UNIVERSIDADE DE CAXIAS DO SUL – UCS
CAMPUS UNIVERSITÁRIO DA REGIÃO DOS VINHEDOS – CARVI
ENGENHARIA ELETRÔNICA

AUGUSTO POLETTO CUTULLI

IMPLEMENTAÇÃO DE UM MÉTODO DE GERAÇÃO DE TRAJETÓRIA BASEADO EM ZMP E UM MÉTODO DE CINEMÁTICA INVERSA BASEADO EM JACOBIANAS PARA A MARCHA DE UM ROBÔ BÍPEDE COM 12 ÂNGULOS DE LIBERDADE

BENTO GONÇALVES
2017

AUGUSTO POLETTO CUTULLI

IMPLEMENTAÇÃO DE UM MÉTODO DE GERAÇÃO DE TRAJETÓRIA BASEADO EM ZMP E UM MÉTODO DE CINEMÁTICA INVERSA BASEADO EM JACOBIANAS PARA A MARCHA DE UM ROBÔ BÍPEDE COM 12 ÂNGULOS DE LIBERDADE

Trabalho de Conclusão II apresentado ao Centro de Ciências Exatas, da Natureza e de Tecnologia da Universidade de Caxias do Sul como requisito parcial para obtenção do título de Engenheiro Eletrônico.

Orientador(a): Prof. Me. Ricardo Becker.

BENTO GONÇALVES
2017

AUGUSTO POLETTO CUTULLI

IMPLEMENTAÇÃO DE UM MÉTODO DE GERAÇÃO DE TRAJETÓRIA BASEADO EM ZMP E UM MÉTODO DE CINEMÁTICA INVERSA BASEADO EM JACOBIANAS PARA A MARCHA DE UM ROBÔ BÍPEDE COM 12 ÂNGULOS DE LIBERDADE

Trabalho de Conclusão II apresentado ao Centro de Ciências Exatas, da Natureza e de Tecnologia da Universidade de Caxias do Sul como requisito parcial para obtenção do título de Engenheiro Eletrônico.

Orientador(a): Prof. Me. Ricardo Becker.

Aprovado em: 12 / 12 / 2017

Banca Examinadora:

_____
Prof. Me. Ricardo Becker
Universidade de Caxias do Sul - UCS

_____
Prof. Dr. Alexandre Mesquita
Universidade de Caxias do Sul - UCS

_____
Prof. Me. Patric Janner Marques
Universidade de Caxias do Sul – UCS

Dedico este trabalho a Deus, à minha família e amigos.

AGRADECIMENTOS

Agradeço a Deus por tudo e, principalmente, pela Sua misericórdia em permitir a realização deste trabalho. Também agradeço à minha família, por todo o amparo, compreensão e força, assim como à minha namorada Larissa Teixeira da Silva, que me deu suporte durante toda essa etapa.

Aos meus amigos e colegas de curso, em especial Emanuel Finatto, Maikon Del Ré Perin e Guilherme Lemos de Lemos por sempre estarem junto comigo desde o início.

Ao Professor Mestre Ricardo Becker, pela sua orientação, inspiração e incentivo.

Aos laboratoristas, em especial, ao Fabiano Rodrigues pelo seu enorme auxílio na elaboração do robô

Ao Professor Dr. Anthony Maciejewski, e à Dra. Megan Emmons, durante meu intercâmbio, na *Colorado State University*, na disciplina de Introdução à Robótica e em outros trabalhos que muito me inspiraram a seguir nesta área.

E ao Professor Me. Patric Janner Marques pelo seu auxílio durante este trabalho.

Os homens tornaram-se cientistas porque esperavam encontrar lei na natureza, e esperavam encontrar lei na natureza porque criam em um Legislador.

C. S. Lewis

RESUMO

A robótica tem se tornado um elemento fundamental no meio industrial, e para o futuro espera-se que os robôs venham a fazer parte, cada vez mais, do meio social humano. Para que isso seja possível são necessários robôs aptos a se locomoverem nos ambientes humanos e, portanto, de métodos que possibilitem sua locomoção. Assim sendo, este trabalho teve como objetivo implementar um método para gerar a trajetória do deslocamento da pélvis e dos pés, baseado em ZMP (*Zero Moment Point*) e interpolações de terceira ordem, e um método de Cinemática Inversa iterativo baseado em Jacobianas, DLS (*Damped Least Squares*), para solucionar os pontos gerados. Foi considerado um robô bípede com 12 ângulos de liberdade (DOF - *Degrees of Freedom*), ou seja, equivalente à parte inferior do corpo humano. O algoritmo completo foi elaborado em MATLAB e um robô simplificado, com propósito demonstrativo, foi desenvolvido na prática com peças impressas e usinadas. Para os atuadores das juntas foram utilizados os servomotores MG996R e a passagem dos ângulos se deu por meio do microcontrolador Arduino Mega 2560. Os conceitos necessários voltados à robótica para manipuladores industriais e bípede foram abordados, assim como a metodologia e desenvolvimento de cada etapa. Por fim, através dos métodos foi possível obter um algoritmo capaz de gerar marchas para n passos com parâmetros variáveis. Para o principal teste realizado com três passos, foi gerado um conjunto de 380 subpontos, levando aproximadamente 620 segundos, com erros em posição, em média de: $1{,}987\ [mm], 2{,}007\ [mm], 2{,}088\ [mm], 2{,}094\ [mm]$, para o erro do tornozelo direito, do tornozelo esquerdo, do pé direito e do pé esquerdo, respectivamente. O robô desenvolvido não foi capaz de realizar a marcha completamente. A considerações constam ao final deste trabalho.

Palavras-chave: Robô Bípede 12 DOF. Marcha Bípede. Geração de Trajetória via ZMP. Cinemática Inversa via Jacobianas.

ABSTRACT

Robotics has become a fundamental element in the industrial environment, and future's expectations hopes that robots will become part of the human social environment even more. For it to be possible, robots are required to move around in human environments and, therefore, methods to allow robot's locomotion. Thus, this work aimed to implement a method to generate the trajectory of the pelvis and feet displacement, based on ZMP (Zero Moment Point) and third order interpolations, and an iterative Inverse Kinematic method based on Jacobian, DLS (Damped Least Squares) to solve the generated points. It was considered a biped robot with 12 angles of freedom (DOF - Degrees of Freedom), that is, equivalent to the lower part of the human body. The complete algorithm was elaborated in MATLAB. A simplified robot, with demonstrative purpose was developed in practice with printed and machined parts. The MG996R servomotors were used for the actuators of the joints and the angles were given through the Arduino Mega 2560 microcontroller. The necessary concepts for robotics for industrial and biped manipulators were approached, as well as the methodology and development of each step. Finally, through the methods it was possible to obtain an algorithm capable of generating gears for *n* steps with variable parameters. For the main test performed with three steps, a set of 380 subpoints was generated, taking approximately 620 seconds, with errors in position, in average of 1,987 [mm], 2,007 [mm], 2,088 [mm], 2,094 [mm], for the right ankle, left ankle, right foot and left foot, respectively. The developed robot was not able to carry out the march completely. The considerations are listed at the end of this paper.

Keywords: Biped Robot 12 DOF. Bipedal Walking. Trajectory Generation by ZMP. Inverse Kinematics by Jacobians.

LISTA DE FIGURAS

LISTA DE ABREVISTURAS E SIGLAS

ABS – Acrilonitrila Butadieno Estireno
AIST – *National Institute of Advanced Industrial Science and Technology*
ASIMO – *Advanced Step in Innovative Mobility*
CAN – *Controller Area Network Protocol*
COM – *Center of Mass*
CPG – *Central Pattern Generator*
DH – *Denavit-Hartenberg*
DLS – *Damped Least Squares*
DOF – *Degrees of Freedom*
DSP – *Double Support Phase*
FZMP – *Fictitious Zero Moment Point*
HOAP – *Humanoid for Open Architecture Platform*
HRP – *Humanoid Robotics Project*
HUBO – *Humanoid Robot*
LAN – *Local Area Network*
LIPM – *Linear Inverted Pendulum Mass*
KAIST – *Korea Advanced Institute of Science and Technology*
KHR – KAIST *Humanoid Robot*
PD – Proporcional Derivativo
PWM – Pulse Width Modulation
SSP – *Single Support Phase*
UHMW – *Ultra High Molecular Weight*
ZMP – *Zero Moment Point*

SUMÁRIO

1 INTRODUÇÃO

As tecnologias de controle de movimento no campo da robótica dominaram diversas aplicações na indústria e atualmente desempenham um papel fundamental na produção industrial, conforme Shimmyo, Sato e Ohnishi (2013). São considerados robôs industriais – manipuladores – se o dispositivo mecânico puder ser programado para realizar diversas tarefas, conforme afirma Craig (2012).

De acordo com Shimmyo, Sato e Ohnishi (2013), nos últimos anos vem ocorrendo uma crescente expectativa sobre robôs que poderão trabalhar fora do ambiente da indústria, agindo como serviçais em domicílios ou ajudantes dos humanos em outras tarefas. Para isso é necessário que os robôs se adaptem aos ambientes humanos. Portanto, robôs que possuírem formas humanas podem ser considerados aptos para esses ambientes. A empresa HONDA[1], por exemplo, construiu o robô ASIMO, com essa finalidade.

Entretanto, um dos principais desafios para tornar essa integração possível é a locomoção. A locomoção humana caracteriza um aspecto próprio em relação a qualquer outro ser vivo: o bipedismo. Seu equilíbrio e deslocamento são complexos uma vez que, conforme Luksch (2010), o tronco cerebral e/ou cerebelo realizam o controle com o auxílio de informações sensoriais visuais e vestibulares para que a perna e o pé estejam posicionados exatamente no local necessário, no tempo correto, para que seja possível manter o equilíbrio.

Esse sistema complexo não pode ser copiado com exatidão atualmente. Entretanto, a robótica voltada para a locomoção bípede já apresenta resultados que permitem que robôs equilibrem-se, mesmo sob influências externas, e superem diversas situações como: subir degraus; evitar colisões; locomover-se em torno de objetos; caminhar em terrenos irregulares; e até mesmo levantar-se em caso de queda (LUO e LIN, 2015). Porém, o nível de complexidade é elevado até mesmo para as funções básicas e as aproximações morfológicas de um robô em relação à locomoção humana são subjetivas ao construtor. Assim, determinados aspectos da imitação dos movimentos são suprimidos (LUKSCH, 2010).

Até então, diversos métodos de controle de marcha foram propostos e desenvolvidos para robôs humanoides e não será o objetivo deste trabalho sua

[1] Site oficial da ASIMO: <http://asimo.honda.com>. Acesso em: 13 de Jun. de 2017.

classificação. Este trabalho terá como enfoque abordar dois aspectos primários do controle de robôs bípedes, e fundamentais na área da robótica: geração de trajetória e Cinemática Inversa.

Uma parte importante do controle estabelece qual ação de movimentos que se esperam que o robô desempenhe: seja uma marcha periódica, seja posicionar um pé de apoio no solo, caso se identifique que o centro de massa do robô indique algum desequilíbrio físico. A execução dessas ações irá se converter, comumente, em trajetória, compostas de diversos pontos bem definidos no espaço onde o robô se encontra, a serem alcançados pelos seus membros (CRAIG, 2012). Porém, independentemente do tipo da ação determinada, ainda é necessário que os pontos sejam traduzidos em posições angulares para cada atuador (motores) contido nos membros. Esse papel, quem desempenha é a Cinemática Inversa e, da mesma forma, é empregado em manipuladores industriais, segundo Craig (2012).

Outras aplicações possíveis, fazendo uso da Cinemática Inversa, está na nas órteses ou próteses robóticas e exoesqueletos. Elas são utilizadas para auxiliar a locomoção ou movimentos de outros membros do corpo humano que estão faltantes ou que perderam sua função, por exemplo, o exoesqueleto de ombro apresentado por Jung e Bae (2015), ou o tornozelo por Hong, Shin, e Wang (2014).

Tomando como referência a implementação de robôs (bípedes) fora do ambiente industrial, e visto a importância da utilização da Cinemática Inversa em diversos campos da robótica – sabendo-se de sua correlação indispensável com os pontos de trajetória – implementaram-se dois métodos (geração de trajetória e Cinemática Inversa) levando em conta um robô bípede de 12 ângulos de liberdade (*Degrees of Freedom* - DOF), sendo 6 DOF por perna.

O método de geração de trajetória abordado, ZMP (*Zero Moment Point* – Ponto de Momento Zero) foi concebido por Vukobratovic e Borovac (2004) para marchas de robôs bípedes em 1968, sendo utilizado ainda atualmente. Muitas vezes é associado a um modelo simplificado de pêndulo invertido que simplifica a movimentação da massa de uma caminhada humana (DALEN, 2012). O ZMP será o principal responsável para obter-se a trajetória da pélvis que será considerada com a mesma do CoM (*Center of Mass* – Centro de Massa) do robô. Para trajetória dos pés, optou-se por um método de equacionamento e interpolação utilizado por Huang, et al. (2001).

Para a Cinemática Inversa foi implementado um método baseado em Jacobianas. Buss (2009) descreve as soluções possíveis para o equacionamento dos ângulos via DLS (*Damped Least Squares* – Quadrados Mínimos Amortecidos) o qual foi empregado para esse trabalho.

1.1 OBJETIVO GERAL

Este trabalho tem por objetivo implementar um método de geração de trajetória baseado em ZMP e pontos interpolados, e um método de Cinemática Inversa iterativa baseada em Jacobianas através de DLS a uma estrutura de robô bípede correspondente à parte inferior do corpo humano.

1.2 OBJETIVOS ESPECÍFICOS

Como forma de destacar alguns dos entregáveis do trabalho, baseado no objetivo geral posto, apresentam-se aqui os objetivos específicos:

a) Implementar um algoritmo de geração de trajetória baseado em ZMP e interpolações;
b) Implementar um algoritmo de Cinemática Inversa iterativo baseado em Jacobianas via DLS;
c) Realizar a construção gráfica dos movimentos do robô;
d) Desenvolver uma estrutura de robô bípede com 6 DOF por perna para demonstração dos conceitos.

1.3 RESTRIÇÕES DO TRABALHO

Os métodos implementados contemplam: determinar os pontos em que o robô deve posicionar a pélvis (centro de massa) e os pés durante uma marcha; e fazer com que cada atuador (junta) encontre o ângulo correto para alcançar os pontos dados em cada instante dessa marcha.

O primeiro método (geração de trajetória) é o responsável pelo equilíbrio físico do robô. Por ser baseado em ZMP, idealmente, os pontos da trajetória gerados já levam

em conta o lugar e o tempo correto em que os pés e pélvis do robô deve se posicionar para que ele não caia. Entretanto, para um caso não ideal, é necessário considerar diversas incertezas inerentes à própria estrutura do bípede e do ambiente, que levam o robô a não se comportar conforme o previsto em uma situação real. Para que isso seja corrigido satisfatoriamente é necessário um vasto monitoramento e compensações na trajetória e ângulos gerados para seu deslocamento, e esse trabalho não prevê tal situação.

Ainda assim, o algoritmo foi aplicado em uma estrutura física robótica real, sem considerar sensoriamentos e realimentações para compensá-lo, apenas para demonstração dos conceitos. A parte referente à dinâmica (forças e torques) também não foi levada em conta. Existem métodos para contornar alguns desses aspectos que serão vistos adiante.

1.4 DESCRIÇÃO GERAL DO TRABALHO

O presente trabalho compõe-se de 7 capítulos, sendo o capítulo 1 introdutório ao tema a ser abordado e apresentando objetivos e restrições ao que se pretende realizar.

O capítulo 2 apresenta, qual o tipo de estrutura de robô bípede escolhida para o trabalho e posteriormente a descrição dos métodos de geração de trajetória para robôs bípedes assim como fundamentos da robótica e conceitos relacionados. Parte-se da descrição matricial de pontos no espaço até a descrição de uma estrutura completa de um manipulador articulado, assim como a descrição da Cinemática Inversa e qual método utilizado para o robô bípede.

O capítulo 3 aborda a metodologia utilizada para o desenvolvimento do projeto baseado no que foi apresentado no capítulo 2, dividindo o desenvolvimento em etapas e apresentando a escolha das tecnologias utilizadas.

No capítulo 4 é apresentado o desenvolvimento do trabalho detalhando a construção do algoritmo e da estrutura física do robô bípede e dificuldades obtidas durante o processo.

O capítulo 5 demonstra os resultados obtidos com o desenvolvimento do trabalho também em comparação com a literatura utilizada.

O capítulo 6 apresenta as considerações finais referentes aos resultados obtidos e relevância do trabalho.

No capítulo 7, são apresentadas possíveis melhorias para futuros trabalhos e as perspectivas para a sua continuidade.

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentada a estrutura do robô bípede. A descrição da estrutura servirá de contextualização para demonstrar o panorama geral de controle de um robô bípede (e para definir àquela que será utilizada neste trabalho). Ainda serão feitas algumas considerações sobre questões específicas de robôs bípedes. Esse descritivo, por sua vez, contextualiza o papel da geração de trajetória e da Cinemática Inversa. Para a compreensão do segundo método adotado, serão apresentados os fundamentos da robótica.

2.1 ESTRUTURA DE ROBÔ BÍPEDE

Primeiramente, faz-se necessário contextualizar qual tipo da estrutura de robô está sendo lidado nesse trabalho, para que posteriormente possa ser realizada uma correta interpretação de suas etapas.

A estrutura do robô bípede idealizada para o projeto foi inspirada com base em outros robôs bípedes atuais, ao menos no que diz respeito à parte inferior do corpo, ou seja, membros inferiores e quadril. De forma que a estrutura escolhida deve estar apta a caminhar. Ou seja, a estrutura do robô está sendo proposta com a expectativa na implementação de métodos de geração de trajetória para marchas.

Entretanto, Luksch (2010) afirma que não é possível um robô possuir todos os ângulos de liberdade (*Degree of Freedom* - DOF) que um ser humano possui. Ainda assim, é possível encontrar um conjunto de DOF menor, mas possível de ser implementado em um robô, e que seja suficiente para permitir uma locomoção bípede flexível e que se assemelhe à humana.

Luksch (2010) também lista alguns robôs analisados onde inclui o número total de DOF do robô e o número de DOF por perna, conforme a Tabela 1. Na Tabela 1 é possível verificar que a maior parte dos robôs possui seis DOF por perna ou mais.

Luksch (2010) ainda apresenta a disposição considerada para seu trabalho. Conforme a Figura 1.a, é possível verificar 3 DOF, 2 nas duas juntas do pé (*pitch e roll*), e 1 na junta do joelho da perna esquerda (*pitch*). Na Figura 1.b é possível verificar 3 DOF correspondentes, às três juntas do quadril esquerdo (*pitch, roll e yaw*). As duas juntas do pé e as três juntas do quadril são sobrepostas. Na Figura 1.c pode-se

visualizar a parte inferior completa do robô bípede com suas juntas representadas em forma de cilindros orientados de acordo com seus eixos de rotação.

Tabela 1: Número de DOF em outros projetos de robôs bípedes.

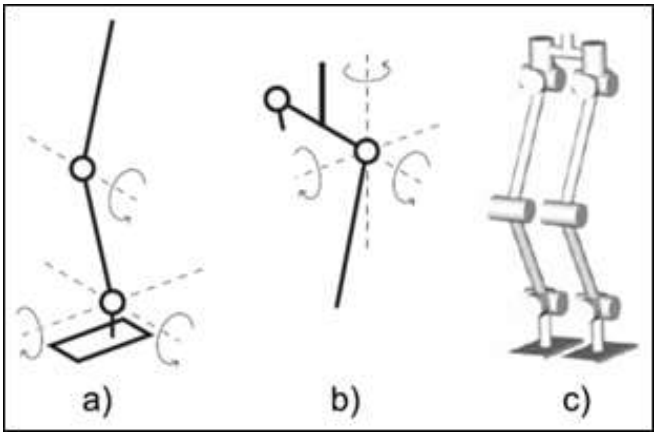
Modelo do Robô	Grupo do Projeto	DOF	DOF por perna
Wabian-2R	*Uni. Waseda*, Japão	41	6+1p*
H6	*Uni. Tokyo*, Japão	35	7
H7	*Uni. Tokyo*, Japão	30	7
P2	Honda, Japão	30	6
P3	Honda, Japão	30	6
ASIMO	Honda, Japão	26	6
ASIMO (pesquisa)	Honda, Japão	34	6
HRP-2	AIST, Japão	30	6
HRP-2L	AIST, Japão	12	6
HRP-3	AIST, Japão	42	6
SDR-4X (QRIO)	Sony, Japão	38	6
Partner Robots	Toyota, Japão	31	6
KHR-2	KAIST, Coréia do Sul	41	6
KHR-3 (HUBO)	KAIST, Coréia do Sul	41	6
BHR-2	*Uni. Beijing*, China	32	6
THBIP-I	*Uni. Beijing*, China	32	6
Johnnie	TU *Munich*, Alemanha	17	6
Lola	TU *Munich*, Alemanha	22	7
BARt-UH	*Uni. Hanover*, Alemanha	13	3
Lisa	*Uni. Hanover*, Alemanha	12	6
BIP	INRIA, França	15	6
Rabbit	INRIA, França	4	2
Rh-1	*Uni. Madrid*, Espanha	21	6
Spring Turkey	MIT, EUA	4	2
* Wabian-2R: há mais uma junta no pé, portanto mais um DOF.			

Fonte: Luksch (2010), adaptado pelo autor (2017).

Pitch, *Roll* e *Yaw*, podem ser simplificados como movimento para frente, para o lado e próprio eixo. O joelho pode somente se mover para frente; o pé pode se mover para frente e para o lado; e o quadril pode mover a perna para frente, para o lado e no próprio eixo. Essas nomenclaturas serão utilizadas neste trabalho dessa forma por convenção.

Mais adiante será definida a orientação de rotação de cada junta para cada DOF, porém, em outros trabalhos também pode ser encontrado o mesmo tipo de estrutura. Geralmente, essa é a disposição dos DOF quando há seis por perna.

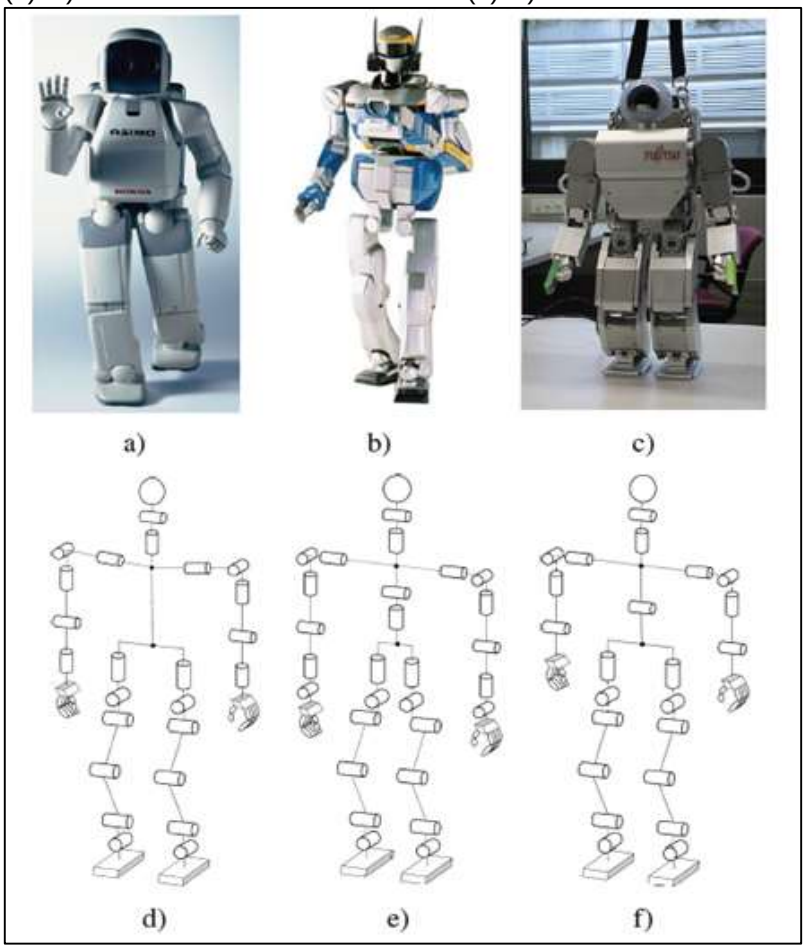
Figura 1: Diagrama da estrutura da parte inferior do robô bípede apresentando 6 DOF: a) Joelho e pé esquerdo. b) Quadril esquerdo. c) Parte inferior completa.



Fonte: Adaptado de Lukcsh (2010).

Conforme a Figura 2, podem ser encontrados outros robôs com mesma estrutura, apresentando as juntas separadas, destacando que, nos pés e quadris, os eixos se sobrepõem.

Figura 2: Modelo do robô e sua estrutura apresentando as juntas: a) HONDA ASIMO e sua estrutura em (d). b) AIST HRP-2 e sua estrutura em (e). c) HOAP-2 e sua estrutura em (f).



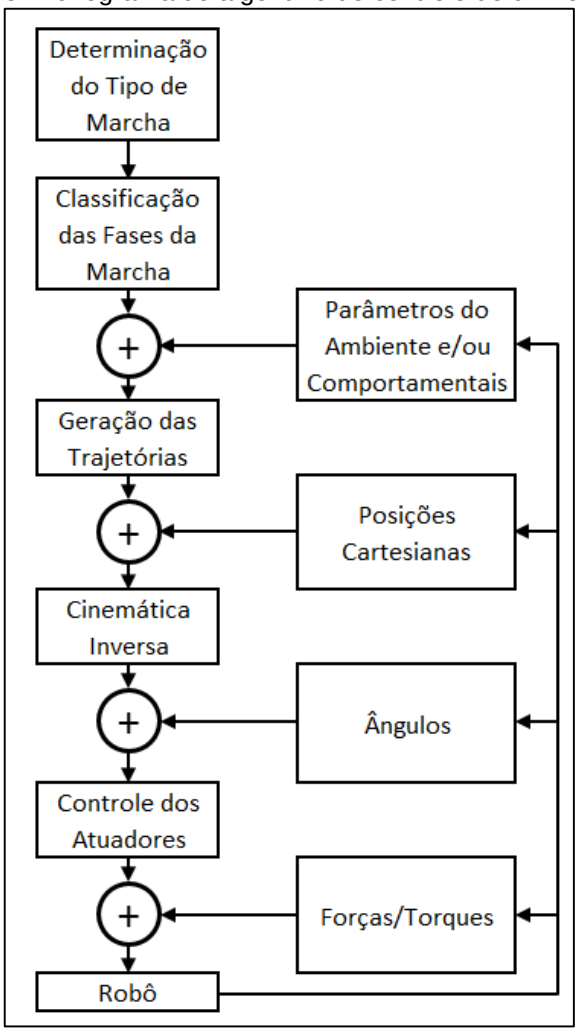
Fonte: Adaptado de Ali, Park, & Lee (2010).

Levando-se em conta o que foi apresentado aqui até então, escolheu-se um modelo de 6 DOF por perna, semelhante aos apresentados na Figura 2, por ser o suficiente para a locomoção bípede. Essa concepção já tem sido usada extensivamente em outros projetos (ALI, PARK, e LEE, 2010).

2.2 PANORAMA DO CONTROLE DA LOCOMOÇÃO DE UM ROBÔ BÍPEDE

Neste tópico é apresentado o fluxograma de um tipo de controle de robôs bípedes. Este tipo de controle apresentado de forma simplificada foi baseado, principalmente, no modelo KHR-2 (HUBO). Este trabalho não irá implementar todas as etapas descritas no fluxograma da Figura 3, entretanto é necessário explicar cada bloco para que se tenha a compreensão correta do funcionamento de um robô bípede.

Figura 3: Fluxograma do algoritmo de controle de um robô bípede.



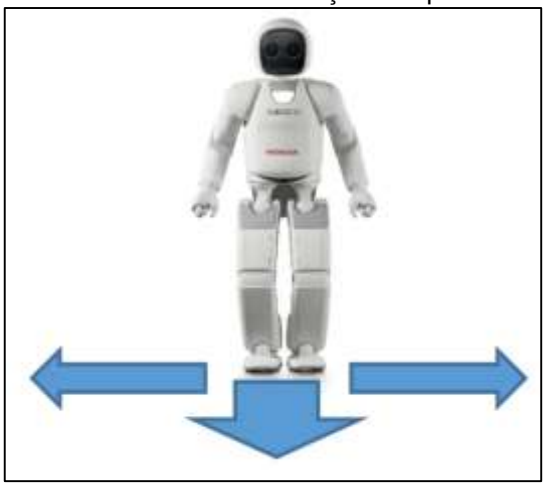
Fonte: Adaptado de Kim, Park, e Oh (2006).

Será considerado um robô de acordo com a estrutura de 12 DOF descrita anteriormente. Ao final deste tópico serão feitas considerações sobre os tipos de arquiteturas de controle que poderiam ser empregadas.

2.2.1 Etapa da Determinação do Tipo de Marcha

O processo inicia-se pela determinação do tipo de marcha estabelecendo os parâmetros iniciais para o robô e o comando que o robô deve executar. Ilustrado pela Figura 4, pode ser, à princípio, qualquer comando que o robô possa desempenhar, ou seja, caminhar para frente, para trás, em curva para os lados, etc.

Figura 4: Comando - Determinação do tipo de marcha.



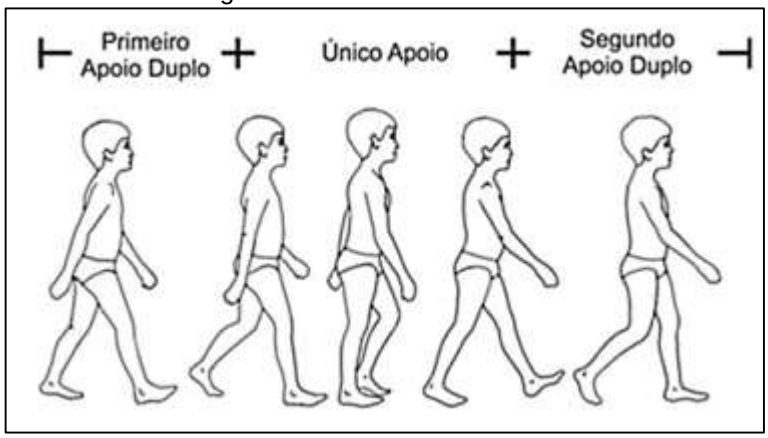
Fonte: Adaptado de Honda (2017).

2.2.2 Etapa da Classificação das Fases da Marcha

A classificação das fases da marcha são estabelecidas por um ciclo de marcha periódica, afirma Kim J.-W. (2014). Conforme Kim, Park e Oh (2006), as fases podem ser distinguidas, principalmente, quando há duas pernas de apoio, DSP (Fase de Apoio Duplo – *Double Support Phase*) no solo e quando há somente uma perna de apoio, SSP (Fase de Apoio Único – *Single Support Phase*). Essa etapa tem o propósito de facilitar e classificar o algoritmo, informando parâmetros para a etapa de geração de trajetória como, por exemplo, a duração e a distância para cada passo; duração de tempo onde ambos os pés tocam o solo; amplitude de cada pé em relação ao solo durante o passo; amplitude do deslocamento lateral da pélvis, e outros. As fases já são pré-definidas de acordo com o comandos dados pela etapa anterior (também pré-

definidos). A Figura 5 apresenta a marcha periódica de um ser humano de oito anos (MIYADAIRA, 2011) e ilustra a definição das fases da marcha para o robô.

Figura 5: Fases da marcha humana

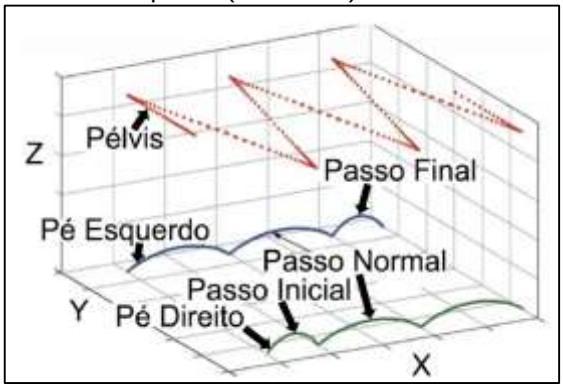


Fonte: Adaptado de Miyadaira (2011).

2.2.3 Etapa da Geração das Trajetória

A geração das trajetória recebe os parâmetros das etapas anteriores, e elabora trajetória para a marcha do robô. De acordo com Craig (2012), e para esse trabalho, uma trajetória se refere à pontos cartesianos intermediários (X, Y, Z). Esses pontos estabelecem as posições por onde cada pé e a pélvis estarão se deslocando a cada instante de tempo. Consequentemente, a trajetória em si também é a principal responsável por garantir o equilíbrio físico do robô, considerando um cenário ideal, informa Kim, Park, e Oh (2007), pois leva em conta a dinâmica do robô. A Figura 6 ilustra uma trajetória, composta de pontos no espaço (X, Y, Z), dos pés e da pélvis, durante uma marcha.

Figura 6: Deslocamento no espaço cartesiano (X,Y,Z) do pé esquerdo (Azul), pé direito (Verde) e pélvis (Vermelho).

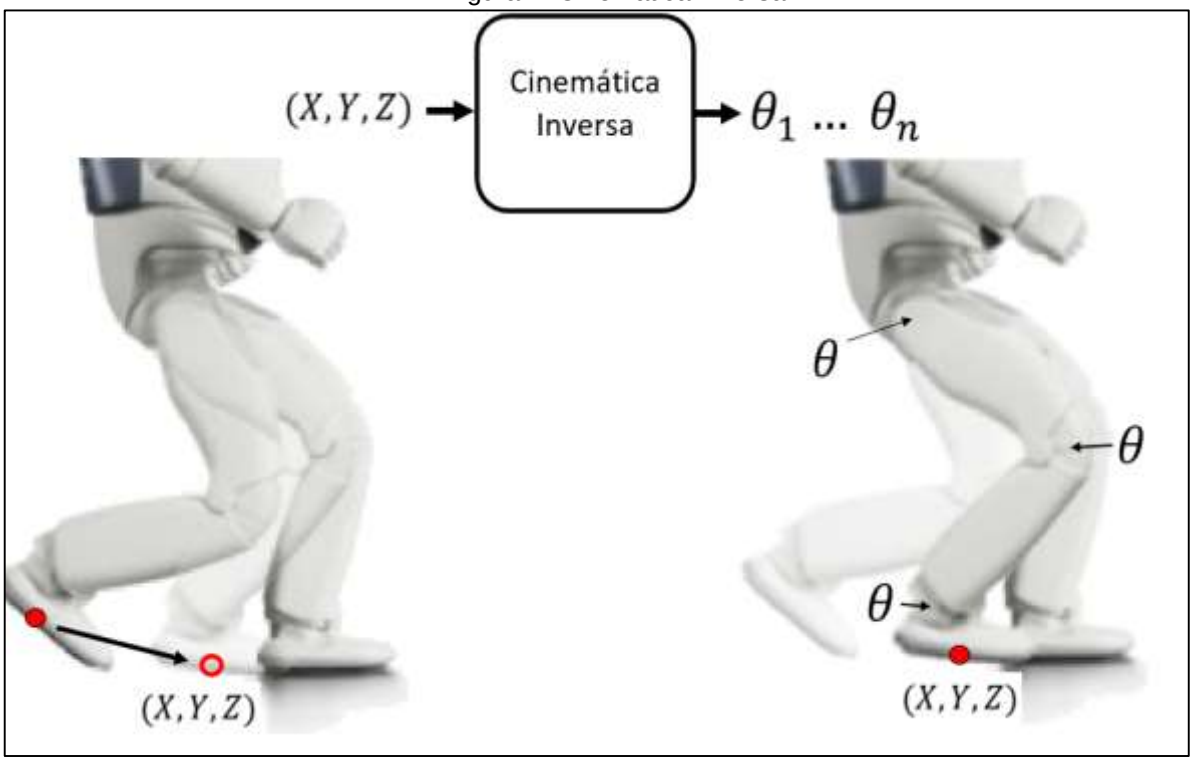


Fonte: Adaptado de Wang, Ceccarelli e Carbone (2016).

2.2.4 Etapa da Cinemática Inversa

A Cinemática Inversa, conforme Craig (2012), é a etapa responsável por gerar os conjuntos de ângulos possíveis de acordo com cada ponto cartesiano de uma trajetória para o robô. Uma vez que os atuadores alcancem os ângulos estabelecidos, espera-se que a posição da pélvis e dos pés esteja na mesma posição dos pontos estabelecidos da trajetória dada, assim como ilustra a Figura 7.

Figura 7: Cinemática Inversa.

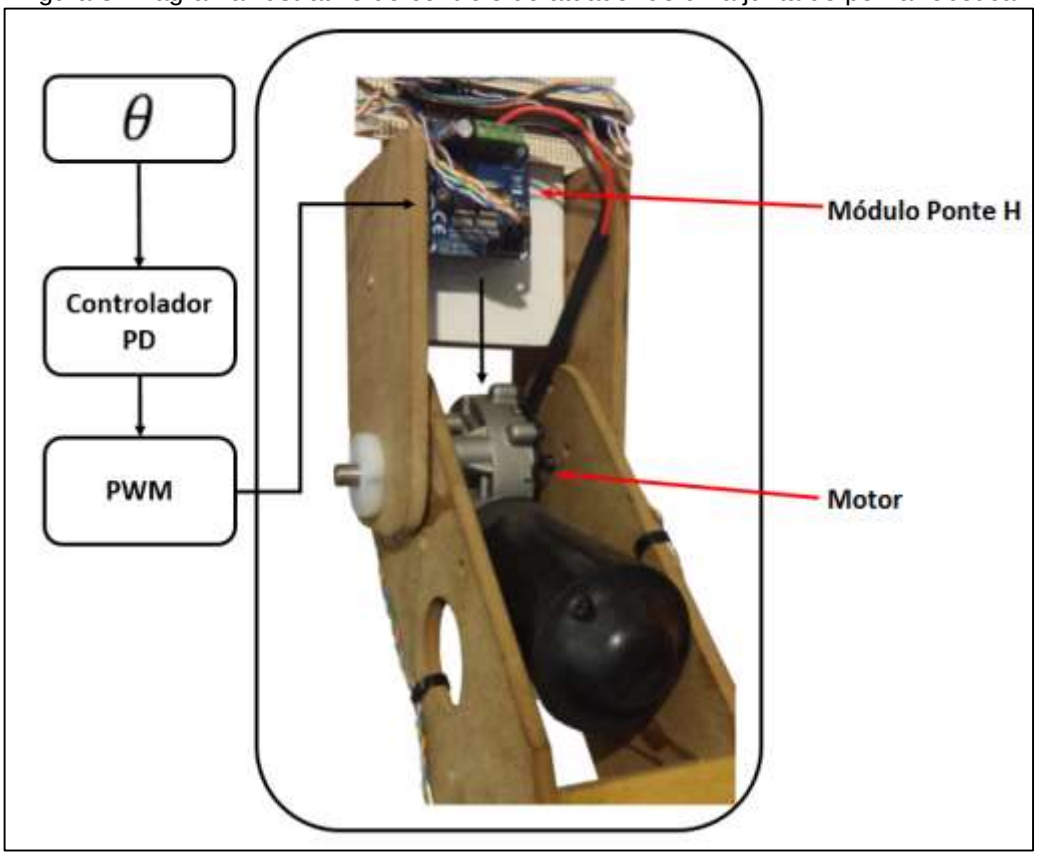


Fonte: Adaptado de New Atlas (2017).

2.2.5 Etapa do Controle do Atuador

O controle do atuador garante que o atuador rotacione especificamente nas posições indicadas para cada ângulo passado. No projeto do KHR-2, informa Kim, Park, e Oh (2006), cada atuador era um servomotor controlado por um controle proporcional derivativo (PD). Entretanto, isso pode variar muito de acordo com o tipo de motor ou servomotor utilizado e o tipo de controle implementado. Uma ilustração do diagrama do controle do atuador pode ser visto na Figura 8.

Figura 8: Diagrama ilustrativo do controle do atuador de uma junta de perna robótica.



Fonte: O próprio autor (2017).

Dessa forma, considerando um cenário ideal, o robô é capaz de desempenhar uma marcha mantendo o equilíbrio físico. Entretanto, ao se considerar um cenário real, os tipos de perturbações encontrados são diversos, desde inclinações e perturbações no solo até ruídos provindos dos atuadores. Assim, sensores podem ser instalados na estrutura do robô coletando informações para atualizar o sistema para cada etapa desejada. O fluxograma da Figura 3, mostrado anteriormente, apresenta alguns parâmetros que podem ser descobertos e informados ao sistema para realizar os ajustes necessários. Deve-se salientar que a maioria dos robôs humanoides são feitos com marchas calculadas para terrenos planos e a menor irregularidade pode causar sérias complicações de equilíbrio físico para o robô, informa, Kim, Park e Oh (2007).

2.2.6 Considerações de Gerais para Robôs Bípedes

Segundo Kim, et al. (2016) a maior parte das pesquisas no controle de robôs bípedes humanoides está ramificada em duas linhas: geração de padrões (trajetória) e controle de estabilização de movimento.

A arquitetura de controle baseada em geração de trajetória de marcha é conduzida sem uma realimentação completa, ou seja, é *off-line* (KIM J.-W, 2014). É uma arquitetura de controle pré-estabelecida, e faz uso de certos princípios como, principalmente, ZMP *(Zero Moment Point* – Ponto de Momento Zero) para equilíbrio físico (KIM, et al. 2016), e geração de padrões de marchas humanoides considerando etapas com um pé de apoio (*Single Support Phase* – SSP) e com dois pés de apoio (*Double Support Phase* – DSP), conforme Kim J.-W, 2014.

Já o controle de estabilização de movimento, emprega métodos utilizando, muitas vezes, CPG (*Central Pattern Generator* - Gerador De Padrões Central), frequentemente faz uso de redes neurais e/ou lógica *Fuzzy* (KIM, et al., 2016), para ajustes imediatos (*online*) dos ângulos das juntas durante a marcha. Também faz um uso maior de sensores externos do que no primeiro tipo de arquitetura de controle (KIM, et al. 2016). Logo, o controle de estabilização de movimento é denominado como uma arquitetura reativa (mais sofisticada, que reage ao meio).

Porém, ainda que sejam traçadas duas linhas distintas de pesquisa, não há normas que impeçam que os princípios dos modelos destacados se misturem e é isso que acontece em muitos casos, criando arquiteturas híbridas (LUKSCH, 2010).

O fluxograma apresentado na Figura 3 pode ser considerado um tipo híbrido, pois, gera as trajetória de forma *offline*. Porém, através de sensores o robô pode receber informações do meio em que se encontra e do estado atual do robô estando apto a fazer ajustes necessários para manter-se na trajetória estabelecida e/ou o equilíbrio físico.

2.3 MÉTODO DE GERAÇÃO DE TRAJETÓRIA

Neste tópico será abordado a contextualização teórica do método de geração de trajetória para o robô bípede deste trabalho. O método de geração de trajetória foi idealizado utilizando o critério de ZMP (*Zero Moment Point*), calculado por meio da simplificação do robô por pêndulo invertido (*Linear Inverted Pendulum Mass* – LIPM). Dessa forma é possível encontrar a referência de trajetória para os pés e pélvis (que será o Centro de Massa - COM) suprimindo a dinâmica, conforme será apresentado posteriormente. Portanto, a forma essencial considerando esse contorno na dinâmica pode ser vista em (23) e (24). Os tópicos 2.3.1 e 2.3.2 serão sobre como encontrar a forma simplificada. Ao fim, também será abordado uma forma de interpolação dos

pontos da trajetória para a definir o deslocamento da perna suspensa que dá o passo durante a marcha.

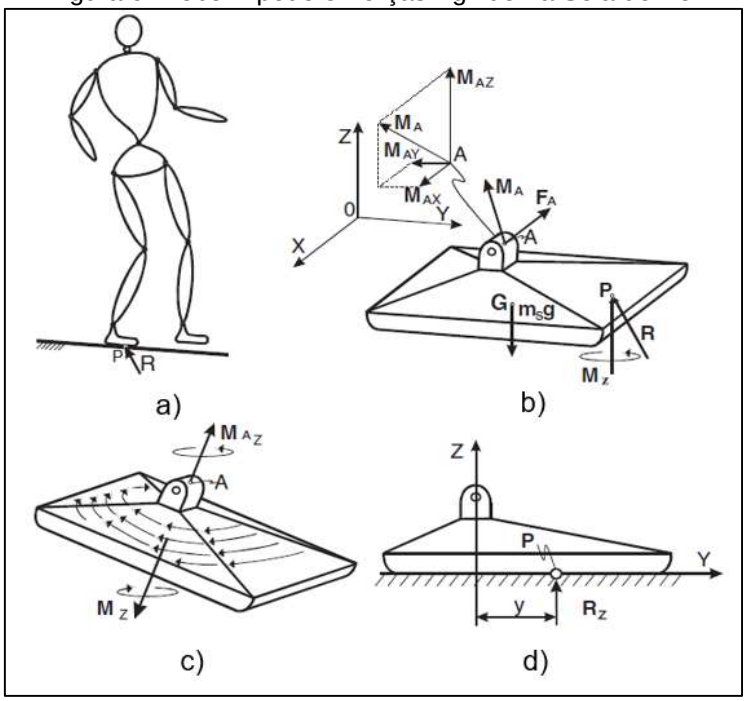
2.3.1 Definição e Uso do *Zero Moment Point* (ZMP)

O *Zero Moment Point* (ZMP) ou Ponto de Momento Zero é um critério importante e pode ser aplicado para gerar trajetória de referência em uma síntese de marcha *off-line* (pré-estabelecida), ou como um indicador para o controle de marcha *online* (durante o movimento), como já mencionado. Formulado por Miomir Vukobratovic em 1968 (LUKSCH, 2010), o ZMP é aplicado especificamente para isso desde então. Portanto, é um conceito considerado confiável, amplamente empregado na análise da marcha de robôs bípedes (ERBATUR e KURT, 2009).

O ZMP pode ser alcançado garantindo que a área total do pé do robô esteja em contato com o solo, assim o único contato que o pé realiza com o ambiente é por meio da força de atrito e por força vertical reativa do solo, conforme explica Vukobratovic e Borovac (2004).

Para que se entenda o conceito do ZMP partindo do que é dito em Vukobratovic e Borovac (2004), e fazendo uso das abreviações de Luksch (2010), considera-se a Figura 9:

Figura 9: Robô Bípede e Forças Agindo na Sola do Pé.



Fonte: Vukobratovic e Borovac (2004).

De acordo com a Figura 9.a, conforme Vukobratovic e Borovac, (2004) considera-se um robô bípede apoiado somente em de seus pés, durante uma marcha. Na Figura 9.b, pode-se negligenciar o que está acima da junta do tornozelo, substituindo a influência do resto do mecanismo pela força F_A e pelo Momento M_A (VUKOBRATOVIC e BOROVAC, 2004). O ponto G é o centro de gravidade dado pelo peso do próprio pé. O ponto P é o ponto onde o pé sofre a reação do solo, que faz com que se mantenha todo o corpo em equilíbrio. A reação total do solo consiste na força $R(R_X, R_Y, R_Z)$ e no momento $M(M_X, M_Y, M_Z)$. As componentes na horizontal de R e M, de acordo com Vukobratovic e Borovac, (2004), são balanceados pelo atrito, uma vez que o atrito age no ponto de contato entre o pé e o solo, considerando o pé em repouso. Rx e Ry representam a força de atrito que está balanceando as componentes horizontais de F_A.

Na Figura 9.c, M_Z, Momento Reativo Vertical, representa as forças reativas de atrito do momento que fazem o balanço do componente vertical de M_A e do momento induzido por F_A. Portanto, conforme Vukobratovic e Borovac (2004), considerando que o pé não desliza, o atrito pode ser representado por (Rx, Ry, Mz), e R_Z representa a força reativa do solo que faz o balanço da força vertical.

Na Figura 9.d, conforme Vukobratovic e Borovac (2004), devido a uma natureza unidirecional da conexão entre o pé e os componentes horizontais do solo de todos os momentos ativos, os componentes podem ser compensados apenas deslocando a posição do ponto P da força reativa R dentro do polígono de suporte (pé). Logo, a componente horizontal do momento M_A irá alterar a força reativa à posição correspondente para balancear a carga adicional. Ainda, segundo os citados autores, se o polígono do pé não for grande o suficiente para uma posição de R, todo o mecanismo perderá o equilíbrio e cairá.

Portanto, as condições para que o robô esteja em equilíbrio dinâmico são que P na sola do pé, onde a força reativa do solo está agindo, sejam $M_X = 0$ e $M_Y = 0$. E uma vez que ambas as componentes relevantes para a realização do balanço dinâmico são iguais a zero, pode-se chamar o Ponto P de Ponto de Momento Zero (ZMP). Ou seja, toda a vez que a reação do solo no pé em repouso pode ser reduzida à força R e à componente vertical do momento M_Z, o ponto P, onde a força reativa está agindo, representa o ZMP (VUKOBRATOVIC e BOROVAC, 2004).

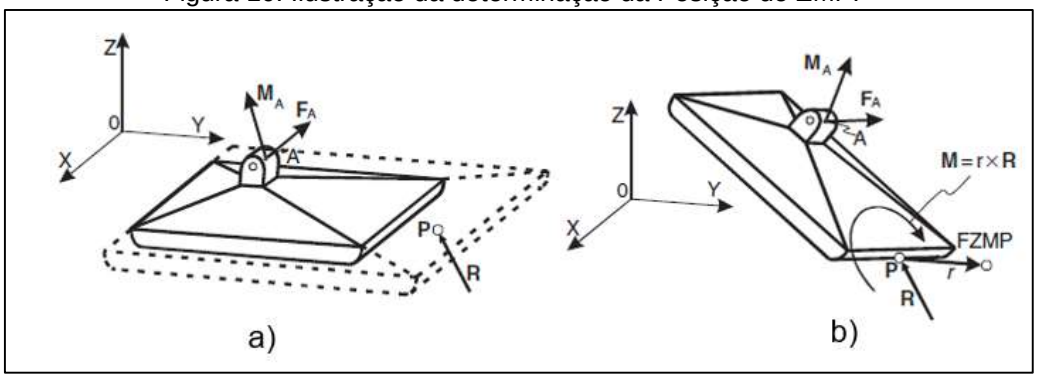
Assim, conforme Vukobratovic e Borovac, (2004), pode-se questionar qual será a posição de ZMP que resultará em equilíbrio dinâmico, considerando-se que toda a superfície da sola do pé esteja em repouso no solo. As equações de equilíbrio estático para o pé de suporte são (1) e (2):

$$R + F_A + m_s g = 0 \tag{1}$$

$$\overrightarrow{OP} \times \vec{R} + \overrightarrow{OG} \times m_s g + M_A + M_Z + \overrightarrow{OA} \times F_A = 0 \tag{2}$$

Onde O é a origem das coordenadas do sistema, P é o ponto onde a força reativa do solo age, G é o ponto do centro de massa do pé, A é a junta do tornozelo e m_s é a massa do pé. Assim, conforme Vukobratovic e Borovac, (2004), P deve estar dentro do polígono do pé, para garantir o equilíbrio dinâmico. Se o ponto computado estiver fora do polígono, chama-se de ZMP-fictício (FZMP), conforme a Figura 10:

Figura 10: Ilustração da determinação da Posição do ZMP.



Fonte: Vukobratovic e Borovac, (2004).

Se a origem do sistema for projetada em P, e (2) for projetada no eixo Z o componente vertical será (3):

$$M_Z = M_{fr} = -\left(M_A^z \left(\overrightarrow{OA} \times F_A\right)^z\right) \tag{3}$$

E portanto, se aplicado em (3), tem-se (4):

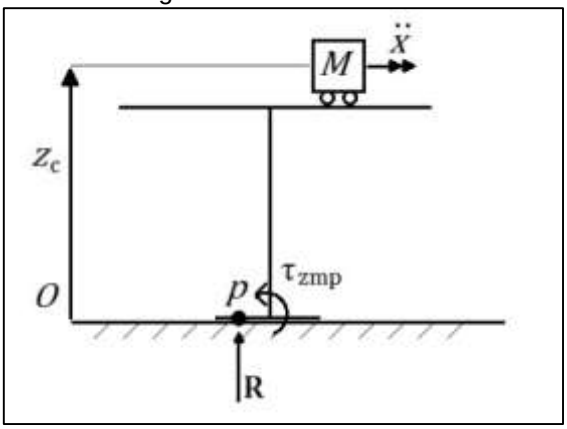
$$\overrightarrow{OP} \times \vec{R} + \overrightarrow{OG} \times m_s g = 0 \tag{4}$$

Geralmente M_z é diferente de zero, e pode ser reduzido a zero somente com dinâmicas apropriadas, explica Vukobratovic e Borovac (2004).

2.3.2 Cálculo do ZMP Utilizando a Simplificação por *Cart-Table-Model*

Dado o caso anterior em (4), Rudy (2014) calcula a posição do ZMP usando a Segunda Lei de Newton, quando a massa e as propriedades do robô são conhecidas. De forma que utiliza-se o chamado *Cart-Table Model* (“modelo de carrinho-de-carga”) de acordo com a Figura 11. Com

Figura 11: *Cart-Table Model.*



Fonte: Rudy (2014).

Esse modelo mostrado na Figura 11, possui uma massa m que percorre uma plano de altura z_c. O plano não cai enquanto o ponto p_{ZMP} permanecer dentro da base, o que significa que o torque τ_{ZMP} é igual a zero. Se a posição horizontal do COM (*Center of Mass* – Centro de Massa) é dada por d, relativa à origem 0, a resultante dos torques sobre p (ponto correspondente ao ZMP), τ_p, pode ser calculado através da função vetorial (5). É possível verificar a similaridade entre (5) e (4), que pode ser simplificada na função escalar (6), e por consequência, pode ser simplificada novamente sabendo-se que $\tau_x = \tau_y = 0$, em (7), explica Rudy (2014).

Assim, pode-se constatar que o ZMP se torna o COM quando não há aceleração (RUDY, 2014), ainda o ZMP possa estar localizado fora do polígono de suporte (área da sola do pé), sendo esse o FZMP (*Fictitious* ZMP – ZMP Fictício).

$$\tau_p = p \times R + \tau_{ZMP} \tag{5}$$

$$\tau_{ZMP} = -mg(d - p) + mz_c\ddot{d} \tag{6}$$

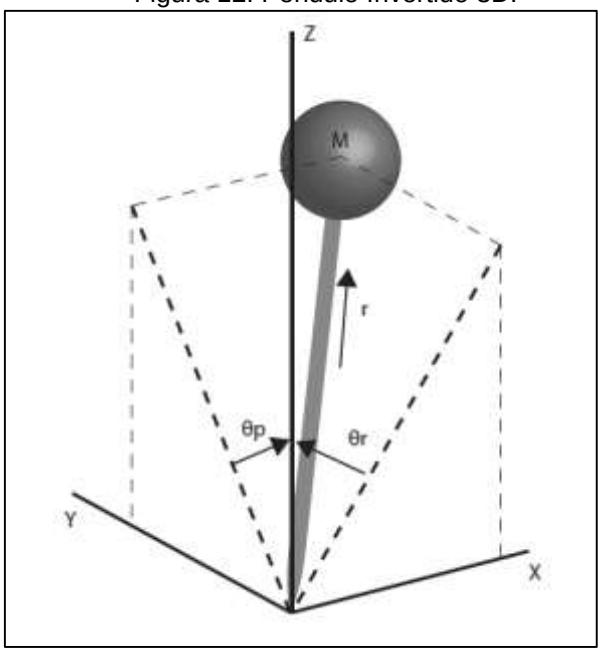
$$p = d - \frac{z_c}{g}\ddot{d} \tag{7}$$

2.3.3 Simplificação da Caminhada Humana Utilizando Massa de Pêndulo Invertido Linear (LIPM)

Conforme afirma Dalen (2012), o modelo de Massa de Pêndulo Invertido Linear (LIPM – *Linear Inverted Pendulum Mass*) é muito utilizado como sendo uma simplificação da caminhada humana, e a relação entre esse modelo simplificado da caminhada humana com o ZMP pode ser dada através da utilização do *Cart-Table Model*, conforme foi apresentado anteriormente. Assim, a seguir tem-se o modelo do LIPM, e como ele pode ser deduzido e simplificado até que se chegue nas mesmas equações obtidas pela simplificação do ZMP. Dessa forma, fica validado a utilização do LIPM em marchas bípedes.

O modelo de um pêndulo invertido 3D, conforme a Figura 12, pode ser descrito (no domínio, onde $|\theta_r + \theta_p| < 0{,}5\pi$), sabendo-se que m é a massa do pêndulo, g é a aceleração gravitacional e r é o comprimento do pêndulo:

Figura 12: Pêndulo Invertido 3D.



Fonte: Dalen (2012).

Assim, tem-se equações que descrevem o comportamento do pêndulo invertido em (8), (9) e (10), onde $C_r \equiv \cos\theta_r$, $C_p \equiv \cos\theta_p$, $S_r \equiv \operatorname{sen}\theta_r$, $S_p \equiv \operatorname{sen}\theta_p$ e $D \equiv \sqrt{1 - S_r^2 - S_p^2}$. Assim, substituindo as relações cinemáticas de: $x = rsen\theta_p$, $y = -rsen\theta_r$ e $z = rD$:

$$m(-z\ddot{y} - y\ddot{z}) = \frac{D}{C_r}\tau_r - mgy \tag{8}$$

$$m(z\ddot{x} - y\ddot{z}) = \frac{D}{C_p}\tau_p + mgx \tag{9}$$

$$m(x\ddot{x} + y\ddot{y} + z\ddot{z}) = rf - mgz \tag{10}$$

Entretanto, dessa forma o comportamento do pêndulo é não-linear e demasiado complexo para gerar o movimento da caminhada simplificado. Logo, uma restrição é imposta para limitar movimento do pêndulo invertido para que seja possível gerar o movimento da caminhada. Essa restrição limita o movimento do centro de massa no plano com um vetor normalizado $(k_x, k_y, -1)$. Assim, tem-se (11) e (12):

$$z = k_x x + k_y y + z_c \tag{11}$$

$$\ddot{z} = k_x\ddot{x} + k_y\ddot{y} \tag{12}$$

Onde z_c é uma constante positiva que indica a altura do plano e a partir dessa restrição a segunda derivada pode satisfazer (11), conforme (12). Para se obter as funções lineares da dinâmica do pêndulo invertido, substitui-se essa restrição nas equações (13), (14) e (15), assim:

$$\ddot{x} = \frac{g}{z_c}x - \frac{k_y}{z_c}(x\ddot{y} - \ddot{x}y) + \frac{1}{mz_c}u_p \tag{13}$$

$$\ddot{y} = \frac{g}{z_c}y - \frac{k_x}{z_c}(x\ddot{y} - \ddot{x}y) - \frac{1}{mz_c}u_r \tag{14}$$

$$\ddot{z} = k_x \ddot{x} - k_y \ddot{y} \tag{15}$$

Sendo $u_r = \frac{D}{C_r}\tau_r$ e $u_p = \frac{D}{C_p}\tau_p$ as entradas virtuais introduzidas. Para simplificar o movimento do pêndulo e encontrar as equações lineares é assumido que o robô bípede está caminhando apenas sobre uma superfície plana, e assim pode ser considerado que $k_x = 0$ e $k_y = 0$, onde obtêm-se as equações (16), (17) e (18):

$$\ddot{x} = \frac{g}{z_c} x + \frac{1}{m z_c} u_p \tag{16}$$

$$\ddot{y} = \frac{g}{z_c} y - \frac{1}{m z_c} u_r \tag{17}$$

$$\ddot{z} = 0 \tag{18}$$

Assim, a posição do ZMP é dada por (19) e (20), e ao se substituir em (16) e (17), as equações obtidas, (21) e (22), são as mesmas utilizadas no *cart-table model* conforme a Figura 11.

$$zmp_x = -\frac{\tau_y}{mg} \tag{19}$$

$$zmp_y = -\frac{\tau_x}{mg} \tag{20}$$

$$\ddot{x} = \frac{g}{z_c}(x - zmp_x) \tag{21}$$

$$\ddot{y} = \frac{g}{z_c}\left(y - zmp_y\right) \tag{22}$$

Assim é possível encontrar a relação entre o ZMP (utilizando *cart-table model*) e o pêndulo invertido (LIPM), que é a forma simplificada da caminhada humana, a partir de equações linearizadas. Manipulando as equações (21) e (22) é possível gerar as trajetória para o robô bípede.

Conforme Erbadur e Kurt (2009), na criação de trajetória, o ZMP durante o movimento dos passos da marcha é mantido fixo no centro da sola do pé de suporte, enquanto a pélvis (COM) segue o caminho do LIPM. Assim, será utilizado a partir de agora $x = c_x$ e $y = c_y$, pois são os pontos do COM, e $zmp_x = p_x$ e $zmp_y = p_y$. Portanto, a partir de (21) e (22) tem-se as equações dos pontos de referência de ZMP e do COM (23) e (24):

$$p_x = c_x - \frac{z_c}{g}\ddot{c_x} \tag{23}$$

$$p_y = c_y - \frac{z_c}{g}\ddot{c_y} \tag{24}$$

2.3.4 Geração das Trajetória para a Pélvis e para os Pés

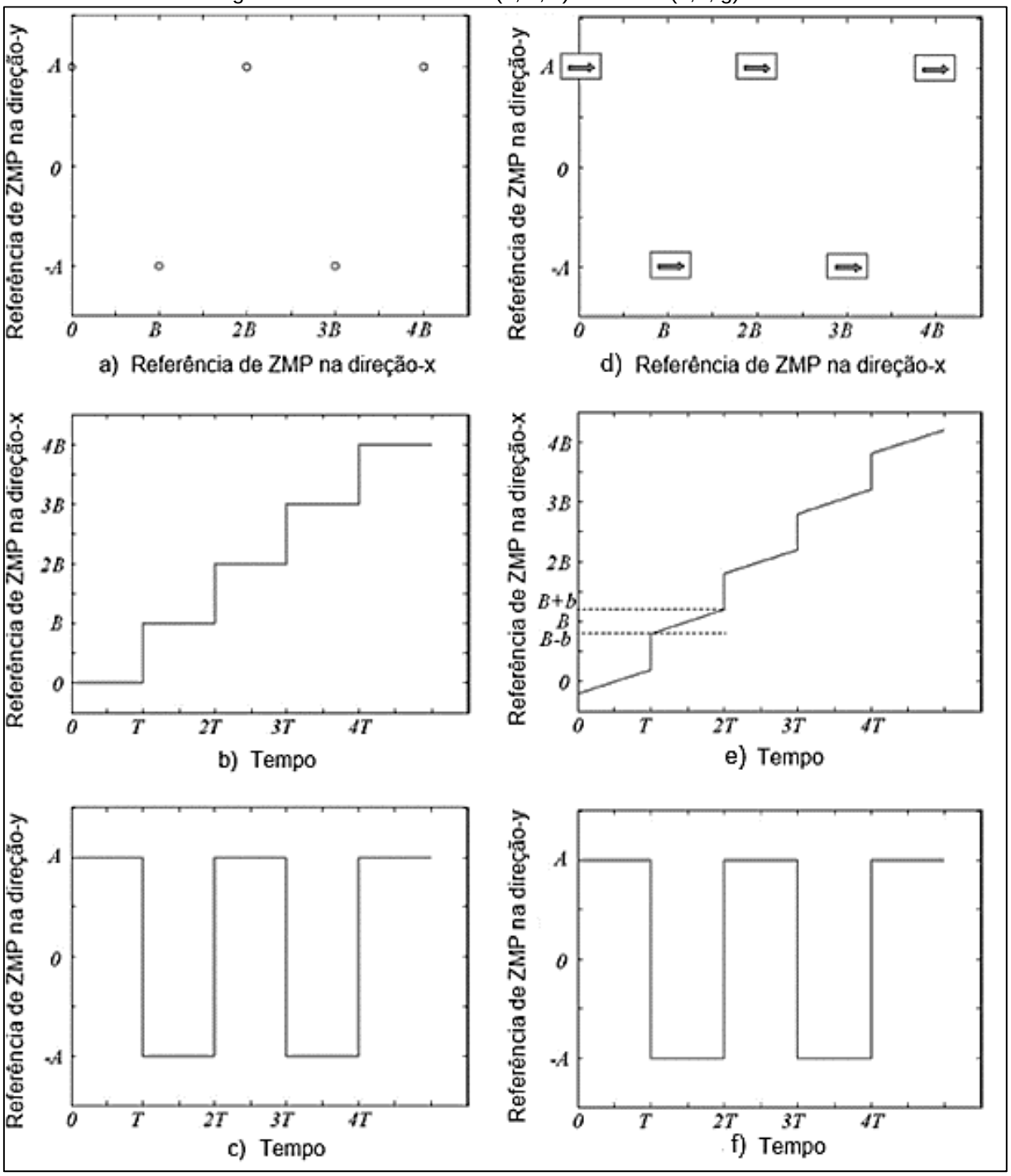
Conforme apresentado no item 2.3.2 e 2.3.3, com a relação entre o ZMP e o LIPM é possível então traçar os pontos da trajetória para os pés e para pélvis de um robô bípede suprimindo os aspectos da dinâmica. Erbatur e Kurt (2009) faz a ressalva de que, idealmente, essa aproximação com o LIPM, considera-se um robô com pernas de massa muito inferior ao restante da massa do robô, mas que o método ainda se aplica a pernas pesadas. Portanto, a pélvis será considerada com o COM e a partir de (23) e (24) é possível extrair os pontos de referência para o ZMP e deste para o COM. Assim a única exigência para o equilíbrio físico do robô é que o ZMP deve sempre estar dentro da área da sola do pé que toca o chão (ERBATUR e KURT, 2009).

Portanto, baseado nisso, é apresentado conforme na Figura 13 (a, b, c), onde A é a distância entre os centros dos pés em y, B é o comprimento do passo e T é a metade do período da marcha. De acordo com Erbatur e Kurt (2009), a seleção das posições dos passos pode ser baseada no tamanho do robô e na natureza da marcha desempenhada. T pode ser definido pelo tamanho físico do robô e suas propriedades.

Conforme Erbatur e Kurt (2009) para se obter uma marcha mais aproximada à naturalidade humana, é feito que o ZMP não seja fixo à sola do pé do robô e sim, seja móvel, conforme a Figura 13 (d, e, f). O parâmetro b, portanto, define o alcance do

ZMP embaixo da sola do pé em SSP. Ele também pode ser interpretado como metade do comprimento do pé, mesmo que isso não seja algo definitivo.

Figura 13: Referências fixas (a, b, c) e móveis (d, f, g) de ZMP.



Fonte: Adaptado de Erbatur e Kurt (2009)

Assim, de acordo com a Figura 13 e conforme Erbatur e Kurt (2009), as equações que descrevem $p_x^{ref}(t)$ e $p_y^{ref}(t)$ podem ser descritas como (25) e (26):

$$p_x^{ref} = \frac{2b}{T}\left(t - \frac{T}{2}\right) + (B - 2b)\sum_{k=1}^{\infty} u(t - kT_o) \tag{25}$$

$$p_y^{ref} = Au(t) + 2A\sum_{k=1}^{\infty} u(t - kT_o) \tag{26}$$

Considerando z_c como constante para o CoM, e $\omega_n = \sqrt{\frac{g}{z_c}}$ como frequência natural do pêndulo tem-se (27) e (28):

$$\ddot{c}_x^{ref} = \omega_n^2 c_x^{ref} - \omega_n^2 p_x^{ref} \tag{27}$$

$$\ddot{c}_y^{ref} = \omega_n^2 c_y^{ref} - \omega_n^2 p_y^{ref} \tag{28}$$

E a partir de (25) e (26), juntamente com (27) e (28), aplicando-se a transformada de Laplace, COM também pode ser dado (29) e (30):

$$\ddot{c}_x^{ref}(s) = \frac{1}{1 - \left(\frac{1}{\omega_n^2}\right)s^2}\left[p_x^{ref}(s) - \frac{1}{\omega_n^2}c_x^{ref}(0)s - \frac{1}{\omega_n^2}\dot{c}_x^{ref}(0)\right] \tag{29}$$

$$\ddot{c}_y^{ref}(s) = \frac{1}{1 - \left(\frac{1}{\omega_n^2}\right)s^2}\left[p_y^{ref}(s) - \frac{1}{\omega_n^2}c_y^{ref}(0)s - \frac{1}{\omega_n^2}\dot{c}_y^{ref}(0)\right] \tag{30}$$

O desenvolvimento completo utilizando aproximação por Fourier apresenta-se em Erbatur e Kurt (2009), p. 5. Na referência é possível encontrar para o COM então (31) e (32):

$$c_x^{ref} = \frac{B}{T}\left(t - \frac{T}{2}\right) + \sum_{k=1}^{\infty} \frac{(B - 2b)T^2\omega_n^2}{k\pi(T^2\omega_n^2 + k^2\omega_n^2)} \sin\left(\frac{2\pi kt}{T}\right) \tag{31}$$

$$c_y^{ref} = \sum_{k=1}^{\infty} \frac{2AT^2\omega_n^2(1 - cosk\pi)}{k\pi(T^2\omega_n^2 + k^2\omega_n^2)} sin\left(\frac{2\pi kt}{2T}\right) \tag{32}$$

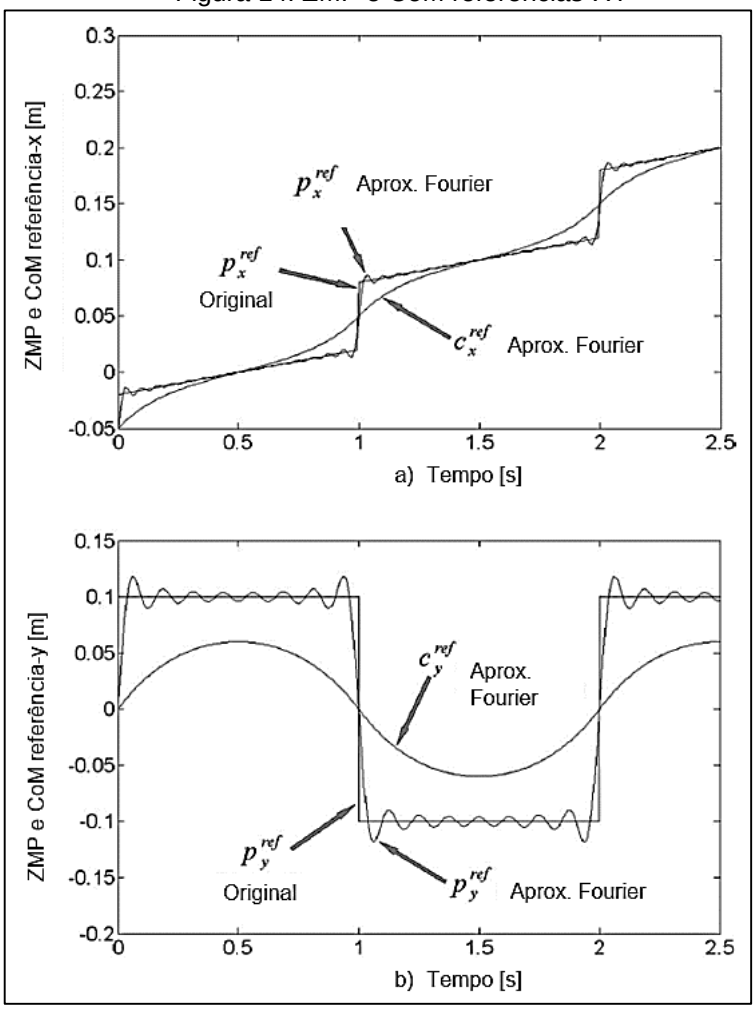
E para a referência do ZMP, tem-se (33) e (34):

$$p_x^{ref} = \frac{B}{T}\left(t - \frac{T}{2}\right) + \sum_{k=1}^{\infty} \frac{(B-2b)T^2\omega_n^2}{k\pi(T^2\omega_n^2 + k^2\omega_n^2)}\left(1 + \frac{\pi^2k^2}{\omega_n^2T^2}\right)\sin\left(\frac{2\pi kt}{T}\right) \quad (33)$$

$$p_y^{ref} = \sum_{k=1}^{\infty} \frac{2AT^2\omega_n^2(1 - cosk\pi)}{k\pi(T^2\omega_n^2 + k^2\omega_n^2)}\left(1 + \frac{\pi^2k^2}{\omega_n^2T^2}\right)sin\left(\frac{2\pi kt}{2T}\right) \quad (34)$$

Erbatur e Kurt (2009) utilizaram em seu trabalho os valores $A = 0{,}1; B = 0{,}1; b = 0{,}02; T = 1$. Outras literaturas destacadas em Erbatur e Kurt (2009) informaram que para o somatório é possível aproximar para $N = 15$. Portanto, as curvas que serão esperadas neste trabalho deverão ser semelhantes às apresentadas na Figura 14 que correspondem a (31), (32), (33) e (34):

Figura 14: ZMP e CoM referências-XY



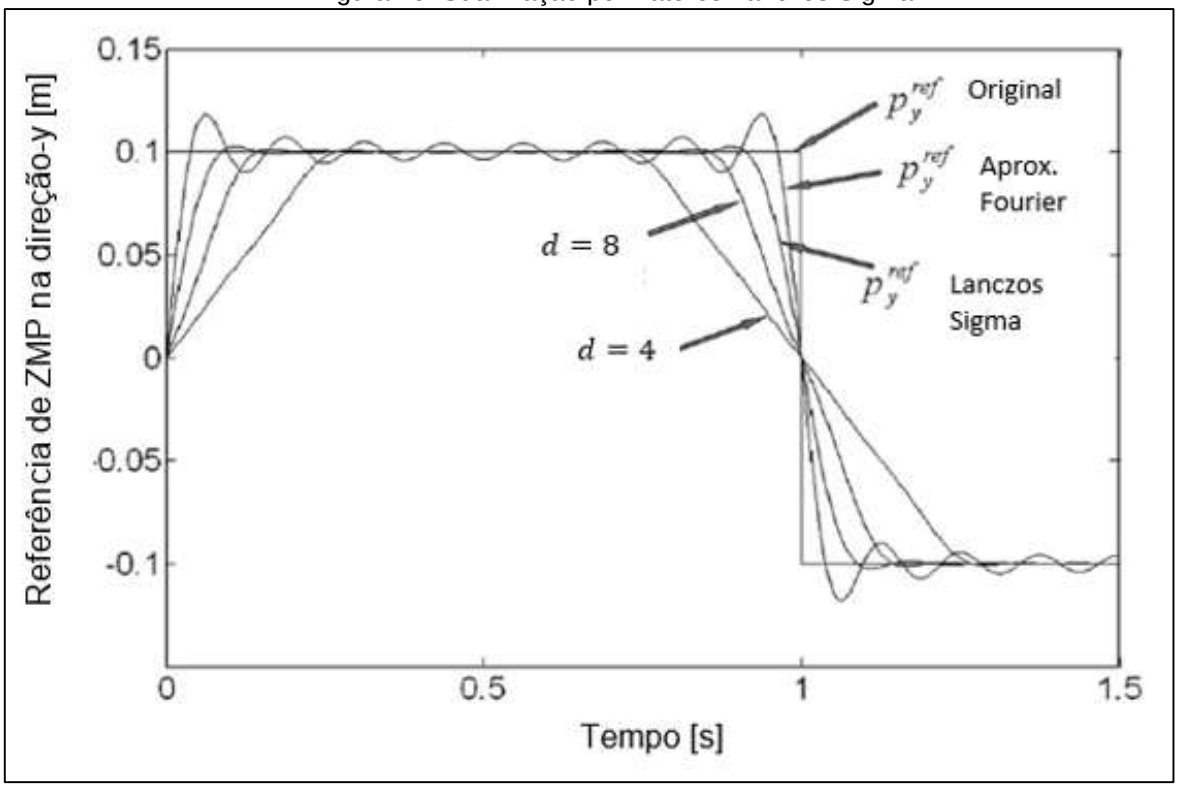
Fonte: Adaptado de Erbatur e Kurt (2009).

Ademais, ainda é necessário estabelecer a DSP (*Double Support Phase*), que pode ser alcançada através de uma função de suavização (ERBATUR e KURT, 2009). Como a Série de Fourier cria picos, pois sua convergência não é uniforme (conforme percebe-se Figura 14) é, então, utilizado Fatores Lanczos Sigma para suavizá-los em (35):

$$sinc\left(\frac{k\pi}{N}\right) = \frac{sin\left(\frac{k\pi}{N}\right)}{\frac{k\pi}{N}} \tag{35}$$

Onde N é o número dos termos da Fourier acrescido de “1”. Aplica-se então a suavização de *Lanczos Sigma* dentro do somatório da série de Fourier em (31), (32), (33) e (34), conforme explica Erbatur e Kurt (2009). Entretanto, esses fatores cumprem outro papel, também, que é o de estabelecer a DSP. Substituindo-se N por d, tem-se o comportamento visto na Figura 15. Quanto menor o valor de d maior o período da DSP.

Figura 15: Suavização por Fatores Lanczos Sigma



Fonte: Adaptado de Erbatur e Kurt (2009).

2.3.5 Pontos de Trajetória do Pé Suspenso

Possuindo os pontos da trajetória para o COM (que corresponde à pélvis), assim como a trajetória de referência de ZMP para os pontos no solo, calcula-se, agora, os pontos da trajetória para os pés.

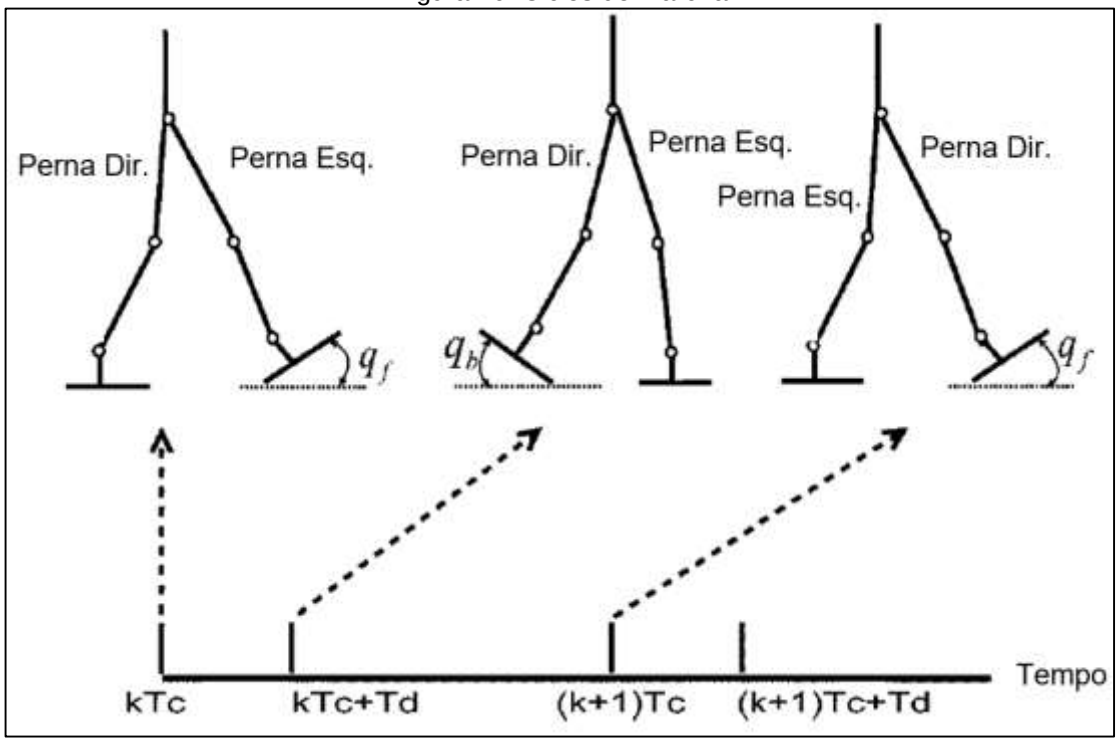
Conforme, Huang, et al. (2001), a marcha bípede é um fenômeno periódico que pode ser dividividido incialmente em SSP e DSP, como já mencionado anteriormente. Como se sabe, a DSP ocorre quando os dois pés estão em contato com o chão. Idealmente, inicia-se quando o calcanhar (humano) toca o chão, e termina quando o dedão do pé (humano) deixa o chão, explica Huang, et al. (2001). Na SSP, enquanto um pé serve de suporte, o outro pé, suspenso, move-se para frente para completar um passo. Isso foi mostrado na Figura 5.

Na robótica, explica Huang, et al. (2001), muitas trajetórias geradas para robôs bípedes acabam por assumir uma DSP instantânea. Isso, além de assemelhar-se menos com uma marcha humana real (ERBATUR e KURT, 2009), obriga que o quadril do robô mova-se muito rapidamente afetando seu equilíbrio físico (HUANG, et al. 2001). Portanto, da mesma forma que Erbatur e Kurt (2009), Huang, et al. (2001), assumiu-se que a DSP não é instantânea e afirma que a DSP de um ser humano é em média 20% do período total da marcha humana. Assim, para este trabalho, visando alcançar uma semelhança com a caminhada humana e prezando pelo equilíbrio físico, esses mesmos princípios serão adotados.

Assumindo que o período necessário para um passo de marcha é T_c, o tempo para k passos é de kT_c até $(k+1)T_c$. T_d é o intervado na DSP. $kT_c + T_m$ é o momento em que o pé encontra-se mais alto. Nesse caso, por simplificação, explica Huang, et al. (2001), foi analizado uma marcha já em andamento e não o início ou o final de uma, cujos pés idealmente começariam e terminariam exatamente em paralelo um ao outro. Por serem movimentos cíclicos, o que for apresentado para uma perna servirá para outra também (HUANG, el al. 2001).

Dessa forma, Huang, et al. (2001) estabelece intervalos para o ciclo marcha conforme conforme a Figura 16.

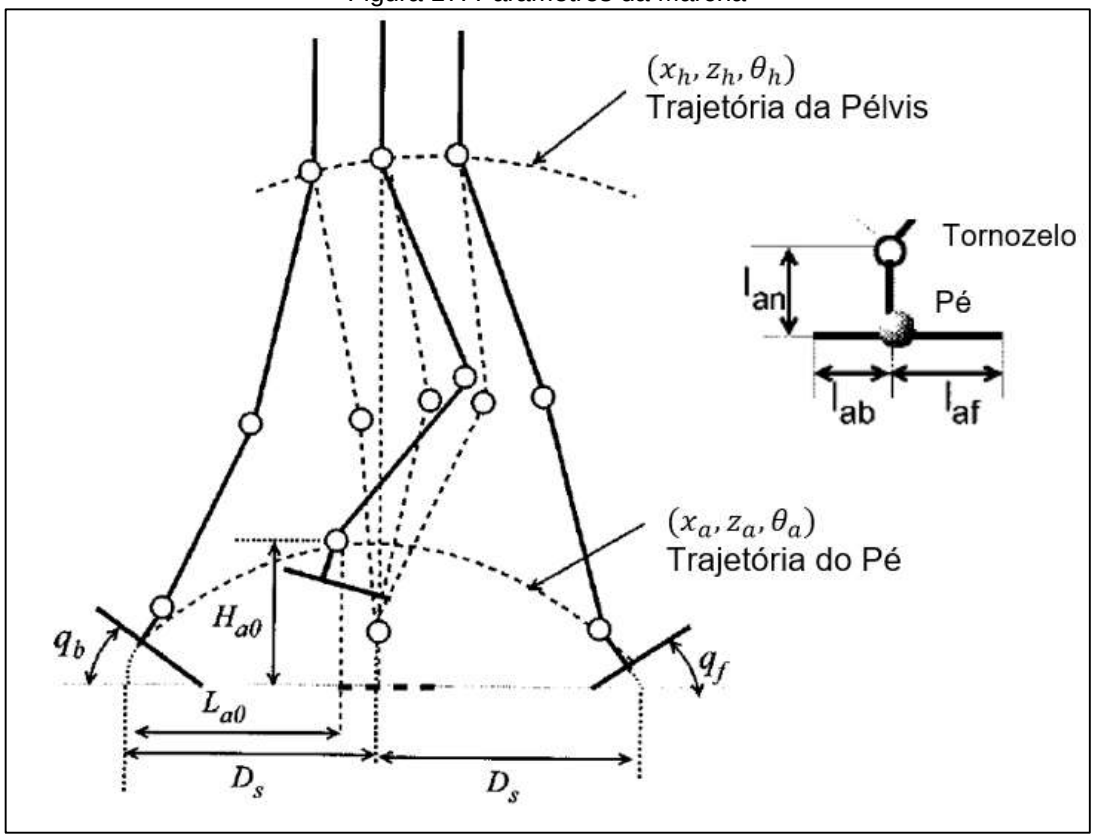
Figura 16: Ciclos de Marcha



Fonte: Adaptado de Huang, et al. (2001).

Conforme a imagem Figura 17, os parâmetros da marcha podem ser definidos.

Figura 17: Parâmetros da Marcha



Fonte: Adaptado de Huang, et al. (2001).

Sendo q_b e q_f os ângulos de “chegada” e “saída” do pé no solo, respectivamente. L_{ao} e H_{ao} são, respectivamente, a posição em X e Z do ponto mais alto do pé suspenso. D_s é o comprimento de um passo. l_{an} é a altura do pé, comprimento até o tornozelo. l_{af} é o comprimento do centro do pé ao “dedão”. l_{ab} é o comprimento do centro do pé ao calcanhar. O equacionamento pode ser dado em (36), (37) e (38)

$$\theta_a(t) = \begin{cases} q_{gs}(k) & t = kT_c \\ q_b & t = kT_c + T_d \\ -q_f & t = kT_c \\ -q_{ge}(k) & t = kT_c \end{cases} \tag{36}$$

$$x_a(t) = \begin{cases} kD_s & t = kT_c \\ kD_s + l_{an}sen(q_b) + l_{af}(1 - cos(q_b) & t = kT_c + T_d \\ kD_s + L_{ao} & t = kT_c + T_m \\ (k+2)D_s - l_{an}sen(q_f) - l_{ab}(1 - cos(q_f) & t = (k{+}1)T_c \\ (k+2)D_s & t = (k{+}1)T_c + T_d \end{cases} \tag{37}$$

$$z_a(t) = \begin{cases} h_{gs}(k) + l_{an} & t = kT_c \\ h_{gs}(k) + l_{af}sen(q_b) + l_{an}cos(q_b) & t = kT_c + T_d \\ H_{ao} & t = kT_c + T_m \\ h_{gs}(k) + l_{ab}sen(q_f) + l_{an}cos(q_f) & t = (k{+}1)T_c \\ h_{ge}(k) + l_{an} & t = (k{+}1)T_c + T_d \end{cases} \tag{38}$$

Uma vez que a sola estará completamente em contato com o solo, considera-se os ângulos da inclinação do solo como $q_{gs} = q_{ge} = 0$. Em pelo menos $t = kT_c$ e $t = (k+1)T_C + T_d$ as condições de (39), (40) e (41) devem ser satisfeitas (HUANG, el al. 2001). No caso deste trabalho, não serão somente essas condições que deverão ser satisfeitas uma vez que por simplifcação pretende-se utilizar $q_b = q_f = 0$, conforme utiliza Olcay e Ozkurt (2017). Vale notar que os pontos são passados para os tornozelos e não diretamente para os pés. O ângulo dos pés são dados por $\theta_a(t)$.

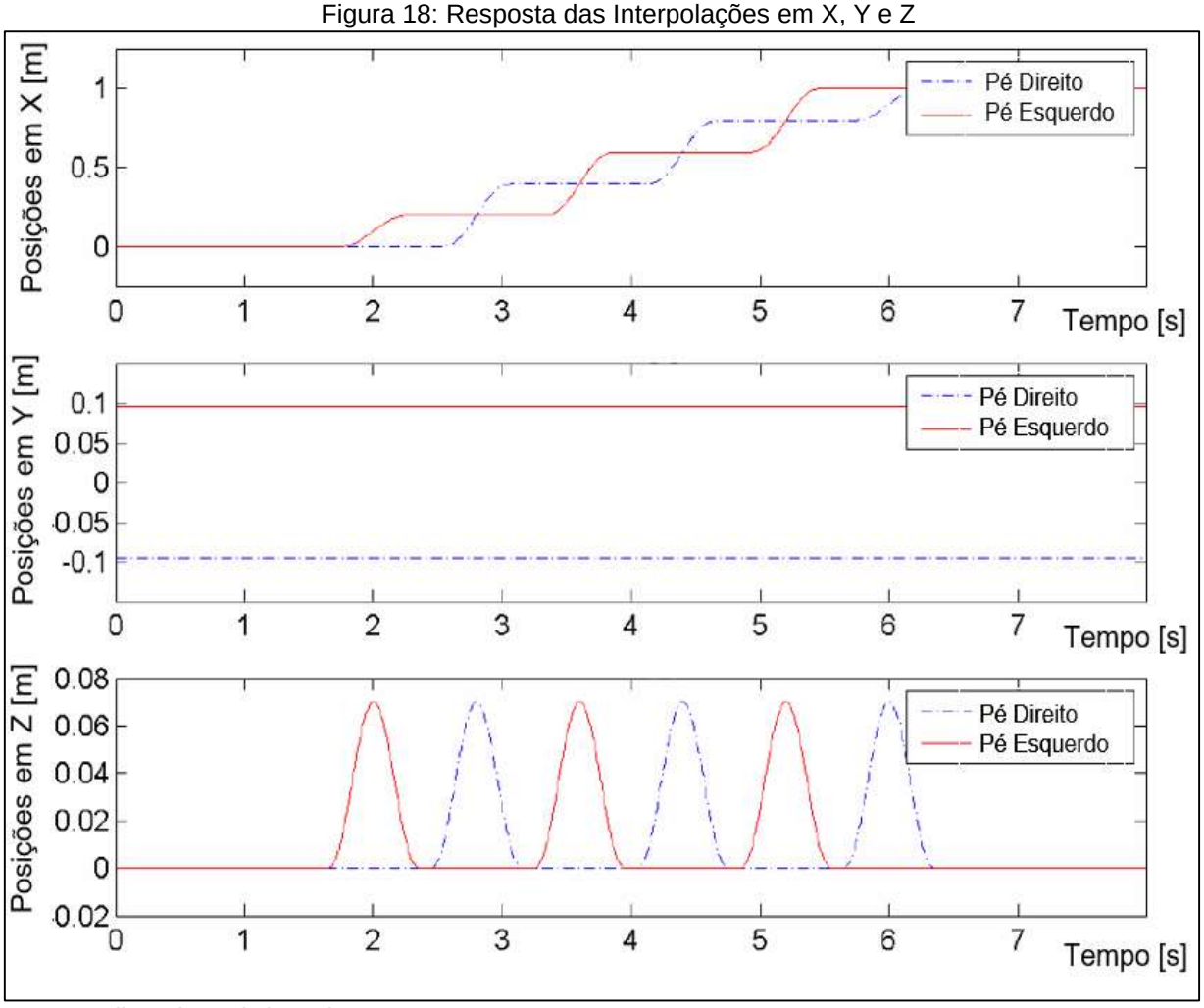
$$\theta_a(t) = \begin{cases} \dot{\theta}_a(kT_c) = 0 \\ \dot{\theta}_a((k+1)T_c + T_d) = 0 \end{cases} \tag{39}$$

$$x_a(t) = \begin{cases} \dot{x}_a(kT_c) = 0 \\ \dot{x}_a((k+1)T_c + T_d) = 0 \end{cases} \tag{40}$$

$$z_a(t) = \begin{cases} \dot{z}_a(kT_c) = 0 \\ \dot{z}_a((k+1)T_c + T_d) = 0 \end{cases} \quad (41)$$

Para gerar uma trajetória suavizada é necessário que a primeira derivada (velocidade), em termos de $\dot{x}_a(t), \dot{z}_a(t) e\ \dot{\theta}_a(t)$, seja diferencial e a segunda derivada (aceleração), em termos de $\ddot{x}_a(t), \ddot{z}_a(t) e\ \ddot{\theta}_a(t)$, seja contínua em t, incluindo a transição de um intervalo para o outro. Para satisfazer de (36)-(41), a interpolação será de uma ordem elevada dificultando sua computação. Porém, pode ser utilizado uma Interpolação *Spline* de Terceira Ordem (*Cubic Spline*). Entretanto, Sellaouti, et al. (2006), utilizou polinômios de quinta ordem para X e Y e de sexta ordem para Z. A Figura 18 mostra as repostas obtidas para seu caso, onde seus parâmetros foram: Largura do Passo = $0{,}2\ [m]$; Altura do Passo = $0.07\ [m]$; Altura do COM =$0{,}814\ [m]$; Ciclo = $0{,}8\ [seg]$; SSP = $0{,}75\ [seg]$; DSP = $0{,}05\ [seg]$.

Figura 18: Resposta das Interpolações em X, Y e Z



Fonte: Sellaouti, et al. (2006).

Dadas as devidas proporções relacionadas aos parâmetros de cada robô, as respostas em Huang, et al. (2001), Olcay e Ozkurt (2017), e Kim, Park e Oh (2007) foram semelhantes. Para todos os casos foram mantidas as posições em Y constantes. A trajetória da Pélvis – COM, já foi definida no item 2.3.3 – e para o caso apresentado neste item z_h, conforme a Figura 17 pode ser considerado constante.

2.4 MÉTODO DE CINEMÁTICA INVERSA

A Cinemática Inversa é responsável por encontrar os ângulos que cada junta deve alcançar para encontrar um dado ponto cartesiano. Portanto, é necessário contextualizá-la através dos fundamentos da robótica.

Os fundamentos da robótica incluem a parte descritiva de pontos e objetos no espaço dada através de matrizes. Estabelecendo-se eixos de referência e utilizando translações e rotações matriciais (transformações), é possível representar a estrutura de elos e juntas um robô, de forma matricial, e assim calcular o comportamento cinemático de um robô.

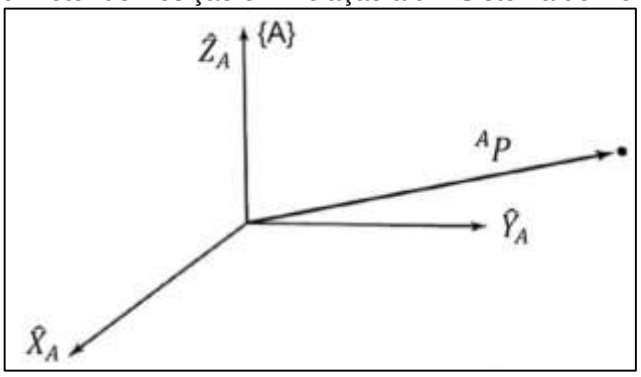
2.4.1 Descrições Espaciais e Transformações Matriciais

Uma vez que a finalidade para o robô bípede é sua a locomoção, faz-se necessário representá-lo num plano espacial. Assim começando apenas com a descrição de um ponto, conforme Craig (2012), de uma posição tem-se (42), onde ${}^{A}P$ é o vetor de posição (3×1). Dessa forma, com um sistema de coordenadas é possível descrever qualquer ponto no universo.

A notação utilizada, provinda de Craig (2012), especifica que $\{A\}$ corresponde a um sistema de referência e que, portanto, ${}^{A}P$ são as distâncias ao longo dos eixos desse sistema de referência (p_x, p_y e p_z), onde $\hat{X}_A$, $\hat{Y}_A$ e $\hat{Z}_A$, são os eixos ortogonais do sistema de referência de $\{A\}$, conforme a Figura 19.

$$ {}^{A}P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{42} $$

Figura 19: Vetor de Posição em Relação a um Sistema de Referência.

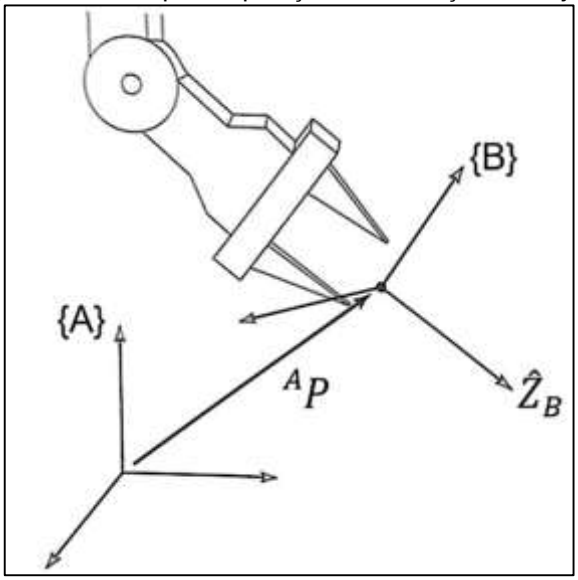


Fonte: Craig (2012)

A descrição da orientação é dada pelos próprios eixos de um sistema de referência, uma vez que, conforme a Figura 20, por exemplo, um novo sistema de referência $\{B\}$ foi fixado a um braço robótico. Dessa forma, é apresentada a sua orientação em relação ao sistema de referência do espaço $\{A\}$. Assim, quando um sistema de referência é descrito em relação a outro, utiliza-se ${}^A\hat{X}_B$, ${}^A\hat{Y}_B$ e ${}^A\hat{Z}_B$. Portanto, para descrever essa relação em uma única matriz, tem-se (43), onde ${}^A_B R$, é o rotacional de $\{B\}$ em relação para $\{A\}$.

$$ {}^A_B R = [\,{}^A\hat{X}_B \;\; {}^A\hat{Y}_B \;\; {}^A\hat{Z}_B] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{43} $$

Figura 20: Exemplo de posição e orientação de objeto.



Fonte: Craig (2012).

Portanto, um conjunto de quatro vetores é capaz de especificar a posição e a orientação conforme visto em (42) e (43). Assim, a nova estrutura é de uma matriz de 4×4, conforme (44), de modo que uma linha $[0\ 0\ 0\ 1]$ é adicionada na última linha de ${}^{A}_{B}R$ e ${}^{A}P$, e é chamada de transformação homogênea.

$$ {}^{A}_{B}T = \begin{bmatrix} {}^{A}_{B}R & & & {}^{A}P \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{44} $$

Assim, conforme Craig (2012), colocando-se em estrutura de matriz de transformação, tem-se que, (45) é a matriz de translação e (46), (47) e (48) são as matrizes de rotação em θ, para os eixos X, Y e Z (rotações mais comuns). Rotações em eixos genéricos não serão abordadas nesse trabalho, pois as rotações em X, Y e Z suprem a demanda dos cálculos matemáticos da implementação a ser proposta.

$$ P_q(\theta) = \begin{bmatrix} 1 & 0 & 0 & p_x \\ 0 & 1 & 0 & p_y \\ 0 & 0 & 1 & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{45} $$

$$ R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos\theta & -sen\theta & 0 \\ 0 & sen\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{46} $$

$$ R_Y(\theta) = \begin{bmatrix} cos\theta & 0 & sen\theta & 0 \\ 0 & 1 & 0 & 0 \\ -sen\theta & 0 & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{47} $$

$$ R_Z(\theta) = \begin{bmatrix} cos\theta & -sen\theta & 0 & 0 \\ sen\theta & cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{48} $$

Dessa forma, percebe-se que as transformações englobam as rotações e translações (49). Conforme destaca Craig (2012), elas podem definir a descrição de um sistema de referência a partir de um sistema inicial e também agir como operadoras de transformação de um sistema de eixos para outro.

$$T(\theta) = \begin{bmatrix} r_{11} & r_{12} & r_{13} & q_x \\ r_{21} & r_{22} & r_{23} & q_y \\ r_{31} & r_{32} & r_{33} & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (49)$$

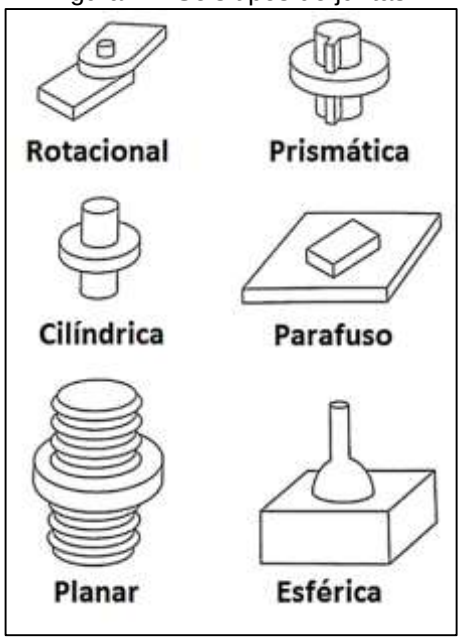
Assim, é possível efetuar transformações compostas conforme (50), essas serão importantes na etapa de descrição de elos e Cinemática Inversa para se descobrir a posição final de um manipulador ou determinados ângulos de juntas.

$$^{0}_{N}T = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T \ldots {}^{N-1}_{N}T \quad (50)$$

2.4.2 Elos e Juntas

Após as descrições espaciais e transformações, é necessário descrever os elos e as conexões desses elos (juntas), uma vez que os conceitos tratados nesse trabalho terão essas definições como base. Portanto, na robótica industrial, segundo Craig (2012), um conjunto de corpos conectados em cadeia pode ser chamado de manipulador e esses corpos chamados de elos. As juntas, por sua vez, conectam um par de elos vizinhos. Assim: “[...] um elo é considerado apenas um corpo rígido que define a relação entre os eixos de duas juntas, vizinhas, de um manipulador.” (CRAIG, 2012, p. 61).

Figura 21: Seis tipos de juntas.

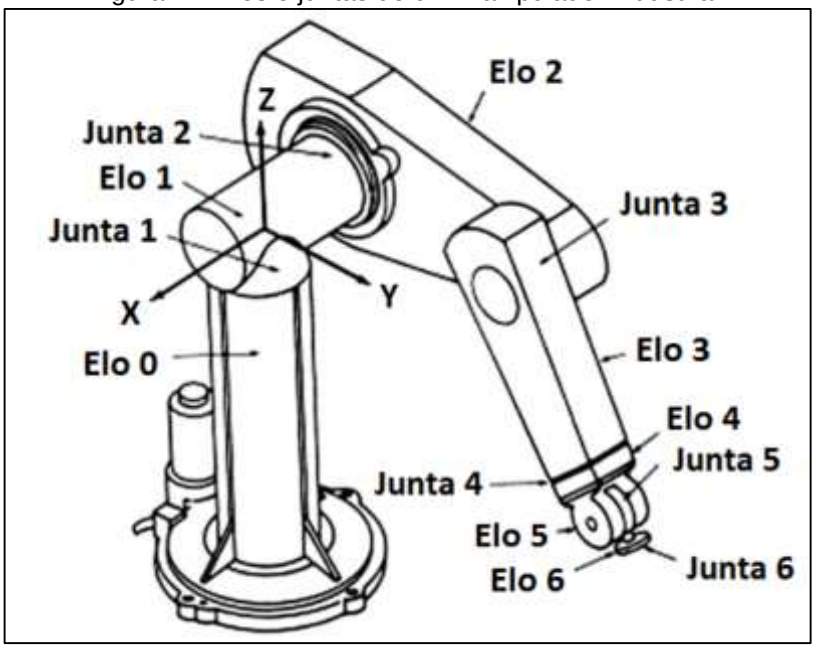


Fonte: Adaptado de Craig (2012).

Cada uma das pernas robóticas poderá ser tratada da mesma forma, considerando as devidas limitações ao imitar os movimentos de uma perna humana. Na Figura 21, são apresentados seis tipos de juntas possíveis que são utilizadas em manipuladores no meio industrial. No caso de robôs bípedes, a mais utilizada para unir dois elos é a do tipo rotacional (para a implementação desse trabalho será a única).

No caso dos manipuladores que possuem uma base fixa (Figura 22), essa é tida como elo 0; primeiro corpo móvel é o elo 1, e assim por diante até a junta 6 (base do *end-effector*[2]). As pernas robóticas podem ser tratadas da mesma forma, porém considera-se que o elo 0 escolhido de uma perna seja em algum lugar correspondente ao quadril do robô.

Figura 22: Elos e juntas de um manipulador industrial.



Fonte: Adaptado de Lopes (2002).

2.4.3 Notação Denavit-Hartenberg

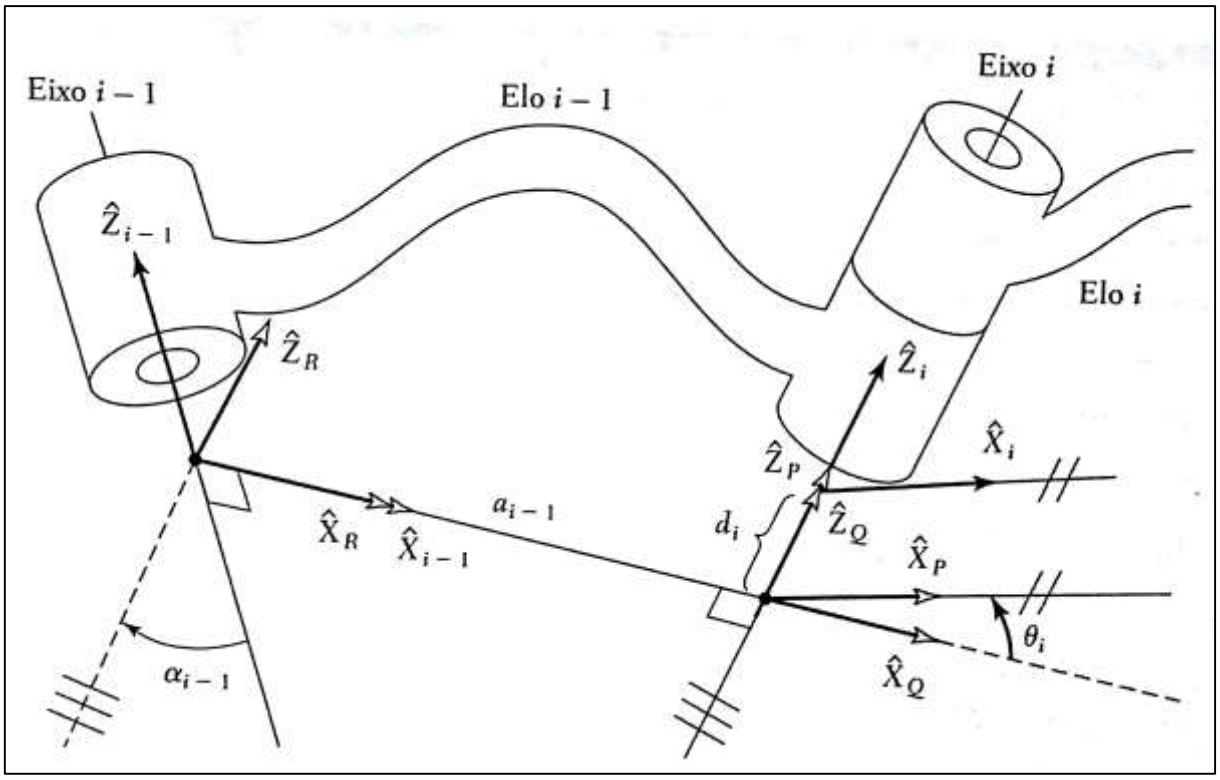
De acordo com Craig (2012), a notação de Denavit-Hartenberg (DH), pode ser definida pelos seguintes parâmetros: θ_i, α_i, a_i e d_i. Para que se compreendam os eixos das juntas podem ser definidos por linhas ou vetores direção no espaço,

[2]*End-effector* (ou efetuador), conforme Craig (2012), é a ferramenta ou dispositivo que pode ser acoplado na ponta da cadeia de elos de um manipulador. Para o caso desse trabalho será considerado como *End-effector* o extremo da cadeia de elos da perna robótica.

conforme a Figura 23. No caso, se existe um eixo i atribuído a uma junta, o elo i rotaciona em relação ao elo $i-1$.

Conforme Cudowski (2009) e Craig (2012), a_{i-1}, comprimento do elo, é a distância entre os dois eixos, ou seja, a distância $\hat{Z}_{i-1}$ e $\hat{Z}_i$ ao longo de $\hat{X}_{i-1}$. Essa linha perpendicular só é inexistente quando os eixos são paralelos, e quando os eixos não são torcidos ela é do comprimento do elo.

Figura 23: Eixo torcido para a demonstração do parâmetro Denavit-Hartenberg e sistema de referência.



Fonte: Adaptado de Craig (2012).

A torção de elo α, é o ângulo entre o eixo $\hat{Z}_{i-1}$ e $\hat{Z}_i$ em torno de $\hat{X}_{i-1}$. Quando o elo não é torcido, como no caso de alguns elos das pernas do robô bípede, esse parâmetro acaba se perdendo, pois alguns eixos das juntas ficam sobrepostos. Mas esses dois parâmetros, comprimento e torção, são necessários para definir a localização relativa de dois eixos.

O deslocamento do elo d_i é a distância medida do eixo da origem de i ao logo de $\hat{X}_i$ até chegar no eixo $\hat{Z}_i$, ou então, é a distância medida no ponto de intersecção no eixo da junta i com a_{i-1} até o ponto de intersecção de i com a_i. E o ângulo de junta θ_i é o ângulo entre $\hat{X}_{i-1}$ e $\hat{X}_i$ em torno de $\hat{Z}_{i-1}$.

Assim, dois dos parâmetros DH, descrevem o elo em si, e dois descrevem a conexão com o elo vizinho, e qualquer robô articulado pode ser descrito por esses parâmetros.

A partir desses parâmetros, também é possível descobrir o ponto onde se encontra o final do manipulador, ou que no caso de um robô bípede poderia ser a sola do pé (dependendo do sistema de referência escolhido), pois tem-se os ângulos, os comprimentos e distâncias que compõem os elos/juntas. A esse cálculo pode-se dar o nome de cinemática direta. Conforme, Craig (2012), o elo 0 (zero) é chamado de base do robô, onde é fixado o sistema de referência, que por sua vez não se move e é escolhido arbitrariamente. Portanto, o mais simples sempre é escolher Z_0.

2.4.4 Transformações dos Elos

Para que se possa manipular com os elos, faz-se necessário montar uma transformação que defina o elo. A descrição da matriz para a transformação de um elo, na forma geral, de acordo com a Figura 23, utiliza os conceitos das transformações compostas, conforme (51) e (52) para montar (53). Essa série de transformações culminando em ${}^{i-i}_{i}T$, define apenas um elo (CRAIG, 2012). Basicamente, está sendo definida a transformação de um sistema de referência para outro (de $\{i\}$ para $\{i-1\}$) fazendo uso dos quatro parâmetros DH. Conforme Craig (2012), para realizar a transformação geral é necessário realizar as transformações intermediárias dados os respectivos sistemas de referência intermediários ($\{P\}$, $\{Q\}$ e $\{R\}$).

$$ {}^{i-i}P = {}^{i-i}_{R}T\,{}^{R}_{Q}T\,{}^{Q}_{P}T\,{}^{P}_{i}T\;\;{}^{i}P \tag{51} $$

Onde:

$$ {}^{i-i}_{i}T = {}^{i-i}_{R}T\,{}^{R}_{Q}T\,{}^{Q}_{P}T\,{}^{P}_{i}T \tag{52} $$

$$ {}^{i-i}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{53} $$

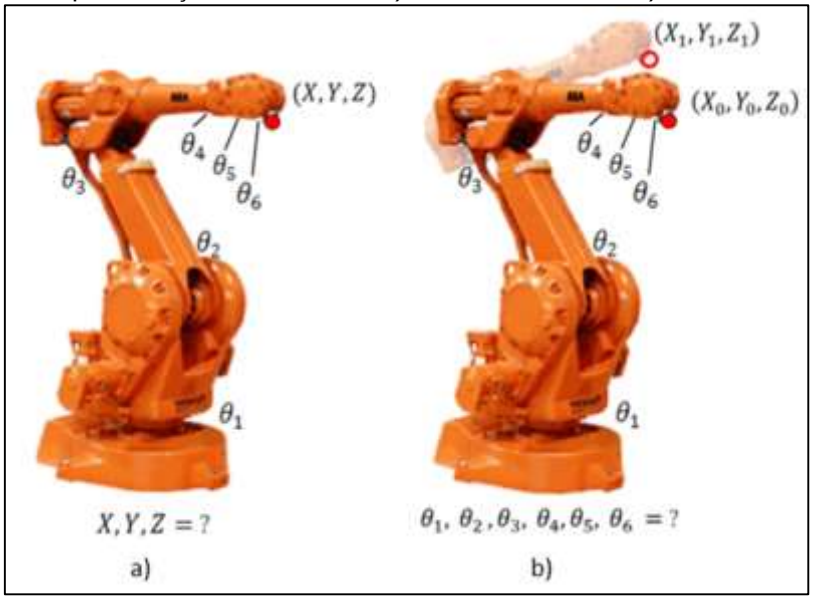
De modo que: $s\theta_i \equiv sin\theta_i$, $c\theta_i \equiv \cos\theta_i$, $s\alpha_i \equiv sin\alpha_i$, $c\alpha_i \equiv \cos\alpha_i$; e $'a', 'a', 'd'$ e $'\theta'$ são os parâmetros DH. Dessa forma, é possível lidar com as transformações de um elo para o outro, tendo-se a matriz de transformação de cada um deles, realizando transformação composta. Conforme Cudowski (2009) também é possível chegar na respectiva matriz realizando Rotações e Translações referentes a cada um dos parâmetros na ordem em que forem estabelecidos ou extraídos. Por exemplo, se fosse na ordem em que foram apresentados no item anterior teriamos, primeiramente $a_i, \alpha_i, d_i, \theta_i$, logo (54):

$$T = P_{X,a} R_{X,\alpha} P_{Z,d} R_{Z,\theta} \tag{54}$$

2.4.5 Cinemática Inversa

Conforme mencionado anteriormente, é possível saber a posição do *end-effector* de um manipulador através dos parâmetros DH: aplica-se a matriz de transformação composta sabendo-se os ângulos de cada junta. Esse processo é chamado de cinemática direta (Figura 24.a). Porém, agora, tem-se o problema inverso. Quer-se descobrir quais os ângulos necessários que cada junta deve ter para que se alcance uma determinada posição (levando em conta de que se sabem os parâmetros DH). A esse processo, chama-se de Cinemática Inversa (Figura 24.b). Esse por sua vez é muito mais complexo de ser obtido.

Figura 24: Representação da saída de: a) Cinemática direta; b) Cinemática Inversa.



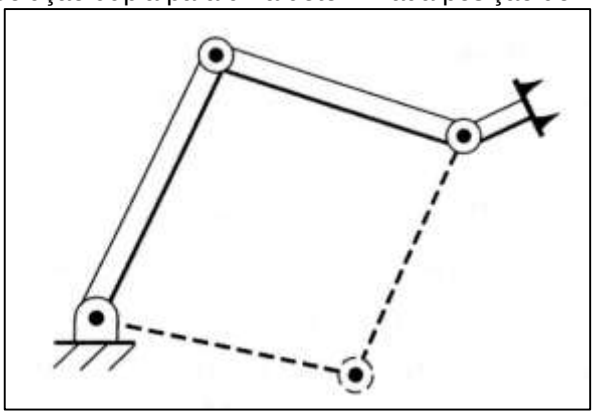
Fonte: Adaptado de Santec, 2017.

Conforme Craig (2012), é necessário resolver ${}_{N}^{0}T$, para se encontrar os valores de θ_1, θ_2 .... θ_n, para as posições desejadas, pois esse problema não é trivial, visto que as equações cinemáticas não são lineares. Também é necessário estabelecer se há uma solução para a posição desejada, ou se há múltiplas soluções, uma vez que a extremidade do manipulador pode chegar em uma mesma posição com mais de um valor de ângulo para θ_1, θ_2 .... θ_n. Logo, o robô deve ser capaz de escolher apenas uma, e para tanto é necessário ter-se um método de cálculo específico.

Para saber se existe uma solução é necessário considerar o espaço de trabalho do manipulador, que é o volume de espaço que ele pode alcançar, e isso é feito considerando que cada junta tem 360º de liberdade, o que geralmente não é verdadeiro, ou seja, é necessário levar em conta os ângulos de cada junta. E também, é necessário levar em conta o número de juntas, sabendo-se que, com menos de seis graus de liberdade o manipulador não é capaz de alcançar posições e orientações genéricas no espaço tridimensional, (CRAIG, 2012). Portanto, se a posição desejada existe dentro desse espaço, ao menos uma solução existe.

No outro extremo do problema, pode ser que mais de uma solução exista, isso pode ser exemplificado conforme Figura 25, onde a linha pontilhada apresenta uma segunda solução para uma mesma posição.

Figura 25: Solução dupla para uma determinada posição do manipulador.



Fonte: Craig (2012).

A existência de mais de uma solução é problemática, pois é necessário que seja escolhida apenas uma, e sendo assim alguns critérios podem ser tomados para que se delimitem os movimentos, como por exemplo: levar em conta a posição atual, para seguir sempre pela rota mais próxima; distribuir peso de relevância nos cálculos para as juntas; considerar obstáculos (CRAIG, 2012).

O número de soluções varia conforme os parâmetros DH e número de juntas. Quanto mais parâmetros diferentes de zero, maior o número de soluções. Um manipulador com seis graus de liberdade pode chegar até a dezesseis soluções para um mesmo ponto de destino, informa Craig (2012).

2.4.5.1 Métodos de Soluções para Cinemática Inversa

Craig (2012) explica que para que um método de cálculo seja levado em conta, ele deve ser capaz de calcular todas as soluções, e classifica os métodos entre soluções de forma fechada e soluções numéricas.

As soluções numéricas são muito mais lentas e possuem uma quantidade elevada de processamento, além de não lidarem tão bem com singularidades – pontos de mapeamento não reversíveis (CRAIG, 2012). Porém, é necessário ressaltar métodos baseados em Jacobianas que utilizam iterações para alcançar os valores dos ângulos das juntas para cada posição e que também podem lidar melhor com as singularidades. Outra que Craig (2012) enfatiza é que há apenas soluções genéricas para manipuladores de seis graus de liberdade usando-se métodos numéricos, e não para métodos de forma fechada: “Todos os sistemas com juntas rotacionais e prismáticas com um total de seis graus de liberdade em uma única cadeia seriada são solucionáveis.” (CRAIG, 2012, p. 98).

Já, as soluções de forma fechada são mais rápidas, pois convergem em expressões analíticas ou na solução de polinômios de quarto grau ou inferior e podem ser subdivididas em: método algébrico e método geométrico (CRAIG, 2012).

O método algébrico utiliza as transformações compostas vistas até então para se encontrar a transformação total. Dela são extraídas as equações para se encontrar os ângulos das juntas.

O método geométrico faz a decomposição da geometria espacial do manipulador em várias equações de geometria plana e a partir daí extraem-se os ângulos das juntas. Entretanto, de acordo com Craig (2012) não há soluções de forma-fechada genéricas, e somente há soluções para manipuladores com um número de até a quatro graus de liberdade, para manipuladores que sejam simples o bastante. Superior a quatro graus é possível calcular somente em casos específicos. Utilizando-se desses casos específicos é que projetistas de manipuladores constroem robôs industriais.

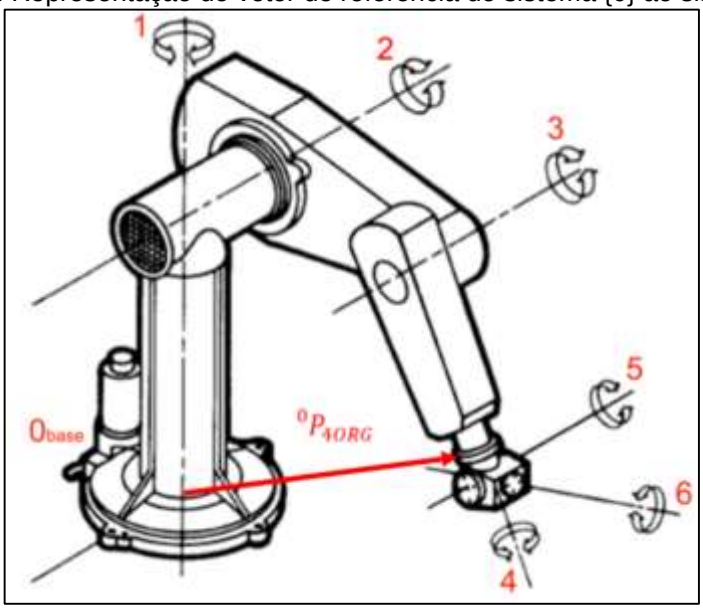
2.4.5.2 Método de Forma-Fechada de Pieper

Inicialmente, tinha-se a proposta de utilizar o Método de Forma-Fechada de Pieper. Portanto, ainda será apresentado mesmo que não tenha sido mais utilizado, para servir de contraste com o método iterativo por Jacobianas que ao fim foi escolhido e implementado. Segundo Craig (2012), o método de Pieper permite a solução da Cinemática Inversa de forma-fechada para um caso específico. Para este trabalho seria visto somente a solução para quando tem-se seis DOF, onde os três eixos das três últimas juntas se cruzam e para o caso onde todas as juntas são rotacionais. Esse método é muito relevante, pois: “O trabalho de Pieper se aplica à maioria dos robôs industriais disponíveis no mercado.” (CRAIG, 2012, p. 107).

O método de Pieper se aplica aos manipuladores industriais, já muito conhecidos, de modo que, e poderia ser utilizado para encontrar as soluções para os ângulos das pernas do robô bípede, como no caso aplicado ao robô KHR-4 (HUBO 2).

Como exemplo na Figura 26 é apresentado um manipulador industrial cuja estrutura é igual à requerida para aplicar o método de Pieper. Como pode ser observado, o manipulador possui seis juntas rotacionais, e os eixos as últimas três juntas se sobrepõe.

Figura 26: Representação do vetor de referência do sistema {0} ao sistema {4}.



Fonte: Adaptado de Universidad de Santiago de Chile Virtual, 2017.

O método concebido por Pieper aplica um desacoplamento entre a posição e a rotação (CRAIG, 2012). Após efetuar a transformada composta final de todos os elos,

o vetor da posição é equacionado separadamente da matriz (3x3) de rotação. Isso é verificável pela própria estrutura apresentada na Figura 26, percebe-se que as últimas três juntas não influenciam, praticamente, na posição final do *end-effector.* Entretanto, elas influenciam na orientação final. Dessa forma, primeiramente, é trabalhado apenas com o vetor responsável pela posição a fim de se obter as três primeiras variáveis: ângulos θ_1, θ_2 e θ_3 (correspondentes a {1}, {2}, {3}, respectivamente). E para isso deseja-se descobrir ${}^0P_{4ORG}$, que é um vetor situado na base $\{0\}$ que localiza a posição de origem do sistema de referência $\{4\}$, uma vez que esse eixo, também é o eixo dos sistemas de referência $\{5\}$ e $\{6\}$.

Conforme a Figura 26, esses três sistemas de referência representam as três últimas juntas. Decompondo ${}^0P_{4ORG}$, nas transformações compostas dos sistemas de referência {0}, {1}, {2}, {3}, e apenas o vetor de posição de {3} até a origem de $\{4\}$, ${}^3P_{4ORG}$. Assim tem-se (55):

$$ {}^0P_{4ORG} = {}^0T_1 \, {}^1T_2 \, {}^2T_3 \, {}^3P_{4ORG} = \begin{bmatrix} p_x \\ p_y \\ p_z \\ 1 \end{bmatrix} \tag{55} $$

O vetor ${}^3P_{4ORG}$ pode ser representado como a coluna da posição (p) da matriz de transformação de um elo genérico ${}^{i-1}_{i}T$, apresentada em (54), e assim tem-se (56):

$$ {}^0P_{4ORG} = {}^0T_1 \, {}^1T_2 \, {}^2T_3 \begin{bmatrix} a_3 \\ -d_4 s\alpha_3 \\ d_4 c\alpha_3 \\ 1 \end{bmatrix} \tag{56} $$

Assim, pode-se montar equações para se encontrar θ_1, θ_2 e θ_3. Uma das formas de equacionamento completo encontra-se em Craig (2012), p. 107. Após, é necessário encontrar θ_4, θ_5 e θ_6 (correspondentes a {4}, {5}, {6} da Figura 26, respectivamente). Esses ângulos afetam apenas a orientação do último elo e podem ser encontrados a partir da porção rotacional ${}^0R_6$. Como já se tem os ângulos das primeiras juntas pode-se resolver ${}^0R_4$, de modo que a orientação de ${}^0R_6$ difere dessa orientação devido às últimas juntas. Também, diz-se que como o sistema de referência agora é {4}, então $\theta_4 = 0$, (57):

$$ {}^{4}R_{6}|_{\theta_4=0} = {}^{0}R^{-1}{}_{4}|_{\theta_4=0} \; {}^{0}R_{6} \tag{57} $$

Sabendo-se que (58):

$$ {}^{4}R_{6}(\theta_4, \theta_5, \theta_6) = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{58} $$

Craig (2012) informa que para calcular (58) pode-se utilizar a notação de Z-Y-Z de Euler (encontrada em CRAIG, 2012, p. 43) para encontrar θ_4, θ_5 e θ_6, sendo ${}^{4}R_{6}$ conforme a notação de (43).

Levando em conta, agora, não mais um manipulador industrial e sim uma perna robótica, considerou-se majoritariamente o trabalho sobre o robô KHR-4 (HUBO 2), descrito em Ali, Park e Lee (2010) como embasamento.

Fazendo-se apenas alguns comentários a essa série de robôs da KAIST (*Korea Advanced Institute of Science and Technology* – Instituto Coreano Avançado de Ciência e Tecnologia*)*, em relação à Cinemática Inversa, pode-se dizer que nenhuma das referências relacionadas ao KHR-2, dos quais foram analisados previamente para este trabalho, foi apresentada, especificamente, como foi efetuada a Cinemática Inversa do robô. Foi informado apenas sobre o uso de um método de forma fechada e que não foram utilizadas Jacobianas na Cinemática Inversa:

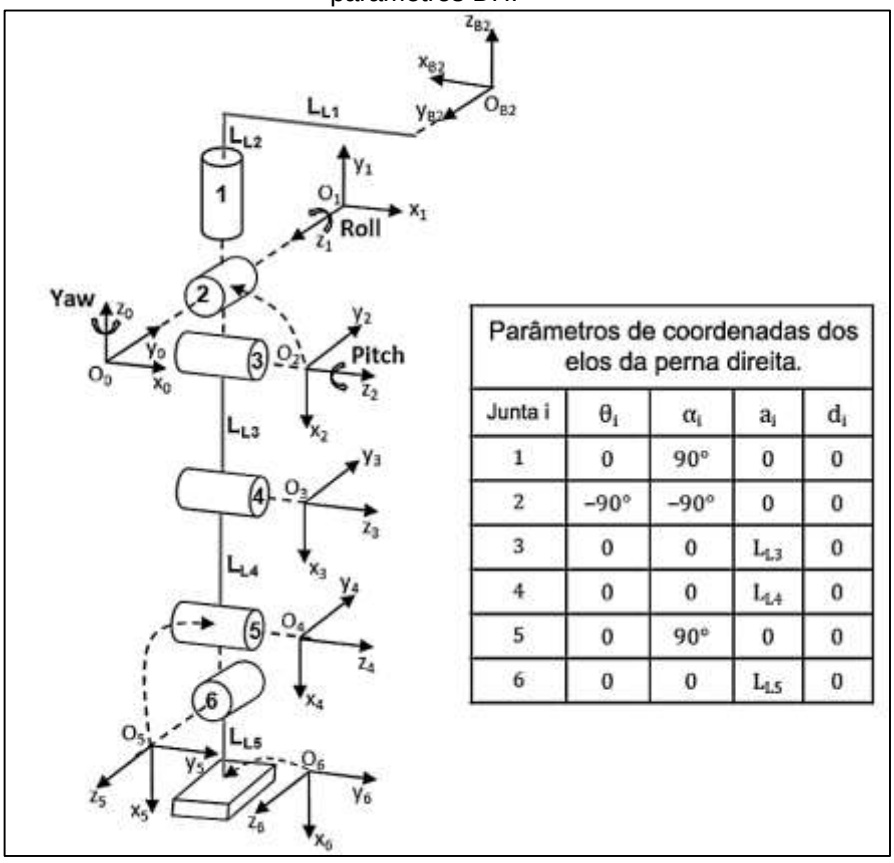
> As juntas do robô foram desenvolvidas para ter uma cinemática simples. Intersectando os eixos das juntas como no quadril (3-eixos), tornozelo (2-eixos), ombro (3-eixos), pulso (2-eixos), uma simples forma-fechada de solução para a Cinemática Inversa foi criada. Nessa solução de forma-fechada funções trigonométricas como seno e cosseno foram envolvidas, mas nenhuma cinemática jacobiana foi inclusa. Assim, a geração do caminho e o desenvolvimento do controle tornaram-se simples. (PARK, KIM, *et al.*, 2004, p. 18, tradução nossa).

Em Kim, et al. (2005) e Kim, Park, & Oh (2006), foi afirmado o mesmo sobre o KHR-2. Portanto, sabe-se que se trata de um método que não é realizado por Jacobianas. Ou seja, implica que o método não é iterativo para encontrar cada ângulo. Já no KHR-4 (HUBO 2) a disposição da perna e das juntas é a mesma que no KHR-2, portanto, os mesmos tipos de equacionamentos poderiam ser efetuados neste modelo cujo método fora divulgado. Inclusive, Ali, Park e Lee (2010) apresentam

outros três robôs (Figura 2) com estruturas cinemáticas similares. A configuração cinemática dos membros inferiores do HOAP-2 e do ASIMO são iguais à do KHR-4, então o mesmo método pode ser aplicado (ALI, PARK e LEE, 2010). O AIST HRP-2 seria o único com alguma diferença, pois possui um deslocamento adicional em uma das juntas do quadril, mas isso alteraria apenas uma das matrizes de transformação (ALI, PARK e LEE, 2010).

Assim, de acordo com a Figura 27, tem-se a estrutura e configurações cinemáticas da perna do KHR-4, as mesmas que serão utilizadas no robô a ser desenvolvido neste trabalho.

Figura 27: Coordenadas das juntas e elos da perna direita do KHR-4 e sua notação em parâmetros DH.



Fonte: Adaptado de Ali, Park e Lee (2010).

Como pode ser verificado na Figura 27, a perna do KHR-4 não possui as três últimas juntas (4, 5 e 6) sobrepostas da maneira descrita anteriormente como no caso de um manipulador industrial, mas possui as três primeiras juntas (1, 2 e 3) de maneira sobreposta. Dessa forma não é possível calcular pelo método de Pieper diretamente, a não ser que a Cinemática Inversa seja calculada pela ordem reversa. Ou seja, com

a posição e orientação das coordenadas da base referenciadas nas coordenadas do pé ao invés da pélvis (ALI, PARK e LEE, 2010).

O método descrito até então, não será utilizado como previa-se incialmente devido à escassez de literatura para o caso de robôs bípedes. Assim, optou-se por utilizar um método iterativo baseado em Jacobianas.

2.4.6 Cinemática Inversa Baseada em Jacobianas

Conforme mencionado no item 2.4.5.2, optou-se por utilizar um método de Cinemática Inversa Baseado em Jacobianas. Os métodos realizados através do equacionamento das Jacobianas aplicam-se para qualquer caso de até pelo menos 6 6DOF, ou seja – conforme já mencionado em 2.4.5.1 – são genéricos. Da mesma forma, apesar de ainda haverem dificuldades, são métodos que lidam melhor com singularidades, em relação aos métodos de forma-fechada. Portanto, a partir de agora será visto o método que foi utilizado para realizar a Cinemática Inversa do robô bípede e seu contexto teórico.

2.4.6.1 Jacobianas

Para o caso onde tem-se funções de variáveis independentes, conforme apresenta Craig, 2012 (59):

$$\begin{aligned} y_1 &= f_1(x_1, x_2, x_3, x_4, x_5, x_6), \\ &\vdots \\ y_6 &= f_6(x_1, x_2, x_3, x_4, x_5, x_6). \end{aligned} \tag{59}$$

De acordo com Craig, 2012, pode-se considerar seis funções com seis variáveis independentes para o caso de um robô com seis juntas. Utilizando a regra da cadeia[3] para descrever os diferenciais de y_i como função dos diferenciais de x_i e tem-se (60):

$$\begin{aligned} \delta y_1 &= \frac{\partial f_1}{\partial x_1}\partial x_1 + \frac{\partial f_1}{\partial x_2}\partial x_2 \dots \frac{\partial f_1}{\partial x_6}\partial x_6, \\ &\vdots \\ \delta y_6 &= \frac{\partial f_6}{\partial x_1}\partial x_1 + \frac{\partial f_6}{\partial x_2}\partial x_2 \dots \frac{\partial f_6}{\partial x_6}\partial x_6, \end{aligned} \tag{60}$$

[3] Regra da Cadeia é utilizada para resolver derivadas de funções compostas.

A partir da notação vetorial tomada de (59) pode ser escrita, também em notação vetorial, a equação de (61):

$$Y = F(X) \rightarrow \delta Y = \frac{\partial F}{\partial X} \partial X \tag{61}$$

A matriz 6×6, $\frac{\partial F}{\partial X}$, de (61), é o que é chamado de Jacobiano, J. É uma forma multidimensional de derivada (CRAIG, 2012). Se as funções $f_1(X)$ a $f_6(X)$ não forem lineares é possível utilizar a notação em (62), pois serão funções de x_i:

$$\delta Y = J(X)\, \partial X \tag{62}$$

Conforme Craig, 2012, se ambos os lados de (62) forem divididos pelo elemento de tempo diferencial, pode-se considerar o Jacobiano como um mapeamento das velocidades em X para as de Y (63). Quando X tem um certo valor, J(X) é uma transformação linear e a cada instante X muda e também assim J(X).

$$\dot{Y} = J(X)\, \dot{X} \tag{63}$$

Assim, de acordo com Craig, 2012, na robótica, utiliza-se Jacobianos para relacionar velocidades de juntas à velocidades cartesianas no *end-effector* (64)*:*

$$v = J(\theta)\dot{\theta} \tag{64}$$

Onde θ é o vetor dos ângulos das juntas do manipulador e v um vetor de velocidades cartesianas para uma relação instantânea. A cada instante o Jacobiano se altera (CRAIG, 2012). Assim, conforme Craig, 2012, para uma perna de 6 DOF, por exemplo, o J é 6×6; θ é 6×1; v é 6×1. Conforme Engardt, Heimburger e Sydhoff (2012), as Jacobianas descrevem o mapeamento linear do espaço de velocidade da junta para o espaço de velocidade do *end-effector*, assim v é composto por 3×1 de velocidade linear v_e e 3×1 de velocidade angular ω_e (CRAIG, 2012). A equação é descrita em (65):

$$\begin{pmatrix} v_e \\ \omega_e \end{pmatrix} = J(\theta)\dot{\theta} \tag{65}$$

De modo que em (66), tem-se:

$$\begin{cases} v_e = \dot{p}_e \\ \omega_e = \dot{\emptyset}_e \end{cases} \tag{66}$$

Onde p_e é o vetor da posição do *end-effector*, e $\emptyset_e$ é o seu vetor de orientação. $J(\theta)$ pode ser escrito como (67):

$$J = \begin{pmatrix} J_P \\ J_O \end{pmatrix} = \begin{cases} v_e = J_P \dot{\theta} \\ \omega_e = J_O \dot{\theta} \end{cases} \tag{67}$$

Simplificando, J_P pode ser chamado somente de J uma vez que $\emptyset_e$ fica fixada à articulação robótica e, portanto, para esse caso ω_e não é necessária, conforme explica Engardt, Heimburger e Sydhoff (2012). O número de linhas é de três (para os três eixos X, Y e Z) e o número de colunas é o número de juntas. Para facilitar o cálculo também pode-se utilizar conforme Gautan e Patil (2015):

$$J = [z_{i-1} \times (p_n - p_{i-1})] \tag{68}$$

Onde pode-se calcular a Jacobiana através do produto vetorial do componente z de $i - 1$ termos, pelo vetor que representa a posição do *end-effector,* p_n, subtraído pelo vetor da posição de cada junta p_{i-1}. Esses componentes podem ser obtidos através da matriz de transformação visto em (54), e estão em função de θ. Para conhecimento, J_o seria o descrito pelo próprio z_{i-1}, somente (GAUTAN e PATIL, 2015), porém não é utilizado nesse caso, como já explicado.

Conforme Buss (2009)[4], a Jacobiana leva a um método iterativo para resolver (64). Supondo que tem-se os valores dos pontos da trajetória de v e θ, a Jacobiana pode ser computada. Ela irá atualizar $\Delta\theta$ incrementando os ângulos das juntas, θ, a cada iteração (69):

$$\theta := \theta + \Delta\theta \tag{69}$$

[4] Buss (2009) faz uma nota explicando que seu trabalho não gerou publicação, pois inicialmente era apenas uma introdução de Buss e Kim (2004). Portanto, segue: <https://www.math.ucsd.edu/~sbuss/ResearchWeb/ikmethods/iksurvey.pdf>. Acesso em: 26 de Out. 2017.

Entretanto, a alteração nas posições do *end-effector*, causada pela alteração nos ângulos das juntas, pode ser estimada conforme (70). Onde é esperado que $\Delta\theta$ leve $\Delta\vec{s}$, aproximadamente, igual a $\vec{e}$, que é o erro entre o ponto desejado e o ponto do *end-effector* naquele instante $(\vec{s})$, informa Buss (2009). Também, pode-se interpretar que os ângulos obtém-se valores de $\Delta\theta$ para que $\Delta\vec{s}$ tenha parcialmente o mesmo comportamento das velocidades dos pontos desejados (BUSS, 2009).

$$\Delta\vec{s} \approx J\Delta\theta \rightarrow \vec{e} = J\Delta\theta \tag{70}$$

Portanto, agora é necessário escolher as estratégias para iterar $\Delta\theta$ para determinar ângulos das juntas (BUSS, 2009). Como deseja-se encontrar θ, a partir de (64), tem-se:

$$\Delta\theta = J^{-1}\vec{e} \tag{71}$$

Em muitos casos (71) pode não ser capaz de encontrar uma resposta, explica Buss (2009). De fato, J pode não ser quadrático ou inversível, e mesmo se for inversível, deixar (71) responsável por encontrar $\Delta\theta$, pode não funcionar apropriadamente, principalmente se J estiver perto de uma singularidade.

2.4.6.2 Singularidades

Como um adendo ao item 2.4.6.1, será dado uma descrição de um aspecto da robótica: as singularidades. Conforme Craig (2012) se for possível inverter a matriz J – (48) – para calcular as velocidades das juntas a partir das velocidades cartesianas, quer dizer que a matriz não é singular. Caso contrário, se não for possível, quer dizer que a matriz é singular.

Caso se deseje, por exemplo, calcular a posição do tornozelo para dados pontos passados para a perna robótica, é possível que, usando (70), nem todos os valores de $\dot{\theta}$ sejam encontrados, pois J não é inversível para todos os valores de θ (CRAIG, 2012). Esses pontos são chamados de singularidades. Craig, 2012 define esses eventos em Singularidades do Limite do Espaço de Trabalho e Singularidades do Interior do Espaço de Trabalho. O primeiro tipo, ocorre quando o manipulador está

totalmente estendido ou recuado estando, assim, no limite do espaço de trabalho. O segundo tipo, ocorre dentro dos limites do espaço de trabalho e geralmente são causados pelo alinhamento de dois ou mais eixos de juntas (CRAIG, 2012). Ou seja, são pontos do Espaço de Trabalho da perna (ou de qualquer manipulador robótico) cujos pontos do *end-effector* não são possíveis de serem encontrados. Conforme Buss (2009), na prática, singularidades verdadeiras raramente são alcançadas ou encontradas, ao invés disso a singularidade é testada verificando se os valores estão próximos a zero.

De acordo com Nakamura e Hanafusa (1986), do ponto de vista do mecanismo, o manipulador não deveria ser solicitado a mover-se em direção a pontos singulares. Entretanto, problemas com singularidades estão relacionadas na movimentação do robô e é um problema de baixo nível, na hierarquia de um sistema de controle. Já o planejamento das trajetória do *end-effector* é um problema de alto nível. Portanto, é importante estabelecer um esquema de controle que possa passar por pontos singulares sem causar erros fatais, assim como planejar trajetória capazes de desviar desses pontos, se for possível (NAKAMURA e HANAFUSA, 1986).

2.4.6.3 Método da Jacobiana Pseudo-Inversa

Como Jacobianas inversas, (70), não são capazes de resolver pontos com singulares, o método da Jacobiana Pseudo-Inversa foi considerado como sendo uma solução para esse problema (NAKAMURA e HANAFUSA, 1986). Jacobianas Pseudo-Inversas, também chamadas de Inversa de Moore-Penrose (BUSS, 2009), podem ser definidas até para matrizes singulares (72):

$$\dot{\theta} = J^{\#}(\theta)v \tag{72}$$

Conforme Nakamura e Hanafusa (1986), a equação descrita fornece uma solução por mínimos quadrados, por uma normalização Euclidiana para (70), ou seja, $min\left\|\dot{\theta}\right\|$, assim (73):

$$min\left\|v - J(\theta)\dot{\theta}\right\| \tag{73}$$

Sendo a normalização Euclidiana (74):

$$\|x\| = \sqrt{\sum_{k=1}^{N} |x_v|^2} \tag{74}$$

De modo que $J^{\#}$ proporciona uma solução aproximada através de (73), e isso pode ser uma melhora satisfatória para solucionar equações algébricas lineares, explica Nakamura e Hanafusa (1986). Deve ser dada atenção ao fato de que a solução que anteriormente era exata, agora é apenas uma aproximação. Também, que os pontos próximos à singularidade forçam v a encontrar uma solução exata, e isso pode acarretar em uma solução impraticável (NAKAMURA e HANAFUSA, 1986). A exatidão da solução é sempre considerada o quão mais significativa ao invés do quão mais viável ela é. Então, a inviabilidade e a descontinuidade de algumas soluções de $J^{\#}(\theta)$ causariam repostas indesejadas.

2.4.6.4 Método de Quadrados Mínimos Amortecidos - *Damped Least Squares* (DLS)

O Método de Quadrados Mínimos Amortecidos (*Damped Least Squares* – DLS), conforme informa Buss (2009), consegue evitar muitos dos problemas da Jacobiana Pseudo-Inversa, relacionados a singularidades. Também chamado de *Levenberg-Marquardt*, um de seus primeiros usos para o cálculo de uma Cinemática Inversa, foi em Nakamura e Hanafusa (1986), comenta Buss (2009).

O DLS é capaz de encontrar o mínimo vetor para $\Delta\theta$ que fornece a melhor solução para (71) e sendo assim, conforme Buss (2009), tem-se (75):

$$\|J\Delta\theta - \vec{e}\|^2 + \lambda^2\|\Delta\theta\|^2 \tag{75}$$

Onde $\lambda \in \mathbb{R}^*$ e é uma constante de amortecimento (BUSS, 2009), pode ser escrita na forma, (76), (77) e (78):

$$(J^T J + \lambda^2 I)\Delta\theta = J^T\vec{e} \tag{76}$$

$$\Delta\theta = (J^T J + \lambda^2 I)^{-1} J^T \vec{e} \tag{77}$$

$$\Delta\theta = J^T (JJ^T + \lambda^2 I)^{-1} J^T \vec{e} \tag{78}$$

Assim em (76) tem-se que $J^T J + \lambda^2 I$ é não-singular, assim o DLS é igual a (77). Porém, $J^T J$ é uma matriz $n \times n$, onde n é o número de DOF. Assim, pode-se escrever a mesma equação como em (78), explica Buss, 2009. A vantagem de se escrever da segunda maneira é que a matriz sendo inversa torna-se uma matriz $m \times m$ onde $m = 3k$, é a dimensão do espaço dos pontos das trajetória, e geralmente é muito inferior a n (BUSS, 2009). No caso da perna robótica deste trabalho, para o primeiro caso tem-se $I = 6 \times 6$ e no segundo caso $I = 3 \times 3$. Adiante $\vec{e}$ será chamado de ΔX.

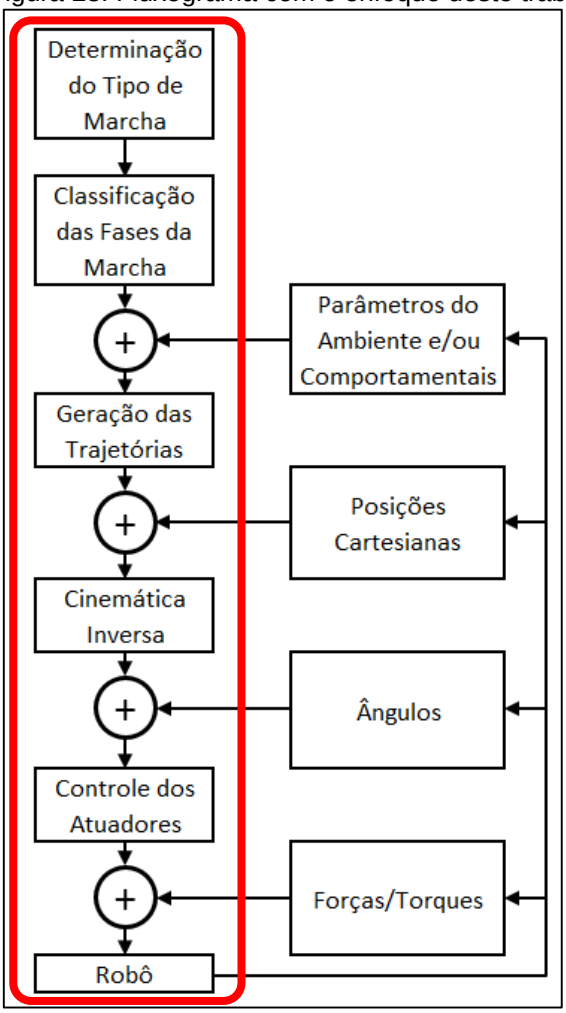
A constante de amortecimento λ depende de parâmetros e detalhes do manipulador e dos pontos da trajetória, e deve ser escolhido cuidadosamente. O fator deve ser elevado para que $\Delta\theta$ lide bem com singularidades, porém, conforme informa Huang e Yan (2007) e Nakamura e Hanafusa (1986), a constante de amortecimento também afeta a resposta em $\dot{\theta}$ e, consequentemente, o erro em posição.

3 METODOLOGIA DO TRABALHO

Este trabalho consiste na implementação do método de geração de trajetória e Cinemática Inversa, o qual foi baseado no que foi descrito em 2.3 e 2.4, respectivamente, em uma estrutura de robô bípede com 6-DOF por perna. O objetivo é que ao final desenvolva-se um algoritmo em MATLAB capaz de gerar pontos cartesianos (X, Y, Z) correspondentes a uma marcha, e que esses pontos possam ser alcançados pela a pélvis (COM) e pés de um robô bípede, conforme o destacado no fluxograma da Figura 28.

Por fim, ao aplicar-se o algoritmo completo, os ângulos obtidos para as juntas são passados para um modelo real de robô simples com a estrutura destacada, através de um Arduino Mega 2560, para fins de demonstração de conceito, onde espera-se que execute uma caminhada frontal simples, sem nenhum tipo de realimentação.

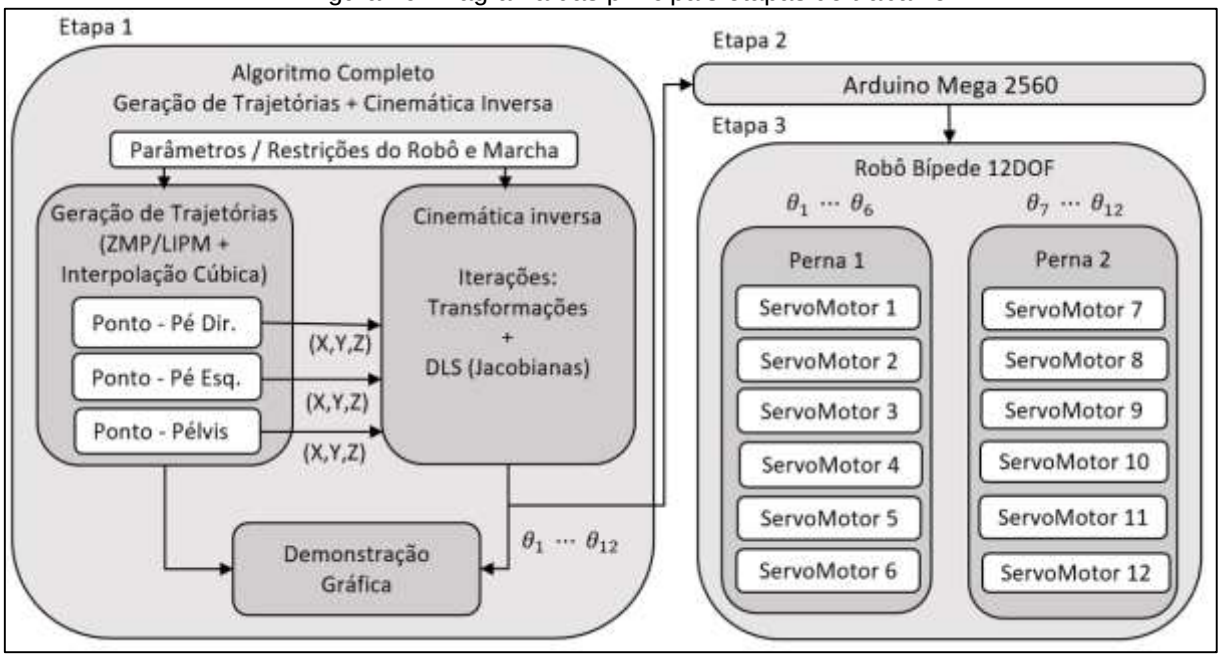
Figura 28: Fluxograma com o enfoque deste trabalho.



Fonte: Adaptado de Kim, Park, e Oh (2006).

Conforme a Figura 29, são apresentadas as três principais etapas que compõem o projeto.

Figura 29: Diagrama das principais etapas do trabalho.



Fonte: O próprio autor (2017).

A Etapa 1 foi realizada no MATLAB (R2015a), onde os pontos e os parâmetros/restrições do robô e da marcha são informados para a execução do algoritmo da geração de trajetória e da Cinemática Inversa. Os parâmetros utilizados, primeiramente, foram extraídos e/ou inspirados pela literatura que em sua grande maioria buscam alcançar uma caminhada com proximidade à naturalidade humana. Baseado nisso, para esse trabalho, foi executado uma marcha para frente de três passos. Em um segundo momento, parâmetros provindos de testes empíricos foram utilizados.

A partir dos pontos gerados foram realizadas iterações que integram o processo de transformações e do DSL com base nas Jacobianas. Com isso foram gerados os ângulos de cada junta para cada atuador. Também, foi realizado uma demonstração gráfica do robô.

A Etapa 2 refere-se à plataforma Arduino MEGA 2560, responsável por receber e transmitir os ângulos gerados em MATLAB para cada respectivo atuador.

A Etapa 3 foi realizada na planta idealizada: o robô desenvolvido. Devido aos altos custos de se adquirir uma estrutura robótica pronta, construiu-se de uma. O robô é composto pelas duas pernas com 6 DOF cada, unidas pela pélvis. E cada atuador

(servomotor) equivale a uma junta. Por fim, o robô foi projetado para executar uma caminhada frontal simples baseados nas posições/ângulos gerados. O principal critério que foi levando em conta para o seu desenvolvimento, foi a disposição correta das juntas. Como pode-se perceber, a estrutura mecânica foi construída apenas com o intuito de demonstrar os conceitos trabalhados nesse trabalho. Portanto, o robô não terá nenhum tipo de realimentação e os ângulos serão aplicados de forma direta.

4 DESENVOLVILMENTO DA IMPLEMENTAÇÃO DOS ALGORITMOS RESPONSÁVEIS PELA MARCHA DO ROBÔ BÍPEDE DE 12 DOF

Neste capítulo serão apresentadas as etapas expostas na metodologia, da forma como foram concebidas utilizando o referencial teórico abordado. Conforme apresentado na Figura 28, o trabalho não consiste em aplicar nenhum tipo de sensor para monitorar e compensar qualquer parâmetro do robô ou da marcha. Sendo assim, já de início concebeu-se o formato de um algoritmo e aplicação incluindo somente o que consta dentro do destacado em vermelho Figura 28.

4.1 DESENVOLVIMENTO DO ALGORITMO REALIZADO EM MATLAB

Partindo-se do que foi apresentado no Item 2 deste trabalho, considerou-se um robô de 12 DOF. Conforme os blocos apresentados na Figura 28, a marcha escolhida a ser executada pelo robô foi uma simples marcha frontal considerando um robô já em pé, terminando a marcha também em pé. Ou seja, partiu-se do caso de locomoção mais simples encontrado na literatura. Concebeu-se também a marcha sendo *off-line*. Isso quer dizer que os pontos da trajetória e os respectivos ângulos das juntas, serão gerados e calculados previamente conforme desejado e, posteriormente, serão aplicados ao robô. Durante a marcha também não haverá qualquer intervenção, para reposicionar pontos, alterar ângulos ou parâmetros se o robô vier a sair da rota. Dessa forma, para executar cada passo, sabe-se que haverá momentos em que o robô estará apoiado em apenas um dos pés (SSP) e momentos em que estará apoiado em ambos os pés (DSP). Existem outros níveis de subdivisões de uma marcha, mas que para este trabalho não foram necessárias, nem utilizadas. Comumente, são necessárias durante uma marcha *online*, onde as mais diversas etapas podem estar recebendo informações corretivas durante uma marcha de um robô e, portanto, se faz necessário maiores classificações.

4.1.1 Desenvolvimento do Algoritmo de Geração de Trajetória para o COM

Para o método de geração de trajetória foi utilizado conforme descrito no item 2.3. Assim, os pontos foram dados a partir do método de ZMP através de uma

simplificação do robô por LIPM, para verificação da trajetória da pélvis (que corresponde ao COM) e de uma referência para os pés, que foi encontrada a partir do equacionamento de interpolações cúbicas.

A trajetória para o COM, a partir de (31) e (32), como pode ser retomado dessas duas equações, correspondem aos pontos em X e Y do percurso da pélvis considerando que em Z, para simplificação, será uma altura constante. Para a referência dos pés, segue-se o mesmo caso utilizando as equações (33) e (34). É necessário salientar que aqui é dado somente uma referência e não os pontos da trajetória em si, e que para esse trabalho o ZMP não terá a mesma visibilidade como se fosse o caso de gerar trajetórias complexas como uma marcha em curva, por exemplo. Mas o ZMP deve coincidir com os passos dados pelas interpolações.

Para que sejam gerados pontos são necessárias duas iterações: uma delas em função do tempo, e outra, interna a essa referente ao somatório que consta em cada uma das equações. A forma como o fator do tempo é obtido será mostrado adiante. Sobre o número de iterações do somatório, como já havia sido mencionado, estabeleceu-se que $N = 15$.

Portanto, como parâmetros de entrada para algoritmo, é necessário informar o comprimento do pêndulo invertido l $[mm]$; número de passos n; largura do passo B $[mm]$; razão da DSP, DSP_ratio $[\%]$. São utilizados como parâmetros também a força da gravidade g $[mm/s^2]$; o período de meia marcha (1 passo) T $[s]$; frequência natural do pêndulo invertido ω_n $[rad/s]$; a distância aproximada entre o centro do pé até uma extremidade b $[mm]$; e o coeficiente do Fator de Lanczos d.

Assim, tem-se a atribuição dos parâmetros, seguido pela geração do fator tempo, e por seguinte dá-se as iterações necessárias. Para cada instante de t estabelecido são realizadas os somatórios para se encontrar um ponto de referência de ZMP dos pés e outro do CoM (pélvis). Ainda é necessário considerar a atipicidade do primeiro e do último passo. É imposta uma condicional para detectar os pontos deslocados em X que surgem de posições negativas na origem e posições superiores ao deslocamento final do último passo. Isso ocorre, pois no caso ideal, conforme explicado anteriormente, foi considerado uma marcha já em movimento. Portanto, a forma encontrada para sanar esse problema, foi fazendo com que os pontos em X, ao exceder esses limites fossem ajustados para o próprio valor máximo do limite. Conforme o APENDICE A (Figura 45) tem-se o esquemático do algoritmo.

4.1.2 Desenvolvimento do Algoritmo de Geração de Trajetória para os Tornozelos

Para a geração dos pontos da perna suspensa são realizados as interpolações dos pontos respeitando a referência ZMP dada. Assim, seguindo os intervalos estipulados em 2.3.5, tem-se as iterações para o tempo, para os pontos em X e Z do tornozelo direto e esquerdo. Y é mantido constante para ambos os tornozelos. Aqui, é necessário salientar que durante todo esse trabalho foi tido que a trajetória seria passada para os pés. Isso se deve pelo fato de que a maior parte da literatura utilizada também denomina desta maneira. Entretanto, as trajetórias são passadas mais especificamente para os tornozelos, ou seja, para o ponto onde se cruzam as duas últimas juntas e não diretamente para o *end-effector* (pé). Portanto, o ponto mínimo em Z dado é correspondente à altura do último elo do robô. Essa alteração, apesar de sutil neste momento, acarretará em uma mudança considerável durante a Cinemática Inversa. Os pontos específicos do *end-effector* do último elo serão dados de forma diferente, e serão tratados mais adiante. Por hora, essa alteração não acarreta nenhuma outra consequência na geração dos pontos da trajetória.

Prosseguindo com o algoritmo tem-se, portanto, os parâmetros iniciais passados para essa etapa. É necessário informar o comprimento do pêndulo invertido l $[mm]$; o comprimento do pé até o dedão $l_{af}[mm]$; o comprimento do pé até o calcanhar l_{ab} $[mm]$; o comprimento do pé até a junta do tornozelo l_{an} $[mm]$; posição em X do ponto mais alto do pé suspenso $L_{ao}[mm]$; posição em Z do ponto mais alto do pé suspenso $H_{ao}[mm]$; número de passos n; largura do passo $D_S = B$ $[mm]$; razão da DSP, DSP_ratio $[\%]$. São utilizados como parâmetros também a força da gravidade g $[mm/s^2]$; frequência natural do pêndulo invertido f $[Hz]$; o período de marcha (2 passo) T_c $[s]$; Intervalo da DSP $T_d[s]$; tempo médio onde o pé encontra-se mais alto durante cada passo $T_m[s]$; ângulos de “aterrisagem” e de “saída” do pé ao solo e de inclinação do solo, são considerados no algoritmo, porém constam como zero $q_b = q_f = q_{gs} = q_{ge} = 0$.

Tendo-se os parâmetros, primeiramente, é realizado a iteração para gerar os valores de t conforme consta em (37) e (38). Essa mesma forma foi utilizada para o algoritmo anterior referente ao COM. A iteração é feita com base em k números de passo ou de marcha (2 passxos). Também são realizadas as iterações para a perna

direita e esquerda em X e Z, em função de k. A iteração de cada perna para X e Z é composta de duas partes. Uma parte é responsável por determinar o período em que a perna fica suspensa e outro em que a perna permanece fixa ao chão. Ambos os movimentos são intercalados e inversos para a perna direita e esquerda. Ou seja, obviamente quando a perna direita está suspensa do solo a perna direita permanece fixa, e vice-versa. Para os deslocamentos em X também foi necessário aplicar uma condicional devido aos pontos que antecedem o primeiro passo ou ultrapassam o último passo.

Os pontos da trajetória foram subdivididos para que posteriormente fossem passados pela interpolação cúbica. Para que ficassem todos com a mesma razão foi utilizado como base o tamanho do vetor de t e foi escolhido uma ordem numérica de subdivisão para gerar os pontos intermediários (chamados aqui de subpontos) e após foram interpolados.

Tendo-se agora o conhecimento de como os valores de t são gerados, é necessário informar que para os pontos de COM e da referência de ZMP é utilizada uma subdivisão de pontos de forma semelhante. Como os comportamentos do COM e ZMP são diretamente em função do tempo, é realizada uma subdivisão inicial gerando novos valores baseados em t para que as dimensões dos vetores que guardam esses valores permaneçam corretos e para que, posteriormente, sejam atribuídos corretamente em cada instante de tempo. Também, é necessário uma subdivisão com um número menor de valores devido às iterações de COM e ZMP. Após essas iterações, os valores são passados em uma nova subdivisão de pontos, sempre mantendo as dimensões dos vetores iguais aos pontos gerados para X e Z dos tornozelos. O diagrama do algoritmo dos pontos do tornozelo pode ser visto no APENDICE B (Figura 46).

4.1.3 Desenvolvimento do Algoritmo de Cinemática Inversa

O algoritmo da Cinemática Inversa consiste na iteração para redução de erro a partir do método salientado no item 2.4.6.3, o *Damped Least Square* (DLS), baseado em Jacobianas. Para executá-lo recebe-se os subpontos dos *end-effectors* e do COM em X, Y e Z de cada instante, para as três localizações. Para cada instante esse conjunto de subpontos é iterado a partir da posição anterior até que sejam

encontrados os ângulos necessários para cada junta. Após estar dentro da margem de erro estipulada, recebe-se o próximo conjunto de subpontos da trajetória, referente ao próximo instante.

Portanto, inicialmente, é realizado a transformação a partir dos ângulos iniciais conhecidos dos parâmetros de comprimento de cada elo, e dos parâmetros DH da estrutura do robô. Após passar os parâmetros, a transformação é realizada, montando matricialmente a estrutura do robô desejada, com cada junta em seus respectivos ângulos iniciais. Em (79), são apresentadas as transformações para uma perna partindo-se de COM, conforme (54) e da Figura 27. Apenas alterando um parâmetro pode-se calcular para a outra perna: $L1 = -L1$. As matrizes podem ser encontradas no APENDICE D.

$$\begin{aligned} T_1^0 &= R_{Z,\theta 1} R_{X,\alpha 1} P_{X,a1} P_{Z,d1} \\ T_2^1 &= R_{Z,\theta 2} R_{X,\alpha 2} P_{X,a2} P_{Z,d2} \\ T_3^2 &= R_{Z,\theta 3} R_{X,\alpha 3} P_{X,a3} P_{Z,d3} \\ T_4^3 &= R_{Z,\theta 4} R_{X,\alpha 4} P_{X,a4} P_{Z,d4} \\ T_6^4 &= R_{Z,\theta 5} R_{X,\alpha 5} P_{X,a5} P_{Z,d5} \\ T_6^5 &= R_{Z,\theta 6} R_{X,\alpha 6} P_{X,a6} P_{Z,d6} \end{aligned} \tag{79}$$

Em (80), tem-se a equação final de transformação para uma perna:

$$T_6^0 = T_1^0 T_2^1 T_3^2 T_4^3 T_6^4 T_6^5 \tag{80}$$

O ponto COM é sempre passado como a base da estrutura do robô a cada instante. Ou seja, é como se a estrutura do robô fosse remontada a cada instante a partir do COM e os pés são responsáveis por encontrar suas respectivas posições – tanto a perna fixa ou a perna suspensa. Dessa forma é necessário descobrir-se a posição de cada *end-effector* (extremidade de cada perna) obtendo-se a posição alcançada após receber os ângulos iniciais através da cinemática direta. A cinemática direta pode ser dada pela quarta coluna (posição) da matriz de cada transformação. A última linha é desconsiderada. Ou seja (81):

$$FK_1^0 = T_1^0(1 \dots 3,4) \tag{81}$$

$$FK_2^0 = T_1^0(1 \dots 3,4)T_2^1(1 \dots 3,4)$$
$$FK_3^0 = T_1^0(1 \dots 3,4)T_2^1(1 \dots 3,4)T_3^2(1 \dots 3,4)$$
$$FK_4^0 = T_1^0(1 \dots 3,4)T_2^1(1 \dots 3,4)T_3^2(1 \dots 3,4)T_4^3(1 \dots 3,4)$$
$$FK_5^0 = T_1^0(1 \dots 3,4)T_2^1(1 \dots 3,4)T_3^2(1 \dots 3,4)T_4^3(1 \dots 3,4)T_6^4(1 \dots 3,4)$$
$$FK_6^0 = T_1^0(1 \dots 3,4)T_2^1(1 \dots 3,4)T_3^2(1 \dots 3,4)T_4^3(1 \dots 3,4)T_6^4(1 \dots 3,4)T_6^5(1 \dots 3,4)$$

Onde $T_n^o(1 \dots 3,4) = [\, T_n^o(1,4);\ T_n^o(2,4);\ T_n^o(3,4)]$. O primeiro conjunto de subpontos dados coincidem com os subpontos alcançados propositalmente, porém não é o caso nos demais instantes e iterações. Assim, após encontrar o primeiro conjunto de subpontos para as extremidades, subtraem-se dos subpontos dados para aquele instante. Essa diferença será utilizada no cálculo do DLS, conforme (82):

$$\overrightarrow{\Delta X} = \left(\left[X_{end-effector}\ \ Y_{end-effector}\ \ Z_{end-effector}\right] - FK_6^0\right) \tag{82}$$

Para o DLS é necessário realizar também o cálculo das Jacobianas. As Jacobianas, conforme descritas no item 2.4.6.1, mais precisamente utilizando a equação (68), irão necessitar do ponto inicial (COM) e dos pontos alcançados em cada junta. O subponto alcançado por cada junta pode ser dado da mesma forma por cinemática direta conforme (82). Dessa forma é realizado o produto vetorial entre o eixo Z de cada junta (83) – que convencionalmente é adotado como o eixo da rotação da junta, conforme já explicado – pelo vetor resultante da subtração entre a posição do *end-effector* do manipulador com a posição atual do subponto de cada junta. A orientação do eixo Z de cada junta pode ser dada pela terceira coluna da transformação, com exceção do primeiro eixo que é considerado genérico, logo é o próprio vetor unitário em Z.

$$\begin{aligned} J_1 &= [[0\ 0\ 1] \times (FK_6^0 - CoM)] \\ J_2 &= [T_1^0(1 \dots 3,3) \times (FK_5^0 - FK_1^0)] \\ J_3 &= [T_2^0(1 \dots 3,3) \times (FK_6^0 - FK_2^0)] \\ J_4 &= [T_3^0(1 \dots 3,3) \times (FK_6^0 - FK_3^0)] \\ J_5 &= [T_4^0(1 \dots 3,3) \times (FK_6^0 - FK_4^0)] \\ J_6 &= [T_5^0(1 \dots 3,3) \times (FK_6^0 - FK_5^0)] \end{aligned} \tag{83}$$

Assim, um vetor 3×6 com a Jacobiana referente a cada junta pode ser montado conforme (84). Lembrando que aqui só está sendo tratado para o caso de uma só perna:

$$J = [J_1 \ J_2 \ J_3 \ J_4 \ J_5 \ J_6] \tag{84}$$

Dessa forma, após obter-se a Jacobiana J e a diferença vetorial $\overrightarrow{\Delta X}$ entre o ponto desejado e o ponto calculado com os ângulos atuais, aplica-se (56). O $\overrightarrow{\Delta X}$ corresponde ao erro em posição $\vec{e}$. Inicialmente, para o fator de amortecimento λ é escolhido um valor empiricamente. Deixar um valor constante, como já mencionado, acarreta em uma convergência menor aos ângulos estabelecidos. Portanto, tem-se conforme (85):

$$\Delta\theta = J^T(JJ^T + \lambda^2 I)^{-1} J^T \overrightarrow{\Delta X} \tag{85}$$

Assim, são obtidos os $\Delta\theta$ que serão somados aos ângulos anteriores θ, conforme (69). Porém, antes que isso seja realizado, é feito uma verificação do erro com normalização Euclidiana conforme em (73) e (74). Também é imposta uma condicional de que somente se a diferença angular normalizada $\|\Delta\theta\|$ de ambos os *end-effectors* de cada perna estiver dentro dos limites aceitáveis estipulados (ξ) é que os ângulos serão válidos. Caso $\|\Delta\theta\|$ ainda for maior, repete-se o processo iterando-se novamente e recalcula-se com os novos valores, até que $\|\Delta\theta\|$ seja reduzido abaixo da margem estipulada.

Quando os valores para o $\|\Delta\theta\|$ forem aceitáveis, passa-se o próximo conjunto de subpontos para os *end-effectors* e COM, para encontrar os ângulos das juntas que irão satisfazer a nova posição. Repete-se esse processo de iterações até que sejam dados todos os pontos da trajetória. O fluxograma completo simplificado do algoritmo pode ser dado no APENDICE C (Figura 47).

4.2 DESENVOLVIMENTO DO ALGORITMO REALIZADO NO ARDUINO MEGA 2560

Para passar os ângulos para os atuadores foi utilizado a plataforma Arduino Mega 2560. Ele possui um microcontrolador baseado no ATmega2560 e possui 54 pinos IO

digital dos quais 14 são dedicados para PWM (Pulse Width Modulation – Modulação por Largura de Pulso), além de possuir comunicação serial via cabo USB. A inspiração para o uso desse microcontrolador, além de ser uma opção barata, proveio do robô ROFI, que será abordado no item 4.3. O Arduino Mega 2560 possui o número necessário de saídas PWM para enviar os ângulos gerados em MATLAB aos servomotores.

Da mesma forma, suas bibliotecas já contam com um acervo de funções que permitem declarar diretamente o ângulo desejado, informando poucos parâmetros. Sendo assim, apenas cria-se um objeto do tipo *Servo*, relaciona-se esse objeto com o pino desejado através da função *attach()* (pino correspondente à PWM). Após, com a função *map()* e *write()* e, realiza-se a conversão do ângulo desejado aos parâmetros do modelo do servo escolhido e escreve-se o ângulo desejado, respectivamente. A função *map()* vai dentro de *write()*. Os parâmetros informados para *map()* são: ângulo gerado, ângulo mínimo, ângulo máximo, parâmetro mínimo e parâmetro máximo. Ou seja, como o servomotor escolhido foi o MG996R (conforme item 4.3), sabe-se que os valores vão de 0º a 180º, e que seus parâmetros mínimos e máximos são 700 e 2300. Com isso basta informar os ângulos provindos do MATLAB.

Inicialmente, foi concebido utilizar-se a porta serial para realizar a comunicação entre MATLAB e Arduino, entretanto, visto que o número de ângulos gerados é alto e que informar todos via serial poderia ser um processo moroso e ainda correr o risco de perder informação. Dessa forma, como o objetivo do trabalho consiste na demonstração dos métodos utilizados, decidiu-se gerar a sintaxe para todos os ângulos externamente utilizando uma planilha eletrônica. Com o uso de fórmulas, todas as sintaxes permanecem em células para o número ângulos que for necessário, bastando copiar do MATLAB para a planilha eletrônica, e da planilha eletrônica para o ambiente de programação do Arduino.

É necessário mencionar que a função *map()*, aceita apenas números inteiros. Assim, os números já foram arredondados e também foram realizadas compensações para ajustar os ângulos iniciais no servomotor. Também, foi aplicado um atraso (*delay*) para ajustar o tempo com que os subpontos eram enviados, visto que na prática o tempo de marcha deve ser o mesmo ao estabelecido na etapa de geração de trajetória.

4.3 DESENVOLVIMENTO DA CONSTRUÇÃO DO ROBÔ BÍPEDE DE 12DOF

Para o desenvolvimento do robô, primeiramente, foi considerado a compra de um modelo comercial. Entretanto, devido à morfologia da estrutura proposta para esse

trabalho, não foi possível encontrar um modelo com um preço acessível à compra. Modelos com 6-DOF por perna ou menos, ainda assim possuíam preços elevados.

Na Tabela 2, é possível verificar o valor de compra de alguns modelos de robôs bípedes conhecidos (ainda que estejam inclusas todas as suas outras funcionalidades).

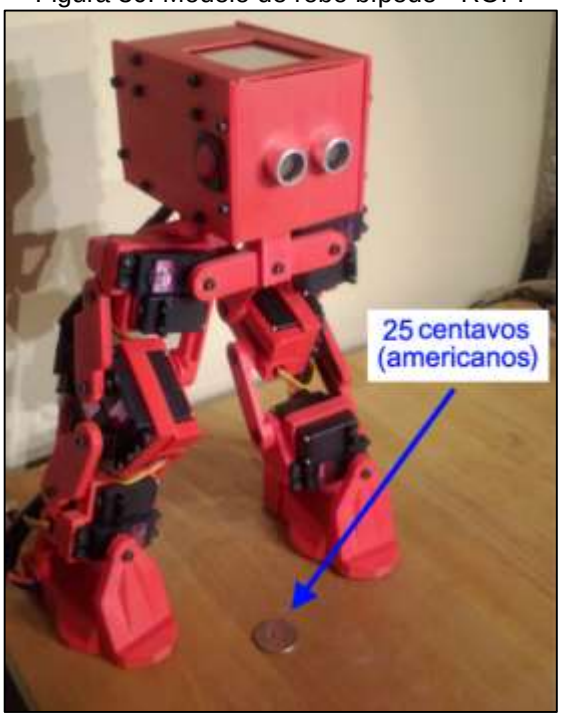
Tabela 2: Preços de robôs bípedes comerciais

Modelo	DOF por perna	Preço
DARwIn Mini (ROBOTIS Mini)	5	U$ 499,00
HOVIS Eco Lite	6	U$ 700 - 1100
HOVIS Eco Plus	6	U$ 1000 - 1200
NAO Evolution	6	U$ 7.500,00
DARwIn-OP 2 (ROBOTIS OP 2)	5	U$ 9.600,00
HRP-4	6	U$ 300.000,00
HUBO 2 Plus (KHR-4)	6	U$ 400.000,00
ASIMO	6	U$ 2.500.000,00

Fonte: Adaptado de Smashing Robotics, 2017.

Dessa forma, foi considerada a utilização do robô ROFI (Figura 30). O site oficial[5] apresenta todas as suas etapas de construção e materiais utilizados.

Figura 30: Modelo de robô bípede - ROFI



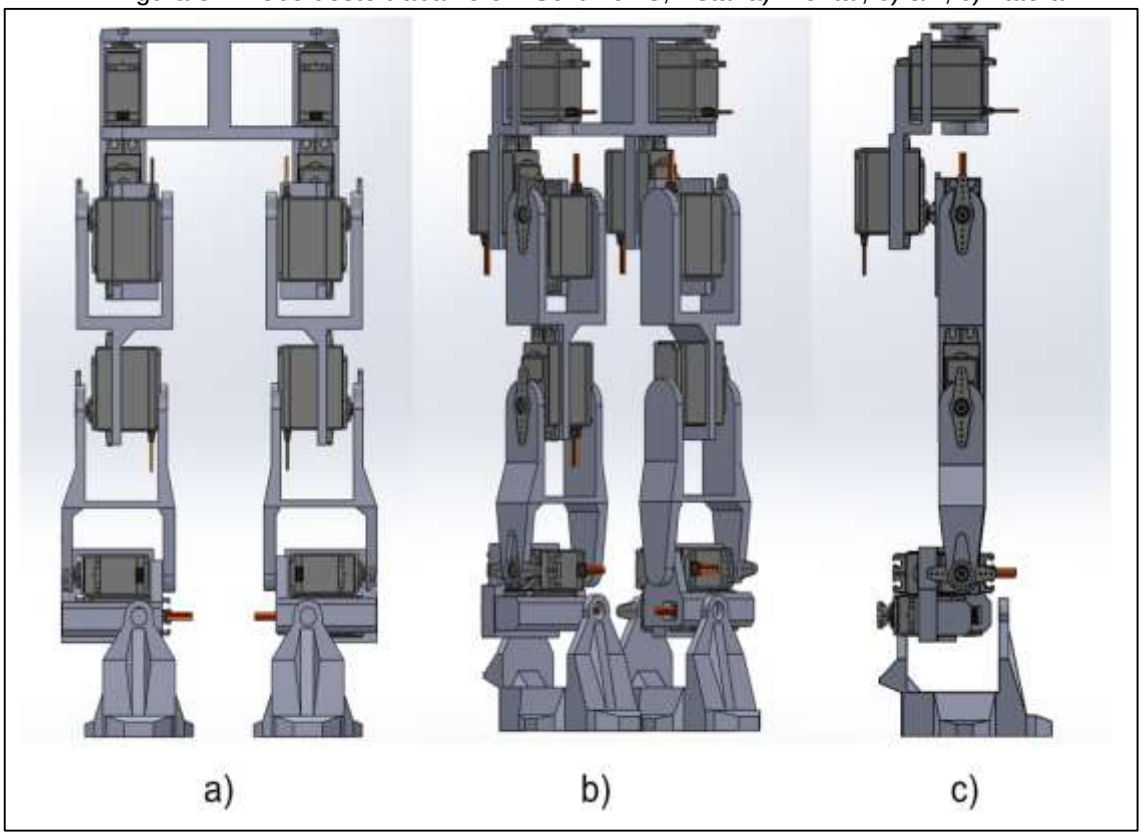
Fonte: Adaptado de Project Biped, 2017.

[5] Site Oficial do ROFI: <http://www.projectbiped.com/prototypes/rofi>. Acesso em: 13 de Jun. de 2017.

Entretanto, pelo fato desse robô também não corresponder com uma estrutura e organizações de juntas semelhantes às propostas até então, essa estrutura de robô não foi utilizada. Porém, o projeto do robô ROFI serviu de inspiração para esse trabalho e foi responsável pela decisão do tipo do servomotor utilizado, do Arduino Mega 2560, e da forma de construção de algumas peças utilizadas. Outras peças tiveram de ser projetadas no *software Solidworks*, e foram usinadas em plástico de engenharia – polímero UHMW (*Ultra High Molecular Weight*) ou impressas em uma impressora 3D – material ABS (Acrilonitrila Butadieno Estireno). Esses materiais são baratos e estavam disponíveis para a realização do trabalho.

Assim, foi desenvolvido uma estrutura de robô bípede que atendia aos critérios morfológicos apresentados. O desenho técnico consta no APÊNDICE N (Figura 60), e o desenho em 3D realizado no *Solidworks,* na Figura 31.

Figura 31: Robô deste trabalho em *Solidworks*, vista: a) Frontal; b) 3/4; c) Lateral.



Fonte: O próprio autor (2017).

Para as juntas do robô deste trabalho será levado em conta o alcance máximo do limite do servomotor aplicado à estrutura. Portanto, o alcance do movimento angular das juntas do robô ficam conforme a Tabela 3.

Tabela 3: Alcance do movimento angular de cada junta do robô deste trabalho.

	Junta	Alcance de Movimento Angular
Quadril	*Roll*	-118,62º a +52,30º
	Pitch	-119,82º a +60,18º
	Yaw	-21,79º a +60º
Joelho	*Pitch*	0º a +138,88º
Tornozelo	*Roll*	-97,38º a +51,45º
	Pitch	-90º a +51,45º

Fonte: O próprio autor (2017).

Como o enfoque desse trabalho não é a locomoção (marcha), e sim a Cinemática Inversa da parte inferior do corpo (bipedismo), não foi realizado um estudo e nem implementado a parte superior do corpo.

Por fim, pode-se afirmar que o custo total da estrutura do robô recaiu principalmente sobre os servomotores: onde o custo de cada um foi R$ 37,99, totalizando R$ 455,88. O polímero UHMW e o processo de usinagem das peças, assim como o material ABS e a impressão 3D foram adquiridos de forma gratuita. O custo dos parafusos (M3), os mesmos utilizados no ROFI, e outras peças, ficou abaixo de R$ 20,00. Algumas dessas peças vieram com o próprio *kit* dos servomotores e outras foram adquiridas separadamente.

O servomotor utilizado é o MG996R da *Tower Pro*. Conforme o fabricante[6], ele pode suportar torques em torno de 10 kg. Sua rotação é de 180º e sua tensão de operação é de 4,8 V à 6,6 V. Sua velocidade fica entre 0,19 [s] /60º (4,8 V) a 0,15 [s] / 60º (6,0 V).

Seu funcionamento se dá através de um sinal de PWM em uma tensão na faixa de operação. Seu período é de 20 ms (50 Hz). O *duty cycle* passado para o MG996R corresponderá a um determinado ângulo. Dessa forma, não se faz necessário o uso de qualquer outro controlador para atuar sobre esse servomotor. As bibliotecas do Arduino Mega 2560, já possuem funções que lidam diretamente com os servos, conforme já mencionado. O Arduino Mega 2560 foi adquirido por R$ 55,00.

[6] Site Official da Tower Pro: <http://www.towerpro.com.tw/product/mg996r/>. Acesso em: 13 de Jun. de 2017.

5 RESULTADOS

Este capítulo do trabalho apresenta os resultados da implementação dos dois métodos e sua atuação conjunta. A literatura consultada não fornece com exatidão uma forma de validação para cada um dos métodos. Um dos motivos identificados é que a validação é auto evidente, ou seja, se o robô executa a marcha ou não executa. A multiplicidade de fatores e complexidade de cada método também dificulta a criação de critérios de avaliação.

Portanto, será apresentado uma marcha com parâmetros baseado nos da literatura e, posteriormente, a marcha que obteve o melhor desempenho com o robô desenvolvido. Também será apresentado um comparativo entre o λ, tempo de execução da cinemática inversa e erro médio em posição dos *end-effectors*.

Os parâmetros DH tomados para o robô foram os mesmos tomados para Figura 27, sendo assim o robô deste trabalho possui: $L_{L1} = 51{,}01\,mm$, $L_{L2} = 91{,}40\,mm$, $L_{L3} = 92{,}30\,mm$, $L_{L4} = 79{,}86\,mm$ *e* $L_{L5} = 77{,}80\,mm$. Para perna esquerda basta inverter $L_{L1}(-51{,}01\,mm)$, conforme informa Ali, Park & Lee (2010). Entretanto, isso não afeta diretamente a construção das matrizes de transformação da perna, nem os parâmetros DH. Ao final deste capítulo será apresentado os resultados aplicando com o modelo real do robô.

Alternando os parâmetros de entrada, os indicadores apresentados para o trabalho serão: os subpontos gerados para a trajetória de uma marcha completa, os ângulos gerados para uma marcha completa e o tempo de execução para a execução de uma marcha completa. O trabalho tem por objetivo verificar a funcionalidade dos conceitos utilizados. Todos os dados que serão apresentados foram gerados pelo MATLAB (R2015a). Porém, primeiramente, será mostrado algumas considerações feitas nos métodos que foram fundamentais para esse trabalho.

5.1 CONSIDERAÇÕES REFERENTES À ELABORAÇÃO DOS MÉTODOS

Durante o desenvolvimento dos métodos em MATLAB foram encontrados impasses dos quais a literatura consultada não abordava. Tais impasses requeriam soluções que eram desconhecidas e que implicariam na não conclusão das tarefas objetivadas.

5.1.1 Consideração para o Primeiro e Último Passo da Trajetória da Marcha

Na literatura consultada, ao tratar sobre uma marcha frontal é considerado que a marcha já está em andamento, ou seja, o primeiro e o último passo que tendem a serem mais curtos não são levados em conta. Dessa forma foi necessário realizar alterações no método para que a marcha fosse executada corretamente.

As soluções encontradas foram forçar os pontos gerados para que se iniciassem e terminassem onde era desejado. Para o primeiro passo o próprio algoritmo gerava pontos negativos em X para o COM e também para a perna suspensa em movimento. Dessa forma foi aplicado uma condicional para que se iniciassem sempre em zero, e para o último passo também foi aplicada uma condicional para que parasse na distância máxima referente ao número de passos. Para o COM em Y também foi aplicado uma condicional, impedindo que o robô deslocasse a pélvis além do ponto central uma vez que ambos os pés já tivessem atingidos a posição final.

Devido a complicações em se realizar uma só função que pudesse englobar todo equacionamento, a função foi separada em duas. Uma função ficou responsável pelas condicionais para realizar a marcha para um número par e a outra para um número ímpar de passos.

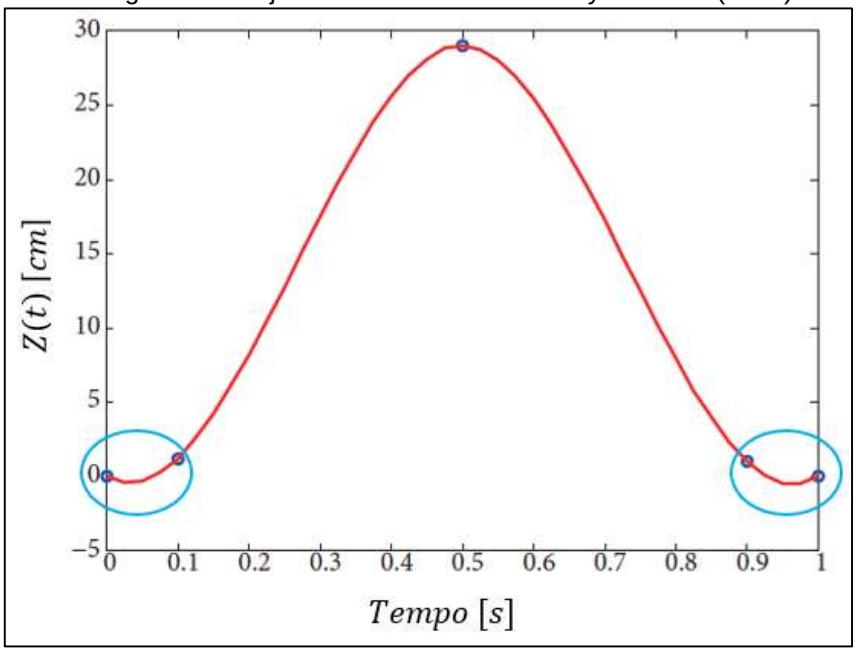
5.1.2 Consideração sobre a Interpolação para a Geração dos Subpontos

Conforme informado anteriormente pretendia-se utilizar *Cubic Spline*. Para o isso o MATLAB possui uma função, *spline()*, que já atende ao que foi descrito no item 2.3.5. Porém, ao invés disso foi usado *Piecewise Cubic Hermite Interpolating Polynomial* (PCHIP) que também possui uma função pronta no MATLAB *pchip()*. Então, a decisão se fez necessária, pois a literatura não informa sobre como contornar o problema de coordenadas negativas dos subpontos gerados em Z. Portanto, foi usado PCHIP que pode ou não manter a continuidade da segunda derivada, segundo a descrição da função em MathWorks[7]. As repostas, entretanto, são similares às apresentas em item (2.3.5). No trabalho de Olcay e Ozkurt (2017), é possível verificar que também há subpontos negativos em Z conforme o círculo azul da Figura 32, porém nada é

[7]Site Mathworks: <https://www.mathworks.com/help/matlab/ref/pchip.html/>. Acesso em: 20 de Nov. de 2017.

informado sobre esse caso. Entretanto, os subpontos não poderiam estar presentes, seria como pontos abaixo do solo.

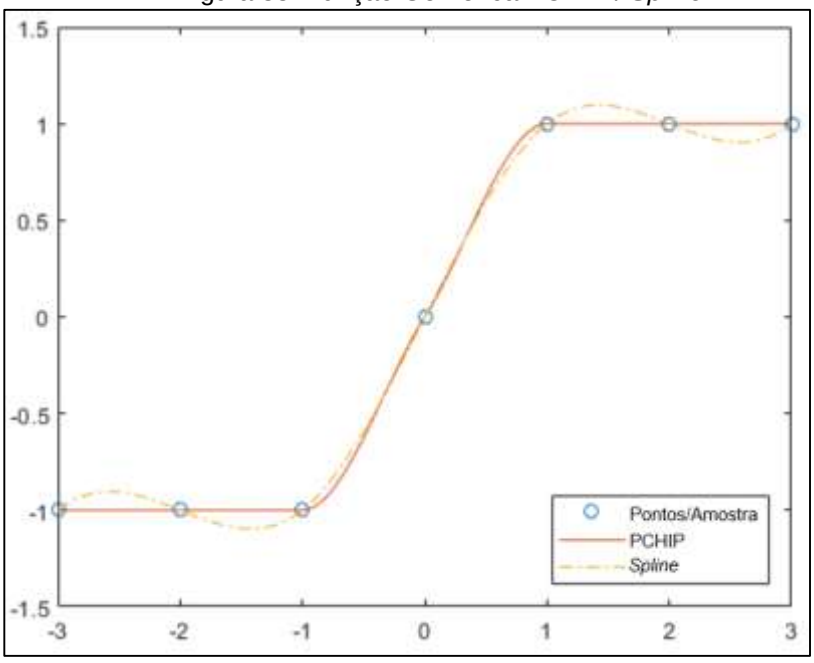
Figura 32: Trajetória do Pé em Z de Olcay e Ozkurt (2017).



Fonte: Adaptado de Olcay e Ozkurt, 2017.

O site do *MathWorks*, dedicado ao software do MATLAB ainda ilustra a diferença entre a função *Spline* e a PCHIP conforme a Figura 33:

Figura 33: Função Gernérica PCHIP / *Spline*



Fonte: Adaptado de MathWorks, 2017.

5.1.3 Consideração para a Cinemática Inversa dos Pés

Conforme mencionado anteriormente foi necessário realizar um ajuste para que os pés mantivessem a posição correta. A trajetória da marcha é passada para os tornozelos, e os pés mantém um comportamento específico. No caso desse trabalho as solas dos pés permanecem sempre em paralelo ao solo e não tem nenhuma inclinação, ou seja, $qs = qe = 0$. Sendo assim, não é possível calcular diretamente as Jacobianas como apresentado anteriormente em (84). Nesse caso as Jacobianas não saberiam que valores atribuir para as duas últimas juntas. Apenas são calculadas para que o erro seja reduzido sem considerar se a sola do pé está em paralelo ao solo, pois os pontos dados são referentes ao tornozelo.

Portanto, a solução encontrada para esse caso foi dividir cada perna em duas. Da pélvis (COM) até o tornozelo, e do tornozelo ao pé. Como se fossem dois manipuladores independentes, porém um conectado ao outro. Assim a cinemática direta é calculada para dois *end-effectors* diferentes, sendo um o tornozelo e o outro o pé. E os pontos passados para o pé são os mesmo valores em X e Y dados para o tornozelo, e o ponto do tornozelo em Z $- L_5$. Sendo L_5 o comprimento do elo pé-tornozelo. Utilizando as equações já apresentadas, tem-se agora (86) e (87):

$$\overrightarrow{\Delta X}_{tornozelo} = \left(\left[X_{end-effector} \;\; Y_{end-effector} \;\; Z_{end-effector}\right] - FK_4^0\right) \tag{86}$$

$$\overrightarrow{\Delta X}_{pé} = \left(\left[X_{end-effector} \;\; Y_{end-effector} \;\; Z_{end-effector} - L_5\right] - FK_6^4\right) \tag{87}$$

As Jacobianas referentes ao tornozelo são separadas conforme (88) e (89):

$$\begin{aligned} J_1 &= [[0\ 0\ 1] \times (FK_4^0 - CoM)] \\ J_2 &= [T_1^0(1 \ldots 3{,}3) \times (FK_4^0 - FK_1^0)] \\ J_3 &= [T_2^0(1 \ldots 3{,}3) \times (FK_4^0 - FK_2^0)] \\ J_4 &= [T_3^0(1 \ldots 3{,}3) \times (FK_4^0 - FK_3^0)] \end{aligned} \tag{88}$$

$$J_{tornozelo} = [J_1 \;\; J_2 \;\; J_3 \;\; J_4] \tag{89}$$

As Jacobianas referentes ao pé são separadas conforme (90) e (91):

$$J_5 = [T_4^0(1 \dots 3,3) \times (FK_6^0 - FK_4^0)]$$
$$J_6 = [T_5^0(1 \dots 3,3) \times (FK_6^0 - FK_5^0)] \quad (90)$$

$$J_{pé} = [J_5 \; J_6] \quad (91)$$

O cálculo do DLS para o tornozelo e para os pés fica como (92) e (93), respectivamente:

$$\Delta\theta_{tornozelo} = J_{tornozelo}{}^T \left(J_{tornozelo} J_{tornozelo}{}^T + \lambda^2 I\right)^{-1} J_{tornozelo}{}^T \overrightarrow{\Delta X}_{tornozelo} \quad (92)$$

$$\Delta\theta_{pé} = J_{pé}{}^T \left(J_{pé} J_{pé}{}^T + \lambda^2 I\right)^{-1} J_{pé}{}^T \overrightarrow{\Delta X}_{pé} \quad (93)$$

Agora os ângulos são gerados separadamente em vetores de 3×4 e 3×2 (tornozelo e pé, respectivamente) para que depois sejam unidos novamente em um só $\Delta\theta$ (94) em 3×6 – para cada perna:

$$\theta = \theta_{tornozelo} + \theta_{pé} \quad (94)$$

Dessa forma na nova iteração os ângulos encontrados são aplicados juntos nas transformações. A verificação do $\|\Delta\theta\|$, agora, deve abranger a condição de ambos pé e tornozelo para cada uma das pernas. Ou seja, um total de quatro verificações que deve ser apurado para que um novo conjunto de pontos seja dado.

Essa forma de cálculo não foi encontrada na literatura utilizada para esse trabalho. Utilizou-se dessa maneira visto que, simplesmente, tentar compensar os ângulos gerados fazendo modificações nas transformações eram necessárias diversas condicionais para tentar avaliar todas as posições em que o pé poderia estar em cada iteração. Essa alternativa foi descartada visto que não se estava obtendo sucesso na correção. Portanto, foi operada essa alteração na própria concepção da estrutura da perna, criando-se dois *end-effectors*, e o tornozelo servindo como base para o pé.

5.2 RESULTADOS OBTIDOS DE PARÂMETROS BASEADOS NA CAMINHADA HUMANA

O número de parâmetros para se testar os algoritmos são diversos. Sabe-se que existem ferramentas matemáticas para o caso de se analisar a influência de múltiplos parâmetros em um sistema, sendo uma delas o MPSA (*Multiparametric Sensitivty Analysis* – Análise Sensitiva Multiparamétrica) conforme menciona Corrêa, et al. (2005). Essa ferramenta seleciona parâmetros a serem testados, e se estabelecem números aleatórios dentro de uma determinada faixa com uma distribuição uniforme. Os testes são realizados utilizando-se uma função matemática que verifica a sensibilidade de cada parâmetro em relação a uma saída.

Infelizmente, não foi possível utilizar tal método ou similar por não ser possível sintetizar os algoritmos dos métodos iterativos em expressões matemáticas diretas. Também é necessário considerar que há múltiplas saídas (pontos e ângulos), e não se tem conhecimento, a partir da literatura pesquisada de algum indicador que classifique o desempenho de uma marcha que leve em conta seus diversos aspectos como, equilíbrio físico, velocidade, tempo de execução, etc.

Tratando-se de forma individual, Erbatur e Kurt (2009) explica que realizou seu trabalho pretendendo alcançar uma trajetória que atingisse a “naturalidade humana” (termo do autor) na sua geração de pontos para a trajetória do COM, portanto parâmetros como altura, comprimento e período do passo, assim como o DSP foram aproximados a dados de caminhada humana. Huang, et al. (2001), em relação à trajetória dos pés e do quadril, também afirma utilizar parâmetros próximos aos dos seres humanos para efetuar seus testes em um simulador dinâmico e um robô real. Seus resultados estão relacionados em possibilitar diversos movimentos com os pés do robô, ajustando os parâmetros referentes às condições do solo, e salientando a criação de um método que possibilita uma movimentação suavizada do quadril sem antes ser necessário traçar uma trajetória ZMP.

Olcay e Ozkurt (2017), cujo trabalho é o que tem maior semelhança a esse (no que diz respeito ao método geração de trajetória) informa que seu método foi aplicado a um robô menor (RUBI), e o resultado apresentado foi uma marcha sem perder o equilíbrio físico. Os autores acrescentam também que as equações da trajetória apresentada foram suficientes para que o robô executasse a marcha sem aplicar outros algoritmos de controle mais sofisticados que iriam requerer unidades de

processamento mais potentes. As equações mencionadas são as mesmas que constam nesse trabalho (tanto para o COM como para o movimento dos tornozelos).

Os aspectos que não estão no escopo desse trabalho não foram comentados nos resultados dos trabalhos da literatura, como questões relacionadas à dinâmica ou eficiência energética, por exemplo. Assim, demonstra-se que a unicidade desses trabalhos em apresentar seus resultados está em um aspecto auto evidente a partir de parâmetros previamente fixados: a própria marcha executada com sucesso utilizando parâmetros à semelhança do ser humano.

Os trabalhos utilizados relacionados ao método de Cinemática Inversa apresentam seus resultados, majoritariamente, em função de tempo de execução, número de iterações necessárias e a lida com as singularidades. Mas, nem todos os trabalhos tratam especificamente sobre robôs bípedes.

Devido à multiplicidade de parâmetros, os parâmetros foram dados com padrões aproximados aos informados pela literatura, para uma caminhada próxima à humana.

Os dados foram aplicados e executados através do MATLAB e os códigos e funções foram elaborados conforme os algoritmos descritos, e encontram-se nos APENDICES E, F, G, H, I, J. O computador (PC) utilizado foi um Lenovo 50Y; Memória RAM: 16Gb; Processador: Intel Core i7-4720HQ – 2.60GHz; Placa de Vídeo: NVIDIA GeForce GTX 860M; Sistema Operacional: Windows 10 – 64 bits.

Os parâmetros para a execução da simulação estão dispostos na Tabela 4:

Tabela 4: Parâmetros da Primeira Simulação

(continua)

L_1	51,01 mm	Comprimento conforme Figura 27;
L_2	91,4 mm	Comprimento conforme Figura 27;
L_3	92,3 mm	Comprimento conforme Figura 27;
L_4	79,86 mm	Comprimento conforme Figura 27;
L_5	77,8 mm	Comprimento conforme Figura 27;
L_{pe}	108,02 mm	Comprimento da sola do pé;
$Número\ de\ Passos$	3	Dois movimentos normais e um movimento final e um inicial curtos;
$Número\ de\ Subpontos$	380	Número de subpontos gerados para 3 Passos;
$B = D_s$	50 mm	Comprimento do Passo;
b	0,2*B mm	ZMP “deslizante” sob a sola do pé de apoio. Valor proporcional extraído de Erbatur e Kurt (2009). Se adequa de acordo com o comprimento do passo.

(conlusão)

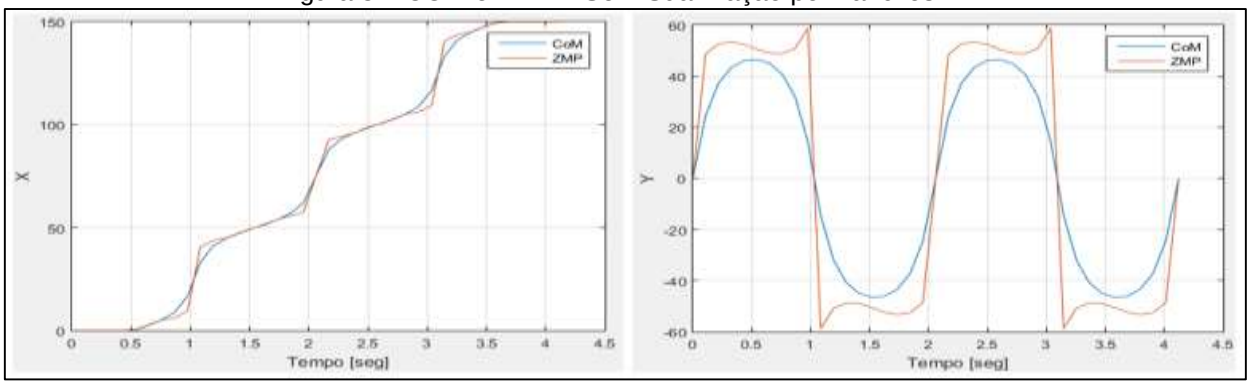
DSP	5%	Kim, Park e Oh (2007) informa que a DSP humana é 10%, como não há dedos nos pés, 5%. Sellaouti, et al. (2006), informa: 0,667%;
d	15	Função: $sinc(k\pi/d)$. Foi feito uma proporcionalidade aproximada com o fator de *Lanczos* em relação ao tempo da DSP a partir de Erbatur e Kurt (2009). Assim, quanto mais próximo de zero, maior o DSP. E quanto mais próximo do número de termos da aproximação de $Fourier + 1\ (N + 1 = 16)$, menor a DSP;
H_{ao}	$20\ mm$	Proporcional à altura máxima em Z do passo utilizado em Huang et al, (2001), e no KHR2 de Kim, Park e Oh (2007);
L_{ao}	$0{,}25 * B$	Proporcional à posição em X da altura máxima do passo utilizado em Huang et al. (2001).
λ	300	Fator de amortização do DLS;
ξ	$\leq\ 0{,}001$	Erro angular através da Normalização Euclidiana ξ=‖Δθ‖, conforme (74). Função *norm()* no MATLAB.

Fonte: O próprio autor (2017).

5.2.1 Resultados do Método de Geração de Trajetória

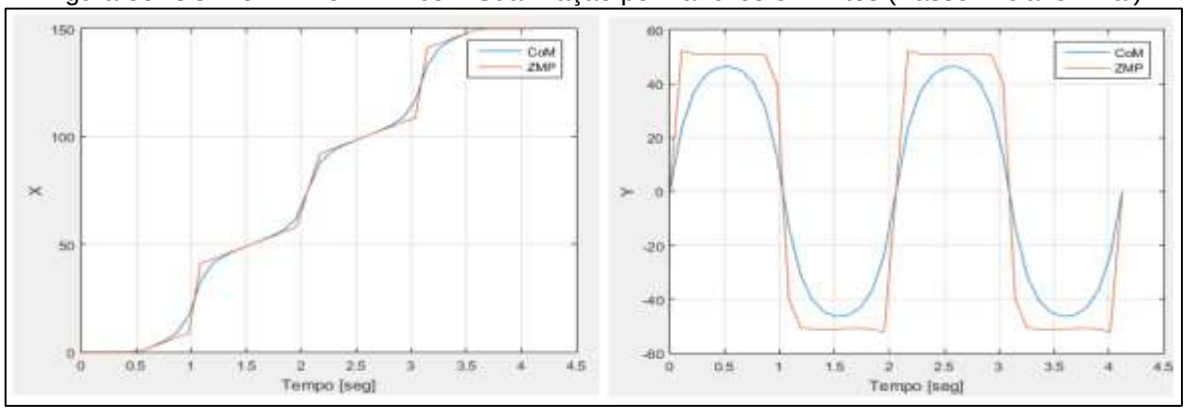
Assim, inicia-se a Etapa 1 descrita no Item 3. Utilizando os parâmetros informados realizou-se a execução do algoritmo descrito, no MATLAB. Inicialmente, faz-se a atribuição dos parâmetros. A função principal chama a função responsável por encontrar os subpontos do COM (pélvis) e os subpontos de referência para o ZMP – *CoMeZMP_Traj*(). Para demonstrar a diferença entre os processos com a suavização por Lanczos e sem a suavização por Lanczos, segue as imagens Figura 34 e Figura 35 o COM e o ZMP em X e Y no tempo, respectivamente.

Figura 34: COM e ZMP – Sem Suavização por Lanczos



Fonte: O próprio autor (2017).

Figura 35: COM e ZMP em Y – com Suavização por Lanczos e Limites (Passo Inicial e Final)

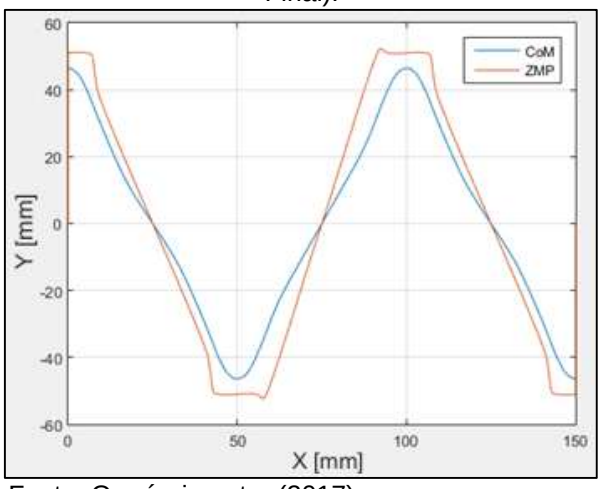


Fonte: O próprio autor (2017).

Os picos da Figura 34 devem, portanto, serem atenuados para que a referência para os subpontos da trajetória dos pés permaneçam corretos. Como já informado os picos são causados pelas harmônicas da aproximação por Série de Fourier. A atenuação pelo fator de Lanczos foi de apenas $d = 15$. Altas frequências das harmônicas ainda permanecem, porém percebe-se pela Figura 35, que de forma inferior. Esse valor foi utilizado, pois ele também afeta o intervalo de DSP. Como não há uma correlação direta entre tempo e d, o fator de Lanczos foi mantido alto (máximo recomendado de 16), pois o DPS é de apenas 5%, assim espera-se alcançar uma compatibilidade entre ambos, uma vez que a literatura também não informa tal.

Na Figura 36 pode-se averiguar o COM e o ZMP sob os aspecto de trajetória somente. Não em função do tempo, mas apenas em distância. A pélvis do robô estará percorrendo a trajetória do COM e a trajetória para os tornozelos serão posicionadas acima dos subpontos máximos do ZMP.

Figura 36: deslocamento COM e ZMP – com Suavização por Lanczos e Limites (Passo Inicial e Final).

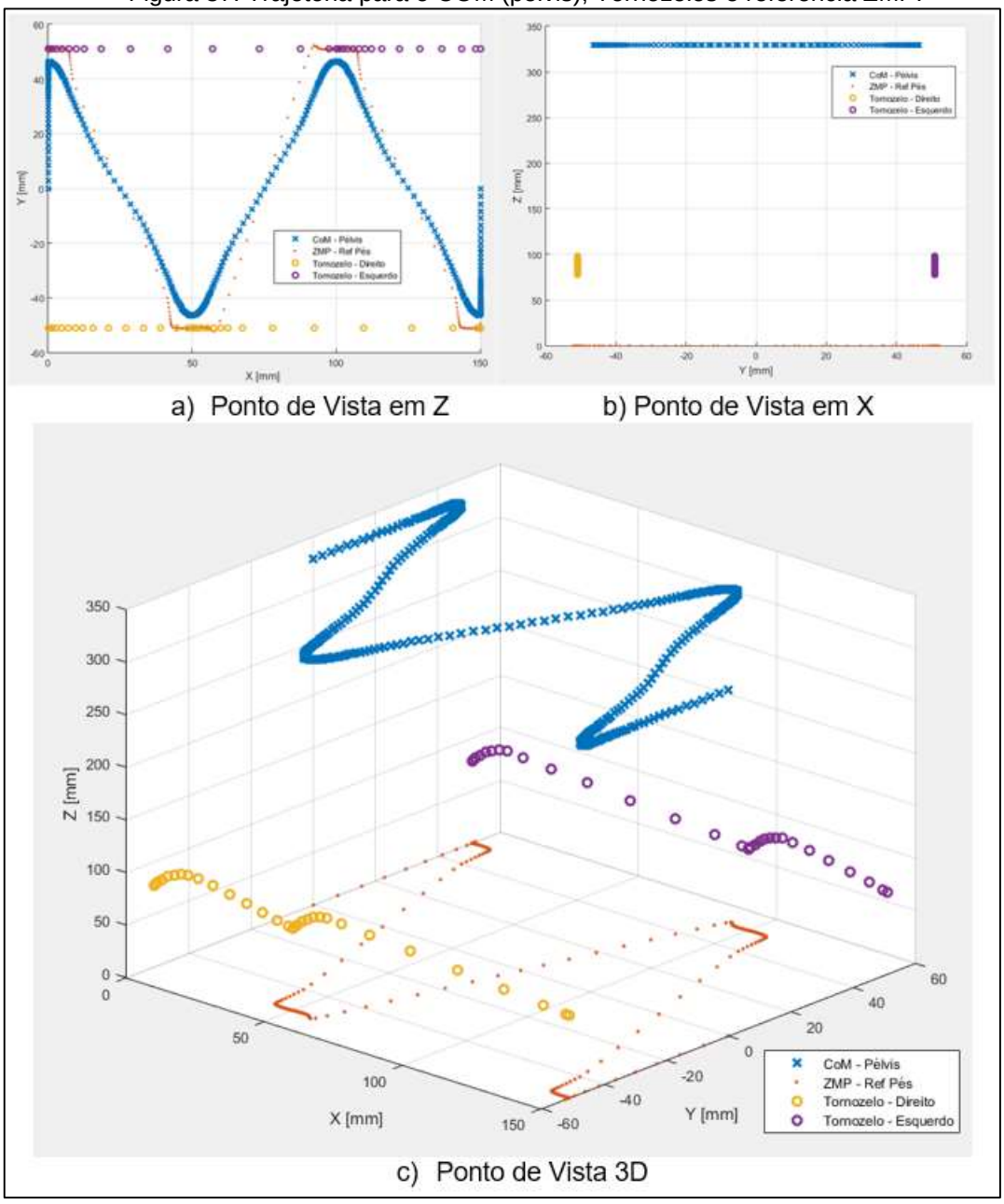


Fonte: O próprio autor (2017).

Dessa forma, prosseguindo o código elaborado para satisfazer o algoritmo tem-se a função *Cubic_3Traj_Impar()* e *Cubic_3Traj_Par()*, responsáveis por encontrar as trajetória para os tornozelos, de números de passos ímpares e pares, respectivamente, conforme já informado. Uma condicional foi aplicada para verificar se o número de passos é ímpar ou par.

Aplicando-se os pontos do tornozelo esquerdo e direito juntamente com o COM e o ZMP tem-se conforme a Figura 37. Na Figura 37.a) tem-se a vista em Z; Na Figura 37.b) tem-se a vista em X; Na Figura 37.c) tem-se a vista 3D.

Figura 37: Trajetória para o COM (pélvis), Tornozelos e referência ZMP.



Fonte: O próprio autor (2017).

É possível perceber também que os pontos não são espaçados uniformemente. Isso se deve ao fato de que a função de *COMeZMP_Traj()* está sob o efeito das harmônicas da série de Fourier e do Fator de Lanczos, e a *Cubic_3Traj_Impar()* e *Cubic_3Traj_Par()* está sob efeito principalmente da razão de DSP: *DPS_ratio* (5%).

Da mesma forma, pode-se observar que onde os subpontos aglomeram-se encontra-se posições onde o robô irá demorar-se mais em relação a posições onde os pontos estão espaçados. Por exemplo, no interior da curva do COM confirma-se que pelo fato dos subpontos estarem mais afastados, o DPS é menor, e nas extremidades o movimento será mais lento, permanecendo mais tempo sob um só pé (em SSP).

São gerados os mesmos números de subpontos para o COM, ZMP e tornozelos. Nesse caso para três passos, 380. Esse número é obtido, pois há cinco equações para cada instante de tempo de meio passo, conforme (37) e (38). Como são 3 passos, itera-se de 0 a 3, ou seja, 4 vezes, para atingir $150\ [mm]$ – distância total. Lembrando que essa marcha executa pé direito/movimento curto – pé esquerdo/movimento longo – pé direito/movimento longo – pé esquerdo/movimento curto. Dessa forma, são gerados 20 valores para 3 passos, em um vetor de 0 a 19. O número de subpontos gerados que irá para PCHIP será com base de 0 a 19 em 0,05 por vez. Tem-se assim 380 subpontos. Para 1 passo, tem-se 180. Para cada passo são acrescidos 100. Na função *COMeZMP_Traj()*, os subpontos são realizados na mesma quantidade para que sejam formados os conjuntos de pontos dos dois tornozelos e do COM.

A escala de tempo para o robô, com os parâmetros informados, totalizou-se em, aproximadamente 4,24 segundos.

5.2.2 Resultados do Método de Cinemática Inversa

Para a próxima etapa, são definidos alguns parâmetros iniciais para o DLS que irá realizar as iterações com base nas Jacobianas. Um dos parâmetros estabelecidos é o fator de amortização λ fixado em 300. A literatura não informa um valor único para se utilizar, e muitas vezes é utilizado uma função que varia esse parâmetro conforme a estrutura e o movimento do robô (isso será visto no próximo item). Para o caso desse trabalho foi escolhido um valor arbitrário (300), conforme os menores valores encontrados de acordo com a Figura 41, Figura 42 e Figura 43, que serão vistas a seguir.

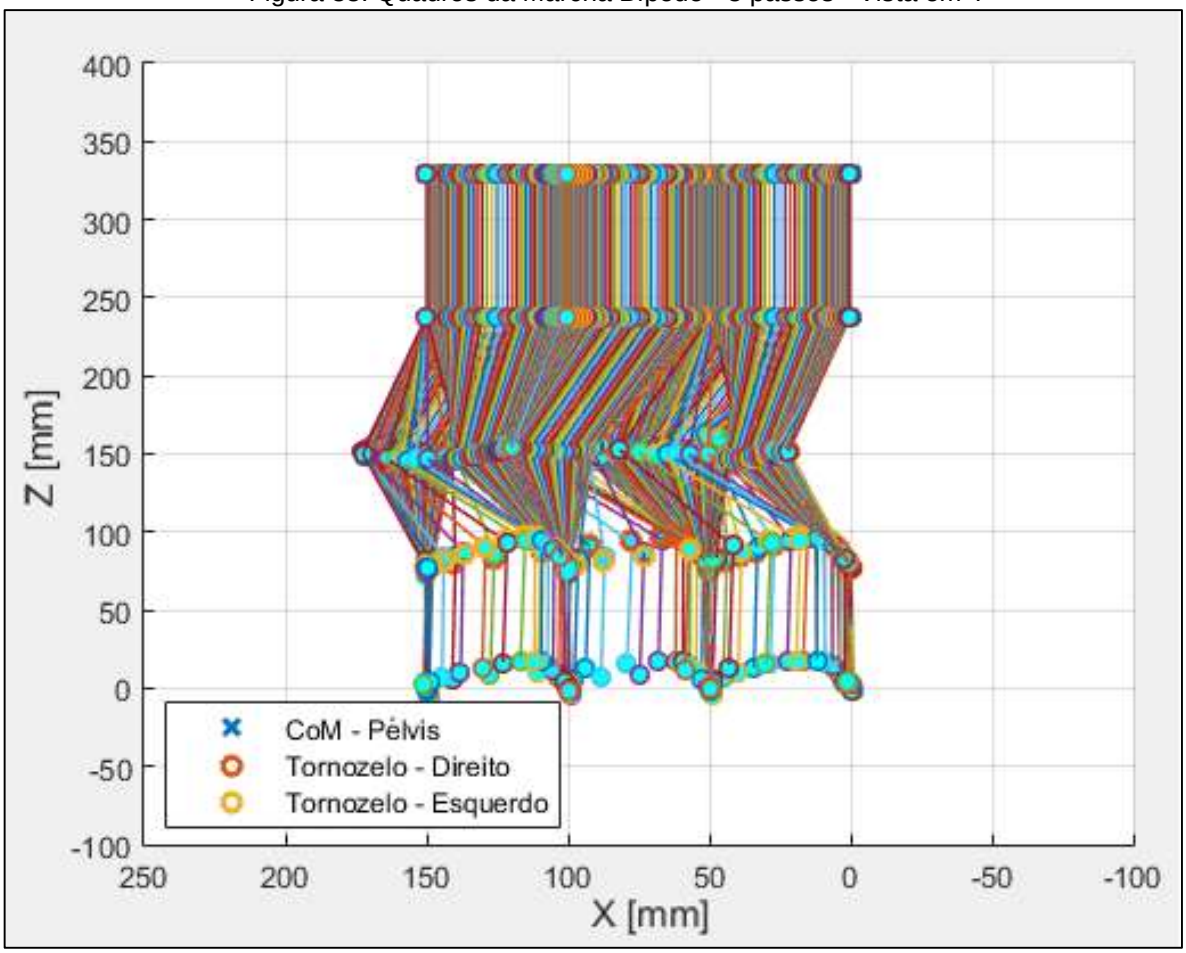
Após, são definidos os ângulos iniciais (θ) para cada perna onde são fixados: [0º -90º -20º 40º -20º 0º], que correspondem a Yaw/Quadril – Roll/Quadril – Pitch/Quadril – Pitch/Joelho – Pitch/Tornozelo – Roll/Tornozelo (para ambas as pernas), respectivamente. Roll/Quadril, recebe -90º por ser um dos parâmetros DH iniciais. Os 3 DOF Pitch recebem [-20º 40º -20º], para formar a curvatura do joelho. Como já relatado, para evitar singularidades e manter a altura do robô constante conforme explica Kim, Park e Oh (2006), simula-se como se os joelhos estivesse flexionados. Os pontos em Z para o COM também não foram dados com a altura total do robô e sim com um valor inferior, aproximado, justamente devido aos joelhos flexionados ($altura\ total - 12\ [mm]$). A literatura consultada não indica valores angulares para esses DOF.

Inicia-se, por fim as iterações do DLS. É atribuído o número do conjunto de subpontos para o número de iterações mínimas que serão necessárias. A cada conjunto, conforme apresentado no fluxograma da Figura 47, é realizado primeiramente, o cálculo das matrizes de transformação de cada perna do robô através de função *T_A*() – APENDICE H, a partir dos valores de posição das transformações, como já apresentado, pode-se encontrar os valores cartesianos dos *end-effectors*, FK (tornozelos e pés) - lembrando-se que cada perna foi dividida em duas Quadril/Tornozelo – Tornozelo/Pé. Após são encontrados as diferenças entre o ponto em que os *end-effectors* se encontram e o ponto desejado $\overrightarrow{\Delta X}$. Com os dados já obtidos é possível realizar o cálculo das Jacobianas, com as funções *Jacobian2()* e *Jacobian_pe()* – APENDICE I. A partir das Jacobianas, o cálculo do DLS é realizado, e complementos aos ângulos são gerados. Os complementos aos ângulos são acrescidos aos originais. Os novos ângulos então são calculados no conjunto completo do algoritmo e ao final será testado o qual o erro da posição do *end-effector* encontrado em relação ao subponto dado. O ξ, como já mencionado, é dado a partir da normalização euclidiana dos valores. O MATLAB possui a função *norm()* para realizar essa operação. E o ξ estabelecido é de $\xi \leq 0{,}001$.

Por fim, ao realizar a iteração com todos os conjuntos de subpontos, encontram-se os ângulos para os 12 DOF. No caso deste trabalho, para os parâmetros informados, apresentam-se os ângulos conforme APENDICE K (Figura 48, Figura 49, Figura 50, Figura 51, Figura 52, Figura 53).

Ainda, para que se possa realizar o desenho de todos os 380 quadros é inserido a função *Desenha_Robo()*, e é chamada toda a vez que um conjunto de ângulos é aceito em $\xi \leq 0{,}001$. Durante a verificação do tempo de execução do algoritmo essa função não foi considerada. Essa função encontra-se em APENDICE J. Para montar a imagem correta é necessário que ela monte a partir das transformadas os pontos de cada junta novamente. Dessa forma, na Figura 38, Figura 39 e Figura 40, é possível visualizar os 380 quadros referentes ao número dos conjuntos de subpontos do robô, em Y, X e 3D, respectivamente.

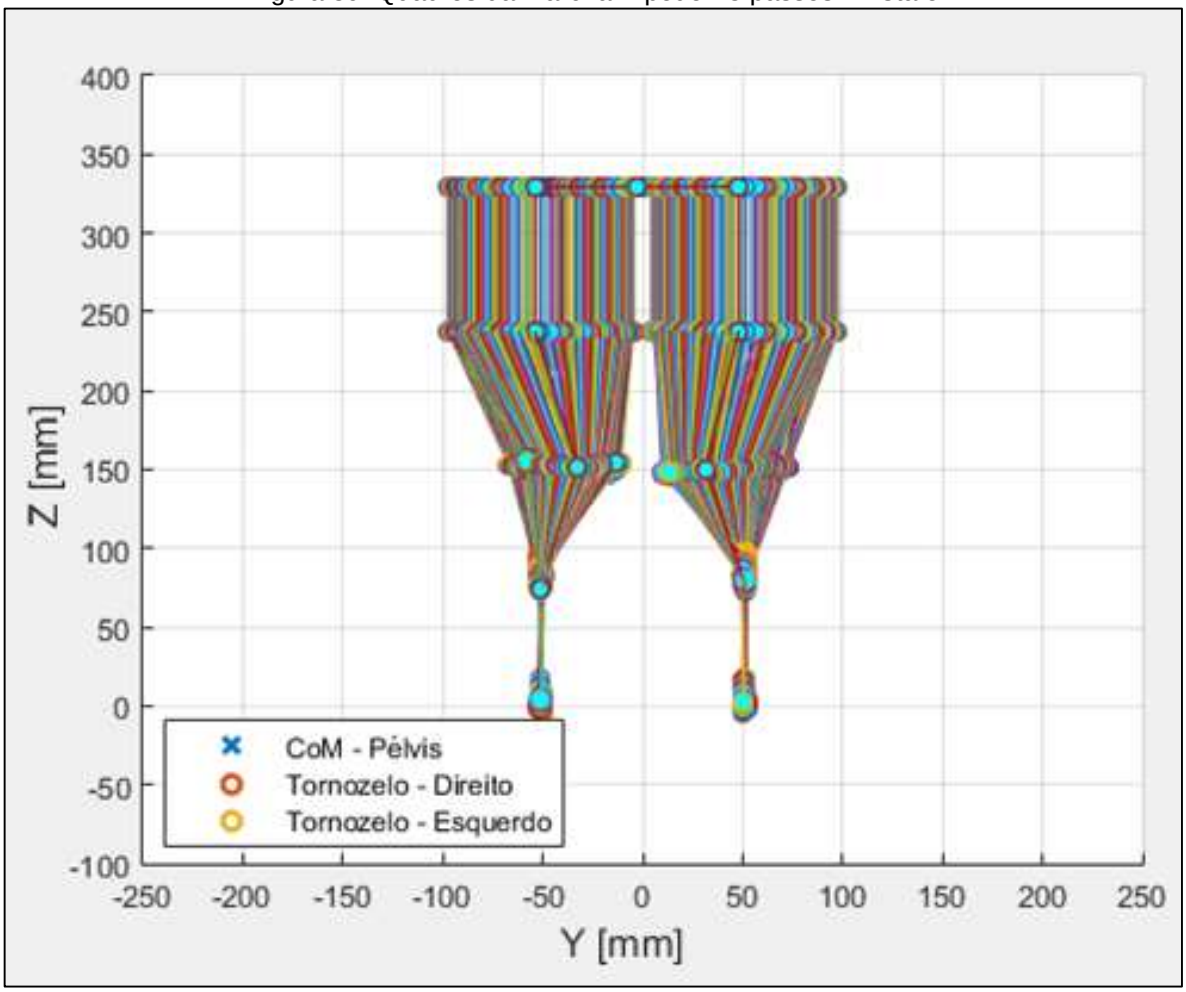
Figura 38: Quadros da Marcha Bípede - 3 passos - Vista em Y



Fonte: O próprio autor (2017).

Devido ao número de quadros percebe-se a visualização dos pontos fica dificultada, mas na vista em Y é possível perceber o robô executando os passos, locomovendo-se em X. É possível perceber também o efeito dos joelhos flexionados.

Figura 39: Quadros da Marcha Bípede - 3 passos - Vista em X

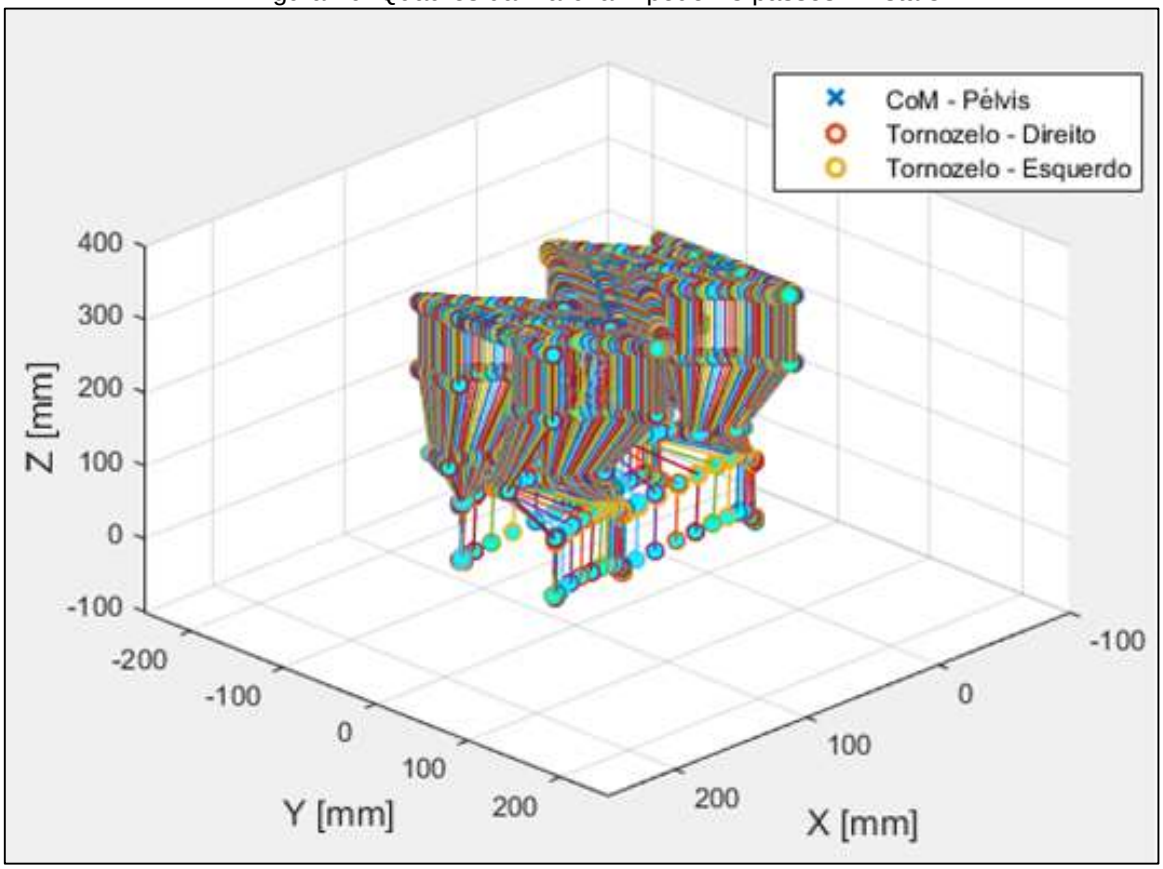


Fonte: O próprio autor (2017).

Na Figura 39 percebe-se que as pernas do robô curvam-se para o interior e isso pode ser corroborado com a Figura 48 onde são apresentados os ângulos de Yaw do quadril de ambas as pernas. Apesar de iniciarem-se em 0º, terminam em aproximadamente +40º e -40º. Testes realizados com um número maior de passos mostram que os ângulos dessa junta tendem a aproximar-se ainda mais dos extremos.

Na Figura 40, pode-se verificar com mais clareza o deslocamento realizado pelo COM (pélvis), passando pelos momentos de SSP e DSP durante a marcha. E da mesma forma também é possível visualizar as pernas curvadas para dentro. Esse efeito foi causado pois foram os melhores ângulos gerados pela Jacobiana/DLS e somente uma modificação nas Jacobianas, ou a aplicação de um método restritivo, poderá alterar isso. Na práticas as consequências disso serão vistas adiante.

Figura 40: Quadros da Marcha Bípede - 3 passos - Vista 3D



Fonte: O próprio autor (2017).

A Tabela 5 apresenta dados referentes ao tempo que o PC levou para realizar cada etapa. ZMP/COM e COM, foram apresentadas separadamente, pois uma vez que o COM e a trajetória dos tornozelos estão estabelecidas, não é mais necessário estabelecer novamente a referência para o ZMP. Porém, para fins comparativos é apresentado a diferença em segundos quando ambos são executados e quando somente o COM é executado.

Tabela 5: Tempo de Execução: ZMP/COM, Trajetória dos Tornozelos e DLS/Jacobianas – Marcha: Três Passos.

Etapa	Tempo $[seg]$
ZMP/COM	299.016
COM	214.060
Traj. Tornozelos	0.177
DLS/Jacobianas	409.486

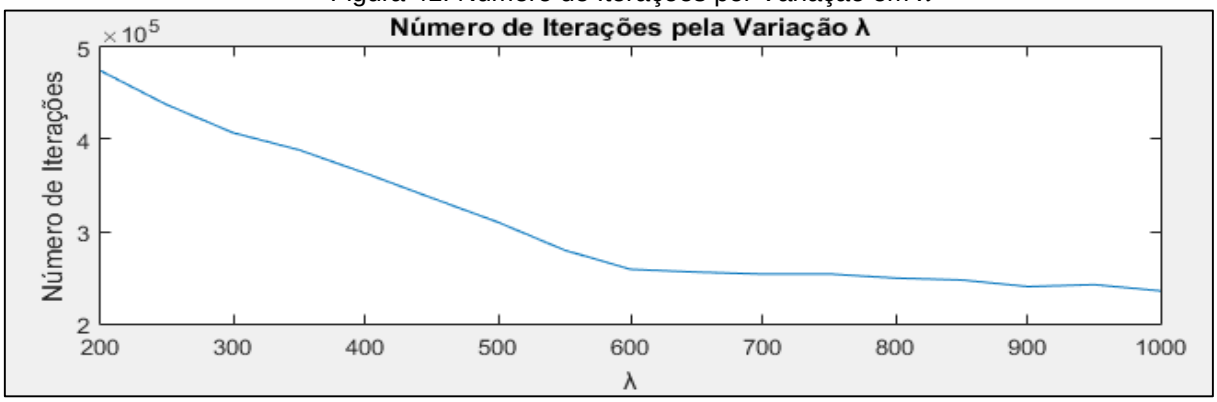
Fonte: O próprio autor (2017).

Os tempos, em segundos, foram extraídos a partir da função *tic toc* do MATLAB. O tempo mensurado pode variar dependendo do rendimento e dos outros processos

executados pelo computador no momento da execução. Ainda assim é possível perceber que o tempo de execução da função da trajetória dos tornozelos é insignificante próximo aos valores obtidos das outras funções. Para o ZMP e COM, tem-se as equações de *Fourier* e *Sinc* utilizando-se variáveis simbólicas. Para o DLS e Jacobianas tem-se um número elevado de iterações com cálculos matriciais. O número de iterações necessárias para realizar o algoritmo, dado os parâmetros informados foi de 407118.

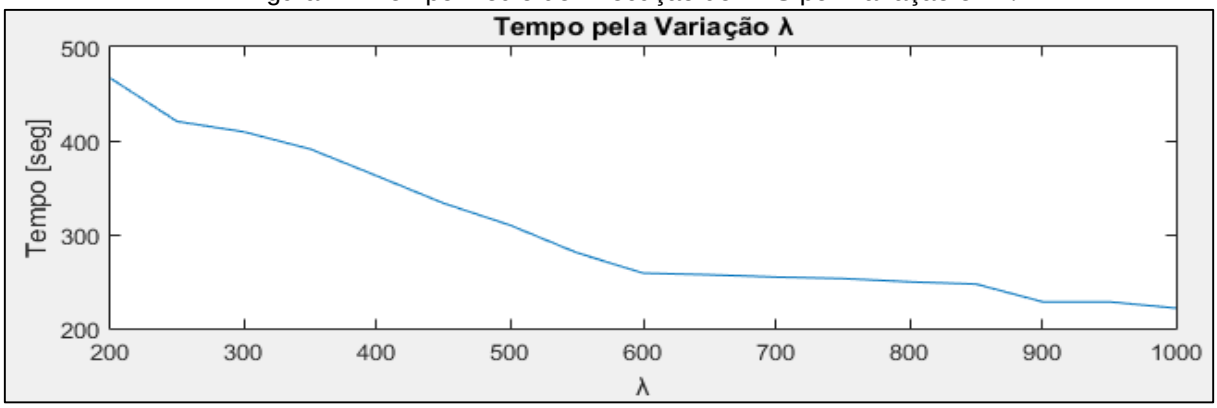
Também foi realizado uma análise onde uma sequência de testes demonstra o efeito do λ em relação ao número de iterações, conforme a Figura 41. Conforme a Figura 42 demonstra-se o tempo de execução do método da Cinemática Inversa, e a Figura 43 o efeito em relação ao erro de posição. Para o presente caso tem-se: $1{,}987\ [mm],\ 2{,}007\ [mm],\ 2{,}088\ [mm],\ 2{,}094\ [mm]$, para o erro do tornozelo direito, do tornozelo esquerdo, do pé direito e do pé esquerdo, respectivamente.

Figura 41: Número de Iterações por Variação em λ



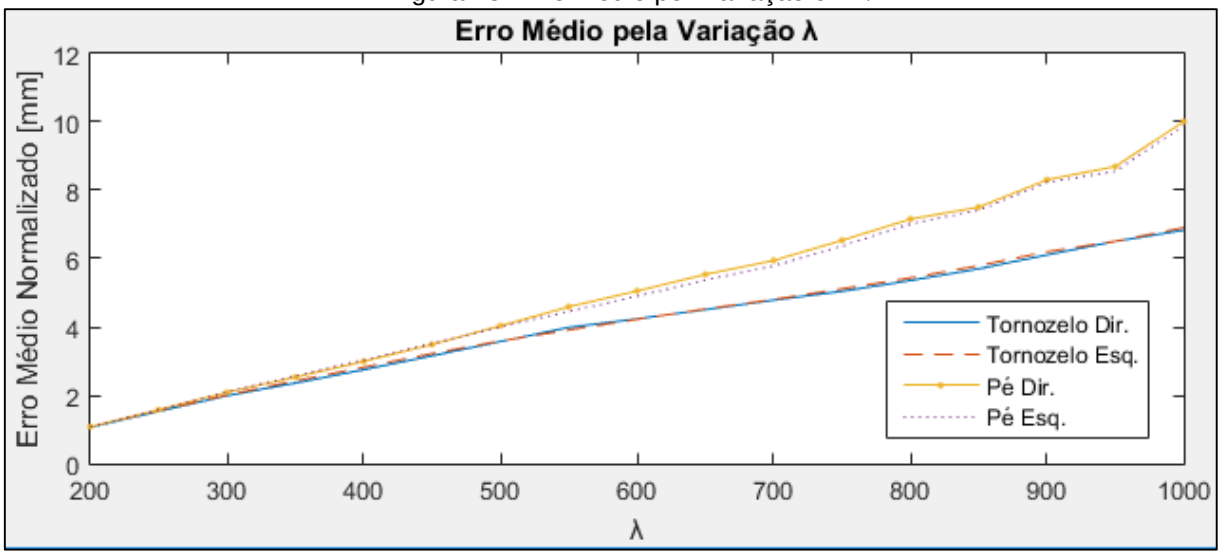
Fonte: O próprio autor (2017).

Figura 42: Tempo Médio de Execução do DLS por Variação em λ



Fonte: O próprio autor (2017).

Figura 43: Erro Médio por Variação em λ



Fonte: O próprio autor (2017).

É possível perceber que quando maior o λ, mais rápido é o tempo de execução e menor o número de iterações, porém, como já mencionado λ afeta a reposta em saída.

No capítulo 6 desse trabalho serão abordadas as considerações finais sobre esses resultados assim como sugestões de possíveis implementações para melhorar o desempenho.

Pode ser feita animações com os quadros obtidos. Imagens separadas puderam ser geradas, e a partir do comando *getframe()* e *movie2avi()* do MATLAB, para facilitar a visualização do desempenho. Um dos parâmetros a serem ressaltados para uma correta visualização é o *fps (frames per second* – quadros por segundo*).* Tem-se 380 quadros e o tempo para a execução da marcha é de 4,24 segundos. Para encontrar-se o *fps* basta dividir um pelo outro, arredondando-se tem-se, 90 *fps.*

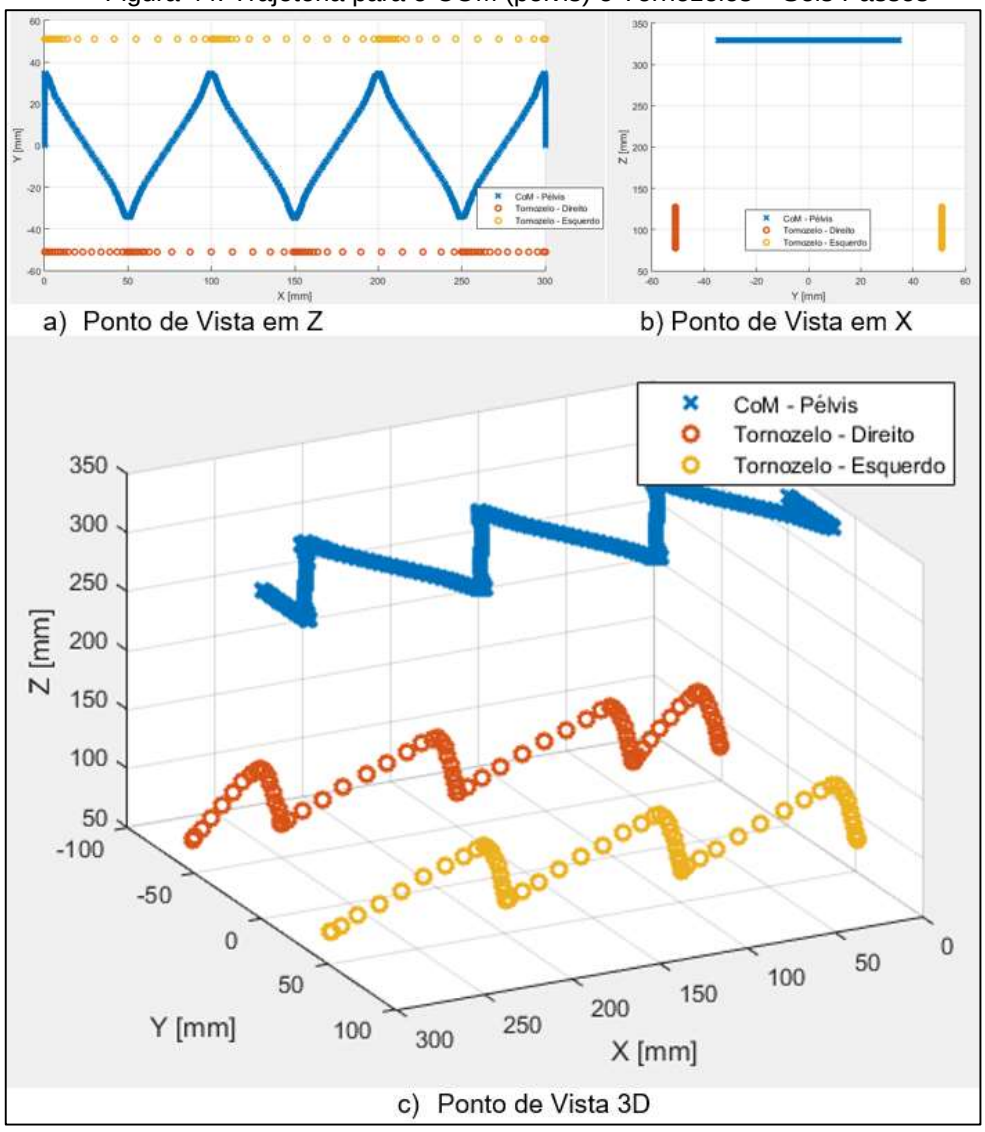
5.3 TESTES DE MARCHA COM O ROBÔ DESENVOLVIDO

Conforme informado, a fim de se facilitar a demonstração prática com o robô, a sintaxe para o Arduino Mega 2560, foi gerada em planilha eletrônica e então passada para o ambiente de programação do microcontrolador. Assim, não se correram riscos como perda de dados. O atraso aplicado ao final de cada passagem dos doze ângulos foi de 10 $[ms]$. Esse valor foi dado para que coincidisse com o valor do tempo da marcha. No APENDICE é apresentado o código.

Foram realizados os testes com os parâmetros informados de acordo com a literatura. Entretanto, os resultados não convergiram para uma marcha com equilíbrio físico. O que pode ser constado como fatores prejudiciais, no que tange aos dados gerados, foram os ângulos de Yaw (direito e esquerdo). As pernas convergiram internamente e acabaram por acavalarem-se em contado com o solo. Constatou-se, também, que os pés não mantinham a elevação de 2 $[cm]$ que fora passada, por não conseguir manter o equilíbrio físico e, consequentemente, não era possível dar nenhum passo sem que o robô fosse à queda.

Uma vez que o robô não se manteve, diversos outros testes foram considerados. E os melhores resultados obtidos até então, foram utilizando-se outros parâmetros. A trajetória do COM, Tornozelo e Pé encontra-se na Figura 44.

Figura 44: Trajetória para o COM (pélvis) e Tornozelos – Seis Passos



Fonte: O próprio autor (2017).

Os comprimentos dos elos mantiveram-se iguais. O número de número de passos foi aumentado para seis, para que o robô pudesse permanecer mais tempo em uma caminhada contínua. O número de subpontos gerados consequentemente passou para 680. Manteve-se o comprimento do passo em $B = 50\ [mm]$, assim como $b = B * 0{,}2\ [mm]$. O DSP foi alterado para 20%, em uma tentativa e manter o robô mais tempo com ambos os pés no solo. Dessa mesma forma o parâmetro de *Lanczos* d foi alterado para $d = 2$. Com isso a amplitude máxima do COM é reduzida, fazendo com que oscile menos para as laterais. A altura máxima do pé foi alterada para $H_{ao} = 50\ mm$ em uma tentativa para se evitar que os pés se chocassem com o solo antes de finalizar uma passada. $L_{ao} = 0{,}25 * B$ foi mantido. Como foram adicionados mais três passos decidiu-se alterar o fator de amortização para $\lambda = 400$, aumentando o erro de posição, porém reduzindo o tempo de iteração. $\xi \leq 0{,}001$ foi mantido. O tempo de espera para cada sequência de ângulos para o Robô também foi mantido em $10\ [ms]$.

Uma vez que as restrições não foram aplicadas, e pelo fato de que os ângulos de Yaw gerados mantiveram a pernas do robô voltadas para dentro produzindo ângulos próximos a +110º e -110º, novos valores também foram gerados, de forma forçada, considerando 0º em ambos os DOF. Os outros ângulos também acabam por ser influenciados por isso, e as próprias iterações vêm a compensar-se.

Mesmo não sendo essa a forma correta de aplicação (conforme será visto no item 7), o equilíbrio físico do robô melhorou. Os ângulos podem ser visualizados no APENCIDE L (Figura 54, Figura 55, Figura 56, Figura 57, Figura 58, Figura 59);

Os tempos de execução obtidos seguem conforme a Tabela 6.

Tabela 6: Tempo de Execução: COM, Trajetória dos Tornozelos e DLS/Jacobianas – Marcha: Seis Passos.

Etapa	Tempo [seg]
COM	151,656
Traj. Tornozelos	0,096
DLS/Jacobianas	790.976

Fonte: O próprio autor (2017).

É possível verificar que o tempo para o caso de seis passo é inferior para o COM, em relação ao COM com três passos. Isso se deve, pelo que foi analisado, que as repostas com d inferiores, por serem mais amortecidas em relação às repostas com d superiores, tão mais rápidas de se executar. Para o algoritmo da trajetória, não é

possível afirmar se o mesmo ocorre devido ao uso da função tic toc. Uma vez que seria necessário analisar os processos do computador no momento da execução. Na escala de tempo registrada foi possível identificar alterações em outras ocasiões.

O número de iterações do DLS/Jacobianas foi de 821471. Na Tabela 7 é possível verificar o erro médio de cada um dos *end-effectors*:

Tabela 7: Erro Médio com λ = 400 – Marcha: Seis Passos.

Erro Médio Tornozelo Direito	2.621 [mm]
Erro Médio Tornozelo Esquerdo	2.391 [mm]
Erro Médio Pé Direito	2.922 [mm]
Erro Médio Pé Esquerdo	2.703 [mm]

Fonte: O próprio autor (2017).

O tempo de execução para o robô foi de, aproximadamente, 7,12 segundos. Foi gerada a animação a partir dos quadros, e o número de *fps* também resultou em 90 para esse caso.

A forma com que os ângulos resultantes foram passados para o Arduino, e desse para os servomotores, foi a mesma informada anteriormente.

Como já mencionado o robô foi criado para servir como demonstração dos métodos fora do ambiente de simulação. Entretanto, diversos problemas podem ser relatados a seu respeito. Como já mencionado o robô foi construído sem nenhum tipo de realimentação de dados ou sensoriamento. Foi realizado dessa maneira, pois não era esse o enfoque do trabalho. Entretanto, tornou-se visível que é necessário que o robô tenha uma forma de avaliar se está de fato cumprindo sua trajetória conforme estipulado. Incertezas e imprecisões mecânicas também deveriam ser relatadas, entretanto não foi realizado um estudo minucioso referente ao centro dos eixos de cada junta e qual é, de fato, o COM do robô. Como é sabido os métodos de geração de trajetória recomendam que o torso/pélvis do robô seja consideravelmente mais pesado que as pernas para que o método seja válido, entretanto, pelo fato de haver seis MG996R em cada perna, e somente uma peça mantendo ambas unidas, sabe-se que essa proporção não está sendo respeitada. Também não houve um estudo referente à criação das peças, sobre qual o material recomendado e sua distribuição de massa.

Constatou-se também que apesar de ter-se utilizado o servo MG996R inspirado em outro robô (ROFI), é recomendável que para um projeto futuro utilize-se outro

motor com maior precisão e resistência. Cinco MG996R foram inviabilizados durante os testes e tiveram de ser substituídos.

A peça traseira do servomotor teve de ser alterada, furando-se para que os eixos pudessem ser acoplados a fim de reduzir a instabilidade. No APENCICE O, pode-se visualizar as quatro peças (vermelhas) que foram utilizadas diretamente do ROFI. Duas peças foram usinadas e alteradas a partir do ROFI (verdes), as originais não foram utilizadas, pois por um erro de fabricação não permitiam comportar os MG996R. A peças pretas foram usinas com material ABS e as peças brancas foram impressas em uma impressora 3D, devido à dificuldade em usiná-las. A peças usinadas oferecem maior resistência e durabilidade que as impressas. Onde seria o torso do robô, foi acoplado o Arduino MEGA 2560 e juntamente uma pequena *protoboard* para conectar os fios provindos dos doze servomotores para o Arduino. Não foi feita uma placa específica para unir os fios devido aos constantes testes e alterações realizados. Ainda que projetado de forma simplificada em relação aos robôs comerciais trata-se de um conjunto de peças frágil e delicado. No APENDICE O (Figura 61, Figura 62 e Figura 63) pode ser verificado o robô desse trabalho.

A alimentação dos servos foi realizado com uma fonte de bancada suportando 32V – 3A. Os servos, conectados em paralelo à fonte foram alimentados com, aproximadamente, 6,5V e momentos de pico foi registrado, aproximadamente 2,9A. O Arduino foi alimentado via USB pelo próprio PC.

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou o levantamento teórico para o desenvolvimento de um algoritmo para marchas de robôs bípedes de 12DOF de forma *off-line*. Os métodos de geração de trajetória e cinemática inversa executaram uma marcha frontal com três e seis passos. Para a realização da marcha com três passos foram utilizados parâmetros baseados/inspirados pela literatura pesquisada para esse trabalho, onde seus objetivos convergiam em realizar marchas que possuíssem uma “naturalidade humana”. Na simulação, os testes realizados foram bem sucedidos, porém o robô desenvolvido não foi capaz de manter-se em equilíbrio durante as performances.

Para o teste com seis passos, algumas alterações nos parâmetros foram realizadas visando corrigir problemas de desempenho apresentados pelo robô, desconsiderando a “naturalidade humana” de caminhada. O número de passos foi dobrado procurando-se manter o robô mais tempo em marcha contínua. O DLS e a altura máxima dos pés foi aumentado. A amplitude máxima do deslocamento da pélvis (COM) em Y foi reduzida, e as juntas do quadril (Yaw) foram suprimidas, para que as pernas não se curvassem para seu interior durante a marcha. Obteve-se sucesso nas simulações realizadas. No caso da execução dos métodos com o robô também não foi possível que se equilibrasse sozinho. Entretanto, identificou-se uma melhora. A melhora consistiu que, além do robô conseguir efetuar os movimentos, ao auxiliar a manter o quadril sem inclinações no eixo, ocasionada por erros mecânicos, o robô conseguiu executar as marchas.

Uma vez que os pontos são gerados a partir do COM e são encontradas as posições dos pés e tornozelos o erro – principalmente mecânico – propaga-se ao se ter o pé como base para toda a estrutura robótica, durante a caminhada. Assim, evidenciou-se a necessidade da construção, ou aquisição, de um robô calibrado, que tenha o devido sensoriamento para percorrer e executar a trajetória proposta.

É necessário salientar as três soluções para problemas encontrados durante o desenvolvimento que não constavam na literatura pesquisada. Primeiramente, a literatura tratava um robô em marcha contínua e não com início e fim – pés em paralelo um ao outro. Assim, condicionais foram aplicadas no algoritmo para que encontrassem o primeiro e o último passo do robô não importando seu número total. Também houve necessidade em se alterar o método de interpolação, mencionado pela literatura, pois não foi possível – ao menos ao se utilizar a função *spline()*, do

MATLAB – que o robô mantivesse todos os subpontos em Z para os pés, acima do solo. Como já apresentado, uma das literaturas colocou o gráfico que descreviam os movimentos em Z do pé de seu robô com pontos negativos. Entretanto, nenhuma delas providenciou nenhum tipo de explicação para esse fato. Por último também foi necessário compensar a junta dos tornozelos. Uma vez que o pé deve manter-se sempre em paralelo ao solo, cada subponto foi passado para o tornozelo e o subponto do pé derivou-se desse mantendo X e Y e decrementando Z do último elo. Dessa forma garantiu-se que o último elo permanecesse idealmente sempre perpendicular ao solo, e a sola do pé em paralelo ao solo. Para que isso fosse possível toda a estrutura do algoritmo teve que ser modificada.

O método de geração de trajetória utilizado, foi capaz de gerar os pontos da marcha através do equacionamento por ZMP, interpolações e outros métodos auxiliares, desprezando os fatores da dinâmica e atendo-se somente aos fatores cinemáticos. Os valores encontrados para o COM e ZMP puderam ser gerados conforme consta na literatura utilizada nesse trabalho. Dessa forma é possível utilizar tal método para marchas *off-line*.

Os ângulos das juntas também puderam ser gerados a partir do método iterativo de Jacobianas/DLS e foi capaz de se encontrar os conjuntos de subpontos passados. O erro angular e, consequentemente, em posição era esperado, pois fica em função de λ. Portanto, são funcionais para uma marcha *off-line.* O método utilizado também funciona para outras topologias robóticas alterando-se os parâmetros DH.

Ambos os métodos, porém com ênfase no que foi desenvolvido para a Cinemática Inversa, podem ser utilizados em trabalhos futuros, incluindo no auxílio a tecnologias assistivas, como já mencionado. Os conceitos apresentados, assim como a literatura de onde foram extraídos, apresentam não só uma contribuição para a robótica bípede, mas também para a robótica industrial e podem ser aplicados para braços/manipuladores industriais.

Por se tratar de um tema amplo, outros aspectos devem ser considerados até que se possa desempenhar e monitorar uma marcha bípede com um robô de fato. Ainda assim, não levando em conta o número de iterações e o tempo necessário para executar, os métodos comportam-se como previsto nos trabalhos consultados. Entretanto, da forma como foram realizados, ainda exigem um tempo elevado de computação e execução se comparados aos outros trabalhos. Na forma como apresentado nesse trabalho ainda não seria possível realizar marchas *online,* por

exemplo, onde a resposta deve vir no espaço de tempo em que o robô está realizando os movimentos.

A literatura utilizada não abordou com profundidade os aspectos necessários de um projeto completo de marcha e de um robô bípede contendo todas as suas etapas. Também não foi possível obter critérios de avaliação e métricas de desempenho especificamente para os métodos. Os trabalhos abortados na maioria das vezes apresentavam o método proposto, mas com resultados que estavam fora do escopo deste trabalho, pelo fato de se utilizar um robô completo e funcional. O próprio robô desenvolvido foi uma tentativa minimizada de suprir essa questão, porém nele apenas foi considerado que se respeitasse a morfologia dos 6 DOF por perna. Isso demonstra a necessidade em se expandir o tema para que se possa reproduzir e comparar resultados.

7 TRABALHOS FUTUROS

Na literatura abordada não foi encontrado métodos para otimizar o processo do cálculo do COM/ZMP e/ou acelerá-lo. Acredita-se que um dos fatores para isso é que por ser um método comumente utilizado para marchas off-line esse fator seja muitas vezes negligenciado. As melhorias mais expressivas encontradas dizem respeito à Cinemática Inversa. Sobre o geração de trajetória há uma variedade de outros métodos a serem considerados conforme o item 2.2, e dos quais Luksch (2010) aborda.

Sobre a Cinemática Inversa, o método por DLS utilizado pode ainda ser expandido de outras formas além da utilizada nesse trabalho. Yan e Huang (2007) informam que para seu trabalho utilizaram RDLS (*Robust Damped Least Squares* – Quadrados Mínimos Amortecidos Robustos) para evitar as singularidades – mesmo método proposto por Nakamura e Hanafusa (1986), e também WLN (*Weighted Least-Norm* – Normalização de Mínimos Ponderados) para evitar a singularidade dos limites das juntas – sendo essa a forma correta de se aplicar os limites. O segundo, afirma Huang e Yan (2007), é importante para que o robô atue de forma mais próxima ao movimento humano real.

No RDLS, conforme informa Huang e Yan (2007) e Nakamura e Hanafusa (1986) a constante de amortecimento ajuda a evitar a singularidade, mas ela também afeta a resposta em $\dot{\theta}$. Portanto, não é interessante que λ seja aplicado em respostas não singulares. O método proposto por Nakamura e Hanafusa (1986) para um método robusto de DLS, consiste em estabelecer um critério para o amortecimento (95):

$$\lambda = \begin{cases} \lambda_o \left(1 - \frac{h}{h^s}\right) & se\ h < h^s \\ 0 & outros \end{cases} \tag{95}$$

Onde (96):

$$h(\theta) = \sqrt{\det(JJ^T)} \tag{96}$$

Quando h se aproxima de zero, significa que está se aproximando de uma singularidade. Assim, se $h < h^s$, uma vez que h^s é o valor limite estabelecido, λ é ajustado automaticamente, conforme informa Huang e Yan (2007). Assim, de acordo com (95), λ se mantém atuante quando está próximo de singularidades e não atua

quando não está próximo. Ainda, λ_o é a constante de fato, dado numericamente para cada configuração de manipulador e trajetória. O h^s é estipulado de acordo um ângulo limite de determinada junta.

Ambos os métodos ajudariam a resolver os problemas da constante de λ estabelecida e dos limites das juntas que não foram estabelecidos, a não per pelas juntas de Yaw que foram forçadas a zero. Os resultados de Yan e Huang (2007), utilizando RDLS e WLN, com uma Jacobiana convencional fora de 90 iterações por passo, aproximadamente e um tempo de iteração de 0,6 segundos, por passo.

Buss (2009) comenta a respeito do SVD (*Singular Value Decomposition* – Decomposição do Valor Singular) que trata-se de uma ferramenta de análise da Jacobiana Pseudo-Inversa e do DLS. Buss e Kim (2004) informa sobre o SDLS (*Selectively Damped Least Squares* – Quadrados Mínimos Amortecidos Seletivos), usado por Maciejewski e Klein (1988), onde utilizam-se dos parâmetros do SVD também para encontrar diferentes λ. Esse é dependente da configuração do corpo articulado, das posições relativas do *end-effector* e do ponto desejado.

Maciejewski e Klein (1985) também oferecem critérios para descrição do espaço de trabalho do robô, e dessa forma pode-se evitar colisão com objetos. Similarmente, poder-se-ia considerar tal aplicação para a não colisão de uma perna na outra, no caso do bípede. Abordando o problema similar de colisão das pernas, Yan e Huang (2007) propõem a *F-Jacobian* (*Fixed Led Jacobian* – Jacobiana de Perna Fixa), um método que toma como referência a perna suspensa, o COM e a inclinação do quadril em relação à perna que permanece no solo durante um passo. Dessa forma as pernas passam a não agir mais independente uma da outra e sim a agir em conjunto. Essa consideração é realizada na própria matriz Jacobiana. Com isso Yan e Huang (2007), informam que os movimentos verificam possíveis colisões da perna com o solo e com a outra perna.

Os resultados de Yan e Huang (2007) utilizando RDLS e WLN, com a F-Jacobiana são ainda mais expressivos em relação a esse trabalho: 30 iterações por passo, aproximadamente, e um tempo de iteração de 0,35 segundos, por passo.

Referente ao desempenho do robô físico desenvolvido é necessário ressaltar, primeiramente, de que um estudo sobre a dinâmica deve ser realizado. Em Craig (2010) é possível encontrar informações sobre isso. O presente trabalho negligenciou as questões dinâmicas propositalmente devido a sua complexidade.

Park, Kim, et al. (2005) informa sobre diversos sensores utilizados no KHR2, como sensor Força/Torque utilizados podem medir a força normal e dois momentos (Roll e Pitch). Esses sensores, são fundamentais para compensar a trajetória do ZMP. Acelerômetros, giroscópios e módulos de sensores de inércia também foram utilizados. O acelerômetro pode detectar a inclinação do robô, porém é sensível a impactos. O giroscópio detecta a velocidade angular, mas pode não responder de forma adequada em baixas frequências. Por isso, é necessário tratar o sinal através de filtros passa-baixa para os acelerômetros e utilizar filtros passa alta para o giroscópio, selecionando a parte do sinal que interessa à inclinação angular.

Ainda, comparando-se com o KHR2, Kim, Park e Oh (2006), informa que foi utilizado como controlador principal uma placa comercial com vários periféricos (assim como módulos LAN e CAN). Ou seja, protocolos de comunicação possivelmente serão necessários para realizar a comunicação entre um controlador principal e os subcontroladores que gerenciarão os sensores e motores.

Com sensores e controladores, de acordo com Kim, Park e Oh (2006), é possível realizar interações com o robô ajustando a marcha e corrigindo a trajetória proposta. Também é possível realizar reduções de impactos, assim como corrigir inclinações indesejadas. Utilizando-se sensores de pressão nas solas também é possível detectar inclinações do solo. O objetivo desse trabalho não incluía uma estrutura dessa magnitude, porém, somente que respeitasse a disposição das juntas. Contudo, pode-se perceber que para que o robô execute as trajetórias propostas, um sistema robusto deve ser construído. Espera-se que esse trabalho venha a ser continuado e que as sugestões mencionadas sejam implementadas e testadas em um próximo projeto.

REFERÊNCIAS

ALI, Muhammad A., PARK, Andy H., & LEE, George S. **Closed-Form Inverse Kinematic Joint Solution for Humanoid Robots**. The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems. Taipé, Taiwan, 2010.

BUSS, Samuel R., **Introduction To Inverse Kinematics With Jacobian Transpose, Pseudoinverse And Damped Least Squares Methods**. Department of Mathematics University of California, 2009.

BUSS, Samuel R., KIM, Jin S. **Selectively Damped Least Squares for Inverse Kinematics.** Journal of Graphics Tools, 2005.

CORRÊA, Jeferson M., FARRET, Felix A., POPOV, Vladimir A., SIMÕES, Marcelo G. **Sensitivity Analysis of the Modeling Parameters Used in Simulation of Proton Exchange Membrane Fuel Cells.** IEEE Transactions on Energy Conversion, 2005.

CRAIG, John J. **Robótica**. 3. ed. São Paulo: Pearson Education do Brasil, 2012.

CUDOWSKI, Andrew. **Design, Simulation and Control of a 12 DOF Biped Robot.** Lakehead University, 2009.

DALEN, Swan van J. **A Linear Inverted Pendulum Walk Implemented on TUlip***.* Eindhoven University of Technology, Eindhoven, 2012.

ENGARDT, Max, HEIMBURGER, Axel, SYDHOFF, Philip. **Manipulability Index Optimization for a Planar Robotic Arm.** Bachelor Thesis in Engineering Physics, Department of Mathematics, Division of Optimization and Systems Theory KTH, Royal Institute of Technology, Estocolmo, Suécia, 2012.

ERBATUR, Kemalettin, KURT, Okan. **Natural ZMP Trajectories for Biped Robot Reference Generation**. IEEE Transactions on Industrial Electronics, 2009.

GAUTAM, Rakesh, PATIL, Akshay, T., **Modeling and Control of Joint Angles of a Biped Robot Leg using PID Controllers,** IEEE International Conference on Engineering and Technology (ICETECH), Coimbatore, Índia, 2015.

HONDA, **Honda-Robotics.** Disponível em: <http://www.honda.co.jp/robotics/>. Acesso em: 13 de Jun. 2017.

HONG, Man B., SHIN, Young J., & WANG, Ji-Hyeun. **Novel Three-DOF Ankle Mechanism for Lower-Limb Exoskeleton: Kinematic Analysis and Design of Passive-Type Ankle Module.** International Conference on Intelligent Robots and Systems, 2014.

HUANG, Han-Pang, YAN, Jiu-Lou, **A Fast and Smooth Walking Pattern Generator of Biped Robot Using Jacobian Inverse Kinematics.** Workshop on Advanced Robotics and Its Social Impacts, 2007.

HUANG, Qiang, YOKOI, Kazuhito, KAJITA, Shuuji, KANEKO, Kenji, ARAI, Hirohiko, KOYACHI, Noriho, TANIE, Kazuo. **Planning Walking Patterns for a Biped Robot.** IEEE Transactions on Robotics And Automation, 2001.

JUNG, Yeongtae, & BAE, Joonbum. **Kinematic Analysis of a 5-DOF Upper-Limb Exoskeleton With a Tilted and Vertically Translating Shoulder Joint.** IEEE/ASME Transactions On Mechatronics, v. 20, n. 3, 2015.

KIM, Jong-Wook. **Online Joint Trajectory Generation of Human-like Biped Walking.** International Journal Of Advanced Robotic Systems, 2014.

KIM, Jong-Wook, TRAN, Tin T., DANG, Chien V., & KANG, Bongsoon. **Motion and Walking Stabilization of Humanoids Using Sensory Reflex Control.** International Journal of Advanced Robotic Systems, 2016.

KIM, Jong-Yup, PARK, Ill-Woo, & OH, Jun-Ho. **Experimental Realization of Dynamic Walking of the Biped Humanoid Robot KHR-2 Using Zero Moment Point Feedback and Inertial Measurement.** Advanced Robotics, v. 20, n. 6. VSP and Robotics Society of Japan, 2006.

KIM, Jung-Yup., PARK, Ill-Woo, & OH, Jun-Ho. **Walking Control Algorithm of Biped Humanoid Robot on Uneven and Inclined Floor.** Journal of Intelligent and Robotic Systems, 2007.

KIM, J.-Y., PARK, Ill-Woo, LEE, Jungho, KIM, Min-Su, CHO, Beak-Kyu, & OH, Jun-Ho. **System Design and Dynamic Walking of Humanoid Robot KHR-2.** International Conference on Robotics and Automation, Barcelona, Spain: IEEE, 2005.

LOPES, António M. **Robótica Industrial - Modelação Cinemática e Dinâmica de Manipuladores de Estrutura em Série.** Mestrado em Automação, Instrumentação e Controlo, 2002.

LUKSCH, Tobias. **Human-like Control of Dynamically Walking Bipedal Robots.** Technischen Universitat Kaiserslautern, 2010.

LUO, Ren C., & LIN, Siang J. **Impedance and Force Compliant Control for Bipedal Robot Walking on Uneven Terrain.** IEEE International Conference on Systems, Man, and Cybernetics, 2015.

MACIEJEWSKI, Anthony A., KLEIN, Charles A. **Numeric filtering for the operation of robotic manipulators through kinematically singular configurations**. Journal of Robotic Systems, 1988.

MACIEJEWSKI, Anthony A., KLEIN, Charles A. **Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments,** International Journal of Robotic Research, 1985.

MIYADAIRA, Alberto N. **Desenvolvimento, Construção e Controle de um Robô Móvel Bípede com Tronco.** Campinas, São Paulo: Universidade Estadual de Campinas, 2011.

NAKAMURA, Yoshihiko, HANAFUSA, Hideo. **Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control.** Journal of Dynamic Systems, Measurement, and Control, 1986.

NEW ATLAS, **Gallery**. Disponível em: <http://newatlas.com/new-honda-asimo-robot/32977/#gallery>. Acesso em: 13 de Jun. 2017

OLCAY, Tolga, OZKURT, Ahmet. **Design and Walking Pattern Generation of a Biped Robot.** Turkish Journal of Electrical Engineering & Computer Sciences, 2017.

PARK, Ill-Woo, KIM, Jung-Yup, PARK, Seo-Wook, & OH, Jun-Ho. **Development of Humanoid Robot Platform KHR-2 (KAIST Humanoid Robot - 2).** Korea Advanced Institute of Science and Technology (KAIST), Guseong-dong, Yuseong-gu, Daejeon - República da Coréia, 2004.

PROJECT BIPED, **Rofi**. Disponível em: <http://www.projectbiped.com/prototypes/rofi>. Acesso em: 04 de Jun. 2017.

RUDY, Joseph. **Zero-Moment Point Walking Controller for Humanoid Walking Using Darwing-Op***.* University of Notre Dame, Notre Dame, Indiana, 2014.

SANTEC, **Manipulador Tridimensionais Robôs Modelo 1**. Disponível em: <https://www.santec.ind.br/manipulador-tridimensionais-robos-modelo-1/> Acesso em: 13 de Jun. 2017.

SHIMMYO, Shuhei, SATO, Tomoya, & OHNISHI, Kouhei. **Biped Walking Pattern Generation by Using Preview Control Based on Three-Mass Model.** IEEE Transactions On Industrial Electronics, v. 60, n. 11, 2013.

SMASHING ROBOTICS, **Thirteen Advanced Humanoid Robots for Sale Today**. Disponível em: https://www.smashingrobotics.com/thirteen-advanced-humanoid-robots-for-sale-today/>. Acesso em: 21 de abr. 2017.

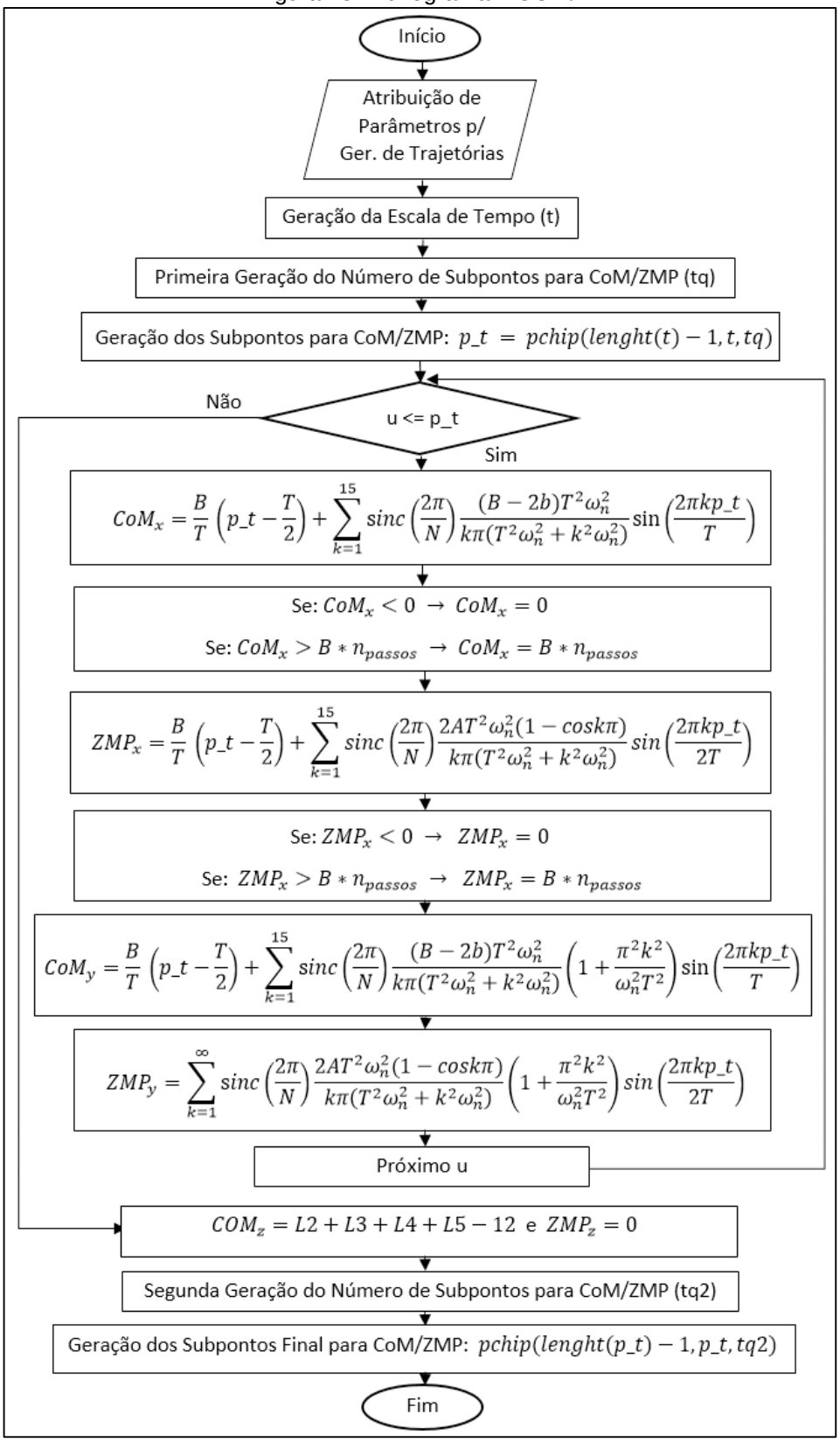
UNIVERSIDAD DE SANTIAGO DE CHILE VIRTUAL. **Elementos Constitutivos de un Robot Industrial.** Disponível em: <http://www.udesantiagovirtual.cl/moodle2/mod/book/tool/print/index.php?id=24908#ch210>. Acesso em: 13 de Jun. 2017.

VUKOBRATOVIC, Miomir, BOROVAC, Branislav. **Zero-Moment Point — Thirty Five Years Of Its Life**. International Journal of Humanoid Robotics, 2004.

WANG, Mingfeng, CECCARELLI, Marco, CARBONE, Giuseppe. **A Feasibility Study on The Design and Walking Operation of Abiped Locomotor Via Dynamic Simulation.** Higher Education Press and Springer-Verlag Berlin Heidelberg, 2016.

APÊNDICE A – ALGORITMO DE GERAÇÃO DE TRAJETÓRIA – COM

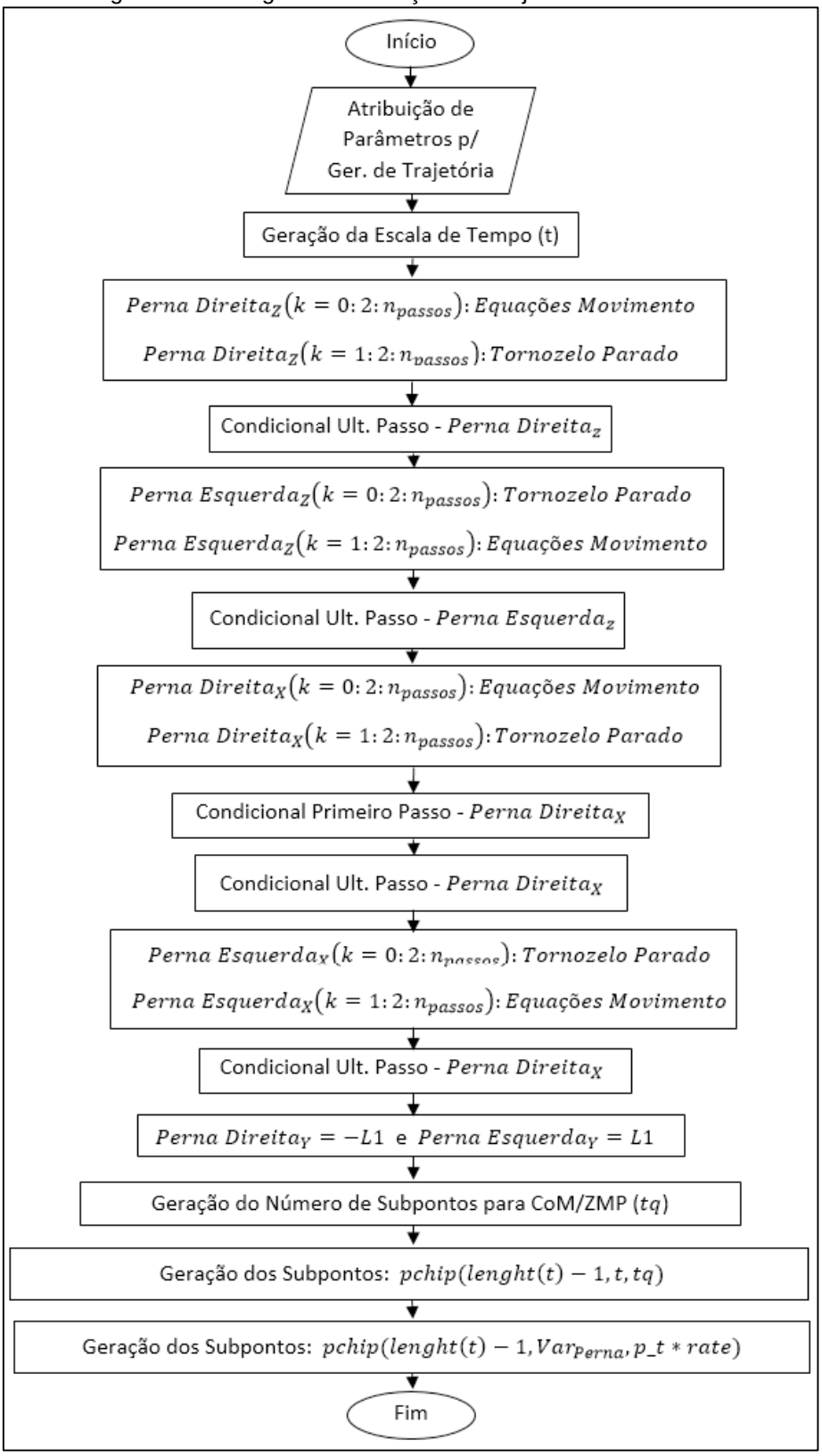
Figura 45: Fluxograma – COM/ZMP



Fonte: O próprio autor (2017).

APÊNDICE B – ALGORITMO DE GERAÇÃO DE TRAJETÓRIA PARA OS TORNOZELOS/PÉS

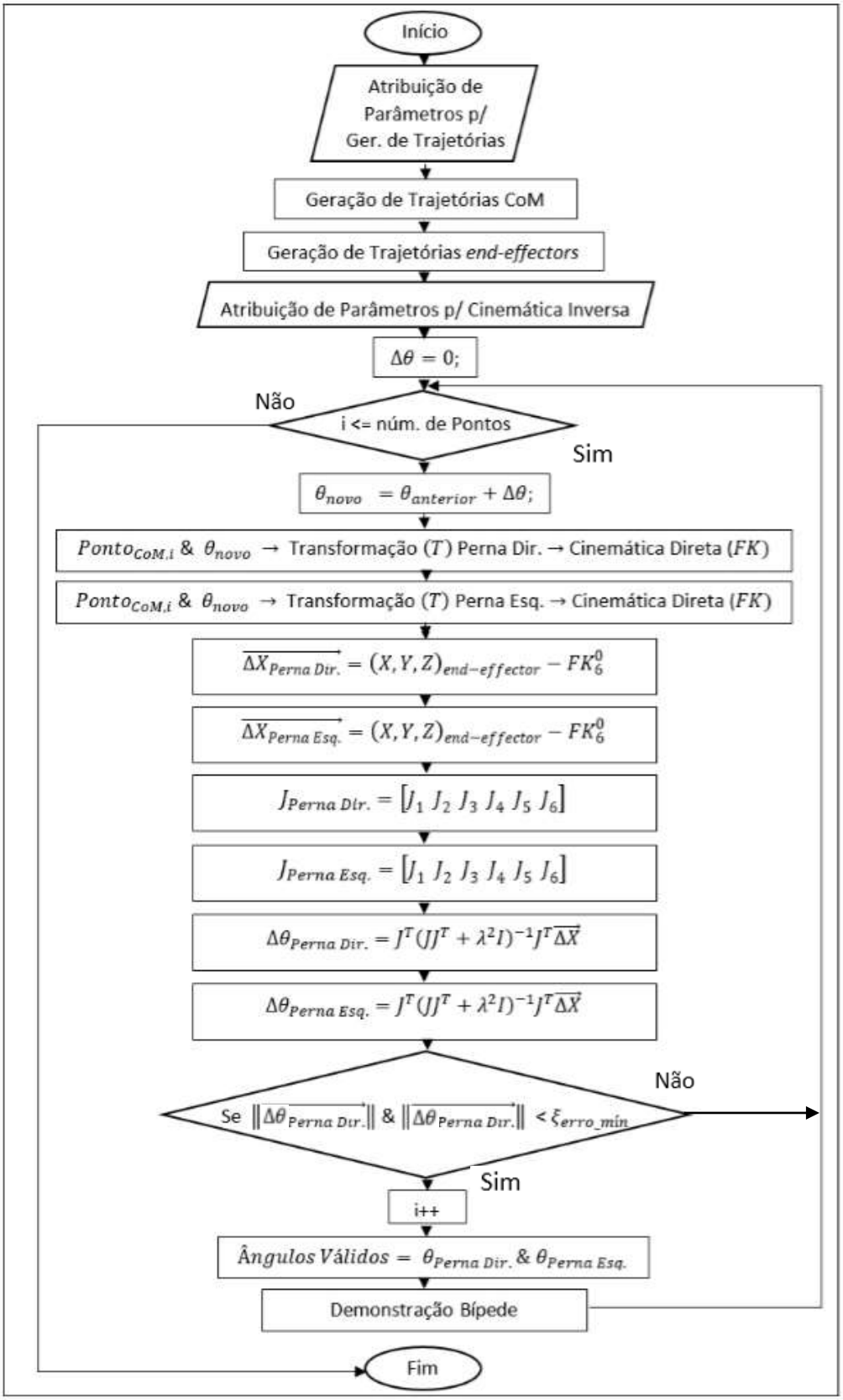
Figura 46: Fluxograma – Geração de Trajetória – Tornozelos



Fonte: O próprio autor (2017).

APÊNDICE C – ALGORITMO DA CINEMÁTICA INVERSA

Figura 47: Fluxograma - Cinemática Inversa



Fonte: O próprio autor (2017).

APÊNDICE D – CÁLCULO DAS MATRIZES DE TRANSFORMAÇÃO

Assumindo que: $s\theta_i \equiv sin\theta_i,\ c\theta_i \equiv \cos\theta_i,\ s\alpha_i \equiv sin\alpha_i,\ c\alpha_i \equiv \cos\alpha_i$; e $'a','\alpha','d'$ e $'\theta'$ são os parâmetros DH. Tem-se:

Orientação Inicial: Perna direita em Y – Marcha para frente em X:

$$T_{Base} = \begin{bmatrix} 0 & -1 & 0 & CoM_X \\ 1 & 0 & 0 & CoM_Y \\ 0 & 0 & 1 & CoM_Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{97}$$

$$Base_{Translação} = \begin{bmatrix} 1 & 0 & 0 & L1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -L2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{98}$$

$$T_0^{Base} = T_{Base} * Base_{Translação} \tag{99}$$

Junta 1:

$$T_1^0 = R_{\theta_1} R_{\alpha_1} P_{a_1} P_{d_1} \tag{100}$$

$$T_1^0 = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_1 & -s\alpha_1 & 0 \\ 0 & s\alpha_1 & c\alpha_1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{101}$$

Junta 2:

$$T_2^1 = R_{\theta_2} R_{\alpha_2} P_{a_2} P_{d_2} \tag{102}$$

$$T_2^1 = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & 0 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_2 & -s\alpha_2 & 0 \\ 0 & s\alpha_2 & c\alpha_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{103}$$

Junta 3:

$$T_3^2 = R_{\theta_3} R_{\alpha_3} P_{a_3} P_{d_3} \tag{104}$$

$$T_3^2 = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & 0 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_3 & -s\alpha_3 & 0 \\ 0 & s\alpha_3 & c\alpha_3 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (105)$$

Junta 4:

$$T_4^3 = R_{\theta_4} R_{\alpha_4} P_{a_4} P_{d_4} \quad (106)$$

$$T_4^3 = \begin{bmatrix} c\theta_4 & -s\theta_4 & 0 & 0 \\ s\theta_4 & c\theta_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_4 & -s\alpha_4 & 0 \\ 0 & s\alpha_4 & c\alpha_4 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_4 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_4 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (107)$$

Junta 5:

$$T_5^4 = R_{\theta_5} R_{\alpha_5} P_{a_5} P_{d_5} \quad (108)$$

$$T_5^4 = \begin{bmatrix} c\theta_5 & -s\theta_5 & 0 & 0 \\ s\theta_5 & c\theta_5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_5 & -s\alpha_5 & 0 \\ 0 & s\alpha_5 & c\alpha_5 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_5 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_5 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (109)$$

Junta 6:

$$T_6^4 = R_{\theta_6} R_{\alpha_6} P_{a_6} P_{d_6} \quad (110)$$

$$T_6^5 = \begin{bmatrix} c\theta_6 & -s\theta_6 & 0 & 0 \\ s\theta_6 & c\theta_6 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & c\alpha_6 & -s\alpha_6 & 0 \\ 0 & s\alpha_6 & c\alpha_6 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_6 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (111)$$

Transformações:

$$\begin{gathered} T_{Base}^1 = T_{Base}^0 T_0^1 \\ T_{Base}^2 = T_{Base}^0 T_0^1 T_1^2 \\ T_{Base}^3 = T_{Base}^0 T_0^1 T_1^2 T_1^3 \\ T_{Base}^4 = T_{Base}^0 T_0^1 T_1^2 T_2^3 T_3^4 \\ T_{Base}^5 = T_{Base}^0 T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 \\ T_{Base}^6 = T_{Base}^0 T_0^1 T_1^2 T_2^3 T_3^4 T_4^5 T_5^6 \end{gathered} \quad (112)$$

APÊNDICE E – CÓDIGO EM MATLAB DA FUNÇÃO DE GERAÇÃO DE TRAJETÓRIA PARA O COM E ZMP

```
function [COMx2, COMy2, COMz2] = COM_Traj(L1, L2, L3, L4, L5, L_pe, B,
n_passos, DSP_ratio)
g = 9.8*1000; %gravidade
A = L1;   %é a distância do pé ao centro, tem que ser L1
l = L2+L3+L4;
f = (1/(2*pi()))*sqrt(g/l);
T = 1/f; %o step time é a metade do período da caminhada
w = 2*pi()*f;

%B = 0.2*1000 %Ds
b = B*0.2;  %no artigo a proporção é 0.2;

syms k
ksum = 1:15;
S = zeros(size(ksum));
j = 1;

%DSP_ratio não é usado, pois ele é um percentual sobre um período, e o N é
um fator do sinc, não diretamente correspondente a tempo
%N = length(ksum)+1;
N = 15;
num=n_passos; %num é uma medida de tempo, mas como o T é 1, então equivale
ao numero de passos;

Kdsp = DSP_ratio;  %double support ratio - esse pode ser dado pelo N da
outra função (mas ZMP só referencia)
Td = T*Kdsp;
Tm = (T+Td)/2;

%-------------------Formação-Escala-Tempo-------------------------------
n=n_passos;
i=1;
for o=0:1:n
    ti(i) = o*T;
    i=i+1;
    ti(i) = o*T+Td;
    i=i+1;
    ti(i) = o*T+Tm;
    i=i+1;
    ti(i) = o*T+T;
    i=i+1;
    ti(i) = o*T+T+Td
    i=i+1;
end
%-------------------Ajusta-Escala-Tempo---------------------------------
%Necessário remover o tempo extra do DSP do último passo, pois é
%inexistente
if ti(i-1)>n*T+T
    ti(i-1)=n*T+T
end
%--------------Primeira Etapa Ajuste de Subpontos---------------------
i
t=0:(ti(i-1)/(i-2)):ti(i-1)  %o i-2 por causa que começa em zero.
x2=0:length(t)-1;
tq1_generic2=0:0.5:length(t)-1;
```

```
p_t = pchip(x2,t,tq1_generic2);
%----------------------------------------

for u = 1:length(p_t) %como T é aproximadamente 1, então t = 4, mas tem que
comp. início e fim.
    for i = 1:length(ksum)
     cx = sinc(k/N)*(((B-
(2*b))*(T^2)*(w^2))/((k*pi())*(((T^2)*(w^2))+((k^2)*(pi()^2)))))*(sin((2*pi
()*k*p_t(u))/T));
     %cx = (((B-
(2*b))*(T^2)*(w^2))/((k*pi())*(((T^2)*(w^2))+((k^2)*(pi()^2)))))*(sin((2*pi
()*k*p_t(u))/T)); %sem sinc
     Scx = ((B/T)*(p_t(u)-(T/2)))+symsum(cx,k,1,ksum(i));
     if Scx < 0
        Scx = 0;
     elseif Scx > B*(num)
        Scx = B*(num);
     end
    end

     for i = 1:length(ksum)
       cy = sinc(k/N)*((2*A*(T^2)*(w^2)*(1-
cos(k*pi()))))/(k*pi()*((T^2)*(w^2)+(k^2)*(pi()^2))))*(sin((2*pi()*k*p_t(u))
/(2*T)));
       %cy = ((2*A*(T^2)*(w^2)*(1-
cos(k*pi()))))/(k*pi()*((T^2)*(w^2)+(k^2)*(pi()^2))))*(sin((2*pi()*k*p_t(u))
/(2*T))); %sem sinc
       Scy = symsum(cy,k,1,ksum(i))+0;
     end

    %Lanczos sigma function é dada por uma função sinc
    %é multplicada no 'py', pois o artigo explica que ela é para
    %suavizar e para definir o DSP.
    %Ela atenua os fatores de maior frequência da aproximação de Fourier.
    %função sinc já leva o Pi


    Sum_cX(j) = double(Scx);
    Sum_cY(j) = double(Scy);
    j = j + 1;
end
for i=1:length(Sum_cX)
    COMz(i) = L2+L3+L4+L5-12;
end

x=0:length(p_t)-1
tq1_generic=0:0.1:length(p_t)-1

COMx2 = pchip(x,Sum_cX,tq1_generic);
COMy2 = pchip(x,Sum_cY,tq1_generic);
COMz2 = pchip(x,COMz,tq1_generic);

    %hold on;
    %grid on;
    %scatter3(COMx2,COMy2,COMz2,'x','LineWidth',2);
end
```

```
function [COMx2, COMy2, COMz2, ZMPx2, ZMPy2, ZMPz2] = COMeZMP_Traj(L1, L2,
L3, L4, L5, L_pe, B, n_passos, DSP_ratio)

g = 9.8*1000; %gravidade
A = L1;   %é a distância do pé ao centro, tem que ser L1
%A= 20;
l = L2+L3+L4;
f = (1/(2*pi()))*sqrt(g/l);
T = 1/f; %o step time é a metade do período da caminhada
w = 2*pi()*f;

%B = 0.2*1000 %Ds
b = B*0.2;  %no artigo é recomendado

syms k
ksum = 1:15;
S = zeros(size(ksum));
j = 1;

%DSP_ratio não é usado, pois ele é um percentual sobre um período, e o N é
um fator do sinc, não diretamente correspondente a tempo
%N = length(ksum)+1;
N = 15;
num=n_passos; %num é uma medida de tempo, mas como o T é 1, então equivale
ao numero de passos;

Kdsp = DSP_ratio;  %double support ratio (trabalho do estágio) - esse pode
ser dado pelo N da outra função (mas ZMP só referência)
Td = T*Kdsp;
Tm = (T+Td)/2;

%----------------------------------------
n=n_passos;
i=1;
for o=0:1:n
    ti(i) = o*T;
    i=i+1;
    ti(i) = o*T+Td;
    i=i+1;
    ti(i) = o*T+Tm;
    i=i+1;
    ti(i) = o*T+T;
    i=i+1;
    ti(i) = o*T+T+Td;
    i=i+1;
end
if ti(i-1)>n*T+T
    ti(i-1)=n*T+T;
end

i
t=0:(ti(i-1)/(i-2)):ti(i-1)  %o i-2 por causa que começa em zero.
x2=0:length(t)-1;
tq1_generic2=0:0.5:length(t)-1;
p_t = pchip(x2,t,tq1_generic2);

%----------------------------------------

for u = 1:length(p_t)
    for i = 1:length(ksum)
```

```
        cx_part = sinc(k/N)*(((B-
(2*b))*(T^2)*(w^2))/((k*pi())*(((T^2)*(w^2))+((k^2)*(pi()^2)))))*(sin((2*pi
()*k*p_t(u))/T));
        %cx_part = (((B-
(2*b))*(T^2)*(w^2))/((k*pi())*(((T^2)*(w^2))+((k^2)*(pi()^2)))))*(sin((2*pi
()*k*p_t(u))/T));
        Scx = ((B/T)*(p_t(u)-(T/2)))+symsum(cx_part,k,1,ksum(i));
     if Scx < 0
        Scx = 0;
     elseif Scx > B*(num)
        Scx = B*(num);
     end
    end
    for i = 1:length(ksum)
        px_part = sinc(k/N)*(((B-
2*b)*(T^2)*(w^2))/((k*pi())*((T^2)*(w^2)+(k^2)*(pi()^2))))*(1+(((pi()^2)*(k
^2))/((w^2)*(T^2))))*(sin((2*pi()*k*p_t(u))/T));
        %px_part = (((B-
2*b)*(T^2)*(w^2))/((k*pi())*((T^2)*(w^2)+(k^2)*(pi()^2))))*(1+(((pi()^2)*(k
^2))/((w^2)*(T^2))))*(sin((2*pi()*k*p_t(u))/T));
        Spx = ((B/T)*(p_t(u)-(T/2)))+symsum(px_part,k,1,ksum(i));
     if Spx < 0
        Spx = 0;
     elseif Spx > B*(num)
        Spx = B*(num);
     end
    end

    for i = 1:length(ksum)
    cy = sinc(k/N)*((2*A*(T^2)*(w^2)*(1-
cos(k*pi())))/(k*pi()*((T^2)*(w^2)+(k^2)*(pi()^2))))*(sin((2*pi()*k*p_t(u))
/(2*T)));
    %cy = ((2*A*(T^2)*(w^2)*(1-
cos(k*pi())))/(k*pi()*((T^2)*(w^2)+(k^2)*(pi()^2))))*(sin((2*pi()*k*p_t(u))
/(2*T)));
    Scy = double(symsum(cy,k,1,ksum(i))+0);

    py = sinc(k/N)*((2*A*(T^2)*(w^2)*(1-
cos(k*pi())))/(k*pi()*((T^2)*(w^2)+(k^2)*(pi()^2))))*(1+(((pi()^2)*(k^2))/(
(w^2)*(T^2))))*(sin((2*pi()*k*p_t(u))/(2*T)));
    %py = ((2*A*(T^2)*(w^2)*(1-
cos(k*pi())))/(k*pi()*((T^2)*(w^2)+(k^2)*(pi()^2))))*(1+(((pi()^2)*(k^2))/(
(w^2)*(T^2))))*(sin((2*pi()*k*p_t(u))/(2*T)));
    Spy = double(symsum(py,k,1,ksum(i))+0);
    end

tempo(j) = p_t(u);
Sum_cX(j) = double(Scx);
Sum_pX(j) = double(Spx);
Sum_cY(j) = double(Scy);
Sum_pY(j) = double(Spy);
j = j + 1;
end
for i=1:length(Sum_cX)
    COMz(i) = L2+L3+L4+L5-12;
end
for i=1:length(Sum_pX)
    ZMPz(i) = 0;
end

x=0:length(p_t)-1
```

```
tq1_generic=0:0.1:length(p_t)-1

COMx2 = pchip(x,Sum_cX,tq1_generic);
COMy2 = pchip(x,Sum_cY,tq1_generic);
COMz2 = pchip(x,COMz,tq1_generic);

ZMPx2 = pchip(x,Sum_pX,tq1_generic);
ZMPy2 = pchip(x,Sum_pY,tq1_generic);
ZMPz2 = pchip(x,ZMPz,tq1_generic);
end
```

APÊNDICE F – CÓDIGO EM MATLAB DA FUNÇÃO DE GERAÇÃO DE TRAJETÓRIA PARA OS *END-EFFECTORS*

```
function [p_xa_dir, y_dir, p_za_dir, p_xa_esq, y_esq, p_za_esq] =
Cubic_3Traj_Impar(L1, L2, L3, L4, L5, L_pe, n_passos, B, Hao_coef,
DSP_ratio, qb, qf)

g = 9.8*1000;
A = L1;   %é a distância do pé ao centro, tem que ser L1
l = L2+L3+L4;
f = (1/(2*pi()))*sqrt(g/l);  %aqui a marcha são dois passos, talvez tenha
que mudar a freq, por causa do período
Tc = 1/f; %o step time é a metade do período da caminhada
Kdsp = DSP_ratio;  %double support ratio
Td = Tc*Kdsp;
Tm = (Tc+Td)/2;

Ds = B;
Lao = Ds*0.25;

laf = L_pe*0.55;    % do meio do pé ao dedão
lab = L_pe*0.45;     % do meio do pé ao calcanhar
lan = L5;    %do meio do pé ao tornozelo
Hao = (lan+(Hao_coef*100));
c = 1;

hgs = 0  %é o nível do solo, é sempre 0 no meu caso
hge = 0  %é o nível do solo, é sempre 0 no meu caso

n=n_passos;
i=1;
for k=0:1:n
    t(i) = k*Tc;
    i=i+1;
    t(i) = k*Tc+Td;
    i=i+1;
    t(i) = k*Tc+Tm;
    i=i+1;
    t(i) = (k+1)*Tc;
    i=i+1;
    t(i) = (k+1)*Tc+Td;
    i=i+1;
end
if t(i-1)>n*Tc+Tc
    t(i-1)=n*Tc+Tc;
end
%-------perna-direita-----------------
i=1;
for k=0:2:n
    if  (k)*Ds<(n)*Ds
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+laf*sind(qb)+lan*cosd(qb);
    i=i+1;
    za_dir(i) = Hao;
    i=i+1;
    za_dir(i) = hge+lab*sind(qf)+lan*cosd(qf);
    i=i+1;
```

```
    za_dir(i) = hge+lan;
    i=i+6;
    else
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+lan;
    i=i+6;
    end
end
i=6;
for k=1:2:n
    za_dir(i) = za_dir(5*k);
    i=i+1;
    za_dir(i) = za_dir(5*k);
    i=i+1;
    za_dir(i) = za_dir(5*k);
    i=i+1;
    za_dir(i) = za_dir(5*k);
    i=i+1;
    za_dir(i) = za_dir(5*k);
    i=i+6;
end
%----------------Perna-Esquerda-z-----------------
i=1;
for k=0:2:n
    if  (k)*Ds<(n)*Ds
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+6;
    else
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+6;
    end
end
i=6;
for k=1:2:n
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+laf*sind(qb)+lan*cosd(qb);
    i=i+1;
```

```
    za_esq(i) = Hao;
    i=i+1;
    za_esq(i) = hge+lab*sind(qf)+lan*cosd(qf);
    i=i+1;
    za_esq(i) = hge+lan;
    i=i+6;
end
%----------------Perna-Direita-x-----------------

i=1;
for k=0:2:n
    if  (k)*Ds<(n)*Ds
    xa_dir(i) = k*Ds - Ds;
    if  xa_dir(i)<0
        xa_dir(i)=0;
    end
    i=i+1;
    xa_dir(i) = k*Ds+ lan*sind(qb)+laf*(1-cosd(qb))- Ds;
    if  xa_dir(i)<0
        xa_dir(i)=0;
    end
    i=i+1;
    xa_dir(i) = k*Ds+Lao-Ds;
    if  xa_dir(i)<0
        xa_dir(i)=k*Ds+Lao;
    end
    i=i+1;
    xa_dir(i) = (k+2)*Ds-lan*sind(qf)-lab*(1-cosd(qf))- Ds;
    i=i+1;
    xa_dir(i) = (k+2)*Ds - Ds;
    i=i+6;
    else
    xa_dir(i) = (n)*Ds;
    i=i+1;
    xa_dir(i) = (n)*Ds;
    i=i+1;
    xa_dir(i) = (n)*Ds;
    i=i+1;
    xa_dir(i) = (n)*Ds;
    i=i+1;
    xa_dir(i) = (n)*Ds;
    i=i+6;
    end
end
i=6;
for k=1:2:n
    xa_dir(i) = xa_dir(5*k);
    i=i+1;
    xa_dir(i) = xa_dir(5*k);
    i=i+1;
    xa_dir(i) = xa_dir(5*k);
    i=i+1;
    xa_dir(i) = xa_dir(5*k);
    i=i+1;
    xa_dir(i) = xa_dir(5*k);
    i=i+6;
end
i=1;

%----------------Perna-Esquerda-x-----------------
for k=0:2:n
```

```
    if  (k)*Ds<(n)*Ds
    xa_esq(i) = (k)*Ds;
    i=i+1;
    xa_esq(i) = (k)*Ds;
    i=i+1;
    xa_esq(i) = (k)*Ds;
    i=i+1;
    xa_esq(i) = (k)*Ds;
    i=i+1;
    xa_esq(i) = (k)*Ds;
    i=i+6;
    else
    xa_esq(i) = (n)*Ds;
    i=i+1;
    xa_esq(i) = (n)*Ds;
    i=i+1;
    xa_esq(i) = (n)*Ds;
    i=i+1;
    xa_esq(i) = (n)*Ds;
    i=i+1;
    xa_esq(i) = (n)*Ds;
    i=i+6;
    end
end
i=6;
for k=1:2:n
    xa_esq(i) = k*Ds - Ds;
    if  xa_esq(i)>(n)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+1;
    xa_esq(i) = k*Ds+ lan*sind(qb)+laf*(1-cosd(qb)) -Ds;
    if  xa_esq(i)>(n+1)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+1;
    xa_esq(i) = k*Ds+Lao - Ds;
    if  xa_esq(i)>(n)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+1;
    xa_esq(i) = (k+2)*Ds-lan*sind(qf)-lab*(1-cosd(qf))- Ds;
    if  xa_esq(i)>(n)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+1;
    xa_esq(i) = (k+2)*Ds - Ds;
    if  xa_esq(i)>(n)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+6;
end
xa_esq
x=0:length(t)-1
tq1_generic=0:0.05:length(t)-1;
p_t = pchip(x,t,tq1_generic)
t
length(x)
length(tq1_generic)
rate=(length(t)-1)/t(length(t));
```

```
p_za_dir = pchip(x,za_dir,p_t*rate);
p_za_esq = pchip(x,za_esq,p_t*rate);
p_xa_dir = pchip(x,xa_dir,p_t*rate);
p_xa_esq = pchip(x,xa_esq,p_t*rate);

for i=1:length(tq1_generic)
    y_dir(i)=-L1;
end

%scatter3(p_xa_dir,y_dir,p_za_dir,'o','LineWidth',2);
%legend('Tornozelo - Direito');
%hold on;

for i=1:length(tq1_generic)
    y_esq(i)=L1;
end
%scatter3(p_xa_esq,y_esq,p_za_esq,'o','LineWidth',2);
%legend('Tornozelo - Esquerdo');
%plot(t,za_dir,'o',p_t,p_za_dir,'-',p_t,p_za_esq,'-.')
%legend('Sample Points','pchip Direita','pchip
Esquerda','Location','SouthEast')
end


%-------------------------------------------------------------------------
function [p_xa_dir, y_dir, p_za_dir, p_xa_esq, y_esq, p_za_esq] =
Cubic_3Traj_Par(L1, L2, L3, L4, L5, L_pe, n_passos, B, Hao_coef, DSP_ratio,
qb, qf)

g = 9.8*1000;
A = L1;   %é a distância do pé ao centro, tem que ser L1
l = L2+L3+L4;
f = (1/(2*pi()))*sqrt(g/l);
Tc = 1/f;
Kdsp = DSP_ratio;  %double support ratio
Td = Tc*Kdsp;
Tm = (Tc+Td)/2;

Ds = B;
Lao = Ds*0.25;

laf = L_pe*0.55;    % meio do pé ao dedão
lab = L_pe*0.45;     % meio do pé ao calcanhar
lan = L5;    %do meio do pé ao tornozelo
Hao = (lan+(Hao_coef*100));
c = 1;

hgs = 0  %é o nível do solo, é sempre 0 nesse caso
hge = 0  %é o nível do solo, é sempre 0 nesse caso

n=n_passos;
i=1;
for k=0:1:n
    t(i) = k*Tc;
    i=i+1;
    t(i) = k*Tc+Td;
    i=i+1;
    t(i) = k*Tc+Tm;
    i=i+1;
    t(i) = (k+1)*Tc;
```

```
    i=i+1;
    t(i) = (k+1)*Tc+Td;
    i=i+1;
end
if t(i-1)>n*Tc+Tc
    t(i-1)=n*Tc+Tc;
end
%-------perna-direita-z-----------------
i=1;
for k=0:2:n
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+laf*sind(qb)+lan*cosd(qb);
    i=i+1;
    za_dir(i) = Hao;
    i=i+1;
    za_dir(i) = hge+lab*sind(qf)+lan*cosd(qf);
    i=i+1;
    za_dir(i) = hge+lan;
    i=i+6;
end
i=6;
for k=1:2:n
    if  (k)*Ds<(n)*Ds
    za_dir(i) = za_dir(5*k);
    i=i+1;
    za_dir(i) = za_dir(5*k);
    i=i+1;
    za_dir(i) = za_dir(5*k);
    i=i+1;
    za_dir(i) = za_dir(5*k);
    i=i+1;
    za_dir(i) = za_dir(5*k);
    i=i+6;
    else
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+lan;
    i=i+1;
    za_dir(i) = hgs+lan;
    i=i+6;
    end
end
%----------------Perna-Esquerda-z-----------------
i=1;
for k=0:2:n
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+6;
end
```

```
i=6;
for k=1:2:n
    if  (k)*Ds<(n)*Ds
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+laf*sind(qb)+lan*cosd(qb);
    i=i+1;
    za_esq(i) = Hao;
    i=i+1;
    za_esq(i) = hge+lab*sind(qf)+lan*cosd(qf);
    i=i+1;
    za_esq(i) = hge+lan;
    i=i+6;
    else
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+1;
    za_esq(i) = hgs+lan;
    i=i+6;
    end
end
%----------------Perna-Direita-x-----------------

i=1;
for k=0:2:n
    if (k)*Ds<(n)*Ds
    xa_dir(i) = k*Ds - Ds;
    if  xa_dir(i)<0
        xa_dir(i)=0;
    end
    i=i+1;
    xa_dir(i) = k*Ds+ lan*sind(qb)+laf*(1-cosd(qb))- Ds;
    if  xa_dir(i)<0
        xa_dir(i)=0;
    end
    i=i+1;
    xa_dir(i) = k*Ds+Lao-Ds;
    if  xa_dir(i)<0
        xa_dir(i)=k*Ds+Lao;
    end
    i=i+1;
    xa_dir(i) = (k+2)*Ds-lan*sind(qf)-lab*(1-cosd(qf))- Ds;
    i=i+1;
    xa_dir(i) = (k+2)*Ds - Ds;
    i=i+6;
    else
    xa_dir(i) = k*Ds - Ds;
    if  xa_dir(i)<0
        xa_dir(i)=0;
    end
    i=i+1;
    xa_dir(i) = k*Ds+ lan*sind(qb)+laf*(1-cosd(qb))- Ds;
    if  xa_dir(i)<0
        xa_dir(i)=0;
    end
    i=i+1;
```

```
    xa_dir(i) = k*Ds+Lao-Ds;
    if  xa_dir(i)<0
        xa_dir(i)=k*Ds+Lao;
    end
    i=i+1;
    xa_dir(i) = (k+2)*Ds-lan*sind(qf)-lab*(1-cosd(qf))- 2*Ds;
    i=i+1;
    xa_dir(i) = (k+2)*Ds - 2*Ds;
    i=i+6;
    end
end
i=6;
for k=1:2:n
    if (k)*Ds<(n)*Ds
    xa_dir(i) = xa_dir(5*k);
    i=i+1;
    xa_dir(i) = xa_dir(5*k);
    i=i+1;
    xa_dir(i) = xa_dir(5*k);
    i=i+1;
    xa_dir(i) = xa_dir(5*k);
    i=i+1;
    xa_dir(i) = xa_dir(5*k);
    i=i+6;
    else
    xa_dir(i) = (n)*Ds;
    i=i+1;
    xa_dir(i) = (n)*Ds;
    i=i+1;
    xa_dir(i) = (n)*Ds;
    i=i+1;
    xa_dir(i) = (n)*Ds;
    i=i+1;
    xa_dir(i) = (n)*Ds;
    i=i+6;
    end
end
i=1;

%----------------Perna-Esquerda-x-----------------
for k=0:2:n
    xa_esq(i) = (k)*Ds;
    i=i+1;
    xa_esq(i) = (k)*Ds;
    i=i+1;
    xa_esq(i) = (k)*Ds;
    i=i+1;
    xa_esq(i) = (k)*Ds;
    i=i+1;
    xa_esq(i) = (k)*Ds;
    i=i+6;
end
i=6;
for k=1:2:n % último passo para ficar finalizado
    if (k)*Ds<(n)*Ds
    xa_esq(i) = k*Ds - Ds;
    if  xa_esq(i)>(n)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+1;
    xa_esq(i) = k*Ds+ lan*sind(qb)+laf*(1-cosd(qb)) -Ds;
```

```
    if  xa_esq(i)>(n+1)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+1;
    xa_esq(i) = k*Ds+Lao - Ds;
    if  xa_esq(i)>(n)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+1;
    xa_esq(i) = (k+2)*Ds-lan*sind(qf)-lab*(1-cosd(qf))- Ds;
    if  xa_esq(i)>(n)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+1;
    xa_esq(i) = (k+2)*Ds - Ds;
    if  xa_esq(i)>(n)*Ds
        xa_esq(i)=(n)*Ds;
    end
    i=i+6;
    else
    xa_esq(i) = (n)*Ds;
    i=i+1;
    xa_esq(i) = (n)*Ds;
    i=i+1;
    xa_esq(i) = (n)*Ds;
    i=i+1;
    xa_esq(i) = (n)*Ds;
    i=i+1;
    xa_esq(i) = (n)*Ds;
    i=i+6;
    end
end
xa_esq
x=0:length(t)-1
tq1_generic=0:0.05:length(t)-1;
p_t = pchip(x,t,tq1_generic);
t;
t(length(t));
length(t)-1;
rate=(length(t)-1)/t(length(t));

p_za_dir = pchip(x,za_dir,p_t*rate);
p_za_esq = pchip(x,za_esq,p_t*rate);
p_xa_dir = pchip(x,xa_dir,p_t*rate);
p_xa_esq = pchip(x,xa_esq,p_t*rate);

for i=1:length(tq1_generic)
    y_dir(i)=-L1;
end

%scatter3(p_xa_dir,y_dir,p_za_dir,'o','LineWidth',2);

for i=1:length(tq1_generic)
    y_esq(i)=L1;
end
%scatter3(p_xa_esq,y_esq,p_za_esq,'o','LineWidth',2);
%plot(t,za_dir,'o',p_t,p_za_dir,'-',p_t,p_za_esq,'-.')
%legend('Sample Points','pchip Direita','pchip
Esquerda','Location','SouthEast')
end
```

APÊNDICE G – CÓDIGO EM MATLAB – MAIN E CINEMÁTICA INVERSA

```
%Augusto Poletto Cutulli
%Trabalho de Conclusão de Curso
%Main
clear
clc
%-----Comprimentos-----------------------------------------------------
L1=51.01;
L2=91.40;
L3=92.30;
L4=79.86;
L5=77.80;
L_pe=108.02;
%---------------------------------------------------------------------------
%----------------------------TRAJETÓRIA-------------------------------------
%---------------------------------------------------------------------------
n_passos=3;  %Passos completos!
B = 0.05*1000;  %Largura do Passo em mm
Hao_coef = 0.2;
DSP_ratio = 0.05; %Percentual de Double Step Phase (KHR2 informa que
utilizou 5%
%ZMP-COM---------------------------------------------------------------------
tic
[COMx, COMy, COMz] = COM_Traj(L1, L2, L3, L4, L5, L_pe, B, n_passos,
DSP_ratio);
tempo_CoM = toc
tic
%[COMx, COMy, COMz, ZMPx, ZMPy, ZMPz] = COMeZMP_Traj(L1, L2, L3, L4, L5,
L_pe, B, n_passos, DSP_ratio);
tempo_ZMP = toc
%-Pontos-Tornozelo-----------------------------------------------------------
qb = 0;  %angulo inicial do pé - 0 é pé flat
qf = 0;  %angulo final do pé - 0 é pé flat
tic
if mod(n_passos,2) == 1
[X_dir, Y_dir, Z_dir, X_esq, Y_esq, Z_esq]  = Cubic_3Traj_Impar(L1, L2, L3,
L4, L5, L_pe, n_passos, B, Hao_coef, DSP_ratio, qb, qf);
else
[X_dir, Y_dir, Z_dir, X_esq, Y_esq, Z_esq]  = Cubic_3Traj_Par(L1, L2, L3,
L4, L5, L_pe, n_passos, B, Hao_coef, DSP_ratio, qb, qf);
end
tempo_TrajPCHIP = toc
%legend('CoM - Pélvis','Tornozelo - Direito','Tornozelo - Esquerdo');
%legend('CoM - Pélvis','ZMP - Ref Pés','Tornozelo - Direito','Tornozelo -
Esquerdo');
%xlabel('X [mm]');
%ylabel('Y [mm]');
%zlabel('Z [mm]');
%---------------------------------------------------------------------------
%-------------------------CINEMATICA-INVERSA--------------------------------
%---------------------------------------------------------------------------
lambda = 300;
I = eye(3);
contador = 0;
d_traj_esq_x = X_esq;
d_traj_esq_y = Y_esq;
d_traj_esq_z = Z_esq;
```

```
d_traj_COM_x = COMx;
d_traj_COM_y = COMy;
d_traj_COM_z = COMz;
FK_APeBase_COM = [d_traj_COM_x; d_traj_COM_y; d_traj_COM_z];

traj_spline_x = X_dir;
traj_spline_y = Y_dir;
traj_spline_z = Z_dir;

%-----DH-Parameters------------------------------------------------------
%-----ângulos-iniciais--------------------------------------------------
Theta_DH_junto_dir = [0 -90 -20 40 -20 0];
Theta_DH_junto_esq = [0 -90 -20 40 -20 0];
Theta_DH_dir_perna = [0 -90 -20 40];
Theta_DH_esq_perna = [0 -90 -20 40];
Theta_DH_dir_pe = [-20 0];
Theta_DH_esq_pe = [-20 0];

Alpha_DH = [90 -90 0 0 90 0];
a_DH = [0 0 L3 L4 0 L5];
d_DH = [0 0 0 0 0 0];

delta_theta_dir = [0; 0; 0; 0; 0; 0];
delta_theta_esq = [0; 0; 0; 0; 0; 0];

%-------------------------ITERAÇÕES-------------------------------------
i = 1;
tic
n_traj = length(d_traj_esq_x)-1;
F(n_traj) = struct('cdata',[],'colormap',[]);
while(i <= n_traj)
%-----Tranformações-----------------------------------------------------
    [T_A_Base_esq, T_A_Base0_esq, T_A_Base1_esq, T_A_Base2_esq,
     T_A_Base3_esq, T_A_Base4_esq, T_A_Base5_esq, T_A_Base6_esq] =
     T_A(FK_APeBase_COM(:,i), Theta_DH_junto_esq, Alpha_DH, a_DH, d_DH, L1,
     L2, L5);
     [T_A_Base_dir, T_A_Base0_dir, T_A_Base1_dir, T_A_Base2_dir,
     T_A_Base3_dir, T_A_Base4_dir, T_A_Base5_dir, T_A_Base6_dir] =
     T_A(FK_APeBase_COM(:,i), Theta_DH_junto_dir, Alpha_DH, a_DH, d_DH, -
     L1, L2, L5);

%-----Forward-Kinematics-----------------------------------------------
      FK_A_Base4_esq = [T_A_Base4_esq(1,4); T_A_Base4_esq(2,4);
      T_A_Base4_esq(3,4)]; %---posição-end-effector_esq-tornozelo
      FK_A_Base4_dir = [T_A_Base4_dir(1,4); T_A_Base4_dir(2,4);
      T_A_Base4_dir(3,4)]; %---posição-end-effector_dir-tornozelo

      FK_A_Base6_esq = [T_A_Base6_esq(1,4); T_A_Base6_esq(2,4);
      T_A_Base6_esq(3,4)]; %---posição-end-effector_esq-pé
      FK_A_Base6_dir = [T_A_Base6_dir(1,4); T_A_Base6_dir(2,4);
      T_A_Base6_dir(3,4)]; %---posição-end-effector_dir-pé

%--------------------Deltas_X--------------------------------------------
     delta_X_dir = [traj_spline_x(i); traj_spline_y(i); traj_spline_z(i)] -
     FK_A_Base4_dir;   %cuidar notação ponto e vetor: rever.
     delta_X_esq = [d_traj_esq_x(i); d_traj_esq_y(i); d_traj_esq_z(i)] -
     FK_A_Base4_esq;   %cuidar notação ponto e vetor: rever.

     ErroPos_delta_X_dir(i) = norm(delta_X_dir);
     ErroPos_delta_X_esq(i) = norm(delta_X_esq);
```

```
     delta_X_dir_pe = [traj_spline_x(i); traj_spline_y(i);
     traj_spline_z(i)-L5] - FK_A_Base6_dir;

     delta_X_esq_pe = [d_traj_esq_x(i); d_traj_esq_y(i); d_traj_esq_z(i)-
     L5] - FK_A_Base6_esq;
     ErroPos_delta_X_dir_pe(i) = norm(delta_X_dir_pe);
     ErroPos_delta_X_esq_pe(i) = norm(delta_X_esq_pe);

%-----Jacobian-----------------------------------------------------------
    J06_dir = Jacobian2(FK_APeBase_COM(:,i), T_A_Base1_dir, T_A_Base2_dir,
    T_A_Base3_dir, T_A_Base4_dir);
    J06_esq = Jacobian2(FK_APeBase_COM(:,i), T_A_Base1_esq, T_A_Base2_esq,
    T_A_Base3_esq, T_A_Base4_esq);
    h_esq_perna(i) = sqrt(det(J06_esq*transpose(J06_esq)));
    h_dir_perna(i) = sqrt(det(J06_dir*transpose(J06_dir)));

%----- J+ --- Damped Least Square ---------------------------------------
    J06_esq_plus = transpose(J06_esq)*inv(((J06_esq*transpose(J06_esq)) +
    (lambda^2)*I));
    delta_theta_esq = J06_esq_plus*(delta_X_esq);
    Theta_DH_esq_perna = Theta_DH_esq_perna + transpose(delta_theta_esq);
    %incrementa os ângulos


    J_dir_plus = transpose(J06_dir)*inv(((J06_dir*transpose(J06_dir)) +
    (lambda^2)*I));
    delta_theta_dir = J_dir_plus*(delta_X_dir); %coloquei um transpose no
    (Theta_DH_esq) pra poder calcular
    Theta_DH_dir_perna = Theta_DH_dir_perna + transpose(delta_theta_dir);
    %incrementa os ângulos

%-----Jacobian-----------------------------------------------------------
   J06_dir_pe = Jacobian_pe(T_A_Base4_dir, T_A_Base5_dir, T_A_Base6_dir,
    L5);
   J06_esq_pe = Jacobian_pe(T_A_Base4_esq, T_A_Base5_esq, T_A_Base6_esq,
    L5);
   h_esq_pe(i) = sqrt(det(J06_esq_pe*transpose(J06_esq_pe)));
   h_dir_pe(i) = sqrt(det(J06_dir_pe*transpose(J06_dir_pe)));
%----- J# ---DLS - Damped Least Square ----------------------------------
    J06_esq_plus_pe =
    transpose(J06_esq_pe)*inv(((J06_esq_pe*transpose(J06_esq_pe)) +
    (lambda^2)*I));
    delta_theta_esq_pe = J06_esq_plus_pe*(delta_X_esq_pe);
    Theta_DH_esq_pe = Theta_DH_esq_pe + transpose(delta_theta_esq_pe);
    %incrementa os ângulos

    J_dir_plus_pe =
    transpose(J06_dir_pe)*inv(((J06_dir_pe*transpose(J06_dir_pe)) +
    (lambda^2)*I));
    delta_theta_dir_pe = J_dir_plus_pe*(delta_X_dir_pe);
    Theta_DH_dir_pe = Theta_DH_dir_pe + transpose(delta_theta_dir_pe);
    %incrementa os ângulos

%--------Limites-----------------------------------------------------------
   % Theta_DH_dir_perna(1) = 0;
   % Theta_DH_esq_perna(1) = 0;
%-----------------Verifica-Erro----------------------------
%-||(J*delta_theta)-delta_X|| Deve ser menor que Epslon
    Ep = 0.001;
```

```
    module_dtheta_dir = norm(delta_theta_dir);
    module_dtheta_esq = norm(delta_theta_esq);
    module_dtheta_dir_pe = norm(delta_theta_dir_pe);
    module_dtheta_esq_pe = norm(delta_theta_esq_pe);
    if (module_dtheta_dir < Ep && module_dtheta_esq < Ep &&
    module_dtheta_dir_pe < Ep && module_dtheta_esq_pe < Ep )
        % (true)
    Theta_DH_junto_dir =  [Theta_DH_dir_perna Theta_DH_dir_pe];
    Theta_DH_junto_esq =  [Theta_DH_esq_perna Theta_DH_esq_pe];
    ThetaDHs(i,:) = Theta_DH_junto_dir;
    ThetaDHf(i,:) = Theta_DH_junto_esq;
%--------Desenha-Quadros-Robo-----------------------------------------------
      % Desenha_Robo(L1, L2, L5, FK_APeBase_COM(:,i), Theta_DH_junto_dir,
Theta_DH_junto_esq, Alpha_DH, a_DH, d_DH, n_traj, traj_spline_x,
traj_spline_y, traj_spline_z);
      % grid on;
%--------Animação-----------------------------------------------------------
      % view([1,1,1]);
      % drawnow
      % F1(i) = getframe(gcf);
      % view([1,1,0]);
      % drawnow
      % F2(i) = getframe(gcf);
      % view([0,1,0]);
      % drawnow
      % F3(i) = getframe(gcf);
      % view([1,0,0]);
      % drawnow
      % F4(i) = getframe(gcf);
     i = i + 1
    end
    contador = contador + 1;
    Theta_DH_junto_dir =  [Theta_DH_dir_perna Theta_DH_dir_pe];
    Theta_DH_junto_esq =  [Theta_DH_esq_perna Theta_DH_esq_pe];
end
tempo=toc
 %ErroPos_delta_X_dir(i) = norm(delta_X_dir);
 %ErroPos_delta_X_esq(i) = norm(delta_X_esq)  ;
 %ErroPos_delta_X_dir_pe(i) = norm(delta_X_dir_pe);
 %ErroPos_delta_X_esq_pe(i) = norm(delta_X_esq_pe) ;

 %Media_erros(1) = mean(ErroPos_delta_X_dir);
 %Media_erros(2) = mean(ErroPos_delta_X_esq);
 %Media_erros(3) =  mean(ErroPos_delta_X_dir_pe);
 %Media_erros(4) =  mean(ErroPos_delta_X_esq_pe);
```

APÊNDICE H – CÓDIGO EM MATLAB DA FUNÇÃO DAS TRANSFORMAÇÕES

```
function [A_Base, A_Base0, A_Base1, A_Base2, A_Base3, A_Base4, A_Base5,
A_Base6] = T_A(COM, theta, alpha, a, d, L1, L2, L5)
%--------------
COM_X = COM(1,1);
COM_Y = COM(2,1);
COM_Z = COM(3,1);
%Base------------Para iniciar certo-------------------------------------
%para começar em -x ou -y tem que ajustar nos parâmetros iniciais +-L1
%Perna dir em x: vai andar para frente em Y
%A_Base=[1 0 0 COM_X; 0 1 0 COM_Y; 0 0 1 COM_Z; 0 0 0 1];
%Perna dir em y: vai andar para frente em X
A_Base=[0 -1 0 COM_X;1 0 0 COM_Y; 0 0 1 COM_Z; 0 0 0 1];
Base_Translacao = [1 0 0 L1; 0 1 0 0; 0 0 1 -L2; 0 0 0 1];
A_Base0=A_Base*Base_Translacao;

%Junta-1-----------------------------------------------------------------
ARtheta=[cosd(theta(1)) -sind(theta(1)) 0 0; sind(theta(1)) cosd(theta(1))
0 0; 0 0 1 0; 0 0 0 1];
ARalpha=[1 0 0 0; 0 cosd(alpha(1)) -sind(alpha(1)) 0; 0 sind(alpha(1))
cosd(alpha(1)) 0; 0 0 0 1];
Aa=[1 0 0 a(1); 0 1 0 0; 0 0 1 0; 0 0 0 1];
Ad=[1 0 0 0; 0 1 0 0; 0 0 1 d(1); 0 0 0 1];
A01=ARtheta*ARalpha*Aa*Ad;

%Junta-2-----------------------------------------------------------------
ARtheta=[cosd(theta(2)) -sind(theta(2)) 0 0; sind(theta(2)) cosd(theta(2))
0 0; 0 0 1 0; 0 0 0 1];
ARalpha=[1 0 0 0; 0 cosd(alpha(2)) -sind(alpha(2)) 0; 0 sind(alpha(2))
cosd(alpha(2)) 0; 0 0 0 1];
Aa=[1 0 0 a(2); 0 1 0 0; 0 0 1 0; 0 0 0 1];
Ad=[1 0 0 0; 0 1 0 0; 0 0 1 d(2); 0 0 0 1];
A12=ARtheta*ARalpha*Aa*Ad;

%Junta-3-----------------------------------------------------------------
ARtheta=[cosd(theta(3)) -sind(theta(3)) 0 0; sind(theta(3)) cosd(theta(3))
0 0; 0 0 1 0; 0 0 0 1];
ARalpha=[1 0 0 0; 0 cosd(alpha(3)) -sind(alpha(3)) 0; 0 sind(alpha(3))
cosd(alpha(3)) 0; 0 0 0 1];
Aa=[1 0 0 a(3); 0 1 0 0; 0 0 1 0; 0 0 0 1];
Ad=[1 0 0 0; 0 1 0 0; 0 0 1 d(3); 0 0 0 1];
A23=ARtheta*ARalpha*Aa*Ad;

%Junta-4-----------------------------------------------------------------
ARtheta=[cosd(theta(4)) -sind(theta(4)) 0 0; sind(theta(4)) cosd(theta(4))
0 0; 0 0 1 0; 0 0 0 1];
ARalpha=[1 0 0 0; 0 cosd(alpha(4)) -sind(alpha(4)) 0; 0 sind(alpha(4))
cosd(alpha(4)) 0; 0 0 0 1];
Aa=[1 0 0 a(4); 0 1 0 0; 0 0 1 0; 0 0 0 1];
Ad=[1 0 0 0; 0 1 0 0; 0 0 1 d(4); 0 0 0 1];
A34=ARtheta*ARalpha*Aa*Ad;

%Junta-5-----------------------------------------------------------------
ARtheta=[cosd(theta(5)) -sind(theta(5)) 0 0; sind(theta(5)) cosd(theta(5))
0 0; 0 0 1 0; 0 0 0 1];
ARalpha=[1 0 0 0; 0 cosd(alpha(5)) -sind(alpha(5)) 0; 0 sind(alpha(5))
cosd(alpha(5)) 0; 0 0 0 1];
Aa=[1 0 0 a(5); 0 1 0 0; 0 0 1 0; 0 0 0 1];
```

```
Ad=[1 0 0 0; 0 1 0 0; 0 0 1 d(5); 0 0 0 1];
A45=ARtheta*ARalpha*Aa*Ad;

%Junta-6-------------------------------------------------------------------
ARtheta=[cosd(theta(6)) -sind(theta(6)) 0 0; sind(theta(6)) cosd(theta(6))
0 0; 0 0 1 0; 0 0 0 1];
ARalpha=[1 0 0 0; 0 cosd(alpha(6)) -sind(alpha(6)) 0; 0 sind(alpha(6))
cosd(alpha(6)) 0; 0 0 0 1];
Aa=[1 0 0 a(6); 0 1 0 0; 0 0 1 0; 0 0 0 1];
Ad=[1 0 0 0; 0 1 0 0; 0 0 1 d(6); 0 0 0 1];
A56=ARtheta*ARalpha*Aa*Ad;

%Transformada-Direta--------------
A_Base1 = A_Base0*A01;
A_Base2 = A_Base0*A01*A12;
A_Base3 = A_Base0*A01*A12*A23;
A_Base4 = A_Base0*A01*A12*A23*A34;
A_Base5 = A_Base0*A01*A12*A23*A34*A45;
A_Base6 = A_Base0*A01*A12*A23*A34*A45*A56;

end
```

APÊNDICE I – CÓDIGO EM MATLAB DA FUNÇÃO DAS JACOBIANAS

```
function J = Jacobian2(FK_APeBase_COM, T1, T2, T3, T4)
%-----Jacobian-Linear-Velocity - Jvi = Zi-1 x (On - Oi-1) -----------------
Jv1 = cross([0; 0; 1],[T4(1,4);T4(2,4);T4(3,4)]-
[FK_APeBase_COM(1,1);FK_APeBase_COM(2,1);FK_APeBase_COM(3,1)]);
Jv2 = cross([T1(1,3);T1(2,3);T1(3,3)],[T4(1,4);T4(2,4);T4(3,4)]-
[T1(1,4);T1(2,4);T1(3,4)]);
Jv3 = cross([T2(1,3);T2(2,3);T2(3,3)],[T4(1,4);T4(2,4);T4(3,4)]-
[T2(1,4);T2(2,4);T2(3,4)]);
Jv4 = cross([T3(1,3);T3(2,3);T3(3,3)],[T4(1,4);T4(2,4);T4(3,4)]-
[T3(1,4);T3(2,4);T3(3,4)]);

%-----Jacobian-Angular-Velocity-Jwi = Zi-----------------------------------
Jv = [Jv1 Jv2 Jv3 Jv4];
J = Jv;
End


function J = Jacobian_pe(T4, T5, T6, L5)
%-----Jacobian-Linear-Velocity - Jvi = Zi-1 x (On - Oi-1) ----------------
Jv5 = cross([T4(1,3);T4(2,3);T4(3,3)],[T6(1,4);T6(2,4);T6(3,4)]-
[T4(1,4);T4(2,4);T4(3,4)]);
Jv6 = cross([T5(1,3);T5(2,3);T5(3,3)],[T6(1,4);T6(2,4);T6(3,4)]-
[T5(1,4);T5(2,4);T5(3,4)]); % (On - Oi-1)

%-----Jacobian-Angular-Velocity-Jwi = Zi-----------------------------------
Jv = [Jv5 Jv6];
J = Jv;
end
```

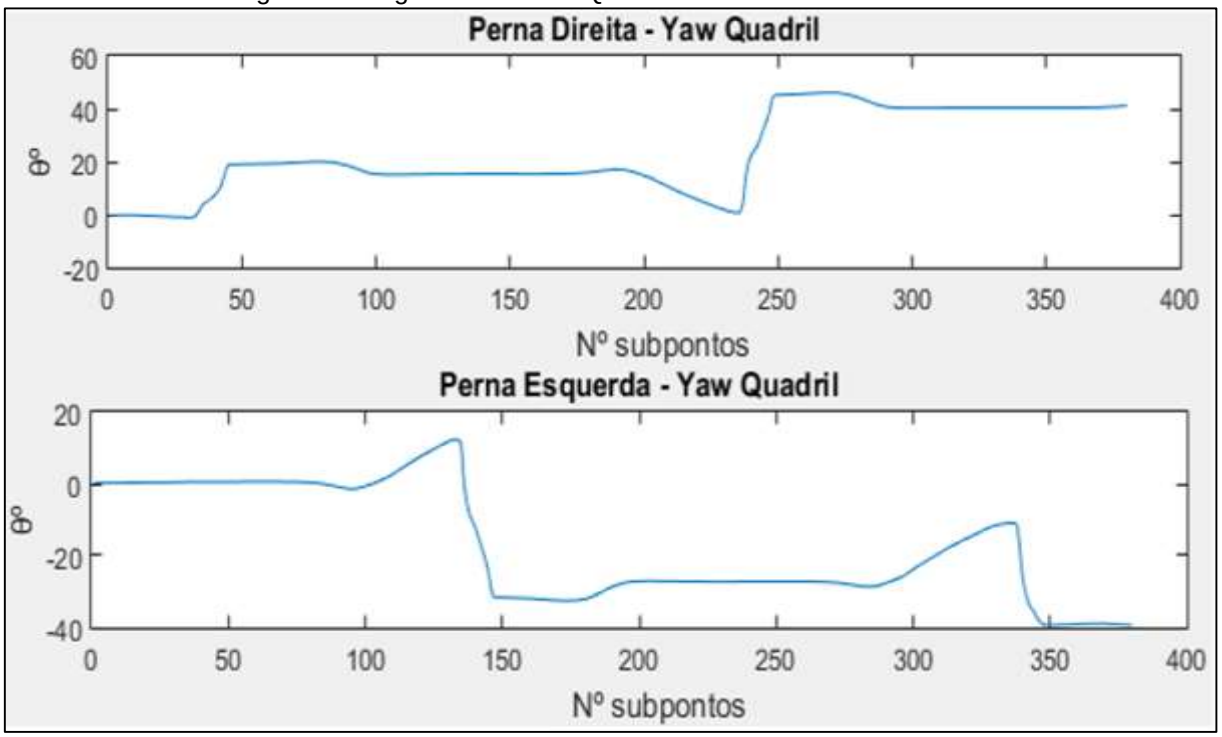
APÊNDICE J – CÓDIGO EM MATLAB DE PLOT DO ROBÔ BÍPEDE

```
function Desenha_Robo(L1, L2, L5, FK_APeBase_COM, Theta_DH_dir,
Theta_DH_esq, Alpha_DH, a_DH, d_DH, n_traj, d_traj_x, d_traj_y, d_traj_z)
%-----Transformações-------------
for i=1:n_traj
[A_Base_e, A_Base0_e, T_A_Base1_e, T_A_Base2_e, T_A_Base3_e, T_A_Base4_e,
T_A_Base5_e, T_A_Base6_e] = T_A(FK_APeBase_COM, Theta_DH_esq, Alpha_DH,
a_DH, d_DH, L1, L2, L5);
[A_Base, A_Base0, T_A_Base1_d, T_A_Base2_d, T_A_Base3_d, T_A_Base4_d,
T_A_Base5_d, T_A_Base6_d] = T_A(FK_APeBase_COM, Theta_DH_dir, Alpha_DH,
a_DH, d_DH, -L1, L2, L5);

FK_dir = [T_A_Base1_d T_A_Base2_d T_A_Base3_d T_A_Base4_d T_A_Base5_d
T_A_Base6_d];
FK_esq = [T_A_Base6_e T_A_Base5_e T_A_Base4_e T_A_Base3_e T_A_Base2_e
T_A_Base1_e A_Base0_e A_Base_e];

%A perna esq começa no pé e termina em 0,0,35, a perna dir
%começa em 0,0,COM e termina no pé. Essa ordem é feita só pra ficar certo o
%desenho, o segundo ponto é metade base0 e metade base, pois é o ponto do
%quadril é para formar o desenho - começa da pélvis

Q1_esq=[FK_esq(1,4)  FK_esq(1,8)  FK_esq(1,12) FK_esq(1,16) FK_esq(1,20)
FK_esq(1,24)  FK_esq(1,28)  FK_esq(1,28)  FK_esq(1,32)];
Q2_esq=[FK_esq(2,4)  FK_esq(2,8)  FK_esq(2,12) FK_esq(2,16) FK_esq(2,20)
FK_esq(2,24)  FK_esq(2,28)  FK_esq(2,28)  FK_esq(2,32)];
Q3_esq=[FK_esq(3,4)  FK_esq(3,8)  FK_esq(3,12) FK_esq(3,16) FK_esq(3,20)
FK_esq(3,24)  FK_esq(3,28)  FK_esq(3,32)  FK_esq(3,32)];

Q_esq=[Q1_esq;Q2_esq;Q3_esq];

Q1_dir=[A_Base(1,4) A_Base0(1,4)  A_Base0(1,4)  FK_dir(1,4)  FK_dir(1,8)
FK_dir(1,12) FK_dir(1,16) FK_dir(1,20) FK_dir(1,24)];
Q2_dir=[A_Base(2,4) A_Base0(2,4)  A_Base0(2,4)  FK_dir(2,4)  FK_dir(2,8)
FK_dir(2,12) FK_dir(2,16) FK_dir(2,20) FK_dir(2,24)];
Q3_dir=[A_Base(3,4) A_Base(3,4)   A_Base0(3,4)  FK_dir(3,4)  FK_dir(3,8)
FK_dir(3,12) FK_dir(3,16) FK_dir(3,20) FK_dir(3,24)];

Q_dir=[Q1_dir;Q2_dir;Q3_dir];

Q(:,:,i) = [Q_esq Q_dir];
end

X = d_traj_x;
Y = d_traj_y;
Z = d_traj_z;
hold on;
grid on;

for i=1:n_traj
plot3(Q(1,:,i),Q(2,:,i),Q(3,:,i),'-
o','LineWidth',1,'MarkerSize',6,'MarkerFaceColor',[0.0,1.0,1.0]);
end

title('Robô 12DOF')
xlabel('X [mm]');
ylabel('Y [mm]');
```

```
zlabel('Z [mm]');
axis([-100 350 -150 150 -100 400]);
view([0,1,0]);
end
```

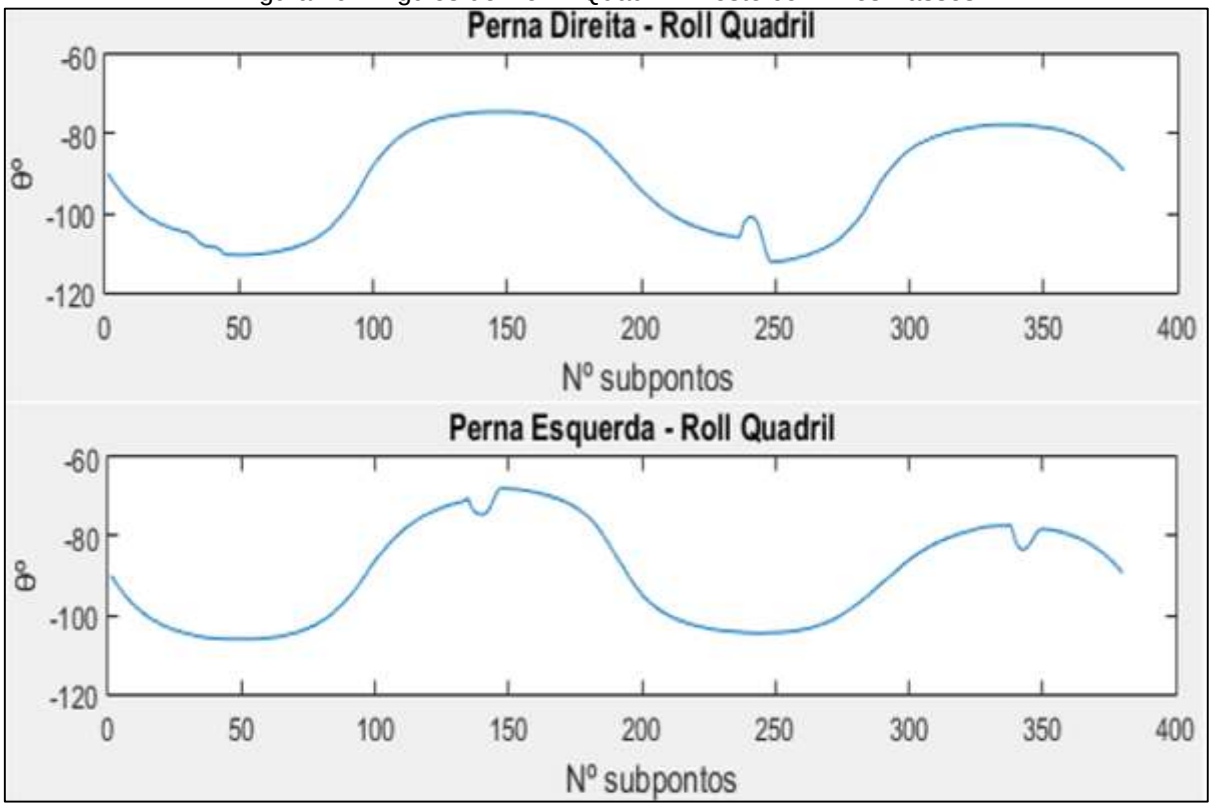
APÊNDICE K – ÂNGULOS OBTIDOS ATRAVÉS IMPLEMENTAÇÃO DOS MÉTODOS PARA UMA MARCHA COM TRÊS PASSOS

Figura 48: Ângulos de Yaw – Quadril – Teste com Três Passos



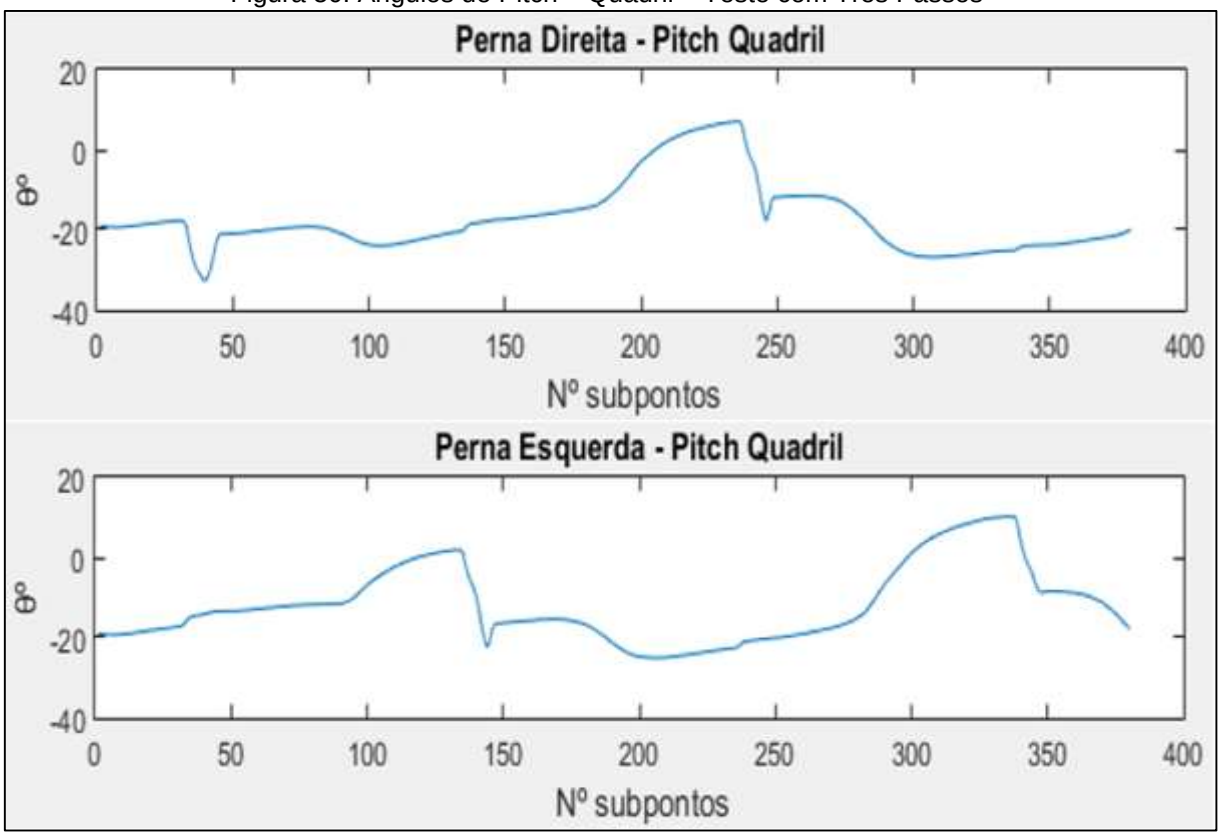
Fonte: O próprio autor (2017).

Figura 49: Ângulos de Roll – Quadril – Teste com Três Passos



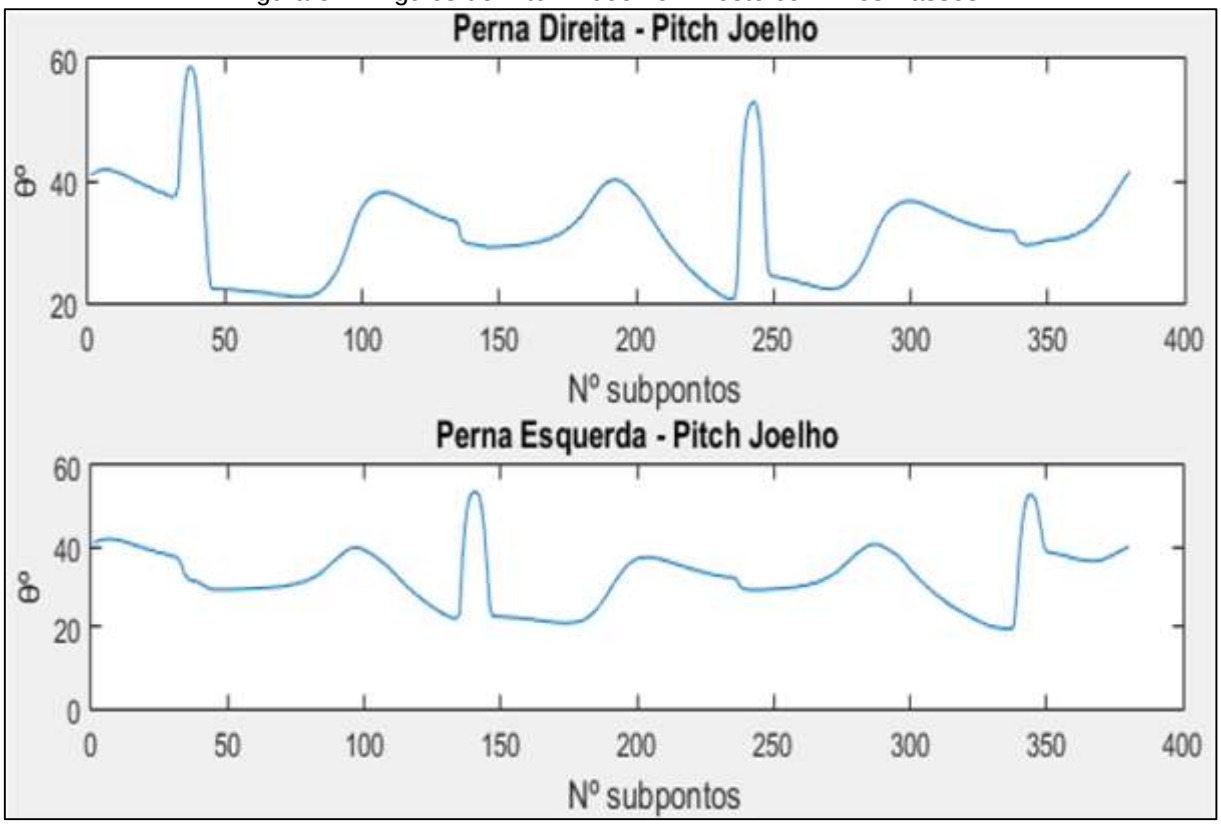
Fonte: O próprio autor (2017).

Figura 50: Ângulos de Pitch – Quadril – Teste com Três Passos



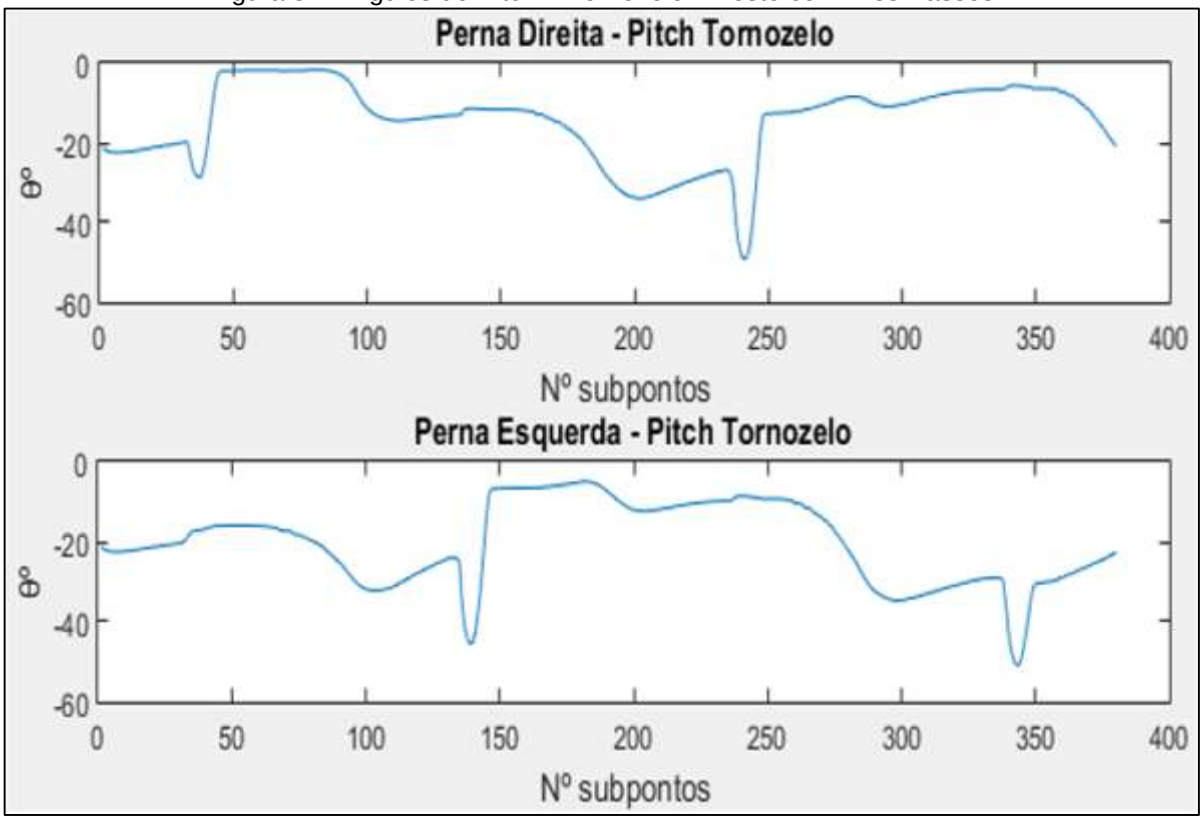
Fonte: O próprio autor (2017).

Figura 51: Ângulos de Pitch – Joelho – Teste com Três Passos



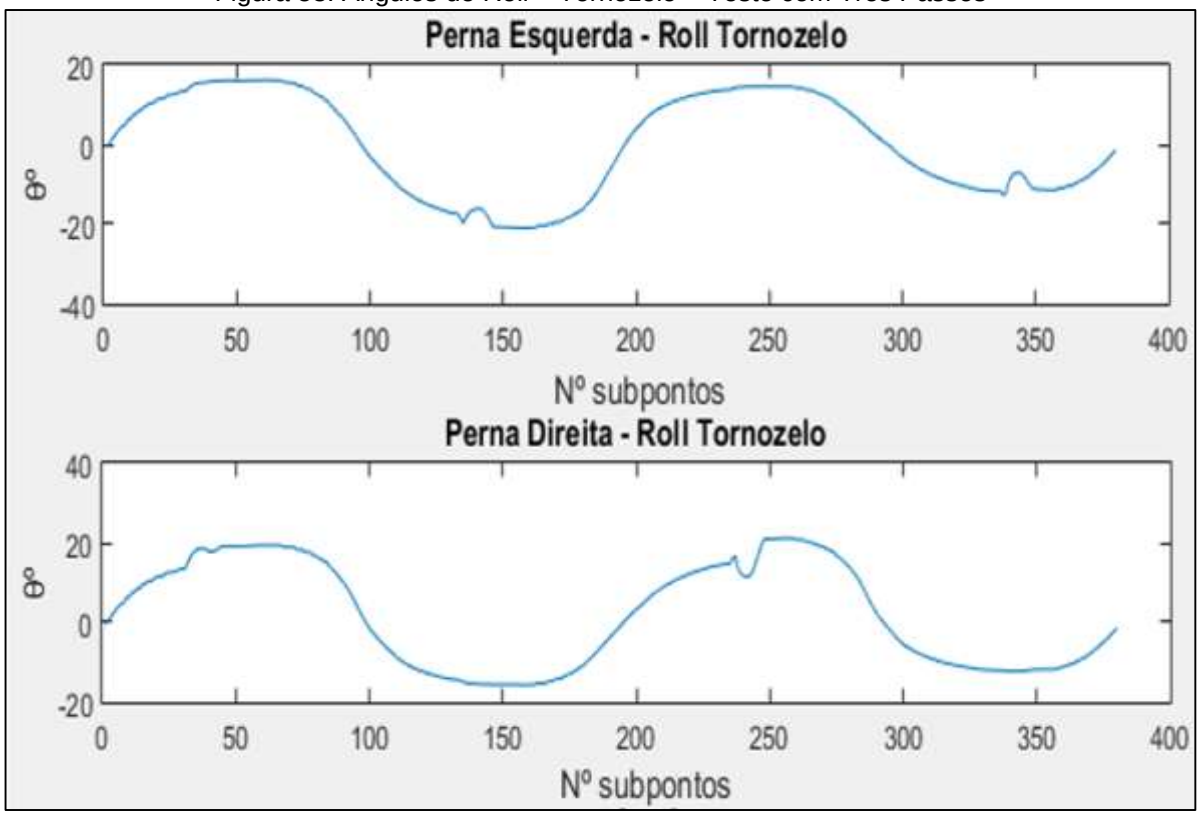
Fonte: O próprio autor (2017).

Figura 52: Ângulos de Pitch – Tornozelo – Teste com Três Passos



Fonte: O próprio autor (2017).

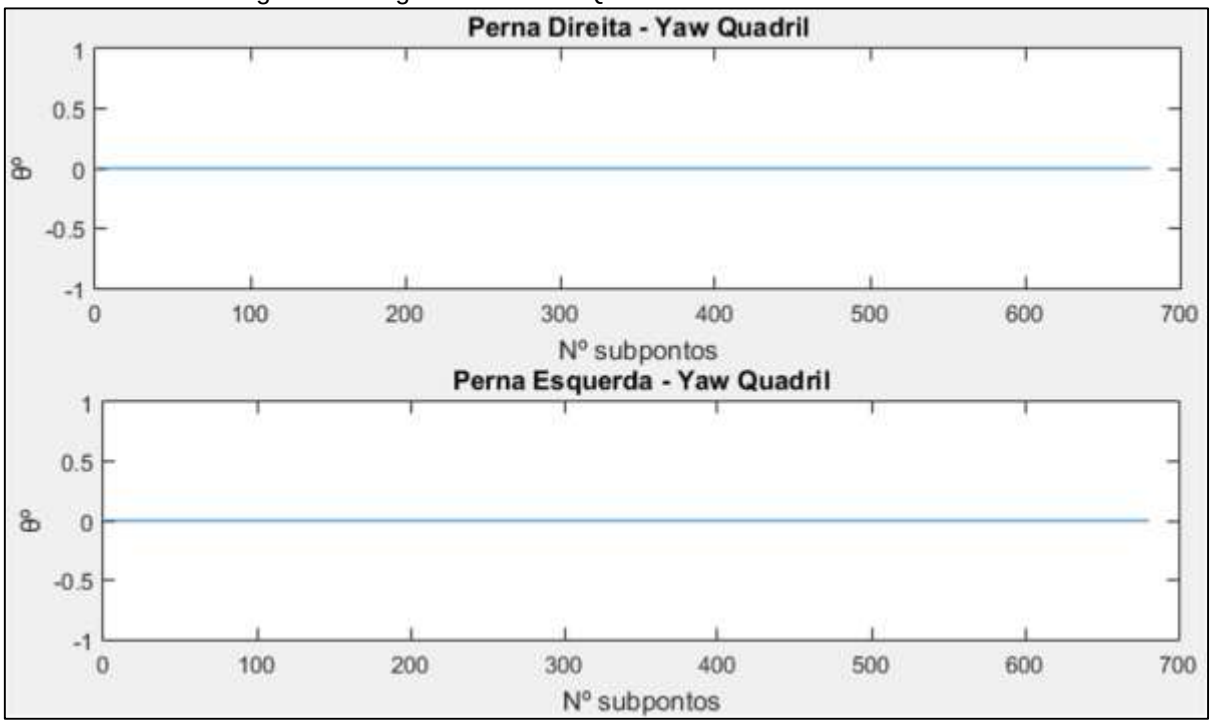
Figura 53: Ângulos de Roll – Tornozelo – Teste com Três Passos



Fonte: O próprio autor (2017).

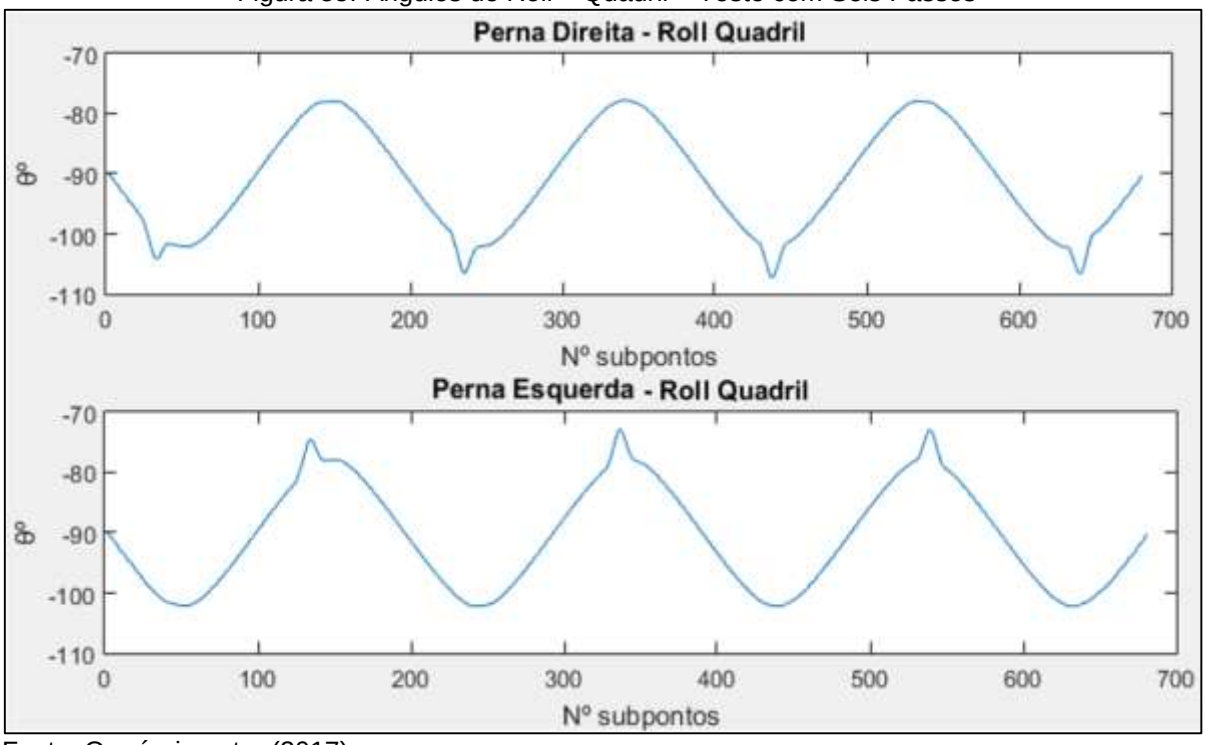
APÊNDICE L – ÂNGULOS OBTIDOS ATRAVÉS IMPLEMENTAÇÃO DOS MÉTODOS PARA UMA MARCHA COM SEIS PASSOS

Figura 54: Ângulos de Yaw – Quadril – Teste com Seis Passos



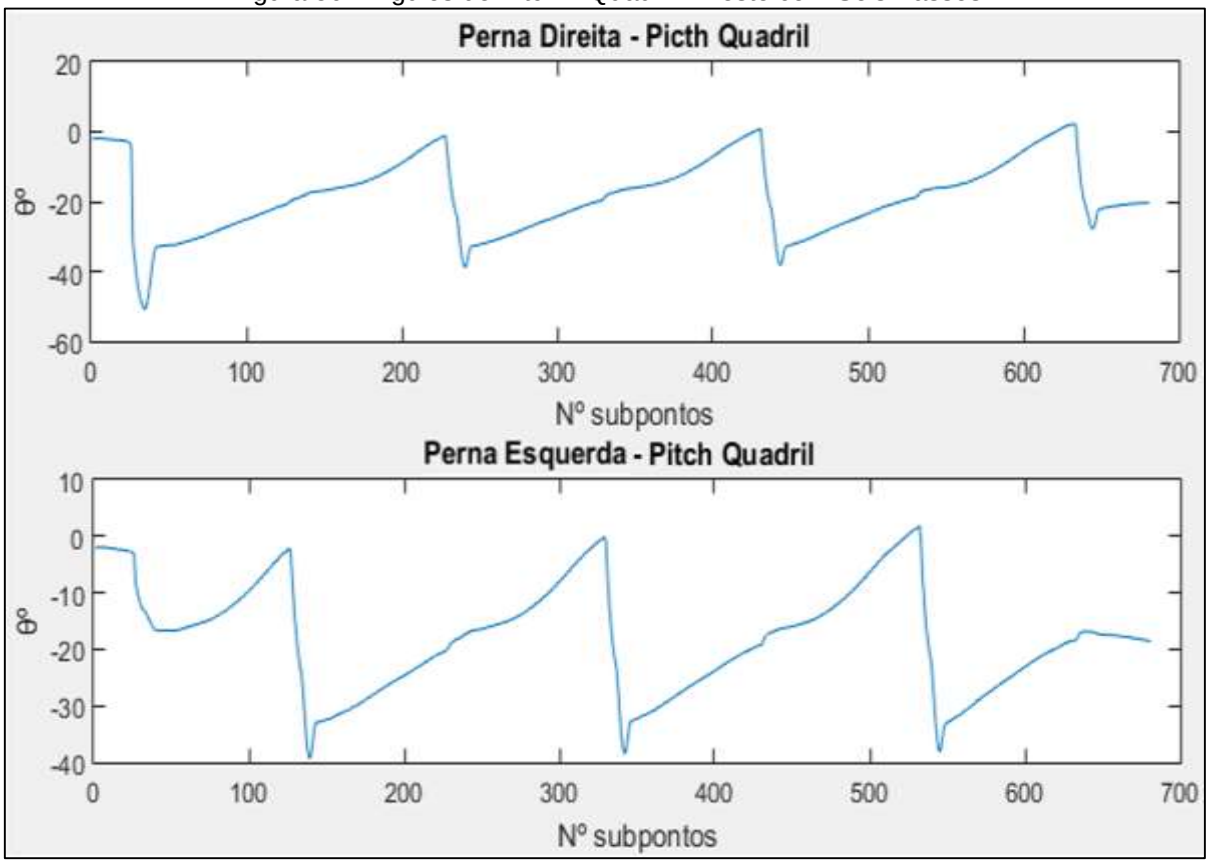
Fonte: O próprio autor (2017).

Figura 55: Ângulos de Roll – Quadril – Teste com Seis Passos



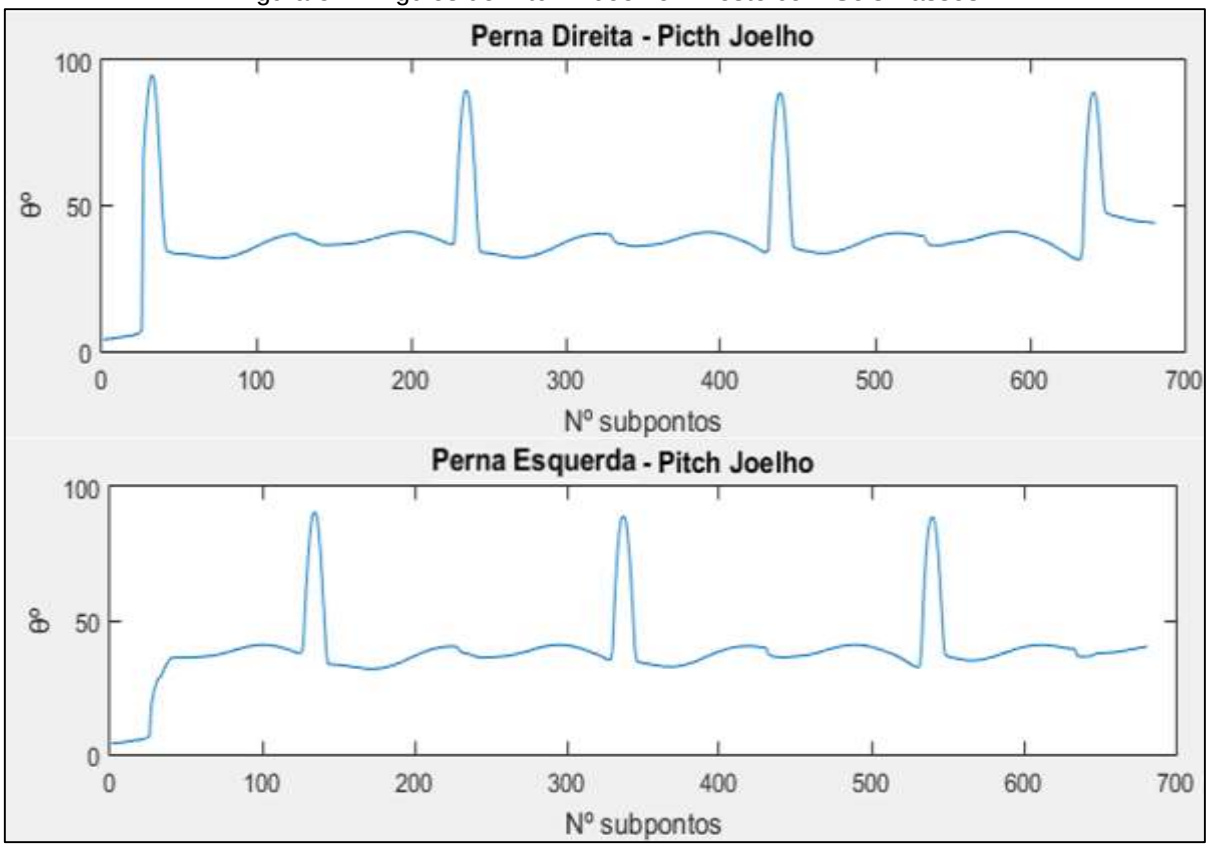
Fonte: O próprio autor (2017).

Figura 56: Ângulos de Pitch – Quadril – Teste com Seis Passos



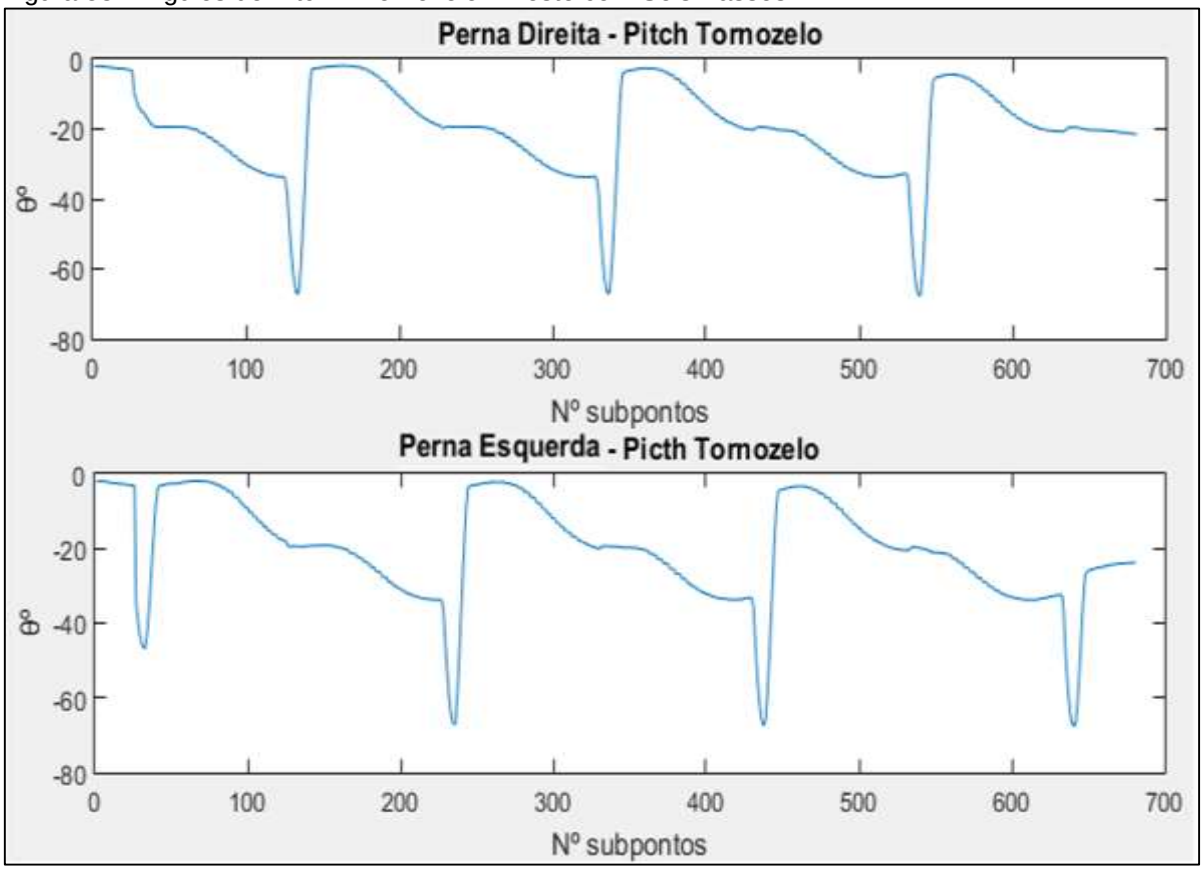
Fonte: O próprio autor (2017).

Figura 57: Ângulos de Pitch – Joelho – Teste com Seis Passos



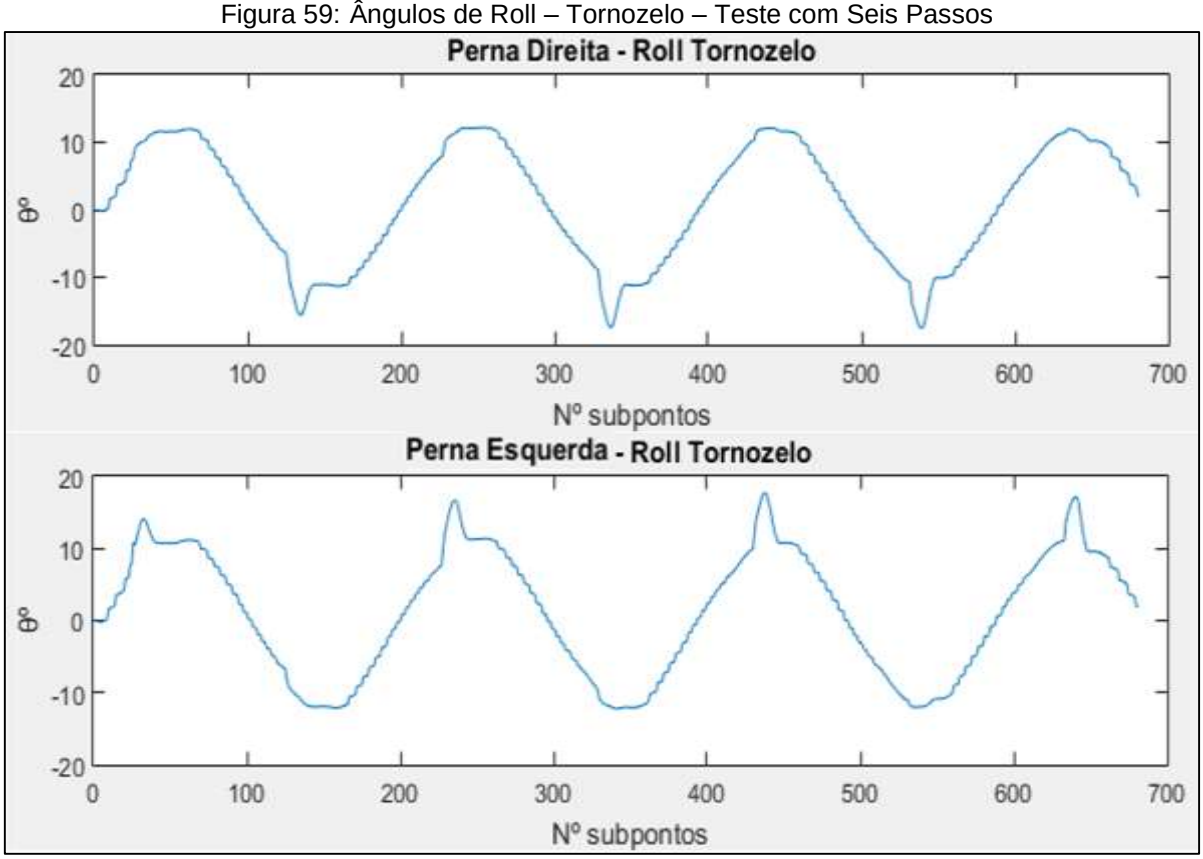
Fonte: O próprio autor (2017).

Figura 58: Ângulos de Pitch – Tornozelo – Teste com Seis Passos



Fonte: O próprio autor (2017).

Figura 59: Ângulos de Roll – Tornozelo – Teste com Seis Passos



Fonte: O próprio autor (2017).

APÊNDICE M – CÓDIGO REALIZADO PARA ARDUINO MEGA 2560

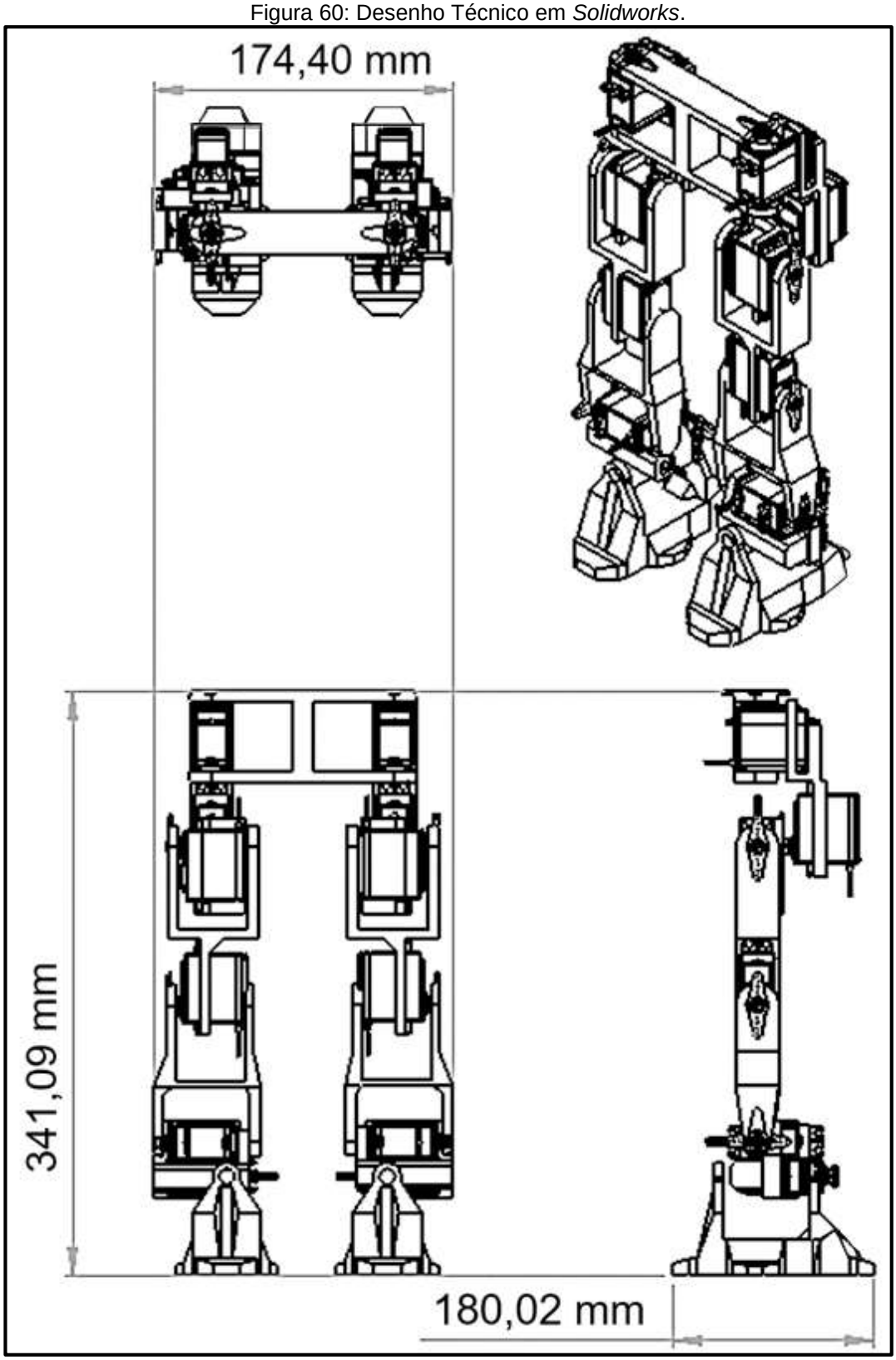
Segue código realizado no ambiente de programação do Arduino Mega 2560. Não são apresentados todos os ângulos para uma marcha aqui. Conforme já informado, para facilitar, não foi utilizado porta serial. Dessa forma evitou-se perdas de informação e tempo de testes. O número completo de ângulos passados para o caso dos três passos fora de 4560 e o de seis, 8160.

```
#include <Servo.h>

// cria objeto tipo Servo para cnto servo
Servo myservo1;
Servo myservo2;
Servo myservo3;
Servo myservo4;
Servo myservo5;
Servo myservo6;
Servo myservo7;
Servo myservo8;
Servo myservo9;
Servo myservo10;
Servo myservo11;
Servo myservo12;

void setup(){
//attach() seleciona o objeto Servo ao pino desejado.
//PWM dos pinos 2 - 13
  myservo1.attach(2);
  myservo2.attach(3);
  myservo3.attach(4);
  myservo4.attach(5);
  myservo5.attach(6);
  myservo6.attach(7);
  myservo7.attach(8);
```

```
  myservo8.attach(9
  myservo9.attach(10);
  myservo10.attach(11);
  myservo11.attach(12);
  myservo12.attach(13);
}

void loop() {
//write() escreve o ângulo desejado através de map(), no objeto no servo
myservo1.write(map(90, 0,180,700,2300));
myservo2.write(map(86, 0,180,700,2300));
myservo3.write(map(111, 0,180,700,2300));
myservo4.write(map(176, 0,180,700,2300));
myservo5.write(map(89, 0,180,700,2300));
myservo6.write(map(90, 0,180,700,2300));
myservo7.write(map(90, 0,180,700,2300));
myservo8.write(map(80, 0,180,700,2300));
myservo9.write(map(74, 0,180,700,2300));
myservo10.write(map(6, 0,180,700,2300));
myservo11.write(map(101, 0,180,700,2300));
myservo12.write(map(85, 0,180,700,2300));
delay(10);
//.....
}
```

APÊNDICE N – DESENHO TÉCNICO DO ROBÔ BÍPEDE DE 12 DOF DESENVOLVIDO PARA O TRABALHO EM *SOLIDWORKS*

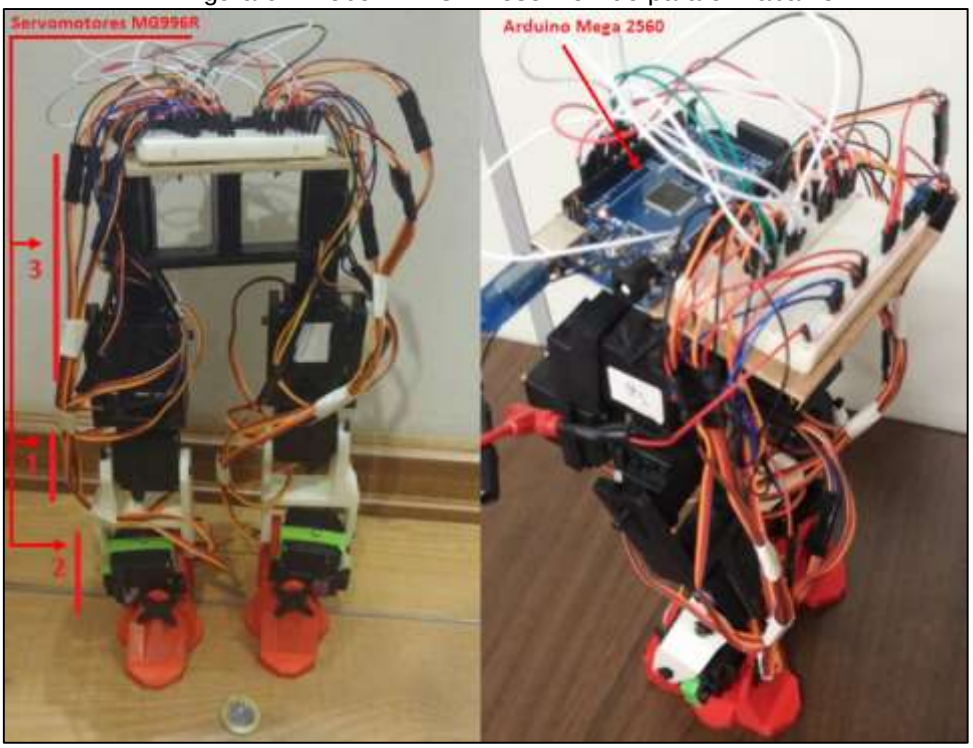
Figura 60: Desenho Técnico em *Solidworks*.



Fonte: O próprio autor (2017).

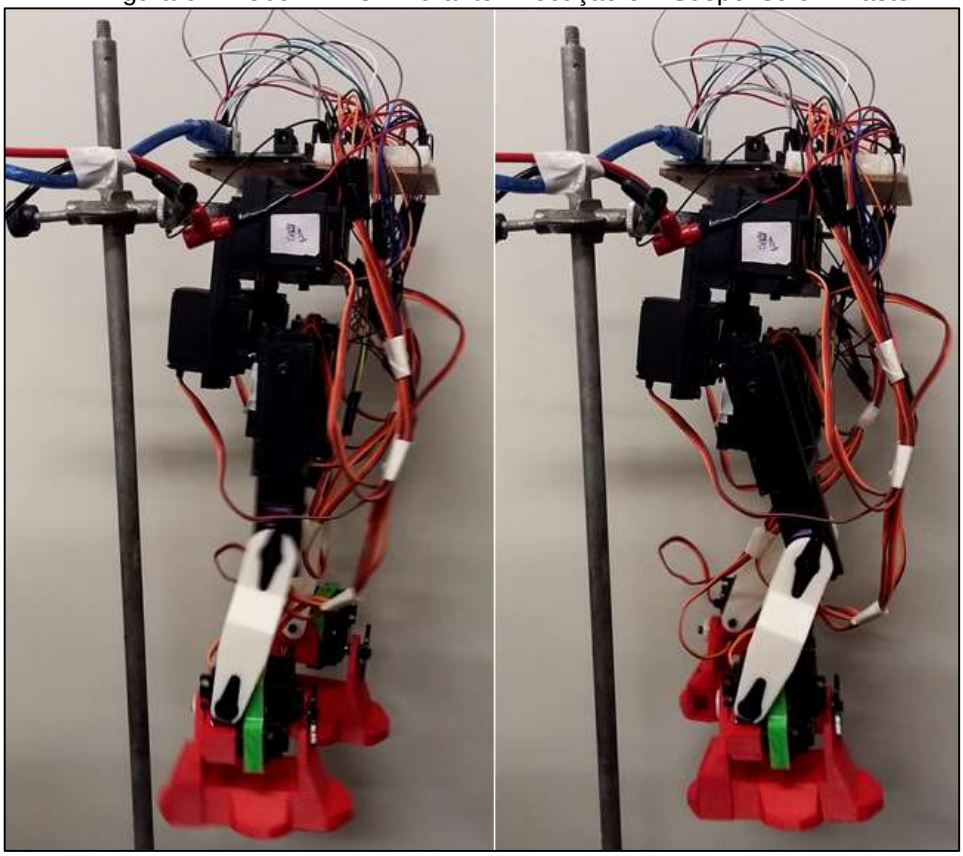
APÊNDICE O – ROBÔ BÍPEDE DE 12 DOF DESENVOLVIDO PARA O TRABALHO

Figura 61: Robô 12 DOF Desenvolvido para o Trabalho



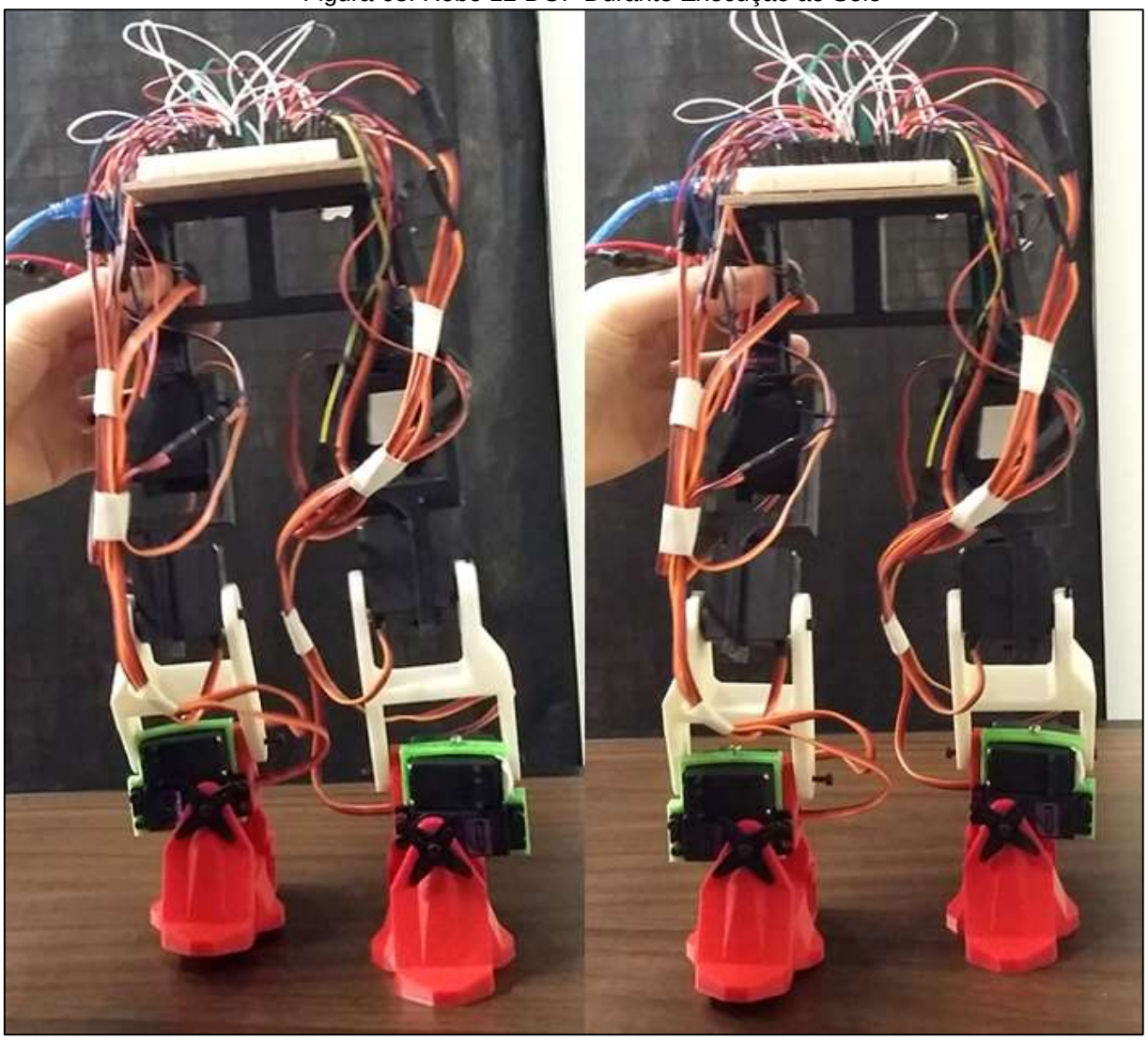
Fonte: O próprio autor (2017).

Figura 62: Robô 12 DOF Durante Execução em Suspenso em Haste



Fonte: O próprio autor (2017).

Figura 63: Robô 12 DOF Durante Execução ao Solo



Fonte: O próprio autor (2017).