ScicosLab/Scicos for robotics applications

M. Morelli

mmorelli@users.sourceforge.net

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

Training course: Simulation and automatic code generation for real time embedded systems using ScicosLab and Erika/Linux

Outline

RTSS–the Robotics Toolbox for Scilab/Scicos

MATLAB® Robotics Toolbox (MRT) as source of inspiration

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.

RTSS–the Robotics Toolbox for Scilab/Scicos

MATLAB® Robotics Toolbox (MRT) as source of inspiration

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.

RTSS–the Robotics Toolbox for Scilab/Scicos

MATLAB® Robotics Toolbox (MRT) as source of inspiration

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.

RTSS–the Robotics Toolbox for Scilab/Scicos

Milestone 2 (2010)

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.

RTSS–the Robotics Toolbox for Scilab/Scicos

Application areas

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.

RTSS–the Robotics Toolbox for Scilab/Scicos

Application areas

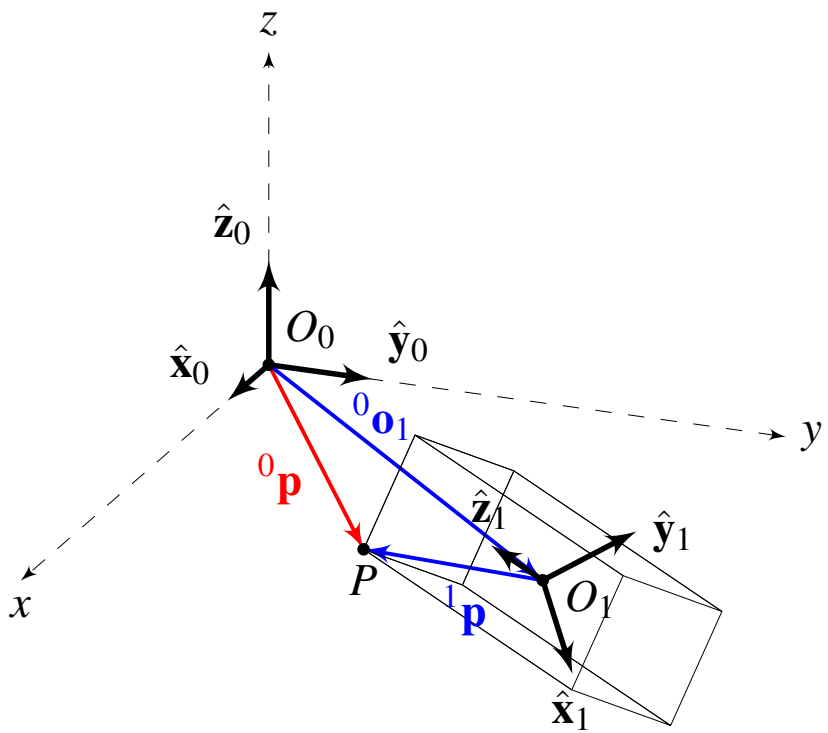
Potential applications in research

- 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

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

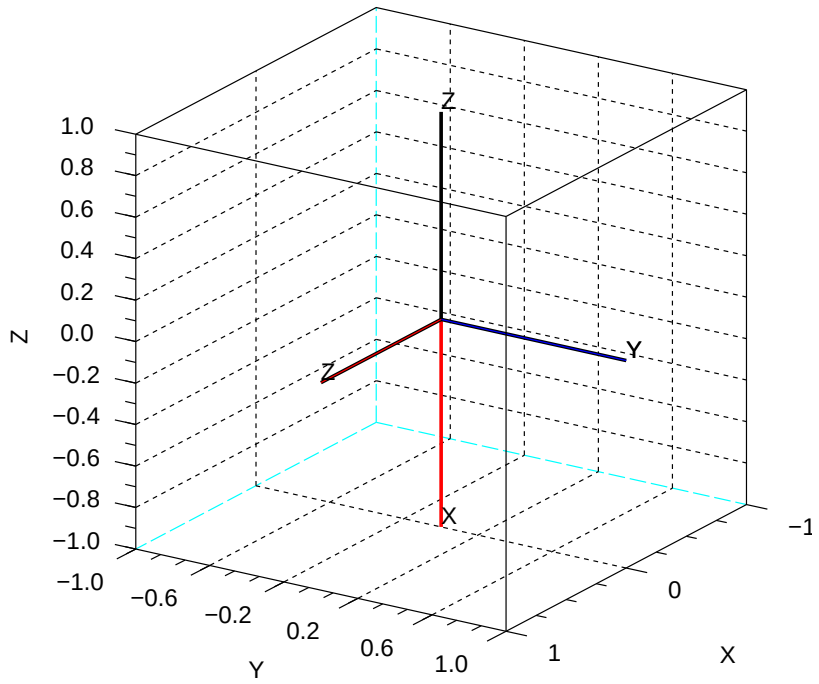
Example

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
```

Outline

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;
- fixed-base robots;
- open kinematic chains;
- arbitrary number of kinematic pairs (n).

Typical industrial manipulator – TX90, courtesy of Stäubli Group

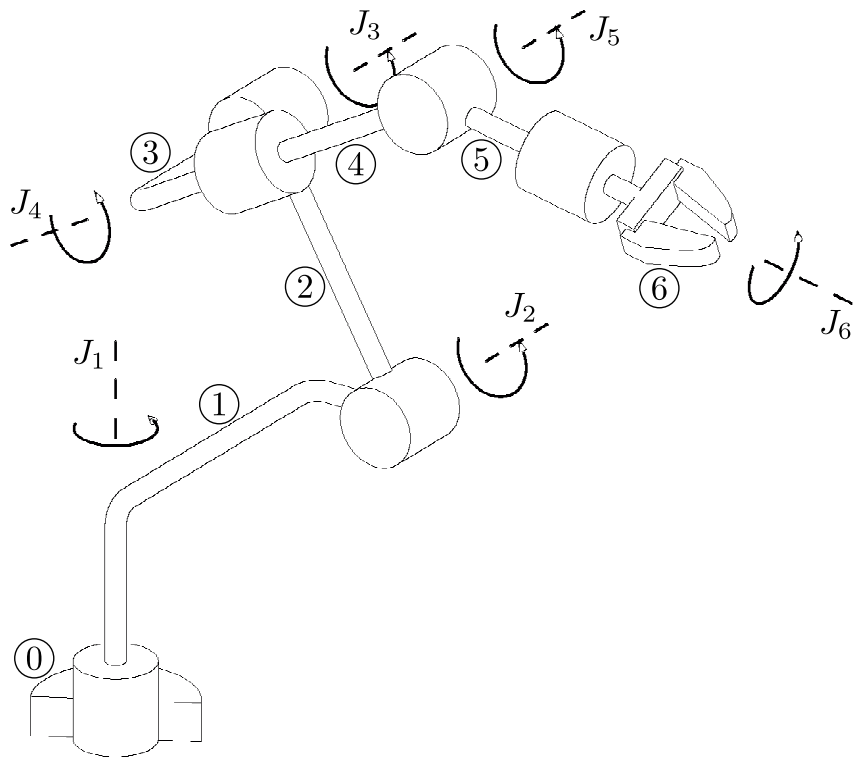
On the robot modelling capabilities of RTSS

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

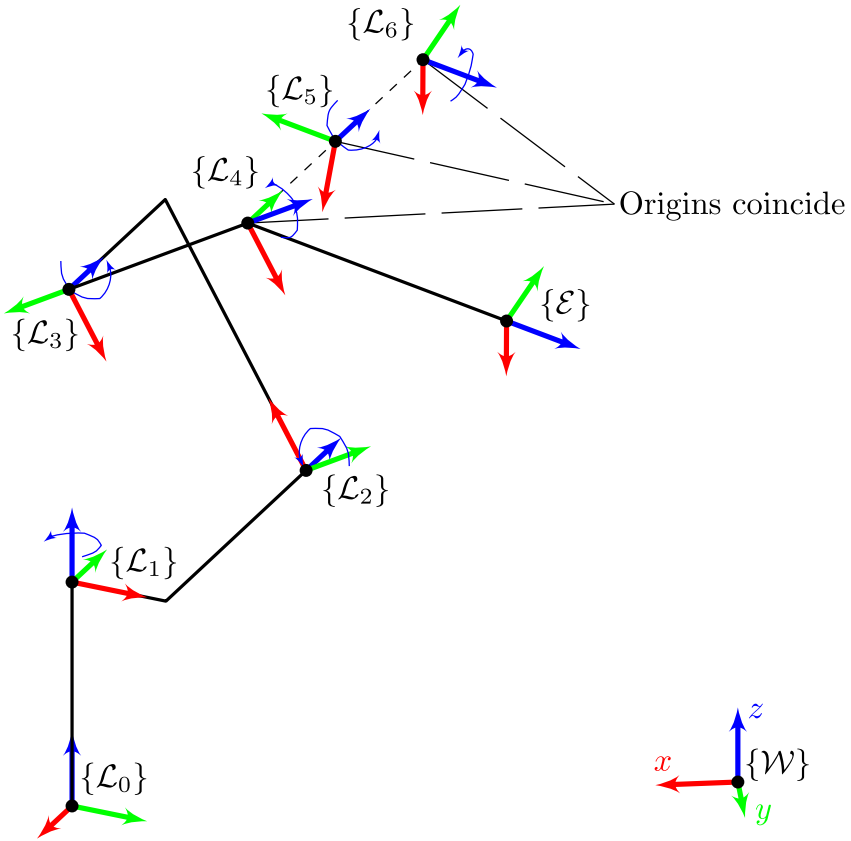
On the robot modelling capabilities of RTSS

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

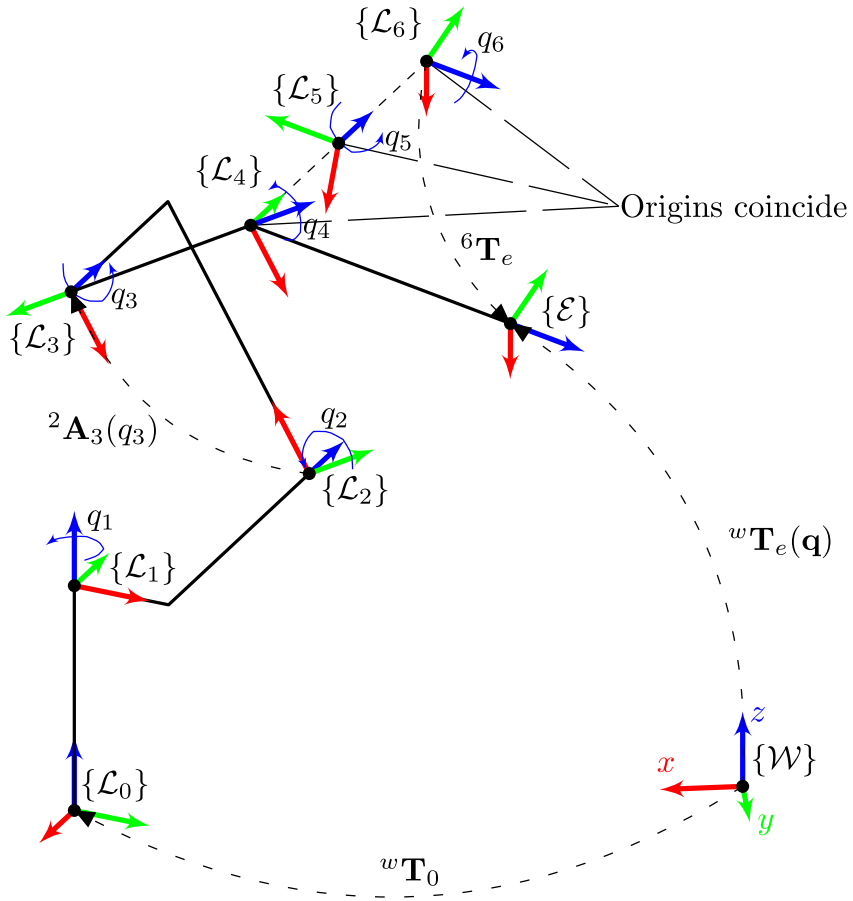
n-DOF manipulator modelling at a glance

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)

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

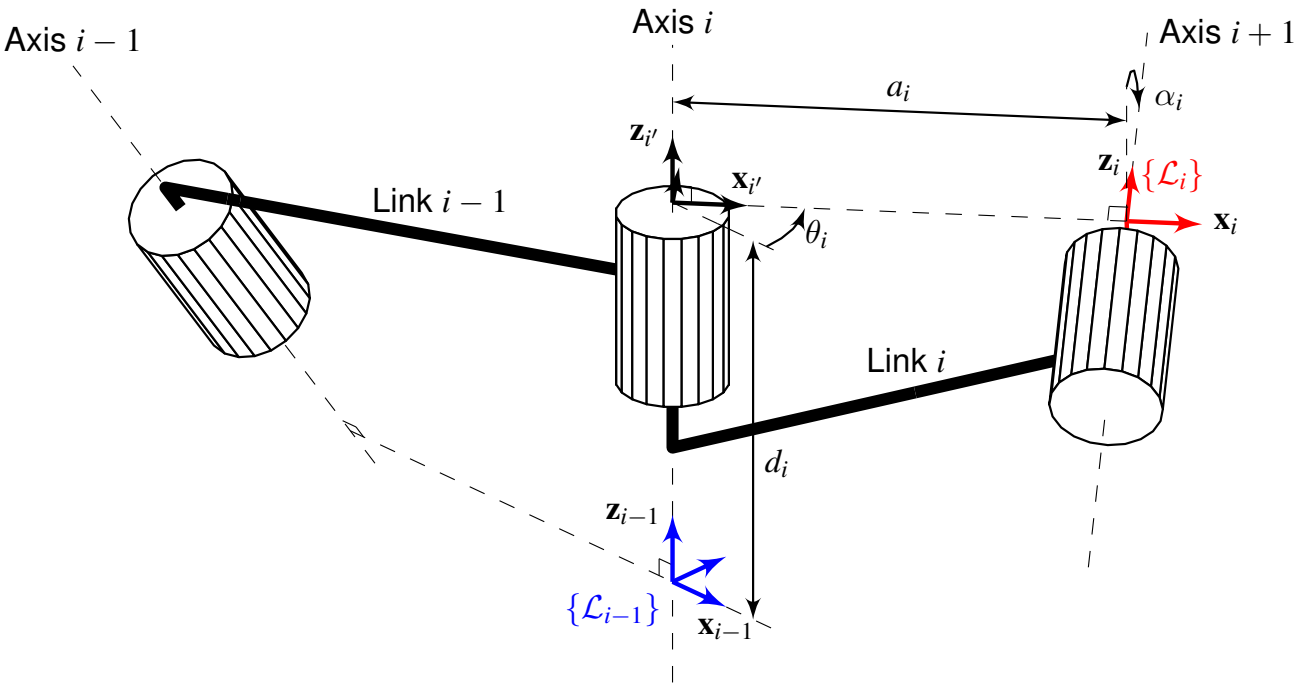
- Effective and coupling inertia torques;
- centrifugal, Coriolis and gravitational torques;
- viscous and static friction torques;
- robot-environment interaction torques.

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



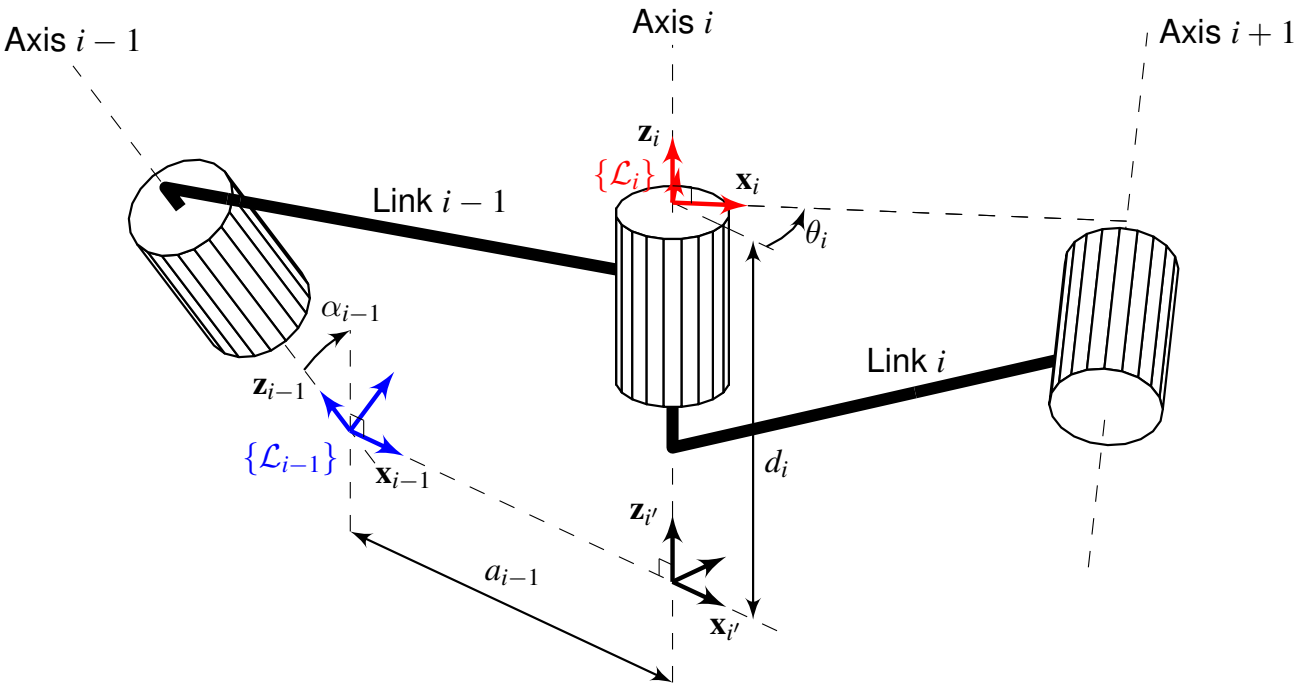
Standard DH frame assignment

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

Modelling and analysis of a 6-DOF industrial robot

The industrial robot Siemens Manutec r3

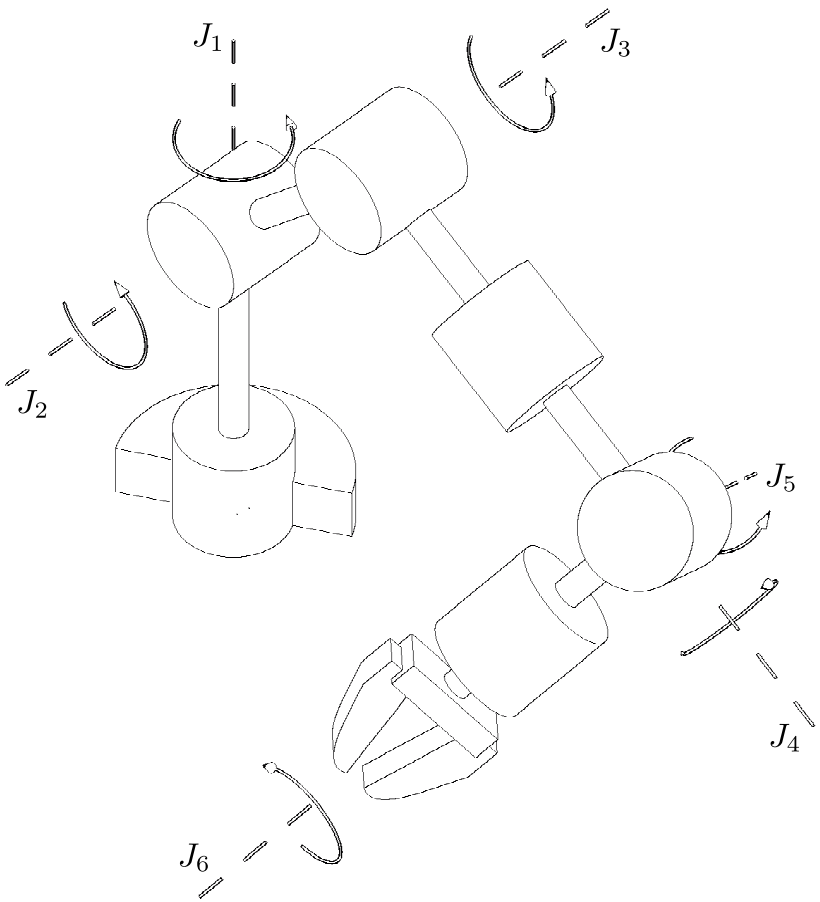
Robot description at a glance

- 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.

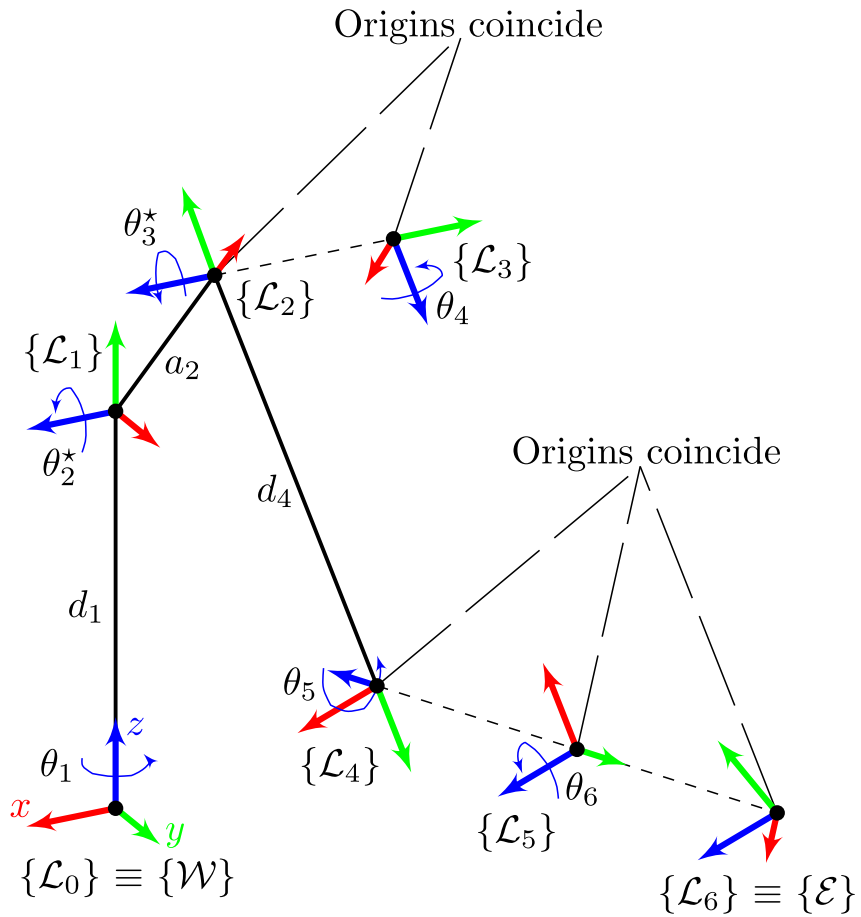
The robot Manutec r3 – [Winkler and Suchý, 2005]

Modelling and analysis of a 6-DOF industrial robot

A geometric model based on the standard Denavit-Hartenberg notation



Kinematic chain



Body frames

Modelling and analysis of a 6-DOF industrial robot

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 $\mathbf{q} = \mathbf{0}$):

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

Geometric model's code

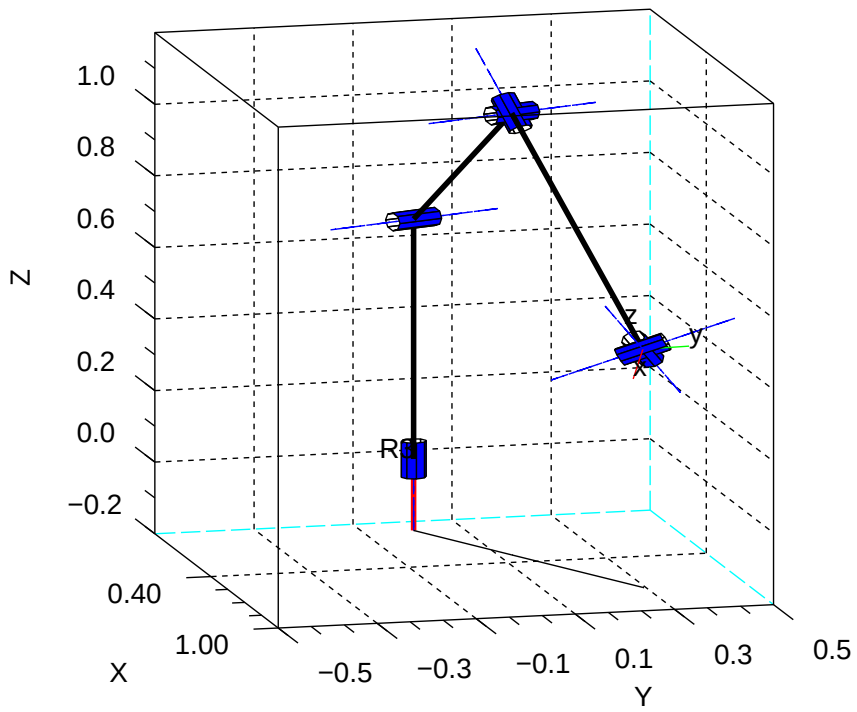
```
// geometric model of each joint-link pair
// alpha, A, theta, D, sigma (R/P)
dh = [..
 %pi/2,      0, 0,    0.67, 0;..
    0,    0.5, 0,       0, 0;..
-%pi/2,      0, 0,       0, 0;..
 %pi/2,      0, 0,    0.73, 0;..
-%pi/2,      0, 0,       0, 0;..
    0,       0, 0,       0, 0];

// creation of the robot model
r3 = rt_robot(dh, "r3", "Manutec");

// joint offsets matching the ref. config.
r3.offset = [0; %pi/2; -%pi/2; 0; 0; 0];
```

Modelling and analysis of a 6-DOF industrial robot

Geometry kinematics of the Manutec r3 robot

Positioning/orientation analysis

```
// 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.

// plot the robot posture
ws = [-0.1, 1.0, -0.5, 0.5, -0.2, 1.2];
rt_plot(r3, q, "base", "workspace", ws);

// 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
```



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

Modelling and 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
```

Modelling and 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

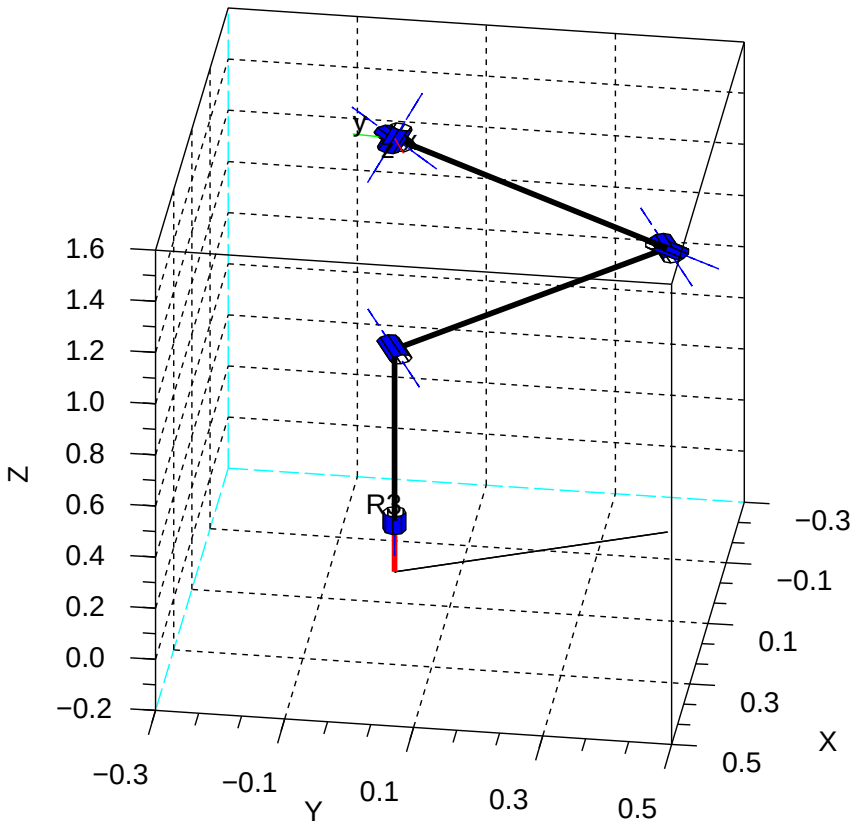
- Boundary, the robot is stretched/retracted;
- internal, occur inside the reachable workspace.

Modelling and 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

Modelling and 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.

Modelling and analysis of a 6-DOF industrial robot

The dynamics of the Manutec r3 robot: standard vs modified DH frame assignments

Question

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

Parameter	Is it convention dependent?
Link mass	No
Distance to link COM	Yes
Inertia tensor about link COM	Yes
Actuator/transmission	No

Dependence of physical parameters on the frame assignments

Modelling and analysis of a 6-DOF industrial robot

The dynamics of the Manutec r3 robot: inertial parameters

Link	Notation	Value
1	m_1	20
2	m_2	56.5
3	m_3	26.4
4	m_4	28.7
5	m_5	5.2
6	m_6	15

Link mass, convention-independent (kg)

Link	Values		
	r_{i_x}	r_{i_y}	r_{i_z}
1	0	−0.172	0
2	−0.295	0	0.172
3	0	−0.064	−0.034
4	0	−0.410	0
5	0	0	0.023
6	0	0	−0.02

Distance to the link COM, standard-DH based (m)

Modelling and analysis of a 6-DOF industrial robot

The dynamics of the Manutec r3 robot: inertial parameters

Link	Moments of inertia			Products of inertia		
	$I_{i_{xx}}$	$I_{i_{yy}}$	$I_{i_{zz}}$	$I_{i_{xy}}$	$I_{i_{yz}}$	$I_{i_{xz}}$
1	0	1.16	0	0	0	0
2	2.58	2.73	0.64	0	0	0
3	0.279	0.413	0.24	0	0	0
4	1.67	1.67	0.81	0	0	0
5	1.25	1.53	0.81	0	0	0
6	0	0	0.0002	0	0	0

Inertia rel. to COM, standard-DH based ($\mathrm{kg\ m^2}$)

Modelling and analysis of a 6-DOF industrial robot

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

Description	Notation	Values Arm 1	Arm 2	Arm 3	Units
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 Arm 4	Arm 5	Arm 6	Units
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

Modelling and analysis of a 6-DOF industrial robot

Dynamic model of the Manutec r3 based on the standard DH frame assignments

Inertial parameters

```
// 3-element row vector of zeros
z = zeros(1,3);

// matrix of inertial parameters
inert_data = [..
m1, r1x, r1y, r1z, I1xx, I1yy, I1zz, z;..
m2, r2x, r2y, r2z, I2xx, I2yy, I2zz, z;..
m3, r3x, r3y, r3z, I3xx, I3yy, I3zz, z;..
m4, r4x, r4y, r4z, I4xx, I4yy, I4zz, z;..
m5, r5x, r5y, r5z, I5xx, I5yy, I5zz, z;..
m6, r6x, r6y, r6z, I6xx, I6yy, I6zz, z];

// matrix of actuator/transmission params.
act_data = [..
Jm1, G1, B1, Tc1, -Tc1;..
Jm2, G2, B2, Tc2, -Tc2;..
Jm3, G3, B3, Tc3, -Tc3;..
Jm4, G4, B4, Tc4, -Tc4;..
Jm5, G5, B5, Tc5, -Tc5;..
Jm6, G6, B6, Tc6, -Tc6];
```

Dynamic model

```
// dynamics matrix
dyn = [dh, inert_data, act_data];

// robot model & joint offsets
r3 = rt_robot(dyn, "r3", "Manutec");
r3.offset = [0; %pi/2; -%pi/2; 0; 0; 0];

// first link in detail
rt_showlink(r3.links(1)),

// ... kinematic data displayed here ...
 m     = 20

!r     =  0          !
!                    !
!         -0.172     !
!                    !
!         0          !

// ... further model data disp. here ...

!Tc    =  0.4 -0.4 !
```

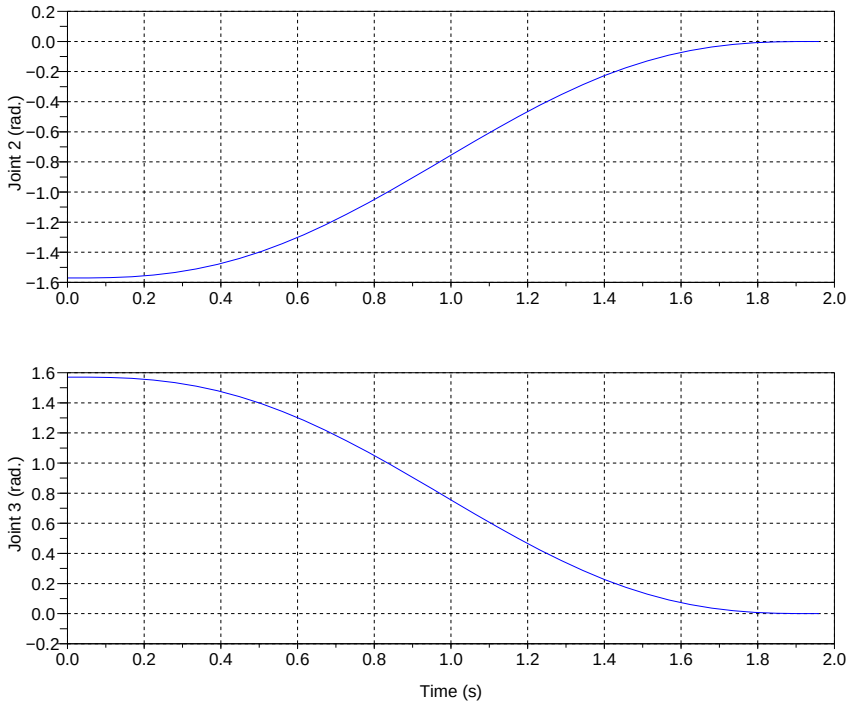
Modelling and analysis of a 6-DOF industrial robot

Basics of joint-space trajectory generation

A a number of toolbox functions can operate on trajectories.

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);
```



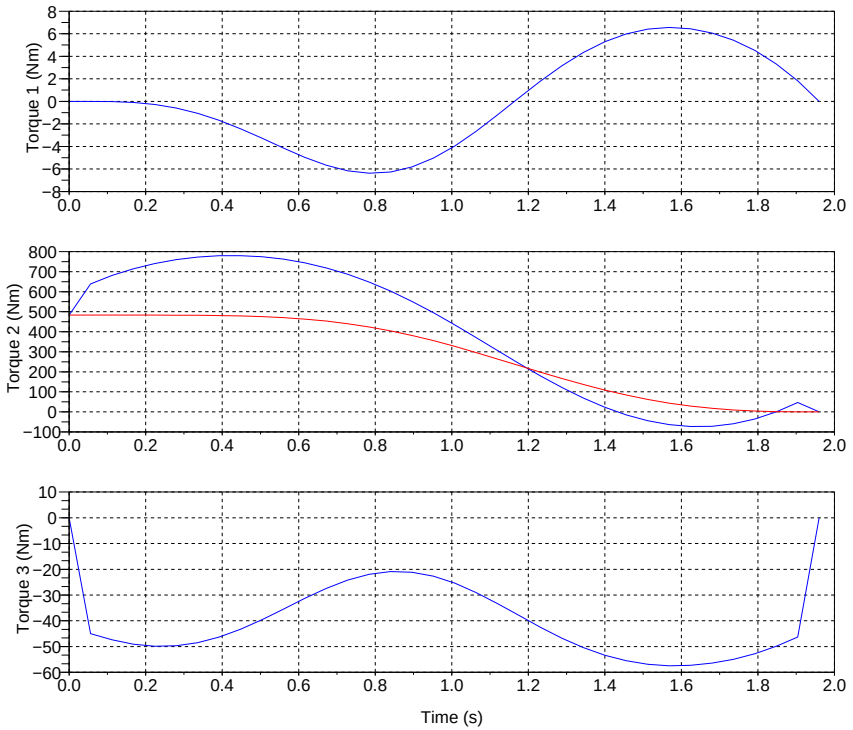
Joint space trajectory

Modelling and analysis of a 6-DOF industrial robot

Analysis of 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)

Modelling and analysis of a 6-DOF industrial robot

Analysis of the joint inertia for the Manutec r3 robot

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

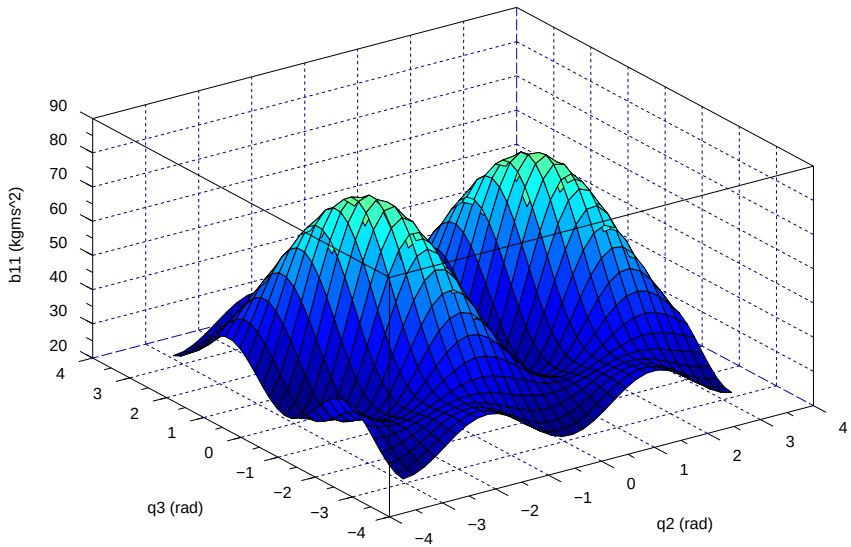
This results in significant variations of joint inertia, most important in joint 1, where the total inertia for the horizontal, loaded arm is more than three times higher than for the vertical arm.

Modelling and analysis of a 6-DOF industrial robot

Analysis of the joint inertia for the 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

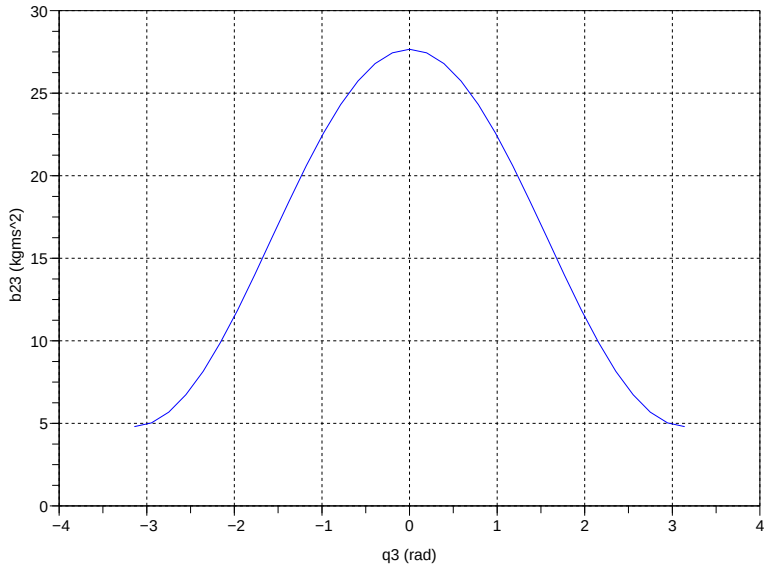
Modelling and analysis of a 6-DOF industrial robot

Analysis of the joint inertia for the Manutec r3 robot

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

Inertial coupling between joint 2 and 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

Modelling and analysis of a 6-DOF industrial robot

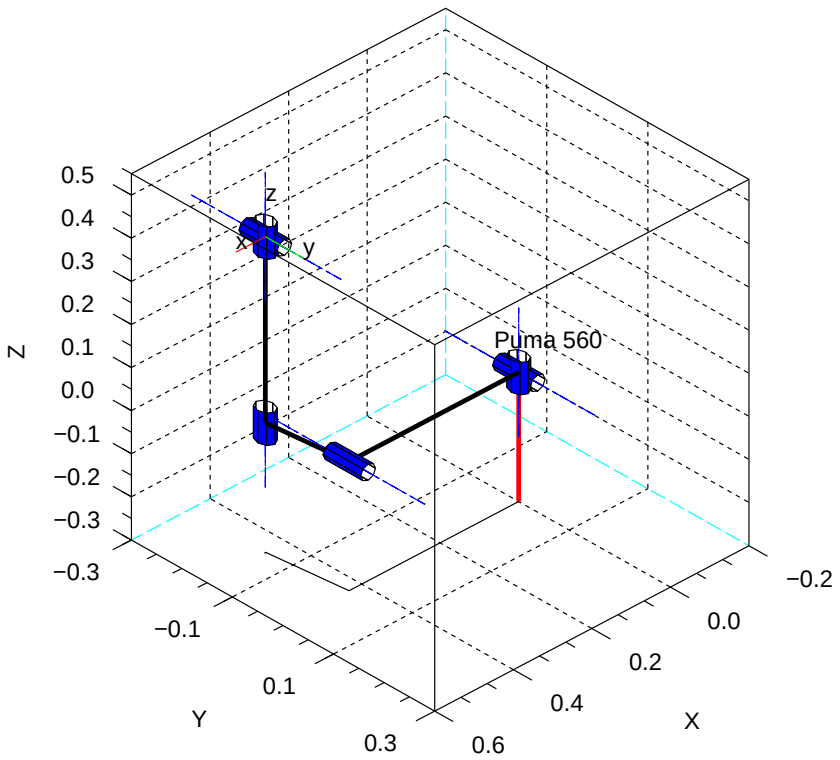
Analysis of the joint inertia for the Manutec r3 robot

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.

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

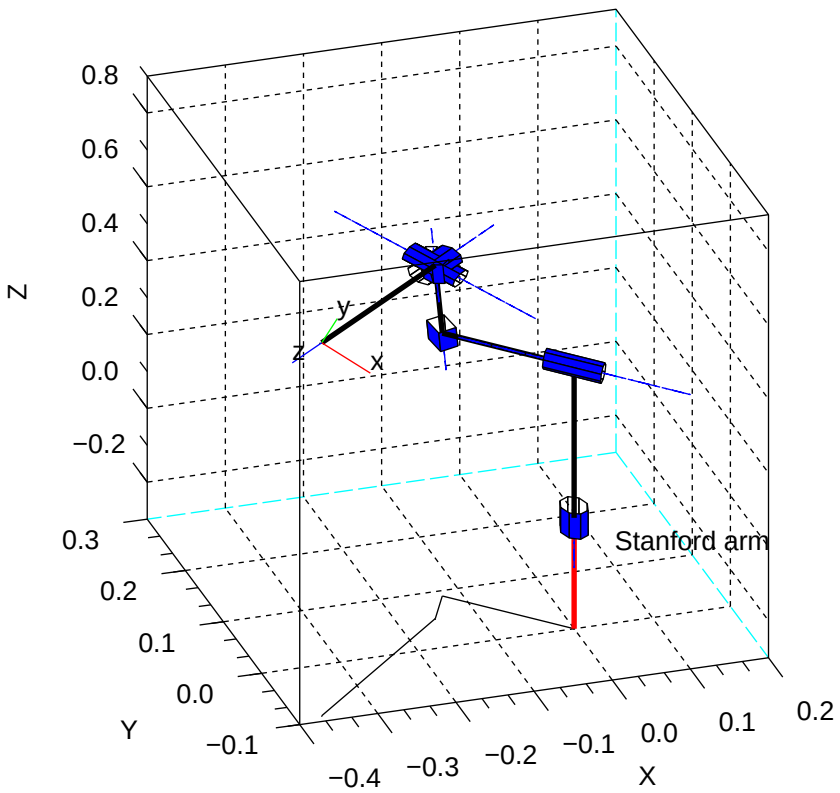
- Kinematic and dynamic data;
- Standard and modified DH-based models;
- quantities in standard SI units.

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.

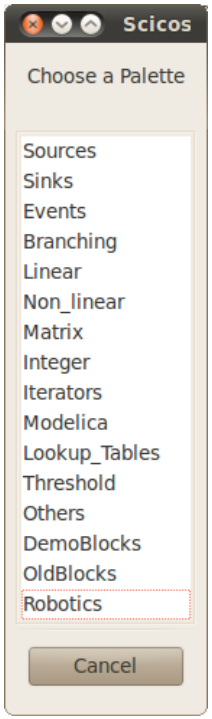
Create a Stanford manipulator

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

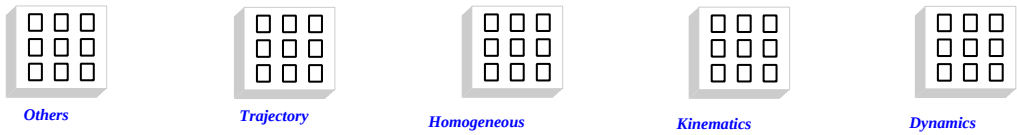
Outline

The Robotics palette

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



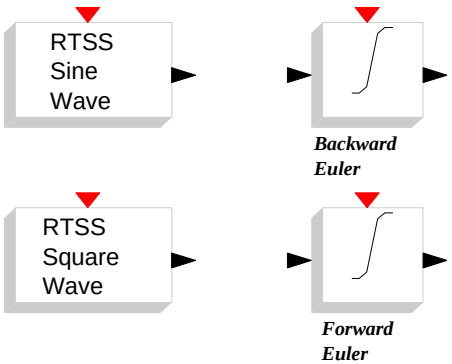
List of the available palettes after the installation of RTSS



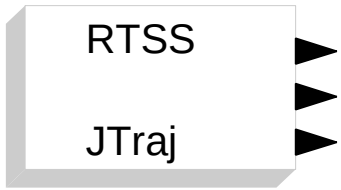
Blocks' categories available with RTSS-1.0.0b1

The Robotics palette

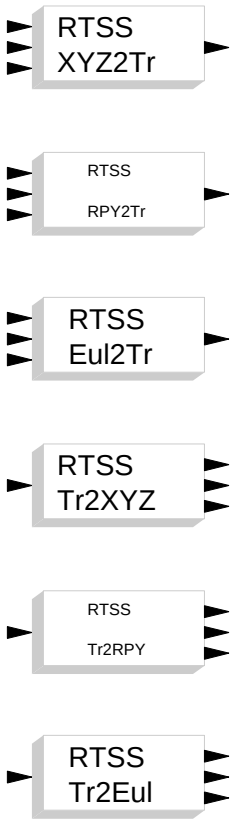
A library of ready-to-use blocks for robotics simulations



Category "Others"



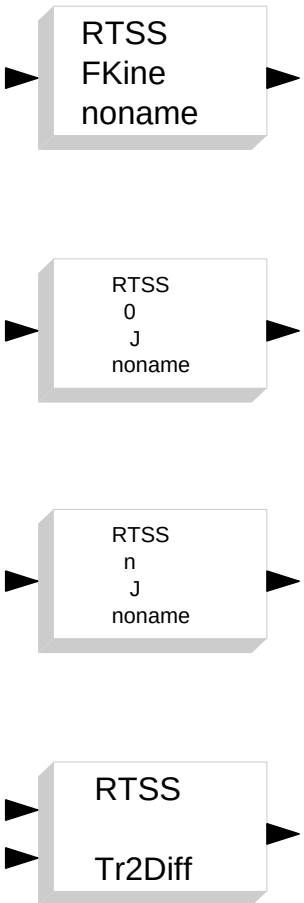
Category "Trajectory"



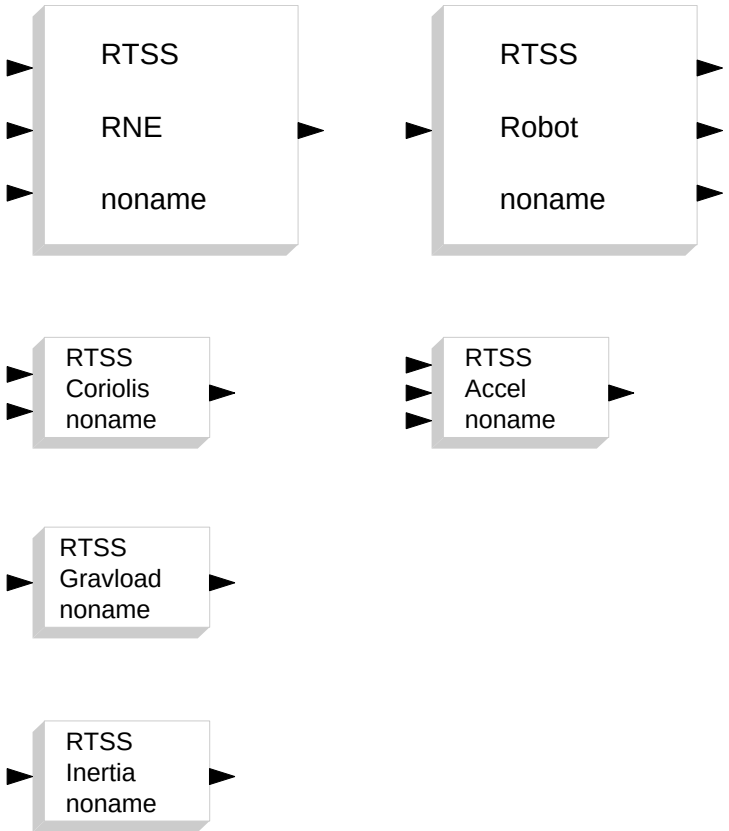
Category "Homogeneous"

The Robotics palette

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



Category "Kinematics"

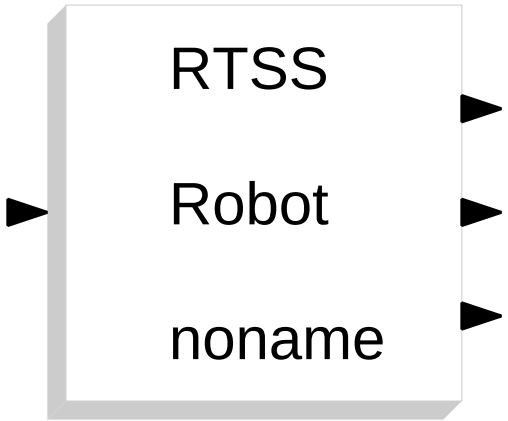


Category "Dynamics"

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)

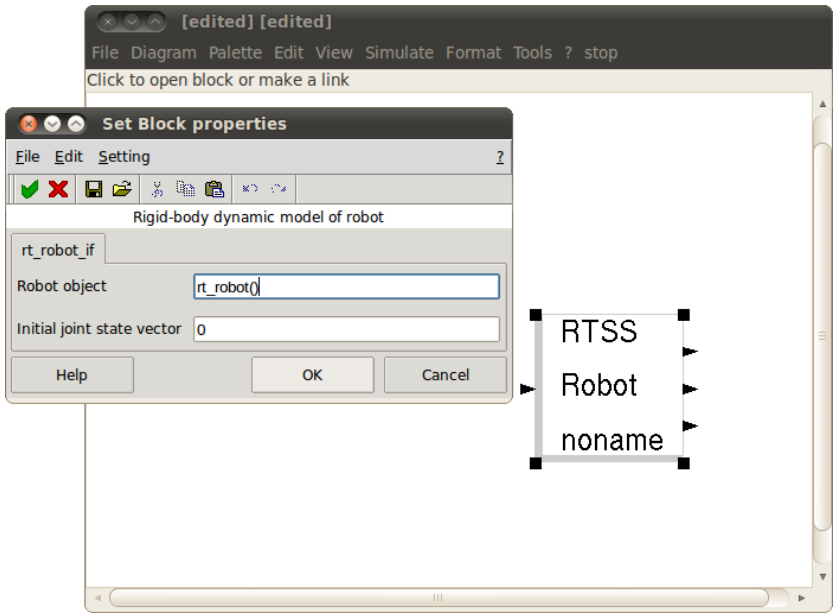
- The user must specify the robot model to be simulated as block parameter;
- the robot model must be a symbolic parameter defined in the context of the diagram.

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

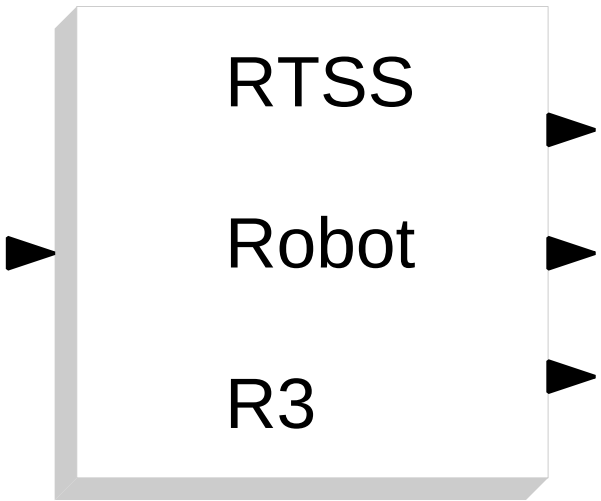
```
// robot model definition
// (pseudocode)
r3 = rt_robot( ... );

// 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



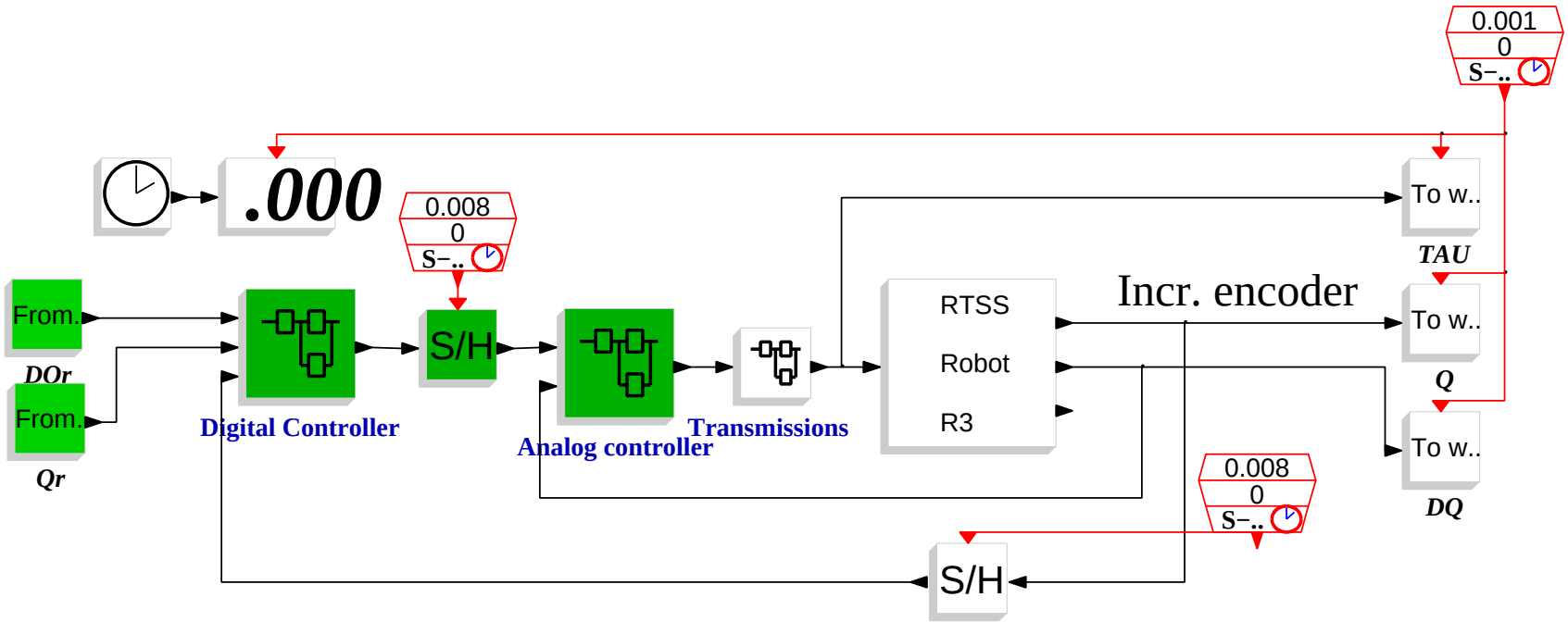
FDB ready to simulate the forward dynamics of the Manutec r3

Outline

Model-based centralized controller for tracking

Limitations of the independent joint control

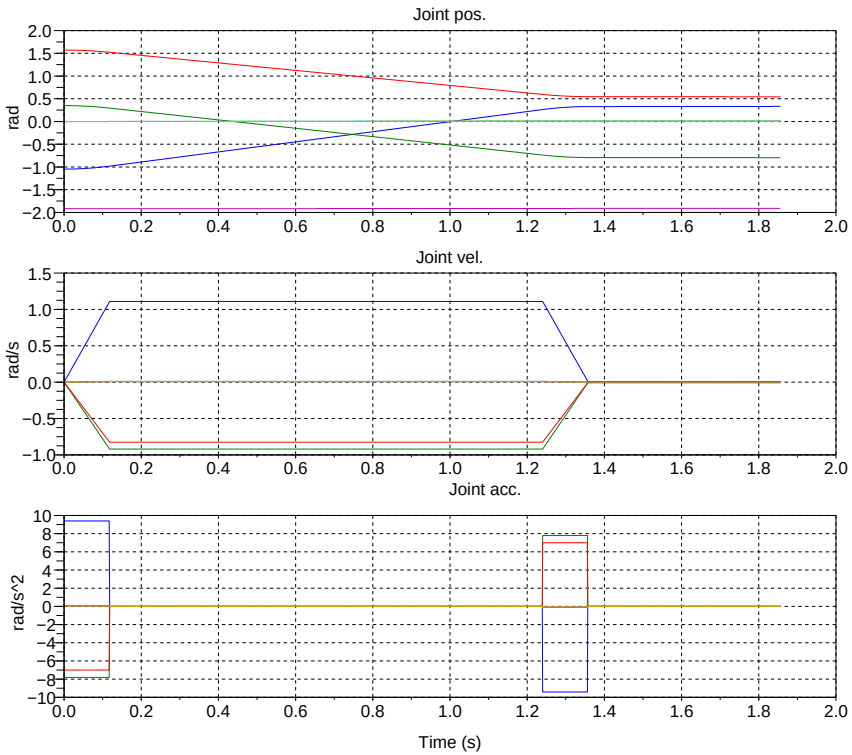
Independent joint control approach can be followed in case of limited performance in terms of required velocities and accelerations.



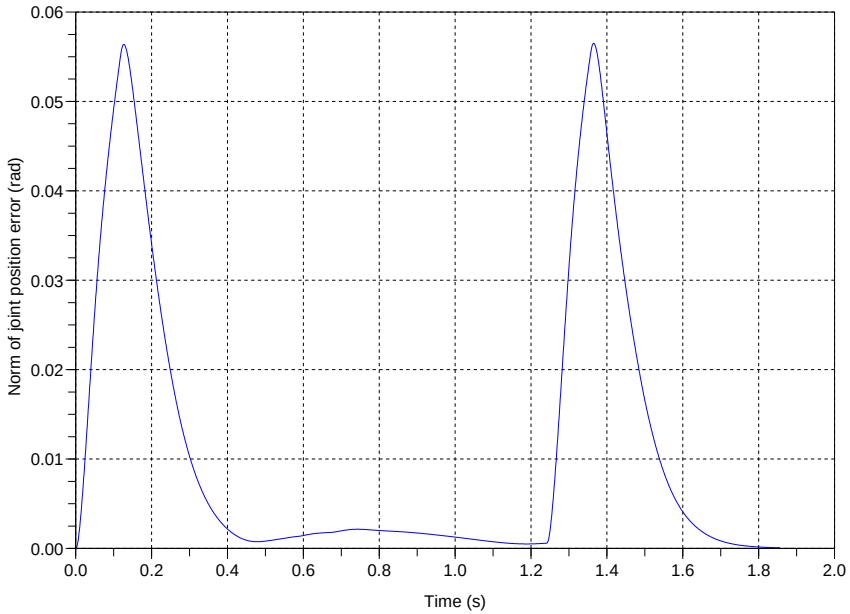
A case study: the Manutec r3 independent joint control system

Model-based centralized controller for tracking

Limitations of the independent joint control



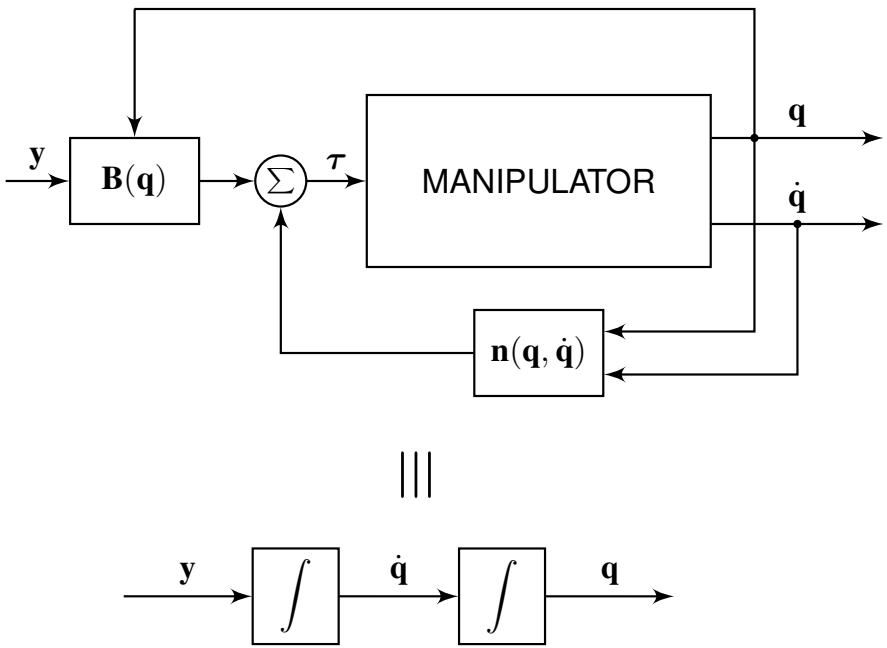
Time history of the given joint trajectory



Norm of joint position error

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

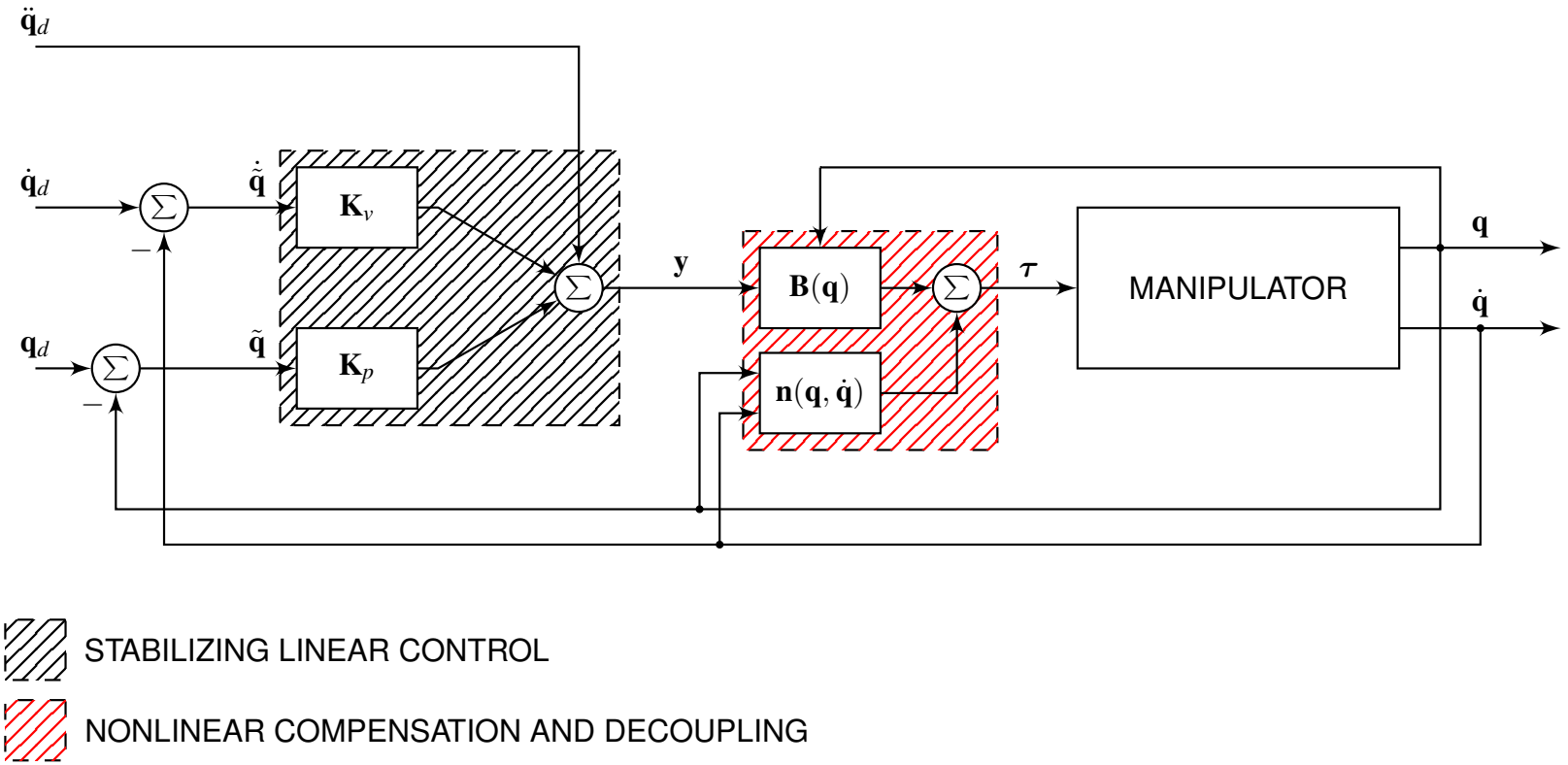
$$\ddot{\tilde{\mathbf{q}}} + \mathbf{K}_v \dot{\tilde{\mathbf{q}}} + \mathbf{K}_p \tilde{\mathbf{q}} = \mathbf{0}$$

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.

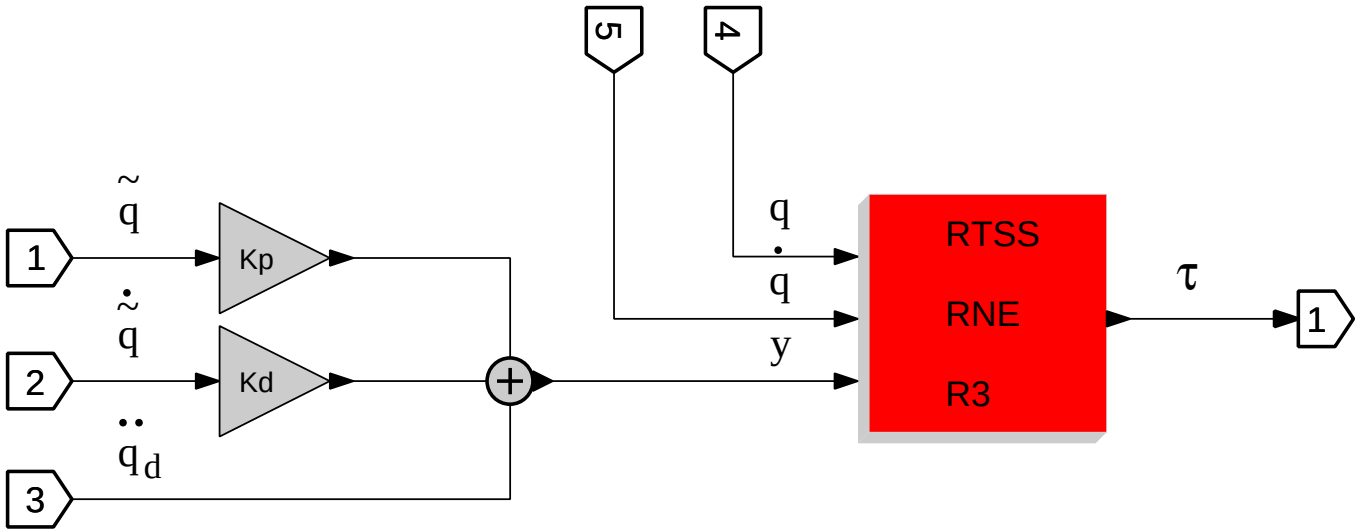


Block scheme of joint space inverse dynamics control

Model-based centralized controller for tracking

Scicos implementation of the inverse dynamics controller for the Manutec r3

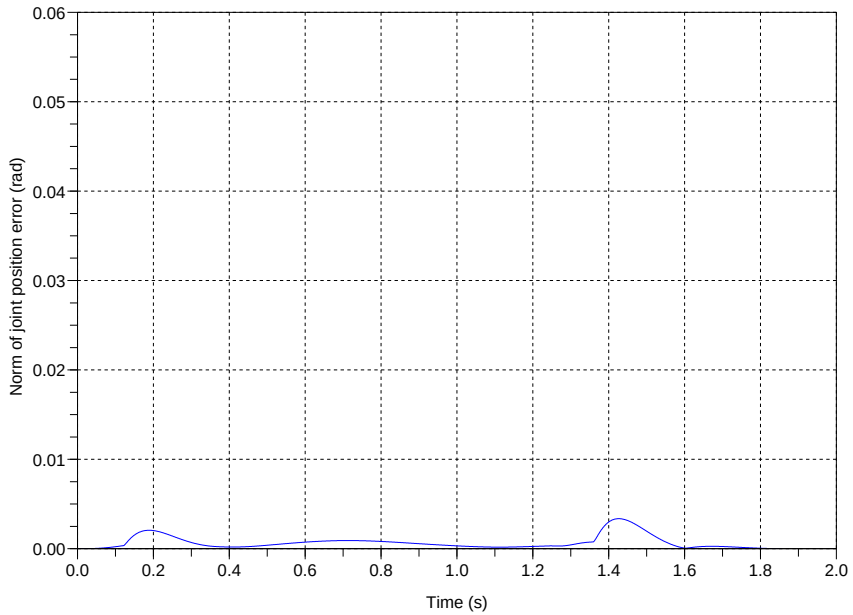
Key component: the inverse dynamics block (IDB) of the `Robotics` palette.



Inverse dynamics controller for the Manutec r3 robot

Model-based centralized controller for tracking

Analysis of simulation results



Norm of joint position error (scale is uniform to the figure above to allow a direct comparison of results)

Inverse dynamics controller design

Implementation and robustness issues

Pratical issues of the inverse dynamics control

- Parameters of the dynamical model must be accurately known;
- control input must be computed in real-time.

In the case of imperfect compensation, the control vector τ should be expressed as

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

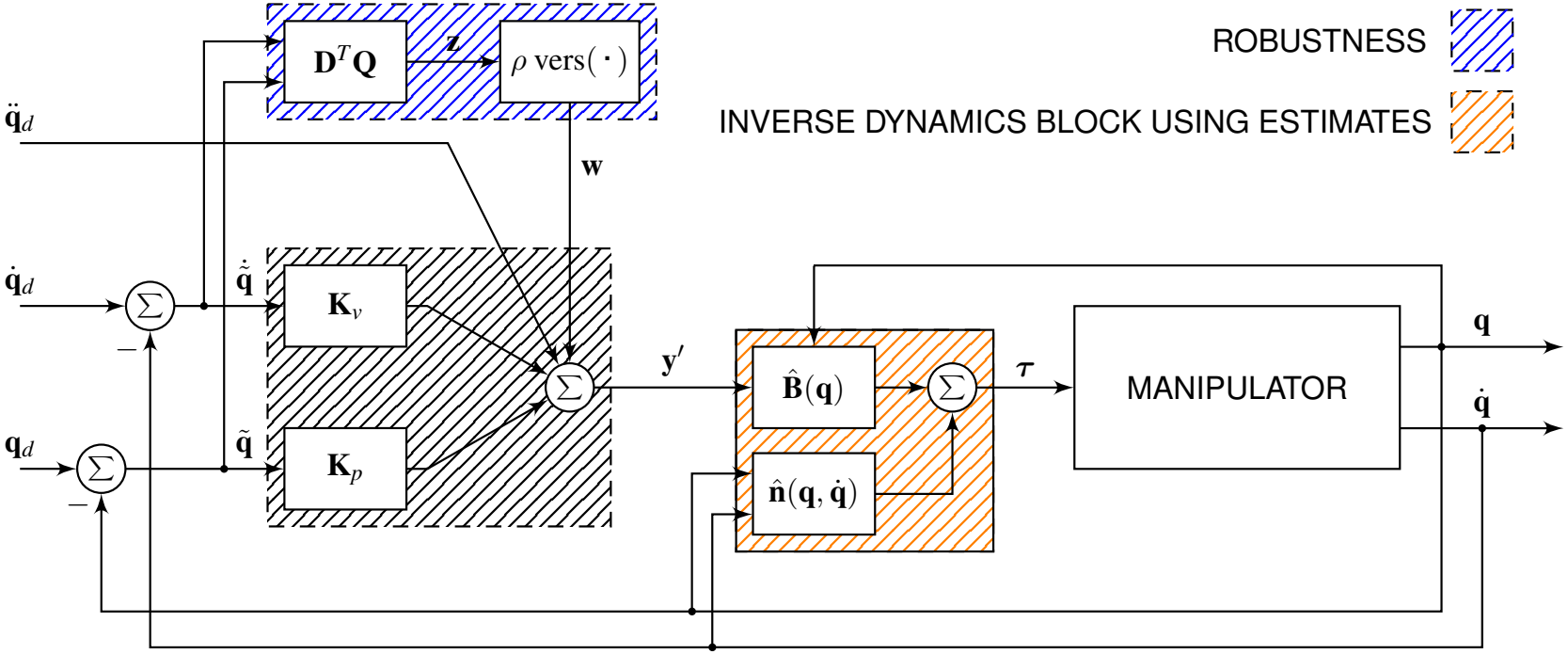
where

- $\hat{\mathbf{B}}$ and $\hat{\mathbf{n}}$ represent the estimates of $\mathbf{B}$ and $\mathbf{n}$,
- $\mathbf{y}' = \mathbf{y} + \mathbf{w}$ provides robustness to the control system.

Inverse dynamics controller design

Adding robustness to the inverse dynamics control system

$\mathbf{w}$ must guarantee robustness to the uncertainty described by $\hat{\mathbf{B}}$ and $\hat{\mathbf{n}}$.



Block scheme of joint space robust control

Inverse dynamics controller design

Including model uncertainty in the inverse dynamics controller

Estimates $\hat{\mathbf{B}}$ and $\hat{\mathbf{n}}$ can be modeled by using the function `rt_perturb()`.

How it works

It takes two inputs:

- Robot object (`rob`);
- perturb. percentage (`pp`).

It randomly modifies `rob` link masses and inertias in function of `pp`.

How to use it

- In the context of the diagram, “perturb” an existing robot;
- specify the modified robot as IDB block parameter.

Inverse dynamics controller design

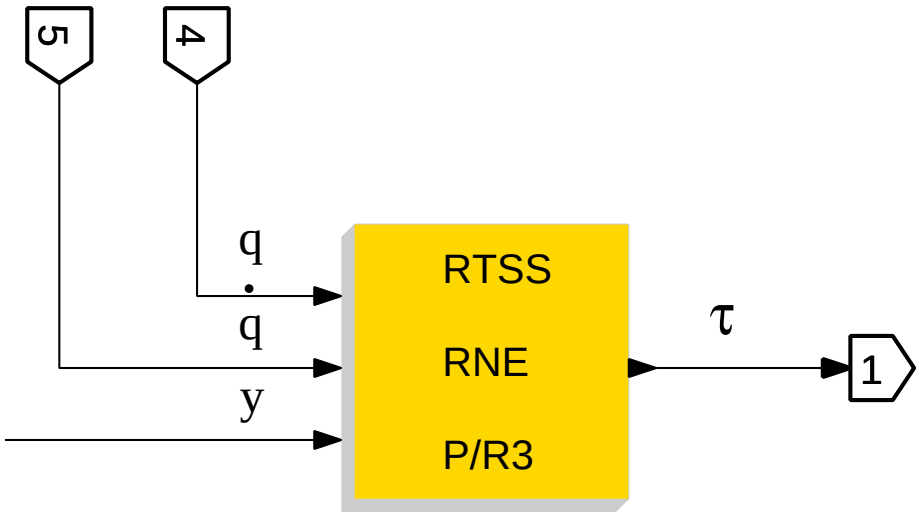
Including model uncertainty in the Manutec r3 inverse dynamics controller

Example

Perturb r3's link masses and inertias with a 25% disturb.

Scilab script in the context of the diagram

```
// include model uncertainty
r3p = rt_perturb(r3, 0.25);

// Other symbolic parameters
// ...
```



IDB with a perturbed model of the r3 robot as block parameter

First-order closed-loop inverse kinematics

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

Considerations regarding all the above control schemes

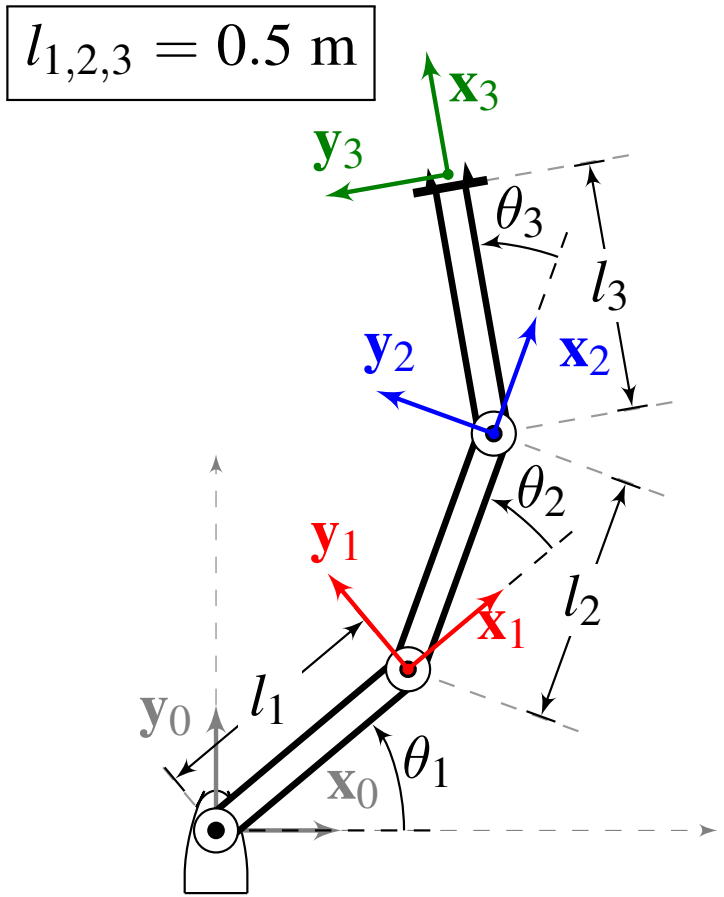
- Vectors $\mathbf{q}_d$, $\dot{\mathbf{q}}_d$ and $\ddot{\mathbf{q}}_d$ were always assumed available;
- joint space references were computed from two joint coordinate poses directly specified by the user.

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



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:

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

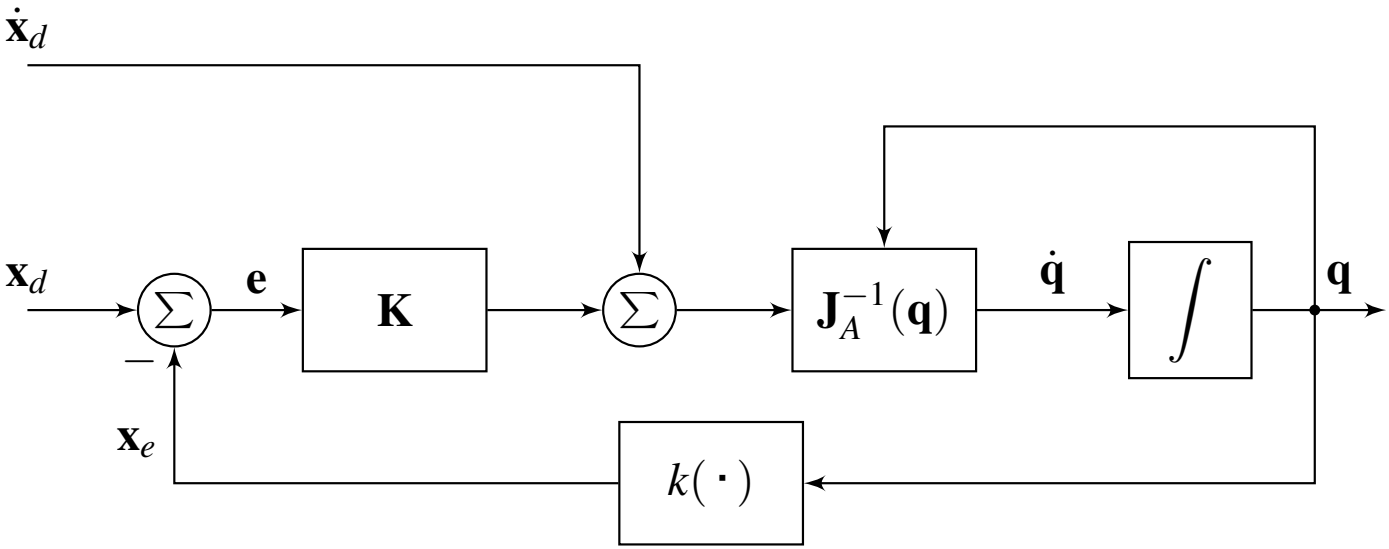
- forward Euler numerical integration scheme ($\Delta t = 1$ ms);
- final simulation time $t_{end} = 5$ s.

First-order closed-loop inverse kinematics

The Jacobian inverse algorithm

The Jacobian inverse algorithm integrates the joint velocity vector

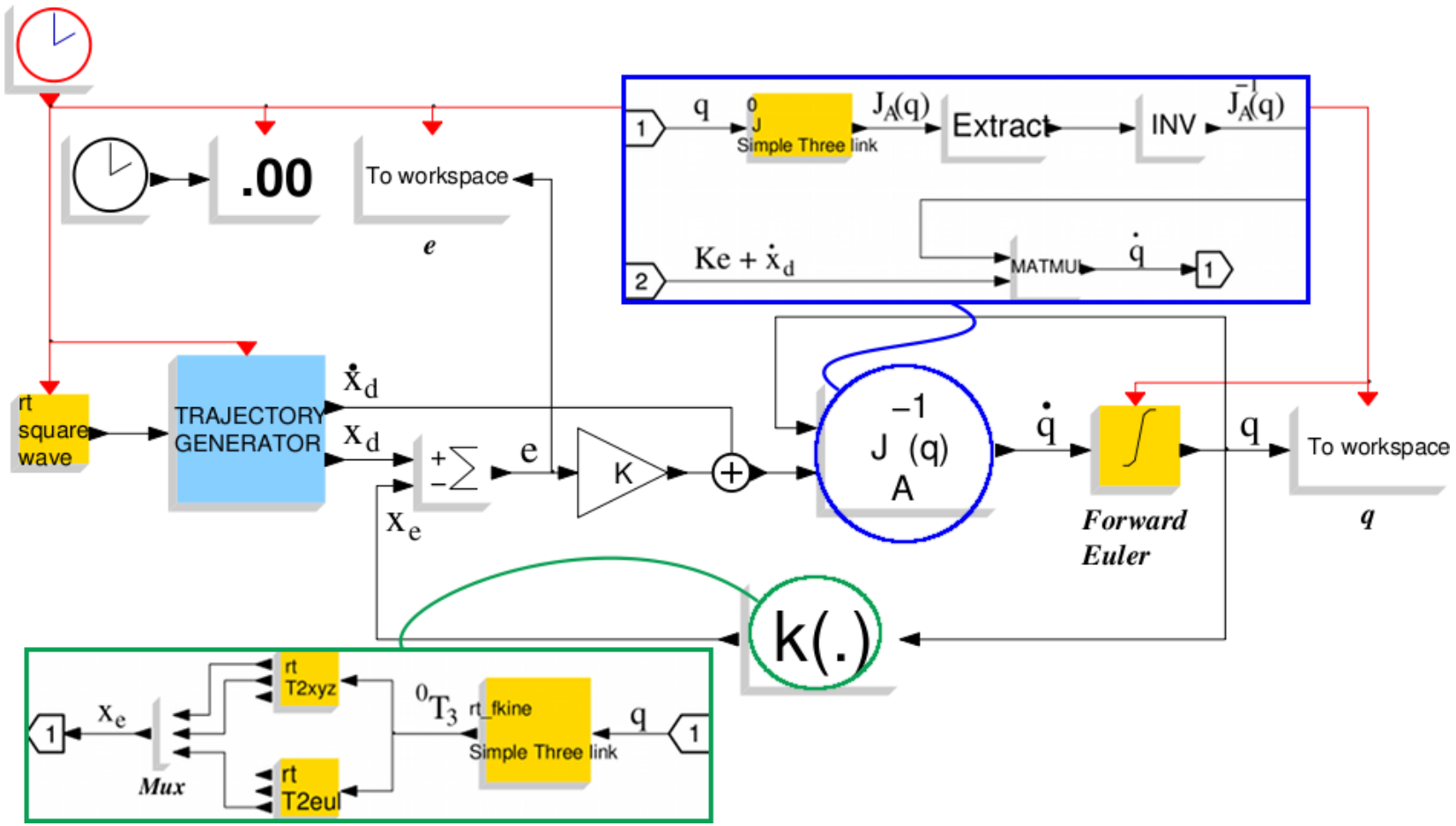
$$\dot{\mathbf{q}} = \mathbf{J}_A^{-1}(\mathbf{q})(\dot{\mathbf{x}}_d + \mathbf{K}\mathbf{e})$$



Jacobian inverse algorithm

First-order closed-loop inverse kinematics

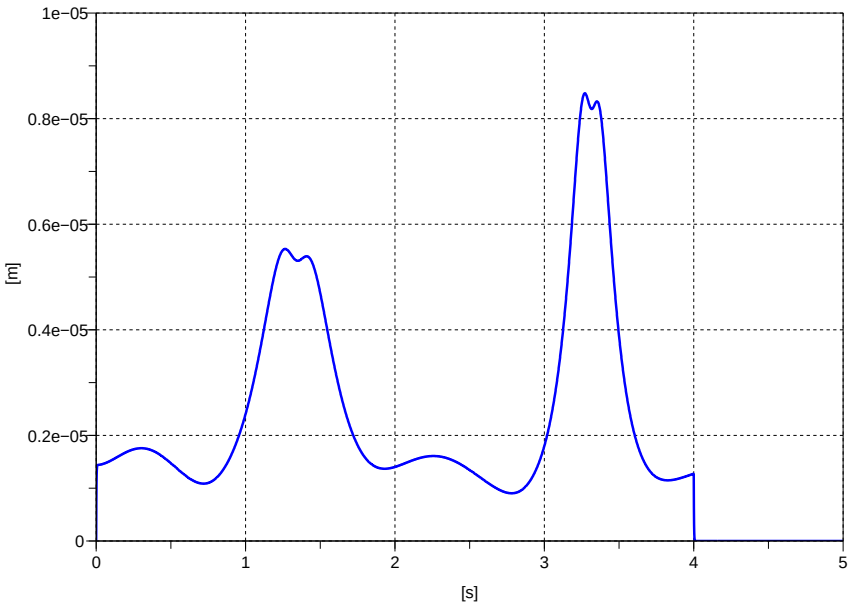
Scicos block diagram for the Jacobian inverse algorithm



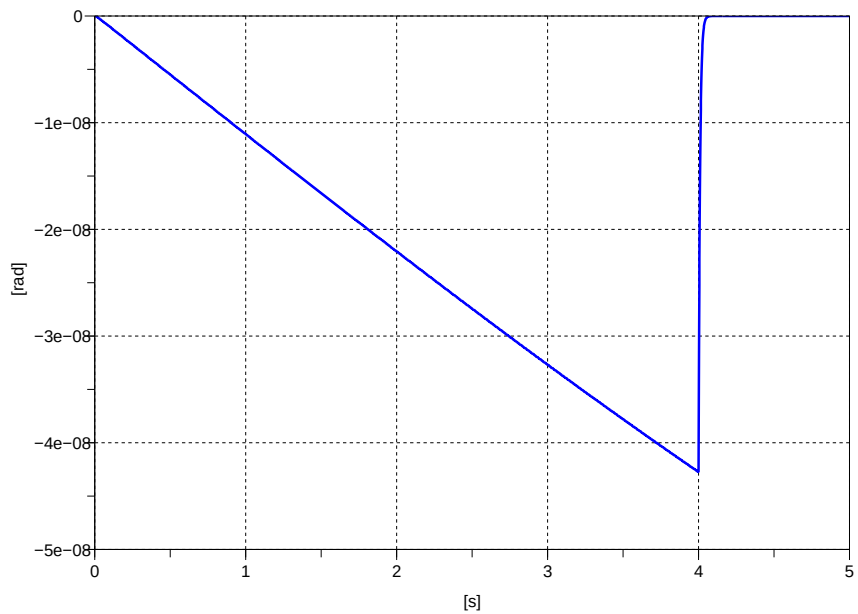
The Jacobian inverse algorithm and the blocks $\mathbf{J}_A^{-1}(\mathbf{q})$ and $k(\cdot)$

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

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

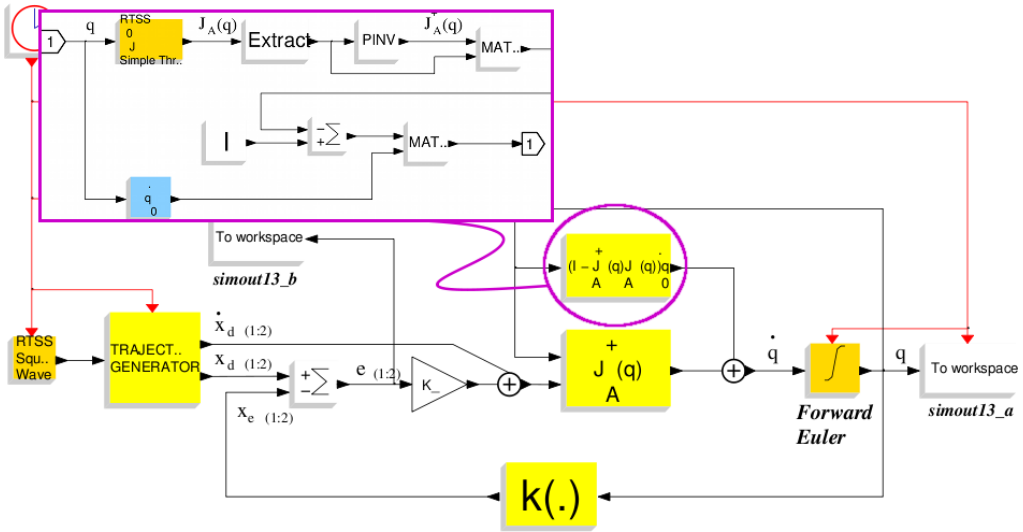
- maximizing the distance;
- q_i current joint angle;
- q_{iM} maximum joint limit;
- q_{im} minimum joint limit;
- $\bar{q}_i$ middle value.

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

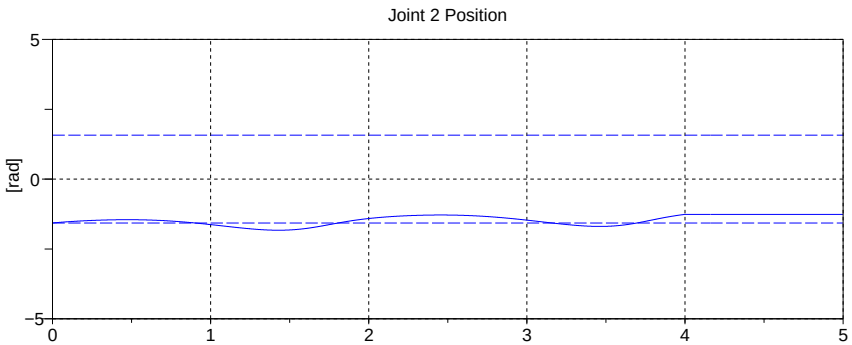
Computational code of block $\dot{q}_0$

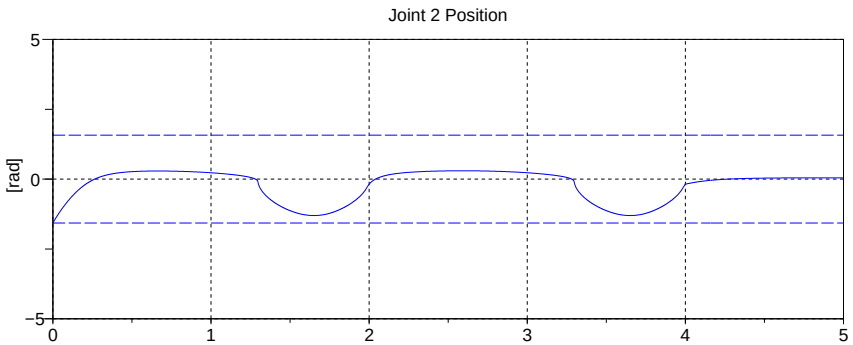
```
double * y, * u;
double pi = 3.1415927;

// 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);
```

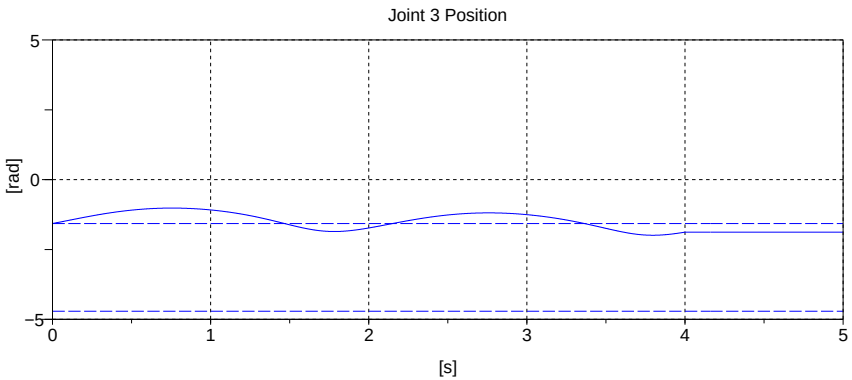
First-order closed-loop inverse kinematics

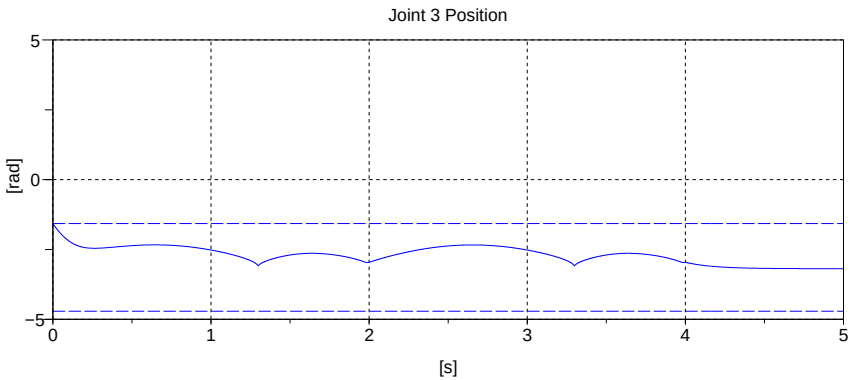
Analysis of simulation results: time history of the joint trajectories

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









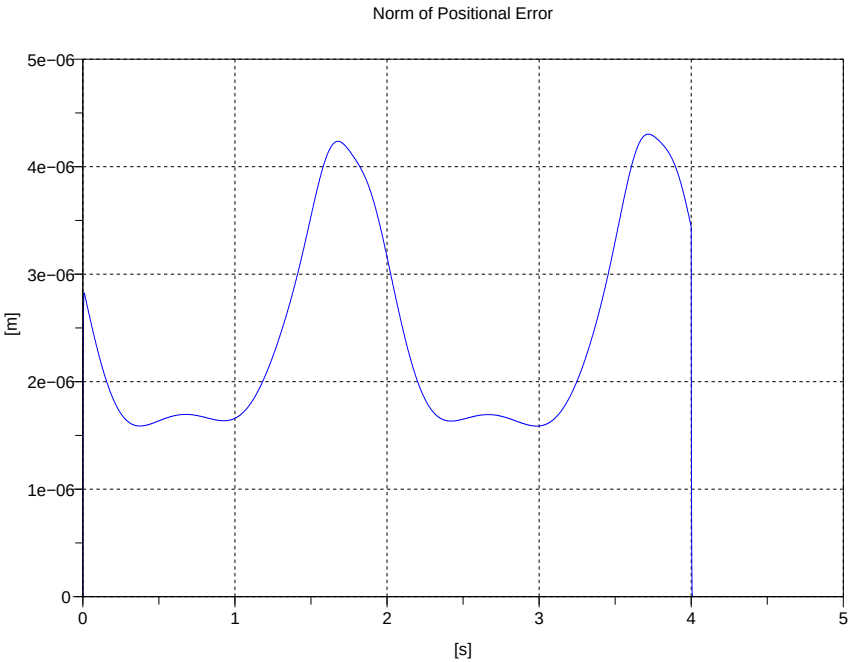
Without redundancy resolution

With utilization of redundancy

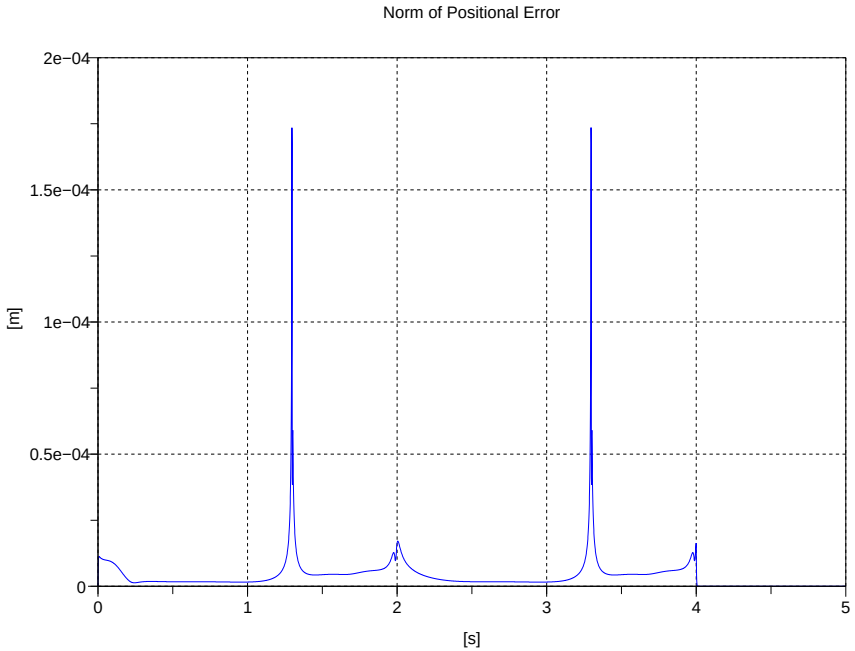
First-order closed-loop inverse kinematics

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

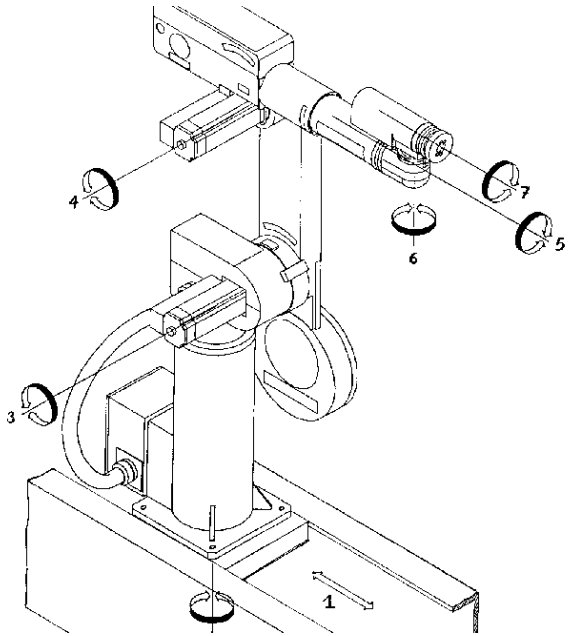


With utilization of redundancy

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.



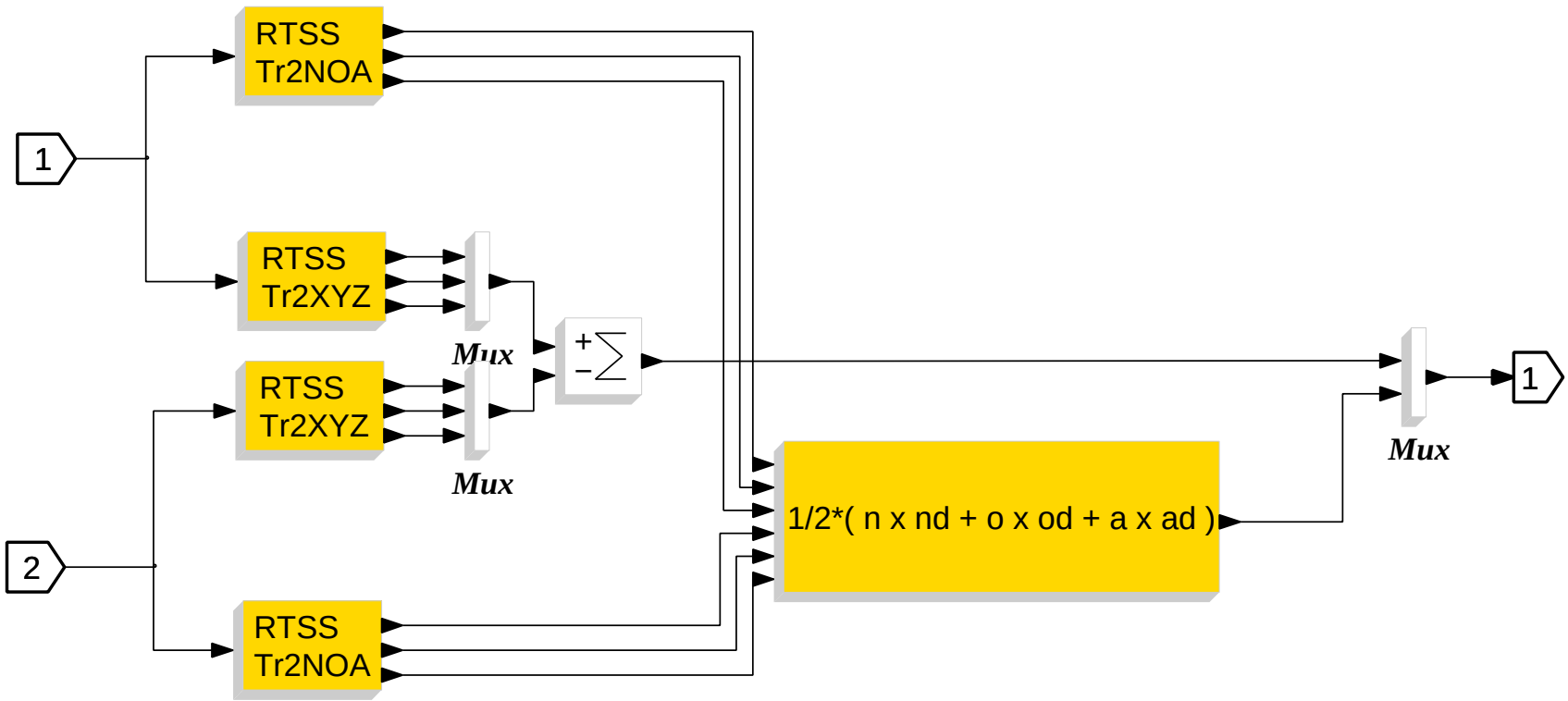
Kinematic structure of the Comau SMART 3-S robot

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

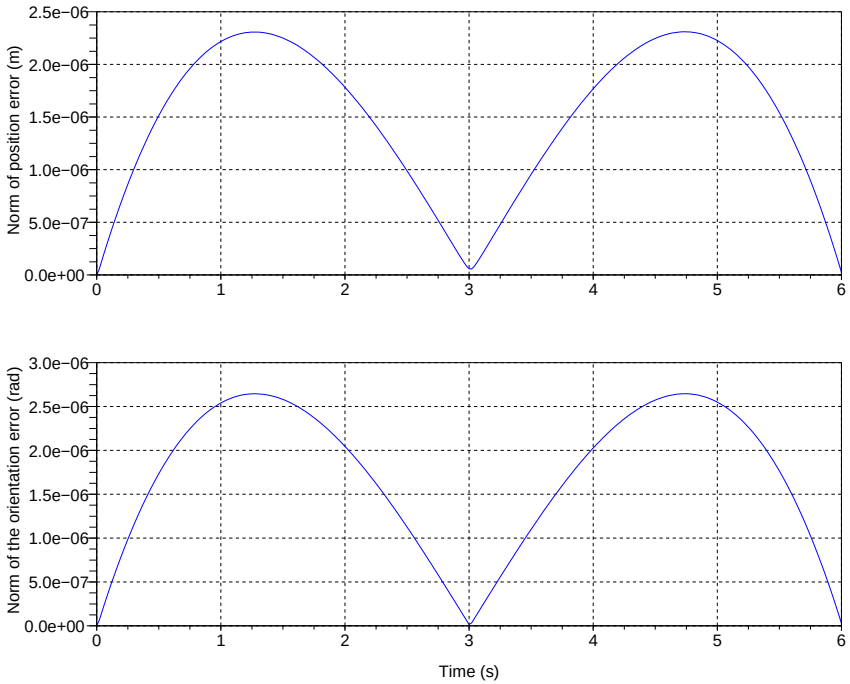
Kinematic inversion of a 7-DOF industrial manipulator with non-spherical wrist



Error block for axis-angle representation of the orientation

First-order closed-loop inverse kinematics

Analysis of simulation results



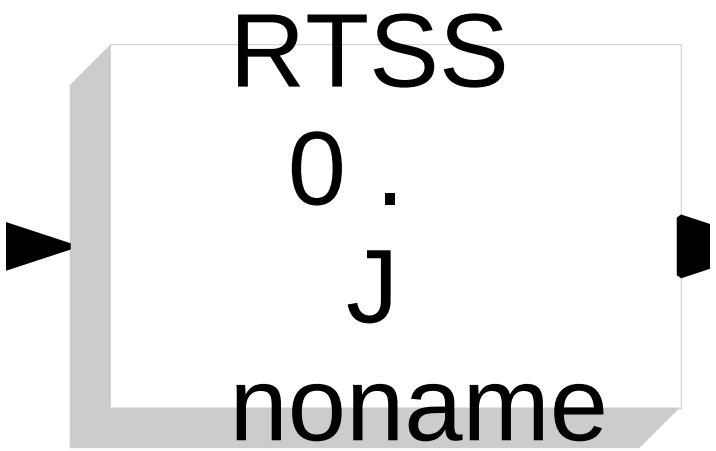
Time history of the EE error

Launch the movie!

Comau SMART 3-S performing the task

Outline

Further applications
Blocks under developments



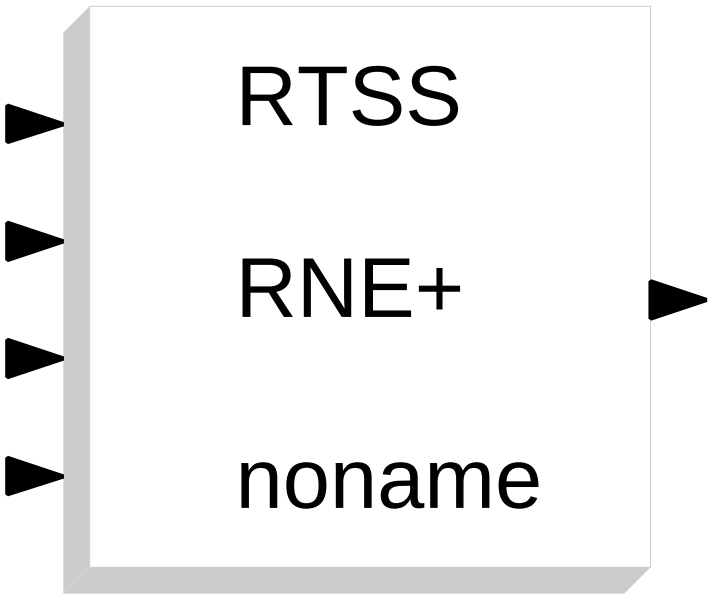
Time-derivative Jacobian block

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.

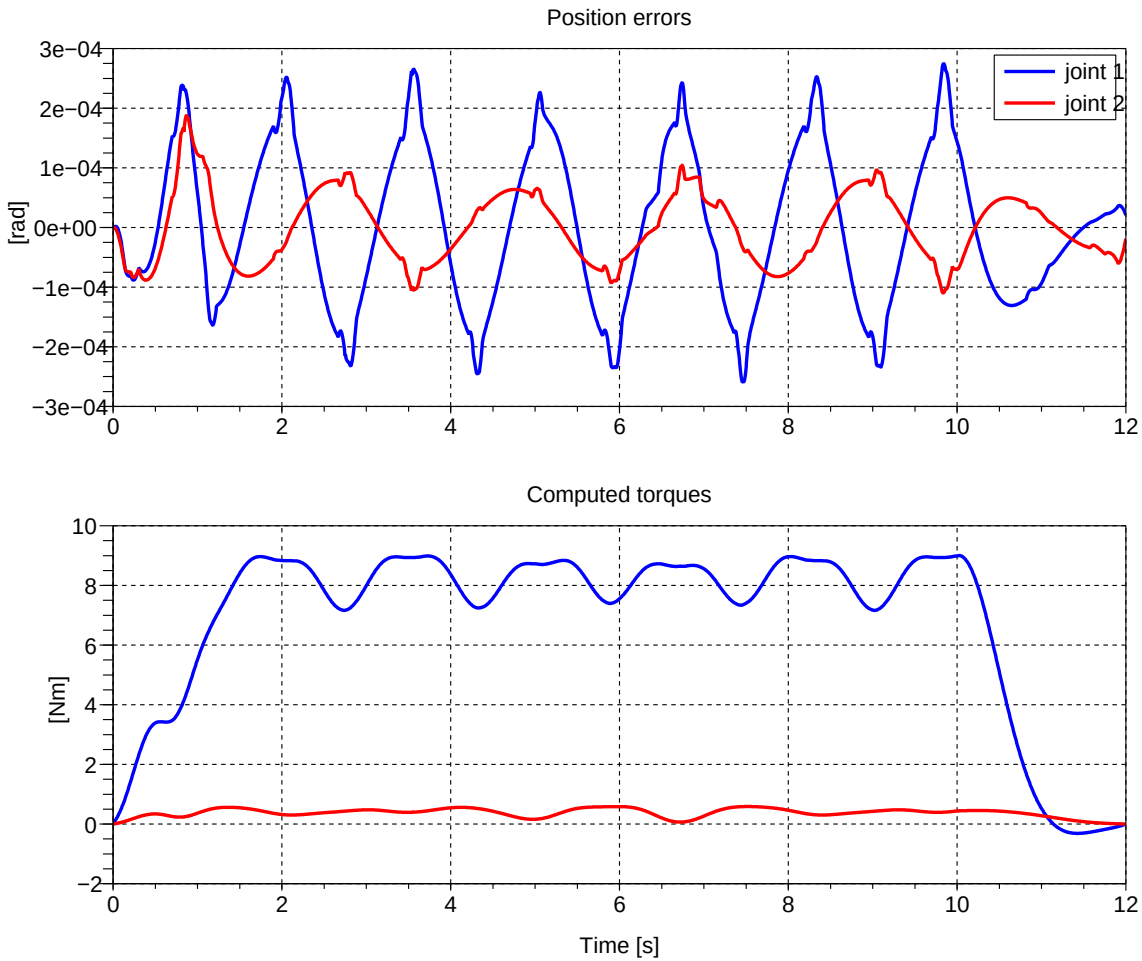
Applications

- Countless.

Outline

Inverse dynamics controller for the Pelican arm

Analysis of simulation results [Morelli, 2009]

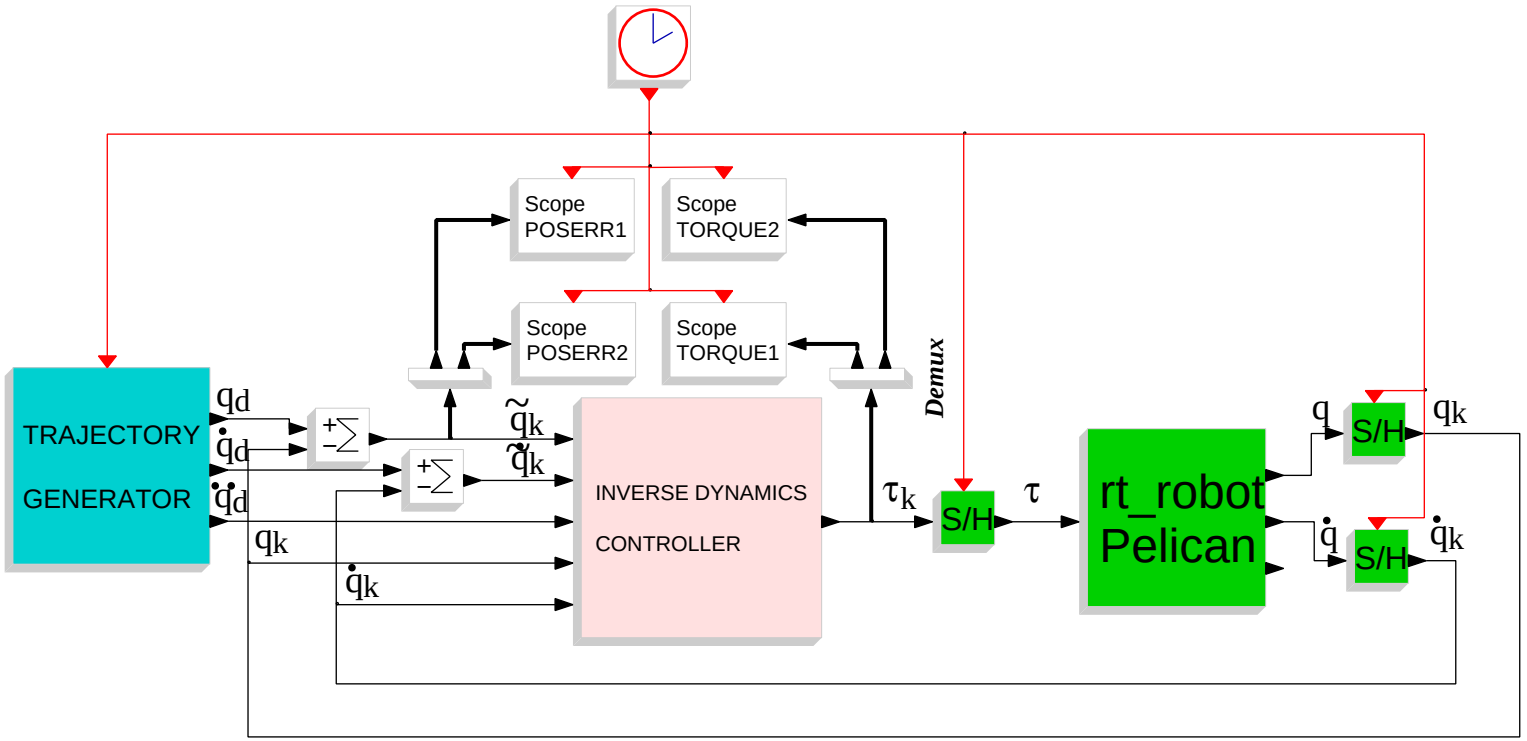


Graph of position errors and computed torques against the time

Inverse dynamics controller for the Pelican arm

Scicos block diagram for real time code generation

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

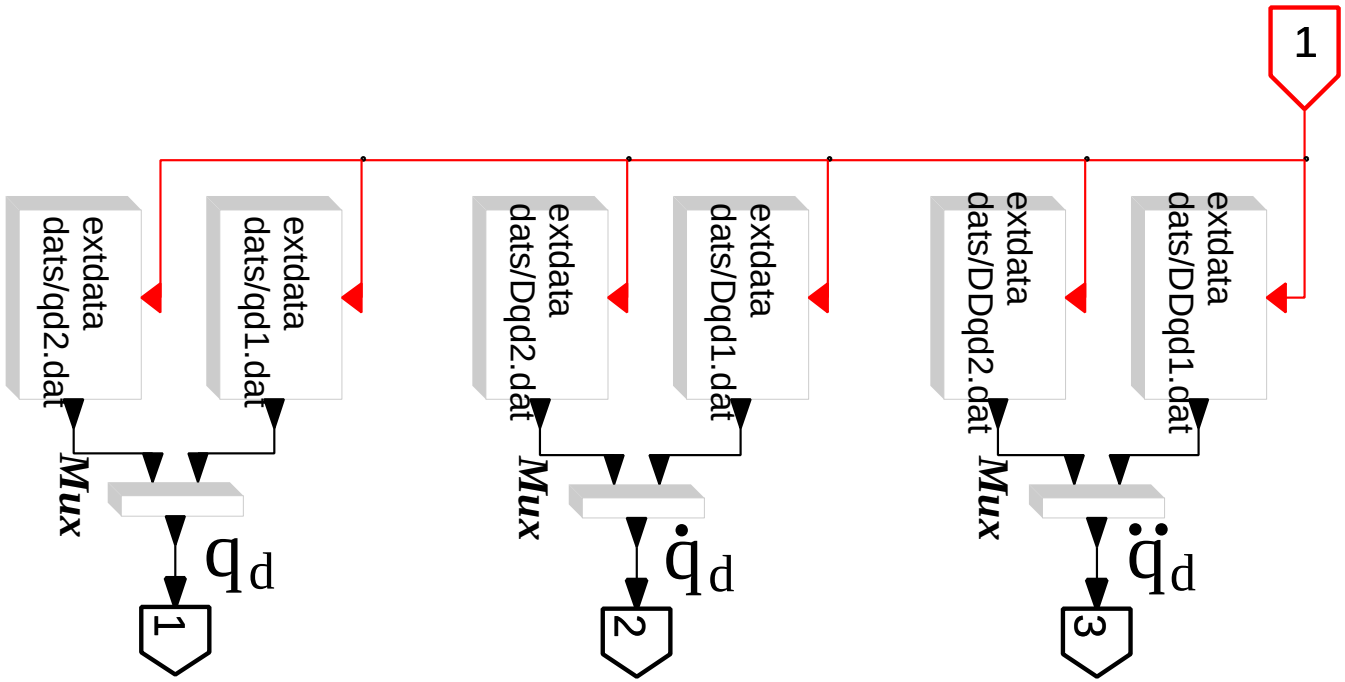


Scicos block diagram for the control system (real time control)

Inverse dynamics controller for the Pelican arm

A look inside the trajectory generator

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

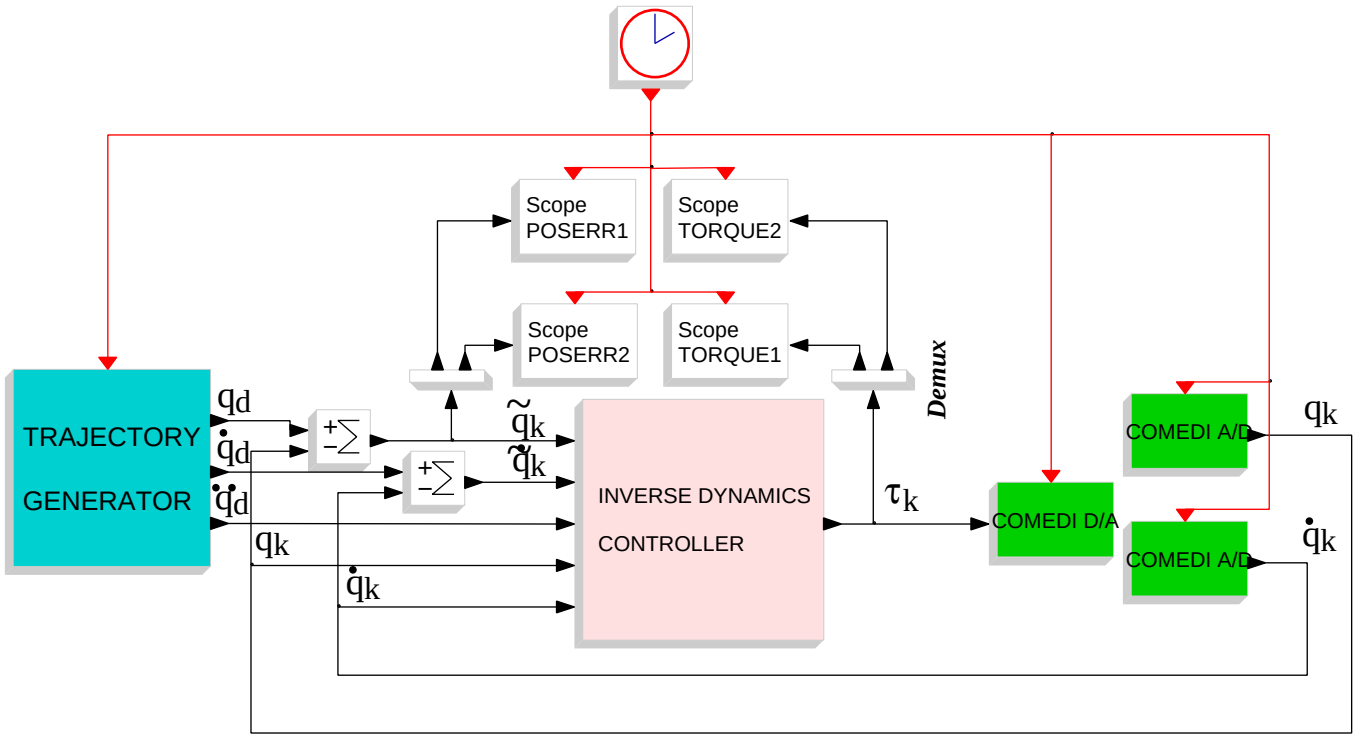


Block diagram for the trajectory generator (real time control)

Inverse dynamics controller for the Pelican arm

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

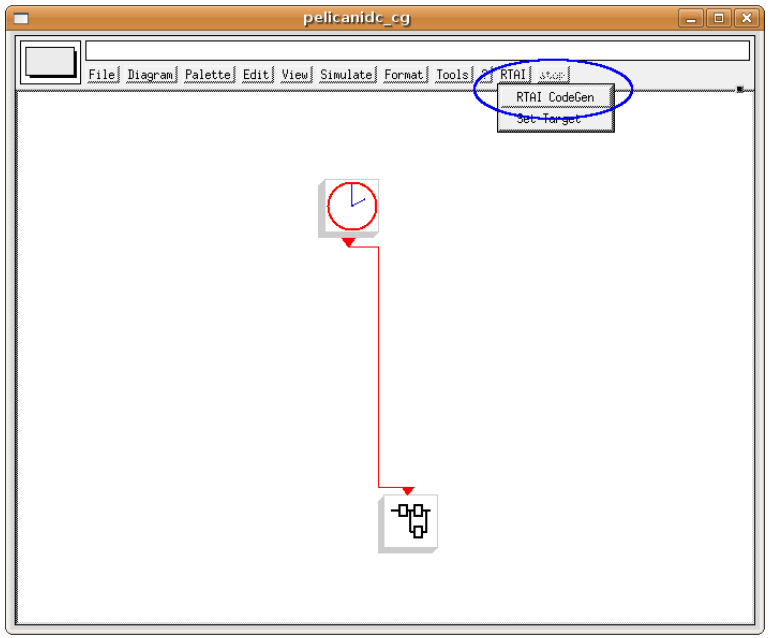
Inverse dynamics controller for the Pelican arm

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.

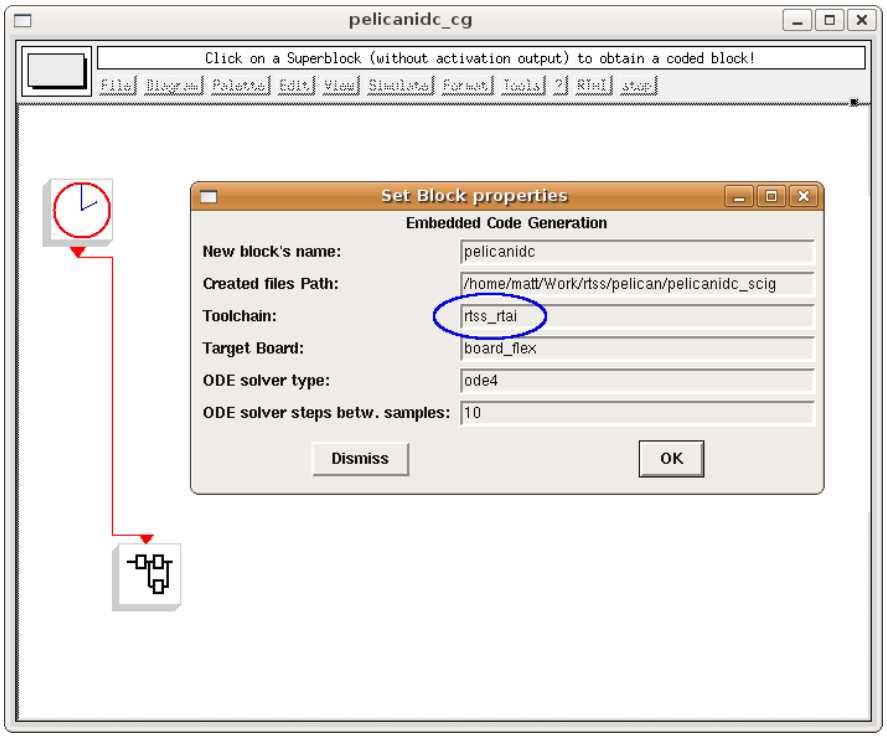


RTC diagram after step 1

Inverse dynamics controller for the Pelican arm

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`.

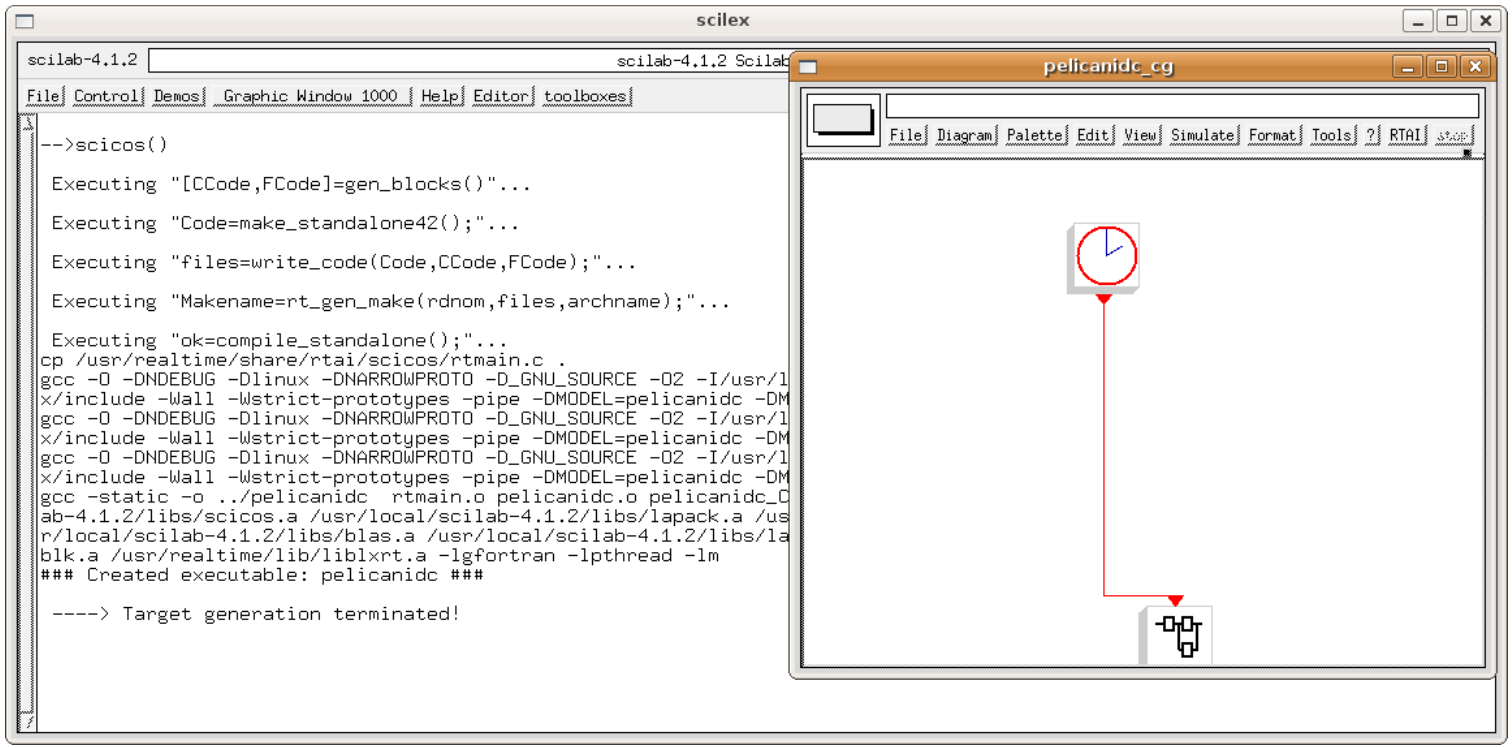


`RTAI CodeGen` dialog box

Inverse dynamics controller for the Pelican arm

Standalone, hard real time controller generation

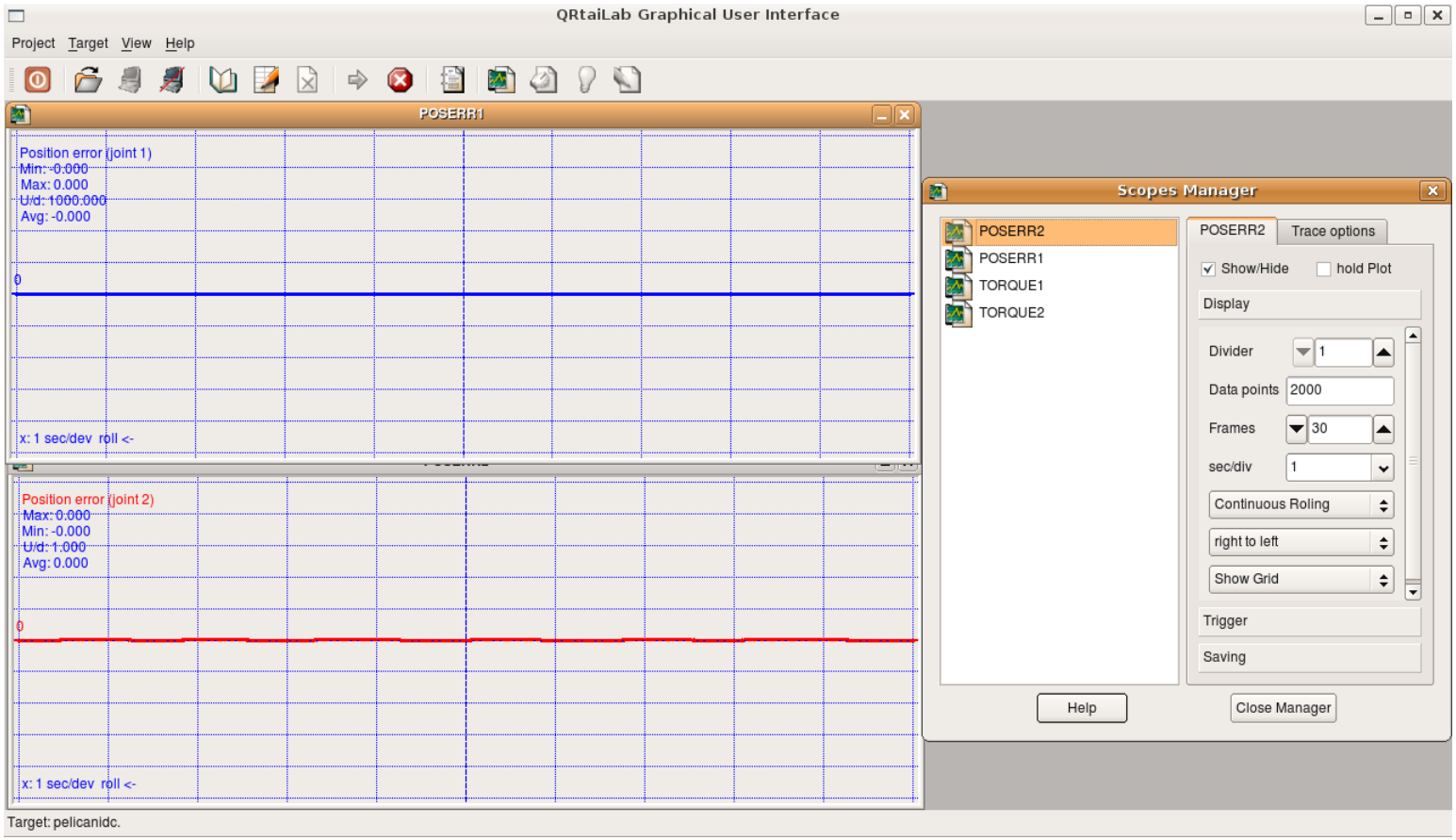
The compilation starts and completes. An executable file called `pelicanidc` is created.



Compilation output in the Scilab window

Inverse dynamics controller for the Pelican arm

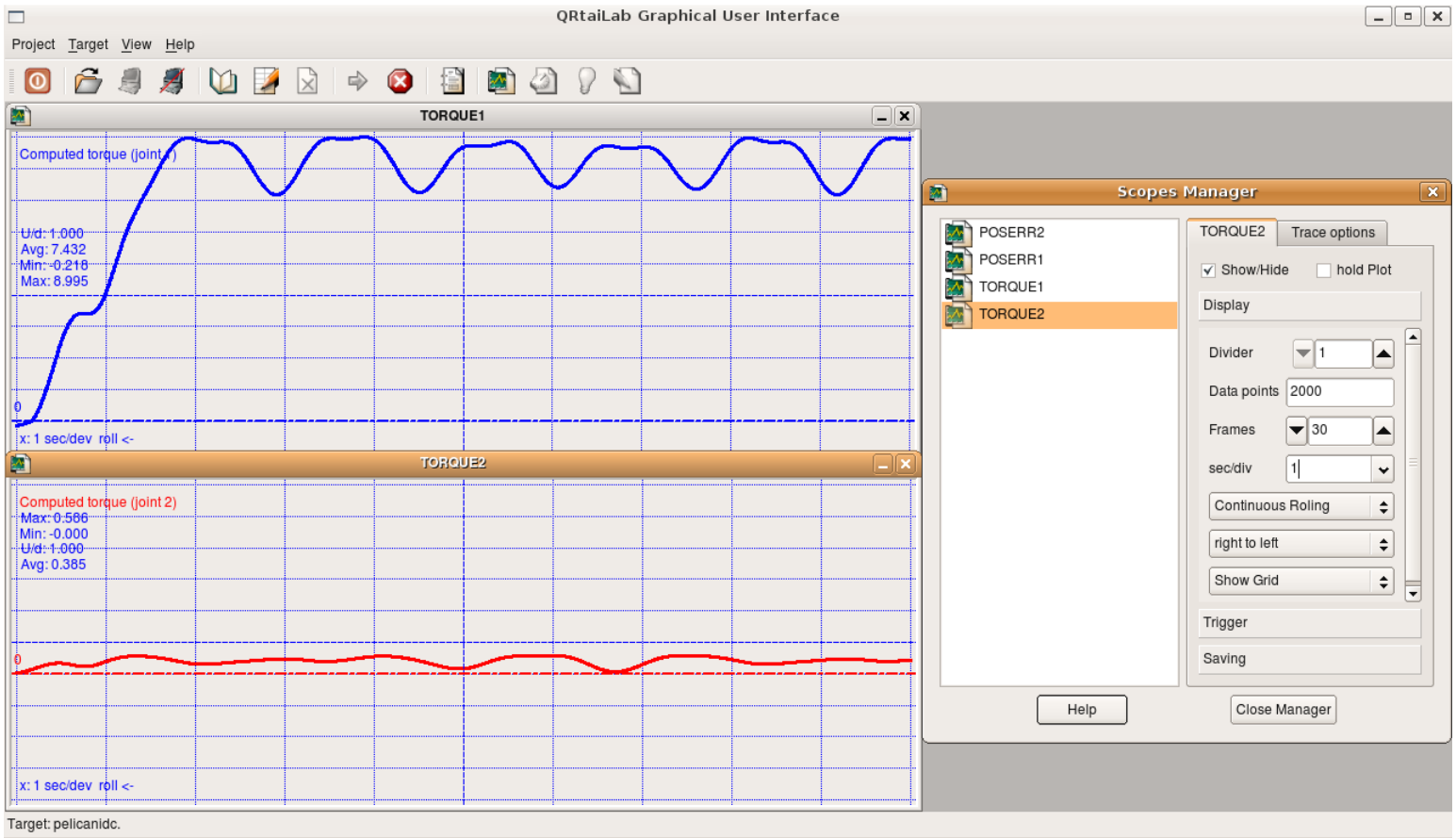
Monitoring the real time controller with the QRtaiLab Graphical User Interface



QRtaiLab with the scope manager and the position errors scopes

Inverse dynamics controller for the Pelican arm

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 3D closed-chain robot 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.


M. Morelli.
Open Source Robotics with Scilab/Scicos.
Presentation at the 1st HeDiSC Workshop On Open Source Software for Control Systems, SUPSI-DTI, Lugano, Switzerland, 2009.

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.