UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

BRUNO LUIZ FAUSTINO

CONTROLE DE ÂNGULOS DE FLEXÃO DE MODELO DE PERNA HUMANA ATRAVÉS DE ESTIMULAÇÃO ELÉTRICA FUNCIONAL EM MALHA-FECHADA E COATIVAÇÃO DE MÚSCULOS

FORTALEZA

2022

BRUNO LUIZ FAUSTINO

CONTROLE DE ÂNGULOS DE FLEXÃO DE MODELO DE PERNA HUMANA ATRAVÉS DE ESTIMULAÇÃO ELÉTRICA FUNCIONAL EM MALHA-FECHADA E COATIVAÇÃO DE MÚSCULOS

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. Fabrício 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)

F271c Faustino, Bruno Luiz.
Controle de ângulos de flexão de modelo de perna humana através de estimulação elétrica funcional em malha-fechada e coativação de músculos / Bruno Luiz Faustino. – 2022.
89 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. Fabrício Gonzalez Nogueira.

1. Controle Digital. 2. RST . 3. FES. 4. Flexão de joelho. 5. Flexão de quadril. I. Título.
CDD 621.3

BRUNO LUIZ FAUSTINO

CONTROLE DE ÂNGULOS DE FLEXÃO DE MODELO DE PERNA HUMANA ATRAVÉS DE ESTIMULAÇÃO ELÉTRICA FUNCIONAL EM MALHA-FECHADA E COATIVAÇÃO DE MÚSCULOS

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. Fabrício 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)

Engª. Aparecida Falcão de Andrade
Universidade Federal do Ceará (UFC)

À minha família, por ser a minha maior motivação e me ensinar e ressignificar todos os dias sobre amor: a vontade e as ações que você faz para fazer o outro feliz.

AGRADECIMENTOS

Ao Doutor em Engenharia Elétrica, Ednardo Moreira Rodrigues, e ao graduado em Engenharia Elétrica, Alan Batista de Oliveira, pela adequação do *template* utilizado neste trabalho para que o mesmo ficasse de acordo com as normas da biblioteca da Universidade Federal do Ceará (UFC).

Ao Programa de Educação Tutorial (PET), projeto que me formou profissional e humanamente, responsável por me trazer alegrias e aprendizados ao longo desses 5 anos. Em particular, ao tutor Prof. Dr. René Pastor Torrico Bascopé.

Ao Grupo de Desenvolvimento Aeroespacial (GDAe), onde pude fazer amigos, desafiar-me e aprender sobre temas relevantes e enriquecedores. Em particular, ao subsistema de Eletrônica, onde tive momentos inesquecíveis.

Ao Ramo Estudantil IEEE UFC - Fortaleza, onde conheci pessoas talentosas que realizam um trabalho diferente dos outros projetos que vivenciei durante a universidade.

Ao Grupo de Processamento, Automação, Controle e Robótica (GPAR) e às pessoas que me ajudaram tirando dúvidas ou conversando e tornaram o ambiente amigável. Em particular, Aparecida Falcão, Eugênio Peixoto Júnior, Renata Rodrigues, Talita Otoni.

Ao Prof. Dr. Fabrício Gonzalez Nogueira, que me orientou ao longo de trabalho e durante o Programa de Iniciação à Docência, sempre se mostrando acessível e disposto a colaborar.

Ao Prof. Dr. Wilkley Bezerra Correia, professor atencioso e que sempre busca trazer conteúdo da melhor forma para seus alunos.

Aos meus veteranos de curso, amigos que me inspiram e que servem de espelho profissional e pessoal: Guilherme Lawrence, João Vitor, Rodrigo Tornisiello e Valessa Valentim.

Às pessoas que conheci e marcaram minha trajetória pela universidade, as quais levo com imenso apreço: Danilo Meireles, Ícaro Farias, Jéssica Aragão, Nathanael Vasconcelos, Saynarah Nabuco, Wesley Bezerra, Weslley Barata, Yago Vaz, Yana Soares.

Aos meus amigos de escola que continuaram e continuarão comigo sempre apoiando: Arthur Lucas, Lucas Matheus, Marcos Filho e Nilson Moura.

Aos meus pais e à minha irmã, os motivos pelos quais, durante os 5 anos, procurei viver o máximo da universidade, me envolvendo em diferentes projetos, organizando e criando eventos. Tudo em busca de atingir pessoas e me tornar melhor para fazer jus à confiança e ao apoio que vocês me deram e dão.

“Mas o homem, porque não tem senão uma vida, não tem nenhuma possibilidade de verificar a hipótese através de experimentos, de maneira que não saberá nunca se errou ou acertou ao obedecer a um sentimento.”

(Milan Kundera)

RESUMO

A aplicação de técnicas de estimulação elétrica funcional, do inglês *Functional Electrical Stimulation* (*FES*), consiste na aplicação de pulsos elétricos nos nervos ou músculos com o objetivo de se obter contração muscular. Essas excitações ocasionam movimentações de membros, em geral, superiores e inferiores. Essa técnica é utilizada no tratamento de pessoas com movimentos limitados devido a doenças neurológicas, acidentes e afins. Contudo, dada à natureza não-linear dos músculos do corpo humano, não se pode garantir uma resposta desejada apenas ao aplicar-se excitações sobre os músculos. Para isso, faz-se necessário o estudo de técnicas de controle. Neste trabalho, foram utilizadas estratégias de controle a fim de controlar os ângulos de flexão do joelho e flexão de quadril através da excitação dos músculos responsáveis por ambos os movimentos. As técnicas de identificação e modelagem implementadas abrangem: estudo do comportamento do sistema a respostas do tipo degrau e aplicação de sinais binários pseudo-aleatórios para identificação de equações que modelam o comportamento observado através de métodos numéricos. Posteriormente, realizou-se o projeto de controladores RST através da alocação de polos auxiliares. Os valores dos ângulos de flexão são obtidos através da integração de softwares OpenSim e MATLAB ©. Primeiramente, são realizados os controles individuais de cada movimento para entendimento da atuação de seus respectivos controladores de forma independente. Posteriormente, a coativação dos músculos e o controle dos ângulos de flexão foram atingidos através de controladores RST e denotaram a possibilidade de expansão e exploração de diferentes técnicas de controle nesse campo de pesquisa. Por fim, exploraram-se os limites das regiões de estabilidades dos controladores para ângulos de flexão de quadril acima de 40°, atingindo-se movimentos instáveis. Assim, técnicas de controle não-linear foram propostas.

Palavras-chave: Controle Digital. RST. *FES*. Flexão de joelho. Flexão de quadril.

ABSTRACT

The application of functional electrical stimulation techniques *FES* consists of applying electrical pulses to nerves or muscles in order to obtain muscle contraction. These excitations cause limb movements, in general, upper and lower. This technique is used in the treatment of people with limited movements due to neurological diseases, accidents etc. However, given the non-linear nature of the muscles in the human body, a desired response cannot be guaranteed just by applying such impulses to the muscles. Therefore, it is necessary to study control techniques. In this work, control strategies were used in order to control the angles of knee flexion and hip flexion through the excitation of the muscles responsible for both movements. The identification and modeling techniques implemented include: study of the behavior of the system to step-type responses and application of pseudo-random binary signals to identify equations that model the behavior observed through numerical methods. Subsequently, the design of RST controllers was carried out through the allocation of auxiliary poles. The values of the bending angles are obtained through the integration of OpenSim and MATLAB ©software. First, the individual controls of each movement are carried out to understand the performance of their respective controllers independently. Muscle coactivation and flexion angle control were achieved through RST controllers and denoted the possibility of expanding and exploring different control techniques in this field of research. At the end, the boundaries of stabilization were explored for hip flexion beyond 40°, getting unstable values. Therefore, techniques of non-linear control were proposed.

Keywords: FES. Digital Control. RST . Knee flexion. Hip flexion

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

ARX	*Autoregressive Model with Exogenous inputs*
FES	*Functional Electrical Stimulation*
GUI	*Graphical User Interface*
PRBS	*Pseudo-Random Binary Sequence*
FFT	*Fast Fourier Transform*
LPV	*Linear Parameter-varying*
MQNR	Método dos Mínimos Quadrados Não Recursivo

LISTA DE SÍMBOLOS

ω_0	Frequência natural do sistema
π	Constante irracional aproximada por 3,14
ζ	Fator de amortecimento relativo
f_{prbs}	Frequência do *Pseudo-Random Binary Sequence* (*PRBS*)
L	Comprimento da sequência do *PRBS*
N	Número de células do *PRBS*
q	Variável contínua
s	Variável de frequência em tempo contínuo
T_b	Período de amostragem do *PRBS*
t_r	Tempo de subida do sinal
z	Variável contínua

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de controle estão presentes na natureza e fazem parte do nosso cotidiano, estando presentes mesmo no corpo humano (NISE, 2013). O estudo dos sistemas visa o mapeamento de suas características e o conseguinte controle do resultado desejado. Em sistemas de controle digital, as estratégias utilizadas levam em consideração técnicas de identificação, modelagem e projeto de controladores digitais.

Dentre as aplicações de controladores, tem-se estudos sobre controle de estimulação elétrica funcional, do inglês *FES*, técnica de estimulação dos músculos por meio de impulsos elétricos através de eletrodos com o objetivo de restaurar funções comprometidas em pacientes com lesões na coluna espinhal (LIAO *et al.*, 2014; VETTE *et al.*, 2015) e atenuação de distúrbios neurológicos como síndrome de Tourette e transtorno obsessivo compulsivo (FODSTAD; HARIZ, 2007). O estudo de controle aplicado a *FES* tem mostrado as vastas possibilidades de sua aplicação, pois a excitação em malha aberta representa um sistema não linear (CRAGO *et al.*, 1980). Assim, há aplicações pertinentes, seja em membros superiores, como a flexão de cotovelo (BO; MOURA, 2015), seja em membros inferiores, como a flexão do calcanhar (VETTE *et al.*, 2007).

Apesar das diversas possibilidades de estudo de controladores em *FES*, ainda há limitações quanto ao uso propagado dessas técnicas no cotidiano de tratamento médico, seja por custos de aquisição, seja por definição da técnica de controle implementada (BO; MOURA, 2015). De acordo com os dados da literatura, o controlador prioritariamente utilizado é o PID clássico, independentemente da plataforma de simulação empregada (ALIBEJI *et al.*, 2013; VETTE *et al.*, 2007; KOZAN, 2012).

A fim de explorar os horizontes desse campo de estudo, aplicar-se-á neste trabalho técnicas de controle multivariável descentralizado, onde as malhas são projetadas de forma independente, e a implementação do controlador RST na atuação de eletroestimuladores funcionais sobre o membro inferior do corpo humano. Em particular, a saída a ser controlada será o ângulo de flexão do joelho e a entrada será a excitação aplicada aos músculos inerentes ao movimento em questão. Posteriormente, o controle simultâneo de ambas coordenadas será realizado, visto que a coativação de músculos alivia a desestabilização causada pelo torque dos movimentos nas juntas (ZHOU *et al.*, 1996).

Através da plataforma de código aberto *OpenSim* ©, versão 4.3, o resultado poderá ser visualizado por meio da interface gráfica do usuário, do inglês *Graphical User Interface*

(*GUI*), aplicado a um modelo de perna onde é possível observar-se os músculos e a estrutura óssea inerente ao movimento de flexão do joelho.

1.1 Objetivos

Como objetivo principal, promover o estudo e a aplicação de técnicas de controle digital em *FES*. Como objetivos específicos:

- Implementar algoritmos de identificação e modelagem de sistemas;
- Projetar controladores RST através da identificação pelo MQNR a fim de obter um modelo ARX;
- Realizar a integração *OpenSim* ©- MATLAB ©;
- Validação do modelo e apresentação gráfica dos resultados obtidos.

1.2 Organização do trabalho

Partindo-se da fundamentação teórica aos resultados obtidos, o trabalho divide-se em:

- Capítulo 1: A introdução objetiva fornecer uma visão geral sobre o projeto, bem como justificar a escolha do tema e a abordagem;
- Capítulo 2: A fundamentação teórica pauta os principais conceitos utilizados como base para a identificação, a modelagem e o controle do sistema de estudo;
- Capítulo 3: A metodologia apresenta as práticas utilizadas, fundamentadas pelo capítulo 2, a fim de alcançar-se os objetivos de projeto;
- Capítulo 4: Os resultados expostos trazem as consequências obtidas a partir da metodologia do projeto;
- Capítulo 5: A conclusão apresenta interpretações, aprendizagens e sugestões de projetos futuros de acordo com as possibilidades encontradas ao longo do trabalho.

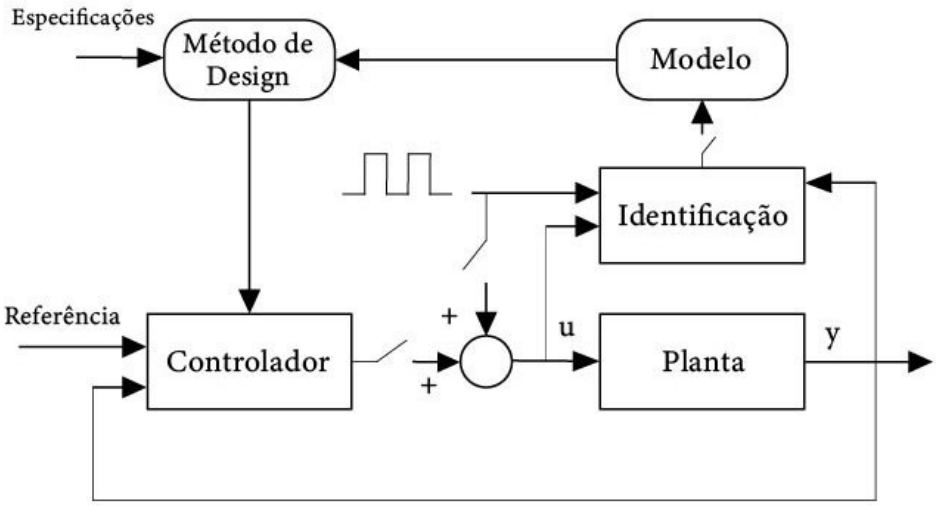
2 FUNDAMENTAÇÃO TEÓRICA

Tendo em vista a identificação, modelagem e projeto dos controladores nos sistemas estudados neste trabalho. Faz-se necessária a fundamentação teórica dos conceitos de controle digital aplicados.

2.1 Visão Geral sobre controle digital

O modelo apresentado na figura 1 servirá de base para o estudo e a implementação do controlador.

Figura 1 – Diagrama modelo de sistema de controle digital



Fonte: Adaptado de (LANDAU; ZITO, 2006)

Assim, aplicando sobre o sistema uma sequência binária pseudo-aleatória, do inglês *PRBS*, a identificação dar-se-á por meio do Método dos Mínimos Quadrados Não Recursivo (MQNR). A partir disso, obter-se-á um modelo auto-regressivo para variável exógena, do inglês *Autoregressive Model with Exogenous inputs* (*ARX*). Com as especificações de projeto definidas, o projeto do controlador é realizado para que siga a referência desejada. Observa-se que, por se tratar de sistemas discretos, os sinais são medidos por amostras e deve-se atentar ao tempo de amostragem definido.

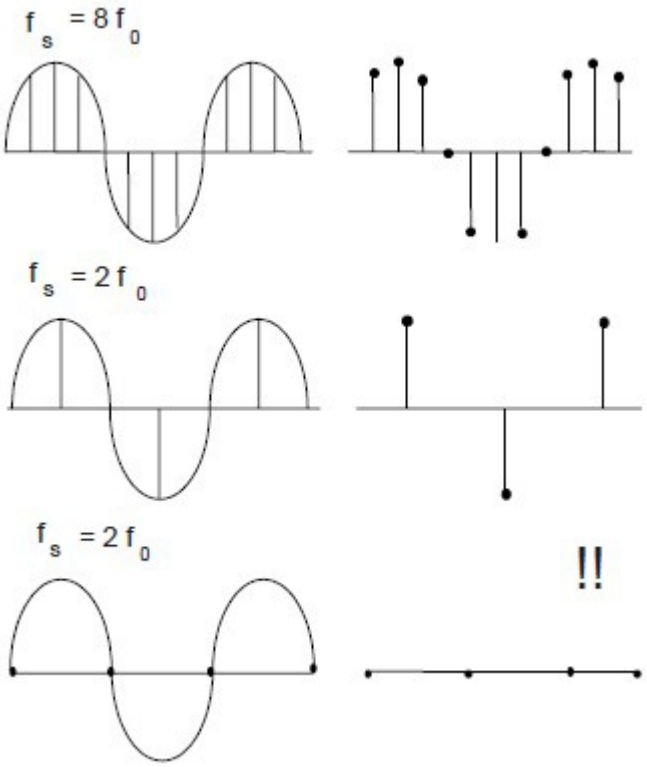
2.2 Identificação da Planta

Este tópico trata dos passos observados no processo de identificação do sistema e seu comportamento perante uma pertubação.

2.2.1 Tempo de amostragem

A escolha do tempo de amostragem deve seguir parâmetros definidos a fim de corresponder a condição imposta pelo teorema de Nyquist, onde $f_s > 2f_{max}$, em que f_s representa a frequência de amostragem escolhida e f_{max} é a frequência máxima de transmissão do sistema.

Figura 2 – Discretização de sinal senoidal



Fonte: (LANDAU; ZITO, 2006)

Nota-se que na figura 2, ao escolher-se uma frequência de amostragem igual ao dobro da frequência natural do sinal, perdem-se informações do real comportamento do sinal. No caso em que a frequência coincide com $n\pi$, onde $n = 1,2,3...$, o sinal discretizado apresentado é nulo.

Para sistemas de controle discreto, a escolha dar-se-á de acordo com a largura de banda em malha fechada, de acordo com (LANDAU; ZITO, 2006):

$$6f_B^{CL} \leq f_s \leq 25f_B^{CL}. \tag{2.1}$$

Onde f_s é a frequência de amostragem e f_B^{CL} é a largura de banda do sistema em malha fechada, visto que a frequência de modo dominante para o sinal controlado deve ser premeditada a fim de se evitar erros de amostragem em questões práticas. Além disso, costuma-se adotar o critério de, no mínimo, 10 vezes a frequência dominante do sistema em malha fechada, dada a riqueza de amostras requerida.

A depender da ordem do sistema estudado, a escolha seguirá parâmetros indicados pelas respectivas funções de transferência.

- **Primeira ordem**

$$H(s) = \frac{1}{1+sT_0}. \tag{2.2}$$

Onde sua frequência de largura de banda é definida por:

$$f_B = f_0 = \frac{1}{2\pi T_0}. \tag{2.3}$$

Relacionando com 2.1, tem-se que:

$$\frac{T_0}{4} \leq T_s \leq T_0. \tag{2.4}$$

Em que $T_s = 1/f_s$ é o tempo de amostragem e T_0 é a constante de tempo desejada para a resposta em malha fechada.

- **Segunda ordem**

$$H(s) = \frac{\omega_0}{s^2 + 2\zeta\omega_0 s + \omega_0^2}. \tag{2.5}$$

Para a frequência de largura de banda:

$$f_{BW} = \frac{\omega_0}{2\pi} \quad (\zeta = 0,7). \tag{2.6}$$

$$f_{BW} = \frac{0,6\omega_0}{2\pi} \quad (\zeta = 1). \tag{2.7}$$

Assim, relacionando ambas equações com 2.1:

$$\frac{0,25}{\omega_0} \leq T_s \leq \frac{1}{\omega_0} \quad (\zeta = 0,7). \tag{2.8}$$

$$\frac{0,4}{\omega_0} \leq T_s \leq \frac{1,75}{\omega_0} \quad (\zeta = 1). \tag{2.9}$$

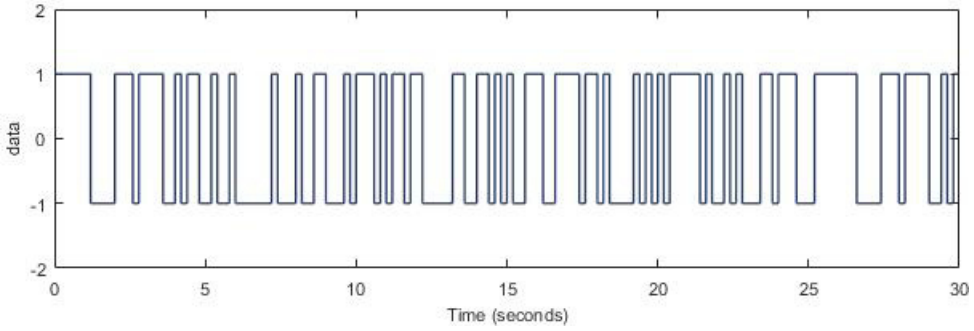
Em geral, deseja-se uma resposta amortecida para o sistema controlado. Assim, adotam-se valores de ζ entre 0,7 e 1. Para essa condição, faz-se uma aproximação entre 2.8 e 2.9:

$$\frac{0,25}{\omega_0} \leq T_s \leq \frac{1,5}{\omega_0} \quad (0,7 \leq \zeta \leq 1). \tag{2.10}$$

2.2.2 PRBS

Na análise de comportamento do sistema, aplica-se esse sinal a fim de observar-se a resposta a uma entrada variável, pois, por vezes, a modelagem do comportamento do sistema por parâmetros físicos pode ser custosa. Assim, analisar experimentalmente o sistema torna-se uma opção mais adequada.

Figura 3 – Exemplo de sinal PRBS gerado para 7 células



Fonte: O autor.

O sinal PRBS é gerado a partir de operações lógicas e o número de células (N) escolhidas influenciará no tamanho de sua sequência, pois segue 2.11

$$L = 2^N - 1. \tag{2.11}$$

Assim, para uma escolha de N=5, o tamanho da sequência a se repetir será 31.

Para a escolha do sinal PRBS, atenta-se aos seguintes parâmetros:

$$T_b N \geq t_r. \tag{2.12}$$

$$\frac{1}{(2^N-1)T_b} \leq f_{prbs} \leq \frac{0,44}{T_b}. \tag{2.13}$$

Para que 2.12 e 2.13 sejam correspondidas, aplica-se uma pertubação ao sistema e observa-se sua frequência dominante. Assim, deve-se garantir que o sinal PRBS gerado possua energia relevante nessa faixa de frequência. Esse estudo pode ser feito por meio da transformada rápida de Fourier, do inglês *Fast Fourier Transform* (FFT).

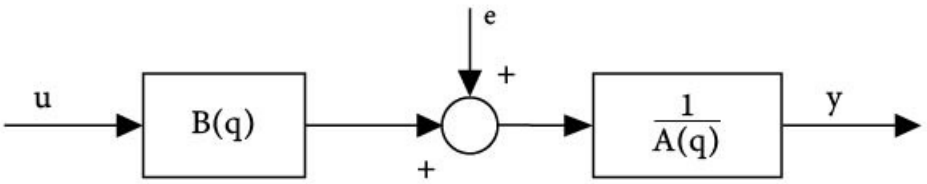
2.3 Modelagem da Planta

Uma vez definido o sinal *PRBS* a ser aplicado na entrada do sistema para sua identificação, dá-se início à modelagem da resposta obtida. Salienta-se que o modelo a ser obtido provém de amostras e utiliza a variável contínua **z**.

2.3.1 Modelo ARX

O modelo autoregressivo com saídas exógenas é a representação de um ambiente determinístico, onde a variável de saída depende linearmente de seus valores prévios (LANDAU; ZITO, 2006). O modelo escolhido para modelar o sistema é o *ARX*, representado na figura 4.

Figura 4 – Diagrama de modelo ARX



Fonte: Autor.

A equação que rege o sistema é dada por 2.14:

$$A(q^{-1})y(k) = q^{-d}B(q^{-1})u(k) + e(k). \tag{2.14}$$

Onde *u(k)* é o sinal de entrada, *e(k)* é o erro, q^{-d} o atraso do sistema e:

$$A(q^{-1}) = 1 + a_1 q^{-1} + ... + a_{n_a} q^{-n_a}. \tag{2.15}$$

$$B(q^{-1}) = b_1 q^{-1} + ... + b_{n_b} q^{-n_b}. \tag{2.16}$$

Associando 2.14, 2.15 e 2.16, obtém-se:

$$y(k) = -a_1 y(k-1) - ... - a_{n_a} y(k-n_a) + b_1 u(k-d-1) + ... + b_{n_b} u(k-d-n_b) + e(t). \tag{2.17}$$

Forma geral de 2.17:

$$y(k) = -\sum_{i=1}^{n_a} a_i y(k-i) + \sum_{i=1}^{n_b} b_i u(k-i-d) + e(k). \tag{2.18}$$

Por fim, em forma de regressores:

$$y(k) = \theta^T \phi. \tag{2.19}$$

Onde $\theta := [a_1 \quad a_2 \quad ... \quad a_{n_a} \quad b_1 \quad ... \quad b_{n_b}]^T$ e
$\phi(k) := [-y(k-1) \quad y(k-2) \quad ... \quad -y(k-n_a) \quad u(k-n-1) \quad ... \quad y(k-d-n_b)]^T$.

2.3.2 Método dos Mínimos Quadrados Não Recursivo

Tendo como base 2.19, o método tem como objetivo diminuir os erros, ou seja, a distância entre as amostras obtidas e o valor estimado.

$$y(k) = \phi^T(k)\theta + e(k). \tag{2.20}$$

Para N amostras:

$$\begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ y(N-1) \end{bmatrix} = \begin{bmatrix} \phi^T(0) \\ \phi^T(1) \\ \vdots \\ \phi^T(N-1) \end{bmatrix} \theta + \begin{bmatrix} e(0) \\ e(1) \\ \vdots \\ e(N-1) \end{bmatrix}. \tag{2.21}$$

Portanto:

$$Y = \phi\theta + E. \tag{2.22}$$

A fim de minimizar o vetor E - vetor de erros, estima-se um vetor $\hat{\theta}$, onde $\hat{Y} = \phi\hat{\theta}$. Assim,

$$E = Y - \hat{Y} = Y - \phi\hat{\theta}. \tag{2.23}$$

O objetivo é determinar o mínimo valor para J_{LS}, em que:

$$J_{LS} = ||Y - \phi\hat{\theta}||^2. \tag{2.24}$$

A equação 2.24 pode ser reescrita como (LIMA; (BRASIL), 2008):

$$\begin{aligned} J_{LS} &= (Y - \phi\hat{\theta})^T (Y - \phi\hat{\theta}) \\ &= Y^T Y - Y^T \phi\hat{\theta} - \hat{\theta}^T \phi^T Y + \hat{\theta}^T \phi^T \phi\hat{\theta} \\ &= Y^T Y - 2Y^T \phi\hat{\theta} + \hat{\theta}^T \phi^T \phi\hat{\theta}. \end{aligned}$$

A fim de minimizar o erro, calcula-se:

$$\frac{\partial J_{LS}}{\partial\hat{\theta}} = 0$$
$$-2\phi^T Y + 2\phi^T \phi\hat{\theta} = 0.$$

Portanto, o vetor $\hat{\theta}$ correspondente será:

$$\hat{\theta} = [\phi^T \phi]^{-1} \phi^T Y. \tag{2.25}$$

Ressaltam-se as condições para que 2.25 possua solução (LIMA; (BRASIL), 2008; LANDAU; ZITO, 2006):

- $\phi^T \phi$ é não-singular (pode ser invertida);
- O sistema deve ser regularmente excitado a fim de evitar linhas em comum;

2.4 Controlador RST

A figura 5 representa uma estrutura de controlador digital funcional a partir de valores de saída, de controle e de referência em instantes distintos (t, t-1, ...). A partir da discretização

dos controladores PI ou PID, obtém-se a lei de controle a partir do diagrama - alternando-se a quantidade de coeficientes. Para fins de demonstração, usar-se-á a estrutura do PI (LANDAU; ZITO, 2006):

$$u(t) = \frac{K(1-q^{-1}) + \frac{KT_s}{T_i}}{1-q^{-1}}[r(t) - y(t)]. \tag{2.26}$$

Ao multiplicar ambos lados de 2.26 por $(1-q^{-1})$:

$$S(q^{-1})u(t) = T(q^{-1})r(t) - R(q^{-1})y(t). \tag{2.27}$$

Em que

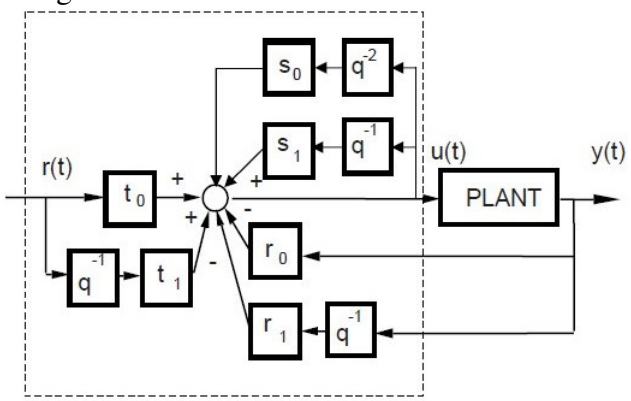
$S(q^{-1}) = 1 - q^{-1}$.

$R(q^{-1}) = T(q^{-1}) = K(1 + T_s/T_i) - Kq^{-1}$.

Ao dividir 2.27 por $S(q^{-1})$, obtém-se:

$$u(t) = -\frac{R(q^{-1})}{S(q^{-1})}y(t) + \frac{T(q^{-1})}{S(q^{-1})}r(t). \tag{2.28}$$

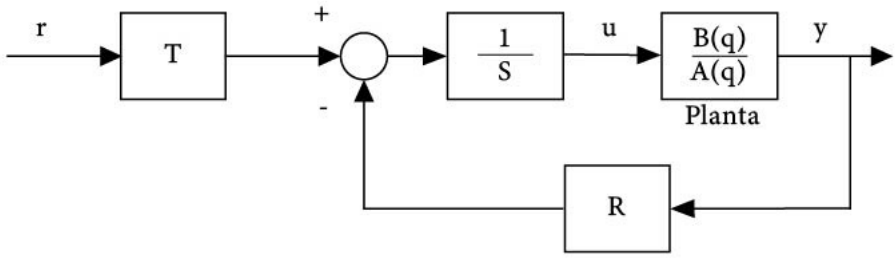
Figura 5 – Controlador digital



Fonte: (LANDAU; ZITO, 2006)

A equação 2.28 serve de base para o controlador RST - em geral, o termo T é diferente de R -, representado na figura 6, o qual atua alocando os polos da planta em malha fechada de acordo com os requisitos do projeto. Tal método é conhecido como **Alocação de Polos**, do inglês *Pole Placement* (PARASKEVOPOULOS, 1996).

Figura 6 – Diagrama de blocos do controlador RST



Fonte: O autor.

Em malha fechada, a equação que descreve o comportamento do sistema é dada por (adicionou-se um operador de atraso q^{-d} multiplicado a $B(q)$ a fim de facilitar a generalização):

$$H_{CL} = \frac{q^{-d}T(q^{-1})B(q^{-1})}{A(q^{-1})S(q^{-1}) + q^{-d}B(q^{-1})R(q^{-1})} = \frac{q^{-d}T(q^{-1})B(q^{-1})}{P(q^{-1})}. \tag{2.29}$$

Onde 2.30 é conhecida como **equação diofantina**.

$$P(q^{-1}) = A(q^{-1})S(q^{-1}) + q^{-d}B(q^{-1})R(q^{-1}). \tag{2.30}$$

Em termos gerais, pode-se representar $P(q^{-1})$ como:

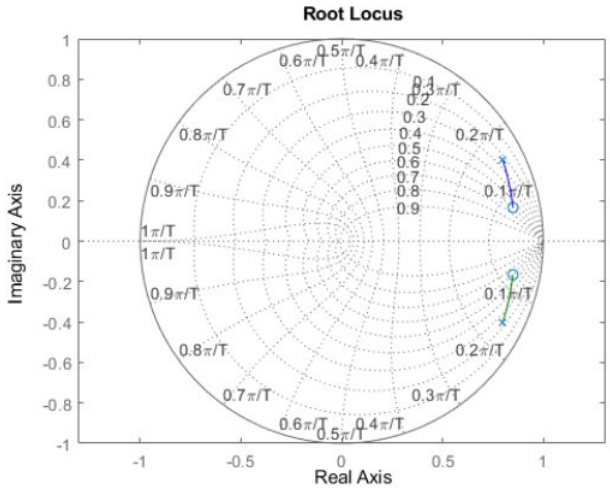
$$P(q^{-1}) = P_D(q^{-1})P_F(q^{-1}).$$

Em que P_D representa polos dominantes e P_F os polos auxiliares - estes serão escolhidos de forma a atingir critérios de robustez do sistema, atuando de forma mais rápida que os polos dominantes e propositalmente próximos do ponto 0 do mapa de polos e zeros, representado na figura 7.

Para casos em que o sistema pode ser representado por uma equação de 2ª ordem - caso abordado neste trabalho -, realiza-se a associação:

$$\begin{aligned} P(q^{-1}) &= 1 + p_1'q^{-1} + p_2'q^{-2} = A(q^{-1})S(q^{-1}) + B(q^{-1})R(q^{-1}) \\ &= (1 + a_1q^{-1} + a_2q^{-2})(1 - q^{-1})(1 + s_1'q^{-1}) + (b_1q^{-1} + b_2q^{-2})(r_0 + r_1q^{-1} + r_2q^{-2}) \\ &= A'(q^{-1})S'(q^{-1}) + B(q^{-1})R(q^{-1}). \end{aligned}$$

Figura 7 – Exemplo de mapa de polos e zeros



Fonte: Inc (2021)

Representando um sistema de 4ª ordem - disso, faz-se uso de mais dois polos auxiliares, representados por α - neste trabalho, os polos auxiliares são escolhidos próximo ao 0 do centro, geralmente em 0,5 e 0,1 ao longo dos testes realizados, dada à estabilidade do ponto:

$$P(q^{-1}) = (1 + p'_1 q^{-1} + p'_2 q^{-2})(1 + \alpha_1 q^{-1})(1 + \alpha_2 q^{-2}).$$

Por fim, o polinômio $P(q^{-1})$ com 4 polos será:

$$\begin{aligned} P(q^{-1}) &= 1 + p_1 q^{-1} + p_2 q^{-2} + p_3 q^{-3} + p_4 q^{-4} \\ &= (1 + a'_1 q^{-1} + a'_2 q^{-2} + a'_3 q^{-3})(1 + s'_1 q^{-1}) + (b_1 q^{-1} + b_2 q^{-2})(r_0 + r_1 q^{-1} + r_2 q^{-2}). \end{aligned} \tag{2.31}$$

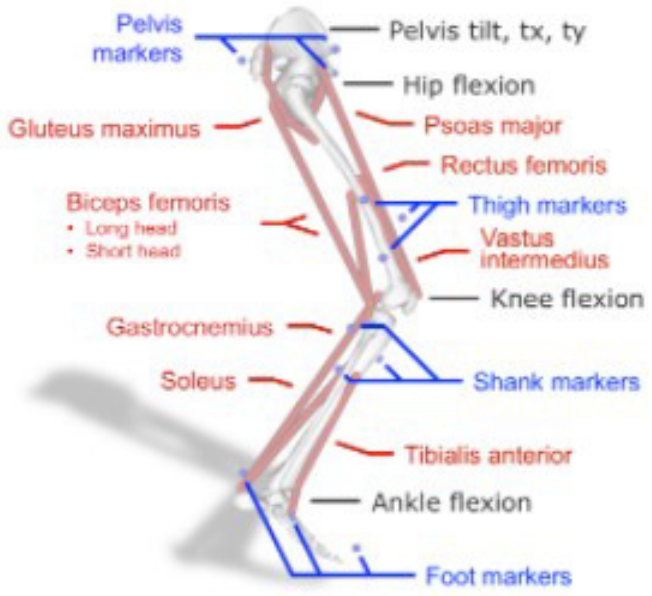
Onde, através de algoritmos, o sistema de equações a ser resolvido será:

$$\begin{aligned} p_1 &= b_1 r_0 + s'_1 + a'_1 \\ p_2 &= b_2 r_0 + b_1 r_1 + s'_1 a'_1 + a'_2 \\ p_3 &= b_2 r_1 + b_1 r_2 + s'_1 a'_2 + a'_3 \\ p_4 &= b_2 r_2 + s'_1 a'_3. \end{aligned}$$

3 METODOLOGIA

Este capítulo trata da identificação e da modelagem do sistema escolhido, bem como o projeto do controlador para os parâmetros escolhidos. O modelo escolhido foi o *leg6dof9musc*, representado na figura 8, com 6 graus de liberdade e 9 músculos.

Figura 8 – Modelo *leg6dof9musc* do Opensim ©



Fonte: Documentation (2021)

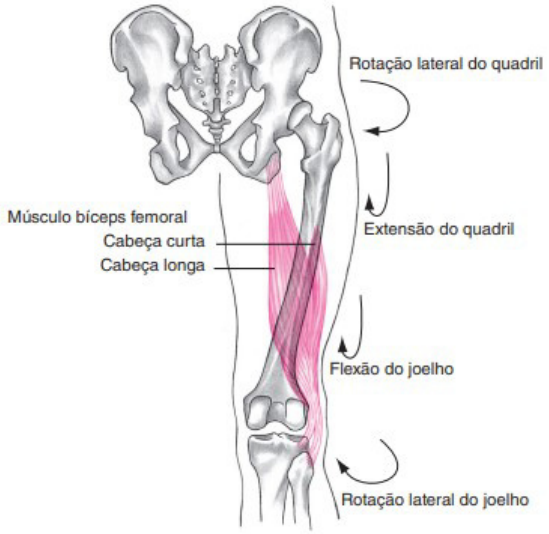
3.1 Escolha dos músculos excitados

A cinesiologia estrutural é a área responsável pelo estudo dos músculos, dos ossos e das articulações na ciência do movimento (FLOYD, 2016). Neste trabalho, o controle de dois movimentos foram estudados: articulação do joelho e articulação do quadril.

Com relação à flexão do joelho, dois músculos principais são responsáveis: bíceps femoral, composto por duas cabeças denominadas curta e longa. Os músculos estão representados na figura 9.

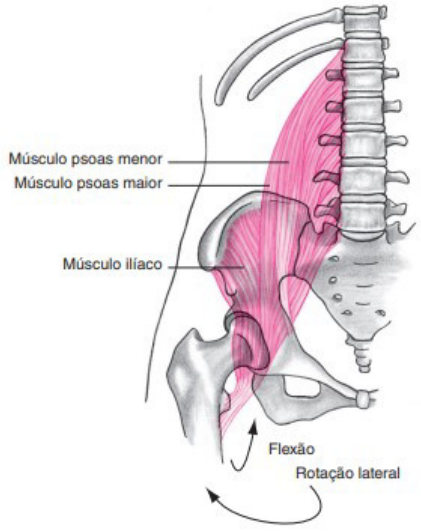
Quanto à flexão de quadril, o músculo atuado foi o psoas - figura 10.

Figura 9 – Músculos responsáveis pela flexão de joelho



Fonte: (FLOYD, 2016)

Figura 10 – Músculos responsáveis pela flexão de quadril



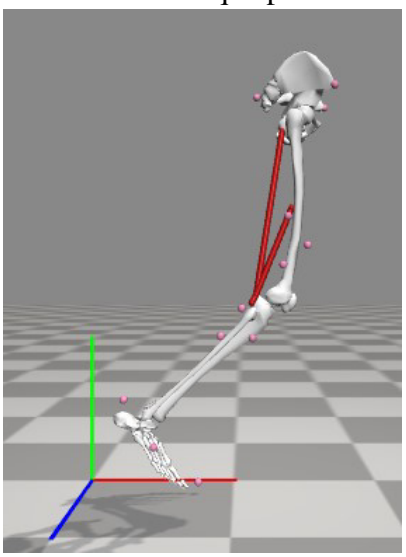
Fonte: (FLOYD, 2016)

3.2 Estudo e modelagem - Flexão do joelho

A fim de conhecer a dinâmica do sistema, aplicou-se um degrau de entrada de amplitude $0,5$ sobre os músculos componentes do bíceps femoral para causar a flexão no joelho, exposto na figura 11. Ao aplicar a excitação, travaram-se as demais coordenadas - através do código - do modelo exceto à própria do joelho. Assim, a resposta obtida está apresentada na figura 12.

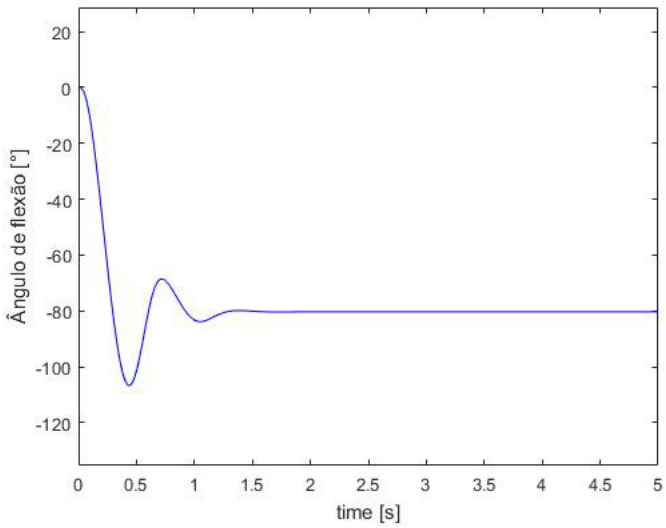
Os valores negativos deram-se tendo em vista que o ângulo formado pela flexão do joelho são negativos em relação à referência - apresentada na imagem 11.

Figura 11 – Visão lateral do modelo com destaque para músculos do bíceps femoral



Fonte: O autor.

Figura 12 – Resposta ao degrau 0,5 da flexão de joelho em malha aberta

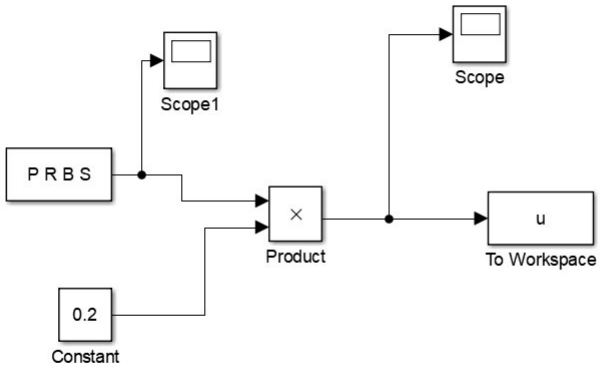


Fonte: O autor.

A resposta adquirida apresentou frequência próxima a *1,6 Hz*. Essa medida será ideal para a aplicação do sinal *PRBS*, uma vez que, realizada a FFT, observou-se a potência do sinal na frequência natural do sistema para que haja identificação satisfatória - o que demonstra a figura 15. Além disso, dada à forma de resposta, adotou-se uma equação de 2ª ordem para a modelagem do sistema e um tempo de amostragem de **0,02 segundos**, segundo a equação 2.10.

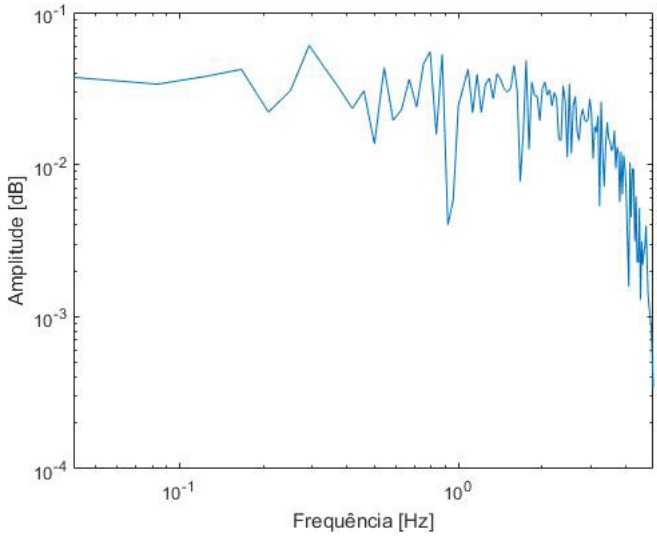
Para geração do sinal *PRBS*, foi utilizado o bloco gerador do Simulink, ambiente periférico do MATLAB ©- figura 13. O sinal aplicado para identificação do modelo *ARX* teve as especificações expressas na tabela 1, em respeito à equação 2.13. A FFT do sinal de entrada a ser aplicado é representado pela figura 14.

Figura 13 – Diagrama de blocos gerador de PRBS



Fonte: O autor.

Figura 14 – FFT do sinal PRBS



Fonte: O autor.

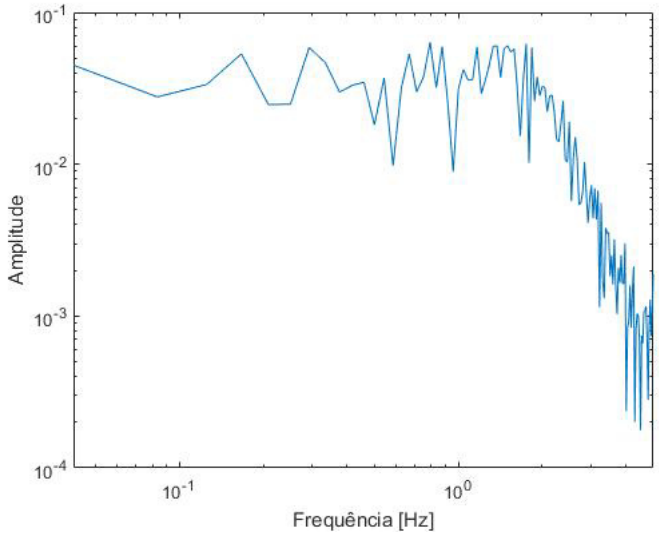
Tabela 1 – Especificações do sinal PRBS

Número de células (N)	Amplitude	Tb	Tempo de geração (s)
7	0,3 a 0,7	0,2	30

Fonte: Autor.

Durante o processo de identificação do modelo *ARX*, eliminaram-se amostras com pouca correlação entre entrada e saída, além da componente DC - calculada através da média das amostras. Com isso, ao aplicar-se o MQNR, o modelo ARX encontrado foi equivalente à seguinte equação:

Figura 15 – FFT da resposta ao PRBS na flexão de joelho

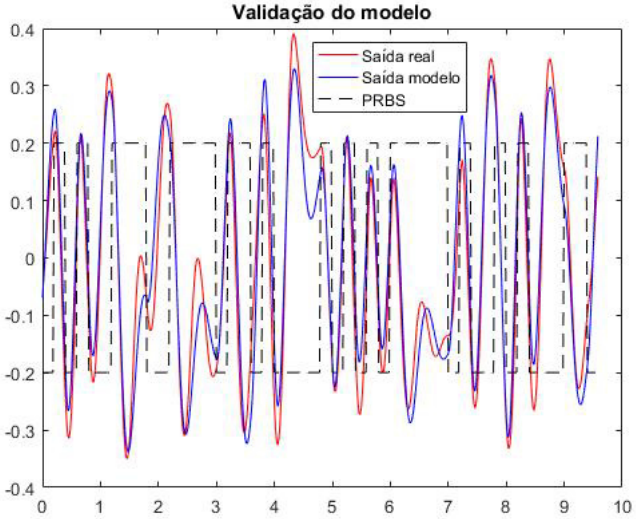


Fonte: O autor.

$$F_d(z^{-1}) = \frac{-0,006851z^{-1} - 0,02963z^{-2}}{1 - 1,843z^{-1} + 0,8915z^{-2}}. \tag{3.1}$$

A figura 16 mostra a validação entre o sinal PRBS e o modelo encontrado, com índice de identificação equivalente a $90,84\%$.

Figura 16 – Validação do modelo para a flexão do joelho



Fonte: Autor.

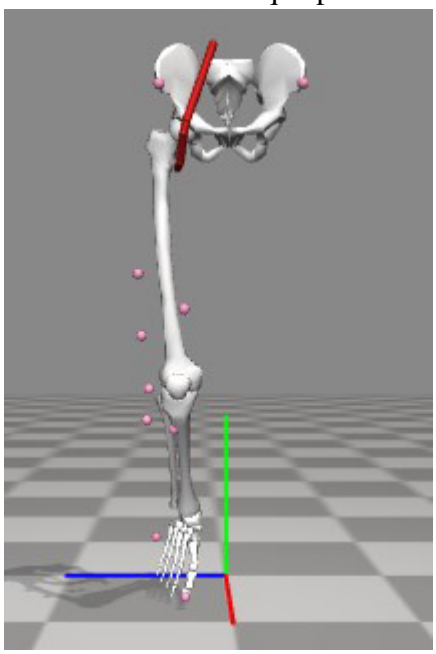
Através do método de alocação de polos, o controlador RST apresentou os seguintes parâmetros:

$$R(q^{-1}) = -28,3347 + 52,7025q^{-1} - 24,7229q^{-2}.$$
$$S(q^{-1}) = 1 - 0,1824q^{-1} - 0,8175q^{-2}. \tag{3.2}$$
$$T(q^{-1}) = -0,3552.$$

3.3 Estudo e modelagem - Flexão de quadril

Assim como a flexão do joelho, os mesmos passos foram aplicados para a flexão do quadril. A diferença principal ocorre na excitação do músculo psoas, representado na figura 17.

Figura 17 – Visão posterior do modelo com destaque para músculo psoas



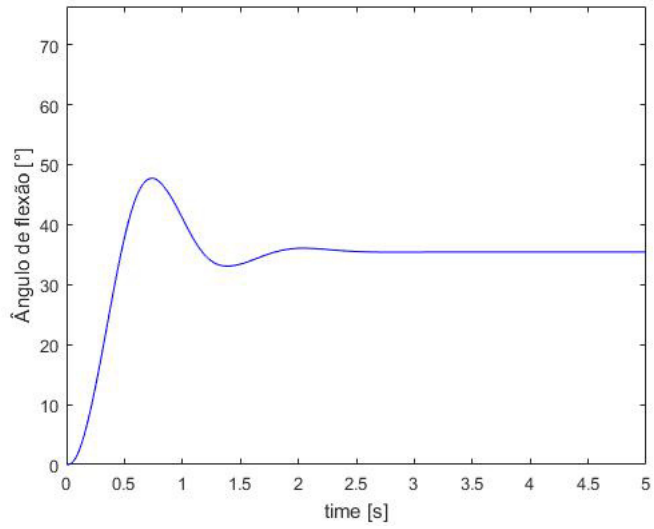
Fonte: O autor.

Para estudo do comportamento do sistema em malha aberta, aplicou-se um degrau de amplitude 0,5. A resposta está representada na figura 18. O modelo escolhido para identificação usará um modelo de 2ª ordem.

A resposta apresenta um sinal estável em regime permanente, com frequência próxima a 0,8 Hz. Após a aplicação do sinal, mediu-se a FFT, a fim de observar a intensidade do sinal na região próxima a essa frequência. A figura 19 demonstra que o sinal apresenta boa quantidade de dados ao longo dessa região.

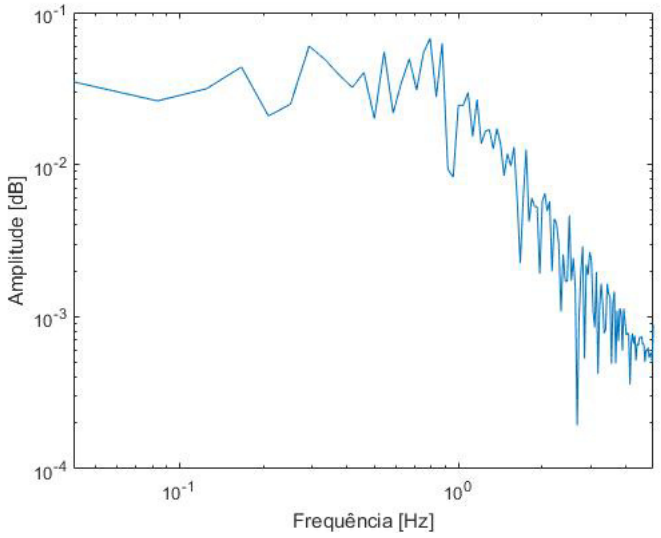
O sinal PRBS utilizado foi o mesmo apresentado na tabela 1. Assim, desconsiderando regiões de pouca relação entre entrada e saída e subtraindo-se a média das amostras, identificou-se o modelo ARX da equação 3.3, com 94,99% de acurácia. O sinal corresponde à figura

Figura 18 – Resposta ao degrau 0,5 da flexão de quadril em malha aberta



Fonte: O autor.

Figura 19 – FFT da resposta ao PRBS na flexão de quadril



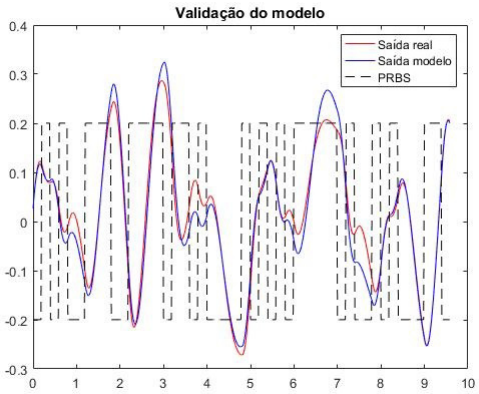
Fonte: O autor.

20.

$$F_d(z^{-1}) = \frac{0,0007291z^{-1} + 0,008516z^{-2}}{1 - 1,938z^{-1} + 0,9473z^{-2}}. \tag{3.3}$$

Através do método de alocação de polos, o controlador RST apresentou os seguintes parâmetros:

Figura 20 – Validação do modelo para a flexão de quadril



Fonte: O autor.

$$
\begin{aligned}
R(q^{-1}) &= 130,0897 - 248,3462q^{-1} - 27,492q^{-2}. \\
S(q^{-1}) &= 1 - 0,04581q^{-1} - 0.9541q^{-2}. \\
T(q^{-1}) &= 0,4181.
\end{aligned} \tag{3.4}
$$

3.4 Implementação do controlador

Para a implementação do controlador via código, faz-se necessário observar o diagrama de blocos apresentado na figura 6. A excitação no músculo é representada pelo sinal *u* na figura e a referência é representada por *r*. Estabelecendo as equações do diagrama de blocos:

$$S * [u(k) \quad u(k-1) \quad u(k-2)] = T * r(k) - R. * [y(k) \quad y(k-1) \quad y(k-2)].$$

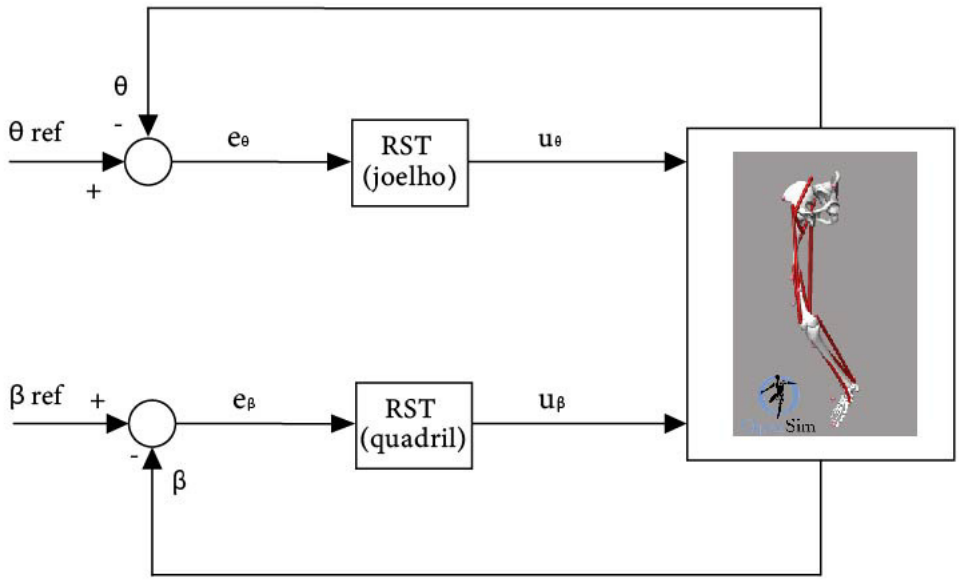
Uma vez que o primeiro termo de S é igual a 1, pode-se evidenciar o termo u(k). Assim, a equação 3.5 representa a atualização que a excitação deve receber.

$$u(k) = T * r(k) - R. * [y(k) \quad y(k-1) \quad y(k-2)] - S. * [0 \quad u(k-1) \quad u(k-2)]. \tag{3.5}$$

4 RESULTADOS

Este capítulo trata dos resultados obtidos após a implementação de ambos controladores. Primeiramente, optou-se por controlar cada coordenada individualmente e depois implementar os controladores atuando simultaneamente. A fim de facilitar o entendimento, a figura 21, inspirada no material apresentado em Sousa (2019), representa o diagrama de blocos geral do sistema estudado.

Figura 21 – Diagrama de blocos geral

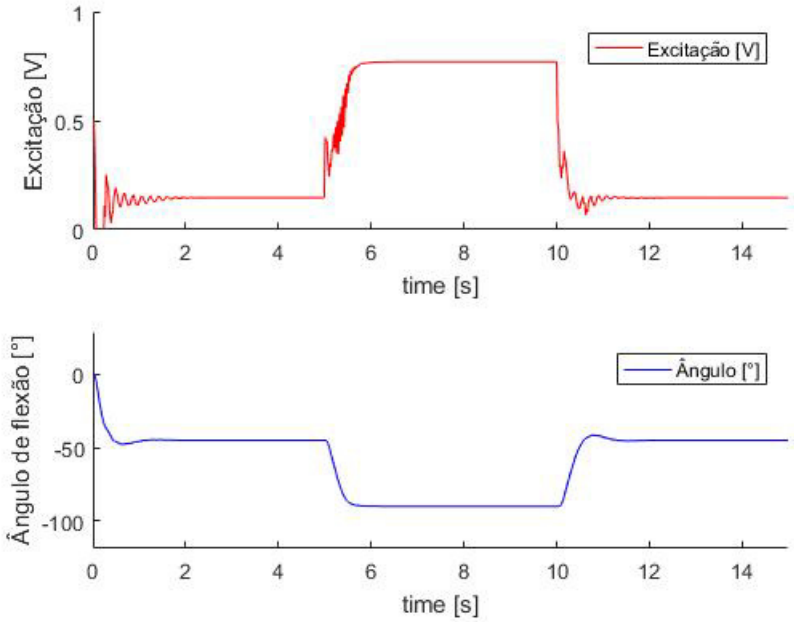


Fonte: Adaptado de Sousa (2019).

4.1 Controle de flexão do joelho

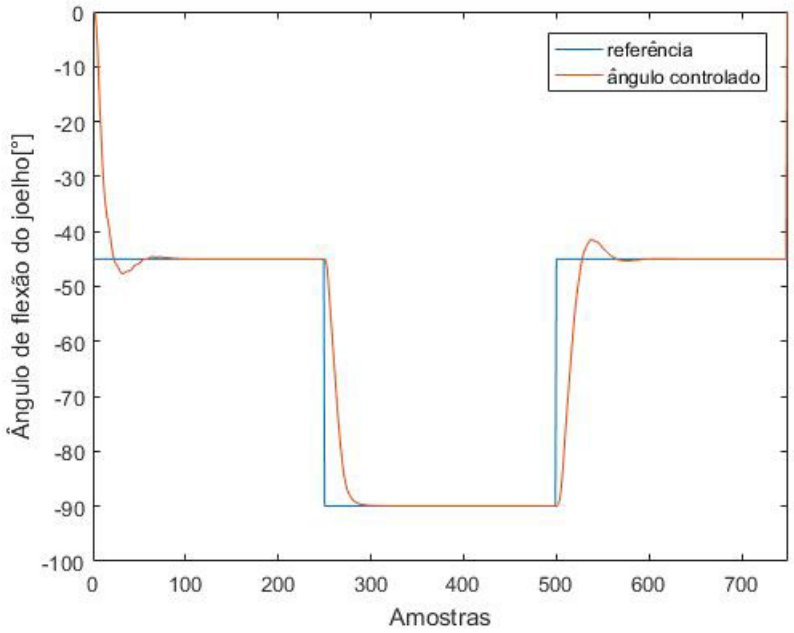
Após a implementação, simulou-se durante 15 segundos a flexão do joelho, como apresentado na figura 22. A referência escolhida foi de -45° até 5 segundos, -90° entre 5 e 10 segundos e -45° a partir de 10 segundos - figura 23. As escolhas se basearam nas zonas de ângulos exploradas durante a fase de identificação do modelo. Link disponível para acessar vídeo do movimento:https://youtu.be/mFXboYXj8sE.

Figura 22 – Excitação e controle da flexão de joelho



Fonte: O autor.

Figura 23 – Comparativo entre referência e ângulo controlado da flexão de joelho

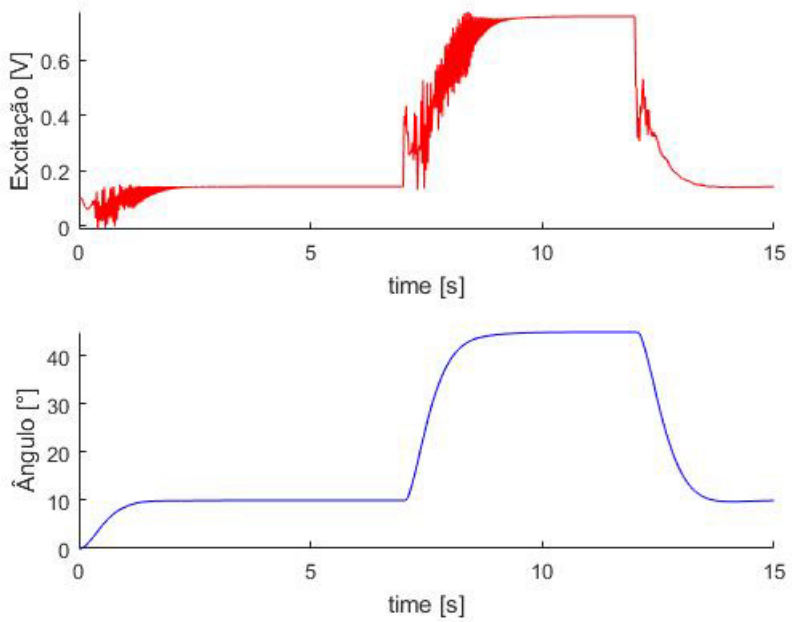


Fonte: O autor.

4.2 Controle de flexão do quadril

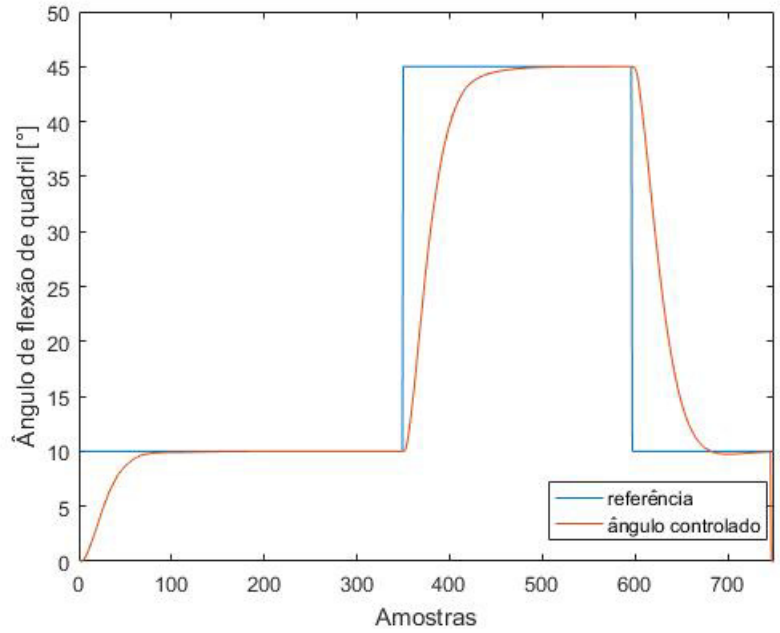
Simulou-se durante 15 segundos a flexão do quadril - figura 24. Diferente da flexão de joelho, os ângulos aqui controlados são positivos. Escolheram-se como referências os ângulos 10° e 45°. O comparativo está representado na figura 25. O link disponível para visualizar o movimento é o seguinte: https://youtu.be/0ZxF7YXJX3A.

Figura 24 – Excitação e controle da flexão de quadril



Fonte: O autor.

Figura 25 – Comparativo entre referência e ângulo controlado da flexão de quadril

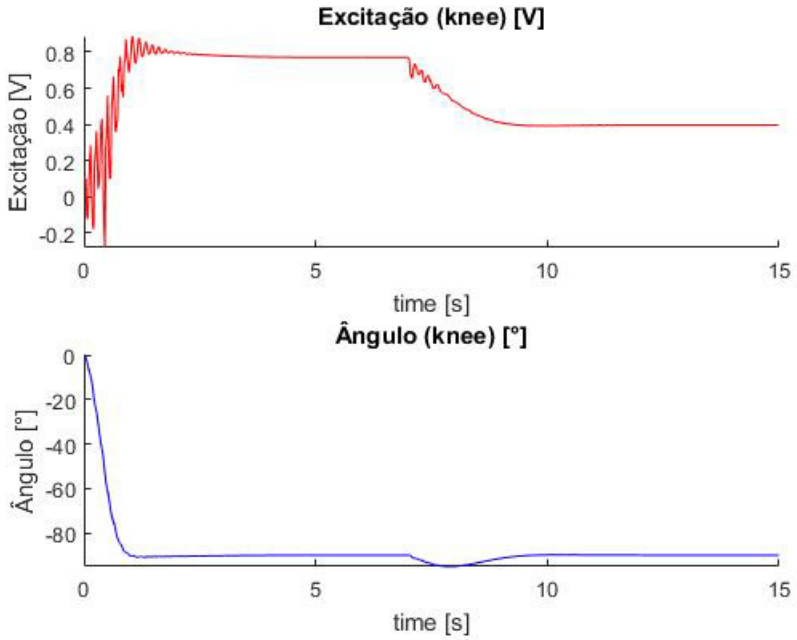


Fonte: O autor.

4.3 Controle simultâneo de flexão de joelho e flexão de quadril

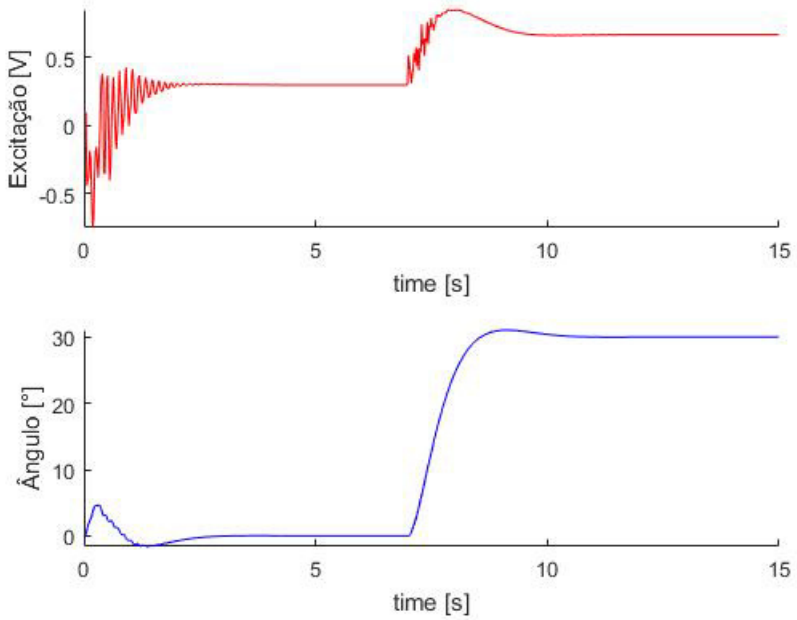
Após os testes individuais, simulou-se para ambos controladores atuando. Na simulação, primeiramente definiu-se a referência inicial para o ângulo de flexão do joelho em -90° e, após 7 segundos, a referência de flexão do quadril para 30°. Os resultados estão nas figuras 26 e 27. Link disponível para acessar vídeo do movimento: https://youtu.be/41ttjjMUxKE.

Figura 26 – Excitação e controle da flexão de joelho com controladores simultâneos



Fonte: O autor.

Figura 27 – Excitação e controle da flexão de quadril com controladores simultâneos

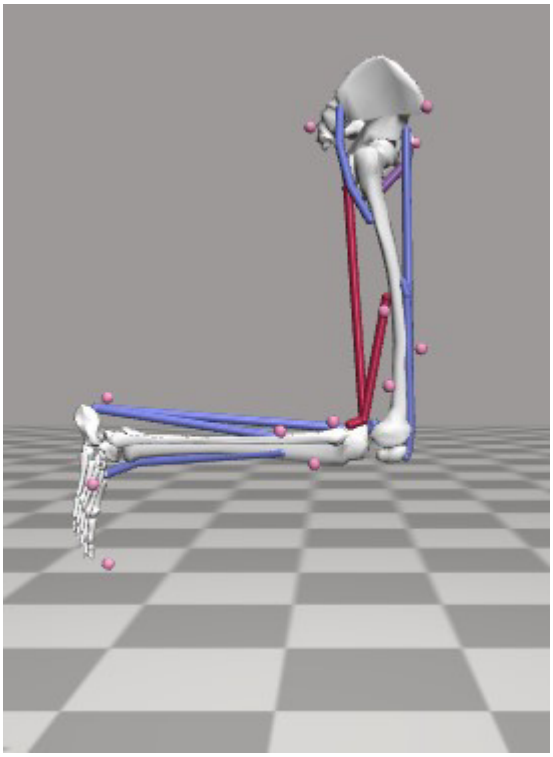


Fonte: O autor.

Em particular, chama-se atenção para o ângulo do joelho (figura 26) a partir de 7 segundos - momento em que o quadril passa a ser movido -, onde ocorre um momento de instabilidade do ângulo, mas logo retorna para a referência definida.

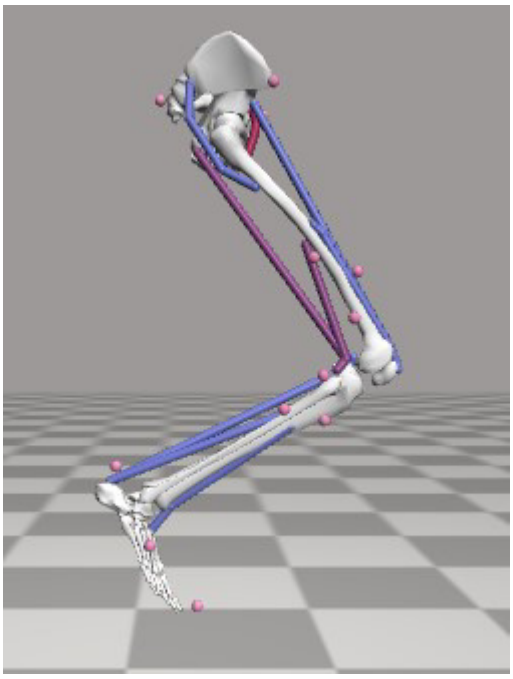
Com relação ao modelo no OpenSim, as figuras 28 e 29 representam, respectivamente, o modelo nos momentos 3 e 11 segundos.

Figura 28 – Modelo controlado no OpenSim para 3 segundos de simulação



Fonte: O autor.

Figura 29 – Modelo controlado no OpenSim para 11 segundos de simulação

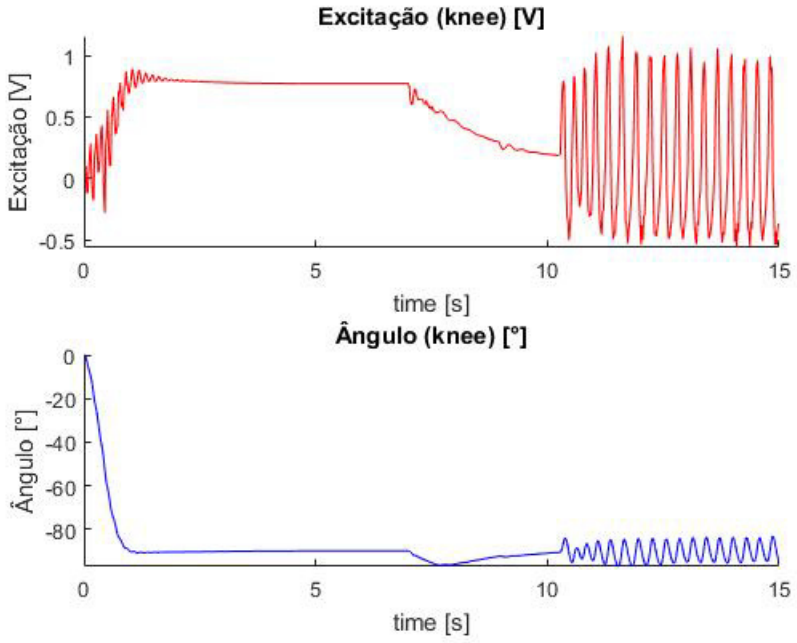


Fonte: O autor.

Dados os critérios de identificação do modelo, figura 18, para o caso onde a referência do quadril é imposta para atingir 45°, o controlador apresenta instabilidade. Portanto, é necessária a aplicação de técnicas de controle não-linear, adaptativo ou robusto que permitam manter a estabilidade do sistema em malha-fechada para todos os possíveis valores de ângulos da perna.

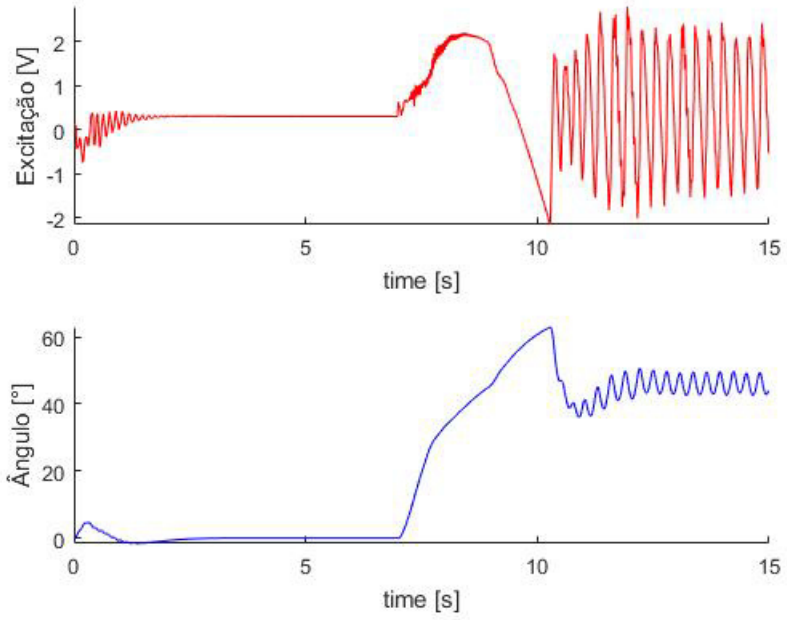
O resultado para essa instabilidade estão representados nas figuras 30 e 31.

Figura 30 – Excitação e controle da flexão de joelho em região de instabilidade



Fonte: O autor.

Figura 31 – Excitação e controle da flexão de quadril em região de instabilidade



Fonte: O autor.

5 CONCLUSÃO

Com o intuito de explorar os conceitos aprendidos na disciplina de Controle Digital aplicados junto ao aprendizado do software OpenSim ©, buscou-se implementar a modelagem dos movimentos de flexão de joelho e flexão de quadril.

O processo de estudo do modelo *leg6dof9musc* proporcionou momentos de descoberta e interdisciplinaridade, em especial as áreas de Biomecânica e Cinesiologia Estrutural, dada à natureza dos atuadores - músculos do corpo humano.

Após a identificação das respostas do modelo às excitações aplicadas, a metodologia empregadas visou substituir o método tradicional de linearização do sistema, o qual consiste na busca por equações que o regem. Adotando-se o método proposto por (LANDAU; ZITO, 2006), foi possível a familiarização com conceitos não estudados convencionalmente, como *PRBS* e MQNR. Com a identificação do modelo ARX para ambas movimentações propostas, realizou o projeto de dois controladores RST's para as coordenadas de estudo.

Dentre os desafios finais, a integração entre os softwares MATLAB © e OpenSim © mostrou-se pertinente. No entanto, graças à interface gráfica, os resultados foram facilmente observáveis, enriquecendo a interpretação dos gráficos e da resposta aos estímulos elétricos funcionais.

Por fim, dada à dinâmica linear do modelo identificado, há um intervalo de ângulos em que ambos controladores atuam satisfatoriamente. No entanto, testando-se ângulos fora desse escopo de identificação do sistema, encontraram-se instabilidades.

5.1 Trabalhos futuros

Ao longo do desenvolvimento do trabalho, diversas possibilidades se mostraram exploráveis. Aqui, listam-se sugestões de trabalhos:

- Aplicação de controladores com parâmetros lineares variantes, do inglês *Linear Parameter-varying* (LPV), devido à variação de parâmetros dos músculos em questão de torque, centro de massa e comprimento, observando-se um maior alcance do ângulo de flexão;
- Aplicação de técnica de *gain scheduling* com rede neural atuando como supervisor de pesos para os controladores atuantes.
- Aplicação da metodologia apresentada para outros modelos do software OpenSim ©, buscando atuar em mais músculos a fim de realizar movimentos completos.

REFERÊNCIAS

ALIBEJI, N.; KIRSCH, N.; SHARMA, N. Control of functional electrical stimulation in the presence of electromechanical and communication delays. In: . [S.l.: s.n.], 2013. p. 299–302.

BO, A. P.; MOURA, H. C. Elbow control using functional electrical stimulation: an experimental comparison of different control strategies. **IFAC-PapersOnLine**, v. 48, n. 20, p. 343–347, 2015. 9th IFAC Symposium on Biological and Medical Systems BMS 2015.

CRAGO, P. E.; PECKHAM, P. H.; THROPE, G. B. Modulation of muscle force by recruitment during intramuscular stimulation. **IEEE Transactions on Biomedical Engineering**, BME-27, n. 12, p. 679–684, 1980.

DOCUMENTATION, O. $_{l}eg69.2021.Disponvelem$: **<>**. Acesso em: 13 jan. 2022.

FLOYD, R. T. **Manual de Cinesiologia Estrutural**. [S.l.]: Manole, 2016. v. 19.

FODSTAD, H.; HARIZ, M. Electricity in the treatment of nervous system disease. **Operative Neuromodulation**, Springer, p. 11–19, 2007.

INC, M. **Generate z-plane grid of constant damping factors and natural frequencies**. 2021. Disponível em: <https://ww2.mathworks.cn/help/control/ref/zgrid.html>. Acesso em: 11 jan. 2022.

KOZAN, R. F. **Controle da Posição da Perna de Pessoas Hígidas Utilizando um Controlador PID**. Dissertação (Mestrado em Engenharia Elétrica) — Programa de Pós-Graduação em Engenharia Elétrica, Universidade Estadual Paulista “Júlio de Mesquita Filho”, Ilha Solteira, 2012.

LANDAU, I. D.; ZITO, G. **Digital Control Systems**: Design, identification and implementation. [*S. l.*]: Springer, 2006.

LIAO, Y.-W.; SCHEARER, E. M.; PERREAULT, E. J.; TRESCH, M. C.; LYNCH, K. M. Multi-muscle fes control of the human arm for interaction tasks—stabilizing with muscle co-contraction and postural adjustment: A simulation study. In: **2014 IEEE/RSJ International Conference on Intelligent Robots and Systems**. [S.l.: s.n.], 2014. p. 2134–2139.

LIMA, E.; (BRASIL), I. de Matematica Pura e A. **Álgebra linear**. [S.l.]: IMPA, 2008. (Coleção matemática universitária). ISBN 9788524400896.

NISE, N. S. **Engenharia de Sistemas de Controle**. [*S. l.*]: LTC - Livros Técnicos e Científicos Editora Ltda, Rio de Janeiro, 2013. v. 6.

PARASKEVOPOULOS, P. **Digital Control Systems**. Prentice Hall, 1996. ISBN 9780133418767. Disponível em: <https://books.google.com.br/books?id=y-lSAAAAMAAJ>.

SOUSA, A. de. **FES Gait**. 2019. Disponível em: <https://github.com/anacsousa1/opensim-fes-knee-control/blob/master/Tutorial.pdf>. Acesso em: 15 jan. 2022.

VETTE, A. H.; MASANI, K.; POPOVIC, M. R. Implementation of a physiologically identified pd feedback controller for regulating the active ankle torque during quiet stance. **IEEE Transactions on Neural Systems and Rehabilitation Engineering**, v. 15, n. 2, p. 235–243, 2007.

VETTE, A. H.; WU, N.; MASANI, K.; POPOVIC, M. R. Low-intensity functional electrical stimulation can increase multidirectional trunk stiffness in able-bodied individuals during sitting. **Medical Engineering Physics**, v. 37, n. 8, p. 777–782, 2015. ISSN 1350-4533.

ZHOU, B. H.; BARATTA, R.; SOLOMONOW, M.; OLIVIER, L.; NGUYEN, G.; D'AMBROSIA, R. Evaluation of isometric antagonist coactivation strategies of electrically stimulated muscles. **IEEE Transactions on Biomedical Engineering**, v. 43, n. 2, p. 150–160, 1996.

APÊNDICE A – CÓDIGOS-FONTES UTILIZADOS PARA APLICAÇÃO DO DEGRAU

Código-fonte 1 – Código de execução

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% by Ana de Sousa (anacsousa@lara.unb.br),Felipe Shimabuko
% and Antonio Padilha L. Bo
% Last update: February 2019
% Editado e adaptado por: Bruno Luiz Faustino e Renata Rodrigues
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; tic; close all; clc;
import org.opensim.modeling.*
% timing
global Tf
global Ts
global n_samples
global controlTime
global controlIteration
global excitation
global excitation1
global hipangle
global kneeangle
global excitationplot
global excitationhipplot
global hipangleplot
global kneeangleplot

%%%%

%% INITIAL CONFIGURATION

```

```
29 Tf = 15; %tempo final
30 Ts = 1/50; n_samples = floor(Tf/Ts); %Ts -> tempo de
      amostragem
31
32
33 controlTime = zeros(1,n_samples);
34 controlIteration = 1;
35
36 excitation=ones(1,n_samples);
37 excitation1=ones(1,n_samples);
38 kneeangle= zeros(1,n_samples);
39 hipangle = zeros(1,n_samples);
40
41 %%%%%%%%%
42
43 kneeangleplot= zeros(1,n_samples);
44 hipangleplot = zeros(1,n_samples);
45
46 %% INITIALIZE MODEL
47 disp('> Initialize simulation')
48 % open and init model
49
50 model = Model('leg6dof9musc.osim');
51
52 %% Declaracao dos musculos como atuadores
53 actuator = model.updForceSet().get('bifemlh_r');
54 bifemlhr = Thelen2003Muscle.safeDownCast(actuator);
55
56 bifemlhr.setDefaultActivation(0);
57
58 actuator = model.updForceSet().get('bifemsh_r');
59 bifemshr = Thelen2003Muscle.safeDownCast(actuator);
```

```
60
61 bifemshr.setDefaultActivation(0);
62
63 actuator = model.updForceSet().get('glut_max2_r');
64 glutmax2r = Thelen2003Muscle.safeDownCast(actuator);
65
66 glutmax2r.setDefaultActivation(0);
67
68 actuator = model.updForceSet().get('psoas_r');
69 psoasr = Thelen2003Muscle.safeDownCast(actuator);
70
71 psoasr.setDefaultActivation(0);
72
73 actuator = model.updForceSet().get('rect_fem_r');
74 rectfemr = Thelen2003Muscle.safeDownCast(actuator);
75
76 rectfemr.setDefaultActivation(0);
77
78 actuator = model.updForceSet().get('vas_int_r');
79 vasintr = Thelen2003Muscle.safeDownCast(actuator);
80
81 vasintr.setDefaultActivation(0);
82
83 actuator = model.updForceSet().get('med_gas_r');
84 medgasr = Thelen2003Muscle.safeDownCast(actuator);
85
86 medgasr.setDefaultActivation(0);
87
88 actuator = model.updForceSet().get('soleus_r');
89 soleusr = Thelen2003Muscle.safeDownCast(actuator);
90
91 soleusr.setDefaultActivation(0);
```

```
92
93
94  actuator = model.updForceSet().get('tib_ant_r');
95  tibantr = Thelen2003Muscle.safeDownCast(actuator);
96
97  tibantr.setDefaultActivation(0);
98
99
100 %% Declaracao de coordenadas
101
102 kneeangle(1)=model.getCoordinateSet().get('knee_angle_r').
        getDefaultValue();
103
104 valorangle =model.getCoordinateSet().get('knee_angle_r');
105
106 valorangle.setDefaultValue(0.0);
107
108 kneeangle(1)  = valorangle.getDefaultValue();
109
110 %valorangle.setDefaultLocked(true);
111
112
113 %% Definicao de excitacoes
114 excitation=0.5*excitation;
115 excitation1=0.5*excitation1;
116
117 %% Travar demais coordenadas em valores de default. Exceto
        hip_flexion_r, coordenada do quadril.
118
119 editableCoordSet_hip = model.updCoordinateSet().get( '
        hip_flexion_r' );
120
```

```
121 editableCoordSet_hip.setDefaultValue(0);
122
123 hipangle(1) = editableCoordSet_hip.getDefaultValue();
124
125 %editableCoordSet_hip.setDefaultLocked(true);
126
127
128 editableCoordSet = model.updCoordinateSet().get( 'pelvis_ty
    ' );
129
130 editableCoordSet.setDefaultValue(1.06);
131
132 editableCoordSet.setDefaultLocked(true);
133
134
135 editableCoordSet = model.updCoordinateSet().get( 'pelvis_tx
    ' );
136
137 editableCoordSet.setDefaultValue(0.058);
138
139 editableCoordSet.setDefaultLocked(true);
140
141
142 editableCoordSet = model.updCoordinateSet().get( '
    pelvis_tilt' );
143
144 editableCoordSet.setDefaultValue(pi*1.147/180);
145
146 editableCoordSet.setDefaultLocked(true);
147
148
```

```
149 editableCoordSet = model.updCoordinateSet().get( '
        ankle_angle_r' );
150
151 editableCoordSet.setDefaultValue(pi*5.578/180);
152
153 editableCoordSet.setDefaultLocked(true);
154
155
156 %% Configuracao de estados - requerimento do OpenSim
157 states = model.initSystem();
158 model.equilibrateMuscles(states);
159
160 % control function handle
161 controlFunctionHandle = @identificacao_control;
162
163 % get muscles and states names
164 %musclesNames = get_muscles_names(osimModel);
165 statesNames = get_states_names(model);
166
167 %% CONFIG & RUN SIMULATION
168 disp('> Run simulation')
169
170 % integrate plant using Matlab integration
171 timeSpan = [0 Tf]; integratorName = 'ode15s';
172 integratorOptions = odeset('AbsTol', 1E-5','MaxStep',.1*Ts)
        ;
173
174 % run simulation using function from Dynamic Walking
        example
175 motionData = IntegrateOpenSimPlant(model,
        controlFunctionHandle, timeSpan, ...
176     integratorName, integratorOptions);
```

```
177
178
179 motionData.data(:,[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15])=
       motionData.data(:,[1 2 9 3 10 4 11 5 12 6 13 7 14 8 15])
       ;
180
181 %% Atribuicao de valores para plotagem de graficos
182 excitationplot=excitation(1:controlIteration);
183 excitationhipplot = excitation1(1:controlIteration);
184 kneeangleplot=kneeangle(1:controlIteration);
185 hipangleplot = hipangle(1:controlIteration);
186 controlTimeplot=controlTime(1:controlIteration);
187
188 %Chamado da funcao para plot de graficos
189 identificacao_plot;
190
191 %% CREATE .STO FILE FOR VISUALIZATION, SAVE WORKSPACE AND
       SAVE FIGURE
192 Nsamples = length(motionData.data(:,1)); Nstates = length(
       statesNames);
193
194 % name of the file
195 str_name = 'identificacao_';
196 % str_name = strcat(str_name,'_Tf',num2str(Tf),'_P',num2str
       (flagPosition),'_Kmax',num2str(flagKneeTorque),...
197 %     '_s',num2str(flagKneeStart),'_r',num2str(
       flagKneeRange),'_L',num2str(flagLoad),'_M',num2str(
       flagMotor),...
198 %     '_STIM',num2str(flagSTIM),'_QHG',num2str(
       flagMusclesQUAD),num2str(flagMusclesHAMS),num2str(
       flagMusclesGLUT),...
```

```
199 %      '_C_',controlType, '_R', num2str(refVel1), '_FatRL',
        num2str(flagFAT_R),'_', num2str(flagFAT_L));
200
201     % create .STO
202     str = strjoin(statesNames,'\t');
203     header = ['identificacao_ teste3\nversion=1 \nnRows='
           num2str(Nsamples) ' \nnColumns=' num2str(Nstates+1)
           '\ninDegrees=no \nendheader \ntime   ' str '\n'];
204
205     fid = fopen(strcat('Results/',str_name,'.sto'),'wt');
206     fprintf(fid,header); fclose(fid);
207
208     fid = fopen(strcat('Results/',str_name,'.sto'),'a+');
209     for i = 1:Nsamples
210         fprintf(fid,'\t%f',motionData.data(i,:)); fprintf(
              fid,'\n');
211     end
212     fclose(fid);
213
214
215 %% The end!
216 fprintf('\n'); toc; disp('> THE END')
```

Código-fonte 2 – Função de controle

```
1 function modelControls = identificacao_control(osimModel,
      osimState)
2     import org.opensim.modeling.*;
3 global Ts
4 global n_samples
5 global controlTime
6 global controlIteration
```

```
7  global excitation
8  global excitation1
9  global kneeangle
10 global hipangle
11
12     %% GET INFO
13     modelControls = osimModel.updControls(osimState);
14     thisTime = osimState.getTime();
15
16     %% UPDATE CONTROL (only update if sampling period has
          passed)
17 if (thisTime - controlTime(controlIteration)) >= (Ts-.01*Ts
      )
18
19         controlIteration = controlIteration + 1;
20         controlTime(controlIteration) = thisTime;
21
22     %salva os atuais valores dos angulos
23     valorangle =osimModel.getCoordinateSet().get('
          knee_angle_r');
24     kneeangle(controlIteration)  = valorangle.getValue(
          osimState);
25
26     valorangle1 =osimModel.getCoordinateSet().get('
          hip_flexion_r');
27     hipangle(controlIteration)  = valorangle1.getValue(
          osimState);
28
29     %mudanca de referencia apos o codigo iniciar
30     if thisTime>7
31         %ref(controlIteration) = -1.57;
32         ref2(controlIteration) = 0.5236;
```

```
33      end
34
35  %% print simulation evolution
36  fprintf('%d/%d: excitacao: %f , angulo (knee): %f, angulo (
        hip): %f\n',controlIteration, n_samples, excitation(
        controlIteration), kneeangle(controlIteration), hipangle
        (controlIteration));
37
38  %% updade de excitacao nos musculos
39
40  osimModel.updActuators().get('bifemlh_r').addInControls(
        Vector(1,excitation(controlIteration)),modelControls);
41
42  osimModel.updActuators().get('bifemsh_r').addInControls(
        Vector(1,excitation(controlIteration)),modelControls);
43
44  %osimModel.updActuators().get('bifemlh_r').addInControls(
        Vector(1,0),modelControls);
45
46  %osimModel.updActuators().get('bifemsh_r').addInControls(
        Vector(1,0),modelControls);
47
48  osimModel.updActuators().get('glut_max2_r').addInControls(
        Vector(1,0),modelControls);
49
50  osimModel.updActuators().get('psoas_r').addInControls(
        Vector(1,excitation1(controlIteration)),modelControls);
51
52  %osimModel.updActuators().get('psoas_r').addInControls(
        Vector(1,0),modelControls);
53
54  osimModel.updActuators().get('rect_fem_r').addInControls(
```

```
    Vector(1,0),modelControls);
55
56  osimModel.updActuators().get('vas_int_r').addInControls(
    Vector(1,0),modelControls);
57
58  osimModel.updActuators().get('med_gas_r').addInControls(
    Vector(1,0),modelControls);
59
60  osimModel.updActuators().get('soleus_r').addInControls(
    Vector(1,0),modelControls);
61
62  osimModel.updActuators().get('tib_ant_r').addInControls(
    Vector(1,0),modelControls);
63
64
65  end
```

Código-fonte 3 – Código para gerar gráficos

```
1  %% plot
2  plot_title = strcat('Sinais de entrada e saida');
3
4  h(1) = figure('NumberTitle', 'off', 'Name', plot_title);
5
6  %% excitacao do joelho
7  ax1 = subplot(2,1,1);
8  hold on;
9  plot(controlTimeplot,excitationplot,'r');
10 hold off;
11 title('Excitacao (knee) [V]');
12 xlabel('time [s]'); ylabel('Excitacao [V]');xlim([0,15]);
   ylim([min(excitationplot),max(excitationplot)]);
```

```
13
14
15 %% angulo do joelho
16 ax2 = subplot(2,1,2); hold on;
17 plot(controlTimeplot,kneeangleplot*180/pi,'b');
18
19 hold off;
20 title('Angulo (knee)')
21 xlabel('time [s]'); ylabel('Angulo');  ylim([min(
      kneeangleplot)*180/pi,max(kneeangleplot)*180/pi]);
22 %
23 %% excitacao do quadril
24 % ax3 = subplot(2,1,1); hold on;
25 % plot(controlTimeplot,excitationhipplot,'r');
26 % hold off;
27 % %title('Excitacao (hip) [V]');
28 % xlabel('time [s]'); ylabel('Excitacao [V]');  ylim([min(
      excitationhipplot),max(excitationhipplot)]);
29 % %
30 % ax4 = subplot(2,1,2); hold on;
31 % plot(controlTimeplot,hipangleplot*180/pi,'b');
32 %
33
34 %% angulo do quadril
35 % hold off;
36 % %title('Angulo (hip)');
37 % xlabel('time [s]'); ylabel('Angulo');  ylim([min(
      hipangleplot)*180/pi,max(hipangleplot)*180/pi]);
38
39 linkaxes([ax1,ax2],'x')
```

APÊNDICE B – CÓDIGOS-FONTES UTILIZADOS PARA APLICAÇÃO DO PRBS

Código-fonte 4 – Código de execução

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% by Ana de Sousa (anacsousa@lara.unb.br),Felipe Shimabuko
% and Antonio Padilha L. Bo
% Last update: February 2019
% Editado e adaptado por: Bruno Luiz Faustino e Renata
    Rodrigues
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; tic; close all; clc;
import org.opensim.modeling.*
% timing
global Tf
global Ts
global n_samples
global controlTime
global controlIteration
global excitation
global excitation1
global hipangle
global kneeangle
global excitationplot
global excitationhipplot
global hipangleplot
global kneeangleplot
global contador
global base
global u
global entrada1

```

```
29 %%%%
30
31 %% INITIAL CONFIGURATION
32
33 Tf = 15; %tempo final
34 Ts = 1/50; n_samples = floor(Tf/Ts); %Ts -> tempo de
      amostragem
35
36 %% Declaracao de vetores
37 u = ones(1,n_samples);
38 entrada1=zeros(1,n_samples);
39
40 contador=0;
41
42 base= 0.5;
43
44 controlTime = zeros(1,n_samples);
45 controlIteration = 1;
46
47 excitation=ones(1,n_samples);
48 excitation1=ones(1,n_samples);
49 kneeangle= zeros(1,n_samples);
50 hipangle = zeros(1,n_samples);
51
52 %%%%%%%%%
53
54 kneeangleplot= zeros(1,n_samples);
55 hipangleplot = zeros(1,n_samples);
56
57 %% INITIALIZE MODEL
58 disp('> Initialize simulation')
59 % open and init model
```

```
60 sim('Gerar PRBS/PRBS1.slx') %abrir simulink com PRBS
61 model = Model('leg6dof9musc.osim');
62
63 %% Declaracao dos musculos como atuadores
64 actuator = model.updForceSet().get('bifemlh_r');
65 bifemlhr = Thelen2003Muscle.safeDownCast(actuator);
66
67 bifemlhr.setDefaultActivation(0);
68
69 actuator = model.updForceSet().get('bifemsh_r');
70 bifemshr = Thelen2003Muscle.safeDownCast(actuator);
71
72 bifemshr.setDefaultActivation(0);
73
74 actuator = model.updForceSet().get('glut_max2_r');
75 glutmax2r = Thelen2003Muscle.safeDownCast(actuator);
76
77 glutmax2r.setDefaultActivation(0);
78
79 actuator = model.updForceSet().get('psoas_r');
80 psoasr = Thelen2003Muscle.safeDownCast(actuator);
81
82 psoasr.setDefaultActivation(0);
83
84 actuator = model.updForceSet().get('rect_fem_r');
85 rectfemr = Thelen2003Muscle.safeDownCast(actuator);
86
87 rectfemr.setDefaultActivation(0);
88
89 actuator = model.updForceSet().get('vas_int_r');
90 vasintr = Thelen2003Muscle.safeDownCast(actuator);
91
```

```
92  vasintr.setDefaultActivation(0);
93
94  actuator = model.updForceSet().get('med_gas_r');
95  medgasr = Thelen2003Muscle.safeDownCast(actuator);
96
97  medgasr.setDefaultActivation(0);
98
99  actuator = model.updForceSet().get('soleus_r');
100 soleusr = Thelen2003Muscle.safeDownCast(actuator);
101
102 soleusr.setDefaultActivation(0);
103
104
105 actuator = model.updForceSet().get('tib_ant_r');
106 tibantr = Thelen2003Muscle.safeDownCast(actuator);
107
108 tibantr.setDefaultActivation(0);
109
110
111 %% Declaracao de coordenadas
112
113 kneeangle(1)=model.getCoordinateSet().get('knee_angle_r').
       getDefaultValue();
114
115 valorangle =model.getCoordinateSet().get('knee_angle_r');
116
117 valorangle.setDefaultValue(0.0);
118
119 kneeangle(1)  = valorangle.getDefaultValue();
120
121 %valorangle.setDefaultLocked(true);
122
```

```
123
124 %% Travar demais coordenadas em valores de default. Exceto
       hip_flexion_r, coordenada do quadril.
125
126 editableCoordSet_hip = model.updCoordinateSet().get( '
       hip_flexion_r' );
127
128 editableCoordSet_hip.setDefaultValue(0);
129
130 hipangle(1) = editableCoordSet_hip.getDefaultValue();
131
132 %editableCoordSet_hip.setDefaultLocked(true);
133
134
135 editableCoordSet = model.updCoordinateSet().get( 'pelvis_ty
       ' );
136
137 editableCoordSet.setDefaultValue(1.06);
138
139 editableCoordSet.setDefaultLocked(true);
140
141
142 editableCoordSet = model.updCoordinateSet().get( 'pelvis_tx
       ' );
143
144 editableCoordSet.setDefaultValue(0.058);
145
146 editableCoordSet.setDefaultLocked(true);
147
148
149 editableCoordSet = model.updCoordinateSet().get( '
       pelvis_tilt' );
```

```
150
151 editableCoordSet.setDefaultValue(pi*1.147/180);
152
153 editableCoordSet.setDefaultLocked(true);
154
155
156 editableCoordSet = model.updCoordinateSet().get( '
       ankle_angle_r' );
157
158 editableCoordSet.setDefaultValue(pi*5.578/180);
159
160 editableCoordSet.setDefaultLocked(true);
161
162
163 %% Configuracao de estados - requerimento do OpenSim
164 states = model.initSystem();
165 model.equilibrateMuscles(states);
166
167 % control function handle
168 controlFunctionHandle = @identificacao_control;
169
170 % get muscles and states names
171 %musclesNames = get_muscles_names(osimModel);
172 statesNames = get_states_names(model);
173
174 %% CONFIG & RUN SIMULATION
175 disp('> Run simulation')
176
177 % integrate plant using Matlab integration
178 timeSpan = [0 Tf]; integratorName = 'ode15s';
179 integratorOptions = odeset('AbsTol', 1E-5','MaxStep',.1*Ts)
       ;
```

```
180
181 % run simulation using function from Dynamic Walking
       example
182 motionData = IntegrateOpenSimPlant(model,
       controlFunctionHandle, timeSpan, ...
183     integratorName, integratorOptions);
184
185
186 motionData.data(:,[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15])=
       motionData.data(:,[1 2 9 3 10 4 11 5 12 6 13 7 14 8 15])
       ;
187
188 %% Atribuicao de valores para plotagem de graficos
189 excitationplot=excitation(1:controlIteration);
190 excitationhipplot = excitation1(1:controlIteration);
191 kneeangleplot=kneeangle(1:controlIteration);
192 hipangleplot = hipangle(1:controlIteration);
193 controlTimeplot=controlTime(1:controlIteration);
194
195 %Chamado da funcao para plot de graficos
196 identificacao_plot;
197
198 %% CREATE .STO FILE FOR VISUALIZATION, SAVE WORKSPACE AND
       SAVE FIGURE
199 Nsamples = length(motionData.data(:,1)); Nstates = length(
       statesNames);
200
201 % name of the file
202 str_name = 'identificacao_';
203 % str_name = strcat(str_name,'_Tf',num2str(Tf),'_P',num2str
       (flagPosition),'_Kmax',num2str(flagKneeTorque),...
```

```
204 %     '_s',num2str(flagKneeStart),'_r',num2str(
    flagKneeRange),'_L',num2str(flagLoad),'_M',num2str(
    flagMotor),...
205 %     '_STIM',num2str(flagSTIM),'_QHG',num2str(
    flagMusclesQUAD),num2str(flagMusclesHAMS),num2str(
    flagMusclesGLUT),...
206 %     '_C_',controlType, '_R', num2str(refVel1), '_FatRL',
    num2str(flagFAT_R),'_', num2str(flagFAT_L));
207
208     % create .STO
209     str = strjoin(statesNames,'\t');
210     header = ['identificacao_ teste3\nversion=1 \nnRows='
        num2str(Nsamples) ' \nnColumns=' num2str(Nstates+1)
        '\ninDegrees=no \nendheader \ntime  ' str '\n'];
211
212     fid = fopen(strcat('Results/',str_name,'.sto'),'wt');
213     fprintf(fid,header); fclose(fid);
214
215     fid = fopen(strcat('Results/',str_name,'.sto'),'a+');
216     for i = 1:Nsamples
217         fprintf(fid,'\t%f',motionData.data(i,:)); fprintf(
            fid,'\n');
218     end
219     fclose(fid);
220
221
222 %% The end!
223 fprintf('\n'); toc; disp('> THE END')
```

Código-fonte 5 – Função de controle

```
1 function modelControls = identificacao_control(osimModel,
```

```
   osimState)
    import org.opensim.modeling.*;
global Ts
global n_samples
global controlTime
global controlIteration
global excitation
global excitation1
global kneeangle
global hipangle

    %% GET INFO
    modelControls = osimModel.updControls(osimState);
    thisTime = osimState.getTime();

    %% UPDATE CONTROL (only update if sampling period has
       passed)
if (thisTime - controlTime(controlIteration)) >= (Ts-.01*Ts
   )

        controlIteration = controlIteration + 1;
        controlTime(controlIteration) = thisTime;

    if thisTime>5 %aplica PRBS na excitacao

    excitation(controlIteration)=base+u(1+contador);
    excitation1(controlIteration)=base+u(1+contador);
    contador=contador+1;

    end

    %salva os atuais valores dos angulos
```

```
31      valorangle =osimModel.getCoordinateSet().get('
           knee_angle_r');
32      kneeangle(controlIteration)  = valorangle.getValue(
           osimState);
33
34      valorangle1 =osimModel.getCoordinateSet().get('
           hip_flexion_r');
35      hipangle(controlIteration)  = valorangle1.getValue(
           osimState);
36
37
38
39  %% print simulation evolution
40  fprintf('%d/%d: excitacao: %f , angulo (knee): %f, angulo (
       hip): %f\n',controlIteration, n_samples, excitation(
       controlIteration), kneeangle(controlIteration), hipangle
       (controlIteration));
41
42  %% updade de excitacao nos musculos
43
44  osimModel.updActuators().get('bifemlh_r').addInControls(
       Vector(1,excitation(controlIteration)),modelControls);
45
46  osimModel.updActuators().get('bifemsh_r').addInControls(
       Vector(1,excitation(controlIteration)),modelControls);
47
48  %osimModel.updActuators().get('bifemlh_r').addInControls(
       Vector(1,0),modelControls);
49
50  %osimModel.updActuators().get('bifemsh_r').addInControls(
       Vector(1,0),modelControls);
51
```

```
52 osimModel.updActuators().get('glut_max2_r').addInControls(
       Vector(1,0),modelControls);
53
54 osimModel.updActuators().get('psoas_r').addInControls(
       Vector(1,excitation1(controlIteration)),modelControls);
55
56 %osimModel.updActuators().get('psoas_r').addInControls(
       Vector(1,0),modelControls);
57
58 osimModel.updActuators().get('rect_fem_r').addInControls(
       Vector(1,0),modelControls);
59
60 osimModel.updActuators().get('vas_int_r').addInControls(
       Vector(1,0),modelControls);
61
62 osimModel.updActuators().get('med_gas_r').addInControls(
       Vector(1,0),modelControls);
63
64 osimModel.updActuators().get('soleus_r').addInControls(
       Vector(1,0),modelControls);
65
66 osimModel.updActuators().get('tib_ant_r').addInControls(
       Vector(1,0),modelControls);
67
68
69 end
```

Código-fonte 6 – Código para gerar gráficos

```
1 %% plot
2 plot_title = strcat('Sinais de entrada e saida');
3
```

```
4  h(1) = figure('NumberTitle', 'off', 'Name', plot_title);
5
6  %% excitacao do joelho
7  ax1 = subplot(2,1,1);
8  hold on;
9  plot(controlTimeplot,excitationplot,'r');
10 hold off;
11 title('Excitacao (knee) [V]');
12 xlabel('time [s]'); ylabel('Excitacao [V]');xlim([0,15]);
       ylim([min(excitationplot),max(excitationplot)]);
13
14
15 %% angulo do joelho
16 ax2 = subplot(2,1,2); hold on;
17 plot(controlTimeplot,kneeangleplot*180/pi,'b');
18
19 hold off;
20 title('Angulo (knee)')
21 xlabel('time [s]'); ylabel('Angulo');  ylim([min(
       kneeangleplot)*180/pi,max(kneeangleplot)*180/pi]);
22 %
23 %% excitacao do quadril
24 % ax3 = subplot(2,1,1); hold on;
25 % plot(controlTimeplot,excitationhipplot,'r');
26 % hold off;
27 % %title('Excitacao (hip) [V]');
28 % xlabel('time [s]'); ylabel('Excitacao [V]');  ylim([min(
       excitationhipplot),max(excitationhipplot)]);
29 % %
30 % ax4 = subplot(2,1,2); hold on;
31 % plot(controlTimeplot,hipangleplot*180/pi,'b');
32 %
```

```
33
34 %% angulo do quadril
35 % hold off;
36 % %title('Angulo (hip)');
37 % xlabel('time [s]'); ylabel('Angulo');  ylim([min(
      hipangleplot)*180/pi,max(hipangleplot)*180/pi]);
38
39 linkaxes([ax1,ax2],'x')
```

APÊNDICE C – CÓDIGOS-FONTES UTILIZADOS PARA IDENTIFICAÇÃO DO MODELO ARX E CONTROLADOR RST

Código-fonte 7 – Código de identificação do modelo ARX

```
1
2  %% Elimina amostras com poucas correlacoes e a componente
       DC
3  y = kneeangle(252:end)-mean(kneeangle);
4  u = excitation(252:end)-mean(excitation);
5
6  %% Escolha da ordem do modelo
7  nb = 1;
8  na = 2;
9  d = 0;
10
11 N=length(y);
12 %% Aplicacao do MQNR
13
14 ci=max(nb+d,na);
15
16 phi=zeros(N-ci,nb+na);
17
18 for i=3:N
19     phi(i-2,:) = [-(y(i-1)) -(y(i-2)) u(i-1)];
20 end
21
22 if det(phi'*phi)==0
23     error('A matriz phiT*phi e singular','A');
24 end
25 Y = y(3:N);
26 teta = inv(phi'*phi)*phi'*Y';
27 den = [1 teta(1) teta(2)];
```

```
num = [0 teta(3)];
F_d = tf(num,den,0.02,'Variable','z^-1')
```

Código-fonte 8 – Código de projeto do controlador RST

```
%% Codigo fornecido por Prof. Dr. Fabricio Nogueira na
   disciplina de Controle Discreto
% Adaptado por Bruno Luiz Faustino

%%

%clear; clc;
Ts = 0.02;  % sampling time
% --------    Open-loop Model --------------------
% [Wo,dmp] = omega_dmp(2,50);
% G = tf(2*[Wo^2],[1  2*Wo*dmp Wo^2]);

% --------   Closed loop performance specification
   ----------
Tr = 0.5;
MaxOvershoot = 0.0;
[Pd] = CL_Perf(Tr,MaxOvershoot,Ts);

% --------    Open-loop Discretization --------------------
% Gd = c2d(G,Ts,'zoh');   % Discretized plant model
% [B,A]=tfdata(Gd,'v');   % Polynomials Ba nd A

% Adiciona modelo ARX

B = [0 7.417418988893712e-04 0.013897867683724];
A = [1 -1.868747540468956 0.878632698201897];

```

```
26 d=0;                          % Transport-delay
27 B = [zeros(1,d) B];           % B augmentation
28
29 % --------        PP solution ---------------
30 [R,S,T,P]=PP(B,A,d,Pd,Ts)
31
32 % Closed-loop step response evaluation
33 Hz=filt(B,A,Ts);
34 CS=filt(1,S,Ts);
35 CR=filt(R,1,Ts);
36 Hrd=CS*Hz;
37
38 Hmf=T*feedback(Hrd,CR);
39 step(Hmf)
40 hold on;
41 step(Hz)
42
43 %% validacao do modelo
44 % Dados para identificacao (60% dos dados sao util. p/
      identificar)
45 yi = y(1:floor(length(y)*0.6));
46 ui = u(1:floor(length(y)*0.6));
47
48 % Dados para validacao (40% dos dados sao util. p/ validar)
49 yv = y(floor(length(y)*0.6)+1:end);
50 uv = u(floor(length(y)*0.6)+1:end);
51
52 N = length(yv);
53 ye = zeros(N,1);
54
55 % Obtencao dos valores estimados
56 for i=ci+1:N
```

```
57      phi = [-(ye(i-1:-1:i-na))' uv(i-1-d:-1:i-nb-d)'];
58      ye(i) = phi*teta;
59  end
60
61  t=[0:Ts:(N-1)*Ts];
62  subplot(1,2,2)
63  figure('name','Modelo real e modelo estimado');
64  plot(t,yv,'r')
65  hold on;
66  plot(t,ye,'b')
67  plot(t,uv,'k--')
68  title('Validacao do modelo');
69
70  % Indice de desempenho, quanto mais proximo de 100 melhor
71  id = max(1-(var(yv -ye))/(var(yv)))*100
```

Código-fonte 9 – Função PP

```
1
2   function [R,S,T,P] = PP(B,A,d,Pd,Ts)
3   %function [R,S,T] = PP(B,A,Tr,MaxOvershoot)
4
5   % plant model
6
7   nA=length(A)-1;
8   nB=length(B)-1;
9
10  % Specification of fixed parts Hs and Hs of polynomials R
        and S, respectively:
11  %Hs=1;
12  Hs=[1 -1];  % Hs - fixed part of S:
13  Hr=1;       % Hr - fixed part of R:
```

```
14
15 % Building of extended plant polynomials
16 nHs = length(Hs)-1;
17 nHr = length(Hr)-1;
18
19 BB=conv(B,Hr);
20 AA=conv(A,Hs);
21
22 % Definition of the degree of AA and BB
23 nBB=length(BB)-1-d;
24 nAA=length(AA)-1;
25
26 % Matriz formada pelos elementos do polinomio A:
27 MA = zeros((nAA+nBB+d),(nBB+d));
28 for i=1:(nBB+d)
29     MA(i:nAA+i,i)=AA';
30 end
31
32 % Matriz formada pelos elementos do polinomio B:
33 MB = zeros((nAA+nBB+d),(nAA));
34
35 for i=1:(nAA)
36 %    MB(i:nBB+i,i)=BB(d+1:nBB+1+d)';
37     MB(i:nBB+i+d,i)=BB';
38 end
39
40 % Building matrix M:
41 MM = [MA MB];
42 %
43 % Specification of Auxiliary poles :
44 %display(sprintf('Forneca os %d Polos Auxiliares de malha
   fechada desejados:', (nAA+nBB+d-1-2)));
```

```
%pdf = input('Polos Auxiliares:');
%Pf = [poly(pdf)];
polosaux=nAA+nBB+d-1-2
%Pf = [poly([0.0 0.1])];
Pf = [poly([0.0 0.1])];
%Polinomio caracteristico desejado:
P = conv(Pd,Pf);
%
X = inv(MM)*P';
%
nS = nBB+d-1;
nR = nAA-1;

So=X(1:nS+1);
Ro=X(nS+2:length(X));

R=conv(Hr,Ro)';
S=conv(Hs,So)';
% T=P/sum(B);
T=sum(R);
```

Código-fonte 10 – Funcao CL$_{perf}$

```
function [Pd] = CL_Perf(Tr,MaxOvershoot,Ts)

% --------   Closed loop performance specification
   ----------
[Wo,Qsi]=omega_dmp(Tr,MaxOvershoot);
Hc = tf([Wo^2],[1  2*Wo*Qsi Wo^2]);
%Ts=0.04;
% --------   Sampling rate definition  ----------
% Sample time ( 0.25 < WoTs < 1.5 ) - minimal value with 1
```

```
   decimal digit
%Ts = round((0.25/Wo)*10)/10;

% --------   Discretization of desired polynomial
   ----------
Hd = c2d(Hc,Ts,'zoh');  % Discretized desired CL polynomial
[Bd,Pd]=tfdata(Hd,'v');
%Polos_d = roots(Ad);
%Pd = poly([Polos_d(1) Polos_d(2)]);
```

APÊNDICE D – CÓDIGOS-FONTES UTILIZADOS PARA IMPLEMENTAÇÃO DO CONTROLADOR

Código-fonte 11 – Código de execução do controle

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% by Ana de Sousa (anacsousa@lara.unb.br),Felipe Shimabuko
% and Antonio Padilha L. Bo
% Last update: February 2019
% Editado e adaptado por: Bruno Luiz Faustino e Renata Rodrigues
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
clear; tic; close all; clc;
import org.opensim.modeling.*
% timing
global Tf
global Ts
global n_samples
global controlTime
global controlIteration
global excitation
global excitation1
global hipangle
global kneeangle
global excitationplot
global excitationhipplot
global hipangleplot
global kneeangleplot
global R
global S
global T
global R_hip
```

```
28 global S_hip
29 global T_hip
30 global u
31 global ref
32 global ref2
33 global entrada1
34
35 %%%%
36
37 %% INITIAL CONFIGURATION
38
39 Tf = 15; %tempo final
40 Ts = 1/50; n_samples = floor(Tf/Ts); %Ts -> tempo de
      amostragem
41
42 %% RST -> Flexao de joelho
43 R = [-28.920231942107243 56.302760209636070
      -27.491992409069680];
44 S = [-0.193273438780072 -0.806726561219928];
45 T = -0.109464141540855;
46 %% RST -> Flexao de quadril
47 R_hip = [1.300897808672957e+02 -2.483461920980293e+02
      1.186745508479820e+02];
48 S_hip = [-0.045814063058777 -0.954185936941223];
49 T_hip = 0.418139617248329;
50 %% Declaracao de vetores
51 u = ones(1,n_samples);
52 entrada1=zeros(1,n_samples);
53 ref=ones(1,n_samples);
54 ref2 = ones(1,n_samples);
55
56 controlTime = zeros(1,n_samples);
```

```
57 controlIteration = 1;
58
59 excitation=ones(1,n_samples);
60 excitation1=ones(1,n_samples);
61 kneeangle= zeros(1,n_samples);
62 hipangle = zeros(1,n_samples);
63
64 %%%%%%%%%
65
66 kneeangleplot= zeros(1,n_samples);
67 hipangleplot = zeros(1,n_samples);
68
69 %% INITIALIZE MODEL
70 disp('> Initialize simulation')
71 % open and init model
72
73 model = Model('leg6dof9musc.osim');
74
75 %% Declaracao dos musculos como atuadores
76 actuator = model.updForceSet().get('bifemlh_r');
77 bifemlhr = Thelen2003Muscle.safeDownCast(actuator);
78
79 bifemlhr.setDefaultActivation(0);
80
81 actuator = model.updForceSet().get('bifemsh_r');
82 bifemshr = Thelen2003Muscle.safeDownCast(actuator);
83
84 bifemshr.setDefaultActivation(0);
85
86 actuator = model.updForceSet().get('glut_max2_r');
87 glutmax2r = Thelen2003Muscle.safeDownCast(actuator);
88
```

```
89  glutmax2r.setDefaultActivation(0);
90
91  actuator = model.updForceSet().get('psoas_r');
92  psoasr = Thelen2003Muscle.safeDownCast(actuator);
93
94  psoasr.setDefaultActivation(0);
95
96  actuator = model.updForceSet().get('rect_fem_r');
97  rectfemr = Thelen2003Muscle.safeDownCast(actuator);
98
99  rectfemr.setDefaultActivation(0);
100
101 actuator = model.updForceSet().get('vas_int_r');
102 vasintr = Thelen2003Muscle.safeDownCast(actuator);
103
104 vasintr.setDefaultActivation(0);
105
106 actuator = model.updForceSet().get('med_gas_r');
107 medgasr = Thelen2003Muscle.safeDownCast(actuator);
108
109 medgasr.setDefaultActivation(0);
110
111 actuator = model.updForceSet().get('soleus_r');
112 soleusr = Thelen2003Muscle.safeDownCast(actuator);
113
114 soleusr.setDefaultActivation(0);
115
116
117 actuator = model.updForceSet().get('tib_ant_r');
118 tibantr = Thelen2003Muscle.safeDownCast(actuator);
119
120 tibantr.setDefaultActivation(0);
```

```
121
122
123 %% Declaracao de coordenadas
124
125 kneeangle(1)=model.getCoordinateSet().get('knee_angle_r').
       getDefaultValue();
126
127 valorangle =model.getCoordinateSet().get('knee_angle_r');
128
129 valorangle.setDefaultValue(0.0);
130
131 kneeangle(1)  = valorangle.getDefaultValue();
132
133 %valorangle.setDefaultLocked(true);
134
135 %% Definicao de referencias
136 ref = -1.57*ref;
137 ref2 = 0*ref2;
138
139 %% Definicao de excitacoes
140 excitation=0.1*excitation;
141 excitation1=0.1*excitation1;
142
143 %% Travar demais coordenadas em valores de default. Exceto
       hip_flexion_r, coordenada do quadril.
144
145 editableCoordSet_hip = model.updCoordinateSet().get( '
       hip_flexion_r' );
146
147 editableCoordSet_hip.setDefaultValue(0);
148
149 hipangle(1) = editableCoordSet_hip.getDefaultValue();
```

```
150
151 %editableCoordSet_hip.setDefaultLocked(true);
152
153
154 editableCoordSet = model.updCoordinateSet().get( 'pelvis_ty
    ' );
155
156 editableCoordSet.setDefaultValue(1.06);
157
158 editableCoordSet.setDefaultLocked(true);
159
160
161 editableCoordSet = model.updCoordinateSet().get( 'pelvis_tx
    ' );
162
163 editableCoordSet.setDefaultValue(0.058);
164
165 editableCoordSet.setDefaultLocked(true);
166
167
168 editableCoordSet = model.updCoordinateSet().get( '
    pelvis_tilt' );
169
170 editableCoordSet.setDefaultValue(pi*1.147/180);
171
172 editableCoordSet.setDefaultLocked(true);
173
174
175 editableCoordSet = model.updCoordinateSet().get( '
    ankle_angle_r' );
176
177 editableCoordSet.setDefaultValue(pi*5.578/180);
```

```
178
179 editableCoordSet.setDefaultLocked(true);
180
181
182 %% Configuracao de estados - requerimento do OpenSim
183 states = model.initSystem();
184 model.equilibrateMuscles(states);
185
186 % control function handle
187 controlFunctionHandle = @identificacao_control;
188
189 % get muscles and states names
190 %musclesNames = get_muscles_names(osimModel);
191 statesNames = get_states_names(model);
192
193 %% CONFIG & RUN SIMULATION
194 disp('> Run simulation')
195
196 % integrate plant using Matlab integration
197 timeSpan = [0 Tf]; integratorName = 'ode15s';
198 integratorOptions = odeset('AbsTol', 1E-5,'MaxStep',.1*Ts)
    ;
199
200 % run simulation using function from Dynamic Walking
    example
201 motionData = IntegrateOpenSimPlant(model,
    controlFunctionHandle, timeSpan, ...
202     integratorName, integratorOptions);
203
204
205 motionData.data(:,[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15])=
    motionData.data(:,[1 2 9 3 10 4 11 5 12 6 13 7 14 8 15])
```

```
    ;
206
207 %% Atribuicao de valores para plotagem de graficos
208 excitationplot=excitation(1:controlIteration);
209 excitationhipplot = excitation1(1:controlIteration);
210 kneeangleplot=kneeangle(1:controlIteration);
211 hipangleplot = hipangle(1:controlIteration);
212 controlTimeplot=controlTime(1:controlIteration);
213
214 %Chamado da funcao para plot de graficos
215 identificacao_plot;
216
217 %% CREATE .STO FILE FOR VISUALIZATION, SAVE WORKSPACE AND
    SAVE FIGURE
218 Nsamples = length(motionData.data(:,1)); Nstates = length(
    statesNames);
219
220 % name of the file
221 str_name = 'identificacao_';
222 % str_name = strcat(str_name,'_Tf',num2str(Tf),'_P',num2str
    (flagPosition),'_Kmax',num2str(flagKneeTorque),...
223 %     '_s',num2str(flagKneeStart),'_r',num2str(
    flagKneeRange),'_L',num2str(flagLoad),'_M',num2str(
    flagMotor),...
224 %     '_STIM',num2str(flagSTIM),'_QHG',num2str(
    flagMusclesQUAD),num2str(flagMusclesHAMS),num2str(
    flagMusclesGLUT),...
225 %     '_C_',controlType, '_R', num2str(refVel1), '_FatRL',
    num2str(flagFAT_R),'_', num2str(flagFAT_L));
226
227     % create .STO
228     str = strjoin(statesNames,'\t');
```

```
229     header = ['identificacao_ teste3\nversion=1 \nnRows='
           num2str(Nsamples) ' \nnColumns=' num2str(Nstates+1)
           '\ninDegrees=no \nendheader \ntime   ' str '\n'];
230
231     fid = fopen(strcat('Results/',str_name,'.sto'),'wt');
232     fprintf(fid,header); fclose(fid);
233
234     fid = fopen(strcat('Results/',str_name,'.sto'),'a+');
235     for i = 1:Nsamples
236         fprintf(fid,'\t%f',motionData.data(i,:)); fprintf(
               fid,'\n');
237     end
238     fclose(fid);
239
240
241 %% The end!
242 fprintf('\n'); toc; disp('> THE END')
```

Código-fonte 12 – Função de controle

```
1  function modelControls = identificacao_control(osimModel,
       osimState)
2      import org.opensim.modeling.*;
3  global Ts
4  global n_samples
5  global controlTime
6  global controlIteration
7  global excitation
8  global excitation1
9  global kneeangle
10 global ref
11 global R
```

```
12 global S
13 global T
14 global ref2
15 global R_hip
16 global S_hip
17 global T_hip
18 global hipangle
19
20     %% GET INFO
21     modelControls = osimModel.updControls(osimState);
22     thisTime = osimState.getTime();
23
24     %% UPDATE CONTROL (only update if sampling period has
          passed)
25 if (thisTime - controlTime(controlIteration)) >= (Ts-.01*Ts
      )
26
27         controlIteration = controlIteration + 1;
28         controlTime(controlIteration) = thisTime;
29
30     %salva os atuais valores dos angulos
31     valorangle =osimModel.getCoordinateSet().get('
          knee_angle_r');
32     kneeangle(controlIteration)  = valorangle.getValue(
          osimState);
33
34     valorangle1 =osimModel.getCoordinateSet().get('
          hip_flexion_r');
35     hipangle(controlIteration)  = valorangle1.getValue(
          osimState);
36
37     %mudanca de referencia apos o codigo iniciar
```

```
38      if thisTime>7
39          %ref(controlIteration) = -1.57;
40          ref2(controlIteration) = 0.5236;
41      end
42
43  %% print simulation evolution
44  fprintf('%d/%d: excitacao: %f , angulo (knee): %f, angulo (
        hip): %f\n',controlIteration, n_samples, excitation(
        controlIteration), kneeangle(controlIteration), hipangle
        (controlIteration));
45
46
47  %% lei de controle -> atua a partir da terceira iteracao,
        pois os primeiros 2 valores sao predefinidos
48
49      if controlIteration > 3
50
51      Y = [kneeangle(controlIteration) kneeangle(
            controlIteration-1) kneeangle(controlIteration-2)]';
52      U = [excitation(controlIteration-1) excitation(
            controlIteration-2)]';
53
54      Y1 = [hipangle(controlIteration) hipangle(
            controlIteration-1) hipangle(controlIteration-2)]';
55      U1 = [excitation1(controlIteration-1) excitation1(
            controlIteration-2)]';
56
57       excitation(controlIteration) = T*ref(controlIteration)
             - R*Y - S*U;
58       excitation1(controlIteration) = T_hip*ref2(
            controlIteration) - R_hip*Y1 - S_hip*U1;
59
```

```
60      end
61
62   end
63
64
65  %% updade de excitacao nos musculos
66
67  osimModel.updActuators().get('bifemlh_r').addInControls(
        Vector(1,excitation(controlIteration)),modelControls);
68
69  osimModel.updActuators().get('bifemsh_r').addInControls(
        Vector(1,excitation(controlIteration)),modelControls);
70
71  %osimModel.updActuators().get('bifemlh_r').addInControls(
        Vector(1,0),modelControls);
72
73  %osimModel.updActuators().get('bifemsh_r').addInControls(
        Vector(1,0),modelControls);
74
75  osimModel.updActuators().get('glut_max2_r').addInControls(
        Vector(1,0),modelControls);
76
77  osimModel.updActuators().get('psoas_r').addInControls(
        Vector(1,excitation1(controlIteration)),modelControls);
78
79  %osimModel.updActuators().get('psoas_r').addInControls(
        Vector(1,0),modelControls);
80
81  osimModel.updActuators().get('rect_fem_r').addInControls(
        Vector(1,0),modelControls);
82
83  osimModel.updActuators().get('vas_int_r').addInControls(
```

```
    Vector(1,0),modelControls);
84
85  osimModel.updActuators().get('med_gas_r').addInControls(
    Vector(1,0),modelControls);
86
87  osimModel.updActuators().get('soleus_r').addInControls(
    Vector(1,0),modelControls);
88
89  osimModel.updActuators().get('tib_ant_r').addInControls(
    Vector(1,0),modelControls);
90
91
92  end
```

Código-fonte 13 – Codigo de execucao do controle

```
1  %% plot
2  plot_title = strcat('Sinais de entrada e saida');
3
4  h(1) = figure('NumberTitle', 'off', 'Name', plot_title);
5
6  %% excitacao do joelho
7  ax1 = subplot(2,1,1);
8  hold on;
9  plot(controlTimeplot,excitationplot,'r');
10 hold off;
11 title('Excitacao (knee) [V]');
12 xlabel('time [s]'); ylabel('Excitacao [V]');xlim([0,15]);
   ylim([min(excitationplot),max(excitationplot)]);
13
14
15 %% angulo do joelho
```

```
16 ax2 = subplot(2,1,2); hold on;
17 plot(controlTimeplot,kneeangleplot*180/pi,'b');
18
19 hold off;
20 title('Angulo (knee)')
21 xlabel('time [s]'); ylabel('Angulo');  ylim([min(
      kneeangleplot)*180/pi,max(kneeangleplot)*180/pi]);
22 %
23 %% excitacao do quadril
24 % ax3 = subplot(2,1,1); hold on;
25 % plot(controlTimeplot,excitationhipplot,'r');
26 % hold off;
27 % %title('Excitacao (hip) [V]');
28 % xlabel('time [s]'); ylabel('Excitacao [V]');  ylim([min(
      excitationhipplot),max(excitationhipplot)]);
29 % %
30 % ax4 = subplot(2,1,2); hold on;
31 % plot(controlTimeplot,hipangleplot*180/pi,'b');
32 %
33
34 %% angulo do quadril
35 % hold off;
36 % %title('Angulo (hip)');
37 % xlabel('time [s]'); ylabel('Angulo');  ylim([min(
      hipangleplot)*180/pi,max(hipangleplot)*180/pi]);
38
39 linkaxes([ax1,ax2],'x')
```