PID Temperaturkontrol

September 1, 2025

1 Introduktion

Formålet med denne øvelse er at opnå kendskab til et meget anvendt kontrolsystem til at holde en given variabel konstant i et system, hvor variablen er påvirket af omskiftelige omgivelser, nemlig PID-regulering. Denne form for regulering blev oprindelig opfundet i forbindelse med regulering af mekaniske systemer såsom dampmaskiner og skibe, men kan benyttes til regulering inden for en næsten uendelig mængde områder. Vi ønsker i denne øvelse at regulere temperaturen i et mindre system, som består af en ganske almindelig elektrisk pære, der er omsluttet af en kop. I dette system kan vi variere spændingsfaldet over pæren og derved styre dens opvarmning. Temperaturen af pæren måles med et termoelement, der udnytter Seebeck-effekten til at generere en spændingsforskel over termoelementet, der er proportional med temperaturgradienten i elementet.

1.1 Opgave

Hvis følsomheden af termoelementet er 41 μV/K med en forstærkning på 100 og datakortet har en 16 bit opløsning på sin 0 − 10 V indgang, hvad er så den mindste ændring i temperaturen, vi kan forvente at måle?

1.2 Udstyr og dataopsamling

PC, dataopsamlingskort NI-6251, NI SC-2345, programmerbar strømforsyning (ISO-TECH IPS-2010), RS-232 forbindelse til PC'en), glødepære (150 W), termopar (type K), kop samt kabeler.

Reguleringen implementeres i Python - dette er dog en øvelse i PID-regulering, ikke i Python, og I får derfor udleveret skelettet til programmet, inklusiv et eksempel på en simpel regulering. Jeres opgave er udvide dette program til en fuld PID-regulering samt at eksperimentere med reguleringsparametre for at undersøge deres betydning for dens ydelse. Programmet vil være tilgængeligt på computerne, og kan også findes på Github:

`https://github.com/nanomade/Exercises10870/tree/main/PID`

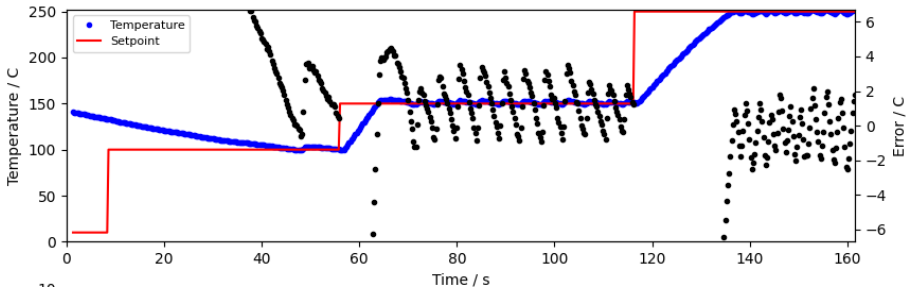

Figure 1: Eksempel på on/off regulering

2 Tænd/sluk regulering

Lad os starte med at betragte en simpel tænd/sluk-regulering, hvor den regulerbare ydelse P enten kan tændes ved maksimal ydelse $P = P_{max}$ eller slukkes $P = 0$. Reguleringen af ydelsen bruges så til at regulere en given variabel T i forhold til et forudbestemt sætpunkt T_s. Et klassisk eksempel på en tænd/sluk regulering er en ovn. De fleste ovne er udstyret med et varmeelement som kan være enten tændt eller slukket, men ikke har nogen mellemindstillinger. Tænder vi nu ovnen med et sætpunkt på eksempelvis 200°C vil varmelegmet tænde fordi ovnen (formentlig) er under 200°C. Temperaturen vil nu gradvist stige indtil den er over setpunktet hvorefter varmelegmet slukker. Når ovnen igen er under setpunktet, tænder varmelegmet igen - denne process vil fortsætte så længe setpunktet fastholdes og den faktiske temperatur vil svinge omkring 200°C med en amplitude og periode som afhænger af en lang række parametere, eksempelvis effekten af varmelegmet, tidskonstanten af systemet, hvor velisoleret ovnen er, og om der sker faseovergange i den opvarmede mad.

2.1 Opgave

Find andre eksempler på tænd/sluk regulering fra jeres hverdag.

2.2 Øvelse

I har fået udleveret et Python program som implementerer en tænd-sluk regulering. Prøv programmet og se hvordan det opfører sig. Prøv at eksperimentere med forskellige værdier af $max\_voltage$ og se hvilken indflydelse det har på reguleringen. Undgå at lade temperaturen kommer over cirka 350°C da forbindelsesledningerne til pæren begynder at få det varmere end de bryder sig om.

3 PID-regulering

Vi så i forrige afsnit, at tænd/sluk-regulering vil resultere i en svingende variabel, som funktion af tiden. Dette er måske tilfredsstillende i visse tilfælde, men

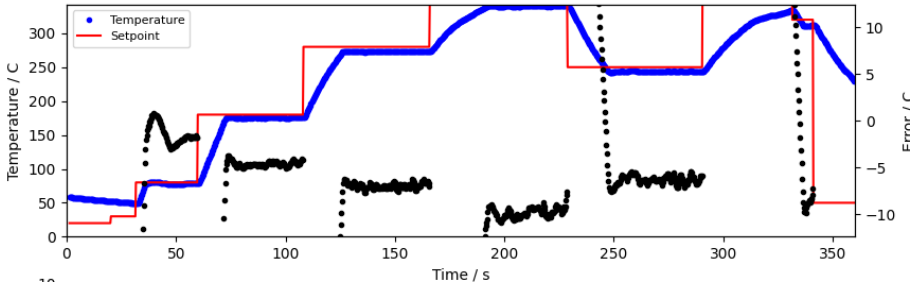

Figure 2: Eksempel på P-regulering

i andre situationer vil disse udsving være alt for store – og i øvrigt vil de pludselige ændringer mellem fuld effekt og nul effekt over tid give unødig slitage i hardwaren - forestiller man sig eksempelvis at reguleringen er af kursen på et skib, vil en regulering som udelukkende tillader fuldt ror-udslag åbenlyst ikke fungere i længden. I disse tilfælde kan vi benytte en PID-regulering, der er en algoritme som tillader trinløs regulering af den styrende parameter. Den består af tre funktioner med meget forskellig virkemåde: Proportional-, Integral, og Differential-regulering. Vi vil i det følgende gennemgå disse virkemåder i detaljer, så I kan få en fornemmelse for deres betydning i reguleringen.

3.1 Proportional-regulering

Lad os tage udgangspunkt i et tog som i henhold til køreplanen ønsker at tilbagelægge en afstand med en på forhånd fastlagt fart. Toget hastighed kan selvsagt ikke forudsiges alene ud fra motoreffekten - ydre omstændigheder så som last, vind og hældning af terrænet vil betyde at en given effekt ikke giver en entydig hastighed. For at holde køreplanen kan toget så vælge at køre noget af strækningen med fuld motoreffekt og andre dele med motoren slukket - denne metode vil godt kunne bringe toget frem til rette tid, men de pludselige accelerationer vil næppe være behagelige for passagererne (og katastrofal for salgtet af kaffe på turen). En langt bedre løsning vil være at regulere motoreffekten trinløst så toget opnår den ønskede hastighed. I første approksimation forsøger vi os med en model hvor motoreffekten, P, reguleres efter hvor langt farten er fra den ønskede værdi. Rent matematisk kan vi skrive

$$P(t) = K_P \cdot E(t) \tag{1}$$

Her er $E(t) = v_s - v(t)$ forskellen mellem sætpunktet og den aktuelle værdi, og K_P er en omregningsfaktor hvis enhed afhænger af dimensionerne for henholdsvis variabel og ydelse.

3.1.1 Øvelse

I har ikke et tog til rådighed, men brug i stedet pære-i-kop opstillingen til at afprøve en propertional regulering ved at udvide Python-programmet med en P-regulering. Teknisk gøres dette lettest ved at lave en ny klasse efter samme princip som *BangBangRegulator* hvor I nedarver *Regulator* og implementerer den nødvendige funktionalitet i form af ligning 1.

Et tip er at vælge et forholdsvist højt sætpunkt for temperaturen, fx 200°C - på den måde køler pæren nogenlunde hurtigt, og det bliver lettere at lave eksperimenter uden at vente urimeligt længe på at pæren køler mellem de enkelte forsøg.

- Find forstærkningen K_P, så temperaturen kommer tættest på sætpunktet uden at den begynder at svinge for meget omkring dette.
- Beskriv, hvordan P-reguleringen virker for forskellige værdier af K_P, prøv med lave værdier og høje værdier (fx $\frac{1}{2}$ og 2). For hvilke værdier af K_P ligner P-reguleringen en tænd/sluk-regulering?

3.2 Integral-regulering

Du så formentlig i øvelse 3.1.1, at P-reguleringen ikke rammer i setpunktet, men opnår ligevægt ved en tempetur under dette. Dette skyldes naturligvis, at med en ren P-regulering bliver effekten nul når det ønskede setpunkt er opnået. Til at afhjælpe denne situation ønsker vi en funktion, der øger effekten gradvist, så længe farten er under sætpunktet. Denne funktion kan repræsenteres ved hjælp af integralet over fejlen, da dette vil stige, hvis fejlen ikke mindskes numerisk. Tilføjes dette til ligning (1) får vi

$$P(t) = K_P E(t) + K_I \int_0^t E(t')dt' \tag{2}$$

Her er K_I en præfaktor til integralværdien, som i lighed med propertionalleddet K_P kan justeres efter ønske. Integral-reguleringen er det vigtigste led for at holde variablen ved sætpunktet for alle systemer som kræver en effekt for at holde et setpunkt (da P-leddet jo her er nul).

3.2.1 Øvelse

Udvid Python programmet med en PI-regulering. Plot og kommentér på I-reguleringens virkemåde samt kommentér på forbedringerne i forhold til P-reguleringen.

3.3 Differential-regulering

I klassisk PID-regulering findes udover P- og I-leddet også et differentialled som har til hensigt at dæmpe reguleringens respons på store ændringer. Vi tilføjer

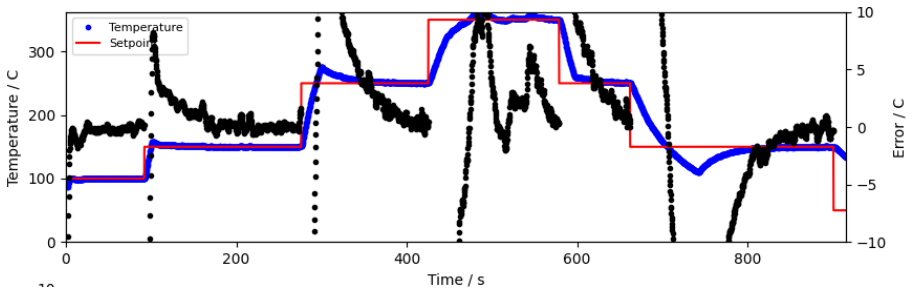

Figure 3: Eksempel på PI-regulering

dette til ligning 2 og får:

$$P(t) = K_P E(t) + K_I \int_0^t E(t')dt' + K_D \frac{d}{dt} E(t) \tag{3}$$

3.3.1 Øvelse

Udvid programmet med en PID-regulering. Plot og kommentér på reguleringens virkemåde. Ser I en forbedring i forhold til PI-reguleringen?

4 But wait; there's more

Hvis I har tid efter at have implementeret og afprøvet de tre dele af PID-reguleringen er her et par ekstra opgaver I kan kaste jer over.

4.1 Overshoot

Du har sikkert set, at temperaturen skyder godt over sætpunktet ved start. Dette er helt normalt og skyldes "hukommelsen" i integral-reguleringen – værdien af integralet stiger så længe temperaturen er under sætpunktet og begynder først at falde, når temperaturen kommer over sætpunktet. Det er typisk ikke muligt at undgå dette ved alene at stille på parametrene (K_P, K_I og K_D), med mindre man accepterer at den normale drift forringes. I mange situationer kan problemet dog reduceres dramatisk ved simpelthen at indføre i program-logikken at integralleddet kun opdateres når man er passende tæt på setpunktet - eksempelvis når bidraget fra p-ledet er mindre end det maksimale udslag

4.1.1 Øvelse

Prøv at implementere en forbedret algoritme som mindsker overshoot-problemet, eksempelvis via metoden nævnt ovenfor.

4.2 Variabelt setpunkt

Meget ofte er det ønskede setpunkt ikke en konstant værdi, men en mere kompliceret funktion. I eksemplet med toget, vil der eksempelvis kunne være områder på strækningen hvor den ønskede hastighed var lavere end andre fordi man passerer en station uden at stoppe. For temperaturreguleringer kunne der være tale om en process hvor man ønsker at udstætte sin prøve for forskellige temperaturer i forskellige tidsintervaller - helt generelt kan man vælge at se setpunktet som en funktion af tid fremfor en konstant værdi.

4.2.1 Øvelse

Implementer et variabelt setpunkt - hvordan det skal variere er op til jer selv - og afprøv om jeres PID kan følge den ønskede tidsafhængighed.