Projeto de um Autopiloto de Carro Autônomo para Ultrapassagem

1º André Andrade Gonçalves
Instituto Tecnológico da Aeronáutica
São José dos Campos, Brasil
Email: andre.goncalves.8750.ga.br

2º Guilherme Saraiva Brasiliense
Instituto Tecnológico da Aeronáutica
São José dos Campos, Brasil
Email: guilherme.brasiliense.8832.ga.br

***Resumo*—Foi implementado um sistema de direção autônoma em um veículo simulado a partir de dois controladores, sendo eles um controlador PF para o ajuste da velocidade e um controlador PID com pré-filtro para o ajuste da posição transversal. A simulação considera duas faixas em linha reta, populadas por carros viajando no mesmo sentido que o veículo controlado. O controlador foi projetado de modo a realizar o processo de ultrapassagem do veículo principal. Os códigos implementados para a realização desse projeto estão disponíveis no repositório online GitHub[1].**

I. Introdução

Esse projeto busca encontrar uma solução para o problema de direção automática de carro com um modelo de seguidor de linha adaptado para o contexto de uma estrada, usando malhas de controle para velocidade e ângulo de direção. Na implementação, foi usado um controlador PID com pré-filtro, implementado com auxílio da transformada Tustin, para o ajuste do ângulo e um controlador PF para o ajuste da velocidade [1].

A modelagem foi construída para uma simulação na qual existem duas faixas, se a faixa da direita populada por carros viajando à velocidade constante no mesmo sentido que o veículo controlado, sendo que o carro controlado pode acelerar até uma velocidade máxima, manobrar para direita ou esquerda e frear. Com o intuito de se aproximar melhor da situação real de uma estrada, a simulação também considera a existência da força de resistência do ar , que faz com que o veículo simulado desacelere proporcionalmente a sua velocidade, de forma que isso também evita que o veículo acelere somente no começo do trajeto, o que não funcionaria em um cenário físico real. As informações coletadas para a alimentação das malhas são o valor da velocidade e da leitura da posição atual no eixo horizontal.

II. Modelagem Física da Planta

A. Cinemática do Veículo

Para descrever o movimento do veículo, utilizar-se-á o modelo apresentado na Figura 4, onde considera-se um carro apenas com tração dianteira, de modo que as rodas traseiras possuem o sentido fixo do carro. O movimento lateral do carro ocorre com o manuseio do volante que altera a angulação das rodas dianteira de modo a gerar rotação, em torno de um ponto de contato R. Note, como apresentado na Figura 4, os ângulos de rotação devem ser diferentes para as diferentes rodas, para que o ponto de contato seja o mesmo. Isso é possível utilizando um sistema de geometria diferencial de Ackerman [4].

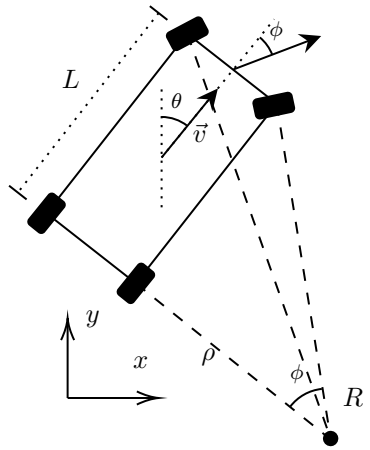

Figura 1: Esquema do modelo utilizado para a cinemática do veículo em questão.

O movimento das rodas traseiras tenderá a permanecer perpendicular à reta entre a roda e o ponto de encontro das linhas de direção, de modo que o veículo segue a rota perpendicular, o que implicará nas Equações 1 e 2. Porém o ângulo de orientação é alterado de modo que $\rho d\theta = vdt$. Assim, como $\rho = L/\tan(\phi)$, é possível encontrar uma expressão para $\dot{\theta}$ [3].

$$\dot{x} = v \cdot \sin(\theta) \tag{1}$$

$$\dot{y} = v \cdot \cos(\theta) \tag{2}$$

$$\dot{\theta} = \frac{v}{L} \cdot \tan(\phi) \tag{3}$$

Nessa equação, x e y são as coordenadas do carro no ambiente simulado, θ é o ângulo do carro com a vertical medido no sentido horário, v é a velocidade de translação do carro, ϕ é o ângulo do volante com relação à orientação do carro medido no sentido horário e L é a dimensão de comprimento do carro. Assumindo velocidade constante para a malha de posição e a aproximação para pequenos ângulos $\sin(\theta) \approx \theta$ e $\tan(\phi) \approx \phi$:

[1]https://github.com/andresafira/Autonomous_Car_Control

$$\dot{x} = v \cdot \theta \tag{4}$$

$$\dot{\theta} = \frac{v}{L} \cdot \phi \tag{5}$$

$$\frac{d}{dt}\begin{bmatrix} x \\ \theta \end{bmatrix} = \begin{bmatrix} 0 & v \\ 0 & 0 \end{bmatrix}\begin{bmatrix} x \\ \theta \end{bmatrix} + \frac{v}{L}\begin{bmatrix} 0 \\ 1 \end{bmatrix}\phi \tag{6}$$

B. Dinâmica do Veículo

No modelo em questão, considera-se a dinâmica de velocidade do veículo baseada em um mecanismo capaz de aplicar uma força determinada ao carro (por meio da tração das rodas dianteiras). Considera-se ainda a presença de uma força de resistência do ar linear, de forma que a equação de dinâmica de velocidade do veículo será como apresentada na Equação 7, em termos da força de aplicação u_f, da massa do carro m, de sua velocidade v e da constante de resistência do ar b [3].

$$u_f = m\dot{v} + bv \tag{7}$$

III. Controle

A. Controlador PF

O Controlador PF trata-se de um controle proporcional adicionado a um termo de *feedforward* da entrada de referência, com o intuito de remover erro em regimesendo a lei de controle associada a seguinte:

$$u = K_p(x_r - x) + K_{ff}x_r, \tag{8}$$

Esse sistema de controle possui o ganho K_p com a função de reagir ao erro atual, assim como uma componente *feedforward* K_{ff} baseado na referência, com o intuito de diminuir o erro e melhorar o tempo de resposta ao sinal.

B. Controlador PID

A lei de controle associada a um controlador PID é:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d\dot{e}(t), \tag{9}$$

em que $e(t)$ é o erro em relação à referência, $u(t)$ é o esforço (comando) de controle, e K_p é o ganho proporcional, K_i é o ganho integrativo e K_d é o ganho derivativo. Assim, em Laplace, tem-se

$$U(s) = \left(K_p + \frac{K_i}{s} + K_d s\right)E(s) \tag{10}$$

$$\implies C(s) = \frac{U(s)}{E(s)} = K_p + \frac{K_i}{s} + K_d s \tag{11}$$

O termo derivativo do controle é responsável por tornar o sistema responsivo rapidamente a mudanças na referência, pois leva em consideração a sua derivada no cálculo de u, além de ajudar a aumentar a estabilidade, contudo apresenta o problema de amplificar ruídos. Enquanto isso, o termo integrativo remove a existência de erro em regime, mesmo com perturbação, tendo a função de corrigir a atuação de uma força externa, contudo amplifica oscilações, aumentando a instabilidade do sistema.

Para demonstrar que o componente integral de um controlador PID elimina o erro em regime permanente, podemos analisar a resposta do sistema a uma entrada degrau. Considere um sistema de controle em malha fechada com um controlador PID. Assumindo que tem-se um sistema com uma função de transferência da planta $G(s)$, a função de transferência em malha aberta é:

$$L(s) = C(s)G(s) = \left(K_p + \frac{K_i}{s} + K_d s\right)G(s) \tag{12}$$

A função de transferência em malha fechada $T(s)$ é:

$$T(s) = \frac{C(s)G(s)}{1 + C(s)G(s)} = \frac{\left(K_p + \frac{K_i}{s} + K_d s\right)G(s)}{1 + \left(K_p + \frac{K_i}{s} + K_d s\right)G(s)} \tag{13}$$

A função de transferência do erro $E(s)$ é:

$$E(s) = \frac{1}{1 + C(s)G(s)} = \frac{1}{1 + \left(K_p + \frac{K_i}{s} + K_d s\right)G(s)} \tag{14}$$

Para uma entrada degrau $R(s) = \frac{1}{s}$, o erro em regime permanente e_∞ é encontrado usando o teorema do valor final:

$$e_\infty = \lim_{t\to\infty} e(t) = \lim_{s\to 0} sE(s)R(s) \tag{15}$$

Substituindo $R(s) = \frac{1}{s}$, a função de transferência do erro $E(s)$ e assumindo que a planta do sistema $G(s)$ é função racional própria (isto é, o grau polinomial do denominador é maior que o do numerador):

$$e_\infty = \lim_{s\to 0} \frac{1}{1 + \left(K_p + \frac{K_i}{s} + K_d s\right)G(s)} = 0 \tag{16}$$

Com isso, tem-se que de fato o erro em regime é nulo.

A função do pré-filtro é a eliminação da influência de zeros na função de transferência, facilitando o alcance dos requerimentos de sistema e da implementação do controlador.

C. Transformada Z e a Técnica de Tustin

Nota-se que a implementação do controlador PID depende do cálculo da derivada do erro e da integral do erro, porém como o cálculo do controlador requer um tempo mínimo de execução, não é possível realizar esses cálculos de modo instantâneo, de modo que é necessário levar em conta tal atraso na implementação do controlador [2]. Isso pode ser alcançado a partir do uso da técnica de transformada z que representa uma aproximação discreta da transformada de Laplace:

$$\mathcal{L}\{f(t)\} = \int_0^\infty f(t)e^{-st}dt \approx \sum_{k=0}^{\infty} f(kT)e^{-skT}T \tag{17}$$

Convencionando $z = e^{-sT}$:

$$\sum_{k=0}^{\infty} f(kT)e^{-skT}T = T\sum_{k=0}^{\infty} f[k]z^{-k} = T\mathcal{Z}\{f[k]\} \tag{18}$$

Assim é possível demonstrar que a transformada Z tem a propriedade de que $f[k]z^{-k} = f[k-n]$, de modo que é possível modelar o atrasado gerado por um tempo de amostragem, a partir de entradas passadas, o que facilita consideravelmente a implementação do controlador.

A técnica de Tustin [2], ou transformação bilinear, consiste em uma aproximação que converte do espaço contínuo de Laplace para o espaço discreto da transformada Z, de modo que basta substituir os termos s da função de transferência controlador ou filtro utilizado pela expressão a seguir:

$$s = \frac{2}{T}\frac{1 - z^{-1}}{1 + z^{-1}} \tag{19}$$

IV. Implementação

A. Simulação

Para criar um ambiente de testes para o modelo veicular, bem como a visualização do efeito do controlador, foi feita uma simulação computacional usando a biblioteca *Pygame* da linguagem *Python*. Na implementação, existe uma classe chamada *Simulation* que permite encapsular os métodos associados a simulação, bem como uma classe *Car*, a qual encapsula os métodos associados ao movimento e atualização de seus parâmetros ao acionar o sistema de controle.

1) Car: O uso de uma classe *Car* visa atender o princípio de encapsulamento da Programação Orientada a Objetos, de forma que as variáveis internas do carro ficam ocultas do resto do código, assim, sempre que for necessário fazer algo com o carro, como movê-lo ou enviar informações ao controle. Para se movimentar, o carro pode ser controlado por um humano com os seguintes métodos:

- **accelerate:** utilizada para incrementar a velocidade do carro;
- **brake:** utilizada para reduzir a velocidade do carro;
- **turn_left:** utilizada para incrementar o ângulo do volante;
- **turn_right:** utilizada para reduzir o ângulo do volante.

O carro também pode se movimentar com o uso apenas da malha de controle, a partir da resposta de comando gerada pelos controladores projetados. Dois controladores são imbutidos, de velocidade e de posição, sendo o primeiro responsável por determinar o comando de força a ser aplicada para alcançar a velocidade de referência, e o segundo de gerar o comando de ângulo do volante para orientar o veículo na posição horizontal de referência.

Para identificar se houve uma colisão, o carro verifica se a *bounding box* de algum obstáculo está dentro de sua *bounding box*, que é uma forma retangular usada para definir a localização e o tamanho do carro. Caso o carro tenha colidido em algo ele é considerado inativo e permanece parado até o fim da simulação.

2) Simulation: A outra classe utilizada contém os métodos mais especificamente associados à simulação, o construtor da classe inicializa o $Pygame$ e a tela na qual a simulação acontecerá, também instancia um elemento da classe Car e adiciona os $sprites$ do plano de fundo e do carro na simulação, os $sprites$ utilizados para deixar a simulação visualmente agradável estão apresentados na Figura 2.

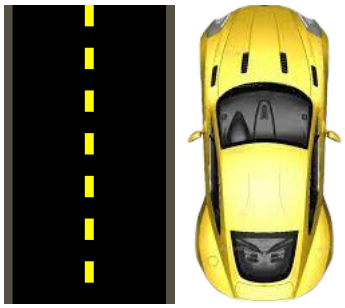

Figura 2: *Sprites* utilizadas para apresentar a simulação de forma visual.

A maioria dos métodos internos da simulação estão associados a desenhar na tela os elementos gráficos, ou seja, o carro e o cenário. A outra função executada pela classe é gerar os obstáculos, estes são outros carros denominados como *Dummies* na simulação, pois não podem ser controlados como o carro principal, apenas possuem uma velocidade fixa na direção vertical. Por fim, o método $update$ da classe é quem de fato chama os métodos que desenham os elementos gráficos, além de atualizar a posição do carro principal e de todos os $Dummies$.

A simulação também determinará qual linha de referência o carro deve seguir, sendo essa atualizada quando o carro se aproxima de algum obstáculo para então o sistema de controle ocasionar a mudança de faixa para realização da ultrapassagem, voltando para a faixa original no fim do processo. O carro deverá ultrapassar, quando se aproxima de uma distância mínima traseira do carro a ser ultrapassado (como fator de segurança, essa distância mínima deve ser maior que os comprimentos dos carros), e então volta para a faixa inicial assim que passar por uma distância mínima frontal do carro a ser ultrapassado.

3) Control: Nesse arquivo, são armazenados e inicializados os controladores a serem utilizados no projeto, utilizando a técnica de implementação discreta de transformada de Tustin.

4) Main Playable: Nesse arquivo ocorre a simulação do veículo controlado pelo próprio usuário, utilizando as teclas direcionais do teclado (key arrows).

5) Main Simulation: No arquivo em questão é realizada a simulação do movimento do carro considerando unicamente a atuação dos controladores de velocidade e posição do sistema em questão.

6) Main Gridsearch: Arquivo responsável por simular o movimento do carro em alguma condição inicial (principalmente com velocidade inicial 0 ou máxima) para vários valores de ξ e ω_n do movimento da malha de posição, de modo a realizar a busca do par de (ξ, ω) que aumenta a reatividade do veículo para as mudanças de faixa.

V. Controladores

O algoritmo conta com dois controladores, um PF para o ajuste da velocidade e um PID para o ajuste do ângulo

do volante. Os valores dos ganhos foram ajustados a fim de diminuir o máximo possível o valor de *overshoot* e tempo de subida,

1) Controlador PF: O controlador PF foi criado para o ajuste da velocidade a partir de sua equação de dinâmica movimento, apresentada na Equação 7. A partir da qual, foi montada a seguinte malha de controle:

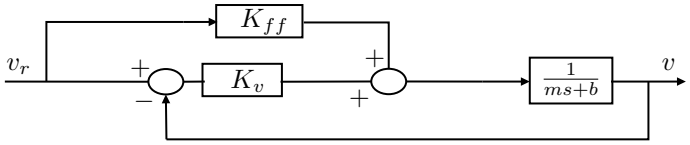

Figura 3: Malha de controle para a dinâmica de velocidade do veículo.

A função de transferência final é:

$$G_v(s) = \frac{K_v + K_{ff}}{ms + b + K_v} \tag{20}$$

A escolha dos ganhos deve ser feito de modo a gerar erro em regime nulo e tempo de subida característico de τ, de modo que:

$$e_\infty = \lim_{s\to 0} 1 - G_f(s) = \frac{b - K_{ff}}{b + K_v} \tag{21}$$

Dessa forma valor de K_{ff} deve ser igual a b e o de K_v foi ajustado de acordo com a constante de tempo $\tau = 0.2$ s com:

$$K_v = \frac{m}{\tau} - b \tag{22}$$

A. Controlador PID

O controlador PID foi implementado com o intuito de ajustar o valor do ângulo do volante, de forma que seja possível seguir a linha corretamente, sendo sua lógica feita a partir das Equações 4, 5 e 6, utilizando a técnica de Tustin para implementar o controlador de modo digital. Adotando ainda um pré-filtro, também implementado discretamente com a transformada de Tustin para eliminar a ação dos zeros e facilitar o desenvolvimento analítico inicial da malha de controle, foi obtida a seguinte malha para o controlador:

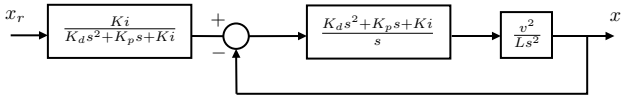

Figura 4: Malha PID implementada para seguir a linha de referência.

A função de transferência resultante é:

$$G_x(s) = \frac{K_i}{Ls^3 + v^2 \cdot K_d \cdot s^2 + v^2 \cdot K_p \cdot s + v^2 \cdot K_i} \tag{23}$$

Como a função de transferência contém 3 polos, para a modelagem dos ganhos ser feita a partir dos dados de ξ e ω_n, relacionados a um sistema de segunda ordem padrão, os polos foram modelados como sendo 2 complexo-conjugados e um polo real negativo de módulo elevado para ser possível a aplicação da aproximação de polos dominantes. Seguindo as recomendações de valores escolheu-se o posicionamento do terceiro polo em $-5\xi\omega_n$.

Os ganhos do controlador foram então ajustados com base nas entradas escolhidas de ξ e ω_n a fim de minimizar o tempo de subida e o *overshoot* do sistema, o que foi alcançado com o uso de uma busca em *grid* num espaço de ξ por ω_n. Para isso, foram testados valores de ξ entre 0.8 e 1.2 com um passo de 0.05 e de ω_n entre 7 e 10 com um passo de 0.5. Para cada par de valores, foi feito o gráfico de movimento a partir da simulação com duração de 2.5 s tanto para o caso do carro partir já com a velocidade máxima (caso em que apenas o comando de ângulo do volante irá mudar) e para o caso de partida do repouso (caso em que ambos comandos irão mudar, de modo a observar-se o efeito da malhas aninhadas).

Os valores que geraram os melhores resultados de tempo de subida e overshoot, isto é, menores valores de ambos, de modo a gerar um movimento reativo e evitar colisões na ultrapassagem e no movimento inicial de mudança de faixa (com partida no repouso), foram os adotados para a simulação final.

VI. Resultados e discussões

A resposta de velocidade do sistema projetado encontra-se na Figura 5. Percebe-se que o sistema demora consideravelmente para atingir a resposta de sinal desejado, o que ocorre devido à limitação de aplicação de comando do controlador de força, que possui uma força máxima que consegue gerar, o que explica a subida praticamente linear no início.

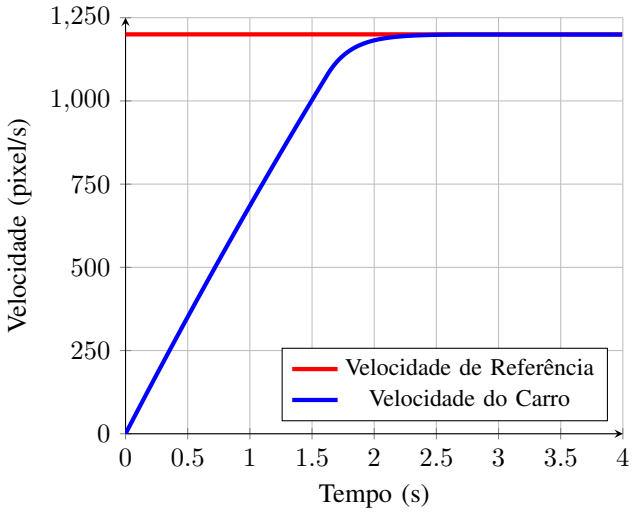

Figura 5: Gráfico de resposta do sistema de controle projetado na simulação de ultrapassagem. Para $t = 0$ o veículo encontra-se na faixa errada e em repouso logo deve aumentar a velocidade e se direcionar à faixa correta. Após $t = 2$ um veículo aparece na mesma faixa de modo que é necessário realizar uma ultrapassagem. Após a ultrapassagem o carro deve voltar à faixa anterior.

Já para a resposta de posição horizontal, foi realizada a busca em grid dos parâmetros de ξ e ω_n que gerariam a melhor resposta para a entrada degrau em termos de saída em repouso e saída já com a velocidade de referência (de modo a evitar colisões em ambos os casos). Os resultados para algumas das simulações (apenas três delas) encontram-se na figura a seguir.

Nota-se que alguns gráficos, para a condição de velocidade inicial nula (partindo do repouso), geram resposta constantes a partir de certo ponto, que representa que o carro da simulação em questão colidiu com o pavimento lateral, de modo que parou.

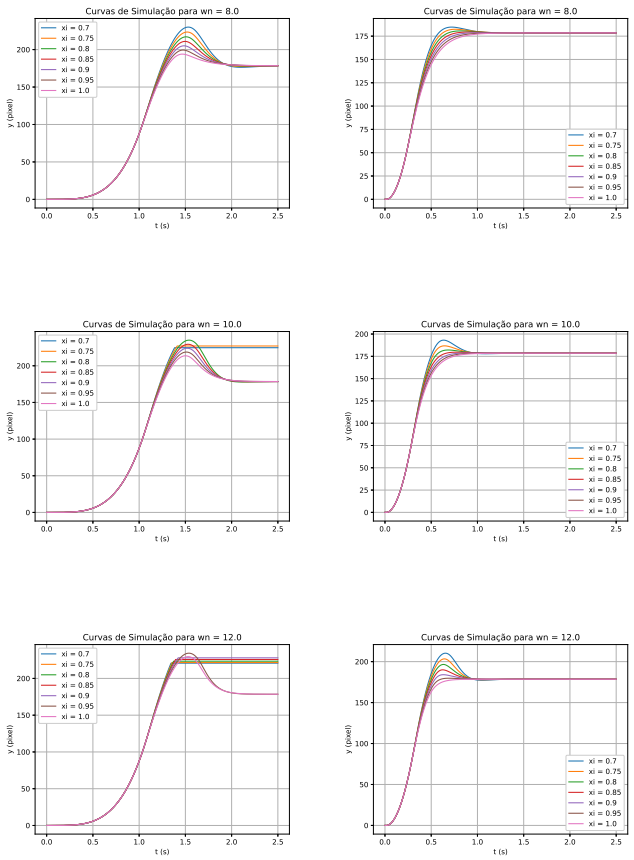

Figura 6: Exemplo de resultados da busca em grid para os parâmetros ξ e ω_n, para $\omega_n \in \{8, 10, 12\}$. Os gráficos da coluna da esquerda mostram o caso de sistema partindo do repouso, e na coluna da direita, o caso de sistema já partindo com a velocidade de referência.

Dessa forma, decidiu-se adotar o par de referência que gerava os menores resultados de overshoot e tempo de subida em ambos os casos de velocidade inicial, e que não geravam colisão, de modo que tomou-se (ξ, ω_n) = (1, 10). Já para a resposta de posição horizontal, o resultado encontra-se na Figura 7. Note que inicialmente o sistema demora consideravelmente para atingir a posição de referência e há *overshoot* em relação à linha de referência, mesmo considerando que o valor projetado de $\xi = 1.0$ não deveria resultar em *overshoot*, segundo a teoria (já que corresponde a um sistema amortecido criticamente). Esse desvio ocorre pois a velocidade do veículo não coincide com a velocidade utilizada para o projeto do controlador de posição, entretanto nota-se que, ainda assim o sistema converge para o resultado esperado.

A queda repentina da linha de referência corresponde à aparição de um carro na faixa da direita, de modo que o carro irá começar o procedimento de ultrapassagem. Percebe-se um atraso na resposta de aproximadamente 1 segundo entre a referência e a posição do carro, valor que está relacionado com o tempo de subida do controlador[2].

Finalmente para a última seção é possível perceber que o sistema se comporta segundo o esperado, com amortecimento crítico (sem *overshoot*) e erro em regime nulo.

VII. Conclusão

O sistema final implementado apresentou resultados correspondentes com o esperado, com o controlador permitindo manter a velocidade desejada e realizar as ultrapassagens de forma rápida e precisa, sem colisões e com uma margem de segurança considerável. O controlador PID concluiu o seu objetivo, permitindo que o carro siga a linha com tempo de resposta razoável, mesmo com a troca repentina da posição de referência que ocorre para a ultrapassagem. Com isso, de fato o controlador projetado possui o grau adequado de complexidade e de resposta em relação ao problema, sendo capaz de alcançar os resultados desejados.

O controlador PF foi capaz, como esperado, de manter a velocidade desejada, mesmo com a existência de uma força de arrasto. Além da precisão do resultado, a velocidade de referência é alcançada rapidamente, apenas limitada pela própria capacidade máxima do controlador de gerar a força de movimento. Com isso, temos que o problema abordado foi solucionado com precisão e resultados razoáveis, com a escolha dos controladores sendo adequada.

Referências

[1] Karl Johan Astrom e Richard M. Murray. *Feedback systems: an introduction for scientists and engineers*. 2ª ed. Princeton University Press, 2018.

[2] Connor W. Herron. *Software Implementation of Digital Filtering via Tustin's Bilinear Transform*. 2024. arXiv: 2401.03071 [cs.RO]. URL: https://arxiv.org/abs/2401.03071.

[3] Steven M LaValle. *A Simple Car*. Abr. de 2012. URL: https://msl.cs.uiuc.edu/planning/node658.html.

[4] William Norris. *Modern steam road wagons*. London, New York, Bombay, Longmans, Green e co., 1906.

[2]Neste caso o tempo de subida se refere à entrada degrau unitária, mas o processo é similar para qualquer outra linha de referência.

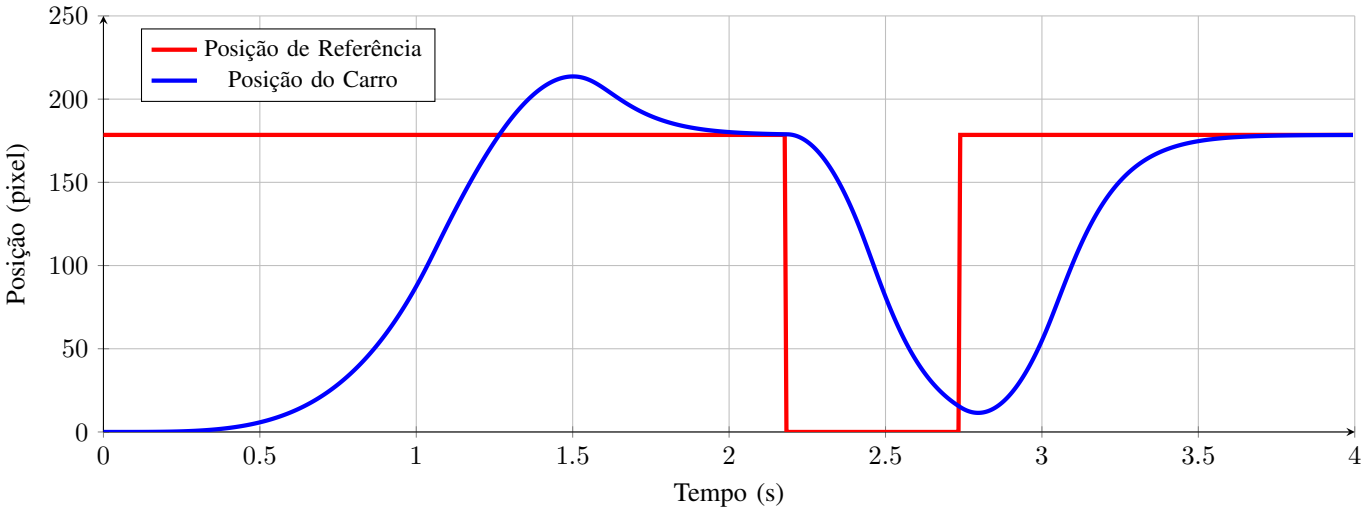

Figura 7: Gráfico de resposta do sistema de controle projetado na simulação de ultrapassagem. Para $t = 0$ o veículo encontra-se na faixa errada e em repouso logo deve aumentar a velocidade e se direcionar à faixa correta. Após $t = 2$ um veículo aparece na mesma faixa de modo que é necessário realizar uma ultrapassagem. Após a ultrapassagem o carro deve voltar à faixa anterior.