These notes are from

F.L. Lewis, S, Jagannathan, and A. Yesildirek, *Neural Network Control of Robots and Nonlinear Systems*, Taylor and Francis, London, 1998.

3.4 FILTERED-ERROR APPROXIMATION-BASED CONTROL

Computed-torque control works very well when all the dynamical terms $M(q)$, $V_m(q,\dot{q})$, $F(\dot{q})$, $G(q)$ are known. In practice, robot manipulator parameters such as friction coefficients are unknown or change with time, and the masses picked up by the arm are often unknown. Therefore, in many applications simplified CT controllers that do not compute all nonlinear terms are used (e.g. classical joint control). These methods often rely on increasing the PD gains to obtain good performance. However, large control signals may result and stability proofs of such controllers are few and far between. A large class of controllers are the *approximation-based controllers* which are based on approximating unknown robot functions. In this class are included various sorts of adaptive and robust control techniques, as well as learning control and the neural network techniques discussed later in the book. These controllers provide good performance even in the presence of unknown dynamics and disturbances. Performance and stability can be mathematically proven and so relied upon in applications. Such advanced techniques also extend directly to more complicated control objectives such as force control for grinding, polishing, and so on where straight PD methods are inadequate.

In this section we discuss approximation-based control based on the *filtered tracking error*, deriving several adaptive and robust controllers which are collected for easy reference in Table 3.4.1.

Table 3.4.1: Filtered-Error Approximation-Based Control Algorithms

Robot Dynamics:

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = \tau,$$

Filtered Tracking Error $r(t)$:

$$\begin{aligned} e &= q_d - q \\ r &= \dot{e} + \Lambda e \end{aligned}$$

with Λ a positive definite design parameter matrix and $q_d(t)$ the desired trajectory.

Computed-Torque Control Variant:

$$\tau = K_v r + M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + F(\dot{q}) + G(q).$$

Adaptive Controller:

$$\begin{aligned} \tau &= W(x)\hat{\phi} + K_v r \\ \dot{\hat{\phi}} &= \Gamma W^T(x) r, \end{aligned}$$

where $W(x)$ is the regression matrix and Γ is a positive definite tuning rate matrix.
Saturation-Based Robust Controller:

$$\begin{aligned} \tau &= \hat{f} + K_v r - v(t) \\ v(t) &= \begin{cases} -r\frac{F(x)}{\|r\|}, & \|r\| \geq \varepsilon \\ -r\frac{F(x)}{\varepsilon}, & \|r\| < \varepsilon \end{cases} \end{aligned}$$

where ε is a small positive design parameter and $F(x)$ is a known scalar function that bounds the uncertainties $\tilde{f} = f - \hat{f}$ so that $\|\tilde{f}\| \leq F(x)$.
Variable Structure Robust Controller:

$$\begin{aligned} \tau &= \hat{f} + K_v r - v \\ v &= -(F(x) + \eta) sgn(r) \end{aligned}$$

where η is a small positive design parameter and $F(x)$ is a known function that bounds the uncertainties $\|f - \hat{f}\|$.
Learning Controller:

$$\begin{aligned} \tau_\ell &= \hat{f}_\ell + K_v r - v \\ v &= -(K_p e + K_s \|e\|^2 r) \\ \hat{f}_\ell &= \hat{f}_{\ell-1} + K_L r \end{aligned}$$

where ℓ is the iteration number, the gains K_v, K_p, K_s are positive diagonal design matrices, and K_L is a positive diagonal learning gain matrix.

3.4.1 A General Controller Design Framework Based on Approximation

The robot dynamics is given by

$$M(q)\ddot{q} + V_m(q,\dot{q})\dot{q} + F(\dot{q}) + G(q) + \tau_d = \tau. \tag{3.4.1}$$

In this section is derived a general tracking controller structure for robots that can be used to design adaptive, robust, and learning controllers, as well as neural network controllers.

The Tracking Problem. In robot control the objective is generally to make the robot manipulator follow a prescribed *desired trajectory*, often expressed in joint space as $q_d(t)$. Finding a control input $\tau(t)$ that causes this to occur is called the *tracking design problem*. A general framework for tracking control that includes many adaptive, robust, learning, and neural network techniques is the *approximation-based technique* now presented. Given the desired trajectory $q_d(t)$ define the *tracking error* $e(t)$ and *filtered tracking error* $r(t)$ by

$$e = q_d - q \tag{3.4.2}$$

$$r = \dot{e} + \Lambda e \tag{3.4.3}$$

with Λ a positive definite design parameter matrix. Common usage is to select Λ diagonal with large positive entries. Then, (3.4.3) is a stable system so that $e(t)$ is bounded as long as the controller guarantees that the filtered error $r(t)$ is bounded. In fact, it is easy to show that one has

$$\|e\| \leq \frac{\|r\|}{\sigma_{min}(\Lambda)}, \qquad \|\dot{e}\| \leq \|r\|, \tag{3.4.4}$$

where $\sigma_{min}(\Lambda)$ is the minimum singular value of Λ and the 2-norm is used.

In practical situations the desired trajectory is specified by the design engineer, so that it always satisfies the following boundedness assumption.

Bounded Trajectory. The desired trajectory is bounded so that

$$\left\| \begin{array}{c} q_d(t) \\ \dot{q}_d(t) \\ \ddot{q}_d(t) \end{array} \right\| \leq q_B, \tag{3.4.5}$$

with q_B a known scalar bound.

Differentiating (3.4.3) and invoking (3.4.1) it is seen that the robot dynamics are expressed in terms of the filtered error as

$$M\dot{r} = -V_m r + f(x) + \tau_d - \tau \tag{3.4.6}$$

where the *nonlinear robot function* is defined as

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + F(\dot{q}) + G(q). \tag{3.4.7}$$

Vector x contains all the time signals needed to compute $f(\cdot)$ and may be defined for instance as $x \equiv [e^T \ \dot{e}^T \ q_d^T \ \dot{q}_d^T \ \ddot{q}_d^T]^T$. It is important to note that $f(x)$ contains all

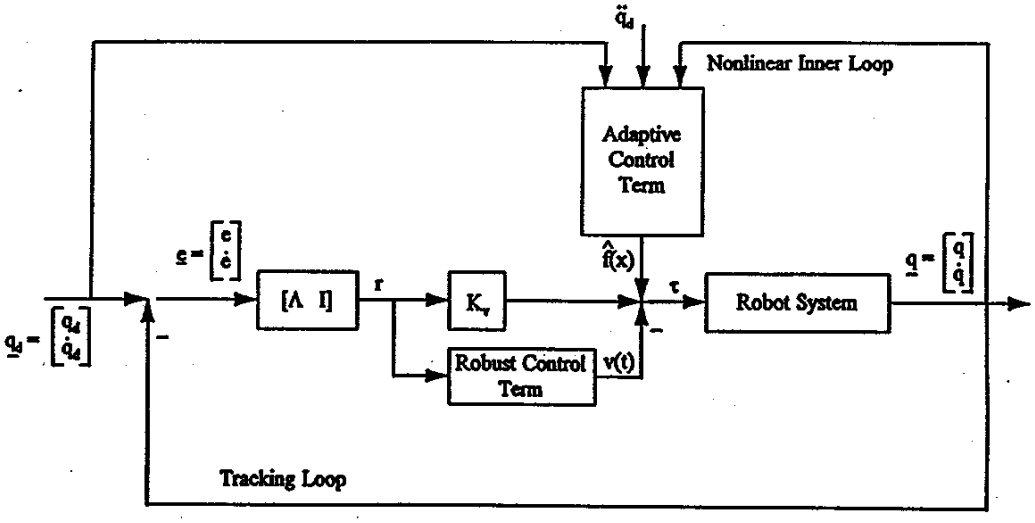


Figure 3.4.1: Filtered error approximation-based controller.

the potentially unknown robot arm parameters, except for the $V_m r$ term in (3.4.6), which cancels out in controller stability Lyapunov proofs.

The definition of the nonlinear Coriolis/centripetal matrix $V_m(q, \dot{q})$ in the robot dynamics (3.4.1) is not unique (Lewis, Abdallah, and Dawson 1993). In approximation-based control using the approach given here, it is necessary to select the *correct* version of $V_m(q, \dot{q})$; specifically, one must use the version of $V_m(q, \dot{q})$ so that the skew-symmetry property P3 holds. See Example 3.2.2 and Problems section.

Approximation-Based Controllers. A general sort of approximation-based controller is now derived by setting

$$\tau = \hat{f} + K_v r - v(t), \tag{3.4.8}$$

with $\hat{f}$ an *estimate* of $f(x)$, $K_v r = K_v \dot{e} + K_v \Lambda e$ an *outer PD tracking loop*, and $v(t)$ an auxiliary signal to provide robustness in the face of disturbances and modelling errors. The estimate $\hat{f}$ and robustifying signal $v(t)$ are defined differently for adaptive control, robust control, neural net control, fuzzy logic control, etc. The *multiloop control structure* implied by this scheme is shown in Fig. 3.4.1. In adaptive control techniques, most of the effort goes into selecting and updating the estimate $\hat{f}(x)$; in robust control, most of the effort goes into selecting the control term $v(t)$.

The Error Dynamics. Using nonlinear stability proofs based on Lyapunov or passivity techniques, it can be shown that tracking error stability can be guaranteed by selecting one of a variety of specific controllers. The controllers are derived and proofs of stability are given based on the all-important *closed-loop error dynamics*. The closed-loop error dynamics are found by substitution of (3.4.8) into (3.4.6) to be

$$M\dot{r} = -V_m r - K_v r + \tilde{f} + \tau_d + v(t), \tag{3.4.9}$$

where the *function approximation error* is given by

$$\tilde{f} = f - \hat{f}. \tag{3.4.10}$$

It is important to note that the tracking error dynamics is disturbed by the functional approximation error.

The *controller design problem* is to select the estimate $\hat{f}$ and the robust term $v(t)$ in control law (3.4.8) (see Fig. 3.4.1) so that this error dynamics is stable, for then the filtered tracking error $r(t)$ is bounded and (3.4.4) implies that the tracking error $e(t)$ is bounded. Consequently, the robot manipulator follows the prescribed trajectory $q_d(t)$. Several specific controllers that guarantee stable tracking are subsequently presented.

The error dynamics satisfy an important *passivity property* that is crucial in the design of robust neural network controllers. The proof uses a technique like those introduced in Chapter 2. State strict passivity (SSP) was defined in Chapter 2.

Theorem 3.4.1 (State Strict Passivity (SSP) of Error Dynamics) :
The error dynamics (3.4.9) are a state strict passive system from $\zeta_0(t) \equiv \tilde{f} + \tau_d + v(t)$ to $r(t)$.
Proof:
Take the nonnegative function

$$L = \frac{1}{2} r^T M(q) r$$

so that, using (3.4.9), one obtains

$$\dot{L} = r^T M \dot{r} + \frac{1}{2} r^T \dot{M} r = -r^T K_v r + \frac{1}{2} r^T (\dot{M} - 2V_m) r + r^T \zeta_0$$

whence the skew-symmetry property P3 yields the power form

$$\dot{L} = r^T \zeta_0 - r^T K_v r.$$

This is the power delivered to the system minus a quadratic term in $\|r\|$, verifying state strict passivity. □

3.4.2 Computed-Torque Control Variant

Several approximation-based controllers of the form (3.4.8) are now presented. A variant of computed-torque control can be used if the nonlinear function $f(x)$ is known. Then, one may select $\hat{f}(x) = f(x)$ so that the control input is

$$\tau = K_v r + M(q)(\ddot{q}_d + \Lambda \dot{e}) + V_m(q, \dot{q})(\dot{q}_d + \Lambda e) + F(\dot{q}) + G(q). \tag{3.4.11}$$

This may be compared to (3.3.5).

3.4.3 Adaptive Control

Adaptive control has proven successful in dealing with modeling uncertainties in general linear and nonlinear systems by on-line tuning of parameters. Variants of adaptive control include the model-reference approach (Landau 1976), hyperstability techniques (Landau 1976), self-tuning regulators (Åström and Wittenmark

1989), gradient-based techniques, and so on. See Narendra and Annaswamy (1989), Ioannou and Sun (1996), Goodwin and Sin (1984), Sastry and Bodson (1989) and other references too prolific to mention.

Included in the class of controllers (3.4.8) are many sorts of adaptive controllers for robot manipultors (Craig 1985; Lewis, Abdallah, Dawson 1993; Slotine and Li 1991; Spong and Vidyasagar 1989). Conventional adaptive controller applications in robotics rely on the following *linear-in-the-parameters (LIP) assumption* (Craig 1985) (see however the work of Colbaugh *et al.* (1994, 1995) where LIP is not needed).

LIP The nonlinear robot function (3.4.7) is *linear in the unknown parameters* (e.g. masses and friction coefficients) so that one can write

$$f(x) = M(q)(\ddot{q}_d + \Lambda\dot{e}) + V_m(q,\dot{q})(\dot{q}_d + \Lambda e) + F(\dot{q}) + G(q) = W(x)\phi \quad (3.4.12)$$

where $W(x)$ is a matrix of known robot functions and ϕ is a vector of unknown parameters (e.g. masses and friction coefficients). The *regression matrix* $W(\cdot)$ can be computed for any specified robot arm.

One adaptive controller is given by Slotine (1988),

$$\tau = W(x)\hat{\phi} + K_v r \quad (3.4.13)$$

$$\dot{\hat{\phi}} = \Gamma W^T(x) r \quad (3.4.14)$$

where Γ is a tuning parameter matrix, generally selected diagonal with positive elements. The adaptive controller manufactures an estimate $\hat{\phi}$ for the unknown parameter vector ϕ by *dynamic on-line tuning* using Equation (3.4.14), thus the controller has its own dynamics. The estimate of the nonlinear function is given by

$$\hat{f}(x) = W(x)\hat{\phi}. \quad (3.4.15)$$

The adaptive controller has the structure shown in Fig. 3.4.1. It has a multiloop structure with an outer PD tracking loop and an inner nonlinear adaptive loop whose function is to estimate the nonlinear function required for feedback linearization of the robot arm. Its performance is described in the following theorem and illustrated in the next example. Uniform ultimate bounded (UUB) stability was defined in Chapter 2.

Theorem 3.4.2 (Adaptive Controller) :

Suppose the disturbance $\tau_d(t)$ in (3.4.1) is zero and the desired trajectory $q_d(t)$ is bounded according to (3.4.5). Assume the linear-in-the-parameters asumption (3.4.12) holds and the unknown parameter vector ϕ is constant. Then, using the control (3.4.13) with adaptive parameter tuning given by (3.4.14), the tracking error $r(t)$ goes to zero and the parameter estimates $\hat{\phi}(t)$ are UUB.

Proof:

Due to the LIP assumption one has

$$\tilde{f} = f - \hat{f} = W(x)\phi - W(x)\hat{\phi} = W(x)\tilde{\phi},$$

with $\tilde{\phi} = \phi - \hat{\phi}$ the parameter estimation error. Therefore, using the proposed control yields the closed-loop error dynamics

$$M\dot{r} = -V_m r - K_v r + W(x)\tilde{\phi}.$$

Select the Lyapunov function candidate

$$L = \frac{1}{2} r^T M(q) r + \frac{1}{2} \tilde{\phi}^T \Gamma^{-1} \tilde{\phi},$$

with Γ a symmetric positive definite weighting matrix. Differentiate to discover

$$\dot{L} = \frac{1}{2} r^T \dot{M} r + r^T M \dot{r} + \tilde{\phi}^T \Gamma^{-1} \dot{\tilde{\phi}},$$

whence substitution from the error dynamics yields

$$\begin{aligned} \dot{L} &= \frac{1}{2} r^T \dot{M} r - r^T V_m r - r^T K_v r + r^T W(x) \tilde{\phi} + \tilde{\phi}^T \Gamma^{-1} \dot{\tilde{\phi}} \\ &= \frac{1}{2} r^T (\dot{M} - 2V_m) r - r^T K_v r + \tilde{\phi}^T (\Gamma^{-1} \dot{\tilde{\phi}} + W^T(x) r). \end{aligned}$$

Now use the skew-symmetry property P3 and select

$$\dot{\tilde{\phi}} = -\Gamma W^T(x) r$$

to yield

$$\dot{L} = -r^T K_v r.$$

In view of the definition $\tilde{\phi} = \phi - \hat{\phi}$, and the assumption that ϕ is constant, the selection for $\dot{\tilde{\phi}}$ yields the tuning law (3.4.14).

Since L is positive definite in the overall state $[r^T \ \tilde{\phi}^T]^T$ and $\dot{L}$ is negative semidefinite, both r and $\tilde{\phi}$ are bounded according to Lyapunov's theorem. Boundedness of the parameter estimate $\hat{\phi}$ follow from the fact that ϕ is bounded.

To show that $r(t)$ goes to zero, one may use Barbalat's Lemma to show that $\dot{L}$ goes to zero with t. To accomplish this, differentiate to obtain

$$\ddot{L} = -2 r^T K_v \dot{r},$$

and substitute the error dynamics to obtain

$$\ddot{L} = -2 r^T K_v M^{-1} [-V_m r - K_v r + W(x) \tilde{\phi}].$$

The right-hand side is bounded due to the bounding assumptions P1-P6 and the demonstrated boundedness of r and $\hat{\phi}$; for one has

$$M^{-1} \leq \frac{1}{\mu_1} I, \qquad \|V_m\| \leq v_B \|\dot{q}\|;$$

but $\dot{q}$ is bounded since r is bounded and the desired trajectory $q_d(t)$ is bounded. Therefore, $\ddot{L}$ is bounded, implying that $\dot{L}$ is uniformly continuous, and by the Barbalat's Lemma Lyapunov extension (Chapter 1), $\dot{L}$ goes to zero with t. Therefore, $r(t)$ vanishes as t becomes large. □

It has only been shown that the parameter error $\tilde{\phi}$ is bounded, not small. It can be shown that under an additional *persistence of excitation* condition (Chapter 2), the parameter error goes to zero so that the parameter estimates $\hat{\phi}$ approach the actual values ϕ (Slotine 1985, 1988). However, for practical controls purposes the result of the theorem is good enough, since the tracking error $r(t)$ is small so that one has tracking of the prescribed trajectory. Note also that the UUB of the parameter estimate $\hat{\phi}$ guarantees the boundedness of the control input $\tau(t)$ in (3.4.13).

This proof is typical of adaptive control and recurs in various forms throughout the book— The parameter to be tuned is weighted in the Lyapunov function candidate L. Then, the derivative of the parameter apppears in $\dot{L}$, yielding the tuning law.

Example 3.4.1 (Performance of Adaptive Controller) :

This example illustrates the sort of performance to be expected from the adaptive controller. It is also shown that if the regression matrix is not exactly known, the adaptive controller does not perform well. In this example, it is desired for the 2-link robot arm in Example 3.2.1 to follow a prescribed trajectory.

To implement the adaptive controller just described, it is necessary to determine the regression matrix $W(x)$. In computing $W(x)$ one must begin with the version of $V_m(q,\dot{q})$ that satisfies the skew-symmetry property P3 (see Example 3.2.2). The unknown parameter vector is selected to contain the masses, so that $\phi = [m_1 \;\; m_2]^T$, where m_2 includes the payload mass. The joint tracking errors are $e_1 = q_{d_1} - q_1, e_2 = q_{d_2} - q_2$. Referring to the dynamics derived in Example 3.2.1, one sees that the elements of the 2×2 matrix $W(x)$ are given (see Problems section) by

$$\begin{aligned}
W_{11} &= a_1^2(\ddot{q}_{d_1} + \lambda_1\dot{e}_1) + a_1 g\cos q_1 \\
W_{12} &= (a_2^2 + 2a_1a_2\cos q_2 + a_1^2)(\ddot{q}_{d_1} + \lambda_1\dot{e}_1) + (a_2^2 + a_1a_2\cos q_2)(\ddot{q}_{d_2} + \lambda_2\dot{e}_2) \\
&\quad - a_1a_2\dot{q}_2(\dot{q}_{d_1}\sin q_2 + \lambda_1 e_1) - a_1a_2(\dot{q}_1 + \dot{q}_2)(\dot{q}_{d_2}\sin q_2 + \lambda_2 e_2) \\
&\quad + a_2 g\cos(q_1 + q_2) + a_1 g\cos q_1 \\
W_{21} &= 0 \\
W_{22} &= (a_2^2 + a_1a_2\cos q_2)(\ddot{q}_{d_1} + \lambda_1\dot{e}_1) + a_2^2(\ddot{q}_{d_2} + \lambda_2\dot{e}_2) \\
&\quad + a_1a_2(\dot{q}_{d_1} + \lambda_1 e_1)\sin q_2 + a_2 g\cos(q_1 + q_2).
\end{aligned}$$

A function M file was written in MATLAB (1994) to simulate the adaptive controller— this procedure is very direct. The adaptive controller is an easy modification of the function routine given in Example 3.3.1. The PD CT controller equations in Fig. 3.3.2 are simply replaced by the adaptive controller equations (3.4.13)-(3.4.14). The lines of code implementing the adaptive controller are given in Fig. 3.4.2.

The arm parameters were taken as $a_1 = a_2 = 1m, m_1 = 0.8kg, m_2 = 2.3kg$. The desired trajectory was selected as $q_{d_1}(t) = \sin t$, $q_{d_2}(t) = \cos t$. The controller parameters were selected as $K_v = diag\{20, 20\}, \Gamma = diag\{10, 10\}, \Lambda = diag\{\lambda_1, \lambda_2\} = diag\{5, 5\}$. The dialog required to run the adaptive controller is the same as in Example 3.3.1. Remember to modify the output function *robout.m* to reflect the new desired trajectory parameters!

a. Adaptive Control. The response with the adaptive controller is given in Fig. 3.4.3, which is excellent even though the masses m_1, m_2 are unknown by the controller. It is seen that, after an initial error, the actual joint angles closely match the desired joint angles. In adaptive control, the controller dynamics allow for learning of the unknown parameters, so that the performance improves over time. Note that the parameter (mass) estimates converge to the correct constant values.

b. Adaptive Control with Unmodelled Dynamics. To show how important it is to model all the dynamics in the regression matrix, the simulation was now repeated using the incorrect value

$$W_{11} = a_1^2(\ddot{q}_{d_1} + \lambda_1\dot{e}_1);$$

For Figure 3.4.2 "Adaptive Controller" see the homework solutions.

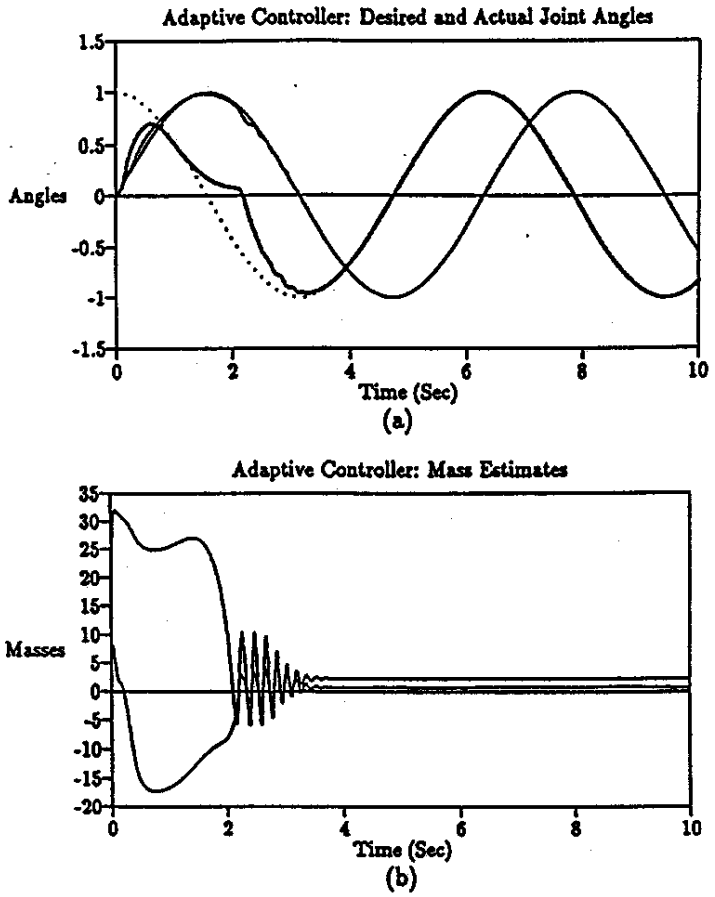


Figure 3.4.3: Response using adaptive controller. (a) Actual and desired joint angles. (b) Mass estimates.