進捗報告資料

安達智哉
to-adachi@ist.osaka-u.ac.jp

2019 年 12 月 9 日

1　ジーグラニコルスの限界感度法

PID 制御においては、比例ゲイン (K_p)、積分ゲイン (K_i)、微分ゲイン (K_d) と呼ばれる 3 つの定数を設定する必要がある。これらの定数を “ジーグラ・ニコルスの限界感度法” と呼ばれる手順に基づき設定した。

ジーグラ・ニコルスの限界感度法に基づくゲインの求め方を以下に示す。

ステップ 1 積分ゲイン (K_i) および微分ゲイン (K_d) を 0 にして調節器が比例動作だけを行うようにする。

ステップ 2 比例ゲイン (K_p) を 0 から徐々に大きくしていき、制御量が安定限界に達して一定振幅振動を持続するようになった時に K_p の増加を止める。

ステップ 3 ステップ 2 の時の比例ゲインを限界感度 (K_u)、振動周期を限界周期 (P_u) とし、これらの値から各ゲインを以下の表 1 のように求める。

ステップ 4 必要に応じてステップ 3 で求めた各ゲインの値を調整する。

以下のシナリオにおいて、ジーグラニコルスの限界感度法を用いて、PID 制御のゲインを求めた。UE 台数は 648,000 台であり、UE の持つ通信周期は 1 day，2 hours，1 hour，30 minutes のいずれかである。それぞれの通信周期を持つ UE の割合は表 3 の通りである。また、各パラメータを表 2 に示す。

限界感度および限界周期を求めるために、K_i および K_d を 0 にして、K_p を 0 から徐々に大きくしていった。その結果、$K_p = 0.53$ の場合は制御が収束せず、一定振幅振動することがわかった。このことから、限界感度 (K_u) を 0.53 とする。また、限界周期 (P_u) は 7450 s とする。

以上の結果を用いて、各ゲインの値は以下の表 4 のように求まる。

表 1: ジーグラ・ニコルスの限界感度法

制御の種類	K_p	K_i	K_d	T_i	T_d
P	$0.5K_u$	0	0	-	-
PI	$0.45K_u$	$K_p/0.83P_u$	0	$0.83P_u$	-
PID	$0.6K_u$	$K_p/0.5P_u$	$K_p \cdot 0.125P_u$	$0.5P_u$	$0.125P_u$

表 2: パラメータ設定

Parameter	Numerical setting
T^{ci}	10 s
$s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{c}}$	0 messages
$s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{ci}}$	0 messages
$s_{\mathrm{MME}}^{\mathrm{c}\to\mathrm{ci}}$	0 messages
$s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{c}}$	0 messages
$s_{\mathrm{MME}}^{\mathrm{ci}\to\mathrm{i}}$	5 messages
$s_{\mathrm{MME}}^{\mathrm{i}\to\mathrm{c}}$	5 messages
$m_{\mathrm{MME}}^{\mathrm{c}}$	17878 bits
$m_{\mathrm{MME}}^{\mathrm{ci}}$	17878 bits
$m_{\mathrm{MME}}^{\mathrm{i}}$	408 bits
$C^{\max}$	1200 messages/s
$M^{\max}$	1,000 MB
d_h	1

表 3: UE の通信周期の分布

	通信周期			
	1 day	2 hours	1 hour	30 minutes
UE 台数の割合	40%	40%	15%	5%

表 4: ジーグラ・ニコルスの限界感度法に基づく設定

制御の種類	K_p	K_i	K_d	T_i	T_d
P	0.265	0	0	-	-
PI	0.2385	0.0000463	0	6183.5	-
PID	0.318	0.0000854	296.14	3725	931.25

表 4 に示した P 制御の値を K_p、K_i および K_d にそれぞれ設定した場合の評価結果を図 1 に示す。

(a) $e(t)$ の変化 ($K_p = 0.265$、$K_i = 0$、$K_d = 0$)

(b) IdleTimer の変化 ($K_p = 0.265$、$K_i = 0$、$K_d = 0$)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.265$、$K_i = 0$、$K_d = 0$)

(d) 各状態にある UE 台数の変化 ($K_p = 0.265$、$K_i = 0$、$K_d = 0$)

図 1

表 4 に示した PI 制御の値を K_p、K_i および K_d にそれぞれ設定した場合の評価結果を図 2 に示す。

(a) $e(t)$ の変化 ($K_p = 0.2385$、$K_i = 0.0000463$、$K_d = 0$)

(b) IdleTimer の変化 ($K_p = 0.2385$、$K_i = 0.0000463$、$K_d = 0$)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.2385$、$K_i = 0.0000463$、$K_d = 0$)

(d) 各状態にある UE 台数の変化 ($K_p = 0.2385$、$K_i = 0.0000463$、$K_d = 0$)

図 2

表 4 に示した PID 制御の値を K_p、K_i および K_d にそれぞれ設定した場合の評価結果を図 3 に示す。

(a) $e(t)$ の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$)

(b) IdleTimer の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$)

(d) 各状態にある UE 台数の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$)

図 3

表 4 に示した PID 制御の値を、K_p および K_d にそれぞれ設定し、K_i に 0 を設定した場合の評価結果を図 4 に示す。

(a) $e(t)$ の変化 ($K_p = 0.318$、$K_i = 0$、$K_d = 296.14$)

(b) IdleTimer の変化 ($K_p = 0.318$、$K_i = 0$、$K_d = 296.14$)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.318$、$K_i = 0$、$K_d = 296.14$)

(d) 各状態にある UE 台数の変化 ($K_p = 0.318$、$K_i = 0$、$K_d = 296.14$)

図 4

以上の図 1、2 図 3 を比較すると、図 1 に示した P 制御が最も制御が安定していることがわかる。また、$E(t)$ が 0 に収束するまでの時間も最も短くなっている。以上の結果より、比例ゲインに 0.265 を設定した P 制御が Idle タイマの制御に最も適していると言える。

一方で、PI 制御は、P 制御と比較して良い結果が得られなかった。積分ゲインを利用しても制御が改善しなかった理由は、比例ゲインのみで制御した場合に発生するオフセット（定常偏差）が小さいことがあげられる。オフセットとは、定常状態の時に出力値と目標値の差である。一般的にオフセットを 0 に収束させるために積分制御が用いられるが、今回評価している Idle タイマの制御では、元々オフセットが小さい。よって、積分制御を導入するのメリットが小さくなっている。

また、PID 制御では、微分ゲインを利用することにより、偏差発生から定常状態に至るまでの過渡応答特性を改善することができた。これは一般的に微分ゲインを導入するメリットの一つである。しかし、一般的に、微分制御は、「対象の変化」ではなく、ノイズにより過敏に反応する危険性がある。本評価では、離散的な負荷の変動が発生しているため、Idle タイマの制御が不安的になっている。

2　実用 PID 制御についての調査

第 1 章の評価において、PID 制御の微分動作が離散的な負荷の変動の影響を受けやすいため、今回の評価においては制御が不安定になることがわかった。通常の微分 (完全微分) を用いて動作する PID 制御のことを “理想 PID 制御” とよび、以下の式 (1) で表せる。以下の図 5 に、理想 PID 制御のブロック図を示す。

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)d\tau + K_d \cdot \frac{de(t)}{dt} \tag{1}$$

ここで、積分ゲイン $K_i = K_p/T_i$、微分ゲイン $K_d = K_p \cdot T_d$ と表し、式 (1) に代入すると、式 (2) になる。

$$u(t) = K_p \cdot (e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau)d\tau + T_d \cdot \frac{de(t)}{dt}) \tag{2}$$

式 (2) をラプラス変換すると式 (3) になり、伝達関数 $C(s)$ は式 (4) となる。

$$\begin{aligned} \mathcal{L}[u(t)] &= \mathcal{L}[K_p \cdot (e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau)d\tau + T_d \cdot \frac{de(t)}{dt})] \\ U(s) &= K_p \cdot (1 + \frac{1}{T_i \cdot s} + T_d \cdot s) \cdot E(s) \end{aligned} \tag{3}$$

$$C(s) = \frac{U(s)}{E(s)} = K_p \cdot (1 + \frac{1}{T_i \cdot s} + T_d \cdot s) \tag{4}$$

(a)　理想PID制御の機能ブロック構成

図 5: 理想 PID 制御のブロック図

理想 PID 制御の持つ、ノイズに弱いという欠点を解決するために、ノイズを抑制するローパスフィルタ (一次遅れフィルタ) を組み込んだ制御を実用 PID 制御という。実用 PID 制御のブロック図を以下の図 6 に示す。実用 PID 制御を伝達関数で表現すると、以下の式 (5) になり、ブロック図は図 7 のようになる。このように、フィルタを導入した微分制御のことを不完全微分という。ここで η は微分係数という定数であり、通常 0.1 から 0.125 の値を設定する [1]

図 6: 実用 PID 制御のブロック図

$$C(t) = \frac{U(s)}{E(s)} = K_p \cdot \{1 + \frac{1}{T_i \cdot s} + \frac{T_d \cdot s}{1 + \eta \cdot T_d \cdot s}\} \tag{5}$$

図 7: 実用 PID 制御のブロック図 (伝達関数での表現)

式 (5) に示した伝達関数のうち、微分制御に関する部分を抽出した式を以下の式 (6) に示す。

$$U_d(s) = \frac{T_d \cdot s}{1 + \eta \cdot T_d \cdot s} \cdot E(s) \tag{6}$$

式 (6) を式 (7) のように式変形する。

$$\begin{aligned} U_d(s) \cdot (1 + \eta \cdot T_d \cdot s) &= T_d \cdot s \cdot E(s) \\ U_d(s) + \eta \cdot T_d \cdot s \cdot U_d(s) &= T_d \cdot s \cdot E(s) \end{aligned} \tag{7}$$

そして、逆ラプラス変換を行うことにより、通常の微分方程式になる (8)。

$$\begin{aligned} \mathcal{L}^{-1}[U_d(s) + \eta \cdot T_d \cdot s \cdot U_d(s)] &= \mathcal{L}^{-1}[T_d \cdot s \cdot E(s)] \\ u_d(t) + \eta \cdot T_d \cdot \frac{d}{dt} u_d(t) &= T_d \cdot \frac{d}{dt} e(t) \\ u_d(t) &= T_d(\frac{d}{dt} e(t) - \eta \cdot \frac{d}{dt} u_d(t)) \end{aligned} \tag{8}$$

以上の議論より、実用 PID 制御は以下の式で表すことができる。(※右辺に $u_d(t)$ が残っており、微分方程式になっている。そのため、プログラムで実装する際には、後進差分近似を行った上で微分方程式を解く必要がある (補足))

$$u(t) \quad = \quad K_p \cdot \{e(t) + \frac{1}{T_i} \cdot \int_0^t e(\tau)d\tau + T_d(\frac{d}{dt}e(t) - \eta \cdot \frac{d}{dt}u_d(t))\} \tag{9}$$

3 移動平均、実用 PID 制御

表 4 に示した PID 制御の値を K_p、K_i および K_d にそれぞれ設定し、シグナリング頻度 ($s(t)$) を指数移動平均で平滑化した値 ($\overline{s(t)}$) を PID 制御への入力とした場合の評価結果を図 8 に示す ($\alpha = 0.125$)。

$$\overline{s(t)} = \alpha \cdot s(t) + (1-\alpha) \cdot \overline{s(t-1)} \tag{10}$$

(a) $e(t)$ の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.125$)

(b) IdleTimer の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.125$)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.125$)

(d) 各状態にある UE 台数の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.125$)

図 8

表 4 に示した PID 制御の値を PID 制御の値を K_p、K_i および K_d にそれぞれ設定し、ローパスフィルタを用いた実用 PID 制御を行った場合の評価結果を図 9 に示す。

(a) $e(t)$ の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、実用 PID)

(b) IdleTimer の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、実用 PID)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、実用 PID)

(d) 各状態にある UE 台数の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、実用 PID)

図 9

図 8 および図 9 の結果より、指数移動平均や実用 PID 制御を用いて入力の離散的な変動を平滑化することによって、そのようにしない場合 (図 3) と比較して制御が安定することがわかる。しかし、依然として Idle タイマの制御が十分安定しているとは言えない。指数移動平均の荷重 (α) や実用 PID 制御における微分係数 (η) の値を変更しても同様に、Idle タイマが安定することはなかった。

図 8 に示した Idle タイマの変化とシグナリング頻度の変化をタイムステップ 50,000 から 55,000 の範囲でプロットしたグラフを図 10 に示す。この図を見ると、Idle タイマおよびシグナリング頻度は周期的に変動していることがわかる。また、それぞれの変動のタイミングおよび周期が一致していることもわかる。これは、シグナリング頻度の変化に Idle タイマを追従させるように PID 制御が機能しているためである。シグナリング頻度が周期的に変化している理由は、Idle タイマの変化によって、UE の状態遷移が集中する期間が周期的に発生するためである。図中の左側に 2 本の破線と 2 本の直線を引いた。2 本の破線に囲まれた期間はシグナリング頻度が大きく、Idle タイマが約 3,500 s に設定されている。一方、2 本の直線に囲まれた期間はシグナリング頻度が小さく、Idle タイマは約 3,100 s に設定されている。すると、破線で示した期間にデータ送信を行った UE は、約 3,500 s 後に Connected Inactive 状態からアイドル状態への状態遷移を発生させる。同様に、直線で示した期間にデータ送信を行った UE は、約 3,100 s 後に状態遷移を発生させる。このタイミングを図中の右側の破線および直線に示す。右側の破線および直線は左側の破線および直線のそれぞれ 3,500 s、3,100 s 後を示している。これを見ると、UE の状態遷移のタイミングが重なっていることがわかる。また、それが原因となり、シグナリング頻度が大きくなっている。上述のように、シグナリング頻度の変動が、一定期間後に同様のシグナリング頻度の変動を引き起こす場合がある。

図 10: Idle タイマと CPU 負荷の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、移動平均)

移動平均に関する式 (10) における α の値を様々に変化させて実験を行った。結果を以下の図**??**に示す。これらの結果を見ると α の値が小さい場合、Idle タイマが小さい周期で変動が安定することがわかる。一方で、オーバシュートの発生により、周期の大きな変動が発生していることがわかる。

(a) $e(t)$ の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.01$)

(b) IdleTimer の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.01$)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.01$)

(d) 各状態にある UE 台数の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.01$)

図 11

(a) $e(t)$ の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.005$)

(b) IdleTimer の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.005$)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.005$)

(d) 各状態にある UE 台数の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.005$)

図 12

(a) $e(t)$ の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.003$)

(b) IdleTimer の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.003$)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.003$)

(d) 各状態にある UE 台数の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.003$)

図 13

(a) $e(t)$ の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.001$)

(b) IdleTimer の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.001$)

(c) CPU 負荷とメモリ使用量の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.001$)

(d) 各状態にある UE 台数の変化 ($K_p = 0.318$、$K_i = 0.0000854$、$K_d = 296.14$、指数移動平均 $\alpha = 0.001$)

図 14

4 今後の予定

- PID 制御に関する学習
- リソースが不足した際の制御

参考文献

[1] “実用 PID に向けての工夫（その１）.” [Online]. Available: https://www.nikko-pb.co.jp/products/k_data/28P_31P_13.pdf

5 補足

$$
\begin{aligned}
u_d(t) &= T_d(\frac{d}{dt}e(t) - \eta \cdot \frac{d}{dt}u_d(t)) \\
u_d(t) &= T_d \cdot (e(t) - e(t-1)) - \eta \cdot T_d \cdot (u_d(t) - u_d(t-1)) \\
u_d(t) + \eta \cdot T_d \cdot u(t) &= T_d \cdot \Delta e(t) + \eta \cdot T_d \cdot u_d(t-1) \\
u_d(t) &= \frac{T_d}{1+\eta \cdot T_d} \cdot \Delta e(t) + \frac{\eta \cdot T_d}{1+\eta \cdot T_d} \cdot u_d(t-1) \qquad (11) \\
*\Delta e(t) &= e(t) - e(t-1)
\end{aligned}
$$