UNIVERSIDADE FEDERAL DO AMAZONAS
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ELETRÔNICA E COMPUTAÇÃO
PROGRAMAÇÃO DE SISTEMAS DE TEMPO REAL
PROF. ANDRÉ CAVALCANTE
SEMESTRE 2024/2

LABORATÓRIO 2 - Simulação de Sistemas 1

Aluno: Matheus Carvalho Reges - 22152027

1. Descrição dos Objetivos

O objetivo principal desta simulação é modelar e analisar o movimento de um robô em um plano bidimensional, controlado por uma velocidade linear constante e uma velocidade angular variável ao longo do tempo. A simulação deve calcular e registrar a trajetória e orientação do robô ao longo de um intervalo de tempo especificado.

2. Descrição do Problema Proposto

O problema envolve a simulação do movimento de um robô que se desloca em um plano bidimensional. Esse robô possui uma velocidade linear constante e uma orientação que varia de acordo com um sinal de controle dependente do tempo. Nos primeiros 10 segundos, o sinal de controle mantém o robô em uma orientação de avanço constante, com velocidade angular igual a 1 rad/s. Após esse período, o sinal de controle muda, impondo uma orientação negativa que ajusta a trajetória do robô. A simulação deve computar e registrar a posição e orientação do robô a partir de uma posição inicial em (0,0) e uma orientação inicial de 0 radianos, avaliando como o controle afeta seu percurso e ângulo ao longo do tempo.

3. Introdução Teórica

A simulação aborda o movimento de um robô em um plano bidimensional, utilizando princípios de cinemática e controle. Em robótica, a cinemática trata de como as variáveis de movimento do robô (posição e orientação) mudam ao longo do tempo com base em suas velocidades linear e angular.
Neste caso, a velocidade linear do robô é constante, enquanto a velocidade angular depende de um sinal de controle, calculado pela função getControlSignal. O sinal de controle aplicado define a direção da rotação do robô em relação ao tempo: inicialmente, ele é positivo e constante (permitindo uma rotação suave em linha reta), mas após 10 segundos, o controle é invertido (passa a ser negativo) com uma taxa angular de -0,2 * π rad/s. Isso faz com que o robô mude a orientação de sua trajetória.
A função computeRobotEstate atualiza o estado do robô, calculando sua posição e orientação em cada incremento de tempo (deltaTime). A posição é atualizada de acordo com os componentes x e y, derivados da velocidade linear e da orientação atual (usando seno e cosseno do ângulo). A orientação é ajustada em cada passo com base no sinal de controle recebido, modelando o comportamento dinâmico do robô.

Essa abordagem permite observar como diferentes sinais de controle influenciam a trajetória e a orientação do robô, sendo aplicável a sistemas que requerem navegação precisa, como veículos autônomos e robôs móveis.

4. Descrição da Estrutura de Diretórios Utilizada

A estrutura de diretórios para o projeto é simples e organizada na raiz, contendo os arquivos de código-fonte e o Makefile para a compilação. Abaixo está uma descrição dos principais arquivos:

- estate: Existem dois arquivos com esse nome, um deles é o cabeçalho (estate.h) e o outro o arquivo de implementação (estate.c). Esses arquivos definem e implementam a função computeRobotEstate, que calcula a posição e orientação do robô em cada instante de tempo.
- model: Similar ao caso de estate, há dois arquivos com o nome model, um sendo o cabeçalho (model.h) e outro a implementação (model.c). Eles contêm a definição da estrutura RobotState e a função getControlSignal, responsável pelo controle da velocidade angular do robô.
- main: Arquivo principal (main.c) que executa a simulação. Ele cria o arquivo de saída (resultados.txt), executa a simulação ao longo do tempo, e registra os dados de posição e orientação do robô.
- makefile: Arquivo de configuração do Makefile, que contém as instruções para compilar o projeto. Ele especifica o compilador, as flags de compilação, e as dependências de cada arquivo.
- LEIA-ME: Documento que contém informações adicionais sobre o projeto, como instruções de compilação e execução.
- graficos: Diretório que contém os gráficos gerados a partir dos dados da simulação, assim como o código Octave responsável por criar esses gráficos.

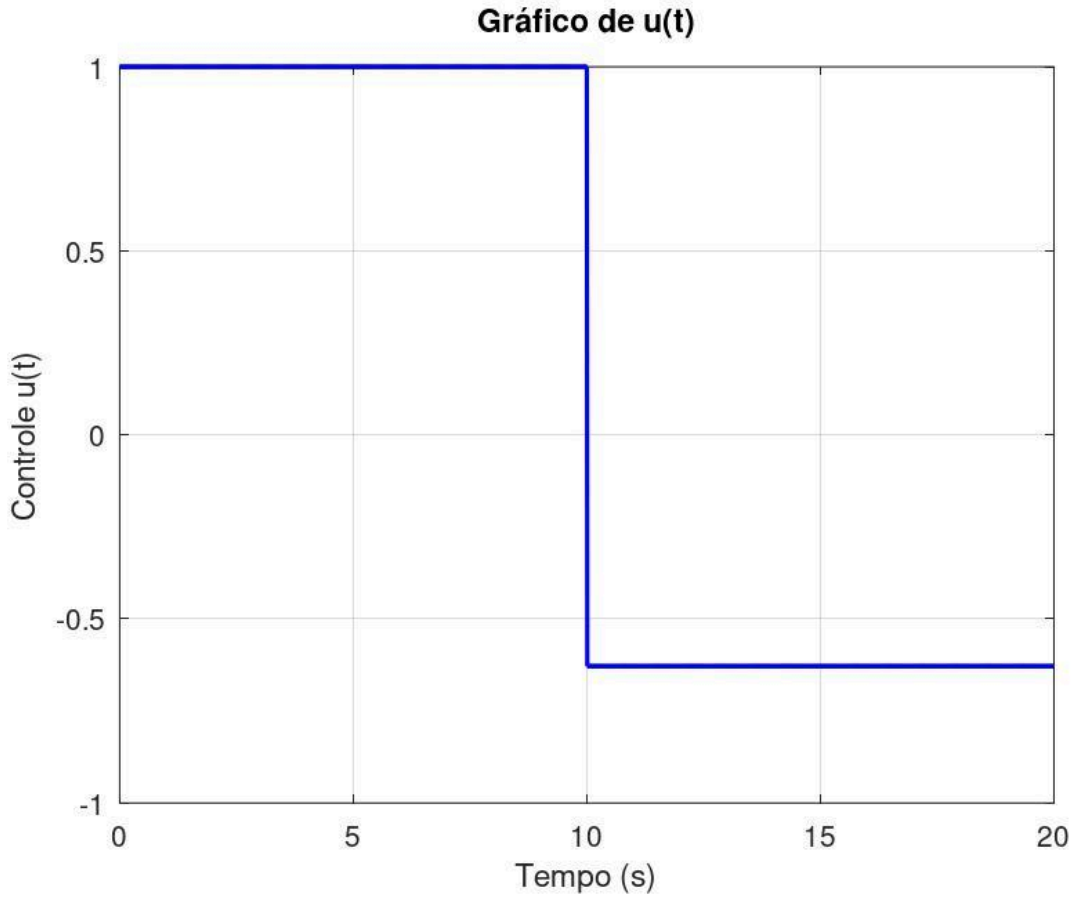
5. Descrição dos Arquivos Fontes

- estate.c: Implementa a função computeRobotEstate, que atualiza a posição e orientação do robô usando a velocidade angular recebida e o intervalo de tempo (deltaTime).
- estate.h: Define a função computeRobotEstate e inclui o cabeçalho model.h para usar o tipo RobotState.
- model.c: Implementa a função getControlSignal, que fornece o sinal de controle do robô com base no tempo decorrido. Esse sinal define a velocidade angular e controla a direção do movimento do robô.
- model.h: Define a estrutura RobotState com os atributos x, y e ang para representar a posição e orientação do robô. Declara também a função getControlSignal.
- main.c: Programa principal que cria um arquivo resultados.txt para armazenar os resultados da simulação. Configura o tempo total da simulação e, em um loop, chama getControlSignal e computeRobotEstate para atualizar o estado do robô em intervalos de tempo especificados.
- Makefile: Automatiza a compilação dos arquivos, especificando o compilador (gcc), as flags, e a criação do diretório build/ para os objetos compilados e o executável. Também contém uma regra para limpeza dos arquivos gerados.

- graficos: Diretório que contém os gráficos gerados a partir dos dados da simulação, assim como o código Octave responsável por criar esses gráficos. Este diretório inclui:
    - u(t).jpg: Gráfico de u(t), que mostra o controle em função do tempo.
    - y(t).jpg: Gráfico de y(t), que mostra as posições Xc, Yc, e a orientação θ ao longo do tempo.
    - Yc(t)xXc(t).jpg: Gráfico de Yc(t)×Xc(t), que representa a trajetória do robô no plano com indicação de orientação.
    - Octave_graficos.m: Código Octave que gera os gráficos. Este script lê os dados do arquivo resultados.txt e cria os gráficos de u(t), y(t), e Yc(t)×Xc(t)

6. Análise Gráfica e Comportamental de um Robô Móvel com Acionamento Diferencial

6.1 Gráfico u(t)



O gráfico apresentado é o de u(t), que representa o controle de entrada do robô em função do tempo.

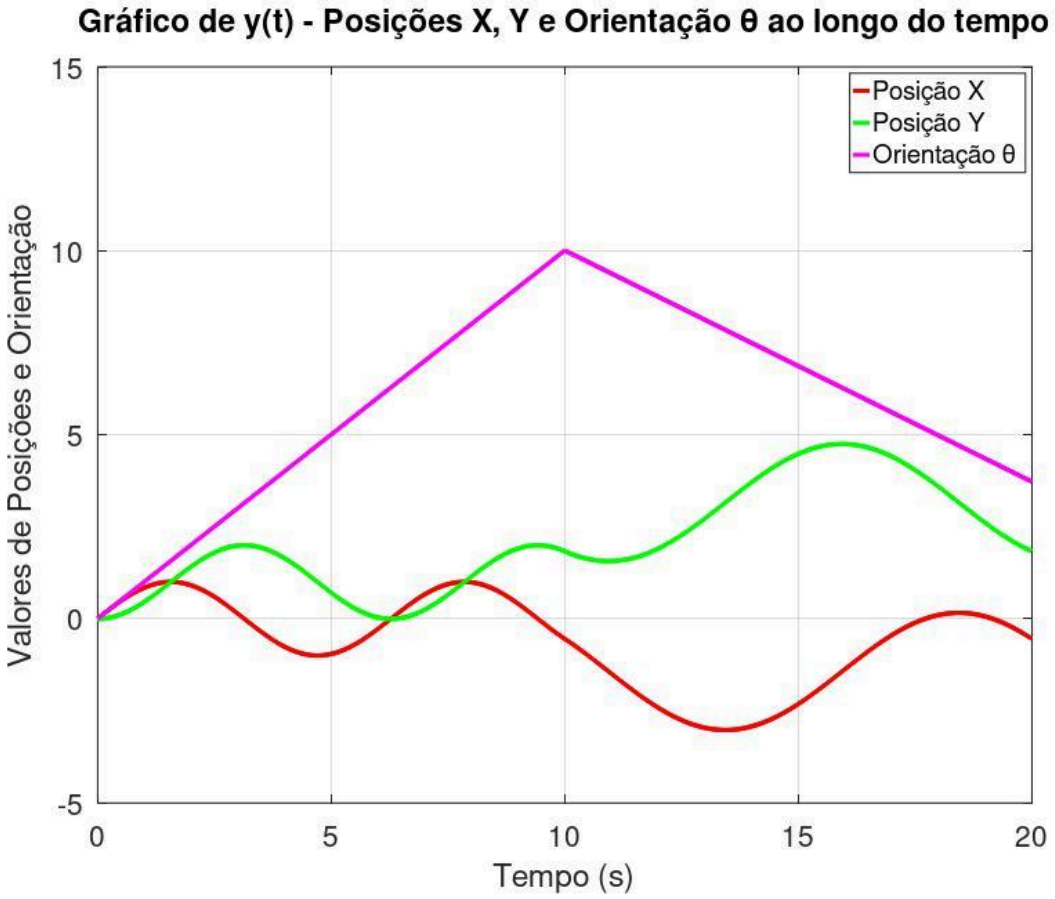
Análise do Gráfico de u(t):

- Eixo X (Tempo): O tempo está em segundos, variando de 0 a 20 segundos, que é o intervalo de simulação definido.
- Eixo Y (Controle u(t)): Representa o valor do controle aplicado ao sistema, com três fases distintas conforme definido no problema:

- Para t<0: u(t)=0 (não aparece no gráfico, pois o intervalo de simulação é de 0 a 20 segundos).
- Para 0≤t<10: u(t)=1. Neste intervalo, o robô recebe um controle constante positivo.
- Para t≥10: u(t)=−1. Após os 10 segundos, o controle muda para um valor negativo constante, alterando a direção da entrada aplicada ao robô.

o gráfico de u(t) está conforme esperado de acordo com as especificações do problema pois essa variação condiz com as instruções fornecidas para o controle u(t), onde inicialmente o robô deve se mover em uma direção até 10 segundos, e, em seguida, o controle inverte o sinal, sugerindo que o robô mudará sua direção a partir de então.
Este gráfico permite entender como o controle varia ao longo do tempo e a resposta do robô nas outras variáveis devido a essa entrada.

6.2 Y(t)



Este gráfico mostra as variáveis de saída y(t), que incluem as posições Y e X e a orientação θ ao longo do tempo t. A legenda específica cada curva:

- **Posição X** (em vermelho): representa a variação da posição Xc(t) do robô ao longo do tempo, exibindo um comportamento oscilatório.
- **Posição Y** (em verde): indica a variação da posição Yc(t), também oscilando, mas com uma amplitude maior que a de Xc(t).
- **Orientação θ** (em lilás): demonstra a variação do ângulo θ(t), com um comportamento linear nos primeiros 10 segundos, seguida de uma inclinação

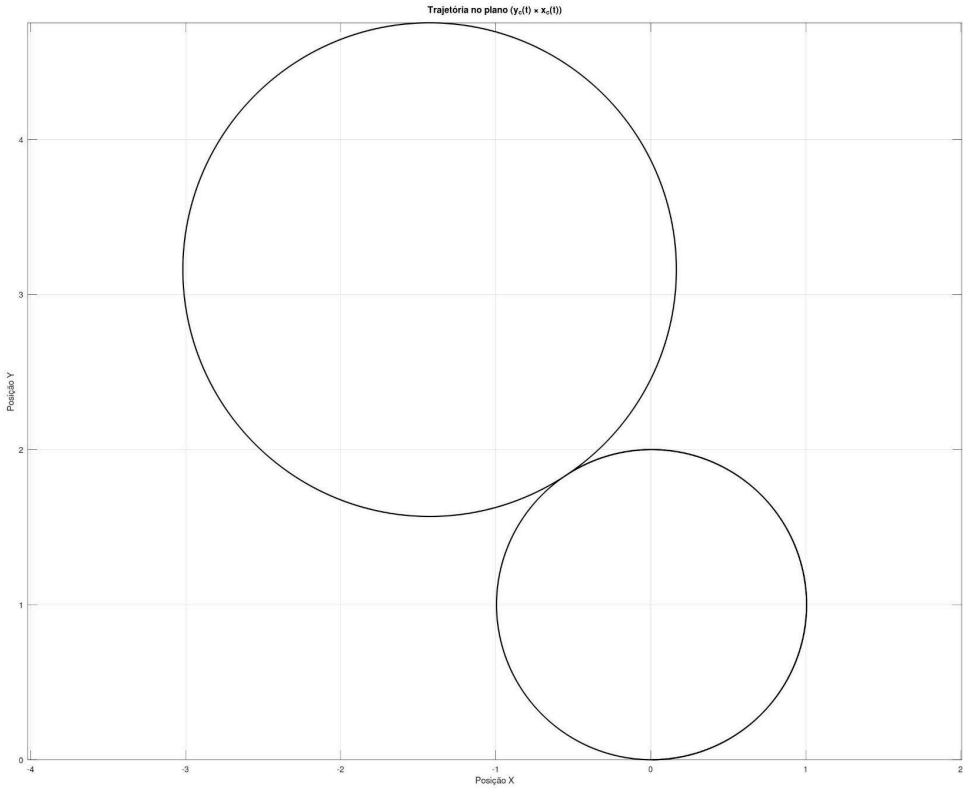
negativa para o restante do intervalo, devido à mudança no sinal da velocidade angular ω em u(t).

Esse comportamento está **conforme o esperado**, pois o modelo descreve que:

- Para $0 \leq t < 10$, o robô se move com velocidade linear $v=1$ e velocidade angular $\omega=0.2\pi$, o que faz com que a orientação aumente linearmente com o tempo.
- Para $t \geq 10$, a velocidade angular inverte para $\omega=-0.2\pi$, resultando na mudança de direção observada em θ(t).

A análise desse gráfico permite observar como a posição e a orientação do robô mudam com a entrada de controle, ilustrando o comportamento esperado do sistema para o intervalo $t \in [0,20]$s.

6.3 Trajetória no plano (Yc(t) × Xc(t)



Este gráfico exibe a trajetória do robô no plano cartesiano, com **Posição X** no eixo horizontal e **Posição Y** no eixo vertical. Ele representa como o robô se move ao longo do tempo com a entrada definida:

- A trajetória forma duas curvas circulares conectadas, uma maior seguida por uma menor, o que reflete a ação de controle aplicada ao robô. A primeira curva é gerada

pela entrada com $\omega>0$ (trajetória no sentido anti-horário), e a segunda curva resulta da inversão de ω para negativo (trajetória no sentido horário).

Este comportamento também está **conforme o esperado**, pois, de acordo com a entrada $u(t)$, a mudança de ω após $t=10$ leva o robô a realizar uma curva em sentido oposto, formando a segunda seção da trajetória circular.