

明石工業高等専門学校専攻科

## メカトロニクス レポート課題

報告者

ME2208 高橋 尚太郎  
(機械・電子システム工学専攻2年)

提出年月日：2023年7月26日

# 1 演習 1

## 1.1 実行プログラム

実行プログラムをソースコード 1 に示す。

ソースコード 1: 演習 1 のプログラム

```
1 #include<stdio.h>
2 int main()
3 {
4     printf("Hello/n");
5     return 0;
6 }
```

## 1.2 実行結果

JTW のコンソール上に Hello と表示される。

# 2 演習 2

## 2.1 実行プログラム

実行プログラムをソースコード 2 に示す。

ソースコード 2: 演習 2 のプログラム

```
1 #include <stdio.h>
2 #include <process.h>
3
4 int main()
5 {
6     int i;
7     FILE *fp;
8
9     if((fp = fopen("test.dat", "wt")) == NULL){
10         printf("Cannot Open test.dat!!\n");
11         exit(1);
12     }
13
14     for(i = 0; i < 100; i++){
15         fprintf(fp, "%3d\n", i);
16     }
17     fclose(fp);
18     return 0;
19 }
```

## 2.2 実行結果

ロボット上で得られたデータファイルをソースコード 3 に示す。0~100 までの数値が出力される。

ソースコード 3: ロボット上で得られたデータファイル (TEST.DAT)

```
1  0
2  1
3  2
4  3
5  4
6  5
7  6
8  7
9  8
10 9
11 10
12 ~~~~~
13 90
14 91
15 92
16 93
17 94
18 95
19 96
20 97
21 98
22 99
23 .....
```

---

### 3 演習 3

#### 3.1 実行プログラム

実行プログラムをソースコード 4 に示す。

ソースコード 4: 演習 3 のプログラム

---

```
1 #include <stdio.h>
2 #include <dos.h>
3 #include "v25.h"
4 #include "ms.h"
5
6 int main(){
7     pokeb( _V25BASE, _PMC2, 0x00 ); /*P20-P27 ポートモード*/
8     pokeb( _V25BASE, _PM2, 0x00 ); /*P20-P27 出力ポート*/
9
10    while(1){
11        ms_led2( 0xff ); /* 全の点灯 LED */
12        ms_wait( 1000 ); /* 待ち時間1s */
13        ms_led2( 0x00 ); /* 全の消灯 LED */
14        ms_wait( 1000 ); /* 待ち時間1s */
15    }
16    return 0;
17 }
```

---

### 3.2 実行結果

ポート 2 の LED が点滅する。

## 4 演習 4

### 4.1 実行プログラム

実行プログラムをソースコード 5 に示す。

ソースコード 5: 演習 4 のプログラム

```
1 int main(){
2     pokeb( _V25BASE, _PMC2, 0x00 ); /*P20-P27 ポートモード*/
3     pokeb( _V25BASE, _PM2, 0x00 ); /*P20-P27 出力モード*/
4
5     while(1){
6         int i;
7         int a = 0x01;
8         for(i=0;i<8;i++)
9         {
10             ms_led2(a);
11             ms_wait(1000);
12             a = a << 1;
13         }
14     }
15     return 0;
16 }
```

### 4.2 実行結果

LED0 から LED7 の順に点灯した後、この一連の動作を繰り返す。

## 5 演習 5

### 5.1 実行プログラム

実行プログラムをソースコード 6 に示す。

ソースコード 6: 演習 5 のプログラム

```
1 #include <stdio.h>
2 #include <dos.h>
3 #include "v25.h"
4 #include "ms.h"
5
6 int main(){
7     int i;
8
9     pokeb( _V25BASE, _PMC1, 0x20 ); /* P15:TOUT 出力*/
10    for( i=0; i<3; i++) {
```

---

```

11     ms_beep( 440, 500 ); /* 440Hz, 0.5s */
12     ms_wait( 500 ); /* 待ち時間0.5s*/
13 }
14 ms_beep( 880, 1000 );
15 return 0;
16 }
```

---

## 5.2 実行結果

440Hz の音が 0.5 秒毎に 3 回鳴った後、880Hz の音が 1 秒間鳴る。

# 6 演習 6

## 6.1 実行プログラム

実行プログラムをソースコード 7 に示す。

ソースコード 7: 演習 6 のプログラム

---

```

1 #include <stdio.h>
2 #include <dos.h>
3 #include "v25.h"
4 #include "ms.h"
5
6 int main(){
7     int i;
8     int f;
9     int ms;
10
11     pokeb( _V25BASE, _PMC1, 0x20 );
12     while(1){
13         scanf("%d %d", &f, &ms);
14         ms_beep( f, ms ); /* 440Hz, 0.5s */
15     }
16     return 0;
17 }
```

---

## 6.2 実行結果

JTW のプロンプト上でキーボードから周波数 [Hz](スペース) 時間 [s] を入力して、エンターキーを入力すると、指定した数値に応じた音がブザーから鳴る。

# 7 演習 7

## 7.1 実行プログラム

実行プログラムをソースコード 8 に示す。

ソースコード 8: 演習 7 のプログラム

---

```

1 #include <stdio.h>
2 #include <dos.h>
```

```
3 #include "v25.h"
4 #include "ms.h"
5
6 int main()
7 {
8     int s[3];
9
10    pokeb(_V25BASE, _PMC0, 0x00);
11    pokeb(_V25BASE, _PM0, 0xff);
12
13    pokeb(_V25BASE, _PMC1, 0x20);
14
15    pokeb(_V25BASE, _PMC2, 0x00);
16    pokeb(_V25BASE, _PM2, 0xff);
17
18    printf("%c[2J", 0x1b);
19    while (1){
20        ms_ifr(s);
21        printf("%c[01;01H", 0x1b);
22        printf("L:%d C:%d R:%d\n", s[0], s[1], s[2]);
23    }
24    return 0;
25 }
```

---

## 7.2 実行結果

左右と真ん中に取り付けられた 3 つの近接センサの値 (0 か 1) が output される。

# 8 演習 8

## 8.1 実行プログラム

実行プログラムは演習 7 と同様である。

## 8.2 実行結果

赤外線センサの 2 次元的な範囲を図 1 に示す。扇型の外側が 0 の値を示す。(検知範囲外である。) 斜線部と白抜きを含めた扇型の内側の範囲がセンサの検知範囲であり、白抜きが 1 の検知範囲、斜線部が 0 と 1 の間でばらつきがある範囲である。

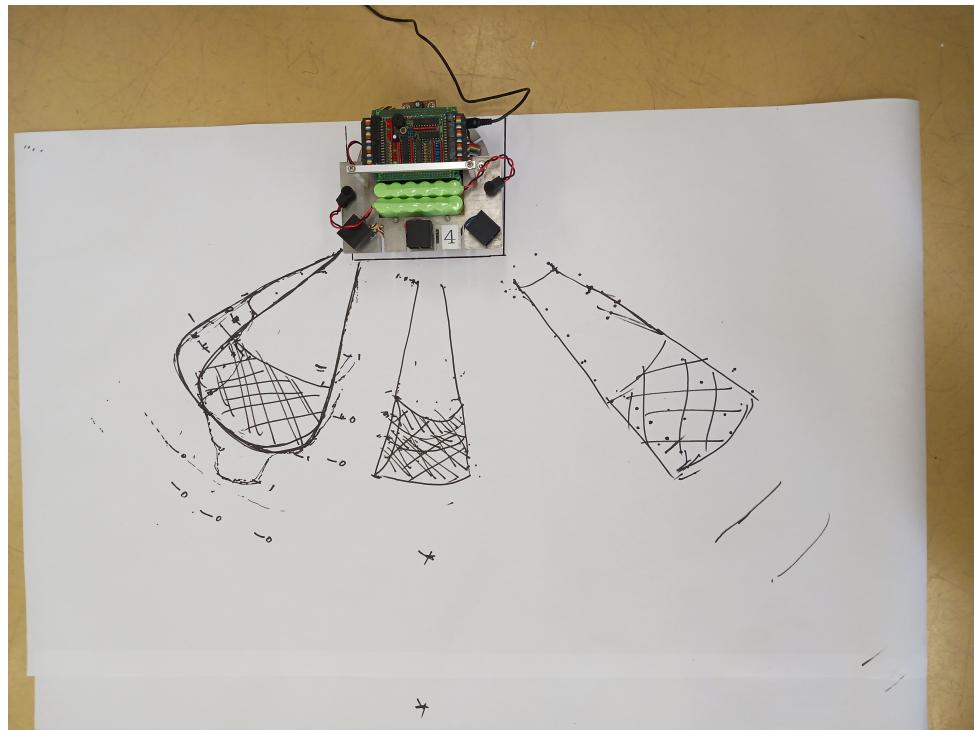


図 1: 赤外線センサの 2 次元的な範囲

## 9 演習 9

### 9.1 実行プログラム

実行プログラムをソースコード 9 に示す。

ソースコード 9: 演習 9 のプログラム

```
1 #include <stdio.h>
2 #include <dos.h>
3 #include "v25.h"
4 #include "ms.h"
5
6 int main(){
7     long cl, cr, ct;
8
9     ms_enc_init();
10    printf("%c[2J", 0x1b);
11    while(1){
12        ms_read_c(&cl, &cr, &ct);
13
14        printf("%c[01;01H", 0x1b);
15        printf("L:%10ld\n", cl);
16        printf("R:%10ld\n", cr);
17        printf("T:%10ld\n", ct);
18    }
}
```

```
19     return 0;  
20 }
```

## 9.2 実行結果

タイヤを正回転させることで、左右のモータのロータリーエンコーダの値が加算されて出力される。タイヤを逆回転させると、減算されて出力される。

# 10 演習 10

## 10.1 実行プログラム

実行プログラムをソースコード 10 に示す。

ソースコード 10: 演習 10 のプログラム

```
1 #include <stdio.h>  
2 #include <dos.h>  
3 #include "v25.h"  
4 #include "ms.h"  
5  
6 int main(){  
7     int i;  
8     long cl, cr, ct;  
9     long f[3] = {0, 0, 1};  
10    long f_old[3] = {0, 0, 0};  
11    float a_l, a_r, v_l, v_r;  
12  
13    ms_enc_init();  
14    printf("%c[2J", 0x1b);  
15    while(1){  
16        ms_read_c(&cl, &cr, &ct);  
17  
18        f[0] = cl;  
19        f[1] = cr;  
20        f[2] = ct*3/1000;  
21  
22  
23        a_l = (1000/3)*(f[0] - f_old[0])/(400*19.225);  
24        a_r = (1000/3)*(f[1] - f_old[1])/(400*19.225);  
25        v_l = (60*a_l);  
26        v_r = (60*a_r);  
27  
28        printf("%c[01;01H", 0x1b);  
29        printf("a_l:%10lf\n", a_l);  
30        printf("a_r:%10lf\n", a_r);  
31        printf("v_l:%10lf\n", v_l);  
32        printf("v_r:%10lf\n", v_r);  
33
```

---

```

34     printf("T:%10ld\n", ct*3/1000);
35
36     for (i = 0; i < 2; i++){
37         f_old[i] = f[i];
38     }
39
40 }
41 return 0;
42 }
```

---

## 10.2 実行結果

左右のロータリーエンコーダの値をモータの回転数に変換され、モータの回転数の瞬時値が出力される。

# 11 演習 11

## 11.1 実行プログラム

実行プログラムをソースコード 11 に示す。

ソースコード 11: 演習 11 のプログラム

---

```

1 #include <stdio.h>
2 #include <dos.h>
3 #include "v25.h"
4 #include "ms.h"
5
6 int main(){
7     int i;
8     int vl, vr;
9     long ct;
10
11     ms_motor_init();
12     ms_motor_on();
13
14     printf("%c[2J", 0x1b);
15
16     for(i=0; i>=-127; i--){
17         ms_set_pwm(i, i);
18         ms_wait(50);
19     }
20
21     for(i=-127; i<=127; i++){
22         ms_set_pwm(i, i);
23         ms_wait(50);
24         ms_read_v(&vl, &vr, &ct);
25         printf("%c[01;01H", 0x1b);
26         printf("pwm:%5d\n", i);
27         printf("vl:%7d\n", vl);
```

```
28     printf("vr:%7d\n", vr);
29 }
30
31 for(i=127; i>=0; i--){
32     ms_set_pwm(i, i);
33     ms_wait(50);
34 }
35
36 ms_motor_off();
37 return 0;
38 }
```

---

## 11.2 実行結果

左右のモータが回転し、その時の回転速度が JTW 上に表示される。

# 12 演習 12

## 12.1 実行プログラム

実行プログラムをソースコード 12 に示す。

ソースコード 12: 演習 12 のプログラム

---

```
1 #include <stdio.h>
2 #include <dos.h>
3 #include "v25.h"
4 #include "ms.h"
5 #include<process.h>
6
7 int main(){
8     int i;
9     int vl, vr;
10    long ct;
11    FILE *fp;
12
13    ms_motor_init();
14    ms_motor_on();
15
16    printf("%c[2J", 0x1b);
17
18    for(i=0; i>=-127; i--){
19        ms_set_pwm(i, i);
20        ms_wait(50);
21    }ファイルのオープン
22
23    /**
24    if ((fp = fopen("12.csv", "wt"))==NULL)
25    {
```

```
26     printf("Can't open file.\n");
27     exit(1);
28 }
29
30 for(i=-127; i<=127; i++){
31     ms_set_pwm(i, i);
32     ms_wait(50);
33     ms_read_v(&vl, &vr, &ct);
34     printf("%c[01;01H", 0x1b);
35     printf("pwm:%5d\n", i);
36     printf("vl:%7d\n", vl);
37     printf("vr:%7d\n", vr);
38
39     fprintf(fp,"%5d,%7d,%7d\n",i,vl,vr); /*値と左右回転速度の書き込み pwm*/
40 }
41
42 for(i=127; i>=0; i--){
43     ms_set_pwm(i, i);
44     ms_wait(50);
45 }
46
47 ms_motor_off(); ファイルのクローズ
48 /**
49 fclose(fp);
50 return 0;
51 }
```

---

## 12.2 実行結果

左右のモータの PWM 値と回転速度のデータファイル ”12.csv” が作成される。12.csv に置けるモータの回転数の特性を図 2 に示す。

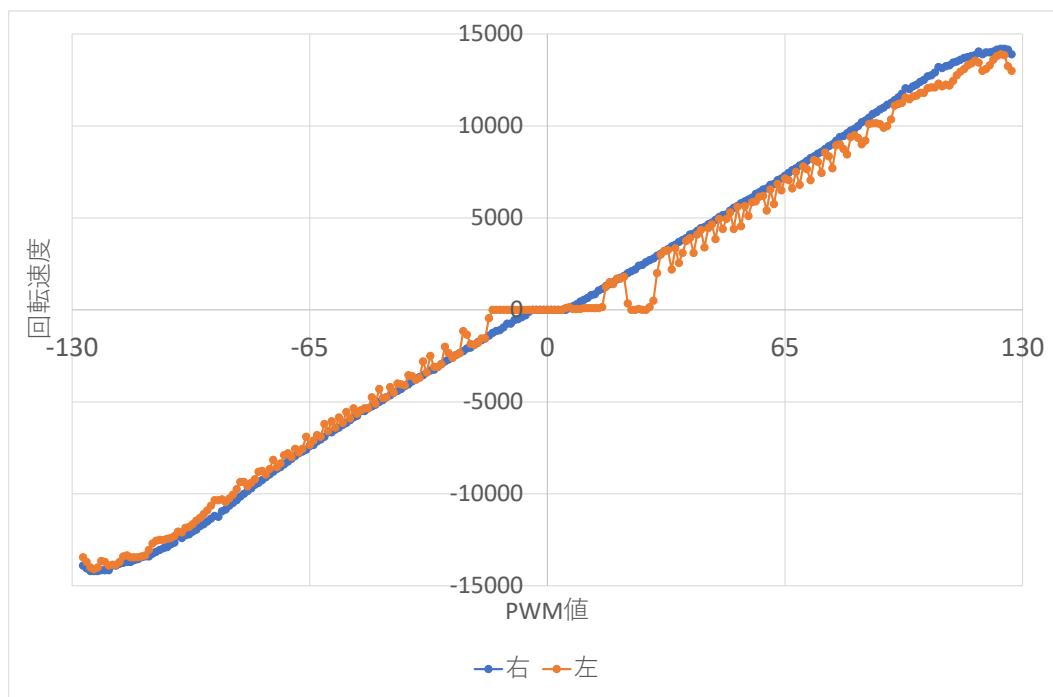


図 2: 左右のモータの回転数の特性

## 13 演習 13

### 13.1 実行プログラム

実行プログラムをソースコード 13 に示す。

ソースコード 13: 演習 13 のプログラム

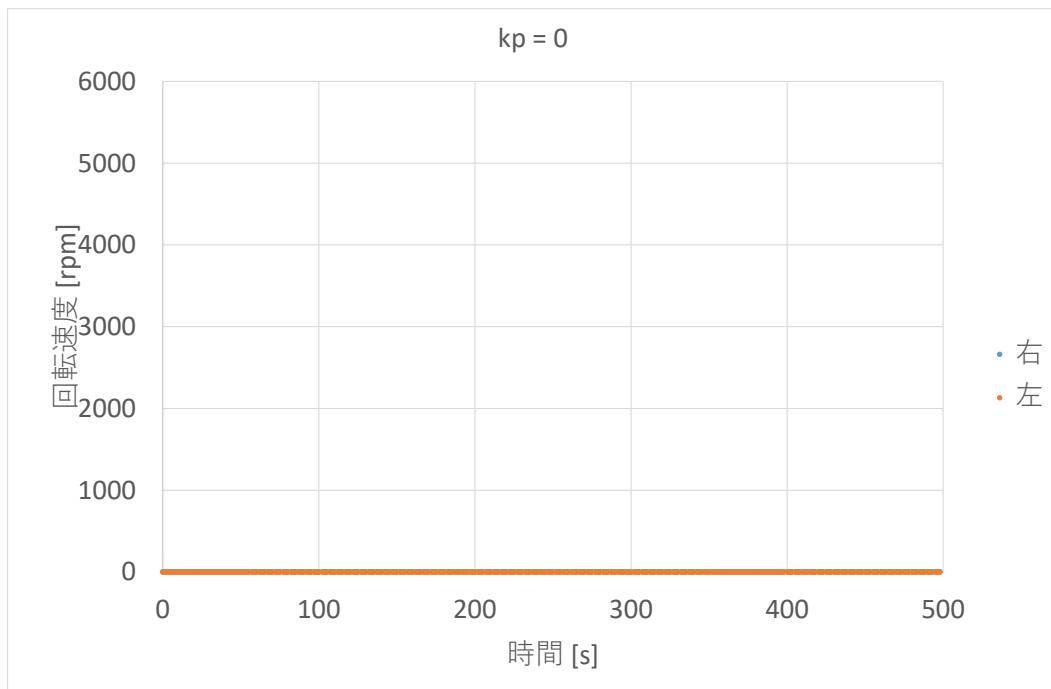
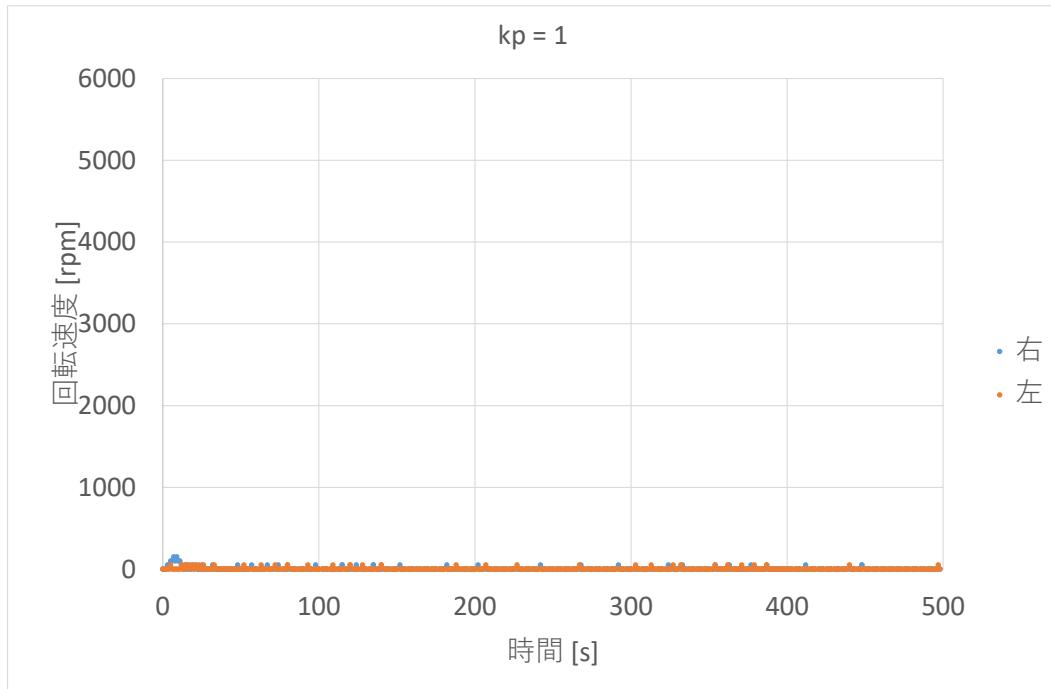
```
1 #include <stdio.h>
2 #include <process.h> /*関数の定義 exit*/
3 #include <dos.h>
4 #include "v25.h"
5 #include "ms.h"
6
7 #define DMAX 500 保存するデータ数/**/
8
9 int main(){
10     int i;
11     int kp, ki_inv;
12     int vrefl, vrefr;
13     int vl[DMAX], vr[DMAX];
```

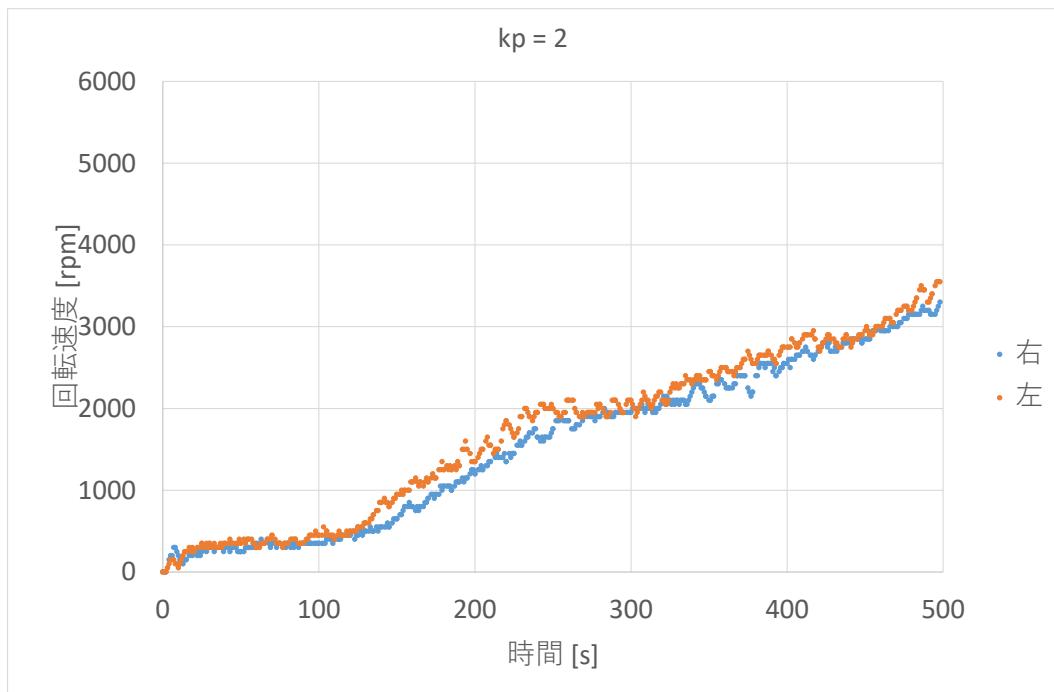
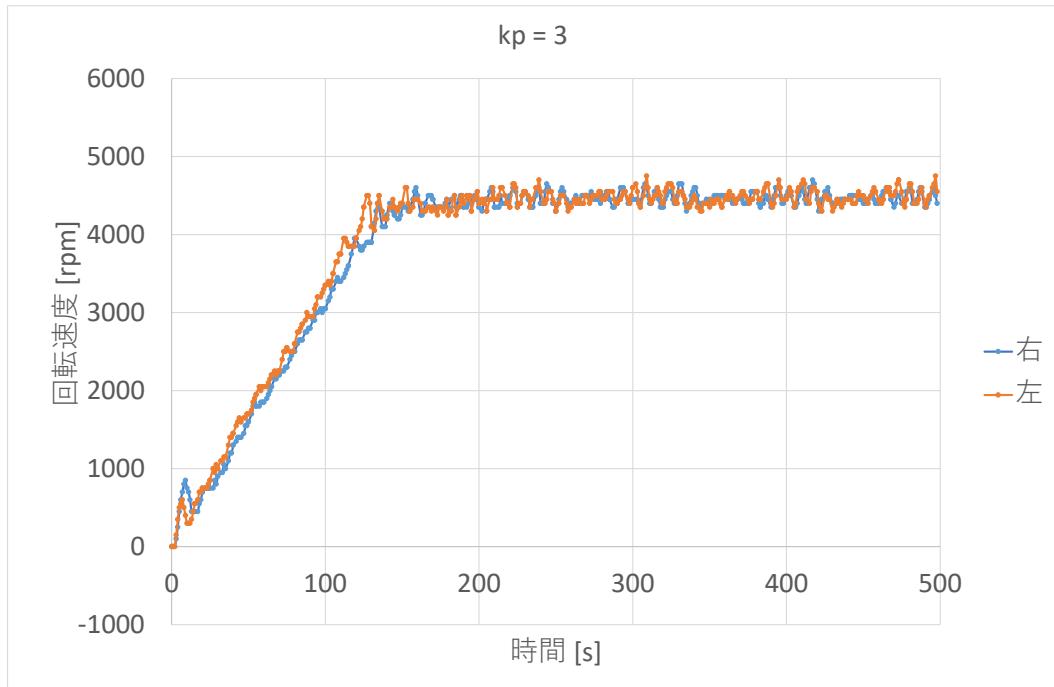
```
14 long ct, ctm, t[DMAX];
15 FILE *fp;
16
17 ms_init();
18
19 kp = ms_read_dip();
20 ki_inv = 10000;
21 vrefl = 5000;
22 vrefr = 5000;
23
24 ms_motor_on();
25 ms_read_v(&vl[0], &vr[0], &ctm);
26 ms_step_res(vrefl, vrefr, kp, ki_inv);
27 for(i=1; i<DMAX; i++){
28     ms_read_v(&vl[i], &vr[i], &ct);
29     t[i] = ct - ctm;
30     ms_wait(2);
31 }
32 ms_motor_off();
33
34 if((fp = fopen("data.dat", "wt")) == NULL){
35     printf("Can't Open File!!\n");
36     exit(1);
37 }
38
39 for(i=1; i<DMAX; i++){
40     fprintf(fp, "%7d %7d %10ld\n", vl[i], vr[i], t[i]);
41 }
42 fclose(fp);
43 return 0;
44 }
```

---

## 13.2 実行結果

モータのステップ応答(回転速度と時間)をデータファイルに保存する。比例フィードバック係数  $k_p$  をスイッチ値(0~15)を変化させた時の応答グラフを図3(a)~10(b)に示す。なお、 $k_p$  の値は7で安定とした。 $k_p = 0 \sim 2$ の場合、ロボットは前進しない。

(a)  $k_p = 0$ (b)  $k_p = 1$

(a)  $k_p = 2$ (b)  $k_p = 3$ 図 4: ステップ応答のグラフ  
14

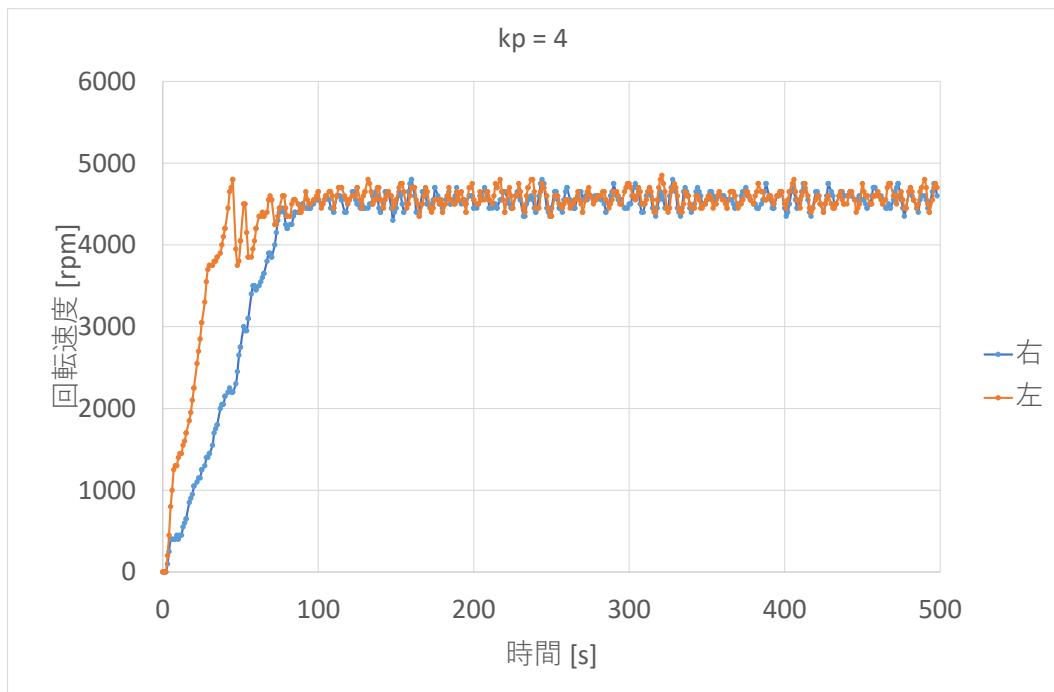
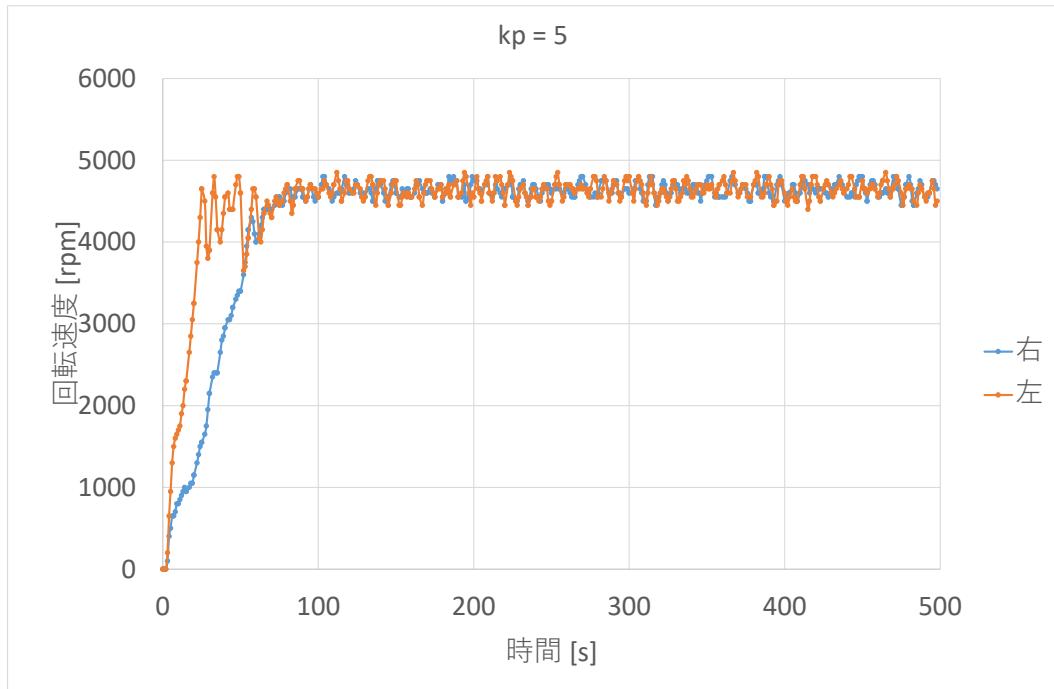
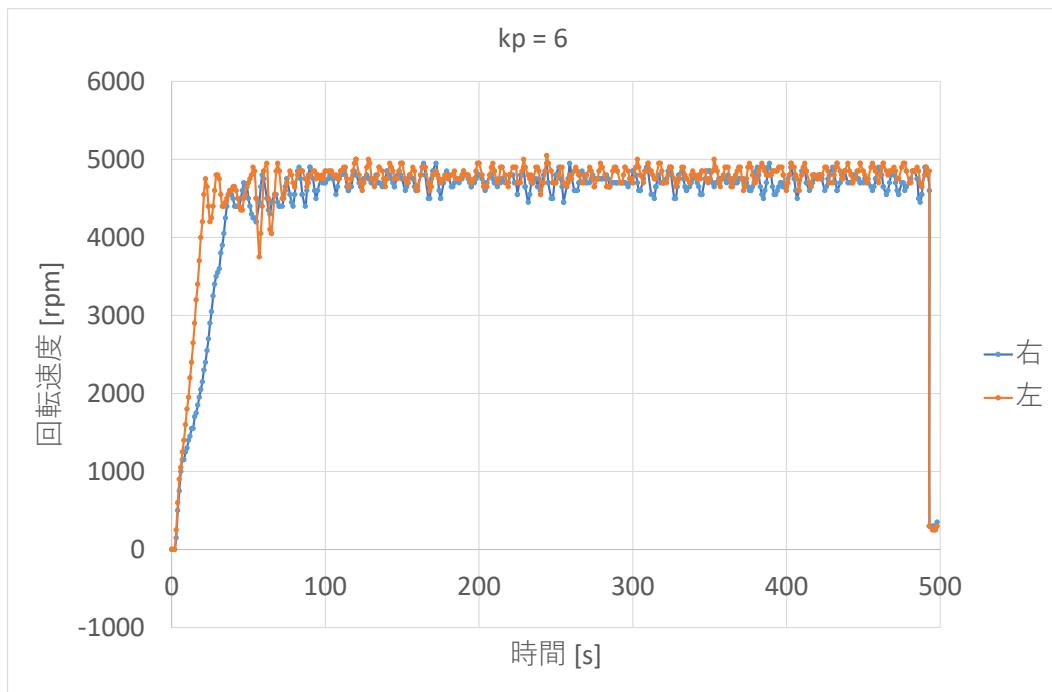
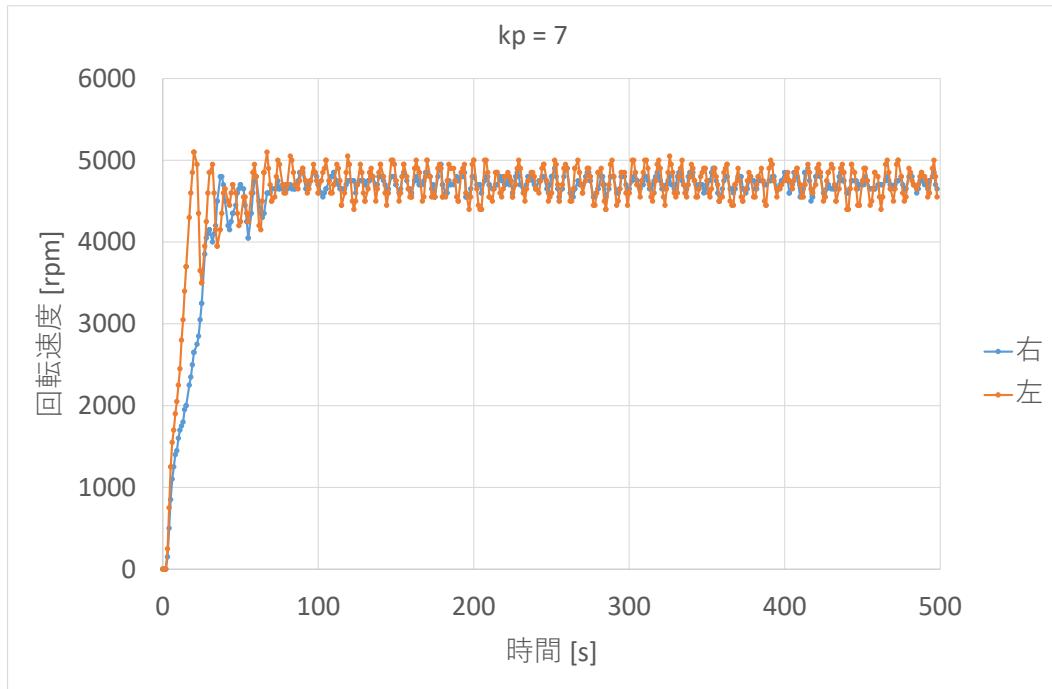
(a)  $k_p = 4$ (b)  $k_p = 5$ 

図 5: ステップ応答のグラフ

(a)  $k_p = 6$ (b)  $k_p = 7$ 図 6: ステップ応答のグラフ  
16

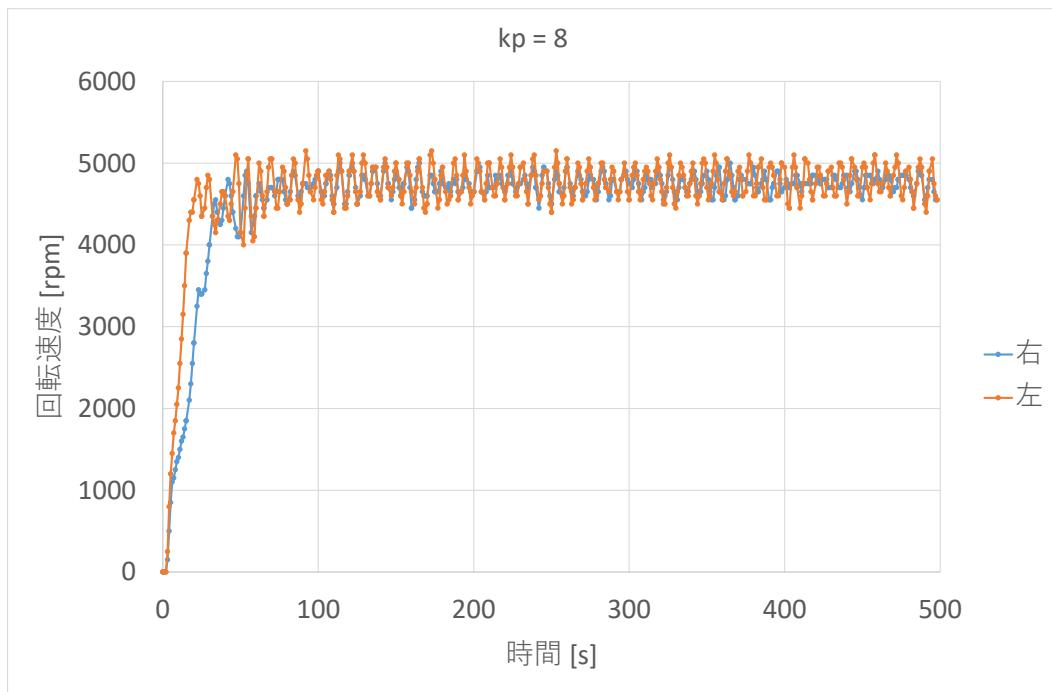
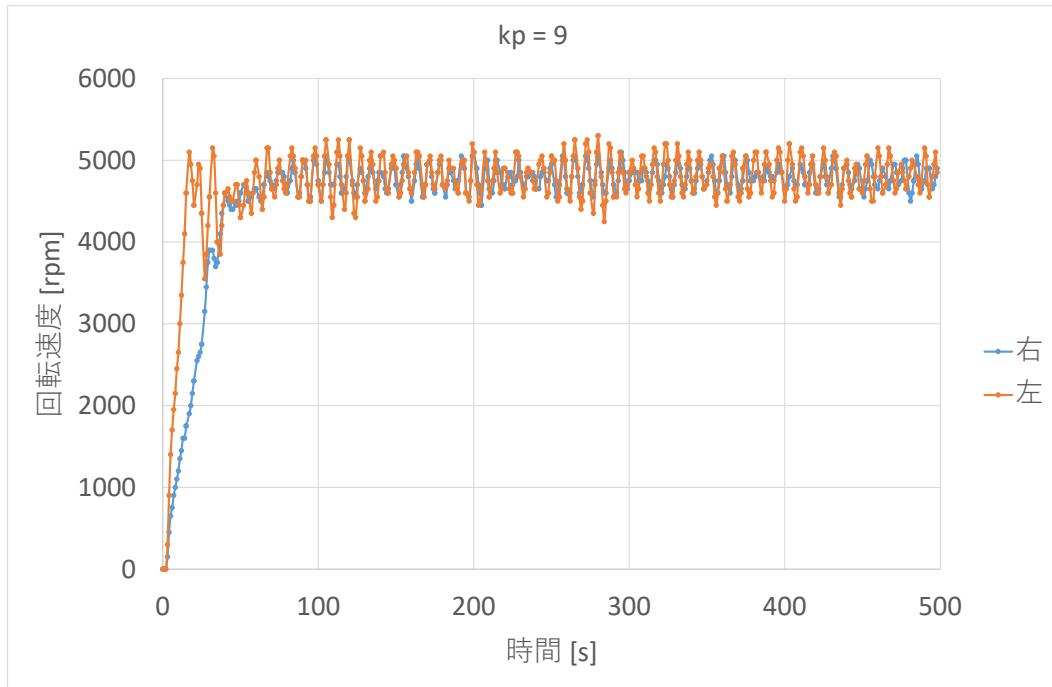
(a)  $k_p = 8$ (b)  $k_p = 9$ 

図 7: ステップ応答のグラフ

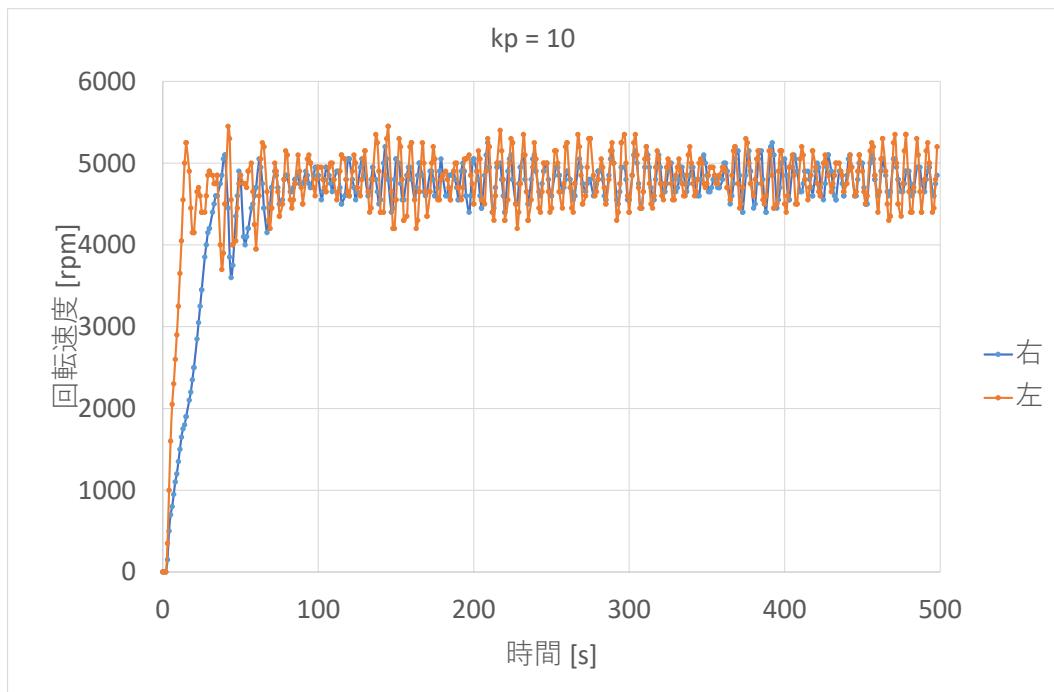
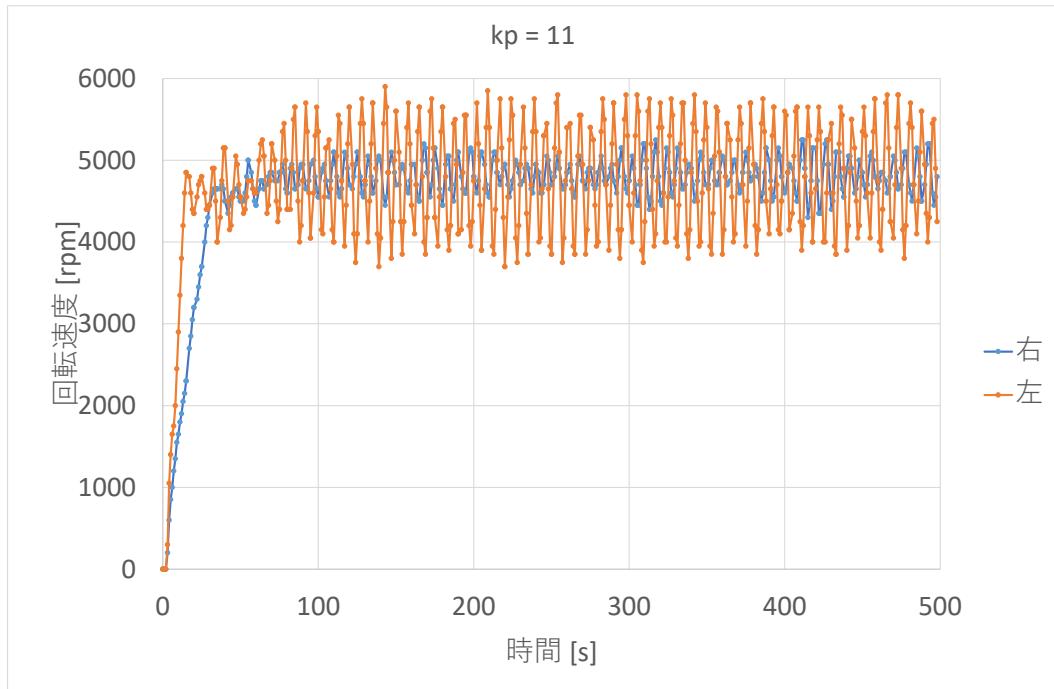
(a)  $k_p = 10$ (b)  $k_p = 11$ 

図 8: ステップ応答のグラフ

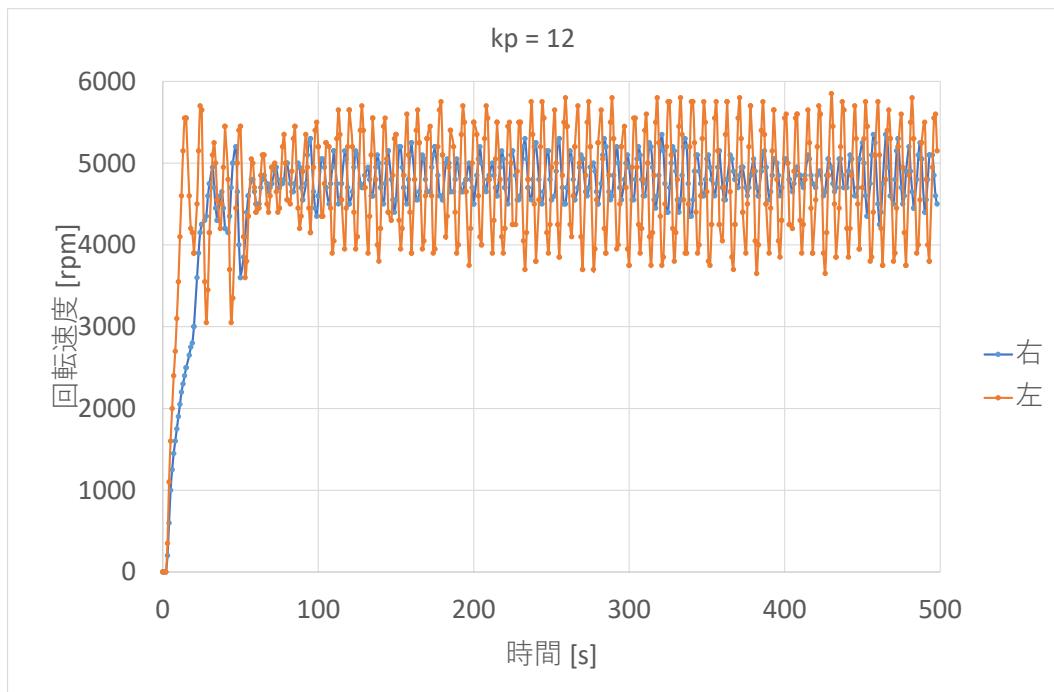
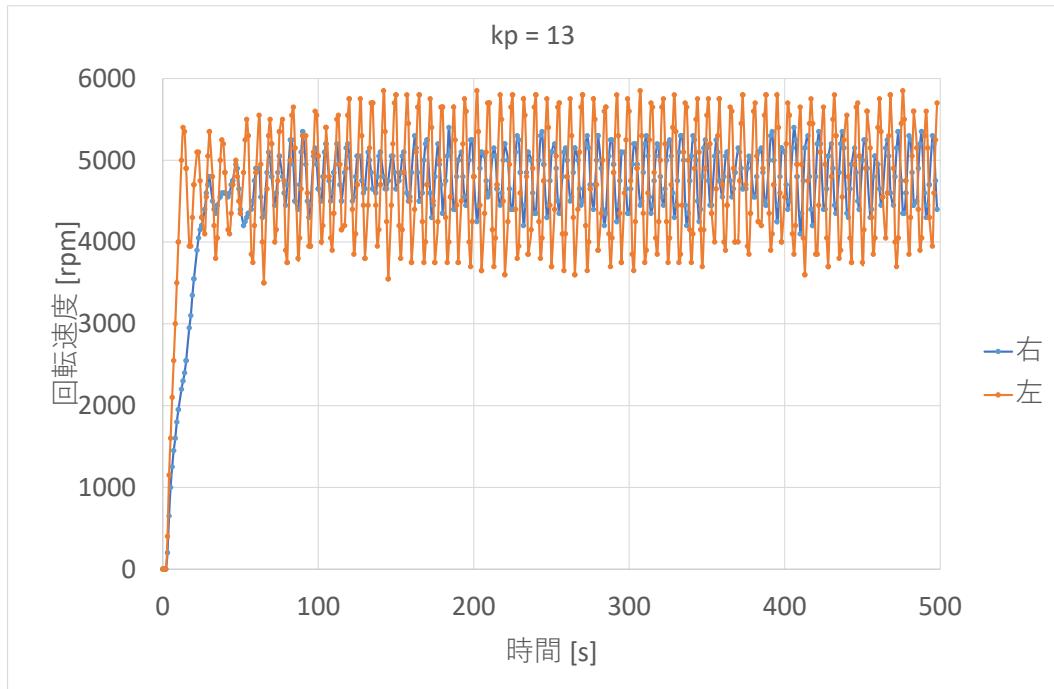
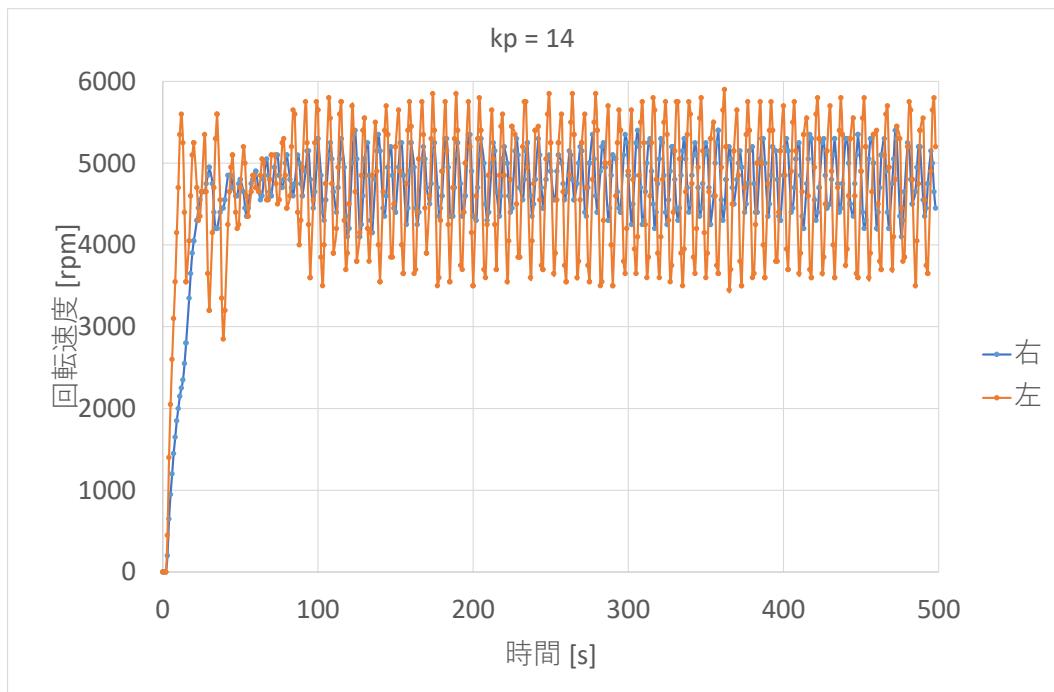
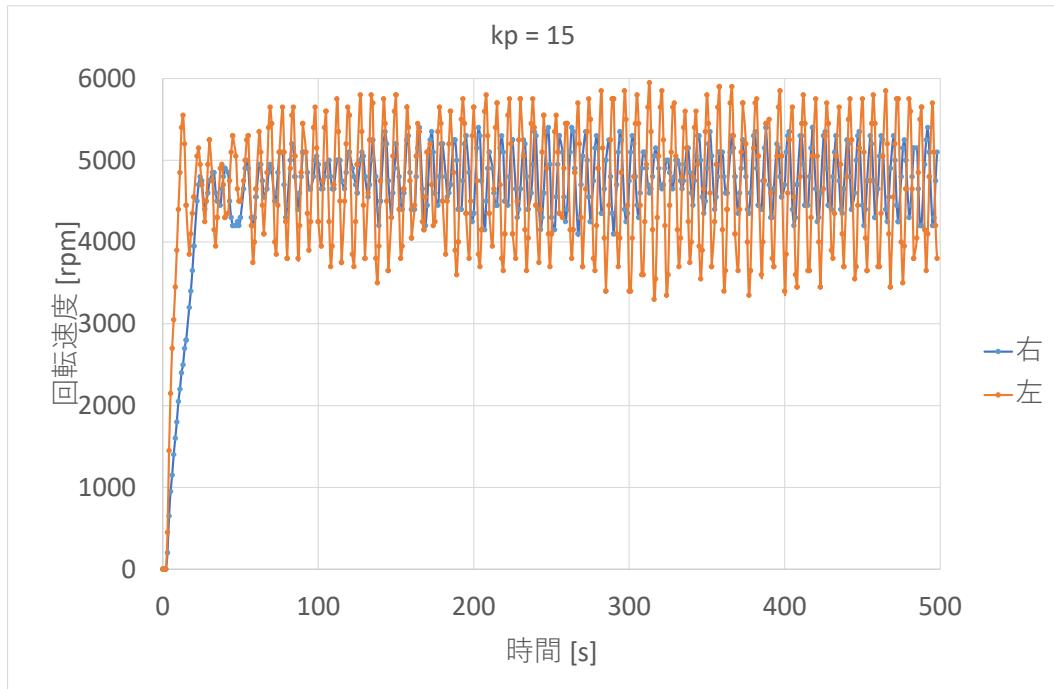
(a)  $k_p = 12$ (b)  $k_p = 13$ 

図 9: ステップ応答のグラフ

(a)  $k_p = 14$ (b)  $k_p = 15$ 図 10: ステップ応答のグラフ  
20

## 14 演習 14

### 14.1 実行プログラム

実行プログラムをソースコード 14 に示す。積分フィードバック係数 `ki_inv=ms_read_dip()*1000` とした。

ソースコード 14: 演習 14 のプログラム

```
1 #include <stdio.h>
2 #include <process.h>
3 #include <dos.h>
4 #include "v25.h"
5 #include "ms.h"
6
7 #define DMAX 500
8
9 int main(){
10     int i;
11     int kp, ki_inv;
12     int vrefl, vrefr;
13     int vl[DMAX], vr[DMAX];
14     long ct, ctm, t[DMAX];
15     FILE *fp;
16
17     ms_init();
18
19     kp = 7;
20     ki_inv = ms_read_dip()*1000;
21     vrefl = 5000;
22     vrefr = 5000;
23
24     ms_motor_on();
25     ms_read_v(&vl[0], &vr[0], &ctm);
26     ms_step_res(vrefl, vrefr, kp, ki_inv);
27     for(i=1; i<DMAX; i++){
28         ms_read_v(&vl[i], &vr[i], &ct);
29         t[i] = ct - ctm;
30         ms_wait(2);
31     }
32     ms_motor_off();
33
34     if((fp = fopen("data.dat", "wt")) == NULL){
35         printf("Can't Open File!!\n");
36         exit(1);
37     }
38
39     for(i=1; i<DMAX; i++){
40         fprintf(fp, "%7d %7d %10ld\n", vl[i], vr[i], t[i]);
41     }
42     fclose(fp);
43     ms_beep(440,500);
```

```

44     printf("owari\n");
45     return 0;
46 }

```

## 14.2 実行結果

演習 13 と同様にモータのステップ応答(回転速度と時間)をデータファイルに保存する。応答グラフを図 11(a)~14(i) に示す。積分フィードバック係数  $k_i = 1/13000$  を最適な値とした。つまり、`ms_read_dip()=13` の値である。

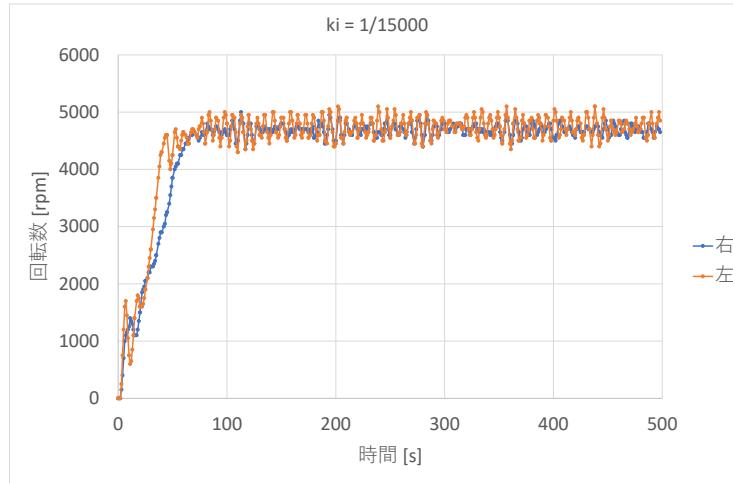
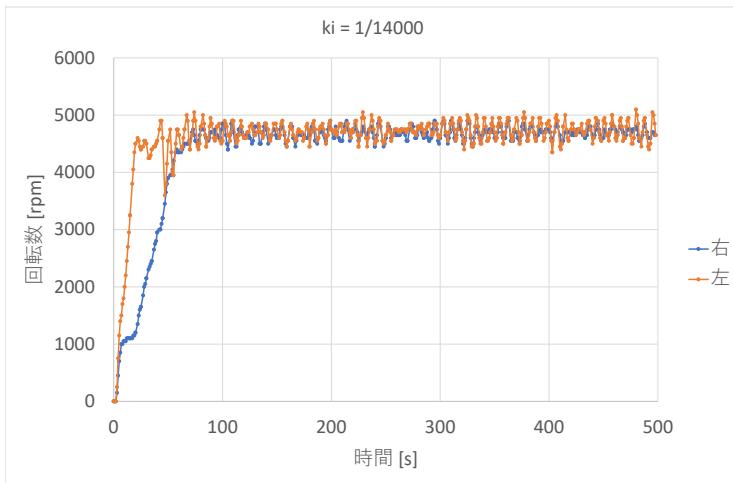
(a)  $k_i = 1/15000$ (b)  $k_i = 1/14000$ 

図 11: ステップ応答のグラフ

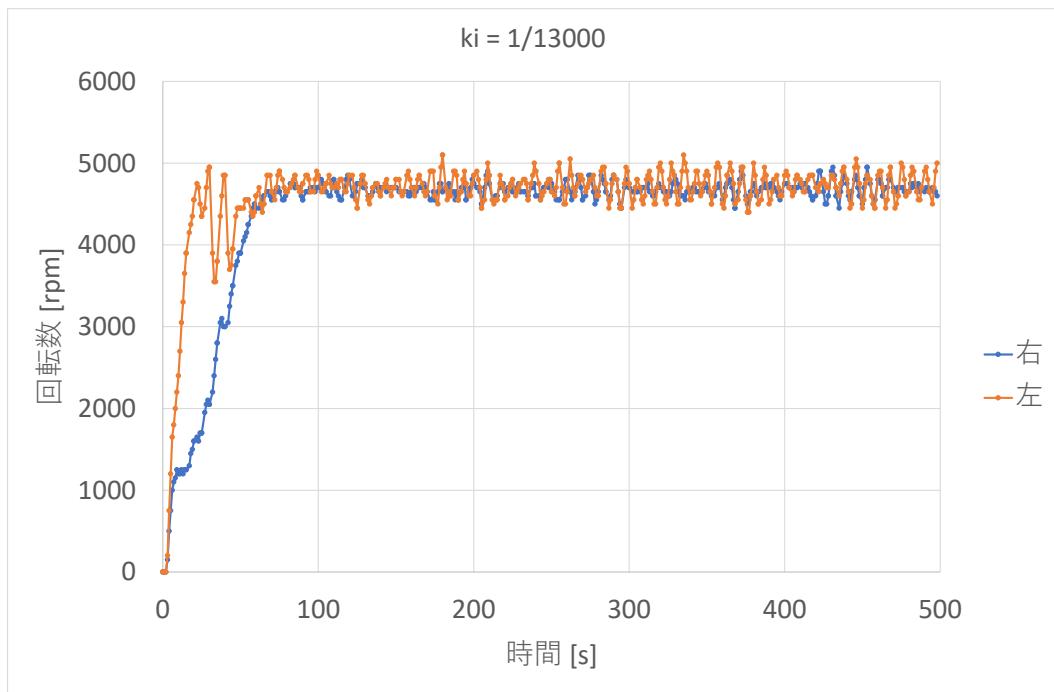
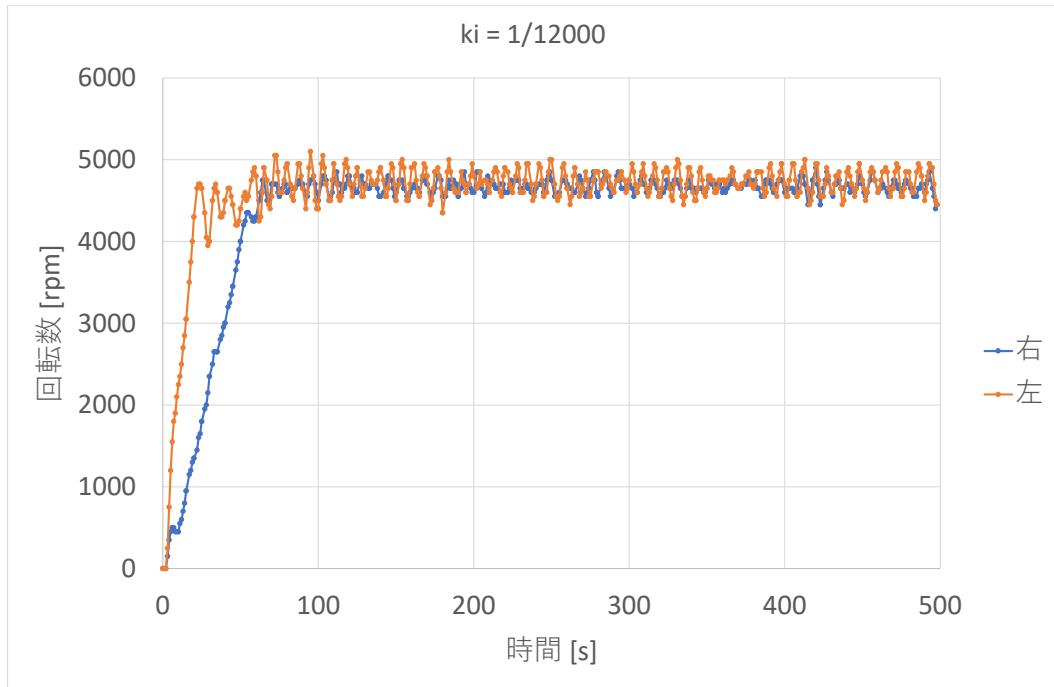
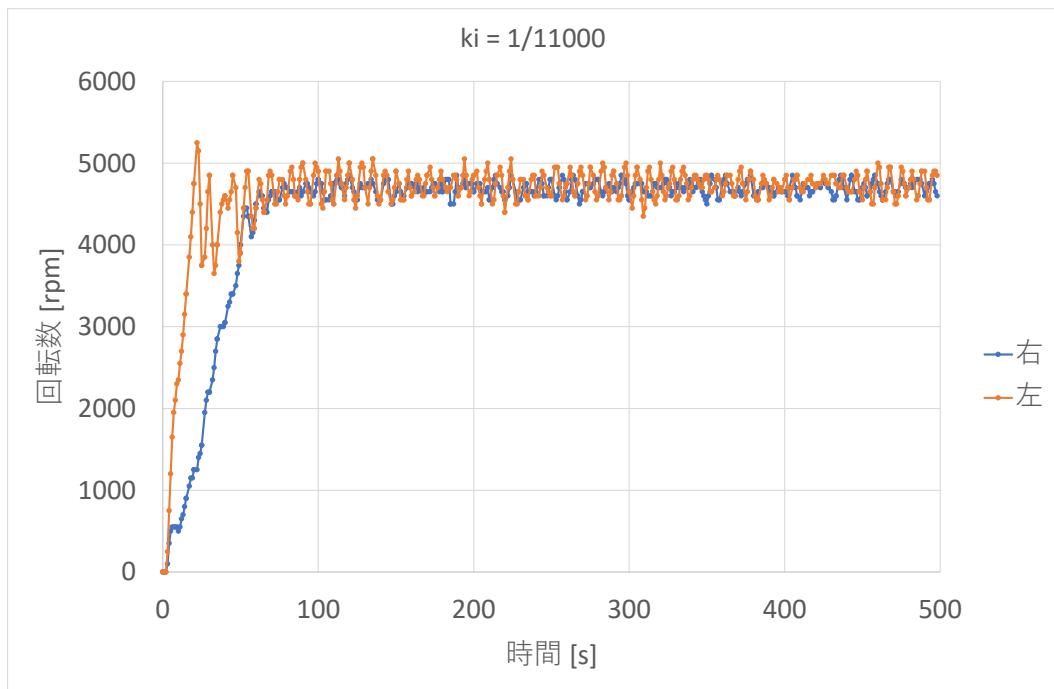
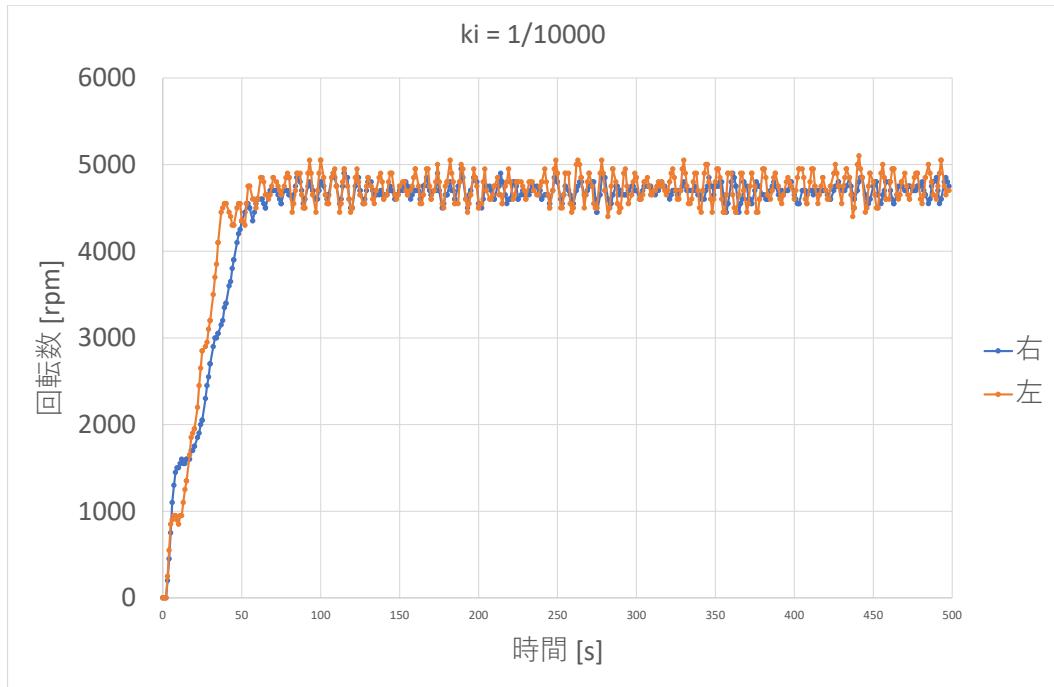
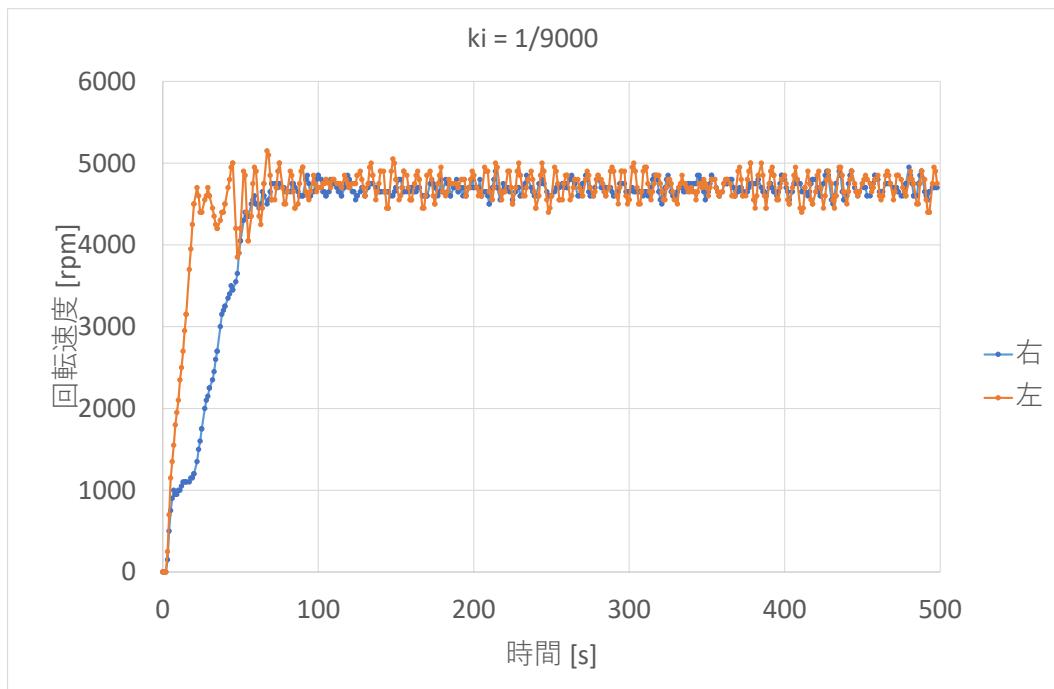
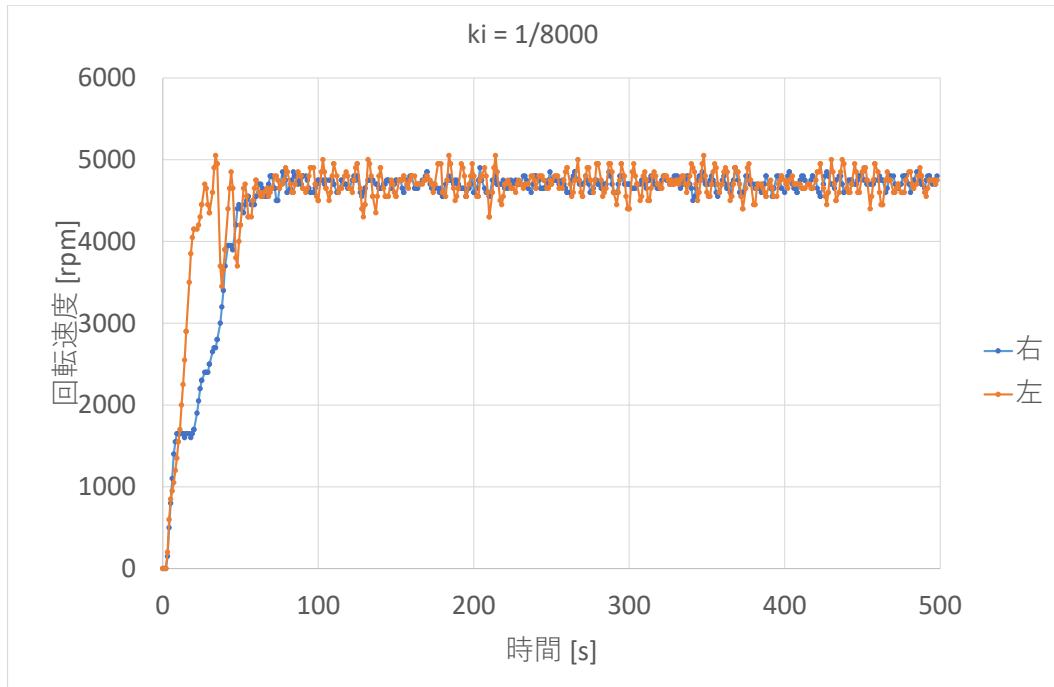
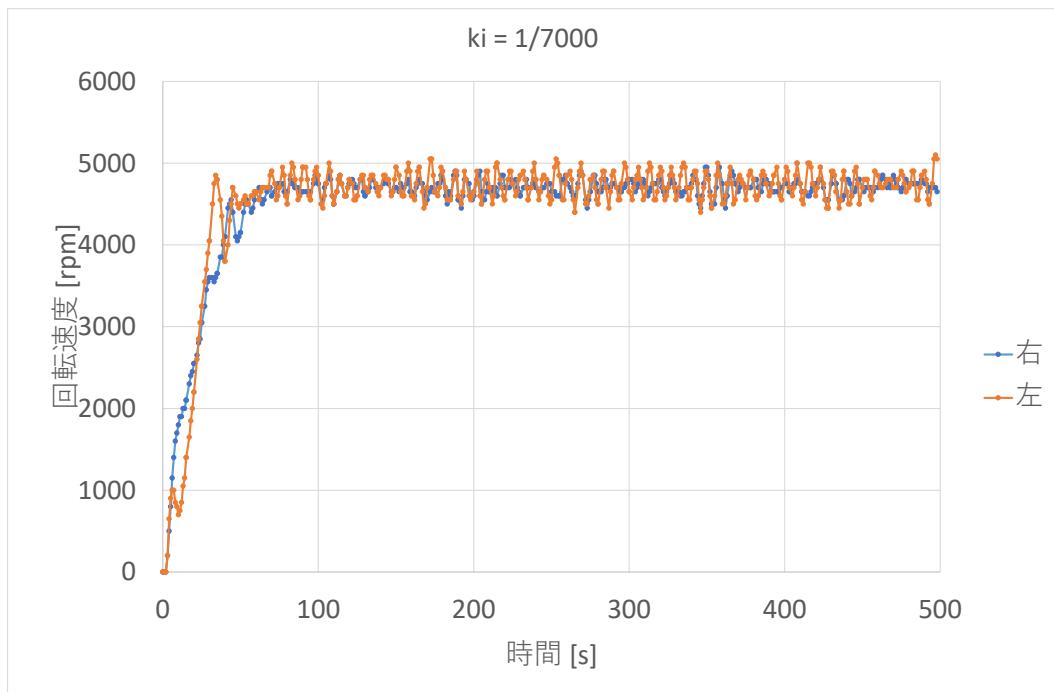
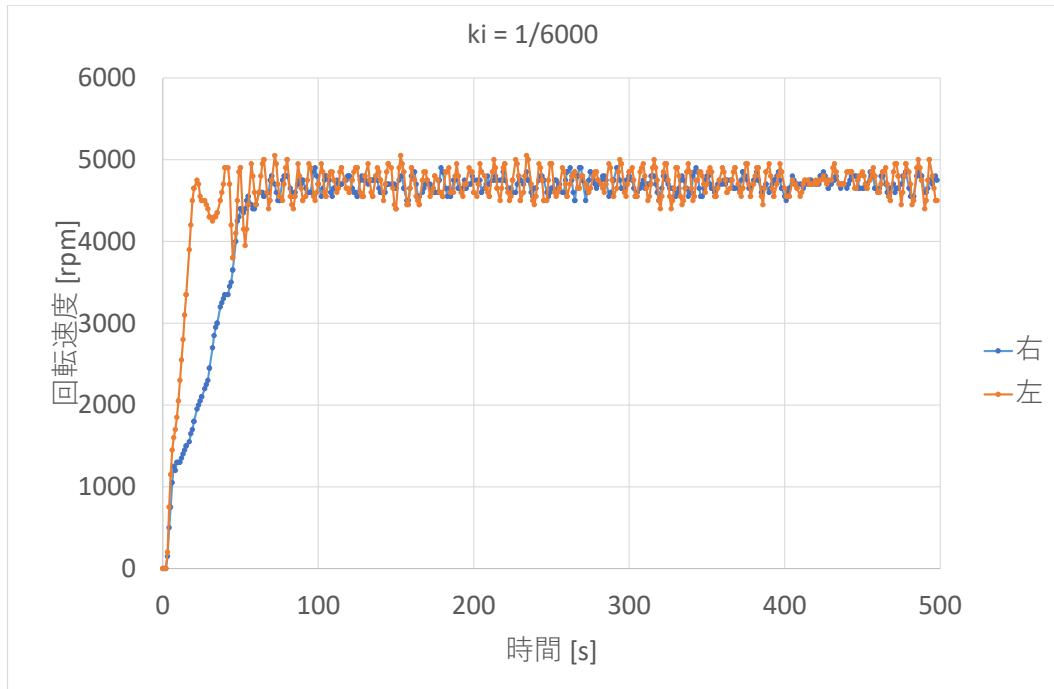
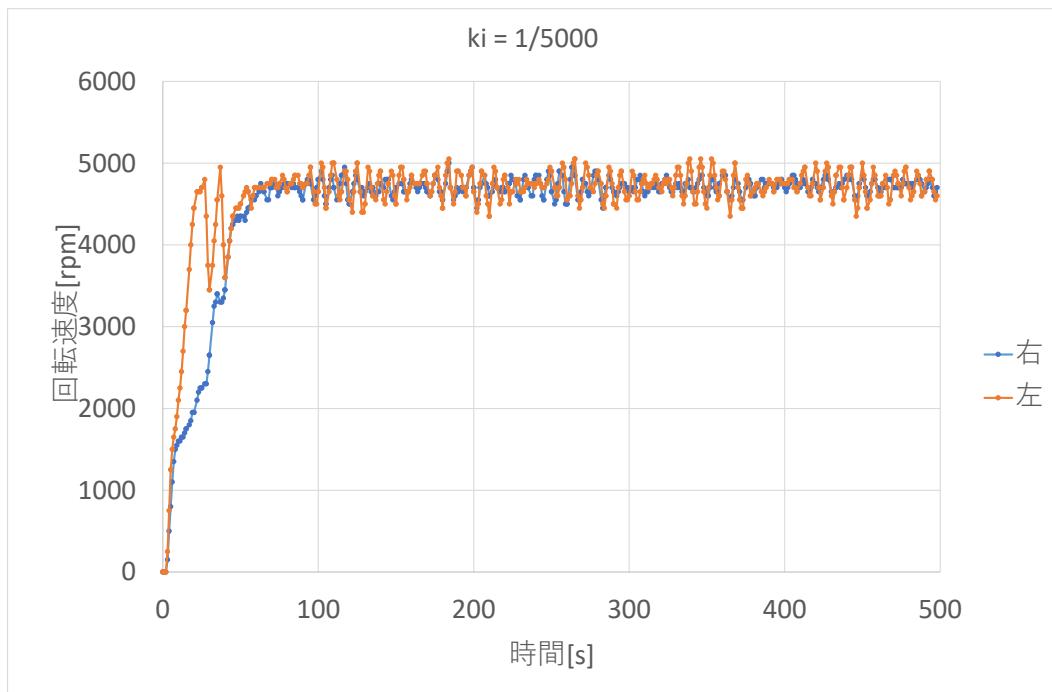
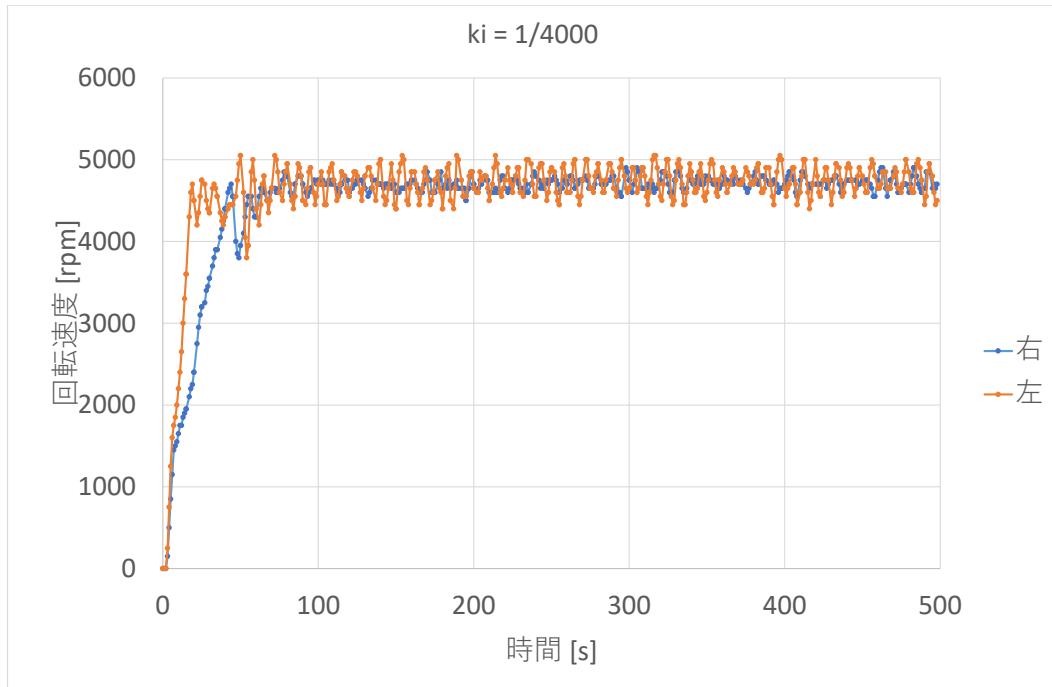
(a)  $k_i = 1/13000$ (b)  $k_i = 1/12000$ 

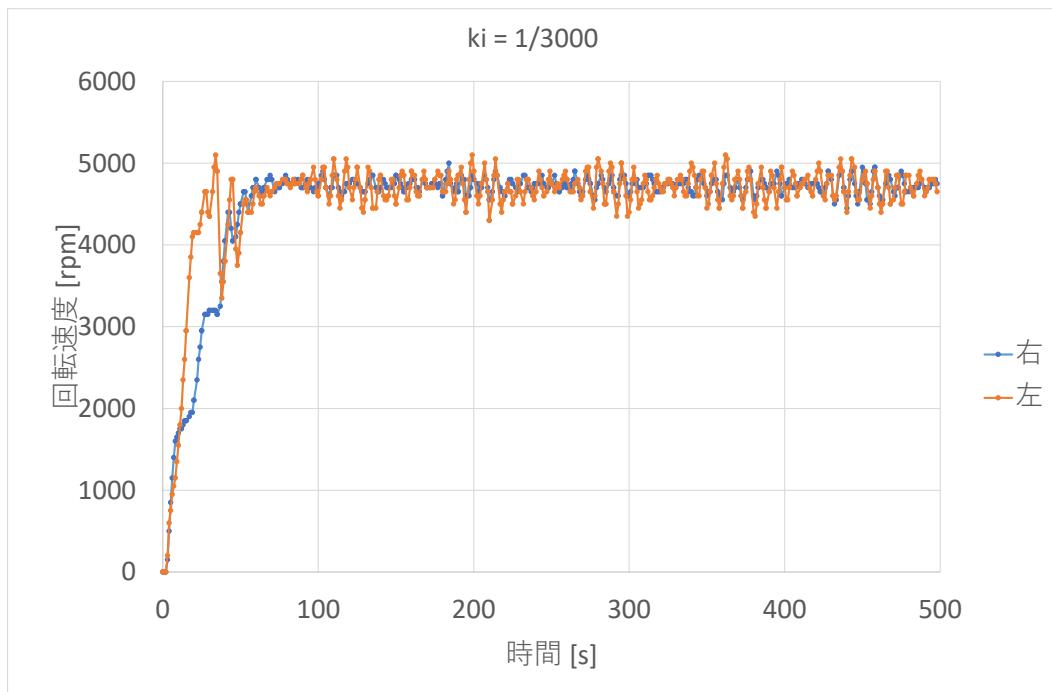
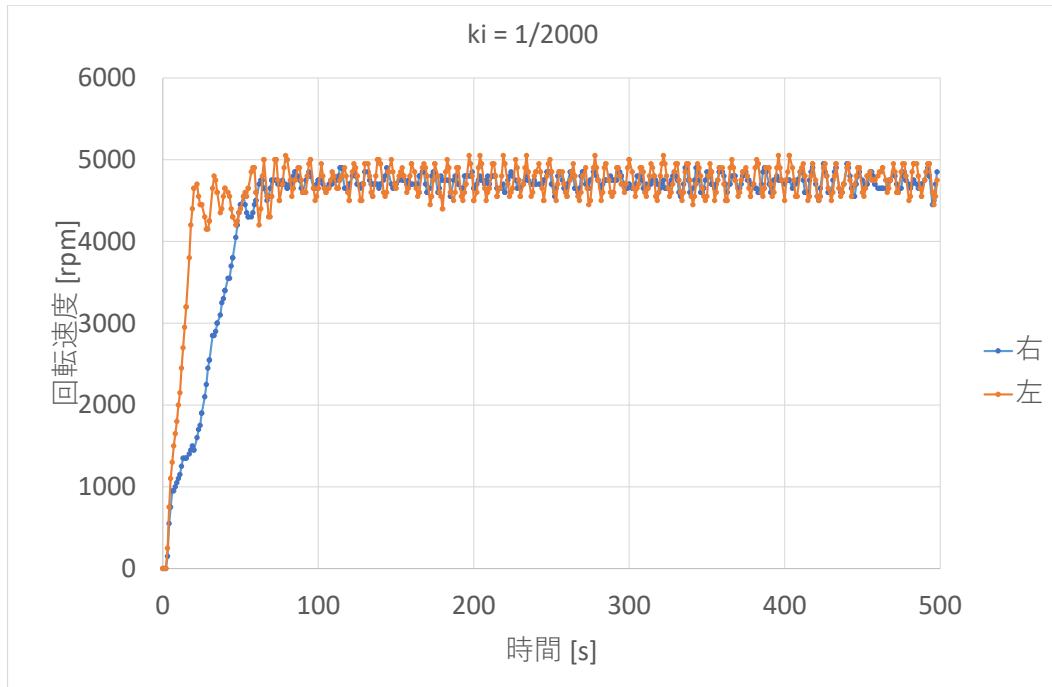
図 12: ステップ応答のグラフ

(a)  $k_i = 1/11000$ (b)  $k_i = 1/10000$ 図 13: ステップ応答のグラフ  
24

(a)  $k_i = 1/9000$ (b)  $k_i = 1/8000$

(c)  $k_i = 1/7000$ (d)  $k_i = 1/6000$

(e)  $k_i = 1/5000$ (f)  $k_i = 1/4000$

(g)  $k_i = 1/3000$ (h)  $k_i = 1/2000$

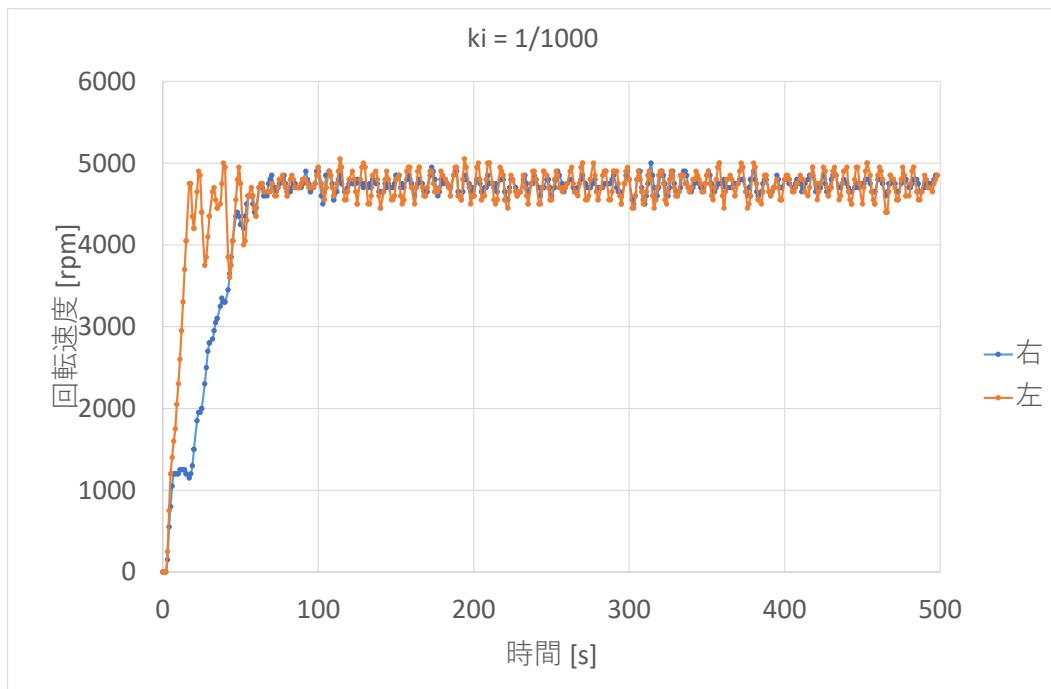
(i)  $k_i = 1/1000$ 

図 14: ステップ応答のグラフ

## 15 演習 15

### 15.1 実行プログラム

実行プログラムをソースコード 15 に示す。

ソースコード 15: 演習 15 のプログラム

---

```

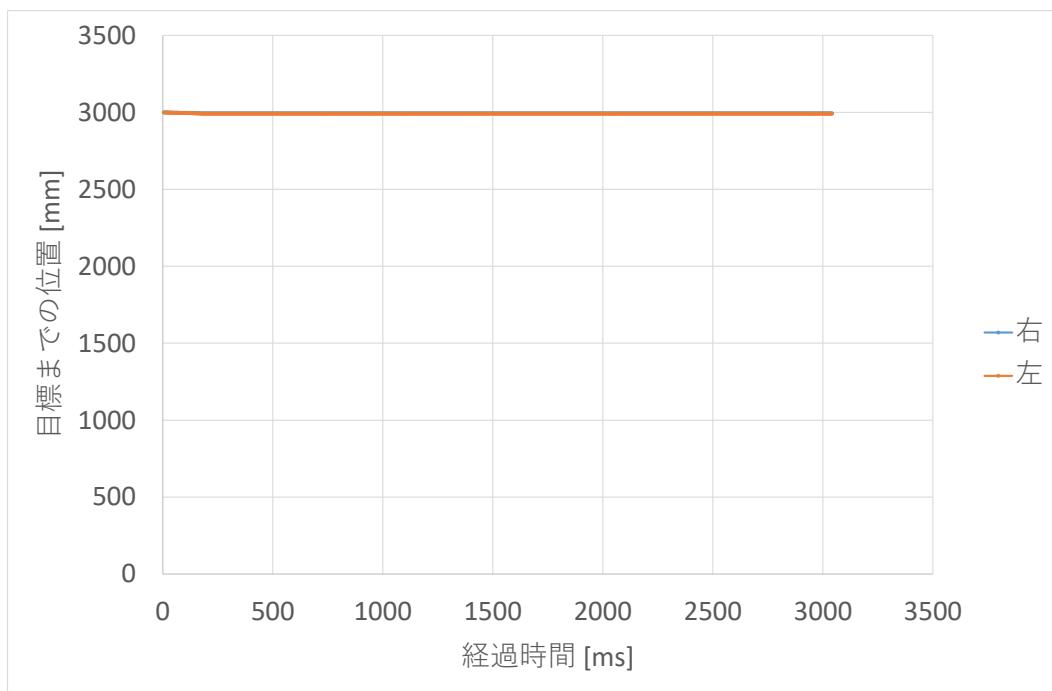
1 #include <stdio.h>
2 #include <process.h> /*関数の定義 exit*/
3 #include <dos.h>
4 #include "v25.h"
5 #include "ms.h"
6
7 #define DMAX 10000
8 #define INTERVAL 100
9 #define DISTANCE 3000
10
11 int main(){
12     int i, n=0;
13     int kp = 3, ki_inv = 10000;
```

```

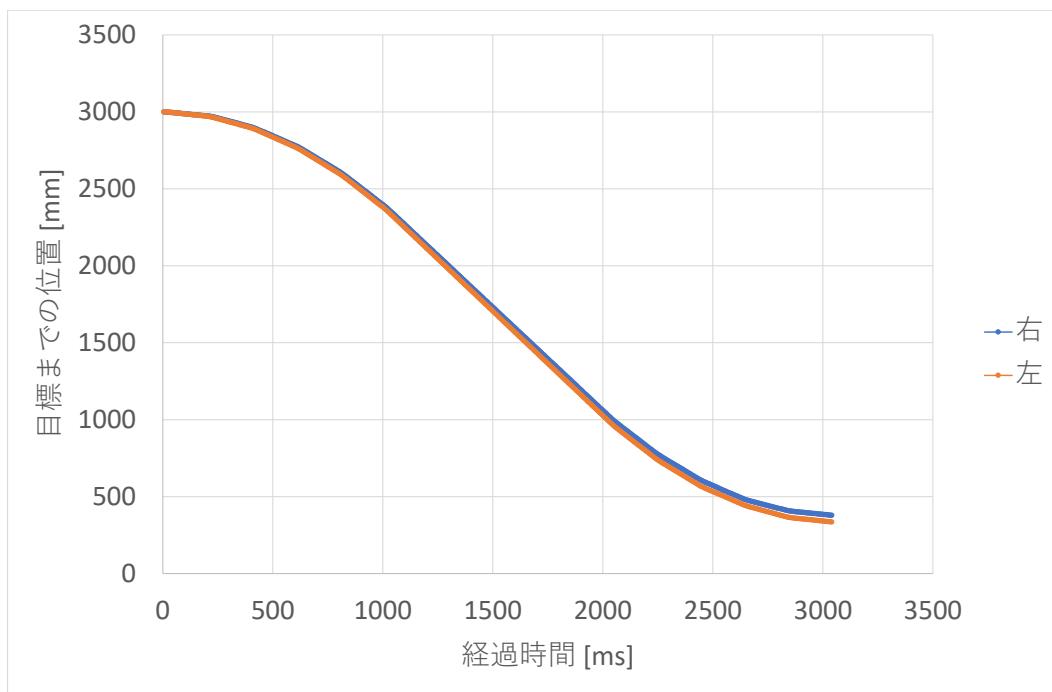
14
15     int vref[11] = {1000, 2000, 3000, 4000, 5000, 6000,
16                     5000, 4000, 3000, 2000, 1000};
17     long time[11] = {200, 200, 200, 200, 200, 1040,
18                     200, 200, 200, 200, 200};
19     long j=0, cl, c10, cr, cr0, ct, ct0, ctm, l[DMAX], r[DMAX], t[DMAX];
20     FILE *fp;
21
22     ms_init();
23     ms_motor_on();
24     ms_set_gain(kp, ki_inv);
25     ms_read_c(&c10, &cr0, &ct0);
26     ctm = ct0;
27     for(i=0; i<11; i++){
28         ms_set_v(vref[i], vref[i]);
29         do{
30             ms_read_c(&cl, &cr, &ct);
31             n = j/INTERVAL;
32             l[n] = cl;
33             r[n] = cr;
34             t[n] = ct;
35             j++;
36         }while(time[i]>ct-ctm);
37         ctm = ct;
38     }
39     ms_motor_off();
40
41     if((fp = fopen("data.dat", "wt")) == NULL){
42         printf("Can't Open File!!\n");
43         exit(1);
44     }
45
46     for(i=0; i<=n; i++){
47         fprintf(fp, "%12.6lf %12.6lf %5ld\n",
48                 DISTANCE-15*2*3.14159*(l[i]-c10)/(19.225*400),
49                 DISTANCE-15*2*3.14159*(r[i]-cr0)/(19.225*400),
50                 t[i]-ct0);
51     }
52     fclose(fp);
53     ms_beep(440,500);
54     return 0;
55 }
```

## 15.2 実行結果

演習 14 と同様にモータの応答(回転速度と目標位置までの距離)を負荷無し、有りの場合分けをしてデータファイルに保存する。負荷無し、有りの場合を図 15(a)、15(b) に示す。負荷有りの場合、ロボットは前進しない。負荷無しの場合、ロボットは目標位置に近づくことが分かる。



(a) 負荷有り



(b) 負荷無し

図 15: モータの応答

## 16 演習 16

### 16.1 実行プログラム

実行プログラムをソースコード 16 に示す。

ソースコード 16: 演習 16 のプログラム

```

1 #include <stdio.h>
2 #include <process.h> /*関数の定義 exit*/
3 #include <dos.h> /*関数の定義 pokeb*/
4 #include "v25.h" /*固有値の定義 V25*/
5 #include "ms.h" /*関数の定義 ms*/
6
7 #define DMAX 10000 保存データ数の最大値*/
8 #define INTERVAL 100 保存データの間隔*/
9 #define INTERVAL2 5 保存データの間隔*/
10 #define DISTANCE 3000 ロボットの移動距離*/
11 #define C_DISTANCE ((DISTANCE*400.*19.225)/(15.*2.*3.14159))ロボットの移動距離カウンタ換算
12 /*()*/
13 #define THRESH 100 閾値*/
14
15 int main()
16 {
17     int i, n = 0, n2 = 0, k = 5;
18     int kp = 6, ki_inv = 10000; /*ゲイン FB*/速度パターン
19     /**
20     int vref[6] = {1000, 2000, 3000, 4000, 5000, 6000};
21     long time[5] = {200, 200, 200, 200, 200};
22     long j = 0, cl, c10, cr, cr0, ct, ct0, ctm, el, er, vl, vr, l[DMAX], r[DMAX], t[DMAX];
23     FILE *fp;
24
25     ms_init(); 初期化*/
26     ms_motor_on(); モータ回路/*ON*/
27     ms_set_gain( kp, ki_inv ); /*ゲインの設定 FB*/
28     ms_read_c( &c10, &cr0, &ct0 ); カウンタ値の初期値の取得*/
29     ctm=ct0;
30     for( i = 0; i < 5; i++ )
31     {
32         ms_set_v( vref[i], vref[i] ); 速度の設定*/
33         do
34         {
35             ms_read_c( &cl, &cr, &ct ); カウンタ値の読み込み*/
36             n = j/INTERVAL;
37             l[n] = cl;
38             r[n] = cr;
39             t[n] = ct;
40             j++;
41         } while ( time[i] > ct-ctm ); 時間間隔のチェック*/
42         ctm = ct;
43     }

```

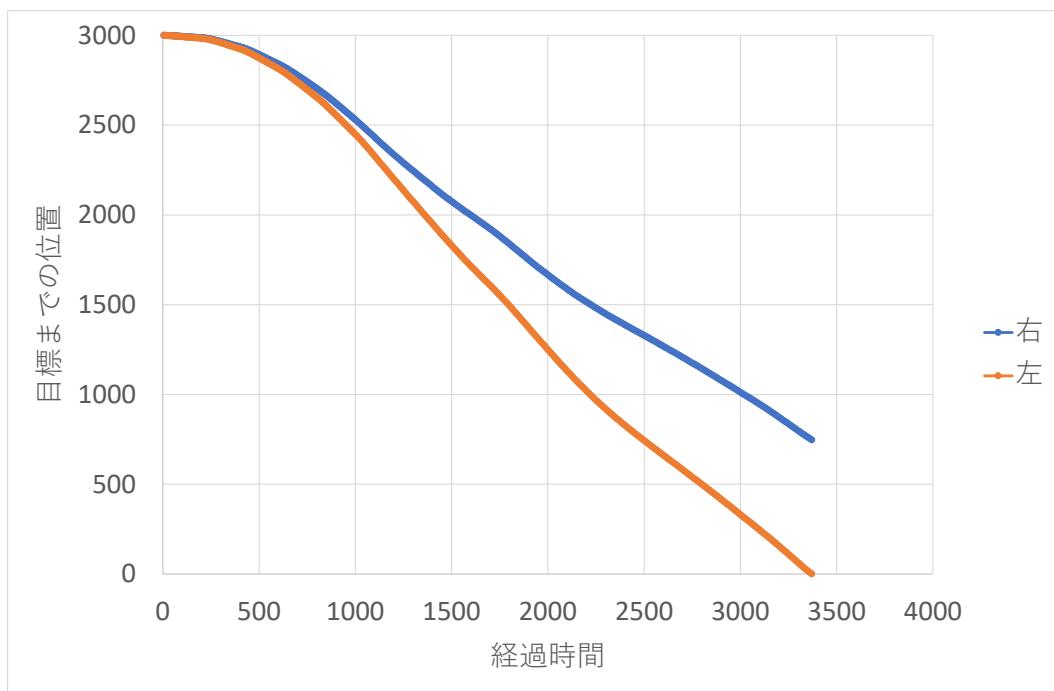
```

44     j = 0;
45     do
46     {
47         el = C_DISTANCE - cl; 目標値との偏差を算出*/
48         er = C_DISTANCE - cr;
49         vl = k * el; 速度の計算*/
50         vr = k * er;
51         if ( vl > vref[5] ) 最高速度の設定*/
52         {
53             vl = vref[5];
54         }
55         if ( vr > vref[5] )
56         {
57             vr = vref[5];
58         }
59         ms_set_v( vl, vr ); 速度の設定*/
60         ms_read_c( &cl, &cr, &ct ); カウンタ値の読み込み*/
61         n2 = j/INTERVAL2 + n + 1;
62         l[n2] = cl;
63         r[n2] = cr;
64         t[n2] = ct;
65         j++;
66     } while ( n2<DMAX && el>THRESH && er>THRESH ); 終了条件のチェック*/
67
68     ms_motor_off(); モータ回路/*OFF*/保存ファイルのオープン
69     /**
70     if ((fp = fopen("data.dat","wt")) == NULL)
71     {
72         printf("Can't open file.\n");
73         exit(1);
74     } ファイルの保存
75     /**
76     for ( i = 0; i < n2; i++)
77     {
78         fprintf(fp,"%12.6lf %12.6lf %5ld\n",DISTANCE-15*2*3.14159*(l[i]-c10)
79             /(19.225*400),DISTANCE-15*2*3.14159*(r[i]-cr0)/(19.225*400),t[i]-ct0);
80     }
81     fclose(fp); ファイルのクローズ*/
82     return 0;
83 }
```

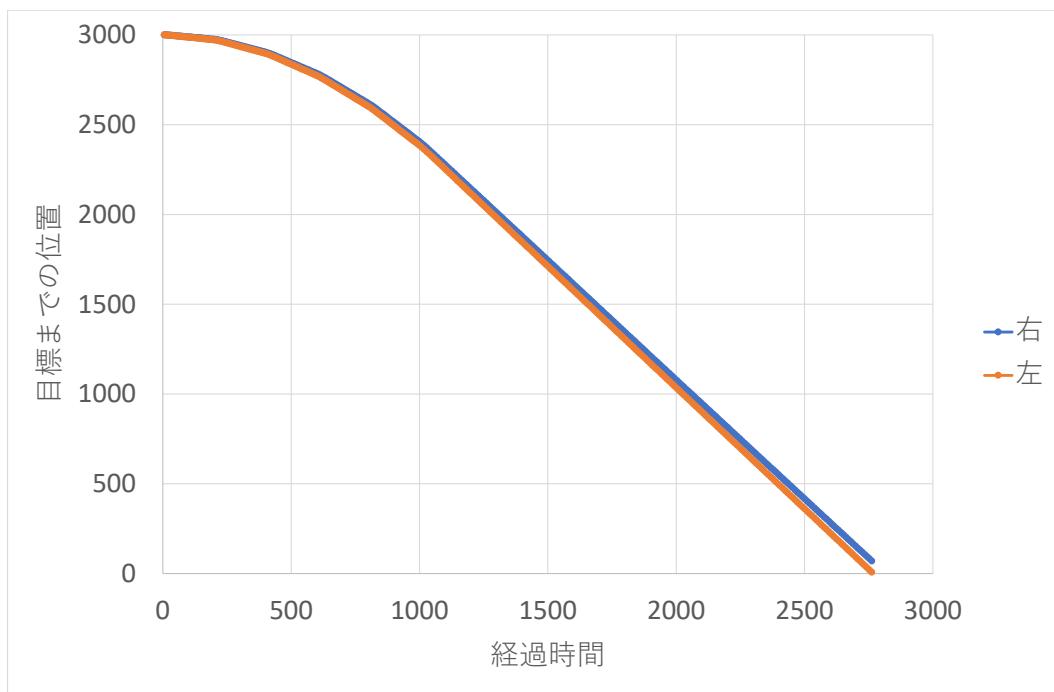
---

## 16.2 実行結果

負荷有りの場合、目標位置までの距離の変化率が左右のモータで均等でない。ロボットの重心が、左右で均等でなかったためであると考えられる。負荷無しの場合、目標位置までの距離が単調的に減少し、目標位置に近づくことが分かる。



(a) 負荷有り



(b) 負荷無し

## 17 演習 17

### 17.1 実行プログラム

実行プログラムをソースコード 17 に示す。

ソースコード 17: 演習 17 のプログラム

```
1 #include <stdio.h>
2 #include <process.h>
3 #include <dos.h>
4 #include "v25.h"
5 #include "ms.h"
6
7 #define DMAX 10000
8 #define INTERVAL 100
9 #define INTERVAL2 5
10 #define DISTANCE 3000
11 #define C_DISTANCE ((DISTANCE*400.*19.225)/(15.*2.*3.14159))
12 #define THRESH 100
13
14 int main()
15 {
16     int i, n = 0, n2 = 0;
17     double k = ms_read_dip();
18     int kp = 3, ki_inv = 10000;
19
20     int vref[6] = {1000, 2000, 3000, 4000, 5000, 6000};
21     long time[5] = {200, 200, 200, 200, 200};
22     long j = 0, cl, c10, cr, cr0, ct, ct0, ctm, el, er, vl, vr, l[DMAX], r[DMAX], t[DMAX];
23     FILE *fp;
24
25     ms_init();
26     ms_motor_on();
27     ms_set_gain( kp, ki_inv );
28     ms_read_c( &c10, &cr0, &ct0 );
29     ctm=ct0;
30     for( i = 0; i < 5; i++ )
31     {
32         ms_set_v( vref[i], vref[i] );
33         do
34         {
35             ms_read_c( &cl, &cr, &ct );
36             n = j/INTERVAL;
37             l[n] = cl;
38             r[n] = cr;
39             t[n] = ct;
40             j++;
41         } while ( time[i] > ct-ctm );
42         ctm = ct;
43     }
```

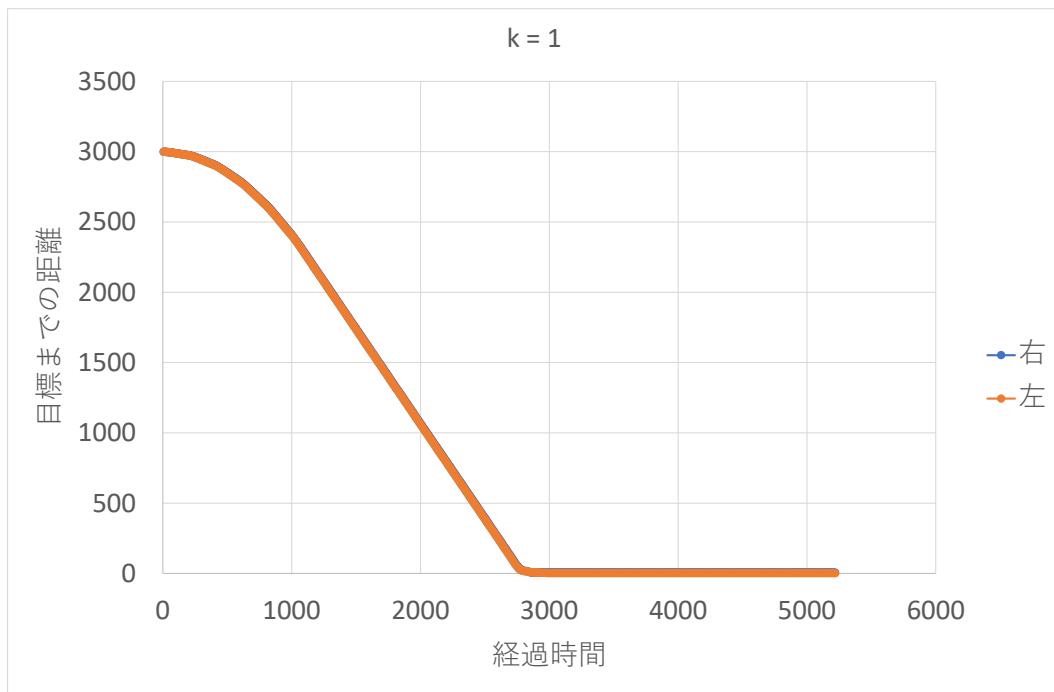
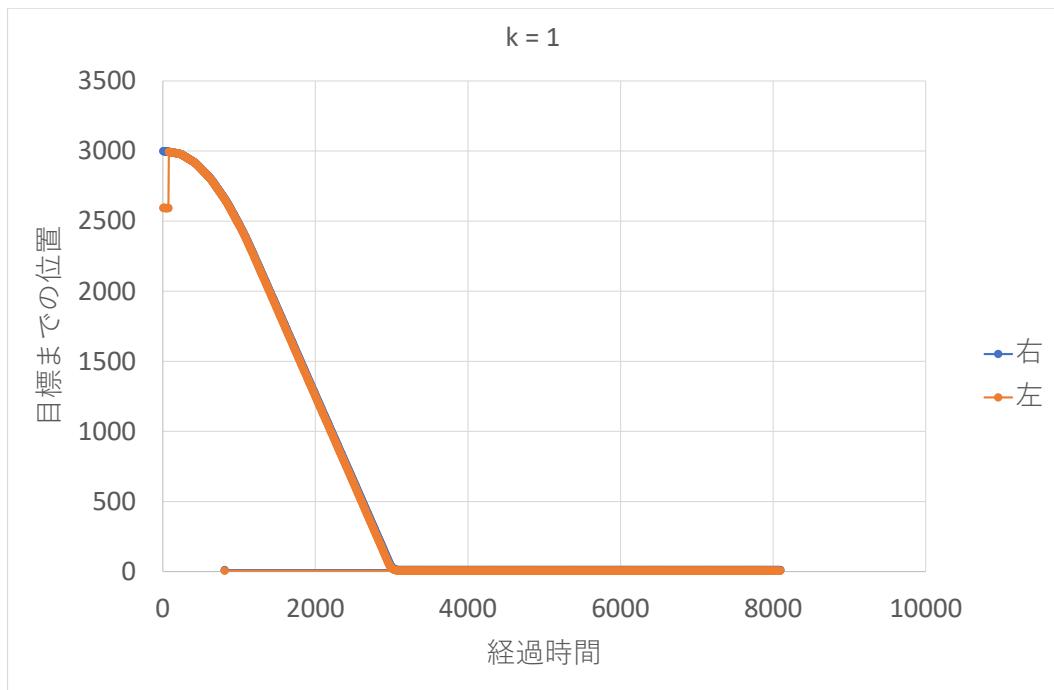
```

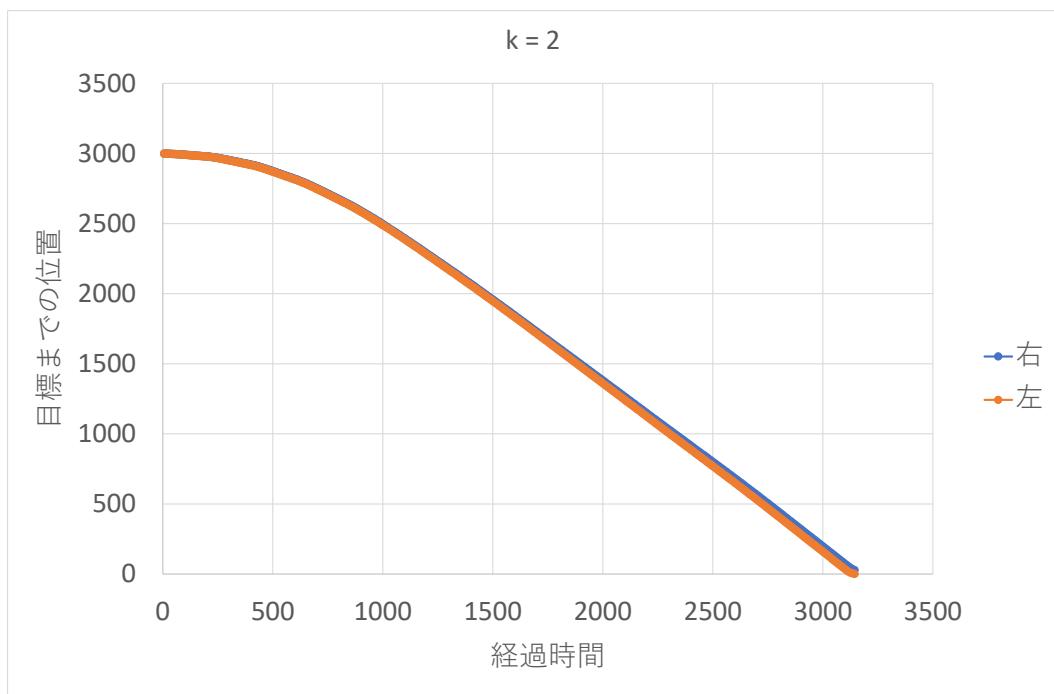
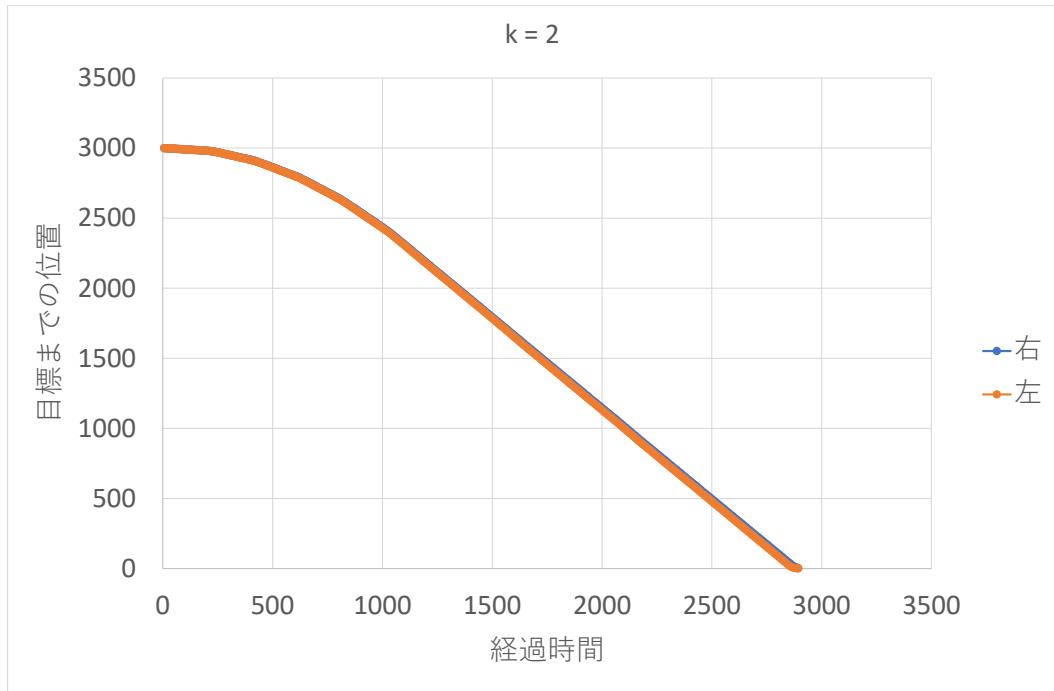
44     j = 0;
45     do
46     {
47         el = C_DISTANCE - cl;
48         er = C_DISTANCE - cr;
49         vl = k * el;
50         vr = k * er;
51         if ( vl > vref[5] )
52         {
53             vl = vref[5];
54         }
55         if ( vr > vref[5] )
56         {
57             vr = vref[5];
58         }
59         ms_set_v( vl, vr );
60         ms_read_c( &cl, &cr, &ct );
61         n2 = j/INTERVAL2 + n + 1;
62         l[n2] = cl;
63         r[n2] = cr;
64         t[n2] = ct;
65         j++;
66     } while ( n2<DMAX && el>THRESH && er>THRESH );
67
68     ms_motor_off();
69
70     if ((fp = fopen("data.dat","wt")) == NULL)
71     {
72         printf("Can't open file.\n");
73         exit(1);
74     }
75
76     for ( i = 0; i < n2; i++)
77     {
78         fprintf(fp,"%12.6lf %12.6lf %5ld\n",DISTANCE-15*2*3.14159*(l[i]-c10)
79                         /(19.225*400),DISTANCE-15*2*3.14159*(r[i]-c10)/(19.225*400),t[i]-ct0);
80     }
81     fclose(fp);
82
83     printf("owari\n");
84     ms_beep(440,500);
85     return 0;
86 }
```

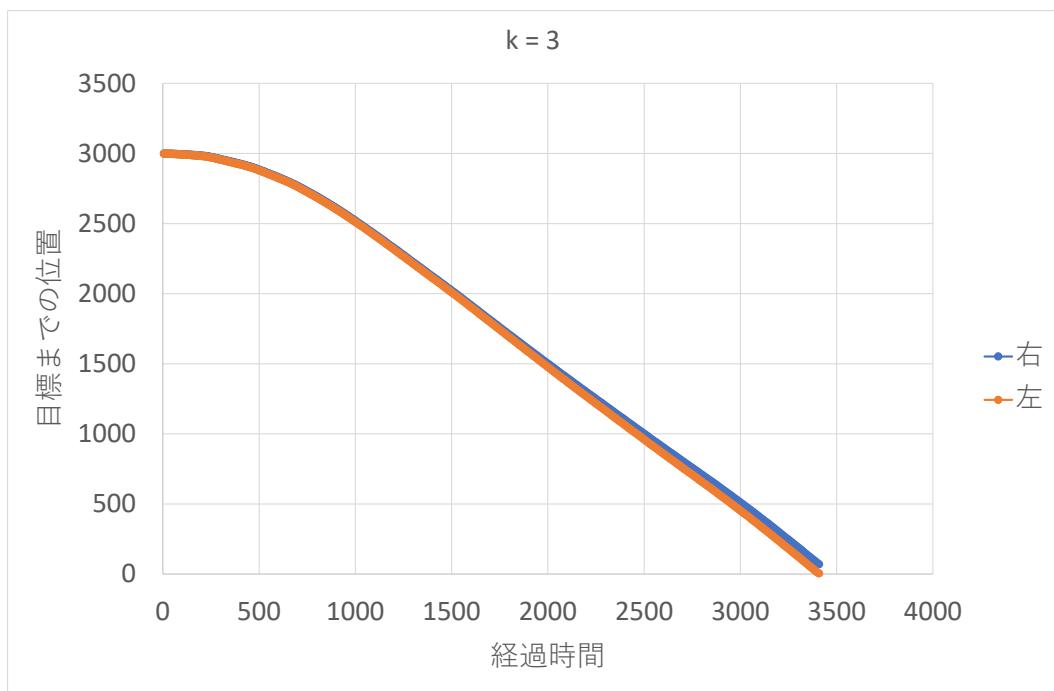
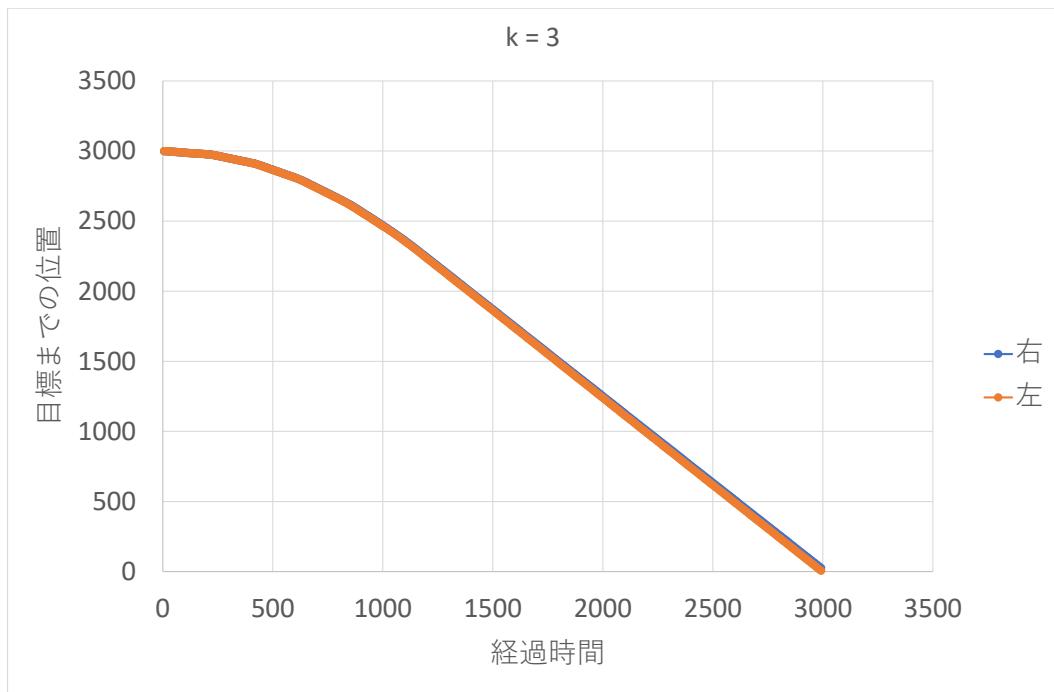
## 17.2 実行結果

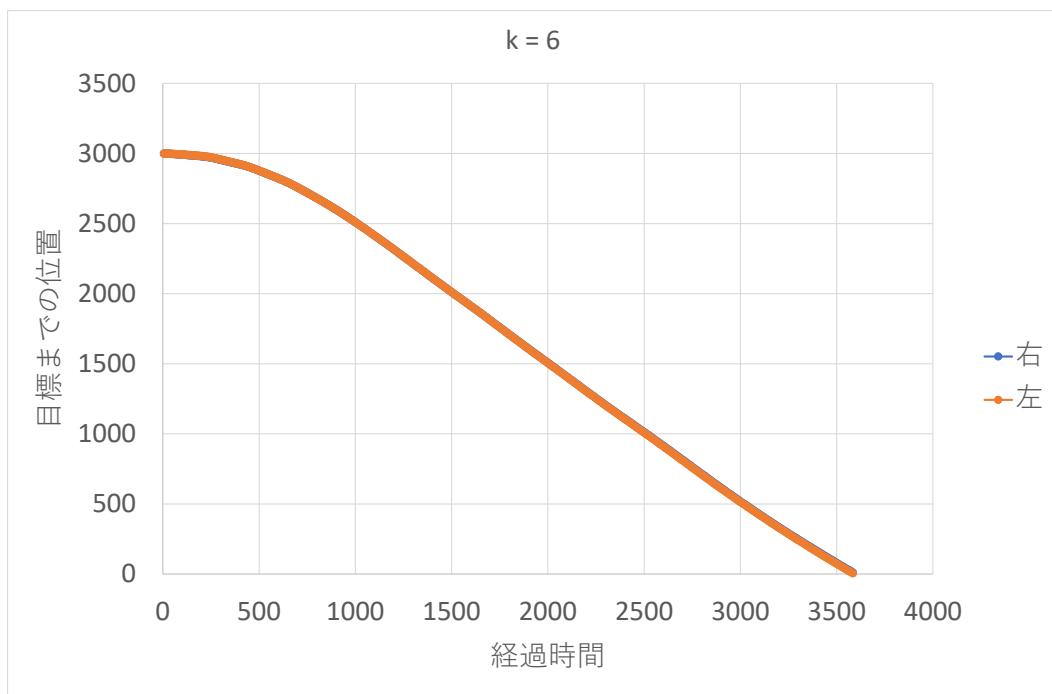
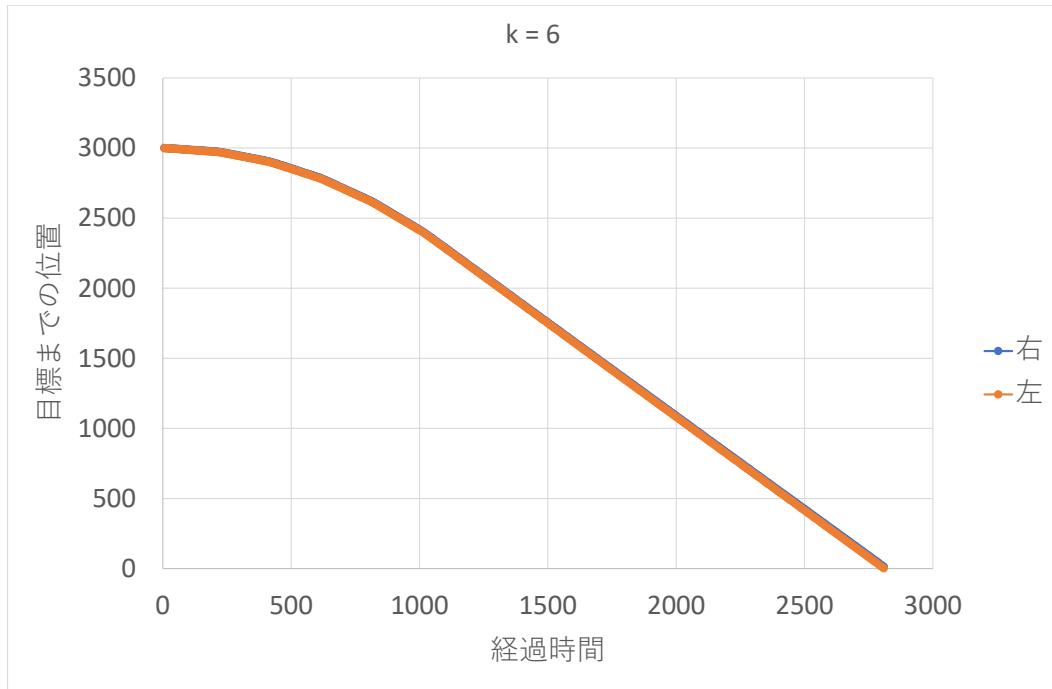
実行結果を図 16(c) から 20(b) に示す。 $k = 1$ において、ロボットが目標位置に到着した際に転倒し、距離データの記録が 5000 秒付近まで続いている。

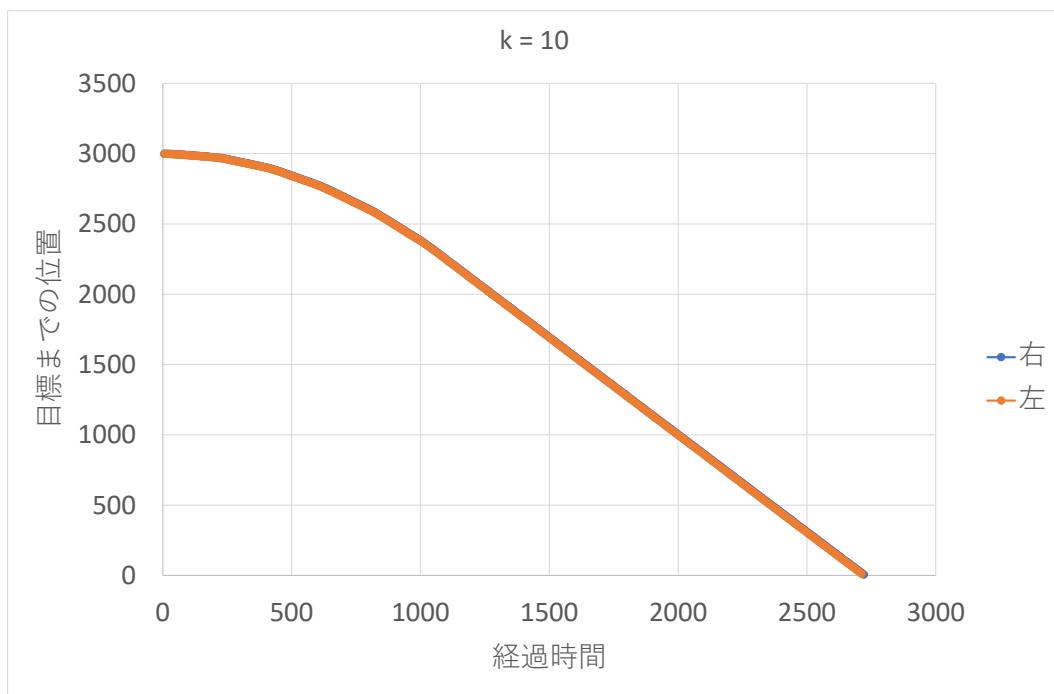
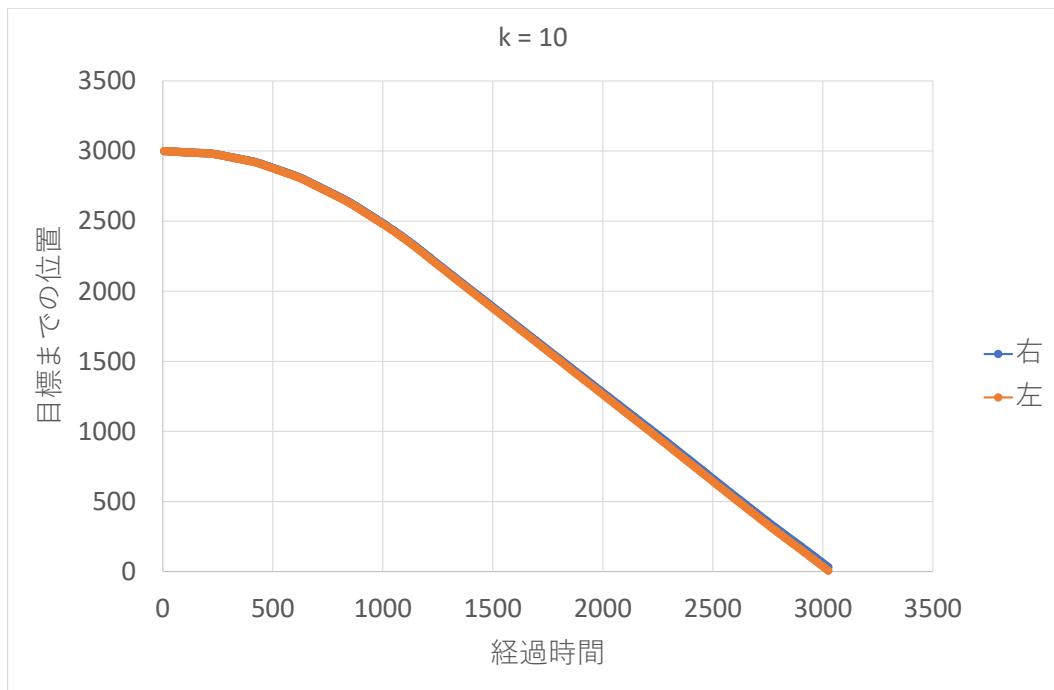
また、負荷有りよりも負荷無しの方が、目標位置への所要時間が少ない。位置のフィードバックゲインの値を大きくすると、目標位置までの所要時間が減少する。

(c)  $k = 1$  負荷有り(d)  $k = 1$  負荷無し図 16:  $k = 1$  のステップ応答

(a)  $k = 2$  負荷有り(b)  $k = 2$  負荷無し

(a)  $k = 3$  負荷有り(b)  $k = 3$  負荷無し

(a)  $k = 6$  負荷有り(b)  $k = 6$  負荷無し図 19:  $k = 6$  のステップ応答

(a)  $k = 10$  負荷有り(b)  $k = 10$  負荷無し

## 18 演習 18

### 18.1 実行プログラム

実行プログラムをソースコード 18 に示す。

ソースコード 18: 演習 18 のプログラム

```
1 #include <stdio.h>
2 #include <math.h>
3 #include <process.h>
4 #include <dos.h>
5 #include "v25.h"
6 #include "ms.h"
7
8 #define D2R (M_PI/180)
9 #define R2D (180/M_PI)
10 #define RADIUS 15.0
11 #define DISTANCE 34.0
12 #define REDUCTION 19.225
13 #define CPR 400
14 #define MMPS (60*REDUCTION/(2.0*M_PI*RADIUS))
15 #define DPC (2.0*M_PI*RADIUS/((float)CPR*REDUCTION))
16 #define APC (DPC/(2*DISTANCE))
17 #define DMAX 1000
18 #define THRESH_D 10
19 #define THRESH_Q 5*D2R
20 #define THRESH_V 50
21
22 #define ABS(x) (x<0?-(x):(x))
23
24 void estimate(double *px,double *py,double *pq,long *pt);
25 void translate(double d,double vmax,double gain);
26 void rotate(double qd,double wmax,double gain);
27
28 int k=0;
29 int xi[DMAX],yi[DMAX],qi[DMAX];
30 long ti[DMAX];
31
32 int main(void)
33 {
34     double x,y,q;
35     int i;
36     int kp=8,ki_inv=8;
37     int vmax = 100;
38     int wmax = 100/DISTANCE;
39     int nl,nr;
40     long t,nt;
41     FILE *fp;
42
43     ms_init();
44     ms_motor_on();
```

```
45     ms_set_gain(kp,ki_inv);
46
47     estimate(&x,&y,&q,&t);
48     rotate(360*D2R,wmax,5);
49     translate(1000,vmax,5);
50
51     ms_set_v(0,0);
52     do{
53         estimate(&x,&y,&q,&t);
54         ms_read_v(&nl,&nr,&nt);
55     }while(nl != 0 || nr != 0);
56
57     ms_motor_off();
58
59     if((fp=fopen("data.dat","wt"))==NULL)
60     {
61         printf("Can't open file...\n");
62     }
63
64     for(i=0;i<k;i++)
65     {
66         fprintf(fp,"%5d %5d %5d %5ld\n",xi[i],yi[i],qi[i],ti[i]);
67     }
68     fclose(fp);
69     return 0;
70 }
71
72 void estimate(double *px,double *py,double *pq,long *pt)
73 {
74     static long cl0=0,cr0=0;
75     static double x=0.0,y=0.0,q=0.0;
76     long cl,cr,ct;
77     int dl,dr;
78     double ds;
79
80     ms_read_c(&cl,&cr,&ct);
81     dl=cl-cl0;
82     dr=cr-cr0;
83     cl0=cl;
84     cr0=cr;
85     ds=DPC/2*(dr+dl);
86     x += ds*cos(q);
87     y += ds*sin(q);
88     q += APC*(dr-dl);
89
90     *px=x;
91     *py=y;
92     *pq=q;
93     *pt=ct;
94     if (k<DMAX)
95     {
96         xi[k]=10*x;
```

```
97         yi[k]=10*y;
98         qi[k]=1000*q;
99         ti[k]=ct;
100        k++;
101    }
102
103 }
104
105 void translate(double d,double vmax,double gain)
106 {
107     double x0,y0,q0,x,y,q;
108     long t0,t,nt;
109     double cosq0,sinq0,dd,vc;
110     int nc, nl, nr;
111
112     estimate(&x0,&y0,&q0,&t0);
113     cosq0=cos(q0);
114     sinq0=sin(q0);
115     x=x0;
116     y=y0;
117     q=q0;
118     t=t0;
119
120     do
121     {
122         dd=cosq0*(x-x0)+sinq0*(y-y0);
123         vc=gain*(d-dd);
124         if (ABS(vc)>vmax)
125         {
126             vc=(vc<0.0)?-vmax:vmax;
127         }
128         nc=vc*MMPS;
129         ms_set_v(nc,nc);
130         estimate(&x,&y,&q,&t);
131         ms_read_v(&nl,&nr,&nt);
132     } while (ABS(d-dd)>THRESH_D||ABS(nl)>THRESH_V||ABS(nr)>THRESH_V);
133 }
134
135 void rotate(double qd,double wmax,double gain)
136 {
137     double x0,y0,q0,x,y,q;
138     long t0,t,nt;
139     double qq,wc,v,dw;
140     int nl, nr;
141
142     estimate(&x0,&y0,&q0,&t0);
143     x=x0;
144     y=y0;
145     q=q0;
146     t=t0;
147
148     do{
```

```

149     qq=q-q0;
150     wc=gain*(qd-qq);
151     if(ABS(wc)>wmax)
152     {
153         wc=(wc<0.0) ? -wmax:wmax;
154     }
155     dw=DISTANCE*wc;
156     nl=-MMPS*dw;
157     nr=MMPS*dw;
158     ms_set_v(nl,nr);
159     estimate(&x,&y,&q,&t);
160     ms_read_v(&nl,&nr,&nt);
161 }while (ABS(qd-qq)>THRESH_Q||ABS(nl)>THRESH_V||ABS(nr)>THRESH_V);
162 }
```

---

## 18.2 実行結果

ロボットが、その場で 360 度回転して 1000[mm] 直進する。回転角度が 360 度ちょうどであるが、直進動作時に、左右のモータの始動に差が生じるため、始動の遅いモータの方にやや曲がってから直進する現象が生じた。

## 19 演習 19

### 19.1 実行プログラム

実行プログラムをソースコード 19 に示す。パラメータ値は、RADIUS を変化させると、回転量が変化し、ロボットの回転と進行距離の両方が代わるため、DISTANCE の値のみを変更した。DISTANCE の実測値を計測し、DISTANCE の値を上下に変化させて、34.0 から 34.05 に変更した。360 度回転は、指示通りに動くが、直進の際の始動にモータの左右差があったため、それを考慮した上で 34.05 の値とした。

ソースコード 19: 演習 19 のプログラム

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <process.h>
4 #include <dos.h>
5 #include "v25.h"
6 #include "ms.h"
7
8 #define D2R (M_PI/180)
9 #define R2D (180/M_PI)
10 #define RADIUS 15.0
11 #define DISTANCE 34.05
12 #define REDUCTION 19.225
13 #define CPR 400
14 #define MMPS (60*REDUCTION/(2.0*M_PI*RADIUS))
15 #define DPC (2.0*M_PI*RADIUS/((float)CPR*REDUCTION))
16 #define APC (DPC/(2*DISTANCE))
17 #define DMAX 1000
18 #define THRESH_D 10
```

```
19 #define THRESH_Q 5*D2R
20 #define THRESH_V 50
21
22 #define ABS(x) (x<0?-(x):(x))
23
24 void estimate(double *px,double *py,double *pq,long *pt);
25 void translate(double d,double vmax,double gain);
26 void rotate(double qd,double wmax,double gain);
27
28 int k=0;
29 int xi[DMAX],yi[DMAX],qi[DMAX];
30 long ti[DMAX];
31
32 int main(void)
33 {
34     double x,y,q;
35     int i;
36     int kp=8,ki_inv=8;
37     int vmax = 100;
38     int wmax = 100/DISTANCE;
39     int nl,nr;
40     long t,nt;
41     FILE *fp;
42
43     ms_init();
44     ms_motor_on();
45     ms_set_gain(kp,ki_inv);
46
47     estimate(&x,&y,&q,&t);
48     rotate(360*D2R,wmax,5);
49     translate(1000,vmax,5);
50
51     ms_set_v(0,0);
52     do{
53         estimate(&x,&y,&q,&t);
54         ms_read_v(&nl,&nr,&nt);
55     }while(nl != 0 || nr != 0);
56
57     ms_motor_off();
58
59     if((fp=fopen("data.dat","wt"))==NULL)
60     {
61         printf("Can't open file...\n");
62     }
63
64     for(i=0;i<k;i++)
65     {
66         fprintf(fp,"%5d %5d %5d %5ld\n",xi[i],yi[i],qi[i],ti[i]);
67     }
68     fclose(fp);
69     return 0;
70 }
```

```
71
72 void estimate(double *px,double *py,double *pq,long *pt)
73 {
74     static long cl0=0,cr0=0;
75     static double x=0.0,y=0.0,q=0.0;
76     long cl,cr,ct;
77     int dl,dr;
78     double ds;
79
80     ms_read_c(&cl,&cr,&ct);
81     dl=cl-cl0;
82     dr=cr-cr0;
83     cl0=cl;
84     cr0=cr;
85     ds=DPC/2*(dr+dl);
86     x += ds*cos(q);
87     y += ds*sin(q);
88     q += APC*(dr-dl);
89
90     *px=x;
91     *py=y;
92     *pq=q;
93     *pt=ct;
94     if (k<DMAX)
95     {
96         xi[k]=10*x;
97         yi[k]=10*y;
98         qi[k]=1000*q;
99         ti[k]=ct;
100        k++;
101    }
102
103 }
104
105 void translate(double d,double vmax,double gain)
106 {
107     double x0,y0,q0,x,y,q;
108     long t0,t,nt;
109     double cosq0,sinq0,dd,vc;
110     int nc,nl,lr;
111
112     estimate(&x0,&y0,&q0,&t0);
113     cosq0=cos(q0);
114     sinq0=sin(q0);
115     x=x0;
116     y=y0;
117     q=q0;
118     t=t0;
119
120     do
121     {
122         dd=cosq0*(x-x0)+sinq0*(y-y0);
```

```

123     vc=gain*(d-dd);
124     if (ABS(vc)>vmax)
125     {
126         vc=(vc<0.0)?-vmax:vmax;
127     }
128     nc=vc*MMPS;
129     ms_set_v(nc,nc);
130     estimate(&x,&y,&q,&t);
131     ms_read_v(&nl,&nr,&nt);
132 } while (ABS(d-dd)>THRESH_D||ABS(nl)>THRESH_V||ABS(nr)>THRESH_V);
133 }
134
135 void rotate(double qd,double wmax,double gain)
136 {
137     double x0,y0,q0,x,y,q;
138     long t0,t,nt;
139     double qq,wc,v,dw;
140     int nl,nr;
141
142     estimate(&x0,&y0,&q0,&t0);
143     x=x0;
144     y=y0;
145     q=q0;
146     t=t0;
147
148     do{
149         qq=q-q0;
150         wc=gain*(qd-qq);
151         if(ABS(wc)>wmax)
152         {
153             wc=(wc<0.0) ? -wmax:wmax;
154         }
155         dw=DISTANCE*wc;
156         nl=-MMPS*dw;
157         nr=MMPS*dw;
158         ms_set_v(nl,nr);
159         estimate(&x,&y,&q,&t);
160         ms_read_v(&nl,&nr,&nt);
161     }while (ABS(qd-qq)>THRESH_Q||ABS(nl)>THRESH_V||ABS(nr)>THRESH_V);
162 }
```

---

## 19.2 実行結果

ロボットが、その場で 360 度回転して 1000[mm] 直進する。回転角度を 360 度より浅くして、直進動作時に左右のモータの始動差を吸収して、360 度方向に直進する。

## 20 演習 20

### 20.1 実行プログラム

実行プログラムをソースコード 20 に示す。ロボットの初期位置を  $(x_s, y_s, \theta_s) = (0, 0, 0)$  としている。

## ソースコード 20: 演習 20 のプログラム

```
1 #include <stdio.h>
2 #include <math.h>
3 #include <process.h>
4 #include <dos.h>
5 #include "v25.h"
6 #include "ms.h"
7
8 #define D2R (M_PI/180)
9 #define R2D (180/M_PI)
10 #define RADIUS 15.0
11 #define DISTANCE 34.05
12 #define REDUCTION 19.225
13 #define CPR 400
14 #define MMPS (60*REDUCTION/(2.0*M_PI*RADIUS))
15 #define DPC (2.0*M_PI*RADIUS/((float)CPR*REDUCTION))
16 #define APC (DPC/(2*DISTANCE))
17 #define DMAX 1000
18 #define THRESH_D 10
19 #define THRESH_Q 5*D2R
20 #define THRESH_V 50
21
22 #define ABS(x) (x<0?-(x):(x))
23
24 void estimate(double *px,double *py,double *pq,long *pt);
25 void translate(double d,double vmax,double gain);
26 void rotate(double qd,double wmax,double gain);
27
28 int k=0;
29 int xi[DMAX],yi[DMAX],qi[DMAX];
30 long ti[DMAX];
31
32 int main(void)
33 {
34     double x,y,q;
35     double x_d=0,x_s=0,y_d=0,y_s=0,th_s=0,th_d=0,th_1=0,th_2=0,a_0=0;
36     int i;
37     int kp=8,ki_inv=8;
38     int vmax = 100;
39     int wmax = 100/DISTANCE;
40     int nl,nr;
41     long t,nt;
42     FILE *fp;
43
44
45
46
47     while(1){
48         ms_init();
49         ms_motor_on();
50         ms_set_gain(kp,ki_inv);
```

```
51     printf("x_d y_d th_d\n");
52     scanf("%lf %lf %lf",&x_d,&y_d,&th_d);
53     estimate(&x,&y,&q,&t);
54     th_1=atan2(y_d-y_s,x_d-x_s)-th_s;
55     rotate(th_1,wmax,5);
56     a_0=sqrt((x_d-x_s)*(x_d-x_s)+(y_d-y_s)*(y_d-y_s));
57     translate(a_0,vmax,5);
58     th_2=th_d*D2R-th_1-th_s;
59     rotate(th_2,wmax,5);
60     printf("aaa\n");
61     x_s=x_d;
62     y_s=y_d;
63     th_s=th_d;
64
65     ms_set_v(0,0);
66     do{
67         estimate(&x,&y,&q,&t);
68         ms_read_v(&nl,&nr,&nt);
69     }while(nl != 0 || nr != 0);
70
71     ms_motor_off();
72 }
73 return 0;
74 }

75

76 void estimate(double *px,double *py,double *pq,long *pt)
77 {
78     static long cl0=0,cr0=0;
79     static double x=0.0,y=0.0,q=0.0;
80     long cl,cr,ct;
81     int dl,dr;
82     double ds;
83
84     ms_read_c(&cl,&cr,&ct);
85     dl=cl-cl0;
86     dr=cr-cr0;
87     cl0=cl;
88     cr0=cr;
89     ds=DPC/2*(dr+dl);
90     x += ds*cos(q);
91     y += ds*sin(q);
92     q += APC*(dr-dl);
93
94     *px=x;
95     *py=y;
96     *pq=q;
97     *pt=ct;
98     if (k<DMAX)
99     {
100         xi[k]=10*x;
101         yi[k]=10*y;
102         qi[k]=1000*q;
```

```
103     ti[k]=ct;
104     k++;
105 }
106
107 }
108
109 void translate(double d,double vmax,double gain)
110 {
111     double x0,y0,q0,x,y,q;
112     long t0,t,nt;
113     double cosq0,sinq0,dd,vc;
114     int nc,nl,nr;
115
116     estimate(&x0,&y0,&q0,&t0);
117     cosq0=cos(q0);
118     sinq0=sin(q0);
119     x=x0;
120     y=y0;
121     q=q0;
122     t=t0;
123
124     do
125     {
126         dd=cosq0*(x-x0)+sinq0*(y-y0);
127         vc=gain*(d-dd);
128         if (ABS(vc)>vmax)
129         {
130             vc=(vc<0.0)?-vmax:vmax;
131         }
132         nc=vc*MMPS;
133         ms_set_v(nc,nc);
134         estimate(&x,&y,&q,&t);
135         ms_read_v(&nl,&nr,&nt);
136     } while (ABS(d-dd)>THRESH_D||ABS(nl)>THRESH_V||ABS(nr)>THRESH_V);
137 }
138
139 void rotate(double qd,double wmax,double gain)
140 {
141     double x0,y0,q0,x,y,q;
142     long t0,t,nt;
143     double qq,wc,v,dw;
144     int nl,nr;
145
146     estimate(&x0,&y0,&q0,&t0);
147     x=x0;
148     y=y0;
149     q=q0;
150     t=t0;
151
152     do{
153         qq=q-q0;
154         wc=gain*(qd-qq);
```

---

```

155     if(ABS(wc)>wmax)
156     {
157         wc=(wc<0.0) ? -wmax:wmax;
158     }
159     dw=DISTANCE*wc;
160     nl=-MMPS*dw;
161     nr=MMPS*dw;
162     ms_set_v(nl,nr);
163     estimate(&x,&y,&q,&t);
164     ms_read_v(&nl,&nr,&nt);
165     printf("%lf %lf %lf\n", ABS(qd-qq), ABS(nl), ABS(nr));
166 }while (ABS(qd-qq)>THRESH_Q || ABS(nl)>THRESH_V || ABS(nr)>THRESH_V);
167 }
```

---

## 20.2 実行結果

目標位置  $(x_d, y_d, \theta_d)$  を入力すると、目標方向を向き、目標の座標  $(x_d, y_d)$  の方向に直進し、到着すると  $\theta_d$  方向に向きを変える。目標位置・姿勢の状態をとると、初期値を現在値に更新し、再び目標位置の入力待機を行う。その後、目標位置の入力と移動の処理を繰り返して行うことができる。その際、目標位置の入力前にモータの電圧をオンと初期化、出力後にモータの電圧をオフを毎ループごとに繰り返し行う。

# 21 演習 21

## 21.1 実行プログラム

実行プログラムをソースコード 21 に示す。

ソースコード 21: 演習 21 のプログラム

---

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <process.h>
4 #include <dos.h>
5 #include "v25.h"
6 #include "ms.h"
7
8 #define D2R (M_PI/180)
9 #define R2D (180/M_PI)
10 #define RADIUS 15.0
11 #define DISTANCE 34.05
12 #define REDUCTION 19.225
13 #define CPR 400
14 #define MMPS (60*REDUCTION/(2.0*M_PI*RADIUS))
15 #define DPC (2.0*M_PI*RADIUS/((float)CPR*REDUCTION))
16 #define APC (DPC/(2*DISTANCE))
17 #define DMAX 1000
18 #define THRESH_D 10
19 #define THRESH_Q 5*D2R
20 #define THRESH_V 1
21
22 #define GOALX 1200
```

```
23 #define GOALY 1000
24 #define GOALQ 0
25 #define VMAX 150
26 #define WMAX (VMAX/(3*DISTANCE))
27 #define REVERSE 200
28
29 #define ABS(x) (x<0?-(x):(x))
30
31 void estimate(double *px,double *py,double *pq,long *pt);
32 void translate(double d,double vmax,double gain);
33 int translate2(double d,double vmax,double gain);
34 void rotate(double qd,double wmax,double gain);
35 double seikika(double angle);
36 void avoid(int j);
37
38 int k=0;
39 int xi[DMAX],yi[DMAX],qi[DMAX];
40 long ti[DMAX];
41
42 int main(void)
43 {
44     double x,y,q,q1,q2,d1,dq;
45     int i,j;
46     int kp=8,ki_inv=8;
47     int nl,nr;
48     long t,nt;
49     FILE *fp;
50
51     ms_init();
52     ms_set_gain(kp,ki_inv);
53     ms_motor_on();
54     ms_wait(1000);
55
56     while (1)
57     {
58         estimate(&x,&y,&q,&t);
59
60         d1=sqrt((GOALX-x)*(GOALX-x)+(GOALY-y)*(GOALY-y));
61         if(d1<THRESH_D) break;
62
63         dq=atan2(GOALY-y,GOALX-x)-q;
64         dq=seikika(dq);
65         rotate(dq,WMAX,5);
66
67         j=translate2(d1,VMAX,5);
68
69         avoid(j);
70     }
71
72     while (1)
73     {
74         estimate(&x,&y,&q,&t);
```

```
75
76     dq=seikika(GOALQ-q);
77     if (ABS(dq)<THRESH_Q) break;
78
79     rotate(dq,WMAX,5);
80 }
81
82 ms_motor_off();
83
84 if((fp=fopen("data.dat","wt"))==NULL)
85 {
86     printf("Can't open file...\n");
87     exit(1);
88 }
89
90 for(i=0;i<k;i++)
91 {
92     fprintf(fp,"%5d %5d %5d %5ld\n",xi[i],yi[i],qi[i],ti[i]);
93 }
94 fclose(fp);
95 return 0;
96 }
97
98 void estimate(double *px,double *py,double *pq,long *pt)
99 {
100     static long cl0=0,cr0=0;
101     static double x=0.0,y=0.0,q=0.0;
102     long cl,cr,ct;
103     int dl,dr;
104     double ds;
105
106     ms_read_c(&cl,&cr,&ct);
107     dl=cl-cl0;
108     dr=cr-cr0;
109     cl0=cl;
110     cr0=cr;
111     ds=DPC/2*(dr+dl);
112     x += ds*cos(q);
113     y += ds*sin(q);
114     q += APC*(dr-dl);
115
116     *px=x;
117     *py=y;
118     *pq=q;
119     *pt=ct;
120     if (k<DMAX)
121     {
122         xi[k]=10*x;
123         yi[k]=10*y;
124         qi[k]=1000*q;
125         ti[k]=ct;
126         k++;
127     }
128 }
```

```
127     }
128
129 }
130
131 void translate(double d,double vmax,double gain)
132 {
133     double x0,y0,q0,x,y,q;
134     long t0,t,nt;
135     double cosq0,sinq0,dd,vc;
136     int nc,nl,nr;
137
138     estimate(&x0,&y0,&q0,&t0);
139     cosq0=cos(q0);
140     sinq0=sin(q0);
141     x=x0;
142     y=y0;
143     q=q0;
144     t=t0;
145
146     do
147     {
148         dd=cosq0*(x-x0)+sinq0*(y-y0);
149         vc=gain*(d-dd);
150         if (ABS(vc)>vmax)
151         {
152             vc=(vc<0.0)?-vmax:vmax;
153         }
154         nc=vc*MMPS;
155         ms_set_v(nc,nc);
156         estimate(&x,&y,&q,&t);
157         ms_read_v(&nl,&nr,&nt);
158     } while (ABS(d-dd)>THRESH_D||ABS(nl)>THRESH_V||ABS(nr)>THRESH_V);
159 }
160
161 int translate2(double d,double vmax,double gain)
162 {
163     double x0,y0,q0,x,y,q,g;
164     long t0,t,nt;
165     double cosq0,sinq0,dd,vc;
166     int nc,nl,nr,s[3],check;
167
168     estimate(&x0,&y0,&q0,&t0);
169     cosq0=cos(q0);
170     sinq0=sin(q0);
171     x=x0;
172     y=y0;
173     q=q0;
174     t=t0;
175
176     do
177     {
178         ms_ifr(s);
```

```
179     check = 1*s[0]+2*s[1]+4*s[2];
180     ms_read_v(&nl,&nr,&nt);
181
182     if (check)
183     {
184         for (g = 1; g > 0; g-=0.01)
185             ms_set_v((int)(nl*g),(int)(nr*g));
186
187         return check;
188     }
189
190     dd=cosq0*(x-x0)+sinq0*(y-y0);
191
192     vc=gain*(d-dd);
193
194     if (ABS(vc)>vmax)
195     {
196         vc = (vc < 0.0) ? -vmax : vmax;
197     }
198     nc=vc*MMPS;
199     ms_set_v(nc,nc);
200     estimate(&x,&y,&q,&t);
201     ms_read_v(&nl,&nr,&nt);
202 } while (ABS(d-dd)>THRESH_D || ABS(nl)>THRESH_V || ABS(nr)>THRESH_V);
203 return 0;
204 }
205
206 void rotate(double qd,double wmax,double gain)
207 {
208     double x0,y0,q0,x,y,q;
209     long t0,t,nt;
210     double qq,wc,v,dw;
211     int nl,nr;
212
213     estimate(&x0,&y0,&q0,&t0);
214     x=x0;
215     y=y0;
216     q=q0;
217     t=t0;
218
219     do{
220         qq=q-q0;
221         wc=gain*(qd-qq);
222         if(ABS(wc)>wmax)
223         {
224             wc=(wc<0.0) ? -wmax:wmax;
225         }
226         dw=DISTANCE*wc;
227         nl=-MMPS*dw;
228         nr=MMPS*dw;
229         ms_set_v(nl,nr);
230         estimate(&x,&y,&q,&t);
231     }
```

---

```

231     ms_read_v(&nl,&nr,&nt);
232     }while (ABS(qd-qq)>THRESH_Q||ABS(nl)>THRESH_V||ABS(nr)>THRESH_V);
233 }
234
235 double seikika(double angle)
236 {
237     while(angle>M_PI) angle -= 2*M_PI;
238     while(angle<=-M_PI) angle += 2*M_PI;
239     return angle;
240 }
241
242 void avoid(int j)
243 {
244     if(j!=0)
245     {
246         translate(-REVERSE,VMAX,5);
247         if(j&1) rotate(-60*D2R,WMAX,5);
248         else rotate(60*D2R,WMAX,5);
249         translate(REVERSE,VMAX,5);
250     }
251 }
```

---

## 21.2 実行結果

ロボットを起動させると、前進し、近接センサが障害物に反応すると、ロボットは後退し、一旦後退する。その後センサ値が 1 であれば 60 度右に旋回し、それ以外ならば、左に 60 度旋回する。その後、前方に進む。

# 22 演習 22

## 22.1 実行プログラム

実行プログラムをソースコード 22 に示す。本プログラムは、移動平均法を用いて、近接センサの値を平均化し、標準化している。過去 5 つのセンサデータ (0 か 1) を保持し、その平均値が 0.8 以上なら障害物を検知した場合の処理を行う。

ソースコード 22: 演習 22 のプログラム

---

```

1 #include <stdio.h>
2 #include <math.h>
3 #include <process.h>
4 #include <dos.h>
5 #include "v25.h"
6 #include "ms.h"
7
8 #define D2R (M_PI/180)
9 #define R2D (180/M_PI)
10 #define RADIUS 15.0
11 #define DISTANCE 34.05
12 #define REDUCTION 19.225
13 #define CPR 400
14 #define MMPS (60*REDUCTION/(2.0*M_PI*RADIUS))
```

```
15 #define DPC (2.0*M_PI*RADIUS/((float)CPR*REDUCTION))
16 #define APC (DPC/(2*DISTANCE))
17 #define DMAX 1000
18 #define THRESH_D 10
19 #define THRESH_Q 5*D2R
20 #define THRESH_V 1
21
22 #define GOALX 1200
23 #define GOALY 1000
24 #define GOALQ 0
25 #define VMAX 150
26 #define WMAX (VMAX/(3*DISTANCE))
27 #define REVERSE 200
28
29 #define ABS(x) (x<0?-(x):(x))
30
31 #define WIDTH 10 センサのデータを保存する個数/**/
32
33 void estimate(double *px,double *py,double *pq,long *pt);
34 void translate(double d,double vmax,double gain);
35 int translate2(double d,double vmax,double gain);
36 void rotate(double qd,double wmax,double gain);
37 double seikika(double angle);
38 void avoid(int j);
39
40 int k=0;
41 int xi[DMAX],yi[DMAX],qi[DMAX];
42 long ti[DMAX];
43
44     int sum_s [3]={0,0,0}; 直近個のデータの合計/*10 左中央右0:,1:,2:(bs)*/
45     int bp=0; 最新データの場所/**/
46     int data_s[3][WIDTH]; 直近個のデータ/*10 (buff)*/
47
48 int main(void)
49 {
50     double x,y,q,q1,q2,d1,dq;
51     int i,j;
52     int kp=8,ki_inv=8;
53     int nl,nr;
54     long t,nt;
55     FILE *fp;
56
57     ms_init();
58     ms_set_gain(kp,ki_inv);
59     ms_motor_on();
60     ms_wait(1000);
61
62     while (1)
63     {
64         estimate(&x,&y,&q,&t);
65
66         d1=sqrt((GOALX-x)*(GOALX-x)+(GOALY-y)*(GOALY-y));
```

```
67         if(dl<THRESH_D) break;
68
69         dq=atan2(GOALY-y,GOALX-x)-q;
70         dq=seikika(dq);
71         rotate(dq,WMAX,5);
72
73         j=translate2(dl,VMAX,5);
74
75         avoid(j);
76     }
77
78     while (1)
79     {
80         estimate(&x,&y,&q,&t);
81
82         dq=seikika(GOALQ-q);
83         if (ABS(dq)<THRESH_Q) break;
84
85         rotate(dq,WMAX,5);
86     }
87
88     ms_motor_off();
89
90     if((fp=fopen("data.dat","wt"))==NULL)
91     {
92         printf("Can't open file...\n");
93         exit(1);
94     }
95
96     for(i=0;i<k;i++)
97     {
98         fprintf(fp,"%5d %5d %5d %5ld\n",xi[i],yi[i],qi[i],ti[i]);
99     }
100    fclose(fp);
101    return 0;
102 }
103
104 void estimate(double *px,double *py,double *pq,long *pt)
105 {
106     static long c10=0,cr0=0;
107     static double x=0.0,y=0.0,q=0.0;
108     long cl,cr,ct;
109     int dl,dr;
110     double ds;
111
112     ms_read_c(&cl,&cr,&ct);
113     dl=cl-c10;
114     dr=cr-cr0;
115     c10=cl;
116     cr0=cr;
117     ds=DPC/2*(dr+dl);
118     x += ds*cos(q);
```

```
119     y += ds*sin(q);
120     q += APC*(dr-dl);
121
122     *px=x;
123     *py=y;
124     *pq=q;
125     *pt=ct;
126     if (k<DMAX)
127     {
128         xi[k]=10*x;
129         yi[k]=10*y;
130         qi[k]=1000*q;
131         ti[k]=ct;
132         k++;
133     }
134
135 }
136
137 void translate(double d,double vmax,double gain)
138 {
139     double x0,y0,q0,x,y,q;
140     long t0,t,nt;
141     double cosq0,sinq0,dd,vc;
142     int nc,nl,nr;
143
144     estimate(&x0,&y0,&q0,&t0);
145     cosq0=cos(q0);
146     sinq0=sin(q0);
147     x=x0;
148     y=y0;
149     q=q0;
150     t=t0;
151
152     do
153     {
154         dd=cosq0*(x-x0)+sinq0*(y-y0);
155         vc=gain*(d-dd);
156         if (ABS(vc)>vmax)
157         {
158             vc=(vc<0.0)?-vmax:vmax;
159         }
160         nc=vc*MMPS;
161         ms_set_v(nc,nc);
162         estimate(&x,&y,&q,&t);
163         ms_read_v(&nl,&nr,&nt);
164     } while (ABS(d-dd)>THRESH_D||ABS(nl)>THRESH_V||ABS(nr)>THRESH_V);
165 }
166
167 int translate2(double d,double vmax,double gain)
168 {
169     double x0,y0,q0,x,y,q,g;
170     long t0,t,nt;
```

```
171     double cosq0,sinq0,dd,vc;
172     int nc,nl,nr,s[3],check;
173
174     int i=0; センサ左中右切り替え用/**/
175
176     estimate(&x0,&y0,&q0,&t0);
177     cosq0=cos(q0);
178     sinq0=sin(q0);
179     x=x0;
180     y=y0;
181     q=q0;
182     t=t0;
183
184     do
185     {
186         ms_ifr(s);過去
187
188         /*個のデータの平均で WIDTHs[i]を判断*/
189         for(i=0;i<3;i++)
190         {
191             // WIDTH 個のデータを捨てるために合計から引く
192             sum_s[i] = sum_s[i] - data_s[i][bp];
193
194             data_s[i][bp] = s[i]; // 最新データを保存
195
196             sum_s[i] = sum_s[i] + s[i]; // 最新のデータで合計
197
198             // 直近のデータを覚える場所を移動
199             bp++;
200             if ( bp >= WIDTH )
201             {
202                 bp = 0;
203             }
204
205             // 直近w d t h個のデータ割る個数で出した平均が1以上の場合にsを1、1未満の場合sをとする
206             if((sum_s[i]/WIDTH)>=0.8)
207             {
208                 s[i]=1;
209             }else{
210                 s[i]=0;
211             }
212         }
213
214         check = 1*s[0]+2*s[1]+4*s[2];
215         ms_read_v(&nl,&nr,&nt);
216
217         if (check)
218         {
219             for (g = 1; g > 0; g-=0.01)
220                 ms_set_v((int)(nl*g),(int)(nr*g));
221     }
```

```
222         return check;
223     }
224
225     dd=cosq0*(x-x0)+sinq0*(y-y0);
226
227     vc=gain*(d-dd);
228
229     if (ABS(vc)>vmax)
230     {
231         vc = (vc < 0.0) ? -vmax : vmax;
232     }
233     nc=vc*MMPS;
234     ms_set_v(nc,nc);
235     estimate(&x,&y,&q,&t);
236     ms_read_v(&nl,&nr,&nt);
237 } while (ABS(d-dd)>THRESH_D || ABS(nl)>THRESH_V || ABS(nr)>THRESH_V);
238 return 0;
239 }
240
241 void rotate(double qd,double wmax,double gain)
242 {
243     double x0,y0,q0,x,y,q;
244     long t0,t,nt;
245     double qq,wc,v,dw;
246     int nl,nr;
247
248     estimate(&x0,&y0,&q0,&t0);
249     x=x0;
250     y=y0;
251     q=q0;
252     t=t0;
253
254     do{
255         qq=q-q0;
256         wc=gain*(qd-qq);
257         if(ABS(wc)>wmax)
258         {
259             wc=(wc<0.0) ? -wmax:wmax;
260         }
261         dw=DISTANCE*wc;
262         nl=-MMPS*dw;
263         nr=MMPS*dw;
264         ms_set_v(nl,nr);
265         estimate(&x,&y,&q,&t);
266         ms_read_v(&nl,&nr,&nt);
267     }while (ABS(qd-qq)>THRESH_Q || ABS(nl)>THRESH_V || ABS(nr)>THRESH_V);
268 }
269
270 double seikika(double angle)
271 {
272     while(angle>M_PI) angle -= 2*M_PI;
273     while(angle<=-M_PI) angle += 2*M_PI;
```

```
274     return angle;
275 }
276
277 void avoid(int j)
278 {
279     if(j!=0)
280     {
281         translate(-REVERSE,VMAX,5);
282         if(j&1) rotate(-60*D2R,WMAX,5);
283         else rotate(60*D2R,WMAX,5);
284         translate(REVERSE,VMAX,5);
285     }
286 }
```

---

## 22.2 実行結果

元々センサの入力が高感度であったため、障害物の検知が不安定であった。しかし、センサの値が平均化することによってそのようなノイズが減り、障害物検知が安定した。