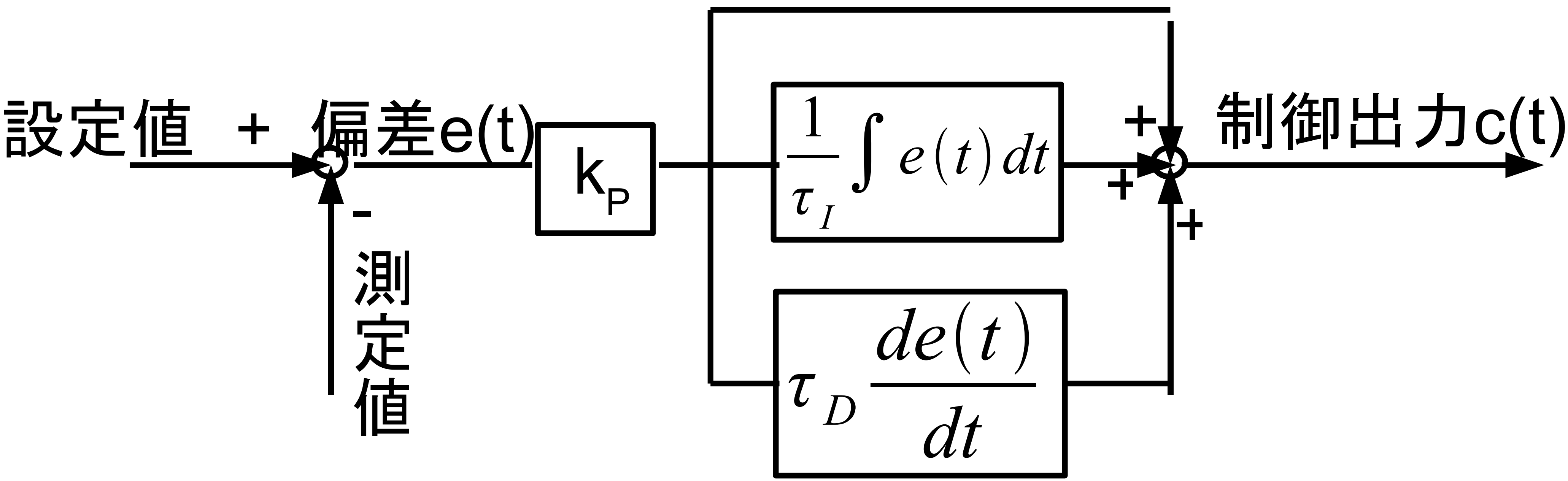
PID制御について

2007年8月30日
三好 修平

PID制御の特徴

1. 機能の完備性
   P: **現在**の偏差に比例した修正動作
   I: **過去**の偏差を積算保持しオフセットを取り除く動作
   D: **未来**の動きを予測する動作
2. 簡易性
   PIDパラメータの影響が定性的に理解し易い
3. 理論解析性
   PIDパラメータの影響の解析などが理論的に行える
4. 現場性
   システム設置後にPIDパラメータを調整する事が容易

PID制御の基本形



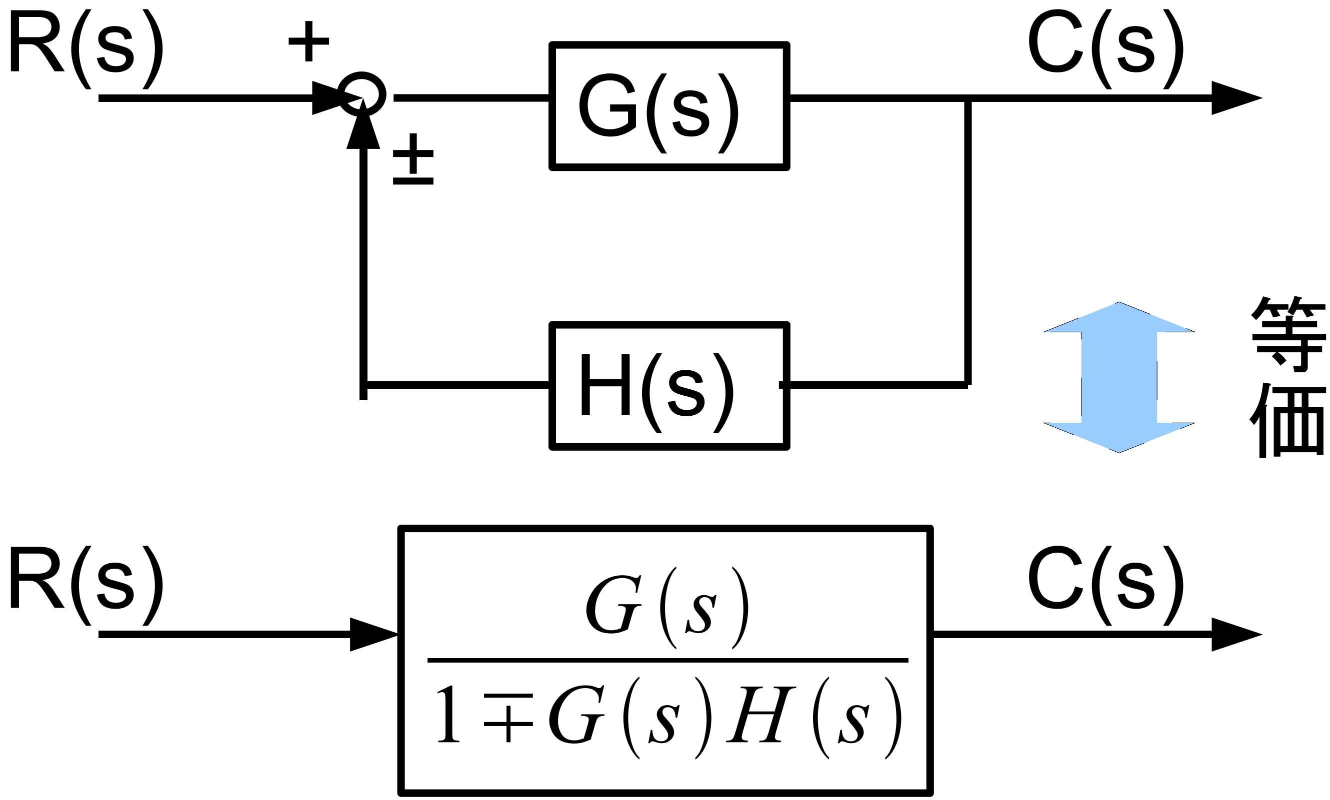
$$c(t) = k_P\left(e(t) + \frac{1}{\tau_I}\int e(t)\,dt + \tau_D\frac{de(t)}{dt}\right)$$

ラプラス変換すると $F(s) = \int_0^\infty f(t)e^{-st}\,dt$

$$C(s) = \boxed{k_P\left(1 + \frac{1}{\tau_I s} + \tau_D s\right)} E(s)$$

伝達関数

ここで



ラプラス変換の最終値定理

$$\lim_{t \to \infty} f(t) = \lim_{s \to 0} s \cdot F(s)$$

比例動作とP動作のみによる制御

比例動作は偏差に比例した出力を出す。

P動作のみのフィードバックを考えると、伝達関数は

$$\frac{k_P}{1+k_P}$$

大きさ1のステップを入力する(目標値を0→1にする)と、

$$C(s)=\frac{k_P}{1+k_P}\frac{1}{s}$$

最終値定理より

$$\lim_{t\to\infty} c(t)=\lim_{s\to 0} s\cdot C(s)=\frac{k_P}{1+k_P}$$

で偏差が残る。

この偏差を定常偏差と呼ぶ。

積分動作

一定の偏差が入って来た時、積分動作は

$$c(t) = \frac{1}{\tau_I}\int e\,dt = \frac{t}{\tau_I}e$$

と表せる。

$t = \tau_I$の時、出力と偏差が等しくなる。

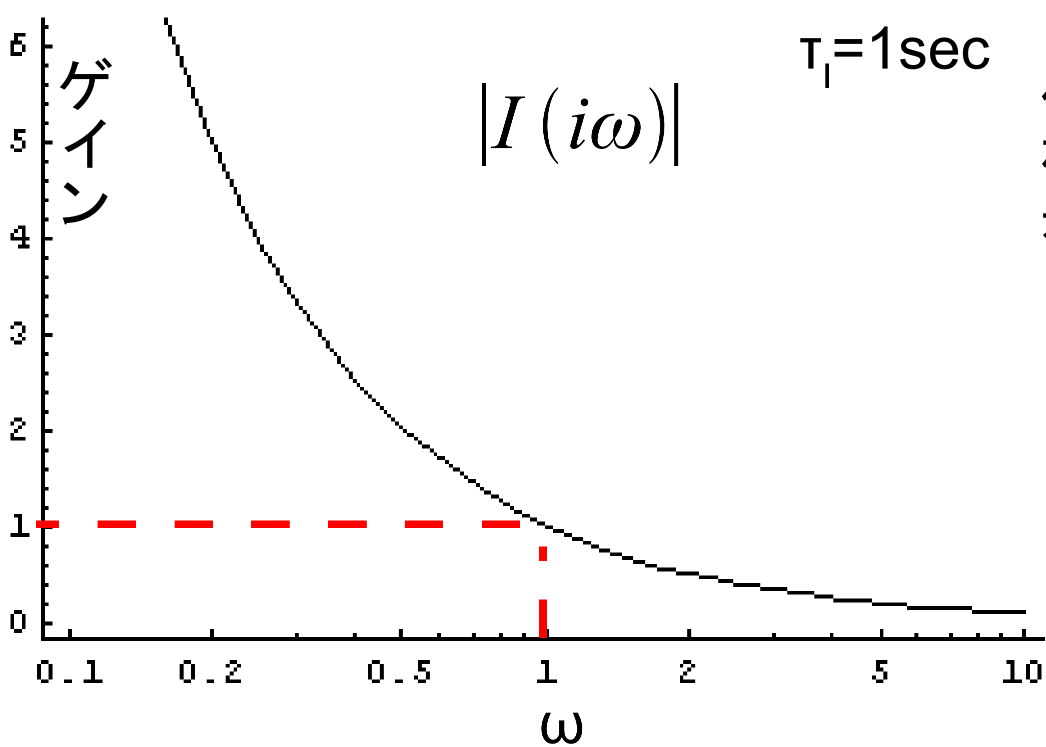
積分動作の出力が偏差に等しくなるまでかかる時間がτ_I

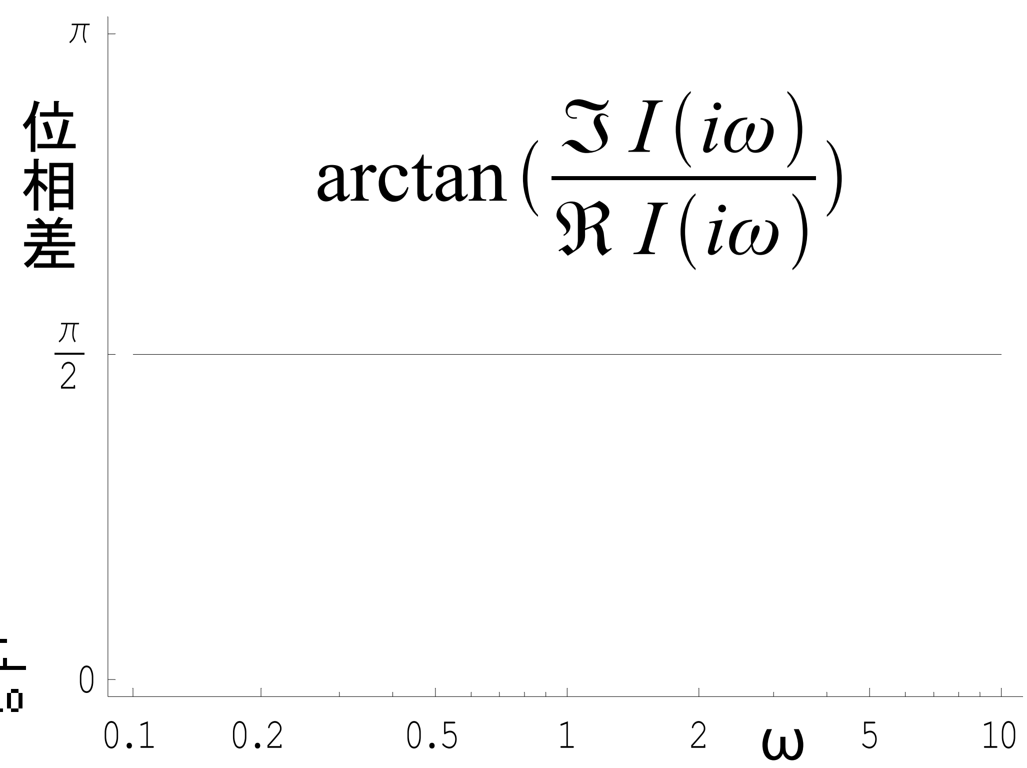
τ_Iが小さい程、積分動作は強くはたらく。

積分動作の周波数特性

伝達関数においてs=iωとおくことで、
伝達関数の周波数特性が分かる。

積分動作　$I(s)=\dfrac{1}{\tau_I s}$ の場合、出力は$1/\tau_I\omega$倍され、
入力に対して出力の位相はπ/2遅れる。





PI動作による制御

PI動作の伝達関数は $\dfrac{k_P(1+\dfrac{1}{\tau_I s})}{1+k_P(1+\dfrac{1}{\tau_I s})}$

P動作と同様にステップ応答を考えると

$$C(s)=\frac{k_P(1+\dfrac{1}{\tau_I s})}{1+k_P(1+\dfrac{1}{\tau_I s})}\frac{1}{s}$$

$$\lim_{t\to\infty} c(t)=\lim_{s\to 0} s\cdot C(s)=\frac{k_P/\tau_I}{k_P/\tau_I}=1$$

目標値=1

I動作は偏差を0にするはたらきを持つ。

微分動作

一定速度で変化する偏差が入って来た時、微分動作は

$$c(t)=\tau_D \frac{d}{dt} et=\tau_D e$$

と表せる。

$t = \tau_D$の時、入力と出力が等しくなる。

一定速度で変化する偏差が
微分動作の出力に等しくなるまでの時間がτ_D

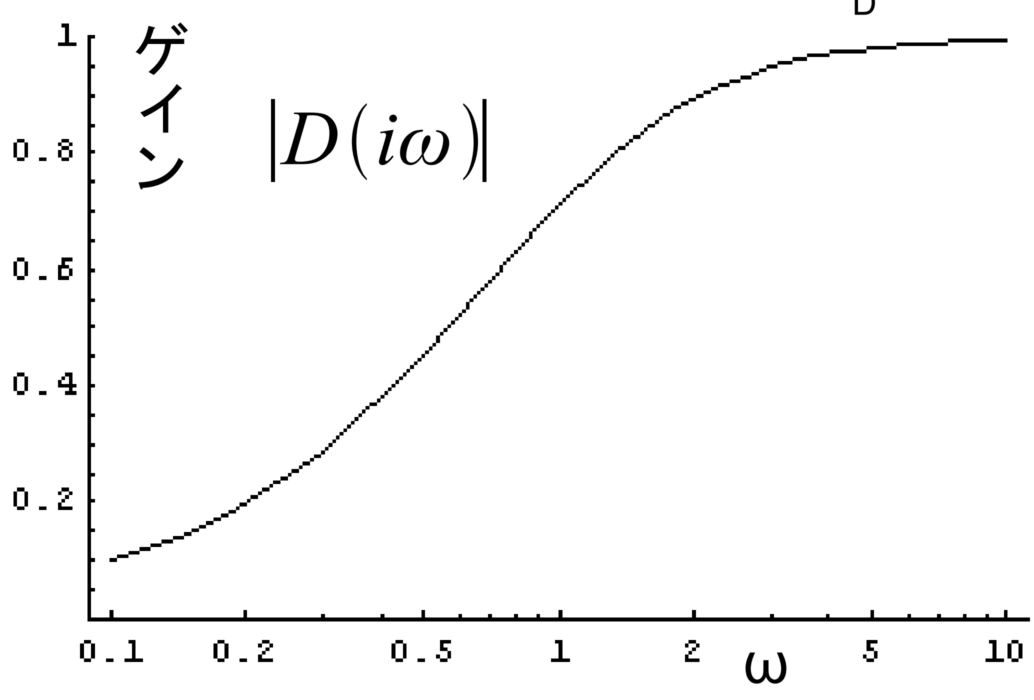
τ_Dが大きい程、微分動作は強くはたらく。

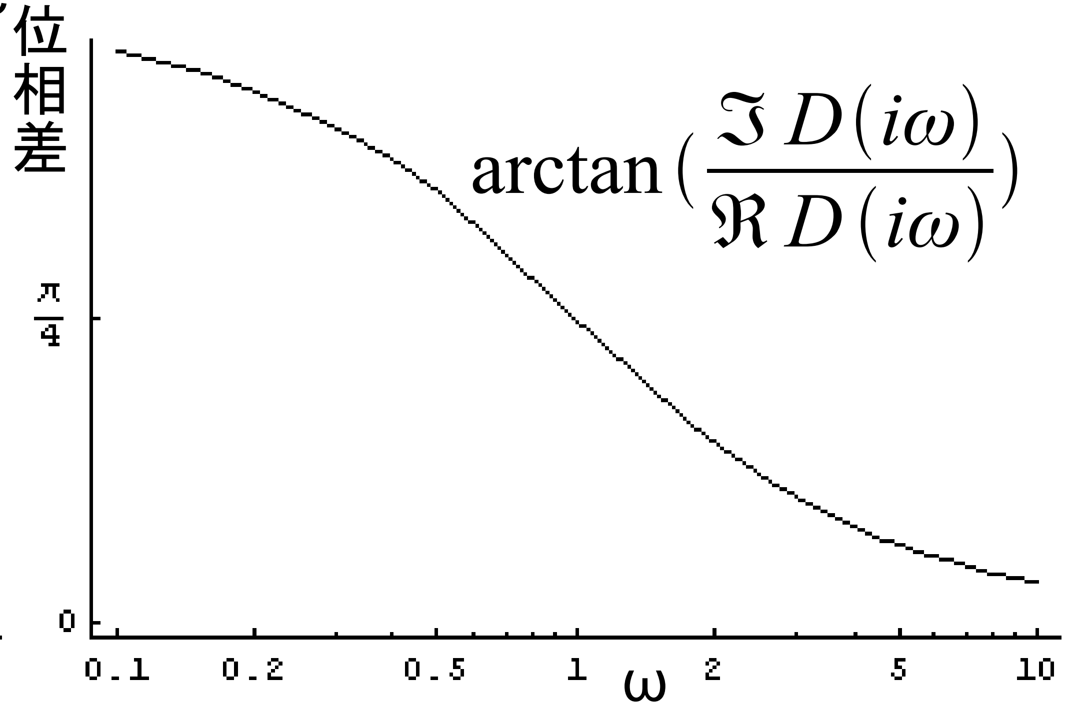
微分動作の周波数特性

微分動作伝達関数 $D(s) = \tau_D s$ は実現できない。
変わりに実用微分を用いる。

実用微分動作 $D(s) = \dfrac{\tau_D s}{1 + \tau_D s}$ の場合、
1/τ_Dよりωが大きくなる程ゲインが1に近付き、
位相差が0に近付く。

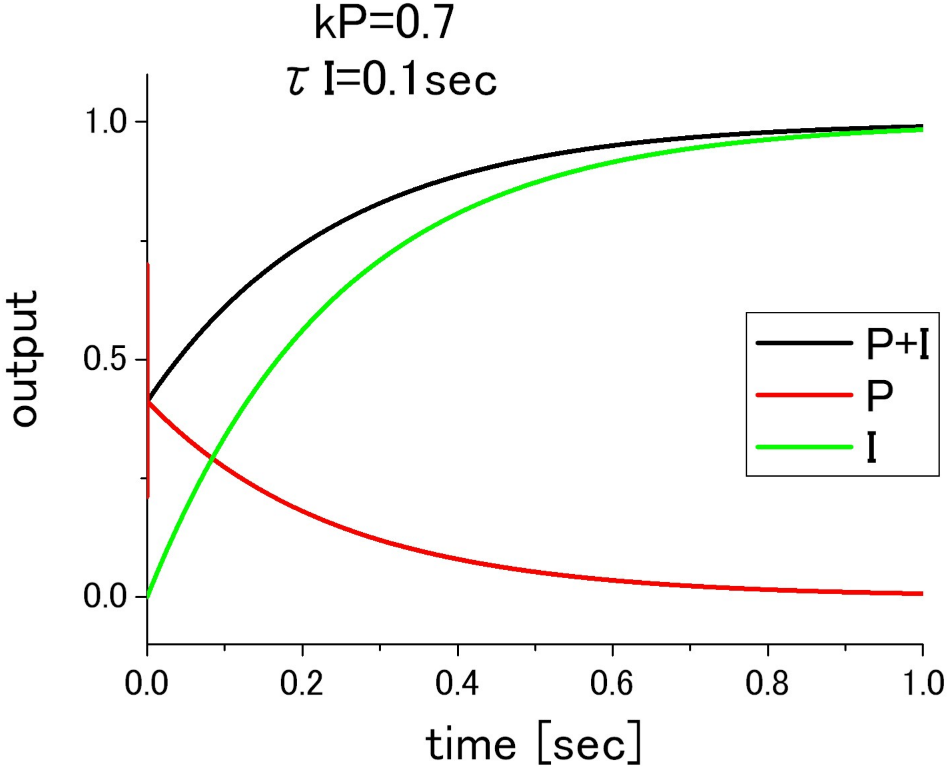
τ_D=1sec



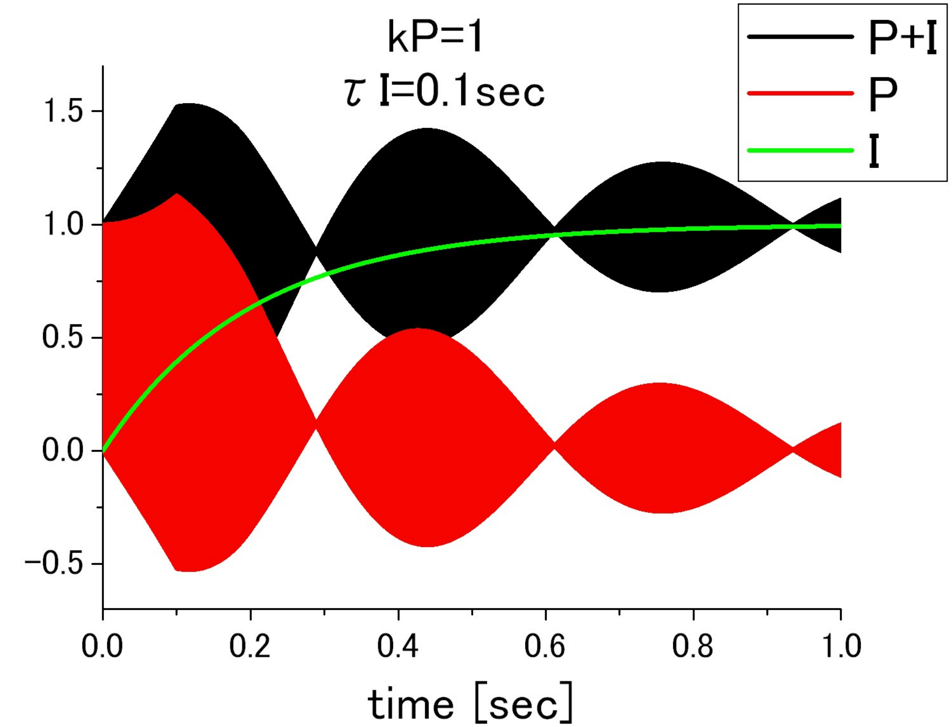


PI動作による制御

目標値1 $k_P(1+\frac{1}{\tau_I s})$



k_P=0.7



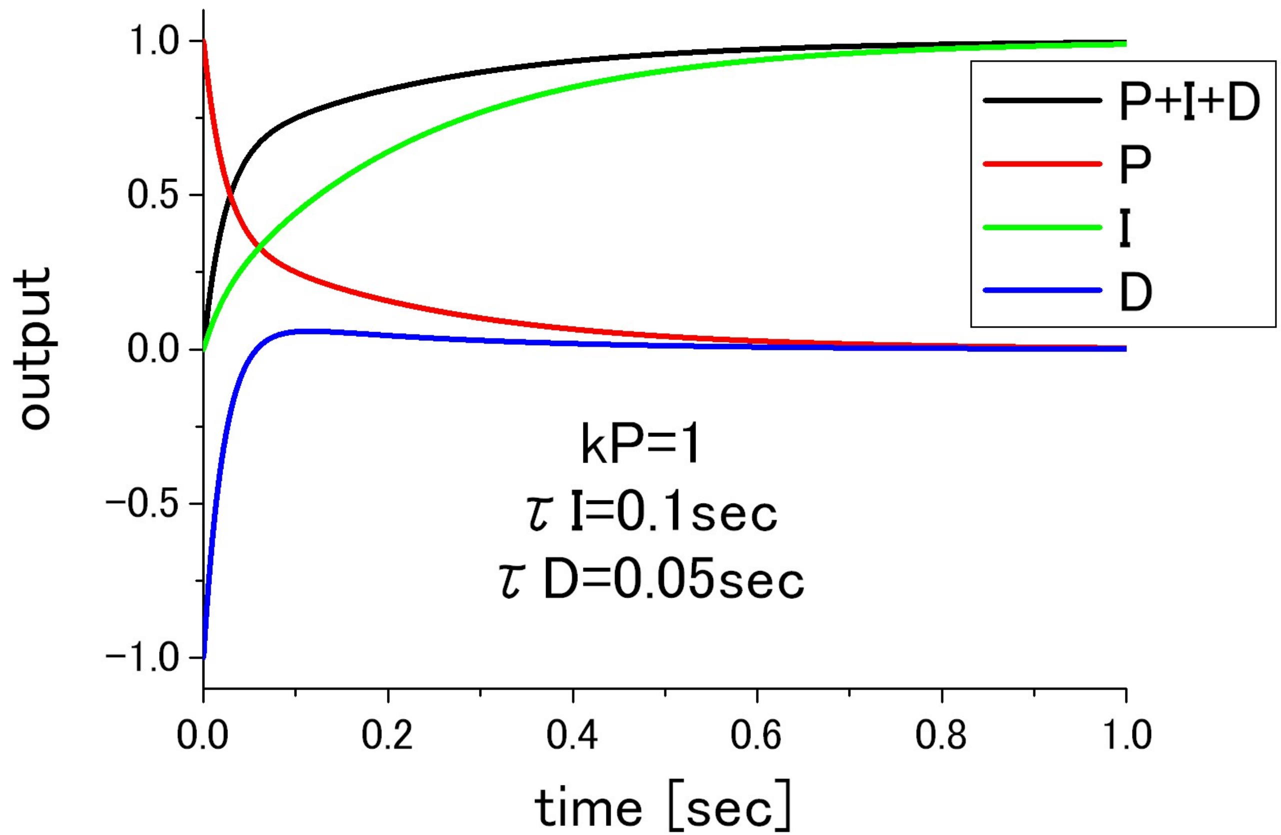
k_P=1

Pのゲインを上げると不安定になる。

PID動作による制御

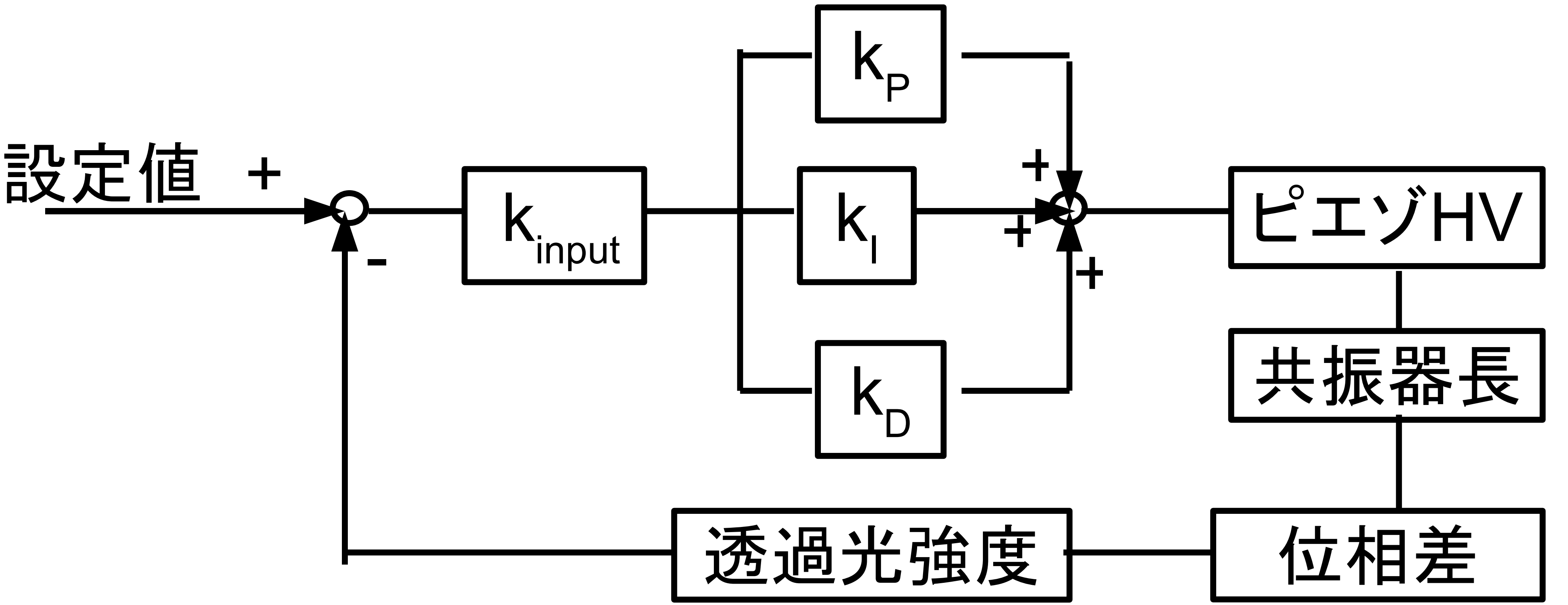
目標値1

$$k_P\left(1+\frac{1}{\tau_I s}-\frac{\tau_D s}{1+\tau_D s}\right)$$



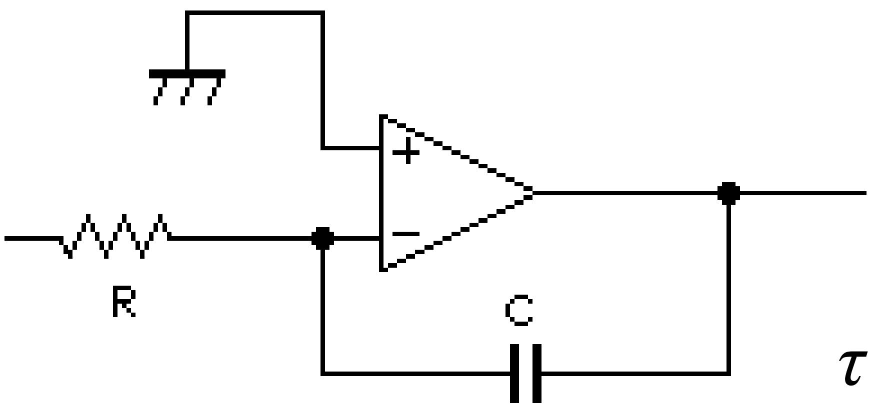
目標値を超えそうな時にD動作がブレーキをかけるので、
比例・積分ゲインを上げ易くなり、
応答速度を速くする事が出来る。

使っているFB系を簡単に描くと

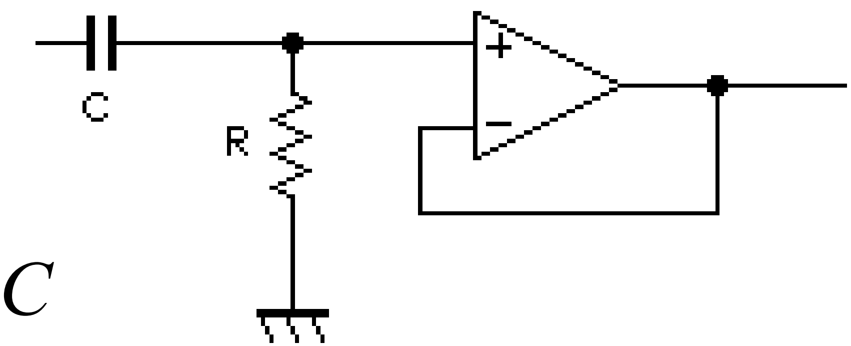


使われている積分回路と微分回路

積分回路



微分回路



$$\tau = RC$$

$$v_{output}(t) = -\frac{1}{\tau_I}\int v_{input}(t)\,dt$$

$$V_{output}(s) = -\frac{1}{\tau_I s} V_{input}(s)$$

$$v_{input}(t) = \frac{Q}{C} + RI$$

$$v_{output}(t) = RI$$

$$V_{output}(s) = \frac{\tau_D s}{1+\tau_D s} V_{input}(s)$$

PID部分だけなら伝達関数はこのようになる。

$$V_{output}(s)=k_{input}\left(k_P-\frac{1}{\tau_I s}+\frac{\tau_D s}{1+\tau_D s}\right)V_{input}(s)$$

この式を逆ラプラス変換すれば
スマートにフィードバックのシミュレーションが出来るかと
考えましたが、
結局以前の数値計算による
シミュレーションの方が良いと考えました。