速度制御のためのシステム同定実験

2022/06/22

Cy20202 稲葉周

1.概要

第三回目のプロジェクト演習は MATLAB、Simulink を用いたフィードバック制御やステップ応答法によるパラメータ同定、時定数とゲインの同定を行う

2.フィードバック制御・比例制御

フィードバック制御を MATLAB 及び Simulink を用いてシミュレーションしていく。実際に使用するブロック図は図 2.1 である。使用する回路は前回に引き続き図 2.2 のような回路を使用する。Kp に 8 を代入し、実行すると得られた波形は図 2.3a である。一方で Kp に 16 を代入し実行して得られた波形は図 2.3b である。Kp=8 の時に比べて Kp=16 は波形が細かく、入力信号をなぞるような動きになっている。一方で Kp＝8 は細かくなく、入力信号に対してあまり同じような動きはしていなかった。

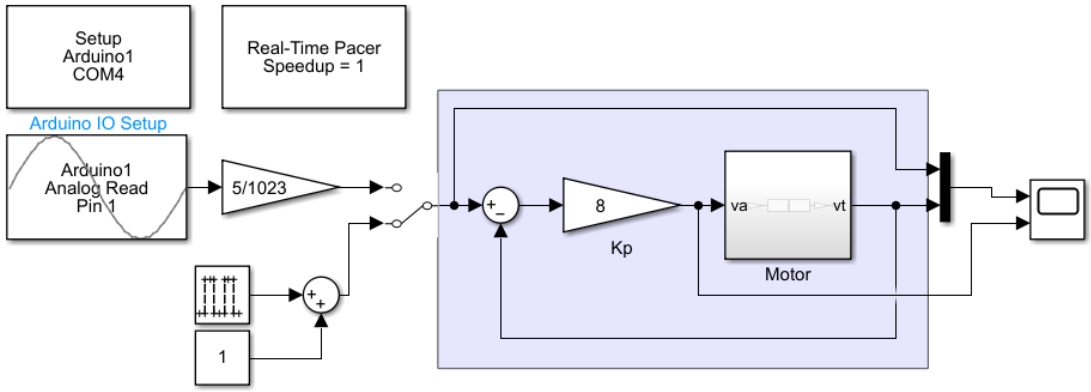


図 2.1　ブロック図

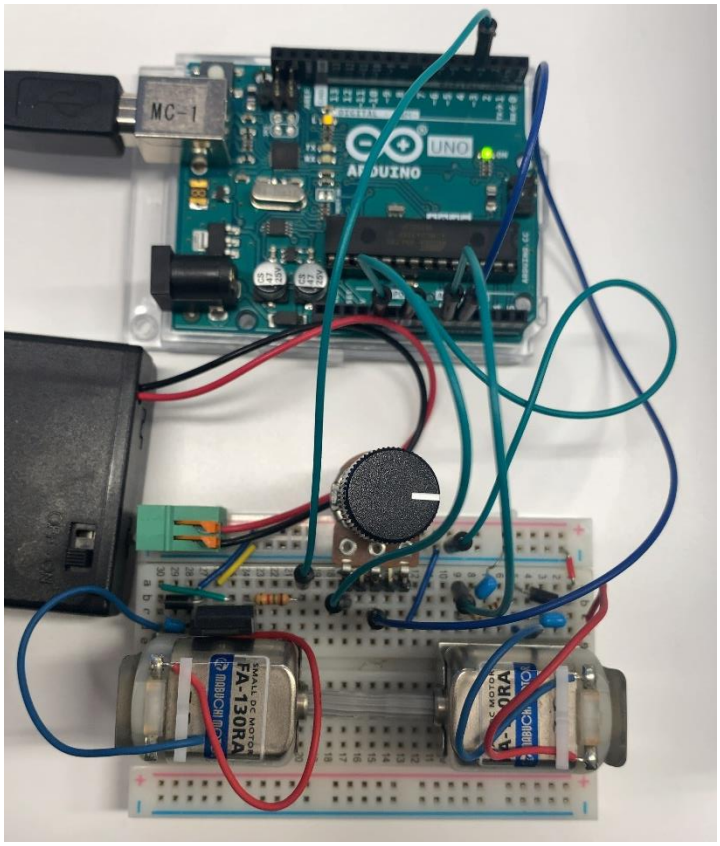


図 2.2　回路図

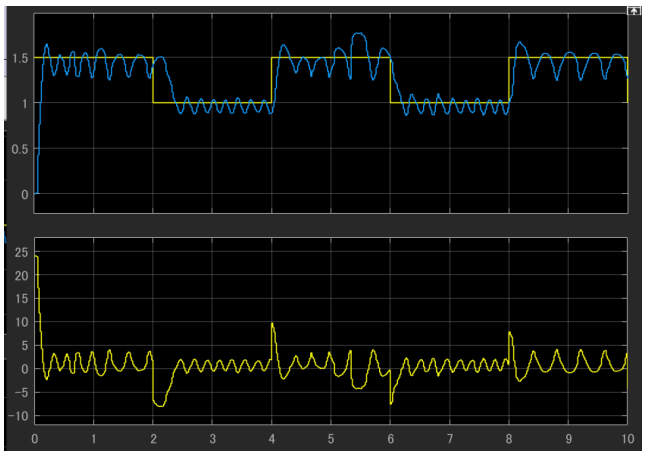
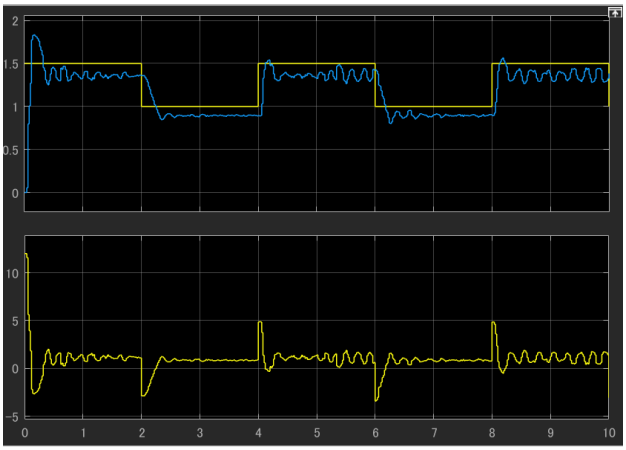


図 2.3 a Kp=8



b Kp=16

3.積分制御系

このセクションでは積分制御系を用いた制御を行う。比例制御では定常偏差が大きい。それから定常偏差を小さくしようと比例ゲインをを大きくすると振動的になってしまう。比例制御では制御入力は偏差の比例ゲイン倍になる。定常偏差を 0 にしようとすると電圧は常にかかり続けるため不可能である。そこで積分制御を用いる。積分制御を用いれば、現時点で偏差が 0 であっても、制御入力は 0 ではないという状況を作ることができる。

積分制御をブロック図で表したものが図 3.1 である。回路は図 1.2 のような回路を使用する。安定になる比例ゲイン Kp や積分ゲイン Ki を探していく。自分は安定になる比例ゲイン Kp＝8、積分ゲイン Ki=2 ということになった。その時に実行した波形は図 3.2 である。様々な比例ゲインと積分ゲインの組み合わせを試したが、自分のモーター回路はあまり安定しているとわかりやすい波形は得られなかった。

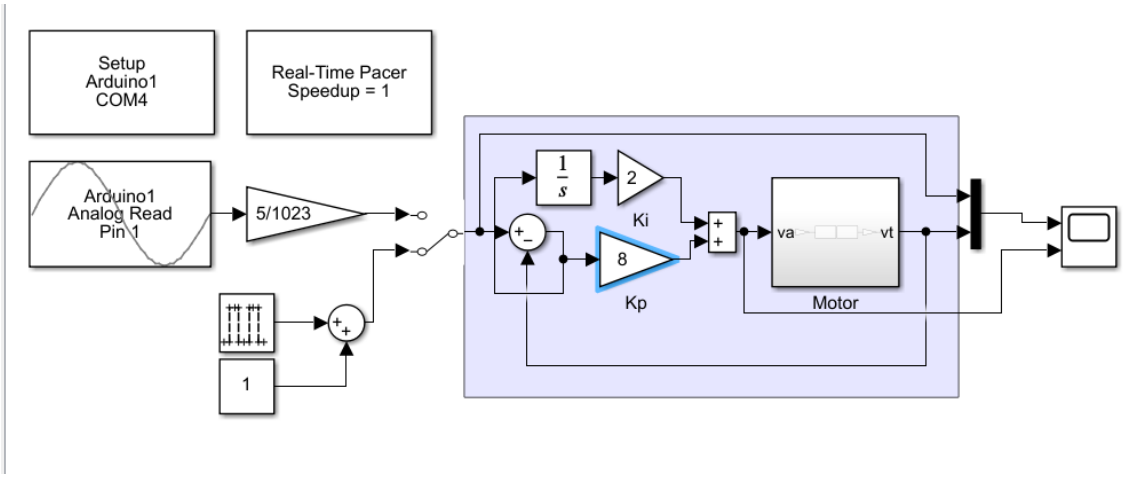


図 3.1　積分制御のブロック図

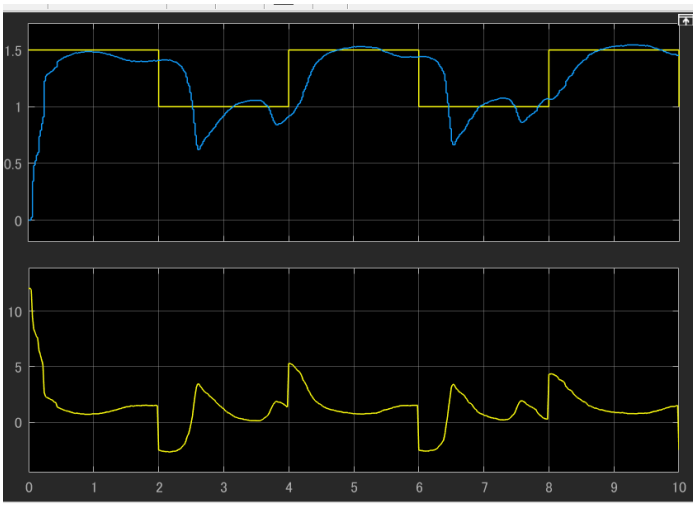


図 3.2 Ki=4,Kp=2

4.伝達関数

システムの中で線形であり、時間とともに特性が変わらないものを線形時不変システムと呼ぶ。伝達関数は入力信号と出力信号をラプラス変換によって関数の次元を変えている。図 4.1 は伝達関数を使用してみたブロック図である。実行すると図 4.3 が得られる。

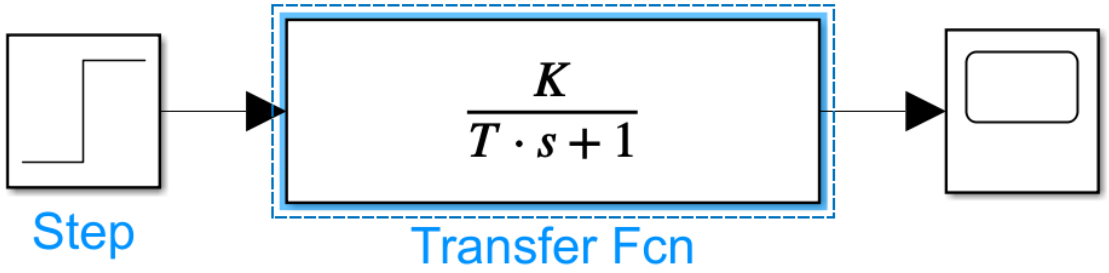


図 4.1　Transfer ブロックの使い方

図 4.2　Transfer のプロパティ

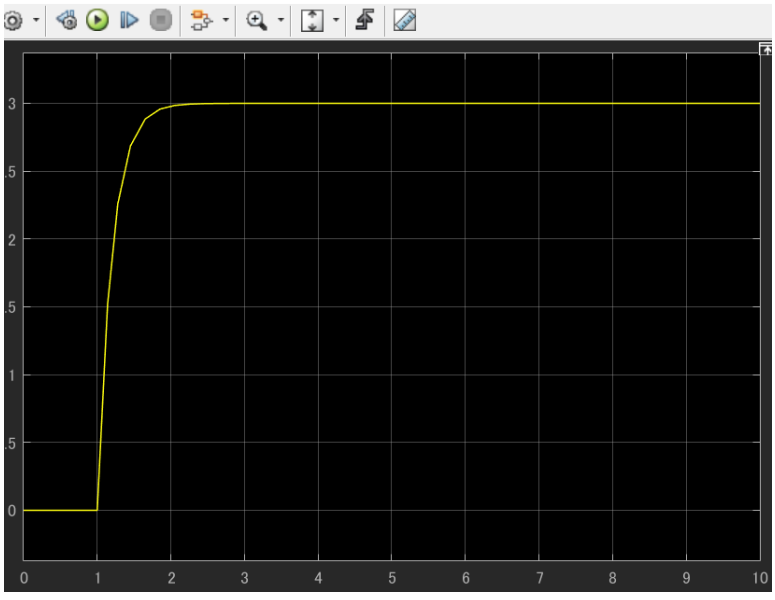


図 4.3 図 4.2 の実行結果の波形

5.時定数 T とゲイン K の同定

ステップ応答法により、K と T をも求める。使用するプログラムは付録 1 に記載している。ブロック図は図 5.1 である。実行結果は図 5.3a のようになった。そしてステップ応答の取得データは図 5.4 のようになり、ステップ応答の取得データは図 5.5、時定数の計算は図 5.5 のようになり、K、T、u_offset の値は図 5.6 のようになっていた。

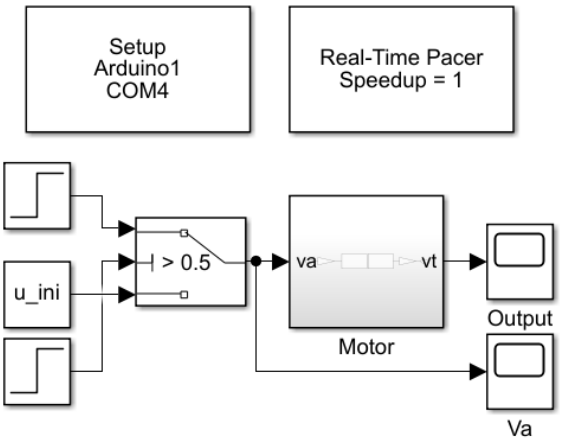


図 5.1 ステップ応答法による同定実験のための Simulink モデル

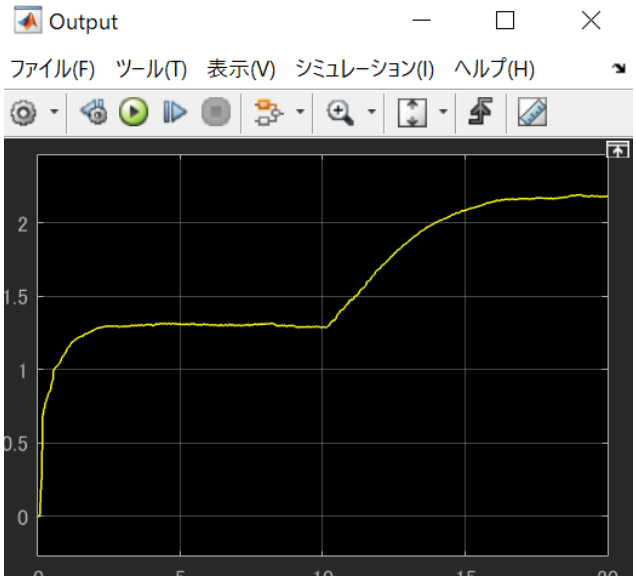
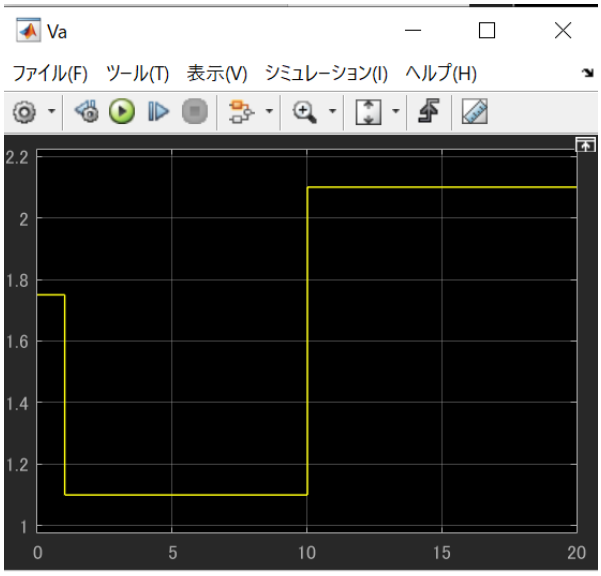


図 5.3a　Output 側の波形



b　ステップ波形

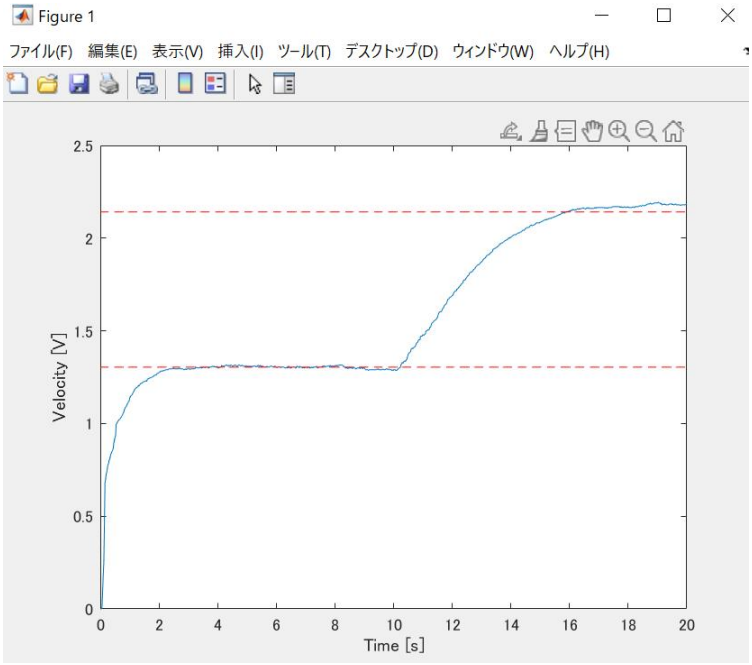


図 5.4　ステップ応答

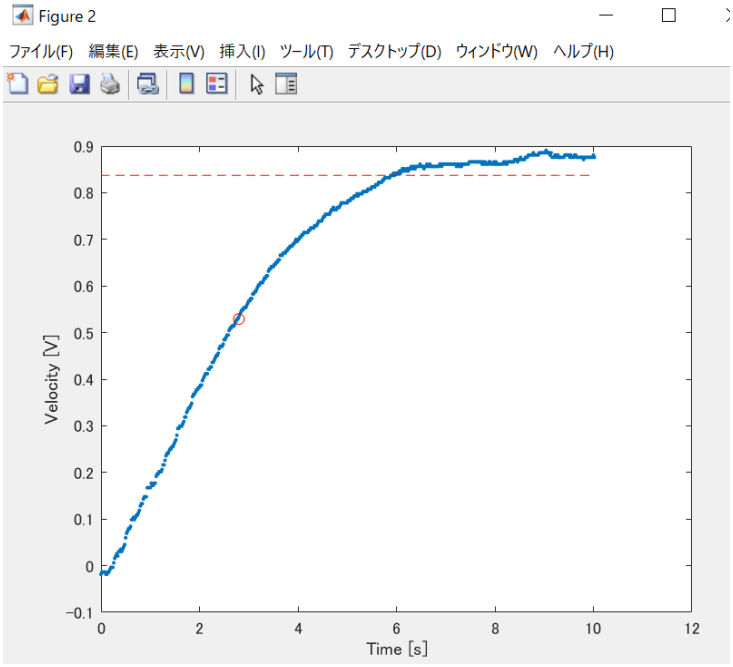


図 5.5　ステップ応答（時定数の計算）

```
== Results==
K          =0.837189
T          =2.800000
u_offset = -0.457393
>>
```

図 5.6　出力ログ

6.同定結果の検証

5 章で行った同定結果の検証を行う。使用するブロック図は図 6.1。検証で用いるプログラムは付録２、実行結果は

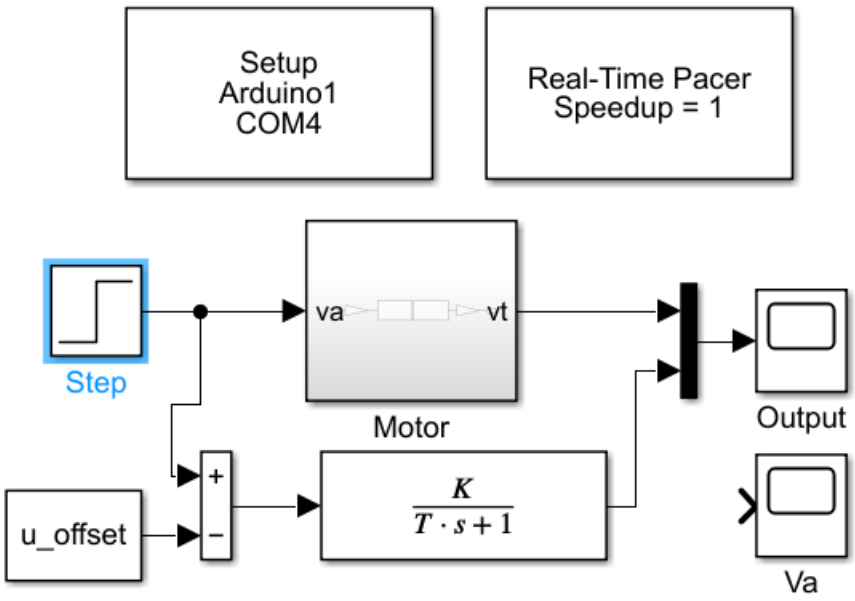


図 6.1　同定結果の検証のためのブロック図

図 6.2　検証実験結果

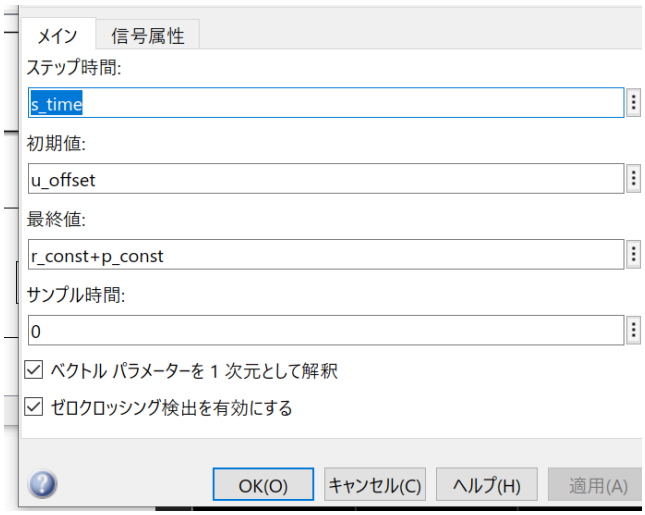


図 6.3　ステップ信号のプロパティ

図 6.2 のようになった。あまり一致はしてないようにみえる結果になった。

参考文献

[1] 平田光男, 『Arduino と MATLAB で制御系設計をはじめよう！』, TechShare ,2012,

```
%%velo_id_tc.m
%% 同定実験２ p82- KとTの同定
%% initialaize
clear all;
clc;

%%パラメータ値の設定
ts=1/50;
u_ini=1.75;
r_const=1.1;
p_const=1.0;
s_time=10;
w_time=4;

%%Open simulink model
open_system('sim_velo_id_tc_sl');

%% 実験
z=sim('sim_velo_id_tc_sl');
y=z.yout.signals.values;
t=z.yout.time;
c1=mean(y(w_time/ts:s_time/ts));
c2=mean(y((s_time+w_time)/ts:end));

figure(1)
plot(t,y,...
    t,ones(size(t))*c1,'r--',...
    t,ones(size(t))*c2,'r--')
xlabel('Time [s]')
ylabel('Velocity [V]')

%%geinnnokeisann
K_id =(c2-c1)/p_const;
u_offset=(K_id*r_const -c1)/K_id;
y2=y(s_time/ts:end)-c1;
```

```
t2=t(s_time/ts:end);
t2=t2-t2(1);

tc_idx=min(find(y2>(c2-c1)*0.632));
T_id=t2(tc_idx);

figure(2)
plot(t2,y2,'.',...
    T_id,(c2-c1)*0.632,'ro',...
    t2,ones(size(t2))*(c2-c1),'r--')
xlabel('Time [s]'), ylabel('Velocity [V]')

%%Display Result
fprintf('== Results==\n')
fprintf('K       =%f\n',K_id)
fprintf('T       =%f\n',T_id)
fprintf('u_offset = %f\n',u_offset)
```

付録1　同定実験で用いたプログラム

```
%%velo_id_tc.m
%% 同定実験２ p86- KとTの同定
%% initialaize
clear all;
clc;

%%パラメータ値の設定
ts=1/50;
p_const=0.75;
s_time=10;
K       =0.837189;
T       =2.800000;
u_offset = -0.457393*1.5;
r_const=1.1;
%%Open simulink model
```

```
open_system('sim_velo_id_verify');
sim('sim_velo_id_verify')
```

付録２　検証実験で用いたプログラム