5. Discussions

5.1. Some Recommendations in AW Design

Before designing the anti-windup compensator, it is important to make such controller, that *unlimited response* of the system is as desired (rise time, overshoot, decay ratio, etc.). The unlimited response could be performed with slight changes of the reference (w). If this can not be done (e.g. if velocity limit exist), we can design a controller by using conditioning technique, because the response of such controller will be the closest to unlimited response.

To show how important is a design of desired unlimited system response, we made an example with the process

$$G_{PR} = \frac{1}{(1+4s)(1+s)^2}, \tag{5.1}$$

the controller

$$K = 10, \;\; T_i = 30s, \;\; T_d = 0.5s, \;\; T_f = 0.05s \tag{5.2}$$

and the following limitations

$$U_{\max} = 2, \;\; U_{\min} = 0 \tag{5.3}$$

The response of such system is shown in Figures 5.1 to 5.3.

Fig. 5.1 represent process output (y). Dotted line shows unlimited process response. We can see that controller is not tuned well. The process have a long settling time. For the system tuned in such a manner, windup occurs (represented with dash-dotted line). We can see that in this case the process response is better than in unlimited case. Of course, it does not mean windup produces useful and desired system behaviour in general. We should only pay *enough attention when designing controller for unlimited response*.

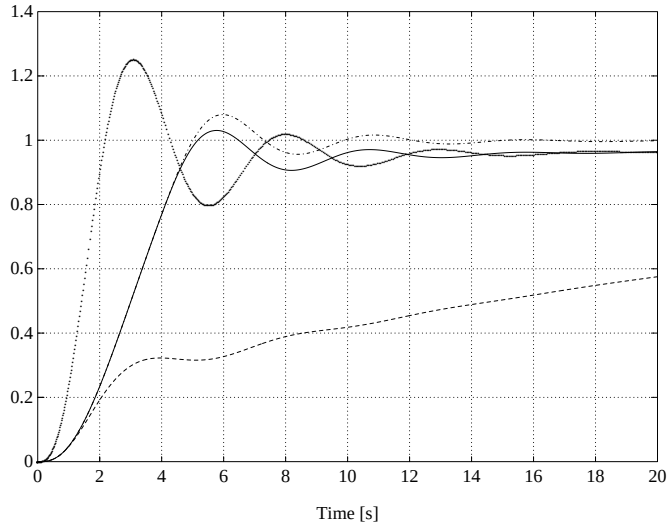

Fig. 5.1. Process output (y); __ Conditioning technique,
-- Incremental algorithm, -.- Without AW protection, ... Unlimited response

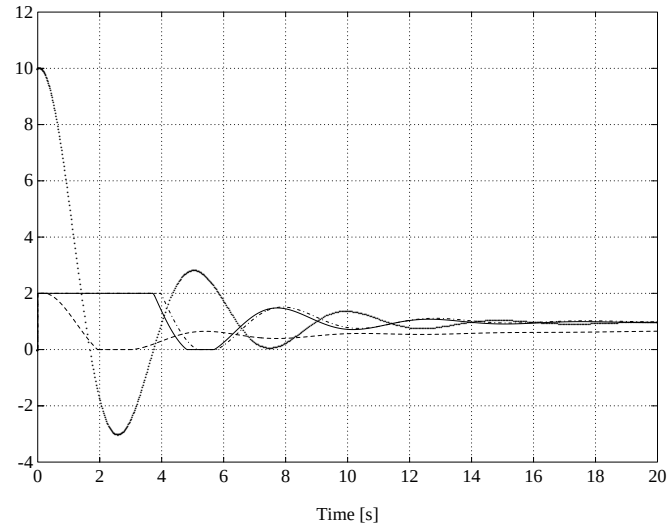

Fig. 5.2. Process input (u^r); __ Conditioning technique,
-- Incremental algorithm, -.- Without AW protection, ... Unlimited response

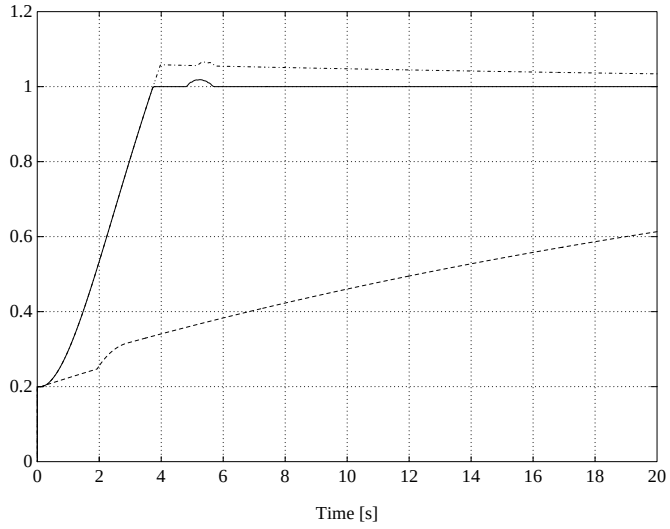

Fig. 5.3. Realisable reference (w^r); __ Conditioning technique, -- Incremental algorithm, -.- Without AW protection

Anyhow, if *desired unlimited response* is such as shown in Fig. 5.1, then the closest response, when limitation occurs, is the one obtained with conditioning technique.

Another problem appears when ***process limitations are too restrictive***.

If this is the case, the use of conditioning technique might produce multiple opposite limitations at the process input and thus it leads to oscillations of the process output [Ronnback et al., 1991] [Walgama et al., 1992]. If this happens*, the controller parameters should be tuned so as to decrease the oscillations* (more sluggish controller). This is normal, because we can not expect superior process response in highly limited systems. To depict above statements, we made an example in which the process

$$G_{PR} = \frac{1}{(1+10s)^2} \tag{5.4}$$

and the following controller

$$K = 20, \quad T_i = 40.8s, \quad T_d = 1.16s, \quad T_f = 0.116s \tag{5.5}$$

were used. The process limitations were very restrictive (specially the rate limitation)

$$U_{\max} = 4,\ \ U_{\min} = 0,\ \ v_{\max} = 0.25s^{-1},\ \ \ v_{\min} = -0.25s^{-1} \tag{5.6}$$

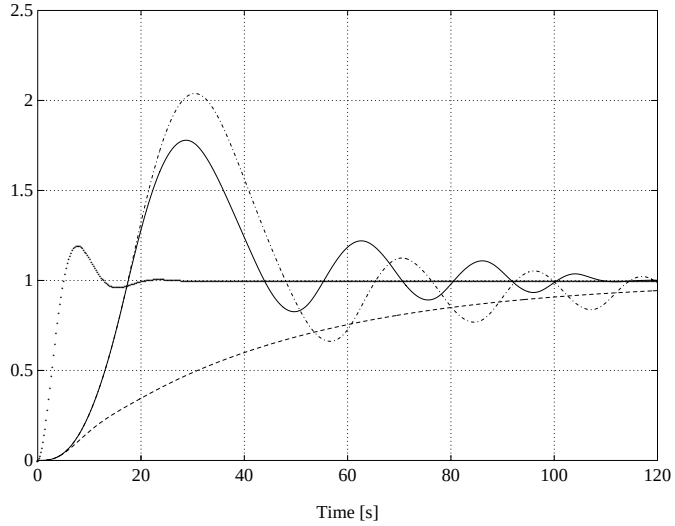

Fig. 5.4. Process output (y); __ Conditioning technique,
-- Incremental algorithm, -.- Without AW protection, ... Unlimited response

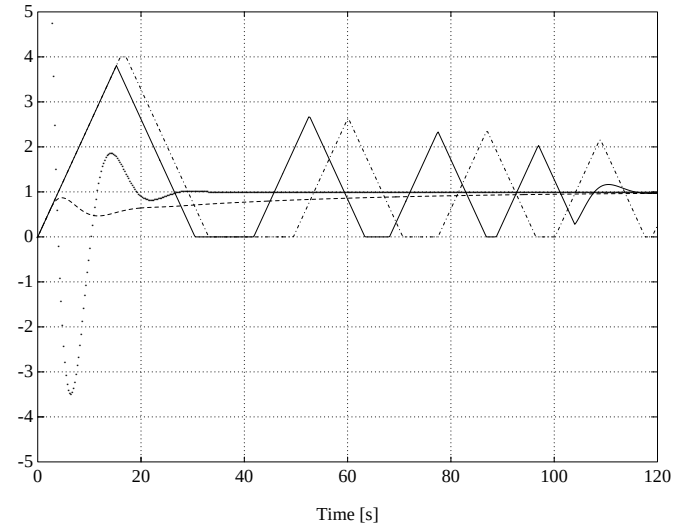

Fig. 5.5. Process input (u^r); __ Conditioning technique,
-- Incremental algorithm, -.- Without AW protection, ... Unlimited response

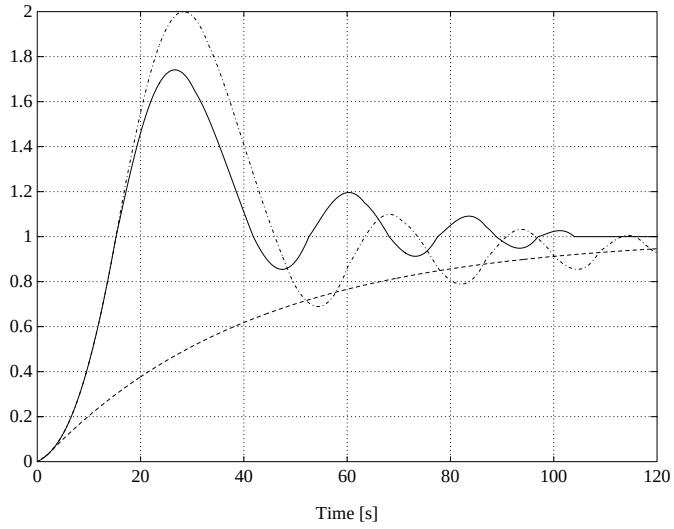

Fig. 5.6. Realisable reference (w^r); __ Conditioning technique, -- Incremental algorithm, -.- Without AW protection

The response of such system is shown in Figures 5.4 to 5.6.

We can see the process response is oscillatory. Incremental algorithm gives us stable, but very sluggish response. The solution of such problem could be in decreasing the proportional gain (K) of the controller, as mentioned above. With the same process (5.4) and limitations (5.6), we chose new controller

$$K = 5, \;\; T_i = 20s, \;\; T_d = 1.16s, \;\; T_f = 0.116s \tag{5.7}$$

Figures 5.7 to 5.9 show process response when the new controller is used. An improved system response can be observed.

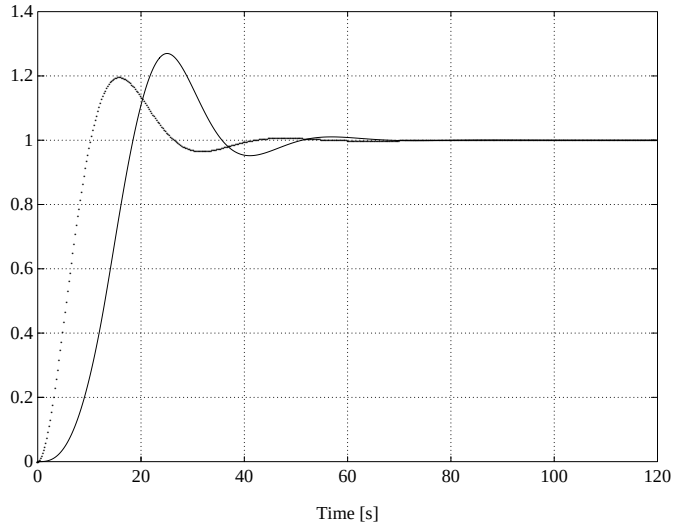

Fig. 5.7. Process output (y); __ Conditioning technique, ... Unlimited response

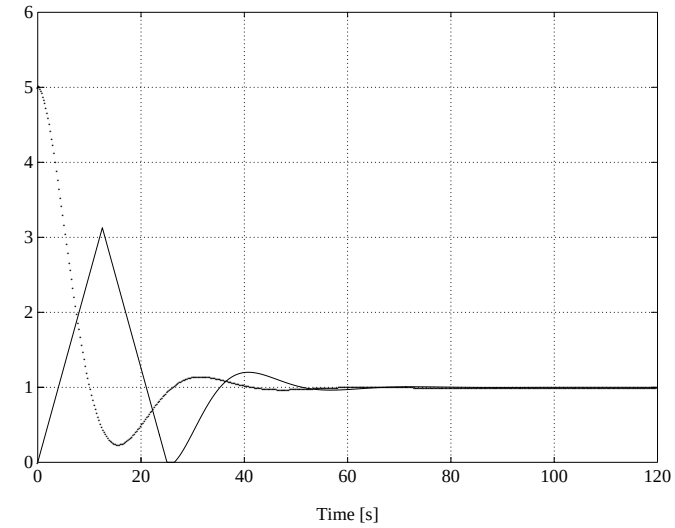

Fig. 5.8. Process input (u^r); __ Conditioning technique, ... Unlimited response

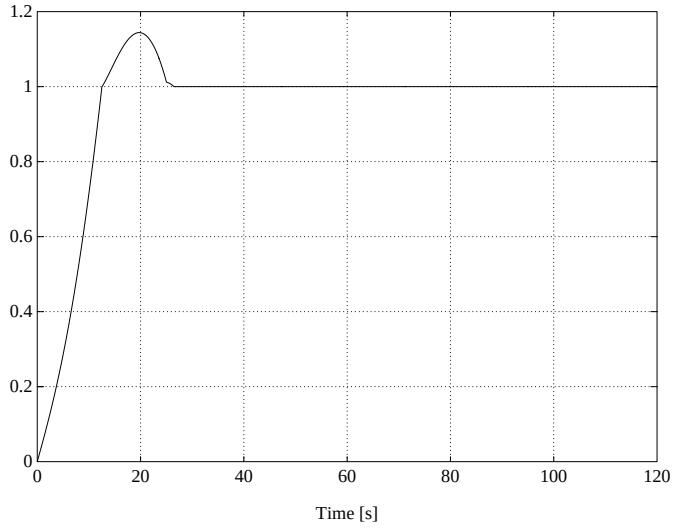

Fig. 5.9. Realisable reference (w^r); __ Conditioning technique

To stabilise the system, we could use changed controller parameters (e.g. changing factor β and γ in generalised PID controller) in order not to change controller disturbance rejection [Hang et al. 1991].

Here we present another solution how to stabilise the system for not frequent multiple opposite limitations. In this case, using variable K_a, a change of controller parameters is not needed.

As an example, K_a can be reduced by factor 2 if the limitation changes sign (from positive to negative or vice versa). To illustrate such solution, we made a simulation with the process (5.4), limits (5.6) and original controller (5.5).

The result of proposed method can be seen in Figures 5.10 to 5.13. We can see that the tracking performance of the proposed method (with changing K_a) is better than the results obtained with the incremental algorithm.

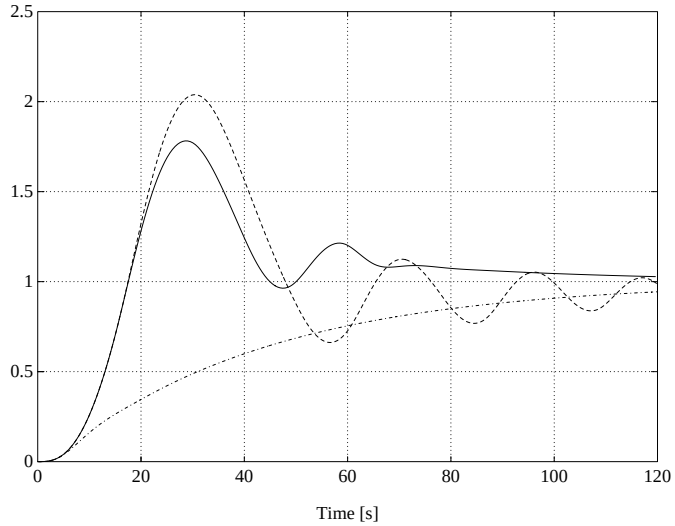

Fig. 5.10. Process output (y); __ Variable K_a method, -- Without AW protection, -.- Incremental algorithm

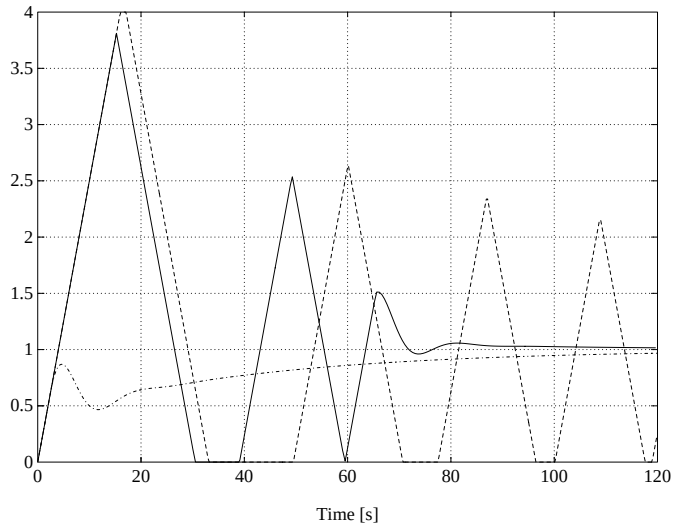

Fig. 5.11. Process input (u^r); __ Variable K_a method, -- Without AW protection, -.- Incremental algorithm

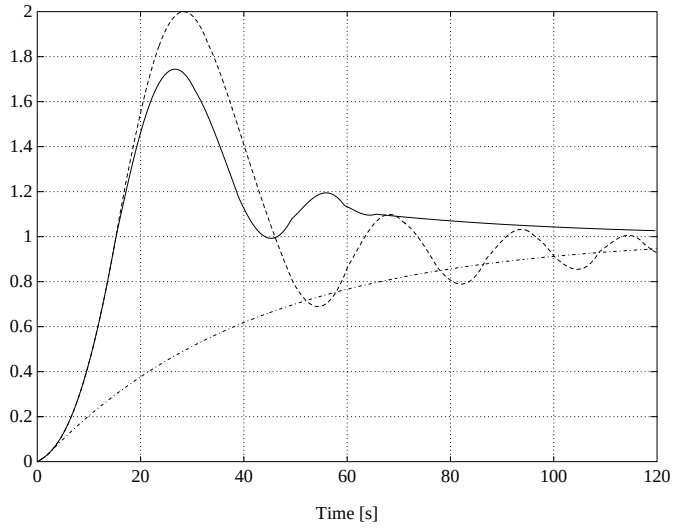

Fig. 5.12. Realisable reference (w^r) ; __ Variable K_a method, -- Without AW protection, -.- Incremental algorithm

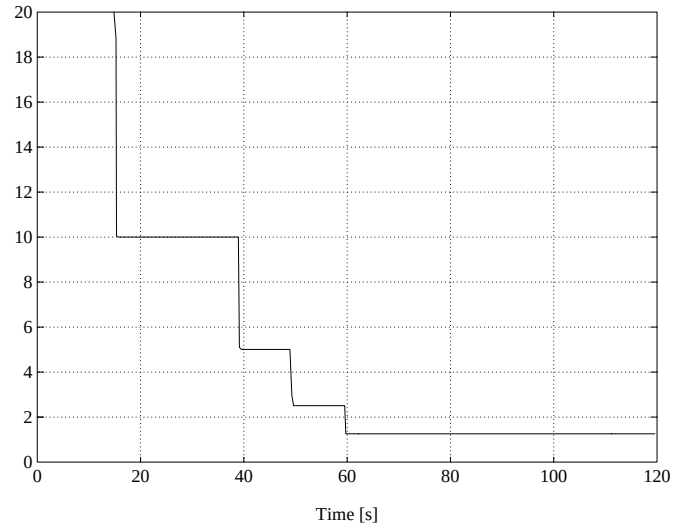

Fig. 5.13. The value of K_a for the variable K_a method

In the sequal we would like to discuss how to obtain the limited control variable u^r in the *real applications*.

Direct measurement of u^r would demand additional controller input, cables, filters and equipment. In a real application it is usually too expensive. In most systems, this problem is solved by estimating the process input limitations inside the controller.

Fig. 5.14. represents the limited system, where LIM_C represents controller limitations and LIM_P represents process limitations.

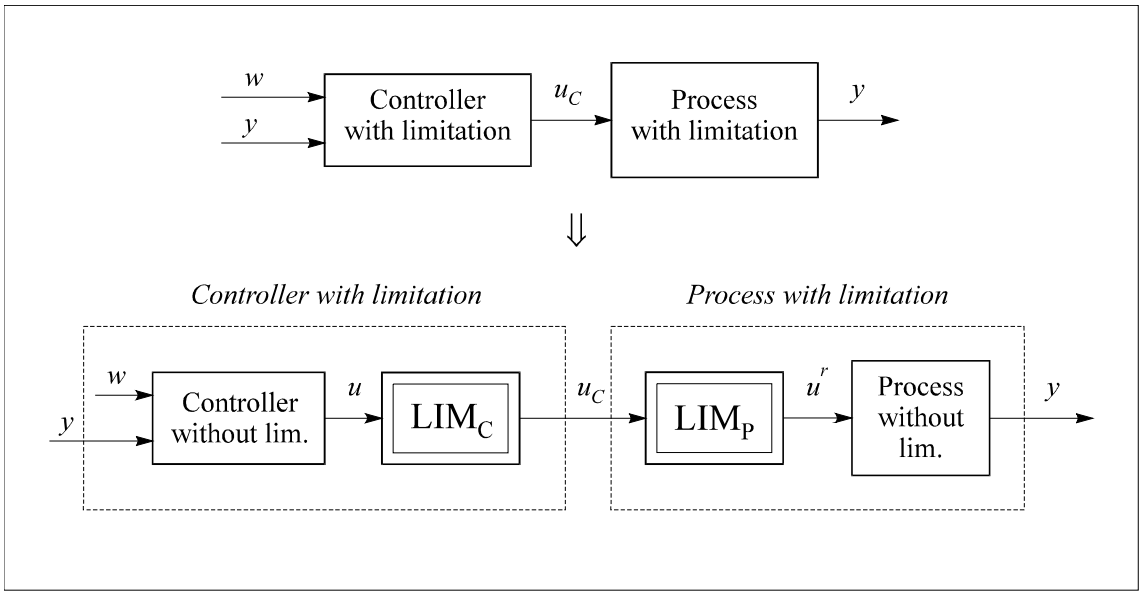

Fig. 5.14. The representation of the limited system

We can put an *estimated limitation* LIM' before the *real process limitation* LIM (Fig 5.15).

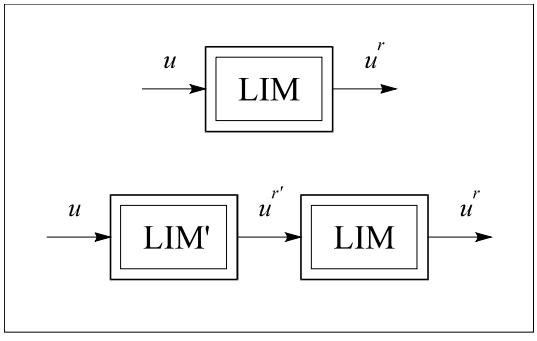

Fig. 5.15. An estimated limitation before real limitation

If LIM' is the same as LIM, then $u^{r'}$ is already limited by LIM' and will not be limited again by LIM. So, u^r will become the same as $u^{r'}$.

Estimation LIM' can be even more "restrictive" than LIM and $u^{r'}$ would still remain equal to u^r.

Limited controller and process can be represented as it is shown in Fig. 5.14. We can put an estimation of the process limitation (LIM_p') into controller and controller output signal ($u^{r'}$) will become the same as u^r as long as LIM_p' is adequate. Fig. 5.16. shows the solution.

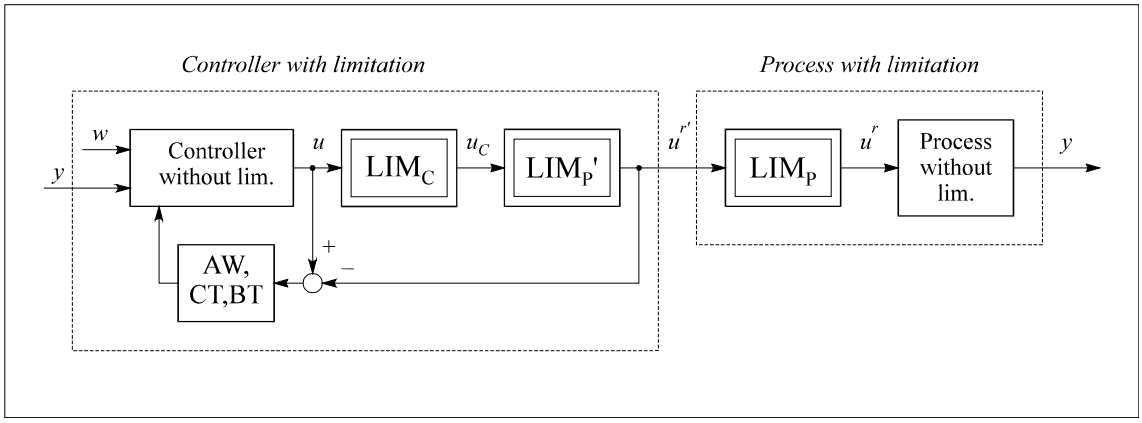

Fig. 5.16. Estimated process limitation inside the controller

Therefore, the controller output signal $u^{r'}$ can be used in AW (BT, CT) algorithms instead of u^r. If LIM_p can not be estimated accurately, we can use a more "restrictive" estimation. However, if a too "restrictive" estimation is used, the system may become oscillatory.

5.2. Non-Linearity and AW

As it have been mentioned in introduction, actuator could have besides the limitations also a non-linear behaviour. Almost every process have a non-linearity, so it is important to know how to "carry out" non-linearities in the framework of anti-windup. The presented anti-windup schemes (see previous chapters) are all capable also to handle with non-linearity if it is of such type that $u=u^r$ when system is in steady state. If

the last assumption is not fulfilled, there control error in steady state will exist (see Fig. 5.17) which is equal to

$$e_s = \frac{u_s - u_s^r}{K_a} \tag{5.8}$$

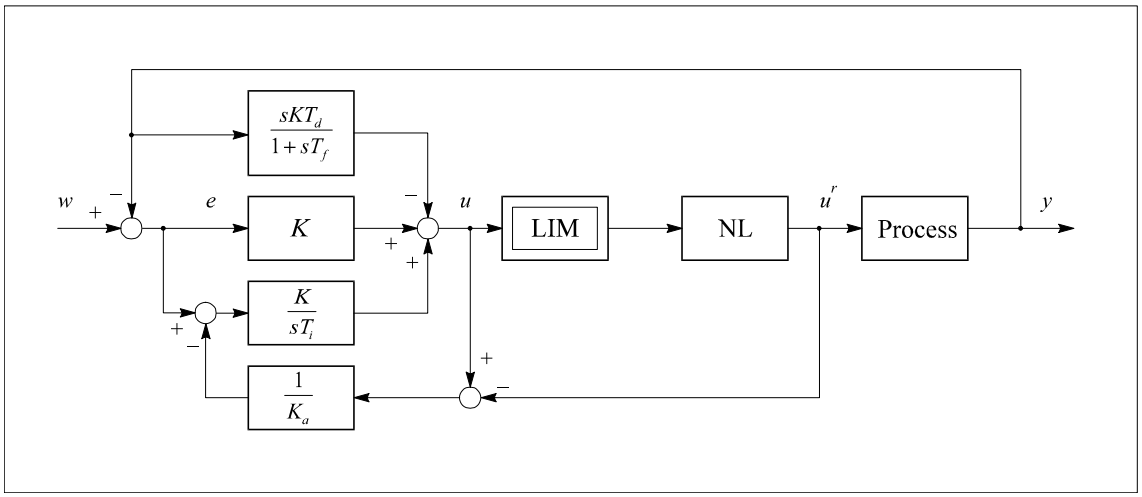

Fig. 5.17. Non-linear system with PID controller

where e_s, u_s and u_s^r are control error, control output and realisable control output in steady state, respectively.

To illustrate the above statements, we made some examples. First example shows the case when actuator has such non-linear behaviour, that $u=u^r$ in steady state (in range from 0 to 2). Fig. 5.18 shows the characteristics of the non-linearity.

We used the process

$$G_{PR} = \frac{1}{(1+8s)(1+4s)} \tag{5.9}$$

and the following controller

$$K = 10, \quad T_i = 15s, \quad T_d = 0.5s, \quad T_f = 0.05s \tag{5.10}$$

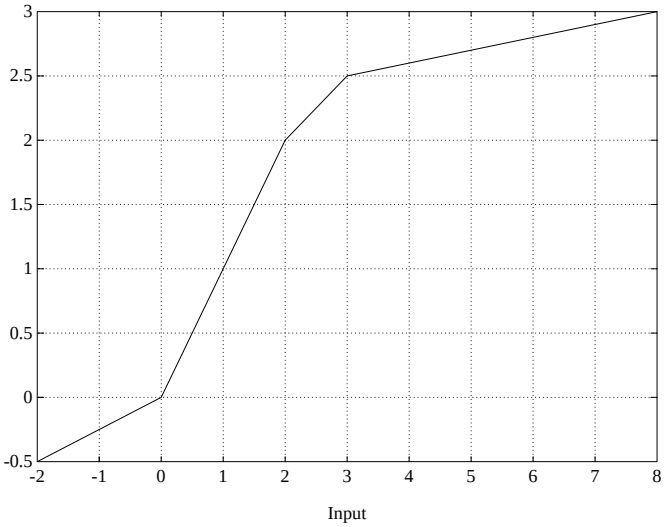

Fig. 5.18. The first non-linearity; abscisse = u, ordinate = u^r

The experiment was made with the reference change from 0 to 1. From (5.9), process static gain is 1, so in steady state u will be 1. From Fig. 5.18 we can see that u^r will be the same as u and we expect no offset in steady state (5.8). The results of experiments are shown in Figures 5.19 to 5.21. We can see expected AW behaviour for three cases (no anti-windup protection ($K_a \to \infty$), conditioning technique ($K_a = K$) and incremental algorithm ($K_a \to 0$)).

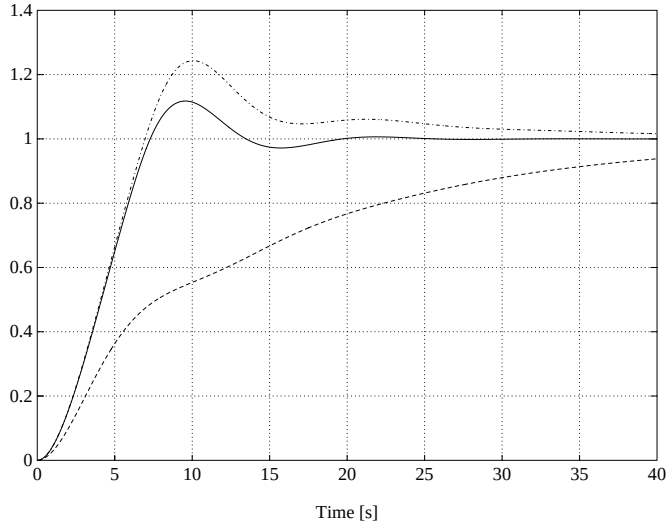

Fig. 5.19. Process output (y); __ Conditioning technique, -- Incremental algorithm, -.- Without AW protection

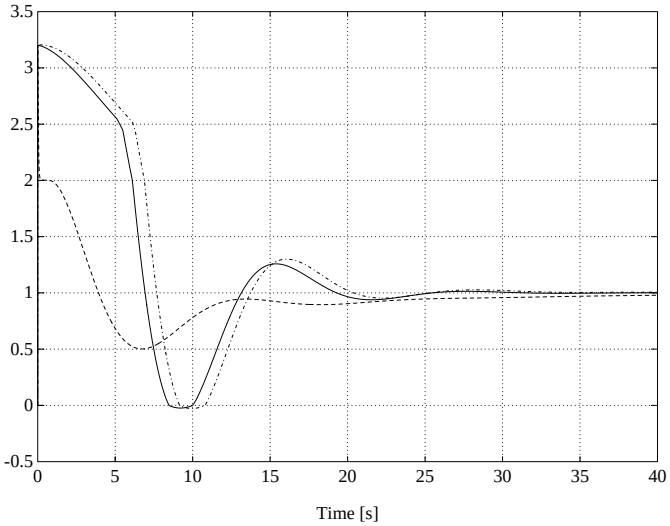

Fig. 5.20. Process input (u^r); __ Conditioning technique, -- Incremental algorithm, -.- Without AW protection

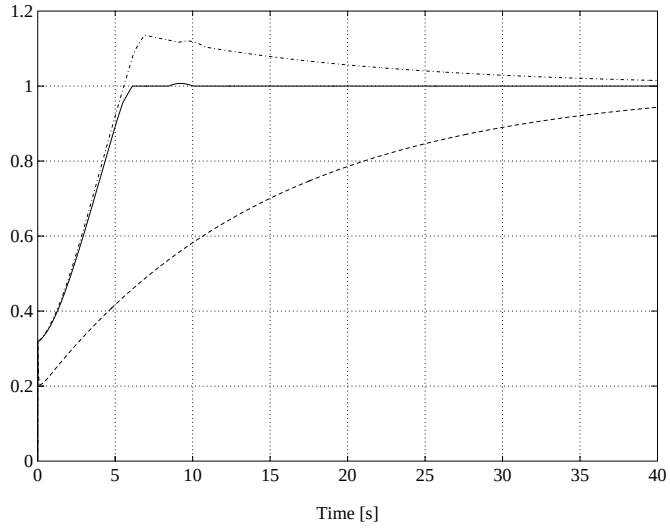

Fig. 5.21. Realisable reference (w^r); __ Conditioning technique, -- Incremental algorithm, -.- Without AW protection

Another experiment shows what can happen if non-linearity is such that in steady state $u \neq u^r$. Non-linearity from Fig. 5.18 is slightly changed in the range from 0 to 3 as it is shown in Fig. 5.22.

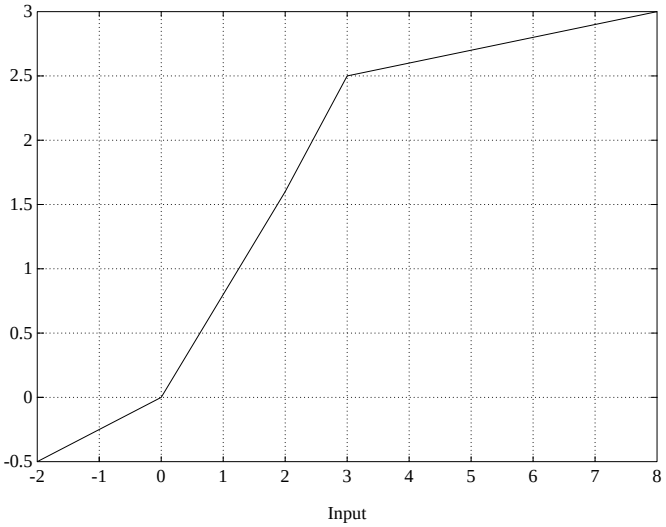

Fig. 5.22. The second non-linearity; abscisse = u, ordinate = u^r

We have used the same process (5.9) and controller (5.10) as before. The results of experiment were changed as shown in Figures 5.23 to 5.25. We can see, that the biggest change from previous experiment occured when using incremental algorithm ($K_a \rightarrow 0$) while no change of steady state position can be observed when using no anti-windup ($K_a \rightarrow \infty$), what is a consequence of equation (5.8). Note slight change of steady state point when using conditioning technique ($K_a = K$).

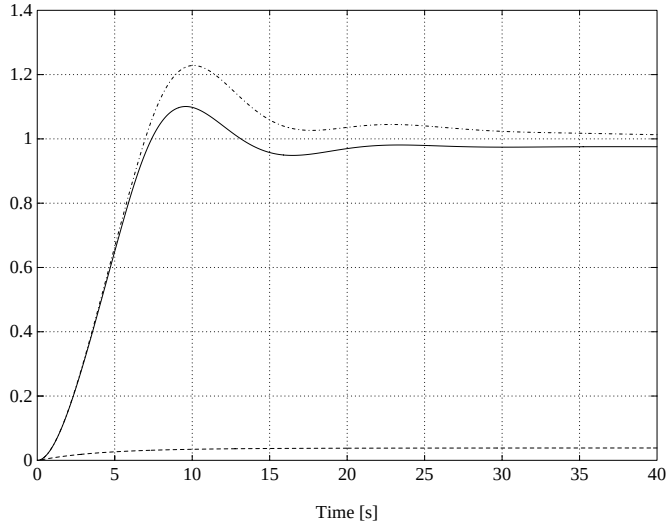

Fig. 5.23. Process output (y); __ Conditioning technique, -- Incremental algorithm, -.- Without AW protection

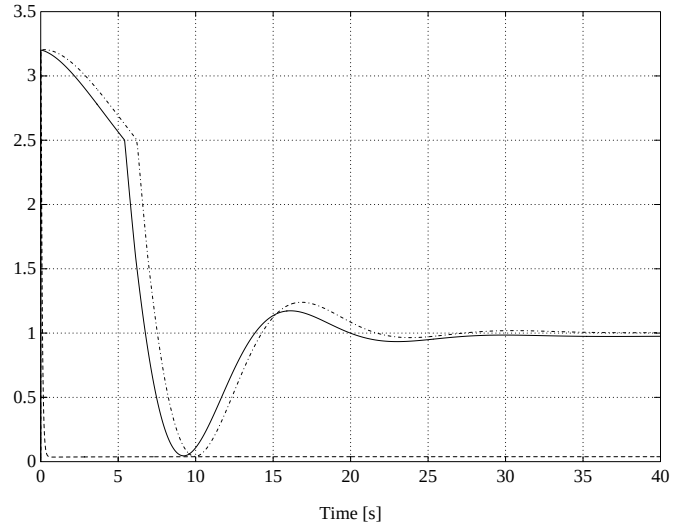

Fig. 5.24. Process input (u^r); __ Conditioning technique, -- Incremental algorithm, -.- Without AW protection

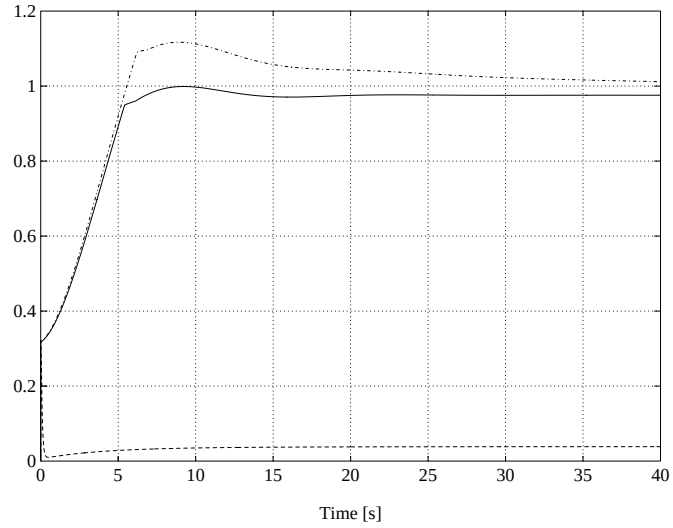

Fig. 5.25. Realisable reference (w^r); __ Conditioning technique, -- Incremental algorithm, -.- Without AW protection

However the possibility exists which solves the problem of the offset in steady state and improve system response. Before the process non-linearity and a limitation we can put an inverse function of an estimation of the process non-linearity as shown in Fig. 5.26.

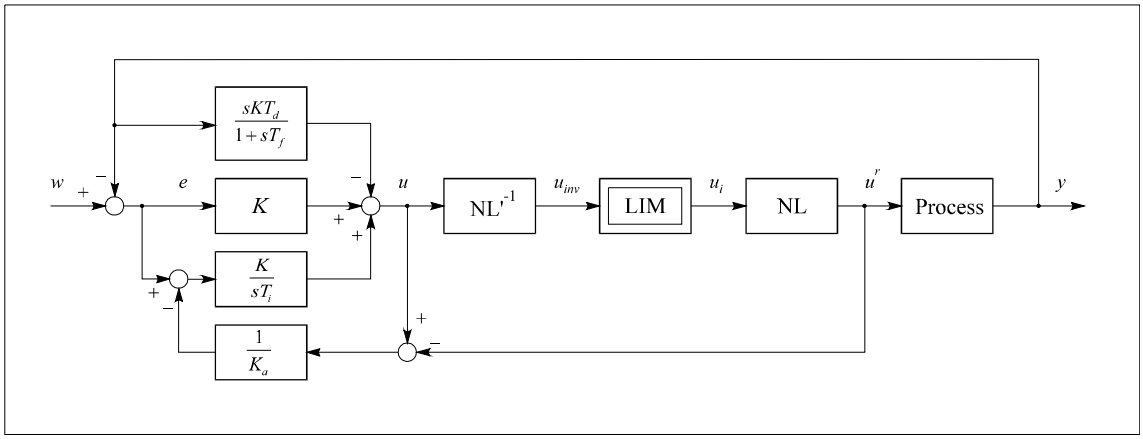

Fig. 5.26. The inverse function of the process non-linearity estimation

where NL and NL'$^{-1}$ represent the process non-linearity and the inverse of the process non-linearity estimation, respectively. Note that it is enough to make a non-linear estimation only for the range where steady state is expected.

In practice, as explained in previous section, we are not measuring u^r directly, but we approximate it inside the controller. Most common realisation is shown in Fig. 5.27.

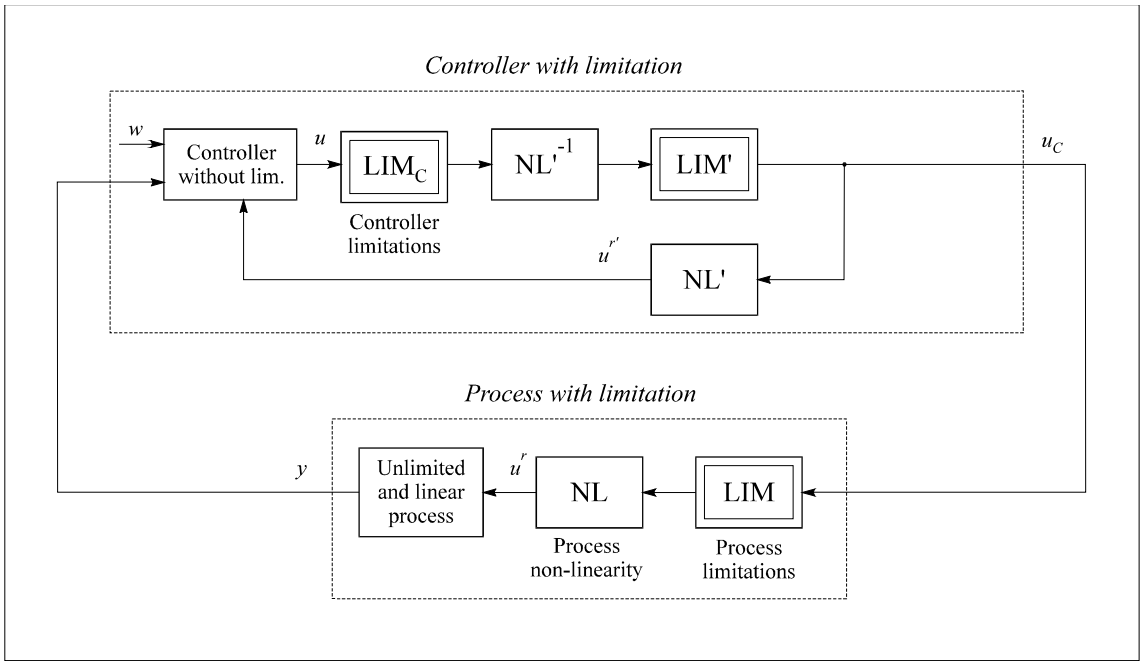

Fig. 5.27. The estimation of u^r inside the controller in the case of non-linearity

where LIM_C represents controller limitations and LIM' is an estimation of process (actuator) limitations. Instead of u^r we use $u^{r\prime}$ in anti-windup scheme. Note that the way of realisation diminishes the problems of exact estimation of the process non-linearity. During the time, when the process operates inside limitations, u and $u^{r\prime}$ in Fig. 5.27 are *always the same* while in scheme 5.26 this is not the case if the estimation of the process non-linearity is not exactly the same as the real one. Solution in Fig. 5.26 could therefore cause an offset in steady state.

We made an example to show improvements achieved by the proposed design. For the same process (5.9) and controller (5.10), we had the following process limitations:

$$U_{\max} = 2, \;\; U_{\min} = 0, \;\; v_{\max} = 0.5s^{-1}, \;\; v_{\min} = -0.5s^{-1} \tag{5.11}$$

and process non-linearity

$$u^r = 0.4|u_l|u_l \tag{5.12}$$

from which we calculated the inverse function

$$u_{inv} = \frac{|u|}{u}\sqrt{2.5|u|} \tag{5.13}$$

Results of an experiment when set point changes from 0 to 1 is shown in Figures 5.28 to 5.30. Dotted line represents the response without limitations (only non-linearity is present in the system). We can see that the previous problems, related to offset, disappear.

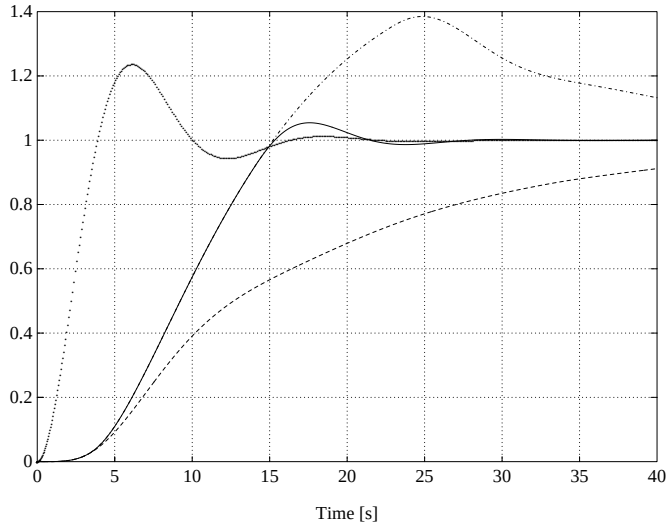

Fig. 5.28. Process output (y); __ Conditioning technique,
-- Incremental algorithm, -.- Without AW protection, ... Unlimited response

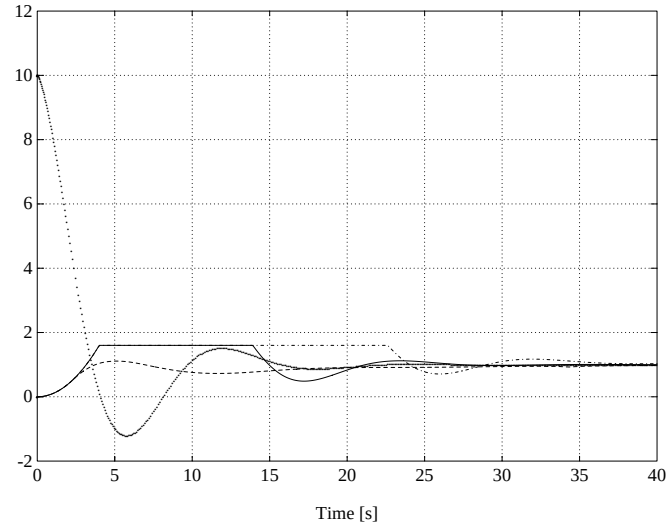

Fig. 5.29. Process input (u^r); __ Conditioning technique,
-- Incremental algorithm, -.- Without AW protection, ... Unlimited response

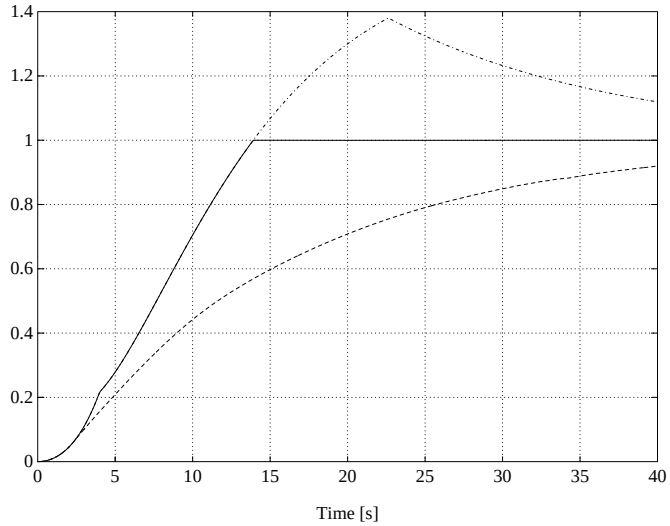

Fig. 5.30. Realisable reference (w^r); __ Conditioning technique, -- Incremental algorithm, -.- Without AW protection