VICTOR CHACON CODESSEIRA

CONTROLE DE TRAJETÓRIA DE BARCO AUTÔNOMO USANDO MODEL PREDICTIVE CONTROL

São Paulo
2020

VICTOR CHACON CODESSEIRA

CONTROLE DE TRAJETÓRIA DE BARCO AUTÔNOMO USANDO MODEL PREDICTIVE CONTROL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Mecatrônico.

São Paulo
2020

VICTOR CHACON CODESSEIRA

CONTROLE DE TRAJETÓRIA DE BARCO AUTÔNOMO USANDO MODEL PREDICTIVE CONTROL

Trabalho apresentado à Escola Politécnica da Universidade de São Paulo para obtenção do Título de Engenheiro Mecatrônico.

Orientador:

Prof. Dr. Eduardo Aoun Tannuri

São Paulo
2020

RESUMO

Embarcações de superfícies são, em geral, sub-atuadas, pois apresentam mais graus de liberdade que atuadores, o que acaba gerando diversas dificuldades no controle. O Model Predictive Control (MPC) é uma técnica de controle moderno, que com uma otimização de horizonte de entradas, busca resolver alguns dos problemas que surgem com as técnicas clássicas de controle. O objetivo central desse trabalho é a implementação de um controlador MPC para controle de trajetória de embarcações, que será testado em um simulador desenvolvido pelo Tanque de Provas Numérico (TPN) da USP. Com isso, buscamos chegar em um controle robusto e eficiente, capaz de ser adaptado a diferentes embarcações e situações ambientais.

Palavras-Chave – Model Predictive Control, Embarcação, Controle Moderno, Controle de Trajetória.

ABSTRACT

Water surface vessels are, generally, underactuated, as they have more degrees of freedom than actuators, which lead to several difficulties in controlling them. Model Predictive Control (MPC) is a modern control technique, which seeks to solve some of the problems that arise with classical control techniques by optimizing inputs over a finite period of time. The central goal in this study is implementing a MPC controller for trajectory tracking, which will be tested in a simulator developed by the Numerical Offshore Tank (TPN) of the University of São Paulo. With such controller, we seek to achieve a robust and efficient control, capable of adapting to different vessels and environmental conditions.

Keywords – Model Predictive Control, Vessel, Boat, Modern Control, Trajectory Tracking.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO E DEFINIÇÃO DOS REQUISITOS

1.1 Introdução

O tema do projeto envolve o desenvolvimento de um controlador de barco autônomo. Técnicas clássicas de controle, apesar de serem usadas, não são as mais adequadas para um barco. Isso porque o sistema é sub-atuado, ou seja, tem 3 graus de liberdade (se considerarmos um movimento plano), e geralmente apenas 1 variável de controle, com no máximo 2 (leme e máquina). Assim, técnicas mais avançadas de controle são necessárias.

Uma possível técnica a ser usada é o Model Predictive Control (MPC), que otimiza a resposta do sistema, levando em conta a sua modelagem dinâmica, e a sua resposta ao esforço de controle. Além disso, com essa técnica é possível também levar em conta restrições dos atuadores, por exemplo um ângulo máximo para o leme, e o controlador não vai exceder esses valores, e decidir por um esforço de atuação maior que o possível, levando à saturação dos atuadores.

Assim, o objetivo deste Trabalho de Conclusão é desenvolver um controlador de trajetória para um barco autônomo, em Python, capaz de seguir um caminho dado por pontos chave (*waypoints*), usando o MPC, e testar sua eficácia e viabilidade usando um módulo em Python que simula as equações dinâmicas de um barco desenvolvido no TPN, chamado de PyDyna.

1.2 Objetivos

1.2.1 Primários

- Obter o modelo simplificado de uma embarcação no PyDyna;
- Implementar os métodos Line of Sight (LOS) e o controlador MPC;
- Testar a eficácia do controle de trajetória;
- Testar a robustez do controlador.

1.2.2 Secundários

- Testar controlador em aplicação real-time, para verificar sua viabilidade;
- Implementar algoritmo de Collision Avoidance, capaz de gerar novos *waypoint*s, caso haja um obstáculo no percurso.

1.3 Estado da Arte

O desenvolvimento de controladores de veículos autônomos é algo extremamente estudado hoje em dia, e então, é muito fácil encontrar fontes sobre o assunto. Mais especificamente para o desenvolvimento de controladores MPC para veículos marítimos de superfície, apenas no último mês já aparecem diversos resultados em qualquer dispositivo de busca, como o Google Scholar.

Assim, a quantidade de informações disponíveis é grande. Nesta revisão, a pergunta principal é se é possível usar MPC para controle de trajetória de barcos autônomo em real-time. Para respoder isso, precisamos sobre a modelagem dinâmica de um barco e algoritmos de controle de trajetória e para evitar colisões. Principalmente, precisamos entender o que é MPC e quais os seus tipos, vantagens e desvantagens, e se é possível a sua aplicação em real-time.

1.3.1 Model Predictive Control

O Model Predictive Control (MPC), como o próprio nome diz, é um controle que leva em conta o modelo dinâmico do sistema a ser controlado, ao longo de uma janela de

tempo, para decidir as ações de controle. O MPC surgiu na indústria, foi desenvolvido pela indústria, e aplicado na indústria [1], o que contribui no seu sucesso.

Uma das principais vantagens do MPC é sua capacidade de levar em conta diversos tipos de limites do controle. Outras formas de controle levam em conta as limitações dos estados da planta controlada, mas o MPC consegue levar em conta também limitações dos atuadores, como saturação, velocidade máxima de atuação, além de poder levar em conta também dinâmicas não controladas do sistema, como a rolagem de um barco. Por exemplo, [2] fala sobre um experimento no qual foi feito um controle de modo deslizante, que sofria por problemas de instabilidade quando as condições iniciais eram muito distantes do objetivo, e foi necessário o uso de um controle PID na inicialização. Com o MPC, não há esse problema, pois na hora de definir a atuação ótima, o controlador leva em conta as limitações dos atuadores.

O MPC funciona, basicamente, simulando uma sequência de esforços de atuação, ao longo de um período de tempo chamado de horizonte deslizante (pois esse período avança, junto da simulação). Então, por métodos numéricos, uma função custo é minimizada ao longo de todo o horizonte, e é definida uma sequência de esforços de atuação ótima. Então, apenas o primeiro valor dessa sequência é aplicado, o horizonte deslizante se move, e a simulação é repetida novamente [3]. A estrutura do controlador é mostrada na figura 1.

A função custo define qual tipo de controle deve ser feito. Geralmente, para controle de trajetória, a função custo é o erro de trajetória ao longo do horizonte, como em [4, 5], mas ela pode ser, por exemplo, uma função que minimize a energia utlilizada para a atuação, em sistemas com uma bateria limitada.

1.3.2 Tipos de MPC

O MPC, no entanto, não é uma única estratégia de controle. Existem diversas variações de implementações, cada uma mais adequada para diferentes situações. Em geral, otimizar a resposta do controlador em um horizonte tem um grande custo computacional e, por isso, diversas alternativas foram desenvolvidas ao longo do tempo. O mais comum é o MPC linear, que, em cada instante de tempo, lineariza o sistema em torno do seu estado atual, e otimiza os esforços de atuação. Isso simplifica a otimização consideravelmente, tornando o algoritmo rápido. No entanto, a desvantagem desse método é que, para horizontes de otimização grandes, o sistema se afasta demais da condição de linearização, o que degrada a resposta [3].

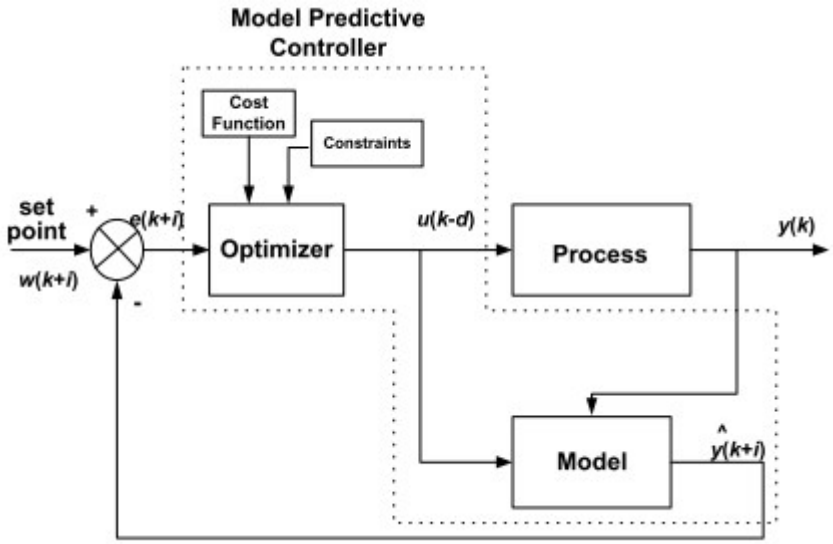


Figura 1: Estrutura geral de um controlador MPC.
Fonte: [1]

Já ao se utilizar diretamente o sistema não-linear, esse problema não surge, porém a otimização é bem mais complexa e demorada, o que impede o seu uso em real-time para sistemas de dinâmica rápida. O chamado Robust MPC, por sua vez, pode ser aplicado em cima do MPC não-linear, e dentro dele se encaixam diversas estratégias para melhorar a robustez do sistema, como por exemplo, a consideração na modelagem de distúrbios não medidos [6], forças externas [7] e a limitação de dinâmicas não diretamente controladas do sistema [8, 9].

Além disso, existe uma outra técnica, chamada de Fast MPC ou Explicit MPC, mostrada em [10], em que o problema de otimização é resolvido analiticamente em regiões, offline, e então o controle se reduz a definir em qual região do controle os estados do sistema estão, e então procurar as ações adequadas numa tabela.

1.3.3 Modelo Dinâmico

Qualquer que seja o tipo de MPC escolhido, o modelo dinâmico do sistema a ser controlado é a parte mais importante do controle. Se o modelo usado estiver errado, o controlador não será capaz de atuar de forma satisfatória, podendo levar até mesmo à instabilidade. Assim, uma das partes mais importantes da implementação do MPC é a modelagem teórica do sistema a ser controlado, e a identificação de seus parâmetros.

Para o modelo dinâmico de um barco, geralmente é utilizado um sistema sub-atuado, de 3 graus de liberdade e apenas 1 ou 2 atuadores, o que é um dos motivos do controle ser dificultado. Algumas modelagens mais complexas, como em [8], levam em conta mais

graus de liberdade, como a rolagem, o que dificulta ainda mais o desenvolvimento do controlador. Uma possível modelagem é dada em [11], junto com os parâmetros necessários para a simulação do CyberShip II. Esse modelo é comumente usado como *benchmark* para sistemas de controle.

Uma possível forma de aproximar o modelo de um sistema dinâmico é pela implementação de um filtro de Kalman estendido, que além dos estados do sistema, também aproxima os parâmetros do modelo, como feito em [12], na qual o filtro é usado para aproximar os parâmetros de uma embarcação para um modelo de Nomoto de segunda ordem.

1.3.4 Trajectory-Tracking e Collision Avoidance

O MPC, ou qualquer outro tipo de controle, serve para levar o sistema ao estado desejado. No caso de um barco, o tipo mais provável de controle para se fazer é o controle de trajetória. Para o controle de trajetória, o controlador recebe um rumo e, possivelmente, uma velocidade, e deve colocar o sistema nesse estado.

Geralmente, a trajetória é dada por uma sequência de pontos, que define o caminho a ser tomado pela embarcação. No caso mais simples, o sistema simplesmente define um rumo entre a posição atual do barco e o próximo ponto, e, ao chegar nesse ponto, segue em direção ao próximo. No entanto, esse não é o caminho ótimo, nem em questão de velocidade nem em energia. Assim, uma possível melhoria é definir um raio em torno do próximo ponto. Assim que a embarcação estiver dentro desse raio, o objetivo já passa a ser o próximo ponto [13].

Mais avançado ainda, e o que é mais utilizado, é um mecanismo de Line of Sight (LOS). Nesse método, é criada uma linha entre o objetivo anterior e o atual do barco. Então, o controlador acha o ponto mais próximo dessa trajetória, em relação a sua posição atual, e define como rumo uma posição mais a frente, por uma distância chamada de *lookahead*. Assim, o controlador consegue seguir a trajetória de forma mais suave e eficiente. Esse procedimento está ilustrado na Figura 2. Uma outra alternativa, que serve para manter o rumo mesmo com perturbações constantes (correntes do mar, por exemplo), é a adoção de uma ação integral no ângulo dado pelo LOS [5].

Outro problema enfrentado por veículos autônomos são as colisões. Assim, pode ser implementado algum tipo de Collision Avoidance (COLAV), de forma que o sistema possa responder a obstáculos no seu caminho. O algoritmo de COLAV pode ser de 2

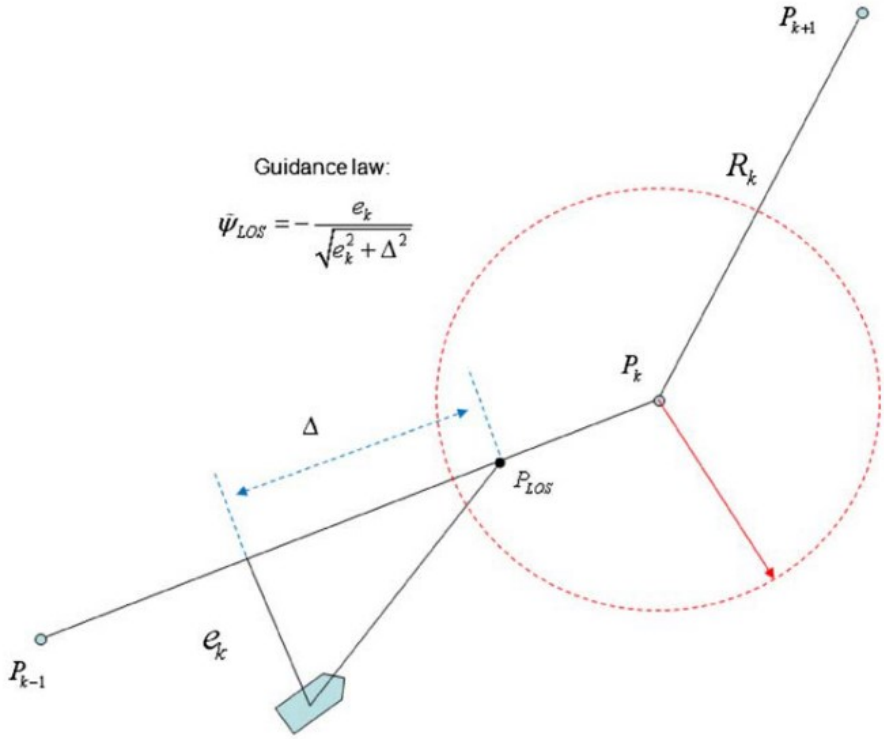


Figura 2: Exemplo de LOS.
Fonte: [4]

tipos. Ele pode ser estático, ou seja, existem obstáculos na trajetória pré-definidos. Nesse caso, o algoritmo deve simplesmente definir novos pontos de trajetória, em que não haja obstáculos, antes mesmo de passar para o controlador. O outro caso é o dinâmico, em que, durante o funcionamento, por meio de sensores, a embarcação detecta um obstáculo. Nesse caso, o sistema deve responder adequadamente, de acordo com as normas internacionais de navegação, para evitar colisões.

1.3.5 Conclusão da Revisão

Para o desenvolvimento de um controlador, é necessário, primeiro, obter um modelo dinâmico simplificado da simulação executada no PyDyna. Isso porque, além do módulo não suportar as funcionalidades necessárias para o MPC, criar um controlador assumindo perfeito conhecimento da embarcação, o que seria feito pelo acesso aos estados do simulador, não é representativo da situação real. Assim, será utilizado o método do filtro de Kalman estendido, como em [12]. Desta forma, a embarcação simulada através do simulador PyDyna é considerada uma "caixa preta", cujos parâmetros serão identificados de forma on-line pelo estimador baseado em Filtro de Kalman.

Como mecanismo de Path-Tracking, a melhor solução é o LOS, por não ter grandes

dificuldades na implementação ou custo computacional, e apresentar uma grande melhoria na capacidade do sistema se manter na trajetória.

Como tipo de MPC, o MPC Linear é o mais fácil de ser implementado e com menor custo computacional, e caso ele não seja suficiente, é possível ir para o MPC não-linear. As embarcações para o qual o controlador será desenvolvido têm dinâmicas relativamente lentas, podendo ter uma frequência de amostragem da ordem de 1 Hz. Assim, será usado diretamente o MPC não-linear, com saturação dos atuadores. Isso porque o controlador linear, além de menos preciso, é mais sensível à definição dos parâmetros, como mostrado em [3], e dificultaria a implementação.

1.4 Requisitos

Os requisitos de projeto envolvem basicamente as características desejadas e a eficácia do controlador. Além disso, temos os requisitos impostos no controlador por limitações físicas.

1.4.1 Atuação

- Ângulo de leme menor que 30º e propulsor limitado por sua potência máxima;
- Variação entre 2 instantes de tempo consecutivos do esforço de atuação limitada para ambos os atuadores. Esse limite ainda será definido pelas características da embarcação a ser controlada, pela velocidade máxima do atuador, e também por otimização energética.

1.4.2 Desempenho

Sendo B a largura do navio (bordo), e L o seu comprimento, temos:

- Ao seguir uma trajetória reta, a embarcação deve se manter a uma distância de no máximo 0.1B da trajetória desejada, como mostrado na Figura 3;
- Trechos de curva são considerados dentro de um raio de $2L$ de um *waypoint*. Nesse raio, não há a restrição acima, pois é possível otimizar o caminho usando LOS. Na Figura 4, o círculo em azul representa o trecho do percurso no qual as restrições de controle de trajetória são relaxadas.

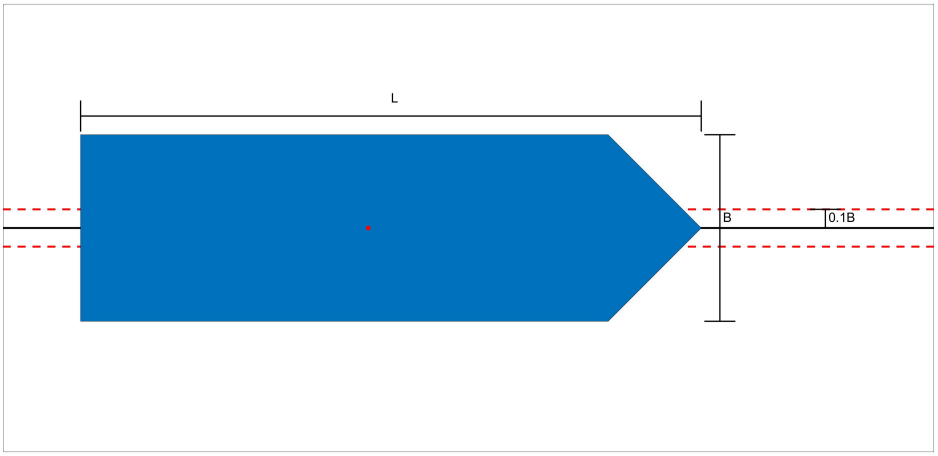


Figura 3: Limite de desempenho na reta
Fonte: Própria

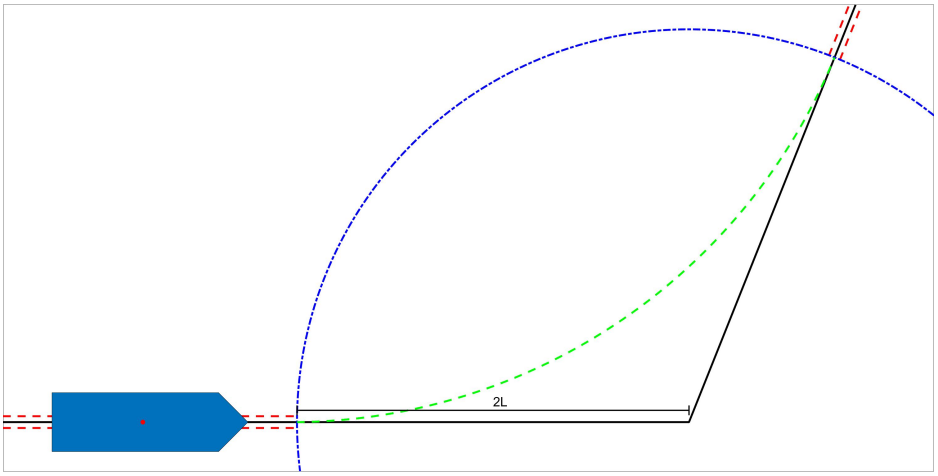


Figura 4: Raio de trechos curvos
Fonte: Própria

1.4.3 Robustez

O controlador deve ser capaz de manter a embarcação dentro dos requisitos de desempenho mesmo com distúrbios, na forma de:

- Correntes marítimas de até 1 nó;
- Ventos de até 20 nós;
- Ondas de até 2 metros;

1.4.4 Computação

- O controle deve ser capaz de atuar em tempo real, ou seja, o tempo que o controlador leva para calcular os esforços de atuação deve ser menor que o tempo de amostragem.

1.4.5 Transiente

- Se a embarcação estiver fora da faixa de desempenho na condição inicial, a uma distância h, perpendicular à trajetória, ela deve ser capaz de atingir a faixa desejada dentro de uma distância, ao longo da trajetória, de M*h, com M aproximadamente 2, ainda a definir com mais precisão. Esse requisito é melhor mostrado na Figura 5. As barras em azul representam a distância dentro da qual o controlador deve atingir a faixa desejada.

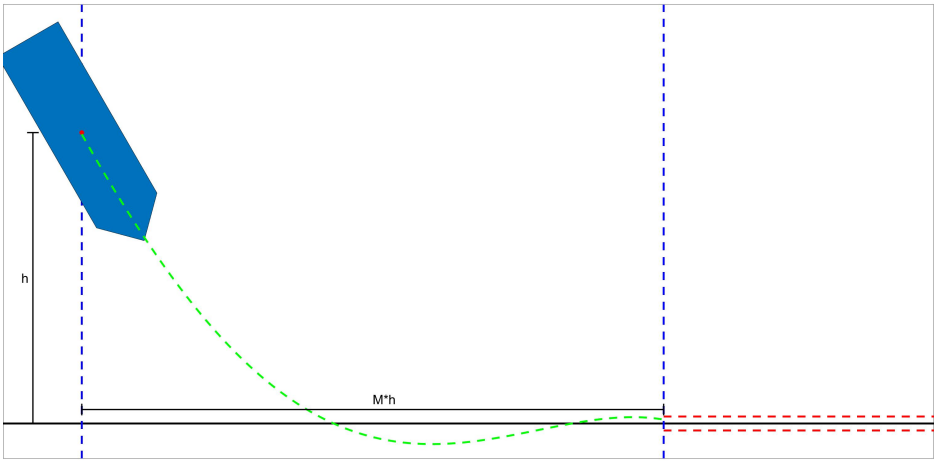


Figura 5: Correção de erro inicial
Fonte: Própria

2 DESENVOLVIMENTO

2.1 Obtenção do modelo estimado

O primeiro passo para o desenvolvimento de um controlador MPC é a obtenção de um modelo dinâmico que represente o sistema. Para isso, foi utilizado o método do Filtro Estendido de Kalman, como mostrado por [12]. Nesse artigo, é adotada apenas a dinâmica rotacional de uma embarcação, mas para quesitos de testes, foi implementado um modelo igual ao do artigo. Assim, temos um modelo dinâmico da forma:

$$\psi^{(3)} = \alpha_1 \dot{\psi}^3 + \alpha_2 \dot{\psi} + \alpha_3 \ddot{\psi} + \beta_1 \delta_\mathrm{R} + \beta_2 \dot{\delta}_\mathrm{R}$$

No qual ψ é o ângulo da embarcação em relação ao sistema global, e δ_R é o ângulo do leme (entrada do sistema). α_1, α_2, α_3, β_1 e β_2 são os parâmetros do controlador. Esse é um modelo similar ao modelo de Nomoto de segunda ordem [13], com uma extensão de permitir um parâmetro extra, multiplicando ψ^3. Assim, temos um filtro de Kalman estendido com 8 estados, que são:

$$x^\mathrm{T}(t) = \left[\psi(t)\ \dot{\psi}(t)\ \ddot{\psi}(t)\ \alpha_1(t)\ \alpha_2(t)\ \alpha_3(t)\ \beta_1(t)\ \beta_2(t)\right]$$

Assim como no artigo, para a convergência, foi adotada uma entrada com grandes variações, com o ângulo de leme alterado de forma aleatória, a cada 10 segundos, de forma que a embarcação precisasse realizar manobras bruscas, e fosse possível captar as dinâmicas mais complexas do sistema. Após a implementação, o funcionamento do filtro para acompanhar os estados do sistema era o esperado, mas ocorreram alguns problemas na convergência de alguns dos parâmetros do modelo. Um possível motivo para esse problema, é que, no artigo citado, o modelo a ser estimado é um modelo com exatamente as mesmas equações dinâmicas que as do filtro. Isso não ocorre no caso do PyDyna, que apresenta uma dinâmica muito mais complexa, que leva em conta todos os graus de liberdade da embarcação. Assim, foram testados outros tipos de modelos. Primeiro, foi

testado o modelo de Nomoto de segunda ordem, que corresponde à EDO:

$$\psi^{(3)} = \alpha_1\dot{\psi} + \alpha_2\ddot{\psi} + \beta_1\delta_\mathrm{R} + \beta_2\dot{\delta}_\mathrm{R}$$

Novamente, alguns dos parâmetros do modelo não convergiram, e então foi testado o modelo de Nomoto de primeira ordem. Esse modelo corresponde à equação:

$$\ddot{\psi} = -\frac{1}{T}\dot{\psi} + \frac{K}{T}\delta$$

Em que K e T dependem dos parâmetros da embarcação, como massa e momento de inércia. Para simplificação da notação, a equação foi adotada na forma:

$$\ddot{\psi} = a\dot{\psi} + b\delta$$

Em que $a = -\frac{1}{T}$ e $b = \frac{K}{T}$. Com esse modelo, não apareceram problemas na convergência dos parâmetros. Além disso, foi acoplado ao filtro de Kalman um simulador que, a partir dos valores dos parâmetros dados pelo filtro em cada instante, com a equação do modelo, simula, em paralelo à simulação do PyDyna, o sistema, e os valores dos estados dados pelos 2 podem ser comparados, para ver se a aproximação dos parâmetros é eficaz. O resultado dessa dinâmica pode ser visto nos 2 primeiros gráficos da figura 7. Como os resultados do modelo de primeira ordem se mostraram satisfatórios, esse modelo foi mantido como a dinâmica rotacional.

Para a dinâmica longitudinal, foi adotado uma equação dinâmica da forma:

$$\dot{u} = cu + dr^2$$

Em que u é a velocidade linear da embarcação, e r é a rotação do propulsor, uma das entradas do sistema dinâmico. Essa forma foi adotada pois c faz papel do arrasto linearizado, multiplicando a velocidade. Já para d, de acordo com a segunda Lei de Newton, a aceleração ($\dot{u}$) deve ser proporcional à força gerada pelo propulsor. A força gerada por um propulsor, por sua vez, é proporcional ao quadrado da sua rotação, e por isso, temos o termo r^2.

Juntando as duas dinâmicas, temos um sistema da forma:

$$\ddot{\psi} = a\dot{\psi} + b\delta$$
$$\dot{u} = cu + dr^2$$

A formulação do Filtro de Kalman Estendido fica, então:

$$x^{\mathrm{T}}(t) = \left[\psi(t)\ \dot{\psi}(t)\ u(t)\ a(t)\ b(t)\ c(t)\ d(t)\right]$$

$$u^{\mathrm{T}}(t) = [\delta_{\mathrm{R}}(t)\ r(t)]$$

$$h^{\mathrm{T}}(x) = [x_1\ x_2\ x_3]$$

$$\dot{x}(t) = f(x,u) = \begin{bmatrix} x_2 \\ x_4 \cdot x_2 + x_5 \cdot u_1 \\ x_6 \cdot x_3 + x_7 \cdot u_2^2 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Sendo $\hat{x}_k$ a predição do estado do filtro de Kalman no instante k, e x_k os valores reais dos estados, ao discretizar o sistema com passo Δ_t, seguimos essa metodologia, para cada passo k.

Predições a priori

$$\hat{x}_{k|k-1} = \hat{x}_{k-1|k-1} + f(\hat{x}_{k-1|k-1}, u_k)\Delta_t$$

$$P_{k|k-1} = P_{k-1|k-1} + (F_k P_{k-1|k-1} + P_{k-1|k-1} F_k + Q_k)\Delta_t$$

Cálculo dos erros

$$z_k = h(x_k)$$

$$\tilde{y}_k = z_k - h(\hat{x}_{k|k-1})$$

$$S_k = H_k P_{k|k-1} H_k^{\mathrm{T}} + R_k$$

Obtenção do ganho de Kalman Ótimo

$$K_k = P_{k|k-1} H_k^{\mathrm{T}} S_k^{-} 1$$

Atualização das predições

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k$$

$$P_{k|k} = (I - K_k H_k) P_{k|k-1}$$

Sendo

$$F_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1|k-1},u_k} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & x_4 & 0 & x_2 & u_1 & 0 & 0 \\ 0 & 0 & x_6 & 0 & 0 & x_3 & u_2^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k|k-1}} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Usando esse modelo, com a entrada de leme e máquina, variáveis de forma aleatória a cada 10 segundos, podemos ver os parâmetros obtidos pelo filtro, na figura 6, e os estados aproximados, na figura 7. Podemos perceber que, apesar dos parâmetro não terem convergido para um valor, principalmente nos casos de a e b, a simulação a partir dos modelos obtidos acima é suficiente para aproximar os estados. No começo, temos alguns erros grandes devido a diferenças nos valores iniciais das variáveis, e também um pico no parâmetro c. O único estado cuja aproximação foi pior foi ψ, devido ao seu caráter de integrar os erros iniciais em $\dot{\psi}$, mas esse erro não é importante, pois o estado ψ não entra na dinâmica da embarcação, e não afetará o controlador. O ângulo ψ só é importante para o mecanismo de LOS, que obterá esse dado diretamente da embarcação.

Foi testado, também, uma simulação com um modelo com parâmetros fixos, cujos parâmetros são aproximadamente os valores médios dos obtidos na figura 6, após a variação inicial, ou seja, $[-0.015, -0.0005, -0.002, 0.015]$. Os estados da simulação também foram inicializados de acordo com os do sistema. Como podemos ver na figura 8, os resultados são bem piores, com maiores diferenças nos estados $\dot{\psi}$ e u, cuja precisão importa mais. A diferença, principalmente na dinâmica rotacional, provavelmente acontece pois a dinâmica é mais complexa que a adotada(de primeira ordem), e os parâmetros precisam variar para se adequar aos estados.

Por último, foi testado novamente o modelo com a simulação usando parâmetros decididos a cada instante, mas já inicializando os parâmetros nos parâmetros obtidos anteriormente. Como podemos ver na figura 9, os resultados foram ainda melhores, pois o sistema ainda foi capaz de acompanhar os estados da embarcação, mas não apresentou erros tão grandes nos instantes iniciais.

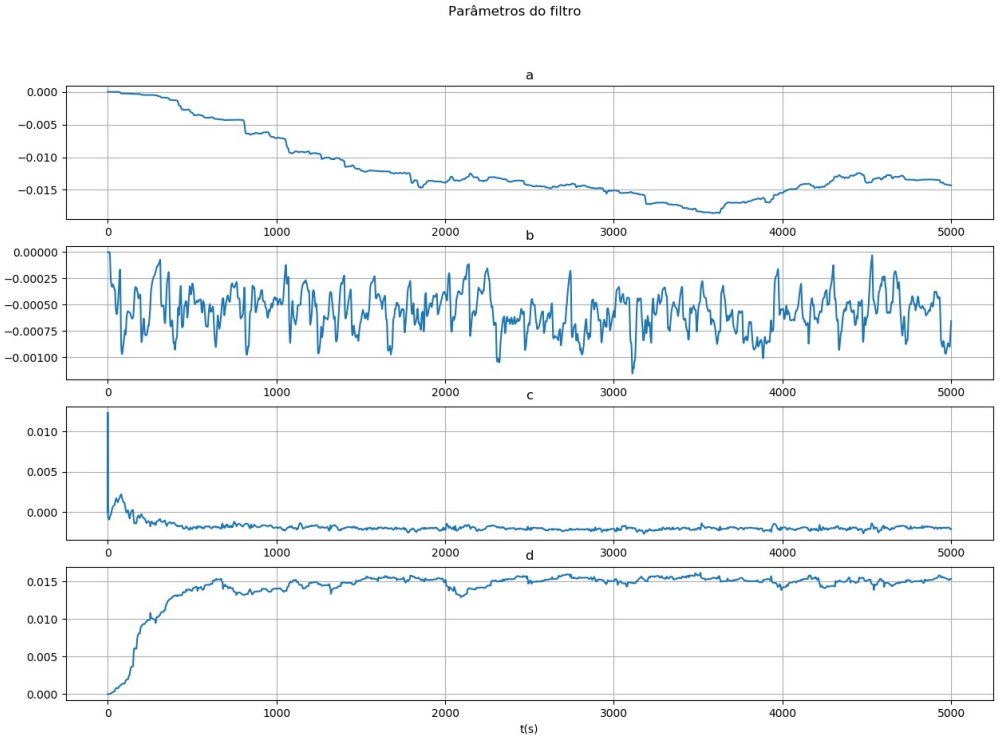


Figura 6: Parâmetros obtidos com o filtro

Figura 7: Estados com o modulo PyDyna ("Real"), e com a simulacão paralela com modo de primeira ordem ("Simulação")

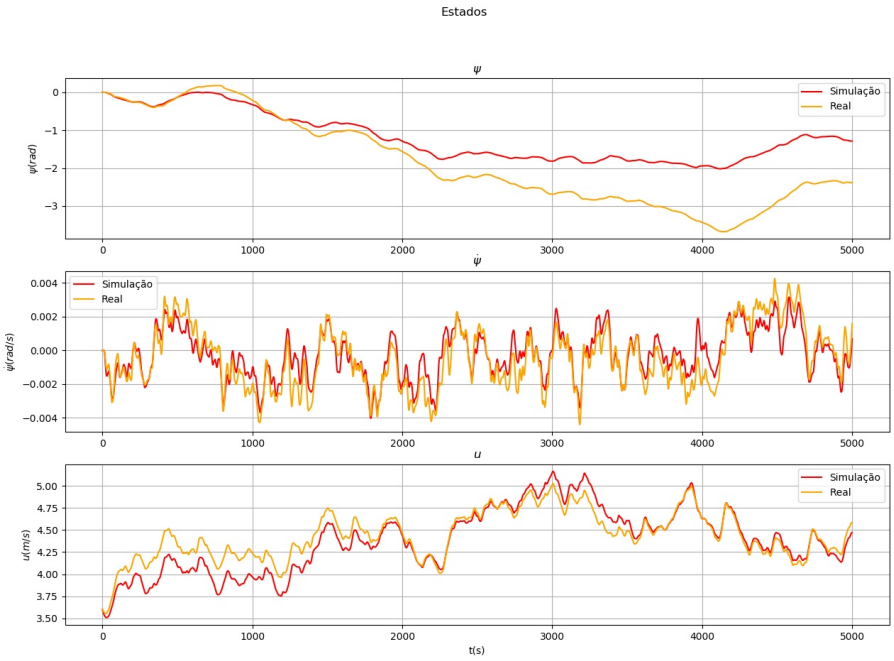


Figura 8: Estados com o modulo PyDyna ("Real"), e com a simulacão de parâmetros fixos com modo de primeira ordem ("Simulação")

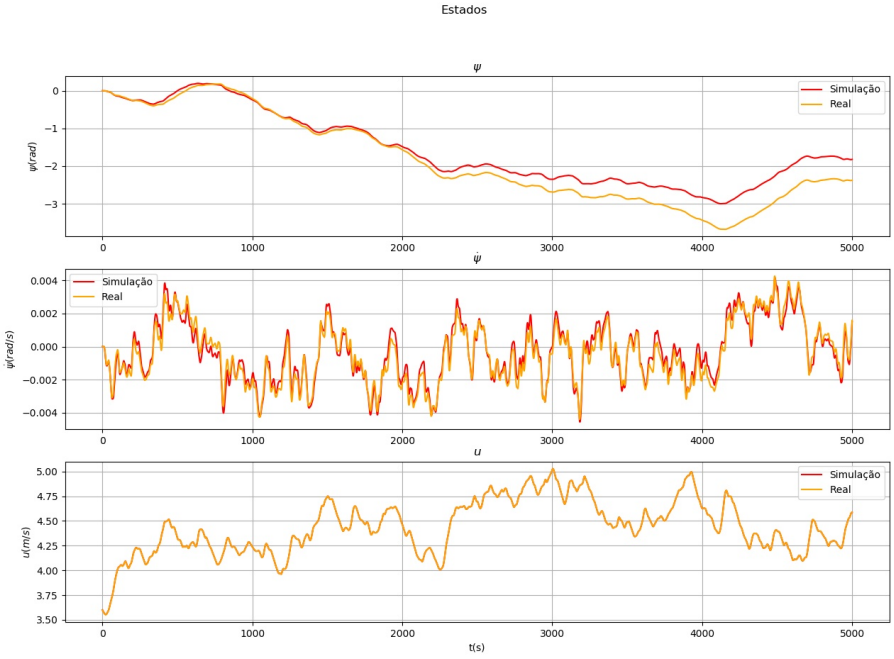


Figura 9: Estados com o modulo PyDyna ("Real"), e com a simulacão paralela com modo de primeira ordem, pré-inicializada ("Simulação")

Assim, apesar de não ter sido obtido um modelo fechado para a embarcação, temos uma forma de estimar os parâmetros, a partir do filtro estendido de Kalman. Como o MPC calcula a próxima ação, a partir do modelo, a cada instante de tempo, podemos ter esse estimador sendo executado em paralelo ao controlador, e usando esses parâmetros para o controlador. Assim, teremos um estimador de parâmetros online, ao invés de obter todos os parâmetros previamente. Isso, além das vantagens vistas nos resultados já obtidos, também permitirá que o sistema responda de forma melhor a distúrbios, como correntes e ventos, já que os parâmetros do controlador vão se alterar nessas condições.

2.2 Estrutura do Controlador

Como o controlador será adaptativo, temos uma estrutura diferente de um controlador MPC padrão, como podemos ver na figura 10. O mecanismo de LOS calcula o ângulo desejado ψ_d, a partir das coordenadas cartesianas da embarcação e da lista de Waypoints. Esse ângulo é comparado com o ângulo real, e o erro é usado como entrada para o controlador. O MPC, então, dá as entradas para o navio (aqui representado pelo modelo simulado no PyDyna). Os estados do navio (PyDyna), e as entradas do sistema são enviados ao filtro de Kalman Estendido, o Estimador de Parâmetros, que então, usa esses parâmetros para alterar a dinâmica do controlador.

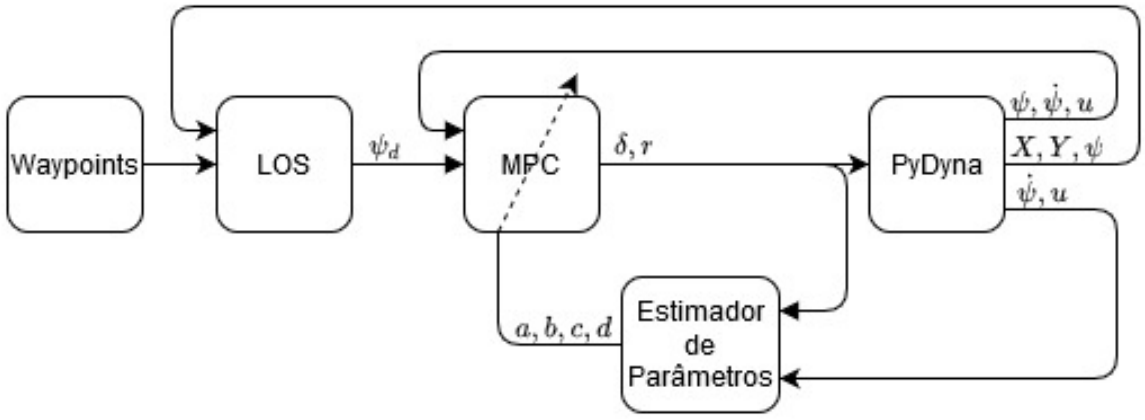


Figura 10: Diagrama de Blocos do Sistema

2.3 Implementação do Line of Sight

A maior utilidade do Line of Sight (LOS), é que, com ele, conseguimos converter uma sequência de waypoints com 2 coordenadas (X e Y) em apenas uma, ψ_d, o ângulo desejado. Podemos ver alguns exemplos gerados com o programa implementado na figura 11. Em 11a, o waypoint alvo é o segundo, em $(3000, 0)$. Podemos ver que o LOS calcula uma direção intermediária, que não segue diretamente para o waypoint (o que o manteria fora do rumo por toda a trajetória), mas também não segue diretamente para a reta que conecta o waypoint anterior e o atual. Em 11b, podemos ver como, ao ficar em um certo raio do waypoint alvo, o LOS segue para o próximo waypoint, de forma a suavizar a transição. Para esses testes, a distância de lookahead foi adotada como 4 vezes o comprimento da embarcação, e o raio de troca de waypoint, 2 vezes esse comprimento.

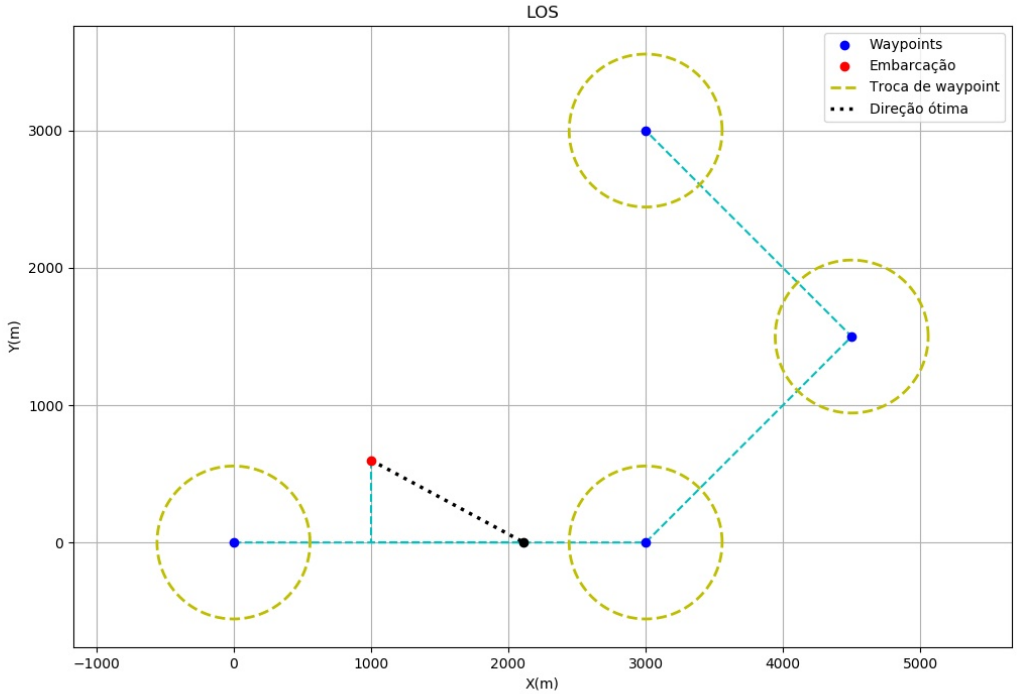
2.4 Implementação do Controlador MPC

Usando o modelo simplificado obtido anteriormente, foi possível implementar um controlador MPC. Para isso, foi utilizada a função *minimize* da biblioteca *scipy*. Essa função permite achar os valores das entradas que minimizam o valor de uma certa função. Para o MPC, a função a ser minimizada é a função de custo, que deve levar em conta essas entradas e o modelo do sistema para calcular um determinado custo.

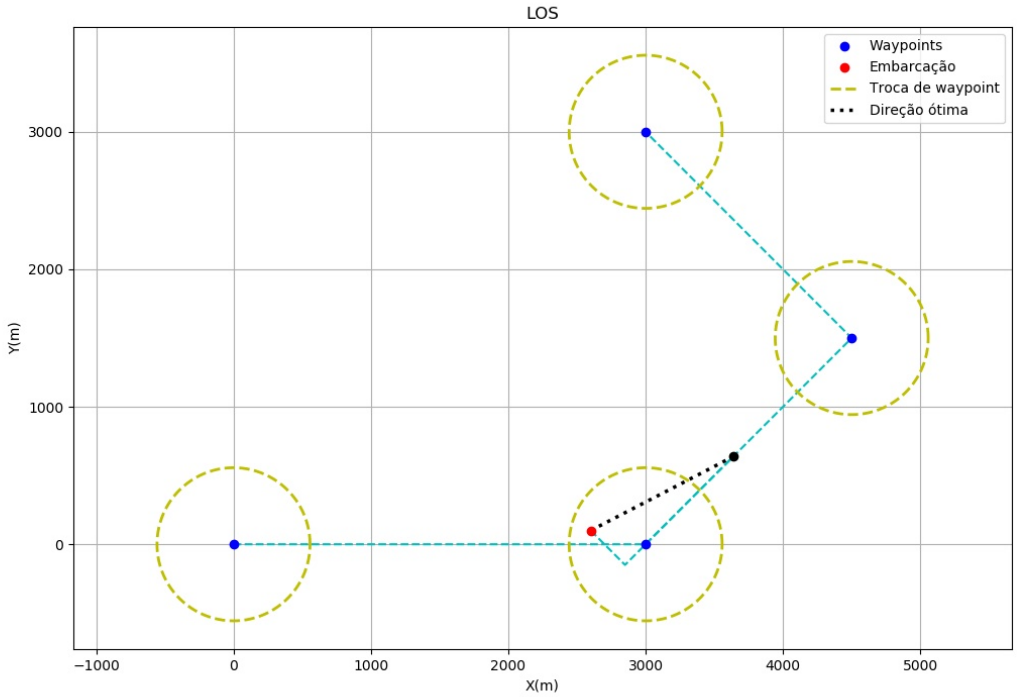
No caso, o modelo do sistema foi adotado como explicado anteriormente, e para uma certa entrada, a função de custo integra os estados do sistema ao longo de um certo horizonte, e calcula o custo para essas entradas, que, para os testes iniciais, foi adotado como uma soma ponderada dos quadrados do erro de ψ ao longo do horizonte, e também da sua derivada $\dot{\psi}$. Assim, estamos otimizando o sistema, não só para convergir ao valor desejado, mas também manter o valor da derivada controlado, para evitar problemas de instabilidade e overshoot. Para os testes iniciais, também foi adotado um alvo estático, de 0.01 rad.

Podemos ver, na figura 12, que com o uso do controlador, o sistema converge para a posição desejada, e então, vemos o controlador diminuindo também o valor da derivada. Sem esse controle da derivada, o sistema só começa a tentar reverter depois de passar no alvo, e teríamos um comportamento oscilatório.

Para uma situação mais real, o controlador foi testado com uma sequência de waypoints, que com o uso do LOS, dão a direção desejada da embarcação em determinado



(a) Alvo é o segundo waypoint



(b) Alvo é o terceiro waypoint

Figura 11: Exemplos de LOS

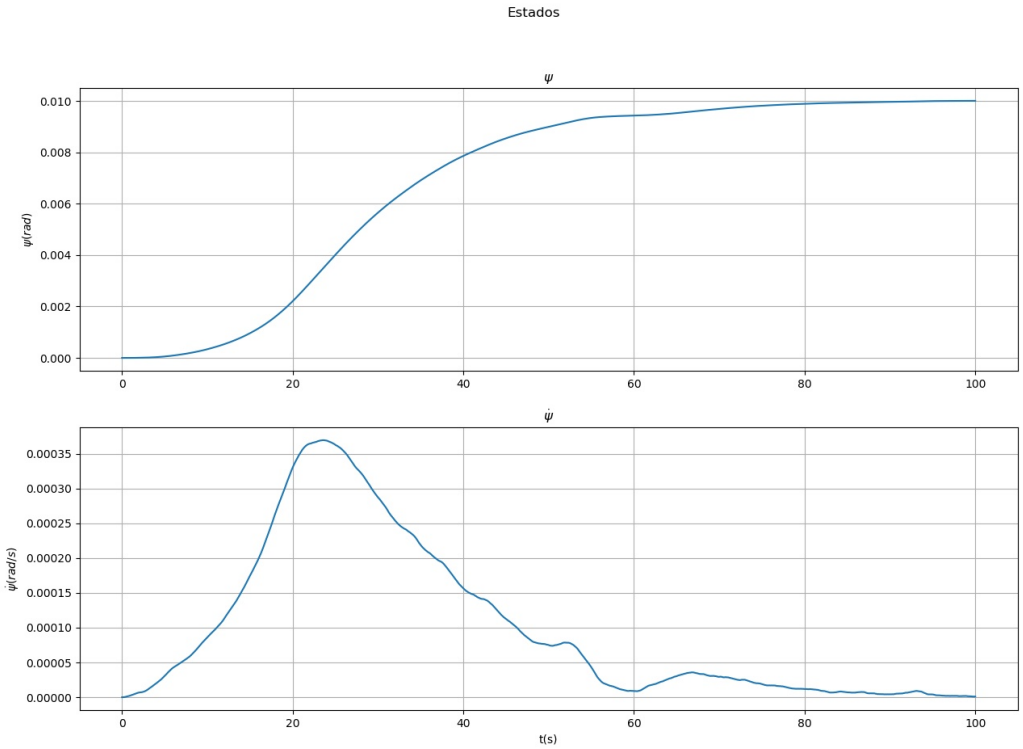


Figura 12: Resultado do controle MPC

momento. Além disso, foram adicionados mais 2 parâmetros na função de otimização do controlador, de forma que ele trabalhasse na minimização do erro, da derivada do erro, do valor da entrada do leme e da sua derivada. Com esses parâmetros, conseguimos um melhor controle, evidenciado tanto pela trajetória seguida pela embarcação, como pelos valores das entradas. Como podemos ver na figura 13, o controlador é capaz de seguir a trajetória imposta pelos waypoints. Já na figura 14, vemos as entradas que a velocidade do sistema, o erro de acompanhamento, e a entrada do leme. Vale ressaltar que essa é apenas a entrada imposta pelo controlador, e a posição real do leme respeita a sua dinâmica, sem essas variações bruscas.

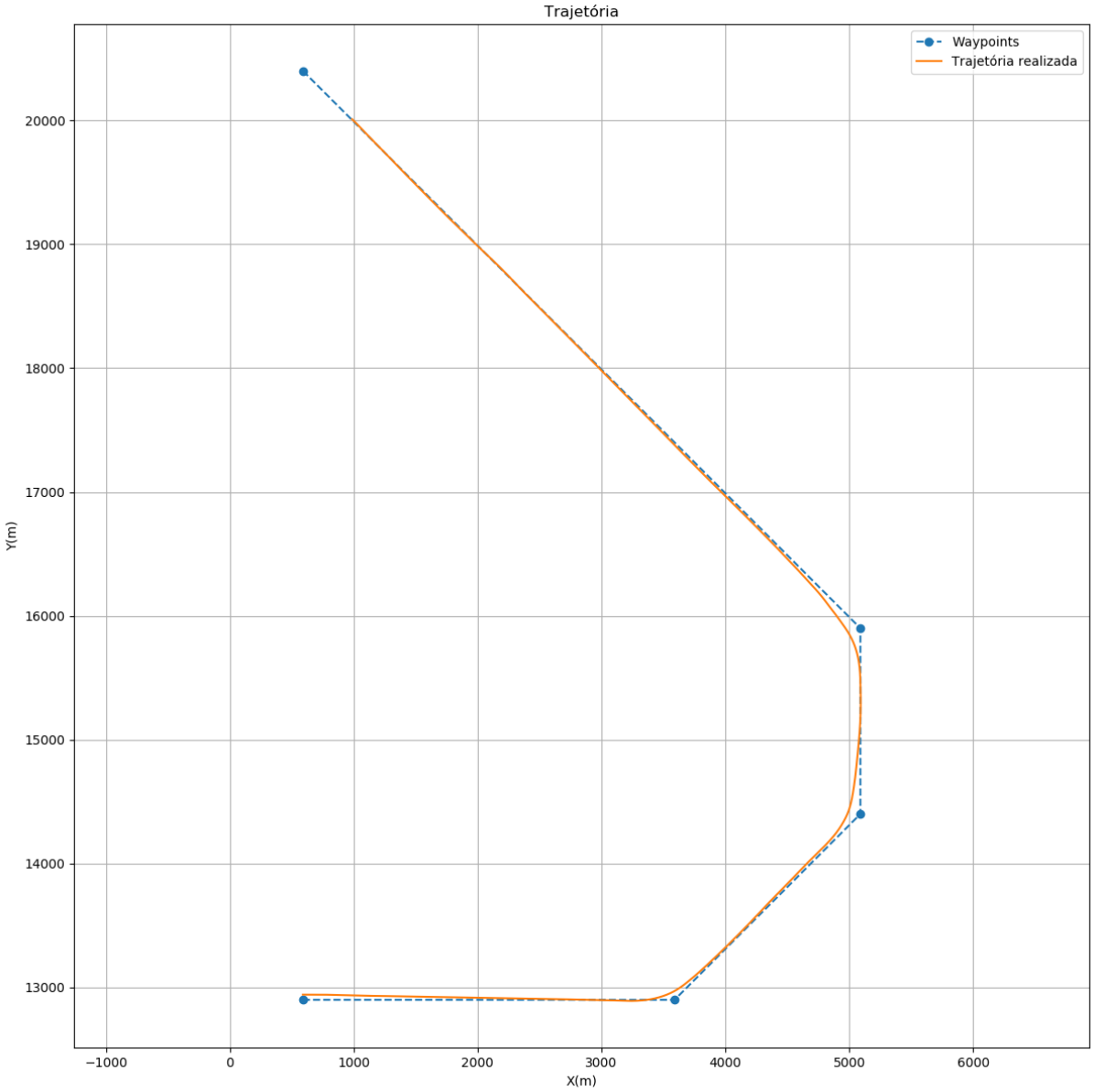


Figura 13: Trajetória realizada pelo caso de teste

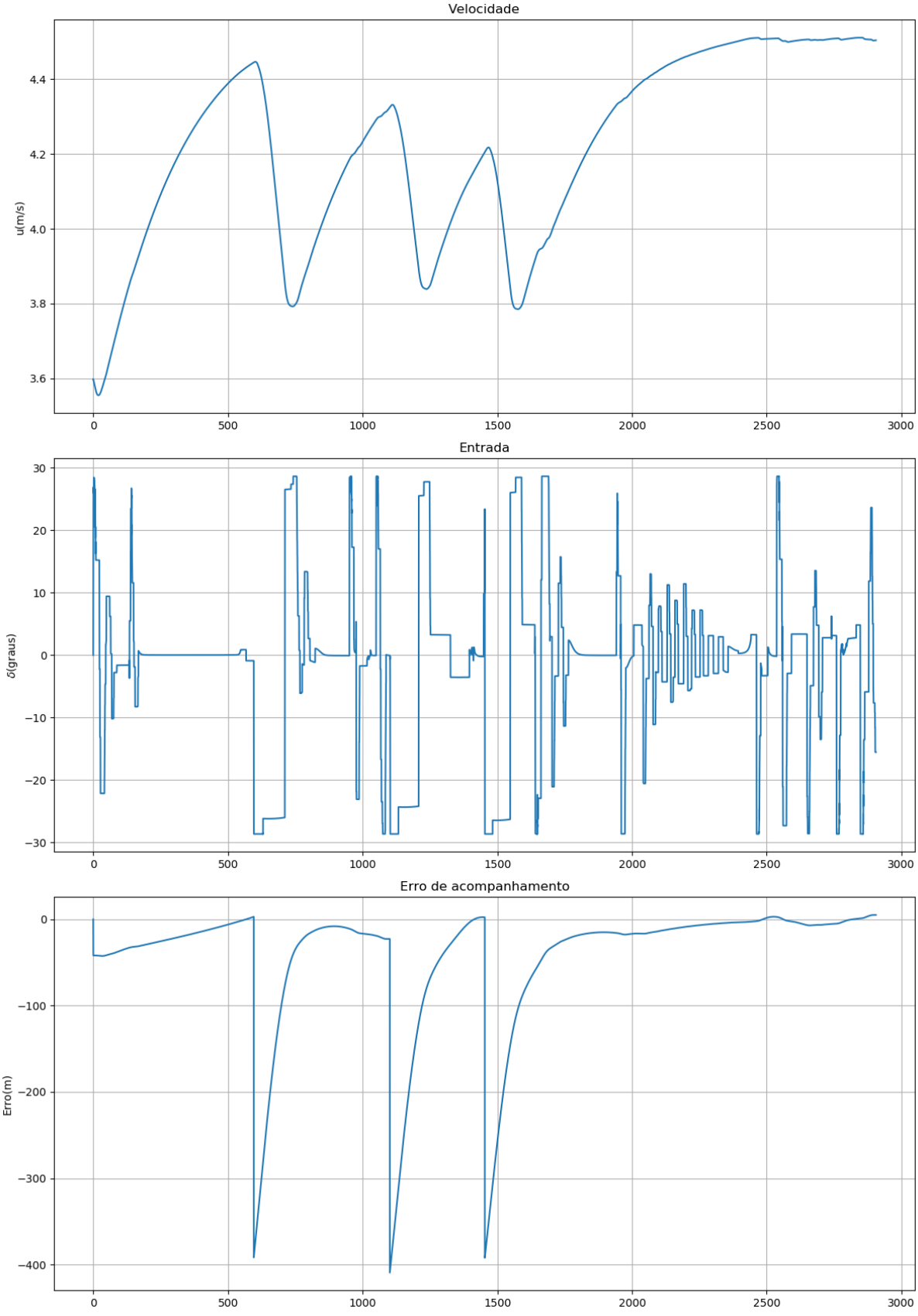


Figura 14: Velocidade, entrada e erro no caso de teste

3 VALIDAÇÃO DE RESULTADOS

3.1 Apresentação de resultados

Para a validação de resultados, utilizamos casos reais, de uma embarcação entrando no porto de Paranaguá, a trajetória a ser seguida pode ser vista na figura 15. Essa trajetória foi simulada para diferentes condições ambientais, como dado pela tabela 1.

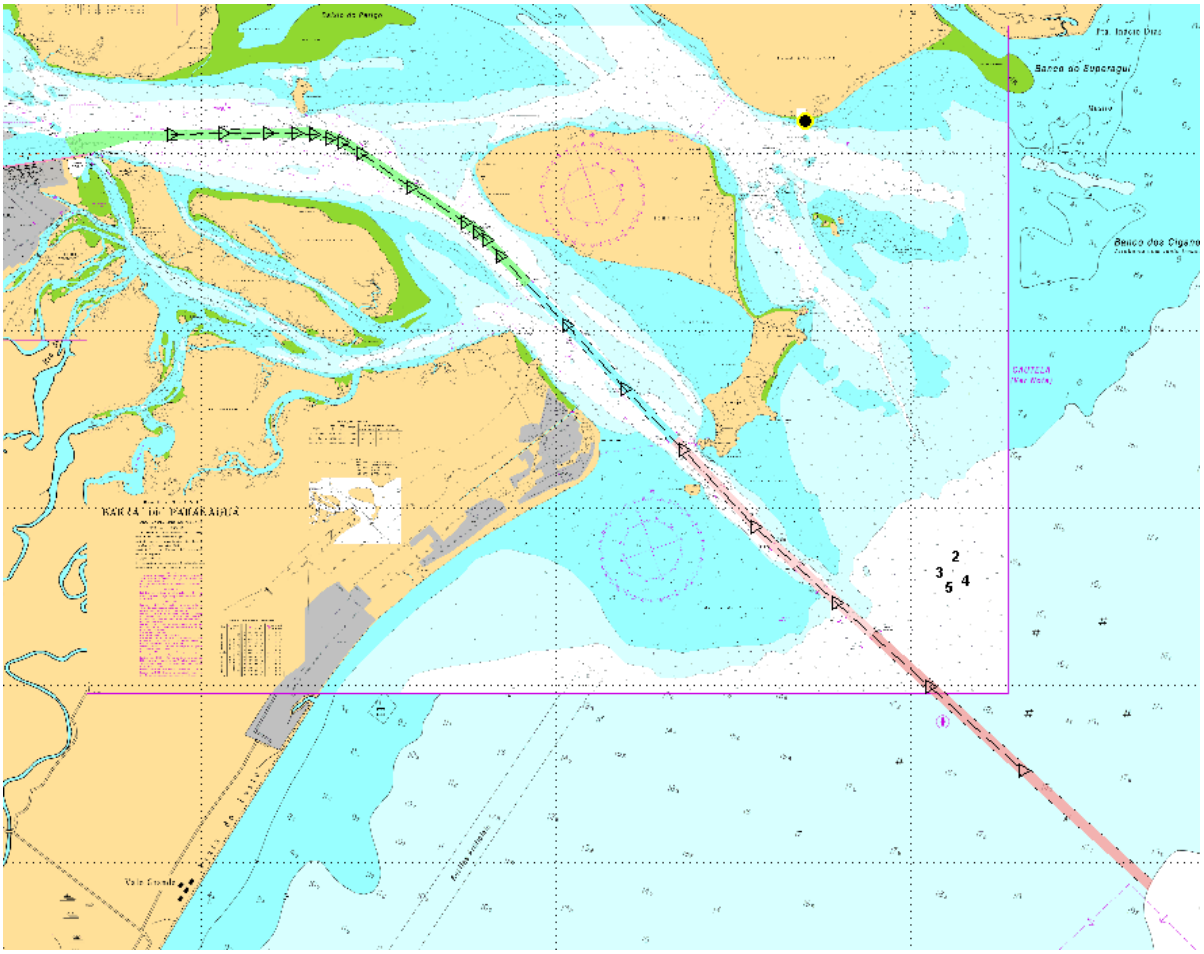


Figura 15: Trajetória a ser seguida

A embarcação utilizada para os testes é diferente da utilizada anteriormente. Assim, foi necessário treinar novamente o filtro de Kalman, para obter novos parâmetros para o

Caso	Corrente	Vento	Onda
1	Enchente - 2kn	SE - 21kn	2m - 12.2s - SE
2	Vazante - 2kn	SW - 21kn	2m - 8.8s - S
3	Estofo - 0kn	NE - 21kn	2m - 9.2s - E
4	Estofo - 0kn	E - 21kn	2m - 11.1s - ESE
5	Estofo - 0kn	S - 21kn	2m - 12.3s - SSE

Tabela 1: Casos simulados

modelo dinâmico da embarcação. Na figura 16, podemos ver que a trajetória seguida pela embarcação nos 5 casos é praticamente idêntica, e coincide fortemente com a trajetória definida pelos waypoints. Nas figuras 17 a 19, temos as trajetórias dos 5 casos sobrepostas sobre a carta náutica do porto, considerando o tamanho da embarcação. Podemos ver a velocidade, a entrada e o erro nas figuras 20 a 24. Apesar de termos entradas bastante irregulares, isso não influencia de forma considerável a dinâmica do erro, cujos picos equivalem aos pontos nos quais existe uma troca de waypoint alvo.

3.2 Requisitos

Os requisitos de atuação são levados em conta na otimização realizada pelo controlador, então eles são automaticamente atendidos. Analisando os dados da saída, estamos também atendendo aos requisitos de desempenho. Os testes foram realizados com condições ambientais até mais severas que as previstas nos requisitos, então eles também foram atendidos. Por último, o sistema também atende aos requisitos de computação real-time, como pode ser visto na tabela 2. No sistema simulado, o atuador aplica a sua entrada a cada 0.5 segundos. Assim, podemos ver que mesmo apenas nos piores casos (menos de 1% dos passos), temos um tempo de cálculo maior do que isso. Esses passos poderiam ser ignorados, ou se necessário, poderíamos aplicar entradas numa cadência menor, e ajustar os parâmetros do controlador, caso se mostrasse necessário.

Estatística	Tempo (s)
Média	0.018
90%	0.045
99%	0.225
Máxima	0.585

Tabela 2: Estatísticas de tempo de execução do controlador

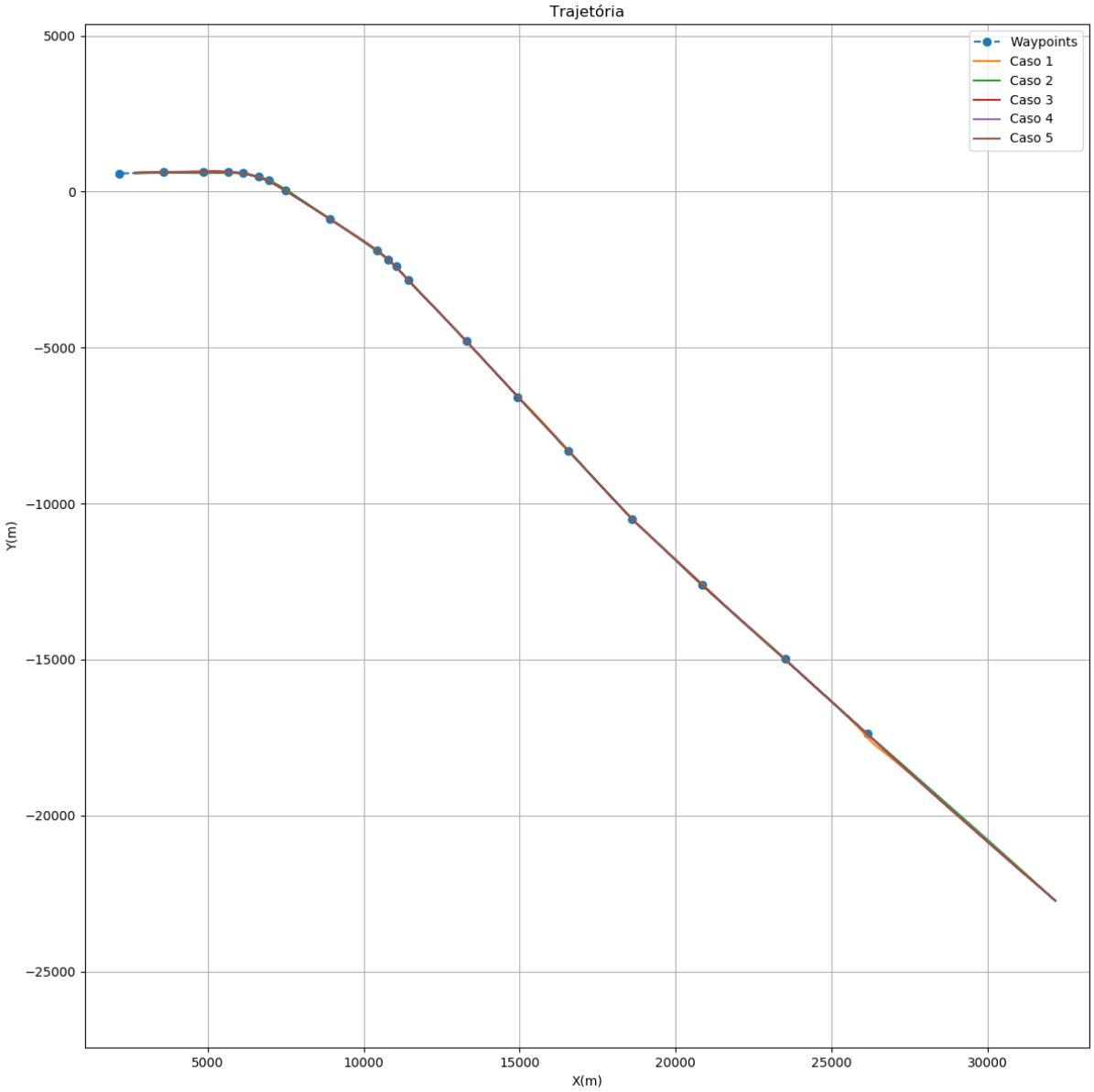


Figura 16: Trajetórias seguidas em todos os casos de teste

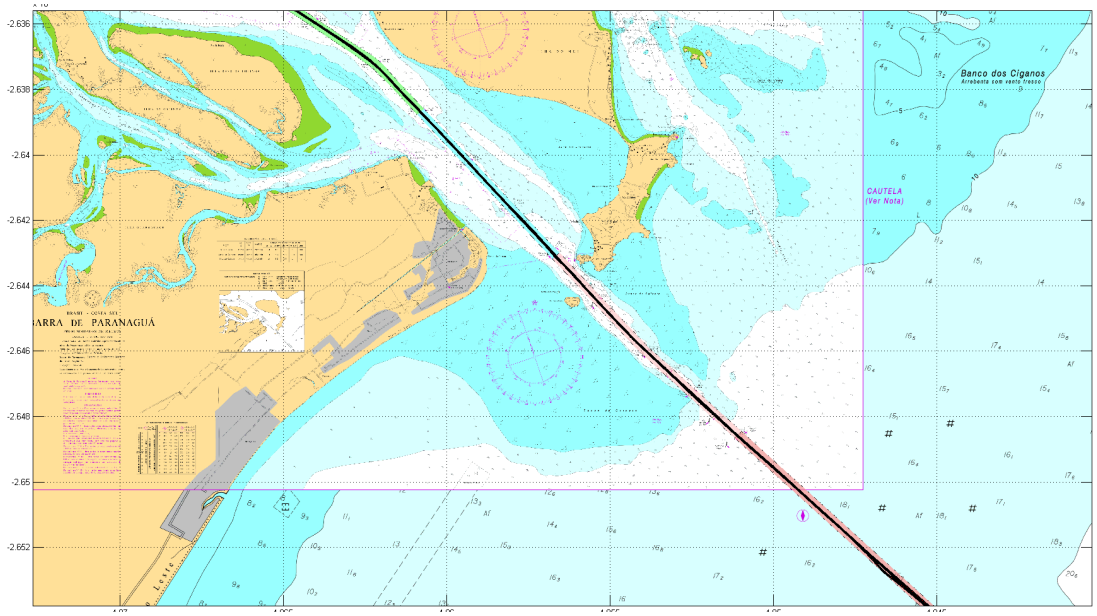


Figura 17: Trecho inicial da trajetória

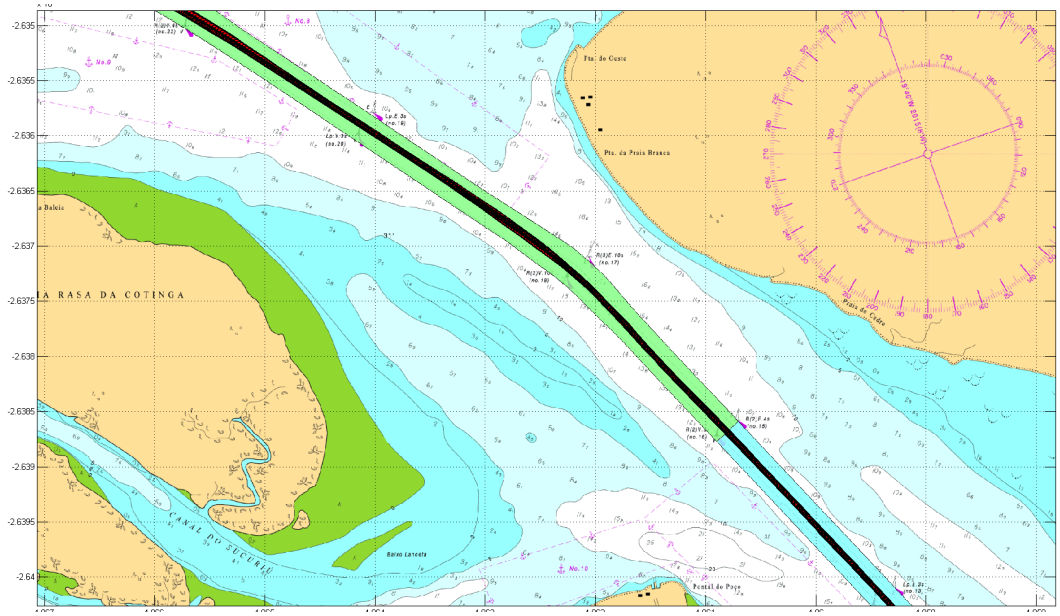


Figura 18: Trecho intermediário da trajetória

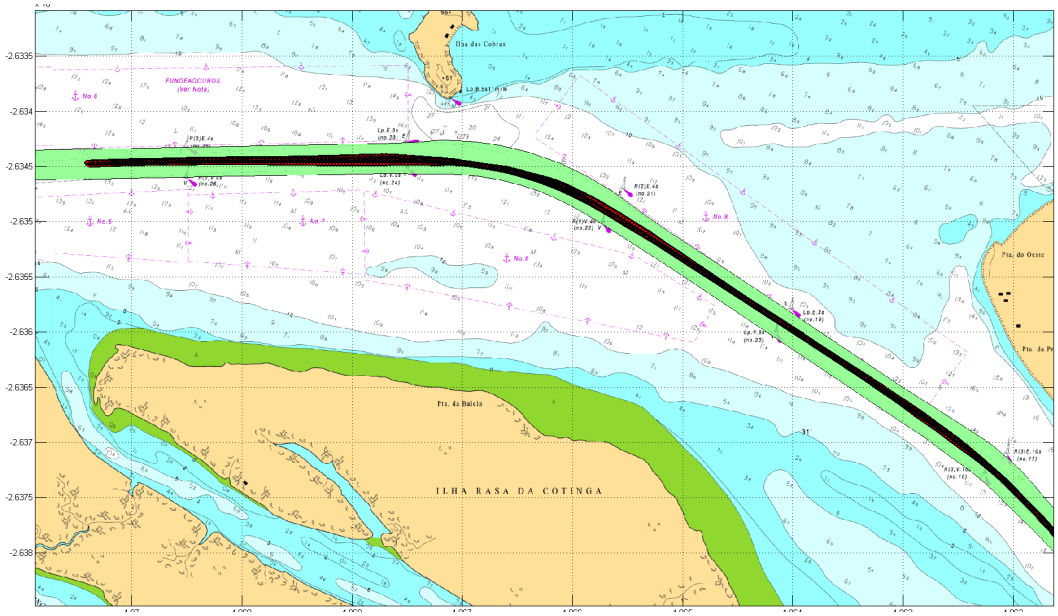


Figura 19: Trecho final da trajetória

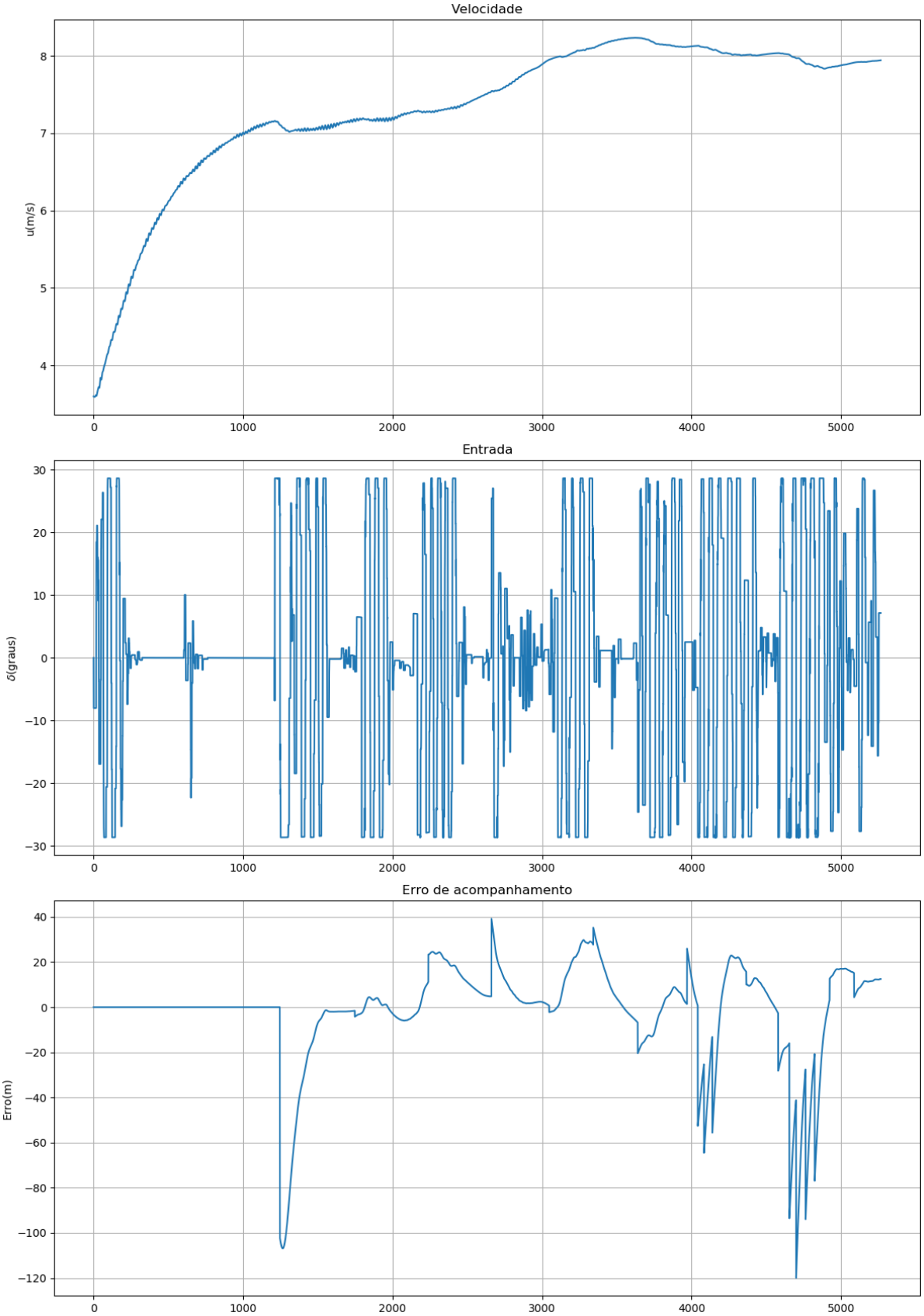


Figura 20: Velocidade, entrada e erro no caso 1

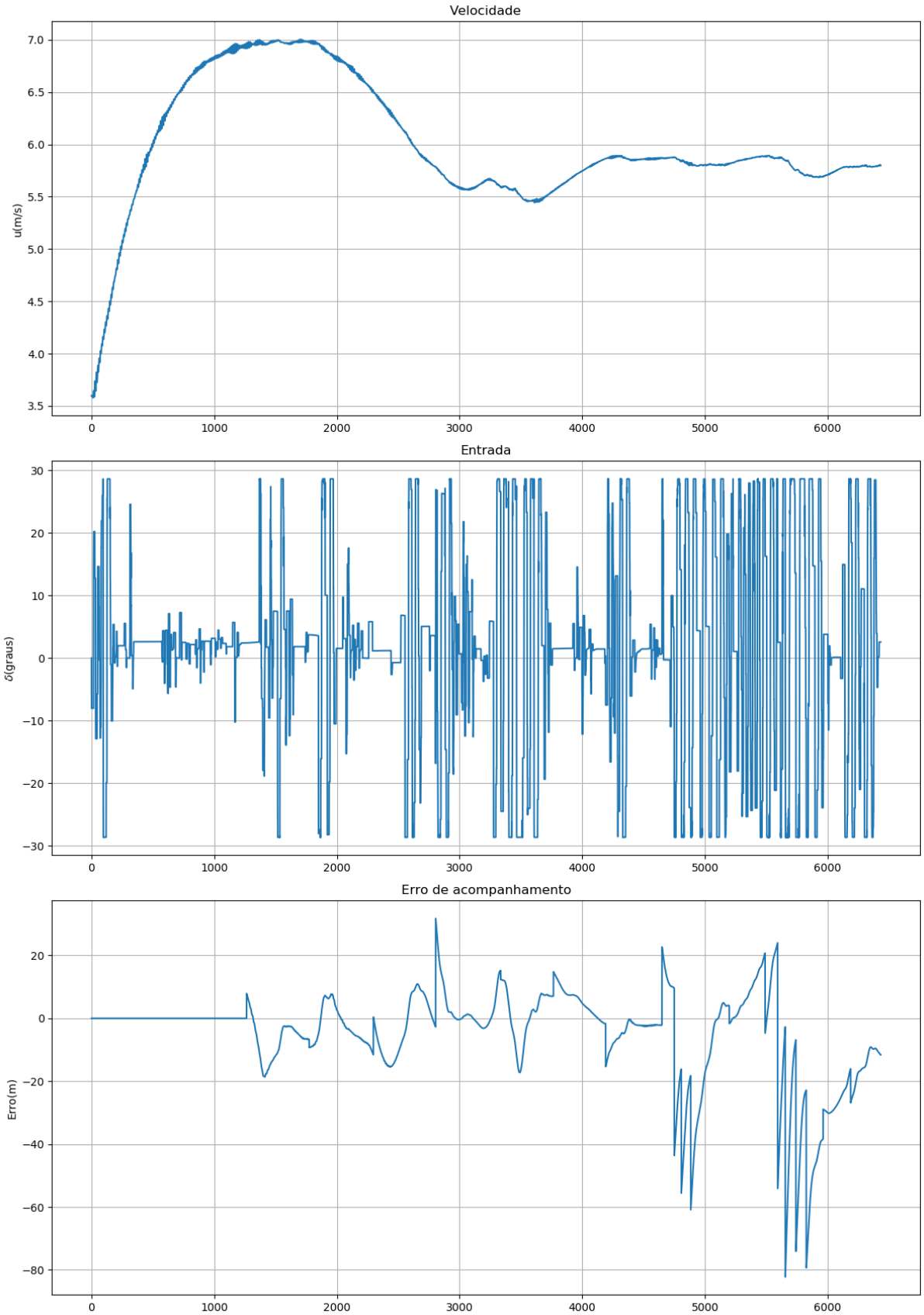


Figura 21: Velocidade, entrada e erro no caso 2

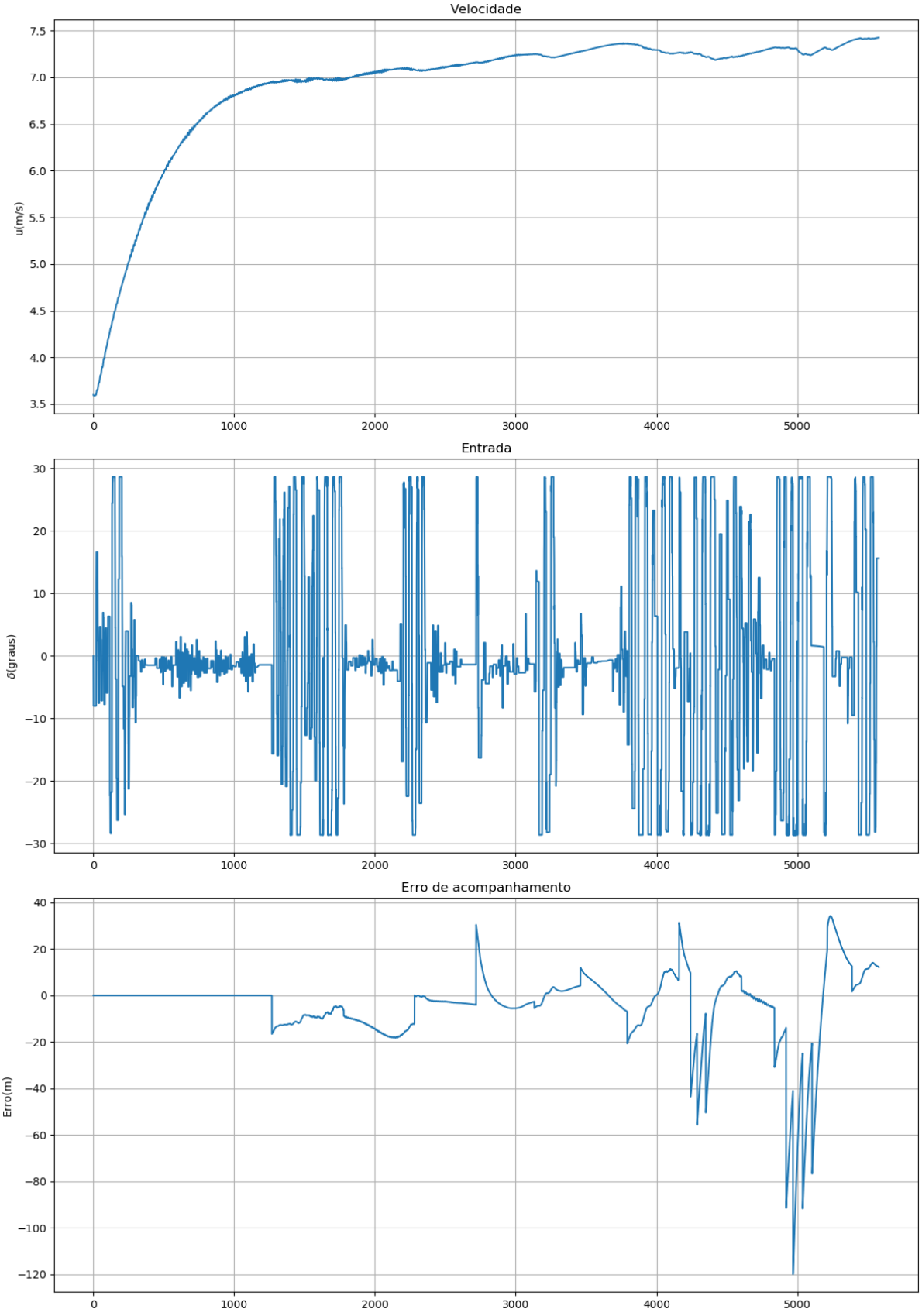


Figura 22: Velocidade, entrada e erro no caso 3

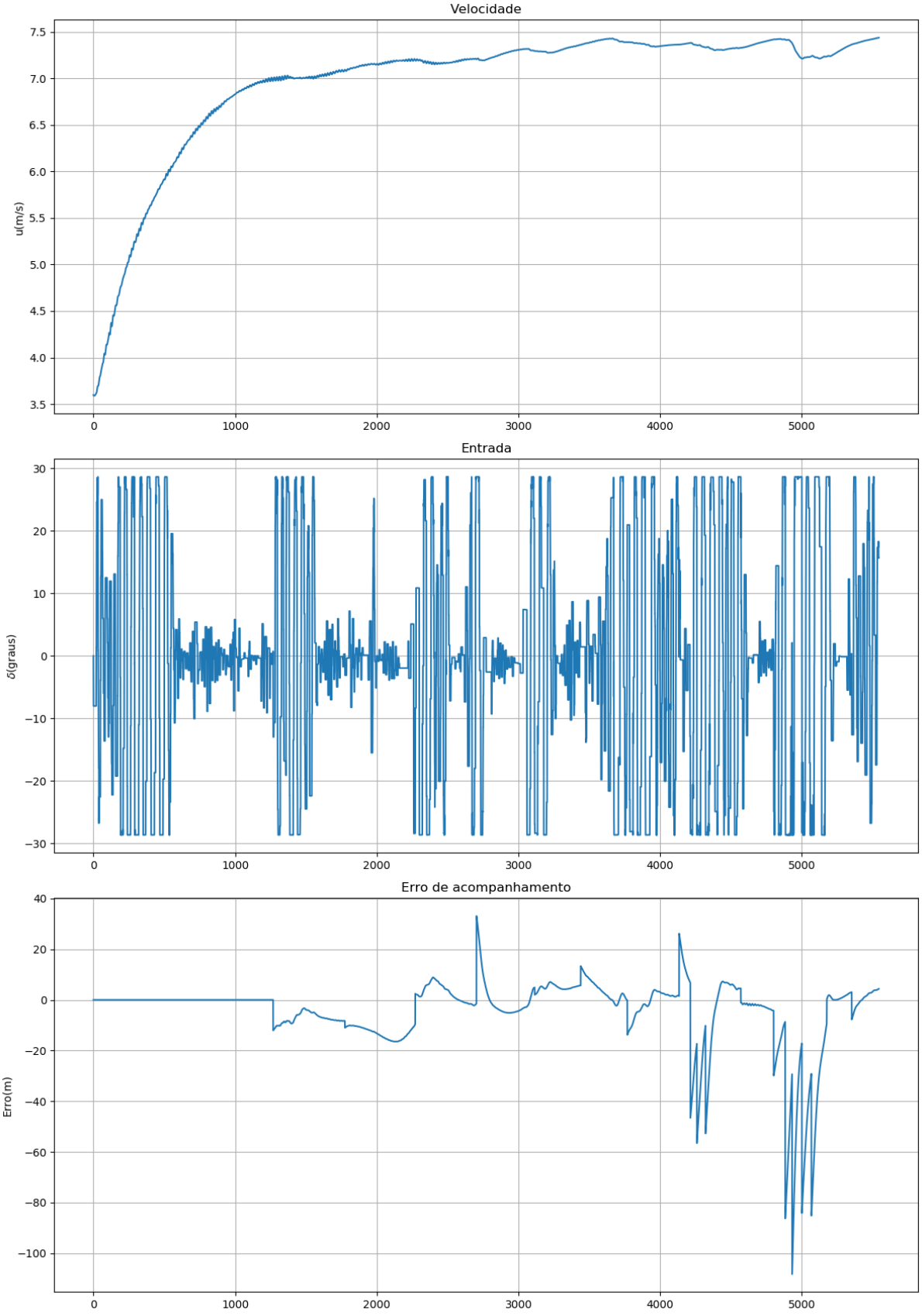


Figura 23: Velocidade, entrada e erro no caso 4

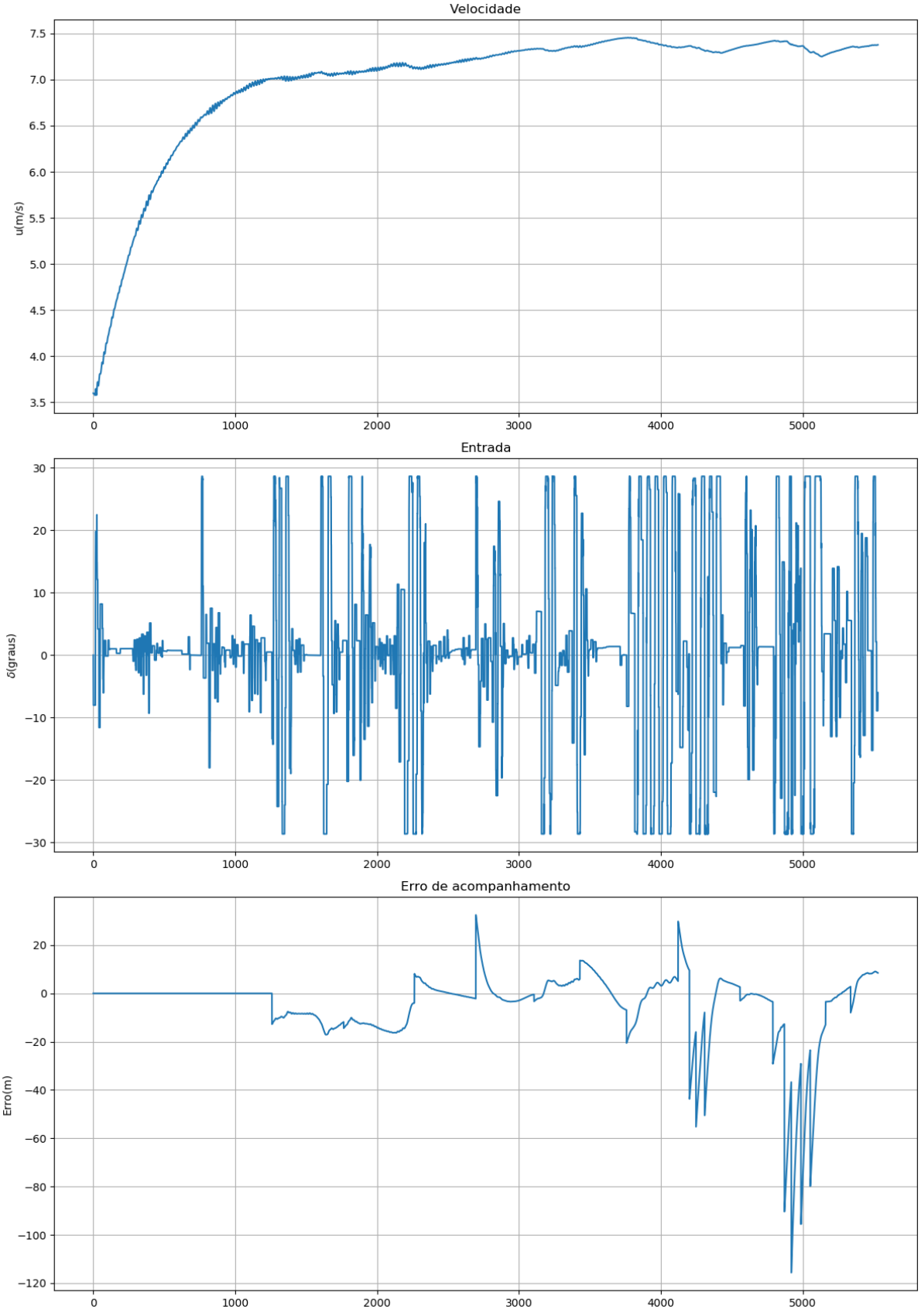


Figura 24: Velocidade, entrada e erro no caso 5

4 CONSIDERAÇÕES FINAIS

Nesse trabalho, implementamos um controlador de uma embarcação usando Model Predictive Control. Com o LOS, otimizamos a trajetória para evitar mudanças bruscas no objetivo e minimizar o overshoot ao chegar em um waypoint. Além disso, foi implementado um filtro de Kalman, capaz de obter os parâmetros necessários para o modelo do MPC. Com o uso do simulador PyDyna, implementado num módulo em Python, fomos capazes de obter os melhores parâmetros para o sistema e validar os nosso resultados.

Como próximo passo, poderíamos otimizar os valores dos parâmetros do filtro de Kalman e do controlador, de forma a ter entradas mais suaves de controle e um sistema de identificação dinâmica mais estável. Além disso, poderíamos implementar também um algoritmo de Collision Avoidance, capaz de gerar novos waypoints e depois retornar à trajetória original, caso encontrasse um obstáculo no caminho.

De forma geral, o controlador foi capaz de otimizar as entradas do controlador, para seguir uma trajetória bem definida, com erros baixos, sendo capaz de funcionar até mesmo em uma situação mais arriscada no qual a dinâmica do sistema varia, como na entrada de um porto, com diferentes situações ambientais. Assim, ao fim do projeto, tivemos uma performance satisfatória, com um sistema capaz de controlar uma embarcação de forma autônoma, em tempo real.

REFERÊNCIAS

1 ANNAMALAI, A. A review of model predictive control and closed loop system identification for design of an autopilot for uninhabited surface vehicles. *Technical Report*, 2012.

2 SIRAMDASU, Y.; FAHIMI, F. Incorporating input saturation for surface vessel control with experiments. *Control and Intelligent Systems*, IEEE, v. 41, n. 1, p. 49–55, 2013. ISSN 14801752.

3 ZHENG, H.; NEGENBORN, R. R.; LODEWIJKS, G. Trajectory tracking of autonomous vessels using model predictive control. *IFAC Proceedings Volumes (IFAC-PapersOnline)*, IFAC, v. 19, n. 3, p. 8812–8818, 2014. ISSN 14746670. Disponível em: ⟨http://dx.doi.org/10.3182/20140824-6-ZA-1003.00767⟩.

4 OH, S. R.; SUN, J. Path following of underactuated marine surface vessels using line-of-sight based model predictive control. *Ocean Engineering*, Elsevier, v. 37, n. 2-3, p. 289–295, 2010. ISSN 00298018. Disponível em: ⟨http://dx.doi.org/10.1016/j.oceaneng.2009.10.004⟩.

5 ZENG, Z.-H. et al. Path Following of Underactuated Marine Vehicles Based on Model Predictive Control. In: INTERNATIONAL SOCIETY OF OFFSHORE AND POLAR ENGINEERS. *The 29th International Ocean and Polar Engineering Conference*. [S.l.], 2019.

6 ABDELAAL, M.; FRÄNZLE, M.; HAHN, A. Nonlinear Model Predictive Control for trajectory tracking and collision avoidance of underactuated vessels with disturbances. *Ocean Engineering*, v. 160, n. May 2016, p. 168–180, 2018. ISSN 00298018.

7 LIU, H.; MA, N.; GU, X. Ship Course Following and Course Keeping in Restricted Waters Based on Model Predictive Control. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, v. 12, n. 2, p. 305–312, 2018. ISSN 2083-6473.

8 ZHANG, J.; SUN, T.; LIU, Z. Robust model predictive control for path-following of underactuated surface vessels with roll constraints. *Ocean Engineering*, Elsevier Ltd, v. 143, n. November 2015, p. 125–132, 2017. ISSN 00298018. Disponível em: ⟨http://dx.doi.org/10.1016/j.oceaneng.2017.07.057⟩.

9 ZHANG, J. et al. Robust model predictive control of the automatic operation boats for aquaculture. *Computers and Electronics in Agriculture*, Elsevier B.V., v. 142, p. 118–125, 2017. ISSN 01681699. Disponível em: ⟨https://doi.org/10.1016/j.compag.2017.08.016⟩.

10 TAKÁCS, B. et al. Export of explicit model predictive control to python. *Proceedings of the 2015 20th International Conference on Process Control, PC 2015*, IEEE, v. 2015-July, p. 78–83, 2015.

11 SKJETNE, R.; SMOGELI, Ø. N.; FOSSEN, T. I. A Nonlinear Ship Manoeuvering Model: Identification and adaptive control with experiments for a model ship. *Modeling, Identification and Control*, v. 25, n. 1, p. 3–27, 2004. ISSN 03327353.

12 PERERA, L. P.; OLIVEIRA, P.; SOARES, G. C. System Identification of Nonlinear Vessel Steering. *Journal of Offshore Mechanics and Arctic Engineering*, v. 137, n. 3, p. 1–7, 2015. ISSN 1528896X.

13 FOSSEN, T. I. *Handbook of marine craft hydrodynamics and motion control.pdf*. [S.l.: s.n.], 2011. ISBN 9781119991496.