6.5 Self-tuning PID Control

■ Introduction

In practical engineering, the most widely used control method is proportional, integral and derivative control, referred as PID control.

- ✓ In the late 1930s, the addition of differential control marked that PID control became a standard structure.
- ✓ In its development process, it has become one of the main technologies of industrial control for its simple structure, good stability, reliable operation and convenient tuning.
- ✓ In PID control, a crucial problem is the tuning of PID parameters.
- ✓ The quality of parameter tuning not only affects the control performance, but also affects the stability and robustness of a control system.

1

What is self-tuning PID control?

- ✓ The typical PID parameter tuning method is to determine the PID parameters according to a certain tuning principle on the basis of obtaining the mathematical model of a controlled plant.
- ✓ The self-tuning PID control is the product of the combination of the self-tuning control idea and the conventional PID control, which absorbs the advantages of both.
- ✓ It not only requires less parameters to be adjusted, but also can modify those parameters online according to the change of the plant characteristics.
- ✓ Thus, this enhances the adaptation ability of a PID controller.

2

■ Conventional PID control

Only an incremental PID control algorithm is introduced here

$$\Delta u(k) = K_p(e(k) - e(k-1)) + K_i e(k) + K_d(e(k) - 2e(k-1) + e(k-2)) \qquad (6.5.1)$$

where K_p, K_i and K_d are PID tuning parameters, and

$$\Delta u(k) = u(k) - u(k-1) \qquad e(k) = y_r(k) - y(k)$$

$u(k)$, $y_r(k)$ and $y(k)$ are the control input, desired output and actual output, respectively.

Eq (6.5.1) can be expressed as

$$\Delta u(k) = g_0 e(k) + g_1 e(k-1) + g_2 e(k-2) \qquad (6.5.2)$$

where $g_i (i = 0, 1, 2)$ is an adjustable parameter.

The discrete-time transfer function of the controller from eq (6.5.2) is

$$\frac{u(k)}{e(k)} = \frac{g_0 + g_1 q^{-1} + g_2 q^{-2}}{1 - q^{-1}}$$

3

The previous equation can be expressed as

$$F(q^{-1})u(k) = R(q^{-1})y_r(k+d) - G(q^{-1})y(k) \qquad (6.5.3)$$

where

$$F(q^{-1}) = 1 - q^{-1}$$

$$R(q^{-1}) = G(q^{-1}) = g_0 + g_1 q^{-1} + g_2 q^{-2}$$

It can be seen that eq (6.5.3) and pole assignment control law (6.4.2) have the same form.

♦ **Example 6.5.1**

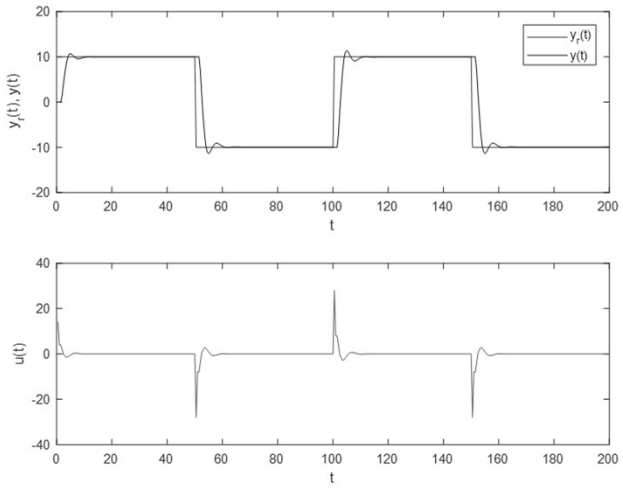
Consider the controlled plant $G(s) = \frac{1}{s(s+1)} e^{-s}$

Take the sampling period $T_s(k) = 0.5s$, and the expected output $y_r(k)$ is a square wave signal with amplitude of 10.

Use the incremental PID control algorithm, where $K_p = 0.4$, $K_i = 0$ and $K_d = 1$.

4

♦ **Simulation results**



The conventional PID algorithm has good control effects for the plant in this example, but the PID controller parameters are not easy to tuning.

5

♦ **Simulation codes**

example651.m

6

■ Self-tuning PID control

The design idea of a self-tuning PID controller

- takes eq (6.5.3) as the basic form of the controller,
- introduces a recursive algorithm to estimate the parameters of the plant, and
- designs the controller parameters using the pole assignment method.

Consider the second order plant

$$A(q^{-1})y(k) = q^{-d}B(q^{-1})u(k) + \xi(k) \qquad (6.5.4)$$

where $u(k)$ and $y(k)$ represent the input and output of the system, $\xi(k)$ is an external disturbance, $\mathrm{d} \geq 1$ is a pure delay, and

$$A(q^{-1}) = 1 + a_1 q^{-1} + a_2 q^{-2}$$
$$B(q^{-1}) = b_0 + b_1 q^{-1} + b_2 q^{-2} + \cdots + b_{n_b} q^{-n_b}$$

7

To eliminate a constant interference, the controller must have an integral function.

From eq (6.5.3), a generic PID controller for practical applications can be expressed as

$$F_1(q^{-1})u(k) = R(q^{-1})y_r(k+d) - G(q^{-1})y(k) \qquad (6.5.5)$$

where

$$F_1(q^{-1}) = (1 - q^{-1})F(q^{-1})$$
$$F(q^{-1}) = f_0 + f_1 q^{-1} + \cdots + f_{n_f} q^{-n_f}$$
$$G(q^{-1}) = g_0 + g_1 q^{-1} + g_2 q^{-2}$$

$R(q^{-1})$ can be chosen as

- ✓ $R(q^{-1}) = G(q^{-1}) = g_0 + g_1 q^{-1} + g_2 q^{-2}$
- ✓ $R(q^{-1}) = \mathrm{G}(1) = g_0 + g_1 + g_2$

Substituting eq (6.5.5) into eq (6.5.4) yields

$$y(k) = \frac{R(q^{-1})B(q^{-1})y_r(k) + F_1(q^{-1})\xi(k)}{A(q^{-1})F_1(q^{-1}) + q^{-d}B(q^{-1})G(q^{-1})}$$

8

Let the closed-loop characteristic polynomial be the denominator polynomial of the desired transfer function, *i.e.*,

$$A(q^{-1})F_1(q^{-1}) + q^{-d}B(q^{-1})G(q^{-1}) = A_m(q^{-1})$$

$$\Delta A(q^{-1})F(q^{-1}) + q^{-d}B(q^{-1})G(q^{-1}) = A_m(q^{-1}) \quad (6.5.6)$$

The selection of the desired transfer function $A_m(q^{-1})$ still needs to meet the corresponding compatibility conditions in "Algorithm 6.4.1".

To ensure that formula (6.5.6) has a unique solution, $\Delta A(q^{-1})F(q^{-1})$ and $q^{-d}B(q^{-1})G(q^{-1})$ have the same order, and the order of each polynomial is required to meet the following relationship:

$$\deg A_m \le \deg B + d + 2$$

$$\deg F = \deg B + d - 1$$

9

When the parameters of $A(q^{-1})$ and $B(q^{-1})$ are unknown, the self-tuning control algorithm needs to be adopted.

♦ **Algorithm 6.5.2** (Self-tuning PID control)

Given model structure n_a, n_b and d, desired closed-loop characteristic polynomial $A_m(q^{-1})$.

Step 1: Set initial value $\hat{\theta}(0)$, $\mathrm{P}(0)$ and forgetting factor λ, and input initial data.

Step 2: Sample the current actual output $y(k)$ and expected output $y_r(k)$.

Step 3: Use the forgetting recursive least squares method to estimate the parameters of the controlled plant online and in real time $\hat{\theta}(\mathrm{k})$ (*i.e.*, $\hat{A}(q^{-1})$, $\hat{B}(q^{-1})$).

Step 4: Solve $F(q^{-1})$ and $G(q^{-1})$ in equation (6.5.6), and then take $F_1 = F\Delta$ and $R = G(1)$.

Step 5: Calculate and implement the control quantity $u(k)$ from formula (6.5.5).

Step 6: Return to Step 2 (k → k+1) and continue the loop.

10

♦ **Example** 6.5.2

The controlled plant:

$$y(k) - 1.6065y(k-1) + 0.6065y(k-2) = 0.1065u(k-3) + 0.0902u(k-4) + \xi(k)$$

where $\xi(k) = 2$ is a constant disturbance.

The desired transfer function polynomial is

$$A_m(q^{-1}) = 1 - 1.3205q^{-1} + 0.4966q^{-2}$$

Take the expected output $y_r(k)$ as a square wave signal with amplitude of 10.
The simulation is conducted in two cases.

Case 1: When the plant parameters are known (pole assignment PID control)

From the system difference equation

$$B(q^{-1}) = 0.1065 + 0.0902q^{-1}$$

$$\Delta A(q^{-1}) = 1 - 2.6065q^{-1} + 2.213q^{-2} - 0.6065q^{-3}$$

$$d = 3$$

11

Using the MATLAB function 'diophantine.m' for the solution of eq (6.5.6) gives

$$G(q^{-1}) = 11.2246 - 15.8984q^{-1} + 5.5691q^{-2}$$

$$F(q^{-1}) = 1 + 1.286q^{-1} + 1.6356q^{-2} + 0.8\,282q^{-3}$$

$$R(q^{-1}) = G(1) = 0.8953$$

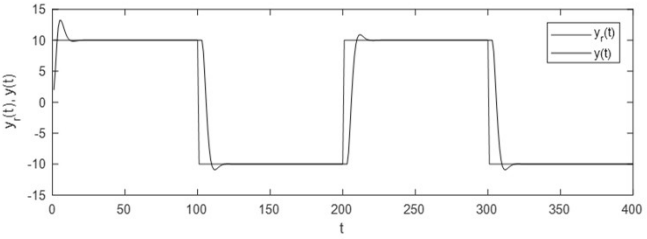
$$F_1(q^{-1}) = \Delta F(q^{-1}) = 1 + 0.286q^{-1} + 0.3496q^{-2} - 0.8073q^{-3} - 0.8282q^{-4}$$

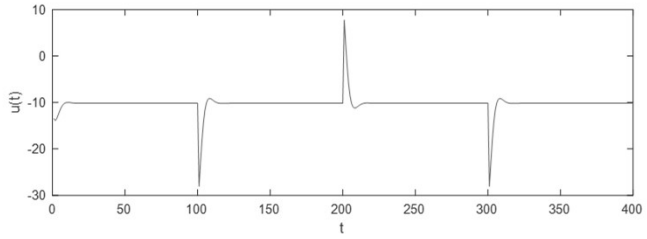
Then the PID control law of pole assignment from equation (6.5.5) is

$$u(k) = 0.8953y_r(k) - \left(11.2246 - 15.8984q^{-1} + 5.5691q^{-2}\right)y(k) - (0.286q^{-1} + 0.3496q^{-2} - 0.8073q^{-3} - 0.8282q^{-4})u(k)$$

12

The results:





♦ **Simulation codes**

example652.m

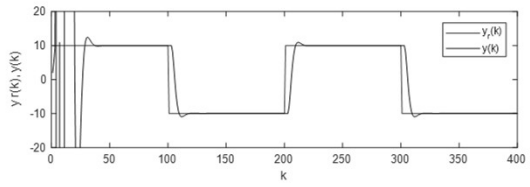
13

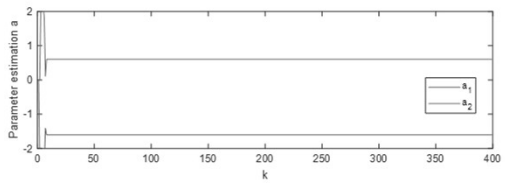
Case 2: When the plant parameters are unknown (indirect algorithm of self-tuning PID control)

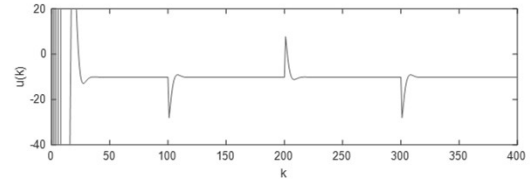
Take the initial value $\hat{\theta}(0) = 0.001$, $\mathrm{P}(0) = 10^6 I$, forgetting factor $\lambda = 1$.

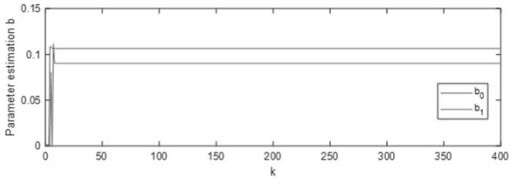
The desired output $y_r(k)$ is a square wave signal with amplitude of 10.

The self-tuning PID control indirect algorithm is adopted.









14

♦ **Simulation codes**

example653.m

15

6.6 Generalized Predictive Control

■ Introduction

Compared with the minimum variance control, the generalized minimum variance control has improved its performance, can control the non-minimum phase system, and has certain constraints on the control effect.

However, in practical engineering applications, there are still some difficulties in selecting appropriate weighted polynomials.

To solve the adaptive control problem of non-minimum phase systems and complex industrial processes, more advanced control algorithms need to be studied.

In the late 1970s and the early and middle 1980s, model predictive control (MPC) based on multi-step prediction and rolling optimization appeared, which provides a new direction for better solving the adaptive control problem of complex industrial plants.

16

♦ **Model predictive control**

- ✓ In 1974, dynamic matrix control (DMC) was applied to the production units of Shell Oil Company in the United States, and was publicly published at the American Chemical Annual Conference by C.R. Culter in 1980. It is a predictive control based on the step response model.
- ✓ In 1978, on the basis of a system impulse response, J. Richlet et al. proposed model predictive heuristic control (MPHC), and introduced its application effect in industrial process control.
- ✓ In 1982, R. Rouhani et al. proposed model algorithmic control (MAC) , based on impulse response.

Those predictive control algorithms take the finite impulse response or finite step response of a process as the model, and carry out online optimal control of the process in the finite time domain (i.e., rolling optimization) in each control cycle by means of rolling transition.

17

Advantages of model predictive control

- ✓ The above predictive control has low requirements on the model of the process, simple algorithm, and easy to implement.
- ✓ At the same time, it constantly uses the measurement information to carry out feedback correction during the optimization process, which overcomes the influence of uncertainties to a certain extent.
- ✓ It shows good control performance in complex industrial process control.

18

♦ **Adaptive control issues**

Generally speaking, a successful adaptive control algorithm should be applicable to the following cases:

- ✓ Non-minimum phase systems because the minimum phase continuous system will become a non-minimum phase system when the sampling frequency is high to a certain degree.
- ✓ Time-varying systems with unknown delay.
- ✓ Open-loop unstable systems or systems which are prone to unstable poles.
- ✓ Systems with unknown model order.

For those systems above, most adaptive control algorithms cannot be applied.

- ✓ The key point is that most adaptive control algorithms need to establish an accurate mathematical model, but the actual industrial process is often difficult to establish an accurate model.
- ✓ The time delay of a system is often difficult to determine accurately.

19

Adaptive predictive control is needed.

- ✓ For the minimum variance controller, if the time delay estimation is not accurate, the control accuracy will be greatly reduced.
- ✓ It is natural to seek a control algorithm with low model requirements and good adaptation ability.
- ✓ Generalized predictive control (GPC) is such an algorithm. It is a predictive control algorithm developed with the research of adaptive control.
- ✓ The commonness of various predictive control algorithms can be summarized as three elements: predictive model, rolling optimization and feedback correction.
- ✓ Those three elements are also the basic characteristics that distinguish predictive control from other control methods, and are also the technical keys to the success of predictive control in practical engineering applications.

20

■ Predictive control

♦ Prediction model

- ✓ The model used for predictive control is called the prediction model.
- ✓ The requirement of predictive control on the model is different from other traditional control methods.
- ✓ It emphasizes the function of the model rather than the structure of the model.
- ✓ As long as the model can predict the future output behavior of a system using the known data information in the past, it can be used as a predictive model.

Therefore, the prediction models can be

- ─ traditional models, such as state-space equations and transfer functions
- ─ impulse response models or step response models that are easily obtained in actual industrial processes
- ─ CARMA models that are easy to identify online and can describe unstable systems

21

Predictive control breaks the strict requirements for model structure in traditional control, and focuses more on building models in the most convenient way, based on information and functional requirements.

This is one of the reasons why it is superior to other control algorithms, and also the premise that it can be widely used in industrial practice.

22

♦ **Rolling optimization**

Predictive control is also an optimal control algorithm, which determines the future control effect through the optimization of a certain performance index.

This performance index relates to the future behavior of a system.

- ✓ For example, the minimum variance of the tracking error of a process is desirable.
- ✓ For another example, it is required to control the energy to the minimum while maintaining the output within a given range.

A common performance index is as follows:

$$J(k) = \mathcal{E}\left\{\sum_{j=n_1}^{n_y}(y(k+j) - y_r(k+j))^2 + \sum_{j=1}^{n_u}\left(\gamma_j \Delta u(k+j-1)\right)^2\right\} \qquad (6.6.1)$$

where $y(k+j)$ and $y_r(k+j)$ are the actual output and desired output of a system at the future time $k+j$, n_1 is the minimum output length, n_y is the maximum output length (also known as the prediction degree), and n_u is the control length, γ_j is the control weighting coefficient, which is generally taken as a constant value.

23

Features of the performance index

- ✓ The optimization in predictive control is different from the conventional discrete optimal control algorithm.
- ✓ This is mainly reflected in the fact that the optimization objective in predictive control is not a static global optimization objective, but a rolling finite-time range optimization strategy.
- ✓ The optimization is not done offline at one time, but online repeatedly.
- ✓ At each sampling moment, the optimization performance index only involves a limited time range in the future, and at the next sampling moment, this optimization time range will move forward at the same time.

Therefore, predictive control has an optimization performance index at each time (relative to that time), so as to update the future control sequence (also known as control projection) according to the latest measured data, that is, to achieve rolling optimization.

24

Discussions on rolling optimization

- ✓ This optimization objective with finite-time range can only make it obtain the global suboptimal solution because this optimization process is repeated online.
- ✓ The optimization process is always based on the latest information obtained from the actual process, which can more timely correct the uncertainties caused by model mismatch, time-varying and interference.
- ✓ So, it can obtain a robust result and maintain the actual optimal control.

25

♦ **Feedback correction**

Why is feedback correction needed?

- ✓ The prediction model is only a rough description of the dynamic characteristics of a process (it is impossible to obtain an accurate mathematical model in practice).
- ✓ Due to uncertainty factors, such as nonlinearity, time-varying, model mismatch and interference in the actual system, the prediction based on an invariant model cannot exactly match the actual situation.
- ✓ It requires additional prediction methods to compensate for the deficiencies of the model prediction, or online correction of the basic model.
- ✓ Only on the basis of this feedback correction can rolling optimization show its advantages.
- ✓ Therefore, predictive control will determine a group of future control actions $[u(k), u(k+1), \ldots, u(k+n_u-1)]$ by optimizing the performance index in each control cycle.

26

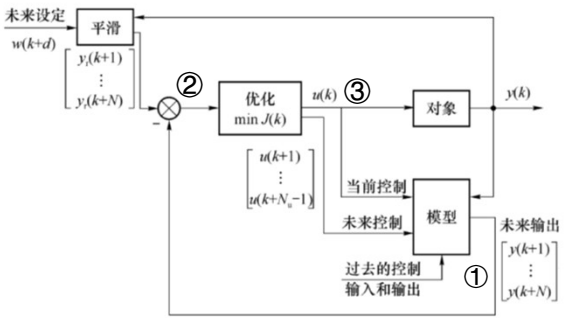
How to implement feedback correction?

- ✓ To prevent model mismatch or environmental interference from causing control deviation from an ideal state, these control actions are not implemented one by one, but only the control action $u(k)$ at the current time.
- ✓ At the next sampling time, the actual output of the process is detected first, and the model-based prediction is modified using this real-time information, and then a new optimization is carried out.
- ✓ The forms of feedback correction are various, which can predict and compensate for the future errors on the basis of keeping the prediction model unchanged, or directly modify the prediction model according to the principle of online identification.
- ✓ No matter what kind of correction form is adopted, predictive control builds optimization on the actual output of the system and tries to make more accurate prediction of the future dynamic behavior of the system.
- ✓ Therefore, the optimization in predictive control is not only based on the model, but also uses the feedback information, thus forming a closed-loop optimization.

27

♦ **The generation process of predictive control**

① At each "current sampling time k", based on a certain prediction model of a process, the output sequence of the system in a certain period of time in the future is predicted (multi-step prediction) by using the past, current and future control inputs, and the past and current system outputs, as shown in $[y(k+1), y(k+2), \ldots, y(k+N)]$ below.



The optimal prediction of the aforementioned minimum variance control and generalized minimum variance control does not include the future control input, and is a single-step prediction.

28

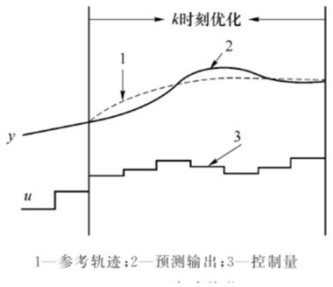
② The future control sequence $[u(k), u(k+1), \dots, u(k+n_u-1)]$ is obtained by minimizing the objective function J(k) of eq (6.6.1), which makes the future output prediction sequence $y(k+j)$ reach the set value along a reference track $y_r(k+j)$, as shown below.

In predictive control, to make the output $y(k)$ smoothly transition to the set value w at a certain response speed, the reference trajectory is usually taken as the following first-order model (first-order smoothing) :

$$y_r(k) = y(k)$$

$$y_r(k+j) = \alpha y_r(k+j-1) + (1-\alpha)w, \qquad j = 1,2,\dots,N$$

where $\alpha \in [0,1]$ is the output softening coefficient.



1—参考轨迹；2—预测输出；3—控制量

- ✓ This can make $y_r(k+j)$ smoothly transition to the set value w.
- ✓ When slow transition is required, you can select α close to 1. At the same time, it can also "soften" the future control sequence.

29

③ Although predictive control can obtain n_u future control actions $[u(k), u(k+1), \dots, u(k+n_u-1)]$ by optimizing a performance index in each control cycle, at the current time k, only the control action $u(k)$ at the current time is implemented for the process.

④ Prepare for the next sampling. After the next sampling, the above process needs to be repeated to update the future control sequence according to the latest measured data, that is, to achieve feedback correction and rolling optimization.

30

■ Generalized predictive control

Generalized predictive control (GPC) is an important adaptive control algorithm proposed by D.W. Clarke et al. in 1987.

On the basis of maintaining the online identification, output prediction, minimum output variance control and the minimum variance self-tuning control, it absorbs the rolling optimization strategy in DMC and MAC, and has the performance of adaptive control and predictive control.

GPC is based on a parameter model and introduces unequal output prediction length and control length. The system design is flexible and convenient.

It has the characteristics of prediction model, rolling optimization and online feedback correction, and shows excellent control performance and robustness.

31

♦ j-step optimal prediction

GPC adopts the following CARIMA (Controlled Auto-Regressive Integrated Moving-Average) model:

$$A(q^{-1})y(k) = q^{-1}B(q^{-1})u(k) + \frac{C(q^{-1})}{\Delta}\xi(k) \qquad (6.6.2)$$

where $y(k)$, $u(k)$ and $\xi(k)$ are the output, input and white noise of the system, $\Delta = 1 - q^{-1}$ is a difference operator, and

$$A(q^{-1}) = 1 + a_1 q^{-1} + \ \dots + a_{n_a}\ q^{-n_a}$$

$$B(q^{-1}) = b_0 + b_1 q^{-1} + \ \dots + b_{n_b} q^{-n_b}$$

$$C(q^{-1}) = 1 + c_1 q^{-1} + \ \dots + c_{n_c} q^{-n_c}$$

32

Equation (6.6.2) can be simplified to

$$\bar{A}(q^{-1})y(k) = B(q^{-1})\Delta u(k-1) + C(q^{-1})\xi(k) \qquad (6.6.3)$$

where

$$\bar{A}(q^{-1}) = \Delta A(q^{-1}) = (1-q^{-1})A(q^{-1}) = 1 + \bar{a}_1 q^{-1} + \ \dots + \bar{a}_{n_{\bar{a}}}\ q^{-\bar{n}_{\bar{a}}}$$

$$\bar{n}_{\bar{a}} = n_a + 1, \bar{a}_0 = 1, \bar{a}_{n_{\bar{a}}} = a_{n_a}, \bar{a}_i = a_i - a_{i-1}, 1 \le i \le n_a$$

In GPC design, similar to the minimum variance control, it is necessary to predict the system output in advance and calculate the required control action according to the obtained optimal prediction.

The difference is that the latter is single-step output prediction, while GPC requires multi-step output prediction.

33

♦ **Theorem 6.6.1** (j-step optimal prediction of CARIMA model)

For the controlled plant (6.6.1), let the output prediction error at time $k+j$ be

$$\tilde{y}(k+j|k) = y(k+j) - \hat{y}\,(k+j|k)$$

Then, the variance of the prediction error: $J(k) = \mathcal{E}\{\tilde{y}^2(k+j|k)\}$

The j-step optimal prediction $y^*(k+j|k)$ is given by the following difference equation

$$C(q^{-1})y^*(k+j|k) = G_j(q^{-1})y(k) + F_j(q^{-1})\Delta u(k) \qquad (6.6.4)$$

where $F_j(q^{-1})$ and $G_j(q^{-1})$ satisfy the following Diophantine equations:

$$C(q^{-1}) = A(q^{-1})E_j(q^{-1}) + q^{-j}G_j(q^{-1}) \qquad F_j(q^{-1}) = B(q^{-1})E_j(q^{-1}) \qquad (6.6.5)$$

$$F_j(q^{-1}) = f_{j,0} + f_{j,1}q^{-1} + \ \dots + f_{j,n_{fj}}q^{-n_{fj}} \qquad (n_{fj} = n_b + j\ - 1)$$
$$G_j(q^{-1}) = g_{j,0} + g_{j,1}q^{-1} + \ \dots + g_{j,n_{gj}}q^{-n_{gj}} \qquad \left(n_{gj} = n_a - 1\right)$$
$$E_j(q^{-1}) = 1 + e_{j,1}q^{-1} + \ \dots + e_{j,n_{ej}}q^{-n_{ej}} \qquad (n_{ej} = j\ - 1)$$

The optimal prediction error is $\tilde{y}^*(k+j|k) = E_j(q^{-1})\xi(k+j)$

34

Proof: (The proof process is similar to Theorem 6.2.1) It can be obtained from eq (6.2.3)

$$y(k+j) = E_j(q^{-1})\xi(k+j) + \frac{B(q^{-1})}{\bar{A}(q^{-1})}\Delta u(k+j-1) + \frac{G_j(q^{-1})}{\bar{A}(q^{-1})}\xi(k) \qquad (6.6.6)$$

It can be obtained from eq (6.6.3)

$$\xi(k) = \frac{\bar{A}(q^{-1})}{C(q^{-1})}y(k) - \frac{B(q^{-1})}{C(q^{-1})}\Delta u(k-1)$$

Substituting the above into (6.6.6) and then using (6.6.5) yield

$$y(k+j) = E_j(q^{-1})\xi(k+j) + \frac{F_j(q^{-1})}{C(q^{-1})}\Delta u(k+j-1) + \frac{G_j(q^{-1})}{C(q^{-1})}y(k) \qquad (6.6.7)$$

35

Substitute (6.6.7) into performance index $J(k)$ to obtain

$$\begin{aligned} J(k) &= \mathcal{E}\{(y(k+j) - \hat{y}\,(k+j|k))^2\} \\ &= \mathcal{E}\left\{\left(E_j(q^{-1})\xi(k+j) + \frac{F_j(q^{-1})}{C(q^{-1})}\Delta u(k+j-1) + \frac{G_j(q^{-1})}{C(q^{-1})}y(k) - \hat{y}\,(k+j|k)\right)^2\right\} \\ &= \mathcal{E}\left\{\left(E_j(q^{-1})\xi(k+j)\right)^2\right\} + \mathcal{E}\left\{\left(\frac{F_j(q^{-1})}{C(q^{-1})}\Delta u(k+j-1) + \frac{G_j(q^{-1})}{C(q^{-1})}y(k) - \hat{y}\,(k+j|k)\right)^2\right\} \end{aligned}$$

To make $J(k)$ be minimum, $\hat{y}\,(k+j|k)$ should be

$$\hat{y}\,(k+j|k) = \frac{F_j(q^{-1})}{C(q^{-1})}\Delta u(k+j-1) + \frac{G_j(q^{-1})}{C(q^{-1})}y(k) = y^*(k+j|k)$$

That is, eq (6.6.4) holds, and at this time

$$J_{min} = \mathcal{E}\left\{\left(E_j(q^{-1})\xi(k+j)\right)^2\right\}$$

The optimal prediction error is $\tilde{y}^*(k+j|k) = E_j(q^{-1})\xi(k+j)$

36

Equation (6.6.4) is the optimal output prediction equation;
Equation (6.6.5) is called the multi-step Diophantine equation;
Equation (6.6.7) is called the output prediction model.

For the GPC algorithm, the multi-step Diophantine equation needs to be solved online, and the recursive algorithm can be used. See Section 6.1 for details.

The performance index function (6.6.2) can be expressed in a matrix form, namely

$$J(k) = \mathcal{E}\{(Y - Y_r)^T(Y - Y_r) + (\Delta U)^T\Gamma(\Delta U)\} \tag{6.6.8}$$

$$Y = \left[y(k+n_1), y(k+n_1+1), \ldots, y(k+n_y)\right]^T$$

$$Y_r = \left[y_r(k+n_1), yr(k+n_1+1), \ldots, yr(k+n_y)\right]^T$$

$$\Delta U = [\Delta u(k), \Delta u(k+1), \ldots, \Delta u(k+n_u-1)]^T$$

$$\Gamma = diag\left(\gamma_1, \gamma_2, \ldots, \gamma_{n_u}\right)$$

37

♦ **Generalized predictive control for $C(q^{-1}) = 1$**

For $C(q^{-1}) = 1$, the plant prediction model (6.6.7) will be simplified as

$$y(k+j) = F_j(q^{-1})\Delta u(k+j-1) + G_j(q^{-1})y(k) + E_j(q^{-1})\xi(k+j)$$

For $n_1 = 1$, $n_y = n_u =$ N, it is obtained from the above and eq (6.6.5) that

$$\begin{aligned} y(k+1) &= F_1(q^{-1})\Delta u(k) + G_1(q^{-1})y(k) + E_1(q^{-1})\xi(k+1) \\ &= f_{1,0}\Delta u(k) + f_{1,1}\Delta u(k-1) + \ \ldots\ + f_{1,n_b}\Delta u(k-n_b) + G_1(q^{-1})y(k) + \xi(k+1) \end{aligned}$$

$$\begin{aligned} y(k+2) &= F_2(q^{-1})\Delta u(k+1) + G_2(q^{-1})y(k) + E_2(q^{-1})\xi(k+2) \\ &= f_{2,0}\Delta u(k+1) + f_{2,1}\Delta u(k) + \ \ldots\ + f_{2,n_b+1}\Delta u(k-n_b) + G_2(q^{-1})y(k) + \xi(k+1) + e_{2,1}\xi(k+2) \end{aligned}$$

$$\begin{aligned} y(k+N) &= F_N(q^{-1})\Delta u(k+N-1) + G_N(q^{-1})y(k) + E_N(q^{-1})\xi(k+N) \\ &= f_{N,0}\Delta u(k+N-1) + f_{N,1}\Delta u(k+N) + \ \ldots\ + f_{N,n_b+N-1}\Delta u(k-n_b) + G_N(q^{-1})y(k) \\ &\quad +\xi(k+N) \ + e_{N,1}\xi(k+N-1) + \cdots + e_{N,N-1}\xi(k+1) \end{aligned}$$

38

Write the above equations in matrix form as

$$\begin{aligned} Y &= \mathcal{F}_1 \Delta U + \mathcal{F}_2 \Delta U(k-j) + \mathcal{G} Y(k) + E\xi \\ &= \mathcal{F}_1 \Delta U + Y_1 + E\xi \end{aligned} \tag{6.6.9}$$

where $Y_1 = \mathcal{F}_2 \Delta U(k-j) + \mathcal{G} Y(k)$ is the output prediction response

$Y = [y(k+1), y(k+2), \ldots, y(k+N)]^T$ the future output

$\Delta U = [\Delta u(k), \Delta u(k+1), \ldots, \Delta U(k+N-1)]^T$ the current and future control increment vector

$\Delta U(k-j) = [\Delta u(k-1), \Delta u(k-2), \ldots, \Delta U(k-n_b)]^T$ the control increment vector in the past

$Y(k) = [y(k), y(k-1), \ldots, y(k-n_a)]^T$ the current and past actual output

$\xi = [\xi(k+1), \xi(k+2), \ldots, \xi(k+N)]^T$ the future white noise vector

$$\mathcal{F}_1 = \begin{bmatrix} f_{1,0} & 0 & \cdots & 0 \\ f_{2,1} & f_{2,0} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ f_{N,N-1} & f_{N,N-2} & \cdots & f_{N,0} \end{bmatrix} \in \mathcal{R}^{N \times N} \qquad \mathcal{F}_2 = \begin{bmatrix} f_{1,1} & f_{1,2} & \cdots & f_{1,n_b} \\ f_{2,2} & f_{2,3} & \cdots & f_{2,n_b+1} \\ \vdots & \vdots & \ddots & \vdots \\ f_{N,N} & f_{N,N+1} & \cdots & f_{N,n_b+N-1} \end{bmatrix} \in \mathcal{R}^{N \times n_b}$$

$$\mathcal{G} = \begin{bmatrix} g_{1,0} & g_{1,1} & \cdots & g_{1,n_a} \\ g_{2,0} & g_{2,1} & \cdots & g_{2,n_a} \\ \vdots & \vdots & \ddots & \vdots \\ g_{N,0} & g_{N,1} & \cdots & g_{N,n_a} \end{bmatrix} \in \mathcal{R}^{N \times (n_a+1)} \qquad E = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ e_{2,1} & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ e_{N,N-1} & e_{N,N-2} & \cdots & 1 \end{bmatrix} \in \mathcal{R}^{N \times N}$$

39

It can be seen that eq (6.6.9) decomposes the predicted output sequence at the future time into two parts, one component $\mathcal{F}_1 \Delta U$ depends on the current and future control input, and the other component Y_1 depends on the past control input and process output.

Substituting (6.6.9) into (6.6.8) leads to

$$J(k) = \mathcal{E}\{(\mathcal{F}_1 \Delta U + Y_1 + E\xi - Y_r)^T (\mathcal{F}_1 \Delta U + Y_1 + E\xi - Y_r) + (\Delta U)^T \Gamma (\Delta U)\}$$

By $\frac{\mathrm{dJ}}{\mathrm{d}\Delta U} = 0$, GPC control increment vector is

$$\Delta U(k) = \left(\mathcal{F}_1^T \mathcal{F}_1 + \Gamma\right)^{-1} \mathcal{F}_1^T (Y_r - \mathcal{F}_2 \Delta U(k-j) - \mathcal{G} Y(k)) \tag{6.6.10}$$

Then, the control input $u(k)$ at the current moment is

$$\begin{aligned} u(k) &= u(k-1) + \Delta u(k) \\ &= u(k-1) + [1, 0, \ldots, 0] \left(\left(\mathcal{F}_1^T \mathcal{F}_1 + \Gamma\right)^{-1} \mathcal{F}_1^T (Y_r - \mathcal{F}_2 \Delta U(k-j) - \mathcal{G} Y(k)) \right) \end{aligned} \tag{6.6.11}$$

40

If the process parameters are unknown, it is necessary to estimate the parameters online and implement the self-tuning GPC algorithm.

$$\begin{aligned}\Delta y(k) &= \left(1 - A(q^{-1})\right)\Delta y(k) + B(q^{-1})\Delta u(k-1) + \xi(k)\\ &= \varphi^T(k)\theta + \xi(k)\end{aligned}$$

$$\varphi(k) = [-\Delta y(k), \cdots, -\Delta y(k-n_a), \Delta u(k-1), \cdots, \Delta u(k-n_b-1),]^T$$

$$\theta = \begin{bmatrix} a_1 & \cdots & a_{n_a} & b_0 & \cdots & b_{n_b} \end{bmatrix}^T$$

The recursive least squares method with forgetting factor is used to estimate the process parameters, namely

$$\hat{\theta}(k) = \hat{\theta}(k-1) + K(k)\left(\Delta y(k) - \hat{\varphi}^T(k-1)\hat{\theta}(k-1)\right)$$

$$K(k) = \frac{P(k-1)\hat{\varphi}(k-1)}{\lambda + \hat{\varphi}^T(k-1)P(k-1)\hat{\varphi}(k-1)}$$

$$P(k) = \frac{1}{\lambda}\left(I - K(k)\hat{\varphi}^T(k-1)\right)P(k-1)$$

41

♦ **Algorithm** 6.6.2 [Clarke GPC when $C(q^{-1}) = 1$] (Self-tuning GPC algorithm)

Given n_a, n_b and d.

Step 1: Set initial value $\hat{\theta}(0)$ and $P(0)$, initial input data, and control parameters, such as n_1, n_y, n_u, and control weighting matrix Γ、Output softening coefficient α、forgetting factor λ, etc.

Step 2: Sample the current actual output $y(k)$ and the reference input $y_r(k+j)$.

Step 3: Use the recursive least squares method to estimate the parameters $\hat{\theta}(k)$ of the controlled plant online in real time, namely $\hat{A}(q^{-1})$, $\hat{B}(q^{-1})$.

Step 4: Solve Diophantine equation to obtain polynomials $E_j(q^{-1})$, $G_j(q^{-1})$ and $F_j(q^{-1})$.

Step 5: Construct vector $Y_r, \Delta U(k), Y(k)$ and matrices $\mathcal{G}(q^{-1})$, $\mathcal{F}_1(q^{-1})$, $\mathcal{F}_2(q^{-1})$.

Step 6: Use formula (6.6.11) to calculate and implement $u(k)$.

Step 7: Return to Step 2 (k → k+1) and continue the loop.

42

♦ **Discussions on the selection of control parameters $\mathbf{n_1}$, $\mathbf{n_2}$ and $\mathbf{n_u}$**

① Selection of n_1: When the pure delay d of the plant is known, $n_1 = d$ can be selected, and the dimensions of matrices $\mathcal{G}(q^{-1})$, $\mathcal{F}_1(q^{-1})$, $\mathcal{F}_2(q^{-1})$ will be reduced.

Thus, the amount of calculation can be reduced.

However, in the case of self-tuning control, d is often time-varying or unknown. At this time, only $n_1 = 1$ can be taken.

The simulation results show that n_1 can be selected in a large range without affecting the stability of the system.

② Selection of n_{y}: Since the performance index function contains future control, the output length n_{y} should include all response segments that are greatly affected by the current control. So, n_{y} should be at least greater than the order n_{b} of polynomial $\mathrm{B}(q^{-1})$.

Physically, n_{y} should be larger, for example, close to the rise time of the process.

43

③ Selection of n_{u}: For a stable plant, select $n_{\mathrm{u}} = 1$, then $F_1^T F_1 + \Gamma = 1$ degenerates to scalar operation.

For an unstable or strongly oscillatory plant, n_{u} can be taken as the sum of the number of unstable poles and the number of underdamped poles of the plant.

The simulation shows that n_{u} should be selected to achieve satisfactory performance for the above controlled plants.

Therefore, in the actual GPC algorithm, $n_1 = 1$, $n_{\mathrm{y}} = N$, $n_{\mathrm{u}} \leq n_{\mathrm{y}}$ are often taken, and

$$\Delta u(k + j - 1) = 0, \qquad \text{for } \mathrm{j} > n_{\mathrm{u}}$$

44

♦ **Example** 6.6.1

The controlled plant is the following unstable non-minimum phase plant:

$$y(k) - 2y(k-1) + 1.1y(k-2) = u(k-4) + 2u(k-5) + \xi(k)/\Delta$$

where $\xi(k)$ is a white noise with variance of 0.01.

Case 1: the plant parameters are known (GPC algorithm).

From the system difference equation,

$$y(k) - 3y(k-1) + 3.1y(k-2) - 1.1y(k-3) = \Delta u(k-4) + 2\Delta u(k-5) + \xi(k)$$

$$\bar{A}(q^{-1}) = 1 - 3q^{-1} + 3.1q^{-2} - 1.1q^{-3}$$

$$B(q^{-1}) = q^{-3} + 2q^{-4}$$

For Diophantine equation (6.6.5), the solution is

$$\mathcal{F}_1 = \begin{bmatrix} 1 & 0 \\ 5 & 1 \\ 11.9 & 5 \\ 21.3 & 11.9 \\ 32.51 & 21.3 \end{bmatrix} \quad \mathcal{F}_2 = \begin{bmatrix} 5 & 11.9 & 21.3 & 19 \\ 11.9 & 21.3 & 32.51 & 27.02 \\ 21.3 & 32.51 & 44.59 & 35.14 \\ 32.51 & 44.59 & 56.419 & 42.558 \\ 44.59 & 56.419 & 66.789 & 48.462 \end{bmatrix} \quad \mathcal{G} = \begin{bmatrix} 13.51 & -22.96 & 10.45 \\ 17.57 & -31.431 & 14.861 \\ 21.279 & -39.606 & 19.327 \\ 24.231 & -46.6379 & 23.4069 \\ 26.0551 & -51.7092 & 26.6541 \end{bmatrix}$$

45

Take the control weighting matrix Γ as a unit matrix,

$$[1, 0, \dots, 0]\left(\mathcal{F}_1^T\mathcal{F}_1 + \Gamma\right)^{-1}\mathcal{F}_1^T = [0.0259, 0.0876, 0.0981, 0.0513, -0.0538]$$

Then the GPC control law from formula (6.6.11) is

$$u(k) = u(k-1) + [0.0259, 0.0876, 0.0981, 0.0513, -0.0538](Y_r - \mathcal{F}_2\Delta U(k-j) - \mathcal{G}Y(k))$$

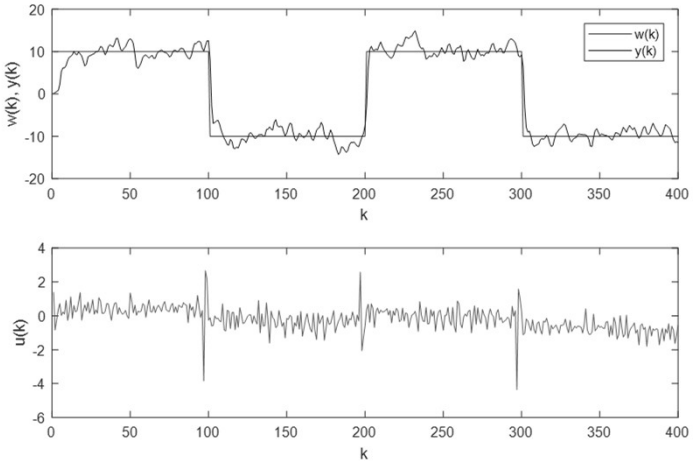
Assume the initial input/output data $\Delta u(-4), \Delta u(-3), \Delta u(-2), \Delta u(-1), \Delta u(0)$, $\Delta u(-4)$, $y(-2)$, $y(-1)$ and $y(0)$ are all 0.

Take the desired output $w(k)$ as the step signal with amplitude of 10, and make the white noise $\xi(k)$ have variance of $\sigma^2 = 0$, then recursively calculate $y(k)$ and $\mathrm{u}(k), \text{for k} = 1, 2, \dots$.

Take $n_1 = d = 4$, $n_y = 8$, $n_u = 2$, the weighting matrix Γ=I, and the output softening coefficient $\alpha = 0.7$.

46

Simulation results:



Simulation codes

example661.m

47

Case 2: the plant parameters are unknown (self-tuning GPC algorithm)

When the parameters of the controlled plant are unknown, the self-tuning control algorithm can be used, that is, the recursive least squares method is employed to estimate the parameters of the plant in real time online, and then the GPC law is designed.
See "Algorithm 6.6.2" for the implementation steps.

Setup:

The initial values $P(0) = 10^6 I,\ \hat{\theta}(0) = 0.001$

Forgetting factor $\lambda = 1$

Control parameters $n_1 = d = 4$, $n_y = 8$, $n_u = 2$,

Control weighting matrix Γ is a unit matrix.

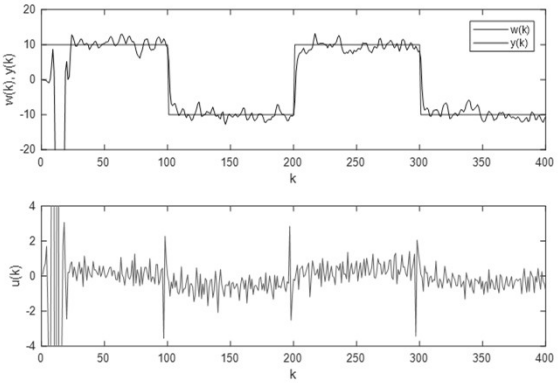
Output softening coefficient $\alpha = 0.7$.

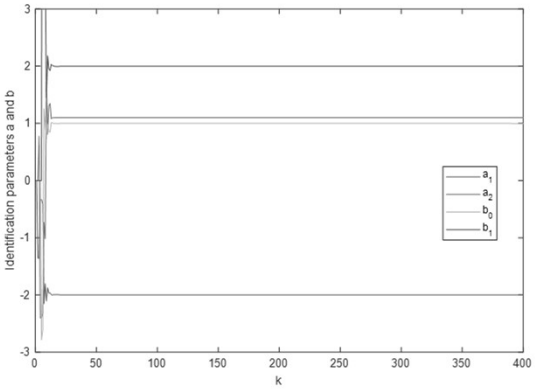
The expected output $w(k)$ is a square wave signal with amplitude of 10

The self-tuning GPC algorithm is adopted

48

Results





Simulation codes

example662.m

49

Exercise 6.3

Consider a time-varying plant

$$y(k) + a_1 y(k-1) + a_0 y(k-2) = b_1 u(k-3) + b_0 u(k-4) + \xi(k) + 0.2\xi(k-1)$$

where $a_1 = -1.7\big(1 + 0.1\sin(0.004\pi k)\big), a_0 = 0.7\big(1 - 0.2\cos(0.005\pi k)\big)$,
$b_1 = \big(1 + 0.1\sin(0.006\pi k)\big),\ b_0 = 0.5\big(1 - 0.05\cos(0.007\pi k)\big)$ and $\xi(k)$ is a white noise with variance of 0.01.

Assume the expected output $w(k)$ is a square wave signal with amplitude of 20 and period of 300 steps and the control weighting matrix Γ is a unit matrix.

Using the self-tuning generalized predictive control scheme, design self-tuning controllers and make simulations.

50