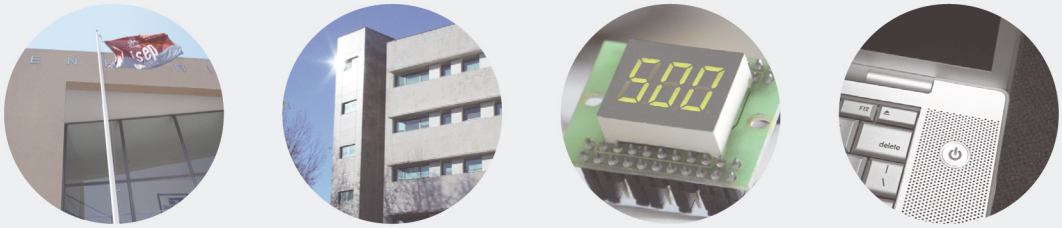
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES



# Modelação, Simulação e Implementação de Padrões de Locomoção para Robôs Hexápodes

**LUIZ FERNANDO PINTO DE OLIVEIRA**
julho de 2016

POLITÉCNICO
DO PORTO

**Instituto Superior de Engenharia do Porto**
Departamento de Engenharia Electrotécnica
Rua Dr. António Bernardino de Almeida 431, 4200-072 Porto

# Modelação, Simulação e Implementação de Padrões de Locomoção para Robôs Hexápodes

Tese/Dissertação de Mestrado em Engenharia Electrotécnica e de Computadores - Área de Especialização de Sistemas Autónomos elaborado por:

Luiz Fernando Pinto de Oliveira

Orientador: Prof. Manuel Fernando dos Santos Silva
Co-orientador: Prof. Flávio Luiz Rossini

Ano Lectivo: 2016

# Agradecimentos

Primeiramente agradeço a Deus por me abençoar em todas as etapas da minha vida.

Agradeço à Universidade Tecnológica Federal do Paraná (UTFPR) por permitir o intercâmbio com o Instituto Superior de Engenharia do Porto (ISEP) que por sua vez me acolheu e instruiu-me ao longo de toda a estadia em Portugal.

Agradeço também ao professor orientador Manuel Silva que prestou todo apoio técnico e científico para a conclusão desta tese, ao professor co-orientador Flávio Rossini que auxiliou-me na correção bibliográfica, ao professor Roberto Neli pelo apoio e esclarecimento quanto ao programa de Dupla Titulação e aos meus queridos e amados pais Luiz Carlos Rodrigues de Oliveira e Rozilda da Silva Pinto que não mediram esforços em me ajudar na carreira académica.

# Resumo

A grande maioria de robôs móveis terrestres usam as rodas como mecanismo de locomoção. É evidente sua vantagem, quanto ao menor consumo de energia, frente aos sistemas multi-pernas. Porém os robôs com pernas conseguem inserir-se em ambientes aos quais os robôs com rodas são simplesmente incapazes de locomoverem-se, como é o caso de obstáculos de grandes dimensões, terreno solto ou encostas íngremes.

Embora os robôs multi-pernas estejam ainda concentrados no campo da investigação, montá-los e executar uma determinada sequência de testes pode ser algo lento e dispendioso, pois o ambiente de simulação possui uma baixa relação custo/benefício, quando relacionado à construção do protótipo.

Deste modo se o projeto de tais robôs for feito em um ambiente de simulação questões como qualidade e/ou limites de operação dos componentes do sistema podem ser desprezados sendo mantido apenas as características consideradas importantes para a análise. A simulação permite tanto viabilizar a pesquisa quanto reproduzir modelos que se aproximem dos valores ideais, embora nada possa ser comparado a experimentos reais, onde o robô de fato opera.

O objetivo deste trabalho consiste em desenvolver a modelagem de robôs hexápodes e simulá-los em um ambiente virtual de sistemas mecânicos, que viabiliza a análise de desempenho de diversos critérios do modelo, bem como averiguar a trajetória virtual construída pelo robô.

**Palavras-Chave**

Robôs com pernas, Hexápodes, SimMechanics™, Modelação, Simulação, Locomoção

# Abstract

The great majority of terrestrial mobile robots uses the wheels as locomotion mechanism. It is obvious their advantage, as to lower energy consumption compared to multi-legs. However, the robots with legs can insert themselves in environments for which the robots with wheels are simply incapable of be inserted, as is the case of obstacles of large dimensions, terrain loose or steep slopes.

Although the robots multi-legs are still concentrated in the field of research, assemble them and perform a particular sequence of tests can be something slow and costly, because the simulation environment has a low cost/benefit ratio, when related to the construction of the prototype.

In this way if the project of such robots will be made in a simulation environment questions as quality and/or limits of operation of the components of the system can be rejected being kept only the characteristics considered important for the analysis. The simulation allows both facilitate research as play models that approximate the ideal values, although nothing can be compared to real experiments, where the fact of robot operates.

The objective of this work is to develop the modeling of robots hexapods and simulate them in a virtual environment of mechanical systems, which makes possible performance analysis of several model criteria and determine the virtual path built by the robot.

**Keywords**

Legged robots, Hexapods, SimMechanics™, Modeling, Simulation, Locomotion

# Conteúdo

# Lista de Figuras

# Glossário

| Sigla | Descrição |
|---|---|
| $\Delta S$ | Deslocamento do centro de gravidade do robô |
| $\Delta S_D$ | Deslocamento do lado direito do robô |
| $\Delta S_E$ | Deslocamento do lado esquerdo do robô |
| $\Delta\theta$ | Ângulo de desvio da trajetória do centro de gravidade do robô |
| 3D | Três dimensões |
| $b$ | Distância entre as ancas direita e esquerda |
| CAD | *Computer aided design* |
| CG | Centro de gravidade |
| CI | Circuito integrado |
| CPU | *Central processing unit* |
| CS | *Coordinate system* |
| cte | Constante |
| DARPA | *Defense advanced research projects agency* |
| $D_{av}$ | Densidade de dispersão da potência média |
| DOF | *Degree of freedom* |
| $E_{av}$ | Densidade de energia absoluta média |
| $F_C$ | Altura de elevação do pé do robô ao solo |
| $F_L$ | Força média na interface corpo-perna |
| GPIO | *General purpose input/output* |
| GPS | *Global positioning system* |
| GPU | *Graphics processing unit* |
| GUIDE | *Graphic user interface development environment* |
| *HB* | Altura do corpo do robô ao solo |
| $I^2C$ | *Inter-integrated circuit* |
| IP | *Internet protocol* |
| ISEP | Instituto superior de engenharia do Porto |
| LIDAR | *Light detection and ranging* |
| $L_s$ | Comprimento do passo |
| MBD | *Multibody dynamic* |
| MEMS | *Microelectromechanical systems* |

| Sigla | Descrição |
|---|---|
| MIT | *Massachusetts institute of technology* |
| NASA | *National aeronautics and space administration* |
| P | Controlador proporcional |
| PD | Controlador proporcional e derivativo |
| PID | Controlador proporcional, integral e derivativo |
| PWM | *Pulse width modulation* |
| SDD | Sistema de direção diferencial |
| SI | Sistema internacional |
| $S_p$ | Distância entre ancas |
| SSH | *Secure shell* |
| $T_L$ | Densidade de potência perdida |
| $V_F$ | Velocidade de deslocamento frontal do corpo do robô |
| $V_L$ | Velocidade de deslocamento da perna do robô |
| ZMP | *Zero moment point* |
| $\beta$ | Fator de ocupação |
| $\phi_i$ | Fase da perna i |

# Capítulo 1

# Introdução

A importância da robótica móvel é evidente nos dias atuais, assim uma de suas tendências é integrar o homem em ambientes por ele classificados como hostis, acidentados, de difíceis acesso como florestas e túneis de mineração ou tarefas do tipo busca e resgate [14]. Os robôs que utilizam rodas como meio de deslocamento, são grandemente empregados em vários setores, quer comerciais quer industriais. Porém, é evidente que na natureza não há sequer um animal que possui rodas como forma de sustentação e deslocamento. Por outro lado, nem todos os seres vivos terrestres possuem pernas, como é o caso das cobras e minhocas que necessitam de se rastejarem para locomoverem-se.

A grande maioria dos animais terrestres possuem duas (ser humano, avestruz), quatro (cães, gatos, cavalos), seis (insetos) e oito (aranhas, escorpiões) pernas. De fato, as pernas de seres biológicos são uma boa forma de locomoção em um ambiente repleto de obstáculos, de formas e tamanhos variados, e ainda não necessitam de um suporte contínuo com o solo, como é o caso das rodas e lagartas. Porém, em sistemas robóticos multi-pernas há certos impasses energéticos, de estabilidade, velocidade, conforto, mobilidade e até o impacto ambiental [15] que os distanciam de uma locomoção ideal.

Assim trabalhos como o [2] com bípedes, [3] com quadrúpedes e [7] com hexápodes apesar de serem realizáveis, existem muitos fatores que agem sobre o sistema, como as limitações físicas dos componentes do sistema. Neste momento vê-se a grande importância do uso de softwares de simulações de sistemas, que permitem analisar os fenómenos de interesse de forma isolada ou integrada ao sistema. Neste caso, apesar de ser necessário possuir um alto grau de conhecimento das ferramentas de simulação, ainda a relação custo benefício permanece a favor dos recursos computacionais.

## 1.1 Contextualização

O presente trabalho visa ser aplicado na modelação de robôs com pernas em ambientes de simulação, por este apresentar inúmeras vantagens face a um teste realizado na prática. O sistema robótico que será modelado será baseado em robôs hexápodes, por permitirem uma maior estabilidade estática de seu corpo em relação aos bípedes e quadrúpedes, possuir maior quantidade de execução de movimentos (vários padrões de locomoção) e menor consumo energético quando comparado aos octópodes.

Quanto a simulação, será utilizado o SimMechanics™, *toolbox* do Simulink®, que por sua vez é um ambiente de simulações incorporado ao *software* de cálculos matemáticos MatLab®. Todas as simulações serão realizadas com base em um modelo CAD desenvolvido em SolidWorks®, *software* que produz dentre outras análises o modelo mecânico de peças, produtos e equipamentos em 3D e possui todas as características de uma peça real, como seu material, massa, volume, centro de massa e inércia.

## 1.2 Problema

A problemática da simulação, quer de robôs, quer de qualquer outro sistema físico realizável, está na sua correta perceção das várias equações que envolve a dinâmica do objeto em análise. Assim, a simulação de tal objeto será tão fiel quanto forem os esforços necessários para extrair as equações matemáticas do sistema.

Devido à complexidade dos algoritmos de controlo [16], a abordagem do tema requer o desenvolvimento de um algoritmo eficiente de forma a não limitar a análise dos recursos utilizados pelas ferramentas dos softwares adotados.

Outra parte do desafio está na integração de diversos softwares, MatLab®, Simulink®, SimMechanics™ e SolidWorks®, para obter os recursos necessários para a exibição do modelo em 3D com todas as características embebidas.

## 1.3 Motivação

A riqueza de informações que se pode obter e integrar em ambientes de simulação é algo de tão grande importância para o desenvolvimento de novos produtos, equipamentos, instrumentos e tecnologias em geral que a corrente obra dedica-se especialmente em abordá-las do ponto de vista robótico.

Quanto ao tipo de robô, os robôs hexápodes permitem explorar e aprimorar diversas áreas da engenharia, como sistemas de controlo, dinâmica avançada, mecânica geral, física, programação e eletrónica.

## 1.4 Objetivos

A intenção da abordagem do tema proposto procura perceber quais são as equações necessárias para modelar um sistema robótico hexápode e assimilar corretamente o uso do software com capacidade de visualização em 3D.

Para efetuar a modelagem do robô pretende-se realizar o estudo de sua cinemática de locomoção, tanto direta quanto inversa, de forma a compreender como os ângulos das diversas juntas constituintes de cada perna do robô relacionam-se com as coordenadas cartesianas da posição desejada para cada momento específico.

De seguida aprofunda-se a abordagem à luz da dinâmica da locomoção, de modo a explorar as forças exercidas em cada componente do sistema, com o intuito de obter os binários envolventes de cada junta e assim obter o binário mínimo necessário para o correto funcionamento do sistema.

Desenvolver o planeamento de trajetória de cada perna do robô com base em equações matemáticas específicas para cada tipo de movimento.

Analisar o comportamento da associação de todas as pernas que interferem na atitude do robô de acordo com as diversas formas de marchas de um robô hexápode.

Simular o contacto dos pés do robô ao solo através de um sistema mola-amortecedor, uma vez que o mesmo afeta diretamente na estabilidade e consumo energético do sistema.

Por fim englobar toda a modelagem do hexápode através do SimMechanics™ com o propósito de executar todo o controlo dos subsistemas com seus respetivos centros de gravidade e inércia baseados nas representações em 3D de cada componente em formato CAD.

## 1.5 Planeamento do projeto

De modo a serem concretizados todos os tópicos propostos, seguiu-se uma cadência de tarefas realizadas diariamente durante o período de seis meses, conforme a calendarização vista na Tabela 1.1. Os primeiros dois meses foram dedicados à busca de informações, tanto de robôs com pernas, quanto da ferramenta de simulações mecânicas utilizada no Simulink®. Entre maio e junho o estudo concentrou-se na modelação do sistema e os dois meses restantes foram dedicados às simulações virtuais e testes práticos, onde procurou-se cumprir categoricamente todos os prazos previstos.

| | Março | Abril | Maio | Junho | Julho | Agosto |
|---|---|---|---|---|---|---|
| Estudo dos sistemas robóticos multi-pernas | √ | | | | | |
| Estudo dos *softwares* de simulação existentes | √ | | | | | |
| Compreensão do princípio de funcionamento do SimMechanics™ | | √ | | | | |
| Estudo cinemático do robô Hexápode | | √ | √ | | | |
| Planeamento de trajetórias | | | √ | | | |
| Análise e desenvolvimento de padrões de locomoção | | | √ | | | |
| Representação do robô em SimMechanics™ | | | √ | √ | | |
| Desenvolvimento de uma GUIDE | | | | √ | | |
| Simulação do robô | | | | √ | √ | |
| Implementação do algoritmo em um Hexápode | | | | √ | √ | √ |
| Revisão | | | | | | √ |

Tabela 1.1: Calendarização da Tese no ando de 2016

## 1.6 Estrutura da dissertação

Para uma compreensão progressiva, este trabalho é particionado em sete capítulos: Introdução, Estado da Arte, SimMechanics™, Modelação do Robô Hexápode, Simulação em SimMechanics™, Implementação e Conclusões.

O primeiro Capítulo busca descrever, de forma clara e objetiva, quais são as circunstâncias que levaram o autor a eleger tal tema, bem como definir o cronograma previsto para a execução completa da Tese.

O Capítulo 2 visa adquirir uma base de dados fundamental para o presente trabalho, proveniente dos sistemas robóticos multi-pernas de excelência desenvolvidos até a data vigente, além de expor as características dos notáveis *softwares* de simulação capazes de reproduzir a locomoção de robôs hexápodes.

De forma a instruir os conceitos básicos de escolha, criação, ligação e funcionamento, o terceiro capítulo dedica cada secção para cada bloco funcional do SimMechanics™ que será empregue no modelo de simulação desenvolvido no Capítulo cinco.

A modelação de um robô hexápode deve ser conduzida de forma progressiva, iniciar através do estudo cinemático e finalizar com o estudo da dinâmica. Desse modo o Capítulo quatro descreve todos os cálculos e equações que devem ser efetuados afim de se obter a correta abstração de um hexápode.

O Capítulo cinco, em conformidade com os conhecimentos indispensáveis referidos no terceiro Capítulo, encarrega-se de produzir o modelo do robô através da ferramenta SimMechanics™ de modo a unir cada perna ao corpo e definir os locais de aplicação das forças de reação do solo. Este Capítulo também é responsável por avaliar os dados de posição, velocidade, aceleração e força da extremidade de cada perna do robô que possui um sistema de controlo para cada junta.

A sexta parte desse trabalho é voltada para a experimentação, colocando em prática a modelação desenvolvida no Capítulo 3, tal como os padrões de locomoção simulados no Capítulo 5. Através de um robô já construído, o Capítulo seis visa efetuar a comparação entre o modelo de simulação e um robô hexápode real.

O último capítulo desta obra descreve uma auto crítica de forma a exibir um balanço dos objetivos e metas alcançados e propor eventuais desenvolvimentos futuros além de relatar as conclusões obtidas.

# Capítulo 2

# Estado da arte

*Existem diversas formas de configurações de robôs multi-pernas, cada qual possuindo suas vantagens e desvantagens. Este capítulo pretende tanto analisar do ponto de vista da robótica de locomoção robôs bípedes, quadrúpedes, hexápodes e octópodes, quanto expor os softwares computacionais atuais que permitem realizar a simulação de tais sistemas.*

## 2.1 Robótica de locomoção

Em robótica móvel terrestre há várias vertentes quanto ao método que os robôs utilizam para se locomover, através de rodas, pernas, esteiras ou até mesmo o próprio corpo (robôs inspirados biologicamente). Concentrando-se nas pernas de um robô, as secções seguintes procuram compreender quais são as várias configurações que estes sistemas podem dispor.

### 2.1.1 Bípedes

Graças a um anúncio publicado em dezembro de 1996 de uma grande empresa japonesa, Honda, robôs humanóides que antes eram considerados artigos de ficção científica puderam arrancar interesse suficiente da sociedade científica de forma a possibilitar novos avanços nesse ramo tecnológico. O robô P2 da Honda, Figura 2.1, teve suas pesquisas e desenvolvimento iniciados em 1986 [1] e foi desenvolvido para servir como um robô industrial, capaz de auxiliar o humano em tarefas industriais.

Desse modo ele deveria ser móvel e permitir suportar cargas extras. Após intensas pesquisas, foi concluído que os sistemas robóticos necessitariam de sensores de inclinação do corpo e sensores de força de reação dos pés com o solo.

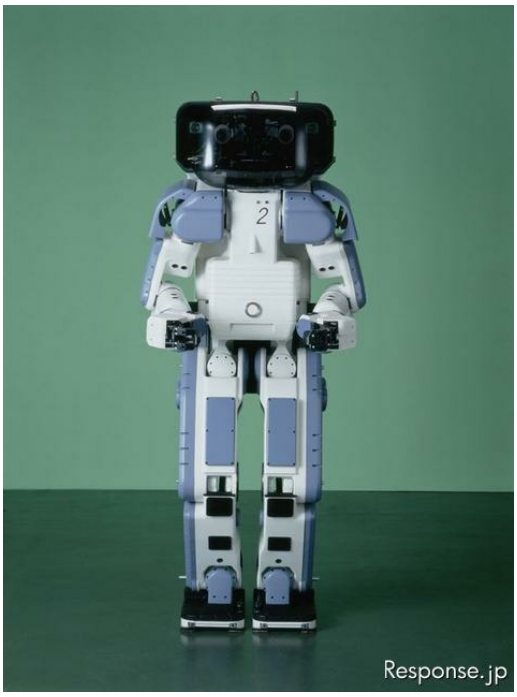
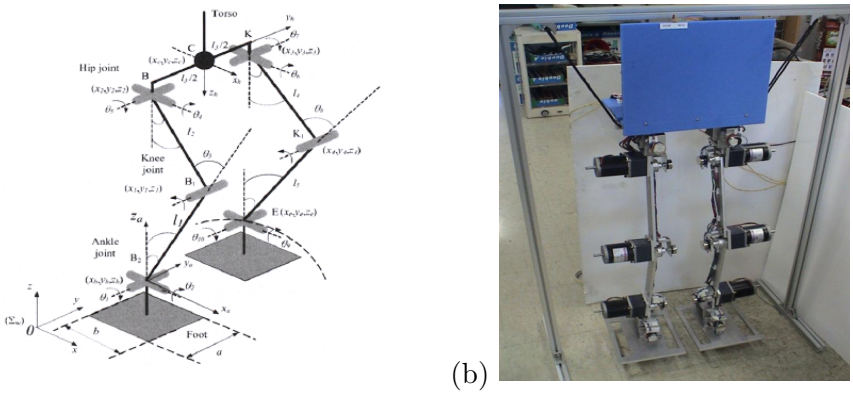



Figura 2.1: Robô humanóide P2, Honda [1].

Notou-se a importância da absorção das forças de impacto com chão recorrendo a mecanismos de absorção de impacto.

Robôs com duas pernas, ou bípedes, utilizam-se apenas de seu par de pernas para moverem-se, realizando sua locomoção em relação ao chão mantendo seu equilíbrio constantemente de modo a evitar sua queda. Segundo Phuong et al sua modelação é baseada em um pêndulo invertido em três dimensões, onde seu sistema de controlo é realizado baseado na solução da cinemática inversa determinada pelo método da geometria dos sólidos, de modo que seu padrão de locomoção é gerado por um sistema de controle de trajetória referido ao *Zero Moment Point* (ZMP) [2] , permitindo assegurar a locomoção dinâmica do robô bípede [17].

A dificuldade encontrada por Phuong et al nesses robôs está na complexidade do controlo de estabilidade perante sua locomoção. A modelação foi feita considerando um robô bípede com 10 *Degrees of Freedom* (DOF) e foi implementada no robô CIMEC-1, Figura 2.2. Segundo [13] por possuir o menor número de pernas dos sistemas robóticos com pernas apresenta o menor consumo energético.




(a) (b)

Figura 2.2: Configuração do modelo do robô bípede de 10 DOF (a), Robô bípede CIMEC-1 (b) [2].

### 2.1.2 Quadrúpedes

Comparado aos bípedes, robôs quadrúpedes têm uma maior escolha de colocação dos pés para manter o equilíbrio estático. Sun et al realizaram a modelação de um quadrúpede de acordo com as características de um cão, Figura 2.3, que foi desenvolvido para o estudo dos animais domésticos robóticos com a função de simular um cão verdadeiro, porém com capacidades de suportar cargas e auxiliar em serviços [3]. Possui doze DOF distribuídos pelas quatro pernas.

Figura 2.3: Robô quadrúpede TIM-I [3]

Quando comparado aos robôs com seis e oito pernas os quadrúpedes desfrutam de uma estrutura mais simples. Porém o desempenho dos quadrúpedes é afetado diretamente quando há uma falha elétrica ou mecânica de uma de suas pernas durante o período de locomoção.

De acordo com [4], conforme o robô adquire velocidade a estabilidade do centro de gravidade (CG) é perdida, denotando a importância do sincronismo

entre cada movimento. O robô TIM-I da Figura 2.3 é inspirado na locomoção de cães, procurando estabelecer uma forma de projetar o movimento das pernas através da relação entre a marcha e o movimento do animal. Toda a modelação do robô foi averiguada e simulada através do software ADAMS™. Na Figura 2.4 é possível visualizar o robô projetado.

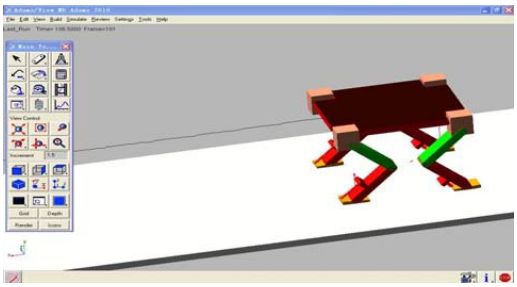


Figura 2.4: Robô quadrúpede TIM-I [4].

As coordenadas do contacto dos pés ao solo foram feitas com base na função *SPLINE* calculada previamente e inserida como comando de entrada no ADAMS™. Na Figura 2.4 ainda é possível verificar uma pequena ondulação na trajetória do CG tanto horizontalmente quanto verticalmente. O autor extraiu as medidas entre o centro do corpo do robô e o solo e as distâncias entre os pés de cada perna ao solo, tudo durante o período de locomoção, onde (embora haja pequenas oscilações) considera o método adotado viável ao tipo de robô e expande o horizonte de aplicações para os demais robôs multi-pernas.

O robô BigDog, desenvolvido pela Boston Dynamics e financiado pela *Defense Advanced Research Projects Agency* (DARPA), é um excelente exemplo de robô quadrúpede, Figura 2.5 [5]. Desenvolvido com o objetivo de construir veículos com pernas autónomos com mobilidade em terrenos acidentados superior aos veículos de rodas e lagartas existentes, possui sensores como *Light Detection And Ranging* (LIDAR), visão estéreo, *global positioning system* (GPS), acelerómetros e giroscópios com o intuito de caracterizarem a atitude do robô e assim deslocar-se em uma ampla variedade de terrenos. Para a sua locomoção, detém um sistema hidráulico com atuadores nas quatro pernas.

Conforme a Figura 2.6, o modelo de suas pernas considera tanto a atuação quanto a complacência em cada uma de suas juntas. Isso permite aproximar o modelo do robô com o modelo biológico de um cão real.

Por se tratar de um robô de alta tecnologia e desempenho percebe-se que toda sua evolução baseou-se em observações de um modelo ideal de locomoção quadrúpede e posteriormente em aproximações matemáticas aplicáveis à robótica de locomoção.

Figura 2.5: Robô BigDog [5].

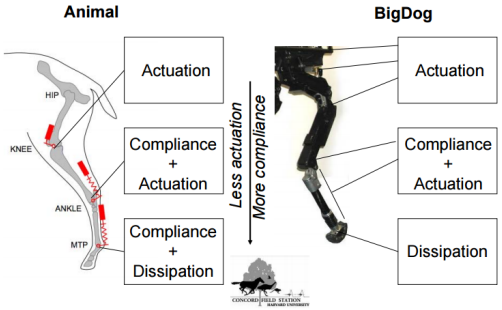



Figura 2.6: Multi-articulamentos da perna do BigDog [5].

### 2.1.3 Hexápodes

Os robôs hexápodes são robôs que possuem um par de pernas a mais do que os robôs quadrúpedes. A priori, a adição de um par de pernas a mais em um sistema robótico que já possui características relevantes para aplicações em terrenos acidentados parece acrescentar apenas mais complexidade e consumo de energia. Porém, apesar de agregar tais características, amplia-se o horizonte de trabalho do robô, pois tanto robôs bípedes quanto quadrúpedes são fortemente afetados quando há uma avaria em uma de suas pernas.

Segundo Gorner e Hirzinger, que estudaram os comportamentos de robôs com 6 e 8 pernas após a remoção de uma de suas pernas baseados no modelo biológico do inseto-pau, perceberam que um inseto após perder um de seus membros rapidamente adapta sua forma de andar de forma a permanecer em uma posição e orientação consideravelmente estável, embora haja uma substancial perda de velocidade de deslocamento e um desvio angular em sua trajetória [18].

Assim devido à sua redundância, hexápodes e octópodes podem, após detetar tanto o mau funcionamento quanto a perda total da perna, reconfigurar seu padrão de locomoção e em certos casos manter o robô estável, com um decréscimo em sua velocidade de movimentação. Para uma cadência estaticamente estável, é fundamental que todas as pernas se movam coordenadamente e que sua marcha siga uma sequência ordenada, evitando assim a colisão de duas pernas vizinhas e mantendo o centro de gravidade sempre dentro do chamado polígono de suporte, área formada pelas coordenadas de posição dos pés em contacto com o solo. Notou-se uma redução de 35% na velocidade em comparação com o robô não danificado. A mudança do CG geralmente melhora o desempenho e a estabilidade de movimento, mas em alguns casos pode até degradar o desempenho.

Conforme Sorin e Nitulescu, dependendo da configuração de suas pernas, o robô poderá ser classificado como [6]:

- estaticamente estável, se a projeção do centro de massa está dentro do polígono de suporte;
- no limite de estabilidade, se a projeção do centro de massa está no limite da área de um dos lados do polígono de suporte;
- estaticamente instável, se a projeção do centro de massa está fora do polígono de suporte.

Este último caso ocorre quando o robô está mudando gradualmente o seu polígono de apoio, na etapa de transição, que devido à gravidade move-o, até que a condição para a estabilidade estática é novamente obtida, etapa de apoio conforme a Figura 2.7.

Sorin e Nitulescu estudaram justamente a análise da estabilidade estática de robôs hexápodes quer em queda livre, onde concluíram que esta análise é útil pois permite investigar o que acontece com o robô em terreno irregular ou queda por acidentes que podem acontecer devido à perda de contacto devido a superfícies escorregadias, insuficiência do sistema atuador ou por falha na alimentação; quer nos instantes de transição, onde descreve a variação do polígono de apoio durante a sequência de marcha, mostrando a importância do correto sincronismo das pernas a fim de satisfazer a condição de estabilidade, Figura 2.8. Toda a simulação

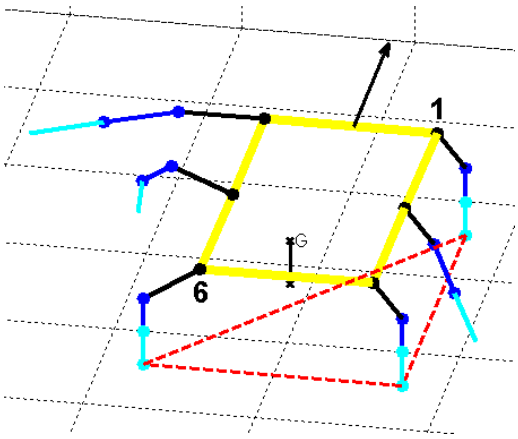



Figura 2.7: Período de transferência - queda [6].

foi realizada através de uma *Graphic User Interface Development Environment* (GUIDE) do MatLab®.

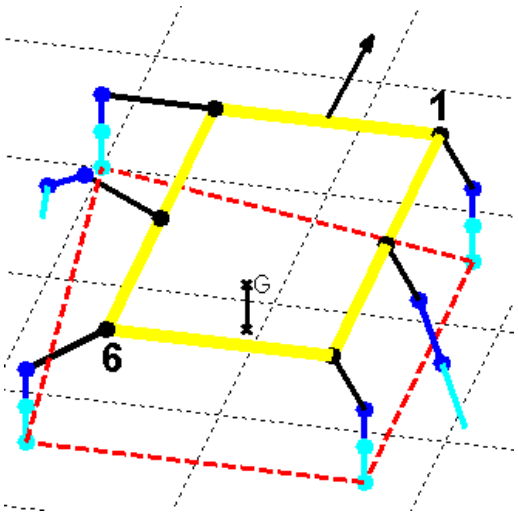



Figura 2.8: Período de suporte - estável [6].

Os *micro-rovers* Attila e Hannibal [7], juntamente com o Genghis, foram os primeiros robôs com pernas do laboratório do *Massachusetts Institute of Technology* (MIT) [19], desenvolvido em parte por uma estudante de graduação do programa de pesquisa da *National Aeronautics and Space Administration* (NASA) administrada através do Laboratório de Propulsão a Jato e pela Agência de Pesquisas Avançadas, Figura 2.9.

Estes robôs não são explicitamente controlados, mas sim possuem um com-

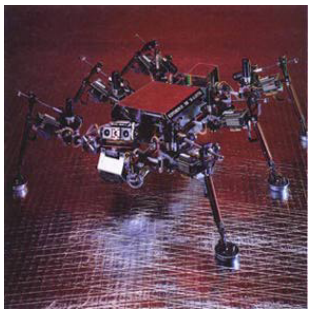
(a)  (b) 

Figura 2.9: Robô Attila, (a). Robô Hannibal, (b) [7].

portamento adquirido da rede chamada de arquitetura de subordinação cuidadosamente projetada para adotar o problema da inteligência de uma perspetiva diferente da tradicional Inteligência Artificial. Foram os responsáveis por criar uma enorme tendência da robótica baseada em comportamento.

A Figura 2.10 mostra um gigantesco hexápode com a capacidade de ser operado por um operador sobre o próprio veículo. Trata-se do Veículo de Suspensão Adaptativa (ASV) desenvolvido por [8, 9]. Com uma capacidade de carga de 220 kg e pesando 2700 kg, o ASV consegue atingir uma velocidade máxima de 8 km/h. Dispõe de 3 DOF por perna, totalizando 18 DOF, permitindo que ele possa executar diferentes marchas de locomoção, tanto periódicas, quanto não periódicas. A vantagem de se permitir um operador humano diminui a complexidade do sistema de controle da marcha das pernas do robô. Porém, dependendo do tipo de marcha utilizada, há substanciais picos de consumo de energia elétrica, logo cada padrão de locomoção possui alguma vantagem dependendo da adaptabilidade do solo, o consumo de energia ou da suavização do movimento do corpo [11].

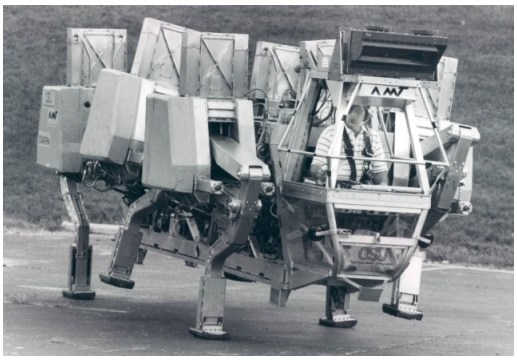


Figura 2.10: Veículo de Suspensão Adaptativa (ASV) [8, 9].

### 2.1.4 Octópodes

Assemelham-se fortemente aos hexápodes por possuírem redundância em sua locomoção. Os robôs SCORPION e DANTE II, Figura 2.11, são exemplos de sucesso desse modo de locomoção. O primeiro foi desenvolvido pela Universidade de Bremen, em parceria com a NASA, com o propósito de aplicações interplanetárias, como trabalhar na superfície do planeta Marte [10, 11]. Já o segundo, DANTE II, foi construido para ser um robô de exploração vulcânica de um vulcão ativo em *Mount Spurr*, Alasca. Embora ele possua suas pernas para descer ao vulcão, possui um cabo umbilical ligado ao seu corpo e a um suporte fixo na superfície do vulcão para evitar sua queda, pois esses cenários são tipicamente muito íngremes [12, 11].

(a)  (b) 

Figura 2.11: SCORPION, (a) [10, 11]. DANTE II, (b) [12, 11].

Porém, octópodes são pouco implementados, pois exigem:

- Vinte e quatro atuadores para o caso das pernas possuírem 3 DOF;
- Alto custo financeiro devido ao tipo de atuador escolhido;
- Alto consumo energético;
- Maior complexidade no sincronismo das pernas;
- Maior peso.

Deste modo, trabalhos como Ye et al e Rynkevic buscam novas alternativas como músculos artificiais e atuadores *Microelectromechanical Systems* (MEMS), afim de minimizar as limitações encontradas [20, 21].

De acordo com Silva que comparou vários dos sistemas de locomoção multipernas anteriormente citados com base em várias variáveis como $E_{av}$, $D_{av}$, $T_L$, $F_L$ e $\tau_L$, medidas globais referentes ao desempenho do mecanismo, observa-se que robôs com 2, 4, 6, 8 e 10 pernas possuem diferentes áreas de atuação, com

diferentes desempenhos, conforme a Figura 2.12, que apresenta um comparativo entre todos os sistemas e as variáveis descritas anteriormente [13].

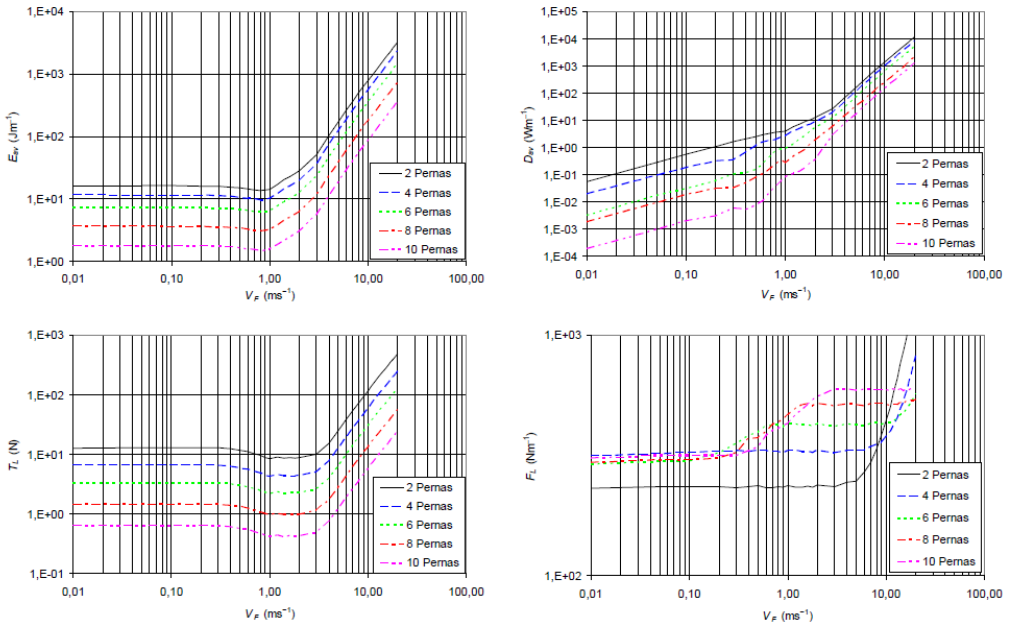



Figura 2.12: $\min[E_{av}(V_F)]$, $\min[D_{av}(V_F)]$, $\min[T_L(V_F)]$ e $\min[F_L(V_F)]$, para 2,4,6,8 e 10 pernas com 1 cm de $F_C$ [13] .

Analisando todas as arquiteturas, variando-se sua velocidade frontal, $V_F$, percebe-se indiscutivelmente que quanto menor for o número de pernas do robô maior é a eficiência em realizar a locomoção. Conclui também que há certos valores de $V_F$ onde cada estrutura se destaca. Portanto os octópodes necessitam de maiores avanços em suas pesquisas de forma a possuírem um maior prestígio dos pesquisadores.

## 2.2 Softwares de simulação

Esta secção dedica-se a descrever a importância dos softwares computacionais que são usados como forma de construção, visualização e análise da modelação de sistemas robóticos, quer no campo da investigação, quer no campo da comercialização, umas vez que a área da robótica apresenta um mercado consumidor ativo.

### 2.2.1 ADAMS™

Por ser utilizado em várias áreas das indústrias aéreas e automobilísticas, bem como na implementação da dinâmica de múltiplos corpos (MBD), o software

ADAMS™ é considerado o software de simulação mais famoso do mundo [22]. Simulações como a sequência de pouso do *Curiosity Rover* atraem os olhares de grandes empresas e pesquisadores que necessitam de modelagem. É frequentemente utilizado para realizar a análise dinâmica do movimento dos sistemas mecânicos e permite a agregação de módulos adicionais para a expansão da análise computacional. O robô quadrúpede TIM-I, visto na Figura 2.4, é um exemplo de aplicação do *software* ADAMS™.

### 2.2.2 SimMechanics™

O Simscape Multibody™ é um pacote de ferramentas disponível no ambiente do Simulink® com o intuito de simular fenómenos elétricos, eletrónicos, mecânicos, hidráulicos e sistemas de potência. A parte formalmente conhecida por simular sistemas mecânicos do ambiente Simulink®/MatLab® é chamada de SimMechanics™. Não só permite a simulação de robôs em um ambiente 3D, mas também a modelação de sistemas de múltiplos corpos.

Toda a modelação é feita através de diagramas de blocos, no qual cada elemento do diagrama representa uma característica do modelo mecânico em análise, como corpos, juntas, limitações, elementos de força e sensores [23]. Permite importar modelos mecânicos finalizados em outros softwares CAD, de forma que características como massa, inércia e toda a geometria do modelo possa ser preservada e contida nos respetivos blocos constituintes do sistema.

### 2.2.3 Webots

O Webots é capaz de modelar, programar e simular robôs móveis. Os controladores do robô podem ser desenvolvidos na própria *Integrated Development Environment* (IDE) do ambiente de desenvolvimento [24]. Por estar à mais de 19 anos em serviço, o software garante a poupança de tempo, integrando diversos recursos previsíveis em aplicações de robótica, além de possuir as versões *Free*, EDU e PRO de forma a trabalhar com diferentes públicos alvo. Especifica o tipo de colisão com objetos independentemente de sua forma, não há limites de quantidade de robôs simulados e detém uma larga quantidade de sensores disponíveis.

### 2.2.4 Análise e seleção de tecnologias

Embora todos os softwares anteriormente citados sejam capazes de realizar a simulação do robô hexápode, o presente trabalho visa implementar a simulação do robô no ambiente de simulação Simulink®, recorrendo ao SimMechanics™, pelas seguintes razões:

- importação de arquivos XML através da interface com plataformas como SolidWorks®;

- em um único ambiente, elabora a simulação de sistemas mecânicos em conjunto com sistemas físicos multi-corpos e algoritmos de controle em Simulink®;
- criação/manipulação de varáveis e algoritmos em *scrips* do MatLab® com o uso de seu *workspace*;
- visualização da animação em 3D da dinâmica do sistema;
- relativa simplicidade de modelação e funcionamento das ferramentas do ambiente de programação Simulink®.

## 2.3 Conclusão

Portanto, com o exposto até aqui, permitiu-se selecionar de forma clara e objetiva a tecnologia que será utilizada pelo *software* computacional e relataram-se suas semelhanças e diferenças com outros sistemas robóticos de locomoção. Por fim descreveram-se os softwares capazes de modelar o robô em questão, do mesmo modo que eleger o mais adequado para a singular aplicação com os robôs hexápodes.

# Capítulo 3

# SimMechanics™

*Este capítulo dedica-se a apresentar os conceitos básicos utilizados no ambiente de simulações Simulink® através das ferramentas de análise do Simscape™. Para tal, serão abordados os blocos relacionados com a mecânica do sistema, isto é, os blocos do SimMechanics™.*

A primeira tarefa a ser feita é o correto ajuste dos parâmetros de configuração do modelo. A Figura 3.1 exibe as configurações adotadas neste trabalho. Todas as escolhas foram feitas de maneira a otimizar o tempo de execução da simulação, logo conforme [25] propõem o *solver*, método que o Simulink® utiliza para calcular os estados do modelo durante a simulação e geração de código, do tipo ode15s pois apresenta maior velocidade de simulação para o modelo corrente.

Quanto ao incremento temporal, foi selecionado *Variable-step* para que o tamanho do passo possa reduzir quando os estados do modelo mudam rapidamente, para manter a precisão, ou aumentar quando os estados do modelo mudam lentamente, para evitar etapas desnecessárias [26], sendo o tamanho do passo máximo, mínimo e inicial calculados automaticamente pelo Simulink®.

O tempo de simulação inicia-se em 0 s e termina em *stop_time* s, sendo *stop_time* uma variável do *workspace* do MatLab®. Por último, mas não menos importante, tem-se a opção de cruzamentos por zero (*Zero-crossing*) que neste caso foi selecionado *Use local settings* para que seja possível computar a passagem por zero pelo bloco *Storage Initial State* da subsecção 5.2.1.

## 3.1 Machine Environment

Antes de iniciar o estudo dos blocos que representaram de fato as partes mecânicas do sistema robótico, deve-se primeiro inserir o bloco virtual *Machine Environment*, Figura 3.2.

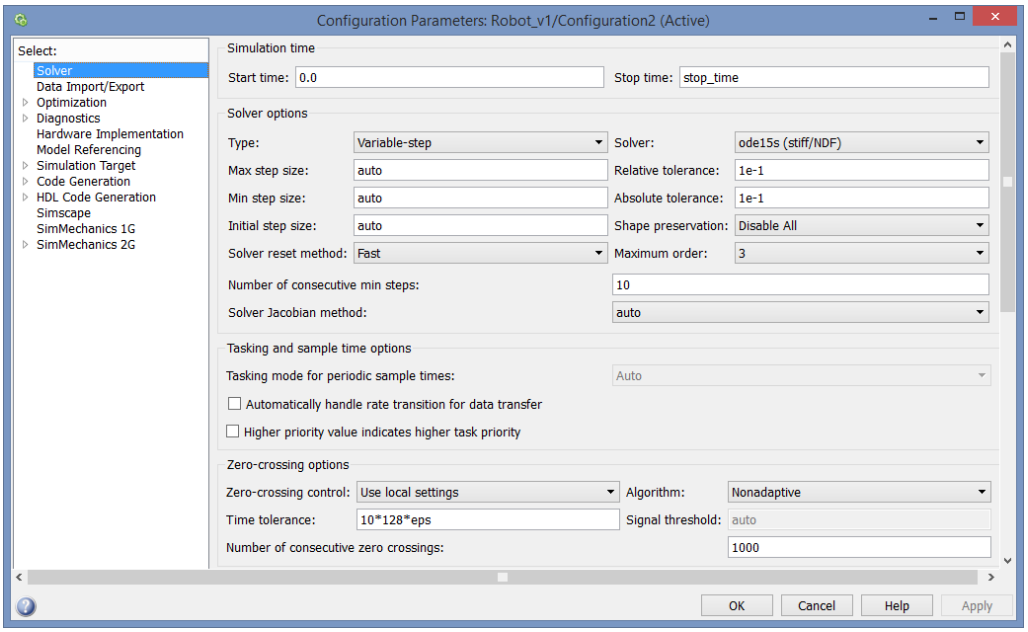



Figura 3.1: Janela de configuração dos parâmetros do modelo em Simulink®.

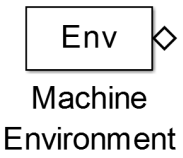



Figura 3.2: Bloco *Machine Environment*, fundamental para a simulação.

Este bloco não possui representação física mas é de extrema importância, pois ele é responsável por determinar:

- modo de análise;
- o vetor da aceleração da gravidade;
- as dimensões do projeto;
- tolerâncias da modelação linear;
- tolerâncias da modelação angular;
- restrições;
- linearização;

- visualização.

O modo de análise permite a escolha do tipo de resolução para a simulação do movimento, sendo que dentre as opções disponíveis será adotada a dinâmica direta. Este modo poupa significativamente os esforços do capítulo posterior, pois através de um algoritmo de planeamento de trajetórias que forneça todos os ângulos das diversas juntas das pernas do robô e de um sistema de controlo que atue sobre estes dados, o mesmo fornecerá como variável de saída o binário necessário para uma junta rotacional posicionar na posição angular de *Setpoint*, Figura 3.3.

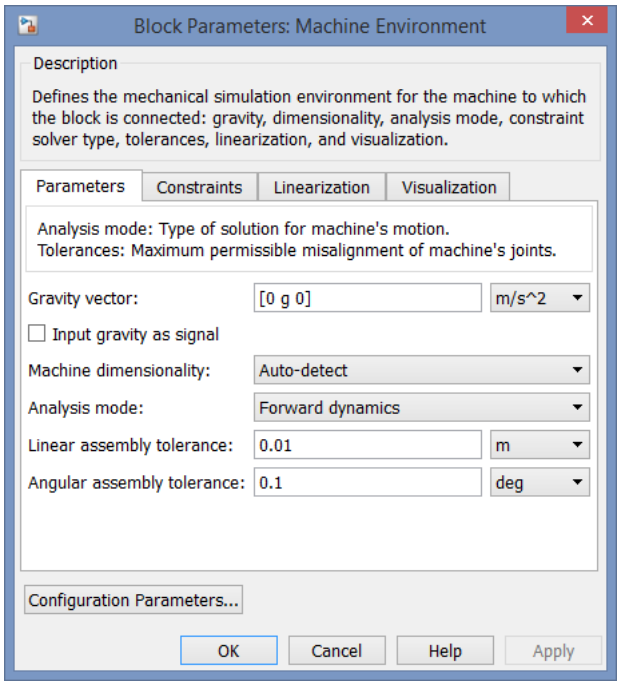



Figura 3.3: Aba *Parameters* do bloco *Machine Environment*.

O vetor de aceleração da gravidade permite escolher em qual eixo será aplicada a aceleração, sendo possível simular gravidade zero ou ainda efetuar a leitura da aceleração da gravidade através de um sinal de entrada. Neste caso, optou-se por aplicar uma aceleração com magnitude de $9,81$ m/s$^2$ sobre o eixo $Y$ e de sentido contrário ao seu eixo, onde $g$ é uma variável do *workspace* do MatLab®. As dimensões do projeto podem ser duas, três ou auto-detetadas. Este campo foi inalterado, permanecendo em auto-detetadas, embora o projeto seja em três dimensões. Quanto às tolerâncias da modelação linear e angular, foram escolhidos valores altos, 0,01 m e 0,1 °, pois se aplicados valores baixos de tolerância aumenta o processamento computacional e consequentemente o tempo de simulação.

As abas "*Constrains*" e "*Linearization*" também foram mantidos intactas, pois as suas configurações iniciais já estão selecionadas para uma maior velocidade de processamento, como se mostra nas Figuras 3.4 e 3.5. Porém, caso haja a necessidade de extrair a máxima resolução das tolerâncias, perturbações e linearização basta alterá-las.

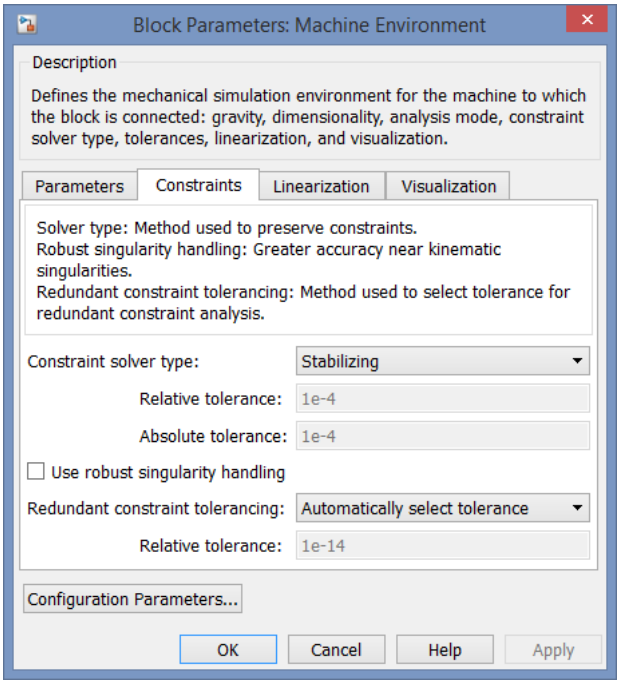



Figura 3.4: Aba *Constrains* do bloco *Machine Environment*.

Por fim, este bloco possui ainda a aba "*Visualization*" que permite alterar o tipo de representação visual do sistema de coordenadas (CS) da simulação. Neste caso optou-se pela representação *deafult*, que representa uma reta para dois CS interligados ou um polígono para três ou mais CS interligados, como se pode ver na Figura 3.6.

## 3.2 Ground

O bloco *Ground*, mostrado na Figura 3.7, é responsável por definir a origem do sistema de coordenadas. Será utilizado para informar tanto a origem do CS do robô como do modelo do solo.

Por permitir a interação com variáveis do *workspace* do MatLab®, criaram-se variáveis para as coordenadas cartesianas $X$, $Y$ e $Z$ do CS do corpo do robô, conforme visto em 3.8, tornando-se possível alterar as coordenadas de sua origem, por motivos posteriormente citados. É neste bloco que o bloco virtual *Machine*

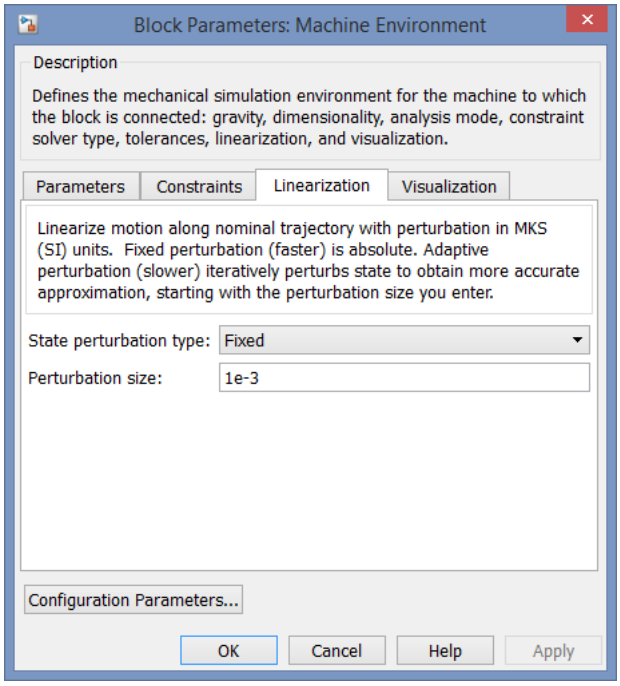



Figura 3.5: Aba *Linearization* do bloco *Machine Environment*.

*Environment* se conecta, inicializando todos os modos de análise citados anteriormente.

## 3.3 Body

O bloco *Body* permite abstrair os aspetos mecânicos do modelo do robô, representará tanto os membros (elos) da perna do hexápode quanto seu corpo, como se mostra na Figura 3.9.

Este bloco possui duas propriedades, a massa e o momento de inércia do *body*. Assim como a massa representa a resistência a um movimento de translação, o momento de inércia representa a resistência a um movimento de rotação. Com a importação do modelo mecânico projetado no SolidWorks® estes campos são preenchidos automaticamente, caso contrário os mesmos devem ser preenchidos de maneira correta, sendo utilizado o teorema dos eixos paralelos para o cálculo do momento de inércia do *body*.

O sistema de posicionamento, aba "*Position*" da Figura 3.10, serve para localizar o *body* no ambiente de simulação. Para isso existem os chamados CS que definem os pontos de referência no *body* e o CG que define o seu centro de gravidade. Tanto os CS quanto o CG possuem um vetor posição da sua origem, porém a sua posição é relativa, isto é, as coordenadas cartesianas do CS/CG po-

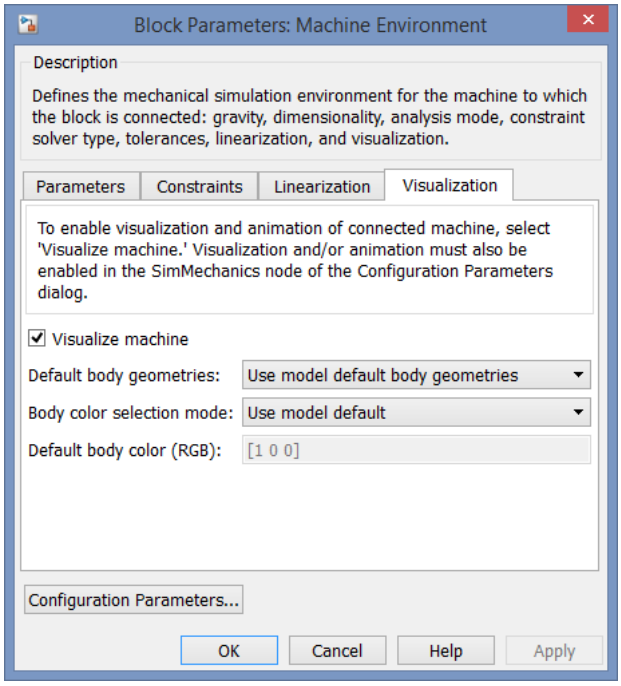



Figura 3.6: Aba *Visualization* do bloco *Machine Environment*.

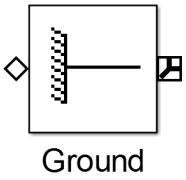



Figura 3.7: Bloco de referência ao solo, *Ground*.

dem ser em relação ao "*World*", "*Adjoining*" ou a um dos CS do *body*. Quando um CS/CG qualquer é referenciado em relação ao *World* as suas coordenadas cartesianas são absolutas; quando são referenciados por *Adjoining* as suas coordenadas de localização são baseadas num outro *body* anexo ao CS/CG do *body* atual; quando são referenciados por um CS arbitrário as suas coordenadas são referentes a este CS arbitrário, sendo que, por sua vez, o mesmo pode possuir outro tipo de posicionamento.

Não há uma regra específica para se definir o sistema de posicionamento do *body*, porém nota-se que se todos os *bodys* do robô forem referenciados uns aos outros e o principal, o corpo, possuir o bloco *ground* como referência, todo o sistema pode ser transportado para qualquer outro referencial sem haver qualquer implicação com os sistemas de posicionamento do restante do robô. Conforme a

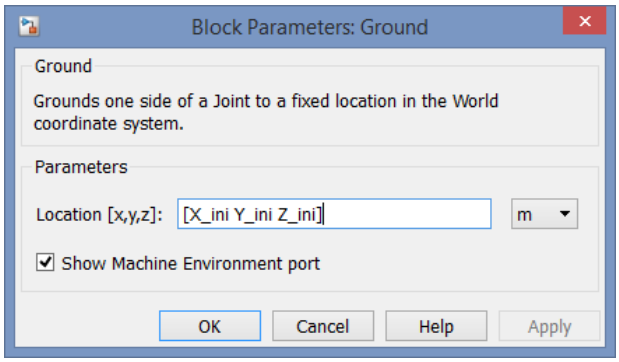



Figura 3.8: Parâmetros do bloco *Ground*.

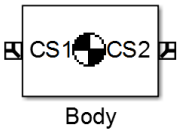



Figura 3.9: Bloco de caracterização de um corpo, *Body*.

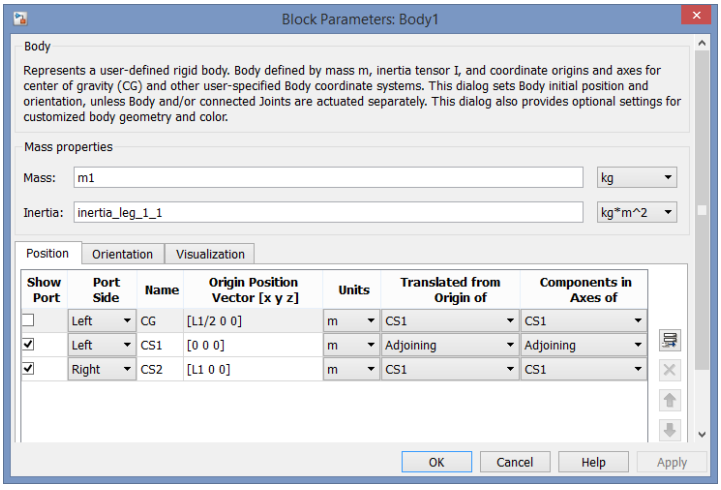


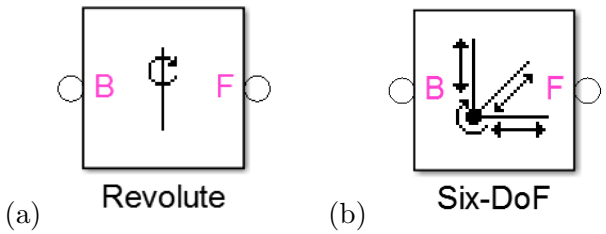

Figura 3.10: Parâmetros do bloco *Body*.

Figura 3.9, para cada bloco *body* deve haver ao menos um CS que o conecte a um bloco de junta.

## 3.4 Revolute e Six-DoF

Para haver o movimento de determinado *body*, deve haver um bloco de junta anexo a ele. Dos diversos blocos de juntas que o SimMechanics™ possui, apenas dois serão utilizados ao longo do projeto, o bloco *Revolute* e o *Six-DoF*, mostrados na Figura 3.11.

O bloco *revolute* é a abstração de uma junta rotacional ao qual pode atuar sobre eixo $X$, $Y$ ou $Z$. Seu sentido de rotação obedece à regra da mão direita, considerando a rotação no sentido anti-horário positiva. Para inverter o sentido de rotação de um eixo basta multiplicar o eixo por $-1$. Todos os blocos de junta possuem dois terminais denominados *Base* e *Follower*. O terminal *Base* deve ser conectado no *body* antecessor e o terminal *Follower* deve ser conectado no *body* sucessor.




(a) (b)

Figura 3.11: Blocos de juntas rotacional ou *Revolute* (a) e rotacional/prismático ou *Six-DoF* (b).

Para o *revolute* a rotação também é relativa, podendo ser em relação ao "*world*", "*base*" ou "*follower*". Para haver a rotação é necessário que haja um bloco que forneça tal movimento e é para este fim que há o parâmetro *Number of sensors/actuators ports*, como o mostrado na Figura 3.12, que gera terminais adicionais no bloco *revolute* para interagir com um atuador de junta, com um sensor de junta, ou com ambos; posteriormente serão abordadas tais escolhas.

O bloco de junta *Six-DoF* é uma junta que possui três DOF de translação e três DOF de rotação. Este bloco tem como função permitir o livre movimento do corpo do hexápode em todas as direções e simular a postura do robô perante o sistema de controlo das seis pernas do hexápode. Dessa forma, ele deve ser conectado entre o bloco *ground* (*base*) e o CG do bloco *body* (*follower*), que representará o corpo do hexápode.

## 3.5 Joint Actuator e Body Actuator

Um atuador é um elemento que transforma o sinal de controlo num sinal de movimento mecânico, quer seja linear ou angular. Os blocos *Joint Actuator* e

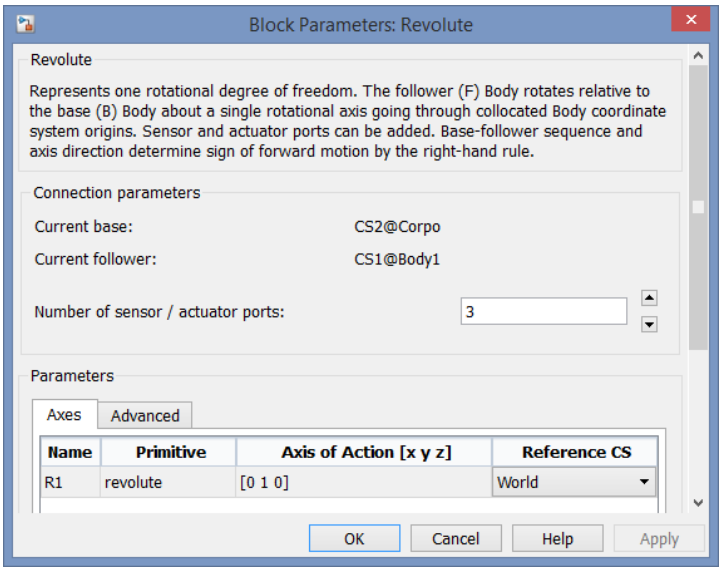
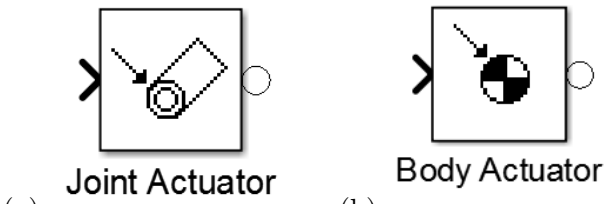



Figura 3.12: Parâmetros do bloco *Revolute*.

*Body Actuator* são os atuadores que o SimMechanics™ utiliza para aplicar força ou binário numa junta ou num corpo, como se apresenta na Figura 3.13.




(a) (b)

Figura 3.13: Blocos atuadores de junta *Joint Actuator* (a) e de corpo *Body Actuator* (b).

Quando conectado a um bloco de junta, o *joint actuator* automaticamente se adequa ao seu modo de movimento, isto é, prismático ou rotacional. Como a modelação do hexápode considerará todas as juntas de suas pernas rotacionais a Figura 3.14 exibe os parâmetros adequados para uma junta rotacional. Seu movimento dá-se de duas formas, ou aplica-se diretamente o binário desejado ou então deve ser fornecida a posição, velocidade e aceleração angulares para que se compute o movimento resultante. Quando se aplica o binário, através da dinâmica direta, a junta desloca-se para a posição relativa ao binário fornecido; por outro lado, quando se aplica um sinal vetorial contendo a posição, velocidade e aceleração para a junta, através da dinâmica inversa, o atuador fornece um binário relativo aos valores do sinal vetorial. Desse modo, com a aplicação de

um sistema de controlo sobre a posição desejada é possível adotar um bloco *joint actuator* configurado para aplicar binário sobre a junta rotacional e assim evitar o extensivo cálculo da dinâmica inversa.

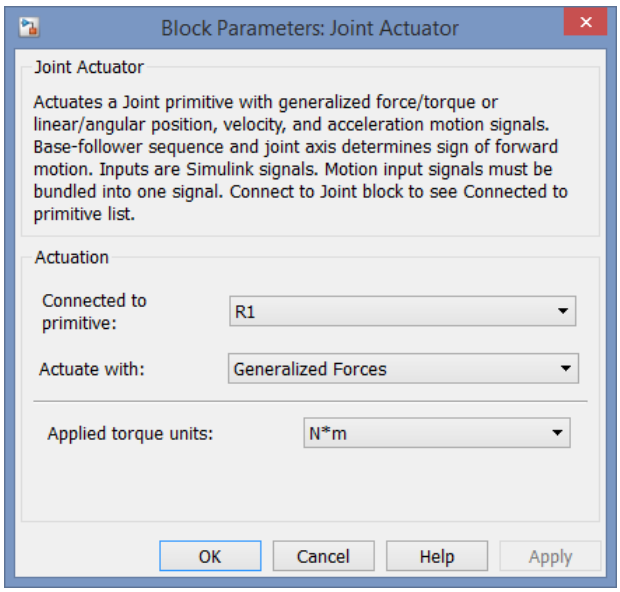



Figura 3.14: Parâmetros do bloco *Joint Actuator*.

Para desempenhar a abstração das forças de contacto com o solo em três dimensões será empregue o bloco *body actuator*. Este bloco é ajustado para aplicar força aceitando um sinal vetorial como entrada e a atuação pode ser em relação ao *body* local ou ao referencial inercial; para este trabalho foi selecionado *Absolute (world)*.

## 3.6 Joint Sensor e Body Sensor

Assim como há atuadores para os blocos de juntas e de corpos, há também os seus respetivos sensores. Os blocos sensores *Joint Sensor* e *Body Sensor* possuem a função de extrair os valores de posição, velocidade e aceleração das juntas e corpos, respetivamente. Esses blocos são essenciais para a simulação, pois o bloco *joint sensor* será utilizado como realimentação para o sistema de controlo das juntas do robô, expressando o *feedback* da junta. Já o bloco *body sensor* será utilizado para extrair as posições das extremidades das pernas do robô que, posteriormente, serão computadas nas equações do sistema mola-amortecedor da modelação do solo. Das juntas extrai-se o seu respetivo ângulo (em graus) e o binário computado (em $N \cdot m$) e para os corpos extrai-se os valores que se pretende analisar na simulação, conforme as Figuras 3.15 e 3.16.

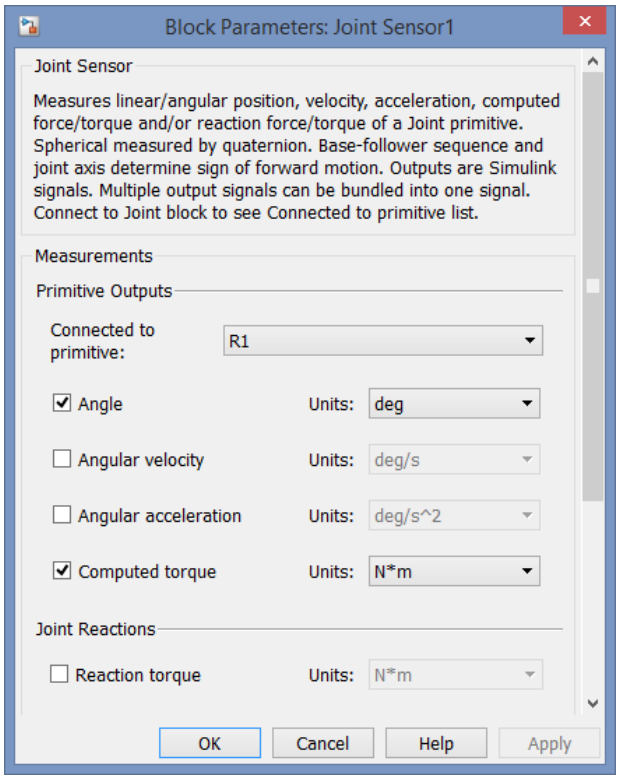



Figura 3.15: Parâmetros do bloco *Joint Sensor*.

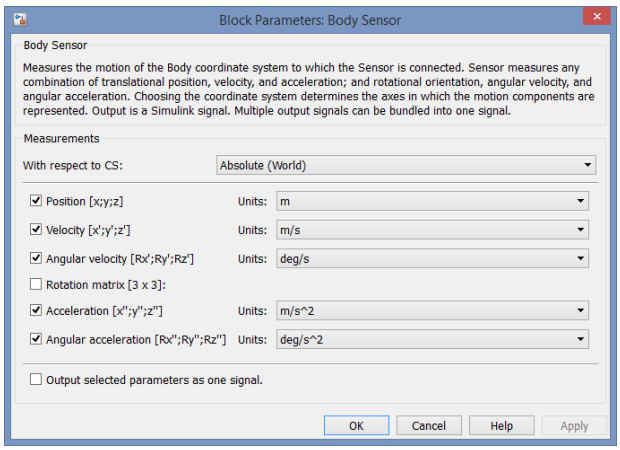



Figura 3.16: Parâmetros do bloco *Joint Body*.

## 3.7 Conclusão

Os conceitos básicos abordados, passo a passo, neste capítulo visam instruir, definir, configurar e implementar de forma correta as ferramentas do SimMechanics™

para um posterior êxito nas análises da simulação.

# Capítulo 4

# Modelação do robô hexápode

*O mecanismo primordial em que a robótica se sustenta consiste na modelagem do sistema. É fundamental ter o conhecimento matemático do sistema, pois é através dele que os fenómenos físicos reais são reconstruidos e considerados no sistema. Desta forma, um robô, quer seja fixo ou móvel, possui um conjunto de equações que representam as suas ações temporais. Restritamente aos robôs móveis, as equações relacionadas com a questão da mobilidade do robô são afetadas diretamente, variando-se consoante ao tipo de locomoção (rodas, lagartas, pernas).*

O processo de modelação de um sistema robótico segue uma sequência ordenada de tarefas, são elas:

- definir um nível de abstração do modelo robótico real para um modelo básico;
- desenvolver o estudo cinemático;
- efetuar o planeamento da trajetória;
- desenvolver o estudo dinâmico;
- implementar o sistema de controlo.

Após todos os critérios acima serem realizados, o robô está apto a ser simulado num ambiente de simulação ou experimentado num ambiente real. A seguir serão abordados cada um dos item definidos acima.

## 4.1 Aproximação do modelo do robô

A modelação inicia-se com a idealização do sistema. O robô disponível para o presente trabalho é visto na Figura 4.1 e foi construído pelo autor num período anterior ao desenvolvimento deste trabalho, possuindo uma montagem caseira e priorizando componentes de fácil aquisição. Trata-se de um robô hexápode com 3 DOF rotacionais por perna que permitem o deslocamento lateral do robô, ampliando a sua mobilidade. Assim, após a sua modelação, a sua simulação poderá mostrar eventuais reparos e otimizações visando os parâmetros ideais para um hexápode com tais dimensões.

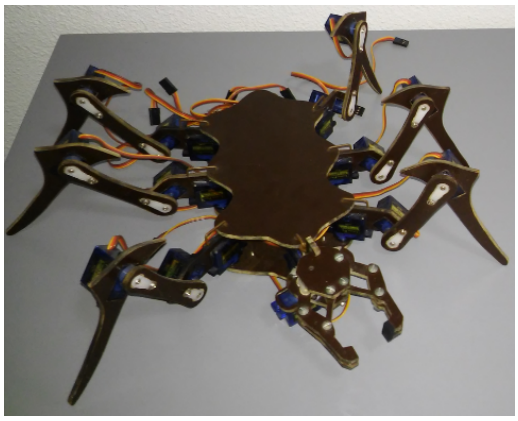


Figura 4.1: Robô hexápode desenvolvido pelo autor.

A primeira etapa da modelação dá-se pela virtualização do robô da Figura 4.1 através do programa de computador SolidWorks®, gerando a Figura 4.2. Este programa permite desenvolver uma réplica do robô real, considerando as suas dimensões e massa reais. Deve-se notar que, embora seja feita uma réplica do robô real, sua construção é de origem artesanal, ocasionando certos desvios entre o robô real e o virtual.

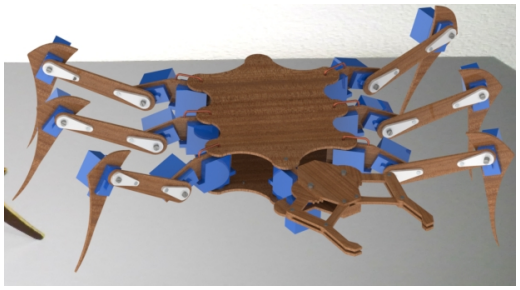


Figura 4.2: Robô hexápode projetado no programa SolidWorks®.

Após o SolidWorks® fornecer os dados de massa, momento de inércia, distâncias e posicionamentos, armazenam-se os dados num *script* do MatLab®. De seguida é realizada mais uma aproximação, dessa vez extrai-se apenas as características primordiais do robô, originando um modelo mais simples, como se mostra na Figura 4.3 . O robô da Figura 4.1 possui, além das suas seis pernas, um *gripper* com dois DOF. Este *gripper* a priori será descartado, pois o objetivo atual está na implementação da sua locomoção e não numa utilização específica, logo o *gripper* foi removido de todos os modelos.

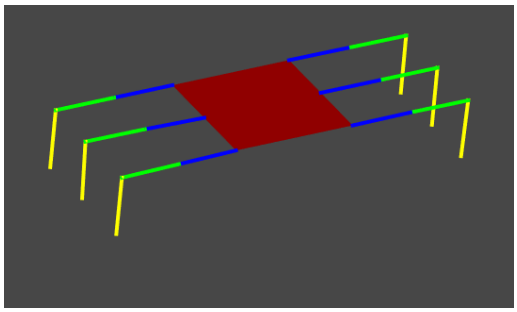


Figura 4.3: Robô hexápode com representação mais simples, considerado ideal.

Para fins de simplicidade e melhor compreensão, o modelo da Figura 4.3 será chamado de modelo ideal. Seu nome deve-se apenas por considerar os dados necessários para os cálculos cinemáticos e de planeamento de trajetórias.

A Figura 4.3 expõe as pernas do robô como segmentos de retas coloridos. Estes segmentos representam os elos das pernas do hexápode. Estes serão os elos $L_1$, $L_2$ e $L_3$ das variáveis de junta $q_1$, $q_2$ e $q_3$. As variáveis de junta são os ângulos $\theta_1$, $\theta_2$ e $\theta_3$, respetivamente, e o posicionamento inicial dos elos e juntas de cada perna do robô é visto na Figura 4.3.

## 4.2 Cinemática

A mecânica é a área da física que estuda o movimento dos corpos, objetos ou partículas. Já a cinemática trata-se de um conjunto dentro da mecânica, que apenas considera o movimento do corpo, desprezando as forças que originam tal movimento. A cinemática é utilizada na robótica para a obtenção da posição e orientação desejada dos diversos componentes integrantes de um robô.

Para descrever o movimento de um robô multi-pernas o estudo cinemático deve ser dividido em duas etapas, ou instantes, chamados de fase de transferência (*swing phase*) e fase de suporte (*stance phase*).

A primeira etapa do estudo, *swing phase*, analisa o movimento durante a fase de transferência da perna. Nesta etapa considera-se a anca como referencial

inercial fixo ao corpo do robô e os pés em movimento consoante a trajetória planeada pelo utilizador. A Figura 4.4 representa os elos e as juntas da perna do robô. Embora seja possível extrair as coordenadas de posição dos pés através da convenção de Denavit-Hartenberg, achou-se viável efetuar os cálculo tendo por base as relações trigonométricas, uma vez que se trata de um sistema com 3 DOF.

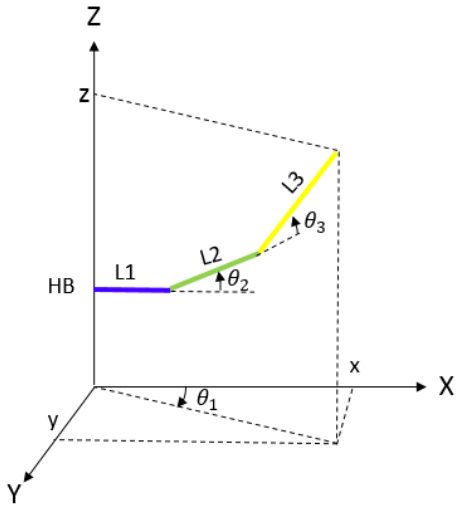



Figura 4.4: Configuração da perna do robô para a realização dos cálculos cinemáticos.

Da Figura 4.4 é possível concluir que a cinemática direta de uma perna do hexápode resulta em:

$$\begin{cases} X = (L_1 + L_2\cos(\theta_2) + L_3\cos(\theta_2+\theta_3))\cos(\theta_1) \\ Y = (L_1 + L_2\cos(\theta_2) + L_3\cos(\theta_2+\theta_3))\sin(\theta_1) \\ Z = H_B + L_2\sin(\theta_2) + L_3\sin(\theta_2+\theta_3) \end{cases} \tag{4.1}$$

Onde:

- $L_1$ é o comprimento do elo que representa a coxa da perna do hexápode;
- $L_2$ é o comprimento do elo que representa o fémur da perna do hexápode;
- $L_3$ é o comprimento do elo que representa a tíbia da perna do hexápode;
- $H_B$ representa a altura do corpo do robô ao solo;
- $\theta_1$ é o valor angular da junta rotacional que representa a anca do hexápode;
- $\theta_2$ é o valor angular da junta rotacional que representa o joelho do hexápode;

- $\theta_3$ é o valor angular da junta rotacional que representa o tornozelo do hexápode.

Todas as pernas do robô são simétricas, logo todas possuem as mesmas expressões da cinemática direta e inversa. Ainda na fase de transferência realiza-se o estudo da cinemática inversa. Pela mesma Figura 4.4 extrai-se $\theta_1$, pois:

$$\tan\theta_1 = \frac{y}{x} \tag{4.2}$$

Logo,

$$\theta_1 = atan_2\left(\frac{y}{x}\right) \tag{4.3}$$

Onde $atan_2$ é a função que retorna o valor correto do par ordenado $(x,y)$, pois caso o valor da tangente seja positivo não se pode distinguir se o ângulo é do primeiro ou terceiro quadrante e se for negativo, se é do segundo ou quarto quadrante; assim, $atan_2$ pode ser representado por:

$$atan_2(y,x) = \begin{cases} \arctan(\frac{y}{x}) & x > 0 \\ \arctan(\frac{y}{x}) + \pi & y \geq 0, x < 0 \\ \arctan(\frac{y}{x}) - \pi & y < 0, x < 0 \\ +\frac{\pi}{2} & y > 0, x = 0 \\ -\frac{\pi}{2} & y < 0, x = 0 \\ indeterminado & y = 0, x = 0 \end{cases} \tag{4.4}$$

Para encontrar a expressão matemática que representa $\theta_3$ é necessário primeiramente encontrar a hipotenusa do triângulo retângulo formado conforme a Figura 4.5.

Identificando-se o triângulo retângulo com catetos $\sqrt{x^2+y^2} - L_1$ e $z - H_B$ encontra-se a sua hipotenusa $\rho$ pelo teorema de Pitágoras:

$$\rho = \sqrt{\left(\sqrt{x^2+y^2} - L_1\right)^2 + (z - H_B)^2} \tag{4.5}$$

Baseando-se na lei dos cossenos, a relação entre os elos $L_2$, $L_3$ e a hipotenusa $\rho$ com o ângulo $\varphi$ é dada por:

$$\rho^2 = {L_2}^2 + {L_3}^2 - 2L_2L_3\cos\varphi \tag{4.6}$$

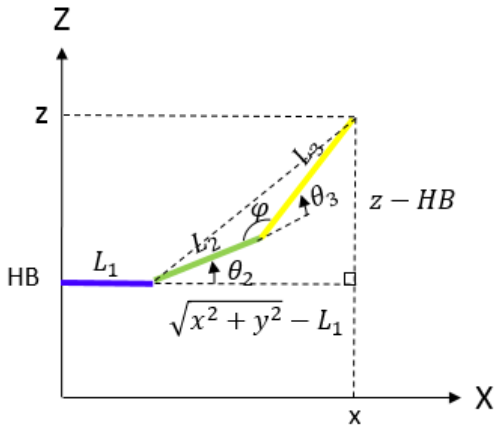



Figura 4.5: Configuração de uma perna do hexápode para os cálculos da cinemática inversa, na fase de transferência.

O ângulo $\varphi$ é complementar, isto é:

$$\varphi = 180^\circ - \theta_3 \tag{4.7}$$

Substituindo $\varphi$ na equação 4.6, simplificando a equação e aplicando-se a identidade trigonométrica do $\cos(180^\circ - \theta_3)$ tem-se:

$$\left(\sqrt{x^2+y^2} - L_1\right)^2 + (z - H_B)^2 = {L_2}^2 + {L_3}^2 - 2L_2L_3[\cos(180^\circ)\cos(\theta_3) + \sin(180^\circ)\sin(\theta_3)] \tag{4.8}$$

$$x^2 + y^2 - 2L_1\sqrt{x^2+y^2} + {L_1}^2 + (z - H_B)^2 = {L_2}^2 + {L_3}^2 + 2L_2L_3\cos\theta_3 \tag{4.9}$$

$$\cos(\theta_3) = \frac{x^2 + y^2 - 2L_1\sqrt{x^2+y^2} + (z - H_B)^2 + {L_1}^2 - {L_2}^2 - {L_3}^2}{2L_2L_3} \tag{4.10}$$

Portanto,

$$\theta_3 = \arccos\left(\frac{x^2 + y^2 - 2L_1\sqrt{x^2+y^2} + (z - H_B)^2 + {L_1}^2 - {L_2}^2 - {L_3}^2}{2L_2L_3}\right) \tag{4.11}$$

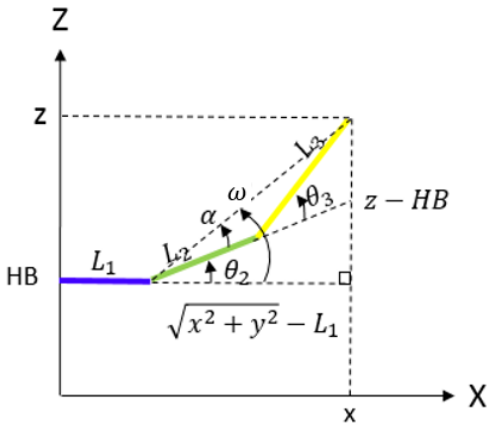



Figura 4.6: Configuração adequada da perna do robô para obtenção de $\theta_2$.

Após ser encontrada a expressão para $\theta_3$ é possível abstrair $\theta_2$ através da análise da Figura 4.6.

O ângulo $\theta_2$ é o resultado da diferença entre:

$$\theta_2 = \omega - \alpha \tag{4.12}$$

Sendo

$$\omega = atan_2\left(\frac{z - H_B}{\sqrt{x^2 + y^2} - L_1}\right) \tag{4.13}$$

e

$$\alpha = atan_2\left(\frac{L_3 \sin(\theta_3)}{L_2 + L_3 \cos(\theta_3)}\right) \tag{4.14}$$

Logo,

$$\theta_2 = atan_2\left(\frac{z - H_B}{\sqrt{x^2 + y^2} - L_1}\right) - atan_2\left(\frac{L_3 \sin(\theta_3)}{L_2 + L_3 \cos(\theta_3)}\right) \tag{4.15}$$

Na cinemática direta não há nenhuma restrição ou problema, originando sempre em única coordenada para o ponto escolhido. Isso já não se verifica na cinemática inversa que possui várias representações para um mesmo ponto, pois há

ângulos que se complementam dando origens a redundâncias. Outro problema encontrado na cinemática inversa está no que é denominado de singularidades. Singularidades são definidas como os lugares onde o determinante da matriz jacobiana do modelo geométrico direto se anula [27], sendo um problema matemático e não físico da perna do robô. A singularidade ocorre quando a perna se encontra totalmente esticada, totalmente retraída ou sobre uma reta que passa verticalmente pela origem (referencial inercial) do sistema. Portanto deve-se evitar que a trajetória da perna passe por esses pontos.

Finalizando a primeira etapa, a análise da fase de transferência, é realizado o estudo cinemático para a segunda etapa, a fase de suporte. A segunda etapa é caracterizada por considerar o pé do robô fixo ao solo, oferecendo sustentação e propulsão para o robô se locomover. Assim, o referencial inercial passa a ser os pés do hexápode e a anca, ou quadril, passa a possuir a trajetória planeada pelo utilizador. Nesta fase a perna é vista como um braço manipulador com a sua base fixa na coordenada do instante de colocação dos pés do robô ao solo, e assim, a coordenada $Z$ desse braço manipulador já expressa a altura do corpo do robô ao solo. As equações da cinemática direta permanecem inalteráveis, porém a cinemática inversa para a fase de suporte possui $H_B = 0$, isto é:

$$\theta_1 = atan_2\left(\frac{y}{x}\right) \tag{4.16}$$

$$\theta_2 = atan_2\left(\frac{z}{\sqrt{x^2+y^2}-L_1}\right) - atan_2\left(\frac{L_3\sin(\theta_3)}{L_2+L_3\cos(\theta_3)}\right) \tag{4.17}$$

$$\theta_3 = \arccos\left(\frac{x^2+y^2-2L_1\sqrt{x^2+y^2}+{L_1}^2+z^2-{L_2}^2-{L_3}^2}{2L_2L_3}\right) \tag{4.18}$$

## 4.3 Planeamento de trajetórias

A posição temporal das extremidades das pernas e do centro de gravidade do corpo do hexápode é feito através do planeamento de trajetórias. Nesta secção descreve-se qual o tipo de trajetória que se deve implementar. Há várias formas de se planear trajetórias, sendo que elas podem ser inspiradas:

- em seres biológicos;
- em funções matemáticas;
- ou em índices de otimização.

Trajetórias inspiradas em seres biológicos buscam simplesmente imitar o movimento real do ser vivo [21]. Há também maneiras de planear trajetórias considerando índices de otimização, neste caso busca-se, por exemplo, uma trajetória com maior eficiência energética [13]. Por sua vez, trajetórias inspiradas em funções matemáticas visam descrever exatamente a função de interesse como retas, parábolas, funções sinusoidais, dentre outras.

Um função, em particular, é comummente empregue no planeamento de trajetória de robôs multi-pernas, a ciclóide [28, 29, 13]. Segundo Silva, que provou que a ciclóide minimiza os correspondentes binários nas juntas melhorando o seguimento das trajetórias das ancas e dos pés [13], as coordenadas cartesianas para uma ciclóide planar são:

$$\begin{bmatrix} x_f \\ y_f \\ z_f \end{bmatrix} = \begin{bmatrix} cte \\ V_L(t - \frac{T\sin(\frac{2\pi t}{T})}{2\pi}) \\ \frac{F_c}{2}(1 - \cos(\frac{2\pi t}{T})) \end{bmatrix} \tag{4.19}$$

Sendo $T$ o período de um ciclo completo de movimento dos pés do hexápode, $V_L$ a velocidade de deslocamento da perna e $F_c$ a altura máxima alcançada pelas extremidades de seus pés.

A velocidade da perna representada na equação 4.19 deve ser relacionada com a velocidade de deslocamento frontal do corpo, de forma que o comprimento percorrido num ciclo completo, tanto por cada perna, quanto pelo corpo, sejam iguais a $L_s$, comprimento do passo. A relação entre $V_L$ e $V_F$ é dada por:

$$\begin{aligned} V_F &= \tfrac{Ls}{T} \\ V_F &= (1-\beta)V_L \\ \therefore V_L &= \tfrac{1}{(1-\beta)}V_F \end{aligned} \tag{4.20}$$

Onde $\beta$ é o fator de ocupação, vide subsecção 4.3.1. Pela simplicidade, eficiência e por permitir ajustes em $V_L$ e $F_c$ esta será a trajetória adotada para o planeamento da perna durante a fase de transferência.

Porém, na fase de suporte as pernas do hexápode devem servir de sustentação e propulsão do corpo; desse modo, considera-se o corpo em movimento retilíneo uniforme deslocando-se ao longo do eixo - $Y$, sendo este sentido adotado no algoritmo do MatLab® apenas devido ao SimMechanics™ considerar este sentido como positivo no momento da simulação. Para representar tal movimento as coordenadas da anca devem ser:

$$\begin{bmatrix} x_h \\ y_h \\ z_h \end{bmatrix} = \begin{bmatrix} 0 \\ -V_F t \\ HB \end{bmatrix} \tag{4.21}$$

Como resultado do planeamento num ciclo completo, as Figuras 4.7 e 4.8 mostram uma perna descrevendo uma curva representando a cicloide e uma reta representando o movimento de seu corpo.

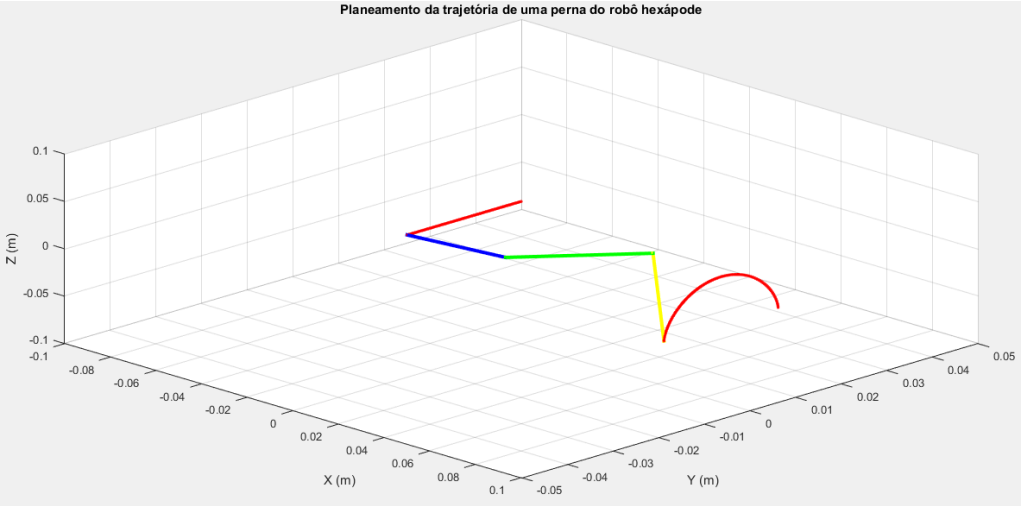



Figura 4.7: Planeamento da trajetória do movimento do robô ao longo de um ciclo (fase de transferência e suporte), vista isométrica.

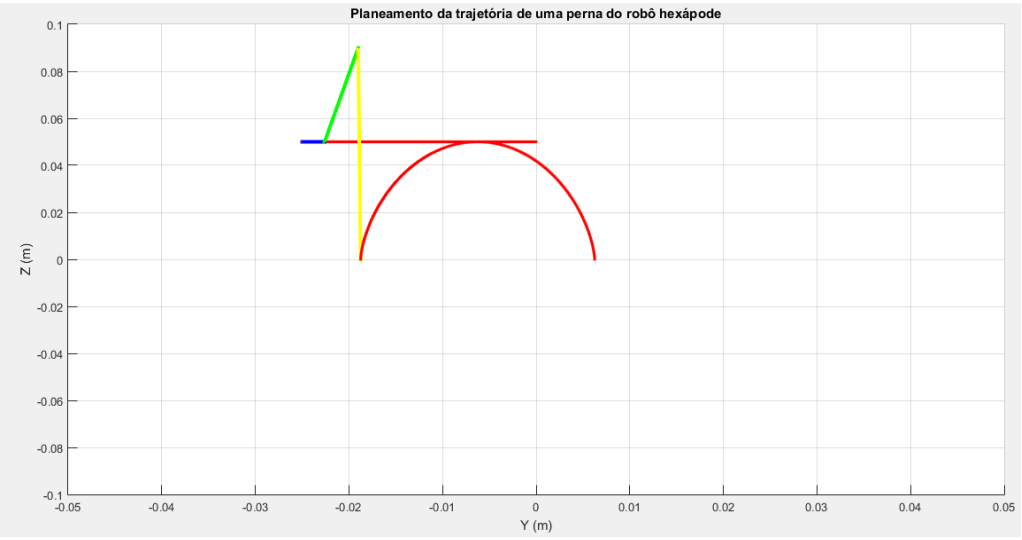



Figura 4.8: Planeamento da trajetória do movimento do robô na fase de transferência e suporte, vista no plano *Y-Z*.

Embora o estudo cinemático tenha sido dividido em duas etapas, fase de transferência e fase de suporte, para elaborar a trajetória de um sistema robótico

multi-pernas deve-se considerar o movimento do corpo constante, tanto na fase de transferência, quanto na fase de suporte.

Outra característica fundamental para a elaboração da trajetória das pernas é o desvio das trajetórias dos pés relativamente às ancas. Este desvio nada mais é do que o sincronismo que deve haver entre os pés e as ancas. Quando os pés do robô, na fase de transferência, atingirem o ponto mais alto da ciclóide, $F_c$, a anca correspondente a esta perna deve possuir a mesma posição em $Y$, com se pode ver na Figura 4.8. Esta correção deve ser feita para que o corpo fique corretamente alinhado com os pés, para cada padrão de locomoção. Este ajuste será mencionado na subsecção seguinte.

### 4.3.1 Padrões de locomoção

Um padrão de locomoção é uma sequência de movimentos, tanto dos membros, quanto do corpo, de um robô ao longo de um ciclo completo de locomoção [13]. Cada animal possui um padrão de locomoção distinto e o mesmo animal pode alterar o seu padrão de locomoção devido à necessidade de se alterar a velocidade da marcha. Existem diferentes estilos de marcha, sendo as mais comuns nos hexápodes a onda metacronal, tetrápode e trípode, variando-se sua velocidade de deslocamento para lenta, média e rápida, respetivamente [30]. A diferença básica entre essas marchas são o uso de uma, duas ou três pernas simultaneamente na fase de transição. Para implementar os padrões de locomoção num robô hexápode é necessário compreender alguns conceitos.

O fator de ocupação, ou *duty factor*, de um robô multi-pernas é a fração temporal em que uma perna permanece na fase de suporte relativamente ao período $T$ [11], em outras palavras:

$$\beta_i = \frac{Tempo\ de\ suporte\ da\ perna\ i}{T}, \qquad i = 1, ..., 6 \tag{4.22}$$

A fase $\phi_i$ de uma perna é a razão entre o instante em que ela entra em contacto com o solo e o período $T$, isto é:

$$\phi_i = \frac{Instante\ em\ que\ a\ perna\ i\ toca\ no\ solo}{T} \tag{4.23}$$

O tempo de aterragem da perna $i$ é medido a partir do toque da perna 1; por conseguinte, tem-se $\phi_1 = 0$ para qualquer marcha. Essas variáveis são utilizadas para a construção de um diagrama de marchas, que é um diagrama que indica o instante em que a perna $i$ inicia a fase de transferência ao longo do período $T$, com tempo normalizado, como se mostra na Figura 4.9.

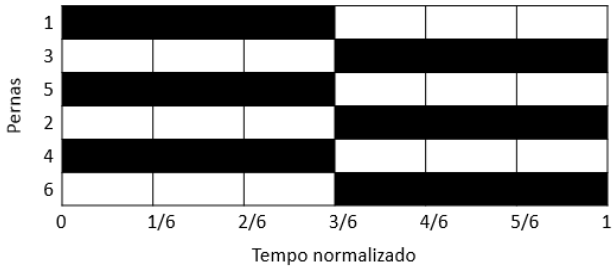



Figura 4.9: Diagrama da marcha trípode, para $\beta = 1/2$. A cor preta indica a fase de transferência e branca a fase de suporte.

A ordem em que as pernas se dispõem ao longo do corpo do robô possui um padrão [11]: as pernas ímpares situam-se do lado esquerdo enquanto as pernas pares se localizam do lado direito, sendo o sentido de movimento ao longo do eixo $+Z$, como verificado na Figura 4.10.

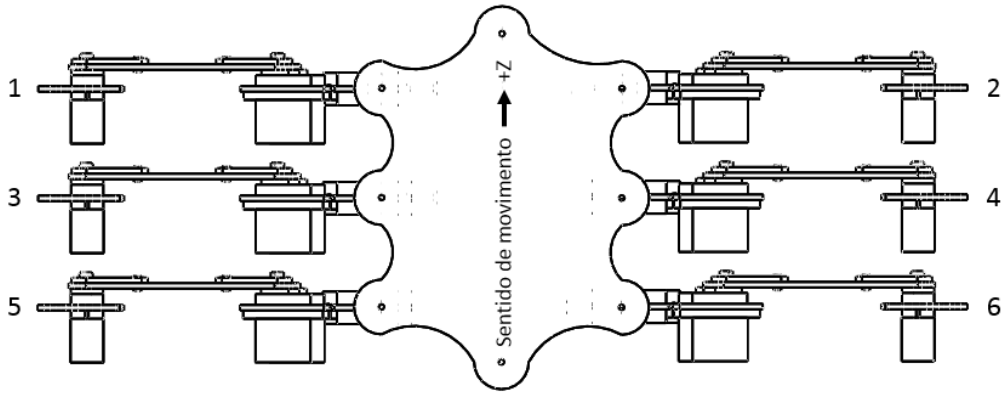



Figura 4.10: Numeração das pernas do hexápode.

Assim, para a marcha trípode da Figura 4.9 as pernas 1, 4 e 5 iniciam juntas a fase de transferência, enquanto as pernas 2, 3 e 6 só irão iniciar esta fase após 50% do período, pois elas iniciam juntas a fase de suporte e possuem $\phi_i$ de 0,5. Esse tipo de marcha é caracterizada por atingir as maiores velocidades frontais. Seja $L_s$ o comprimento do passo da perna: ao fim de um ciclo cada perna do hexápode ter-se-á deslocado exatamente $L_s$, movimentando seu corpo $L_s$.

A marcha tetrápode é caraterizada por manter apenas duas pernas simultaneamente na fase de transferência. A Figura 4.11 representa uma das possíveis combinações de pernas para este tipo de marcha. Neste caso $\beta$ não poderá ser igual ao da marcha trípode, pois o $\beta$ mínimo admissível deve ser de $\beta = 4/6$, caso contrário o robô irá cair ao chão.

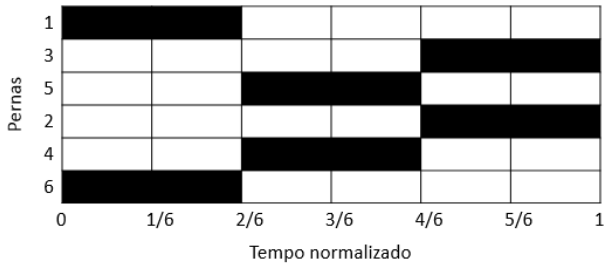



Figura 4.11: Diagrama da marcha tetrápode, para $\beta = 4/6$.

Movimentando apenas uma perna de cada vez, a marcha onda metacronal alcança uma velocidade máxima inferior ao das duas marchas mencionadas, como se pode ver na Figura 4.12. Para desenvolver tal marcha deve-se ter $\beta = 5/6$. Portanto o fator de ocupação pode ser utilizado para distinguir se o robô está a caminhar, com $\beta > 1/2$, ou a correr,com $\beta < 1/2$ [11].

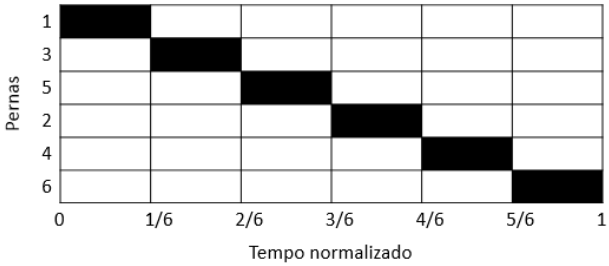



Figura 4.12: Diagrama da marcha em onda metacronal, para $\beta = 5/6$.

Após a descrição do fator de ocupação, a correção do desvio das trajetórias dos pés relativamente às ancas pode ser estabelecida, resultando em:

$$\varepsilon = L_s \frac{\beta}{2} \tag{4.24}$$

O ajuste do desvio foi feito com um decréscimo de $\varepsilon$ na coordenada $Y$ da função da cicloide, pois $\beta/2$ multiplicado por $L_s$ resulta exatamente no deslocamento necessário para que ambos, cicloide e corpo, coincidam no instante que é atingido $F_c$ na fase de transferência.

## 4.4 Dinâmica

A Dinâmica é a parte da mecânica que se dedica ao estudo dos movimentos considerando as forças e/ou binários que o originam [31]. Após realizar os cálculos

da cinemática e do planeamento de trajetórias, resultando na evolução das posições, velocidades e acelerações das juntas entre os elos de cada perna do robô, considera-se a posteriori o binário necessário do atuador de junta para que o mesmo possa seguir a manter o movimento. Assim como há cinemática direta e inversa, há também dinâmica direta e inversa.

A dinâmica direta calcula as posições, velocidades e acelerações das juntas do robô a partir das forças/binários produzidas pelos atuadores das juntas. A dinâmica inversa calcula as forças/binários resultantes das posições, velocidades e acelerações das juntas do robô.

Por se tratar de uma simulação realizada em SimMechanics™, não há necessidade do cálculo da dinâmica inversa, mesmo porque conforme o exposto no capítulo anterior o SimMechanics™ efetua automaticamente este cálculo.

### 4.4.1 Forças de contacto com o solo em três dimensões

Assim como a modelação de uma aeronave é feita baseando-se na aerodinâmica e a modelação de um submarino baseando-se na hidrodinâmica, a modelação de robôs multi-pernas baseia-se em parte nas forças de contacto dos pés do robô com o solo. Ao estabelecer um contacto com o solo, os pés do robô exercem forças de sustentação devido ao solo exprimir forças de igual magnitude, porém de sentido contrário. Em casos onde a modelação da perna do robô considera apenas um movimento planar [25] consideram-se apenas as forças de contacto em duas direções, normal e tangencial.

Como o SimMechanics™ considera as direções vertical, lateral e frontal sobre os eixos $+Z$, $+X$ e $+Y$ respetivamente, as componentes da força de reação serão referentes a tais coordenadas e o planeamento de trajetórias será convertido conforme o Anexo A. A força normal é exercida ao longo do eixo $+Y$, normal à superfície, proporcionando a sustentação das pernas. A força tangencial é a força exercida ao longo do eixo $+Z$, que impõe movimento frontal ao robô. Robôs multi-pernas que possuem 3 DOF rotacionais por perna, conforme o robô hexápode modelado neste trabalho, necessitam de forças de contacto atuando nas três direções, logo também há uma força tangencial ao longo do eixo $+X$, que impõe movimento lateral ao robô.

Esta modelagem é representada por um sistema mola-amortecedor que visa caracterizar a complacência da superfície do solo. De acordo com Castro, no momento em que o pé toca no solo a componente elástica da força normal da equação 4.25 é quase nula em contraste com a componente de amortecimento, que é positiva e muito superior, e no momento em que o pé tende a erguer-se a componente de amortecimento torna-se negativa devido à velocidade positiva dos

pés, uma força que tende a puxar os pés [25].

$$\begin{aligned} F_y &= -K_{yF}(\Delta_{iyF0}) - B_{yF}\dot{\Delta}_{iyF0} \\ \Delta_{iyF0} &= y_{iF} - y_{iF0} \\ \dot{\Delta}_{iyF0} &= \dot{y}_{iF} - \dot{y}_{iF0} \end{aligned} \tag{4.25}$$

Onde:

- $y_{iF}$ é a coordenada atual do pé $i$;
- $y_{iF0}$ é a coordenada de colocação do pé $i$ no solo;
- $K_{yF}$ é o coeficiente de elasticidade da mola do sistema mola-amortecedor;
- $B_{yF}$ é o coeficiente de atrito viscoso do amortecedor do sistema mola-amortecedor.

O autor soluciona este problema reescrevendo a equação, resultando na equação 4.26. Na forma de um sistema mola-amortecedor não linear o contacto com o solo torna-se mais suave, concedendo um incremento gradativo das forças, onde o expoente $\nu$ é um fator que depende das propriedades do terreno, variando entre zero e um.

$$F_{iyF} = -K_{yF}\Delta_{iyF0} - B_{yF}(-\Delta_{iyF0})^{\nu}\dot{\Delta}_{iyF0} \tag{4.26}$$

Ajustando a força normal conforme realizou Castro e estendendo a conclusão para as forças tangenciais consoante Silva propõem, as forças de contacto com o solo adotadas nesta obra serão [25, 13]:

$$\begin{bmatrix} F_{ixF} \\ F_{iyF} \\ F_{izF} \end{bmatrix} = \begin{bmatrix} -K_{xF}\Delta_{ixF0} - B_{xF}(-\Delta_{ixF0})\dot{\Delta}_{ixF0} \\ -K_{yF}\Delta_{iyF0} - B_{yF}(-\Delta_{iyF0})^{\nu}\dot{\Delta}_{iyF0} \\ -K_{zF}\Delta_{izF0} - B_{zF}(-\Delta_{izF0})\dot{\Delta}_{izF0} \end{bmatrix} \tag{4.27}$$

## 4.5 Sistema de controlo

O sistema de controlo é o agente responsável por garantir que os valores angulares das juntas rotacionais sejam atingidos, de forma que as trajetórias planeadas, tanto para a anca, quanto para as pernas do robô, possuam as coordenadas pretendidas no espaço operacional. A diferença entre o valor desejado $\theta_d$ e o valor real $\theta_r$ da junta rotacional é chamada de erro e há várias formas de reduzi-lo ou

eliminá-lo. Quando o ganho do controlador é proporcional ao erro o controlador é do tipo P, quando é proporcional ao integral do erro o controlador é do tipo I e quando é proporcional à taxa de variação do erro o controlador é do tipo D.

Após empregar um controlador PID num robô hexápode com 2 DOF por perna, Carvalho percebeu que mesmo que o termo integral sirva para anular o erro em regime permanente, para qualquer valor de ganho integral I não há grandes variações de resposta, uma vez que o pé ao entrar em contacto com o solo produz uma perturbação na trajetória do robô [15]. Castro também constatou, após várias simulações em SimMechanics™, que ao adotar um controlador PD o binário exercido nas juntas foi reduzido, validando os resultados da simulação, pois com um controlador PID os atuadores produziam elevados binários provocando perfurações indesejadas no solo [25]. Por esse motivo, optou-se pelo controlador PD para o sistema de controlo da Figura 4.13.

Para controlar uma perna do robô é considerado o controlo individual das suas juntas, onde cada junta possui uma malha de controlo de acordo com a Figura 4.13. O planeamento de trajetória age como sinal de entrada para o sistema de

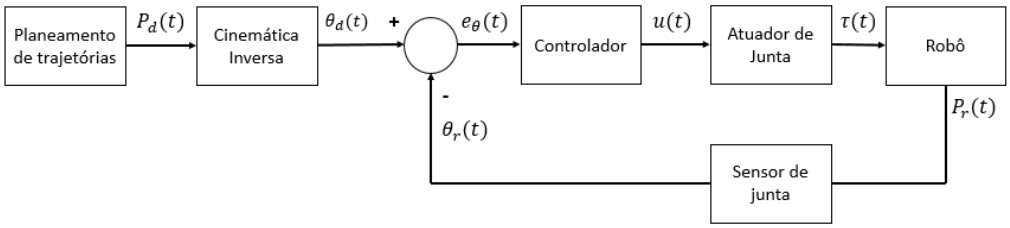



Figura 4.13: Sistema de controlo adotado para cada junta de cada perna do robô hexápode.

controlo, porém o controlo é feito no espaço das juntas e o planeamento é efetuado no espaço operacional, logo é inserido entre eles o algoritmo da cinemática inversa. Em seguida faz-se a soma do valor angular desejado com o valor angular da realimentação negativa, sendo este valor oriundo do sensor de junta. Como resultado tem-se o erro $e(t)$ como entrada do controlador PD. O sinal da lei de controle, sinal de saída do controlador, deve passar pelo atuador de junta, pois este age de forma adversa ao sensor de junta, gerando o binário para a junta da perna controlada.

## 4.6 Conclusão

Ao findar este capítulo nota-se uma relação hierárquica entre cada etapa da modelação, partindo dos critérios de análise do estudo cinemático, o planeamento das curvas executadas pelas pernas e corpo do hexápode, os modos de locomoção,

as forças de contacto com o solo, até chegar ao sistema de controlo completo das juntas. Todas as equações deduzidas e critérios requisitados até este momento servirão como base de dados para que possam ser aplicados na simulação do robô hexápode descrito na secção 4.1.

# Capítulo 5

# Simulação em SimMechanics™

*Tendo o conhecimento das equações matemáticas, fenómenos físicos e da arquitetura de controlo que envolvem o sistema robótico a ser simulado, todos descritos no capítulo anterior, e reunindo as ferramentas básicas de simulação mecânica do SimMechanics™, detalhadas no Capítulo 3, este capítulo visa reproduzir a locomoção de um robô hexápode em SimMechanics™.*

## 5.1 Representação da perna do robô

A caracterização de uma perna do robô pode ser feita interligando um bloco *body* a uma *revolute joint*. Dado que o robô possui três elos por perna, serão necessários três blocos *body* ligados através de três blocos *revolute joint*. A Figura 5.1 exibe o esquema de ligação dos blocos formadores da primeira perna do hexápode; nela, cada bloco *revolute joint* possui um bloco de *initial conditions* em anexo, admitindo uma configuração inicial das juntas.

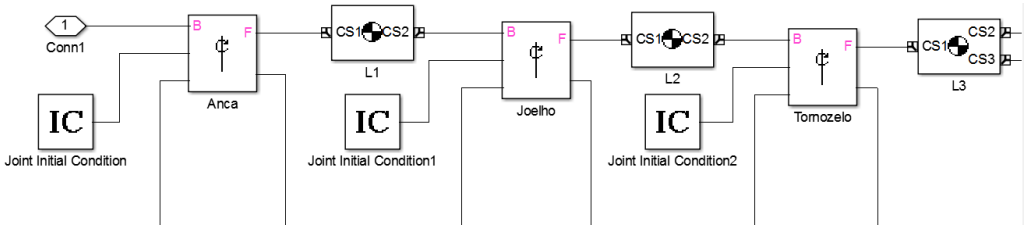



Figura 5.1: Representação dos elos e juntas presentes na perna do robô.

O bloco L1 corresponde à coxa, L2 ao fémur e L3 à tíbia da perna do hexápode e as suas configurações são vistas nas Figuras 5.2, 5.3 e 5.4 , respetivamente,

sendo que cada bloco possui variáveis do *workspace* do MatLab® representadas de forma a permitir uma alteração posterior dos comprimentos dos elos.

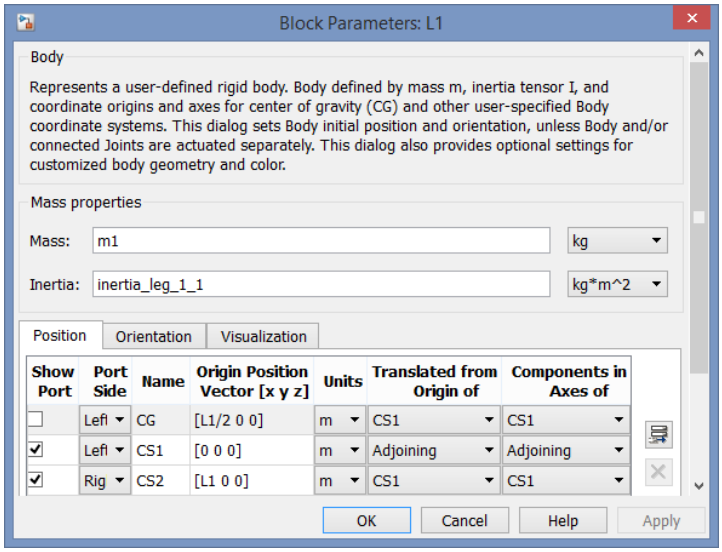



Figura 5.2: Parâmetros do elo L1 da perna do robô.

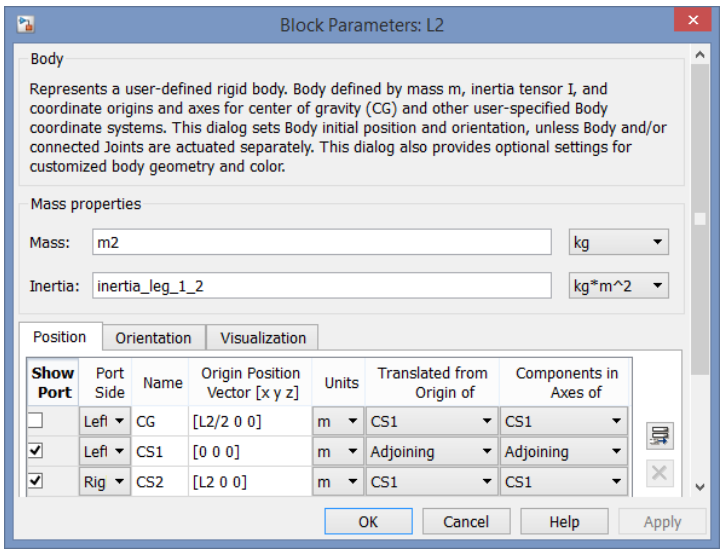



Figura 5.3: Parâmetros do elo L2 da perna do robô.

Os sinais que não são vistos na Figura 5.1 são os sinais dos controladores e das forças de reação (no caso do elo L3) e o conector *Conn1* é responsável por unir a perna ao corpo. Para as pernas ímpares a anca gira no sentido horário do eixo $Y$ e o joelho e o tornozelo giram no sentido anti-horário do eixo $Z$; já para as pernas

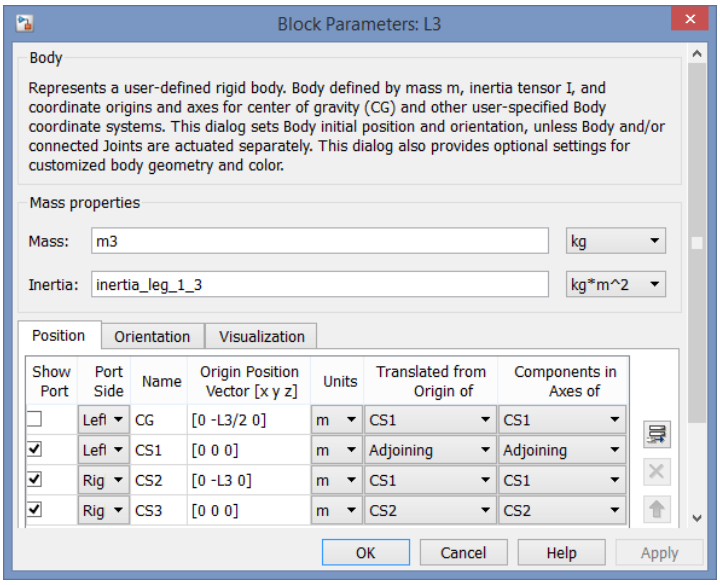



Figura 5.4: Parâmetros do elo L3 da perna do robô.

pares todos as juntas giram no sentido oposto. Devido à riqueza de detalhes do modelo desenvolvido no SolidWorks®, onde várias peças do robô se conectavam através de sistemas de posicionamento, no arquivo de montagem, sua simulação tornou-se muito lenta inviabilizando a sua utilização. Porém, o modelo simples da Figura 4.3 permanece com os valores de massa importados do SolidWorks® mas não com o valores dos momentos de inércia, por diferirem em suas formas geométricas. Desse modo, o código desenvolvido e exposto no anexo B além de definir o valor de cada variável da simulação, efetua o cálculo do momento de inércia em relação aos eixos $X$, $Y$ e $Z$ considerando os elos como cilindros e o corpo como um prisma retangular.

## 5.2 Representação das forças de contacto

Responsável por aplicar as forças de reação aos pés do hexápode, o sistema de perceção e atuação das forças nas três direções da Figura 5.5 é efetuado sobre o *body* L3. Os CS2 e CS3 do *body* L3 são os sistemas de coordenadas da extremidade da perna e possuem coordenadas idênticas. O CS3 é o ponto onde é feita a leitura da posição, velocidade linear, aceleração linear, velocidade angular e aceleração angular dos pés através do sensor de corpo *Body Sensor*. A posição e velocidade linear são utilizadas como variáveis de entrada para os blocos que calculam as forças normal e de atrito, e restantes variáveis são adicionais servindo apenas para sua posterior visualização.

Os valores obtidos pelo *Body Sensor* são vetoriais, isto é, possuem componen-

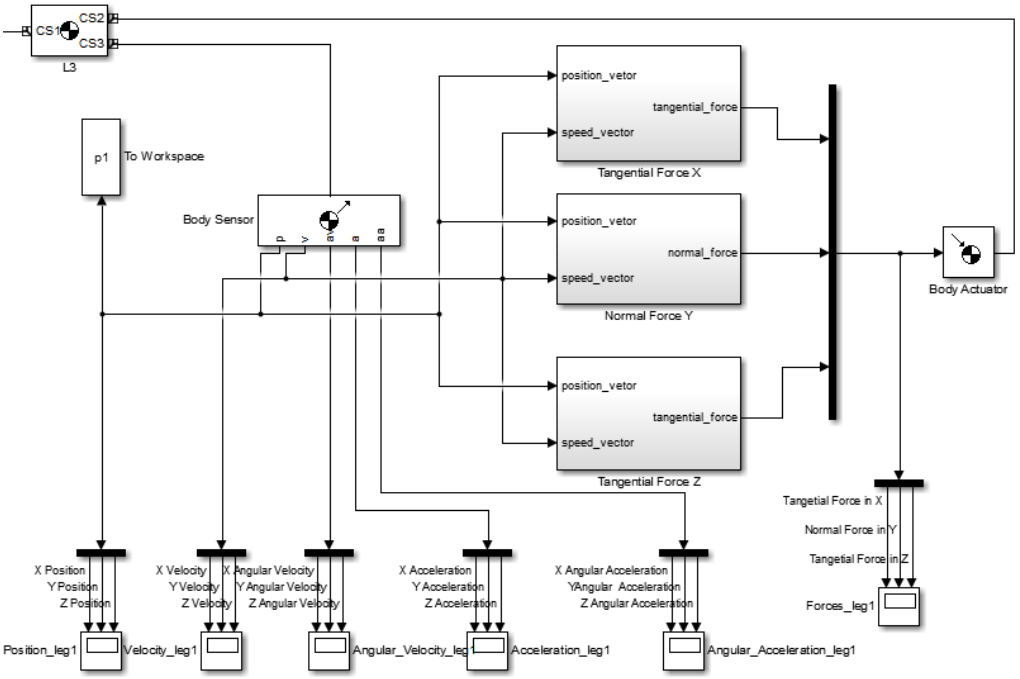



Figura 5.5: Representação das forças de contacto que atuam em L3.

tes sobre os eixos $X$, $Y$ e $Z$. Os blocos *Tangential Force X*, *Normal Force Y* e *Tangential Force Z*, descritos nas subsecções seguintes, fornecem como variáveis de saída as forças de reação em cada direção. Para computar essa força é utilizado um multiplexador de três entradas, seguido de um *Body Actuator*, permitindo que o valor numérico calculado por cada bloco de entrada do multiplexador seja convertido num sinal de força mecânica aplicado no mesmo local de contacto - é por esta razão que CS2 possui as mesmas coordenadas de CS3.

### 5.2.1 Força tangencial

Os blocos responsáveis por aplicar uma força tangencial na direção dos eixos $X$ e $Z$ são os blocos *Tangential Force X* e *Tangential Force Z*. Dentro de cada um desses blocos há um sistema de deteção do contacto dos pés com o solo, como o visto na Figura 5.6. A necessidade de haver este sistema, conforme concluem Castro e Woering, é devido ao facto de que as forças devem ser atuadas apenas quando o valor da posição do pé em análise for menor que zero [25, 29]. Para recriar este conceito utilizam-se os blocos de condição *IF* e *ELSE*: se a coordenada $Y$ da posição do pé em questão for menor do que zero é selecionado o bloco *Friction Force*, senão aplica-se força nula.

O bloco *Merge* combina as suas entradas para uma única linha de saída,

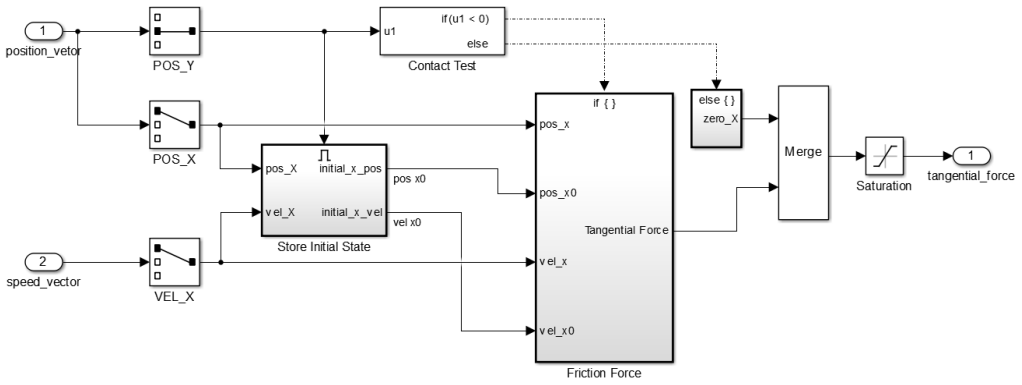



Figura 5.6: Sistema de deteção de colisão com o solo.

cujo valor, em qualquer instante, é equivalente à saída mais recente calculada pelos blocos de condição e o bloco *Saturation* permite limitar a magnitude da forca aplicada. A única diferença entre o bloco *Tangential Force X* e *Tangential Force Z* está nos seletores de sinais, responsáveis por extrair apenas a coordenada desejada dos vetores de posição e velocidade, sendo $X$ para a força tangencial na direção do eixo $X$ e $Z$ para a força tangencial na direção do eixo $Z$.

Para as componentes $F_x$ e $F_z$ da equação 4.27 poderem ser calculadas deve haver o cálculo da diferença entre a posição atual e posição inicial de contacto dos pés. Para manter a posição inicial do contacto memorizada utiliza-se a coordenada de posição em $Y$ como sinal de controlo do bloco *Storage Initial State*. Assim, quando o seu valor for positivo (pés sem contacto com o solo) o sinal de entrada do bloco *Storage Initial State* é transferido para a sua saída e quando o sinal de controlo for negativo (pés em contacto com o solo) manterá o último valor válido registado de seu sinal de entrada.

O conteúdo do bloco *Friction Force* visto na Figura 5.7 é um sistema que possui as entradas coeficiente de elasticidade da mola, diferença entre a posição atual e início de contacto do pé com o solo, coeficiente de atrito viscoso do amortecedor e diferença entre a velocidade linear do pé atual e no instante do contacto e como saída tem-se a função matemática das componentes $F_x$ e $F_z$ da equação 4.27.

Os coeficientes de elasticidade da mola e do atrito viscoso do amortecedor foram selecionados conforme propõem [25], aplicados para um terreno de argila compacta, permitindo a deformação do solo para uma posterior análise e concretização teórica, cujos valores são de 1302151 $Nm^{-1}$ e 3423 $Nsm^{-1}$, respetivamente.

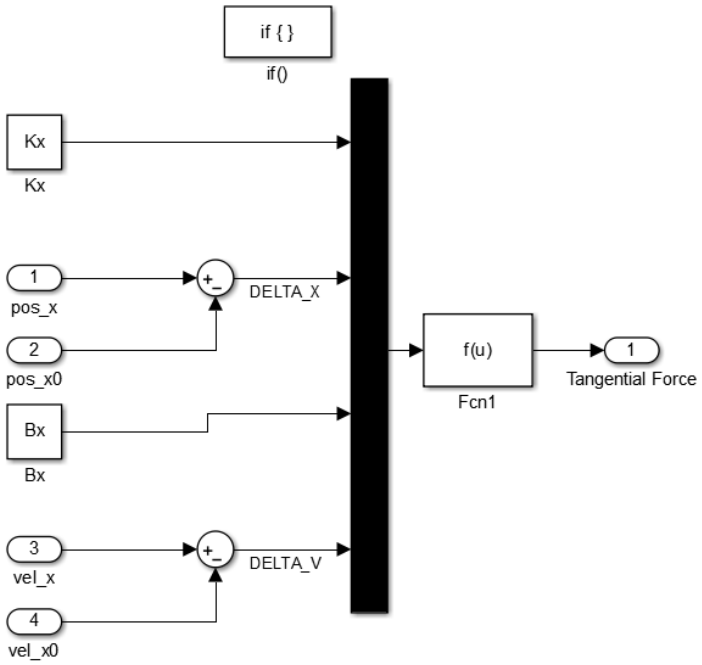



Figura 5.7: Representação da força de atrito na direção do eixo $X$.

### 5.2.2 Força normal

Para o caso da força normal o sistema possui a mesma arquitetura diferenciando-se apenas devido ao expoente $\nu$ de $F_y$ da equação 4.27. Este expoente permite ajustar, de acordo com estudos da mecânica dos solos, as características do solo com maior precisão possuindo valores entre zero e um. Relativamente ao princípio de funcionamento, a força normal só atua no *body* quando o CS3 possuir a coordenada de posição em $Y$ inferior a zero e os valores de $K_y$ e $B_y$ são respetivamente 170000 $Nm^{-1}$ e 270000 $Nsm^{-1}$.

## 5.3 Controlo das juntas do robô

O controlo das juntas do robô é feito de forma independente possuindo cada junta um controlador PID. O sinal de referência provém das variáveis do *workspace* do MatLab™, geradas após a execução do algoritmo de planeamento de trajetórias; desse modo é utilizado o bloco *From workspace* para cada variável de junta do robô, com se pode ver na Figura 5.8.

Dentro dos blocos $Controller_1$, $Controller_2$ e $Controller_3$ há um controlador PID, cuja a arquitetura se mostra na Figura 5.9, de tempo contínuo na forma paralela com os parâmetros $P$, $I$, $D$, $N$ também preenchidos por variáveis providas do *workspace*. A fórmula do controlador considerada pelo Simulink® é:

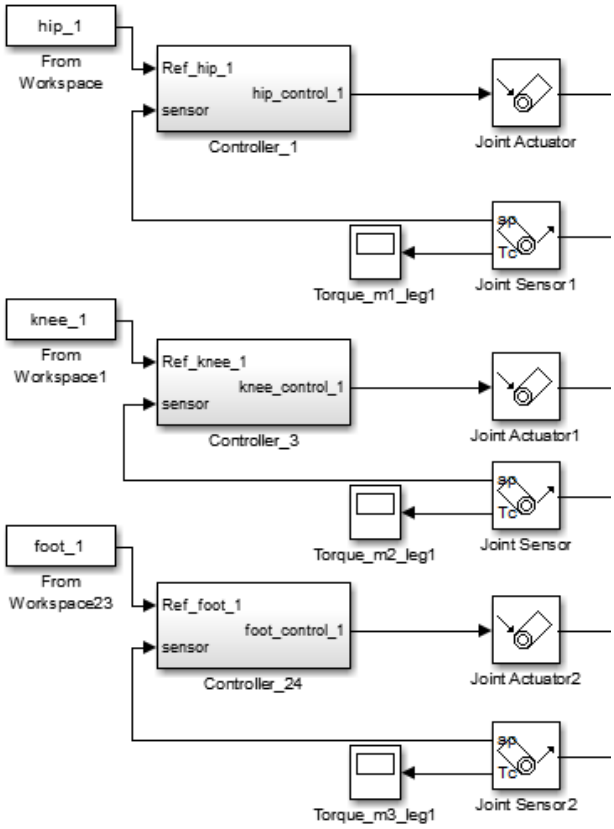



Figura 5.8: Sistema de controlo das juntas de cada perna do robô.

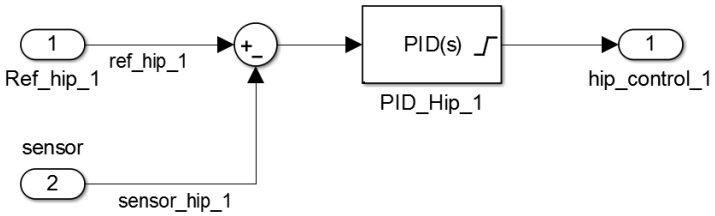



Figura 5.9: Bloco $Controller_1$, controlador da junta da anca da primeira perna do hexápode.

$$G(s) = P + I\frac{1}{s} + D\frac{N}{1 + N\frac{1}{s}} \tag{5.1}$$

Sendo $N$ o coeficiente do filtro e $I$ o ganho integral que, para o sistema de controlo PD, possui valor zero. A sintonia dos controladores inicialmente foi realizada baseando-se na técnica de linearização do sistema através da ferramenta

*Looptune*, disponibilizada pelo próprio Simulink®. Esta ferramenta permite o ajuste automático dos ganhos dos controladores PID do sistema completo do robô.

Apesar de proporcionar o cálculo automático dos ganhos dos controladores, tais ganhos possuem valores baixos e distintos, ocasionando a perda do controlo e maior tempo de execução. Por essa razão foi realizada a correção manual dos ganhos que tornam o sistema controlável para cada junta e replicado para o restante das pernas.

A Figura 5.8 exibe ainda os sinais de binários sendo reconhecidos pelos sensores de junta que posteriormente serão utilizados para o cálculo do binário mínimo dos motores.

## 5.4 Corpo do robô

Após o desenvolvimento e o controlo de cada uma das seis pernas do robô, as mesmas devem ser conectadas ao *body* que representa o corpo do robô. A Figura 5.10 apresenta a ligação das pernas, com os seus respetivos CS localizados ao longo do corpo do hexápode.

O movimento do corpo, conforme descrito na secção 3.4, é realizado através de uma junta rotacional de seis DOF, que, por sua vez, é anexa ao solo por meio do bloco *ground*. O CS7 possui as mesmas coordenadas do CG do corpo pois será utilizado para a análise do seu movimento.

Já a Figura 5.11 apresenta todos os cálculos feitos com base nas variáveis criadas e inicializadas no Anexo B de modo que a alteração dessas variáveis produza uma representação geométrica automática do robô em análise.

O bloco *To workspace* funciona de forma análoga ao bloco *From workspace*, isto é, ele envia os valores da posição vetorial do CG, reconhecidas pelo *Body Sensor*, para uma variável do *workspace* do MatLab®; neste caso a variável também se chama CG.

## 5.5 Solo

Embora não seja obrigatório a representação do solo no ambiente de simulação, uma vez que o sistema de deteção de colisão dos pés com o solo é consoante a posição dos pés na direção do eixo $+Y$, optou-se pela sua reprodução apenas com carácter simbólico. Logo, criou-se um bloco *body* com CS representando um prisma quadrangular com dimensões descritas no Anexo B. Conforme a Figura 5.12 para que o solo permaneça imóvel, a junta entre o *body* e o *ground* deve ser do tipo *weld*, unindo rigidamente o CG do *body* do solo à origem do sistema.

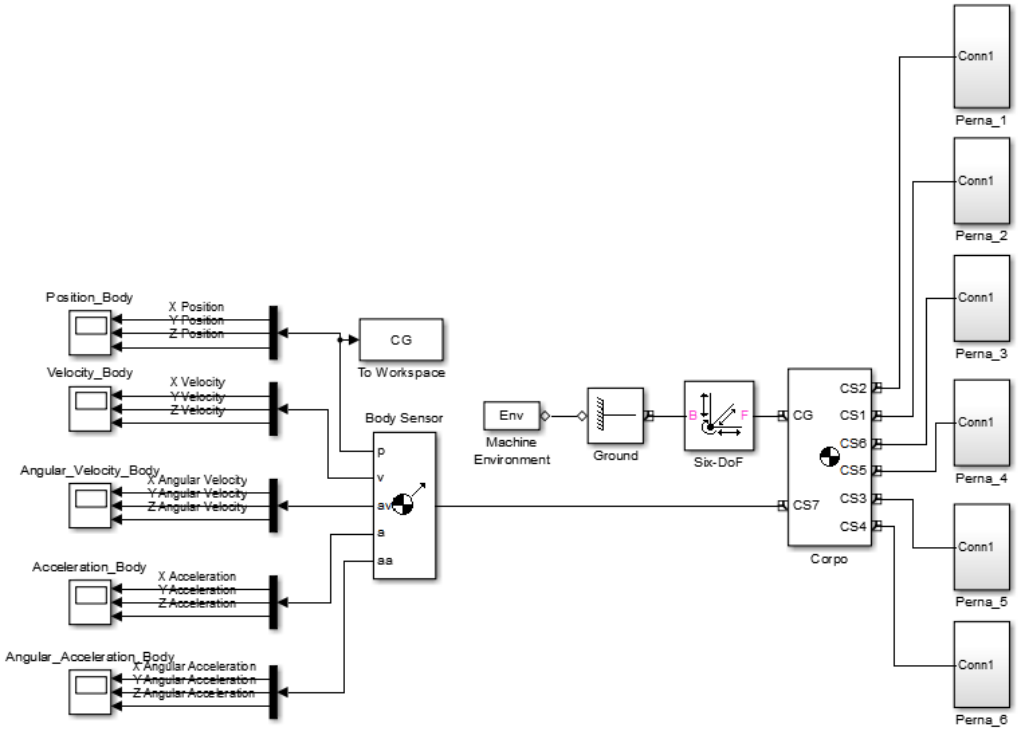



Figura 5.10: Representação da ligação das pernas ao corpo do robô.

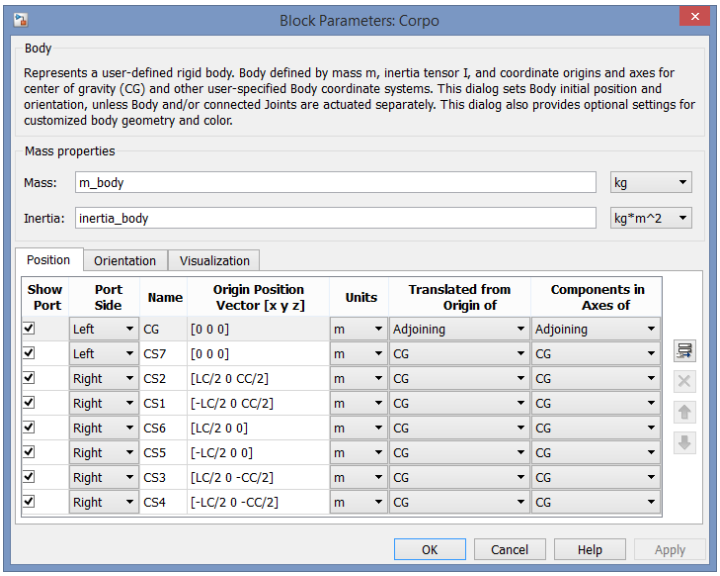



Figura 5.11: Parâmetros do corpo do robô.

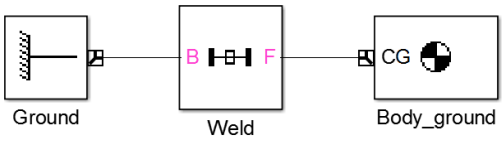



Figura 5.12: Representação do solo no ambiente de simulação.

## 5.6 Análise dos dados

Com todos os membros do robô devidamente construidos e interligados passou-se para a etapa de análise dos dados. Neste momento vê-se o robô como um sistema robótico que possui vários parâmetros configuráveis e vários pontos de inspeção tornando o uso de uma GUIDE proveitoso. Dessa forma, foi desenvolvida uma GUIDE que possibilita ao utilizador efetuar alterações na estrutura mecânica e nos parâmetros de simulação, bem como a visualização direta da posição, velocidade, aceleração e força aplicada a cada pé do hexápode, com se mostra no Anexo C.

Esta GUIDE provê também um algoritmo capaz de avaliar a estabilidade do tipo de locomoção adotado. O cálculo da estabilidade é baseado no número de amostras da posição do CG do robô que se situam dentro do polígono de suporte, e o seu valor é expresso em percentagem; assim, 50% de estabilidade significa dizer que metade das amostras recolhidas se situam dentro do polígono de suporte.

Para criar tal polígono é utilizada a função *fill(x,z)* do MatLab®, que recebe como parâmetros de entrada as coordenadas cartesianas $X$ e $Z$ de cada um dos seis pés do robô, e para determinar se o CG do robô se encontra dentro desse polígono é utilizada a função *inpolygon(xq,yq,xv,yv)*, que retorna uma estrutura com dois campos, um que indica se o CG está contido dentro do polígono e outro que indica se o CG está na extremidade do polígono, onde $xq$ e $yq$ são as coordenadas do CG e $xv$ e $yv$ são as coordenadas dos pés em contacto com o solo. Todas as amostras foram recolhidas através dos blocos *To Workspace*, guardadas no formato vetorial e com tempo de amostragem *sample_time*. A Figura 5.13 exemplifica para o caso da leitura da posição cartesiana do pé da perna 1 do robô; os demais pés possuem as mesmas configurações.

Este algoritmo foi utilizado para avaliar os três padrões de locomoção descritos na subsecção 4.3.1 e encontra-se no Anexo C. Devido aos gráficos de posição do CG do corpo, posição dos pés, velocidade frontal e forças de contacto serem semelhantes para os três tipos de marchas, cada secção seguinte analisa apenas um dos dados anteriormente mencionados.

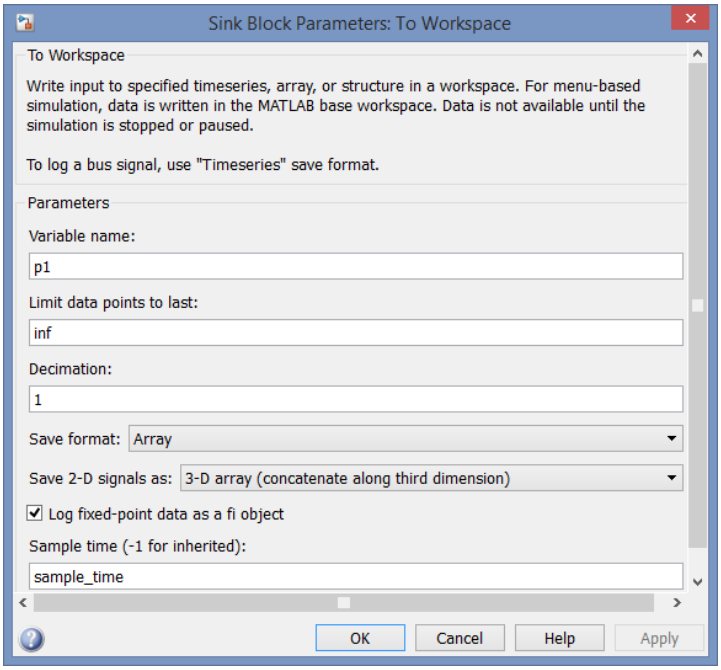



Figura 5.13: Configurações do bloco *To Workspace*.

## 5.7 Análise do padrão de locomoção onda metacronal

O primeiro padrão de locomoção simulado será o da Onda Metacronal. Este padrão possui a característica de haver somente a transferência de uma perna de cada vez. A Figura 5.14 exibe a sequência de locomoção sendo iniciada pela perna 1 que possui $\phi_1 = 0$.

De forma a aferir os três tipos de locomoção, os parâmetros de simulação foram:

- $F_c = 0{,}05$ m;
- $L_s = 0{,}025$ m;
- $H_B = 0{,}09$ m;
- $\beta = 0{,}83$;
- $T = 1$ s;
- número de ciclos = 4;
- Taxa de amostragem = 0,0005 s;
- $P = 1$;

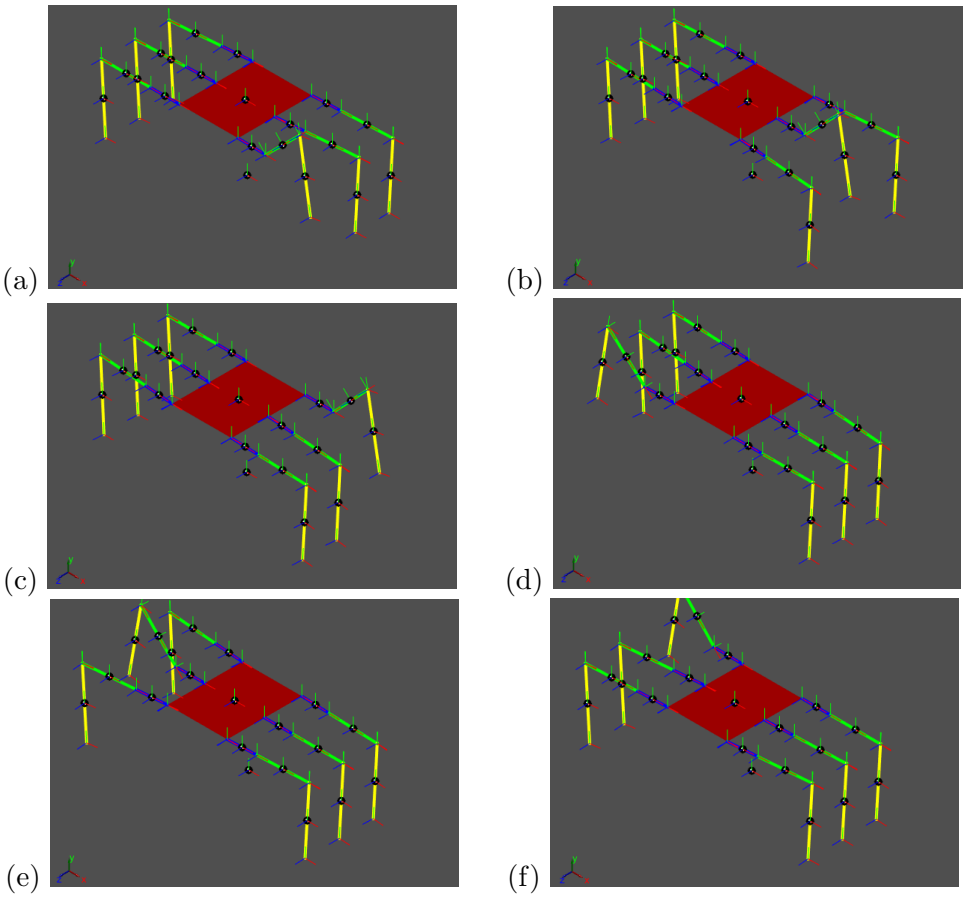



Figura 5.14: Sequência de movimentos das pernas 1-3-5-2-4-6, respetivos às sub-figuras (a)-(b)-(c)-(d)-(e)-(f), para a locomoção onda metacronal.

- $I = 0$;
- $D = 0{,}01$;
- $N = 100$.

No final da simulação é possível analisar a posição do CG do robô (ver Figura 5.15) onde a posição no eixo $+X$ indica o desvio lateral do robô da sua trajetória retilínea com amplitude máxima de 0,16 mm. Como o robô se desloca na direção +Z (na simulação), com comprimento do passo de 0,025 m, a cada 1 s durante quatro ciclos, a trajetória do CG em $+Z$ deve ser uma reta inclinada partindo da origem e finalizando em 0,1 m. Quanto aos dados referentes ao eixo $+Y$, nota-se uma oscilação periódica em torno de zero. Essa oscilação ocorre em virtude da movimentação do robô perante as forças de reação do solo.

Com a taxa de amostragem de 0,0005 s, durante 4 s são produzidas as 8000 amostras que serão analisadas pelo algoritmo do cálculo de estabilidade. Após

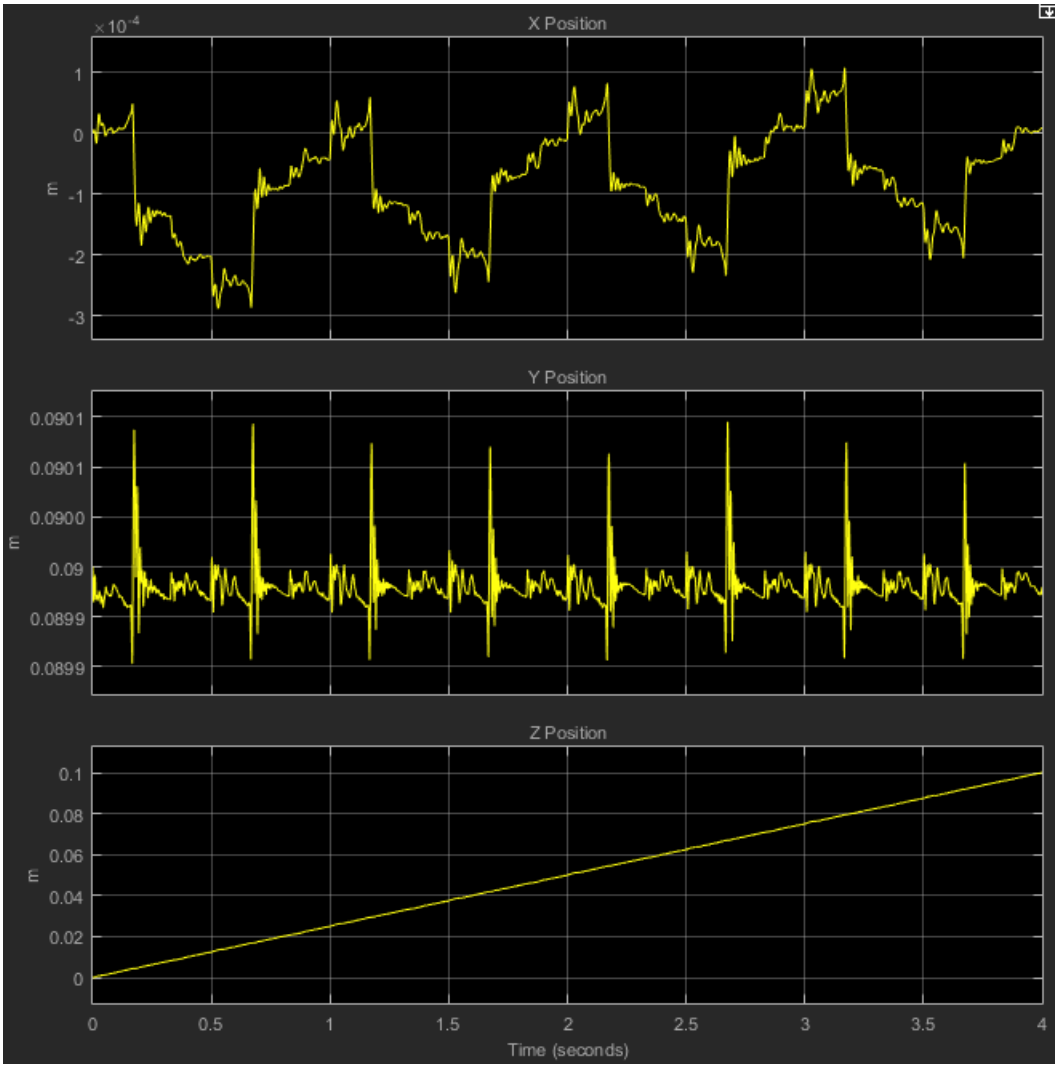



Figura 5.15: Desvios na trajetória do CG do corpo do robô.

todas as amostras serem processadas, os resultados da estabilidade para a locomoção em onda metacronal são:

Estável = 82,96%

Marginalmente estável = 0%

Instável = 17,03%

O regime "Marginalmente estável" dificilmente é processado devido à baixa probabilidade do CG se encontrar no limite da área do polígono de suporte no

instante de amostragem. O robô é consideravelmente leve, com massa de 0,360 kg, o que faz com que os seus pés penetrem pouco a superfície do solo (cerca de 6 $\mu m$) e assim qualquer oscilação do corpo pode elevar a posição dos pés para cima do solo e contribuir para o cálculo da instabilidade (neste caso de 17,03% do tempo da simulação).

Ainda para esta análise, é possível determinar qual deve ser o valor do binário mínimo que os servo motores devem apresentar para que o robô possa executar a locomoção. Através da função *max()* do MatLab® extrai-se o valor do binário máximo de todas as juntas do robô, armazenam-se os dados num vetor, extrai-se o valor do binário máximo desse vetor e, por fim, realiza a conversão de N·m (unidade de medida do SI) para kgf·cm, para o binário será expresso nesta unidade de medida apenas devido a esta ser requisitada comercialmente. Para a marcha em onda metacronal tem-se:

Binário mínimo do servo motor = 8,15 kgf·cm

Este deve ser o valor mínimo fornecido pelo servo motor de forma a evitar a incapacidade de execução da locomoção.

## 5.8 Análise do padrão de locomoção tetrápode

Caracterizada por movimentar duas pernas de cada vez, isto é, cada par de pernas com fases iguais, a marcha tetrápode foi completamente simulada de forma que a sua sequência de movimentos pode ser vista na Figura 5.16.

Para realizar a sequência de movimentos indicados na Figura 5.16 cada pé do robô teve de suportar as forças de reação do solo (como exemplo, vê-se na Figura 5.17 as forças que atuam sobre o pé da perna 1). A força tangencial na direção do eixo $X$ revela os instantes em que a perna 1 exerce força para permanecer imóvel (vide entre os instantes 0,3 s e 1 s). A força tangencial na direção do eixo $Z$ também possui os mesmos propósitos, de forma que os instantes coincidem diferindo apenas na magnitude do sinal, em virtude do momento de inércia na direção do eixo $Z$ ser maior. Pela análise da força normal nota-se a origem das oscilações do CG do corpo, bem como os quatro instantes em que o pé toca ao solo.

Para a marcha atual, o algoritmo de estabilidade resultou em:

Estável = 97,44 %

Marginalmente estável = 0 %

Instável = 2,56%

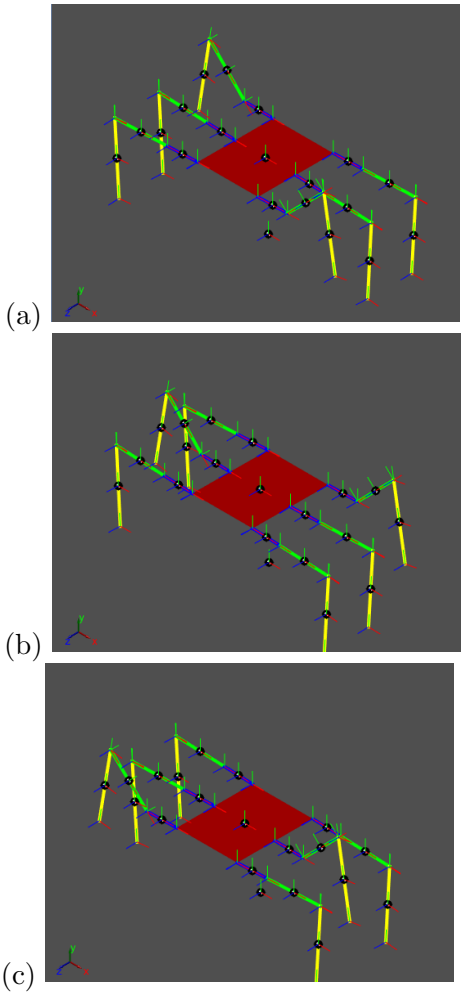



Figura 5.16: Sequência de movimentos das pernas (1,6)-(4,5)-(2,3), respetivos às subfiguras (a)-(b)-(c), para a locomoção tetrápode.

É constatado um aumento de 14,48 % na estabilidade e um decréscimo do binário mínimo exigido para os servo motores de 8,15 kgf·cm para:

$$\text{Binário mínimo do servo motor} = 3{,}36 \text{ kgf·cm}$$

O binário fornecido por um servo motor é aproximadamente proporcional ao preço e tamanho. Assim, para essa marcha específica, basta que o servo motor desenvolva o binário mínimo de modo a evitar um excessivo custo.

## 5.9 Análise do padrão de locomoção trípode

Conforme a subsecção 4.3.1 do capítulo anterior, é possível visualizar (ver Figura 5.18) através do ambiente dinâmico do SimMechanics™ o padrão de locomoção

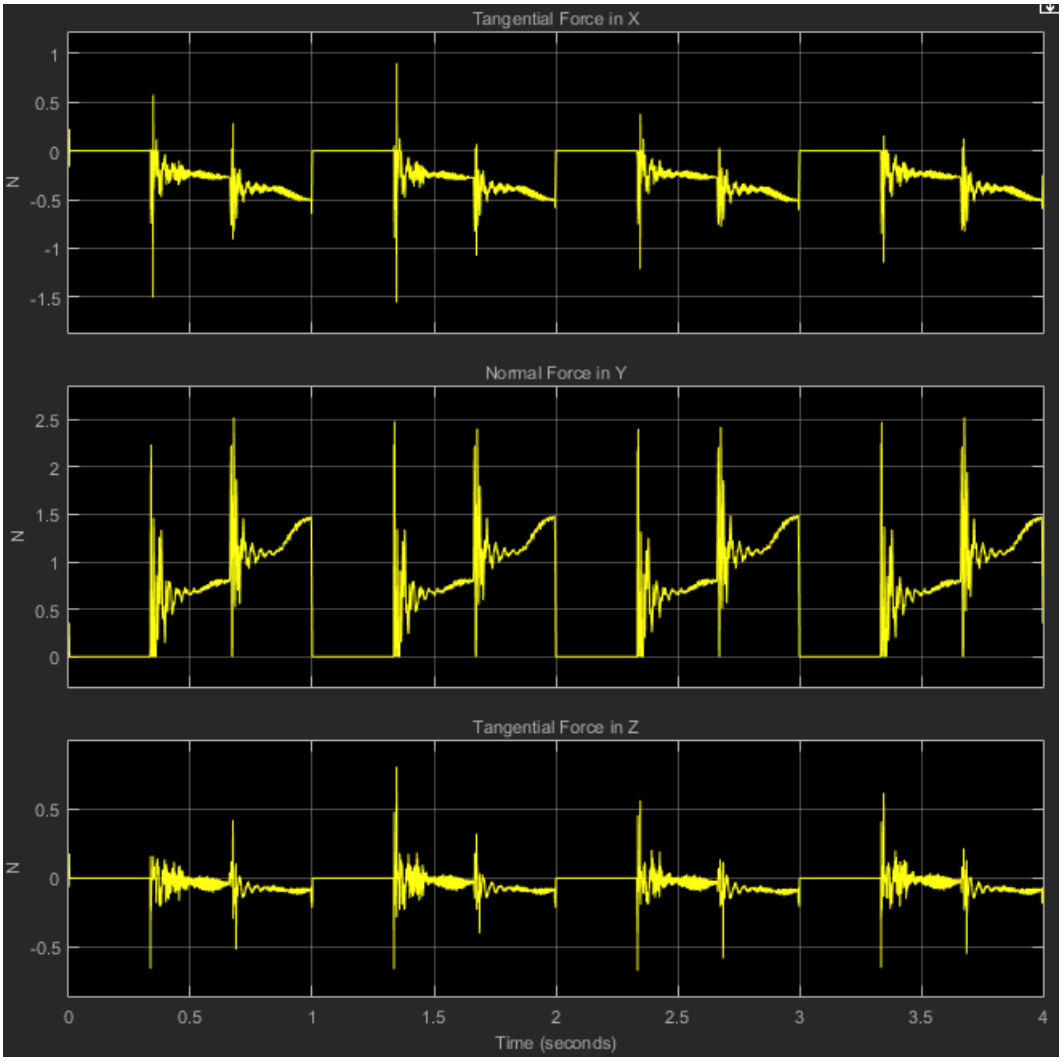



Figura 5.17: Forças de reação do contacto do pé da perna 1 com o solo em três dimensões.

trípode. As pernas 1, 4 e 5 movimentam-se com fase $\phi_{1,4,5} = 0$ e as pernas 2, 3 e 6 movimentam-se com fase $\phi_{2,3,6} = 0{,}5$.

Ao verificar as coordenadas da perna 1 nas três direções percebe-se o planeamento da trajetória modelado no Capítulo 3, Figura 5.19. A coordenada $X$ da posição da perna 1 mostra que embora haja uma variação na fase de transferência trata-se de um desvio da ordem de 0,1 mm. Já na fase de suporte a posição em $X$ permanece constante.

A coordenada $Y$ representa claramente a ciclóide modelada. É possível visualizar $F_c = 0{,}05$ m e o próprio $\beta = 0{,}5$, resultando na igualdade tanto das

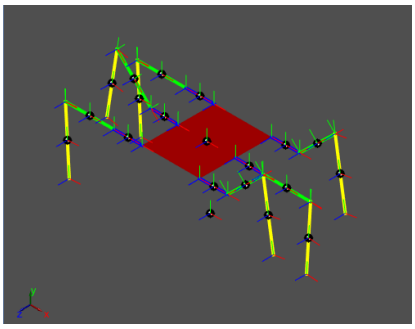
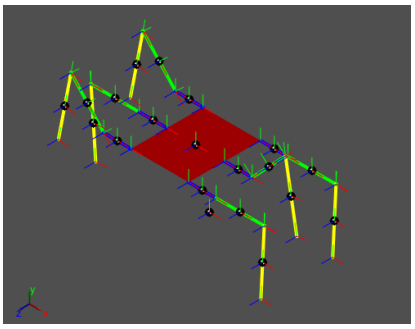
(a)  (b) 

Figura 5.18: Sequência de movimentos das pernas (1,4,5)-(2,3,6), respetivos às subfiguras (a)-(b), para a locomoção trípode.

amplitudes quanto entre as quatro fases de transferência e suporte.

A coordenada $Z$ indica o deslocamento da perna 1 ao longo dos 4 s de simulação. Somente quando o pé se encontra na fase de transferência é que ele se move na direção $Z$; quando em contacto com o solo permanece imóvel. Somando cada movimento em $Z$ ou realizando a diferença entre a posição final e inicial constatam-se os quatro passos dados pelo robô, cada um possuindo o comprimento $L_s$ = 0,025 m.

Quanto ao cálculo da estabilidade, a Figura 5.20 mostra a projeção do CG do corpo do hexápode sobre o polígono de suporte formado pelos pés 2, 3 e 6 em contacto com o solo.

O robô, possuindo os mesmos parâmetros de simulação das marchas anteriores, obteve:

Estável = 99,35 %

Marginalmente estável = 0 %

Instável = 0.65 %

Além do indicativo de maior percentagem de estabilidade, a marcha trípode obteve também o menor binário mínimo para um servo motor, apresentando:

Binário mínimo do servo motor = 2,32 kgf·cm

Para que o robô possa executar seguramente os três padrões de locomoção analisados os seus motores devem dispor de um binário mínimo de 8,15 kgf·cm.

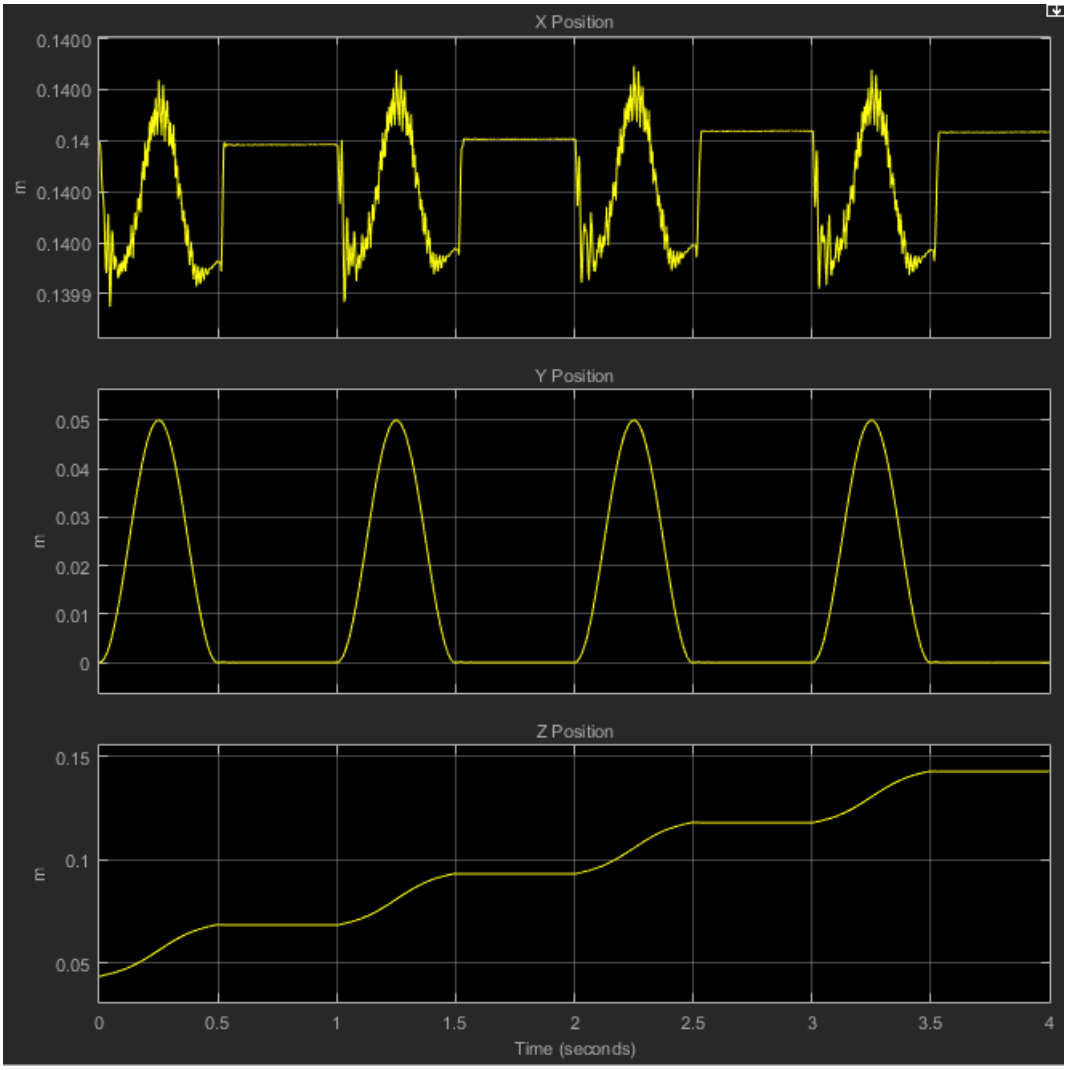



Figura 5.19: Posição do pé da perna 1 durante todo período de simulação.

## 5.10 Conclusão

Através dos resultados da simulação nota-se que o robô, embora seja leve, consegue-se sustentar firmemente sobre o solo, percorre a trajetória retilínea na fase de suporte, realiza as ciclóides nas fases de transferência, desloca-se frontalmente com velocidade constante e possui diferentes valores de estabilidade para diferentes tipos de locomoção. Constatou-se também que quanto maior for a estabilidade da marcha, menor é o binário mínimo exigido para os servo motores desenvolverem.

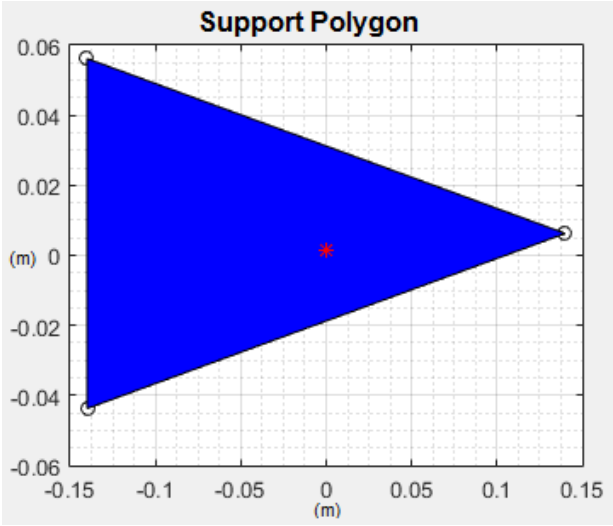



Figura 5.20: Polígono de suporte formado pelo pé 3 (lado direito) e os pés 2 e 6 (lado esquerdo de cima para baixo) com a respetiva projeção do CG do robô (ponto vermelho).

# Capítulo 6

# Implementação

*De forma a comprovar que o estudo cinemático modelado no Capítulo 4, e o estudo dinâmico simulado no capítulo anterior, são capazes de representar um sistema robótico multi-pernas real, este capítulo destina-se à realização de testes de locomoção do robô hexápode disponível e descrito no Capítulo 4 e estabelecer uma posterior avaliação.*

## 6.1 O robô

O robô hexápode disponível para a realização dos testes possui todos os seus membros (coxa, fémur e tíbia) e tronco feitos de madeira, com dimensões descritas no Anexo D. Para evitar que os pés do robô deslizem sobre o solo, os mesmos possuem borrachas nas suas extremidades. Todas as partes mecânicas são de origem artesanal, sendo que as dimensões relevantes para o algoritmo computar o padrão de locomoção são:

- $S_p = 0{,}05$ m;
- $L_1 = 0{,}038$ m;
- $L_2 = 0{,}067$ m;
- $L_3 = 0{,}09$ m.

E possui massas de:

- $m_{L1} = 0{,}021$ kg;
- $m_{L2} = 0{,}0025$ kg;

- $m_{L3} = 0{,}011$ kg;
- $m_{corpo} = 0{,}15$ kg.

Os valores das massas foram todos extraídos do modelo 3D feito em Solidworks®. O corpo é constituído de duas placas paralelas, separadas por espaçadores hexagonais com comprimento igual à altura do elo $L_1$. Por fim, para que o motor do elo $L_1$ não saia do seu eixo de rotação, fixou-se em todos os elos $L_1$ das pernas um conector unindo a parte superior do corpo com o eixo de rotação do motor.

## 6.2 Servo motores

Um servo motor é um motor elétrico que possui um sistema de controlo embebido cuja finalidade é controlar a posição angular de seu eixo de acordo com o sinal de referência. Para isso é utilizado um sistema com realimentação baseado num potenciómetro como elemento sensor. O potenciómetro é o elemento que supervisiona o rotação do eixo do motor, convertendo o movimento mecânico num sinal de tensão, com a finalidade de que o circuito de controlo faça a comparação da posição angular final e a atual.

Basicamente esses motores possuem três fios, dois destinados à alimentação e um ao sinal de controlo, sendo controlado através da Modulação por Largura de Pulso (PWM). Acoplado ao seu eixo há uma caixa redutora de rotação, que visa diminuir a velocidade angular para elevar o binário máximo desenvolvido.

Todos os servo motores do robô avaliado são do modelo Tower Pro SG90 que, de acordo com o *datasheet* do fabricante [32], possui as características vistas na Tabela 6.1.

| | Faixa de operação | Unidade de medida |
|---|---|---|
| Binário | 1,8-2,2 | kgf·cm |
| Variação Angular | -90 - +90 | Grau |
| Velocidade Angular | 300 | rpm |
| Tensão de trabalho | 4,8-6 | V |
| Frequência | 50 | Hz |
| Ciclo de trabalho | 1-2 | ms |
| Massa | 9 | g |
| Dimensões (CxLxA) | 22*11,5*22,5 | mm |

Tabela 6.1: Características de funcionamento do servo motor Tower Pro SG90.

## 6.3 Sistema embebido

Um sistema embebido é um sistema computacional mínimo dedicado à uma aplicação ou produto particular, de forma a executar um conjunto de tarefas específicas. Para esta aplicação procurou-se um sistema embebido que seja capaz de administrar os sinais de PWM, suportar o sistema operativo Linux e de comunicar com outros dispositivos. Neste caso, optou-se por utilizar o Raspberry Pi B+ por possuir baixo custo, maior quantidade de periféricos de propósito geral (GPIO), suportar o sistema operativo Raspbian (Linux), possuir uma *toolbox* exclusiva para a comunicação com o MatLab® e Simulink®, além de apresentar um *slot* para câmera, sendo útil para futuros trabalhos.

Embora atualmente haja versões mais sofisticadas, como o Raspiberry Pi 3 MODEL B, no momento de aquisição (2014) o modelo Raspberry Pi B+ detinha uma maior relação custo-benefício frente aos modelos Raspberry Pi A e B, atualmente obsoletos. A Tabela 6.2 exibe as características do Raspberry Pi B+ [33, 34].

| | Raspberry Pi B+ |
|---|---|
| *System-on-a-chip* (SoC) | Broadcom BCM2835 |
| CPU | 700 MHz ARM11 ARM1176JZF-S core |
| GPU | Broadcom VideoCore IV |
| Memória RAM | 512 MB |
| USB | 4 x USB 2.0 |
| GPIO | 40 pinos |
| SPI | 1 periférico |
| $I^2C$ | 2 periféricos |
| Serial | 1 periférico |
| Armazenamento Interno | MicroSD *slot* |
| Rede | 10/100 wired Ethernet RJ45 |
| Potência | 0,5-1 W |
| Tensão de Alimentação | 5 V |
| Massa | 40 g |
| Preço (2014) | US$35 |

Tabela 6.2: Características do sistema embebido Raspberry Pi B+.

Foi criada uma comunicação entre o computador principal e Raspberry do tipo *Secure Shell* (SSH) através de um cabo de rede, logo criou-se uma rede local *Ethernet* no computador principal com as seguintes configurações:

Endereço de *Internet Protocol* (IP): 192.168.137.1

Máscara de sub-rede: 255.255.255.0

E no Raspberry foi necessário alterar o ficheiro *interfaces*, contido em */etc/-network*, através do comando:

```
sudo nano /etc/network/interfaces
```

As alterações no ficheiro podem ser vistas na Figura 6.1 onde, após serem feitas, devem ser guardadas através das teclas *ctrl + X* e da confirmação da alteração. Para que as alterações tenham efeito é efetuada uma reinicialização do sistema através do comando:

```
sudo reboot
```

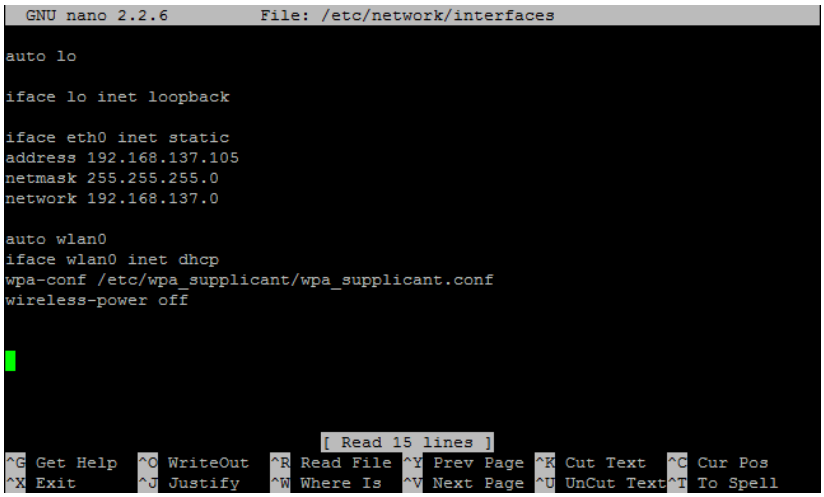



Figura 6.1: Configurações da rede criada.

## 6.4 Alimentação

A alimentação do sistema embebido deve ser independente da alimentação dos servo motores, pois os servo motores podem gerar ruídos na fonte de alimentação acarretando uma flutuação significativa na tensão do sistema embebido, podendo este apresentar mau funcionamento ou até ser danificado. A fonte de alimentação do Raspberry Pi B+ deve fornecer uma tensão fixa de 5 V e uma corrente de 1 A. A fonte de alimentação dos servo motores pode fornecer tensões entre 4,8 V e 6 V, porém deve ser capaz de fornecer uma corrente média de 220 mA para os 18 servo motores, logo deve fornecer no mínimo 4 A. Outro critério notório é que o peso e as dimensões das baterias devem ser mínimos para evitar a sobrecarga da estrutura do robô e, consequentemente, dos servo motores.

Por questões de peso e pelos servo motores disponíveis no robô não desenvolverem o binário mínimo exigido conforme o capítulo anterior, foi utilizada uma fonte de 5 V/2 A para alimentar o Raspberry Pi e uma fonte de alimentação de bancada, com duas saídas de tensão configuradas para operarem em paralelo, com uma tensão de 6 V e 4 A para alimentar os servo motores.

## 6.5 Servo driver

Para a distribuição dos sinais de controlo emitidos pelo sistema embebido são utilizadas duas placas Adafruit 16-*Channel* 12 bit. Este circuito possui o Circuito Integrado (CI) multiplexador PCA9685, que comunica através do protocolo *Inter-Integrated Circuit* ($I^2C$) controlando até 16 saídas com modulação PWM e frequência ajustável até cerca 1,6 kHz, sendo que cada saída possui resolução de 12 bit, alimentação de 5 V com proteção de polaridade reversa para o circuito e alimentação independente para os servo motores. O circuito possui seis bits de endereçamento que são configurados pelo utilizador através da soldagem dos terminais do canto superior direito da placa, que por padrão possui o endereço 0x40. Para a utilização de duas placas basta conectar os terminais de controlo, $I^2C$ e tensão de uma placa a outra, nos seus respetivos terminais, e alterar o endereço da segunda placa. Com 6 bits de endereçamento é possível controlar 992 saídas PWM, através da conexão de 62 placas eletrónicas.

Como há 18 servo motores, os servo motores do lado direito do robô foram conectados à placa de endereço 0x40 e os servo motores dispostos do lado esquerdo do robô foram conectados à placa de endereço 0x41. Como se trata de um barramento, apenas uma placa é conectada ao *master*, isto é, ao dispositivo controlador do barramento que neste caso é Raspberry B+.

## 6.6 Arquitetura

Para que os testes possam ser realizados, os diversos componentes do sistema foram esquematizados conforme a arquitetura de baixo nível vista na Figura 6.2.

Após os três tipos de locomoção serem simulados são gerados três ficheiros de texto contendo todas as amostras da posição angular de todas as dezoito juntas ao longo do tempo de simulação. Esses ficheiros serão utilizados por um código em linguagem Python, que será executado no Raspberry, com o intuito de efetuar a leitura de cada posição angular, converter o valor angular num sinal PWM correspondente e multiplexar esses sinais de PWM para os dezoito servo motores.

Através do pacote de suporte do MatLab® para o Raspberry Pi é realizada a conexão entre o ambiente do MatLab® e o Raspberry via SSH, através do seguinte comando:

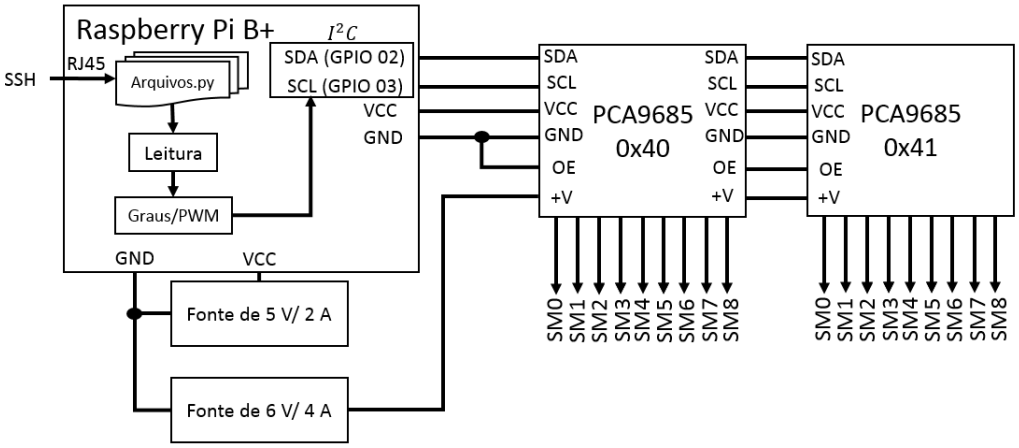



Figura 6.2: Arquitetura de baixo nível do sistema.

```
mypi = raspi('192.168.137.105','pi','raspberry')
```

O primeiro campo da função *raspi*() é o endereço IP do Raspberry Pi e os restantes são o nome de utilizador e senha de acesso, tendo como retorno:

```
mypi =

raspi with properties:

       DeviceAddress: '192.168.137.105'
                Port: 18726
           BoardName: 'Raspberry Pi Model B+'
       AvailableLEDs: {'led0'}
AvailableDigitalPins: [4 5 6 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27]
AvailableSPIChannels: {'CE0'  'CE1'}
   AvailableI2CBuses: {'i2c-1'}
         I2CBusSpeed: 100000
```

Esses são os valores padrão de inicialização da estrutura. Para verificar os dispositivos conectados ao barramento $I^2C$ basta digitar:

```
scanI2CBus(mypi,'i2c-1')

ans =

    '0x40'    '0x41'    '0x70'
```

Como resultado têm-se os endereços das duas placas Adafruit 16-*Channel* 12 bit (0x40 e 0x41) e o endereço padrão reservado dos CIs PCA9685 (0x70), apenas

para permitir a programação de todos os PCA9685s ao mesmo tempo [35].

Para enviar os ficheiros do computador principal para o Raspberry Pi é feito:

```
putFile(mypi,'log_raspi.txt','/home/pi/log_raspi_tripod.txt')
```

Esta função permite à estrutura *mypi* enviar o ficheiro *log_raspi.txt* ,localizado no interior da pasta de trabalho do MatLab®, para o endereço de destino */home/pi/log_raspi_tripod.txt*, localizado no *Raspberry Pi*.

Para ter acesso à linha de comandos do sistema operativo Raspbian executa-se o comando:

```
openShell(mypi)
```

A partir desse momento é aberta uma nova janela requerendo o nome de utilizador e senha de acesso, idênticos ao comando *raspi*(). Para realizar o controlo das placas Adafruit 16-*Channel* 12 bit usou-se a sua respetiva biblioteca de funções, sendo para isso necessário primeiro instalar a ferramenta $python - smbus$ requerida para adicionar o suporte $I^2C$ à linguagem Python:

sudo apt-get install python-smbus

Após instalação do pacote *smbus* é realizada a instalação do comando *git* que clona o repositório especificado para o diretório */home/pi*, isto é:

sudo apt-get install git build-essential python-dev

cd

git clone $https://github.com/adafruit/Adafruit\_Python\_PCA9685.git$

cd $Adafruit\_Python\_PCA9685$

sudo python *setup.py* install

Desse modo é possível criar um código em linguagem Python que herda as funções oriundas da biblioteca $Adafruit\_Python\_PCA9685$. O código desenvolvido encontra-se no Anexo A e seu funcionamento dá-se da seguinte forma: inicialmente é selecionado um dos três tipos de locomoção desenvolvidos, abre-se o ficheiro correspondente e lê-se seus dados. Cada valor de junta é armazenado numa coluna do ficheiro, logo, para lê-los é utilizado o recurso *split*() (vide Anexo E) que extrai todos os valores contidos numa linha e, posteriormente, efetua-se a leitura de cada elemento contido nesta linha.

Após reconhecidos os valores angulares é feita a conversão para o seu respetivo valor em PWM. Cada saída do CI PCA9685 possui uma resolução de 12 bit podendo representar o sinal de saída de 0 a 4095. Porém, a faixa de valores digitais em que o servo motor opera (-90° a 90° correspondente à faixa de 1 ms a 2 ms) varia de 220 a 430, logo a relação entre a largura de pulso (*duty cycle*) e o ângulo *theta* é dada por:

*Duty Cycle* = 1,1666*Theta + 325

Obtido o valor do *duty cycle* cria-se a estrutura com o nome "pwm" que receberá tal parâmetro. Essa estrutura é do tipo PCA9685, que possui um campo para a definição da frequência de trabalho (que para os servo motores empregados é de 50 Hz) e um campo para a emissão do sinal PWM para um canal específico.

Por fim é executado o ficheiro através do comando:

```
sudo python roboHexapod.py
```

Desse ponto em diante o código executa o padrão de locomoção selecionado de modo que o robô possua os mesmos comportamentos vistos no Capítulo 5.

## 6.7 Teste, depuração e resultados

Inicialmente, antes de colocar o hexápode em contacto com o solo, os testes foram efetuados sobre um suporte que manteve o robô suspenso. Após verificar o funcionamento de todos os padrões de locomoção o robô foi colocado sobre o solo.

O primeiro teste realizado foi com a marcha em onda metacronal. De acordo com os resultados da simulação, onde o binário mínimo desenvolvido para este tipo de marcha deve ser de 8,15 kgf·cm, espera-se que o robô se mova de forma semelhante à simulação, porém com uma tendência a aproximar-se do solo indicando insuficiência dos binários desenvolvidos.

A Figura 6.3 exibe a sequência de movimentos realizada pelo robô real e pode ser comparada com a Figura 5.14.

A cada passo efetuado as posições de cada pé do hexápode diferenciam-se das posições simuladas devido ao erro sistemático do binário dos motores, sendo claro que ao final de quatro ciclos de locomoção a posição do CG do robô não coincide com a simulação. A Figura 6.4 exibe tal comportamento tendo-se retirado todos os dados a partir da posição inicial e final de todos os seis pés de modo a medir

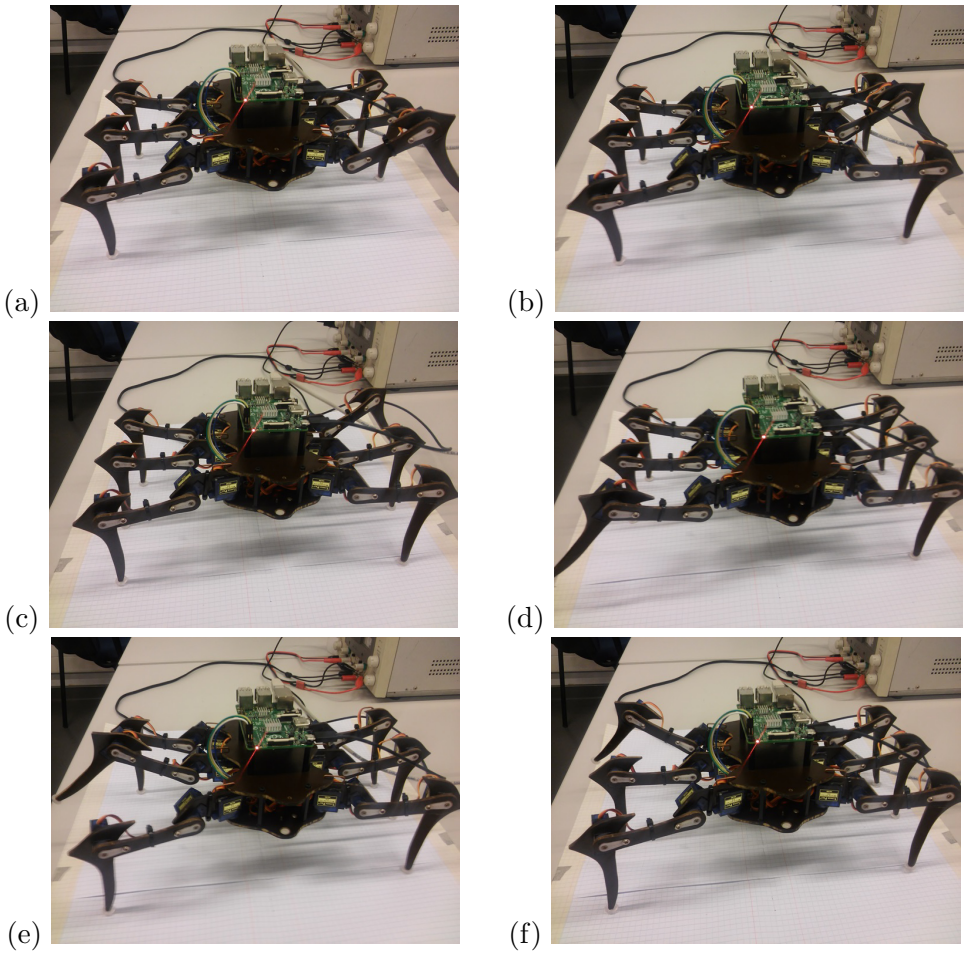


Figura 6.3: Sequência de movimentos das pernas 1-3-5-2-4-6 respetivos às subfiguras (a)-(b)-(c)-(d)-(e)-(f), para o hexápode real em locomoção Onda Metacronal.

o deslocamento do CG do robô através do Sistema de Direção Diferencial (SDD) [36]. Para isso foi utilizada a seguinte equação:

$$\Delta S = \frac{\Delta S_D + \Delta S_E}{2} \tag{6.1}$$

e

$$\Delta\theta = \frac{\Delta S_D - \Delta S_E}{b} \tag{6.2}$$

Onde:

- $\Delta S$ é o deslocamento do CG do robô;

- $\Delta S_D$ e $\Delta S_E$ são os deslocamentos do lado direito e esquerdo do robô, respetivamente;
- $\Delta\theta$ é o ângulo de desvio;
- $b$ é a distância entre as ancas direita e esquerda.

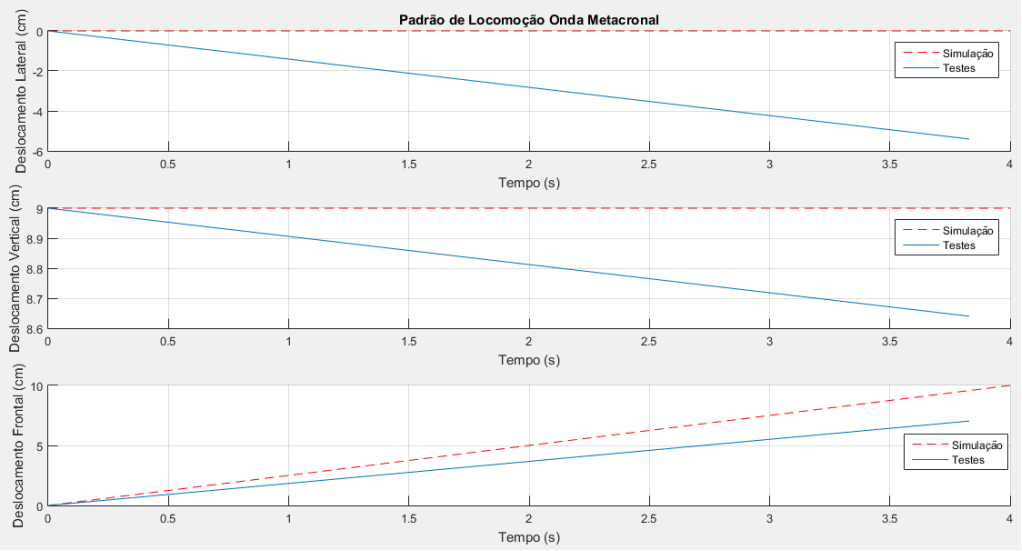
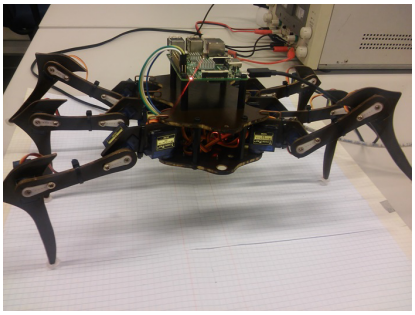



Figura 6.4: Posição do CG do robô ao longo do tempo de execução de quatro ciclos de locomoção em Onda Metacronal, com $L_s = 0{,}025$ m e $H_B = 0{,}09$ m.

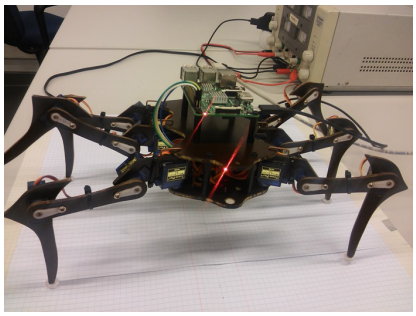
As retas da Figura 6.4 são feitas somente com os dados da posição inicial e final e, portanto, não expressam o comportamento instantâneo do robô, sendo que para verificar a evolução temporal de tais dados é necessário possuir um dispositivo sensor. Os dados $\Delta S_D$ e $\Delta S_E$ são os valores médios dos dez testes realizados para cada padrão de locomoção. A Figura 6.4 também revela que, para os testes efetuados, o tempo de duração de quatro ciclos de locomoção não foi igual ao da simulação. Isso ocorreu devido a dois fatores: o primeiro trata-se da velocidade de rotação do servo motor, que quanto maior for o número de amostras menor será a sua velocidade, pois o erro entre a posição de referência e a atual será menor; o segundo fator diz respeito ao tempo de execução do código que também interfere no tempo de locomoção, pois o código efetua a leitura de um ficheiro, converte a posição angular num sinal de PWM e, posteriormente, envia este valor para o respetivo servo motor. Para solucionar o primeiro problema reduziu-se o número de amostras de 1000 para 200, porém, por questões de tempo, não foi possível solucionar o segundo problema e assim o tempo de execução cronometrado para quatro ciclos de locomoção foi de 3,83 s.

O teste feito para a locomoção tetrápode também se revelou não ser idêntico ao da simulação - os seus passos podem ser vistos na Figura 6.5 . Notou-se que o robô se desviou para o lado esquerdo, em contraste com o padrão Onda Metacronal, que se desviou para a direita (os valores positivos do deslocamento lateral da Figura 6.6 indicam o movimento para o lado esquerdo). Neste caso, o tempo de execução dos quatro ciclos de locomoção foi de 4,10 s.

(a)



(b)



(c)

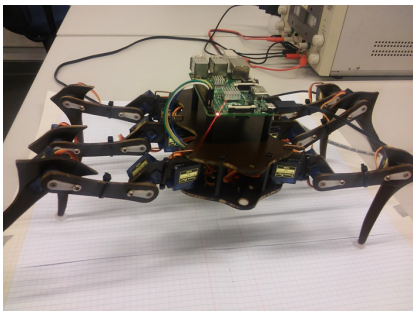


Figura 6.5: Sequência de movimentos das pernas (1,6)-(4,5)-(2,3) respetivos às subfiguras (a)-(b)-(c) para o hexápode real em locomoção Tetrápode.

O último teste efetuado foi relativo à marcha com o padrão de locomoção trípode. A Figura 6.7 exibe a sequência de movimentos feitos pelo robô real e pode ser comparada com a Figura 5.18. A posição do CG do robô ao longo do tempo pode ser vista na Figura 6.8. Como os servo motores operaram com o máximo valor de binário houve momentos de sobre aquecimento que obrigaram a ter intervalos de pausa durante os testes. Para a marcha trípode o tempo de execução dos quatro ciclos de locomoção foi de 3,73 s (para medir o tempo de execução gravaram-se, através de vídeos, todos os testes e posteriormente este

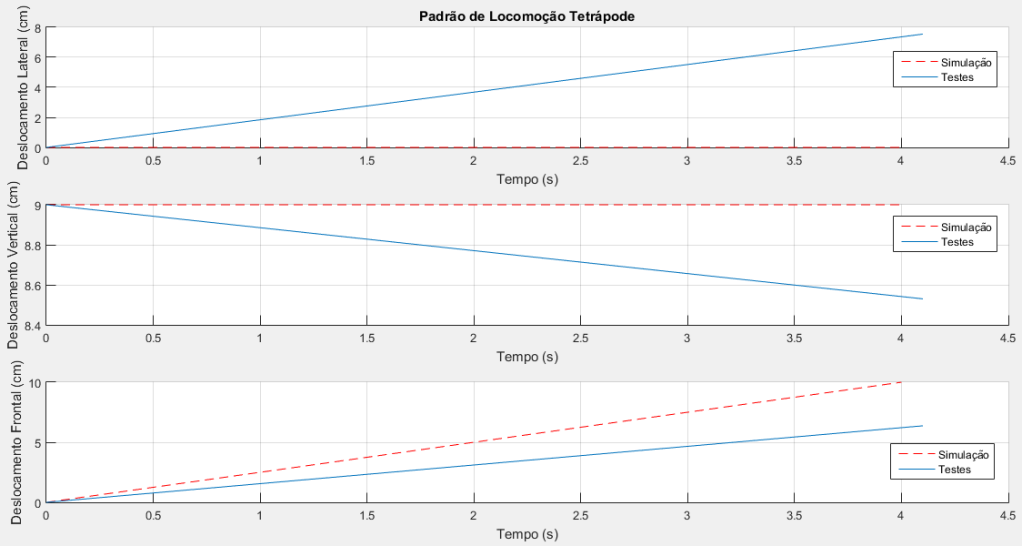



Figura 6.6: Posição do CG do robô ao longo do tempo de execução de quatro ciclos de locomoção Tetrápode com $L_s = 0{,}025$ m e $H_B = 0{,}09$ m.

foram analisados no computador). Outro problema tido durante os testes foi relativo às engrenagens das caixas de redução dos servo motores - devido aos excessivos binários aplicados, os dentes das engrenagens saltavam provocando erros de posição angular.

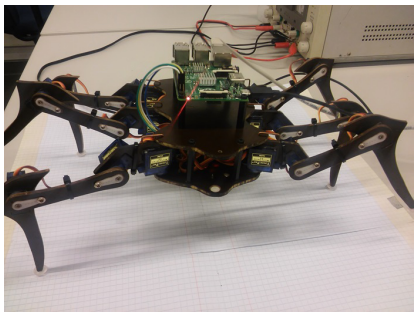
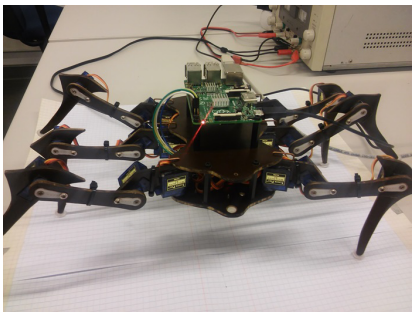
(a)  (b) 

Figura 6.7: Sequência de movimentos das pernas (1,4,5)-(2,3,6) respetivos às subfiguras (a)-(b) para o hexápode real em locomoção Trípode.

Observou-se também que devido à baixa resolução angular dos servo motores a postura inicial do robô, com todos os servo motores posicionados à 0°, não foi igual ao da Figura 4.3, possuindo pequenas variações no posicionamento de cada perna do robô.

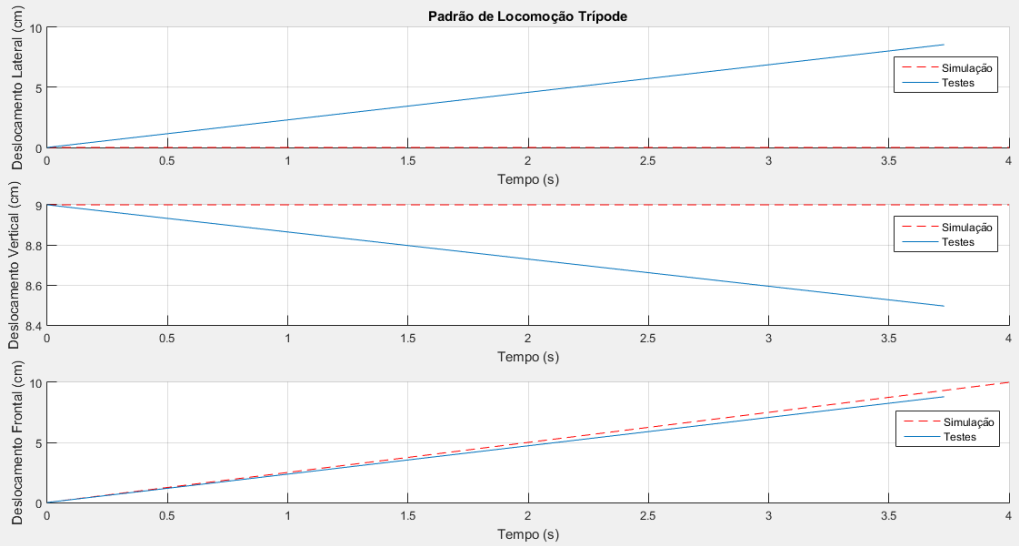



Figura 6.8: Posição do CG do robô ao longo do tempo de execução de quatro ciclos de locomoção Trípode com $L_s = 0{,}025$ m e $H_B = 0{,}09$ m.

## 6.8 Conclusão

Verificou-se a importância da correta escolha dos servo motores a serem utilizados, que interferiram diretamente na postura do robô. Verificaram-se também desvios angulares da trajetória do CG na direção $+Z$ variando-se para cada padrão de locomoção. Os resultados obtidos neste capítulo eram esperáveis para esta aplicação pois para suportar a estrutura mecânica do robô os motores devem atender às especificações mencionadas no Capítulo 5.

# Capítulo 7

# Conclusões

*Neste capítulo é descrita a relevância dos dados, são efetuadas propostas para o desenvolvimento de futuros trabalhos bem como o encerramento da Dissertação.*

## 7.1 Balanço

Ao longo do trabalho desenvolvido estipularam-se algumas metas para conduzir aos objetivos propostos. As metas alcançadas foram:

- introduzir os conceitos básicos de simulação em SimMechanics™;
- desenvolver o estudo cinemático;
- simular um sistema robótico em SimMechanics™;
- reproduzir diferentes padrões de locomoção.

Em contrapartida, os temas e assuntos que ou foram difíceis de alcançar devido às restrições temporais ou não foram capazes de serem observados foram:

- simulação do modelo desenvolvido em SolidWorks® pois demorou cerca de dez minutos para simular um segundo de movimento;
- verificação prática do movimento correto do robô, que se distanciou da simulação devido ao modelo dos servo motores utilizados;
- por questões de tempo não foi possível implementar os movimentos de rotação e deslocamento lateral.

## 7.2 Desenvolvimentos futuros

Visto que a presente Dissertação se focou na correta modelação e simulação de um sistema robótico multi-pernas, há inúmeras propostas para o desenvolvimento de futuros trabalhos, tais como implementar outros padrões de locomoção de forma a analisar os critérios de estabilidade e binário desejáveis.

Para o planeamento de trajetórias foi utilizada a função ciclóide para descrever o movimento das pernas, logo poderia haver um estudo comparativo dos modos de planeamento de uma perna na sua fase de transferência e analisar os seus impactos na locomoção do robô.

Outra proposta seria procurar formas de otimização temporal das simulações em SimMechanics™ para possibilitar a visualização dos modelos robóticos importados de *softwares* CAD.

Para além da locomoção frontal, poder-se-á estudar algoritmos de locomoção lateral, de rotação do corpo e de inclinação do solo, de forma a ampliar a diversidade de ambientes de aplicação de tais robôs.

Devido à falta de sensores no robô utilizado, é sugerido o desenvolvimento de uma rede de sensores de força de contacto para os pés do robô podendo assim ser comprovada a existência e magnitude das forças em três dimensões, bem como fazer uso de um acelerómetro para monitorar o deslocamento do CG do robô ao longo do tempo de execução dos padrões de locomoção.

Por fim, um projeto interessante seria a inserção de uma câmera digital no sistema para fazer o processamento digital das imagens recolhidas do ambiente e possibilitar o ajuste automático da marcha do robô.

## 7.3 Conclusão

A robótica de locomoção é uma área em que as aproximações dos modelos robóticos são tão fiéis quanto forem os esforços gastos onde a simulação de tais sistemas torna-se viável por questões económicas e de agilidade. Através dos estudos cinemáticos desenvolvidos foi possível executar o movimento de um robô hexápode, tanto no Simulink®, quanto em testes práticos. Foi possível também averiguar a amplitude das forças de sustentação distribuídas ao longo das seis pernas do robô.

Com a implementação do algoritmo de determinação da estabilidade foi possível verificar que para diferentes padrões de locomoção houve uma progressão na estabilidade da marcha em onda metacronal para as marchas tetrápode e trípode. Consoante esses resultados, houve também um decréscimo consecutivo do binário mínimo exigido para os motores reproduzirem.

Através de experiências foi possível comprovar que para os servo motores utilizados, modelo Tower Pro SG90, o robô não pôde reproduzir fielmente os movimentos planeados e simulados ocasionando uma redução constante da altura do corpo ao solo.

Portanto, embora os resultados práticos não tenham sido empolgantes, tais comportamentos eram previsíveis pela simulação, que possibilitou a revelação das forças e binários envolvidos em cada pé do robô.

# Bibliografia


[1] Hiarai, Kazuo and Hirose, Masato and Haikawa, Yuji and Takenaka, Toru, "The Development of Honda Humanoid Robot," *Robotics and Automation, 1998. Proceedings. 1998 IEEE International Conference on*, pp. 1321–1326 vol.2, 1998. [citado na p. v, 7, 8]

[2] Phuong, Nguyen T. and Kim, Dae W. and Kim, Hak K. and Kim, Sang B., "An Optimal Control Method for Biped Robot with Stable Walking Gait," *Humanoids 2008 - 8th IEEE-RAS International Conference on Humanoid Robots*, pp. 211–218, 2008. [citado na p. v, 1, 8, 9]

[3] Sun, Lei and Zhoul, Yajing and Chen, Wanming and Liang, Huawei and Meil, Tao, "Modeling and Robust Control of Quadruped Robot," *2007 International Conference on Information Acquisition*, pp. 356–360, 2007. [citado na p. v, 1, 9]

[4] Wen, Jintao and Wang, Jianhua and Chen, Weihai and Zhang, Jingbing, "A Gait Planning Approach for Locomotion Stability of Four-legged Robots," *2012 7th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 324–329, 2012. [citado na p. v, 9, 10]

[5] Raibert, Marc and Blankespoor, Kevin and Nelson, Gabriel and Playter, Rob, "BigDog, the Rough-Terrain Quaduped Robot," *Boston Dynamics*, 2008. [citado na p. v, 10, 11]

[6] Sorin, Mănoiu O. and Niţulescu, Mircea, "Hexapod Robot. Virtual Models for Preliminary Studies," *System Theory, Control, and Computing (ICSTCC), 2011 15th International Conference on*, pp. 1–6, 2011. [citado na p. v, 12, 13]

[7] Ferrell, Cynthia, "Robust Agent Control of an Autonomous Robot with Many Sensors and Actuators," *MIT*, 2001. [citado na p. v, 1, 13, 14]

[8] Song, Shin M. and Waldron, Kenneth J., "Machines that Walk: theAdaptive Suspension Vehicle," *The MIT Press*, 1989. [citado na p. v, 14]

[9] Waldron, Kenneth J. and McGhee, Robert B., "The adaptive suspension vehicle," *IEEE Contr. Syst. Mag.*, 1986. [citado na p. v, 14]

[10] Kirchner, Frank. NASA evaluates eight-legged Scorpion robot for future exploration. [Online]. Available: http://www.nasa.gov/centers/ames/research/exploringtheuniverse/scorpion_robot.html [citado na p. v, 15]

[11] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*, 1st ed. Springer, 2008, vol. Único. [citado na p. v, 14, 15, 41, 42, 43]

[12] Bares, John E. and Wettergreen, David S., "Dante II: technical description, results and lessons learned," *Int. J. Robot*, 1999. [citado na p. v, 15]

[13] M. F. S. Silva, "Sistemas robóticos de locomoção multipernas," Ph.D. dissertation, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2005. [citado na p. v, 8, 16, 39, 41, 45]

[14] K. Iagnemma and S. Dubowsky, *Mobile Robots in Rough Terrain*, 1st ed. Springer-Verlag Berlin Heidelberg, 2004, vol. 12. [citado na p. 1]

[15] S. P. O. Carvalho, "Optimização dos parâmetros de um robô hexápode através de algoritmos genéticos," Master's thesis, Faculdade de Engenharia da Universidade do Porto, Porto, Portugal, 2008. [citado na p. 1, 46]

[16] J. L. Jones and A. M. Flynn, *Mobile Robots: Inspiration to Implementation*, 2nd ed. A. K. Peters, Ltd., 1998, vol. Único. [citado na p. 2]

[17] L. R. Douat, "Estabilização do caminhar de um robô bípede de 5 elos com compensação do movimento dorsal," Master's thesis, Universiade Federal de Santa Catarina, Florianópolis, Brasil, 2008. [citado na p. 8]

[18] Gorner, Martin and Hirzinger, Gerd, "Analysis and Evaluation of the Stability of a Biologically Inspired, Leg Loss Tolerant Gait for Six- and Eight-Legged Walking Robots," *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 4728–4735, 2010. [citado na p. 12]

[19] Breazeal, Cynthia. Switch Sound File Converter. [Online]. Available: http://www.ai.mit.edu/projects/hannibal/hannibal.html [citado na p. 13]

[20] Ye, Xiufen and Guo, Baofeng G. S. and Feng, Weixing and Wang, Kejun, "Gait Analysis of Underwater Octopod Microrobot," *2006 IEEE International Conference on Robotics and Biomimetics*, pp. 1316–1321, 2006. [citado na p. 15]

[21] R. Rynkevic, "Biomedical modeling and simulation of the spider crab (maja brachydactyla)," Master's thesis, Instituto Superior de Engenharia do Porto, Porto, Portugal, 2012. [citado na p. 15, 39]

[22] MscSoftware™. ADAMS - The Multibody Dynamics Simulation Solution. [Online]. Available: http://www.mscsoftware.com/product/adams [citado na p. 17]

[23] MathWorks®. Simscape Multibody - Model and simulate multibody mechanical systems. [Online]. Available: http://www.mathworks.com/products/simmechanics/?refresh=true [citado na p. 17]

[24] Cyberbotics. Webotis. [Online]. Available: https://www.cyberbotics.com/webots.php [citado na p. 17]

[25] T. F. V. S. Castro, "Simulação de robôs quadrúpedes utilizando o simmechanics," Master's thesis, Instituto Superior de Engenharia do Porto, Porto, Portugal, 2012. [citado na p. 19, 44, 45, 46, 52, 53]

[26] MathWorks®. Solver Pane. [Online]. Available: http://www.mathworks.com/help/simulink/gui/solver-pane.html [citado na p. 19]

[27] Oliveira, Giovana T. S. and Saramago, Sezimária F. P. and Nogueira, Antônio C., "Estudo das Singularidades de Robôs Manipuladores usando Base de Groebner," *XXXIII Congresso de Matemática Aplicada e Computacional*, 2010. [citado na p. 38]

[28] Gao, Haibo and Jin, Ma and Liu, Yiqun and Ding, Liang and Yu, Haitao and Deng, Zongquan, "Turning gait planning and simulation validation of a hydraulic hexapod robot," *Fluid Power and Mechatronics (FPM), 2015 International Conference on*, pp. 842–847, 2015. [citado na p. 39]

[29] I. R. Woering, "Simulating the "first steps"of a walking hexapod robot," Master's thesis, Eindhoven University of Technology, Eindhoven, Netherlands, 2011. [citado na p. 39, 52]

[30] D. Thilderkvist and S. Svensson, "Motion control of hexapod robot using model-based design," Master's thesis, Lund University, Sweden, 2015. [citado na p. 41]

[31] Marques, Gil C. Dinâmica - As leis de Newton. [Online]. Available: http://efisica.if.usp.br/mecanica/universitario/dinamica/intro/ [citado na p. 43]

[32] Micropik. SG90 9 g Micro Servo. [Online]. Available: http://www.micropik.com/PDF/SG90Servo.pdf [citado na p. 70]

[33] elinux. RPi Hardware. [Online]. Available: http://elinux.org/RPi_Hardware [citado na p. 71]

[34] Raspberry Pi Foundation. RASPBERRY PI 1 MODEL B+. [Online]. Available: https://www.raspberrypi.org/products/model-b-plus/ [citado na p. 71]

[35] NXP Semiconductors. PCA9685. [Online]. Available: https://cdn-shop.adafruit.com/datasheets/PCA9685.pdf [citado na p. 75]

[36] Bianchi, Reinaldo. Robótica. [Online]. Available: http://slideplayer.com.br/slide/1357065/ [citado na p. 77]

# Anexos

# Anexo A

# Função utilizada para o planeamento de trajetórias no MatLab®

```
function [theta1,theta2,theta3,Posicao_X,Posicao_Y,Posicao_Z] = ...
Path_Planning(Fc,Ls,T,n,beta,L1,L2,L3,HB,sample_time)
    t = 0:sample_time:n*T;
    TS = (beta*T);      % Support Phase
    ts = 0:sample_time:TS;
    TT = (1-beta)*T;    % Transfer Phase
    tt = 0:sample_time:TT;
    VF = Ls/T;          % Forward Velocity of the Body (-Y direction)
    VF_tt = (1/(1-beta))*VF;

    theta_1 = [];
    theta_2 = [];
    theta_3 = [];
    theta_1_2 = [];
    theta_2_2 = [];
    theta_3_2 = [];

    %------------------------------Cycloid-------------------------------
    Posicao_X1(1,1:(length(tt))) = 0.1;
    Posicao_Y1 = VF_tt*(tt-sin(2*pi*tt/TT)*TT/(2*pi)) - VF.*tt - Ls*beta/2;
    Posicao_Z1 = Fc/2*(1-cos(2*pi*tt/TT));

    %Inverte horizontalmente o vetor posicao X
    Posicao_X1 = wrev(Posicao_X1);
    %Inverte horizontalmente o vetor posicao Y
    Posicao_Y1 = wrev(Posicao_Y1);
```

```
%Inverte horizontalmente o vetor posicao Z
Posicao_Z1 = wrev(Posicao_Z1);

for i = 1:1:length(tt),
    Theta_1 = atan2(Posicao_Y1(1,i),Posicao_X1(1,i));
    Theta_3 = -acos((Posicao_X1(1,i)^2+Posicao_Y1(1,i)^2+ ...
    (Posicao_Z1(1,i)-HB)^2+L1^2-L2^2-L3^2- ...
    2*L1*sqrt(Posicao_X1(1,i)^2+Posicao_Y1(1,i)^2))/(2*L2*L3));
    Theta_2 = atan2((Posicao_Z1(1,i)-HB), ...
    sqrt(Posicao_X1(1,i)^2+Posicao_Y1(1,i)^2) - L1)- ...
    atan2((L3*sin(Theta_3)),(L2+L3*(Posicao_X1(1,i)^2+ ...
    Posicao_Y1(1,i)^2 + (Posicao_Z1(1,i)-HB)^2+L1^2-L2^2 - ...
    L3^2-2*L1*sqrt(Posicao_X1(1,i)^2+Posicao_Y1(1,i)^2)) ...
    /(2*L2*L3)));

    theta_1 = [theta_1 180/pi*Theta_1];
    theta_2 = [theta_2 180/pi*Theta_2];
    theta_3 = [theta_3 180/pi*Theta_3];
end

X = ((L1 + L2*cos(Theta_2) + ...
L3*cos(Theta_2 + Theta_3))*cos(Theta_1));
%--------------------------------Movimento-Corpo-----------------
Posicao_X1_2(1,1:length(ts)) = X;
Posicao_Y1_2 = -VF.*ts + Ls*beta/2;
Posicao_Z1_2(1,1:length(ts)) = HB;

%Inverte horizontalmente o vetor posicao X
Posicao_X1_2 = wrev(Posicao_X1_2);
%Inverte horizontalmente o vetor posicao Z
Posicao_Z1_2 = wrev(Posicao_Z1_2);

for i = 1:1:length(ts),
    Theta_1 = atan2(Posicao_Y1_2(1,i),Posicao_X1_2(1,i));
    Theta_3 = acos(((sqrt(Posicao_X1_2(1,i)^2+ ...
    Posicao_Y1_2(1,i)^2) - L1)^2 + Posicao_Z1_2(1,i)^2 - ...
    L2^2 - L3^2)/(2*L2*L3));
    Theta_2 = atan2(Posicao_Z1_2(1,i), ...
    sqrt(Posicao_X1_2(1,i)^2+Posicao_Y1_2(1,i)^2)-L1) - ...
    atan2((L3*sin(Theta_3)),(L2+L3*cos(Theta_3)));

    theta_1_2 = [theta_1_2 180/pi*Theta_1];
    theta_2_2 = [theta_2_2 180/pi*Theta_2];
    theta_3_2 = [theta_3_2 180/pi*Theta_3];
end

%Salva o valor de theta1 em um unico vetor
theta1 = horzcat(theta_1,-theta_1_2(1,2:end));
%Salva o valor de theta2 em um unico vetor
theta2 = horzcat(theta_2,-theta_2_2(1,2:end));
%Salva o valor de theta3 em um unico vetor
```

```
    theta3 = horzcat(theta_3,-theta_3_2(1,2:end));

    Theta1 = theta1;
    Theta2 = theta2;
    Theta3 = theta3;

    %Salva o valor da posicao X em um unico vetor
    Posicao_X = horzcat(Posicao_X1,Posicao_X1_2(1,2:end));
    %Salva o valor da posicao Y em um unico vetor
    Posicao_Y = horzcat(Posicao_Z1,Posicao_Z1_2(1,2:end));
    %Salva o valor da posicao Z em um unico vetor
    Posicao_Z = horzcat(Posicao_Y1,Posicao_Y1_2(1,2:end));

    posicao_X = Posicao_X;
    posicao_Y = Posicao_Y;
    posicao_Z = Posicao_Z;

    %Ajusta o numero de ciclos
    if n>1,
        for i = 1:1:n-1,
            theta1 = horzcat(Theta1,theta1(1,2:end));
            theta2 = horzcat(Theta2,theta2(1,2:end));
            theta3 = horzcat(Theta3,theta3(1,2:end));

            Posicao_X = horzcat(posicao_X,Posicao_X(1,2:end));
            Posicao_Y = horzcat(posicao_Y,Posicao_Y(1,2:end));
            Posicao_Z = horzcat(posicao_Z,Posicao_Z(1,2:end));
        end
    end
end
```

# Anexo B

# Código com as inicializações das variáveis e o cálculo dos momentos de inércia

```
clc
close all
clear all
run('Robot_GUI.m');
%-------------------------Parametros-de-simulacao-----------------------
global sample_time stop_time L1 L2 L3 LC CC AC P1 P2 P3 I1 I2 I3 ...
D1 D2 D3 N1 N2 N3 Kx Ky Kz Bx By Bz m X_ini Y_ini Z_ini m1 m2 m3 ...
m_body r g t p1 p2 p3 p4 p5 p6 CG i in on count count2 stable ...
marginally_stable instable gait leg_1_x leg_1_y leg_1_z ...
leg_2_x leg_2_y leg_2_z leg_3_x leg_3_y leg_3_z leg_4_x ...
leg_4_y leg_4_z leg_5_x leg_5_y leg_5_z leg_6_x leg_6_y ...
leg_6_z hip_1 hip_2 hip_3 hip_4 hip_5 hip_6 knee_1 knee_2 ...
knee_3 knee_4 knee_5 knee_6 foot_1 foot_2 foot_3 foot_4 foot_5 ...
foot_6 Log T n velocidade Torque_m1_leg1 Torque_m2_leg1 ...
Torque_m3_leg1 Torque_m1_leg2 Torque_m2_leg2 Torque_m3_leg2 ...
Torque_m1_leg3 Torque_m2_leg3 Torque_m3_leg3 Torque_m1_leg4 ...
Torque_m2_leg4 Torque_m3_leg4 Torque_m1_leg5 Torque_m2_leg5 ...
Torque_m3_leg5 Torque_m1_leg6 Torque_m2_leg6 Torque_m3_leg6 ...
Servo_Motor_Minimo

sample_time = 0.0005;
stop_time = 1;

X_ini = 0;
Y_ini = 0.09;
Z_ini = 0;

g = -9.81;

upper_normal_saturation = inf;
lower_normal_saturation = 0;
upper_friction_saturation = inf;
lower_friction_saturation = -inf;

%-------------------------Dimensoes-do-chao-----------------------------
m_ground = 400;
```

```
h_ground = 0.001;
w_ground = 2;
d_ground = 2;

%-------------------------Dimensoes-das-pernas-------------------------
L1 = 0.04;
L2 = 0.07;
L3 = 0.09;

m_body = 0.0281866;
m1 = 0.0210488;
m2 = 0.00258293;
m3 = 0.0112208;

r = 0.01;
%-------------------------Dimensoes-do-corpo-------------------------
LC = 0.08;
CC = 0.1;
AC = 0.08;
ALT = L3;

%-------------------------------Inercias-------------------------------
%Chao
inertia_ground = [(1/12*m_ground*(h_ground^2+d_ground^2)) 0 0; ...
    0 (1/12*m_ground*(w_ground^2+d_ground^2)) 0; ...
    0 0 (1/12*m_ground*(w_ground^2+h_ground^2))];

%Corpo
inertia_body = [(1/12*m_body*(AC^2+CC^2) + m_body*(ALT)^2) 0 0; ...
    0 1/12*m_body*(CC^2+LC^2) 0; ...
    0 0 (1/12*m_body*(LC^2+AC^2) + m_body*(ALT)^2)];

% %Perna 1
inertia_leg_1_1 = [(1/2*m1*r^2 + m1*((CC/2)^2+ALT^2)) 0 0; ...
    0 (1/12*m1*(3*r^2+L1^2) + m1*((CC/2)^2 + (LC/2+L1/2)^2)) 0; ...
    0 0 (1/12*m1*(3*r^2+L1^2) + m1*((LC/2+L1/2)^2 + ALT^2))];

inertia_leg_1_2 = [(1/2*m2*r^2 + m2*((CC/2)^2+ALT^2)) 0 0; ...
    0 (1/12*m2*(3*r^2+L2^2) + m2*((CC/2)^2 + (LC/2+L1+L2/2)^2)) 0; ...
    0 0 (1/12*m2*(3*r^2+L2^2) + m2*((LC/2+L1+L2/2)^2 + ALT^2))];

inertia_leg_1_3 = [(1/12*m3*(3*r^2+L3^2) + m3*((CC/2)^2 + (ALT/2)^2)) 0 0; ...
    0 (1/2*m3*r^2 + m3*((CC/2)^2 + (LC/2+L1+L2)^2)) 0; ...
    0 0 (1/12*m3*(3*r^2+L3^2) + m3*((LC/2+L1+L2)^2 + (ALT/2)^2))];

%Perna 2
inertia_leg_2_1 = [(1/2*m1*r^2 + m1*((CC/2)^2+ALT^2)) 0 0; ...
    0 (1/12*m1*(3*r^2+L1^2) + m1*((CC/2)^2 + (LC/2+L1/2)^2)) 0; ...
    0 0 (1/12*m1*(3*r^2+L1^2) + m1*((LC/2+L1/2)^2 + ALT^2))];

inertia_leg_2_2 = [(1/2*m2*r^2 + m2*((CC/2)^2+ALT^2)) 0 0; ...
    0 (1/12*m2*(3*r^2+L2^2) + m2*((CC/2)^2 + (LC/2+L1+L2/2)^2)) 0; ...
    0 0 (1/12*m2*(3*r^2+L2^2) + m2*((LC/2+L1+L2/2)^2 + ALT^2))];

inertia_leg_2_3 = [(1/12*m3*(3*r^2+L3^2) + m3*((CC/2)^2 + (ALT/2)^2)) 0 0; ...
    0 (1/2*m3*r^2 + m3*((CC/2)^2 + (LC/2+L1+L2)^2)) 0; ...
    0 0 (1/12*m3*(3*r^2+L3^2) + m3*((LC/2+L1+L2)^2 + (ALT/2)^2))];

%Perna 3
inertia_leg_3_1 = [(1/2*m1*r^2 + m1*(ALT)^2) 0 0; ...
    0 (1/12*m1*(3*r^2+L1^2) + m1*(LC/2+L1/2)^2) 0; ...
    0 0 (1/12*m1*(3*r^2+L1^2) + m1*(LC/2+L1/2)^2 + (ALT)^2)];

inertia_leg_3_2 = [(1/2*m2*r^2 + m2*(ALT)^2) 0 0; ...
    0 (1/12*m2*(3*r^2+L2^2) + m2*(LC/2+L1+L2/2)^2) 0; ...
    0 0 (1/12*m2*(3*r^2+L2^2)) + m2*((LC/2+L1+L2/2)^2 + ALT^2)];

inertia_leg_3_3 = [(1/12*m3*(3*r^2+L3^2) + m3*(ALT/2)^2) 0 0; ...
    0 (1/2*m3*r^2 + m3*(LC/2+L1+L2)^2) 0; ...
    0 0 (1/12*m3*(3*r^2+L3^2) + m3*((LC/2+L1+L2)^2 + (ALT/2)^2))];
```

```
%Perna 4
inertia_leg_4_1 = [(1/2*m1*r^2 + m1*(ALT)^2) 0 0; ...
    0 (1/12*m1*(3*r^2+L1^2) + m1*(LC/2+L1/2)^2) 0; ...
    0 0 (1/12*m1*(3*r^2+L1^2) + m1*(LC/2+L1/2)^2 + (ALT)^2)];

inertia_leg_4_2 = [(1/2*m2*r^2 + m2*(ALT)^2) 0 0; ...
    0 (1/12*m2*(3*r^2+L2^2) + m2*(LC/2+L1+L2/2)^2) 0; ...
    0 0 (1/12*m2*(3*r^2+L2^2)) + m2*((LC/2+L1+L2/2)^2 + ALT^2)];

inertia_leg_4_3 = [(1/12*m3*(3*r^2+L3^2) + m3*(ALT/2)^2) 0 0; ...
    0 (1/2*m3*r^2 + m3*(LC/2+L1+L2)^2) 0; ...
    0 0 (1/12*m3*(3*r^2+L3^2) + m3*((LC/2+L1+L2)^2 + (ALT/2)^2))];

%Perna 5
inertia_leg_5_1 = [(1/2*m1*r^2 + m1*((CC/2)^2+ALT^2)) 0 0; ...
    0 (1/12*m1*(3*r^2+L1^2) + m1*((CC/2)^2 + (LC/2+L1/2)^2)) 0; ...
    0 0 (1/12*m1*(3*r^2+L1^2) + m1*((LC/2+L1/2)^2 + ALT^2))];

inertia_leg_5_2 = [(1/2*m2*r^2 + m2*((CC/2)^2+ALT^2)) 0 0; ...
    0 (1/12*m2*(3*r^2+L2^2) + m2*((CC/2)^2 + (LC/2+L1+L2/2)^2)) 0; ...
    0 0 (1/12*m2*(3*r^2+L2^2) + m2*((LC/2+L1+L2/2)^2 + ALT^2))];

inertia_leg_5_3 = [(1/12*m3*(3*r^2+L3^2) + m3*((CC/2)^2 + (ALT/2)^2)) 0 0; ...
    0 (1/2*m3*r^2 + m3*((CC/2)^2 + (LC/2+L1+L2)^2)) 0; ...
    0 0 (1/12*m3*(3*r^2+L3^2) + m3*((LC/2+L1+L2)^2 + (ALT/2)^2))];

%Perna 6
inertia_leg_6_1 = [(1/2*m1*r^2 + m1*((CC/2)^2+ALT^2)) 0 0; ...
    0 (1/12*m1*(3*r^2+L1^2) + m1*((CC/2)^2 + (LC/2+L1/2)^2)) 0; ...
    0 0 (1/12*m1*(3*r^2+L1^2) + m1*((LC/2+L1/2)^2 + ALT^2))];

inertia_leg_6_2 = [(1/2*m2*r^2 + m2*((CC/2)^2+ALT^2)) 0 0; ...
    0 (1/12*m2*(3*r^2+L2^2) + m2*((CC/2)^2 + (LC/2+L1+L2/2)^2)) 0; ...
    0 0 (1/12*m2*(3*r^2+L2^2) + m2*((LC/2+L1+L2/2)^2 + ALT^2))];

inertia_leg_6_3 = [(1/12*m3*(3*r^2+L3^2) + m3*((CC/2)^2 + (ALT/2)^2)) 0 0; ...
    0 (1/2*m3*r^2 + m3*((CC/2)^2 + (LC/2+L1+L2)^2)) 0; ...
    0 0 (1/12*m3*(3*r^2+L3^2) + m3*((LC/2+L1+L2)^2 + (ALT/2)^2))];

%-------------------------------Controladores-------------------------------
%hip
P1 = 1;
I1 = 0;
D1 = 0.01;
N1 = 100;

%knee
P2 = 1;
I2 = 0;
D2 = 0.01;
N2 = 100;

%foot
P3 = 1;
I3 = 0;
D3 = 0.01;
N3 = 100;

%-------------------Constantes-do-sistema-mola-amortecedo-------------------
Kx = 1302152;
Bx = 3423;

Ky = 170000;
By = 270000;

Kz = 1302152;
Bz = 3423;

m = 0.9;
```

# Anexo C

# Código do cálculo de estabilidade, servo motor mínimo e da GUIDE

```
global sample_time stop_time L1 L2 L3 P1 P2 P3 I1 I2 I3 D1 D2 D3 ...
N1 N2 N3 LC CC AC X_ini Y_ini Z_ini Kx Ky Kz Bx By Bz m m1 m2 m3 ...
m_body r g t p1 p2 p3 p4 p5 p6 CG i in on count count2 stable ...
marginally_stable instable gait leg_1_x leg_1_y leg_1_z leg_2_x ...
leg_2_y leg_2_z leg_3_x leg_3_y leg_3_z leg_4_x leg_4_y leg_4_z ...
leg_5_x leg_5_y leg_5_z leg_6_x leg_6_y leg_6_z hip_1 hip_2 hip_3 ...
hip_4 hip_5 hip_6 knee_1 knee_2 knee_3 knee_4 knee_5 knee_6 foot_1 ...
foot_2 foot_3 foot_4 foot_5 foot_6 Log n T Ls Fc beta HB theta1 ...
theta2 theta3 Posicao_X Posicao_Y Posicao_Z velocidade ...
Torque_m1_leg1 Torque_m2_leg1 Torque_m3_leg1 Torque_m1_leg2 ...
Torque_m2_leg2 Torque_m3_leg2 Torque_m1_leg3 Torque_m2_leg3 ...
Torque_m3_leg3 Torque_m1_leg4 Torque_m2_leg4 Torque_m3_leg4 ...
Torque_m1_leg5 Torque_m2_leg5 Torque_m3_leg5 Torque_m1_leg6 ...
Torque_m2_leg6 Torque_m3_leg6 Servo_Motor_Minimo

sample_time = str2num(get(handles.in_sample_time,'string'));
stop_time = str2num(get(handles.in_stop_time,'string'));

X_ini = str2num(get(handles.in_X_ini,'string'));
Y_ini = str2num(get(handles.in_Y_ini,'string'));
Z_ini = str2num(get(handles.in_Z_ini,'string'));

L1 = str2num(get(handles.in_L1,'string'));
L2 = str2num(get(handles.in_L2,'string'));
L3 = str2num(get(handles.in_L3,'string'));

LC = str2num(get(handles.in_LC,'string'));
CC = str2num(get(handles.in_CC,'string'));
AC = str2num(get(handles.in_AB,'string'));

m1 = str2num(get(handles.in_m1,'string'));
m2 = str2num(get(handles.in_m2,'string'));
m3 = str2num(get(handles.in_m3,'string'));
m_body = str2num(get(handles.in_m_body,'string'));

r = str2num(get(handles.in_r,'string'));
g = str2num(get(handles.in_g,'string'));

P1 = str2num(get(handles.in_P1,'string'));
```

```
P2 = str2num(get(handles.in_P2,'string'));
P3 = str2num(get(handles.in_P3,'string'));
I1 = str2num(get(handles.in_I1,'string'));
I2 = str2num(get(handles.in_I2,'string'));
I3 = str2num(get(handles.in_I3,'string'));
D1 = str2num(get(handles.in_D1,'string'));
D2 = str2num(get(handles.in_D2,'string'));
D3 = str2num(get(handles.in_D3,'string'));
N1 = str2num(get(handles.in_N1,'string'));
N2 = str2num(get(handles.in_N2,'string'));
N3 = str2num(get(handles.in_N3,'string'));

Kx = str2num(get(handles.in_Kx,'string'));
Ky = str2num(get(handles.in_Ky,'string'));
Kz = str2num(get(handles.in_Kz,'string'));
Bx = str2num(get(handles.in_Bx,'string'));
By = str2num(get(handles.in_By,'string'));
Bz = str2num(get(handles.in_Bz,'string'));
m = str2num(get(handles.in_m,'string'));

Fc = str2num(get(handles.in_Fc,'string'));
Ls = str2num(get(handles.in_Ls,'string'));
T = str2num(get(handles.in_T,'string'));
n = str2num(get(handles.in_n,'string'));
beta = str2num(get(handles.in_beta,'string'));
HB = str2num(get(handles.in_HB,'string'));

t = 0:sample_time:n*T;

[theta1,theta2,theta3,Posicao_X,Posicao_Y,Posicao_Z] = ...
Path_Planning(Fc,Ls,T,n,beta,L1,L2,L3,HB,sample_time);

gait = get(handles.gaits,'Value');

switch gait
    case 1
        % Metacronal-Wave-Gait
        A1 = 0/n;
        A2 = 0.5/n;
        A3 = 0.16/n;
        A4 = 0.66/n;
        A5 = 0.33/n;
        A6 = 0.83/n;
    case 2
        % Tetrapod Gait
        A1 = 0/n;
        A2 = 0.66/n;
        A3 = 0.66/n;
        A4 = 0.33/n;
        A5 = 0.33/n;
        A6 = 0/n;
    case 3
        % Tripod-Gait
        A1 = 0/n;
        A2 = 0.5/n;
        A3 = 0.5/n;
        A4 = 0/n;
        A5 = 0/n;
        A6 = 0.5/n;
end
%-------------------------Conversion-phase-to-shift-----------------------
F1 = round(A1*length(t));
F2 = round(A2*length(t));
F3 = round(A3*length(t));
F4 = round(A4*length(t));
F5 = round(A5*length(t));
F6 = round(A6*length(t));

%-----------------------------------Shift---------------------------------
shift = 1:1:length(t);
```

```
dt1 = circshift(shift,[0,F1]);
dt2 = circshift(shift,[0,F2]);
dt3 = circshift(shift,[0,F3]);
dt4 = circshift(shift,[0,F4]);
dt5 = circshift(shift,[0,F5]);
dt6 = circshift(shift,[0,F6]);

%-------------------------Initial-Position-------------------------
leg_1_x = Posicao_X(1,dt1(1,1)) + LC/2;
leg_1_y = Posicao_Y(1,dt1(1,1)) - HB;
leg_1_z = Posicao_Z(1,dt1(1,1)) + CC/2;

leg_2_x = -Posicao_X(1,dt2(1,1)) - LC/2;
leg_2_y = Posicao_Y(1,dt2(1,1)) - HB;
leg_2_z = Posicao_Z(1,dt2(1,1)) + CC/2;

leg_3_x = Posicao_X(1,dt3(1,1)) + LC/2;
leg_3_y = Posicao_Y(1,dt3(1,1)) - HB;
leg_3_z = Posicao_Z(1,dt3(1,1));

leg_4_x = -Posicao_X(1,dt4(1,1))- LC/2;
leg_4_y = Posicao_Y(1,dt4(1,1)) - HB;
leg_4_z = Posicao_Z(1,dt4(1,1));

leg_5_x = Posicao_X(1,dt5(1,1)) + LC/2;
leg_5_y = Posicao_Y(1,dt5(1,1)) - HB;
leg_5_z = Posicao_Z(1,dt5(1,1)) - CC/2;

leg_6_x = -Posicao_X(1,dt6(1,1)) - LC/2;
leg_6_y = Posicao_Y(1,dt6(1,1)) - HB;
leg_6_z = Posicao_Z(1,dt6(1,1)) - CC/2;

%-------------------------Joint-Values-------------------------
hip_1 = [];
hip_1(:,1) = t(1,:);
hip_1(:,2) = theta1(1,dt1);

hip_2 = [];
hip_2(:,1) = t(1,:);
hip_2(:,2) = theta1(1,dt2);

hip_3 = [];
hip_3(:,1) = t(1,:);
hip_3(:,2) = theta1(1,dt3);

hip_4 = [];
hip_4(:,1) = t(1,:);
hip_4(:,2) = theta1(1,dt4);

hip_5 = [];
hip_5(:,1) = t(1,:);
hip_5(:,2) = theta1(1,dt5);

hip_6 = [];
hip_6(:,1) = t(1,:);
hip_6(:,2) = theta1(1,dt6);

knee_1 = [];
knee_1(:,1) = t(1,:);
knee_1(:,2) = theta2(1,dt1);

knee_2 = [];
knee_2(:,1) = t(1,:);
knee_2(:,2) = theta2(1,dt2);

knee_3 = [];
knee_3(:,1) = t(1,:);
knee_3(:,2) = theta2(1,dt3);

knee_4 = [];
```

```
knee_4(:,1) = t(1,:);
knee_4(:,2) = theta2(1,dt4);

knee_5 = [];
knee_5(:,1) = t(1,:);
knee_5(:,2) = theta2(1,dt5);

knee_6 = [];
knee_6(:,1) = t(1,:);
knee_6(:,2) = theta2(1,dt6);

foot_1 = [];
foot_1(:,1) = t(1,:);
foot_1(:,2) = theta3(1,dt1)+90;

foot_2 = [];
foot_2(:,1) = t(1,:);
foot_2(:,2) = theta3(1,dt2)+90;

foot_3 = [];
foot_3(:,1) = t(1,:);
foot_3(:,2) = theta3(1,dt3)+90;

foot_4 = [];
foot_4(:,1) = t(1,:);
foot_4(:,2) = theta3(1,dt4)+90;

foot_5 = [];
foot_5(:,1) = t(1,:);
foot_5(:,2) = theta3(1,dt5)+90;

foot_6 = [];
foot_6(:,1) = t(1,:);
foot_6(:,2) = theta3(1,dt6)+90;

Log = [hip_1(:,2) knee_1(:,2) foot_1(:,2) ...
    hip_3(:,2) knee_3(:,2) foot_3(:,2) ...
    hip_5(:,2) knee_5(:,2) foot_5(:,2) ...
    hip_2(:,2) knee_2(:,2) foot_2(:,2)...
    hip_4(:,2) knee_4(:,2) foot_4(:,2) ...
    hip_6(:,2) knee_6(:,2) foot_6(:,2)];

i = 0;

set(handles.samples,'string', i);
set(handles.out_stable,'string', 'Waiting');
set(handles.out_marginally_stable,'string','Waiting');
set(handles.out_instable,'string','Waiting');

model = 'Robot_v1.mdl';
load_system(model);
sim(model);

count = 0;
count2 = 0;

samples = (1-(((length(t)-2) - i)/(length(t)-2)))*100;

set(handles.out_stable,'string', 'Calculating');
set(handles.out_marginally_stable,'string','Calculating');
set(handles.out_instable,'string','Calculating');

hold on
axis auto;
grid on

a = [];
l1 = [];
l2 = [];
l3 = [];
```

```
l4 = [];
l5 = [];
l6 = [];

for i = 1:1:(length(t)-2)

    x = [];
    y = [];

    if (p1(i,2) <= 0)
        x = [x;p1(i,1)];
        y = [y;p1(i,3)];
        l1 = plot(p1(i,1),p1(i,3),'ok');
    end

    if (p3(i,2) <= 0)
        x = [x;p3(i,1)];
        y = [y;p3(i,3)];
        l3 = plot(p3(i,1),p3(i,3),'ok');
    end

    if (p5(i,2) <= 0)
        x = [x;p5(i,1)];
        y = [y;p5(i,3)];
        l5 = plot(p5(i,1),p5(i,3),'ok');
    end

    if (p6(i,2) <= 0)
        x = [x;p6(i,1)];
        y = [y;p6(i,3)];
        l6 = plot(p6(i,1),p6(i,3),'ok');
    end

    if (p4(i,2) <= 0)
        x = [x;p4(i,1)];
        y = [y;p4(i,3)];
        l4 = plot(p4(i,1),p4(i,3),'ok');
    end

    if (p2(i,2) <= 0)
        x = [x;p2(i,1)];
        y = [y;p2(i,3)];
        l2 = plot(p2(i,1),p2(i,3),'ok');
    end

    if (size(x) ~= 0 & size(y) ~= 0)
        a = fill(x,y,'b');
        [in,on] = inpolygon(CG(i,1),CG(i,3),x,y);
        if (in > 0)
            count = count + 1;
        end
        if (on > 0)
            count2 = count2 + 1;
        end
    end

    c = plot(CG(i,1),CG(i,3),'*r');
    pause(0.001);

    delete(a);
    delete(c);
    delete(l1);
    delete(l2);
    delete(l3);
    delete(l4);
    delete(l5);
    delete(l6);
    x2 = x;
    y2 = y;
    clear x;
```

```
    clear y;
    samples = (1-(((length(t)-2) - i)/(length(t)-2)))*100;
    set(handles.samples,'string',samples);
end
hold off

Servo_Motor_Minimo = ...
10.19716212978*max([max(Torque_m1_leg1.signals.values) ...
max(Torque_m2_leg1.signals.values) max(Torque_m3_leg1.signals.values) ...
max(Torque_m1_leg2.signals.values) max(Torque_m2_leg2.signals.values) ...
max(Torque_m3_leg2.signals.values) max(Torque_m1_leg3.signals.values) ...
max(Torque_m2_leg3.signals.values) max(Torque_m3_leg3.signals.values) ...
max(Torque_m1_leg4.signals.values) max(Torque_m2_leg4.signals.values) ...
max(Torque_m3_leg4.signals.values) max(Torque_m1_leg5.signals.values) ...
max(Torque_m2_leg5.signals.values) max(Torque_m3_leg5.signals.values) ...
max(Torque_m1_leg6.signals.values) max(Torque_m2_leg6.signals.values) ...
max( Torque_m3_leg6.signals.values)]);

stable = (count/length(p1))*100;
marginally_stable = (count2/length(p1))*100;
instable = 100 - (stable + marginally_stable);

set(handles.samples,'string','Finish');
set(handles.out_stable,'string',stable);
set(handles.out_marginally_stable,'string',marginally_stable);
set(handles.out_instable,'string',instable);
```

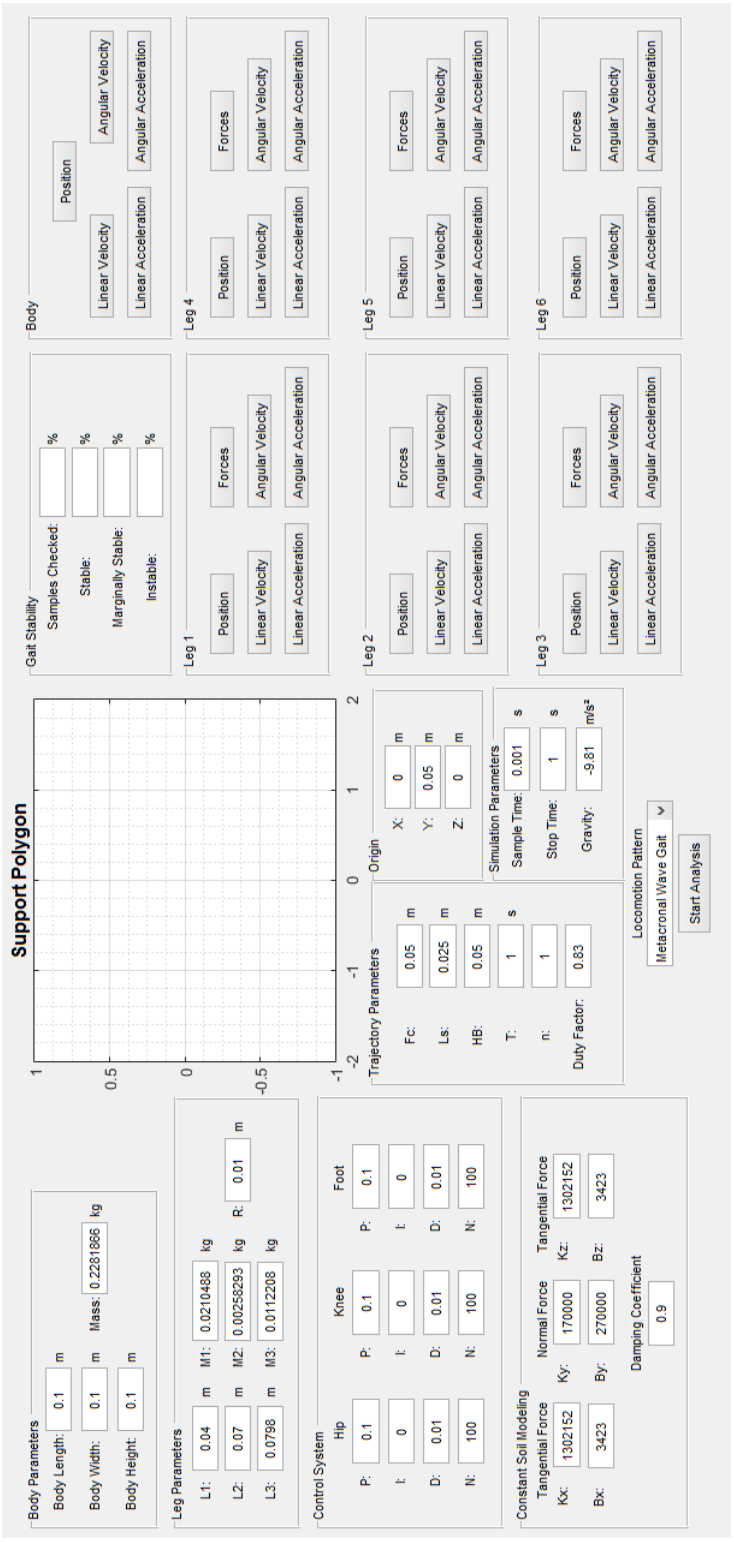



Figura C.1: Interface gráfica desenvolvida para a condução das simulações.

# Anexo D

# Desenho mecânico do robô hexápode

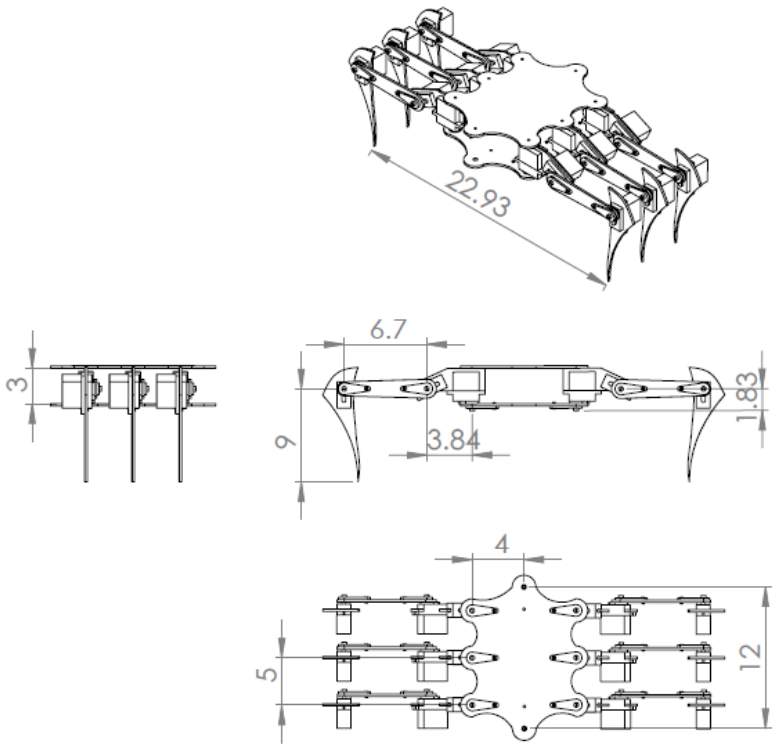



Figura D.1: Esquema mecânico do robô hexápode desenvolvido em Solidworks® (unidades em centímetros).

# Anexo E

# Código de controlo dos servo motores

```
import time

# Import the PCA9685 module.
import Adafruit_PCA9685

# Initialise the PCA9685 using the default address (0x40).
pwm = Adafruit_PCA9685.PCA9685()
pwm2 = Adafruit_PCA9685.PCA9685(0x41)

# Set frequency to 50hz, good for servos.
pwm.set_pwm_freq(50)
pwm2.set_pwm_freq(50)

def actuator(i):
        if i<=8:
                Theta = float(valores[i])
                Duty_Cycle = int(1.1666*Theta + 325)
                pwm.set_pwm(i, 0, Duty_Cycle)

        else:
                Theta = -1*float(valores[i])
                Duty_Cycle = int(1.1666*Theta + 325)
                i = i-9
                pwm2.set_pwm(i, 0, Duty_Cycle)

def resetar():
        for i in range(0, 9):
                pwm.set_pwm(i, 0, 325)
                pwm2.set_pwm(i, 0, 325)
```

```
menu = 0
while (menu >= 0 and menu < 4):
        print('Robot Hexapod: ')
        print('1 - Metacronal Wave Gait')
        print('2 - Tetrapod Gait')
        print('3 - Tripod Gait')
        print('4 - Exit')
        menu = input()
        cont = 1
        while (menu > 0 and menu < 4 and cont <= 4):
                if menu == 1:
                    ref_arquivo = open("/home/pi/log_raspi_metacronal_3.txt","r")

                if menu == 2:
                    ref_arquivo = open("/home/pi/log_raspi_tetrapod_3.txt","r")

                if menu == 3:
                    ref_arquivo = open("/home/pi/log_raspi_tripod_3.txt","r")

                if menu != 4:
                        for linha in ref_arquivo:
                                valores = linha.split()
                                for j in range(0, 18):
                                        actuator(j)
                                        time.sleep(0.005)
                                print('The Hexapod Robot is Moving')
                        ref_arquivo.close()
                cont +=1
        print('The Hexapod Robot is Stopped')

if menu < 0 or menu >= 4:
        resetar()
        time.sleep(1)
        print('The Robot was turned off.')
```