led to break-up of Japanese astronomy satellite*. 2017. Disponível em: <https://spaceflightnow.com/2016/04/18/spinning-japanese-astronomy-satellite-may-be-beyond-saving/ >.

NUDEHI, S. S. et al. Satellite attitude control using three reaction wheels. In: IEEE. *American Control Conference, 2008*. [S.l.], 2008. p. 4850–4855.

OHTSUKA, T.; OZAKI, K. Practical issues in nonlinear model predictive control: real-time optimization and systematic tuning. In: *Nonlinear Model Predictive Control*. [S.l.]: Springer, 2009. p. 447–460.

OLIVEIRA, A.; KUGA, H.; CARRARA, V. Air bearing platforms for simulation of spacecraft attitude control systems. 02 2015.

PARK, H.-E. et al. Integrated orbit and attitude hardware-in-the-loop simulations for autonomous satellite formation flying. *Advances in Space Research*, v. 52, n. 12, p. 2052 – 2066, 2013. ISSN 0273-1177.

PECK, M. A. et al. An airbearing-based testbed for momentum control systems and spacecraft line of sight. *Advances in the Astronautical Sciences*, Citeseer, v. 114, p. 427–446, 2003.

PIROUZMAND, F.; GHAHRAMANI, N. O. Robust model predictive control based on mras for satellite attitude control system. In: *The 3rd International Conference on Control, Instrumentation, and Automation*. [S.l.: s.n.], 2013. p. 53–58.

PRADO, J. et al. Three-axis air-bearing based platform for small satellite attitude determination and control simulation. *Journal of Applied Research and Technology*, UNAM, Centro de Ciencias Aplicadas y Desarrollo Tecnológico, v. 3, n. 3, p. 222–237, 2005.

PTAK, A.; FOUNDY, K. Real-time spacecraft simulation and hardware-in-the-loop testing. In: IEEE. *Real-Time Technology and Applications Symposium, 1998. Proceedings. Fourth IEEE*. [S.l.], 1998. p. 230–236.

RAZIEI, S. A.; JIANG, Z. Model predictive control for complex dynamic systems. In: *2016 IEEE National Aerospace and Electronics Conference (NAECON) and Ohio Innovation Summit (OIS)*. [S.l.: s.n.], 2016. p. 193–200.

RICHALET, J. et al. Algorithmic control of industrial processes. *In Proceedings of 4th IFCA Symposium on Identification and System Parameter Estimation*, p. 1119–1167, 1976.

RICHALET, J. et al. Model predictive heuristic control. *Automatica (Journal of IFAC)*, Pergamon Press, Inc., v. 14, n. 5, p. 413–428, 1978.

RODRIGUES, L. de O. et al. A hardware-in-the-loop simulation approach for testing control systems of small satellites. 2013.

ROMANO, M.; FRIEDMAN, D. A.; SHAY, T. J. Laboratory experimentation of autonomous spacecraft approach and docking to a collaborative target. *Journal of Spacecraft and Rockets*, [Easton, Pa., American Institute of Aeronautics and Astronautics], v. 44, n. 1, p. 164–173, 2007.

SABATINI, M.; FARNOCCHIA, M.; PALMERINI, G. B. Design and tests of a frictionless 2d platform for studying space navigation and control subsystems. In: *2012 IEEE Aerospace Conference*. [S.l.: s.n.], 2012. p. 1–12. ISSN 1095-323X.

SCHUBERT, H.; HOW, J. P. Space Construction: An Experimental Testbed to Develop Enabling

Technologies. In: *Telemanipulator and Telepresence Technologies IV*. Pittsburg, PA: [s.n.], 1997. Disponível em: <http://www.stanford.edu/group/arl/cgi-bin/drupal/sites/default/files/public/publications/SchubertH97.pdf>.

SCHWARTZ, J. L.; PECK, M. A.; HALL, C. D. Historical review of air-bearing spacecraft simulators. *Journal of Guidance, Control, and Dynamics*, 2003.

SHANGGUAN, L. et al. Implementation of a simulation platform for bcm with xpc target. In: IEEE. *Industrial Electronics and Applications (ICIEA), 2016 IEEE 11th Conference on*. [S.l.], 2016. p. 55–60.

SHORT, M.; ABUGCHEM, F. A microcontroller-based adaptive model predictive control platform for process control applications. *Electronics*, Multidisciplinary Digital Publishing Institute, v. 6, n. 4, p. 88, 2017.

SILANI, E.; LOVERA, M. Magnetic spacecraft attitude control: a survey and some new results. *Control Engineering Practice*, v. 13, n. 3, p. 357 – 371, 2005. ISSN 0967-0661. Aerospace IFAC 2002.

SILVA, M. de Athayde Costa e et al. A framework for development of satellite attitude control algorithms. *Journal of Control, Automation and Electrical Systems*, v. 25, n. 6, Dec 2014. ISSN 2195-3899.

SILVA, R. R. *Projeto de controladores para um sistema de direção elétrica utilizando a metodologia de projeto baseado em modelos*. Dissertação (Mestrado) — Dissertação de Mestrado do Curso de Pós-Graduação em Sistema Mecatrônicos, UNB, Brasília, DF, 2017.

SIMON, D. Model predictive control in flight control design. *Licentiate Thesis, Linköping University, Dept. of Electrical Engineering*, p. 17, 2014.

SLAFER, L. Use of hardware-in-the loop simulation for spacecraft mission preparation, planning and support. In: *Aerospace Design Conference*. [S.l.: s.n.], 1993. p. 1047.

SNIDER, R. E. *Attitude control of a satellite simulator using reaction wheels and a PID controller*. [S.l.], 2010.

SOUZA, L.; GONZALES, R. Application of the state-dependent riccati equation and kalman filter techniques to the design of a satellite control system. *Shock and vibration*, Hindawi Publishing Corporation, v. 19, n. 5, p. 939–946, 2012.

TAPSAWAT, W.; SANGPET, T.; KUNTANAPREEDA, S. Development of a hardware-in-loop attitude control simulator for a cubesat satellite. In: IOP PUBLISHING. *IOP Conference Series: Materials Science and Engineering*. [S.l.], 2018. v. 297, n. 1, p. 012010.

TAYYEBTAHER, M.; ESMAEILZADEH, S. M. Model predictive control of attitude maneuver of a geostationary flexible satellite based on genetic algorithm. *Advances in Space Research*, v. 60, n. 1, p. 57 – 64, 2017. ISSN 0273-1177.

TOMAN, J.; KERLÍN, T.; SINGULE, V. Application of the v-cycle development in the aerospace industry. *Engineering Mechanics*, Association for Engineering Mechanics, v. 18, n. 5-6, p. 297–306, 2011.

TOMIOKA, R. Convex optimization : Old tricks for new problems. Unpublished. 2012.

TRAGESSER, S.; AGNES, G.; FULTON, J. Simsat: a ground-based platform for demonstrating satellite attitude dynamics and control. *age*, v. 7, p. 1, 2002.

TULPULE, P. et al. Model based design (mbd) and hardware in the loop (hil) validation: Curriculum development. In: *2017 American Control Conference (ACC)*. [S.l.: s.n.], 2017. p. 5361–5366.

URE, N. K.; KAYA, Y. B.; INALHAN, G. The development of a software and hardware-in-the-loop test system for itu-psat ii nano satellite adcs. In: *2011 Aerospace Conference*. [S.l.: s.n.], 2011. p. 1–15. ISSN 1095-323X.

VALMORBIDA, A. Development and testing of model predictive control strategies for spacecraft formation flying. 2014.

VILATHGAMUWA, D. M.; YUE, X.; TSENG, K. J. Development and control of a 3-axis motion simulator for satellite adcs hardware-in-the-loop simulation. In: *30th Annual Conference of IEEE Industrial Electronics Society, 2004. IECON 2004*. [S.l.: s.n.], 2004. v. 1, p. 524–529 Vol. 1.

VIRGILI-LLOP, J. et al. Experimental evaluation of model predictive control and inverse dynamics control for spacecraft proximity and docking maneuvers. *CEAS Space Journal*, Springer, p. 1–13, 2016.

VOTEL, R.; SINCLAIR, D. Comparison of control moment gyros and reaction wheels for small earth-observing satellites. 2012.

WAHBA, M. et al. A ros-simulink real-time communication bridge using udp with a driver-in-the-loop application. In: AMERICAN SOCIETY OF MECHANICAL ENGINEERS. *ASME 2016 Dynamic Systems and Control Conference*. [S.l.], 2016. p. V002T23A002–V002T23A002.

WANG, L. *Model predictive control system design and implementation using MATLAB®*. [S.l.]: Springer Science & Business Media, 2009.

WERTZ, J. R. *Spacecraft attitude determination and control*. [S.l.]: Springer Science & Business Media, 1978. v. 73.

XIAO, B.; HU, Q.-L.; MA, G.-F. Adaptive l-two-gain controller for flexible spacecraft attitude tracking. *Control Theory & Applications*, Huanan Ligong Daxue, v. 28, n. 1, p. 101–107, 2011.

XING, Y. T.; LOW, K. S.; PHAM, M. D. Distributed model predictive control of satellite attitude using hybrid reaction wheels and magnetic actuators. In: *2012 IEEE Symposium on Industrial Electronics and Applications*. [S.l.: s.n.], 2012. p. 230–235.

YAO, H. et al. Implementation of three dofs small satellite ground simulation system. In: *54th AIAA Aerospace Sciences Meeting*. [S.l.: s.n.], 2016. p. 0697.