Relatório de Atividade

Controle de Posição um Robô de Tração Diferencial

Débora Oliveira
Prof Antonio Marcus, Automação Inteligente 20.3

9 de Novembro de 2020

Esse documento tem por objetivo descrever um controle de posição "*go to goal*" de um robô de tração diferencial (RTD). Esse trabalho trata da comparação dos resultados da malha fechada adquiridos a partir simulação computacional do modelo no MATLAB e do modelo de um Pioneer P3DX na plataforma CoppeliaSim.

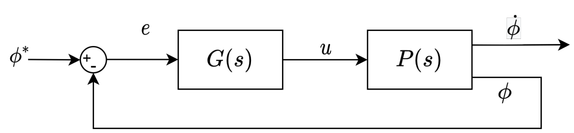
1 Fundamentação teórica

O modelo controlador-planta empregado nesse trabalho está ilustrado na Fig. 1. A planta do sistema, cuja função de transferência é $P(s)$, é definida pelos seguintes estados:

$$\dot{z} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \nu_0 \cos\phi \\ \nu_0 \sin\phi \\ u \end{bmatrix} \tag{1}$$

para $z = [x \quad y \quad \phi]^T$ a pose do robô. No caso em estudo, a velocidade linear do RTD é constante ν_0 e o rastreio do ponto alvo (*goal*) será feito por meio da atualização da velocidade angular do RTD ω.

Fig. 1. Diagrama do modelo simulado.



Dessa forma, o controlador definido pela função de transferência é $G(s)$ tem por entrada o erro entre o ângulo de referência ϕ^* e a orientação atual ϕ do RTD. O ângulo ϕ^* é dado por

$$\phi^* = \arctan\frac{y^* - y}{x^* - y} \tag{2}$$

para $[x^* \quad y^*]$ a posição do alvo no plano XY. É importante destacar que o erro $e = \phi^* - \phi$ é tal que $e \subset [-\pi;\ \pi]$.

1.1 Controlador proporcional-derivativo

A lei de controle modelada será generalizada por uma formulação PD. Entretanto, o termo derivativo será acompanhado por um filtro passa-baixa com polo em $-c$. Esse filtro derivativo rejeita as componentes de alta frequência do erro enquanto garante o amortecimento de e.

A lei de controle é definida por

$$\begin{aligned} G(s) = \frac{U(s)}{E(s)} &= K_p + K_d\frac{s(c)}{s+c} \\ &= \frac{(K_p + K_d c)s + K_p c}{s+c} \\ &= G_1(s)G_2(s) \end{aligned} \tag{3}$$

para

$$G_1(s) = \frac{R(s)}{E(s)} = \frac{1}{s+c} \tag{4}$$

$$G_2(s) = \frac{U(s)}{R(s)} = (K_p + K_d c)s + K_p c \tag{5}$$

A partir de equação (4), sabe-se que

$$\dot{r}(t) + cr(t) = e(t) \tag{6}$$

Com base na equação (5), obtem-se

$$u(t) = (K_p + K_d c)\dot{r}(t) + (K_p c)r(t) \tag{7}$$

Substituindo $\dot{r}$ definido na equação (6) na equação (7), encontra-se o seguinte espaço de estados.

$$\begin{aligned} \dot{r} &= -cr + e \\ u &= -(K_d c^2)r + (K_p + K_d c)e \end{aligned} \tag{8}$$

Unindo a equação (1) com a equação (8), tem-se o espaço de estados de malha aberta a seguir.

$$
\dot{z}' = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \\ \dot{r} \end{bmatrix} = \begin{bmatrix} \nu_0 \cos\phi \\ \nu_0 \sin\phi \\ -(K_d c^2) r + (K_p + K_d c) e \\ -cr + e \end{bmatrix} \tag{9}
$$
$$
y = \omega = \dot{\phi}
$$

Ao linearizar o sistema sobre a pose de origem $z_0 = [x_0 \quad y_0 \quad \phi_0 \quad r_0]^T$, tem-se o espaço de estados linear de malha aberta.

$$
\begin{aligned} \dot{z}' &= [\dot{x} \ \ \dot{y} \ \ \dot{\phi} \ \ \dot{r}]^T = Az' + Be \\ y &= \dot{\phi} = Cz' + De \end{aligned} \tag{10}
$$

para

$$
A = \begin{bmatrix} 0 & 0 & -\nu_0 s\phi_0 & 0 \\ 0 & 0 & \nu_0 c\phi_0 & 0 \\ -\frac{K(y_0-y^*)}{\sigma} & \frac{K}{\sigma} & -K & -K_d c^2 \\ -\frac{(y_0-y^*)}{\sigma} & \sigma^{-1} & -1 & -c \end{bmatrix} \tag{11}
$$
$$
\rightarrow K = (K_p - K_d c)
$$
$$
\sigma = (y_0 - y^*)^2 + (x_0 - x^*)^2
$$

Para o ponto de operação $[\dot{x}_0 \quad \dot{y}_0 \quad \dot{\phi}_0]^T = [0 \ 0 \ 0.25\pi]^T$, $\nu_0 = 0.2$ e $x^* = y^* = 2$, o determinante da matriz $(sI - A)$ apresentada na equação (11) é caracterizado pelo polinômio

$$
\begin{aligned} p = {} & s^4 + (K_p + c + K_d c)s^3 \\ & + (\frac{\sqrt{2}}{20}(K_p + K_d c) + K_p c)s^2 \\ & + \frac{\sqrt{2}}{20} K_p s \end{aligned} \tag{12}
$$

É sabido que em um filtro passa baixa, a frequência de corte $\omega_c = c^{-1}$ para $-c$ o polo do filtro. Por critério de projeto, foi definido $c = 100$. A equação (12) pode ser reescrita como

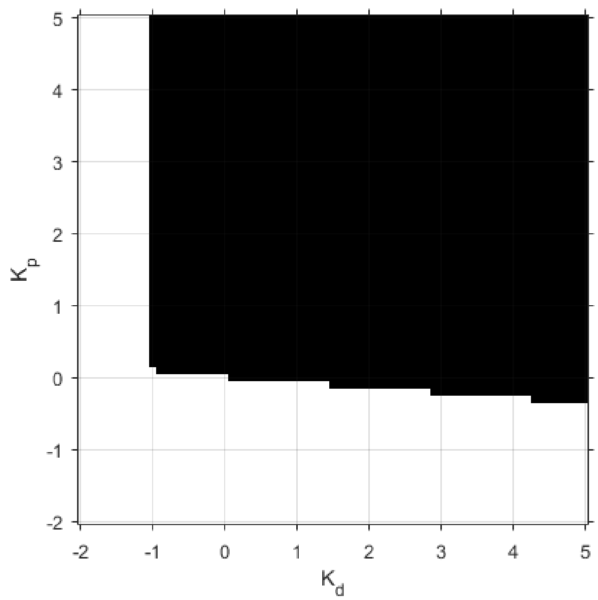
$$
\begin{aligned} p = {} & s^4 + (K_p + 100 + 100K_d)s^3 \\ & + (\frac{\sqrt{2}}{20}(K_p + 100K_d) + 100K_p)s^2 \\ & + \frac{\sqrt{2}}{20} K_p s \end{aligned} \tag{13}
$$

Considerando que há pelo um polo cuja parte real é nula, pelo critério de estabilidade de Routh-Hurwitz, obtém-se as seguintes condições para a estabilidade a partir da equação (13) para $p = 0$:

$$
K_p + 100K_d + 100 > 0
$$
$$
\left(\frac{\sqrt{2}}{20} + 100\right)K_p + 5\sqrt{2}K_d - \frac{5\sqrt{2}K_p}{K_p + 100 + 100K_d} > 0
$$

Os valores de K_p e K_d para os quais as inequações são válidas estão ilustrados na Fig. 2

Fig. 2. Valores válidos de K_p e K_d conforme o critério de estabilidade de Routh-Hurwitz.



Logo, $\forall K_p, K_d$ pertencente a região escura ilustrada na Fig. 2 o sistema descrito pela equação (11) não possui polos no semiplano direito. Contudo, há um polo cuja parte real é nula. Dessa forma, a estabilidade do sistema linear é marginal e do sistema não linear é local para o ponto de linearização $[x_0, y_0, \phi_0]$.

O controlador ideal deveria realizar a verificação da estabilidade para cada ponto de linearização sobre a trajetória. Caso inválido, os ganhos K_p e K_d podem ser ajustado conforme uma tabela verdade. Nesse trabalho, foi proposta uma solução simplificada. A região de estabilidade do sistema linear foi considerada um círculo de raio 2 metros cuja origem está sobre o ponto de linearização. Dessa forma, toda a trajetória está inclusa sobre a área de estabilidade local do robô.

1.2 Controlador proporcional

Para uma lei de controle cujo sinal de controle do sistema é proporcional ao erro entre a orientação atual do RTD e o ângulo até o ponto alvo, a partir da equação (9), tem-se

$$
\dot{z} = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \nu_0 \cos\phi \\ \nu_0 \sin\phi \\ K_p e \end{bmatrix} \tag{14}
$$
$$
y = \dot{\phi} = Cz + De
$$

Ao linearizar a equação (14) para um ponto de operação $[\dot{x}_0\ \dot{y}_0\ \dot{\phi}_0]^T$, utilizando a equação (2), obtém-se o seguinte espaço de estados

$$\begin{aligned} \dot{z} &= [\dot{x}\ \ \dot{y}\ \ \dot{\phi}]^T = Az + Be \\ y &= \dot{\phi} = Cz' + De \end{aligned} \tag{15}$$

para

$$A = \begin{bmatrix} 0 & 0 & -\nu_0 \sin \phi_0 \\ 0 & 0 & \nu_0 \cos \phi_0 \\ \frac{-K_p(y_0 - y^*)}{\sigma} & \frac{K_p(x_0 - x^*)}{\sigma} & -K_p \end{bmatrix}$$

$$\rightarrow \sigma = (y_0 - y^*)^2 + (x_0 - x^*)^2$$

Para o ponto de operação $[\dot{x}_0\ \ \dot{y}_0\ \ \dot{\phi}_0]^T = [0\ 0\ 0.25\pi]^T$, $\nu_0 = 0.2$ e $x^* = y^* = 2$, o determinante da matriz $(sI - A)$ apresentada na equação (15) é caracterizado pelo polinômio

$$p = s^3 + K_p s^2 + \frac{\sqrt{2} K_p s}{20} \tag{16}$$

Considerando que há pelo um polo cuja parte real é nula, pelo critério de estabilidade de Routh-Hurwitz, obtém-se as seguintes condições para a estabilidade:

$$K_p > 0$$

$$\frac{\sqrt{2} K_p}{20} > 0$$

Logo, $\forall K_p \in \mathbb{R}_+^*$ o sistema descrito pela equação (15) não possui polos no semiplano direito. Contudo, há um polo cuja parte real é nula. Dessa forma, a estabilidade do sistema linear é marginal e do sistema não linear é local para o ponto de linearização $[x_0, y_0, \phi_0]$.

O controlador ideal deveria realizar a verificação da estabilidade para cada ponto de linearização sobre a trajetória. Caso inválido, o ganho K_p pode ser ajustado conforme uma tabela verdade. Nesse trabalho, foi proposta uma solução simplificada. A região de estabilidade do sistema linear foi considerada um círculo de raio 2 metros cuja origem está sobre o ponto de linearização.

2 Desenvolvimento

O espaço de estado representado na equação (9) foi implementado no Matlab. Foi definido um intervalo de tempo para a simulação, a pose inicial do robô e a posição do alvo almejado. O método numérico selecionado foi Runge-Kutta (`ode45`). Esse cenário foi simulado e as 20 posições foram amostradas da trajetória planejada.

Em seguida, foi simulado o RTD *Pioneer P3DX* na plataforma Coppelia. O controlador foi implementado em código Lua com os mesmos ganhos da implementação no Matlab. A simulação foi realizada no modo síncrono para garantir a captura de todas as amostras pelo servidor.

Para todos os experimentos, a pose inicial foi definida em $x_0 = -1,5m$, $y_0 = -1,5m$, $\phi_0 = 225°$ e $r_0 = 0$. Por sua vez, ponto alvo foi definido em $x^* = 2,0m$, $y^* = 2,0m$. Dessa forma, o RTD deve rotacionar π radianos para atingir o alvo. Essa magnitude do ângulo giro permitirá uma melhor comparação dos resultados obtidos a partir da simulação de diferentes ganhos.

A velocidade linear do robô foi definida em $\nu_0 = 0.2m/s$. A escolha desse valor é justificada pela tentativa de não provocar a derrapagem, o qual acontece quando altos torques são impostos ao motor. O critério de parada foi definido a $0.005m$ do ponto alvo. O tempo de simulação foi escolhido 30 segundos.

3 Resultados e discussões

3.1 Controlador proporcional

O primeiro cenário foi configurado para um controlador puramente proporcional. Conforme a Sec. 1.2, adotou-se $K_p = 0.5$ para simulação de um sistema estável.

O resultado da implementação computacional e do RTD simulado na plataforma Coppelia estão apresentados na Fig. 3. A trajetória planejada (*Path planned*) corresponde a simulação do espaço de estados da equação (15). Por sua vez, a trajetória de simulação corresponde a trajetória percorrida pelo *P3DX* simulado no CoppeliaSim.

A diferença entre a trajetória planejada e simulada é justificada pelas considerações na modelagem do sistema no Matlab e no Coppelia. A plataforma Coppelia considera as forças dissipativas de atrito sobre as rodas e o torque imposto pela a rotação da roda de apoio caster (*swivel caster wheel*).

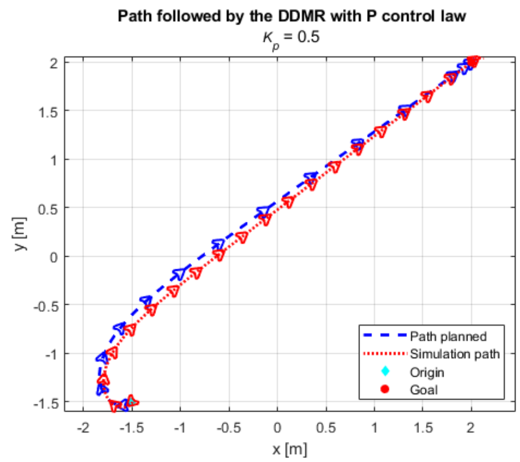
Essas forças são desprezadas na modelagem de uniciclo implementada no Matlab, a qual é fundamentada em um modelo cinemático com velocidade linear constante. Nesse modelo, há um único ponto de contato entre a roda e a superfície.

O modelo do RTD implementado no CoppeliaSim, diferentemente do implementado no Matlab, introduz a saturação da saída da planta para $\omega_{sat} = 2,4\,rad/s$. Logo, as velocidades de rotação de cada motor são limitadas no intervalo $[-4,69;\ 4,69]\,rad/s$. As velocidades angulares de cada roda são calculadas por

$$\dot{\phi_D} = \frac{2\nu_0 + \dot{\phi} L}{2R} \qquad \dot{\phi_E} = \frac{2\nu_0 - \dot{\phi} L}{2R}$$

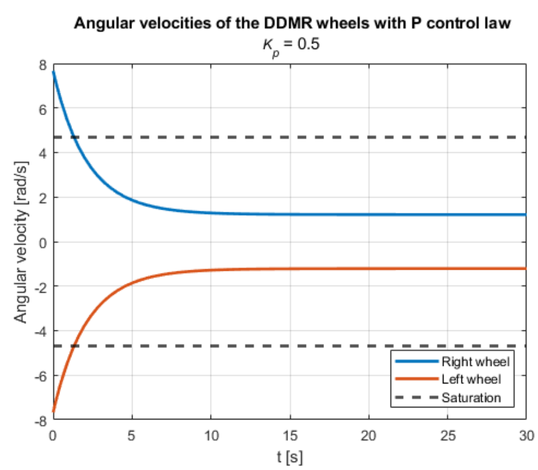
para R o raio da roda, L a distância entre as rodas, $\dot{\phi}_D$ e $\dot{\phi}_E$ as velocidades angulares da roda direita e esquerda do RTD, respectivamente.

Fig. 3. Trajetória planejada e percorrida percorrida pelo robô com um controlador P para $K_p = 0.5$.



Na Fig. 4 estão ilustradas as velocidades angulares das rodas adquiridas pela simulação no Matlab e o nível de saturação do modelo do *P3DX* no Coppelia.

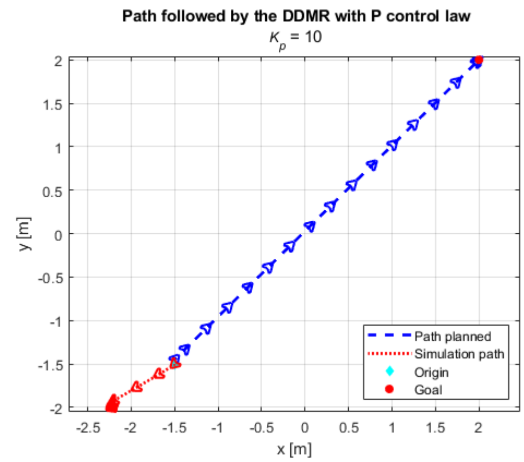
Fig. 4. Velocidades angulares do robô simulado no Matlab com um controlador P para $K_p = 0.5$.



Tendo em vista a não saturação das velocidades angulares das rodas, é claro que a trajetória calculada pelo modelo no Matlab será mais rápida do que a simulada no Coppelia. Dessa forma, conforme ilustrado na Fig. 3, o intervalo entre as amostras percurso planejado é maior que as posições recuperadas do CoppeliaSim.

O segundo cenário de teste foi composto pela modificação do ganho $K_p = 10$. Com essa alteração, é prevista a execução de uma trajetória mais rápida, uma vez que a entrada realimentada é maior em magnitude ao valor para $K_p = 0.5$. O resultado da implementação computacional e do RTD simulado no Coppelia estão apresentados na Fig. 5.

Fig. 5. Trajetória planejada e percorrida percorrida pelo robô com um controlador P para $K_p = 10$.



Observando a Fig. 5, é possível visualizar que o robô simulado no CoppeliaSim não conseguiu realizar a curva. Tendo em vista que esse modelo considera a dimensão do eixo entre as rodas, é possível concluir que o robô não consegue girar π radianos no intervalo tempo no qual o robô ideal alinha-se com o alvo na trajetória planejada. Essa situação é explicada pela derrapagem do RTD simulado no Coppelia quando a velocidade de rotação imposta sobre o motor é maior que a força de atrito que age sobre a roda.

Essa consideração é desprezada pela trajetória planejada no Matlab, tendo em vista que essa modelagem é baseada no uniciclo ideal. Para este último caso, é considerado que o RTD consegue rotacionar e alinhar-se com o alvo em um curto período de tempo.

3.2 Controlador proporcional-derivativo

O terceiro cenário foi configurado para um controlador proporcional-derivativo. Adotou-se $K_p = 0.5$, $K_d = 1$ e $c = 100$. Segundo a Fig. 2, o sistema de controle é estável.

O resultado da implementação computacional e do RTD simulado na plataforma Coppelia estão apresentados na Fig. 6. Na Fig. 7 estão ilustradas as velocidades angulares das rodas adquiridas pela simulação no Matlab e o nível de saturação do modelo do *P3DX* no Coppelia.

É esperado que o controle PD seja mais lento que o controle tipo P, uma vez que há um amortecimento maior do sinal de controle. Comparando a Fig. 7 com a Fig. 4, é confirmada essa hipótese, uma vez que a velocidade angular de cada roda é menor para a lei de controle PD em relação ao controlador P para $t > 5s$.

Para $0 < t < 5s$ o robô está executando o trecho curvo da trajetória. Nesse período, a derivada do erro e é alta e contribuí, a partir de K_d, no aumento da magnitude do sinal de entrada da planta quando com o controlador PD. Para $0 < t < 5s$, Tendo em vista que esse termo é nulo para o controlador P, a velocidade do sistema com lei de controle PD é maior que para a lei de controle P.

Fig. 6. Trajetória planejada e percorrida percorrida pelo robô com um controlador PD para $K_p = 0.5$ e $K_d = 1$.

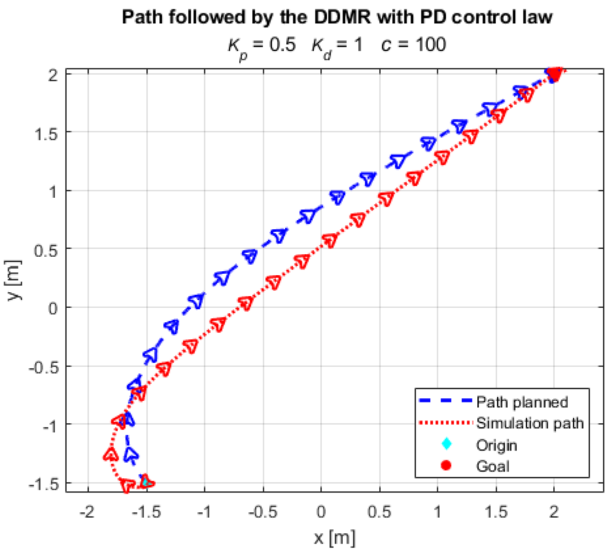
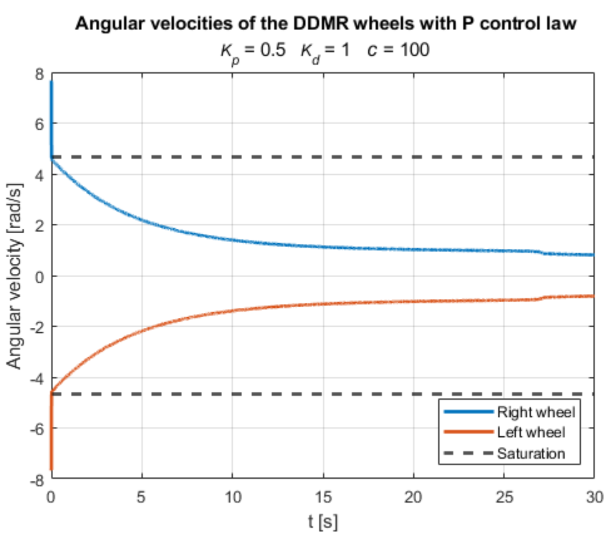


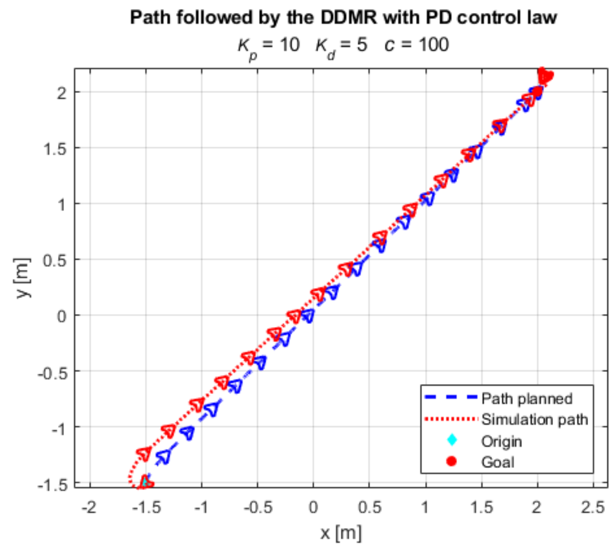
Fig. 7. Velocidades angulares do robô simulado no Matlab com um controlador PD para $K_p = 0.5$ e $K_d = 1$.



A maior abertura da curva da trajetória planejada em relação a simulada é também justificável pela menor velocidade angular do robô, a qual atende a saturação imposta pelo Coppelia. Nesse último há uma menor variação do erro $\phi^* - \phi$ em relação ao modelo ideal implementado no Matlab. Consequentemente, há uma menor contribuição do termo derivativo no sinal de controle.

Para $K_d = 5$ e $K_p = 10$, o resultado da implementação computacional e do RTD simulado na plataforma Coppelia estão apresentados na Fig. 8. Segundo a Fig. 2, o sistema de controle é estável.

Fig. 8. Trajetória planejada e percorrida percorrida pelo robô com um controlador PD para $K_p = 10$ e $K_d = 5$.



Comparando a Fig. 8 com a Fig. 5, o termo derivativo para o mesmo ganho proporcional contrapõe os altos torque sobre o motor, contendo o deslize do RTD.

Comparando a Fig. 8 com a Fig. 7, é claro que o ângulo de abertura da curva planejada é, nesta última, menor que a curva na trajetória simulada no CoppeliaSim. Essa troca em relação a Fig. 7 se deve a capacidade do robô no modelo de uniciclo implementado no Matlab rotacionar π radianos para um alto ganho proporcional.

Conforme a Fig. 2, para $K_p = 0.5$ e $K_d = -1.5$ o sistema é instável. O resultado da implementação computacional e do RTD simulado na plataforma Coppelia para esses ganhos estão apresentados na Fig. 9. Fica claro que o controlador não é capaz de rastrear a referência.

Também segundo a Fig. 2, para $K_p = 0.5$ e $K_d = -1$ o sistema é estável. O resultado da implementação computacional e do RTD simulado na plataforma Coppelia para esses ganhos estão apresentados na Fig. 10. Enquanto o controlador simulado no Matlab é capaz de rastrear a referência, a trajetória do robô no Coppelia caracteriza um sistema instável.

Entretanto, comparando as curvas em vermelho da Fig. 9 e Fig. 10, verifica-se que o comportamento do robô no Coppelia não é uma trajetória em espiral como no sistema instável.

Fig. 9. Trajetória planejada e percorrida percorrida pelo robô com um controlador PD para $K_p = 0.5$ e $K_d = -1$.

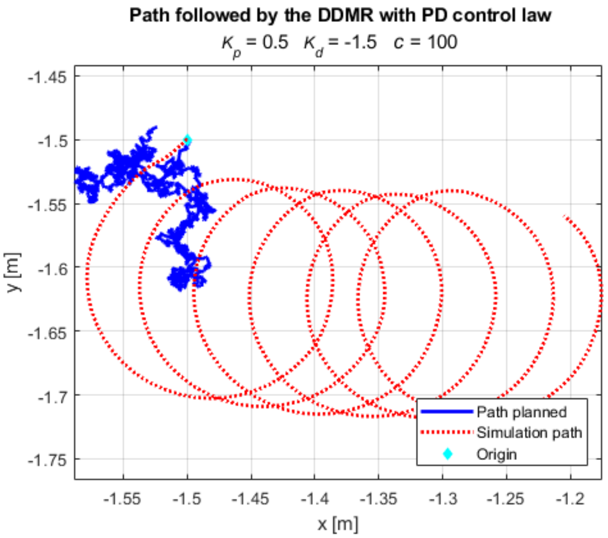
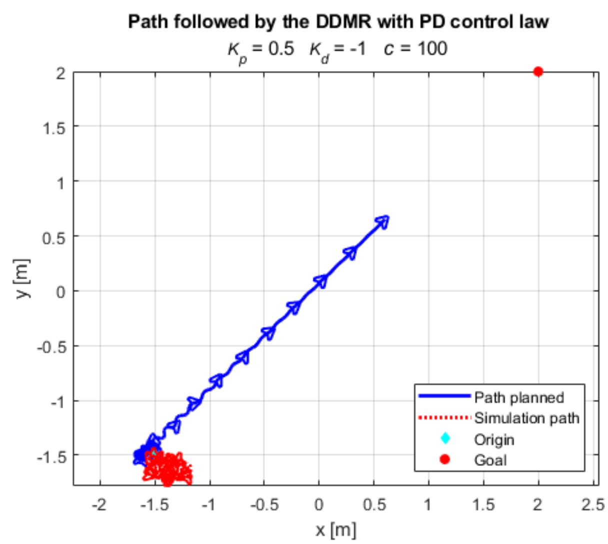


Fig. 10. Trajetória planejada e percorrida percorrida pelo robô com um controlador PD para $K_p = 0.5$ e $K_d = -1.5$.



Ao observar a simulação, foi verificado que o corpo do robô colidia com a parede, uma vez que buscava realizar a curva pelo lado esquerdo. Caso esse bloquei fosse removido do cenário, o robô poderia seguir até o alvo.

4 Conclusões

Esse relatório descreve a implementação e análise da do modelo de um robô de tração diferencial controlado por leis P e PD no Matlab. A partir da comparação dos resultados obtidos pela solução numérica com a simulação do mesmo sistema na plataforma CoppeliaSim, concluiu-se que o modelo implementado no Matlab é ideal: são desprezíveis as contribuições de força pela roda acessória caster e forças de atrito da superfície e a distância do eixo entre as rodas. Para a aproximação dos resultados entre os dois modelos, deve-se acrescentar na modelagem no Matlab a consideração do momento de inércia do corpo. A partir dos resultados obtidos para os ganhos válidos K_p, K_d e para a frequência de corte do filtro derivativo c, também é clara que a modelagem do controlador P e PD desenvolvida na Sec.1 são válidas para o modelo de um uniciclo. Por fim, é importante destacar que o sistema implementado no Matlab é não linear, enquanto o executado pelo CoppeliaSim é linear. Logo, ao passo que o robô simulado é distanciado do ponto de operação da linearização, é necessário recalcular os ganhos do controlador.