PART 3

A Summary of Control Design for Robotic Manipulators

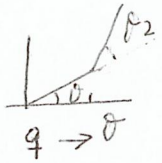
"See Also chapters 9 and 10 of your text-book"

B. Shafai

Linear Control of Manipulators

Recall Dynamical Equation of Manipulator

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau$$



where for two links manipulator we have

Inertia or Mass Matrix → $M(q)$, $q \to \theta$:

$$M(q) = \begin{bmatrix} (m_1+m_2)l_1^2 + m_2 l_2^2 + 2m_2 l_1 l_2 \cos\theta_2 & m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2 \\ m_2 l_2^2 + m_2 l_1 l_2 \cos\theta_2 & m_2 l_2^2 \end{bmatrix}$$

Terms with joint velocities → $V(q,\dot{q})$, θ $\dot{\theta}$:

$$V(q,\dot{q}) = \begin{bmatrix} -m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_2^2 - 2m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1\dot{\theta}_2 \\ m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1^2 \end{bmatrix}$$

($-m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_2^2$: related to Coriolis; $m_2 l_1 l_2 \sin\theta_2 \dot{\theta}_1^2$: related to Centrifugal)

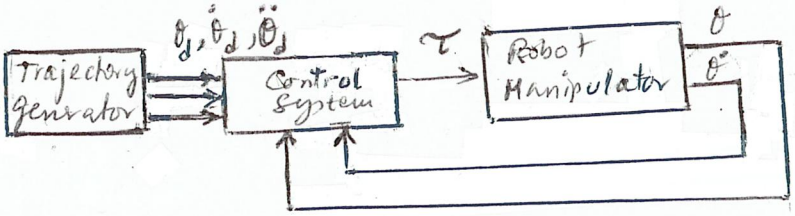
Terms with Gravity → $G(q)$, θ:

$$G(q) = \begin{bmatrix} (m_1+m_2) g l_1 \cos\theta_1 + m_2 g l_2 \cos(\theta_1+\theta_2) \\ m_2 g l_2 \cos(\theta_1+\theta_2) \end{bmatrix}$$

Given θ_d, $\dot{\theta}_d$, and $\ddot{\theta}_d$ by the trajectory generator, one can compute the torques that realize the desired trajectory, i.e.

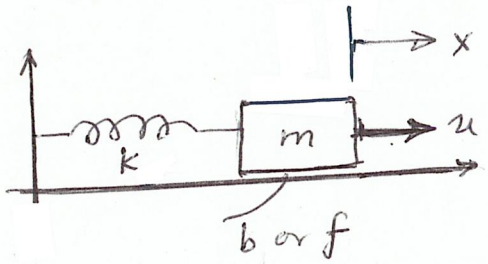
$$\tau = M(\theta_d)\ddot{\theta}_d + V(\theta_d,\dot{\theta}_d) + G(\theta_d) \qquad \text{"open loop"}$$

However, for closed-loop control system to improve the performance, τ should be a function of $\theta, \dot{\theta}$ such that $e = \theta_d - \theta$ is minimized or $\lim_{t\to\infty} e(t) \to 0$.



"closed-loop"
τ is the control

A simplified dynamical equation that resembles the above manipulator dynamics is the following spring-mass system with friction



$$m\ddot{x} + f\dot{x} + kx = u$$

The system without forcing function u is in motion with certain initial conditions $x(0)$, $\dot{x}(0)$.

The characteristic equation of the system is

$$ms^2 + fs + k = 0$$

$$s_{1,2} = \frac{-f \pm \sqrt{f^2 - 4mk}}{2m}$$

"poles of the system"

Three cases:

1. Real and Unequal Roots: $f^2 > 4m$ — Overdamped
   - friction dominates → sluggish
2. Complex Roots: $f^2 < 4mk$ — Underdamped
   - stiffness dominates → oscilatory
3. Real and Equal Roots: $f^2 = 4mk$ — Critically damped
   - friction and stiffness are balanced

$$s^2 + 2\xi\omega_n s + \omega_n^2 = 0$$

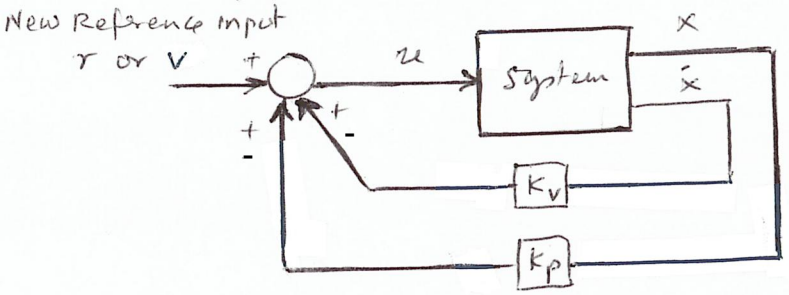
One can also analyze the response with $u(t) \neq 0$

Control of second Order systems

$$m\ddot{x} + f\dot{x} + Kx = u$$

▷ Control law for position regulation

$$u = -k_p x - k_v \dot{x} + v$$



$$m\ddot{x} + f\dot{x} + Kx = -k_p x - k_v \dot{x} + v$$

$$m\ddot{x} + (f + k_v)\dot{x} + (K + k_p)x = v$$

$$m s^2 + \underbrace{(f + k_v)}_{f'} s + \underbrace{(K + k_p)}_{k'} = 0 \quad \cdots \quad s^2 + \underbrace{\frac{f + k_v}{m}}\, s + \underbrace{\frac{K + k_p}{m}} = 0$$

$$\equiv \quad s^2 + 2\zeta\omega_n s + \omega_n^2$$

Example: Let $m = 1$, $f = 1$, $K = 1$. Then if we wish k' to be 16 then for critical damping we require that $f' = 2\sqrt{mk'} = 8$. So we need $k_p = 15$ and $k_v = 7$.

Compare the above with state space approach.

▷ Control law Partitioning "Method of Computed" "Flexible"

Force or Torque

Two steps :

Model-based Portion

$$m\ddot{x} + f\dot{x} + kx = u \qquad m\ddot{x} = \underbrace{u - kx - f\dot{x}}_{mu'}$$

$$u = \alpha u' + \beta$$

$$m\ddot{x} + f\dot{x} + kx = \alpha u' + \beta$$

Servo Portion

$$\alpha = m$$

$$\beta = f\dot{x} + kx$$

$$\Rightarrow \quad \ddot{x} = u'$$

→ One can use classical PID based on Transferfunction

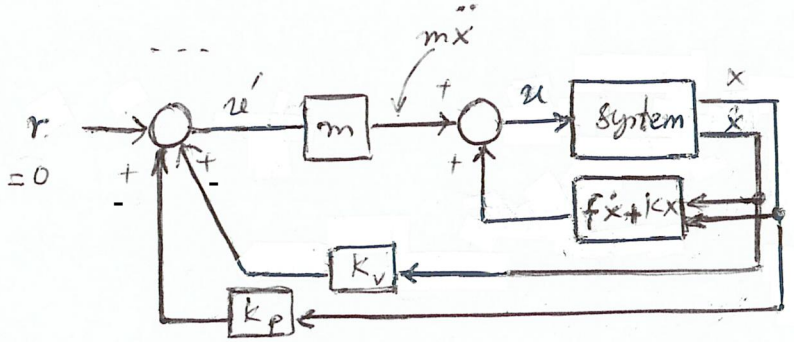
→ Or using state space techniques e.g. State Feedback

$$u' = -k_v\dot{x} - k_p x + r$$

← let the new reference be $r = 0$

$$\ddot{x} + k_v\dot{x} + k_p x = 0$$

$$s^2 + k_v s + k_p \equiv s^2 + 2\zeta\omega_n s + \omega_n^2$$



▷ Trajectory-Following control

Using the set-up of control law partitioning i.e. Method of computed Force / Torque, we can generalize position control at a desired location to desired trajectory following control.

We want u such that

$$m\ddot{x} + f\dot{x} + kx = u$$

follows a given trajectory $x_d(t)$.

We define the servo error between the desired and actual trajectory as $e = x_d - x$. Then from the previous page

$$\ddot{x} = u'$$

and a servo control law that achieves trajectory following is given by

$$u' = \ddot{x}_d + k_v(\dot{x}_d - \dot{x}) + k_p(x_d - x)$$

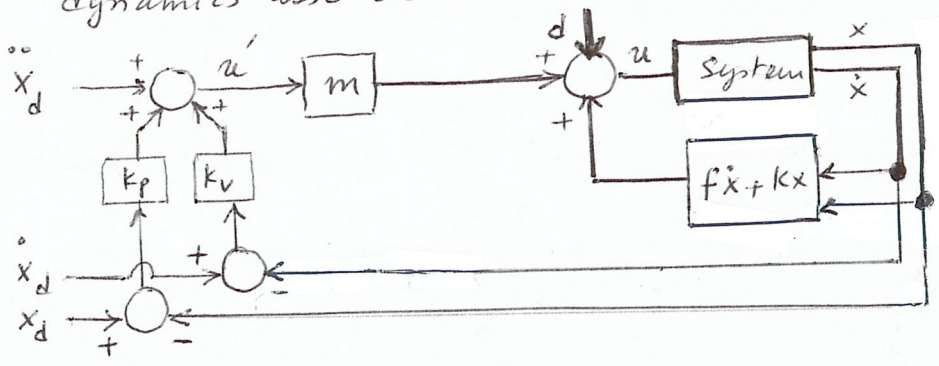
Thus,

$$\ddot{x} = \ddot{x}_d + k_v\dot{e} + k_p e$$

or

$$\ddot{e} + k_v\dot{e} + k_p e = 0$$

Now, one can use k_v and k_p to make the second-order dynamics associated with error achieve desired response.



Steady state analysis → $\ddot{e} + k_v \dot{e} + k_p e = \dot{d}$ — assume const. (*)

system at rest $\ddot{e} = 0, \dot{e} = 0$

$k_p e = d \;\; : \;\; e = d/k_p$.

$$\begin{cases} m\ddot{x} + f\dot{x} + kx + d = u \\ u = \alpha u' + \beta \\ m\ddot{x} + f\dot{x} + kx + d = \alpha u' + \beta \\ m = \alpha, \;\; \beta = f\dot{x} + kx \Rightarrow \ddot{x} + d = u' \\ u' = \ddot{x}_d + k_v \dot{e} + k_p e \Rightarrow \ddot{e} + k_v \dot{e} + k_p e = d \end{cases}$$

▷ Disturbance Rejection

Suppose in the previous figure a **bounded disturbance** is entered as shown by $d(t)$ at the summing junction

A simple steady-state error analysis reveals that integral action should be employed to achieve

$\lim_{t \to \infty} e(t) = 0$.

Thus, u' should be modified to

$$u' = \ddot{x}_d + k_v \underbrace{(\dot{x}_d - \dot{x})}_{\dot{e}} + k_p \underbrace{(x_d - x)}_{e} + k_I \int \underbrace{(x_d - x)}_{e} dt$$

which results in the error equation

$$\dddot{e} + k_v \ddot{e} + k_p \dot{e} + k_I e = \dot{d} \qquad (*)$$

Taking the derivative of (*) we get

$$\dddot{e} + k_v \ddot{e} + k_p \dot{e} + k_I e = \dot{d}$$ — let d = constant Then

By proper choice of k_p, k_v and k_I we make the third order error dynamic stable desired and we have in the steady state $k_I e = 0$

or $e = 0$.

The above is a PID controller design directly employed from differential equation.

See also additional examples from your text book.

✓

Nonlinear Control of Manipulators

The model of manipulators is represented by vector matrix differential equation and it is nonlinear MIMO system, as we discussed before i.e.

$$M(q)\ddot{q} + V(q,\dot{q}) + G(q) = \tau$$

Generally, there are two approaches to deal with nonlinear systems in order to apply control design. One method is the linearization around the operating point. The other method is known as feedback linearization, which is more desirable in variety of situations including robotic manipulators.

Let us first consider the scalar (SISO) nonlinear system for example

$$m\ddot{x} + b\dot{x} + qx^3 = u \qquad \circledast$$

(or f)

Based on the procedure outlined in the linear control technique "Control law Partitioning", The model-based portion of the control is

$$u = \alpha u' + \beta$$

where $\alpha = m$

$\beta = b\dot{x} + qx^3$

and the servo portion is reduced to control of

$$\ddot{x} = u'$$

when $u = \alpha u' + \beta$ is substituted in $\circledast$.

Consequently, the stabilization can be performed by applying the classical transfer function based approach using PID controller or state-space technique of state feedback control law.
Applying the stabilization directly through the differential equation, we can write

$$u' = -K_v \dot{x} - K_p x + r$$

← let the new reference $r = 0$

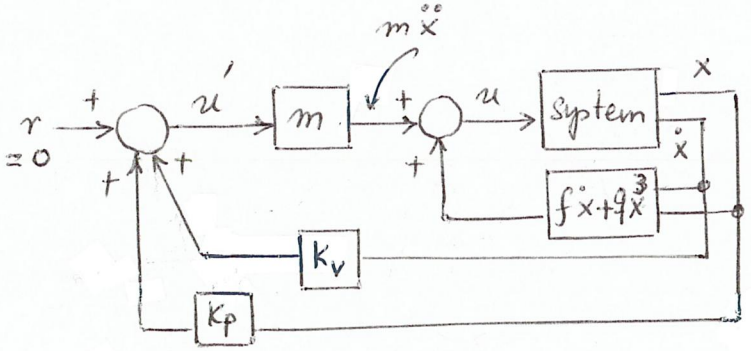
and we get

$$\ddot{x} + K_v \dot{x} + K_p x = 0$$

set $$s^2 + K_v s + K_p = s^2 + 2\zeta\omega_n s + \omega_n^2$$

↑ with desired ζ, ω_n for stabilization or performance

Once K_v and K_p are obtained, one can implement the control system as shown in the figure below.



For Trajectory_Following control we can also perform the same steps as we did for the linear case. Thus, the partitioning part remains the same as in the stabilization i.e.

$$u = \alpha u' + \beta$$

where

$$\alpha = m$$
$$\beta = b\dot{x} + qx^3$$

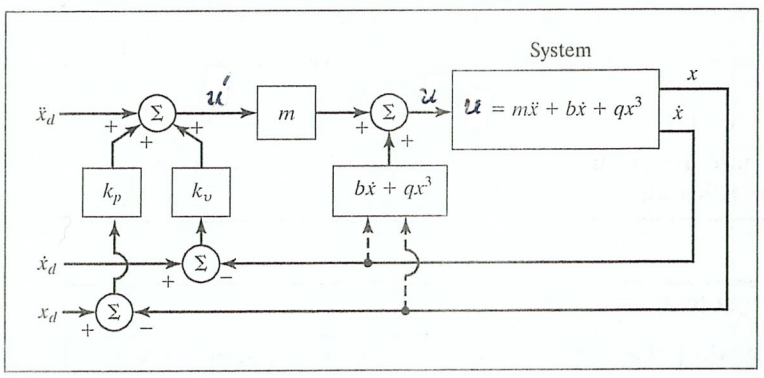
and the servo portion is, as we did for linear case becomes

$$u' = \ddot{x}_d + k_v \dot{e} + k_p e, \quad e = x_d - x$$

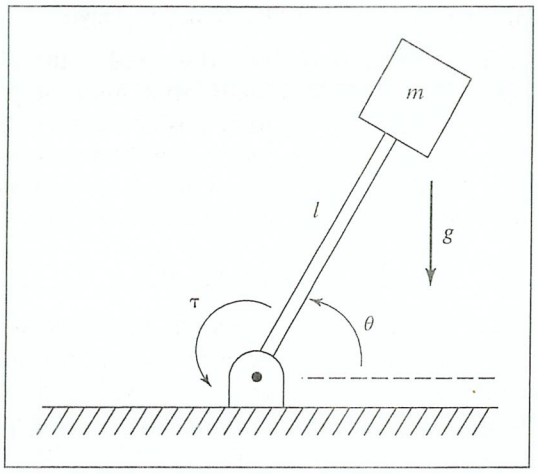
which leads to

$$\ddot{e} + k_v \dot{e} + k_p e = 0$$

Now, one can find k_v and k_p to realize trajectory following by making the error dynamics stable. The implementation of final controller is shown below.



As a follow-up example, consider the single-link manipulator shown below.



The model of the manipulator with viscous friction and coulomb friction is given by

$$\tau = ml^2\ddot{\theta} + v\dot{\theta} + c\,sgn(\dot{\theta}) + mlg\cos(\theta)$$

The control problem has two parts as before. The model-based portion of the control

$$u = \alpha u' + \beta$$

where

$$\alpha = ml^2$$

$$\beta = v\dot{\theta} + c\,sgn(\dot{\theta}) + mlg\cos(\theta)$$

The servo portion is obtained as

$$u' = \ddot{\theta}_d + k_v\dot{e} + k_p e$$

The rest is similar to the previous example.

Control of Manipulators "MIMO"

The rigid-body dynamics of robotic manipulators without and with friction terms are given by

$$\tau = \overset{n\times n}{M(q)}\,\ddot{q} + \overset{n\times 1}{V(q,\dot{q})} + \overset{n\times 1}{G(q)} \qquad q \text{ is } n\times 1$$

$$\tau = M(q)\,\ddot{q} + V(q,\dot{q}) + G(q) + \overset{n\times 1}{F(q,\dot{q})}$$

Control of the above systems can also be handled by the partitioned controller scheme that we have used for SISO case.

So, let

$$\tau = \alpha\,\tau' + \beta$$

control is $u \rightarrow \tau$ joint torques

"partitioning"

$$\alpha = M(q)$$
$$\beta = V(q,\dot{q}) + G(q) + F(q,\dot{q})$$

Then for stabilization we have $\ddot{q} = \tau'$

$$\tau' = -K_v\,\dot{q} - K_p\,q \qquad \text{"servo law"}$$

$$\Rightarrow \quad \ddot{q} + K_v\,\dot{q} + K_p\,q = 0$$

Using K_v and K_p with diagonal constant matrices we can decouple the above equation to stabilize the manipulator joint-by-joint as

$$\ddot{q}_i + K_{vi}\,\dot{q}_i + K_{pi}\,q_i = 0$$

or select K_{vi}, K_{pi}

$$s^2 + K_{vi}\,s + K_{pi} = s^2 + 2\zeta_i\,\omega_{ni}\,s + \omega_{ni}^2$$

with desired ζ_i and ω_{ni}

For Tracking we have the servo law

$$\tau' = \ddot{q}_d + K_V \dot{e} + K_p e$$

where $e = q_d - q$

Thus, the closed-loop system is characterized by the error dynamics

$$\ddot{e} + K_V \dot{e} + K_p e = 0$$

So, in this case K_V and K_p are chosen to stabilize the error dynamics joint-by-joint as

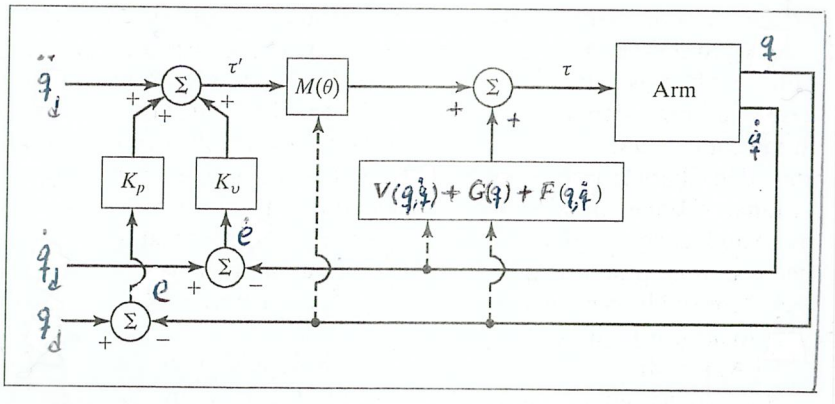
$$\ddot{e}_i + K_{Vi} \dot{e}_i + K_{pi} e = 0 \quad \leftarrow$$

choose diagonal matrices K_p, K_V with elements K_{pi}, K_{vi}

or

$$s^2 + K_{Vi} s + K_{pi} = s^2 + 2\zeta_i \omega_{ni} s + \omega_{ni}^2$$

choose ζ_i, ω_{ni} with desired performance



The case of disturbance rejection can also be handled similar to the case of SISO systems.

▷ Individual_joint PID Control

Most industrial robotic manipulators are controlled by a scheme with

$$\alpha = I$$
$$\beta = 0$$

so that the servo portion is

$$\tau' = \ddot{q}_d + K_V \dot{e} + K_P e + K_I \int e \, dt$$

where K_P, K_V, and K_I are diagonal constant matrices. In many cases $\ddot{q}_d$ is not available and it is set to zero, which means that the model-based component is not used in the control law.

▷ Method of Computed_Torque Control "State Space"

The partitioned control scheme, which is also known as the method of computed torque control can be formulated with state space notation. It is also common to devide the partitioning as two separate steps called "inner feedforward loop" and "outer loop feedback design"

Again consider the robotic manipulator dynamics

$$M(q)\ddot{q} + \underbrace{V(q,\dot{q}) + G(q)}_{N(q,\dot{q})} + \tau_d = \tau \qquad (1)$$

where we combine V and G terms as N and disturbance torque is introduced by τ_d.

Derivation of Inner Feedforward Loop

$$M(q)\ddot{q} + N(q,\dot{q}) + \tau_d = \tau \qquad (2)$$

$$\ddot{q} = M(q)^{-1}\left[\tau - \tau_d - N(q,\dot{q})\right]$$

Suppose that a desired trajectory $q_d(t)$ has been selected for the arm motion. To ensure trajectory tracking by the joint variable, define an output or tracking error as

$$e(t) = q_d(t) - q(t) \qquad (3)$$

To demonstrate the influence of the input $\tau(t)$ on the tracking error, differentiate twice $e(t)$ to obtain,

$$\dot{e} = \dot{q}_d - \dot{q}$$

$$\ddot{e} = \ddot{q}_d - \ddot{q}$$

Using $\ddot{q}$ from (2) and substituting into $\ddot{e}$ equation yields

$$\ddot{e} = \ddot{q}_d + M^{-1}(N - \tau_d - \tau) \qquad (4)$$

Defining the control input function

$$u = \ddot{q}_d + M^{-1}(N - \tau) \qquad (5)$$

and the disturbance function

$$W = M^{-1}\tau_d \qquad (6)$$

we have

$$\ddot{e} = u + W \qquad (7)$$

We may define a state vector $x(t) \in R^{2n}$ by

$$x = \begin{bmatrix} e \\ \dot{e} \end{bmatrix}$$

and write the tracking error dynamics as

$$\dot{x} = \begin{bmatrix} 0 & I \\ 0 & 0 \end{bmatrix} x + \begin{bmatrix} 0 \\ I \end{bmatrix} u + \begin{bmatrix} 0 \\ I \end{bmatrix} w \qquad (8)$$

The feedback linearizing transformation (5) may be inverted to get

$$\tau = M(\ddot{q}_d - u) + N \qquad (9)$$

we call this the computed torque control law. In fact substituting (9) into (2) yields

$$M\ddot{q} + N + \tau_d = M(\ddot{q}_d - u) + N$$

or

$$\ddot{e} = u + \underbrace{M^{-1}\tau_d}_{w}$$

which is exactly (7) or alternative state space representation (8).

Now, one can apply various techniques for stabilization, tracking and disturbance rejection using (8).

o Outer Loop Design using PD

One way to find the control law u is to use proportional-plus-derivative feedback

$$u = -K_v \dot{e} - K_p e$$

Then the overall robot arm input becomes

$$\tau = M(q)\left[\ddot{q}_d + K_v \dot{e} + K_p e\right] + N(q,\dot{q})$$

With respect to (8), the closed-loop system becomes

$$\dot{x} = \begin{bmatrix} 0 & I \\ -K_p & -K_v \end{bmatrix} x + \begin{bmatrix} 0 \\ I \end{bmatrix} w$$

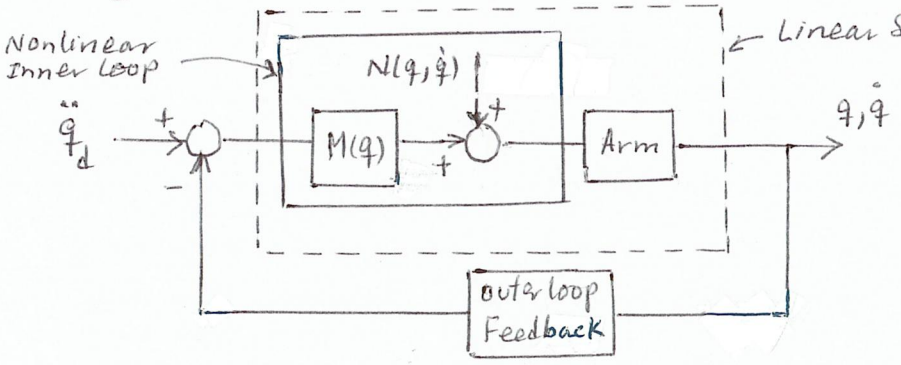
($\leftarrow$ closed-loop system matrix)

$K_v = \text{diag}\{K_{vi}\}$
$K_p = \text{diag}\{K_{pi}\}$

Select K_p, K_v as diagonal matrices such that the closed-loop system matrix becomes stable i.e. the closed-loop characteristic polynomial $\Delta_c(s)$ yields

$$\Delta_c(s) = \det\left(s^2 I + K_v s + K_p\right) = \Pi\left(s^2 + K_{vi} s + K_{pi}\right) =$$

$$\Delta_d(s) = \Pi\left(s^2 + 2\xi_i \omega_{ni} s + \omega_{ni}^2\right)$$ → "overall desired characteric poly."



The above figure shows the overall closed-loop control system.