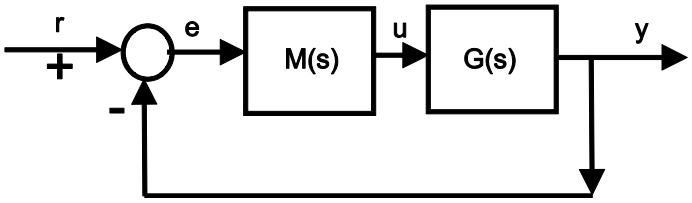
Model Predictive control for beginners *by Anthony Rossiter*

The University Of Sheffield.

Classical control 1.4: Constraints

Classical control techniques such as root-loci, Bode, Nyquist and indeed state-space methods such as LQR assume linear analysis is valid. This includes any discussion of robust stability margins and thus an inherent capacity to deal with some level of model uncertainty. However, all real process include constraints such as limits in absolute values and rates for actuators (inputs), desired safety limits on outputs and states, desired quality limits on outputs (linked to profit) and so forth. Whenever a system comes up against a constraint, then the overall system behaviour is highly likely to be come non-linear and therefore any linear analysis is not longer valid. Indeed, one can easily come up with examples where a linear feedback systems is supposedly very robust to parameter uncertainty, but the inclusion of a constraint causes instability. The main purpose of this note is to give some illustrations of the dangers of constraints but not to discuss possible solutions.

For this note, a simple feedback structure is assumed as follows with compensator M(s) and process G(s).



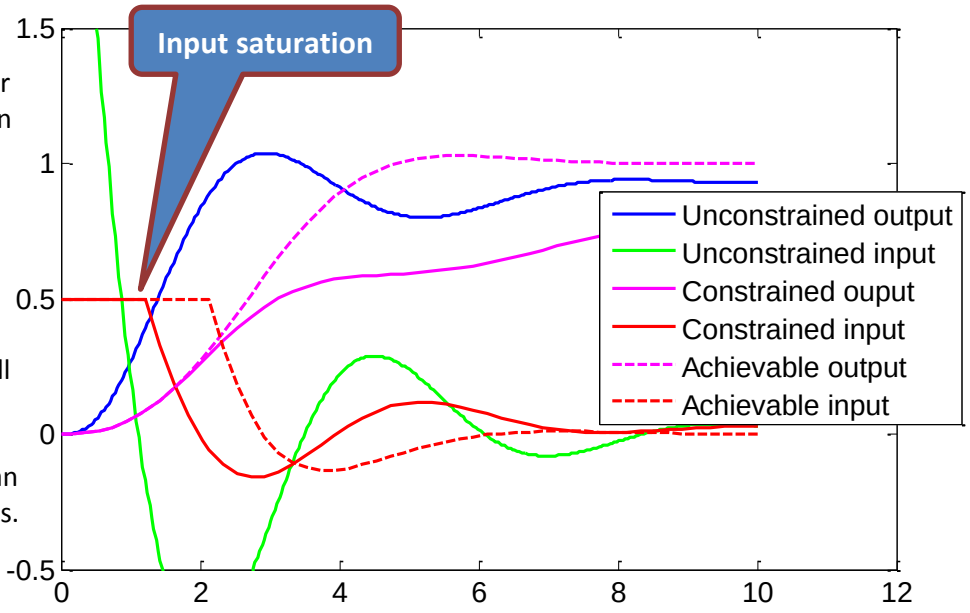
Example 1 – constraints have only a small impact on behaviour

In some cases, the presence of a constraint may have a relatively small impact. One such case would be rate constraints on the inputs in conjunction with a process having benign dynamics. In this case, the best control one can achieve is often to saturate the input.

Consider the following system and lead compensator pairing and upper input limit:

$$G = \frac{1}{s(s+1)^2}; \quad M = 4\frac{s+0.2}{s+2}; \quad u \le 0.5$$

The figure compares the behaviour with a and without constraints from which it is clear that the simple use of saturation has led to rather sluggish performance, especially when compared to the achievable constrained behaviour with an alternative approach (shown in dashed lines). Nevertheless, the constrained behaviour is still acceptable.



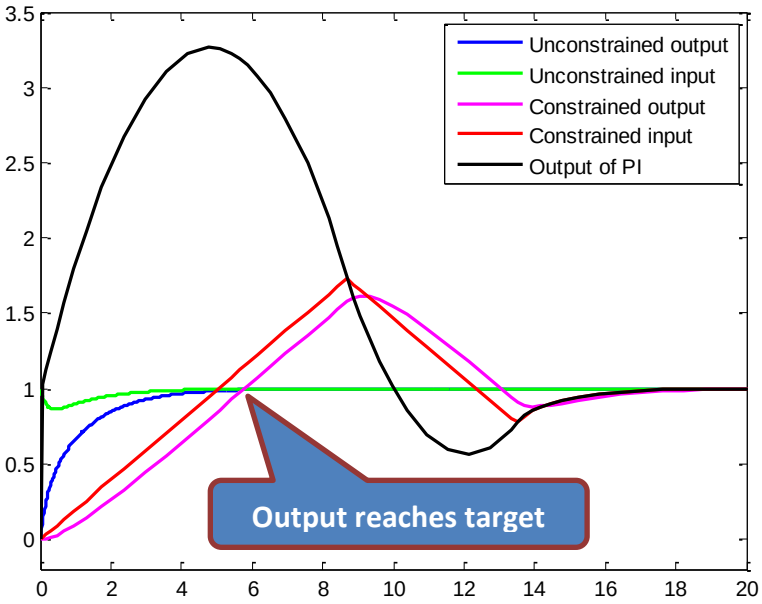
In summary, input saturation can work satisfactorily in some cases.

Example 2 – constraints have a major impact on behaviour

In some cases, the presence of a constraint may have a huge impact on behaviour if not properly accounted for. Here an illustration of a process with almost 1st order dynamics is used to demonstrate the point. This example includes a rate constraint on the input.

$$G = \frac{2s+4}{(s+4)(s+1)}; \quad M = \frac{s+1}{s}; \quad -0.5 \le u \le 4; \quad \left|\frac{du}{dt}\right| \le 0.2$$

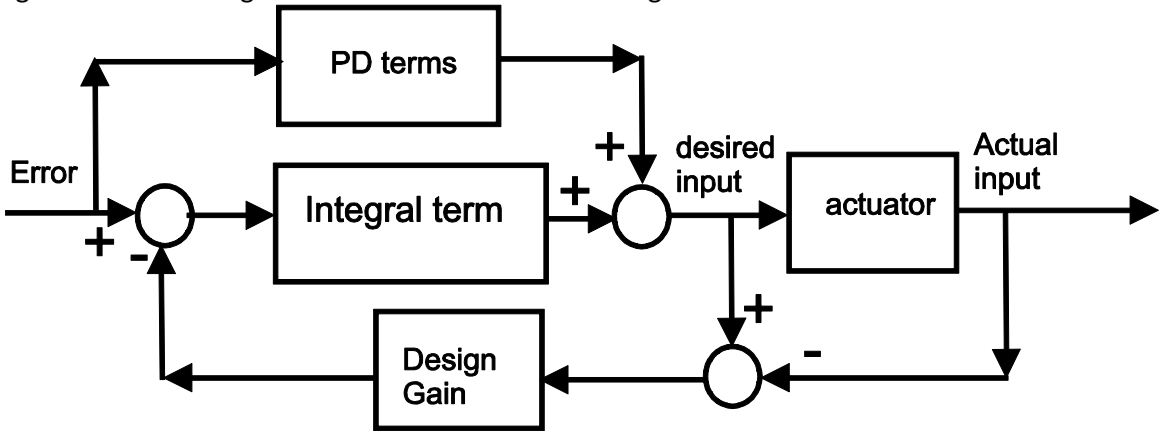
The corresponding closed-loop responses are shown here. It is immediately clear that although the unconstrained responses are excellent, the constrained responses are unacceptable and show no sign of converging to the desired output of one in a reasonable manner. Indeed, the responses seem to enter some form of oscillation which is quite unexpected from the underlying linear dynamics.



The problem here is caused by so called **integral saturation** or windup, that is where the integral term keeps increasing even though the input has saturated. The PI is proposing to use the signal in BLACK, but the implemented input is the signal in RED. Hence, even though the output reaches and passes the target around t=6, the integral term is up at around 3.2 and there a significant period of negative error is required before the integration brings this value back down to the sorts of steady-state input values required. Consequently, the saturated input (RED) continues to grow even though one really wants this to start reducing immediately. The input only begins reducing again once the output of the PI (BLACK) drops below the RED line and this is not until around t=9. In this case it is the rate limit which has dominated the behaviour but one can easily create examples where the absolute limit is equally problematic.

Possible solutions and anti-windup solutions

A common solution is to put some form of anti-windup into the controller, that is to detect that the actual input is not the same as the controller output. In the era of hardware controller implementations, a remedy involved feeding this error signal back into the compensator as a means of ensuring the compensator output stayed close to constraints. A typical structure is shown here but viewers should note that there is still a design gain in this and no guarantee it will be effective enough.



Nevertheless, in the modern era with ready access to online computing and indeed the expectation that even routine PI compensators are implemented via a computer, it is possible to be a little more systematic and utilise the error between the actual input and the desired input more precisely to reset the integral as required, moreover without the need for further tuning. This discussion is not pursued at this point, suffice to say that doing this systematically is still a major challenge.