Adaptive Control: an introduction

Claudio Melchiorri

Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione (DEI)

Università di Bologna

email: claudio.melchiorri@unibo.it

Summary

Adaptive regulators

An adaptive regulator is able to modify *automatically* its own behaviour in order to react to variations in the process dynamics and/or to external disturbances. The goal is to guarantee in any case the achievement of pre-assigned constraints (design specifications) on the controlled system. Adaptive control schemes allow to:

- *estimate online* the value of the plant parameters (not known or time-varying),
- *adapt the control parameters* on the basis of this estimation.

In an adaptive control system, two feedback loops may be defined:

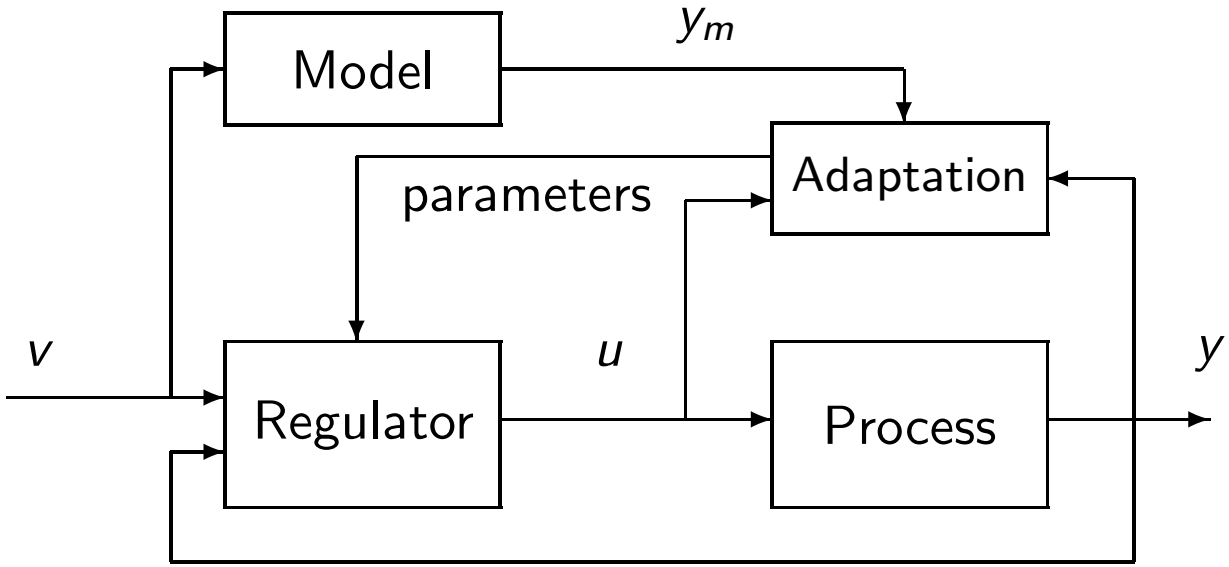
- the standard control feedback loop based on the output signal and acting on the input of the process (higher frequency)
- a loop taking into account the design specifications, acting on the parameters of the controller (lower frequency).

Adaptive regulators

Two general control schemes have been defined in the literature:

- Model Reference Adaptive System (MRAS)
- Self Tuning Regulator (STR)

In MRAS schemes, the adaptation law affects the control parameters in order to keep y, the output of the process, as similar as possible to y_m, the output of a reference model.

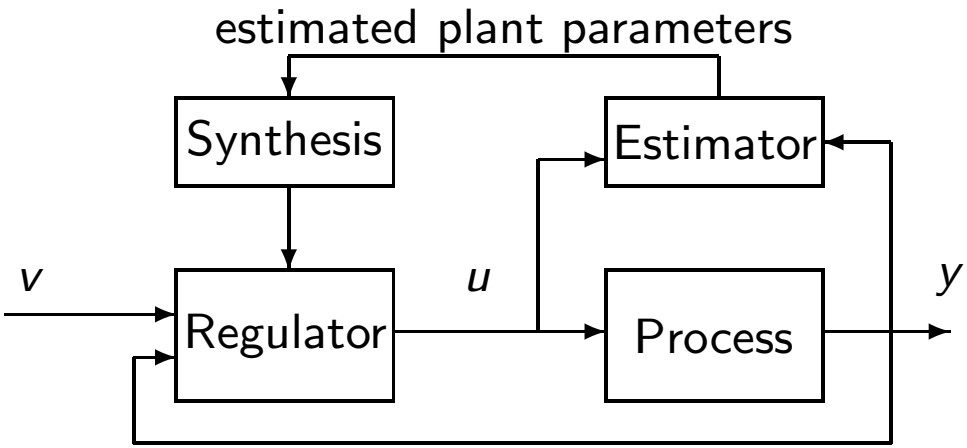


Adaptive regulators

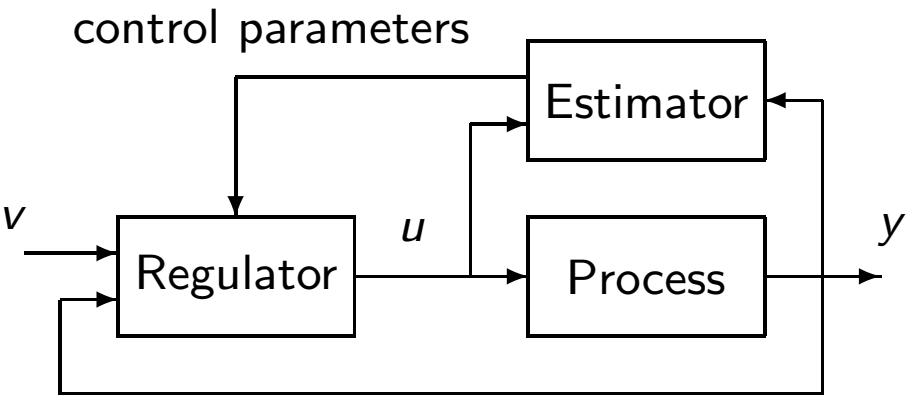
STR (Self Tuning Regulator) schemes:

- Indirect methods: the adaptation algorithm estimates the parameters of the plant, and then a synthesis procedure is used for the control algorithm
- Direct methods: the parameters of the control law are directly modified by the adaptation algorithm (as for MRAS)

Indirect STR scheme



Direct (implicit) STR scheme



Adaptive regulators

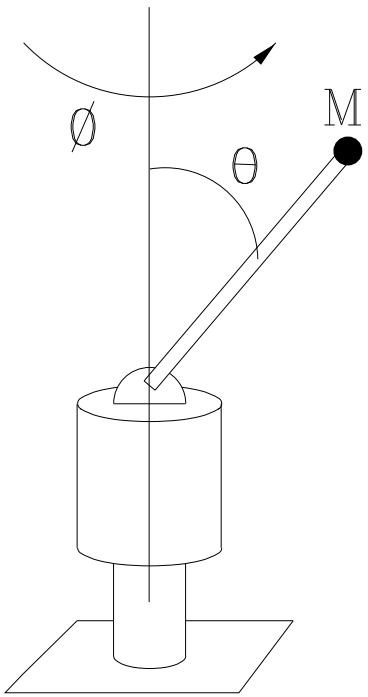
Adaptive control schemes:

- Certainty equivalence property: the parameters estimated online are considered, for the control synthesis, equal to the true ones (not known)
- Adaptive control schemes are *non linear* dynamic systems with *time-varying* parameters

Only a specific class of adaptive controllers is considered here: STR with *least squares estimation* and synthesis based on *pole/zero placement*.

Adaptive regulators - An example

Let us consider the control of the variable $\dot{\phi} = \omega$ (angular velocity about the vertical axis).



Define:

τ_m: input torque applied by the motor (joint ϕ)

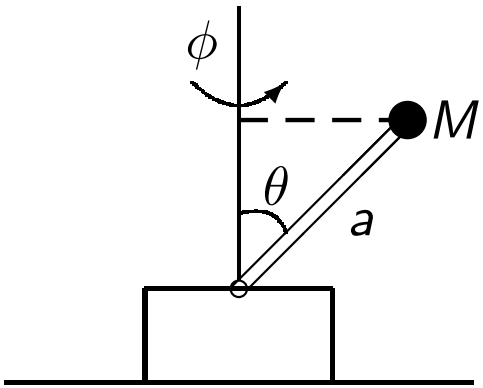
τ_a: friction torque

J_l: load inertia seen at the motor side, $J_l = J_l(M, \theta)$

Balance of torques at joint axis (if M is constant) and variation of angular momentum:

$$\sum \tau_i = \frac{d}{dt}\left(J\dot{\phi}\right) = J\ddot{\phi} + \frac{d\,J}{d\theta}\dot{\theta}\dot{\phi}, \qquad \text{if } \dot{\theta} = 0 \implies J\ddot{\phi} = \tau_a + \tau_m$$

Adaptive regulators - An example



The inertia J seen at the motor is given by

$$J = J_m + \frac{J_l}{k_r^2}$$

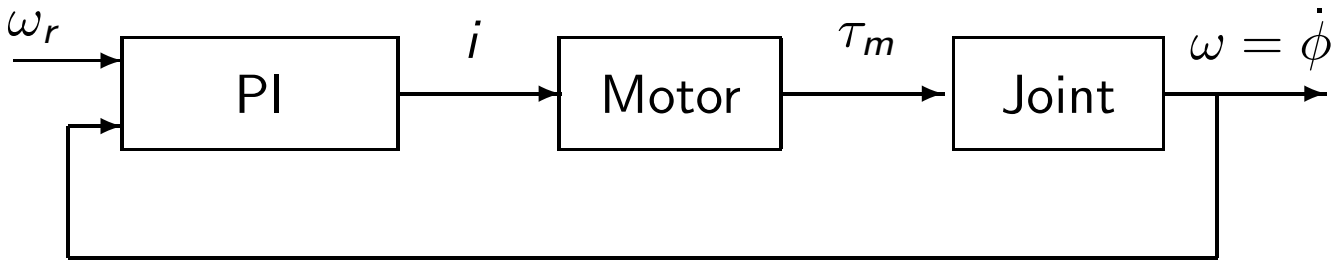
where

$$J_l = M\ a^2 \sin^2(\theta)$$

and k_r is the reduction ratio of the motor.

Adaptive regulators - An example

Assuming θ constant and $\tau_a = 0$, then $J\ddot{\phi} = \tau_m$, with $\tau_m = k_m i$



With a PI controller, the motor current i is computed as

$$i = k\left[(\omega_r - \omega) + \frac{1}{T_i}\int_0^t (\omega_r - \omega)dt\right]$$

Therefore $$J\frac{d^2\,\omega}{dt^2} + k_m k\frac{d\,\omega}{dt} + \frac{k_m k}{T_i}\omega = k_m k\frac{d\,\omega_r}{dt} + \frac{k_m k}{T_i}\omega_r$$

that is $$\frac{\omega(s)}{\omega_r(s)} = G_0(s) = \frac{2\delta_0\omega_0 s + \omega_0^2}{s^2 + 2\delta_0\omega_0 s + \omega_0^2}$$ with $$\begin{cases} k &= \dfrac{2\delta_0\omega_0 J}{k_m} \\ T_i &= \dfrac{2\delta_0}{\omega_0} \end{cases}$$

Adaptive regulators - An example

If the PI parameters k, T_i are computed for a nominal inertia value J_0, while the true value is J, one obtains:

$$G_0'(s) = \frac{2\delta_0\omega_0 s J_0/J + \omega_0^2 J_0/J}{s^2 + 2\delta_0\omega_0 s J_0/J + \omega_0^2 J_0/J}$$

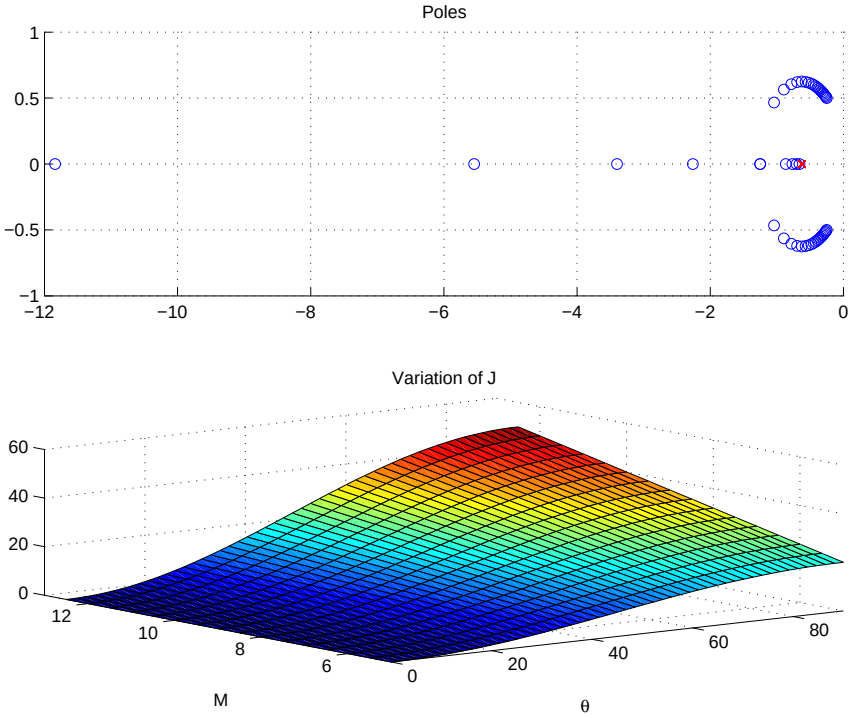
with natural frequency and dumping coefficient

$$\omega_n = \omega_0\sqrt{\frac{J_0}{J}}, \qquad \delta = \delta_0\sqrt{\frac{J_0}{J}}$$

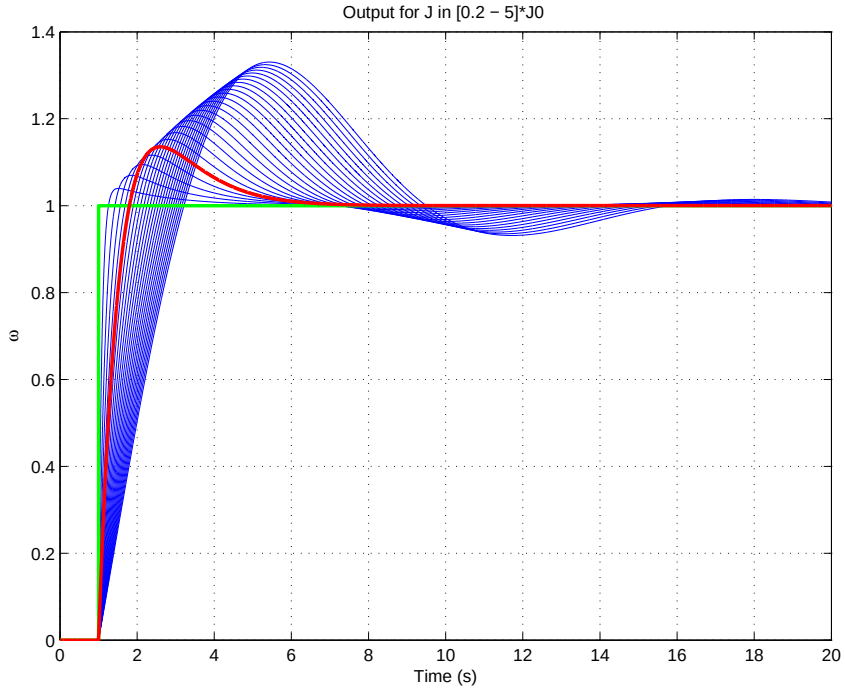
Assuming $\omega_0 = 1.25\ rad/s, \quad \delta_0 = 1,$

with	$J = 2J_0$	$\rightarrow$	$\omega_n = 0.8839,$	$\delta \approx 0.7071$
with	$J = \frac{1}{2}J_0$	$\rightarrow$	$\omega_n = 1.7678,$	$\delta \approx 1.4142$

Adaptive regulators - An example



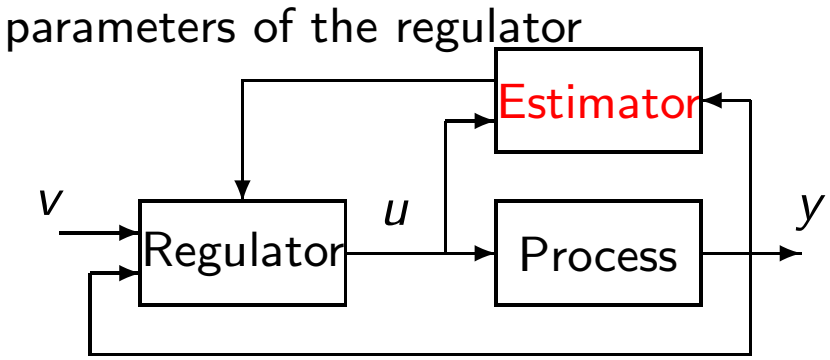
1) Poles when $J \in [0.2 \div 2]J_0$
2) Value of $J_l = M\ a^2 \sin^2(\theta)$



Plot of ω for $J \in [0.2 \div 2]J_0$
in red the nominal behaviour

Least squares estimation

An essential component of a STR control scheme is the *parameter estimation algorithm*, used to obtain the (unknown) parameters of the process.



- A quite common method is based on the *Least Squares algorithm*.
- Need of a *recursive formulation* of the algorithm.

In case of linear dynamic systems expressed by

$$G(z) = \frac{b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-m}} = \frac{Y(z)}{U(z)}$$

the output y in a given time instant k is expressed as

$$\begin{aligned} y(k) \quad = \quad & -a_1 y(k-1) - a_2 y(k-2) - \cdots - a_m y(k-m) + \\ & + b_1 u(k-1) + b_2 u(k-2) + \cdots + b_m u(k-m) \end{aligned} \tag{1}$$

that is as a *linear function* of the parameters $a_i, b_i,\ i = 1, \ldots, m$

Least squares estimation

A more general expression of this equation is given by the following *regression model* (linear in the parameters α_i):

$$y_k = \phi_1(x_k)\alpha_1 + \phi_2(x_k)\alpha_2 + \ldots + \phi_n(x_k)\alpha_n + e_k$$

where the variable e_k (the error) takes into account the uncertainties in the parameters.

If the values $\{y_k, x_k\}$, $k = 1, \ldots, N$, are known, the goal is to determine the parameters $\alpha_i, i = 1, \ldots, n$ (note: $n \leq N$) so that the error e_k is minimised.

Assuming that a proper norm can be defined (i.e. $\|e\| = \sum_k e_k^2 = e^T e$), a formal way to achieve this result is to compute the parameters α_i in order to minimize, for example, the function:

$$V = \sum_{k=1}^{N} e_k^2, \qquad \text{i.e.} \qquad \min_{\alpha_i} \sum_{k=1}^{N} e_k^2$$

Least squares estimation

This problem may be written in vector form by defining

$$\begin{aligned} y &= [y_1, y_2, \ldots, y_N]^T \\ e &= [e_1, e_2, \ldots, e_N]^T \\ \alpha &= [\alpha_1, \alpha_2, \ldots, \alpha_n]^T \end{aligned}$$

$$\phi(k) = [\phi_1(x_k), \phi_2(x_k), \ldots, \phi_n(x_k)]^T \qquad \Phi = \begin{bmatrix} \phi^T(1) \\ \phi^T(2) \\ \ldots \\ \phi^T(N) \end{bmatrix}$$

Therefore, the following minimisation problem must be solved

$$\begin{cases} \min\limits_{\alpha} e^T e \\ y = \Phi\alpha + e \end{cases}$$

Least squares estimation

Problem:

$$\begin{cases} \min\limits_{\alpha} e^T e \\ y = \Phi\alpha + e \end{cases}$$

The solution $\hat{\alpha}$ of this problem, quadratic with linear constraints, satisfies

$$\Phi^T\Phi\ \hat{\alpha} = \Phi^T y$$

If $\Phi^T\Phi$ is non singular, the solution is unique and given by

$$\hat{\alpha} = (\Phi^T\Phi)^{-1}\Phi^T y \tag{2}$$

As a matter of fact, from $y = \Phi\alpha + e$ (assuming $e = 0$) we have

$$\Phi^T\Phi\ \alpha = \Phi^T y$$

from which, if $(\Phi^T\Phi)^{-1}$ exists, we obtain $\alpha = (\Phi^T\Phi)^{-1}\Phi^T y$, that is eq. (2).

Least squares estimation - Geometric interpretation

The regression model

$$y_k = \phi_1(x_k)\alpha_1 + \phi_2(x_k)\alpha_2 + \ldots + \phi_n(x_k)\alpha_n + e_k$$

for $k = 1, \ldots, N$ can be written as

$$\begin{bmatrix} y_1 \\ y_2 \\ \ldots \\ y_N \end{bmatrix} - \begin{bmatrix} \phi_1(x_1) \\ \phi_1(x_2) \\ \ldots \\ \phi_1(x_N) \end{bmatrix} \alpha_1 - \ldots \begin{bmatrix} \phi_n(x_1) \\ \phi_n(x_2) \\ \ldots \\ \phi_n(x_N) \end{bmatrix} \alpha_n = \begin{bmatrix} e_1 \\ e_2 \\ \ldots \\ e_N \end{bmatrix}$$

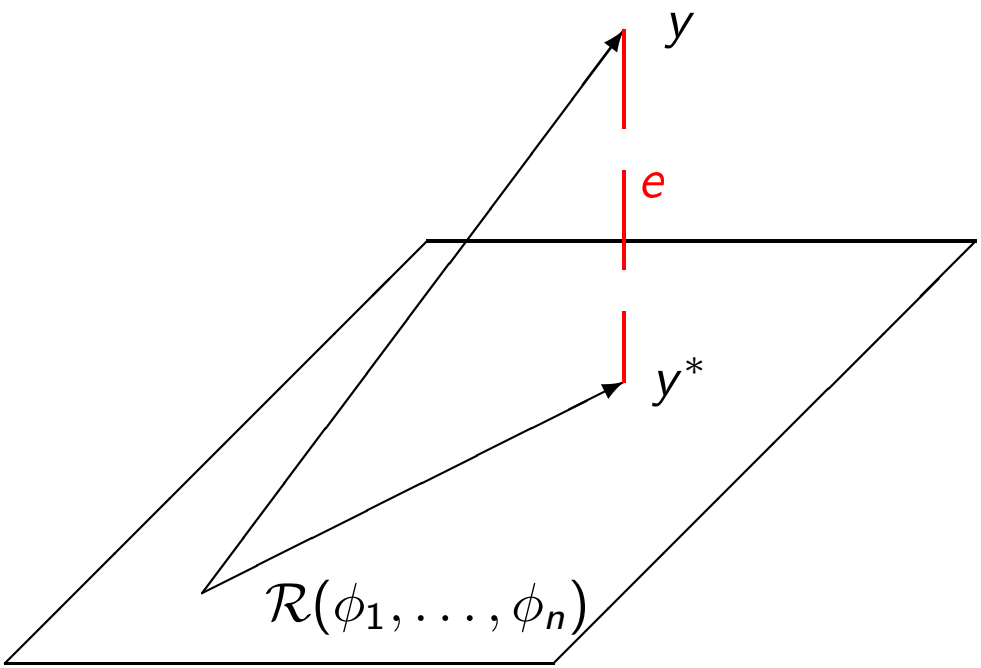
or

$$y - \phi_1\alpha_1 - \phi_2\alpha_2 - \ldots - \phi_n\alpha_n = e$$

Assume that vectors y, $\phi_1, \ldots, \phi_n$ are elements of an Eucledian N-dimensional vector space with norm $\|x\| = x^T x$.

Least squares estimation - Geometric interpretation

If y is the true value, and $y^* = \Phi\ \hat{\alpha}$ its value computed on the basis of the estimated $\hat{\alpha}$ parameters, then the following geometrical interpretation can be obtained:



$$y^* = \Phi\ \hat{\alpha}\ \in\ \mathcal{R}(\phi_1, \ldots, \phi_n)$$

$$\mathcal{R}(\phi_1, \ldots, \phi_n) = \mathcal{R}(\Phi)$$

range space of $\{\phi_1, \ldots, \phi_n\}$

The vector y^* is the orthogonal projection of y on the subspace $\mathcal{R}(\phi_1, \ldots, \phi_n)$.
In this manner the error, defined as $e = y - y^*$, has
minimum norm $\|e\| = \|y - y^*\| \Leftrightarrow$ vectors e and y^* are orthogonal.

Least squares estimation - Geometric interpretation

Since the error $e = (y - y^*)$ is orthogonal to $\mathcal{R}(\phi_1, \ldots, \phi_n)$, then

$$\begin{cases} (y - y^*)^T \phi_1 = 0 \\ \cdots \\ (y - y^*)^T \phi_n = 0 \end{cases}$$

Moreover, since $y^* = \alpha_1 \phi_1 + \ldots + \alpha_n \phi_n$ we can write:

$$\begin{bmatrix} \phi_1^T \phi_1 & \phi_1^T \phi_2 & \cdots & \phi_1^T \phi_n \\ \phi_2^T \phi_1 & \phi_2^T \phi_2 & \cdots & \phi_2^T \phi_n \\ \vdots & & & \\ \phi_n^T \phi_1 & \phi_n^T \phi_2 & \cdots & \phi_n^T \phi_n \end{bmatrix} \alpha = \begin{bmatrix} y^T \phi_1 \\ y^T \phi_2 \\ \vdots \\ y^T \phi_n \end{bmatrix}$$

Therefore

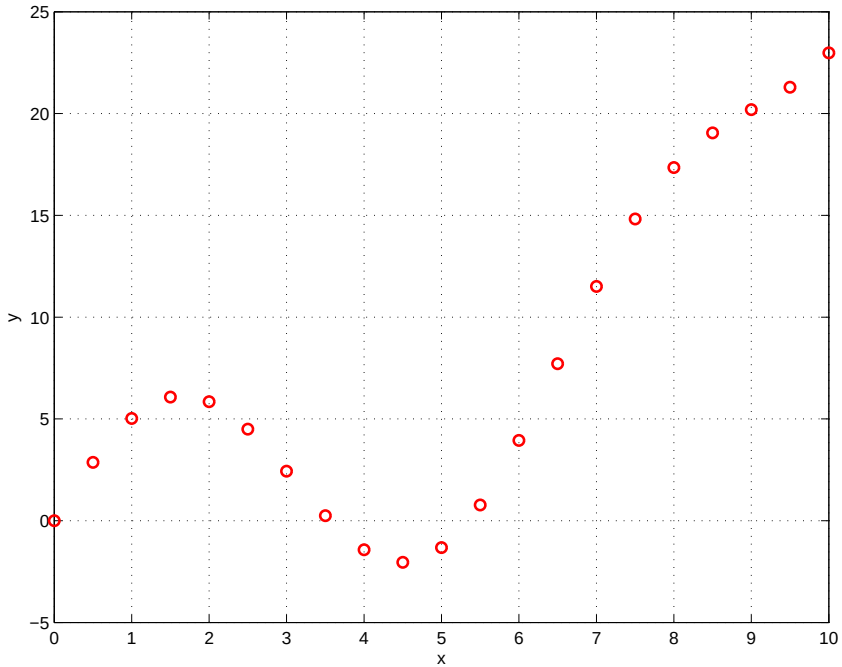
$$\Phi^T \Phi \hat{\alpha} = \Phi^T y \qquad \Rightarrow \qquad \hat{\alpha} = (\Phi^T \Phi)^{-1} \Phi^T y$$

Least squares estimation - Example

Let us assume to have the sequence of $N = 21$ data

$$x = [0.00, 0.50, 1.00, 1.50, 2.00, 2.50, 3.00, 3.50, 4.00, 4.50, 5.00, 5.50, 6.00, 6.50, 7.00, 7.50, 8.00, 8.50, 9.00, 9.50, 10.00]^T$$

$$y = [0.0000, 2.8628, 5.0224, 6.0693, 5.8465, 4.4955, 2.4306, 0.2455, -1.4240, -2.0470, -1.3196, 0.7692, 3.9429, 7.7137, 11.5099, 14.8244, 17.3468, 19.0481, 20.1956, 21.2961, 22.9799]^T$$



Least squares estimation - Example

Also, let assume that the function that interpolates the data is

$$y(x) = ax^3 + bx^2 + cx + d \sin x$$

We want to estimate, given the data $\{x_k, y_k\}$, $k = 1, \ldots, 21$, the unknown parameters a, b, c, d. Therefore:

1) we define

$$\begin{aligned} \alpha &= [a,\ b,\ c,\ d]^T, \\ y &= [y_1, y_2, \ldots, y_{21}]^T \\ \Phi &= \begin{bmatrix} x_1^3 & x_1^2 & x_1 & \sin x_1 \\ x_2^3 & x_2^2 & x_2 & \sin x_2 \\ \ldots & & & \\ x_{21}^3 & x_{21}^2 & x_{21} & \sin x_{21} \end{bmatrix} \end{aligned}$$

2) we use the equation

$$\hat{\alpha} = (\Phi^T \Phi)^{-1} \Phi^T y$$

to estimate the unknown parameters α.

Least squares estimation - Example

Therefore:

$$\Phi = \begin{bmatrix} 0.0000 & 0.0000 & 0.0000 & 0.0000 \\ 0.1250 & 0.2500 & 0.5000 & 0.4794 \\ 1.0000 & 1.0000 & 1.0000 & 0.8415 \\ 3.3750 & 2.2500 & 1.5000 & 0.9975 \\ 8.0000 & 4.0000 & 2.0000 & 0.9093 \\ 15.6250 & 6.2500 & 2.5000 & 0.5985 \\ 27.0000 & 9.0000 & 3.0000 & 0.1411 \\ 42.8750 & 12.2500 & 3.5000 & -0.3508 \\ 64.0000 & 16.0000 & 4.0000 & -0.7568 \\ 91.1250 & 20.2500 & 4.5000 & -0.9775 \\ 125.0000 & 25.0000 & 5.0000 & -0.9589 \\ 166.3750 & 30.2500 & 5.5000 & -0.7055 \\ 216.0000 & 36.0000 & 6.0000 & -0.2794 \\ 274.6250 & 42.2500 & 6.5000 & 0.2151 \\ 343.0000 & 49.0000 & 7.0000 & 0.6570 \\ 421.8750 & 56.2500 & 7.5000 & 0.9380 \\ 512.0000 & 64.0000 & 8.0000 & 0.9894 \\ 614.1250 & 72.2500 & 8.5000 & 0.7985 \\ 729.0000 & 81.0000 & 9.0000 & 0.4121 \\ 857.3750 & 90.2500 & 9.5000 & -0.0752 \\ 1000.0000 & 100.0000 & 10.0000 & -0.5440 \end{bmatrix}$$

and

$$\hat{\alpha} = [0.045, \; -0.300, \; 1.070, \; 5.000]$$

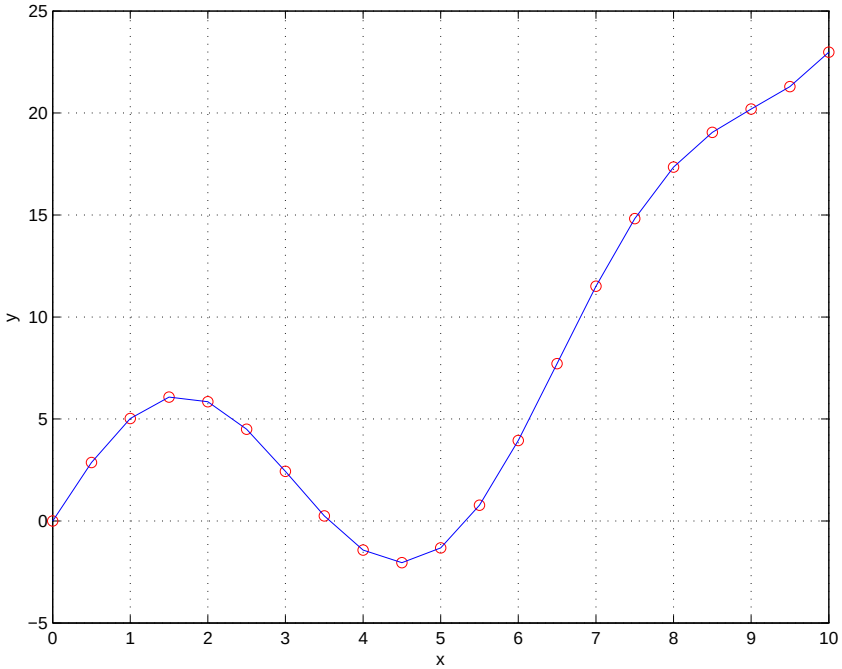
In this case, the parameters have been exactly identified, and

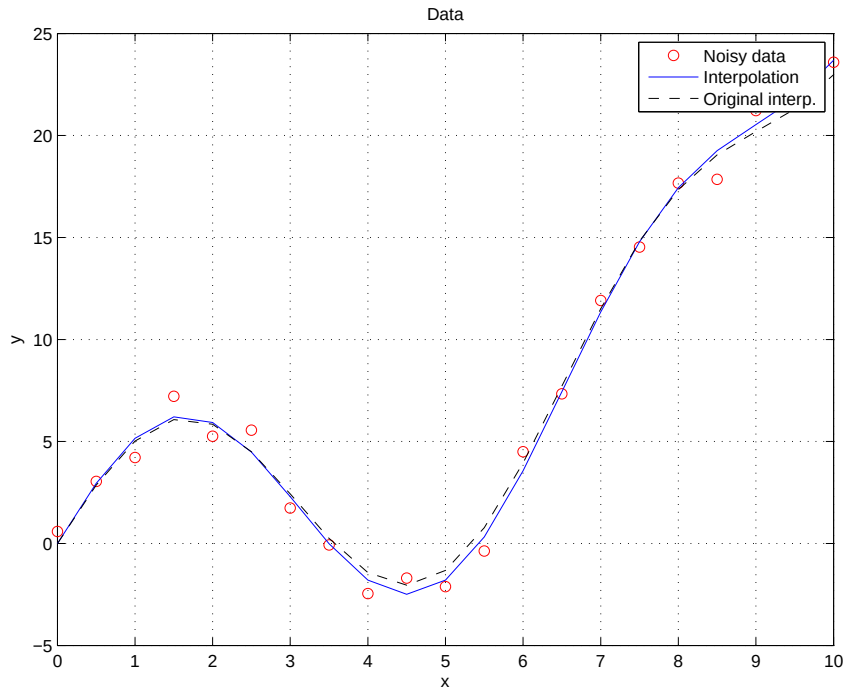
$$V = \sum e_k^2 = \sum (y_k - y_k^*)^2 = 0$$

Least squares estimation - Example

Original data (red) and interpolation (blue) with the function $y(x) = ax^3 + bx^2 + cx + d \sin x$ and with $\hat{\alpha} = [0.045, -0.300, 1.070, 5.000]^T$

Noisy data (random values, in the range $[-2.5, 2.5]$, added to each y_k).
In this case, the estimated parameters are $\hat{\alpha} = [0.05, -0.3467, 1.1132, 5.1583]^T$
and $V = \sum e_k^2 = 11.7293$





Least squares estimation - Example

Note that in the above case the interpolating function
$y(x) = ax^3 + bx^2 + cx + d \sin x$ was known, i.e. only the parameters α had to be estimated, and not the structure of the function itself.

In a more general case, also the interpolating function is not known and must be defined.

For data interpolation, quite often polynomial functions of proper order are used, such as:

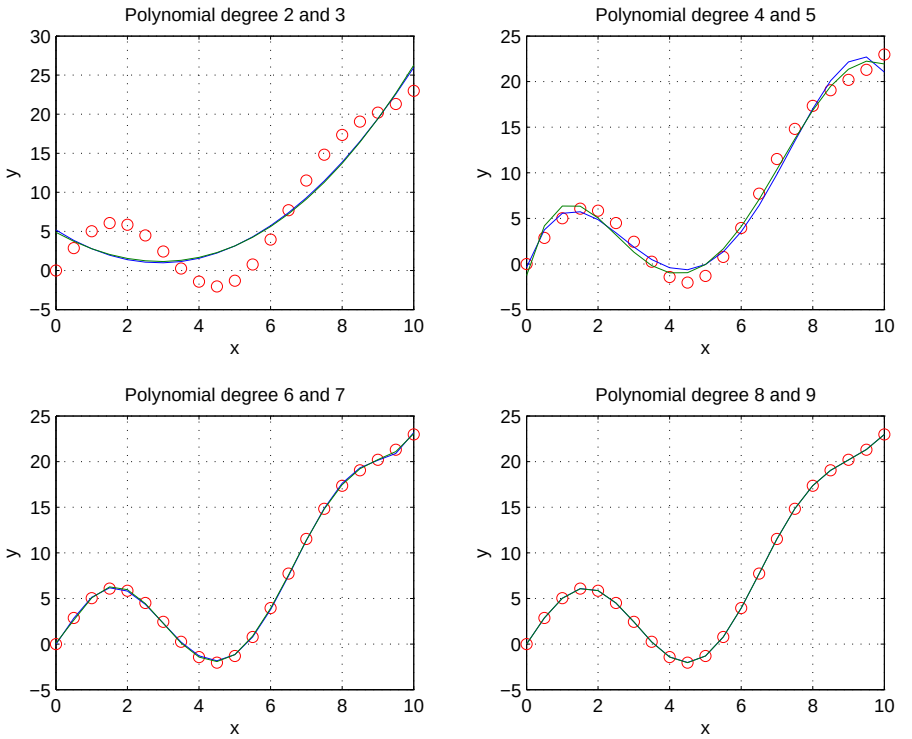
$$y(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$$

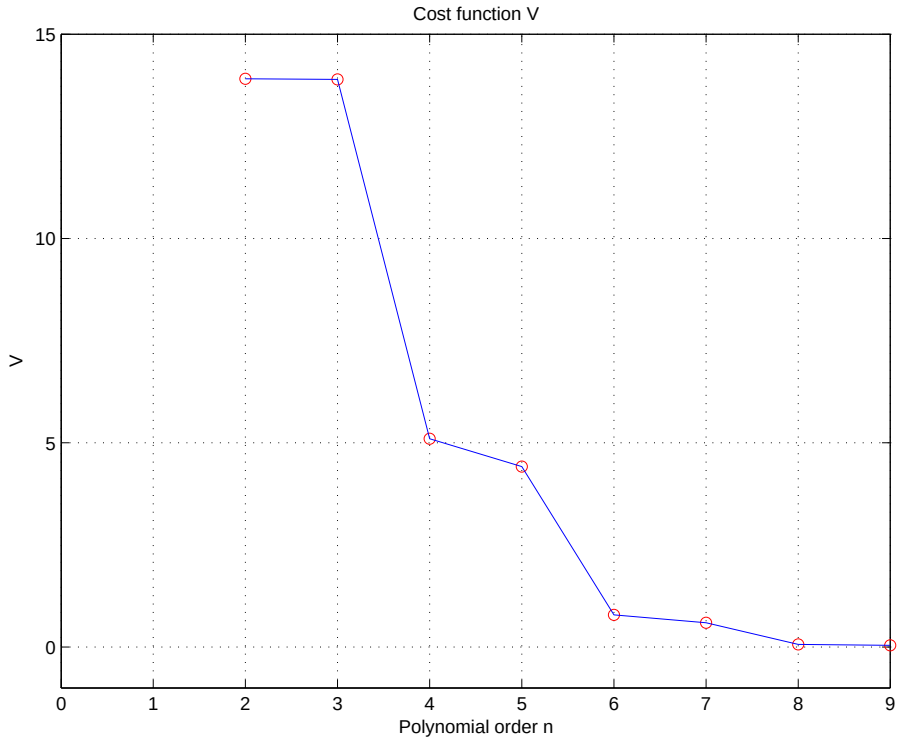
where both the order n and the $n+1$ parameters a_i must be identified.

Note that often a tradeoff between the complexity of the function (that is the order n) and the quality of the result must be defined.

Least squares estimation - Example

Interpolation of the data $\{x_k, y_k\}$ with polynomial functions of order n, with $n \in [2, \ldots, 9]$ (left) and corresponding cost function $V = \sum e_k^2$ (right). Notice that for $n \geq 8$ the cost function is almost null, and therefore the proper value for the polynomial order is 8.





In case of linear dynamic systems expressed as $G(z) = \dfrac{b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-m}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}}$,

it is necessary to define the degrees m, n of the polynomials.

Least squares estimation: Recursive formulation

According to the Least Square technique, an estimation of a set α of parameters is given by

$$\hat{\alpha} = (\Phi^T \Phi)^{-1} \Phi^T y$$

Note that in this manner the vector $\hat{\alpha}$ can be computed only once all the data $y_k, x_k, \ k = 1, \ldots N$ are available (see the previous example).

On the other hand, in many practical application, it is of interest to compute $\hat{\alpha}$ in *real time*, that is updating the current estimation $\hat{\alpha}_k$ when new data $\{y_{k+1}, x_{k+1}\}$ are available. In particular, this is important in control applications, and/or when the parameters are not constant in time.

For this purpose, it is therefore convenient to define a *recursive formulation* of the Least Square estimation algorithm.

Least squares estimation: Recursive formulation

Let us define

- $\Phi(N),\ y(N),\ \alpha(N) \ \rightarrow$ elements relative to N couples of data $\{x_i, y_i\}$

and let assume that a new couple of data $\{x_{N+1}, y_{N+1}\}$ is available.

Then

$$\Phi(N+1) = \begin{bmatrix} \Phi(N) \\ \gamma^T(N+1) \end{bmatrix}, \qquad y(N+1) = \begin{bmatrix} y(N) \\ y_{N+1} \end{bmatrix}$$

$$\gamma^T(N+1) = [\phi_1(x_{N+1}), \dots \phi_n(x_{N+1})]$$

Therefore, the new parameter estimation

$$\hat{\alpha}(N+1) = [\Phi^T(N+1)\Phi(N+1)]^{-1}\Phi^T(N+1)y(N+1)$$

may be rewritten as

$$\hat{\alpha}(N+1) = [\Phi^T(N)\Phi(N) + \gamma(N+1)\gamma^T(N+1)]^{-1}[\Phi^T(N)y(N) + \gamma(N+1)y_{N+1}]$$

Least squares estimation: Recursive formulation

We exploit now the *Inversion Lemma*

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B(C^{-1} + DA^{-1}B)^{-1}DA^{-1}$$

Therefore, by defining:

$$A = \Phi^T\Phi, \qquad B = D^T = \gamma, \qquad C = 1$$

we have:

$$[\Phi^T\Phi + \gamma\gamma^T]^{-1} = (\Phi^T\Phi)^{-1} - (\Phi^T\Phi)^{-1}\gamma[1 + \gamma^T(\Phi^T\Phi)^{-1}\gamma]^{-1}\gamma^T(\Phi^T\Phi)^{-1}$$

from which it follows:

$$\begin{aligned}\hat{\alpha}(N+1) &= \hat{\alpha}(N) - k(N+1)\gamma^T\hat{\alpha}(N) + k(N+1)y_{N+1} \\ &= \hat{\alpha}(N) + k(N+1)[y_{N+1} - \gamma^T\hat{\alpha}(N)]\end{aligned}$$

where

$$k(N+1) = (\Phi^T\Phi)^{-1}\gamma[1 + \gamma^T(\Phi^T\Phi)^{-1}\gamma]^{-1}$$

Least squares estimation: Recursive formulation

Moreover, in order to have also $k(N+1)$ in a recursive formulation, we define

$$P(N) = [\Phi^T(N)\Phi(N)]^{-1} \tag{3}$$

Then

$$k(N+1) = P(N)\gamma[1+\gamma^T P(N)\gamma]^{-1}$$

and

$$\begin{aligned} P(N+1) &= (\Phi^T\Phi + \gamma\gamma^T)^{-1} \\ &= P(N) - P(N)\gamma[1+\gamma^T P(N)\gamma]^{-1}\gamma^T P(N) \\ &= [I - k(N+1)\gamma^T]P(N) \end{aligned}$$

Finally, the recursive formulation of the LS algorithm is

$$\begin{cases} \hat{\alpha}(N+1) &= \hat{\alpha}(N) + k(N+1)[y_{N+1} - \gamma^T\hat{\alpha}(N)] \\ k(N+1) &= P(N)\gamma[1+\gamma^T P(N)\gamma]^{-1} \\ P(N+1) &= [I_n - k(N+1)\gamma^T]P(N) \end{cases}$$

Least squares estimation: Recursive formulation

By analyzing the expression of $\hat{\alpha}(N+1)$

$$\hat{\alpha}(N+1) = \hat{\alpha}(N) + k(N+1)[y_{N+1} - \gamma^T \hat{\alpha}(N)]$$

it may be noticed that the new value is obtained from the previous one $\hat{\alpha}(N)$ by adding a correction term proportional to $[y_{N+1} - \gamma^T \hat{\alpha}(N)]$.

This is the difference between the new measured value of y (i.e. y_{N+1}) and y^*, that is its (one step) prediction based on the data available up to step N

$$\begin{aligned} y^* &= \gamma^T \hat{\alpha}(N) \\ &= [\phi_1(x_{N+1}), \dots \phi_n(x_{N+1})]^T \begin{bmatrix} \hat{\alpha}_1(N) \\ \hat{\alpha}_2(N) \\ \dots \\ \hat{\alpha}_n(N) \end{bmatrix} \end{aligned}$$

Least squares estimation: Recursive formulation

Initialization of the algorithm

For the recursive algorithm, an initialisation problem exists, since matrix $P(N) = (\Phi^T\Phi)^{-1} \in \mathbb{R}^{n\times n}$ may be non singular only for values $N \geq n$.

Necessary condition. Matrix $P(N)$ may be non singular only if $N \geq n$; in this case, Φ may be full column rank (Φ is a $N \times n$ matrix).

Therefore, a value $N_0 > n$ should be chosen such that

$$\begin{aligned} P(N_0) &= [\Phi^T(N_0)\Phi(N_0)]^{-1} \\ \hat{\alpha}(N_0) &= [\Phi^T(N_0)\Phi(N_0)]^{-1}\Phi^T(N_0)\, y(N_0) \end{aligned}$$

However, it is possible to use the recursive algorithm starting with the first pair of data $\{x_1, y_1\}$ by admitting an arbitrarily small error. This is possible by defining

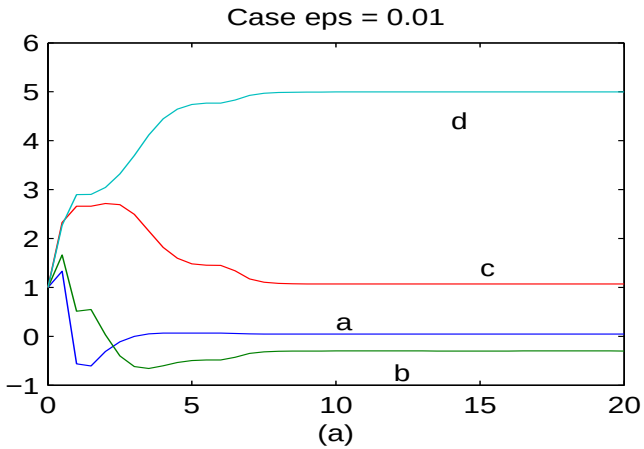
$$P(N) = [P_0^{-1} + \Phi^T(N)\Phi(N)]^{-1}, \qquad P(0) = P_0 = \frac{1}{\epsilon} I_n, \qquad \epsilon \ll 1 \qquad (4)$$

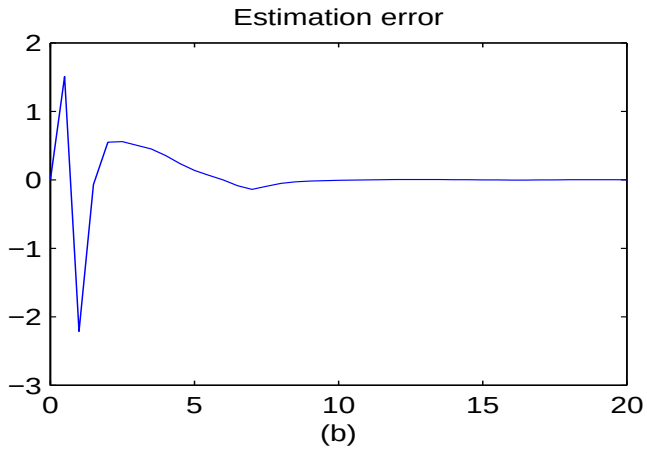
It is clear that (4) is always invertible, and that it differs by an arbitrary small quantity from (3) (I_n is a $n \times n$ identity matrix).

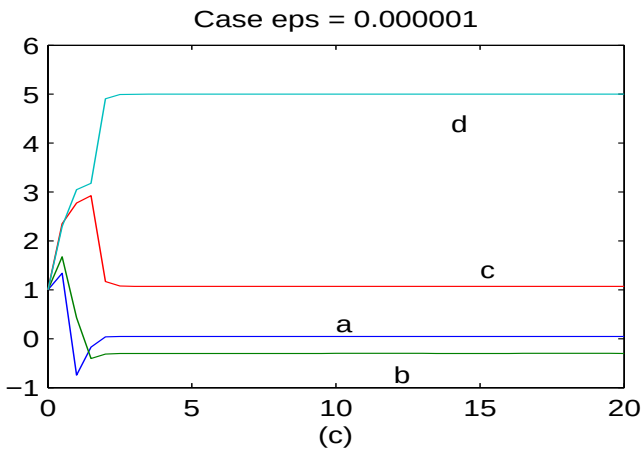
Least squares estimation: Recursive formulation - example

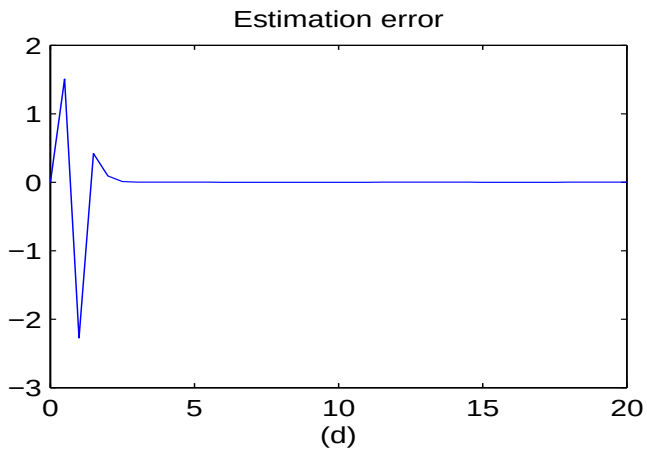
Recursive estimation of the parameters a, b, c, d of the previous example.

Two different initialisation of the algorithm are used: $\epsilon = 10^{-2}$ and $\epsilon = 10^{-6}$, and initial value for $\hat{\alpha}(0) = [1, 1, 1, 1]^T$, no noise.





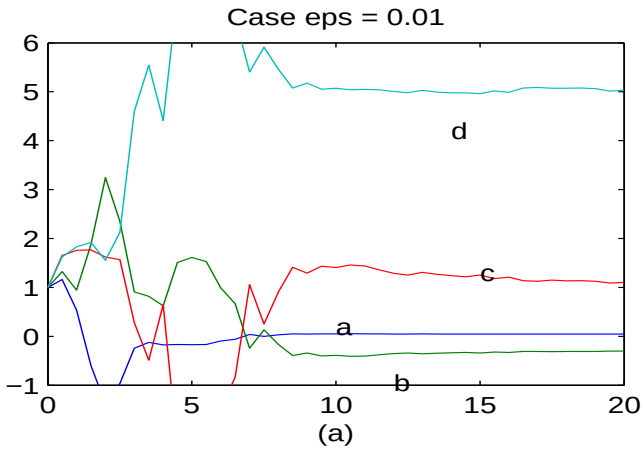


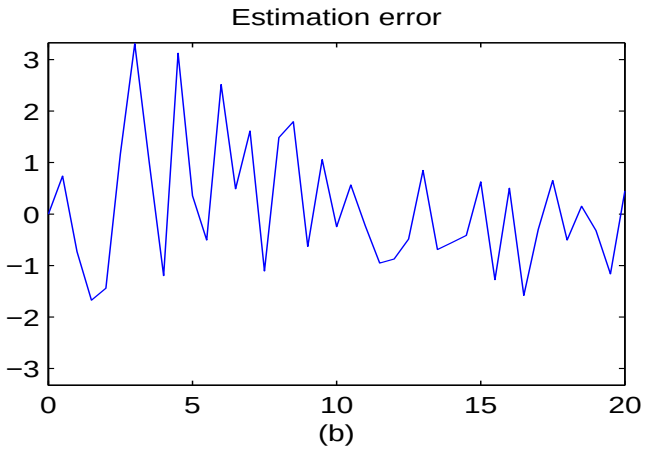


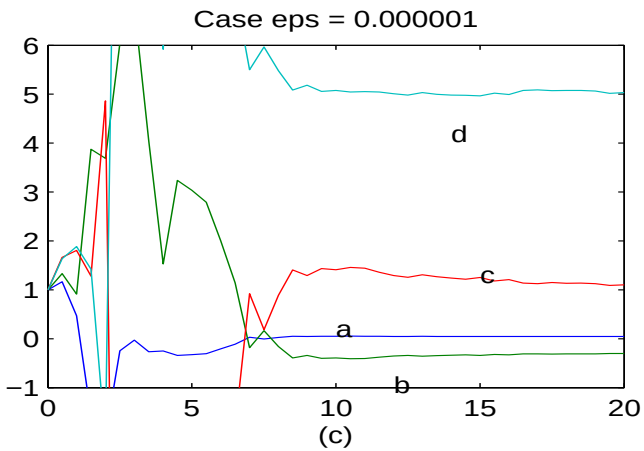
Least squares estimation: Recursive formulation - example

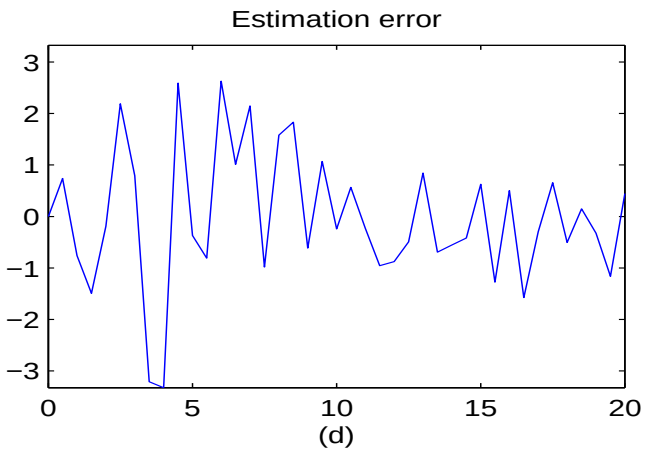
Recursive estimation of the parameters a, b, c, d of the previous example.

Two different initialisation of the algorithm are used: $\epsilon = 10^{-2}$ and $\epsilon = 10^{-6}$, and initial value for $\hat{\alpha}(0) = [1, 1, 1, 1]^T$, with noise.









Least squares estimation: Time-varying parameters

In the previous expression of the RLS algorithm , the unknown parameters α have been assumed as constant in time. Therefore, the cost function to be minimized has been defined as

$$V = \sum_{k=1}^{N} e_k^2,$$

where all the samples e_k have the same "importance", i.e they have the same (unit) cost. A more general expression of the cost function is

$$V = \sum_{k=1}^{N} w_k e_k^2,$$

where w_k is a proper (non constant) weight to be defined according to some proper criterion.

As a matter of fact, in many practical control applications (some of) the parameters of the controlled plant may vary in time, with variation that can be considered "slow" with respect to the dynamics of the plant.

Least squares estimation: Time-varying parameters

In these cases, in order to have a better estimation of the parameters, it is necessary to use a cost function V that gives more importance to recent data with respect to old ones.

In other words, it is necessary to define an algorithm with a *finite lenght* memory.

The "length" of the period must be properly tuned on the basis of the velocity of variation of the parameters.

A solution is to define the cost function as

$$V = \sum_{k=1}^{N} \beta^{N-k} e_k^2, \qquad \rightarrow \qquad \min_{\alpha_i} \sum_{k=1}^{N} \beta^{N-k} e_k^2, \qquad 0 < \beta \leq 1$$

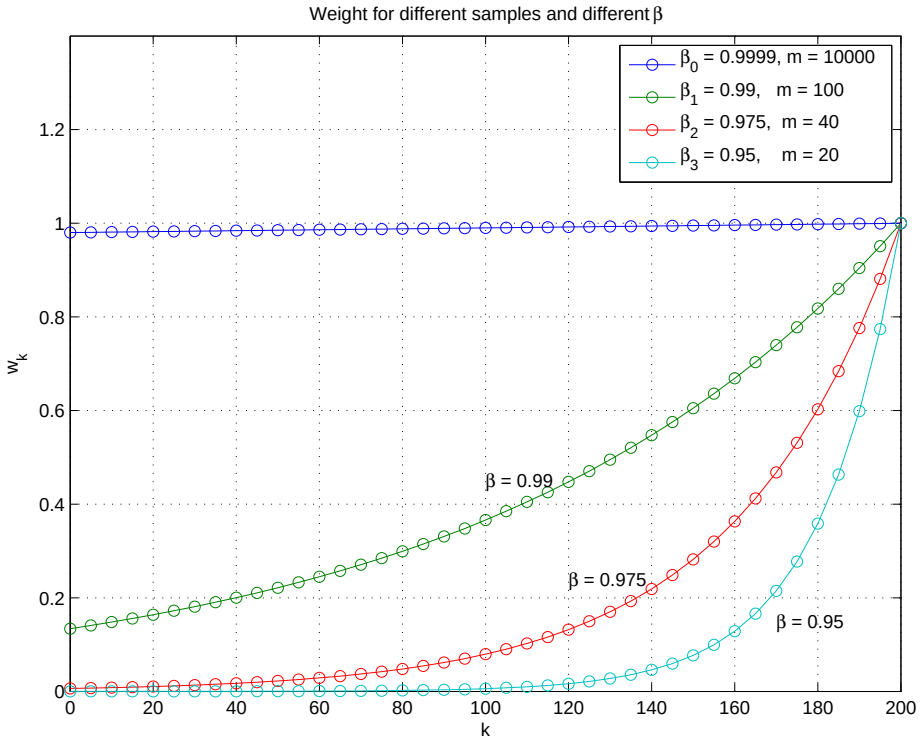
Least squares estimation: Time-varying parameters

$$V = \sum_{k=1}^{N} \beta^{N-k} e_k^2, \qquad 0 < \beta \leq 1$$

- The parameter β is called the *forgetting factor*
- With $\beta = 1$, the standard case is obtained
- Typically $0.95 \leq \beta \leq 0.99$
- The "equivalent" number of samples considered in the RLS algorithm is given by

$$m = \frac{1}{1-\beta}$$

$\beta = 0.99 \rightarrow m = 100,$
$\beta = 0.95 \rightarrow m = 20$



Least squares estimation: Time-varying parameters

$$V = \sum_{k=1}^{N} \beta^{N-k} e_k^2, \qquad 0 < \beta \leq 1$$

- In this way, the most recent data ($k = N$) has unit weight, while data of previous p steps, corresponding to time instants in which the values of the parameters may be different from the current ones, are weighted with $\beta^p < 1$.
- "Low" values ($\beta = 0.95$) are used when there are fast variations in the parameters, viceversa "high" values ($\beta = 0.99$) are adopted for slow varying parameters.
- With low values for β, the tracking of the parameters is better, but on the other side there is a larger variance of the estimated parameters (possible problems with noise).

Least squares estimation: Time-varying parameters

With the introduction of the forgetting factor, the RLS algorithm becomes:

$$\begin{cases} \hat{\alpha}(N+1) &= \hat{\alpha}(N) + k(N+1)[y_{N+1} - \gamma^T \hat{\alpha}(N)] \\ k(N+1) &= P(N)\gamma[\beta + \gamma^T P(N)\gamma]^{-1} \\ P(N+1) &= \frac{1}{\beta}[I - k(N+1)\gamma^T]P(N) \end{cases}$$

At each step, the matrix P is *multiplied* by a factor $1/\beta > 1$, and therefore the weight vector k is always non zero.

This is justified since it may be verified that, without the multiplication by $1/\beta$, $\|P\| \to 0$ for $N \to \infty$. In this case, also $\|k\| \to 0$ and then $\hat{\alpha}(N+1) = \hat{\alpha}(N)$.

In these conditions, the algorithm is non sensitive to estimation errors $e(k) = [y_{k+1} - \gamma^T \hat{\alpha}(k)]$, generated by parameters variations.

If matrix P is multiplied at each step by a factor $1/\beta > 1$, its norm (and therefore the value of k) is maintained non null, and the updated value $\hat{\alpha}(N+1)$ takes into consideration possible estimation errors generated by changes in the parameters.

Least squares estimation: Time-varying parameters

Another version of the RLS algorithm achieving the forgetting property (based on an *addition* operation) is the following

$$\begin{cases} \hat{\alpha}(N+1) &= \hat{\alpha}(N) + k(N+1)[y_{N+1} - \gamma^T \hat{\alpha}(N)] \\ k(N+1) &= P(N)\gamma[r_2 + \gamma^T P(N)\gamma]^{-1} \\ P(N+1) &= \mathbf{R}_1 + [I - k(N+1)\gamma^T]P(N) \end{cases}$$

where usually $\mathbf{r}_2 = 1$ and $R_1 = q\ I$, with $q \in [10^{-4} \div 10^{-2}]$.

The three algorithms, i.e. the standard RLS, with a multiplicative β factor or in the additive version, coincide for $r_2 = 1$, $q = 0$ e $\beta = 1$.

Least squares estimation: Time-varying parameters

A further improvement for the RLS algorithm with forgetting factor consists in computing the value of the parameter β as a function of the "variability" of the system's parameters.

Indeed, as already pointed out, "small" values of β are better when parameters changes rapidly, while for slow changing (or even constant) parameters an "high" value is preferable.

For example, β could be computed as follows

$$
\begin{aligned}
&\text{if } |e(k)| > \bar{e} \\
&\qquad \text{then} \quad \beta(k) = 0.95 \\
&\qquad \text{else} \quad \beta(k) = 1 - \lambda \, [1 - \beta(k-1)] \\
&\text{endif}
\end{aligned}
$$

where $e(k) = [y_{k+1} - \gamma^T \hat{\alpha}(k)]$ is the prediction error, and $\lambda < 1$ a proper parameter used to tune the transition between the "low" value (0.95) and 1.

Application to linear dynamic systems

Let us consider a dynamic systems expressed by the transfer function

$$G(z) = \frac{b_1 z^{-1} + b_2 z^{-2} + \cdots + b_n z^{-n}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} = \frac{Y(z)}{U(z)}$$

The output, at the generic time instant k, is given by

$$\begin{aligned} y(k) \quad = \quad & -a_1 y(k-1) - a_2 y(k-2) - \cdots - a_n y(k-n) + \\ & + b_1 u(k-1) + b_2 u(k-2) + \cdots + b_n u(k-n) + e(k) \end{aligned}$$

Assume that the parameters $a_i,\ b_i,\ i = 1, \ldots, n$ are not known.

Application to linear dynamic systems

In this case, in $n+N$ pairs of data $u(k), y(k)$ are available, let us define:

$$y = \begin{bmatrix} y(n+1) \\ y(n+2) \\ \dots \\ y(n+N) \end{bmatrix}, \qquad e = \begin{bmatrix} e(n+1) \\ e(n+2) \\ \dots \\ e(n+N) \end{bmatrix}, \qquad \alpha = [-a_1, \dots, -a_n, b_1, \dots, b_n]^T$$

$$\Phi = \left[\begin{array}{cccc|ccc} y(n) & y(n-1) & \cdots & y(1) & u(n) & \cdots & u(1) \\ y(n+1) & y(n) & \cdots & y(2) & u(n+1) & \cdots & u(2) \\ y(n+2) & y(n+1) & \cdots & y(3) & u(n+2) & \cdots & u(3) \\ \vdots & & & & \vdots & & \\ y(n+N-1) & y(n+N-2) & \cdots & y(N) & u(n+N-1) & \cdots & u(N) \end{array}\right]$$

Note that Φ is a $N \times 2n$ matrix.

Application to linear dynamic systems

As before, the estimation of the unknown parameters is given by

$$\hat{\alpha} = (\Phi^T \Phi)^{-1} \Phi^T y$$

where

$$\Phi^T \Phi = \begin{bmatrix} A & B \\ B^T & C \end{bmatrix}$$

with

$$A = A^T = \begin{bmatrix} \sum_{k=n}^{N+n-1} y^2(k) & \sum_{k=n}^{N+n-1} y(k)y(k-1) & \cdots & \sum_{k=n}^{N+n-1} y(k)y(k-n+1) \\ & \sum_{k=n}^{N+n-2} y^2(k) & \cdots & \sum_{k=n}^{N+n-2} y(k)y(k-n+2) \\ \vdots & & \ddots & \\ & & & \sum_{k=1}^{N} y^2(k) \end{bmatrix}$$

Application to linear dynamic systems

and

$$B = \begin{bmatrix} \sum_{k=n}^{N+n-1} y(k)u(k) & \cdots & \sum_{k=n}^{N+n-1} y(k)u(k-n+1) \\ \sum_{k=n}^{N+n-2} y(k)u(k+1) & \cdots & \sum_{k=n}^{N+n-2} y(k)u(k-n+2) \\ \sum_{k=1}^{N} y(k)u(n+k-1) & \cdots & \sum_{k=1}^{N} y(k)u(k) \end{bmatrix}$$

$$C = C^T = \begin{bmatrix} \sum_{k=n}^{N+n-1} u^2(k) & \cdots & \sum_{k=n}^{N+n-1} u(k)u(k-n+1) \\ \vdots & \ddots & \\ & & \sum_{k=1}^{N} u^2(k) \end{bmatrix}$$

Application to linear dynamic systems

Moreover

$$\Phi^T y = \begin{bmatrix} p \\ q \end{bmatrix}$$

with

$$p = \begin{bmatrix} \sum_{k=n+1}^{N+n} y(k)y(k-1) \\ \sum_{k=n+1}^{N+n} y(k)y(k-2) \\ \vdots \\ \sum_{k=n+1}^{N+n} y(k)y(k-n) \end{bmatrix} \qquad q = \begin{bmatrix} \sum_{k=n+1}^{N+n} y(k)u(k-1) \\ \sum_{k=n+1}^{N+n} y(k)u(k-2) \\ \vdots \\ \sum_{k=n+1}^{N+n} y(k)u(k-n) \end{bmatrix}$$

Application to linear dynamic systems: implementation aspects

The RLS algorithm

$$\begin{cases} \hat{\alpha}(N+1) &= \hat{\alpha}(N) + k(N+1)[y_{N+1} - \gamma^T \hat{\alpha}(N)] \\ k(N+1) &= P(N)\gamma[\beta + \gamma^T P(N)\gamma]^{-1} \\ P(N+1) &= \frac{1}{\beta}[I - k(N+1)\gamma^T]P(N) \end{cases}$$

gives at each iteration the estimation $\hat{\alpha}$ of the plant's parameters.

However, there are some aspects in the implementation of this algorithms that have to be properly taken into account.

Application to linear dynamic systems: implementation aspects

The issues to be considered are:

- The input signal has to be adequately "exciting" for the system dynamics

  For the limit case of constant input, the RLS algorithm may only estimate the static gain of the process: during the estimation phases, the input signal must excite all the dynamics of the process (*persistently exciting signals*)
- Initialization of the algorithm (as discussed)
- Matrix P, for numerical reasons could result not symmetric and positive definite. Therefore, it could be defined as a factorization:

  $$P = UDU^T$$

  with U upper triangular and D diagonal, or

  $$P = SS^T$$

  being S the square root of P

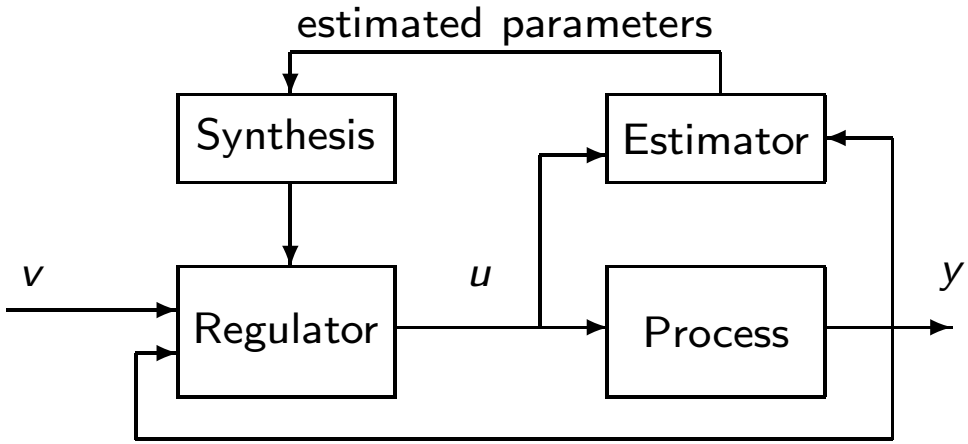
Application to linear dynamic systems: implementation aspects

- Wind-up problem of the matrix P.

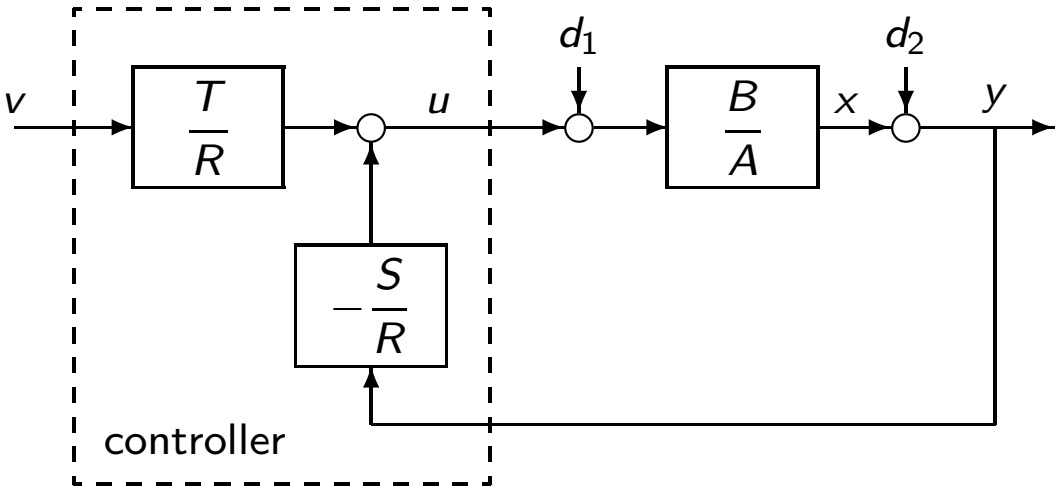
  Since with the forgetting factor and in case of good estimation of the parameters, it results $P(N+1) \approx P(N)/\beta$, then its norm grows exponentially in time. Since $\hat{\alpha}$ is good, the difference $[y_{N+1} - \gamma^T \hat{\alpha}(N)]$ is practically null, and the fact that the norm of P grows is not important.
  On the other hand, if a parameter or the reference signal changes, the RLS algorithm generates wrong estimations, with a "burst" behaviour due to the high value of **P** (and then of k).
  As already discussed, a *variable* forgetting factor can be adopted (equal to 1 when the estimation is good). As an alternative, the RLS algorithm could be deactivated when the prediction error is lower than a given threshold.

ST regulators by poles/zeros assignment

We analyze now the design of a ST regulator, according to the scheme



The controller design is based on the poles/zeros assignment, based on the scheme



where the polynomials S, T, R have to be defined according to some given specifications.

ST regulators by poles/zeros assignment

Given a desired transfer function to be obtained

$$G_m(z) = \frac{Y(z)}{V(z)} = \frac{B_m(z)}{A_m(z)} \quad \text{or} \quad G_m(z) = \frac{A_0(z)B_m(z)}{A_0(z)A_m(z)}$$

the design equation is

$$\frac{B\,T}{A\,R + B\,S} = \frac{A_0 B_m}{A_0 A_m} = G_m(z)$$

with

$$B = B^+ B^- \qquad B^- \text{ unstable zeros } \rightarrow \; B_m = B^- B'_m$$

and then

$$\begin{aligned} A\,R' + B^- S &= A_0 A_m \\ T &= A_0 B'_m \end{aligned}$$

with $R = B^+ R'$.

ST regulators by poles/zeros assignment

Since the design equations have to be computed in real time (because of variations of the parameters in A, B^-), it is necessary to implement the algorithm in a computationally efficient way.

For this purpose, if possible, it is better *not to factorize the polynomial B*.

This can be obtained in two ways:

- E1 eliminating all the zeros, that is considering $B^+ = B$ and $B^- = 1$
- E2 leaving all the zeros, that is assuming $B^+ = 1$ and $B^- = B$

Note that the first method E1 can be applied only if the transfer function $G_p(z)$ of the plant is minimum phase (the zeros are within the unit circle), while E2 can be used in any case.

ST regulators by poles/zeros assignment

Algorithm E1 (zeros are cancelled: $B^+ = B$ and $B^- = 1$)

Given A_m, $B_m(=1)$, and A_0, at each iteration:

- the parameters $\hat{A}$ and $\hat{B}$ are computed with the RLS method
- the equation

$$\hat{A}\,R' + S = A_0 A_m$$

  is solved with respect to S and R'
- the value of T is computed as

$$T = kA_0, \qquad k = A_m(1)$$

- the new control value u is computed according to

$$Ru = Tv - Sy$$

  where $R = B^+\,R' = \hat{B}\,R'$

ST regulators by poles/zeros assignment

Algorithm E2 (zeros are not cancelled: $B^+ = 1$ and $B^- = B$)

Given A_m, $B_m (= B^- = B)$ and A_0, at each iteration:

- the parameters $\hat{A}$ and $\hat{B}$ are computed with the RLS method
- the equation
$$\hat{A}\,R + \hat{B}\,S = A_0\,A_m$$
is solved with respect to S and R
- the new control value u is computed according to
$$Ru = Tv - Sy, \qquad \text{with} \quad T = k = \frac{A_m(1)}{\hat{B}(1)}$$

Note that in this manner we get

$$G_m(z) = \frac{k\hat{B}(z)}{A_m(z)}$$

- Variations in the parameters affect the controlled dynamics (i.e. $G_m(z)$)
- This algorithm can be applied even if $G_p(z)$ is non minimum phase.

ST regulators by poles/zeros assignment

The above is the "explicit" formulation of the control design procedure. Another version is the "implicit" one, where the parameters of the controller (not of the process) are estimated in real time. From

$$A\,R'y + B^-\,Sy = A_0\,A_m y$$

$$Ay = Bu$$

we get

$$A_0\,A_m y = B\,R'u + B^-\,Sy = B^-(Ru + Sy)$$

If $B^- = 1$, then this equation is linear in the parameters of the R, S polynomials, and therefore the RLS algorithm can be used.

The overall procedure is simpler but can be applied only for minimum phase systems, since all the zeros must be cancelled ($B^- = 1$).

ST regulators by poles/zeros assignment

Algorithm I1 (zeros are cancelled)

Given A_m, B_m and A_0, at each iteration:

- the parameters $\hat{R}$ and $\hat{S}$ of the model

$$A_0 A_m y = Ru + Sy$$

  are estimated with the RLS method
- the new control input u is computed by

$$\hat{R}u = Tv - \hat{S}y$$

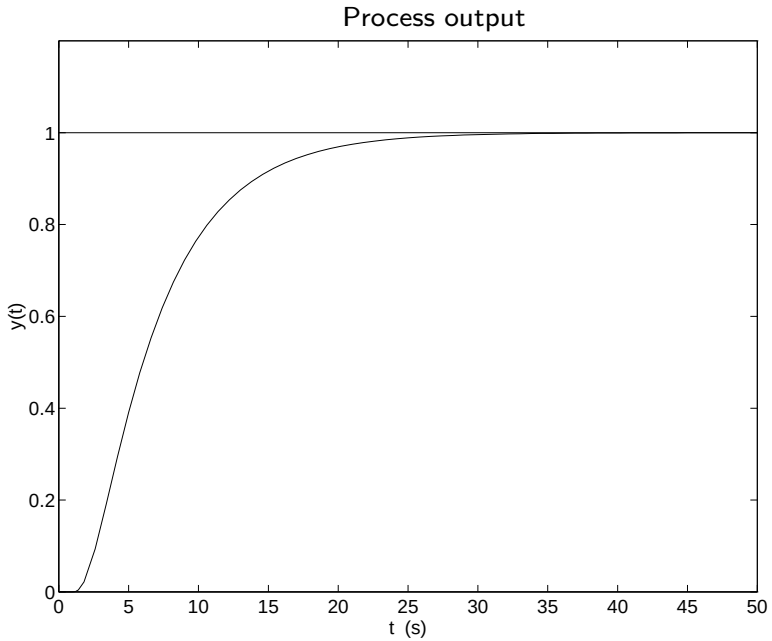
  with $T = k\,A_0$, $k = A_m(1)$ (as in E1).

ST regulators by poles/zeros assignment: Example

Let us consider a system described by

$$G(s) = \frac{0.4}{(s+1)(s+2)(s+0.2)} = \frac{Y(s)}{U(s)}$$

Open loop response to a unit step



Two design procedures will be examined:

- algorithm E2 (zeros not cancelled)
- algorithm I1 (zeros cancelled)

ST regulators by poles/zeros assignment: Example

Let us assume that the following discrete-time model of the process is given

$$G(z) = z^{-1}\frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} = \frac{Y(z)}{U(z)} \quad \rightarrow \quad y(k) = -a_1 y(k-1) + b_0 u(k-1) + b_1 u(k-2)$$

with a sampling period $T = 1\ s$.

Moreover, define the desired polynomial $A_m(z)$ as

$$\begin{aligned} A_m(z) &= 1 - 2e^{-\delta\omega_n T}\cos\omega_n T\sqrt{1-\delta^2}z^{-1} + e^{-2\delta\omega_n T}z^{-2} \\ &= 1 + c_1 z^{-1} + c_2 z^{-2} \end{aligned}$$

in which the values $\delta = 0.7, \quad \omega_n = 0.25\ rad/s$ are considered.

ST regulators by poles/zeros assignment: Example – E2

ST controller according to algorithm E2 - zeros NOT canceled

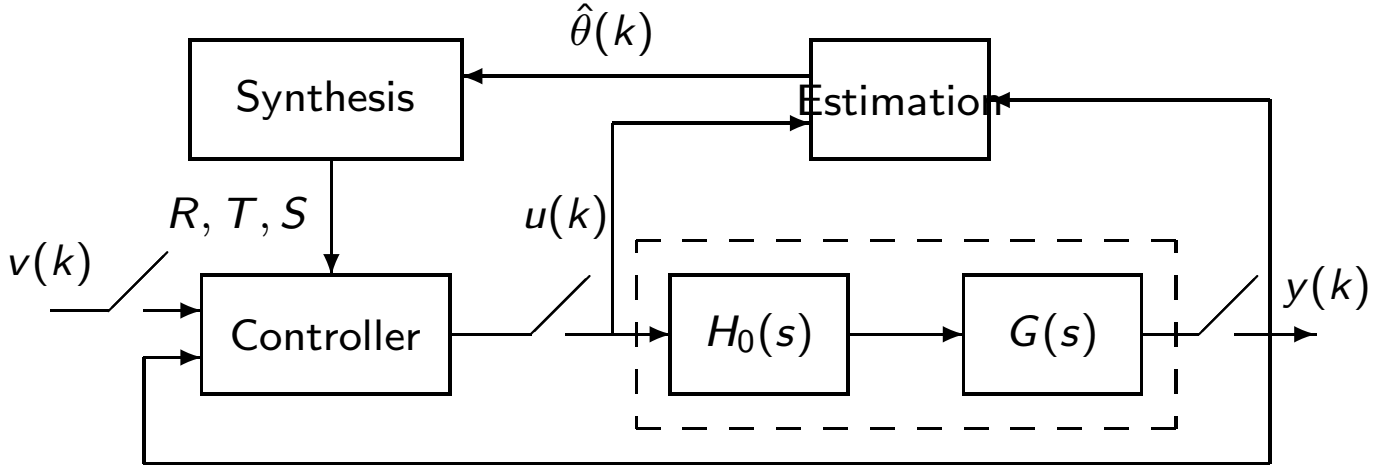
The parameters a_1, b_0, b_1 of the model

$$y(k) = -a_1 y(k-1) + b_0 u(k-1) + b_1 u(k-2)$$

are estimated. Consider

$$A_0(z) = 1, \qquad R(z) = 1 + r_1 z^{-1}, \qquad S(z) = s_0$$

Block scheme of the algorithm E2



ST regulators by poles/zeros assignment: Example – E2

The design equation is

$$\begin{aligned} (1+\hat{a}_1 z^{-1})(1+r_1 z^{-1}) + z^{-1}(\hat{b}_0+\hat{b}_1 z^{-1})s_0 &= 1+c_1 z^{-1}+c_2 z^{-2} \\ \rightarrow \quad 1+(r_1+\hat{a}_1+\hat{b}_0 s_0)z^{-1}+(\hat{a}_1 r_1+\hat{b}_1 s_0)z^{-2} &= 1+c_1 z^{-1}+c_2 z^{-2} \end{aligned}$$

to be solved with respect to r_1 and s_0:

$$\begin{aligned} s_0 &= (c_2-\hat{a}_1 c_1+\hat{a}_1^2)/(\hat{b}_1-\hat{a}_1\hat{b}_0) \\ r_1 &= c_1-\hat{a}_1-s_0\hat{b}_0 \end{aligned}$$

The control equation is

$$(1+r_1 z^{-1})u(k) = K\, v(k) - s_0 y(k)$$

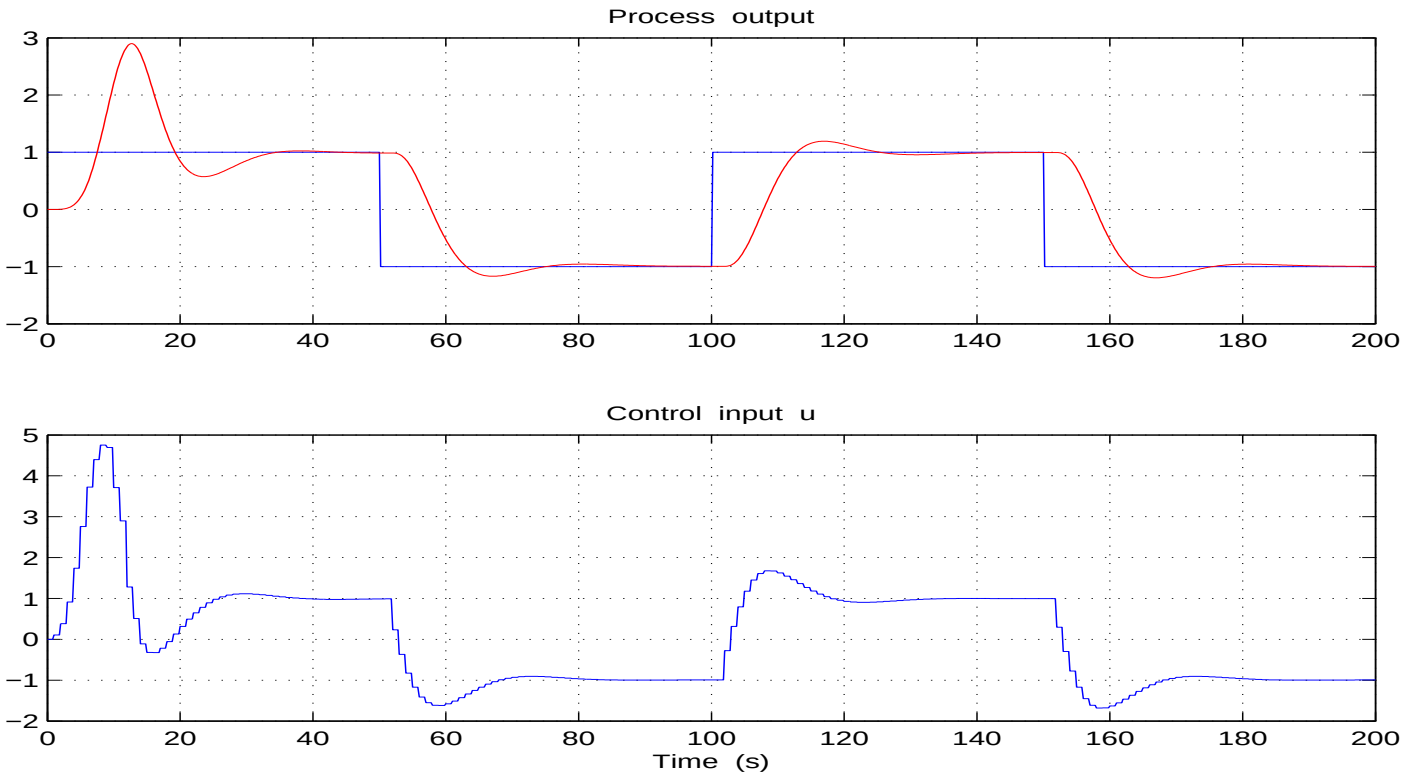
from which

$$u(k) = K\, v(k) - s_0 y(k) - r_1 u(k-1)$$

with $K = (1+c_1+c_2)/(\hat{b}_0+\hat{b}_1)$ (unit gain for $G_m(z)$).

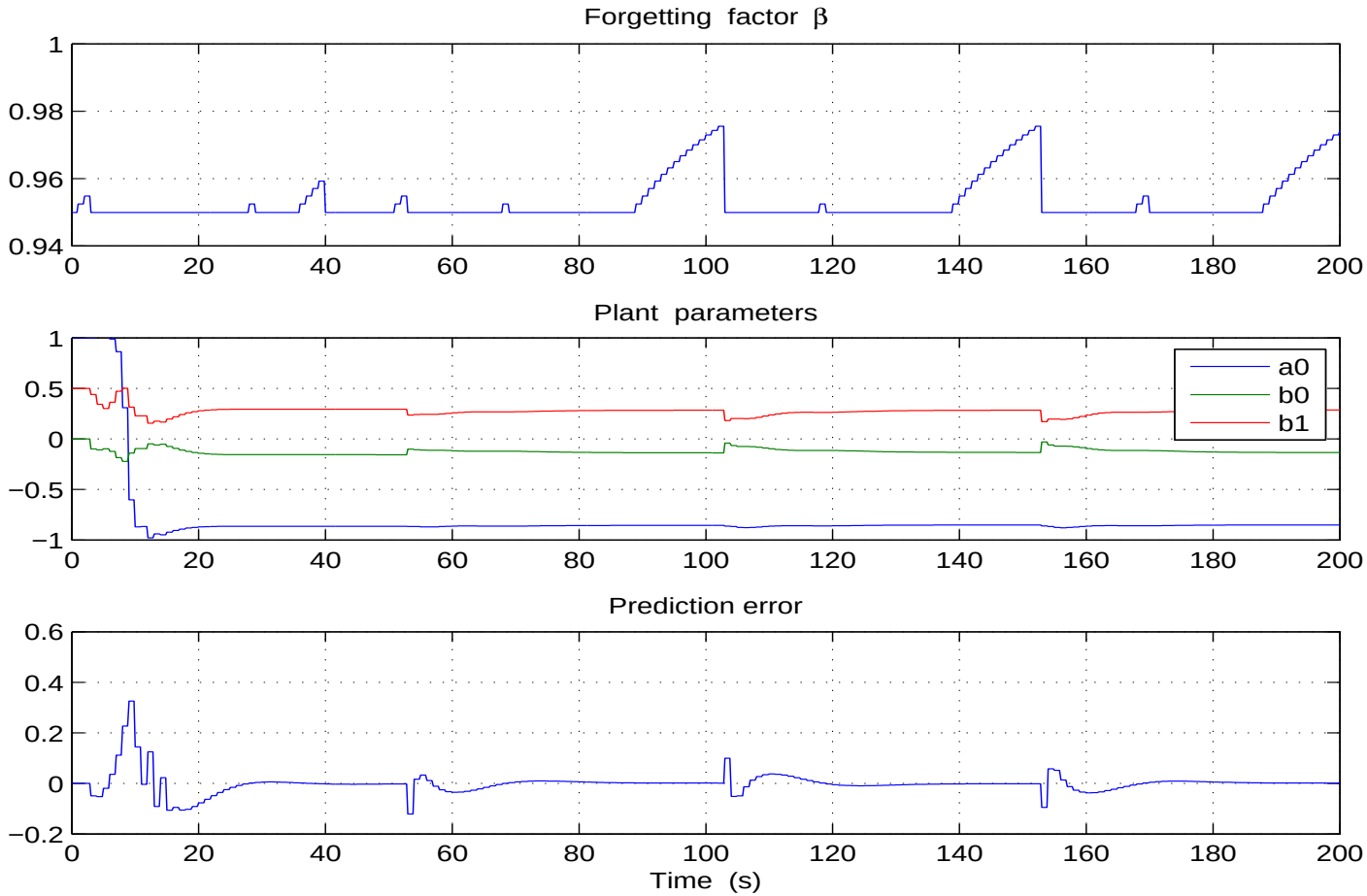
ST regulators by poles/zeros assignment: Example – E2

System response with controller E2



ST regulators by poles/zeros assignment: Example – E2

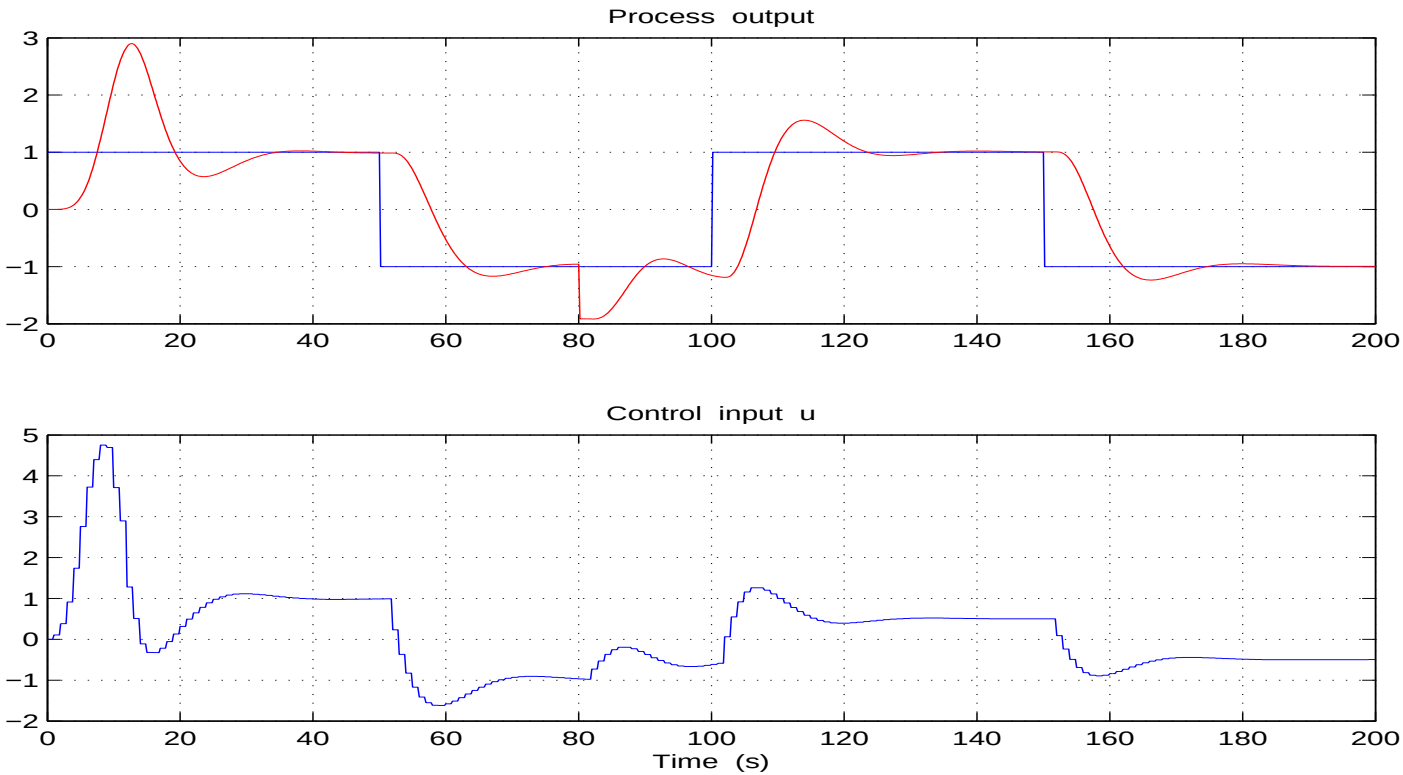
System response with controller E2



Forgetting factor β, estimation of the parameters a_1, b_0, b_1 and estimation error. Note that the RLS algorithm gives satisfying results even at the second step of the reference signal.

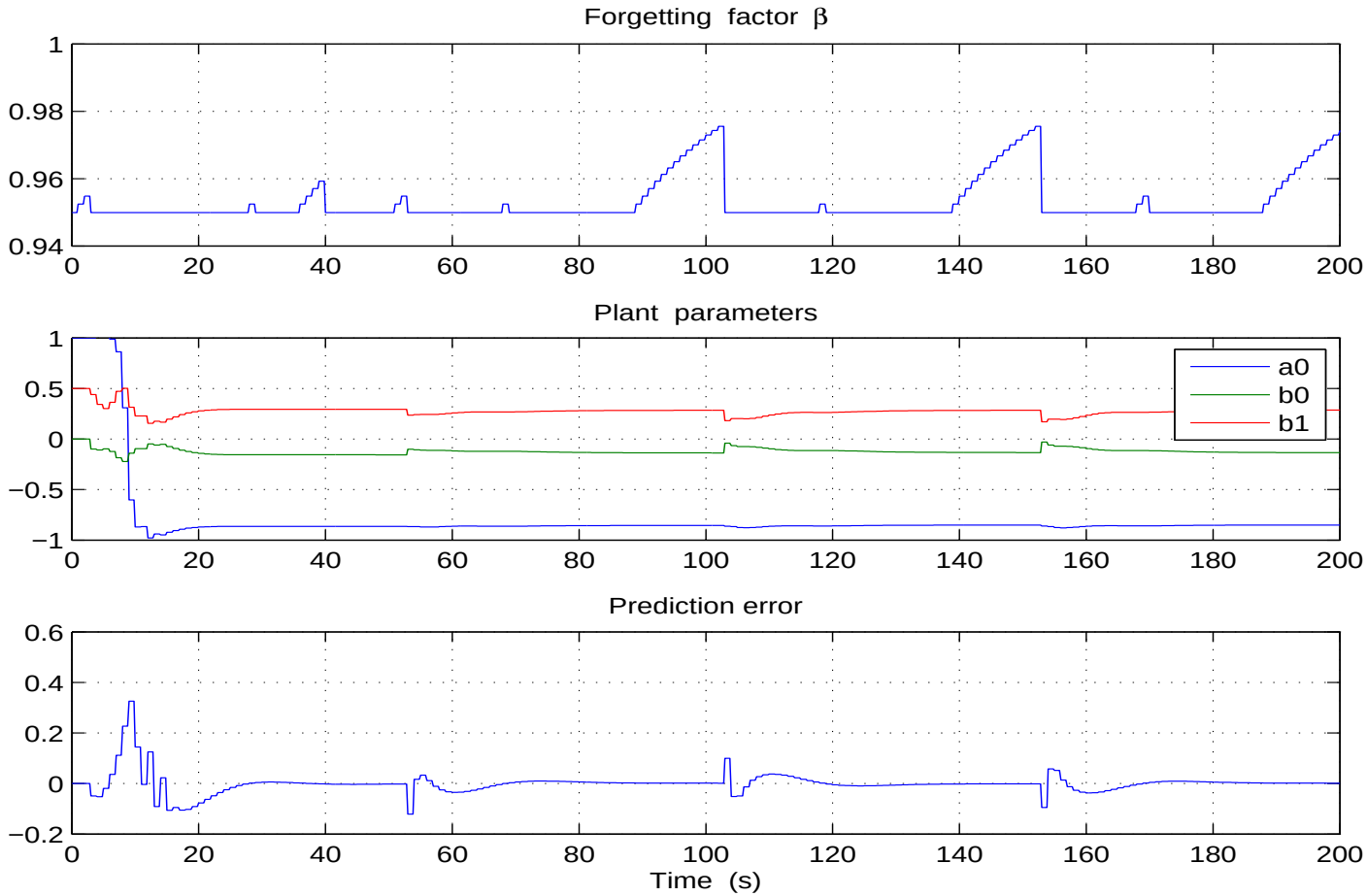
ST regulators by poles/zeros assignment: Example – E2

System response with controller E2: variation of the process's gain at $t = 80\ s$.



ST regulators by poles/zeros assignment: Example – E2

System response with controller E2: variation of the process's gain at $t = 80\ s$.



ST regulators by poles/zeros assignment: Example – E2

The discrete time model has been assumed as:

$$G(z) = z^{-1}\frac{b_0 + b_1 z^{-1}}{1 + a_1 z^{-1}} = \frac{Y(z)}{U(z)} \quad \rightarrow \quad y(k) = -a_1 y(k-1) + b_0 u(k-1) + b_1 u(k-2)$$

The final values of the parameters estimated by the RLS algorithm are:

In case of constant gain:

$$a_1 = -0.850393, \quad b_0 = -0.135577, \quad b_1 = 0.286491$$

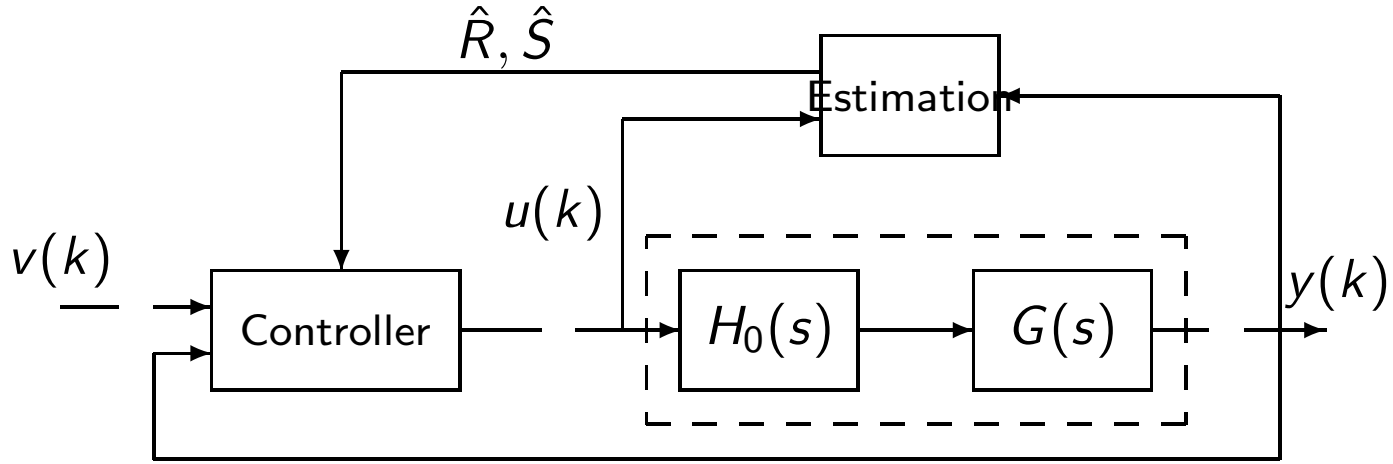
In case of variable gain:

$$a_1 = -0.850609, \quad b_0 = -0.269460, \quad b_1 = 0.570891$$

In both cases:
- the pole is $p \approx 0.85$, stable
- the zero is $z = 2.11$, $\rightarrow$ **unstable!**

ST regulators by poles/zeros assignment: Example – I1

ST controller according to algorithm I1 - zeros are canceled



By assuming

$$A_0(z) = 1, \qquad R(z) = r_0 + r_1 z^{-1}, \qquad S(z) = s_0 + s_1 z^{-1}$$

we get

$$(1 + c_1 z^{-1} + c_2 z^{-2}) y(k) = z^{-1}[(r_0 + r_1 z^{-1}) u(k) + (s_0 + s_1 z^{-1}) y(k)]$$

ST regulators by poles/zeros assignment: Example – I1

From which

$$y(k)+c_1y(k-1)+c_2y(k-2) =$$

$$= s_0y(k-1)+s_1y(k-2)+r_0u(k-1)+r_1u(k-2)$$

The parameters s_0, s_1, r_0, r_1 are estimated. One gets:

$$(\hat{r}_0 + \hat{r}_1 z^{-1})u(k) = kv(k) - (\hat{s}_0 + \hat{s}_1 z^{-1})y(k)$$

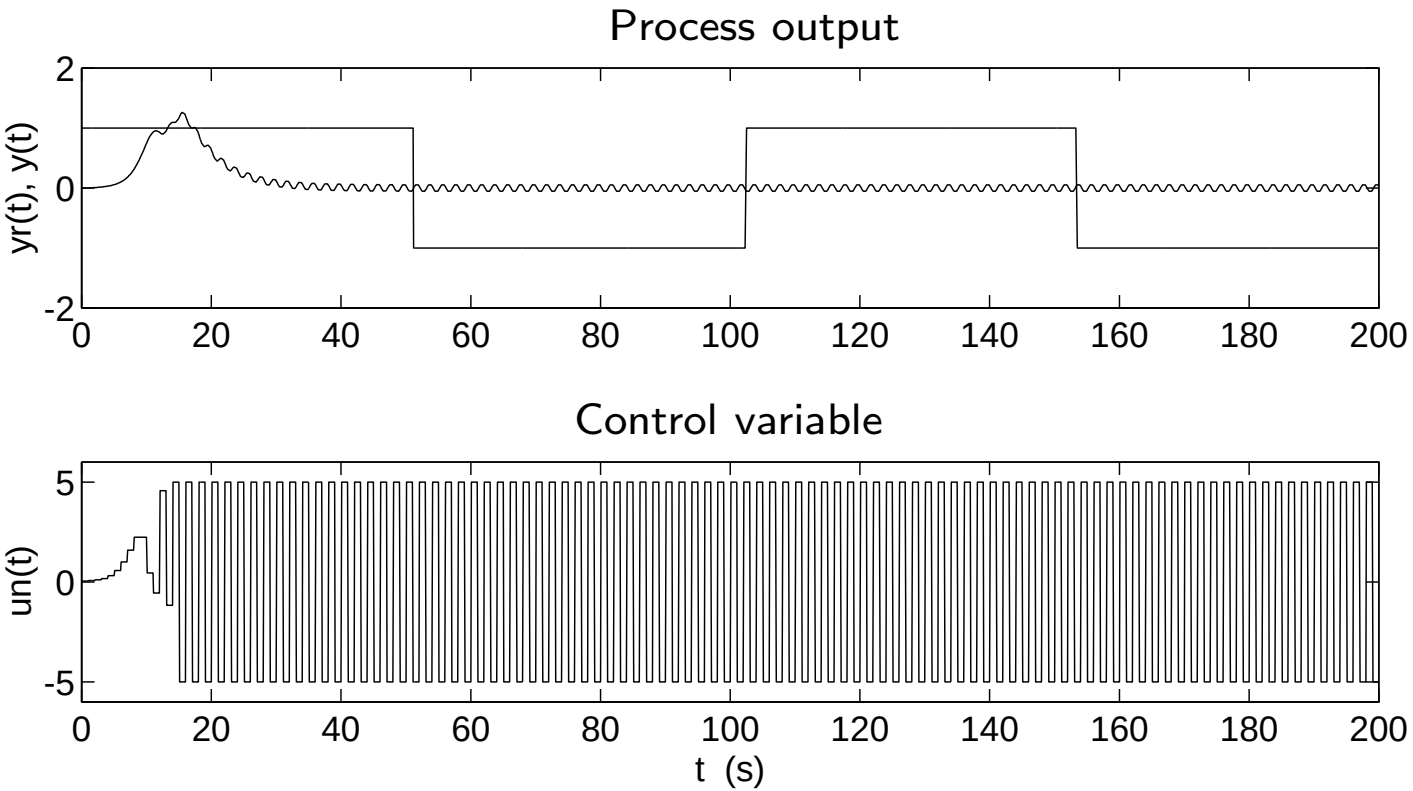
and then

$$u(k) = \frac{1}{\hat{r}_0}[k\ v(k) - \hat{s}_0 y(k) - \hat{s}_1 y(k-1) - \hat{r}_1 u(k-1)]$$

with $k = 1 + c_1 + c_2$.

ST regulators by poles/zeros assignment: Example – I1

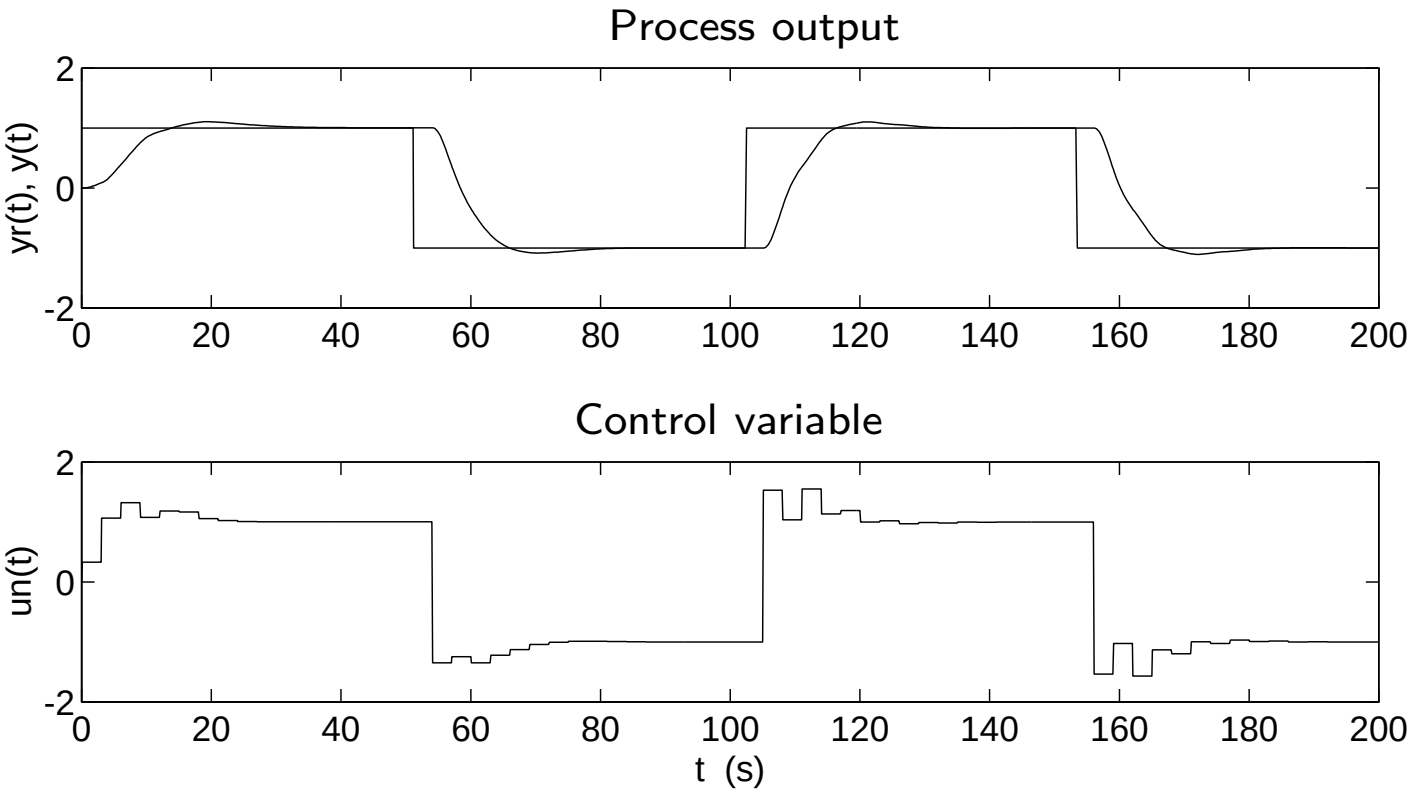
System response with controller I1 and $T = 1\ s$.



Unstable behaviour due to cancelation of non-minimum phase zero ($z = -2.11$).

ST regulators by poles/zeros assignment: Example – I1

System output with algorithm I1, $T = 3\ s$.



With a higher sampling period, non-minimum phase zeros are absent.

ST regulators by poles/zeros assignment: Example – I1