UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Projeto de um pacote ROS para a gestão de alocação de tarefas complexas em sistemas multi-robôs

Adriano Henrique Rossette Leite

Itajubá, 15 de outubro de 2017

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Adriano Henrique Rossette Leite

Projeto de um pacote ROS para a gestão de alocação de tarefas complexas em sistemas multi-robôs

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Área de Concentração: Automação e Sistemas Elétricos Industriais

Orientador: Prof. Dr. Guilherme Sousa Bastos

15 de outubro de 2017
Itajubá

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Projeto de um pacote ROS para a gestão de alocação de tarefas complexas em sistemas multi-robôs

Adriano Henrique Rossette Leite

Dissertação aprovada por banca examinadora em 01 de Fevereiro de 2018, conferindo ao autor o título de **Mestre em Ciências em Engenharia Elétrica.**

Banca Examinadora:
Prof. Dr. Guilherme Sousa Bastos (Orientador)
(Coorientador)
Prof. Dr. Convidado1
Prof. Dr. Convidado2
Prof. Dr. Convidado3

Itajubá
2018

Adriano Henrique Rossette Leite

Projeto de um pacote ROS para a gestão de alocação de tarefas complexas em sistemas multi-robôs/ Adriano Henrique Rossette Leite. – Itajubá, 15 de outubro de 2017-

35 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Guilherme Sousa Bastos

Dissertação (Mestrado)
Universidade Federal de Itajubá
Programa de Pós-Graduação em Engenharia Elétrica, 15 de outubro de 2017.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 07:181:009.3

Adriano Henrique Rossette Leite

Projeto de um pacote ROS para a gestão de alocação de tarefas complexas em sistemas multi-robôs

Dissertação submetida ao Programa de Pós-Graduação em Engenharia Elétrica como parte dos requisitos para obtenção do Título de Mestre em Ciências em Engenharia Elétrica.

Trabalho aprovado. Itajubá, 01 de Fevereiro de 2018:

Prof. Dr. Guilherme Sousa Bastos
Orientador

Coorientador

Prof. Dr. Convidado1

Prof. Dr. Convidado2

Prof. Dr. Convidado3

Itajubá
15 de outubro de 2017

Agradecimentos

À Deus ...

À meus amigos e familiares ...

Ao meu orientador ...

À banca examinadora, ...

Aos amigos e colegas do LRO, ...

À Capes pelo apoio financeiro durante estes 2 anos.

À Fapemig pelo financiamento do projeto de pesquisa TEC-APQ-00666-12, o qual possibilitou a compra do robô utilizado neste trabalho (Pioneer-3DX).

"Colocar a frase aqui."
(Autor)

Resumo

Esse trabalho apresenta...

Palavras-chaves: ABC. DEF. GHI.

Abstract

This work presents ...

Key-words: ABC. DEF. GHF.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

ABNT	Associação Brasileira de Normas Técnicas
CBR	Competição Brasileira de Robótica
LRO	Laboratório de Robótica
MAS	Multi-Agent System
MRS	Multi-Robot System
MRTA	Multi-Robot Task Allocation
P3DX	Adept MobileRobots Pioneer 3 DX
ROS	Robot Operating System
UNIFEI	Universidade Federal de Itajubá

Lista de símbolos

n	Número inteiro
t	Tempo
T	Período

Sumário

1 Introdução

Colocar texto aqui

1.1 Visão Geral

Colocar texto aqui

1.2 Organização do trabalho

Colocar texto aqui

2 Revisão teórica

Colocar texto aqui

2.1 ROS

2.1.1 ROS

The Robot Operation System (ROS) (1) is a framework that has encouraged the robotics researchers community to work together since its release.

Como o ROS tem incentivado essa comunidade a trabalhar em conjunto?

No que isso resultou?

Over the years, it has been noticed that: (1) the number of contributors (academic researchers and industry) and projects have increased; (2) the applications have became more sophisticated; (3) the hardness level of solved problems has ??? in different areas of robotics field; (4) the robotic industry has been more interested to contribute.

Many external libraries has been wrapped in ROS.

falar sobre ferramentas de avaliação e validação.

Como o ROS conseguiu fazer isso?

O que o ROS facilita?

Modular applications

Multi-language

ROS is flexible.

2.2 Sistemas Multi-robos

Colocar texto aqui

Whenever a group of robots aim to perform some collective behavior, it increases perfomance, realiability, and ability to resolve complex tasks. In this case, one of the most challengong problems is how to optimally assign a set of robots to a set of tasks in such a way that optimizes the overall system performance subject to a set of constraints. This problem is known as Multi Robot Task Allocation (MRTA).

2.2.1 Taxonomias

Colocar texto aqui

2.3 Alocação de Tarefas

Colocar texto aqui

2.4 Recursos

Colocar texto aqui

2.5 Sistema Multi-Agente

2.5.1 Agentes

According to (2):

an **agent** is anything that can be viewed as **perceiving** its environment through **sensors** and **acting** upon that environment through **effectors**.

Talking about **rational agent**:

the **performance measure** determines how successful an agent is. The complete perceptual history is called **percept sequence**.

So, it is desired to measure its performance over the long run. a rational agent is not omniscient.

Saying that, we cannot blame an agent for failing to take into account something it could not perceive, or for failing to take an action that it is incapable of taking.

In other words, what is rational at any given time depends on:

- The performance measure that defines degree of success;
- Everything that the agent has perceived so far, that is, the percept sequence;
- What the agent knowsabout the environment;
- and, finally, the actions that the agent can perform.

(2) defines an **ideal rational agent** as: *For each possible percept sequence, an ideal rational agent should do whatever action is expected to maximize its performance measure, an the basis of the evidence provided by the percept sequence and whatever built-in knowledge the agent has.*

ideal mapping: Specifying which action an agent ought to take in response to any given percept sequence provides a design for an ideal agent.

Talking about autonomy: If the agent's actions are based completely on built-in knownledge, such that it need pay no attention to its percepts, then we say that tthe agent lacks **autonomy**.

The job of AI is to design the **agent program**: a function that implements the agent mapping from percepts to actions. We assume this program will run on sort of computing device: the **architecture**.

There exists fours types of agent program:

- **Simple reflex agents**:
- **Agents that keep track of the world**:
- **Goal-based agents**:
- **Utility-based agents**:

This paper deals with utility-based agents!!!

To sum up, ...

2.5.2 Objetos

2.5.3 Ambiente

2.6 Sistema Multi-Robô

2.7 Alocação de Tarefas Multi-Robô

2.8 Arquiteturas de Alocação de Tarefas Multi-Robô

2.8.1 Arquiteturas baseadas em Mercado

- **Sold!**:
- **M+**:

2.8.2 Arquiteturas baseadas em Comportamento

- **Alliance**:

Resource-constrained project scheduling problem The RCPSP problem is a generalization of the production-specific Job-Shop, Flow-Shop and Open-Shop scheduling problems. Given

a set of q resources with given capacities, a set of q resources with given capacities, a network of precedence constraints between the activities, and for each activity and each resource the amount of the resource required by the activity over its execution,

the goal of the RCPSP problem is to find a schedule meeting all the constraints whose makespan (i.e., the time at which all activities are finished) is minimal.

Resource

There are two main classes of resources: *reusable resources* and *consumable resources*.

A reusable resource is "borrowed"by an action (task) duting its execution. It is released, unchanged, when the action (task) is completed or is interrupted. A reusable resource r has a total capacity Q_r

2.9 Introduction

A new generation of applications in robotics field has been growing.

Whenever a group of robots aim to perform some collective behavior, it increases perfomance, realiability, flexibility, scalability, and versatility to resolve complex tasks. In this case, one of the most challenging problems is how to optimally assign a set of robots to a set of tasks in such a way that optimizes the overall system performance subject to a set of constraints. This problem is known as Multi-Robot Task Allocation (MRTA).

the new generation of robotics application (3), where there exists multiple robots that are composed of heterogeneous interconnected hardware components

2.9.1 ROS

The Robot Operation System (ROS) (1) is a framework that has encouraged the robotics researchers community to work together since its release.

Firstly, ROS is flexible. An atomic ROS-based project, known as package, may be developed in multiple languages. Therefore, ROS developers can take advantage of each supported language, based on its runtime efficiency, reliability, resources, syntax, semantics, and existing documentation.

In addition, ROS is a tools-based software development framework. There are many tools that may be used during building and running time of ROS packages; such

as, get and set configuration parameters, visualize the peer-to-peer connection topology, measure bandwidth utilization, graphically plot message data, and so on. The usage of its tools is highly recommended, as long as they may ensure the stability and reliability of the developed packages despite their complexity.

Moreover, it has filled the gap in middleware services once existed in the new generation of robotics applications. As a middleware services provider, ROS (1) simplifies the development process, (2) supports communications and interoperability, (3) offers and facilitates often-needed robot services, and also provides (4) efficient utilization of available resources, (5) heterogeneity abstractions and (6) automatic recourse discovery and configutarion. In order to enclose all middleware requirements, ROS 2.0 attempts to support embedded components and low-resource-devices, as well.

2.9.2 Comparing DAIs, MASs, and MRSs

Distributed Artificial Intelligence (DAI) has existed as a subfield of AI for less than two decades. DAI is concerned with systems that consist of multiple independent entities that interact in a domain. Traditionally, DAI has been divided into two sub-disciplines: Distributed Problem Solving (DPS) focuses on the information management aspects of systems with several components working together towards a common goal; Multiagent Systems (MAS) deals with behavior management in collections of several independent entities, or agents. This survey of MAS is intended to serve as an introduction to the field and as an organizational framework. A series of general multiagent scenarios are presented. For each scenario, the issues that arise are described along with a sampling of the techniques that exist to deal with them. The presented techniques are not exhaustive, but they highlight how multiagent systems can be and have been used to build complex systems. When options exist, the techniques presented are biased towards machine learning approaches. Additional opportunities for applying machine learning to MAS are highlighted and robotic soccer is presented as an appropriate test bed for MAS. This survey does not focus exclusively on robotic systems. However, we believe that much of the prior research in non-robotic MAS is relevant to robotic MAS, and we explicitly discuss several robotic MAS, including all of those presented in this issue (4).

3 Sistemas Multi-Robôs

Colocar texto aqui

3.1 Taxonomia

Colocar texto aqui

3.2 Alocação de Tarefas

Colocar texto aqui

4 Alocação de Tarefas em Sistemas Multi-Robôs

Colocar texto aqui

4.1 Definição Formal

Colocar texto aqui

4.2 Taxonomias

Colocar texto aqui

4.3 Arquiteturas

Colocar texto aqui

4.3.1 ALLIANCE

Esta é uma arquitetura totalmente distribuída, tolerante à falhas, que visa atingir controle cooperativo e atender os requisitos de uma missão de robôs heterogêneos (5). Cada robô é modelado usando uma aproximação baseada em comportamentos. A partir do estado do ambiente e dos outros robôs cooperadores, uma configuração de comportamento é selecionada conforme sua respectiva função de realização de tarefa na camada de alto nível de abstração. Cada configuração de comportamento permite controlar os atuadores do robô em questão de uma modo diferente.

Sejam $R = \{r_1, r_2, ..., r\}$, o conjunto de n robôs heterogêneos, e $T = \{t_1, t_2, ..., t_m\}$, o conjunto de m subtarefas independentes que compõem uma dada missão. Na arquitetura ALLIANCE, cada robô r_i possui um conjunto de p configurações de comportamento, dado por $A = \{a_{i1}, a_{i2}, ...a_{ip}\}$. Cada configuração de comportamento fornece ao seu robô uma função de realização de tarefa em alto nível, conforme definido em (6).

melhorar esse parágrafo Além disso, o conjunto de n funções $H_k = \{h_1(a_{1k}), h_2(a_{2k}), h_n(a_{nk}$ em que $h_i(a_{ik})$ é uma função que retorna a tarefa em T que o robô r_i executa quando sua configuração de ativação a_{ik} é ativada.

A seguir, serão discutidas as funções necessárias para a ativação de uma dada configuração de comportamento a_{ij} do robô r_i para a execução da tarefa $h_i(a_{ij})$.

$$aplicável_{ij}(t) = \begin{cases} 1 & \text{se o módulo de } \textit{feedback} \text{ sensorial da configuração de comportamento } a_{ij} \text{ do robô } r_i \text{ indicar que esta configuração é aplicável mediante ao estado atual do ambiente no instante } t; \\ 0 & \text{caso contrário.} \end{cases} \tag{4.1}$$

$$recebida_{ij}(k, t_1, t_2) = \begin{cases} 1 & \text{se o robô } r_i \text{ recebeu mensagem do robô } r_k \text{ referente à tarefa } h_i(a_{ij}) \text{ dentro do intervalo de tempo } (t_1, t_2), \text{ em que } t_1 < t_2; \\ 0 & \text{caso contrário.} \end{cases} \tag{4.2}$$

$$inibida_{ij}(t) = \begin{cases} 1 & \text{se outra configuração de comportamento } a_{ik} \text{ (com } k \neq j) \text{ está ativa no robô } r_i \text{ no instante } t; \\ 0 & \text{caso contrário.} \end{cases} \tag{4.3}$$

$$impaciência_{ij}(t) = \begin{cases} \min\limits_{x} \delta_{slow_{ij}}(x, t) & \text{se } recebida_{ij}(x, t - \tau_i, t) \ \wedge \ \neg recebida_{ij}(x, 0, t - \phi_{ij}(x, t)); \\ \delta_{fast_{ij}}(t) & \text{caso contrário.} \end{cases} \tag{4.4}$$

$$reiniciar_{ij}(t) = \begin{cases} 1 & \text{se } \exists x, recebida_{ij}(x, t - \delta t, t) \ \wedge \ \neg recebida_{ij}(x, 0, t - \delta t)), \text{ onde } \delta t \text{ é o tempo desde a última verificação de comunicação;} \\ 0 & \text{caso contrário.} \end{cases} \tag{4.5}$$

$$ativa_{ij}(\Delta t, t) = \begin{cases} 1 & \text{se a configuração de comportamento } a_{ij} \text{ do robô } r_i \text{ estiver ativa por mais de } \Delta t \text{ unidades de tempo no instante } t; \\ 0 & \text{caso contrário.} \end{cases} \tag{4.6}$$

$$aquiescente_{ij}(t) = \begin{cases} 1 & \text{se } (ativa_{ij}(\psi_{ij}(t), t) \ \wedge \ \exists x, recebida_{ij}(x, t - \tau_i, t)) \vee ativa_{ij}(\lambda_{ij}(t), t); \\ 0 & \text{caso contrário.} \end{cases} \tag{4.7}$$

$$
\begin{aligned}
motivação_{ij}(0) &= 0 \\
motivação_{ij}(t) &= (motivação_{ij}(t-\delta t) + impaciência_{ij}(t))\times \\
&\quad aplicável_{ij}(t)\times \\
&\quad inibida_{ij}(t)\times \\
&\quad reiniciar_{ij}(t)\times \\
&\quad aquiescente_{ij}(t).
\end{aligned}
\tag{4.8}
$$

5 Recursos

Existem duas classes de recursos, basicamente: aqueles que podem ser usados novamente, os quais são denominados *recursos reusáveis*; e aqueles que são descartáveis, denominados *recursos consumíveis* (7). Cada classe de recurso é caracterizada por um *perfil*. O perfil de um recurso r é uma função do tempo, $z_r(t)$, que define sua quantidade instantânea.

A seguir, cada classe de recurso será definida, bem como, será detalhado o perfil de cada uma delas.

5.1 Recursos Reutilizáveis

Um recurso reusável é "emprestado" por uma tarefa durante sua execução. E quando ela é interrompida ou finalizada, esse recurso é liberado sem alterações. Assim, um recurso reusável r tem uma capacidade total Q_r e uma quantidade corrente $z_r(t) \in [0; Q_r]$.

Seja uma tarefa a que requer durante a sua execução uma quantidade q do recurso r. Ao ser iniciada, no tempo $s(a)$, a quantidade corrente de r é diminuída em um montante q. Entretanto, quando a é finalizada, no tempo $e(a)$, a quantidade corrente de r é aumentada pelo mesmo montante, q. Portanto, um recurso reusável possui um perfil característico conforme mostra a Figura 1. *(explicar Figura 1 melhor (usar a palavra "evolução").)*

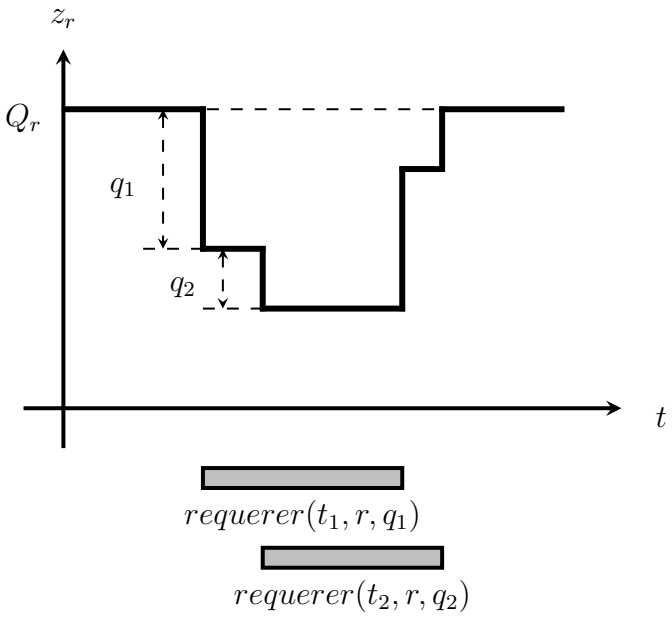


Figura 1 – Perfil característico de um recurso reusável.

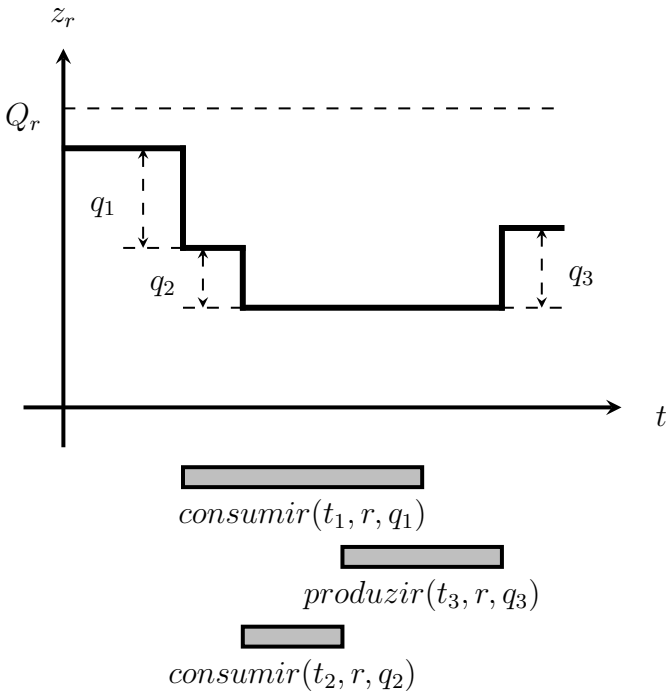


Figura 2 – Perfil característico de um recurso consumível.

5.2 Recursos Consumíveis

Um recurso consumível pode ser produzido ou consumido durante a execução de uma tarefa. Esta classe de recurso pode ser modelada como um reservatório de capacidade máxima limitada Q_r e nível (quantidade) corrente $z_r(t) \in [0; Q_r]$.

Seja, pois, uma tarefa a que produz durante a sua execução uma quantidade q do recurso r. Quando iniciada, em $s(a)$, aumenta um montante q do seu nível $z_r(t)$ ao longo do tempo. Essa produção é modelada por uma função dependente do tempo, crescente no intervalo temporal $[s(a); e(a)]$. Seja, agora, uma tarefa a que consome durante a sua execução uma quantidade q do recurso r. Quando iniciada, no instante $s(a)$, reduz um montante q do seu nível $z_r(t)$ ao longo do tempo. Essa redução/consumo é modelada como uma função do tempo, decrescente no intervalo de tempo $[s(a); e(a)]$. Portanto, um recurso consumível possui um perfil característico conforme mostrado na Figura 2. *(explicar Figura 2 melhor (usar a palavra "evolução").)*

(dar exemplos gráficos de algumas funções de consumo/produção: step, pulse, linear e exponential). A

5.3 Tipos de Recurso

Recursos possuem um tipo, podendo ele ser: (1) contínuo, (2) discreto ou (3) unário.

Primeiramente, em *recursos contínuos*, a capacidade total do recurso é definida por um número pertencente ao conjunto dos números reais estritamente positivos, enquanto sua quantidade corrente é uma representação numérica que pertence ao conjunto dos números reais não-negativos. Assim temos,

$$z_r : t \in \mathbb{R}_+ \rightarrow z \in [0; Q_r] \subset \mathbb{R}_+ \mid Q_r \in \mathbb{R}_+^* \tag{5.1}$$

Exemplificando ... *(dar exemplo(s) de recursos reusáveis contínuos e recursos consumíveis contínuos. Falar do tipo de funções tbm: step, pulse, linear e exponential)*

Recursos discretos possuem capacidade total definida por um número inteiro estritamente positivo, isto é, um número natural, e quantidade corrente por um número inteiro não-negativo, obtendo:

$$z_r : t \in \mathbb{R}_+ \rightarrow z \in [0; Q_r] \subset \mathbb{Z}_+ \mid Q_r \in \mathbb{N} \tag{5.2}$$

Exemplificando ... *(dar exemplo(s) de recursos reusáveis discretos e recursos consumíveis discretos. Falar do tipo de funções tbm: step, pulse, linear e exponential)*

E, finalmente, um *recurso unário* sempre possui capacidade total igual à 1 e sua quantidade corrente pode assumir os valores 0. Com isso, podemos concluir que a quantidade corrente do recurso informa sua disponibilidade ao longo do tempo: quando 0, o recurso se encontra indisponível; e, quando 1, o recurso está disponível. Em outras palavras,

$$z_r : t \in \mathbb{R}_+ \rightarrow z \in \{0; 1\} \quad \text{e} \quad Q_r = 1 \tag{5.3}$$

Exemplificando ... *(dar exemplo(s) de recursos reusáveis unários e recursos consumíveis unários. Falar do tipo de funções tbm: step, pulse, linear e exponential)*

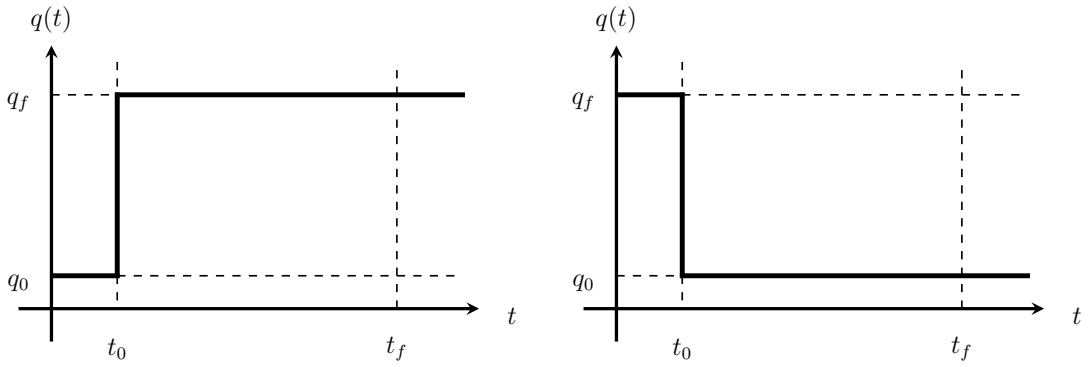
... *(dar de uma aplicação com várias tarefas que utilizam recursos de classes e tipos variados.)*

Apêndices

APÊNDICE A – Funções Temporais

A.1 Funções Degraus

A Figura 3 mostra duas funções degraus: ascendente 3a e descendente 3b.



(a) Função degrau ascendente.

(b) Função degrau descendente.

Figura 3 – Funções degraus.

$$q(t) = \begin{cases} q_0, & t \leq t_0 \\ q_f, & t > t_0 \end{cases} \tag{A.1}$$

$$q(t) = \begin{cases} q_f, & t \leq t_0 \\ q_0, & t > t_0 \end{cases} \tag{A.2}$$

A.2 Funções Pulsos

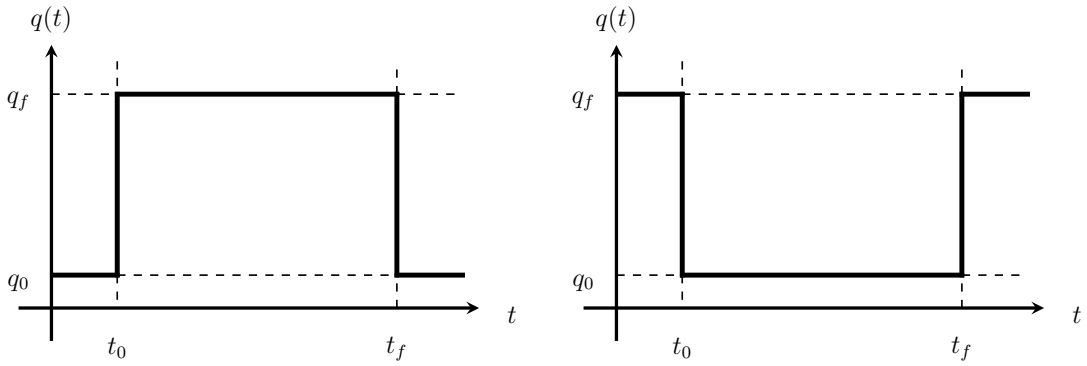
A Figura 4 mostra duas funções pulsos: ascendente 4a e descendente 4b.

$$q(t) = \begin{cases} q_0, & t \leq t_0 \\ q_f, & t > t_0 \end{cases} \tag{A.3}$$

$$f(t) = \begin{cases} q_f, & t \leq t_0 \\ q_0, & t > t_0 \end{cases} \tag{A.4}$$

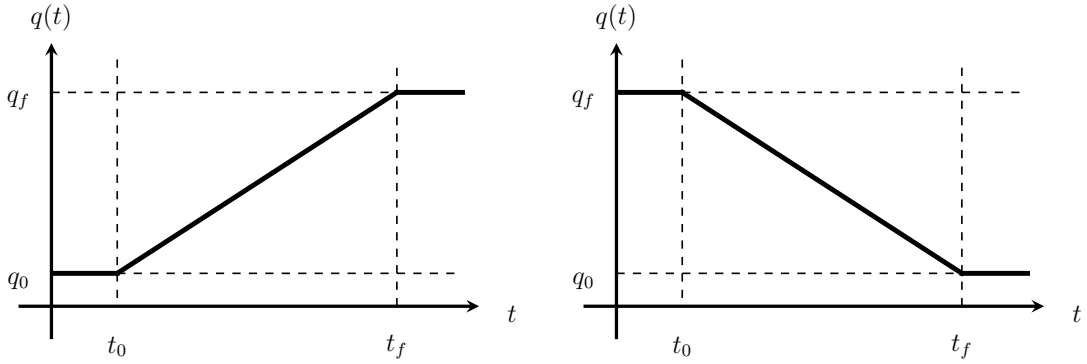
A.3 Funções Lineares

A Figura 5 mostra duas funções lineares: ascendente 5a e descendente 5b.



(a) Função pulso ascendente. (b) Função degrau descendente.

Figura 4 – Funções pulsos.



(a) Função linear ascendente. (b) Função linear descendente.

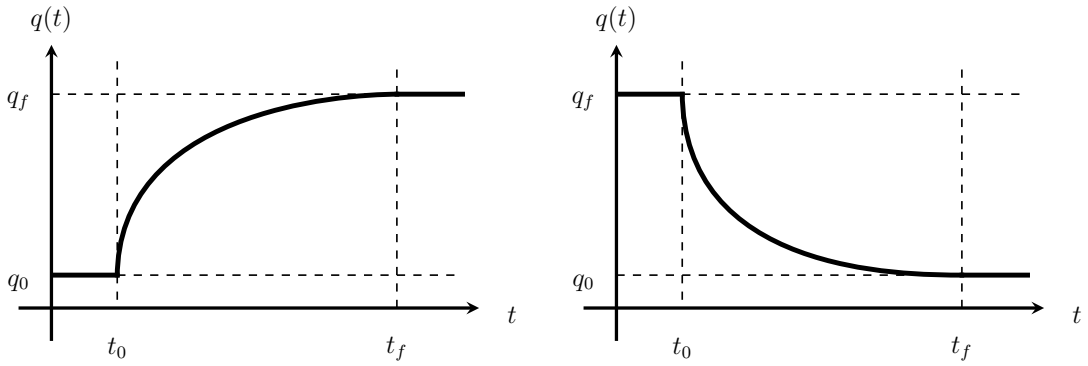
Figura 5 – Funções lineares.

$$q(t) = \begin{cases} q_0, & t \leq t_0 \\ (q_f - q_0)\dfrac{t - t_0}{t_f - t_0} + q_0, & t_0 < t \leq t_f \\ q_f, & t > t_f \end{cases} \tag{A.5}$$

$$q(t) = \begin{cases} q_f & t \leq t_0 \\ (q_0 - q_f)\dfrac{t - t_0}{t_f - t_0} + q_f, & t_0 < t \leq t_f \\ q_0 & t > t_f \end{cases} \tag{A.6}$$

A.4 Função Exponenciais

A Figura 6 mostra duas funções exponenciais: ascendente 6a e descendente 6b.



(a) Função exponencial ascendente. (b) Função exponencial descendente.

Figura 6 – Funções exponenciais.

$$q(t) = \begin{cases} q_0 & t \leq t_0 \\ q_f - (q_f - q_0)e^{-K\dfrac{t - t_0}{t_f - t_0}}, & t_0 < t \leq t_f \\ q_f & t > t_f \end{cases} \tag{A.7}$$

$$q(t) = \begin{cases} q_f & t \leq t_0 \\ q_0 - (q_0 - q_f)e^{-K\dfrac{t - t_0}{t_f - t_0}}, & t_0 < t \leq t_f \\ q_0 & t > t_f \end{cases} \tag{A.8}$$

Anexos

ANEXO A – Artigo publicado

Sed mattis, erat sit amet gravida malesuada, elit augue egestas diam, tempus scelerisque nunc nisl vitae libero. Sed consequat feugiat massa. Nunc porta, eros in eleifend varius, erat leo rutrum dui, non convallis lectus orci ut nibh. Sed lorem massa, nonummy quis, egestas id, condimentum at, nisl. Maecenas at nibh. Aliquam et augue at nunc pellentesque ullamcorper. Duis nisl nibh, laoreet suscipit, convallis ut, rutrum id, enim. Phasellus odio. Nulla nulla elit, molestie non, scelerisque at, vestibulum eu, nulla. Ut odio nisl, facilisis id, mollis et, scelerisque nec, enim. Aenean sem leo, pellentesque sit amet, scelerisque sit amet, vehicula pellentesque, sapien.

Referências

1 QUIGLEY, M. et al. Ros: an open-source robot operating system. In: KOBE. *ICRA workshop on open source software*. [S.l.], 2009. v. 3, p. 5. 17, 20

2 RUSSELL, S.; NORVIG, P.; INTELLIGENCE, A. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, Citeseer, v. 25, 1995. 18

3 MOHAMED, N.; AL-JAROODI, J.; JAWHAR, I. Middleware for robotics: A survey. In: IEEE. *Robotics, Automation and Mechatronics, 2008 IEEE Conference on*. [S.l.], 2008. p. 736–742. 20

4 STONE, P.; VELOSO, M. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, v. 8, n. 3, p. 345–383, 2000. 21

5 PARKER, L. E. Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE transactions on robotics and automation*, IEEE, v. 14, n. 2, p. 220–240, 1998. 23

6 BROOKS, R. A robust layered control system for a mobile robot. *IEEE journal on robotics and automation*, IEEE, v. 2, n. 1, p. 14–23, 1986. 23

7 GHALLAB, M.; NAU, D.; TRAVERSO, P. *Automated Planning: theory and practice*. [S.l.]: Elsevier, 2004. 26