MC202GH - Estrutura de Dados - Turmas G e H
Laboratório 4 - *Recursão e Backtracking*

Docente: Marcelo da Silva Reis

Monitor PED: Matheus Abrantes Cerqueira

Monitores PAD: Andreas Cisi Ramos
Wallace Gustavo Santos Lima

16 de outubro de 2022

Data de entrega: 21/10/2022

Entrega no codePost[1]

Informações gerais

Neste laboratório trabalharemos com os conceitos de recursão e backtracking. Para isso, como de praxe serão fornecidos arquivos com protótipos a serem modificados e enviados na plataforma de avaliação. O presente laboratório tem como motivação a Robótica, tema relacionado à Engenharia de Controle e Automação.

Observações importantes:

1. Neste laboratório será permitido o uso apenas das bibliotecas `stdio.h`, `stdlib.h`, `string.h` e `math.h`, além do arquivo de interface do laboratório, que é o `robot.h`.
2. O arquivo `robot.h` que contém os protótipos de tipos e operadores do nosso TAD, não deve ser modificado em hipótese alguma. Alterações devem ser feitas apenas nos arquivos de implementação e de cliente (**robot.c** e **questao_x.c**).
3. Para cada questão, entre na pasta correspondente à mesma e compile o código utilizando o arquivo `Makefile` contindo nela.
4. Estejam cientes da necessidade de conversão entre graus e radianos, uma vez que as entradas serão feitas em graus mas as funções ***cos*** e ***sin*** da math.h esperam valores em radianos.

[1] https://codepost.io/signup/join?code=ZW239C3IID

Questão 1 (3 pontos) - Robô planar de um eixo

Em um robô cinemática se refere a obter uma relação entre suas variáveis de junta e a posição final de seu efetuador terminal ("braço" do robô). Existem duas formas de se obter a cinemática de um robô, que são de maneira direta e indireta.

A cinemática direta é uma forma de obter a posição efetuador terminal, inicialmente desconhecida, a partir de posições de junta conhecidas. A cinemática inversa é uma maneira de obter variáveis de junta desconhecidas a partir do conhecimento da posição onde o efetuador terminal está.

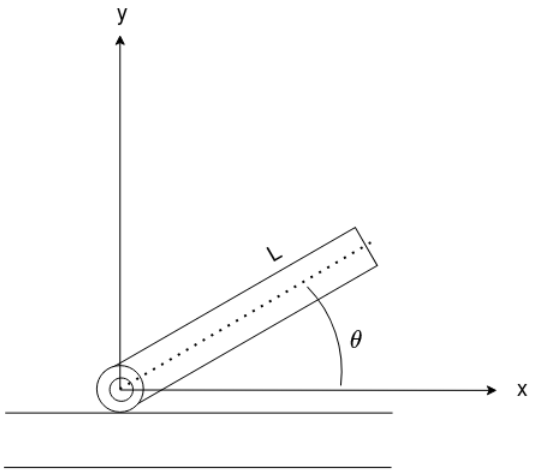

Figura 1: Robô de um eixo.

Suponha um robô planar, o qual é formado por elos (anéis de ligação) e juntas móveis, assim como mostra a figura 1 para um robô com um elo e uma junta que forma um ângulo θ. Seja o efetuador terminal a ponta do elo a cinemática direta do robô pode ser dada por:

$$x = l_1 cos(\theta) \tag{1}$$

e

$$y = l_1 sen(\theta). \tag{2}$$

O objetivo dessa questão é realizar o cálculo da cinemática inversa do robô dado sua configuração atual (Veja TAD de **robo** no arquivo robot.h) utilizando-se de recursão. O uso da recursão pode parecer desnecessário uma vez que as equações acima podem ser facilmente resolvidas, porém essa questão irá ser servir como base para a próxima questão.

Para isso desenvolva a função ***cinematica_inversa_escalar*** contida no arquivo robot.h, deve-se também implementar a função de cinemática direta ***cinematica_direta***. **Observação:** use como critério de parada distância de 0.01 entre o ponto (x,y) desejado e o calculado pela aproximação numérica na função de cinemática inversa.

A questão (arquivo *questao_1.c* em sí deve esperar a entrada das configurações do robô na primeira linha (tamanho do eixo e a pose inicial data por θ **em graus**) e em

seguida deve-se calcular a cinemática inversa para diversas entradas na forma X Y até receber um par 00:

```
10 30
7.07 7.07
5.74 8.19
-5  8,66
0 0
```

e a saída será algo como (observe que como (possiveis alterações são válidas dado a tolerância de parada):

```
ROBO CRIADO: (30.00)
PROXIMO ESTADO (45.00)
PROXIMO ESTADO (55.00)
PROXIMO ESTADO (120.00)
```

Questão 2 (3 pontos) - Robô planar de dois eixos

Agora iremos trabalhar com um exemplo um pouco mais complexo, que é o robô de dois eixos, assim como mostra a figura 2, o qual possui dois elos e dois graus de liberdade θ_1, θ_2 e que possui cinemática direta e inversa mais complexas que o exemplo anterior, sendo a cinemática direta definida por:

$$x = l_1 cos(\theta_1) + l_2 cos(\theta_2 + \theta 1) \tag{3}$$

e

$$y = l_1 sen(\theta_1) + l_2 sen(\theta_2 + \theta 1). \tag{4}$$

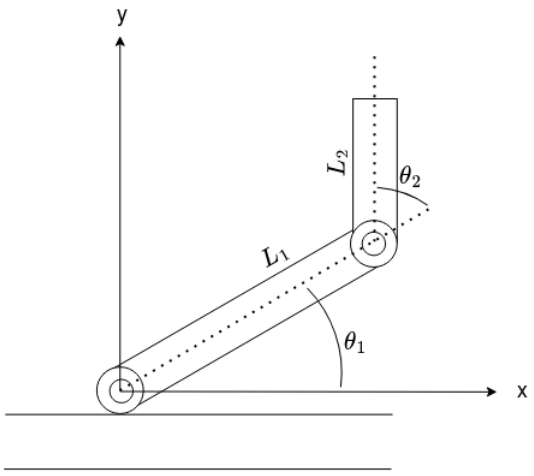

Figura 2: Robô de grau de liberdade igual a 2.

Da mesma forma o exemplo anterior, utilize a TAD **robo** que é flexível quanto ao número de graus de liberdade, porém agora desenvolva a função ***cinematica_inversa_vetorial*** para essa questão, utilizando tolerância de parada de 0.2, ou seja, o destino final deve estar a uma distância de 0.2 do desejado.

O exemplo a seguir contém a configuração de um robô com $L1 = 10$, $L2 = 5$, $\theta_1 = 30$ e $\theta_2 = 0$, além de 4 outras posições diferentes (pares θ_1, θ_2):

```
10 5 30 0
10.61 10.61
11.40 9.57
5 10
-5 10
0 0
```

e a saída será algo como (de novo, devido à aproximação alguns ângulos poderão estar diferentes):

```
ROBO CRIADO: (30.00, 0.00)
PROXIMO ESTADO (45.00, 0.00)
PROXIMO ESTADO (45.00, -15.00)
PROXIMO ESTADO (90.00, -90,00)
PROXIMO ESTADO (90.00, 90,00)
```

Dica: Poderá ser utilizada o cálculo por meio de descida do gradiente, onde cada iteração será a seguinte:

$$\theta_i{}^{t+1} = \theta_i{}^{t} - LR * grad(\theta_i{}^{t}) \tag{5}$$

Onde $\theta_i{}^{t+1}$ representa a junta i na iteração $t+1$. Se assim desejar utilize a função já implementada ***compute_gradient2d***, a qual recebe a posição de destino, o robô atual (2 graus) e o gradiente a ser computado (**gradiente esse na forma de um vetor 2d, sendo cada valor para uma junta**).

Questão 3 (4 pontos) - Labirinto

Nessa questão iremos utilizar do *backtracking* para resolver um problema de um robô perdido em um labirinto, sendo que serão fornecidos os tamanhos do labirinto, sua organização, a posição inicial do robô e o seu destino. Por exemplo, seja o labirinto da figura 3 espera-se que o programa comporte-se da seguinte maneira, sendo que o robô pode se mover em 4 direções:

```
4 3
R 0 0
1 1 0
1 1 0
1 G 0
```

e a saída:

```
PATH IS (0,0) (0,1) (0,2) (1,2) (2,2) (3,2) (3,1)
```

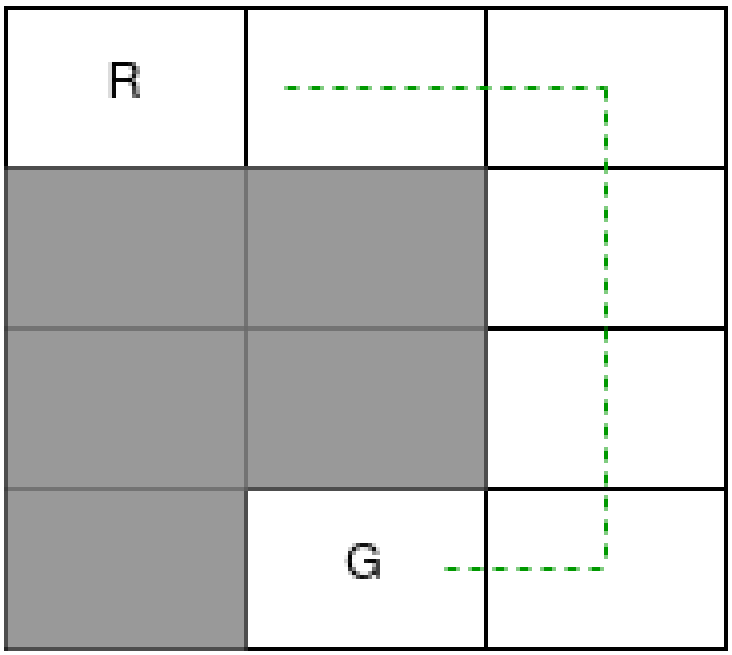

Figura 3: Robô móvel em um labirinto.