Motorstyring bidrag

Simon Mylius Rasmussen

25. maj 2019

Indhold

1 Baggrund

Motorstyringen for systemet vil bestå i præcis justering af forbrændingsmotorens omdrejninger i forhold til dronecopterens strømforbrug. Det valgtes at arbejde med et simplere system for at fokuserer på principperne i motorstyring. Det betød at motorstyringen skulle bestå i udvikling af PID-regulering af forbrændingsmotoren sådan at der skulle foregå korrektion af motorens spjældvinkel i forhold til motorens omdrejninger.

Fjare[1] er en artikel der beskriver PID regulering af en forbrændingsmotor som skal drive en aerocopter. Der er taget inspiration fra denne tilgang inklusiv PID-regulering og PID-koefficienter.

1.1 Kravsspecifikation

Med baggrund i Fjare[2] sættes følgende krav:

- Overshoot skal ikke være mere en 197 rpm.
- Justeringstiden må max være 8,8 sekunder.
- Ifm. et step respons skal 90 % af målet være opnået i mindre end 3 sekunder.
- Det skal være muligt at vedligeholde en konstant hastighed over længere tid (dvs. mere end 5 minutter).
- Servomotoren skal kunne reguleres.
- Motoren skal kunne startes.
- Omdrejningerne skal kunne måles.

1.2 PID regulering

PID-regulering består i regulering af et output på baggrund af forskellen mellem det tilstræbte og det faktiske - dette kan kaldes fejlen. PID regulering kan beskrives med

$$u(t) = k_P e(t) + k_I \int_0^t e(\tau)\mathrm{d}\tau + k_D \frac{\mathrm{d}e(t)}{\mathrm{d}t} \tag{1}$$

hvor $e(t)$ er fejl som funktion af tiden, $e(\tau)$ er fejl som funktion af den akkumulerede tid og k_P, k_I og k_D er koefficienter svarende til henholdsvis det proportionale, integrede og deriverede forhold. Det proportionale vil sige forholdet mellem tilstræbte og det faktiske for et givent tidspunkt. Det integrede vil sige forskellen mellem tilstræbte og det faktiske akkummuleret over tid. Det betyder at den integrerede koefficient viser en trend for fejlen. Det deriverede er et mål for hvor hurtigt fejlen ændrer sig over tid.

Formålet ved udarbejdelse af PID-regulering er at finde de optimale værdier af k_P, k_i og k_d.

Som udgangspunkt er det tidligere vist([3]) at $K_p = 0,065$ og $K_i = 0,000005$ har været optimale, samt at K_d helt blev droppet eftersom det ikke var en gavnlig parameter.

1.3 Software

Fra Fjare[4] kunne et eksempel på et program være skrevet i pseudo C-kode:

[1] Paul D. Fjare. „Feedback Speed Control of a Small Two-Stroke Internal Combustion Engine that Propels an Unmanned Aerial Vehicle“. Speciale. University of Nevada, Las Vegas, jan. 2014.
[2] Ibid.
[3] Ibid.
[4] Ibid.

```
void velPID ( ) {
  lowpassSpeed = alpha * lastLowpassSpeed + (1-alpha) * measuredSpeed;
  K1 = kp * setpointWeight * (setpointSpeed - lastSetpointSpeed) + kp *
     (lastMeasuredSpeed - lowpassSpeed);
  K2 = ki * (setpointSpeed - lowpassSpeed);
  K3 = kd * (2 * lastMeasuredSpeed - lowpassSpeed -
     lastLastMeasuredSpeed);
  output = lastOutput - K1 - K2 - K3;
  throttlePos = floor(output + 0.5);
  if(throttlePos<throttleopen){
    output = (double) throttleopen;
    throttlePos=throttleopen;
  }
  if(throttlePos>throttlesafe){
    output = ( double )throttlesafe;
    throttlePos=throttlesafe;
  }
  lastLowpassSpeed = lowpassSpeed;
  lastLastMeasuredSpeed = lastMeasuredSpeed;
  lastMeasuredSpeed = lowpassSpeed;
  lastSetpointSpeed = setpointSpeed;
  lastOutput = output;
  throttle.writeMicroseconds(throttlePos);
}
```

2 Metode og analyser

2.1 Stepinput og overføringsfunktion

PID-koefficienterne skal defineres udfra en overføringsfunktion for vores system som skal reguleres. For at udlede overføringsfunktionen blev et forsøg med optagelse af steprespons udarbejdet.

Som udgangspunkt skulle det interessante for dronecopteren være at strømmen fra batteri til dronens propelmotorer er konstant. Tilgengæld skal spjældet til forbrændingsmotoren konstant reguleres. Derfor vil en teststand som udgangspunkt have følgende:

- input: strøm fra batteri til drone
- output: regulering af spjæld via servomotor

Her vil $e(t)$ være et mål for forskellen mellem den ønskede strømstyrke og den faktiske.

Det var ikke muligt at måle strømmen fra batteri til drone. For at lave en teststand som kan tjene som demonstration af PID-regulering samt som skabelon for senere tests fremstilles istedet en teststand som skal justere spjældet i forhold til motorens omdrejninger. Dvs:

- input: motorens omdrejninger
- output: regulering af spjæld via servomotor

I dette tilfælde vil $e(t)$ være et mål for forskellen mellem det ønskede omdrejningstal og det faktiske.

For at få en grov ide om niveauet af koefficienterne kan en simulink simulation være gunstig. I denne skal gøres en række antagelser af forholdet mellem spjældvinkel og omdrejningstallet. Når koefficienterne er tunet i simulink kan der laves en test.

2.1.1 Steprespons

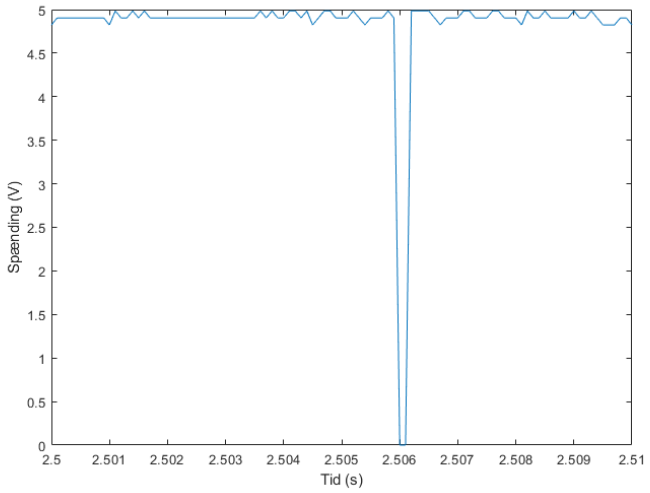
Der vil tages udgangspunkt i måling af et stepsrespons på motorens omdrejninger. Til målinger af omdrejninger anvendtes et oscilloskop som optog data fra Hall-sensoren. Optagelsen blev tricket fra KL25Z.

Data fra oscilloskopet bestod i 50000 målinger som bestod af Hall-sensorens spænding, spændingen fra servomotoren, der styrer spjældet, samt et tidsmål. Nedenfor ses et udsnit af de 50000 målinger:

second	Volt	Volt1
-1	0.215206000000000	4.98492460000000
-0.999900000000000	0.215206000000000	4.90452260000000
-0.999800000000000	0.134804000000000	4.90452260000000
-0.999700000000000	0.215206000000000	4.98492460000000
-0.999600000000000	0.215206000000000	4.90452260000000
-0.999500000000000	0.215206000000000	4.90452260000000
-0.999400000000000	0.215206000000000	4.90452260000000
-0.999300000000000	0.215206000000000	4.82412060000000
-0.999200000000000	0.215206000000000	4.90452260000000

Tabel 1:

Spændingen fra Hall-sensoren, Volt1, anvendes. Når data plottes for et kort tidsrum ses støj omkring 5 volt



Figur 1:

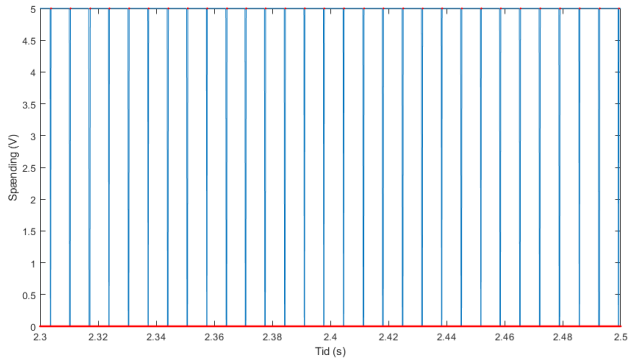
Data processeres nu med følgende Matlab-script:

```
close all
data = stepdata5;
t = data.second;
v = data.Volt1;
border1 = 4.5;
border2 = 0.1;
v2 = data.Volt;
b = 0;
```

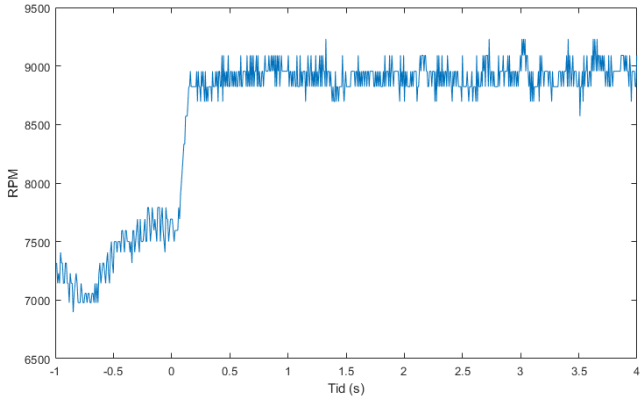
```
stop = 0;
tp = [];
tpt = [];
vpt0 = [];
i = 1;
nyv = [];
while (i <= length(t))
    if (v(i)>border1)
        vn = 5;
    else
        vn = 0;
    end
    nyv = [nyv vn];
    i = i + 1;
end
i = 1;


while (i < length(t))
    if ((nyv(i+1)>border1 & nyv(i)<border2) & b == 0)
        t1 = t(i+1);
        stop = 1;
        b = 1;
    end
    i = i + 1;
    vpt0 = [vpt0 0];
    while (stop ~= 2 & b == 1 & i < length(t))
        if (stop == 1)
            while (stop ~= 2 & i < length(t))
                if (nyv(i+1)>border1 & nyv(i)<border2)
                    t2 = t(i+1);
                    stop = 2;
                end
                i = i + 1;
                vpt0 = [vpt0 0];
            end
        end
        if (stop == 2)
            period = t2-t1;
            tpt = [tpt t2];
            vpt0(i) = nyv(i);
            tp = [tp period];
            t1 = t2;
            stop = 1;
        end
    end
end
```

I scriptet fortolkes spænding over 4,5 V som 5 V. Markering af periodegrænser ved 5 Volt kan ses her (de røde prikker øverst):

Herudover udregnes tiden for hver omdrejningsperiode og gemmes i vektoren "tp". Ved plot at tp givet ved rpm overfor tid fås:



Figur 2:



Figur 3:

2.1.2 Overføringsfunktion

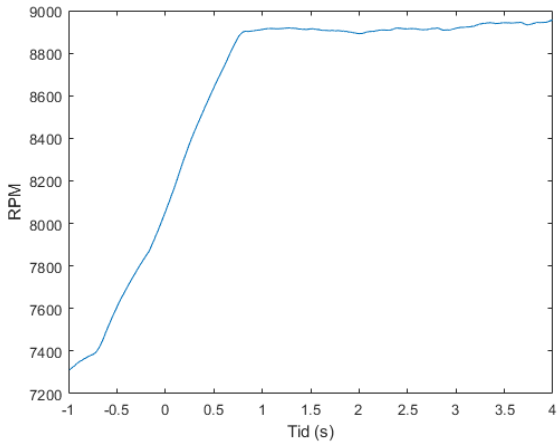
Udfra stepresponset skal følgende identificeres:

- Rise time, t_r
- Settling time, t_s
- Overshoot, M_p
- Peak time, t_p

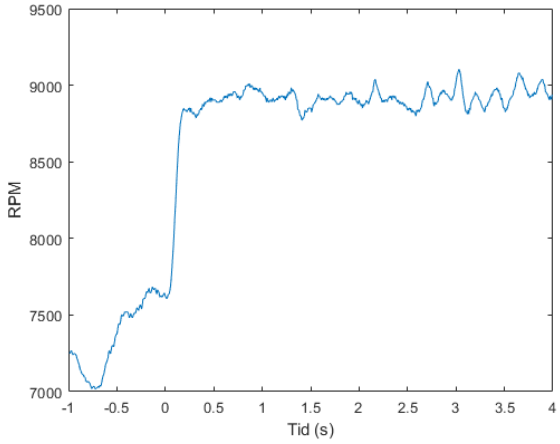
Scopet der har målt data gemmer også data op til stepresponset. Stepresponset ses ved tiden 0. Ved at applicere et midlingsfilter på 200 fås et indtryk af et steady-state niveau ved 8900, med start fra 7600, dvs 1300.

Ved at applicere et midlingsfilter på 10 fås et indtryk af en 1. grads overføringsfunktion.
Det måles at steady-state er opnået ved 0,19 sekunder. Dvs. $0.81 = 5\tau \Leftrightarrow \frac{0,19}{5} = \tau = 0,04$.
Hermed kan overføringsfunktionen sættes som

$$G(s) = \frac{1300}{0,04s + 1} \tag{2}$$

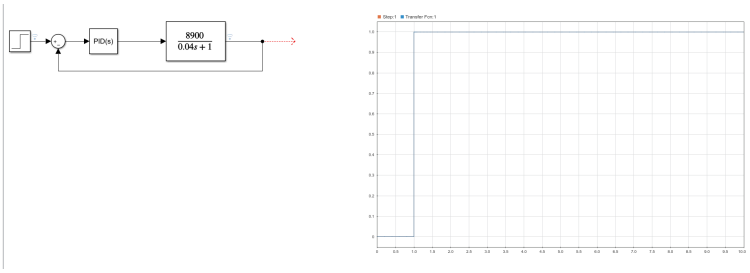


Figur 4:



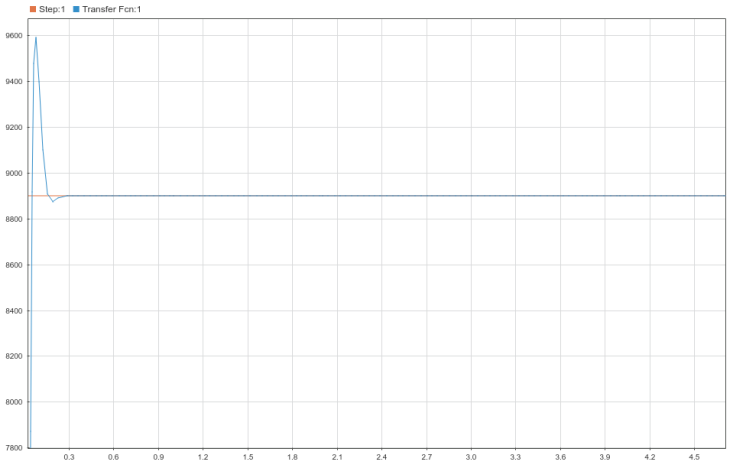
Figur 5:

I Simulink oprettes følgende:



Figur 6: Simulink - diagram

Herefter laves autotuning i PID-modulet. Der findes koefficienter svarende til $P = 0,0010$, $I = 0,0514$, $D = -1,4946$ og følgende respons:



Figur 7: Simulink - diagram 2

2.2 Software

Der udarbejdes nu et udkast til PID-kontrol software som også benytter sig af software til kontrol af servomotor samt aflæsning af motorens omdrejninger (se afsnit XXX).

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL25Z4.h"
#include "fsl_debug_console.h"
#include "rpm-detect.h"
#include "servo_driver.h"

// Filtrering
double alpha = 0.2;
```

```
double measuredSpeed = 0 ;

// Koefficienter
double kp = 0.001;
double ki = 0.0514;
double kd = -1.4946;
double K1;
double K2;
double K3;

// Vægtning af koefficienter
double setpointWeight = 0.2;
double lowpassSpeed;
double setpointSpeed;

// Diverse
double output;
double throttleopen = 1.0;
float throttlePos;

int rpmch;
int MAX_RPM = 9000;

// Initialiering
double lastSetpointSpeed = 0;
double lastMeasuredSpeed = 0;
double lastLowpassSpeed = 0;
double lastOutput = 0;
double lastLastMeasuredSpeed = 0;

void velPID (int setpointSpeed, int measuredSpeed) {
    lowpassSpeed = alpha * lastLowpassSpeed + (1-alpha ) * measuredSpeed;
    K1 = kp * setpointWeight * (setpointSpeed-lastSetpointSpeed) + kp * (
       lastMeasuredSpeed-lowpassSpeed);
    K2 = ki * (setpointSpeed-lowpassSpeed);
    K3 = kd * (2 * lastMeasuredSpeed-lowpassSpeed-lastLastMeasuredSpeed);
    output = lastOutput-K1-K2-K3;
    if (output < 0) {
      output = 0;
    }
    throttlePos = output/MAX_RPM;
    lastLowpassSpeed = lowpassSpeed;
    lastLastMeasuredSpeed = lastMeasuredSpeed;
    lastMeasuredSpeed = lowpassSpeed;
    lastSetpointSpeed = setpointSpeed;
    lastOutput = output;
    angle_throttle(throttlePos);
}

init_read_rpm()
```

```
int main(void) {
  /* Init board hardware. */
  BOARD_InitBootPins();
  BOARD_InitBootClocks();
  BOARD_InitBootPeripherals();
  /* Init FSL debug console. */
  BOARD_InitDebugConsole();
  rpmch = 7000;
  init_pwm();
  start();
  velPID(rpmch, measuredSpeed);
  return 0 ;
}
```

3 Resultater

Der blev lavet implementering af PID-koden. Resultatet var desværre at der ved øgning af mindskning af omdrejningstal blev lavet en stigning i omdrejningstal og omvendt.