



INSTITUTO FEDERAL
ESPIRITO SANTO



Robótica Móvel

Engenharia de Controle e Automação – 7º Período

PROF. LUCAS VAGO SANTANA

lucas@ifes.edu.br



Aula 03

- Robôs móveis de tração diferencial:
 - Fundamentos de navegação
 - Modelagem
 - Simulação
 - Controle



Referências Bibliográficas

- EGERSTEDT, Magnus. [Control of Mobile Robots](#). Georgia Institute of Technology – Coursera. 2013. Acesso em 28/02/2019.
- HELLSTRÖM, Thomas. [Kinematics Equations for Differential Drive and Articulated Steering](#). Department of Computing Science. Umeå University . 2011. Acesso em 02/04/2019.
- SAKAI, Atsushi; et. al. [PythonRobotics: a Python code collection of robotics algorithms](#). arXiv:1808.10703. 2018. Acesso em 02/04/2019.
- MALU, S. K.; MAJUMDAR, J. [Kinematics, Localization and Control of Differential Drive Mobile Robot](#). *Global Journal of Researches in Engineering: (H) Robotics & Nano-Tech*, USA, v. 14, n. 1, 2014. Acesso em 09/04/2019.
- OLSON, E. [A Primer on Odometry and Motor Control](#). MIT. 2009. Acesso em 08/04/2019.
- CONDIT, R.; JONES, D. W. [AN907 Stepping Motors Fundamentals – Applications Note – Microchip](#). Acesso em 15/04/2019.



INSTITUTO FEDERAL
ESPIRITO SANTO

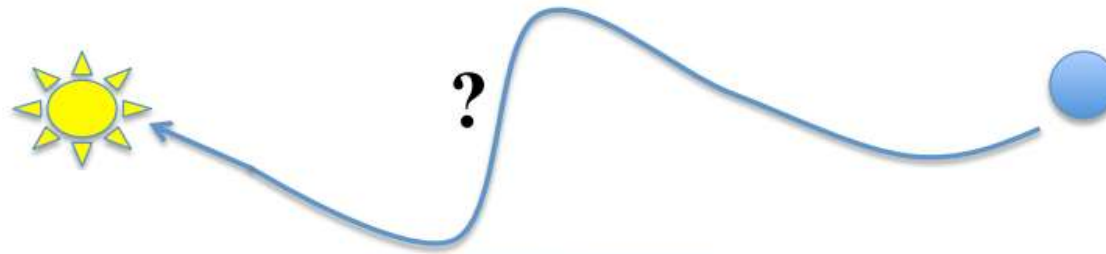


Fundamentos de Navegação



Navegação de Robôs Móveis

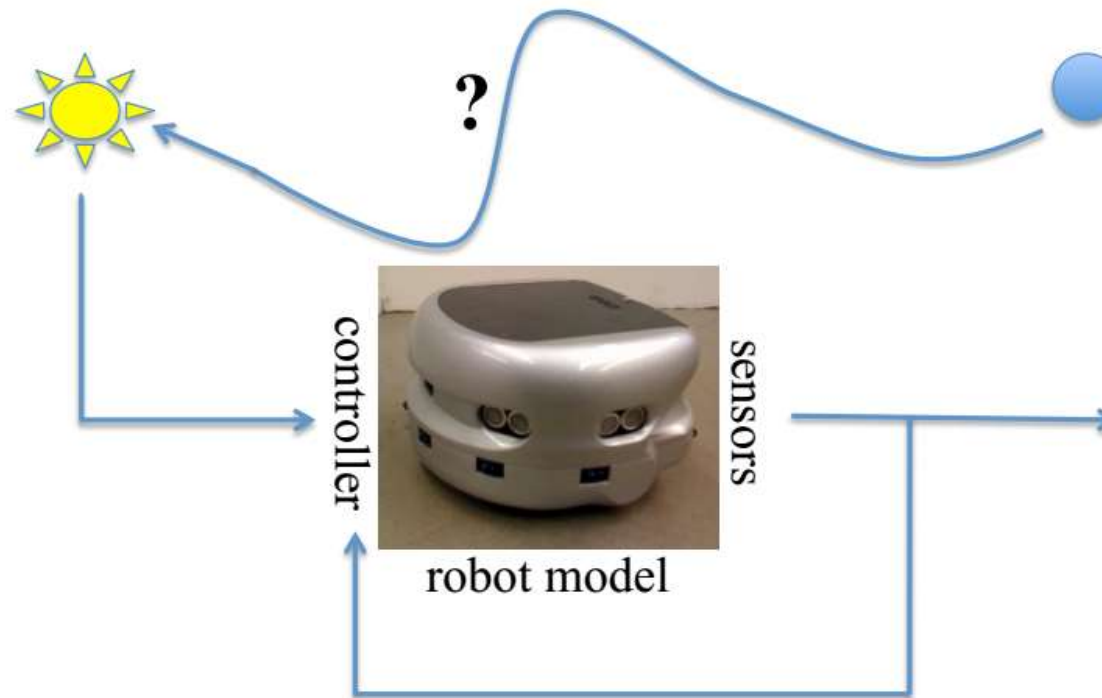
- O que é necessário para levar um robô móvel do ponto A até o B?





Navegação de Robôs Móveis

- Fundamentalmente, apenas 3 itens:





Navegação de Robôs Móveis

- Entretanto, os robôs operam em ambientes **dinâmicos**:
 - Casas;
 - Laboratórios;
 - Indústrias;
 - Florestas;
 - Mar;
 - Etc ...
- E seus controladores devem responder a possíveis **mudanças no ambiente**;



Navegação de Robôs Móveis

- Projetar um único controlador que realize todas as tarefas de um robô é inviável;
- É mais comum **dividir** o problema de controle em diferentes **comportamentos**:
 - Ir de um ponto a outro;
 - Evitar obstáculos;
 - Seguir parede;
 - Rastrear um alvo;
 - ...
- E implementar controladores diferentes para cada caso.



Navegação de Robôs Móveis

- Vídeo contendo exemplo de controle baseado em comportamentos:
 - *Behavior based robot design software progress - rectangle bot*
 - <https://www.youtube.com/watch?v=e2Ag9PaM8NU>
 - Acesso em 01/04/2019



INSTITUTO FEDERAL
ESPIRITO SANTO



Modelagem e Simulação



Robôs de Tração Diferencial

- Arquitetura extremamente comum:



Khepera



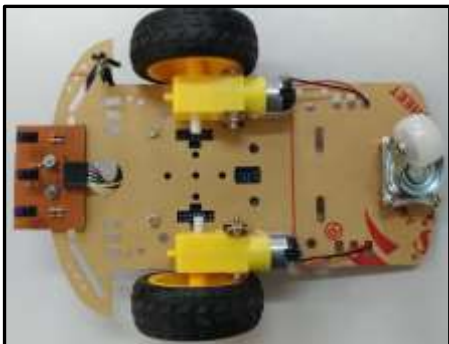
E-puck



Roomba



Pioneer



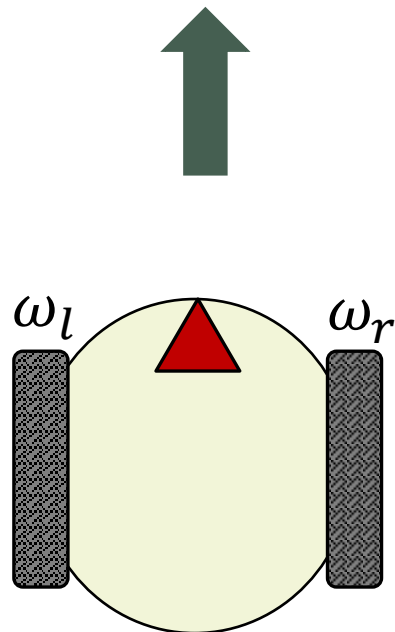
Kits para Arduino/Raspberry Pi



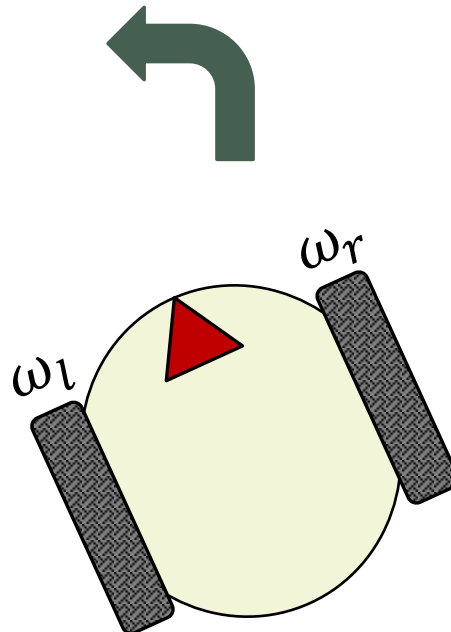
Robôs de Tração Diferencial

- Princípio de funcionamento:

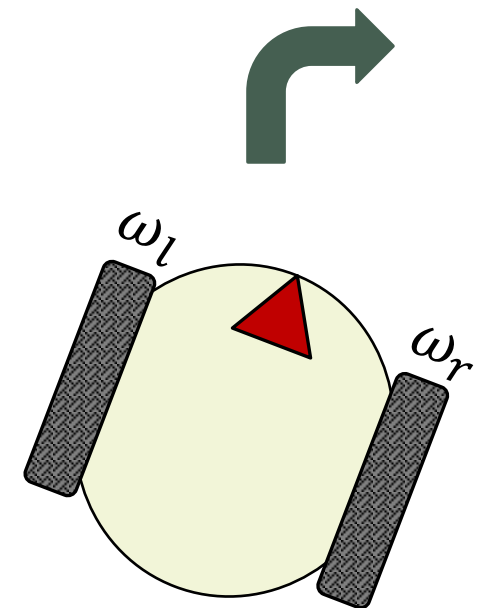
ω_l - velocidade angular da roda esquerda [rad/s]
 ω_r - velocidade angular da roda direita [rad/s]



$$\omega_l = \omega_r$$



$$\omega_l < \omega_r$$



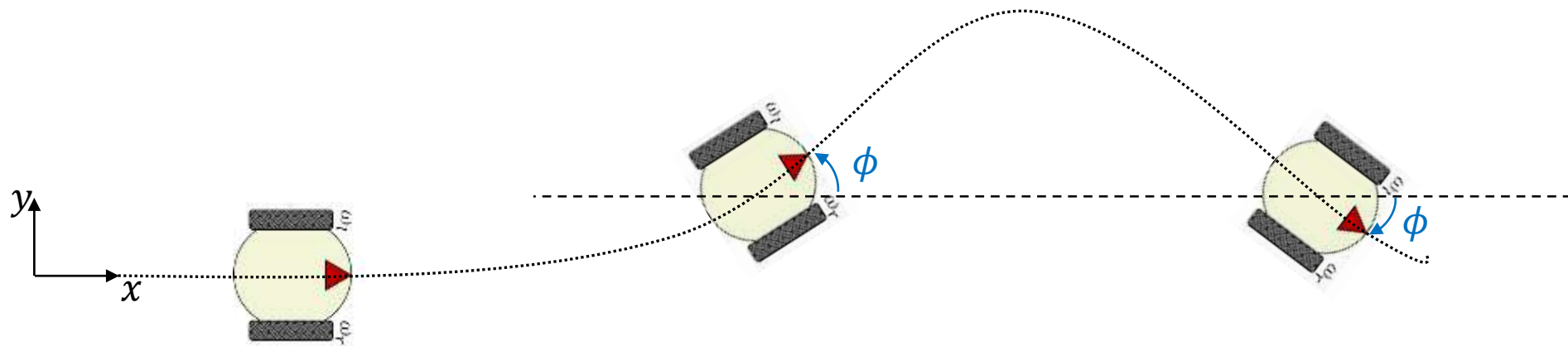
$$\omega_l > \omega_r$$



Robôs de Tração Diferencial

- O que significa controlar um robô deste tipo?
- Significa alcançar **poses desejadas** no espaço bidimensional;

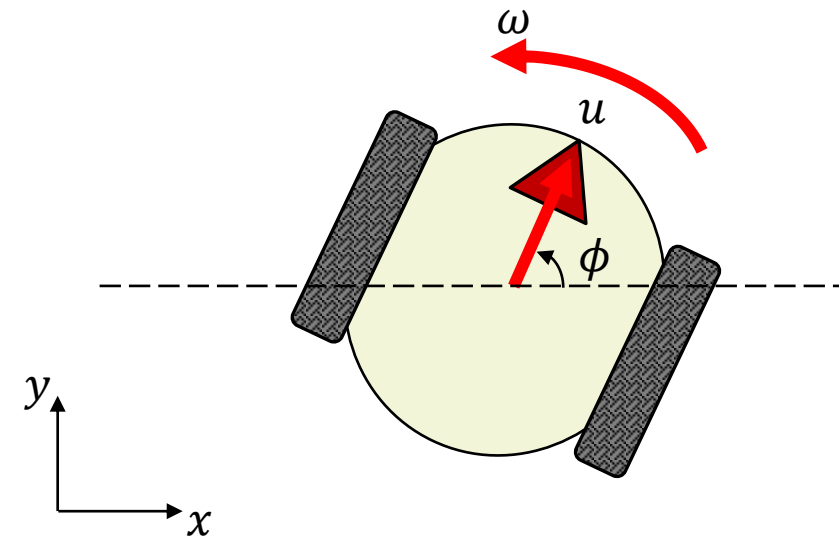
$$p_d = [x_d \quad y_d \quad \phi_d]^T$$





Robôs de Tração Diferencial

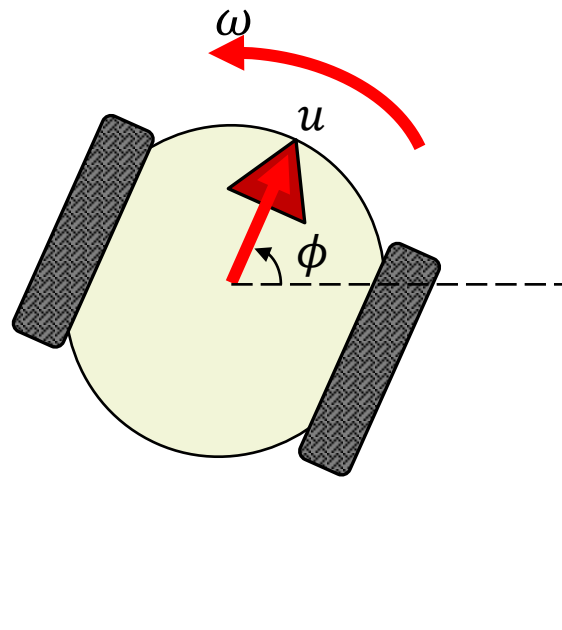
- **PROBLEMA:** pensar no movimento cartesiano do robô a partir das velocidades angulares das rodas, **não é intuitivo**;
- **SOLUÇÃO:** Para fins de controle, utiliza-se um modelo simplificado denominado **uniciclo**;
- Nele, considera-se que os estados do robô são controlados pelas velocidades:
 - u – velocidade linear do robô na direção do nariz;
 - ω – velocidade angular do robô em torno de seu próprio eixo;





Robôs de Tração Diferencial

- Equações do unicycle:



Modelo Uniciclo:

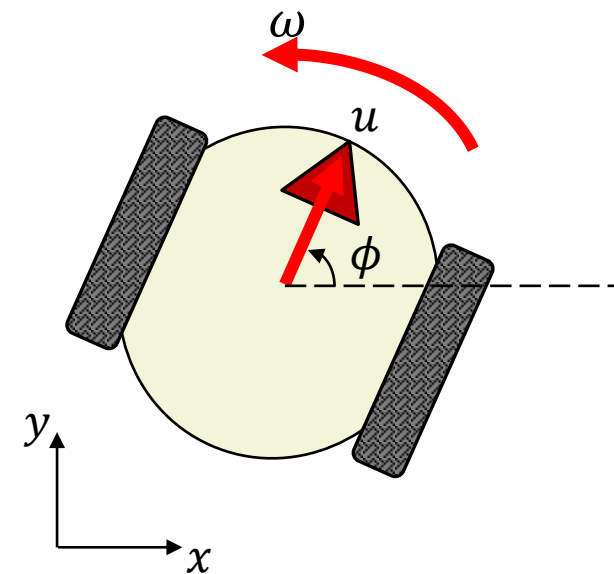
$$\begin{cases} \dot{x} = u \cos\phi \\ \dot{y} = u \sin\phi \\ \dot{\phi} = \omega \end{cases}$$



Robôs de Tração Diferencial

- O projeto de controladores consiste em selecionar os sinais u e ω para alcançar $p_d = [x_d \ y_d \ \phi_d]^T$, observando que:
 - A relação de \dot{x} e \dot{y} com u não é linear;
 - A relação de $\dot{\phi}$ com ω é linear;
 - Há relação entre ϕ , \dot{x} e \dot{y} ;
 - Há relação entre u , \dot{x} e \dot{y} ;
- Características que impactam na escolha da técnica de projeto adequada;

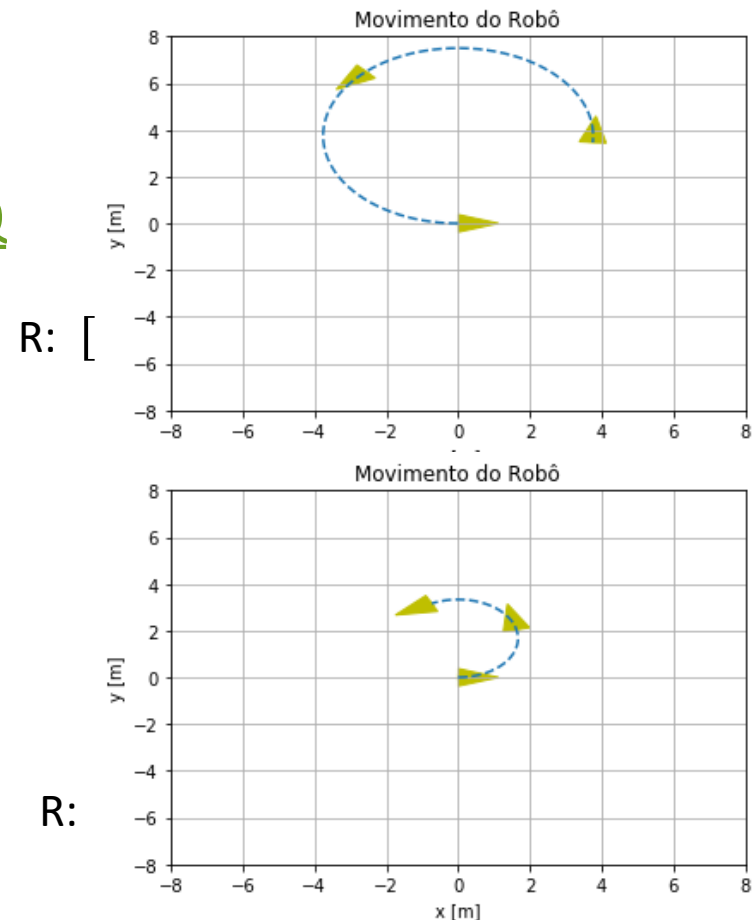
$$\begin{cases} \dot{x} = u \cos \phi \\ \dot{y} = u \sin \phi \\ \dot{\phi} = \omega \end{cases}$$





Exercício de Simulação 01

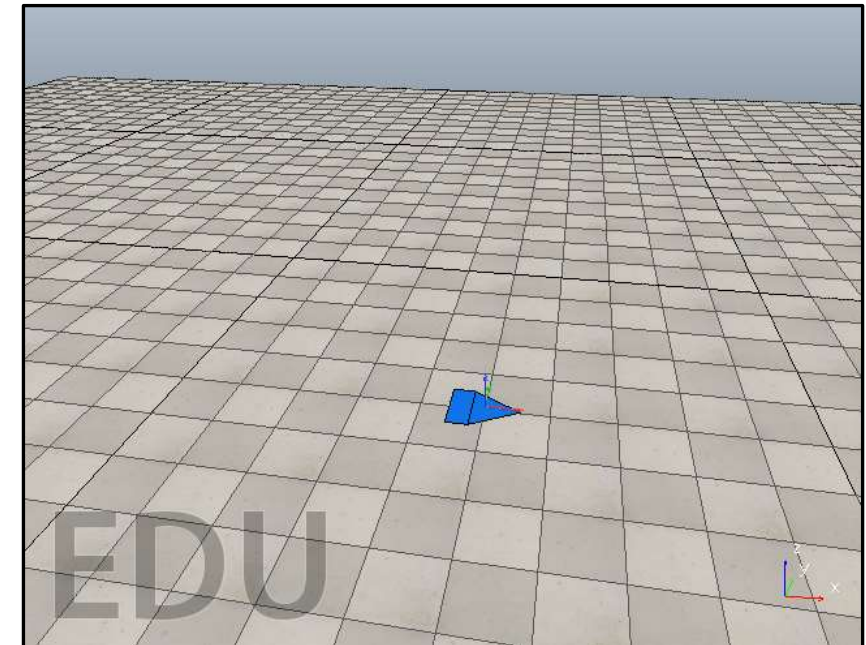
- Acesse o código deste exemplo no *Jupyter Notebook* disponível em:
 - <https://colab.research.google.com/drive/1fVVJQpLPFO5DZ0LfDT2f4Zc34JQDNWtT>
 - Siga as instruções;





Exercício de Simulação 02 – CoppeliaSim (V-REP)

- Acesse os arquivos da pasta aula03-ex02 contendo o modelo de uma simulação do Uniciclo usando V-REP e API Remota para Python;
- Rode o código e compreenda seu funcionamento;
- Modifique-o para frear o robô depois de 60 segundos de simulação;
- Compare o movimento executado com do Exercício de Simulação 02, cujos parâmetros de controle na simulação foram:
 - $v = -0.3 \text{ m/s}$
 - $\omega = -0.08 \text{ rad/s}$





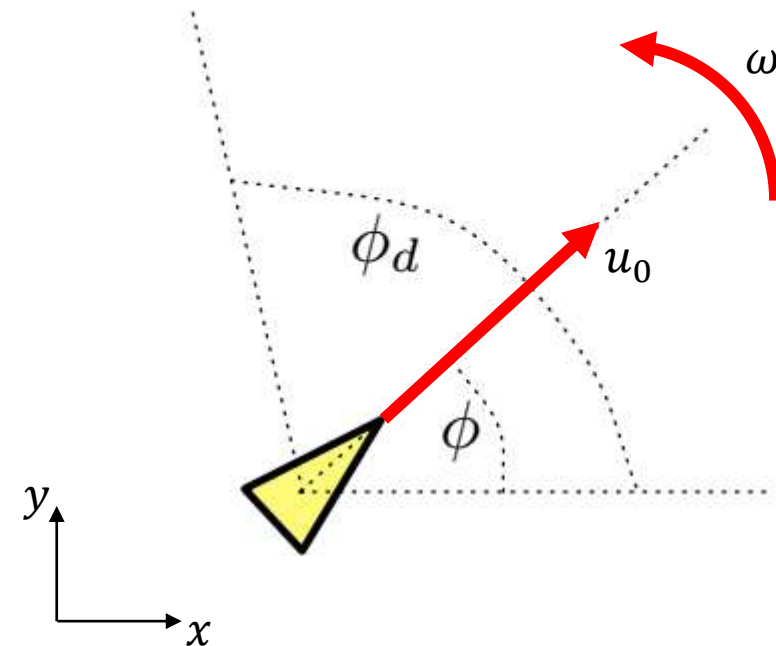
Controle de Orientação (*Heading*)



Controle de Orientação (Heading)

- Suponha o veículo navegando a uma **velocidade linear** constante $u = u_0$.
- Qual **sinal de controle** ω leva o robô de sua orientação atual (ϕ) para uma orientação desejada (ϕ_d)?

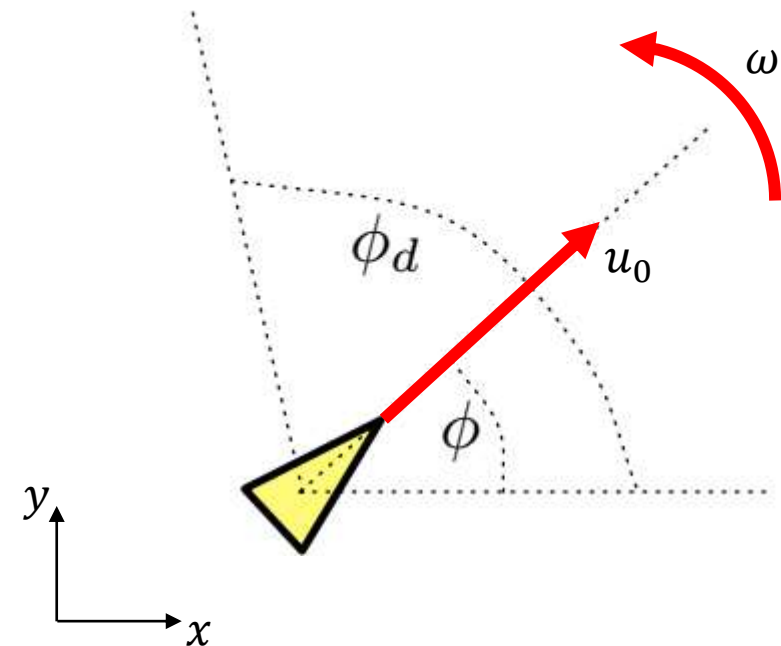
$$\begin{cases} \dot{x} = u_0 \cos\phi \\ \dot{y} = u_0 \sin\phi \\ \dot{\phi} = \omega \end{cases}$$





Controle de Orientação (*Heading*)

- Observe que neste problema existe:
 - Uma referência de ângulo desejada: ϕ_d ;
 - Um modelo linear: $\dot{\phi} = \omega$;
- Pode-se escrever: $e = \phi_d - \phi$;
- O que implica em: $\omega = k_p e + k_i \int e d\tau + k_d \dot{e}$
- Será que isso funciona?





Controle de Orientação (*Heading*)

- **Sim**, desde que os sinais que são **ângulos** sejam tratados;

- Suponha um **caso particular**:

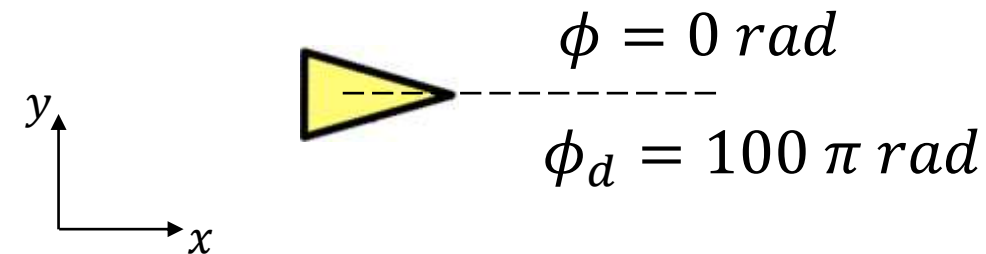
- $\phi = 0 \text{ rad}$;
- $\phi_d = 100 \pi \text{ rad}$;

- Seria correto usar no controlador um erro:

- $e = \phi_d - \phi = 100 \pi \text{ rad}$?

- Não é adequado, pois:

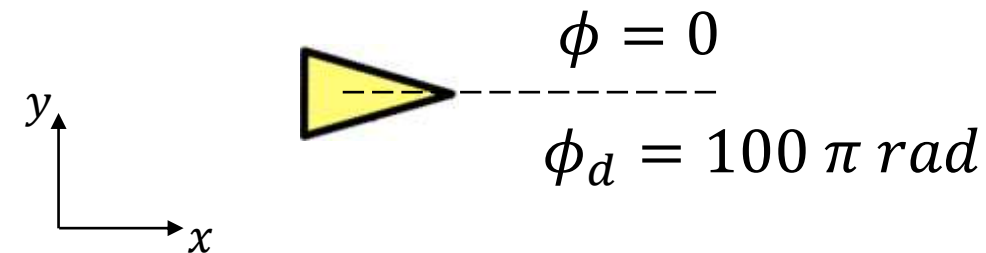
- $100 \pi \text{ rad} = 0 \text{ rad}$;
- Logo o sinal de erro deveria ser $e = 0 \text{ rad}$;





Controle de Orientação (*Heading*)

- Solução:
 - **Garantir** que todos os sinais de controle que sejam **dependentes de ângulos** pertençam ao domínio $[-\pi, +\pi]$;
 - No caso deste exemplo, garantir que: $e \in [-\pi, +\pi]$
 - Há um truque computacional elegante para este fim, realizado através da função matemática da **inversa da tangente** ou **arco tangente**;





Controle de Orientação (*Heading*)

- Dado um sinal angular qualquer α , sua tangente é:

$$\operatorname{tg} \alpha = \frac{\sin \alpha}{\cos \alpha}$$

- Aplicando o arco tangente, em ambos os lados dessa igualdade resta:

$$\alpha = \operatorname{tg}^{-1} \left(\frac{\sin \alpha}{\cos \alpha} \right)$$

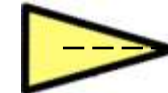
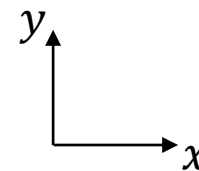
- Via *software*, pode-se usar comandos como:

Linguagem	Código Fonte
C/C++	$\alpha_c = \operatorname{atan2}(\sin(\alpha), \cos(\alpha))$
Python	$\alpha_c = \operatorname{np.arctan2}(\operatorname{np.sin}(\alpha), \operatorname{np.cos}(\alpha))$
Matlab/Octave	$\alpha_c = \operatorname{atan2}(\sin(\alpha), \cos(\alpha))$



Exercício de Simulação 03

- Acesse o código deste exemplo no *Jupyter Notebook* disponível em:
 - https://colab.research.google.com/drive/1zFMxQlqcfz9VA3Ve2sBh6_5UwzAkxtNF
 - Siga as instruções;



$\phi = 0 \text{ rad?}$

$\phi = 2\pi \text{ rad?}$

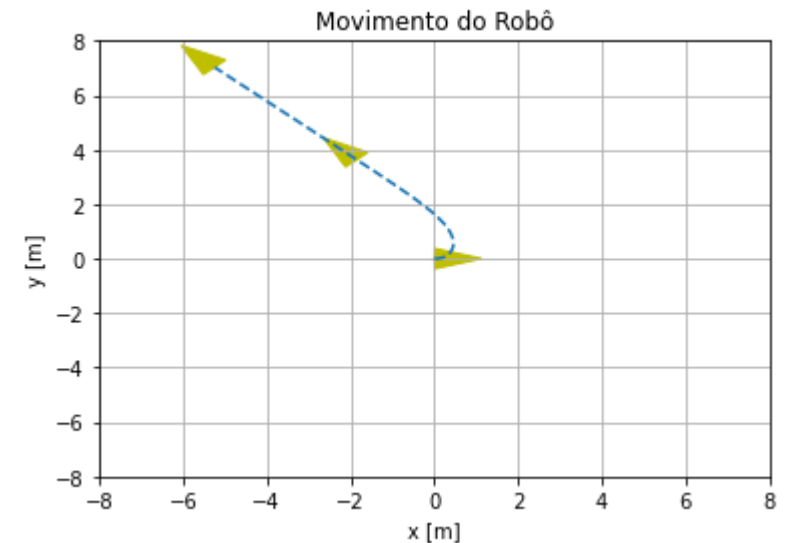
$\phi = 4\pi \text{ rad?}$

...



Exercício de Simulação 04

- Acesse o código deste exemplo no *Jupyter Notebook* disponível em:
 - https://colab.research.google.com/drive/1zFMxQIqcfz9VA3Ve2sBh6_5UwzAkxtNF
 - Siga as instruções;





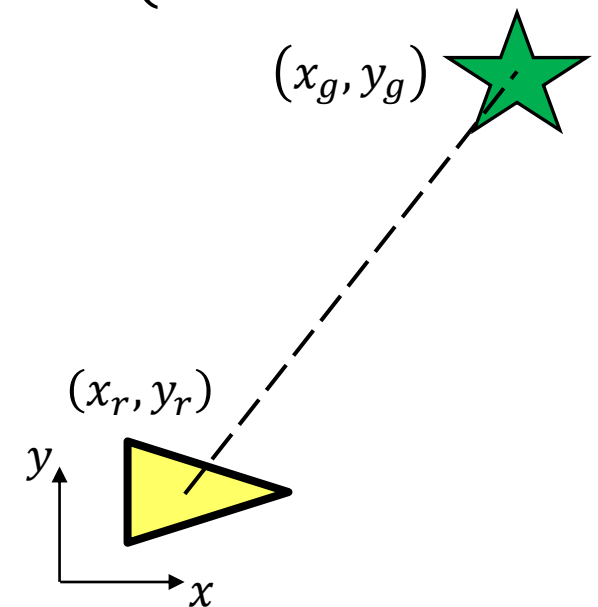
Controle de Posição (*Goal-to-Goal*)



Controle de Posição (*Goal-to-Goal*)

- Como gerar o comportamento de ir até um ponto?

$$\begin{cases} \dot{x} = u \cos \phi \\ \dot{y} = u \sin \phi \\ \dot{\phi} = \omega \end{cases}$$





Controle de Posição (*Goal-to-Goal*)

- Uma solução intuitiva consiste em:

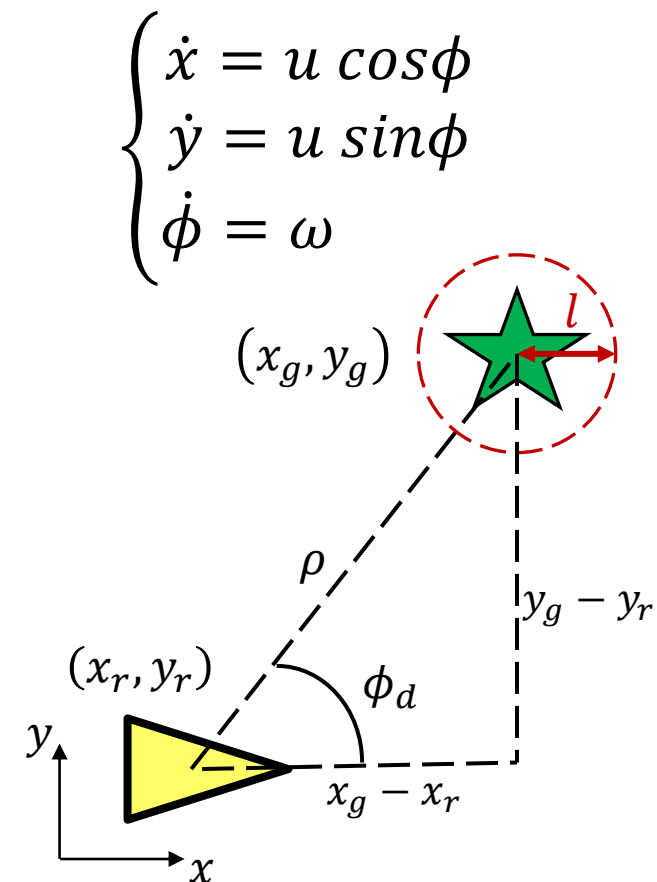
- Estipular uma velocidade linear de avanço $u = u_0$;
- A cada iteração, calcular uma nova orientação desejada ϕ_d como:

$$\phi_d = \text{tg}^{-1} \left(\frac{y_g - y_r}{x_g - x_r} \right)$$

- Usar ϕ_d para executar o controle de *heading*;
- Parar o robô assim que ele adentrar uma região de raio l , no entorno do destino;
- Como regra, pode-se utilizar :

$$\text{Calcular: } \rho = \sqrt{(x_g - x_r)^2 + (y_g - y_r)^2}$$

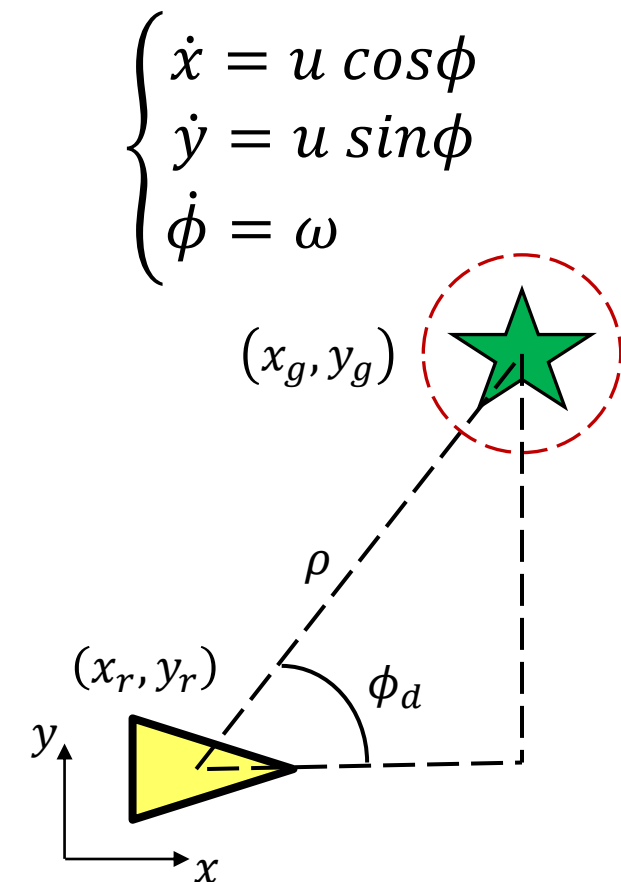
Testar: Se $\rho < l$, então: $u = 0$ e $\omega = 0$





Controle de Posição (*Goal-to-Goal*)

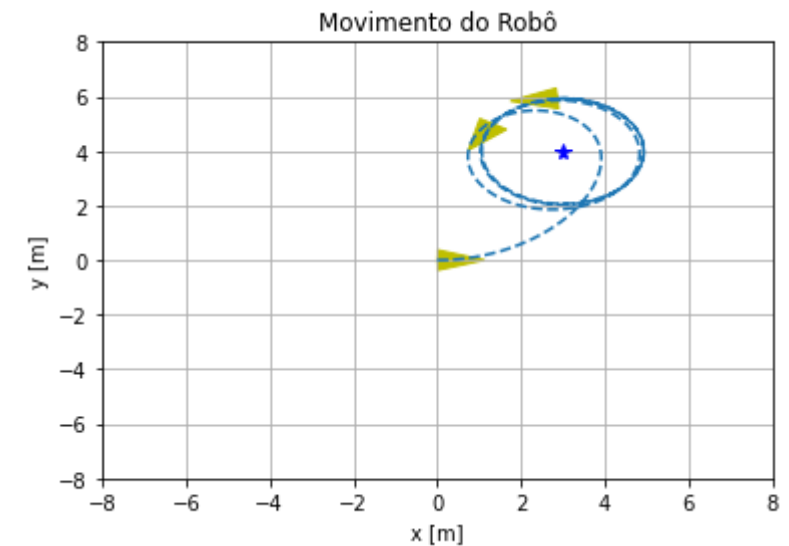
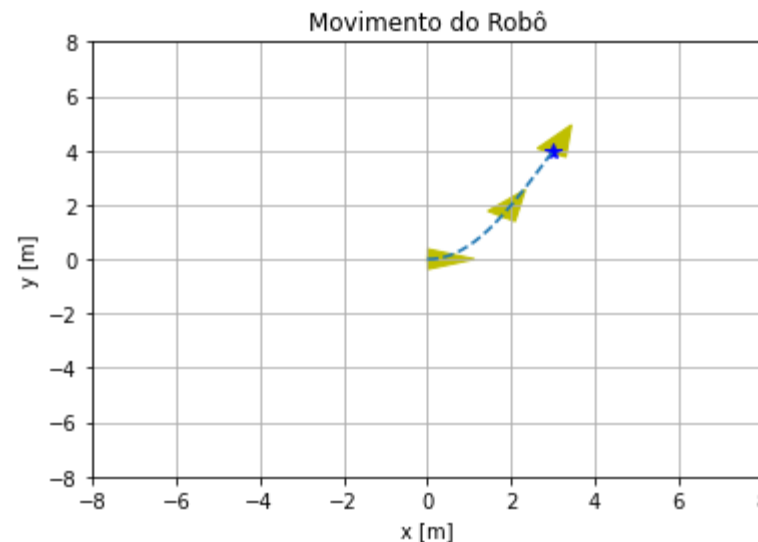
- Observações sobre este método:
 - Não há garantia da orientação final do robô;
 - Não há garantias de convergência se os parâmetros estiveram mal ajustados;





Exercício de Simulação 05

- Acesse o código deste exemplo no *Jupyter Notebook* disponível em:
 - <https://colab.research.google.com/drive/1HpXpySzeUxecNDSimdd8NQdC8Y4yX3mK>
 - Siga as instruções;





Exercício de Simulação 06 – CoppeliaSim (V-REP)

- Acesse os arquivos da pasta aula03-ex06 contendo o modelo de uma simulação do Uniciclo com controle *Goal-to-Goal* usando V-REP e API Remota para Python;
- Rode o código e compreenda seu funcionamento;
- Movimente o alvo na cena e perceba o comportamento do uniciclo

