Sintonia de Ganhos para Leis de Controle Não-Lineares em Robôs Móveis de Tração Diferencial

Matheus Lucas Tavares de Farias

11 de Agosto de 2025

Automação Inteligente

Introdução

Introdução — Problema

- Desempenho do controle depende da sintonia adequada dos ganhos.
- Ganhos influenciam:
  - Estabilidade
  - Desempenho
  - Resposta a perturbações
- Sintonia exige considerar:
  - Critérios de desempenho
  - Limitações físicas do sistema (velocidades máxima linear e angular)

Introdução — Objetivos e Metodologia

- Objetivo: ajuste de ganhos para três leis de controle não-lineares.
- Comparação de abordagens:
  - Otimização não-linear com restrições via *fmincon*
  - Algoritmo genético *SpeedyGA*
- Implementação:
  - Linguagem: Python
  - Modelo: *unicycle*
  - Robô de referência: Pioneer 3-DX

Métodos de Otimização

Métodos de Otimização

- Processo de encontrar a melhor solução para um problema, respeitando restrições e critérios.
- Envolve ajustar variáveis de decisão para:
  - Minimizar ou maximizar uma função objetivo (custo, erro, desempenho).
- Aplicações na sintonia de controladores:
  - Evita tentativa e erro.
  - Busca soluções de forma sistemática e eficiente.
- Neste trabalho:
  - Método tradicional baseado em gradiente (`scipy.optimize.minimize`)
  - Método heurístico (*SpeedyGA*)

Otimização com Restrições — SLSQP (fmincon)

- Implementado em Python com `scipy.optimize.minimize` (SLSQP).
- Resolve problemas de otimização contínua com restrições.
- Minimiza função custo que representa:
  - Erro de posição
  - Tempo de chegada ao destino
  - Combinação ponderada desses fatores
- Restrições:
  - Limites físicos e operacionais.
  - Condições de estabilidade e segurança.
- Vantagens:
  - Simples e rápido para espaços de busca bem comportados.
- Limitação: sensível a mínimos locais.

Algoritmos Genéticos — Conceitos

- Inspirados na seleção natural e evolução biológica.
- Operam sobre população de soluções candidatas.
- Etapas principais:
  1. Inicialização
  2. Avaliação (*fitness*)
  3. Seleção
  4. Cruzamento (*crossover*)
  5. Mutação
  6. Substituição
- Garantem diversidade e evitam soluções subótimas.

SpeedyGA — Aplicação no Trabalho

- Baseado na implementação de Burjorjee.
- Adaptado de MATLAB para Python.
- Cada indivíduo: sequência binária representando ganhos do controlador.
- Vantagens:
  - Busca global.
  - Pouco sensível a mínimos locais.
  - Não requer derivadas.
- Limitação: maior custo computacional em simulações complexas.

Modelo Cinemático e Leis de Controle

Modelo Cinemático — Unicycle

- Representa robôs móveis de tração diferencial.
- Captura restrições não-holonômicas.
- Equações do modelo:

$$\begin{cases} \dot{x} = v\cos\theta \\ \dot{y} = v\sin\theta \\ \dot{\theta} = \omega \end{cases}$$

- Estados: (x, y, θ) — posição e orientação.
- Entradas de controle:
  - Velocidade linear v
  - Velocidade angular ω

Variáveis de Erro

- Controle *go-to-goal* baseado no referencial do robô.
- Alvo: (x_g, y_g).
- Variáveis:

$$\begin{cases} e = \sqrt{(x_g - x)^2 + (y_g - y)^2} \\ \alpha = \arctan\left(\dfrac{y_g - y}{x_g - x}\right) - \theta \end{cases}$$

- e: erro de posição (distância até o alvo).
- α: erro de orientação (ângulo em relação ao alvo).

Lei de Controle — Aicardi et al. (1995)

$$\begin{cases} v = \gamma e \cos\alpha \\ \omega = k\alpha + \dfrac{v}{e\alpha} \sin\alpha \left(\alpha + h(\alpha + \theta)\right) \end{cases}$$

- Parâmetros de controle:
  - γ
  - k
  - h
- Objetivo: garantir convergência do robô ao alvo com estabilidade.

Leis de Controle — Karim (2013) e Breno (2023)

Karim et al. (2013):

$$\begin{cases} v = K_v e \cos\alpha \\ \omega = K_\omega \alpha + \dfrac{v}{e} \sin\alpha \end{cases}$$

- Parâmetros: K_v, K_ω

Breno et al. (2023):

$$\begin{cases} v = \tau e \cos\alpha \\ \omega = \kappa \alpha + \dfrac{v}{e} \sin\alpha \end{cases}$$

- Parâmetros: τ, κ

Resultados

Configuração dos Testes

- Definições para execução:
  - Cena de teste padronizada.
  - Restrições físicas do Pioneer 3-DX:
    - Velocidade linear máx: $v_{max} = 1.2$ m/s
    - Velocidade angular máx: $\omega_{max} = 300^{\circ}/s$
  - Intervalo de ganhos: $[0, 4]$

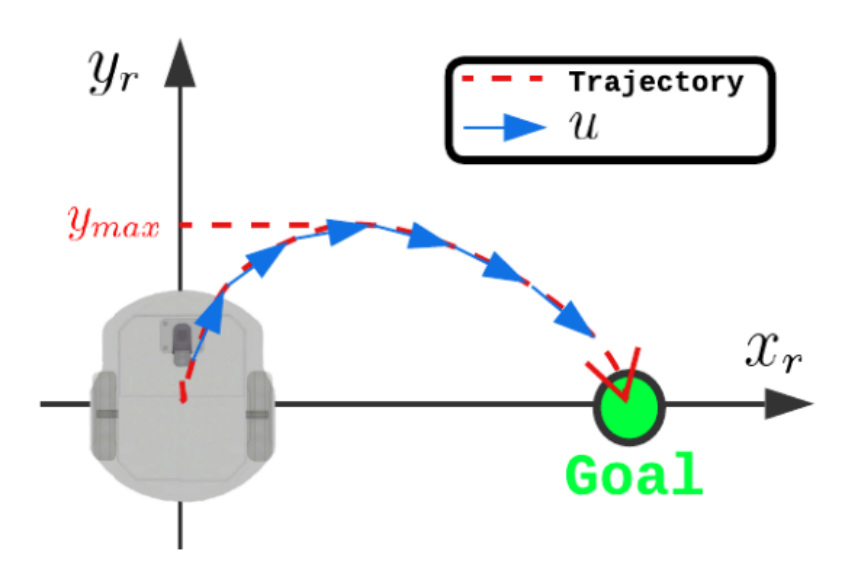


Figura 1: Cena de teste utilizada

SpeedyGA — Conversão de Representação

- No SpeedyGA, cada indivíduo é codificado como uma sequência binária.
- Conversão para valor real:

$$x_{real} = \frac{4}{2^{n_{bits}} - 1} \cdot x_{inteiro}$$

- Configuração utilizada:
  - $n_{bits} = 20$ (por ganho), conforme.
  - Representação inteira sem sinal.
  - Mapeamento para faixa de $[0, 4]$.
- Vantagem: flexibilidade para representar valores contínuos.

Funções Objetivo e Ajuste

- Função objetivo definida para cada abordagem, considerando restrições do modelo.
- Diferença entre métodos:
  - **Otimização tradicional** (SLSQP): função de custo $\rightarrow$ valores menores indicam melhor desempenho.
  - **Algoritmo genético** (SpeedyGA): função de ajuste (*fitness*) $\rightarrow$ valores maiores indicam melhor desempenho.
- Relação geral:

$$\text{Função de custo} \approx -\text{Função de ajuste}$$

- Permite comparação coerente entre métodos diferentes.

Aicardi et al. — Funções Objetivo

- Referência não define método de otimização.
- Função de ajuste definida neste trabalho:

$$F = \frac{1}{|\max(v) - v_{ref}|} + \frac{1}{|\max(\omega) - \omega_{ref}|}$$

- Objetivo: manter velocidades máximas próximas aos valores de referência:

$$v_{ref} = 1.2\,\mathrm{m/s}, \quad \omega_{ref} = 300^{\circ}/\mathrm{s}$$

- Função de custo (para *fmincon*):

$$J = -F$$

Aicardi et al. — Configuração do SpeedyGA

- População: 200 indivíduos
- Gerações: 40
- Probabilidade de mutação: 3%
- Probabilidade de *crossover*: 100%

Aicardi et al. — Ganhos Obtidos e Máximos

Método	γ	k	h
SpeedyGA	1.63	3.33	3.47
fmincon	3.38	2.74	2.26

Método	$\max(v)$	$\max(\omega)$
SpeedyGA	1.20	300.00
fmincon	2.17	300.00

Aicardi et al. — Trajetórias

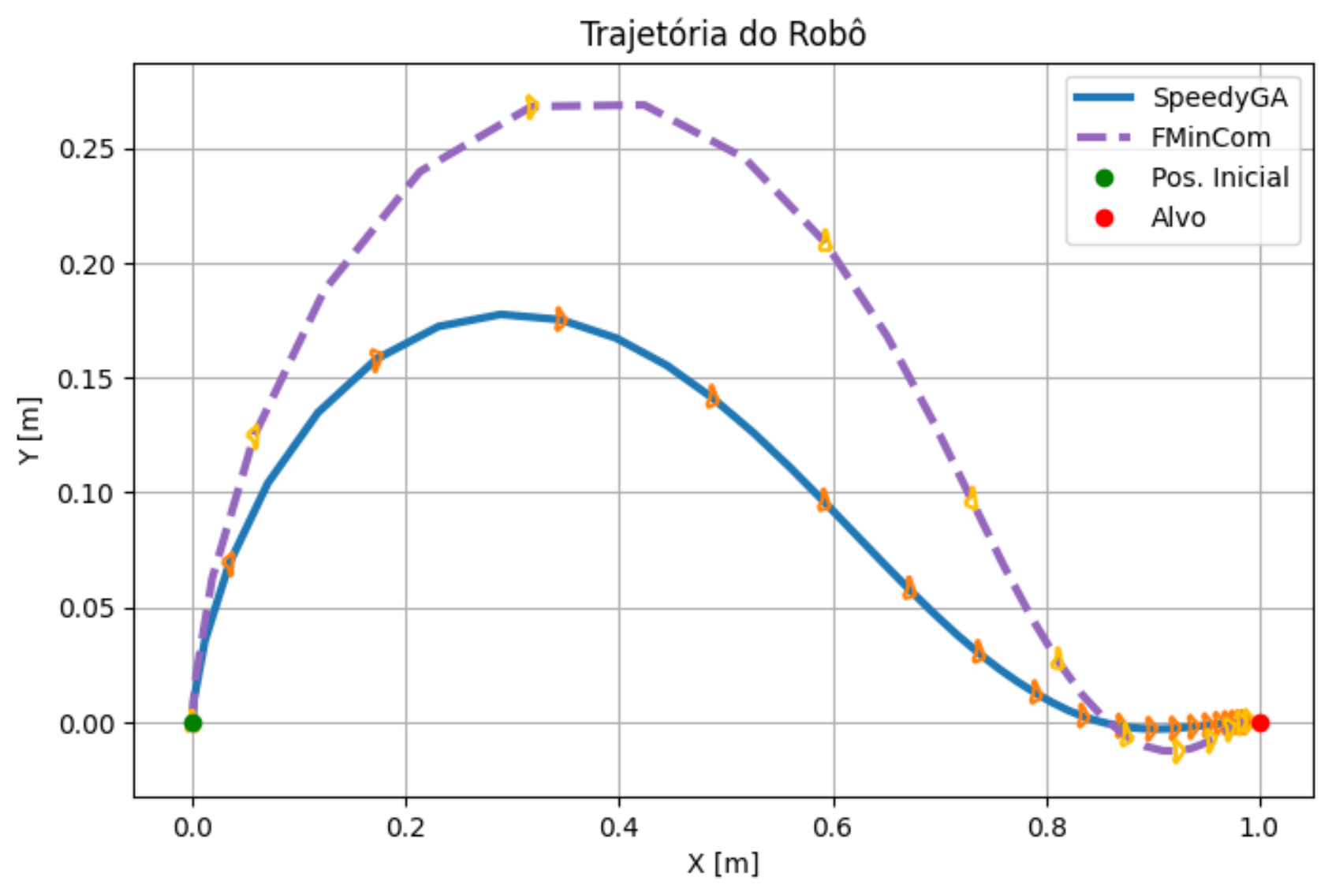


Figura 2: Trajetórias obtidas

Benbouabdallah et al. — Função de Custo Original

- Referência propõe:

$$J = \frac{1}{2} \int \left(\frac{e}{e_o} \right)^2 + \left(\frac{\alpha}{\alpha_o} \right)^2 dt$$

- Onde:
  - e_o: erro inicial de posição
  - α_o: erro inicial de orientação
- Objetivo: minimizar erros de posição e orientação no menor tempo.
- Limitação: não considera restrições físicas do robô.

Benbouabdallah et al. — Função de Custo Modificada

- Acrescentados termos para considerar limites físicos:

$$J = \frac{1}{2}\int \left(\frac{e}{e_o}\right)^2 + \left(\frac{\alpha}{\alpha_o}\right)^2 dt - \frac{1}{|\max(v) - v_{ref}|} - \frac{1}{|\max(\omega) - \omega_{ref}|}$$

- Função de ajuste associada:

$$F = -J$$

- Valores de referência:

$$v_{ref} = 1.2\,\mathrm{m/s}, \quad \omega_{ref} = 300^\circ/\mathrm{s}$$

Benbouabdallah et al. — Configuração

- **Configuração do SpeedyGA:**
  - População: 100 indivíduos
  - Gerações: 100
  - Mutação: 3%
  - *Crossover*: 100%

Benbouabdallah et al. — Ganhos e Máximos

Método	K_v	K_ω
SpeedyGA	1.70	3.33
fmincon	1.87	2.21

Método	$\max(v)$	$\max(\omega)$
SpeedyGA	1.20	300.00
fmincon	1.20	199.00

Benbouabdallah et al. — Trajetórias

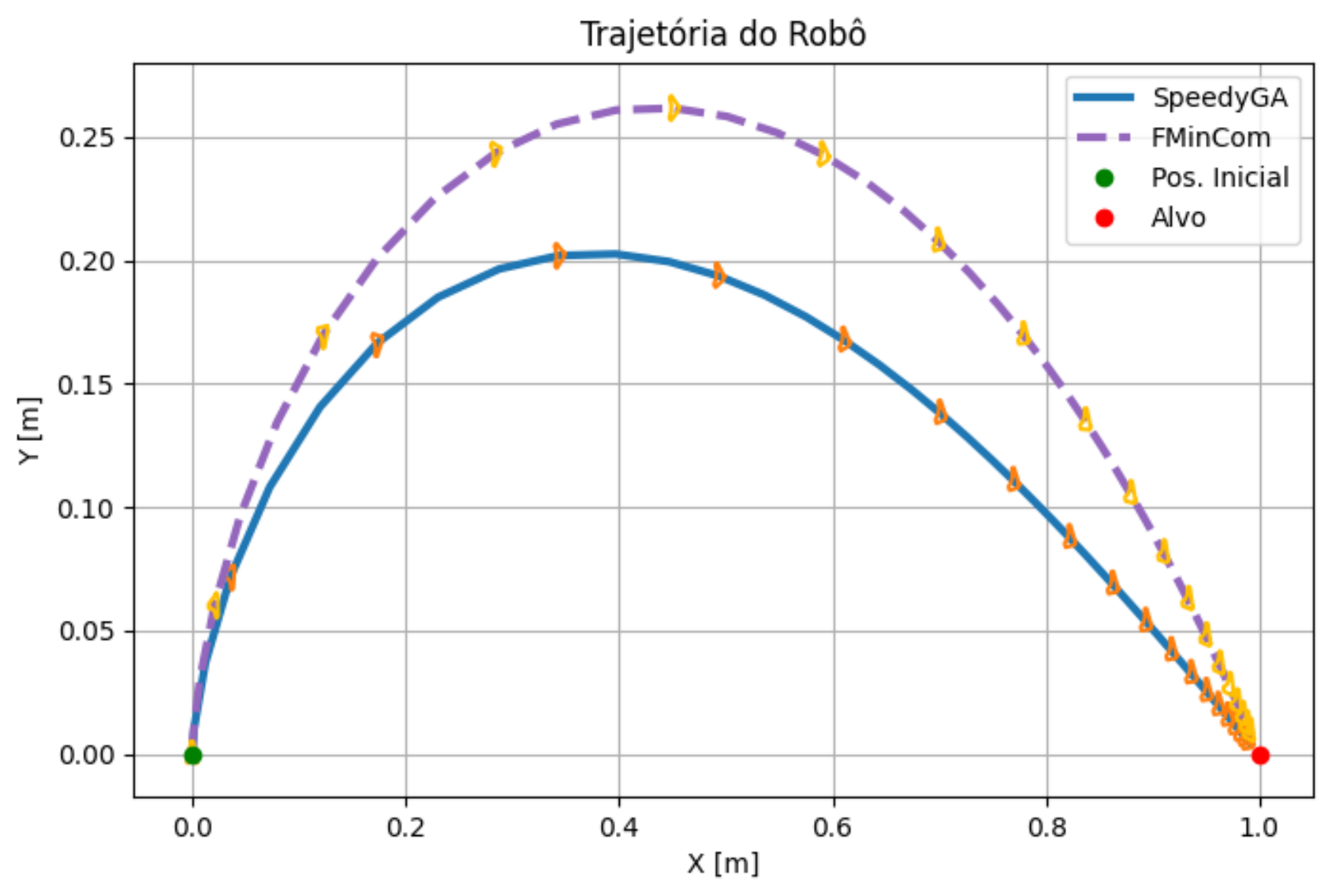


Figura 3: Trajetórias obtidas

Abordagem de Breno et al. - Função de Ajuste

Função de ajuste original:

$$F = \frac{1}{|\max(y) - y_{ref}|} + \frac{1}{|\max(v) - v_{ref}|}$$

- Objetivo: manter $\max(y)$ e $\max(v)$ próximos dos valores de referência:

$$v_{ref} = 1.2 \text{ m/s}, \quad y_{ref} = 0.15 \text{ m}$$

- Limitação: não considera a velocidade angular.

Abordagem de Breno et al. - Função de Ajuste Modificada

Função de ajuste modificada:

$$F = \frac{1}{|\max(y) - y_{ref}|} + \frac{1}{|\max(v) - v_{ref}|} + \frac{1}{|\max(\omega) - \omega_{ref}|}$$

- Agora inclui a restrição de $\max(\omega)$.
- Função de custo:

$$J = -F$$

Abordagem de Breno et al. - Configuração

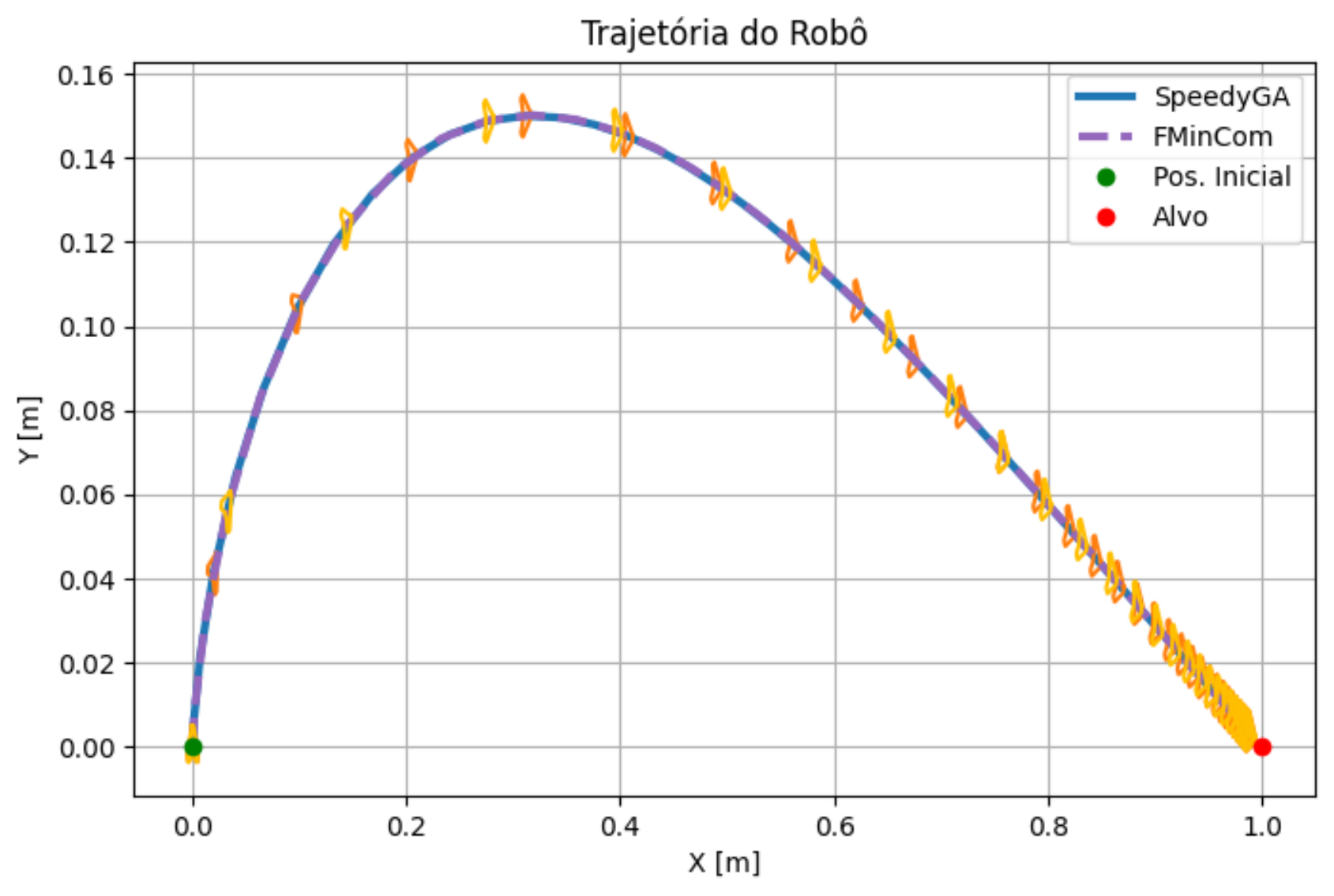
- **Configuração do SpeedyGA:**
  - População: 200 indivíduos
  - Gerações: 40
  - Mutação: 3%
  - Crossover: 100%

Abordagem de Breno et al. - Ganhos obtidos e Máximos

Método	τ	κ
SpeedyGA	0.97	3.24
fmincon	1.21	3.96

Método	max(y) [m]	max(v) [m/s]	max(ω) [º/s]
SpeedyGA	0.15	0.76	291.83
fmincon	0.15	0.93	356.32

Trajetórias - Abordagem de Breno et al.



Conclusão

Conclusão

- Os ganhos obtidos pelos diferentes métodos apresentaram discrepâncias, mas isso não representa necessariamente um problema.
- Diversas combinações de ganhos podem gerar desempenhos equivalentes.
- A maioria das soluções manteve o comportamento do modelo dentro dos limites físicos.
- Exceção: *fmincon* na abordagem de Breno *et al.* excedeu o limite de velocidade angular.

Conclusão

- O bom desempenho geral está associado à escolha adequada das funções objetivo.
- Essas funções induziram a consideração das restrições físicas no processo de otimização.
- Fora a exceção citada, ambos os algoritmos mostraram eficácia equivalente.
- *SpeedyGA* apresentou menor tempo para alcançar o alvo, mas sem impacto significativo no desempenho global.

Conclusão

- O algoritmo genético demanda mais tempo de execução que o baseado em gradiente.
- Quando a otimização é feita *offline*, essa diferença não afeta o uso prático.
- A escolha do algoritmo deve considerar:
  - Tempo de execução
  - Robustez frente a mínimos locais
  - Familiaridade com as ferramentas

Referências

Referências

- Aicardi, M., Casalino, G., Bicchi, A., & Balestrino, A. (1995). *Closed loop steering of unicycle-like vehicles via Lyapunov techniques.* IEEE Robotics & Automation Magazine, 2(1), 27–35.
- Benbouabdallah, K., & Zhu, Q. (2013). *Improved Genetic Algorithm Lyapunov-Based Controller for Mobile Robot Tracking a Moving Target.* Research Journal of Applied Sciences, Engineering and Technology, 5(15), 4023–4028.
- de Meneses, B. P., da Silva, G. H. V., Sobral, L. R., Marques, M. S., de Araujo, R. T., & Lima, A. M. N. (2023). *Navigation of a two-wheel differential drive robot in a partially unknown environment.* In Simpósio Brasileiro de Automação Inteligente (SBAI), 1(2).

Referências (cont.)

SciPy Community. (2025).
scipy.optimize.minimize – Minimization of scalar function of one or more variables.
Disponível em: `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.minimize.html`
Acessado em: 06 ago. 2025.

Burjorjee, K. (2025).
SpeedyGA: A Fast Simple Genetic Algorithm.
MATLAB Central File Exchange. Disponível em: `https://www.mathworks.com/matlabcentral/fileexchange/15164-speedyga-a-fast-simple-genetic-algorithm`
Acessado em: 06 ago. 2025.

Obrigado!

Perguntas?