An open-source package for analysis and control of serial-link robotic manipulators

M. Morelli

mmorelli@users.sourceforge.net

Centro “E.Piaggio”
Facoltà di Ingegneria
Università di Pisa
Pisa, Italy

25/05/2011, A21, Robotica 1

Outline

1. Robotic manipulators modelling with ScicosLab
   - Rigid body transformations
   - Building serial-link manipulator models
   - Analysis of serial-link manipulators
2. Robot control systems design using Scicos
   - Basic concepts
   - Motion control
   - Further applications
3. Robot control code generation for use with Linux RTAI
   - Developing RTAI-based centralized controllers with RTAI-Lab

About MRT [Corke, 1996]

- Developed by Dr. Peter Corke (CSIRO, Australia);
- very popular toolbox (more than 10500 downloads!);
- release 8 (December 2008) is under GNU LGPL;
- available at `http://petercorke.com/`.

Purpose of MRT

Enable students and teachers to better understand the theoretical concepts behind classical robotics.

Features of MRT

- Manipulation of fundamental datatypes such as:
  - homogeneous transformations;
  - quaternions;
  - trajectories.
- Functions for serial-link rigid-body manipulators:
  - forward and inverse kinematics;
  - differential kinematics;
  - forward and inverse dynamics.
- SIMULINK® blockset library.

Points of strenght

Based on MATLAB®, MRT takes advantage of:

- a powerful environment for linear algebra;
- a powerful environment for graphical presentation;
- an easy integration with other toolboxes;
- the SIMULINK® environment for dynamic systems simulation.

Main drawback

MRT is an open source software, requiring a proprietary and expensive software environment to run.

RTSS–the Robotics Toolbox for Scilab/Scicos
Milestone 1 (2008)

Development objective

Porting the functionalities of MRT to Scilab/Scicos version 4.0+, under a free software license.

Achievements/Features

- Porting process successfully completed;
- free software licensed under GNU GPL.

Release history

- Initial release, the 0.1.0, released in Oct. 2007;
- release 0.3.0 available for Scilab-4.1.2 in Feb. 2008;
- more than 3000 downloads.

Development objectives

Internal code refactoring and integration with the Scicos's code generation tools.

Achievements/Features

- Modular framework facilitating maintainability and portability;
- fully integrated with Linux RTAI/Xenomai code generators.

Release history

- Release 1.0.0b1 available for Scilab-BUILD4 in Sep. 2009;
- more than 1300 downloads.

Education

Gain insights on subjects such as:

- homogeneous transformations;
- Cartesian and joint-space trajectories;
- forward and inverse kinematics;
- differential motions and manipulator Jacobians;
- forward and inverse dynamics.

Potential applications in research

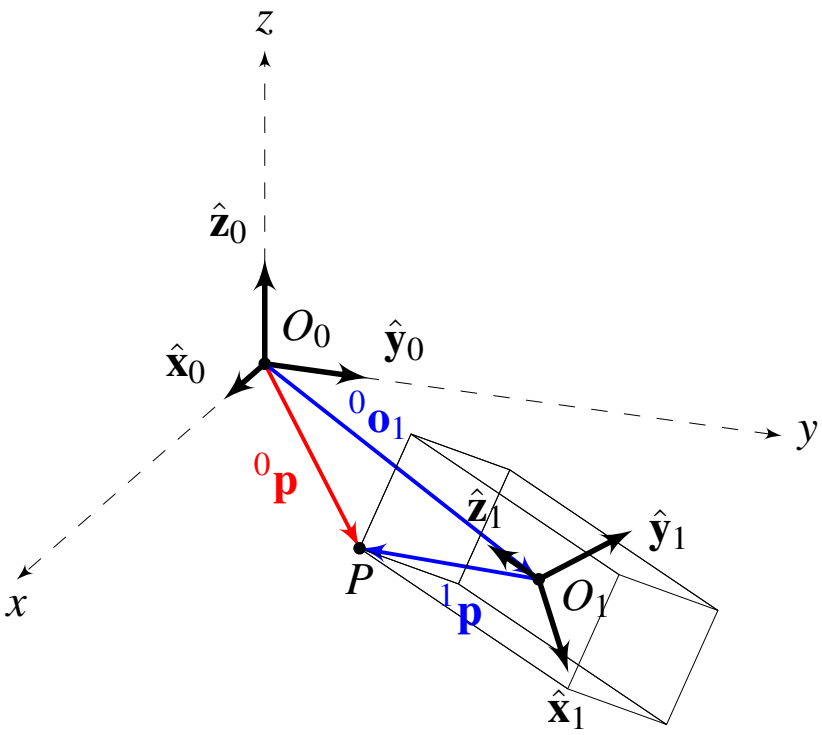
- Robot design optimization
  - Singularity analysis and kinematic optimization;
  - torques analysis for different arm configurations;
  - design verification in real-time simulations.
- Rapid control prototyping
  - Development and verification of control tasks;
  - automatic control code generation;
  - design optimization through HIL simulations.

Outline

1. Robotic manipulators modelling with ScicosLab
   - Rigid body transformations
   - Building serial-link manipulator models
   - Analysis of serial-link manipulators
2. Robot control systems design using Scicos
   - Basic concepts
   - Motion control
   - Further applications
3. Robot control code generation for use with Linux RTAI
   - Developing RTAI-based centralized controllers with RTAI-Lab

Representing 3D translations and orientations

Homogeneous transformations



Point P represented in different coordinate frames

Coordinate transformation

$$ {}^0\mathbf{p} = {}^0\mathbf{o}_1 + {}^0\mathbf{R}_1 {}^1\mathbf{p} $$

Compact representation

$$ \begin{bmatrix} {}^0\mathbf{p} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^0\mathbf{R}_1 & {}^0\mathbf{o}_1 \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} {}^1\mathbf{p} \\ 1 \end{bmatrix} $$

Representing 3D translations and orientations

Playing with homogeneous transformations

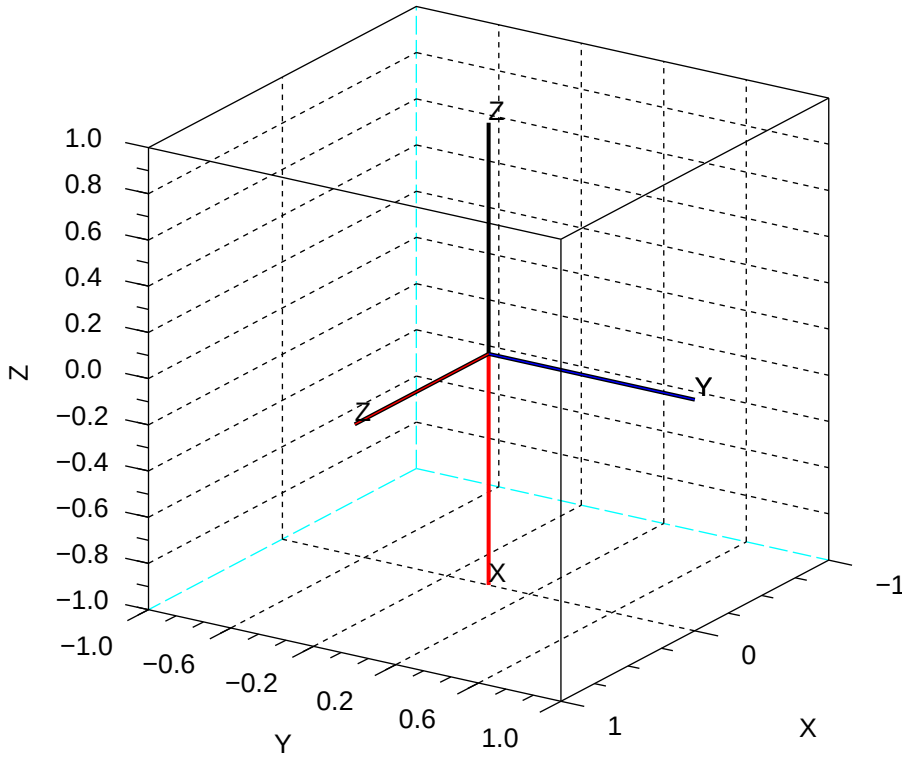
Example

Create the homogeneous transform

$$ {}^{w}\mathbf{T}_{b} = Transl_{\hat{\mathbf{x}}}(0.5\mathrm{m})Rot_{\hat{\mathbf{y}}}(\pi/2) $$

Solution

```
twb = rt_transl(0.5, 0, 0)*..
      rt_roty(%pi/2),
 twb =

   0.   0.   1.   0.5
   0.   1.   0.   0.
 - 1.   0.   0.   0.
   0.   0.   0.   1.
```



Orientation of frame $\{\mathcal{B}\}$ (colored) with respect to frame $\{\mathcal{W}\}$ (black)

Representing 3D translations and orientations

Other representations of orientation

RTSS also provides full support for other representations:

- Euler angles (ZYZ);
- Roll/Pitch/Yaw angles;
- angle and axis;
- unit quaternion.

Euler angles (ZYZ)

Find a vector of Euler angles describing the rotational part of

$$ {}^{w}\mathbf{T}_{b} = Transl_{\hat{\mathbf{x}}}(0.5\mathrm{m})Rot_{\hat{\mathbf{y}}}(\pi/2) $$
$$ Rot_{\hat{\mathbf{z}}}(-\pi/2) $$

Solution

```
twb = rt_transl(0.5, 0, 0) *..
            rt_roty(%pi/2) *..
               rt_rotz(-%pi/2);
eul = rt_tr2eul(twb),
 eul =

   0.   1.5707963 - 1.5707963
```

Representing 3D translations and orientations
Demonstration script

Transformations

`demos/rt_rttrdemo.sce`

Outline

1. Robotic manipulators modelling with ScicosLab
   - Rigid body transformations
   - Building serial-link manipulator models
   - Analysis of serial-link manipulators
2. Robot control systems design using Scicos
   - Basic concepts
   - Motion control
   - Further applications
3. Robot control code generation for use with Linux RTAI
   - Developing RTAI-based centralized controllers with RTAI-Lab

On the robot modelling capabilities of RTSS

n-DOF manipulator modelling at a glance

RTSS allows to model the typical industrial manipulator.

Modelling with RTSS

- Rigid-body manipulators;
- open kinematic chains;
- arbitrary number of kinematic pairs (n).

Typical industrial manipulator – TX90, courtesy of Stäubli Group

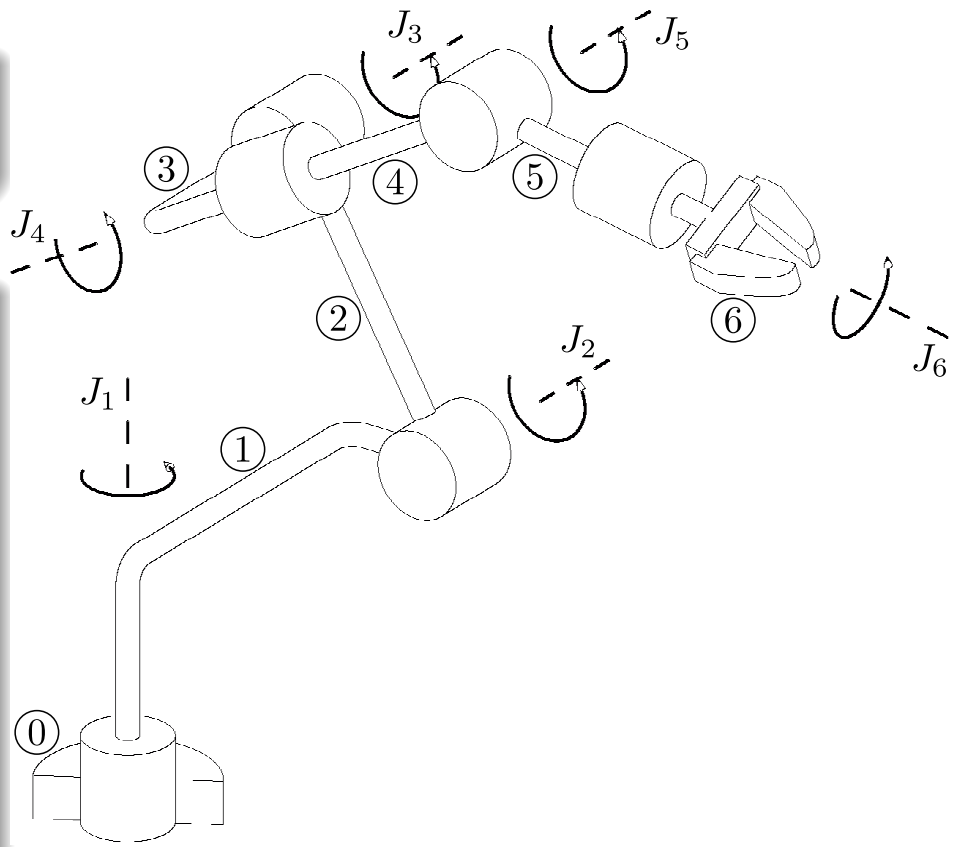
On the robot modelling capabilities of RTSS

n-DOF manipulator modelling at a glance

The robot is modelled as a kinematic chain skeleton.

Kinematic chain representation

- $n+1$ rigid links connected by n joint articulations;
- single-DOF joints: revolute (R) or prismatic (P);
- one end constrained to a base;
- an end-effector mounted to the other end (EE).



Kinematic chain – TX90

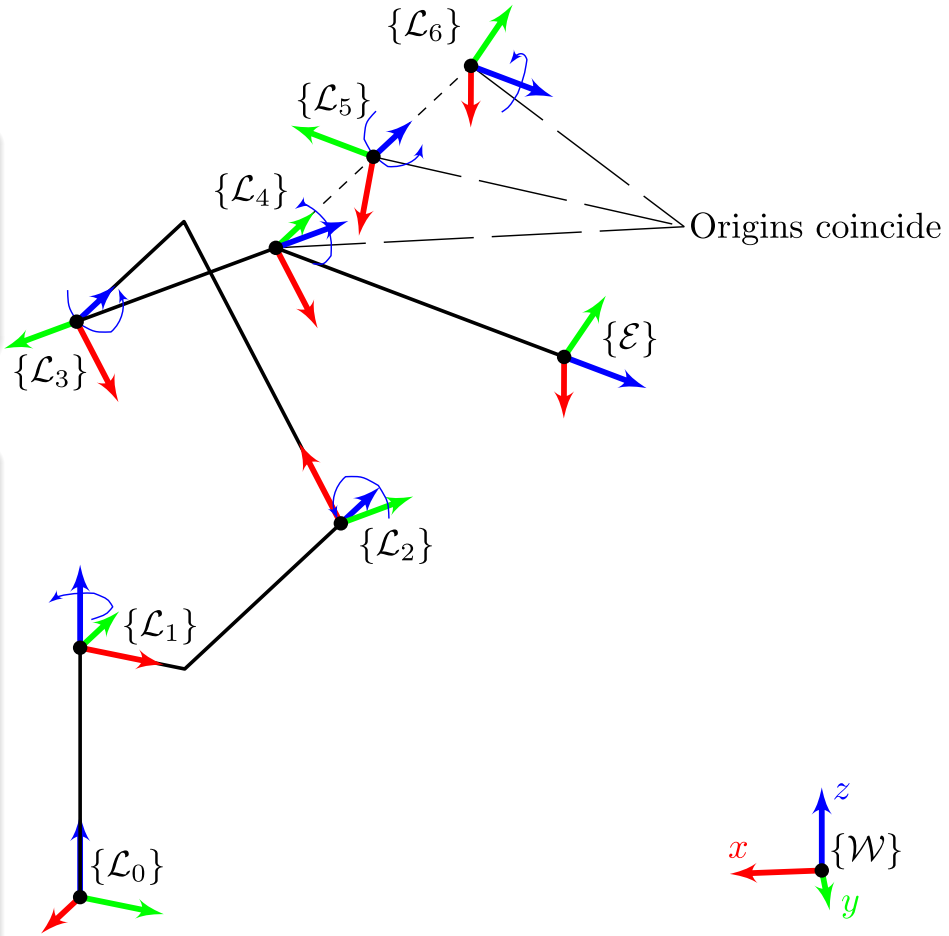
On the robot modelling capabilities of RTSS

n-DOF manipulator modelling at a glance

The geometry of the robot is defined by attaching reference frames to each body.

Body frames

- $\{\mathcal{W}\}$ is the world frame;
- $\{\mathcal{L}_0\}$ is termed base frame;
- $\{\mathcal{L}_i\}$ is the body-fixed frame attached to link i;
- $\{\mathcal{E}\}$ is the tool frame.



Body frames – TX90

On the robot modelling capabilities of RTSS

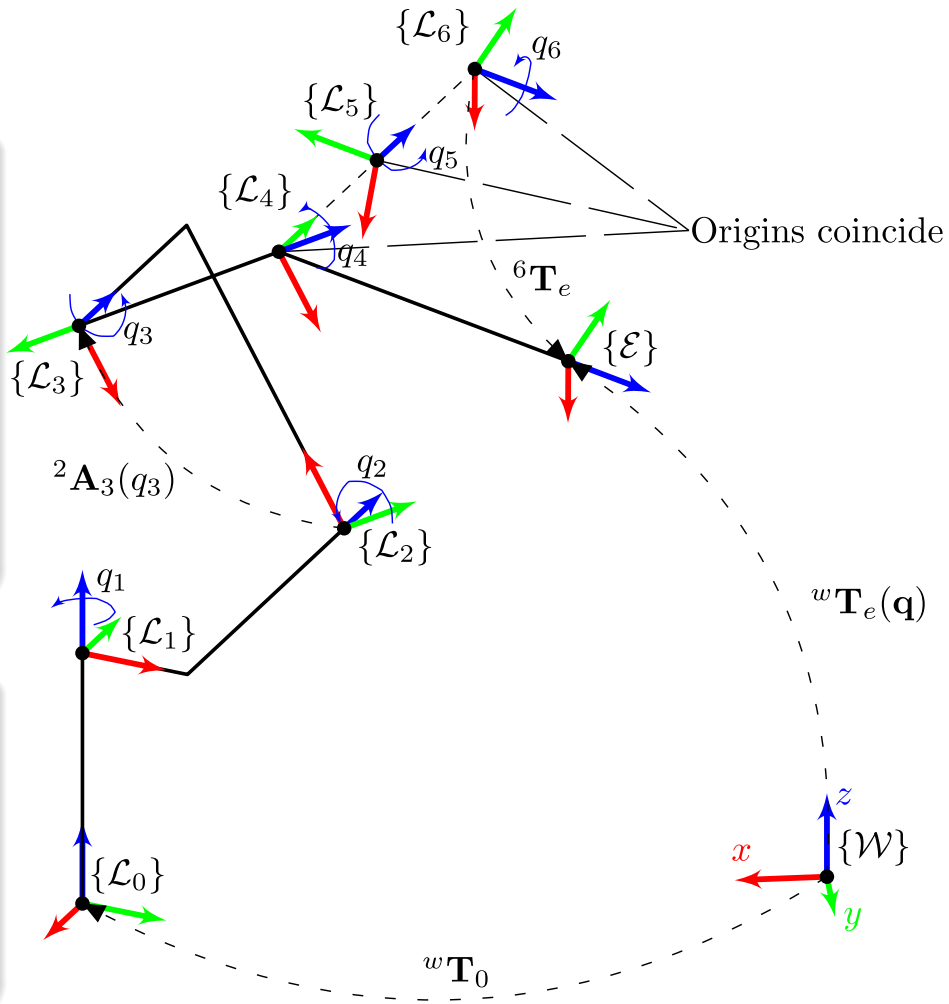
n-DOF manipulator modelling at a glance

Geometry kinematics equation

$$ {}^{w}\mathbf{T}_e(\mathbf{q}) = {}^{w}\mathbf{T}_0 \underbrace{\prod_{i=1}^{n} {}^{i-1}\mathbf{A}_i(q_i)}_{{}^{0}\mathbf{T}_n(\mathbf{q})} {}^{n}\mathbf{T}_e $$

Required model informations

- Base/tool transforms;
- Geometric relationship between consecutive links .



Coordinate transforms – TX90

On the robot modelling capabilities of RTSS

n-DOF manipulator modelling at a glance

Dynamics equations (based on Newton-Euler formulation)

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{F}_s\mathbf{sgn}(\dot{\mathbf{q}}) = \boldsymbol{\tau} - \mathbf{J}^T(\mathbf{q})\mathbf{h}_e$$

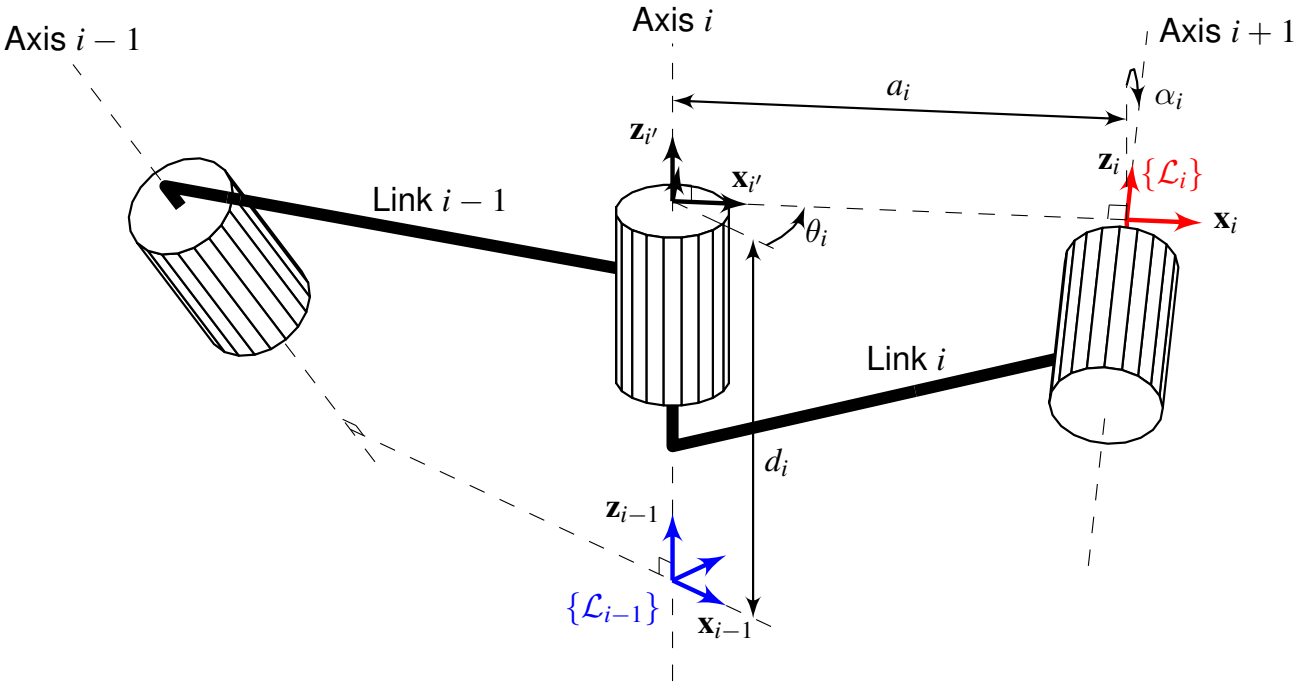
- Effective and coupling inertia torques;
- centrifugal, Coriolis and gravitational torques;
- viscous and static friction torques;
- robot-environment interaction torques.

Required model informations

- Dynamic model of each joint-link pair;
- dynamics of the robot-environment interaction (if any).

Geometric model of a joint-link pair

The standard Denavit-Hartenberg notation



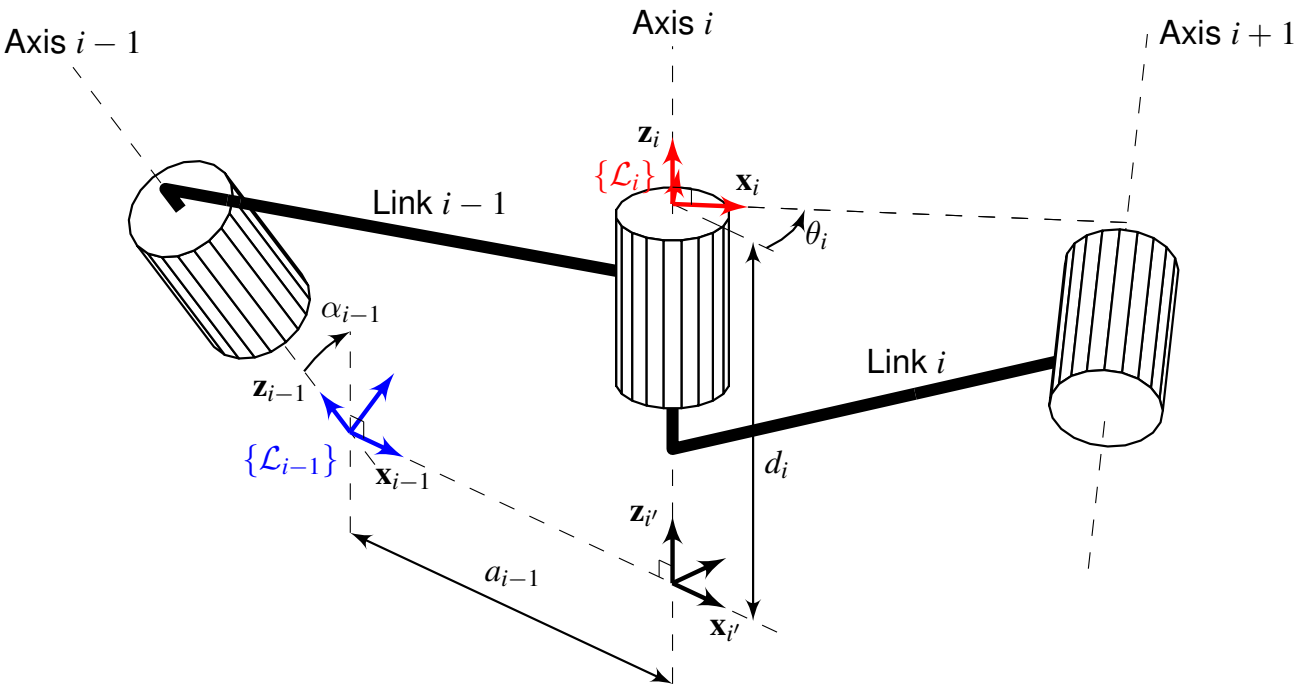
Standard DH frame assignment

Specifying $\{\mathcal{L}_i\}$ with respect to $\{\mathcal{L}_{i-1}\}$

$$^{i-1}\mathbf{A}_i(q_i) = Transl_{\hat{\mathbf{z}}}(d_i)Rot_{\hat{\mathbf{z}}}(\theta_i)Transl_{\hat{\mathbf{x}}}(a_i)Rot_{\hat{\mathbf{x}}}(\alpha_i)$$

Geometric model of a joint-link pair

The modified Denavit-Hartenberg notation



Modified DH frame assignment

Specifying $\{\mathcal{L}_i\}$ with respect to $\{\mathcal{L}_{i-1}\}$

$$^{i-1}\mathbf{A}_i(q_i) = Rot_{\hat{\mathbf{x}}}(\alpha_{i-1})Transl_{\hat{\mathbf{x}}}(a_{i-1})Rot_{\hat{\mathbf{z}}}(\theta_i)Transl_{\hat{\mathbf{z}}}(d_i)$$

Dynamic model of a joint-link pair

Inertial and actuators/transmissions parameters

Inertial parameters (10)

m	link mass
r_x, r_y, r_z	link COM
$I_{xx}, I_{yy}, I_{zz}, I_{xy}, I_{xz}, I_{yz}$	six components of the inertia tensor about the link COM

Actuator/transmission model parameters (5)

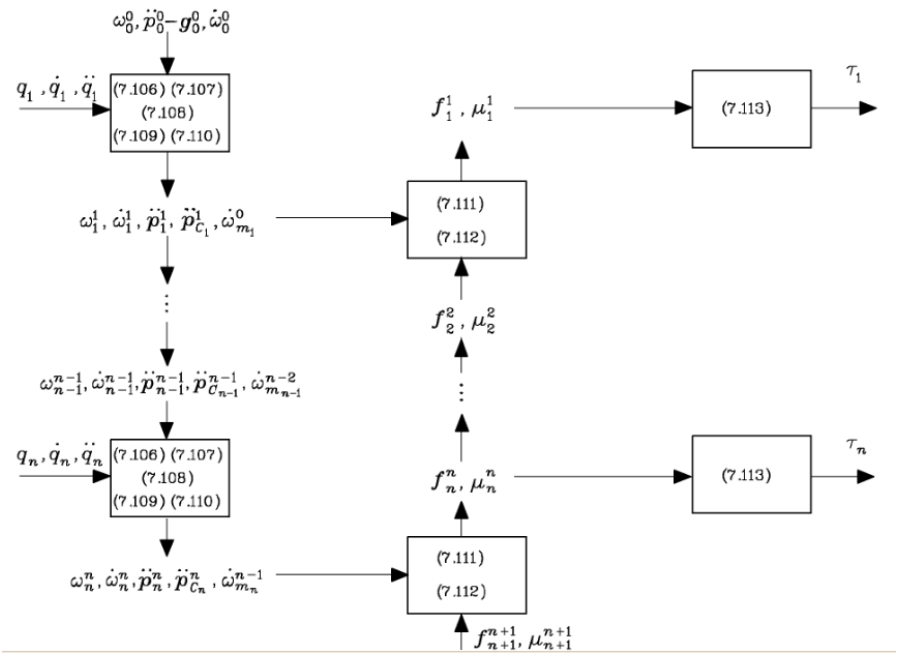
J_m	moment of inertia of the rotor
G	reduction gear ratio: joint speed/link speed
B	viscous friction coefficient
τ_C^+	Coulomb friction (positive rotation) coefficient
τ_C^-	Coulomb friction (negative rotation) coefficient

Dynamic model of a joint-link pair
Newton-Euler Formulation (notes)

Approach
Balance of forces and moments acting on each link.

Forward recursion
Velocities and accelerations.

Backward recursion
Forces and moments along the structure.



Computational structure of the Newton-Euler recursive algorithm

Modelling a direct-drive planar elbow manipulator

Pelican: experimental robot arm at CICESE (Mexico), Robotics lab.

The Pelican prototype robot arm [Kelly et al., 2005]

Geometric and kinematic properties

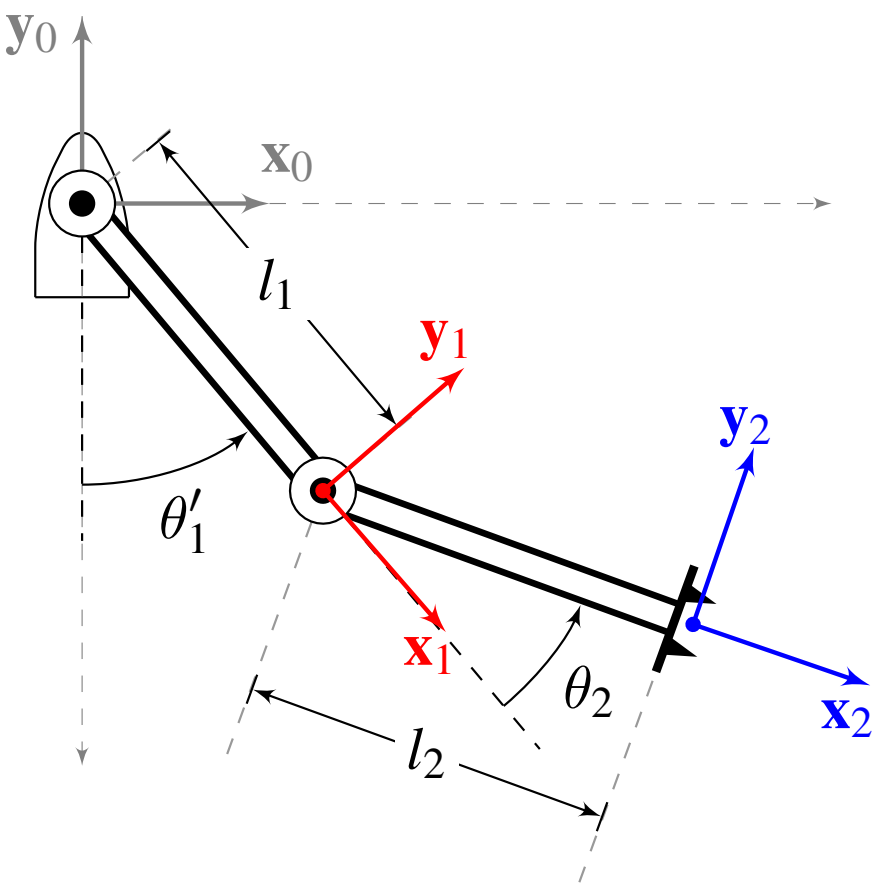
- Vertical planar manipulator;
- two rigid links connected via revolute joints.

Link	Notation	Value (m)
1	l_1	0.26
2	l_2	0.26

Link lengths of Pelican

Modelling a direct-drive planar elbow manipulator

A kinematic model based on the standard Denavit-Hartenberg notation



Standard DH frame assignment

Joint angle of axis 1 has an offset of $-\pi/2$ rad, according to DH notation.

Link	α_i	a_i	θ_i	d_i
1	0	l_1	$\theta_1'^{\star}$	0
2	0	l_2	$\theta_2^{\star}$	0

Denavit-Hartenberg table

Modelling a direct-drive planar elbow manipulator

A kinematic model based on the standard Denavit-Hartenberg notation

Link	α_i	a_i	θ_i	d_i
1	0	l_1	$\theta_1'^{\star}$	0
2	0	l_2	$\theta_2^{\star}$	0

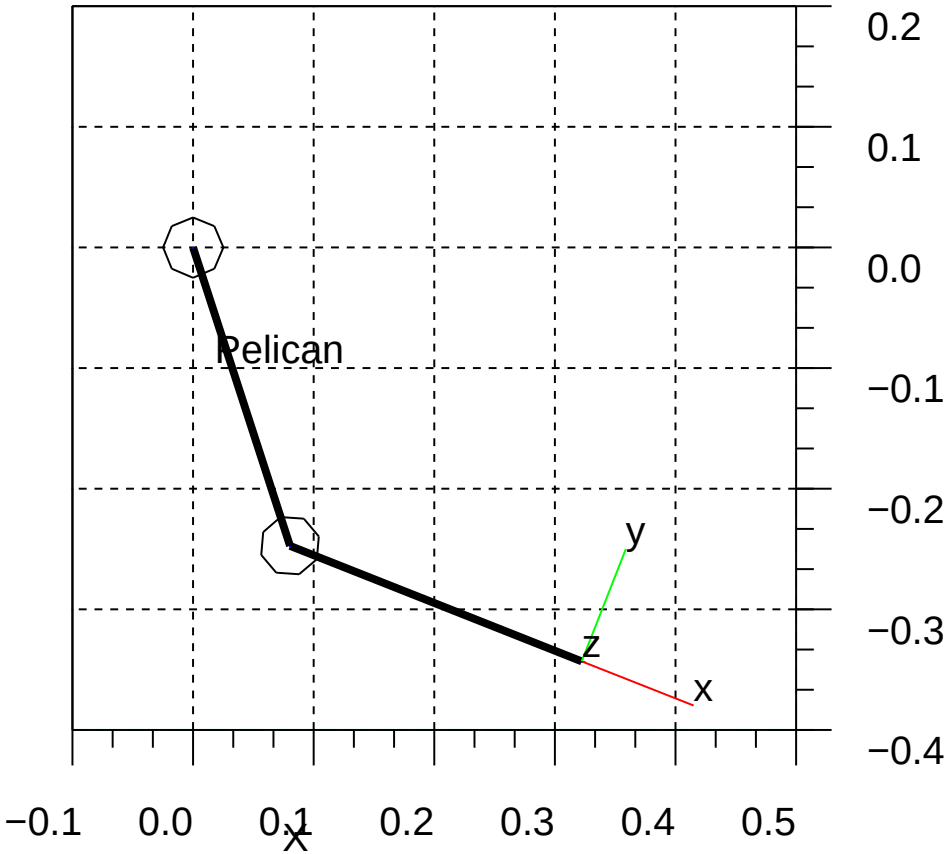
Denavit-Hartenberg table (previous slide)

Building the kinematic model

```
-->L1 = rt_link([0,l1,0,0,0], "standard");
-->L2 = rt_link([0,l2,0,0,0], "standard");
-->CL = list(L1, L2);
-->pel = rt_robot(CL, "Pelican",..
   "CICESE, Robotics Lab.",..
   "Kelly et al. 2005");
-->pel.offset = [-%pi/2; 0];
```

Modelling a direct-drive planar elbow manipulator

Model validation by simulation: comparison of results with reasonable expectations



Top view of Pelican pose at

$\mathbf{q} = [\ \pi/10 \quad 7\pi/25\ ]$ rad

Example: Where is the tool?

Pose: $\mathbf{q} = [\ \pi/10 \quad 7\pi/25\ ]$ rad

Forward kinematics

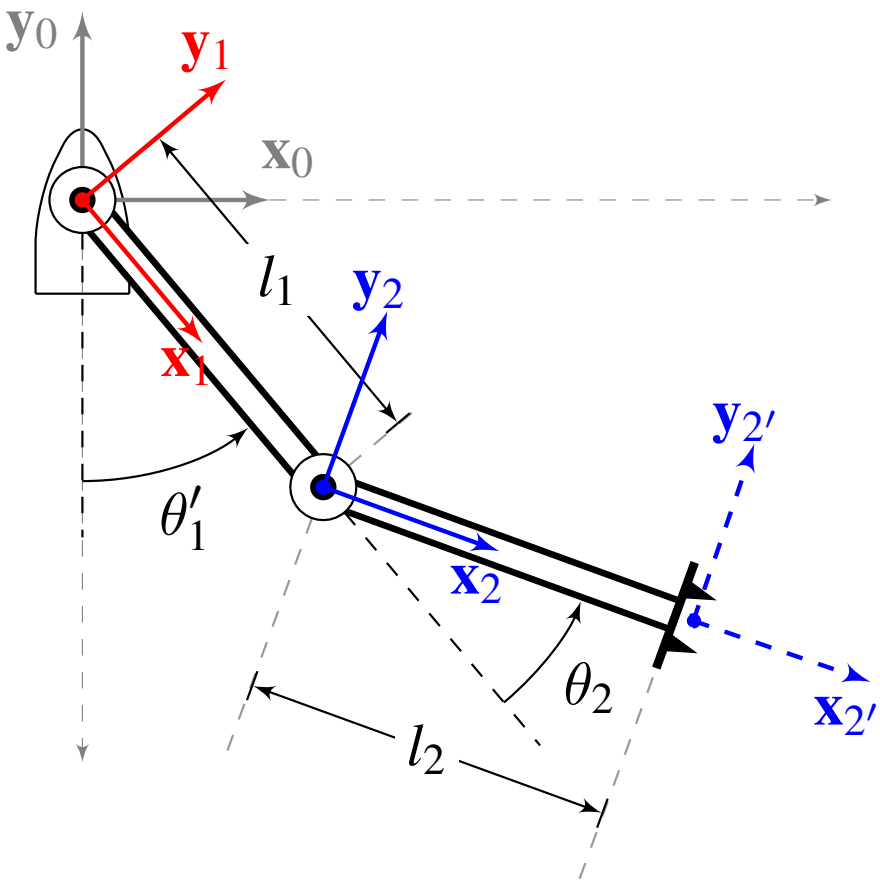
```
-->>q = [%pi/10, 7/25*%pi];
-->T02 = rt_fkine(pel, q),
 T02 =

    0.9298    0.3681    0.      0.3221
  - 0.3681    0.9298    0.    - 0.3430
    0.        0.        1.      0.
    0.        0.        0.      1.
```

Draw the Pelican pose

```
-->ws = [-0.1, 0.5, -0.4, 0.2, -1, 1];
-->rt_plot(pel, q, "workspace", ws);
```

Modelling a direct-drive planar elbow manipulator

An equivalent kinematic description using the modified Denavit-Hartenberg notation



Modified DH frame assignment

Link	α_{i-1}	a_{i-1}	θ_i	d_i
1	0	0	$\theta'^{\star}_1$	0
2	0	l_1	$\theta^{\star}_2$	0

Modified Denavit-Hartenberg table

Modelling a direct-drive planar elbow manipulator

An equivalent kinematic description using the modified Denavit-Hartenberg notation

Link	α_{i-1}	a_{i-1}	θ_i	d_i
1	0	0	$\theta_1'^{\star}$	0
2	0	l_1	$\theta_2^{\star}$	0

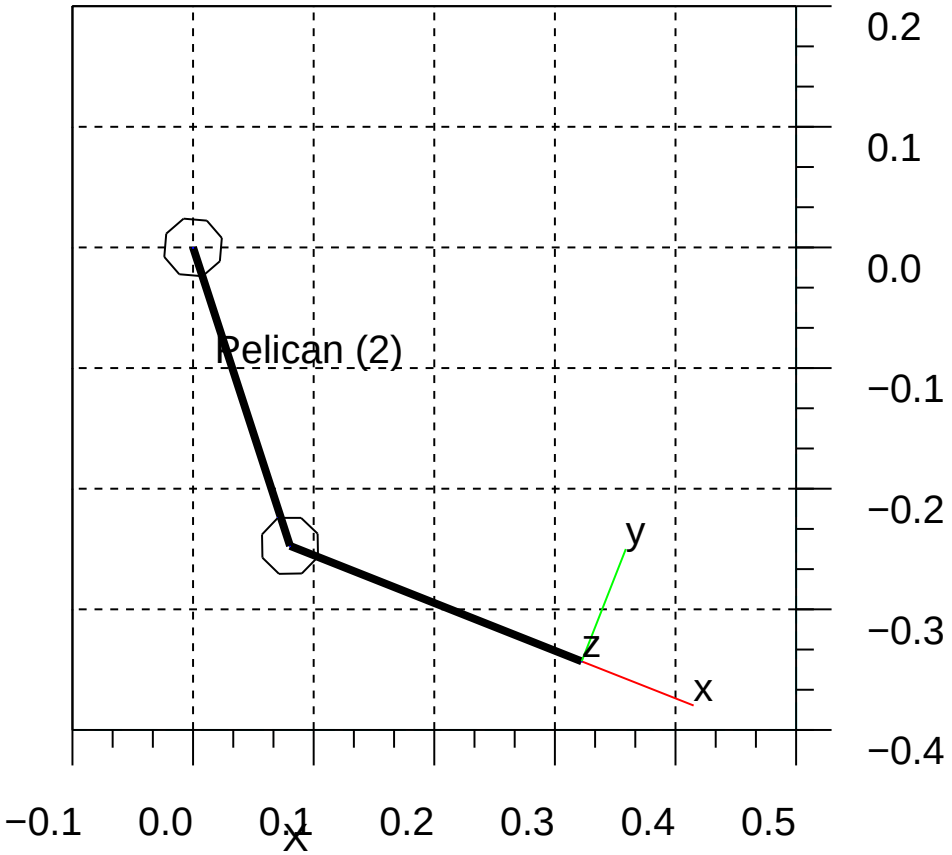
Modified Denavit-Hartenberg table (previous slide)

Building the kinematic model: the quick way

```
-->DH = [0,0,0,0;0,l1,0,0];
-->pelm = rt_robot(DH, "Pelican (2)",..
   "CICESE,Robotics Lab.", "Modified DH");
-->pelm.mdh = 1;
-->pelm.tool = rt_transl(l2,0,0);
-->pelm.offset = [-%pi/2; 0];
```

Modelling a direct-drive planar elbow manipulator

Model validation through kinematic simulation



MDH-based Pelican, at

$\mathbf{q} = [\ \pi/10 \quad 7\pi/25\ ]$ rad

Show that the two approaches are equivalent.

Forward kinematics

```
-->q = [%pi/10, 7*%pi/25];
-->T02m = rt_fkine(pelm, q),
 T02m =

    0.9298    0.3681    0.      0.3221
  - 0.3681    0.9298    0.    - 0.3430
    0.        0.        1.      0.
    0.        0.        0.      1.

-->T02 - T02m
 ans =

   0.   0.   0.   0.
   0.   0.   0.   0.
   0.   0.   0.   0.
   0.   0.   0.   0.
```

Modelling a direct-drive planar elbow manipulator

The dynamics of the Pelican arm: standard vs modified DH frame assignments

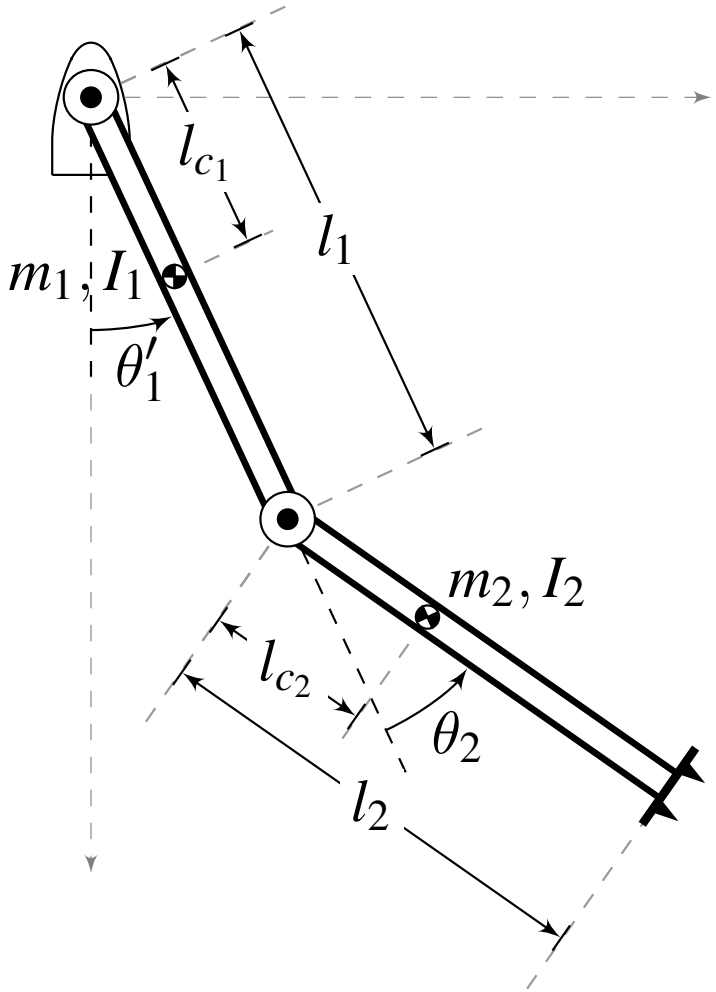


Diagram of the Pelican

Question

Which are the parameters that depend on the adopted frame assignment?

Modelling a direct-drive planar elbow manipulator

The dynamics of the Pelican arm: standard vs modified DH frame assignments

Parameter	Is it convention dependent?
Link mass	No
Inertia tensor about link COM	No (in this case of study)
Distance to link COM	Yes
Actuator/transmission	No

Dependence of physical parameters on the frame assignments

Modelling a direct-drive planar elbow manipulator

The dynamics of the Pelican arm: inertial parameters

Description	Notation	Value	Units
Mass of link 1	m_1	6.5225	kg
Mass of link 2	m_2	2.0458	kg
Inertia rel. to COM (link 1)	I_1	0.1213	kg m^2
Inertia rel. to COM (link 2)	I_2	0.0116	kg m^2

Convention-independent inertial parameters

Link	Notation	Value (m)	
		Standard DH	Modified DH
1	l_{c_1}	-0.1617	0.0983
2	l_{c_2}	-0.2371	0.0229

Distance to the link COM (convention-dependent)

Modelling a direct-drive planar elbow manipulator

The dynamics of the Pelican arm: actuator and transmission parameters

Actuators of Pelican

Two brushless DC motors located at the base and at the elbow.

Description	Motor/Joint 1		Motor/Joint 2		Units
	Sym.	Value	Sym.	Value	
Inertia (COM)	J_{m_1}	0.012	J_{m_2}	0.0025	$\mathrm{kg\ m^2}$
Gear ratio	G_1	1:1	G_2	1:1	
Viscous frict.	B_1	0.2741	B_2	0.1713	$\mathrm{Nm\ s/rad}$
Coulomb frict.	τ_{C_1}	1.29	τ_{C_2}	0.965	Nm

Actuator/transmission parameters

Modelling a direct-drive planar elbow manipulator

Simplified dynamic model based on the standard DH frame assignments

Robot model without actuators and transmissions.

Inertial parameters and gravity acceleration vector

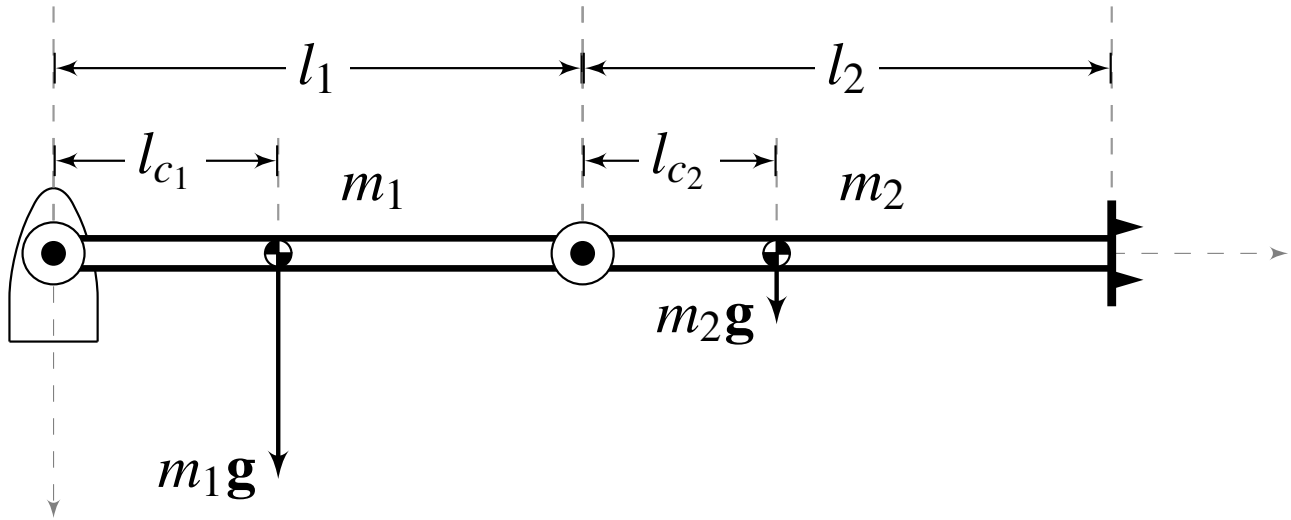
```
-->CL(1).I = [0,0,I1,0,0,0];
-->CL(2).I = [0,0,I2,0,0,0];
-->CL(1).m = m1;
-->CL(2).m = m2;
-->CL(1).r = [lc1;0;0];
-->CL(2).r = [lc2;0;0];
-->pel = rt_robot(pel, CL);
-->pel.gravity = [0;9.81;0]; // X-Y plane
```

Show 1st link data in detail

```
-->rt_showlink(pel.links(1)),
 // kinematic data displayed here...
 m     = 6.5225
!r     =    -0.1617  !
!                    !
!           0        !
!                    !
!                    !
!           0        !
!I     =   0  0  0        !
!                         !
!         0  0  0         !
!                         !
!         0  0  0.1213    !
 Jm    =
 G     =
 B     = 0
!Tc    =   0  0  !
```

Modelling a direct-drive planar elbow manipulator

Robot dynamics model validation: reasonable expectations



Stretched Pelican (in X-direction)

Gravitational torque contribution at $\mathbf{q} = [\ \pi/2 \quad 0\ ]$ ($\dot{\mathbf{q}} = \ddot{\mathbf{q}} = 0$)

$$\begin{bmatrix} \tau_{g_1} \\ \tau_{g_2} \end{bmatrix} = \begin{bmatrix} g(m_1 l_{c_1} + m_2(l_1 + l_{c_2})) \\ g m_2 (l_{c_2} + l_2) \end{bmatrix}$$

Modelling a direct-drive planar elbow manipulator

Robot dynamics model validation: simulation results

Comparison between expectations and simulation results

```
-->etau = [g*(m1*lc1 + m2*(l1 + lc2)), g*m2*lc2],
 etau =

   11.967401 0.4595869

-->etau - rt_frne(pel, [%pi/2,0], [0,0], [0,0]),
 ans =

   0. - 5.551D-17
```

Modelling a direct-drive planar elbow manipulator

Complete dynamic model based on the standard DH frame assignments

Robot model including the dynamics of actuators and transmissions.

Modelling motor parameters

```
-->CL(1).Jm = Jm1;
-->CL(2).Jm = Jm2;
-->CL(1).G = G1;
-->CL(2).G = G2;
-->CL(1).B = B1;
-->CL(2).B = B2;
-->CL(1).Tc = Tc1*[1,1];
-->CL(2).Tc = Tc2*[1,1];
```

Launch the movie!

Modelling a 6-DOF industrial robot (notes)

The industrial robot Siemens Manutec r3

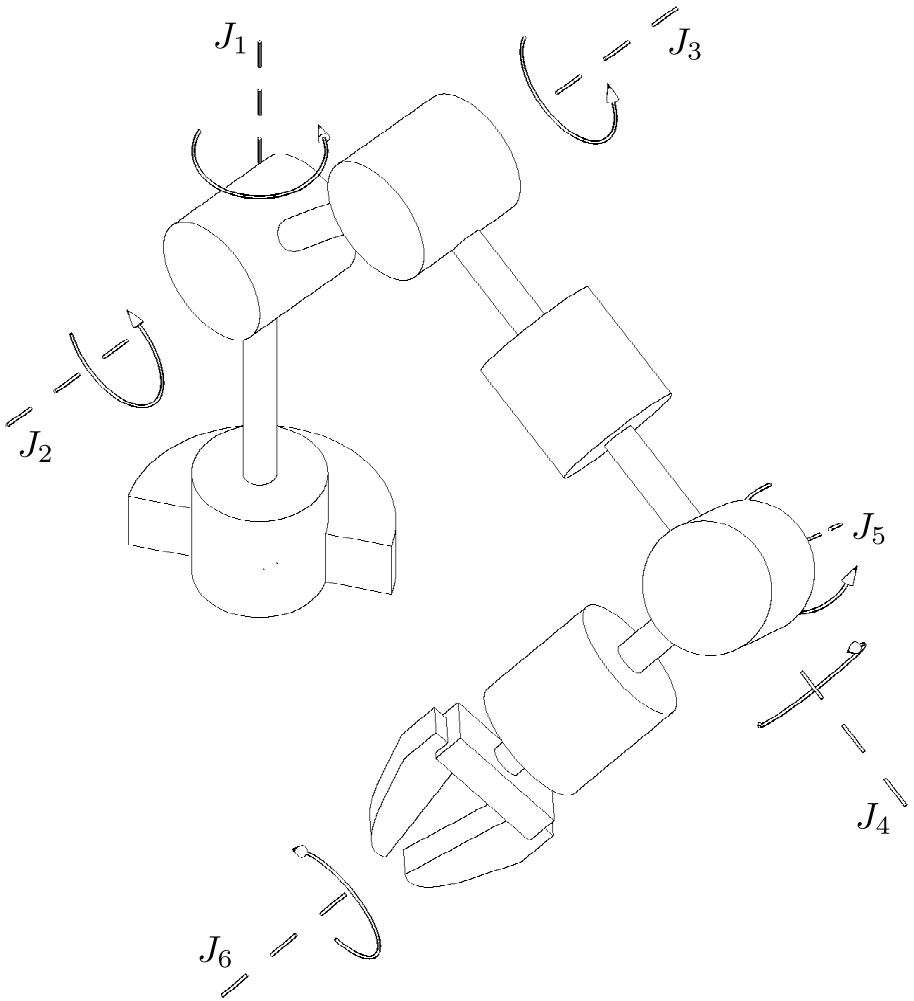
Robot description at a glance

- arms mechanically very stiff;
- six rotational joints;
- current-controlled DC motors;
- each motor embedded in the preceeding arm;
- encoders on the motors' axes;
- friction in the gears;
- gear ratios of the base axes fairly low;
- payload of 15 kg.

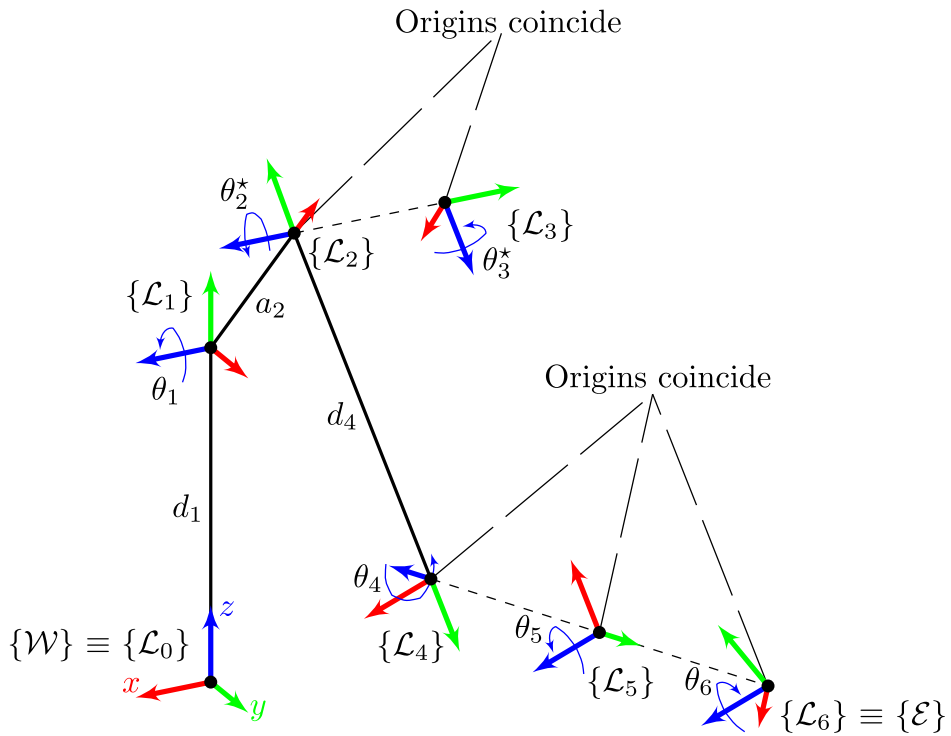
The robot Manutec r3 – [Winkler and Suchý, 2005]

Modelling a 6-DOF industrial robot (notes)

A geometric model based on the standard Denavit-Hartenberg notation



Kinematic chain



Body frames

Modelling a 6-DOF industrial robot (notes)

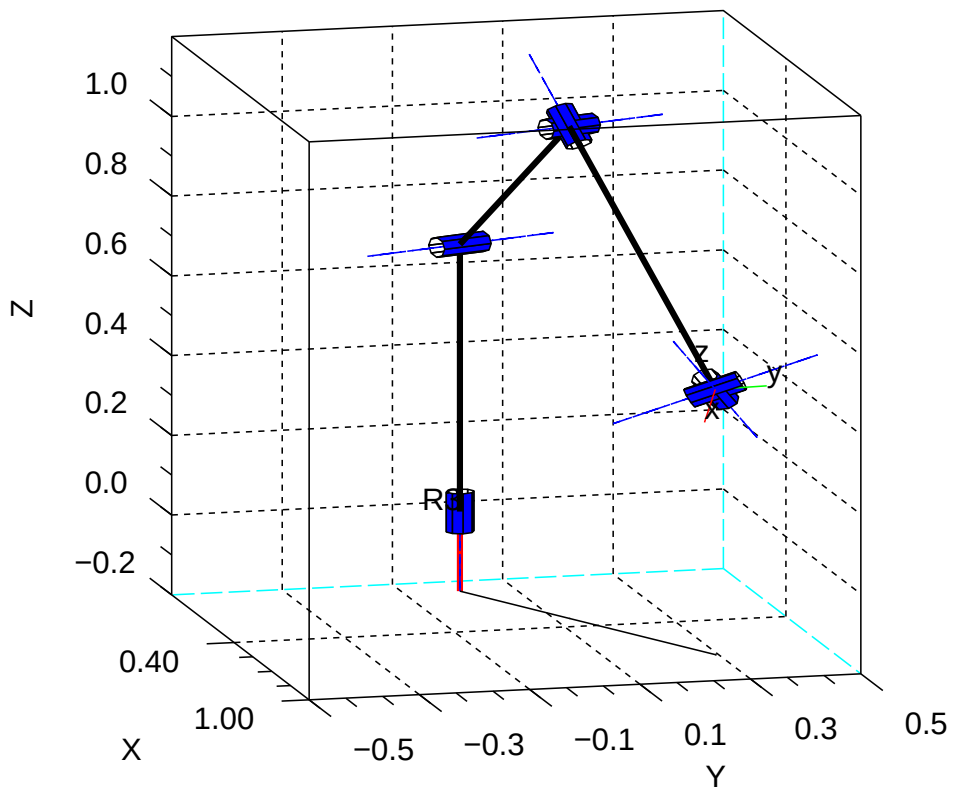
A geometric model based on the standard Denavit-Hartenberg notation

Link	α_i	a_i	θ_i	d_i
1	$\pi/2$	0	θ_1	d_1
2	0	a_2	$\theta_2^\star$	0
3	$-\pi/2$	0	$\theta_3^\star$	0
4	$\pi/2$	0	θ_4	d_4
5	$-\pi/2$	0	θ_5	0
6	0	0	θ_6	0

Denavit-Hartenberg table[a]

[a]Offsets to match the rest config. (the robot is stretched-up at $\mathbf{q} = \mathbf{0}$):

$$\theta_2^\star = \theta_2 + \pi/2, \quad \theta_3^\star = \theta_3 - \pi/2$$



Visualization of the Manutec r3 robot at a given $\mathbf{q}$

Modelling a 6-DOF industrial robot (notes)

The dynamics of the Manutec r3 robot: actuator and transmission parameters

Description	Notation	Values			Units
		Arm 1	Arm 2	Arm 3	
Inertia (COM)	J_{m_i}	0.0013	0.0013	0.0013	kg m^2
Gear ratio	G_i	-105	210	60	
Viscous frict. [1]	B_i	$8.125 \cdot 10^{-4}$	$7.692 \cdot 10^{-4}$	$1.5385 \cdot 10^{-3}$	Nm s/rad
Coulomb frict.	τ_{C_i}	0.4	0.5	0.7	Nm

Description	Notation	Values			Units
		Arm 4	Arm 5	Arm 6	
Inertia (COM)	J_{m_i}	$1.6 \cdot 10^{-4}$	$1.8 \cdot 10^{-4}$	$4.3 \cdot 10^{-5}$	kg m^2
Gear ratio	G_i	-99	79.2	-99	
Viscous frict. [1]	B_i	$71.2 \cdot 10^{-6}$	$82.6 \cdot 10^{-6}$	$36.7 \cdot 10^{-6}$	Nm s/rad
Coulomb frict.	τ_{C_i}	0.22	0.38	0.11	Nm

Actuator/transmission parameters, base and wrist axes

[1] These parameters have been given reasonable values

Outline

1. Robotic manipulators modelling with ScicosLab
   - Rigid body transformations
   - Building serial-link manipulator models
   - Analysis of serial-link manipulators
2. Robot control systems design using Scicos
   - Basic concepts
   - Motion control
   - Further applications
3. Robot control code generation for use with Linux RTAI
   - Developing RTAI-based centralized controllers with RTAI-Lab

Analysis of a 6-DOF industrial robot

Positioning/orientation of the Manutec r3 robot

Forward kinematics

```
// robot configuration
q = [0.377, -0.754, -1.711,..
         0.754, 2.011, -0.440];

// pose of the EE
twe = rt_fkine(r3, q),
 twe =
     0.680    0.572    0.458    0.743
   - 0.348    0.802  - 0.485    0.294
   - 0.645    0.170    0.745    0.465
     0.       0.       0.       1.
```

Inverse kinematics

```
// solve the inverse kinematics (IKP)
q0 = qz; q0(2:3) = [-1, -1];
modulo(rt_ikine(r3, twe, q0), 2*%pi),
 ans =
      0.377  - 0.754  - 1.711
        0.754    2.011   - 0.440
```

Analysis of a 6-DOF industrial robot

Kinetostatics of the Manutec r3 robot

Differential kinematics

The Jacobian matrix maps velocity between joint and Cartesian space.

$$\mathbf{v}_e = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}$$

Jacobian analysis

```
// manipulator's Jacobian matrix in base coordinates, and its determinant
j0 = rt_jacob0(r3, q),
 j0 =
   - 0.294    0.190    0.529    0.       0.       0.
     0.743    0.075    0.210    0.       0.       0.
     0.000    0.799    0.457    0.       0.       0.
     0.000    0.368    0.368    0.582  - 0.228    0.458
     0.000  - 0.930  - 0.930    0.231  - 0.874  - 0.485
     1.       0.000    0.000  - 0.780  - 0.429    0.745

det(j0), // j0 is not singular
 ans =
  - 0.261
```

Analysis of a 6-DOF industrial robot

Kinetostatics of the Manutec r3 robot

Those configurations at which $\mathbf{J}(\mathbf{q})$ is rank-deficient are termed kinematic singularities.

Problems at (near) a singularity

- The mobility of the robot is reduced;
- infinitely many solutions to the IKP may exist;
- small velocities at the EE cause large joint velocities.

Classification of singularities

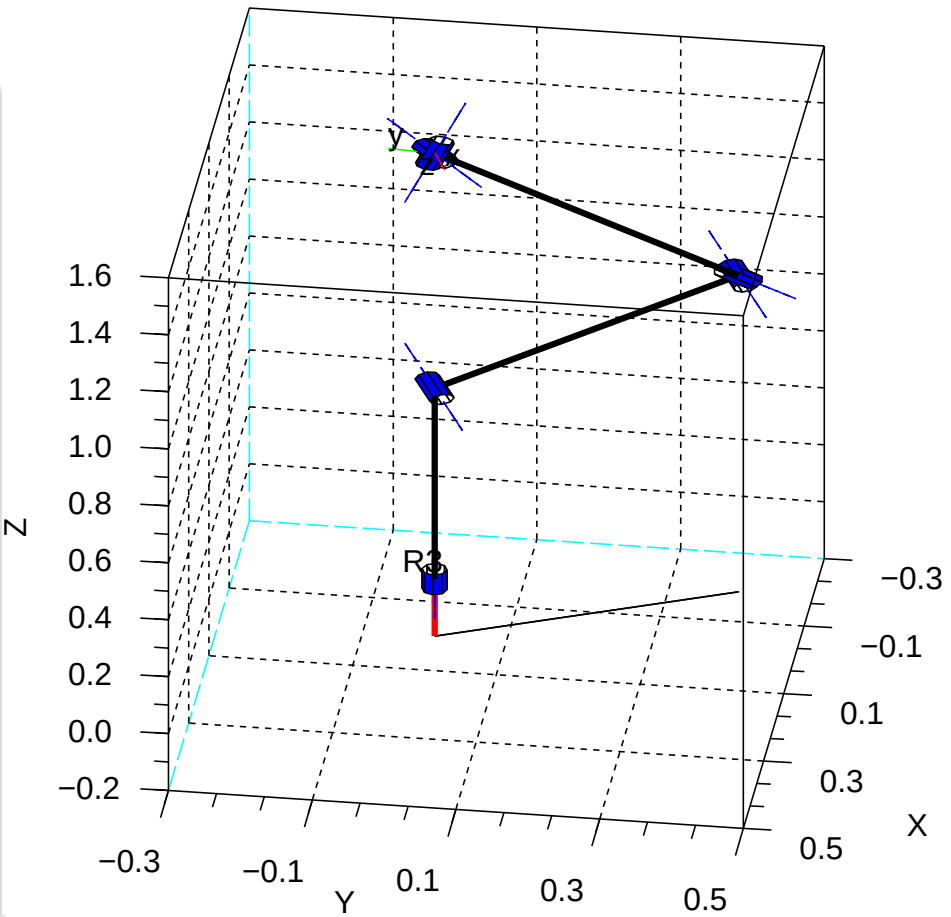
- Boundary, the robot is stretched/retracted;
- internal, occur inside the reachable workspace.

Analysis of a 6-DOF industrial robot

Kinetostatics of the Manutec r3 robot

r3's shoulder singularity

```
// singular robot configuration
qs = [2.011, -1.068, 1.711,..
      0.251, 2.073, -1.257];

// manipulator's Jacobian matrix
js = rt_jacob0(r3, qs);

// singular values
svd(js).',
 ans =

   1.949  1.571  0.744
          0.529  0.342  0.000

// large joint velocities
dqs = inv(js)*[0.1; 0; 0; 0; 0; 0]; dqs.',
ans =

  - 219.6  0.052  0.000
           105.6  32.66 - 145.6
```



The Manutec r3 at shoulder singularity

Analysis of a 6-DOF industrial robot

Kinetostatics of the Manutec r3 robot

Statics: the manipulator is at an equilibrium configuration

The Jacobian transpose matrix maps the force between Cartesian and joint space.

$$\boldsymbol{\tau} = \mathbf{J}^T(\mathbf{q})\mathbf{w}_e$$

Torques acting at the actuators

```
// manipulator's Jacobian matrix (wrt EE)
jn = rt_jacobn(r3, q);

// forces acting on the EE (wrt EE)
wn = [-2.887; 2.56; -4.998;..
          -1.697; 1.654; 1.284];

// joint torques to be exerted to keep
// the robot in static equilibrium
tn = jn'*wn; tn',
 ans =
   7.228  - 2.323 - 2.044
         - 1.299 - 2.219 1.284
```

The Jacobian matrix characterize completely the kinetostatics of the manipulator.

Trajectory generation and visualization (notes)

A a number of toolbox functions can operate on trajectories

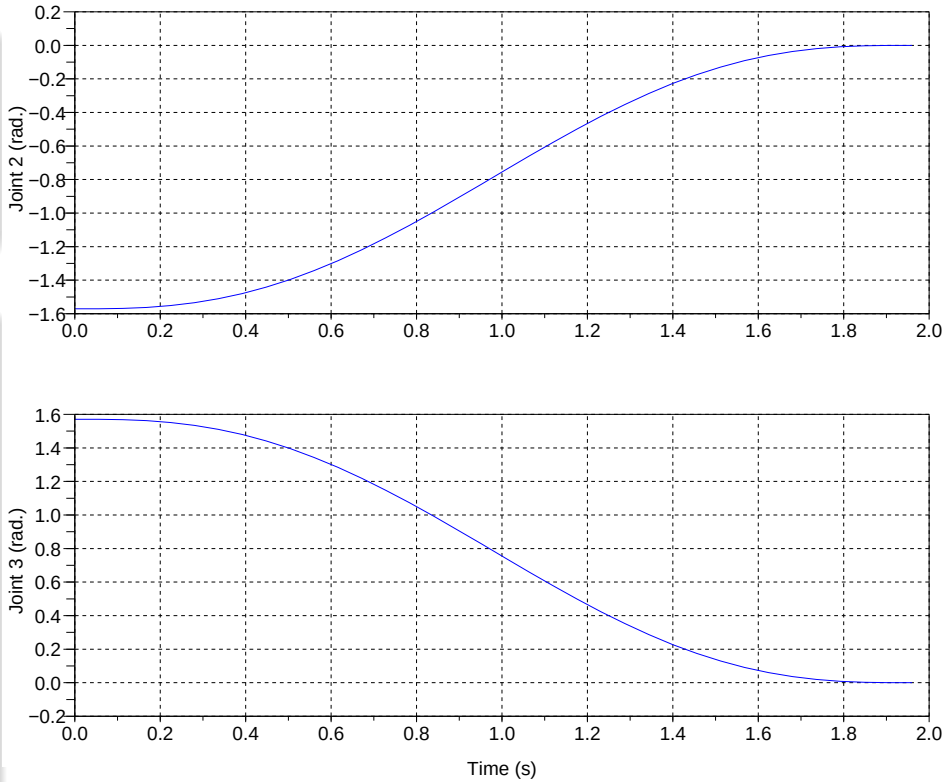
Trajectories, Animation

demos/rt_rttgdemo.sce

demos/rt_rtandemo.sce

Joint space trajectory

```
// initial configuration
qi = [0, -%pi/2, %pi/2, 0, 0, 0];

// final configuration
qf = [0, 0, 0, 0, 0, 0];

// travelling time
t = [0:.056:2]';

// 5th order interpolating polynomial
[q, qd, qdd] = rt_jtraj(qi, qf, t);
```



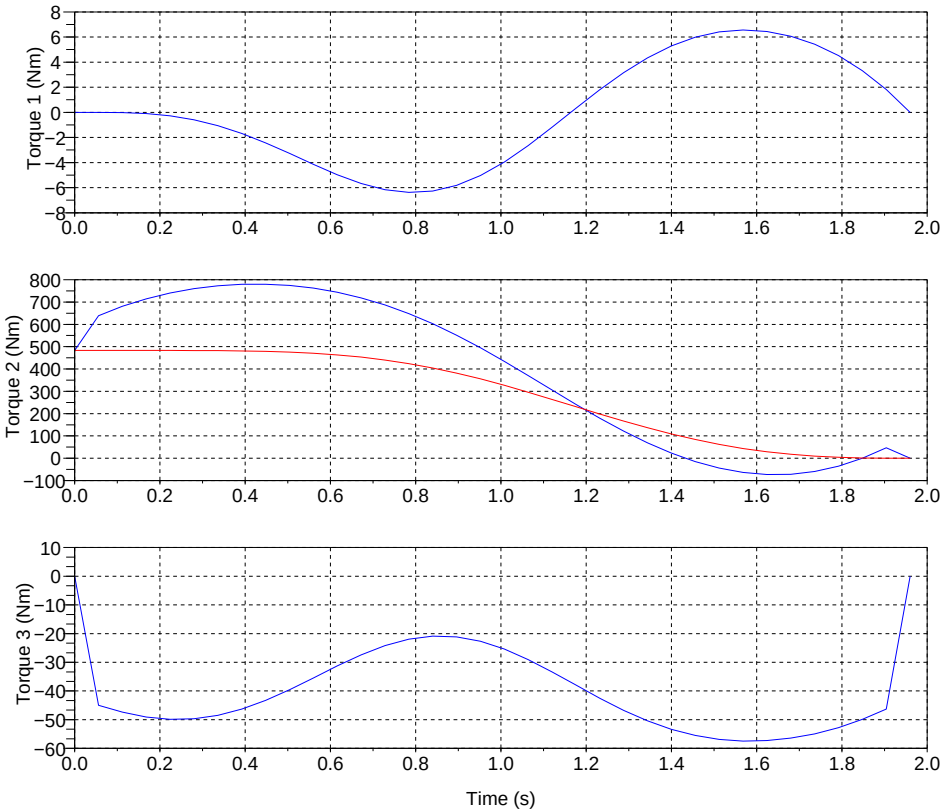
Joint space trajectory

Analysis of a 6-DOF industrial robot

The inverse dynamics problem for the Manutec r3 robot

Joint-torques analysis

```
// joint torques computation
tau = rt_frne(r3, q, qd, qdd);

// contribution of gravitational torques
taug = rt_gravload(r3, q);
```



Joint torques for the given joint space trajectory (grav. torque in red)

Analysis of a 6-DOF industrial robot

Study of joint inertia of Manutec r3 robot

The gear ratios of the base axes are fairly low, so the actuator inertia do not dominate the total joint inertia.

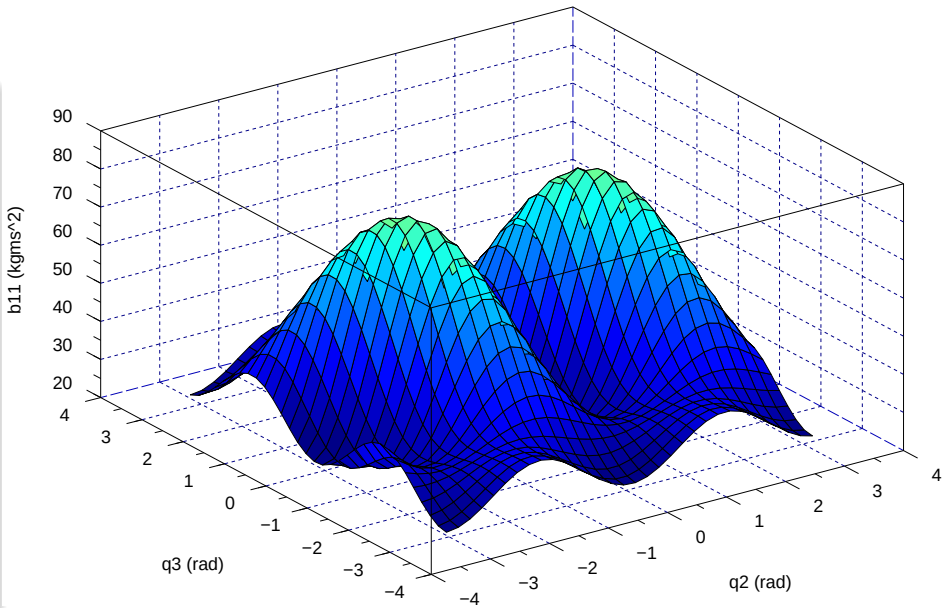
This results in significant variations of joint inertia, most important in joint 1, where the total inertia for the horizontal, loaded arm is more than three times higher than for the vertical arm.

Analysis of a 6-DOF industrial robot

Study of joint inertia of Manutec r3 robot

Inertia of joint 1

```
// manipulator's pose definition as a
// function of joint angles q2 and q3
[q2, q3] = meshgrid(-%pi:0.2:%pi);
[r, c] = size(q2);
q=[zeros(r*c,1) q2(:) q3(:) zeros(r*c,3)];

// compute inertia matrix & plot results
b = rt_inertia(r3, q); b11 = b(1,1,:);
b11 = matrix(b11,r,c); surf(q2, q3, b11);

// factor of variation over the path
bM = max(b11); bm = min(b11); bM/bm,
 ans =
   3.5667326
```



Inertia seen by the joint 1 of the r3 as a function of joint angles q_2 and q_3

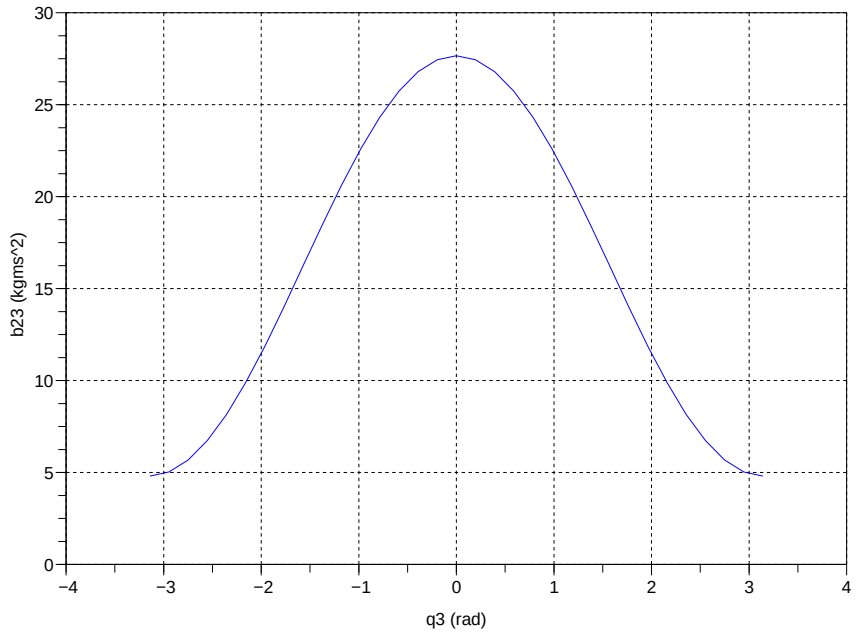
Analysis of a 6-DOF industrial robot

Study of joint inertia of Manutec r3 robot

Furthermore, the inertial coupling effects, in particular between joint 2 and 3, cannot be neglected.

Inertial coupling between joints 2, 3

```
// manipulator's pose definition as a
// function of joint angle q3
q3 = -%pi:%pi/16:%pi;
[r, c] = size(q3);
q = [zeros(r*c,2) q3(:) zeros(r*c,3)];

// compute inertia matrix & plot results
b = rt_inertia(r3, q); b23=b(2,3,:);
b23=matrix(b23,r,c); plot(q3, b23);
```



Coupling effects between joint 2 and 3, while joint 3 is free

Analysis of a 6-DOF industrial robot

Study of joint inertia of Manutec r3 robot

In short, the properties of this particular six-axis manipulator are a cross between direct drive robots and high-geared, well-balanced types that perform well enough with ordinary, robust, independent joint control.

This makes it a good platform for experimental evaluation of sophisticated model-based control algorithms.

Inner beauties of RNE (notes)

Implementation details of robot dynamics

Dynamics equation (simplified)

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{G}(\mathbf{q}) = \boldsymbol{\tau}$$

Equivalent code in RTSS

```
tau = rt_frne(robot, q, Dq, DDq);
```

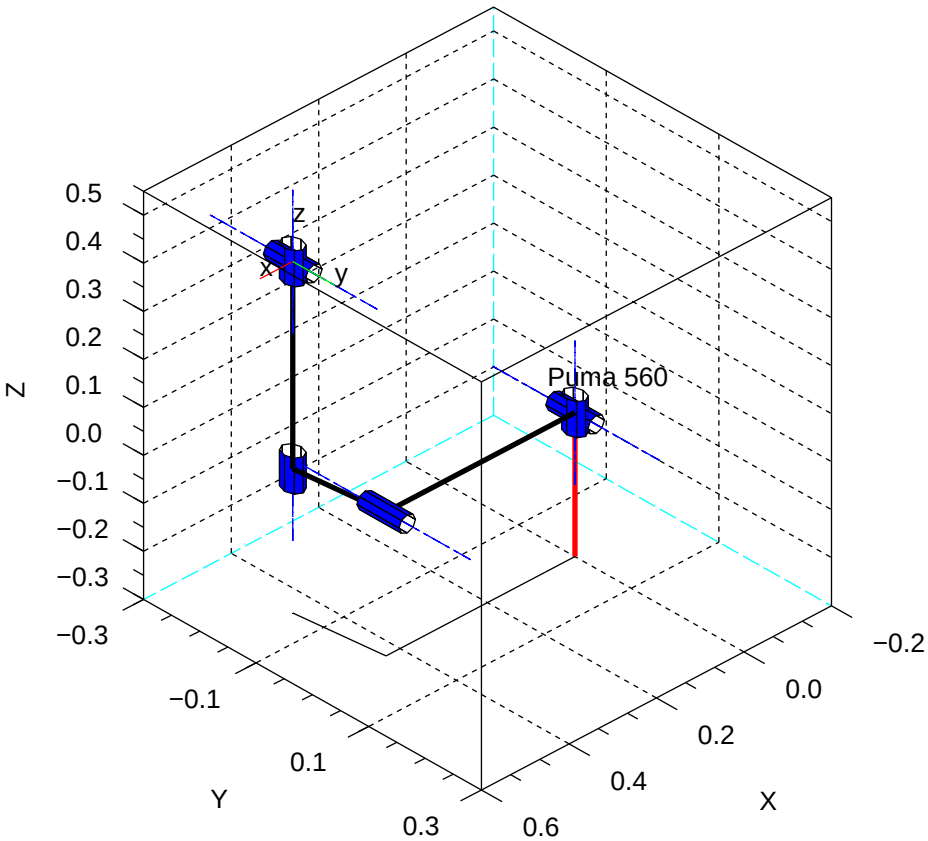
Question

How to use the (recursive) balance of all the forces acting on each link to compute:

- $\mathbf{G}(\mathbf{q})$;
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}$;
- $\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}}$;
- $\mathbf{B}(\mathbf{q})$;
- $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ (argh!).

Ready-to-use manipulator models provided by RTSS

The Unimation Puma 560 arm



Standard DH-based Puma 560 at its zero angle pose

The Unimation Puma 560

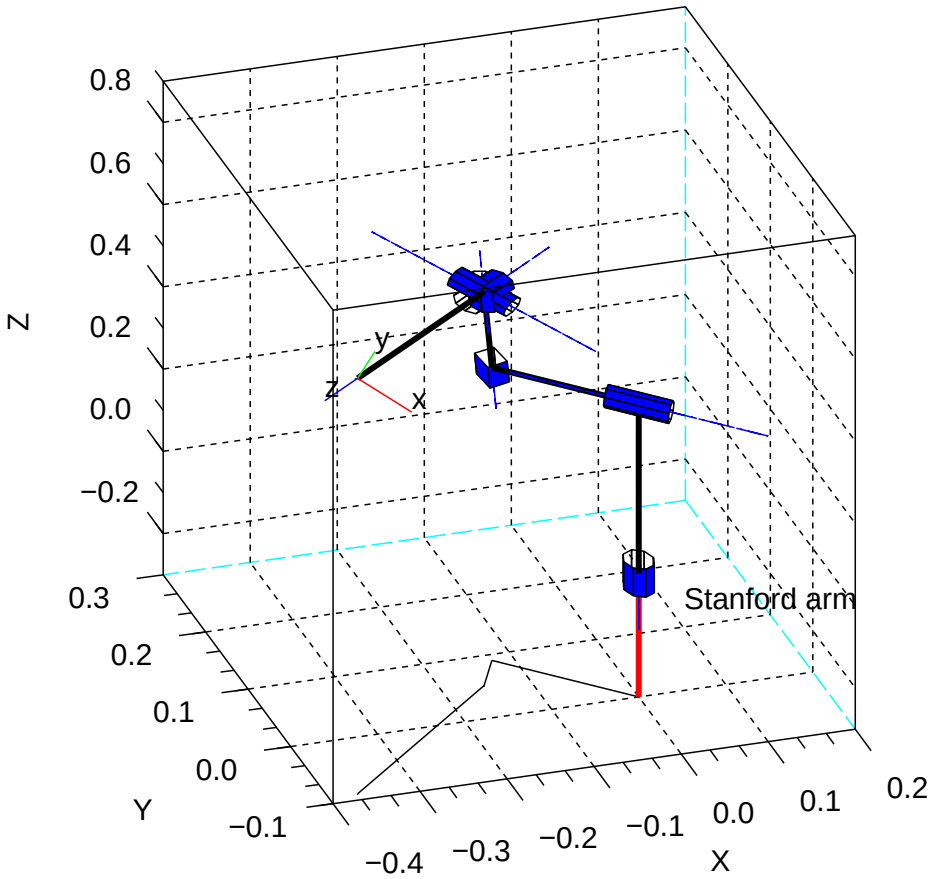
- Kinematic and dynamic data;
- Standard and modified DH-based models;
- quantities in standard SI units.

Create a Puma 560 robot

```
// standard DH-based robot object (p560)
exec <RTSS-DIR>/models/rt_puma560.sce;

// modified DH-based robot object (p560m)
exec <RTSS-DIR>/models/rt_puma560akb.sce;
```

Ready-to-use manipulator models provided by RTSS

The Stanford manipulator



Standard DH-based Stanford arm at a generic pose

The Stanford manipulator

- Kinematic and dynamic data;
- Standard DH-based model;
- quantities in standard SI units.

Create a Stanford manipulator

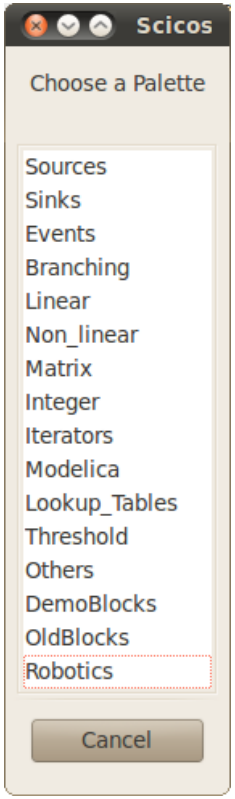
```
// standard DH-based robot object (stanf)
exec <RTSS-DIR>/models/rt_stanford.sce;
```

Outline

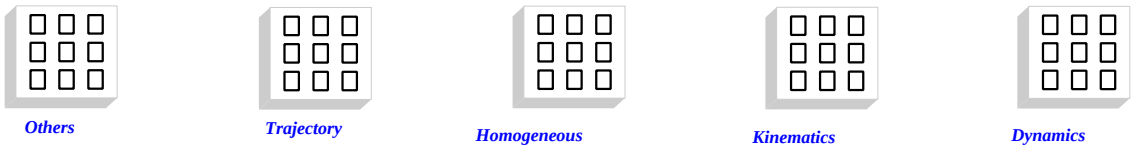
1. Robotic manipulators modelling with ScicosLab
   - Rigid body transformations
   - Building serial-link manipulator models
   - Analysis of serial-link manipulators
2. Robot control systems design using Scicos
   - Basic concepts
   - Motion control
   - Further applications
3. Robot control code generation for use with Linux RTAI
   - Developing RTAI-based centralized controllers with RTAI-Lab

The Robotics palette

A library of ready-to-use blocks for robotics simulations



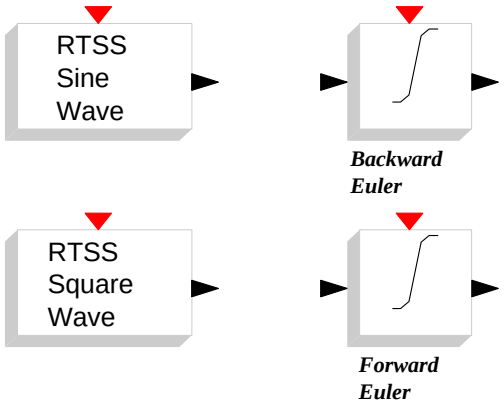
List of the available palettes after the installation of RTSS



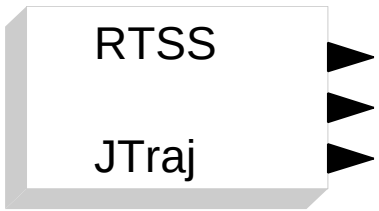
Blocks' categories available with RTSS-1.0.0b1

The Robotics palette

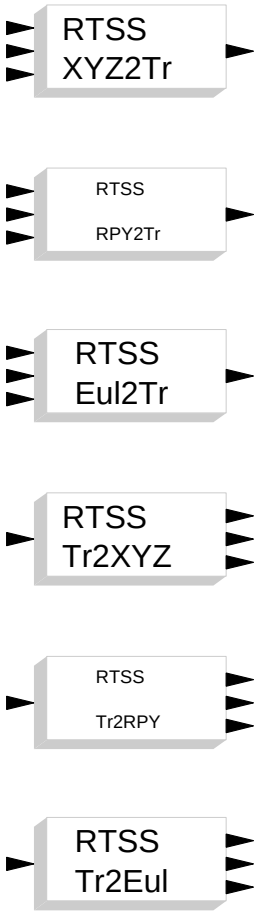
A library of ready-to-use blocks for robotics simulations



Category "Others"



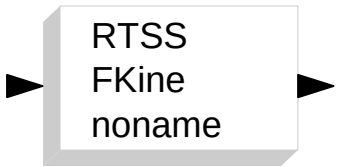
Category "Trajectory"

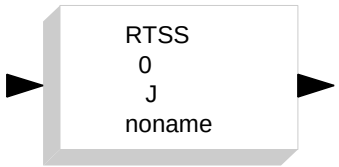


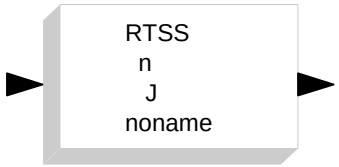
Category "Homogeneous"

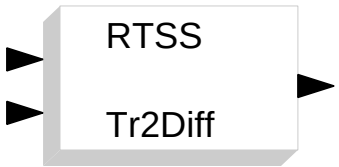
The Robotics palette

A library of ready-to-use blocks for robotics simulations

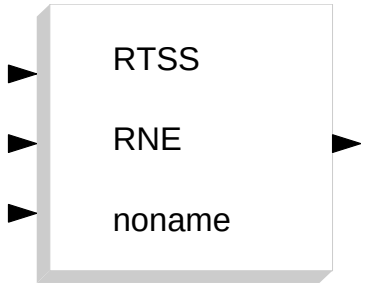


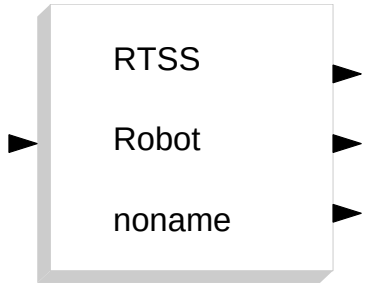


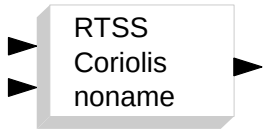


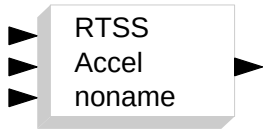


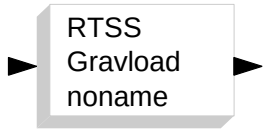
Category “Kinematics”

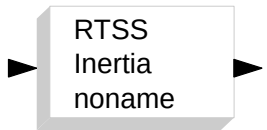










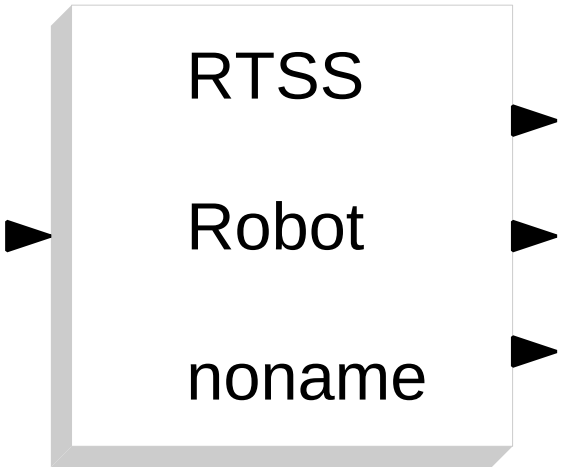


Category “Dynamics”

Proper setting of model-based blocks

How to set the robot model to be simulated in a block operating on robot objects

Blocks in `Kinematics` and `Dynamics` need a model to work.



A block which operates on a robot model: the Forward Dynamics Block (FDB)

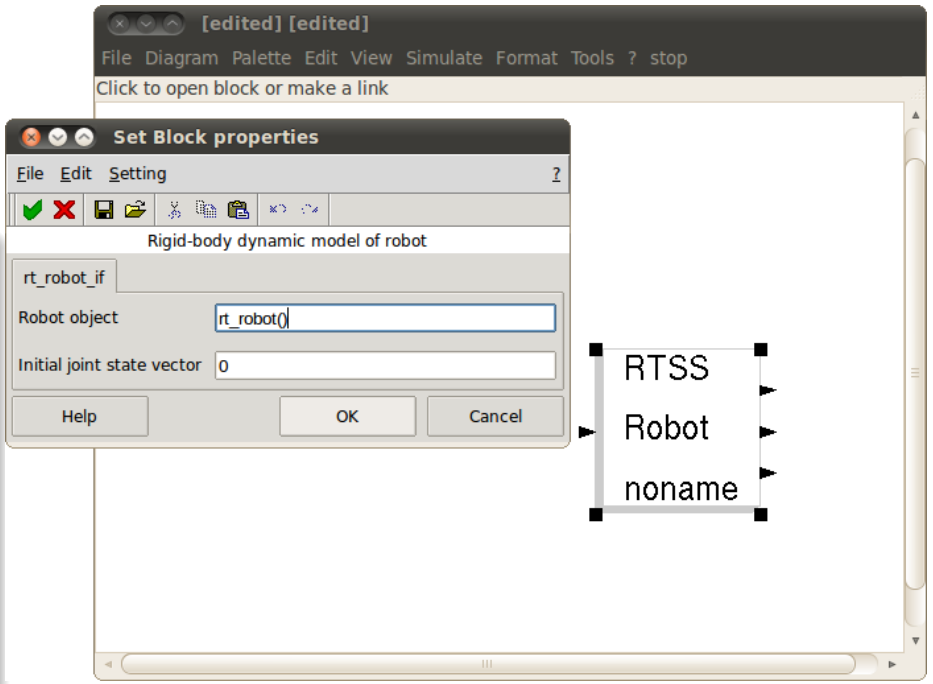
- The user must specify the robot model to be simulated as block parameter;
- the robot model must be a symbolic parameter defined in the context of the diagram.

Proper setting of model-based blocks

How to set the robot model to be simulated in a block operating on robot objects

r3 robot model defined in the context of the diagram

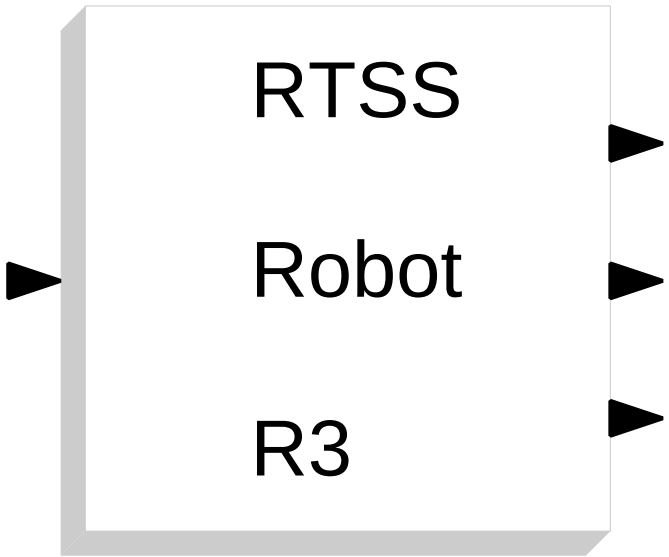
```
// robot model definition
// (pseudocode)
r3 = rt_robot( ... );

// Other symbolic parameters
 ...
```



FDB's block properties dialog – r3 must be entered in the field "Robot object"

Proper setting of model-based blocks

How to set the robot model to be simulated in a block operating on robot objects



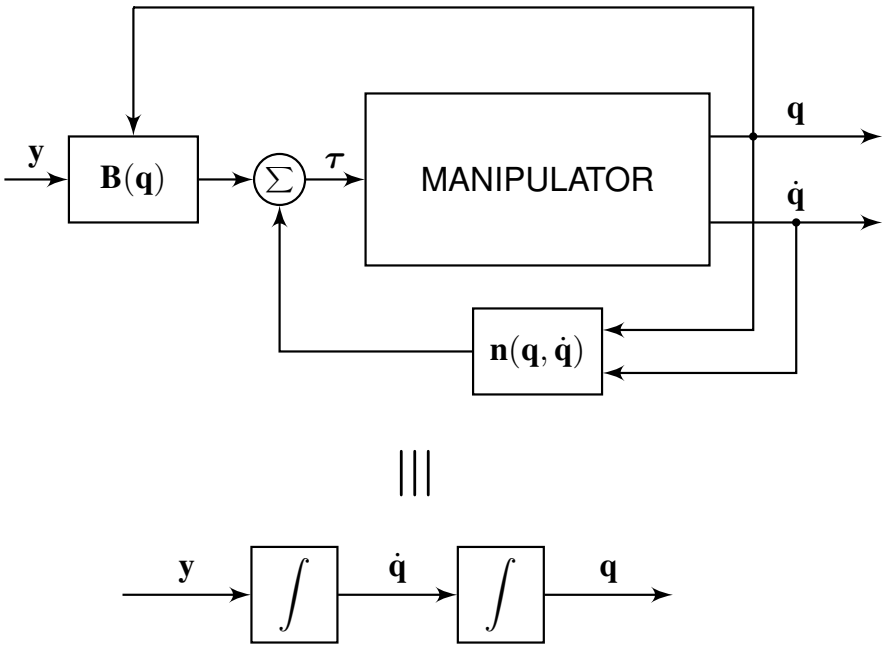
FDB ready to simulate the forward dynamics of the Manutec r3

Outline

1. Robotic manipulators modelling with ScicosLab
   - Rigid body transformations
   - Building serial-link manipulator models
   - Analysis of serial-link manipulators
2. Robot control systems design using Scicos
   - Basic concepts
   - Motion control
   - Further applications
3. Robot control code generation for use with Linux RTAI
   - Developing RTAI-based centralized controllers with RTAI-Lab

Model-based centralized controller for tracking

The inverse dynamics control: short review of the mathematics involved



Exact lineatization performed by inverse dynamics control

The dynamic model of a robot arm can be rewritten as

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}}) = \boldsymbol{\tau}$$

Inverse dynamics control law

$$\boldsymbol{\tau} = \mathbf{B}(\mathbf{q})\mathbf{y} + \mathbf{n}(\mathbf{q}, \dot{\mathbf{q}})$$

which leads the system to

$$\ddot{\mathbf{q}} = \mathbf{y}$$

Model-based centralized controller for tracking

The inverse dynamics control: short review of the mathematics involved

Typical choice for $\mathbf{y}$ is

$$\mathbf{y} = \ddot{\mathbf{q}}_d + \mathbf{K}_v \dot{\tilde{\mathbf{q}}} + \mathbf{K}_p \tilde{\mathbf{q}}$$

leading to the error dynamics equation

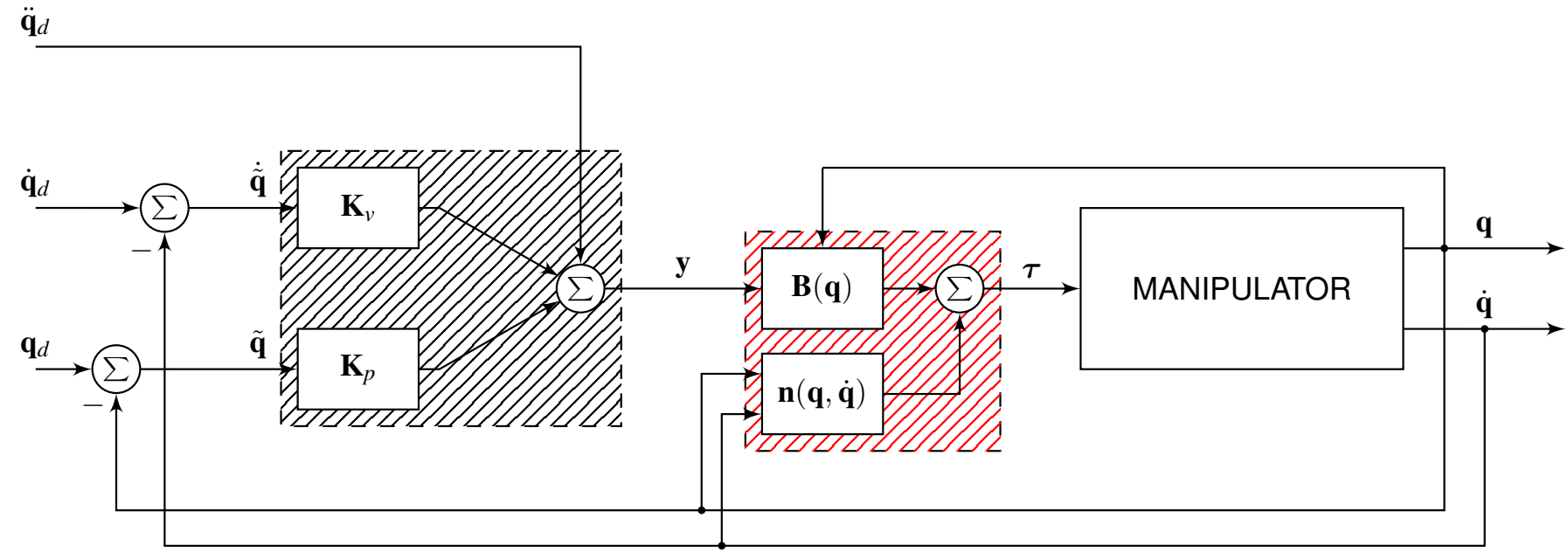
$$\ddot{\tilde{\mathbf{q}}} + \mathbf{K}_v \dot{\tilde{\mathbf{q}}} + \mathbf{K}_p \tilde{\mathbf{q}} = \mathbf{0}$$

which converges to zero with a speed depending on the matrices $\mathbf{K}_v$ and $\mathbf{K}_p$ chosen.

Model-based centralized controller for tracking

Inverse dynamics controller design

The inner feedback loop is based on the robot dynamic model.



STABILIZING LINEAR CONTROL

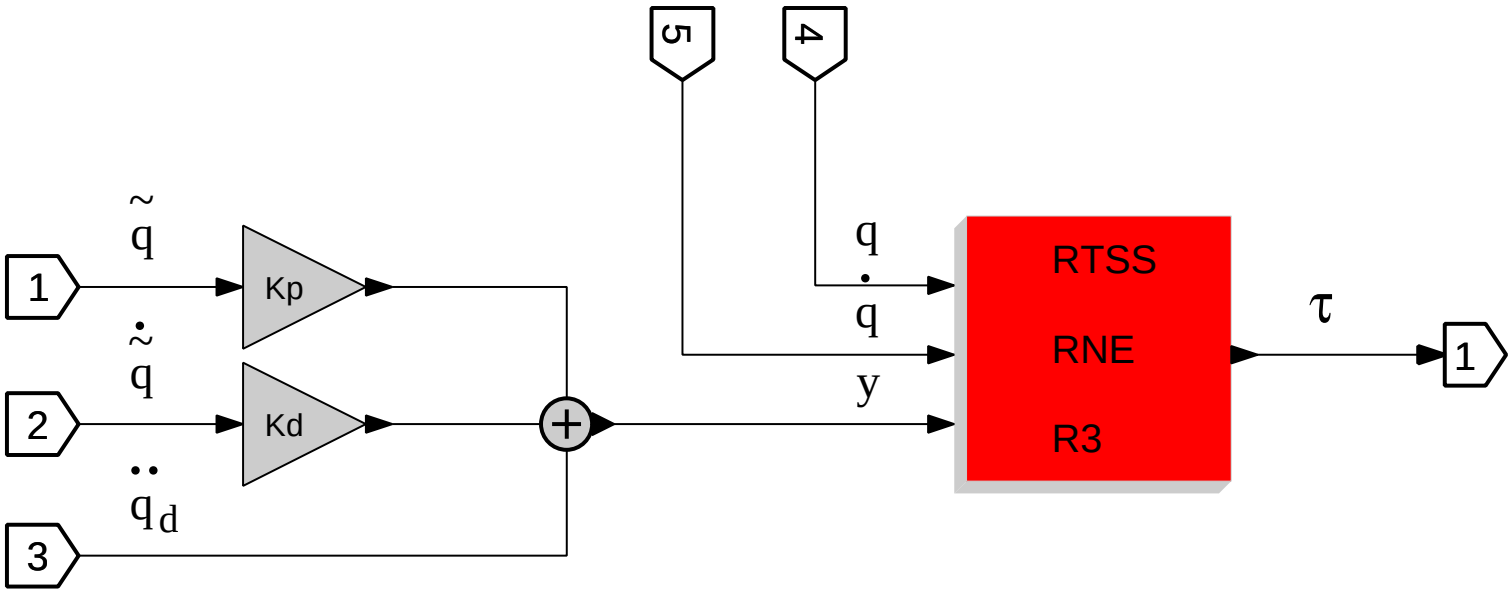
NONLINEAR COMPENSATION AND DECOUPLING

Block scheme of joint space inverse dynamics control

Model-based centralized controller for tracking

Scicos implementation of the inverse dynamics controller for the Manutec r3

Key component: the inverse dynamics block (IDB) of the `Robotics` palette.



Inverse dynamics controller for the Manutec r3 robot

First-order closed-loop inverse kinematics

On the generation of joint space reference inputs to the motion control system

Considerations regarding the above control scheme

- Vectors $\mathbf{q}_d$, $\dot{\mathbf{q}}_d$ and $\ddot{\mathbf{q}}_d$ were assumed available;
- joint space references were computed from two joint coordinate poses directly specified by the user.

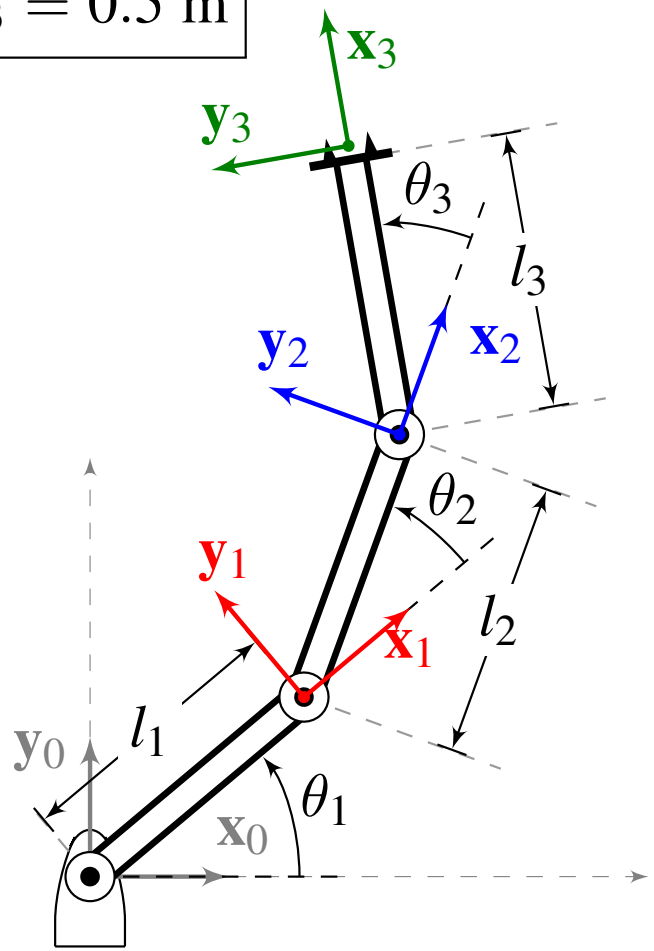
However, motion specifications are usually assigned in the operational space.

Inverse kinematics algorithms transform task space references into joint space references.

First-order closed-loop inverse kinematics

Kinematic inversion of a simple three-link RRR planar arm

$l_{1,2,3} = 0.5$ m



RRR arm at generic pose

Link	α_i	a_i	θ_i	d_i
1	0	l_1	θ_1	0
2	0	l_2	θ_2	0
3	0	l_3	θ_3	0

Denavit-Hartenberg table

Robot kinematic model

```
dh_123=[0,l_123,0,0];
R3arm = rt_robot([dh_123;dh_123;dh_123],..
       "RRR arm", "", "Sciavicco-Siciliano");
```

First-order closed-loop inverse kinematics

Kinematic inversion of a simple three-link RRR planar arm

Simulation scenario [Sciavicco and Siciliano, 2000]

- $\mathbf{q}(0) = \begin{bmatrix} \pi & -\pi/2 & -\pi/2 \end{bmatrix}^T$ [rad];
- circular desired motion trajectory for $0 \leq t \leq 4$ s:

$$\mathbf{x}_d(t) = \begin{bmatrix} \mathbf{p}_d(t) \\ \phi_d(t) \end{bmatrix} = \begin{bmatrix} 0.25(1 - \cos \pi t) \\ 0.25(2 + \sin \pi t) \\ \sin \dfrac{\pi}{24} t \end{bmatrix};$$

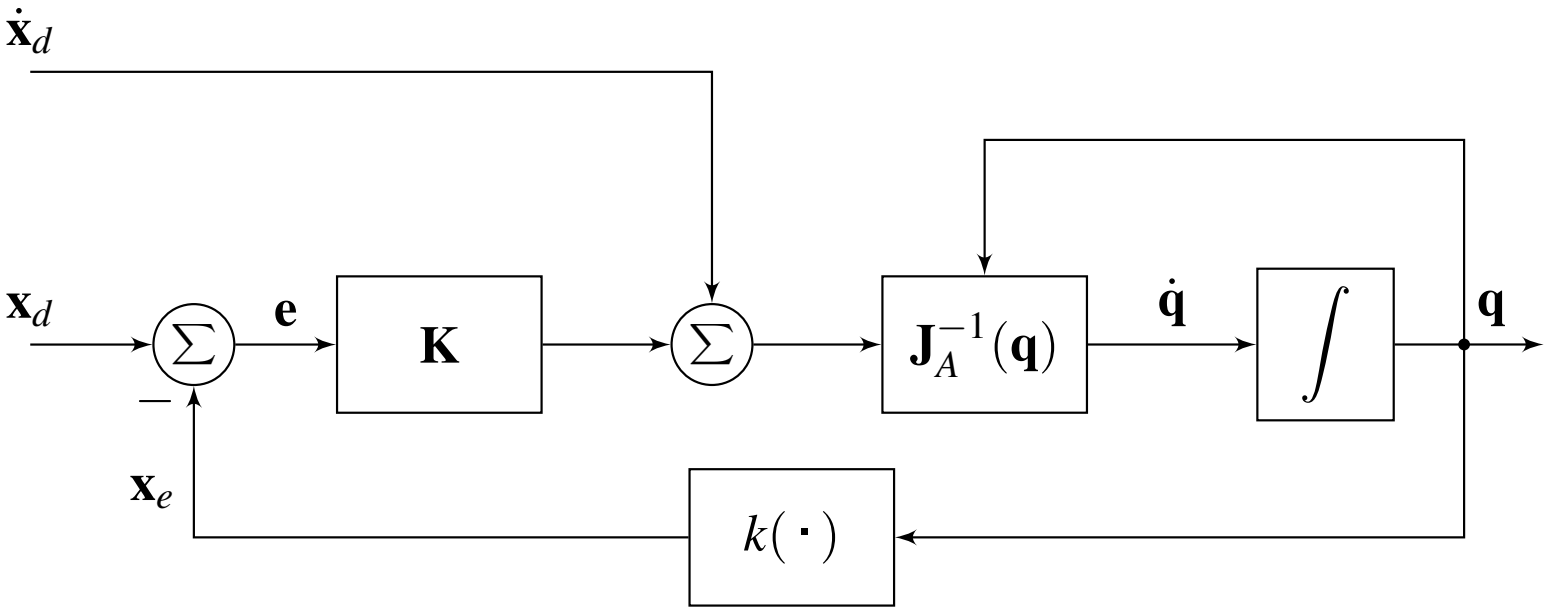
- forward Euler numerical integration scheme ($\Delta t = 1$ ms);
- final simulation time $t_{end} = 5$ s.

First-order closed-loop inverse kinematics

The Jacobian inverse algorithm

The Jacobian inverse algorithm integrates the joint velocity vector

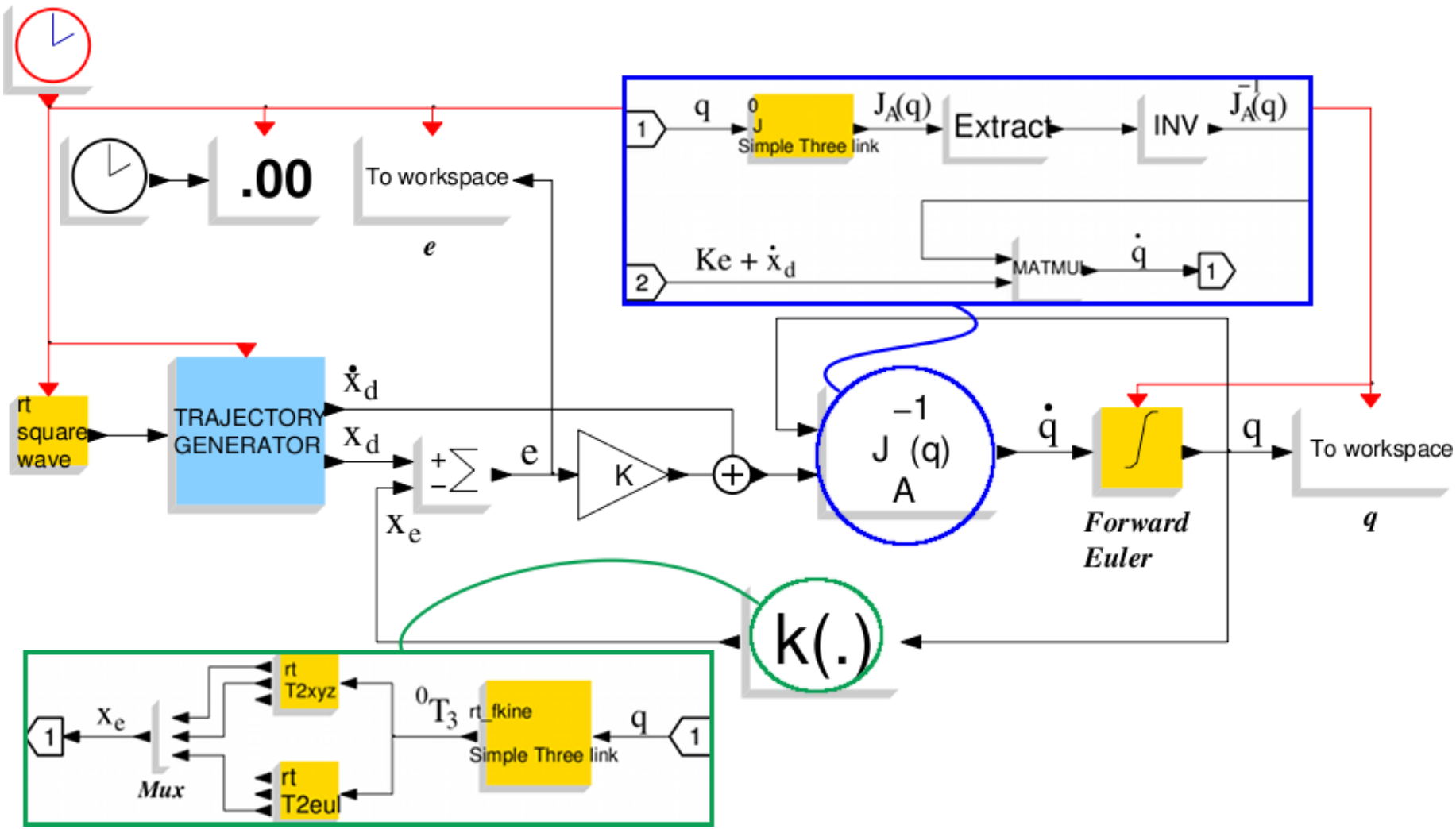
$$\dot{\mathbf{q}} = \mathbf{J}_A^{-1}(\mathbf{q})(\dot{\mathbf{x}}_d + \mathbf{K}\mathbf{e})$$



Jacobian inverse algorithm

First-order closed-loop inverse kinematics

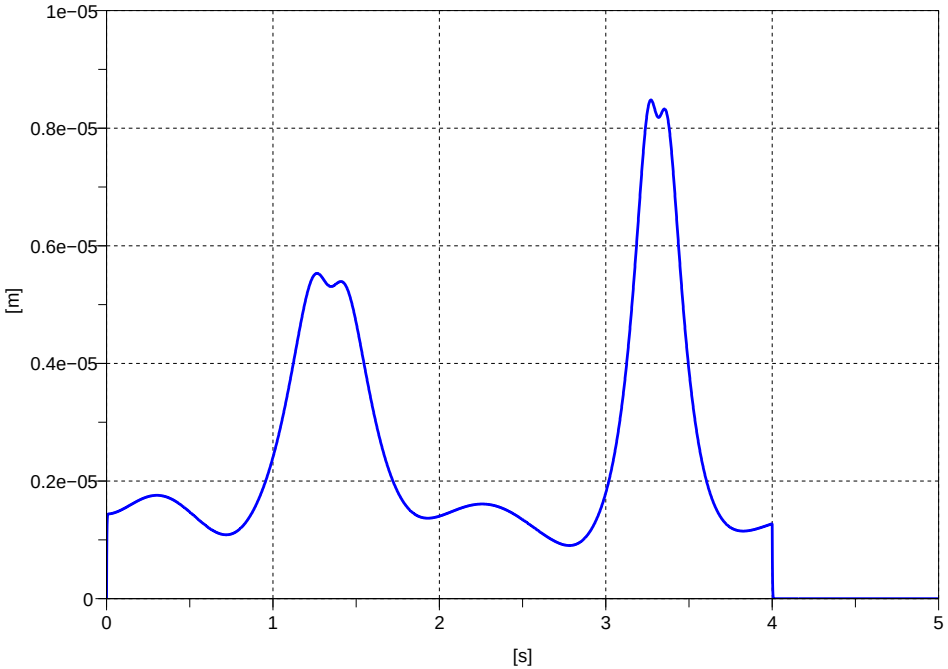
Scicos block diagram for the Jacobian inverse algorithm



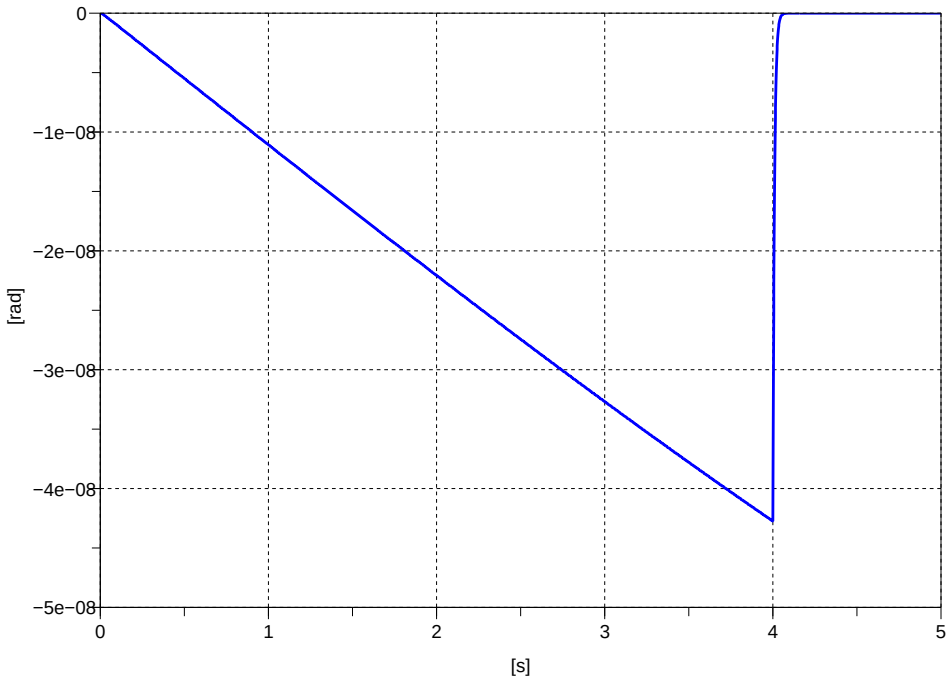
The Jacobian inverse algorithm and the blocks $\mathbf{J}_A^{-1}(\mathbf{q})$ and $k(\cdot)$

First-order closed-loop inverse kinematics

Analysis of simulation results



Time history of the norm of end-effector position error



Time history of the end-effector orientation error

RRR planar arm performing the task Launch the movie!

First-order closed-loop inverse kinematics

Redundancy resolution: short review of the mathematics involved

If the EE orientation is not constrained, a redundant degree of mobility is available.

The solution of the Jacobian inverse algorithm is generalized into

$$\dot{\mathbf{q}} = \mathbf{J}_A^{\dagger}(\mathbf{q})(\dot{\mathbf{x}}_d + \mathbf{K}\mathbf{e}) + (\mathbf{I} - \mathbf{J}_A^{\dagger}\mathbf{J}_A)\dot{\mathbf{q}}_0$$

Convenient utilization of redundant DOFs

$$\dot{\mathbf{q}}_0 = k_0 \left(\frac{\partial w(\mathbf{q})}{\partial \mathbf{q}} \right)^T$$

- $k_0 > 0$;
- $w(\mathbf{q})$ secondary objective function.

First-order closed-loop inverse kinematics

Solution which maximizes the distance from mechanical joint limits of the RRR arm

Distance from joint limits

$$w(\mathbf{q}) = -\frac{1}{6}\sum_{i=1}^{3}\left(\frac{q_i - \bar{q}_i}{q_{iM} - q_{im}}\right)^2$$

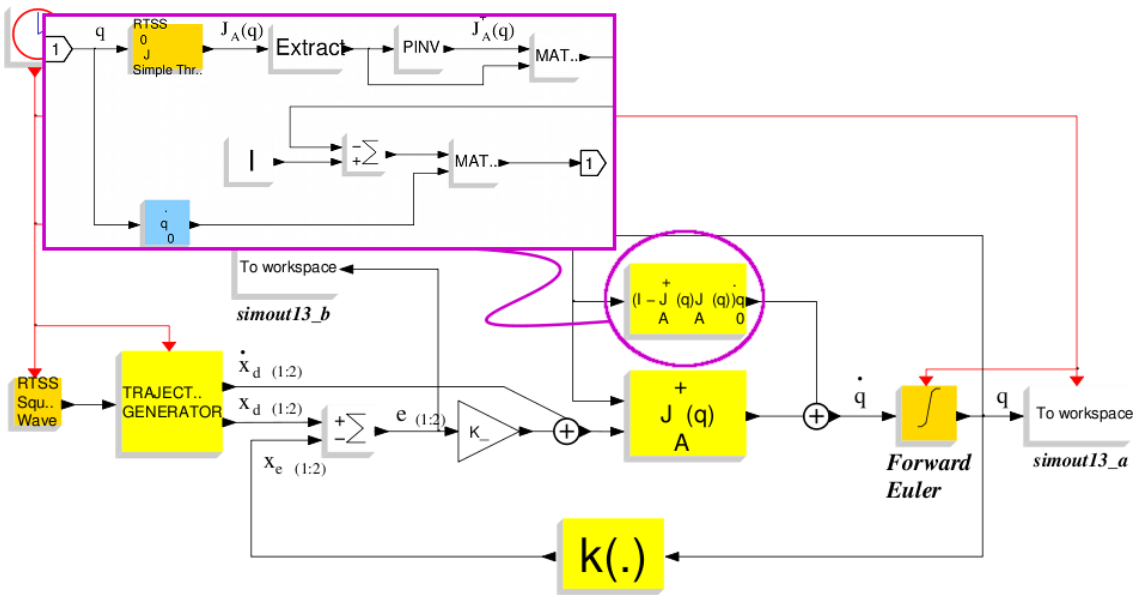
- maximizing the distance;
- q_i current joint angle;
- q_{iM} maximum joint limit;
- q_{im} minimum joint limit;
- $\bar{q}_i$ middle value.

Simulation scenario

- $q_{1m} = -2\pi$, $q_{1M} = 2\pi$;
- $q_{2m} = -\pi/2$, $q_{2M} = \pi/2$;
- $q_{3m} = -3\pi/2$, $q_{3M} = -\pi/2$;
- $k_0 = 250$.

First-order closed-loop inverse kinematics

Solution which maximizes the distance from mechanical joint limits of the RRR arm



The Jacobian pseudo-inverse algorithm with redundancy resolution

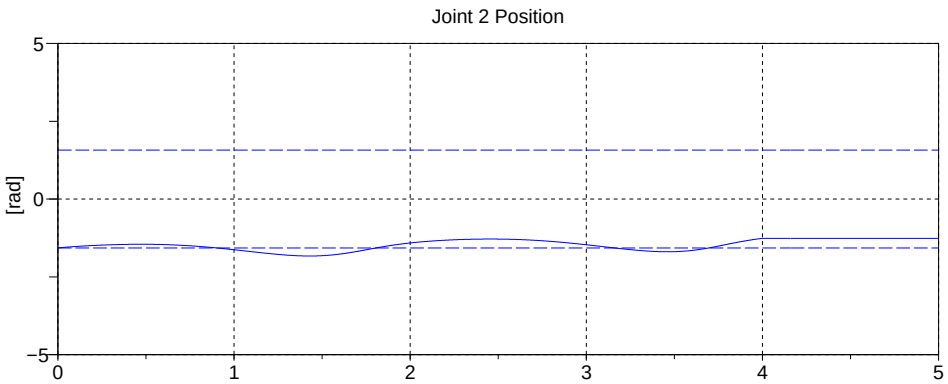
Computational code of block $\dot{q}_0$

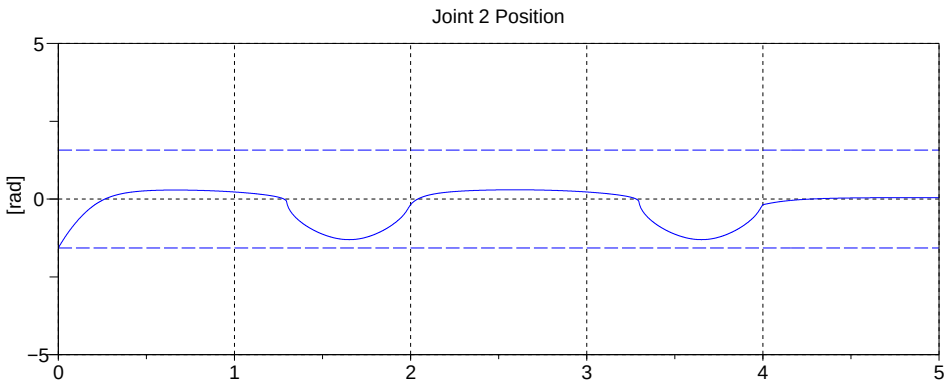
```
double * y, * u;
double pi = 3.1415927;

// in/out pointers
y = GetRealOutPortPtrs(block, 1);
u = GetRealInPortPtrs(block, 1);

// computational code
y[0] = -250/3 * u[0]/(16*pi*pi);
y[1] = -250/3 * u[1]/(pi*pi);
y[2] = -250/3 * (u[2]+pi)/(pi*pi);
```

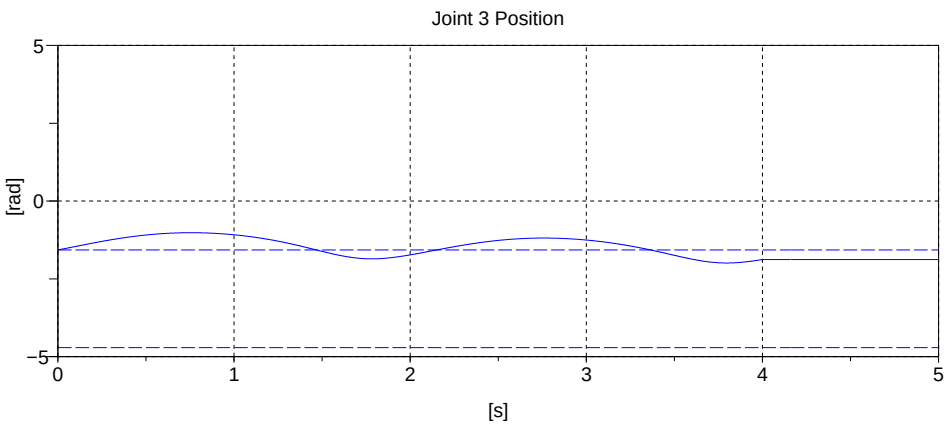
First-order closed-loop inverse kinematics

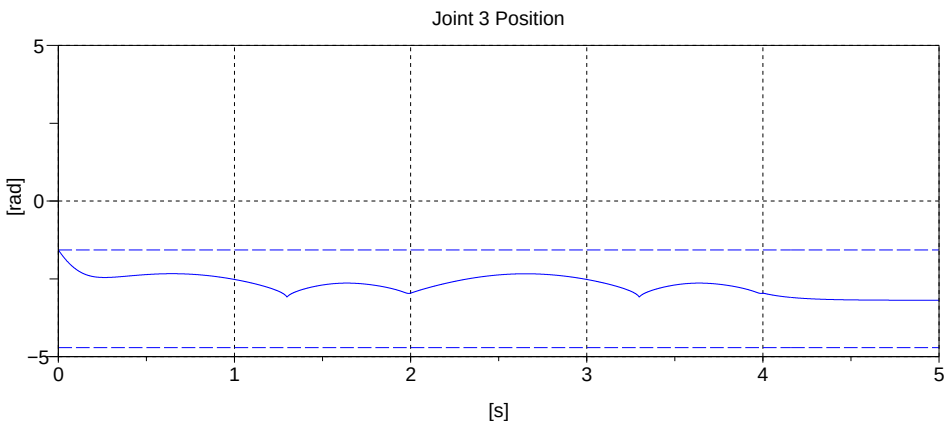
Analysis of simulation results: time history of the joint trajectories

Joints 2 and 3 keep far from their min. and max. limit, respectiv.









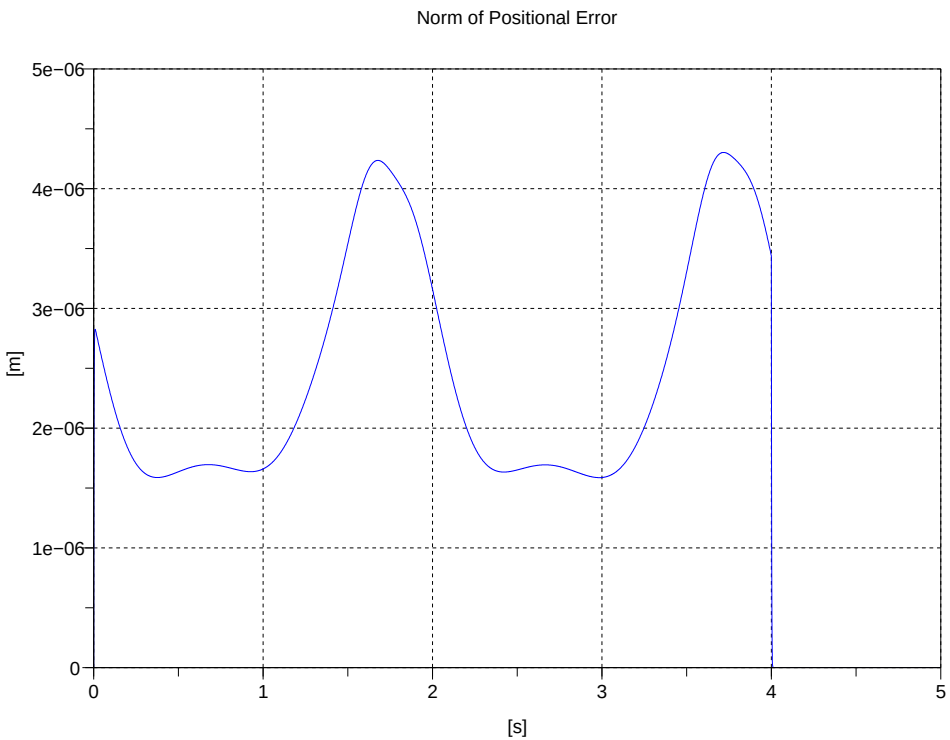
Without redundancy resolution

With utilization of redundancy

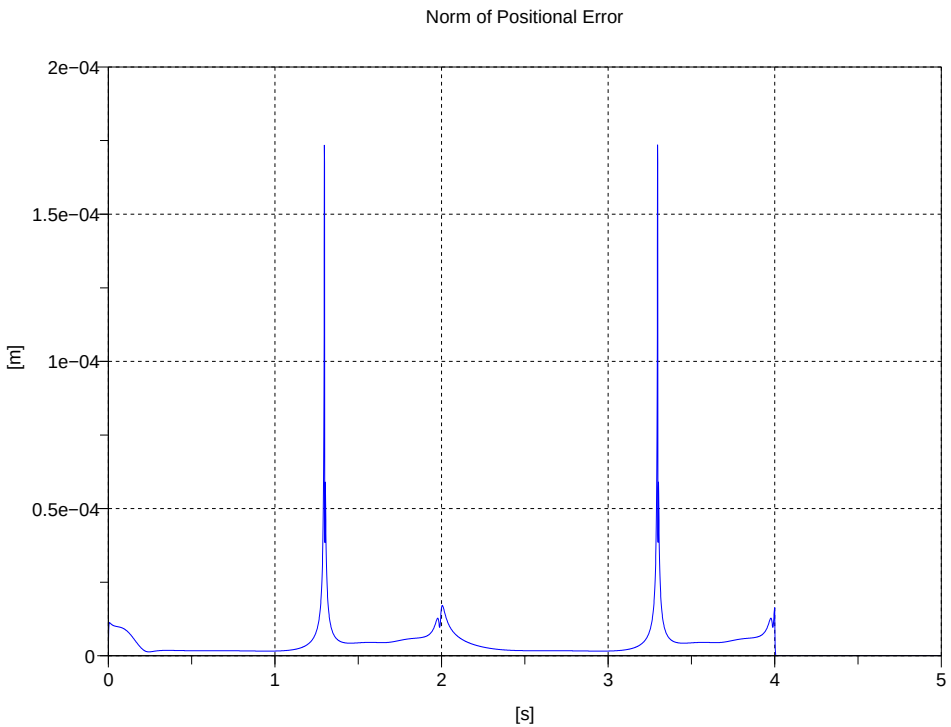
First-order closed-loop inverse kinematics

Analysis of simulation results: time history of the norm of EE position error

Such an effort does not appreciably influences the position tracking error.



Without redundancy resolution

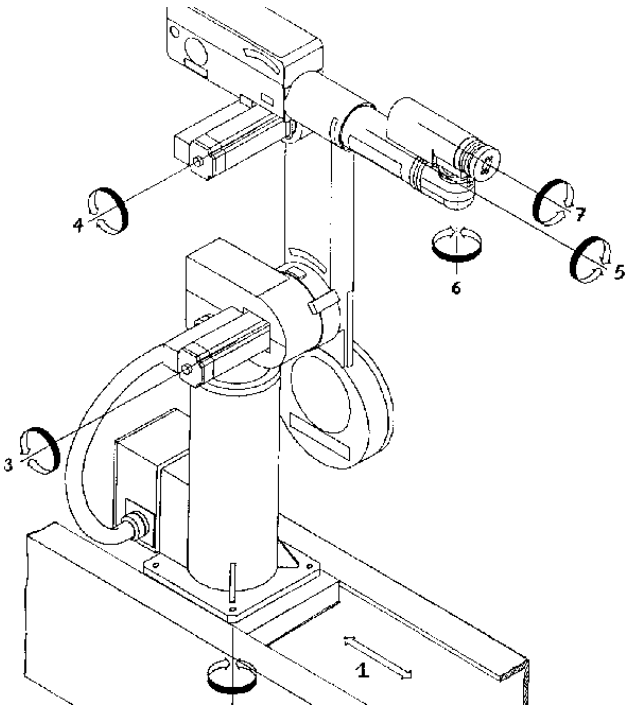


With utilization of redundancy

First-order closed-loop inverse kinematics

Kinematic inversion of a 7-DOF industrial manipulator with non-spherical wrist

A different definition of the orientation error must be used.



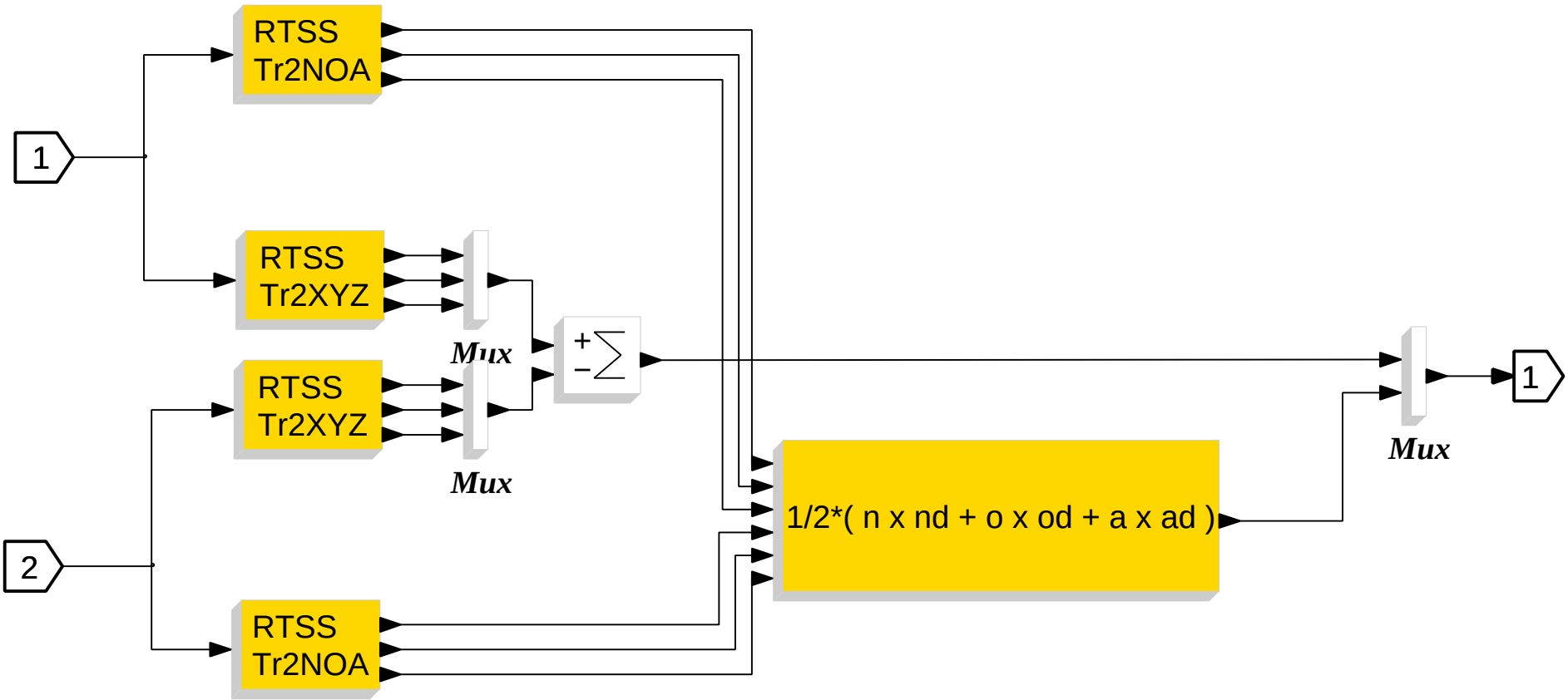
Kinematic structure of the Comau SMART 3-S robot

Simulation scenario [Caccavale et al., 1996]

```
// initial configuration
tri = rt_roty(%pi)*rt_rotz(-2/3*%pi);
tpi = rt_transl([.975, -2.194, 1.288]);
ti = tpi*tri;

// final configuration
trf = rt_roty(%pi)*rt_rotz(3/4*%pi);
tpf = rt_transl([.975, -.594, 1.288]);
tf = tpf*trf;

// travelling time 6 sec.
```

First-order closed-loop inverse kinematics

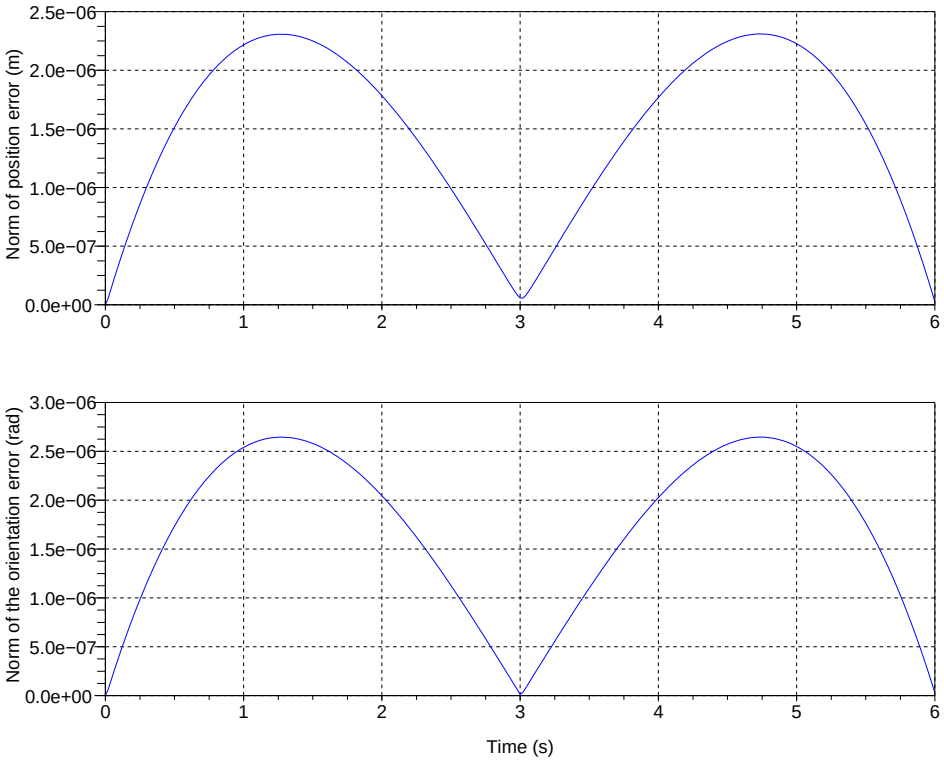
Kinematic inversion of a 7-DOF industrial manipulator with non-spherical wrist



Error block for axis-angle representation of the orientation

First-order closed-loop inverse kinematics

Analysis of simulation results



Time history of the EE error

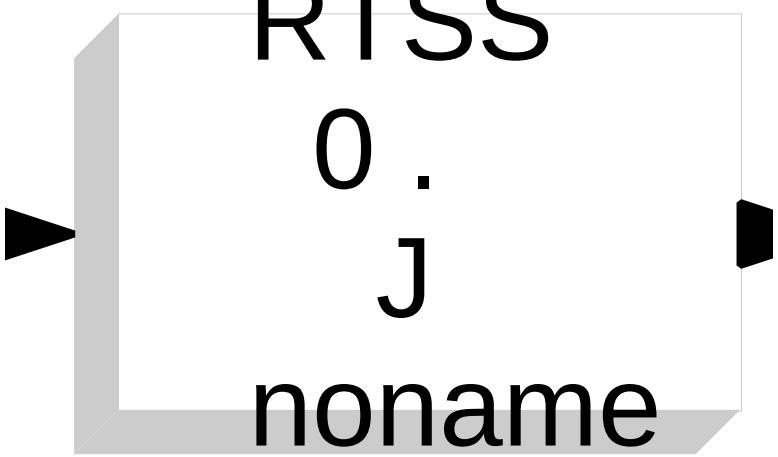
Launch the movie!

Comau SMART 3-S performing the task

Outline

1. Robotic manipulators modelling with ScicosLab
   - Rigid body transformations
   - Building serial-link manipulator models
   - Analysis of serial-link manipulators
2. Robot control systems design using Scicos
   - Basic concepts
   - Motion control
   - Further applications
3. Robot control code generation for use with Linux RTAI
   - Developing RTAI-based centralized controllers with RTAI-Lab

Further applications
Blocks under developments

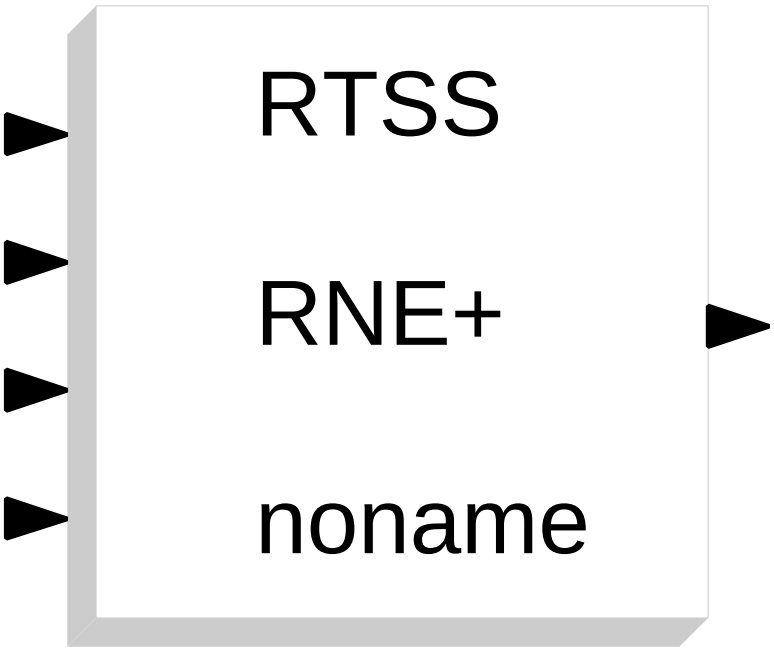


Time-derivative Jacobian block

Applications

- Second order CLIK algorithms;
- operational space control;
- interaction control.

Further applications
Blocks under developments



Inverse dynamics block plus environment interaction

Applications

- interaction control.

Further applications
Blocks under developments

- Unit quaternion blocks;
- other blocks for homogeneous transforms.

Applications

- Countless.

Outline

1. Robotic manipulators modelling with ScicosLab
   - Rigid body transformations
   - Building serial-link manipulator models
   - Analysis of serial-link manipulators
2. Robot control systems design using Scicos
   - Basic concepts
   - Motion control
   - Further applications
3. Robot control code generation for use with Linux RTAI
   - Developing RTAI-based centralized controllers with RTAI-Lab

Real-Time inverse dynamics controller for the Pelican

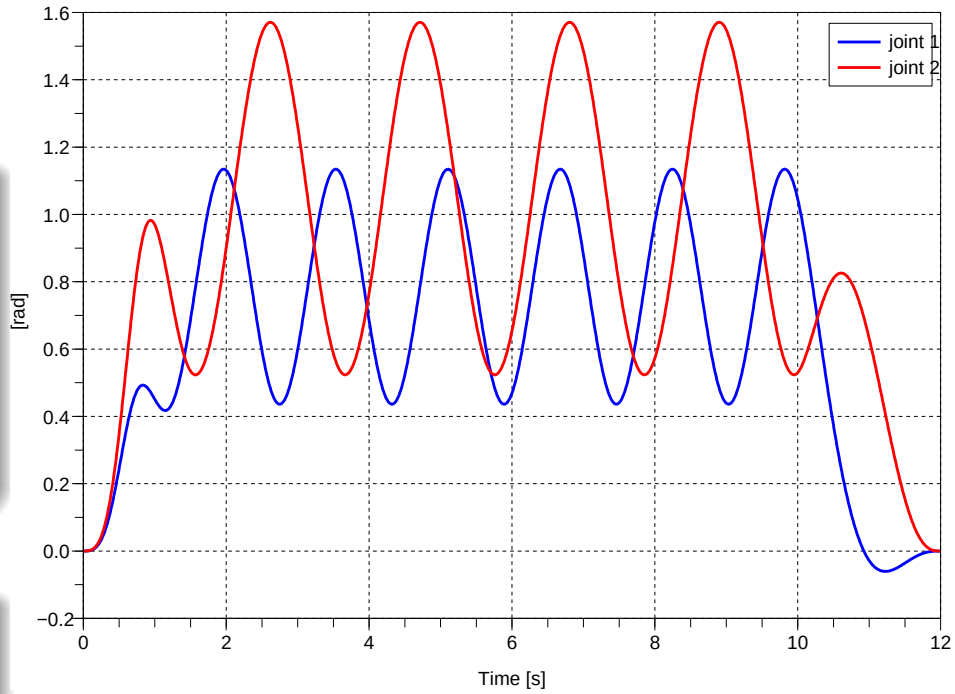
Desired reference trajectories in joint space

Positions

- Sinusoidal term (0–10 s);
- homing traj. (10–12 s);
- sampling time: 5 ms.

Velocities and accelerations

- By direct differentiation.

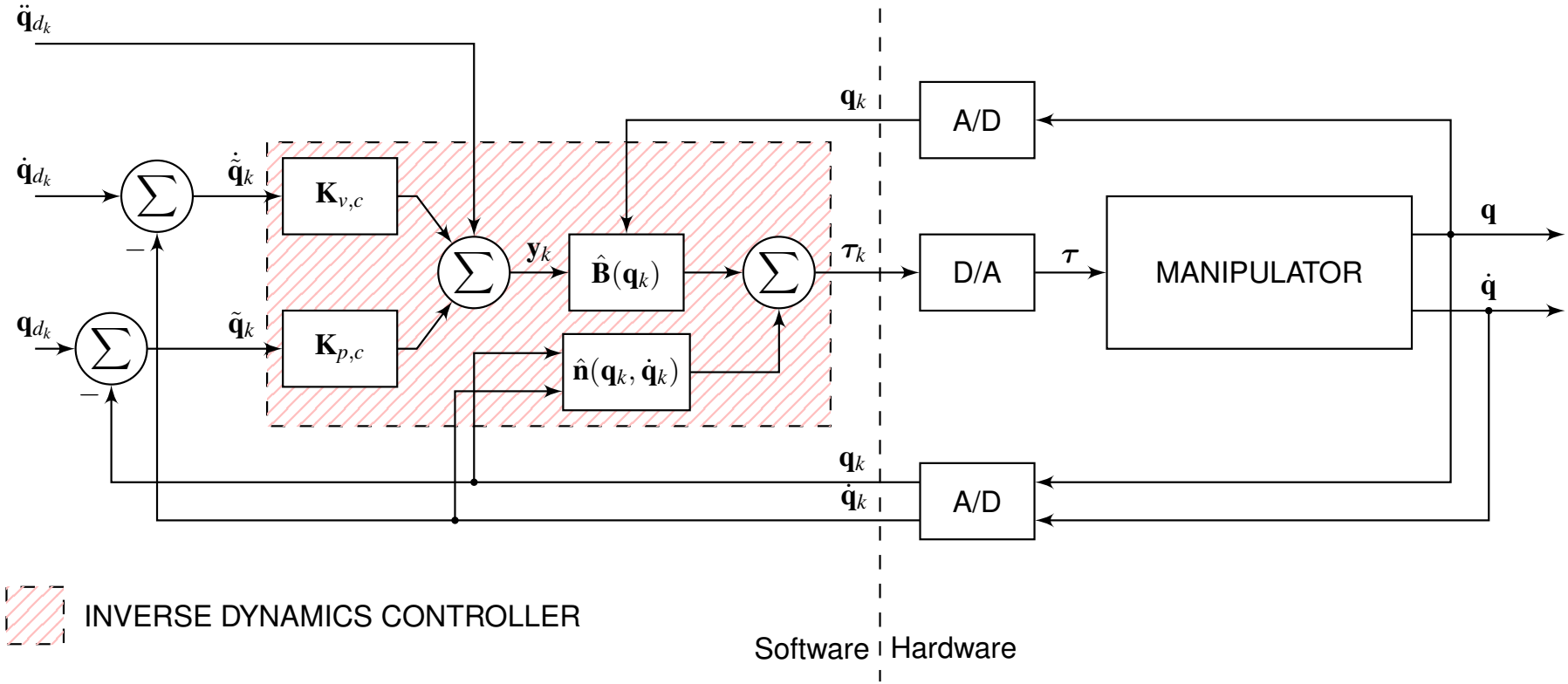


Graphs of the reference positions against time

Real-Time inverse dynamics controller for the Pelican

On the digital implementation of the control system

Digital implementation of the control law amounts to discretizing the inner and outer control loops.

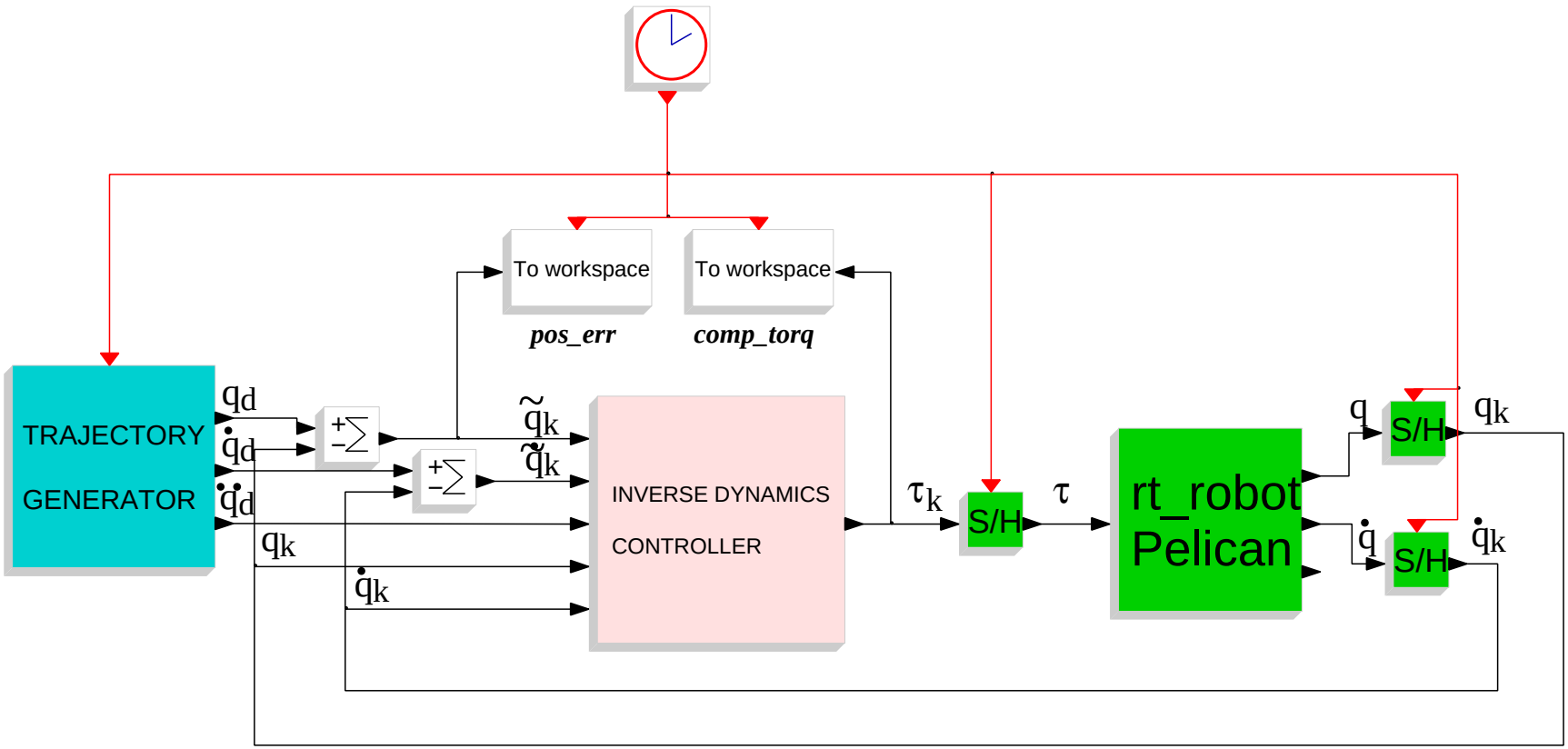


Digital implementation of the inverse dynamics control scheme

Real-Time inverse dynamics controller for the Pelican

Scicos block diagram for simulation

Trajectory generator uses a set of “From Workspace” blocks from the standard `Sources` palette.



Scicos block diagram for the control system (simulation)

Real-Time inverse dynamics controller for the Pelican

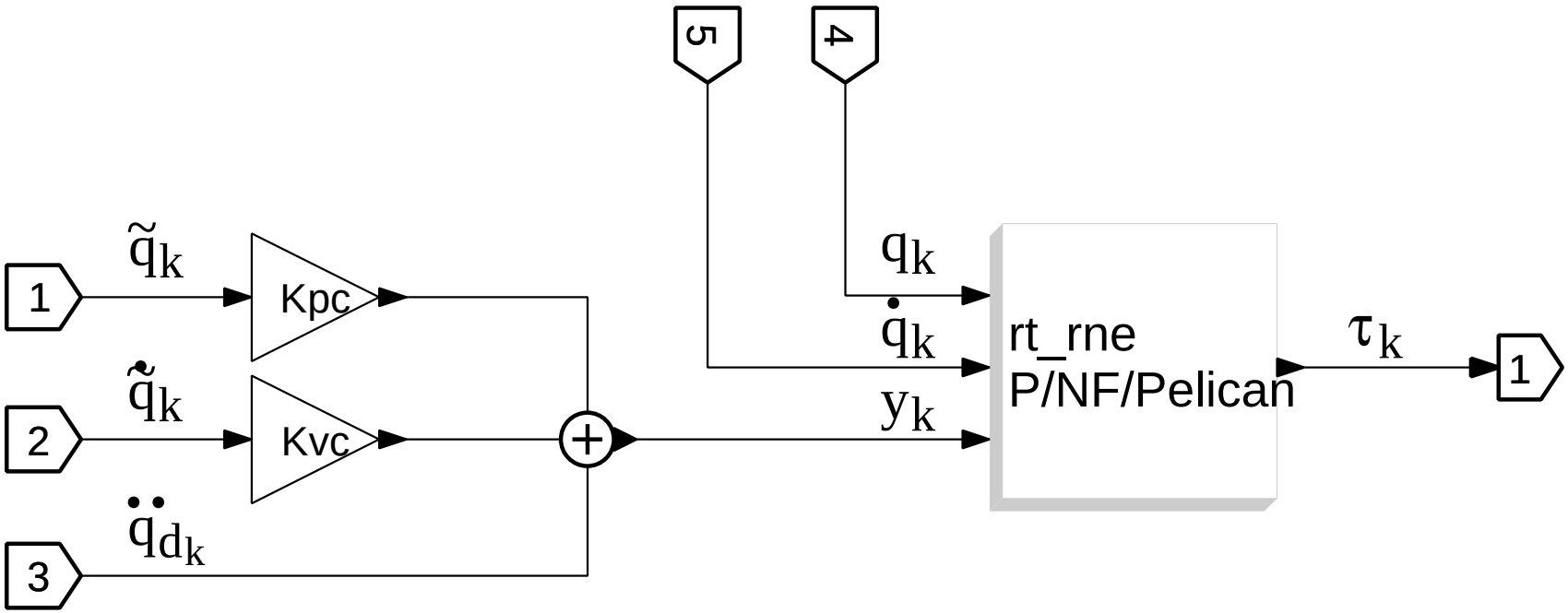
A look inside the “Inverse dynamics controller” block

Coded forms of the PD gains

$$\mathbf{K}_{p,c} = \mathrm{diag}\{1500, 14000\} \quad [\mathrm{s}^{-2}]$$
$$\mathbf{K}_{v,c} = \mathrm{diag}\{76.21, 353.71\} \quad [\mathrm{s}^{-1}]$$

Context of the diagram

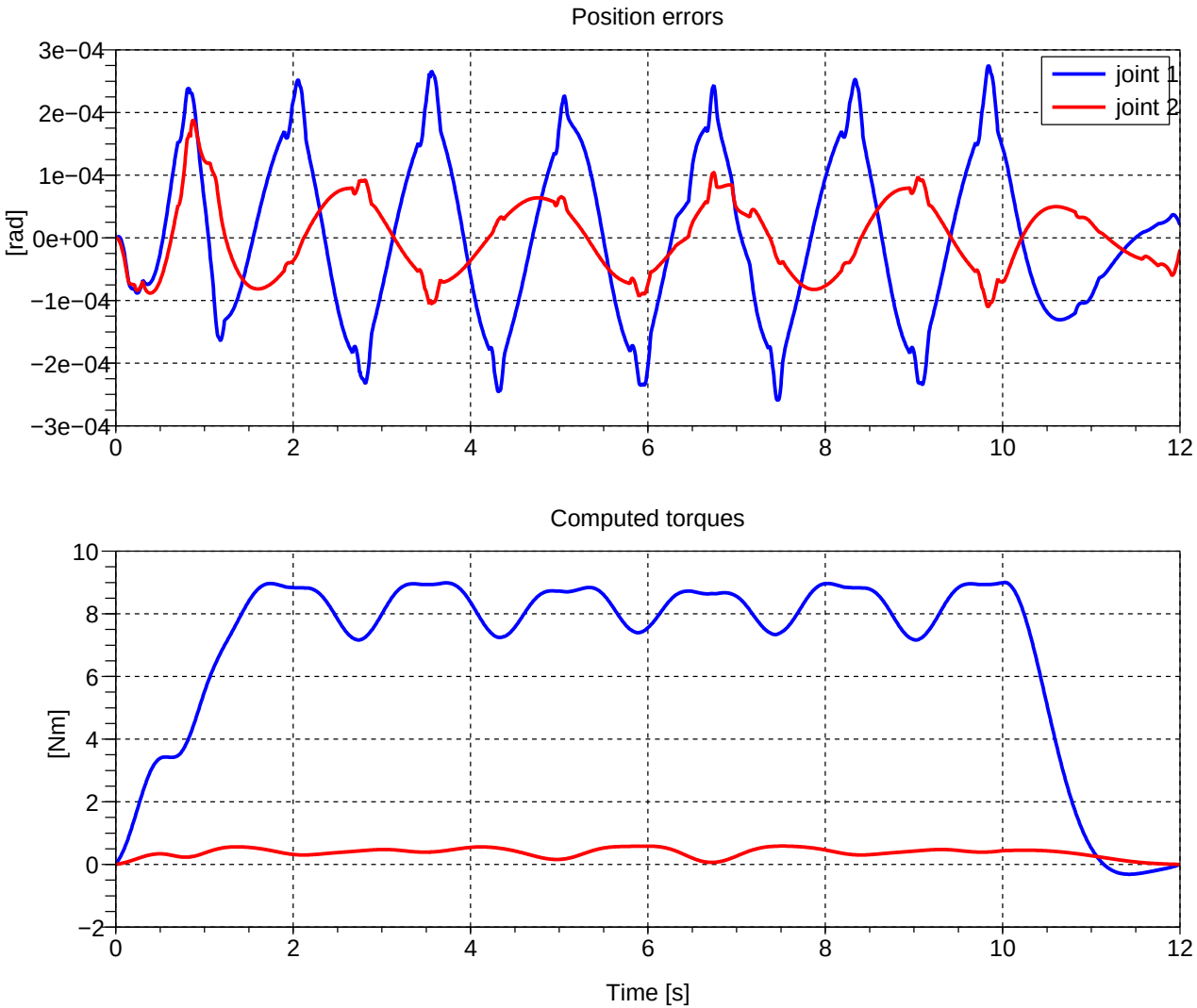
```
pelnf = rt_nofriction(pel, 'coulomb');
pelp = rt_perturb(pelnf, 0.25);
Kpc = diag([1500, 14000]);
Kvc = diag([76.21, 353.71]);
```



Block diagram for the digital controller

Real-Time inverse dynamics controller for the Pelican

Analysis of simulation results

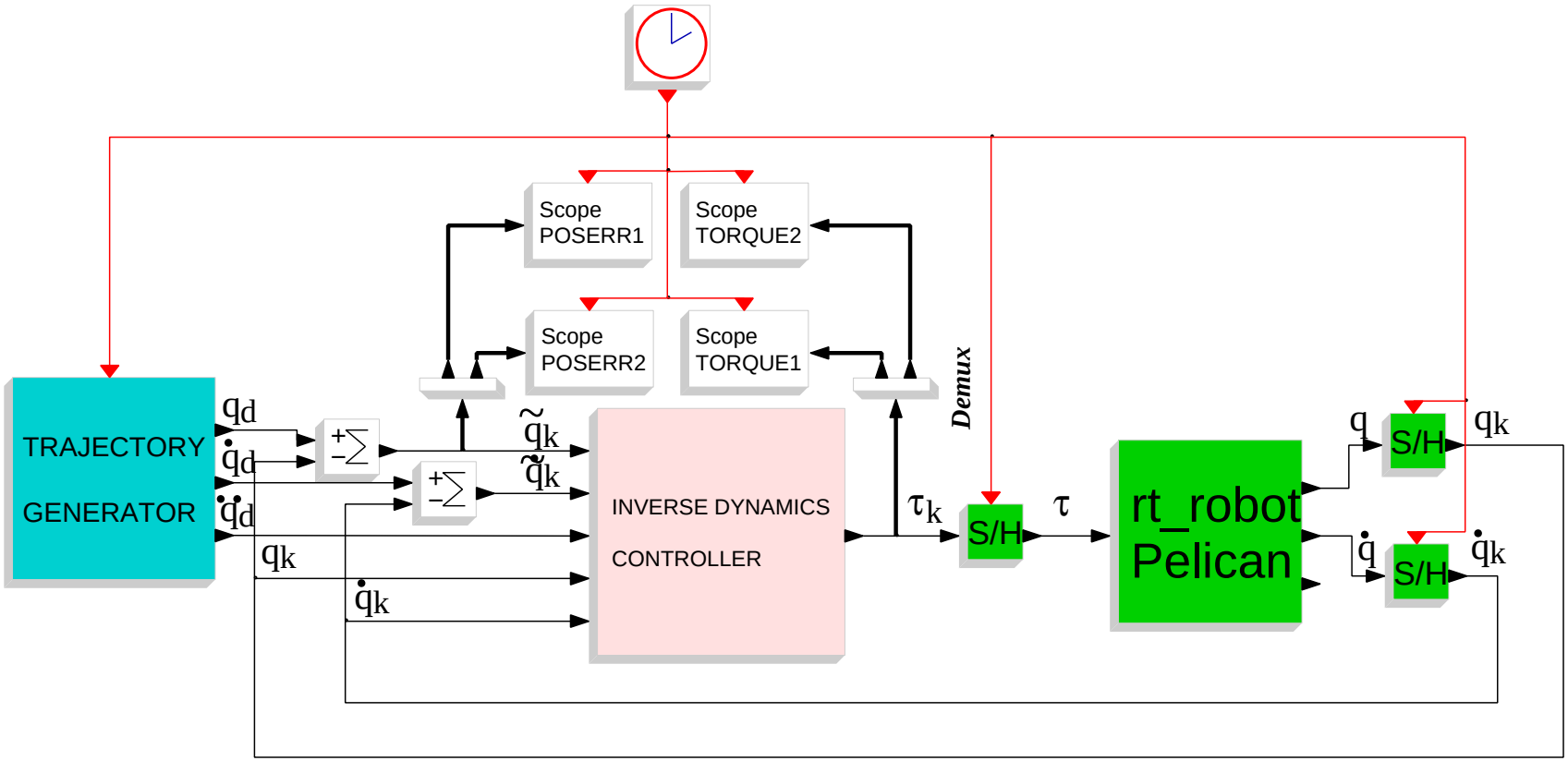


Graph of position errors and computed torques against the time

Real-Time inverse dynamics controller for the Pelican

Scicos block diagram for real time code generation

Difference between the diagrams for simulation and real time code generation: trajectory generator and scope blocks.

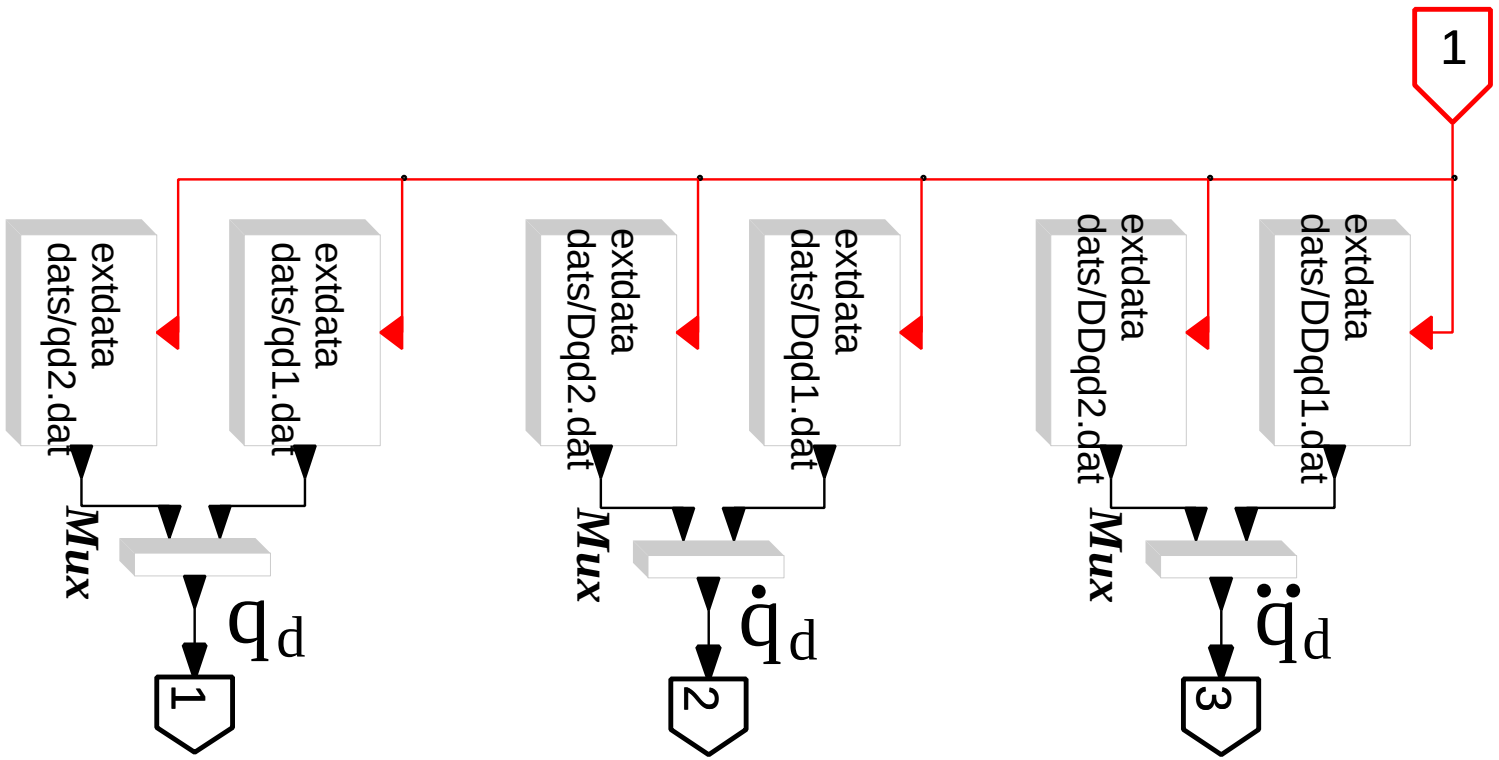


Scicos block diagram for the control system (real time control)

Real-Time inverse dynamics controller for the Pelican

A look inside the trajectory generator

Trajectory generator uses a set of “extdata” blocks from the `RTAI-Lib` palette.

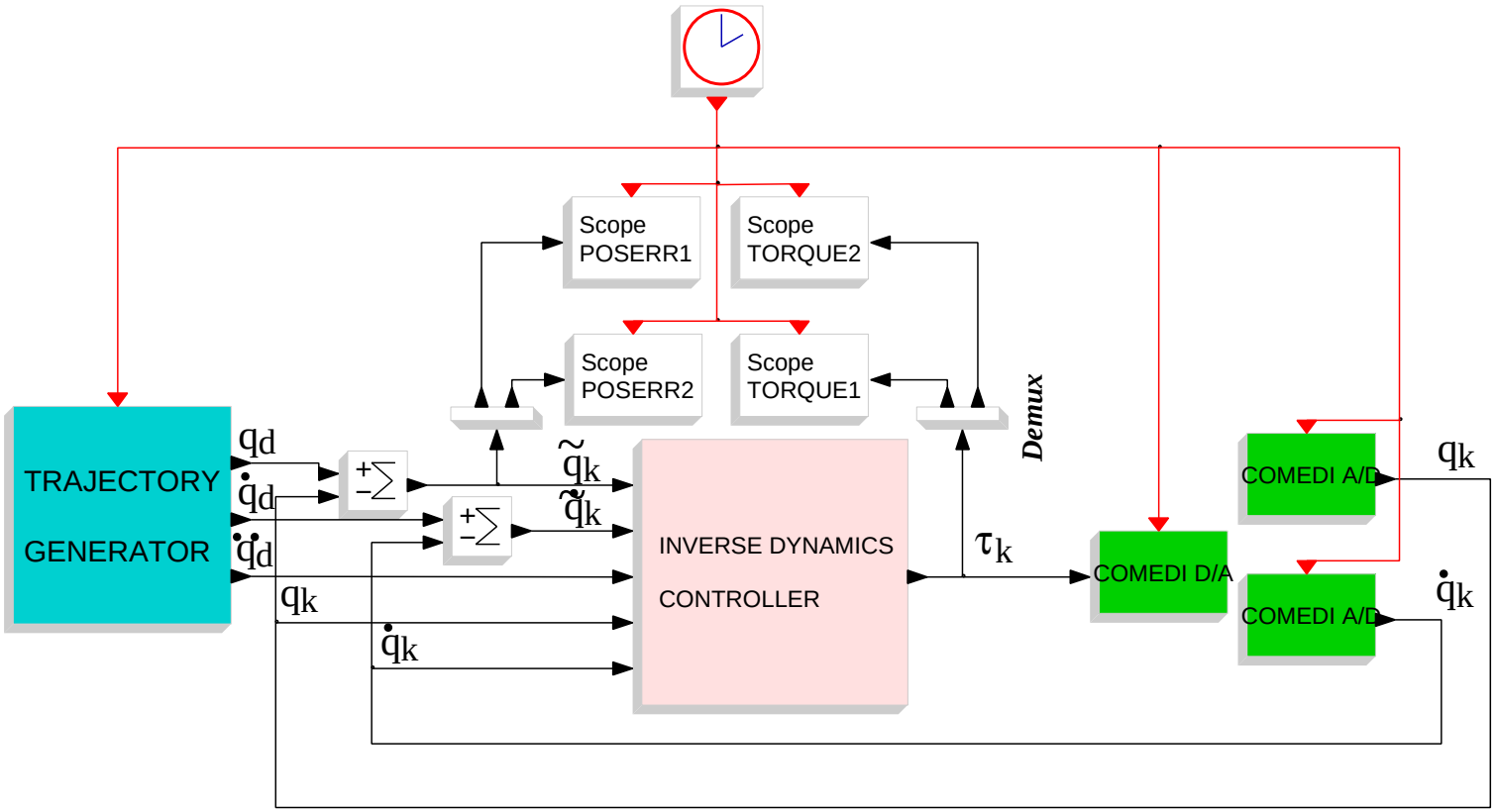


Block diagram for the trajectory generator (real time control)

Real-Time inverse dynamics controller for the Pelican

What if the robot arm were physically available?

The mathematical representation of the robot should be substituted by `COMEDI` DAC/ADC blocks from `RTAI-Lib`.



Block diagram for real time control with a set of `COMEDI` blocks

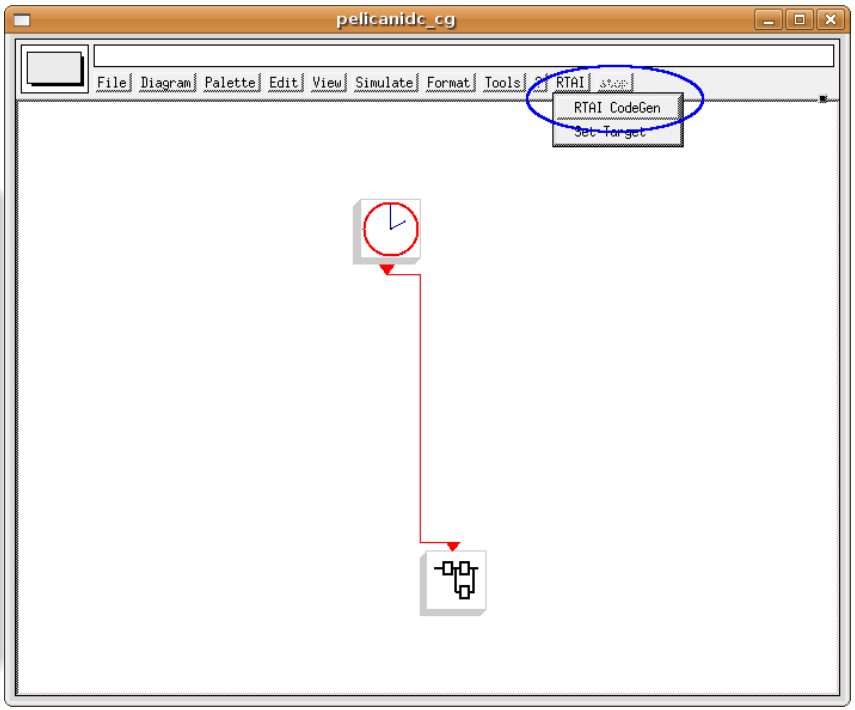
Real-Time inverse dynamics controller for the Pelican

Standalone, hard real time controller generation in two steps (1)

Step 1: Excluding the Clock, construct a super block (SB) out of the diagram for real time control (RTC).

Actions

1. Use the `Region-to-Super-block` facility to construct the SB;
2. click the `RTAI CodeGen` button;
3. click on the SB.

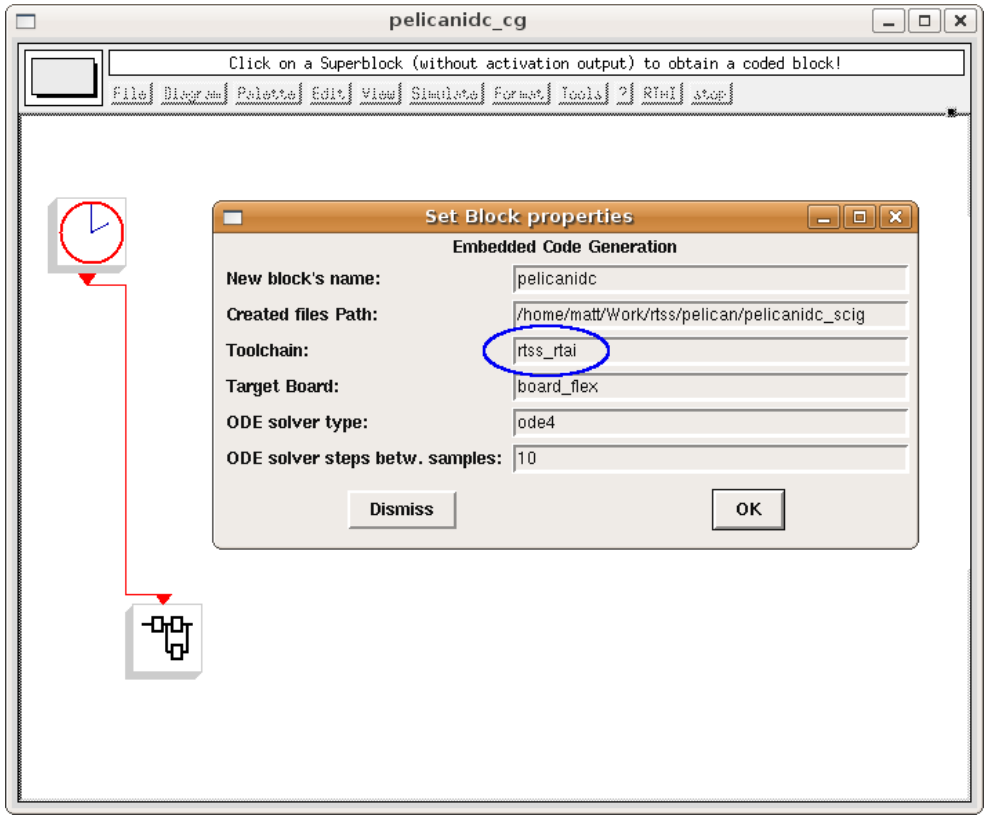


RTC diagram after step 1

Real-Time inverse dynamics controller for the Pelican

Standalone, hard real time controller generation in two steps (2)

Step 2: Adjust the properties for code generation by setting `rtss_rtai` as Toolchain and press `OK`.

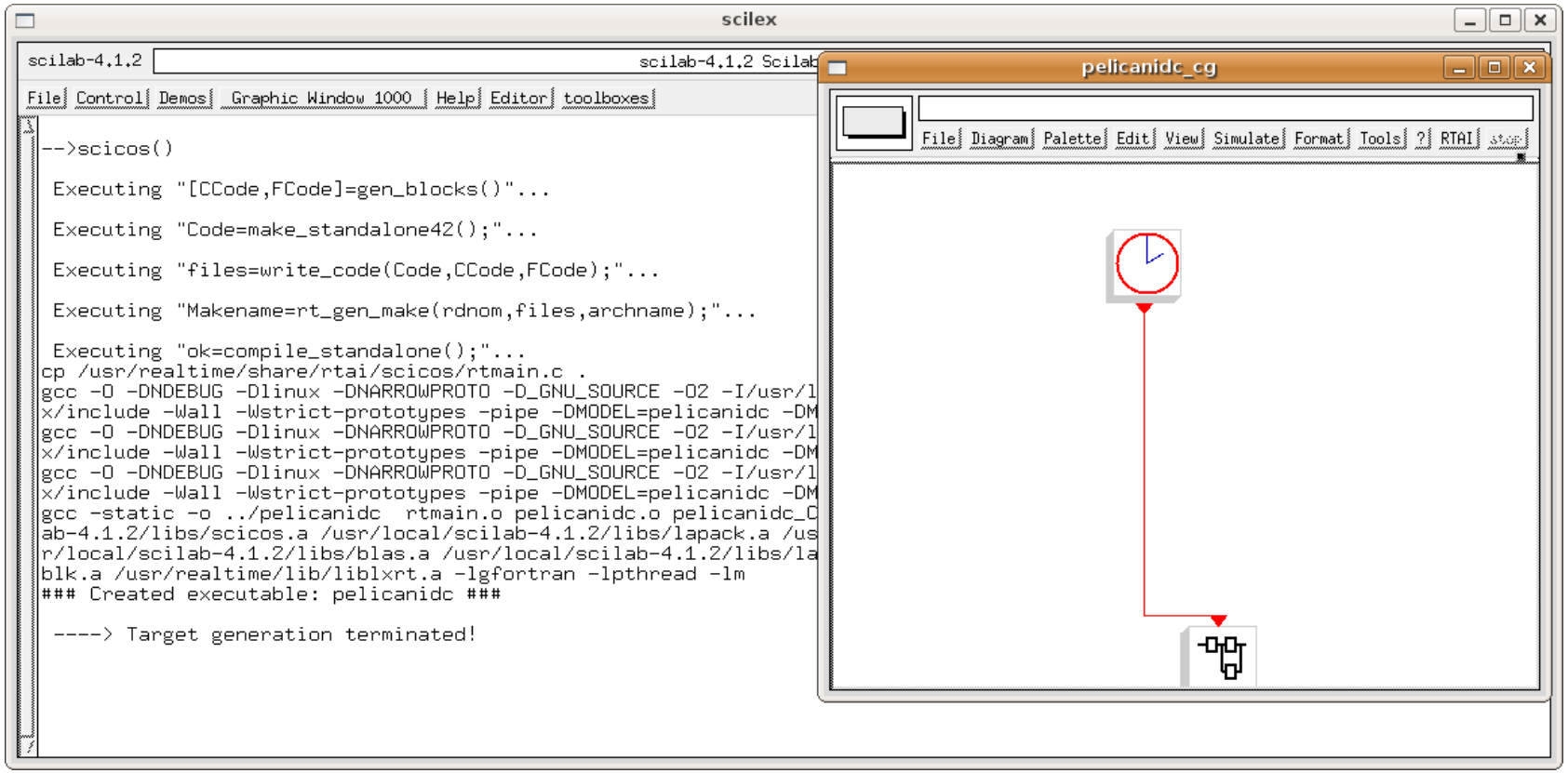


`RTAI CodeGen` dialog box

Real-Time inverse dynamics controller for the Pelican

Standalone, hard real time controller generation

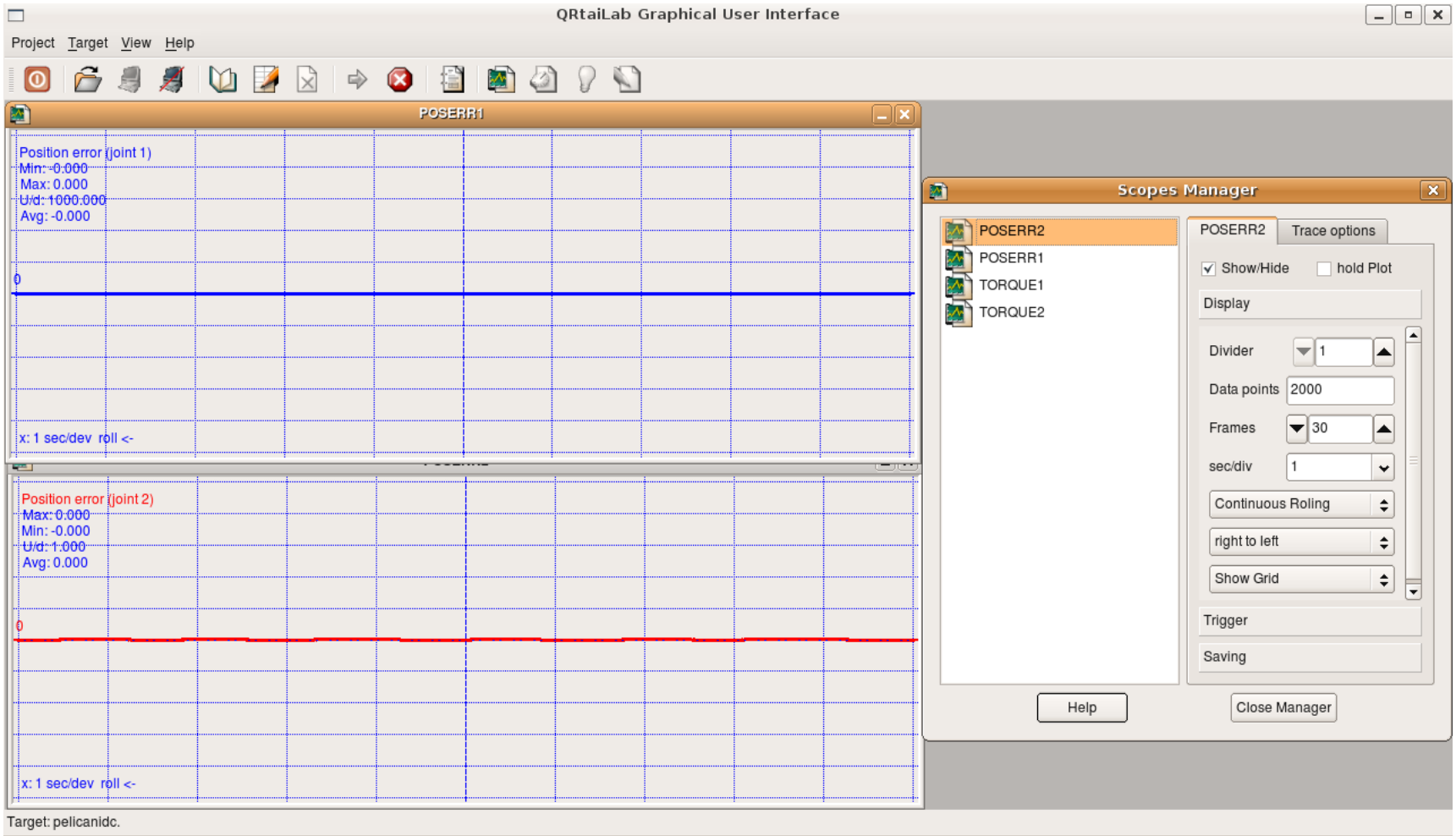
The compilation starts and completes. An executable file called `pelicanidc` is created.



Compilation output in the Scilab window

Real-Time inverse dynamics controller for the Pelican

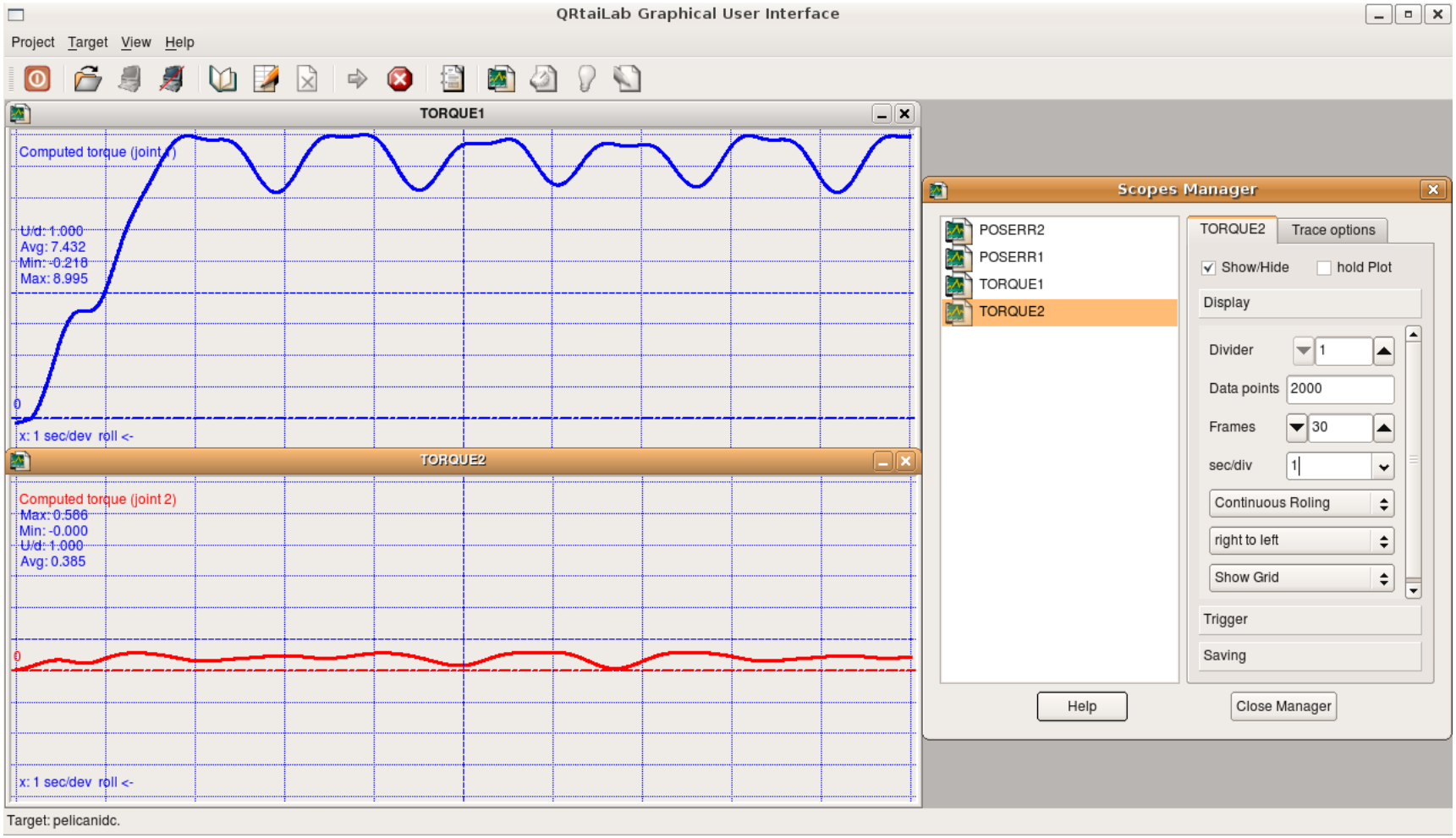
Monitoring the real time controller with the QRtaiLab Graphical User Interface



QRtaiLab with the scope manager and the position errors scopes

Real-Time inverse dynamics controller for the Pelican

Monitoring the real time controller with the QRtaiLab Graphical User Interface



QRtaiLab with the scope manager and the computed torques scopes

Summary

Main features of RTSS

- Modelling and simulation of robotic manipulators in the ScicosLab/Scicos environment;
- Development of soft/hard real time control systems with the Scicos-HIL toolbox and the Scicos RTAI Code Generator.

Current and future developments in RTSS

- Development of new blocks;
- support for robot hands and closed-chain systems;

Further readings about RTSS

General information: `http://rtss.sourceforge.net/`

- Introduction and key features;
- software download and licensing information;
- support and contributions.

Development reference source: `http://sourceforge.net/apps/mediawiki/rtss/`

- Roadmap and updates about the status of development;
- technical documentation for developers and advanced users;
- notes about the compatibility among different Scilab versions.

References
Generic references

P.I. Corke.
A robotics toolbox for MATLAB.
IEEE Robotics and Automation Magazine, vol. 3, pp. 24–32, Mar. 1996.

A. Winkler and J. Suchý.
Novel joint space force guidance algorithm with laboratory robot system.
Proceedings of the 16th IFAC World Congress, 2005.

References

Technical Literature of the Pelican arm model treated in this presentation

R. Kelly, V. Santibáñez and A. Loría.
Control of Robot Manipulators in Joint Space.
Advanced Textbooks in Control and Signal Processing,
Springer-Verlag, London, 2005.

References

Technical Literature of the Manutec r3 robot model treated in this presentation

M. Otter and S. Türk.
The DFVLR Models 1 and 2 of the Manutec r3 Robot.
Technical Report DFVLR–Mitt.88–13, DLR, Institut für Robotik und Systemdynamik, Postfach 11 16, D–82234 Wessling, May 1988.

J. Franke and M. Otter.
The Manutec r3 Benchmark Models for the Dynamic Simulation of Robots.
Technical Report TR R101-93, DLR, Institut für Robotik und Systemdynamik, Postfach 11 16, D–82234 Wessling, March 1993.

References

Technical Literature of the Manutec r3 robot model treated in this presentation

P. Anell.
Modeling of multibody systems in Omola.
Master thesis ISRN LUTFD2/TFRT5516SE, Department of Automatic Control, Lund Institute of Technology, Lund, Sweden, Sep 1994.

M. Vukobratović et al.
Dynamics and Robust Control of Robot-environment Interaction.
World Scientific Publishing Co., Inc., River Edge, NJ, USA, 2009.

D. Katić and M. Vukobratović.
Intelligent Control of Robotic Systems.
Kluwer Academic Publishers, 2003.

References

Technical Literature of the other models and simulation scenarios presented in this work

F. Caccavale et al.
Experiments of kinematic control on a redundant robot manipulator with non-spherical wrist.
Laboratory Robotics and Automation, vol. 8, pp. 25–36, 1996.

L. Sciavicco and B. Siciliano.
Modelling and Control of Robot Manipulators.
Advanced Textbooks in Control and Signal Processing, London, UK: Springer-Verlag, 2nd ed., 2000.