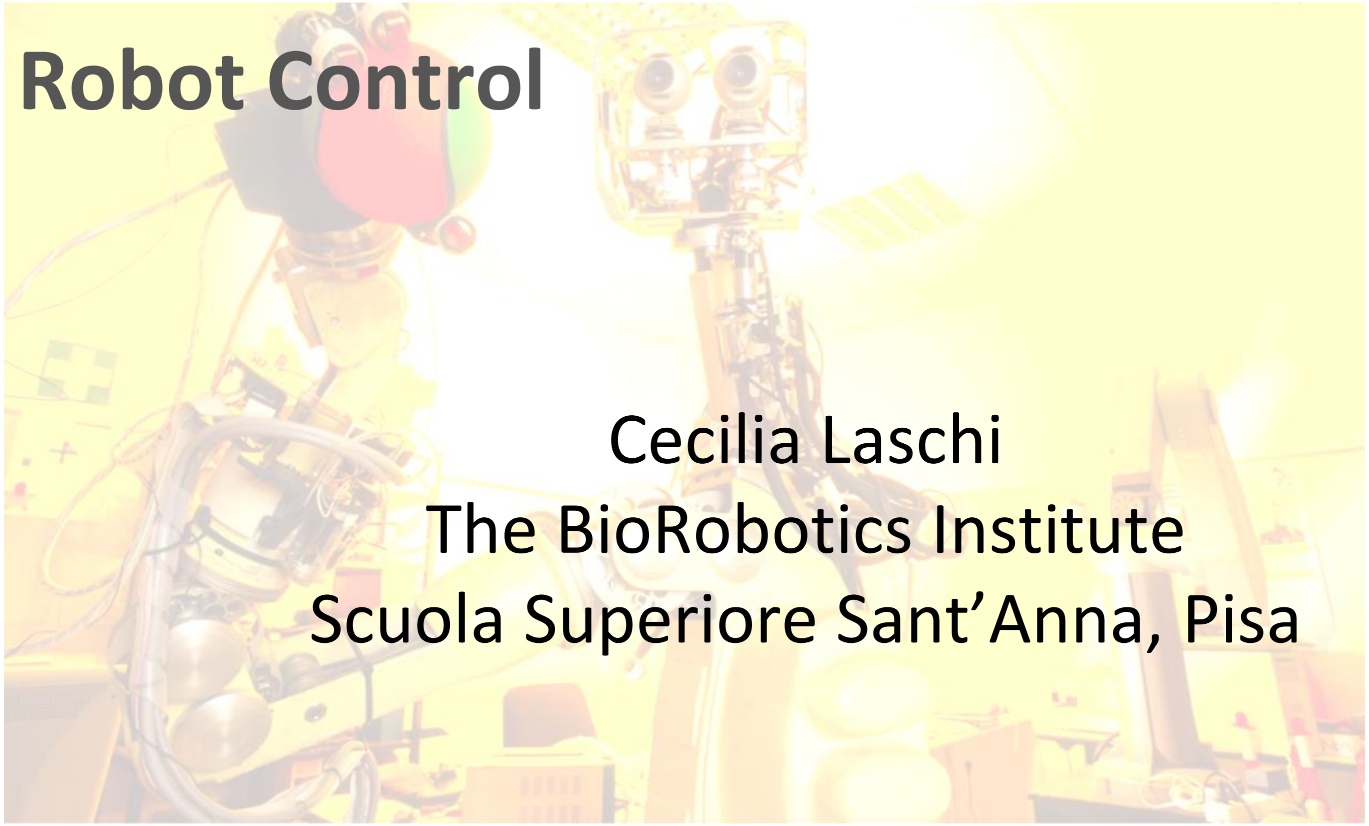
University of Pisa
Master of Science in Computer Science
Course of Robotics (ROB)
A.Y. 2016/17



Robot Control

Cecilia Laschi
The BioRobotics Institute
Scuola Superiore Sant'Anna, Pisa

cecilia.laschi@santannapisa.it

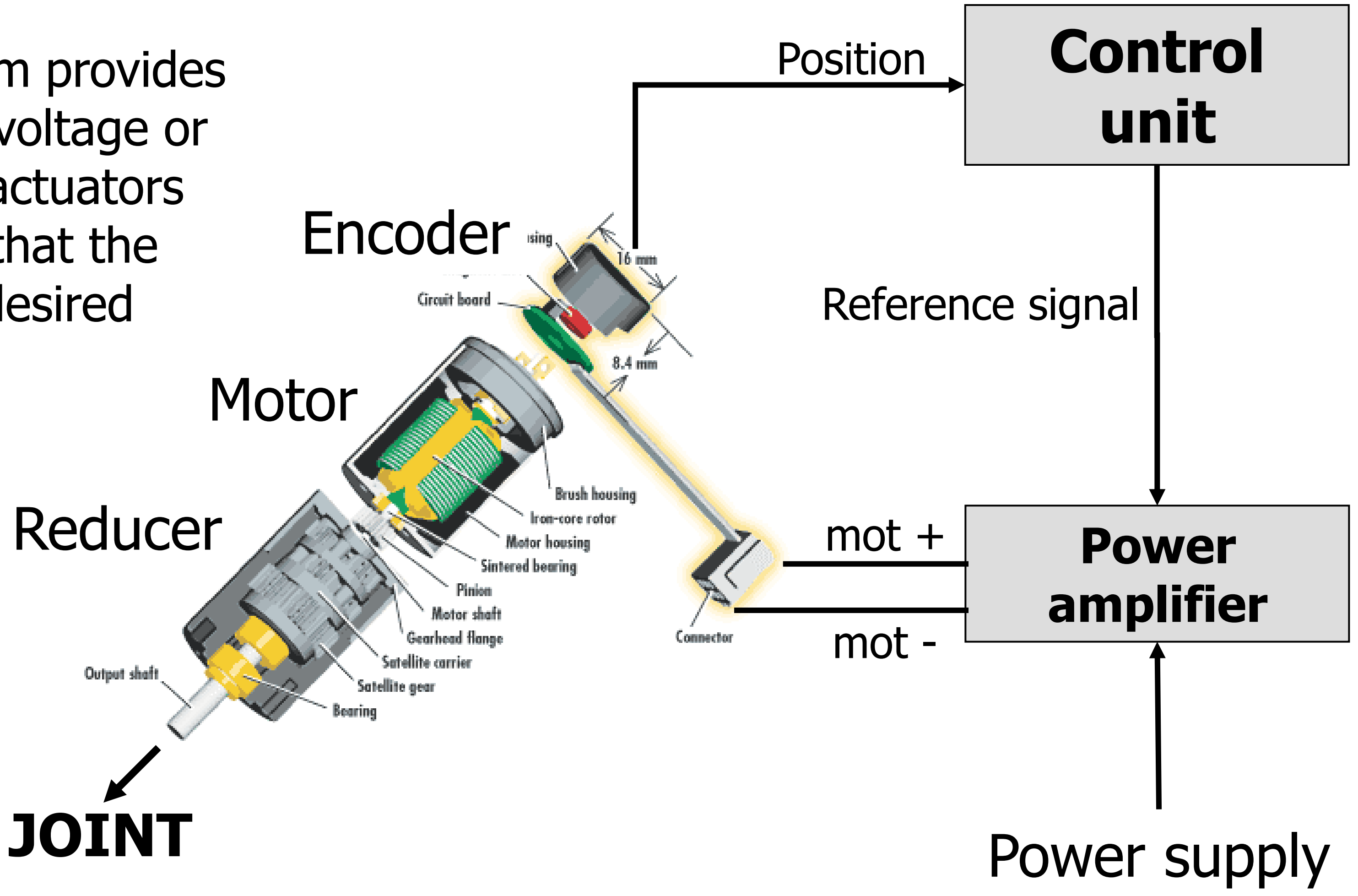
http://didawiki.cli.di.unipi.it/doku.php/magistraleinformatica/rob/start

Robot Control

- Control of one joint motion:
  - PID controller
- Control of the manipulator motion:
  - Trajectory planning
  - Motion control in joint space
  - Motion control in operational space
- The Dexter Arm example:
  - Mechanics, Kinematics, Control, Software interfaces

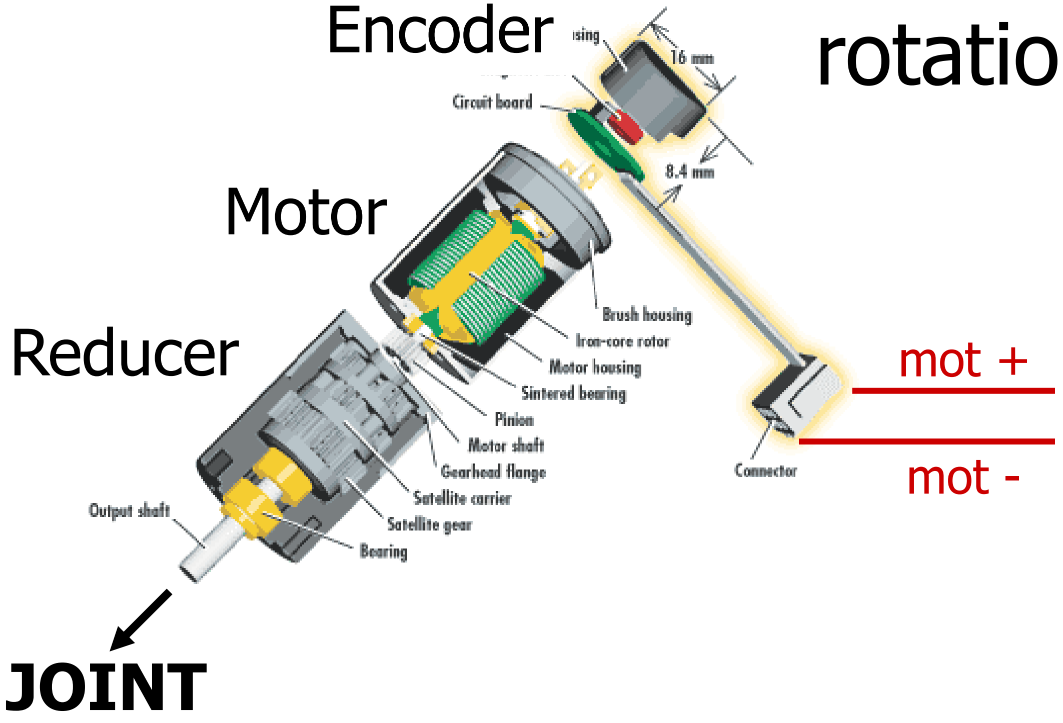
Scheme of a control system

A control system provides a command in voltage or current to the actuators (motors) such that the joints reach a desired configuration



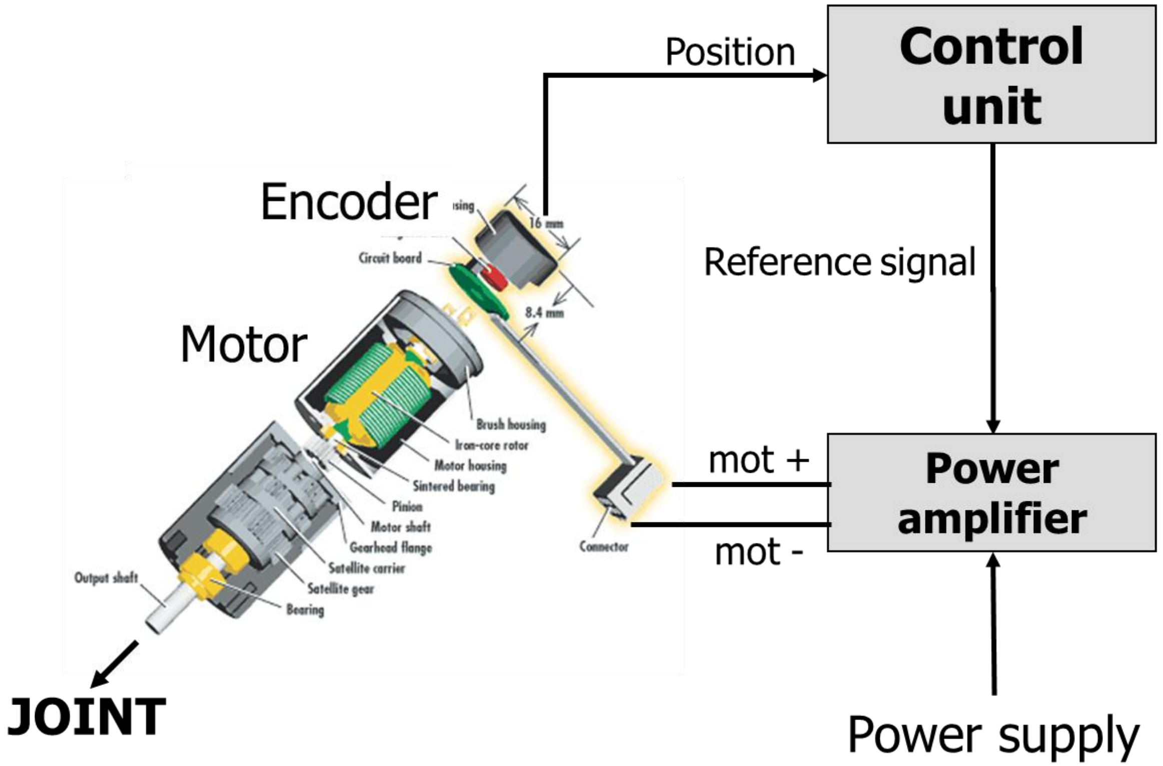
Scheme of a control system

Opposite-sign voltages produce opposite rotations of the motor



Scheme of a control system

- **Encoder**: sensor measuring joint rotations, either as an absolute or a relative value. The measurement is given in "*encoder steps*"
- **Reducer:** mechanism reducing the motor rotations with respect to the rotations of the axis mounted on the motor (ex. 1:k reduction)
- **Power amplifier:** it amplifies a reference signal into a power signal for moving the joint
- **Control unit**: unit producing the reference signal for the motor



Relations between joint position and encoder position

- q: joint angular position (in degrees)
- θ: joint position in encoder steps
- k: motor reduction ratio
- R: encoder resolution (number of steps per turn)

$$q = \frac{\theta \times 360^\circ}{R \times k}$$

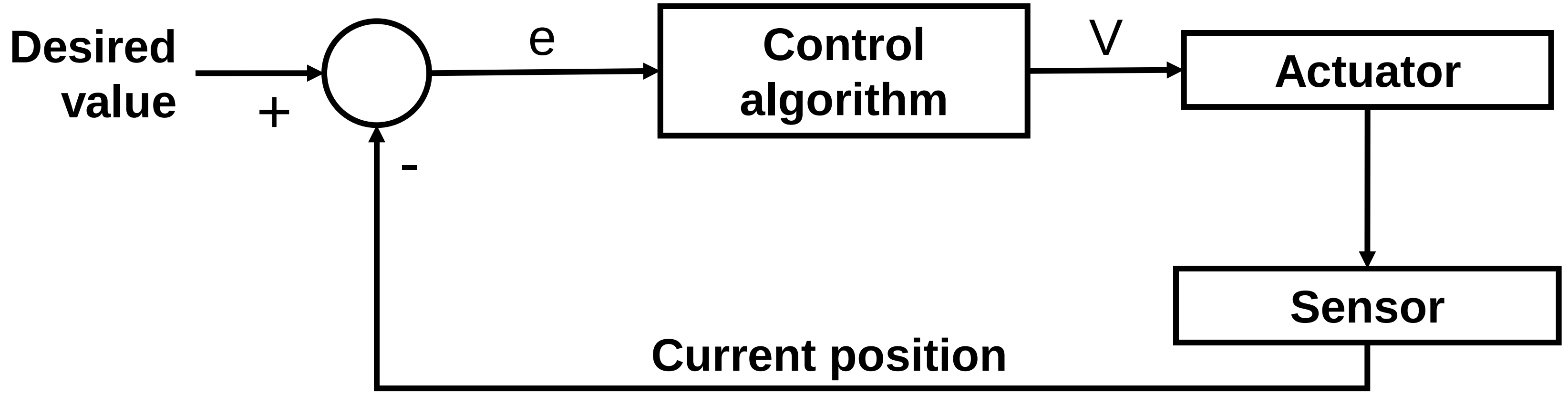
Control of one joint motion

- Objective: move the joint from the current position q_i (in degrees) to the desired position q_f, in a time interval t :

$$q_i \Rightarrow q_f$$

Closed-loop (feedback) control

- The variable to control is measured and compared with the desired value
- The difference, or error, is processed by an algorithm
- The result of processing is the input value for the actuator

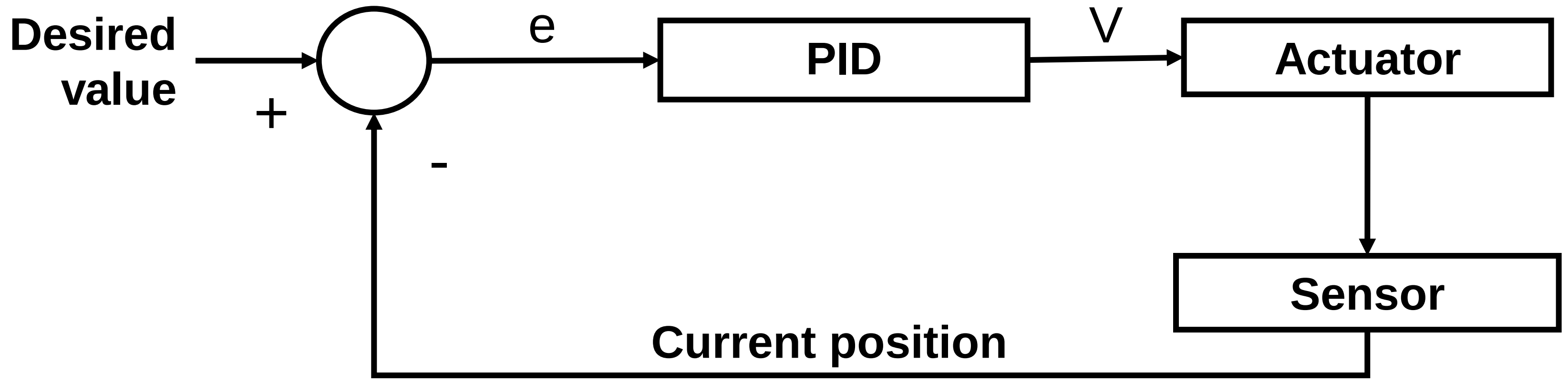


PID control
(Proportional, Integral, Derivative)

- It is a closed-loop control in which the error is processed with an algorithm including **Proportional, Integral and Derivative** components.
- The algorithm processes the error and provides an input to the actuator, with 3 components:
  - **Proportional**, producing a correction proportional to the error;
  - **Integral**, producing a correction given by the error integral in time;
  - **Derivative**, producing a correction which is a function of the error first derivative.
- Not all closed-loop control systems use a PID algorithm

PID control
(Proportional, Integral, Derivative)

- In a PID control system, the error is given to the control algorithm, which calculates the derivative and integral terms and the output signal V



PID control
(Proportional, Integral, Derivative)

$$V = K_p e_q + K_d \dot{e}_q + K_i \int e_q(t)\,dt$$

$$e_q = q_d - q_a$$

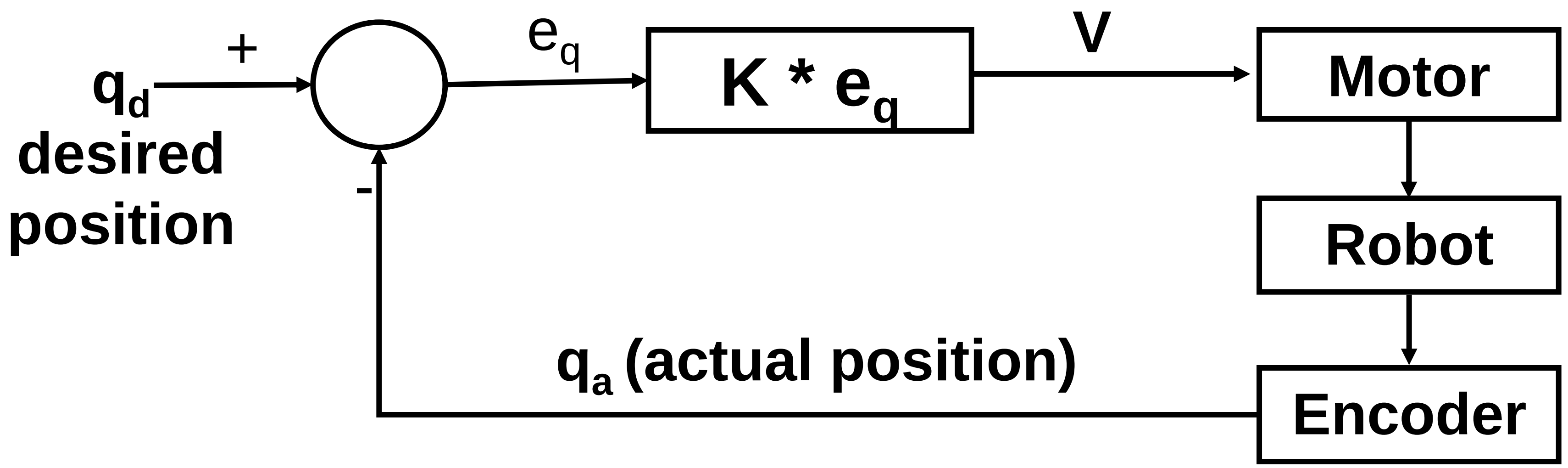
$$\dot{e}_q = \frac{de_q}{dt}$$

- ***K_p*** is the *proportional* gain or constant
- ***K_i*** is the *integral* gain or constant
- ***K_d*** is the *derivative* gain or constant
- ***e_q*** is the error, i.e. the difference between the desired position and the current position

PID control

Proportional term

- The voltage *V* given to the motor is proportional to the difference between the actual position measured by the sensor and the desired position



PID control

Proportional term:

- The voltage V given to the motor is proportional to the difference between the actual position measured by the sensor and the desired position

$$V = K_p e_q$$

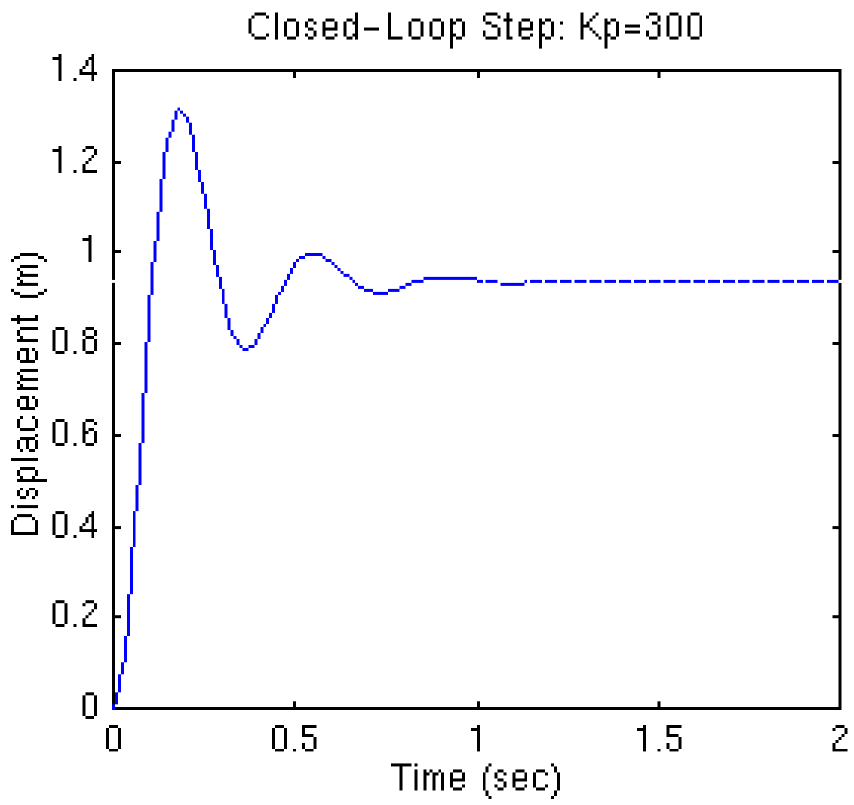
$$e_q = q_d - q_a$$

K_P: proportional constant

PID control

Proportional term: system behaviour

Desired position: 1



- The motor oscillates before converging towards the desired position
- The system may settle without cancelling the error

PID control

Derivative term:

$$\dot{e}_q = \frac{de_q}{dt}$$ Error derivative in time

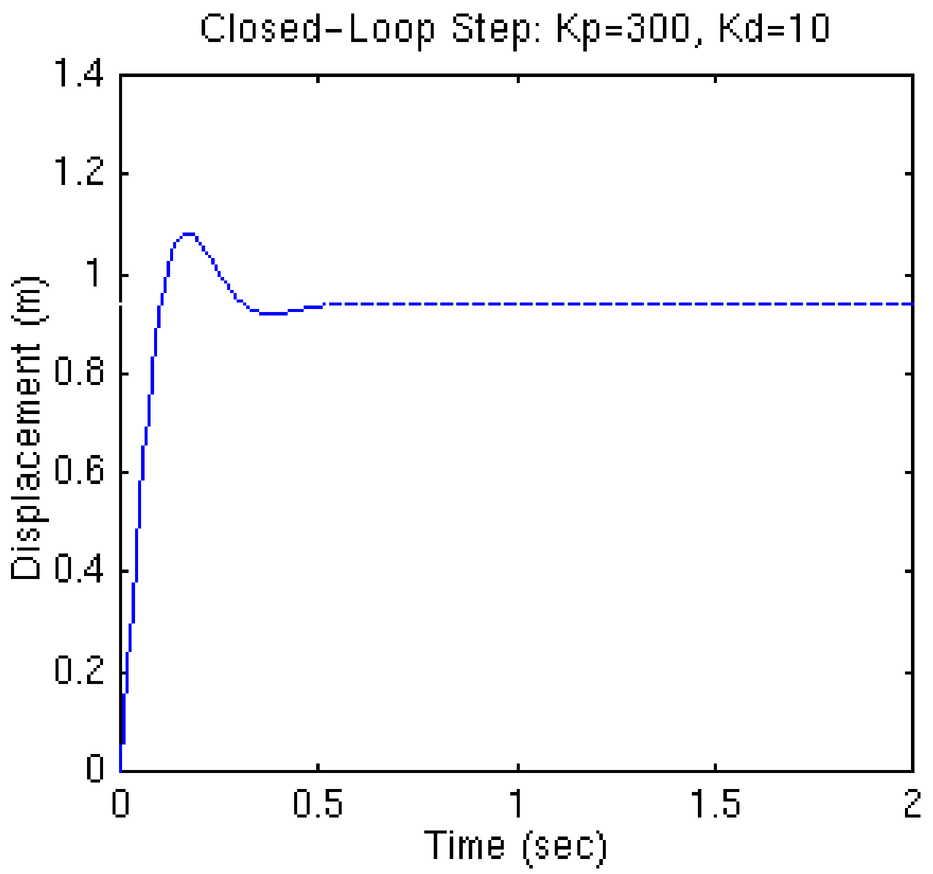
$$V = K_p e_q + K_d \dot{e}_q$$ K_d : derivative constant

$$e_q = q_d - q_a$$

PID control

Proportional and derivative terms:

Desired position: 1



- Oscillation reductions
- Reduction of settlement time
- The system may settle without cancelling the error

PID control

Integral terms:

$K_i \int e_q(t)dt$ Error integral in time

$$V = K_p e_q + K_i \int e_q(t)dt$$

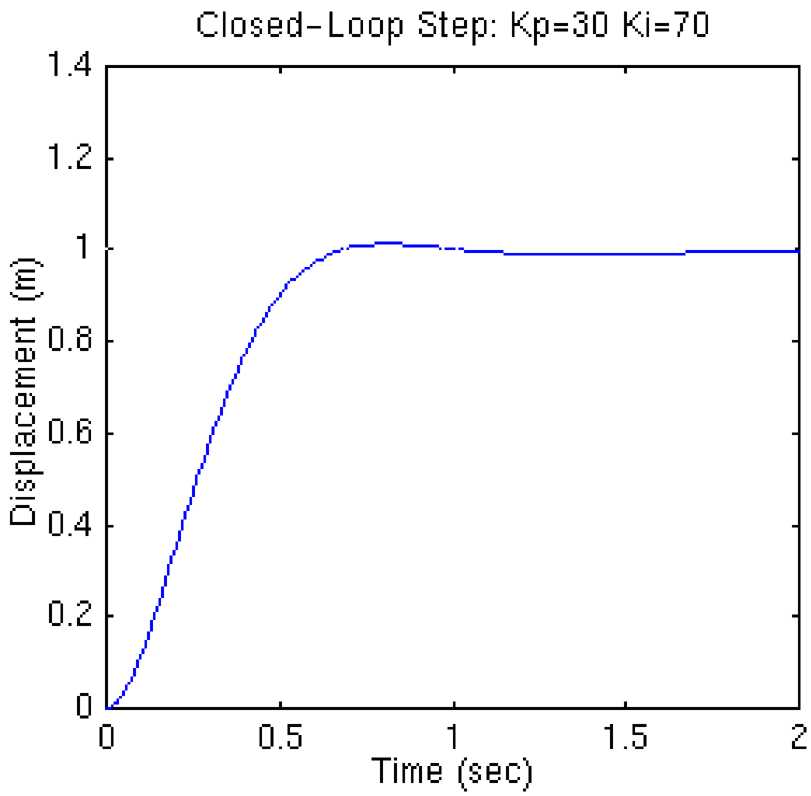
$$e_q = q_d - q_a$$

K_i: integral constant

PID control

Proportional and integral terms:

Desired position: 1



- The system settles and cancels the error

PID control

- **Proportional, Integral and Derivative terms:**

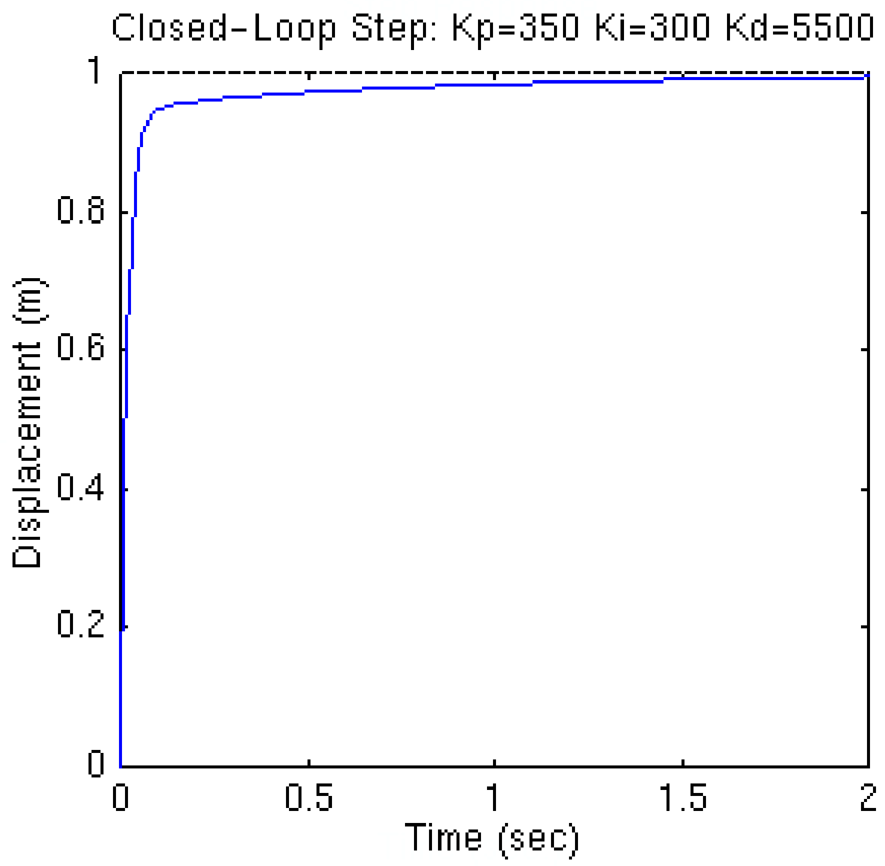
$$V = K_p e_q + K_d \dot{e}_q + K_i \int e_q(t)\,dt$$

$$e_q = q_d - q_a$$

$$\dot{e}_q = \frac{de_q}{dt}$$

PID control

- **Proportional, Integral and Derivative terms:**



- K_p, K_d, K_i constants are set empirically or with specific methods

Control of robot manipulator motion

- Objective: to have the robot arm moving from a starting position to a final position, both expressed in operational space coordinates
- In general, the control problem consists in finding the torques that the actuators have to give to the joints, so that the resulting arm motion follows a planned trajectory

Trajectory planning

Objective: generate the reference inputs to the robot control system, which will ensure that the robot end effector will follow a desired trajectory when moving from x_{start} to x_f

- PATH: set of points, in joint space or operational space, that the robot has to reach in order to perform the desired movement
- TRAJECTORY: path with a specified time course (velocity and acceleration at each point)

Trajectory planning

Objective: generate the reference inputs to the robot control system, which will ensure that the robot end effector will follow a desired trajectory when moving from x_{start} to x_f

- INPUT DATA:
  - Path definition
  - Path constraints
  - Constraints given by the robot dynamics

- OUTPUT DATA:
  - **in joint space**: joint trajectories $\{\mathbf{q}(t),\ \dot{\mathbf{q}}(t),\ \ddot{\mathbf{q}}(t)\}$
  - **in operational space**: end-effector trajectory $\{\mathbf{p}(t),\ \boldsymbol{\phi}(t),\ \mathbf{v}(t),\ \boldsymbol{\Omega}(t)\}$

Trajectories in joint space

- Between two points: the robot manipulator must be displaced from the initial to the final joint configuration, in a given time interval t.
- In order to give the time course of motion for each joint variable, we can choose a trapezoidal velocity profile or polynomial functions:
  - Cubic polynom: it allows to set
    - the initial and final values of joint variables q_i and q_d
    - the initial and final velocities (usually null).
  - Fifth-degree polynom: it allows to set
    - the initial and final values of joint variables q_i and q_d
    - the initial and final velocities
    - the initial and final accelerations.

Trajectories in joint space

Trapezoidal velocity profile:

- Constant acceleration in the starting phase
- Constant cruise velocity
- Constant deceleration in the arrival phase.

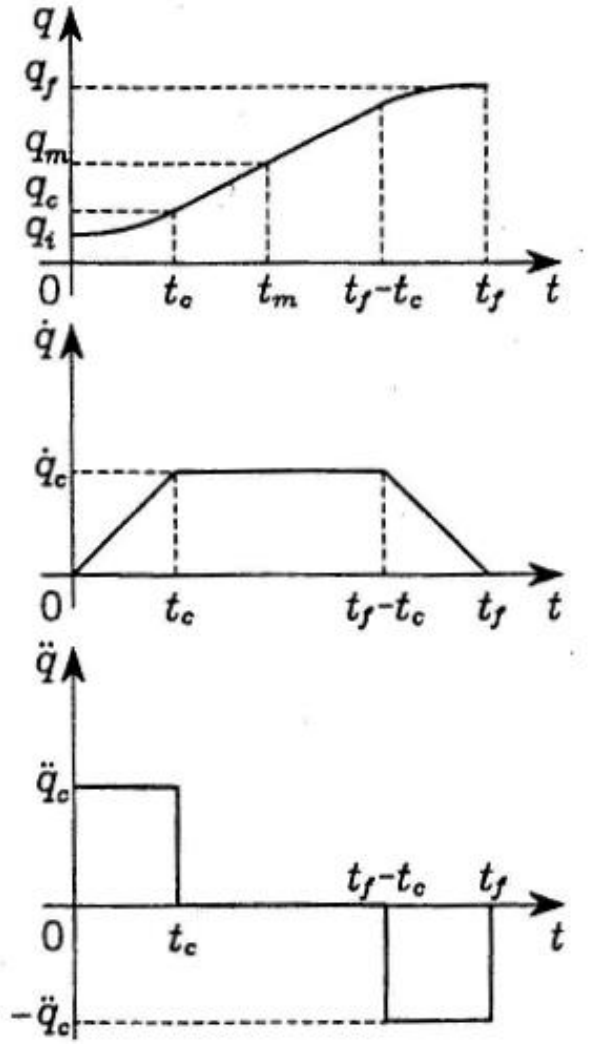
The corresponding trajectory is mixed polynomial: a linear segment connected to parabolic parts in the neighbourhood of the initial and final positions.

Trapezoidal velocity profile

Position

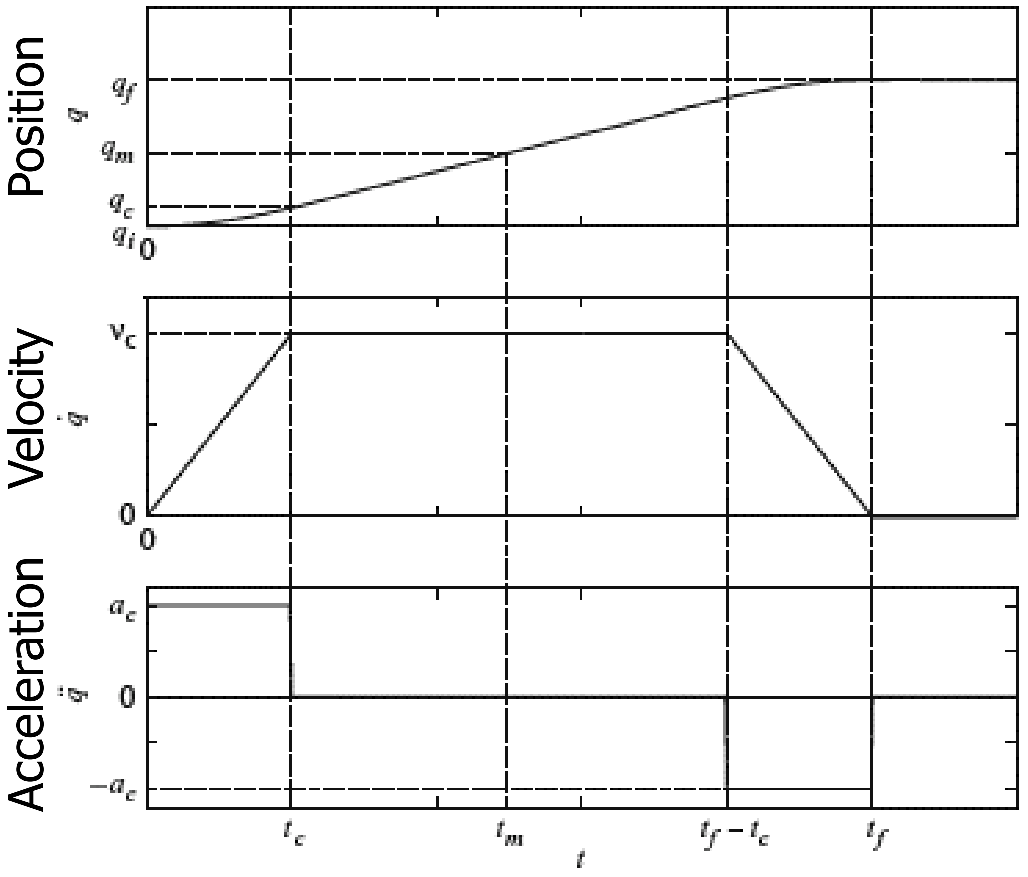
Velocity

Acceleration



Note: velocities and accelerations at the initial and final times can be different from zero

Trapezoidal velocity profile



First phase:

$$q_1(t) = q_i + \frac{1}{2}a_c t^2 \qquad 0 \le t \le t_c$$

Second phase:

$$q_2(t) = q_i + a_c t_c \left(t - \frac{t_c}{2}\right) \qquad t_c < t \le (t_f - t_c)$$

Third phase:

$$q_3 = q_f - \frac{1}{2}a_c (t - t_f)^2 \qquad (t_f - t_c) < t \le t_f$$

Trajectory interpolation

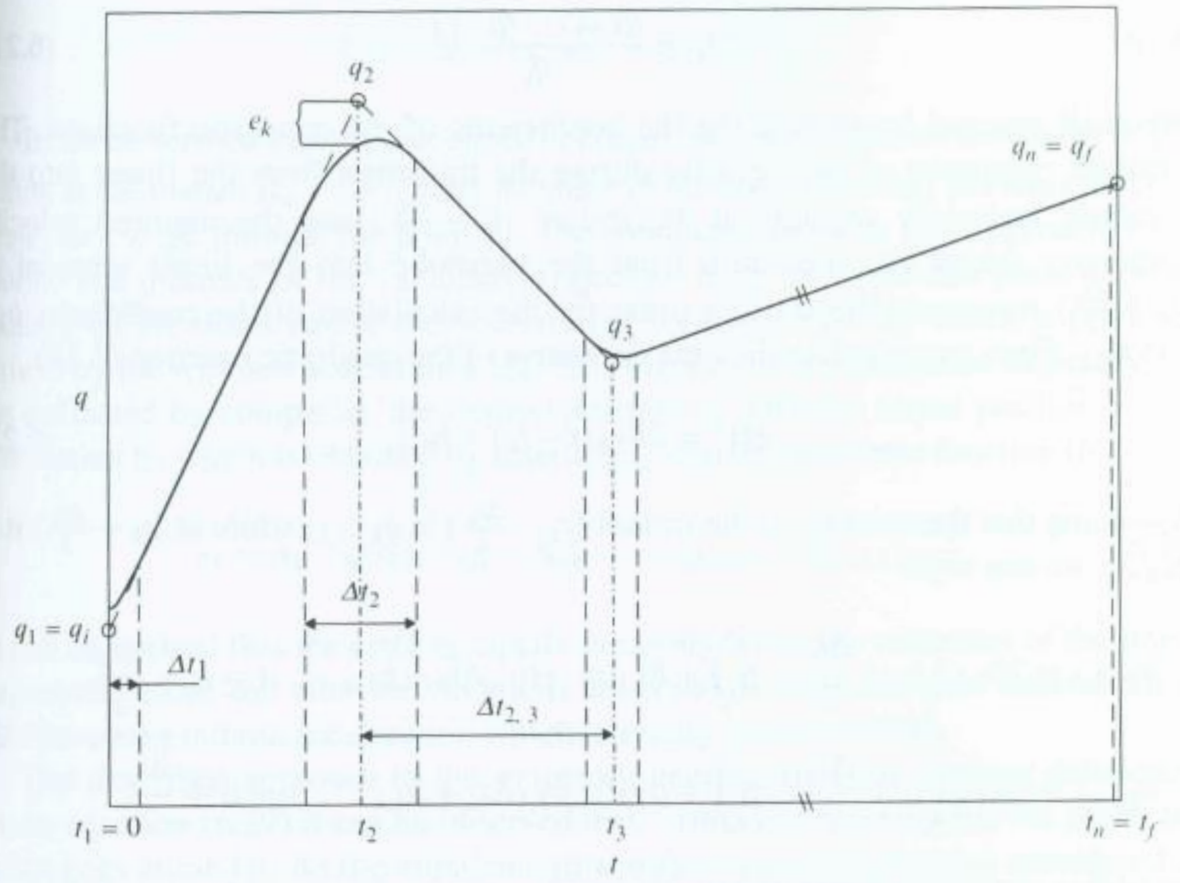


Fig. 6.2 Trajectory interpolation through n via points – linear segments with parabolic transitions are used

Trajectories in operational space

- The trajectory planning algorithm generates the time course of motion of the end effector, according to a path of geometric characteristics defined in the operational space.

- The result of planning is a sequence of n-uples: $(p(t), \Phi(t), v(t), \omega(t))$

Robot kinematics and differential kinematics

Kinematics

$$x = k(q)$$
$$q = k^{-1}(x)$$

$$x = \begin{bmatrix} x \\ y \\ z \\ \varphi \\ \vartheta \\ \psi \end{bmatrix} \qquad q = \begin{bmatrix} q_o \\ q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \end{bmatrix}$$

$k(\cdot)$ = equations of direct kinematics

Differential kinematics

$$\dot{x} = J(q)\dot{q}$$
$$\dot{q} = J^{-1}(q)\dot{x}$$

Velocity space

$J(q)$ = Jacobian matrix

Robot kinematics and dynamics

Robot dynamic model

$$\tau = B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + g(q)$$

τ = torque

B = inertia term

C = Coriolis term

F_v = friction coefficients

g = gravity terms

Robot control

Motion control can be done in

- **joint space**
- **operational space**

Motion control in joint space

- It can be used for moving the end-effector from x_i to x_d expressed in the operational space, without taking into account the trajectory followed by the end effector
- The final position x_d is transformed in the corresponding final position in joint space q_d, by using the inverse kinematics transformation

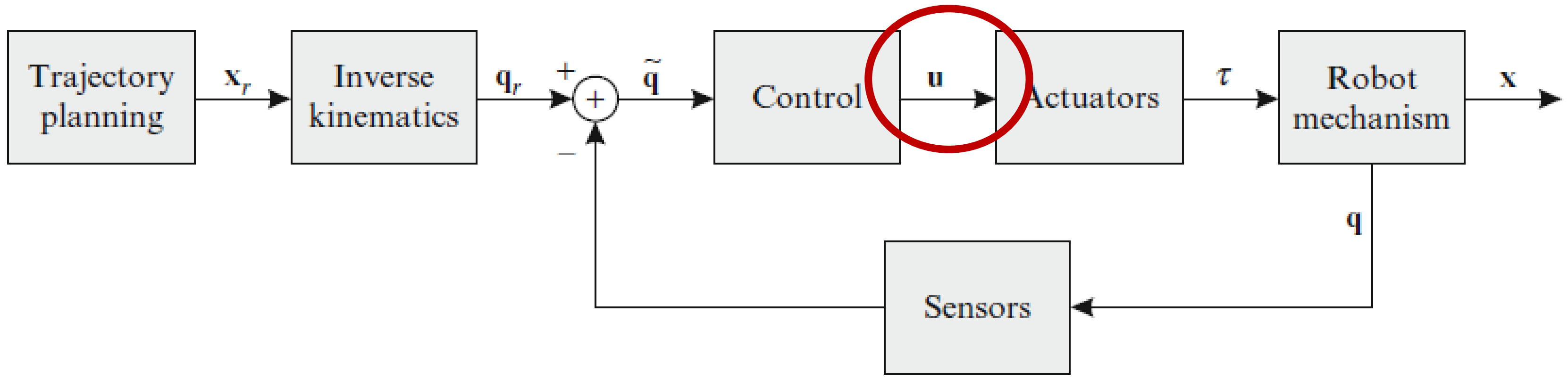
$$q_d = K^{-1} (x_d)$$

- All joints are moved from the current position q_i to the desired position q_d

Motion control in joint space

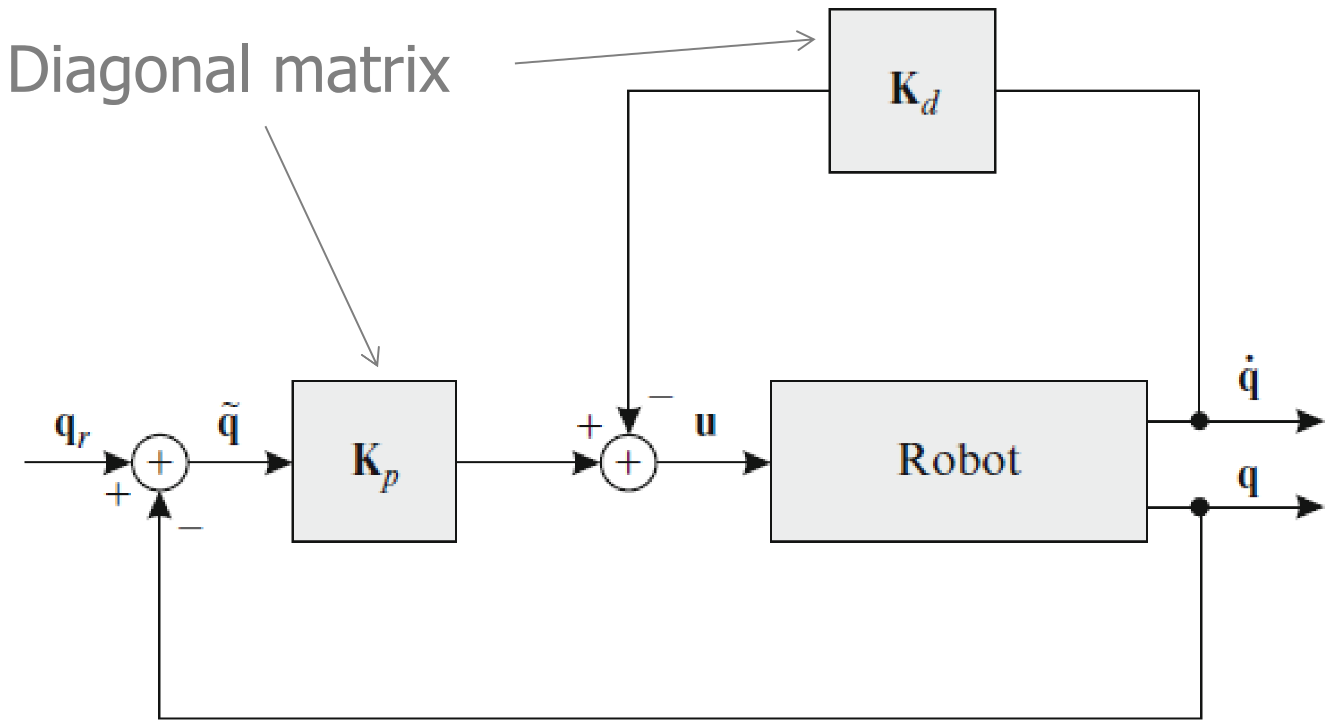
- The trajectory of the end effector in the operational space is not controlled and it is not predictable, due to the non-linear effects of direct kinematics

General scheme of robot control in joint space



Motion control in joint space

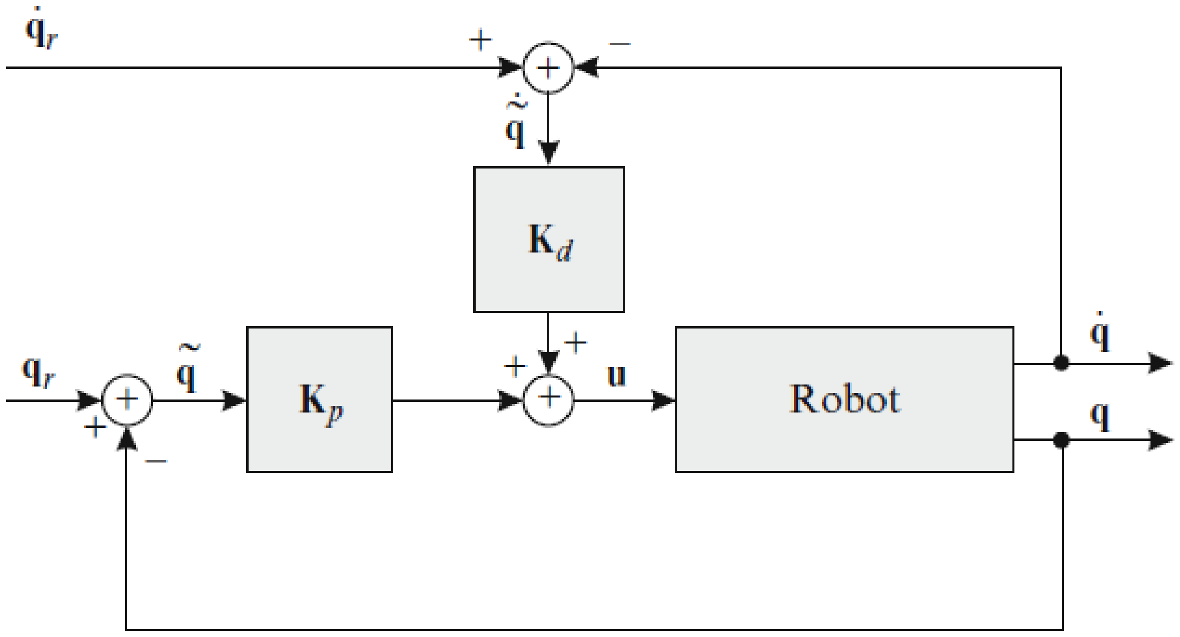
PD position control



$$\mathbf{u} = \mathbf{K}_p(\mathbf{q}_r - \mathbf{q}) - \mathbf{K}_d\dot{\mathbf{q}},$$

Motion control in joint space

PD position control



$$\mathbf{u} = \mathbf{K}_p(\mathbf{q}_r - \mathbf{q}) + \mathbf{K}_d(\dot{\mathbf{q}}_r - \dot{\mathbf{q}})$$

Motion control in joint space
PD position control

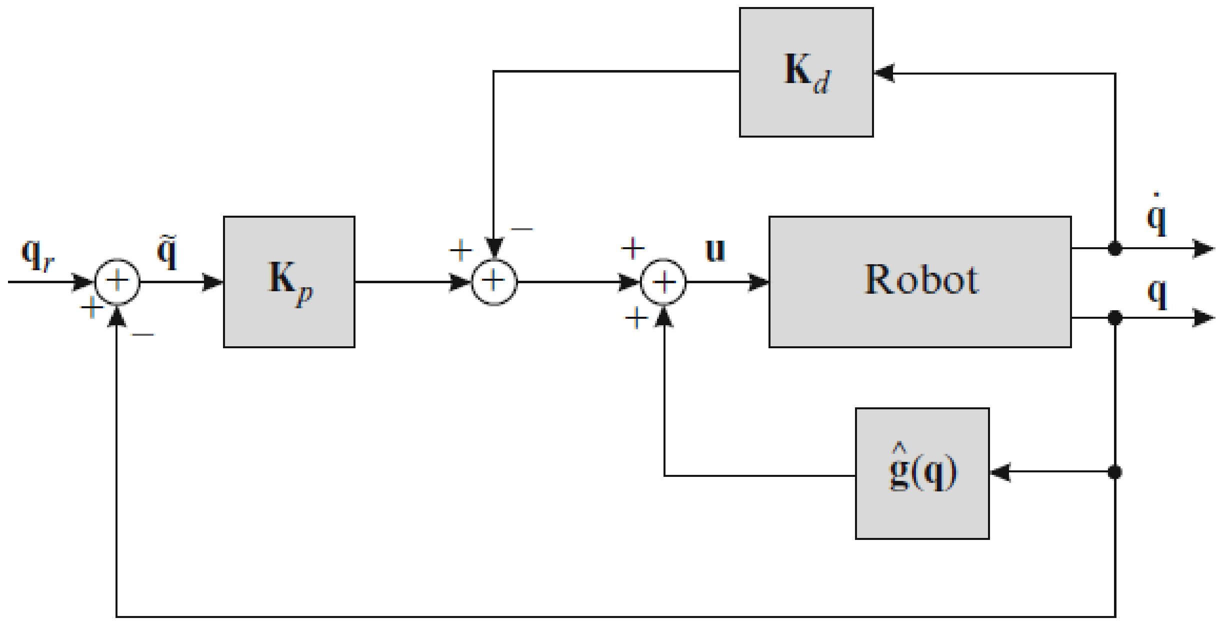
Setting the K_p e K_d parameter matrices:

- Fast response: high K_p
- K_d sets the best damping and guarantees a fast response without oscillations
- The K parameters needs to be set independently for each joint

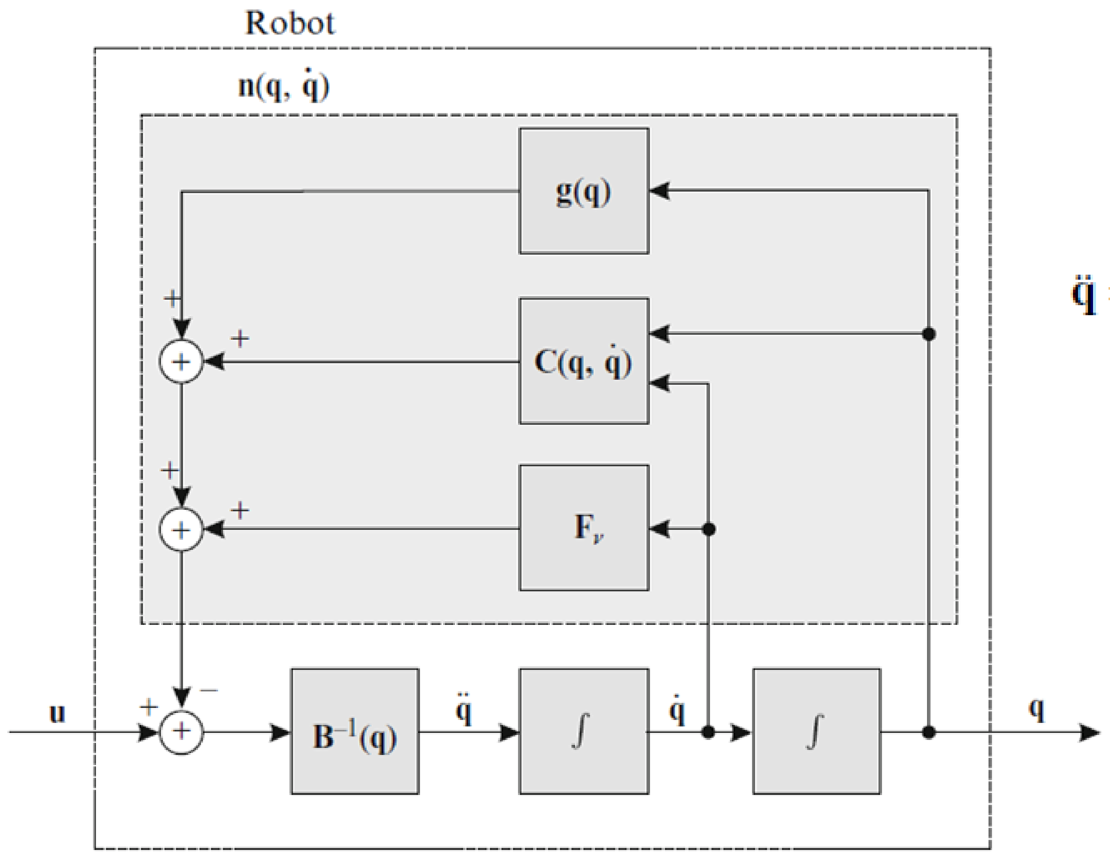
Motion control in joint space

PD position control with gravity compensation

$$\mathbf{u} = \mathbf{K}_p(\mathbf{q}_r - \mathbf{q}) - \mathbf{K}_d\dot{\mathbf{q}} + \hat{\mathbf{g}}(\mathbf{q}).$$



Motion control in joint space based on inverse dynamics



The direct dynamic model of a robot mechanism

$$B(q)\ddot{q} + C(q,\dot{q})\dot{q} + F_v\dot{q} + g(q) = u.$$

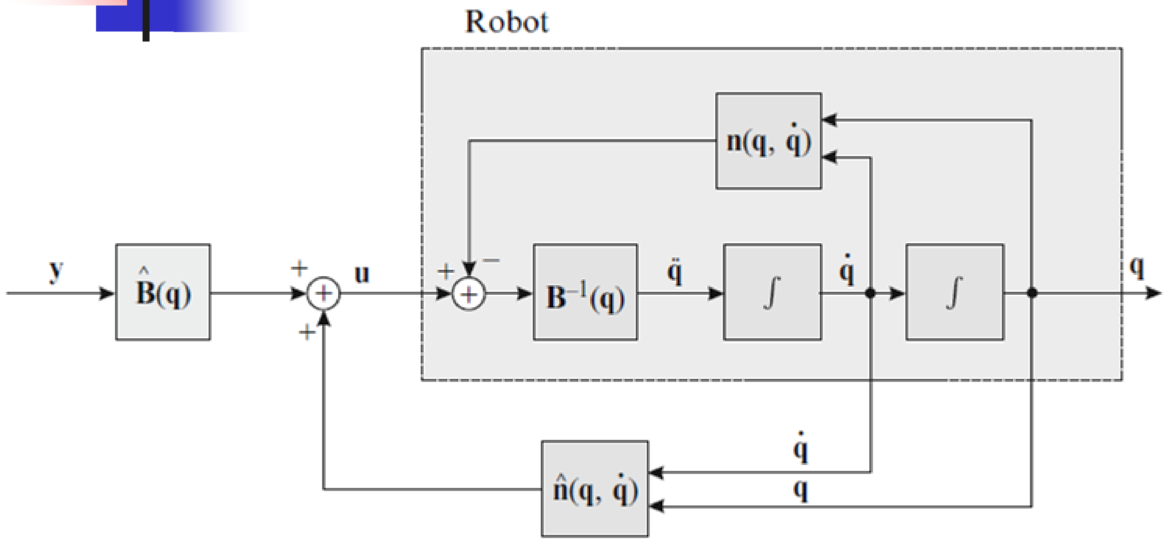
$$\ddot{q} = B^{-1}(q)\left(u - \left(C(q,\dot{q})\dot{q} + F_v\dot{q} + g(q)\right)\right).$$

$$n(q,\dot{q}) = C(q,\dot{q})\dot{q} + F_v\dot{q} + g(q).$$

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

$$\ddot{q} = B^{-1}(q)\left(u - n(q,\dot{q})\right).$$

Motion control in joint space based on inverse dynamics



. 7.7 Linearization of the control system by implementing the inverse dynamic model

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \mathbf{u}.$$

$$\ddot{\mathbf{q}} = \mathbf{B}^{-1}(\mathbf{q})\left(\mathbf{u} - \left(\mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q})\right)\right).$$

$$\mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}).$$

Let us assume that the robot dynamic model is known. The inertial matrix $\hat{\mathbf{B}}(\mathbf{q})$ is an approximation of the real values $\mathbf{B}(\mathbf{q})$, while $\hat{\mathbf{n}}(\mathbf{q},\dot{\mathbf{q}})$ represents an approximation of $\mathbf{n}(\mathbf{q},\dot{\mathbf{q}})$ as follows

$$\hat{\mathbf{n}}(\mathbf{q},\dot{\mathbf{q}}) = \hat{\mathbf{C}}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \hat{\mathbf{F}}_v\dot{\mathbf{q}} + \hat{\mathbf{g}}(\mathbf{q}).$$

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

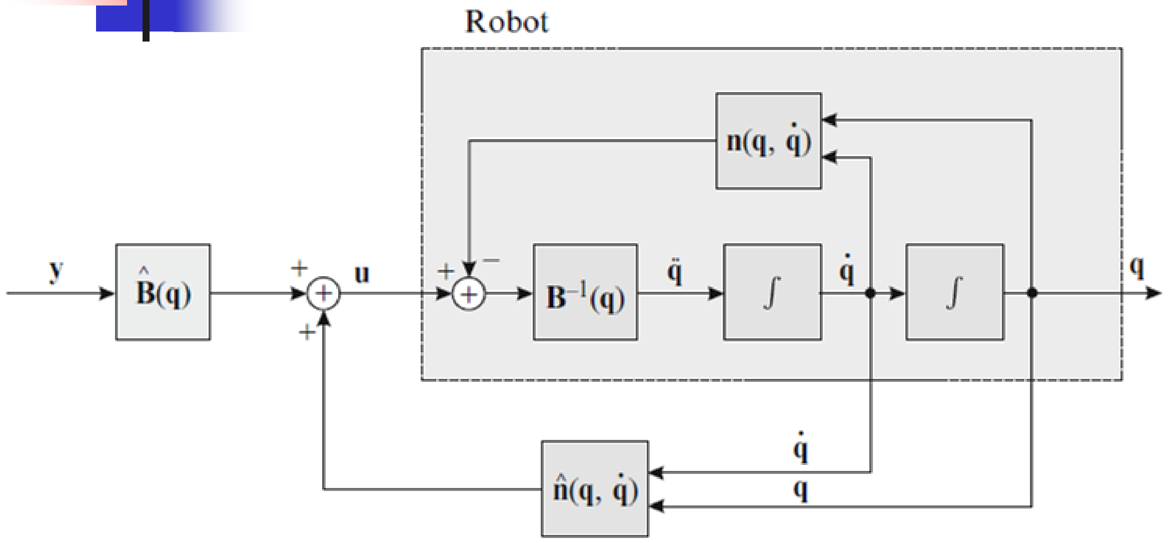
The controller output $\mathbf{u}$ is determined by the following equation

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

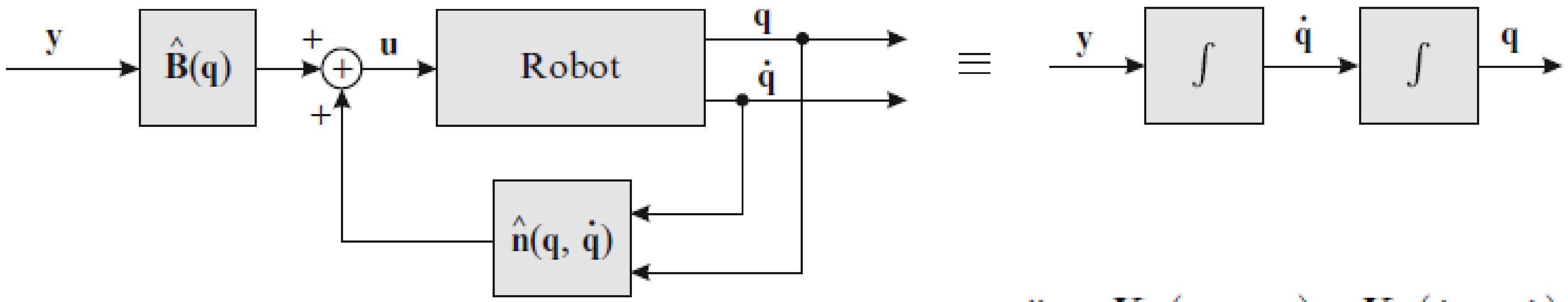
where the approximate inverse dynamic model of the robot was used.

$$\ddot{\mathbf{q}} = \mathbf{B}^{-1}(\mathbf{q})\left(\mathbf{u} - \mathbf{n}(\mathbf{q},\dot{\mathbf{q}})\right).$$

Motion control in joint space based on inverse dynamics



. 7.7 Linearization of the control system by implementing the inverse dynamic model



$$y = \ddot{q}_r + K_p(q_r - q) + K_d(\dot{q}_r - \dot{q}).$$

Motion control in joint space based on inverse dynamics

$$\mathbf{y} = \ddot{\mathbf{q}}_r + \mathbf{K}_p(\mathbf{q}_r - \mathbf{q}) + \mathbf{K}_d(\dot{\mathbf{q}}_r - \dot{\mathbf{q}}).$$

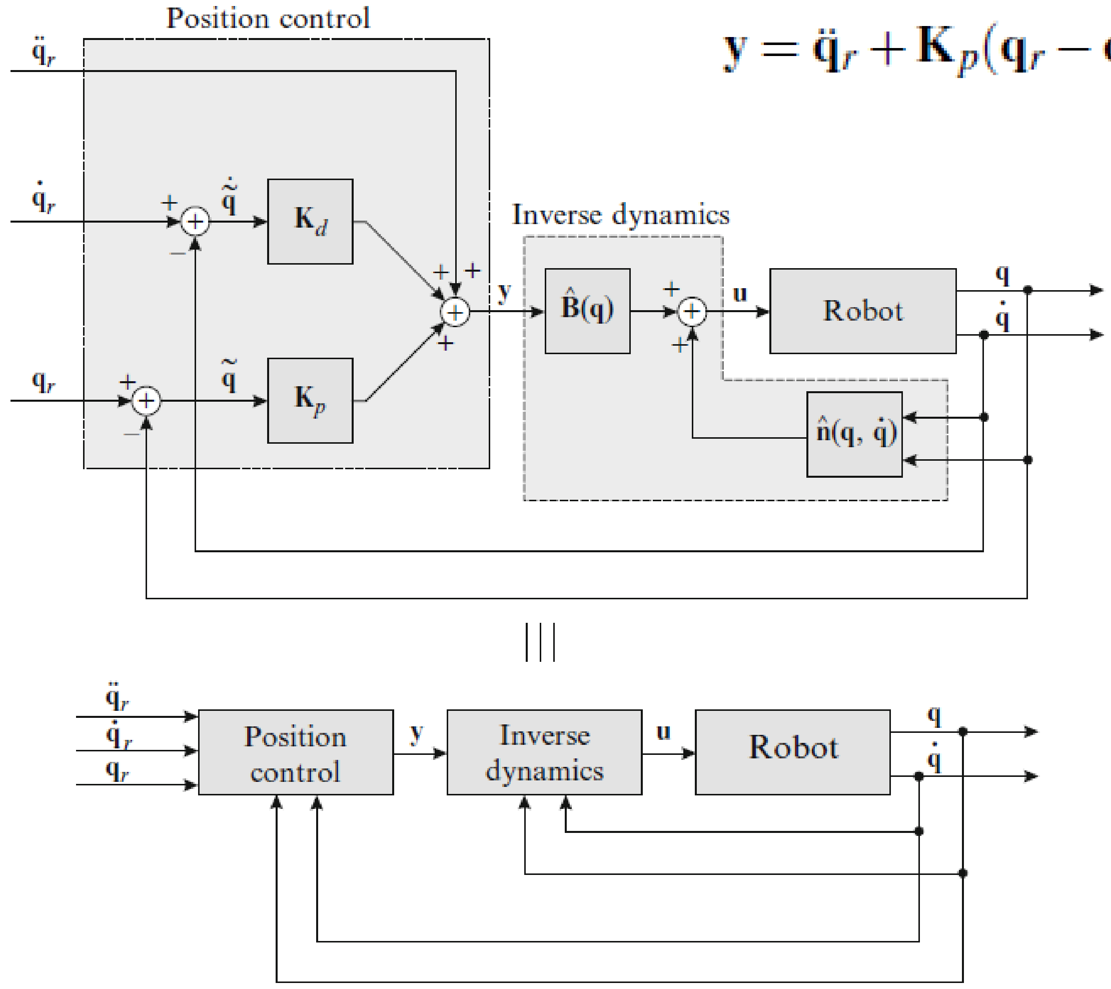


Fig. 7.9 Control of the robot based on inverse dynamics

Motion control in operational space

- In the movement from x_i to x_d the robot end effector follows a trajectory in the operational space, according to a planned time law.
- e.g. linear or curvilinear trajectory

Motion control in operational space

- To make the robot follow a trajectory

  $(t, p(t), \Phi(t), v(t), \omega(t))$
- To set joint velocities and accelerations in time, in order to reach the final desired position, expressed in Cartesian coordinates (Jacobian)
- To set voltages and currents to give to the motors in order to apply to the joints the velocities and the accelerations calculated with the Jacobian

Differential kinematics

Set the relations between the **joint velocities** and the corresponding **angular and linear velocities** of the end effector. Such relations are decribed in a transformation matrix (Jacobian) which depends on the robot configuration.

Differential kinematics

Geometric Jacobian = transformation matrix depending on the current robot configuration

$$v = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = J(q)\dot{q}$$

J(q) = geometric Jacobiano
$\dot{p}$ = linear velocity of the end effector
ω = angular velocity of the end effector
$\dot{q}$ = joint velocity

Differential kinematics

To find the joint velocities given the end effector velocity in operational space

$$v = \begin{bmatrix} \dot{p} \\ \omega \end{bmatrix} = J(q)\dot{q}$$

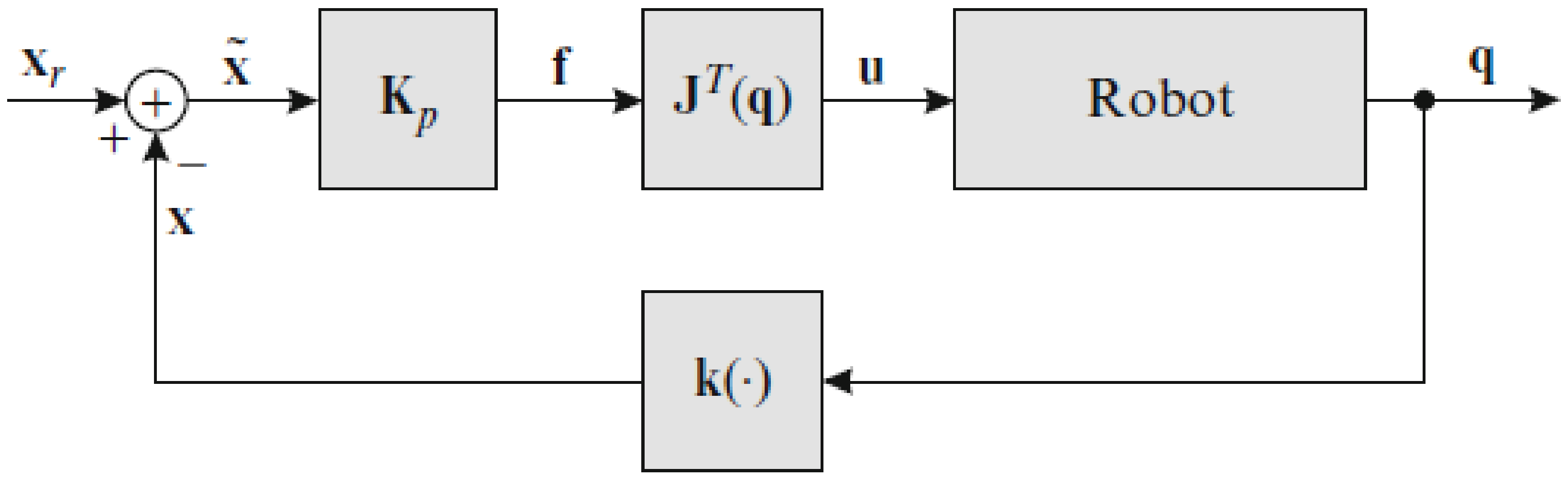
$$\dot{q} = J^{-1}(q)\begin{bmatrix} \dot{p} \\ \omega \end{bmatrix}$$

J^{-1} is the inverse Jacobian

Integral numerical methods allows to find the q vector from the vector of joint velocities

Motion control in operational space

based on the transposed Jacobian matrix



Control based on the transposed Jacobian matrix

$$\mathbf{f} = \mathbf{K}_p \tilde{\mathbf{x}}. \qquad \mathbf{u} = \mathbf{J}^T(\mathbf{q})\mathbf{f}.$$

f = force at end effector

Motion control in operational space

based on the inverse Jacobian matrix

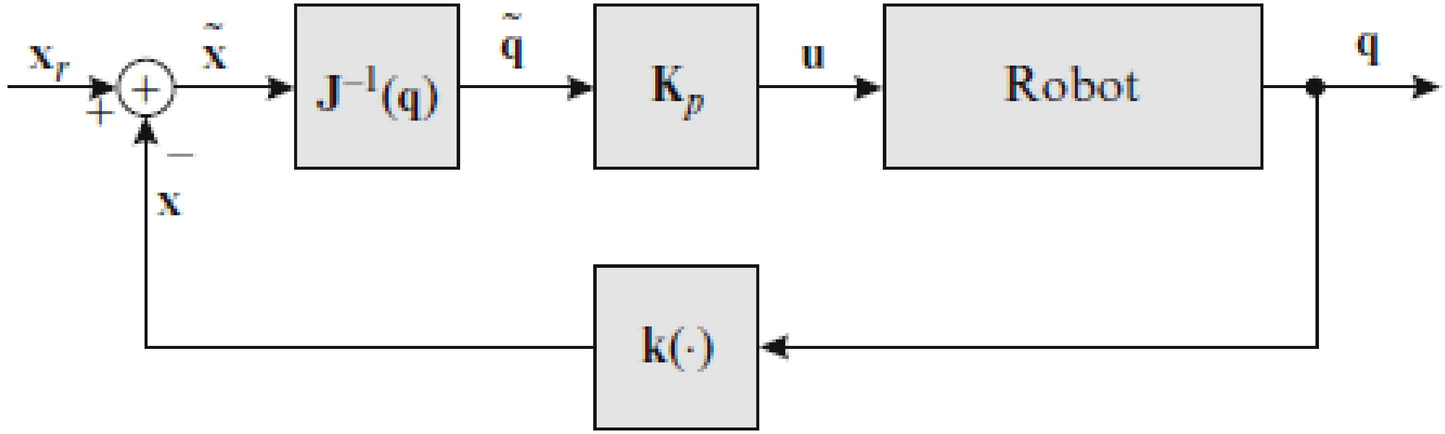


Fig. 7.11 Control based on the inverse Jacobian matrix

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \quad \Leftrightarrow \quad \frac{d\mathbf{x}}{dt} = \mathbf{J}(\mathbf{q})\frac{d\mathbf{q}}{dt}.$$

$$d\mathbf{x} = \mathbf{J}(\mathbf{q})d\mathbf{q}.$$

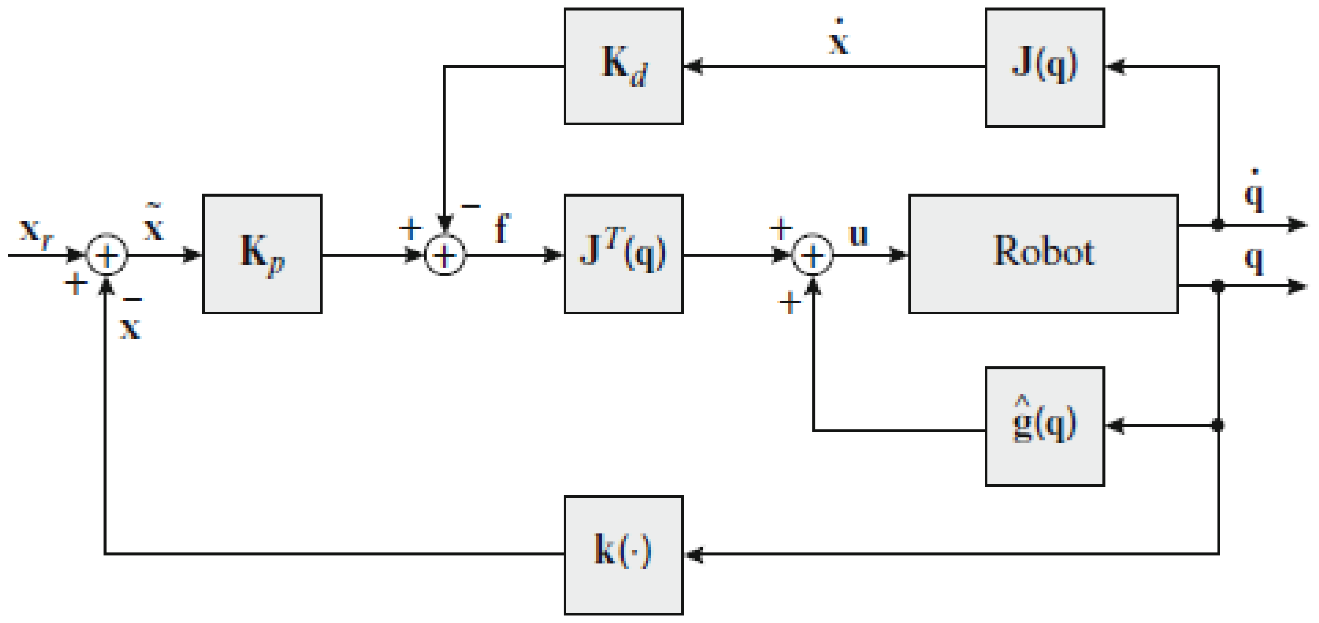
for small displacements

$$\tilde{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{q})\tilde{\mathbf{x}}.$$

$$\mathbf{u} = \mathbf{K}_p\tilde{\mathbf{q}}.$$

Motion control in operational space

PD control with gravity compensation



PD control with gravity compensation in external coordinates

$$\dot{x} = J(q)\dot{q}.$$

$$f = K_p\tilde{x} - K_d\dot{x}.$$

$$u = J^T(q)f + \hat{g}(q).$$

Motion control in operational space based on inverse dynamics

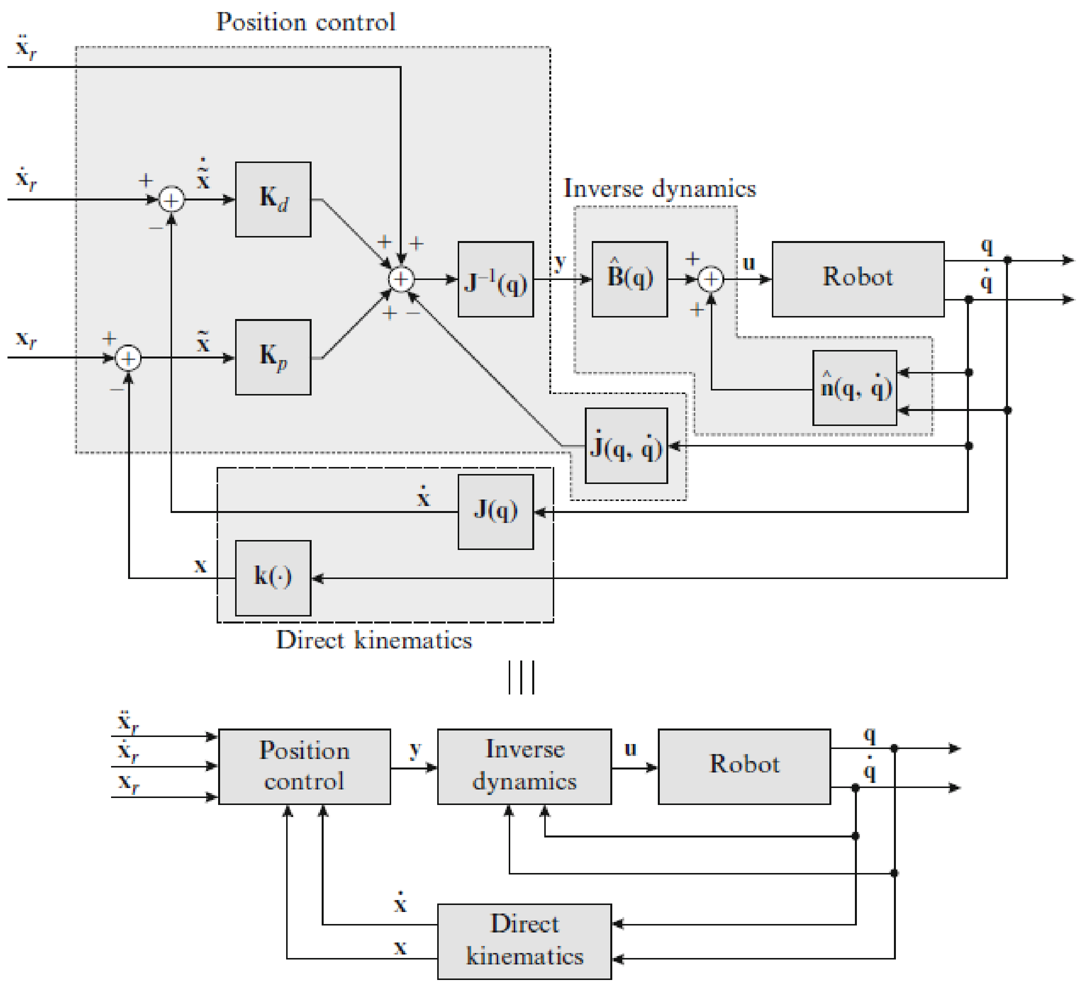


Fig. 7.13 Robot control based on inverse dynamics in external coordinates

$$\dot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}. \tag{7.33}$$

By calculating the time derivative of equation (7.33), we obtain

$$\ddot{\mathbf{x}} = \mathbf{J}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}. \tag{7.34}$$

The error of the pose of the robot end-effector is determined as the difference between its desired and its actual pose

$$\tilde{\mathbf{x}} = \mathbf{x}_r - \mathbf{x} = \mathbf{x}_r - \mathbf{k}(\mathbf{q}). \tag{7.35}$$

In a similar way the velocity error of the robot end-effector is determined

$$\dot{\tilde{\mathbf{x}}} = \dot{\mathbf{x}}_r - \dot{\mathbf{x}} = \dot{\mathbf{x}}_r - \mathbf{J}(\mathbf{q})\dot{\mathbf{q}}. \tag{7.36}$$

The acceleration error is the difference between the desired and the actual acceleration

$$\ddot{\tilde{\mathbf{x}}} = \ddot{\mathbf{x}}_r - \ddot{\mathbf{x}}. \tag{7.37}$$

When developing the inverse dynamics based controller in the internal coordinates, equation (7.19) was derived describing the dynamics of the control error in the form $\ddot{\tilde{\mathbf{q}}} + \mathbf{K}_d\dot{\tilde{\mathbf{q}}} + \mathbf{K}_p\tilde{\mathbf{q}} = \mathbf{0}$. An analogous equation can be written for the error of the end-effector pose. From this equation the acceleration $\ddot{\mathbf{x}}$ of the robot end-effector can be expressed

$$\ddot{\tilde{\mathbf{x}}} + \mathbf{K}_d\dot{\tilde{\mathbf{x}}} + \mathbf{K}_p\tilde{\mathbf{x}} = \mathbf{0} \quad \Rightarrow \quad \ddot{\mathbf{x}} = \ddot{\mathbf{x}}_r + \mathbf{K}_d\dot{\tilde{\mathbf{x}}} + \mathbf{K}_p\tilde{\mathbf{x}}. \tag{7.38}$$

From equation (7.34) we express $\ddot{\mathbf{q}}$ taking into account the equality $\mathbf{y} = \ddot{\mathbf{q}}$

$$\mathbf{y} = \mathbf{J}^{-1}(\mathbf{q})\left(\ddot{\mathbf{x}} - \dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}\right). \tag{7.39}$$

By replacing $\ddot{\mathbf{x}}$ in equation (7.39) with expression (7.38), the control algorithm based on inverse dynamics in the external coordinates is obtained

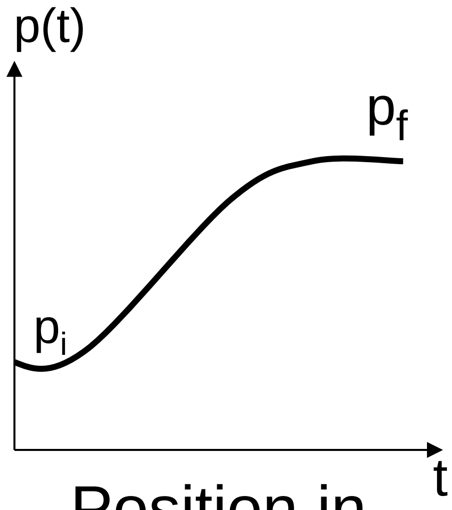
$$\mathbf{y} = \mathbf{J}^{-1}(\mathbf{q})\left(\ddot{\mathbf{x}}_r + \mathbf{K}_d\dot{\tilde{\mathbf{x}}} + \mathbf{K}_p\tilde{\mathbf{x}} - \dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}})\dot{\mathbf{q}}\right). \tag{7.40}$$

The control scheme encompassing the linearization of the system based on inverse dynamics (7.31) and the closed loop control (7.40) is shown in Figure 7.13.

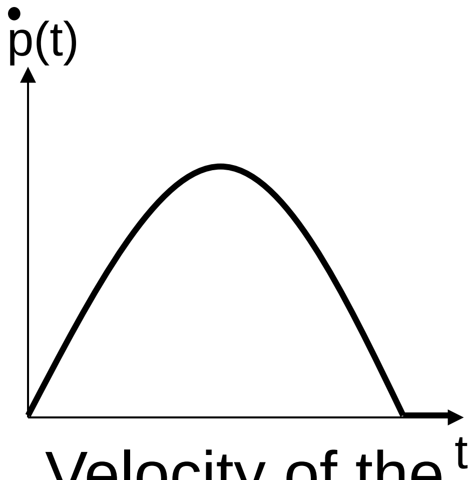
Motion control in operational space

Trajectory planning

Joint velocities

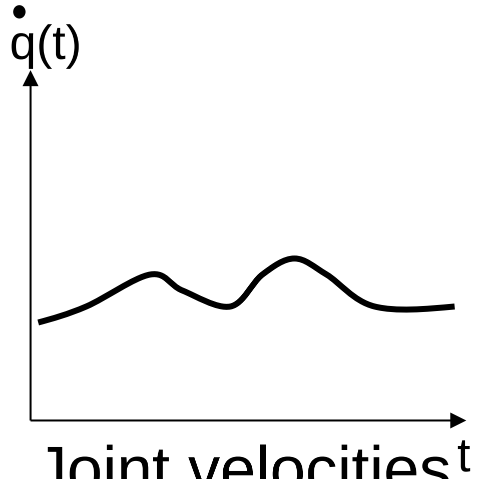


Position in operational space, in time



Velocity of the end effector, in time

$J^{-1}(p(t))$ $\longrightarrow$



Joint velocities, in time

$(t, p(t), \Phi(t), \dot{p}(t), \omega(t)) \xrightarrow{J^{-1}(p(t))} (t, \dot{q}(t))$

Force control

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}_v\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}) = \boldsymbol{\tau} - \mathbf{J}^T(\mathbf{q})\mathbf{f},$$

Effect of external forces

$$\mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) = \mathbf{C}(\mathbf{q},\dot{\mathbf{q}})\dot{\mathbf{q}} + \mathbf{F}\dot{\mathbf{q}} + \mathbf{g}(\mathbf{q}),$$

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) = \boldsymbol{\tau} - \mathbf{J}^T(\mathbf{q})\mathbf{f}.$$

$$\mathbf{B}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{J}^T(\mathbf{q})\mathbf{f} = \mathbf{u}.$$

$$\ddot{\mathbf{q}} = \mathbf{B}^{-1}(\mathbf{q})\left(\mathbf{u} - \mathbf{n}(\mathbf{q},\dot{\mathbf{q}}) - \mathbf{J}^T(\mathbf{q})\mathbf{f}\right).$$

$$\mathbf{u} = \hat{\mathbf{B}}(\mathbf{q})\mathbf{y} + \hat{\mathbf{n}}(\mathbf{q},\dot{\mathbf{q}}) + \mathbf{J}^T(\mathbf{q})\mathbf{f},$$

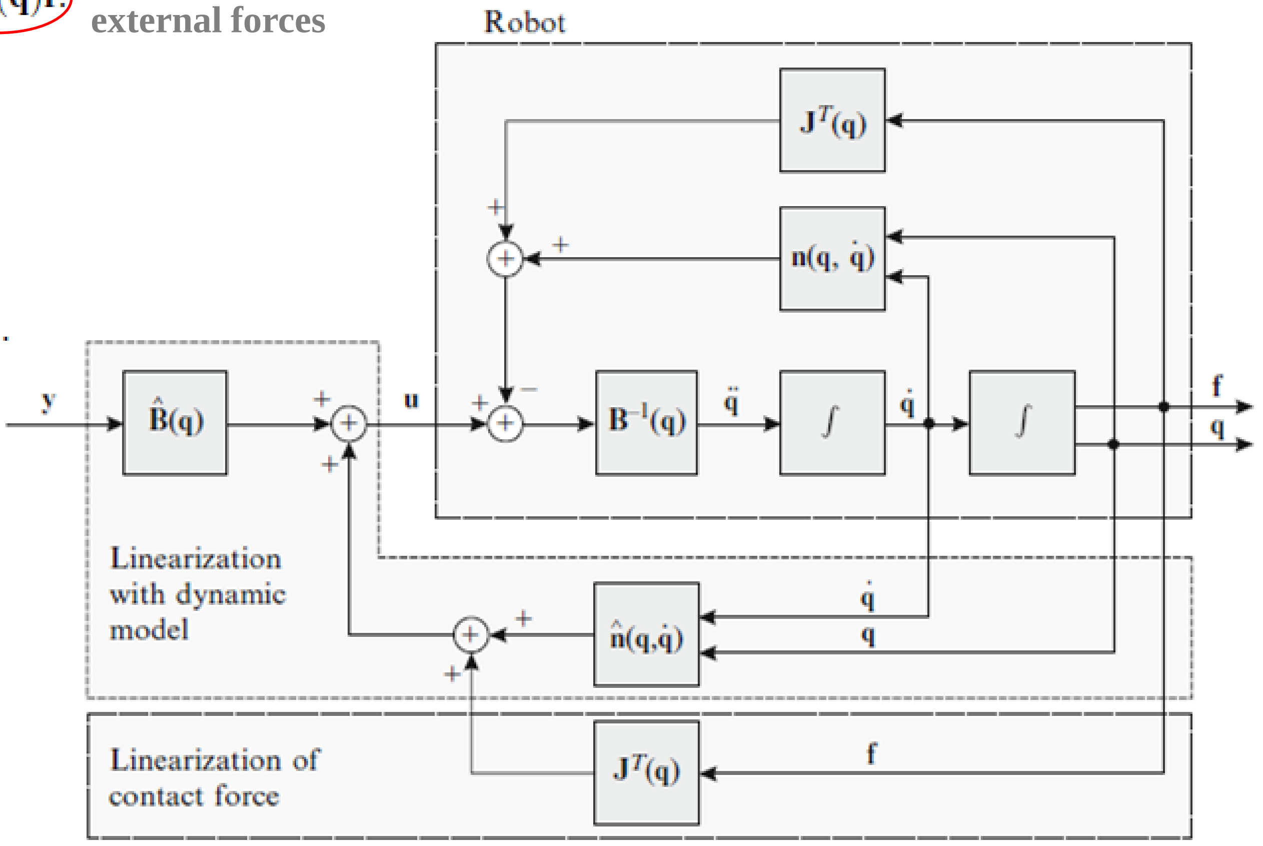

Robot
$\mathbf{J}^T(\mathbf{q})$
$\mathbf{n}(\mathbf{q},\dot{\mathbf{q}})$
$\mathbf{y}$
$\hat{\mathbf{B}}(\mathbf{q})$
$\mathbf{u}$
$\mathbf{B}^{-1}(\mathbf{q})$
$\ddot{\mathbf{q}}$
$\int$
$\dot{\mathbf{q}}$
$\int$
$\mathbf{f}$
$\mathbf{q}$
Linearization with dynamic model
$\hat{\mathbf{n}}(\mathbf{q},\dot{\mathbf{q}})$
Linearization of contact force
$\mathbf{J}^T(\mathbf{q})$

Fig. 7.14 Linearization of the control system by implementing the inverse dynamic model and the measured contact force

Force control

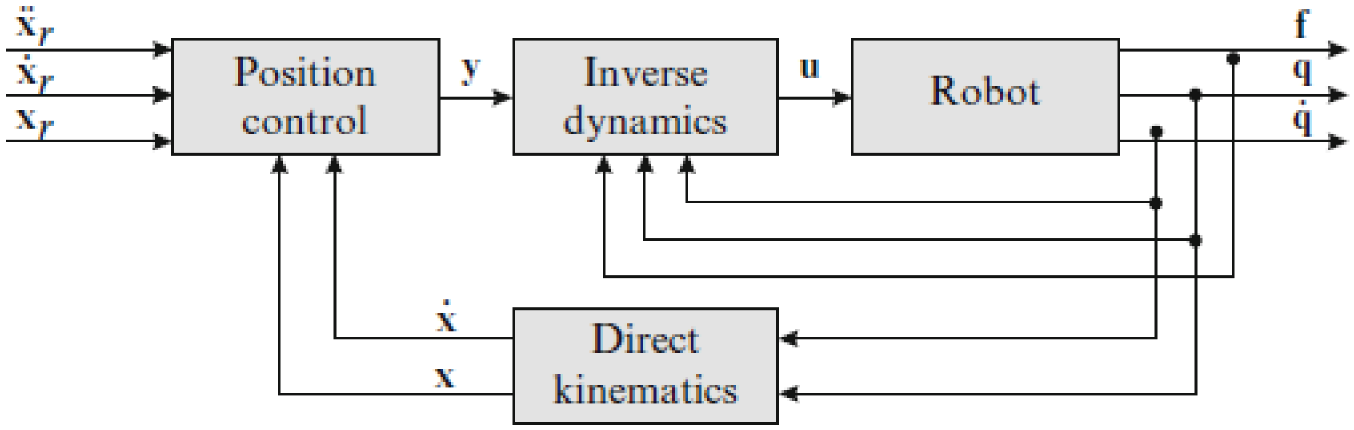


Fig. 7.15 Robot control based on inverse dynamics in external coordinates including the contact force

Force control

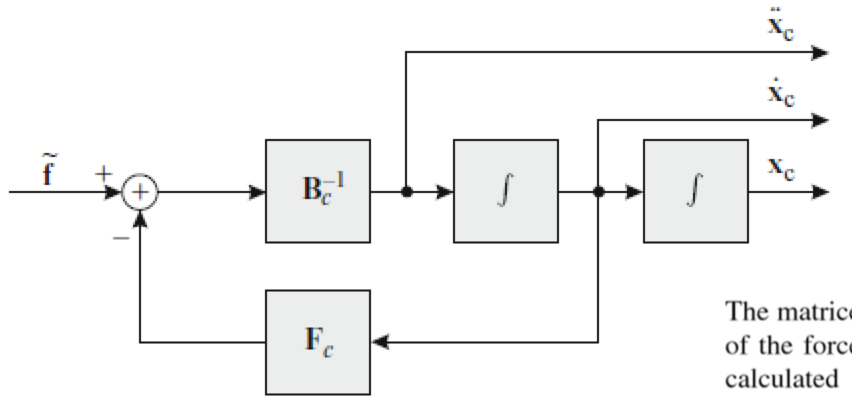


Fig. 7.16 Force control translated into control of the pose of robot end-effector

$$\tilde{\mathbf{f}} = \mathbf{B}_c \ddot{\mathbf{x}}_c + \mathbf{F}_c \dot{\mathbf{x}}_c. \tag{7.48}$$

The matrices $\mathbf{B}_c$ and $\mathbf{F}_c$ determine the movement of the object under the influence of the force $\tilde{\mathbf{f}}$. From equation (7.48) the acceleration of the virtual object can be calculated

$$\ddot{\mathbf{x}}_c = \mathbf{B}_c^{-1}\left(\tilde{\mathbf{f}} - \mathbf{F}_c \dot{\mathbf{x}}_c\right). \tag{7.49}$$

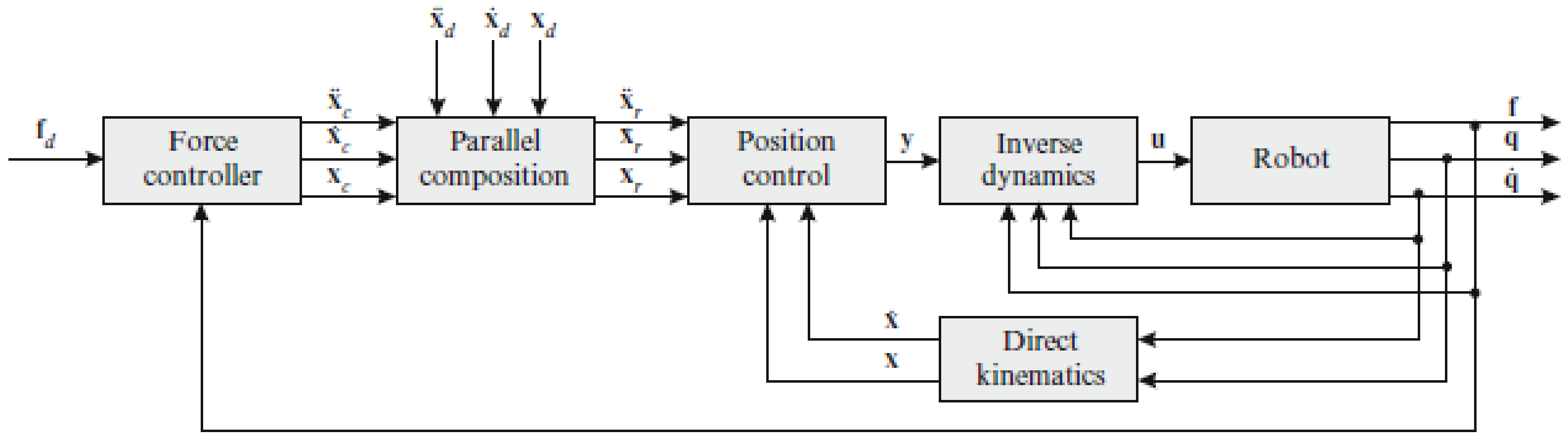


Fig. 7.17 Direct force control in the external coordinates

Performance of a manipulator

- **Payload**: maximum load
- **Velocity**: maximum velocity in operational space
- **Accuracy:** difference between the position calculated by the control system and the actual position
- **Repeatability:** measure of the robot capability to reach the same position (function of the control system and algorithm, in addition to the robot characteristics)

KUKA KR 15/2

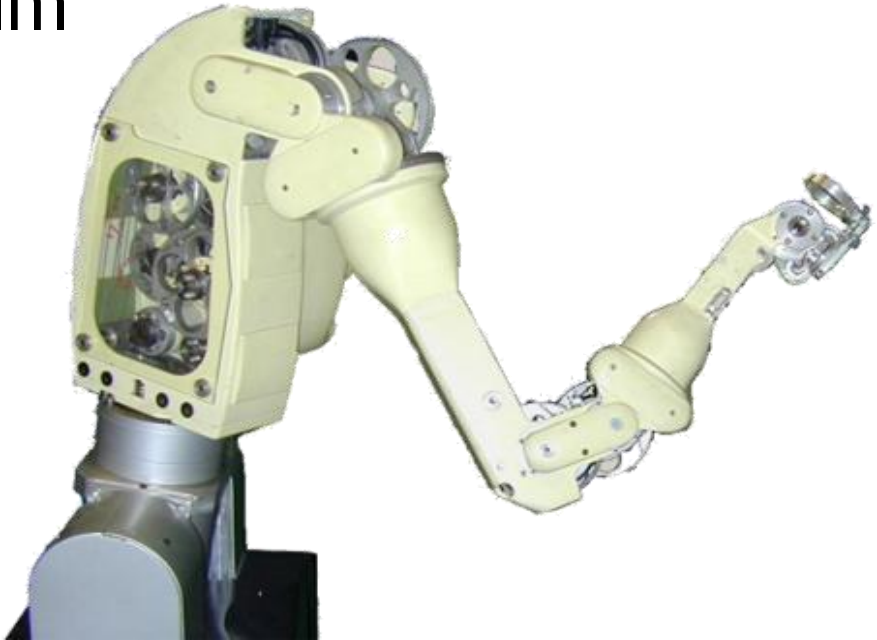
- Dof: 6
- Payload: 15 kg
- Max. reach: 1570 mm
- Repeatability: <± 0.1 mm
- Weight: 222 kg

PUMA 560

- Dof: 6
- Payload: 2 kg
- Velocity: 1.0 m/s
- Repeatability: <± 0.1 mm
- Weight: 120 lb = 55 Kg

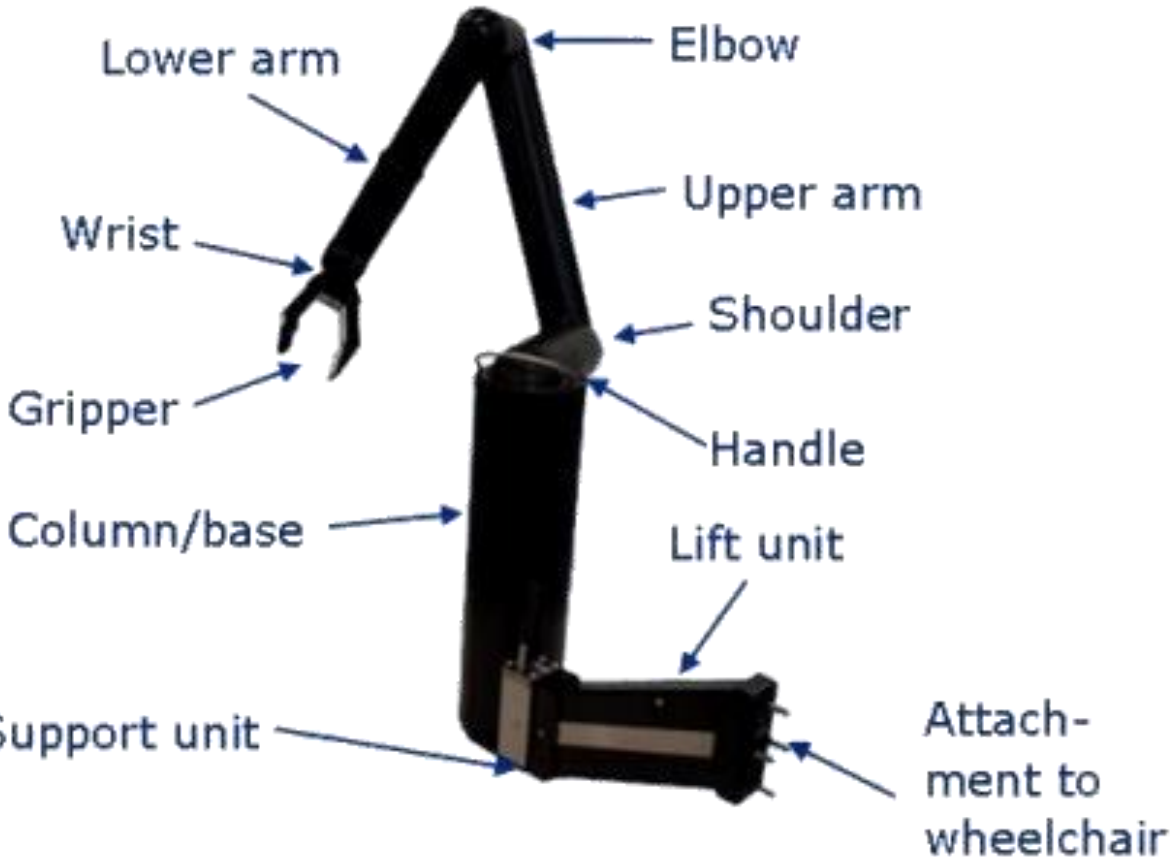
Dexter Arm

- Cable actuated
- d.o.f.: 8
- Workspace: 1200 mm x 350°
- Repeatability: ± 1mm
- Velocity: 0.2 m/s
- Payload: 2 Kg
- Weight: 40 Kg



Manus

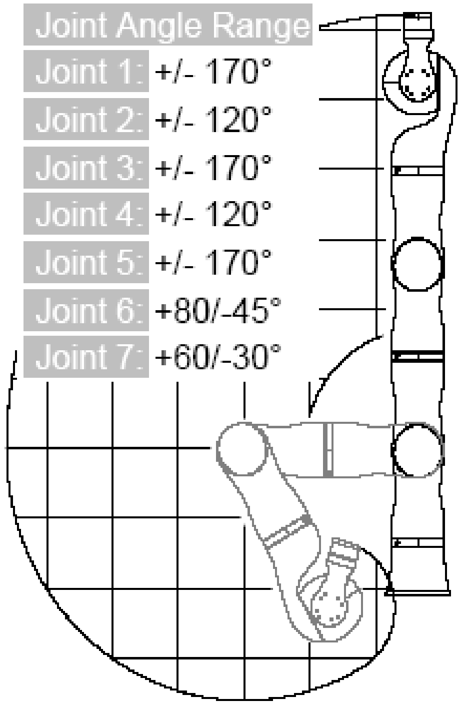
- Cable actuated
- d.o.f.: 6
- Velocity: 0.2 m/s
- Payload: 2 Kg
- Power: 24V DC
- Weight: 12 Kg

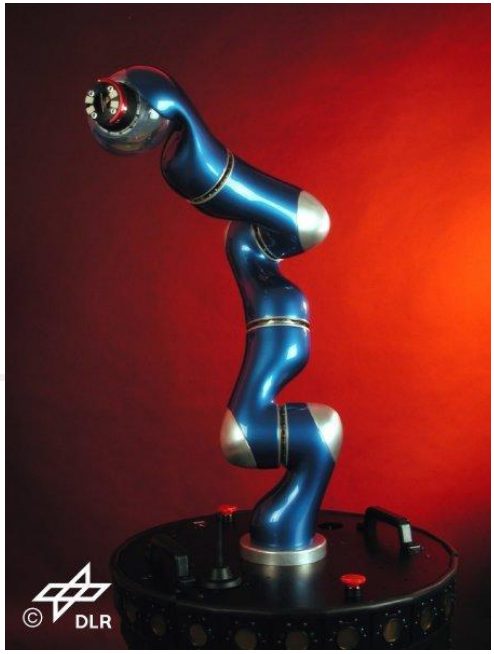


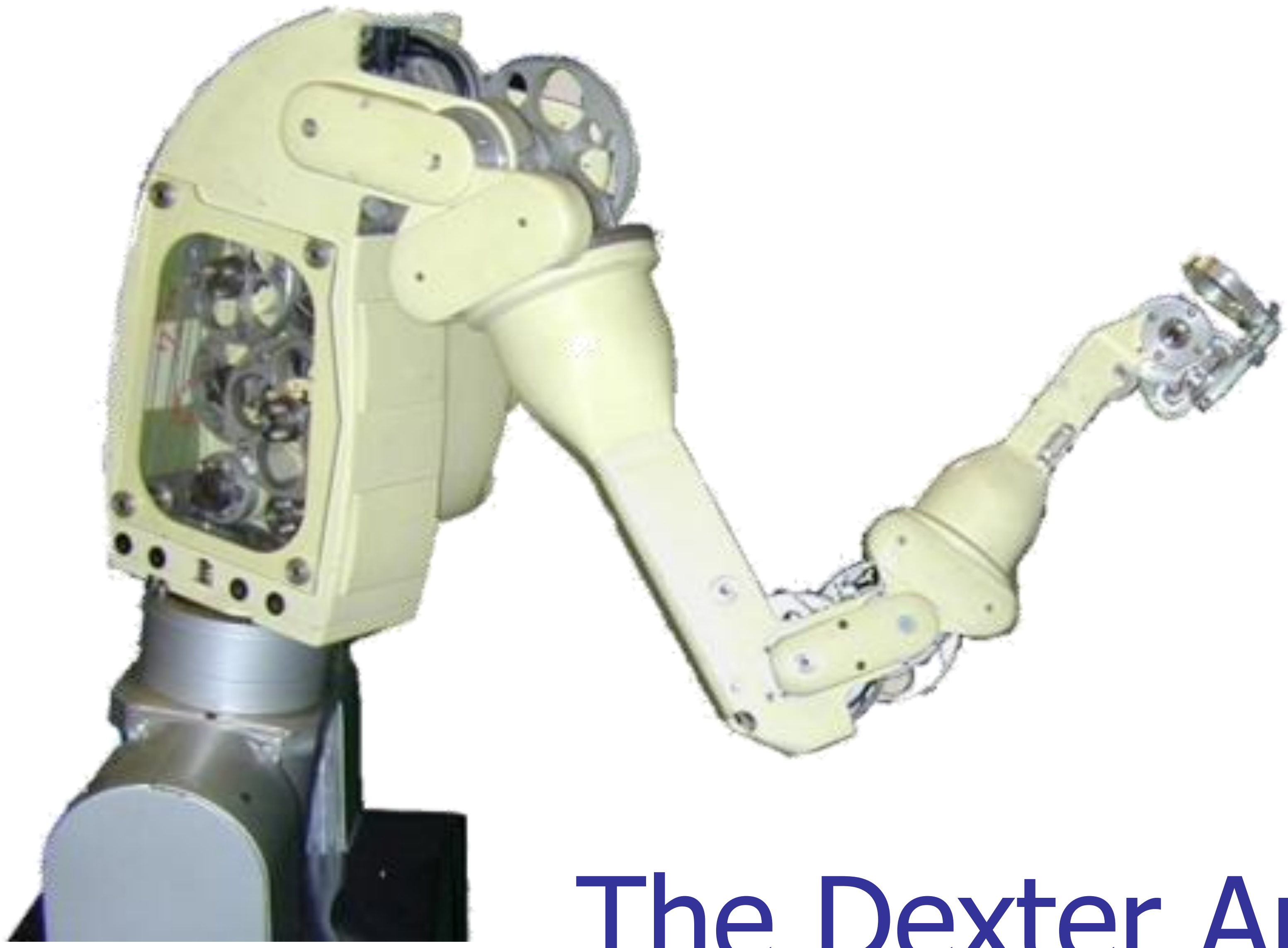
DLR Arm

Total Weight	14 kg
Max. Payload	14 kg
Max. Joint Speed	120°/s
Nr. of Axes	7 (R - P - R - P - R - P - P)
Maximum Reach	936 mm
Motors	DLR-Robodrive
Gears	Harmonic Drive
Sensors (each Joint)	2 Position, 1 Torque Sensor
Sensor (wrist)	6-DOF Force/Torque Sensor
Brakes	Electromagnetic Safty Brake
Power Supply	48 V DC
Control	Position-, Torque,- Impedance Control Control Cycles: Current 40 kHz; Joint 3 kHz; Cartesian 1 kHz
Electronics	Integrated Electronics, internal Cabling, Communications by optical SERCOS-Bus

Joint Angle Range	
Joint 1:	+/- 170°
Joint 2:	+/- 120°
Joint 3:	+/- 170°
Joint 4:	+/- 120°
Joint 5:	+/- 170°
Joint 6:	+80/-45°
Joint 7:	+60/-30°



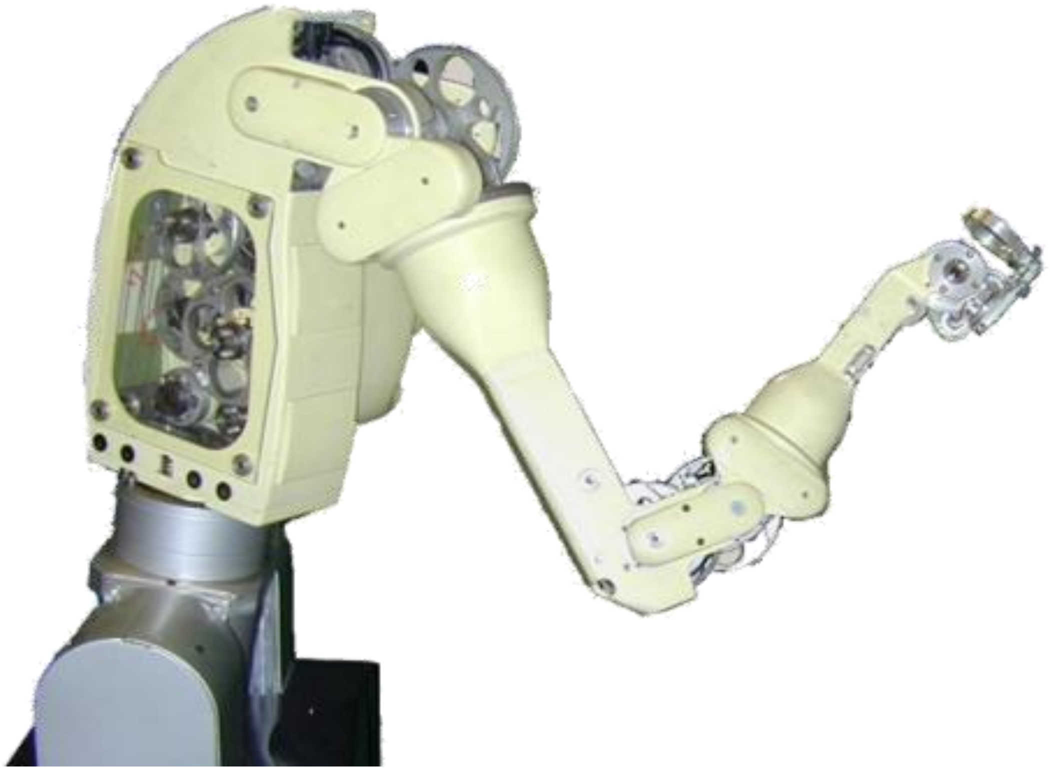




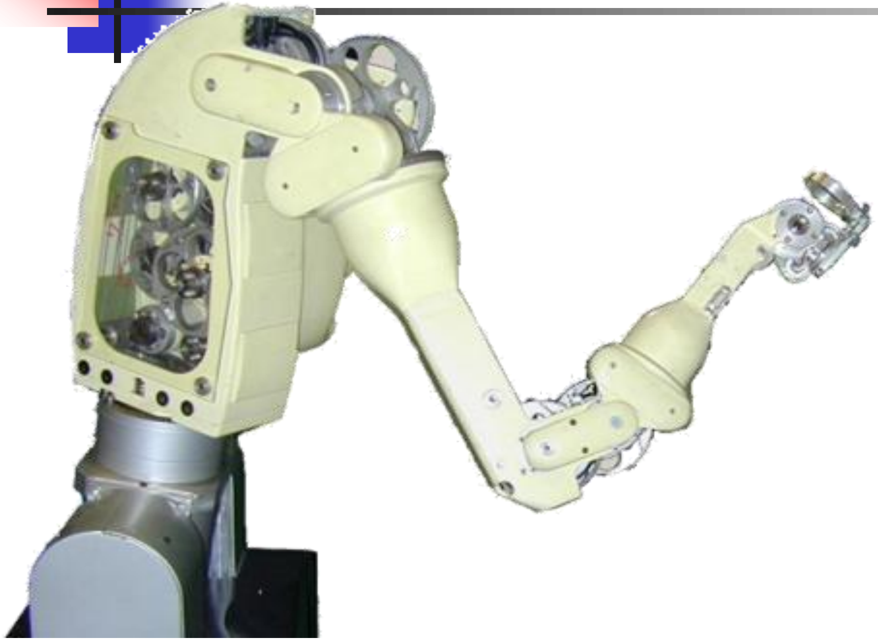
The Dexter Arm

The Dexter Arm

Workspace:	1200 mm x 350°
Repeatability:	$\underline{+}$ 1mm
Velocity:	0.2 m/s
Payload:	2 Kg
D.o.f.:	8
Power:	24V DC
Weight:	40 Kg



The Dexter Arm

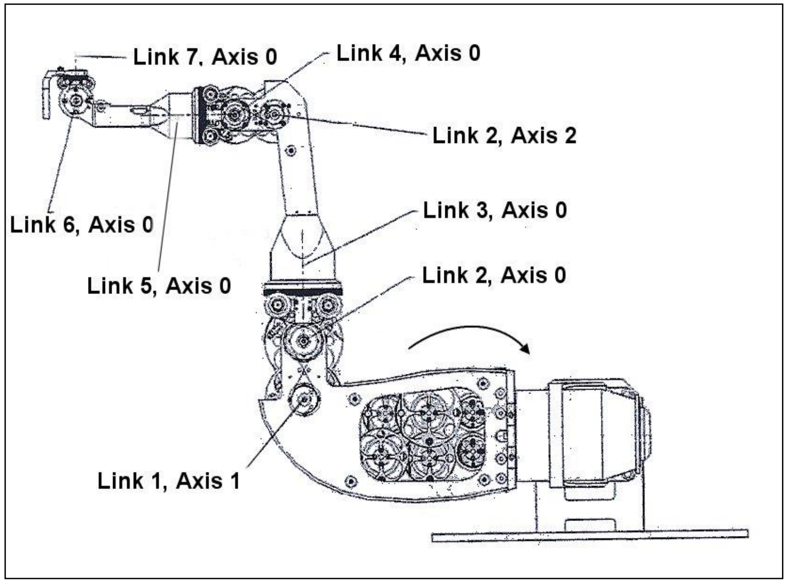


- 8-d.o.f. anthropomorphic redundant robot arm, composed of trunk, shoulder, elbow and wrist
- designed for service applications and personal assistance in residential sites, such as houses or hospitals

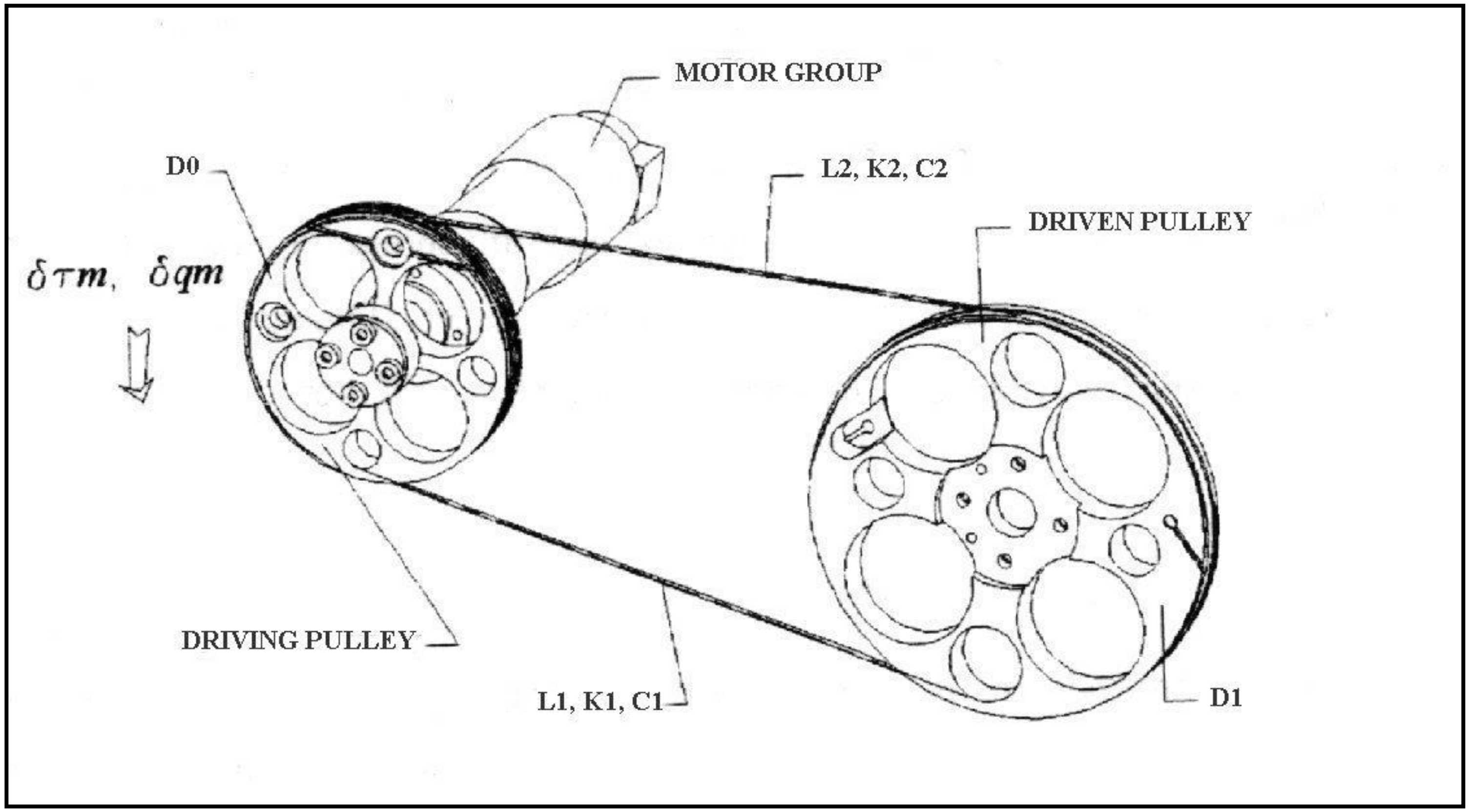
- mechanically coupled structure: the mechanical transmission system is realized with pulleys and steel cables
- main characteristics: reduced accuracy,lighter mechanical structure, safe and intrinsically compliant structure

The Dexter arm

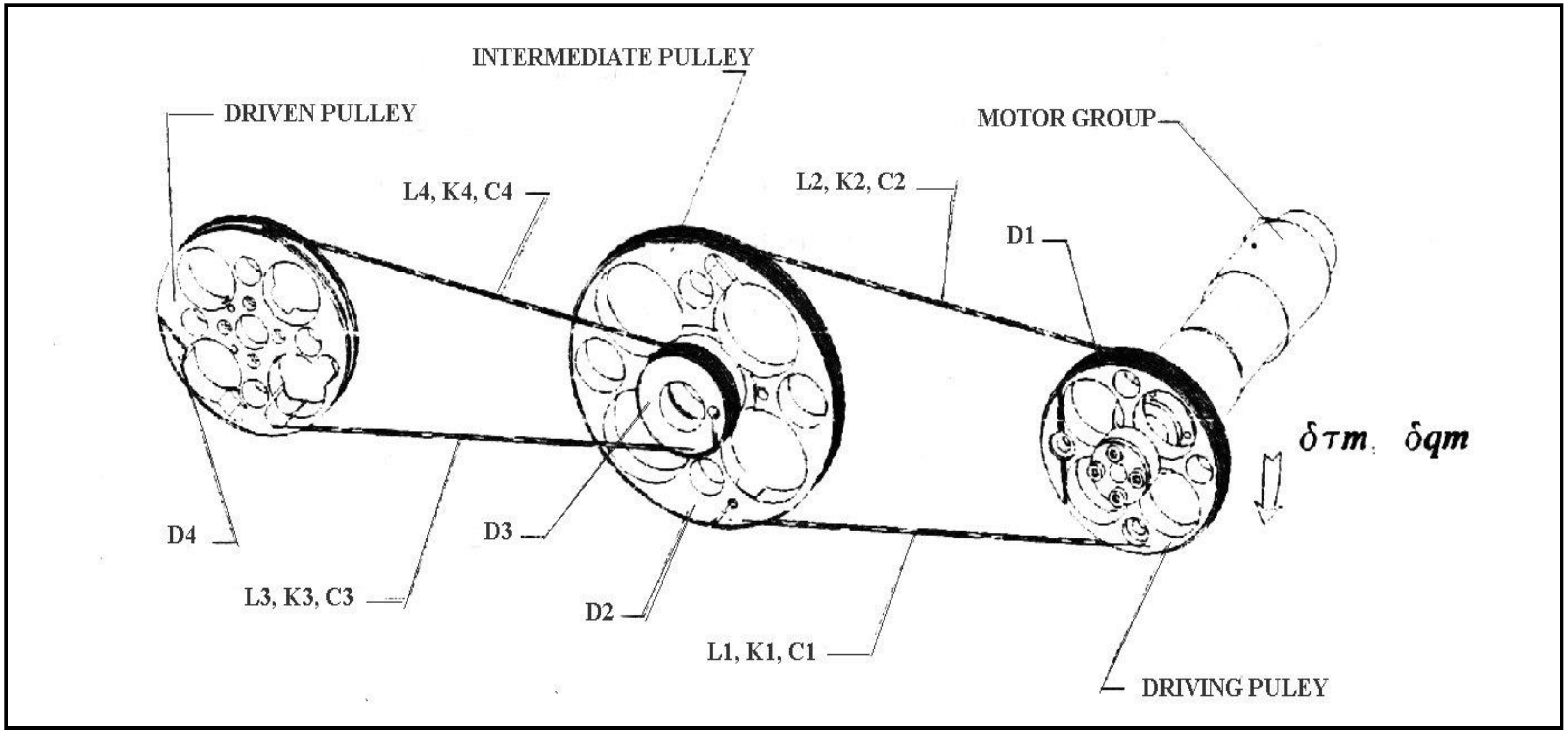
- Transmission system realized with pulleys and steel cables
- Joints J0 and J1 are actuated by motors and driving gear-boxes directly connected to the articulation axis
- Joints J2,..,J7 are actuated by DC-motors installed on link 1



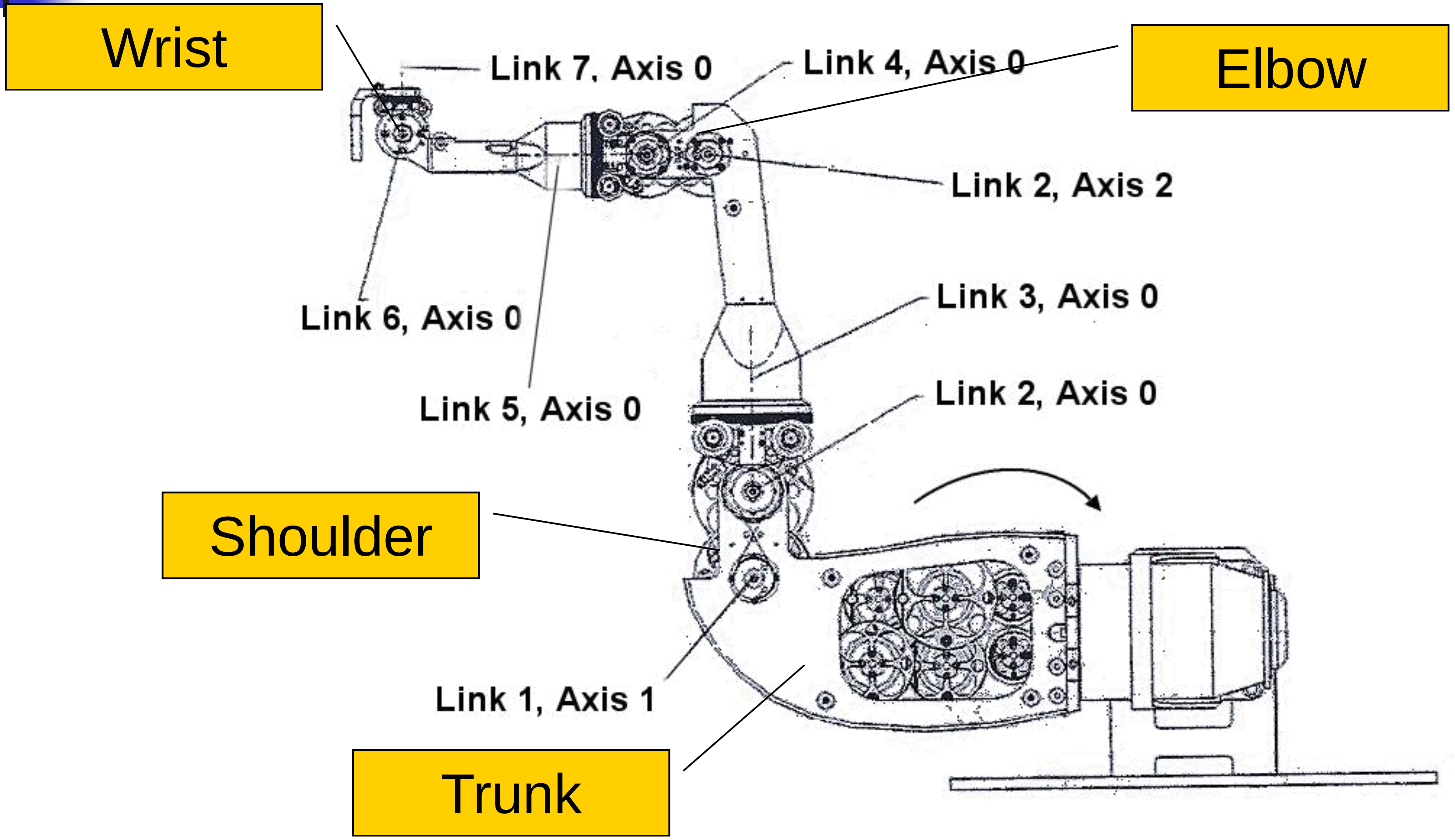
Transmission #6



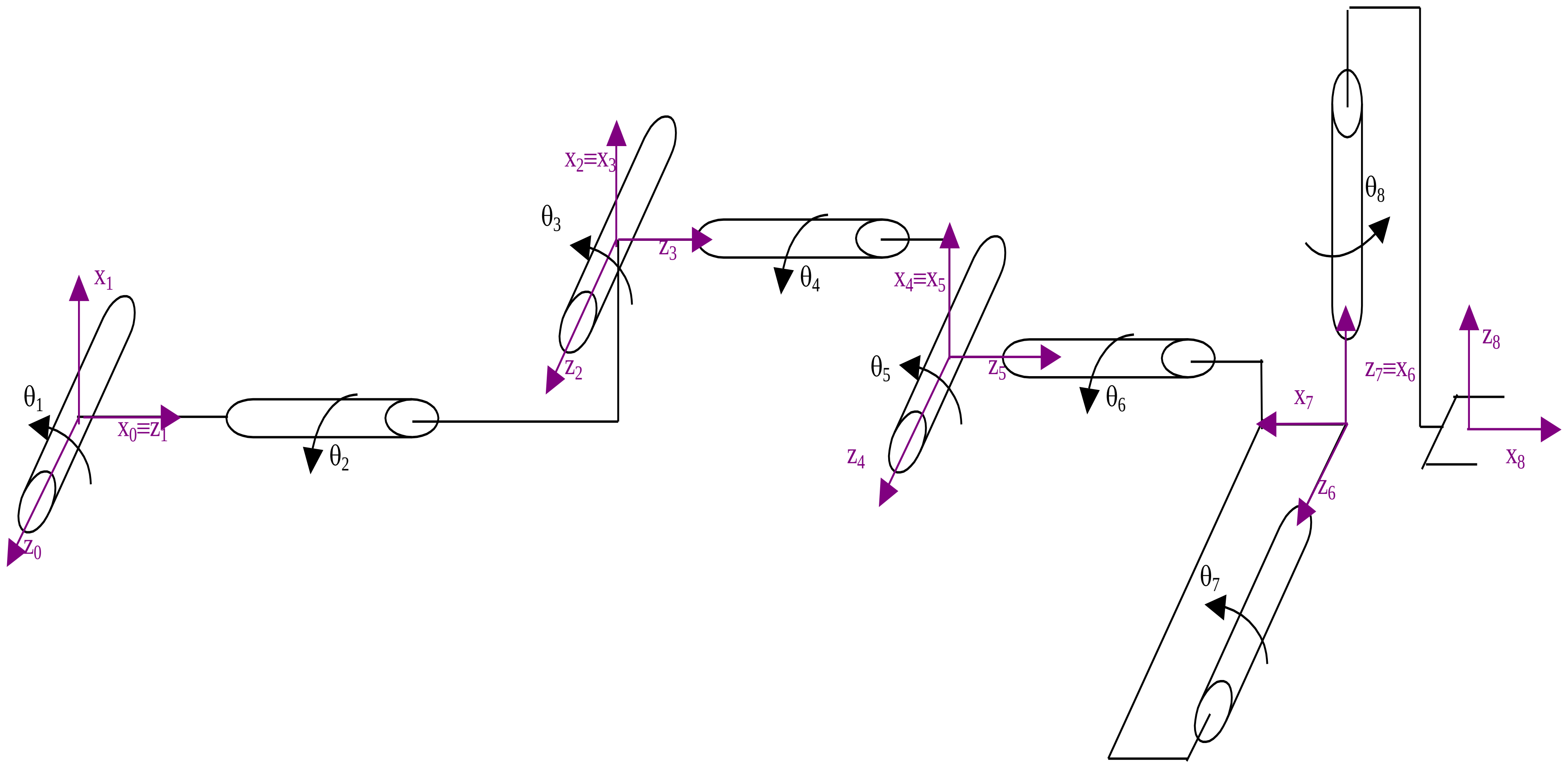
Transmissions #2-5 and 7



Anthropomorphic structure



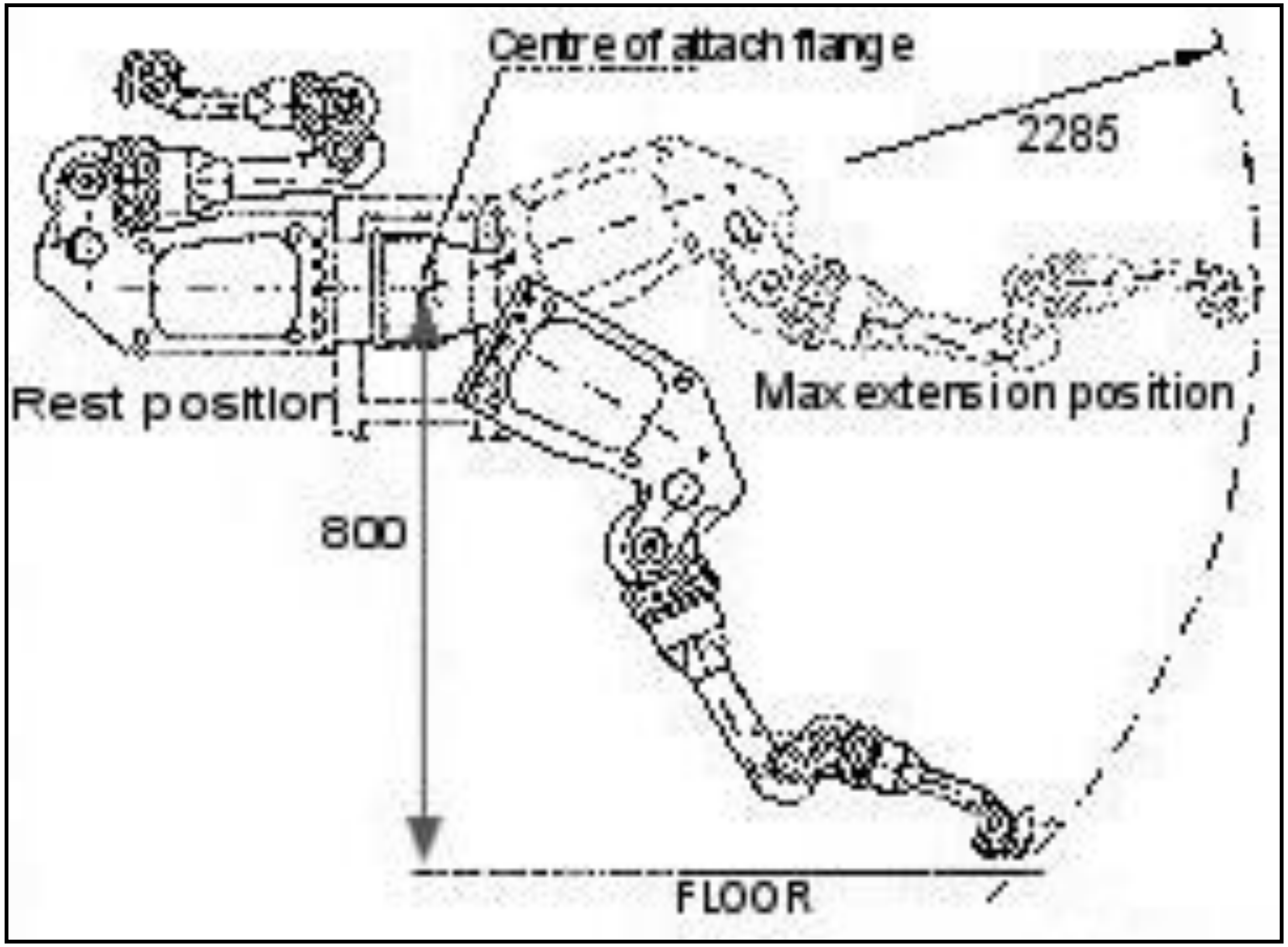
Kinematic Configuration



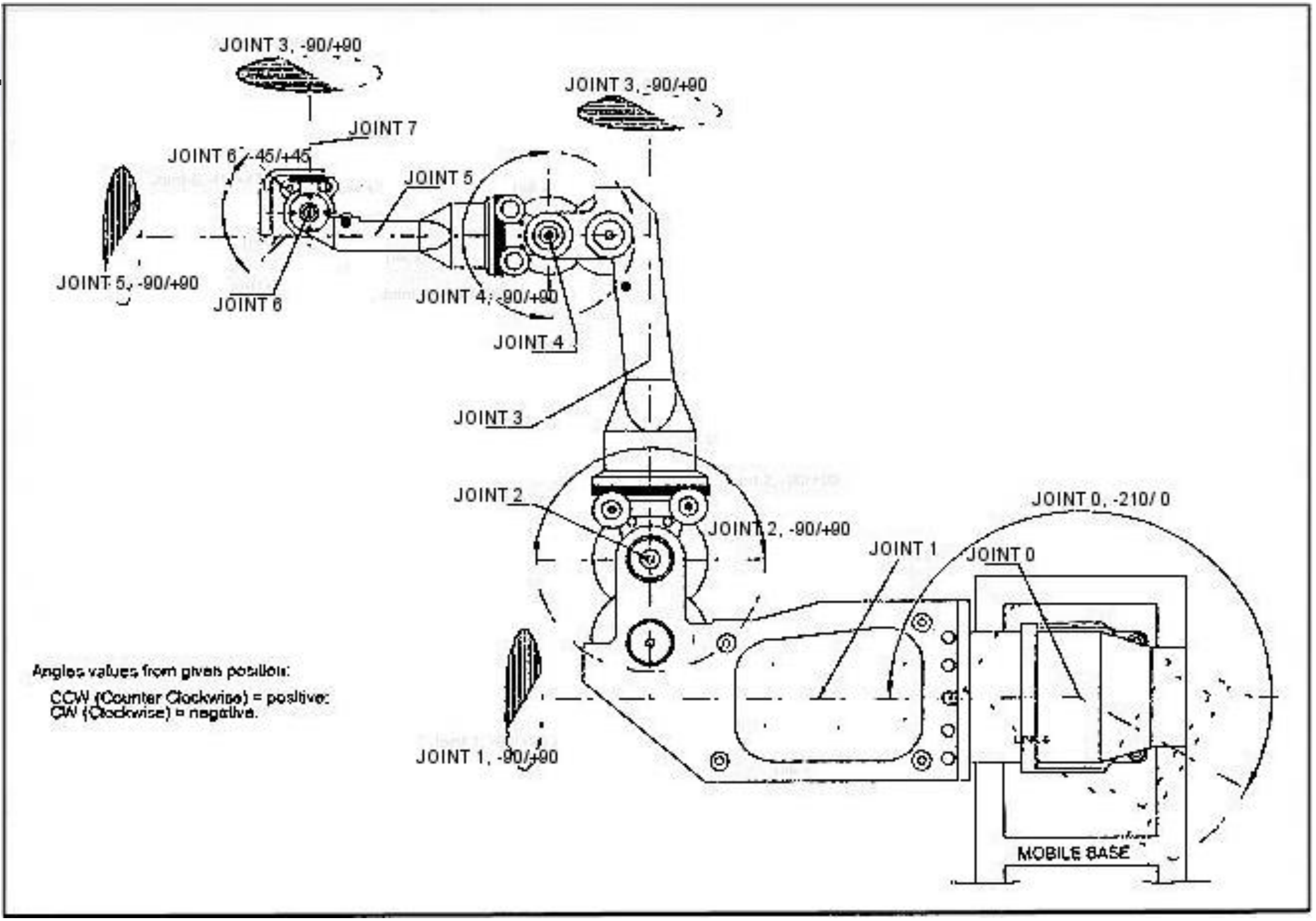
Denavit-Hartenberg Parameters

Joint	a_i [mm]	d_i [mm]	α_i [rad]	θ_i [rad]
1	0	0	$\pi/2$	θ_1
2	144	450	$-\pi/2$	θ_2
3	0	0	$\pi/2$	θ_3
4	-100	350	$-\pi/2$	θ_4
5	0	0	$\pi/2$	θ_5
6	-24	250	$-\pi/2$	θ_6
7	0	0	$\pi/2$	θ_7
8	100	0	0	θ_8

The Dexter Workspace



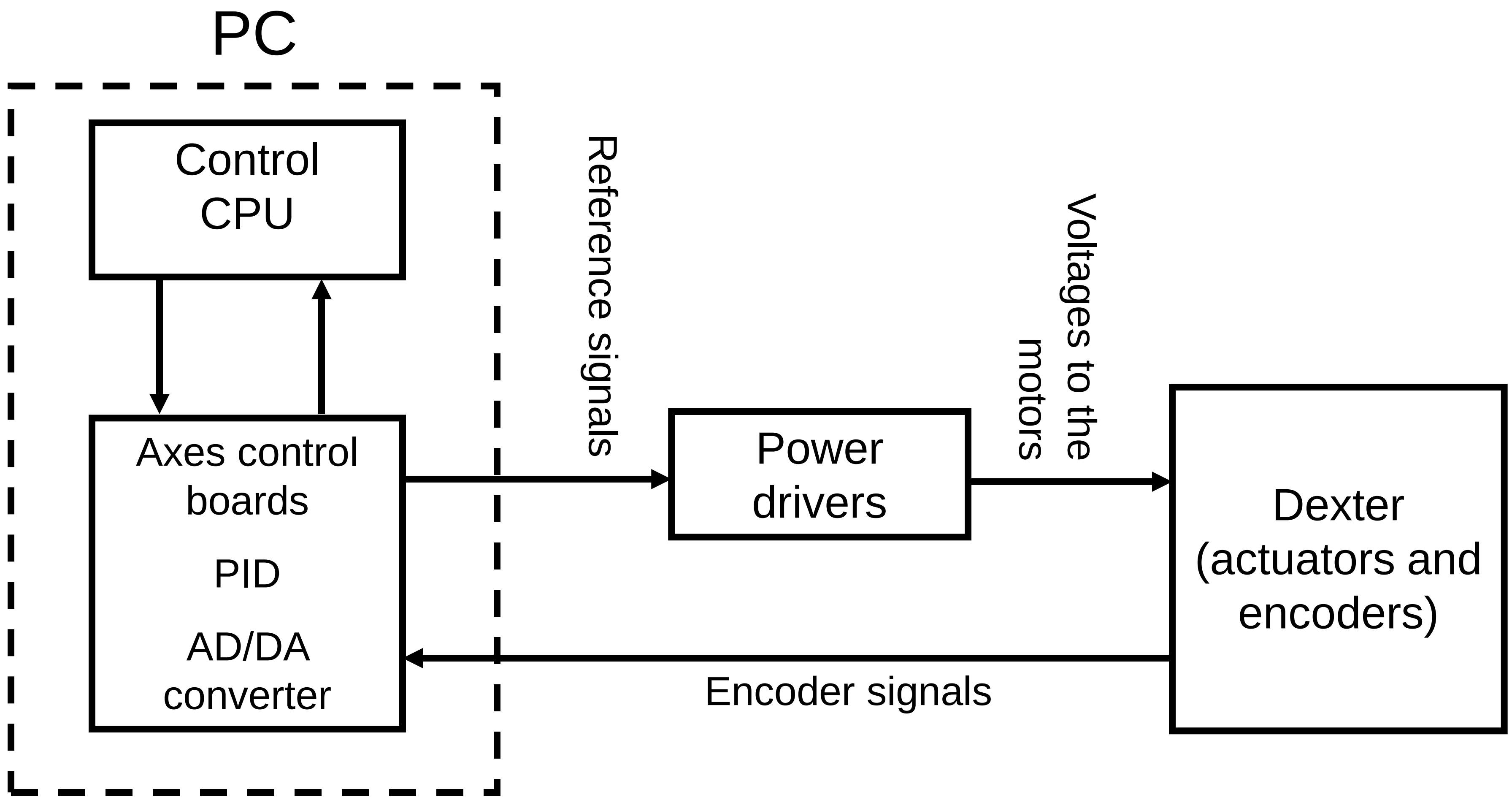
Joint Ranges



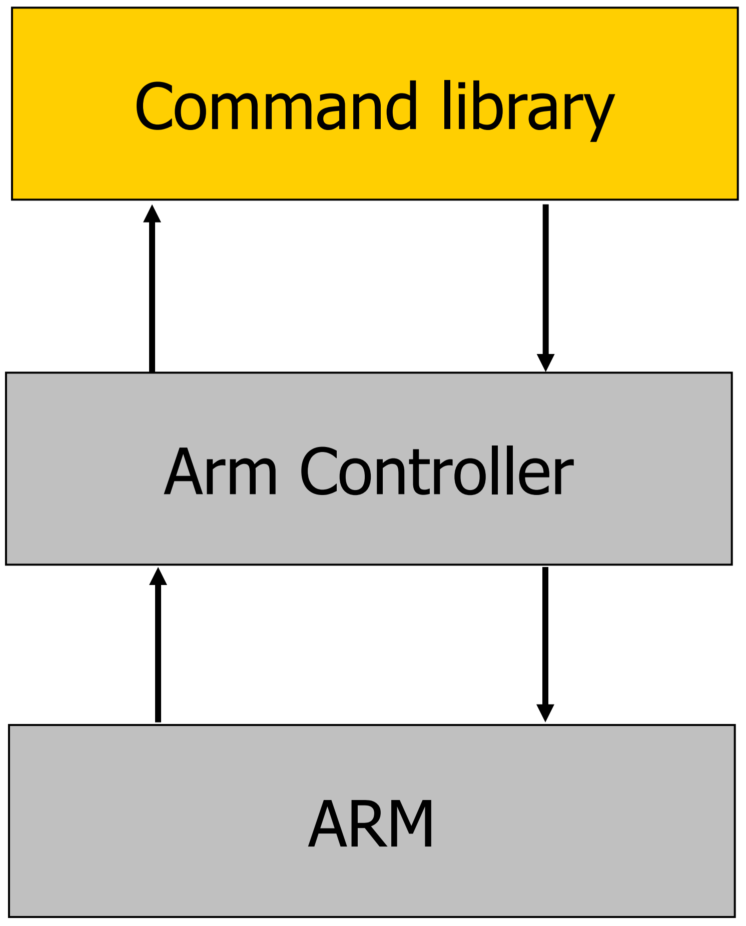
Angles values from given position:

CCW (Counter Clockwise) = positive;
CW (Clockwise) = negative.

Control system



Software architecture



Software interfaces

Current position reading

- In joint space:

bool read_arm_q (double q)*

- *q*: pointer to a 8-double array containing the arm position in joint degrees

- In Cartesian space:

bool read_arm_c (double p)*

- p: pointer to a 6-double array containing the end-effector position in mm and orientation in degrees, in Cartesian space

Software interfaces

Motion commands

- In joint space:

bool move_arm_q(double q)*

- *q*: pointer to a 8-double array containing the arm position in joint degrees

Software interfaces

Motion commands

- In Cartesian space:

bool move_arm_c7(double p, double elbow, double J0, double velocity)*

- *p*: pointer to a 6-double array containing the end-effector position in mm and orientation in degrees, in Cartesian space
- *Elbow*: elbow angle in degrees
- *J0*: final position of joint 0
- Velocity: ratio of maximum velocity

Kinematic inversion on 7 dof

Software interfaces

Motion commands

- In Cartesian space:

bool move_arm_c(double p, double elbow, double velocity)*

- *p*: pointer to a 6-double array containing the end-effector position in mm and orientation in degrees, in Cartesian space
- *Elbow*: elbow angle in degrees
- Velocity: ratio of maximum velocity

Kinematic inversion on 8 dof