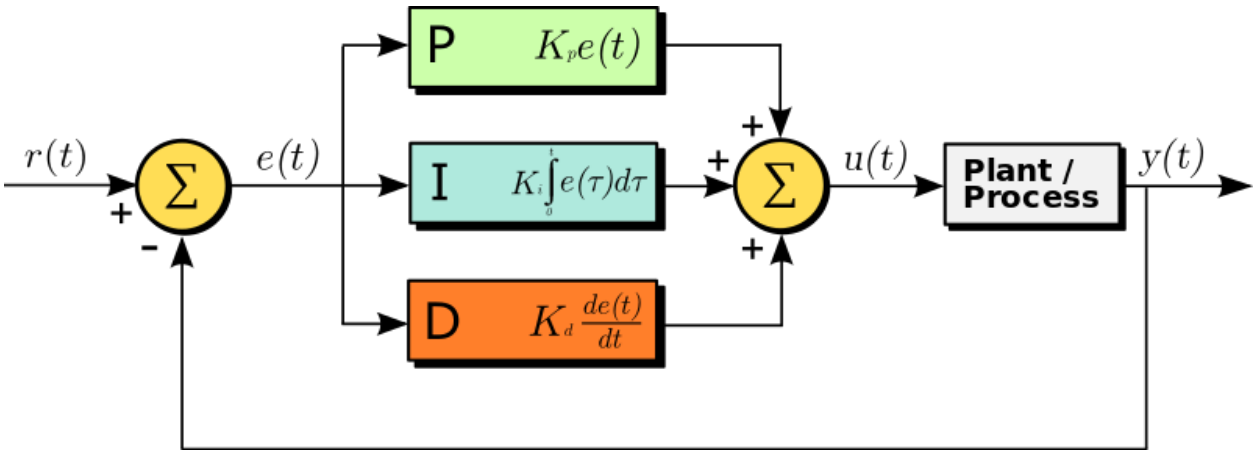
Notes on AO Servo Loop

Theo ten Brummelaar
July 10th 2018

This is a short note to make sure we are on the same page concerning the servo loop being used in the AO code. This is not a statement of why this loop is necessarily the best one, it may not be, but just a description of what has been implemented so far.

So, as taken from the Wikipedia page on PID loops we have:



where in our case the demand position r(t) is always 0, that is, a flat wavefront. The statement that we should start with a "pure integral", means that the K_p and K_d terms are zero and the K_i term is non-zero. This is 100% correct if the *Plant Process* does nothing and $y(t) = u(t)$.

The problem is that the code as it currently stands calculates a change in actuator position, not an absolute position. The plant process is $y(t) = \int_0^t u(t)\, dt$, or in other words $u(t) = \frac{dy(t)}{dt}$ and so we have a *velocity* servo not a *position* servo and I think this is where the confusion has arisen.

The currently implemented steps of the servo for each actuator are

1. Calculate the error e_n using the spot positions and the reconstructor matrix.

2. Using this error calculate how much we need to move the actuator ΔA_n to correct for this error using $\Delta A_n = K_p\, e_n - K_d\, \Delta A_{n-1} + K_i\, \Sigma_{i=0}^{i=n} e_i$.

3. The final actuator position is then $A_n = A_{n-1} + \Delta A_n$.

If we write this in analogue space, we have the differential equation

$$\frac{dA}{dt} = K_p e(t) - K_d \frac{dA(t)}{dt} + K_i \int_0^t e(t)dt.$$

In my mind this is the appropriate way of looking at this as each actuator is constantly moving, and so we need to control its velocity in order to have it arrive at the correct position before the next error measurement. If we integrate this term and only consider the first (proportional) term we have

$$A(t) = K_p \int_0^t e(t)dt$$

a "pure" integral servo. Pure-gain in a velocity servo is pure-integral in a position servo and there is no disagreement about what we should do, just a difference in nomenclature.

There is another part in the code, not related to this discussion. The change in actuator position in cycle n is, as before, given by

$$\Delta A_n = K_p\, e_n - K_d\, \Delta A_{n-1} + K_i\, \Sigma_{i=0}^{i=n} e_i.$$

However, the final actuator position is then calculated using

$$P_n = K_m A_{n-1} + \Delta A_n + F$$

where we have now added a *memory* term K_m, a zero point defined by the default flat F, and we always set $A_0 = 0.0$.

If we set $K_m = 1.0$ we have (Someone tell me if I'm wrong here!) the PID velocity control loop discussed above. If we set $0.0 < K_m < 1.0$ we have something that in the absence of any error terms will "relax" eventually to the default flat F.

In the code we have the defaults $K_m = 0.9$, $K_p = 0.5$, $K_i = 0.0$, and $K_d = 0.0$. This, apart from the memory part, is indeed a pure-integral servo as JB describes it, and a pure gain velocity servo as I (and the GUI) describe it.