Robotics 2

Trajectory Tracking Control

Prof. Alessandro De Luca

Dipartimento di Ingegneria Informatica
Automatica e Gestionale Antonio Ruberti

Sapienza
Università di Roma

Inverse dynamics control

given the robot dynamic model

$$\boxed{M(q)\ddot{q} + \underbrace{n(q,\dot{q})}_{c(q,\dot{q}) + g(q) + \text{friction model}} = u}$$

and a twice-differentiable desired trajectory for $t \in [0, T]$

$$q_d(t) \quad \rightarrow \quad \dot{q}_d(t), \ddot{q}_d(t)$$

applying the feedforward torque in nominal conditions

$$\boxed{u_d = M(q_d)\ddot{q}_d + n(q_d, \dot{q}_d)}$$

yields exact reproduction of the desired motion, provided that $q(0) = q_d(0), \dot{q}(0) = \dot{q}_d(0)$ (initial matched state)

In practice ...

a number of differences from the nominal condition

- initial state is "not matched" to the desired trajectory $q_d(t)$
- disturbances on the actuators, truncation errors on data, ...
- inaccurate knowledge of robot dynamic parameters (link masses, inertias, center of mass positions)
- unknown value of the carried payload
- presence of unmodeled dynamics (complex friction phenomena, transmission elasticity, ...)

Introducing feedback

$$\boxed{\hat{u}_d = \widehat{M}(q_d)\ddot{q}_d + \hat{n}(q_d, \dot{q}_d)}$$

with $\widehat{M}$, $\hat{n}$ estimates of terms (or coefficients) in the dynamic model

note: $\hat{u}_d$ can be computed off line [e.g., by $\widehat{NE}_\alpha(q_d, \dot{q}_d, \ddot{q}_d)$]

feedback is introduced to make the control scheme more robust

different possible implementations depending on amount of computational load share

- OFF LINE ($\Leftrightarrow$ open loop)
- ON LINE ($\Leftrightarrow$ closed loop)

two-step control design:

1. compensation (feedforward) or cancellation (feedback) of nonlinearities
2. synthesis of a linear control law stabilizing the trajectory error to zero

A series of trajectory controllers

1. inverse dynamics compensation (FFW) + PD

$$\boxed{u = \hat{u}_d + K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})}$$

2. inverse dynamics compensation (FFW) + variable PD

$$\boxed{u = \hat{u}_d + \widehat{M}(q_d)[K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})]}$$

typically, only local stabilization of trajectory error $e(t) = q_d(t) - q(t)$

3. feedback linearization (FBL) + [PD+FFW] = "COMPUTED TORQUE"

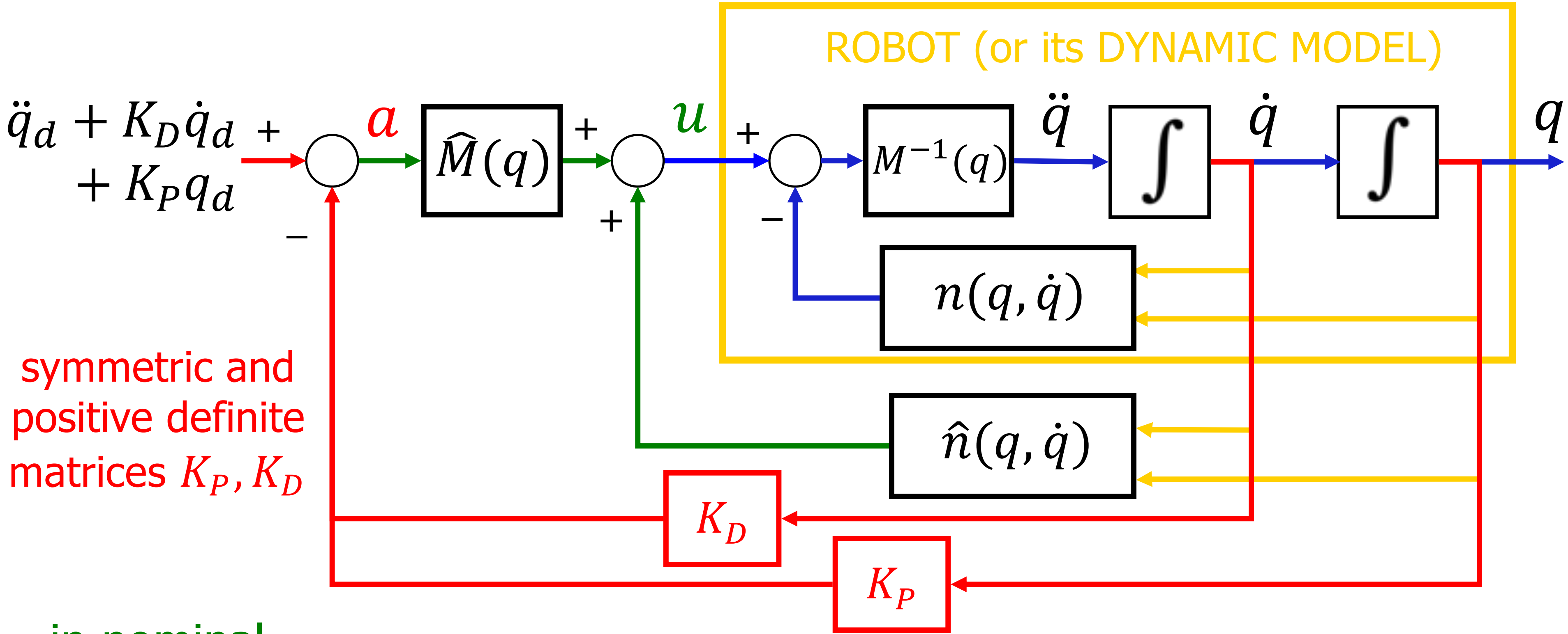
$$\boxed{u = \widehat{M}(q)[\ddot{q}_d + K_P(q_d - q) + K_D(\dot{q}_d - \dot{q})] + \hat{n}(q, \dot{q})}$$

4. feedback linearization (FBL) + [PID+FFW]

$$\boxed{u = \widehat{M}(q)\left[\ddot{q}_d + K_P(q_d - q) + K_D(\dot{q}_d - \dot{q}) + K_I \int (q_d - q)dt\right] + \hat{n}(q, \dot{q})}$$

more robust to uncertainties, but also more complex to implement in real time

Feedback linearization control



symmetric and positive definite matrices K_P, K_D

in nominal conditions $(\hat{M} = M, \hat{n} = n)$

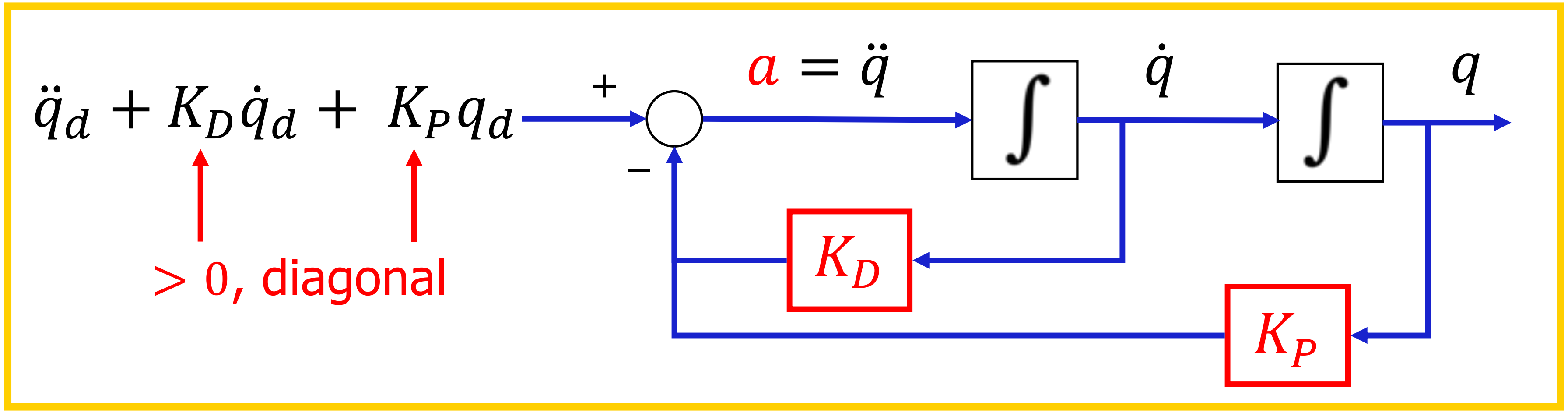
$$\boxed{\underbrace{M(q)\ddot{q} + n(q,\dot{q})}_{\text{nonlinear robot dynamics}} = u = \underbrace{M(q)a + n(q,\dot{q})}_{\text{nonlinear control law}}} \;\Rightarrow\; \boxed{\ddot{q} = a}$$

linear and decoupled system

global asymptotic stabilization

$$\boxed{a = \ddot{q}_d + K_D(\dot{q}_d - \dot{q}) + K_P(q_d - q\,)}$$

Interpretation in the linear domain



under feedback linearization control, the robot has a dynamic behavior that is invariant, linear and decoupled in its whole workspace ($\forall(q, \dot{q})$)

a unitary mass ($m = 1$) in the joint space !!

linearity

error transients $e_i = q_{di} - q_i \rightarrow 0$ exponentially, prescribed by K_{Pi}, K_{Di} choice

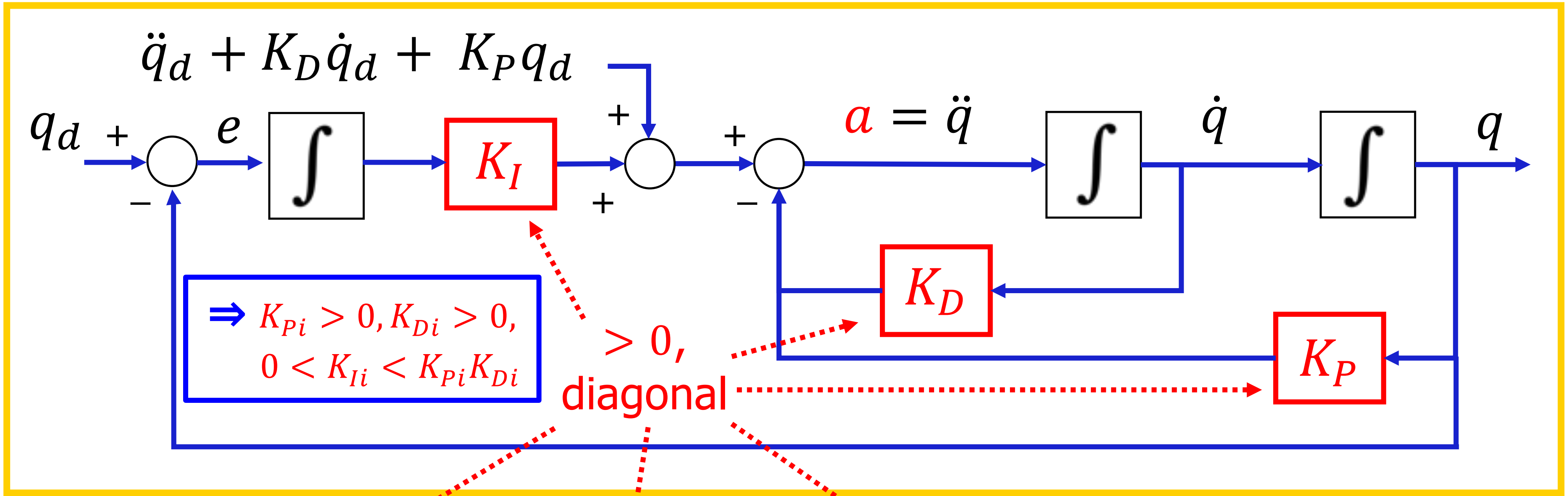
decoupling

each joint coordinate q_i evolves independently from the others, forced by a_i

$$\ddot{e} + K_D\dot{e} + K_P e = 0 \quad \Leftrightarrow \quad \ddot{e}_i + K_{Di}\dot{e}_i + K_{Pi}e_i = 0$$

Addition of an integral term: PID

whiteboard…



$$\ddot{q} = a = \ddot{q}_d + K_D(\dot{q}_d - \dot{q}) + K_P(q_d - q) + K_I \int (q_d - q)\, d\tau \qquad e = q_d - q$$

$\Rightarrow$ **(1)** $e_i = q_{di} - q_i \ (i = 1, \dots, N)$ $\Rightarrow$ **(2)** $\ddot{e}_i + K_{Di}\dot{e}_i + K_{Pi}e_i + K_{Pi}\int e_i d\tau = 0$

$\mathcal{L}[e_i(t)] \Rightarrow$ **(3)** $\left(s^2 + K_{Di}s + K_{Pi} + K_{Ii}\frac{1}{s}\right) e_i(s) = 0$

$s \times \Rightarrow$ **(4)** $(s^3 + K_{Di}s^2 + K_{Pi}s + K_{Ii})e_i(s) = 0$ $\Rightarrow$ **(5)**

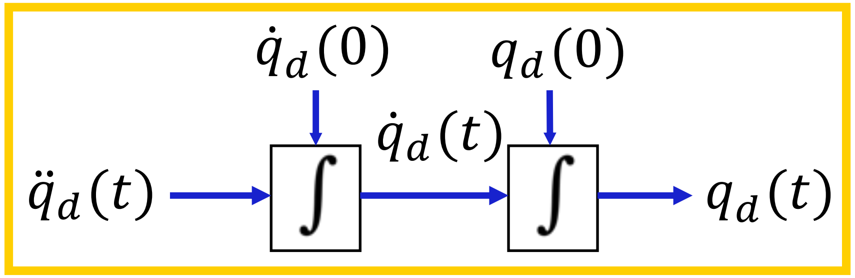
3	1	K_{Pi}
2	K_{Di}	K_{Ii}
1	$(K_{Di}K_{Pi} - K_{Ii})/K_{Di}$	
0	K_{Ii}	

$\Rightarrow$ **(6)** exponential stability conditions by Routh criterion

Remarks

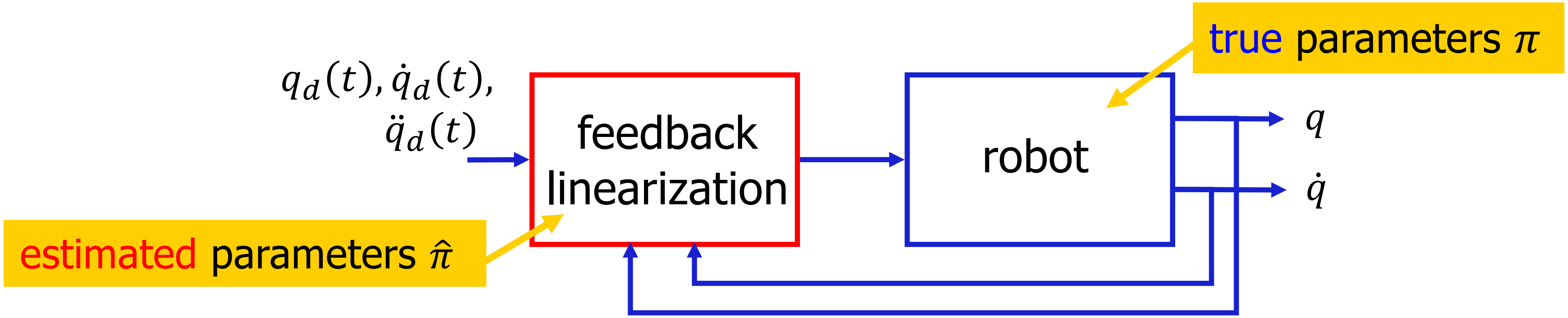
- desired joint trajectory can be generated from Cartesian data

$$\ddot{p}_d(t), \dot{p}_d(0), p_d(0)$$



$$q_d(0) = f^{-1}(p_d(0))$$
$$\dot{q}_d(0) = J^{-1}(q_d(0))\dot{p}_d(0)$$
$$\ddot{q}_d(t) = J^{-1}(q_d)[\ddot{p}_d(t) - \dot{J}(q_d)\dot{q}_d]$$

- real-time computation by Newton-Euler algo: $u_{FBL} = \widehat{NE}_\alpha(q, \dot{q}, a)$
- simulation of feedback linearization control

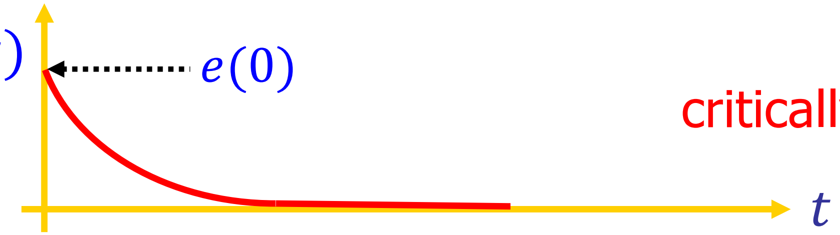


Hint: there is no use in simulating this control law in ideal case ($\hat{\pi} = \pi$); robot behavior will be identical to the linear and decoupled case of stabilized double integrators!!

Further comments

- choice of the diagonal elements of K_P, K_D (and K_I)
  - for shaping the error transients, with an eye to motor saturations...

$e(t) = q_d(t) - q(t)$



critically damped transient

- parametric identification
  - to be done in advance, using the property of linearity in the dynamic coefficients of the robot dynamic model
- choice of the sampling time of a digital implementation
  - compromise between computational time and tracking accuracy, typically $T_c = 0.5 \div 10$ ms
- exact linearization by (state) feedback is a general technique of nonlinear control theory
  - can be used for robots with elastic joints, wheeled mobile robots, ...
  - non-robotics applications: satellites, induction motors, helicopters, ...

Another example of feedback linearization design

- dynamic model of robots with elastic joints
  - q = link position
  - θ = motor position (after reduction gears)
  - B_m = diagonal matrix (> 0) of inertia of the (balanced) motors
  - K = diagonal matrix (> 0) of (finite) stiffness of the joints

$\left.\begin{matrix} q \\ \theta \end{matrix}\right\}$ $2N$ generalized coordinates (q, θ)

$4N$ state variables
$x = (q, \theta, \dot{q}, \dot{\theta})$

$$\begin{cases} M(q)\ddot{q} + c(q,\dot{q}) + g(q) + K(q-\theta) = 0 & (1) \\ B_m\ddot{\theta} + K(\theta - q) = u & (2) \end{cases}$$

- is there a control law that achieves exact linearization via feedback?

$$u = \alpha(q,\theta,\dot{q},\dot{\theta}) + \beta(q,\theta,\dot{q},\dot{\theta})\, a$$

$\boxed{\text{YES}}$ and it yields $\dfrac{d^4 q_i}{dt^4} = a_i, \quad i = 1,..,N$

linear and decoupled system: N chains of 4 integrators (to be stabilized by linear control design)

Hint: differentiate (1) *w.r.t. time until motor acceleration* $\ddot{\theta}$ *appears; substitute this from* (2)*; choose* u *so as to cancel all nonlinearities …*

Alternative global trajectory controller

$$u = M(q)\ddot{q}_d + S(q,\dot{q})\dot{q}_d + g(q) + F_V\dot{q}_d + K_P e + K_D\dot{e}$$

SPECIAL factorization such that $\dot{M} - 2S$ is skew-symmetric

symmetric and positive definite matrices

- global asymptotic stability of $(e,\dot{e}) = (0,0)$ (trajectory tracking)
- proven by Lyapunov+Barbalat+LaSalle
- does not produce a complete cancellation of nonlinearities
  - the $\dot{q}$ and $\ddot{q}$ that appear linearly in the model are evaluated on the desired trajectory
- does not induce a linear and decoupled behavior of the trajectory error $e(t) = q_d(t) - q(t)$ in the closed-loop system
- lends itself more easily to an adaptive version
- cannot be computed directly by the standard NE algorithm...

Analysis of asymptotic stability
of the trajectory error - 1

$$M(q)\ddot{q} + S(q,\dot{q})\dot{q} + g(q) + F_V\dot{q} = u$$ robot dynamics (including friction)

control law $$u = M(q)\ddot{q}_d + S(q,\dot{q})\dot{q}_d + g(q) + F_V\dot{q}_d + K_P e + K_D\dot{e}$$

- Lyapunov candidate and its time derivative

$$V = \frac{1}{2}\dot{e}^T M(q)\dot{e} + \frac{1}{2}e^T K_P e \geq 0 \;\Rightarrow\; \dot{V} = \frac{1}{2}\dot{e}^T \dot{M}(q)\dot{e} + \dot{e}^T \underbrace{M(q)\ddot{e}} + e^T K_P \dot{e}$$

- the closed-loop system equations yield

$$M(q)\ddot{e} = -S(q,\dot{q})\dot{e} - (K_D + F_V)\dot{e} - K_P e$$

- substituting and using the skew-symmetric property

$$\dot{V} = -\dot{e}^T (K_D + F_V)\dot{e} \leq 0 \qquad \dot{V} = 0 \;\Leftrightarrow\; \dot{e} = 0$$

- since the system is time-varying (due to $q_d(t)$), direct application of LaSalle theorem is NOT allowed ⇒ use Barbalat lemma…

$$q = q_d(t) - e, \dot{q} = \dot{q}_d(t) - \dot{e} \;\Rightarrow\; V = V(\underbrace{e, \dot{e}}_{\text{error state } x}, t) = V(x,t)$$

⇒ go to slide 10 in block 8

Analysis of asymptotic stability
of the trajectory error - 2

- since i) V is lower bounded and ii) $\dot{V} \leq 0$, we can check condition iii) in order to apply Barbalat lemma

$$\ddot{V} = -2\dot{e}^T(K_D + F_V)\ddot{e}$$ ... is this bounded?

- from i) + ii), V is bounded $\Rightarrow$ e and $\dot{e}$ are bounded
- assume that the desired trajectory has bounded velocity $\dot{q}_d$

$\Rightarrow$ $\dot{q}$ is bounded

- using the following two properties of dynamic model terms

$$0 < m \leq \|M^{-1}(q)\| \leq M < \infty \qquad \|S(q,\dot{q})\| \leq \alpha_S \|\dot{q}\|$$

then also $\ddot{e}$ will be bounded (in norm) since

$$\ddot{e} = -M^{-1}(q)[S(q,\dot{q})\dot{e} + K_P e + (K_D + F_V)\dot{e}]$$

$M^{-1}(q)$: bounded in norm by M

$S(q,\dot{q})$: bounded in norm by $\alpha_S\|\dot{q}\|$ $\longleftarrow$ bounded

$\dot{e}$: bounded; e: bounded; $\dot{e}$: bounded

$\Rightarrow \lim_{t\to\infty} \dot{V}(t) = 0$

Analysis of asymptotic stability
of the trajectory error – end of proof

- we can now conclude by proceeding as in LaSalle theorem

$$\dot{V} = 0 \quad \Leftrightarrow \quad \dot{e} = 0$$

- the closed-loop dynamics in this situation is

$$M(q)\ddot{e} = -K_P e$$

$$\Longrightarrow \quad \ddot{e} = 0 \quad \Leftrightarrow \quad e = 0 \quad \Rightarrow \quad (e, \dot{e}) = (0, 0)$$

is the largest invariant set in $\dot{V} = 0$

$\Rightarrow$ (global) asymptotic tracking will be achieved

Regulation as a special case

- what happens to the control laws designed for trajectory tracking when q_d is constant? are there simplifications?
- feedback linearization

$$u = M(q)[K_P(q_d - q) - K_D\dot{q}] + c(q,\dot{q}) + g(q)$$

  - no special simplifications
  - however, this is a solution to the regulation problem with exponential stability (and decoupled transients at each joint!)
- alternative global controller

$$u = K_P(q_d - q) - K_D\dot{q} + g(q)$$

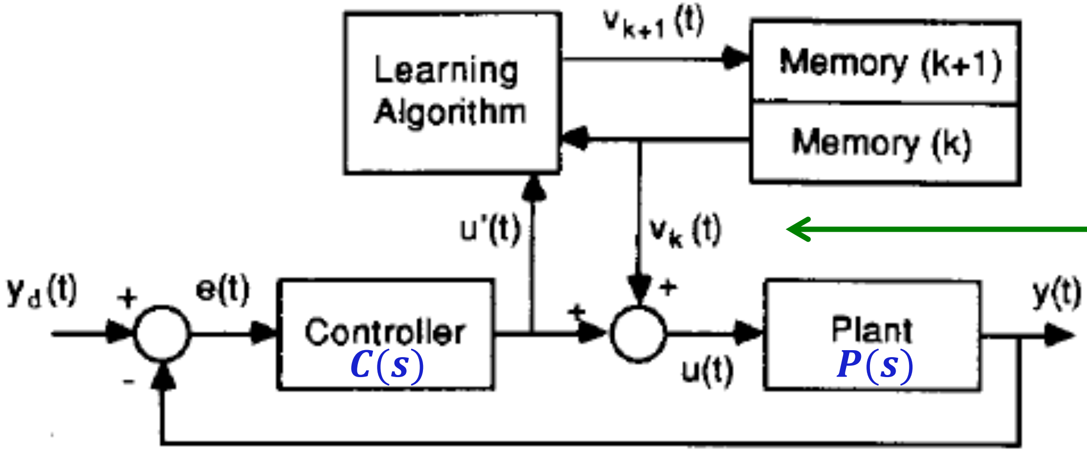
  - we recover the PD + gravity cancellation control law!!

Trajectory execution without a model

- is it possible to accurately reproduce a desired smooth joint-space reference trajectory with reduced or no information on the robot dynamic model?
- this is feasible in case of repetitive motion tasks over a finite interval of time
  - trials are performed iteratively, storing the trajectory error information of the current execution [k-th iteration] and processing it off line before the next trial [$(k+1)$-iteration] starts
  - the robot should be reinitialized in the same initial position at the beginning of each trial
  - the control law is made of a non-model based part (typically, a decentralized PD law) + a time-varying feedforward which is updated at every trial
- this scheme is called iterative trajectory learning

Scheme of iterative trajectory learning

- control design can be illustrated on a SISO linear system in the Laplace domain



"plug-in" signal: $v_1(t) \equiv 0$ at first ($k = 1$) iteration

$$W(s) = \frac{y(s)}{y_d(s)} = \frac{P(s)C(s)}{1 + P(s)C(s)}$$

closed-loop system without learning ($C(s)$ is, e.g., a PD control law)

$$u_k(s) = u'_k(s) + v_k(s) = C(s)e_k(s) + v_k(s)$$

control law at iteration k

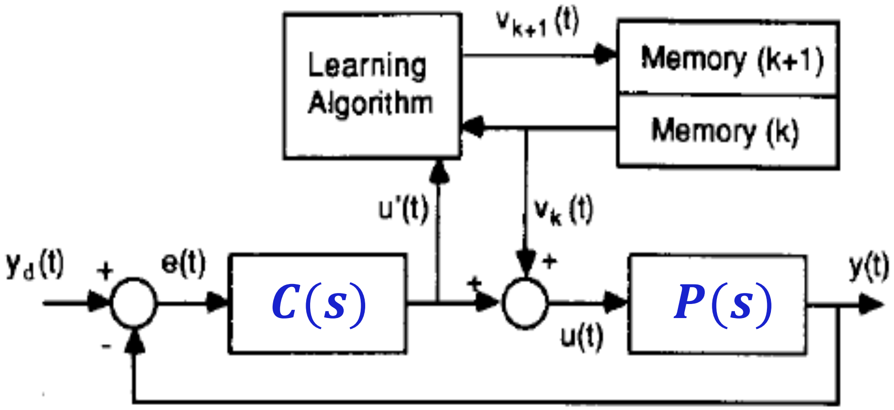
$$y_k(s) = W(s)y_d(s) + \frac{P(s)}{1 + P(s)C(s)} v_k(s)$$

system output at iteration k

Background math on feedback loops

whiteboard…

- algebraic manipulations on block diagram signals in the Laplace domain: $x(s) = \mathcal{L}[x(t)]$, $x = \{y_d, y, u', v, e\} \Rightarrow \{y_d, y_k, u'_k, v_k, e_k\}$, with transfer functions



$$\begin{aligned} y(s) &= P(s)u(s) = P(s)\big(v(s) + u'(s)\big) \\ &= P(s)v(s) + P(s)C(s)e(s) \\ &= P(s)v(s) + P(s)C(s)(y_d(s) - y(s)) \end{aligned}$$

$$\Rightarrow (1 + P(s)C(s))\, y(s) = \\ = P(s)v(s) + P(s)C(s)y_d(s)$$

$$\Rightarrow \; y(s) = \frac{P(s)C(s)}{1 + P(s)C(s)}\, y_d(s) + \frac{P(s)}{1 + P(s)C(s)}\, v(s) = W(s)y_d(s) + W_v(s)v(s)$$

- feedback control law at iteration k

$$u'_k(s) = C(s)\big(y_d(s) - y_k(s)\big) = C(s)y_d(s) - P(s)C(s)\big(v_k(s) + u'_k(s)\big)$$

$$\Rightarrow \; u'_k(s) = \frac{C(s)}{1 + P(s)C(s)} y_d(s) - \frac{P(s)C(s)}{1 + P(s)C(s)} v_k(s) = W_c(s)y_d(s) - W(s)v_k(s)$$

- error at iteration k

$$e_k(s) = y_d(s) - y_k(s) = y_d(s) - \big(W(s)y_d(s) + W_v(s)v_k(s)\big) = \big(1 - W(s)\big)y_d(s) - W_v(s)v_k(s)$$

$$W_e(s) = 1/(1 + P(s)C(s))$$

Learning update law

- the update of the feedforward term is designed as

$$\boxed{v_{k+1}(s) = \alpha(s)u'_k(s) + \beta(s)v_k(s)}$$

with α and β suitable filters (also non-causal, of the FIR type)

recursive expression of feedforward term
$$v_{k+1}(s) = \frac{\alpha(s)C(s)}{1+P(s)C(s)}y_d(s) + (\beta(s) - \alpha(s)W(s))v_k(s)$$

recursive expression of error $e = y_d - y$
$$e_{k+1}(s) = \frac{1-\beta(s)}{1+P(s)C(s)}y_d(s) + \big(\beta(s) - \alpha(s)W(s)\big)e_k(s)$$

- if a contraction condition can be enforced

$$\boxed{|\beta(s) - \alpha(s)W(s)| < 1}$$
(for all $s = j\omega$ frequencies such that ...)

then convergence is obtained for $k \to \infty$

$$v_\infty(s) = \frac{y_d(s)}{P(s)}\frac{\alpha(s)W(s)}{1-\beta(s)+\alpha(s)W(s)} \qquad e_\infty(s) = \frac{y_d(s)}{1+P(s)C(s)}\frac{1-\beta(s)}{1-\beta(s)+\alpha(s)W(s)}$$

Proof of recursive updates

whiteboard…

- recursive expression for the feedworward v_k

$$v_{k+1}(s) = \alpha(s)u'_k(s) + \beta(s)v_k(s) = \alpha(s)C(s)e_k(s) + \beta(s)v_k(s)$$
$$= \alpha(s)C(s)[W_e(s)y_d(s) - W_v(s)v_k(s)] + \beta(s)v_k(s)$$
$$= \frac{\alpha(s)C(s)}{1+P(s)C(s)}y_d(s) + (\beta(s) - \alpha(s)W(s))\, v_k(s)$$

- recursive expression for the error e_k

$$e_k(s) = y_d(s) - y_k(s) = y_d(s) - P(s)(v_k(s) + u'_k(s))$$
$$\Rightarrow \quad v_k(s) = \frac{1}{P(s)}\big(y_d(s) - e_k(s)\big) - u'_k(s)$$

$$y_{k+1}(s) = P(s)\big(v_{k+1}(s) + u'_{k+1}(s)\big) = P(s)\big(\alpha(s)u'_k(s) + \beta(s)v_k(s) + u'_{k+1}(s)\big)$$
$$= P(s)\left(\alpha(s)C(s)e_k(s) + \beta(s)\frac{1}{P(s)}\big(y_d(s) - e_k(s)\big) - \beta(s)C(s)e_k(s) + C(s)e_{k+1}(s)\right)$$

$$e_{k+1}(s) = y_d(s) - y_{k+1}(s)$$
$$= (1-\beta(s))\, y_d(s) - \big[\big(\alpha(s) - \beta(s)\big)P(s)C(s) - \beta(s)\big]e_k(s) - P(s)C(s)e_{k+1}(s)$$

$$\Rightarrow \quad e_{k+1}(s) = \frac{1-\beta(s)}{1+P(s)C(s)}y_d(s) + \big(\beta(s) - \alpha(s)W(s)\big)\, e_k(s)$$

Proof of convergence

whiteboard…

from recursive expressions

$$v_{k+1}(s) = \frac{\alpha(s)C(s)}{1+P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s))\, v_k(s)$$

$$e_{k+1}(s) = \frac{1-\beta(s)}{1+P(s)C(s)} y_d(s) + \big(\beta(s) - \alpha(s)W(s)\big)\, e_k(s)$$

compute variations from k to $k+1$ (repetitive term in trajectory y_d vanishes!)

$$\Delta v_{k+1}(s) = v_{k+1}(s) - v_k(s) = (\beta(s) - \alpha(s)W(s))\, \Delta v_k(s)$$

$$\Delta e_{k+1}(s) = e_{k+1}(s) - e_k(s) = \big(\beta(s) - \alpha(s)W(s)\big)\, \Delta e_k(s)$$

by contraction mapping condition $|\beta(s) - \alpha(s)W(s)| < 1 \Rightarrow \{v_k\} \to v_\infty,\ \{e_k\} \to e_\infty$

$$v_\infty(s) = \frac{\alpha(s)C(s)}{1+P(s)C(s)} y_d(s) + (\beta(s) - \alpha(s)W(s))\, v_\infty(s)$$

$$e_\infty(s) = \frac{1-\beta(s)}{1+P(s)C(s)} y_d(s) + \big(\beta(s) - \alpha(s)W(s)\big)\, e_\infty(s)$$

$$\Rightarrow \quad v_\infty(s) = \frac{y_d(s)}{P(s)} \frac{\alpha(s)W(s)}{1-\beta(s)+\alpha(s)W(s)} \qquad e_\infty(s) = \frac{y_d(s)}{1+P(s)C(s)} \frac{1-\beta(s)}{1-\beta(s)+\alpha(s)W(s)}$$

Comments on convergence

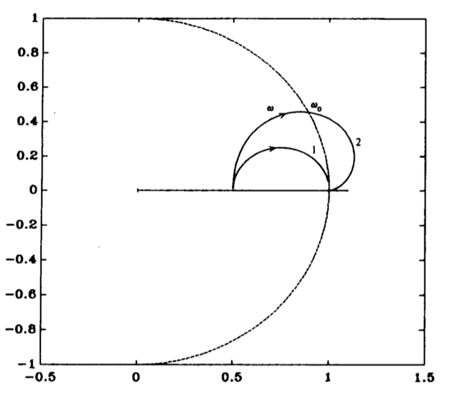
- if the choice $\beta = 1$ allows to satisfy the contraction condition, then convergence to zero tracking error is obtained

$$e_\infty(s) = 0$$

and the inverse dynamics command has been learned

$$v_\infty(s) = \frac{y_d(s)}{p(s)}$$

- in particular, for $\alpha(s) = 1/w(s)$ convergence would be in 1 iteration only!
- the use of filter $\beta(s) \neq 1$ allows to obtain convergence (but with residual tracking error) even in presence of unmodeled high-frequency dynamics
- the two filters can be designed from very poor information on system dynamics, using classic tools (e.g., Nyquist plots)



Application to robots

- for N-dof robots modeled as

$$[B_m + M(q)]\ddot{q} + [F_V + S(q,\dot{q})]\dot{q} + g(q) = u$$

we choose as (initial = pre-learning) control law

$$u = u' = K_P(q_d - q) + K_D(\dot{q}_d - \dot{q}) + \hat{g}(q)$$

and design the learning filters (at each joint) using the linear approximation

$$W_i(s) = \frac{q_i(s)}{q_{di}(s)} = \frac{K_{Di}s + K_{Pi}}{\hat{B}_m s^2 + \left(\hat{F}_{Vi} + K_{Di}\right)s + K_{Pi}} \qquad i = 1,\cdots,N$$

- initialization of feedforward uses the best estimates

$$v_1 = \left[\hat{B}_m + \widehat{M}(q_d)\right]\ddot{q}_d + \left[\hat{F}_V + \hat{S}(q_d,\dot{q}_d)\right]\dot{q}_d + \hat{g}(q_d)$$

or simply $v_1 = 0$ (in the worst case) at first trial $k = 1$

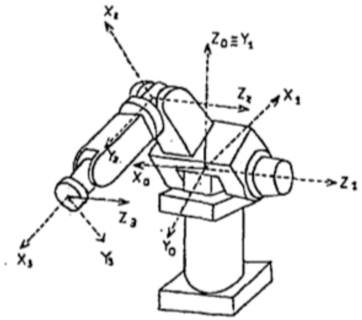
Experimental set-up

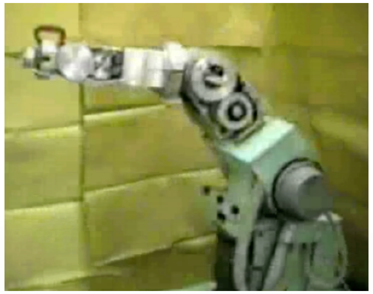
- joints 2 and 3 of 6R MIMO CRF robot prototype @DIS

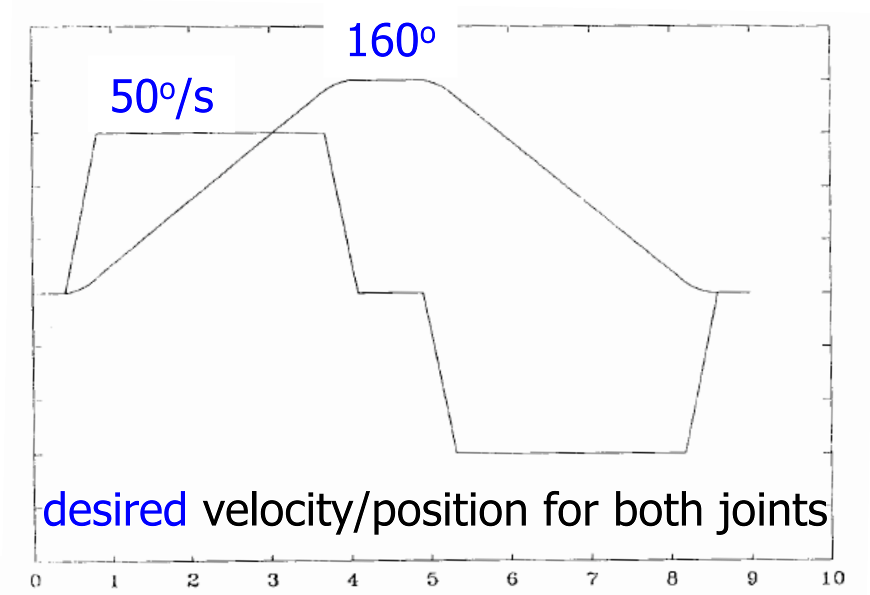
≈ 90% gravity balanced through springs

high level of dry friction

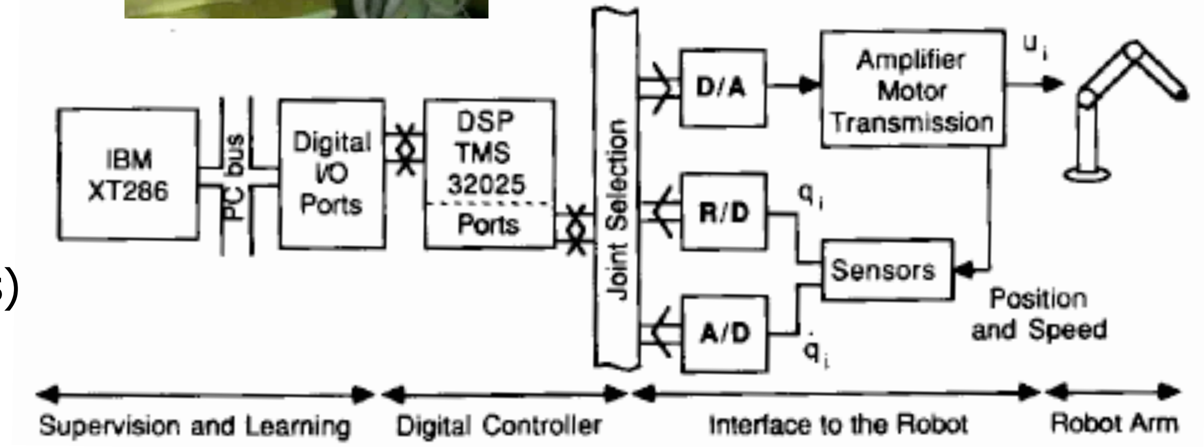
Harmonic Drives transmissions with ratio 160:1







DSP T_c = 400µs
D/A = 12 bit
R/D = 16 bit/2π
A/D = 11 bit/(rad/s)



DC motors with current amplifiers

resolvers and tachometers

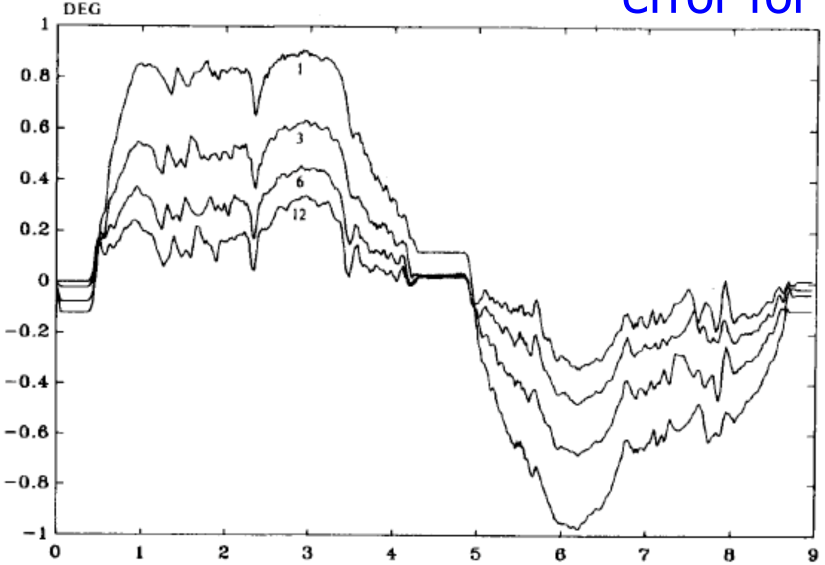
De Luca, Paesano, Ulivi: IEEE Trans Ind Elect, 1992

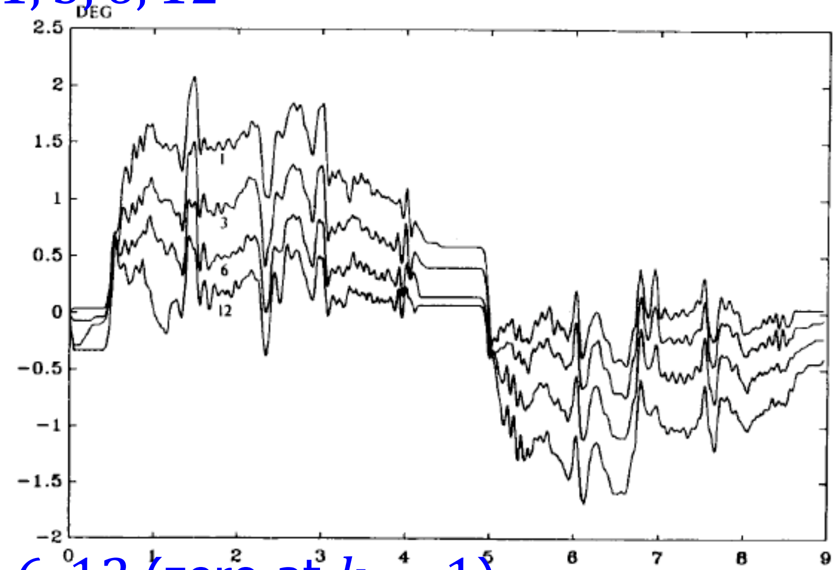
Experimental results

error for $k = 1, 3, 6, 12$

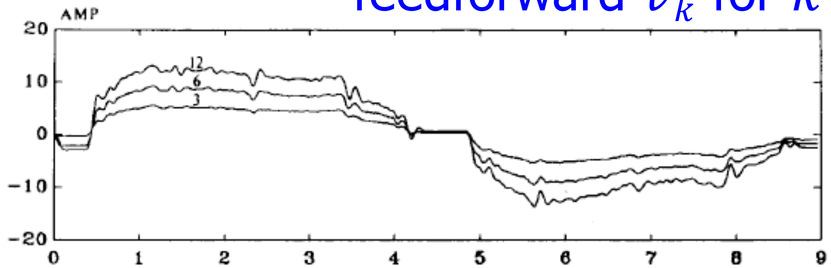
joint 2

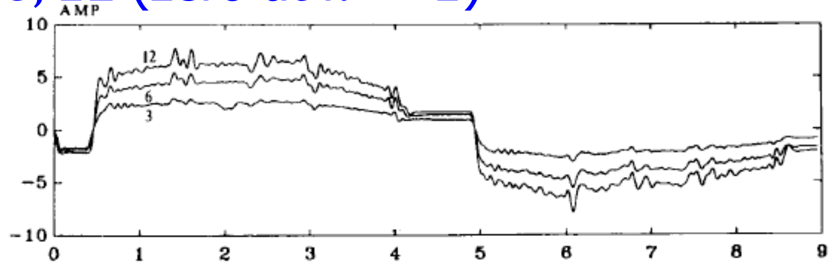
joint 3

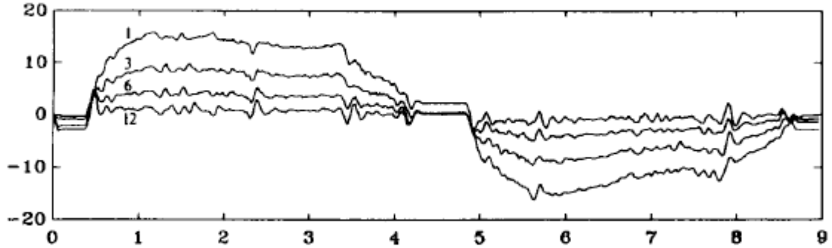


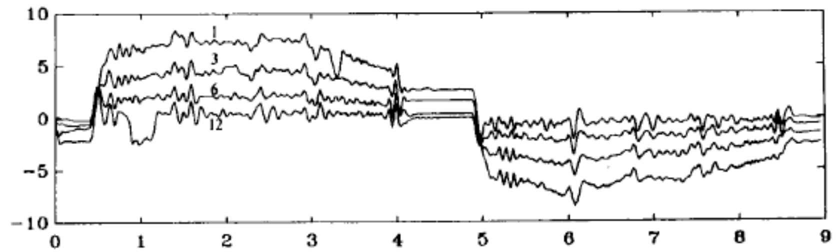


feedforward v_k for $k = 3, 6, 12$ (zero at $k = 1$)









feedback u'_k for $k = 1, 3, 6, 12$