Faculdade de Engenharia da Universidade do Porto

Controlo das trajetórias de um robô móvel de alto desempenho

Sandro Augusto Costa Magalhães

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Orientador: Professor Doutor António Paulo Gomes Mendes Moreira
Co-orientador: Professor Doutor Paulo José Cerqueira Gomes da Costa

16 de Julho de 2017

Resumo

Desde o início da robótica móvel que o problema de controlo das trajetórias a serem efetuadas pelo robô se tornou um problema, nomeadamente quando este está integrado num ambiente de elevada dinâmica e se pretende que ele circule a elevadas velocidades, em perseguição de um determinado alvo, como é o caso do futebol robótico.

Nesse sentido, esta tese pretende explorar este problema, avaliando o comportamento entre três estruturas de controlo para o seguimento de trajetórias. Numa primeira fase, aborda-se o controlo reativo, que tem por base a anulação do erro através de um controlador PD com afinação por protótipo de Bessel, recorrendo aos algoritmos de seguimento de trajetórias *go to position* e *follow line*. De seguida, implementa-se o modelo de controlo preditivo não linear usando um modelo simplificado do sistema, modelado por uma rampa de aceleração.

Todos os controladores foram projetados e afinados no simulador SimTwo após a parametrização do robô, por métodos experimentais e por métodos de aproximação numérica, por forma a que o robô simulado fosse o mais idêntico possível ao robô real.

Entre todos os controladores, pode-se concluir que há uma clara vantagem na utilização do modelo de controlo preditivo quando o objetivo é a anulação do erro de seguimento. No entanto, este também demonstra ser mais dispendioso computacionalmente, o que pode comprometer o problema de controlo quando há restrições no tempo de processamento.

Palavras-chave: robótica móvel omnidirecional, modelação robô móvel, protótipo de Bessel, seguimento de trajetórias, modelo de controlo preditivo, controlo clássico

Abstract

Since the begining of mobile robotics that the issue of trajectory control has had a particular matter, specially when the robots are integrated in a high dynamic environment where it is required that they persuit a target at high speed, such as in the robot soccer.

By this way, this thesis intends to explore this matter, evaluating the behaviour between three control structures to trajectory tracking. In a first step, the recative control is approched, implementing go to position and follow line trajectory tracking algorithms, where the error is cancelled by a PD controller tuned with the Bessel prototype. Next, it is implemented the non-linear model predictive control, using a simplified model of the system based on the acceleration ramp.

All the controller were designed and tuned in the simulator SimTwo, after tunning the robot model parameters, by experimental methods and by approximation with numerical methods, so that the robot model was identical as possible to the real robot.

Among all the controllers, it can be concluded that there is a clear advantage in using the model predictive control when the purpose it is to cancel the follow-up error. However, this also shows to be more computacionally expensive, which can compromise the control problem when there are processing time restrictions.

Keywords: holonomic mobile robot, mobile robot modelling, Bessel prototype, trajectory tracking, model predictive control, classic control

Agradecimentos

Antes de mais, queria começar por agradecer aos meus orientadores, ao Professor Doutor António Paulo Moreira e ao Professor Doutor Paulo Costa pela sua disponibilidade, paciência, apoio e por todo o conhecimento que partilharam comigo ao longo destes últimos meses. É a eles que devo grande parte do trabalho e sucesso que consegui atingir ao longo desta tese.

De seguida, queria agradecer aos meus colegas de laboratório, Pedro Guedes, Pedro Moura, Emanuel, Gonçalo e Sérgio, que partilharam comigo a alegria do momento, atenuando as dificuldades e partilhando, também, a sua opinião por forma a conseguir melhores soluções. Foram eles que permitiram as longas horas de trabalho, sem que o cansaço me pegasse, deixando-me chegar mais longe.

Gostaria ainda de agradecer aos meus colegas da equipa de equipa de futebol robótico 5DPO, nomeadamente ao Pedro Relvas, ao Tiago Raul, ao Eurico e ao Francisco, pela sua disponibilidade e preocupação, por me ajudarem a resolver os mais diversos problemas e por todos os momentos que disponibilizaram para me ajudar e explicar a forma como funciona esta equipa e como toda a arquitetura está construída.

De seguida, queria agradecer a todos os meus colegas de curso, especialmente ao Sérgio, ao Tiago Mendonça, ao João Mesquita, ao Telmo e ao Tiago Torres, pelo apoio e pelos momentos de diversão, como as noites de S. João bem passadas, os jantares e todos os outros momentos. Gostaria ainda de agradecer a todos os outros professores do curso pela sua disponibilidade e por todo o conhecimento que partilharam.

Fora do ambiente académico, queria agradecer ao meus amigos e colegas dos escuteiros (que são demais para se enumerar), por todos os momentos bem passados, fins de semana, noites e outros momentos; pelos jogos, desafios e reflexões que me deixaram pensar mais alto e perceber que o céu é o limite. Estes momentos, fizeram-me nunca desistir, saber impelir a minha própria canoa, evitando os diferentes escolhos.

Queria ainda agradecer ao meus instrutores de karaté, Sensei Diana, Sensei Victor, Sensei Álvaro e Sensei Paulo, por todo o trabalho físico e técnico que me ajudaram a desenvolver, foi durante estes treinos que consegui libertar todas as minhas tensões e elevar o meu poder de concentração, libertando-me dos problemas que me ocupavam no momento.

Com um ênfase especial, queria agradecer à minha família, nomeadamente aos meus pais, António e Elisabete, e ao meu irmão, Leandro. Foram eles que me ajudaram a ultrapassar todas as minhas dificuldades, me aconchegaram nas minhas mágoas e me fizeram olhar a vida com um novo olhar e me guiaram nas diferentes direções da vida, dando-me sempre a liberdade de escolher, tornando verdadeiro o conceito de família.

Por fim, queria agradecer a todas as outras pessoas que de uma forma ou de outra contribuíram para o meu desempenho ao longo de todos estes anos.

Sandro Magalhães

"I have always believed:
That if there is the right spirit, we can kick out the "im" from "impossible"."

Robert Baden-Powell

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

DC	*Direct current* (corrente direta ou contínua)
IDE	*Integrated development environment* (Ambiente de desenvolvimento integrado)
IMU	*Inertial measurement unit* (unidade de medição da inércia)
IRV	*Ideal reference velocities*
MPC	Modelo de controlo preditivo
NMPC	Modelo de controlo preditivo não linear
MSL	*Medium size league*
ODE	*Open dynamics engine*
PD	Proporcional e derivativo
PID	Proporcional, integral e derivativo
RPROP	*Resilient propagation*
SSL	*Small size league*
UDP	*User datagram protocol*
XML	*Extensible markup language*

s	variável complexa da transformada de Laplace
v_i	elemento i do vetor $\mathbf{v}$
$M(i,j)$	denota o elemento ij da matriz M, a utilização de $\cdots$ denota todos os elementos dessa linha ou coluna
$\bar{(\cdot)}$	conjugado do valor complexo $(\cdot)$
$(\cdot)^T$	denota a transposta de $(\cdot)$
$\|(\cdot)\|$	valor absoluto de $(\cdot)$
$\|\|\mathbf{v}\|\|_l$	norma l de $\mathbf{v}$, na ausência de l, assume-se $l = 2$, ou seja, a norma euclidiana
$\hat{x}$	valor estimado de x
$\hat{x}(t+j\|t)$	valor estimado de x no instante $t+j$ com a informação disponível no instante t

Notação

$\mathbf{A}(\cdot)$	letras a negrito e maiúsculas denotam matrizes polinomiais
M	letras em itálico e maiúsculas denotam matrizes reais
$\mathbf{b}$	letras a negrito e minúsculas denotam vetores reais
b	letra minúscula denota um valor escalar

Capítulo 1

Introdução

1.1 Contexto e motivação

A robótica móvel tem despertado um elevado interesse em diversas aplicações, como é o caso das aplicações militares, vigilância, busca e salvamento, limpeza, inspeção e até mesmo em áreas com um carácter mais distinto como o futebol robótico, que ao longo dos anos diversos investigadores têm usado para testar novas abordagens científicas para a resolução de diversos problemas nesta área. A elevada procura deste tipo de tecnologia provém de diversas necessidades da humanidade como as de segurança, saúde ou governamentais, tal como a procura pela redução do risco de vida ou pela melhoria das condições de trabalho. Há ainda outros tipos de interesse como é o caso da procura por uma maior vantagem económica, quer seja esta proveniente de produtos com melhor qualidade ou do incremento da produção.

Embora esta tecnologia se sintetize com uma redução para três graus de liberdade dos robôs convencionais, a verdade é que as suas situações de controlo se tornam bem mais exigentes do que, por exemplo, um manipulador robótico. Com esta tecnologia, deixa de se ter um ambiente controlado, onde todas a situações foram previamente pensadas e prevenidas com os devidos sistemas de segurança, passando agora a ter um ambiente livre e partilhado com pessoas, objetos e outros robôs.

Estes meios estão sujeitos a mudanças constantes e espera-se que os robôs sejam capazes de agir de uma forma rápida e segura. Demonstra-se, portanto, que a navegação de robôs móveis não é uma tarefa simples e implica um planeamento cuidado das trajetórias, por forma a conseguir-se evitar colisões e ter precisão na execução dessas trajetórias mesmo quando executadas a alta velocidade.

Para conseguir atingir as expectativas que lhes são impostas, os robôs móveis têm de fundir os dados que lhes são fornecidos pelos seus diferentes sensores e atuadores, que interagem com o ambiente. Assim, estes tipicamente possuem uma estrutura em camadas que permite um processamento separado da informação e uma posterior fusão que é garantida por uma conjugação de *software* mais *hardware*.

Além do mais, existem ainda diversas tipologias de robôs móveis passíveis de serem utilizadas, como é o caso dos robôs de tração diferencial ou os robôs omnidirecionais. São estes últimos aqueles que despertam mais interesse para esta dissertação, nomeadamente os robôs móveis com três rodas omnidirecionais, quer pela maior simplicidade do movimento, quer pela maior liberdade em se movimentarem, uma vez que se conseguem mover facilmente em qualquer direção, ultrapassando assim as restrições de espaço para manobras e permitindo maiores velocidades.

1.2 Objetivos

Tendo em conta os assuntos explicitados na secção 1.1, com esta dissertação, pretende-se o desenvolvimento de um algoritmo que seja capaz de movimentar um robô móvel com rodas omnidirecionais em elevadas velocidades, com o desvio de obstáculos dinâmicos e com a interseção de um alvo igualmente dinâmico.

Portanto, consegue-se desde já definir os dois principais indicadores de sucesso da dissertação como o controlo preciso de trajetórias e a estimativa da posição e da velocidade dos obstáculos móveis e do alvo a intersetar.

Com base na descrição efetuada, pode-se definir como principais etapas da dissertação as seguintes:

1. Calibração do simulador para corresponder aos robôs reais.
2. Implementação de um algoritmo de desvio de obstáculos.
3. Implementação do sistema de controlo de trajetórias para intersecção do alvo.

Como ambiente de testes e desenvolvimento, utilizar-se-á um dos robôs omnidirecionais da equipa de futebol robótico 5DPO FEUP/INESC TEC.

Capítulo 2

Revisão Bibliográfica

Ao longo dos anos, muitos são os estudos que se têm vindo a desenvolver no âmbito de encontrar uma solução ótima para o problema de planeamento, geração e seguimento de trajetórias. Assim, a complexidade do tema permite que este assunto seja subdivisível em dois subtemas claramente distintos: o controlo de trajetórias para interseção de alvos e o planeamento de trajetórias para evitar colisões com obstáculos dinâmicos.

2.1 Controlo de trajetórias

O controlador de trajetórias tem como papel principal seguir a trajetória realizada pelo robô, garantindo que este cumpre, com erro mínimo, a trajetória de referência à velocidade pretendida. Esta trajetória é comummente definida por um conjunto de pontos (x, y) mapeados no espaço.

Existem várias estratégias que poderão ser utilizadas para o seguimento de trajetórias, no entanto, Scolari [3] sugere duas estratégias de controlo: controlo reativo e controlo preditivo. O controlo reativo tem por base a realimentação do estado do robô, tendo em conta a sua posição atual e a referência. Por outro lado, o controlo preditivo, pretende controlar a posição futura do robô,por forma a evitar mudanças bruscas de direção ou outras situações que possam comprometer a trajetória do robô. Portanto, o controlo preditivo é baseado num modelo de antecipação que depende do modelo do processo a controlar (na situação em estudo, do modelo dinâmico e cinemático do robô) para ajustar o sinal de entrada, minimizando uma determinada função custo.

2.1.1 Controlo reativo

Estratégias relativamente eficientes e complexas para resolver o problema inicialmente enunciado, nomeadamente no âmbito de seguimento de trajetórias, remontam aos finais dos anos 80 quando Muir e Neuman [1] procuraram modelar cinematicamente um robô omnidirecional de quatro rodas *mecanum* que designaram como Uranus. Eles utilizaram este modelo para estimar a posição do robô, detetar derrapagens e realizar um controlo

por realimentação (tal como demonstra a solução proposta pelos autores da figura 2.1). A posição do robô era determinada por odometria, integrando as velocidades das rodas e o controlador comparava os resultados da odometria com as equações da cinemática, para detetar eventuais perdas de tração.

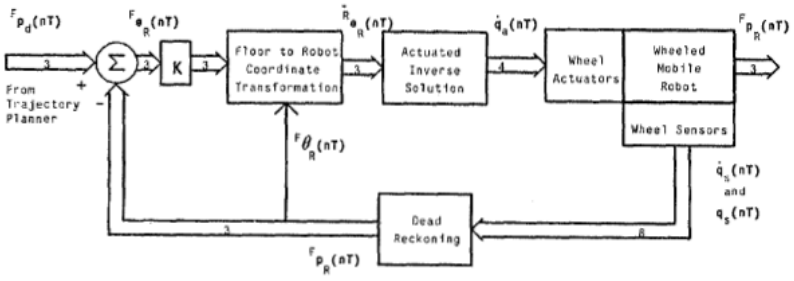


Figura 2.1: Solução proposta por Muir e Neuman [1]

Mais tarde, Watanabe *et. al* [2] mantêm uma sugestão de resolução do problema de seguimento de trajetórias com base em estruturas de controlo realimentadas, mas com uma estrutura de controlo do tipo *resolved-acceleration* com um controlador do tipo proporcional-derivativo (PD) e outro do tipo proporcional-integral (PI), tal como demonstra a estrutura de controlo proposta na figura 2.2.

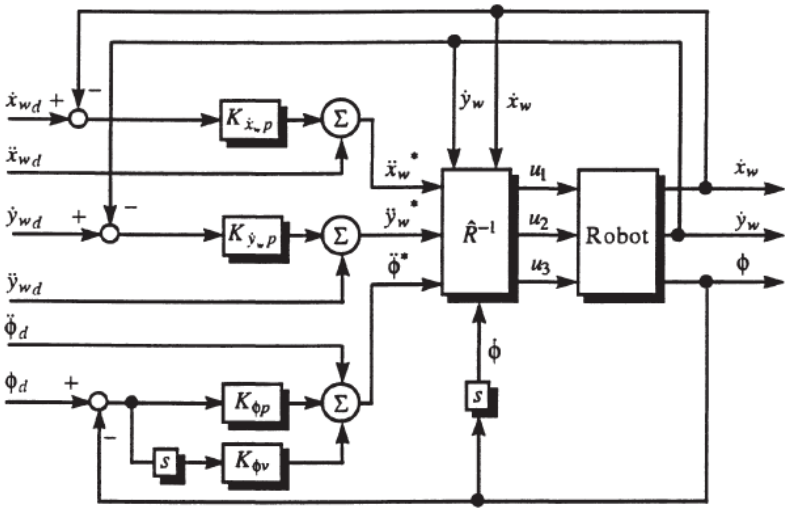


Figura 2.2: Sistema de controlo do tipo *acceleration-resolved* proposto por Watanabe [2]

Em 2001, Li *et al.* [12] alteraram a abordagem de resolução, propondo uma solução baseada em controlo não linear, mais especificamente, por sistemas de controlo baseados em lógica difusa. Nesta solução, os autores sugerem duas estruturas de controlo para um veículo de tração Ackerman, uma para o seguimento de trajetórias e outra para o planeamento das mesmas (mecanismo de decisão de trajetórias). O controlador de seguimento de trajetórias recebia como entradas o erro da distância e da orientação do robô e tinha

como saída a velocidade das rodas e a orientação das mesmas. Com estas estratégias, conseguiram obter um resultado bastante satisfatório, nomeadamente no âmbito em que esta solução é capaz de operar em *online* e com obstáculos.

Mais tarde, em 2007, na sua dissertação de doutoramento [3] e sintetizada em [13], Scolari sugere uma estratégia de controlo de baixa carga computacional, baseada em realimentação parcial do estado, capaz de corrigir a posição (x, y) e a orientação θ do robô. Para conseguir atingir o seu objetivo, o autor tira partido do conceito de que qualquer trajetória pode ser aproximada por um conjunto de segmentos de reta, que são, por sua vez, definidos por dois pontos de coordenadas (x, y), tal como demonstra a figura 2.3. Para cada ponto do segmento de reta é possível atribuir-se uma velocidade linear e um ângulo final de orientação, permitindo-se, assim, um ajuste refinado das velocidades.

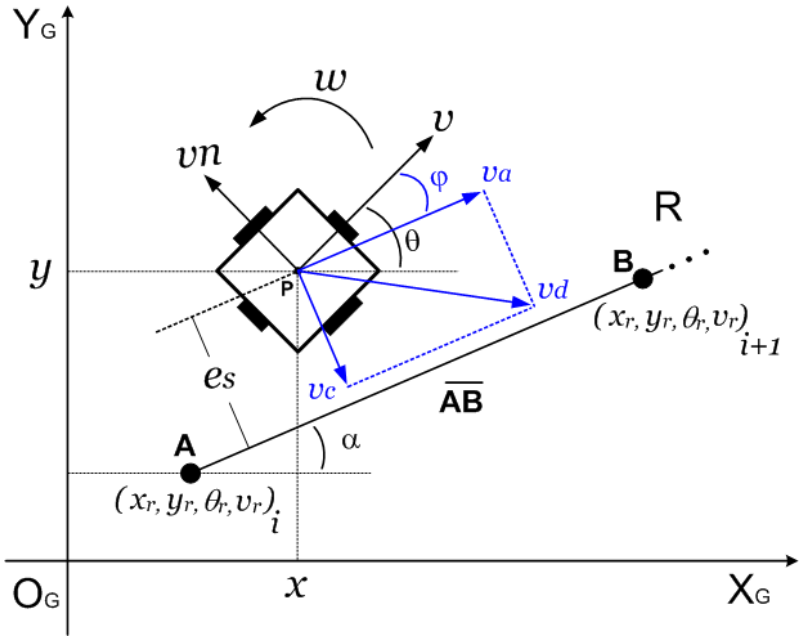


Figura 2.3: Representação esquemática do controlador e do robô móvel, segundo a estratégia de Scolari [3]

Há semelhança de todas as estratégias de controlo, neste, também, pode ocorrer a saturação dos motores, causando um desvio não intencional da trajetória de referência. Por isso, Scolari [3] sugere uma implementação adicional de um bloco de restrição dos sinais de controlo que tende a reduzir a velocidade dos outros motores, de acordo com a saturação que poderá acontecer num dos motores, segundo uma estrutura de planos de velocidades máximas, representada na figura 2.4. Por observação da figura, constata-se que todas as velocidades impostas deverão estar no interior do sólido, cujas faces são os planos anteriormente referidos. Nas situações em que ocorre a saturação, é traçada uma reta da origem até ao ponto de velocidade pretendida e a velocidade será mapeada no ponto de interseção com um dos planos.

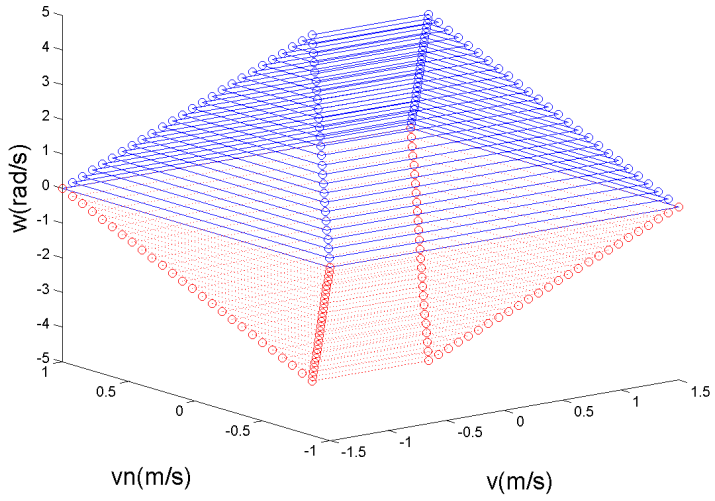


Figura 2.4: Planos das velocidades máximas do robô nos eixos v, v_n, ω para restrição dos sinais de controlo [3]

Em 2009, Nascimento [14] propõe, com sucesso, um sistema de controlo multi-variável em cascata linearizante para robôs móveis omnidirecionais. Para tal, apresenta uma abordagem com modelação em espaço de estados para uma modelação dinâmica e cinemática e implementa uma lei de controlo multivariável para o controlo de trajetórias, conservando o acoplamento das velocidades das rodas. O sistema de controlo é considerado linearizante porque controla um sistema não-linear, considerando-o linearizado em malha fechada, através de um controlador linear multi-variável.

Em 2011, Shojaei *et al.* [4], verificando que uma solução de controlo do seguimento de trajetórias baseada em controladores lineares, nomeadamente os controladores PID, não era suficientemente robusta, propõe uma lei de controlo não linear adaptativa (figura 2.5) que é projetada tendo por base a técnica de linearização da relação entrada-saída, para conseguir um cancelamento assimptótico das incertezas dadas pelos parâmetros do sistema. A lei de adaptação é deduzida por projeto de SPR-Lyapunov.

O sistema de controlo adaptativo demonstrou ter um desempenho satisfatório, quando os parâmetros de controlo fornecidos estão bem afinados. Assim, os autores concluíram que o controlador proposto apresenta um controlo robusto num ambiente de simulação e um controlo eficiente nos testes experimentais com um robô real de tração diferencial (figura 2.6).

No mesmo ano, Suster e Jadlovská [15] propõe, também, uma abordagem não linear ao problema, mas agora no âmbito da inteligência artificial (AI). Assim, eles sugerem duas abordagens de controlo, uma recorrendo a algoritmos genéticos de computação evolutiva e outra recorrendo à teoria das redes neuronais.

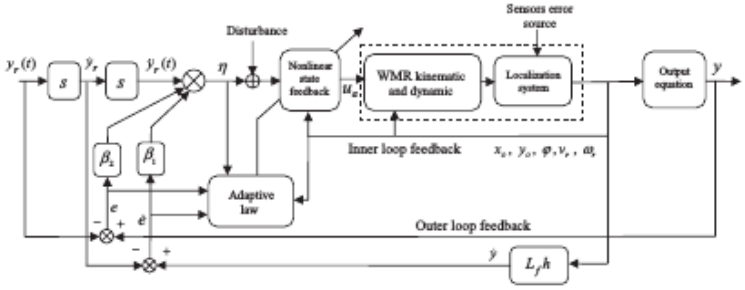


Figura 2.5: Sistema de controlo realimentado linearizado adaptativo, proposto por Shojaei *et al.*, para o seguimento de trajetórias de um robô móvel com rodas [4]

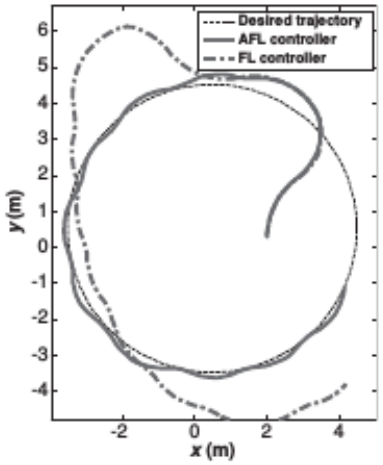


Figura 2.6: Resultados experimentais, para um robô móvel de tração diferencial, do sistema de controlo proposto por Shojaei *et al.*. A trajetória desejada é representada pela linha tracejada; a trajetória executada pelo robô com um controlador realimentado linearizado (FL) é representada pela linha tracejada e pontilhada e trajetória do robô com um controlador realimentado linearizado adaptativo (controlador proposto) é representada por uma linha sólida.

Os algoritmos genéticos são utilizados para a otimização dos parâmetros do controlador PID e do parâmetro K (parâmetro da velocidade de rotação do seguimento da trajetória de referência dentro da estrutura de controlo).

As redes neuronais serviram para criar um modelo neuronal direto e inverso. O modelo neuronal direto foi implementado dentro da estrutura de controlo IMC (*Internal Model Control*), junto com o modelo inverso, que é usado como um controlador neuronal não-paramétrico.

Apesar de tudo, o modelo sugerido não chegou a ser implementado numa situação real de controlo, apresentando bons resultados, apenas no âmbito de um ambiente de simulação.

Em 2015, Mobayen [16] sugere uma abordagem baseada no modo de deslizamento,

recorrendo ao método de modo de deslizamento de terminal rápido. A escolha do método foi baseada no facto de que este é capaz de evitar eventuais singularidades que possam ocorrer durante a fase de controlo. O sistema de controlo desenvolvido foi testado num robô móvel não omnidirecional e garantiu que o erro de seguimento caiu para zero num tempo finito com uma taxa de caimento exponencial.

Em abril de 2017, Mou Chen [5] publica um artigo onde propõe uma estratégia de controlo robusto para seguimento de trajetórias para robôs móveis com deslizamentos e perturbações no sinal de entrada. Para desenvolver o sistema pretendido, o autor teve de projetar um conjunto de leis de controlo e de condições que garantiam um bom desempenho do sistema de controlo, compensando eventuais perturbações. Este começou então por projetar uma lei de controlo de velocidade virtual baseada em observação de perturbações. De seguida, desenvolveu um esquema de controlo de seguimento, tendo em conta o requisito de desempenho para o seguimento da trajetória prescrita e utilizando o observador de perturbações. Este último é responsável por lidar com os deslizamentos e com as perturbações de entrada.

Na figura 2.7 verifica-se que esta estratégia de resolução apresenta um bom resultado no âmbito do seguimento de trajetória na existência de derrapagens e de perturbações do sinal, no entanto, há um erro de assimptótico que necessita de ser devidamente condicionado.

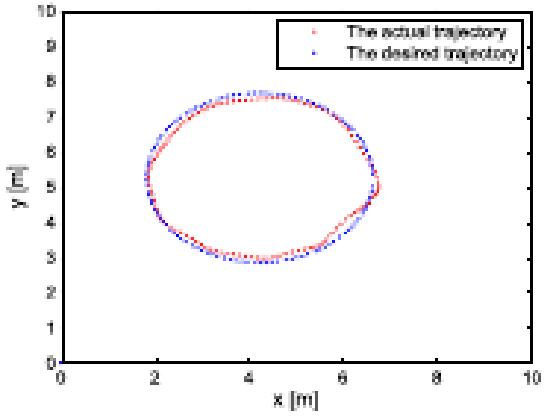


Figura 2.7: Resultado experimental para o seguimento de trajetórias, utilizando uma estratégia de controlo robusta para robôs com deslizamentos e perturbações do sinal de entrada [5]

Em 2017, Alakshendra e Chiddarwar [6] apresentam uma solução de seguimento de trajetórias que combina as abordagens de controlo adaptativo e modo de deslizamento para o controlo de trajetórias de um robô móvel omnidirecional *mecanum*, que designaram como um sistema de controlo de modo de deslizamento de segunda ordem robusto adaptativo (ASRRMC), representado na figura 2.8. Para projetar esta estratégia de controlo, os autores começaram por modelar a dinâmica e a cinemática do robô para deduzir as equações do movimento, por uma abordagem de Newton-Euler, na presença de fricções,

perturbações de forças externas e incertezas. De seguida, deduziram uma lei de controlo de segunda ordem para o modo de deslizamento, de modo a definir uma superfície de modo de deslizamento que garantisse uma solução robusta. Por fim, deduziram leis adaptativas para concretizarem um auto ajuste dos ganhos de comutação em resposta às incertezas do sistema. A estabilidade assimptótica da lei de controlo proposta foi provada pela teoria da estabilidade de Lyapunov.

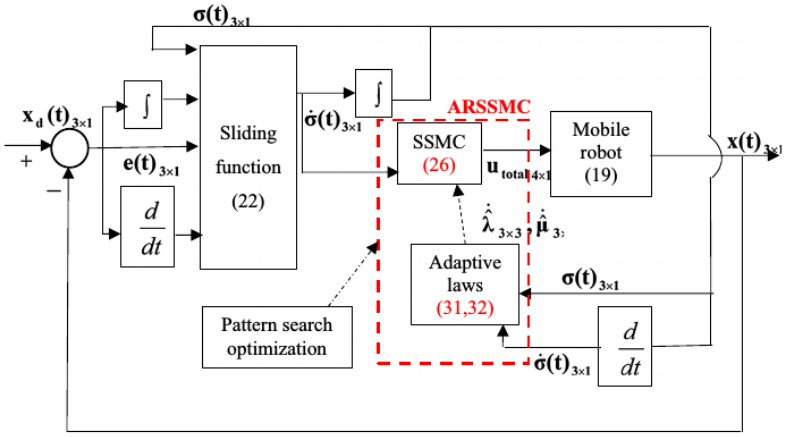


Figura 2.8: Diagrama do sistema de controlo proposto por Alakshendra e Chiddarwar [6]

Adicionalmente, é ainda proposta uma nova implementação de baixo custo para uma lei de controlo num robô real. Conclui-se que esta nova abordagem é passível de ser utilizada em situações de tempo real, sem comprometer a dinâmica do robô, embora tome mais tempo de processamento que o tradicional controlador PID ou o mesmo controlador mas utilizando uma lei de controlo de primeira ordem.

Os resultados do controlador foram comparados com um controlador PID e com um controlador de modo de deslizamento robusto adaptativo (ASRMC) e concluí-se que o novo sistema de controlo é superior, quer no âmbito de simulação, quer no âmbito de ambiente real, ao nível do erro quadrático integral (ISE), erro absoluto integral (IAE), erro absoluto integral de tempo pesado (ITAE), energia de controlo e variância total (TV). Os resultados experimentais em tempo real podem ser observados na figura 2.9.

Embora tivessem sido apresentadas diversas estratégias de controlo, todas estas possuem uma característica em comum: todas as ações de controlo são baseadas no erro perante a trajetória de referência. Isto significa que qualquer ação de controlo só será tomada depois de ter ocorrido um erro.

No sentido de contornar este problema, tentando minimizar o erro da trajetória em relação à de referência, foram desenvolvidas diversas estratégias de controlo preditivo que serão apresentadas na secção seguinte.

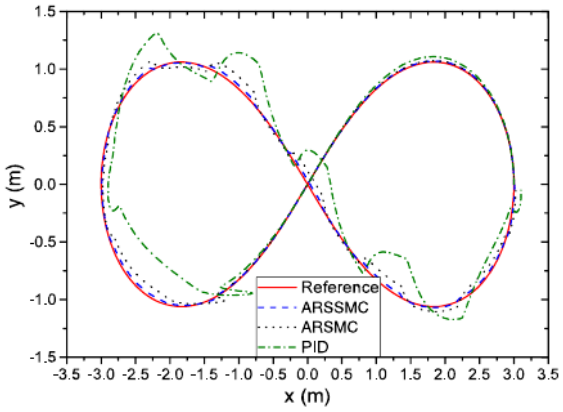


Figura 2.9: Resultado experimental do sistema de controlo proposto por Alakshendra e Chiddarwar [6]. O controlador proposto (ASRRMC) foi comparado com um controlador de modo de deslizamento robusto adaptativo (ASRMC) e com um controlador PID [6]

2.1.2 Controlo preditivo

O controlo preditivo, na comunidade cientifica é frequentemente apresentado como *Model Predictive Control* (MPC) [7, 17], tem encontrado uma vasta aceitação nas aplicações industriais, sendo atualmente também muito explorado nos meios académicos para novas aplicações. A razão por de trás desta grande popularidade, prende-se essencialmente com o facto de estes controladores produzirem sistemas de controlo de elevado desempenho capazes de operar sem intervenção durante longos períodos de tempo. Além disso, as técnica MPC fornecem uma metodologia capaz de lidar com as restrições e perturbações do sistema de uma forma sistemática e preditiva. O MPC, na sua forma mais genérica, não é restritivo em termos de modelo, função custo e restrições funcionais [7].

Os primeiros conceitos que levam ao aparecimento do MPC, surgem durante os anos 60. Primeiramente com Zadeh e Whalen [18] em 1962 que apresentam o controlo ótimo de tempo mínimo e a programação linear. Mais tarde, em 1963, Propoi [19] sugere uma abordagem de horizonte recuante (figura 2.10) que é o núcleo de resolução de todos os algoritmos MPC.

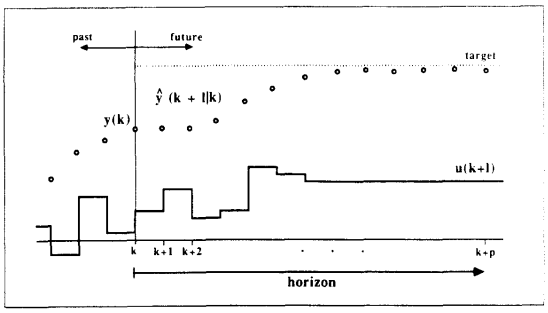


Figura 2.10: Abordagem horizonte recuante do MPC [7]

No entanto, apenas nos finais dos anos 70 é que surgiram os primeiros *papers* que se revelaram com particular interesse para a indústria, através da aplicação com sucesso da metodologia do *Model Predictive Heuristic Control* [20] e da formulação da *Dynamic Matrix Control* (DMC) [21, 22].

O nome *Model Predictive Control* (controlo preditivo), surge da forma como a lei de controlo é calculada (através do conceito de horizonte recuante – figura 2.10). No momento atual k, o comportamento do processo sobre o horizonte p é considerado. Portanto, o controlador recorre de um modelo do sistema que deve ser suficientemente preciso e robusto, para deduzir futuras alterações na trajetória do robô, minimizando o erro futuro. Camacho e Bordons [17] comparam o controlo preditivo à forma como conduzimos um carro. Enquanto que o controlo reativo se assemelha à condução de um veículo olhando apenas para os espelhos retrovisores, o que faz com que a correção da trajetória seja feita posteriormente à curva, o controlo preditivo, comporta-se como o condutor que olha para a frente e ajusta a sua trajetória de acordo com a trajetória de referência, isto é, começa a fazer a curva antes de lá chegar, procurando minimizar o erro e evitar acidentes.

Há diversas abordagens para os sistemas de controlo MPC, no entanto todas elas possuem um conjunto de componentes base, diferindo entre si pela forma como as opções são tomadas, sabendo que todas têm por base o horizonte recuante (tal como referido anteriormente):

- Modelo do processo
- Modelo ou tipo de perturbações
- Horizonte de predição
- Horizonte de controlo
- Critério a otimizar (função custo)

Garcia *et al.* [7] descrevem todas as abordagens existentes até 1989:

***Dynamic Matrix Control* (DMC)** As variáveis manipuladas são selecionadas para minimizar uma função quadrática. A predição da saída envolve três termos: o estado atual e parte do estado futuro (m) no horizonte de predição (p) das variáveis manipuladas, com $m < p$; apenas o estado passado das variáveis manipuladas; previsão das perturbações.

***Model Algorithmic Control* (MAC)** O MAC é similar ao DMC, diferindo em três aspetos: em vez de se utilizar um modelo de resposta em degrau envolvendo Δu, utiliza-se antes uma modelo em resposta impulsional, envolvendo u; passa a ser considerado todo o horizonte de predição (m = p); a perturbação estimada é filtrada.

Unconstrained MPC Corresponde a encontrar um controlador linear e invariante no tempo para um problema sem restrições que pode ser facilmente resolvido pela forma *standard* de um problema de mínimos quadrados.

Constrained MPC Esta abordagem tem em conta as restrições do sistema. Assim, nesta estratégia, o engenheiro ou o operador pode introduzir de uma forma direta as restrições do sistema no algoritmo e este irá automaticamente procurar a melhor solução, satisfazendo todas estas.

Non-linear MPC Existem diversas abordagens dentro dos sistemas de controlo preditivos não lineares como o controlo ótimo não linear, controlo ótimo linearizado e a inversão. Sempre que possível, procura-se recorrer a sistemas de controlo lineares, quer considerando um sistema de controlo linear, quer tentando linearizar um sistema em torno de um ponto de funcionamento. No entanto, isto nem sempre é possível e por vezes é necessário recorrer a estes sistemas de controlo.

Ramirez *et al.* [8] apresentam, em 1999, uma estratégia de controlo do tipo *Model Based Predictive control* (MBPC). Nesta estratégia, os autores penalizam, na função objetivo, o erro de seguimento da trajetória. Além disso, é ainda penalizada a possibilidade de colisão com um obstáculo, sendo considerado, para tal a distância ao obstáculo, que é obtida pelos sensores existentes no robô, na função objetivo (figura 2.11). Esta função é, por sua vez, otimizada em tempo real recorrendo aos algoritmos genéticos.

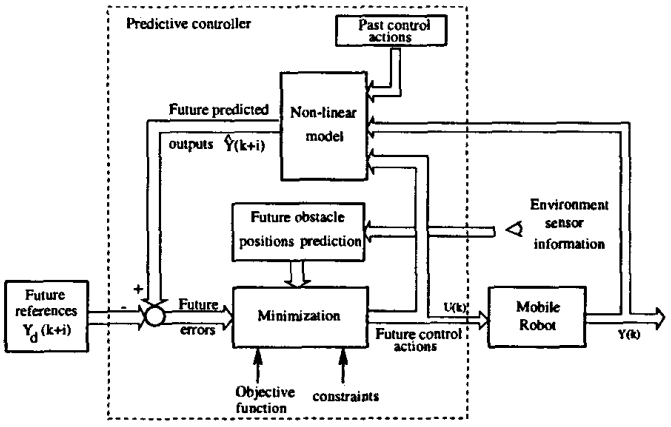


Figura 2.11: Esquema de controlo utilizado por Ramirez *et al.* [8], recorrendo a algoritmos genéticos [8]

Em 2000, Gu e Hu [23], utilizam uma estratégia de controlo destinada a robôs móveis que têm complexidade, não linearidades e incertezas. Neste método, o modelo não linear da dinâmica do robô é conseguido através de uma rede neuronal multi-camada *back propagation*, que é construída por decomposição ortogonal *wavelet*, e as variáveis de controlo

são calculadas por uma estratégia de otimização online de um índice de desempenho, recorrendo ao algoritmo *steepest gradient descendent*. Assim, Gu e Hu conseguiram obter uma estratégia de controlo do tipo *wavelet neural network based predictive control*.

Em 2005, Jiang *et al.* [24] introduzem um esquema de controlo combinado entre MPC e lógica difusa. O MPC é utilizado para prever a posição e a orientação do robô, por forma a compensar os atrasos provocados pela resposta lenta dos sensores. Por outro lado, a lógica difusa é desenhada para lidar com as características não lineares do sistema.

Klancar e Skrjane [25] retornam, em 2006, à formulação clássica do MPC, utilizando um sistema de controlo do tipo *MPC constrained* linear e não linear. Para conseguirem um bom controlador, recorrem a um modelo dinâmico do robô linearizado em torno de um ponto de funcionamento e uma função custo que penaliza tanto o erro de seguimento da trajetória, como o esforço no controlo do robô. É ainda discutida de forma breve uma possível abordagem ao problema utilizando uma estratégia de controlo baseada em *MPC unconstrained* não linear e/ou um sistema de tempo-variante.

Mais tarde, Vougioukas [26] utiliza um modelo de seguimento de trajetórias preditivo não linear (NMPT), onde recorre do modelo cinemático do robô e calcula em *online* uma sequência de controlo ótimo *M-Step-Ahead*. Na presença de obstáculos, o robô tem a capacidade de se desviar dos mesmos.

Em 2009, Kanjanawanishkul e Zell [27] abordam o problema de seguimento de trajetórias para robôs omnidirecionais. Nesta abordagem, utilizam um modelo linearizado do sistema e modelo dinâmico do erro é derivado do estado dos robôs e do estado da trajetória. A estratégia de controlo MPC é abordada juntamente com uma metodologia de seguimento de um veículo virtual. Assim, os autores conseguem transformar este problema de otimização num problema de programação quadrática.

Na sua tese de doutoramento, em 2007, Scolari [3] aborda o controlo de trajetórias para robôs móveis omnidireccionais através de uma aborgem MPC, mais especificamente, NMPC (figura 2.12) baseada na minimização da função custo por métodos numéricos de otimização não linear.

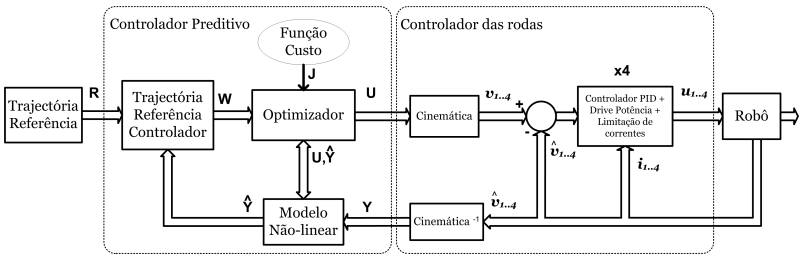


Figura 2.12: Esquema de controlo utilizado por Scolari [3]

A função custo penaliza, essencialmente, erros de seguimento de trajetórias. Esta é minimizada através de técnicas numéricas de otimização, nomeadamente métodos de gradiente descendente, tendo obtido os resultados mais favoráveis pelo método de otimização

dos gradientes conjugados.

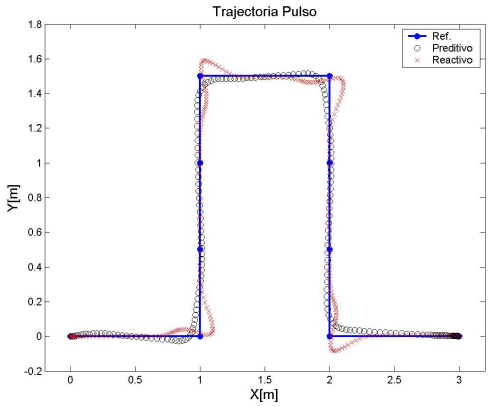


Figura 2.13: Comparação entre o controlador preditivo e o controlador reativo [3]

Em 2010, na sua tese de mestrado, Rodrigo [9] retoma os trabalhos realizados por Scolari [3] e reformula o controlo NMPC tendo por base o exposto por Findeisen e Allgöwer [28] (figura 2.14).

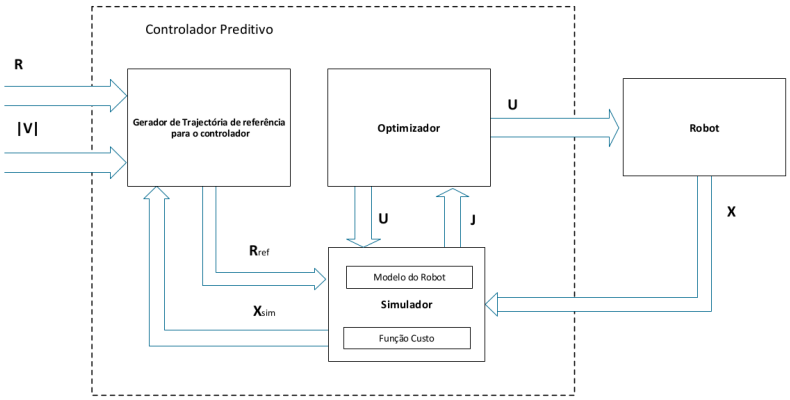


Figura 2.14: Esquema de controlo NMPC utilizado por Rodrigo [9]

No que se refere à função custo, Rodrigo [9], além de penalizar as diferenças entre a posição e a orientação do robô em relação à referência, tal como Scolari [3], penaliza ainda as variações na entrada de controlo, por forma a minimizar o esforço no controlo do robô. Analisando os trabalhos feito por Scolari, Rodrigo [9] conclui-se que a função objetivo pode ser facilmente, rapidamente e corretamente otimizada se recorrer ao algoritmo *steepest descendent*, uma variante dos algoritmos de gradiente descendente.

Em 2014, Barreto S. *et al.* [10] propõe um esquema de controlo MPC com compensação de fricções para o seguimento de trajetórias de robôs móveis omnidirecionais com três rodas. Para atingirem este objetivo, utilizaram uma estrutura de controlo em cascata com um bloco cinemático invertido que gera velocidades de referência para o controlador

preditivo. Parte do esforço de controlo é utilizado para compensar o efeito das fricções estáticas, o que confere uma maior robustez para a utilização de algoritmos MPC lineares com restrições.

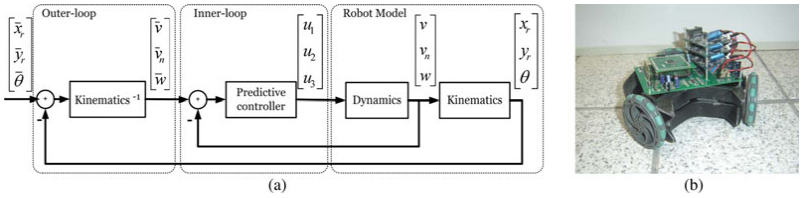


Figura 2.15: (a) Esquema de controlo. (b) Robô móvel omnidirecional utilizado [10]

Ostafew et al. [29] abordaram o problema do controlo de trajetórias de robôs em ambientes externos acidentados com robôs móveis baseados em visão. Assim, propuseram uma estrutura de controlo NMPC baseada em aprendizagem (LB-NMPC). Nesta estrutura, as perturbações são modeladas como um processo gaussiano com base nas perturbações observadas como função de variáveis relevantes como o estado do sistema e os sinais de entrada. Uma vez que este esquema de controlo é desenhado para robôs que estão sempre a repetir a mesma trajetória, verifica-se que uma modelação de perturbações baseada em aprendizagem, permite melhorar o modelo das perturbações, melhorando a robustez do sistema de controlo.

Em 2017, Xiao *et al.* [30] apresentam um modelo robusto MPC com uma otimização baseada em redes neuronais que é projetado por forma a estabilizar as restrições físicas do robô. Aplicando uma transformação de escalamento do estado, o controlador intrínseco no robô pode ser reajustado, incorporando dentro da entrada de controlo um termo adicional de caimento exponencial. De seguida é projetado um controlador MPC para o robô, tendo em conta as perturbações externas. A otimização da função custo do MPC é formulada como um problema de minimização não linear convexo e recorre-se a uma rede neuronal *primal-dual* para otimizar o problema num horizonte recuante finito.

2.2 Seguimento de trajetórias

Tal como referido em [3, 9], uma trajetória pode ser definida como um conjunto de coordenadas no espaço. Assim, tendo em conta esta característica, é importante selecionar um abordagem que permita definir a forma como o robô se desloca entre pontos, entre muitas abordagens, Moreira [11] apresenta algumas, tais como:

- *Go to xy theta* (dirigir-se para um ponto de coordenadas (x, y) com um ângulo θ)
- *Follow line* (seguir uma linha)
- *Follow parametric segment* (seguir um segmento paramétrico)

A abordagem *Go to xy theta* ([11, 31] que especificam uma solução para um robô de tração diferencial), corresponde a uma abordagem de deslocar o robô para um ponto, sem que haja grande controlo sobre o que acontece nos pontos intermédios, sendo apenas garantido que o robô chega ao ponto desejado com uma determinada tolerância e com o ângulo desejado. Esta abordagem pode ser implementada recorrendo a máquinas de estado (figura 2.16) ou a redes de petri. As variáveis de controlo são o erro da distância ao ponto pretendido e o erro absoluto entre o ângulo atual do robô e o ângulo de se o robô estivesse direcionado para a posição final (figura 2.17).

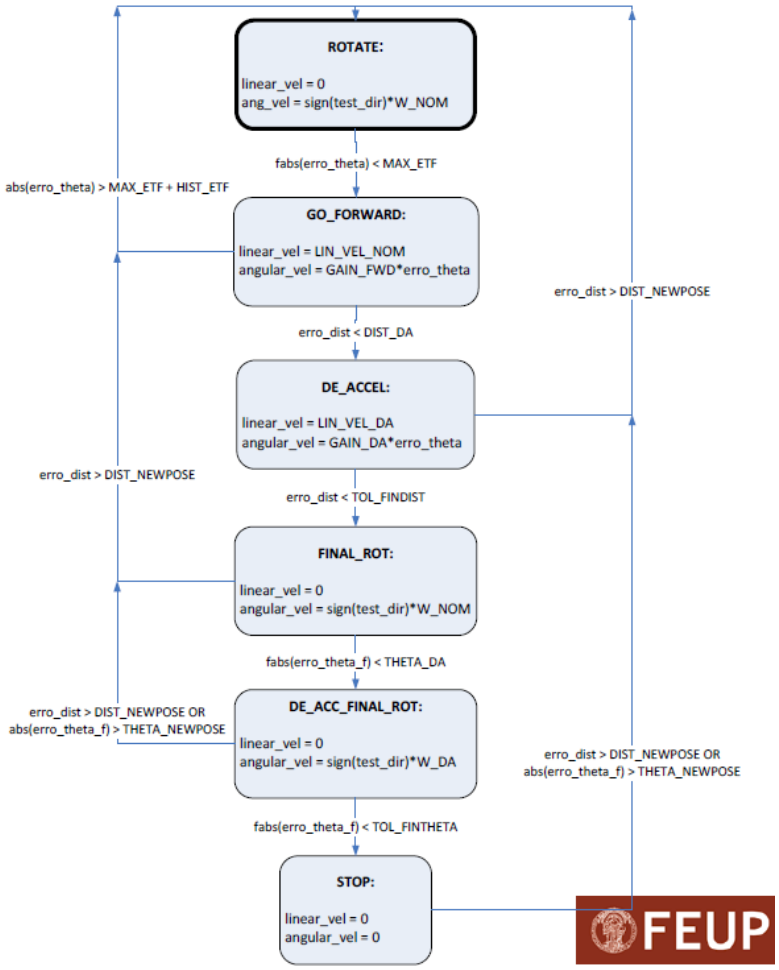


Figura 2.16: Máquina de estados de uma trajetória do tipo *GoToXYTheta* aplicada a um robô de tração diferencial [11]

Numa opção sobre uma trajetória do tipo *follow line* [32], é garantido que o robô se move sobre uma linha que é definida pelos dois pontos inicial e final. Tipicamente, este tipo de controlo pode ser definido como uma máquina de estados com três estados. O primeiro estado tende a resolver o problema de o robô não estar perto da reta definida, como tal, nesta fase, o robô move-se perpendicularmente a esta até estar sobre esta. Quando este

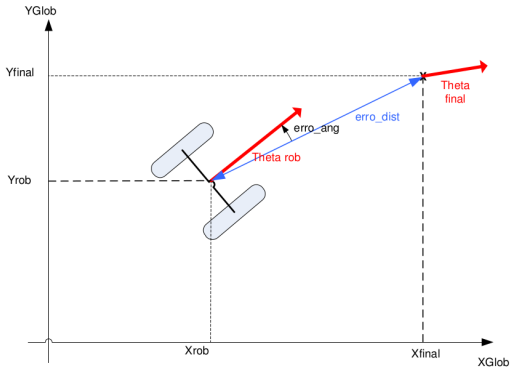


Figura 2.17: Variáveis de controlo de uma trajetória do tipo *GoToXYTheta* aplicada a um robô de tração diferencial [11]

atinge a reta, transita para um segundo estado que faz com que o ele navegue sobre a linha até ao ponto final. Uma vez neste ponto, transita-se para um terceiro estado de parado.

Por fim, a terceira abordagem do tipo *follow parametric segment* é semelhante à abordagem anterior, no entanto, esta permite que o robô siga linhas curvas e a trajetória é definida de uma forma paramétrica (figura 2.18), tal como definido na equação 2.1.

$$Trajectory(t) = \begin{bmatrix} f_x(t) \\ f_y(t) \end{bmatrix} \qquad \text{em que } t \in [0,1] \tag{2.1}$$

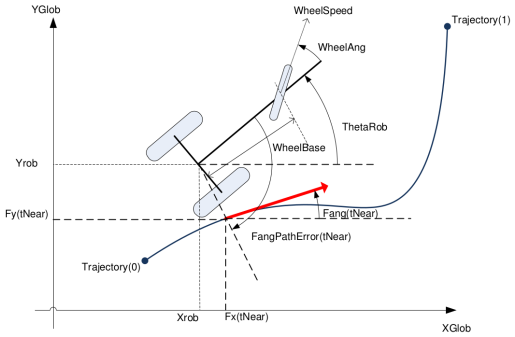


Figura 2.18: *Follow parametric segment* [11]

Capítulo 3

Caracterização do Problema

3.1 Definição do problema

Desde o início da robótica móvel que o controlo da trajetória efetuada pelo robô exigiu uma atenção particular.

Primeiramente, começou-se por abordar o problema, implementando trajetórias fixas e procurando garantir apenas que o robô não se desviava da trajetória inicialmente definida. Para tal recorria-se a sistemas de controlo em malha fechada, do tipo reativo, com um controlador do tipo PID. Com o avanço do tempo e da investigação, começou-se a optar por outras estruturas de controlo mais complexas, que permitissem lidar com sistemas a controlar mais complexos, portanto, surgiu assim o aparecimento dos primeiros sistemas de controlo não lineares, baseados em algoritmos de modo de deslizamento, redes neuronais, algoritmos genéticos, entre outros. Por fim, uma vez que estes métodos de controlo dependem do que aconteceu e estão sujeitos a erros de trajetória elevados, procurou-se implementar sistemas de controlo preditivos, que permitissem prever a trajetória que vai ser efetuada, minimizando o erro de seguimento.

Tendo estes aspetos em consideração, pode-se concluir que esta dissertação tem como principal objetivo o controlo preciso de trajetórias para interseção de um alvo, de modo a permitir a circulação do robô a elevadas velocidades. O ambiente de simulação e teste dos algoritmos implementados será um robô da equipa de futebol robótico 5DPO FEUP/INESC TEC.

3.2 Solução proposta

Tendo em vista o problema exposto na secção 3.1 e o estudo do estado da arte feito em todo o capítulo 2, pode-se abordar o problema em três fases:

- Modelação do robô móvel para um simulador
- Implementação e estudo de estruturas de controlo reativo

- Implementação e estudo de estruturas de controlo preditivo

No que concerne ao sistema de controlo de trajetórias, pretende-se abordar o problema com um algoritmo de seguimento de trajetórias do tipo *go to position* e outro do tipo *follow line*. O controlo será assegurado inicialmente por um controlador reativo do tipo PID, sendo posteriormente substituído por uma abordagem de controlo MPC, tal como apresentado por Scolari e Rodrigo [3, 9].

Todo o algoritmo será implementado numa linguagem de programação de alto nível e testado num simulador. Posteriormente e em caso de sucesso, o algoritmo será testado no robô real.

Capítulo 4

Equipa de futebol robótico 5DPO

4.1 Contexto

A equipa 5DPO é a equipa de futebol robótico da FEUP/INESC TEC e é constituída por professores da FEUP, investigadores do INESC TEC, estudantes do Mestrado Integrado em Engenharia Eletrotécnica e de Computadores, do Mestrado Integrado em Engenharia Informática e Computação e do Programa Doutoral em Engenharia Eletrotécnica e de Computadores.

Nos últimos anos esta equipa tem participado na competição MSL (liga de tamanho médio) no Festival Nacional de Robótica que se gere pelas regras da RoboCup que são uma adaptação das regras oficiais da FIFA. Além disso, esta mesma equipa já participou em algumas outras competições como a SSL (liga de tamanho pequeno) de futebol robótico.

As competições de futebol robótico surgem como uma proposta da RoboCup Federation [33] para impulsionar o estado da arte através de um desafio e exigente e a longo prazo. por conseguinte, é proposto como desafio último do RoboCup o seguinte:

> No meio do século XXI, um equipa completamente autónoma de robôs de futebol robótico humanoide devem ganhar um jogo de futebol, cumprindo as regras da FIFA, contra o vencedor do mais recente Campeonato do Mundo.
>
> — *RoboCup Federation*

Portanto, este torna-se um desafio ambicioso que sente a necessidade de lidar com as complexidades do tempo real, promovendo uma forte investigação nas áreas da robótica e da inteligência artificial, como por exemplo na fusão sensorial em tempo real, comportamento reativo, estratégias de aquisição de dados, aprendizagem máquina, planeamento em tempo real, controlo em ambientes dinâmicos, entre outros.

4.2 *Hardware*

A equipa MSL 5DPO conta com seis robôs de futebol que respeitam as características das estabelecidas nas regras do RoboCup para a competição MSL. Estes possuem uma tração omnidirecional de três rodas (como as da figura 4.2a) que lhes confere a possibilidade de efetuar movimentos simples e compostos em qualquer direção.

Para que consigam jogar em equipa e fazer remates este têm um sistema de *kicker* (representado na figura 4.2c) constituído por um solenoide que é alimentado, durante o remate ou o passe, por 100 V provenientes de uma bateria de condensadores. Adicionalmente, há uma segunda solenoide que faz deslizar um batente secundário, conforme se pretende um passe alto ou baixo. Além disso, para poderem manobrar a bola, os robôs estão ainda dotados de um mecanismo de *rollers* (também representados na figura 4.2c) que puxam a bola e de um sensor ótico que deteta quando o robô está na posse da bola ou não.

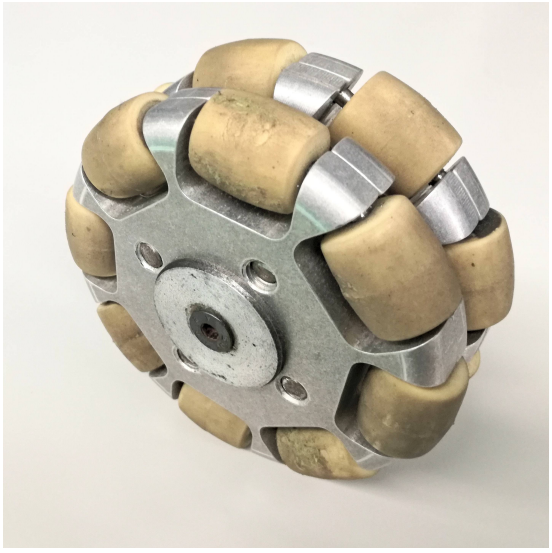
Para se localizar no campo, os robôs recorrem a diferentes estratégias, sendo a mais importante o sistema de visão catadióptrico. Neste, há uma câmara industrial que aponta para um espelho convexo (figura 4.2b), dotando, desta forma, o robô de um sistema de visão em 360º. Adicionalmente, recorre à bússola de uma IMU para distinguir o lado do campo onde está, dada a sua simetria.

A figura 4.1 representa todos os robôs da equipa, sendo visíveis algumas diferenças entre eles, que refletem o trabalho de investigação que se tem vindo a desenvolver, para se conseguir melhorar o sistema.

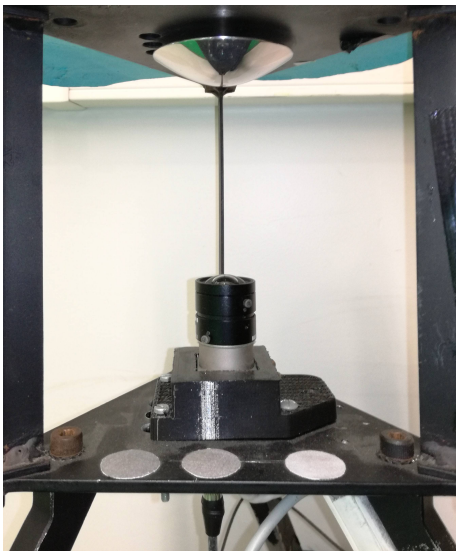
(a) Equipa de futebol robótico

(b) Robô utilizado nos testes

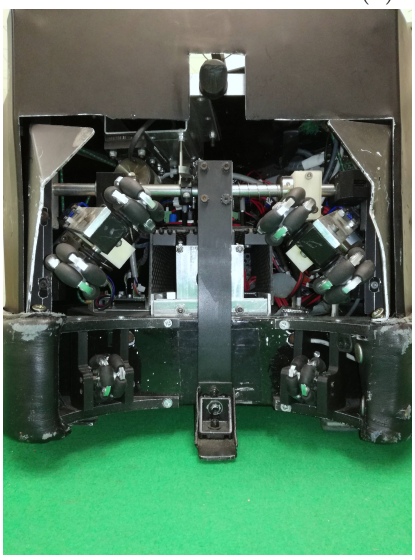
Figura 4.1: Robôs da equipa de futebol robótico



(a) Roda omnidirecional



(b) Visão catadióptrica



(c) Mecanismo de *kicker* e *rollers*

Figura 4.2: Componentes do robô

4.3 *Software*

Um jogo de futebol robótico é tipicamente caracterizado pela cooperação entre múltiplos agentes. Como tal, a equipa de futebol robótico adotou um sistema distribuído igual ao representado na figura 4.3 para os robôs comunicarem entre si e tomarem decisões.

O *Hardware Control* é uma aplicação de interface entre o *hardware* do robô e o *software* de decisão, *Decision*. Este comunica com o *hardware*, nomeadamente um Arduino (que processa a odometria do robô e envia as velocidades de referência para os *drivers* dos motores), o sistema de *kicker* e os *rollers* por porta série. As informações recebidas do *hardware* são processadas por si e posteriormente enviadas para o *Decision* por ZeroMQ.

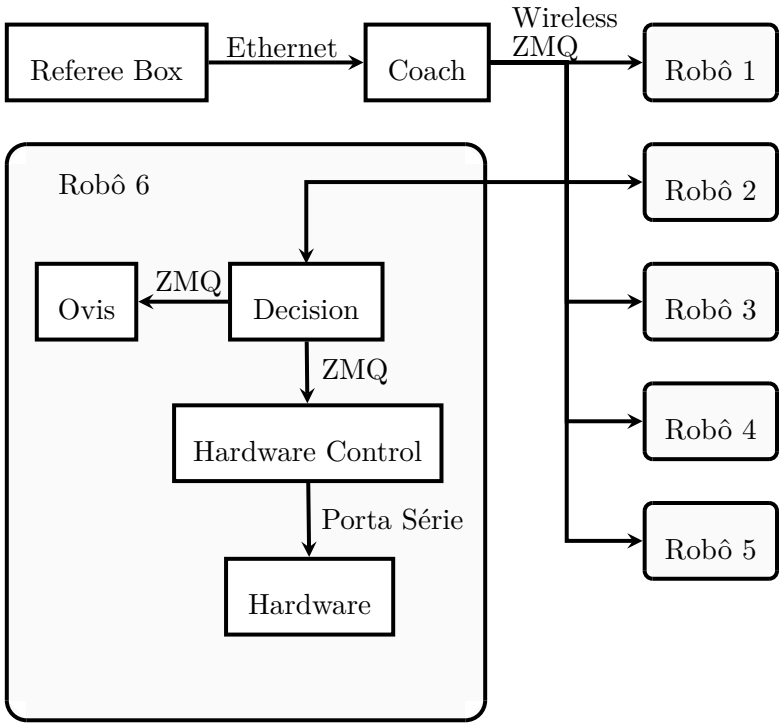


Figura 4.3: Arquitetura do sistema

Contrariamente ao que acontecia com o seu antecessor, *Flashbus*, esta aplicação já é mais completa, permitindo a visualização e teste do hardware que controla.

Por sua vez, o *Ovis* também é uma aplicação de interface que faz parte do sistema de visão catadióptrico. Este é responsável pelo processamento da imagem recebida a cada 40 ms (a câmara funciona com um frequência de 25 fps), por forma a detetar o campo, os obstáculos e a bola, por distinção da cor de cada objeto, ou seja, a bola é amarela ou laranja, as linhas do campo são brancas e o campo é verde (atualmente, ainda não é possível identificar outros obstáculos).

O *Coach* (ou treinador) é a aplicação de supervisão do estado do jogo e tem a responsabilidade de garantir a cooperação entre todos os agentes. Este recebe diferentes instruções da *referee box* (árbitro) e efetua a sua tradução em táticas e jogadas para a equipa, atribuindo diferentes papéis a cada agente, de acordo com esta decisão.

Por fim, o *Decision* é a aplicação central de todo os sistema do robô. Este tem de executar ações e tomar decisões de acordo com as ordens e informações que recebe do treinador e da perceção que tem do campo e do estado do jogo.

Capítulo 5

Modelação do robô móvel

Em qualquer estrutura de controlo é sempre útil e importante existir um modelo analítico do sistema a modelar, por própria exigência do controlador, para se compreender o modo de funcionamento do sistema, procurando uma solução mais facilmente ou para efeitos de simulação.

Assim, neste capítulo, pretende-se descrever um modelo do robô móvel omnidirecional da equipa de futebol robótico 5DPO, para efeitos de projeto dos controladores e de simulação no simulador SimTwo [34]. Tendo em vista a estrutura do robô, pode-se dividir o problema em duas partes: modelo cinemático e modelo dinâmico.

5.1 Modelo cinemático

No modelo cinemático, procura-se caracterizar o robô tendo em conta os seus aspetos construtivos. Neste caso, o facto deste possuir três rodas omnidirecionais, que o permitem mover-se em qualquer direção, separadas entre si em 120º e a uma distância d do centro, tal como demonstra a figura 5.1.

Assim, considerando que um robô tem sempre por base pelo menos dois referenciais para se localizar e mover (o seu próprio referencial e o referencial de mundo ou global), pode-se especificar as variáveis da figura 5.1. Desta figura entende-se:

- d [m] – Distância das rodas ao centro
- (x_r, y_r) [m] – Posição do robô em relação ao referencial global
- θ [rad] – Ângulo do robô em relação ao referencial global
- $\mathbf{v}, \mathbf{v_n}$ [m/s] – Velocidade do robô no seu próprio referencial, considerando a sua direção e sentido
- ω [rad/s] – Velocidade angular do robô sobre o seu próprio eixo
- $\mathbf{v_1}, \mathbf{v_2}, \mathbf{v_3}$ [m/s] – Velocidade linear das rodas

- $\mathbf{v_x}, \mathbf{v_y}$ [m/s] – Velocidade do robô no referencial global.

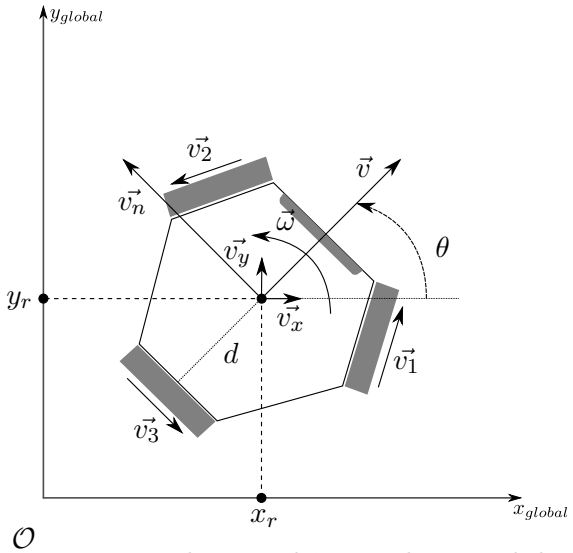


Figura 5.1: Esquema de um robô omnidirecional de três rodas

5.1.1 Sistema de coordenadas

Tal como já foi referido, é considerado que existem pelo menos dois referenciais distintos: um associado ao mundo e outro associado ao robô. O sistema de coordenadas associado ao mundo é fixo no espaço e denominado, na figura 5.1, como o sistema de coordenadas global. Por outro lado, o sistema de coordenadas do robô está centrado no seu centro geométrico e é constituído pelo versor com a mesma direção e sentido do robô e seu versor perpendicular.

Assim, dada a existência deste dois referenciais, é relevante existir um conjunto de matrizes de transformação que permitam converter os dados registados num sistema de coordenadas no outro.

A matriz de rotação R^r_{global}, na equação 5.1, permite transformar as velocidades do robô no sistema de coordenadas global, nas mesmas no sistema de coordenadas do robô.

$$\mathbf{v_r} = \begin{bmatrix} \cos(\theta(t)) & \sin(\theta(t)) & 0 \\ -\sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{v_{global}} \tag{5.1}$$

Calculando a matriz inversa de R^r_{global}, consegue-se transformar as velocidades no sistema de coordenadas do robô para o sistema de coordenadas global. Assim, tem-se que a matriz R^{global}_r é dada pela matriz na equação 5.2.

$$\mathbf{v_{global}} = \begin{bmatrix} \cos(\theta(t)) & -\sin(\theta(t)) & 0 \\ \sin(\theta(t)) & \cos(\theta(t)) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \mathbf{v_r} \tag{5.2}$$

5.1.2 Cinemática direta e inversa

Tipicamente, os robôs são controlados tendo em consideração a sua direção e sentido, isto é, as velocidades são expressas segundo os versores $\mathbf{v}$, $\mathbf{v_n}$ e ω. Assim, é importante estabelecer uma matriz de cinemática direta que transforme estas velocidades, nas correspondentes velocidades das rodas $\mathbf{v_1}$, $\mathbf{v_2}$ e $\mathbf{v_3}$, por forma a garantir um correto movimento do robô. Esta matriz pode ser obtida pelo princípio da sobreposição, tal como explica Vítor Pinto e António Moreira [35].

Considerando $v \neq 0$, $v_n = 0$ e $\omega = 0$ tem se a matriz de cinemática direta 5.3.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \sin(\alpha) \\ -\sin(\alpha) \\ 0 \end{bmatrix} \cdot v \tag{5.3}$$

Considerando agora $v = 0$, $v_n \neq 0$ e $\omega = 0$, tem se que a cinemática direta é dada por 5.4.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \cos(\alpha) \\ \cos(\alpha) \\ -1 \end{bmatrix} \cdot v_n \tag{5.4}$$

Por fim, considerando apenas a componente ω, isto é, $v = 0$, $v_n = 0$ e $\omega \neq 0$, a matriz de cinemática direta é dada pela equação 5.5.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} d \\ d \\ d \end{bmatrix} \cdot \omega \tag{5.5}$$

Aplicando agora o princípio da sobreposição a 5.3, 5.4 e 5.5 e sabendo que nos robôs omnidirecionais de três rodas $\alpha = \frac{\pi}{3}$, tem-se que a matriz de cinemática direta que modela completamente o sistema é dada pela equação 5.6.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \sin\left(\frac{\pi}{3}\right) & \cos\left(\frac{\pi}{3}\right) & d \\ -\sin\left(\frac{\pi}{3}\right) & \cos\left(\frac{\pi}{3}\right) & d \\ 0 & -1 & d \end{bmatrix} \cdot \begin{bmatrix} v \\ v_n \\ \omega \end{bmatrix} \tag{5.6}$$

A obtenção da matriz de cinemática inversa, que permite obter as velocidades v, v_n e ω a partir das velocidades das rodas, pode ser obtida por inversão da matriz de cinemática direta. Considerando os mesmos princípios que em 5.6, tem-se que a matriz de cinemática inversa é dada pela equação 5.7.

$$\begin{bmatrix} v \\ v_n \\ \omega \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{3} & -\frac{\sqrt{3}}{3} & 0 \\ \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \\ \frac{1}{3d} & \frac{1}{3d} & \frac{1}{3d} \end{bmatrix} \cdot \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} \tag{5.7}$$

5.1.3 Odometria

Embora existam diferentes métodos de localização de robôs em espaços, o recurso à odometria é um método recorrente, quer pelo facto de ser intrínseco ao robô, sendo suficiente a adição de *encoders* às rodas, quer porque permite aumentar a precisão da sua localização em distâncias curtas, quando comparado com outros sistemas. Assim, é importante traduzir corretamente o movimento das rodas no deslocamento do robô em x, y e θ.

No caso dos robôs omnidirecionais de três rodas, este problema é dividido em dois conjuntos de equações, por forma a solucionar o problema de divisão por zero quando $\omega = 0$. Assim, num primeiro conjunto de equações, considera-se que o robô não possui movimento angular e tem-se que a evolução da posição do robô depende apenas das componentes $\vec{v}$ e $\vec{v_n}$ (equações 5.8). De seguida, considera-se que o robô também está a rodar sobre o seu próprio eixo, isto é, $\omega \neq 0$, e obtém-se que o deslocamento do robô é obtido pelas equações 5.9.

$$\begin{aligned} x(i+1) =& x(i) + d(i)\cos(\theta(i)) - d_n \sin(\theta(i)) \\ y(i+1) =& y(i) + d(i)\sin(\theta(i)) + d_n \cos(\theta(i)) \qquad \omega = 0 \\ \theta(i+1) =& \theta(i) \end{aligned} \tag{5.8}$$

$$\begin{aligned} x(i+1) = x(i) &+ (d(i)\sin(\Delta\theta(i)) + d_n(\cos(\Delta\theta(i)) - 1)) \cdot \frac{\cos\left(\theta(i) + \frac{\Delta\theta(i)}{2}\right)}{\Delta\theta} \\ &- (d(i)(1 - \cos(\Delta\theta(i))) + d_n \sin(\Delta\theta(i))) \cdot \frac{\sin\left(\theta(i) + \frac{\Delta\theta(i)}{2}\right)}{\Delta\theta} \\ y(i+1) = y(i) &+ (d(i)\sin(\Delta\theta(i)) + d_n(\cos(\Delta\theta(i)) - 1)) \cdot \frac{\sin\left(\theta(i) + \frac{\Delta\theta(i)}{2}\right)}{\Delta\theta} \\ &+ (d(i)(1 - \cos(\Delta\theta(i))) + d_n \sin(\Delta\theta(i))) \cdot \frac{\cos\left(\theta(i) + \frac{\Delta\theta(i)}{2}\right)}{\Delta\theta} \\ \theta(i+1) &= \theta(i) + \Delta\theta(i) \end{aligned} \quad ,\omega \neq 0 \tag{5.9}$$

Nas equações 5.8 e 5.9, d e d_n correspondem ao deslocamento produzido segundo as componente $\vec{v}$ e $\vec{v_n}$, respetivamente, e $\Delta\theta$ a variação do ângulo do robô durante um determinado período de tempo (Δt). Assim, $d(i)$, $d_n(i)$ e $\Delta\theta(i)$ podem ser obtidos por:

$$\begin{aligned} d(i) =& v(i) \cdot \Delta t \\ d_n(i) =& v_n(i) \cdot \Delta t \\ \Delta\theta(i) =& \omega(i) \cdot \Delta t \end{aligned} \tag{5.10}$$

5.2 Modelo dinâmico

Complementarmente ao modelo cinemático, o modelo dinâmico pretende caracterizar o comportamento do robô, tendo em consideração as suas características descritíveis através de equações diferenciais. Este pode variar de acordo com o seu peso, tipo de materiais utilizados, a sua forma, entre outros.

Scolari [3] estudou e caracterizou de forma exaustiva o modelo dinâmico do robô. Aqui pretende-se ter uma abordagem simplificada para um modelo simplificado do mesmo.

5.2.1 Caracterização do modelo dinâmico do robô

A dinâmica de translação e de rotação do robô pode ser descrita pelas leis de Newton para o movimento linear e para a rotação [36] nas equações 5.11, 5.12 e 5.13.

$$M \cdot \frac{\mathrm{d}v}{\mathrm{d}t} = F_v(t) + F_{B_v}(t) + F_{C_v}(t) \tag{5.11}$$

$$M \cdot \frac{\mathrm{d}v_n}{\mathrm{d}t} = F_{v_n}(t) + F_{B_{v_n}}(t) + F_{C_{v_n}}(t) \tag{5.12}$$

$$J \cdot \frac{\mathrm{d}\omega}{\mathrm{d}t} = T(t) + T_{B_\omega} + T_{C_\omega} \tag{5.13}$$

- M [Kg] – Massa do robô
- F_v e F_{v_n} [N] – Força resultante segundo as componentes v e v_n, respetivamente
- F_{B_v} e $F_{B_{v_n}}$ [N] – Força de atrito viscoso segundo as componentes v e v_n, respetivamente
- F_{C_v} e $F_{C_{v_n}}$ [N] – Força de atrito de Coulomb segundo as componentes v e v_n, respetivamente
- J [Kg·m^2] – Momento de inércia do robô
- T [N·m] – Binário do robô
- T_{B_ω} [N·m] – Binário de atrito viscoso
- T_{C_ω} [N·m] – Binário de atrito de Coulomb

As forças e os binários resultantes segundo cada uma das direções pode ser obtida pela força desenvolvida por cada um dos motores f_i e pela geometria do próprio robô, equações 5.14, 5.15 e 5.16.

$$F_v(t) = (f_1(t) - f_2(t)) \cdot \sin\left(\frac{\pi}{3}\right) \tag{5.14}$$

$$F_{v_n}(t) = -f_3(t) + (f_1(t) + f_2(t)) \cdot \cos\left(\frac{\pi}{3}\right) \tag{5.15}$$

$$T(t) = (f_1(t) + f_2(t) + f_3(t)) \cdot d \tag{5.16}$$

A força de cada motor é obtida pela relação do seu binário com o contacto com a superfície, equação 5.17. Enquanto que o binário depende do conjunto motor e caixa redutora, equação 5.18.

$$f_i(t) = \frac{T_i(t)}{r} \tag{5.17}$$

$$T_i(t) = i_i \cdot l \cdot K_t \tag{5.18}$$

Destas equações, r é o raio das rodas, i_i é a corrente na armadura do motor i, l é o factor de redução da caixa redutora e K_t a constante de binário do motor.

As força de atrito viscoso são proporcionais à velocidade do robô e podem ser determinadas por 5.19, 5.20, 5.21, onde B_v, B_{v_n} e B_ω são constantes.

$$F_{B_v} = -B_v \cdot v(t) \tag{5.19}$$

$$F_{B_{v_n}} = -B_{v_n} \cdot v_n(t) \tag{5.20}$$

$$T_{B_\omega} = -B_\omega \cdot \omega(t) \tag{5.21}$$

Por outro lado, a força de atrito de Coulomb é de amplitude fixa e pode ser determinada por 5.22, 5.23 e 5.24, também com C_v, C_{v_n} e C_ω constantes.

$$F_{C_v} = -C_v \cdot \text{sign}(v(t)) \tag{5.22}$$

$$F_{C_{v_n}} = -C_{v_n} \cdot \text{sign}(v_n(t)) \tag{5.23}$$

$$F_{C_\omega} = -C_\omega \cdot \text{sign}(\omega(t)) \tag{5.24}$$

5.2.2 Caracterização do modelo dinâmico dos motores

O robô utilizado recorre a três motores DC com escovas para se movimentar. Estes podem ser simplificados pelo esquema elétrico na figura 5.2. Desta figura, retira-se, ainda, que o modelo dos motores pode ser aproximado pelo conjunto das equações 5.25, 5.26, 5.27.

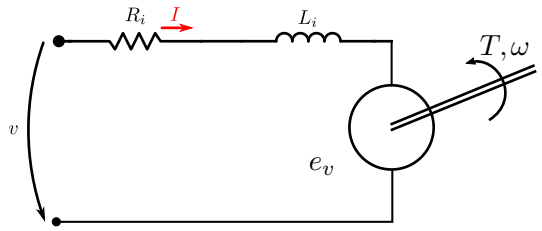


Figura 5.2: Esquema elétrico simplificado de um motor DC

$$u(t) = R_i \cdot I(t) + L_i \cdot \frac{\mathrm{d}I(t)}{\mathrm{d}t} + e_v(t) \tag{5.25}$$

$$e_v(t) = K_e \cdot \omega(t) \tag{5.26}$$

$$T(t) = K_t \cdot I(t) \tag{5.27}$$

Considerando o regime permanente, é possível eliminar a componente $L_i \frac{\mathrm{d}I}{\mathrm{d}t}$ de 5.25, resultando no modelo simplificado dos motores 5.28.

$$u(t) = R_i \cdot I(t) + K_e \cdot \omega_m \tag{5.28}$$

A dinâmica de aceleração dos motores pode ser obtida pela equação 5.29.

$$J_m \cdot \frac{\mathrm{d}\omega}{\mathrm{d}t} = K_t \cdot I(t) - B_v \cdot \omega - F_c \tag{5.29}$$

- u [V] – Tensão nos motores
- L [H] – Indutância do motor
- I [A] – Corrente no motor
- R [Ω] – Resistência no motor
- e_v [V] – Tensão eletromotriz do motor
- K_e [V/(rad/s)] – Constante da força eletromotriz
- T [N·m] – Binário do motor
- K_t [N·m/A] – Constante de binário
- ω_m [rad/s] – Velocidade angular do motor
- J_r [Kg·m^2] – Momento de inércia do rotor
- ω [rad/s] – Velocidade do motor
- B_v – Constante de atrito viscoso do motor

- F_c – Constante de atrito de Coulomb

5.3 Modelo completo do robô

Uma vez caracterizadas, através de equações diferenciais, as diferentes componentes do robô, resta descrever da mesma forma todo o sistema do robô.

Dada a existência de uma caixa redutora, a velocidade das rodas, ω_r, pode ser descrita pela equação 5.30, onde ω_m é a velocidade do motor, r é o raio das rodas e l o fator de redução da caixa redutora.

$$\omega_r = \frac{\omega_m \cdot l}{r} \tag{5.30}$$

Portanto, sabendo que, quando expressas nas unidades do sistema internacional, a constante da força eletromotriz do motor e a constante de binário são iguais, isto é, $K = K_e = K_t$, então, utilizando as equações 5.17, 5.18, 5.28 e 5.30, podemos descrever a força exercida por cada motor através de 5.31, onde v_i é a velocidade da roda i e u_i é a tensão no motor i.

$$f_i = \frac{l \cdot K}{R \cdot r} \cdot u_i - \frac{K^2 \cdot l^2}{R \cdot r} \cdot v_i \tag{5.31}$$

A matriz de cinemática inversa apresentada em 5.6, pode ainda ser reescrita sob a forma apresentada em 5.32.

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & d \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & d \\ 0 & -1 & d \end{bmatrix} \cdot \begin{bmatrix} v \\ v_n \\ \omega \end{bmatrix} \tag{5.32}$$

Assim, é possível, através de 5.31 e 5.32, reescrever a força efetuada pelos três motores do robô em função da tensão aplicada em cada um deles e da velocidade do robô (v, v_n, ω) tal como ilustra a equação 5.33.

$$\begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = \frac{l \cdot K}{R \cdot r} \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \end{bmatrix} - \frac{l^2 \cdot K^2}{R \cdot r^2} \cdot \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & d \\ -\frac{\sqrt{3}}{2} & \frac{1}{2} & d \\ 0 & -1 & d \end{bmatrix} \cdot \begin{bmatrix} v \\ v_n \\ \omega \end{bmatrix} \tag{5.33}$$

O conjunto de equações 5.14, 5.15 e 5.16 podem ser traduzidas para a forma matricial (5.34).

$$\begin{bmatrix} F_v(t) \\ F_{v_n}(t) \\ T(t) \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2M} & -\frac{\sqrt{3}}{2M} & 0 \\ \frac{1}{2} & \frac{1}{2} & -1 \\ \frac{d}{J} & \frac{d}{J} & \frac{d}{J} \end{bmatrix} \cdot \begin{bmatrix} f_1(t) \\ f_2(t) \\ f_3(t) \end{bmatrix} \tag{5.34}$$

Assim, pode-se, também, reescrever as equações da dinâmica do sistema de uma forma matricial (5.35), utilizando as equações 5.11 a 5.13, 5.19 a 5.24 e 5.34.

$$\begin{bmatrix} \frac{\mathrm{d}v}{\mathrm{d}t} \\ \frac{\mathrm{d}v_n}{\mathrm{d}t} \\ \frac{\mathrm{d}\omega}{\mathrm{d}t} \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{3}}{2M} & -\frac{\sqrt{3}}{2M} & 0 \\ \frac{1}{2} & \frac{1}{2} & -1 \\ \frac{d}{J} & \frac{d}{J} & \frac{d}{J} \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} + \tag{5.35}$$
$$+ \begin{bmatrix} -\frac{B_v}{M} & 0 & 0 \\ 0 & -\frac{B_{v_n}}{M} & 0 \\ 0 & 0 & -\frac{B_\omega}{J} \end{bmatrix} \cdot \begin{bmatrix} v \\ v_n \\ \omega \end{bmatrix} + \begin{bmatrix} -\frac{C_v}{M} & 0 & 0 \\ 0 & -\frac{C_{v_n}}{M} & 0 \\ 0 & 0 & -\frac{C_\omega}{J} \end{bmatrix} \cdot \begin{bmatrix} \mathrm{sign}(v) \\ \mathrm{sign}(v_n) \\ \mathrm{sign}(\omega) \end{bmatrix}$$

Portanto, das equações 5.33 e 5.35, depreende-se que é possível reescrever dinâmica completa do robô como um sistema de estado de espaços mais uma componente não linear (5.36).

$$\frac{\mathrm{d}X(t)}{\mathrm{d}t} = A \cdot X(t) + B \cdot U(t) + N \cdot \mathrm{sign}(X(t)) \tag{5.36}$$

$$X = \begin{bmatrix} v & v_n & \omega \end{bmatrix}^T \tag{5.37}$$

$$A = \begin{bmatrix} -\frac{3 \cdot K^2 \cdot l^2}{2 \cdot M \cdot r^2 \cdot R} - \frac{B_v}{M} & 0 & 0 \\ 0 & -\frac{3 \cdot K^2 \cdot l^2}{2 \cdot M \cdot r^2 \cdot R} - \frac{B_{v_n}}{M} & 0 \\ 0 & 0 & -\frac{3 \cdot K^2 \cdot l^2 \cdot d^2}{J \cdot r^2 \cdot R} - \frac{B_\omega}{J} \end{bmatrix} \tag{5.38}$$

$$B = \frac{K \cdot l}{R \cdot r} \cdot \begin{bmatrix} \frac{\sqrt{3}}{2M} & -\frac{\sqrt{3}}{2M} & 0 \\ \frac{1}{2} & \frac{1}{2} & -1 \\ \frac{d}{J} & \frac{d}{J} & \frac{d}{J} \end{bmatrix} \tag{5.39}$$

$$N = \begin{bmatrix} -\frac{C_v}{M} & 0 & 0 \\ 0 & -\frac{C_{v_n}}{M} & 0 \\ 0 & 0 & -\frac{C_\omega}{J} \end{bmatrix} \cdot \tag{5.40}$$

Desta forma é possível compreender como é que evolui o sistema e analisar mais facilmente o seu comportamento e algumas das suas não linearidades. Estas podem, eventualmente, ser rejeitadas, caso possuam uma ordem de grandeza pequena, podendo-se, assim, aproximar o comportamento do robô por um sistema linear.

Capítulo 6

Simulador

Uma vez que grande parte do trabalho foi feito em ambiente de simulação, neste capítulo pretende-se abordar a forma como funciona o simulador utilizado e as adaptações que foram feitas, para que a simulação obtida fosse o mais próxima quanto possível do robô real.

Para este efeito, recorreu-se ao simulador SimTwo [34] desenvolvido pelo professor Paulo Costa no âmbito de projetos de sistemas robóticos.

6.1 Ambiente de Simulação

O SimTwo é um sistema de simulação realista que tem por base a biblioteca ODE [37] para a simulação dos corpos rígidos. Com este simulador é possível implementar diferentes tipos de robôs:

- Robôs de tração diferencial
- Robôs de tração omnidirecional
- Manipuladores
- Quadrúpedes
- Humanoides
- Veículos aéreos e submarinos com ou sem hélices para propulsão

Para se conseguir o realismo desejado, cada robô é decomposto por um conjunto de corpos rígidos, motores e hélices, descritos num ou mais ficheiros XML, figura 6.2. A dinâmica associada a estes corpos é simulada numericamente, considerando as suas características e restrições físicas.

Adicionalmente, as articulações entre os corpos podem ser especificadas, garantindo o efeito desejado, podendo também ser-lhes associado um sistema de acionamento e sensores.

O sistema de acionamento das articulações e das rodas (no caso dos robôs terrestres) é, tipicamente, constituído por um motor DC com ou sem caixa redutora e um controlador. É também possível simular, no motor, algumas das suas não linearidades, como é o caso do limite de corrente, do atrito de Coulomb e viscoso e da saturação da tensão aplicada.

Além do controlo de baixo nível que pode ser implementado numa folha XML, o SimTwo permite ainda que seja feito um de alto nível, onde se pode implementar controladores de trajetórias, entre outros. O acesso a este nível pode ser feito através de um programa externo que comunica por UDP ou porta série, ou implementando e compilando código Pascal num IDE do próprio simulador, figura 6.3.

Este permite ainda uma elevada interação com o utilizador enquanto está a ser executada a simulação, permitindo a observação e o envio de dados e instruções para o programa, utilizando uma folha de cálculo, conforme ilustra a figura 6.4.

Figura 6.1: Visualização da simulação

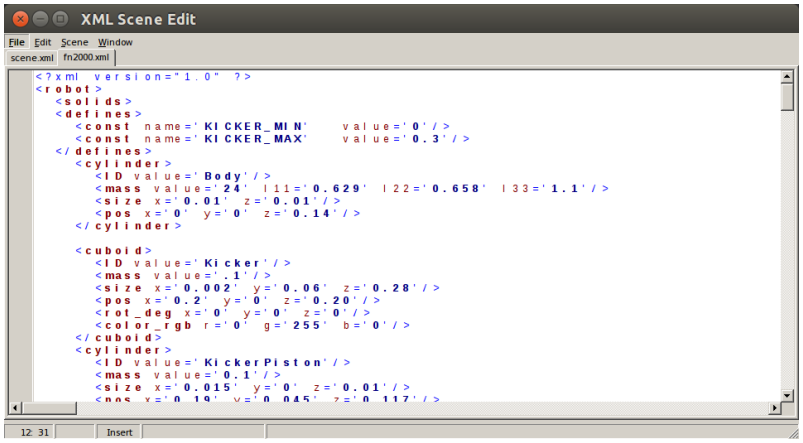


Figura 6.2: Ambiente de configuração do robô em XML

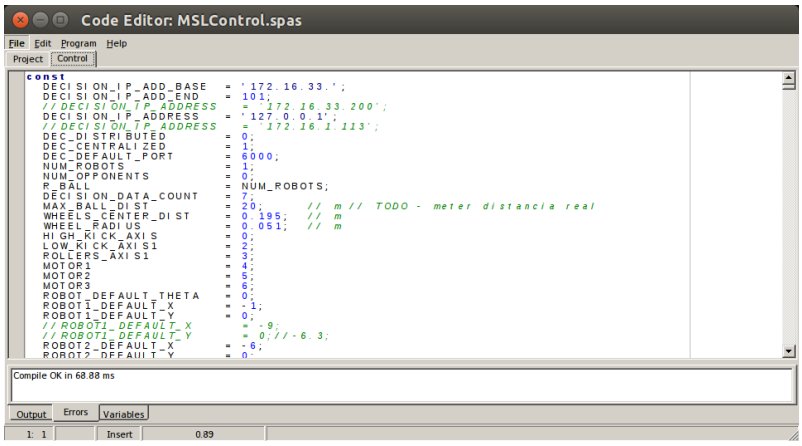


Figura 6.3: Editor Pascal script para o controlo de alto nível

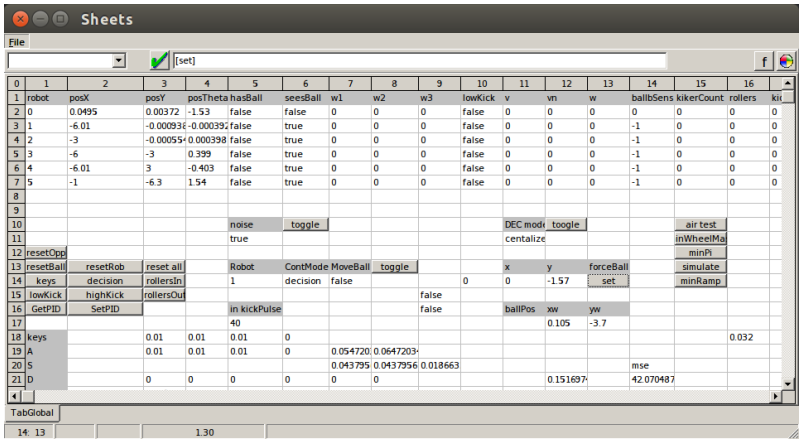


Figura 6.4: Registo de dados e envio de dados para o simulador (interação com o utilizador)

6.2 Modelação do robô para simulação

Tal como referido no capítulo 5, para que os resultados obtidos em simulação sejam o mais próximos possíveis dos obtidos no robô real é importante fazer-se uma correta parametrização deste, para se garantir um bom modelo de simulação, equivalente ao real. Assim, tomou-se como ponto de partida o modelo construído por Pedro Relvas [38] e as medições feitas por Nascimento *et al.* [39] na tabela 6.2.

A calibração do modelo do robô foi feita em duas fases principais. Numa primeira etapa procurou-se estimar os parâmetros dos motores, tendo em conta, apenas, as suas características. De seguida, afinou-se as restantes componentes (controladores dos motores, momento de inércia e rampas de aceleração) tendo e conta o comportamento global do robô. De modo a facilitar o processo de estimação, foram considerados os valores especificados nas folhas de características dos motores, *encoders* e caixa redutora, tabela 6.1.

Descrição	**Medida**	**Unidade**
Redução da caixa	1:12	
Resistência do motor (R_i)	$0,317$	Ω
Constante elétrica do motor (K)	$0,0302$	N·m/A
Encoder	256	PPR

Tabela 6.1: Parâmetros característicos dos motores, caixa redutora e *encoders*

Descrição	**Medida**	**Unidade**
Massa do robô	26,2	kg
Massa das rodas	0,660	kg
Raio das rodas	0,0513	m
Momento de inércia em x	0,629	kg·m^2
Momento de inércia em y	0,658	kg·m^2
Momento de inércia em z	0,705	kg·m^2

Tabela 6.2: Parâmetros de inicialização do modelo do robô no simulador

6.2.1 Determinação do modelo de atrito dos motores

Tal como mencionado na secção 5.2.2, os motores DC com escovas utilizados no robô podem ser aproximados pelo modelo da equação 5.28, também transcrita abaixo.

$$u(t) = R_i \cdot I(t) + K_e \cdot \omega_m \tag{6.1}$$

Pelas leis de Newton para a rotação [36], sabe-se que o somatório dos binários aplicados é igual ao momento de inércia multiplicado pela aceleração angular, ver equação 5.29 ou 6.2.

$$J \cdot \frac{\mathrm{d}\omega}{\mathrm{d}t} = K \cdot I - B_v \cdot \omega - F_c \tag{6.2}$$

Considerando as equações 5.26 e 5.27, sabe-se que as constantes K_i e K_e são iguais quando expressas em unidades do sistemas internacional, como referido na secção 5.3. Assim, se se considerar o regime permanente, isto é, $J \cdot \frac{\mathrm{d}\omega}{\mathrm{d}t} = 0$, pode-se, pelas equações 6.1 e 6.2, relacionar linearmente a tensão entre os terminais do motor, v, e a velocidade de rotação do motor, ω, obtendo-se a equação 6.3.

$$v = \left(\frac{R_i \cdot B_v}{K} + K\right) \cdot \omega + \frac{R_i \cdot F_c}{K} \tag{6.3}$$

Dada a relação linear que se verifica na equação 6.3, é possível determinar B_v e F_c, excitando-se os motores com diferentes níveis de tensão e medindo a sua velocidade em regime permanente. O atrito de Coloumb corresponde à primeira tensão cuja velocidade é diferente de zero, enquanto que o atrito viscoso caracteriza o aumento da velocidade com a tensão.

Para determinar ambos os atritos, realizou-se quatro ensaios em dois motores, aplicando-se diferentes tensões com um passo de 1V. De seguida, procurou-se a reta que melhor aproxima o conjunto de ponto medidos, obtendo-se o resultado ilustrado na figura 6.5, que corresponde a um atrito viscoso $B_v = 0,0324$ N·m/rad/s e a um atrito estático $F_c = 0,036735$ N·m.

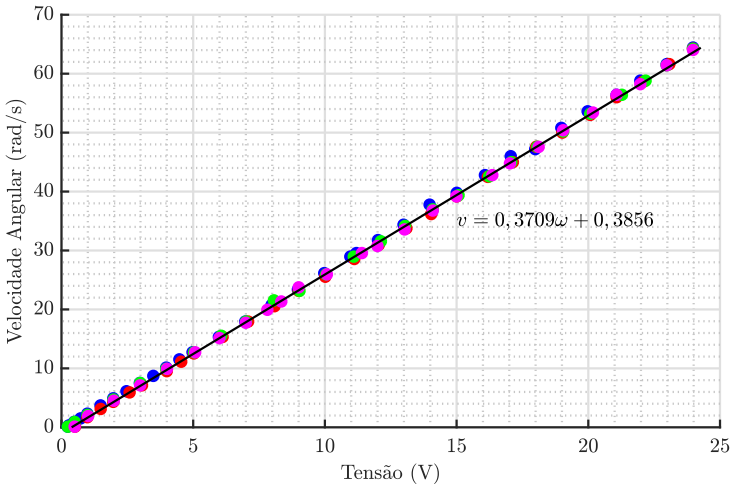


Figura 6.5: Reta de regressão linear da relação tensão–velocidade angular

Uma vez caracterizado o atrito viscoso e de Coulomb, efetuou-se um teste semelhante ao que foi feito nos motores do robô real nos motores do robô do simulador, com o intuito de validar os resultados obtido. Assim, verifica-se que os resultados são relativamente próximos, com um erro quadrático médio de 1,3112, tal como ilustra a figura 6.6.

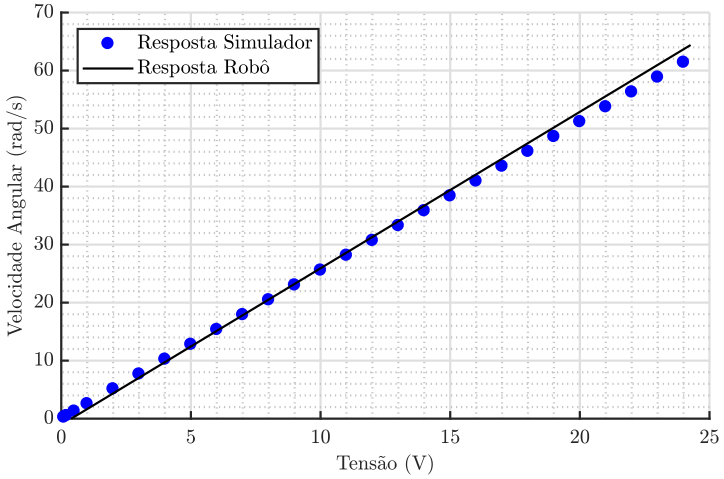


Figura 6.6: Validação do atrito viscoso e de Coulomb entre os motores do robô e do simulador

6.2.2 Determinação da altura do centro de massa

Para simplificação dos sistemas, procurando reduzir o peso computacional da simulação, é comum representar-se a massa dos corpos acumulada no seu centro de massa. Considerando o sistema do robô como um único corpo, o centro de massa do robô corresponde ao ponto hipotético onde toda a massa do sistema está concentrada e que se move como se todas as forças externas fossem aplicadas nesse ponto.

Portanto, por simplificação, assume-se que o centro de massa está centrada no robô segundo as componentes x e y. A altura do centro de massa, corresponde ao ponto onde o robô se equilibra horizontalmente quando suspenso, tal como ilustra a figura 6.7. Experimentalmente, concluiu que este se estabelece a 15 cm de altura.

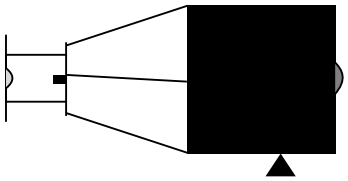


Figura 6.7: Determinação da altura do centro de massa

6.2.3 Determinação do limite de aceleração dos motores

Da figura 6.8 verifica-se que o desempenho do robô está fortemente limitado por uma rampa de aceleração. Portanto, para se conseguir obter um bom modelo do robô no simulador, é importante caracterizar este valor. Da leitura das configurações dos *drivers* dos motores, verifica-se um limite de aceleração máximo do motor de 7000 rpm/s.

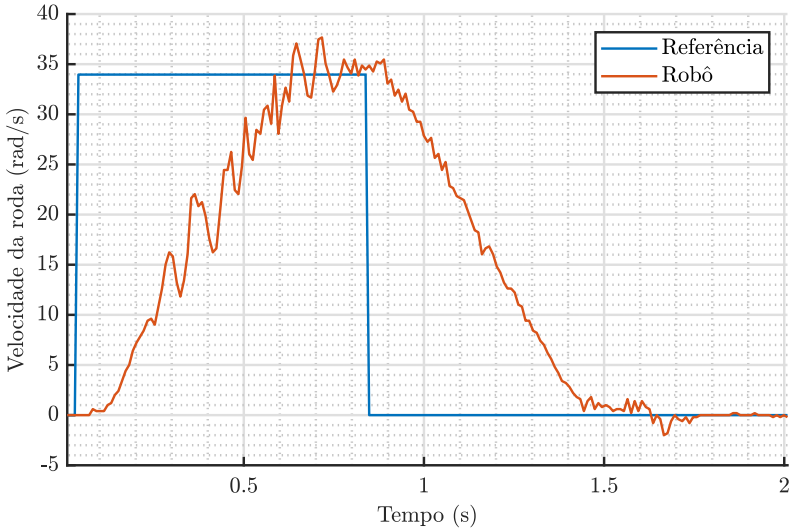


Figura 6.8: Curva de aceleração de uma roda

Dado que o simulador está configurado em unidades do sistema internacional e não permite uma configuração direta deste limite de aceleração, que tem de ser implementado por *software* no *script* de controlo, através da equação 6.4, tem-se que o limite de aceleração das rodas é de 61,09 rad/s^2.

$$a_{\mathrm{rad/s^2}} = \frac{a_{\mathrm{rpm/s}} \cdot \frac{2\pi}{60}}{l} \tag{6.4}$$

Da equação 6.4 tem-se que l é o fator de redução da caixa redutora que está especificado na tabela 6.1.

6.2.4 Estimativa dos restantes parâmetros por métodos numéricos

Dada a complexidade destes sistemas, em vez de se determinar experimentalmente ou analiticamente os outros parâmetros que caracterizam o robô, tais como os ganhos dos controladores dos motores, o momento de inércia, a limitação da aceleração ou o atrito de contacto do robô com a superfície, pretende-se aproximá-los através de métodos iterativos de otimização como o gradiente descendente ou o *resilient propagation* (anexo A), no sentido de minimizar uma função custo, como a ilustrada na equação 6.5, onde n é o número de pontos medidos, $\hat{x}$ o valor medido no simulador e x o valor real medido do robô.

$$Custo = \frac{1}{n} \cdot \sum_{i=1}^{n} (\hat{x_i} - x_i)^2 \tag{6.5}$$

Assim, para a estimação dos parâmetros especificados no parágrafo anterior, recorreu-se ao algoritmo de *resilient propagation*, tal como enunciado no anexo A, em conjunto com a função custo 6.5. Para a excitação do sistema, consideraram-se dois modelos: um contendo apenas movimento angular, como (figura 6.9), e outro contendo movimento linear segundo a componente **v**, (figura 6.10). Em ambos, os sinais foram escolhidos por forma a garantir alguma excitabilidade do sistema, sem amplitudes baixas, afastando-se, desta forma, da zona morta, e sem a saturação dos motores. A duração de cada degrau é selecionada por forma a garantir que a velocidade se aproxima do valor desejado, sem prolongar o estado em regime permanente. A duração total dos modelos para a excitação garante que ensaios possam ser realizados num robô real num espaço limitado.

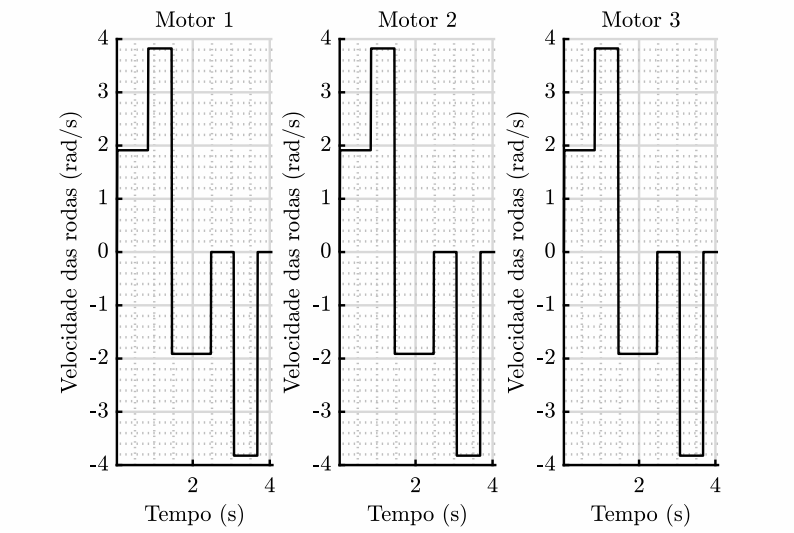


Figura 6.9: Modelo para excitação dos motores com movimento angular

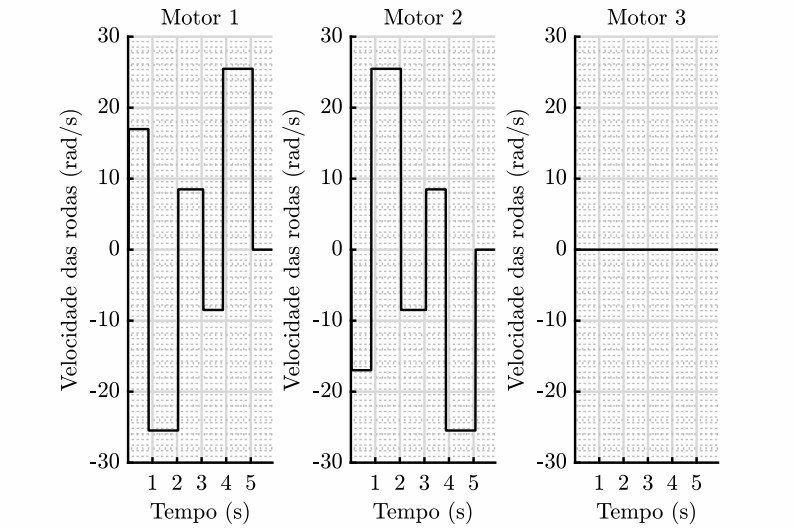


Figura 6.10: Modelo para excitação dos motores com movimento linear

Cada motor do robô pressupõe um controlador de velocidade, por forma a garantir a velocidade de referência. Este pode ser formulado como 6.6.

$$\begin{aligned} v_{out} &= k_p \cdot e + k_i \int e \ \mathrm{d}t + k_d \frac{\mathrm{d}e}{\mathrm{d}t} + k_f \cdot \omega_{ref} \\ e &= \omega_{ref} - \omega \end{aligned} \tag{6.6}$$

Dado que não se tem acesso ao código implementado nos drives dos motores e não se sabe qual a equação de controlo implementada, não é possível determinar quais os

ganhos dos motores equivalentes para o modelo do robô simulado. Desta forma, sente-se a necessidade de aproximar estes valores numericamente por aproximação a uma resposta similar.

Assim, começou-se por estimar os ganhos dos controladores PID das rodas, tendo por base o modelo de excitação com movimento angular, evitando desta forma o efeito do limite de aceleração implementado no robô, dado que neste modelo as velocidades das rodas são mais baixas. Uma vez minimizado o erro, obteve-se a aproximação da figura 6.11, com os ganhos do controlador dos motores das rodas: $k_p = 0,06472$, $k_i = 0,043796$ e $k_d = 0$.

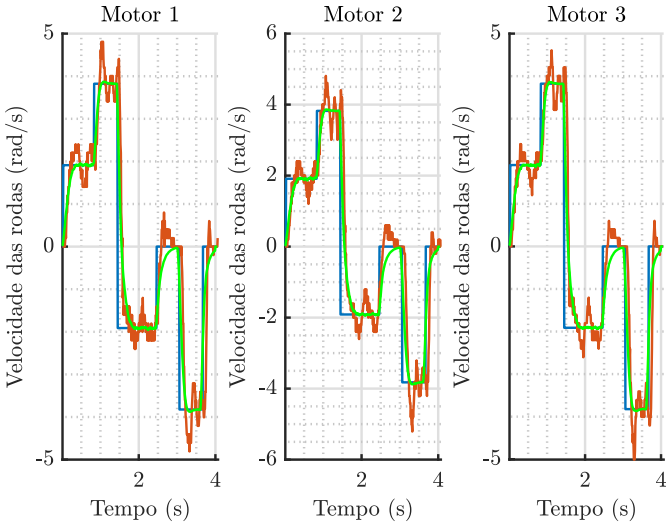


Figura 6.11: Aproximação dos ganhos do controlador das rodas (azul – referência; vermelho – robô; verde – simulador)

De seguida, ajustou-se o momento de inércia segundo a componente z do robô, concluindo-se que o valor que melhor caracteriza o seu comportamento é de 1,1 kg·m^2 (figura 6.12).

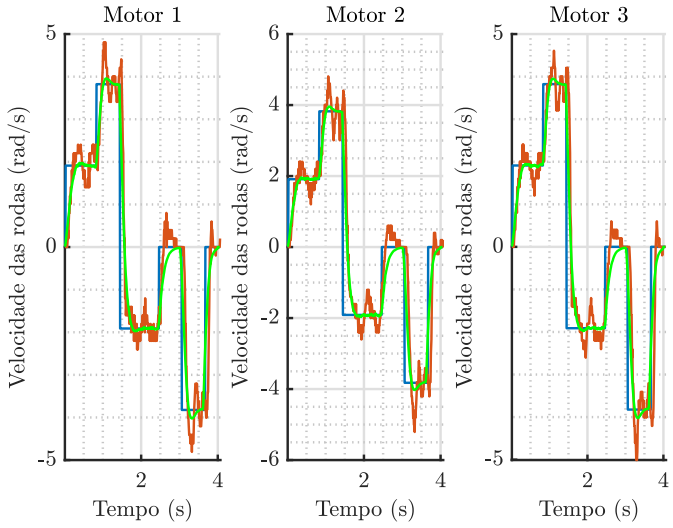


Figura 6.12: Aproximação do momento de inércia do robô segundo a componente z (azul – referência; vermelho – robô; verde – simulador)

6.2.5 Conclusão

Uma vez definido e calibrados todos os parâmetros, conclui-se que aqueles que melhor caracterizam o sistema do robô em estudo são os especificados na tabela 6.3 que se ajustam aos modelos de excitação do sistema (figuras 6.9 e 6.10) conforme se demonstra nas figuras 6.13 e 6.14, obtidas no final do processo de calibração.

Restrição	**Medida**	**Unidade**
Redução da caixa	12	
Resistência do motor (R_i)	0,317	Ω
Constante elétrica do motor (K)	0,0302	N·m/A
Encoder	256	PPR
Massa do robô	26,2	kg
Massa das rodas	0,660	kg
Raio das rodas	0,0513	m
Momento de inércia em x	0,629	kg·m^2
Momento de inércia em y	0,658	kg·m^2
Momento de inércia em z	1,1	kg·m^2
Coeficiente de multiplicação da força de atrito de Coulomb	0,036735	N·m
Coeficiente de multiplicação da força de atrito viscoso	0,0324	N·m/rad/s
Altura do centro de massa	0,015	m
Ganho proporcional do controlador	0,06472	
Ganho integral do controlador	0,043796	
Ganho derivativo do controlador	0	
Ganho *feedforward* do controlador	0	
Limite de aceleração	61,09	rad/s^2

Tabela 6.3: Parâmetros finais de ajuste do modelo do robô no simulador

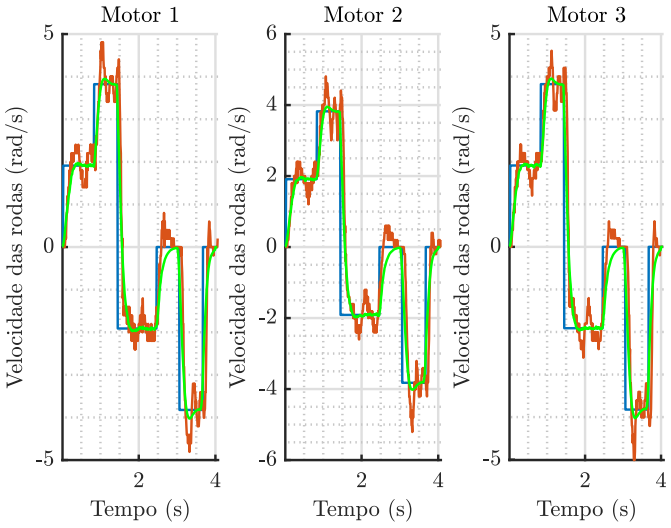


Figura 6.13: Aproximação ao robô com movimento rotacional (azul – referência; vermelho – robô; verde – simulador)

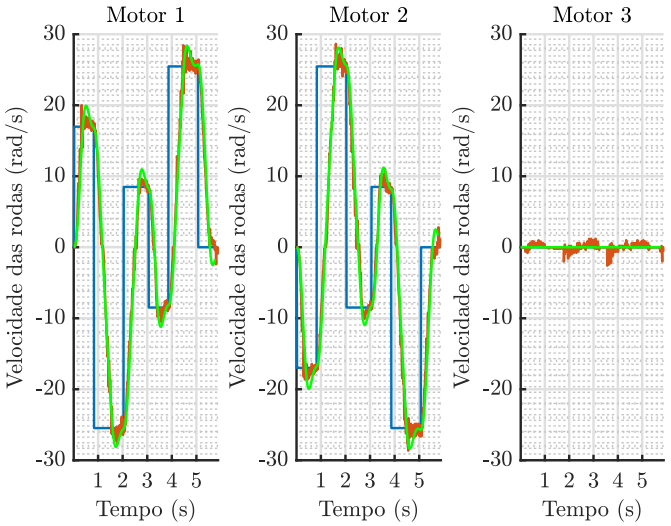


Figura 6.14: Aproximação ao robô com movimento linear (azul – referência; vermelho – robô; verde – simulador)

Capítulo 7

Controlo Reativo

O controlo reativo é normalmente caracterizado pela realimentação parcial do estado do robô para se calcular a lei de controlo que minimiza o erro de seguimento. Esta é tipicamente projetada sobre o formato de um controlador PID, tal como se ilustra na figura 7.1

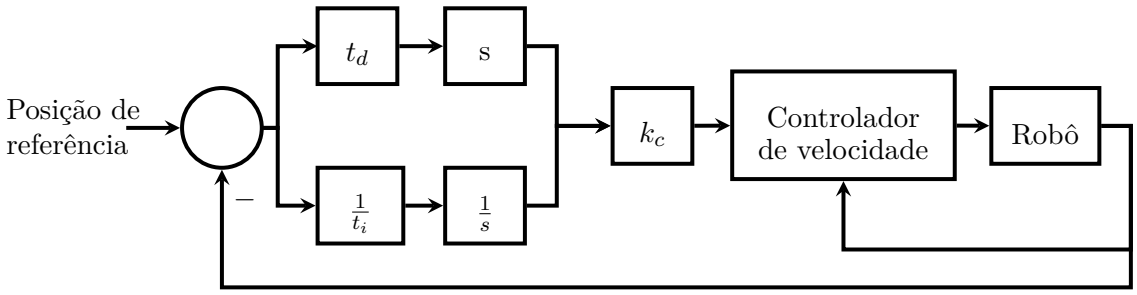


Figura 7.1: Estrutura de controlo reativa baseada num controlador PID

7.1 Controlador *go to position*

Se se considerar uma trajetória como um conjunto de pontos, é possível estabelecer o controlo de trajetórias como o percurso de o robô transitar de um ponto para o outro. Desta forma, uma estrutura de seguimento de trajetórias do tipo *go to position* pode ser uma boa solução para este problema, ficando o movimento linear do robô caracterizado por um único vetor que aponta para o ponto de destino, tal como ilustra a figura 7.2.

Dado o facto de o robô ser omnidirecional, o que lhe confere a possibilidade de se movimentar em qualquer direção, a máquina de estados que caracteriza esta estrutura de controlo pode ser estabelecida por três estados. Um primeiro que movimenta o robô até ao ponto, cujo movimento é caracterizado pelo vetor **p** da figura 7.2. Quando o robô está relativamente próximo do final da trajetória, entra num segundo estado onde lhe é conferida uma velocidade mais reduzida, por forma a parar no último ponto desta. O robô

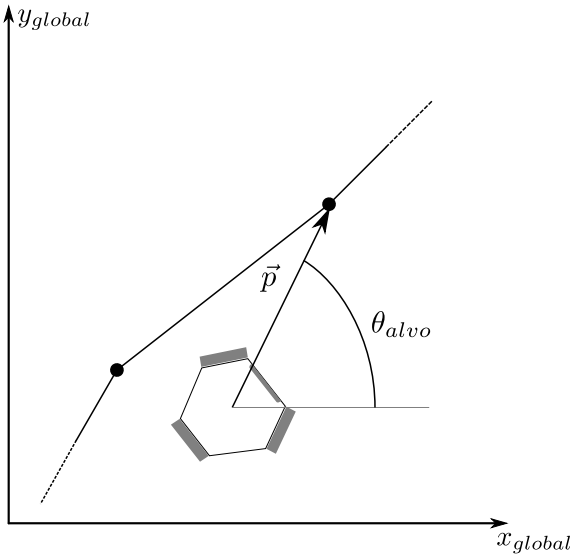


Figura 7.2: Descrição de uma trajetória para o controlador *go to position*

entra no terceiro estado quando atinge o ponto final e neste apenas é induzido a rodar sobre si próprio até anular o erro entre o seu ângulo e ângulo de referência, caso ainda não o tenha feito durante a trajetória.

A figura 7.3 descreve graficamente a evolução deste algoritmo, onde e_{dist} é a distância que falta até ao final da trajetória e $\Delta\theta$ é o erro entre o ângulo atual do robô e o ângulo de referência. $F(\cdot)$, $H(\cdot)$ e $I(\cdot)$ são funções que calculam a lei de controlo.

Por fim, a distância de desaceleração é dada pela equação 7.1, onde a_v é a aceleração máxima do robô, v_{ref} é a velocidade de referência à qual o robô está a executar a trajetória, v_{da} é a velocidade do robô na zona de desaceleração e d_{min} é a distância mínima para o robô entrar nesta.

$$dist\hat{a}ncia\ desacelera\c{c}\tilde{a}o = d_{min} - \frac{a_v}{2} \cdot \left(\frac{v_{ref} - v_{da}}{a_v} \right)^2 + v_{ref} \cdot \frac{v_{ref} - v_{da}}{a_v} \tag{7.1}$$

7.1.1 Cálculo da lei de controlo

Para calcular o sinal de controlo, optou-se por um controlador do tipo PD, posicionando-se os polos em malha fechada do sistema segundo o protótipo de Bessel [40]. Este protótipo coloca estes polos nas raízes dos polinómios de Bessel, garantindo na ausência de zeros na função de transferência, uma resposta ao degrau sem sobrelongação.

Considerando um sistema do tipo $B_k(0)/B_k(s)$, a tabela 7.1 especifica as raízes do polinómio de Bessel para sistemas até ordem 3 com um tempo de estabelecimento padrão de 1 s. Como tipicamente se pretende que o sistema responda e atinja o regime permanente o mais rápido possível, pode-se alterar o tempo de estabelecimento, dividindo estes polinómios pelo tempo de estabelecimento desejado.

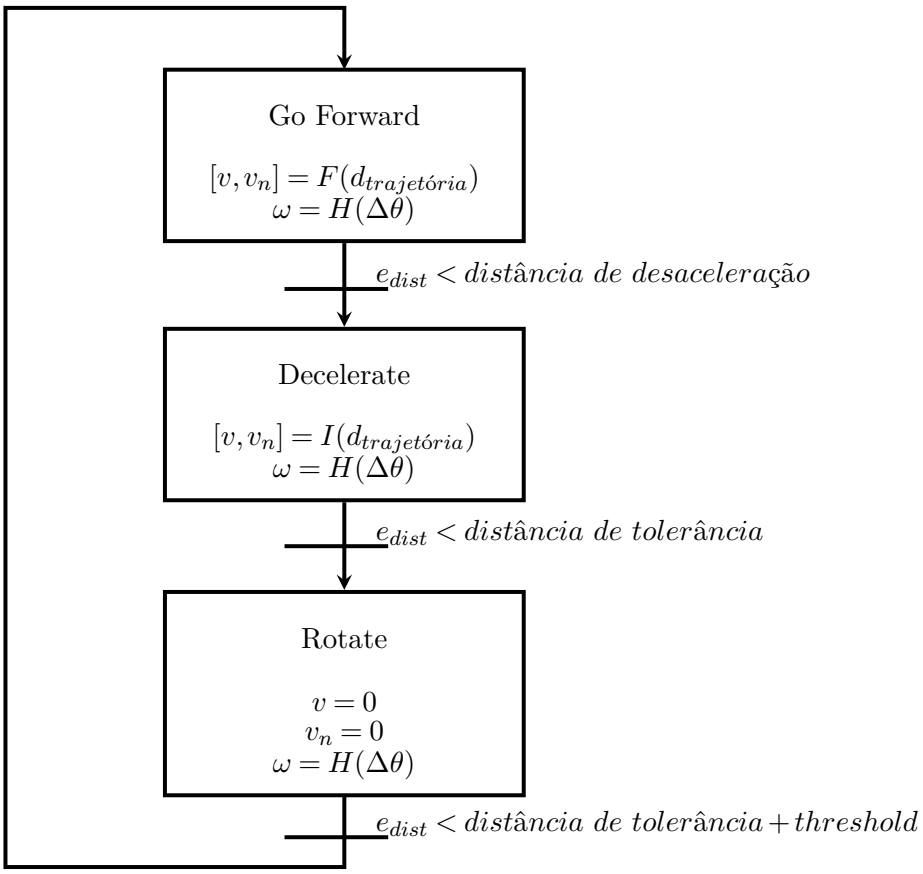


Figura 7.3: Descrição de uma trajetória para o controlador *go to position*

k	Localização dos pólos de $B_k(s)$
1	$-4,6200$
2	$-4,0530 \pm j2,3400$
3	$-5,0093,\ -3,9668 \pm j3,7845$

Tabela 7.1: Raízes normalizadas dos polinómios de Bessel correspondentes a um tempo de estabelecimento de 1 s

Portanto, para se poder utilizar este controlador, é necessário que haja um modelo do sistema. Embora na secção 6.2.3 se tenha concluído que o modelo do sistema está fortemente limitado por uma rampa de aceleração, optou-se por aproximá-lo pelo modelo de primeira ordem da equação 7.2, onde a constante de tempo, τ, é adaptativa em função da velocidade de referência à qual se pretende que o robô execute uma determinada trajetória e o ganho do sistema, k, é constante e igual a 1.

$$G_v(s) = \frac{k}{\tau s + 1} \tag{7.2}$$

Para se determinar a constante de tempo, excitou-se o sistema com vários degraus para

diferentes velocidades de referência entre 0 e 2 m/s, depois mediu-se esta constante em dois pontos, ao acelerar e ao travar, resultando nos valores expostos no gráfico da figura 7.4. Se se aproximar estes pontos por uma função polinomial de segunda ordem, tem-se que a constante de tempo do sistema a uma determinada velocidade linear de referência, v em metros por segundo, pode ser obtida pela função da equação 7.3, sendo que se se eliminar a rampa de aceleração que caracteriza o comportamento do robô, a constante de tempo mínima do sistema é de 0,089 s.

$$\tau = 0,06048 \cdot v_{ref}^2 + 0,02721 \cdot v_{ref} + 0,1 \tag{7.3}$$

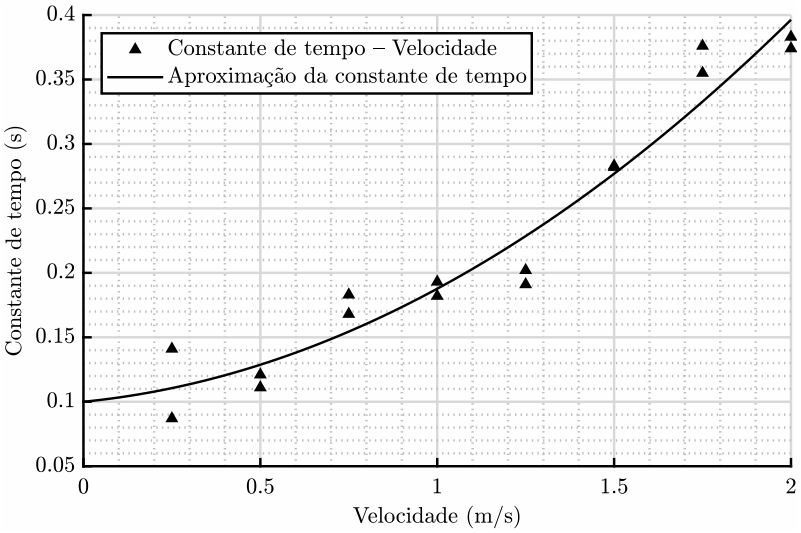


Figura 7.4: Evolução da constante de tempo do sistema com o aumento da velocidade

Das equações 7.2 e 7.3 obtém-se um modelo do sistema para a resposta em velocidade. Uma vez que se pretende controlar o robô em posição, é necessário converter-se o modelo para este requisito, sendo suficiente a multiplicação de $G_v(s)$ por um integrador, $\frac{1}{s}$, obtendo-se a função de transferência apresentada em 7.4, cuja entrada é a posição para onde o robô pretende ir e a saída é a velocidade linear.

$$G(s) = \frac{k_p}{\tau \cdot s^2 + s} \tag{7.4}$$

Antes de se poder reposicionar os polos, é necessário calcular o sistema em malha fechada, de acordo com o modelo da figura 7.5, onde k_c é o ganho da componente proporcional do controlador, $k_d = k_c \cdot t_d$ é o ganho da componente derivativa e $G(s)$ é um modelo do sistema, que neste caso é igual ao mencionado na equação 7.4.

Substituindo-se (7.4) em (7.5), resulta a função de transferência em malha fechada 7.6.

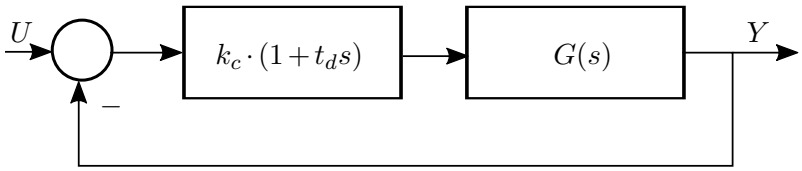


Figura 7.5: Estrutura do controlador para o projeto do protótipo de Bessel

$$\frac{Y(s)}{U(s)} = \frac{G(s) \cdot C(s)}{1 + G(s) \cdot C(s)} \tag{7.5}$$

$$= \frac{k_c \cdot k_p \cdot (1 + t_d s)}{(\tau s + 1) \cdot s + k_c \cdot k_p \cdot (1 + t_d s)} \tag{7.6}$$

Para reposicionar os polos do sistema apenas é necessário manipular os valores k_c e t_d do denominador do sistema, $D(s)$. Portanto, considerando apenas esta parte e simplificando-a para a forma canónica dos polinómios $x^2 + b_1 x + b_2$, o denominador do sistema pode ser apresentado conforme se ilustra na equação 7.7.

$$D(s) = s^2 + \left(\frac{1}{\tau} + \frac{k_c \cdot k_p \cdot t_d}{\tau}\right) \cdot s + \frac{k_p \cdot k_c}{\tau} \tag{7.7}$$

Por sua vez, o polinómio de Bessel é descrito pela equação 7.8, onde b_2 e $\bar{b_2}$ são, respetivamente, o polo do polinómio de Bessel de ordem 2 e o seu complexo conjugado tal como especificados na tabela 7.1, r_{b_2} e i_{b_2} são a componente real e imaginária do polo b_2 e t_{set} é o tempo de estabelecimento desejado para o sistema a controlar.

$$\left(s - \frac{b_2}{t_{set}}\right) \cdot \left(s - \frac{\bar{b_2}}{t_{set}}\right) = s^2 - \left(2 \cdot \frac{r_{b_2}}{t_{set}}\right) s + \frac{r_{b_2}^2 + i_{b_2}^2}{t_{set}} \tag{7.8}$$

Dada a similaridade entre as equações 7.7 e 7.8, é possível estabelecer-se a igualdade entre ambas, de onde se obtém que o k_c e t_d podem ser calculados diretamente das equações 7.9 e 7.10.

$$k_c = \frac{r_{b_2}^2 + i_{b_2}^2}{t_{set}} \cdot \frac{\tau}{k_p} \tag{7.9}$$

$$t_d = -2 \cdot \frac{r_{b_2}}{t_{set}} \cdot \frac{\tau}{k_p \cdot k_c} \tag{7.10}$$

Da figura 7.3, a função $F(d_{trajetória})$ calcula o vetor $[v\ v_n]^T$ que direciona o robô para o final da trajetória, passando pelos pontos desta. Esta função depende da distância que falta percorrer até ao final desta, $d_{trajetória}$ e da direção para o próximo ponto da trajetória, θ_{alvo}, sendo assim obtida pela equação 7.11, onde k_p é o ganho de um controlador

proporcional para a distância, que experimentalmente aproximou-se como igual a 0,1 (valor que minimiza o erro de seguimento e o *overshoot*).

$$\begin{aligned} F_v &= k_p \cdot d_{trajetória} \cdot \cos(\theta_{alvo}) \\ F_{v_n} &= k_p \cdot d_{trajetória} \cdot \sin(\theta_{alvo}) \end{aligned} \tag{7.11}$$

Da mesma figura, a função $H(\Delta\theta)$ calcula a velocidade angular do robô para atingir, com sucesso, o ângulo desejado (onde $\Delta\theta$ é o erro entre o ângulo atual do robô e o ângulo desejado), sendo esta velocidade controlada por um controlador PD, tal como especificado até agora, considerando-se um tempo de estabelecimento de 1,3 s.

Do estado *decelerate*, tem-se que $I(d_{trajetória})$ é um controlador PD afinado por protótipo de Bessel, considerando um tempo de estabelecimento de 1,5 s, medido experimentalmente.

7.2 Controlador *follow line*

Adicionalmente, optou-se por compreender se há vantagem na utilização de um controlador *follow line* para que o robô consiga executar, com sucesso, a trajetória pretendia. Neste, o movimento linear do robô é caracterizado pela soma do vetor **p** com o vetor **l**, ilustrados na figura 7.6. O primeiro vetor, há semelhança do que acontece com o controlador de trajetórias *go to position* estabelece o sentido e a direção do movimento do robô para o ponto alvo. Por outro lado, uma vez que se pretende seguir um linha, o vetor **l** é perpendicular à linha a seguir e estabelece o movimento do robô de aproximação à linha. Desta forma, garante-se que este é capaz de seguir uma linha e parar num ponto alvo especificado.

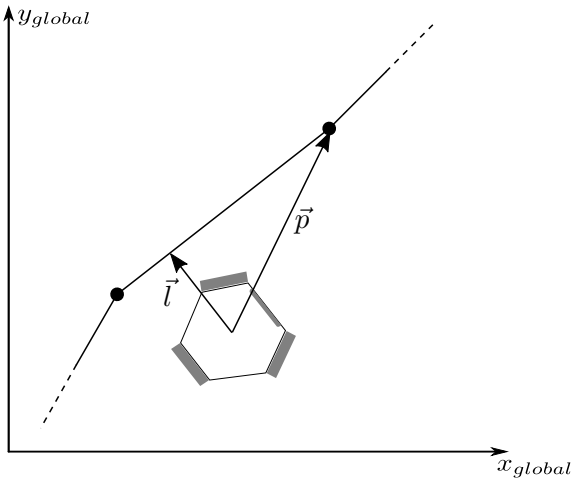


Figura 7.6: Descrição de uma trajetória para o controlador *follow line*

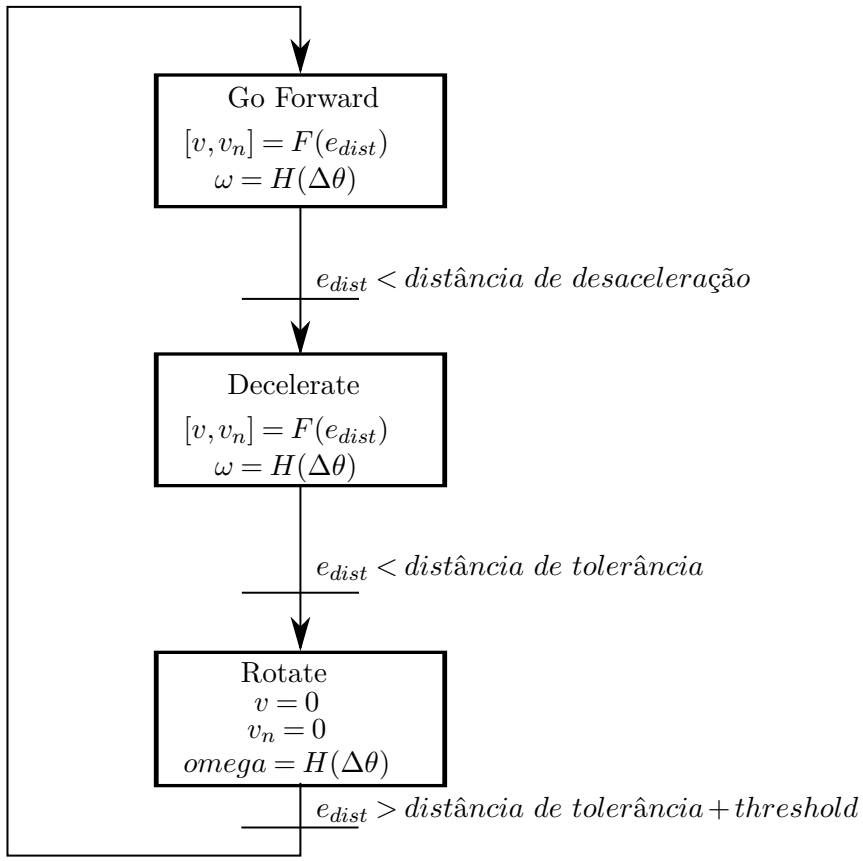


Figura 7.7: Máquina de estados do controlador de trajetórias do tipo *follow line*

A figura 7.7 descreve, graficamente, a máquina estado que permite o controlo da trajetória do robô através de uma linha até uma posição com um determinado ângulo de referência. Nela, e_{dist} é distância que falta até ao final da trajetória e $\Delta\theta$ é o erro em relação ao ângulo de referência. $F(\cdot)$ e $H(\cdot)$ são funções que calculam a lei de controlo, onde a última é um controlador PD afinado usando o protótipo de Bessel. Por fim, a distância de desaceleração é dada pela equação 7.1.

7.2.1 Cálculo da lei de controlo

De forma similar ao controlador de trajetórias *go to position*, o *follow line* recorre a uma lei de controlo baseada num controlador PD com posicionamento dos polos segundo o protótipo de Bessel, tal como como explicado na secção 7.1.1.

No entanto, diferentemente à solução anterior, neste caso é sempre garantido que o robô executa a trajetória à velocidade de referência, exceto quando transita para o estado *decelerate*, onde entra num momento de desaceleração para velocidades mais baixas, procurando parar no ponto desejado.

Assim sendo, a velocidade linear do robô é dada pela equação 7.12, sendo que k_{ponto} é o resultado do controlador proporcional, cujo parâmetro se arbitrou, experimentalmente, como 4,5 e k_{linha} é o resultado de um controlador PD, cujos polos em malha fechada foram posicionados segundo o protótipo de Bessel, considerando um tempo de estabelecimento de 1 s.

Para a determinação destes parâmetros, começou-se por ajustar o ganho proporcional, k_{ponto}, até que o robô segui-se uniformemente uma trajetória linear sem variação do ângulo de referência, de seguida, considerou-se a trajetória em pulso e partiu-se de um tempo de estabelecimento elevado, reduzindo-o à procura de um ponto de equilíbrio entre o baixo *overshoot* de seguimento da trajetória e a baixa oscilação do robô. Nos pontos críticos, quando o robô começava a oscilar tentou-se incrementar o valor de k_{ponto} para verificar se haveria a redução desta última, mantendo-se um *overshoot* reduzido.

Os vetores $\mathbf{p}$ e $\mathbf{l}$ mencionados correspondem aos ilustrados na figura 7.6.

$$\mathbf{v_{global}} = k_{ponto} \cdot \mathbf{p} + k_{linha} \cdot \mathbf{l} \tag{7.12}$$

$$k_{ponto} = k_p \cdot ||\mathbf{p}|| \tag{7.13}$$

$$k_{line} = k_c \cdot (||\mathbf{l}|| + t_d \cdot \Delta||\mathbf{l}||) \tag{7.14}$$

Uma vez que o valor calculado pode ultrapassar a velocidade à qual se pretende que a trajetória seja executada, posteriormente, normalizou-se $\mathbf{v_{global}}$ e multiplicou-se pela velocidade de referência, tal como se ilustra na equação 7.15.

$$\mathbf{v_{global}} = v_{ref} \cdot \frac{\mathbf{v_{global}}}{||\mathbf{v_{global}}||} \tag{7.15}$$

A velocidade angular do robô pode ser calculada por um controlador PD segundo o protótipo de Bessel, tal como explicitado na secção 7.1.1, considerando um tempo de estabelecimento de 1,3 s, obtido experimentalmente.

7.2.2 Obtenção do vetor normal à trajetória

Da figura 7.6, o vetor $\mathbf{l}$ é um vetor que indica a direção e o sentido do robô para a trajetória, sendo ortogonal a esta. Portanto, para se obter este vetor é necessário obter-se a projeção ortogonal [40] deste na trajetória, considerando a linha formada pelos dois pontos mencionados na mesma figura.

Tome-se como exemplo a figura 7.8, tem-se que o vetor normal à trajetória, $\mathbf{l}$, pode ser obtido pelo conhecimento da posição do robô, X_r, no referencial global (sendo $\mathbf{x_r}$ o vetor que caracteriza esta posição em relação à origem do referencial), da localização atual do robô na trajetória, P_0, (ou seja, o *waypoint* atual do robô) e da direção e sentido da trajetória, definida pelo versor $\mathbf{n}$, sabendo que $\mathbf{n} = \mathbf{p_1} - \mathbf{p_0}$, onde $\mathbf{p_0}$ é o vetor que vai da

origem do referencial a P_0 e $\mathbf{p_1}$ é o vetor que vai da mesma origem até P_1, isto é, $\mathbf{p_0} = P_0$ e $\mathbf{p_1} = P_1$.

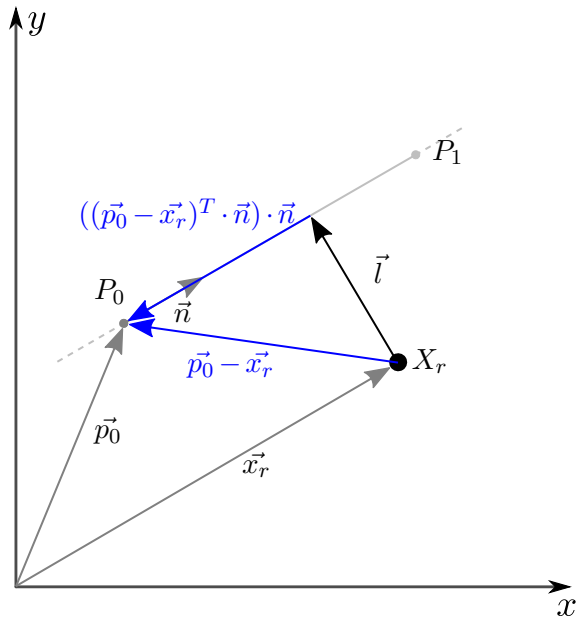


Figura 7.8: Obtenção do vetor normal à trajetória

Portanto, o vetor perpendicular à trajetória e que passa pelo robô, $\mathbf{l}$, pode ser obtido pela equação 7.16.

$$\mathbf{l} = (\mathbf{p_0} - \mathbf{x_r}) - ((\mathbf{p_0} - \mathbf{x_r})^T \cdot \mathbf{n}) \cdot \mathbf{n} \tag{7.16}$$

7.3 Demonstração e discussão de resultados

Os ensaios realizados pretendem avaliar o desempenho entre ambos os controladores projetados e em comparação com o controlador inicial, já existente no robô. Para tal, considerou-se uma trajetória em pulso com 3 m de comprimento para cada segmento de reta, tal como descrito na figura 7.9 e fez-se variar o ângulo de referência de uma forma suave ao longo de toda a trajetória, tal como mencionado na tabela 7.2.

7.3.1 Comparação entre o controlador *go to position* e o *follow line*

A figura 7.12 ilustra o comportamento do robô para o seguimento de uma trajetória com uma variação suave do ângulo de referência, no simulador.

De forma geral, verifica-se que ambos os controladores apresentam um resultado idêntico para o seguimento da trajetória em posição. Quando se procura uma validação semelhante para o seguimento da trajetória angular, verifica-se, na mesma, que ambos têm

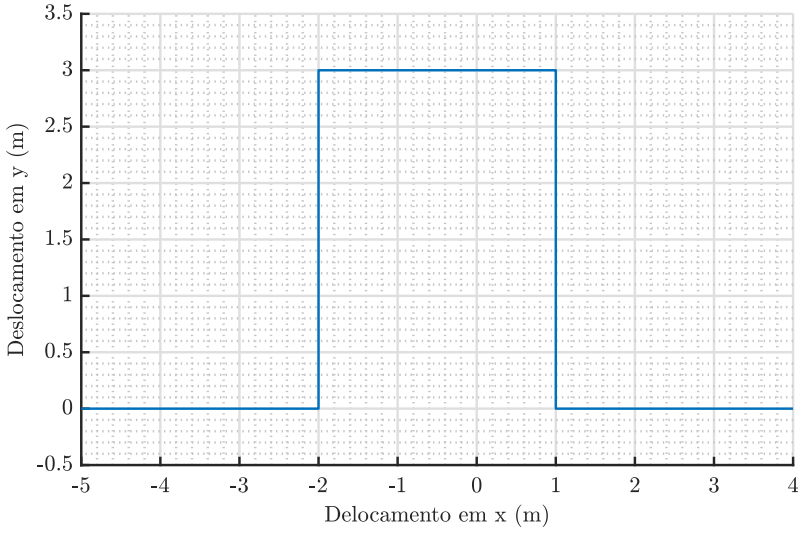


Figura 7.9: Trajetória de referência para o ensaio dos controladores

canto ou extremidade	x_{ref} (m)	y_{ref} (m)	θ_{ref} (rad)
0	-5	0	0
1	-2	0	$\frac{\pi}{2}$
2	-2	3	0
3	1	3	$-\frac{\pi}{2}$
5	1	0	0
6	4	0	0

Tabela 7.2: Trajetória de referência para o ângulo

um comportamento similar em termos de erro de seguimento, havendo uma maior vantagem na utilização do controlador *follow line* que aparenta ter um comportamento mais uniforme.

Assim, para validar esta conclusões, pode-se analisar as figuras 7.10 e 7.11, onde se verifica que há uma ligeira vantagem na utilização do controlador *follow line* para velocidades superiores a 1,5 m/s. No entanto,esta conclusão é duvidosa, dada a elevada oscilação que se observa no gráfico do seguimento da trajetória a 2 m/s da figura 7.12.

Portanto, pode-se concluir que ambos os controladores são idênticos e têm uma reposta razoável para velocidades iguais ou inferiores a 1,5 m/s, para velocidades superiores a utilização de qualquer um deles pode apresentar uma resposta instável.

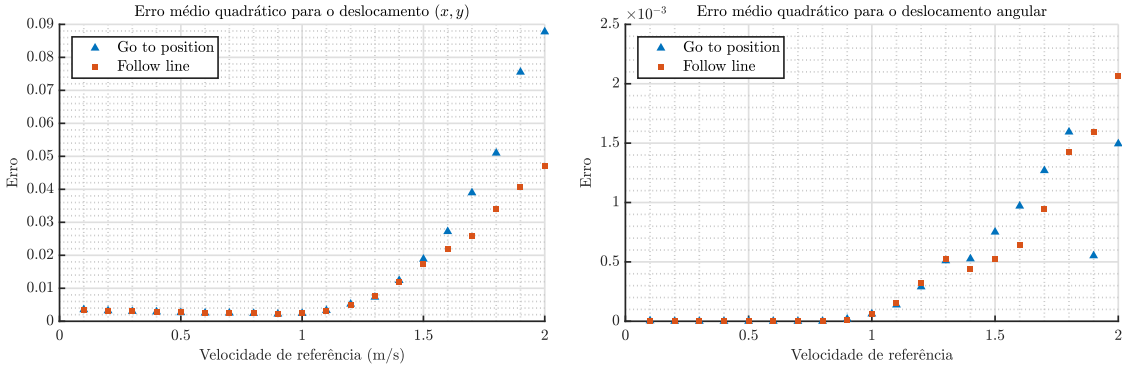


Figura 7.10: Erro de seguimento da trajetória de referência sem variação do ângulo para validar a utilização do controlador *go to position* ou *follow line*

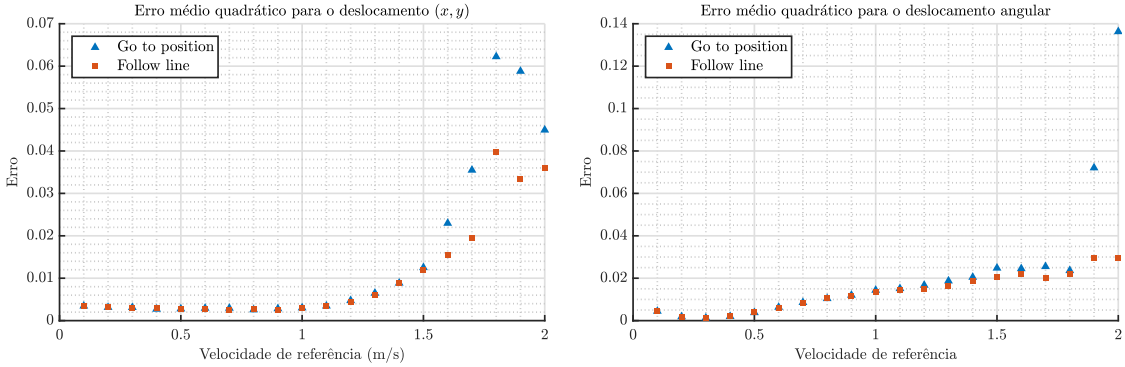


Figura 7.11: Erro de seguimento da trajetória de referência com variação do ângulo para validar a utilização do controlador *go to position* ou *follow line*

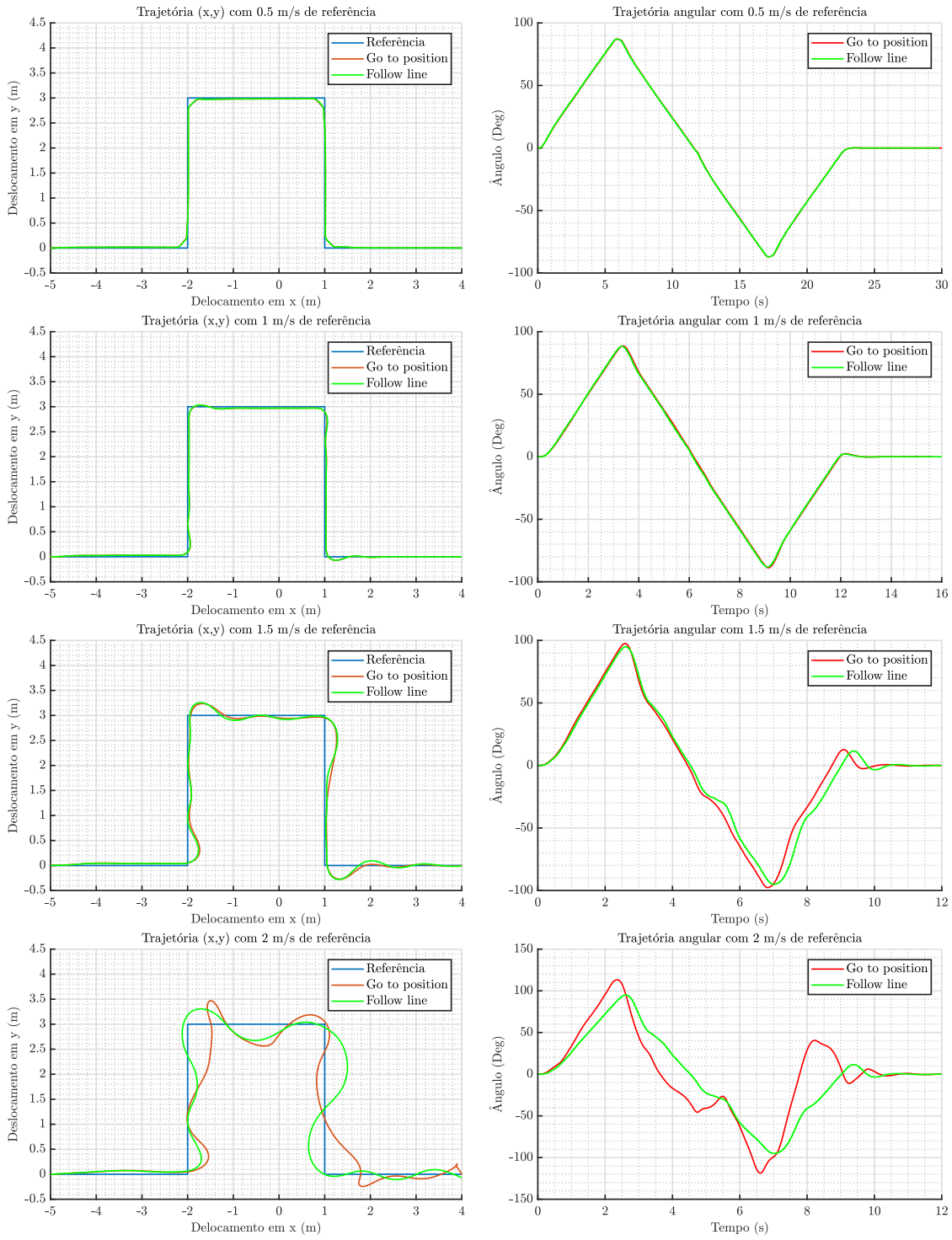


Figura 7.12: Resultados dos ensaios realizados para o seguimento da trajetória pulso da figura 7.9 com variação do ângulo tal como exposto na tabela 7.2, para a comparação entre os controladores *go to position* e *follow line*

7.3.2 Comparação do controlador *follow line* com o *go to position* já implementado

Embora com pouca vantagem, da secção anterior concluí-se que o *follow line* aparenta ter melhores resultado que o controlador *go to position*. Por isso, adicionalmente, acha-se conveniente fazer um estudo semelhante entre o primeiro controlador e o controlador *go to position* atualmente utilizado, cujos resultados estão representados nos gráficos da figura 7.15, considerando a trajetória da figura 7.9 com uma variação suave do ângulo tal como descrito na tabela 7.2.

Destes resultados, concluí-se que ambos os controladores têm um comportamento semelhante e, como tal, as conclusões são idênticas às da secção anterior, isto é, o erro entre eles é equivalente, com uma ligeira vantagem do controlador *follow line*, tal como se comprova nas figuras 7.13 e 7.14. Para a trajetória angular, verifica-se que é vantajoso manter-se o controlador *go to position* atualmente utilizado pela equipa, que aparenta ter um comportamento mais uniforme e com menor *overshoot* nos ângulos limite ($+90^o$ e -90^o).

Para velocidades superiores a 1,5 m/s, por inspeção da figura 7.15, conclui-se que o *follow line* tem melhores resultados, uma vez que apresenta um *overshoot* menor. No entanto, a utilização deste tipo de controlador a estas velocidades ainda continua a ser desaconselhado, dada a elevada oscilação da trajetória executada pelo robô, que pode revelar a existência de instabilidade no controlo do sistema.

Portanto, uma vez que no caso em que estamos na perseguição de um determinado alvo apenas é relevante o ângulo do robô na aproximação ao ponto final, pode-se deduzir que haverá uma ligeira vantagem na utilização do controlador *follow line*, sendo ainda possível ajustar-se o controlador que calcula a velocidade angular do robô, procurando-se obter um melhor resultado.

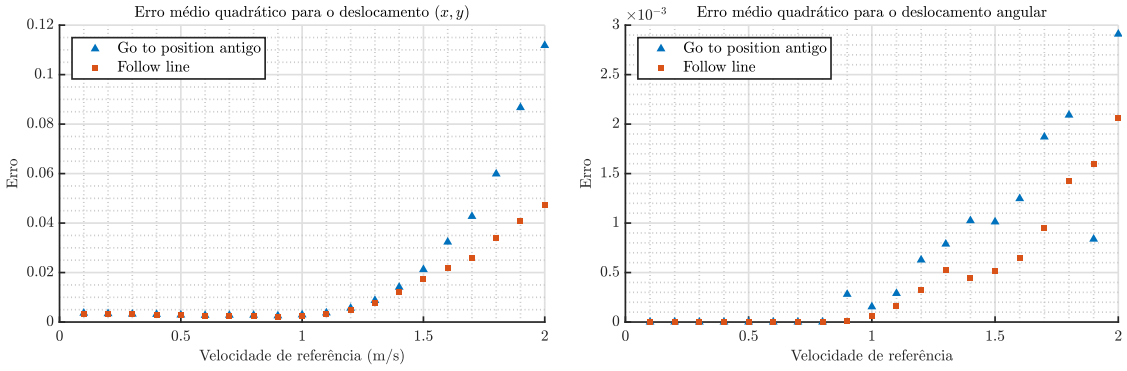


Figura 7.13: Erro de seguimento da trajetória de referência sem variação do ângulo para validar a utilização do controlador *go to position* ou *follow line*

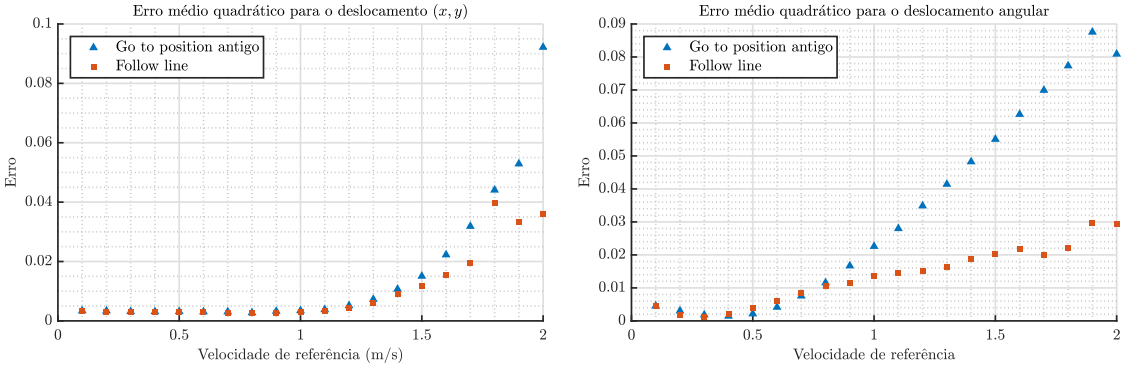


Figura 7.14: Erro de seguimento da trajetória de referência com variação do ângulo para validar a utilização do controlador *go to position* anteriormente implementado ou *follow line*

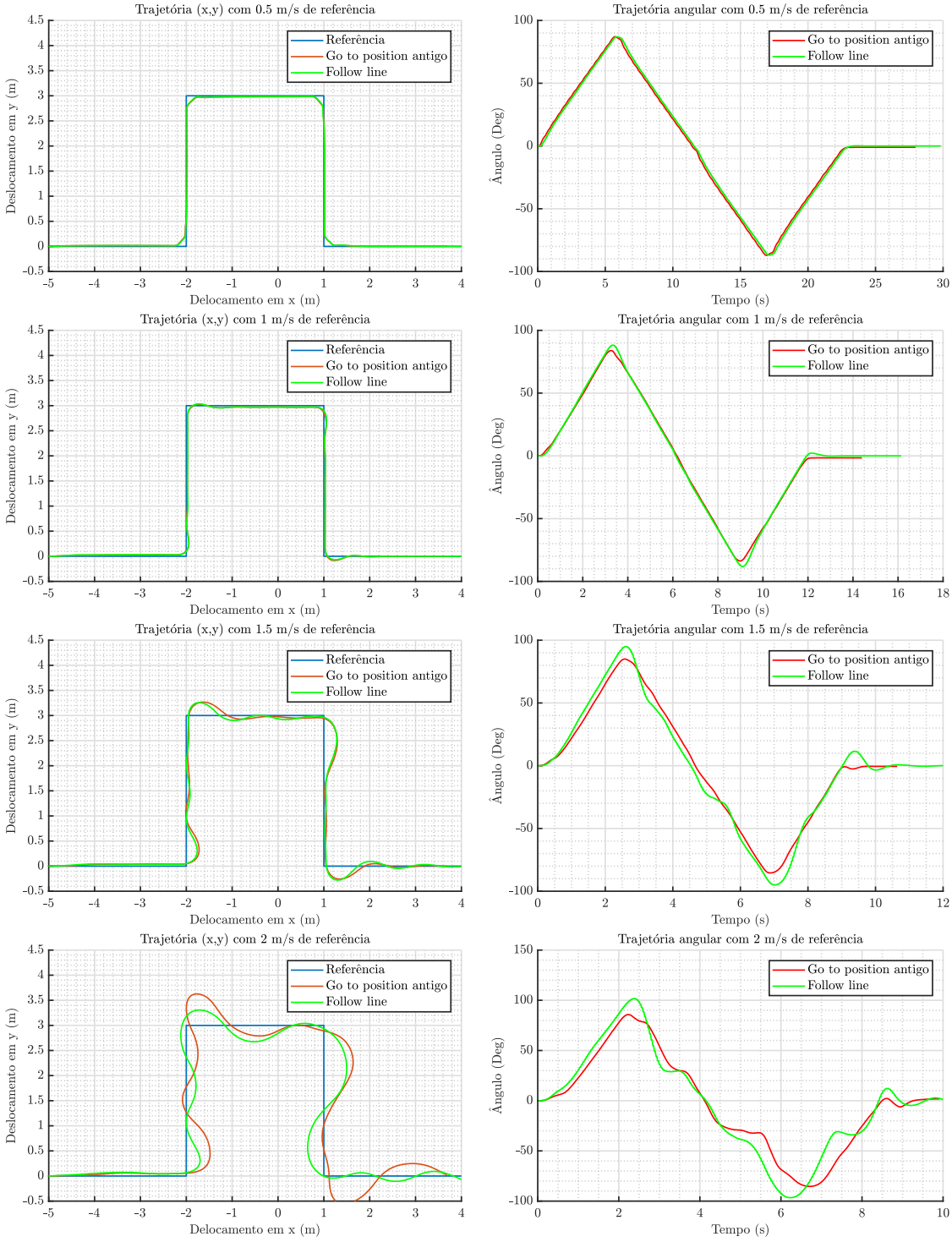


Figura 7.15: Resultados dos ensaios realizados para o seguimento da trajetória pulso da figura 7.9 com variação do ângulo tal como exposto na tabela 7.2, para a comparação entre os controladores *go to position* anteriormente implementado e *follow line*

7.3.3 Comparação entre o controlador *go to position* e *follow line* no robô

Pelas conclusões feitas nas duas secções anteriores, pode-se, desde já, desconsiderar a realização de ensaios no robô real, em laboratório, com velocidades superiores a 1,5 m/s, uma vez que em sistemas reais os resultados tendem a piorar.

Para a realização destes ensaios, devido a restrições de espaço, considerou-se uma segunda trajetória em pulso, idêntica à anteriormente utilizada, mas com segmentos de reta de 2 m, tal como ilustrado na figura 7.16. De forma similar aos ensaios anteriores, considerou-se uma variação suave da trajetória angular, tal como descrito na tabela 7.3.

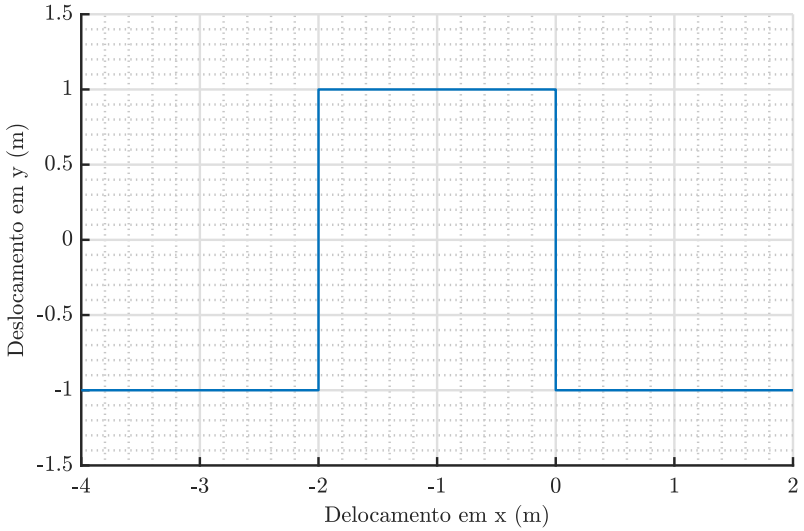


Figura 7.16: Trajetória de referência para o ensaio do controlador no robô

canto ou extremidade	x_{ref} (m)	y_{ref} (m)	θ_{ref} (rad)
0	-4	-1	0
1	-2	-1	$\frac{\pi}{2}$
2	-2	1	0
3	0	1	$-\frac{\pi}{2}$
5	0	-1	0
6	2	-1	0

Tabela 7.3: Trajetória angular de referência para trajetória da figura 7.16

As figuras 7.17 e 7.18 demonstram a prestação de ambos os controladores quando não é considerada a variação do ângulo de referência e quando é considerada, respetivamente. De ambas se tiram as mesmas ilações que as feitas até agora, isto é, que ambos os controladores têm um resultado semelhante, com um controlo suave no seguimento da trajetória de referência para velocidades até 1 m/s.

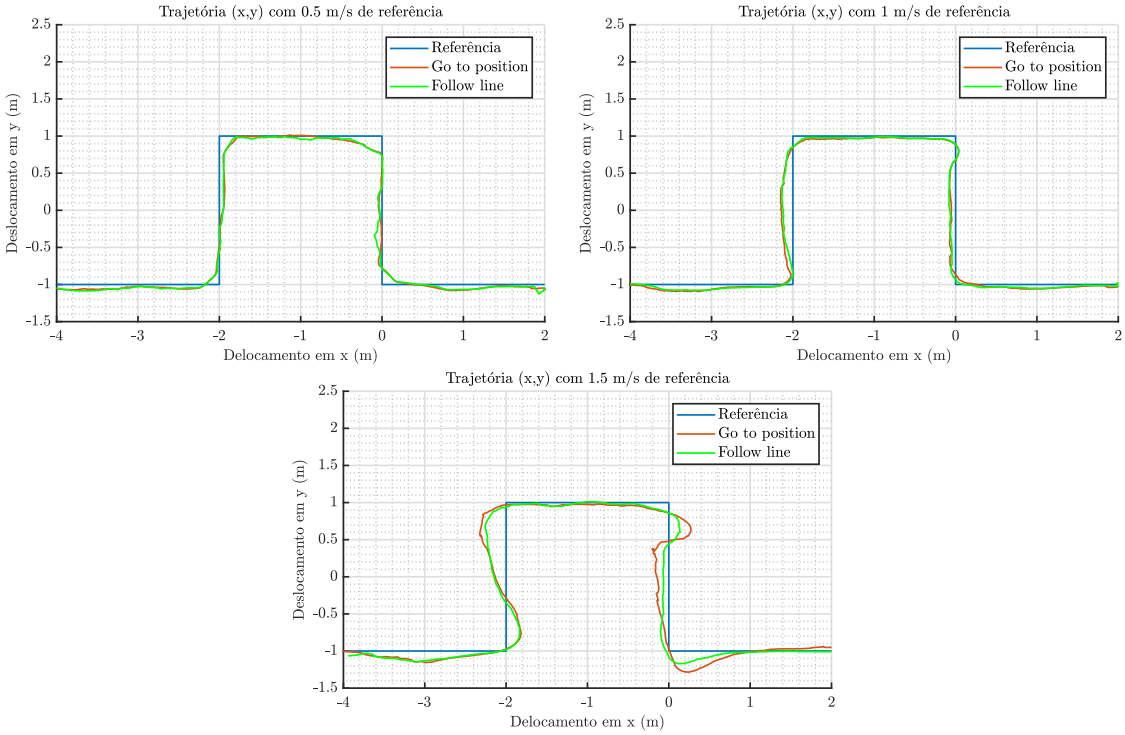


Figura 7.17: Resultados dos ensaios realizados para o seguimento da trajetória pulso da figura 7.9 com variação do ângulo tal como exposto na tabela 7.2, para a comparação entre os controladores *go to position* e *follow line*

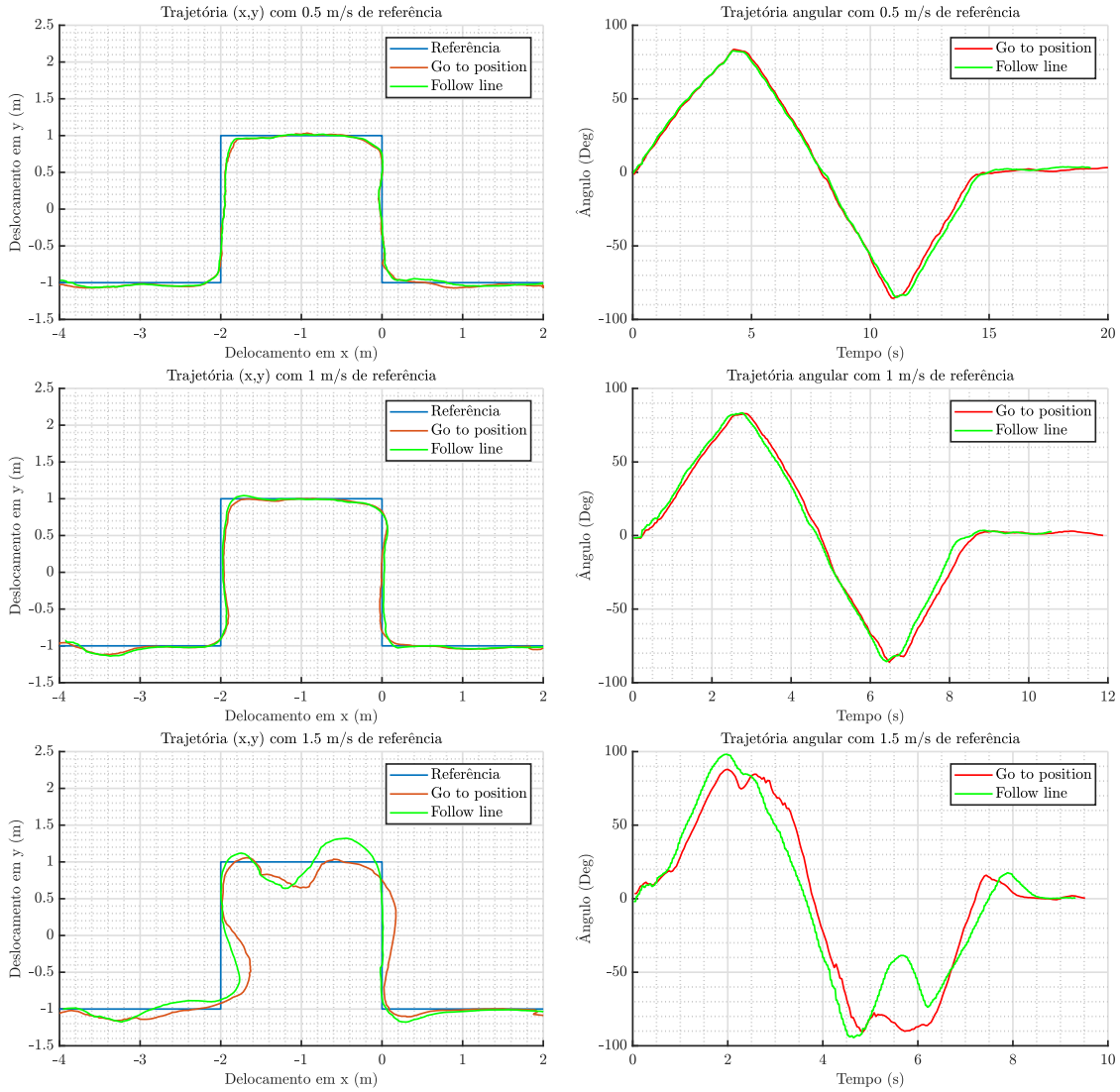


Figura 7.18: Resultados dos ensaios realizados para o seguimento da trajetória pulso da figura 7.9 com variação do ângulo tal como exposto na tabela 7.2, para a comparação entre os controladores *go to position* e *follow line*

Capítulo 8

Controlo Preditivo

De acordo com Camacho e Bordons [17], o modelo de controlo preditivo (MPC) não é nenhuma estratégia específica de controlo, mas antes um conjunto de métodos de controlo que são desenvolvidos em torno de um mesmo conjunto de ideias: o uso explícito de um modelo para prever a saída do processo nos instantes de tempo futuros; cálculo dos sinais de controlo que minimizam uma função custo específica; estratégia onde apenas o primeiro sinal de controlo calculado é aplicado sendo tudo recalculado no ciclo de controlo seguinte.

As várias estratégias MPC apenas variam entre si no que concerne ao modelo utilizado, para representar o processo a controlar e o ruído, e ao método de minimização da função objetivo.

O MPC apresenta um conjunto de vantagens sobre outros métodos, como por exemplo:

- os conceitos por detrás do controlador são relativamente simples e os parâmetros são fáceis de ajustar
- pode ser utilizado para controlar vários tipos de processos
- lida facilmente com os problemas multivariável
- possui intrinsecamente a compensação de atrasos
- introduz naturalmente um controlo *feedforward* para compensar as perturbações medíveis
- a extensão da estrutura de controlo para tratar as restrições é concetualmente simples e estas podem ser sistematicamente introduzidas ao longo do processo de projeto do controlador
- é muito vantajoso quando as referências futuras são conhecidas

Da mesma forma que este possui vantagens, também possui algumas desvantagens associadas:

- a derivação da lei de controlo é mais complexa do que nos controladores PID, podendo, quando a dinâmica do sistema é linear e não muda, ser derivada durante a fase de projeto
- no caso de controlo adaptativo, todos os cálculos têm de ser processados a cada ciclo de controlo, o que se torna num problema, em particular quando estamos perante sistemas não lineares que têm um curto limite de tempo para processamento
- o algoritmo de controlo depende de um conhecimento prévio do modelo a controlar, e este funciona tanto melhor, quanto mais preciso e completo for o modelo obtido

8.1 A estratégia MPC

Todos os controladores da família MPC são caracterizados pela estratégia de um horizonte de predição e um horizonte de controlo, como ilustra a figura 8.1.

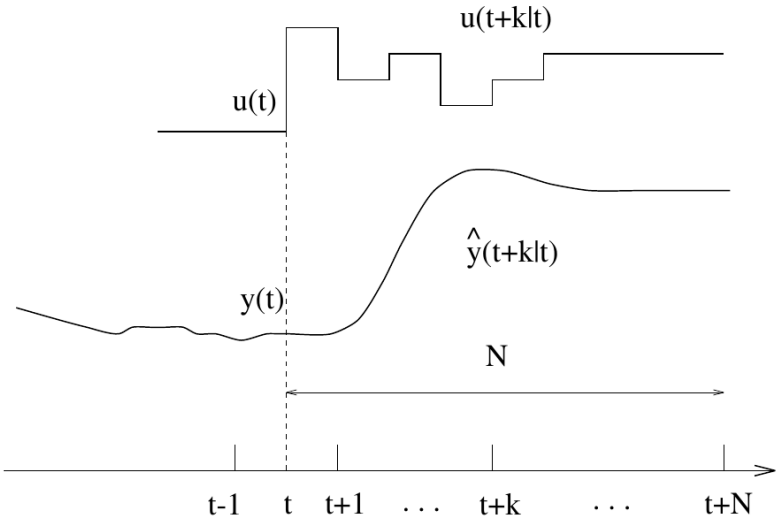


Figura 8.1: Horizonte recuado [17]

Portanto, as saídas futuras para um determinado horizonte N, que é designado como horizonte de predição, são previstas em cada instante t, recorrendo ao modelo do processo a controlar. Estas dependem essencialmente do estado atual do processo no instante t onde começa a simulação e dos sinais de controlo futuros que são enviados pelo sistema para serem testados no simulador. Este conjunto de sinais de controlo futuros são calculados por forma a minimizar um determinado critério, tipicamente quadrático, e tem como objetivo manter o processo o mais próximo possível da trajetória de referência. O sinal de controlo $u(t|t)$ é enviado para o processo, enquanto que os restantes sinais são rejeitados. A estratégia agora descrita, pode ser implementada através de um esquema de controlo básico como o da figura 8.2.

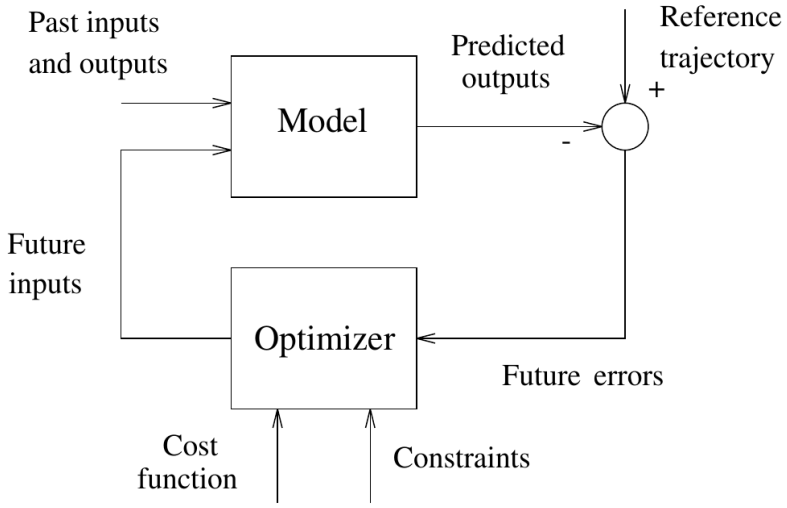


Figura 8.2: Estrutura básica do MPC [17]

Assim, tendo em consideração os pressupostos anteriores, optou-se por recorrer a um algoritmo de controlo NMPC, que pode ser representado pelo modelo da figura 8.3.

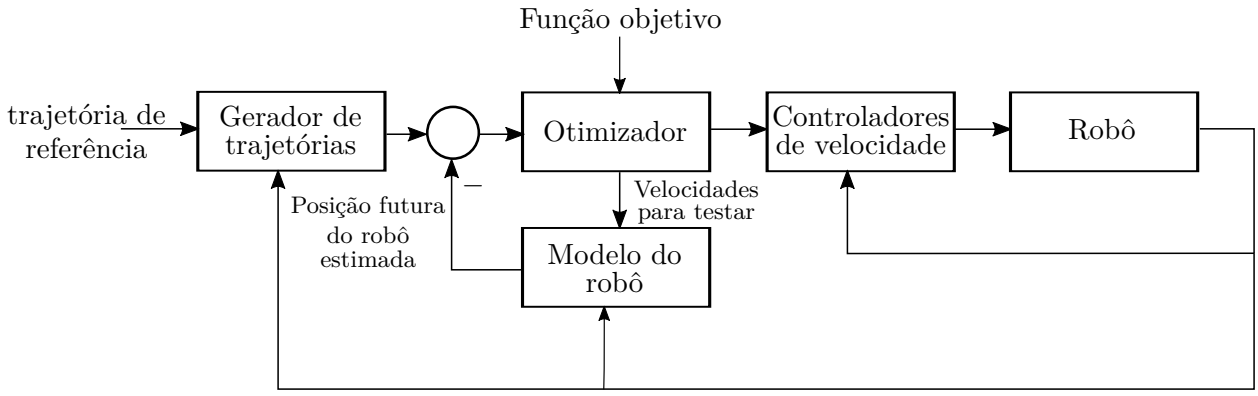


Figura 8.3: Estrutura de controlo NMPC utilizada

8.2 Elementos dos controladores MPC

Tal como já referido, todos os algoritmos de controlo baseados em MPC têm em comum três elementos principais, diferindo entre si das opções que são tomadas em cada um:

- Modelo de predição
- Função objetivo
- Método de obtenção da lei de controlo

8.2.1 Modelo de predição

No âmbito dos algoritmos MPC, o modelo de predição é o elemento principal para se obter uma boa lei de controlo. Este deve ser o mais completo possível, captando desta forma toda a dinâmica do processo, por forma a garantir uma solução o mais próximo possível da solução ótima.

Para se obter este modelo completo, o modelo de predição deve ser dividido em duas componentes: o modelo do processo e o modelo de perturbações. O modelo do processo é necessário para a predição da saída nos instantes futuros. Por outro lado, o modelo de perturbações procura descrever os comportamentos do processo que não são descritos pelo seu modelo, como é o caso das entrada não medíveis, do ruído e dos erros e aproximações do modelo.

No que se refere ao modelo de predição do robô, pretende-se um modelo simplificado do mesmo, para se conseguir uma rápida minimização da função objetivo. Por este motivo, considerou-se apenas o modelo de processo, ignorando a existência de perturbações no sistema. Portanto, começou-se por considerar apenas o modelo cinemático do robô, indo-se, iterativamente, adicionando novos elementos até se obter uma solução satisfatória.

8.2.1.1 Odometria

Tal como referido anteriormente, começou-se por aproximar o modelo do robô por um modelo puramente cinemático. Este é caracterizado pelas equações da odometria 5.8 e 5.9 da secção 5.1.3, sendo que d, d_n e ω podem ser obtidos pelo conjunto de equações 8.1, onde Δt é o tempo do ciclo da simulação, que no caso em estudo adotou-se ser 10 ms, minimizando, desta forma o erro por aproximações em tempo discreto.

$$\begin{aligned} d &= v \times \Delta t \\ d_n &= v_n \times \Delta t \\ \Delta\theta &= \omega \times \Delta t \end{aligned} \tag{8.1}$$

8.2.1.2 Limitação da velocidade dos motores

Uma das características não lineares do robô a controlar é a saturação dos motores. Desta forma, é importante caracterizar esta situação e estabelecer o que acontece quando ocorre este facto. Uma das estratégias que procura compensar esta situação é o escalonamento das velocidades dos motores para valores fora da zona de saturação, tal como explica Scolari [3] através do algoritmo IRV.

Este algoritmo estabelece que primeiramente são calculadas as velocidades máximas do robô, definindo-se desta forma planos de velocidades máximas (tal como se observa no exemplo da figura 2.4 para um robô omnidirecional de quatro rodas). De seguida, caso o conjunto dos valores seja maior que o sólido formado pelo conjunto dos planos de

velocidades máximas, é calculada a reta que o une o ponto caracterizado pelo conjunto das três velocidades e a origem e as novas velocidades de controlo do robô passam a ser aquelas que resultam da interseção da reta com o plano das velocidades máximas.

O IRV pode ser simplificado, se se considerar apenas as velocidades das rodas, estabelecendo-se qual a roda que está mais saturada. De seguida, calcula-se o coeficiente que escala esta velocidade para a velocidade máxima e aplica-se este coeficiente a todos os motores, tal como é ilustrado pelo algoritmo 1.

Algoritmo 1: Algoritmo IRV

Input: v_i é a velocidade atual da roda i.
Output: v_{out_i} é a velocidade da roda i fora da zona de saturação
Data: v_{limite} é a velocidade máxima à qual a roda consegue rodar

```
1  begin
2      v_máximo ← máximo(|v1|, |v2|, |v3|)
3      if v_máximo > v_limite then
4          σ = v_máximo / v_limite
5          v_out1 = v1 / σ
6          v_out2 = v2 / σ
7          v_out3 = v3 / σ
8      else
9          v_out1 = v1
10         v_out2 = v2
11         v_out3 = v3
```

As velocidades máximas das rodas do robô podem ser medidas experimentalmente, tal como se ilustra na figura 6.5 da secção 6.2.1, de onde se concluí que é igual a, aproximadamente, 64 rad/s.

8.2.1.3 Limitação da aceleração dos motores

Tal como explicado na secção 6.2.4, devido a problemas relacionados com a derrapagem das rodas durante as acelerações e as travagens, sentiu-se a necessidade de limitar a aceleração do motores.

Da figura 8.4, verifica-se que a limitação da aceleração do robô prevalece sobre o seu modelo aproximado de primeira ordem, como tal, pode-se afirmar que o modelo dinâmico do robô pode ser aproximado pela sua curva de saturação da aceleração.

Desta forma, a saturação da aceleração pode ser implementada através de uma função de limitação, que é matematicamente descrita pela equação 8.2.

$$F(x, x_{máximo}) = \begin{cases} x_{máximo} & \text{se } x > x_{máximo} \\ x & \text{se } x < x_{máximo} \ \wedge \ x > -x_{máximo} \\ -x_{máximo} & \text{se } x < -x_{máximo} \end{cases} \tag{8.2}$$

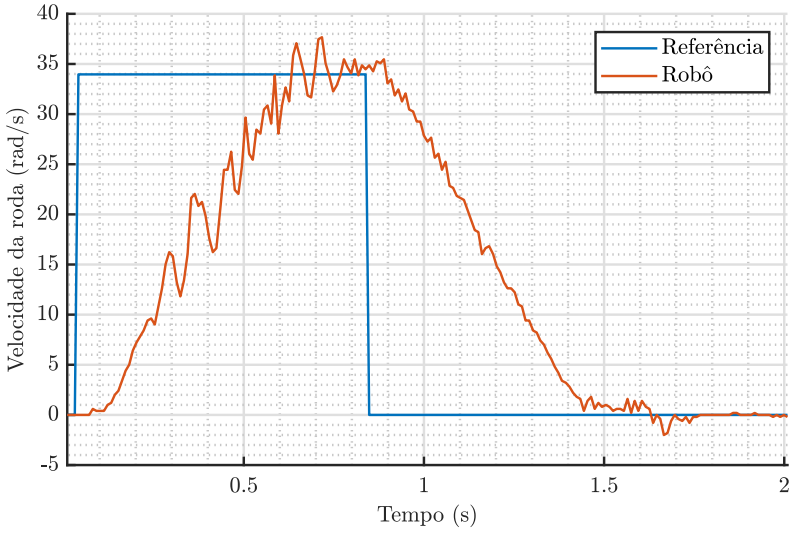


Figura 8.4: Curva de excitação de um motor do robô com um referência em pulso

A aceleração das rodas é, por sua vez, obtida através da velocidade das mesmas, tal como ilustra a equação 8.3. Por forma a minimizar o erro das aproximações feitas durante a simulação, em vez de se considerar o tempo de ciclo do programa para a simulação do sistema, faz-se, para cada ponto do horizonte de predição a estimar, quatro simulações, o que resulta num tempo de simulação, Δt, de 10 ms.

$$a = \frac{v_{atual} - v_{anterior}}{\Delta t} \tag{8.3}$$

Este limite de aceleração das rodas foi anteriormente medido na secção 6.2.3, onde se constatou que era de 61,09 rad/s^2.

8.2.2 Função objetivo

Diferentes funções objetivo podem ser especificadas, no entanto, é comum recorrer-se a um problema quadrático para minimizar o erro futuro. O principal objetivo desta função é penalizar o erro, por forma a garantir um bom seguimento de uma determinada referência. Adicionalmente, também é comum penalizar-se o esforço de controlo Δu, procurando-se, desta forma, um controlo suave do processo e ao minimizar-se as variações e não o valor absoluto, evitam-se erros em regime permanente. Esta função objetivo pode ser descrita por 8.4.

$$J = Y^T M Y + \Delta U^T \Lambda \Delta U \tag{8.4}$$

Numa primeira fase, optou-se por não utilizar a componente referente ao esforço de controlo, pois esta dificulta o processo de otimização e não costuma variar muito. Portanto, a função objetivo 8.4 pode ser substituída pela função objetivo 8.5, onde λ_i é um coeficiente

de multiplicação que procura pesar cada componente da função, $\hat{x}(t+i|t)$, $\hat{y}(t+i|t)$ e $\hat{\theta}(t+i|t)$, ou seja, a posição e o ângulo previstos do robô no instante $t+i$ quando este está no momento t e $x(t+i)$, $y(t+i)$ e $\theta(t+i)$ são a posição e o ângulo do robô da trajetória de referência no instante $t+i$.

$$\begin{aligned} J(N_1,N_2) = \lambda_1 \sum_{i=N_1}^{N_2} (\hat{x}(t+i|t) - x(t+i))^2 + \\ + \lambda_1 \sum_{i=N_1}^{N_2} (\hat{y}(t+i|t) - y(t+i))^2 + \\ + \lambda_2 \sum_{i=N_1}^{N_2} (\hat{\theta}(t+i|t) - \theta(t+i))^2 \end{aligned} \tag{8.5}$$

8.2.3 Método de obtenção da lei de controlo

Dado que o processo a controlar é um problema multivariável, a lei de controlo a calcular tem como objetivo encontrar os melhores valores de controlo $U(\cdots,k)$, que no caso do robô omnidireccional corresponde à sua velocidade segundo as componentes v, v_n, ω, que melhor minimizam a função objetivo da equação 8.5.

Em condições onde o modelo de predição é linear e não existem restrições, esta minimização pode ser feita por métodos analíticos. No entanto, quando alguma destas condições não é cumprida, é necessário recorrer-se a métodos numéricos que procuram iterativamente a solução ótima.

No caso em estudo, o modelo de predição é do tipo não linear e existem algumas restrições, como é o caso da saturação da velocidade das rodas. Portanto, é necessário recorrer-se a método numéricos que procuram a solução do problema. Para efeitos de estudo foram considerados dois algoritmos de minimização: o gradiente descendente e o *resilient propagation*, descritos no anexo A.

Assim, o algoritmo que permite otimizar a lei de controlo $U(\cdots,k) = [v(k)\ v_n(k)\ \omega(k)]^T$ no período de amostragem k pode ser descrito pelo algoritmo 2.

8.3 Geração da trajetória de referência

Da figura 8.3, verifica-se que uma das primeiras etapas do controlador é gerar uma nova trajetória de referência, através da trajetória recebida. Isto provém do facto de que a trajetória recebida pelo controlador para ser seguida pelo robô, W, corresponde a um conjunto ordenado de pontos no sistema de coordenadas global, sem qualquer compromisso temporal ou de igual espaçamento entre estes, tal como ilustra a figura 8.5.

Uma vez que a função objetivo necessita da posição de referência do robô em cada instante de tempo para conseguir calcular a lei de controlo ótima, é necessário gerar-se uma nova trajetória de referência, $T_{ref}(\cdots,k) = [x_{ref}(k)\ y_{ref}(k)\ \theta_{ref}(k)]^T$, ao longo de

Algoritmo 2: Lei de controlo

Input: *Waypoints* de referência, W, e velocidade linear de referência, v_{ref}
Output: Sinais de controlo otimizados, **u**
Data: ε (erro máximo considerado para a função objetivo J)

```
1 begin
2     Cálculo da trajetória de referência, T(···, k+j), com j = 1..N_p, a partir de
        W(···, k)
3     Inicialização de U(···, k) com as velocidades atuais do robô
4     while J < ε ou número máximo de iterações atingido do
5         Predição do estado futuro do robô por simulação, recorrendo ao modelo do
            robô e a T(k)
6         Cálculo da função objetivo, J
7         Otimização de U
8     Retorna U(···, k)
```

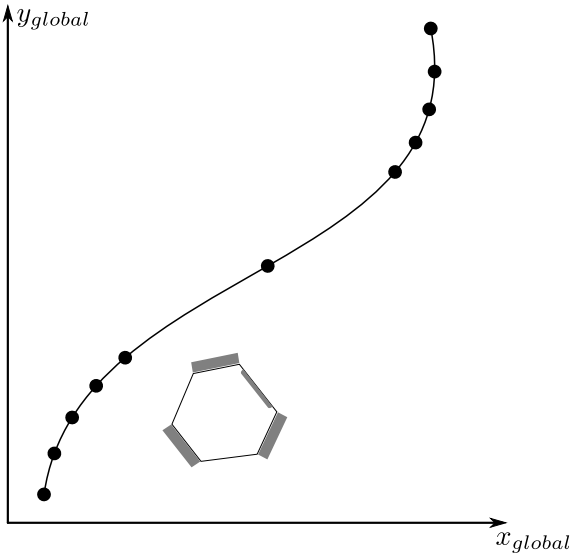


Figura 8.5: Conjunto de *waypoints* para geração da trajetória de referência

todo o horizonte de predição, $k \in [0, N_p]$, que respeite a condição temporal de execução da trajetória, tendo por base a velocidade de referência desejada.

8.3.1 Cálculo do espaçamento entre os pontos da trajetória

Tal como referido na secção 8.2.1.3, sabe-se à partida que o robô tem a sua aceleração fortemente limitada. Desta forma, sabe-se à partida que nem sempre o robô será capaz de cumprir os objetivos propostos para o seguimento da trajetória, portanto, propõe-se a adição de uma restrição no limite de aceleração do robô para a geração da nova trajetória de referência que contempla restrições temporais.

Assim, o cálculo do espaçamento entre os pontos da trajetória de referência fica dividido em duas etapas: limitação da aceleração e cálculo da distância entre pontos. A distância entre pontos será obtida pela equação 8.6, onde v é a velocidade de referência depois de limitada. Considera-se que é estimado uma posição futura do robô para cada iteração do ciclo do programa tendo em consideração o horizonte de predição pretendido, como tal, neste caso, Δt é igual 40 ms.

$$distancia = |v| \times \Delta t \tag{8.6}$$

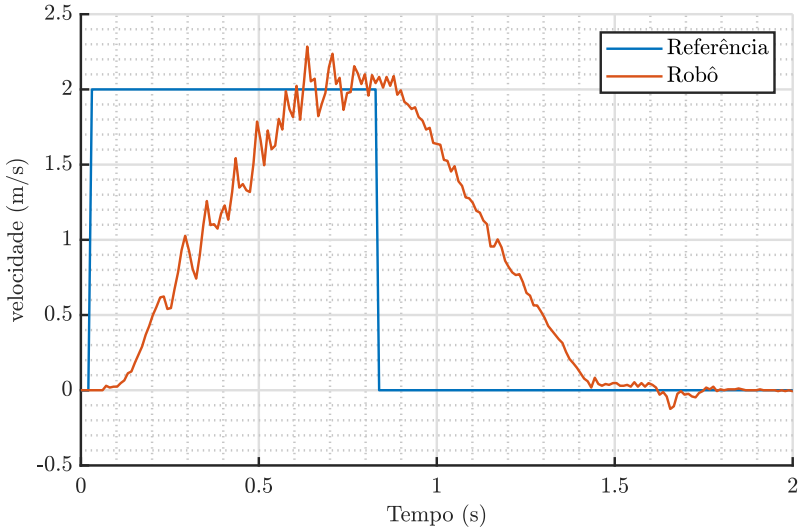


Figura 8.6: Curva de aceleração do robô com um movimento linear

O limite de aceleração linear pode ser obtido através das equações da cinemática inversa (5.7). Assim, considerando a aceleração máxima das rodas igual a 61,09 rad/s^2 (tal como calculado na secção 8.2.1.3) e multiplicando pelo raio das rodas, igual a 0,0513 m (tal mo mencionado na tabela 6.2), tem-se que a aceleração máxima de uma roda, também, é igual a, aproximadamente, 3,13 m/s^2. Para se obter a aceleração máxima linear, considera-se que as rodas da frente estão a rodar à velocidade máxima, ou seja, $v_1 = 3,13$ m/s e $v_2 = -3,13$ m/s (v_1 e v_2 estão descritos na figura 5.1) e aplicando as equações da cinemática inversa, tem-se que a aceleração máxima linear do robô é igual a 3,61 m/s.

8.3.2 Geração da trajetória linear

Para a geração da nova trajetória de referência, T, considera-se que o posição $T(0)$ tem início no ponto da trajetória W mais perto do robô. Este corresponde ao ponto da interseção da reta perpendicular à trajetória W, que passa pelo robô, com a própria trajetória W, cuja distância W–robô é mais curta, tal como exemplifica a figura 8.7. A obtenção deste ponto, pode ser feita tal como se explica na secção 7.2.2 para o algoritmo *follow line*.

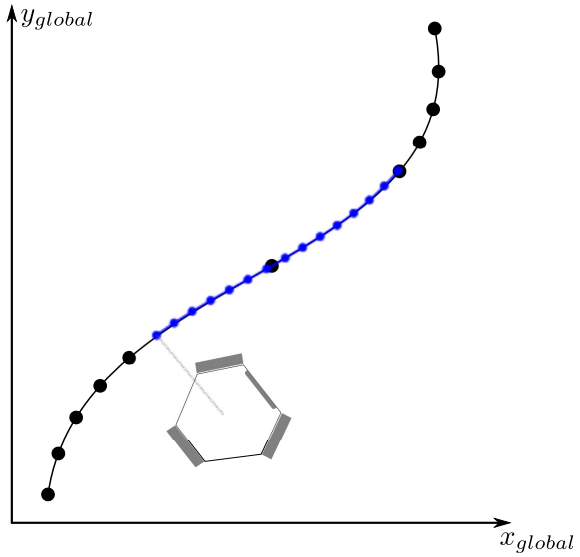


Figura 8.7: Nova trajetória gerada a partir da trajetória inicial (preto – trajetória W, azul – trajetória T)

Os restantes pontos da trajetória são obtidos sequencialmente, a partir do ponto inicial, com um espaçamento fixo, obtido de 8.6, e acompanhando sempre a trajetória gerada pelos *waypoints*, W. Portanto, o ponto $T(K)$ é gerado pelo conjunto de equações 8.7 e 8.8.

$$\mathbf{p} = \frac{W(\cdots, i+1) - W(\cdots, i)}{||W(\cdots, i+1) - W(\cdots, i)||} \tag{8.7}$$

$$T(\cdots, k) = T(\cdots, k-1) + dist\hat{a}ncia \times \mathbf{p} \tag{8.8}$$

Na equação 8.7, $\mathbf{p}$ é um versor de dimensão 2×1 que indica a direção e sentido da trajetória entre o atual *waypoint* e o *waypoint* seguinte. Portanto, da equação 8.8 resulta um vetor $T(\cdots, k)$, também de dimensão 2×1, que corresponde aos pontos x e y da trajetória gerada, no sistema de coordenadas global.

Uma vez, que o controlador efetua a otimização da lei de controlo sobre um horizonte de predição, apenas serão gerados tantos pontos quanto o tamanho do horizonte de predição, N_p, como se observa na figura 8.7.

8.3.3 Geração da trajetória angular

Para a geração da componente angular da nova trajetória, considera-se que o ponto inicial $T(\cdots, 0)$ corresponde ao ângulo atual do robô.

Uma vez que não se considera nenhuma velocidade angular de referência, é necessário obter-se este valor por métodos indiretos por forma a garantir que quando o robô atinge o *waypoint* seguinte na sua componente linear, também o consegue na sua componente

angular. Portanto, a velocidade angular que garante a restrição imposta pode ser obtida pela equação 8.10, que resulta, posteriormente num incremento do ângulo obtido de 8.11.

$$\Delta T = \frac{\delta}{v} \tag{8.9}$$

$$\omega = \frac{W_3(\cdots, k+1) - \theta_{inicial}}{\Delta T} \tag{8.10}$$

$$\theta_{inc} = \omega \times \Delta t \tag{8.11}$$

Da equação 8.9, entende-se ΔT como o tempo restante até o robô atingir o próximo *waypoint*, que depende de δ como a distância restante até o mesmo e v como a velocidade linear de referência para a execução da trajetória. Da equação 8.10, tem-se que W_3 corresponde à terceira componente do vetor W, que é o ângulo desejado quando o robô atinge o *waypoint* k e tem-se que $\theta_{inicial}$ refere-se ao ângulo inicial do segmento considerado que pode ser o ângulo atual do robô ou o ângulo de referência de um *waypoint*.

Portanto, de forma similar com a obtenção dos pontos da trajetória de referência linear, para se obter os diferentes pontos para a trajetória de referência angular, com tamanho igual ao horizonte de predição, recorre-se à equação 8.12.

$$T_3(\cdots, k) = T_3(\cdots, k-1) + \theta_{inc} \tag{8.12}$$

8.4 Ajuste dos parâmetros do controlador

Apesar de ser relativamente simples de se implementar, os algoritmos NMPC têm alguns parâmetros que necessitem de ser bem ajustados para se conseguir garantir a máxima rapidez do algoritmo, nomeadamente quando existem elevadas restrições em termos temporais e se está perante ambientes com elevada dinâmica.

8.4.1 Ajuste do otimizador

Um dos componentes que é necessário começar-se por ajustar é o algoritmo otimizador da lei de controlo. Tal como já referido, foram estudados dois algoritmos, descritos no anexo A, que têm por base a análise da evolução da derivada de primeira ordem da função objetivo.

Começando pelo algoritmo de gradiente descendente, através de alguns ensaios verifica-se que o coeficiente multiplicativo, α, que melhor se ajusta ao problema, corresponde a um valor igual a 1. Desta forma, consegue-se obter uma otimização relativamente rápida, sem que o passo de atualização do sinal de controlo seja demasiado grande que ultrapasse o mínimo ou muito pequeno que faça o algoritmo demorar muito tempo a convergir. Um outro parâmetro que é possível ajustar é o tamanho do passo do sinal de controlo em cada direção, para que o algoritmo possa testar os diferentes sentidos e procurar o de maior

crescimento, seguindo depois no sentido inverso a este. Procedendo de forma similar ao parâmetro anterior, conclui-se que o valor que melhor se ajusta é um passo de 0,01 m/s (ou rad/s, dado que este é um problema multivariável, onde se pretende determinar o melhor valor de v, v_n, ω).

Por um procedimento semelhante, é possível determinar quais os parâmetros que melhor favorecem o algoritmo RPROP. Assim, o coeficiente que determina o crescimento da lei de controlo quando este tende para o mínimo local, η^+, é bem caracterizado por um valor igual a 1,2. Quando o algoritmo passou o valor ótimo, é necessário diminuir o tamanho do passo, por forma a fazê-lo regressar e convergir novamente para o mínimo; o valor que melhor favorece este coeficiente é de η^- igual a $0,1$. Tal como referido no anexo A, este algoritmo apenas usa o sinal da derivada de primeira ordem da função objetivo, para definir o sentido para o qual deve evoluir o sinal de controlo. Portanto, é ainda importante selecionar o tamanho de passo inicial, $\Delta_{ij}^{(0)}$, que se arbitrou como 0,1, garantindo desta forma que na primeira simulação, o sinal de controlo não evolui demasiado o que obrigaria o algoritmo a recuar.

Uma vez ajustado todos os parâmetros, realizaram-se duas otimizações do sinal de controlo, uma para cada algoritmo, a partir do repouso e do ponto inicial de uma trajetória igual para ambos. A figura 8.8 representa a evolução da função objetivo, onde se observa que o RPROP consegue convergir mais rapidamente que o gradiente descendente. Concluí-se, portanto, que este é mais indicado para o problema de controlo de trajetórias quando se tem um período de tempo restrito.

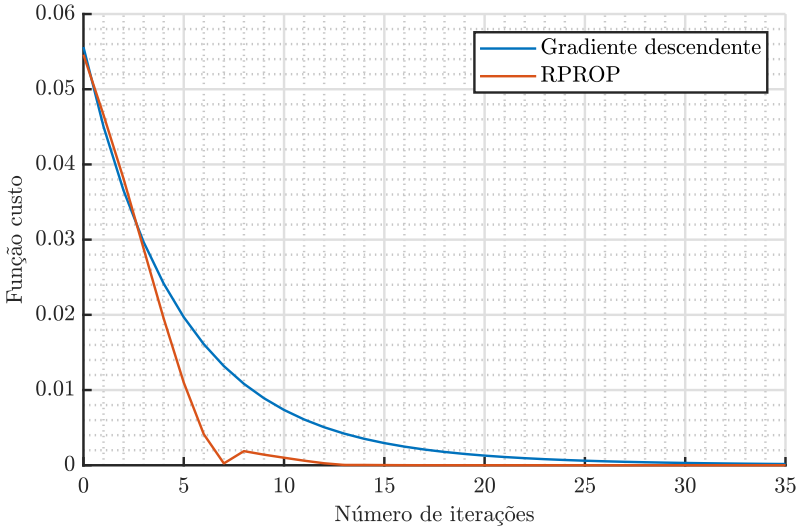


Figura 8.8: Evolução da função objetivo

Tal como referido no anexo A, em ambos os algoritmos utilizados geralmente não se consegue atingir a solução num número limitado de iterações. No entanto, é possível obter-se uma boa aproximação deste valor, o que para o caso do controlo de trajetórias é

suficiente. Por consequência, é necessário definir-se uma condição de paragem, ε, para que quando este pare quando se aproxima de uma solução se considera boa o suficiente. Por inspeção visual do gráfico da figura 8.8 e através de alguns ensaios realizados no simulador, conclui-se que a solução é relativamente boa quando o erro da função objetivo é menor que 0,0002. Neste momento, a evolução da função objetivo já começa a tender ser horizontal, o que significa que não existe uma grande variação no sinal de controlo e como tal este está próximo da solução ótima.

Adicionalmente, tal como já referido, o controlo de robôs está intrinsecamente ligado a restrições temporais por ciclo de programa. Neste caso, este ciclo é determinado pela frequência de aquisição da imagem da câmara de 25 fps, que despoleta o ciclo do programa. Assim, o programa tem um tempo de controlo máximo de 40 ms. Dado que neste período tempo é, ainda, necessário realizar outras tarefas, como executar o algoritmo de localização, processar os dados da câmara, pode-se arbitrar que o controlador tem, aproximadamente, metade deste período para encontrar uma solução. Portanto, além da condição de paragem anterior, é ainda necessário implementar-se uma nova que limita o número máximo de iterações do otimizador, que se concluiu ser de, aproximadamente, 20 iterações.

8.4.2 Ajuste dos parâmetros do controlador

Além da definição das taxas de evolução do algoritmo de otimização numérica, é ainda necessário ajustar-se alguns parâmetros relativos ao próprio controlador, nomeadamente, os coeficientes da função objetivo 8.5, o tamanho do horizonte de predição (N_p) e o tamanho do horizonte de controlo (N_u).

De acordo com Findeisen e Algöwer [28], idealmente seria vantajosa a utilização da formulação do NMPC de horizonte infinito, uma vez que, no caso nominal, a trajetória de malhas fechada coincide com a prevista em malha aberta. A grande dificuldade é que os esquemas de horizonte infinitos não podem, frequentemente, ser aplicados na prática com um tempo de otimização da lei de controlo suficientemente rápido.

Da formulação do MPC, sabe-se que $0 < N_u \leq N_p$, portanto e uma vez que o horizonte de controlo é aquele que mais exige do algoritmo de otimização, começou-se por estudar o comportamento deste, procurando otimizar o seu tamanho, sem comprometer a execução dos restante algoritmos do programa. Após alguns ensaios, constata-se que a utilização de horizontes de controlo superiores a 1 comprometem a execução programa, tendo um tempo de otimização muito próximo do tempo de ciclo do mesmo 40 ms.

A utilização do horizonte de predição está intrinsecamente relacionado com o regime transitório do controlador. Desta forma, a utilização de horizontes de predição muito curtos pode resultar numa má estimação da lei de controlo, obtendo-se um sistema instável. Por outro lado, a utilização de horizontes de predição muito longos resulta na otimização da lei de controlo considerando o regime permanente, o que adiciona ruído ao algoritmo de otimização, aumentando o seu tempo de processamento, sem se obter grandes vantagens

e, por ventura, até mesmo desvantagens. Dadas estas condições, realizaram-se diferentes ensaios considerando diferentes horizontes de predição, concluindo-se os resultados da tabela 8.1, que são semelhantes entre os diferentes algoritmos. Embora já ultrapasse o limite temporal inicialmente definido de 20 ms, verifica-se que a utilização de um horizonte de predição de 10 pontos é vantajoso para o algoritmo, uma vez que se aproxima o suficiente do regime permanente, sem que haja prejuízo na execução do restante programa.

N_p	$T_{máximo}$ (ms)	$T_{médio}$ (ms)
8	24	20,95
9	25	22,53
10	28	24,71
11	31	26,11
12	35	28,67

Tabela 8.1: Tempo de execução do programa em relação ao horizonte de predição considerado

Relativamente aos coeficientes da função objetivo, estes foram determinados experimentalmente. Após alguns ensaios, concluiu-se que os parâmetros que melhor favorecem o controlo em questão é um coeficiente $\lambda_1 = 4$, para o erro da distância à trajetória, e $\lambda_2 = 1$, para o erro angular.

8.5 Demonstração e discussão de resultados

Os ensaios realizados pretendem avaliar o desempenho do controlador desenvolvido. Para se conseguir proceder corretamente estes testes, escolheu-se uma trajetória em pulso com 3 m de comprimento em cada segmento de reta, tal como representado na figura 8.9. Esta trajetória representa uma excitação moderada e é longa o suficiente para se obter uma variação angular suave, mas, ao mesmo tempo, curta por forma a garantir que a existência de variação angular possa representar uma dificuldade acrescida para o controlo de trajetórias, permitindo desta forma validar o desempenho do controlador entre quando existe variação do ângulo e quando não este é constante.

A variação do ângulo ao longo da trajetória de referência é descrita, em cada canto e extremidade da mesma, pela tabela 8.2.

8.5.1 Análise ao algoritmo de otimização

No âmbito do controlador desenvolvido, foram testados dois algoritmos diferentes, para se compreender qual é que convergia mais rapidamente para a solução ou ficava mais próxima desta (para o caso em que se excedia o número máximo de iterações permitidas). Da primeiro teste feito na secção 8.4.1, concluí-se que o RPROP tem um melhor desempenho que o gradiente descendente.

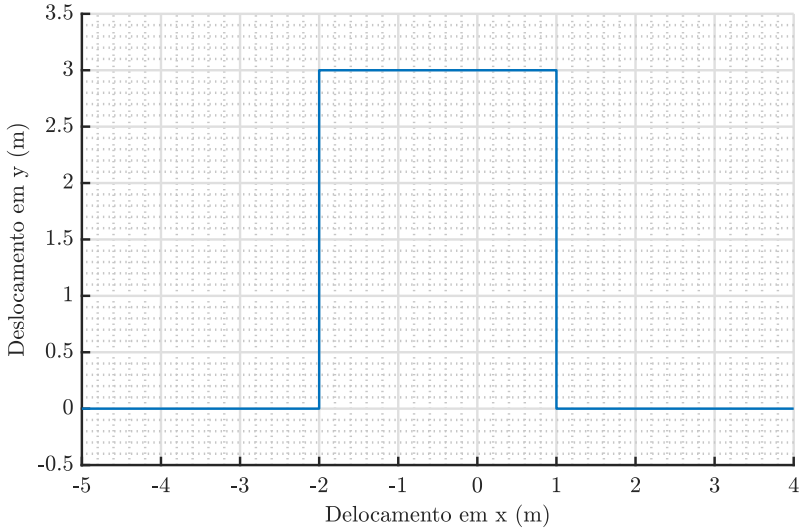


Figura 8.9: Trajetória de referência para o ensaio do controlador

canto ou extremidade	x_{ref} (m)	y_{ref} (m)	θ_{ref} (rad)
0	-5	0	0
1	-2	0	$\frac{\pi}{2}$
2	-2	3	0
3	1	3	$-\frac{\pi}{2}$
5	1	0	0
6	4	0	0

Tabela 8.2: Trajetória de referência para o ângulo

No entanto, falta ainda validar o comportamento do robô ao longo da execução da trajetória. Para este ensaio, utilizou-se o modelo de excitação completo, isto é, que contempla tanto a variação da posição do robô assim como do seu ângulo, com velocidades de referência de 0,5 m/s, 1 m/s, 1,5 m/s e 2 m/s.

Das figuras 8.12, verifica-se que com o aumento a velocidade de referência, a resposta entre ambos os algoritmos é idêntica, sendo que da velocidades mais elevadas, o gradiente descendente tem um comportamento mais oscilatório, sem que este se torne significativo para o desempenho do controlador.

Por outro lado, quando se analisa o comportamento do controlador em termos de execução da trajetória angular, verifica-se que o RPROP apresenta uma resposta mais rápida e mais próxima da desejada, isto é, de variação linear, com picos nos 90^o e -90^o.

Para aprofundar as conclusões sobre qual é o melhor algoritmo de otimização numérica, estendeu-se a análise efetuada a todas as velocidades entre 0 e 2 m/s, com um passo entre si de 0,1 m/s, através do estudo do erro médio quadrático, cujos resultados estão ilustrados nos gráficos das figuras 8.10 e 8.11. Desta forma, concluí-se que ambos os algoritmos

são razoavelmente bons para o seguimento de trajetórias lineares. No entanto, quando o assunto são as trajetórias angulares, o algoritmo RPROP prevalece sobre o gradiente descendente.

Em suma, dado o sucesso do algoritmo RPROP para minimização numérica da função objetivo, optou-se por utilizar apenas este algoritmo nas minimizações futuras.

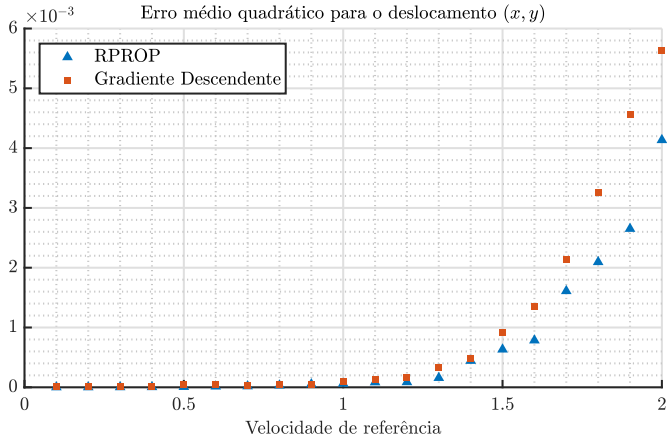


Figura 8.10: Erro de seguimento da trajetória de referência para validar o melhor algoritmo de otimização

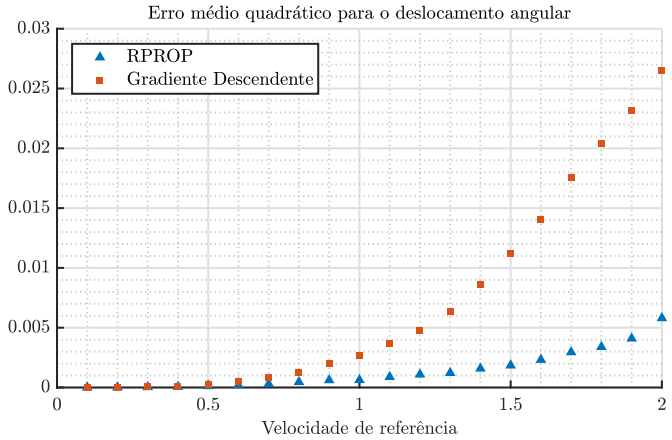


Figura 8.11: Erro de seguimento da trajetória angular de referência para validar o melhor algoritmo de otimização

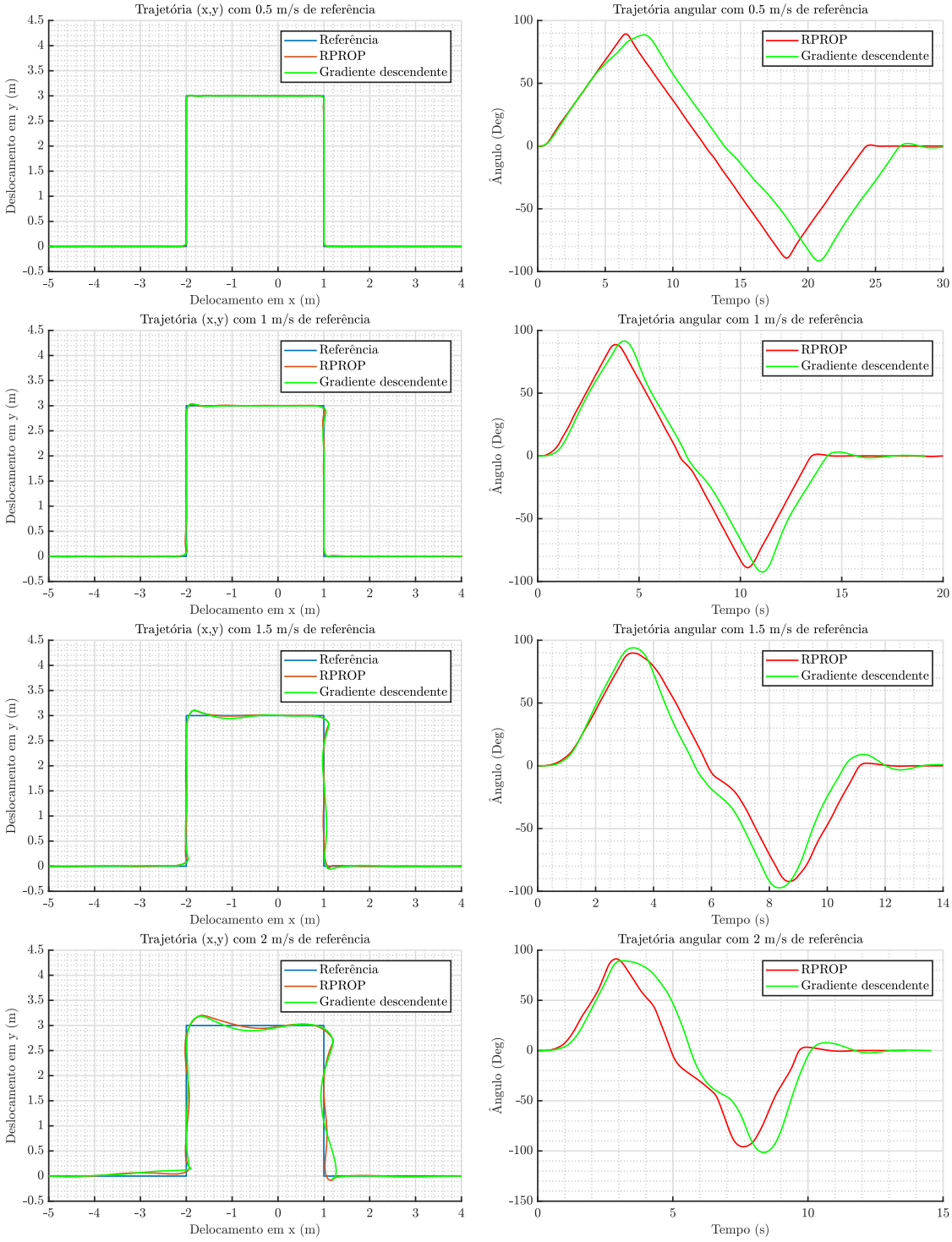


Figura 8.12: Resultado dos ensaios realizados para o estudo do melhor algoritmo de otimização

8.5.2 Análise ao comportamento do modelo do robô

Tal como referido anteriormente, o robô está fortemente limitado por uma rampa de aceleração nos motores, portanto, aqui pretende-se validar se é significativa a utilização do modelo dinâmico como complemento ao modelo cinemático.

Para apurar estas conclusões, o robô executou a trajetória da figura 8.9 com a variação do ângulo, tal como ilustrado na tabela 8.2, a quatro velocidades de referência distintas: 0,5 m/s, 1 m/s, 1,5 m/s e 2 m/s, que resultaram nos gráficos da figura 8.14.

Dos gráficos, verifica-se que não há alterações significativas entre os diferentes modelos. No entanto, para velocidades superiores, nota-se uma ligeira vantagem na utilização do modelo dinâmico, uma vez que este oscila ligeiramente, no entanto, tal facto é imperceptível e, como tal, insignificante, logo podemos assumir que em termos de seguimento da trajetória ambos os modelo são idênticos. Por outro lado, para o seguimento da trajetória angular, verifica-se que a variação angular é idêntica e que ambos seguem bem a trajetória angular de referência, no entanto, o MPC com o modelo cinemático consegue ser mais rápido que o mesmo com o modelo dinâmico.

Para complementar este estudo, pode-se proceder de forma similar à análise anterior, analisando-se o erro de seguimento, cujos resultados estão ilustrados nos gráficos da figura 8.13. De ambos, consegue-se validar as conclusões anteriormente feitas, em que a utilização dos dois modelos é idêntica.

Dado que os tempos de otimização da função objetivo são idênticos em ambos os casos, e uma vez que se está em ambiente de simulação, optou-se por utilizar o modelo dinâmico nos próximos ensaios, uma vez que o robô real terá um comportamento mais similar a este do que ao modelo cinemático.

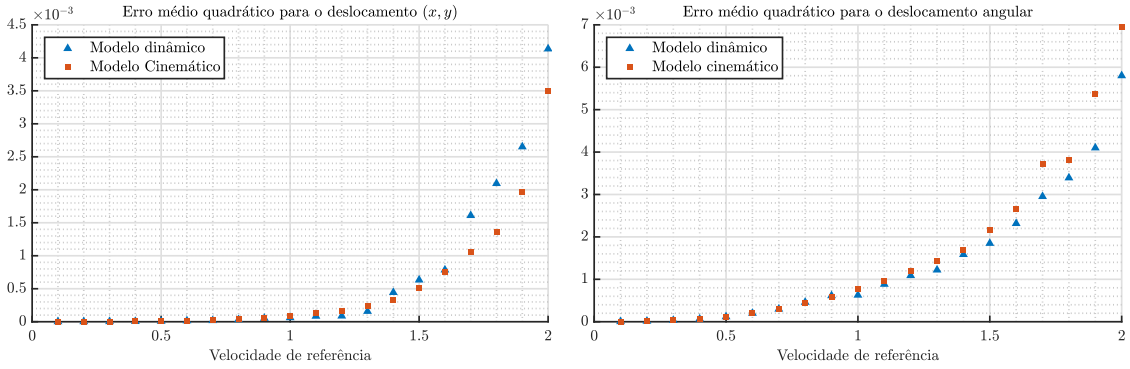


Figura 8.13: Erro de seguimento da trajetória de referência para validar a utilização do modelo cinemático e dinâmico

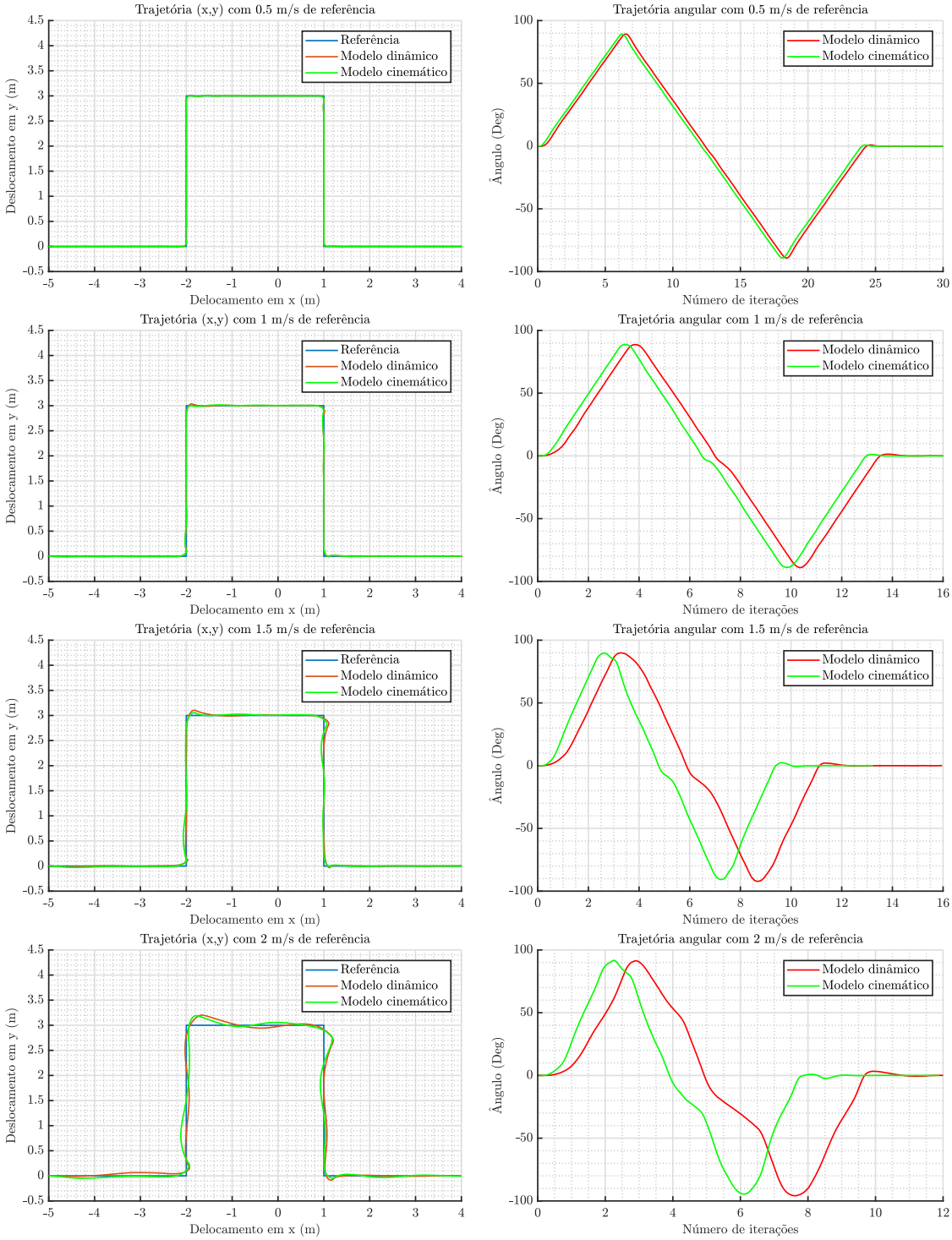


Figura 8.14: Resultados dos ensaios realizados para análise do modelo do robô

8.5.3 Análise ao controlador

Uma vez analisados os diferentes componentes da estrutura interna do controlador projetado, avaliando as diferentes opções feitas e se estas são significativas entre si, é ainda importante validar se há vantagem na transição entre um controlador PID para este controlador de trajetórias.

Desta forma, procederam-se a dois conjuntos de ensaios, por forma a garantir uma análise completa do comportamento dos controladores. Num primeiro conjunto de ensaios, considerou-se a trajetória em pulso, representada na figura 8.9, sem variação do ângulo do robô, isto é, a trajetória angular é representa por uma reta horizontal de valor 0 rad no tempo. No segundo conjunto de ensaios, considerou-se a mesma trajetória, mas com a variação da trajetória angular tal como descrito na tabela 8.2.

A figura 8.15 representa o resultado do seguimento da trajetória pulso do primeiro conjunto de ensaios, considerando as velocidades lineares de referência de 0,5 m/s, 1 m/s, 1,5 m/s e 2 m/s. A uma velocidade de 0,5 m/s não se verifica uma vantagem significativa na utilização do controlador MPC, tendo em conta a sua exigência computacional. No entanto, quando se consideram as velocidades lineares de referência superiores a 1 m/s, começa-se a tornar clara as vantagem do controlador preditivo; este consegue, à partida, tomar atitudes de controlo corretivas de acordo com a referência futura do robô, por outro lado, o facto de o controlador PID apenas considerar uma atitude corretiva ao erro entre a posição atual do robô e a sua posição de referência, torna a lei de controlo mais difícil e com um regime oscilatório.

A figura 8.16, que retrata o erro de seguimento da trajetória em pulso sem variação do ângulo em termos de posição e de ângulo, vem confirmar as conclusões até agora obtidas, verificando se, desta forma, que há vantagem, do ponto de vista de seguimento de trajetórias em posição, na utilização de um controlador de trajetórias do tipo MPC, nomeadamente para velocidades superiores a aproximadamente 1,3 m/s, velocidade a partir da qual se o observa um maior declive no aumento do erro de seguimento. No que concerne ao seguimento de uma trajetória angular, verifica-se o MPC apresenta menos erro de seguimento e como tal fornece uma melhor opção de controlo.

Para analisar o comportamento dos controladores quando considerada a trajetória em pulso com variação do ângulo de referência, pode-se estudar a figura 8.18 que representa o comportamento do robô com velocidades lineares de referência de 0,5 m/s, 1 m/s, 1,5 m/s e 2 m/s. Nestes casos é possível, a 1 m/s, verificar que há uma ligeira vantagem na utilização do controlador MPC, nomeadamente na execução dos pontos onde a trajetória muda sentido, onde o *overshoot* deste é inferior. Esta premissa continua a ser válida e nas mesmas condições quando se observa o comportamento dos controladores a velocidades superiores. Por este motivo, também se verifica que o erro quadrático médio do seguimento da trajetória (figura 8.19) aumenta com maior intensidade no controlador PID com o aumento da velocidade.

Da figura 8.20, verifica-se que é vantajoso a utilização do controlador MPC para o seguimento da trajetória de referência angular, quando comparado com o PID. No entanto, esta conclusão pode ser de certa forma precipitada e duvidosa quando se observa o comportado do robô no seguimento da mesma nos gráficos da figura 8.18, onde o PID apresenta um comportamento no seguimento mais rápido, linear e idêntico à trajetória angular de referência.

Em suma, se se considerar a análise entre controladores de um posto de vista de seguimento de trajetórias, constata-se que é vantajoso a utilização do MPC, nomeadamente se se pretender fazer o seguimento de posição. Não obstante, se se considerar o desempenho do controlador de uma forma global, pode-se concluir que a sua utilização não é sustentável, os elevados tempos de otimização numérica da função objetivo podem condicionar o funcionamento global do programa e, como tal, a ação de controlo, que começa a ser prejudicada por atrasos não considerados e não previsíveis.

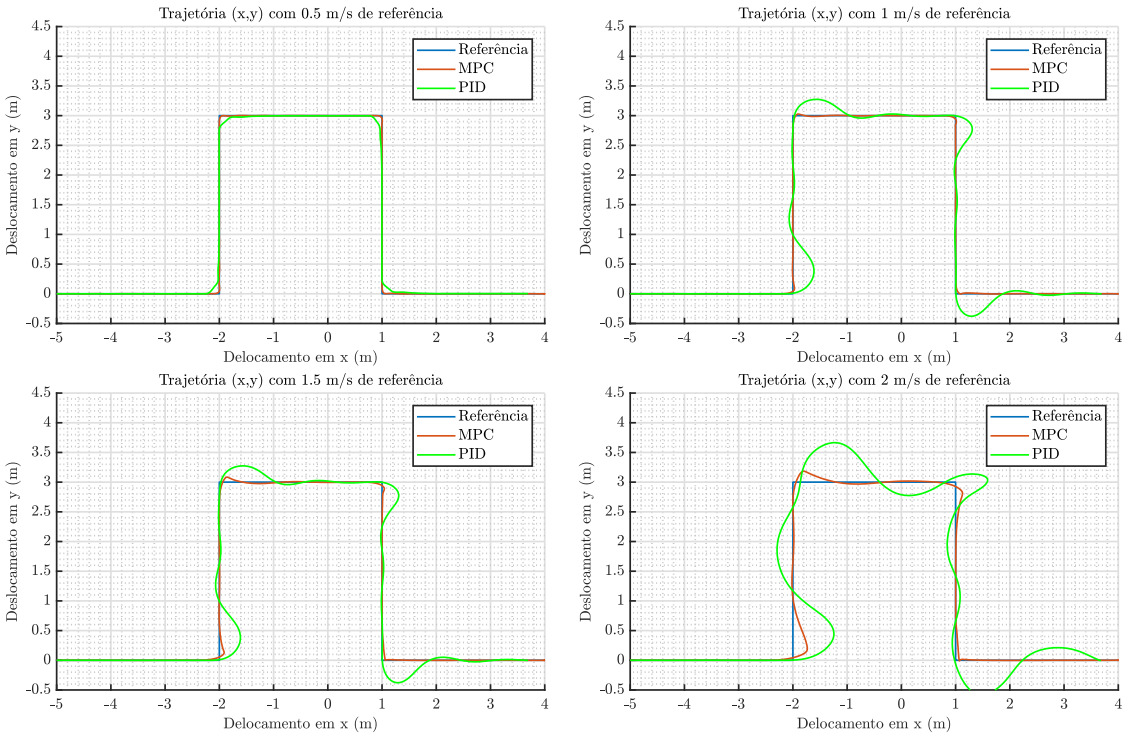


Figura 8.15: Resultados dos ensaios realizados para o seguimento da trajetória pulso da figura 8.9 sem variação do ângulo, para a comparação entre os controladores PID e MPC

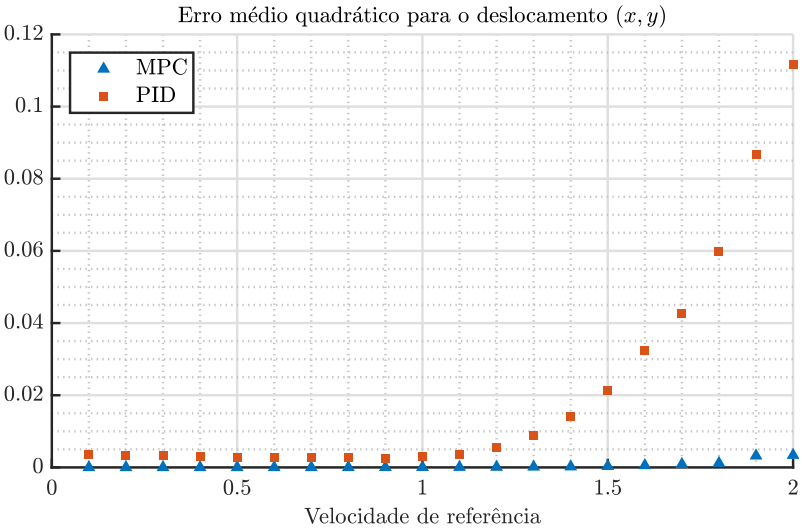


Figura 8.16: Erro de seguimento da trajetória de referência em posição, sem variação do ângulo, para validar a seleção do controlo PID ou MPC

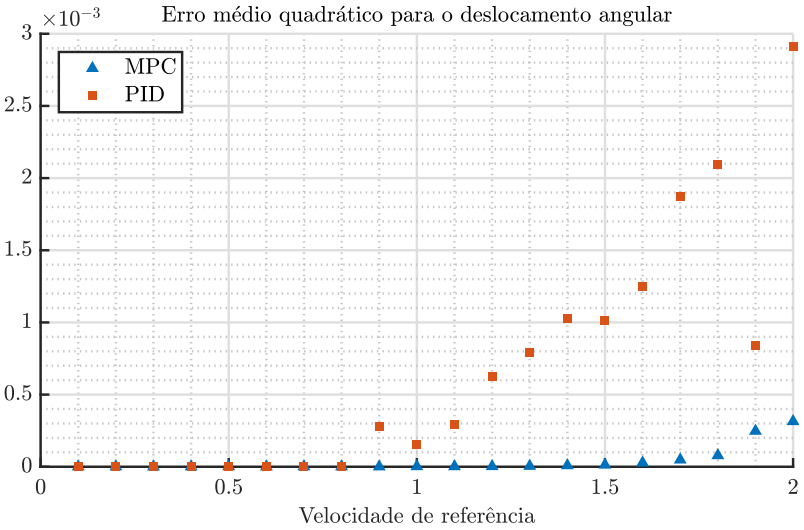


Figura 8.17: Erro de seguimento da trajetória angular de referência, sem variação do ângulo, para validar a seleção do controlo PID ou MPC

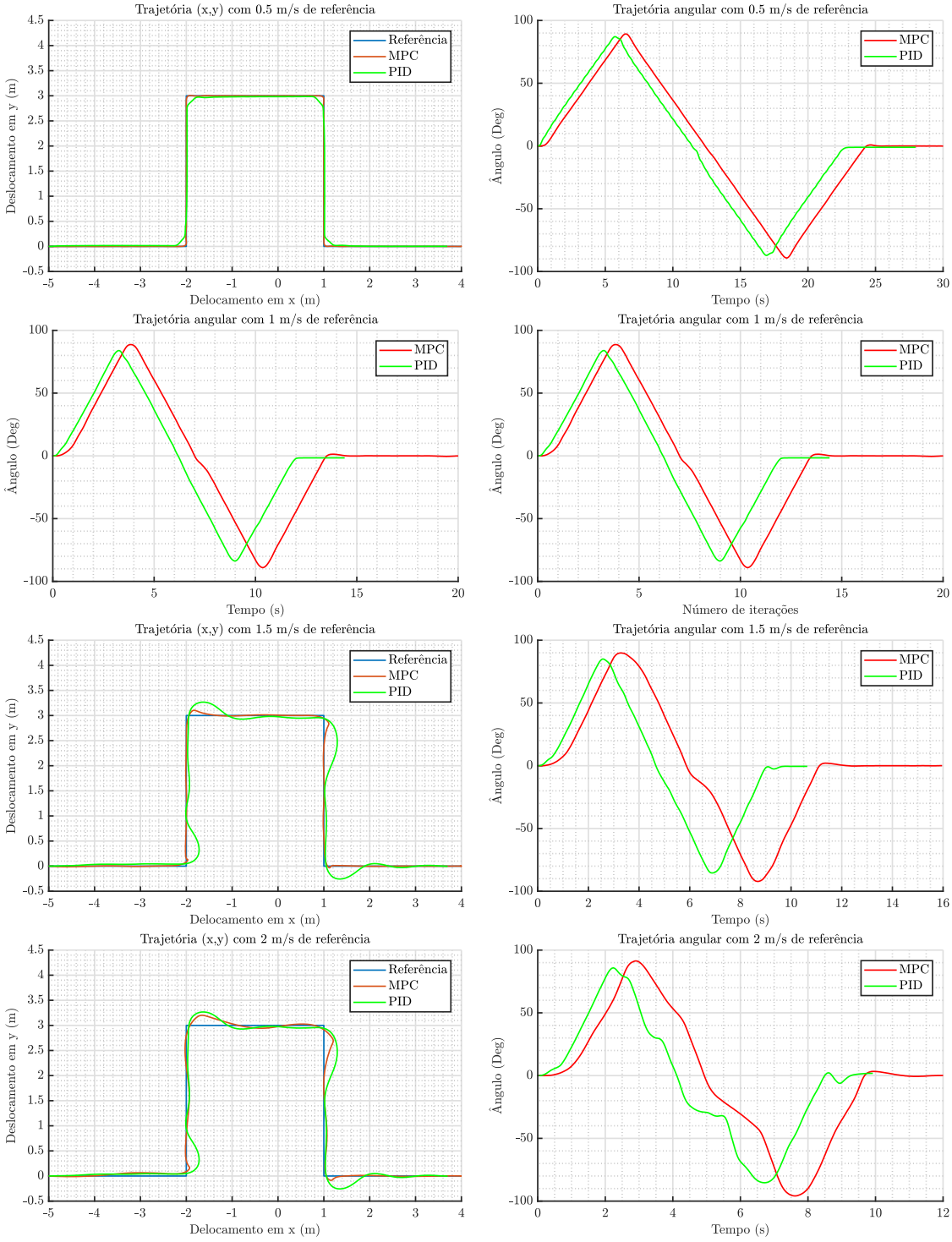


Figura 8.18: Resultados dos ensaios realizados para o seguimento da trajetória pulso da figura 8.9 com variação do ângulo tal como exposto na tabela 8.2, para a comparação entre os controladores PID e MPC

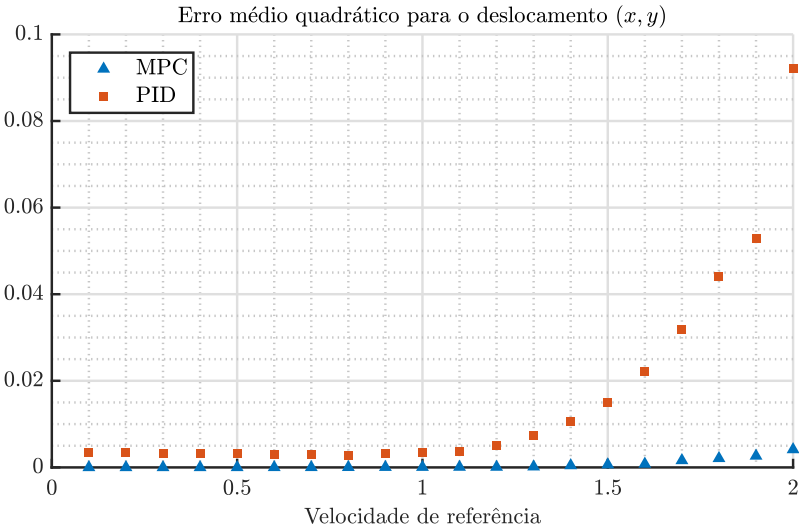


Figura 8.19: Erro de seguimento da trajetória de referência em posição, com variação do ângulo, para validar a seleção do controlo PID ou MPC

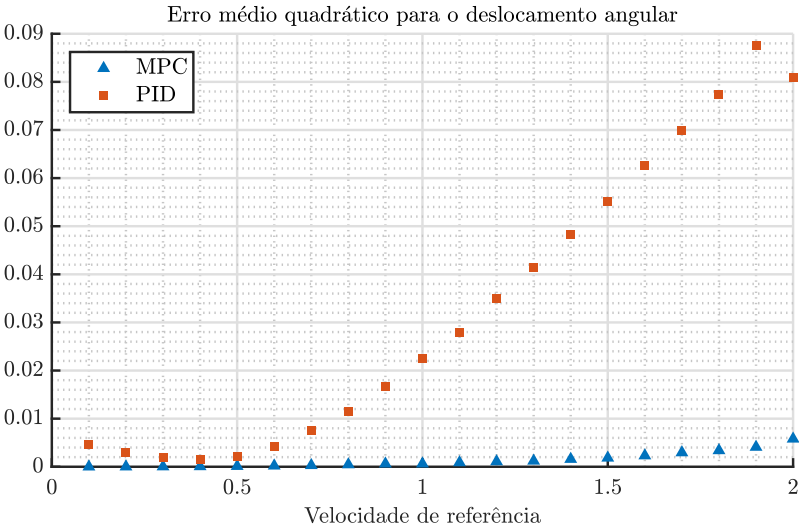


Figura 8.20: Erro de seguimento da trajetória angular de referência, com variação do ângulo, para validar a seleção do controlo PID ou MPC

8.5.4 Ensaio do algoritmo no robô

Das conclusões obtidas dos resultados anteriores, é possível afirmar que o robô apresenta um comportamento controlado. No entanto, por razões de segurança e de espaço, considerou-se que o robô estaria apto a circular dentro do laboratório até uma velocidade de 1,5 m/s.

Pela última razão supramencionada, a trajetória em pulso considerada na figura 8.9 foi modificada para uma trajetória similar com segmentos de reta de 2 m. Assim sendo, a trajetória a ser executada pelo robô em laboratório é representada figura 8.21 com uma variação suave do ângulo, tal como descrito na tabela 8.3.

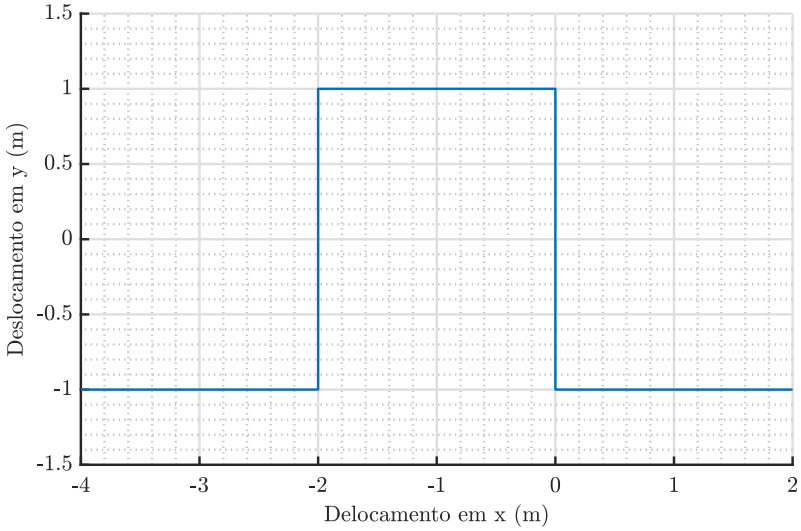


Figura 8.21: Trajetória de referência para o ensaio do controlador no robô

canto ou extremidade	x_{ref} (m)	y_{ref} (m)	θ_{ref} (rad)
0	-4	-1	0
1	-2	-1	$\frac{\pi}{2}$
2	-2	1	0
3	0	1	$-\frac{\pi}{2}$
5	0	-1	0
6	2	-1	0

Tabela 8.3: Trajetória angular de referência para trajetória da figura 8.21

Uma vez que neste ensaio se pretende fazer uma validação semelhante à que foi feita na secção 8.5.3, isto é, validar o desempenho do controlador projetado em relação ao controlador PID original, foram, da mesma forma, considerados os dois testes. Numa primeira fase considera-se a trajetória de referência da figura 8.21 sem a variação do ângulo

de referência do robô e no segundo conjunto de testes considera-se a mesma trajetória, mas com a variação suave do ângulo de referência, tal como descrito na tabela 8.3.

Do primeiro ensaio, verifica-se desde a velocidade de 0,5 m/s a clara vantagem do MPC para o seguimento de trajetórias, tal como ilustram os gráficos da figura 8.22.

Embora com mais erro, algo semelhante se sucede no segundo ensaio, gráficos da figura 8.23, onde se considerou a variação do ângulo do robô. Neste caso, o robô foi capaz de fazer um bom seguimento da posição com *overshoot* máximo inferior a 20 cm, mas em grande parte da trajetória inferior a 10 cm, para velocidades inferiores a 1 m/s. Para uma velocidade linear de referência de 1,5 m/s, o seguimento da trajetória também foi razoável, embora por vezes, o erro de seguimento tivesse atingido os 40 cm.

No que concerne ao seguimento do ângulo, verifica-se que o robô é capaz de fazer um bom seguimento com ambos os controladores. Mais particularmente entre eles, verifica-se que ambos têm um bom desempenho, no entanto, o MPC efetua um seguimento mais suave e nos casos onde há mudança do ângulo de referência, verifica-se que há a sua previsão e o controlador antecede os sinais de controlo para se adaptarem à realidade futura.

Apesar do sucesso do MPC no robô para seguimento quer da trajetória linear como angular, constatou-se uma realidade distinta no que concerne ao tempo de otimização da lei de controlo. O seu tempo médio subiu para aproximadamente 35 ms sendo que o seu máximo ficou muito próximo do tempo de ciclo, isto é, de 40 ms. Embora o software consiga processar normalmente o algoritmo de controlo, alguns algoritmos ou partes do programa poderão começar a ficar comprometidos, nomeadamente a interface gráfica que deixa de permitir a interação com o utilizador ou o algoritmo de localização, também ele baseado em métodos numéricos, que deixa de ter tempo livre para ser processado. Além disso, o ultrapassar do ciclo de controlo pode levar a que haja perda de dados, nomeadamente os relacionados com a localização do robô, como a imagem, cuja perda de *frames* leva a que não seja a atualizada a posição do robô e por esse motivo, a trajetória executada e registada pelo robô pode não corresponder exatamente à real.

Por conseguinte, embora os resultados produzidos pelo controlador sejam razoáveis, o tempo que este toma para otimizar a lei de controlo é muito grande, o que origina problemas na execução normal do programa. Portanto, pode-se antever que esta não é uma solução praticável para o seguimento de trajetórias em robôs móveis de alto desempenho, onde o cumprimento do ciclo do programa é crucial, para que se consiga garantir a estabilidade do sistema e que todos os algoritmos são devidamente executados.

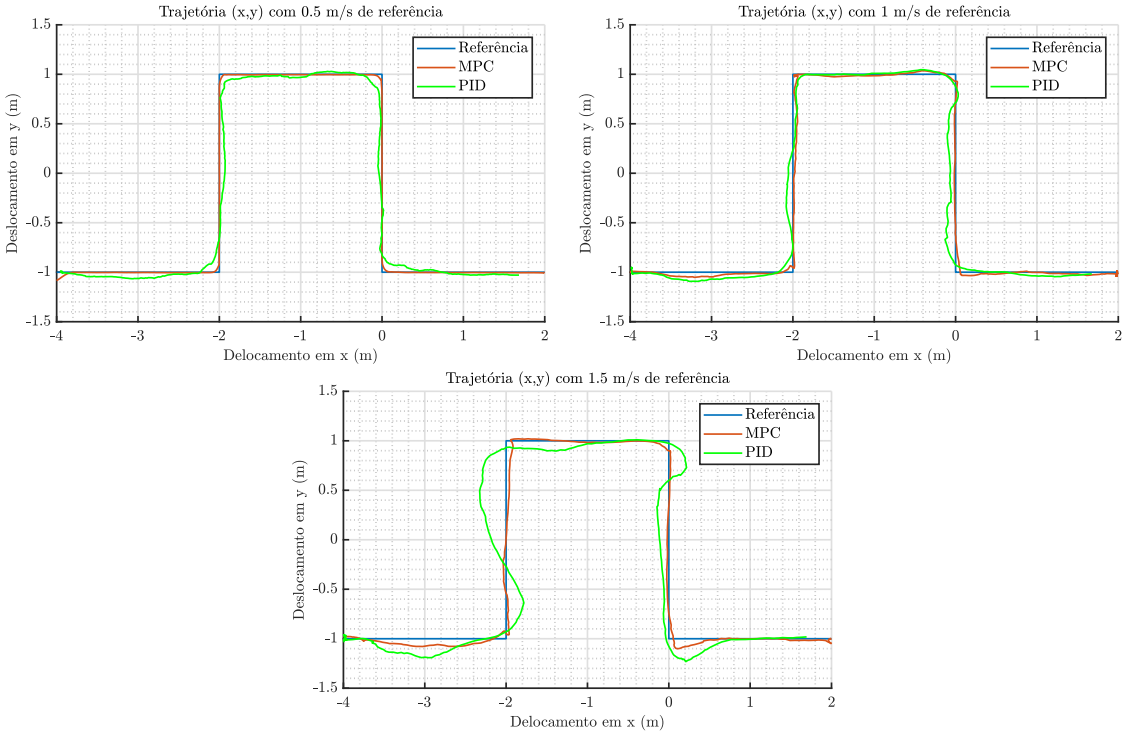


Figura 8.22: Resultados dos ensaios realizados para o seguimento da trajetória pulso da figura 8.9 sem variação do ângulo, para a comparação entre os controladores PID e MPC

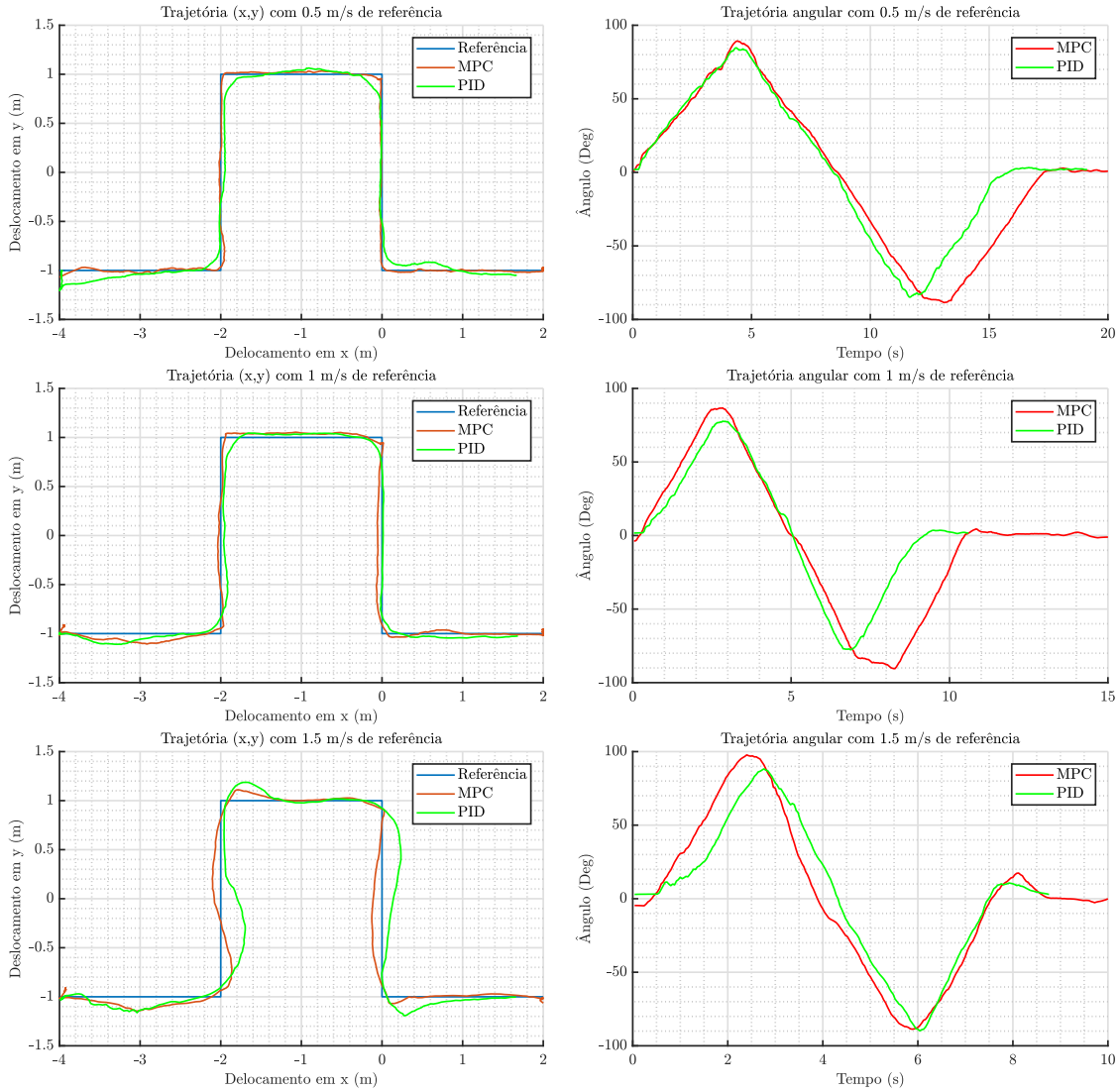


Figura 8.23: Resultados dos ensaios realizados para o seguimento da trajetória pulso da figura 8.9 com variação do ângulo tal como exposto na tabela 8.2, para a comparação entre os controladores PID e MPC

Capítulo 9

Conclusão e trabalho futuro

9.1 Conclusão

No âmbito desta dissertação, pretendeu-se projetar um controlador que fosse rápido e robusto o suficiente para garantir um controlo estável de um robô a elevadas velocidades.

Desta forma, começou-se por dar continuidade ao trabalho realizado por Pedro Relvas [38] no que se refere ao modelo do robô no simulador SimTwo. Neste pretendeu-se obter um modelo realista do robô, sem aumentar a exigência computacional da simulação. Este modelo foi obtido quer por métodos experimentais, quer por aproximação por métodos numéricos, através da utilização do algoritmo RPROP. Por métodos experimentais, foi calculado o atrito viscoso e estático dos motores, recorrendo-se para tal a um conjunto de ensaios dedicados, no robô. Foi ainda determinada, experimentalmente, a altura do centro de massa do robô e o limite de aceleração dos motores. Os ganhos do controlador PID de velocidade dos motores e o momento de inércia foram aproximados numericamente à resposta resultante do robô de um conjunto de sinais de excitação gerados que garantissem alguma excitabilidade do sistema, sem amplitudes baixas, afastando-se, assim, da zona morta, e sem a saturação dos motores.

A obtenção de um modelo realista do sistema a controlar no simulador foi deveras importante, permitindo a realização de ensaios seguros do controlador projetado no robô simulado, facilitando a visualização dos resultados do processo simulado e a realização de múltiplos ensaios num menor período de tempo.

De seguida, procurou-se melhorar a resposta do controlador utilizado pela equipa de futebol robótico, projetando-se dois novos controladores reativos, um baseado no algoritmo de seguimento de trajetórias *go to position* e o segundo baseado no algoritmo *follow line*. Para a anulação do erro de seguimento, considerou-se um controlador PD, onde os polos do sistema em malha fechada foram reposicionados segundo o protótipo de Bessel.

Entre este controladores projetados, constata-se que ambos têm uma resposta similar entre si, havendo uma ligeira vantagem na utilização do algoritmo *follow line*, nomeadamente às velocidades mais elevadas, sendo esta conclusão obtida, quer em ambiente

de simulação, quer nos ensaios realizados com o robô real. Em relação ao controlador inicial, atualmente utilizado pela equipa, constata-se que houve uma ligeira melhoria na resposta dos novos controladores projetados. Apesar destas melhorias, verifica-se, ainda, que há uma oscilação elevada do sistema às velocidades mais elevadas e com *overshoot* relativamente elevado.

Portanto, numa terceira fase de implementações, optou-se por projetar um controlador preditivo, que apresenta a característica de adaptar o sinal de controlo do robô por forma a minimizar o erro futuro. Este surge no seguimento do trabalho realizado por Ferreira [9], sendo que se adaptou o modelo do robô à nova realidade do mesmo, isto é, o sistema de primeira ordem que aproximava o comportamento do sistema foi substituído por uma rampa que caracteriza limite de aceleração. Além disso, a geração da trajetória de referência passou a comportar a aceleração máxima do robô. Para a minimização da função objetivo, utilizou-se o algoritmo RPROP que demonstrou convergir mais rapidamente que o gradiente descendente.

Dos resultados obtidos, demonstra-se que há uma grande vantagem na utilização do controlo preditivo comparativamente ao controlo PID. Este demonstra ter um menor erro de seguimento, menor *overshoot* e menor oscilação. No entanto, também se comprovou que demora muito mais tempo a calcular a lei de controlo que o controlador PID, o que, por vezes, pode ser crítico, nomeadamente quando é ultrapassado o tempo de ciclo de programa.

Em suma, ao longo desta tese, conseguiu-se obter um modelo dinâmico do robô, por forma a garantir um simulação fidedigna do sistema a controlar, e fez-se um estudo entre as diferente estruturas de controlo, PID e MPC, balizando algumas das vantagens e desvantagens entre elas.

9.2 Trabalho futuro

Das conclusões feitas na secção anterior, isto é, no caso do controlador PID o erro de seguimento e o elevado *overshoot* e no caso do controlador preditivo o elevado tempo para o cálculo da lei de controlo, sugere-se, nesta secção duas abordagens.

No caso do controlador MPC, sugere-se o estudo e implementação de um controlador baseado em *Fast MPC*, que procura otimizar o algoritmo de cálculo numérico da lei de controlo, minimizando, desta forma, o tempo de processamento da lei de controlo.

Por outro lado, para o controlador PID, sugere-se a implementação da componente *feedforward*, tal como se ilustra na figura 9.1. Esta terá como entrada a trajetória de referência que deverá ser aproximada por uma função recorrendo a um algoritmo, como por exemplo, os mínimos quadrados. Em vez de se considerar toda a trajetória, dever-se-á seguir uma estratégia semelhante ao MPC e considerar um horizonte futuro limitado.

No que se refere ao modelo do sistema a utilizar, recomenda-se um modelo representativo do limite de aceleração do robô. Dada a estrutura deste, que lhe confere a

possibilidade de se mover em dois eixo (x, y) e de rodar sobre si próprio, este controlador deverá ser implementado em três controladores PID com *feedforward*, um cada movimento anteriormente referido.

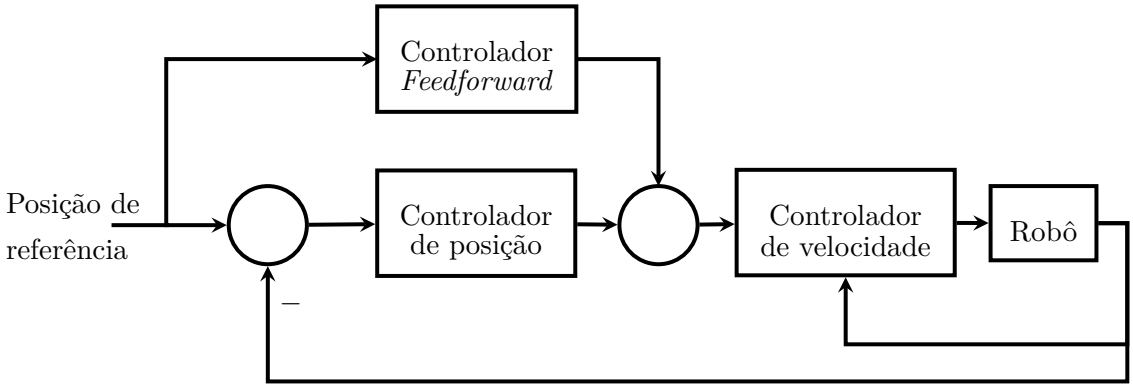


Figura 9.1: Estrutura de controlo com *feedforward* proposta

Por fim, mantendo o trabalho executado nesta dissertação, sugere-se a realização de ensaios intensivos com o robô real, no sentido de ajustar os parâmetros do controlador para este, dado que o controlador projetado foi afinado no simulador. Além disso, sugere-se ainda o uso de novos métodos que procurem testar e validar, de forma robusta, o modelo de simulação obtido.

Anexo A

Algoritmos de otimização numérica

Os métodos iterativos são amplamente utilizados em problemas de otimização numérica onde uma solução analítica não é possível quer pela característica não linear do problema, quer pela existência de restrições que a inviabilizam. Desta forma, estes algoritmos procuram a solução do problema de uma forma iterativa, percorrendo a evolução de uma determinada função objetivo à procura de um mínimo local ou, idealmente, global.

Existem vários algoritmos de otimização numérica, como é o caso do gradiente descendente e do *resilient propagatition* (comummente designado como RPROP). Uma particularidade comum a ambos é o facto destes evoluírem com base na informação fornecida pelo gradiente da função objetivo, mas no entanto, nunca atingirem a solução ótima, aproximando-se, no entanto, bastante desta, o que é suficiente para grande parte dos problemas.

A.1 Gradiente descendente

O método do gradiente descendente (ou máximo declive, do inglês *steepest descendent*) é um algoritmo de otimização numérica que tem por base o estudo da evolução da derivada de primeira ordem da função objetivo. Assim, para encontrar um mínimo de uma função, este recorre a um esquema iterativo, como o representado na figura A.1, que evolui no sentido oposto ao do sentido de maior crescimento do gradiente da função objetivo.

Portanto, dada uma determinada função objetivo $E(x)$, tem-se que a pesquisa pela solução do problema é feita de acordo como o padrão da equação A.1.

$$x_n = x_{n-1} + \alpha_n \cdot d_n \tag{A.1}$$

Nesta equação, tem-se que α_n é um coeficiente multiplicativo e que d_n é, neste caso, o gradiente da função objetivo. Assim, tem-se que a evolução do valor a otimizar é dada

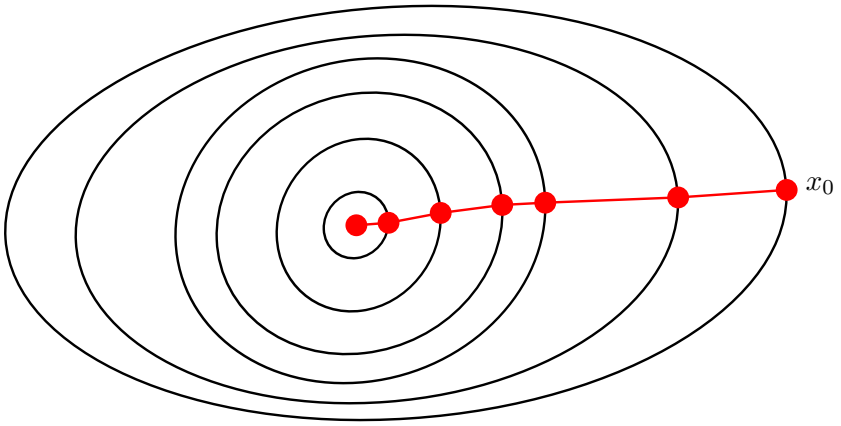


Figura A.1: Esquema iterativo para otimização da função objetivo

pela equação A.2.

$$x_n = x_{n-1} - \alpha_n \cdot \nabla E(x_n) \tag{A.2}$$

O procedimento de otimização de uma variável por gradiente descendente pode ser descrito pelo algoritmo 3.

Algoritmo 3: Gradiente descendente

```
1 begin
2     Inicialização de x_0
3     for ||∇E(x_{n+1})|| < ε do
4         x_{n+1} ← x_n − α_n · ∇E(x_n)
```

A.2 *Resilient Propagation*

O algoritmo RPROP foi pela primeira vez introduzido por Riedmiller e Braun [41] em 1993. Este efetua uma adaptação direta do passo de atualização recorrendo à informação fornecida pelo gradiente local, ignorando a sua magnitude.

Para atingir o seu objetivo, o algoritmo introduz o seu próprio valor de atualização Δ_{ij}, que substitui a magnitude do gradiente descendente, e é utilizado para atualizar o peso de atualização, ω_{ij}, da variável a ser estimada. Assim, de forma similar ao gradiente descendente, o RPROP também depende do gradiente da função objetivo, que é utilizada de forma distinta. Portanto, o valor de atualização Δ_{ij} pode ser obtido pelo sistema de equações A.3.

$$\Delta_{ij}^{(t)} = \begin{cases} \eta^+ \times \Delta_{ij}^{(t-1)} & \text{, se } \frac{\partial E}{\partial \omega_{ij}}^{(t-1)} \times \frac{\partial E}{\partial \omega_{ij}}^{(t)} > 0 \\ \eta^- \times \Delta_{ij}^{(t-1)} & \text{, se } \frac{\partial E}{\partial \omega_{ij}}^{(t-1)} \times \frac{\partial E}{\partial \omega_{ij}}^{(t)} < 0 \\ \Delta_{ij}^{(t-1)} & \text{, se outro} \end{cases} \qquad \text{onde } 0 < \eta^- < 1 < \eta^+ \tag{A.3}$$

Uma vez adaptado o valor de atualização para cada peso, a atualização do peso ω_{ij} pode ser feito através de uma regra simples ilustrada em A.4 e A.5.

$$\Delta\omega_{ij}^{(t)} = \begin{cases} -\Delta_{ij}^{(t)} & \text{, se } \frac{\partial E}{\partial \omega_{ij}}^{(t)} > 0 \\ +\Delta_{ij}^{(t)} & \text{, se } \frac{\partial E}{\partial \omega_{ij}}^{(t)} < 0 \\ 0 & \text{, se outro} \end{cases} \tag{A.4}$$

$$\omega_{ij}^{(t+1)} = \omega_{ij}^{(t)} + \Delta\omega_{ij}^{(t)} \tag{A.5}$$

Desta forma, tem-se que o procedimento demonstrado, pode ser implementado tal como se ilustra no algoritmo 4.

Algoritmo 4: RPROP

Input: ω_{ij} para otimizar; $\frac{\partial E}{\partial \omega_{ij}}^{(t)}$ (gradiente local do erro)
Output: $\omega_(ij)$ otimizado
Data: $\Delta_{máximo}$ e $\Delta_{mínimo}$ (tolerâncias máximas e mínimas para o valor de atualização Δ_{ij})

begin
  if $\frac{\partial E}{\partial \omega_{ij}}^{(t)} \neq 0$ **then**
    if $\frac{\partial E}{\partial \omega_{ij}}^{(t-1)} \times \frac{\partial E}{\partial \omega_{ij}}^{(t)} > 0$ **then**
      $\Delta_{ij}^{(t)} \leftarrow$ mínimo($\Delta_{ij}^{(t-1)} \times \eta^-$, $\Delta_{máximo}$)
    else if $\frac{\partial E}{\partial \omega_{ij}}^{(t-1)} \times \frac{\partial E}{\partial \omega_{ij}}^{(t)} < 0$ **then**
      $\Delta_{ij}^{(t)} \leftarrow$ máximo($\Delta_{ij}^{(t-1)} \times \eta^+$, $\Delta_{mínimo}$)
      $\frac{\partial E}{\partial \omega_{ij}}^{(t)} \leftarrow 0$
    else
      $\Delta_{ij}^{(t)} \leftarrow \Delta_{ij}^{(t-1)}$
  if $\frac{\partial E}{\partial \omega_{ij}}^{(t)} > 0$ **then**
    $\omega_{ij}(t+1) \leftarrow \omega_{ij}(t) - \Delta_{ij}$
  else if $\frac{\partial E}{\partial \omega_{ij}}^{(t)} < 0$ **then**
    $\omega_{ij}(t+1) \leftarrow \omega_{ij}(t) + \Delta_{ij}$
  else
    $\omega_{ij}(t+1) \leftarrow \omega_{ij}(t)$

Referências

[1] Patrick F Muir e Charles P Neuman. Kinematic modeling for feedback control of an omnidirectional wheeled mobile robot. Em *in Proc. of Int. Conf. on Robotics and Automation*, volume 4, páginas 1772–1778, New York City, 1987. Institute of Electrical and Electronics Engineers. URL: `http://ieeexplore.ieee.org/document/1087767/`, doi:10.1109/ROBOT.1987.1087767.

[2] K Watanabe, Y Shiraishi, S G Tzafestas, J Tang, e T Fukuda. Feedback Control of an Omnidirectional Autonomous Platform for Mobile Service Robots. *Journal of Intelligent and Robotic Systems: Theory and Applications*, 22(3-4):315–330, 1998. doi:10.1023/A:1008048307352.

[3] André Gustavo Scolari Conceição. *Controlo e cooperação de robôs móveis autónomos omnidireccionais*. Doutoramento, Universidade do Porto, 2007. URL: `http://hdl.handle.net/10216/11398`.

[4] Khoshnam Shojaei, Alireza Mohammad Shahri, Ahmadreza Tarakameh, e Behzad Tabibian. Adaptive trajectory tracking control of a differential drive wheeled mobile robot. *Robotica*, 29(3):391–402, may 2011. URL: `http://www.journals.cambridge.org/abstract{_}S0263574710000202`, doi:10.1017/S0263574710000202.

[5] Mou Chen. Disturbance Attenuation Tracking Control for Wheeled Mobile Robots With Skidding and Slipping. *IEEE Transactions on Industrial Electronics*, 64(4):3359–3368, apr 2017. URL: `http://ieeexplore.ieee.org/document/7577731/`, doi:10.1109/TIE.2016.2613839.

[6] Veer Alakshendra e Shital S. Chiddarwar. Adaptive robust control of Mecanum-wheeled mobile robot with uncertainties. *Nonlinear Dynamics*, 87(4):2147–2169, mar 2017. URL: `http://link.springer.com/10.1007/s11071-016-3179-1`, doi:10.1007/s11071-016-3179-1.

[7] Carlos E. García, David M. Prett, e Manfred Morari. Model predictive control: Theory and practice-A survey. *Automatica*, 25(3):335–348, may 1989. URL: `http://linkinghub.elsevier.com/retrieve/pii/0005109889900022`, doi:10.1016/0005-1098(89)90002-2.

[8] D.R. Ramirez, D Limon, J Gomez-Ortega, e E.F. Camacho. Nonlinear MBPC for mobile robot navigation using genetic algorithms. Em *Proceedings 1999 IEEE International Conference on Robotics and Automation*, volume 3, páginas 2452–2457, Detroit, MI, USA, 1999. IEEE. URL: `http://ieeexplore.ieee.org/document/770473/`, doi:10.1109/ROBOT.1999.770473.

[9] João Rodrigo Alvelos Ferreira. *Controlo coordenado de equipas de robots móveis*. Tese de doutoramento, Universidade do Porto, 2010. URL: http://hdl.handle.net/10216/59661.

[10] J C L Barreto S., A G S Conceição, C E T Dórea, L Martinez, e E R de Pieri. Design and Implementation of Model-Predictive Control With Friction Compensation on an Omnidirectional Mobile Robot. *IEEE/ASME Transactions on Mechatronics*, 19(2):467–476, 2014. doi:10.1109/TMECH.2013.2243161.

[11] António Moreira. Apontamentos das aulas de sistemas robóticos autónomos - capítulo 3 cinemática, dinâmica e controlo de robôs, 2013.

[12] T.-H.S. Li, Sheng-Sung Jian Sheng-Sung Jian, e Ming-Che Tsai Ming-Che Tsai. Design and implementation of fuzzy trajectory following and\nplanning control for mobile robots. Em *Proceedings of IEEE Region 10 International Conference on Electrical and Electronic Technology. TENCON 2001 (Cat. No.01CH37239)*, volume 1, páginas 453–458, Singapore, 2001. IEEE. URL: http://ieeexplore.ieee.org/document/949634/, doi:10.1109/TENCON.2001.949634.

[13] André S. Conceição, António P. Moreira, e Paulo Costa. Controller optimization and modelling of an omni-directional mobile robot. Em *CONTROLO 2006*, 2006. URL: http://hdl.handle.net/10216/79811.

[14] Tiago Pereira Do Nascimento. *Controle de trajetória de robôs móveis omnidirecionais : uma abordagem multivariável*. Tese de doutoramento, Universidade Federal da Bahia, 2009. URL: http://www.ppgee.eng.ufba.br/teses/ac64916ca6ef3bdd2f466bffdd30af7e.pdf.

[15] Peter Šuster e Anna Jadlovská. Tracking Trajectory of the Mobile Robot Khepera II Using Approaches of Artificial Intelligence. *Acta Electrotechnica et Informatica*, 11(1):38–43, 2011. URL: http://www.aei.tuke.sk/papers/2011/1/06{_}Suster.pdf, doi:10.2478/v10198-011-0006-y.

[16] S. Mobayen. Fast terminal sliding mode tracking of non-holonomic systems with exponential decay rate. *IET Control Theory and Applications*, 9(8):1294–1301, may 2015. URL: http://digital-library.theiet.org/content/journals/10.1049/iet-cta.2014.1118, doi:10.1049/iet-cta.2014.1118.

[17] Eduardo Fernandez Camacho e Carlos Bordons Alba. *Model Predictive Control in the Process Industry*. Advances in Industrial Control. Springer-Verlag, London, 1 edição, 1995. URL: http://link.springer.com/10.1007/978-0-85729-398-5.

[18] L Zadeh e B Whalen. On optimal control and linear programming. *IRE Transactions on Automatic Control*, 7(4):45–46, 1962. URL: http://ieeexplore.ieee.org/xpls/abs{_}all.jsp?arnumber=1105469{%}0Ahttp://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1105469, doi:10.1109/TAC.1962.1105469.

[19] A. I. Propoi. Use of linear programming methods for synthesizing sampled-data automatic systems. *Automation and Remote Control*, 24(7):837–844, 1963.

[20] J.L. Testud J. Richalet, A. Rault e J. Papon. Model predictive heuristic control: Applications to industrial processes. *Automatica*, 14(5):413–428, 1978. URL: `http://www.sciencedirect.com/science/article/pii/0005109878900018`, doi:https://doi.org/10.1016/0005-1098(78)90001-8.

[21] C. R. Cutler e B. L. Ramaker. Dynamic matrix control – a computer control algorithm. Em *Joint Automatic Control Conference*, volume 17, página 72, San Francisco, California, 1980. doi:10.1109/JACC.1980.4232009.

[22] D.M. Prett e R.D. Gillette. Optimization and constrained multivariable control of a catalytic cracking unit. Em *Joint Automatic Control Conference*, volume 17, página 73, San Francisco, California, 1980. doi:10.1109/JACC.1980.4232010.

[23] Dongbing Gu e Huosheng Hu. Wavelet neural network based predictive control for mobile robots. Em *SMC 2000 Conference Proceedings. 2000 IEEE International Conference on Systems, Man and Cybernetics. 'Cybernetics Evolving to Systems, Humans, Organizations, and their Complex Interactions' (Cat. No.00CH37166)*, volume 5, páginas 3544–3549, Nashville, TN, USA, 2000. IEEE. URL: `http://ieeexplore.ieee.org/document/886558/`, doi:10.1109/ICSMC.2000.886558.

[24] Xianhua Jiang, Yuichi Motai, e Xingquan Zhu. Predictive fuzzy logic controller for trajectory tracking of a mobile robot. Em *Proceedings of the 2005 IEEE Midnight-Summer Workshop on Soft Computing in Industrial Applications, 2005. SMCia/05.*, número m, páginas 29–32, Espoo, Finland, Finland, 2005. IEEE. URL: `http://ieeexplore.ieee.org/document/1466943/`, doi:10.1109/SMCIA.2005.1466943.

[25] Gregor Klancar e Igor Skrjanc. Predictive Trajectory Tracking Control for Mobile Robots. Em *2006 12th International Power Electronics and Motion Control Conference*, páginas 373–378, Portoroz, Slovenia, aug 2006. IEEE. URL: `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4061733`, doi:10.1109/EPEPEMC.2006.283188.

[26] Stavros G Vougioukas. Reactive Trajectory Tracking for Mobile Robots based on Non Linear Model Predictive Control. Em *Proceedings 2007 IEEE International Conference on Robotics and Automation*, número April, páginas 3074–3079, Roma, Italy, apr 2007. IEEE. URL: `http://ieeexplore.ieee.org/document/4209557/`, doi:10.1109/ROBOT.2007.363939.

[27] K. Kanjanawanishkul e A. Zell. Path following for an omnidirectional mobile robot based on model predictive control. Em *2009 IEEE International Conference on Robotics and Automation*, páginas 3341–3346, Kobe, Japan, may 2009. IEEE. URL: `http://ieeexplore.ieee.org/document/5152217/`, doi:10.1109/ROBOT.2009.5152217.

[28] Rolf Findeisen e Frank Allgöwer. An Introduction to Nonlinear Model Predictive Control. Em *21st Benelux Meeting on Systems and Control*, páginas 1–23, Veldhoven, 2002.

[29] C J Ostafew, A P Schoellig, e T D Barfoot. Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments. Em *2014 IEEE International Conference on Robotics and Automation (ICRA)*, páginas 4029–4036, Hong Kong, China, 2014. IEEE. doi:10.1109/ICRA.2014.6907444.

[30] H Xiao, Z Li, C Yang, L Zhang, P Yuan, L Ding, e T Wang. Robust Stabilization of a Wheeled Mobile Robot Using Model Predictive Control Based on Neurodynamics Optimization. *IEEE Transactions on Industrial Electronics*, 64(1):505–516, 2017. doi:10.1109/TIE.2016.2606358.

[31] Sandro Magalhães e Tiago Mendonça. Implementação de rotinas básicas de controlo do movimento de um robô (GoToXYTheta). Relatório técnico, Faculdade de Engenharia da Universidade do Porto, Porto, 2017.

[32] Sandro Magalhães e Tiago Mendonça. Implementação de uma rotina para seguir um segmento de reta (FollowLine). Relatório técnico, Faculdade de Engenharia da Universidade do Porto, Porto, 2017.

[33] RoboCup Federation. Objective, 2016. URL: `http://www.robocup.org/objective`.

[34] Paulo Costa. SimTwo, 2015. URL: `https://paginas.fe.up.pt/{~}paco/wiki/index.php?n=Main.SimTwo`.

[35] Vítor Pinto e António Moreira. Apontamentos das aulas teóricas de sistemas robóticos autónomos - cinemática e dinâmica, 2017.

[36] Paul A. Tipler. *Física para cientistas e engenheiros (vol. 1)*. Rio de Janeiro, 5 edição, 2006.

[37] Russel Smith. Open Dynamics Engine, 2007. URL: `http://www.ode.org`.

[38] Relvas Pedro Miguel da Silva Rocha. *Fusão sensorial e cooperação em equipas de robôs móveis*. Mestrado, Universidade do Porto, 2017. URL: `http://hdl.handle.net/10216/105366`.

[39] André Gustavo Conceição, António Moreira, Paulo Costa, Pedro Costa, e Tiago Nascimento. Modeling omnidirectional mobile robots: an approach using SimTwo. Em *Controlo'2012*, página 6, Porto, jul 2012. URL: `http://www.apca.pt/index.php/site/pubdetails/24`.

[40] Richard J. Vaccaro. *Digital Control: A state space approach*. McGraw-Hill, Inc., Singapore, 1 edição, 1995.

[41] Martin Riedmiller e Heinrich Braun. A direct adaptive method for faster backpropagation learning: The RPROP algorithm. *IEEE International Conference on Neural Networks - Conference Proceedings*, 1993-Janua:586–591, 1993. doi:10.1109/ICNN.1993.298623.