Automatic Control Techniques Review: Part 2

Withit Chatlatanagulchai
Control of Robot and Vibration Laboratory (CRVLAB),
Department of Mechanical Engineering, Faculty of Engineering,
Kasetsart University, Bangkok, Thailand
E-mail: fengwtc@ku.ac.th

ABSTRACT

Existing control techniques in the engineering literature are diverse. This paper attempts to classify control techniques of which the author is aware from his literature review and experience with industrial control projects. Due to the vast number of techniques, most of the presentations will be brief; however, the reader will be pointed toward further excellent references. This paper should act as a starting point for readers, who may or may not have already become familiar with control theory, but are eager to see an overview of the control techniques, in order to choose techniques that suit their needs and to study them deeper. The work is divided into Part 1 and Part 2. In Part 1, techniques that have been discussed are those of basic control, adaptive control, and robust control. In this Part 2, techniques that will be discussed are those of nonlinear control, optimal control, and control supplements. The reader who is interested in the field of control and would like to know more details, is referred to an informative control handbook [1] and references therein.

Keywords :

control techniques, nonlinear control, optimal control, control supplements

1. NONLINEAR CONTROL

In fact, all a actual systems are nonlinear. However, the nonlinear model is usually too complicated for controller design. Therefore, control engineers have designed controllers from a linear model, which is viewed as an estimate of the nonlinear system. The linear control theory is simple and well understood.

In this section, we gather some techniques that devise control law straight from the nonlinear model. Nonlinear control theory is still an active and open research area.

1.1 Lyapunov Redesign

The Lyapunov's stability theories, considered to be a backbone of the nonlinear control theory, can be stated in a simple form, as follows. Suppose a nonlinear plant model to be controlled is given by

$$\dot{x} = f(x,u).$$

Let V be a positive energy-like function that contains all the state variables of $\dot{x} = f(x,u)$. If we can find a controller u that results in $\dot{V} < 0$, the energy-like function V and hence the state variables will reduce to zero. If the state variables are tracking errors, u is then a tracking controller that will drive the tracking error to zero.

The Lyapunov redesign method uses the Lyapunov function of the nominal system $\dot{x} = f(x,u)$. to design an additional control component for uncertain systems, making the new controller robust. An example of an uncertain system is

$$\dot{x} = F(x) + G(x)[u + \delta(t, x, u)], \quad (1)$$

where F and G are known functions and $\delta(t, x, u)$ is an unknown function with known bounds that lumps together various uncertain terms due to model uncertainty and disturbances.

1.2 Nonlinear Damping

Nonlinear damping is an additional controller term that can be viewed as providing additional damping to the system, making the system robust against broader classes of uncertainty and disturbances. The nonlinear damping term is widely used in the control community and works by dissipating energy from the system, making it more stable. One example of using the nonlinear damping term is as follows. Consider the uncertain system (1) with

$$\delta(t, x, u) = \Gamma(t, x)\, \delta_0(t, x, u),$$

where Γ is precisely known but δ_0 and its bound are unknown. It can be proved that a controller

$$u = \psi(t, x) + v,$$

where ψ is designed for the nominal system and v is the additional nonlinear damping term, can achieve uniform boundedness of the solution of (1). The additional nonlinear damping term is given by

$$v = -kw\left|\left|\Gamma(t, x)\right|\right|_2^2, \; k > 0,$$

where $w^T = [\partial V \mid \partial x]G.$

1.3 Backstepping

The backstepping technique breaks the design problem for the full system into a sequence of design problems for lower-order subsystems. Virtual control is designed for each subsystem with the objective of reducing the error between the local state of the subsystem and its desired value. Backstepping is useful for our control strategy, since it allows control effort to be inserted into each subsystem. Therefore, uncertainties are allowed to exist at each subsystem.

Figure 1 depicts an example of backstepping control for a 3rd-order system where all functions are known and all states are available. Suppose the objective is to make x_1 track x_{1d} as closely as possible. We let $z_i = x_i - x_{id}$, $i = 1, 2, 3$ be the error at each subsystem. The virtual control inputs are given by

$$z_{2d} = G_1^{-1}(-F_1 + \dot{x}_{1d} - k_1 z_1),$$

$$z_{3d} = G_2^{-1}(-F_2 + \dot{x}_{2d} - k_2 z_2 - G_1 z_1),$$

and the actual control is given by

$$u = G_3^{-1}(-F_3 + \dot{x}_{3d} - k_3 z_3 - G_2 z_2).$$

By using the Lyapunov candidate $V = 0.5\,(z_1^2 + z_2^2 + z_3^2)$, it can be shown that all the errors approach zero asymptotically.

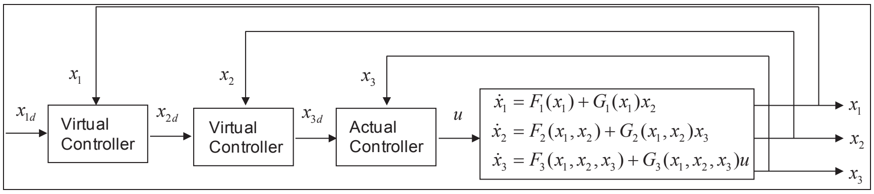

Figure 1 : Diagram of a Backstepping Control

1.4 Sliding Mode Control

Sliding mode control is a robust control that has fast action to counteract disturbances. There are two phases: reaching and sliding phases. In the reaching phase, initial state trajectories are controlled to move toward a sliding surface and, once on the surface, to maintain it there. The sliding surface is designed such that any trajectories, when on the surface, move toward the origin. Figure 2 depicts a phase portrait under the sliding mode control.

The controller during the reaching phase is normally a fast-acting control law of the form $u = -\beta(x)\ \mathrm{sgn}\ (s)$, where $\beta(x)$ satisfies some properties, sgn represents the signum function, and s is the sliding surface. This control law moves any initial trajectories toward the surface $s = 0$ and keeps them on the surface thereafter. The surface $s = 0$ is designed such that any trajectories on it move toward the origin.

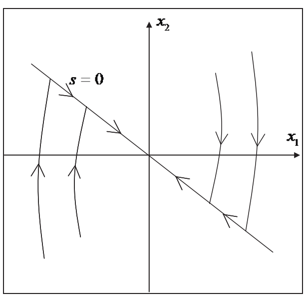

Figure 2 : Typical Phase Portrait under Sliding Mode Control

In tracking problems, the trajectories x can be set as the tracking error. Hence, the sliding mode control is able to achieve zero error once the trajectories are driven to the origin. The controller is robust against model uncertainty and disturbances, because designing the sliding surface does not require the plant model and the fast-acting control law can be designed to overcome the effects of the disturbances.

There are several ways to reduce control chattering from the discontinuous signum function in the control law. The most convenient way is to replace the signum function by a smooth version of it, for example, by an inverse tangent function.

1.5 Passivity-Based Control

A MIMO system

$$\dot{x} = f(x,u),\ y = h(x)$$

is passive if there exists a continuously differentiable positive semi-definite function $V \geq 0$ such that $u^T y \geq \dot{V}$. A passive system has a stable origin. All we need to asymptotically stabilize the origin is to inject damping $u = -\phi(y)$ into the system, so that the energy will dissipate whenever $x(t)$ is not identically zero.

For a non-passive system, the controller contains two parts: one to transform the non-passive system to a passive system, the other to inject damping into the system to achieve zero $x(t)$. For example, a system

$$\dot{x} = f(x) + G(x)u, y = h(x)$$

may use a control law $u = \alpha(x) + \beta(x)v$ with $v = -\phi(y)$ to drive the state $x(t)$ to zero.

1.6 Singular Perturbation Control

Singular perturbation control is a controller designed from a so-called standard singular perturbation model

$$\dot{x} = f(t, x, z, \varepsilon),$$
$$\varepsilon \dot{z} = g(t, x, z, \varepsilon),$$

where ε is a small number, x is a state variable in the slow time scale, and z is a state variable in the fast time scale.

The singular perturbation method is suitable for systems that can be divided into fast and slow time scales. For example, an actuator system such as a motor is usually operated faster than the plant, which is a system driven by the motor.

1.7 Linearization about Equilibrium Point

In general, controller design and analysis of linear systems are well-understood compared to those of nonlinear systems, since most practical systems operate only around a limited number of operating points. The nonlinear model can be linearized about these operating points using the first-order Taylor series. Hence, the nonlinear model becomes many linear models about several operating points. For each linear model, a linear control system can then be designed, and a gain scheduling algorithm can be used to switch among the controllers in a smooth or abrupt way.

A general nonlinear system

$$\dot{x} = f(x, u)$$

has its first-order Taylor series approximation about the origin as

$$\dot{x} = Ax + Bu,$$

where

$$A = \frac{\partial f}{\partial x}(x, u)\Big|_{x=0,\, u=0},\ B = \frac{\partial f}{\partial u}(x, u)\Big|_{x=0,\, u=0}.$$

This also applies to any operating point, since any point can be transferred to the origin by a change of variables.

Then, any control design techniques for linear systems can be applied. For example, a controller

$$u = -Kx,\ K > 0$$

results in the closed-loop system

$$\dot{x} = (A - BK)\, x,$$

whose poles can be placed by the state-feedback controller.

1.8 Feedback Linearization

Instead of approximating about an operating point, feedback linearization is a technique that uses feedback and possibly a change of variables to transform nonlinear systems to linear systems. This linearization approach is exact, not approximated as in the previous linearization technique. However, the feedback linearization technique only applies to a special class of nonlinear systems.

There are two types of feedback linearization. The first is linearizing the mapping from input to state; and the second is linearizing the mapping from input to output.

1.8.1 Input-to-state linearization

A nonlinear plant model

$$\dot{x} = Ax + By(x)[u - \alpha(x)] \qquad (2)$$

can be feedback linearized by a controller

$$u = \alpha(x) - \beta(x)v,$$

where $\beta(x) = y^{-1}(x)$, to obtain a linear plant model

$$\dot{x} = Ax + Bv.$$

It can be shown that a broader nonlinear model in the form

$$\dot{x} = f(x) + G(x)u$$

can be transformed to (2) using a change of variables $z = T(x)$. The change of variables requires some assumptions to be met and in general is not applicable for some types of systems.

1.8.2 Input-to-output linearization

Input-to-output linearization uses feedback to achieve linear mapping from input to output. Suppose we want to control certain output variables. Linearizing the state equation, as was done in input-to-state linearization, does not necessarily linearize the output equation. For example, consider a system

$$\dot{x}_1 = a \sin x_2,$$
$$\dot{x}_2 = -x_1^2 + u,$$
$$y = x_2.$$

The change of variables and state-feedback control

$$z_1 = x_1,\ z_2 = a \sin x_2,$$
$$u = x_1^2 + \frac{1}{a \cos x_2} v$$

yield

$$\dot{z}_1 = z_2,$$
$$\dot{z}_2 = v,$$
$$y = \sin^{-1}\left(\frac{z_2}{a}\right).$$

In the system above, the state equation is linear, but y is nonlinear, resulting in complications in solving the tracking control problem.

Suppose, instead, we use

$$u = x_1^2 + v,$$

we would get

$$\dot{x}_1 = a \sin x_2,$$
$$\dot{x}_2 = v,$$
$$y = x_2,$$

Note that the state variable x_1 does not connect to the output y but the input-output map from v to y is linear. This is called input-to-output linearization.

When we design tracking control, we must make sure that the variable x_1 is well behaved, that is, stable or bounded. A naive control design that does not consider x_1 might end up with an ever-growing, unstable signal x_1 (internal instability.)

In general, the input-to-output linearization technique can be applied to a nonlinear system in the form

$$\dot{x} = f(x) + g(x)u,$$
$$y = h(x)$$

with some restrictions.

Nonlinear control can be studied from many excellent textbooks. For example, we have used [2]-[11].

2. OPTIMAL CONTROL

Given a plant model

$$\dot{x} = h(x, u, t),$$

an optimal control determines a control input u that causes the plant to satisfy some physical constraints and at the same time optimizes a certain performance criterion,

$$J = \int_{t_0}^{t_f} f(x, u, t)\ dt,$$

which can include, for example, control effort, state trajectories, or initial conditions.

2.1 Time Optimal Control (TOC)

The objective of the time optimal control is to drive the system's output from one point to another using the shortest time possible. For a simple second-order plant without spring and damping

$$\ddot{y}(t) = au(t), \tag{3}$$

it can be shown by solving the optimization problem that

$$u(t) = \begin{cases} +u_{\max}, & for\ t \in \left(0, \frac{t^*}{2}\right] \\ -u_{\max}, & for\ t \in \left(\frac{t^*}{2}, t^*\right] \end{cases} \tag{4}$$

drives the output y from 0 to r in the shortest time possible, where

$$t^* = \sqrt{\frac{4r}{au_{\max}}}$$

is the time to reach the target and $u_{\max}$ is the maximum allowable control effort.

The open-loop control law in the previous paragraph suffers from model uncertainty and disturbances. To receive the benefits of closed-loop control, the following closed-loop TOC is proposed. For the plant (3) and control law (4), by eliminating t in their state-space model, we obtain

$$x_1(t) = \frac{1}{2au_{\max}} x_2^2(t) + c_5, \quad for\ u = +u_{\max},$$

$$x_1(t) = -\frac{1}{2au_{\max}} x_2^2(t) + c_6, \quad for\ u = -u_{\max},$$

where c_5 and c_6 are appropriate constants. The phase plot between $e = r - x_1$ and x_2 is then given as shown in Figure 3. From the phase plot, we see that by appropriately switching u between $\pm u_{\max}$. the trajectory will be driven to the origin.

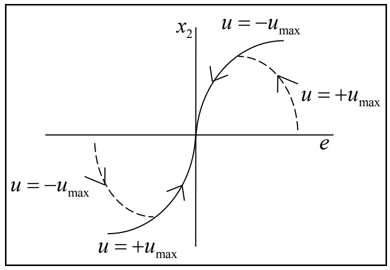

Figure 3 : Phase Plot between e and x_2

Let $f_t(e) = \text{sgn}(e)\sqrt{2au_{\max}|e|}$ be an equation representing the solid line in the phase plot. The control law is then given by $u = u_{\max}\,\text{sgn}(f_t(e) - x_2)$, and the closed-loop block diagram is shown in Figure 4.

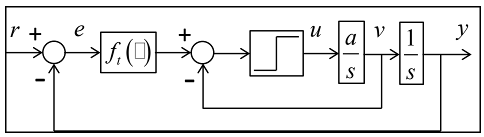

Figure 4 : Typical Closed-loop Time-optimal Control

2.2 Proximate Time Optimal Servomechanism (PTOS)

The previous closed-loop TOC has two drawbacks. First, even the smallest system process or measurement noise will cause control "chatter", which wastes energy and can excite the high-frequency modes. Second, any error in the plant model will cause limit cycles to occur.

The two drawbacks can be eliminated by replacing the signum function with the saturation function with smooth slope. An example is

$$u = u_{\max} sat\left(\frac{k_2\left[f_p(e) - x_2\right]}{u_{\max}}\right),$$

where

$$sat(x)=\begin{cases}+1, & if\ x>1,\\ x, & if\ -1\le x\le 1,\\ -1, & if\ x<-1,\end{cases}$$

and

$$f_p(e)=\begin{cases}\dfrac{k_1}{k_2}e, \text{ for } |e|\le y_l,\\ \operatorname{sgn}(e)\left[\sqrt{2au_{\max}\alpha|e|}-\dfrac{u_{\max}}{k_2}\right],\\ \text{for } |e|>y_l,\end{cases}$$

where k_1 and $0<\alpha<1$ are designed parameters and $y_l = u_{\max}/k_1$ and $k_2=\sqrt{2k_1/a\alpha}$. More details of the proof can be found in [12]. A closed-loop block diagram is given in Figure 5.

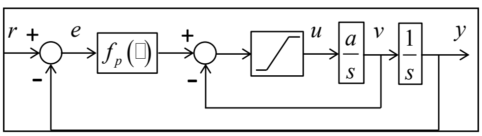

Figure 5 : Typical Proximate Time-optimal Servomechanism

Details of standard time-optimal control can be found in [13] and [14].

2.3 LQR (Linear Quadratic Regulator)

Given the system equation

$$\dot{x} = Ax + Vu,$$

the LQR method determines the matrix K of the optimal control input

$$u = -Kx$$

to minimize the performance index

$$J = \int_0^{\infty} (x^*Qx + u^*Ru)\ dt, \qquad (5)$$

where Q and R are positive-definite Hermitian matrices that give relative importance of reducing the states or reducing the control input used. To minimize J, we substitute $u = -Kx$ into (5) and set $\partial J / \partial K = 0$. It is then not difficult to show that $K = R^{-1} B^* P$, where P satisfies the so-called reduced-matrix Riccati equation $A^* P + PA - PBR^{-1} B^* P + Q = 0$. Detailed derivation can be found in chapter 12 of [15].

Minimizing J results in minimizing the states x and the control effort u, making them close to zero, which is desirable if we want to drive x to zero or use small amount of control effort.

2.4 LQG/LTR (Linear Quadratic Gaussian/ Loop Transfer Recovery)

It can be said that LQG is an LQR with a state observer. It was proved that although LQR and state observer both have good stability margins, when combined to be LQG, there is no guaranteed stability margin. This is because the loop transfer functions at input and output of the LQG controller differ from those of LQR and state observer and have no guaranteed stability margin. The method to make the loop transfer of the LQG controller approach those of the LQR and the observer is called loop transfer recovery or LTR.

However, it is argued that the loop transfer recovery does not work in practice, due to two main reasons. First, the method involves canceling the plant zeros, so it cannot be used with a non-minimum-phase plant. Second, the method requires infinity gains to completely recover the loop transfer function, so the high gains can cause instability with unmodeled dynamics.

Nevertheless, if the plant is minimum phase and the plant model is accurate, the LQG/LTR method still applies. One more note is that not having a guaranteed margin does not mean the closed-loop system does not have any stability margin. Therefore, we would still see many applications using the LQG/LTR method successfully.

Figure 6 shows a diagram of the LQG controller. The plant model is in the form

$$\dot{x} = Ax + Bu, + w_d,$$

$$y = Cx + Du, + w_n,$$

where w_d and w_n are Gaussian white noises. The plant model is linear and can be with or without disturbance w_d and noise w_n. The LQR control using the estimated state is $u = -K\hat{x}$.. The state observer used to estimate states can be a Kalman filter (with noise) or a Luenberger observer (without noise.) The observer will be discussed in a later section.

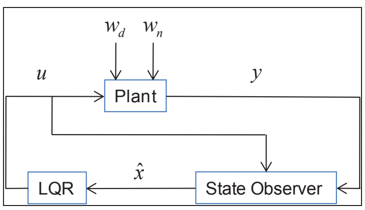

Figure 6 : Diagram of the LQG Controller

LQG/LTR is discussed in details in [16].

2.5 Model Predictive Control (MPC)

Model predictive control computes a sequence of control input that minimizes the error between the predicted output and the reference trajectory. The method is most suitable with chemical processes, where there is more computation time available between samples for the more complex control algorithm and there is a need to handle the time delay between applied input and resulting output.

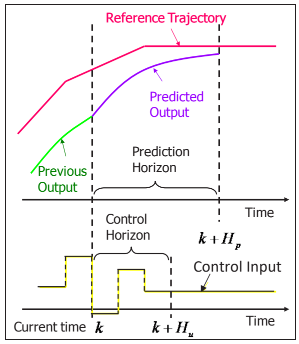

Figure 7 : Diagram Illustrating the Model Predictive Control Concept

Figure 7 depicts a diagram explaining the concept of the model predictive control. At the current time step k, the output $y(k)$ is measured. The control input sequence $\{u(k), u(k+1),..., u(k+H_u)\}$ is then computed over the control horizon H_u by minimizing the error between the predicted output and the reference trajectory computed over the prediction horizon H_p. After the input sequence is computed, only the first element $u(k)$ is implemented at the next time step.

For systems with time delay, the current input $u(k)$ will affect the future output $y(k + T_d)$ only after a delay time T_d. In MPC, since $u(k)$ is computed to minimize the tracking error over the prediction horizon H_p, if $H_p >> T_d$, the effect of time delay is practically eliminated.

In MPC, constraints such as bounds on input effort u, bounds on input effort rate Δu, and bounds on output y, can also be specified by imposing these constraints on the optimization problem.

Model predictive control is discussed in [17].

2.6 Extremum Seeking

Extremum seeking is a gradient-based method to find a set of parameters that locally optimizes a cost function. What makes this technique interesting and practical is that the technique does not require plant model, as is a typical requirement in any gradient-based algorithm. Hence, it is convenient to apply this technique to complicated systems whose models are difficult to find.

An additive probing term $\cos(wk)$ is used in the algorithm, so that the gradient of the cost function can be estimated without the plant model. More details can be found in [18].

One of the effective uses of this technique is to tune PID gains, as was presented in [19]. Figure 8 contains a diagram of this scheme.

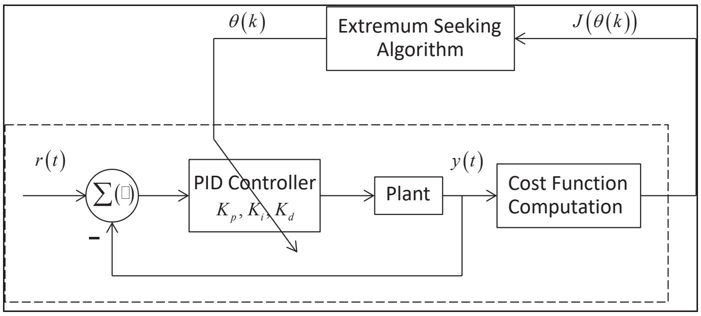

Figure 8 : Extremum-seeking PID Tuning Scheme

The cost function is given by

$$J(\theta) = \frac{1}{T - t_0} \int_{t_0}^{T} e^2(t, \theta)\, dt,$$

where $[t_0, T]$ is the time interval to compute the cost function, $e = r - y$ is the tracking error, t is the current time, and $\theta = [K_p, K_i, K_d]^T$ is the PID gains to be adjusted.

k represents an epoch. In the k^{th}-epoch, a step reference r is given to the system to obtain the output y. Then, the cost function $J(\theta)$ is computed over the time interval $[t_0, T]$, which normally includes transient and steady-state periods. Afterward, the cost function of the k^{th}-epoch, $J(\theta(k))$ is sent to the extremum-seeking algorithm to compute $\theta(k) = [K_p, K_i, K_d]^T$ which will be used in the next $k+1^{th}$ epoch. The process repeats until the cost function cannot be optimized further, then the optimum PID gains are obtained.

3. SUPPLEMENTS

In this section, we present some techniques that help improve the quality of the control system already in place.

3.1 Augmented Integrator

Adding an integrator is known to eliminate steady-state error. An integrator can be added in parallel to the existing controller, built into the existing controller structure, or augmented from the plant model. The first and second cases are obvious. In this section we discuss the augmented integrator.

Suppose we have a state-space model representing a plant to be controlled

$$\dot{x}_1 = f_1(x, u),$$
$$\dot{x}_2 = f_2(x, u),$$

where in the tracking problem x_1 usually represents the tracking error. We can augment an integrator to the system by adding another state variable as

$$\dot{x}_0 = x_1,$$
$$\dot{x}_1 = f_1(x, u),$$
$$\dot{x}_2 = f_2(x, u),$$

By designing the control input u to asymptotically stabilize the system, that is, all states approach zeros in finite time, x_0, which is the integral of the tracking error, also approaches zero in a sense analogous to having an integrator in the system.

3.2 Observer

Not all states are measured in practice due to cost limitations, lack of space, or no available sensor technology. For example, we may install an optical encoder to measure the position signal but not the velocity sensor. An observer is designed to estimate the missing states from signals we know, which are measured output y and control effort u. There are three broad types of the observer.

3.2.1 Luenberger observer

Luenberger observer is a standard observer for linear systems. Consider a linear state-space plant model

$$\dot{x} = Ax + Bu,$$
$$y = Cx.$$

The Luenberger observer is given by

$$\dot{\hat{x}} = A\hat{x} + Bu + L\left(y - C\hat{x}\right),$$

where $\hat{x}$ is the estimated state and L is the constant gain to be designed. It can be shown that, if we let $e = x - \hat{x}$ be the estimated error, we would get $\dot{e} = \left(A - LC\right)e$. Therefore, L can be easily designed to place the closed-loop pole to achieve zero estimated error.

3.2.2 Kalman filter

Kalman filter is a powerful technique to estimate the states of the system when there is plant-input disturbance and the output measurement is crippled with noise.

Consider the discrete plant model

$$x[n+1] = Ax[n] + B\left(u[n] + w[n]\right),$$
$$y[n] = Cx[n],$$
$$y_v[n] = Cx[n] + v[n],$$

where w and v are some Gaussian white noises. The available signals to us are the control input u and the measured output y_v. Note that the noise w is applied at the plant input, and the noise v is sensor noise, which makes the measured output y_v differ from the actual output y. Since we want to control the actual output y and not the measured output y_v. we use a Kalman filter to estimate the actual output y using u and y_v.

There are two stages in the Kalman filter algorithm. First, the measurement update stage is when the state estimate is updated for the current time step n using the measured output just received at the current time step $y_v[n]$. The algorithm is given by

$$\hat{x}[n \mid n] = \hat{x}[n \mid n-1] + M\left(y_v[n] - C\hat{x}[n \mid n-1]\right),$$

where the innovation gain M is chosen to minimize the steady-state covariance of the estimation error given the noise covariances

$$E\left(w[n]w[n]^T\right) = Q, E\left(v[n]v[n]^T\right) = R.$$

For notation, $\hat{x}[n \mid n-1]$ is the estimate of $x[n]$ given past measurements up to $y_v[n-1]$, and $\hat{x}[n \mid n]$ is the updated estimate based on the last measurement $y_v[n]$. Second, the time update stage is when the control input at the current time step n is used to compute the estimated state of the next time step $n + 1$. The algorithm is given by

$$\hat{x}[n+1 \mid n] = A\hat{x}[n \mid n] + Bu[n].$$

We can combine the time and measurement update equations into one state-space model (called the Kalman filter)

$$\hat{x}[n+1 \mid n] = A(I - MC)\hat{x}[n \mid n-1] + [B, AM]\begin{bmatrix} u[n] \\ y_v[n] \end{bmatrix},$$

$$\hat{y}[n \mid n] = C(I - MC)\hat{x}[n \mid n-1] + CMy_v[n].$$

This filter generates an optimal estimate $\hat{y}[n \mid n]$ of $y[n]$ Note that the filter state is $\hat{x}[n \mid n-1]$.

The Kalman filter above is called a steady-state Kalman filter. When the plant is time-varying or is with non-stationary noise covariance, we can use a generalization of the steady-state Kalman filter called time-varying Kalman filter. The time-varying Kalman filter is given as follows. Consider the discrete plant model

$$x[n+1] = Ax[n] + Bu[n] + Gw[n],$$
$$y[n] = Cx[n],$$
$$y_v[n] = Cx[n] + v[n].$$

The measurement update stage is given by

$$\hat{x}[n \mid n] = \hat{x}[n \mid n-1] + M[n]\left(y_v[n] - C\hat{x}[n \mid n-1]\right),$$
$$M[n] = P[n \mid n-1]C^T \left(R[n] + CP[n \mid n-1]C^T\right)^{-1},$$
$$P[n \mid n] = \left(I - M[n]C\right)P[n \mid n-1].$$

The time update stage is given by

$$\hat{x}[n+1 \mid n] = A\hat{x}[n \mid n] + Bu[n],$$
$$P[n+1 \mid n] = AP[n \mid n]A^T + GQ[n]G^T,$$

where

$$Q[n] = E\left(w[n]w[n]^T\right),$$
$$R[n] = E\left(v[n]v[n]^T\right).$$

Details of the Kalman filter can be found in [20].

3.2.3 High-gain observer

High-gain observer applies to various broader types of nonlinear systems. As an example, consider a second-order nonlinear system

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = \phi(x, u),$$
$$y = x_1,$$

where $x = [x_1, x_2]^T$.

Suppose $u = \gamma(x)$ is a state feedback control law that stabilizes the origin x = 0 of the closed-loop system

$$\dot{x}_1 = x_2,$$
$$\dot{x}_2 = \phi(x, \gamma(x)).$$

The high-gain observer is given by

$$\dot{\hat{x}}_1 = \hat{x}_2 + h_1(y - \hat{x}_1),$$
$$\dot{\hat{x}}_2 = \phi_0(\hat{x}, u) + h_2(y - \hat{x}_1),$$

where $\phi_0(x, u)$ is the nominal model of the nonlinear function $\phi(x, u)$.

Let

$$\tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} = \begin{bmatrix} x_1 - \hat{x}_1 \\ x_2 - \hat{x}_2 \end{bmatrix}$$

be the estimation error. We have

$$\dot{\tilde{x}}_1 = -h_1\tilde{x}_1 + \tilde{x}_2,$$
$$\dot{\tilde{x}}_2 = -h_2\tilde{x}_1 + \delta(x, \tilde{x}),$$

where $\delta(x, \tilde{x}) = \phi(x, \gamma(\hat{x})) - \phi_0(\hat{x}, \gamma(\hat{x}))$.

We want to design the observer gain $H = [h_1, h_2]^T$ so that $_{t \to \infty} \tilde{x}(t) = 0$ Without the disturbance term δ, asymptotic error convergence is achieved by designing H such that

$$A_0 = \begin{bmatrix} -h_1 & 1 \\ -h_2 & 0 \end{bmatrix}$$

is Hurwitz, which is when h_1 and h_2 are any positive constants.

With the disturbance term δ, the observer gain H can be designed with an additional goal of rejecting the effect of δ on $\tilde{x}$.

Details of the high-gain observer can be found in [2].

3.3 Output Feedback

If all state variables are measured and used in the control law, the controller is called a state-feedback controller, but if some state variables are not measured and their estimates from the state observer are used in the control law, the controller is called an output-feedback controller. There are two categories.

3.3.1 Full-order

This is when all state variables, including the measured output, are estimated, and only the estimated state variables are used in the control law. To make this point clear, consider a plant model given by

$$\dot{x}_1 = f_1(x, u),$$
$$\dot{x}_2 = f_2(x, u),$$
$$y = x_1,$$

where y is the measured output. A state observer uses the control input u and the measured output y to obtain estimated state variables $\hat{x}_1$ and $\hat{x}_2$. In full-order output feedback, a control law $u = f(\hat{x}_1, \hat{x}_2)$ is devised in terms of the estimated state variables.

3.3.2 Reduced-order

Instead of using $\hat{x}_1$ in the control law, we may use its actual value EMBED Equation.DSMT4 resulting in a reduced-order output feedback control law $u = f(y, \hat{x}_2)$. One advantage of the reduced-order control over the full-order control is that it has lower order and hence consumes less computing time when implemented.

Details on the output feedback control for linear systems can be found in any standard textbooks that discuss state-space model, for example, [15]. The output feedback control for non-linear systems can be found, for example, in [2].

3.4 Signal Shaping

Signal shaping shapes signal so that its spectrum energy around the system's natural frequencies is reduced to avoid resonance. There are two signal shaping techniques, the so-called input shaping and command shaping.

Input shaping convolves the reference input with a sequence of pre-designed impulses. The resulting shaped reference input has practically reduced spectrum energy around the designed natural frequencies. The shaped reference input is normally shaped around the closed-loop system's natural frequencies to avoid exciting them. As a result, we obtain a smoother output.

Command shaping reconstructs the reference input from some basis functions, such as ramped sinusoidal function or versine function. The new reference input has its spectrum energy reduced around the system's natural frequencies to avoid resonance.

Another shaping, but usually performed on the system itself, is the use of a notch filter. A notch filter is in the form

$$G_n(s) = \frac{s^2 + \omega_n^2}{s^2 + \frac{\omega_n}{Q}s + \omega_n^2}.$$

Its Bode's magnitude plot reveals that there is a trough around frequencies near ω_n. The depth is determined by Q. The notch filter is used to multiply directly with the system to reduce its gain around the natural frequency. Multiple notch filters can be used when there are more-than-one natural frequencies.

Details of the input shaping and the command shaping techniques can be found in the various publications posted at http://www.crvlab.eng.ku.ac.th/.

3.5 Composite Control

The idea is to combine two or more controllers in a control law. For example, in step reference tracking, the first controller to be used may possess low damping for fast movement. However, when the controlled output approaches the reference, another controller with higher damping is used to break the system to reduce overshoot. Figure 9 depicts this scenario. In the literature, if the switching between controllers is gradual, the controller is called composite control, but if the switching is abrupt, the controller is called mode switching control.

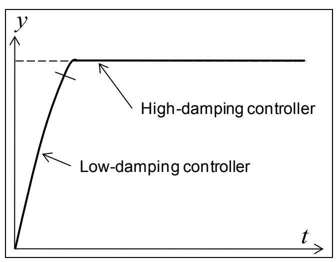

Figure 9 : Diagram of a Composite Control

The idea of composite and switching controls first came to the author from [21].

3.6 Repetitive Control (RC) and Iterative Learning Control (ILC)

For a system that is required to track the same reference input many times, such as a robot manipulator, or to reject periodic disturbance, such as repetitive runout rejection in a hard disk drive's head positioning, RC or ILC can be used to improve the controller's performance by learning from the previous movement of the system.

Suppose a reference input $y_d(t)$ repeats every $t = T$ seconds, that is, $y_d(T) = y_d(0)$. Let $\tau_k(t)$ be the control input to the system during the k^{th} cycle, which produces an output $y_k(t)$, $0 \le t \le T$. The input/output pair $[\tau_k(t), y_k(t)]$ is stored and used to improve tracking of the next $k+1^{th}$ cycle. The learning control problem is to determine a recursive control

$$\tau_{k+1}(t) = F(\tau_k(t), \Delta y_k(t)),\ 0 \le t \le T,$$

where $\Delta y_k(t) = y_k(t) - y_d(t)$, such that $\|\Delta y_k\| \to 0$ as $k \to \infty$.

Several approaches have been used to generate a suitable learning law F. A PID-type learning law takes the form

$$\tau_{k+1}(t) = \tau_k(t) - k_p \Delta y_k(t) - k_i \int \Delta y_k(t)\, dt - k_d \frac{d\Delta y_k(t)}{dt}.$$

The RC and ILC controls are attractive because no plant model is required and they can be used to improve any control law already in use for tracking of the repetitive reference. More details of the RC and ILC can be found in [22].

3.7 Digital Control and Controller Implementation

Nature creates analog signals (in continuous time), but we perceive these signals as digital signals (in discrete time.) In fact, it is believed that the human brain has a very fast sampling period. The implementation of the controller is also done in discrete time with signals sampling with a certain period.

Digital control designs the controller directly in discrete-time domain, resulting in either transfer function in a z-operator or a discrete-time state-space model. However, most of the controller design techniques available in the literature are performed in continuous time. This is probably because traditional mathematics has always assumed continuous function, and some theories exist only in continuous time and do not have any meaning in discrete time.

Although we can almost always find a discrete-time counterpart of the continuous-time control, it may be more convenient to,instead of designing a digital control anew, approximate the existing continuous-time controller. The continuous-time controller is mostly given in a transfer function in s-operator (for example, QFT) or continuous-time state-space model (for example, adaptive control algorithm.)

For the transfer function in s-operator, we can simply let $s = (z-1) / T$, for forward difference or Euler's method, $s = (z-1) / (zT)$, for the backward difference method, $s = \left[2\left(z-1\right)\right] / \left[T\left(z+1\right)\right]$, for Tustin's approximation method, or

$$s = \frac{\omega_1}{\tan\left(\omega_1 h/2\right)} \cdot \frac{z-1}{z+1},$$

for Tustin with prewarping method, where z is the z-transform operator and T is the chosen sampling period. The resulting transfer function in z-operator differs slightly from the original transfer function in s-operator, usually in some high-frequency ranges. By prewarping, more accuracy is obtained at $\omega = \omega_1$.

The resulting transfer function in z-operator can then be implemented using the time-shifting theorem

$$Z\left(x\left(t-nT\right)\right) = z^{-n} X\left(z\right).$$

For example, a controller

$$G\left(z\right) = \frac{5.469z - 5.43}{z^2 - 1.63z + 0.6562} = \frac{5.469z^{-1} - 5.43z^{-2}}{1 - 1.63z^{-1} + 0.6562z^{-2}} = \frac{U\left(z\right)}{E\left(z\right)}$$

results in

$$u\left(t\right) = 5.469e\left(t-T\right) - 5.43e\left(t-2T\right) + 1.63u\left(t-T\right) - 0.6562u\left(t-2T\right)$$

after cross multiplication and inverse transformation with the time-shifting theorem.

For the continuous-time state-space model, the approximation might not be as simple. However, numerical approximation of the derivative of the state usually suffices. For example, in Euler's method,

$$\dot{x} \approx \frac{x\left(t+T\right) - x\left(t\right)}{T},$$

which is readily implementable.

Many excellent textbooks discuss digital control. We have used [23] and [24]. The controller implementation is covered in [25].

4. REFERENCES

[1] Levine WS (1996). The Control Handbook. Boca Raton, FL: CRC Press.

[2] Khalil HK (2002). Nonlinear Systems. Upper Saddle River, NJ: Prentice Hall.

[3] Sastry S (1999). Nonlinear Systems: Analysis, Stability, and Control. New York, NY: Springer.

[4] Slotine JJE. and Li W. (1991). Applied Nonlinear Control. Upper Saddle River, NJ: Prentice Hall.

[5] Vidyasagar M (2002). Nonlinear Systems Analysis. Philadelphia, PA: SIAM.

[6] Isidori A. (1995). Nonlinear Control Systems. New York, NY: Springer.

[7] Isidori A. (1999). Nonlinear Control Systems II. New York, NY: Springer.

[8] Spooner JT., Maggiore M and Ordonez R. (2001). Stable Adaptive Control and Estimation for Nonlinear Systems: Neural and Fuzzy Approximator Techniques. Hoboken, NJ: Wiley.

[9] Kokotovic P., Khalil, HK. and Oreilly J. (1987). Singular Perturbation Methods in Control: Analysis and Design. Philadelphia, PA: SIAM.

[10] Qu Z. (1998). Robust Control of Nonlinear Uncertain Systems. Hoboken, NJ: Wiley.

[11] Krstic M., Kanellakopoulos I. and Kokotovic PV. (1995). Nonlinear and Adaptive Control Design. Hoboken, NJ: Wiley.

[12] Workman ML. (1987). Adaptive proximate time optimal servomechanisms. PhD Dissertation, Information Systems Laboratory, Stanford University, USA.

[13] Lewis FL. and Syrmos VL (1995). Optimal Control. Hoboken, NJ: Wiley.

[14] Kirk DE. (2004). Optimal Control Theory: An Introduction. Mineola, NY: Dover.

[15] Ogata K. (2002). Modern Control Engineering. Upper Saddle River, NJ: Prentice Hall.

[16] Anderson BDO. and Moore JB. (2007). Optimal Control: Linear Quadratic Methods. Mineola, NY: Dover.

[17] Maciejowski JM. (2000). Predictive Control with Constraints. Upper Saddle River, NJ: Prentice Hall.

[18] Ariyur KB. and Krstic M. (2003). Real-Time Optimization by Extremum-Seeking Control. Hoboken, NJ: Wiley.

[19] Killingsworth NJ. and Krstic M. (2006). PID tuning using extremum seeking. IEEE Control Systems Magazine, 26: 70–79.

[20] Simon D. (2006). Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches. Hoboken, NJ: Wiley.

[21] Chen BM., Lee TH. and Peng K (2006). Hard Disk Drive Servo Systems. New York, NY: Springer.

[22] Longman RW. (2000). Iterative learning control and repetitive control for engineering practice. Int. Journal of Control, 73: 930–954.

[23] Franklin GF., Powell JD. and Workman ML (1997). Digital Control of Dynamic Systems. Upper Saddle River, NJ: Prentice Hall.

[24] Ogata K. (1995). Discrete-Time Control Systems. Upper Saddle River, NJ: Prentice Hall.

[25] Astrom KJ. and Wittenmark B (1996). Computer-Controlled Systems: Theory and Design. Upper Saddle River, NJ: Prentice Hall.