Novas Tecnologias de Manufatura

Cinemática Direta em Manipuladores Robóticos
Robotics Toolbox for MATLAB

Engenharia Aeronáutica
2015/02

ENGENHARIA AERONÁUTICA
USP - São Carlos

- ⇨ Álgebra linear no *MATLAB*
- ⇨ *Robotics toolbox*
  - ⇨ Caracterização dos elos através de D-H
  - ⇨ Cinemática direta
  - ⇨ Geração de trajetória
  - ⇨ Animação gráfica

Agenda

Toolbox de livre distribuição e adaptação (LGPL) – versão 9.6 (Julho/2012);
Criado por *Peter Corke (Queensland University of Technology - Austrália)*
Permite a manipulação de modelos de manipuladores existentes:
- **PUMA560**
- **STANFORD**
- **(...)**

Permite a criação de modelos próprios com as opções:
- Matrizes de transformação homogênea;
- Planejamento de trajetórias;
- Cinemática direta;
- Cinemática inversa;
- Animação gráfica de rotação e translação;
- Matrizes *Jacobianas* algébrica e geométrica;
- Modelagem dinâmica direta e inversa;

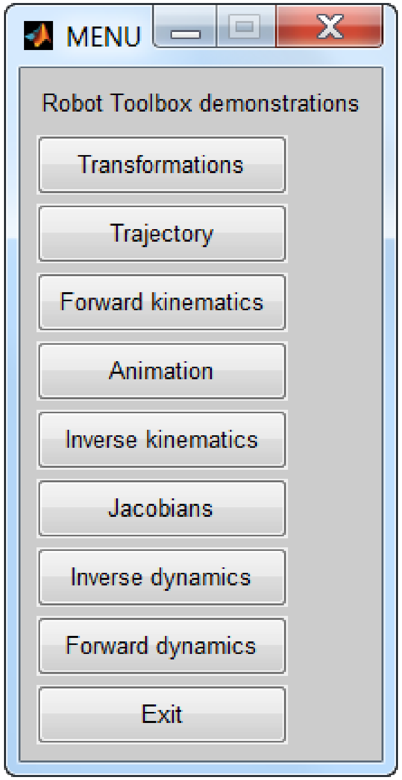
Observação para a instalação do *toolbox*:

- Descompactar o arquivo em: C:\Program Files\MATLAB**R2011b**\toolbox

- Será extraída uma pasta chamada '*rvctools*' – (nas versões anteriores '*robot*')

-Abrir o Matlab, clicar em **File**, **Set Path**: **Add Folder** (direcionar para a pasta '*rcvtools*'), **Add with Subfolders** (direcione para a pasta '*rcvtools*'), clicar em **Save**, **Close**.

- Executar exemplos: >> *rtdemo*



Robotics Toolbox

A partir da tabela com os parâmetros de D-H, e considerando:

a_i: a distância entre os eixos z_{i-1} e z_i medida sobre o eixo x_i.
α_i: o ângulo entre os eixos z_{i-1} e z_i medida sobre o eixo x_i. (EM RADIANOS)
d_i: a distância entre a origem do sistema de referência i - 1 ao eixo x_i, medida sobre o eixo z_{i-1}.
θ_i: o ângulo entre os eixos x_{i-1} e x_i medidos sobre o eixo z_{i-1}. (EM RADIANOS)

Criando LINKS (ELOS):

>> linkXY = link ([α_i a_i θ_i d_i σ_i])

```
>> link01 = link ([alpha01 a01 theta01 d01 sigma])
```

→ Onde 'sigma' (opcional) é o tipo de junta: 0-rotacional (default)/1-prismática

→ O OFFSET é determinado para a aproximação do modelo real e comparação com o volume de trabalho, para evitar pontos de singularidade.

Criando o robô:

```
>> r = robot ({link01 link02 ...}); %*robot = SerialLink
```

*Na versão 9.6, a classe robot é denominada como SerialLink. Para compatibilidade de funções com versões anteriores ao MATLAB R2011B, está sendo usada a versão 8 (2009) do Toolbox.

Method	Operations	Returns
`link.alpha`	r+a	link twist angle
`link.A`	r+a	link length
`link.theta`	r+a	link rotation angle
`link.D`	r+a	link offset distance
`link.sigma`	r+a	joint type; 0 for revolute, non-zero for prismatic
`link.RP`	r	joint type; 'R' or 'P'
`link.mdh`	r+a	DH convention: 0 if standard, 1 if modified
`link.I`	r	3×3 symmetric inertia matrix
`link.I`	a	assigned from a 3×3 matrix or a 6-element vector interpretted as $[I_{xx}\, I_{yy}\, I_{zz}\, I_{xy}\, I_{yz}\, I_{xz}]$
`link.m`	r+a	link mass
`link.r`	r+a	3×1 link COG vector
`link.G`	r+a	gear ratio
`link.Jm`	r+a	motor inertia
`link.B`	r+a	viscous friction
`link.Tc`	r	Coulomb friction, 1×2 vector where $[\tau^+\ \tau^-]$
`link.Tc`	a	Coulomb friction; for symmetric friction this is a scalar, for asymmetric friction it is a 2-element vector for positive and negative velocity
`link.dh`	r+a	row of legacy DH matrix
`link.dyn`	r+a	row of legacy DYN matrix
`link.qlim`	r+a	joint coordinate limits, 2-vector
`link.islimit(q)`	r	return true if value of `q` is outside the joint limit bounds
`link.offset`	r+a	joint coordinate offset (see discussion for `robot` object).

Funções associadas ao LINK

method	Operation	Returns
robot.n	r	number of joints
robot.link	r+a	cell array of link objects
robot.name	r+a	robot name string
robot.manuf	r+a	robot manufacturer string
robot.comment	r+a	general comment string
robot.gravity	r+a	3-element vector defining gravity direction
robot.mdh	r	DH convention: 0 if standard, 1 if modified. Determined from the link objects.
robot.base	r+a	homogeneous transform defining base of robot
robot.tool	r+a	homogeneous transform defining tool of robot
robot.dh	r	legacy DH matrix
robot.dyn	r	legacy DYN matrix
robot.q	r+a	joint coordinates
robot.qlim	r+a	joint coordinate limits, $n \times 2$ matrix
robot.islimit	r	joint limit vector, for each joint set to -1, 0 or 1 depending if below low limit, OK, or greater than upper limit
robot.offset	r+a	joint coordinate offsets
robot.plotopt	r+a	options for **plot**()
robot.lineopt	r+a	line style for robot graphical links
robot.shadowopt	r+a	line style for robot shadow links
robot.handle	r+a	graphics handles

Funções associadas ao ROBOT

Para um **manipulador cartesiano de 2 elos**:

- 2 juntas rotacionais no eixo ***Z***;
- *Links* de 0.30m cada;

Parâmetros de D-H: α_i a_i θ_i d_i (σ_i)

LINK	α_i	a_i	θ_i	d_i
1	0	0.30	θ_1	0
2	0	0.30	θ_2	0

Criando os links:

>> L1 = link ([0 0.30 0 0 0]);

>> L2 = link ([0 0.30 0 0 0]);

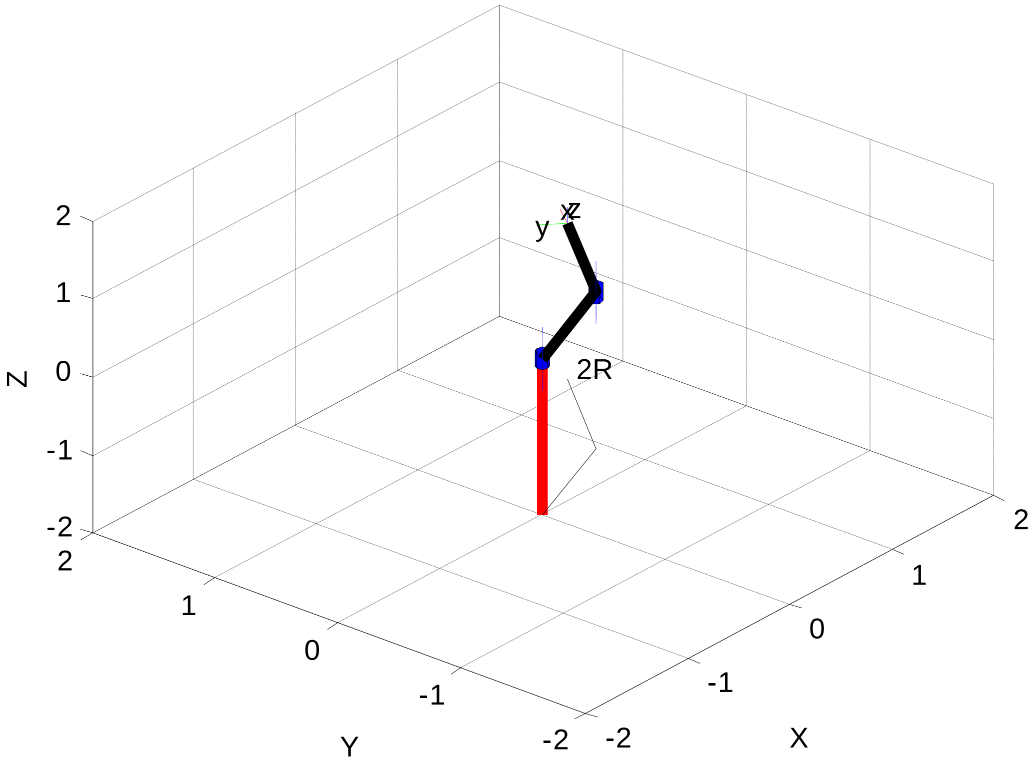
Criando o robô:

>> r1 = robot({L1 L2})

Inserindo um nome para o robô:

>> r.name = '2R'

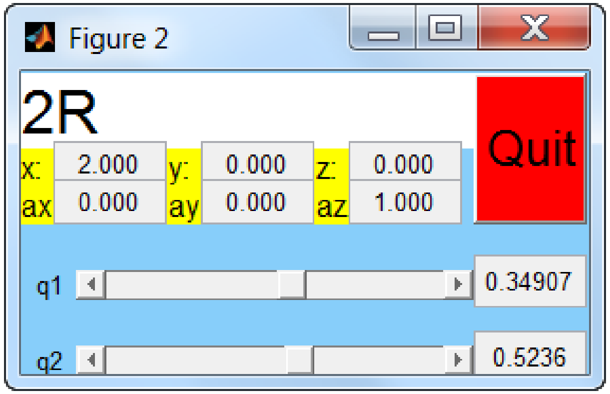
```
>> L1

L1 =

  0.000000  1.000000    0.000000    0.000000    R   (std)

>> L2

L2 =

  0.000000  1.000000    0.000000    0.000000    R   (std)

>> r

r =

2R (2 axis, RR)
        grav = [0.00 0.00 9.81]      standard D&H parameters

  alpha      A         theta        D       R/P
0.000000    1.000000    0.000000    0.000000    R   (std)
0.000000    1.000000    0.000000    0.000000    R   (std)

>> q = [pi/9 pi/6];
>> plot(r, q)
>> drivebot(r)
```



Considerando o vetor posição ***q***:

q = [pi/9 pi/6];

θ_1 = 20°; θ_2 = 30°;



A função ***fkine*** é usada para computar a cinemática direta e retorna a matriz de transformação homogênea final:

```
>> fkine (robot, q)
```

Onde: **robot** = modelo do robô; e **q** = vetor na posição das juntas.
Se ***q = [pi/9 pi/6]***:

```
>> fkine (r1, q)

ans =

   0.6428   -0.7660        0             0.4747
   0.7660    0.6428        0             0.3324
        0         0        1.0000        0
        0         0        0             1.0000
```

A função ***transl()*** é usada para computar o componente translacional de uma matriz de transformação homogênea.

E as funções ***rotx, roty, rotz*** retornam as matrizes de rotação em θ radianos em torno de X, Y e Z.

```
>> m = fkine (r1, q);
>> transl(m);
ans =

   0.4747
   0.3324
        0

>> R1z = rotz(pi/9)
>> R2z = rotz(pi/6)
```

Symbol	Dimensions	Description
`l`	**link**	manipulator link object
`q`	$1 \times n$	joint coordinate vector
`q`	$m \times n$	m-point joint coordinate trajectory
`qd`	$1 \times n$	joint velocity vector
`qd`	$m \times n$	m-point joint velocity trajectory
`qdd`	$1 \times n$	joint acceleration vector
`qdd`	$m \times n$	m-point joint acceleration trajectory
`robot`	**robot**	robot object
`T`	4×4	homogeneous transform
`T`	$4 \times 4 \times m$	m-point homogeneous transform trajectory
`Q`	**quaternion**	unit-quaternion object
`M`	1×6	vector with elements of 0 or 1 corresponding to Cartesian DOF along X, Y, Z and around X, Y, Z. 1 if that Cartesian DOF belongs to the task space, else 0.
`v`	3×1	Cartesian vector
`t`	$m \times 1$	time vector
`d`	6×1	differential motion vector

*Funções para manipuladores com **n eixos***

Homogeneous Transforms	
`eul2tr`	Euler angle to homogeneous transform
`oa2tr`	orientation and approach vector to homogeneous transform
`rotvec`	homogeneous transform for rotation about arbitrary vector
`rotx`	homogeneous transform for rotation about X-axis
`roty`	homogeneous transform for rotation about Y-axis
`rotz`	homogeneous transform for rotation about Z-axis
`rpy2tr`	Roll/pitch/yaw angles to homogeneous transform
`tr2eul`	homogeneous transform to Euler angles
`tr2rot`	homogeneous transform to rotation submatrix
`tr2rpy`	homogeneous transform to roll/pitch/yaw angles
`transl`	set or extract the translational component of a homogeneous transform
`trnorm`	normalize a homogeneous transform

Trajectory Generation	
`ctraj`	Cartesian trajectory
`jtraj`	joint space trajectory
`trinterp`	interpolate homogeneous transforms

Quaternions

`/`	divide quaternion by quaternion or scalar
`*`	multiply quaternion by a quaternion or vector
`inv`	invert a quaternion
`norm`	norm of a quaternion
`plot`	display a quaternion as a 3D rotation
`q2tr`	quaternion to homogeneous transform
`quaternion`	construct a quaternion
`qinterp`	interpolate quaternions
`unit`	unitize a quaternion

Manipulator Models

`link`	construct a robot link object
`nofriction`	remove friction from a robot object
`perturb`	randomly modify some dynamic parameters
`puma560`	Puma 560 data
`puma560akb`	Puma 560 data (modified Denavit-Hartenberg)
`robot`	construct a robot object
`showlink`	show link/robot data in detail
`stanford`	Stanford arm data
`twolink`	simple 2-link example

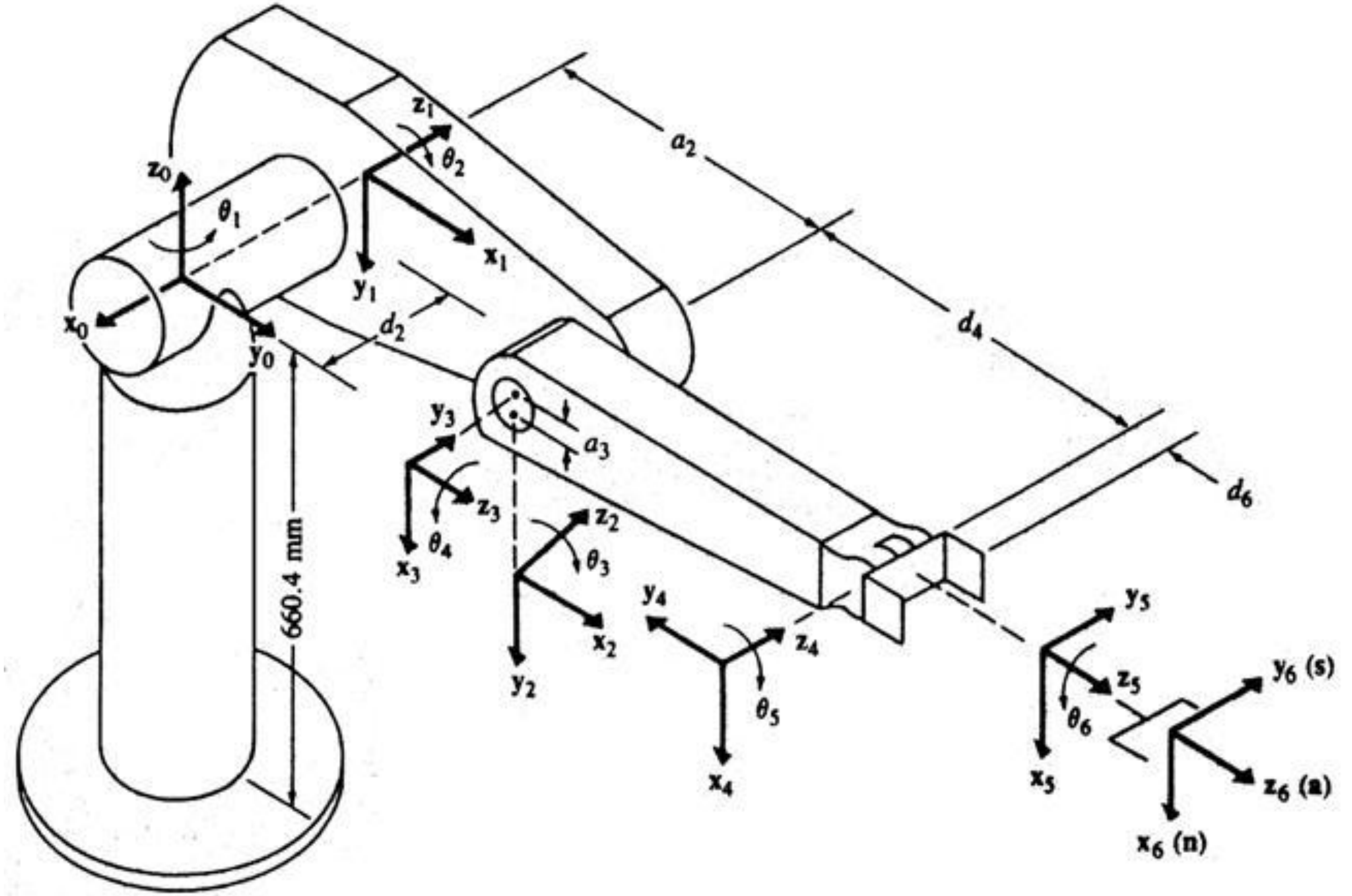
Kinematics	
`diff2tr`	differential motion vector to transform
`fkine`	compute forward kinematics
`ftrans`	transform force/moment
`ikine`	compute inverse kinematics
`ikine560`	compute inverse kinematics for Puma 560 like arm
`jacob0`	compute Jacobian in base coordinate frame
`jacobn`	compute Jacobian in end-effector coordinate frame
`tr2diff`	homogeneous transform to differential motion vector
`tr2jac`	homogeneous transform to Jacobian

Graphics	
`drivebot`	drive a graphical robot
`plot`	plot/animate robot

Dynamics	
`accel`	compute forward dynamics
`cinertia`	compute Cartesian manipulator inertia matrix
`coriolis`	compute centripetal/coriolis torque
`fdyn`	forward dynamics (motion given forces)
`friction`	joint friction
`gravload`	compute gravity loading
`inertia`	compute manipulator inertia matrix
`itorque`	compute inertia torque
`rne`	inverse dynamics (forces given motion)

Other	
`ishomog`	test if matrix is 4×4
`maniplty`	compute manipulability
`rtdemo`	toolbox demonstration
`unit`	unitize a vector

Robot PUMA560:



Cinemática direta

- **Craig, J.C.**, 2005, *Introduction to Robotics: Mechanics and Control*, 3rd Edition, Pearson Education Inc., ISBN 0-201-54361-3
- **Fu, K.S., Gonzales, R.C.,** and **Lee, C.S.G.**, 1987, *Robotics: Control, Sensing, Vision, and Intelligence*, McGraw-Hill Int. Editions, ISBN 0-07-100421-1.
- **Paul, R. P.**, 1981, *Robot Manipulators. Mathematics, Programming and Control*, The MIT Press.
- **Hartenberg, R. S.** and **Denavit, J.**, 1964, *Kinematic Synthesis of Linkages*, McGraw Hill, ISBN 64-23251.
- **Corke, P.**, Robotics Toolbox for MatLab (Release 7).

Bibliografia