UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

RENATA RODRIGUES LIMA

CONTROLE DA FLEXÃO E EXTENSÃO DO COTOVELO VIA FES EM MALHA-FECHADA UTILIZANDO CONTROLADORES RST

FORTALEZA

2022

RENATA RODRIGUES LIMA

CONTROLE DA FLEXÃO E EXTENSÃO DO COTOVELO VIA FES EM MALHA-FECHADA UTILIZANDO CONTROLADORES RST

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Orientador: Prof. Dr. Fabricio Gonzalez Nogueira

FORTALEZA

2022

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

L71c Lima, Renata Rodrigues.
Controle da flexão e extensão do cotovelo via FES em malha-fechada utilizando controladores RST / Renata Rodrigues Lima. – 2022.
105 f. : il. color.

Trabalho de Conclusão de Curso (graduação) – Universidade Federal do Ceará, Centro de Tecnologia, Curso de Engenharia Elétrica, Fortaleza, 2022.
Orientação: Prof. Dr. Fabricio Gonzalez Nogueira.

1. Flexão e extensão do cotovelo. 2. Opensim. 3. Identificação de sistemas. 4. Controlador RST. I. Título.
CDD 621.3

RENATA RODRIGUES LIMA

CONTROLE DA FLEXÃO E EXTENSÃO DO COTOVELO VIA FES EM MALHA-FECHADA UTILIZANDO CONTROLADORES RST

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Engenharia Elétrica do Centro de Tecnologia da Universidade Federal do Ceará, como requisito parcial à obtenção do grau de bacharel em Engenharia Elétrica.

Aprovada em:

BANCA EXAMINADORA

Prof. Dr. Fabricio Gonzalez Nogueira (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. Bismark Claure Torrico
Universidade Federal do Ceará (UFC)

Prof. Dr. George André Pereira Thé
Universidade Federal do Ceará (UFC)

Me. Eugenio Peixoto Junior
Universidade de Fortaleza (UNIFOR)

À minha família, por todo o suporte que me deram durante essa jornada.

AGRADECIMENTOS

À minha família, especialmente à minha mãe Socorro, minha tia Karina, minha avó Maria e meu irmão Renan, por todo o suporte que me deram para que eu pudesse completar minha graduação. Agradecer também pelo apoio quando decidi trocar o curso de Engenharia de Produção pela Engenharia Elétrica quando estava no meio da graduação.

Gostaria de agradecer também aos grupos no qual fiz parte desde a minha entrada à universidade, que me moldaram como pessoa e profissional.

Primeiramente gostaria de agradecer à todo o grupo no qual trabalhei no PET Engenharia de Produção, primeira experiência em gestão e liderança que tive como pessoa.

Ao Centro de Empreendedorismo da UFC e os companheiros que lá fiz, que me abriram os olhos para o mundo do empreendedorismo e das startups.

À Elogroup, que deu minha primeira experiência profissional, dando a oportunidade de fazer parte do time da Casa Azul Ventures, aceleradora de startups, que me proporcinou um grande ensino tanto pessoal como profissional.

Ao professor Dr. Fabrício Nogueira, por me dar a oportunidade de ingressar no projeto de estudo de controle utilizando FES, que deu origem ao presente trabalho, e realizar minha orientação no mesmo. Aos também integrantes do projeto FES, Bruno Faustino, Aparecida Falcão e Eugenio Peixoto, pela contribuição e apoio.

Aos engenheiros eletricistas Ednardo Moreira Rodrigues e Alan Batista de Oliveira pela adequação do template utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da Associação Brasileira de Normas Técnicas (ABNT).

Eu sou o mestre do meu destino,

Eu sou o capitão da minha alma.

(William Ernest Henley)

RESUMO

Novos estudos que implementam a técnica de Estimulação Elétrica Funcional/*Functional Electrical Stimulation* (FES) em malha fechada estão surgindo, com o intuito de controlar e gerar movimentos desejados nos membros de pessoas acometidas por deficiências motoras. Contudo, ainda falta eficiência e robustez nos seus resultados. A maioria dos projetos utiliza funções de transferência para movimentos biomecânicos presentes na literatura para se projetar os parâmetros de controle. Levando em consideração que muitos fatores podem levar os modelos reais a se distanciarem dos modelos biomecânicos teóricos, o presente trabalho apresenta o estudo e utilização de uma técnica de identificação de modelos discretos a partir de dados experimentais, com o intuito de se projetar controladores *RST* para os movimentos de flexão e extensão do cotovelo. Por ser um estudo inicial, todos os experimentos foram feitos por meio do $software$ *Opensim*, plataforma em código aberto para sistemas musculoesqueléticos. Entradas *PRBS* foram projetadas para serem inseridas em um modelo que contém os músculos responsáveis pela flexão e extensão do cotovelo. Os dados de entrada e saída foram coletados para que um modelo discreto do sistema fosse identificado. Posteriormente, os parâmetros *RST* de dois controladores foram obtidos, um para o movimento de flexão e outro para o movimento de extensão. O funcionamento dos controladores foi avaliado operando de maneira individual e, posteriormente, de maneira paralela, constatando-se um desempenho satisfatório considerando os parâmetros desejados. Ao final são propostos novos estudos para que o controlador obtido na simulação possa se adaptar a testes reais.

Palavras-chave: Flexão e extensão do cotovelo. Opensim. Identificação de sistemas. Controlador RST.

ABSTRACT

New studies that implement the technique of Functional Electrical Stimulation (FES) in closed loop are emerging, in order to control and generate desired movements in the limbs of people with motor disabilities. However, it still lacks efficiency and robustness in its results. Most projects use transfer functions for biomechanical movements present in the literature to design the control parameters. Taking into account that many factors can lead real models to distance themselves from theoretical biomechanical models, the present work presents the study and use of a technique for identifying discrete models from experimental data, in order to design *RST* controllers for elbow flexion and extension movements. As an initial study, all experiments were performed using *software Opensim*, an open source platform for musculoskeletal systems. *PRBS* entries were designed to be inserted into a model that contains the muscles responsible for elbow flexion and extension. Input and output data were collected so that a discrete model of the system could be identified. Subsequently, the *RST* parameters of two controllers were obtained, one for the flexion movement and the other for the extension movement. The performance of the controllers was evaluated operating individually and, later, in parallel, verifying a satisfactory performance considering the desired parameters. At the end, new studies are proposed to adapt the controller obtained in the simulation to be used in tests with real limbs.

Keywords: Elbow flexion and extension. Opensim. Systems Identification. RST Controller.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

FES	Estimulação Elétrica Funcional/*Functional Electrical Stimulation*
CC	*Componente Contrátil*
CES	*Componente Elástico em Série*
CEP	*Componente Elástico em Paralelo*
API	Interface de Programação de Aplicações/*Applications Protocol Interface*
CAD	*Conversor Analógico Digital*
CDA	*Conversor Digital Analógico*
ZOH	Segurador de Ordem Zero/*Zero-Order Hold*
ARX	Auto-Regressivo Exógeno/*Auto-Regressive Exogenous*
PRBS	Sequência Binária Pseudoaleatória/*Pseudo-Random Binary Sequence*

LISTA DE SÍMBOLOS

$u(k)$	Entrada de controle no tempo discreto
$y(k)$	Saída do sistema de controle no tempo discreto
$u(t)$	Entrada de controle no tempo contínuo
$y(t)$	Saída do sistema de controle no tempo contínuo
Θ_O	Ângulo da articulação do ombro
Θ_C	Ângulo da articulação do cotovelo
f_S	Frequência de amostragem
T_S	Tempo de amostragem
$B(q^{-1})$	Polinômio do numerador da função de transferência do modelo ARX
n_b	Ordem do polinômio $B(q^{-1})$
$A(q^{-1})$	Polinômio do denominador da função de transferência do modelo ARX
n_a	Ordem do polinômio $A(q^{-1})$
$e(k)$	Erro, ou resíduo, entre modelo identificado e modelo real
φ	Matriz com as entradas de controle e saídas do sistema
Φ	Matriz com φ de diversas amostras
Θ	Matriz com os coeficientes de $B(q^{-1})$ e $A(q^{-1})$
E	Matriz com o resíduo de diversas amostras
Y	Matriz com a saída do sistema de controle de diversas amostras
R^2	Correlação múltipla entre saída real e saída estimada
$r_{u\varepsilon}$	Correlação cruzada entre entrada e resíduo
$r_{\varepsilon\varepsilon}$	Auto correlação entre resíduo
$R(q^{-1})$	Polinômio multiplicativo da saída da planta em um controlador RST
n_R	Ordem do polinômio $R(q^{-1})$
$S(q^{-1})$	Polinômio multiplicativo da entrada de controle em um controlador RST
n_S	Ordem do polinômio $S(q^{-1})$
$T(q^{-1})$	Polinômio multiplicativo da referência em um controlador RST

$P(q^{-1})$	Polinômio caracterísitico de um controlador RST
ξ	Fator de amortecimento de uma função de transferência de segunda ordem
ω_n	Frequência natural de uma função de transferência de segunda ordem
M	Matriz com os coeficientes adaptados dos polinômios $B(q^{-1})$ e $A(q^{-1})$
p	Matriz com os coeficientes desejados do polinômio $P(q^{-1})$
x	Matriz com os coeficientes adaptados dos polinômios $S(q^{-1})$ e $R(q^{-1})$

SUMÁRIO

1 INTRODUÇÃO

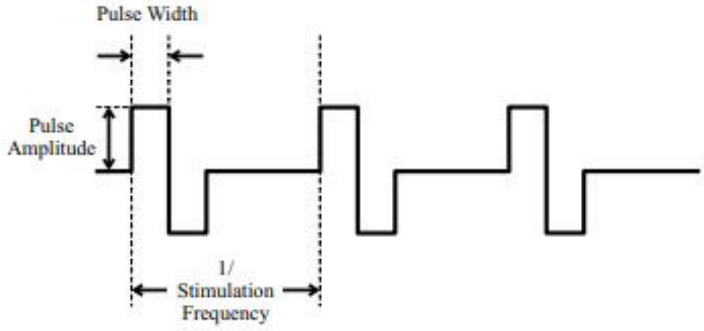
Em 2000, segundo o censo IBGE, o Brasil possuía 955.287 deficientes físicos. Já no censo de 2011 o número de pessoas que apresentava deficiência motora foi um total de 13.273.969. Nos Estados Unidos, 11 mil novos casos de lesões medulares são relatados a cada ano, sendo 52% relacionados a pessoas paraplégicas (SANCHES, 2013).

Após lesões medulares os músculos atrofiam rapidamente, e uma das consequências é a atividade reduzida do coração e do pulmão, diminuindo a qualidade de vida dos acometidos. A Estimulação Elétrica Funcional/*Functional Electrical Stimulation* (*FES*) é uma técnica que ativa eletricamente os músculos em uma sequência coordenada, que é utilizada durante anos para a restauração da atividade muscular de pacientes paraplégicos (KOZAN, 2012).

No geral os equipamentos FES trabalham em malha aberta, não promovendo uma associação adequada para a eliminação de erros entre o movimento desejado e o movimento obtido (PAZ *et al.*, 2020). Por conta dessa limitação no uso do FES muitos modelos de controle em malha fechada utilizando a técnica estão surgindo na literatura, contudo, a maioria desses estudos não chegaram a atingir a eficiência e a robustez suficientes para serem comercializados e utilizados em usuários finais (Bó *et al.*, 2016).

Nos estudos via FES experimentos iniciais são realizados para entender quais níveis de estímulos podem levar os músculos a apresentarem máxima contração ou mesmo desconforto e dor, estabelecendo os níveis limites que são usados nos projetos. A ativação é a variável de controle de um músculo, via FES ela se apresenta como uma função da carga trasmitida ao músculo, que depende da amplitude de corrente, da largura de pulso e da frequência do estímulo enviado. Esses parâmetros podem ser vistos na Figura 1.

Figura 1 – Parâmetros da Estimulação Elétrica Funcional



Fonte: (CHIN; MILOS, 2020).

Existem estudos que realizam o controle do músculo a partir da variação da amplitude

(PAZ *et al.*, 2020) e outros a partir da variação da largura de pulso (Bó; MOURA, 2015). Para não realizar experimentos iniciais em pessoas, o presente trabalho estabelece o projeto de um controlador via simulação em software de sistemas neuromusculares.

Para o projeto de controladores é necessário entender a planta a ser controlada, no caso da presente proposta de estudo, é necessário ter um modelo matemático que descreva a atividade neuromuscular. Para a obtenção desse modelo podem ser feitas duas técnicas: a utilização de leis físicas que regem o comportamento muscular ou a utilização de dados experimentais do músculo a ser trabalhado.

Levando em consideração que o movimento muscular é uma ação muito complexa, com muitas variáveis que influenciam em sua resposta, uma modelagem baseado em leis físicas do mesmo pode gerar modelos muito distantes da realidade. Por isso o estudo propõe a identificação de um modelo para a atividade neuromuscular com base na experimentação.

1.1 Objetivos

De modo geral, serão realizados experimentos em um *software* de simulação com o objetivo de se identificar modelos discretos para a ação dos músculos de flexão e extensão do cotovelo. De posse desses modelos, técnicas de determinação dos parâmetros de um controlador discreto serão utilizadas para o projeto de dois controladores, uma para a flexão e outro para a extensão do cotovelo. Posteriormente será feita uma avaliação da eficiência do uso dos controladores de forma individual e em conjunto.

1.2 Organização do Trabalho

O presente trabalho está dividido no seguinte formato:

- O capítulo 1 traz a introdução e os objetivos do estudo.
- O capítulo 2 traz as referências para a teoria que serão utilizadas.
- O capítulo 3 apresenta o projeto prático e o resultado do mesmo.
- O capítulo 4 apresenta as conclusões feitas com os resultados obtidos e ideias para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Para o projeto de um controlador que atuará na simulação de movimentos biomecânicos é necessário a compreensão da fisiologia dos movimentos desejados, o conhecimento da ferramenta de *software* usada para a simulação, e o estudo de técnicas de controle adequadas para o mesmo.

2.1 Fisiologia do movimento de flexão e extensão do cotovelo

Para entender como ocorre a dinâmica dos movimentos de flexão e extensão do braço é necessário entender dois grandes sistemas que trabalham para a obtenção desses movimentos: o sistema muscular e o sistema nervoso.

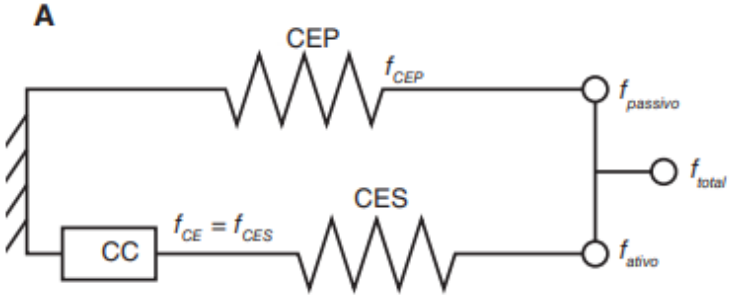
Os músculos são fibras que desempenham diversas funções dentro do corpo humano, na qual conseguimos destacar duas: contribuição para a produção de movimento do esqueleto e manutenção da postura e posicionamento do corpo. Já o sistema nervoso é o responsável pela identificação e pelo estímulo dos músculos responsáveis por determinado movimento (HAMILL *et al.*, 2015).

Os dois sistemas atuam juntos no processo de contração muscular, quando a partir de um impulso elétrico enviado pelo sistema nervoso, reações químicas acontecem na unidade muscular forçando a contração da mesma. Essa contração muscular é a responsável pela transferência de tensão dos músculos e tendões para o esqueleto, e pela geração de movimento e força.

Uma unidade motora é conhecida como um grupo de fibras musculares que são inervadas por um mesmo motoneurônio (neurônios que transmitem os impulsos elétricos até os músculos). Quanto maior a quantidade de fibras na unidade motora, maior seu potencial de ação. Existem modelos mecânicos que simulam a ação muscular, como o modelo de Hill, presente na Figura 2.

No modelo de Hill o *Componente Contrátil* (*CC*), é a estrutura que converte o estímulo vindo do sistema nervoso em força, refletindo a contração muscular, e o *Componente Elástico em Série* (*CES*) representa as forças elásticas que estão em série com essa contração, que oferecem uma resistência a esse movimento. O músculo também oferece resistência a forças externas aplicadas sobre ele, exercendo força passiva, por isso é necessário o *Componente Elástico em Paralelo* (*CEP*). Modelos como o de Hill são amplamente usados em modelos

Figura 2 – Modelo de Hill

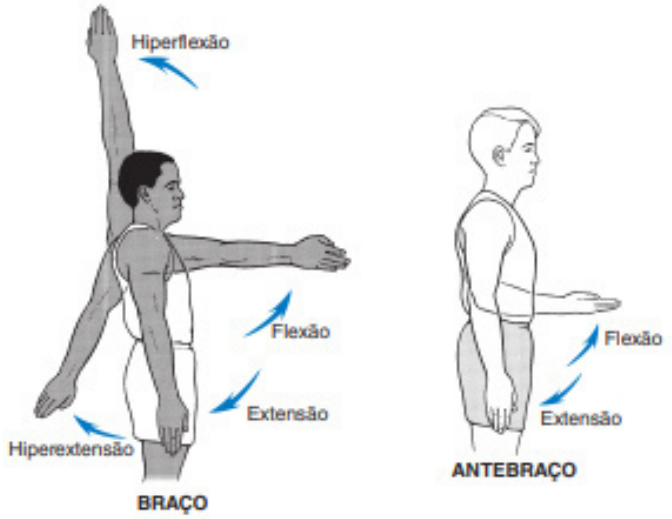


Fonte: (HAMILL *et al.*, 2015).

computadorizados que buscam simular as atividades musculares.

O movimento de flexão é considerado o movimento de curvatura que diminui o ângulo entre dois membros adjacentes. Já a extensão é o movimento antagônico, ela aumenta o ângulo entre dois membros. A Figura 3 mostra uma exemplificação dos movimentos de flexão e extensão do braço e antebraço, também conhecidos como flexão e extensão do ombro e do cotovelo, respectivamente.

Figura 3 – Movimentos de flexão e extensão do braço e antebraço.

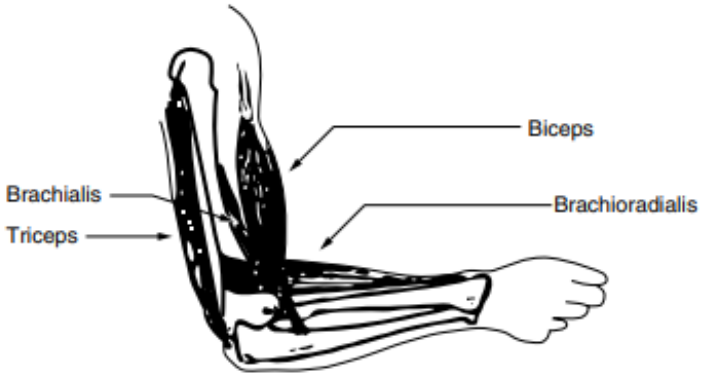


Fonte: (HAMILL *et al.*, 2015).

Nesse presente estudo vamos estudar o movimento de flexão e extensão do antebraço a partir do estímulo dos músculos responsáveis por eles. A Figura 4 mostra a localização dos músculos responsáveis pela flexão e extensão do antebraço.

Os músculos responsáveis pela flexão do antebraço são o músculo braquial, o braquiorradial e o biceps braquial, que é dividido em biceps cabeça longa e biceps cabeça curta. E o músculo responsável pela extensão é o triceps braquial, que é divido em triceps cabeça longa,

Figura 4 – Músculos responsáveis pela flexão e extensão do antebraço.

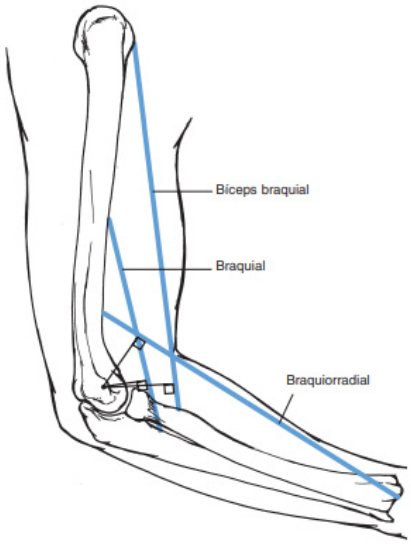


Fonte: (FREIVALDS, 2004).

triceps cabeça medial e triceps cabeça lateral.

Para entender um pouco como o movimento é feito a partir da contração muscular podemos observar a Figura 5.

Figura 5 – Esquemático da flexão do antebraço.



Fonte: (HAMILL *et al.*, 2015).

A contração do biceps, do braquial e do braquiorradial gera um momento no antebraço de sentido anti-horário, com eixo no cotovelo, fazendo com que o ângulo deste com o braço diminua. Podemos ver também que o músculo braquiorradial apresenta o maior braço do momento, tendo maior vantagem mecânica, seguido do biceps e do braquial. O movimento de extensão segue uma lógica semelhante, com a contração do triceps gerando um momento no antebraço que o faz girar no sentido horário, aumentando o ângulo entre o braço e antebraço.

O movimento de flexão pode ser adquirido com a contração dos músculos biceps,

braquial e braquiorradial, já a extensão pode ser adquirida pelo relaxamento dos mesmos. O mesmo pode ser dito para os músculos que compõe o triceps braquial, com o movimento de extensão sendo adquirido com a sua contração, e o movimento de flexão com seu relaxamento.

Com o intuito de facilitar a comunicação no presente trabalho chamaremos de músculos de flexão aqueles que geram o movimento de flexão a partir de sua contração (biceps, braquial e braquiorradial) e músculos de extensão aqueles que geram o movimento de extensão a partir de sua contração (triceps).

2.2 Opensim: Software de simulação biomecânica

Diversos *softwares* podem ser usados para simulações biomecânicas, levando em consideração o modelo de *Hill* ou semelhantes, como o *Visual 3-D*, *AnyBody* e *SIMM*. Contudo, esses sistemas apresentam limitações como a não permissão de acesso direto à Interface de Programação de Aplicações/*Applications Protocol Interface* (*API*) dos mesmos. O acesso a *API* permite que outros *softwares* possam ser usados em conjunto para se atingir o objetivo de estudo (PINHEIRO, 2019).

O *Opensim* é uma plataforma de código aberto que permite a modelagem, simulação e análise de sistemas neuro-músculo-esqueléticos (DELP *et al.*, 2007). Nele é possível a criação de modelos, análise dos mesmos, simulações de movimento entre diversas outras funcionalidades. Por essas funcionalidades aliado ao livre acesso de sua API, o *Opensim* se torna a ferramenta mais indicada para estudos de modelos musculares.

Usando somente o *Opensim* é possível realizar simulações de controle a partir de estímulos enviados aos músculos desejados. Esse estímulo é chamado de ativação, e tem seus valores variando de 0 a 1. Quando um sinal de valor zero é enviado ao músculo esse gera uma força igual a zero, ou seja, não sofre contração. A medida que o valor de ativação aumenta, a força e a contração do músculo aumentam, até se chegar ao valor 1, quando o músculo gera a força máxima que está a seu alcance. Podemos ver então que a ativação do músculo é o sinal de controle do mesmo.

Contudo, o uso somente do *Opensim* não permite a realização de controle em malha fechada, somente em malha aberta. Para se fazer o controle em malha fechada é necessário a utilização de outros *softwares* em conjunto com o *Opensim*, através de sua API. Nesse presente estudo, o software utilizado para a fusão com o *Opensim* foi o *software MATLAB*.

Para muitos projetos é mais eficiente utilizar modelos musculoesqueléticos já desen-

volvidos por outros estudos. O *Opensim* disponibiliza uma plataforma online que incentiva o compartilhamento dos estudos que são feitos com o mesmo.

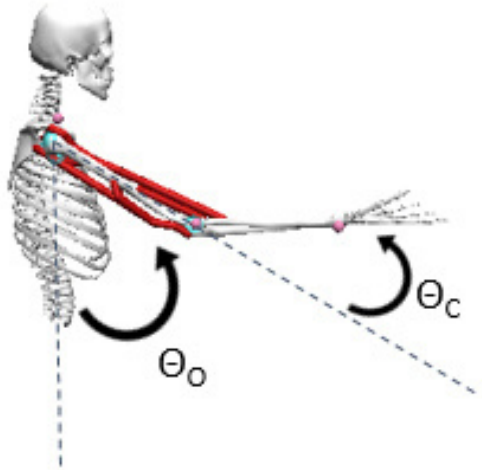
Junto à instalação do *software Opensim* também são instalados modelos já existem que foram elaborados pela equipe de desenvolvimento da plataforma. Nesse estudo iremos utilizar um desses modelos.

2.2.1 Modelo musculoesquelético Arm26

O modelo *Arm*26, presente junto a instalação do *Opensim*, é um modelo musculoesquelético da região superior do corpo humano, que conta com 6 músculos do braço direito, onde 3 deles são o biceps cabeça longa, biceps cabeça curta e braquial, responsáveis pela flexão do cotovelo, e os outros 3 são o triceps cabeça longa, triceps cabeça medial e triceps cabeça lateral, responsáveis pela extensão. O modelo apresenta ainda dois graus de liberdade, o movimento de flexão e extensão do ombro e o movimento de flexão e extensão do cotovelo.

A Figura 6 mostra o modelo *Arm*26 com a referência para seus dois graus de liberdade, onde Θ_O é o ângulo do ombro e Θ_C o ângulo do cotovelo.

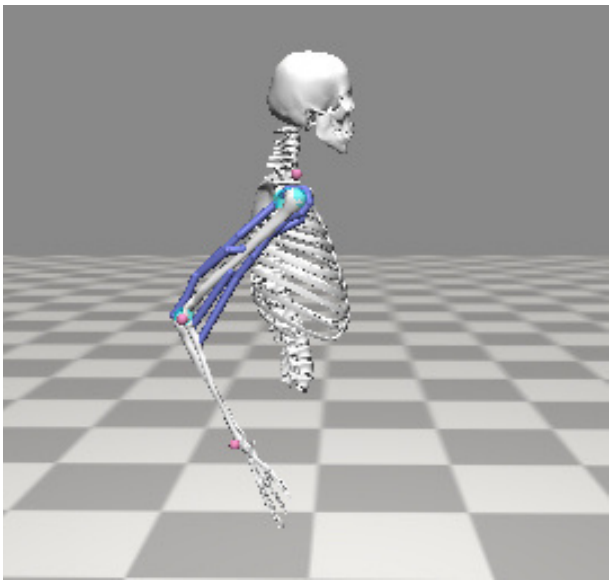
Figura 6 – Referência dos ângulos do ombro e cotovelo no modelo *Arm*26



Fonte: Próprio Autor.

No presente trabalho faremos o controle do ângulo do cotovelo, contudo, o ângulo do ombro influência para determinar quais músculos irão atuar no movimento. Para evidenciar isso podemos ver a Figura 7.

Figura 7 – Posição sem estímulos com ângulo do ombro em -60°



Fonte: Próprio Autor.

Nessa figura foi estabelecido que o ângulo do ombro deveria permanecer fixo em -60°. Depois foi aplicada uma simulação onde nenhum músculo foi estimulado, para saber em qual ângulo do cotovelo o braço iria se estabilizar. Como visto na imagem, sem nenhuma força e com o ombro em -60°, o braço se estabilizou com uma angulação de cotovelo igual à 68°.

Analisando essa posição do ombro, para termos um ângulo de cotovelo maior do que 68° o movimento de flexão deve ser feito. Já para termos um ângulo menor que 68° o movimento de extensão deve ser realizado. Para posições diferentes do ombro essa divisão dos graus para os movimentos de flexão e extensão também será diferente. Com isso vemos que mesmo o ângulo do ombro não sendo o objetivo de controle, seu parâmetro influência em quais músculos deverão atuar para que a referência seja seguida.

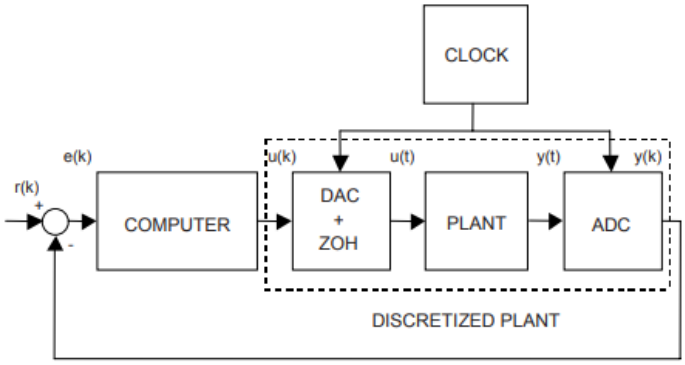
É importante destacar também que o modelo *Arm*26 não apresenta o músculo braquiorradial, músculo este que também participa do movimento de flexão. Temos então que todo o estudo da flexão será feito sem a presença desse músculo.

2.3 Teoria do controle discreto

Em muitas aplicações práticas de controle é necessário que os dados do projeto sejam trabalhados por computadores, que funcionam de maneira discreta. A planta estudada, por ser uma simulação, apresenta natureza discreta, e faremos o seu controle utilizando as teorias para o controle digital. Uma metodologia para a aplicação do controle discreto está presente na

Figura 8.

Figura 8 – Controle discreto em planta contínua



Fonte: (LANDAU; ZITO, 2006).

Na Figura 8 a sequência *CDA-Planta-CAD* pode ser chamada de modelo discreto da planta e estabelece a relação entre a entrada de controle *u(k)* e a saída do modelo *y(k)* (LANDAU; ZITO, 2006).

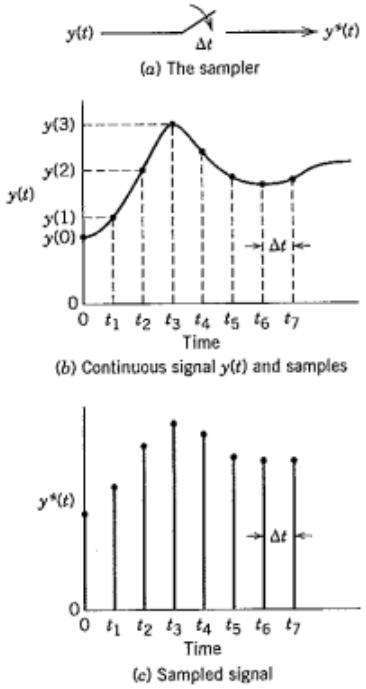
Também é visto a necessidade do bloco *CLOCK* que irá determinar o tempo de amostragem dos sinais, tempo que estabelece o intervalo em que será feita a coleta dos valores do sinal contínuo que serão transformados em sinal discreto. A Figura 9 mostra o funcionamento do *Conversor Analógico Digital* (CAD) que converte a saída analógica em um sinal aceito pelo sistema computacional.

Para que o sinal de saída do sistema computacional seja aplicado à planta contínua é necessário fazer o caminho inverso. Se utiliza primeiro um *Conversor Digital Analógico* (CDA) e um Segurador de Ordem Zero/*Zero-Order Hold* (*ZOH*), que transforma o sinal de uma amostra em um sinal fixo e contínuo durante o tempo de amostragem. A Figura 10 mostra como o *CDA* e o *ZOH* atuam juntos para a geração de um sinal contínuo a partir de um discreto.

A planta estudada, por ser uma simulação, apresenta natureza discreta, mas ainda se faz necessário o estudo do tempo de amostragem, pois a amostragem de uma simulação ocorre de maneira mais rápida do que a amostragem necessária em um experimento prático. Portanto usaremos todos os parâmetros teóricos para projetar um controlador discreto, mas não será necessária a presença de um *CAD*, *CDA* e *ZOH*.

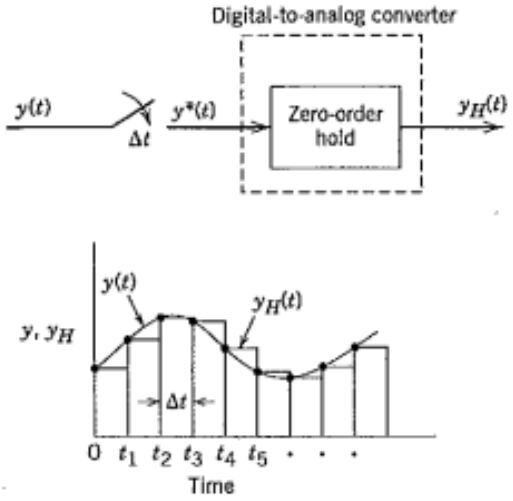
Primeiramente serão analisadas técnicas para se determinar o tempo de amostragem ideal do sistema estudado, juntamente com métodos para a identificação de um modelo discreto para a planta, por fim, será feito o estudo da determinação dos parâmetros necessários para um controlador digital.

Figura 9 – Ação de um *CAD*



Fonte: (SEBORG *et al.*, 2004).

Figura 10 – Ação de um *CDA*



Fonte: (SEBORG *et al.*, 2004).

2.3.1 Tempo de amostragem

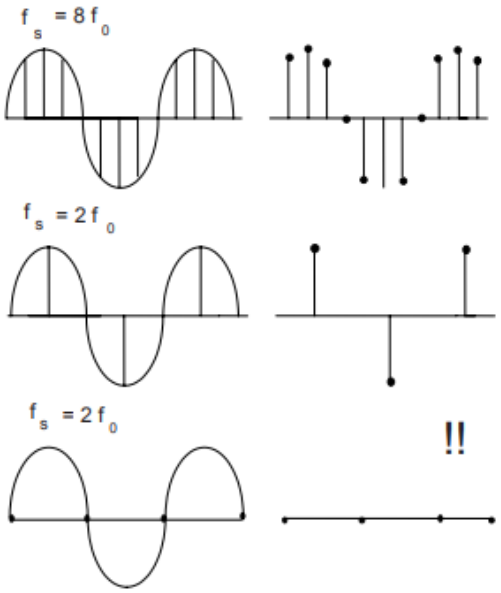
Durante uma conversão analógico digital pouca informação do sinal analógico é perdida se as amostras coletadas se encontrarem muito próximas entre si, mas caso elas estejam mais distantes, sinais distorcidos do original podem ser obtidos (ASTROM; WITTENMARK, 1997).

Em contrapartida, diminuir o tempo de amostragem acarreta em um sistema de controle mais caro. Uma frequência de amostragem menor também pode trazer o benefício

de disponibilizar mais tempo para os cálculos de controle do sistema e uma maior eficiência do mesmo (FRANKLIN *et al.*, 1998). Vê-se então a importância de se escolher um tempo de amostragem que não seja tão alto, capaz de perder informações importantes do sinal, e não tão baixo, comprometendo a eficiência do sistema de controle.

Para (SEBORG *et al.*, 2004, p. 445, tradução nossa): "A seleção do período de amostragem permanece mais como uma arte do que uma ciência". Contudo, diversos autores apresentam métodos para se determinar o tempo de amostragem de maneira mais técnica. Para exemplificar um teorema de amostragem importante podemos observar a Figura 11.

Figura 11 – Amostragem de um sinal contínuo a partir de diferentes frequências



Fonte: (LANDAU; ZITO, 2006).

Pode-se ver pela Figura 11 que uma amostragem com frequência 8 vezes maior que a frequência do sinal analógico consegue gerar um sinal digital sem grandes perdas de informações. Quando a frequência de amostragem é o dobro da frequência do sinal analógico e o sinal é coletado em instantes não múltiplos de π, o sinal digital consegue manter a mesma frequência, mas apresenta perda de informação significativa. Já quando o sinal é coletado em instantes múltiplos de π, vemos que o sinal fica equivalente a zero, perdendo toda sua informação. Se uma frequência de amostragem ainda menor for aplicada, o sinal discreto obtido terá uma frequência diferente do sinal original (LANDAU; ZITO, 2006). Isso consegue exemplificar de maneira simples o *Teorema de Nyquist*, que estabelece que

$$f_S > 2f_{max}, \tag{2.1}$$

Onde

f_S: frequência de amostragem,

f_{max}: frequência máxima a ser transmitida para o sinal discreto.

O *Teorema de Nyquist* estabelece um limite inferior para a frequência de amostragem do sistema, na prática essa frequência deve apresentar valores um pouco maiores que $2f_{max}$, dependendo do sinal analógico. Segundo (LANDAU; ZITO, 2006) a seguinte regra pode ser aplicada:

$$f_S = (6 \quad a \quad 25) f_B^{CL}, \tag{2.2}$$

Onde

f_S: frequência de amostragem,

f_B^{CL}: largura de banda do sistema em malha fechada.

Trabalhando a Equação 2.2 para um sistema de primeira ordem em malha fechada temos

$$\frac{T_0}{4} < T_S < T_0, \tag{2.3}$$

Onde

T_S: tempo de amostragem,

T_0: tempo de subida da saída aplicando-se uma entrada degrau.

Em teoria, a Equação 2.2 irá gerar de 2 à 9 amostras dentro do tempo de subida de uma resposta à uma entrada em degrau (LANDAU; ZITO, 2006).

Na prática muitos projetos adotam a escolha de ter pelo menos 10 amostras dentro do tempo de subida de um sistema estimulado a entrada em degrau, valor esse que se assemelha a Equação 2.2. Essa metodologia prática de 10 amostras será a adotada no presente trabalho.

2.3.2 Identificação de sistemas

Um sistema dinâmico pode ser identificado por meio de uma abordagem experimental que segue os seguintes passos (LANDOU *et al.*, 2011):

1. Aquisição de dados de entrada e saída durante um experimento.
2. Determinação da estrutura do modelo que vai descrever o sistema estudado.
3. Estimação dos parâmetros que compõe o modelo estabelecido.
4. Validação do modelo identificado.

Como visto anteriormente, nesse estudo vamos utilizar técnicas de controle discreto, com isso vamos utilizar modelos, técnicas de estimação e validação que se adequem ao mesmo.

2.3.2.1 Modelo ARX para plantas em tempo discreto

O modelo de um sistema é uma descrição das propriedades do mesmo, que não necessariamente deve ser uma descrição exata, mas sim uma descrição que atenda aos propósitos de estudo (LJUNG, 1987).

Nesse presente trabalho iremos usar o modelo conhecido como Auto-Regressivo Exógeno/*Auto-Regressive Exogenous* (*ARX*), modelo muito utilizado para descrever sistemas discretos lineares e invariantes no tempo.

Para entender sua fórmula vamos primeiro ver a estrutura de um modelo em tempo discreto da resposta impulso de um sistema. Segundo (KEESMAN, 2011) a soma de convolução pode ser escrita como

$$y(t) = G(q)u(t), \tag{2.4}$$

Onde $G(q) = \sum_{k=0}^{\infty} g(k)^{-k}$, é um polinômio infinito que pode ser considerado uma função de transferência de um sistema discreto linear e invariante no tempo.

Devido a diferença entre as saídas reais medidas e as saídas estimadas dos modelos, para representar modelos identificados é necessário adicionar nas equações uma variável de erro. Esse termo em muitas literaturas é conhecido como ruído do modelo. Utilizando a Equação 2.4, substituindo $G(q)$ por $B(q)$ e adicionado o termo referente ao ruído temos:

$$y(t) = b_1u(t-1) + b_2u(t-2) + \cdots + e(t) = B(q)u(t) + e(t), \tag{2.5}$$

E

$$B(q) = \sum_{k=1}^{\infty} b_k q^{-k} = b_1q^{-1} + b_2q^{-2} + \cdots, \tag{2.6}$$

Onde

$e(t)$:termo referente ao erro do modelo.

Na Equação 2.6 o coeficiente k tem seu valor dado a partir de 1, pois em um sistema real a saída $y(t)$ não sofre um efeito direto da entrada $u(t)$, somente de entradas anteriores a essa. Podemos ainda ver que a equação apresenta um polinômio infinito, na prática, contudo, é suficiente expressar esse polinômio em valores finitos que chamaremos de n_b, transformando a

equação em

$$B(q) = \sum_{k=1}^{n_b} b_k q^{-k} = b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}. \tag{2.7}$$

Ainda segundo (KEESMAN, 2011), podemos assumir que a saída $y(t)$ tem relação com saídas anteriores, adicionando novos termos as equações temos

$$y(t) + a_1 y(t-1) + \cdots + a_{n_a} y(t-n_a) = b_1 u(t-1) + \cdots + b_{n_b} u(t-n_b) + e(t). \tag{2.8}$$

Reescrevendo para um formato que possa se extrair a função de transferência temos

$$A(q)y(t) = B(q)u(t) + e(t), \tag{2.9}$$

E

$$y(t) = \frac{B(q)}{A(q)} u(t) + \frac{1}{A(q)} e(t), \tag{2.10}$$

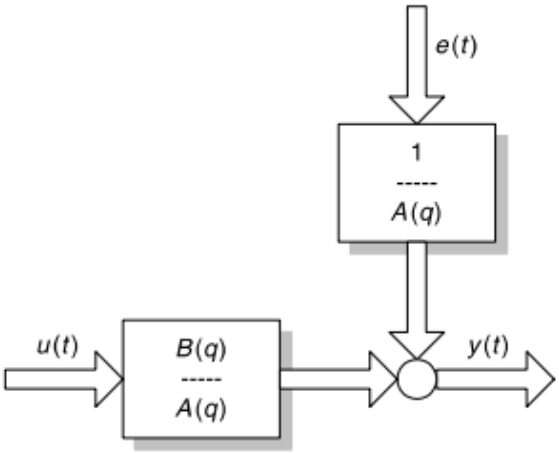
Onde

$A(q) = \sum_{k=0}^{n_a} a_k q^{-k} = a_0 + a_1 q^{-1} + a_2 q^{-2} + ... + a_{n_a} q^{-n_a}$,

$a_0 = 1$.

A estrutura presente na Equação 2.10 é conhecida como modelo ARX, e será o modelo adotado no presente trabalho para representar as plantas em tempo discreto do estudo. Um diagrama de blocos do modelo ARX pode ser visto na Figura 12.

Figura 12 – Diagrama de blocos do modelo *ARX*



Fonte: (KEESMAN, 2011).

Para termos então o modelo do sistema estudado em tempo discreto devemos estimular o sistema e coletar seus dados de entrada e saída, para, com esses valores, projetar os parâmetros A e B que minimizam o erro $e(t)$ do modelo.

2.3.2.2 *Teoria dos mínimos quadrados*

Depois de estabelecido qual modelo será usado para representar o sistema de controle em tempo discreto, é preciso estabelecer qual método será usado para estimar os parâmetros do modelo. No caso do modelo ARX, serão estimados os parâmetros A e B.

O método de estimação escolhido foi o método dos mínimos quadrados. Segundo (ASTROM; WITTENMARK, 1994) o método dos mínimos quadrados pode ser aplicado em uma grande variedade de problemas que apresentem a seguinte estrutura:

$$y(i) = \varphi_1(i)\theta_1^0 + \varphi_2(i)\theta_2^0 + \cdots + \varphi_n(i)\theta_n^0 = \varphi^T(i)\theta^0, \tag{2.11}$$

Onde

$\varphi^T(i) = \begin{bmatrix} \varphi_1(i) & \varphi_2(i) & \cdots & \varphi_n(i) \end{bmatrix}$,

$\theta^0 = \begin{bmatrix} \theta_1^0 & \theta_2^0 & \cdots & \theta_n^0 \end{bmatrix}^T$.

Podemos reescrever a Equação 2.8 com suas variáveis em tempo discreto e isolar a saída, obtendo o seguinte formato:

$$y(k) = -a_1 y(k-1) \cdots - a_{n_a} y(k-n_a) + b_1 u(k-1) + \cdots + b_{n_b} u(k-n_b) + e(k). \tag{2.12}$$

Vemos que a Equação 2.12 se enquadra no formato da Equação 2.11, onde

$$y(k) = \varphi^T(k)\theta^0 + e(k), \tag{2.13}$$

E

$\varphi^T(k) = \begin{bmatrix} -y(k-1) & \cdots & -y(k-n_a) & u(k-1) & \cdots & u(k-n_n) \end{bmatrix}$,

$\theta^0 = \begin{bmatrix} a_1 & \cdots & a_{n_a} & b_1 & \cdots & b_{n_b} \end{bmatrix}^T$.

A variável k denota a amostra em tempo discreto. Durante um experimento, N amostras serão coletadas, transformando a Equação 2.13 em

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} \varphi^T(0) \\ \varphi^T(1) \\ \vdots \\ \varphi^T(N-1) \end{bmatrix} \theta^0 + \begin{bmatrix} e(0) \\ e(1) \\ \vdots \\ e(N-1) \end{bmatrix},$$

Que pode ser simplificada para

$Y = \Phi\Theta + E$.

Temos então que o erro do modelo pode ser descrito como

$$E = Y - \Phi\Theta. \tag{2.14}$$

O método dos mínimos quadrados estabelece que o erro quadrático do modelo deve ser mínimo. O erro quadrático apresenta a forma de

$$V = \frac{1}{2}\Sigma_{k=0}^{N} e^2(k) = \frac{1}{2}||E||^2,$$

Onde

V: erro quadrático.

Usando a Equação 2.14 temos

$$V = \frac{1}{2}||Y - \Phi\Theta||^2,$$
$$2V = (Y - \Phi\Theta)^T(Y - \Phi\Theta),$$

$$2V = Y^TY - Y^T\Phi\Theta - \Theta^T\Phi^TY + \Theta^T\Phi^T\Theta\Phi. \tag{2.15}$$

Segundo (HSIA, 1977) o mínimo valor para o erro quadrático pode ser encontrado diferenciando V em relação à Θ e igualando a zero. Fazendo isso temos

$$\frac{\partial V}{\partial \Theta} = -2\Phi^TY + 2\Phi^T\Phi\Theta,$$
$$0 = -2\Phi^TY + 2\Phi^T\Phi\Theta,$$

$$\Theta = [\Phi^T\Phi]^{-1}\Phi^TY. \tag{2.16}$$

Temos então que para descobrir os parâmetros A e B de um modelo ARX temos que realizar experimentos na planta estudada para que tenhamos os dados de entrada e saída do sistema, e posteriormente deve ser estabelecido o grau das variáveis n_a e n_b. De posse desses dados podemos montar as matrizes Φ e Y, e posteriormente utilizar a Equação 2.16 para descobrir os parâmetros.

É importante salientar que, observando a Equação 2.16, a matriz $\Phi^T\Phi$ deve ser invertível, portanto, seus valores não devem apresentar repetições. Para que isso aconteça, durante o experimento, deve-se excitar o sistema de controle em valores diferente para que este não atinja o regime permanente.

2.3.2.3 Validação do modelo identificado

O processo de validação serve para determinar se o modelo identificado é apropriado para o estudo. Uma análise visual pode ser feita ao se plotar o gráfico da resposta real do sistema com a resposta do modelo identificado quando se estabelece entre eles uma mesma entrada. Contudo, uma análise visual se torna um critério subjetivo, e devemos ter parâmetros objetivos

para determinar a confiabilidade de um modelo. Um desses parâmetros é a *Correlação Múltipla* (R^2), expressa pela seguinte equação:

$$R^2 = 1 - \frac{\sum_{k=1}^{N}[y(k) - \hat{y}(k)]^2}{\sum_{k=1}^{N}[y(k) - \bar{y}]^2}, \tag{2.17}$$

Onde

$y(K)$: saída real do sistema,

$\hat{y}(k)$: saída estimada pelo modelo identificado,

$\bar{y}$: valor médio da saída real.

Uma correlação múltipla igual a 1 indica um sistema com adequação exata a planta identificada. Valores entre 0,9 e 1 podem ser considerados suficientes para se constatar um modelo adequado para estudos práticos (COELHO; COELHO, 2004).

Podemos fazer também uma análise dos erros do sistema. Vimos anteriormente sobre o erro que os modelos que descrevem um sistema apresentam, e como devemos identifica-los de maneira a minimizá-los. Os erros em muitas literaturas são referidos como resíduos dos modelos. Se o resíduo do modelo for considerado um ruído branco, com valores aleatórios entre si, significa que o modelo apresenta a melhor previsão dos resultados do sistema, pois o resíduo do mesmo não tem relação com sua entrada (LANDOU *et al.*, 2011).

Existem métodos para verificar se um resíduo apresenta valores aleatórios e independentes da saída, sendo assim considerado um resíduo branco. Um desses métodos é avaliar a auto correlação do resíduo e a correlação cruzada entre resíduo e entrada do sistema.

Segundo (KEESMAN, 2011), a correlação cruzada entre entrada e resíduo é dado por

$$r_{u\varepsilon} = \frac{1}{N-l}\sum_{i=1}^{N-1} u(i)\varepsilon(i+l), \tag{2.18}$$

Onde

N: quantidade de amostras,

ε: resíduo do modelo.

Já a auto correlação dos resíduos do modelo é dada por

$$r_{\varepsilon\varepsilon} = \frac{1}{N-l}\sum_{i=1}^{N-1} \varepsilon(i)\varepsilon(i+l). \tag{2.19}$$

Se os resíduos forem perfeitamente brancos e a quantidade de amostras for muito grande, tendendo ao infinito, as correlações ditas anteriormente terão seus valores em zero,

contudo, em um sistema real, essas condições não acontecem (LANDOU *et al.*, 2011). Modelos identificados tendem a apresentar ordens baixas, com o intuito de deixar o sistema menos complexo, sistemas reais também apresentam não-linearidades e a quantidade de amostras em um experimento é sempre finita. Esses fatores contribuem para que em modelos de sistemas reais o resíduo nunca se apresente como branco.

Sabendo que as correlações nunca serão zero, devemos apresentar algum critério que avalie qual valor de correlação é aceitável para se considerar que o resíduo se aproxima de um resíduo branco, e, com isso, o erro não apresenta relação com a entrada do sistema. (LANDOU *et al.*, 2011) apresenta o seguinte critério para análise do sistema

$$r_{u\varepsilon}, r_{\varepsilon\varepsilon} \leq \frac{2,17}{\sqrt{N}}, \tag{2.20}$$

Quando

$l \geq 1$.

Usaremos as Equações 2.17 e 2.20 como parâmetro para analisar a assertividade dos modelos identificados nesse presente trabalho.

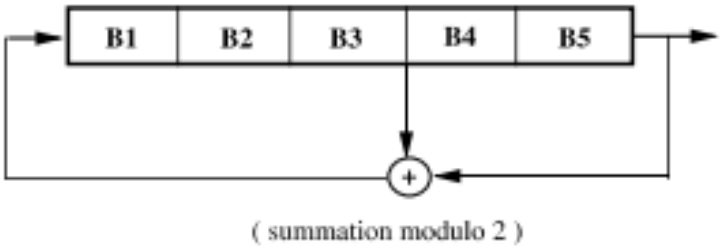
2.3.2.4 Sinal de entrada para a identificação do sistema

Por fim, para iniciarmos a identificação de um modelo, temos que pensar na entrada que excitará o sistema durante a experimentação. Para uma melhor estimação dos parâmetros o sinal de entrada não poderá ser qualquer um, como uma entrada em degrau. É necessário um sinal rico, que excitará o sistema de maneira persistente.

Segundo (HSIA, 1977), o ruído branco se apresenta como um desejável sinal de entrada para a identificação de um sistema, pois, sob sua atuação, a matriz $\Phi^T\Phi$ da Equação 2.16 é diagonal, tornando mais simples o processo de sua inversão. Esse e outros aspectos do sinal conseguem fazer a identificação do sistema ser a mais eficiente, com a menor variância do erro.

Um sinal que se aproxima de um ruído branco e pode ser facilmente gerado digitalmente é o proveniente de uma Sequência Binária Pseudoaleatória/*Pseudo-Random Binary Sequence* (*PRBS*). O PRBS é uma sequência de pulsos retangulares, com somente dois valores, na qual o intervalo de transição entre eles ocorre em múltiplos de um tempo Δt específico. Esse intervalo varia praticamente de maneira independente de seus valores anteriores, por isso o nome $pseudoaleatrio$. A Figura 13 mostra como um sinal *PRBS* é criado.

Figura 13 – Diagrama de blocos de um sinal *PRBS*



Fonte: (LANDOU *et al.*, 2011).

Temos células que apresentam sinais binários, 0 ou 1, e fazem a transição de seus valores para as células a direita a cada intervalo de tempo Δt. O sinal da primeira célula recebe a soma dos sinais das duas últimas células, e o sinal visto pelo *PRBS* é sempre o sinal presente na última célula. Podemos ver então que para se projetar um sinal *PRBS* temos dois parâmetros, a quantidade de células de sua estrutura, que denotaremos de N, e o intervalo de tempo na qual essas células transmitem seus valores, que denotaremos de T_b.

Temos então que o intervalo máximo para um pulso do sinal *PRBS* é a multiplicação do parâmetro N com o parâmetro T_b. Se o valor inicial de todas as células for 1, temos que o primeiro pulso será o pulso de intervalo máximo. Para uma melhor identificação do sistema é recomendável que o primeiro pulso do sinal *PRBS* consiga abranger toda a parte transitória da resposta, ou seja,

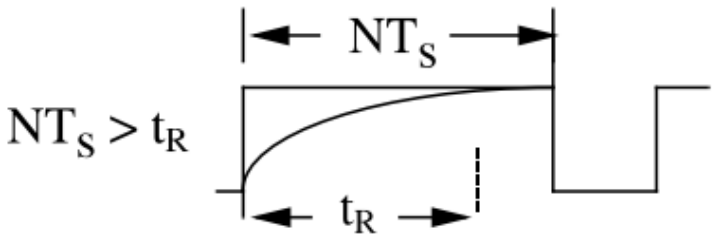
$$NT_b > t_R, \tag{2.21}$$

Onde

t_R: tempo de subida da resposta.

A visualização da Equação 2.21 pode ser vista na Figura 14.

Figura 14 – Pulso do sinal *PRBS* e tempo de subida do sistema



Fonte: (LANDOU *et al.*, 2011).

A Equação 2.21 apresenta um dos critérios que deve ser levado em conta para

o projeto do *PRBS*. Para conseguirmos os outros critérios temos que estabelecer a faixa de frequência na qual devemos excitar a entrada do sistema.

Para uma melhor identificação dos parâmetros do modelo, a entrada do sistema deve excitá-lo em frequências de acordo com as frequências dominantes da saída. Isto é, um teste com entrada em degrau pode ser feito no sistema, para se verificar o sinal de saída e as frequências dominantes do mesmo. O sinal *PRBS* deve então apresentar essas mesmas frequências dominantes.

De posse do intervalo de frequência requerido, as seguintes equações são sugeridas para se obter os parâmetros N e T_b (COELHO; COELHO, 2004):

$$\underline{\omega} = \frac{1}{\beta_S \tau_R} \leq \omega \leq \frac{\alpha_S}{\tau_L} = \overline{\omega}, \tag{2.22}$$

$$T_b \leq \frac{2,8\tau_L}{\alpha_S}, \tag{2.23}$$

$$2^N - 1 \geq \frac{2\pi\beta_S\tau_R}{T_b}, \tag{2.24}$$

Onde

$\underline{\omega}$: frequência mínima para o sinal *PRBS*, em radianos,

$\overline{\omega}$: frequência máxima para o sinal *PRBS*, em radianos,

ω: faixa de frequência para o sinal *PRBS*, em radianos,

τ_L: menor constate de tempo do sistema,

τ_R: maior constante de tempo do sistema,

β_S: fator representativo do tempo de estabilização,

α_S: fator representativo do tempo de resposta em malha fechada como um múltiplo do tempo de resposta em malha aberta.

Trabalhando as três equações, conseguimos reduzi-las para uma equação em função de N e T_b,

$$\frac{1}{(2^N - 1)T_b} \leq f_{desejada} \leq \frac{0,44}{T_b}, \tag{2.25}$$

Onde

$f_{desejada}$: frequência dominante da resposta do sistema ao degrau unitário.

Seguindo as Equações 2.21 e 2.25 iremos projetar os parâmetros N e T_b dos sinais *PRBS* do presente estudo.

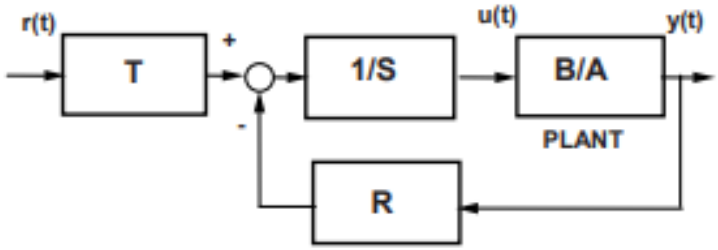
2.3.3 Controlador digital

Em controladores analógicos, como controladores *PI* e *PID*, são usadas em seu funcionamento operações proporcionais, integradores e/ou derivativas. Em sistemas digitais essas operações não são possíveis, sendo somente possível operações como adições, multiplicações e armazenamento de dados (LANDAU; ZITO, 2006).

Um controlador digital faz uso dessas operações, com o sinal de controle sendo originado da multiplicação por pesos dos valores atuais e passados da referência, da saída do sistema e do sinal de controle, e a posterior soma ou subtração desses valores. Os pesos são chamados de coeficientes do sistema, e cada parâmetro do controlador tem sua ordem, que define até quantas amostras passadas dos sinais serão utilizados nas operações.

Os controladores digitais apresentam a estrutura conhecida como *RST*. A Figura 15 mostra a estrutura RST, e a Figura 16 mostra a mesma estrutura, mas exemplificando os coeficientes de seus parâmetros.

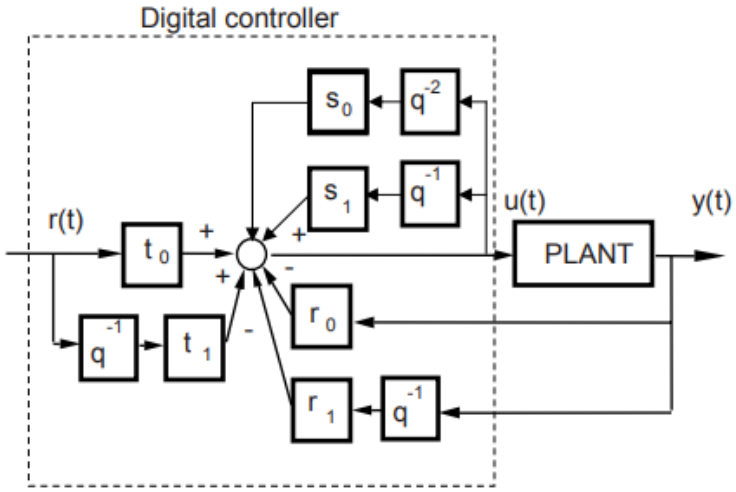
Figura 15 – Estrutura RST de um controlador digital



Fonte: (LANDAU; ZITO, 2006).

No exemplo da Figura 16 o parâmetro T apresenta ordem 1, com o coeficiente t_0 multiplicando a amostra atual da referência e o coeficiente t_1 multiplicando a referência obtida com uma amostra anterior. Já o parâmetro R também apresenta ordem 1, mas agora com o coeficiente r_0 multiplicando a amostra atual da saída, e o coeficiente r_1 a saída obtida com uma amostra passada. Por fim temos o parâmetro S com ordem 2, com o coeficiente s_1 multiplicando a amostra passada do sinal de controle, e o coeficiente s_2 multiplicando o sinal de controle com duas amostras passadas. O parâmetro S não apresenta o coeficiente multiplicando o sinal de

Figura 16 – Exemplo de um controlador RST e seus coeficientes



Fonte: (LANDAU; ZITO, 2006).

controle atual, pois o sinal atual não deve depender dele mesmo. Ao final os valores obtidos com as operações dos coeficientes de T e S são somados e subtraídos pelos obtidos com a operação dos coeficientes de R.

Todos os controladores digitais apresentam essa mesma estrutura, variando entre si somente a ordem dos parâmetros da mesma (LANDAU; ZITO, 2006).

Utilizando o diagrama da Figura 15, a função de transferência de malha fechada entre a referência e a saída fica representado nas Equações 2.26 e 2.27 (LANDAU; ZITO, 2006).

$$H_{MF}(q^{-1}) = \frac{B(q^{-1})T(q^{-1})}{A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1})} = \frac{B(q^{-1})T(q^{-1})}{P(q^{-1})}, \tag{2.26}$$

$$P(q^{-1}) = A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1}) = 1 + p_1 q^{-1} + p_2 q^{-2} + \cdots . \tag{2.27}$$

Temos então que o polinômio $P(q^{-1})$ determina os polos da função de transferência, e, por consequência, termina os parâmetros de desempenho da resposta do mesmo. A Equação 2.27 é comumente conhecida na literatura como *Equação Diofantina*.

2.3.3.1 Controlador PID digital

Já vimos a estrutura de um controlador digital. Vamos ver agora qual deve ser o formato de um controlador *PID* digital. A função de transferência de um controlador PID

contínuo se apresenta como (LANDAU; ZITO, 2006):

$$H_{PID}(s) = K\left[1 + \frac{1}{T_i s} + \frac{T_d s}{1 + \frac{T_d}{N} s}\right], \tag{2.28}$$

Onde

K: ganho proporcional,

T_i: ação integradora,

T_d: ação derivativa,

$\frac{T_d}{N}$: filtro passa-baixa para a ação derivativa.

Para a discretização do modelo contínuo vamos atribuir que a ação derivativa s no tempo discreto é aproximadamente $\frac{(1-q^{-1})}{T_S}$. Fazendo essa substituição na Equação 2.28 e trabalhando algebricamente obtemos a seguinte função de transferência em tempo discreto para o controlador *PID* (LANDAU; ZITO, 2006):

$$H_{PIDd}(q^{-1}) = \frac{R(q^{-1})}{S(q^{-1})} = K\left[1 + \frac{T_S}{T_i} \cdot \frac{1}{1-q^{-1}} + \frac{\frac{NT_S}{T_d+NT_S}(1-q^{-1})}{1 - \frac{T_d}{T_d+NT_S} q^{-1}}\right]. \tag{2.29}$$

Somando as três expressões da Equação 2.29 é fácil observar que os parâmetros R e S serão de ordem 2. Podemos então escrevê-los da seguinte forma:

$$R(q^{-1}) = r_0 + r_1 q^{-1} + r_2 q^{-2}, \tag{2.30}$$

$$S(q^{-1}) = 1 + s_1 q^{-1} + s_2 q^{-2} = (1-q^{-1})(1 + s_1^{'} q^{-1}), \tag{2.31}$$

Onde

$s_1^{'} = -\frac{T_d}{T_d+NT_S}$,

$r_0 = K\left(1 + \frac{T_S}{T_i} - N\frac{T_S}{T_d} s_1^{'}\right)$,

$r_1 = K\left[s_1\left(1 + \frac{T_S}{T_i} + 2N\frac{T_S}{T_d}\right) - 1\right]$,

$r_2 = -K s_1\left(1 + N\frac{T_S}{T_d}\right)$.

Podemos ver então que ao se projetar um controlador *PID* contínuo os parâmetros do mesmo podem ser usados para se projetar o controlador digital. Contudo essa metodologia não é prática, principalmente quando não é conhecido o modelo contínuo da planta e está deve ser identificada pelos métodos discretos explicitados anteriormente. Temos também a limitação

que a metodologia para o projeto do controlador *PID* explicitada só é aplicável para plantas que apresentam funções de transferência de no máximo segunda ordem.

Para conseguirmos projetar um controlador sem a necessidade do seu projeto contínuo, devemos utilizar uma metodologia para a resolução da *Equação Diofantina*. Mas antes da resolução dessa equação devemos conhecer os coeficientes desejados para o polinômio $P(q^{-1})$, e fazemos isso ao determinar os parâmetros desejados para a resposta do sistema de controle.

2.3.3.2 Parâmetros de desempenho

Os parâmetros de desempenho dos sistemas são facilmente determinados, se esses são de primeira ou segunda ordem. A Equação 2.32 caracteriza uma função de transferência contínua em malha fechada de segunda ordem.

$$H_{2O}(s) = \frac{\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}, \tag{2.32}$$

Onde

ξ: fator de amortecimento,

ω_n: frequência natural.

As Equações 2.33 e 2.34 demostram alguns dos parâmetros de desempenho mais importantes a serem determinados para um sistema de segunda ordem (DORF; BISHOP, 2001).

$$T_{a_{2\%}} \approx \frac{4}{\xi\omega_n}, \tag{2.33}$$

$$UP\% = 100e^{\frac{\xi\pi}{\sqrt{1-\xi^2}}}, \tag{2.34}$$

Onde

$T_{a_{2\%}}$: tempo de assentamento usando o critério de 2%, ou seja, tempo necessário para que a resposta do sistema atinge um erro de 2% em relação a resposta em regime permanente,

$UP\%$: ultrapassagem percentual, que denota o percentual máximo atingido pela resposta do sistema em relação ao regime permanente.

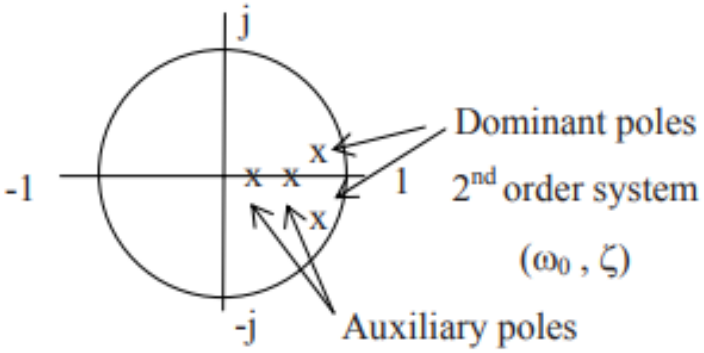
Utilizando as Equações 2.33 e 2.34 é possível determinar os parâmetros ξ e ω_n a partir do desempenho estabelecido quanto ao tempo de assentamento e a ultrapassagem percentual, e, por consequência, determinar os polos de malha fechada que satisfazem esses

parâmetros, com base na Equação 2.32. Esses polos encontrados são polos contínuos, e não podem ser utilizados para o controlador discreto. É necessário utilizar a função de transferência contínua determinada e encontrar sua função de transferência discreta correspondente. Essa transformação de uma função de transferência contínua para uma discreta pode ser facilmente implementada por meio de uma função no *software MATLAB*.

Encontrando a função de transferência discreta, temos os polos desejados do sistema. Esses polos devem ser os polos do polinômio $P(q^{-1})$, para que tenhamos um sistema de controle com os mesmos parâmetros projetados. Contudo, a função de transferência desejada é de segunda ordem, e é fácil observar pela *Equação Diofantina* que, se os parâmetros $A(q^{-1})$ e $B(q^{-1})$ forem de segunda ordem, o polinômio $P(q^{-1})$ terá uma ordem maior que dois. Para que o sistema de controle continue seguindo os padrões de desempenho os polos desejados devem ser os polos dominante do polinômio $P(q^{-1})$, e devem ser determinados polos auxiliares que influenciem pouco na resposta do sistema.

Para que esses polos auxiliares tenham pouca influência na resposta eles devem ter transitório rápido e alto amortecimento. Por isso a localização dos polos auxiliares no plano z deve ser no eixo real, para ter alto amortecimento, e a esquerda dos polos dominantes, para ter resposta mais rápida a estes e, portanto, com influência menor. A localização dos polos auxiliares em relação aos polos dominantes pode ser melhor vista na Figura 17.

Figura 17 – Localização dos polos auxiliares no plano z



Fonte: (LANDAU; ZITO, 2006).

De posse dos polos auxiliares e dos dominantes, é possível descobrir os coeficientes desejados do polinômio $P(q^{-1})$, e resolver a *Equação Diofantina* para descobrir os parâmetros R e S do controlador.

2.3.3.3 *Determinação dos parâmetros R, S e T do controlador*

Apresentando novamente a *Equação Diofantina*, temos

$P(q^{-1}) = A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1}) = 1 + p_1 q^{-1} + p_2 q^{-2} + \cdots$.

Com os coeficientes do polinômio determinados de acordo com o desempenho desejado do sistema, temos que determinar quais parâmetros R e S que satisfazem a equação.

Para o projeto do controlador RST devemos atribuir primeiro qual a ordem de cada parâmetro de controle. Vimos na Seção 2.3.3.1 que o controlador *PID* digital pode ser usado para plantas com funções de transferência de no máximo segunda ordem, e para esses projetos podemos utilizar ordem 2 para os parâmetros R e S do controlador.

É necessário, portanto, estabelecer a ordem dos parâmetros de controle para plantas que apresentem ordem maior que 2. Atribuindo n_S para a ordem do parâmetro S e n_R para a ordem do parâmetro R, segundo (LANDAU; ZITO, 2006), a *Equação Diofantina* apresentará uma solução única com o mínimo grau possível quando:

$n_S = n_B - 1$,

$n_R = n_A - 1$.

Temos então que os parâmetros R e S terão os seguintes formatos:

$$S(q^{-1}) = 1 + s_1 q^{-1} + s_2 q^{-2} + \cdots + s_{n_S} q^{-n_S}, \tag{2.35}$$

$$R(q^{-1}) = r_0 + r_1 q^{-1} + r_2 q^{-2} + \cdots + r_{n_R} q^{-n_R}. \tag{2.36}$$

No momento de projetar os parâmetros de controle, devemos atribuir algumas partes fixas aos mesmos, de modo a segurar que eles apresentem determinadas características. Os parâmetros com suas partes fixas se apresentam como:

$S(q^{-1}) = H_S(q^{-1})S'(q^{-1})$,

$R(q^{-1}) = H_R(q^{-1})R'(q^{-1})$,

Onde

$H_S(q^{-1})$: parte fixa atribuída ao parâmetro S,

$H_R(q^{-1})$: parte fixa atribuída ao parâmetro R,

$S'(q^{-1})$: parte restante do S ao se retirar sua parte fixa,

$R'(q^{-1})$: parte restante do R ao se retirar sua parte fixa.

Para que o sistema de controle tenha erro nulo em regime permanente devemos garantir uma ação integradora dentro do polinômio S. Um integrador em tempo discreto tem o formato de $(1-q^{-1})$ no domínio da frequência. Garantindo que o integrador faça parte de sua estrutura, o polinômio S se apresenta da seguinte forma:

$$S(q^{-1}) = (1-q^{-1})(1+s_1^{'}q^{-1}+\cdots+s_{n_S-1}^{'}q^{-(n_S-1)}), \tag{2.37}$$

$$H_S(q^{-1}) = (1-q^{-1}),$$

$$S^{'}(q^{-1}) = (1+s_1^{'}q^{-1}+\cdots+s_{n_S-1}^{'}q^{-(n_S-1)}).$$

Nesse presente estudo não iremos atribuir uma parte fixa ao parâmetro R, mantendo ele no formato da Equação 2.36 . De modo a simplificar a resolução da *Equação Diofantina*, vamos incluir o parâmetro fixo H_S dentro do parâmetro A da seguinte forma:

$$A^{'}(q^{-1}) = H_S(q^{-1})A(q^{-1}) = (1-q^{-1})(1+a_1q^{-1}+a_2q^{-2}+\cdots+a_{n_a}q^{-n_a}),$$

$$A^{'}(q^{-1}) = 1+a_1^{'}q^{-1}+\cdots+a_{n_a+1}^{'}q^{-(n_a+1)} .$$

Temos então que a Equação 2.27 se reapresentará da seguinte forma:

$$P(q^{-1}) = A^{'}(q^{-1})S^{'}(q^{-1})+B(q^{-1})R(q^{-1}) = 1+p_1q^{-1}+p_2q^{-2}+\cdots . \tag{2.38}$$

Atribuídos os passos dados anteriormente, para exemplificar como pode ser feita a resolução da Equação 2.38, vamos definir alguns padrões fixos e depois apresentar a resolução geral. Esses padrões fixos são a ordem dos parâmetros A e B. Definindo ambos os parâmetros como sendo de ordem 2, temos

$$A^{'}(q^{-1}) = 1+a_1^{'}q^{-1}+a_2^{'}q^{-2}+a_3^{'}q^{-3},$$

$$B(q^{-1}) = b_1q^{-1}+b_2q^{-2},$$

$$S^{'}(q^{-1}) = 1+s_1^{'}q^{-1},$$

$$R(q^{-1}) = r_0+r_1q^{-1}+r_2q^{-2}.$$

Resolvendo a Equação 2.38, com os parâmetros anteriores temos:

$$P(q^{-1}) = (1+a_1^{'}q^{-1}+a_2^{'}q^{-2}+a_3^{'}q^{-3})(1+s_1^{'}q^{-1})+(b_1q^{-1}+b_2q^{-2})(r_0+r_1q^{-1}+r_2q^{-2}) = 1+p_1q^{-1}+p_2q^{-2}+\cdots,$$

$$p_1 = b_1r_0+s_1^{'}+a_1^{'},$$

$$p_2 = b_2r_0+b_1r_1+s_1^{'}a_1^{'}+a_2^{'},$$

$$p_3 = b_2r_1+b_1r_2+s_1^{'}a_2^{'}+a_3^{'},$$

$p_4 = b_2 r_2 + s_1^{'} a_3^{'}$.

Para resolver o sistema de equações anterior de maneira eficiente podemos colocá-las na forma matricial, onde separamos uma matriz somente para os coeficientes dos parâmetros R e $S^{'}$ que queremos identificar. Obtemos então 3 matrizes:

$$x^T = \begin{bmatrix} 1 & s_1^{'} & r_0 & r_1 & r_2 \end{bmatrix},$$

$$p^T = \begin{bmatrix} 1 & p_1 & p_2 & p_3 & p_4 \end{bmatrix},$$

$$M = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ a_1^{'} & 1 & b_1 & 0 & 0 \\ a_2^{'} & a_1^{'} & b_2 & b_1 & 0 \\ a_3^{'} & a_2^{'} & 0 & b_2 & b_1 \\ 0 & a_3^{'} & 0 & 0 & b_2 \end{bmatrix}.$$

Que obedecem a seguinte equação:

$$Mx = p.$$

Por fim a matriz com coeficientes de R e $S^{'}$ fica:

$$x = M^{-1} p. \tag{2.39}$$

Com os coeficientes encontramos pela lógica matricial, e aplicando-os nas Equações 2.36 e 2.37 encontrarem os parâmetros R e S do sistema de controle.

As Equações 2.40, 2.41 e 2.42 apresentam a resolução da Equação 2.38 de forma matricial para sistemas de qualquer ordem.

$$x^T = \begin{bmatrix} 1 & s_1^{'} & \cdots & s_{n_S-1}^{'} & r_0 & \cdots & r_{n_R} \end{bmatrix}, \tag{2.40}$$

$$p^T = \begin{bmatrix} 1 & p_1 & \cdots & p_i & \cdots & pn_p & 0 & \cdots & 0 \end{bmatrix}, \tag{2.41}$$

$$M = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & \cdots & \cdots & 0 \\ a_1^{'} & 1 & \cdots & 0 & b_1 & 0 & \cdots & 0 \\ a_2^{'} & a_1^{'} & \cdots & 0 & b_2 & b_1 & \cdots & 0 \\ \vdots & \vdots & \vdots & 1 & \vdots & \vdots & \vdots & b_1 \\ a_{n_a}^{'} & \cdots & \cdots & a_1^{'} & b_{n_b} & \cdots & \cdots & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & \cdots & a_{n_a}^{'} & 0 & \cdots & \cdots & b_{n_b} \end{bmatrix}. \tag{2.42}$$

Por fim, falta a definição do parâmetro T. Para determiná-lo devemos voltar novamente à função de transferência do sistema de controle em malha fechada:

$$H_{MF}(q^{-1}) = \frac{B(q^{-1})T(q^{-1})}{A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1})} = \frac{B(q^{-1})T(q^{-1})}{P(q^{-1})}. \tag{2.43}$$

Devemos estabelecer um valor para o parâmetro T de modo que a saída tenha o valor da referência em regime permanente, ou seja, a função de transferência deve ter valor 1 nessa faixa de resposta.

Como uma função integradora apresenta valor em regime permanente igual a zero, ao adicionarmos essa função no parâmetro S, esse também assume o valor de zero. Temos então que em regime permanente a multiplicação $A(q^{-1})S(q^{-1})$ é igual a zero, e a Equação 2.43 se transforma em:

$$H_{MF}(1) = \frac{B(1)T(1)}{B(1)R(1)}.$$

Pode ser observado que para que a saída tenha o valor da referência em regime permanente o parâmetro T deve ser igual ao R. Para não surgir problemas ao adicionarmos zeros na função, o parâmetro T será igual ao parâmetro R durante o regime permanente, ou seja, o parâmetro T será uma constante com valor igual à soma dos coeficientes de R.

3 PROJETO DE CONTROLADOR RST PARA O CONTROLE DOS MÚSCULOS DE FLEXÃO E EXTENSÃO DO COTOVELO

Como visto na Seção 2.2, o *software Opensim* não dispõe de ferramentas para que se possa fazer um controle em malha fechada dos modelos estudados. Para se fazer isso é necessário usar a *API* do sistema junto a outros *softwares*. O *software* utilizado nesse trabalho em conjunto com o *Opensim* foi o *software MATLAB*, e para fazer a comunicação entre os dois foram usados códigos disponíveis pela própria equipe de desenvolvimento do *Opensim*. Para realizar a comunicação e análise de dados em malha fechada foi usado como base o código disponibilizado por (SOUSA *et al.*, 2019), sendo este alterado e adaptado para se adequar aos objetivos do presente estudo.

Como visto na Seção 2.2.1, o modelo estudado apresenta somente dois graus de liberdade, o ângulo do cotovelo e o ângulo do ombro, e a posição do ombro influência nos limites de atuação dos músculos de flexão e extensão do cotovelo. Por conta disso, para uma melhor identificação de cada modelo, diferentes angulações para o ombro foram estabelecidas no estudo da flexão e da extensão, com o intuito de utilizar a posição do braço onde os músculos fossem mais exigidos.

Na prática, durante um movimento, cada músculo é contraído em diferentes níveis, e com isso, cada músculo apresenta um sinal de controle distinto. Como esse modelo consta com 3 músculos que controlam a flexão do cotovelo (músculos braquial, biceps cabeça longa e bíceps cabeça curta) e 3 que controlam a extensão (músculos tríceps cabeça medial, tríceps cabeça lateral e tríceps cabeça longa), para uma simplificação foi projetado que um mesmo sinal de controle seria enviado para os 3 músculos da flexão, e outro sinal de controle seria enviado para os 3 músculos da extensão.

Como temos duas entradas que atuam em movimentos antagônicos, foram projetados dois controladores RST, um para a flexão e outro para a extensão. Com isso, cada movimento foi primeiramente identificado, testado e controlado separadamente, e, posteriormente, foi feita a análise dos dois controladores RST funcionando em paralelo.

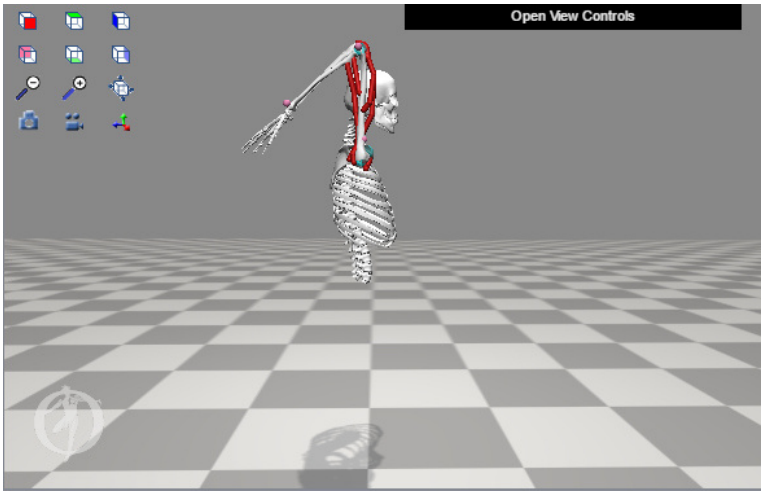
3.1 Estudo do controle da extensão do cotovelo

O primeiro passo para estudar o movimento de extensão do modelo simulado foi estabelecer em qual posição do braço ele teria uma maior atuação. Como o presente estudo é feito para o ângulo do cotovelo a posição de estudo seria determinada ao estabelecermos uma

posição fixa para o ângulo do ombro.

Como podemos ver na Figura 18, o ângulo de 180° para o ombro é a posição que mais exige a atuação da extensão. Sendo essa a ação de força protagonista no movimento do cotovelo em boa parte de seu intervalo de angulação.

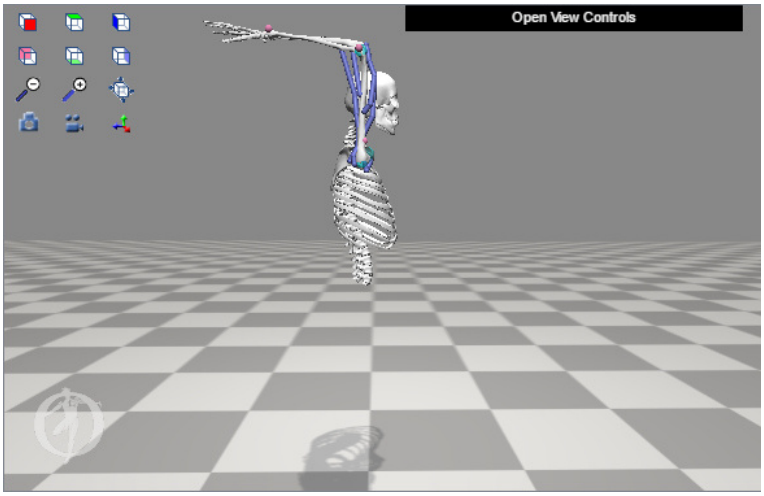
Figura 18 – Modelo estudado com ângulo do ombro em 180°



Fonte: Próprio Autor.

Foi feita uma simulação com o ombro em 180°, sem nenhum estímulo em seus músculos, para saber em qual ângulo do cotovelo ocorreria uma estabilização do braço. A posição de equilíbrio do cotovelo apresentou uma angulação de aproximadamente 79° como pode ser visto na Figura 19.

Figura 19 – Posição sem estímulos com ângulo do ombro em 180°



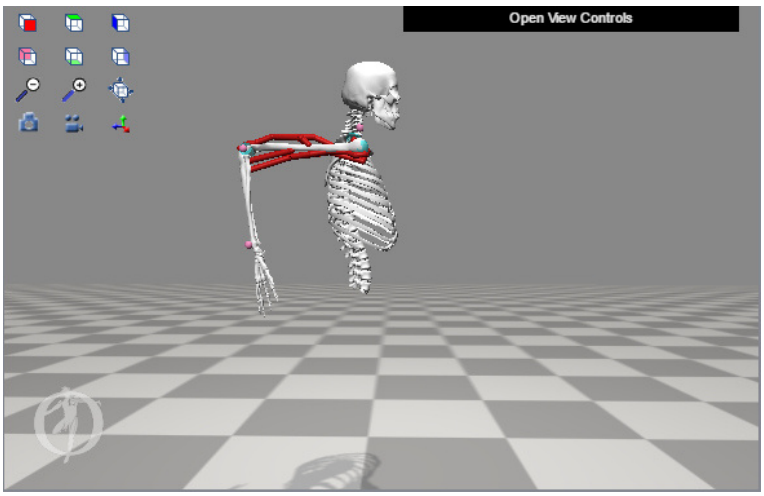
Fonte: Próprio Autor.

A estabilização nessa posição, quando o braço não apresenta estímulos, não condiz

com o movimento real de uma pessoa sem limitações no braço, pois a força peso do mesmo o puxa para baixo, fazendo a angulação do cotovelo atingir valores mais elevados. Vemos então que essa posição apresenta uma limitação do modelo *Arm*26.

Por conta dessa inconstância, outra posição fixa para o ombro foi estabelecida. A posição escolhida foi o ângulo de -90°, que pode ser visto na Figura 20 já com seu ângulo de cotovelo estabilizado quando não há estímulo de seus músculos.

Figura 20 – Posição sem estímulos com ângulo do ombro em -90°



Fonte: Próprio Autor.

Nessa posição o braço se estabilizou com uma angulação de cotovelo de 95,8°, fazendo com que o estímulo dos músculos de extensão consiga produzir movimentos que vão do intervalo de 0° à 95,8°. Esse intervalo foi considerado bom para a análise do movimento e essa posição foi a escolhida para o estudo da extensão do cotovelo.

Estabelecida a posição de estudo, o primeiro passo para a definição dos parâmetros de controle é estabelecer o tempo de amostragem do sistema. Já vimos que a planta estudada, por ser uma simulação, apresenta natureza discreta, e faremos o sistema de controle no *software MATLAB* se comunicando com o modelo *Arm*26 no *software Opensim*. Contudo, essa comunicação é feita com um tempo de amostragem muito pequeno, e, para uma maior eficiência do sistema, deve ser projetado um tempo de amostragem maior.

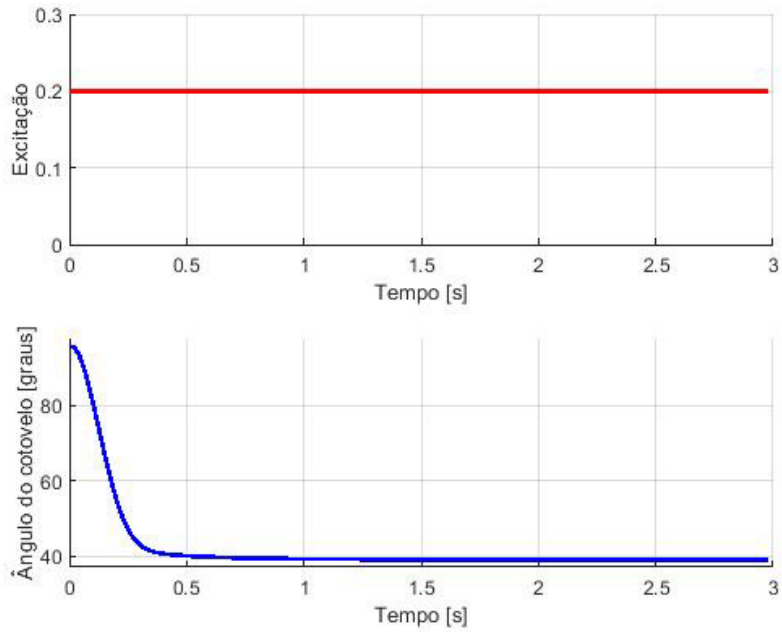
3.1.1 Resposta do sistema a entrada degrau e tempo de amostragem

O primeiro passo para a identificação do modelo foi aplicar um degrau como sinal de ativação para os 3 músculos responsáveis pela extensão do braço. Para esse teste inicial foi

estabelecido uma frequência de amostragem de 50Hz. Diferentes valores para esse degrau foram testados, com o intuito de ver como o ângulo do cotovelo varia a medida que diferentes níveis de ativação são colocados nos músculos. Como na simulação a única força contrária ao movimento é a força peso do braço, pequenos estímulos já geravam um movimento com angulação elevada. Depois desses testes foi estabelecido para a análise do sistema uma entrada em degrau com magnitude de 0,2 de ativação. Os códigos utilizados para a simulação do sistema com entrada degrau podem ser vistos nos Apêndices A e B.

A resposta a entrada em degrau do sistema, mantendo o ângulo do ombro fixo em -90°, e a posição inicial do cotovelo em 95,8° pode ser vista no Figura 21.

Figura 21 – Resposta do sistema a entrada em degrau nos músculos de extensão

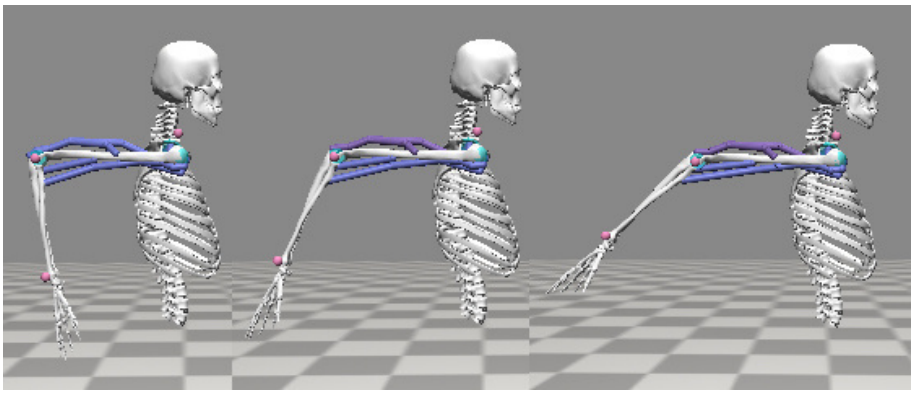


Fonte: Próprio Autor.

A Figura 22 mostra as diferentes posições adotadas durante o movimento no *software Opensim*.

Analisando o gráfico da resposta a uma entrada em degrau, foi estabelecido que o tempo de resposta do sistema estudado, tempo para a resposta ir de 10% à 90% da diferença entre a posição final e inicial, foi de aproximadamente 0,24s. Para esse estudo foi estipulado que a frequência de amostragem ideal do mesmo deve ser capaz de gerar pelo menos 10 amostras no tempo de resposta do sistema, e não se distanciar desse valor. Com a frequência de amostragem inicial de 50 Hz foi possível ter 12 amostras no intervalo de resposta de 0,24s, valor considerado

Figura 22 – Movimento do modelo com entrada em degrau nos músculos de extensão

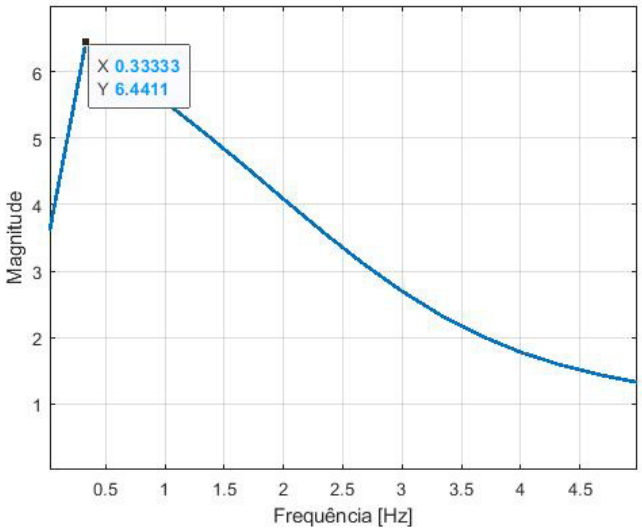


Fonte: Próprio Autor.

bom para esse estudo, sendo essa a frequência de amostragem adotada para o mesmo.

Em seguida o sinal de saída do sistema teve seu valor contínuo retirado para que a Transformada de Fourier Rápida do sinal fosse feita, com o intuito de determinar as frequências dominantes do mesmo. O gráfico da magnitude da saída em função da frequência pode ser visto na Figura 23.

Figura 23 – Frequência dominante da saída à entrada degrau nos músculos de extensão



Fonte: Próprio Autor.

Podemos ver pelo gráfico acima que a frequência dominante tem seu valor estabelecido em aproximadamente 0,33 Hz.

Como foi visto no referencial teórico, a entrada em degrau não é ideal para que possa se identificar o modelo *ARX* do sistema. Para isso iremos projetar um sinal *PRBS* com base na frequência dominante da resposta do sistema vista pela entrada em degrau.

3.1.2 Aplicação de entrada PRBS

Para o dimensionamento do sinal *PRBS*, usaremos a frequência dominante de 0,33 Hz. Como visto na Seção 2.3.2.4, devemos ter dois parâmetros para conseguir obter um sinal *PRBS*, o parâmetro N e o parâmetro T_b, que são escolhidos de acordo com as Equações 2.21 e 2.25.

$$NT_b \leq t_R,$$

$$\frac{1}{(2^N-1)T_b} \leq f_{desejada} \leq \frac{0,44}{T_b}.$$

Devemos definir uma faixa de frequência que contenha a frequência dominante de 0,33 Hz. A faixa escolhida foi:

$$0 \leq f_{desejada} \leq 2.$$

Para definir T_b temos que,

$$\frac{0,44}{T_b} \approx 2,$$

$$T_b \approx 0,22.$$

Escolhemos então o valor de $T_b = 0,25$. Para escolher um valor de N, temos que,

$$\frac{1}{(2^N-1)T_b} \approx 0,$$

$$NT_b \leq t_R.$$

O valor que consegue obedecer essas expressões é $N = 7$, pois

$$\frac{1}{(2^7-1)0,25} \approx 0,031,$$

$$7 \cdot 0,25 \leq t_R.$$

Como visto na Seção 3.1.1 o tempo de resposta (t_R) do sistema é de aproximadamente 0,24s, fazendo com que a escolha de $N = 7$ e $Tb = 0,25$ satisfaça as duas equações propostas.

O próximo passo para projetar o sinal *PRBS* foi definir a sua magnitude. Foram feitos testes para definir qual variação da entrada gerava respostas dentro de uma faixa desejada. Foi observado que pequenas variações de ativação dos músculos já geravam resposta de grande magnitude no sistema. Por conta disso uma variação pequena de 0,02 foi escolhida para atuar

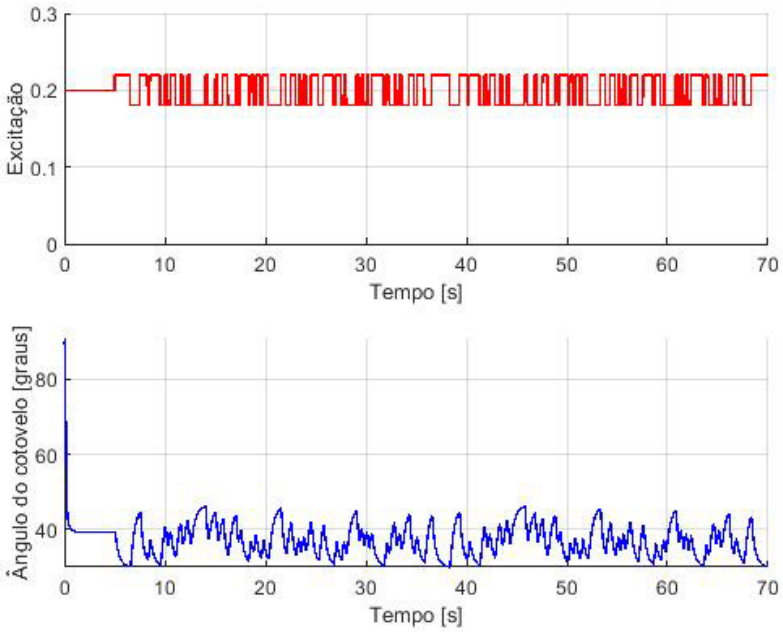
dentro do valor de 0,2 de magnitude colocado anteriormente na entrada em degrau, fazendo a entrada variar em uma magnitude de 0,18 a 0,22 com a frequência estipulada para o *PRBS*.

Por fim, para iniciar a entrada *PRBS* quando o sistema já estivesse em regime permanente na faixa de análise desejada, o sinal só foi adicionado 5 segundos após uma entrada em degrau de 0,2 de magnitude.

Esses parâmetros foram colocados em um bloco de simulação dentro do *software Simulink*, e depois passados para o *software MATLAB* para posteriormente serem simulados como entrada dentro do *Opensim*.

A Figura 24 mostra o sinal de entrada *PRBS* e a resposta do sistema que foi obtida a partir desse.

Figura 24 – Resposta do sistema a entrada PRBS nos músculos de extensão



Fonte: Próprio Autor.

Com a entrada *PRBS* temos um sinal de resposta de um sistema que é estimulado persistentemente e, com isso, capaz de identificar o sistema estudado com maior precisão.

3.1.3 Identificação do modelo ARX para a extensão do cotovelo

Com o sinal de entrada *PRBS* e a resposta do sistema é possível utilizar o método dos mínimos quadrados, visto na Seção 2.3.2.2 para conseguir estabelecer os parâmetros do modelo *ARX* que descrevem com acurácia a planta do sistema. Para utilizar essa metodologia

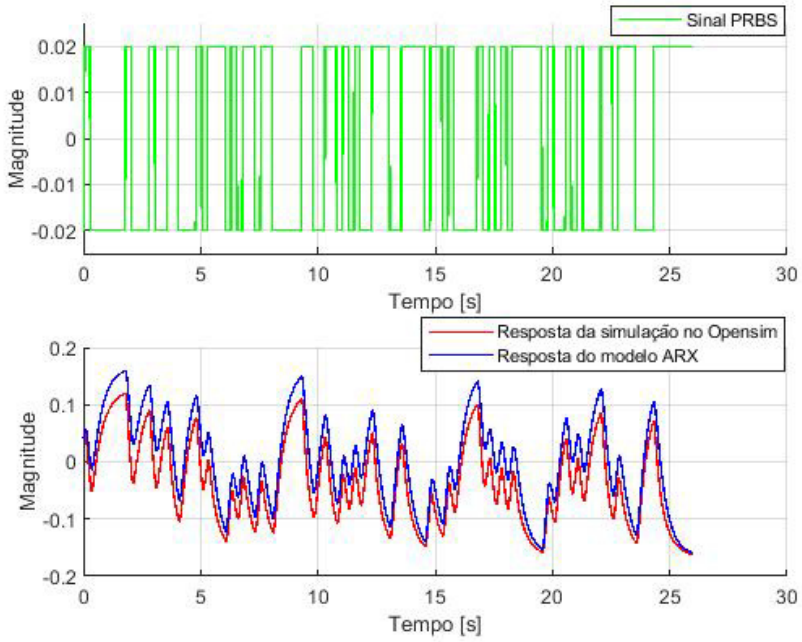
foi usado um código dentro do *software MATLAB*, que atua recebendo a entrada e a saída do sistema simulado e devolve os parâmetros B e A do modelo ARX. Esse código de identificação pode ser visto no Apêndice C.

Dos sinais de entrada e saída vindos da simulação no Opensim, 60% foram utilizados para obter o modelo *ARX*, que foi inicialmente projetado para ser de ordem 2 nos parâmetros A e B. Utilizando o programa, a planta de estudo se assemelhou a seguinte função de transferência no tempo discreto:

$$\frac{y(k)}{u(k)} = \frac{-0,0239q^{-1} - 0,0673q^{-2}}{1 - 1,7215q^{-1} + 0,7325q^{-2}}. \tag{3.1}$$

Com os parâmetros da planta obtido, os outros 40% dos dados de entrada e saída foram utilizados para serem aplicados no modelo e comparar os valores obtidos com os valores reais da simulação. Esse último passo é necessário para sabermos se o modelo obtido se assemelha a resposta real de maneira satisfatória. A Figura 25 mostra a comparação entre as respostas obtidas aplicando-se o sinal *PRBS* no modelo *ARX* e a resposta real obtida por meio da simulação no *Opensim*.

Figura 25 – Comparação entre a saída real e a saída estimada



Fonte: Próprio Autor.

Podemos ver que visualmente os valores estimados pelo modelo *ARX* estão próximos dos valores reais, porém uma análise visual não é suficiente para que se possa avaliar o desempenho.

Para que esse desempenho possa ser avaliado de maneira assertiva foram utilizados os métodos presentes na Seção 2.3.2.3. Primeiramente foi calculada a *Correlação Múltipla* do sinal real e do estimado, e foi obtido um valor de 0,6848. Esse valor se encontra fora do intervalo de 0,9 à 1 visto na Seção 2.3.2.3.

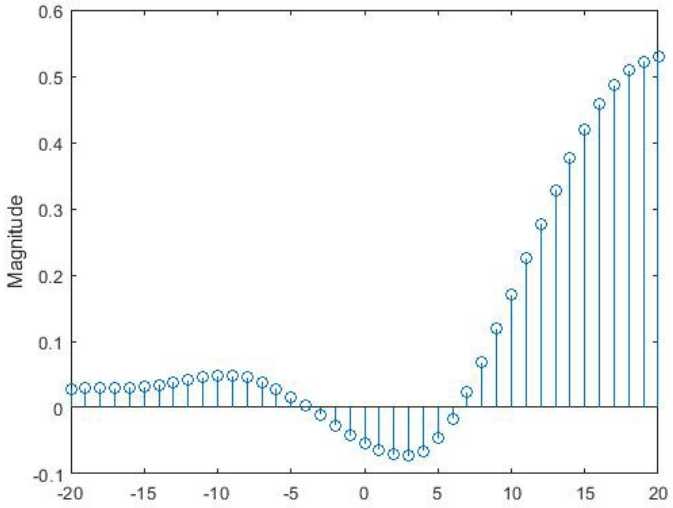
As outras formas de avaliação da identificação do sistema foram a correlação cruzada entre resídio e entrada e a auto correlação do resíduo. Vimos na Equação 2.20 um critério para determinar qual valor deve ser atingido por esses índices para que o resíduo possa ser considerado como um ruído branco. Substituindo o valor de N por 1299, quantidade de amostras do experimento, temos:

$$r_{u\varepsilon}, r_{\varepsilon\varepsilon} \leq \frac{2,17}{\sqrt{1299}},$$

$$r_{u\varepsilon}, r_{\varepsilon\varepsilon} \leq 0,06.$$

As Figuras 26 e 27 apresentam os gráficos com a correlação cruzada e a auto correlação, respectivamente.

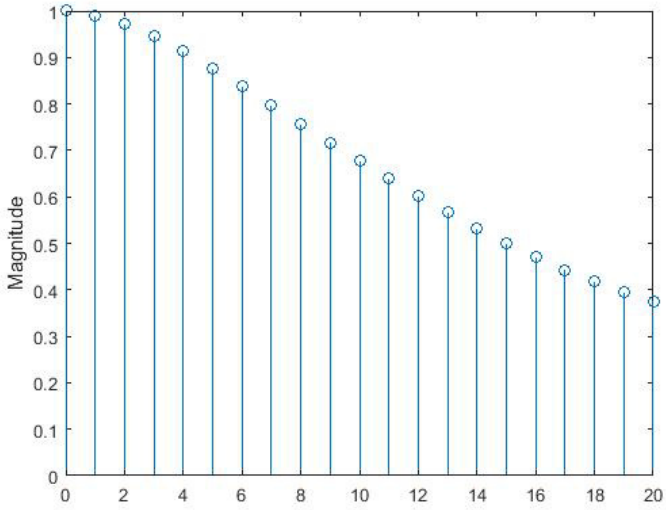
Figura 26 – Correlação cruzada do modelo identificado de 2ª ordem para a extensão



Fonte: Próprio Autor.

Podemos ver pelos gráficos que os índices dos mesmos superaram muito o valor de 0,06 estabelecido anteriormente. Por conta disso foi feita uma nova identificação do sistema, agora utilizando ordem 3 para os parâmetros A e B do modelo ARX. A função de transferência

Figura 27 – Autocorrelação do modelo identificado de 2ª ordem para a extensão



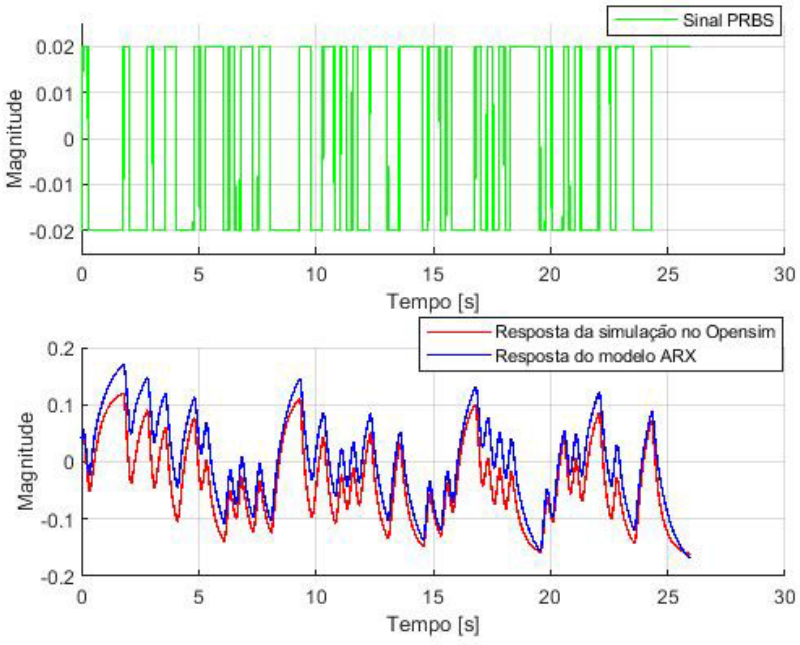
Fonte: Próprio Autor.

do modelo obtido foi:

$$\frac{y(k)}{u(k)} = \frac{-0,0189q^{-1} - 0,0608q^{-2} + 0.0579q^{-3}}{1 - 2,5545q^{-1} + 2,1992q^{-2} - 0,6425q^{-3}}. \tag{3.2}$$

A Figura 28 mostra a comparação entre as respostas obtidas aplicando-se o sinal *PRBS* no modelo *ARX* de 3ª ordem e a resposta real obtida por meio da simulação no *Opensim*.

Figura 28 – Comparação entre a saída real e a saída estimada com modelo de 3ª ordem

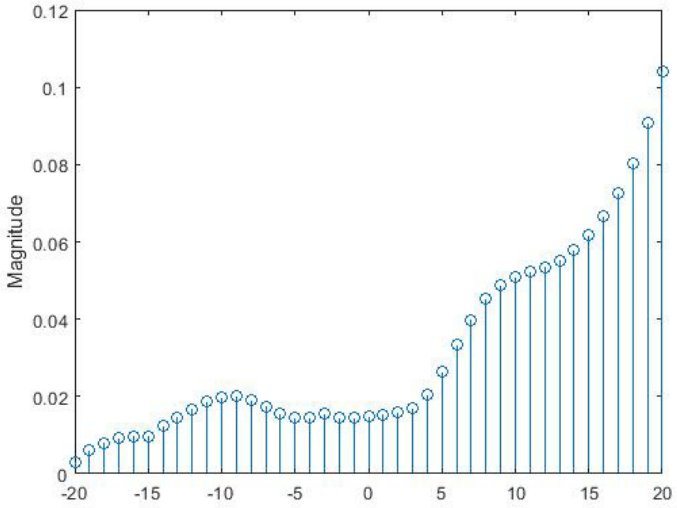


Fonte: Próprio Autor.

Posteriormente, fazendo-se o cálculo para a *Correlação Múltipla* foi obtido um valor de 0,6365, valor um pouco inferior ao do modelo identificado de 2ª ordem, mas, por sua proximidade, podemos considerá-los iguais.

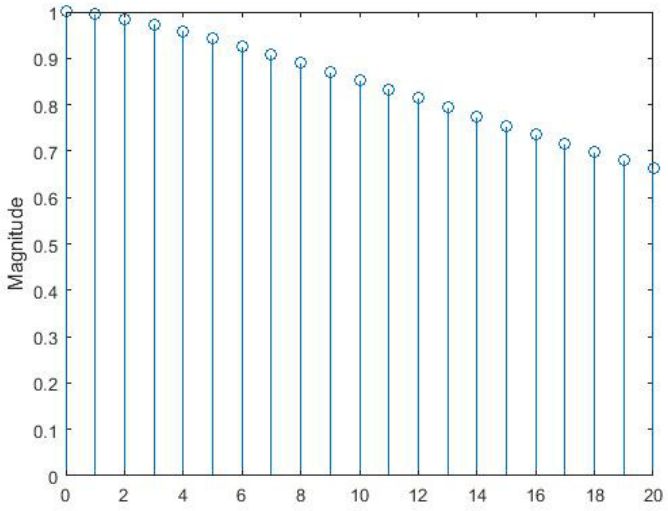
Por fim, os gráficos para a correlação cruzada e para a autocorrelação do novo modelo podem ser vistos nas Figuras 29 e 30.

Figura 29 – Correlação cruzada do modelo identificado de 3ª ordem para a extensão



Fonte: Próprio Autor.

Figura 30 – Autocorrelação do modelo identificado de 3ª ordem para a extensão



Fonte: Próprio Autor.

Podemos ver uma melhora nos índices de correlação cruzada, mas uma leve piora

nos índices de auto correlação. A melhora obtida com o modelo de 3ª ordem foi considerada pequena, e por isso o modelo mais simples de 2ª ordem foi o utilizado para o decorrer do estudo.

3.1.4 Identificação dos parâmetros de controle RST

A função de transferência de um sistema RST tem o seguinte formato:

$$H_{MF}(q^{-1}) = \frac{B(q^{-1})T(q^{-1})}{A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1})} = \frac{B(q^{-1})T(q^{-1})}{P(q^{-1})}.$$

O primeiro passo para definição dos polinômios R, S e T é a definição dos parâmetros que queremos para o sistema de malha fechada. Foi estabelecido que o sistema em malha fechada deve apresentar tempo de assentamento de 1 segundo e máximo percentual de ultrapassagem de 0%. Com esses parâmetros, utilizando as Equações 2.33 e 2.34, uma função de transferência contínua de segunda ordem foi projetada, e depois foi discretizada, obtendo a seguinte função:

$$H_d = \frac{0,001843q^{-1} + 0,001773q^{-2}}{1 - 1,887q^{-1} + 0,8904q^{-2}}. \tag{3.3}$$

Temos então que para ter os padrões de desempenho desejados a função H_{MF} deverá apresentar os mesmos polos da função H_d. Contudo, se analisarmos que os polinômios B e A dimensionados no sistema apresentam ordem 2 e usarmos um controle PID discreto, temos que o polinômio H_{MF} apresentará ordem 4, e temos que acrescentar mais dois polos ao sistema desejado. Com o intuito de não influenciar nos parâmetros desejados do sistema os dois polos auxiliares escolhidos foram os polos 0,0 e 0,1, pois estes apresentam valores reais e de alto amortecimento para o sistema. Com isso um novo polinômio desejado para o denominador da função foi obtido, sendo:

$$P(q^{-1}) = 1 - 1,9868q^{-1} + 1,0791q^{-2} - 0,0890q^{-3} + 0q^{-4}. \tag{3.4}$$

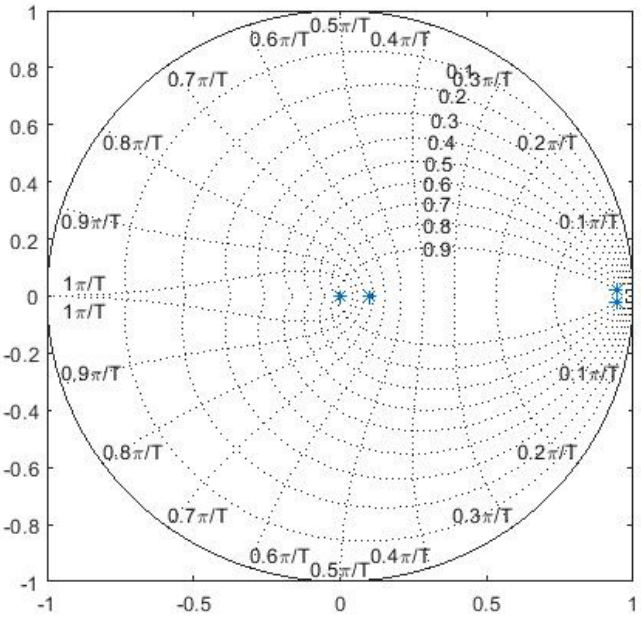
O gráfico das raízes desse polinômio pode ser visto na Figura 31.

Pela posição das raízes auxiliares, as duas raízes mais a esquerda, podemos ver que elas apresentam frequência e amortecimento que não terão uma influência maior do que as raízes desejadas.

Estabelecido o polinômio desejado para a função, por meio de um código no *software MATLAB*, construído a partir da metodologia de resolução presente na Seção 2.3.3.3, foram calculados os parâmetros R, S e T:

$$R = -7,1084 + 13,2230q^{-1} - 6,1503q^{-2}, \tag{3.5}$$

Figura 31 – Polos dominantes e auxiliares para a função de transferência de controle



Fonte: Próprio Autor.

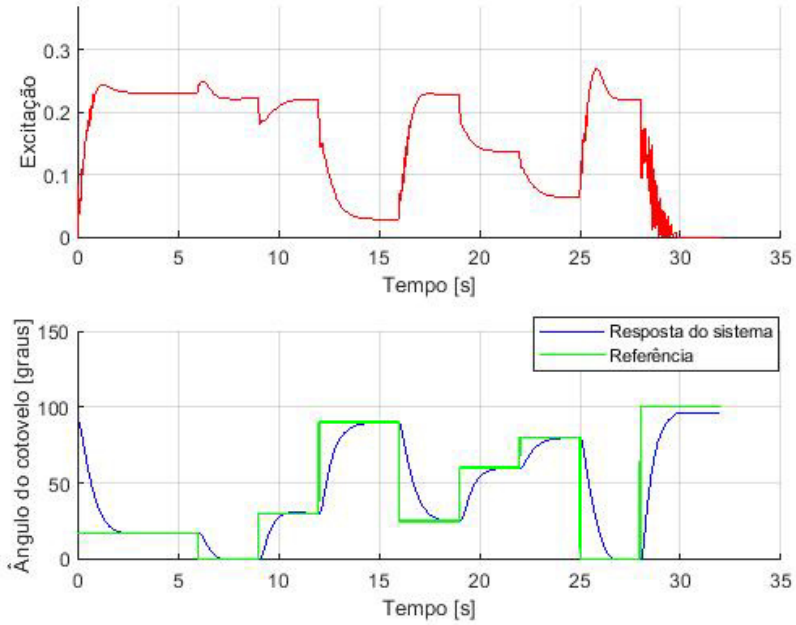
$$S = 1,0000 - 0,4353q^{-1} - 0,5647q^{-2}, \tag{3.6}$$

$$T = -0.0357. \tag{3.7}$$

Com os parâmetros R, S e T estabelecidos, um novo código para o sinal de controle enviado ao *Opensim* foi elaborado no *MATLAB*. Foi projetado para que ao longo da simulação o valor de referência para o ângulo do cotovelo fosse mudando em diferentes magnitudes. A resposta em malha fechada do controlador RST atuando nos músculos de extensão do braço pode ser visto na Figura 32.

Pode ser visto que a resposta em malha fechada do controlador está seguindo a referência sem ultrapassagem e com tempo de assentamento próximo de 1 segundo, parâmetros estabelecidos anteriormente. A referência só não é seguida quando ela passa a valer 100°. Esse comportamento já era esperado do controlador, pois com o ângulo do ombro estabelecido como -90° o trabalho de extensão só poderia ser feito para ângulos menores do que 95,8°. Para ângulos maiores do que esse valor o trabalho que deve ser feito é o de flexão. Como os músculos de extensão têm esse limite de atuação, quando a referência passa para um valor maior a esse limite o ângulo tende a se estabilizar nesse valor.

Figura 32 – Sinal de controle e a resposta dos músculos de extensão



Fonte: Próprio Autor.

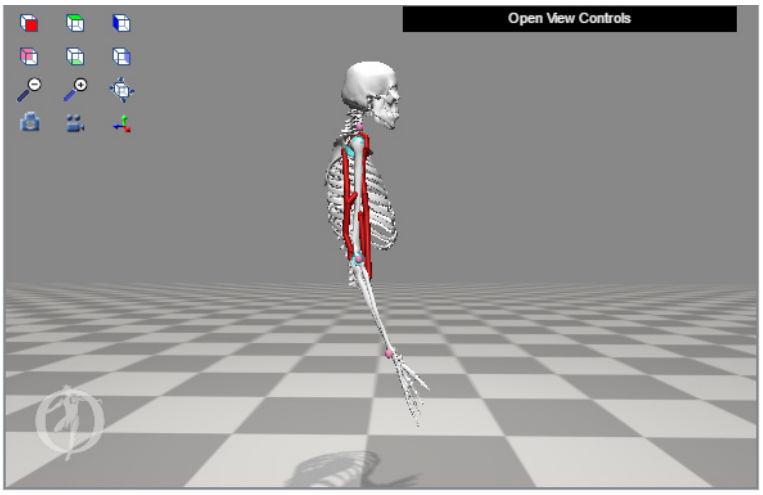
Podemos então ver pela simulação que com o controle somente dos músculos encarregados da extensão do cotovelo, não conseguimos cobrir todos os limites de ângulos que existem para o movimento, sendo necessário estabelecer também o controle da flexão do cotovelo.

3.2 Estudo do controle da flexão do cotovelo

Para o movimento de estímulo aos músculos de flexão foi estabelecida que a posição que mais exige dos mesmos é quando o ângulo do ombro apresenta um valor de 0°, sistema esse presente na Figura 33.

Foi estabelecido que o ângulo do ombro deveria permanecer fixo em 0°, e depois foi aplicada uma simulação onde nenhum músculo foi estimulado, para saber em qual ângulo do cotovelo o braço iria se estabilizar. Como visto na imagem, sem nenhuma força e com o ombro em 0°, o braço se estabilizou com uma angulação de cotovelo igual à 18,5°. Isso condiz com a realidade, pois por conta da largura dos músculos e da força passiva dos mesmos, quando não recebe nenhum estímulo o braço se estabiliza em um ângulo de cotovelo maior que zero. Portanto essa foi a posição adotada para os testes que identificaram o modelo de controle da flexão do braço.

Figura 33 – Posição sem estímulos com ângulo do ombro em 0°



Fonte: Próprio Autor.

3.2.1 Resposta do sistema a entrada degrau e tempo de amostragem

O primeiro passo para a identificação do modelo foi aplicar um degrau como sinal de ativação para os 3 músculos responsáveis pela flexão do braço. Para esse teste inicial foi estabelecido uma frequência de amostragem de 50Hz. Diferentes valores para esse degrau foram testados, com o intuito de ver como o ângulo do cotovelo varia a medida que diferentes níveis de ativação são colocados nos músculos. Depois desses testes foi estabelecido para a análise do sistema uma entrada em degrau com magnitude de 0,06.

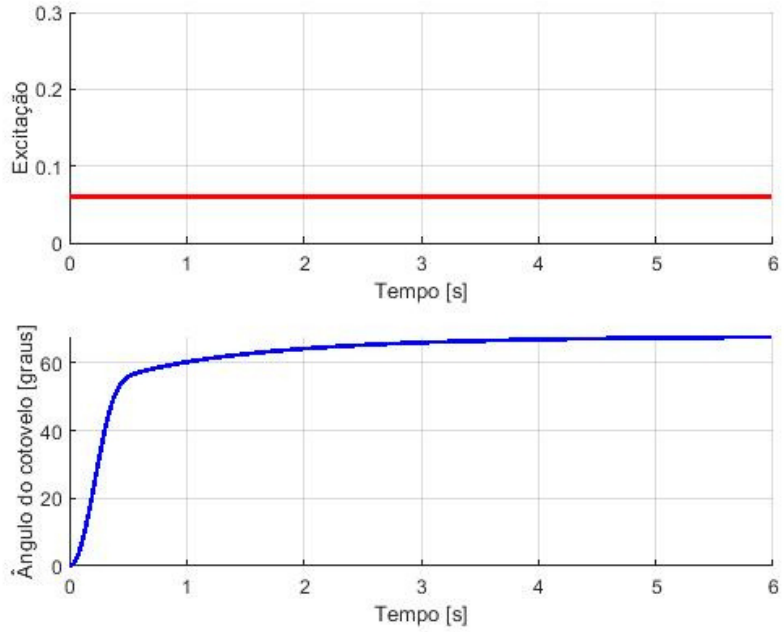
A resposta a entrada em degrau do sistema, mantendo o ângulo do ombro fixo em 0°, pode ser vista na Figura 34.

A Figura 35 mostra as diferentes posições adotadas durante o movimento do software Opensim.

Analisando o gráfico da resposta a uma entrada em degrau, foi estabelecido que o tempo de subida do sistema estudado foi de aproximadamente 1,1s. Se estivéssemos controlando somente a flexão do braço, com esse tempo de resposta a frequência de amostragem poderia ser menor, mas vamos controlar também a extensão do braço, que como vimos anteriormente, necessita de uma frequência de amostragem perto de 50 Hz, por isso vamos fazer todo o estudo de identificação e controle da flexão também com uma frequência de 50 Hz.

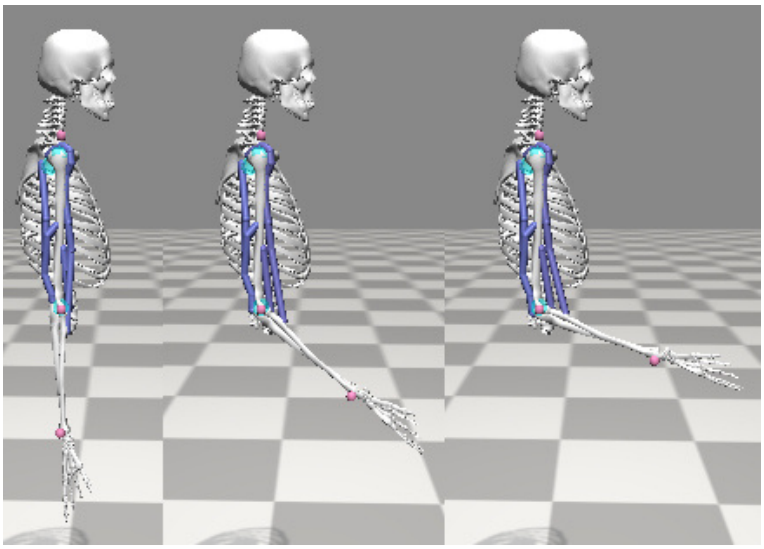
Em seguida, a Transformada de Fourier Rápida do sinal de saída foi feita, para se determinar as frequências dominantes do mesmo. O gráfico da magnitude da saída em função da frequência pode ser visto na Figura 36.

Figura 34 – Resposta do sistema a entrada em degrau nos músculos de flexão



Fonte: Próprio Autor.

Figura 35 – Movimento do modelo com entrada em degrau nos músculos de flexão



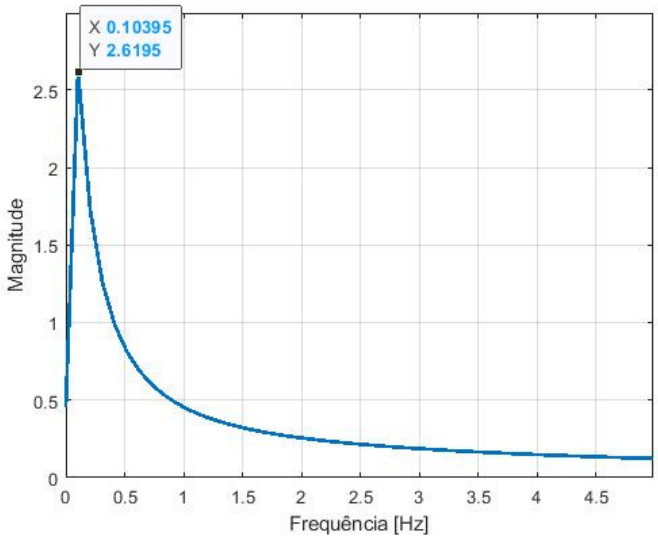
Fonte: Próprio Autor.

Podemos ver pelo gráfico acima que a frequência dominante tem seu valor estabelecido em aproximadamente 0,10 Hz, e iremos projetar um sinal *PRBS* com base nessa frequência.

3.2.2 Aplicação de entrada PRBS

Para o dimensionamento do sinal *PRBS*, usaremos a mesma metodologia feita no dimensionamento da extensão do braço, usando a frequência dominante de 0,10 Hz.

Figura 36 – Frequência dominante da saída à entrada degrau nos músculos de flexão



Fonte: Próprio Autor.

Considerando a faixa de frequência dominante temos:

$$0 \leq f_{desejada} \leq 2.$$

Para definir T_b temos que,

$$\frac{0,44}{T_b} \approx 2,$$

$$T_b \approx 0,22.$$

Escolhemos então o valor de $T_b = 0,25$. Para escolher um valor de N, temos que,

$$\frac{1}{(2^N - 1)T_b} \approx 0,$$

$$NT_b \leq t_R.$$

O valor que consegue obedecer essas expressões é $N = 10$, pois

$$\frac{1}{(2^{10} - 1)0,25} \approx 0,004,$$

$$10 \cdot 0,25 \leq t_R.$$

Como visto na Seção 3.2.1 o tempo de resposta (t_R) do sistema é de aproximadamente 1,1s, fazendo com que a escolha de $N = 10$ e $Tb = 0,25$ satisfaça as duas equações propostas.

O próximo passo para projetar o sinal *PRBS* foi definir a sua magnitude. Foram

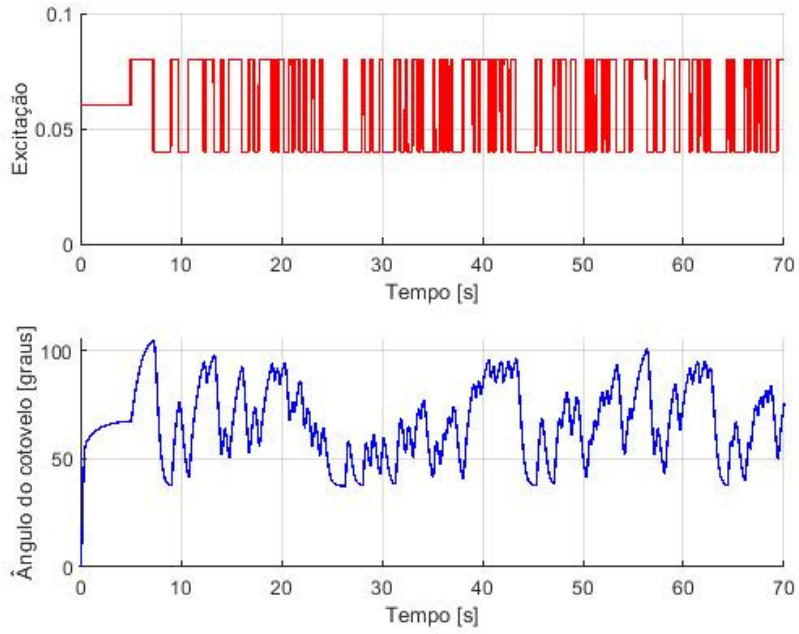
feitos testes para definir qual variação da entrada gerava respostas dentro de uma faixa desejada. Por conta disso uma variação pequena de 0,02 foi escolhida para atuar dentro do valor de 0,06 de magnitude colocado anteriormente na entrada em degrau, fazendo a entrada variar em uma magnitude de 0,04 à 0,08 com a frequência estipulada pelo *PRBS*.

Por fim, para iniciar a entrada *PRBS* quando o sistema já estivesse em regime permanente na faixa de análise desejada, o sinal só foi adicionado 5 segundos após uma entrada em degrau de 0,06 de magnitude.

Esses parâmetros foram colocados em um bloco de simulação dentro do *Simulink*, e depois passados para o *MATLAB* para posteriormente serem simulados como entrada dentro do *Opensim*.

A Figura 37 mostra o sinal de entrada *PRBS* e a resposta do sistema que foi obtida a partir desse.

Figura 37 – Resposta do sistema a entrada PRBS nos músculos de flexão



Fonte: Próprio Autor.

O próximo passo é a identificação de um modelo ARX para o sistema de flexão do braço.

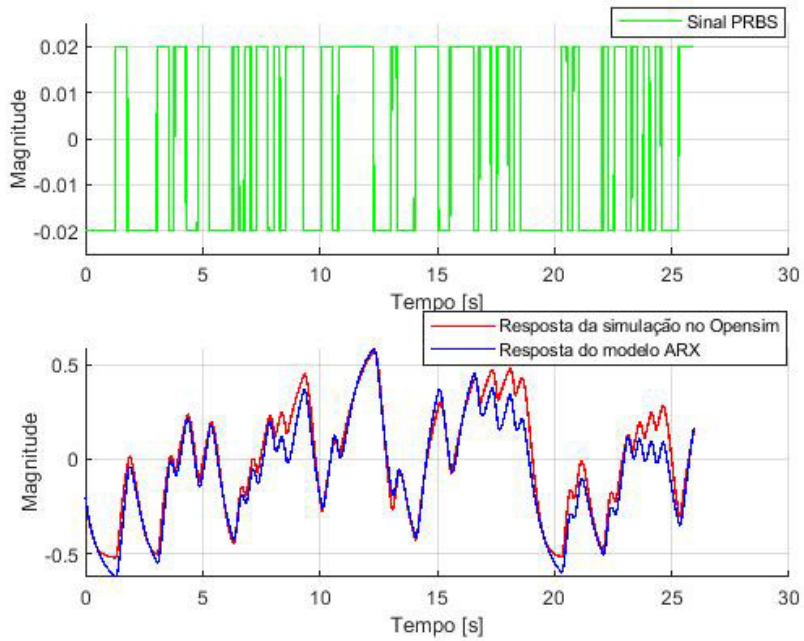
3.2.3 Identificação do modelo ARX para a flexão do cotovelo

A metodologia usada na identificação do modelo ARX da extensão também foi usada para a flexão. Simulando o sistema com 60% de sua entrada PRBS a resposta a planta de estudo se assemelhou a seguinte função de transferência no tempo discreto em malha aberta:

$$\frac{y(k)}{u(k)} = \frac{0,0318q^{-1} + 0,1187q^{-2}}{1 - 1,8195q^{-1} + 0,8239q^{-2}}. \tag{3.8}$$

Com os parâmetros da planta obtidos, os outros 40% dos dados de entrada e saída foram utilizados para serem aplicados no modelo e comparar os valores obtidos com os valores reais da simulação. A Figura 38 mostra a comparação entre as respostas obtidas aplicando-se o sinal PRBS no modelo *ARX* e a resposta real obtida por meio da simulação no *Opensim*.

Figura 38 – Comparação entre a saída real e a saída estimada da flexão



Fonte: Próprio Autor.

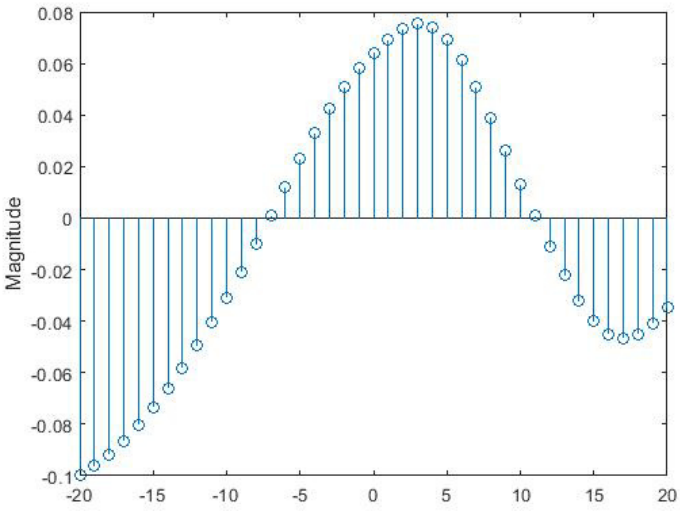
Podemos ver que visualmente os valores estimados pelo modelo *ARX* estão próximos dos valores reais, porém uma análise visual não é suficiente para que se possa avaliar o desempenho.

Utilizando a *Correlação Múltipla* como índice de desempenho do sistema, obtemos um valor de 0,9166 para a mesma. Esse valor se encontra dentro do intervalo de 0,9 à 1 visto na Seção 2.3.2.3.

As outras formas de avaliação da identificação do sistema foram a correlação cruzada entre resíduo e entrada e a auto correlação do resíduo. Vamos utilizar o mesmo critério da Equação 2.20 usado na análise da extensão. Como o número de amostras desse estudo é o mesmo, o valor limite que os índices devem atingir também deve ser 0,06.

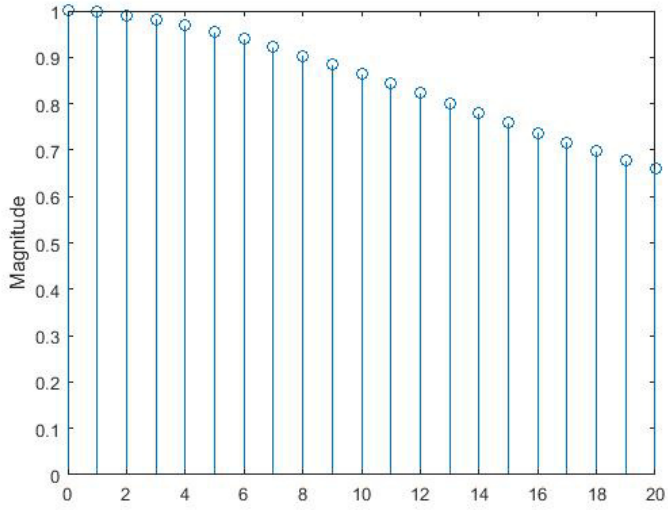
As Figuras 39 e 40 apresentam os gráficos com a correlação cruzada e a auto correlação, respectivamente.

Figura 39 – Correlação cruzada do modelo identificado de 2ª ordem para a flexão



Fonte: Próprio Autor.

Figura 40 – Autocorrelação do modelo identificado de 2ª ordem para a flexão



Fonte: Próprio Autor.

Podemos ver pelos gráficos que os índices da autocorrelação superaram muito o valor de 0,06 estabelecido anteriormente. Já o índice da correlação cruzada apresentou ultrapassagem, mas com valores próximos ao limite estabelecido, e a correlação múltipla apresentou um valor superior a 0,9. Por conta desses fatores positivos de dois índices de desempenho, esse modelo foi o escolhido para dar continuidade aos estudos.

3.2.4 Identificação dos parâmetros de controle RST

A mesma metodologia usada para encontrar os parâmetros *RST* para a extensão do braço foi usado para a flexão, assim como os parâmetros de ultrapassagem percentual e tempo de assentamento desejados. Com isso o polinômio desejado para o sistema também foi o mesmo:

$$P(q^{-1}) = 1 - 1,9868q^{-1} + 1,0791q^{-2} - 0,0890q^{-3} + 0q^{-4}. \tag{3.9}$$

Estabelecido o polinômio desejado para a função, por meio de um código no *MATLAB* foram calculados os parâmetros R, S e T, como:

$$R = 5,2645 - 9,8594q^{-1} + 4,6165q^{-2}, \tag{3.10}$$

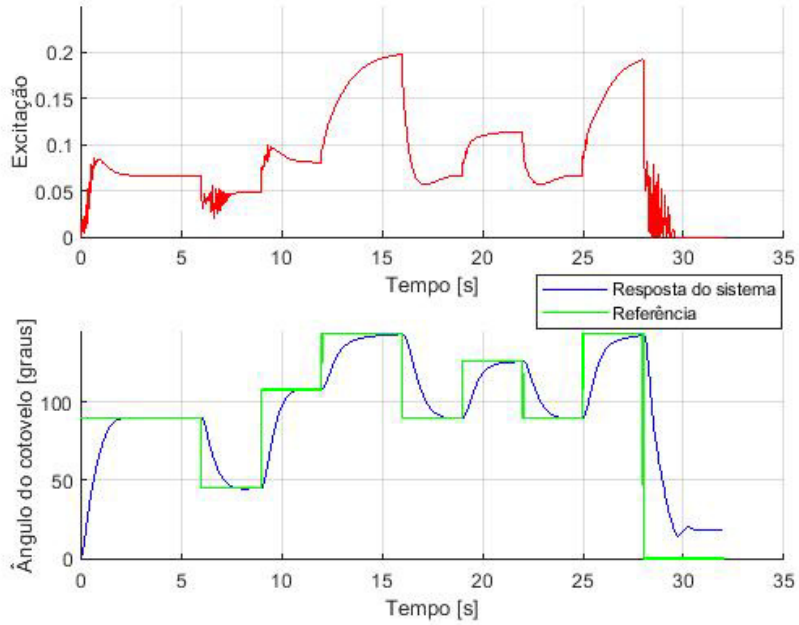
$$S = 1,0000 - 0,3346q^{-1} - 0,6654q^{-2}, \tag{3.11}$$

$$T = 0,0216. \tag{3.12}$$

Com os parâmetros R, S e T estabelecidos, um novo código para o sinal de controle enviado ao *Opensim* foi elaborado no *Matlab*. Foi projetado para que ao longo da simulação o valor de referência para o ângulo do cotovelo fosse mudando em diferentes magnitudes. A resposta em malha fechada do controlador RST atuando nos músculos de flexão do braço pode ser visto na Figura 41.

Podemos ver que a resposta em malha fechada do controlador está seguindo a referência sem ultrapassagem e com tempo de assentamento próximo de 1 segundo, parâmetros estabelecidos anteriormente. A referência só não é seguida quando ela passa a valer zero. Esse comportamento já era esperado do controlador, pois com o ângulo do ombro estabelecido como zero o trabalho de flexão só poderia ser feito para ângulos maiores do que 18,5°, para ângulos

Figura 41 – Sinal de controle e a resposta dos músculos de flexão



Fonte: Próprio Autor.

menores do que esse valor o trabalho que deve ser feito é o de extensão. Como os músculos de flexão têm esse limite de atuação, quando a referência passa para um valor menor a esse limite o ângulo tende a se estabilizar nesse valor e com uma leve oscilação.

Podemos então ver pela simulação que com o controle somente dos músculos encarregados da flexão do cotovelo, não conseguimos cobrir todos os limites de ângulos que existem para o movimento, sendo necessário estabelecer também o controle da extensão do mesmo.

3.3 Controle simultâneo da flexão e da extensão

Com as simulações descritas nas seções anteriores viu-se a necessidade de se implementar um sistema que faça o controle da flexão e da extensão ao mesmo tempo. Nesse presente trabalho os dois controladores *RST* identificados para a flexão e extensão foram colocados em simulação, funcionando de maneira independente entre si, e os resultados foram avaliados.

Para uma melhor análise desses resultados, foram feitas simulações com três diferentes posições fixas para a flexão do ombro. E para cada posição foram feitas duas simulações, uma na qual a referência variava entre valores próximos aos valores em que os modelos *ARX* foram identificados, e outra na qual a referência variava em valores mais extremos de angulação. Essas duas simulações servem para vermos se o controlador *RST* responde bem a não linearidade do

sistema identificado.

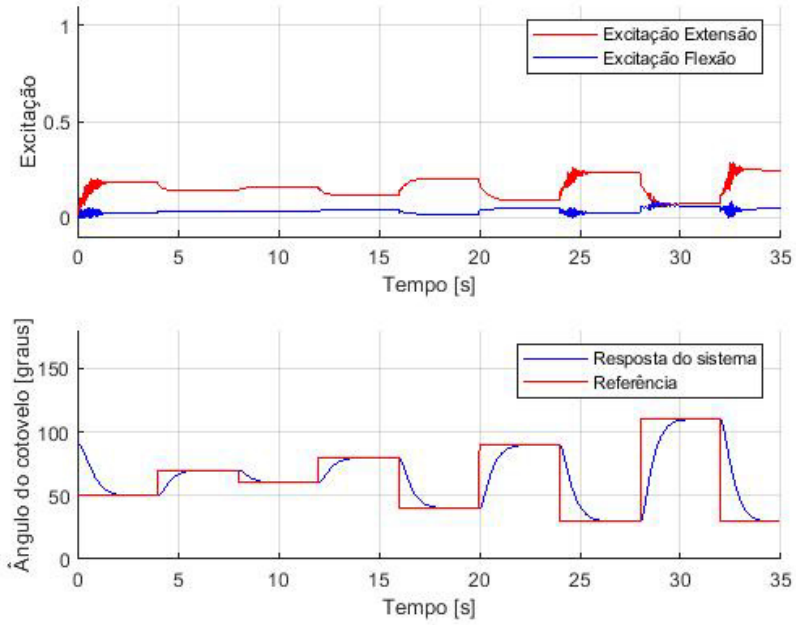
3.3.1 Controle com ombro flexionado à -90º

A primeira posição escolhida foi a posição em que a flexão do ombro se encontra fixa em um valor de -90°. Essa posição já foi vista na Figura 20.

Nessa posição os músculos de extensão do cotovelo são os protagonistas nos movimentos referentes a angulações inferiores a 95,8°, já os músculos de flexão são os protagonistas nos movimentos superiores a esse ângulo.

A resposta a primeira simulação, com a referência variando em valores próximo aos usados na identificação dos modelos *ARX*, pode ser visto na Figura 42.

Figura 42 – Resposta do sistema com ombro em -90° e referência com média amplitude de variação

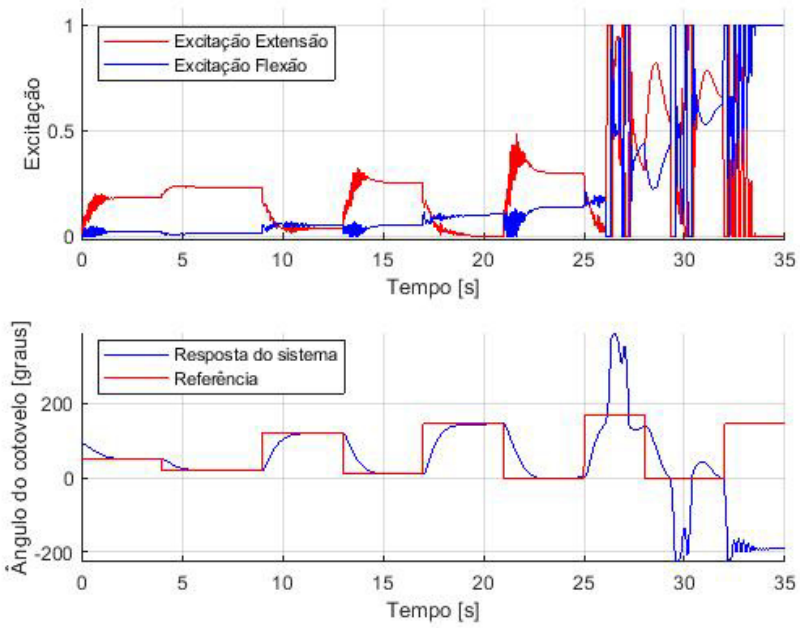


Fonte: Próprio Autor.

Nessa simulação podemos ver que o modelo atendeu bem as especificações projetadas para o controlador, sem ultrapassagem e com tempo de resposta próximo de 1 segundo. Vemos também que referências superiores e inferiores a 95,8° foram seguidas, mostrando que o sistema de controle conseguiu fazer os movimentos de flexão e extensão do cotovelo.

O passo seguinte foi fazer a simulação onde a referência varia em valores próximos aos limites de angulação do cotovelo, para analisar o desempenho do sistema de controle a não linearidade do mesmo. Essa segunda simulação pode ser vista na Figura 43.

Figura 43 – Resposta do sistema com ombro em -90° e referência com alta amplitude de variação



Fonte: Próprio Autor.

Nessa segunda simulação conseguimos observar não conformidades com o que foi projetado e também limitações do uso do modelo *Arm*26. Em movimentos reais não é possível obter, com a excitação dos músculos, ângulos para o cotovelo superiores à 180° e muito inferiores à 0°, nossas articulações e braço impedem que esses movimentos sejam feitos. Contudo podemos ver pela resposta da simulação que o modelo utilizado no *software Opensim* não apresenta esses limites de movimentação.

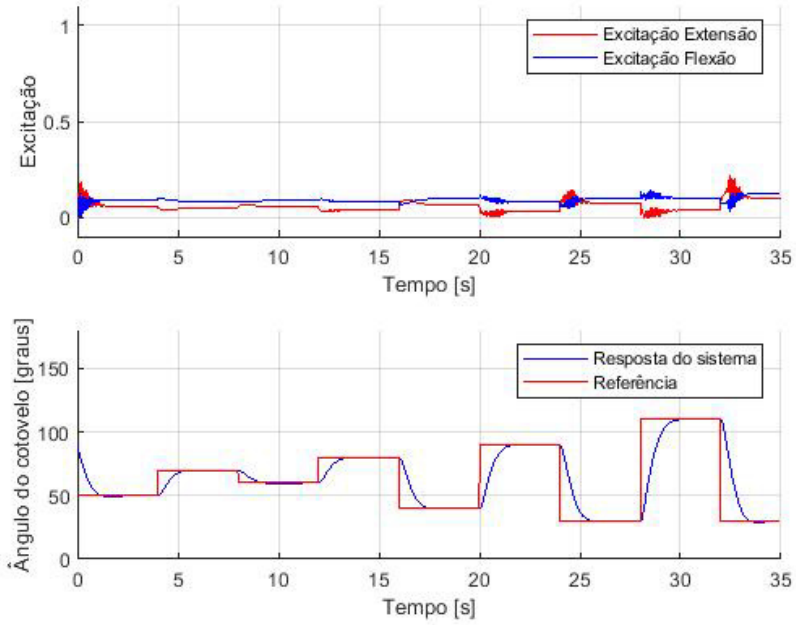
Quanto as não conformidades do controlador, podemos ver que o mesmo se comportou bem para a faixa de limite inferior de referência, conseguindo atingi-la quando está se encontra em 0°, como pode ser visto na Figura 43 dos 21 segundos aos 25 segundos. Agora para o limite superior de referência o sistema de controle não se comportou da maneira projetada. Mais simulações foram feitas e foi visto que o controlador não consegue seguir a referência para valores desta a partir de 145°, não conseguindo retornar para a referência posteriormente, mesmo quanto esta volta para valores inferiores a 145°.

3.3.2 Controle com ombro flexionado à 0º

A posição seguinte a ser simulada foi a posição onde o ângulo de flexão do ombro se encontra fixo em 0°. Essa posição já foi vista anteriormente na Figura 33.

Nessa posição a força responsável pela extensão do cotovelo é a protagonista nos movimentos referente a angulações inferiores a 18,5°, já a flexão é protagonista nos movimentos superiores a esse ângulo. A resposta a simulação pode ser vista na Figura 44.

Figura 44 – Resposta do sistema com ombro em 0° e referência com média amplitude de variação



Fonte: Próprio Autor.

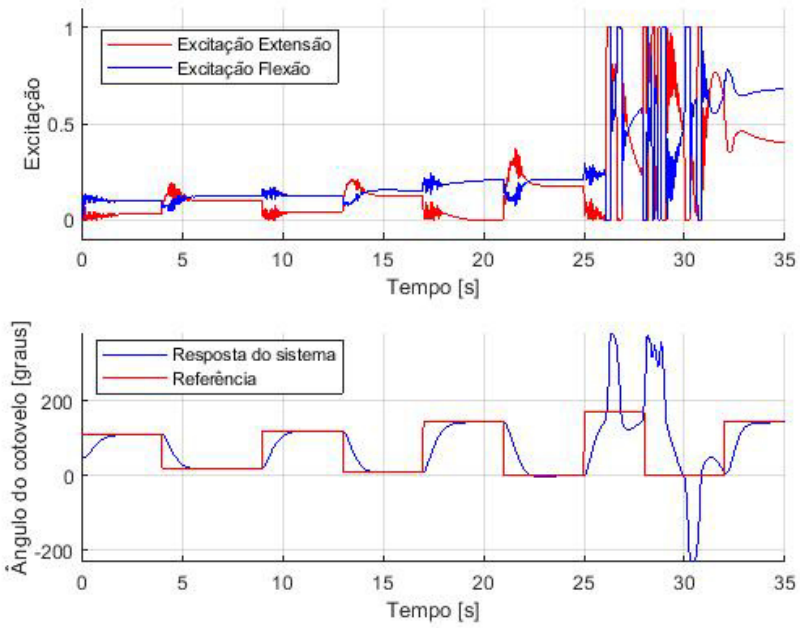
Na posição do ombro em 0°, grande parte dos movimentos serão feitos pelas forças responsáveis pela flexão, e as variações de referência feitas na simulação da Figura 44 foram dentro desses ângulos. Podemos constatar isso quando vemos que a ativação dos músculos de flexão se encontra mais alta que a ativação dos músculos de extensão em grande parte da simulação, mesmo os músculos de flexão tendo mais fibras e precisando de menos ativação.

Podemos ver que o sistema de controle se comporta conforme os parâmetros estabelecidos para o mesmo. Somente em alguns pontos podemos ver uma pequena ultrapassagem, que pode ser devido a adição da ação de extensão, pois essas ultrapassagens só são observadas na simulação quando ocorre uma transição para um ângulo menor, momento onde uma ação de extensão é exigida.

O passo seguinte foi fazer a simulação onde a referência varia em valores próximos aos limites de angulação do cotovelo. Essa segunda simulação pode ser vista na Figura 45.

Nessa simulação podemos ver um comportamento muito semelhante ao observado na simulação do ombro à -90°. Podemos ver que o sistema de controle consegue atender o

Figura 45 – Resposta do sistema com ombro em 0° e referência com alta amplitude de variação



Fonte: Próprio Autor.

limite inferior de referência, de 0°, mas o limite superior, ângulos maiores que 145°, não são seguidos. Contudo também podemos notar uma pequena diferença, depois de algumas mudanças de referência o sistema consegue se estabilizar novamente, seguindo a referência estabelecida no final.

3.3.3 Controle com ombro flexionado à 180º

A posição seguinte a ser simulada foi a posição onde o ângulo de flexão do ombro se encontra fixo em 180°. Essa posição já foi vista anteriormente na Figura 18.

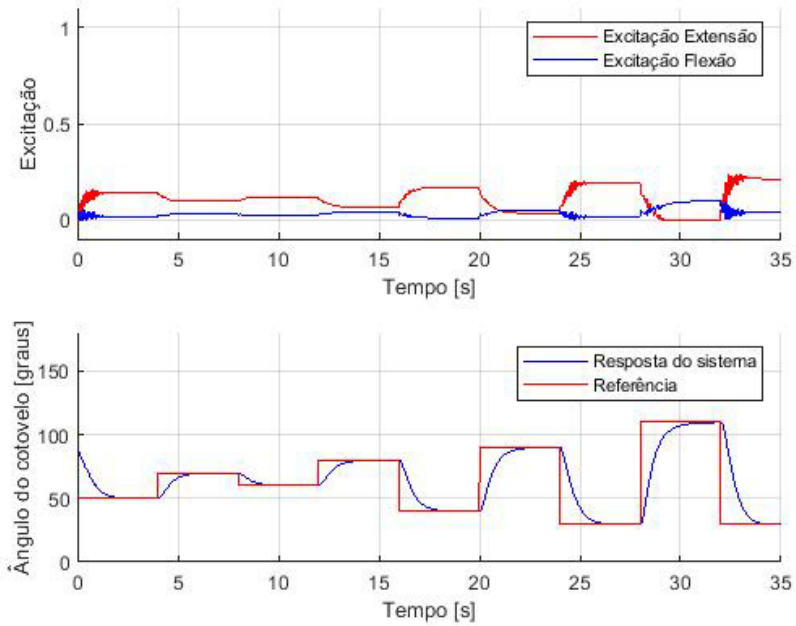
A resposta a primeira simulação, com a referência variando em valores próximo aos usados na identificação dos modelos *ARX*, pode ser visto na Figura 46.

Nessa simulação podemos ver que o modelo atendeu bem as especificações projetadas para o controlador, sem ultrapassagem e com tempo de resposta próximo de 1 segundo.

O passo seguinte foi fazer a simulação onde a referência varia em valores próximos aos limites de angulação do cotovelo. Essa segunda simulação pode ser vista na Figura 47.

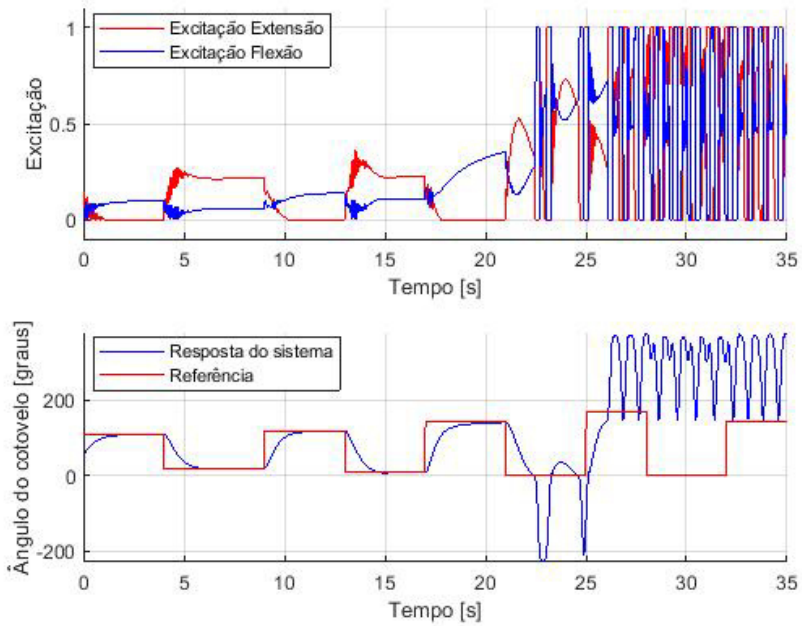
Podemos ver pela simulação que nessa posição o sistema de controle não conseguiu seguir a referência para o limite inferior da mesma, de 0°, diferentemente das simulações feitas anteriormente. Analisando melhor a posição de 180° do ombro podemos ver o motivo de isso

Figura 46 – Resposta do sistema com ombro em 180° e referência com média amplitude de variação



Fonte: Próprio Autor.

Figura 47 – Resposta do sistema com ombro em 180° e referência com alta amplitude de variação



Fonte: Próprio Autor.

ocorrer. Nessa posição quando o cotovelo atinge valores menores do que 0° a gravidade passa a atuar a favor da diminuição do ângulo do cotovelo, essa ação da gravidade puxa o braço para baixo de maneira mais expressiva do que a ação de flexão imposta pelo controlador. Contudo esse problema ocorre somente na simulação, pois em um braço real a ação da articulação do

cotovelo impede que o ângulo do mesmo atinja valores muito negativos.

Podemos ver pela simulação que a limitação da referência superior também ocorre para essa posição, com a referência não podendo atingir valores superiores a 145°.

4 CONCLUSÕES E TRABALHOS FUTUROS

Esse trabalho visou a identificação de modelos discretos que descrevam a ação dos músculos de flexão e extensão do cotovelo a partir de experimentos feitos dentro do *software Opensim*, sendo um modelo para a flexão e outro para a extensão. Como o *Opensim* não apresenta ferramentas para se realizar o controle em malha fechada, fez-se necessária o uso do *software MATLAB* junto à *API* do *Opensim*.

A partir dos modelos identificados foram projetados dois controladores RST, que foram primeiro avaliados quanto ao seu desempenho individual, depois quanto ao seu desempenho em conjunto.

O desempenho individual dos controladores para os músculos da flexão e da extensão obtiveram resultados satisfatórios, com a referência sendo seguida quando essa apresentava uma média amplitude de variação ao longo da simulação. Contudo, com o desempenho individual, viu-se que algumas referências não conseguiam ser seguidas, mas esse comportamento já era esperado, tendo em vista que o movimento completo do cotovelo não pode ser atingido somente com uma ação muscular, de flexão ou extensão.

Visto isso, os dois controladores foram simulados em conjunto, um operando em paralelo em relação ao outro, sendo feitas simulações com diferentes posições do braço, para validar a assertividade do controlador mesmo em posições diferentes da qual o modelo foi identificado. Com uma média amplitude de variação da referência foi possível observar que o controlador obteve um ótimo desempenho, tendo a resposta seguindo os parâmetros desejados para o mesmo.

O passo seguinte foi realizar simulações com alta amplitude de variação da referência. Nessas simulações foram observados limites para o uso do controlador, o mesmo não conseguindo seguir a referência quanto esta passa a valer 0° ou valores maiores que 145°. Contudo o controlador não consegue seguir a referência de 0° por limitações do modelo utilizado no *Opensim*, que não conta com a articulação do cotovelo, que impede o mesmo de atingir valores muito negativo. Em testes com modelos reais esse problema não ocorreria e a referência conseguiria ser seguida. Para o problema do não alcance de referências maiores que 145° são necessários maiores testes para se determinar se o mesmo é um problema do controlador projetado, devido a não linearidade do sistema, ou também é um problema do modelo utilizado no *software*.

O uso de dois controladores de maneira independente pode apresentar problemas

de eficiência do projeto, pois, como as ações de flexão e extensão são antagônicas, uma mesma referência pode ser atingida com níveis diferentes de excitação dos músculos, podendo esses níveis estarem acima do ideal e ocasionarem, na prática, problemas como a fadiga muscular. Para trabalhos futuros é então necessário o estudo do projeto de um único controlador que consiga realizar as ações de flexão e extensão.

Posteriormente também pode ser feito o estudo de como realizar esse tipo de controle em um experimento real, aliando o controlador RST à técnica FES. Para isso deve-se entender quais músculos podem ser estimulados utilizando a técnica, e realizar a identificação do modelo de flexão e extensão a partir desses músculos.

Na prática também pode ser necessário excitações distintas para cada músculo durante um movimento. Como no presente trabalho os 3 músculos de flexão e os 3 de extensão receberam o mesmo sinal de controle, um projeto de controlador com diferentes sinais de controle para cada músculo pode ser feito.

Por fim, o modelo *Arm*26, utilizado nesse estudo, pode apresentar limitações e não conformidades com modelos reais, com isso o modelo identificado para a planta pode não ser assertivo para experiências práticas de controle do braço. Caso isso acontece, deve ser estudada a possibilidade da identificação de modelos de flexão e extensão com base em experiência reais.

Podemos ver que o presente estudo ainda necessita de muitas adaptações para que possa ser utilizado fora de uma simulação, contudo, o potencial de impacto do mesmo serve de motivação para a sua continuidade.

REFERÊNCIAS

ASTROM, K. J.; WITTENMARK, B. **Adaptive Control**. 2. ed. [*S. l.*]: Pearson, 1994.

ASTROM, K. J.; WITTENMARK, B. **Computer-Controlled Systems**: Theory and design. 3. ed. [*S. l.*]: Prentice Hall, 1997.

Bó, A. P. L.; FONSECA, L. O. da; SOUSA, A. C. C. de. Fes-induced co-activation of antagonist muscles for upper limb control and disturbance rejection. **Medical Engineering and Physics**, v. 38, p. 1176–1184, 2016.

Bó, A. P. L.; MOURA, H. C. Elbow control using functional electrical stimulation: an experimental comparison of different control strategies. **IFAC-PapersOnLine**, v. 48(20), p. 343–347, 2015.

CHIN, C. M.; MILOS, R. P. Functional electrical stimulation therapy for restoration of motor function after spinal cord injury and stroke: a review. **BioMed Eng OnLine**, p. 19–34, 2020.

COELHO, A. A. R.; COELHO, L. d. S. **Identificação de Sistemas Dinâmicos Lineares**. [*S. l.*]: Editora da Universidade Federal de Santa Catarina, 2004.

DELP, S. L.; ANDERSON, F. C.; ARNOLD, A. S.; LOAN, P.; HABIB, A.; JOHN, C. T.; GUENDELMAN, E.; THELEN, D. G. Opensim: opensource software to create and analyze dynamic simulations of movement. **IEEE transactions on biomedical engineering**, v. 54, n. 11, p. 1940–1950, 2007.

DORF, R. C.; BISHOP, R. H. **Sistemas de Controle Modernos**. [*S. l.*]: LTC Editora, 2001.

FRANKLIN, G. F.; POWELL, J. D.; WORKMAN, M. L. **Digital Control of Dynamic Systems**. 3. ed. [*S. l.*]: Addison Wesley Longman. Inc., 1998.

FREIVALDS, A. **Biomechanics of The Upper Limbs**: Mechanics, modeling, and musculoskeletal injuries. [*S. l.*]: CRC Press, 2004.

HAMILL, J.; KNUTZEN, K. M.; DERRICK, T. R. **Bases biomecânicas do movimento humano**. 4. ed. [*S. l.*]: Manole, 2015.

HSIA, T. C. **System Identification**. [*S. l.*]: Lexington Books, 1977.

KEESMAN, K. J. **System Identification**: An introduction. [*S. l.*]: Springer, 2011.

KOZAN, R. F. **Controle da Posição da Perna de Pessoas Hígidas Utilizando um Controlador PID**. 96 p. Dissertação (Mestrado em Engenharia Elétrica), São Paulo, 2012.

LANDAU, I. D.; ZITO, G. **Digital Control Systems**: Design, identification and implementation. London: Springer, 2006.

LANDOU, I. D.; LOZANO, R.; M'SAAD, M.; KARIMI, A. **Adaptive Control**: Algorithms, analysis and applications. 2. ed. [*S. l.*]: Springer, 2011.

LJUNG, L. **System Identification**: Theory for the user. [*S. l.*]: Prentice Hall, 1987.

PAZ, P. L.; OLIVEIRA, T. R.; PINO, A. V.; FONTAN, A. P. Model-free neuromuscular electrical stimulation by stochastic extremum seeking. **IEEE transactions on control systems technology**, v. 28, n. 1, p. 238–253, 2020.

PINHEIRO, W. C. **Modelo Biomecânico de Punho para a Simulação Computacional de Tremores Patológicos em Malha Fechada Utilizando Controle** H_∞. 134 p. Dissertação (Mestrado em Engenharia Biomédica), Rio de Janeiro, 2019.

SANCHES, M. A. A. **Sistema Eletrônico para Geração e Avaliação de Movimentos em Paraplégicos**. 186 p. Dissertação (Doutorado em Engenharia Elétrica), São Paulo, 2013.

SEBORG, D. E.; EDGAR, T. F.; MELLICHAMP, D. A. **Process Dynamics and Control**. 2. ed. New York: John Wiley and. Sons, 2004.

SOUSA, A. C. C. de; SOUSA, F. S. C.; Bó, A. P. L. Simulation of the assistance of passive knee orthoses in fes cycling. **In 2019 41st Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)**, p. 3811–3814, 2019.

APÊNDICE A – CÓDIGO INICIAL DA ENTRADA DEGRAU NO MATLAB

Código-fonte 1 – Implementação da entrada degrau no MATLAB para ser inserida no Opensim

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script adaptado do trabalho de Ana de Sousa (anacsousa@lara.unb.br),
% Felipe Shimabuko,
% e Antonio Padilha L. Bo

% Para a utilizacao do MATLAB junto ao opensim eh necessario seguir
% os passos presentes em:
% https://simtk-confluence.stanford.edu:8443/display/OpenSim/Scripting+with+Matlab

% Para que a simulacao possa ser feita eh necessaria a presenca
% dos arquivos 'IntegrateOpenSimPlant.m' e 'OpenSimPlantFunction.m'
% presentes em:
% https://simtk-confluence.stanford.edu:8443/pages/viewpage.action?pageId=28777060


clear; tic; close all; clc;

%acao obrigatoria, ela importa a biblioteca do opensim
import org.opensim.modeling.*

% criacao das variaveis que serao utilizadas
global Tf
global Ts
```

```
24 global n_samples
25 global controlTime
26 global controlIteration
27 global excitation
28 global elbowangle
29 global excitationplot
30 global elbowangleplot
31
32
33 %% CONFIGURACAO INICIAL
34 % Escolha do tempo de simulacao e de amostragem
35
36 % tempo de simulacao
37 Tf = 3;
38
39 % tempo de amostragem e quantidade de amostras
40 Ts = 1/50; n_samples = floor(Tf/Ts);
41
42 % vetor que vai captar o tempo de cada amostra
43 controlTime = zeros(1,n_samples);
44
45 %variavel que informa em qual amostra esta a simulacao
46 controlIteration = 1;
47
48 %vetor de ativacao dos musculos
49 excitation=ones(1,n_samples);
50
51 %vetor que armazena os valores para o angulo do cotovelo
52 elbowangle= zeros(1,n_samples);
53
54
55 %% INICIALIZACAO DO MODELO
```

```
56 disp('> Initialize simulation')
57
58 % atribui o modelo do opensim a variavel 'model'
59 model = Model('arm26.osim');
60
61 %%%%%%%
62 % codigos para setar a ativacao inicial dos musculos em
      zero
63 % eh utilizado pois pode existir no modelo alguma ativacao
      inicial
64
65 % capta a variavel forca 'BIClong' (biceps cabeca longa)
66 actuator1 = model.updForceSet().get('BIClong');
67
68 % transfere a variavel de forca para uma variavel de
      musculo
69 % esse passo eh necessario pois nao eh possivel pegar uma
70 % variavel musculo diretamente do opensim utilizando o
      matlab,
71 % sendo necessario pegar a variavel como forca e depois
      converte-la
72 % para musculo
73 biclong = Thelen2003Muscle.safeDownCast(actuator1);
74
75 % coloca ativacao inicial de zero no biceps cabeca longa
76 biclong.setDefaultActivation(0);
77
78 % repete o mesmo processo para os outros musculos
79
80 % biceps cabeca curta
81 actuator2 = model.updForceSet().get('BICshort');
82 bicshort = Thelen2003Muscle.safeDownCast(actuator2);
```

```
83  bicshort.setDefaultActivation(0);
84
85  % triceps cabeca longa
86  actuator3 = model.updForceSet().get('TRIlong');
87  trilong = Thelen2003Muscle.safeDownCast(actuator3);
88  trilong.setDefaultActivation(0);
89
90  % triceps cabeca lateral
91  actuator4 = model.updForceSet().get('TRIlat');
92  trilat = Thelen2003Muscle.safeDownCast(actuator4);
93  trilat.setDefaultActivation(0);
94
95  % triceps cabeca medial
96  actuator5 = model.updForceSet().get('TRImed');
97  trimed = Thelen2003Muscle.safeDownCast(actuator5);
98  trimed.setDefaultActivation(0);
99
100 % braquial
101 actuator6 = model.updForceSet().get('BRA');
102 bra = Thelen2003Muscle.safeDownCast(actuator6);
103 bra.setDefaultActivation(0);
104
105 %%%%%%%%
106
107 % coloca a ativacao para ter magnitude de 0.2
108 excitation=0.2*excitation;
109
110 % pega a variavel da coordenada do angulo do cotovelo
111 valorangle =model.getCoordinateSet().get( "r_elbow_flex" );
112
113 % atribui para iniciar no angulo de 95.8 graus, em radianos
114 valorangle.setDefaultValue(95.8*pi/180);
```

```
115
116 %coloca esse primeiro valor no vetor do angulo do cotovelo
117 elbowangle(1)  = valorangle.getDefaultValue();
118
119 % atribui o angulo de -90 graus, em radianos, para a
      coordenada do angulo
120 % do ombro
121 editableCoordSet = model.updCoordinateSet().get( "
      r_shoulder_elev" );
122 editableCoordSet.setDefaultValue(-90*pi/180);
123
124 % coloca essa coordenada para se manter fixa nesse valor
125 editableCoordSet.setDefaultLocked(true);
126
127 %%%%%%%%
128
129 % procedimento para se iniciar a simulacao
130 states = model.initSystem();
131 model.equilibrateMuscles(states);
132
133 % chama a funcao de controle
134 controlFunctionHandle = @degrau_control;
135
136 %%%%%%%%%%
137
138 % pega o nome de todas as variaveis(estados)do modelo
139 % eh necessario para se gerar o arquivo que possibilita ver
      o movimento no
140 % opensim
141     states_vector = model.getStateVariableNames();
142     n = states_vector.getSize;
143     if n < 1
```

```
144         disp('Issue getting states names from model!');
145     end
146 
147     statesNames = cell(1, n);
148     for i = 1:n
149         state = char(states_vector.get(i-1));
150         statesNames{i} = state;
151     end
152 %%%%%%%
153 
154 %% SIMULACAO
155 disp('> Run simulation')
156 
157 %%%%%%%
158 % codigo padrao para realizar a integracao
159 timeSpan = [0 Tf]; integratorName = 'ode15s';
160 integratorOptions = odeset('AbsTol', 1E-5','MaxStep',.1*Ts)
    ;
161 
162 % comeca a simulacao, chamando a funcao de controle a cada
    intervalo de
163 % simulacao do opensim
164 motionData = IntegrateOpenSimPlant(model,
    controlFunctionHandle, timeSpan, ...
165     integratorName, integratorOptions);
166 
167 
168 % alguns modelos apresentam ordem errada entre a posicao do
     nome de suas
169 % variaveis e os dados delas, esse codigo organiza esses
    dados
```

```
170 motionData.data(:,[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
    17])=motionData.data(:,[1 2 4 3 5 6 7 8 9 10 11 12 13 14
     15 16 17]);
171
172
173 % algumas simulacoes podem terminar antes da quantidade de
    amostras
174 % previstas
175 % e deve-se filtrar os vetores ate onde foram simulados
176 excitationplot=excitation(1:controlIteration);
177 elbowangleplot=elbowangle(1:controlIteration);
178 controlTimeplot=controlTime(1:controlIteration);
179
180
181 % roda o codigo feito para a plotagem dos dados
182 degrau_plot;
183
184 %% CRIAR ARQUIVOS DE MOVIMENTO (.STO FILE) PARA
    VISUALIZACAO
185 %% DENTRO DO OPENSIM
186     Nsamples = length(motionData.data(:,1)); Nstates =
        length(statesNames);
187
188     str_name = 'degrau_';
189
190     str = strjoin(statesNames,'\t');
191     header = ['degrau_simulation \nversion=1 \nnRows='
        num2str(Nsamples) ' \nnColumns=' num2str(Nstates+1)
        '\ninDegrees=no \nendheader \ntime ' str '\n'];
192
193     fid = fopen(strcat('Results/',str_name,'.sto'),'wt');
194     fprintf(fid,header); fclose(fid);
```

```
195
196     fid = fopen(strcat('Results/',str_name,'.sto'),'a+');
197     for i = 1:Nsamples
198         fprintf(fid,'\t%f',motionData.data(i,:)); fprintf(
                fid,'\n');
199     end
200     fclose(fid);
201
202 %% FINAL DA SIMULACAO
203 fprintf('\n'); toc; disp('> THE END')
```

APÊNDICE B – FUNÇÃO DE CONTROLE PARA A ENTRADA DEGRAU NO MATLAB

Código-fonte 2 – Implementação da função de controle da entrada degrau

```
1
2  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
3  % Script adaptado do trabalho de Ana de Sousa (
       anacsousa@lara.unb.br),
4  % Felipe Shimabuko,
5  % e Antonio Padilha L. Bo
6
7  function modelControls = degrau_control(osimModel,
       osimState)
8
9      import org.opensim.modeling.*;
10
11     global Ts
12     global n_samples
13     global controlTime
14     global controlIteration
15     global excitation
16     global elbowangle
17
18
19
20     %% PEGA AS INFORMACOES DA SIMULACAO NO TEMPO ATUAL
21     modelControls = osimModel.updControls(osimState);
22     thisTime = osimState.getTime();
23
24
25     %% CODIGO QUE ATUALIZA O SISTEMA DE CONTROLE
```

```
26      % SOMENTE NO TEMPO DE AMOSTRAGEM ESTABELECIDO
           ANTERIORMENTE
27      if (thisTime - controlTime(controlIteration)) >= (Ts
           -.01*Ts)
28
29          % atualiza o vetor de tempo e em qual amostra a
               simulacao esta
30          controlIteration = controlIteration + 1;
31          controlTime(controlIteration) = thisTime;
32
33          % capta o valor atual do angulo do cotovelo
34          valorangle =osimModel.getCoordinateSet().get( "
               r_elbow_flex" );
35
36
37          % coloca esse valor no vetor do angulo
38          elbowangle(controlIteration)  = valorangle.getValue
               (osimState);
39
40          % printa informacoes da simulacao durante a mesma
41          fprintf('%d/%d: excitacao: %f , angulo: %f\n',
               controlIteration, n_samples, excitation(
               controlIteration), elbowangle(controlIteration))
               ;
42
43      end
44
45  %% ADICAO DO SINAL DE CONTROLE NOS MUSCULOS
46
47  % adiciona o sinal de controle somente nos musculos de
       extensao
48
```

```
49 osimModel.updActuators().get('BRA').addInControls(Vector
       (1,0),modelControls);
50
51 osimModel.updActuators().get('BIClong').addInControls(
       Vector(1,0),modelControls);
52
53 osimModel.updActuators().get('BICshort').addInControls(
       Vector(1,0),modelControls);
54
55 osimModel.updActuators().get('TRIlat').addInControls(Vector
       (1,excitation(controlIteration)),modelControls);
56
57 osimModel.updActuators().get('TRImed').addInControls(Vector
       (1,excitation(controlIteration)),modelControls);
58
59 osimModel.updActuators().get('TRIlong').addInControls(
       Vector(1,excitation(controlIteration)),modelControls);
60
61
62 end
```

APÊNDICE C – CÓDIGO PARA A IDENTIFICAÇÃO DO MODELO ARX NO MATLAB

Código-fonte 3 – Identificação do modelo ARX do sistema

```
% Coleta os sinais de entrada PRBS e saida do sistema
% a partir dos 5 segundos, e tira o sinal continuo dos
   mesmos
y = elbowangleplot(252:end)-elbowangleplot(251);
u = excitationplot(252:end)-excitationplot(251);

% transpoe as matrizes
y=y';
u=u';

% Dados para identificacao (60 por cento dos dados sao util
   . p/ identificar)
yi = y(1:floor(length(y)*0.6));
ui = u(1:floor(length(y)*0.6));

% Dados para validacao (40 por cento dos dados sao util. p/
    validar)
yv = y(floor(length(y)*0.6)+1:end);
uv = u(floor(length(y)*0.6)+1:end);

% Definicao da ordem do modelo
nb = 2;
na = 2;
d = 0;

% Identificacao do modelo
N=length(yi);

```

```
26 ci=max(nb+d,na);
27
28 phi=zeros(N-ci,nb+na);
29
30 for i=ci+1:N
31     phi(i-ci,:) = [-(yi(i-1:-1:i-na))' ui(i-1-d:-1:i-nb-d)
       ']';
32 end
33
34 if det(phi'*phi)==0
35     error('A matriz phiT*phi    singular','A');
36 end
37
38 teta = inv(phi'*phi)*phi'*yi(ci+1:N);
39
40 A=[1 teta(1:na,:)'];
41 B=[0 teta(na+1:na+nb,:)'];
42
43 % Validacao do Modelo
44 N = length(yv);
45 ye = zeros(N,1);
46
47 % Setar os primeiros valores
48 for i=1:ci
49    ye(i) = yv(i);
50 end
51
52 % Obtencao dos valores estimados
53 for i=ci+1:N
54     phi = [-(ye(i-1:-1:i-na))' uv(i-1-d:-1:i-nb-d)'];
55     ye(i) = phi*teta;
56 end
```

APÊNDICE D – CÓDIGO PARA A IDENTIFICAÇÃO DOS PARÂMETROS RST NO MATLAB

Código-fonte 4 – Identificação dos parâmetros RST do controlador de extensão

```
% --   Especificacao dos parametros desejados do sistema  -
% tempo de subida
Tr = 1;
% percentual de ultrapassagem
MaxOvershoot = 0.0;

% Funcao que entra com os parametros de desempenho e
   retorna
% os parametros de uma funcao de transferencia de 2 ordem
[Wo,Qsi]=omega_dmp(Tr,MaxOvershoot);

% constroi a funcao de 2 ordem continua
Hc = tf([Wo^2],[1  2*Wo*Qsi Wo^2]);

% transforma a funcao de continua para discreta
Hd = c2d(Hc,Ts,'zoh');

% extrai o numerador e denominador, sendo Pd o polinomio
   desejado
[Bd,Pd]=tfdata(Hd,'v');

% ordem dos parametros
nA=length(A)-1;
nB=length(B)-1;

% Especificacao das partes fixas desejadas do polinomio;
Hs=[1 -1];  % Hs - parte fixa de S:
Hr=1;       % Hr - parte fixa de R:
```

```
27
28 % ordem das partes fixas
29 nHs = length(Hs)-1;
30 nHr = length(Hr)-1;
31
32 % novos polinomios a' e b'
33 BB=conv(B,Hr);
34 AA=conv(A,Hs);
35
36 % ordem dos novos polinomios
37 nBB=length(BB)-1-d;
38 nAA=length(AA)-1;
39
40 % Matriz formada pelos elementos do polinomio A:
41 MA = zeros((nAA+nBB+d),(nBB+d));
42 for i=1:(nBB+d)
43     MA(i:nAA+i,i)=AA';
44 end
45
46 % Matriz formada pelos elementos do polinomio B:
47 MB = zeros((nAA+nBB+d),(nAA));
48
49 for i=1:(nAA)
50     MB(i:nBB+i+d,i)=BB';
51 end
52
53 % Matriz M:
54 MM = [MA MB];
55 %
56 % Especificacao dos polos auxiliares :
57 Pf = [poly([0.0 0.1])];
58 %Polinomio caracteristico desejado:
```

```
59 P = conv(Pd,Pf);
60 %
61 % matriz x
62 X = inv(MM)*P';
63 %
64
65 % ordem dos parametros de controle
66 nS = nBB+d-1;
67 nR = nAA-1;
68
69 % definicao dos parametros iniciais
70 So=X(1:nS+1);
71 Ro=X(nS+2:length(X));
72
73 %definicao dos parametros finais
74 R=conv(Hr,Ro)';
75 S=conv(Hs,So)';
76 T=sum(R);
77
78
79 function [omega_0,dmp]=omega_dmp(tm_req,M_req);
80
81
82 %inputs:
83 %tm_req ... required rise time in seconds
84 %M_req ... required maximum overshoot
85 %outputs:
86 %omega_0 ... natural frequency of the continues system
87 %dmp ... damping of the continues system
88 %
89 %written by: H. Prochazka, I.D. Landau
90 %7th june 2002
```

```
91
92  %damping computing
93  precision=0.1;%precision of 0.1%
94  omega_0=6;%initial value 6 rad/s to compute damping for
       overshoot
95  dmp_min=0;
96  dmp_max=1;
97  error=precision*10;%initial setting
98  while error>precision,
99    dmp_act=dmp_min+ (dmp_max-dmp_min)/2;
100   sys=tf([omega_0^2],[1 2*dmp_act*omega_0 omega_0^2]);
101   [resp,tim]=step(sys);
102   M_act=(max(resp)-1)*100;
103   if M_act<M_req,
104      dmp_max=dmp_act;
105   else
106      dmp_min=dmp_act;
107    end;
108    error=abs(M_act-M_req);
109 end;
110 dmp=dmp_act;
111
112 %natural frequency computing
113 k=1;while resp(k)<0.1, k=k+1;end;
114 t01=tim(k-1);
115 k=1;while resp(k)<0.9, k=k+1;end;
116 t09=tim(k);
117 tm_act=t09-t01;
118 omega_0=tm_act*omega_0/tm_req;
119 end
```

APÊNDICE E – CÓDIGO INICIAL PARA O CONTROLE DA EXTENSÃO E FLEXÃO NO MATLAB

Código-fonte 5 – Implementação do controle da flexão e extensão no MATLAB para ser inserida no Opensim

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script adaptado do trabalho de Ana de Sousa (
    anacsousa@lara.unb.br),
% Felipe Shimabuko,
% e Antonio Padilha L. Bo


clear; tic; close all; clc;

import org.opensim.modeling.*

global Tf
global Ts
global n_samples
global controlTime
global controlIteration
global elbowangle
global elbowangleplot
global entradaE
global entradaF
global excitationE
global excitationF
global ref
global RE
global SE
global S1E
global TE
```

```
27 global RF
28 global SF
29 global S1F
30 global TF
31
32 %% PARAMETROS DE CONTROLE
33
34 % Controlador da extensao
35 RE=[-7.108413286218948,13.223041413632808,
36 -6.150320816044573];
37 SE=[1,-0.435307599175617,-0.564692400824383];
38 TE=-0.035692688630713;
39 S1E=SE(2:end);
40
41 % Controlador da flexao
42 RF=[5.264496298010700,-9.859396532466395,
43 4.616518568118580];
44 SF=[1,-0.334623050377750,-0.665376949622250];
45 S1F=SF(2:end);
46 TF= 0.021618333662885;
47
48
49 %% CONFIGURACAO INICIAL
50
51 Tf = 35;
52
53 Ts = 1/50; n_samples = floor(Tf/Ts);
54
55 controlTime = zeros(1,n_samples);
56
57 controlIteration = 1;
58
```

```
59 elbowangle= zeros(1,n_samples);
60
61 % parametros utilizados dentro do controlador
62 entradaE=zeros(1,n_samples);
63 entradaF=zeros(1,n_samples);
64
65 % vetor de referencia
66 ref=ones(1,n_samples);
67
68 % sinal de ativacao para a extensao e para a flexao
69 excitationE=ones(1,n_samples);
70 excitationF=ones(1,n_samples);
71
72
73 %% INICIALIZACAO DO MODELO
74 disp('> Initialize simulation')
75
76 model = Model('arm26.osim');
77
78 % biceps cabeca longa
79 actuator1 = model.updForceSet().get('BIClong');
80 biclong = Thelen2003Muscle.safeDownCast(actuator1);
81 biclong.setDefaultActivation(0);
82
83 % biceps cabeca curta
84 actuator2 = model.updForceSet().get('BICshort');
85 bicshort = Thelen2003Muscle.safeDownCast(actuator2);
86 bicshort.setDefaultActivation(0);
87
88 % triceps cabeca longa
89 actuator3 = model.updForceSet().get('TRIlong');
90 trilong = Thelen2003Muscle.safeDownCast(actuator3);
```

```
91  trilong.setDefaultActivation(0);
92
93  % triceps cabeca lateral
94  actuator4 = model.updForceSet().get('TRIlat');
95  trilat = Thelen2003Muscle.safeDownCast(actuator4);
96  trilat.setDefaultActivation(0);
97
98  % triceps cabeca medial
99  actuator5 = model.updForceSet().get('TRImed');
100 trimed = Thelen2003Muscle.safeDownCast(actuator5);
101 trimed.setDefaultActivation(0);
102
103 % braquial
104 actuator6 = model.updForceSet().get('BRA');
105 bra = Thelen2003Muscle.safeDownCast(actuator6);
106 bra.setDefaultActivation(0);
107
108 %%%%%%%%
109
110 valorangle =model.getCoordinateSet().get( "r_elbow_flex" );
111 valorangle.setDefaultValue(95.8*pi/180);
112 elbowangle(1)  = valorangle.getDefaultValue();
113
114
115 editableCoordSet = model.updCoordinateSet().get( "
       r_shoulder_elev" );
116 editableCoordSet.setDefaultValue(-90*pi/180);
117 editableCoordSet.setDefaultLocked(true);
118
119 % valor inicial da referencia
120 ref=(50*pi/180)*ref;
121
```

```
122 % inicia os vetores de ativacao em zero
123 excitationE=0*excitationE;
124 excitationF=0*excitationF;
125
126
127 %%%%%%%%
128
129 states = model.initSystem();
130 model.equilibrateMuscles(states);
131
132 controlFunctionHandle = @controlador_control;
133
134 %%%%%%%%%%
135 %%
136 % pega o nome de todas as variaveis(estados)do modelo
137     states_vector = model.getStateVariableNames();
138     n = states_vector.getSize;
139     if n < 1
140         disp('Issue getting states names from model!');
141     end
142
143     statesNames = cell(1, n);
144     for i = 1:n
145         state = char(states_vector.get(i-1));
146         statesNames{i} = state;
147     end
148 %%%%%%%
149
150 %% SIMULACAO
151 disp('> Run simulation')
152
153 %%%%%%%
```

```
154 timeSpan = [0 Tf]; integratorName = 'ode15s';
155 integratorOptions = odeset('AbsTol', 1E-5','MaxStep',.1*Ts)
    ;
156
157 motionData = IntegrateOpenSimPlant(model,
    controlFunctionHandle, timeSpan, ...
158     integratorName, integratorOptions);
159
160
161 motionData.data(:,[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
    17])=motionData.data(:,[1 2 4 3 5 6 7 8 9 10 11 12 13 14
     15 16 17]);
162
163
164 excitationEplot=excitationE(1:controlIteration);
165 excitationFplot=excitationF(1:controlIteration);
166 elbowangleplot=elbowangle(1:controlIteration);
167 refplot=ref(1:controlIteration);
168 controlTimeplot=controlTime(1:controlIteration);
169
170
171 % roda o codigo feito para a plotagem dos dados
172 controlador_plot;
173
174 %% CRIAR ARQUIVOS DE MOVIMENTO (.STO FILE) PARA
    VISUALIZACAO
175 %% DENTRO DO OPENSIM
176     Nsamples = length(motionData.data(:,1)); Nstates =
        length(statesNames);
177
178     str_name = 'controlador_';
179
```

```
180     str = strjoin(statesNames,'\t');
181     header = ['controlador_simulation \nversion=1 \nnRows='
            num2str(Nsamples) ' \nnColumns=' num2str(Nstates+1)
            '\ninDegrees=no \nendheader \ntime   ' str '\n'];
182
183     fid = fopen(strcat('Results/',str_name,'.sto'),'wt');
184     fprintf(fid,header); fclose(fid);
185
186     fid = fopen(strcat('Results/',str_name,'.sto'),'a+');
187     for i = 1:Nsamples
188         fprintf(fid,'\t%f',motionData.data(i,:)); fprintf(
               fid,'\n');
189     end
190     fclose(fid);
191
192 %% FINAL DA SIMULACAO
193 fprintf('\n'); toc; disp('> THE END')
```

APÊNDICE F – FUNÇÃO DE CONTROLE PARA A FLEXÃO E EXTENSÃO NO MATLAB

Código-fonte 6 – Implementação da função de controle para a flexão e extensão do cotovelo

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Script adaptado do trabalho de Ana de Sousa (anacsousa@lara.unb.br),
% Felipe Shimabuko,
% e Antonio Padilha L. Bo

function modelControls = controlador_control(osimModel, osimState)
    import org.opensim.modeling.*;

    global Ts
    global n_samples
    global controlTime
    global controlIteration
    global elbowangle
    global excitationE
    global excitationF
    global sinalE
    global sinalF
    global entradaE
    global entradaF
    global ref
    global RE
    global S1E
    global TE
    global RF
    global S1F
    global TF
```

```
27
28 %%%%
29
30
31     %% PEGA AS INFORMACOES DA SIMULACAO NO TEMPO ATUAL
32     modelControls = osimModel.updControls(osimState);
33     thisTime = osimState.getTime();
34
35
36     %% CODIGO QUE ATUALIZA O SISTEMA DE CONTROLE
37     % SOMENTE NO TEMPO DE AMOSTRAGEM ESTABELECIDO
          ANTERIORMENTE
38     if (thisTime - controlTime(controlIteration)) >= (Ts
          -.01*Ts)
39
40         % atualiza o vetor de tempo e em qual amostra a
              simulacao esta
41         controlIteration = controlIteration + 1;
42         controlTime(controlIteration) = thisTime;
43
44         % capta o valor atual do angulo do cotovelo
45         valorangle =osimModel.getCoordinateSet().get( "
              r_elbow_flex" );
46
47         % coloca esse valor no vetor do angulo
48         elbowangle(controlIteration)  = valorangle.getValue
              (osimState);
49
50
51         %% muda a referencia durante a simulacao
52         if thisTime>4 & thisTime<8
53             ref(controlIteration)=70*pi/180;
```

```
54          end
55          if thisTime>=8 & thisTime<12
56              ref(controlIteration)=60*pi/180;
57          end
58          if thisTime>=12 & thisTime<16
59              ref(controlIteration)=80*pi/180;
60          end
61          if thisTime>=16 & thisTime<20
62              ref(controlIteration)=40*pi/180;
63          end
64          if thisTime>=20 & thisTime<24
65              ref(controlIteration)=90*pi/180;
66          end
67          if thisTime>=24 & thisTime<28
68              ref(controlIteration)=30*pi/180;
69          end
70          if thisTime>=28 & thisTime<32
71              ref(controlIteration)=110*pi/180;
72          end
73          if thisTime>=32
74              ref(controlIteration)=30*pi/180;
75          end
76
77          %% inicia o controle
78          if controlIteration > 2
79
80              % controle da flexao
81              phiF= [elbowangle(controlIteration) elbowangle(
                    controlIteration-1) elbowangle(
                    controlIteration-2)]';
82
83              sinalF=RF*phiF;
```

```
84
85              entradaF(controlIteration)= ref(
                    controlIteration)*TF-sinalF;
86
87              phi2F= [excitationF(controlIteration-1)
                    excitationF(controlIteration-2)]';
88
89              excitationF(controlIteration)=entradaF(
                    controlIteration) - S1F*phi2F;
90
91              % controle da extensao
92              phiE= [elbowangle(controlIteration) elbowangle(
                    controlIteration-1) elbowangle(
                    controlIteration-2)]';
93
94              sinalE=RE*phiE;
95
96              entradaE(controlIteration)= ref(
                    controlIteration)*TE-sinalE;
97
98              phi2E= [excitationE(controlIteration-1)
                    excitationE(controlIteration-2)]';
99
100             excitationE(controlIteration)=entradaE(
                    controlIteration) - S1E*phi2E;
101
102         end
103
104         %% nao permite que as ativacoes sejam maiores que 1
                e menores que 0
105         if excitationE(controlIteration)>1
106             excitationE(controlIteration)=1;
```

```
107         end
108         if excitationE(controlIteration)<0
109             excitationE(controlIteration)=0;
110         end
111
112
113         if excitationF(controlIteration)>1
114             excitationF(controlIteration)=1;
115         end
116         if excitationF(controlIteration)<0
117             excitationF(controlIteration)=0;
118         end
119
120
121         % printa informacoes da simulacao durante a mesma
122         fprintf('%d/%d: excitacaoF: %f, excitacaoE: %f ,
               angulo: %f, ref: %f \n',controlIteration,
               n_samples, excitationF(controlIteration),
               excitationE(controlIteration), elbowangle(
               controlIteration),ref(controlIteration));
123
124
125     end
126
127 %% ADICAO DO SINAL DE CONTROLE NOS MUSCULOS
128
129 osimModel.updActuators().get('BIClong').addInControls(
       Vector(1,excitationF(controlIteration)),modelControls);
130
131 osimModel.updActuators().get('BICshort').addInControls(
       Vector(1,excitationF(controlIteration)),modelControls);
132
```

```
133 osimModel.updActuators().get('BRA').addInControls(Vector(1,
        excitationF(controlIteration)),modelControls);
134
135 osimModel.updActuators().get('TRIlat').addInControls(Vector
        (1,excitationE(controlIteration)),modelControls);
136
137 osimModel.updActuators().get('TRImed').addInControls(Vector
        (1,excitationE(controlIteration)),modelControls);
138
139 osimModel.updActuators().get('TRIlong').addInControls(
        Vector(1,excitationE(controlIteration)),modelControls);
140
141
142 end
```