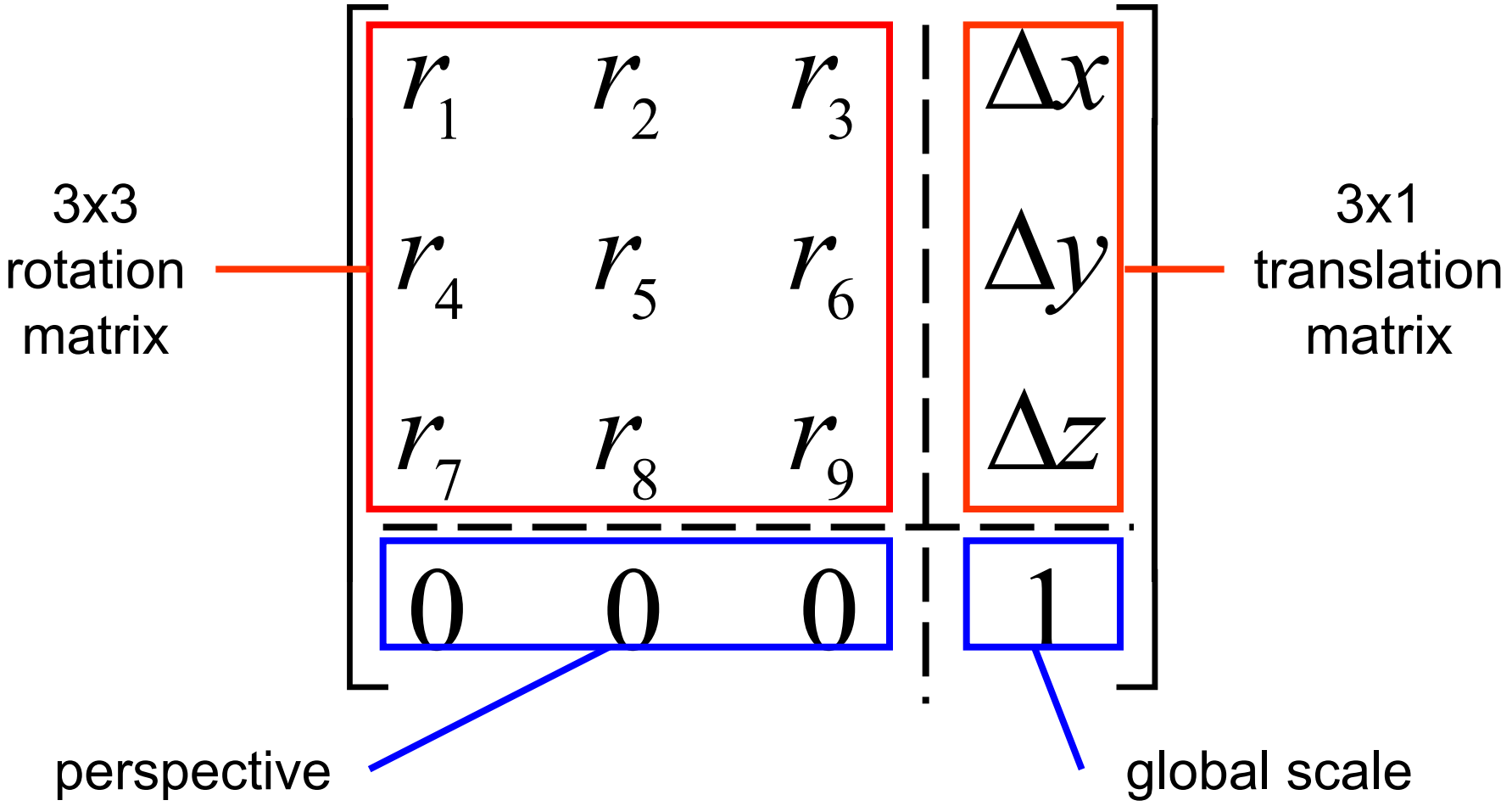
Robótica

Prof. Reinaldo Bianchi
Centro Universitário da FEI
2016

3ª Aula

Parte B - Matlab

Matriz de Transformação Homogênea 3D



Notação de Denavit-Hartenberg

- Um robô pode ser especificado ao se descrever os valores de 4 parâmetros para cada elo:
  - comprimento (i-1), torção (i-1), offset (i) e ângulo (i).
- A definição da mecânica de um manipulador usando estes parâmetros segue a notação de Denavit-Hartenberg.
- A Notação D-H especifica ainda...

Sistemas de referências

- Cada corpo elementar (*elo*) da cadeia cinemática deve ser fixado em um sistema de referência (*frame*).
- Existe uma convenção para anexar sistemas de referências aos elos, dada pela Notação D-H:
  - *Frames* são numerados de acordo com o elo ao qual ele está ligado.
  - *Frame* $\{i\}$ está ligado ao elo i.

Notação D-H a partir dos *frames*

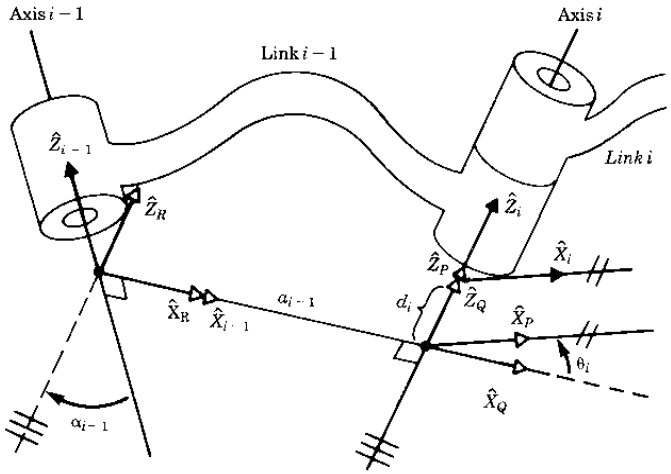
- a_i: a distância entre os eixos Z_i e Z_{i+1} medida sobre o eixo X_i.
- α_i: o ângulo entre os eixos Z_i e Z_{i+1} medida sobre o eixo X_i.
- d_i: a distância entre os eixos $X_{i\text{-}1}$ e X_i medida sobre o eixo Z_i.
- θ_i: o ângulo entre os eixos $X_{i\text{-}1}$ e X_i medidos sobre o eixo Z_i.

Transformação para um elo.

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_{i-1} & -sen\alpha_{i-1} & 0 \\ 0 & sen\alpha_{i-1} & \cos\alpha_{i-1} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & a_{i-1} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta_i & -sen\theta_i & 0 & 0 \\ sen\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} \cos\theta_i & -sen\theta_i & 0 & a_{i-1} \\ sen\theta_i\cos\alpha_{i-1} & \cos\theta_i\cos\alpha_{i-1} & -sen\alpha_{i-1} & -sen\alpha_{i-1}d_i \\ sen\theta_i sen\alpha_{i-1} & \cos\theta_i sen\alpha_{i-1} & \cos\alpha_{i-1} & \cos\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$${}^{i-1}_{i}T = {}^{i-1}_{R}T\,{}^{R}_{Q}T\,{}^{Q}_{P}T\,{}^{P}_{i}T$$



Notação para diminuir o tamanho:
sen = s
cos = c

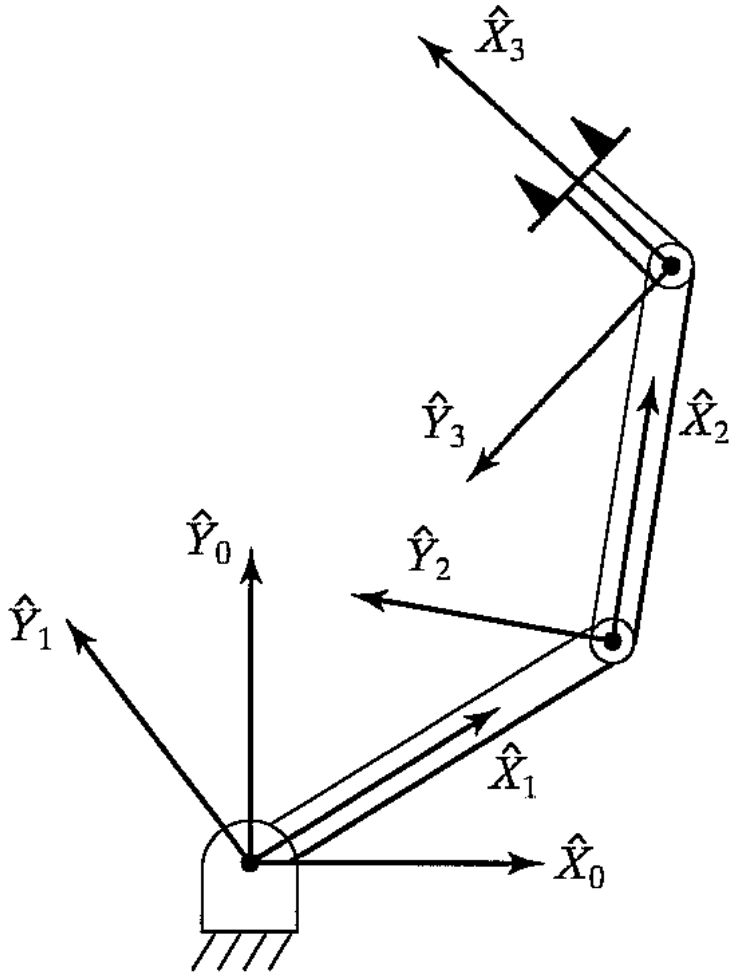
Matriz cinemática

- Relaciona o sistema de coordenadas solidárias à base do robô com o sistema de coordenadas associadas à sua ferramenta terminal.
- Em coordenadas homogêneas.
- Resulta do produto das matrizes de transformação de cada elo:
  - Transforma passo a passo.

$$ {}^{0}_{n}T = T_1 T_2 T_3 \cdots T_i \cdots T_n $$

Exemplo 3: Matriz Cinemática para o robô 3R

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3



Matriz cinemática para o robô 3R

$$
{}^{0}_{3}T = \begin{bmatrix} \cos(\theta_1) & -\sin(\theta_1) & 0 & 0 \\ \sin(\theta_1) & \cos(\theta_1) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_2) & -\sin(\theta_2) & 0 & L_1 \\ \sin(\theta_2) & \cos(\theta_2) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \cos(\theta_3) & -\sin(\theta_3) & 0 & L_2 \\ \sin(\theta_3) & \cos(\theta_3) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{0}_{3}T = \begin{bmatrix} \cos(\theta_1+\theta_2+\theta_3) & -\sin(\theta_1+\theta_2+\theta_3) & 0 & L_1\cos(\theta_1)+L_2\cos(\theta_1+\theta_2) \\ \sin(\theta_1+\theta_2+\theta_3) & \cos(\theta_1+\theta_2+\theta_3) & 0 & L_1\sin(\theta_1)+L_2\sin(\theta_1+\theta_2) \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

$$
{}^{0}_{3}T = \begin{bmatrix} c_{123} & -s_{123} & 0 & L_1c_1+L_2c_{21} \\ c_{123} & c_{123} & 0 & L_1s_1+L_2s_{12} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
$$

Robotics Toolbox para o Matlab

rvctools

Robotics Toolbox para o Matlab

- Toolbox de livre distribuição (9ª edição):
  - http://petercorke.com/Robotics_Toolbox.html
- Possui modelo de alguns manipuladores prontos:
  - PUMA560
  - Stanford Arm
  - ...
- Permite criar seu próprio modelo.

Uso

- Copie o diretório do toolbox de :
  - w:\eng\ele\bianchi\robotica\rvctools
- para o diretório c:\alunos
- Mude de diretorio no matlab:
  - cd c:\alunos\rvctools
- O comando abaixo deve ser executado antes de iniciar o uso do toolbox:
  - startup_rvc

Criando um robô no Matlab

Ângulos em radianos

- Criando os links:
  - `L = Link([`θ_i d_i a_i α_i`], opção)`
  - onde opção (que é opcional) =
    - ‘standard’ – parametros padrão.
    - ‘r’ para Rotacional (default) e
    - ‘p’ para Prismática
- Criando o robô:
  - `r = SerialLink([Link1 Link2 ...])`

Notação D-H

- a_i: a distância entre os eixos Z_i e Z_{i+1} medida sobre o eixo X_i.
- α_i: o ângulo entre os eixos Z_i e Z_{i+1} medida sobre o eixo X_i.
- d_i: a distância entre os eixos $X_{i\text{-}1}$ e X_i medida sobre o eixo Z_i.
- θ_i: o ângulo entre os eixos $X_{i\text{-}1}$ e X_i medidos sobre o eixo Z_i.

Métodos associados ao link

Methods

A	link transform matrix
RP	joint type: 'R' or 'P'
friction	friction force
nofriction	Link object with friction parameters set to zero
dyn	display link dynamic parameters
islimit	test if joint exceeds soft limit
isrevolute	test if joint is revolute
isprismatic	test if joint is prismatic
display	print the link parameters in human readable form
char	convert to string

Propriedades associadas ao link

Properties (read/write)

theta	kinematic: joint angle
d	kinematic: link offset
a	kinematic: link length
alpha	kinematic: link twist
sigma	kinematic: 0 if revolute, 1 if prismatic
mdh	kinematic: 0 if standard D&H, else 1
offset	kinematic: joint variable offset
qlim	kinematic: joint variable limits [min max]
m	dynamic: link mass
r	dynamic: link COG wrt link coordinate frame 3×1
I	dynamic: link inertia matrix, symmetric 3×3, about link COG.
B	dynamic: link viscous friction (motor referred)
Tc	dynamic: link Coulomb friction
G	actuator: gear ratio
Jm	actuator: motor inertia (motor referred)

Métodos associados ao robô

Methods

plot	display graphical representation of robot
teach	drive the graphical robot
isspherical	test if robot has spherical wrist
islimit	test if robot at joint limit
fkine	forward kinematics
ikine6s	inverse kinematics for 6-axis spherical wrist revolute robot
ikine3	inverse kinematics for 3-axis revolute robot
ikine	inverse kinematics using iterative method
jacob0	Jacobian matrix in world frame
jacobn	Jacobian matrix in tool frame
maniplty	manipulability
jtraj	a joint space trajectory
accel	joint acceleration
coriolis	Coriolis joint force
dyn	show dynamic properties of links
fdyn	joint motion
friction	friction force
gravload	gravity joint force
inertia	joint inertia matrix
nofriction	set friction parameters to zero
rne	joint torque/force
payload	add a payload in end-effector frame
perturb	randomly perturb link dynamic parameters

Propried. associadas ao robô

Properties (read/write)

links	vector of Link objects ($1 \times N$)
gravity	direction of gravity [gx gy gz]
base	pose of robot's base (4×4 homog xform)
tool	robot's tool transform, T6 to tool tip (4×4 homog xform)
qlim	joint limits, [qmin qmax] ($N \times 2$)
offset	kinematic joint coordinate offsets ($N \times 1$)
name	name of robot, used for graphical display
manuf	annotation, manufacturer's name
comment	annotation, general comment
plotopt	options for plot() method (cell array)

Object properties (read only)

n	number of joints
config	joint configuration string, eg. 'RRRRRR'
mdh	kinematic convention boolean (0=DH, 1=MDH)

Criando um manipulador 2R

- Queremos criar o seguinte manipulador 2R:
  - 2 juntas rotacionais no eixo z
  - links de 1 metro cada.
- Parâmetros D-H:

Link	a_i	α_i	d_i	θ_i
1	1	0	0	θ_1
2	1	0	0	θ_2

Comandos para criar um 2R

- Criando os links:

```
L1 = Link([0 0 1 0])
L2 = Link([0 0 1 0])
```

- Criando o robô:

```
r = SerialLink([L1 L2])
```

Gerou um robô...

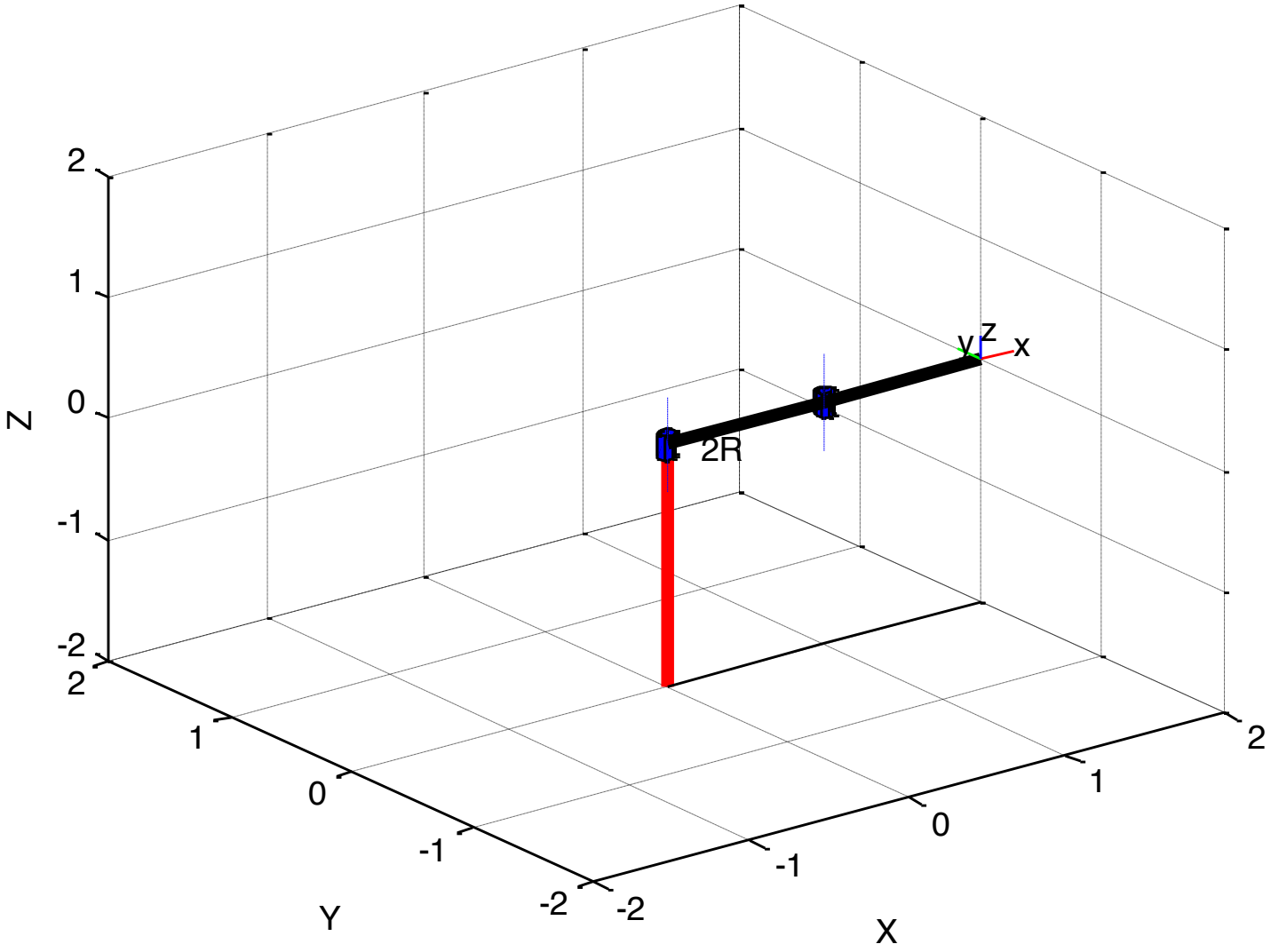
```
>> L1 = Link([ 0 0 1 0])
L1 =
 theta=q, d= 0, a= 1, alpha=          0 (R,stdDH)
>> L2 = Link([ 0 0 1 0])
L2 =
 theta=q, d= 0, a= 1, alpha=          0 (R,stdDH)
>> r = SerialLink([L1 L2])
r =
robot (2 axis, RR, stdDH)
+---+----------+----------+----------+----------+
| j |     theta |         d |         a |     alpha |
+---+----------+----------+----------+----------+
|  1|         q1|          0|          1|          0|
|  2|         q2|          0|          1|          0|
+---+----------+----------+----------+----------+

grav =    0  base = 1  0  0  0   tool =  1  0  0  0
          0          0  1  0  0           0  1  0  0
       9.81          0  0  1  0           0  0  1  0
                     0  0  0  1           0  0  0  1
```

Para visualizar o robô

- **`r.plot(q)`:**
  - Desenha o robô **`r`** na posição **`q`**, um vetor que define os ângulos das juntas.
- **`r.teach()`:**
  - Permite visualizar e modificar os valores das juntas e gravar posições.
- Use as ferramentas  para:
  - Ampliar, Reduzir, Girar.

r.plot([0 0])



`r.teach()`

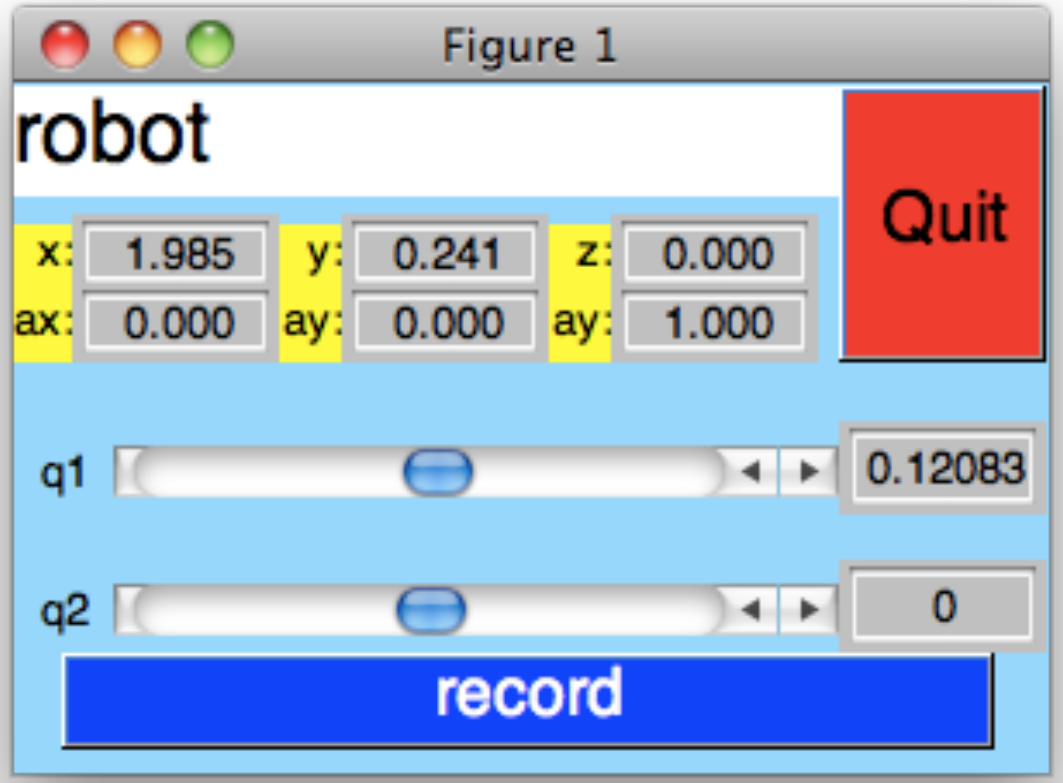


Figure 1
robot
Quit
x: 1.985
y: 0.241
z: 0.000
ax: 0.000
ay: 0.000
ay: 1.000
q1
0.12083
q2
0
record

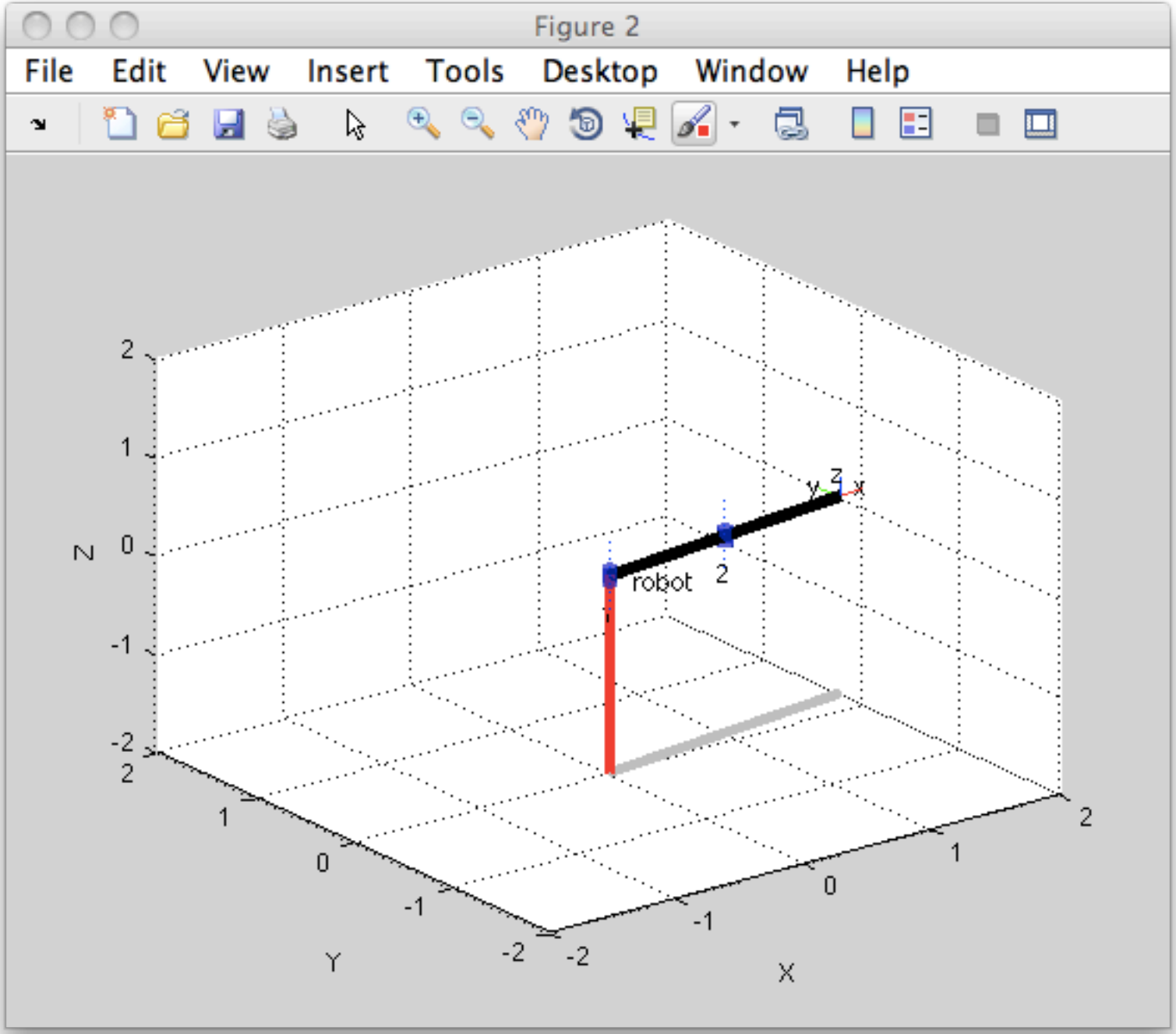


Figure 2
File Edit View Insert Tools Desktop Window Help
robot
2
z
y
x
N
Y
X
2
1
0
-1
-2

Usando teach e plot

- Voce pode usar o teach para gravar pontos, que ficam salvos na variável qplot.
- Depois, pode executar os movimentos de uma vez, usando:

```
r.plot(r.qteach)
r.plot(r.qteach,'delay',1)
```

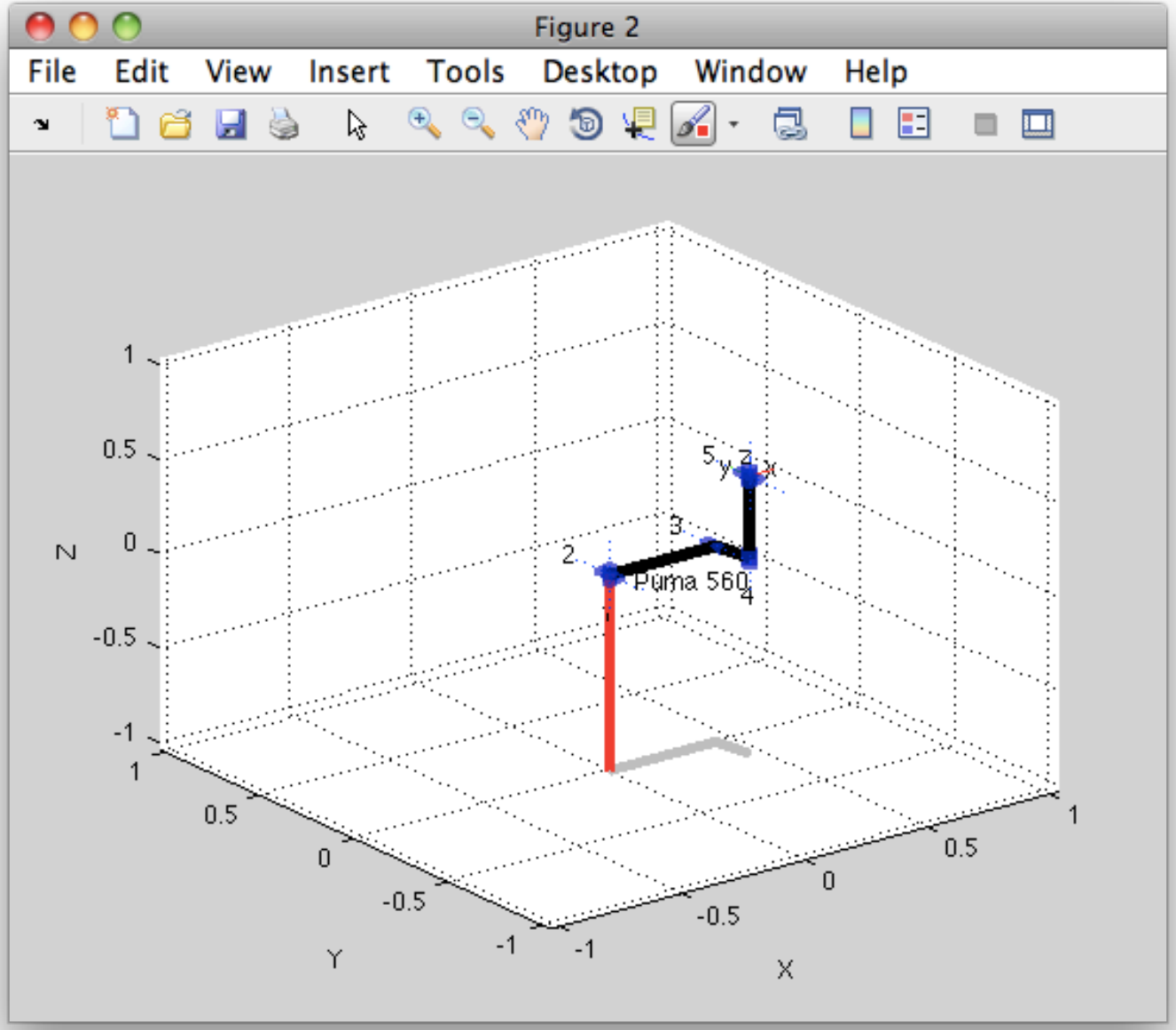
Outros métodos…

- R.display() displays the robot parameters in human-readable form.
- R.dyn() displays the inertial properties of the robot:
  - mass, centre of mass, inertia, gear ratio, motor inertia and motor friction.
- Entre outras…

Robô Puma 560

- O toolkit possui uma função que cria um robo tipo Puma 560:
- Criando um robô PUMA:
  - **`mdl_puma560`**
- Exibindo o robô :
  - **`p560.plot([0 0 0 0 0 0])`**
- Note que o nome do robô criado é p560

```
p560.plot([0 0 0 0 0 0])
```



Computando a cinemática direta

- O método **`fkine`** é usada para computar a cinemática direta:
  - **`robot.fkine(q)`**

  onde:
  - **`robot`** = variável do robô.
  - **`q`** = vetor da posição das juntas.
  - RETORNA: a matriz de transformação...
- Teste:
  - **`r.fkine([0 0])`**

```
r.fkine([0 0])
```

```
ans =

     1     0     0     2
     0     1     0     0
     0     0     1     0
     0     0     0     1
```

r.plot([0 0])

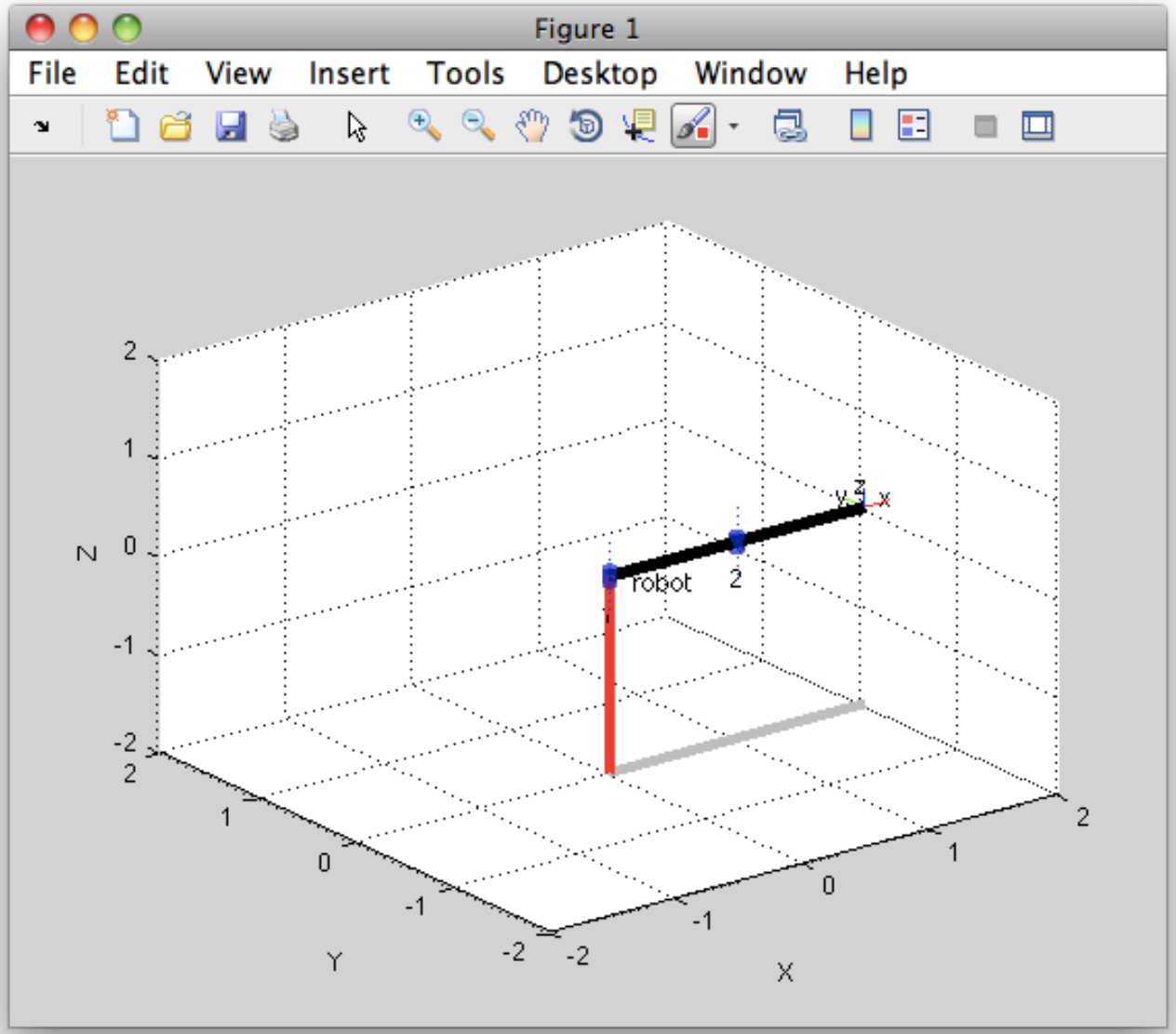


Figure 1
File Edit View Insert Tools Desktop Window Help
robot
2
Z
X
Y
N
2
1
0
-1
-2

r.fkine([0 pi/2])

```
ans =

    0.00   -1.00         0    1.00
    1.00    0.00         0    1.00
       0       0    1.0000       0
       0       0         0    1.00
```

r.plot([0 pi/2])

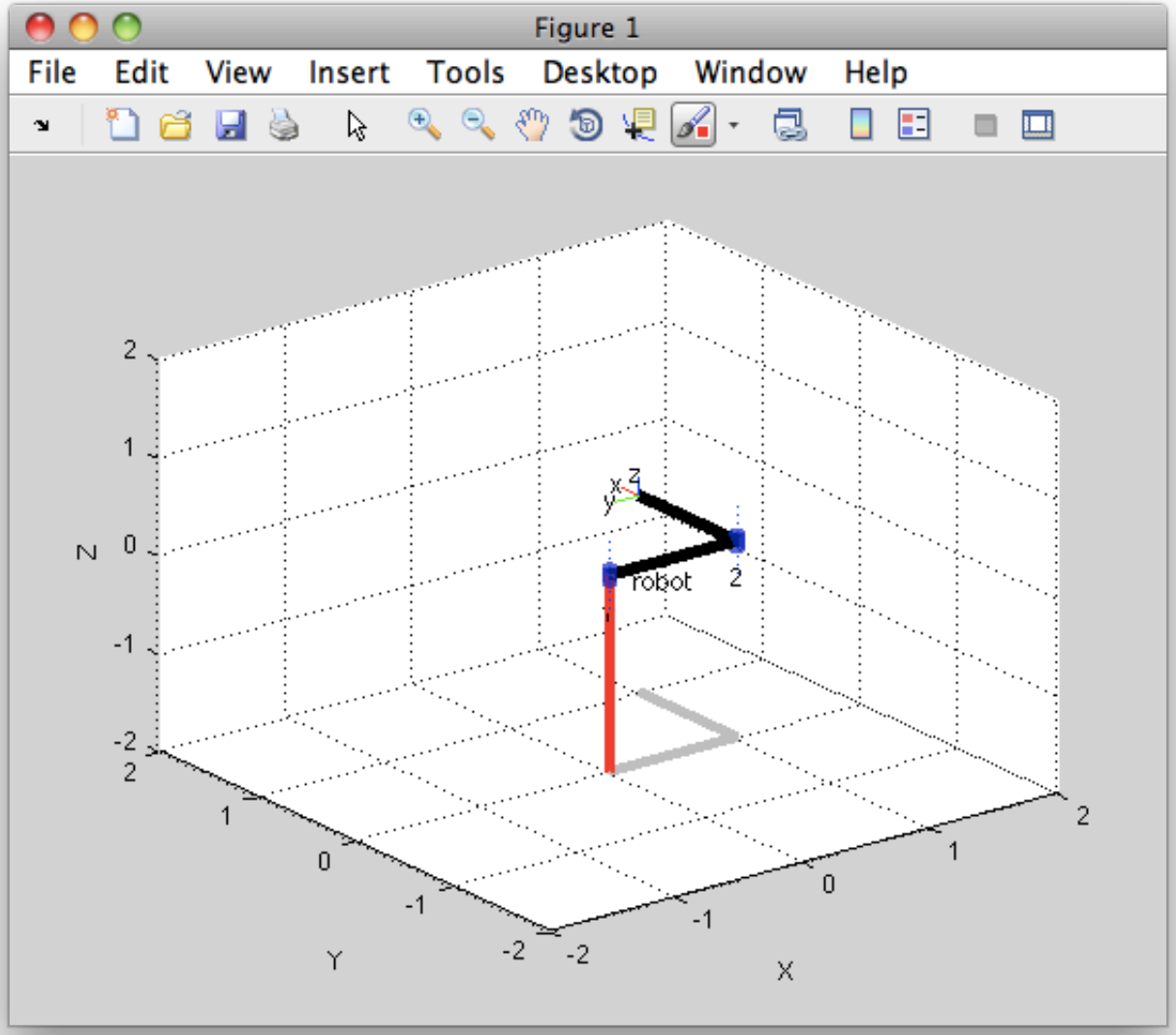
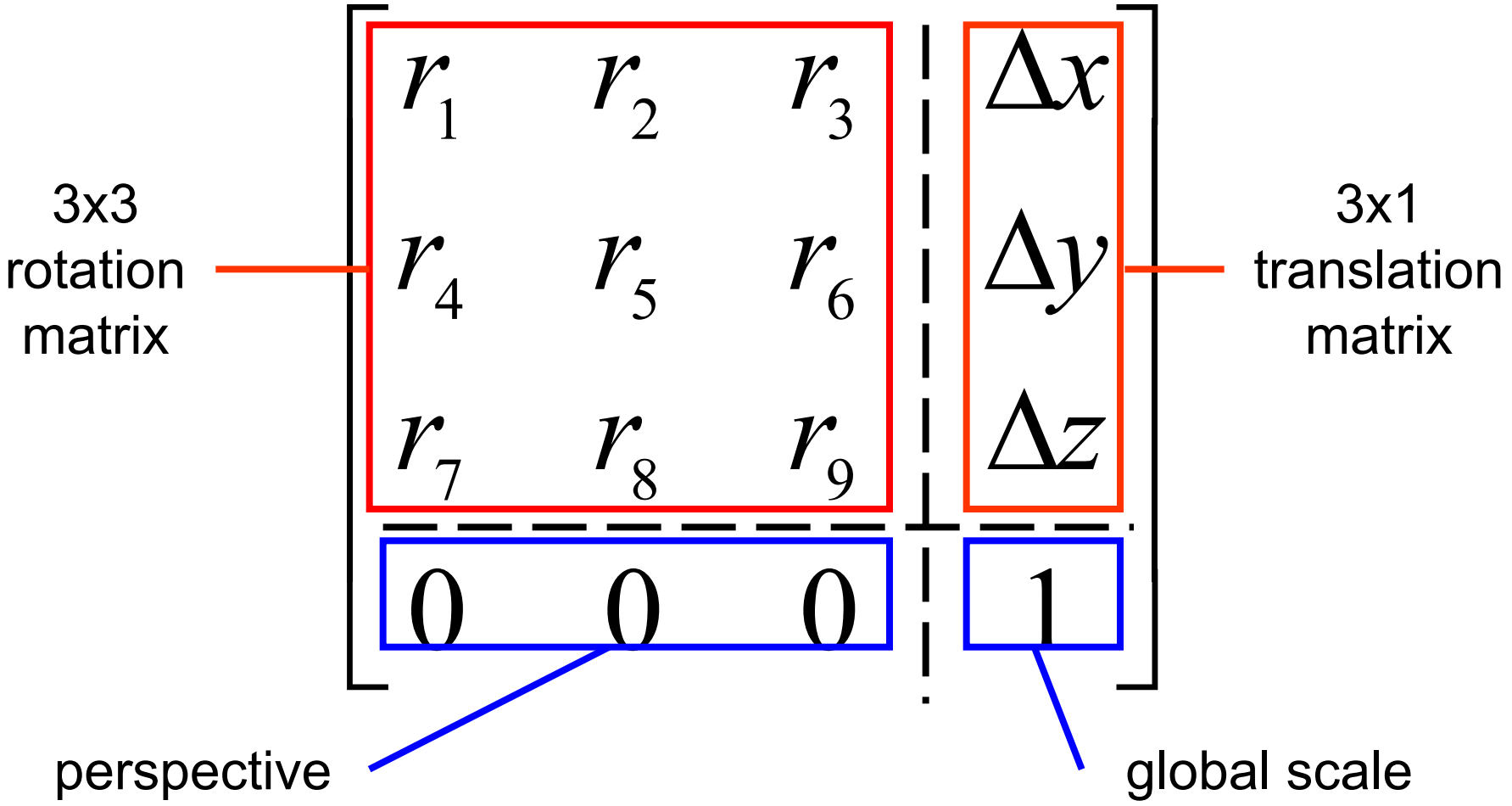


Figure 1
File Edit View Insert Tools Desktop Window Help
robot
Z
X
Y
2
1
0
-1
-2
2
1
0
-1
-2
2
1
0
-1
-2

Ou seja, fkine retorna... A Matriz de Transformação Homogênea 3D



Exercício 1:

- Utilize a função teach para gravar os seguintes pontos:

  (0 0), (π/8 0), (π/4 0), (3π/8 0), (π/2 0),
  (π/2 π/8), (π/2 π/4), (π/2 3π/8), (π/2 π/2),
  (π/2 5π/8), (π/2 3π/4), (π/2 7π/8), (π/2 π)

- E use a função plot para simular o movimento do manipulador:

```
r.plot(r.qteach,'delay',1)
```

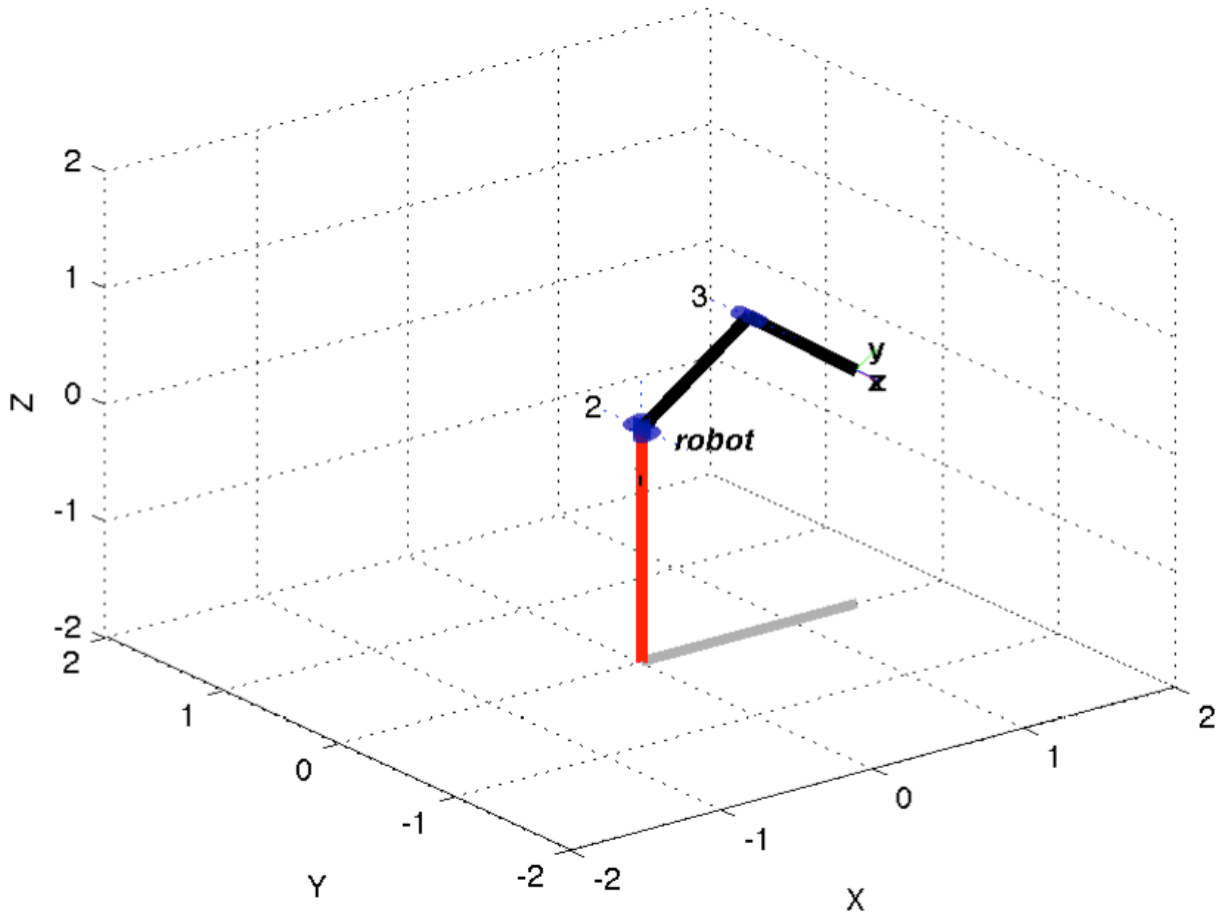
Exercício 2:

- Calcular a posição do manipulador 2R usando fkine para os pontos:

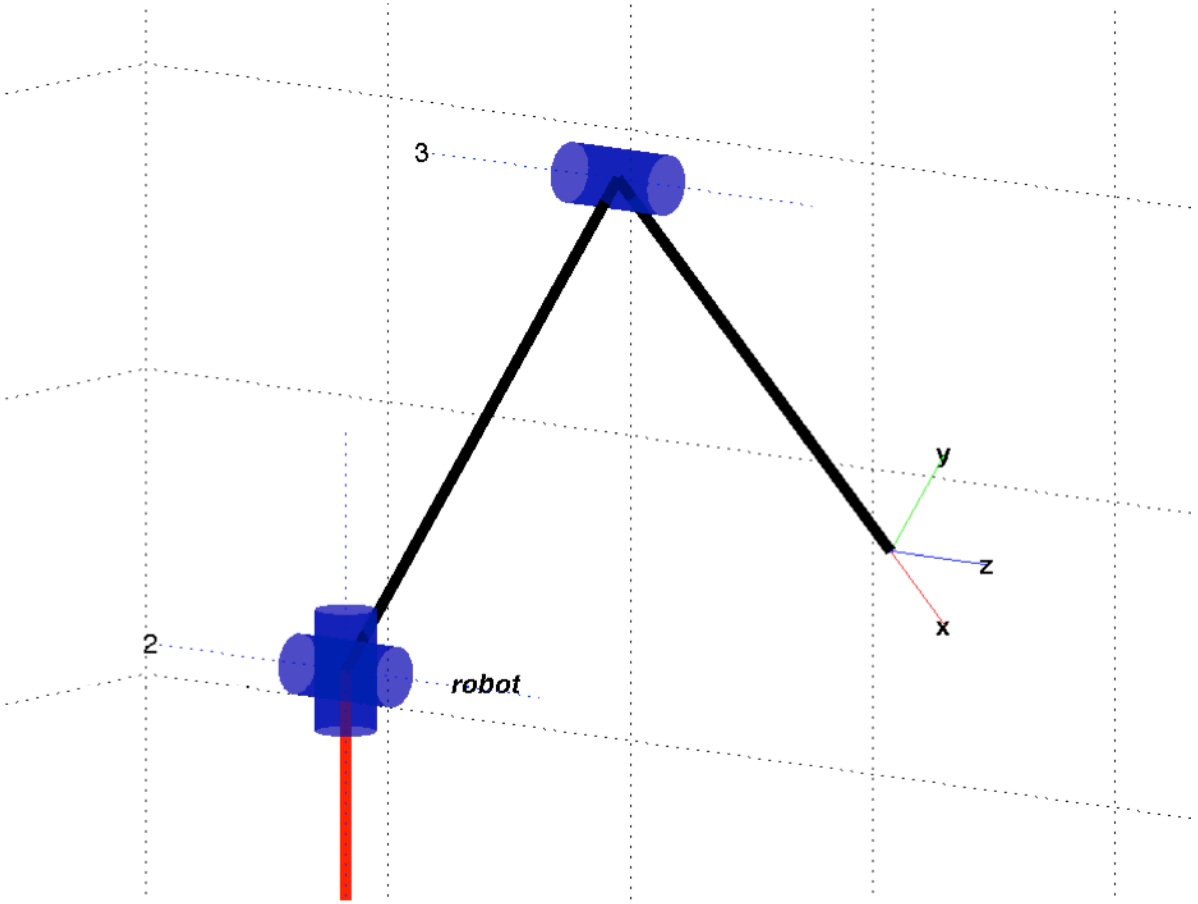
  (0 0), (π/8 0), (π/4 0), (3π/8 0), (π/2 0),
  (π/2 π/8), (π/2 π/4), (π/2 3π/8), (π/2 π/2),
  (π/2 5π/8), (π/2 3π/4), (π/2 7π/8), (π/2 π)

- E plote em um gráfico...
  - plot ([x1 x2 ...], [y1 y2 ..])

Exercício 3:
Crie o robô da figura abaixo



Exercício 3:
Crie o robô da figura abaixo



Para brincar em casa...

- Crie um robô PUMA:
  - mdl_puma560 %define um robô puma
- Brinque com ele:
  - P560.teach() % permite controle
- Calcule a cinemática direta para o puma, em algumas posições diferentes a sua escolha.

Matlab help

- who: mostra as variáveis
- clear: limpa a memória.
- clc: limpa a tela de comando.
- cd: muda de diretório.
- home: limpa a tela.
- help general: o básico
- workspace: mostra graficamente os objetos existentes.

Fim… próxima aula (de teoria)…