Self-Tuning Regulators

Björn Wittenmark

Department of Automatic Control
Lund Institute of Technology, Box 118
S-221 00 Lund, Sweden

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

1. The basic idea

The basic ideas and components of self-tuning regulators are discussed in this section without too much of details.

The 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 of 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.

* For Encyclopedia of Electrical and Electronics Engineering

The specifications for the closed-loop system depend on things such as quality constraints on the controlled variable, available magnitude (power) of the control signal, 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 the specifications and the design method are thus usually done by the designer of the control loop. In the self-tuning regulators, as well as the 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 Figure 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 self-tuning regulator, the updating loop.

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

It is necessary that the controller feedback-loop is 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, i.e. 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 continuously update the process model and the controller parameters. We then have an adaptive controller. This implies that a self-tuning regulator is an adaptive controller if the parameter updating is not switched off. The self-tuning regulators are thus a special class of adaptive controllers.

One of the first descriptions of the ideas with self-tuning regulators is found in Kalman(1958) where the updating using parameter estimation and design is described. The term self-tuning regulator was coined in Åström and Wittenmark (1973) which gave the first analysis of the steady-state properties of the self-tuning regula-

tor based on minimum variance control. The stability of the closed-loop system and the convergence properties were analyzed in Goodwin, Ramadge and Caines (1980). More details of the properties of self-tuning and adaptive controllers can be found in Wellstead and Zarrop (1991) and Åström and Wittenmark (1995).

Classification of self-tuning regulators

The self-tuning regulator in Figure 1 contains both a block for estimation and a block for design. A self-tuning regulator 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 occasions it is possible to make a re-parameterization of the process and the controller such that the controller parameters can be estimated directly. This leads to a direct self-tuning regulator. Ways to do this re-parameterization are discussed below.

Applications of self-tuning regulators

The computations in a self-tuning regulator are quite straight forward, but contains nonlinear and logical operations. This implies that self-tuning regulators are implemented using computers. The algorithm can either be a block in a software package that is used for larger process control applications. The self-tuning regulator can also be implemented in dedicated hardware for a few control loops.

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

The 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 a time-varying characteristics

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

2. Algorithms for self-tuning control

This section describes in more detail how self-tuning regulators are constructed. It also gives the main properties of self-tuning regulators. 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 self-tuning regulators are done using computers. It is, however, also possible to derive continuous-time self-tuning regulators.

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

$$\begin{aligned} y(k) + a_1 y(k-1) + \ldots a_n y(k-n) \\ = b_0 u(k-d) + b_1 u(k-d-1) + \cdots + b_m u(k-d-m) \end{aligned} \tag{1}$$

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

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

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

$$\begin{aligned} A^*(q^{-1}) &= 1 + a_1 q^{-1} + a_2 q^{-2} + \ldots a_n q^{-n} \\ B^*(q^{-1}) &= b_0 + b_1 q^{-1} + b_2 q^{-2} + \ldots b_m q^{-m} \end{aligned}$$

the system can be written as

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

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

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

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

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

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 (4) on the process (2) gives the closed-loop system

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

Estimator

Estimation of process models can be done in many different ways. Summaries of methods and their properties can be found in Ljung (1987), Söderström and Stoica (1989), and Johansson (1993). Here only the recursive least squares method (RLS) will be discussed. Define the vectors

$$\theta^T = \begin{pmatrix} a_1, & a_2, & \ldots, & a_n, & b_0, & \ldots, & b_m \end{pmatrix}$$

$$\varphi^T(k-1) = \begin{pmatrix} -y(k-1), & -y(k-2), & \ldots, & -y(k-n), & u(k-d), & \ldots, & u(k-d-m) \end{pmatrix}$$

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 (1809), implies that the estimate of θ should be chosen as $\hat{\theta}$, which minimizes the loss function

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

Given an initial value of the parameters $\theta(0)$ 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

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

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 on the disturbance that is 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 (2). For other noise or disturbance assumptions there are variants of the recursive least squares method that can be used.

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

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

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

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

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

Design methods

The final step in the construction of a self-tuning regulator is the design procedure. The basic self-tuning regulators 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 also include 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 in the following:

- Pole-placement design
- Minimum variance control

In depth treatments of of the design methods can be found in Åström and Wittenmark (1997).

Pole-placement design We will now discuss how the parameters in the controller (4) can be determined using the method of pole-placement for the design of the controller. The closed-loop system is defined by (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 fulfills this equation. Equation (8) is called a Diophantic equation. The desired closed-loop system from the reference signal to the output defined by (3) gives 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 only the poles are specified. Whether model-following can be obtained depends on the model, the process, and the complexity of the controller.

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

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

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 (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 (11) with respect to $R^{*\prime}$ and S^*.

Step 2: The controller is given by (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 polynomials has sufficiently many parameters.

Minimum variance control Most design procedures can be interpreted as a 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 (2), where C^* is a stable polynomial and $e(k)$ is white noise. The minimum variance controller is obtained by solving the the Diophantine equation

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

and using the control law

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

Also the linear quadratic Gaussian (LQG) controllers can be interpreted as solving a special form of the Diophantine equation, see Åström and Wittenmark (1997).

Design of self-tuning regulators

The design of self-tuning regulators can be summarized in 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 (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 equation (8).

Control: Update the parameters of the controller, for instance, in the form of the controller in (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 in 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 of overshoot. Sampling the system with the sampling interval $h = 0.5$ s gives the sampled-data description

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

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 the desired closed-loop system be

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

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, i.e. $B_m^*(1)/A_m^*(1) = 1$. 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 Figure 3. The example shows that the self-tuning regulator very quickly can find good controller parameters 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, in some occasions, be desirable to avoid the computations done in the design step, for instance, due to computing time limitations. One way to do this is to convert the indirect self-tuning regulator into a direct self-tuning regulator. This implies that the controller parameters are estimated instead of the process parameters. How to do this re-parameterization will be illustrated on the minimum variance controller.

Let the system to be controlled be described by

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

The design specification is to minimize the variance of the output signal d steps ahead. Minimum variance control is equivalent to predicting the output signal 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 Åström and Wittenmark (1995) or Åström and Wittenmark (1997) it follows that the output of (14) can be written as

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

where F^* and G^* are obtained from the Diophantine equation (11). The predicted output d steps ahead is given by the second and third terms on the right hand side of (15). The prediction error is given by the first term on the right hand side of (15). 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 (13). Using the underlying design principle it has been possible to re-parameterize the model (14) such that the re-parameterized model explicitly contains the controller parameters. The controller parameters can then be estimated directly. Using the minimum variance controller the closed-loop system becomes

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

The idea behind basic direct self-tuning regulator is to estimate the parameters in the prediction model

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

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.

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 system is $d = 1$. Estimate the parameters r_0 and s_0 in the model

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

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 $y(k) = e(k)$, i.e. 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 self-tuning regulator has converged to the optimal minimum variance controller. This can also bee seen by looking at the gain of the controller shown in Figure 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 (2). The signal $v(k)$ is an input load disturbance. If this is, for instance, a step then there need to be an integrator in the controller to eliminate the influence of this disturbance. There are several ways to cope with this in a self-tuning regulator. One way is to estimate the magnitude of the disturbance and to compensate for it in the controller. To do so it is required that the tuning is 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 step have a particular 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 (17) 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 have the equation error $e(k)$ instead of $v(k)$. The process model can now be estimated from (18). 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 (16) and estimating the controller parameters from

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

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 At 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 also the outdoor temperature 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 self-tuning regulators is exemplified with the direct algorithm. The estimated model is changed from (16) to

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

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 in self-tuning regulators. The reason is that 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.

3. Some theoretical problems

The previous section described the basic ideas of the self-tuning regulators. 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 self-tuning regulators very difficult. The self-tuning regulators 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 too 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 self-tuning regulators.

In Åström and Wittenmark (1973) it was shown how to characterize the stationary properties of the self-tuning regulators. I.e. the properties if and when the parameter estimation has converged. The algorithms were used in several applications before several of the theoretical problems were solved. Goodwin, Ramadge, and Caines (1980) gave the first results showing when the algorithm converges and that the closed-loop system remains stable during the estimation phase. These results have lated been refined and extended, see Wellstead and Zarrop (1991) and Åström and Wittenmark (1995).

One important theoretical aspect is the influence of unmodeled dynamics. Unmodeled dynamics is 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 self-tuning regulator. This type of problems has successfully been analyzed using averaging theory.

4. Practical issues and implementation

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

There are many aspects of the controller implementation that is general for implementations of digital controllers, see Åström and Wittenmark (1997). Some im-

portant issues for self-tuning regulators are

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

It is important that the computer code is organized such that there are as little delay as possible introduced by the controller. In the self-tuning regulators 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 is 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 4–10 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 to also 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 only be necessary to increase the order of the estimated model to incorporate the filters into the estimated dynamics.

The indirect self-tuning regulators contain a design calculation that normally involves the solution of a Diophantine equation such as (8). This equation has no solution if the A^* and B^* has 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 the there almost common factors in A^* and B^*. These polynomials are obtained through the 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 self-tuning regulator. The estimation is relatively simple for processes with disturbances that excite the process all the time. If there is not enough excitation of the process it is necessary to introduce a code 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 a self-tuning regulator is a difficult task that requires good knowledge of 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.

References

Åström, K. J., and B. Wittenmark (1973): "On self-tuning regulators." Automatica, vol 9, pp 185–199

Åström, K. J., and B. Wittenmark (1995): *Adaptive Control*, second edition, Addison-Wesley, Reading, MA

Åström, K. J., and B. Wittenmark (1997): *Computer-Controlled Systems*, third edition, Prentice-Hall, Englewood Cliffs, N.J.

Gauss, K. F. (1809): *Theoria motus corposum coelestium,* (In Latin). English translation (1963): *Theory of the Motion of the Heavenly Bodies.* Dover, New York

Goodwin, G. C., P. J. Ramadge, and P. E. Caines (1980): "Discrete-time multivariable adaptive control." IEEE Trans. Automat. Contr., vol AC-25, pp 449–456

Johansson, R. (1992): *System Modeling and Identification.* Prentice-Hall, Englewood Cliffs, N.J.

Kalman, R. E. (1958): "Design of Self-optimizing Control Systems." ASME Transactions, vol 80, pp 468–478

Ljung, L. (1987): *System Identification—Theory for the User.* Prentice-Hall, Englewood Cliffs, N.J.

Söderström, T., and P. Stoica (1988): *System Identification.* Prentice-Hall International, Hemel Hempstead, U.K.

Wellstead, P. E., and M. B. Zarrop (1991): *Selftuning Systems: Control and Signal Processing*, John Wiley & Sons, Chichester, U.K.

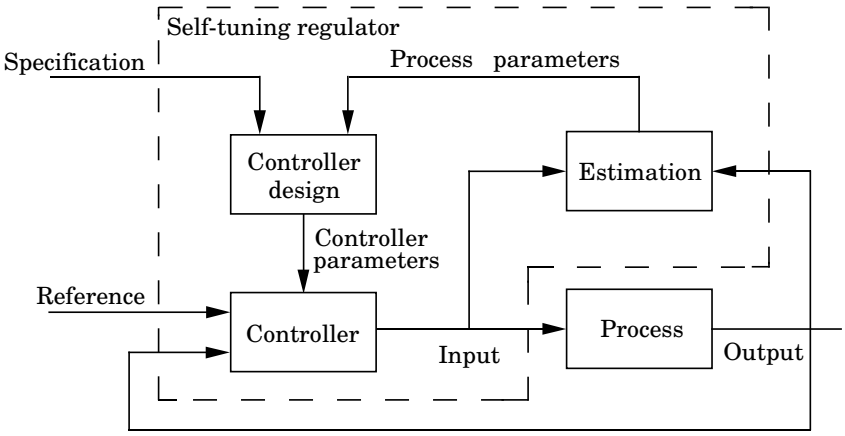

Figure 1 Block diagram of a self-tuning regulator (STR).

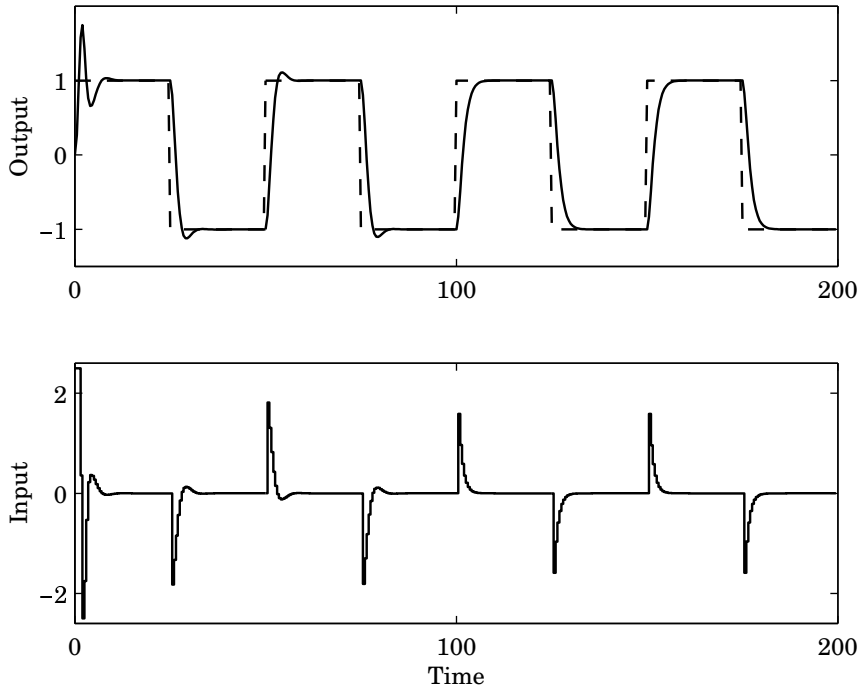

Figure 2 Process output and input when an indirect self-tuning regulator is used to control the process in Example 1. The specifications are changed at time 100. The reference signal is shown as a dashed curve.

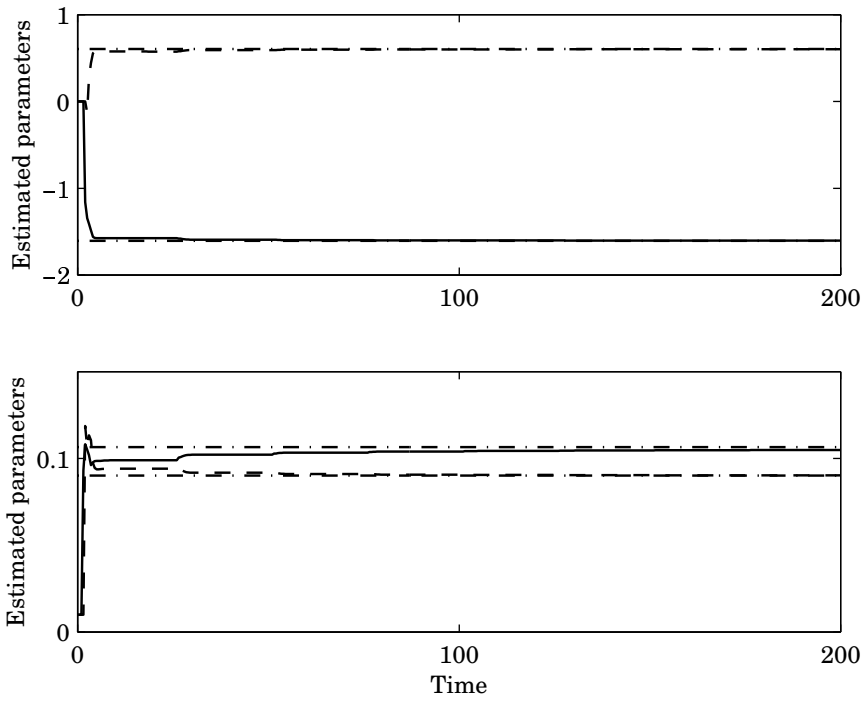

Figure 3 Parameter estimates corresponding to the simulation in Figure 2. Upper diagram: $\hat{a}_1$ (full) and $\hat{a}_2$ (dashed), lower diagram: $\hat{b}_0$ (full) and $\hat{b}_1$ (dashed). The true parameter are shown by dashed-dotted lines.

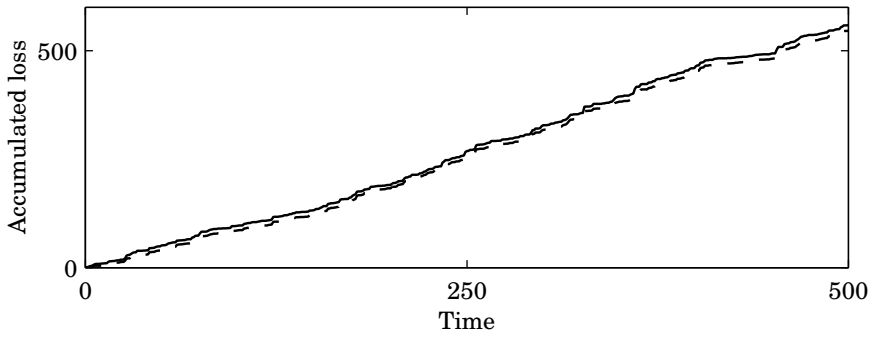

Figure 4 The accumulated loss function $V(k)$ when the direct self-tuning algorithm (full) and the optimal minimum variance controller (dashed) are used on the process in Example 2.

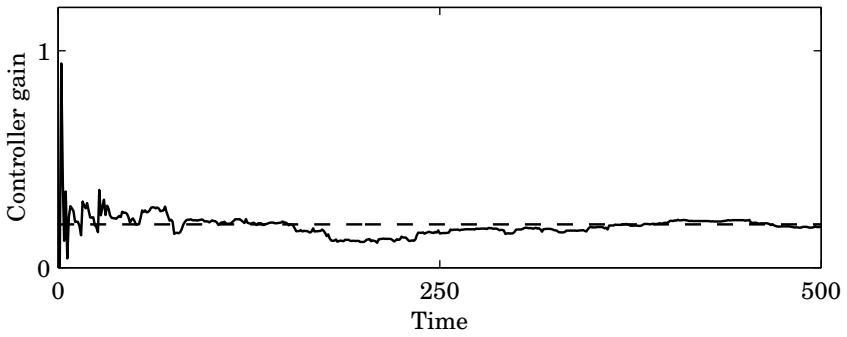

Figure 5 The controller gain $\hat{s}_0/\hat{r}_0$ when the self-tuning algorithm is used (full). The gain of the optimal minimum controller is show as a dashed line.