Robótica Espacial
Aula prática nº 2
Cinemática direta

Vitor Santos

Universidade de Aveiro

20 fev 2025

Sumário

1. Representação e simulação de robôs básicos
2. Representação e simulação de um robô RRR antropomórfico
3. Criação e simulação de um robô SCARA

Exercício 1 - Criação e representação de um robô RR planar

Criar um robot RR planar

- Definir os comprimentos dos elos L1 e L2 (e.g. 2 e 1)
- Criar os dois elos com juntas rotacionais com o comando `Link()` e os parâmetros de Denavit-Hartenberg (DH) dados na seguinte tabela:

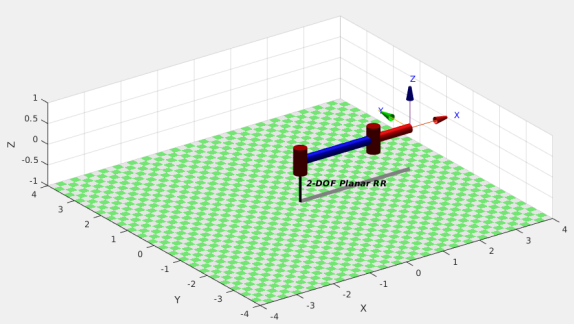
elo i	θ_i	a_i/l_i	d_i	α_i
1	θ_1	L_1	0	0
2	θ_2	L_2	0	0

  - Como as juntas são rotacionais, o valor da junta (`'theta'`) é variável e logo não se define, definindo-se apenas os outros 3 parâmetros:
  - `L(1) = Link('a', L1, 'd', 0, 'alpha', 0);`
  - `L(2) = Link('a', L2, 'd', 0, 'alpha', 0);`
- Com o comando `SerialLink` criar a cadeia cinemática do robô RR planar dando-lhe um nome à escolha, e.g. `'2-DOF Planar RR'` :
  - `RR_robot = SerialLink(L, 'name', '2-DOF Planar RR');`

Exercício 1

Representação inicial do robô RR planar

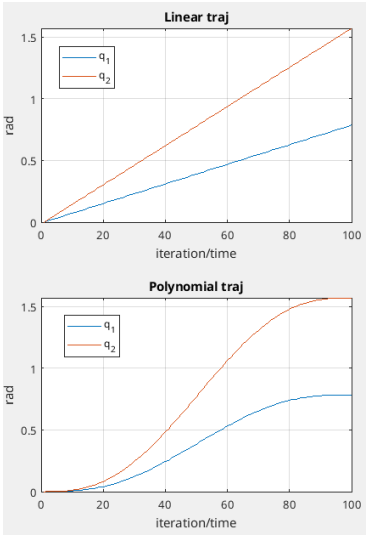
- Representar o robô na posição [0 0] usando o seu método `plot` e com parâmetros adequados para a visualização como, por exemplo:
  - `RR_robot.plot([0 0], 'workspace', [-4 4 -4 4 -1 1],'delay',0)`



Nota: o parâmetro " `'delay',0` " usado acima dá a indicação de que quando este modelo for animado não haverá atraso entre *frames* sucessivas da animação.

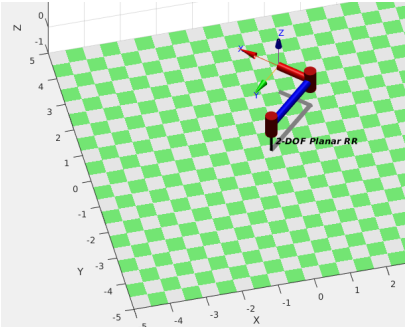
Exercício 2 - Definição de um caminho nas juntas

- Criar variáveis para conter os valores iniciais e finais das duas juntas para um determinado movimento em juntas.
  - `qi = [0, 0];`
  - `qf = [pi/4, pi/2];`
- Criar vários valores intermédios (e.g. 100) entre os pontos inicial e final, e colocá-os na variável `qtraj` que será uma matriz de 100×2.
- Criar os pontos de duas formas diferentes:
  1. Usando o comando `jtraj()` (polinomial)
  2. Usando o comando `linspace()` (linear)
- Os resultados são diferentes como se deve verificar fazendo o `plot` da evolução das juntas, como ilustrado ao lado:



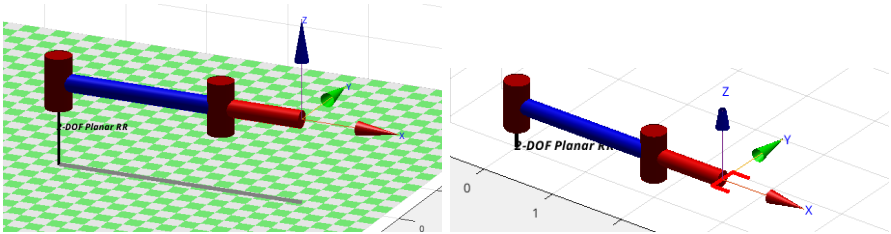
Exercício 3 - Animação do movimento da trajetória

- Usando a função `RR_robot.animate()` com o parâmetro adequado, animar o movimento do robô até ao fim da trajetória
- O programa deve executar dentro de um ciclo `for`
- Dentro do ciclo deve haver um comando de `drawnow` ou de `pause()` para atualizar o gráfico antes de passar à frame seguinte



Exercício 4 - Incorporação de uma garra no robô

- Para obter um sistema mais realista, incluir uma garra simples
- Pode ser uma simples linha como a ilustrada de seguida à direita:



- Esta linha pode ser definida com 4 pontos apenas
- Deve ser definida no referencial base do robô e depois "movida" para a extremidade
- As dimensões podem ser:
  - Comprimento dos dedos: 0.25
  - Afastamento entre os dedos: 0.4

Exercício 4

Metodologia

- Criar uma matriz (com nome `GripperDef`) com os 4 pontos em que o primeiro e o último são as pontas dos dedos (as coordenadas z de todos os pontos são 0)
- Obter a transformação da ponta do robô na posição inicial
  - `T = RR_robot.fkine(qi);`
  - Esta matriz T é um objeto especial (SE3) que inclui muitas propriedades e informação para além da própria matriz homogénea.
  - Uma das propriedades é que pode ser multiplicada a pontos que não estejam no formato homogéneo sem erro!
- Calcular os pontos reais de localização da garra no robô: `Gripper=T*GripperDef`
- Desenhar a linha a vermelho com `plotp()` e guardar o "handle": `gr=plotp(Gripper,'r');`
- Dentro do ciclo `for` animar também a posição da garra:
  - calculando as suas novas posições sucessivas e ...
  - ... atualizando os campos `XData`, `YData` e `ZData` do "handle".

Forma alternativa de especificar elos e juntas

- Nos exercícios anteriores usou-se o comando `Link()` para criar os elos/juntas.
- Existem comandos alternativos como casos particulares que são mais explícitos, embora os parâmetros sejam similares:
  - `Revolute()`;
  - `Prismatic()`;
- Portanto, os comandos dos exercícios anteriores que criam cadeias cinemáticas (robôs) poderiam ter sido usados com `Revolute()` em vez de `Link()`.
- Estes comandos alternativos têm outras opções que podem ser úteis como, por exemplo:
  - `qLim` para as os limites da variável de junta ou
  - `offset` para indicar o valor inicial de uma junta quando o robô está no *zero hardware*.

- **N.B.** É recomendado consultar o manual dos comandos e estruturas de dados (*help*) para verificar parâmetros e explicações adicionais sobre o seu funcionamento.

Exercício 5 - Criação de um robô RRR antropomórfico

- Definir os comprimentos dos elos L_A, L_B e L_C (e.g. 2,2,1)
- Criar os três elos com juntas rotacionais com o comando `Link()` (ou `Revolute()`) e os parâmetros de Denavit-Hartenberg (DH) dados na seguinte tabela:

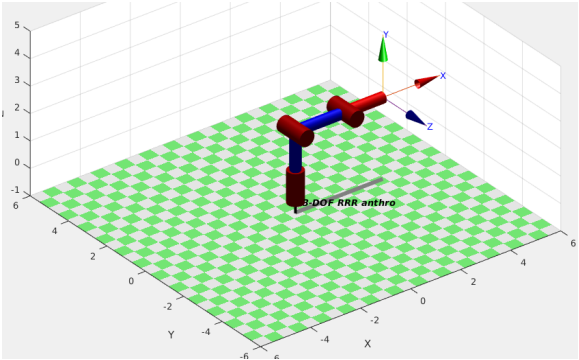
elo i	θ_i	a_i/l_i	d_i	α_i
1	θ_1	0	L_A	$\frac{\pi}{2}$
2	θ_2	L_B	0	0
3	θ_3	L_C	0	0

  - Como as juntas são rotacionais, o valor da junta (`'theta'`) é variável e logo não se define na criação dos elos, definindo-se apenas os outros 3 parâmetros:
    - `L(1) = Link('a', 0, 'd', LA, 'alpha', pi/2);`,
    - ou a alternativa com a função de dedicada a juntas rotacionais:
    - `L(1) = Revolute('a', 0, 'd', LA, 'alpha', pi/2);`,
    - etc...
- Com o comando `SerialLink` criar a cadeia cinemática do robô dando-lhe um nome:
  - `RRR_anthro = SerialLink(L, 'name', '3-DOF Anthro RRR');`

Exercício 5

Representação inicial do robô RRR antropomórfico

- Representar o robô na posição [0 0 0] usando o seu método `plot` e com parâmetros adequados para a visualização como, por exemplo:
  - `RRR_anthro.plot([0 0 0], 'workspace', [-6 6 -6 6 -1 5], 'delay', 0);`



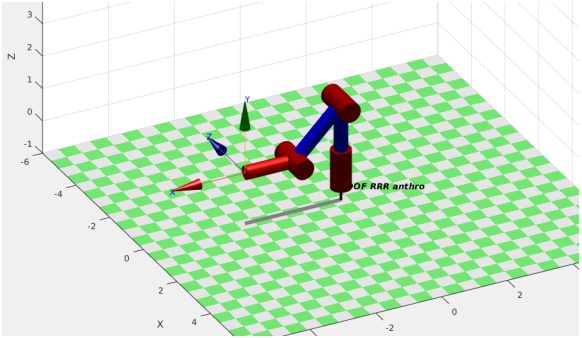
Nota: o parâmetro "`'delay',0`" usado acima dá a indicação de que quando este modelo for animado não haverá atraso entre *frames* sucessivas da animação.

Exercício 6 - Definição de um caminho nas juntas

- Criar variáveis para conter os valores das juntas em três instantes: inicial, intermédio e final:
  - `qi=[0 0 0];`
  - `qA=[pi/2 pi/4 -pi/4];`
  - `qf=[-pi/2 -pi/4 pi/4];`
- Criar vários valores intermédios (e.g. 100) entre os pontos inicial e intermédio e entre o intermédio e o final, e colocá-os na variável `qtraj` que será uma matriz de 200×3.
- Usar o comando `jtraj()` por duas vezes e combinar os resultados numa única matriz.
- Fazer também o percurso de regresso partindo de `qf` indo para `qA` e terminar em `qi`
- não é preciso voltar a usar o comando `jtraj` de novo: basta concatenar à matriz `qtraj` uma sua cópia com a ordem das linhas revertida:
  - Isso pode-se fazer com o comando `flipud()`

Exercício 7 - Animação do movimento da trajetória

- Usando a função `RRR_anthro.animate()` com o parâmetro adequado, animar o movimento do robô ao longo da trajetória
- O programa deve executar dentro de um ciclo `for`
- Dentro do ciclo deve haver um comando de `drawnow` ou de `pause()` para atualizar o gráfico antes de passar à frame seguinte



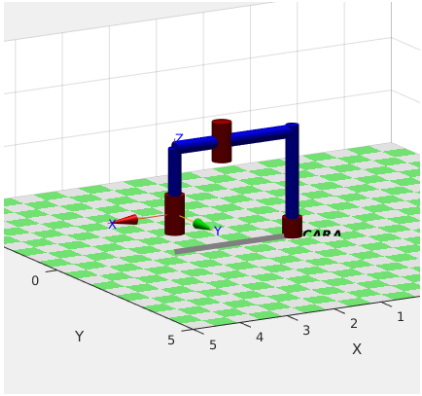
Exercício 8 - Criação de um robô SCARA

1. Um robô SCARA tem 4 DOF e inclui uma junta prismática.
2. A sua tabela de DH pode ser algo como:

elo i	θ_i	a_i/l_i	d_i	α_i
1	θ_1	L_A	L_B	0
2	θ_2	L_C	0	0
3	0	0	$d_3 - L_0$	0
4	θ_4	0	0	0

Os valores sugeridos das medidas são:

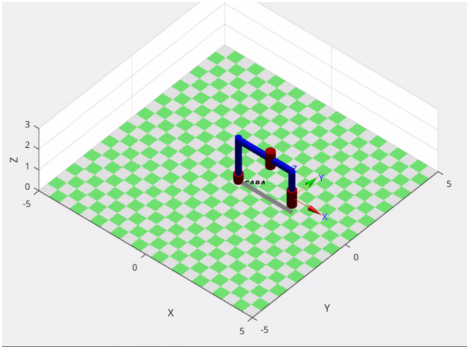
- $L_A = 2$;
- $L_B = 1.5$;
- $L_C = 1$;
- $L_0 = -1.5$; "offset"inicial



SCARA na sua posição inicial com as juntas nos seus valores *zero hardware*

Exercício 9 - Um processo de montagem

- Criar trajetórias que emulem um processo de montagem
- Os passos principais são:
  1. O robô começa da sua posição zero
  2. Eleva a sua junta prismática 1 unidade
  3. Coloca as suas juntas 1 e 2 em 60 e -45 graus respetivamente
  4. Baixa a junta prismática até ao mínimo
  5. Volta a subir a junta prismática em 1 unidade
  6. Move-se até à posição inicial



External player

Alguns passos podem eventualmente ser feitos em simultâneo para abreviar o tempo de ciclo, mas cada problema pode ter a sua especificidade e sem sempre se recomende.