SELF-TUNING REGULATORS

A self-tuning regulator (STR) is a controller that automatically finds its parameters in the control law. Another name or synonym is self-adjusting controller. STR is a class of adaptive controllers used when the process to be controlled has constant but unknown parameters. However, STR can also be used in an adaptive context.

BASIC IDEA

Controller Design Procedure

The design of a controller contains several steps:

1. Finding specifications for the closed-loop system
2. Determination of a model for the process to be controlled
3. Decision on a design method
4. Calculation of the parameters in the controller

In many cases, it is desirable to automate these steps. This is the idea behind adaptive and self-tuning regulators.

The specifications for the closed-loop system depend on such things as quality constraints on the controlled variable, available magnitude (power) of the control signal, and nonlinearities of the system to be controlled. This implies that the specifications are determined by the process engineer at the start of the design procedure. The specifications often lead to a natural choice of the design method. For instance, if the main specification is to keep the process output constant and if the disturbances are occasional large disturbances, then the design procedure can be a method that as quickly as possible eliminates the influence of the disturbance. The choice of specifications and design method is thus usually made by the designer of the control loop. In STRs, as well as in adaptive controllers, steps 2 and 4 above are automatically taken care of by the controller.

The structure of a self-tuning controller is best described from the block diagram in Fig. 1. The self-tuning regulator consists of two closed loops. The first loop is a conventional controller feedback-loop consisting of the process and the controller where the output of the process is measured and compared with the desired output (reference signal) of the closed-loop system. The mismatch between the reference and output signals is used to compute the control action that is sent to the process. The controller has parameters that determine its properties. These parameters are determined by the second loop in the STR, the updating loop.

In Fig. 1, the updating loop has two main blocks. The first block is an estimator, which determines a mathematical model of the process based on the measured inputs and outputs. The second block carries out the design of the controller. This block uses the process model and the specifications to determine the controller parameters that then are sent to the controller.

It is necessary that the controller feedback-loop be closed all the time to take care of the influence of disturbances and changes in the reference signal. The updating loop for the controller parameters can be switched off as soon as the estimated parameters have converged to their final values, that is, when the controller has tuned or adjusted itself to the specifications and the process. The result is a self-tuning regulator. However, if the process is changing over time it is necessary to update continuously the process model and the controller parameters. We then have an adaptive controller. This implies that an STR is an adaptive controller if the parameter updating is not switched off. The STRs are thus a special class of adaptive controllers.

One of the first descriptions of the idea of STRs is found in Kalman (1) where updating using parameter estimation and design is described. The term self-tuning regulator was coined by Åström and Wittenmark (2) who gave the first analysis of the steady-state properties of the STR based on minimum variance control. The stability of the closed-loop system and the convergence properties were analyzed in Goodwin, Ramadge and Caines (3). More details of the properties of self-tuning and adaptive controllers can be found in Wellstread and Zarrop (4) and Åström and Wittenmark (5).

Classification of Self-Tuning Regulators

The STR in Fig. 1 contains both a block for estimation and a block for design. An STR in this configuration is usually called an indirect self-tuning regulator. The reason is that the controller parameters are obtained indirectly by first finding a process model. In many cases it is possible to make a reparameterization of the process and the controller such that the controller parameters can be estimated directly. This leads to a direct STR. Ways to do this reparameterization are discussed below.

Applications of Self-Tuning Regulators

The computations in an STR are quite straightforward, but contain nonlinear and logical operations. This implies that STRs are implemented using computers. The algorithm can be a block in a software package that is used for larger process control applications, or the STR can be implemented in dedicated hardware for a few control loops.

Self-tuning control has, since the mid-1970s, been used for many applications, mainly in the process industry. Applications are found in areas of pulp and paper, chemical reactors, autopilots, and dialysis machines.

Self-tuning regulators and adaptive controllers in general have found their main uses in three categories of applications:

- When the process has long time delays
- When feedforward can be used
- When the disturbances acting on the process have time-varying characteristics

The main reason self-tuning or adaptive controllers have a great advantage in these cases is that for good control of these types of processes it is necessary to have models of the process and/or of the disturbances to be controlled. The estimator part of the self-tuning controller can make an estimate of the process and use that in the design.

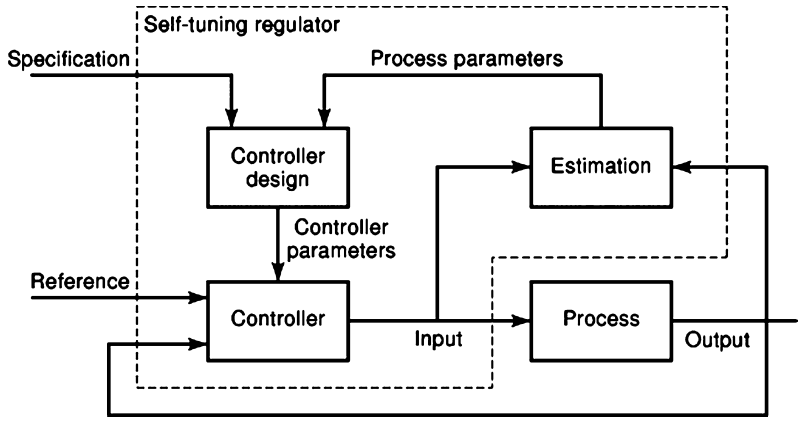

Figure 1. Block diagram of a self-tuning regulator.

Linear STRs are not appropriate to use when the process is very nonlinear. The updating mechanism will then not be sufficiently fast. In such cases the nonlinearities should be built into the process model and the controller.

ALGORITHMS FOR SELF-TUNING CONTROL

This section describes in more detail how STRs are constructed. It also gives the main properties of STRs. To describe the algorithms we need to specify the process model, the specifications, the controller, the estimator, and the design method. We will use discrete-time models for the process and the controller since most implementations of STRs are done using computers. It is, however, also possible to derive continuous-time STRs.

Process Model

The process is described as a sampled-data linear system. The process is also assumed to have a single input and a single output. The model is given as a difference equation

$$y(k)+a_1y(k-1)+\ldots+a_ny(k-n) = b_0u(k-d)+b_1u(k-d-1)+\cdots+b_mu(k-d-m) \quad (1)$$

where $y(k)$ is the output signal at sampling instant k and $u(k)$ is the control signal. Disturbances will be introduced below. It is assumed that the time is scaled such that the sampling period is one time unit. The parameter d is the time delay of the system. Equation (1) is a general description of a linear sampled-data system. To get a more compact way of describing the system, we introduce the backward-shift operator q^{-1}. The backward-shift operator is defined in the following way

$$q^{-1}y(k) = y(k-1)$$

That is, operating on a time sequence it shifts the time argument one step backwards. Using the backward-shift operator and the polynomials

$$A^*(q^{-1}) = 1+a_1q^{-1}+a_2q^{-2}+\ldots+a_nq^{-n}$$
$$B^*(q^{-1}) = b_0+b_1q^{-1}+b_2q^{-2}+\ldots+b_mq^{-m}$$

the system can be written as

$$A^*(q^{-1})y(k) = B^*(q^{-1})u(k-d) \quad (2)$$

or

$$y(k) = \frac{B^*(q^{-1})}{A^*(q^{-1})}u(k-d) = H(q^{-1})u(k-d)$$

where $H(q^{-1})$ is called the pulse-transfer function.

Specifications

The specifications give the desired performance of the closed-loop system. The specifications can be given in many different ways depending on the purpose of the closed-loop system. It is common to distinguish between the servo and regulator cases.

In the servo case, we give the desired performance in the form of the time or frequency response when the reference value is changed or when an occasional large disturbance has influenced the system. The typical specifications are bandwidth or response time. Further, things as overshoot or damping can be specified. One way to give the specifications is in the form of a reference model H_m defining the desired output y_m

$$y_m(k) = \frac{B_m^*(q^{-1})}{A_m^*(q^{-1})}u_c(k-d_m) = H_m(q^{-1})u_c(k-d_m) \quad (3)$$

where u_c is the reference signal. Normally, $d_m = d$, but may also be longer.

In the regulator case, we study the closed-loop performance when disturbances essentially are acting on the system while the reference signal is constant. The disturbance is then usually modeled as a stochastic process, in general, filtered white noise. Typical performance indices are to minimize the variance of the output signal around a desired reference value or to minimize a combination of output and input variations.

Controller

The controller is defined as

$$R^*(q^{-1})u(k) = -S^*(q^{-1})y(k)+T^*(q^{-1})u_c(k) \quad (4)$$

The controller has a feedback part defined by the polynomials $R^*(q^{-1})$ and $S^*(q^{-1})$ and a feedforward part defined by $R^*(q^{-1})$ and $T^*(q^{-1})$. Using Eq. (4) on the process of Eq. (2) gives the closed-loop system

$$\begin{aligned} y(k) &= \frac{B^*(q^{-1})T^*(q^{-1})}{A^*(q^{-1})R^*(q^{-1}) + B^*(q^{-1})S^*(q^{-1})}u_c(k-d) \\ &= H_c(q^{-1})u_c(k-d) \end{aligned} \tag{5}$$

Estimator

Estimation of process models can be done in many different ways. Summaries of methods and their properties can be found in Ljung (6), Söderström and Stoica (7), and Johansson (8). Here only the recursive least squares (RLS) method will be discussed. Define the vectors

$$\begin{aligned} \theta^T &= [a_1, a_2, \ldots, a_n, b_0, \ldots, b_m] \\ \varphi^T(k-1) &= [-y(k-1), -y(k-2), \ldots, \\ &\quad -y(k-n), u(k-d), \ldots, u(k-d-m)] \end{aligned}$$

The vector θ contains the unknown process parameters, while the vector ϕ contains known old inputs and outputs of the process. The process model can now be written as

$$y(k) = \varphi^T(k-1)\theta$$

The least squares method, first stated in Gauss (10), implies that the estimate of θ should be chosen as $\hat{\theta}$, which minimizes the loss function

$$V(\hat{\theta}, k) = \frac{1}{2}\sum_{i=1}^{k}[y(i) - \varphi^T(i-1)\hat{\theta}]^2 \tag{6}$$

Given an initial value of the parameters $\hat{\theta}(0)$ and the uncertainty of the parameter estimate $P(0)$, it is possible to derive a recursive solution to the least squares problem. The parameter estimate can be updated recursively using

$$\begin{aligned} \hat{\theta}(k) &= \hat{\theta}(k-1) + K(k)[y(k) - \varphi^T(k-1)\hat{\theta}(k-1)] \\ K(k) &= P(k)\varphi(k)[I + \varphi^T(k)P(k-1)\varphi(k)]^{-1} \\ P(k) &= P(k-1) - P(k-1)\varphi(k)[I + \varphi^T(k)P(k-1)\varphi(k)]^{-1} \\ &\quad \varphi^T(k)P(k-1) \\ &= [I - K(k)\varphi^T(k)]P(k-1) \end{aligned} \tag{7}$$

This is called the recursive least squares algorithm. The estimate at time k is obtained as an update of the estimate at time $k-1$. The correction term depends on the latest process output, which is compared with the predicted output based on the parameter estimate at time $k-1$. The matrix $P(k)$ can be interpreted as an estimate of the uncertainty of the parameter estimate at time k. The statistical interpretation can be made rigorous by making assumptions about the disturbances that are acting on the system.

The recursive least squares method is well suited for process parameter estimation when there are no disturbances or when a white noise process is added to the right-hand side of Eq. (2). For other noise or disturbance assumptions, there are variants of the recursive least squares method that can be used.

Since the updating formulas of Eq. (7) are recursive, they can be used also for a continuous updating of the parameters. In such cases it is, however, necessary to introduce a weighting of old inputs and outputs. The loss function of Eq. (6) puts equal weight on all data. A measurement collected a long time ago is as important as the latest measurement. Newer measurements can be given more weight by changing the loss function of Eq. (6) to

$$V(\hat{\theta}, k) = \frac{1}{2}\sum_{i=1}^{k}\lambda^{k-i}[y(i) - \varphi^T(i-1)\hat{\theta}]^2$$

where λ is the forgetting factor. Since the weights are exponentially decaying, the resulting algorithm is called recursive least squares with exponential forgetting. The updating formulas are only slightly modified into

$$\begin{aligned} \hat{\theta}(k) &= \hat{\theta}(k-1) + K(k)[y(k) - \varphi^T(k)\hat{\theta}(k-1)] \\ K(k) &= P(k)\varphi(k)[\lambda I + \varphi^T(k)P(k-1)\varphi(k)]^{-1} \\ P(k) &= \big(P(k-1) - P(k-1)\varphi(k)[\lambda I + \varphi^T(k)P(k-1)\varphi(k)]^{-1} \\ &\quad \varphi^T(k)P(k-1)\big)/\lambda \\ &= [I - K(k)\varphi^T(k)]P(k-1)/\lambda \end{aligned}$$

In the following we will use the recursive least squares algorithm, with or without exponential forgetting, to illustrate the properties of STRs.

Design Methods

The final step in the construction of an STR is the design procedure. The basic STRs are based on the certainty equivalence principle. This implies that the process parameter estimates obtained from the estimator are used as if they are the true ones. The design principles can, however, be extended to include also the uncertainty of the estimates, given by the P matrix. This leads to so-called cautious or dual controllers.

Two different design principles will be discussed in detail, pole-placement design and minimum variance control. In-depth treatments of the design methods can be found in Åström and Wittenmark (9).

Pole-Placement Design. We will now discuss how the parameters in the controller in Eq. (4) can be determined using the method of pole placement for the design of the controller. The closed-loop system is defined by Eq. (5). The closed-loop characteristic polynomial is thus

$$A^*R^* + B^*S^* = A_c^* \tag{8}$$

where A^*_c is given as a specification by the designer. The key idea is now to find the controller polynomials R^* and S^* that fulfill this equation. Equation (8) is called a diophantine equation. The desired closed-loop system from the reference signal to the output defined by Eq. (3) requires that the following condition must hold

$$\frac{B^*T^*}{A^*R^* + B^*S^*} = \frac{B^*T^*}{A_c^*} = \frac{B_m^*}{A_m^*} \tag{9}$$

This design procedure is called model-following design, and also pole-placement design, if the poles only are specified.

Whether model following can be obtained depends on the model, the process, and the complexity of the controller.

The characteristic polynomial of Eq. (8) will, in general, have higher degree than the model polynomial A_m^*. This implies that there must be a pole–zero cancellation in Eq. (9). The consequences of this will now be discussed. The B^* polynomial of the process is first factored into

$$B^* = B^{+*}B^{-*}$$

where B^{+*} corresponds to the process zeros that can be cancelled in the design. These zeros must be located inside the unit circle. The zeros corresponding to B^{-*}, which are not allowed to be canceled, must then be a factor of B_m^*, which must have the form

$$B_m^* = B^{-*}B_m^{*\prime}$$

Since B^{+*} is canceled, it must be a factor of A_c^*. The closed-loop characteristic polynomials is thus of the form

$$A_c^* = A_o^* A_m^* B^{+*} = A^* R^* + B^* S^* \qquad (10)$$

The polynomial A_o^* is called the observer polynomial, and can be interpreted as the dynamics of a state observer. The observer polynomial influences, for instance, how fast the system will recover after a disturbance. A_o^* is determined by the designer and should be a stable polynomial.

Since B^{+*} is a factor of B^* and A_c^*, it follows from Eq. (10) that it also is a factor of R^*, which implies that

$$R^* = R^{*\prime}B^{+*}$$

and the diophantine equation reduces to

$$A^* R^{*\prime} + B^{-*}S^* = A_o^* A_m^*$$

Finally, the polynomial T^* is given by

$$T^* = A_o^* B_m^{*\prime}$$

The design procedure can now be summarized into:

Data. Given the process polynomials A^*, $B^* = B^{+*}B^{-*}$, and the observer polynomial A_o^*

Step 1. Solve the diophantine equation with respect to $R^{*\prime}$ and S^*.

Step 2. The controller is given by Eq. (4) with $R^* = R^{*\prime}B^{+*}$ and $T^* = A_o^* B_m^{*\prime}$.

The diophantine equation can always be solved if there are no common factors between the A^* and B^* polynomials and if the controller polynomial has sufficiently many parameters.

Minimum Variance Control. Most design procedures can be interpreted as a pole-placement or model-following design. For instance, the minimum variance controller can easily be formulated in this form. The minimum variance controller is a controller that minimizes the variance of the output from the process. In this case, we add a disturbance term $C^*(q^{-1})e(k)$ on the right-hand side of Eq. (2), where C^* is a stable polynomial and $e(k)$ is white noise. The minimum variance controller is obtained by solving the diophantine equation

$$C^*(q^{-1}) = A^*(q^{-1})F^*(q^{-1}) + q^{-d}G^*(q^{-1}) \qquad (11)$$

and using the control law

$$u(k) = -\frac{G^*(q^{-1})}{B^*(q^{-1})F^*(q^{-1})}y(k) = -\frac{S^*(q^{-1})}{R^*(q^{-1})}y(k) \qquad (12)$$

Also, linear quadratic gaussian controllers can be interpreted as solving a special form of the diophantine equation, see Åström and Wittenmark (9).

Design of Self-Tuning Regulators

The design of STRs can be summarized by the following procedure:

Specifications. Determine the class of controller by determining the specifications on the closed-loop system.

Estimation. Estimate the process parameters using, for instance, the recursive least squares algorithm of Eq. (7).

Design Procedure. Determine the controller parameters using the estimated process parameters as if they are the correct ones. The controller design is usually reduced to the solution of an equation such as the diophantine Eq. (8).

Control. Update the parameters of the controller, for instance, in the form of the controller in Eq. (4).

The estimation, design, and control steps are done at each sampling interval. In some situations, it may be sufficient to update the estimation at a slower rate than the rate of the control loop. The behavior of the basic indirect self-tuning algorithm will be described by an example.

Example 1: Indirect Deterministic Self-Tuning Regulator Assume that the open-loop process is described by the continuous-time system

$$G(s) = \frac{1}{s(s+1)}$$

The process has an integrator and a time constant of 1 s. There are no disturbances acting on the system and the specifications are that the controlled system should be able to follow constant reference signals without too much overshoot. Sampling the system with the sampling interval $h = 0.5$ s gives the sampled-data description

$$H(q^{-1}) = \frac{b_0 q^{-1} + b_1 q^{-2}}{1 + a_1 q^{-1} + a_2 q^{-2}} = \frac{0.1065q^{-1} + 0.0902q^{-2}}{1 - 1.60065q^{-1} + 0.6065q^{-2}}$$

There is a process zero in $-b_1/b_0 = -0.85$. The zero is inside the stability boundary, but it is still decided not to cancel the zero. Let the desired closed-loop system be

$$\frac{B_m^*(q^{-1})}{A_m^*(q^{-1})} = \frac{K(0.1065q^{-1} + 0.0902q^{-2})}{1 - 1.3205q^{-1} + 0.4966q^{-2}}$$

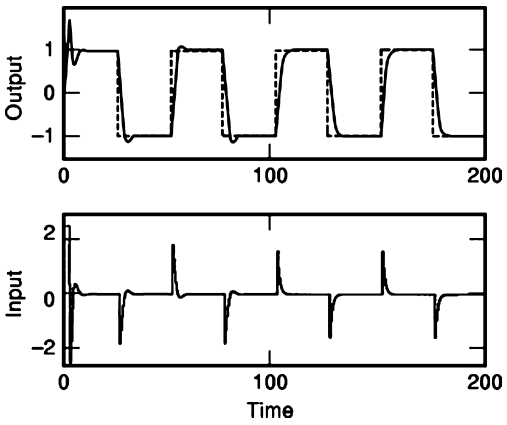

Figure 2. Process output and input when an indirect self-tuning regulator is used to control the process in Example 1. The specifications are changed at time 100. The reference signal is shown as a dashed curve.

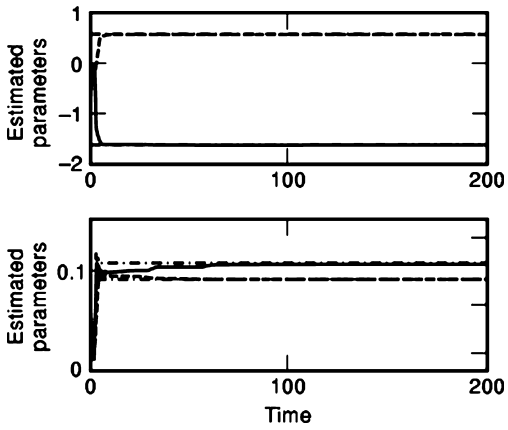

Figure 3. Parameter estimates corresponding to the simulation in Fig. 2. Upper diagram: $\hat{a}_1$ (full) and $\hat{a}_2$ (dashed), lower diagram: $\hat{b}_0$ (full) and $\hat{b}_1$ (dashed). The true parameter is shown by dashed-dotted lines.

This corresponds to a continuous-time system with natural frequency $\omega = 1$ and a damping of $\zeta = 0.7$ sampled with the sampling period $h = 0.5$. The gain K is chosen such that the steady-state gain from the reference signal to the output is equal to one, that is, $B_m^*(1)/A_m^*(1) = 1$. The controller solving the design problem will have the structure

$$(1 + r_1 q^{-1})u(k) = -(s_0 + s_1 q^{-1})y(k) + (t_0 + t_1 q^{-1})u_c(k)$$

Figure 2 shows the output and the control signal when the process is controlled by a self-tuning controller. The reference signal is a square wave. It is seen that the output behaves well already at the second change of the reference signal. At time 100 the design specifications are changed and the damping is changed from $\zeta = 0.7$ to $\zeta = 1$. The closed-loop response is immediately changed. The process model has four unknown parameters, b_0, b_1, a_1, and a_2. These parameters are estimated using the RLS algorithm, and the estimated process parameters are shown in Fig. 3. The example shows that the STR can find good controller parameters very quickly and that the design parameters can be changed. The transient in the beginning depends on the choice of initial values in the estimator.

Direct Self-Tuning Regulators

The self-tuning algorithm described above relies on a separation between the estimation and the design. The design step is repeated at each sampling instant. It can, on some occasions, be desirable to avoid the computations done in the design step, for instance, because of computing time limitations. One way to do this is to convert the indirect STR into a direct STR. This implies that the controller parameters are estimated instead of the process parameters. How to do this reparameterization will be illustrated on the minimum variance controller.

Let the system to be controlled be described by

$$A^*(q^{-1})y(k) = B^*(q^{-1})u(k-d) + C^*(q^{-1})e(k) \qquad (13)$$

The design specification is to minimize the variance of the output signal. Minimum variance control is equivalent to predicting the output signal d steps ahead and choosing the control signal such that the predicted value is equal to zero, or any other desired set point value. The prediction horizon should be equal to d, which is the delay in the process.

From Ref. 5 or Ref. 9 it follows that the output of Eq. (13) can be written as

$$y(k+d) = F^*e(k+d) + \frac{G^*}{C^*}y(k) + \frac{B^*F^*}{C^*}u(k) \qquad (14)$$

where F^* and G^* are obtained from the diophantine Eq. (11). The predicted output d steps ahead is given by the second and third terms on the right-hand side of Eq. (14). The prediction error is given by the first term on the right-hand side of Eq. (14). The prediction error is a moving average stochastic process that is independent of the predicted output. The predicted output is zero if the control law is chosen according to Eq. (12). Using the underlying design principle, it has been possible to reparameterize the model of Eq. (13) such that the reparameterized model explicitly contains the controller parameters. The controller parameters then can be estimated directly. Using the minimum variance controller, the closed-loop system becomes

$$y(k) = F^*(q^{-1})e(k)$$

The idea behind the basic direct STR is to estimate the parameters in the prediction model

$$y(k+d) = S^*(q^{-1})y(k) + R^*(q^{-1})u(k) + \epsilon(k+d) \qquad (15)$$

and use the controller

$$u(k) = -\frac{S^*(q^{-1})}{R^*(q^{-1})}y(k)$$

The estimated parameters are thus the same as the controller parameters, and the design step has been eliminated.

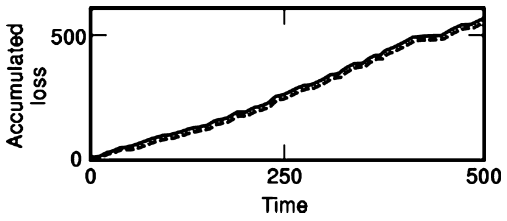

Figure 4. The accumulated loss function $V(k)$ when the direct self-tuning algorithm (full) and the optimal minimum variance controller (dashed) are used on the process in Example 2.

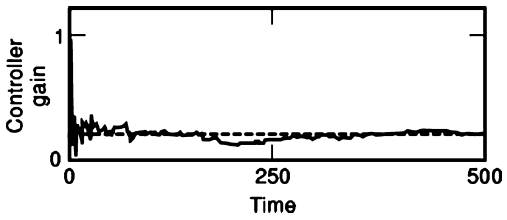

Figure 5. The controller gain $\hat{s}_0/\hat{I}_0$ when the self-tuning algorithm is used (full). The gain of the optimal minimum controller is shown as a dashed line.

Example 2: Direct Minimum Variance Self-Tuning Algorithm Assume that the open-loop process is described by the sampled-data model

$$y(k) - 0.9y(k-1) = 3u(k-1) + e(k) + 0.3e(k-1)$$

where $e(k)$ is white noise with variance 1. The time delay in the system is $d = 1$. Estimate the parameters r_0 and s_0 in the model

$$y(k+1) = s_0 y(k) + r_0 u(k) + \epsilon(k+1)$$

and use the controller

$$u(k) = -\frac{\hat{s}_0(k)}{\hat{r}_0(k)} y(k)$$

The optimal minimum variance controller is given by $u(k) = -0.2y(k)$, which is a proportional controller. Using this controller gives the output $y(k) = e(k)$, that is, the output should be white noise with a variance of 1. One way to compare the optimal and the self-tuning regulators is to compare the accumulated loss functions

$$V(k) = \sum_{i=1}^{k} y^2(i)$$

The slope of the accumulated loss function is an estimate of the variance of the output.

Figure 4 shows the loss function when the self-tuning algorithm and when the optimal minimum variance controller is used. After a short initial transient, the slopes of the loss functions are the same, which indicates that the STR has converged to the optimal minimum variance controller. This can also be seen by looking at the gain of the controller shown in Fig. 5.

Elimination of Disturbances

We will now see how the influence of disturbances can be reduced by introducing integrators and by using feedforward.

Introduction of Integrators. Consider the process

$$y(k) = \frac{B^*(q^{-1})}{A^*(q^{-1})}[u(k-d) + v(k)] \tag{16}$$

which is a slight variant of Eq. (2). The signal $v(k)$ is an input load disturbance. If this is, for instance, a step, then there needs to be an integrator in the controller to eliminate the influence of this disturbance. There are several ways to cope with this in an STR. One way is to estimate the magnitude of the disturbance and to compensate for it in the controller. To do so, the tuning has to be active all the time since the disturbance may change over time. A recommended way is to introduce an integrator directly into the controller. This can be done by postulating that the R^* polynomial contains the factor $1 - q^{-1}$. This can be done in the direct as well as in the indirect algorithms.

In the indirect algorithm, it is necessary to modify the estimator since the disturbance will change the relations between the inputs and outputs. Load disturbances such as steps have a particularly bad influence on the estimated model in the low-frequency range. Let the disturbance be modeled as

$$A_d^* v(k) = e(k)$$

where $e(k)$ is a pulse, a set of widely separated pulses, or white noise. For instance, a step disturbance is generated by

$$A_d^* = 1 - q^{-1}$$

The model can now be described as

$$A_d^* A^* y(k) = A_d^* B^*[u(k-d) + v(k)] = A_d^* B^* u(k-d) + e(k)$$

Introduce the filtered signals $y_f(k) = A_d^* y(k)$ and $u_f(k) = A_d^* u(k)$. We thus get

$$A^* y_f(k) = B^* u_f(k-d) + e(k) \tag{17}$$

The new model has the equation error $e(k)$ instead of $v(k)$. The process model can now be estimated from Eq. (17). Based on the estimated model the controller design is done by solving the diophantine equation

$$A^* R^{*\prime}(1 - q^{-1}) + B^{-*} S^* = A_o^* A_m^*$$

and using the controller

$$R^{*\prime} A_d^* u(k) = -S^* y(k) + T^* u_c(k)$$

The controller now contains the factor A_d^*, which will eliminate the influence of the disturbance v.

In the direct minimum variance self-tuning algorithm an integrator can be introduced by changing the model of Eq. (16) and estimating the controller parameters from

$$y(k+d) = S^*(q^{-1}) y(k) + R^*(q^{-1}) \Delta u(k) + \epsilon(k+d)$$

where $\Delta u(k) = u(k) - u(k-1)$ and using the controller

$$u(k) = -\frac{S^*(q^{-1})}{\Delta R^*(q^{-1})}y(k) = -\frac{S^*(q^{-1})}{(1-q^{-1})R^*(q^{-1})}y(k)$$

which contains an integrator.

Feedforward from Measurable Disturbances. On many occasions it is possible to measure some of the disturbances acting on the system. A typical example is control of indoor temperatures. By measuring the outdoor temperature also it is possible to use this signal to compensate for changing outdoor temperatures before the disturbance has influenced the process too much. One way to introduce feedforward in STRs is exemplified with the direct algorithm. The estimated model is changed from Eq. (15) to

$$\begin{aligned} y(k+d) =& S^*(q^{-1})y(k) + R^*(q^{-1})u(k) \\ & - T^*(q^{-1})v_m(k) + \epsilon(k+d) \end{aligned}$$

where $v_m(k)$ is the measurable disturbance. The controller is now

$$u(k) = -\frac{S^*(q^{-1})}{R^*(q^{-1})}y(k) + \frac{T^*(q^{-1})}{R^*(q^{-1})}v_m(k)$$

The first part of the controller is the feedback from the measurement $y(k)$ and the second is the feedforward from the measurable disturbance $v_m(k)$. Feedforward is, in general, very useful in STRs because to make effective feedforward, it is necessary to have a good model of the process. By combining the measurement of the disturbance and the self-tuning property of the controller it is possible to eliminate much of the disturbance before it reaches the output of the process.

SOME THEORETICAL PROBLEMS

The previous section described the basic ideas of STRs. Self-tuning regulators are inherently nonlinear. The nonlinearities are due to the estimation part and the changing parameters in the controller. This makes the analysis of STRs very difficult. The STRs contain two feedback loops and it is necessary to investigate the stability and convergence properties of the closed-loop systems. This is a difficult question because of the interaction between the two feedback loops. One way to circumvent this problem is to make a time separation between the two loops. The controller loop is assumed to be fast compared to the updating loop. This makes it possible to use averaging theory to analyze the updating loop on a much longer time-scale. This approach has made it possible to derive results concerning stability and convergence of STRs.

Åström and Wittenmark (2) showed how to characterize the stationary properties of STRs, that is, the properties if and when the parameter estimation has converged. The algorithms were used in a number of applications before several of the theoretical problems were solved. Goodwin, Ramadge, and Caines (3) gave the first results showing when the algorithm converges and that the closed-loop system remains stable during the estimation phase. These results have lately been refined and extended [see Wellstead and Zarrop (4) and Åström and Wittenmark (5)].

One important theoretical aspect is the influence of unmodeled dynamics. Unmodeled dynamics are present if the estimator is trying to fit a too-simple model to the data. The unmodeled dynamics may cause severe stability problems, which must be avoided by introducing counter measures such as careful filtering of the signals in the STR. This type of problem has successfully been analyzed using averaging theory.

It is important that a controller is robust against assumptions and choices of controller parameters. Much theoretical research has been devoted to make STRs and adaptive controllers more robust. This work has resulted in practical rules of thumb for their implementation (see Ref. 5). Robust design methods are complementary to self-tuning and adaptive control. In robust control one fixed controller is designed to cope with a variety of processes. By using tuning and adaptation the parameters of the controller are instead tuned to adjust to the present process dynamics.

PRACTICAL ISSUES AND IMPLEMENTATION

Some problems in the implementation of STRs are discussed briefly in this section. Self-tuning regulators as well as adaptive controllers will run unattended on the processes. It is therefore very important that there be a good safety net around the self-tuning algorithm.

There are many aspects of STR implementation that are important for implementations of digital controllers in general [see Åström and Wittenmark (9)]. Some important issues for STRs are

- Organization of the computer code
- Sampling and filtering
- Antireset windup
- Design calculations
- Excitation
- Safety nets

It is important that the computer code be organized so that as little delay as possible is introduced by the controller. In STRs this usually implies that the estimation and the design calculations are done after the controlled signal is sent out to the process. The latest measurement is thus used in the computation of the control signal. The estimation and the design are then performed, which implies that the controller parameters are based on estimates from the previous sampling instant. This is usually no drawback since the estimated parameters are changing very little between samples, after the initial transient.

In all sampled-data controllers it is important that the sampling interval be chosen properly. The sampling interval should be chosen in relation to the desired closed-loop behavior. A common rule of thumb is that there should be four to ten samples per rise time of the closed-loop system. It is also necessary to filter the analog signals before they are sampled. The reason is the aliasing effect, which

implies that all frequencies over the Nyquist frequency π/h, where h is the sampling period, will be interpreted as a lower frequency signal after the sampling. These filters are called antialiasing filters. In the design of the controllers it is important also to incorporate the dynamics of the antialiasing filters since they introduce a phase lag in the system. The dynamics of the antialiasing filters will automatically be included in the estimated dynamics when self-tuning or adaptive controllers are used. It may be necessary only to increase the order of the estimated model to incorporate the filters into the estimated dynamics.

The indirect STRs contain a design calculation that normally involves the solution of a diophantine equation such as Eq. (8). This equation has no solution if the A^* and B^* polynomials have a common factor that is not also a factor in A_c^*. This also implies that the solution of the diophantine equation is a numerically ill-conditioned problem if there are almost common factors in A^* and B^*. These polynomials are obtained through estimation and there is no guarantee that there are no common factors. The factors that are close must thus be eliminated before solving the diophantine equation.

Parameter estimation is a crucial element of STRs. The estimation is relatively simple for processes with disturbances and set-point changes that excite the process all the time. If there is not enough excitation of the process, it is necessary to introduce a logical condition in the algorithm that ensures that controller parameters are not changed when there is no excitation of the process. The design of a good safety net for an STR is a difficult task that requires thorough knowledge of the details of the algorithms and an understanding of where difficulties may occur. Experience shows that a good safety net normally occupies much more code than the basic controller algorithm.

BIBLIOGRAPHY

1. R. E. Kalman, Design of self-optimizing control systems. *ASME Trans.*, **80**: 468–478, 1958.
2. K. J. Åström, B. Wittenmark, On self-tuning regulators. *Automatica*, **9**: 185–199, 1973.
3. G. C. Goodwin, P. J. Ramadge, P. E. Caines, Discrete-time multivariable adaptive control, *IEEE Trans. Autom. Control*, **AC-25**: 449–456, 1980.
4. P. E. Wellstead, M. B. Zarrop, *Selftuning Systems: Control and Signal Processing*, Chichester, U.K.: Wiley, 1991.
5. K. J. Åström, B. Wittenmark, *Adaptive Control*, 2nd ed. Reading, MA: Addison-Wesley, 1995.
6. L. Ljung, *System Identification—Theory for the User.* Englewood Cliffs, NJ: Prentice-Hall, 1987.
7. T. Söderström, P. Stoica, *System Identification.* Hemel Hempstead, U.K.: Prentice-Hall International, 1988.
8. R. Johansson, *System Modeling and Identification.* Englewood Cliffs, NJ: Prentice-Hall, 1993.
9. K. J. Åström, B. Wittenmark, *Computer-Controlled Systems*, 3rd ed. Englewood Cliffs, NJ: Prentice-Hall, 1997.
10. K. F. Gauss, Theoria motus corposum coelestium, (1809). English translation, *Theory of the Motion of the Heavenly Bodies.* New York: Dover, 1963.

BJÖRN WITTENMARK
Lund University
Lund, Sweden